University of Milano-Bicocca
School of Science
Department of Informatics, System and Communication
Master degree in Data Science

How weather affects airline flight delays from Malpensa.

Report for Data Management project

Andrea Muscio [889901], Alberto Porrini [826306], Francesco Paolo Luigi Caruso [906416]

20 June 2023- academic year 2022-2023

Contents

1 Introduction

Airline flight delays are a daily occurrence worldwide, with enormous impacts in terms of inconvenience and economic costs.

The causes can be multiple, with one of them being weather conditions. The increasing frequency and intensity of extreme weather events due to climate change necessitates a more in-depth understanding of the relationship between weather and air travel.

In this project, we aim to examine the correlation between weather patterns and airline flight delays.

We have chosen to focus on departure delays from Milan-Malpensa Airport, as it is the second-largest airport in Italy in terms of passenger and flight numbers. It is also one of the major European and global hubs. Additionally, its location in the Po Valley and proximity to the Alpine region make it interesting to study the potential implications of weather conditions in the area.

Regarding the time frame, we have decided to concentrate on the month of March 2023, encompassing exactly four weeks. We believe this period provides sufficient time for a preliminary study, and it exhibits relevant weather variability. In future could be possible to follow the guidelines of this project to execute a large scale study.

1.1 Objective

The objective of this project is to discover which weather conditions have the greatest impact on flight delays at Malpensa airport, and to determine if there are any airline companies that are particularly prone to such delays.

So the questions we want to answer are:

1. What are the weather conditions that have the greatest impact on these delays?
2. For each wind condition, what is the average delay recorded?
3. For each weather condition, what are the top 3 routes (and therefore the destinations) that have recorded the highest average delay? For

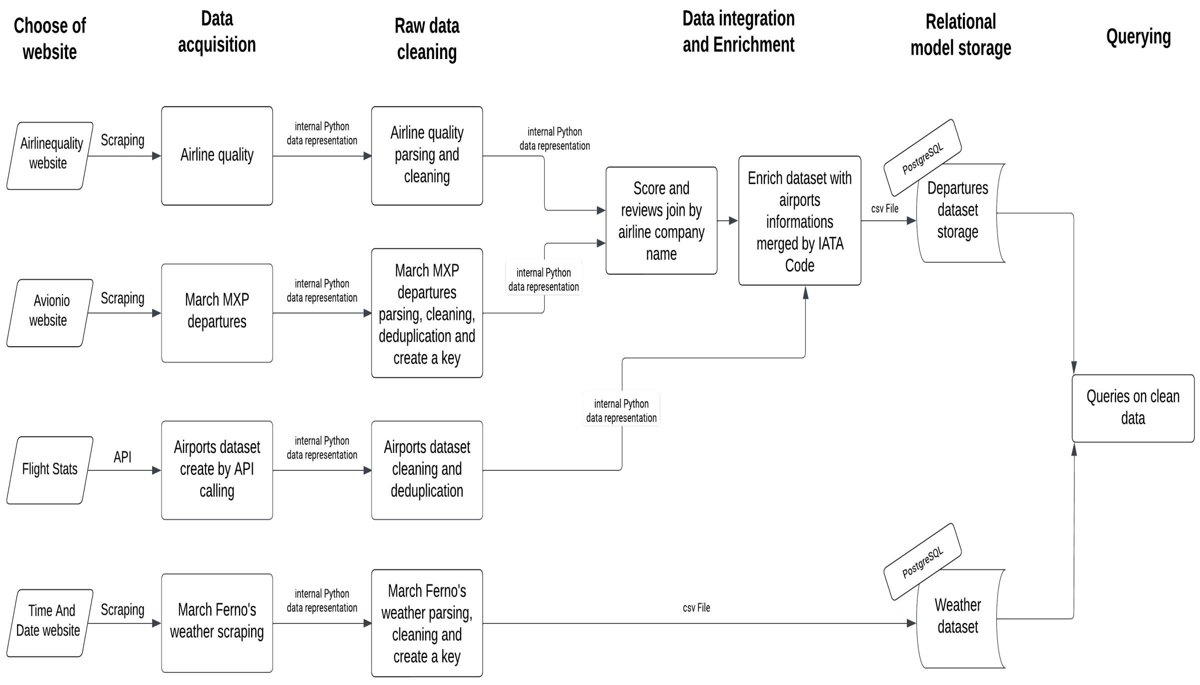
each of these routes and the corresponding arrival airports, what is the IATA code, ICAO code, and official airport name?

4. For each wind condition, what are the top 4 airlines that have recorded the highest average delay? For each of these airlines, what is the score and how many reviews do they have?

The research, if carried out on a larger scale, could potentially help consumers in selecting an airline company, while also aiding airline companies in paying greater attention to flight planning in relation to weather forecasts. Additionally, it may inspire the purchase of additional insurance coverage for flight delays due to weather conditions.

1.2 Solution

This is the workflow of our proposed solution. In the following chapters we will explain every step and the choices we made.


Choose of website
Data acquisition
Raw data cleaning
Data integration and Enrichment
Relational model storage
Querying
Airlinequality website
Scraping
Airline quality
internal Python data representation
Airline quality parsing and cleaning
Score and reviews join by airline company name
Enrich dataset with airports informations merged by IATA Code
csv File
PostgreSQL
Departures dataset storage
Avionio website
March MXP departures
March MXP departures parsing, cleaning, deduplication and create a key
Flight Stats
API
Airports dataset create by API calling
Airports dataset cleaning and deduplication
Queries on clean data
Time And Date website
March Ferno's weather scraping
March Ferno's weather parsing, cleaning and create a key
Weather dataset

2 Data Aquisition

Data acquisition is a critical aspect of any research project, and its importance cannot be overstated. Accurate and reliable data serves as the foundation for analysis, decision-making, and drawing meaningful insights. In our particular study, we recognized the significance of acquiring comprehensive and up-to-date information for flight departures from Malpensa Airport and weather conditions during the specified period.

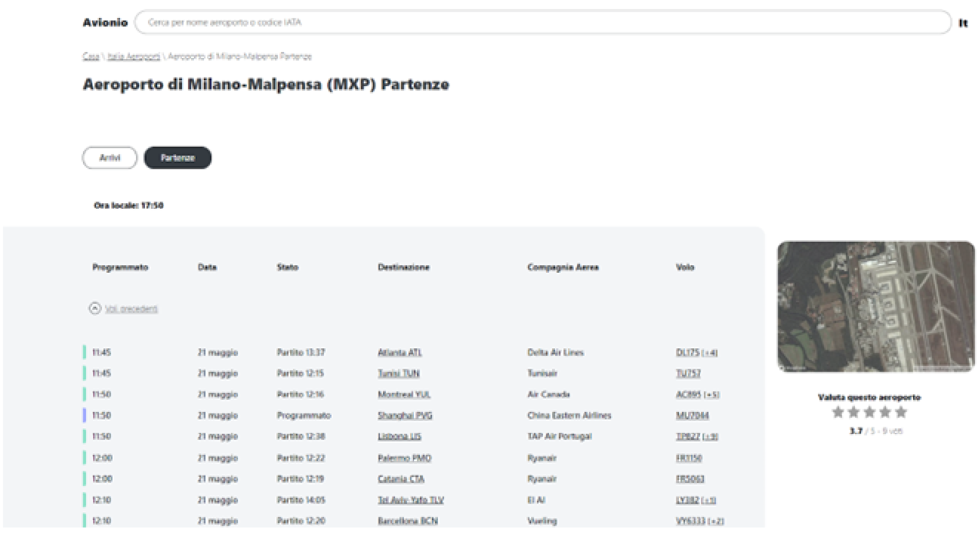
We made a deliberate choice to scrape data from the Avionio website for several reasons. Avionio is a renowned platform that provides comprehensive and real-time flight information. By extracting data from Avionio, we ensured access to a vast dataset containing various attributes of flight departures from Malpensa. This rich and diverse dataset allows us to explore patterns, trends, and relevant factors that contribute to a deeper understanding of the flight operations from the airport.

Furthermore, to incorporate weather conditions into our analysis, we turned to the trusted source of Time And Date. Their platform provides accurate and detailed weather information for specific locations and dates. By incorporating weather data, we can examine how meteorological factors influence flight schedules and potential disruptions during the chosen period.

To enhance the comprehensiveness of our research, we also utilized the Flight Stats API for obtaining airport-related information. This API offers a wide range of data, including airport codes, locations, and additional details that are instrumental in enriching our analysis of the flights departing from Malpensa. By integrating data from the Flight Stats API, we gain valuable insights into the airports involved, that could be useful for future employment.

2.1 Departures scraping

We have considered the website avionio.com, where you can find all the departures and arrivals, along with various information specified for each civilian airport worldwide.


Avionio
Cerca per nome aeroporto o codice IATA
It
Casa \ Italia Aeroporti \ Aeroporto di Milano-Malpensa Partenze
Arrivi
Partenze
Voli precedenti
Valuta questo aeroporto
3.7 / 5 - 9 voti

Aeroporto di Milano-Malpensa (MXP) Partenze

Ora locale: 17:50

Programmato	Data	Stato	Destinazione	Compagnia Aerea	Volo
11:45	21 maggio	Partito 13:37	Atlanta ATL	Delta Air Lines	DL175 (+4)
11:45	21 maggio	Partito 12:15	Tunisi TUN	Tunisair	TU757
11:50	21 maggio	Partito 12:16	Montreal YUL	Air Canada	AC895 (+5)
11:50	21 maggio	Programmato	Shanghai PVG	China Eastern Airlines	MU7044
11:50	21 maggio	Partito 12:38	Lisbona LIS	TAP Air Portugal	TP827 (+9)
12:00	21 maggio	Partito 12:22	Palermo PMO	Ryanair	FR1150
12:00	21 maggio	Partito 12:19	Catania CTA	Ryanair	FR5063
12:10	21 maggio	Partito 14:05	Tel Aviv-Yafo TLV	El Al	LY382 (+1)
12:10	21 maggio	Partito 12:20	Barcellona BCN	Vueling	VY6333 (+2)

The objective was to retrieve data from the website for all departures from Milan Malpensa Airport from March 1st to March 28th (we chose a reference period of 4 weeks in March since the data is updated every 28 days).

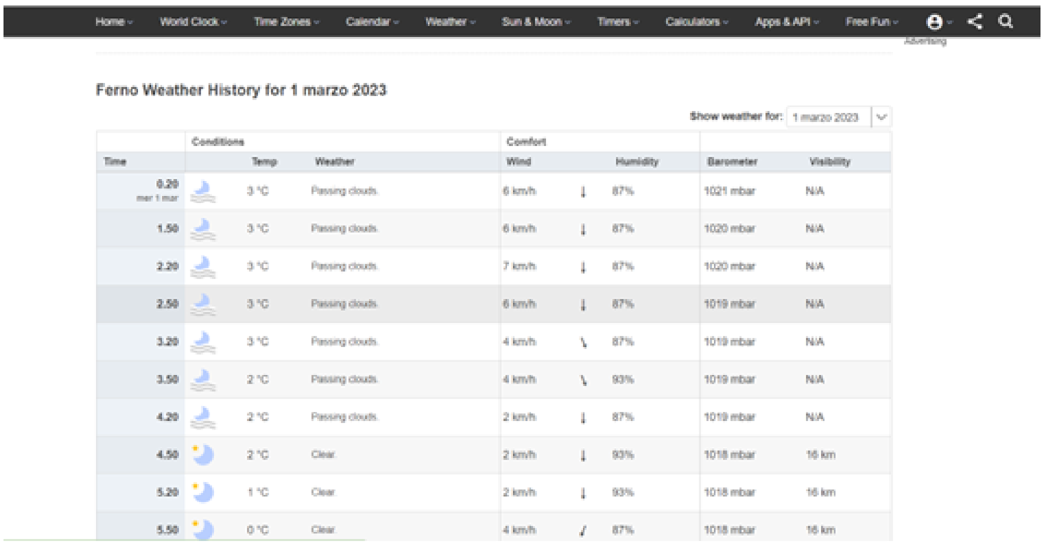
Using web scraping techniques requests and BeautifulSoup libraries, we extracted the values of the fields of interest related to air departures from Milan Malpensa Airport from the Departures URL. We then implemented these values into lists created specifically for each of the six columns, with the aim of creating a DataFrame containing all the columns. Finally, we exported the dataset in .csv format, resulting in the file named ”departures_2803.csv”.

This raw file was subsequently manipulated for data cleaning operations. Specifically, for each flight, the following data was obtained:

- Date : the day of the month.
- Time : the scheduled departure time.
- Status : information about the actual or missed departure of the flight.
- Arrival_Airport : the destination of the flight.
- Airline : the airline operating the flight.
- Flight : an alphanumeric code that identifies the flight.

2.2 Weather scraping

We have considered the website timeanddate.com, which, in the Weather section, provides current weather information (Worldwide, Local Weather), forecasts (2-Week Forecast and Hour-by-Hour), past weather data (Yesterday-Past Weather), and average climate information (Climate, Averages) for each location.



Ferno Weather History for 1 marzo 2023

	Conditions		Comfort				
Time	Temp	Weather	Wind		Humidity	Barometer	Visibility
0.20 mer 1 mar	3 °C	Passing clouds.	6 km/h	↓	87%	1021 mbar	N/A
1.50	3 °C	Passing clouds.	6 km/h	↓	87%	1020 mbar	N/A
2.20	3 °C	Passing clouds.	7 km/h	↓	87%	1020 mbar	N/A
2.50	3 °C	Passing clouds.	6 km/h	↓	87%	1019 mbar	N/A
3.20	3 °C	Passing clouds.	4 km/h	↘	87%	1019 mbar	N/A
3.50	2 °C	Passing clouds.	4 km/h	↘	93%	1019 mbar	N/A
4.20	2 °C	Passing clouds.	2 km/h	↓	87%	1019 mbar	N/A
4.50	2 °C	Clear.	2 km/h	↓	93%	1018 mbar	16 km
5.20	1 °C	Clear.	2 km/h	↓	93%	1018 mbar	16 km
5.50	0 °C	Clear.	4 km/h	↙	87%	1018 mbar	16 km

Specifically, we are interested in the section of the historical weather archive (Past-Weather) to retrieve information relevant to the desired time period.

The objective was to retrieve weather data for the specified time period from the website timeanddate.com, specifically from the ”Weather - Past Weather” section. To obtain accurate meteorological information for Milan Malpensa Airport, we chose the municipality of Ferno (VA), one of the four municipalities covering the airport area.

Web scraping from the Weather URL was performed using the requests and BeautifulSoup libraries, similar to the process for retrieving flight departure data. We extracted the relevant values, stored them in lists, created a DataFrame, and finally exported the dataset in .csv format.

The dataset included the following hourly data for each step:

- `Hours` : the time of the weather observation.
- `Temperature` : the recorded temperature.
- `Weather` : the description of the weather condition (e.g., sunny, fog...).
- `Wind` : information about the wind speed.
- `Humidity` : the recorded humidity level.
- `Barometer` : atmospheric pressure.
- `Visibility` : distance at which an object can be clearly seen.
- `Day` : the date of the weather observation.

This dataset provides historical weather information for the specified time period at Milan Malpensa Airport, based on the data retrieved from the ”Weather - Past Weather” section of timeanddate.com.

2.3 Airports API

To integrate the FlightStats API into our project, we implemented a series of API calls that resulted in the creation of a dictionary containing various elements.

We used the key "airport" to store the information retrieved from the API. The process involved establishing a for loop that iterated over a list of IATA codes present in our main dataset, "departures". Within the loop, we made API requests using the FlightStats API to obtain detailed information about each airport corresponding to the IATA code.

This information included essential data such as airport codes, locations, operating airlines, and flight schedules. By leveraging the power of the FlightStats API, we were able to gather comprehensive and up-to-date data for each airport.

The loop iterated through each IATA code in the list, and for each iteration, it called the API with the corresponding code as a parameter. The API responded with a JSON object containing the relevant airport information.

We then extracted the necessary details from the JSON response and stored them in the "airport" dictionary, associating each airport's IATA code as the key.

This approach allowed us to enrich our dataset, "departures", with comprehensive information about each airport, providing valuable context for analyzing and understanding the departure data.

The integration of the airports dataset enhanced the accuracy and completeness of our analysis, enabling us to gain deeper insights into flight departures and related factors.

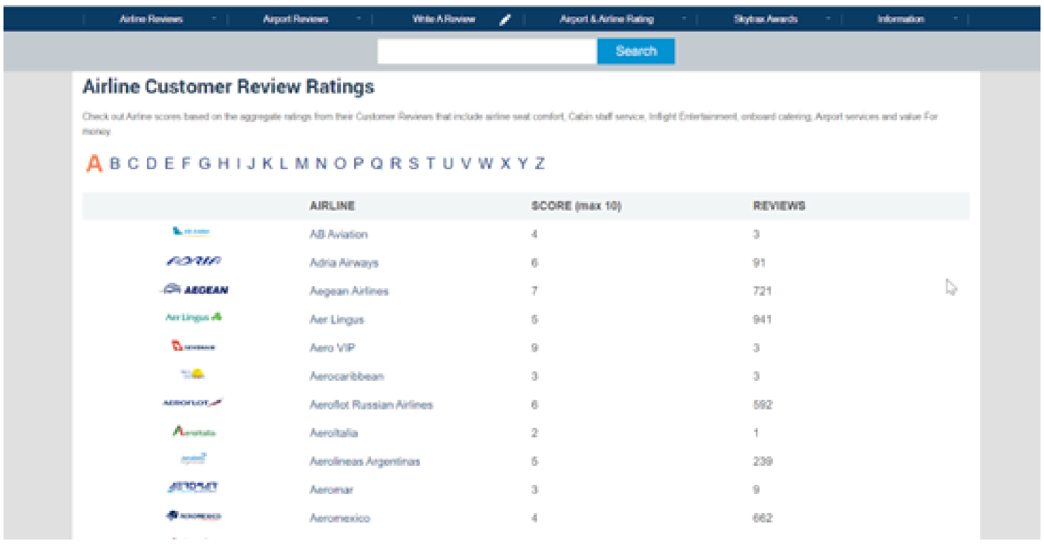
From the API call, we obtained the following information:

- `fs` : FlightStats airport code
- `iata` : International Air Transport Association airport code
- `icao` : International Civil Aviation Organization airport code
- `faa` : Federal Aviation Administration airport code
- `name` : Airport name
- `city` : City where the airport is located

- cityCode : City code where the airport is located
- stateCode : State code where the airport is located
- countryCode : Country code where the airport is located
- countryName : Country name where the airport is located
- regionName : Continent where the airport is located
- timeZoneRegionName : Time zone of the airport
- weatherZone : Weather zone code where the airport is located
- localTime : Local time at the airport at the time of data acquisition
- utcOffsetHours : Time zone offset of the airport
- latitude : Latitude of the airport
- longitude : Longitude of the airport
- elevationFeet : Elevation of the airport above sea level
- classification : Classification of airport services (ranging from 1 to 4) based on traffic and importance, assigned by FlightStats
- active : Airport status (”True” for active, ”False” for inactive)
- street1 : Primary address of the airport
- postalCode : Postal code of the airport
- street2 : Secondary address of the airport

2.4 Airline quality scraping

The website airlinequality.com allows users to assign a score from 1 to 10, evaluating the quality of service received from a particular airline. Users can provide these reviews after demonstrating that they have traveled with the airline by registering their ticket in the appropriate section.



Airline Customer Review Ratings

Check out Airline scores based on the aggregate ratings from their Customer Reviews that include airline seat comfort, Cabin staff service, Inflight Entertainment, onboard catering, Airport services and value For money.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

AIRLINE	SCORE (max 10)	REVIEWS
AB Aviation	4	3
Adria Airways	6	91
Aegean Airlines	7	721
Aer Lingus	5	941
Aero VIP	9	3
Aerocaribbean	3	3
Aeroflot Russian Airlines	6	592
Aeroitalia	2	1
Aerolineas Argentinas	5	239
Aeromar	3	9
Aeromexico	4	662

Additionally, the website keeps track of the total number of reviews for each airline. This refers to the number of reviews provided by users specifically for that airline.

The objective was to retrieve information about the quality of airlines based on user reviews from the website airlinequality.com. The website provides an ”score” and the number of ”reviews” for each airline.

Web scraping from the Airline Quality URL was conducted in a similar manner to the previous examples. A dataset was created in .csv format, containing the following information for each airline:

- Airline : The name of the airline.
- Score : A score ranging from 1 to 10.
- Reviews : The number of reviews received for the airline.

3 Data profiling

Below, we analyze the structure and characteristics of the four datasets obtained through the web scraping process (departures, weather, and airline quality) and via API (airports).

3.1 Departures data

The data is structured in a table containing the previously mentioned information and it consists of 11,439 observations.

We also note that the dataset includes flights with different statuses such as departed, canceled, scheduled, and unknown.

Additionally, we observed that some flights shared the same routes, actual departures times, and dates, with only the airline varying. We found that this was a case of data redundancy, and the first airline listed was the one actually operating the route (these are cases of code sharing as described better in cap.5.2).

Lastly, we observe that dataset does not contain any null values.

3.2 Weather data

The data is structured in a table containing the aforementioned information. There are 1,206 observations in total. The website provides detailed information every 30 minutes of each day, although some time slots on certain days, as specified later, are missing. However, our main interest lies in obtaining weather information closest to the scheduled departure time of the aircraft. Therefore, we conducted a subsequent search for such information, which represented the actual weather conditions at the desired moment.

The data provides both qualitative indications of weather conditions (sunny, fog, clear) and quantitative physical measurements such as temperature, wind speed, humidity,atmospheric pressure and visibility. We noticed that these measurements are in units of measurement that are not typically used in Italy and were subsequently converted.

Lastly, it should be noted that approximately 52% of the visibility values are missing.

3.3 Airports data

The dataset contains information about 165 airports distributed worldwide and consists of 20 columns. It includes various data related to airports, such as identifying details like IATA and ICAO codes, airport names, cities, regions, countries, and other characteristics. Additionally, the dataset provides information about the time zone, UTC offset, geographic coordinates (latitude and longitude), airport elevation, and airport classification.

Note that some columns may have missing values.

3.4 Airline Quality data

The dataset consists of a table with the previously mentioned information and contains 548 observations. The "score" rating, ranging from 1 to 10, is assigned by users who have demonstrated that they have traveled with the airline by registering their ticket. The "reviews" column indicates the number of user reviews related to the airline. Therefore, the "score" provides an indication of the quality, while the "reviews" give an indication of the reliability of the score, as a higher number of user votes makes the rating more reliable.

Note that cargo airlines are not included in the dataset, as well as some less commonly used airlines. Cargo airlines cannot be reviewed, and evidently, some less commonly used airlines are not present on the review website.

Finally, it is worth mentioning that the dataset does not contain any missing values.

4 Data quality

The quality dimensions that have been taken into account are: accuracy, completeness, currency and consistency.

4.1 Accuracy

Accuracy evaluates how well the data represents the real phenomenon.

In our case, we choose the Avionio website, which communicates with the Malpensa site and retrieves validated departure flights from Malpensa airport. Therefore, we can say that data are highly reliable.

Regarding the airlines quality, the reviews are truthful because users must upload evidence of their trip (flight ticket).

Similarly, for the weather dataset, due to the nature of the data, which comprises measurements of meteorological variables taken at approximately regular intervals, there can be discrepancies in weather conditions. For example, temperature measurements are subject to measurement uncertainties caused by the sensitivity of the instruments used, such as barometers. The website used has been operational since 1995 and is highly reputable, however, it is not possible for us to evaluate the accuracy of these measurements.

Airports data are validate from Flight Stats, one of the primary API providers for flights applications, we can consider our data accurate.

4.2 Completeness

Completeness has been evaluated at the single attribute level for each dataset.

- For the dataset of **weather** obtained from the **Time And Date** site, on a total of 1206 rows and 8 columns:

Attribute	Total NaN	% NaN
Hours	0	0,00
Temperature	0	0,00
Weather	0	0,00
Wind	0	0,00
Humidity	0	0,00
Barometer	0	0,00
Visibility	625	0,52
Day	0	0,00

The percentage of completeness of this dataset is 93.5% and all null values are focus on the visibility attribute.

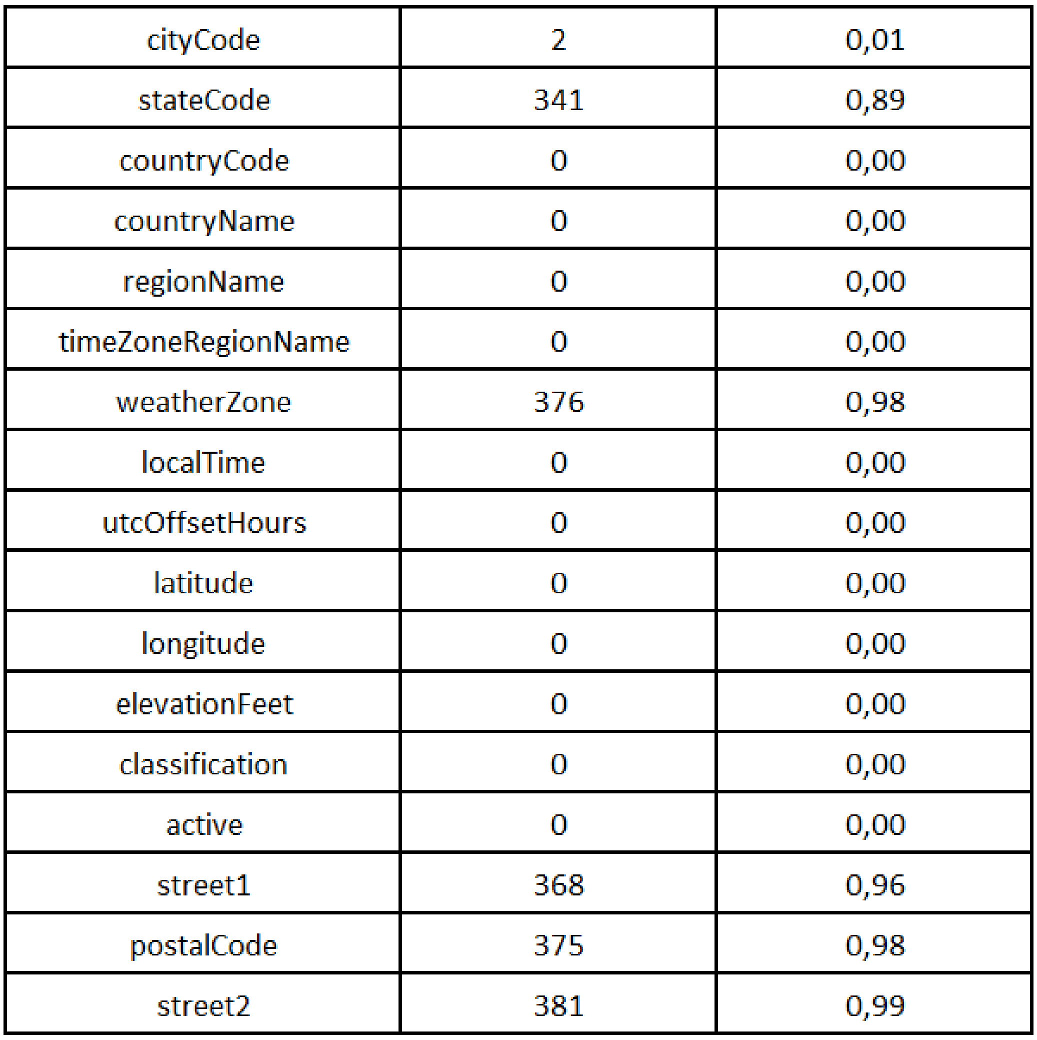
- For the dataset of **departures** obtained from the **Avionio** site, on a total of 11439 rows and 6 columns:

Attribute	Total NaN	% NaN
Date	0	0,00
Time	0	0,00
Status	0	0,00
Arrival_Airport	0	0,00
Airline	0	0,00
Flight	0	0,00

The percentage of completeness of this dataset is 100%. The site necessarily must have all of these pieces of information about a flight, so there are no null values.

- For the dataset of **airports** obtained from **Flight Stats API** , on a total of 383 rows and 23 columns:

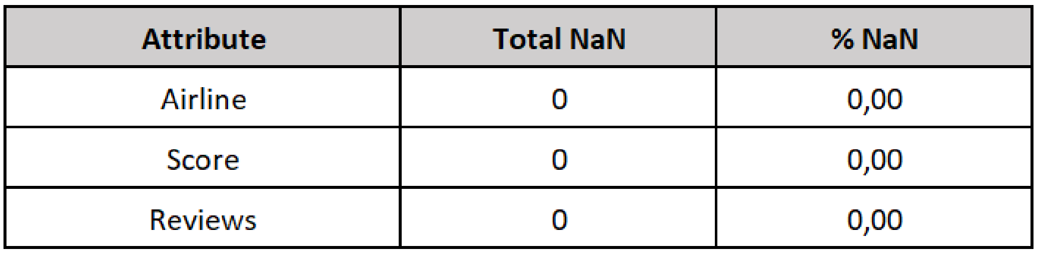
Attribute	Total NaN	% NaN
fs	0	0,00
iata	0	0,00
icao	2	0,01
faa	376	0,98
name	0	0,00
city	0	0,00



cityCode	2	0,01
stateCode	341	0,89
countryCode	0	0,00
countryName	0	0,00
regionName	0	0,00
timeZoneRegionName	0	0,00
weatherZone	376	0,98
localTime	0	0,00
utcOffsetHours	0	0,00
latitude	0	0,00
longitude	0	0,00
elevationFeet	0	0,00
classification	0	0,00
active	0	0,00
street1	368	0,96
postalCode	375	0,98
street2	381	0,99

The percentage of completeness of this dataset is 75%. We have some attribute such as faa, stateCode, weatherZone, street1, postalCode, street2 that are not of interest for our purpose or are redundant. Therefore, it doesn't matter if they are null.

- For the dataset of **airline quality** obtained from **Airlinequality** site, on a total of 548 rows and 3 columns:



Attribute	Total NaN	% NaN
Airline	0	0,00
Score	0	0,00
Reviews	0	0,00

The percentage of completeness of this dataset is 100%.

4.3 Currency

We know that currency measures how quickly data is updated with respect to the real phenomenon.

The data for Malpensa departures are continuously updated minute by minute, as well, the weather measurements are taken every 30 minutes to record the variations in weather and store on a daily basis, which are crucial for air traffic.

Regarding the airlines quality dataset, the site from which we scrape the data is updated each time a customer writes a review.

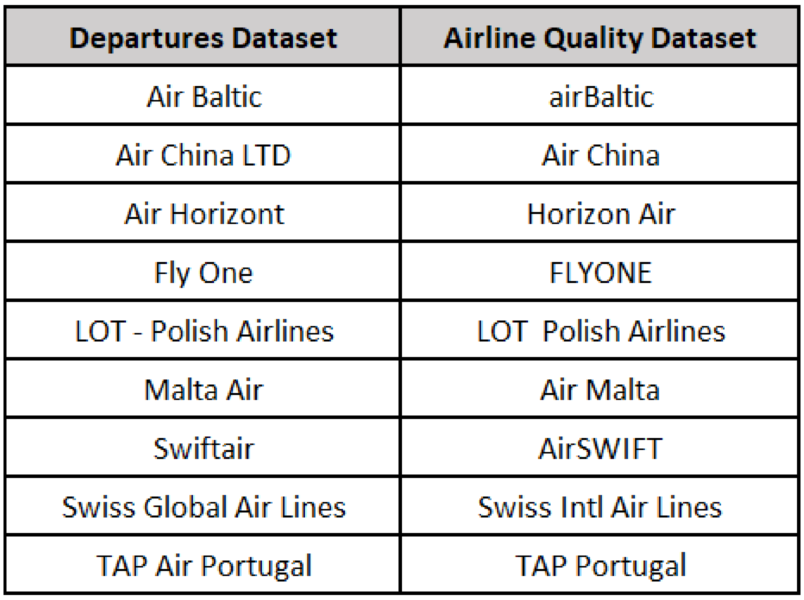
The airport dataset is updated only if an airport changes its name, its not active, or a new airport is inaugurated, as the airport codes and their geolocation remain unchanged.

4.4 Consistency

Consistency evaluates how well the different representations of the same object in the two datasets match.

In our case, we can only compare information of interest from two different sources: the airline names and the airport codes (IATA). This data has been obtained from the "departures" dataset, "Airline Quality" dataset and "Airports" dataset.

Regarding the airline names, some are reported slightly differently between the ”departures” dataset and the ”Airline Quality” dataset. Below, we list the airlines that have slightly different names in the two datasets (the first is from ”departures” and the second is from ”Airline Quality”).



Departures Dataset	Airline Quality Dataset
Air Baltic	airBaltic
Air China LTD	Air China
Air Horizont	Horizon Air
Fly One	FLYONE
LOT - Polish Airlines	LOT Polish Airlines
Malta Air	Air Malta
Swiftair	AirSWIFT
Swiss Global Air Lines	Swiss Intl Air Lines
TAP Air Portugal	TAP Portugal

Regarding the Airports codes (IATA) there is a consistency between the both datasets,since IATA is determined by a international aviation organization.

5 Data cleaning e Data Enrichment/Integration

We have performed data cleaning operations on our dataset, which is the process of identifying, correcting, and removing errors, inconsistencies, and incomplete or duplicate data.

Furthermore, we have decided to add additional information about airlines and airports (using the Airports and Airline quality datasets) to the original flight dataset (departures).

Below, we describe the cleaning and enrichment/integration operations for each dataset, using the Python programming language.

5.1 Dataset weather

We have performed several operations on the weather dataset, which we describe in detail below. Firstly, we have carried out unit conversions for some meteorological measurements, specifically:

- The temperature in Fahrenheit has been converted to Celsius.
- The wind speed in miles per hour has been converted to kilometers per hour.
- The atmospheric pressure measurement in inches of Mercury (mmHg) has been converted to millibars.
- The visibility measurement in miles has been converted to kilometers.

Not all observations had the same time format, particularly the first timestamps for each day (at 00:20) had the date appended to them, and as a result, it was removed. Additionally, the time in 12-hour format was converted to 24-hour format in order to match the 'Departures' table (which presents time data in 24-hour format).

A primary key called "ID" was then created, containing the day and time of the weather measurement. This key was subsequently used to associate the rows in the weather dataset with the departures dataset. This was made possible due to the earlier conversion of the time to 24-hour format.

A column named DateTime was also created, containing both the day and time in datetime format, which was useful for subsequent operations. The cleaned dataset was saved as "weather_final_official.csv".

5.2 Dataset departures

Several operations have been performed on the Departures dataset, which are described in detail below.

Firstly, we considered the "Status" attribute, which contains the flight status ("Departed," "Cancelled," etc.) and the actual departure time (in the case of a departed flight); this attribute was split into two columns: "Status_String" (containing the status) and "Status_Hour" (containing the actual departure time). In this operation, only observations with a non-null value for "Status_Hour" were considered, resulting in only the departed flights being included. Therefore, the resulting table no longer contains cancelled, scheduled, or unknown flights since it would not be possible to calculate delays for such flights.

Next, it was observed that some flights share the same date, time, and destination but have different airlines. This indicates the phenomenon of code-sharing, where "marketing airlines" sell tickets to customers on behalf of another single friendly airline, known as the "operating airline".

This data redundancy was addressed by deduplicating based on date, time, status, and destination, keeping only the first airline, which is the one that actually operated the flight (operating airline). Consequently, flights associated with the other airlines (marketing airlines) were removed.

The column containing the actual departure time, "Status_Hour," and the column with the scheduled departure time, "Time," were converted to datetime format. This conversion was done to calculate the difference between the two times in order to obtain the delay between the actual and estimated departure times. During this process, it was observed that some delays appeared to be negative.

This occurred due to two different reasons, which were handled differently:

1. If the delay was negative and between 0 and -1000 minutes (approximately -16.5 hours), it generally indicates flights that departed slightly early. Therefore, in such cases, a delay of 0 was assigned.

2. If the delay was negative and less than -1000 minutes (approximately -16.5 hours), it evidently indicates cases where the flight was scheduled to depart before midnight but actually departed after midnight. The discrepancy between the two days resulted in a large (and negative) delay. To address this issue in such cases, since 24 hours equals 1440 minutes, the delay was assigned as follows:

$$\text{Delay} = 1440 \text{ minutes} + (\text{negative delay calculated})$$

Firstly, a primary key was associated with each row of flights. This key was specifically created by combining the alphanumeric information related to that specific flight into a single string. This key serves to identify the row and was constructed using the flight code, along with the day and month. Additionally, the time was included because we observed that some flights had the same code even on the same day.

Next, the "Weather_hour" column was created to contain the time of the closest weather forecast (from the weather dataset) to the scheduled departure time of each flight. The following operations were performed to achieve this:

1. Create a column called "DateTime" in the Departures dataset, containing both the date and the scheduled departure time in datetime format.

2. Define a function called "find_nearest_time" that finds the closest time to "Time" in the Weather dataset for the same day. This function should take into account that both columns are in datetime format.

3. Apply the "find_nearest_time" function to the "DateTime" column in the Departures dataset. Once the nearest time is found, print it in a new column called "Weather_hour" in the Departures dataset.

4. Finally, create a column called "ID_Weather" in the Departures dataset as a foreign key to link with the Weather dataset. This column can be created by combining the attributes "Weather_hour" and "Date". This key will be used to establish a join between the two tables during querying.

By following these steps, you will have added the necessary columns and established the useful connections between the Departures and Weather datasets to facilitate further analysis and querying.

The attribute "Arrival Airport" has been split into two columns: "IATA" for the IATA code of the airport and "Airport" for the destination city. This division was done to facilitate the merge process between the Departures dataset and the Airport dataset since the column containing the IATA code serves as a foreign key to the Airport dataset in the Departures dataset.

5.3 Dataset airports

Several operations were performed on the Airports dataset. Firstly, we deduplicated the rows based on the fields "name," "IATA," and "ICAO" to ensure that each airport had only one row. The redundancy was due to a data inconsistency in the "localTime" column, which had multiple variations for the same airport and represented time in milliseconds.

Next, we decided to remove the attributes "fs," "faa," "city," "stateCode," "weatherZone," "street1," "postalCode," "street2," "active," and "localTime." Some of these attributes were deemed redundant as their values were present in other attributes, while others contained numerous null values.

Finally we used the merge function from the Pandas package to combine the remaining columns in the Airports dataset with the Departures dataset, using the unique IATA key present in both datasets. Following this operation, to avoid redundancy for the attributes "IATA" and "IATA_Code" in the merged tables, one of them was removed.

5.4 Dataset airline quality

No data cleaning operations were required on the dataset.

The scores and corresponding reviews for the airlines in the Departures dataframe were associated based on the airline names.

A dictionary called "airline_dict" was created using the dict(zip()) function. This dictionary consisted of keys containing the values from the "Airlines" column, and they were associated with the values of the "Score" (scores) and "Reviews" attributes. Therefore, the dictionary linked the airline name with a tuple containing the corresponding score and review.

A new column called "Airline_Score" was added to the Departures dataframe. This column was populated using the apply() function along with a lambda function. The lambda function searched for the airline name in the "airline_dict" dictionary to find a match. It returned the score associated with the corresponding airline or None if no match was found.

Finally, another new column called "Airline_Reviews" was added to the Departures dataframe. This column was populated similarly to the previous column but returned the review associated with the corresponding airline in the "airline_dict" dictionary.

It was observed that the generated dataset had several null values in the "Airline_Score" and "Airline_Reviews" columns. It was noticed that many of these corresponded to companies that did not have a review on airlinequality.com because they were either cargo flights or not reviewed. However, some of the null values were due to differences in the airline names between the two datasets, causing the previous process to fail in matching the corresponding rows.

Two lists were created containing the names of these companies as they appeared in the two datasets, and it was decided to consider the names of these companies in the Departures dataset as correct. A new dictionary ("replace_dict") was generated with the incorrect names from the Airline Quality dataset as keys and the correct names from the Departures dataset as values. This dictionary was used to replace the names of the companies in question in the Airline Quality dataset with those listed in the Departures dataset. Finally, with this modification to the Airline Quality dataset, the previously described code (regarding "airline_dict") was executed again, replacing the previous steps.

The datasets "Airports" and "Airline Quality" have been merged to create a single CSV file called "departures_final_official.csv".

6 Data quality (post-cleaning and post enrichment / integration)

The previously described processes of data cleaning, integration, and enrichment have resulted in the creation of two final datasets with changed data quality.

- **Accuracy**: Following the cleaning and enrichment/integration operations, the data accuracy has not been modified since these operations did not impact this aspect.
- **Completness**:
  - **Departures final** Following the cleaning and integration/enrichment, the completeness has improved. Previously, the Airports dataset contained columns with null values, but these columns were not considered in the integration/enrichment process. However, it should be noted that while the Airline Quality dataset did not originally have null values, after integration/enrichment, the final Departures dataset has null values for the "score" and "reviews" columns due to the presence of airlines without this information (mainly cargo airlines).
  - **Weather final** the completnees has not been modified.
- **Currency** Following the cleaning and enrichment/integration operations, the completeness remained unchanged.
- **Consistency**:
  - **Departures final** Following the cleaning and integration/enrichment, the consistency has improved as the merging between Airline Quality and Departures datasets has resolved the issue of companies having different names in the two datasets.
  - **Weather final** the consistency has not been modified.

7 Data Storage

As a data collection platform, we have chosen a relational database, specifically PostgreSQL, for several reasons explained below.

A relational database is a type of data storage that allows for storing and accessing related information. These databases are based on the relational model, which represents data in the form of tables. In a relational database, each row of the table represents a specific record identified by a unique ID, known as the primary key.

In our case, we need a relationship between weather measurements and flight departures: this is a one-to-many relation because each instance of the "weather final" dataset can correspond to multiple instances of the "departures final" dataset, while each instance of "departures final" can have one and only one associated instance of "weather". Based on this, we have decided to use a relational database. In fact, the relational paradigm is the best option because relational databases excel at managing one-to-many relationships, especially when it comes to efficient indexing. A relational database is the most suitable for our case because we are dealing with structured data organized in tables. The data we have is structured, as it is represented using tables, where each column represents a different field and each row represents a single instance of data.

We choose PostgreSQL as our relational database. It is a comprehensive relational database management system that allows defining relationships between tables. It supports the use of the SQL language for querying the database. Furthermore, PostgreSQL is open-source and can be used with the user interface of pgAdmin, making the user interaction less complex.

The choice of a relational database (PostgreSQL) is more suitable compared to alternatives such as document-based databases (MongoDB), graph databases and column-based for the following reasons :

- **Comparison with document-based (MongoDB)** While the relational model supports the implementation of relationships between tables through primary keys and foreign keys, document-based databases do not involve traditional join operations and require a different approach to manage data links. Furthermore, PostgreSQL (relational model) offers a sophisticated query engine that supports advanced features such as aggregations, joins, and complex filtering operations. On the other hand, MongoDB (document-based), although providing query capabilities, may not be equally efficient when it comes to complex queries and data aggregations.

- **Comparison with graph-based databases** Relational databases allow for defining a well-structured schema to store and query such data, while graph-based databases are more suitable for unstructured or semi-structured data, where relationships between entities are more significant. Relational databases offer a wide range of query functionalities, such as joins, aggregations, and advanced filters. On the other hand, graph-based databases are more suitable for queries based on connectivity and graph structure. They excel at traversing relationships between nodes and are designed to efficiently handle complex graph-based queries.
- **Comparison with column-based databases** The relational model provides a well-defined structure for managing complex relationships between entities. It ensures data integrity through constraints and supports complex querying using the SQL language. On the other hand, the column-based model offers more flexibility in the schema and is better suited for semi-structured or variable data. However, in our case, we are working with structured data, and therefore the relational model is more appropriate.

Using pgAdmin, we have created two tables using the CREATE TABLE statement, one for departures and one for weather. For each of the tables, we have used the COPY command to insert data from their respective CSV files.

TABEL WEATHER

```
CREATE TABLE public.weather_final_official(
ID NUMERIC PRIMARY KEY,
Hours TIME WITHOUT TIME ZONE,
Temperature float,
Weather TEXT,
Wind float,
Humidity TEXT,
Barometer float,
Visibility float,
Day numeric
);
```

```
COPY weather_final_official
FROM 'C:/Users/public/weather_final_official.csv'
DELIMITER ',' CSV HEADER
```

TABEL DEPARTURES

```
CREATE TABLE public.Departures_final_official(
ID TEXT PRIMARY KEY,
Date DATE,
Time TIME WITHOUT TIME ZONE,
Arrival_Airport TEXT,
Airline TEXT,
Flight TEXT,
Status_String TEXT,
Status_Hour TIME WITHOUT TIME ZONE,
Delay NUMERIC,
IATA_Code TEXT,
Weather_Hour TIME WITHOUT TIME ZONE,
ID_Weather bigint,
icao TEXT,
name TEXT,
cityCode TEXT,
countryCode TEXT,
countryName TEXT,
regionName TEXT,
timeZoneRegionName TEXT,
utcOffsetHours FLOAT,
latitude FLOAT,
longitude float,
elevationFeet INTEGER,
classification INTEGER,
Airlines_Score FLOAT,
Airlines_Reviews FLOAT
);

COPY Departures_final_official
FROM 'C:/Users/public/departures_final_official.csv'
DELIMITER ','
CSV HEADER
```

8 Queries with PostgreSQL / Data Exploration

After creating the two databases in PostgreSQL (departures and weather), we queried them using the SQL language to achieve the objectives we set at the beginning (see chapter 1.1).

For each query, we provide the code and the resulting table from PostgreSQL.

1. **What are the weather conditions that have the greatest impact on these delays?** (Considering only the weather conditions with which at least 5 flights have departed within the given time frame)

```
SELECT
    public.weather_final_official.weather,
    ROUND(AVG(public.Departures_final_official.delay), 2) AS delay,
    COUNT(public.Departures_final_official.id) AS tot_flight
FROM
    public.Departures_final_official
JOIN
    public.weather_final_official ON
    public.Departures_final_official.id_weather =
    = public.weather_final_official.id
GROUP BY
    public.weather_final_official.weather
HAVING
    COUNT(public.Departures_final_official.id) > 4
ORDER BY
    delay DESC;
```

   result:

	weather	delay	tot_flight
1	Light rain. Partly sunny.	49.36	11
2	Chilly.	44.67	12
3	More clouds than sun.	44.42	12
4	Rain. Fog.	43.00	13
5	Rain. More clouds than S...	42.83	6
6	Broken clouds.	37.61	133
7	Rain. Passing clouds.	35.97	36
8	Passing clouds.	33.14	2532
9	Light rain. Passing clouds.	31.52	52
10	Partly sunny.	31.00	417
11	Sunny.	29.16	995
12	Light fog.	28.53	17
13	Scattered clouds.	28.17	757
14	Partly cloudy.	27.12	65
15	Light rain. Overcast.	27.10	10
16	Clear.	25.97	1249
17	Fog.	19.17	175
18	Cool.	15.74	19

2. **For each wind condition (0-7 km/h,7-13 km/h,13-40 km/h), what is the average delay recorded?**

```
WITH wind_range_data AS (
  SELECT
    CASE
      WHEN weather.Wind >= 0 AND weather.Wind <= 7
        THEN 'a)␣Light␣wind␣[0-7␣km/h]'
      WHEN weather.Wind > 7 AND weather.Wind <= 13
        THEN 'b)␣Average␣wind␣[7-13␣km/h]'
      WHEN weather.Wind > 13 AND weather.Wind <= 40
        THEN 'c)␣Heavy␣wind␣[13-40␣km/h]'
    END AS Wind_Range,
    AVG(departures.Delay) AS Average_Delay,
    COUNT(*) AS Flight_Count
  FROM public.weather_final_official weather
  JOIN public.departures_final_official departures ON weather.ID =
    = departures.ID_Weather
  WHERE weather.Wind BETWEEN 0 AND 40
  GROUP BY Wind_Range
```

```
)
SELECT Wind_Range, ROUND(Average_Delay, 2) AS Average_Delay,
    Flight_Count
FROM wind_range_data
ORDER BY wind_range_data
;
```

result:

	wind_range	average_delay	flight_count
1	a) Light wind [0-7 km/h]	26.61	2940
2	b) Average wind [7-13 km/h]	31.54	2763
3	c) Heavy wind [13-40 km/h]	38.20	813

3. **For each weather condition, what are the top 3 routes (and therefore the destinations) that have recorded the highest average delay? For each of these routes and the corresponding arrival airports, what is the IATA code, ICAO code, and official airport name?** (Considering only the routes that had at least 5 flights in that weather condition within the given time period)

```
SELECT weather, arrival_airport, IATA_Code, icao, name,
        delay, tot_flight
FROM (
  SELECT weather, arrival_airport, IATA_Code, icao, name,
        delay, tot_flight,
       ROW_NUMBER() OVER (PARTITION BY weather ORDER BY delay DESC)
       AS row_num
  FROM (
    SELECT w.weather, d.arrival_airport, d.IATA_Code, d.icao,
            d.name, ROUND(AVG(d.delay), 2) AS delay,
           COUNT(d.id) AS tot_flight
    FROM departures_final_official AS d
    JOIN weather_final_official AS w ON d.id_weather = w.id
    GROUP BY w.weather, d.arrival_airport, d.IATA_Code,
            d.icao, d.name
    HAVING COUNT(d.id) > 4
  ) AS subquery
) AS final_query
WHERE row_num <= 3
ORDER BY weather, delay DESC;
```

result:

	weather	arrival_airport	iata_code	icao	name	delay	tot_flight
1	Broken clouds.	Barcellona	BCN	LEBL	Barcelona-El Prat Airport	60.38	8
2	Broken clouds.	Lisbona	LIS	LPPT	Lisbon Portela Airport	43.33	9
3	Broken clouds.	New York	JFK	KJFK	John F. Kennedy International Airport	41.60	5
4	Clear.	Delhi	DEL	VIDP	Indira Gandhi International Airport	82.57	7
5	Clear.	Reykjavik	KEF	BIKF	Keflavik International Airport	56.63	8
6	Clear.	Dubai	DXB	OMDB	Dubai Airport	55.43	14
7	Fog.	Londra	LGW	EGKK	London Gatwick Airport	30.73	11
8	Fog.	Monaco	MUC	EDDM	Franz Josef Strauss Airport	23.00	5
9	Fog.	Barcellona	BCN	LEBL	Barcelona-El Prat Airport	22.00	5
10	Partly sunny.	Tenerife	TFS	GCTS	Tenerife South Airport	54.00	5
11	Partly sunny.	New York	JFK	KJFK	John F. Kennedy International Airport	50.33	12
12	Partly sunny.	Lisbona	LIS	LPPT	Lisbon Portela Airport	46.46	13
13	Passing clouds.	Ancoraggio	ANC	PANC	Ted Stevens Anchorage International Airport	127.56	9
14	Passing clouds.	Seoul	ICN	RKSI	Incheon International Airport	100.28	18
15	Passing clouds.	Montreal	YUL	CYUL	Montreal-Pierre Elliott Trudeau International Airpo	93.00	5
16	Scattered clouds.	Seoul	ICN	RKSI	Incheon International Airport	82.57	7
17	Scattered clouds.	Cairo	CAI	HECA	Cairo International Airport	68.36	11
18	Scattered clouds.	Newark	EWR	KEWR	Newark Liberty International Airport	55.67	9
19	Sunny.	Baku	GYD	UBBB	Heydar Aliyev International Airport	74.33	9
20	Sunny.	Seoul	ICN	RKSI	Incheon International Airport	61.88	8
21	Sunny.	Cairo	CAI	HECA	Cairo International Airport	52.77	13

4. **For each wind condition, what are the top 4 airlines that have recorded the highest average delay? For each of these airlines, what is the score and how many reviews do they have?** (Considering only the airlines that operated at least 5 flights in total within the given time period, and at least 2 flights with that wind condition.)

```
WITH flight_counts AS (
  SELECT d.Airline, COUNT(*) AS Flight_Count
  FROM public.departures_final_official d
  GROUP BY d.Airline
  HAVING COUNT(*) > 4
), ranked_data AS (
  SELECT
    w.Wind_Range,
    d.Airline,
    ROUND(AVG(d.Delay), 2) AS Average_Delay,
    d.airlines_score,
    d.airlines_reviews,
    fc.Flight_Count,
    COALESCE(fr.Flights_in_Wind_Range, 0)
            AS Flights_in_Wind_Range,
    ROW_NUMBER() OVER (PARTITION BY w.Wind_Range
                  ORDER BY AVG(d.Delay) DESC) AS row_num
  FROM (
    SELECT
      CASE
        WHEN weather.Wind >= 0 AND weather.Wind <= 7
            THEN 'a)␣Light␣wind␣[0-7␣km/h]'
        WHEN weather.Wind > 7 AND weather.Wind <= 13
            THEN 'b)␣Average␣wind␣[7-13␣km/h]'
        WHEN weather.Wind > 13 AND weather.Wind <= 40
            THEN 'c)␣Heavy␣wind␣[13-40␣km/h]'
      END AS Wind_Range,
      departures.Airline,
      departures.ID_Weather
    FROM public.weather_final_official weather
    JOIN public.departures_final_official departures ON weather.ID=
         =departures.ID_Weather
    WHERE weather.Wind BETWEEN 0 AND 40
  ) AS w
  JOIN public.departures_final_official d ON w.ID_Weather =
    = d.ID_Weather AND w.Airline = d.Airline
  JOIN flight_counts fc ON d.Airline = fc.Airline
  LEFT JOIN (
    SELECT Airline, Wind_Range, COUNT(*) AS Flights_in_Wind_Range
    FROM (
      SELECT
        CASE
          WHEN weather.Wind >= 0 AND weather.Wind <= 7
```

```
            THEN 'a)␣Light␣wind␣[0-7␣km/h]'
          WHEN weather.Wind > 7 AND weather.Wind <= 13
            THEN 'b)␣Average␣wind␣[7-13␣km/h]'
          WHEN weather.Wind > 13 AND weather.Wind <= 40
            THEN 'c)␣Heavy␣wind␣[13-40␣km/h]'
        END AS Wind_Range,
        departures.Airline,
        departures.ID_Weather
    FROM public.weather_final_official weather
    JOIN public.departures_final_official departures ON weather.ID=
        = departures.ID_Weather
      WHERE weather.Wind BETWEEN 0 AND 40
    ) AS w
    GROUP BY Airline, Wind_Range
  ) AS fr ON d.Airline = fr.Airline AND
            w.Wind_Range = fr.Wind_Range
  GROUP BY
    w.Wind_Range,
    d.Airline,
    d.airlines_score,
    d.airlines_reviews,
    fc.Flight_Count,
    fr.Flights_in_Wind_Range
)
SELECT Wind_Range, Airline, Average_Delay, airlines_score,
       airlines_reviews, Flight_Count, Flights_in_Wind_Range
FROM ranked_data
WHERE row_num <= 4
  AND Flights_in_Wind_Range > 1
ORDER BY Wind_Range, Average_Delay DESC;
```

result:

	wind_range	airline	average delay	airlines score	airlines reviews	flight_count	flights in wind range
1	a) Light wind [0-7 km/h]	Asiana Airlines	352.50	8	497	6	2
2	a) Light wind [0-7 km/h]	Air Canada	100.83	4	2143	16	6
3	a) Light wind [0-7 km/h]	Air Cairo	88.71	5	14	21	7
4	a) Light wind [0-7 km/h]	NCA Nippon Cargo Airlines	87.71	[null]	[null]	17	7
5	b) Average wind [7-13 km/h]	Air Canada	119.63	4	2143	16	8
6	b) Average wind [7-13 km/h]	Korean Air	110.95	7	552	32	19
7	b) Average wind [7-13 km/h]	NCA Nippon Cargo Airlines	92.20	[null]	[null]	17	5
8	b) Average wind [7-13 km/h]	Fly One	80.75	1	8	5	4
9	c) Heavy wind [13-40 km/h]	Delta Air Lines	104.17	4	2763	39	6
10	c) Heavy wind [13-40 km/h]	Tunisair	89.75	5	80	23	4
11	c) Heavy wind [13-40 km/h]	Air Canada	88.00	4	2143	16	2
12	c) Heavy wind [13-40 km/h]	Cargolux	87.29	[null]	[null]	48	7

9 Conclusions / Final Observations and Future Developments

We were able to achieve the objective of studying the delays in departures from Milan Malpensa Airport in relation to weather conditions.

We successfully created two tables containing information about flight situations and the various weather conditions experienced by the flights. Additionally, we were able to incorporate user feedback data for airlines and information regarding destination airports.

Furthermore, we were able to address all the initial project questions and obtained detailed and satisfactory results.

As a future development, it would be interesting to extend the time interval of the study to cover a longer period. Additionally, utilizing historical data from previous periods could provide a comparison between longer-term and more recent information, allowing for the prediction of future delay trends. Furthermore, conducting delay analyses for departures from other airports, such as Linate, would be of interest for comparison purposes.

Lastly, a comparison with data from a different time period could also be conducted.

Roles

Alberto Porrini: scraping, data quality, data storage

Andrea Muscio: api airports, data storage, data exploration, cleaning enrich/integration

Francesco Paolo Luigi Caruso: data quality, cleaning enrich/integration, data exploration