

Claude Bernard University Lyon 1

Data Processing and Analytics (DISS - DPA)

Analyzing Flight Interconnected Data

**Students:**

LE Thi Hoa - 12310380 TRAN Hai Linh - 12310487

**Supervised by:**

Prof. Angela Bonifati Prof. Andrea Mauri

27, November 2023

**Link Code:** <https://github.com/tranhailinh97/Graph-PageRank/tree/master>

# Introduction

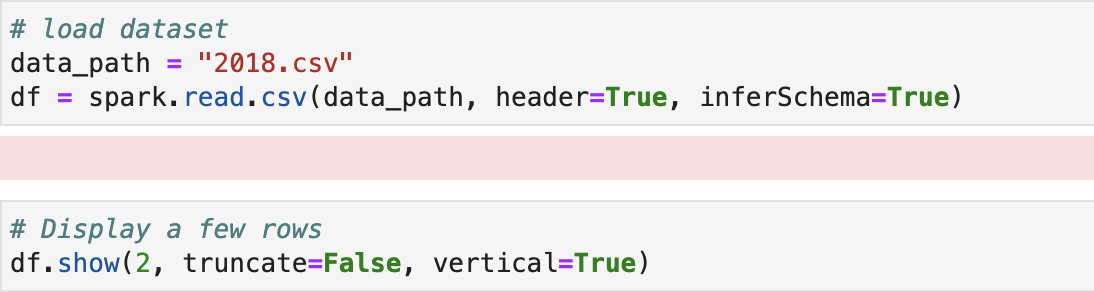
The objective of this project is to use Spark's APIs to analyze the flight interconnected data to find which are the most popular airports.

# Dataset

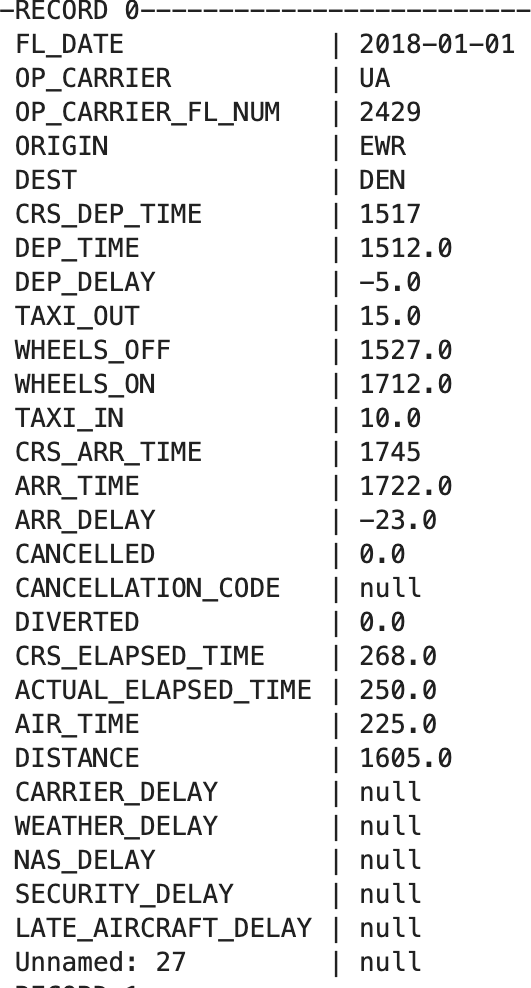
The data has been collected and organized concerning flights throughout the year 2018. The dataset contains detailed information about each flight, including date, airline, origin and destination, estimated and actual times, as well as factors influencing the flights.

## Load Data

Use spark to read dataset



Display several rows of dataset



## Data structure

Data set with **7213446** flights

The dataset is structured with the following columns:

* + - **FL\_DATE**: Date of the flight.
    - **OP\_CARRIER**: Operating Carrier code, indicating the airline that performed the flight.
    - **OP\_CARRIER\_FL\_NUM:** Flight number assigned by the operating carrier.
    - **ORIGIN:** Three-letter code for the origin airport.
    - **DEST:** Three-letter code for the destination airport.
    - **CRS\_DEP\_TIME**: Scheduled departure time.
    - **DEP\_TIME:** Actual departure time.
    - **DEP\_DELAY:** Difference in minutes between actual and scheduled departure times.
    - **TAXI\_OUT:** Time in minutes from departure from the origin gate to wheels off.
    - **WHEELS\_OFF**: Actual departure time when the aircraft wheels leave the ground.
    - **WHEELS\_ON:** Actual arrival time when the aircraft wheels touch the ground.
    - **TAXI\_IN:** Time in minutes from wheels-on to arrival at the destination gate.
    - **CRS\_ARR\_TIME:** Scheduled arrival time.
    - **ARR\_TIME:** Actual arrival time.
    - **ARR\_DELAY:** Difference in minutes between actual and scheduled arrival times.
    - **CANCELLED:** Indicates whether the flight was canceled (1) or not (0).
    - **CANCELLATION\_CODE:** Code specifying the reason for cancellation.
    - **DIVERTED:** Indicates whether the flight was diverted to another airport (1) or not (0).
    - **CRS\_ELAPSED\_TIME:** Scheduled elapsed time of the flight.
    - **ACTUAL\_ELAPSED\_TIME:** Actual elapsed time of the flight.
    - **AIR\_TIME:** Time the aircraft spends airborne.
    - **DISTANCE:** Distance between airports.
    - **CARRIER\_DELAY:** Delay attributed to the carrier.
    - **WEATHER\_DELAY:** Delay attributed to weather conditions.
    - **NAS\_DELAY:** Delay attributed to the National Airspace System.
    - **SECURITY\_DELAY:** Delay attributed to security-related issues.
    - **LATE\_AIRCRAFT\_DELAY**: Delay attributed to issues with the aircraft.
    - **Unnamed: 27**: An unnamed column (seems to have no specific meaning).

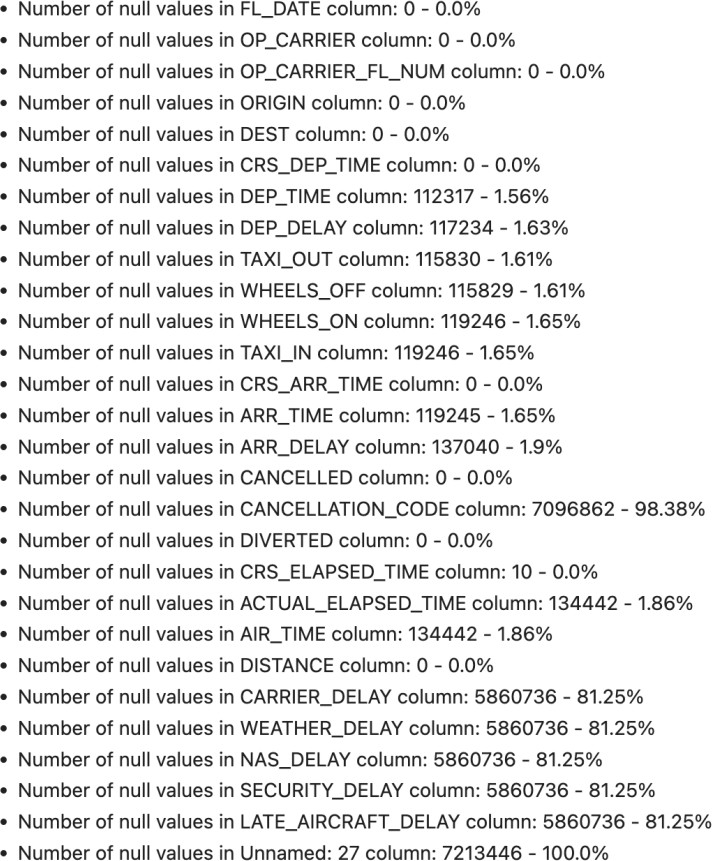
# Data Preprocessing

## Check for missing values

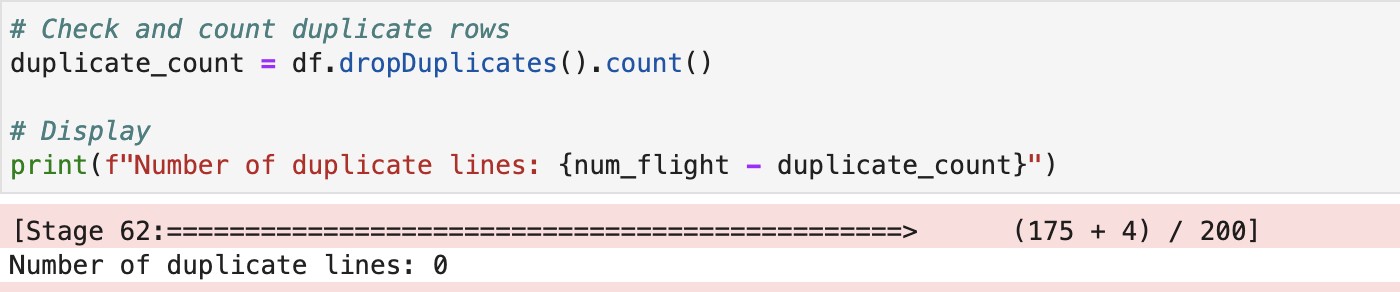
First, we check to see if there are any missing values in the data set. We count the missing values in each column and then calculate the missing percentage



After checking for missing values, we can see that:



## Check for duplicate values

We count the duplicate values

We can see there is no duplicate value

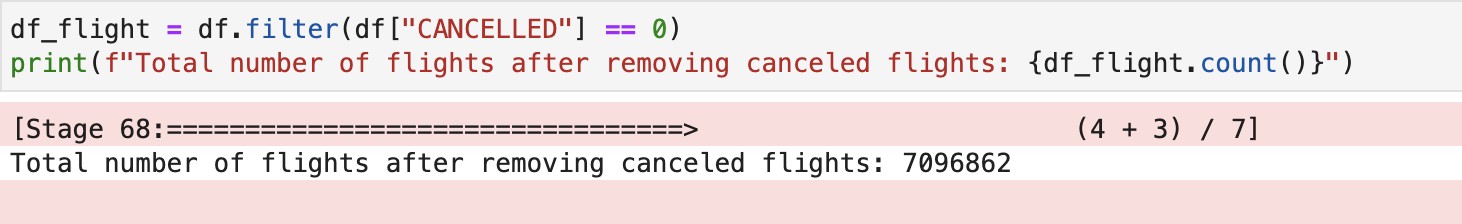
## Check for canceled flights

We count the canceled flights then calculate the missing percentage

There are 116584 canceled flights, accounts for 1.62% of the total data

## Data Cleaning

We removed canceled flights



After removing canceled flights, there are 7096862 flights

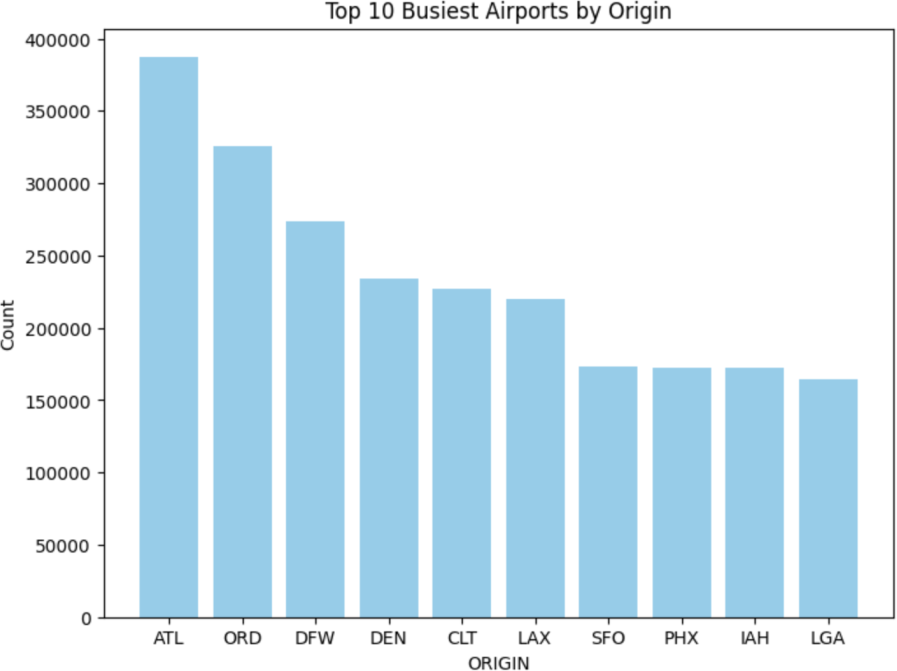
## Data Exploring

### Top 10 airports as the most popular origins

This analysis provides insights into the busiest departure airports, helping us understand the distribution of flights and identify key hubs in the dataset.

There are 385 airports that is the departure airport Visualize the top 10 most popular origins

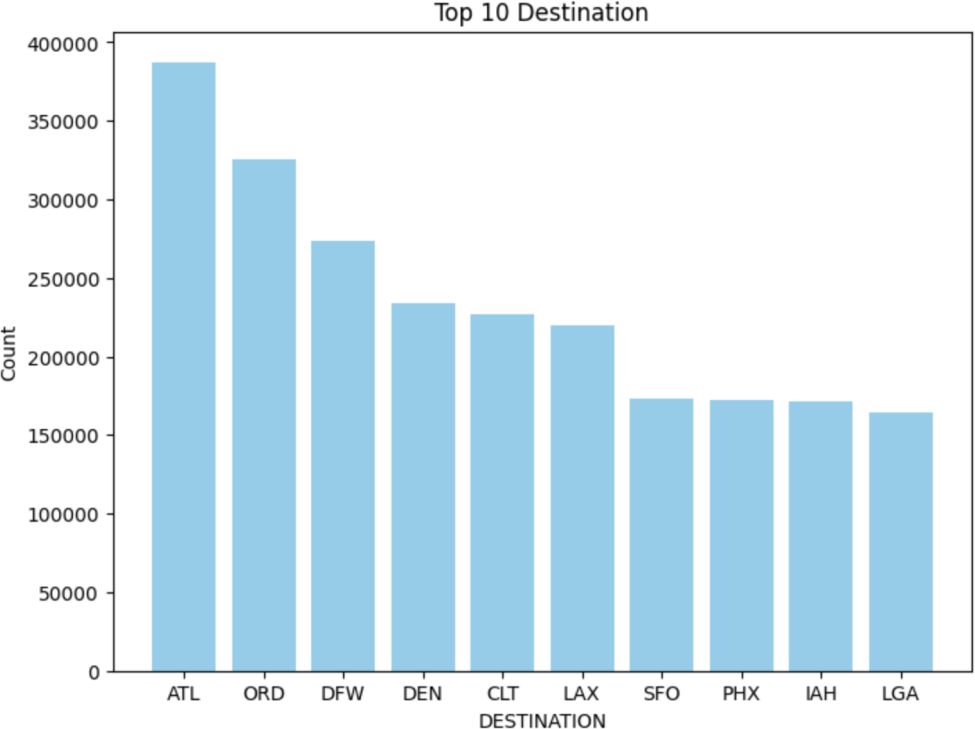




*Figure 1: Top 10 most popular origins*

### Top 10 airports as the most popular destinations

Similar to the top 10 most popular origins, we take the top 10 most popular destinations

There are 385 airports that is the destination airport

*Figure 1: Top 10 the most popular destinations*

# Build A Graph

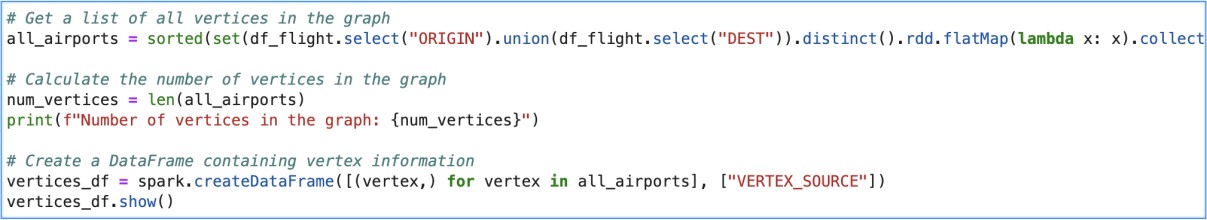
In the process of constructing a graph based on flight data, we define the vertices, edges, and computation of edge weights. This graph serves as a representation of the connections between airports and the frequency of flights between them.

## Vertices

The vertices in this graph are the airports. Each unique airport code from the dataset is considered a vertex. For instance, if there are N unique airports, the graph will have N vertices, each representing a specific airport.

## Solution:

* Extracts unique airport codes from both the "ORIGIN" and "DEST" columns in the DataFrame (df\_flight).
* The airports are collected into a set to remove duplicates
* Then sorted to create a list of all unique vertices (airports) in the graph.



There are 358 vertices in the graph

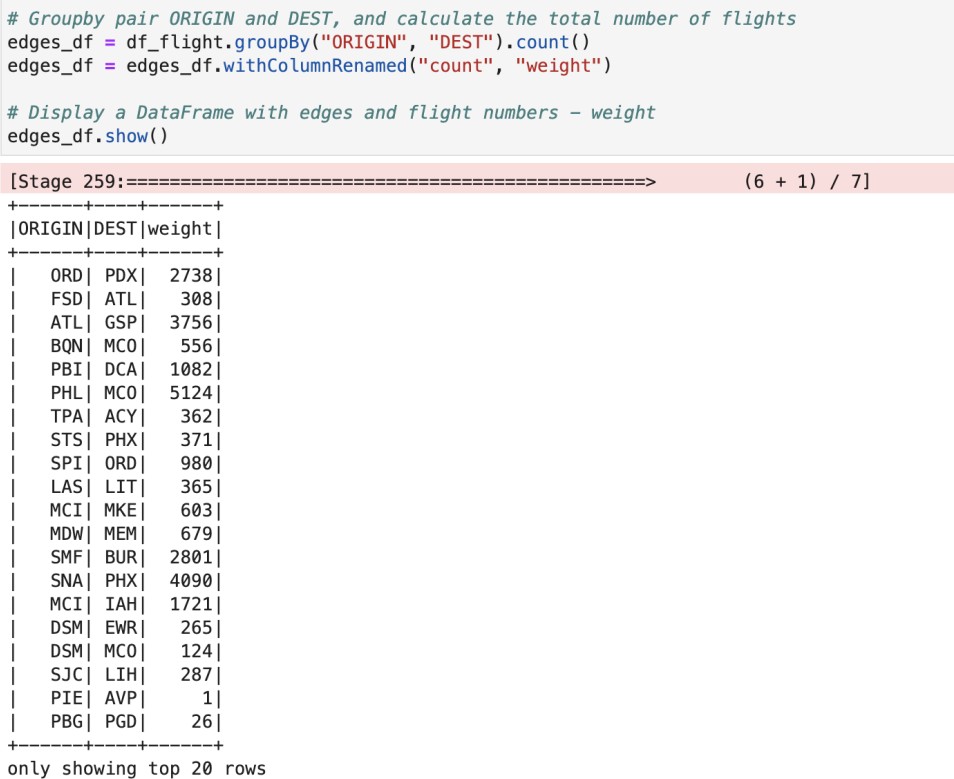
## Edges and weight

The edges in the graph represent the routes between airports. If there is a direct flight from airport A to airport B, then there is an edge connecting vertex A to vertex B in the graph. The presence of an edge indicates a direct connection or route between two airports.

The weight of an edge, representing the connection between two airports, is computed as the total number of flights or the frequency of flights between those two airports. For example, if there were 100 flights from airport A to airport B in a given time period, the weight of the edge between A and B would be 100.

## Solution:

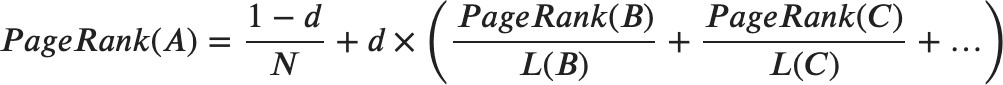
* Group by pair ORIGIN and DEST, and calculate the total number of flights
* Renames the count column to "weight"



# Implement The PageRank Algorithm

## PageRank Formula

The PageRank algorithm calculates the importance of vertices (nodes) in a graph based on the structure of links between them. The formula for PageRank of a vertex A is expressed as follows:



In there:

* PageRank(A) is the PageRank of page A.
* N is the total number of nodes in the graph.
* d is the damping factor.
* PageRank(B),PageRank(C),… are the PageRanks of pages that link to page A.
* L(B),L(C),… are the number of outbound links on pages B, C, etc.

## Damping Factor Explanation

The damping factor is a parameter used in the PageRank algorithm to model the probability that a user will continue clicking on links rather than jumping to a new page. It introduces a level of randomness to the model, simulating the behavior of a web surfer who, with a certain probability, might decide to navigate to a random page instead of following links.

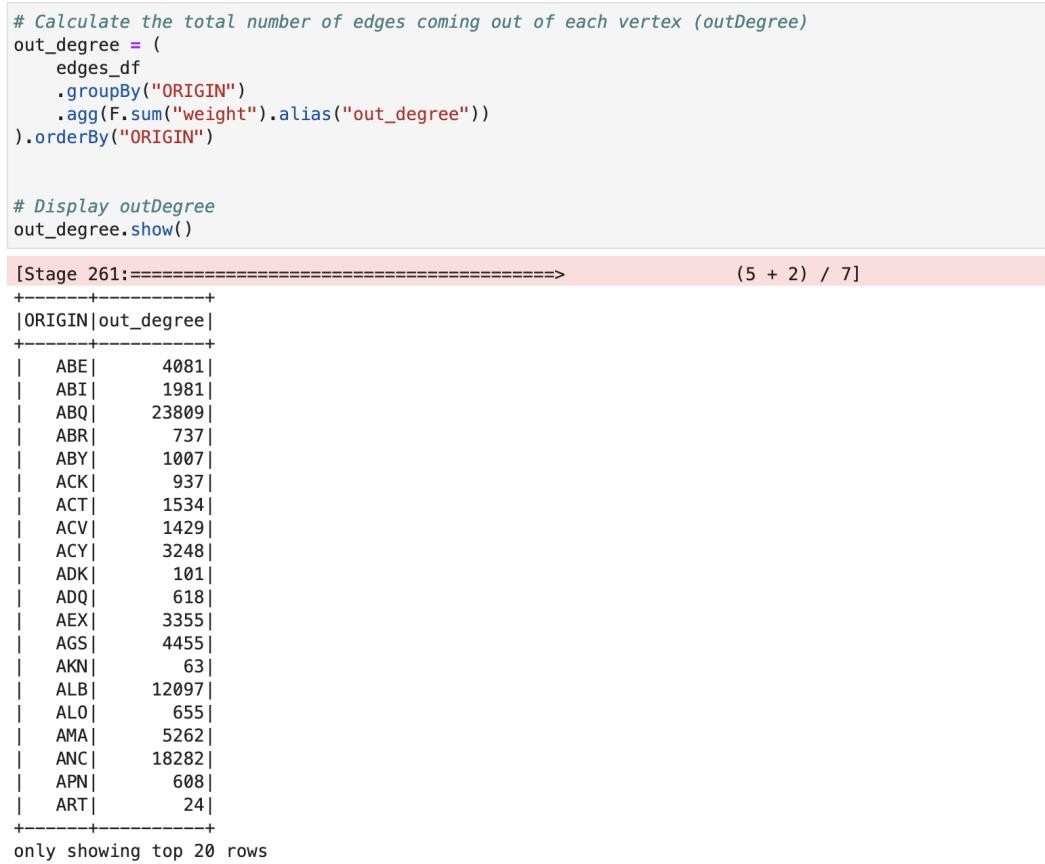
In the context of the PageRank algorithm, the damping factor is typically denoted by the symbol d. The standard value for d is often set to 0.85, but it can vary depending on the specific application. The remaining probability, 1−d, is distributed evenly among all the nodes in the graph, reflecting the surfer's chance of jumping to any page at random.

## Steps in PageRank Algorithm

* **It starts by calculating the total number of edges coming out of each vertex (outDegree)**
  + The outDegree is calculated for each airport, representing the total number of flights leaving each airport.
  + This helps quantify the level of "power" of each vertex in transmitting its rank score to other vertices.

## Solution:

* Calculates the outDegree by grouping the edges\_df DataFrame by "ORIGIN”
* Aggregating the sum of weights (flights) for each origin airport.



## Constructing an adjacency matrix

Adjacency matrix - representing the connections (flights) between each pair of airport

## Solution:

* It does a cross join on the vertices\_df to create all possible combinations of source and destination airports.
* Then joins with the edges\_df to get the weights (flight counts) for each airport pair
* The result is a DataFrame (adjacency\_matrix) with airports as rows, airports as columns, and flight counts as values.

## Normalizing it to a probability matrix Solution:

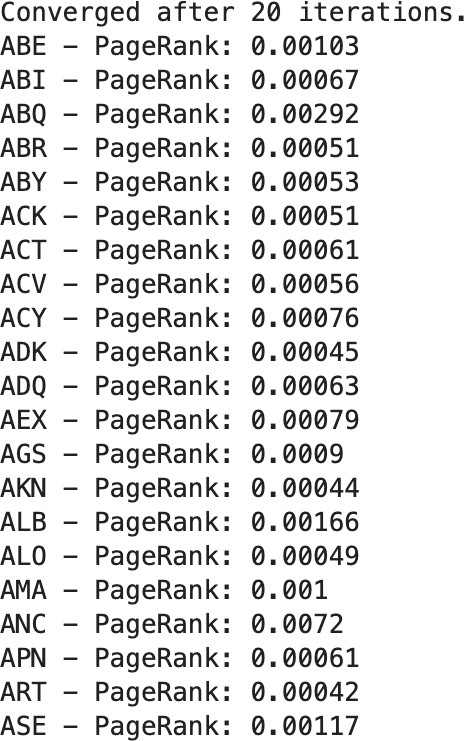
* Convert adjacency matrix from DataFrame to NumPy array
* The adjacency matrix is normalized to create a probability matrix.

- Each element in the matrix is divided by the corresponding outDegree to convert the counts into probabilities.



## Then iteratively calculating PageRank values until convergence Solution:

* The initial PageRank values for all vertices are set. All vertices are given equal initial importance.
* Iteratively calculates the PageRank values for each airport.
* It uses the PageRank formula, considering the normalized adjacency matrix and damping factor.
* The loop continues until convergence (change in PageRank values is below a tolerance level), or until the specified number of iterations is reached.

With a convergence threshold of 1e-6, the algorithm converges after 20 iterations. We can see the PageRank values of some airports:

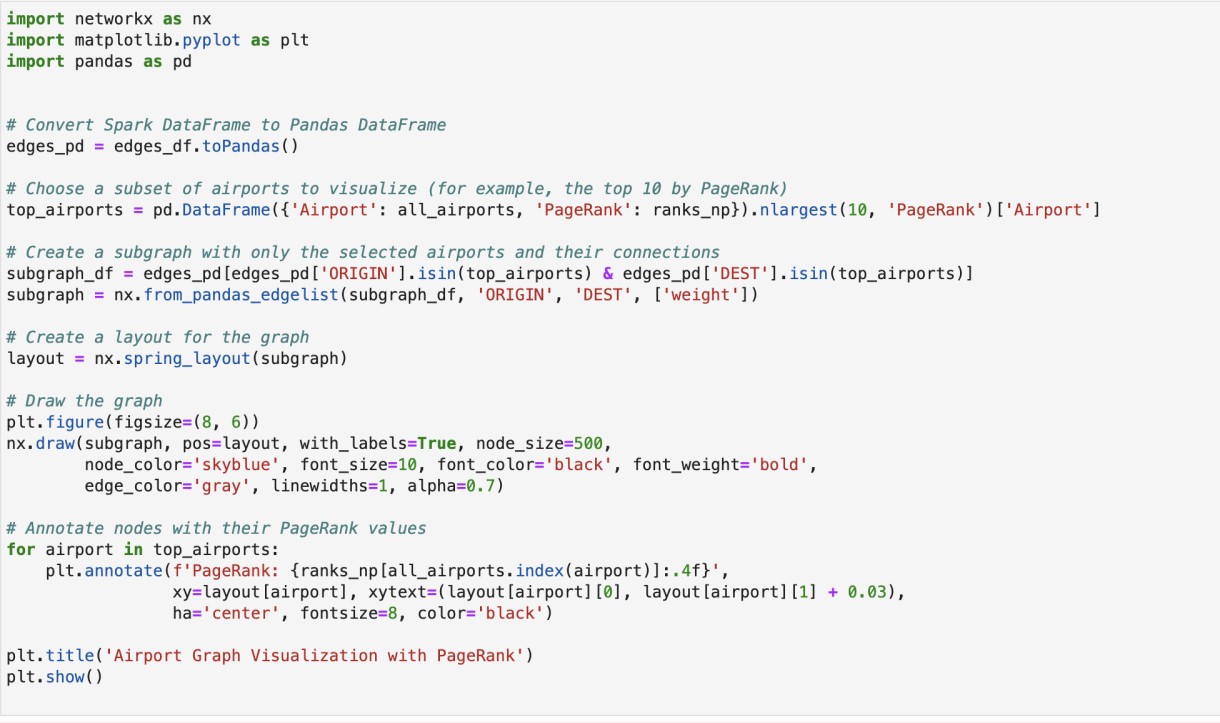
# Graph Visualization

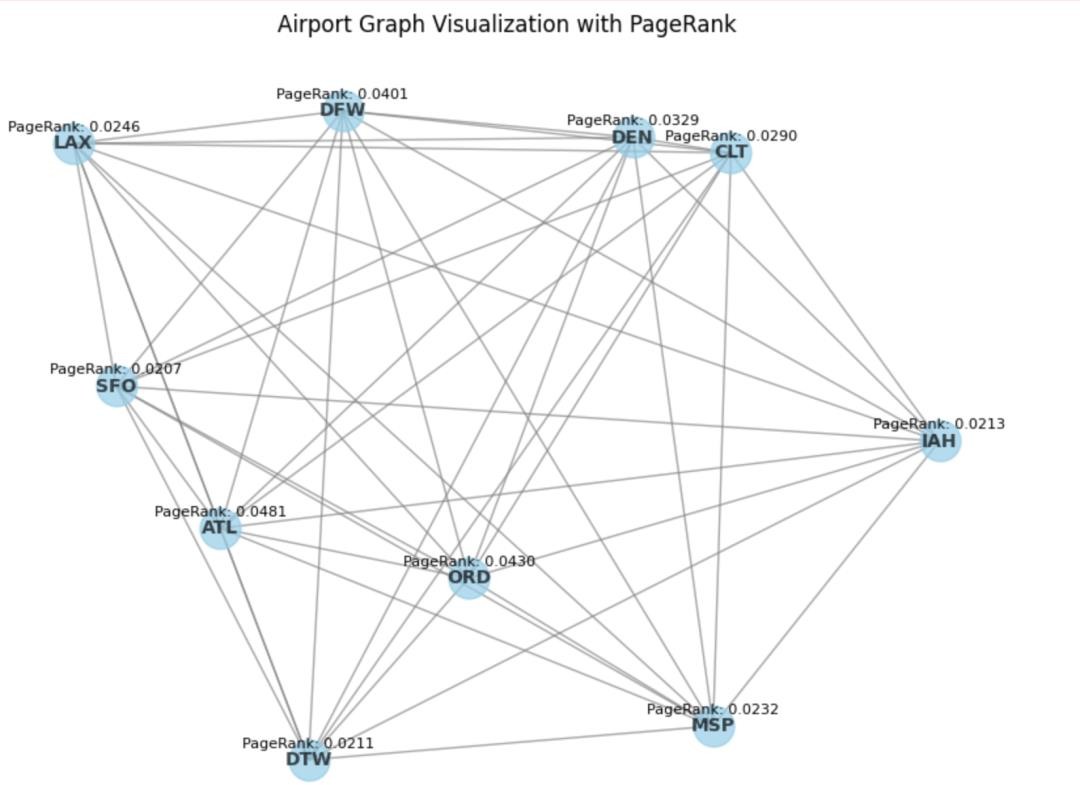
To gain insights into the airport network based on the PageRank algorithm, we visualize a subgraph containing the top 10 airports ranked by PageRank.

Use the “networkx” library to create graphs and matplotlib to draw graphs.

## Solution:

* Select a subset of airports based on their PageRank values, focusing on the top 10 airports.
* Generate a subgraph containing only the selected airports and their connections.
* Defines the layout, draws the graph with specified visual attributes
* Annotates nodes with PageRank values, and finally displays the graph with a title.



=> Result:

*Figure 3: the top 10 airports ranked visualization by PageRank.*