Data Warehouse For Flight Delays

Based on US domestic flights in 2022

Prepared by: Talha Caliskan
MAT. Number: 271310
Responsible Faculty Prof. Giorgio Terracina

1. Conceptual Design

1.1 Choosing The Data

For my data warehouse project, I selected a comprehensive flight delays dataset containing operational flight data from the United States in 2022. The dataset includes multiple aspects of flight operations including weather conditions, airport information, aircraft details, and delay specifics.

Source: 2022 US Airlines Domestic Departure dataset on Kaggle

Required Files: CompleteData.csv, Stations.csv, Carriers.csv (Entries from ActiveWeather.csv and Cancellations.csv was also used in the data loading process)

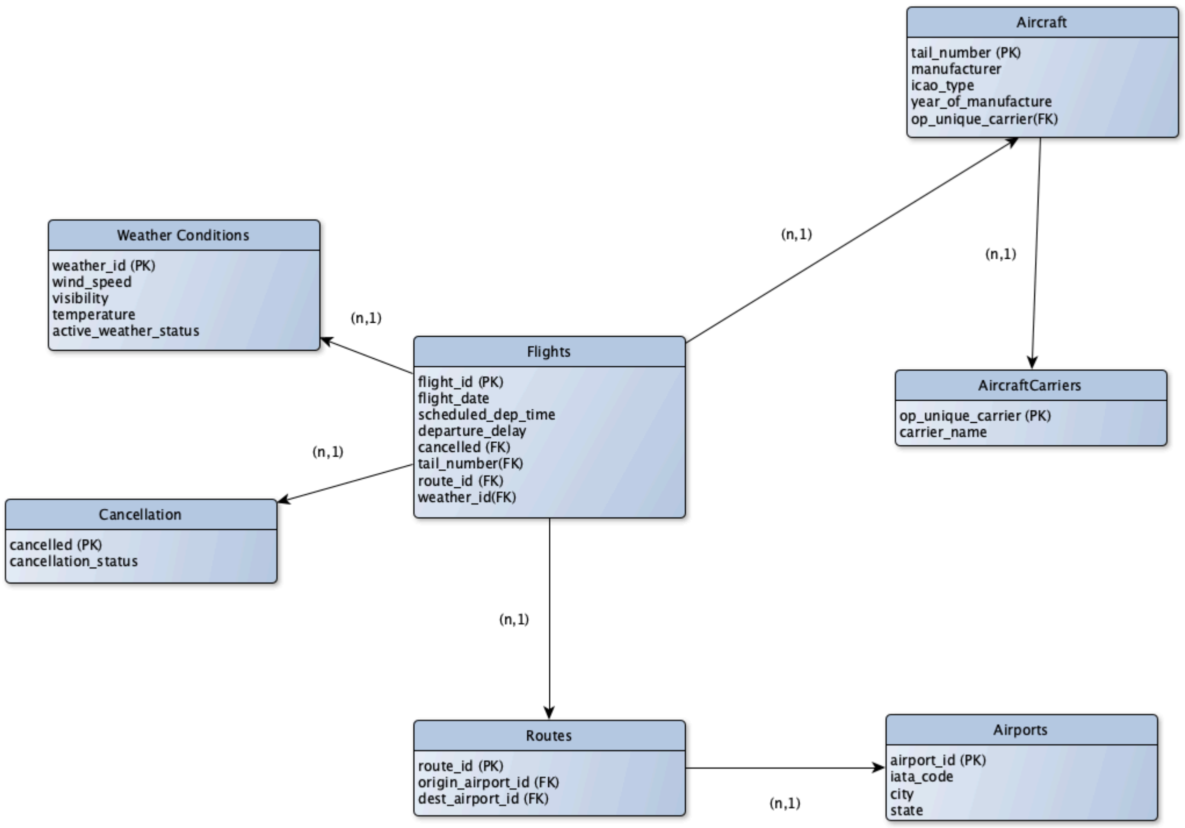
Scope: Covers all US domestic flights in 2022, combining:

- Departure performance
- Airport metadata
- Aircraft details
- Weather conditions

Data Size: 6,954,636 rows

Purpose in Project: Enables analysis of delay patterns, cancellation rates, and weather-driven scheduling behaviors

1.2 Choosing The Fact From The Reconciled Database



1.3 Create Attribute Tree

The chosen fact and root is: **Flight Events**

Each flight event represents a single scheduled flight operation with its associated measures and contextual information. This granularity was chosen because:

- It provides the most detailed level of analysis
- It captures all delay-related metrics at the flight level
- It allows aggregation across multiple dimensions (time, location, carrier, etc.)

Starting from the flight event fact, I identified the following attribute trees based on the functional dependencies in the data:

1. Date/Time Hierarchy:

- Flight Date
  - Date
  - Month
  - Quarter
  - Year (only one year)

2. Origin/Destination Airport Hierarchy:

- Airport
  - IATA code
  - City
  - State

3. Aircraft Hierarchy:

- Aircraft (Tail Number)
  - Manufacturer
  - ICAO Type
  - Year of Manufacture

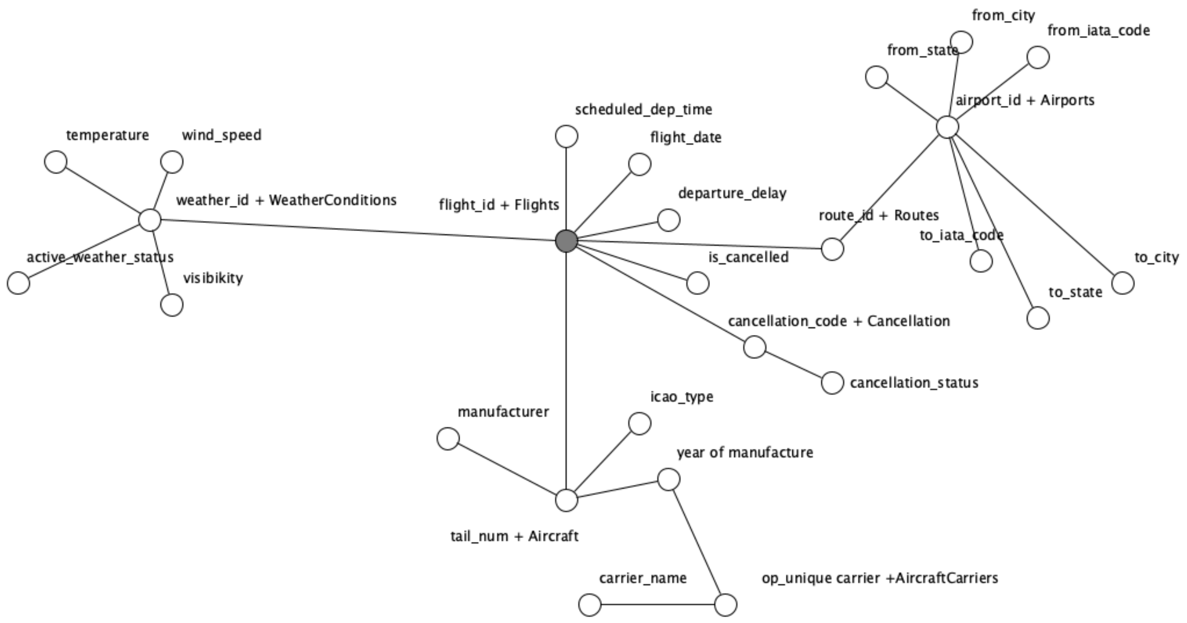
4. Weather Conditions Hierarchy:

- Weather Event
  - Wind Speed, Temperature, Visibility, Weather Status Description
- Necessary for understanding weather impact on delays

5. Cancellation Dimension:

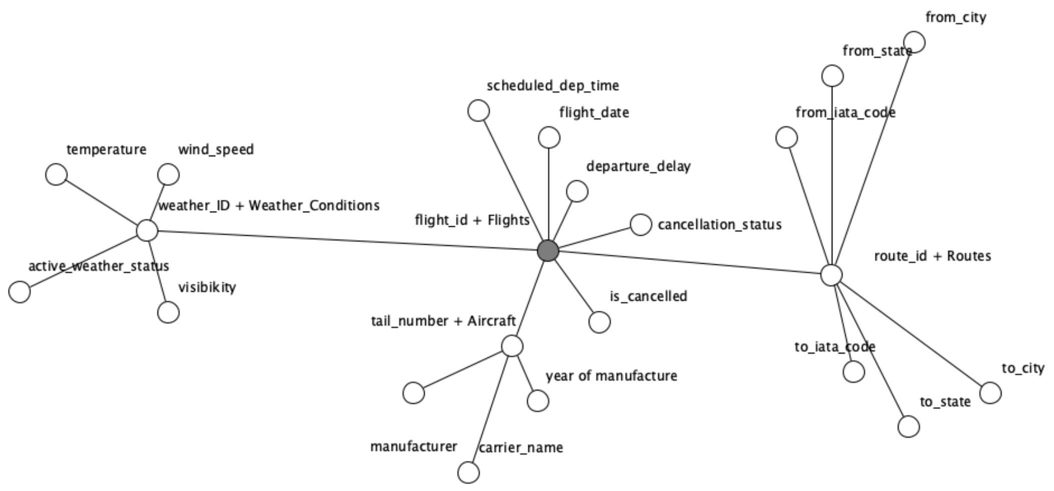
- Cancellation Code
  - Cancellation Status (Not Cancelled, Carrier, Weather, NAS, Security)
- Enables root cause analysis of cancellations

This design supports analysis of flight delays from multiple perspectives including time, position, carrier performance, weather impact, and cancellation reasons.



1.4 Create Pruned Tree

Apply pruning and grafting to intermediary nodes to get the following pruned attribute tree:

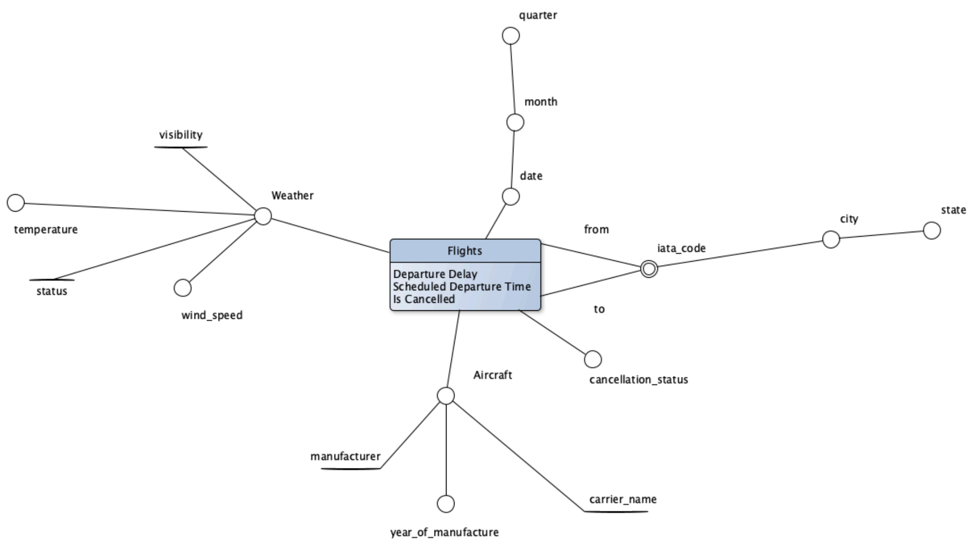


1.5 Create Fact Schema

This results in the final fact schema :

The measures of the fact are

- Departure Delay (minutes)
- Scheduled Departure Time
- Is Cancelled (binary indicator)



2. Data Cleaning

2.1 Extract Data

I used Python for the entire ETL process, because of its versatility, the vast amount of resources online and its compatibility with LLMs.

2.1.1 Select relevant columns from csv files

We need to drop multiple columns from our original dataset to reduce it to the ones we want for the final warehouse.

The following are the final and required columns for the warehouse:

- FL_DATE
- DEP_HOUR
- CRS_DEP_TIME
- DEP_DELAY
- CANCELLED
- TAIL_NUM
- MANUFACTURER
- ICAO TYPE
- YEAR OF MANUFACTURE
- OP_UNIQUE_CARRIER
- ORIGIN
- DEST
- WIND_SPD
- TEMPERATURE
- ACTIVE_WEATHER
- VISIBILITY

The original data for **carriers** only had the carrier code and name, therefore the removal of columns was not necessary.

These are the required columns from the **stations** file which holds the information for airport city and state:

- AIRPORT_ID
- DISPLAY_AIRPORT_CITY_NAME_FULL
- AIRPORT_STATE_NAME

2.1.2 Identify targets for cleaning (Transform)

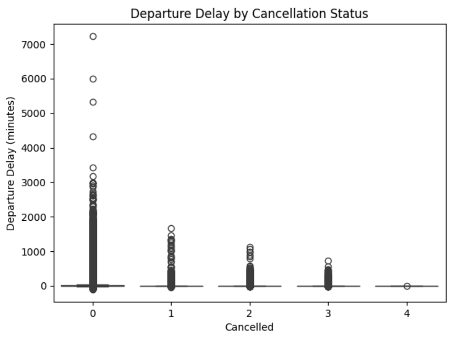
I started by dropping duplicate entries in the flights table:

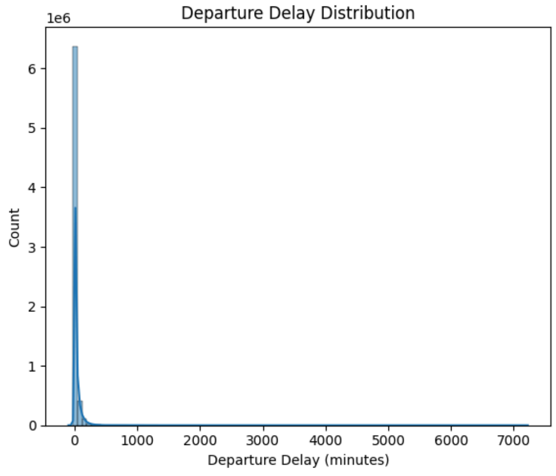
```
# Remove duplicate entries (check by flight time, tail number for plane and departure time (there cant be mult
filtered_flights_csv = remove_duplicates(filtered_flights_csv, subset=['FL_DATE','TAIL_NUM', 'CRS_DEP_TIME'])
```

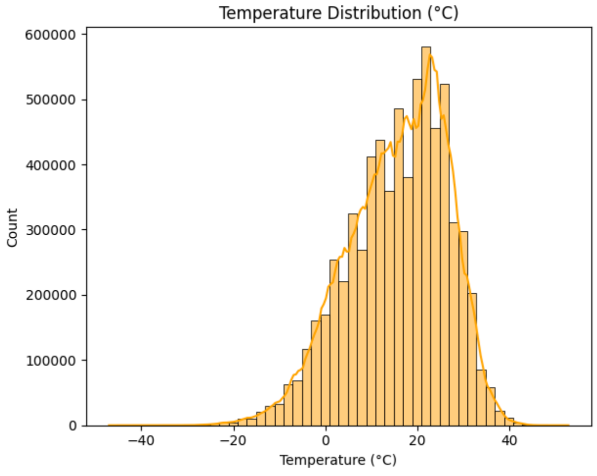
This depends on the date, tail number and departure time, since it should be impossible for the same plane to take off multiple times at the same date and time.

Next I created multiple plots to identify outliers. The delays by cancellation status did not show any unusual behaviour considering their meaning:

- 0 - Not Cancelled
- 1 - Carrier Cancellation
- 2 - Weather Cancellation
- 3 - National Air System Cancellation
- 4 - Security Cancellation







Departure Delay distribution and temperature distribution did not have extreme outliers either. However after some research I came to the conclusion to add some validation for both of these columns.

For **DEP_DELAY**, I set the minimum to -1 hour. Since the focus is on delays.

```
df = df[(df['DEP_DELAY'] >= -60)]
```

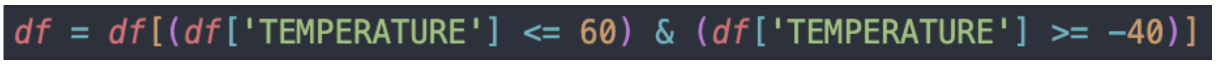
-> Removed 24 invalid rows after delay cleaning

Afterwards I removed flights which had a **DEP_HOUR** that was not set to 0 even though they were cancelled. This indicates a wrong entry since all cancelled entries should have their departure hour set to 0 (a flight is either delayed or cancelled)

```
df = df[~((df['DEP_HOUR'] > 0) & (df['CANCELLED'] > 0))]
```

-> Removed 4194 invalid rows after cancellation cleaning

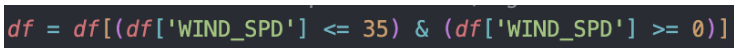
For **TEMPERATURE** I checked the hottest recorded temperature in 2022 and added it as an upper bound (Heatwave 2022). For the lower bound I took -40°C (US lowest recorded temperature 2022).



```
df = df[(df['TEMPERATURE'] <= 60) & (df['TEMPERATURE'] >= -40)]
```

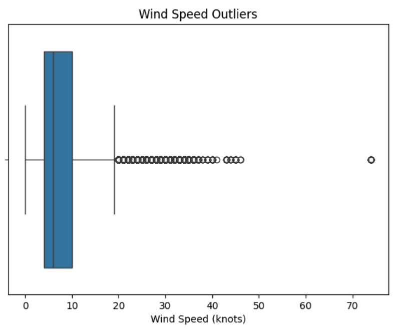
-> Removed 10 invalid rows after temperature cleaning

For **WIND_SPD** there were many outliers, which could skew the final measures for commercial flights. For this I took 35 knots as an upper bound (Cancellations Wind Speed). Additionally the wind speed should not be lower than 0 since thats not possible:

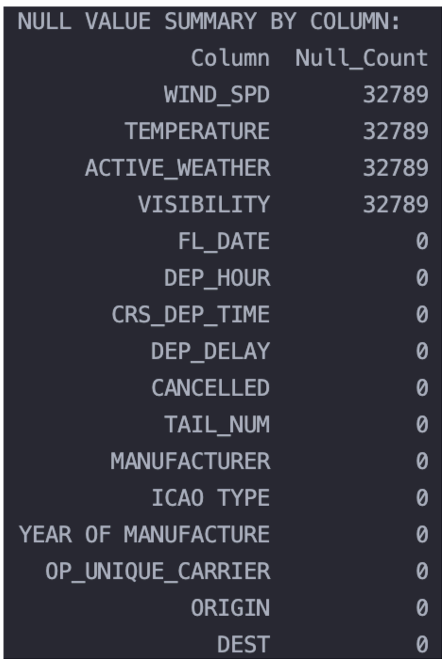


```
df = df[(df['WIND_SPD'] <= 35) & (df['WIND_SPD'] >= 0)]
```

-> Removed 206 invalid rows after wind speed cleaning



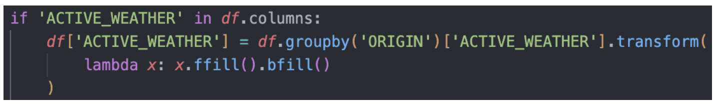
I created a function to analyse each column for the amount of null values it produces to Identify which entries are incomplete/faulty.



```
NULL VALUE SUMMARY BY COLUMN:
                 Column  Null_Count
               WIND_SPD       32789
            TEMPERATURE       32789
         ACTIVE_WEATHER       32789
             VISIBILITY       32789
                FL_DATE           0
               DEP_HOUR           0
           CRS_DEP_TIME           0
              DEP_DELAY           0
              CANCELLED           0
               TAIL_NUM           0
           MANUFACTURER           0
              ICAO TYPE           0
    YEAR OF MANUFACTURE           0
      OP_UNIQUE_CARRIER           0
                 ORIGIN           0
                   DEST           0
```

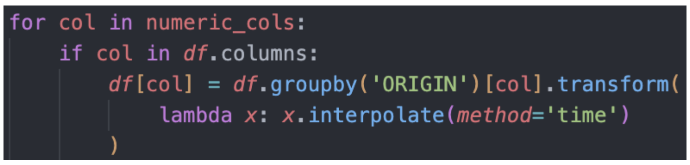
From this it is clear that there is only missing entries for the weather columns, likely due to the dataset not having corresponding weather data for all flight entries.

Since 32000 is a lot of entries I tried to create a function to fill these empty cells with accurate values.
For all weather cells I sorted the date by the date column so the previous and following events are as close as possible in a temporal context.
Since **ACTIVE_WEATHER** is a binary value I filtered by origin and used forward filling to get the most likely value.



```
if 'ACTIVE_WEATHER' in df.columns:
    df['ACTIVE_WEATHER'] = df.groupby('ORIGIN')['ACTIVE_WEATHER'].transform(
        lambda x: x.ffill().bfill()
    )
```

For **WIND_SPD**, **TEMPERATURE** and **VISIBILITY** I used pandas' interpolate function with the method "time" to get an estimate based on the closest entry for the same airport depending on the time.

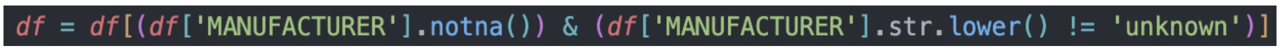


```
for col in numeric_cols:
    if col in df.columns:
        df[col] = df.groupby('ORIGIN')[col].transform(
            lambda x: x.interpolate(method='time')
        )
```

Afterwards there were still some weather related columns without values, which I removed from the data since filling them with incorrect values would just create noise in the data.

Amount of NaN values before interpolation: 32774
Amount of NaN values after interpolation: 3175

Finally I removed a few rows that had no specified manufacturer for the aircraft (“unknown”).



```
df = df[(df['MANUFACTURER'].notna()) & (df['MANUFACTURER'].str.lower() != 'unknown')]
```

-> Removed 9 invalid rows after manufacturer cleaning

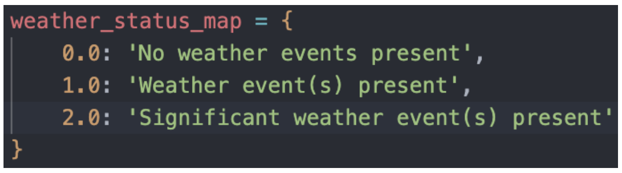
2.1.3 Load data into tables

For my reconciled database I settled for a csv file for each table. During the loading process I had to make some adjustments to the data.

Creating Dimension Tables:

I started by creating the **Date Dimension** from the unique flight dates and added the temporal attributes (month, quarter, year) needed for hierarchical analysis.

For the **Weather Dimension**, I mapped the numeric ACTIVE_WEATHER codes to descriptive text:



```
weather_status_map = {
    0.0: 'No weather events present',
    1.0: 'Weather event(s) present',
    2.0: 'Significant weather event(s) present'
}
```

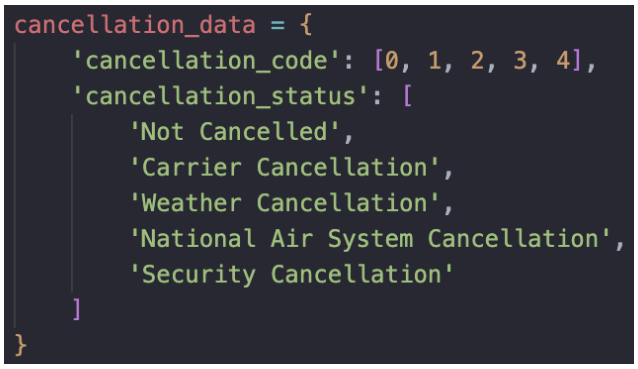
I then created unique combinations of all weather attributes (wind speed, temperature, visibility, weather status. Moreover I rounded the the values for the numeric columns to ease use and readability in Tableau.

The **Aircraft Dimension** was created by extracting unique aircrafts based on tail number, keeping the manufacturer, ICAO type, and year of manufacture. This produced 6,129 unique aircraft records.

For the **Carrier Dimension**, I merged the carrier codes with their descriptive names from the reference data. Additionally I renamed the column “DESCRIPTION” to “carrier_name” to reflect the schema.

The **Airport Dimension** required generating surrogate keys (airport_id) while keeping the IATA codes as attributes. I filtered to only include the 376 airports that appeared in the flight data.

Finally, I created a **Cancellation Dimension** as a simple lookup table mapping the codes 0-4 to their business meanings.



```python
cancellation_data = {
    'cancellation_code': [0, 1, 2, 3, 4],
    'cancellation_status': [
        'Not Cancelled',
        'Carrier Cancellation',
        'Weather Cancellation',
        'National Air System Cancellation',
        'Security Cancellation'
    ]
}
```

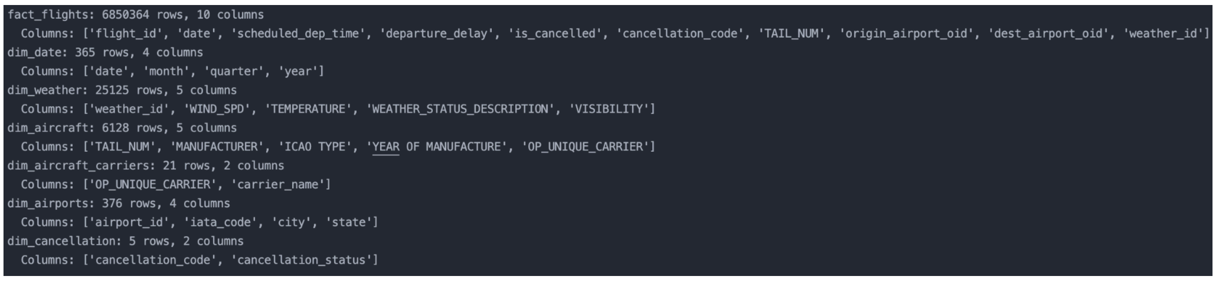
Creating the Fact Table:

For the fact table, I generated a sequential flight_id as the primary key. I then replaced all natural keys with their corresponding foreign keys:

- Joined origin and destination IATA codes to get airport_ids
- Matched weather conditions on all four attributes to get weather_id

I added the measure **is_cancelled** as a binary indicator (0 or 1) based on the cancellation code.

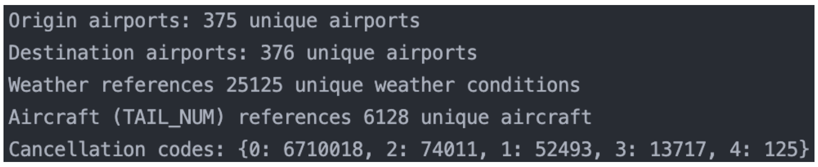
This resulted in the following tables for the star schema



```
fact_flights: 6850364 rows, 10 columns
  Columns: ['flight_id', 'date', 'scheduled_dep_time', 'departure_delay', 'is_cancelled', 'cancellation_code', 'TAIL_NUM', 'origin_airport_oid', 'dest_airport_oid', 'weather_id']
dim_date: 365 rows, 4 columns
  Columns: ['date', 'month', 'quarter', 'year']
dim_weather: 25125 rows, 5 columns
  Columns: ['weather_id', 'WIND_SPD', 'TEMPERATURE', 'WEATHER_STATUS_DESCRIPTION', 'VISIBILITY']
dim_aircraft: 6128 rows, 5 columns
  Columns: ['TAIL_NUM', 'MANUFACTURER', 'ICAO TYPE', 'YEAR OF MANUFACTURE', 'OP_UNIQUE_CARRIER']
dim_aircraft_carriers: 21 rows, 2 columns
  Columns: ['OP_UNIQUE_CARRIER', 'carrier_name']
dim_airports: 376 rows, 4 columns
  Columns: ['airport_id', 'iata_code', 'city', 'state']
dim_cancellation: 5 rows, 2 columns
  Columns: ['cancellation_code', 'cancellation_status']
```

One issue I encountered was that the airport IDs were being stored as floats, so I explicitly converted them to Int64 to handle potential null values while maintaining integer types.

Finally I verified the foreign keys and saved the tables into csv files:



```
Origin airports: 375 unique airports
Destination airports: 376 unique airports
Weather references 25125 unique weather conditions
Aircraft (TAIL_NUM) references 6128 unique aircraft
Cancellation codes: {0: 6710018, 2: 74011, 1: 52493, 3: 13717, 4: 125}
```

3. Data Visualization

3.1 The Story

My goal was to create a data warehouse and a visualization which enable the user to identify root causes for flight delays and cancellations. It should serve as a foundation to identify correlations between various dimensions in the database.

The analysis for this visualization is mostly focused on weather, airlines and individual aircrafts. However it is also possible to do a location based analysis which I will demonstrate as part of my visualization.

3.2 Analysis Plan: Flight Cancellations and Delays

3.2.1. Overview: Reasons for Flight Cancellations

I will begin by presenting a surface level breakdown of flight cancellation reasons, distinguishing between carrier-related, weather-related, national air system, and security cancellations.
This overview (see "Reasons for flight cancellations" dashboard) sets the stage for deeper analysis by quantifying the relative impact of each factor and highlighting that weather and carrier issues are the predominant causes.

3.2.2. Weather's Double Impact: Cancellations and Delays

Next, I will analyze how weather conditions—specifically temperature, weather events, wind speed, and visibility—affect both cancellation rates and average departure delays. The "Weather Dashboard" visualizes these relationships:

- Temperature & Weather Events: I'll show that extreme temperatures and significant weather events (e.g., storms) are associated with sharp increases in both cancellations and delays.
- Wind Speed & Visibility: The data suggests that while cancellation rates rise slightly with increasing wind speed and low visibility, the effect is less pronounced than for temperature extremes or major weather events. Especially since most flights dont seem to get scheduled at all for low visibility and high wind speeds

3.2.3. Carrier Performance: Delays vs. Cancellations

I will compare airlines by both average departure delay and cancellation rate ("Carrier Performance Dashboard"). This dual-metric approach helps identify carriers that consistently outperform or underperform their peers, and opens discussion on possible contributing factors such as operational efficiency, hub locations, or fleet management.
This view also allows roll-down functionality to help identify patterns and reflect specific events for airlines (e.g. Southwest Airlines Scheduling Crisis)

3.2.4. Impact of aircraft age on flight cancellation

Using the "Carrier Age Dashboard" and related sheets, I will explore how cancellation rates vary by aircraft year of manufacture and by carrier. This section will address whether older aircraft are more prone to cancellations and how different carriers' fleet compositions might influence their operational reliability. The treemap visualization further clarifies which aircraft age groups and carriers are most affected.

3.2.5. Regional hotspots for each month

Finally, I will map cancellation rates by state, city, and airport ("Flight Cancellation Rates by State/City/Airport") using roll-down functionality. This highlights potential hotspots, providing insight into how local factors and time-of-year impact cancellation rates.