Decarbonisation and carbon neutrality by using Sustainable Aviation Fuels? A business case for Lufthansa

Laurin Schmidt

Dissertation written under the supervision of Professor Miguel Nogueira

Dissertation submitted in partial fulfilment of requirements for the MSc in Business Analytics, at the Universidade Católica Portuguesa, 03.01.2023.

Abstract

Title: Decarbonisation and carbon neutrality by using Sustainable Aviation Fuels? A business case for Lufthansa

Author: Laurin Schmidt

Sustainable Aviation Fuel is one of the key technologies to reach the goal of carbon neutrality in aviation. With the Kyoto Protocol, the Paris Agreement and the European Green Deal many countries have committed to reduce global warming below 2 degrees and taking efforts to limit global warming to 1.5 degrees compared to pre-industrial levels. Within a short timeframe until 2050, the aviation industry has committed itself to reach carbon neutrality by using different pathways and technologies. Sustainable Aviation Fuel (SAF) is mentioned as one of the key technologies to reach the goal of carbon neutrality in aviation. Because SAF is not yet commercially used and there are many uncertainties about the effectiveness and availability of this technology, this paper will evaluate the effectiveness of SAF for one of the biggest airlines in Europe – Lufthansa. To do so multiple demand growth scenarios are evaluated and used to predict the future aviation market. In addition, multiple scenarios of SAF penetration rates and fleet change scenarios are considered to assess the possible CO2 reduction with the available technologies. This paper concludes that by only relying on SAF it is not possible to reach the goal of carbon neutrality for Lufthansa. Even when including fleet renewals, the airline would only be able to reduce their total emissions by 68.14% in the best case by 2050. Lufthansa must improve and rely also on other technologies like carbon offsetting and improving operations to come closer to the goal of carbon neutrality in 2050.

Keywords: Sustainable Aviation Fuel (SAF), Lufthansa, decarbonisation, fleet renewal, aviation

Abstract Portuguese

Title: Decarbonisation and carbon neutrality by using Sustainable Aviation Fuels? A business case for Lufthansa

Author: Laurin Schmidt

Com o Protocolo de Quioto, o Acordo de Paris e o Pacto Ecológico Europeu, muitos países comprometeram-se a reduzir o aquecimento global para menos de 2 graus e a envidar esforços para limitar o aquecimento global a 1,5 graus em relação aos níveis pré-industriais. Até 2050, o sector da aviação comprometeu-se a alcançar a neutralidade carbónica através da utilização de diferentes vias e tecnologias. O SAF é mencionado como uma das tecnologias-chave para atingir o objetivo da neutralidade carbónica na aviação. Uma vez que o SAF ainda não é utilizado comercialmente e que existem muitas incertezas quanto à eficácia e disponibilidade desta tecnologia, o presente documento avaliará a eficácia do SAF para uma das maiores companhias aéreas da Europa - a Lufthansa. Para o efeito, são avaliados vários cenários de crescimento da procura, que são utilizados para prever o futuro mercado da aviação. São considerados vários cenários de taxas de penetração de SAF e cenários de mudança de frota para avaliar a possível redução de CO2 com as tecnologias disponíveis. O presente documento conclui que não é possível atingir o objetivo de neutralidade carbónica da Lufthansa apenas com base na SAF. Mesmo incluindo as renovações da frota, a companhia aérea só conseguiria reduzir as suas emissões totais em 68,14%, no melhor dos casos, até 2050. Lufthansa tem de melhorar e recorrer também a outras tecnologias, como a compensação de carbono e a melhoria das operações, para se aproximar do objetivo de neutralidade de carbono em 2050.

Keywords: Sustainable Aviation Fuel (SAF), Lufthansa, decarbonisation, fleet renewal, aviation

Table of Contents

List of figures

List of Tables

List of Abbreviations

CO2	Carbon Dioxide
ICAO	International Civil Aviation Organization
IATA	International Air Transport Association
Air Transport Action Group	ATAG
GHG	Greenhouse gas
UN	United Nations
Corsia	Carbon Offsetting and Reduction Scheme for International Aviation
LTAG	long-term global aspirational goal
SAF	Sustainable Aviation Fuel

List of Equations

$$Fuel\ Flow\ per\ Hour\ per\ Person = \frac{Fuel\ Flow\ per\ Hour}{Total\ Number\ of\ Passengers}$$

$$liter\ per\ 100\ pkm = \frac{total\ fuel\ consumption\ in\ liter}{(total\ passenger * total\ distance)}$$

1. Introduction

Man-made climate change is scientifically irrefutable. As a result, the world as we know it today is facing a significant transformation. In 2015, 196 countries committed "to limit the temperature increase to 1.5 degrees above pre-industrial levels" in the Paris Agreement. This 1.5-degree mark is crucial to limit the effects of climate change on our environment, the Intergovernmental Panel on Climate Change or IPCC of the UN has highlighted that crossing the 1.5-degree threshold would lead to for more severe climate change impacts, such as more frequent and extreme rainfall, heatwaves, and droughts. (United Nations Framework Convention on Climate Change, UNFCCC, n.d.)

Countries worldwide have joined forces to combat climate change with the Paris Agreement and the European Green Deal, with the goal of achieving climate neutrality or a 95% reduction in Carbon Dioxide (CO2) emissions compared to 1990. To reach this goal CO2 must be radically reduced in all economic and governmental sectors around the world. Given the ambitiousness of the international community to achieve high reductions in CO2 emissions, their goals can only be achieved through decarbonisation and new technologies that are not dependent on scarce resources but rely on renewable raw materials. (Wang et al., 2021)

One such industry that requires radical change to contribute to the climate change goals outlined in the Paris Agreement and the European Green Deal is aviation. Over 20% of the worldwide CO2 emissions result from the transportation sector, highlighting the importance of change in this industry. Within the sector, air travel is the most detrimental mode of transportation for the environment. (Statista, 2023b)

According to the International Civil Organization (ICAO), International Air Transport Association (IATA), and Air Transport Action Group (ATAG), 13% of CO2 emissions from all transportation sectors are caused by aviation, and aviation is accountable for 2–3% of overall greenhouse gases emitted. (Environment Branch of the International Civil Aviation Organization, 2015)

The demand for air travel has increased steadily in the past, as seen in the figure 1 below, and is projected to grow even further in the future. The only exception was in 2020 and 2021 during the COVID-19 pandemic. The COVID-19 pandemic severely impacted almost every industry and the global economy. Due to heavy restrictions for the travel sector, several airlines had to announce bankruptcy, liquidations, and cash burns during 2020 and 2021. Countrywide lockdowns were responsible for an enormous decrease in national and international air travel. (Dube et al., 2021)

These effects are clearly reflected in figure 1, demonstrating that in 2020, the annual growth in global air traffic passenger demand decreased by over 65%. However, since then, the industry has seen a quick recovery in the following years, 2021 and 2022. The IATA stated in early December 2023 that globally, the traffic in 2023 is at 98.2% of pre-COVID levels. (International Air Transport Association, 2023a) Moreover, global demand is also forecasted to grow even further. While the strong growth rates of recent post-COVID years will not be maintained, demand is projected to grow between 2.3 % and 3.3% annually until 2050. (International Air Transport Association, 2023; Air Transport Action Group, 2021)

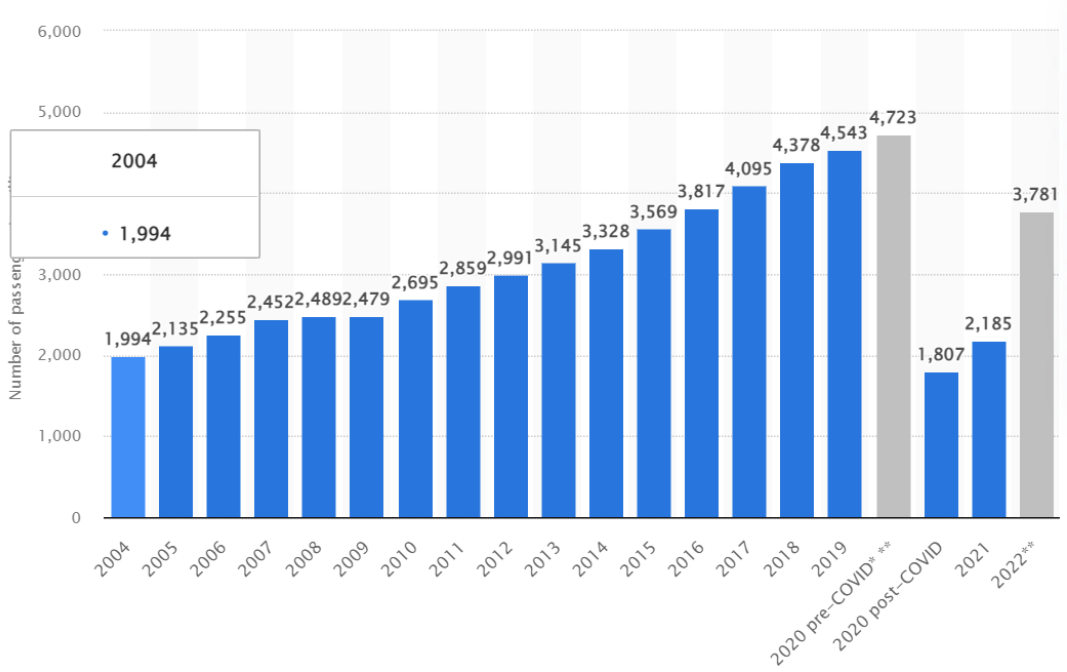

Figure 1 Number of scheduled passengers boarded by the global airline industry from 2004 to 2022(in millions) (Statista, 2023a)

To adhere to the international community's agreements to combat climate change and protect the environment while still being able to supply the continuously growing demand, it is essential to reduce the CO2 emissions of air travel by new technologies and improved processes.

Several studies indicate that Greenhouse Gas (GHG) emissions must peak before 2025 and be reduced by almost 50% by 2030 to limit global warming to 1.5 Degrees Celsius, which highlights the urgency for innovative transformation necessary in just the next few years. As the Intergovernmental Panel on Climate Change notes; "without immediate and deep emissions reductions across all sectors, it will be impossible." (Intergovernmental Panel on Climate Change, 2022, p. 2)

The International Civil Aviation Organization (ICAO), as a specialized agency of the United Nations (UN), introduced a Carbon Offsetting and Reduction Scheme for International Aviation (Corsia). These are measures to reduce emissions from international aviation, minimizing market distortion while respecting the specific circumstances, and respective capabilities of ICAO member states. However, focusing only on reducing the total emission of CO2 into the atmosphere by relying on increasingly efficient future technologies is not enough. Particularly because the ICAO has committed itself to a long-term global aspirational goal (LTAG) to achieve net-zero carbon emissions by 2050 in the aviation sector. Thus, to reach this goal, the organization outlined five different roadmaps: carbon removal, sustainable aviation fuels, hydrogen, improving operations and improving efficiencies. Studies concluded that improving efficiency and sustainable aviation fuels (SAF) will have the most significant impact on reducing carbon emissions to net zero by 2050. (International Civil Aviation Organization, 2022a).

1.2 Project objectives

This thesis aims to assess the steps that must be taken by airlines to effectively meet the climate goals of the ICAO and the Paris Agreement to stay within the 1.5-degree goal. To do so, the following sections will investigate how sustainable aviation fuel and renewals of aircraft fleets with newer, more efficient jets to combat CO2 emissions will affect the total amount of CO2 produced. To obtain the most comprehensive and up-to-date results possible, this comparison will analyse Europe's biggest airline, Lufthansa. Lufthansa operates locally and globally with hubs in Germany. So, it is possible to collect and analyse the effect of short, middle, and long-haul flights and the results of the different measures to reduce CO2 emissions.

To stay within the scope of work, the analysis focuses on Lufthansa Airlines. It excludes subsidiary airlines like Swiss, Eurowings, Austrian Airlines and Brussels Airlines. The main body of this thesis will analyse if the change from fossil fuels to SAF is realistic and how much impact it has. Therefore, different flight demand growth scenarios and SAF penetration rates are considered to analyse and compare multiple possible future cases. In addition, several scenarios for the renewal of Lufthansa's aircraft fleet will be considered to assess the impact on CO2 emissions. To conclude this work, an evaluation of the current strategies and measurements developed by Lufthansa and the ICOA against the analysis of this paper will be conducted to develop an action plan to improve the impact of their current strategies.
This thesis is structured as followed; the next section will outline and review relevant literature to set the scene for the statistical analysis, provide necessary context and demonstrate the importance and urgency of the topics investigated here.

The review is followed by a description of the methodology employed within the statistical analysis, paying particular attention to the data collection, cleansing and transformation process that preceded the main analysis. The first section of the core statistical analysis will explore the different flight demand growth scenarios to understand how they will impact the total level of CO2 emission of aviation without any countermeasures and determine which flight distances (i.e., long haul vs. short haul) account for the most CO2 emissions. This is followed by analyses that will investigate the impact of fleet changes and SAF introductions on each of the flight demand growth scenarios as countermeasures to combat CO2 emissions.

The introduction of both of these countermeasures will be analysed through different scenarios to demonstrate how different modes and rates of change in the aviation industry will affect CO2 emissions. Finally, this thesis will conclude by evaluating the statistical results against the current ICOA action plan to determine how well this plan is suited to achieve the ambitious goal of net-zero carbon emissions by 2050.

2. Review of relevant literature

The world's climate is mainly dependent on the chemical composition of the atmospheric layer, which has dramatically changed since the Industrial Revolution. The atmosphere is comprised of three primary gases: Nitrogen, Oxygen and Argon, but trace gases like Carbon Dioxide, Methane, Nitrous Oxide, and Ozone have the most significant impact on the world's climate. These gas molecules, called greenhouse gases, captivate thermal infrared radiation and heat. (Gabric, 2023)

The uncontrolled discharge of greenhouse gases into the atmosphere by burning fossil fuels, farming livestock, and cutting down forests increasingly influence the climate and the earth's temperature. (Calvin et al., 2023; Lashof & Ahuja, 1990 ; Mohajan, 2011 ; Intergovernmental Panel on Climate Change, 2023)

By 2020, the atmospheric concentration of carbon dioxide has increased to 48% above its pre-industrial level. (Buis, 2022 ; Chen et al., 2019 ; European Commission, n.d.)

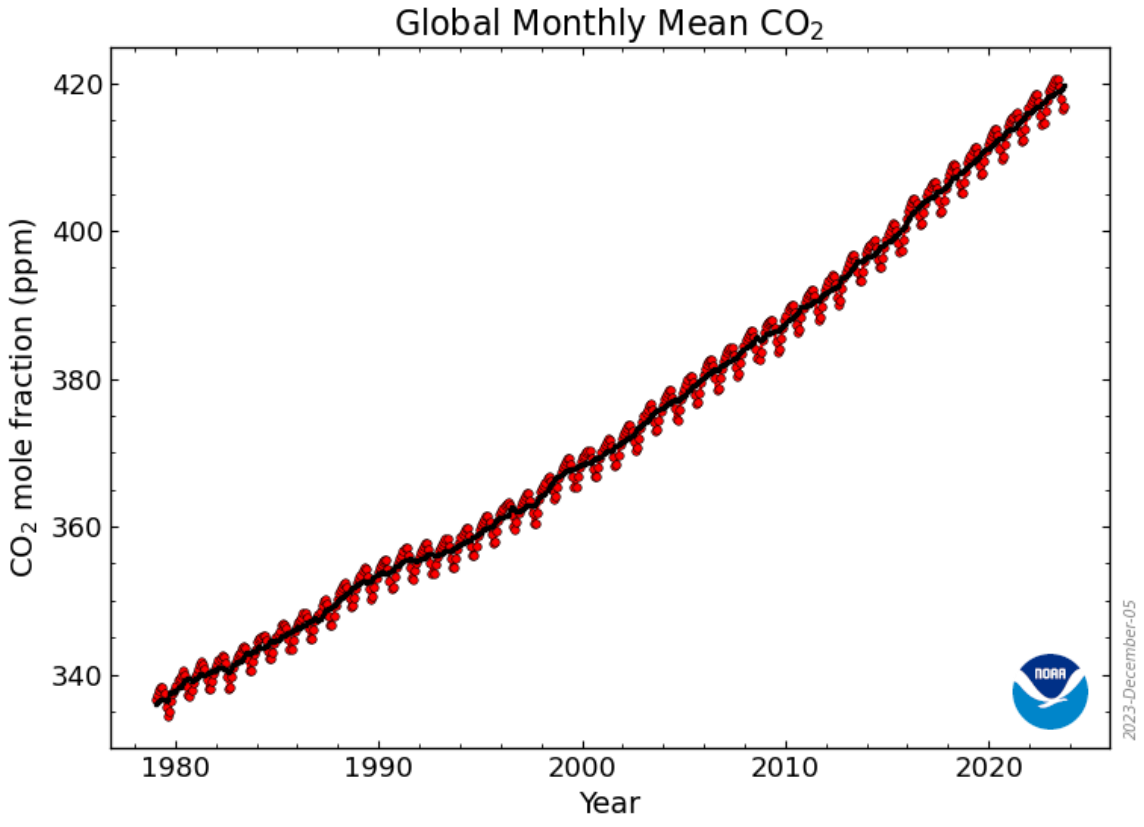

Figure 2 Atmospheric Co2 in ppm at Mauna Loa Observatory

(Global Monitoring Laboratory, 2023)

These changes in our atmosphere's composition severely affect all aspects of life. The effects are multidimensional, with possible impacts of global warming and climate change being increasingly extreme weather events and slower onset environmental change processes. Extreme weather includes flooding, wildfires, droughts, heat waves, extreme rainfall, and storms. Slow onset processes with a longer timeline are the already noticeable rise in sea levels, loss of biodiversity, reduction of permafrost, desertification and salinisation, glacial retreat and ocean acidification. All these effects pose severe risks to health, livelihoods, food security, water supply, human security, and economic growth. (Global Programme on Risk Assessment and Management for Adaptation to Climate Change (Loss and Damage), 2021) The dependency on fossil fuels on the global society and economy has increased these effects, and climate change has now been labelled as a state of emergency. (United Nations, 2022)

Climate change impacts are projected to slow down economic growth, diminish the effects of poverty reduction measures, further erode food security, and prolong existing and create new poverty traps. (IPCC 2014). As mentioned in the introduction, the transportation industry accounts for over 20% of global CO2 emissions, making it the second most environmentally detrimental sector in terms of CO2 emissions after the energy sector. (Ritchie, 2023) This implies that there is a need for drastic change that must have a significant impact on decreasing the global emissions of the transportation industry. A particular focus of any emission reduction efforts within transportation should be on individual transportation by car, road freight and aviation, as they are the primary contributors to greenhouse gas emissions from the transport sector. (International Energy Agency, n.d.)

The main reason for these modes of transportation having the highest carbon emission rates is the combustion of fossil fuels. To evaluate the specific emissions caused by fuels, conversion factors for different fuel types have been invented. They are widely used in all industries. (US EPA, 2023) For example, to calculate the CO2 emissions of 1 kilogram of jet fuel (Jet A / Jet A-1), the IATA has committed to an emission factor of 3.16. For every kilogram of jet fuel, 3.16 kg of CO2 is emitted into the atmosphere. (International Air Transport Association, n.d.)

The expected demand growth rates within aviation and the possible impacts of a further increase in atmospheric CO2 levels underline the urgency for change to happen now. Therefore, the ICAO developed different action plans to tackle the issue of CO2 emission in aviation to reach the goal of becoming net zero and complying with the Paris Agreement. The action plans or long-term aspiration goals are summarised into five categories: carbon removal, sustainable aviation fuels, Hydrogen as a jet fuel alternative, improving operations and improving efficiencies. (International Civil Aviation Organization, 2022)

Sustainable aviation fuel (SAF) is an alternative to conventional fuel. It can be produced from either biological or non-biological sources that are nondepletable. SAF can originate from various different sources, for example, feedstock waste fats, oils and greases, municipal solid waste, agricultural and forestry residues, wet wastes, and non-food crops cultivated on marginal land. SAF can also be produced synthetically via a process that captures carbon directly from the air. SAF can be considered carbon neutral because it recycles the CO2 absorbed by the biomass used in the feedstock during its life. This means sustainable aviation fuel will still emit CO2 in the combustion process. However, during its production, CO2 is deprived of the atmosphere, which makes it carbon neutral. (International Air Transport Association, 2023b)

Bio-renewable SAF fuels use transesterification or hydrotreating of plant-based lipids, fats, oils, and greases. The American Society for Testing and Materials (ASTM) has approved multiple production ways for Bio-renewable and synthetic fuels for commercial use as “drop-in” replacements for commercial jet fuels, meaning they must comply with the same criteria and requirements that conventional jet fuels have to adhere to.

For international aviation, regulatory organisations like the ICAO, Federal Aviation Administration (FAA) and the European Authority for aviation safety (EASA) have acknowledged the drop-in method, which requires no further certification once it is designated as a drop-in fuel. (Undavalli et al., 2023) To be eligible for use, this fuel must be certified under the ASTM jet fuel standards. SAF can meet the same quality levels as conventional Jet A-1 fuel. This means it can be used without modification on the aircraft, and it is possible to use this with a 100% unblended penetration rate. (Ng et al., 2021 ; Airbus, 2023a)

There are multiple certified methods to produce SAF today. One of the most common for commercial use is the Hydrotreated Esters and Fatty Acids (HEFA) method, whereby vegetable oils, waste oils, or fats are refined into SAF. This is the primary production process for bio-renewable fuels. (Airbus, 2023a; SkyNRG, 2023 ; Tanzil et al., 2021)
Alternative drop-in fuels are the most promising technologies for decarbonisation in aviation because they do not require any modification. While other technologies to reduce carbon emissions require changes to aircrafts, engines, and the refuelling infrastructure, which significantly impacts the cost and timeline to implement these technologies, SAF can be introduced to existing planes and infrastructures, even extending their usability, bringing down the lifecycle costs for the aircraft and refuelling infrastructure. Today, SAF is produced in low volumes, however, production costs are significantly higher compared to conventional aviation fuel. (Bauen et al., 2020)

SAF is proven to reduce the carbon footprint and emit fewer greenhouse gases than conventional aviation fuels. Certified methods allow a blend-in rate of up to 50% in the past, which means it must be mixed with conventional jet fuel. This decreases the CO2 emissions by up to 40%. (Undavalli et al., 2023)

Current advancements in the development of aviation technology enable total usage of SAF, meaning that no blend-in with conventional jet fuel is required to power the latest generation of aircrafts. Airbus successfully performed multiple tests to use 100% SAF exclusively, the latest test in March 2023 with the Airbus A321Neo, widely used in the Lufthansa fleet. Today, all commercial aircrafts from Airbus can operate with a SAF blend of up to 50%. (European Authority for aviation safety., n.d.-b.; (Airbus, 2023b)

The assessment of the ICAOs action plans to combat CO2 emissions has concluded that drop-in fuels have the most significant impact on the reduction of CO2 by 2050. Because of the low feasibility and commercial viability, other roadmaps outlined in their plan, particularly Hydrogen is not expected to impact CO2 reduction by 2050 significantly, but it’s impact may increase after 2050. (International Civil Aviation Organization, 2022)

While SAF has been used since 2011 in more than 350,000 commercial flights, according to the Air Transport Action Group, the current production is still less than 1% of the global jet

fuel demand. The amount has increased in the past and subsequent years and is projected to cover 2% of conventional jet fuel by 2025. (Levingston, 2021)

In contrast, improving efficiencies through technical advancements in aircrafts and engines as well as the introduction of SAF has clear short-term potential, primarily because of fuel consumption reduction and the ability to use existing infrastructure. New technical advancements for Hydrogen and electric aircrafts would exhibit worse energy efficiency considering the life cycle emissions and would need a new infrastructure. In addition, operational improvements have opportunities to reduce CO2 emissions by improving flight and ground performance. (International Civil Aviation Organization, 2022)

Carbon removal or offsetting will also be a big part of the strategy and measures to achieve net zero emissions by 2050. The goal of net zero allows the emission of carbon dioxide and other greenhouse gases into the atmosphere as long as the equivalent amount is removed. This is not solving the issue of emitting CO2 and greenhouse gases into the atmosphere but is used widely across all industries around the globe to comply with the set goals of the Paris Agreement. Especially in carbon-intense industries, there is mostly no immediate technology change possible. Therefore, the goal has to be reached with other options. (The Economist, 2020) Other industries can remove carbon from the atmosphere, like reforestation and wetland and peatland restoration, which are nature-based removals. In addition, there is the option for technology-based removals, for example, biochar and bio-oil, bioenergy with carbon capture and storage and direct ocean capture. However, today, there is no possibility to remove enough CO2 from the atmosphere as it is currently emitted. Therefore, only some companies can use this option to operate generally as before. (Mannion et al., 2023)

With the great potential that SAF promises, the aviation industry is currently focussing on technological progress and these alternative fuels to achieve the highest possible reduction in CO2 emission for the industry. The renewal of existing fleets and elimination of old and inefficient aircraft are measures that can be started immediately and are already taking place at some airlines, such as Lufthansa. With continued innovation, new generations that will follow in the coming years and decades will likely increase the degree to which they can rely on SAF even further, making 100% SAF usage not just experimentally but commercially feasible and a net-zero emission aviation industry possible. The main factors to consider here

are production volume and availability. However, this will be able to cover around 20 % of total demand by 2035 and over 60 % by 2050. (Air Transport Action Group, 2021)

The biggest limitation for Sustainable Aviation Fuel today is the production capacity and penetration rate compared to conventional jet fuel. Today SAF makes only 0.05% of the European fuel use. (European Authority for aviation safety, n.d.) With dedicated projects and partnerships, airlines try to accelerate the percentage and their availability. Lufthansa already used 13,000 tons of SAF in 2022. This corresponds to 0.2% of Lufthansa's total fuel demand and 5% of the SAF available worldwide, making Lufthansa one of the five largest buyers worldwide. With multiple strategic partnerships, the Airline tries to have an advantage in availability and divaricates by investing in multiple techniques like Power-to-Liquid, Sun-to-Liquid and Biofuel options (Lufthansa, n.d.-c) From 2024 to 2030, together with OMV, Lufthansa has signed a memorandum of Understanding to supply 800,000 tons of Sustainable Aviation Fuel. (Lufthansa, 2022)

As previously explained, there are several approaches and proposed technologies to tackle the net zero goal. The following part will describe the project structure and methodology and the statistical methods used to identify and analyse the reduction potentials of fleet renewal and Sustainable Aviation Fuel in multiple scenarios.

3. Project methodology and limitations

3.1 Overview

The analysis will be split into three parts. In the first part, Lufthansa's historical and current flight and fleet statistics will be determined through the use of descriptive statistics, which is based on several open-source datasets that were retrieved and merged for this purpose. Subsequently, in the second part, several market growth-, fleet change and SAF reduction scenarios are built and justified by data from industry and official authorities. By comparing the yearly and total emissions, it is possible to assess the feasibility and reachability of the emissions goals using these measurements. The final part is the evaluation of the sustainability.

3.2 Project methodology

This research aims to forecast Lufthansa's future emissions under different circumstances and future technologies. The methodology used in this study is oriented on other action research studies that aim to solve a problem based on action plans or recommendations. (Igwenagu, 2016)

This includes, at first, the data collection. In this step, the needed data is collected from several sources. Within the collection, a first assessment of the data is made to ensure reliability and correctness by cleansing and transforming the data sets to suit the intended use. Activities included cleaning not-needed columns and outliers, null values, computing fuel consumption and emissions per flight and transforming measuring units. In addition, viable sources of this topic and the already existing forecasts and scenarios are investigated to determine if they can be used as a research base for this case.

Consequently, descriptive data analysis is performed to get basic information and assess the historical and current situation based on the transformed and cleaned data, followed by prescriptive data analysis to investigate future development, and apply the forecasts and predictions based on selected scenarios. The selected scenarios of demand growth and SAF penetration were chosen based on previous research and use by authorities and the aviation industry for reporting. The fleet change scenarios are based on the published information from Lufthansa. The scenarios will be employed to assess whether the given forecasts and scenarios also apply in this case and, in fact, can provide the reduction rates proposed by the ICAO and ATAG.

The last step is a detailed description of the results and an assessment of Lufthansa's progress to comply with the given goals of the ICAO to reduce CO2 emissions and reach carbon neutrality in 2050 drastically.

3.3 Limitations of Data Sources

The effectiveness of this study depends on having the most appropriate data available, thus requiring the use of multiple independent sources. The needed data is sensitive and not easily accessible to the public. Many providers use paywalls or subscriptions, and Lufthansa will not make their data available for private and research purposes. In addition, the scope of this work is used to determine and evaluate possible scenarios for the future. It is only possible to access historical data and use forecasts to predict the outcomes. The available forecasts use different approaches and do not fully disclose their forecasting methodology and process. To handle this lack of transparency, this analysis relies on official sources for historical data and different future scenarios. Building the scenarios themselves would go beyond the scope of this thesis.

Moreover, this paper investigates sustainable aviation fuel, which is not commonly used and is only certified for commercial use to blend with conventional jet fuel. Airlines and manufacturers are still testing the use of SAF. Therefore, the foundation of research and data needs to be applied in real-world scenarios.

3.4 Data Collection

Data was collected from several sources to answer the proposed research questions. There is no one dataset publicly available that includes all the needed data points. Different official sources, including Eurocontrol, IATA, ICAO, OAG Aviation, RDC Aviation, destatis and Lufthansa, have been contacted to access first-hand data without success.

3.4.1 Openflights

As publicly available data provides the foundation for this thesis, Openflights, provided a broad dataset about historic routes and airports worldwide. Their data is based on the Digital Aeronautical Flight Information File, the official dataset provided by the National Geospatial-Intelligence Agency of the United States of America and other third-party databases. This data is checked periodically by Openflights and its users to ensure data quality. This specific database is used because it provides data from reliable sources and combines multiple provides.

Two datasets were of particular interest for the research presented here. The routes dataset comprises 67665 routes and gives information about the airline, source, and destination airport, stops and the aircraft tape used on these routes. After filtering for the Lufthansa routes, the dataset measures 923 routes. Secondly, the airports' dataset comprises 7698 airports with information about city, country, ICAO and AITA codes, latitude, longitude, altitude, and time zones.

3.4.2 Opensky

Additionally, Opensky provides an SQL query tool for their historical database to retrieve flight data, accessed via Impala shell over SSH with a putty client. Opensky is a non-profit organization based in Switzerland that provides reliable and accurate flight data to the public. For this thesis, over 2.000.000 flights were recorded from Lufthansa from 2016 to 30.09.2023, where the last query was made.

The retrieved dataset has 2.710.467 flights and provides information about the ICOA24, time and date of departure and arrival, duration, callsign of this flight and source and departure

airport. Moreover, the aircraft database from Opensky is publicly available for download in October 2023. This includes 579491 aircrafts including information about the ICAO24, registration, manufacturer, aircraft model and type, serial number, operator, callsigns and categories.

3.4.3 Star Alliance and flight radar

The fleet data was retrieved from Star Alliance's official website and flight radar. This contains accurate data on the Lufthansa airline's historical and current fleet and the aircraft's delivery and exit dates. This dataset consists of 379 aircrafts with information about the registration, name, current location, total flown hours, ICAO code, airline, passenger capacities, cargo capacity, range, fuel capacity and fuel flow per hour, service ceiling, cruising speed and operational status. All this data was checked against press releases and reports from the Lufthansa Group to ensure accuracy.

Taken together, the data collected from all these sources is the basis for answering whether sustainable aviation fuel and aircraft fleet change can help Lufthansa reach the goal of carbon neutrality by 2050.

3.5 Data operations

After retrieving all datasets, they are merged into one with the needed data points using shared columns such as different IDs. This created a total dataset of 2051769 rows and 27 columns. By collating with other sources, the retrieved data was not representative for the months Juli, August, and September 2023. Therefore, they have been excluded.

To assess the entire year for 2023, the data was forecasted based on the trends and seasonality of the previous years except the COVID–19 years 2020 and 2021, as they are not representative of future demand behaviour. To forecast the demand for 2023, a time series analysis was used to forecast the emissions and total number of flights by employing the Seasonal Auto-Regressive Integrated Moving Average (SARIMA) model to identify historical patterns, trends and seasonality.

This forecast is used to determine the total flights and emissions. Specific forecasts for routes and used aircraft types are not considered.

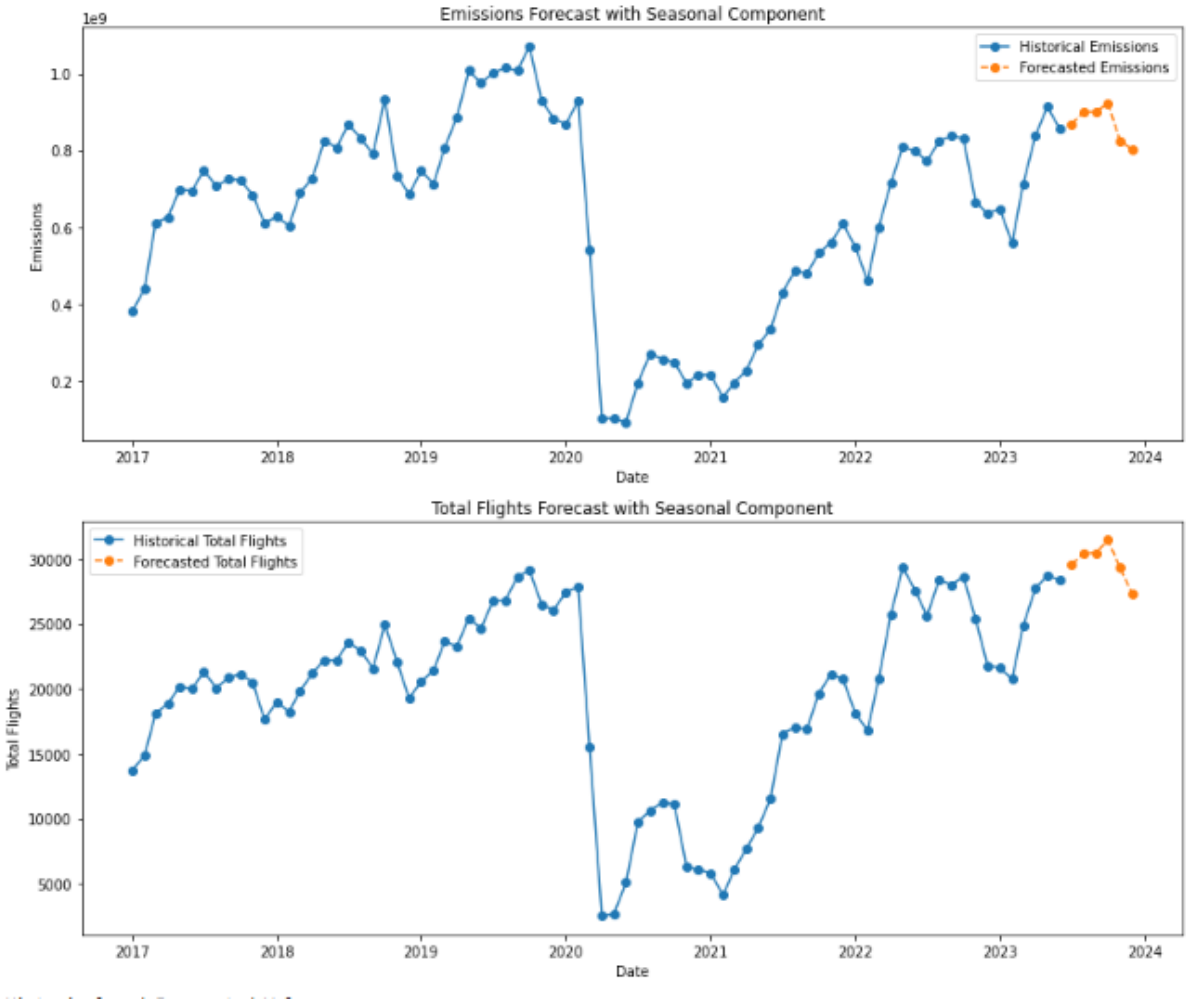

Figure 3 Flights and emissions forecast for 2023

To assess the flown distance, geodesic from the geopy library calculates the difference between latitude and longitude. By converting different values, e.g. the duration from minutes to hours, and creating new columns by calculating the fuel consumption per flight, it was possible to calculate the emissions per flight. The emissions are calculated by using a constant of 3.16. This constant represents the number of kilos of CO2 produced by burning a kilo of aviation fuel. It was provided by the ICAO in 2018, which is widely used in the industry as well by Lufthansa. (International Civil Aviation Organization, 2018 ; Lufthansa, 2023d, p. 11)

While inspecting the dataset, multiple flights in the available data were operated by different airlines, and information was missing. Therefore, further cleaning through transformation, removing non-random null-values and outliers and inspecting relevant connections among the

data was necessary. The final dataset has 1.528.191 rows and 23 columns. Compared to the published information for the number of flights and Emissions, the dataset is not complete but represents between 68% and 72% per year of the official flights reported by Lufthansa. The dataset can therefore be regarded as a large representative study.

ICAO24	unique 24-bit identifier of the aircraft concerned. It is assigned by the International Civil Aviation Organisation
registration	code unique to a single aircraft
aircraft_model	Model and maker of the Aircraft
type_code	alphanumeric code for type of aircraft model
Total	Total number of passenger seats available
Fuel_flow_hour	Fuel flow per hour in liter
firstseen	Date of departure
lastseen	Date of arrival
Src_airport_icao	Departure airport
Dest_airport_icao	destination airport
Name_src	Name of departure airport
Name_dest	Name of destination airport
distance	Distance between departure and destination airport
Duration_hours	Duration of flight in hours
delivered	Date of aircraft delivery to the fleet
age	Age of aircraft
retired	Retired from fleet
Fuel_consumption	Fuel consumption of flight (fuel flow hour* duration hours)
Fuel_consumption_total	Fuel consumption of aircraft (fuel flow hour * total hours)
Emissions_hour	Emissions per hour (fuel flow hour * 3.16)
Emissions_flight	Emissions per flight (fuel flow hour * duration hours)
Total_emissions	Total emissions of aircraft (fuel flow hours * total hours)
Cumulative_duration_hours	Total duration of specific aircarft in hours

Table 1 Explanation of Column names in dataset

3.6 Exploratory Data analysis

The first step is to group the relevant data points to obtain an overview and first insights into the available data.
By analysing the total number of flights, it is recognizable that the number of flights has significantly dropped in 2020, which can be justified by the COVID-19 pandemic and the following restrictions. From 2020, an increase in the following years of 14.99 % in 2021, 89.25 % in 2022 and 11.64 % in 2023 compared to the previous year is observable and can be explained by easing COVID restrictions.
Lufthansa announced an increase of demand of 13% in the first nine months of 2023 compared to 2022. (Lufthansa, 2023b)

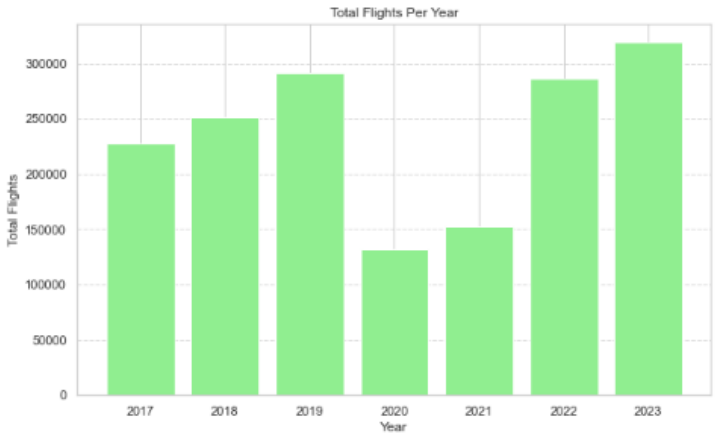

Figure 4 Total Flights per Year

Table 2 provides information about the percentage of in-country flights in the dataset. It is detectable that the percentage of flights within Germany decreased in 2020 when COVID-19 hit and slowly recovered in 2022. The table in Figure 5 shows the percentage of short-haul, medium-haul and long-haul flights of total flights and their percentage of emissions. Eurocontrol defines flights with a distance between 1500-4000km as medium-haul flights. Below 1500km is a short-haul flight, and above 4000km is a long-haul flight. (*EUROCONTROL Data Snapshot #18*, 2021)

This supports the assumption that long-haul flights are responsible for the most emissions in air travel. We can see that over 70% of the emissions are caused by long-haul flights. Since Covid, we can also see that the percentage of long-haul flights has dropped significantly.

Year	Letter	Percentage of Routes	Percentage of Emissions
2017	ED	4.75%	5.44%
2018	ED	4.67%	4.82%
2019	ED	5.66%	4.81%
2020	ED	2.58%	5.66%
2021	ED	2.37%	4.50%
2022	ED	4.65%	4.58%
2023	ED	2.50%	4.57%

Table 2 Percentage of flights in Germany

Year	Short Haul (<1500 km) Flights	Medium Haul (1500-4000 km) Flights	Long Haul (>4000 km) Flights	Short Haul (<1500 km) Emissions	Medium Haul (1500-4000 km) Emissions	Long Haul (>4000 km) Emissions
2017	85.27%	4.96%	9.72%	20.41%	3.21%	75.91%
2018	78.08%	9.11%	10.37%	18.97%	5.73%	72.77%
2019	77.28%	8.09%	10.75%	18.25%	4.93%	72.40%
2020	79.36%	8.99%	8.78%	21.89%	6.60%	68.67%
2021	75.12%	12.39%	9.69%	22.05%	9.80%	66.36%
2022	79.20%	8.53%	9.03%	23.03%	6.87%	67.81%
2023	79.89%	7.73%	9.37%	21.89%	5.88%	70.20%

Table 3 Percentage of short, middle and longhaul flights and their emissions

In addition, as seen in figure 5 and table 4, the total emissions per year are declining more than the total number of flights, which the change in short and long-haul flights can explain. The slight increase in 2023 is justifiable by the reactivation of the older A380 models to meet the increased demand of long-haul flights. (Caswell, 2023 ; Lufthansa, 2023a, p. 12)

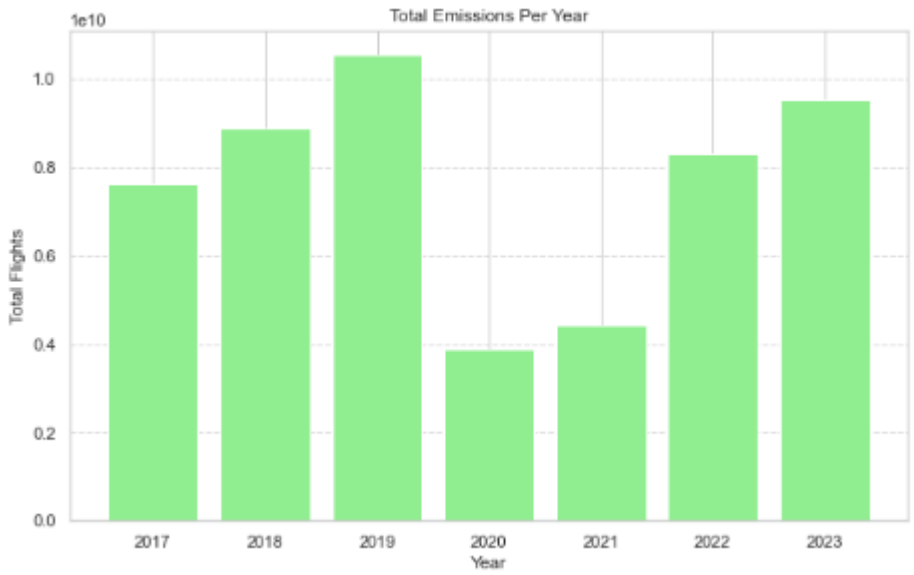

Figure 5 Total Emissions per Year

	year	Emissions	Emissions Growth Rate	Total Flights	Flights Growth Rate
0	2017.00	7648914406.60	NaN	227258.00	NaN
1	2018.00	9127975468.07	19.34	257059.00	13.11
2	2019.00	11047868732.61	21.03	302945.00	17.85
3	2020.00	4011164506.01	-63.69	136116.00	-55.07
4	2021.00	4514907222.75	12.56	156515.00	14.99
5	2022.00	8498031754.22	88.22	296203.00	89.25
6	2023.00	9749450205.82	14.73	330669.13	11.64

Table 4 Growth rates for flights and emissions

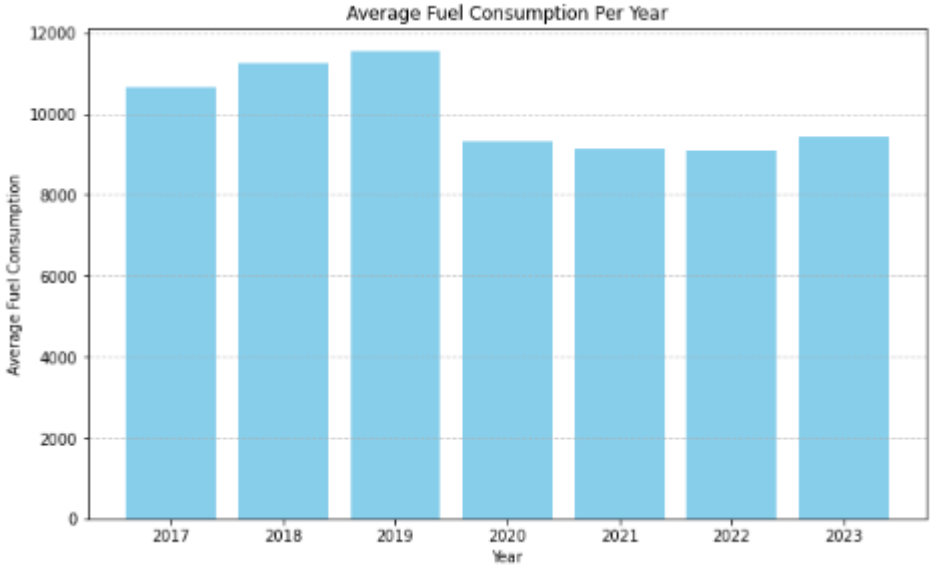

Figure 6 Average fuel consumption per year per aircraft

The average fuel consumption per aircraft dropped significantly in 2020 due to the retirement of 35 aircraft in that year, more than ever before. This graph also shows that even in the following years and with many flights, the average fuel consumption and, therefore, the average CO2 emissions grew at a different rate. This can be explained by retiring older, less efficient aircraft and using newer, more efficient models. This observation is also shared by Lufthansa. “Since the start of the coronavirus crisis, 110 aircraft have been retired. In the same period, i.e. since the end of 2019, 57 aircraft have been added.” (Lufthansa, 2023c, p. 26)

The fleet's average age decreased from 15.3 years in 2016 to 12.05 years in 2023.
The increase in total and long-haul flights can explain the increase in average fuel consumption. The renewed interest in long-haul flights led Lufthansa to reactivate the A380

fleet, which has a high fuel consumption due to its size and possible distance. (Caswell, 2023 ; Lufthansa, 2023a, p. 12)

Lufthansa sees an efficiency gain between the transport performance and the fuel consumption. This is called the decoupling effect: “Decoupling occurs when the growth rate of an environmental pressure is less than that of its economic driving force (e.g. GDP) over a given period.” (Organization for Economic Cooperation and Development (OECD), n.d., p. 1)

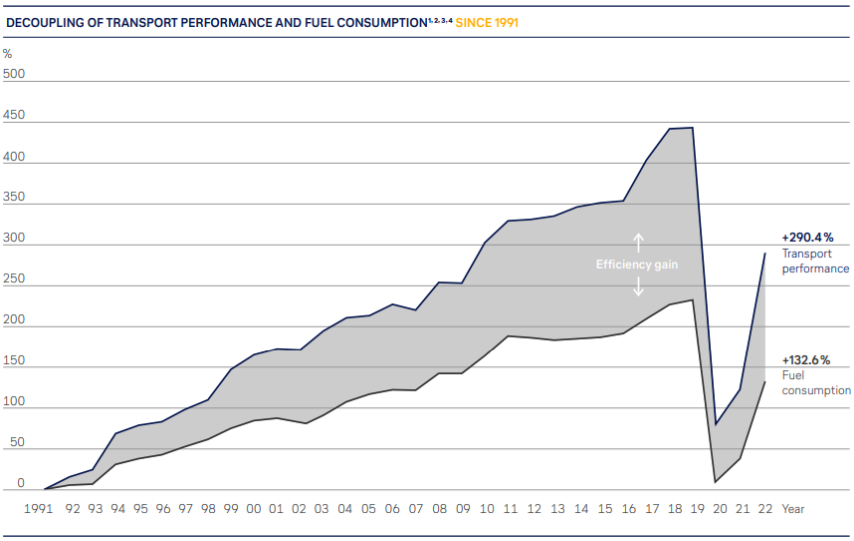

Figure 7 Decoupling of transport performance and fuel consuption since 1991 (Lufthansa, 2023)

To further investigate this, the next step will analyse the differences of specific aircraft types to assess their impact on fuel consumption, emissions, and percentage of total flights. Therefore, we distinguish the fleet into short-, medium-haul-, and long-haul aircrafts (Figure 6). (Lufthansa, n.d.-b)

ICAO Type code	Aircraft Model	Distance
A319	Airbus 319-100	short
A320	Airbus 320-200	short
A20N	Airbus A320neo	short
A321	Airbus 321-100/200	short
A21N	Airbus A321neo	short
A333	Airbus A330-300	long
A343	Airbus A340-300	long
A346	Airbus A340-600	long
A359	Airbus A350-900	long
A388	Airbus A380-800	long
B744	Boeing 747-400	long
B748	Boeing 747-8	long
B789	Boeing 787 Dreamliner	long
CRJ9	Bombardier CRJ900	short
E190	Embraer 190	short

Table 5 Short and longhaul category of Type codes

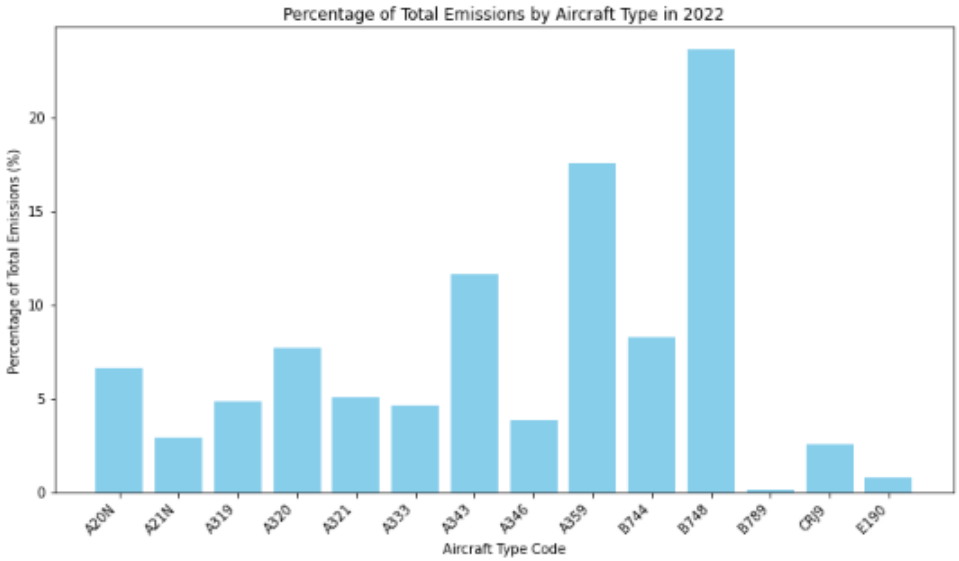

Figure 8 Percentage of total emissions by Aircraft type in 2022

The Barplot in figure 8 shows the percentage of emissions in 2022 by aircraft type. We can compare it to the percentage of total flights and distance to further investigate. Over 23% of the total emissions in 2022 have been emitted by the long-haul aircraft Boing 747-800, despite the aircraft only accounting for 2.35 % of the total flights in 2022 and covering 13% of the total distance. All long-haul aircrafts cause more CO2 emissions than typical short- and medium-haul aircrafts. A good example would be that A346 are responsible for 3.83% of emissions and flew 2.35% of the total distance. In comparison, the A20N is responsible for 6.64% of the total emissions, accounting for 10.67% of the total distance (Table 6).

Type Code	Percentage of Total Emissions	Percentage of Total Flights	Percentage of Total Distance	Average Age
A21N	2.93%	4.40%	5.19%	3.05
A319	4.81%	17.06%	7.71%	21.39
E190	0.76%	3.72%	1.32%	13.59
A20N	6.64%	15.10%	10.67%	5.00
A359	17.56%	2.99%	16.39%	5.48
A320	7.65%	23.04%	12.81%	10.00
CRJ9	2.54%	17.35%	5.37%	14.26
B748	23.61%	2.35%	13.17%	10.11
A346	3.83%	0.46%	2.35%	16.58
A321	5.07%	9.87%	7.53%	13.40
A343	11.58%	1.68%	9.10%	24.27
B744	8.28%	0.90%	4.38%	23.57
A333	4.61%	0.93%	3.90%	14.29
B789	0.13%	0.14%	0.10%	4.23

Table 6 Percentage of total emissions, total flights, total distance and average age per type code

To better understand the efficiency of those aircraft models, it is essential to consider the size, range, and age of those aircrafts. Two measurements can shed further light: fuel flow per hour per person and litre per 100-passenger kilometre.

The fuel flow per hour per person is calculated as follows:

$$Fuel\ Flow\ per\ Hour\ per\ Person = \frac{Fuel\ Flow\ per\ Hour}{Total\ Number\ of\ Passengers}$$

This reflects a ratio between the total passenger capacity and the fuel flow per hour for a more justified comparison. The passenger capacity directly impacts the size of the aircraft. The following scatterplot shows the fuel flow per hour per person ratio. The widebody or long-haul aircrafts tend to have a higher fuel flow per hour per person than the narrowbody aircrafts.

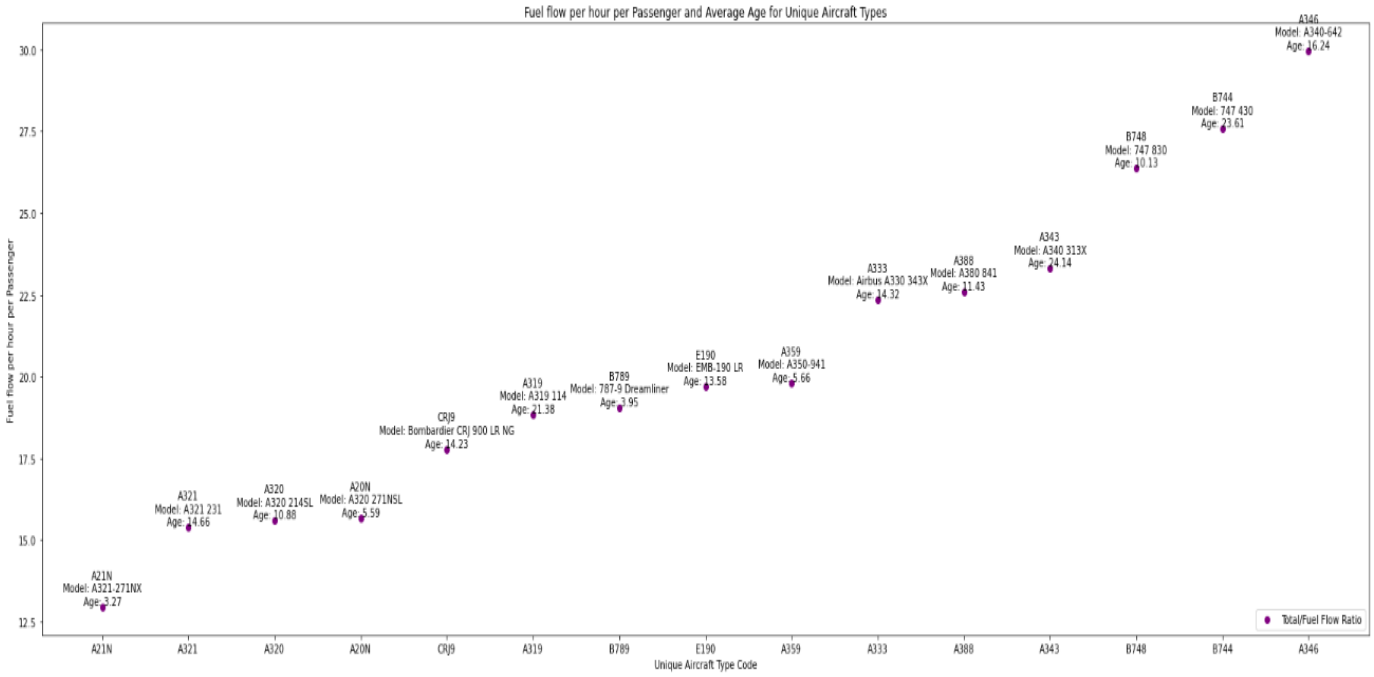

Figure 9 Fuel flow per hour per person by type code

Another widely used metric in the aviation industry for aircraft efficiency is the Litre per 100 passenger kilometres. This metric states how much fuel is consumed by 100 passenger kilometres. Passenger kilometres is calculated by the total number of passengers and the total distance of the flight.

$$liter\ per\ 100\ pkm = \frac{total\ fuel\ consumption\ in\ liter}{(total\ passenger * total\ distance)}$$

Crucial for this metric is the seat load factor, indicating how many available seats were occupied. The historical average load factor of the years can be evaluated from the traffic reports from Lufthansa for each year. The load factor 2023 has to be estimated from the available data until Q3 2023. (Lufthansa, 2020 ; Lufthansa, 2021a ; Lufthansa, 2023a)

This measurement makes it possible to evaluate the fuel efficiency per passenger. The newer generation of aircrafts appear to be more efficient. Changing from the Boing 747 types to the new Airbus 350 series for long-haul flights can decrease the Liter per 100pkm by about 30%. The A350 will operate more efficiently, increase cost savings in operations, and positively impact the carbon footprint. (Lufthansa, n.d.-a)

Lufthansa states that older models should replace newer, more efficient ones. They want to retire all older A340, A320, A330 and B747 and replace with A20N, A21N, A350-900 and B787-900. The A340 are the least efficient models in the current fleet.

	type_code	fuel_per_100_passenger_km_mean	average_age
1	A21N	2.65	3.32
4	A321	3.16	14.63
8	A359	3.26	5.67
3	A320	3.34	10.82
0	A20N	3.62	5.59
9	A388	3.74	11.49
12	B789	3.77	4.02
11	B748	3.96	10.13
10	B744	4.07	23.60
2	A319	4.32	21.38
7	A346	4.32	16.25
5	A333	4.48	14.32
6	A343	5.02	24.13
13	CRJ9	5.11	14.23
14	E190	5.14	13.58

Table 7 Litre per 100 pkm and average age per type code

About 30% less than many current and previously operated long-haul aircraft models will have an equally positive impact on Lufthansa's carbon footprint. (Lufthansa, 2021b)
By calculating these values from the given data and using the load factor, we can estimate the average litre per 100pkm per year for Lufthansa.

Year	Published Lufthansa data	Available dataset
2017	3.76	3.17
2018	3.76	3.79
2019	3.74	3.72
2020	4.21	4.96
2021	3.97	4.98
2022	3.63	3.61
2023		3.45

Table 8 Comparison of Litre per 100 pkm per year from dataset and Lufthansa official communication

The calculation of the l/100pkm shows that Lufthansa's fleet is getting more efficient. Only during the COVID-19-impacted years did the average l/100pkm increase. This can be justified by the change in average distance in those years because long-haul flights decreased, and no new airplanes joined the fleet. In addition, airplanes could not operate in total capacity due to measurements for testing, social distancing, and quarantine. (International Civil Aviation Organization, n.d.)

4. Full development of findings/arguments/reasoned analysis

4.1 Predictive analytics

4.1.1 Demand forecast

To forecast the future years until 2050, the Waypoint 2050 report from the Air Transportation Action Group provides the demand assumptions and scenarios. Three different scenarios for demand growth (low growth, central scenario, high growth) have been published. These scenarios are based on historical data, socio-economic events, and trends.

			Compound annual growth rates			
Scenario	Description	RPKs in 2050	2019-2050	2019-2030	2030-2040	2040-2050
L Low growth	Protectionism deepens along with a reduction in mobility on top of Covid-19 impact.	15 trillion	2.3%	0.6%	2.8%	2.3%
C Central scenario	Continuation of historical trends, but a reduction compared with recent high-growth and taking into account the impact of Covid-19.	22 trillion	3.1%	3.1%	3.2%	3.1%
H High growth	Return to globalisation with a continuation of high growth trends seen in recent years, but from a revised base due to the impact of Covid-19.	25 trillion	3.3%	3.7%	3.4%	3.3%

Figure 10 Compound annual demand growth rates

(Air Transport Action Group, 2021)

To assess these scenarios, we must iterate over each scenario and calculate the total emissions for each year based on the defined growth rates. Based on the predefined growth scenarios from the Air Transportation Action Group, the scenarios project how Lufthansa's total emissions might evolve until 2050.

Using loops, the different growth rates for the scenarios can be added to display the emissions in future years without countermeasures like changing routes, modernizing fleets, or improving operations.

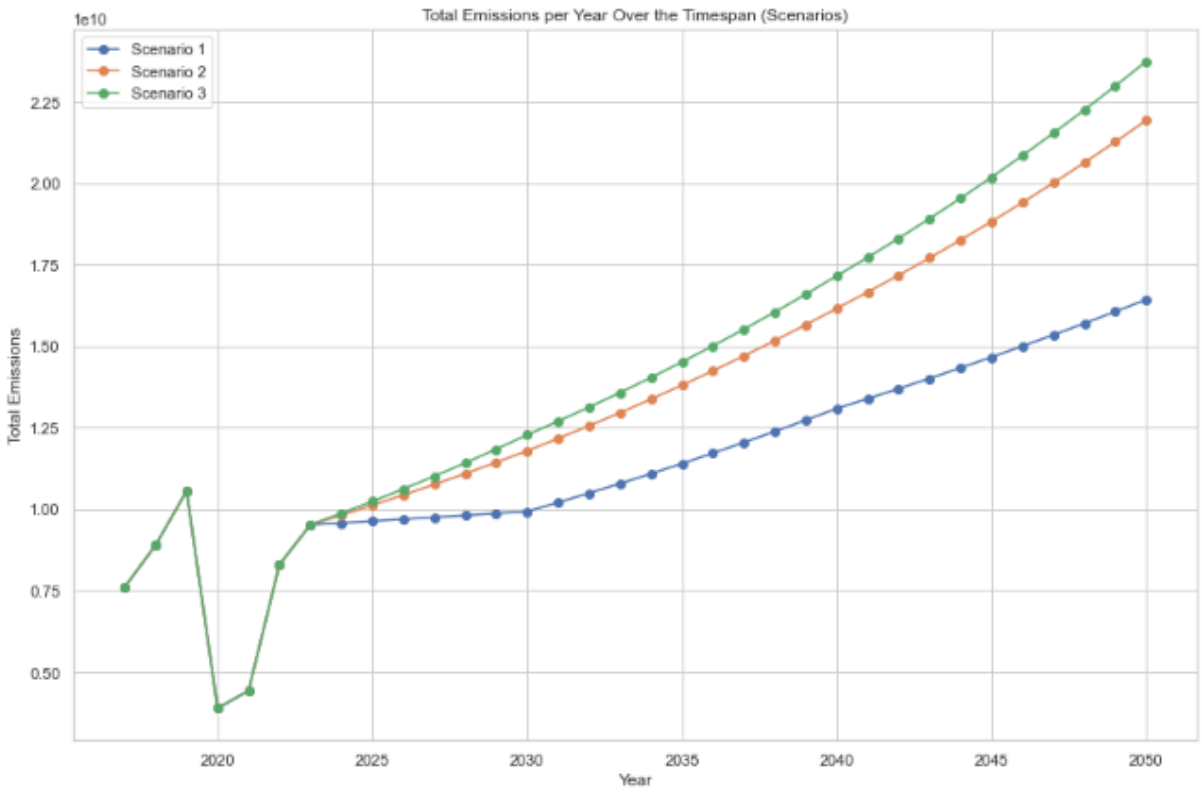

Figure 11 Total emissions forecast to 2050

Applying the officially predicted demand growth rates for aviation demand, in accumulated results in 2050, there will be a difference of 33% or 44% in total emissions compared to the low growth scenario. These scenarios show that no fleet or route change will emit between 16,821,064,428.45 and 24,301,888,448.15 Kilograms of CO2 in 2050. This is a total growth of 280.76 % or 405.63% compared to the initial value in 2023 of 5,991,076,540,42 Kilograms of CO2.

The measures to reduce emissions as effectively as possible must be quite harsh, considering the potential growth of demand in the future. As previously explained, different scenarios are already in place to reduce emissions and reach the net-zero emissions goal by 2050.

The first measurement this thesis investigates is fleet change. Lufthansa has announced it is retiring older, less efficient models and using the newest technology. By switching the aircrafts to the newest version, Lufthansa can decrease fuel consumption and emissions by about 30 percent, compared to the older generation. (Lufthansa, n.d.-a)

Therefore, they have published the new orders for aircrafts until 2029 at Airbus and Boeing. The airline will retire all A340, A320, A330, B777 and B747 models and replace them with the newer generation A20N, A21N, A350-900 and B787-900. (Lufthansa, 2021b; Lufthansa, n.d.-a)

4.2 Prescriptive analytics

4.2.1 Fleet change scenarios

With the published information about new fleet orders until 2029, the following scenarios will assess how much CO2 can be reduced by a renewal of older aircraft models. An aircraft has an average life expectancy of 30 years. (M.Durgut, 2023) Therefore, this would include multiple changes in the timeline until 2050. To rely on actual data and not make assumptions, the scenarios only represent the announced changes.

This can give an insight into how much impact this change can have, especially over time. Therefore, three fleet change scenarios have been developed to outline the importance and impact of fast action. The fleet change information was published by Lufthansa, and all scenarios include the presented demand growth scenarios.

The first fleet change scenario investigated is that the announced models will all be replaced in 2024 with the same amount that retires, seen in figure 12.

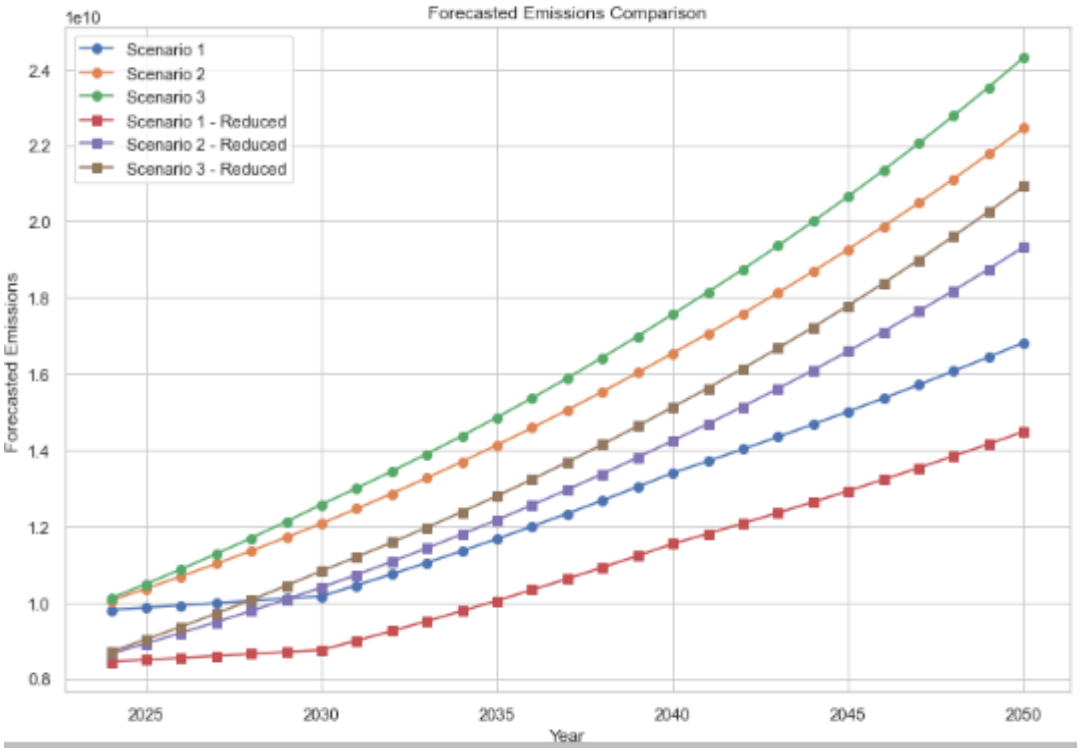

Figure 12 Total emissions forecast for fleet change in 2024 to 2050

If Lufthansa could change all announced retiring models in 2024, the airline could decrease its total CO2 emissions by about 13.8 % across all scenarios. This is not a realistic scenario but highlights the impact of acting quickly. This scenario displays the best-case scenario to reduce emissions as much as possible by replacing aircrafts.

The second fleet change scenario conducts a realistic continuous change over the years from 2024 until 2029. In 2030, the announced replacement is complete. Therefore, the proportional reduction was assessed – how much less CO2 would be emitted when the designated aircrafts are replaced. Thus, with this scenario the designated aircrafts, which have been assessed by the icoa24 the unique identifier, are replaced in a continuous rate per year for the analysis. This means that a subset of the ordered aircrafts will replace the less efficient ones every year. Therefore, the more efficient models have a higher decrease in emissions than the demand forecast increase. The retirement of older models dramatically affects the total emissions until 2029, when all less efficient models are replaced. In 2030, the growth of demand will increase emissions yearly. With a continuous fleet change over time, Lufthansa can reduce its emissions by 10.5% - 12.3%, depending on the growth scenario.

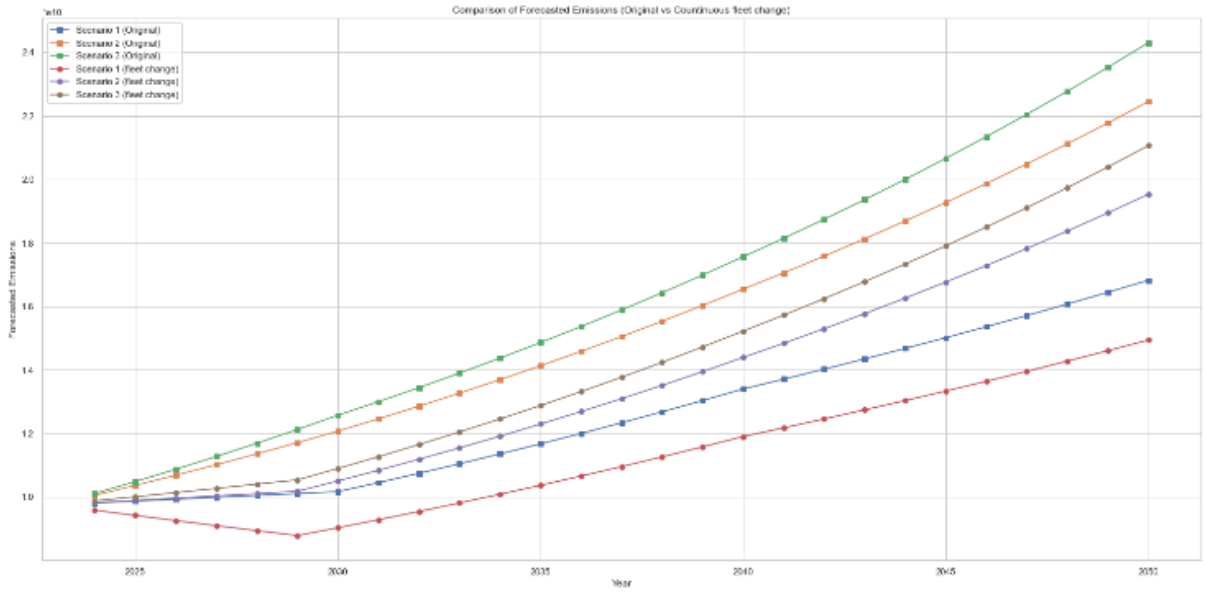

Figure 13 Total emission forecast for continuous fleet change to 2050

The third fleet change scenario investigates what happens if the aircrafts are all delivered in 2029. So, by 2030, they will be operational, and until then, normal growth scenarios will occur. This would be the worst-case scenario in this report and is used again to show and highlight the importance of acting fast. With this approach, Lufthansa could reduce their emissions between 8.8% and 11.1 %.

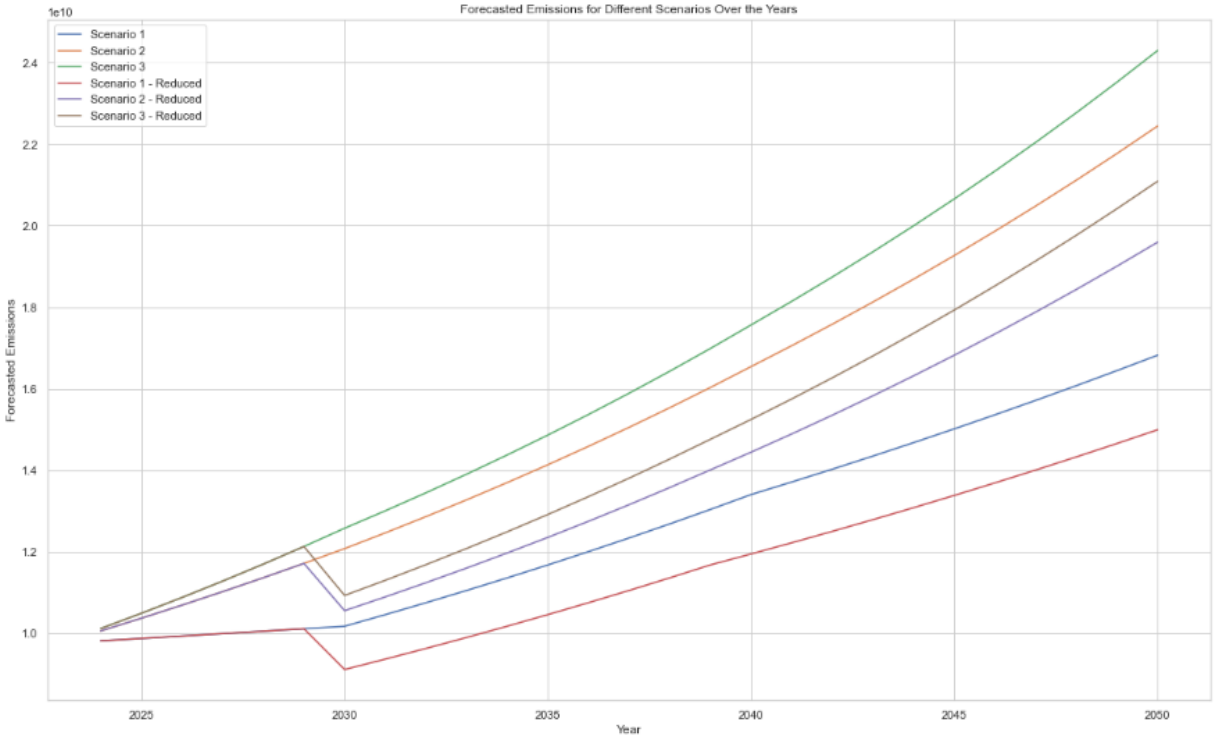

Figure 14 Total emission forecast for fleet change in 2029 to 2050

	Scenario	Forecasted Emissions_Original	Forecasted Emissions_FleetChange2024	Forecasted Emissions	Forecasted Emissions_FleetChangeContinuous	Diff_FleetChange2024_Percent	Diff_FleetChange2029_Percent	Diff_FleetChangeContinuous_Percent
0	Scenario 1	340789041301.10	293472953817.67	310705513932.38	304688349103.77	-13.88	-8.83	-10.59
1	Scenario 2	417817894685.31	359806909409.52	373139085643.20	366780319911.03	-13.88	-10.69	-12.22
2	Scenario 3	441837185508.52	380491295854.52	392391949196.63	386462145213.82	-13.88	-11.19	-12.53

Table 9 Forecasted emissions and difference in % for all fleet change scenarios

In conclusion, the earlier Lufthansa manages to retire the currently operational older models and replace them with the newer generation of aircrafts, they could reduce by 2.7% - 5%. Lufthansa has only published their orders at Airbus and Boing until 2029 for the fleet change scenarios. This means even more older models will get replaced. Also, in the timeframe until 2050, there will be several new generations and models from the manufacturers.

4.2.2 SAF forecast

The second measurement this report investigates is the potential reduction from Sustainable Aviation Fuel. As previously explained, SAF can have various reduction and blending rates, depending on the production method. All certified Sustainable Aviation Fuels can be blended in at a rate of 50%. This means the 50% of SAF will not emit CO2 and the 50% conventional jet fuel will emit 3.16 KG of CO2 per KG fuel. With recent advancements in technology and tests, it is already possible to fully operate an aircraft with 100% SAF. Therefore, this report will investigate three reduction rates of 70%, 85% and 100% to depict what potential SAF has in a real-world example. These reduction rates are also considered in studies from the ICOA. (Air Transport Action Group, 2021)

In addition, the penetration rate is crucial. The penetration rate describes the percentage of SAF in the jet fuel. Because of production complexities, the European Union Safety Agency has forecasted a realistic penetration rate until 2050, given the current market environment. (European Authority for aviation safety., n.d.-a)

	2025	2030	2035	2040	2045	2050
Percentage of SAF used in air transport:	2%	5%	20%	32%	38%	63%
Of which: sub-mandate Synthetic fuels (or e-fuels):	-	0.7%	5%	8%	11%	28%

Figure 15 Percentage of SAF used in air transport until 2050 (European Authority for aviation safety., n.d.-a)

Other, more aggressive scenarios considered in the Waypoint 2050 report for the production capacity and penetration rate have not been considered feasible by other authorities in the economic and technological environment and, therefore, are not considered. (Air Transport Action Group, 2021, European Authority for aviation safety., n.d.-a; International Air Transport Association, 2023d)

While considering the growth scenarios and the possible CO2 emission reduction of 70% from SAF, Lufthansa could reduce their total emissions between 16.22% and 17.06% compared to the growth rates without SAF usage. In 2050, this corresponds to a reduction rate of 44.1% (Figure 16).

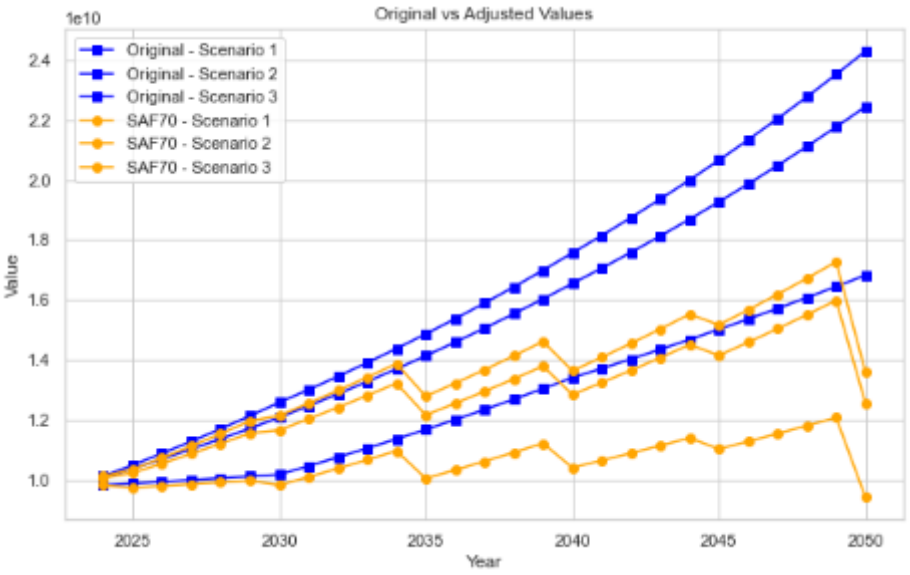

Figure 16 Total emission forecast for SAF with 70% reduction rate

When considering a reduction rate of 85%, Lufthansa could reduce 19.69% and 20.71% of their total emissions until 2050, depending on the demand growth scenario. In 2050, this corresponds to a reduction rate of 53.55% (Figure 17).

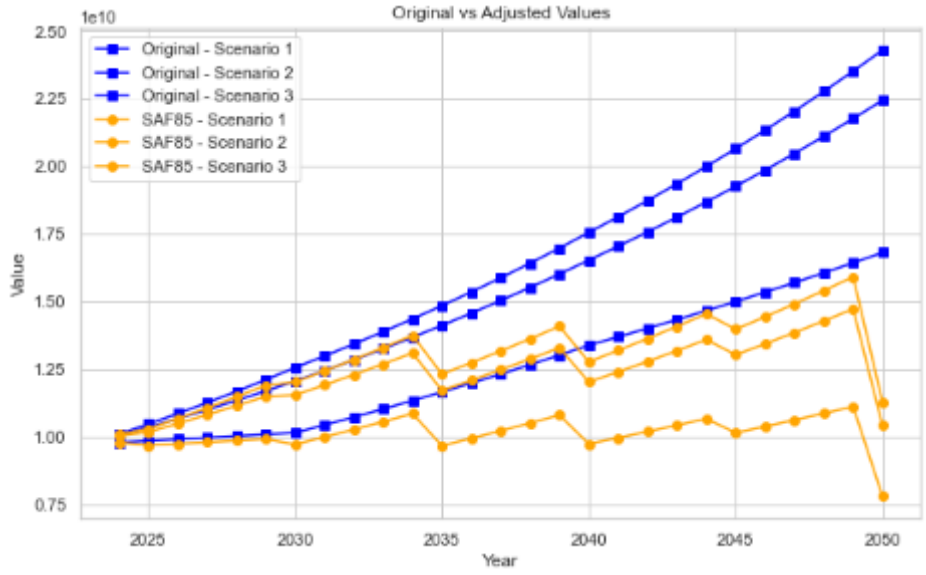

Figure 17 Total emission forecast for SAF with 85% reduction rate

By assuming a 100% reduction in emissions from SAF, Lufthansa could reduce their emissions by 23.17% up to 24.37% of total emissions between 2024 and 2050. In 2050, this corresponds to a reduction rate of 63.0% (Figure 18).

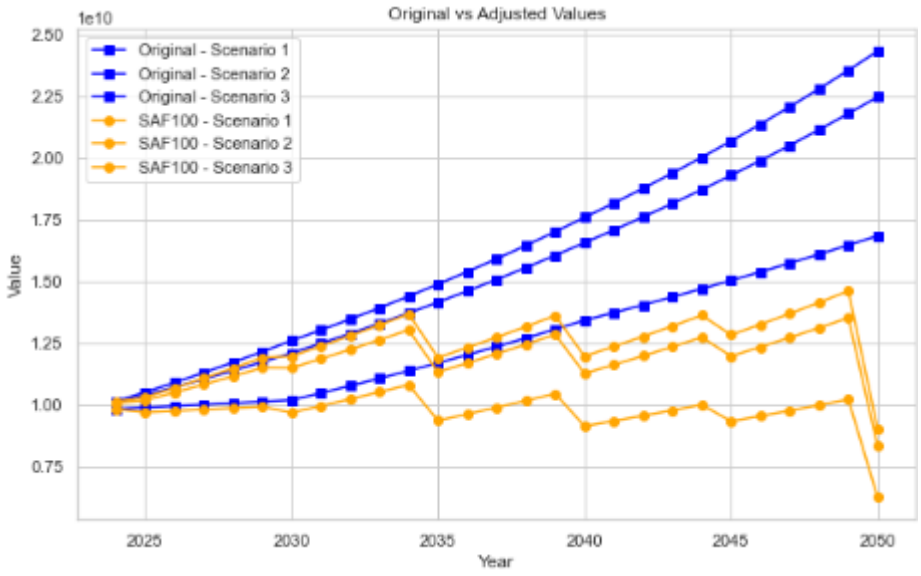

Figure 18 Total emission forecast for SAF with 100% reduction rate

While considering the technical possible scenarios of the penetration rate of SAF in the fuel and the production capacities, it is possible to reduce the CO2 emissions between 16.22% and up to 24.37%, depending on the reduction factor. To reach the goal of carbon neutrality in 2050, Lufthansa could reduce the CO2 emissions between 44.1% and 63% in 2050 compared to the growth scenario values in 2050. Therefore, Lufthansa should focus on operating their aircrafts with as much SAF as possible and intensify their efforts in this technology.

	Scenario	Forecasted Emissions	SAF70	SAF85	SAF100	Diff_SAF70_Percent	Diff_SAF85_Percent	Diff_SAF100_Percent
0	Scenario 1	340789041301.10	285528018761.74	2736863710 74.74	261844723387.73	-16.22	-19.69	-23.17
1	Scenario 2	417817894685.31	347368603833.30	332272327222.16	317176050611.01	-16.86	-20.47	-24.09
2	Scenario 3	441837185508.52	366473645318.48	350324315277.76	334174985237.04	-17.06	-20.71	-24.37

Table 10 Total forecasted emissions and differences in % across all SAF Scenarios

```
Merged Reduction Rate Table in 2050:
     Scenario  Reduction Rate_SAF70  Reduction Rate_SAF85  Reduction Rate
0  Scenario 1                 44.10                 53.55           63.00
1  Scenario 2                 44.10                 53.55           63.00
2  Scenario 3                 44.10                 53.55           63.00
```

Table 11 Reduction rates in 2050 for mulitple SAF scenarios

4.2.3 Combined scenarios for fleet change and SAF

As previously explained, multiple measurements for the reduction of CO2 should take place simultaneously. The constant fleet change is already in place, and so is the usage of SAF. Thus, the most realistic future scenario analysis needs to combine both methods of emission reduction to calculate the full impact of the two measurements.

To evaluate the impact of the two measurements, the different fleet change scenarios and SAF scenarios are combined. This means for the fleet change in 2024, the SAF scenarios for 70%, 85% and 100% reduction. The same is done for the continuous fleet change and the fleet change in 2029.

The table below shows the possible reduction rates considering the different fleet change scenarios and the different reduction rates of the Sustainable Aviation Fuel combined. As suspected, updating the fleet to the newer generations sooner rather than later would impact the possible reductions more, but increasing the CO2 reduction factor of the Sustainable Aviation Fuel would have a more significant impact. By scaling the SAF usage from 70% to 100%, Lufthansa could increase the total reduction by more than 7%. The highest reduction can be achieved by updating the fleet immediately. Furthermore, it would be possible to achieve a higher reduction if the penetration rate of SAF would be higher which was not assessed in this work, which only considered one penetration rate. Thus, the production of SAF has to be cheaper, and ease of availability needs to increase. With the considered penetration rate and realistic growth, it is not possible to reduce the emissions of Lufthansa by more than 33.66% when comparing the total emissions from 2024 to 2050.

	Scenario	Diff_SAF70_Percent_Original	Diff_SAF85_Percent_Original	Diff_SAF100_Percent_Original	Diff_SAF70_Percent_Copied	Diff_SAF85_Percent_Copied	Diff_SAF100_Percent_Copied	Diff_SAF70_Percent	Diff_SAF85_Percent	Diff_SAF100_Percent	Diff_SAF70_Percent_Final	Diff_SAF85_Percent_Final	Diff_SAF100_Percent_Final
0	Scenario 1	-16.22	-19.69	-23.17	-27.85	-30.84	-33.83	-25.00	-28.09	-31.17	-23.31	-26.42	-29.52
1	Scenario 2	-16.86	-20.47	-24.09	-28.40	-31.52	-34.63	-26.89	-30.04	-33.18	-25.44	-28.60	-31.76
2	Scenario 3	-17.06	-20.71	-24.37	-28.57	-31.72	-34.87	-27.32	-30.49	-33.66	-26.02	-29.20	-32.38

Table 12 Total difference in % for all combined scenarios between 2024 and 2050

To conclude all scenarios, the reduction rates in 2050 will be compared to the growth scenarios in 2050.

When solely applying SAF, Lufthansa could reduce their emissions in 2050 by 44.1% up to 63.00%. When changing the fleet in 2029 and using SAF, the worst-case scenario for fleet change in this analysis, Lufthansa could reduce their emissions from 50.18% up to 67.89% in 2050. Applying the continuous fleet change and SAF scenarios, Lufthansa could reduce emissions between 50.35% and 67.93%.

The best-case scenario depicts the full fleet change in 2024 and applying SAF with a 100% reduction rate. This could lead to a reduction of 68.14% compared to the value of the demand growth scenario in 2050. Compared to the current emissions in 2023, that would be a reduction up to 85% depending on the growth scenario.

Merged Dataframe:

	Scenario	Reduction Rate	Source
0	Scenario 1	44.10	Demand grwoth scenario SAF70
1	Scenario 2	44.10	Demand grwoth scenario SAF70
2	Scenario 3	44.10	Demand grwoth scenario SAF70
3	Scenario 1	53.55	Demand grwoth scenario SAF85
4	Scenario 2	53.55	Demand grwoth scenario SAF85
5	Scenario 3	53.55	Demand grwoth scenario SAF85
6	Scenario 1	63.00	Demand grwoth scenario SAF100
7	Scenario 2	63.00	Demand grwoth scenario SAF100
8	Scenario 3	63.00	Demand grwoth scenario SAF100
9	Scenario 1	51.86	Fleet change in 2024 SAF70
10	Scenario 2	51.86	Fleet change in 2024 SAF70
11	Scenario 3	51.86	Fleet change in 2024 SAF70
12	Scenario 1	60.00	Fleet change in 2024 SAF85
13	Scenario 2	60.00	Fleet change in 2024 SAF85
14	Scenario 3	60.00	Fleet change in 2024 SAF85
15	Scenario 1	68.14	Fleet change in 2024 SAF100
16	Scenario 2	68.14	Fleet change in 2024 SAF100
17	Scenario 3	68.14	Fleet change in 2024 SAF100
18	Scenario 1	50.35	continuous fleet change SAF70
19	Scenario 2	51.37	continuous fleet change SAF70
20	Scenario 3	51.55	continuous fleet change SAF70
21	Scenario 1	58.74	continuous fleet change SAF85
22	Scenario 2	59.59	continuous fleet change SAF85
23	Scenario 3	59.74	continuous fleet change SAF85
24	Scenario 1	67.14	continuous fleet change SAF100
25	Scenario 2	67.81	continuous fleet change SAF100
26	Scenario 3	67.93	continuous fleet change SAF100
27	Scenario 1	50.18	Fleet change in 2029 SAF70
28	Scenario 2	51.20	Fleet change in 2029 SAF70
29	Scenario 3	51.48	Fleet change in 2029 SAF70
30	Scenario 1	58.61	Fleet change in 2029 SAF85
31	Scenario 2	59.45	Fleet change in 2029 SAF85
32	Scenario 3	59.69	Fleet change in 2029 SAF85
33	Scenario 1	67.03	Fleet change in 2029 SAF100
34	Scenario 2	67.70	Fleet change in 2029 SAF100
35	Scenario 3	67.89	Fleet change in 2029 SAF100

Table 13 Difference in % for all combined scenarios in 2050

5. Conclusions and limitations of the dissertation

This thesis aimed to explore the impact of different flight demand growth scenarios on CO2 emissions within the aviation industry and understanding how the ICOA's proposed countermeasures to combat carbon emission, namely fleet renewal, and the introduction of SAF, will impact overall CO2 emissions in each of the demand growth scenarios. This final section will first consider limitations that should be kept in mind when reading the analysis, before concluding with an assessment of the ICOA's countermeasure plan against the above analysis.

This work only accessed open-source data and does not cover all emissions of the Lufthansa airline. To increase the credibility of the data, multiple sources have been used to reduce discrepancies. The forecasts on penetration rate and demand growth in the industry are based on publications from authorities and other research to not exceed the scope of work. Thus, one should consider that there are other possible future scenarios that were not included in this work, which may provide a more realistic forecast for the future.

There are too many uncertainties to cover during the analysis, especially when considering data from the last seven years and predicting 26 years into the future. Many geopolitical factors like the COVID-19 pandemic, the Russian-Ukrainian war, and the recent Israel-Palestine conflict impact the aviation industry by increasing inflation, increasing oil and gas prices, limiting flight routes and operations, etc. Minor factors like Germany's budgetary crisis can also tremendously affect this case. During the course of this work, Germany announced an increase in the CO2 tax, planning to implement a tax on conventional aviation fuel. These plans shifted now to an increase of the already existing aviation tax within a few days. (Flottau & Flottau, 2023; International Air Transport Association, 2023e)

This could lead to an effect on the cost structure of the airline and increase their efforts to move quicker towards Sustainable Aviation Fuel.

In conclusion, the analysis of the data and the topic has shown that becoming carbon neutral/ net zero by 2050 is a very ambitious goal. All operations and technical components must increase their efficiency to even come close to that goal.
This thesis only considered the two most promising methods of reducing CO2 emissions for an airline. The presented analysis shows that, if realistically employed with a continuous fleet change and a SAF reduction rate of 100%, those two methods will reduce the total CO2 emissions up to 67.93% in 2050. Considering the total emissions between 2024 and 2050, this would correspond to a reduction of 33.66%. This would be an outstanding achievement considering maintaining profitability and the short time frame. However, to comply with the Paris Agreement and have the peak of emissions in 2035, more than these two measurements are needed.

The analysis also concludes that the earlier the measurement occurs, the less CO2 will be emitted. The most realistic method would be changing older, less efficient models to the newest generation as quickly as possible. This would ensure that Lufthansa can cope with the forecasted growth of the industry and still operate cost-effectively and in an environmentally conscious manner. Today, Lufthansa only announced the newly ordered aircrafts until 2029. After 2029, this analysis could not consider possible reduction from newer technologies. However, further advancements are likely to happen and will further reduce emissions. Nonetheless, other methods to reduce CO2 must be implemented, such as route optimization and ground improvements. Those can also be realized in the short term and could impact the 2035 peak. Additional to the presented approaches, the most significant impact on the net zero goal could have carbon offsetting or other carbon mitigation measures. As previously explained, this would not solve the fundamental problem of emitting Carbon Dioxide but to reach carbon neutrality, this will be an immense factor and it can be implemented as early as possible without major dependencies on technological progress.

The final conclusion of this analysis is a strong recommendation to invest in Sustainable Aviation Fuel technologies and infrastructure to achieve a higher penetration rate as predicted. There are already projects to use 100% SAF in commercial flights, and Lufthansa is partnering with several companies to produce SAF for their needs.

Lufthansa's efforts already seem high and effective in comparison to the global industry. However, to achieve the goal of becoming net zero by 2050, the measurements must be intensified in all areas of their operation. Sustainable Aviation Fuel cannot realistically save the aviation industry from being detrimental to the environment.
This analysis only investigates the case of Lufthansa. As previously explained, Lufthansa is already one of top five largest buyers of SAF worldwide. This means that Lufthansa is already one of the industry pioneers.

The goal to become net zero by 2050 also impacts all other airlines that might have to catch up and are not implementing this technology as fast as Lufthansa. This might influence the strategy of operating cost-efficiently as every other airline has to apply the same or similar measurements that will affect the cost structure of the companies.

6. References

Air Transport Action Group. (2021). Waypoint 2050. In *aviationbenefits.org*. Retrieved December 30, 2023, from https://aviationbenefits.org/media/167417/w2050_v2021_27sept_full.pdf

Airbus. (2023a). *Sustainable aviation fuel*. airbus.com. Retrieved December 26, 2023, from https://www.airbus.com/en/sustainability/respecting-the-planet/decarbonisation/sustainable-aviation-fuel

Airbus. (2023b, March 9). *Airbus' most popular aircraft takes to the skies with 100% sustainable aviation fuel*. Retrieved December 22, 2023, from https://www.airbus.com/en/newsroom/stories/2023-03-airbus-most-popular-aircraft-takes-to-the-skies-with-100-sustainable

Bauen, A., Bitossi, N., German, L., & Harris, A. (2020). Sustainable aviation fuels. *technology.matthey.com*, *64*, https://technology.matthey.com/. https://doi.org/10.1595/205651320X15816756012040

Buis, A. (2022, November 16). The atmosphere: Getting a handle on carbon dioxide. *Climate Change: Vital Signs of the Planet*. https://climate.nasa.gov/news/2915/the-atmosphere-getting-a-handle-on-carbon-dioxide/#:~:text=The%20concentration%20of%20carbon%20dioxide,it%20was%20near%20370%20ppm.

Calvin, K., Dasgupta, D., Krinner, G., Mukherji, A., Thorne, P., Trisos, C. H., Romero, J., Aldunce, P., Barrett, K., Blanco, G., Cheung, W. W. L., Connors, S., Denton, F., Diongue-Niang, A., Dodman, D., Garschagen, M., Geden, O., Hayward, B., Jones, C. D., . . . Ha, M. (2023). *IPCC, 2023: Climate Change 2023: Synthesis Report. Contribution of Working Groups I, II and III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change [Core Writing Team, H. Lee and J. Romero (eds.)]. IPCC, Geneva, Switzerland.* https://doi.org/10.59327/ipcc/ar6-9789291691647

Caswell, M. (2023, September 1). *Lufthansa to reactivate more A380s – Business Traveller*. Business Traveller. https://www.businesstraveller.com/business-travel/2023/09/01/lufthansa-to-reactivate-more-a380s/

Chen, W., Zhu, D., Ciais, P., Huang, C., Viovy, N., & Kageyama, M. (2019). Response of vegetation cover to CO2 and climate changes between Last Glacial Maximum and pre-industrial period in a dynamic global vegetation model. *Quaternary Science Reviews*, *218*, 293–305. https://doi.org/10.1016/j.quascirev.2019.06.003

Dube, K., Nhamo, G., & Chikodzi, D. (2021). COVID-19 pandemic and prospects for recovery of the global aviation industry. *Journal of Air Transport Management*, *92*, 102022. https://doi.org/10.1016/j.jairtraman.2021.102022

Environment Branch of the International Civil Aviation Organization. (2015). AO ENVIRONMENTAL REPORT 2016. In *icao.int*. Retrieved December 30, 2023, from https://www.icao.int/environmental-protection/pages/env2016.aspx

Eurocontrol. (2021, September 28). *EUROCONTROL Data Snapshot #18*. Retrieved December 27, 2023, from https://www.eurocontrol.int/publication/eurocontrol-data-snapshot-18-56-lower-emissions-53-less-flights

European Authority for aviation safety. (n.d.-a). *Fit for 55 and ReFuelEU Aviation | EASA*. EASA. Retrieved December 25, 2023, from https://www.easa.europa.eu/en/light/topics/fit-55-and-refueleu-aviation

European Authority for aviation safety. (n.d.-b). *How sustainable are SAF? | EASA Eco*. EASA Eco. Retrieved December 25, 2023, from https://www.easa.europa.eu/eco/eaer/topics/sustainable-aviation-fuels/how-sustainable-are-saf#ghg-emissions-reductions

European Authority for aviation safety. (n.d.). *Sustainable Aviation Fuels | EASA EcO*. EASA Eco. Retrieved December 12, 2023, from https://www.easa.europa.eu/eco/eaer/topics/sustainable-aviation-fuels#:~:text=Current%20SAF%20supply%20remains%20low,Power%2Dto%2DLiquid%20SAF.

European Commission. (n.d.). *Causes of climate change*. Climate Action. Retrieved December 30, 2023, from https://climate.ec.europa.eu/climate-change/causes-climate-change_en

Flottau, J., & Flottau, J. (2023, December 21). *Germany abandons fuel tax plans, raises aviation tax*. Aviation Week Network. https://aviationweek.com/air-transport/safety-ops-regulation/germany-abandons-fuel-tax-plans-raises-aviation-tax

Gabric, A. J. (2023). The Climate Change Crisis: A Review of its causes and Possible responses. *Atmosphere*, *14*(7), 1081. https://doi.org/10.3390/atmos14071081

Global Monitoring Laboratory. (2023, December). *Global Monitoring Laboratory - Carbon Cycle Greenhouse Gases*. Retrieved December 29, 2023, from https://gml.noaa.gov/ccgg/trends/

Global Programme on Risk Assessment and Management for Adaptation to Climate Change (Loss and Damage). (2021). Climate Risk Management. In *giz.de*. Deutsche Gesellschaft für Internationale Zusammenarbeit (GIZ) GmbH. https://www.giz.de/en/downloads/giz2021_en_climate-risk-management.pdf

Igwenagu, C. (2016). *Fundamentals of Research Methodology and Data Collection*. https://www.researchgate.net/publication/303381524_Fundamentals_of_research_methodology_and_data_collection

Intergovernmental Panel on Climate Change. (2015). *Climate Change 2014: Synthesis Report*. https://ar5-syr.ipcc.ch/ipcc/ipcc/resources/pdf/IPCC_SynthesisReport.pdf

Intergovernmental Panel on Climate Change. (2022). The evidence is clear: the time for action is now. We can halve emissions by 2030. In *ipcc.ch*.

Intergovernmental Panel on Climate Change. (2023). Urgent climate action can secure a liveable future for all — IPCC. In *IPCC*. Retrieved December 23, 2023, from https://www.ipcc.ch/2023/03/20/press-release-ar6-synthesis-report/

International Air Transport Association. (n.d.). RP 1726 Passenger CO2 Calculation Methodology. In *iata.org*. https://www.iata.org/en/programs/environment/passenger-emissions-methodology/?_fs=16295728731-15014572231#tab-1

International Air Transport Association. (2023a, October 4). *Passenger demand recovery continues in August*. https://www.iata.org/en/pressroom/2023-releases/2023-10-04-01/

International Air Transport Association. (2023b). Net zero 2050: sustainable aviation fuels. In *iata.org*. Retrieved December 30, 2023, from https://www.iata.org/en/iata-repository/pressroom/fact-sheets/fact-sheet---alternative-fuels/

International Air Transport Association. (2023c, December 5). *Passenger demand recovery continues on track in October*. https://www.iata.org/en/pressroom/2023-releases/2023-12-05-02/

International Air Transport Association. (2023d, December 6). *SAF volumes growing but still missing opportunities*. Retrieved December 29, 2023, from https://www.iata.org/en/pressroom/2023-releases/2023-12-06-02/#:~:text=%E2%80%9CThe%20doubling%20of%20SAF%20production,supply%20and%20keeps%20prices%20high.

International Air Transport Association. (2023e, December 13). *Statement on proposed German domestic kerosene tax*. https://www.iata.org/en/pressroom/2023-releases/2023-12-13-01/

International Civil Aviation Organization. (n.d.). *Guidance for Air Travel through the COVID-19 Public Health Crisis*. https://www.icao.int/covid/cart/Pages/CART-Take-off.aspx

International Civil Aviation Organization. (2018). *ICAO Carbon Emissions Calculator Methodology*. Retrieved December 30, 2023, from https://applications.icao.int/icec/Methodology%20ICAO%20Carbon%20Calculator_v11.1-2018.pdf?_gl=1*1xdxqla*_ga*ODA3MjE1NDAwLjE2OTc2NjA5ODA.*_ga_992N3YDLBQ*MTY5OTIwMDQ0Ny40LjEuMTY5OTIwMDQ3Mi4wLjAuMA

International Civil Aviation Organization. (2022). Report on the feasibility of a long-term aspirational goal (LTAG) for international civil aviation co2 emission reductions. In *icao.int*. Retrieved December 30, 2023, from https://www.icao.int/environmental-protection/LTAG/Documents/REPORT%20ON%20THE%20FEASIBILITY%20OF%20A%20LONG-TERM%20ASPIRATIONAL%20GOAL_en.pdf

International Energy Agency. (n.d.). *Aviation - IEA*. IEA. Retrieved December 21, 2023, from https://www.iea.org/energy-system/transport/aviation

Janson, M. (2023, December 4). 10 Länder verursachen zwei Drittel der CO2-Emissionen. *Statista Daily Data*. https://de.statista.com/infografik/23383/anteil-der-laender-an-den-weltweiten-co2-emissionen/

Lashof, D. A., & Ahuja, D. R. (1990). Relative contributions of greenhouse gas emissions to global warming. *Nature*, *344*(6266), 529–531. https://doi.org/10.1038/344529a0

Levingston, C. (2021, September 13). *Four Things to Know About Sustainable Aviation Fuel (SAF) - The GE Aerospace Blog | Aviation & Flight News*. The GE Aerospace Blog | Aviation & Flight News. https://blog.geaerospace.com/sustainability/four-things-to-know-about-sustainable-aviation-fuel-saf/)

Lufthansa. (n.d.-a). *Continuous fleet modernization* [Press release]. https://www.lufthansagroup.com/en/responsibility/climate-environment/modern-fleet.html

Lufthansa. (n.d.-b). *Lufthansa fleet*. lufthansa.com. https://www.lufthansa.com/de/en/seat-maps

Lufthansa. (n.d.-c). *SUSTAINABLE AVIATION FUEL*. SAF Is a Genuine Alternative to Fossil Aviation Fuel and Essential for the Energy Transition in Aviation. Retrieved December 26, 2023, from https://www.lufthansagroup.com/en/responsibility/climate-environment/sustainable-aviation-fuel.html

Lufthansa. (2020). Investor Info Q4 2020. In *Lufthansa.com*. https://investor-relations.lufthansagroup.com/fileadmin/downloads/en/financial-reports/traffic-figures/lufthansa/2020/LH-Investor-Info-2020-04-e.pdf

Lufthansa. (2021a). Investor Info Q4 2021. In *lufthansa.com*. https://investor-relations.lufthansagroup.com/fileadmin/downloads/en/financial-reports/traffic-figures/lufthansa/2021/LH-Investor-Info-2021-04-e.pdf

Lufthansa. (2021b, May 3). *Lufthansa Group pushes ahead with fleet modernization and purchases ten state-of-the-art long-haul aircraft* [Press release]. https://newsroom.lufthansagroup.com/en/lufthansa-group-pushes-ahead-with-fleet-modernization-and-purchases-ten-state-of-the-art-long-haul-aircraft/

Lufthansa. (2022, September 13). *Lufthansa Group and OMV further strengthen partnership on Sustainable Aviation Fuel* [Press release]. https://newsroom.lufthansagroup.com/en/lufthansa-group-and-omv-further-strengthen-partnership-on-sustainable-aviation-fuel/

Lufthansa. (2023a). Deutsche Lufthansa AG - 3rd Interim Report 2023. In *lufthansa.com*. https://investor-relations.lufthansagroup.com/fileadmin/downloads/en/financial-reports/interims-reports/LH-QR-2023-3-e.pdf

Lufthansa. (2023b). Investor Info Q3 2023. In *lufthansa.com*. https://investor-relations.lufthansagroup.com/fileadmin/downloads/en/financial-reports/traffic-figures/lufthansa/2023/Traffic-figures-Q3-2023_EN.pdf

Lufthansa. (2023c). Lufthansa Group Annual Report 2022. In *lufthansa.com*. https://investor-relations.lufthansagroup.com/fileadmin/downloads/en/financial-reports/annual-reports/LH-AR-2022-e.pdf#page=27

Lufthansa. (2023d). Sustainability 2022 fact sheet. In *Lufthansa.com*. Deutsche Lufthansa AG. Retrieved December 30, 2023, from https://www.lufthansagroup.com/media/downloads/en/responsibility/LH-Factsheet-Sustainability-2022.pdf

Lufthansa. (2023e, November 1). *Deutsche Lufthansa AG - 3rd Interim Report* [Press release]. https://investor-

relations.lufthansagroup.com/fileadmin/downloads/en/financial-reports/interims-reports/LH-QR-2023-3-e.pdf

Mannion, P., Parry, E., Patel, M., Ringvold, E., & Scott, J. (2023). Carbon removals: How to scale a new gigaton industry. In *McKinsey & Company*. https://www.mckinsey.com/capabilities/sustainability/our-insights/carbon-removals-how-to-scale-a-new-gigaton-industry

M.Durgut. (2023, December 22). *How long does a commercial aircraft last?* Aviation Related Posts, Aviation Pioneers and Aviation Accidents. https://www.aviationfile.com/how-long-does-a-commercial-aircraft-last/#:~:text=With%20proper%20maintenance%20and%20repair,to%20economic%20and%20operational%20reasons.

Mohajan, H. K. (2011). Greenhouse gas emissions increase global warming. In *International Journal of Economic and Political Integration*. https://mpra.ub.uni-muenchen.de/50839/1/MPRA_paper_50839.pdf

Ng, K. S., Farooq, D., & Yang, A. (2021). Global biorenewable development strategies for sustainable aviation fuel production. *Renewable & Sustainable Energy Reviews*, *150*, 111502. https://doi.org/10.1016/j.rser.2021.111502

Organization for Economic Cooperation and Development (OECD). (n.d.). Indicators to measure decoupling of environmental pressure from economic growth. In *oecd.org*. https://www.oecd.org/environment/indicators-modelling-outlooks/1933638.pdf

Ritchie, H. (2023, December 28). *Emissions by sector: where do greenhouse gases come from?* Our World in Data. https://ourworldindata.org/emissions-by-sector?ref=ctvc.co

ScienceDirect.com | science, health and medical journals, full text articles and books. (n.d.). https://pdf.sciencedirectassets.com/271425/1-s2.0-S0376042122X00080/1-s2.0-S0376042122000689/main.pdf?X-Amz-Security-Token=IQoJb3JpZ2luX2VjEAYaCXVzLWVhc3QtMSJHMEUCIQD62TUuJDVIaroxmFaCf0sP0roC37SG97n6w8duli0GKAIgESCIsohB7rwoe0%2B3cyMH7q69mYbuzn1Q9IVZYR%2BHzsoqvAUIj%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FARAFGgwwNTkwMDM1NDY4NjUiDLyrWj53jyahSacrPyqQBcU1EH70hvNGiDQU69pr9A6Dy9eusfiMq1aqT4JfU7tZEdBVemkTuJLjFYocYVOVQXJzx9qTbCD9zyDmFMcbrZw0%2FduMHR6qgIohnDzu5yPEwt%2FPqVp6EnkJzNovLLV7VS%2FHsbrG9pIJZhUFu90vhmTFXXelBN8rSLZU2KW08%2FGdcNp9D2nJ2OcVuJhRzcU3QSRPXNZpYr9KAPjaKIbffG3OUY5RJYuicHRyUUca1j%2Bu%2B6GvQGcdmgnI5TsTHY36NhViZ%2BpC5u47GV5Q%2Fzg9PtwTKiS9UQhKLK6UZ6TdrBv%2FDQzw

jA3ARtiDiLE7eGD8pGF6WYbyaACrrHZTPPXEDeaN9YJuCrbdCppeFx9Qi9y13M4xzzEm%2FortG3tnr2UMSYI2KueQxmQ%2BHjB81iIMGARPM6tuLIfaI1PMLsZbh3G3FP8DQkn3%2BoaFRp%2FKK6J0NHdLHYqL6J0XgvCZ4Om3iFoHSj9B9UFTpr0sqP5wFZNniE%2FvQbp1gPMv87YCLGFnsrOuyZIYr6F21ss%2FNUT6n25GTF1ITmV4pfICz5etAQJBBmXu6lHEM5afTr2RIAtR8Jo7pzetwjxefLBg%2BhcO7U7EEuQEdFTeVTDcRikp1JuQN3XRD%2F7AyO5d7nrptTVtaoLYqdRwV

SkyNRG. (2023, September 15). *The basics of SAF Technology | The HEFA process | SkyNRG*. https://skynrg.com/sustainable-aviation-fuel/technology-basics/

Statista. (2023a, August 29). *Airline industry - passenger traffic worldwide 2004-2022 | Statista*. https://www.statista.com/statistics/564717/airline-industry-passenger-traffic-globally/

Statista. (2023b, November 14). *CO2-Ausstoß weltweit nach Sektoren | Statista*. https://de.statista.com/statistik/daten/studie/167957/umfrage/verteilung-der-co-emissionen-weltweit-nach-bereich/

Tanzil, A. H., Brandt, K., Wolcott, M. P., Zhang, X., & Garcìa-Pèrez, M. (2021). Strategic assessment of sustainable aviation fuel production technologies: Yield improvement and cost reduction opportunities. *Biomass and Bioenergy*, *145*, 105942. https://doi.org/10.1016/j.biombioe.2020.105942

The Economist. (2020, May 21). Carbon offsetting is essential to tackling climate change. *The Economist*. https://www.economist.com/briefing/2020/05/21/carbon-offsetting-is-essential-to-tackling-climate-change?utm_medium=cpc.adword.pd&utm_source=google&ppccampaignID=18151738051&ppcadID=&utm_campaign=a.22brand_pmax&utm_content=conversion.direct-response.anonymous&gad_source=1&gclid=EAIaIQobChMIw4e6lZe8gwMV8odQBh1iAAeeEAAYAiAAEgJC0fD_BwE&gclsrc=aw.ds

Undavalli, V. K., Olatunde, O. B., Boylu, R., Wei, C., Haeker, J., Hamilton, J., & Khandelwal, B. (2023). Recent advancements in sustainable aviation fuels. *Progress in Aerospace Sciences*, *136*, 100876. https://doi.org/10.1016/j.paerosci.2022.100876

United Nations. (2022, April 4). *Secretary-General Warns of Climate Emergency, Calling Intergovernmental Panel's Report 'a File of Shame', While Saying Leaders 'Are Lying', Fuelling Flames | UN Press*. https://press.un.org/en/2022/sgsm21228.doc.htm

United Nations Framework Convention on Climate Change, UNFCCC. (n.d.). *The Paris Agreement*. unfccc.int. Retrieved December 30, 2023, from https://unfccc.int/process-and-meetings/the-paris-agreement

US EPA. (2023, May 30). *Greenhouse Gases Equivalencies Calculator - Calculations and References | US EPA*. Retrieved December 19, 2023, from https://www.epa.gov/energy/greenhouse-gases-equivalencies-calculator-calculations-and-references

Wang, F., Harindintwali, J. D., Yuan, Z., Wang, M., Wang, F., Li, S., Yin, Z., Huang, L., Fu, Y., Li, L., Chang, S. X., Zhang, L., Rinklebe, J., Yuan, Z., Zhu, Q., Xiang, L., Tsang, D. C., Xu, L., Jiang, X., . . . Chen, J. M. (2021). Technologies and perspectives for achieving carbon neutrality. *The Innovation*, *2*(4), 100180. https://doi.org/10.1016/j.xinn.2021.100180

7. Appendix

```
# Master Thesis SAF and fleet reduction

#loading packages

import pandas as pd
import numpy as np
from typing import List, Tuple
from geopy.distance import geodesic
import logging
import matplotlib.pyplot as plt
from prettytable import PrettyTable
import seaborn as sns
import statsmodels.api as sm
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.statespace.sarimax import SARIMAX

# loading data and convert into dataframes

data_path=r'C:\Users\lauri\Downloads'
routes_filepath=f'{data_path}/routes.dat.xlsx'
column_names=['airline','airline_id','src_airport','src_airport_id','dest_airport','dest_airport_id','codeshare','stops','equipment']
routes_df=pd.read_excel(routes_filepath,names=column_names)
lh_df=routes_df[routes_df['airline']=='LH'].copy()

airports_filepath=f'{data_path}/airports2.csv'
column_names=['airport_id','name','city','country','iata','icao','latitude','longitude','altitude','timezone','dst','tz','zone','type']
airports_df=pd.read_csv(airports_filepath,names=column_names)

lh_flights_filepath=f'{data_path}/all_lh_flights3.csv'
lh_flights_df=pd.read_csv(lh_flights_filepath )
lh_flights_df = lh_flights_df.drop(['Unnamed: 0', 'Unnamed: 10'], axis=1)
lh_flights_df = lh_flights_df.rename(columns={'departure': 'src_airport_icao', 'destination': 'dest_airport_icao'})


aircraft_type_filepath=f'{data_path}/aircraft-database-complete-2023-10.xlsx'
column_names=['icao24','registration','manufacturer_icao','manufacturer_name','aircraft_model','type_code','serial_number','linenumber','icao_aircraft_type','operator','company','operator_c
allsign','operator_icao','operator_iata','category']
aircraft_type_df=pd.read_excel(aircraft_type_filepath,names=column_names)

lh_fleet_filepath = f'{data_path}/lh_fleet_data.xlsx'
column_names=['registration','Name','Current_location','total_hours','ICAO_code','Airline','First_Class','Business_Class','Premium_Economy_Class','Economy_Class','Total','Cargo_Capaci
ty','Range','MLW','MTOW','ZFW','Fuel_Capacity','fuel_flow_hour','Service_Ceiling','Cruising_Speed','Cost_Index','retired']
lh_fleet_df = pd.read_excel(lh_fleet_filepath)


lh_fleet_age_filepath = f'{data_path}/lh_fleet_age.xlsx'
column_names=['registration','aircraft_type','serial_number','age']
lh_fleet_age_df = pd.read_excel(lh_fleet_age_filepath, names=column_names)


lh_historic_fleet_filepath=f'{data_path}/lh_historic_fleet.xlsx'
column_names=['registration','aircraft_type','configuration','delivered','exit_date','Status','remark','aircraft_name']
lh_historic_fleet_df=pd.read_excel(lh_historic_fleet_filepath ,names=column_names )

lh_current_fleet_filepath=f'{data_path}/lh_current_fleet.xlsx'
column_names=['registration','aircraft_type','configuration','delivered','remark','aircraft_name', 'age']
lh_current_fleet_df=pd.read_excel(lh_current_fleet_filepath ,names=column_names )

ulh_flights_df = lh_flights_df.dropna(subset=['src_airport_icao', 'dest_airport_icao'])

# Reset the index after dropping rows
ulh_flights_df = ulh_flights_df.reset_index(drop=True)
ulh_flights_df
```

```
# Claculatin the fleet age for historic fleet

# Convert 'DELIVERED' and 'EXIT DATE' columns to datetime objects
lh_historic_fleet_df['delivered'] = pd.to_datetime(lh_historic_fleet_df['delivered'])
lh_historic_fleet_df['exit_date'] = pd.to_datetime(lh_historic_fleet_df['exit_date'])

# Calculate the age in years and create a new 'AGE' column
lh_historic_fleet_df['age'] = (lh_historic_fleet_df['exit_date'] - lh_historic_fleet_df['delivered']).dt.days / 365.25

# Display the updated DataFrame
lh_historic_fleet_df

# merge dataframes

# Merge the two DataFrames on the common column (registration)
merged_fleet_df = pd.merge(lh_fleet_df, lh_current_fleet_df[['registration', 'delivered', 'remark', 'aircraft_name', 'age']],
                           left_on='registration', right_on='registration', how='left')



# Display the resulting DataFrame
merged_fleet_df



ulh_flights_df = lh_flights_df.dropna(subset=['src_airport_icao', 'dest_airport_icao']).copy()

# Convert 'date' to datetime if it's not already
ulh_flights_df['lastseen'] = pd.to_datetime(ulh_flights_df['lastseen'])
ulh_flights_df['firstseen'] = pd.to_datetime(ulh_flights_df['firstseen'])
# Filter data for the specified date range
start_date = '2017-01-01'
end_date = '2023-06-30'
ulh_flights_df = ulh_flights_df[(ulh_flights_df['firstseen'] >= start_date) & (ulh_flights_df['lastseen'] <= end_date)]

# Reset the index after dropping rows and filtering by date
ulh_flights_df = ulh_flights_df.reset_index(drop=True)
ulh_flights_df

# Selecting specific columns from airports_df
new_airports_df = airports_df[['airport_id', 'name', 'city', 'country', 'iata', 'icao', 'latitude', 'longitude', 'altitude']]
new_airports_df

merged_df = pd.merge(ulh_flights_df, new_airports_df, how='left', left_on='src_airport_icao', right_on='icao', suffixes=('_flight', '_src'))
merged_df = pd.merge(merged_df, new_airports_df, how='left', left_on='dest_airport_icao', right_on='icao', suffixes=('_src', '_dest'))

merged_df

# Calculating the distance between airports

# Define a function to calculate distance
def calculate_distance(row):
    src_coords = (row['latitude_src'], row['longitude_src'])
    dest_coords = (row['latitude_dest'], row['longitude_dest'])

    try:
        distance = geodesic(src_coords, dest_coords).kilometers
        return distance
    except ValueError:
        return None  # Handle NaN or invalid coordinates

# Create a new 'distance' column
merged_df['distance'] = merged_df.apply(calculate_distance, axis=1)

#  merge based on the common column "icao24"
merged_df = merged_df.merge(aircraft_type_df, on="icao24", how="left")



columns_to_add = ['manufacturer_icao','manufacturer_name','aircraft_model','type_code','serial_number']

# Add the selected columns
```

```
for column in columns_to_add:
    merged_df[column] = merged_df[column]

# merge based on the common column "icao24"
merged_df = merged_df.merge(merged_fleet_df, on="registration", how="left")

# Select the columns
columns_to_add = ['registration','Total', 'total_hours', 'fuel_flow_hour','age']
# Add the selected columns
for column in columns_to_add:
    merged_df[column] = merged_df[column]

# creating a copy of Dataframe to simulate fleet change scenarios

# Define the mapping for type code and fuel flow values
type_code_mapping = {
    'A320': 'A20N',
    'A333': 'A21N',
    'A343': 'A359',
    'A346': 'A359',
    'B744': 'B787',
    'B748': 'B787'
}

fuel_flow_mapping = {
    'A320': 2600,
    'A333': 2600,
    'A343': 5800,
    'A346': 5800,
    'B744': 5600,
    'B748': 5600
}

# Create a copy of the DataFrame with replaced values
copied_df = merged_df.copy()

# Replace values in 'type_code' column
copied_df['type_code'].replace(type_code_mapping, inplace=True)

# Replace values in 'fuel_flow_hour' column
copied_df['fuel_flow_hour'].replace(fuel_flow_mapping, inplace=True)
copied_df

# Data operations

merged_df['duration'] = pd.to_numeric(merged_df['duration'], errors='coerce')

merged_df['duration_hours'] = merged_df['duration'] / 60
merged_df['fuel_consumption'] = merged_df['fuel_flow_hour'] * merged_df['duration_hours']
merged_df['fuel_consumption_total'] = merged_df['fuel_flow_hour'] * merged_df['total_hours']
merged_df['emissions_hour'] = merged_df['fuel_flow_hour'] * 3.16
merged_df['emissions_flight'] = merged_df['fuel_consumption'] * 3.16
merged_df['total_emissions'] = merged_df['fuel_consumption_total'] * 3.16
merged_df['cumulative_duration_hours'] = merged_df['duration_hours'].cumsum()

# List of column names
columns_to_keep = ['icao24', 'registration','aircraft_model','type_code','Total','fuel_flow_hour','firstseen', 'lastseen',
'src_airport_icao','dest_airport_icao','name_src','name_dest','distance','duration_hours','delivered','age','Retired','fuel_consumption',
            'fuel_consumption_total','emissions_hour','emissions_flight','total_emissions', 'cumulative_duration_hours']

# Create a new DataFrame with only the selected columns
merged_df = merged_df[columns_to_keep].copy()


# Print the new DataFra
(merged_df)

# Set display options for float formatting
pd.set_option('display.float_format', '{:.2f}'.format)
```

```
print(merged_df)


statistics = merged_df.describe()

# Print the basic statistics
(statistics)

merged_df['lastseen'] = pd.to_datetime(merged_df['lastseen'], errors='coerce')

# Group by 'lastseen' year and calculate the total number of flights
total_flights_per_year = merged_df.groupby(merged_df['lastseen'].dt.year).size().reset_index(name='Total Flights')

# Rename columns for clarity
total_flights_per_year.columns = ['Year', 'Total Flights']

# Display the result
print(total_flights_per_year)

plt.figure(figsize=(10, 6))
plt.bar(total_flights_per_year['Year'], total_flights_per_year['Total Flights'], color='skyblue')
plt.title('Total Number of Flights Over the Years')
plt.xlabel('Year')
plt.ylabel('Total Flights')
plt.grid(axis='y')
plt.show()

# cleaning data and excluding all flights from different operators

#Exclude all the rows because operated by external airline
merged_df = merged_df[merged_df['registration'].str.startswith('D-') & merged_df['registration'].notna()]
# Exclude rows with missing values in 'fuel_flow_hour'
merged_df = merged_df.dropna(subset=['fuel_flow_hour'])
# Exclude all rows with same source and destination
merged_df = merged_df[merged_df['src_airport_icao'] != merged_df['dest_airport_icao']]
merged_df

# Discriptive Statistics

# overview statistics

merged_df['lastseen'] = pd.to_datetime(merged_df['lastseen'], errors='coerce')

# Group by 'lastseen' year and calculate the total number of flights
total_flights_per_year = merged_df.groupby(merged_df['lastseen'].dt.year).size().reset_index(name='Total Flights')

total_flights_per_year.columns = ['Year', 'Total Flights']

print(total_flights_per_year)

plt.figure(figsize=(10, 6))
plt.bar(total_flights_per_year['Year'], total_flights_per_year['Total Flights'], color='skyblue')
plt.title('Total Number of Flights Over the Years')
plt.xlabel('Year')
plt.ylabel('Total Flights')
plt.grid(axis='y')
plt.show()

# Extract year and month from the 'lastseen' column
merged_df['year'] = merged_df['lastseen'].dt.year
merged_df['month'] = merged_df['lastseen'].dt.month

# Count the number of rows for each year and month
count_per_year_month = merged_df.groupby(['year', 'month']).size().reset_index(name='count')

# Calculate the average flights per month for each year
average_flights_per_year_month = count_per_year_month.groupby('year')['count'].mean().reset_index()


average_flights_per_year_month
```

```
# Extract the year from the 'lastseen' column
merged_df['year'] = merged_df['lastseen'].dt.year

# Count occurrences for each year
yearly_count = merged_df.groupby('year').size()

# Plot autocorrelation function (ACF) and partial autocorrelation function (PACF)
fig, ax = plt.subplots(2, 1, figsize=(12, 8))

# ACF plot
sm.graphics.tsa.plot_acf(yearly_count, lags=min(10, len(yearly_count)-1), ax=ax[0])
ax[0].set_title('Autocorrelation Function (ACF)')

# PACF plot
# Reduce the number of lags for PACF to avoid the ValueError
sm.graphics.tsa.plot_pacf(yearly_count, lags=min(3, len(yearly_count)//2-1), ax=ax[1], method='ywm')
ax[1].set_title('Partial Autocorrelation Function (PACF)')

plt.tight_layout()
plt.show()

# Ljung-Box test for autocorrelation
# Null hypothesis: The data are independently distributed (no autocorrelation)
# If p-value < 0.05, reject the null hypothesis
lags = min(5, len(yearly_count)-1)  # Adjusted the number of lags here
results = sm.stats.acorr_ljungbox(yearly_count, lags=lags, return_df=True)

print("Ljung-Box Test Results:")
print(results)
```

```
import statsmodels.api as sm


acf_result, confint = sm.tsa.acf(yearly_count, nlags=min(4, len(yearly_count)-1), alpha=0.05)

print("Autocorrelation Coefficients:")
print(acf_result)

print("\nConfidence Intervals:")
print(confint)


# Extract the year from the 'lastseen' column
merged_df['year'] = merged_df['lastseen'].dt.year

# Exclude years 2020 and 2021
filtered_df = merged_df[~merged_df['year'].isin([2020, 2021])]

# Count occurrences for each year (excluding 2020 and 2021)
yearly_count = filtered_df.groupby('year').size()

# Plot autocorrelation function (ACF) and partial autocorrelation function (PACF)
fig, ax = plt.subplots(2, 1, figsize=(12, 8))

# ACF plot
sm.graphics.tsa.plot_acf(yearly_count, lags=min(10, len(yearly_count)-1), ax=ax[0])
ax[0].set_title('Autocorrelation Function (ACF)')

# PACF plot
sm.graphics.tsa.plot_pacf(yearly_count, lags=min(3, len(yearly_count)//2-1), ax=ax[1], method='ywm')
ax[1].set_title('Partial Autocorrelation Function (PACF)')

plt.tight_layout()
plt.show()

# Ljung-Box test for autocorrelation
# Null hypothesis: The data are independently distributed (no autocorrelation)
# If p-value < 0.05, reject the null hypothesis
```

```
lags = min(5, len(yearly_count)-1)  # Adjusted the number of lags here
results = sm.stats.acorr_ljungbox(yearly_count, lags=lags, return_df=True)

print("Ljung-Box Test Results (excluding 2020 and 2021):")
print(results)
```

```
import statsmodels.api as sm


acf_result = sm.tsa.acf(yearly_count, nlags=min(4, len(yearly_count)-1))
print("Autocorrelation Coefficients:")
print(acf_result)
```

```
import statsmodels.api as sm


acf_result, confint = sm.tsa.acf(yearly_count, nlags=min(4, len(yearly_count)-1), alpha=0.05)

print("Autocorrelation Coefficients:")
print(acf_result)

print("\nConfidence Intervals:")
print(confint)
```

```
# Group by year and month to calculate emissions and total flights
grouped_df = merged_df.groupby(['year', 'month']).agg({'emissions_flight': 'sum', 'lastseen': 'count'}).reset_index()
grouped_df.rename(columns={'lastseen': 'total_flights'}, inplace=True)

# Calculate percentual difference against the previous month
grouped_df['emissions_diff_percent'] = grouped_df['emissions_flight'].pct_change() * 100
grouped_df['total_flights_diff_percent'] = grouped_df['total_flights'].pct_change() * 100

# Create a new column for x-axis as datetime
grouped_df['date'] = pd.to_datetime(grouped_df[['year', 'month']].assign(day=1))

# Plot the results
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 10))

# Plot emissions
grouped_df.plot(x='date', y='emissions_flight', marker='o', linestyle='-', ax=ax1, label='Emissions')
ax1.set_title('Emissions Over Time')
ax1.set_xlabel('Date')
ax1.set_ylabel('Emissions')

# Plot total flights
grouped_df.plot(x='date', y='total_flights', marker='o', linestyle='-', ax=ax2, label='Total Flights', color='orange')
ax2.set_title('Total Flights Over Time')
ax2.set_xlabel('Date')
ax2.set_ylabel('Total Flights')

plt.tight_layout()
plt.show()

# Display the DataFrame with percentual differences
print("DataFrame with Percentual Differences:")
print(grouped_df)
```

```
# Group by year and month to calculate emissions and total flights
grouped_df = merged_df.groupby(['year', 'month']).agg({'emissions_flight': 'sum', 'lastseen': 'count'}).reset_index()
grouped_df.rename(columns={'lastseen': 'total_flights'}, inplace=True)

# Calculate percentual difference against the previous month
grouped_df['emissions_diff_percent'] = grouped_df['emissions_flight'].pct_change() * 100
grouped_df['total_flights_diff_percent'] = grouped_df['total_flights'].pct_change() * 100

# Create a new column for x-axis as datetime
grouped_df['date'] = pd.to_datetime(grouped_df[['year', 'month']].assign(day=1))

# Extract data for the years 2017, 2018, 2019, and 2022
```

```
selected_years = [2017, 2018, 2019, 2022]
seasonal_data = grouped_df[grouped_df['year'].isin(selected_years)]

# Calculate average seasonal component for emissions
seasonal_decomposition_emissions = seasonal_decompose(seasonal_data['emissions_flight'], period=12)
average_seasonal_component_emissions = seasonal_decomposition_emissions.seasonal.mean()

# Calculate average seasonal component for total flights
seasonal_decomposition_flights = seasonal_decompose(seasonal_data['total_flights'], period=12)
average_seasonal_component_flights = seasonal_decomposition_flights.seasonal.mean()

# Forecast using SARIMA with the calculated seasonal component
model_emissions = SARIMAX(grouped_df['emissions_flight'], order=(1, 1, 1), seasonal_order=(1, 1, 1, 12))
fit_model_emissions = model_emissions.fit()
forecast_emissions = fit_model_emissions.get_forecast(steps=6, seasonal='add', seasonal_periods=12)
forecast_emissions_values = forecast_emissions.predicted_mean + average_seasonal_component_emissions

model_flights = SARIMAX(grouped_df['total_flights'], order=(1, 1, 1), seasonal_order=(1, 1, 1, 12))
fit_model_flights = model_flights.fit()
forecast_flights = fit_model_flights.get_forecast(steps=6, seasonal='add', seasonal_periods=12)
forecast_flights_values = forecast_flights.predicted_mean + average_seasonal_component_flights



# Create DataFrames for the forecasted values
forecast_df = pd.DataFrame({
    'date': pd.date_range(start=grouped_df['date'].max() + pd.DateOffset(months=1), periods=6, freq='MS'),
    'emissions_flight_forecast': forecast_emissions_values,
    'total_flights_forecast': forecast_flights_values
})

# Concatenate historical and forecasted data
result_df = pd.concat([grouped_df, forecast_df], ignore_index=True)

# Plot the forecast results
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 10))

# Plot emissions
ax1.plot(result_df['date'], result_df['emissions_flight'], label='Historical Emissions', marker='o')
ax1.plot(result_df['date'], result_df['emissions_flight_forecast'], label='Forecasted Emissions', linestyle='--', marker='o')
ax1.set_title('Emissions Forecast with Seasonal Component')
ax1.set_xlabel('Date')
ax1.set_ylabel('Emissions')
ax1.legend()

# Plot total flights
ax2.plot(result_df['date'], result_df['total_flights'], label='Historical Total Flights', marker='o')
ax2.plot(result_df['date'], result_df['total_flights_forecast'], label='Forecasted Total Flights', linestyle='--', marker='o')
ax2.set_title('Total Flights Forecast with Seasonal Component')
ax2.set_xlabel('Date')
ax2.set_ylabel('Total Flights')
ax2.legend()

plt.tight_layout()
plt.show()

# Display the DataFrame with historical and forecasted values
print("Historical and Forecasted Values:")
print(result_df)

# Fill missing values in 'year' column with appropriate values
result_df['year'] = result_df['year'].fillna(result_df['year'].ffill())

month_values = [7, 8, 9, 10, 11, 12]
result_df['month'] = result_df['month'].fillna(result_df.groupby('year').cumcount().mod(len(month_values)).add(7))


# Add forecast values to emissions_flight where not NaN
```

```
result_df['emissions_flight'] = result_df['emissions_flight'].add(result_df['emissions_flight_forecast'], fill_value=0)

# Add forecast values to total_flights where not NaN
result_df['total_flights'] = result_df['total_flights'].add(result_df['total_flights_forecast'], fill_value=0)

# Drop unnecessary columns
result_df = result_df.drop(['emissions_diff_percent', 'total_flights_diff_percent', 'date',
                            'emissions_flight_forecast', 'total_flights_forecast'], axis=1)

# Print the updated DataFrame
print(result_df)

result_df = result_df.groupby('year').agg({
    'emissions_flight': 'sum',
    'total_flights': 'sum'
}).reset_index()

print(result_df)

# Group by year to calculate total flights per year
total_flights_per_year = result_df.groupby('year')['total_flights'].sum().reset_index()
total_emissions_per_year = result_df.groupby('year')['emissions_flight'].sum().reset_index()

# Plot the total flights per year
plt.figure(figsize=(10, 6))
plt.bar(total_flights_per_year['year'], total_flights_per_year['total_flights'], color='lightgreen')
plt.title('Total Flights Per Year')
plt.xlabel('Year')
plt.ylabel('Total Flights')
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()

# Plot the total emissions per year
plt.figure(figsize=(10, 6))
plt.bar(total_emissions_per_year['year'], total_emissions_per_year['emissions_flight'], color='lightgreen')
plt.title('Total Emissions Per Year')
plt.xlabel('Year')
plt.ylabel('Total Flights')
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()

result_df = result_df.sort_values('year')

# Calculate growth rates
result_df['Emissions Growth Rate'] = result_df['emissions_flight'].pct_change() * 100
result_df['Flights Growth Rate'] = result_df['total_flights'].pct_change() * 100

# Display the table
table = result_df[['year', 'emissions_flight', 'Emissions Growth Rate', 'total_flights', 'Flights Growth Rate']]
table = table.rename(columns={'emissions_flight': 'Emissions', 'total_flights': 'Total Flights'})
(table)

## average fuel consumption per year

# Group by year to calculate average fuel consumption per year
average_fuel_consumption_per_year = merged_df.groupby('year')['fuel_consumption'].mean().reset_index()

# Plot the average fuel consumption per year
plt.figure(figsize=(10, 6))
plt.bar(average_fuel_consumption_per_year['year'], average_fuel_consumption_per_year['fuel_consumption'], color='skyblue')
plt.title('Average Fuel Consumption Per Year')
plt.xlabel('Year')
plt.ylabel('Average Fuel Consumption')
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()

# percentage of total emissions per aircraft type
```

```
merged_df['age'] = pd.to_numeric(merged_df['age'], errors='coerce')
merged_df['total_fuel_ratio'] = merged_df['fuel_flow_hour'] / merged_df['Total']

# Filter routes in the year 2022
routes_2022 = merged_df[merged_df['lastseen'].dt.year == 2022]

# Group by 'type_code' and calculate total emissions, average age, average fuel flow per hour, and total number of flights
summary_per_model = routes_2022.groupby('type_code').agg(
    total_emissions=('emissions_flight', 'sum'),
    average_age=('age', 'mean'),
    total_fuel_ratio=('total_fuel_ratio', 'mean'),  # Average fuel flow per hour
    total_flights=('lastseen', 'count'),  # Count the number of flights
).reset_index()

# Calculate the total emissions, average fuel flow per hour, and number of flights for all routes in 2022
total_emissions_2022 = summary_per_model['total_emissions'].sum()
total_fuel_flow_hour_2022 = (routes_2022['fuel_flow_hour'] * routes_2022['duration_hours']).sum()  # Calculate total fuel flow hours
total_flights_2022 = summary_per_model['total_flights'].sum()

# Calculate the percentage of emissions, average fuel flow per hour, and flights for each aircraft model
summary_per_model['percentage_of_emissions'] = (
    summary_per_model['total_emissions'] / total_emissions_2022 * 100

)
summary_per_model['percentage_of_flights'] = (
    summary_per_model['total_flights'] / total_flights_2022 * 100
)

# Display the result
(summary_per_model)

plt.figure(figsize=(10, 6))
plt.bar(summary_per_model['type_code'], summary_per_model['percentage_of_emissions'], color='skyblue')
plt.xlabel('Aircraft Type Code')
plt.ylabel('Percentage of Total Emissions (%)')
plt.title('Percentage of Total Emissions by Aircraft Type in 2022')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better visibility
plt.tight_layout()

# Show the plot
plt.show()

#Percentage of In country (germany) flights

# Calculate and display percentages for 'ED'  per year
unique_years = sorted(merged_df['year'].unique())

# Display the information in a table using PrettyTable
table = PrettyTable()
table.field_names = ['Year', 'Letter', 'Percentage of Routes', 'Percentage of Emissions']

for year in unique_years:
    year_df = merged_df[merged_df['year'] == year]

    total_rows = len(year_df)
    total_emissions = year_df['emissions_flight'].sum()

    # Calculate the percentage for 'ED'
    ed_routes = year_df[
        (year_df['src_airport_icao'].str.startswith('ED')) &
        (year_df['dest_airport_icao'].str.startswith('ED'))
    ]
    percentage_ed_routes = len(ed_routes) / len(merged_df) * 100
    percentage_ed_emissions = ed_routes['emissions_flight'].sum() / total_emissions * 100

    # Add rows to the table
    table.add_row([year, 'ED', f"{percentage_ed_routes:.2f}%", f"{percentage_ed_emissions:.2f}%"])
```

```
print(table)

# percentage of shourt and longhaul flights

# Calculate and display percentages for short-haul, medium-haul, and long-haul flights per year
unique_years = sorted(merged_df['year'].unique())

# Display the information in a table using PrettyTable
table = PrettyTable()
table.field_names = ['Year', 'Short Haul (<1500 km) Flights', 'Medium Haul (1500-4000 km) Flights', 'Long Haul (>4000 km) Flights', 'Short Haul (<1500 km) Emissions', 'Medium Haul (1500-4000 km) Emissions', 'Long Haul (>4000 km) Emissions']

for year in unique_years:
    year_df = merged_df[merged_df['year'] == year]

    total_flights = len(year_df)
    total_emissions = year_df['emissions_flight'].sum()

    # Calculate the percentage for short-haul flights (<1500 km)
    short_haul_flights = year_df[year_df['distance'] < 1500]
    percentage_short_haul_flights = len(short_haul_flights) / total_flights * 100
    percentage_short_haul_emissions = short_haul_flights['emissions_flight'].sum() / total_emissions * 100

    # Calculate the percentage for medium-haul flights (1500-4000 km)
    medium_haul_flights = year_df[(year_df['distance'] >= 1500) & (year_df['distance'] <= 4000)]
    percentage_medium_haul_flights = len(medium_haul_flights) / total_flights * 100
    percentage_medium_haul_emissions = medium_haul_flights['emissions_flight'].sum() / total_emissions * 100

    # Calculate the percentage for long-haul flights (>4000 km)
    long_haul_flights = year_df[year_df['distance'] > 4000]
    percentage_long_haul_flights = len(long_haul_flights) / total_flights * 100
    percentage_long_haul_emissions = long_haul_flights['emissions_flight'].sum() / total_emissions * 100

    # Add rows to the table
    table.add_row([year, f"{percentage_short_haul_flights:.2f}%", f"{percentage_medium_haul_flights:.2f}%", f"{percentage_long_haul_flights:.2f}%",
f"{percentage_short_haul_emissions:.2f}%", f"{percentage_medium_haul_emissions:.2f}%", f"{percentage_long_haul_emissions:.2f}%"])

(table)

# percentage of total emissions and total flights by aircraft type

# Filter data for the year 2022
merged_df_2022 = merged_df[merged_df['year'] == 2022]

# Display the information in a table using PrettyTable
table = PrettyTable()
table.field_names = ['Type Code', 'Percentage of Total Emissions', 'Percentage of Total Flights', 'Percentage of Total Distance', 'Average Age']

for type_code in merged_df_2022['type_code'].unique():
    type_code_df = merged_df_2022[merged_df_2022['type_code'] == type_code]

    total_flights = len(type_code_df)
    total_emissions = type_code_df['emissions_flight'].sum()
    total_distance = type_code_df['distance'].sum()
    average_age = type_code_df['age'].mean()

    # Calculate percentages for type_code
    percentage_type_code_flights = total_flights / len(merged_df_2022) * 100
    percentage_type_code_emissions = total_emissions / merged_df_2022['emissions_flight'].sum() * 100
    percentage_type_code_distance = total_distance / merged_df_2022['distance'].sum() * 100

    # Add rows to the table
    table.add_row([type_code, f"{percentage_type_code_emissions:.2f}%", f"{percentage_type_code_flights:.2f}%", f"{percentage_type_code_distance:.2f}%", f"{average_age:.2f}"])

print(table)

# total passenger capacity and fuel flow ratio to assess efficiency per aircraft type
```

```
# drop NaN values
def custom_mean_age(values):
    non_nan_values = values.dropna()
    if len(non_nan_values) > 0:
        return np.mean(non_nan_values)
    else:
        return np.nan

# Rank the unique 'type_code' values by the 'fuel_flow_hour', 'emissions_hour', 'Total', and 'age' columns
unique_type_ranked_df = merged_df.groupby('type_code').agg({
    'fuel_flow_hour': 'first',
    'emissions_hour': 'first',
    'Total': 'first',
    'age': custom_mean_age,
    'aircraft_model': 'first'
}).reset_index()

# Calculate the ratio between 'Total' and 'fuel_flow_hour' and sort by total/fuel ratio
unique_type_ranked_df['total_fuel_ratio'] = unique_type_ranked_df['fuel_flow_hour'] / unique_type_ranked_df['Total']
sorted_unique_types = unique_type_ranked_df.sort_values(by='total_fuel_ratio')


print(sorted_unique_types[['type_code', 'aircraft_model', 'fuel_flow_hour', 'Total', 'total_fuel_ratio', 'age']])


plt.figure(figsize=(30, 10))
plt.scatter(sorted_unique_types['type_code'], sorted_unique_types['total_fuel_ratio'], color='purple', label='Total/Fuel Flow Ratio')

for i, txt in enumerate(sorted_unique_types['type_code']):
    plt.annotate(f"{txt}\nModel: {sorted_unique_types['aircraft_model'].iloc[i]}\nAge: {sorted_unique_types['age'].iloc[i]:.2f}",
                 (sorted_unique_types['type_code'].iloc[i], sorted_unique_types['total_fuel_ratio'].iloc[i]),
                 ha='center', va='bottom')

plt.title('Fuel flow per hour per Passenger and Average Age for Unique Aircraft Types')
plt.xlabel('Unique Aircraft Type Code')
plt.ylabel('Fuel flow per hour per Passenger')

plt.legend(loc='lower right')
plt.show()

#liter per 100pkm

# Calculate fuel consumption per passenger kilometer
merged_df['fuel_per_passenger_km'] = merged_df['fuel_flow_hour'] / (merged_df['Total'] * merged_df['distance'])

# Convert fuel consumption to liters per 100 passenger kilometers
merged_df['fuel_per_100_passenger_km'] = merged_df['fuel_per_passenger_km'] * 100

# Replace inf values with NaN
merged_df.replace([np.inf, -np.inf], np.nan, inplace=True)

# Drop rows with NaN values in the ratio column
merged_df.dropna(subset=['fuel_per_100_passenger_km'], inplace=True)

# Calculate the overall mean for the ratio
overall_average_ratio = merged_df['fuel_per_100_passenger_km'].mean()

# Display the result
print("Overall Average Ratio for all type_codes (NaN and inf handled):", overall_average_ratio)


# Load factors
load_factors = {
    2018: 0.814,
    2019: 0.825,
    2020: 0.621,
    2021: 0.603,
    2022: 0.798,
```

```
    2023: 0.829
}

# Step 1: Calculate adjusted total number of passengers using load factors
merged_df['adjusted_total'] = merged_df.apply(lambda row: row['Total'] * load_factors.get(row['year'], 1), axis=1)

# Calculate fuel consumption per passenger kilometer using adjusted total
merged_df['fuel_per_passenger_km'] = merged_df['fuel_consumption'] / (merged_df['adjusted_total'] * merged_df['distance'])

# Convert fuel consumption to liters per 100 passenger kilometers
merged_df['fuel_per_100_passenger_km'] = merged_df['fuel_per_passenger_km'] * 100

# Group by 'type_code' and calculate the mean for the ratio and average age
grouped_df = merged_df.groupby('type_code').agg(
    fuel_per_100_passenger_km_mean=('fuel_per_100_passenger_km', 'mean'),
    average_age=('age', 'mean')
).reset_index()

# If you want the result as a DataFrame
pkm_age_df = grouped_df.sort_values(by='fuel_per_100_passenger_km_mean')

# Rank the results based on the calculated ratio from lowest to highest
pkm_age_df['Rank'] = pkm_age_df['fuel_per_100_passenger_km_mean'].rank()

# Display the ranked result with average age
print(pkm_age_df[['type_code', 'fuel_per_100_passenger_km_mean', 'average_age']])



# Load factors
load_factors = {
    2018: 0.814,
    2019: 0.825,
    2020: 0.621,
    2021: 0.603,
    2022: 0.798,
    2023: 0.829
}

# Step 1: Calculate adjusted total number of passengers using load factors
merged_df['adjusted_total'] = merged_df.apply(lambda row: row['Total'] * load_factors.get(row['year'], 1), axis=1)

# Step 2: Calculate distance * adjusted_total
merged_df['distance_adjusted_total'] = merged_df['distance'] * merged_df['adjusted_total']

# Step 3: Calculate liters per 100 passenger kilometers using adjusted total
merged_df['liters_per_100pkm'] = (merged_df['fuel_consumption'] / merged_df['distance_adjusted_total']) * 100

# Step 4: Group by year and calculate the mean liters per 100 passenger kilometers for each year
result_table = merged_df.groupby('year')['liters_per_100pkm'].mean().reset_index()

# Display the result table
print(result_table)

# Create a line plot
plt.figure(figsize=(10, 6))
sns.lineplot(x='year', y='liters_per_100pkm', data=result_table, marker='o')
plt.title('Average Liters per 100 Passenger Kilometers Over the Years')
plt.xlabel('Year')
plt.ylabel('Liters per 100 Passenger Kilometers')

# Show the line plot
plt.show()

#Growth forecast according to Waypoint 2050 RPK forecasts 2015


# Group by year and calculate total emissions
total_emissions_per_year = result_df.groupby(result_df['year'])['emissions_flight'].sum().reset_index()
```

```
# Define growth rates for different scenarios
growth_rates_scenarios = {
    'Scenario 1': {2024: 0.006, 2030: 0.028, 2040: 0.023},
    'Scenario 2': {2024: 0.031, 2030: 0.032, 2040: 0.031},
    'Scenario 3': {2024: 0.037, 2030: 0.034, 2040: 0.033}
}

# Plotting for each scenario
fig, ax = plt.subplots(figsize=(12, 8))

for scenario, growth_rates in growth_rates_scenarios.items():
    total_emissions_scenario = total_emissions_per_year.copy()

    for year in range(2024, 2051):
        if year <= 2030:
            growth_rate = growth_rates.get(2024, 0)
        elif year <= 2040:
            growth_rate = growth_rates.get(2030, 0)
        else:
            growth_rate = growth_rates.get(2040, 0)

        # Forecast total emissions for the current year
        total_emissions_scenario = pd.concat([total_emissions_scenario, pd.DataFrame({'year': [year], 'emissions_flight': [total_emissions_scenario['emissions_flight'].iloc[-1] * (1 +
growth_rate)]})], ignore_index=True)

    ax.plot(total_emissions_scenario['year'], total_emissions_scenario['emissions_flight'], marker='o', linestyle='-', label=scenario)

ax.set_xlabel('Year')
ax.set_ylabel('Total Emissions')
ax.set_title('Total Emissions per Year Over the Timespan (Scenarios)')
ax.legend()
plt.tight_layout()
plt.show()


total_emissions_per_year = result_df.groupby(result_df['year'])['emissions_flight'].sum().reset_index()

# Define growth rates for different scenarios
growth_rates_scenarios = {
    'Scenario 1': {2024: 0.006, 2030: 0.028, 2040: 0.023},
    'Scenario 2': {2024: 0.031, 2030: 0.032, 2040: 0.031},
    'Scenario 3': {2024: 0.037, 2030: 0.034, 2040: 0.033}
}

# Create a list to store DataFrames for each scenario
frames = []

# Populate the forecast_table
for scenario, growth_rates in growth_rates_scenarios.items():
    total_emissions_scenario = total_emissions_per_year.copy()

    for year in range(2024, 2051):
        if year <= 2030:
            growth_rate = growth_rates.get(2024, 0)
        elif year <= 2040:
            growth_rate = growth_rates.get(2030, 0)
        else:
            growth_rate = growth_rates.get(2040, 0)

        # Forecast total emissions for the current year
        forecasted_emissions = total_emissions_scenario['emissions_flight'].iloc[-1] * (1 + growth_rate)

        # Create a DataFrame for the current scenario and year
        df = pd.DataFrame({'Scenario': [scenario], 'Year': [year], 'Forecasted Emissions': [forecasted_emissions]})

        # Append the DataFrame to the list
        frames.append(df)
```

```
        # Update total_emissions_scenario for the next iteration
        total_emissions_scenario = pd.concat([total_emissions_scenario, pd.DataFrame({'year': [year], 'emissions_flight': [forecasted_emissions]})], ignore_index=True)

# Set display options to show all rows
pd.set_option('display.max_rows', None)

# Concatenate all DataFrames in the list into the final forecast_table
forecast_table = pd.concat(frames, ignore_index=True)

# Display the forecast_table
print(forecast_table)

# Calculate the sum of Forecasted Emissions for each scenario in forecast_table
sum_forecasted_emissions_by_scenario = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum()

print("Sum of Forecasted Emissions by Scenario for forecast_table:")
print(sum_forecasted_emissions_by_scenario)



# Create a list to store DataFrames for each scenario
frames = []

# Create a dictionary to store the values for Scenario 1
scenario_1_values = {}

# Populate the forecast_table
for scenario, growth_rates in growth_rates_scenarios.items():
    total_emissions_scenario = total_emissions_per_year.copy()

    for year in range(2024, 2051):
        if year <= 2030:
            growth_rate = growth_rates.get(2024, 0)
        elif year <= 2040:
            growth_rate = growth_rates.get(2030, 0)
        else:
            growth_rate = growth_rates.get(2040, 0)

        # Forecast total emissions for the current year
        forecasted_emissions = total_emissions_scenario['emissions_flight'].iloc[-1] * (1 + growth_rate)

        # Create a DataFrame for the current scenario and year
        df = pd.DataFrame({'Scenario': [scenario], 'Year': [year], 'Forecasted Emissions': [forecasted_emissions]})

        # Append the DataFrame to the list
        frames.append(df)

        # Update total_emissions_scenario for the next iteration
        total_emissions_scenario = pd.concat([total_emissions_scenario, pd.DataFrame({'lastseen': [pd.to_datetime(f'{year}-01-01')], 'emissions_flight': [forecasted_emissions]})],
ignore_index=True)

        # Store values for Scenario 1
        if scenario == 'Scenario 1':
            scenario_1_values[year] = forecasted_emissions

# Set display options to show all rows
pd.set_option('display.max_rows', None)

# Concatenate all DataFrames in the list into the final forecast_table
forecast_table = pd.concat(frames, ignore_index=True)

# Calculate the difference in percent compared to Scenario 1
forecast_table['Difference %'] = 100 * (forecast_table['Forecasted Emissions'] / forecast_table['Year'].map(scenario_1_values) - 1)

# Filter the table for the years 2030, 2040, and 2050
result_table = forecast_table[forecast_table['Year'].isin([2030, 2040, 2050])]
```

```
# Display the result_table
print(result_table)

# fleet change scenrios


type_code_values = {
    'A321': 2874, 'E190': 1970, 'A319': 2374, 'A20N': 2600,
    'A21N': 2600, 'A359': 5800, 'B787': 5600, 'A388': 11500,
    'CRJ9': 1600, 'B789': 5600, 'B77L': 6800
}

# Adjust fuel_flow_hour for specified type codes
for type_code, fuel_flow_value in type_code_values.items():
    copied_df.loc[copied_df['type_code'] == type_code, 'fuel_flow_hour'] = fuel_flow_value

copied_df['duration'] = pd.to_numeric(copied_df['duration'], errors='coerce')

copied_df['duration_hours'] = copied_df['duration'] / 60
copied_df['fuel_consumption'] = copied_df['fuel_flow_hour'] * copied_df['duration_hours']
copied_df['fuel_consumption_total'] = copied_df['fuel_flow_hour'] * copied_df['total_hours']
copied_df['emissions_hour'] = copied_df['fuel_flow_hour'] * 3.16
copied_df['emissions_flight'] = copied_df['fuel_consumption'] * 3.16
copied_df['total_emissions'] = copied_df['fuel_consumption_total'] * 3.16
copied_df['cumulative_duration_hours'] = copied_df['duration_hours'].cumsum()

copied_df['lastseen'] = pd.to_datetime(copied_df['lastseen'], errors='coerce')

# Group by 'lastseen' year and calculate the total number of flights
total_flights_per_year = copied_df.groupby(merged_df['lastseen'].dt.year).size().reset_index(name='Total Flights')

# Rename columns for clarity
total_flights_per_year.columns = ['Year', 'Total Flights']

# Display the result
print(total_flights_per_year)

plt.figure(figsize=(10, 6))
plt.bar(total_flights_per_year['Year'], total_flights_per_year['Total Flights'], color='skyblue')
plt.title('Total Number of Flights Over the Years')
plt.xlabel('Year')
plt.ylabel('Total Flights')
plt.grid(axis='y')
plt.show()

# Extract emissions for the year 2029 for each scenario
emissions_2029 = forecast_table[(forecast_table['Year'] <= 2029)]

# Display the sum of emissions for each scenario until 2029
sum_emissions_2029 = emissions_2029.groupby('Scenario')['Forecasted Emissions'].sum()
print(sum_emissions_2029)



# Convert the 'lastseen' column to datetime format
merged_df['lastseen'] = pd.to_datetime(merged_df['lastseen'])

# Filter the DataFrame for the specified type_codes
type_codes_of_interest = ['A320', 'A343', 'A346', 'A333', 'B744', 'B748']
filtered_df = merged_df[merged_df['type_code'].isin(type_codes_of_interest)]

# Get unique icao24 values
unique_icao24_list = filtered_df['icao24'].unique()

# Display the result
print("List of unique icao24 values with specified type_codes:", unique_icao24_list)

#Continous forecast until 2029 fleet change


# Group by year and calculate total emissions
```

```
total_emissions_per_year = result_df.groupby(result_df['year'])['emissions_flight'].sum().reset_index()

# Define growth rates for different scenarios
growth_rates_scenarios = {
    'Scenario 1': {2024: 0.006, 2030: 0.028, 2040: 0.023},
    'Scenario 2': {2024: 0.031, 2030: 0.032, 2040: 0.031},
    'Scenario 3': {2024: 0.037, 2030: 0.034, 2040: 0.033}
}

# Create a list to store DataFrames for each scenario
frames = []



# Create new scenarios with fleet change and subtract 2.31% from forecasted values between 2024 and 2029
for i in range(1, 4):
    scenario_name = f'Scenario {i}'
    growth_rates = {2024: 0.006, 2030: 0.028, 2040: 0.023}

    if i == 2:
        growth_rates = {2024: 0.031, 2030: 0.032, 2040: 0.031}
    elif i == 3:
        growth_rates = {2024: 0.037, 2030: 0.034, 2040: 0.033}

    total_emissions_scenario = total_emissions_per_year.copy()

    for year in range(2024, 2030):
        growth_rate = growth_rates.get(2024, 0)
        forecasted_emissions = total_emissions_scenario['emissions_flight'].iloc[-1] * (1 + growth_rate) * (1 - 0.0231)

        df = pd.DataFrame({'Scenario': [scenario_name], 'Year': [year], 'Forecasted Emissions': [forecasted_emissions]})
        frames.append(df)
        total_emissions_scenario = pd.concat([total_emissions_scenario, pd.DataFrame({'year': [year], 'emissions_flight': [forecasted_emissions]})], ignore_index=True)

    for year in range(2030, 2051):
        if year <= 2040:
            growth_rate = growth_rates.get(2030, 0)
        else:
            growth_rate = growth_rates.get(2040, 0)

        forecasted_emissions = total_emissions_scenario['emissions_flight'].iloc[-1] * (1 + growth_rate)

        df = pd.DataFrame({'Scenario': [scenario_name], 'Year': [year], 'Forecasted Emissions': [forecasted_emissions]})
        frames.append(df)
        total_emissions_scenario = pd.concat([total_emissions_scenario, pd.DataFrame({'year': [year], 'emissions_flight': [forecasted_emissions]})], ignore_index=True)

# Set display options to show all rows
pd.set_option('display.max_rows', None)

# Concatenate all DataFrames in the list into the final forecast_table
forecast_table_continuous = pd.concat(frames, ignore_index=True)

# Display the forecast_table
print(forecast_table_continuous)



# Plotting
plt.figure(figsize=(25, 12))


# Plot forecasted values from forecast_table
for scenario in forecast_table['Scenario'].unique():
    scenario_data = forecast_table[forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data['Year'], scenario_data['Forecasted Emissions'], marker='s', linestyle='-', label=f'{scenario} (Original)')

# Plot forecasted values from copied_forecast_table
for scenario in forecast_table_continuous['Scenario'].unique():
```

```
    scenario_data = forecast_table_continuous[forecast_table_continuous['Scenario'] == scenario]
    plt.plot(scenario_data['Year'], scenario_data['Forecasted Emissions'], marker='o', linestyle='-', label=f'{scenario} (fleet change)')

plt.title('Comparison of Forecasted Emissions (Original vs Countinuous fleet change)')
plt.xlabel('Year')
plt.ylabel('Forecasted Emissions')
plt.legend()
plt.grid(True)
plt.show()



# Convert 'year' column to datetime object
result_df['year'] = pd.to_datetime(result_df['year'], format='%Y', errors='coerce')

# Group by year and calculate total emissions
total_emissions_per_year = result_df.groupby(result_df['year'].dt.year)['emissions_flight'].sum().reset_index()

# Define growth rates for different scenarios
growth_rates_scenarios = {
    'Scenario 1': {2024: 0.006, 2030: 0.028, 2040: 0.023},
    'Scenario 2': {2024: 0.031, 2030: 0.032, 2040: 0.031},
    'Scenario 3': {2024: 0.037, 2030: 0.034, 2040: 0.033}
}

# Create a list to store DataFrames for each scenario
frames = []

# Populate the forecast_table
for scenario, growth_rates in growth_rates_scenarios.items():
    total_emissions_scenario = total_emissions_per_year.copy()

    for year in range(2024, 2051):
        if year <= 2030:
            growth_rate = growth_rates.get(2024, 0)
        elif year <= 2040:
            growth_rate = growth_rates.get(2030, 0)
        else:
            growth_rate = growth_rates.get(2040, 0)

        # Forecast total emissions for the current year
        forecasted_emissions = total_emissions_scenario['emissions_flight'].iloc[-1] * (1 + growth_rate)

        # Create a DataFrame for the current scenario and year
        df = pd.DataFrame({'Scenario': [scenario], 'Year': [year], 'Forecasted Emissions': [forecasted_emissions]})

        # Append the DataFrame to the list
        frames.append(df)

        # Update total_emissions_scenario for the next iteration
        total_emissions_scenario = pd.concat([total_emissions_scenario, pd.DataFrame({'year': [year], 'emissions_flight': [forecasted_emissions]})], ignore_index=True)

# Create new scenarios with fleet change and subtract 2.31% from forecasted values between 2024 and 2029
for scenario, growth_rates in growth_rates_scenarios.items():
    for i in range(1, 4):
        scenario_name = f'{scenario} - Fleet Change'
        total_emissions_scenario = total_emissions_per_year.copy()

        for year in range(2024, 2030):
            growth_rate = growth_rates.get(2024, 0)
            forecasted_emissions = total_emissions_scenario['emissions_flight'].iloc[-1] * (1 + growth_rate) * (1 - 0.0231)

            df = pd.DataFrame({'Scenario': [scenario_name], 'Year': [year], 'Forecasted Emissions': [forecasted_emissions]})
            frames.append(df)
            total_emissions_scenario = pd.concat([total_emissions_scenario, pd.DataFrame({'year': [year], 'emissions_flight': [forecasted_emissions]})], ignore_index=True)

        for year in range(2030, 2051):
            if year <= 2040:
```

```
            growth_rate = growth_rates.get(2030, 0)
        else:
            growth_rate = growth_rates.get(2040, 0)

        forecasted_emissions = total_emissions_scenario['emissions_flight'].iloc[-1] * (1 + growth_rate)

        df = pd.DataFrame({'Scenario': [scenario_name], 'Year': [year], 'Forecasted Emissions': [forecasted_emissions]})
        frames.append(df)
        total_emissions_scenario = pd.concat([total_emissions_scenario, pd.DataFrame({'year': [year], 'emissions_flight': [forecasted_emissions]})], ignore_index=True)
# Set display options to show all rows
pd.set_option('display.max_rows', None)

# Concatenate all DataFrames in the list into the final forecast_table
forecast_table_24 = pd.concat(frames, ignore_index=True)

# Plotting
plt.figure(figsize=(12, 8))

# Iterate through scenarios and plot
for scenario in forecast_table_24['Scenario'].unique():
    scenario_data = forecast_table_24[forecast_table_24['Scenario'] == scenario]
    marker = 'o' if 'Fleet Change' in scenario else 's'
    plt.plot(scenario_data['Year'], scenario_data['Forecasted Emissions'], marker=marker, linestyle='-', label=scenario)


plt.title('Forecasted Emissions for Different Scenarios')
plt.xlabel('Year')
plt.ylabel('Forecasted Emissions')
plt.legend()
plt.grid(True)
plt.show()


#fleet change scenario changing ordered fleet in 2024



copied_df['lastseen'] = pd.to_datetime(copied_df['lastseen'])

# Extract the year and month from the 'lastseen' column
copied_df['year'] = copied_df['lastseen'].dt.year.astype('Int64')  # Nullable integer type
copied_df['month'] = copied_df['lastseen'].dt.month

# Group by year and month to calculate emissions and total flights
copied_grouped_df = copied_df.groupby(['year', 'month']).agg({'emissions_flight': 'sum', 'lastseen': 'count'}).reset_index()
copied_grouped_df.rename(columns={'lastseen': 'total_flights'}, inplace=True)

# Calculate percentual difference against the previous month
copied_grouped_df['emissions_diff_percent'] = copied_grouped_df['emissions_flight'].pct_change() * 100
copied_grouped_df['total_flights_diff_percent'] = copied_grouped_df['total_flights'].pct_change() * 100

# Create a new column for x-axis as datetime
copied_grouped_df['date'] = pd.to_datetime(copied_grouped_df[['year', 'month']].assign(day=1))

# Extract data for the years 2017, 2018, 2019, and 2022
selected_years = [2017, 2018, 2019, 2022]
copied_seasonal_data = copied_grouped_df[copied_grouped_df['year'].isin(selected_years)]

# Calculate average seasonal component for emissions
copied_seasonal_decomposition_emissions = seasonal_decompose(copied_seasonal_data['emissions_flight'], period=12)
copied_average_seasonal_component_emissions = copied_seasonal_decomposition_emissions.seasonal.mean()

# Calculate average seasonal component for total flights
copied_seasonal_decomposition_flights = seasonal_decompose(copied_seasonal_data['total_flights'], period=12)
copied_average_seasonal_component_flights = copied_seasonal_decomposition_flights.seasonal.mean()

# Forecast using SARIMA with the calculated seasonal component
copied_model_emissions = SARIMAX(copied_grouped_df['emissions_flight'], order=(1, 1, 1), seasonal_order=(1, 1, 1, 12))
```

```
copied_fit_model_emissions = copied_model_emissions.fit()
copied_forecast_emissions = copied_fit_model_emissions.get_forecast(steps=6, seasonal='add', seasonal_periods=12)
copied_forecast_emissions_values = copied_forecast_emissions.predicted_mean + copied_average_seasonal_component_emissions

copied_model_flights = SARIMAX(copied_grouped_df['total_flights'], order=(1, 1, 1), seasonal_order=(1, 1, 1, 12))
copied_fit_model_flights = copied_model_flights.fit()
copied_forecast_flights = copied_fit_model_flights.get_forecast(steps=6, seasonal='add', seasonal_periods=12)
copied_forecast_flights_values = copied_forecast_flights.predicted_mean + copied_average_seasonal_component_flights

# Create DataFrames for the forecasted values
copied_forecast_df = pd.DataFrame({
    'date': pd.date_range(start=copied_grouped_df['date'].max() + pd.DateOffset(months=1), periods=6, freq='MS'),
    'emissions_flight_forecast': copied_forecast_emissions_values,
    'total_flights_forecast': copied_forecast_flights_values
})

# Concatenate historical and forecasted data
copied_result_df = pd.concat([copied_grouped_df, copied_forecast_df], ignore_index=True)

# Plot the forecast results
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 10))

# Plot emissions
ax1.plot(copied_result_df['date'], copied_result_df['emissions_flight'], label='Historical Emissions', marker='o')
ax1.plot(copied_result_df['date'], copied_result_df['emissions_flight_forecast'], label='Forecasted Emissions', linestyle='--', marker='o')
ax1.set_title('Emissions Forecast with Seasonal Component')
ax1.set_xlabel('Date')
ax1.set_ylabel('Emissions')
ax1.legend()

# Plot total flights
ax2.plot(copied_result_df['date'], copied_result_df['total_flights'], label='Historical Total Flights', marker='o')
ax2.plot(copied_result_df['date'], copied_result_df['total_flights_forecast'], label='Forecasted Total Flights', linestyle='--', marker='o')
ax2.set_title('Total Flights Forecast with Seasonal Component')
ax2.set_xlabel('Date')
ax2.set_ylabel('Total Flights')
ax2.legend()

plt.tight_layout()
plt.show()

# Display the DataFrame with historical and forecasted values
print("Historical and Forecasted Values:")
print(copied_result_df)

# Fill missing values in 'year' column with appropriate values
copied_result_df['year'] = copied_result_df['year'].fillna(copied_result_df['year'].ffill())

month_values = [7, 8, 9, 10, 11, 12]
copied_result_df['month'] = copied_result_df['month'].fillna(copied_result_df.groupby('year').cumcount().mod(len(month_values)).add(7))

# Add forecast values to emissions_flight where not NaN
copied_result_df['emissions_flight'] = copied_result_df['emissions_flight'].add(copied_result_df['emissions_flight_forecast'], fill_value=0)

# Add forecast values to total_flights where not NaN
copied_result_df['total_flights'] = copied_result_df['total_flights'].add(copied_result_df['total_flights_forecast'], fill_value=0)

# Drop unnecessary columns
copied_result_df = copied_result_df.drop(['emissions_diff_percent', 'total_flights_diff_percent', 'date',
                                          'emissions_flight_forecast', 'total_flights_forecast'], axis=1)

# Print the updated DataFrame
print(copied_result_df)


copied_result_df['year'] = copied_result_df['year'].astype(str).str[-4:].astype(int)

copied_result_df
```

```
copied_total_emissions_per_year = copied_result_df.groupby('year')['emissions_flight'].sum().reset_index()

# Display the result
print(copied_total_emissions_per_year)

total_emissions_per_year = result_df.groupby('year')['emissions_flight'].sum().reset_index()

# Display the result
print(total_emissions_per_year)

copied_total_emissions_per_year = copied_result_df.groupby(copied_result_df['year'])['emissions_flight'].sum().reset_index()

# Define growth rates for different scenarios
growth_rates_scenarios = {
    'Scenario 1': {2024: 0.006, 2030: 0.028, 2040: 0.023},
    'Scenario 2': {2024: 0.031, 2030: 0.032, 2040: 0.031},
    'Scenario 3': {2024: 0.037, 2030: 0.034, 2040: 0.033}
}

# Create a list to store DataFrames for each scenario
frames = []

# Populate the forecast_table
for scenario, growth_rates in growth_rates_scenarios.items():
    total_emissions_scenario = copied_total_emissions_per_year.copy()

    for year in range(2024, 2051):
        if year <= 2030:
            growth_rate = growth_rates.get(2024, 0)
        elif year <= 2040:
            growth_rate = growth_rates.get(2030, 0)
        else:
            growth_rate = growth_rates.get(2040, 0)

        # Forecast total emissions for the current year
        forecasted_emissions = total_emissions_scenario['emissions_flight'].iloc[-1] * (1 + growth_rate)

        # Create a DataFrame for the current scenario and year
        df = pd.DataFrame({'Scenario': [scenario], 'Year': [year], 'Forecasted Emissions': [forecasted_emissions]})

        # Append the DataFrame to the list
        frames.append(df)

        # Update total_emissions_scenario for the next iteration
        total_emissions_scenario = pd.concat([total_emissions_scenario, pd.DataFrame({'year': [year], 'emissions_flight': [forecasted_emissions]})], ignore_index=True)

# Set display options to show all rows
pd.set_option('display.max_rows', None)

# Concatenate all DataFrames in the list into the final forecast_table
copied_forecast_table = pd.concat(frames, ignore_index=True)

# Display the forecast_table
print(copied_forecast_table)



# Sum of emissions_flight for each scenario
sum_emissions_original = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum()
sum_emissions_copied = copied_forecast_table.groupby('Scenario')['Forecasted Emissions'].sum()

# Display the sums and the difference
result_summary = pd.DataFrame({
    'Scenario': sum_emissions_original.index,
    'Sum Emissions (Original)': sum_emissions_original.values,
    'Sum Emissions (Copied)': sum_emissions_copied.values,
    'Difference': sum_emissions_original.values - sum_emissions_copied.values,
    'Percentage Difference': ((sum_emissions_original.values - sum_emissions_copied.values) / sum_emissions_original.values) * 100
})
```

```
print(result_summary)


# Convert 'year' column to datetime object
total_emissions_per_year['year'] = pd.to_datetime(total_emissions_per_year['year'], errors='coerce')

# Plotting
plt.figure(figsize=(12, 8))


# Plot forecasted values from forecast_table
for scenario in forecast_table['Scenario'].unique():
    scenario_data = forecast_table[forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data['Year'], scenario_data['Forecasted Emissions'], marker='o', linestyle='-', label=scenario)

# Plot forecasted values from copied_forecast_table
for scenario in copied_forecast_table['Scenario'].unique():
    scenario_data = copied_forecast_table[copied_forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data['Year'], scenario_data['Forecasted Emissions'], marker='s', linestyle='-', label=scenario + ' - Reduced')

plt.title('Forecasted Emissions Comparison')
plt.xlabel('Year')
plt.ylabel('Forecasted Emissions')
plt.legend()
plt.grid(True)
plt.show()



# Create a list to store DataFrames for each scenario
frames = []

# Create a dictionary to store the values for Scenario 1
scenario_1_values = {}

# Populate the forecast_table
for scenario, growth_rates in growth_rates_scenarios.items():
    total_emissions_scenario = total_emissions_per_year.copy()

    for year in range(2024, 2051):
        if year <= 2030:
            growth_rate = growth_rates.get(2024, 0)
        elif year <= 2040:
            growth_rate = growth_rates.get(2030, 0)
        else:
            growth_rate = growth_rates.get(2040, 0)

        # Forecast total emissions for the current year
        forecasted_emissions = total_emissions_scenario['emissions_flight'].iloc[-1] * (1 + growth_rate)

        # Create a DataFrame for the current scenario and year
        df = pd.DataFrame({'Scenario': [scenario], 'Year': [year], 'Forecasted Emissions': [forecasted_emissions]})

        # Append the DataFrame to the list
        frames.append(df)

        # Update total_emissions_scenario for the next iteration
        total_emissions_scenario = pd.concat([total_emissions_scenario, pd.DataFrame({'lastseen': [pd.to_datetime(f'{year}-01-01')], 'emissions_flight': [forecasted_emissions]})],
ignore_index=True)

        # Store values for Scenario 1
        if scenario == 'Scenario 1':
            scenario_1_values[year] = forecasted_emissions

# Set display options to show all rows
pd.set_option('display.max_rows', None)
```

```
# Concatenate all DataFrames in the list into the final forecast_table
forecast_table_copied = pd.concat(frames, ignore_index=True)

# Calculate the difference in percent compared to Scenario 1
forecast_table_copied['Difference %'] = 100 * (forecast_table_copied['Forecasted Emissions'] / forecast_table_copied['Year'].map(scenario_1_values) - 1)

# Filter the table for the years 2030, 2040, and 2050
result_table_copied = forecast_table_copied[forecast_table_copied['Year'].isin([2030, 2040, 2050])]

# Display the result_table
print(result_table_copied)


total_emissions_result = result_df['emissions_flight'].sum()
total_emissions_copied = copied_result_df['emissions_flight'].sum()

# Calculate the percentage difference
percentage_difference = ((total_emissions_result - total_emissions_copied) / total_emissions_copied) * 100

print(f'Total Emissions in result_df: {total_emissions_result}')
print(f'Total Emissions in copied_df: {total_emissions_copied}')
print(f'Percentage Difference: {percentage_difference:.2f}%')

# missing fleet change sceanrio over time and scenario coplete change in 2029


# Extract emissions for the year 2029 for each scenario
emissions_2029 = forecast_table[forecast_table['Year'] == 2029]

# Define reduction percentage
reduction_percentage = 0.0991

# Create a list to store DataFrames for each new scenario
new_frames = []

# Build new scenarios based on 2029 values with a 12.49% reduction
for index, row in emissions_2029.iterrows():
    scenario = row['Scenario']
    emissions_2029_value = row['Forecasted Emissions']

    # Create a DataFrame for the new scenario and year (2024 to 2029) without reduction
    for year in range(2024, 2030):
        original_value = forecast_table.loc[(forecast_table['Scenario'] == scenario) & (forecast_table['Year'] == year), 'Forecasted Emissions'].values[0]
        new_df = pd.DataFrame({'Scenario': [f'{scenario}'], 'Year': [year], 'Forecasted Emissions': [original_value]})
        new_frames.append(new_df.copy())

    # Create a DataFrame for the new scenario and year (2030) with reduced emissions
    new_df = pd.DataFrame({'Scenario': [f'{scenario}'], 'Year': [2030], 'Forecasted Emissions': [emissions_2029_value * (1 - reduction_percentage)]})
    new_frames.append(new_df.copy())

    # Forecast emissions for the new scenario from 2031 to 2050
    for year in range(2031, 2051):
        if 2030 <= year <= 2039:
            growth_rate = growth_rates_scenarios[scenario].get(2030, 0)
        elif 2040 <= year <= 2050:
            growth_rate = growth_rates_scenarios[scenario].get(2040, 0)
        else:
            growth_rate = 0

        forecasted_emissions = new_df['Forecasted Emissions'].iloc[-1] * (1 + growth_rate)

        # Create a DataFrame for the new scenario and year
        new_df = pd.DataFrame({'Scenario': [f'{scenario}'], 'Year': [year], 'Forecasted Emissions': [forecasted_emissions]})

        # Append the DataFrame to the list for each year
        new_frames.append(new_df.copy())

# Concatenate all DataFrames in the new list into the final new_forecast_table
final_forecast_table = pd.concat(new_frames, ignore_index=True)
```

```
# Set display options to show all rows
pd.set_option('display.max_rows', None)

# Display the final_forecast_table
print(final_forecast_table)



# Optional: set a better style using seaborn
sns.set(style="whitegrid")

# Plot the data
plt.figure(figsize=(20, 12))


# Plot forecasted values from forecast_table
for scenario in forecast_table['Scenario'].unique():
    scenario_data = forecast_table[forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data['Year'], scenario_data['Forecasted Emissions'], marker='s', linestyle='-', label=scenario)

# Iterate over unique scenarios in the final_forecast_table
for scenario in final_forecast_table['Scenario'].unique():
    scenario_data = final_forecast_table[final_forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data['Year'], scenario_data['Forecasted Emissions'], marker='o', linestyle='-', label=scenario)

# Set labels and title
plt.xlabel('Year')
plt.ylabel('Forecasted Emissions')
plt.title('Forecasted Emissions for Different Scenarios Over the Years')

# Add legend
plt.legend()

# Show the plot
plt.show()

total_emissions_per_scenario = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
print("\nTotal Emissions per Scenario grwoth rates:")
print(total_emissions_per_scenario)


total_emissions_per_scenario = copied_forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
print("\nTotal Emissions per Scenario fleet change in 2024:")
print(total_emissions_per_scenario)


total_emissions_per_scenario = final_forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
print("\nTotal Emissions per Scenario fleet chanage 2029:")
print(total_emissions_per_scenario)



total_emissions_per_scenario = forecast_table_continuous.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
print("\nTotal Emissions per Scenario fleet chnage continuous:")
print(total_emissions_per_scenario)

# Group by scenario and calculate total forecasted emissions
total_emissions_original = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()

# Group by scenario and calculate total forecasted emissions (fleet change in 2024)
total_emissions_fleet_change_2024 = copied_forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()

# Group by scenario and calculate total forecasted emissions (fleet change in 2029)
total_emissions_fleet_change_2029 = final_forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
```

```
# Group by scenario and calculate total forecasted emissions (fleet change continuous)
total_emissions_fleet_change_continuous = forecast_table_continuous.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()

# Merge the tables
merged_table = pd.merge(total_emissions_original, total_emissions_fleet_change_2024, on='Scenario', suffixes=('_Original', '_FleetChange2024'))
merged_table = pd.merge(merged_table, total_emissions_fleet_change_2029, on='Scenario', suffixes=('', '_FleetChange2029'))
merged_table = pd.merge(merged_table, total_emissions_fleet_change_continuous, on='Scenario', suffixes=('', '_FleetChangeContinuous'))

# Calculate the differences in percent against the original values
merged_table['Diff_FleetChange2024_Percent'] = ((merged_table['Forecasted Emissions_FleetChange2024'] - merged_table['Forecasted Emissions_Original']) / merged_table['Forecasted Emissions_Original']) * 100
merged_table['Diff_FleetChange2029_Percent'] = ((merged_table['Forecasted Emissions'] - merged_table['Forecasted Emissions_Original']) / merged_table['Forecasted Emissions_Original']) * 100  # Adjust column name here
merged_table['Diff_FleetChangeContinuous_Percent'] = ((merged_table['Forecasted Emissions_FleetChangeContinuous'] - merged_table['Forecasted Emissions_Original']) / merged_table['Forecasted Emissions_Original']) * 100

# Print the merged table with differences in percent
print("\nMerged Table with Differences in Percent:")
(merged_table)

# Filter each dataframe to include only the rows for the year 2050
forecast_table_2050 = forecast_table[forecast_table['Year'] == 2050]
copied_forecast_table_2050 = copied_forecast_table[copied_forecast_table['Year'] == 2050]
forecast_table_continuous_2050 = forecast_table_continuous[forecast_table_continuous['Year'] == 2050]
final_forecast_table_2050 = final_forecast_table[final_forecast_table['Year'] == 2050]

# Calculate reduction rates compared to forecast_table for each scenario in the year 2050
reduction_rates_copied_forecast_table_2050 = copied_forecast_table_2050.copy()
reduction_rates_copied_forecast_table_2050['Reduction Rate'] = (forecast_table_2050['Forecasted Emissions'] - copied_forecast_table_2050['Forecasted Emissions']) / forecast_table_2050['Forecasted Emissions']

reduction_rates_forecast_table_continuous_2050 = forecast_table_continuous_2050.copy()
reduction_rates_forecast_table_continuous_2050['Reduction Rate'] = (forecast_table_2050['Forecasted Emissions'] - forecast_table_continuous_2050['Forecasted Emissions']) / forecast_table_2050['Forecasted Emissions']

reduction_rates_final_forecast_table_2050 = final_forecast_table_2050.copy()
reduction_rates_final_forecast_table_2050['Reduction Rate'] = (forecast_table_2050['Forecasted Emissions'] - final_forecast_table_2050['Forecasted Emissions']) / forecast_table_2050['Forecasted Emissions']

# Print the results
print("\nEmissions per Scenario in the year 2050 for forecast_table:")
print(forecast_table_2050[['Scenario', 'Forecasted Emissions']])

print("\nEmissions and Reduction Rates per Scenario in the year 2050 for copied_forecast_table:")
reduction_rates_copied_forecast_table_2050['Reduction Rate'] = reduction_rates_copied_forecast_table_2050['Reduction Rate'].round(4)
print(pd.concat([copied_forecast_table_2050[['Scenario', 'Forecasted Emissions']], reduction_rates_copied_forecast_table_2050[['Reduction Rate']]], axis=1))

print("\nEmissions and Reduction Rates per Scenario in the year 2050 for forecast_table_continuous:")
reduction_rates_forecast_table_continuous_2050['Reduction Rate'] = reduction_rates_forecast_table_continuous_2050['Reduction Rate'].round(4)
print(pd.concat([forecast_table_continuous_2050[['Scenario', 'Forecasted Emissions']], reduction_rates_forecast_table_continuous_2050[['Reduction Rate']]], axis=1))

print("\nEmissions and Reduction Rates per Scenario in the year 2050 for final_forecast_table:")
reduction_rates_final_forecast_table_2050['Reduction Rate'] = reduction_rates_final_forecast_table_2050['Reduction Rate'].round(6)
print(pd.concat([final_forecast_table_2050[['Scenario', 'Forecasted Emissions']], reduction_rates_final_forecast_table_2050[['Reduction Rate']]], axis=1))


# SAF Scenarios

# SAF Penetration rate:
#2025:2%
#2030:5%
#2035:20%
#2040:32%
#2045:38%
#2050:63%
# Co2 reduction with SAF: 70-100%

# SAF Scenario 1: Lufthansa will cope the given penetration rate and SAF will reduce CO" from fuel on average 70%
#-> 70% less emissions per kg fuel
```

```
#3.16 * 0,3 = 0,948 kg Co2 per kg fuel


# SAF Scenrio 2 : Lufthansa will cope the given penetration rate and SAF will reduce CO" from fuel on average 85%
#-> 85% less emissions per kg fuel
#3.16 * 0,15 = 0,474 kg Co2 per kg fuel

# SAF Scenario 3: Lufthansa will cope the given penetration rate and SAF will reduce CO" from fuel on average 100%
#-> 100% less emissions per kg fuel
#3.16 * 0 = 0 kg Co2 per kg fuel

# SAF scenario with 70% Co2 reduction



# Define the penetration rates for each year
penetration_rates = { 2024: 0.00,
    2025: 0.02, 2026: 0.02, 2027: 0.02, 2028: 0.02, 2029: 0.02,
    2030: 0.05, 2031: 0.05, 2032: 0.05, 2033: 0.05, 2034: 0.05,
    2035: 0.2, 2036: 0.2, 2037: 0.2, 2038: 0.2, 2039: 0.2,
    2040: 0.32, 2041: 0.32, 2042: 0.32, 2043: 0.32, 2044: 0.32,
    2045: 0.38, 2046: 0.38, 2047: 0.38, 2048: 0.38, 2049: 0.38,
    2050: 0.63
}

reduction_factor = 0.3

for year in range(2024, 2051):
    # Calculate the adjusted value for each year
    forecast_table.loc[forecast_table['Year'] == year, 'SAF70'] = (
        forecast_table.loc[forecast_table['Year'] == year, 'Forecasted Emissions'] * (1 - penetration_rates[year]) +
        forecast_table.loc[forecast_table['Year'] == year, 'Forecasted Emissions'] * penetration_rates[year] * reduction_factor
    )

# Calculate total emissions per scenario
total_original = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
total_adjusted70 = forecast_table.groupby('Scenario')['SAF70'].sum().reset_index()

# Display the tables
print("Total Emissions - Original")
print(total_original)

print("\nTotal Emissions - SAF70")
print(total_adjusted70)

# Plotting
plt.figure(figsize=(10, 6))

# Iterate through original scenarios and plot
for scenario in forecast_table['Scenario'].unique():
    scenario_data_original = forecast_table[forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data_original['Year'], scenario_data_original['Forecasted Emissions'], label=f'Original - {scenario}', marker='s', linestyle='-', color='blue')

# Iterate through continuous scenarios and plot
for scenario in forecast_table['Scenario'].unique():
    scenario_data = forecast_table[forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data['Year'], scenario_data['SAF70'], label=f'SAF70 - {scenario}', marker='o', linestyle='-', color='orange')

plt.title('Original vs Adjusted Values')
plt.xlabel('Year')
plt.ylabel('Value')
plt.legend()
plt.grid(True)
plt.show()

# Calculate total emissions per scenario for the year 2050
total_original_2050 = forecast_table[forecast_table['Year'] == 2050].groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
total_adjusted70_2050 = forecast_table[forecast_table['Year'] == 2050].groupby('Scenario')['SAF70'].sum().reset_index()
```

```
# Display the emissions per scenario in 2050
print("\nTotal Emissions in 2050 - Original")
print(total_original_2050)

print("\nTotal Emissions in 2050 - SAF70")
print(total_adjusted70_2050)

# Calculate the reduction rate for each scenario in 2050
reduction_rate_70_2050 = ((total_original_2050['Forecasted Emissions'] - total_adjusted70_2050['SAF70']) / total_original_2050['Forecasted Emissions']) * 100

# Display the reduction rate per scenario in 2050
print("\nReduction Rate in 2050 - SAF70")
reduction_rate_70_2050_df = pd.DataFrame({
    'Scenario': total_original_2050['Scenario'],
    'Reduction Rate': reduction_rate_70_2050
})
print(reduction_rate_70_2050_df)

# SAF scenario with 85% Co2 reduction



# Define the penetration rates for each year
penetration_rates = { 2024: 0.00,
    2025: 0.02, 2026: 0.02, 2027: 0.02, 2028: 0.02, 2029: 0.02,
    2030: 0.05, 2031: 0.05, 2032: 0.05, 2033: 0.05, 2034: 0.05,
    2035: 0.2, 2036: 0.2, 2037: 0.2, 2038: 0.2, 2039: 0.2,
    2040: 0.32, 2041: 0.32, 2042: 0.32, 2043: 0.32, 2044: 0.32,
    2045: 0.38, 2046: 0.38, 2047: 0.38, 2048: 0.38, 2049: 0.38,
    2050: 0.63
}

reduction_factor = 0.15

for year in range(2024, 2051):
    # Calculate the adjusted value for each year
    forecast_table.loc[forecast_table['Year'] == year, 'SAF85'] = (
        forecast_table.loc[forecast_table['Year'] == year, 'Forecasted Emissions'] * (1 - penetration_rates[year]) +
        forecast_table.loc[forecast_table['Year'] == year, 'Forecasted Emissions'] * penetration_rates[year] * reduction_factor
    )

# Calculate total emissions per scenario
total_original = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
total_adjusted85 = forecast_table.groupby('Scenario')['SAF85'].sum().reset_index()

# Display the tables
print("Total Emissions - Original")
print(total_original)

print("\nTotal Emissions - SAF85")
print(total_adjusted85)

# Plotting
plt.figure(figsize=(10, 6))

# Iterate through original scenarios and plot
for scenario in forecast_table['Scenario'].unique():
    scenario_data_original = forecast_table[forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data_original['Year'], scenario_data_original['Forecasted Emissions'], label=f'Original - {scenario}', marker='s', linestyle='-', color='blue')

# Iterate through continuous scenarios and plot
for scenario in forecast_table['Scenario'].unique():
    scenario_data = forecast_table[forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data['Year'], scenario_data['SAF85'], label=f'SAF85 - {scenario}', marker='o', linestyle='-', color='orange')

plt.title('Original vs Adjusted Values')
plt.xlabel('Year')
plt.ylabel('Value')
```

```
plt.legend()
plt.grid(True)
plt.show()

# Calculate total emissions per scenario for the year 2050
total_original_2050 = forecast_table[forecast_table['Year'] == 2050].groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
total_adjusted85_2050 = forecast_table[forecast_table['Year'] == 2050].groupby('Scenario')['SAF85'].sum().reset_index()

# Display the emissions per scenario in 2050
print("\nTotal Emissions in 2050 - Original")
print(total_original_2050)

print("\nTotal Emissions in 2050 - SAF85")
print(total_adjusted85_2050)

# Calculate the reduction rate for each scenario in 2050
reduction_rate_85_2050 = ((total_original_2050['Forecasted Emissions'] - total_adjusted85_2050['SAF85']) / total_original_2050['Forecasted Emissions']) * 100

# Display the reduction rate per scenario in 2050
print("\nReduction Rate in 2050 - SAF85")
reduction_rate_85_2050_df = pd.DataFrame({
    'Scenario': total_original_2050['Scenario'],
    'Reduction Rate': reduction_rate_85_2050
})
print(reduction_rate_85_2050_df)

# SAF scenario with 100% Co2 reduction



# Define the penetration rates for each year
penetration_rates = { 2024: 0.00,
    2025: 0.02, 2026: 0.02, 2027: 0.02, 2028: 0.02, 2029: 0.02,
    2030: 0.05, 2031: 0.05, 2032: 0.05, 2033: 0.05, 2034: 0.05,
    2035: 0.2, 2036: 0.2, 2037: 0.2, 2038: 0.2, 2039: 0.2,
    2040: 0.32, 2041: 0.32, 2042: 0.32, 2043: 0.32, 2044: 0.32,
    2045: 0.38, 2046: 0.38, 2047: 0.38, 2048: 0.38, 2049: 0.38,
    2050: 0.63
}

reduction_factor = 0

for year in range(2024, 2051):
    # Calculate the adjusted value for each year
    forecast_table.loc[forecast_table['Year'] == year, 'SAF100'] = (
        forecast_table.loc[forecast_table['Year'] == year, 'Forecasted Emissions'] * (1 - penetration_rates[year]) +
        forecast_table.loc[forecast_table['Year'] == year, 'Forecasted Emissions'] * penetration_rates[year] * reduction_factor
    )

# Calculate total emissions per scenario
total_original = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
total_adjusted100 = forecast_table.groupby('Scenario')['SAF100'].sum().reset_index()

# Display the tables
print("Total Emissions - Original")
print(total_original)

print("\nTotal Emissions - SAF100")
print(total_adjusted100)

# Plotting
plt.figure(figsize=(10, 6))

# Iterate through original scenarios and plot
for scenario in forecast_table['Scenario'].unique():
    scenario_data_original = forecast_table[forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data_original['Year'], scenario_data_original['Forecasted Emissions'], label=f'Original - {scenario}', marker='s', linestyle='-', color='blue')

# Iterate through continuous scenarios and plot
```

```
for scenario in forecast_table['Scenario'].unique():
    scenario_data = forecast_table[forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data['Year'], scenario_data['SAF100'], label=f'SAF100 - {scenario}', marker='o', linestyle='-', color='orange')

plt.title('Original vs Adjusted Values')
plt.xlabel('Year')
plt.ylabel('Value')
plt.legend()
plt.grid(True)
plt.show()

# Calculate total emissions per scenario for the year 2050
total_original_2050 = forecast_table[forecast_table['Year'] == 2050].groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
total_adjusted100_2050 = forecast_table[forecast_table['Year'] == 2050].groupby('Scenario')['SAF100'].sum().reset_index()

# Display the emissions per scenario in 2050
print("\nTotal Emissions in 2050 - Original")
print(total_original_2050)

print("\nTotal Emissions in 2050 - SAF100")
print(total_adjusted100_2050)

# Calculate the reduction rate for each scenario in 2050 (since reduction_factor is 0, it should be 0%)
reduction_rate_100_2050 = ((total_original_2050['Forecasted Emissions'] - total_adjusted100_2050['SAF100']) / total_original_2050['Forecasted Emissions']) * 100

# Display the reduction rate per scenario in 2050
print("\nReduction Rate in 2050 - SAF100")
reduction_rate_100_2050_df = pd.DataFrame({
    'Scenario': total_original_2050['Scenario'],
    'Reduction Rate': reduction_rate_100_2050
})
print(reduction_rate_100_2050_df)

print("Total Emissions - Original")
print(total_original)

print("\nTotal Emissions - SAF_70")
print(total_adjusted70)

print("\nTotal Emissions - SAF_85")
print(total_adjusted85)

print("\nTotal Emissions - SAF_100")
print(total_adjusted100)

# Calculate total emissions per scenario for original and adjusted scenarios
total_original = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
total_adjusted70 = forecast_table.groupby('Scenario')['SAF70'].sum().reset_index()
total_adjusted85 = forecast_table.groupby('Scenario')['SAF85'].sum().reset_index()
total_adjusted100 = forecast_table.groupby('Scenario')['SAF100'].sum().reset_index()

# Merge the tables
SAF_merged_table = pd.merge(total_original, total_adjusted70, on='Scenario', suffixes=('_Original', '_SAF70'))
SAF_merged_table = pd.merge(SAF_merged_table, total_adjusted85, on='Scenario', suffixes=('', '_SAF85'))
SAF_merged_table = pd.merge(SAF_merged_table, total_adjusted100, on='Scenario', suffixes=('', '_SAF100'))

# Calculate the differences in percent against the original values
SAF_merged_table['Diff_SAF70_Percent'] = ((SAF_merged_table['SAF70'] - SAF_merged_table['Forecasted Emissions']) / SAF_merged_table['Forecasted Emissions']) * 100
SAF_merged_table['Diff_SAF85_Percent'] = ((SAF_merged_table['SAF85'] - SAF_merged_table['Forecasted Emissions']) / SAF_merged_table['Forecasted Emissions']) * 100
SAF_merged_table['Diff_SAF100_Percent'] = ((SAF_merged_table['SAF100'] - SAF_merged_table['Forecasted Emissions']) / SAF_merged_table['Forecasted Emissions']) * 100

# Print the merged table with differences in percent
print("\nSAF Merged Table with Differences in Percent:")
(SAF_merged_table)

# Merge reduction rate tables on 'Scenario'
merged_reduction_rates = pd.merge(reduction_rate_70_2050_df, reduction_rate_85_2050_df, on='Scenario', suffixes=('_SAF70', '_SAF85'))
merged_reduction_rates = pd.merge(merged_reduction_rates, reduction_rate_100_2050_df, on='Scenario')
```

```
# Display the merged table
print("Merged Reduction Rate Table in 2050:")
print(merged_reduction_rates)

#combined Scenarios

#copied & SAF70



# Define the penetration rates for each year
penetration_rates = { 2024: 0.00,
    2025: 0.02, 2026: 0.02, 2027: 0.02, 2028: 0.02, 2029: 0.02,
    2030: 0.05, 2031: 0.05, 2032: 0.05, 2033: 0.05, 2034: 0.05,
    2035: 0.2, 2036: 0.2, 2037: 0.2, 2038: 0.2, 2039: 0.2,
    2040: 0.32, 2041: 0.32, 2042: 0.32, 2043: 0.32, 2044: 0.32,
    2045: 0.38, 2046: 0.38, 2047: 0.38, 2048: 0.38, 2049: 0.38,
    2050: 0.63
}

reduction_factor = 0.3


for year in range(2024, 2051):
    copied_forecast_table.loc[copied_forecast_table['Year'] == year, 'SAF70'] = (
        copied_forecast_table.loc[copied_forecast_table['Year'] == year, 'Forecasted Emissions'] * (1 - penetration_rates[year]) +
        copied_forecast_table.loc[copied_forecast_table['Year'] == year, 'Forecasted Emissions'] * penetration_rates[year] * reduction_factor
    )

# Calculate total emissions per scenario
total_original = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
copied_total_adjusted70 = copied_forecast_table.groupby('Scenario')['SAF70'].sum().reset_index()

# Display the tables
print("Total Emissions - Original")
print(total_original)

print("\nTotal Emissions - SAF70")
print(copied_total_adjusted70)

# Plotting
plt.figure(figsize=(10, 6))
# Iterate through original scenarios and plot
for scenario in forecast_table['Scenario'].unique():
    scenario_data_original = forecast_table[forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data_original['Year'], scenario_data_original['Forecasted Emissions'], label=f'Original - {scenario}', marker='s', linestyle='-', color='blue')

# Iterate through continuous scenarios and plot
for scenario in copied_forecast_table['Scenario'].unique():
    scenario_data = copied_forecast_table[copied_forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data['Year'], scenario_data['SAF70'], label=f'SAF70 - {scenario}', marker='o', linestyle='-', color='orange')

plt.title('Original vs Adjusted Values')
plt.xlabel('Year')
plt.ylabel('Value')
plt.legend()
plt.grid(True)
plt.show()

# Calculate total emissions per scenario for the year 2050
total_original_2050 = forecast_table[forecast_table['Year'] == 2050].groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
total_adjusted70_2050 = copied_forecast_table[copied_forecast_table['Year'] == 2050].groupby('Scenario')['SAF70'].sum().reset_index()

# Display the emissions per scenario in 2050
print("\nTotal Emissions in 2050 - Original")
print(total_original_2050)

print("\nTotal Emissions in 2050 - SAF70")
print(total_adjusted70_2050)
```

```
# Calculate the reduction rate for each scenario in 2050
reduction_rate_70_2050 = ((total_original_2050['Forecasted Emissions'] - total_adjusted70_2050['SAF70']) / total_original_2050['Forecasted Emissions']) * 100

# Display the reduction rate per scenario in 2050
print("\nReduction Rate in 2050 - SAF70")
copied_reduction_rate_70_2050_df = pd.DataFrame({
    'Scenario': total_original_2050['Scenario'],
    'Reduction Rate': reduction_rate_70_2050
})
print(copied_reduction_rate_70_2050_df)

#copied & SAF85



# Define the penetration rates for each year
penetration_rates = { 2024: 0.00,
    2025: 0.02, 2026: 0.02, 2027: 0.02, 2028: 0.02, 2029: 0.02,
    2030: 0.05, 2031: 0.05, 2032: 0.05, 2033: 0.05, 2034: 0.05,
    2035: 0.2, 2036: 0.2, 2037: 0.2, 2038: 0.2, 2039: 0.2,
    2040: 0.32, 2041: 0.32, 2042: 0.32, 2043: 0.32, 2044: 0.32,
    2045: 0.38, 2046: 0.38, 2047: 0.38, 2048: 0.38, 2049: 0.38,
    2050: 0.63
}

reduction_factor = 0.15


for year in range(2024, 2051):
    copied_forecast_table.loc[copied_forecast_table['Year'] == year, 'SAF85'] = (
        copied_forecast_table.loc[copied_forecast_table['Year'] == year, 'Forecasted Emissions'] * (1 - penetration_rates[year]) +
        copied_forecast_table.loc[copied_forecast_table['Year'] == year, 'Forecasted Emissions'] * penetration_rates[year] * reduction_factor
    )

# Calculate total emissions per scenario
total_original = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
copied_total_adjuste85 = copied_forecast_table.groupby('Scenario')['SAF85'].sum().reset_index()

# Display the tables
print("Total Emissions - Original")
print(total_original)

print("\nTotal Emissions - SAF85")
print(copied_total_adjuste85)

# Plotting
plt.figure(figsize=(10, 6))

# Iterate through original scenarios and plot
for scenario in forecast_table['Scenario'].unique():
    scenario_data_original = forecast_table[forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data_original['Year'], scenario_data_original['Forecasted Emissions'], label=f'Original - {scenario}', marker='s', linestyle='-', color='blue')

# Iterate through continuous scenarios and plot
for scenario in copied_forecast_table['Scenario'].unique():
    scenario_data = copied_forecast_table[copied_forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data['Year'], scenario_data['SAF85'], label=f'SAF85 - {scenario}', marker='o', linestyle='-', color='orange')

plt.title('Original vs Adjusted Values')
plt.xlabel('Year')
plt.ylabel('Value')
plt.legend()
plt.grid(True)
plt.show()

# Calculate total emissions per scenario for the year 2050
total_original_2050 = forecast_table[forecast_table['Year'] == 2050].groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
total_adjusted85_2050 = copied_forecast_table[copied_forecast_table['Year'] == 2050].groupby('Scenario')['SAF85'].sum().reset_index()
```

```
# Display the emissions per scenario in 2050
print("\nTotal Emissions in 2050 - Original")
print(total_original_2050)

print("\nTotal Emissions in 2050 - SAF85")
print(total_adjusted85_2050)

# Calculate the reduction rate for each scenario in 2050
reduction_rate_85_2050 = ((total_original_2050['Forecasted Emissions'] - total_adjusted85_2050['SAF85']) / total_original_2050['Forecasted Emissions']) * 100

# Display the reduction rate per scenario in 2050
print("\nReduction Rate in 2050 - SAF85")
copied_reduction_rate_85_2050_df = pd.DataFrame({
    'Scenario': total_original_2050['Scenario'],
    'Reduction Rate': reduction_rate_85_2050
})
print(copied_reduction_rate_85_2050_df)

#copied & SAF100



# Define the penetration rates for each year
penetration_rates = { 2024: 0.00,
    2025: 0.02, 2026: 0.02, 2027: 0.02, 2028: 0.02, 2029: 0.02,
    2030: 0.05, 2031: 0.05, 2032: 0.05, 2033: 0.05, 2034: 0.05,
    2035: 0.2, 2036: 0.2, 2037: 0.2, 2038: 0.2, 2039: 0.2,
    2040: 0.32, 2041: 0.32, 2042: 0.32, 2043: 0.32, 2044: 0.32,
    2045: 0.38, 2046: 0.38, 2047: 0.38, 2048: 0.38, 2049: 0.38,
    2050: 0.63
}

reduction_factor = 0


for year in range(2024, 2051):
    copied_forecast_table.loc[copied_forecast_table['Year'] == year, 'SAF100'] = (
        copied_forecast_table.loc[copied_forecast_table['Year'] == year, 'Forecasted Emissions'] * (1 - penetration_rates[year]) +
        copied_forecast_table.loc[copied_forecast_table['Year'] == year, 'Forecasted Emissions'] * penetration_rates[year] * reduction_factor
    )

# Calculate total emissions per scenario
total_original = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
copied_total_adjusted100 = copied_forecast_table.groupby('Scenario')['SAF100'].sum().reset_index()

# Display the tables
print("Total Emissions - Original")
print(total_original)

print("\nTotal Emissions - SAF100")
print(copied_total_adjusted100)

# Plotting
plt.figure(figsize=(10, 6))

# Iterate through original scenarios and plot
for scenario in forecast_table['Scenario'].unique():
    scenario_data_original = forecast_table[forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data_original['Year'], scenario_data_original['Forecasted Emissions'], label=f'Original - {scenario}', marker='s', linestyle='-', color='blue')

# Iterate through continuous scenarios and plot
for scenario in copied_forecast_table['Scenario'].unique():
    scenario_data = copied_forecast_table[copied_forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data['Year'], scenario_data['SAF100'], label=f'SAF100 - {scenario}', marker='o', linestyle='-', color='orange')

plt.title('Original vs Adjusted Values')
plt.xlabel('Year')
```

```
plt.ylabel('Value')
plt.legend()
plt.grid(True)
plt.show()

# Calculate total emissions per scenario for the year 2050
total_original_2050 = forecast_table[forecast_table['Year'] == 2050].groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
total_adjusted100_2050 = copied_forecast_table[copied_forecast_table['Year'] == 2050].groupby('Scenario')['SAF100'].sum().reset_index()

# Display the emissions per scenario in 2050
print("\nTotal Emissions in 2050 - Original")
print(total_original_2050)

print("\nTotal Emissions in 2050 - SAF100")
print(total_adjusted100_2050)

# Calculate the reduction rate for each scenario in 2050
reduction_rate_100_2050 = ((total_original_2050['Forecasted Emissions'] - total_adjusted100_2050['SAF100']) / total_original_2050['Forecasted Emissions']) * 100

# Display the reduction rate per scenario in 2050
print("\nReduction Rate in 2050 - SAF100")
copied_reduction_rate_100_2050_df = pd.DataFrame({
    'Scenario': total_original_2050['Scenario'],
    'Reduction Rate': reduction_rate_100_2050
})
print(copied_reduction_rate_100_2050_df)

# Calculate total emissions per scenario for original and adjusted scenarios
total_original = forecast_table.groupby(['Scenario', 'Year'])['Forecasted Emissions'].sum().reset_index()
copied_total_adjusted100 = copied_forecast_table.groupby(['Scenario', 'Year'])['SAF100'].sum().reset_index()

# Merge the tables on 'Scenario' and 'Year'
comparison_merged_table = pd.merge(total_original, copied_total_adjusted100, on=['Scenario', 'Year'], suffixes=('_Original', '_SAF100'))

# Calculate the reduction percentage
comparison_merged_table['Reduction_Percent'] = ((comparison_merged_table['Forecasted Emissions'] - comparison_merged_table['SAF100']) / comparison_merged_table['Forecasted
Emissions']) * 100

# Display the table
print("\nEmissions Table - Original vs SAF100 with Reduction Percent:")
print(comparison_merged_table)

print("Total Emissions - Original")
print(total_original)

print("\nTotal Emissions - SAF70")
print(copied_total_adjusted70)

print("\nTotal Emissions - SAF85")
print(copied_total_adjuste85)

print("\nTotal Emissions - SAF100")
print(copied_total_adjusted100)

# Calculate total emissions per scenario for original and adjusted scenarios
copied_total_original = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
copied_total_adjusted70 = copied_forecast_table.groupby('Scenario')['SAF70'].sum().reset_index()
copied_total_adjusted85 = copied_forecast_table.groupby('Scenario')['SAF85'].sum().reset_index()
copied_total_adjusted100 = copied_forecast_table.groupby('Scenario')['SAF100'].sum().reset_index()

# Merge the tables
copied_SAF_merged_table = pd.merge(copied_total_original, copied_total_adjusted70, on='Scenario', suffixes=('_Original', '_SAF70'))
copied_SAF_merged_table = pd.merge(copied_SAF_merged_table, copied_total_adjusted85, on='Scenario', suffixes=('', '_SAF85'))
copied_SAF_merged_table = pd.merge(copied_SAF_merged_table, copied_total_adjusted100, on='Scenario', suffixes=('', '_SAF100'))

# Calculate the differences in percent against the original values
copied_SAF_merged_table['Diff_SAF70_Percent'] = ((copied_SAF_merged_table['SAF70'] - copied_SAF_merged_table['Forecasted Emissions']) / copied_SAF_merged_table['Forecasted
Emissions']) * 100
copied_SAF_merged_table['Diff_SAF85_Percent'] = ((copied_SAF_merged_table['SAF85'] - copied_SAF_merged_table['Forecasted Emissions']) / copied_SAF_merged_table['Forecasted
```

```
Emissions']) * 100
copied_SAF_merged_table['Diff_SAF100_Percent'] = ((copied_SAF_merged_table['SAF100'] - copied_SAF_merged_table['Forecasted Emissions']) /
copied_SAF_merged_table['Forecasted Emissions']) * 100

# Print the merged table with differences in percent
print("\nCopied SAF Merged Table with Differences in Percent:")
(copied_SAF_merged_table)

#continuous & SAF70



# Define the penetration rates for each year
penetration_rates = { 2024: 0.00,
    2025: 0.02, 2026: 0.02, 2027: 0.02, 2028: 0.02, 2029: 0.02,
    2030: 0.05, 2031: 0.05, 2032: 0.05, 2033: 0.05, 2034: 0.05,
    2035: 0.2, 2036: 0.2, 2037: 0.2, 2038: 0.2, 2039: 0.2,
    2040: 0.32, 2041: 0.32, 2042: 0.32, 2043: 0.32, 2044: 0.32,
    2045: 0.38, 2046: 0.38, 2047: 0.38, 2048: 0.38, 2049: 0.38,
    2050: 0.63
}

reduction_factor = 0.3


for year in range(2024, 2051):
    forecast_table_continuous.loc[forecast_table_continuous['Year'] == year, 'SAF70'] = (
        forecast_table_continuous.loc[forecast_table_continuous['Year'] == year, 'Forecasted Emissions'] * (1 - penetration_rates[year]) +
        forecast_table_continuous.loc[forecast_table_continuous['Year'] == year, 'Forecasted Emissions'] * penetration_rates[year] * reduction_factor
    )

# Calculate total emissions per scenario
total_original = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
continuous_total_adjusted70 = forecast_table_continuous.groupby('Scenario')['SAF70'].sum().reset_index()

# Display the tables
print("Total Emissions - Original")
print(total_original)

print("\nTotal Emissions - SAF70")
print(continuous_total_adjusted70)

# Plotting
plt.figure(figsize=(10, 6))

# Iterate through original scenarios and plot
for scenario in forecast_table['Scenario'].unique():
    scenario_data_original = forecast_table[forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data_original['Year'], scenario_data_original['Forecasted Emissions'], label=f'Original - {scenario}', marker='s', linestyle='-', color='blue')

# Iterate through continuous scenarios and plot
for scenario in forecast_table_continuous['Scenario'].unique():
    scenario_data_continuous = forecast_table_continuous[forecast_table_continuous['Scenario'] == scenario]
    plt.plot(scenario_data_continuous['Year'], scenario_data_continuous['SAF70'], label=f'SAF70 - {scenario}', marker='o', linestyle='-', color='orange')

plt.title('Original vs Adjusted Values')
plt.xlabel('Year')
plt.ylabel('Value')
plt.legend()
plt.grid(True)
plt.show()

# Calculate total emissions per scenario for the year 2050
total_original_2050 = forecast_table[forecast_table['Year'] == 2050].groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
total_adjusted70_2050 = forecast_table_continuous[forecast_table_continuous['Year'] == 2050].groupby('Scenario')['SAF70'].sum().reset_index()

# Display the emissions per scenario in 2050
print("\nTotal Emissions in 2050 - Original")
print(total_original_2050)
```

```
print("\nTotal Emissions in 2050 - SAF70")
print(total_adjusted70_2050)

# Calculate the reduction rate for each scenario in 2050
reduction_rate_70_2050 = ((total_original_2050['Forecasted Emissions'] - total_adjusted70_2050['SAF70']) / total_original_2050['Forecasted Emissions']) * 100

# Display the reduction rate per scenario in 2050
print("\nReduction Rate in 2050 - SAF70")
continuous_reduction_rate_70_2050_df = pd.DataFrame({
    'Scenario': total_original_2050['Scenario'],
    'Reduction Rate': reduction_rate_70_2050
})
print(continuous_reduction_rate_70_2050_df)

#continuous & SAF85



# Define the penetration rates for each year
penetration_rates = { 2024: 0.00,
    2025: 0.02, 2026: 0.02, 2027: 0.02, 2028: 0.02, 2029: 0.02,
    2030: 0.05, 2031: 0.05, 2032: 0.05, 2033: 0.05, 2034: 0.05,
    2035: 0.2, 2036: 0.2, 2037: 0.2, 2038: 0.2, 2039: 0.2,
    2040: 0.32, 2041: 0.32, 2042: 0.32, 2043: 0.32, 2044: 0.32,
    2045: 0.38, 2046: 0.38, 2047: 0.38, 2048: 0.38, 2049: 0.38,
    2050: 0.63
}

reduction_factor = 0.15


for year in range(2024, 2051):
    forecast_table_continuous.loc[forecast_table_continuous['Year'] == year, 'SAF85'] = (
        forecast_table_continuous.loc[forecast_table_continuous['Year'] == year, 'Forecasted Emissions'] * (1 - penetration_rates[year]) +
        forecast_table_continuous.loc[forecast_table_continuous['Year'] == year, 'Forecasted Emissions'] * penetration_rates[year] * reduction_factor
    )

# Calculate total emissions per scenario
total_original = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
continuous_total_adjusted85 = forecast_table_continuous.groupby('Scenario')['SAF85'].sum().reset_index()

# Display the tables
print("Total Emissions - Original")
print(total_original)

print("\nTotal Emissions - SAF85")
print(continuous_total_adjusted85)

# Plotting
plt.figure(figsize=(10, 6))

# Iterate through original scenarios and plot
for scenario in forecast_table['Scenario'].unique():
    scenario_data_original = forecast_table[forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data_original['Year'], scenario_data_original['Forecasted Emissions'], label=f'Original - {scenario}', marker='s', linestyle='-', color='blue')

# Iterate through continuous scenarios and plot
for scenario in forecast_table_continuous['Scenario'].unique():
    scenario_data_continuous = forecast_table_continuous[forecast_table_continuous['Scenario'] == scenario]
    plt.plot(scenario_data_continuous['Year'], scenario_data_continuous['SAF85'], label=f'SAF85 - {scenario}', marker='o', linestyle='-', color='orange')

plt.title('Original vs Adjusted Values')
plt.xlabel('Year')
plt.ylabel('Value')
plt.legend()
plt.grid(True)
plt.show()
```

```
# Calculate total emissions per scenario for the year 2050
total_original_2050 = forecast_table[forecast_table['Year'] == 2050].groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
total_adjusted85_2050 = forecast_table_continuous[forecast_table_continuous['Year'] == 2050].groupby('Scenario')['SAF85'].sum().reset_index()

# Display the emissions per scenario in 2050
print("\nTotal Emissions in 2050 - Original")
print(total_original_2050)

print("\nTotal Emissions in 2050 - SAF85")
print(total_adjusted85_2050)

# Calculate the reduction rate for each scenario in 2050
reduction_rate_85_2050 = ((total_original_2050['Forecasted Emissions'] - total_adjusted85_2050['SAF85']) / total_original_2050['Forecasted Emissions']) * 100

# Display the reduction rate per scenario in 2050
print("\nReduction Rate in 2050 - SAF85")
continuous_reduction_rate_85_2050_df = pd.DataFrame({
    'Scenario': total_original_2050['Scenario'],
    'Reduction Rate': reduction_rate_85_2050
})
print(continuous_reduction_rate_85_2050_df)

#continuous & SAF100



# Define the penetration rates for each year
penetration_rates = { 2024: 0.00,
    2025: 0.02, 2026: 0.02, 2027: 0.02, 2028: 0.02, 2029: 0.02,
    2030: 0.05, 2031: 0.05, 2032: 0.05, 2033: 0.05, 2034: 0.05,
    2035: 0.2, 2036: 0.2, 2037: 0.2, 2038: 0.2, 2039: 0.2,
    2040: 0.32, 2041: 0.32, 2042: 0.32, 2043: 0.32, 2044: 0.32,
    2045: 0.38, 2046: 0.38, 2047: 0.38, 2048: 0.38, 2049: 0.38,
    2050: 0.63
}

reduction_factor = 0


for year in range(2024, 2051):
    forecast_table_continuous.loc[forecast_table_continuous['Year'] == year, 'SAF100'] = (
        forecast_table_continuous.loc[forecast_table_continuous['Year'] == year, 'Forecasted Emissions'] * (1 - penetration_rates[year]) +
        forecast_table_continuous.loc[forecast_table_continuous['Year'] == year, 'Forecasted Emissions'] * penetration_rates[year] * reduction_factor
    )

# Calculate total emissions per scenario
total_original = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
continuous_total_adjusted100 = forecast_table_continuous.groupby('Scenario')['SAF100'].sum().reset_index()

# Display the tables
print("Total Emissions - Original")
print(total_original)

print("\nTotal Emissions - SAF100")
print(continuous_total_adjusted100)

# Plotting
plt.figure(figsize=(10, 6))

# Iterate through original scenarios and plot
for scenario in forecast_table['Scenario'].unique():
    scenario_data_original = forecast_table[forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data_original['Year'], scenario_data_original['Forecasted Emissions'], label=f'Original - {scenario}', marker='s', linestyle='-', color='blue')

# Iterate through continuous scenarios and plot
for scenario in forecast_table_continuous['Scenario'].unique():
    scenario_data_continuous = forecast_table_continuous[forecast_table_continuous['Scenario'] == scenario]
    plt.plot(scenario_data_continuous['Year'], scenario_data_continuous['SAF100'], label=f'SAF100 - {scenario}', marker='o', linestyle='-', color='orange')
```

```
plt.title('Original vs Adjusted Values')
plt.xlabel('Year')
plt.ylabel('Value')
plt.legend()
plt.grid(True)
plt.show()

# Calculate total emissions per scenario for the year 2050
total_original_2050 = forecast_table[forecast_table['Year'] == 2050].groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
total_adjusted100_2050 = forecast_table_continuous[forecast_table_continuous['Year'] == 2050].groupby('Scenario')['SAF100'].sum().reset_index()

# Display the emissions per scenario in 2050
print("\nTotal Emissions in 2050 - Original")
print(total_original_2050)

print("\nTotal Emissions in 2050 - SAF100")
print(total_adjusted100_2050)

# Calculate the reduction rate for each scenario in 2050
reduction_rate_100_2050 = ((total_original_2050['Forecasted Emissions'] - total_adjusted100_2050['SAF100']) / total_original_2050['Forecasted Emissions']) * 100

# Display the reduction rate per scenario in 2050
print("\nReduction Rate in 2050 - SAF100")
continuous_reduction_rate_100_2050_df = pd.DataFrame({
    'Scenario': total_original_2050['Scenario'],
    'Reduction Rate': reduction_rate_100_2050
})
print(continuous_reduction_rate_100_2050_df)

print("Total Emissions - Original")
print(total_original)

print("\nTotal Emissions - SAF_70")
print(continuous_total_adjusted70)

print("\nTotal Emissions - SAF_85")
print(continuous_total_adjusted85)

print("\nTotal Emissions - SAF_100")
print(continuous_total_adjusted100)

# Calculate total emissions per scenario for original and adjusted scenarios
continuous_total_original = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
continuous_total_adjusted70 = forecast_table_continuous.groupby('Scenario')['SAF70'].sum().reset_index()
continuous_total_adjusted85 = forecast_table_continuous.groupby('Scenario')['SAF85'].sum().reset_index()
continuous_total_adjusted100 = forecast_table_continuous.groupby('Scenario')['SAF100'].sum().reset_index()

# Merge the tables
continuous_SAF_merged_table = pd.merge(continuous_total_original, continuous_total_adjusted70, on='Scenario', suffixes=('_Original', '_SAF70'))
continuous_SAF_merged_table = pd.merge(continuous_SAF_merged_table, continuous_total_adjusted85, on='Scenario', suffixes=('', '_SAF85'))
continuous_SAF_merged_table = pd.merge(continuous_SAF_merged_table, continuous_total_adjusted100, on='Scenario', suffixes=('', '_SAF100'))

# Calculate the differences in percent against the original values
continuous_SAF_merged_table['Diff_SAF70_Percent'] = ((continuous_SAF_merged_table['SAF70'] - continuous_SAF_merged_table['Forecasted Emissions']) /
continuous_SAF_merged_table['Forecasted Emissions']) * 100
continuous_SAF_merged_table['Diff_SAF85_Percent'] = ((continuous_SAF_merged_table['SAF85'] - continuous_SAF_merged_table['Forecasted Emissions']) /
continuous_SAF_merged_table['Forecasted Emissions']) * 100
continuous_SAF_merged_table['Diff_SAF100_Percent'] = ((continuous_SAF_merged_table['SAF100'] - continuous_SAF_merged_table['Forecasted Emissions']) /
continuous_SAF_merged_table['Forecasted Emissions']) * 100

# Print the merged table with differences in percent
print("\nContinuous SAF Merged Table with Differences in Percent:")
(continuous_SAF_merged_table)

#final & SAF70



# Define the penetration rates for each year
```

```
penetration_rates = { 2024: 0.00,
    2025: 0.02, 2026: 0.02, 2027: 0.02, 2028: 0.02, 2029: 0.02,
    2030: 0.05, 2031: 0.05, 2032: 0.05, 2033: 0.05, 2034: 0.05,
    2035: 0.2, 2036: 0.2, 2037: 0.2, 2038: 0.2, 2039: 0.2,
    2040: 0.32, 2041: 0.32, 2042: 0.32, 2043: 0.32, 2044: 0.32,
    2045: 0.38, 2046: 0.38, 2047: 0.38, 2048: 0.38, 2049: 0.38,
    2050: 0.63
}

reduction_factor = 0.3


for year in range(2024, 2051):
    final_forecast_table.loc[final_forecast_table['Year'] == year, 'SAF70'] = (
        final_forecast_table.loc[final_forecast_table['Year'] == year, 'Forecasted Emissions'] * (1 - penetration_rates[year]) +
        final_forecast_table.loc[final_forecast_table['Year'] == year, 'Forecasted Emissions'] * penetration_rates[year] * reduction_factor
    )

# Calculate total emissions per scenario
total_original = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
final_total_adjusted70 = final_forecast_table.groupby('Scenario')['SAF70'].sum().reset_index()

# Display the tables
print("Total Emissions - Original")
print(total_original)

print("\nTotal Emissions - SAF70")
print(final_total_adjusted70)

# Plotting
plt.figure(figsize=(10, 6))

# Iterate through original scenarios and plot
for scenario in forecast_table['Scenario'].unique():
    scenario_data_original = forecast_table[forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data_original['Year'], scenario_data_original['Forecasted Emissions'], label=f'Original - {scenario}', marker='s', linestyle='-', color='blue')

# Iterate through continuous scenarios and plot
for scenario in final_forecast_table['Scenario'].unique():
    scenario_data_final = final_forecast_table[final_forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data_final['Year'], scenario_data_final['SAF70'], label=f'SAF70 - {scenario}', marker='o', linestyle='-', color='orange')

plt.title('Original vs Adjusted Values')
plt.xlabel('Year')
plt.ylabel('Value')
plt.legend()
plt.grid(True)
plt.show()

# Calculate total emissions per scenario for the year 2050
total_original_2050 = forecast_table[forecast_table['Year'] == 2050].groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
final_total_adjusted70_2050 = final_forecast_table[final_forecast_table['Year'] == 2050].groupby('Scenario')['SAF70'].sum().reset_index()

# Display the emissions per scenario in 2050
print("\nTotal Emissions in 2050 - Original")
print(total_original_2050)

print("\nTotal Emissions in 2050 - SAF70")
print(final_total_adjusted70_2050)

# Calculate the reduction rate for each scenario in 2050
reduction_rate_2050 = ((total_original_2050['Forecasted Emissions'] - final_total_adjusted70_2050['SAF70']) / total_original_2050['Forecasted Emissions']) * 100

# Display the reduction rate per scenario in 2050
print("\nReduction Rate in 2050")
final_reduction_rate_70_2050_df = pd.DataFrame({
    'Scenario': total_original_2050['Scenario'],
    'Reduction Rate': reduction_rate_2050
```

```
})
print(final_reduction_rate_70_2050_df)

#final & SAF85



# Define the penetration rates for each year
penetration_rates = { 2024: 0.00,
    2025: 0.02, 2026: 0.02, 2027: 0.02, 2028: 0.02, 2029: 0.02,
    2030: 0.05, 2031: 0.05, 2032: 0.05, 2033: 0.05, 2034: 0.05,
    2035: 0.2, 2036: 0.2, 2037: 0.2, 2038: 0.2, 2039: 0.2,
    2040: 0.32, 2041: 0.32, 2042: 0.32, 2043: 0.32, 2044: 0.32,
    2045: 0.38, 2046: 0.38, 2047: 0.38, 2048: 0.38, 2049: 0.38,
    2050: 0.63
}

reduction_factor = 0.15


for year in range(2024, 2051):
    final_forecast_table.loc[final_forecast_table['Year'] == year, 'SAF85'] = (
        final_forecast_table.loc[final_forecast_table['Year'] == year, 'Forecasted Emissions'] * (1 - penetration_rates[year]) +
        final_forecast_table.loc[final_forecast_table['Year'] == year, 'Forecasted Emissions'] * penetration_rates[year] * reduction_factor
    )

# Calculate total emissions per scenario
total_original = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
final_total_adjusted85 = final_forecast_table.groupby('Scenario')['SAF85'].sum().reset_index()

# Display the tables
print("Total Emissions - Original")
print(total_original)

print("\nTotal Emissions - SAF85")
print(final_total_adjusted85)

# Plotting
plt.figure(figsize=(10, 6))

# Iterate through original scenarios and plot
for scenario in forecast_table['Scenario'].unique():
    scenario_data_original = forecast_table[forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data_original['Year'], scenario_data_original['Forecasted Emissions'], label=f'Original - {scenario}', marker='s', linestyle='-', color='blue')

# Iterate through continuous scenarios and plot
for scenario in final_forecast_table['Scenario'].unique():
    scenario_data_final = final_forecast_table[final_forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data_final['Year'], scenario_data_final['SAF85'], label=f'SAF85 - {scenario}', marker='o', linestyle='-', color='orange')

plt.title('Original vs Adjusted Values')
plt.xlabel('Year')
plt.ylabel('Value')
plt.legend()
plt.grid(True)
plt.show()

# Calculate total emissions per scenario for the year 2050
total_original_2050 = forecast_table[forecast_table['Year'] == 2050].groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
final_total_adjusted85_2050 = final_forecast_table[final_forecast_table['Year'] == 2050].groupby('Scenario')['SAF85'].sum().reset_index()

# Display the emissions per scenario in 2050
print("\nTotal Emissions in 2050 - Original")
print(total_original_2050)

print("\nTotal Emissions in 2050 - SAF85")
print(final_total_adjusted85_2050)

# Calculate the reduction rate for each scenario in 2050
```

```
reduction_rate_85_2050 = ((total_original_2050['Forecasted Emissions'] - final_total_adjusted85_2050['SAF85']) / total_original_2050['Forecasted Emissions']) * 100

# Display the reduction rate per scenario in 2050
print("\nReduction Rate in 2050 - SAF85")
final_reduction_rate_85_2050_df = pd.DataFrame({
    'Scenario': total_original_2050['Scenario'],
    'Reduction Rate': reduction_rate_85_2050
})
print(final_reduction_rate_85_2050_df)

#final & SAF100



# Define the penetration rates for each year
penetration_rates = { 2024: 0.00,
    2025: 0.02, 2026: 0.02, 2027: 0.02, 2028: 0.02, 2029: 0.02,
    2030: 0.05, 2031: 0.05, 2032: 0.05, 2033: 0.05, 2034: 0.05,
    2035: 0.2, 2036: 0.2, 2037: 0.2, 2038: 0.2, 2039: 0.2,
    2040: 0.32, 2041: 0.32, 2042: 0.32, 2043: 0.32, 2044: 0.32,
    2045: 0.38, 2046: 0.38, 2047: 0.38, 2048: 0.38, 2049: 0.38,
    2050: 0.63
}

reduction_factor = 0


for year in range(2024, 2051):
    final_forecast_table.loc[final_forecast_table['Year'] == year, 'SAF100'] = (
        final_forecast_table.loc[final_forecast_table['Year'] == year, 'Forecasted Emissions'] * (1 - penetration_rates[year]) +
        final_forecast_table.loc[final_forecast_table['Year'] == year, 'Forecasted Emissions'] * penetration_rates[year] * reduction_factor
    )

# Calculate total emissions per scenario
total_original = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
final_total_adjusted100 = final_forecast_table.groupby('Scenario')['SAF100'].sum().reset_index()

# Display the tables
print("Total Emissions - Original")
print(total_original)

print("\nTotal Emissions - SAF100")
print(final_total_adjusted100)

# Plotting
plt.figure(figsize=(10, 6))

# Iterate through original scenarios and plot
for scenario in forecast_table['Scenario'].unique():
    scenario_data_original = forecast_table[forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data_original['Year'], scenario_data_original['Forecasted Emissions'], label=f'Original - {scenario}', marker='s', linestyle='-', color='blue')

# Iterate through continuous scenarios and plot
for scenario in final_forecast_table['Scenario'].unique():
    scenario_data_final = final_forecast_table[final_forecast_table['Scenario'] == scenario]
    plt.plot(scenario_data_final['Year'], scenario_data_final['SAF100'], label=f'SAF100 - {scenario}', marker='o', linestyle='-', color='orange')

plt.title('Original vs Adjusted Values')
plt.xlabel('Year')
plt.ylabel('Value')
plt.legend()
plt.grid(True)
plt.show()

# Calculate total emissions per scenario for the year 2050
total_original_2050 = forecast_table[forecast_table['Year'] == 2050].groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
final_total_adjusted100_2050 = final_forecast_table[final_forecast_table['Year'] == 2050].groupby('Scenario')['SAF100'].sum().reset_index()

# Display the emissions per scenario in 2050
```

```
print("\nTotal Emissions in 2050 - Original")
print(total_original_2050)

print("\nTotal Emissions in 2050 - SAF100")
print(final_total_adjusted100_2050)

# Calculate the reduction rate for each scenario in 2050
reduction_rate_100_2050 = ((total_original_2050['Forecasted Emissions'] - final_total_adjusted100_2050['SAF100']) / total_original_2050['Forecasted Emissions']) * 100

# Display the reduction rate per scenario in 2050
print("\nReduction Rate in 2050 - SAF100")
final_reduction_rate_100_2050_df = pd.DataFrame({
    'Scenario': total_original_2050['Scenario'],
    'Reduction Rate': reduction_rate_100_2050
})
print(final_reduction_rate_100_2050_df)

print("Total Emissions - Original")
print(total_original)

print("\nTotal Emissions - SAF_70")
print(final_total_adjusted70)

print("\nTotal Emissions - SAF_85")
print(final_total_adjusted85)

print("\nTotal Emissions - SAF_100")
print(final_total_adjusted100)

# Calculate total emissions per scenario for original and adjusted scenarios
final_total_original = forecast_table.groupby('Scenario')['Forecasted Emissions'].sum().reset_index()
final_total_adjusted70 = final_forecast_table.groupby('Scenario')['SAF70'].sum().reset_index()
final_total_adjusted85 = final_forecast_table.groupby('Scenario')['SAF85'].sum().reset_index()
final_total_adjusted100 = final_forecast_table.groupby('Scenario')['SAF100'].sum().reset_index()

# Merge the tables
final_SAF_merged_table = pd.merge(final_total_original, final_total_adjusted70, on='Scenario', suffixes=('_Original', '_SAF70'))
final_SAF_merged_table = pd.merge(final_SAF_merged_table, final_total_adjusted85, on='Scenario', suffixes=('', '_SAF85'))
final_SAF_merged_table = pd.merge(final_SAF_merged_table, final_total_adjusted100, on='Scenario', suffixes=('', '_SAF100'))

# Calculate the differences in percent against the original values
final_SAF_merged_table['Diff_SAF70_Percent'] = ((final_SAF_merged_table['SAF70'] - final_SAF_merged_table['Forecasted Emissions']) / final_SAF_merged_table['Forecasted Emissions']) * 100
final_SAF_merged_table['Diff_SAF85_Percent'] = ((final_SAF_merged_table['SAF85'] - final_SAF_merged_table['Forecasted Emissions']) / final_SAF_merged_table['Forecasted Emissions']) * 100
final_SAF_merged_table['Diff_SAF100_Percent'] = ((final_SAF_merged_table['SAF100'] - final_SAF_merged_table['Forecasted Emissions']) / final_SAF_merged_table['Forecasted Emissions']) * 100

# Print the merged table with differences in percent
print("\nFinal SAF Merged Table with Differences in Percent:")
(final_SAF_merged_table)

# Merge all SAF merged tables into one
all_SAF_merged_table = pd.merge(SAF_merged_table, copied_SAF_merged_table, on='Scenario', suffixes=('_Original', '_Copied'))
all_SAF_merged_table = pd.merge(all_SAF_merged_table, continuous_SAF_merged_table, on='Scenario', suffixes=('', '_Continuous'))
all_SAF_merged_table = pd.merge(all_SAF_merged_table, final_SAF_merged_table, on='Scenario', suffixes=('', '_Final'))


# Display only columns related to differences in percent
diff_columns = [col for col in all_SAF_merged_table.columns if 'Diff' in col]

# Print the merged table with differences in percent
print("\nCombined SAF Merged Table with Differences in Percent:")
(all_SAF_merged_table[['Scenario'] + diff_columns])

# Assuming all the dataframes are already defined
# Replace these with your actual dataframes

# Add a 'Source' column to reduction_rate_70_2050_df, reduction_rate_85_2050_df, and reduction_rate_100_2050_df
```

```
reduction_rate_70_2050_df['Source'] = 'Demand grwoth scenario SAF70'
reduction_rate_85_2050_df['Source'] = 'Demand grwoth scenario SAF85'
reduction_rate_100_2050_df['Source'] = 'Demand grwoth scenario SAF100'

# Add a 'Source' column to copied_reduction_rate_70_2050_df, copied_reduction_rate_85_2050_df, and copied_reduction_rate_100_2050_df
copied_reduction_rate_70_2050_df['Source'] = 'Fleet change in 2024 SAF70'
copied_reduction_rate_85_2050_df['Source'] = 'Fleet change in 2024 SAF85'
copied_reduction_rate_100_2050_df['Source'] = 'Fleet change in 2024 SAF100'

# Add a 'Source' column to continuous_reduction_rate_70_2050_df, continuous_reduction_rate_85_2050_df, and continuous_reduction_rate_100_2050_df
continuous_reduction_rate_70_2050_df['Source'] = 'continuous fleet change SAF70'
continuous_reduction_rate_85_2050_df['Source'] = 'continuous fleet change SAF85'
continuous_reduction_rate_100_2050_df['Source'] = 'continuous fleet change SAF100'

# Add a 'Source' column to final_reduction_rate_70_2050_df, final _reduction_rate_85_2050_df, and final_reduction_rate_100_2050_df
final_reduction_rate_70_2050_df['Source'] = 'Fleet change in 2029 SAF70'
final_reduction_rate_85_2050_df['Source'] = 'Fleet change in 2029 SAF85'
final_reduction_rate_100_2050_df['Source'] = 'Fleet change in 2029 SAF100'

# Merge the dataframes as before
merged_df_1 = pd.concat([reduction_rate_70_2050_df, reduction_rate_85_2050_df, reduction_rate_100_2050_df], ignore_index=True)
merged_df_2 = pd.concat([copied_reduction_rate_70_2050_df, copied_reduction_rate_85_2050_df, copied_reduction_rate_100_2050_df], ignore_index=True)
merged_df_3 = pd.concat([continuous_reduction_rate_70_2050_df, continuous_reduction_rate_85_2050_df, continuous_reduction_rate_100_2050_df], ignore_index=True)
merged_df_4 = pd.concat([final_reduction_rate_70_2050_df, final_reduction_rate_85_2050_df, final_reduction_rate_100_2050_df], ignore_index=True)

# Merge all the merged dataframes into a single dataframe
merged_final_df = pd.concat([merged_df_1, merged_df_2, merged_df_3, merged_df_4], ignore_index=True)

# Display the merged dataframe
print("Merged Dataframe:")
print(merged_final_df)

import pypandoc

# Convert Jupyter Notebook to Word document
pypandoc.convert_file('final thesis code.ipynb', 'docx', outputfile='thesis_code1.docx')

"
```