DNSC 6305-ISTM 4212 Section 81 - Fall 2023

Final Project - Group 3 - Airline Data

[redacted]

Ask 1 – Selection and Analysis of the Flight Delay and Cancellation Dataset

Dataset Identification

For our project, we analyzed the "Flight Delay and Cancellation Dataset (2019-2023)" This dataset provides detailed information on flight delays and cancellations from January 2019 through August 2023. It includes data on flight routes, times, delay durations, reasons for delays and cancellations, and segmented details related to time allocations, estimates, and flight routes. Our analysis is based on a 3,000,000 record simple random sample from all 29,380,334 flights recorded.

- **Project Dataset Link** (Kaggle): https://www.kaggle.com/datasets/patrickzel/flight-delay-and-cancellation-dataset-2019-2023
- **Source Data Link** (DOT, On-Time : Reporting Carrier On-Time Performance (1987-present): https://www.transtats.bts.gov/DL_SelectFields.aspx?gnoyr_VQ=FGJ&QO_fu146_anzr=b0-gvzr

Dataset Source

This data is published directly by the U.S. Department of Transportation's Bureau of Transportation Statistics and is publicly accessible and can be downloaded using their “On-Time Reporting Carrier On-Time Performance” tool. Aggregated and sampled representative versions of the data is available on Kaggle. Our dataset was retrieved from the source, merged, sampled, and published to Kaggle for accessibility and consistency. This dataset represents a detailed representative collection of recent airline and airport performance details.

- **Size**: 147 MB
- **Collection Date**: 11/21/2023
- **Timeframe**: January 2019 - August 2023
- **Number of Records**: 3,000,000 row sample of the 29,380,334 flights
    - Sample generated in Python pandas 'data.sample(3000000)'
    - Joined airline friendly names on industry identifier codes dictionaries
    - 32 variables selected from source total of 109

- **Details**: Each record in the dataset represents an individual flight, with detailed information about flight schedules, delays, cancellations, granular time attributions to the minute-level at various stages of the flight sequence and segments

Why This Dataset?

Relevance to Current Issues

- **Impact on Passenger Experience**: Flight delays and cancellations significantly impact passenger experience and satisfaction. Analyzing this dataset allows us to understand the extent and causes of these disruptions, which is vital for improving customer experience.
- **Operational Efficiency**: Delays and cancellations are not just passenger inconveniences; they represent operational inefficiencies for airlines. By studying this data, we can identify patterns and root causes, contributing to more efficient airline operations.
- **Economic Implications**: The aviation sector is a crucial component of the global economy. Delays and cancellations have far-reaching economic implications, from direct costs to airlines to indirect costs associated with passenger delays. This dataset provides a window into understanding these economic impacts.

Potential for Predictive Analysis

- **Forecasting and Planning**: The dataset's detailed historical records are ideal for developing models to predict future delays and cancellations. Such predictive insights are invaluable for airlines and airports for better planning and resource allocation.
- **Machine Learning Applications**: The richness of the dataset opens avenues for advanced machine learning applications. These can range from simple regression models to complex neural networks, aiming to predict delays and cancellations with high accuracy.
- **Benefit to Stakeholders**: Predictive models based on this dataset can benefit various stakeholders, including airlines, airports, regulatory bodies, and passengers, by enabling proactive measures against potential disruptions.

Data Richness and Quality

- **Comprehensive Coverage**: Wide range of variables, from flight times and routes to specific reasons for delays and cancellations.
- **High-Quality Data Collection**: Data sourced from the U.S. Department of Transportation, is collected and maintained with high standards, ensuring reliability and accuracy.
- **Suitability for Multifaceted Analysis**: The richness and quality of the data make it suitable for various analyses, from basic descriptive statistics to complex multivariate analyses.

Suitability for Dimensional Modeling and Analytical Analysis

Comprehensive Data Structure

- **Diverse Data Types**: Mix of categorical (e.g., airline names, airport codes) and numerical data (e.g., delay durations, time of day), which is ideal for creating a robust dimensional model. This diversity allows for a multifaceted analysis, ranging from simple aggregations to complex queries.

- **Time-Series Data**: Given that the dataset spans several years, it provides a rich time-series component. This aspect is crucial for analyzing trends over time and conducting time-based comparisons, which are essential in understanding the dynamics of flight operations.

Richness in Dimensionality

- **Multiple Dimensions for Analysis**: The dataset offers various dimensions for analysis, such as time (date, month, year), geography (origin and destination airports), and operational factors (airlines, flight numbers). This richness allows for creating detailed dimension tables in a data warehouse.

- **Potential for Hierarchical Structuring**: Some of the dimensions, like time and geography, lend themselves to hierarchical structuring (e.g., days nested within months, airports within regions). This structure is beneficial for more sophisticated analyses, such as drill-down and roll-up operations in OLAP (Online Analytical Processing) systems.

Facilitating Analytical Queries

- **Enabling Complex Queries**: The dataset's structure supports complex queries, essential for answering specific analytical questions. For instance, one could query the average delay times for different airlines during holiday seasons or compare cancellation rates between major airports.
- **Custom Attribute Creation**: The dataset allows for the creation of custom attributes or derived metrics, such as 'percentage of flights delayed' or 'average delay per flight', which can be used in both reporting and deeper analytical studies.

Business/Analytical Questions

1. What is the departure delay rate by month across all airlines and airports?

2. In general, which airlines have the highest percentage of departure delays?

3. In general, which airlines have the highest percentage of arrival delays?

4. In general, which airports handle high volumes of traffic with relatively fewer departure delays?

Concerns with the Data and Our Potential Adjustments

Large Dataset Size and Storage Constraints

- **Concern**: The dataset is extensive, encompassing several years of flight delay and cancellation data, which could pose challenges in terms of processing and storage capacity.
- **Our Plan**: Given the constraints, we might need to strategically reduce the dataset's size to manage it more effectively. Our approach will involve using a 3 million row sample of the

original 30 million dataset. This will allow us to conduct a detailed analysis while ensuring the dataset remains manageable within our storage and processing capabilities. We will use relational databases and Unix command line tools for filtering and selecting the relevant subset of data, ensuring that our analysis, while more focused, remains robust and insightful.

Data Time Frame Limitation

- **Concern**: The dataset is currently incomplete for the year 2023. As of data retrieval date, 11/21/2023, data ran through August 2023.
- **Our Plan**: Even with the limited data for 2023, we still have the ability to compare 8 months to prior years' worth of data. Additionally, patterns seem to be similar across the years, in which 2023's incomplete set only adds to the analysis of patterns. New data was uploaded ~1 week prior to submission – due to the late addition of data, we kept the original download (January 2021-August 2023) in order to stick to current processes.

Granularity of Delay and Cancellation Reasons

- **Concern**: The dataset might not provide sufficient detail in the categorization of delay and cancellation reasons for nuanced analysis.
- **Our Plan**: Utilizing Spark, we will analyze the level of detail available for these reasons. If the granularity is insufficient, we will focus our analysis on broader trends and acknowledge this as a limitation in our findings.

Set up SQL Database.

In [1]:
```
%load_ext sql
```

In [2]:
```
!dropdb -U student Final_Airline_Dataset3
```

In [3]:
```
!createdb -U student Final_Airline_Dataset3
```

In [4]:
```
%sql postgresql://student@/Final_Airline_Dataset3
```

Loading Airline data sample of 3M -- *Only download data one time**

In [5]:
```
!pwd
```

```
/home/ubuntu/notebooks/Final_Project
```

In [6]:
```
%cd /home/ubuntu/notebooks/
```

```
/home/ubuntu/notebooks
```

In [7]:
```
!pip install opendatasets
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: opendatasets in /home/ubuntu/.local/lib/python3.8/site
-packages (0.1.22)
Requirement already satisfied: kaggle in /home/ubuntu/.local/lib/python3.8/site-packa
ges (from opendatasets) (1.5.16)
Requirement already satisfied: click in /home/ubuntu/.local/lib/python3.8/site-packag
es (from opendatasets) (8.1.3)
Requirement already satisfied: tqdm in /home/ubuntu/.local/lib/python3.8/site-package
s (from opendatasets) (4.66.1)
Requirement already satisfied: python-dateutil in /home/ubuntu/.local/lib/python3.8/s
ite-packages (from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: urllib3 in /home/ubuntu/.local/lib/python3.8/site-pack
ages (from kaggle->opendatasets) (1.26.12)
Requirement already satisfied: bleach in /home/ubuntu/.local/lib/python3.8/site-packa
ges (from kaggle->opendatasets) (5.0.1)
Requirement already satisfied: certifi in /home/ubuntu/.local/lib/python3.8/site-pack
ages (from kaggle->opendatasets) (2022.9.24)
Requirement already satisfied: python-slugify in /home/ubuntu/.local/lib/python3.8/si
te-packages (from kaggle->opendatasets) (6.1.2)
Requirement already satisfied: six>=1.10 in /home/ubuntu/.local/lib/python3.8/site-pa
ckages (from kaggle->opendatasets) (1.16.0)
Requirement already satisfied: requests in /home/ubuntu/.local/lib/python3.8/site-pac
kages (from kaggle->opendatasets) (2.28.1)
Requirement already satisfied: webencodings in /home/ubuntu/.local/lib/python3.8/site
-packages (from bleach->kaggle->opendatasets) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /home/ubuntu/.local/lib/python
3.8/site-packages (from python-slugify->kaggle->opendatasets) (1.3)
Requirement already satisfied: idna<4,>=2.5 in /home/ubuntu/.local/lib/python3.8/site
-packages (from requests->kaggle->opendatasets) (3.4)
Requirement already satisfied: charset-normalizer<3,>=2 in /home/ubuntu/.local/lib/py
thon3.8/site-packages (from requests->kaggle->opendatasets) (2.1.1)

[notice] A new release of pip available: 22.3 -> 23.3.1
[notice] To update, run: python3 -m pip install --upgrade pip
```

In [8]:
```python
import opendatasets as od
```

In [9]:
```python
dataset='https://www.kaggle.com/datasets/patrickzel/flight-delay-and-cancellation-data
```

Below, use included kaggle.json file to copy and paste the Kaggle username and key.

In [10]:
```python
od.download(dataset, force=True)
```

```
Please provide your Kaggle credentials to download this dataset. Learn more: http://b
it.ly/kaggle-creds
Your Kaggle username:

Your Kaggle Key:

Downloading flight-delay-and-cancellation-dataset-2019-2023.zip to ./flight-delay-and
-cancellation-dataset-2019-2023
```

```
100%|█████████████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████| 140M/140M [00:01<00:00, 125MB/s]
```

In [11]:
```python
import os
```

In [12]:
```
data_dir='/home/ubuntu/notebooks/flight-delay-and-cancellation-dataset-2019-2023'
```

In [13]:
```
os.listdir(data_dir)
```

Out[13]:
```
['flights_sample.csv',
 'flights_sample3.csv',
 'dictionary.html',
 'flights_sample_3m.csv']
```

In [14]:
```
%cd /home/ubuntu/notebooks/flight-delay-and-cancellation-dataset-2019-2023
```

```
/home/ubuntu/notebooks/flight-delay-and-cancellation-dataset-2019-2023
```

In [15]:
```
!mv flights_sample_3m.csv flights_sample.csv
```

Data Wrangling

In [16]:
```
import os
from IPython.display import Image
```

In [17]:
```
!pwd
```

```
/home/ubuntu/notebooks/flight-delay-and-cancellation-dataset-2019-2023
```

In [18]:
```
%cd /home/ubuntu/notebooks/flight-delay-and-cancellation-dataset-2019-2023
```

```
/home/ubuntu/notebooks/flight-delay-and-cancellation-dataset-2019-2023
```

Total number of lines in the csv (3,000,001 including the header)

In [19]:
```
!wc -l flights_sample.csv
```

```
3000001 flights_sample.csv
```

In [20]:
```
!csvcut -n flights_sample.csv
```

```
  1: FL_DATE
  2: AIRLINE
  3: AIRLINE_DOT
  4: AIRLINE_CODE
  5: DOT_CODE
  6: FL_NUMBER
  7: ORIGIN
  8: ORIGIN_CITY
  9: DEST
 10: DEST_CITY
 11: CRS_DEP_TIME
 12: DEP_TIME
 13: DEP_DELAY
 14: TAXI_OUT
 15: WHEELS_OFF
 16: WHEELS_ON
 17: TAXI_IN
 18: CRS_ARR_TIME
 19: ARR_TIME
 20: ARR_DELAY
 21: CANCELLED
 22: CANCELLATION_CODE
 23: DIVERTED
 24: CRS_ELAPSED_TIME
 25: ELAPSED_TIME
 26: AIR_TIME
 27: DISTANCE
 28: DELAY_DUE_CARRIER
 29: DELAY_DUE_WEATHER
 30: DELAY_DUE_NAS
 31: DELAY_DUE_SECURITY
 32: DELAY_DUE_LATE_AIRCRAFT
```

No errors

In [21]:
```
!csvclean flights_sample.csv
```

```
No errors.
```

Our analysis does not depend on certain columns such as taxi_in, delay_due_carrier, etc. For this reason we will cut said columns in order to only look at ones relevant to our analysis.

Wrangling and Cleaning

In [24]:
```
!csvcut -c 1,2,5,6,7,9,11,12,13,18,19,20,21,24,25,26,27 flights_sample.csv > flights_s
```

We do not include origin_city and dest_city, as we already have airport codes indicated by origin and dest fields. This ensures that there is less room for errors/typos and we can stick to the three-letter airport codes. Additionally, some cities have multiple airports, and our analysis is specific to airports/airlines, not necessarily to cities as a whole.

In [25]:
```
!csvcut -n flights_sample2.csv
```

```
  1: FL_DATE
  2: AIRLINE
  3: DOT_CODE
  4: FL_NUMBER
  5: ORIGIN
  6: DEST
  7: CRS_DEP_TIME
  8: DEP_TIME
  9: DEP_DELAY
 10: CRS_ARR_TIME
 11: ARR_TIME
 12: ARR_DELAY
 13: CANCELLED
 14: CRS_ELAPSED_TIME
 15: ELAPSED_TIME
 16: AIR_TIME
 17: DISTANCE
```

In the above portion, we cut out some columns in order to only look at the relevant columns for our analysis.

Now, we will look at a sample of the data to ensure the data looks right.

In [26]:
```
!head -n 10 flights_sample2.csv | csvlook
```

```
/home/ubuntu/.local/lib/python3.8/site-packages/agate/table/from_csv.py:74: RuntimeWa
rning: Error sniffing CSV dialect: Could not determine delimiter
|    FL_DATE | AIRLINE                     | DOT_CODE | FL_NUMBER | ORIGIN | DEST | CRS_DE
P_TIME | DEP_TIME | DEP_DELAY | CRS_ARR_TIME | ARR_TIME | ARR_DELAY | CANCELLED | CRS
_ELAPSED_TIME | ELAPSED_TIME | AIR_TIME | DISTANCE |
| ---------- | ---------------------- | -------- | --------- | ------ | ---- | ------
------ | -------- | --------- | ------------ | -------- | --------- | --------- | ---
------------- | ------------ | -------- | -------- |
| 2019-01-09 | United Air Lines Inc.  |   19,977 |     1,562 | FLL    | EWR  |
1,155 |    1,151 |        -4 |        1,501 |    1,447 |       -14 |         0 |
186 |          176 |      153 |    1,065 |
| 2022-11-19 | Delta Air Lines Inc.   |   19,790 |     1,149 | MSP    | SEA  |
2,120 |    2,114 |        -6 |        2,315 |    2,310 |        -5 |         0 |
235 |          236 |      189 |    1,399 |
| 2022-07-22 | United Air Lines Inc.  |   19,977 |       459 | DEN    | MSP  |
954 |    1,000 |         6 |        1,252 |    1,252 |         0 |         0 |
118 |          112 |       87 |      680 |
| 2023-03-06 | Delta Air Lines Inc.   |   19,790 |     2,295 | MSP    | SFO  |
1,609 |    1,608 |        -1 |        1,829 |    1,853 |        24 |         0 |
260 |          285 |      249 |    1,589 |
| 2020-02-23 | Spirit Air Lines       |   20,416 |       407 | MCO    | DFW  |
1,840 |    1,838 |        -2 |        2,041 |    2,040 |        -1 |         0 |
181 |          182 |      153 |      985 |
| 2019-07-31 | Southwest Airlines Co. |   19,393 |       665 | DAL    | OKC  |
1,010 |    1,237 |       147 |        1,110 |    1,331 |       141 |         0 |
60 |           54 |       36 |      181 |
| 2023-06-11 | American Airlines Inc. |   19,805 |     2,134 | DCA    | BOS  |
1,010 |    1,001 |        -9 |        1,159 |    1,130 |       -29 |         0 |
109 |           89 |       58 |      399 |
| 2019-07-08 | Republic Airline       |   20,452 |     4,464 | HSV    | DCA  |
1,643 |    1,637 |        -6 |        1,945 |    2,008 |        23 |         0 |
122 |          151 |       88 |      613 |
| 2023-02-12 | Spirit Air Lines       |   20,416 |       590 | IAH    | LAX  |
530 |      527 |        -3 |          717 |      706 |       -11 |         0 |
227 |          219 |      200 |    1,379 |
```

Verify 3M rows plus header are retained

In [27]:
```
!wc -l flights_sample2.csv
```

```
3000001 flights_sample2.csv
```

Now, we want to derive the fl_month, fl_year, and fl_day from the fl_date field.

In [28]:
```
!csvcut -c FL_DATE flights_sample2.csv | csvformat -T | awk -F- '{print $2}' > new_col
!paste -d',' flights_sample2.csv new_column.csv > new_file.csv
```

In [29]:
```
!csvcut -c FL_DATE new_file.csv | csvformat -T | awk -F- '{print $1}' > new_column.csv
!paste -d',' new_file.csv new_column.csv > new_file2.csv
```

In [30]:
```
!csvcut -c FL_DATE new_file2.csv | csvformat -T | awk -F- '{print $3}' > new_column.cs
!paste -d',' new_file2.csv new_column.csv > new_file3.csv
```

In [31]:
```
!echo "FL_DATE,AIRLINE,DOT_CODE,FL_NUMBER,ORIGIN,DEST,CRS_DEP_TIME,DEP_TIME,DEP_DELAY,
```

In [32]:
```
!tail -n +2 new_file3.csv > new_file4.csv
```

In [33]:
```
!csvstack new_file4.csv >> header.csv
```

In [34]:
```
!mv header.csv flights_sample3.csv
```

In [35]:
```
!rm flights_sample_out.csv flights_sample2.csv flights_sample.csv new_file4.csv new_fi
```

Time to gather summary statistics

In [36]:
```
!head -n 10000 flights_sample3.csv | csvstat
```

```
/home/ubuntu/.local/lib/python3.8/site-packages/agate/table/from_csv.py:74: RuntimeWa
rning: Error sniffing CSV dialect: Could not determine delimiter
  1. "FL_DATE"

        Type of data:          Date
        Contains null values:  False
        Unique values:         1680
        Smallest value:        2019-01-01
        Largest value:         2023-08-31
        Most common values:    2019-09-27 (21x)
                               2019-05-25 (17x)
                               2019-12-13 (16x)
                               2019-07-08 (15x)
                               2022-04-14 (15x)

  2. "AIRLINE"

        Type of data:          Text
        Contains null values:  False
        Unique values:         18
        Longest value:         34 characters
        Most common values:    Southwest Airlines Co. (1938x)
                               Delta Air Lines Inc. (1393x)
                               American Airlines Inc. (1291x)
                               SkyWest Airlines Inc. (1126x)
                               United Air Lines Inc. (794x)

  3. "DOT_CODE"

        Type of data:          Number
        Contains null values:  False
        Unique values:         18
        Smallest value:        19393
        Largest value:         20452
        Sum:                   199689492
        Mean:                  19970.946
        Median:                19930
        StDev:                 377.408
        Most common values:    19393 (1938x)
                               19790 (1393x)
                               19805 (1291x)
                               20304 (1126x)
                               19977 (794x)

  4. "FL_NUMBER"

        Type of data:          Number
        Contains null values:  False
        Unique values:         4685
        Smallest value:        1
        Largest value:         8799
        Sum:                   25052006
        Mean:                  2505.451
        Median:                2148
        StDev:                 1738.72
        Most common values:    671 (10x)
                               706 (9x)
```

```
                        440 (8x)
                        1950 (8x)
                        883 (8x)

5. "ORIGIN"

      Type of data:          Text
      Contains null values:  False
      Unique values:         314
      Longest value:         3 characters
      Most common values:    ATL (529x)
                             ORD (436x)
                             DFW (427x)
                             DEN (380x)
                             CLT (331x)

6. "DEST"

      Type of data:          Text
      Contains null values:  False
      Unique values:         302
      Longest value:         3 characters
      Most common values:    ATL (556x)
                             DFW (418x)
                             ORD (400x)
                             DEN (388x)
                             CLT (323x)

7. "CRS_DEP_TIME"

      Type of data:          Number
      Contains null values:  False
      Unique values:         1068
      Smallest value:        7
      Largest value:         2359
      Sum:                   13221178
      Mean:                  1322.25
      Median:                1315
      StDev:                 486.649
      Most common values:    600 (240x)
                             700 (169x)
                             800 (89x)
                             900 (72x)
                             730 (68x)

8. "DEP_TIME"

      Type of data:          Number
      Contains null values:  True (excluded from calculations)
      Unique values:         1185
      Smallest value:        1
      Largest value:         2400
      Sum:                   12923027
      Mean:                  1325.711
      Median:                1319
      StDev:                 500.708
      Most common values:    None (251x)
```

```
                          555 (37x)
                          557 (33x)
                          556 (27x)
                          653 (27x)

 9. "DEP_DELAY"

       Type of data:          Number
       Contains null values:  True (excluded from calculations)
       Unique values:         277
       Smallest value:        -31
       Largest value:         1180
       Sum:                   93192
       Mean:                  9.56
       Median:                -2
       StDev:                 46.836
       Most common values:    -5 (838x)
                              -4 (772x)
                              -3 (727x)
                              -2 (663x)
                              -6 (624x)

10. "CRS_ARR_TIME"

       Type of data:          Number
       Contains null values:  False
       Unique values:         1174
       Smallest value:        1
       Largest value:         2359
       Sum:                   14841609
       Mean:                  1484.309
       Median:                1512
       StDev:                 515.09
       Most common values:    1710 (39x)
                              2359 (39x)
                              1915 (37x)
                              1855 (32x)
                              1540 (32x)

11. "ARR_TIME"

       Type of data:          Number
       Contains null values:  True (excluded from calculations)
       Unique values:         1239
       Smallest value:        1
       Largest value:         2400
       Sum:                   14258859
       Mean:                  1463.498
       Median:                1502
       StDev:                 534.652
       Most common values:    None (256x)
                              1852 (19x)
                              1507 (19x)
                              1845 (18x)
                              1728 (18x)

12. "ARR_DELAY"
```

```
        Type of data:          Number
        Contains null values:  True (excluded from calculations)
        Unique values:         304
        Smallest value:        -62
        Largest value:         1185
        Sum:                   35644
        Mean:                  3.664
        Median:                -7
        StDev:                 49.177
        Most common values:    -13 (314x)
                               -11 (309x)
                               -9 (301x)
                               -15 (292x)
                               -12 (289x)

 13. "CANCELLED"

        Type of data:          Number
        Contains null values:  False
        Unique values:         2
        Smallest value:        0
        Largest value:         1
        Sum:                   253
        Mean:                  0.025
        Median:                0
        StDev:                 0.157
        Most common values:    0 (9746x)
                               1 (253x)

 14. "CRS_ELAPSED_TIME"

        Type of data:          Number
        Contains null values:  False
        Unique values:         388
        Smallest value:        33
        Largest value:         665
        Sum:                   1419551
        Mean:                  141.969
        Median:                125
        StDev:                 70.971
        Most common values:    85 (190x)
                               80 (170x)
                               90 (166x)
                               75 (165x)
                               110 (144x)

 15. "ELAPSED_TIME"

        Type of data:          Number
        Contains null values:  True (excluded from calculations)
        Unique values:         391
        Smallest value:        27
        Largest value:         722
        Sum:                   1325674
        Mean:                  136.26
        Median:                119
```

```
        StDev:                  71.042
        Most common values:     None (270x)
                                80 (92x)
                                108 (91x)
                                74 (88x)
                                79 (87x)

16. "AIR_TIME"

        Type of data:           Number
        Contains null values:   True (excluded from calculations)
        Unique values:          373
        Smallest value:         14
        Largest value:          661
        Sum:                    1088982
        Mean:                   111.932
        Median:                 94
        StDev:                  69.072
        Most common values:     None (270x)
                                44 (103x)
                                69 (92x)
                                56 (91x)
                                52 (91x)

17. "DISTANCE"

        Type of data:           Number
        Contains null values:   False
        Unique values:          1294
        Smallest value:         61
        Largest value:          5095
        Sum:                    8057898
        Mean:                   805.87
        Median:                 649
        StDev:                  583.85
        Most common values:     337 (66x)
                                733 (51x)
                                862 (48x)
                                594 (47x)
                                335 (46x)

18. "FL_MONTH"

        Type of data:           Number
        Contains null values:   False
        Unique values:          12
        Smallest value:         1
        Largest value:          12
        Sum:                    62826
        Mean:                   6.283
        Median:                 6
        StDev:                  3.377
        Most common values:     3 (971x)
                                8 (948x)
                                7 (944x)
                                6 (919x)
                                1 (879x)
```

```
19. "FL_YEAR"

        Type of data:           Number
        Contains null values:   False
        Unique values:          5
        Smallest value:         2019
        Largest value:          2023
        Sum:                    20206764
        Mean:                   2020.878
        Median:                 2021
        StDev:                  1.412
        Most common values:     2019 (2498x)
                                2022 (2345x)
                                2021 (1996x)
                                2020 (1628x)
                                2023 (1532x)

20. "FL_DAY"

        Type of data:           Number
        Contains null values:   False
        Unique values:          31
        Smallest value:         1
        Largest value:          31
        Sum:                    157363
        Mean:                   15.738
        Median:                 16
        StDev:                  8.749
        Most common values:     14 (368x)
                                27 (354x)
                                21 (349x)
                                18 (348x)
                                15 (346x)

Row count: 9999
```

The data looks good so it is now time to analyze.

The data is very clean. Very few fields have any nulls. For the ones that do, such as dep_time, the cases where this occurs are when the flight was canceled. For that reason, the dep_time should naturally be null.

Create Table and import

In [37]:

```
%%sql
DROP TABLE IF EXISTS flights;

CREATE TABLE flights (
    fl_date DATE NOT NULL,
    airline varchar(60) NOT NULL,
    dot_code numeric NOT NULL,
    fl_number numeric NOT NULL,
    origin varchar(3) NOT NULL,
    dest varchar(3) NOT NULL,
```

```
    crs_dep_time numeric NOT NULL,
    dep_time numeric,
    dep_delay numeric,
    crs_arr_time numeric NOT NULL,
    arr_time numeric,
    arr_delay numeric,
    cancelled numeric(3) NOT NULL,
    crs_elapsed_time numeric,
    elapsed_time numeric,
    air_time numeric,
    distance numeric NOT NULL,
    fl_month numeric NOT NULL,
    fl_year numeric NOT NULL,
    fl_day numeric NOT NULL
)
```

```
 * postgresql://student@/Final_Airline_Dataset3
Done.
Done.
```

Out[37]: []

In [38]:

```
%%sql
COPY flights FROM '/home/ubuntu/notebooks/flight-delay-and-cancellation-dataset-2019-2
CSV
HEADER;
```

```
 * postgresql://student@/Final_Airline_Dataset3
3000000 rows affected.
```

Out[38]: []

Verifying that the flights table has the correct number of records.

In [39]:

```
%%sql
SELECT COUNT(*) FROM flights;
```

```
 * postgresql://student@/Final_Airline_Dataset3
1 rows affected.
```

Out[39]:

count
3000000

To confirm the statement about the nulls at the conclusion of the prior section, we will run one quick query.

In [40]:

```
%%sql
SELECT * FROM flights
where dep_time IS NULL AND cancelled=0.0
```

```
 * postgresql://student@/Final_Airline_Dataset3
0 rows affected.
```

Out[40]:

fl_date	airline	dot_code	fl_number	origin	dest	crs_dep_time	dep_time	dep_delay	crs_arr_time	a

As we can see above, there are 0 instances of dep_time being null, where the flight was not cancelled. If the flight was cancelled, cancelled=1.0. This further confirms the reasons behind any nulls and thus further proves the cleanliness of this data.

In [41]:
```
!wc -l flights_sample3.csv
```

```
3000001 flights_sample3.csv
```

In [42]:
```
%%sql
SELECT * FROM flights
LIMIT 10
```

```
 * postgresql://student@/Final_Airline_Dataset3
10 rows affected.
```

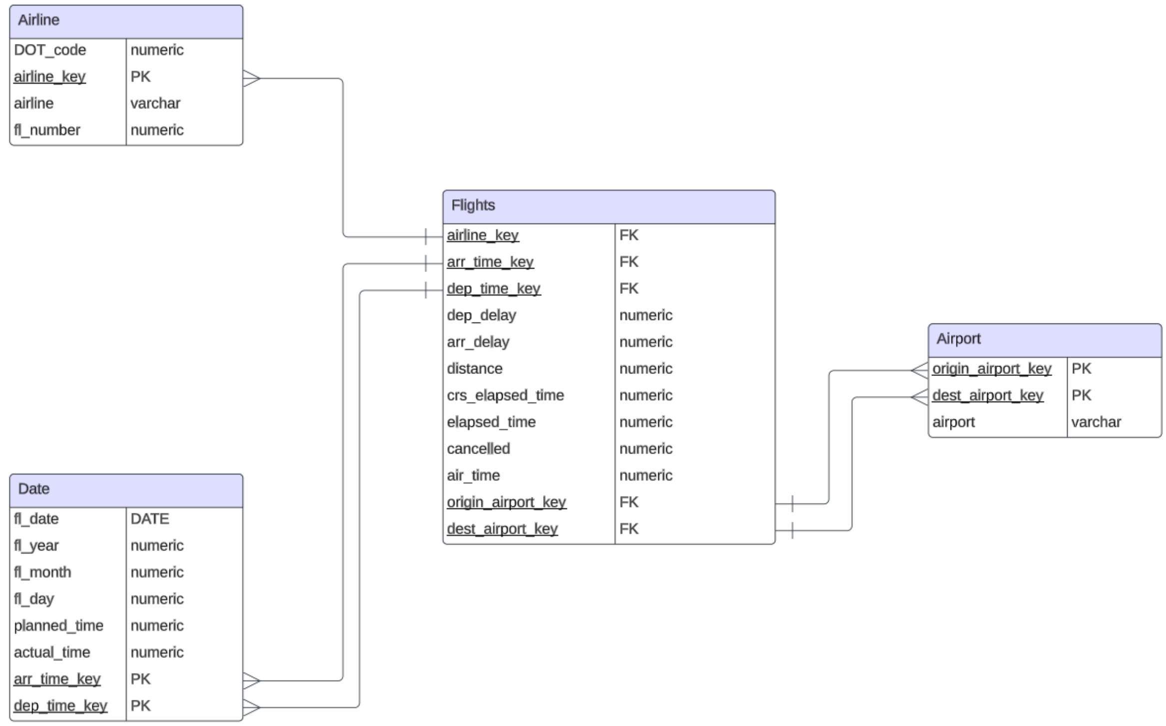
Out[42]:

fl_date	airline	dot_code	fl_number	origin	dest	crs_dep_time	dep_time	dep_delay	crs_arr_tim
2019-01-09	United Air Lines Inc.	19977	1562	FLL	EWR	1155	1151.0	-4.0	150
2022-11-19	Delta Air Lines Inc.	19790	1149	MSP	SEA	2120	2114.0	-6.0	231
2022-07-22	United Air Lines Inc.	19977	459	DEN	MSP	954	1000.0	6.0	125
2023-03-06	Delta Air Lines Inc.	19790	2295	MSP	SFO	1609	1608.0	-1.0	182
2020-02-23	Spirit Air Lines	20416	407	MCO	DFW	1840	1838.0	-2.0	204
2019-07-31	Southwest Airlines Co.	19393	665	DAL	OKC	1010	1237.0	147.0	111
2023-06-11	American Airlines Inc.	19805	2134	DCA	BOS	1010	1001.0	-9.0	115
2019-07-08	Republic Airline	20452	4464	HSV	DCA	1643	1637.0	-6.0	194
2023-02-12	Spirit Air Lines	20416	590	IAH	LAX	530	527.0	-3.0	71
2020-08-22	Alaska Airlines Inc.	19930	223	SEA	FAI	2125	2116.0	-9.0	235

More ETL with SQL

In [43]:
```
from IPython.display import Image
Image(url="https://github.com/sophietitlebaum1/Final_Project_/blob/b1927e181e7dfbb3e61
```

Out[43]:



Work on Airport dimension, modify fact and link them together

First we will union the origin and destination to see the unique number of airports. This is unioned to remove any duplicates within the origin airport and destination airport.

As we can see, there are 380 unique airports in this dataset.

Now we can create a new dimension table to house the unique airports.

In [44]:

```
%%sql
DROP TABLE IF EXISTS airport;

CREATE TABLE airport (
    key SERIAL PRIMARY KEY,
    airport VARCHAR(3) UNIQUE
);
```

```
 * postgresql://student@/Final_Airline_Dataset3
Done.
Done.
```

Out[44]: []

In [45]:

```
%%sql
INSERT INTO airport (airport)
SELECT DISTINCT origin as airport FROM flights
UNION
SELECT DISTINCT dest as airport FROM flights;
```

```
 * postgresql://student@/Final_Airline_Dataset3
380 rows affected.
```

Out[45]: []

In [46]:
```
%%sql
SELECT COUNT(*) FROM airport;
```

```
 * postgresql://student@/Final_Airline_Dataset3
1 rows affected.
```

Out[46]:

count
380

We add these new identifiers (surrogate key) back to the fact table.

In [47]:
```
%%sql
ALTER TABLE flights
ADD COLUMN origin_airport_key INTEGER,
ADD CONSTRAINT fk_origin_airport
    FOREIGN KEY (origin_airport_key)
    REFERENCES airport (key);
```

```
 * postgresql://student@/Final_Airline_Dataset3
Done.
```

Out[47]: []

In [48]:
```
%%sql
SELECT * FROM flights LIMIT 10;
```

```
 * postgresql://student@/Final_Airline_Dataset3
10 rows affected.
```

Out[48]:

fl_date	airline	dot_code	fl_number	origin	dest	crs_dep_time	dep_time	dep_delay	crs_arr_tim
2019-01-09	United Air Lines Inc.	19977	1562	FLL	EWR	1155	1151.0	-4.0	150
2022-11-19	Delta Air Lines Inc.	19790	1149	MSP	SEA	2120	2114.0	-6.0	231
2022-07-22	United Air Lines Inc.	19977	459	DEN	MSP	954	1000.0	6.0	125
2023-03-06	Delta Air Lines Inc.	19790	2295	MSP	SFO	1609	1608.0	-1.0	182
2020-02-23	Spirit Air Lines	20416	407	MCO	DFW	1840	1838.0	-2.0	204
2019-07-31	Southwest Airlines Co.	19393	665	DAL	OKC	1010	1237.0	147.0	111
2023-06-11	American Airlines Inc.	19805	2134	DCA	BOS	1010	1001.0	-9.0	115
2019-07-08	Republic Airline	20452	4464	HSV	DCA	1643	1637.0	-6.0	194
2023-02-12	Spirit Air Lines	20416	590	IAH	LAX	530	527.0	-3.0	71
2020-08-22	Alaska Airlines	19930	223	SEA	FAI	2125	2116.0	-9.0	235

In [49]:

```
%%sql
UPDATE flights
SET origin_airport_key = airport.key
FROM airport
WHERE flights.origin = airport.airport;
```

```
 * postgresql://student@/Final_Airline_Dataset3
3000000 rows affected.
```

Out[49]: []

In [50]:

```
%%sql
ALTER TABLE flights
ADD COLUMN dest_airport_key INTEGER,
ADD CONSTRAINT fk_dest_airport
    FOREIGN KEY (dest_airport_key)
    REFERENCES airport (key);
```

```
 * postgresql://student@/Final_Airline_Dataset3
Done.
```

Out[50]: []

In [51]:

```
%%sql
SELECT * FROM flights
LIMIT 10;
```

```
 * postgresql://student@/Final_Airline_Dataset3
10 rows affected.
```

Out[51]:

fl_date	airline	dot_code	fl_number	origin	dest	crs_dep_time	dep_time	dep_delay	crs_arr_time
2022-03-26	Horizon Air	19687	2149	ANC	FAI	2030	None	None	2135
2020-04-06	Republic Airline	20452	5807	SDF	LGA	1225	None	None	1431
2019-01-24	Republic Airline	20452	5983	LGA	DFW	1959	None	None	2321
2022-08-10	Envoy Air	20398	3324	DFW	LBB	1654	None	None	1802
2019-01-21	JetBlue Airways	20409	2601	BUF	JFK	1712	None	None	1842
2022-12-22	Envoy Air	20398	3366	CLT	ATW	2225	None	None	2343
2021-02-14	Envoy Air	20398	3668	DFW	LAW	1050	None	None	1158
2023-03-27	JetBlue Airways	20409	1124	LAX	JFK	1635	None	None	105
2020-05-08	Allegiant Air	20368	2795	BOS	AVL	1100	None	None	1320
2022-12-23	Endeavor Air Inc.	20363	4673	JFK	CLT	1929	None	None	2150

In [52]:

```
%%sql
UPDATE flights
SET dest_airport_key = airport.key
FROM airport
WHERE flights.dest = airport.airport;
```

```
 * postgresql://student@/Final_Airline_Dataset3
3000000 rows affected.
```

Out[52]: []

In [53]:

```
%%sql
SELECT * FROM flights LIMIT 10;
```

```
 * postgresql://student@/Final_Airline_Dataset3
10 rows affected.
```

Out[53]:

fl_date	airline	dot_code	fl_number	origin	dest	crs_dep_time	dep_time	dep_delay	crs_arr_time
2022-12-23	Endeavor Air Inc.	20363	4673	JFK	CLT	1929	None	None	2150
2020-04-03	Envoy Air	20398	4064	XNA	ORD	1229	None	None	1417
2019-08-20	Mesa Airlines Inc.	20378	5901	DFW	GPT	1659	None	None	1836
2022-12-23	Envoy Air	20398	3810	ORD	FNT	1258	None	None	1511
2022-04-03	Spirit Air Lines	20416	505	AUS	LAS	1910	None	None	2007
2020-04-23	PSA Airlines Inc.	20397	5585	CLT	SHV	1810	None	None	1939
2022-05-16	Endeavor Air Inc.	20363	5226	JFK	CHS	1820	None	None	2029
2022-03-07	PSA Airlines Inc.	20397	5398	DCA	BDL	1659	None	None	1817
2019-04-19	Envoy Air	20398	3434	MIA	TYS	2130	None	None	2343
2020-	Endeavor	20363	4863	RDU	PHL	1156	None	None	1328

In [54]:

```
%%sql
ALTER TABLE flights
DROP COLUMN origin,
DROP COLUMN dest;
```

```
 * postgresql://student@/Final_Airline_Dataset3
Done.
```

Out[54]: []

In [55]:

```
%%sql
SELECT *
FROM flights
LIMIT 10;
```

```
 * postgresql://student@/Final_Airline_Dataset3
10 rows affected.
```

Out[55]:

fl_date	airline	dot_code	fl_number	crs_dep_time	dep_time	dep_delay	crs_arr_time	arr_time	ar
2020-03-26	Endeavor Air Inc.	20363	4863	1156	None	None	1328	None	
2022-05-02	Spirit Air Lines	20416	3164	1320	None	None	1540	None	
2022-02-03	Envoy Air	20398	4210	845	None	None	1002	None	
2021-12-22	Mesa Airlines Inc.	20378	6013	1240	None	None	1615	None	
2019-01-30	Mesa Airlines Inc.	20378	5938	722	None	None	955	None	
2023-07-27	Endeavor Air Inc.	20363	4924	1550	None	None	1816	None	
2023-04-12	JetBlue Airways	20409	17	1829	None	None	1953	None	
2020-03-22	Envoy Air	20398	4241	1615	None	None	1728	None	
2022-08-24	Mesa Airlines Inc.	20378	6100	1818	None	None	1950	None	
2022-	Republic	20452	4378	1848	None	None	2009	None	

Work on Date dimension, modify fact and link them together

In [56]:

```
%%sql
SELECT DISTINCT fl_date,
    fl_year AS year,
    fl_month AS month,
    fl_day AS day,
    crs_dep_time,
    dep_time,
    crs_arr_time,
    arr_time
FROM flights
LIMIT 10;
```

```
 * postgresql://student@/Final_Airline_Dataset3
10 rows affected.
```

Out[56]:

fl_date	year	month	day	crs_dep_time	dep_time	crs_arr_time	arr_time
2019-01-01	2019	1	1	19	17.0	625	619.0
2019-01-01	2019	1	1	40	38.0	814	747.0
2019-01-01	2019	1	1	115	115.0	715	654.0
2019-01-01	2019	1	1	238	228.0	542	530.0
2019-01-01	2019	1	1	239	245.0	433	439.0
2019-01-01	2019	1	1	310	346.0	519	614.0
2019-01-01	2019	1	1	353	428.0	704	733.0
2019-01-01	2019	1	1	500	453.0	550	547.0
2019-01-01	2019	1	1	500	456.0	620	612.0
2019-01-01	2019	1	1	500	459.0	852	925.0

In [57]:

```
%%sql
DROP TABLE IF EXISTS date;

CREATE TABLE date (
    key SERIAL PRIMARY KEY,
    fl_date DATE,
    fl_year INTEGER,
    fl_month INTEGER,
    fl_day INTEGER,
    planned_time numeric,
    actual_time numeric
);
```

```
 * postgresql://student@/Final_Airline_Dataset3
Done.
Done.
```

Out[57]: []

In [58]:

```
%%sql
INSERT INTO date (fl_date, fl_year, fl_month, fl_day, planned_time, actual_time)
SELECT DISTINCT fl_date,
    fl_year AS year,
    fl_month AS month,
    fl_day AS day,
    crs_arr_time as planned_time,
    arr_time as actual_time
FROM flights
UNION
SELECT DISTINCT fl_date,
    fl_year AS year,
    fl_month AS month,
    fl_day AS day,
    crs_dep_time as planned_time,
    dep_time as actual_time
FROM flights;
```

```
 * postgresql://student@/Final_Airline_Dataset3
5476008 rows affected.
```

Out[58]: []

In [59]:
```
%%sql
SELECT * FROM date
LIMIT 10
```

```
 * postgresql://student@/Final_Airline_Dataset3
10 rows affected.
```

Out[59]:

key	fl_date	fl_year	fl_month	fl_day	planned_time	actual_time
1	2019-01-01	2019	1	1	5	21.0
2	2019-01-01	2019	1	1	7	2351.0
3	2019-01-01	2019	1	1	9	3.0
4	2019-01-01	2019	1	1	10	9.0
5	2019-01-01	2019	1	1	10	48.0
6	2019-01-01	2019	1	1	12	12.0
7	2019-01-01	2019	1	1	14	2355.0
8	2019-01-01	2019	1	1	15	125.0
9	2019-01-01	2019	1	1	17	16.0
10	2019-01-01	2019	1	1	18	31.0

In [60]:
```
%%sql
ALTER TABLE flights
ADD COLUMN arr_time_key INTEGER,
ADD CONSTRAINT fk_arr_time
    FOREIGN KEY (arr_time_key)
    REFERENCES date (key);
```

```
 * postgresql://student@/Final_Airline_Dataset3
Done.
```

Out[60]: []

In [61]:
```
%%sql
UPDATE flights
SET arr_time_key = date.key
FROM date
WHERE flights.arr_time = date.actual_time
AND flights.fl_date = date.fl_date;
```

```
 * postgresql://student@/Final_Airline_Dataset3
2920058 rows affected.
```

Out[61]: []

In [62]:
```
%%sql
ALTER TABLE flights
ADD COLUMN dep_time_key INTEGER,
ADD CONSTRAINT fk_dep_time
    FOREIGN KEY (dep_time_key)
    REFERENCES date (key);
```

```
 * postgresql://student@/Final_Airline_Dataset3
Done.
```

Out[62]: []

In [63]:
```
%%sql
UPDATE flights
SET dep_time_key = date.key
FROM date
WHERE flights.dep_time = date.actual_time
AND flights.fl_date = date.fl_date;
```

```
 * postgresql://student@/Final_Airline_Dataset3
2922385 rows affected.
```

Out[63]: []

In [64]:
```
%%sql
SELECT * FROM flights
WHERE cancelled != 1
LIMIT 10
```

```
 * postgresql://student@/Final_Airline_Dataset3
10 rows affected.
```

Out[64]:

fl_date	airline	dot_code	fl_number	crs_dep_time	dep_time	dep_delay	crs_arr_time	arr_time	ar
2022-01-27	Frontier Airlines Inc.	20436	571	2254	1.0	67.0	29	133.0	
2023-07-20	Allegiant Air	20368	3110	1901	1855.0	-6.0	2158	133.0	
2019-02-24	JetBlue Airways	20409	2067	1400	1742.0	222.0	1700	137.0	
2021-12-27	JetBlue Airways	20409	355	1830	2019.0	109.0	2200	147.0	
2021-07-03	JetBlue Airways	20409	2019	1645	1930.0	165.0	2100	148.0	
2022-11-04	Envoy Air	20398	3576	2130	28.0	178.0	2245	159.0	
2019-08-21	American Airlines Inc.	19805	1328	1447	1441.0	-6.0	1542	204.0	
2022-09-08	Spirit Air Lines	20416	505	2050	2047.0	-3.0	2147	228.0	
2023-02-08	Envoy Air	20398	3759	2123	2120.0	-3.0	2258	229.0	
2019-10-06	American Airlines Inc.	19805	1316	1201	1204.0	3.0	1747	239.0	

One Final Cleanup

We noticed that 802 flights have departure times, but no arrival times. Additionally, these flights were not indicated to have been cancelled. Therefore, to clean the data, we will remove these 802 instances.

In [65]:
```
%%sql
DELETE FROM flights
WHERE cancelled != 1
AND arr_time IS NULL
```

```
 * postgresql://student@/Final_Airline_Dataset3
802 rows affected.
```

Out[65]: []

In [66]:
```
%%sql
ALTER TABLE flights
DROP COLUMN fl_date,
DROP COLUMN crs_dep_time,
DROP COLUMN dep_time,
DROP COLUMN crs_arr_time,
DROP COLUMN arr_time,
DROP COLUMN fl_month,
DROP COLUMN fl_year,
DROP COLUMN fl_day;
```

```
 * postgresql://student@/Final_Airline_Dataset3
Done.
```

Out[66]: []

In [67]:
```
%%sql
SELECT * FROM flights
LIMIT 10
```

```
 * postgresql://student@/Final_Airline_Dataset3
10 rows affected.
```

Out[67]:

airline	dot_code	fl_number	dep_delay	arr_delay	cancelled	crs_elapsed_time	elapsed_time	air_t
American Airlines Inc.	19805	1316	3.0	None	0	226.0	None	N
Envoy Air	20398	3740	22.0	None	0	199.0	None	N
Delta Air Lines Inc.	19790	796	56.0	None	0	238.0	None	N
JetBlue Airways	20409	820	342.0	None	0	207.0	None	N
JetBlue Airways	20409	152	161.0	None	0	173.0	None	N
Southwest Airlines Co.	19393	4020	59.0	None	0	105.0	None	N
Delta Air Lines Inc.	19790	430	-1.0	None	0	340.0	None	N
JetBlue Airways	20409	403	84.0	77.0	0	233.0	226.0	2(
JetBlue Airways	20409	2711	None	None	1	159.0	None	N
JetBlue	20409	403	38.0	41.0	0	230.0	233.0	1!

Work on Airline dimension, modify fact and link them together

In [68]:

```
%%sql
DROP TABLE IF EXISTS airline;

CREATE TABLE airline (
    key SERIAL PRIMARY KEY,
    airline varchar(40),
    DOT_code numeric,
    fl_number numeric
);
```

```
 * postgresql://student@/Final_Airline_Dataset3
Done.
Done.
```

Out[68]: []

In [69]:

```
%%sql
INSERT INTO airline (airline, dot_code, fl_number)
SELECT DISTINCT airline, dot_code, fl_number
FROM flights;
```

```
 * postgresql://student@/Final_Airline_Dataset3
37599 rows affected.
```

Out[69]: []

In [70]:
```
%%sql
SELECT * FROM airline
LIMIT 10
```

```
 * postgresql://student@/Final_Airline_Dataset3
10 rows affected.
```

Out[70]:

key	airline	dot_code	fl_number
1	Alaska Airlines Inc.	19930	1
2	Alaska Airlines Inc.	19930	2
3	Alaska Airlines Inc.	19930	3
4	Alaska Airlines Inc.	19930	4
5	Alaska Airlines Inc.	19930	5
6	Alaska Airlines Inc.	19930	6
7	Alaska Airlines Inc.	19930	7
8	Alaska Airlines Inc.	19930	8
9	Alaska Airlines Inc.	19930	9
10	Alaska Airlines Inc.	19930	10

In [71]:
```
%%sql
ALTER TABLE flights
ADD COLUMN airline_key INTEGER,
ADD CONSTRAINT fk_airline
    FOREIGN KEY (airline_key)
    REFERENCES airline (key);
```

```
 * postgresql://student@/Final_Airline_Dataset3
Done.
```

Out[71]: []

In [72]:
```
%%sql
SELECT * FROM flights  LIMIT 10;
```

```
 * postgresql://student@/Final_Airline_Dataset3
10 rows affected.
```

Out[72]:

airline	dot_code	fl_number	dep_delay	arr_delay	cancelled	crs_elapsed_time	elapsed_time	air_t
JetBlue Airways	20409	403	38.0	41.0	0	230.0	233.0	1!
Envoy Air	20398	3409	None	None	1	59.0	None	N
Southwest Airlines Co.	19393	2347	46.0	None	0	140.0	None	N
Endeavor Air Inc.	20363	5072	None	None	1	74.0	None	N
Frontier Airlines Inc.	20436	486	-9.0	-28.0	0	215.0	196.0	1'
JetBlue Airways	20409	2269	None	None	1	206.0	None	N
American Airlines Inc.	19805	1682	-3.0	1.0	0	197.0	201.0	1'
Spirit Air Lines	20416	1691	None	None	1	143.0	None	N
American Airlines Inc.	19805	640	-4.0	-23.0	0	225.0	206.0	1!
Mesa Airlines	20378	5953	None	None	1	109.0	None	N

In [73]:

```
%%sql
UPDATE flights
SET airline_key = airline.key
FROM airline
WHERE flights.airline = airline.airline
AND flights.dot_code = airline.dot_code
AND flights.fl_number = airline.fl_number;
```

```
 * postgresql://student@/Final_Airline_Dataset3
2999198 rows affected.
```

Out[73]: []

In [74]:

```
%%sql
SELECT *
FROM flights
LIMIT 10
```

```
 * postgresql://student@/Final_Airline_Dataset3
10 rows affected.
```

Out[74]:

airline	dot_code	fl_number	dep_delay	arr_delay	cancelled	crs_elapsed_time	elapsed_time	air_tim
JetBlue Airways	20409	1371	17.0	30.0	0	176.0	189.0	158.(
JetBlue Airways	20409	1371	15.0	6.0	0	176.0	167.0	135.(
JetBlue Airways	20409	1371	-10.0	-7.0	0	176.0	179.0	160.(
JetBlue Airways	20409	1371	-1.0	-18.0	0	181.0	164.0	150.(
JetBlue Airways	20409	1371	-22.0	-41.0	0	176.0	157.0	143.(
JetBlue Airways	20409	1371	1.0	-2.0	0	172.0	169.0	142.(
JetBlue Airways	20409	1371	-11.0	33.0	0	170.0	214.0	154.(
JetBlue Airways	20409	1371	36.0	41.0	0	192.0	197.0	142.(
JetBlue Airways	20409	1371	18.0	2.0	0	203.0	187.0	147.(
JetBlue	20409	1371	17.0	18.0	0	188.0	189.0	143.(

In [75]:

```
%%sql
ALTER TABLE flights
DROP COLUMN airline,
DROP COLUMN dot_code,
DROP COLUMN fl_number;
```

```
 * postgresql://student@/Final_Airline_Dataset3
Done.
```

Out[75]: []

In [76]:

```
%%sql
SELECT *
FROM flights
LIMIT 10
```

```
 * postgresql://student@/Final_Airline_Dataset3
10 rows affected.
```

Out[76]:

dep_delay	arr_delay	cancelled	crs_elapsed_time	elapsed_time	air_time	distance	origin_airport_key
17.0	30.0	0	176.0	189.0	158.0	1076.0	354
15.0	6.0	0	176.0	167.0	135.0	1076.0	354
-10.0	-7.0	0	176.0	179.0	160.0	1076.0	354
-1.0	-18.0	0	181.0	164.0	150.0	1076.0	354
-22.0	-41.0	0	176.0	157.0	143.0	1076.0	354
1.0	-2.0	0	172.0	169.0	142.0	1076.0	354
-11.0	33.0	0	170.0	214.0	154.0	1076.0	354
36.0	41.0	0	192.0	197.0	142.0	1076.0	354
18.0	2.0	0	203.0	187.0	147.0	1076.0	354

Data Exploration

1. In which year did airlines have the highest percentage of departure delays?

In [77]:

```
%%sql result1 <<
SELECT
    date.fl_year,
    COUNT(*) AS total_flights,
    SUM(CASE WHEN flights.dep_delay > 15 THEN 1 ELSE 0 END) AS delayed_flights,
    ROUND((SUM(CASE WHEN flights.dep_delay > 15 THEN 1 ELSE 0 END) * 100.0) / COUNT(*)
FROM
    flights
JOIN
    date ON flights.arr_time_key = date.key
WHERE
    flights.cancelled != 1
GROUP BY
    date.fl_year
ORDER BY
    date.fl_year ASC
```

```
 * postgresql://student@/Final_Airline_Dataset3
5 rows affected.
Returning data to local variable result1
```

In [78]:

```
df1 = result1.DataFrame()
df1
```

Out[78]:

	fl_year	total_flights	delayed_flights	percentage_delays
0	2019	743788	134020	18.02
1	2020	450518	39293	8.72
2	2021	600960	100956	16.80
3	2022	669236	137799	20.59
4	2023	455556	100556	22.07

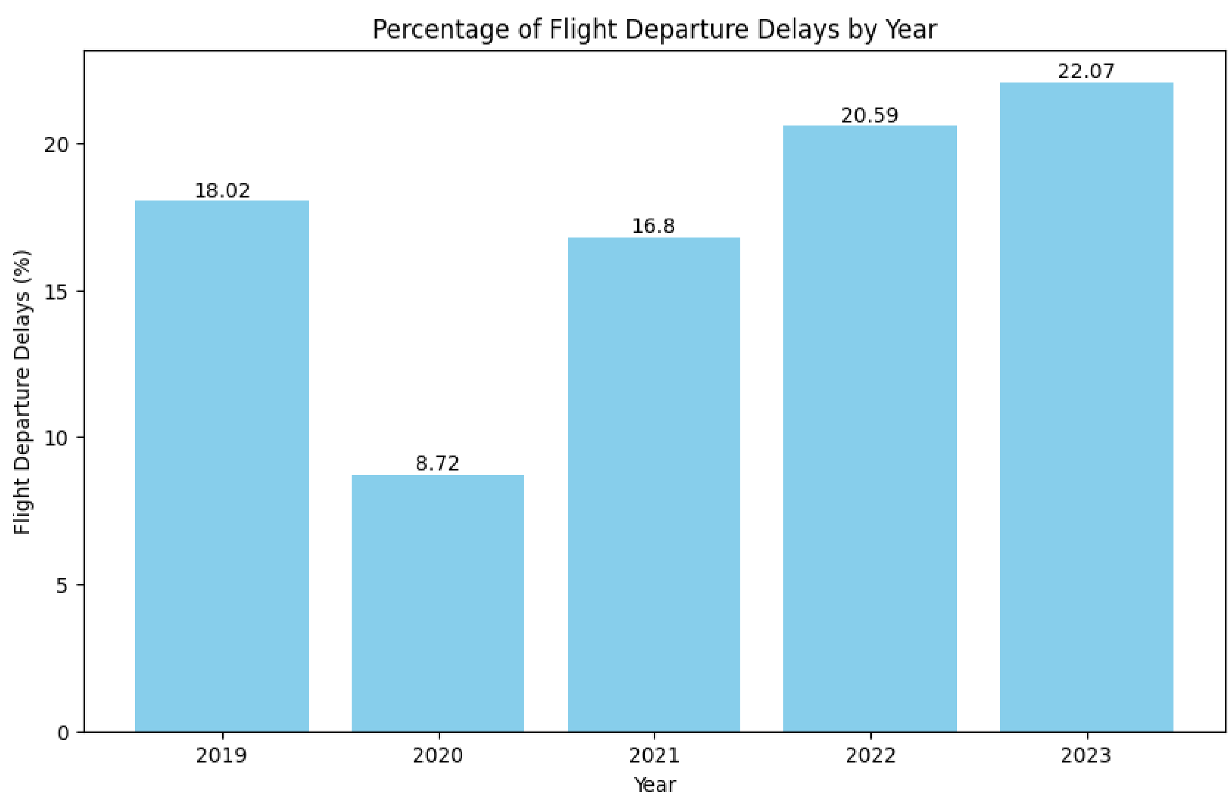
In [79]:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
bars = plt.bar(df1['fl_year'].astype(str), df1['percentage_delays'], color='skyblue')

plt.xlabel('Year')
plt.ylabel('Flight Departure Delays (%)')
plt.title('Percentage of Flight Departure Delays by Year')
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), va='bottom', ha='c

plt.show()
```



In [80]:

```
%%sql result2 <<
SELECT
    date.fl_year,
    COUNT(*) AS total_flights,
    SUM(CASE WHEN flights.arr_delay > 15 THEN 1 ELSE 0 END) AS delayed_flights,
    ROUND((SUM(CASE WHEN flights.arr_delay > 15 THEN 1 ELSE 0 END) * 100.0) / COUNT(*)
FROM
```

```
    flights
JOIN
    date ON flights.arr_time_key = date.key
WHERE
    flights.cancelled != 1
GROUP BY
    date.fl_year
ORDER BY
    date.fl_year ASC
```

```
 * postgresql://student@/Final_Airline_Dataset3
5 rows affected.
Returning data to local variable result2
```

In [81]:

```
df2 = result2.DataFrame()
df2
```

Out[81]:

	fl_year	total_flights	delayed_flights	percentage_delays
0	2019	743788	136520	18.35
1	2020	450518	42212	9.37
2	2021	600960	99542	16.56
3	2022	669236	135721	20.28
4	2023	455556	101294	22.24

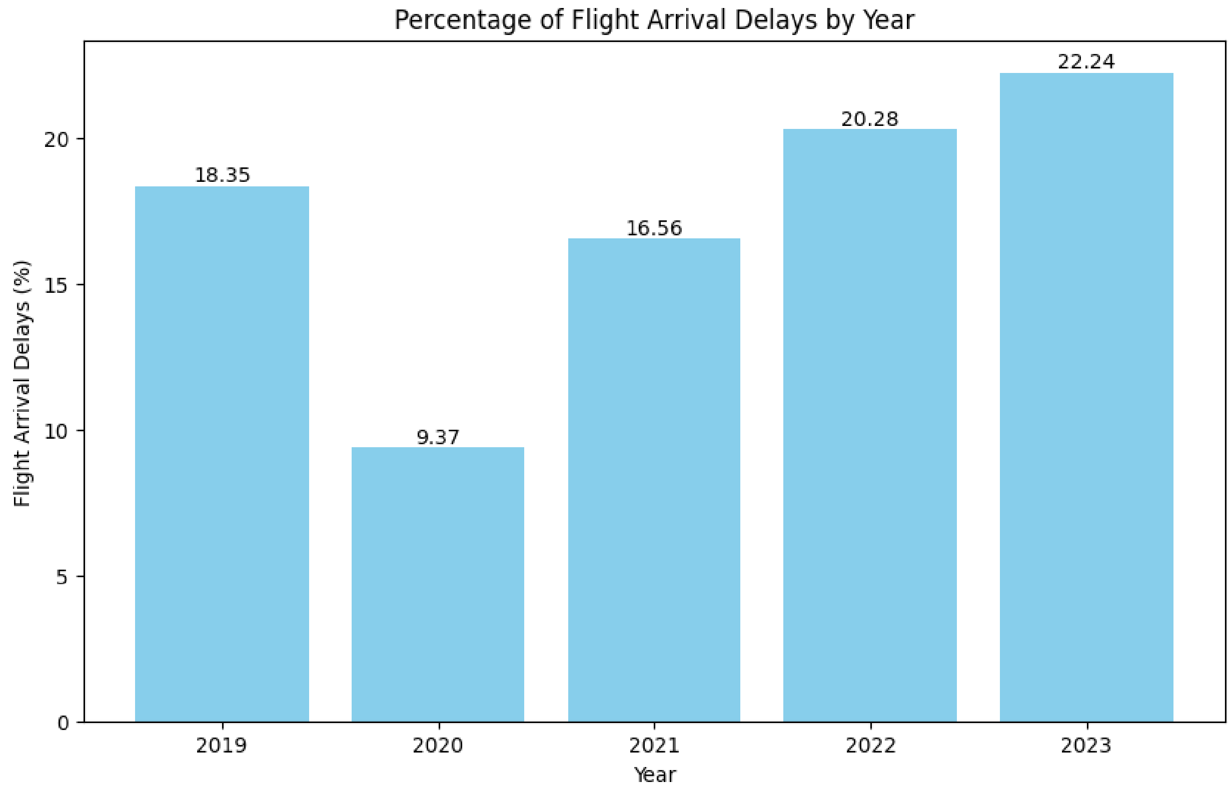
In [82]:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
bars = plt.bar(df2['fl_year'].astype(str), df2['percentage_delays'], color='skyblue')

plt.xlabel('Year')
plt.ylabel('Flight Arrival Delays (%)')
plt.title('Percentage of Flight Arrival Delays by Year')
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), va='bottom', ha='c

plt.show()
```



2. In 2023, which airports have had the highest percentage of departure delays? Note, that we are specifically looking at 2023 because the previous query indicated that this year had the highest percentage of both departure and arrival delays. If FAA is interested in a different year, the user can tweak the query to specify as necessary.

In [83]:

```
%%sql result3 <<
SELECT
    a.airport,
    COUNT(*) AS total_flights,
    SUM(CASE WHEN f.dep_delay > 15 THEN 1 ELSE 0 END) AS delayed_flights,
    ROUND((SUM(CASE WHEN f.dep_delay > 15 THEN 1 ELSE 0 END)::NUMERIC / COUNT(*)) * 10
FROM
    flights f
JOIN
    airport a ON f.origin_airport_key = a.key
JOIN
    date dep_date ON f.dep_time_key = dep_date.key
JOIN
    date arr_date ON f.arr_time_key = arr_date.key
WHERE
    EXTRACT(YEAR FROM dep_date.fl_date) = 2023
GROUP BY
    a.airport
ORDER BY
    delay_percentage DESC;
```

```
 * postgresql://student@/Final_Airline_Dataset3
348 rows affected.
Returning data to local variable result3
```

Adding a parameter, in which we are only looking at airports in which the airports' total flights are "greater than average"

In [84]:
```
%%sql result3 <<
WITH AirportFlightAverages AS (
    SELECT
        AVG(total_flights) AS avg_total_flights
    FROM (
        SELECT
            COUNT(*) AS total_flights
        FROM
            flights f
        JOIN
            date dep_date ON f.dep_time_key = dep_date.key
        WHERE
            EXTRACT(YEAR FROM dep_date.fl_date) = 2023
        GROUP BY
            f.origin_airport_key
    ) AS SubQuery
)
SELECT
    a.airport,
    COUNT(*) AS total_flights,
    SUM(CASE WHEN f.dep_delay > 15 THEN 1 ELSE 0 END) AS delayed_flights,
    ROUND((SUM(CASE WHEN f.dep_delay > 15 THEN 1 ELSE 0 END)::NUMERIC / COUNT(*)) * 10
FROM
    flights f
JOIN
    airport a ON f.origin_airport_key = a.key
JOIN
    date dep_date ON f.dep_time_key = dep_date.key
WHERE
    EXTRACT(YEAR FROM dep_date.fl_date) = 2023
GROUP BY
    a.airport
HAVING
    COUNT(*) > (SELECT avg_total_flights FROM AirportFlightAverages)
ORDER BY
    delay_percentage DESC;
```

```
 * postgresql://student@/Final_Airline_Dataset3
67 rows affected.
Returning data to local variable result3
```

In [85]:
```
df3 = result3.DataFrame()
top_20 = df3.head(20)
top_20
```

Out[85]:

	airport	total_flights	delayed_flights	delay_percentage
0	FLL	6066	2046	33.73
1	BWI	6238	1910	30.62
2	DEN	19001	5762	30.32
3	MCO	10961	3302	30.12
4	LAS	12665	3739	29.52
5	MDW	5623	1624	28.88
6	SJU	2215	625	28.22
7	MIA	6746	1870	27.72
8	EWR	9257	2504	27.05
9	DFW	18860	4872	25.83
10	PBI	1852	472	25.49
11	IAH	7679	1939	25.25
12	JFK	8973	2185	24.35
13	MSY	3366	812	24.12
14	ATL	22467	5366	23.88
15	DAL	4931	1173	23.79
16	AUS	6183	1442	23.32
17	CLT	12838	2972	23.15
18	STL	4046	921	22.76
19	BOS	9326	2114	22.67

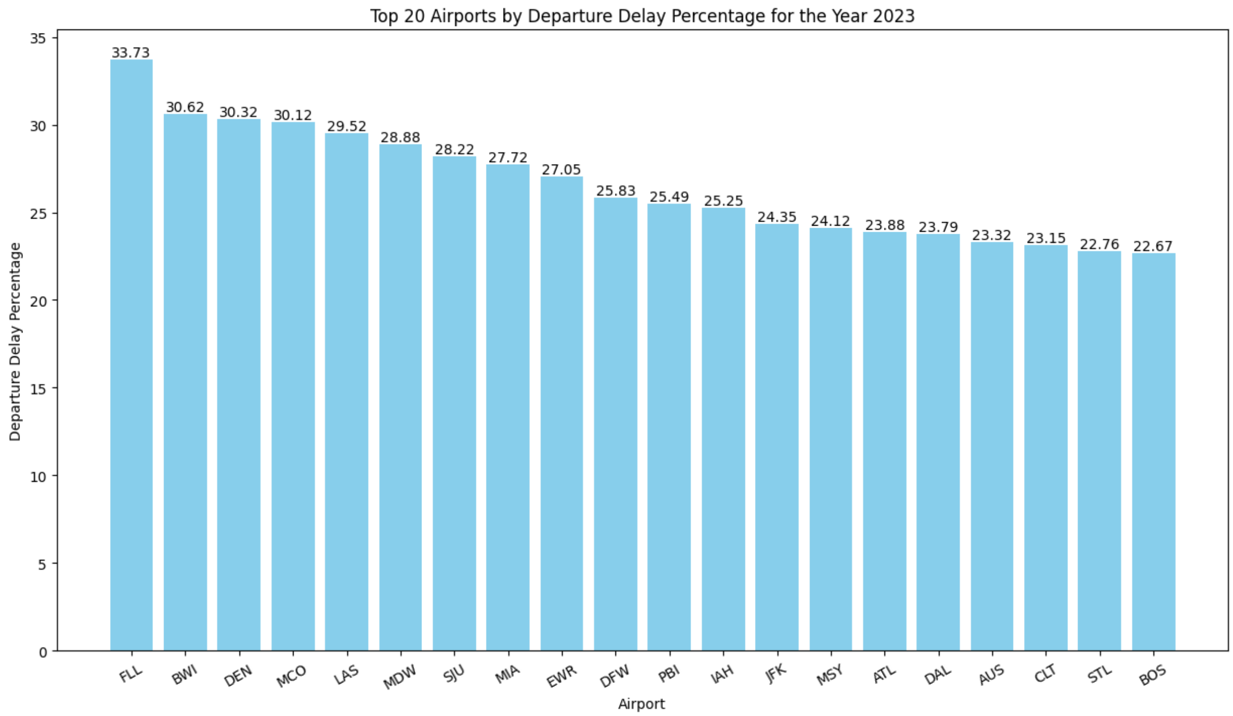
In [86]:

```
plt.figure(figsize=(15, 8))
bars = plt.bar(top_20['airport'], top_20['delay_percentage'], color='skyblue')

plt.xlabel('Airport')
plt.ylabel('Departure Delay Percentage')
plt.title('Top 20 Airports by Departure Delay Percentage for the Year 2023')
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), va='bottom', ha='c

plt.xticks(rotation=30)

plt.show()
```



3. In general, which airports had the highest percentage of delays?

-- first we need to determine the percentage of delays for each airport

In [87]:
```
%%sql result4 <<
SELECT
    a.airport,
    COUNT(*) AS total_flights,
    SUM(CASE WHEN f.dep_delay > 15  THEN 1 ELSE 0 END) AS delayed_flights,
    ROUND((SUM(CASE WHEN f.dep_delay > 15 THEN 1 ELSE 0 END)::NUMERIC / COUNT(*)) * 10
FROM
    flights f
JOIN
    airport a ON f.origin_airport_key = a.key
GROUP BY
    a.airport
ORDER BY
    delay_percentage DESC;
```

```
 * postgresql://student@/Final_Airline_Dataset3
380 rows affected.
Returning data to local variable result4
```

In [88]:
```
%%sql result4 <<
WITH AirportFlightAverages AS (
    SELECT
        AVG(total_flights) AS avg_total_flights
    FROM (
        SELECT
            COUNT(*) AS total_flights
        FROM
            flights f
        JOIN
            airport a ON f.origin_airport_key = a.key
        GROUP BY
```

```
            a.airport
    ) AS SubQuery
)
SELECT
    a.airport,
    COUNT(*) AS total_flights,
    SUM(CASE WHEN f.dep_delay > 15 THEN 1 ELSE 0 END) AS delayed_flights,
    ROUND((SUM(CASE WHEN f.dep_delay > 15 THEN 1 ELSE 0 END)::NUMERIC / COUNT(*)) * 10
FROM
    flights f
JOIN
    airport a ON f.origin_airport_key = a.key
GROUP BY
    a.airport
HAVING
    COUNT(*) > (SELECT avg_total_flights FROM AirportFlightAverages)
ORDER BY
    delay_percentage DESC
```

```
 * postgresql://student@/Final_Airline_Dataset3
69 rows affected.
Returning data to local variable result4
```

In [89]:

```
df4 = result4.DataFrame()
top_20_overall = df4.head(20)
top_20_overall
```

Out[89]:

	airport	total_flights	delayed_flights	delay_percentage
0	MDW	35066	8829	25.18
1	DAL	30798	7322	23.77
2	BWI	41033	9513	23.18
3	EWR	52978	12018	22.68
4	HOU	24428	5461	22.36
5	MCO	63863	14268	22.34
6	DEN	119874	26755	22.32
7	FLL	40260	8983	22.31
8	LAS	73454	15783	21.49
9	SJU	13009	2692	20.69
10	MIA	41970	8501	20.25
11	DFW	130287	25577	19.63
12	JFK	50456	9880	19.58
13	PBI	10982	2142	19.50
14	OAK	20161	3746	18.58
15	AUS	32663	6018	18.42
16	LGA	62567	11519	18.41
17	MSY	21558	3968	18.41
18	STL	26784	4845	18.09
19	BNA	37470	6774	18.08

In [90]:

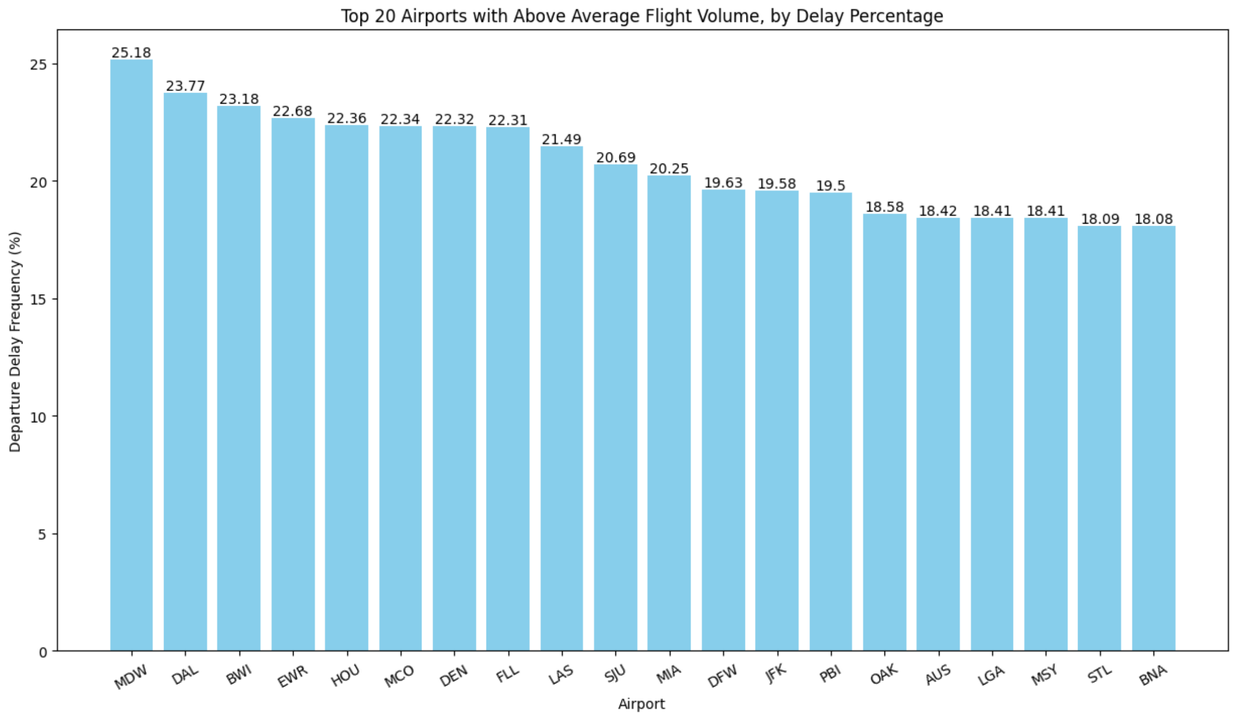
```
plt.figure(figsize=(15, 8))
bars = plt.bar(top_20_overall['airport'], top_20_overall['delay_percentage'], color='s

plt.xlabel('Airport')
plt.ylabel('Departure Delay Frequency (%)')
plt.title('Top 20 Airports with Above Average Flight Volume, by Delay Percentage')

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), va='bottom', ha='c

plt.xticks(rotation=30)

plt.show()
```



In [91]:

```
%%sql result5 <<
SELECT
    a.airport,
    COUNT(*) AS total_flights,
    SUM(CASE WHEN f.arr_delay > 15  THEN 1 ELSE 0 END) AS delayed_flights,
    ROUND((SUM(CASE WHEN f.arr_delay > 15 THEN 1 ELSE 0 END)::NUMERIC / COUNT(*)) * 10
FROM
    flights f
JOIN
    airport a ON f.origin_airport_key = a.key
GROUP BY
    a.airport
ORDER BY
    delay_percentage DESC;
```

```
 * postgresql://student@/Final_Airline_Dataset3
380 rows affected.
Returning data to local variable result5
```

In [92]:

```
%%sql result5 <<
WITH AirportFlightAverages AS (
    SELECT
        AVG(total_flights) AS avg_total_flights
    FROM (
        SELECT
            COUNT(*) AS total_flights
        FROM
            flights f
        JOIN
            airport a ON f.origin_airport_key = a.key
        GROUP BY
            a.airport
    ) AS SubQuery
)
SELECT
```

```sql
    a.airport,
    COUNT(*) AS total_flights,
    SUM(CASE WHEN f.arr_delay > 15 THEN 1 ELSE 0 END) AS delayed_flights,
    ROUND((SUM(CASE WHEN f.arr_delay > 15 THEN 1 ELSE 0 END)::NUMERIC / COUNT(*)) * 10
FROM
    flights f
JOIN
    airport a ON f.origin_airport_key = a.key
GROUP BY
    a.airport
HAVING
    COUNT(*) > (SELECT avg_total_flights FROM AirportFlightAverages)
ORDER BY
    delay_percentage DESC
```

```
 * postgresql://student@/Final_Airline_Dataset3
69 rows affected.
Returning data to local variable result5
```

In [93]:

```python
df5 = result5.DataFrame()
top_20_overall = df5.head(20)
top_20_overall
```

Out[93]:

	airport	total_flights	delayed_flights	delay_percentage
0	EWR	52978	11967	22.59
1	MCO	63863	14190	22.22
2	FLL	40260	8677	21.55
3	SJU	13009	2746	21.11
4	DEN	119874	25301	21.11
5	MIA	41970	8842	21.07
6	MDW	35066	7335	20.92
7	LAS	73454	15024	20.45
8	DFW	130287	26375	20.24
9	BWI	41033	8180	19.94
10	JFK	50456	10002	19.82
11	PBI	10982	2158	19.65
12	ORD	122249	23889	19.54
13	DAL	30798	5967	19.37
14	LGA	62567	11843	18.93
15	HOU	24428	4495	18.40
16	BOS	55388	10075	18.19
17	IAH	62526	11066	17.70
18	AUS	32663	5759	17.63
19	DCA	53269	9351	17.55

In [94]:

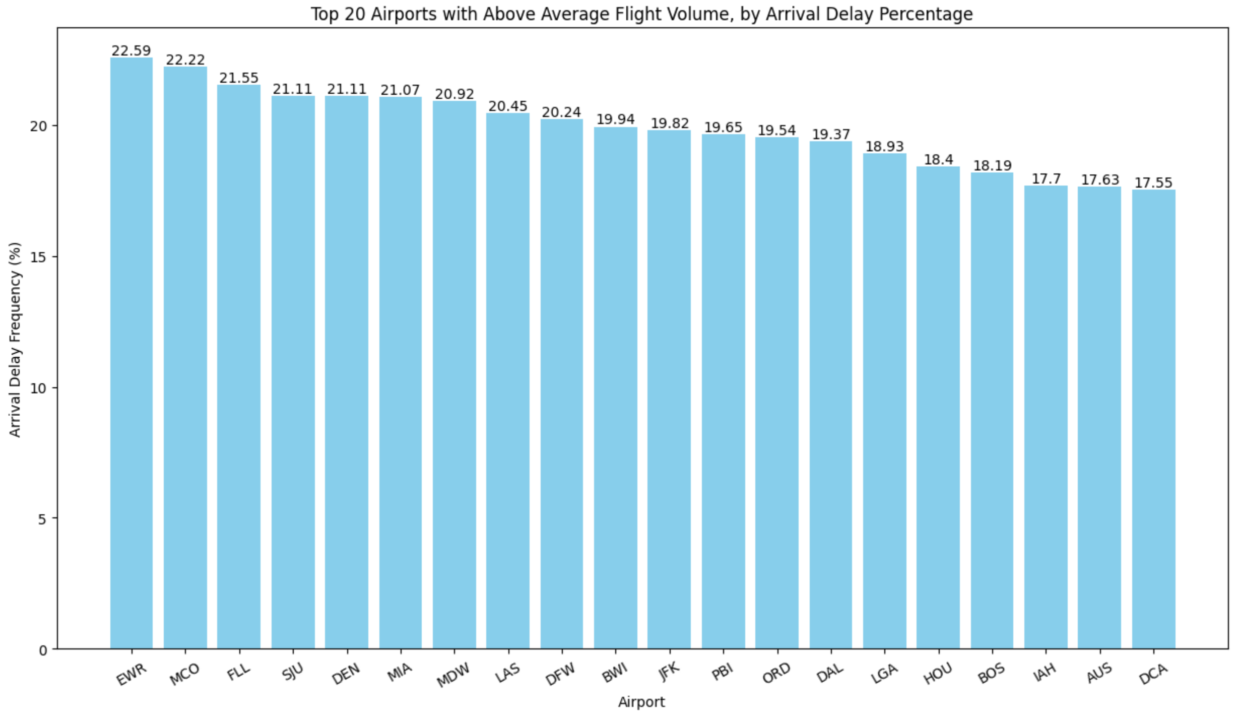
```
plt.figure(figsize=(15, 8))
bars = plt.bar(top_20_overall['airport'], top_20_overall['delay_percentage'], color='s

plt.xlabel('Airport')
plt.ylabel('Arrival Delay Frequency (%)')
plt.title('Top 20 Airports with Above Average Flight Volume, by Arrival Delay Percenta

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), va='bottom', ha='c

plt.xticks(rotation=30)

plt.show()
```



4. In general (2019-2023), which airlines had the highest percentage of delays?

In [95]:
```
%%sql result6 <<
SELECT
    a.airline,
    COUNT(*) AS total_flights,
    SUM(CASE WHEN f.dep_delay > 15 THEN 1 ELSE 0 END) AS delayed_flights,
    ROUND((SUM(CASE WHEN f.dep_delay > 15 THEN 1 ELSE 0 END) * 100.0) / COUNT(*), 2) A
FROM
    flights f
JOIN
    airline a ON f.airline_key = a.key
GROUP BY
    a.airline
ORDER BY
    percentage_delays DESC;
```

```
 * postgresql://student@/Final_Airline_Dataset3
18 rows affected.
Returning data to local variable result6
```

In [96]:
```
df6 = result6.DataFrame()
top_10_overall = df6.head(10)
top_10_overall
```

Out[96]:

	airline	total_flights	delayed_flights	percentage_delays
0	JetBlue Airways	112801	28927	25.64
1	Frontier Airlines Inc.	64456	16217	25.16
2	Allegiant Air	52720	11541	21.89
3	Southwest Airlines Co.	576336	121467	21.08
4	Spirit Air Lines	95696	19530	20.41
5	American Airlines Inc.	383053	69850	18.24
6	United Air Lines Inc.	254443	44733	17.58
7	ExpressJet Airlines LLC d/b/a aha!	19074	3213	16.84
8	Mesa Airlines Inc.	65005	10613	16.33
9	Alaska Airlines Inc.	100376	15056	15.00

In [97]:

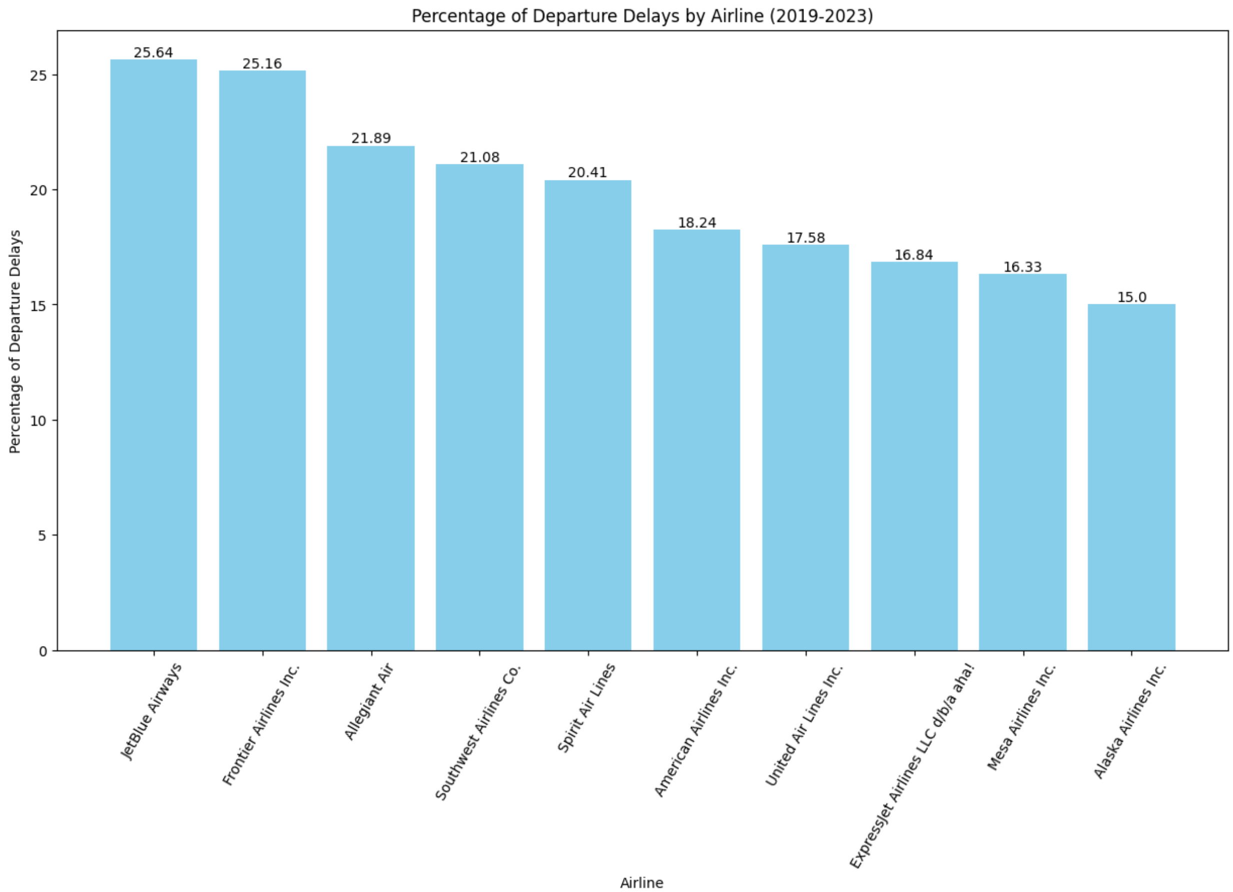
```
plt.figure(figsize=(15, 8))
bars = plt.bar(top_10_overall['airline'], top_10_overall['percentage_delays'], color='

plt.xlabel('Airline')
plt.ylabel('Percentage of Departure Delays')
plt.title('Percentage of Departure Delays by Airline (2019-2023)')

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), va='bottom', ha='c

plt.xticks(rotation=60)

plt.show()
```



In [98]:
```
%%sql result7 <<
SELECT
    a.airline,
    COUNT(*) AS total_flights,
    SUM(CASE WHEN f.arr_delay > 15 THEN 1 ELSE 0 END) AS delayed_flights,
    ROUND((SUM(CASE WHEN f.arr_delay > 15 THEN 1 ELSE 0 END) * 100.0) / COUNT(*), 2) A
FROM
    flights f
JOIN
    airline a ON f.airline_key = a.key
GROUP BY
    a.airline
ORDER BY
    percentage_delays DESC;
```

```
 * postgresql://student@/Final_Airline_Dataset3
18 rows affected.
Returning data to local variable result7
```

In [99]:
```
df7 = result7.DataFrame()
top_10_overall = df7.head(10)
top_10_overall
```

Out[99]:

	airline	total_flights	delayed_flights	percentage_delays
0	JetBlue Airways	112801	28774	25.51
1	Frontier Airlines Inc.	64456	16256	25.22
2	Allegiant Air	52720	12680	24.05
3	Spirit Air Lines	95696	19976	20.87
4	ExpressJet Airlines LLC d/b/a aha!	19074	3737	19.59
5	American Airlines Inc.	383053	71461	18.66
6	United Air Lines Inc.	254443	46021	18.09
7	Southwest Airlines Co.	576336	103143	17.90
8	Mesa Airlines Inc.	65005	11286	17.36
9	Alaska Airlines Inc.	100376	17081	17.02

In [100…

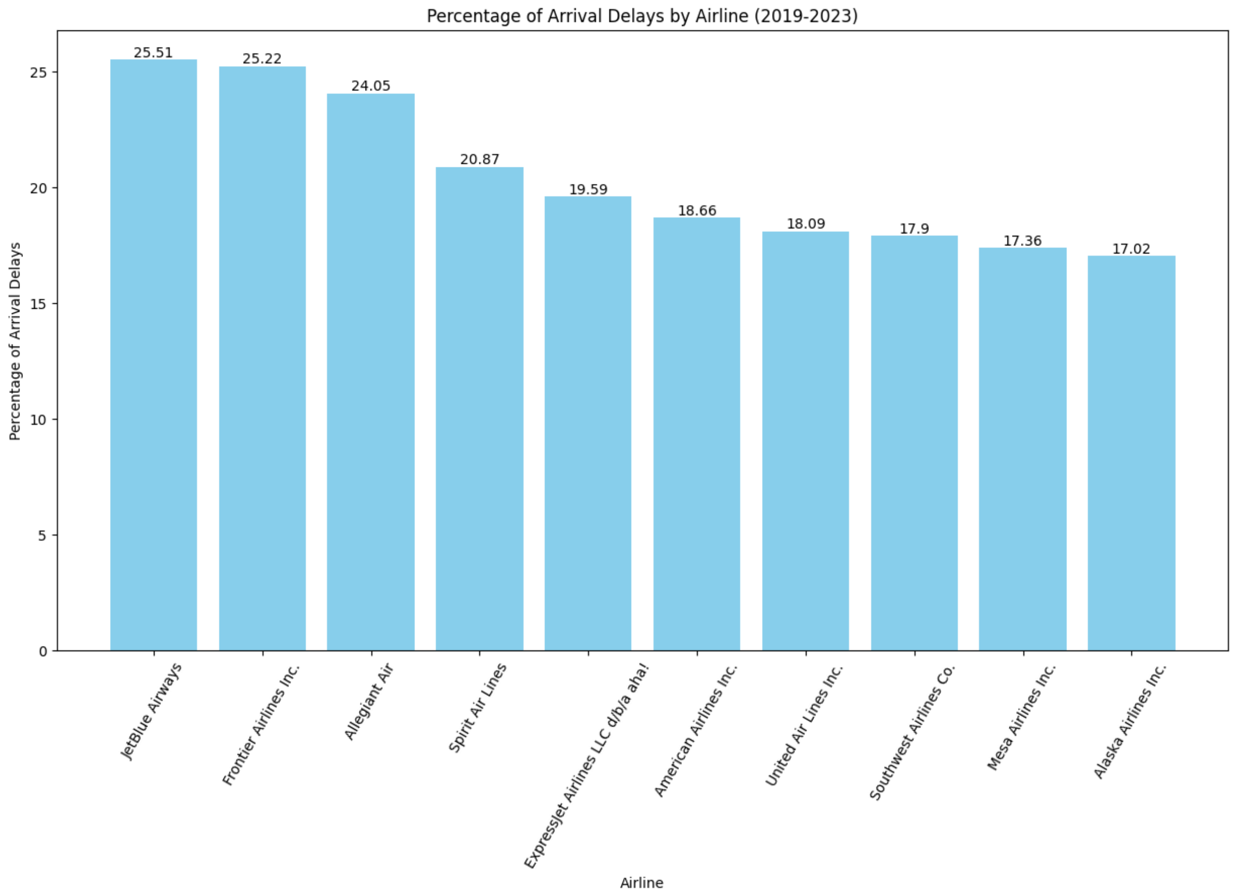
```
plt.figure(figsize=(15, 8))
bars = plt.bar(top_10_overall['airline'], top_10_overall['percentage_delays'], color='

plt.xlabel('Airline')
plt.ylabel('Percentage of Arrival Delays')
plt.title('Percentage of Arrival Delays by Airline (2019-2023)')

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), va='bottom', ha='c

plt.xticks(rotation=60)

plt.show()
```



5. In the specific year (2023), which airlines had the highest percentage of delays?

In [101…

```
%%sql result8 <<
SELECT
    a.airline,
    COUNT(*) AS total_flights,
    SUM(CASE WHEN f.dep_delay > 15 THEN 1 ELSE 0 END) AS delayed_flights,
    ROUND((SUM(CASE WHEN f.dep_delay > 15 THEN 1 ELSE 0 END) * 100.0) / COUNT(*), 2) A
FROM
    flights f
JOIN
    airline a ON f.airline_key = a.key
JOIN
    date d ON f.dep_time_key = d.key
WHERE
    d.fl_year = 2023
GROUP BY
    a.airline
ORDER BY
    percentage_delays DESC;
```

```
 * postgresql://student@/Final_Airline_Dataset3
15 rows affected.
Returning data to local variable result8
```

In [102…

```
df8 = result8.DataFrame()
top_10_2023 = df8.head(10)
top_10_2023
```

Out[102]:

	airline	total_flights	delayed_flights	percentage_delays
0	Frontier Airlines Inc.	11140	3933	35.31
1	JetBlue Airways	18790	5968	31.76
2	Spirit Air Lines	17253	5389	31.24
3	Southwest Airlines Co.	94952	24211	25.50
4	Allegiant Air	8055	2033	25.24
5	American Airlines Inc.	63201	15264	24.15
6	Hawaiian Airlines Inc.	5460	1296	23.74
7	United Air Lines Inc.	48554	11181	23.03
8	Delta Air Lines Inc.	66101	13127	19.86
9	Alaska Airlines Inc.	16685	3129	18.75

In [103…

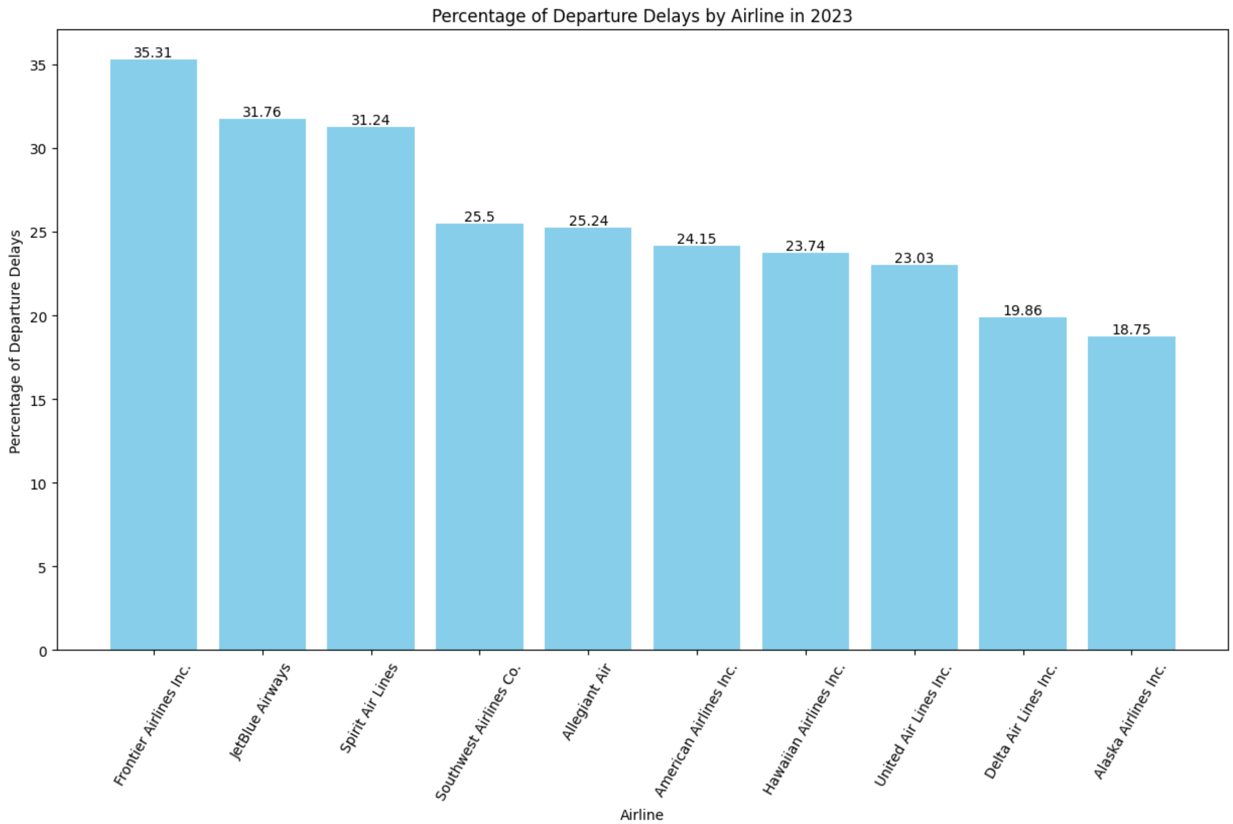
```
plt.figure(figsize=(15, 8))
bars = plt.bar(top_10_2023['airline'], top_10_2023['percentage_delays'], color='skyblu

plt.xlabel('Airline')
plt.ylabel('Percentage of Departure Delays')
plt.title('Percentage of Departure Delays by Airline in 2023')

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), va='bottom', ha='c

plt.xticks(rotation=60)

plt.show()
```



In [104…

```
%%sql result9 <<
SELECT
    a.airline,
    COUNT(*) AS total_flights,
    SUM(CASE WHEN f.arr_delay > 15 THEN 1 ELSE 0 END) AS delayed_flights,
    ROUND((SUM(CASE WHEN f.arr_delay > 15 THEN 1 ELSE 0 END) * 100.0) / COUNT(*), 2) A
FROM
    flights f
JOIN
    airline a ON f.airline_key = a.key
JOIN
    date d ON f.dep_time_key = d.key
WHERE
    d.fl_year = 2023
GROUP BY
    a.airline
ORDER BY
    percentage_delays DESC;
```

```
 * postgresql://student@/Final_Airline_Dataset3
15 rows affected.
Returning data to local variable result9
```

In [105…

```
df9 = result9.DataFrame()
top_10_2023 = df9.head(10)
top_10_2023
```

Out[105]:

	airline	total_flights	delayed_flights	percentage_delays
0	Frontier Airlines Inc.	11140	3988	35.80
1	JetBlue Airways	18790	5925	31.53
2	Spirit Air Lines	17253	5426	31.45
3	Allegiant Air	8055	2245	27.87
4	Hawaiian Airlines Inc.	5460	1479	27.09
5	American Airlines Inc.	63201	15792	24.99
6	United Air Lines Inc.	48554	11461	23.60
7	Southwest Airlines Co.	94952	21712	22.87
8	Alaska Airlines Inc.	16685	3393	20.34
9	Delta Air Lines Inc.	66101	12529	18.95

In [106…

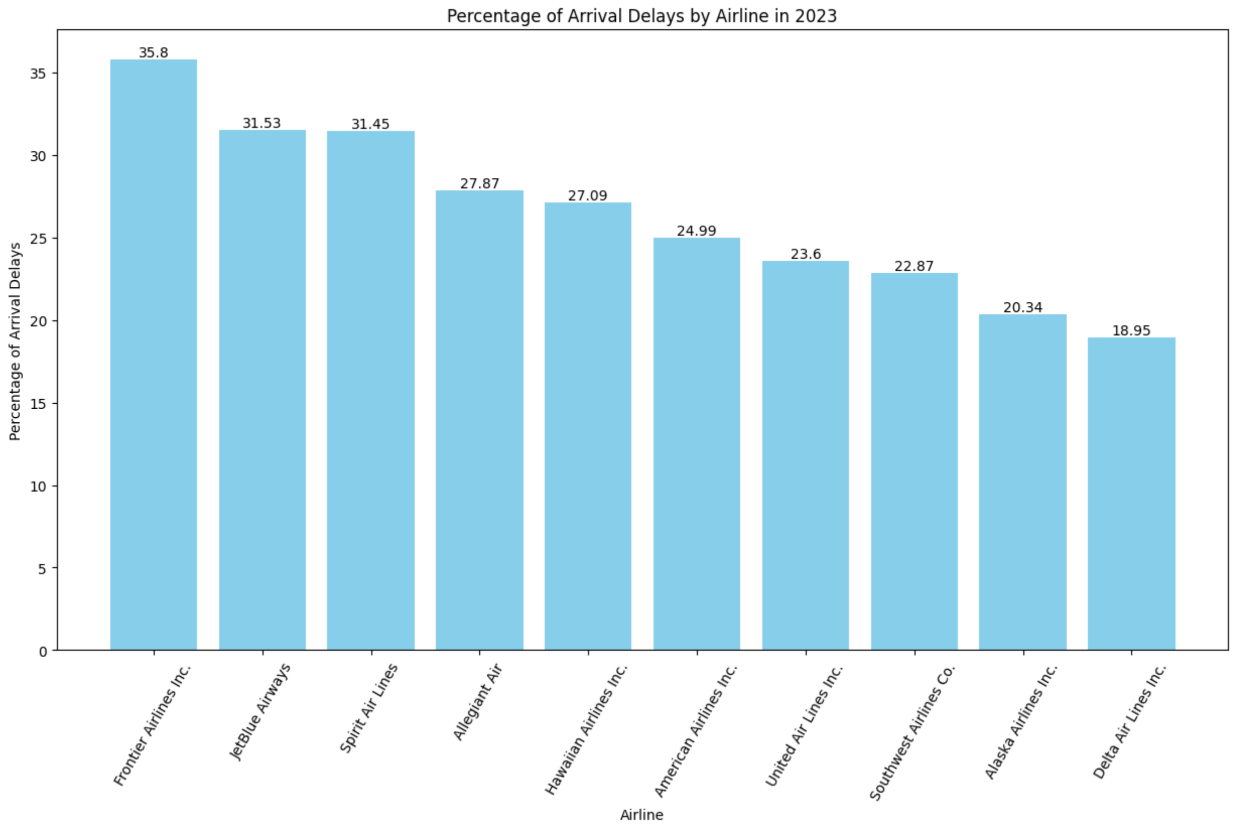
```
plt.figure(figsize=(15, 8))
bars = plt.bar(top_10_2023['airline'], top_10_2023['percentage_delays'], color='skyblu

plt.xlabel('Airline')
plt.ylabel('Percentage of Arrival Delays')
plt.title('Percentage of Arrival Delays by Airline in 2023')

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), va='bottom', ha='c

plt.xticks(rotation=60)

plt.show()
```



6. In the specific year (2023) and airport (MDW), which airlines had the highest percentage of delays, with the flights volume larger then average for this airport? The year, 2023, and the airport, MDW, comes from above questions in which this year and airport had high percentages of delays.

If FAA is interested in analyzing a different airport or year, the user can change both 'WHERE' clauses -- replace ap.airport='MDW' and d.fl_year=2023, according to the necessary analysis.

In [107…

```
%%sql resultz <<
SELECT
    a.airline,
    COUNT(*) AS total_flights,
    SUM(CASE WHEN f.dep_delay > 15 THEN 1 ELSE 0 END) AS delayed_flights,
    ROUND((SUM(CASE WHEN f.dep_delay > 15 THEN 1 ELSE 0 END) * 100.0) / COUNT(*), 2) A
FROM
    flights f
JOIN
    airline a ON f.airline_key = a.key
JOIN
    date d ON f.dep_time_key = d.key
JOIN
    airport ap ON f.origin_airport_key = ap.key
WHERE
    d.fl_year = 2023 AND ap.airport = 'MDW'
GROUP BY
    a.airline
HAVING
    COUNT(*) > (
        SELECT AVG(sub.total_flights)
        FROM (
```

```
            SELECT COUNT(*) AS total_flights
            FROM flights f2
            JOIN date d2 ON f2.dep_time_key = d2.key
            JOIN airport ap2 ON f2.origin_airport_key = ap2.key
            WHERE d2.fl_year = 2023 AND ap2.airport = 'MDW'
            GROUP BY f2.airline_key
        ) AS sub
    )
ORDER BY
    percentage_delays DESC;
```

```
 * postgresql://student@/Final_Airline_Dataset3
6 rows affected.
Returning data to local variable resultz
```

In [108…

```
dfz = resultz.DataFrame()
top_5_2023_FLL = dfz.head(8)
top_5_2023_FLL
```

Out[108]:

	airline	total_flights	delayed_flights	percentage_delays
0	Frontier Airlines Inc.	244	80	32.79
1	Southwest Airlines Co.	5143	1495	29.07
2	Delta Air Lines Inc.	90	25	27.78
3	Allegiant Air	34	8	23.53
4	Endeavor Air Inc.	43	8	18.60
5	SkyWest Airlines Inc.	69	8	11.59

In [109…

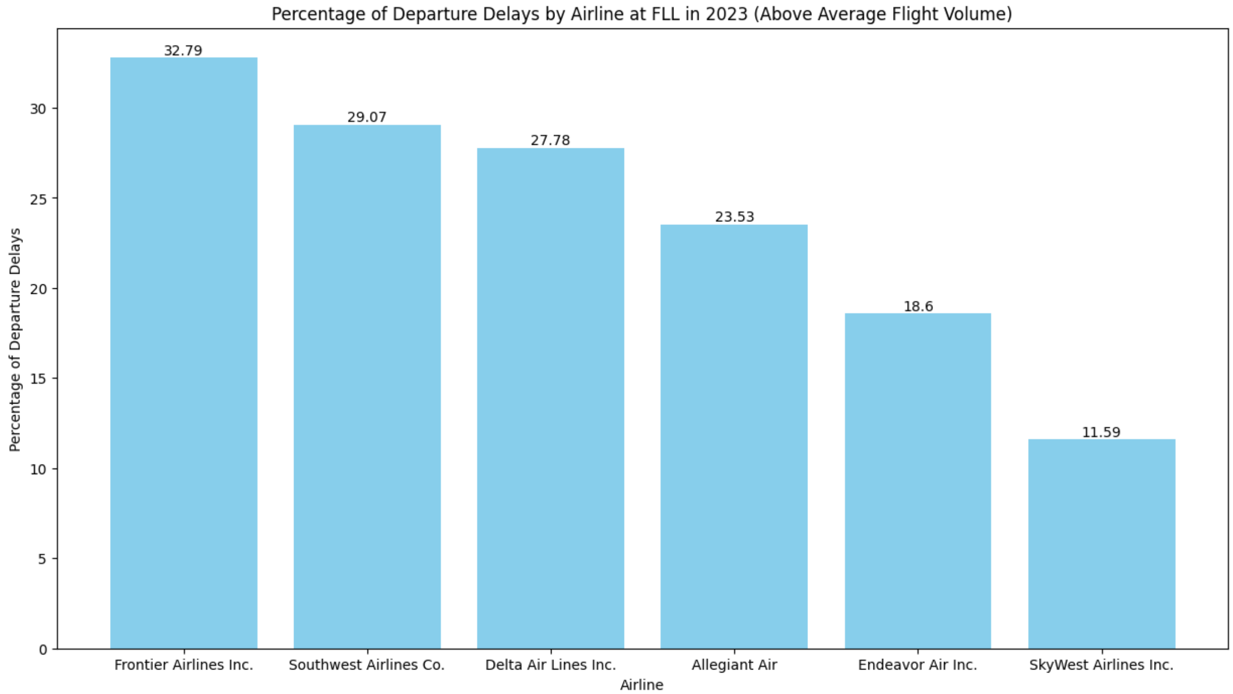
```
plt.figure(figsize=(15, 8))
bars = plt.bar(dfz['airline'], dfz['percentage_delays'], color='skyblue')

plt.xlabel('Airline')
plt.ylabel('Percentage of Departure Delays')
plt.title('Percentage of Departure Delays by Airline at FLL in 2023 (Above Average Fli

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), va='bottom', ha='c

plt.xticks(rotation=0)

plt.show()
```



7. How does the performance of the same airline vary across different airports in terms of delays and cancellations?

In [110…

```
%%sql result10 <<
SELECT
    airline.airline,
    a.airport,
    COUNT(*) AS total_flights,
    COUNT(*) FILTER (WHERE CAST(f.dep_delay AS DOUBLE PRECISION) > 15) AS dep_delay_co
    COUNT(*) FILTER (WHERE CAST(f.arr_delay AS DOUBLE PRECISION) > 15) AS arr_delay_co
    COUNT(*) FILTER (WHERE f.cancelled = '1') AS cancellation_count
FROM
    Flights f
JOIN
    airline ON f.airline_key = airline.key
JOIN
    airport a ON f.origin_airport_key = a.key
WHERE airline = 'Alaska Airlines Inc.'
GROUP BY
    airline.airline, a.airport
ORDER BY
    airline.airline, dep_delay_count DESC, total_flights DESC;
```

```
 * postgresql://student@/Final_Airline_Dataset3
91 rows affected.
Returning data to local variable result10
```

To analyze a specific airline, one can change the 'where' statement to reflect the necessary airline's analysis.

In [111…

```
df10 = result10.DataFrame()
df10
```

Out[111]:

	airline	airport	total_flights	dep_delay_count	arr_delay_count	cancellation_count
0	Alaska Airlines Inc.	SEA	30442	5020	5579	504
1	Alaska Airlines Inc.	SFO	5822	1007	1119	149
2	Alaska Airlines Inc.	LAX	5397	835	952	99
3	Alaska Airlines Inc.	PDX	7028	817	1016	144
4	Alaska Airlines Inc.	ANC	6247	598	763	113
...	...	...	...	...	...	...
86	Alaska Airlines Inc.	ELP	47	7	4	0
87	Alaska Airlines Inc.	EUG	28	6	4	0
88	Alaska Airlines Inc.	GST	39	3	3	2
89	Alaska Airlines Inc.	BLI	23	3	4	1
90	Alaska Airlines Inc.	PSC	31	2	3	0

91 rows × 6 columns

8. Are there specific airports where certain airlines consistently outperform others in terms of fewer delays and cancellations?

In [112…

```
%%sql

SELECT
    a.airport,
    airline.airline,
    COUNT(*) AS total_flights,
    COUNT(*) FILTER (WHERE CAST(f.dep_delay AS DOUBLE PRECISION) > 15) AS dep_delay_co
    COUNT(*) FILTER (WHERE CAST(f.arr_delay AS DOUBLE PRECISION) > 15) AS arr_delay_co
    COUNT(*) FILTER (WHERE f.cancelled = '1') AS cancellation_count
FROM
    Flights f
JOIN
    airline ON f.airline_key = airline.key
JOIN
    airport a ON f.origin_airport_key = a.key
GROUP BY
    a.airport, airline.airline
ORDER BY
    a.airport, airline.airline, dep_delay_count DESC, total_flights DESC
LIMIT 50;
```

```
 * postgresql://student@/Final_Airline_Dataset3
50 rows affected.
```

Out[112]:

airport	airline	total_flights	dep_delay_count	arr_delay_count	cancellation_count
ABE	Allegiant Air	579	89	119	21
ABE	Delta Air Lines Inc.	45	3	8	2
ABE	Endeavor Air Inc.	386	53	53	5
ABE	Envoy Air	179	18	23	6
ABE	ExpressJet Airlines LLC d/b/a aha!	13	1	2	2
ABE	PSA Airlines Inc.	483	53	40	13
ABE	SkyWest Airlines Inc.	338	60	60	2
ABI	Envoy Air	840	120	149	24
ABI	SkyWest Airlines Inc.	61	6	6	3
ABQ	Alaska Airlines Inc.	232	37	37	3
ABQ	Allegiant Air	52	14	12	6
ABQ	American Airlines Inc.	1137	201	200	31
ABQ	Delta Air Lines Inc.	432	34	36	6
ABQ	Envoy Air	328	30	41	7
ABQ	ExpressJet Airlines LLC d/b/a aha!	54	11	13	0
ABQ	Frontier Airlines Inc.	38	4	6	1
ABQ	Horizon Air	20	1	1	0
ABQ	JetBlue Airways	117	39	36	7
ABQ	Mesa Airlines Inc.	568	69	82	16
ABQ	Republic Airline	70	4	8	3
ABQ	SkyWest Airlines Inc.	1655	172	188	17
ABQ	Southwest Airlines Co.	4048	707	632	119
ABQ	Spirit Air Lines	43	15	15	1
ABQ	United Air Lines Inc.	485	81	82	17
ABR	SkyWest Airlines Inc.	332	32	34	4
ABY	Endeavor Air Inc.	305	23	22	1
ABY	SkyWest Airlines Inc.	131	17	14	4
ACK	Endeavor Air Inc.	72	14	13	3
ACK	Envoy Air	4	0	0	0
ACK	JetBlue Airways	391	103	101	19
ACK	Republic Airline	129	18	21	5
ACT	Envoy Air	610	66	81	27

airport	airline	total_flights	dep_delay_count	arr_delay_count	cancellation_count
ACT	SkyWest Airlines Inc.	109	24	30	7
ACV	SkyWest Airlines Inc.	842	105	122	39
ACY	Spirit Air Lines	1354	200	218	46
ADK	Alaska Airlines Inc.	41	16	16	0
ADQ	Alaska Airlines Inc.	373	44	57	20
AEX	Endeavor Air Inc.	473	49	47	8
AEX	Envoy Air	391	48	66	22
AEX	ExpressJet Airlines LLC d/b/a aha!	139	16	33	4
AEX	PSA Airlines Inc.	33	1	1	0
AEX	SkyWest Airlines Inc.	152	21	22	0
AGS	American Airlines Inc.	7	3	3	0
AGS	Delta Air Lines Inc.	163	15	10	2
AGS	Endeavor Air Inc.	820	84	87	9
AGS	Envoy Air	181	33	31	4
AGS	Mesa Airlines Inc.	20	4	5	3
AGS	PSA Airlines Inc.	499	89	92	22
AGS	Republic Airline	6	1	1	0
AGS	SkyWest Airlines Inc.	138	31	23	1

9. Which months experienced the highest percentage of arrival delays?

In [113…

```
%%sql result11 <<
SELECT
    SUM(CASE WHEN flights.arr_delay >15 THEN 1 END) AS negative_delay_count,
    COUNT(airline.fl_number) AS total_flights,
    ROUND(SUM(CASE WHEN flights.arr_delay >15 THEN 1 END)::DECIMAL / COUNT(airline.fl_
    date.fl_month
FROM flights
JOIN date ON flights.arr_time_key = date.key
JOIN airline ON flights.airline_key = airline.key
GROUP BY date.fl_month
ORDER BY percentage_arrival_delays DESC;
```

```
 * postgresql://student@/Final_Airline_Dataset3
12 rows affected.
Returning data to local variable result11
```

In [114…

```
df11 = result11.DataFrame()
df11
```

Out[114]:

	negative_delay_count	total_flights	percentage_arrival_delays	fl_month
0	57493	255801	0.22	6
1	60666	279803	0.22	7
2	42323	209896	0.20	12
3	54062	281338	0.19	8
4	42935	241195	0.18	2
5	40744	236473	0.17	4
6	41491	248141	0.17	5
7	45520	278292	0.16	3
8	42909	261272	0.16	1
9	31757	214388	0.15	10
10	29216	209228	0.14	11
11	26173	204231	0.13	9

In [115…

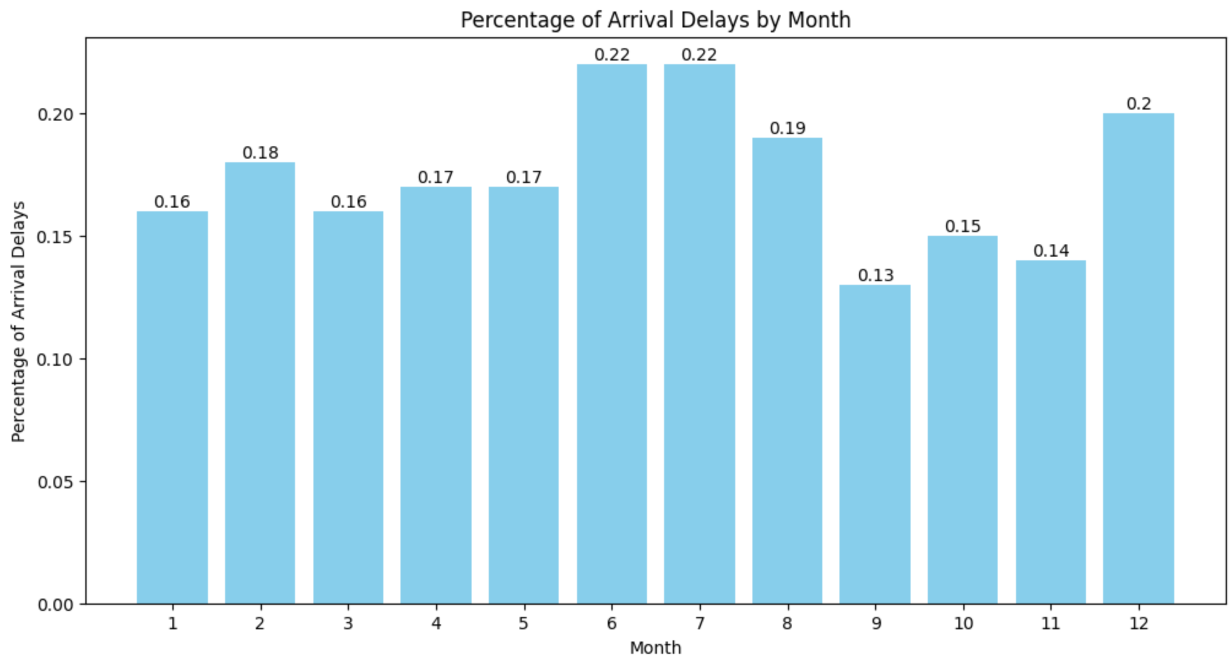
```
plt.figure(figsize=(12, 6))
bars = plt.bar(df11['fl_month'], df11['percentage_arrival_delays'], color='skyblue')

plt.xlabel('Month')
plt.ylabel('Percentage of Arrival Delays')
plt.title('Percentage of Arrival Delays by Month')

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), va='bottom', ha='c

plt.xticks(df11['fl_month'], [month for month in df11['fl_month']])

plt.show()
```



10. Which months experienced the highest percentage of departure delays?

In [116...

```
%%sql result12 <<
SELECT
    SUM(CASE WHEN flights.dep_delay >15 THEN 1 END) AS negative_delay_count,
    COUNT(airline.fl_number) AS total_flights,
    ROUND(SUM(CASE WHEN flights.dep_delay >15 THEN 1 END)::DECIMAL / COUNT(airline.fl_
    date.fl_month
FROM flights
JOIN date ON flights.dep_time_key = date.key
JOIN airline ON flights.airline_key = airline.key
GROUP BY date.fl_month
ORDER BY percentage_departure_delays DESC;
```

```
 * postgresql://student@/Final_Airline_Dataset3
12 rows affected.
Returning data to local variable result12
```

In [117...

```
df12 = result12.DataFrame()
df12
```

Out[117]:

	negative_delay_count	total_flights	percentage_departure_delays	fl_month
0	57968	255997	0.23	6
1	61133	280038	0.22	7
2	42315	210010	0.20	12
3	54353	281548	0.19	8
4	42046	248252	0.17	5
5	41038	241317	0.17	2
6	40812	236574	0.17	4
7	44637	278383	0.16	3
8	41611	261424	0.16	1
9	31980	214444	0.15	10
10	29146	209300	0.14	11
11	26394	204296	0.13	9

In [118…

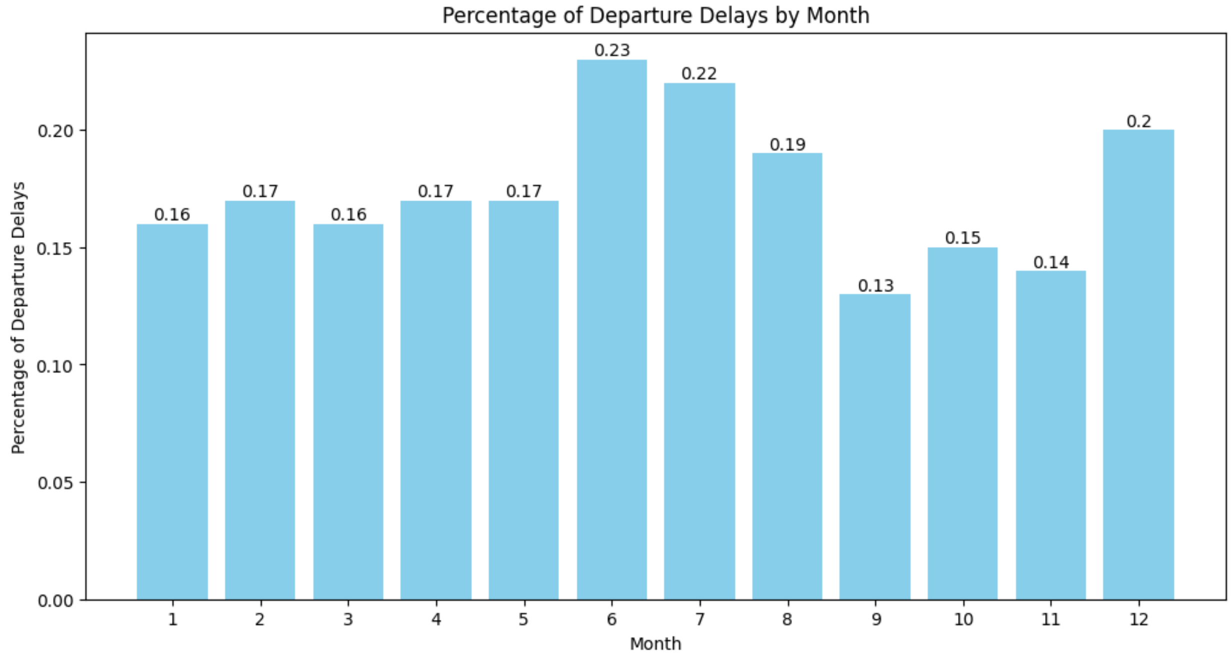
```
plt.figure(figsize=(12, 6))
bars = plt.bar(df12['fl_month'], df12['percentage_departure_delays'], color='skyblue')

plt.xlabel('Month')
plt.ylabel('Percentage of Departure Delays')
plt.title('Percentage of Departure Delays by Month')

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), va='bottom', ha='c

plt.xticks(df12['fl_month'], [month for month in df12['fl_month']])

plt.show()
```



In [119…

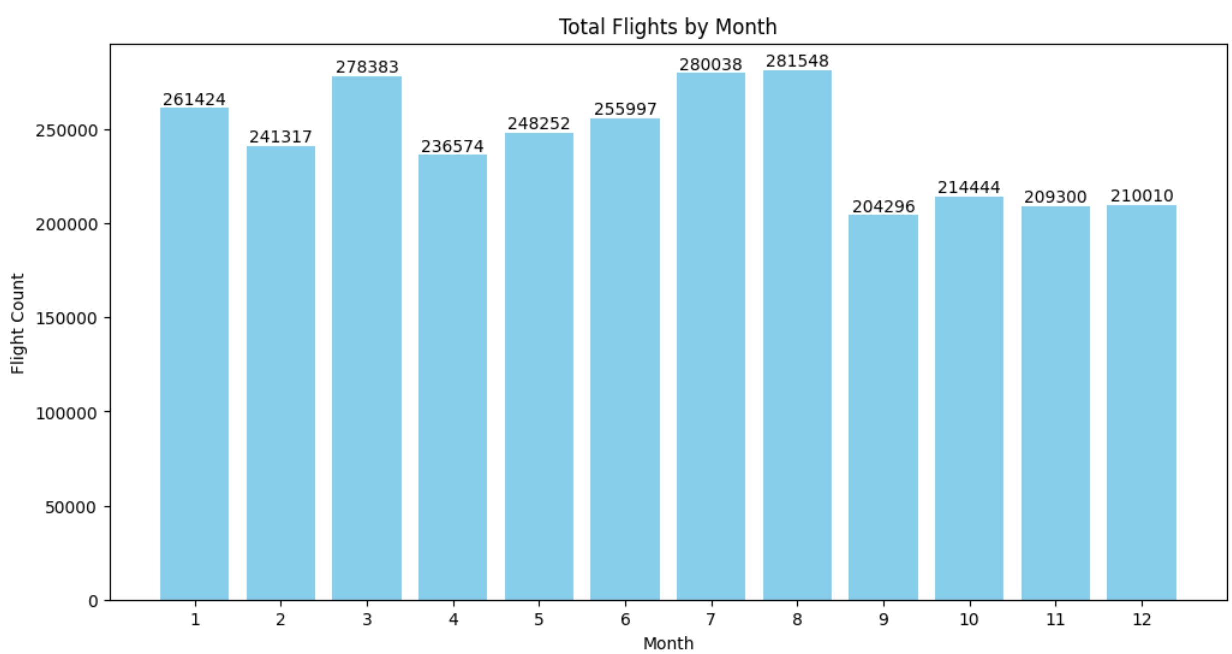
```
plt.figure(figsize=(12, 6))
bars = plt.bar(df12['fl_month'], df12['total_flights'], color='skyblue')

plt.xlabel('Month')
plt.ylabel('Flight Count')
plt.title('Total Flights by Month')

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), va='bottom', ha='c

plt.xticks(df12['fl_month'], [month for month in df12['fl_month']])

plt.show()
```



11. Which origin airports handle high volumes of traffic with relatively fewer departure delays?

In [120…

```
%%sql result13 <<
SELECT
    airport.airport,
    COUNT(airline.fl_number) AS total_flights,
    ROUND(SUM(CASE WHEN flights.dep_delay >15 THEN 1 END)::DECIMAL / COUNT(airline.fl_
FROM flights
JOIN airport ON flights.origin_airport_key = airport.key
JOIN airline ON flights.airline_key = airline.key
GROUP BY airport.airport
ORDER BY percentage_departure_delays ASC
LIMIT 10;
```

```
 * postgresql://student@/Final_Airline_Dataset3
10 rows affected.
Returning data to local variable result13
```

In [121…

```
df13 = result13.DataFrame()
df13
```

Out[121]:

	airport	total_flights	percentage_departure_delays
0	SPN	169	0.04
1	VCT	170	0.05
2	BTM	363	0.06
3	RKS	260	0.07
4	EKO	245	0.07
5	ALW	162	0.07
6	LWS	467	0.07
7	HIB	316	0.07
8	HLN	665	0.07
9	INL	299	0.07

In [122…

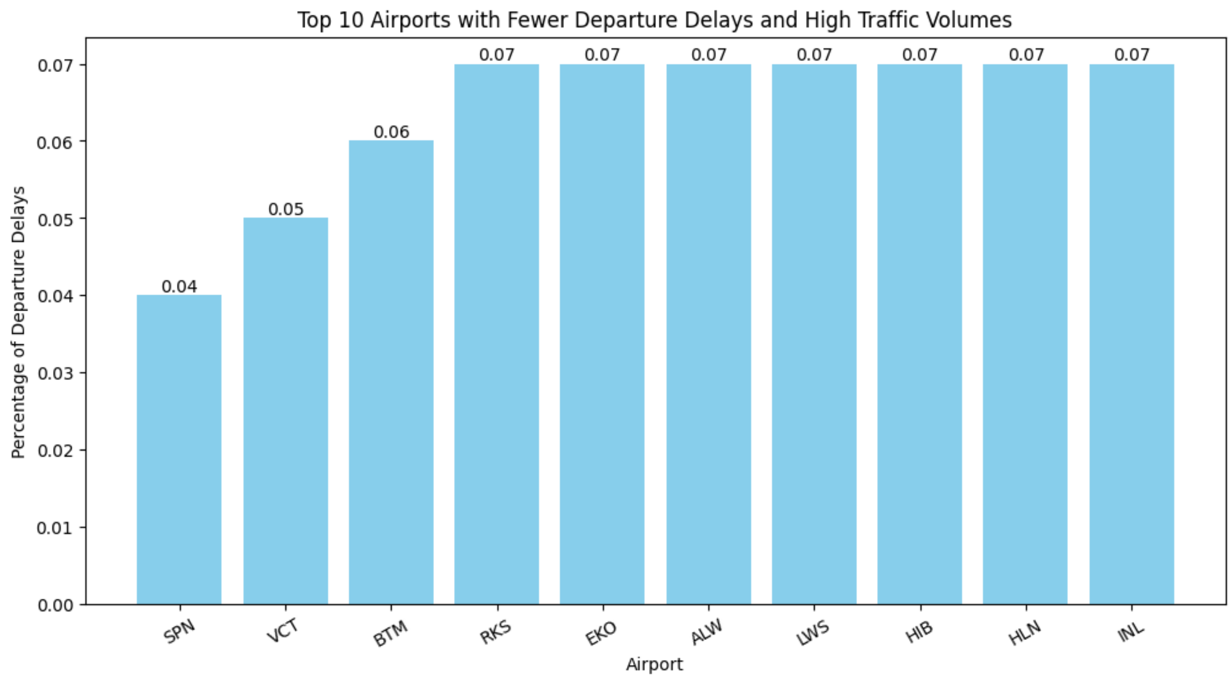
```
plt.figure(figsize=(12, 6))
bars = plt.bar(df13['airport'], df13['percentage_departure_delays'], color='skyblue')

plt.xlabel('Airport')
plt.ylabel('Percentage of Departure Delays')
plt.title('Top 10 Airports with Fewer Departure Delays and High Traffic Volumes')

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), va='bottom', ha='c

plt.xticks(rotation=30)

plt.show()
```



Additional short queries with different visualizations

a. How have average flight departure delays changed over time (monthly) in the last year?

In [123…

```
%%sql resulta <<
SELECT
    date.fl_month,
    date.fl_year,
    AVG(f.dep_delay) AS avg_departure_delay
FROM
    flights f
JOIN
    date ON f.dep_time_key = date.key
WHERE
    date.fl_year = 2022
GROUP BY
    date.fl_month, date.fl_year
ORDER BY
    date.fl_year, date.fl_month;
```

```
 * postgresql://student@/Final_Airline_Dataset3
12 rows affected.
Returning data to local variable resulta
```

In [124…

```
dfa = resulta.DataFrame()
dfa
```

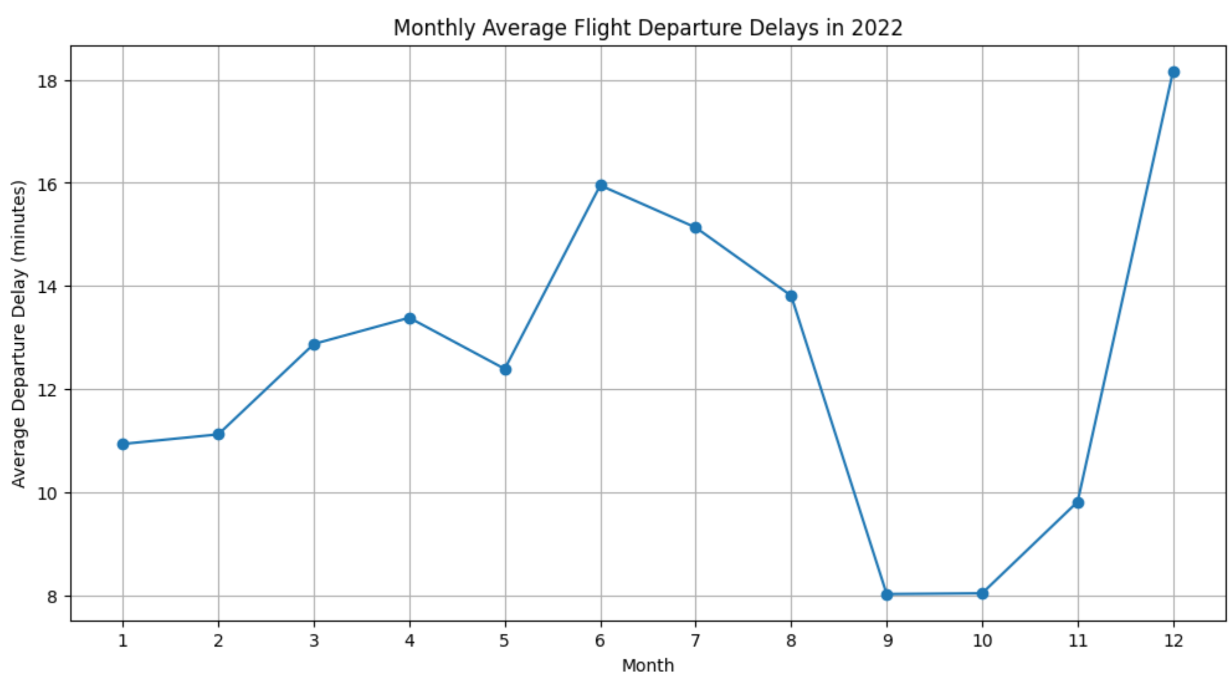
Out[124]:

	fl_month	fl_year	avg_departure_delay
0	1	2022	10.9341801161413117
1	2	2022	11.1202939233817701
2	3	2022	12.8736821725891899
3	4	2022	13.3831175855479032
4	5	2022	12.3907793822980723
5	6	2022	15.9517478130554491
6	7	2022	15.1341079560675060
7	8	2022	13.8107050496801416
8	9	2022	8.0218753337367840
9	10	2022	8.0366575984021488
10	11	2022	9.8119895429550167
11	12	2022	18.1681993473746663

In [125…

```
import matplotlib.pyplot as plt
import pandas as pd

plt.figure(figsize=(12, 6))
plt.plot(dfa['fl_month'], dfa['avg_departure_delay'], marker='o')
plt.xlabel('Month')
plt.ylabel('Average Departure Delay (minutes)')
plt.title('Monthly Average Flight Departure Delays in 2022')
plt.xticks(dfa['fl_month'])
plt.grid(True)
plt.show()
```



b. Which airlines have the best on-time performance (least delays)?

In [126…

```
%%sql resultb <<
SELECT
    a.airline,
    ROUND((SUM(CASE WHEN f.dep_delay <= 15 THEN 1 ELSE 0 END)::DECIMAL / COUNT(*)) * 1
FROM
    flights f
JOIN
    airline a ON f.airline_key = a.key
GROUP BY
    a.airline
ORDER BY
    on_time_performance DESC
LIMIT 10;
```

```
 * postgresql://student@/Final_Airline_Dataset3
10 rows affected.
Returning data to local variable resultb
```

In [127…

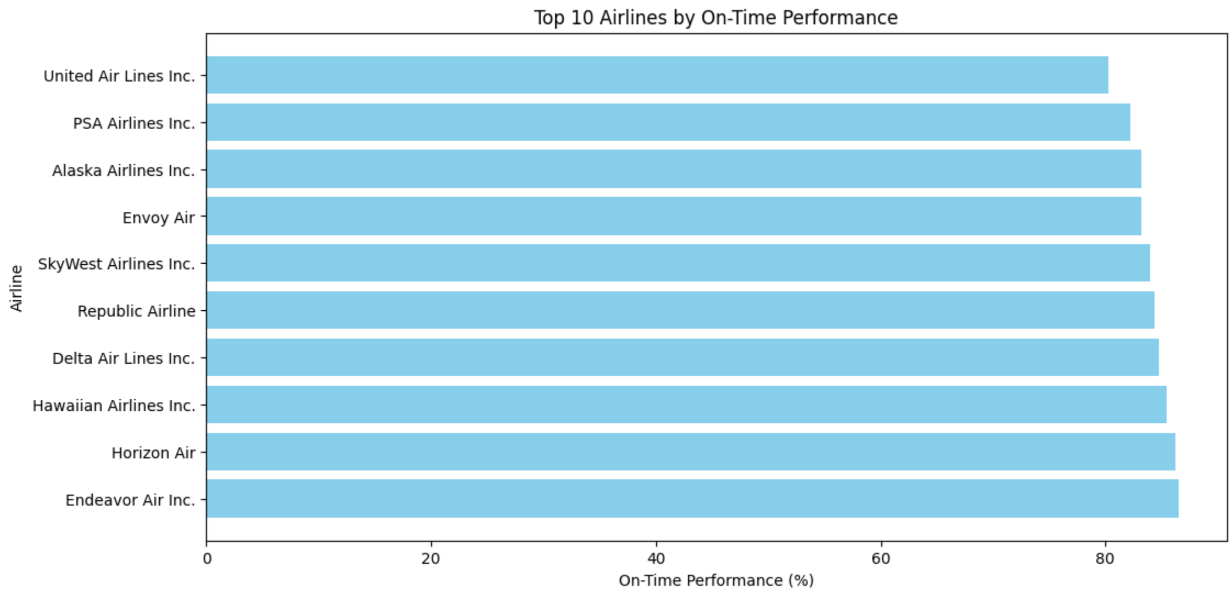
```
dfb = resultb.DataFrame()
dfb
```

Out[127]:

	airline	on_time_performance
0	Endeavor Air Inc.	86.50
1	Horizon Air	86.20
2	Hawaiian Airlines Inc.	85.48
3	Delta Air Lines Inc.	84.76
4	Republic Airline	84.32
5	SkyWest Airlines Inc.	83.93
6	Envoy Air	83.18
7	Alaska Airlines Inc.	83.16
8	PSA Airlines Inc.	82.17
9	United Air Lines Inc.	80.29

In [128…

```
plt.figure(figsize=(12, 6))
plt.barh(dfb['airline'], dfb['on_time_performance'], color='skyblue')
plt.xlabel('On-Time Performance (%)')
plt.ylabel('Airline')
plt.title('Top 10 Airlines by On-Time Performance')
plt.show()
```



d. What is the cancellation rate by month across all airlines?

In [129…

```
%%sql resultd <<
SELECT
    date.fl_month,
    ROUND((SUM(CASE WHEN f.cancelled = '1' THEN 1 ELSE 0 END)::DECIMAL / COUNT(*)) * 1
FROM
    flights f
JOIN
    date ON f.dep_time_key = date.key
GROUP BY
    date.fl_month
ORDER BY
    date.fl_month;
```

```
 * postgresql://student@/Final_Airline_Dataset3
12 rows affected.
Returning data to local variable resultd
```

In [130…

```
dfd = resultd.DataFrame()
dfd
```

Out[130]:

	fl_month	cancellation_rate
0	1	0.06
1	2	0.05
2	3	0.03
3	4	0.04
4	5	0.04
5	6	0.08
6	7	0.08
7	8	0.07
8	9	0.03
9	10	0.03
10	11	0.03
11	12	0.05

In [131…

```
plt.figure(figsize=(12, 6))
plt.fill_between(dfd['fl_month'], dfd['cancellation_rate'], color='skyblue', alpha=0.5
plt.xlabel('Month')
plt.ylabel('Cancellation Rate (%)')
plt.title('Monthly Cancellation Rates')
plt.xticks(dfd['fl_month'])
plt.show()
```