# Project Description: Data Analysis and Insights for Flight Delays and Airlines

In this project, we will perform a comprehensive analysis of flight data to understand the factors contributing to flight delays and make data-driven recommendations for travelers. The project consists of several tasks aimed at collecting, preprocessing, visualizing, and analyzing data related to flights, airports, runways, airlines, and passenger traffic. We will also perform hypothesis testing to investigate the relationships between various variables and flight delays.

```
import requests # for making standard html requests
from bs4 import BeautifulSoup # magical tool for parsing html data
import pandas as pd # for data manipulation
import numpy as np # for data manipulation
import seaborn as sns # for data visualization
import matplotlib.pyplot as plt # for data visualization
from sklearn.metrics import mean_absolute_percentage_error # for model
evaluation
import statsmodels.api as sm # for data exploration
import scipy.stats as stats # for statistical analysis
from sklearn.linear_model import SGDClassifier # for classification
from sklearn.model_selection import StratifiedKFold,
RandomizedSearchCV, train_test_split # for cross validation and
hyperparameter tuning
from statsmodels.formula.api import glm # for classification
from sklearn.preprocessing import OrdinalEncoder, StandardScaler # for
data preprocessing
from sklearn.pipeline import Pipeline # for data preprocessing
from sklearn.metrics import classification_report, accuracy_score #
for model evaluation
from sklearn.tree import DecisionTreeClassifier # for classification
from xgboost import XGBClassifier # for classification
```

Importing data and aggregating to create a fine tuned dataset for the project.

Collating information specific to flights that may cause delays for the final dataset.

```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

airlines=pd.read_excel("/content/drive/MyDrive/Simpli Learn
Dataset/Airlines.xlsx")
airports=pd.read_excel("/content/drive/MyDrive/Simpli Learn
Dataset/airports.xlsx")
```

```
runways=pd.read_excel("/content/drive/MyDrive/Simpli Learn
Dataset/runways.xlsx")
```

Lets look at the imported data.

```
airlines.head()

   id Airline  Flight AirportFrom AirportTo  DayOfWeek  Time  Length
Delay
0   1      CO     269         SFO       IAH          3    15     205
1
1   2      US    1558         PHX       CLT          3    15     222
1
2   3      AA    2400         LAX       DFW          3    20     165
1
3   4      AA    2466         SFO       DFW          3    20     195
1
4   5      AS     108         ANC       SEA          3    30     202
0

airports.head()

       id ident           type                               name  \
0    6523   00A       heliport                  Total Rf Heliport
1  323361  00AA  small_airport               Aero B Ranch Airport
2    6524  00AK  small_airport                       Lowell Field
3    6525  00AL  small_airport                       Epps Airpark
4    6526  00AR         closed  Newport Hospital & Clinic Heliport

   latitude_deg  longitude_deg  elevation_ft continent iso_country
iso_region  \
0     40.070801     -74.933601          11.0       NaN          US
US-PA
1     38.704022    -101.473911        3435.0       NaN          US
US-KS
2     59.947733    -151.692524         450.0       NaN          US
US-AK
3     34.864799     -86.770302         820.0       NaN          US
US-AL
4     35.608700     -91.254898         237.0       NaN          US
US-AR

   municipality scheduled_service gps_code iata_code local_code
home_link  \
0      Bensalem                no      00A       NaN        00A
NaN
1         Leoti                no     00AA       NaN       00AA
NaN
2  Anchor Point                no     00AK       NaN       00AK
NaN
```

```
3       Harvest                no     00AL       NaN       00AL
NaN
4       Newport                no      NaN       NaN        NaN
NaN

  wikipedia_link keywords
0            NaN      NaN
1            NaN      NaN
2            NaN      NaN
3            NaN      NaN
4            NaN     00AR
```

```
runways.head()
```

```
       id  airport_ref airport_ident  length_ft  width_ft surface
lighted  \
0  269408         6523           00A       80.0      80.0  ASPH-G
1
1  255155         6524          00AK     2500.0      70.0    GRVL
0
2  254165         6525          00AL     2300.0     200.0    TURF
0
3  270932         6526          00AR       40.0      40.0   GRASS
0
4  322128       322127          00AS     1450.0      60.0    Turf
0

   closed le_ident  le_latitude_deg  le_longitude_deg  le_elevation_ft
\
0       0       H1              NaN               NaN              NaN

1       0        N              NaN               NaN              NaN

2       0        1              NaN               NaN              NaN

3       0       H1              NaN               NaN              NaN

4       0        1              NaN               NaN              NaN


   le_heading_degT  le_displaced_threshold_ft he_ident
he_latitude_deg  \
0              NaN                        NaN      NaN
NaN
1              NaN                        NaN        S
NaN
2              NaN                        NaN       19
NaN
3              NaN                        NaN       H1
NaN
```

```
4                 NaN                        NaN        19
NaN

   he_longitude_deg  he_elevation_ft  he_heading_degT  \
0               NaN              NaN              NaN
1               NaN              NaN              NaN
2               NaN              NaN              NaN
3               NaN              NaN              NaN
4               NaN              NaN              NaN

   he_displaced_threshold_ft
0                        NaN
1                        NaN
2                        NaN
3                        NaN
4                        NaN
```

Let's **merge** airports and runways and create airport_run using the **ident column in airports**, and the **airport_ident column in runways**. We are doing a left join i.e. all of airports columns and only the matching values from runways will be imported.

```
airport_run = pd.merge(airports, runways, left_on='ident',
right_on='airport_ident', how='left')
airport_run.head()

     id_x ident           type                              name  \
0    6523   00A       heliport                 Total Rf Heliport
1  323361  00AA  small_airport              Aero B Ranch Airport
2    6524  00AK  small_airport                      Lowell Field
3    6525  00AL  small_airport                      Epps Airpark
4    6526  00AR         closed  Newport Hospital & Clinic Heliport

   latitude_deg  longitude_deg  elevation_ft continent iso_country
iso_region  \
0     40.070801     -74.933601          11.0       NaN          US
US-PA
1     38.704022    -101.473911        3435.0       NaN          US
US-KS
2     59.947733    -151.692524         450.0       NaN          US
US-AK
3     34.864799     -86.770302         820.0       NaN          US
US-AL
4     35.608700     -91.254898         237.0       NaN          US
US-AR

   ... le_longitude_deg le_elevation_ft le_heading_degT  \
0  ...              NaN             NaN             NaN
1  ...              NaN             NaN             NaN
2  ...              NaN             NaN             NaN
```

```
3  ...                NaN              NaN              NaN
4  ...                NaN              NaN              NaN

  le_displaced_threshold_ft he_ident he_latitude_deg he_longitude_deg
\
0                       NaN      NaN             NaN              NaN

1                       NaN      NaN             NaN              NaN

2                       NaN        S             NaN              NaN

3                       NaN       19             NaN              NaN

4                       NaN       H1             NaN              NaN


  he_elevation_ft  he_heading_degT  he_displaced_threshold_ft
0             NaN              NaN                        NaN
1             NaN              NaN                        NaN
2             NaN              NaN                        NaN
3             NaN              NaN                        NaN
4             NaN              NaN                        NaN

[5 rows x 38 columns]
```

```
airport_run.columns
```

```
Index(['id_x', 'ident', 'type', 'name', 'latitude_deg',
'longitude_deg',
       'elevation_ft', 'continent', 'iso_country', 'iso_region',
       'municipality', 'scheduled_service', 'gps_code', 'iata_code',
       'local_code', 'home_link', 'wikipedia_link', 'keywords',
'id_y',
       'airport_ref', 'airport_ident', 'length_ft', 'width_ft',
'surface',
       'lighted', 'closed', 'le_ident', 'le_latitude_deg',
'le_longitude_deg',
       'le_elevation_ft', 'le_heading_degT',
'le_displaced_threshold_ft',
       'he_ident', 'he_latitude_deg', 'he_longitude_deg',
'he_elevation_ft',
       'he_heading_degT', 'he_displaced_threshold_ft'],
      dtype='object')
```

Lets count the number of runways each airport has. Here we will **group the rows** based on the **common airport_ident** values while keep a **count of the non-null values in id_y** which has the runway entries of a particular airport.

```
count_runway = airport_run.groupby('airport_ident')
[['id_y']].count().sort_values(by='id_y', ascending =
```

```
False).reset_index()
count_runway.head()

  airport_ident  id_y
0          KORD    11
1          KNHU    10
2           JRA     9
3          TA12     8
4           SXS     8
```

Now that we have a count of runways for each airport, let's create a **new dataset containing airport iata code, type of airport, elevation of airport, and the number of runways**. All of which could play a role in delays.

```
air_run = pd.merge(airports, count_runway, left_on = 'ident', right_on
= 'airport_ident', how = 'left')
air_run.rename(columns = {'id_y':'runway_count'}, inplace = True)
air_run.head()

       id ident           type                                name  \
0    6523   00A       heliport                   Total Rf Heliport
1  323361  00AA  small_airport                Aero B Ranch Airport
2    6524  00AK  small_airport                        Lowell Field
3    6525  00AL  small_airport                        Epps Airpark
4    6526  00AR         closed  Newport Hospital & Clinic Heliport

   latitude_deg  longitude_deg  elevation_ft continent iso_country
iso_region  \
0     40.070801     -74.933601          11.0       NaN          US
US-PA
1     38.704022    -101.473911        3435.0       NaN          US
US-KS
2     59.947733    -151.692524         450.0       NaN          US
US-AK
3     34.864799     -86.770302         820.0       NaN          US
US-AL
4     35.608700     -91.254898         237.0       NaN          US
US-AR

   municipality scheduled_service gps_code iata_code local_code
home_link  \
0      Bensalem                no      00A       NaN        00A
NaN
1         Leoti                no     00AA       NaN       00AA
NaN
2  Anchor Point                no     00AK       NaN       00AK
NaN
3       Harvest                no     00AL       NaN       00AL
NaN
4       Newport                no      NaN       NaN        NaN
```

```
NaN
```

```
  wikipedia_link keywords airport_ident  runway_count
0            NaN      NaN           00A           1.0
1            NaN      NaN           NaN           NaN
2            NaN      NaN          00AK           1.0
3            NaN      NaN          00AL           1.0
4            NaN     00AR          00AR           1.0
```

```
air_run = air_run[['iata_code','type','elevation_ft','runway_count']]
```

```
air_run.head()
```

```
  iata_code           type  elevation_ft  runway_count
0       NaN       heliport          11.0           1.0
1       NaN  small_airport        3435.0           NaN
2       NaN  small_airport         450.0           1.0
3       NaN  small_airport         820.0           1.0
4       NaN         closed         237.0           1.0
```

```
air_run.shape
```

```
(73805, 4)
```

```
air_run.isnull().sum()
```

```
iata_code       64645
type                0
elevation_ft    14122
runway_count    36540
dtype: int64
```

```
# Removing null values and saving the rest as an excel file.
air_run.dropna().to_excel('air_run.xlsx', index = False)
```

```
airlines.head()
```

```
   id Airline  Flight AirportFrom AirportTo  DayOfWeek  Time  Length
Delay
0   1      CO     269         SFO       IAH          3    15     205
1
1   2      US    1558         PHX       CLT          3    15     222
1
2   3      AA    2400         LAX       DFW          3    20     165
1
3   4      AA    2466         SFO       DFW          3    20     195
1
4   5      AS     108         ANC       SEA          3    30     202
0
```

Lets **add info about the AirportFrom** values by combining the airlines and air_run datasets based on the AirportFrom and iata_code columns, this would give us the **count of runways,**

**elevation, and iata_code of the airports from where flights take off**. Something that could be a factor in delays.

```
combined_data = pd.merge(airlines, air_run, how='left',
left_on='AirportFrom', right_on = 'iata_code')

new_names = list(combined_data[air_run.columns].columns +
'_source_airport')
old_names = list(combined_data[air_run.columns].columns)

combined_data.rename(columns = {old:new for old, new in zip(old_names,
new_names)}, inplace = True)
combined_data.head()
```

```
   id Airline  Flight AirportFrom AirportTo  DayOfWeek  Time  Length
Delay  \
0   1      CO     269         SFO       IAH          3    15     205
1
1   2      US    1558         PHX       CLT          3    15     222
1
2   3      AA    2400         LAX       DFW          3    20     165
1
3   4      AA    2466         SFO       DFW          3    20     195
1
4   5      AS     108         ANC       SEA          3    30     202
0

  iata_code_source_airport type_source_airport
elevation_ft_source_airport  \
0                       SFO       large_airport
13.0
1                       PHX       large_airport
1135.0
2                       LAX       large_airport
125.0
3                       SFO       large_airport
13.0
4                       ANC       large_airport
152.0

   runway_count_source_airport
0                          4.0
1                          3.0
2                          4.0
3                          4.0
4                          3.0
```

```
combined_data.columns
```

```
Index(['id', 'Airline', 'Flight', 'AirportFrom', 'AirportTo',
'DayOfWeek',
```

```
       'Time', 'Length', 'Delay', 'iata_code_source_airport',
       'type_source_airport', 'elevation_ft_source_airport',
       'runway_count_source_airport'],
      dtype='object')
```

Let's **add similar info about the destination airports**. This would also be a factor in flights delays. For eg: if the destination airport has only one runway, flights would have to park themselves and wait for their turn to land resulting in delays.

But this time we will use combined_data instead of airlines since we have all the info from airlines dataset from previous steps.

```
combined_data = pd.merge(combined_data, air_run, how='left', left_on =
'AirportTo', right_on='iata_code')

new_names = list(combined_data[air_run.columns].columns +
'_dest_airport')
old_names = list(combined_data[air_run.columns].columns)

combined_data.rename(columns = {old:new for old,new in zip(old_names,
new_names)}, inplace = True)
combined_data.head()
```

```
   id Airline  Flight AirportFrom AirportTo  DayOfWeek  Time  Length
Delay  \
0   1      CO     269         SFO       IAH          3    15     205
1
1   2      US    1558         PHX       CLT          3    15     222
1
2   3      AA    2400         LAX       DFW          3    20     165
1
3   4      AA    2466         SFO       DFW          3    20     195
1
4   5      AS     108         ANC       SEA          3    30     202
0

  iata_code_source_airport type_source_airport
elevation_ft_source_airport  \
0                      SFO       large_airport
13.0
1                      PHX       large_airport
1135.0
2                      LAX       large_airport
125.0
3                      SFO       large_airport
13.0
4                      ANC       large_airport
152.0

   runway_count_source_airport iata_code_dest_airport
```

```
type_dest_airport  \
0                           4.0                     IAH
large_airport
1                           3.0                     CLT
large_airport
2                           4.0                     DFW
large_airport
3                           4.0                     DFW
large_airport
4                           3.0                     SEA
large_airport

   elevation_ft_dest_airport  runway_count_dest_airport
0                       97.0                        5.0
1                      748.0                        4.0
2                      607.0                        7.0
3                      607.0                        7.0
4                      433.0                        4.0

# We don't need iata_code columns now, lets drop them.
combined_data.drop(columns =
list(combined_data.columns[combined_data.columns.str.startswith('iata_
code')]), inplace = True)

combined_data.head()

   id Airline  Flight AirportFrom AirportTo  DayOfWeek  Time  Length
Delay  \
0   1      CO     269         SFO       IAH          3    15     205
1
1   2      US    1558         PHX       CLT          3    15     222
1
2   3      AA    2400         LAX       DFW          3    20     165
1
3   4      AA    2466         SFO       DFW          3    20     195
1
4   5      AS     108         ANC       SEA          3    30     202
0

  type_source_airport  elevation_ft_source_airport  \
0       large_airport                         13.0
1       large_airport                       1135.0
2       large_airport                        125.0
3       large_airport                         13.0
4       large_airport                        152.0

   runway_count_source_airport type_dest_airport
elevation_ft_dest_airport  \
0                           4.0     large_airport
97.0
```

```
1                                 3.0     large_airport
748.0
2                                 4.0     large_airport
607.0
3                                 4.0     large_airport
607.0
4                                 3.0     large_airport
433.0

   runway_count_dest_airport
0                        5.0
1                        4.0
2                        7.0
3                        7.0
4                        4.0
```

As is with every profession, the amount of experience we have plays an important role in our performance. Applying that logic to airline delays, let's **extract experience info about each airline in our dataset** from "https://en.wikipedia.org/wiki/List_of_airlines_of_the_United_States" using Beautiful Soup class from bs4 and requests lib.

```
# Extracting the number of tables in the URL.

airexp_url =
requests.get('https://en.wikipedia.org/wiki/List_of_airlines_of_the_United_States').text
soup = BeautifulSoup(airexp_url, 'lxml')
tables_found = soup.findAll("table", {"class":"wikitable"})

len(tables_found)

7

airlines_wiki_list = []
for tab in tables_found:
    temp = pd.read_html(str(tab))
    temp = pd.DataFrame(temp[0])
    airlines_wiki_list.append(temp)

airlines_wiki = pd.concat(airlines_wiki_list)
```

Now that we have extracted all the relevant information from the wiki URL, its time to add that into our master dataset i.e. combined_data.

```
# Let's start by idetifying the founding year of the airlines in
combined_data.

airlines_founded =
pd.merge(combined_data[['Airline']].drop_duplicates(),
```

```
airlines_wiki[['IATA', 'Founded']].drop_duplicates(), how='left',
left_on= 'Airline', right_on='IATA')

airlines_founded
```

```
   Airline IATA  Founded
0       CO  NaN      NaN
1       US  NaN      NaN
2       AA   AA   1926.0
3       AS   AS   1932.0
4       DL   DL   1924.0
5       B6   B6   1998.0
6       HA   HA   1929.0
7       OO   OO   1972.0
8       9E   9E   1985.0
9       OH   OH   1979.0
10      EV  NaN      NaN
11      XE   XE   2016.0
12      YV   YV   1980.0
13      UA   UA   1926.0
14      MQ   MQ   1984.0
15      F9   F9   1994.0
16      WN   WN   1967.0
```

Apart from the experience, the traffic of the airport is also a factor in delays. If your source/destination airport sees a lot of traffic, naturally the take-off and landing times will be affected.

For this step, **we will extract info from** https://en.wikipedia.org/wiki/List_of_the_busiest_airports_in_the_United_States

But this wikipedia page was updated recently, so I am **using a previous version of the wikipedia page of 13 April 2023** from this URL: https://en.wikipedia.org/w/index.php?title=List_of_the_busiest_airports_in_the_United_States&oldid=1149689977

```
traffic_url = requests.get('https://en.wikipedia.org/w/index.php?
title=List_of_the_busiest_airports_in_the_United_States&oldid=11496899
77').text
soup = BeautifulSoup(traffic_url,'lxml')
traffic_tables = soup.findAll("table", {"class":"wikitable"})

hub_data = {}
i=0
for tab in traffic_tables:
    hub_data[i] = pd.read_html(str(tab))
    hub_data[i] = pd.DataFrame(hub_data[i][0])
    i+=1

large_hub = hub_data[0].copy()
med_hub = hub_data[1].copy()
```

```
large_hub.insert(loc =1, column = 'Hub_type', value = 'large')
med_hub.insert(loc =1, column = 'Hub_type', value = 'medium')

large_hub.head()
```

```
   Rank (2021) Hub_type                              Airports (large
hubs)  \
0            1    large  Hartsfield–Jackson Atlanta International
Airport
1            2    large           Dallas/Fort Worth International
Airport
2            3    large                      Denver International
Airport
3            4    large                      O'Hare International
Airport
4            5    large                 Los Angeles International
Airport

  IATA Code  Major cities served State   2021[3]   2020[4]
2019[5]  \
0       ATL              Atlanta    GA  36676010  20559866  53505795

1       DFW  Dallas & Fort Worth    TX  30005266  18593421  35778573

2       DEN               Denver    CO  28645527  16243216  33592945

3       ORD              Chicago    IL  26350976  14606034  40871223

4       LAX          Los Angeles    CA  23663410  14055777  42939104


    2018[6]   2017[7]   2016[8]   2015[9]  2014[10]  2013[11]
2012[12]
0  51865797  50251964  50501858  49340732  46604273  45308407
45798928
1  32821799  31816933  31283579  31589839  30804567  29038128
28022904
2  31362941  29809097  28267394  26280043  26000591  25496885
25799841
3  39873927  38593028  37589899  36305668  33843426  32317835
32171795
4  42624050  41232432  39636042  36351272  34314197  32425892
31326268
```

```
med_hub.head()
```

```
   Rank (2021) Hub_type                    Airports (medium hubs) IATA
Code  \
0           31   medium                         Dallas Love Field
DAL
```

```
1           32   medium   Daniel K. Inouye International Airport
HNL
2           33   medium          Portland International Airport
PDX
3           34   medium                William P. Hobby Airport
HOU
4           35   medium  Southwest Florida International Airport
RSW

  City served State  2021[3]  2020[4]  2019[5]  2018[6]  2017[7]
2016[8]  \
0      Dallas    TX  6487563  3669930  8408457  8134848  7876769
7554596
1    Honolulu    HI  5830928  3126391  9988678  9578505  9743989
9656340
2    Portland    OR  5759879  3455877  9797408  9940866  9435473
9071154
3     Houston    TX  5560780  3127178  7069614  6937061  6741870
6285181
4  Fort Myers    FL  5080805  2947139  5144467  4719568  4461304
4350650

   2015[9]  2014[10]  2013[11]  2012[12]
0  7040921   4522341   4023779   3902628
1  9656340   9463000   9466995   9225848
2  8340234   7878760   7452603   7142620
3  5937944   5800726   5377050   5043737
4  4231134   4025959   3788870   3634152

# Let's clean the column names from special characters or things in
bracket.

column_temp =
large_hub.columns.str.split('[([]').str[0].str.strip().str.lower().str
.replace(' ','_').values
column_temp[list(map( lambda x:x.isnumeric(), column_temp))] = 'data_'
+ column_temp[list(map( lambda x:x.isnumeric(), column_temp))]
large_hub.columns = column_temp
large_hub.columns

Index(['rank', 'hub_type', 'airports', 'iata_code',
'major_cities_served',
       'state', 'data_2021', 'data_2020', 'data_2019', 'data_2018',
       'data_2017', 'data_2016', 'data_2015', 'data_2014',
'data_2013',
       'data_2012'],
      dtype='object')

column_temp =
med_hub.columns.str.split('[([]').str[0].str.strip().str.lower().str.r
```

```
eplace(' ','_').values
column_temp[list(map( lambda x : x.isnumeric(), column_temp))]  =
'data_' + column_temp[list(map( lambda x : x.isnumeric(),
column_temp))]
med_hub.columns = column_temp
med_hub.columns
```

```
Index(['rank', 'hub_type', 'airports', 'iata_code', 'city_served',
'state',
       'data_2021', 'data_2020', 'data_2019', 'data_2018',
'data_2017',
       'data_2016', 'data_2015', 'data_2014', 'data_2013',
'data_2012'],
      dtype='object')
```

```
# Renaming 'major cities served' to 'city served'.

large_hub.rename(columns = {'major_cities_served' : 'ciity_served'},
inplace = True)

# Collating the info about large and medium hubs into one dataframe.

final_hub_data = pd.concat([large_hub, med_hub])
final_hub_data.head()
```

```
   rank hub_type                                          airports
iata_code  \
0     1    large  Hartsfield–Jackson Atlanta International Airport
ATL
1     2    large            Dallas/Fort Worth International Airport
DFW
2     3    large                       Denver International Airport
DEN
3     4    large                       O'Hare International Airport
ORD
4     5    large                  Los Angeles International Airport
LAX

          ciity_served state  data_2021  data_2020  data_2019
data_2018  \
0              Atlanta    GA   36676010   20559866   53505795
51865797
1  Dallas & Fort Worth    TX   30005266   18593421   35778573
32821799
2               Denver    CO   28645527   16243216   33592945
31362941
3              Chicago    IL   26350976   14606034   40871223
39873927
4          Los Angeles    CA   23663410   14055777   42939104
42624050
```

```
   data_2017  data_2016  data_2015  data_2014  data_2013  data_2012  \
0   50251964   50501858   49340732   46604273   45308407   45798928
1   31816933   31283579   31589839   30804567   29038128   28022904
2   29809097   28267394   26280043   26000591   25496885   25799841
3   38593028   37589899   36305668   33843426   32317835   32171795
4   41232432   39636042   36351272   34314197   32425892   31326268

  city_served
0         NaN
1         NaN
2         NaN
3         NaN
4         NaN
```

```
final_hub_data.data_2021.isnull().sum()
```

```
0
```

```
# Let's add traffic info in our master dataset i.e. combined_data and
create a new dataset combined_data_traffic.

combined_data_traffic = pd.merge(combined_data,
final_hub_data[['iata_code','data_2021']], how='left',
left_on='AirportFrom', right_on='iata_code')

# Renaming the iata_code and data_2021 columns to reflect the source
airport.
combined_data_traffic.rename(columns =
{'iata_code':'iata_code_source',
'data_2021':'data_2021_source_airport'}, inplace=True)

combined_data_traffic.head()
```

```
   id Airline  Flight AirportFrom AirportTo  DayOfWeek  Time  Length
Delay  \
0   1      CO     269         SFO       IAH          3    15     205
1
1   2      US    1558         PHX       CLT          3    15     222
1
2   3      AA    2400         LAX       DFW          3    20     165
1
3   4      AA    2466         SFO       DFW          3    20     195
1
4   5      AS     108         ANC       SEA          3    30     202
0

  type_source_airport  elevation_ft_source_airport  \
0       large_airport                         13.0
1       large_airport                       1135.0
2       large_airport                        125.0
3       large_airport                         13.0
```

```
4       large_airport                               152.0

   runway_count_source_airport type_dest_airport
elevation_ft_dest_airport  \
0                          4.0     large_airport
97.0
1                          3.0     large_airport
748.0
2                          4.0     large_airport
607.0
3                          4.0     large_airport
607.0
4                          3.0     large_airport
433.0

   runway_count_dest_airport iata_code_source
data_2021_source_airport
0                        5.0              SFO
11725347.0
1                        4.0              PHX
18940287.0
2                        7.0              LAX
23663410.0
3                        7.0              SFO
11725347.0
4                        4.0              ANC
2184959.0

# Merging traffic info for destination airports.

combined_data_traffic = pd.merge(combined_data_traffic,
final_hub_data[['iata_code', 'data_2021']], how = 'left',
left_on='AirportTo', right_on='iata_code')

# Renaming the added columns to reflect the destination airport.

combined_data_traffic.rename(columns = {'iata_code' :
'iata_code_dest','data_2021':'data_2021_dest_airport'}, inplace =
True)

combined_data_traffic

            id Airline  Flight AirportFrom AirportTo  DayOfWeek  Time
Length  \
0            1      CO     269         SFO       IAH          3    15
205
1            2      US    1558         PHX       CLT          3    15
222
2            3      AA    2400         LAX       DFW          3    20
165
3            4      AA    2466         SFO       DFW          3    20
```

```
195
4              5      AS     108         ANC       SEA          3    30
202
...          ...     ...     ...         ...       ...        ...   ...
...
518551  539377      B6     717         JFK       SJU          5  1439
220
518552  539378      B6     739         JFK       PSE          5  1439
223
518553  539379      CO     178         OGG       SNA          5  1439
326
518554  539382      UA      78         HNL       SFO          5  1439
313
518555  539383      US    1442         LAX       PHL          5  1439
301

        Delay type_source_airport  elevation_ft_source_airport  \
0           1       large_airport                         13.0
1           1       large_airport                       1135.0
2           1       large_airport                        125.0
3           1       large_airport                         13.0
4           0       large_airport                        152.0
...       ...                 ...                          ...
518551      1       large_airport                         13.0
518552      1       large_airport                         13.0
518553      0      medium_airport                         54.0
518554      1       large_airport                         13.0
518555      1       large_airport                        125.0

        runway_count_source_airport type_dest_airport  \
0                               4.0     large_airport
1                               3.0     large_airport
2                               4.0     large_airport
3                               4.0     large_airport
4                               3.0     large_airport
...                             ...               ...
518551                          4.0     large_airport
518552                          4.0    medium_airport
518553                          2.0     large_airport
518554                          6.0     large_airport
518555                          4.0     large_airport

        elevation_ft_dest_airport  runway_count_dest_airport
iata_code_source  \
0                            97.0                        5.0
SFO
1                           748.0                        4.0
PHX
2                           607.0                        7.0
LAX
```

```
3                                607.0                             7.0
SFO
4                                433.0                             4.0
ANC
...                                ...                             ...
...
518551                             9.0                             2.0
JFK
518552                            29.0                             1.0
JFK
518553                            56.0                             2.0
OGG
518554                            13.0                             4.0
HNL
518555                            36.0                             4.0
LAX

        data_2021_source_airport iata_code_dest
data_2021_dest_airport
0                      11725347.0            IAH
16242821.0
1                      18940287.0            CLT
20900875.0
2                      23663410.0            DFW
30005266.0
3                      11725347.0            DFW
30005266.0
4                       2184959.0            SEA
17430195.0
...                           ...            ...                     ..
.
518551                 15273342.0            SJU
4738725.0
518552                 15273342.0            NaN
NaN
518553                  2933315.0            SNA
3807205.0
518554                  5830928.0            SFO
11725347.0
518555                 23663410.0            PHL
9820222.0

[518556 rows x 19 columns]
```

Adding the founding year of respective airlines.

```
airlines_founded
```

```
   Airline IATA  Founded
0       CO  NaN      NaN
1       US  NaN      NaN
2       AA   AA   1926.0
3       AS   AS   1932.0
4       DL   DL   1924.0
5       B6   B6   1998.0
6       HA   HA   1929.0
7       OO   OO   1972.0
8       9E   9E   1985.0
9       OH   OH   1979.0
10      EV  NaN      NaN
11      XE   XE   2016.0
12      YV   YV   1980.0
13      UA   UA   1926.0
14      MQ   MQ   1984.0
15      F9   F9   1994.0
16      WN   WN   1967.0
```

```
# Merging it with combined_data_traffic using 'Airline' column.

combined_data_traffic = pd.merge(combined_data_traffic,
airlines_founded[['Airline', 'Founded']], how='left',
left_on='Airline', right_on='Airline')

combined_data_traffic.head()
```

```
   id Airline  Flight AirportFrom AirportTo  DayOfWeek  Time  Length
Delay  \
0   1      CO     269         SFO       IAH          3    15     205
1
1   2      US    1558         PHX       CLT          3    15     222
1
2   3      AA    2400         LAX       DFW          3    20     165
1
3   4      AA    2466         SFO       DFW          3    20     195
1
4   5      AS     108         ANC       SEA          3    30     202
0

  type_source_airport  elevation_ft_source_airport  \
0       large_airport                         13.0
1       large_airport                       1135.0
2       large_airport                        125.0
3       large_airport                         13.0
4       large_airport                        152.0

   runway_count_source_airport type_dest_airport
elevation_ft_dest_airport  \
0                          4.0     large_airport
```

```
97.0
1                           3.0     large_airport
748.0
2                           4.0     large_airport
607.0
3                           4.0     large_airport
607.0
4                           3.0     large_airport
433.0

   runway_count_dest_airport iata_code_source
data_2021_source_airport  \
0                         5.0              SFO
11725347.0
1                         4.0              PHX
18940287.0
2                         7.0              LAX
23663410.0
3                         7.0              SFO
11725347.0
4                         4.0              ANC
2184959.0

  iata_code_dest  data_2021_dest_airport  Founded
0            IAH              16242821.0      NaN
1            CLT              20900875.0      NaN
2            DFW              30005266.0   1926.0
3            DFW              30005266.0   1926.0
4            SEA              17430195.0   1932.0
```

Missing value treatment for the final dataset.

```
combined_data_traffic.isnull().sum()

id                                0
Airline                           0
Flight                            0
AirportFrom                       0
AirportTo                         0
DayOfWeek                         0
Time                              0
Length                            0
Delay                             0
type_source_airport              31
elevation_ft_source_airport      31
runway_count_source_airport      31
type_dest_airport                31
elevation_ft_dest_airport        31
runway_count_dest_airport        31
```

```
iata_code_source                   83582
data_2021_source_airport           83582
iata_code_dest                     83531
data_2021_dest_airport             83531
Founded                            83601
dtype: int64
```

```
# Let's start with type of airport.

# For source airport.
combined_data_traffic[combined_data_traffic.type_source_airport.isnull
()].AirportFrom.unique()
```

```
array(['CYS'], dtype=object)
```

```
# For destination airport.
combined_data_traffic[combined_data_traffic.type_dest_airport.isnull()
].AirportTo.unique()
```

```
array(['CYS'], dtype=object)
```

```
# Lets see what the data dictionary has to say about CYS.

airport_dic=pd.read_excel("/content/drive/MyDrive/Simpli Learn
Dataset/Data Dictionary.xlsx", sheet_name = 'airlines')

airport_dic.head()
```

```
        Column                               Description Unnamed: 2
0      Airline  Different types of commercial airlines        NaN
1       Flight                         Types of Aircraft        NaN
2  AirportFrom                            Source Airport        NaN
3    AirportTo                       Destination Airport        NaN
4    DayOfWeek              Tells you about the day of week        NaN
```

```
# Checking for the Description column for 'CYS'

row = airport_dic[airport_dic['Column'] ==
'CYS'].Description.values.astype(str)
print(row)
```

```
['Cheyenne Regional Jerry Olson Field']
```

```
airports.head()
```

```
       id ident           type                                name  \
0    6523   00A       heliport                   Total Rf Heliport
1  323361  00AA  small_airport                Aero B Ranch Airport
2    6524  00AK  small_airport                        Lowell Field
3    6525  00AL  small_airport                        Epps Airpark
4    6526  00AR         closed  Newport Hospital & Clinic Heliport
```

```
   latitude_deg  longitude_deg  elevation_ft continent iso_country
iso_region  \
0     40.070801     -74.933601          11.0       NaN          US
US-PA
1     38.704022    -101.473911        3435.0       NaN          US
US-KS
2     59.947733    -151.692524         450.0       NaN          US
US-AK
3     34.864799     -86.770302         820.0       NaN          US
US-AL
4     35.608700     -91.254898         237.0       NaN          US
US-AR

   municipality scheduled_service gps_code iata_code local_code
home_link  \
0      Bensalem                no      00A       NaN        00A
NaN
1         Leoti                no     00AA       NaN       00AA
NaN
2  Anchor Point                no     00AK       NaN       00AK
NaN
3       Harvest                no     00AL       NaN       00AL
NaN
4       Newport                no      NaN       NaN        NaN
NaN

  wikipedia_link keywords
0            NaN      NaN
1            NaN      NaN
2            NaN      NaN
3            NaN      NaN
4            NaN     00AR

# Extracting the type of airport and elevantion ft for 'CYS' from
airports dataset.

nan_airport = airports.loc[airports.name.str.lower() ==
row[0].lower(), ['ident', 'name', 'iata_code', 'type',
'elevation_ft']]
nan_airport

      ident                                 name iata_code
type  \
34675  KCYS  Cheyenne Regional Jerry Olson Field       NaN
medium_airport

       elevation_ft
34675        6159.0
```

```python
# Getting the runway count for 'CYS' from runway dataset.

miss_comb = pd.merge(nan_airport, runways[['airport_ident', 'id']],
how = 'left', left_on = 'ident', right_on = 'airport_ident')

miss_runway_count = miss_comb.groupby('ident')
[['id']].count().reset_index()
miss_runway_count
```

```
  ident  id
0  KCYS   2
```

```python
air_miss_data = pd.merge(nan_airport,
miss_runway_count).rename(columns = {'id': 'runway_count'})
[['iata_code', 'type', 'elevation_ft', 'runway_count']]
air_miss_data
```

```
  iata_code            type  elevation_ft  runway_count
0       NaN  medium_airport        6159.0             2
```

```python
combined_data_traffic.loc[combined_data_traffic.AirportFrom == 'CYS',
['type_source_airport']] = air_miss_data[['type']].values
combined_data_traffic.loc[combined_data_traffic.AirportFrom == 'CYS',
['elevation_ft_source_airport']] =
air_miss_data[['elevation_ft']].values
combined_data_traffic.loc[combined_data_traffic.AirportFrom == 'CYS',
['runway_count_source_airport']] =
air_miss_data[['runway_count']].values

combined_data_traffic.loc[combined_data_traffic.AirportTo == 'CYS',
['type_dest_airport']] = air_miss_data[['type']].values
combined_data_traffic.loc[combined_data_traffic.AirportTo == 'CYS',
['elevation_ft_dest_airport']] =
air_miss_data[['elevation_ft']].values
combined_data_traffic.loc[combined_data_traffic.AirportTo == 'CYS',
['runway_count_dest_airport']] =
air_miss_data[['runway_count']].values

combined_data_traffic.isnull().sum()
```

```
id                                 0
Airline                            0
Flight                             0
AirportFrom                        0
AirportTo                          0
DayOfWeek                          0
Time                               0
Length                             0
Delay                              0
type_source_airport                0
```

```
elevation_ft_source_airport          0
runway_count_source_airport          0
type_dest_airport                    0
elevation_ft_dest_airport            0
runway_count_dest_airport            0
iata_code_source                 83582
data_2021_source_airport         83582
iata_code_dest                   83531
data_2021_dest_airport           83531
Founded                          83601
dtype: int64
```

Let's work on the founded column.

```
airline_dict = pd.read_excel("/content/drive/MyDrive/Simpli Learn
Dataset/Data Dictionary.xlsx", sheet_name = 'airlines',header = 10,
usecols = [0,1])
airline_dict.head(2)
```

```
  Airlines ID Description
0          WN   Southwest
1          DL       Delta
```

```
miss_founded =
combined_data_traffic[combined_data_traffic.Founded.isnull()].Airline.
unique()
miss_founded
```

```
array(['CO', 'US', 'EV'], dtype=object)
```

```
# Let's look for the description of these in our data dictionary.

miss_airline = airline_dict[airline_dict['Airlines
ID'].isin(miss_founded)]
miss_airline
```

```
  Airlines ID                        Description
5          US  PSA (initially US Airway Express)
7          EV                          ExpressJet
9          CO     United Airlines (initially CO)
```

```
miss_val = {'US':1967, 'EV':1986, 'CO':1931}

for aline in miss_founded:

combined_data_traffic.loc[(combined_data_traffic.Founded.isnull())&
(combined_data_traffic.Airline ==aline),'Founded'] = miss_val[aline]

combined_data_traffic.isnull().sum()
```

```
id                                 0
Airline                            0
Flight                             0
AirportFrom                        0
AirportTo                          0
DayOfWeek                          0
Time                               0
Length                             0
Delay                              0
type_source_airport                0
elevation_ft_source_airport        0
runway_count_source_airport        0
type_dest_airport                  0
elevation_ft_dest_airport          0
runway_count_dest_airport          0
iata_code_source               83582
data_2021_source_airport       83582
iata_code_dest                 83531
data_2021_dest_airport         83531
Founded                            0
dtype: int64
```

```
combined_data_traffic.drop(columns = ['iata_code_source',
'iata_code_dest'], inplace = True)
```

```
(combined_data_traffic.isna().sum().sort_values(ascending =
False)/combined_data_traffic.shape[0])*100
```

```
data_2021_source_airport       16.118221
data_2021_dest_airport         16.108386
id                              0.000000
Airline                         0.000000
runway_count_dest_airport       0.000000
elevation_ft_dest_airport       0.000000
type_dest_airport               0.000000
runway_count_source_airport     0.000000
elevation_ft_source_airport     0.000000
type_source_airport             0.000000
Delay                           0.000000
Length                          0.000000
Time                            0.000000
DayOfWeek                       0.000000
AirportTo                       0.000000
AirportFrom                     0.000000
Flight                          0.000000
Founded                         0.000000
dtype: float64
```

We will use the median airport traffic for 16% missing value treatment.

```
med_val = combined_data_traffic.groupby('type_source_airport')
[['data_2021_source_airport']].median()
med_val
```

| | data_2021_source_airport |
|---|---|
| type_source_airport | |
| large_airport | 14514049.0 |
| medium_airport | 2273259.0 |
| small_airport | NaN |

```
for typ in combined_data_traffic.type_source_airport.unique():

combined_data_traffic.loc[(combined_data_traffic.type_source_airport
== typ)& (combined_data_traffic.data_2021_source_airport.isna()),
                      'data_2021_source_airport'] =
med_val.loc[typ].values[0]

med_val_dest = combined_data_traffic.groupby('type_dest_airport')
[['data_2021_dest_airport']].median()
med_val_dest
```

| | data_2021_dest_airport |
|---|---|
| type_dest_airport | |
| large_airport | 14514049.0 |
| medium_airport | 2273259.0 |
| small_airport | NaN |

```
for typ in combined_data_traffic.type_source_airport.unique():

combined_data_traffic.loc[(combined_data_traffic.type_dest_airport ==
typ)& (combined_data_traffic.data_2021_dest_airport.isna()),
                      'data_2021_dest_airport'] =
med_val.loc[typ].values[0]

(combined_data_traffic.isna().sum().sort_values(ascending =
False)/combined_data_traffic.shape[0])*100
```

```
data_2021_source_airport       0.226205
data_2021_dest_airport         0.224855
id                             0.000000
Airline                        0.000000
runway_count_dest_airport      0.000000
elevation_ft_dest_airport      0.000000
type_dest_airport              0.000000
runway_count_source_airport    0.000000
elevation_ft_source_airport    0.000000
type_source_airport            0.000000
Delay                          0.000000
Length                         0.000000
Time                           0.000000
DayOfWeek                      0.000000
```

```
AirportTo                      0.000000
AirportFrom                    0.000000
Flight                         0.000000
Founded                        0.000000
dtype: float64

combined_data_traffic.dropna(subset=['data_2021_source_airport',
'data_2021_dest_airport'], inplace=True)

(combined_data_traffic.isna().sum().sort_values(ascending =
False)/combined_data_traffic.shape[0])*100

id                             0.0
Airline                        0.0
data_2021_dest_airport         0.0
data_2021_source_airport       0.0
runway_count_dest_airport      0.0
elevation_ft_dest_airport      0.0
type_dest_airport              0.0
runway_count_source_airport    0.0
elevation_ft_source_airport    0.0
type_source_airport            0.0
Delay                          0.0
Length                         0.0
Time                           0.0
DayOfWeek                      0.0
AirportTo                      0.0
AirportFrom                    0.0
Flight                         0.0
Founded                        0.0
dtype: float64
```

**combined_data_traffic**

The dataset is now ready for further analytics.

## Data Visualization

- According to data provided, around 70% of the flights are delayed for Southwest Airlines. Visualize to compare the same for other airlines.

```
airline_dict = pd.read_excel("/content/drive/MyDrive/Simpli Learn
Dataset/Data Dictionary.xlsx", sheet_name = 'airlines')
airline_dict.head(2)

    Column                             Description Unnamed: 2
0  Airline  Different types of commercial airlines        NaN
1   Flight                       Types of Aircraft        NaN

southwestid =
airline_dict.loc[airline_dict['Description'].str.strip().str.lower()
```

```
== 'southwest', 'Column'].values[0]
southwestid
```

{"type":"string"}

*Creating copy of the final dataset with a shorter name.*

```
# Creating a copy for the main dataset with a shorter name.
cdt = combined_data_traffic.copy()

# Calculating the % of delays for Southwest Airlines.
round((cdt[cdt.Airline == southwestid].Delay.sum()/
      cdt[cdt.Airline == southwestid].Delay.size)*100)
```

```
70
```

**Yes, ~70% of flights are delayed for SouthWest Airlines.**

```
# Lets define a function to calculate the % of delays for any airline.
def delay_percent(x):
    return round(x.sum()/x.size*100,2)

# Calculating the % of delays for all airlines.

delay_for_all = cdt.groupby('Airline')
['Delay'].apply(delay_percent).reset_index()

delay_for_all.head()
```

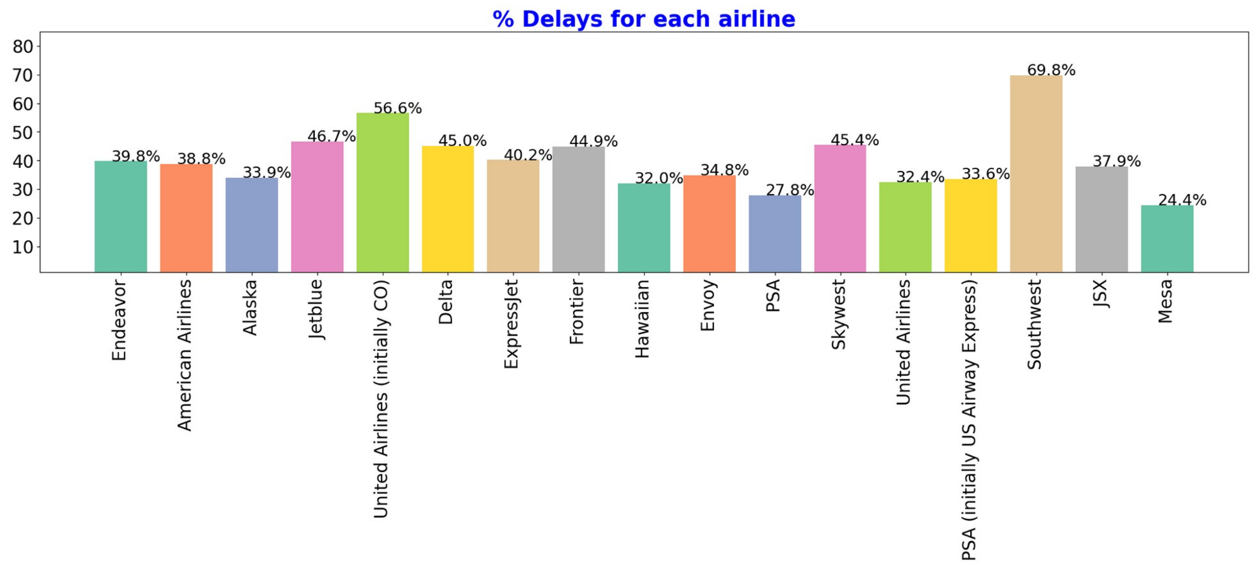
| | Airline | Delay |
|---|---|---|
| 0 | 9E | 39.78 |
| 1 | AA | 38.84 |
| 2 | AS | 33.93 |
| 3 | B6 | 46.70 |
| 4 | CO | 56.58 |

```
# Adding description to the delay_for_all dataset to be used for
plotting.
plot_data = pd.merge(delay_for_all, airline_dict, left_on='Airline',
right_on='Column', how='left')[['Airline','Description','Delay']]
plot_data.head()
```

| | Airline | Description | Delay |
|---|---|---|---|
| 0 | 9E | Endeavor | 39.78 |
| 1 | AA | American Airlines | 38.84 |
| 2 | AS | Alaska | 33.93 |
| 3 | B6 | Jetblue | 46.70 |
| 4 | CO | United Airlines (initially CO) | 56.58 |

```
# Plotting a bar chart for a comparative view of delays in all the
airlines vs the SouthWest Airlines.xlsx
```

```
plt.figure(figsize=(25,5))
plt.bar(plot_data.Description, height = plot_data.Delay, color =
plt.get_cmap('Set2').colors, align='center')
for v, idx in zip(plot_data.Delay.values, plot_data.index):
    plt.annotate('{:.1f}%'.format(v), xy=(idx-0.15,v), size =18)
plt.ylim(1,85)
plt.xticks(size=20, rotation=90)
plt.yticks(size=20)
plt.title('% Delays for each airline', size=25, color='blue',
weight='heavy')
plt.show()
```




**As we can see in the above bar chart, SouthWest airlines has the most delayed flgihts i.e. 69.8% to be exact.**

**Mesa** is the airlines with least delays.

- Number of delayed flights on different weekdays, which days of the week are the safest to travel?

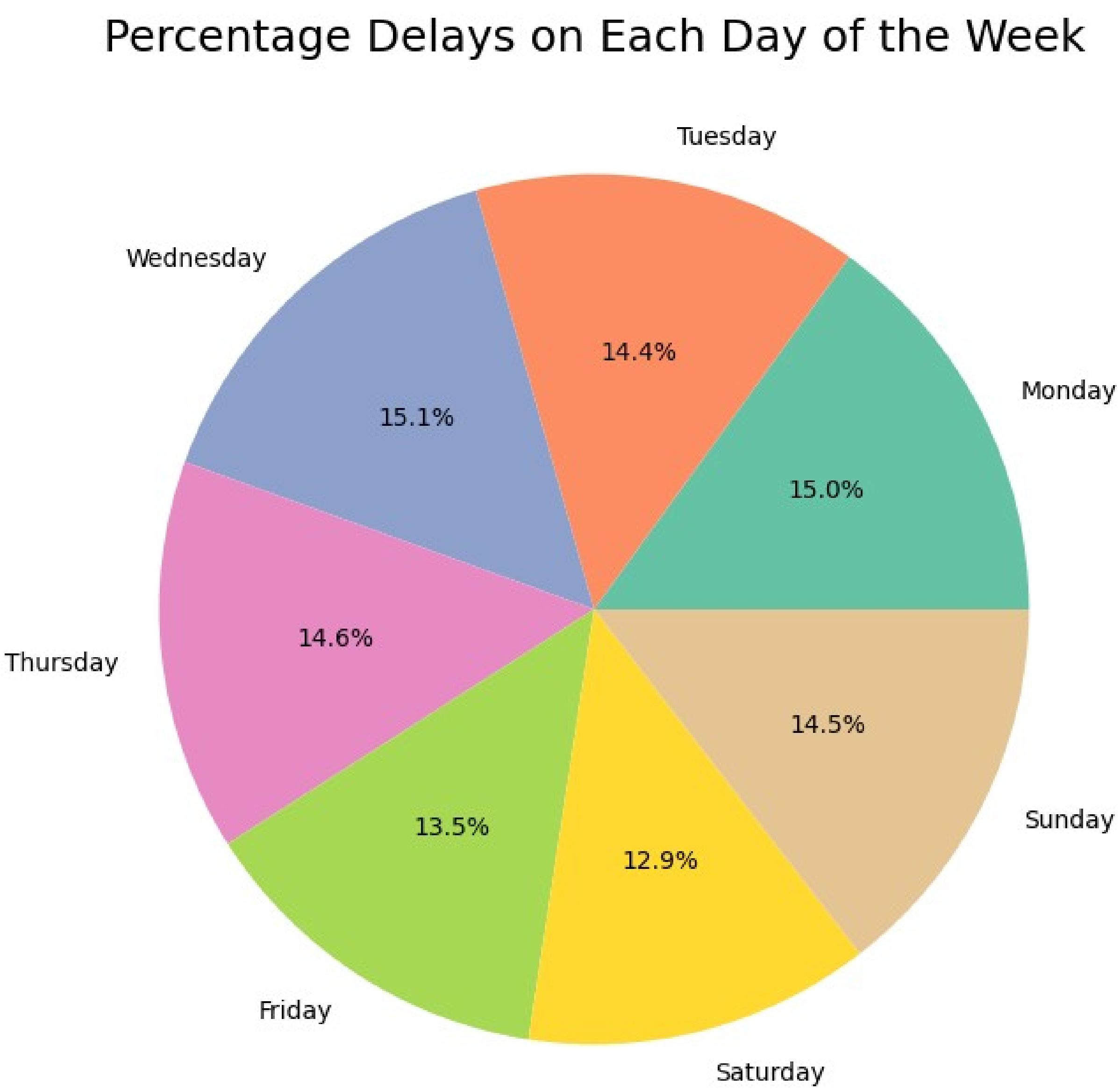
```
cdt.head()

   id Airline  Flight AirportFrom AirportTo  DayOfWeek  Time  Length
Delay  \
0   1      CO     269         SFO       IAH          3    15     205
1
1   2      US    1558         PHX       CLT          3    15     222
1
2   3      AA    2400         LAX       DFW          3    20     165
1
3   4      AA    2466         SFO       DFW          3    20     195
1
4   5      AS     108         ANC       SEA          3    30     202
```

```
0

  type_source_airport  elevation_ft_source_airport  \
0       large_airport                         13.0
1       large_airport                       1135.0
2       large_airport                        125.0
3       large_airport                         13.0
4       large_airport                        152.0

   runway_count_source_airport type_dest_airport
elevation_ft_dest_airport  \
0                          4.0     large_airport
97.0
1                          3.0     large_airport
748.0
2                          4.0     large_airport
607.0
3                          4.0     large_airport
607.0
4                          3.0     large_airport
433.0

   runway_count_dest_airport  data_2021_source_airport  \
0                        5.0                11725347.0
1                        4.0                18940287.0
2                        7.0                23663410.0
3                        7.0                11725347.0
4                        4.0                 2184959.0

   data_2021_dest_airport  Founded
0              16242821.0   1931.0
1              20900875.0   1967.0
2              30005266.0   1926.0
3              30005266.0   1926.0
4              17430195.0   1932.0
```

```
# Lets calculate the % of delays for each airline for each weekday.

delay_each_day = cdt.groupby('DayOfWeek')
['Delay'].apply(delay_percent).reset_index()
delay_each_day
```

```
   DayOfWeek  Delay
0          1  47.28
1          2  45.25
2          3  47.63
3          4  45.84
4          5  42.58
5          6  40.57
6          7  45.77
```

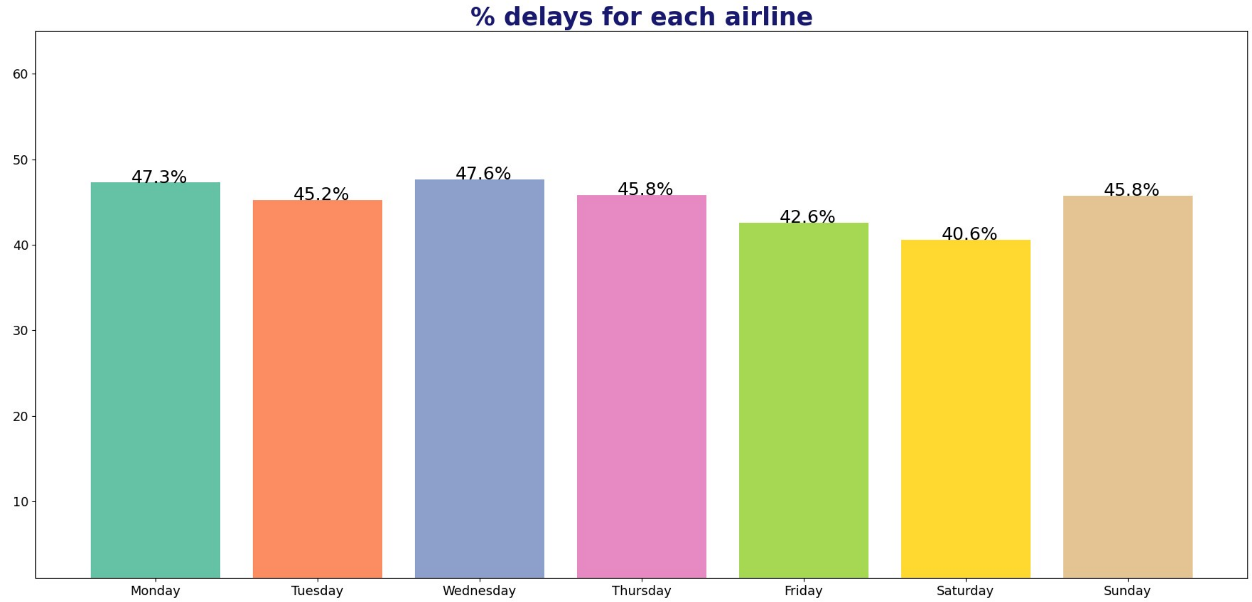
```python
# Data for the pie chart
day_numbers = delay_each_day['DayOfWeek']
day_names = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
'Saturday', 'Sunday']
values = delay_each_day['Delay'].values

# Plotting the pie chart
plt.figure(figsize=(8, 8))
plt.pie(values, labels=day_names, autopct='%.1f%%',
colors=plt.get_cmap('Set2').colors)

# Adding a title
plt.title('Percentage Delays on Each Day of the Week', size=18)

# Displaying the pie chart
plt.show()
```




**From the pie chart above the best days to travel are: Friday and Saturday.**

```
plt.figure(figsize = (22,10))
plt.bar(delay_each_day['DayOfWeek'], height =
delay_each_day['Delay'].values, color = plt.get_cmap('Set2').colors)
for v, idx in zip(delay_each_day['Delay'].values, range(1,
len(delay_each_day['DayOfWeek'])+1)):
    plt.annotate('{:.1f}%'.format(v), xy = (idx-0.15, v), size = 18)
plt.ylim(1,65)
plt.xticks(range(1, len(delay_each_day['DayOfWeek'])+1), day_names,
size=13)
plt.yticks(size = 13)
```

```
plt.title('% delays for each airline', size = 25, color =
'midnightblue', weight = 'heavy')
plt.show()
```




**From the bar chart above, we can see that Friday and Saturday have the shortest bars for % delays, hence these two days are the safest to travel.**

- Which airlines to recommend for short, medium, and long length travel?

```
duration = cdt[['Airline', 'Length', 'Delay']].copy()
duration.head()
```

| | Airline | Length | Delay |
|---|---|---|---|
| 0 | CO | 205 | 1 |
| 1 | US | 222 | 1 |
| 2 | AA | 165 | 1 |
| 3 | AA | 195 | 1 |
| 4 | AS | 202 | 0 |

```
# Dataset with delay % of each airline for short, medium, and long
range travel.

duration['travel_time'] = pd.cut(duration.Length, 3, labels =
['short', 'medium', 'long'])
duration_grp = duration.groupby(['Airline', 'travel_time'])
['Delay'].apply(delay_percent).reset_index().pivot(index = 'Airline',
columns = 'travel_time').fillna(0)['Delay']
duration_grp.columns = duration_grp.columns.astype(str)
duration_grp.reset_index()
```

| travel_time | Airline | short | medium | long |
|---|---|---|---|---|
| 0 | 9E | 39.78 | 0.00 | 0.00 |

```
1                 AA  37.61   43.25  60.40
2                 AS  32.58   38.17   0.00
3                 B6  45.70   51.05   0.00
4                 CO  52.82   64.95  66.87
5                 DL  43.87   50.24  48.62
6                 EV  40.20   50.00   0.00
7                 F9  45.04   43.56   0.00
8                 HA  30.16   40.48   0.00
9                 MQ  34.81   27.42   0.00
10                OH  27.71   39.20   0.00
11                OO  45.40   53.03   0.00
12                UA  29.92   37.10  39.26
13                US  31.96   40.72   0.00
14                WN  69.12   77.61   0.00
15                XE  37.87   53.70   0.00
16                YV  24.37   25.86   0.00
```

```
airline_dict.rename(columns = {'Column':'Airline'}, inplace = True)

airline_dict.Description = airline_dict.Description.str.strip()

duration_grp = pd.merge(duration_grp, airline_dict[['Airline',
'Description']], how = 'left', left_on = 'Airline', right_on =
'Airline')

duration_grp
```

|  | Airline | short | medium | long | Description |
|---|---|---|---|---|---|
| 0 | 9E | 39.78 | 0.00 | 0.00 | Endeavor |
| 1 | AA | 37.61 | 43.25 | 60.40 | American Airlines |
| 2 | AS | 32.58 | 38.17 | 0.00 | Alaska |
| 3 | B6 | 45.70 | 51.05 | 0.00 | Jetblue |
| 4 | CO | 52.82 | 64.95 | 66.87 | United Airlines (initially CO) |
| 5 | DL | 43.87 | 50.24 | 48.62 | Delta |
| 6 | EV | 40.20 | 50.00 | 0.00 | ExpressJet |
| 7 | F9 | 45.04 | 43.56 | 0.00 | Frontier |
| 8 | HA | 30.16 | 40.48 | 0.00 | Hawaiian |
| 9 | MQ | 34.81 | 27.42 | 0.00 | Envoy |
| 10 | OH | 27.71 | 39.20 | 0.00 | PSA |
| 11 | OO | 45.40 | 53.03 | 0.00 | Skywest |
| 12 | UA | 29.92 | 37.10 | 39.26 | United Airlines |
| 13 | US | 31.96 | 40.72 | 0.00 | PSA (initially US Airway Express) |
| 14 | WN | 69.12 | 77.61 | 0.00 | Southwest |
| 15 | XE | 37.87 | 53.70 | 0.00 | JSX |
| 16 | YV | 24.37 | 25.86 | 0.00 | Mesa |

```
cdt.Airline.nunique()

17

duration_grp.short.min()
```

```
24.37

long = duration_grp[duration_grp.long ==
duration_grp.long.min()].Description.values.tolist()
print(len(long),'Airlines with minimum delays (0%) for long flights:\
n',','.join(long))

medium= duration_grp[duration_grp.medium ==
duration_grp.medium.min()].Description.values.tolist()
print('\n',len(medium),'Airline with minimum delays (0%) for medium
flights:\n',','.join(medium))

short = duration_grp[duration_grp.short ==
duration_grp.short.min()].Description.values.tolist()
print('\n', len(short),'Airline with minimum delays (24.37%) for short
flights:\n',','.join(short))

13 Airlines with minimum delays (0%) for long flights:

Endeavor,Alaska,Jetblue,ExpressJet,Frontier,Hawaiian,Envoy,PSA,Skywest
,PSA (initially US Airway Express),Southwest,JSX,Mesa

 1 Airline with minimum delays (0%) for medium flights:
 Endeavor

 1 Airline with minimum delays (24.37%) for short flights:
 Mesa
```

**Here are the recommended airlines with minimum delays that are safest to travel for each kind of travel distance:**

- **Long flights (0% delay)**: Endeavor, Alaska, Jetblue, ExpressJet, Frontier, Hawaiian, Envoy, PSA, Skywest, PSA (initially US Airway Express), Southwest, JSX, Mesa
- **Medium flights (0% delays)**: Endeavor
- **Short flights (24.37% delays)**: Mesa
- Do you observe any pattern in the time of departure of flights of long duration?

```
# Creating three equal width bins for length of flights and adding
them to the dataset.
cdt['duration'] = pd.cut(cdt.Length,3, labels = ['short', 'medium',
'long'])

cdt.head()

   id Airline  Flight AirportFrom AirportTo  DayOfWeek  Time  Length
Delay  \
0   1      CO     269         SFO       IAH          3    15     205
1
1   2      US    1558         PHX       CLT          3    15     222
1
2   3      AA    2400         LAX       DFW          3    20     165
```

```
1
3   4      AA    2466         SFO       DFW          3    20     195
1
4   5      AS     108         ANC       SEA          3    30     202
0

  type_source_airport  elevation_ft_source_airport  \
0       large_airport                         13.0
1       large_airport                       1135.0
2       large_airport                        125.0
3       large_airport                         13.0
4       large_airport                        152.0

   runway_count_source_airport type_dest_airport
elevation_ft_dest_airport  \
0                          4.0     large_airport
97.0
1                          3.0     large_airport
748.0
2                          4.0     large_airport
607.0
3                          4.0     large_airport
607.0
4                          3.0     large_airport
433.0

   runway_count_dest_airport  data_2021_source_airport  \
0                        5.0                11725347.0
1                        4.0                18940287.0
2                        7.0                23663410.0
3                        7.0                11725347.0
4                        4.0                 2184959.0

   data_2021_dest_airport  Founded duration
0              16242821.0   1931.0    short
1              20900875.0   1967.0   medium
2              30005266.0   1926.0    short
3              30005266.0   1926.0    short
4              17430195.0   1932.0    short
```

```
cdt.isna().sum()
```

```
id                             0
Airline                        0
Flight                         0
AirportFrom                    0
AirportTo                      0
DayOfWeek                      0
Time                           0
Length                         0
```

```
Delay                            0
type_source_airport              0
elevation_ft_source_airport      0
runway_count_source_airport      0
type_dest_airport                0
elevation_ft_dest_airport        0
runway_count_dest_airport        0
data_2021_source_airport         0
data_2021_dest_airport           0
Founded                          0
duration                         0
dtype: int64
```

```
pd.crosstab(cdt.Time, cdt.duration)['long']
```

```
<ipython-input-105-07602ef52afb>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  pd.crosstab(cdt.Time, cdt.duration)['long']
```

```
Time
10      0
15      0
20      0
21      0
25      0
       ..
1428    0
1430    0
1431    0
1435    0
1439    0
Name: long, Length: 1131, dtype: int64
```

```
y = pd.crosstab(cdt.Time, cdt.duration)['long'].index
x = pd.crosstab(cdt.Time, cdt.duration)['long'].values
```

```
filter_data = cdt.loc[cdt.duration == 'long', ['Time', 'duration']]
filter_data.head()
```

```
      Time duration
4232   565     long
4772   595     long
5738   650     long
6104   670     long
6477   690     long
```

```
filter_data.Time.describe()
```

```
count     559.000000
mean      840.635063
std       221.020092
min       540.000000
25%       670.000000
50%       717.000000
75%      1045.000000
max      1310.000000
Name: Time, dtype: float64
```

# Departure Time Analysis

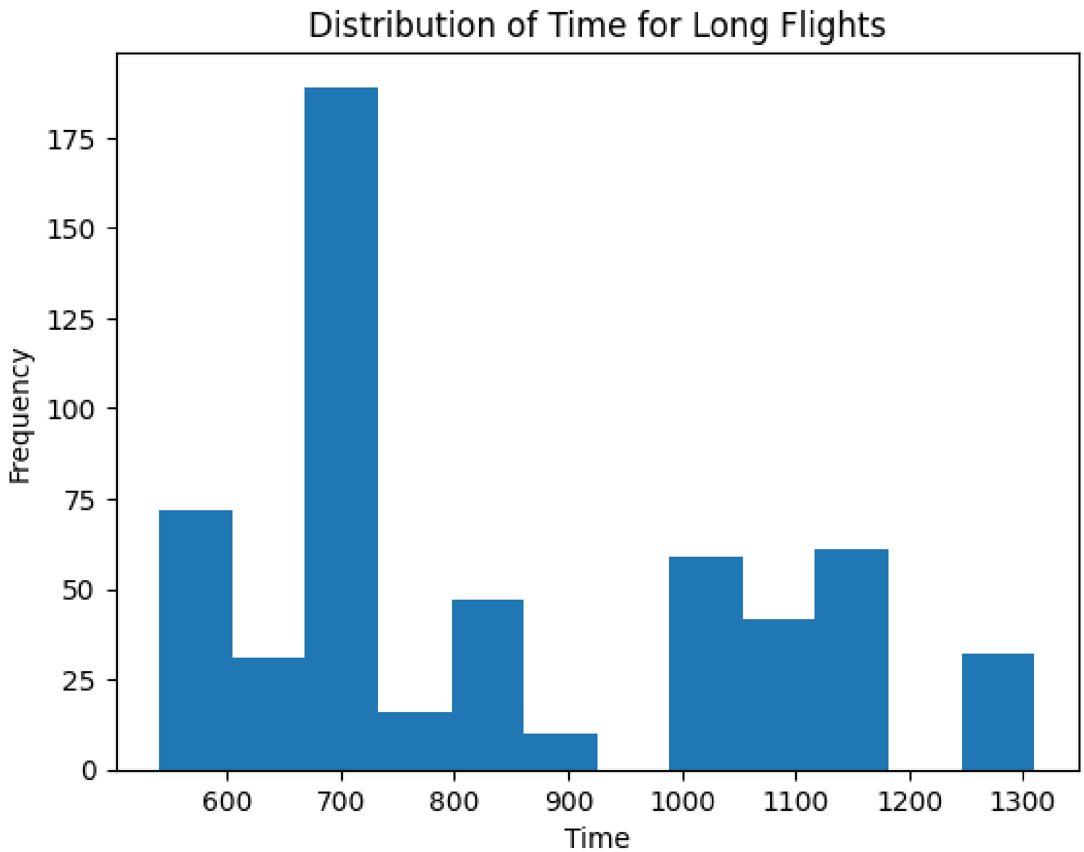
- **Departure Time Range**: The dataset's departure times span from a minimum of 540 minutes (equivalent to after midnight) to a maximum of 1310 minutes. This wide range indicates that flights in the dataset have departure times covering the entire day.

- **Average Departure Time**: The average departure time, represented by the mean of approximately 840.64 minutes, suggests that, on average, flights tend to depart around 8:40 AM (assuming midnight as the starting point).

- **Variability of Departure Times**: The standard deviation, which is approximately 221.02 minutes, implies that departure times exhibit a moderate level of variability or spread around the mean. This indicates that departure times are not tightly clustered around a specific time but rather show some degree of dispersion.

- **Quartiles**: Examining quartiles, the 25th percentile (first quartile) value of 670 minutes and the 75th percentile (third quartile) value of 1045 minutes provide insight into the distribution of departure times. Specifically, it reveals that 25% of the flights depart before 6:10 AM (approximately), while 75% of the flights depart before 5:45 PM (approximately).

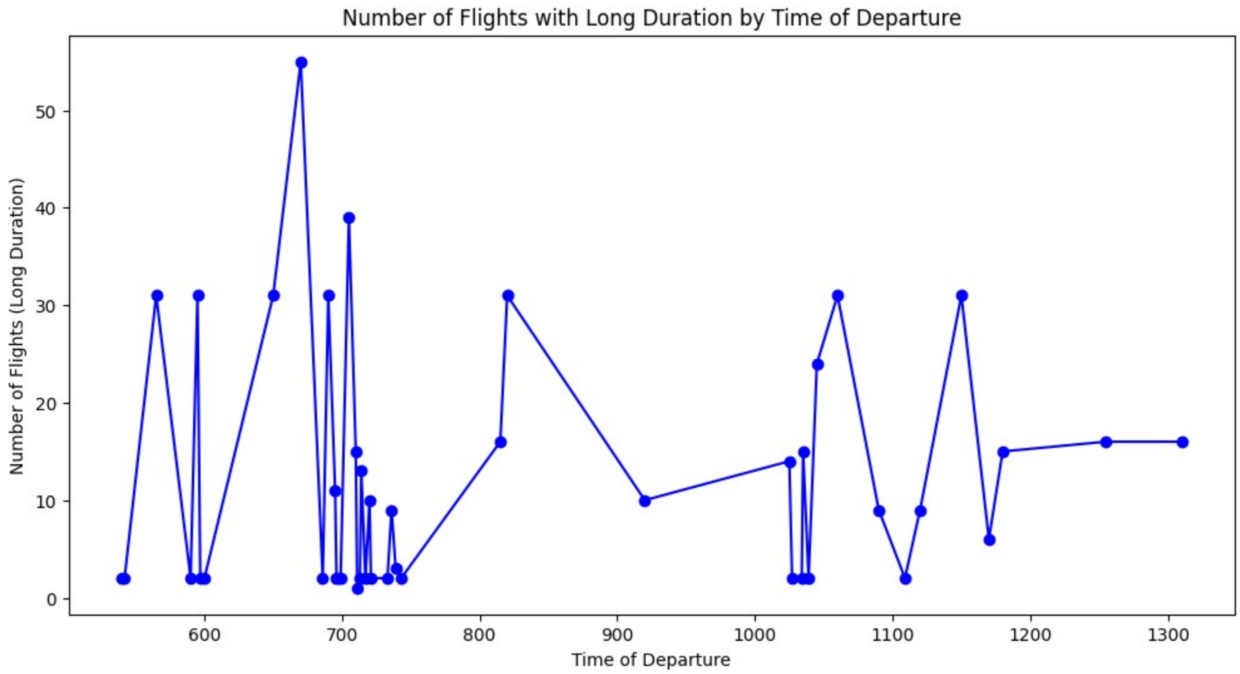
```
# Plotting a bar chart for visualizing the distribution of time for
long flights.
plt.hist(filter_data.Time, bins=12)
plt.xlabel('Time')
plt.ylabel('Frequency')
plt.title('Distribution of Time for Long Flights')
plt.show()
```




- **Maximum** number of flights are departing at ~700 minutes, i.e. around 11:40 AM.
- There are **two empty time zones**, ~(900-1000) and ~(1160-1240) when no long distances flights depart.

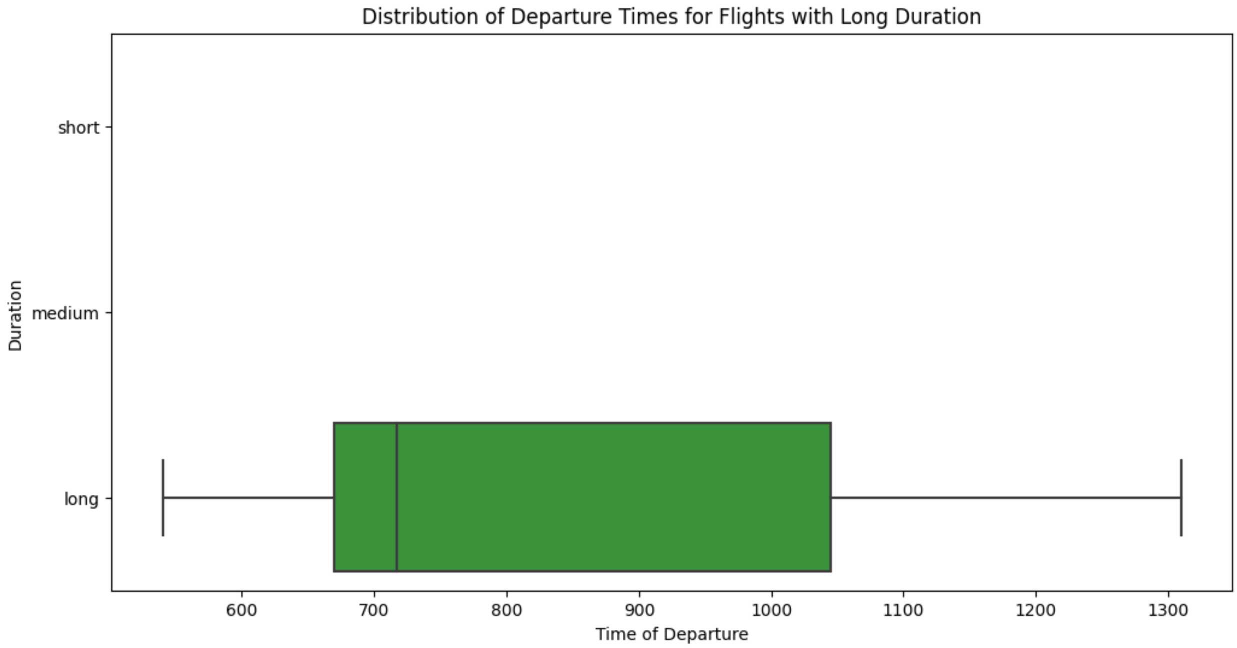
```
# Plotting a line graph to visualize the distribution of departure
time of long flights.
# Filter the data for flights with long duration
long_duration_flights = cdt[cdt['duration'] == 'long']

# Group the data by time and count the number of flights
time_counts =
long_duration_flights['Time'].value_counts().sort_index()

# Plot the line chart
plt.figure(figsize=(12, 6))
plt.plot(time_counts.index, time_counts.values, marker='o',
linestyle='-', color='blue')

# Set the axis labels and title
plt.xlabel('Time of Departure')
plt.ylabel('Number of Flights (Long Duration)')
plt.title('Number of Flights with Long Duration by Time of Departure')
```

```python
# Show the plot
plt.show()
```




- Maxium number of flights depart between ~(660-710) minutes from midnight.
- The two empty zones identified above are not ture. There are some flights departing at almost every time stamp.

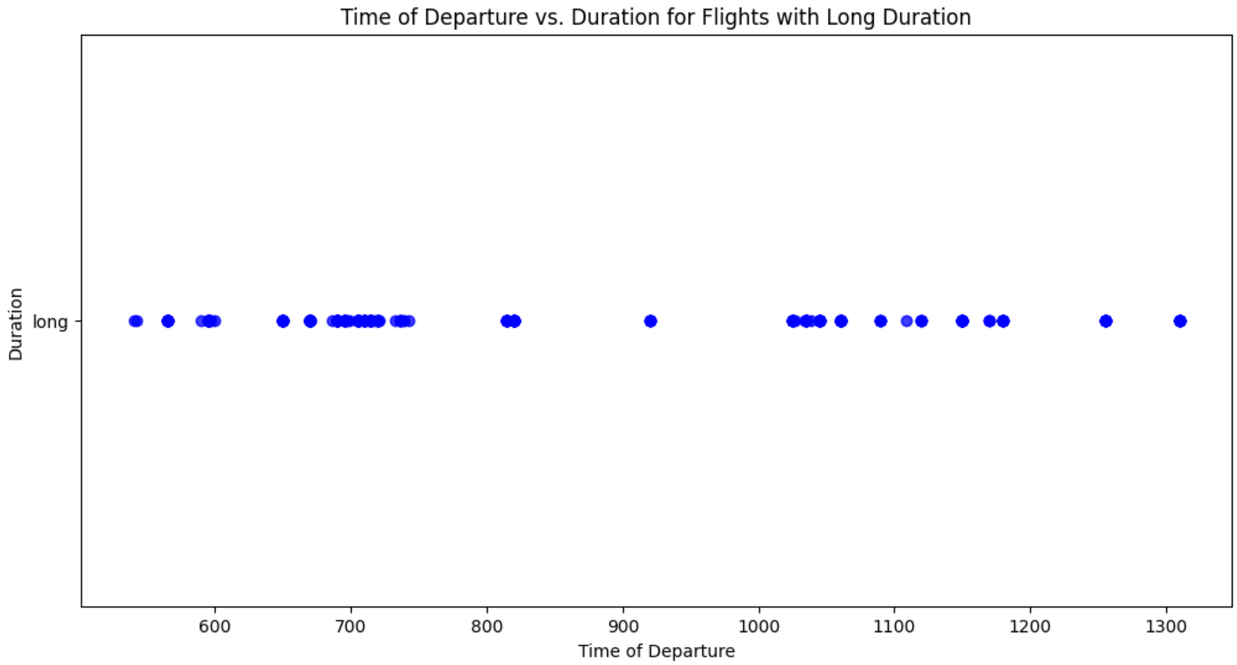
```python
# Filter the data for flights with long duration
long_duration_flights = cdt[cdt['duration'] == 'long']

# Plot the box plot
plt.figure(figsize=(12, 6))
sns.boxplot(data=long_duration_flights, x='Time', y='duration')

# Set the axis labels and title
plt.xlabel('Time of Departure')
plt.ylabel('Duration')
plt.title('Distribution of Departure Times for Flights with Long
Duration')

# Show the plot
plt.show()
```




Observations till now:

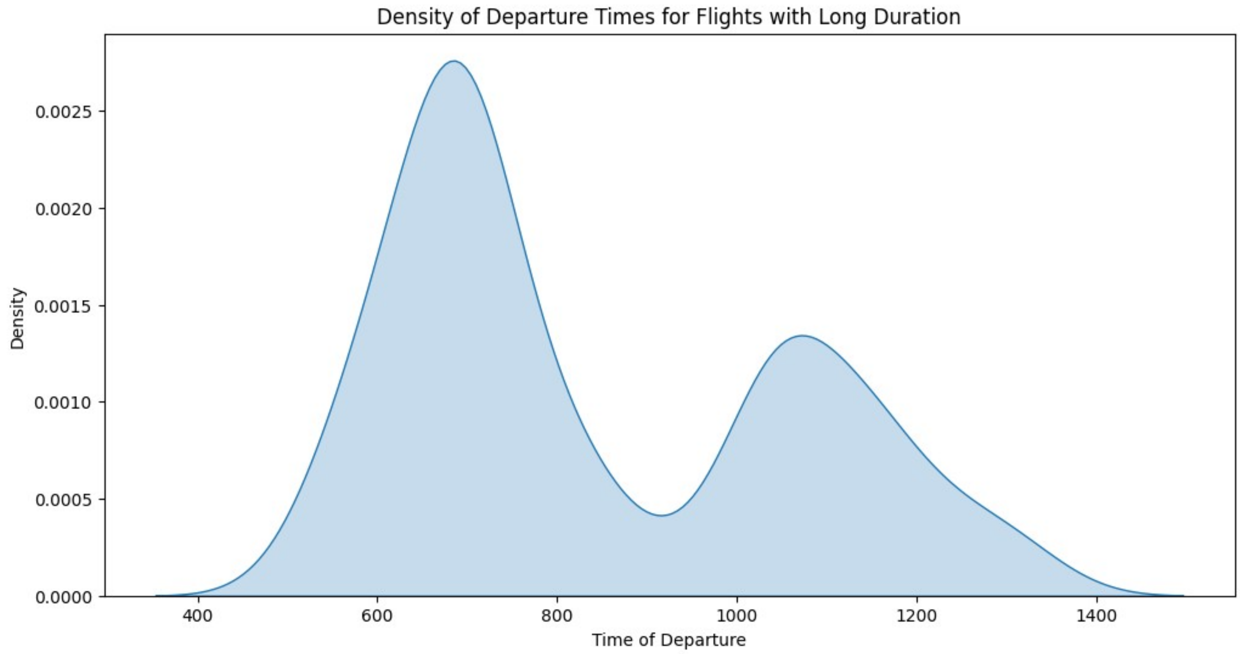
- 75% of all the long duration flights depart before 6:00 PM in the evening.
- 50% of flights depart before 12:00 PM in the afternoon.
- The earliest long duration flight departs at 9:00 AM.
- The last long duration flight departs at 10:50 PM.

```
# Plotting a scatter plot for visualizing the distribution of
departure time of long flights.
# Filter the data for flights with long duration
long_duration_flights = cdt[cdt['duration'] == 'long']

# Plot the scatter plot
plt.figure(figsize=(12, 6))
plt.scatter(long_duration_flights['Time'],
long_duration_flights['duration'], color='blue', alpha=0.5)

# Set the axis labels and title
plt.xlabel('Time of Departure')
plt.ylabel('Duration')
plt.title('Time of Departure vs. Duration for Flights with Long
Duration')

# Show the plot
plt.show()
```




The clustered scatter plot confirms the observations made above.

```
# Plotting a KDE plot for visualizing the distribution of departure
time of long flights.
# Filter the data for flights with long duration
long_duration_flights = cdt[cdt['duration'] == 'long']

# Plot the KDE plot
plt.figure(figsize=(12, 6))
sns.kdeplot(data=long_duration_flights, x='Time', fill=True)

# Set the axis labels and title
plt.xlabel('Time of Departure')
plt.ylabel('Density')
plt.title('Density of Departure Times for Flights with Long Duration')

# Show the plot
plt.show()
```




The KDE plot confirms the observatios made above.

```
from scipy.stats import ttest_ind, f_oneway
import pandas as pd

# Convert the 'Time' column to numeric data type
cdt['Time'] = pd.to_numeric(cdt['Time'], errors='coerce')

# Filter the data for flights with long duration and other durations
long_duration_flights = cdt[cdt['duration'] == 'long']
other_duration_flights = cdt[cdt['duration'] != 'long']

# Perform t-test to compare the means of departure times
t_statistic, p_value = ttest_ind(long_duration_flights['Time'],
other_duration_flights['Time'], nan_policy='omit')

# Perform ANOVA to compare the means of departure times across
different durations
f_statistic, p_value_anova = f_oneway(cdt['Time'],
pd.Categorical(cdt['duration']).codes)

print("T-test: T-statistic =", t_statistic, "p-value =", p_value)
print("ANOVA: F-statistic =", f_statistic, "p-value =", p_value_anova)

T-test: T-statistic = 3.335989828914716 p-value =
0.0008500226450162412
ANOVA: F-statistic = 4299971.900179982 p-value = 0.0
```

- **T-test Result**: The t-statistic of 3.336 indicates a significant difference in the means of departure times between long-duration flights and other-duration flights. This suggests that there is a distinct pattern or variation in the departure times of long-duration flights compared to other-duration flights.

- **ANOVA Result**: The ANOVA test indicates a significant difference in departure times across different flight durations, including long-duration flights. This suggests that the departure times of long-duration flights are not only different from other-duration flights but also show variation within the group of long-duration flights themselves. This further supports the existence of patterns or systematic differences in the departure times of long-duration flights.

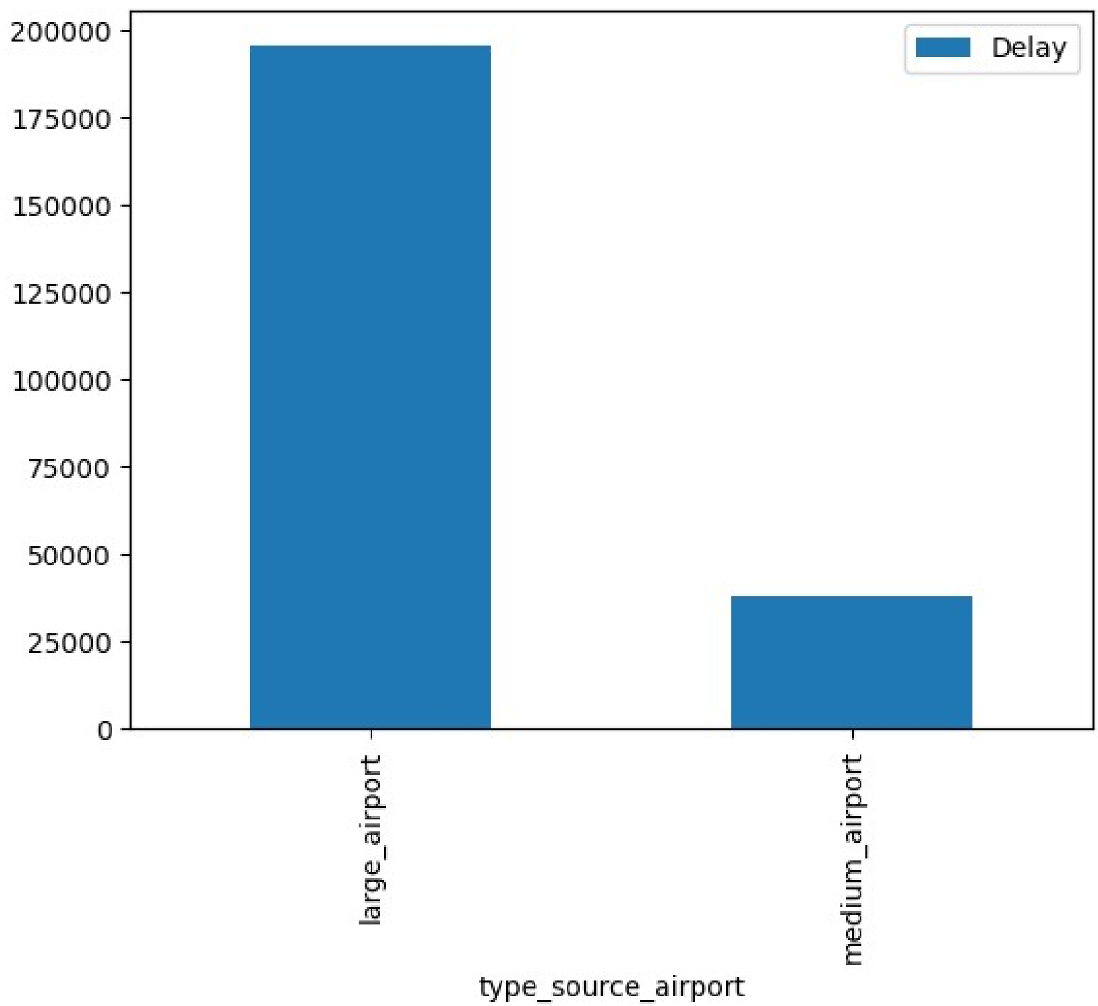
- How large hubs compare to medium hubs in terms of count of delayed flights? Use appropriate visualization to represent your findings.

```
cdt.head()

   id Airline  Flight AirportFrom AirportTo  DayOfWeek  Time  Length
Delay  \
0   1      CO     269         SFO       IAH          3    15     205
1
1   2      US    1558         PHX       CLT          3    15     222
1
2   3      AA    2400         LAX       DFW          3    20     165
1
3   4      AA    2466         SFO       DFW          3    20     195
1
4   5      AS     108         ANC       SEA          3    30     202
0

  type_source_airport  elevation_ft_source_airport  \
0       large_airport                         13.0
1       large_airport                       1135.0
2       large_airport                        125.0
3       large_airport                         13.0
4       large_airport                        152.0

   runway_count_source_airport type_dest_airport
elevation_ft_dest_airport  \
0                          4.0     large_airport
97.0
1                          3.0     large_airport
748.0
2                          4.0     large_airport
607.0
3                          4.0     large_airport
607.0
4                          3.0     large_airport
433.0

   runway_count_dest_airport  data_2021_source_airport  \
0                        5.0                11725347.0
```

```
1                        4.0                   18940287.0
2                        7.0                   23663410.0
3                        7.0                   11725347.0
4                        4.0                    2184959.0

   data_2021_dest_airport  Founded duration
0              16242821.0   1931.0    short
1              20900875.0   1967.0   medium
2              30005266.0   1926.0    short
3              30005266.0   1926.0    short
4              17430195.0   1932.0    short
```

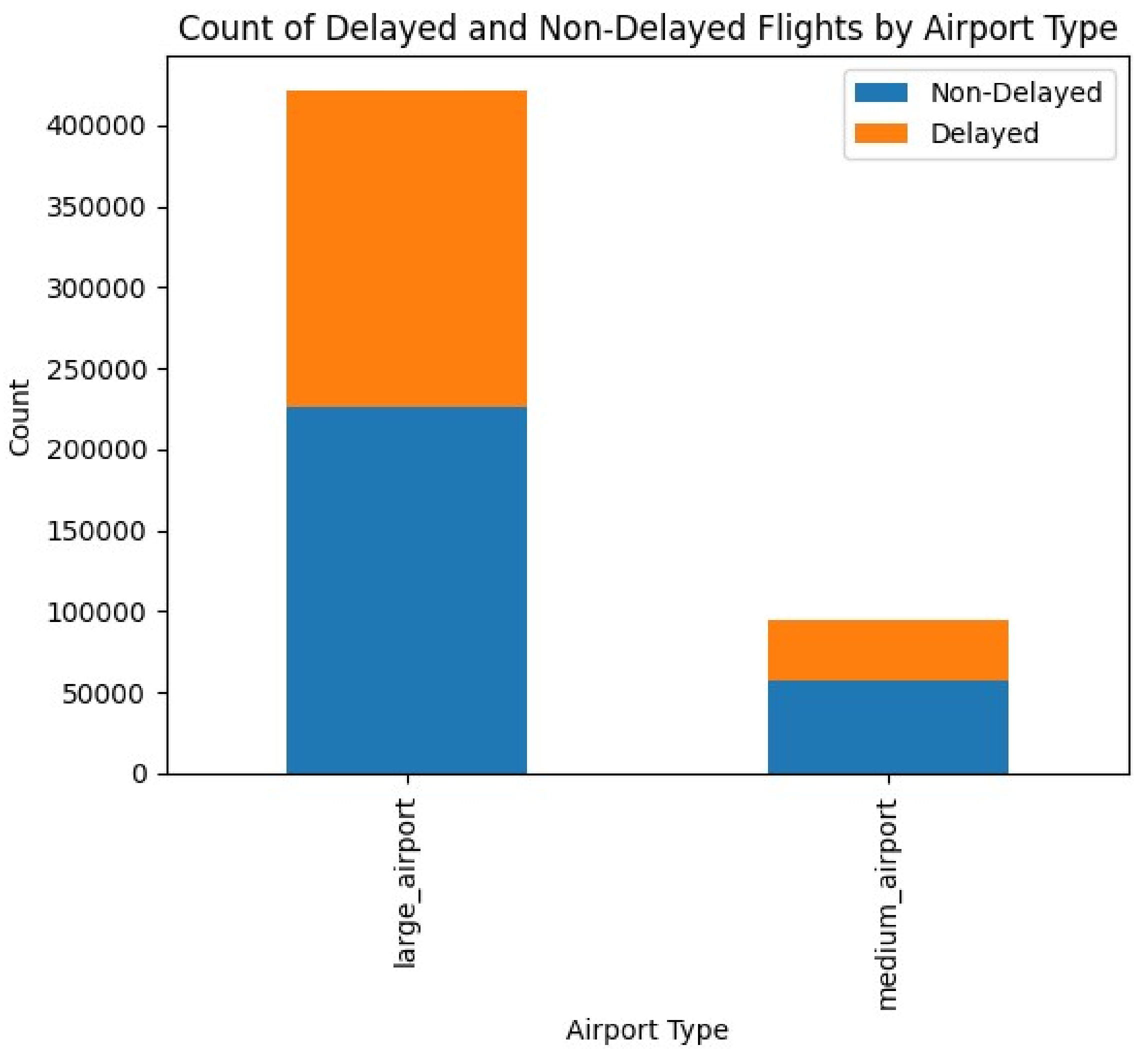
```
cdt.groupby('type_source_airport')[['Delay']].agg('sum').plot.bar()
```

```
<Axes: xlabel='type_source_airport'>
```



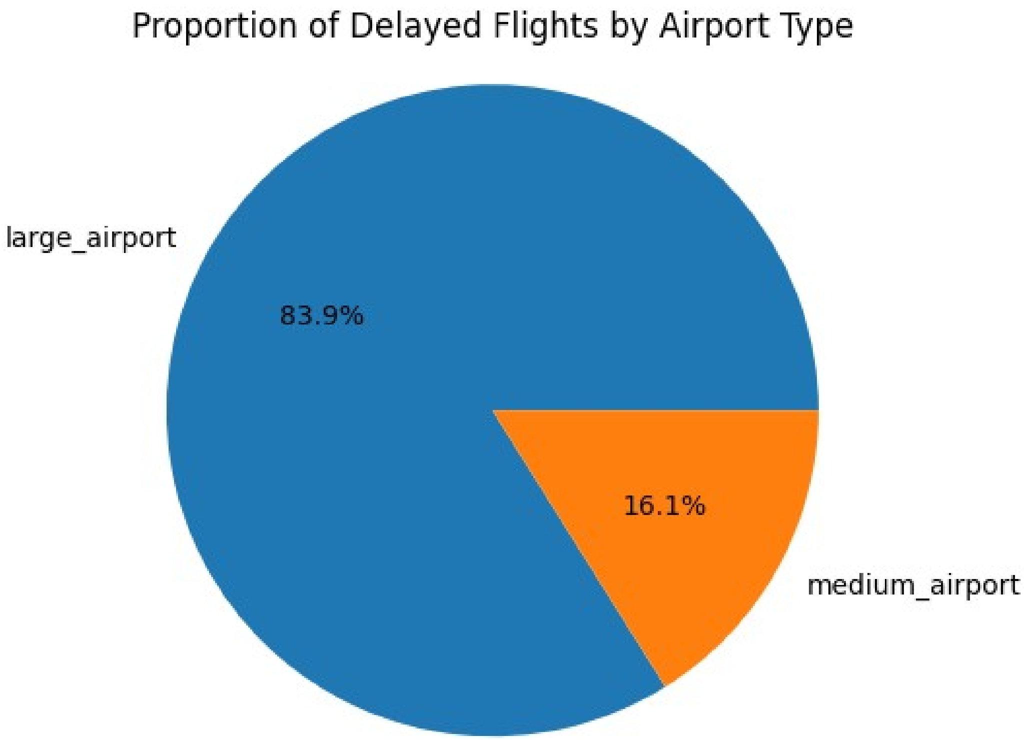

Large airports have **higher count of delayed flights**. But this could be due to the fact that **they serve a larger number of people with more number of flights and routes**.

```python
# Creating a stacked bar plot to visualize the balance between delayed and non-delayed flights.
delay_counts = cdt.groupby('type_source_airport')
['Delay'].value_counts().unstack()
delay_counts.plot(kind='bar', stacked=True)
plt.xlabel('Airport Type')
plt.ylabel('Count')
plt.title('Count of Delayed and Non-Delayed Flights by Airport Type')
plt.legend(['Non-Delayed', 'Delayed'])
plt.show()
```




- The plot shows us that there is a significant difference between the number of flights large and medium airports handle. **Medium airports have a higher ratio of non-delayed flights**. This could mean that given the traffic each type of hub handles, medium airports are better at ensuring timely take-offs.

```
delay_counts = cdt[cdt['Delay'] ==
1].groupby('type_source_airport').size()
delay_counts.plot(kind='pie', autopct='%1.1f%%')
plt.axis('equal')
plt.title('Proportion of Delayed Flights by Airport Type')
plt.show()
```




```
# Calculate the proportion of delayed flights for each airport type
delay_proportion = cdt.groupby('type_source_airport')
['Delay'].mean()*100

delay_proportion

type_source_airport
large_airport     46.408639
medium_airport    39.601642
Name: Delay, dtype: float64
```

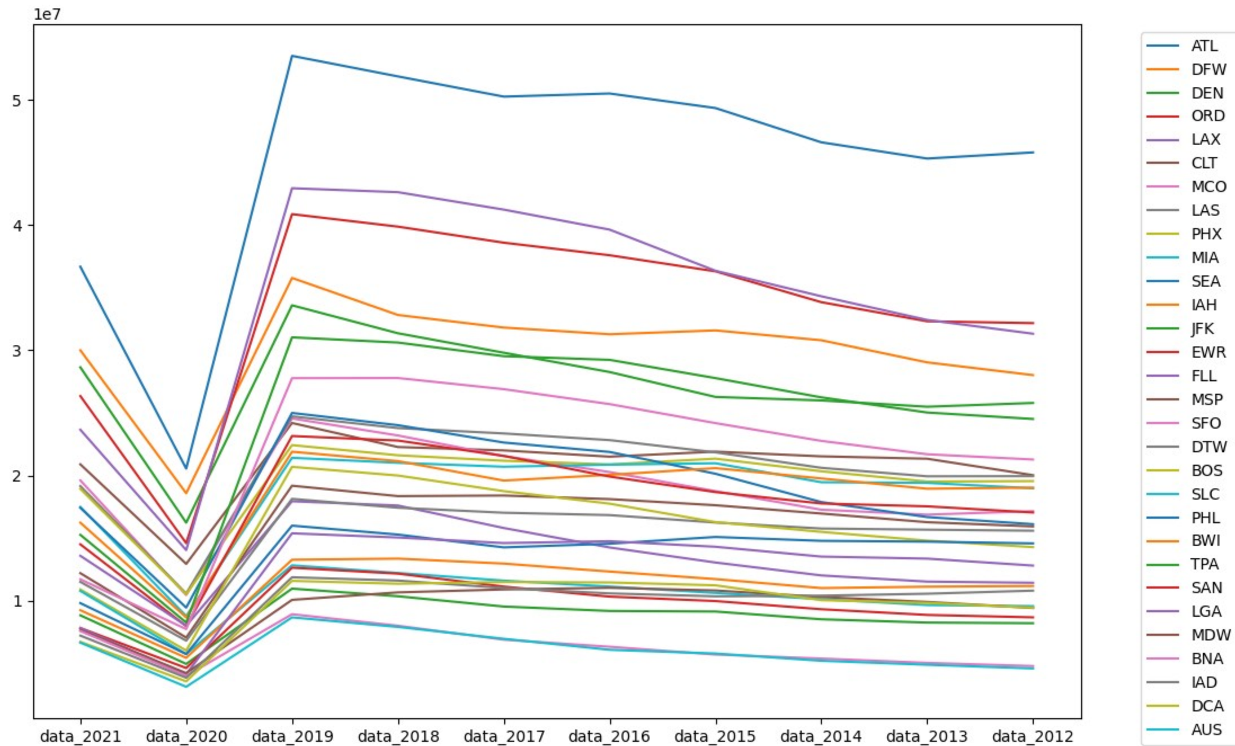
Large airports have a 46% delay where as medium airports have 39% delay. Therefore, we can conclude that **large_airports have a higher chance of a delayed flights than medium_airports**.

- For large hubs, forecast the number of passengers for 2022 using simple moving average method.

```
# Lets develop a series of traffic data for large airports.
```

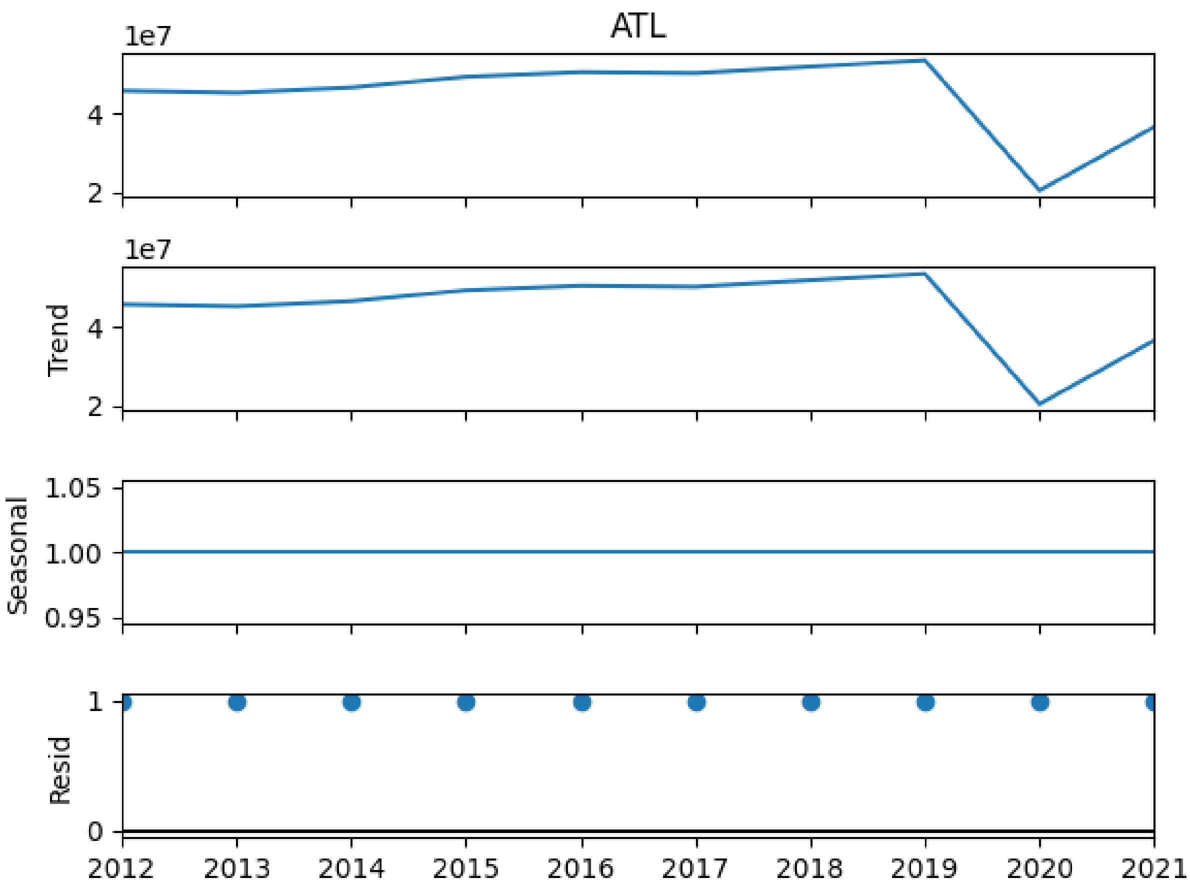
```
cols = ['iata_code']+
final_hub_data.columns[final_hub_data.columns.str.startswith('data_')]
.tolist()

final_hub_data.head()

   rank hub_type                                          airports
iata_code  \
0     1    large  Hartsfield–Jackson Atlanta International Airport
ATL
1     2    large           Dallas/Fort Worth International Airport
DFW
2     3    large                      Denver International Airport
DEN
3     4    large                      O'Hare International Airport
ORD
4     5    large                 Los Angeles International Airport
LAX

          ciity_served state  data_2021  data_2020  data_2019
data_2018  \
0              Atlanta    GA   36676010   20559866   53505795
51865797
1  Dallas & Fort Worth    TX   30005266   18593421   35778573
32821799
2               Denver    CO   28645527   16243216   33592945
31362941
3              Chicago    IL   26350976   14606034   40871223
39873927
4          Los Angeles    CA   23663410   14055777   42939104
42624050

   data_2017  data_2016  data_2015  data_2014  data_2013  data_2012  \
0   50251964   50501858   49340732   46604273   45308407   45798928
1   31816933   31283579   31589839   30804567   29038128   28022904
2   29809097   28267394   26280043   26000591   25496885   25799841
3   38593028   37589899   36305668   33843426   32317835   32171795
4   41232432   39636042   36351272   34314197   32425892   31326268

  city_served
0         NaN
1         NaN
2         NaN
3         NaN
4         NaN

time_series = final_hub_data.loc[final_hub_data.hub_type == 'large',
cols].set_index('iata_code').T

time_series['ATL']
```

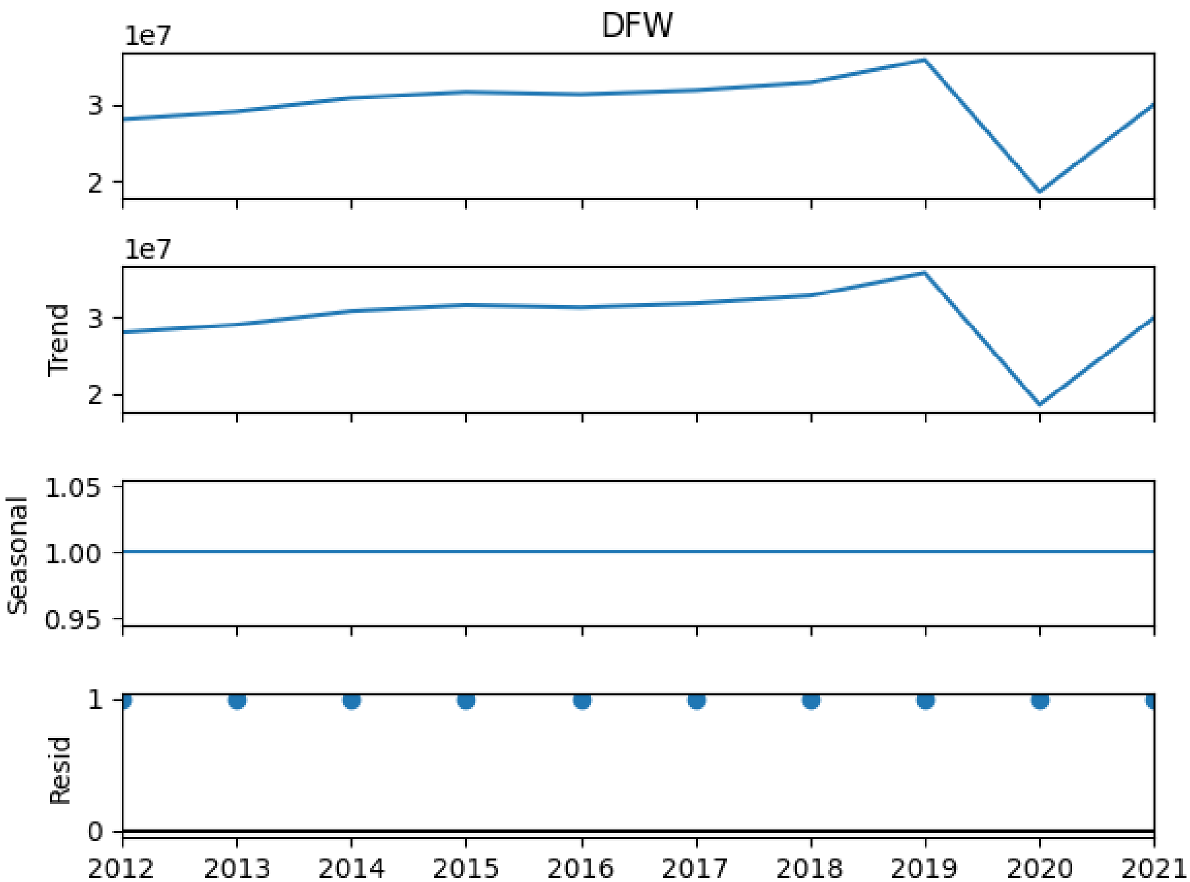
```
data_2021    36676010
data_2020    20559866
data_2019    53505795
data_2018    51865797
data_2017    50251964
data_2016    50501858
data_2015    49340732
data_2014    46604273
data_2013    45308407
data_2012    45798928
Name: ATL, dtype: int64

# Plotting the time series for each airport.
plt.figure(figsize=(12, 8))
for ser in time_series.columns:
    plt.plot(time_series[ser], label=ser)

# Set the legend outside the chart
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')

plt.show(block=True)
```

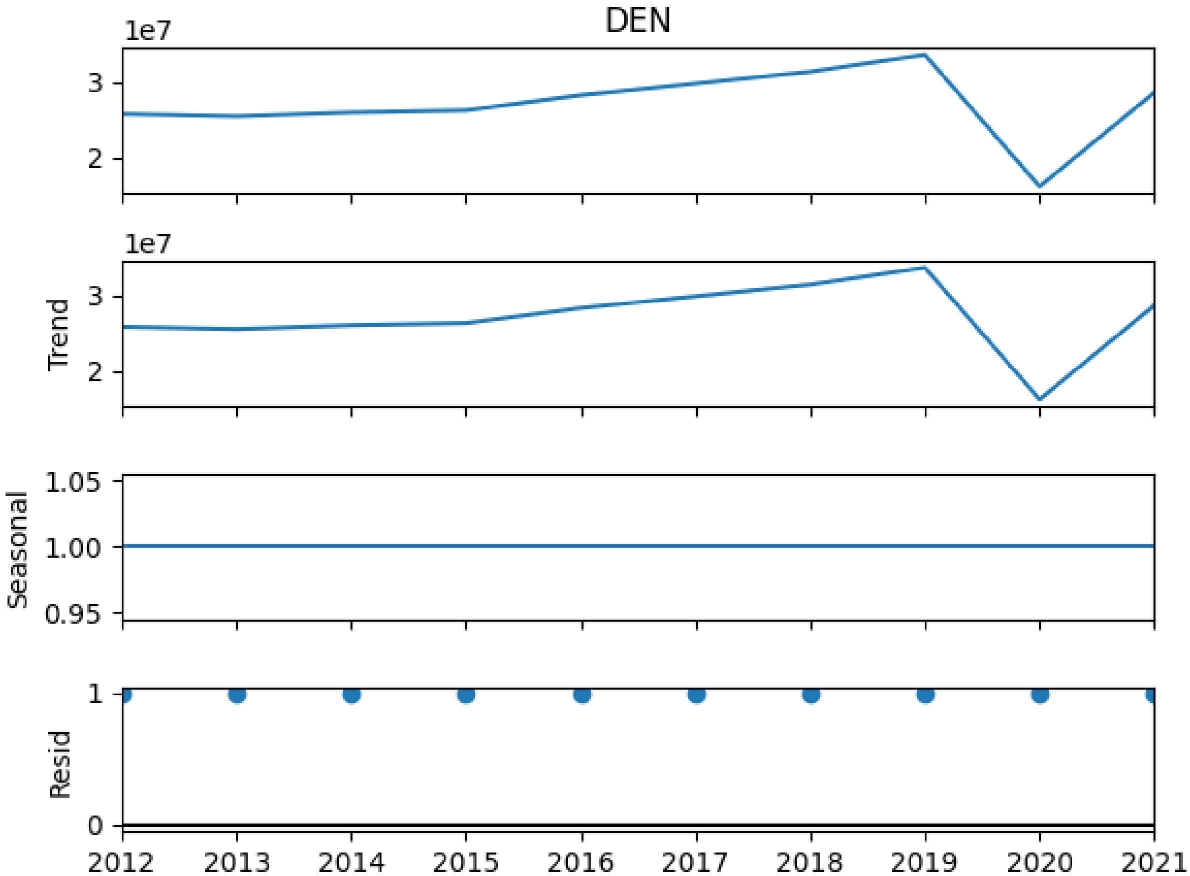


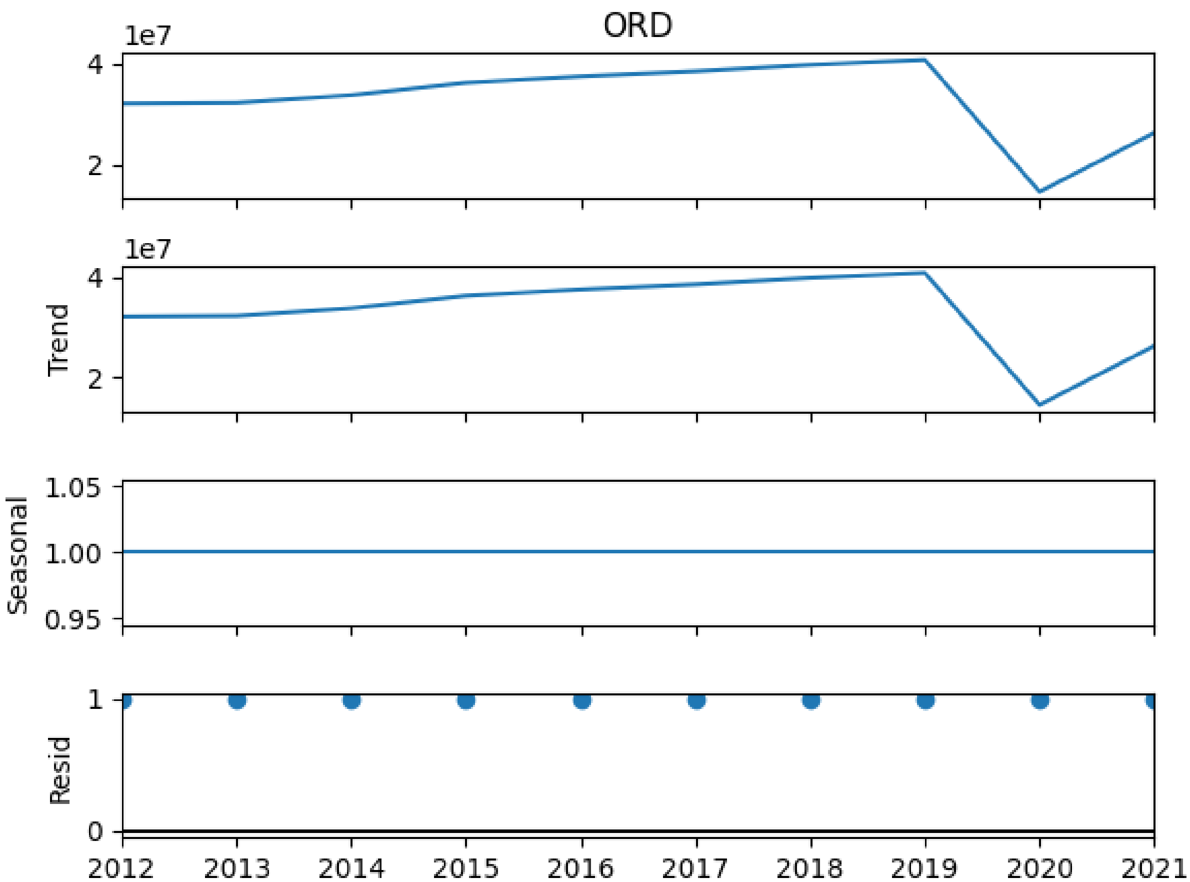

```
# Performing seasonal decomposition on first 4 airports.
for ser in time_series.columns[:4]:
    series = time_series[ser].copy()
    series.index =
```

```
pd.to_datetime(series.index.str.replace('data_',''))
    series.sort_index(inplace = True)
    decomposition = sm.tsa.seasonal_decompose(series,
model='multiplicative')
    decomposition.plot()
plt.show()
```




DFW
1e7
3
2
1e7
Trend
3
2
Seasonal
1.05
1.00
0.95
Resid
1
0
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021


DEN
1e7
3
2
1e7
Trend
3
2
Seasonal
1.05
1.00
0.95
Resid
1
0
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021




Observations:

- The **trend** for all four airports is that the traffic is on an uptrend after a major drop in 2019 due to COVID.
- The seasonality shows that there are no recurring patterns in the traffic data.
- The residual shows that there are no abnormnal patterns in the traffic data.

```
error = {}
forecast_2022 = {}
f = {}
wind_min = {}
win_min_mape = {}
for ser in time_series.columns:
    series = time_series[ser].copy()
    series.index = 
pd.to_datetime(series.index.str.replace('data_',''))
    series.sort_index(inplace = True)
    test = series[-1:]
    train = series[:-1]
    err_temp = {}
    fore_2022 = {}
    for window in range(2,10):
        forecast = series.rolling(window).mean()
```

```
        # accuracy
        mape = round(mean_absolute_percentage_error(test, forecast[-
1:]),4)
        err_temp.update({window : mape})
        # forecast for 2022
        fore_2022.update({window : series[-window:].mean()})
    err_ser = pd.Series(err_temp)
    min_wind = err_ser[(err_ser == err_ser.min())].index.values[0]
    forecast_2022.update({ser : round(series[-min_wind:].mean(),2)})
    wind_min.update({ser : min_wind})
    win_min_mape.update({ser :err_temp[min_wind] })
    f.update({ser :pd.Series(fore_2022).round(2) })
    error.update({ser : err_ser})

    # forecast for 2022

win_min_mape
```

```
{'ATL': 0.0065,
 'DFW': 0.0015,
 'DEN': 0.023,
 'ORD': 0.0351,
 'LAX': 0.1362,
 'CLT': 0.0006,
 'MCO': 0.0077,
 'LAS': 0.0212,
 'PHX': 0.0001,
 'MIA': 0.0182,
 'SEA': 0.0076,
 'IAH': 0.0389,
 'JFK': 0.1912,
 'EWR': 0.0486,
 'FLL': 0.0122,
 'MSP': 0.0502,
 'SFO': 0.1697,
 'DTW': 0.0559,
 'BOS': 0.1502,
 'SLC': 0.0062,
 'PHL': 0.0719,
 'BWI': 0.0082,
 'TPA': 0.0029,
 'SAN': 0.0686,
 'LGA': 0.1655,
 'MDW': 0.0453,
 'BNA': 0.0598,
 'IAD': 0.0595,
 'DCA': 0.0844,
 'AUS': 0.0017}
```

```
sma_forecast = pd.DataFrame(f)
sma_error = pd.DataFrame(error)

sma_prediction = pd.DataFrame(forecast_2022.values(), index =
forecast_2022.keys(), columns = ['forecast_2022'] )
sma_prediction['window_used'] = wind_min.values()
sma_prediction['mape_at_window'] = win_min_mape.values()

sma_prediction.sort_values('forecast_2022', ascending=False)
```

| | forecast_2022 | window_used | mape_at_window |
|---|---|---|---|
| ATL | 36913890.33 | 3 | 0.0065 |
| DFW | 30049928.50 | 6 | 0.0015 |
| DEN | 27986853.33 | 6 | 0.0230 |
| ORD | 27276077.67 | 3 | 0.0351 |
| LAX | 26886097.00 | 3 | 0.1362 |
| CLT | 20913675.38 | 8 | 0.0006 |
| LAS | 19566943.50 | 4 | 0.0212 |
| MCO | 19467356.50 | 8 | 0.0077 |
| PHX | 18942662.60 | 5 | 0.0001 |
| JFK | 18193272.00 | 3 | 0.1912 |
| SEA | 17298122.67 | 3 | 0.0076 |
| MIA | 17182193.50 | 4 | 0.0182 |
| IAH | 15610229.33 | 3 | 0.0389 |
| EWR | 15220095.33 | 3 | 0.0486 |
| FLL | 13765555.89 | 9 | 0.0122 |
| MSP | 12824682.00 | 3 | 0.0502 |
| BOS | 12548215.33 | 3 | 0.1502 |
| DTW | 12161020.00 | 3 | 0.0559 |
| SLC | 10729401.33 | 6 | 0.0062 |
| PHL | 10526616.67 | 3 | 0.0719 |
| SFO | 9735202.00 | 2 | 0.1697 |
| BWI | 9329867.67 | 3 | 0.0082 |
| LGA | 9122674.67 | 3 | 0.1655 |
| TPA | 8872731.89 | 9 | 0.0029 |
| SAN | 8374302.67 | 3 | 0.0686 |
| IAD | 7658216.67 | 3 | 0.0595 |
| MDW | 7333000.33 | 3 | 0.0453 |
| DCA | 7300226.67 | 3 | 0.0844 |
| BNA | 7140261.25 | 4 | 0.0598 |
| AUS | 6677268.60 | 5 | 0.0017 |

**As we saw in the line chart earlier, the airports: ATL, DFW, DEN,and ORD will witness the maximum traffic in 2022.**

1. Use hypothesis testing strategies to discover:

- If the airport's altitude has anything to do with flight delays for incoming and departing flights
- If the number of runways at an airport affects flight delays

- If the duration of a flight (length) affects flight delays Hint: Test this from the perspective of both the source and destination airports

---

- If the airport's altitude has anything to do with flight delays for incoming and departing flights

For outgoing flights.

```
# 2 sample t test for outgoing flights with the following hypothesis.
# H0 : avg elevation for Delayed flights - avg elevation for not
Delayed flights = 0
# Ha : avg elevation for Delayed flights - avg elevation for not
Delayed flights != 0

sample1 = cdt[cdt.Delay == 1].elevation_ft_source_airport
sample2 = cdt[cdt.Delay == 0].elevation_ft_source_airport

t, p = stats.ttest_ind(sample1, sample2)

if p < 0.05:
    result = 'reject null'
else :
    result = 'fail to reject null'

print(result)

reject null
```

There is a statistically significant difference in the average elevation between delayed and not delayed flights. This suggests that the **elevation of the source airport may play a role in flight delays**.

For incoming flights.

```
# 2 sample t test for incoming flights with the following hypothesis.
# H0 : avg elevation for Delayed flights - avg elevation for not
Delayed flights = 0
# Ha : avg elevation for Delayed flights - avg elevation for not
Delayed flights != 0

sample1 = cdt[cdt.Delay == 1].elevation_ft_dest_airport
sample2 = cdt[cdt.Delay == 0].elevation_ft_dest_airport

t, p = stats.ttest_ind(sample1, sample2)

if p < 0.05:
    result = 'reject null'
else :
    result = 'fail to reject null'

print(result)
```

```
reject null
```

There is a statistically significant difference in the average elevation between delayed and not delayed flights. This suggests that the **elevation of the destination airport may play a role in flight delays**.

- If the number of runways at an airport affects flight delays

```
# t test :
# H0 : avg runway count for delayed filghts - avg runway count for non
delayed flights => 0
# Ha : avg runway count for delayed filghts - avg runway count for non
delayed flights < 0

cdt.head()

   id Airline  Flight AirportFrom AirportTo  DayOfWeek  Time  Length
Delay  \
0   1      CO     269         SFO       IAH          3    15     205
1
1   2      US    1558         PHX       CLT          3    15     222
1
2   3      AA    2400         LAX       DFW          3    20     165
1
3   4      AA    2466         SFO       DFW          3    20     195
1
4   5      AS     108         ANC       SEA          3    30     202
0

  type_source_airport  elevation_ft_source_airport  \
0       large_airport                         13.0
1       large_airport                       1135.0
2       large_airport                        125.0
3       large_airport                         13.0
4       large_airport                        152.0

   runway_count_source_airport type_dest_airport
elevation_ft_dest_airport  \
0                           4.0     large_airport
97.0
1                           3.0     large_airport
748.0
2                           4.0     large_airport
607.0
3                           4.0     large_airport
607.0
4                           3.0     large_airport
433.0

   runway_count_dest_airport  data_2021_source_airport  \
```

```
0                         5.0                11725347.0
1                         4.0                18940287.0
2                         7.0                23663410.0
3                         7.0                11725347.0
4                         4.0                 2184959.0

   data_2021_dest_airport  Founded duration
0              16242821.0   1931.0    short
1              20900875.0   1967.0   medium
2              30005266.0   1926.0    short
3              30005266.0   1926.0    short
4              17430195.0   1932.0    short
```

```
s1 = cdt[cdt.Delay == 1].runway_count_source_airport
s2 = cdt[cdt.Delay == 0].runway_count_source_airport

t, p = stats.ttest_ind(s1, s2)
if p < 0.05:
    result = 'reject null'
else :
    result = 'fail to reject null'
print(result)
```

```
reject null
```

```
s1 = cdt[cdt.Delay == 1].runway_count_dest_airport
s2 = cdt[cdt.Delay == 0].runway_count_dest_airport

t, p = stats.ttest_ind(s1, s2)
if p < 0.05:
    result = 'reject null'
else :
    result = 'fail to reject null'
print(result)
```

```
reject null
```

The resulting output "reject null" for both tests suggests that there is evidence to support the alternative hypothesis, indicating that **the average runway count for delayed flights is significantly lower than the average runway count for non-delayed flights in both the source and destination airports.**

- If the duration of a flight (length) affects flight delays Hint: Test this from the perspective of both the source and destination airports

```
# t test :
# H0 : avg duration for delayed filghts - avg duration for non delayed
flights = 0
# Ha : avg duration for delayed filghts - avg duration for non delayed
flights != 0
```

```
s1 = cdt[cdt.Delay == 1].Length
s2 = cdt[cdt.Delay == 0].Length

t, p = stats.ttest_ind(s1, s2)

if p < 0.05:
    result = 'reject null'
else :
    result = 'fail to reject null'
print(result)
```

```
reject null
```

The result is 'reject null', indicating that there is a **significant difference in the average duration between delayed and non-delayed flights**.

```
cs = pd.crosstab(cdt.duration, cdt.Delay)
cs
```

| Delay | 0 | 1 |
|---|---|---|
| duration | | |
| short | 253865 | 203612 |
| medium | 28983 | 29198 |
| long | 252 | 307 |

```
chi, p, df, ex = stats.chi2_contingency(cs)
if p < 0.05:
    result = 'reject null'
else :
    result = 'fail to reject null'
print(result)
```

```
reject null
```

The result of the chi-square test was "reject null," it means that there is evidence to suggest a **significant relationship between the duration of flights and flight delays**.

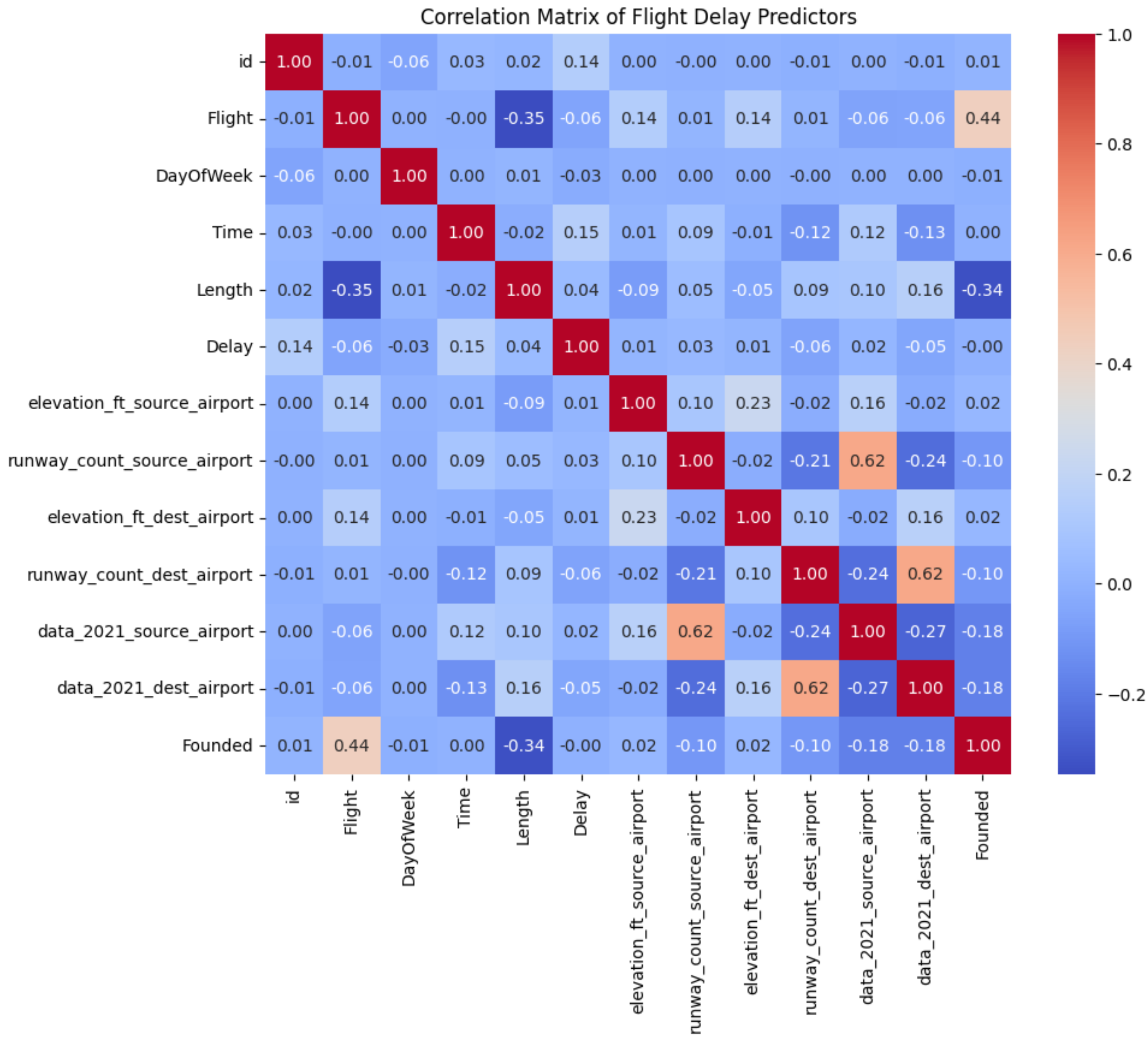
```
# t test :
# H0 : avg duration for delayed filghts - avg duration for non delayed flights <= 0
# Ha : avg duration for delayed filghts - avg duration for non delayed flights > 0

t, p = stats.ttest_ind(s1, s2)
if p < 0.05:
    result = 'reject null'
else :
    result = 'fail to reject null'
print(result)
```

```
reject null
```

The result of the t-test was "reject null," it means that there is evidence to suggest a significant difference in the average duration between delayed flights and non-delayed flights. Specifically, **the average duration of delayed flights is significantly greater than the average duration of non-delayed flights.**

- Find the correlation matrix between the flight delay predictors, create a heatmap to visualize this, and share your findings

```
cdt.columns

Index(['id', 'Airline', 'Flight', 'AirportFrom', 'AirportTo',
'DayOfWeek',
       'Time', 'Length', 'Delay', 'type_source_airport',
       'elevation_ft_source_airport', 'runway_count_source_airport',
       'type_dest_airport', 'elevation_ft_dest_airport',
       'runway_count_dest_airport', 'data_2021_source_airport',
       'data_2021_dest_airport', 'Founded', 'duration'],
      dtype='object')

numeric_columns = cdt.select_dtypes(include=['float64',
'int64']).columns
correlation_with_delay = cdt[numeric_columns].corr()['Delay']

correlation_with_delay

id                             0.140434
Flight                        -0.057371
DayOfWeek                     -0.025832
Time                           0.149801
Length                         0.040162
Delay                          1.000000
elevation_ft_source_airport    0.012551
runway_count_source_airport    0.029099
elevation_ft_dest_airport      0.013180
runway_count_dest_airport     -0.061431
data_2021_source_airport       0.020315
data_2021_dest_airport        -0.051622
Founded                       -0.003102
Name: Delay, dtype: float64

# Calculate the correlation matrix
correlation_matrix = cdt[numeric_columns].corr()

# Create heatmap with correlation values
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, cmap='coolwarm', annot=True,
fmt=".2f")
plt.title('Correlation Matrix of Flight Delay Predictors')
plt.show()
```




Observations:

- The **'Time'** variable has the highest positive correlation with 'Delay' (0.149801), indicating that flights scheduled for later times may have a higher likelihood of being delayed.
- The **'Length'** variable has a slightly positive correlation with 'Delay' (0.040162), suggesting that longer flights may be associated with a slightly higher chance of delays.
- The **'Flight'** variable shows a weak negative correlation with 'Delay' (-0.057371), implying that certain flight numbers may be associated with a lower likelihood of delays.
- The **'runway_count_dest_airport'** variable has a moderate negative correlation with 'Delay' (-0.061431), indicating that airports with a higher number of runways at the destination may be associated with a lower probability of delays.
- Other variables such as 'DayOfWeek', 'elevation_ft_source_airport', 'elevation_ft_dest_airport', 'data_2021_source_airport', 'data_2021_dest_airport', and 'Founded' show weak correlations with 'Delay'.

Exporting the final dataset to csv for future use.

```
cdt.to_csv('usairlinesfinaldata.csv', index = False)
```

Machine Learning

- Use OneHotEncoder and OrdinalEncoder to deal with categorical variables

```
cdt.isna().sum()

id                            0
Airline                       0
Flight                        0
AirportFrom                   0
AirportTo                     0
DayOfWeek                     0
Time                          0
Length                        0
Delay                         0
type_source_airport           0
elevation_ft_source_airport   0
runway_count_source_airport   0
type_dest_airport             0
elevation_ft_dest_airport     0
runway_count_dest_airport     0
data_2021_source_airport      0
data_2021_dest_airport        0
Founded                       0
duration                      0
dtype: int64

cdt.drop(columns = ['id', 'Flight','duration'], inplace = True)

cdt.head()

  Airline AirportFrom AirportTo  DayOfWeek  Time  Length  Delay  \
0      CO         SFO       IAH          3    15     205      1
1      US         PHX       CLT          3    15     222      1
2      AA         LAX       DFW          3    20     165      1
3      AA         SFO       DFW          3    20     195      1
4      AS         ANC       SEA          3    30     202      0

  type_source_airport  elevation_ft_source_airport  \
0       large_airport                         13.0
1       large_airport                       1135.0
2       large_airport                        125.0
3       large_airport                         13.0
4       large_airport                        152.0

   runway_count_source_airport type_dest_airport
elevation_ft_dest_airport  \
0                          4.0     large_airport
```

```
97.0
1                          3.0     large_airport
748.0
2                          4.0     large_airport
607.0
3                          4.0     large_airport
607.0
4                          3.0     large_airport
433.0

   runway_count_dest_airport  data_2021_source_airport  \
0                        5.0                11725347.0
1                        4.0                18940287.0
2                        7.0                23663410.0
3                        7.0                11725347.0
4                        4.0                 2184959.0

   data_2021_dest_airport  Founded
0              16242821.0   1931.0
1              20900875.0   1967.0
2              30005266.0   1926.0
3              30005266.0   1926.0
4              17430195.0   1932.0
```

```
cdt.type_dest_airport.unique()
```

```
array(['large_airport', 'medium_airport'], dtype=object)
```

```
ordinal = OrdinalEncoder(categories=[['medium_airport',
'large_airport'],['medium_airport', 'large_airport']])
ordinal.fit(cdt[['type_source_airport', 'type_dest_airport']])
```

```
OrdinalEncoder(categories=[['medium_airport', 'large_airport'],
                           ['medium_airport', 'large_airport']])
```

```
cdt[['type_source_airport', 'type_dest_airport']] =
ordinal.transform(cdt[['type_source_airport', 'type_dest_airport']])
```

```
model_data = cdt.drop(columns = ['Airline', 'AirportFrom',
'AirportTo'])
```

```
model_data.shape
```

```
(516217, 13)
```

```
dummy = pd.get_dummies(model_data)
dummy.shape
```

```
(516217, 13)
```

```
dummy.Founded = 2023 - dummy.Founded
dummy.head()
```

```
   DayOfWeek  Time  Length  Delay  type_source_airport  \
0          3    15     205      1                  1.0
1          3    15     222      1                  1.0
2          3    20     165      1                  1.0
3          3    20     195      1                  1.0
4          3    30     202      0                  1.0

   elevation_ft_source_airport  runway_count_source_airport  \
0                         13.0                          4.0
1                       1135.0                          3.0
2                        125.0                          4.0
3                         13.0                          4.0
4                        152.0                          3.0

   type_dest_airport  elevation_ft_dest_airport
runway_count_dest_airport  \
0                1.0                       97.0
5.0
1                1.0                      748.0
4.0
2                1.0                      607.0
7.0
3                1.0                      607.0
7.0
4                1.0                      433.0
4.0

   data_2021_source_airport  data_2021_dest_airport  Founded
0                11725347.0              16242821.0     92.0
1                18940287.0              20900875.0     56.0
2                23663410.0              30005266.0     97.0
3                11725347.0              30005266.0     97.0
4                 2184959.0              17430195.0     91.0
```

Perform the following model building steps:

- Apply logistic regression (use stochastic gradient descent optimizer) and decision tree models
- Use the stratified five fold method to build and validate the models **Note:** Make sure you use standardization effectively, ensuring no data leakage and leverage pipelines to have a cleaner code
- Use RandomizedSearchCV for hyperparameter tuning, and use k fold for cross validation
- Keep a few data points (10%) for prediction purposes to evaluate how you would make the final prediction, and do not use t his data for testing or validation **Note:** The final prediction will be based on the voting (majority class by 5 models created using the stratified 5 fold method)
- Compare the results of logistic regression and decision tree classifier

```
model_data.reset_index(drop = True, inplace = True)
```

```
# Generating a randon set of indices.

np.random.seed(12)
deploy_idx = np.random.choice(model_data.index, replace = False, size
= 5000)

deploy = model_data.loc[deploy_idx]

X_deploy = deploy.drop(columns = 'Delay')

model_dev = model_data.loc[~model_data.index.isin(deploy.index)]

deploy.reset_index(drop = True, inplace = True)
model_dev.reset_index(drop = True, inplace = True)

X = model_dev.drop(columns = 'Delay')
y = model_dev.Delay

folds = StratifiedKFold(n_splits=5, shuffle = True, random_state=12)
accuracy_train = {}
accuracy_test = {}
final_predictions_sgd = {}
i = 1
for train_index, test_index in folds.split(X,y):
    print('iter ', i)
    train, test = model_dev.loc[train_index,],
model_dev.loc[test_index,]
    sc = StandardScaler()
    sgd = SGDClassifier()

    # define search space

    space = dict()
    space['sgd__penalty'] = ['l1', 'l2', 'elasticnet']
    space['sgd__l1_ratio'] = [0,.1,.2,.8,1]
    space['sgd__alpha'] = [1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 10, 100,
1000,10000]
    space['sgd__learning_rate'] = ['constant', 'adaptive']
    space['sgd__eta0']=[1e-5, 1e-4, 1e-3, 1e-2, 1e-1 , 2e-1, 3e-1, 5e-
1, 8e-1, 4e-1, 8e-1, 1, 10, 100]

    pipe = Pipeline([('sc',sc), ('sgd', sgd)])

    # define search
    search = RandomizedSearchCV( pipe, space, scoring='accuracy',
                                cv=5, refit=True, return_train_score =
True,
                                random_state = 12, n_jobs = -1, n_iter
= 2
                           )
```

```
    # execute search
    X_train = train.drop(columns = 'Delay')
    y_train = train.Delay

    result = search.fit(X_train, y_train)

    train_pred = result.predict(X_train)

    X_test = test.drop(columns = 'Delay')
    y_test = test.Delay
    test_pred = result.predict(X_test)

final_predictions_sgd.update({'Fold{}'.format(i):result.predict(X_depl
oy)})

    # get rmse for each fold for train data
    accuracy_train.update({'Fold{}'.format(i):
round(accuracy_score(y_true = y_train, y_pred = train_pred)*100,3)})
    accuracy_test.update({'Fold{}'.format(i):
round(accuracy_score(y_true = y_test, y_pred = test_pred) * 100,3)})
    i += 1
```

```
iter  1
iter  2
iter  3
iter  4
iter  5
```

```
folds = StratifiedKFold(n_splits=5, shuffle = True, random_state=12)
dt_accuracy_train = {}
dt_accuracy_test = {}
final_predictions_dt = {}
i = 1
for train_index, test_index in folds.split(X,y):
    print('iter ', i)

    train, test = model_dev.loc[train_index,],
model_dev.loc[test_index,]

    sc = StandardScaler()
    dt = DecisionTreeClassifier()


    # define search space
    space = dict()
    space['dt__min_samples_split'] = [25000, 30000, 35000, 40000,
45000, 50000, 60000 ]
    space['dt__min_samples_leaf'] = [10000, 15000, 20000]

    pipe = Pipeline([('sc',sc), ('dt', dt)])
```

```
    # define search
    search = RandomizedSearchCV( pipe, space, scoring='accuracy',
                                cv=5, refit=True, return_train_score =
True,
                                random_state = 12, n_jobs = -1, n_iter
= 2
                           )

    # execute search
    X_train = train.drop(columns = 'Delay')
    y_train = train.Delay

    result = search.fit(X_train, y_train)

    train_pred = result.predict(X_train)

    X_test = test.drop(columns = 'Delay')
    y_test = test.Delay
    test_pred = result.predict(X_test)

final_predictions_dt.update({'Fold{}'.format(i):result.predict(X_deplo
y)})

    # get rmse for each fold for train data
    dt_accuracy_train.update({'Fold{}'.format(i):
round(accuracy_score(y_true = y_train, y_pred = train_pred)*100,3)})
    dt_accuracy_test.update({'Fold{}'.format(i):
round(accuracy_score(y_true = y_test, y_pred = test_pred) * 100,3)})
    i += 1
```

```
iter  1
iter  2
iter  3
iter  4
iter  5
```

```
# compare results :
train_results = pd.DataFrame ({'sgd' : accuracy_train.values(), 'dt':
dt_accuracy_train.values() },
                             index = ['Fold {}'.format(i) for i in
range(1,6)])
train_results
```

| | sgd | dt |
|---|---|---|
| Fold 1 | 57.163 | 61.643 |
| Fold 2 | 57.168 | 61.669 |
| Fold 3 | 57.154 | 61.649 |
| Fold 4 | 57.228 | 61.487 |
| Fold 5 | 57.105 | 61.597 |

```
test_results = pd.DataFrame ({'sgd' : accuracy_test.values(), 'dt':
dt_accuracy_test.values() },
                             index = ['Fold {}'.format(i) for i in
range(1,6)])
test_results
```

| | sgd | dt |
|---|---|---|
| Fold 1 | 57.173 | 61.431 |
| Fold 2 | 57.182 | 61.304 |
| Fold 3 | 57.221 | 61.928 |
| Fold 4 | 56.891 | 61.421 |
| Fold 5 | 57.313 | 61.456 |

```
final_predictions_dt
```

```
{'Fold1': array([0, 0, 0, ..., 0, 0, 0]),
 'Fold2': array([0, 0, 0, ..., 0, 0, 0]),
 'Fold3': array([0, 1, 0, ..., 0, 1, 0]),
 'Fold4': array([0, 1, 1, ..., 0, 1, 0]),
 'Fold5': array([0, 1, 0, ..., 0, 1, 0])}
```

```
final_predictions_sgd
```

```
{'Fold1': array([0, 1, 0, ..., 1, 1, 0]),
 'Fold2': array([0, 1, 0, ..., 1, 1, 0]),
 'Fold3': array([0, 1, 0, ..., 1, 1, 0]),
 'Fold4': array([0, 1, 0, ..., 1, 1, 0]),
 'Fold5': array([0, 1, 0, ..., 1, 1, 0])}
```

```
folds = StratifiedKFold(n_splits=5, shuffle = True, random_state=12)
xgb_accuracy_train = {}
xgb_accuracy_test = {}
final_predictions_xgb = []

i = 1
for train_index, test_index in folds.split(X,y):
    print('iter ', i)
    train, test = model_dev.loc[train_index,],
model_dev.loc[test_index,]
    sc = StandardScaler()
    xgb_r = XGBClassifier(random_state = 12, use_label_encoder =
False)

    # define search space
    space = dict()
    space['xgb_r__n_estimators'] = [40,50,60]
    space['xgb_r__max_depth'] = [3,4,5]
    space['xgb_r__colsample_bytree']:[0.4,.5,.6]
    space['xgb_r__lambda'] = [.0001,.002,.0004,.0003]
    space['xgb_r__alpha'] = [.01,.02,.1,.4]
```

```
    pipe = Pipeline([('sc',sc), ('xgb_r', xgb_r)])

    # define search
    search = RandomizedSearchCV( pipe, space,
scoring='neg_root_mean_squared_error',
                                cv=5, refit=True, return_train_score =
True,
                                random_state = 12, n_jobs = -1, n_iter
= 2
                           )

    # execute search
    X_train = train.drop(columns = 'Delay')
    y_train = train.Delay

    result = search.fit(X_train, y_train)

    train_pred = result.predict(X_train)

    X_test = test.drop(columns = 'Delay')
    y_test = test.Delay
    test_pred = result.predict(X_test)

    final_predictions_xgb.append(result.predict(X_deploy))

    # get rmse for each fold for train data
    xgb_accuracy_train.update({'Fold{}'.format(i):
round(accuracy_score(y_true = y_train, y_pred = train_pred),3)})
    xgb_accuracy_test.update({'Fold{}'.format(i):
round(accuracy_score(y_true = y_test, y_pred = test_pred),3)})
    i += 1
```

```
iter  1
iter  2
iter  3
iter  4
iter  5
```

```
xgb_accuracy_train
```

```
{'Fold1': 0.646,
 'Fold2': 0.645,
 'Fold3': 0.645,
 'Fold4': 0.646,
 'Fold5': 0.647}
```

```
xgb_accuracy_train.values()
```

```
dict_values([0.646, 0.645, 0.645, 0.646, 0.647])
```

```
train_results['xgb'] = xgb_accuracy_train.values()
test_results['xgb'] = xgb_accuracy_test.values()

train_results
```

| | sgd | dt | xgb |
|---|---|---|---|
| Fold 1 | 57.163 | 61.643 | 0.646 |
| Fold 2 | 57.168 | 61.669 | 0.645 |
| Fold 3 | 57.154 | 61.649 | 0.645 |
| Fold 4 | 57.228 | 61.487 | 0.646 |
| Fold 5 | 57.105 | 61.597 | 0.647 |

```
test_results
```

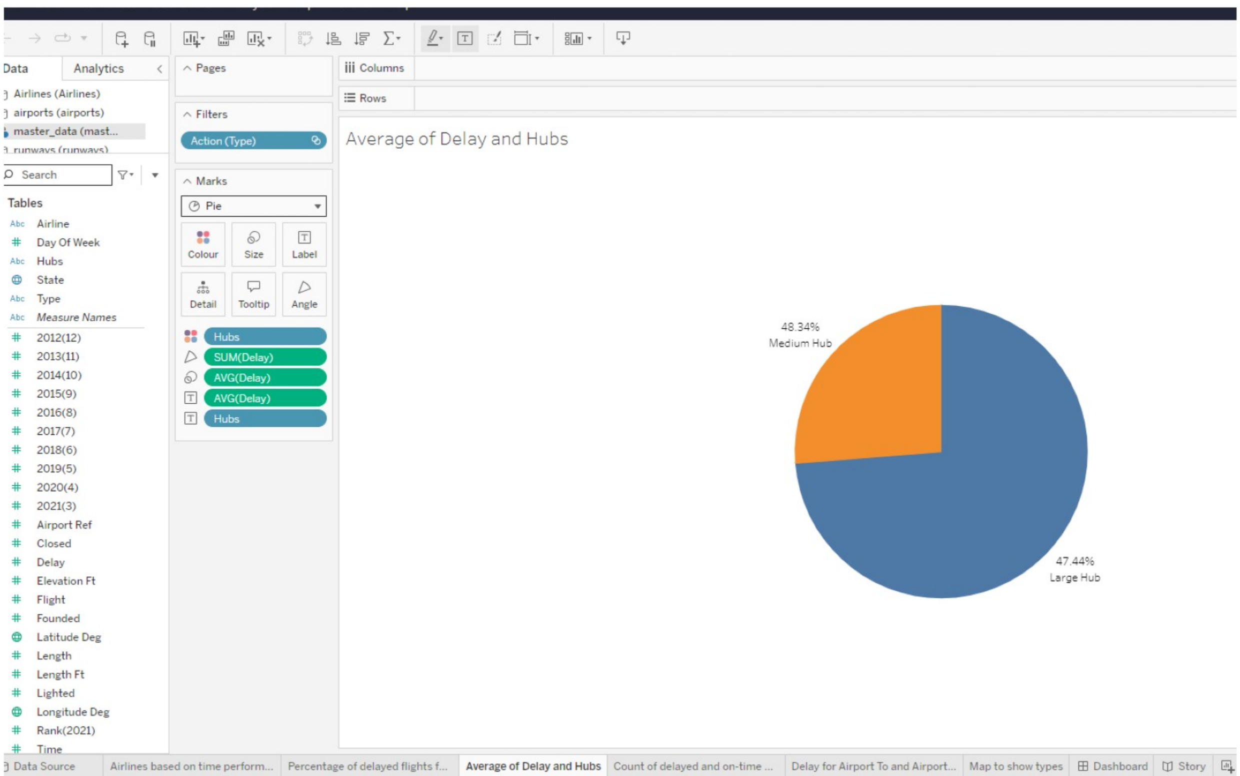
| | sgd | dt | xgb |
|---|---|---|---|
| Fold 1 | 57.173 | 61.431 | 0.642 |
| Fold 2 | 57.182 | 61.304 | 0.642 |
| Fold 3 | 57.221 | 61.928 | 0.646 |
| Fold 4 | 56.891 | 61.421 | 0.642 |
| Fold 5 | 57.313 | 61.456 | 0.646 |

**The logistic regression and decision tree models perform similarly and achieve higher accuracy compared to the XGBoost model.**

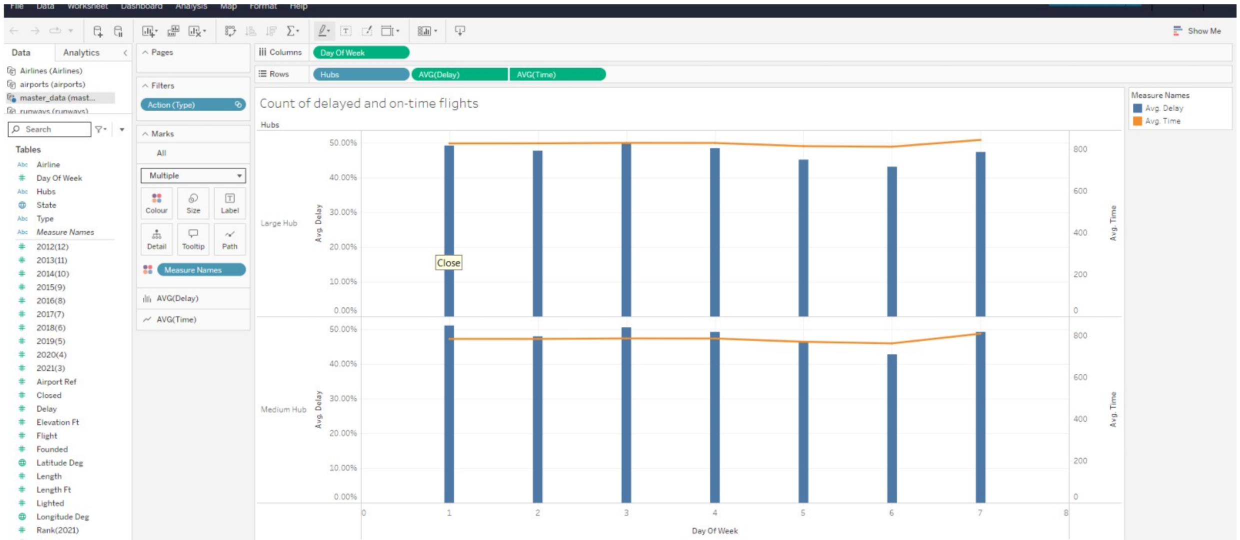
## Tableau Story

```
import os
from matplotlib import pyplot as plt
from matplotlib.image import imread

# Path to the folder containing images
folder_path = '/content/'

# List all files in the folder
files = os.listdir(folder_path)

# Filter out non-image files
image_files = [f for f in files if f.lower().endswith(('.png', '.jpg',
'.jpeg'))]

# Display each image
for image_file in image_files:
    image_path = os.path.join(folder_path, image_file)
    image = imread(image_path)
    plt.figure(figsize=(20,20))
    plt.imshow(image)
    plt.axis('off')
    plt.show()
```

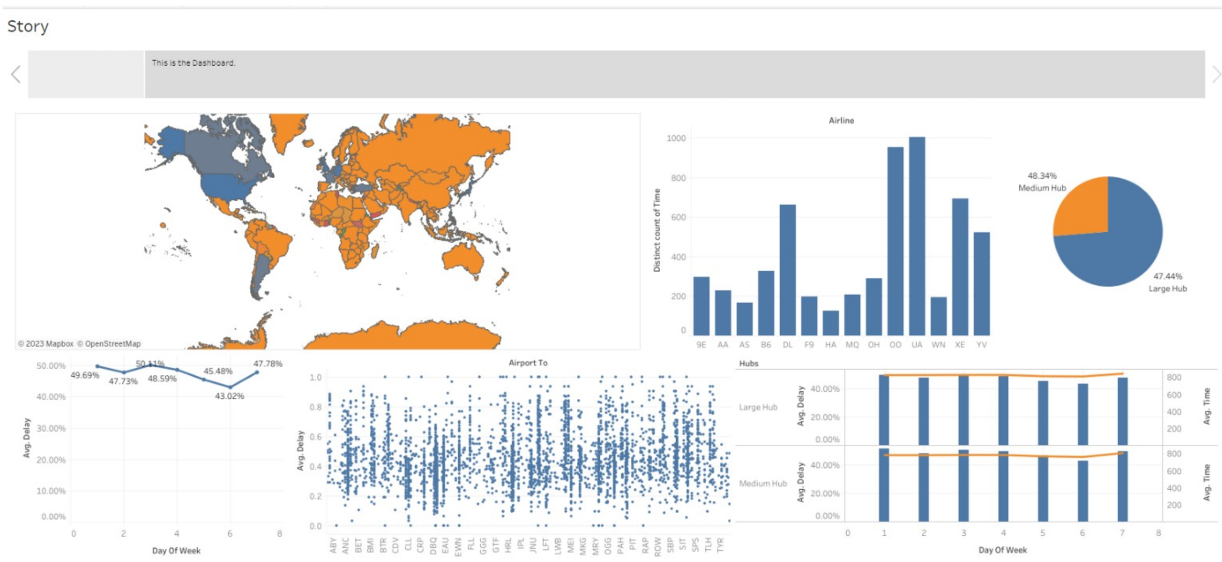
Dashboard
Layout
Device Preview
Sheets
Airlines based on time p...
Percentage of delayed fl...
Average of Delay and H...
Count of delayed and o...
Delay for Airport To and...
Map to show types
Objects
Horizontal Container
Vertical Container
Text
Extension
Ask Data
Image
Blank
Workflow
Web Page
Navigation
Download
Tiled
Floating
Show dashboard title
Airline
48.34%
Medium Hub
47.44%
Large Hub
Airport To
Day Of Week
Avg. Delay
Hubs
Large Hub
Medium Hub
Avg. Time

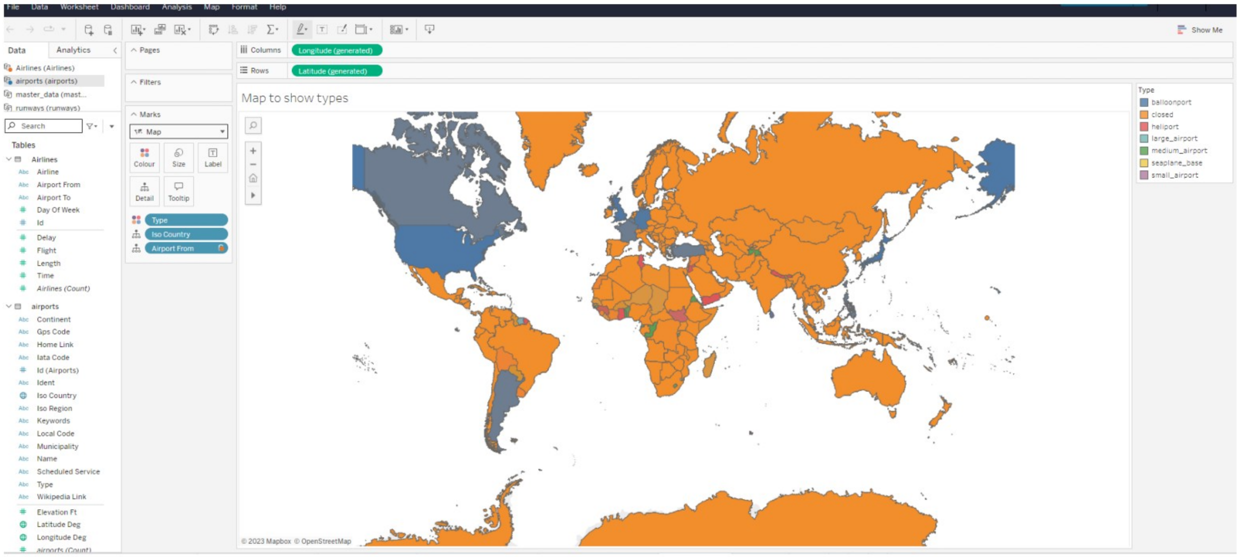
Data
Analytics
Pages
Columns
Rows
Airlines (Airlines)
airports (airports)
master_data (mast...
runways (runways)
Filters
Action (Type)
Search
Tables
Airline
Day Of Week
Hubs
State
Type
Measure Names
2012(12)
2013(11)
2014(10)
2015(9)
2016(8)
2017(7)
2018(6)
2019(5)
2020(4)
2021(3)
Airport Ref
Closed
Delay
Elevation Ft
Flight
Founded
Latitude Deg
Length
Length Ft
Lighted
Longitude Deg
Rank(2021)
Time
Marks
Pie
Colour
Size
Label
Detail
Tooltip
Angle
Hubs
SUM(Delay)
AVG(Delay)
AVG(Delay)
Hubs
Average of Delay and Hubs
48.34%
Medium Hub
47.44%
Large Hub
Data Source
Airlines based on time perform...
Percentage of delayed flights f...
Average of Delay and Hubs
Count of delayed and on-time ...
Delay for Airport To and Airport...
Map to show types
Dashboard
Story

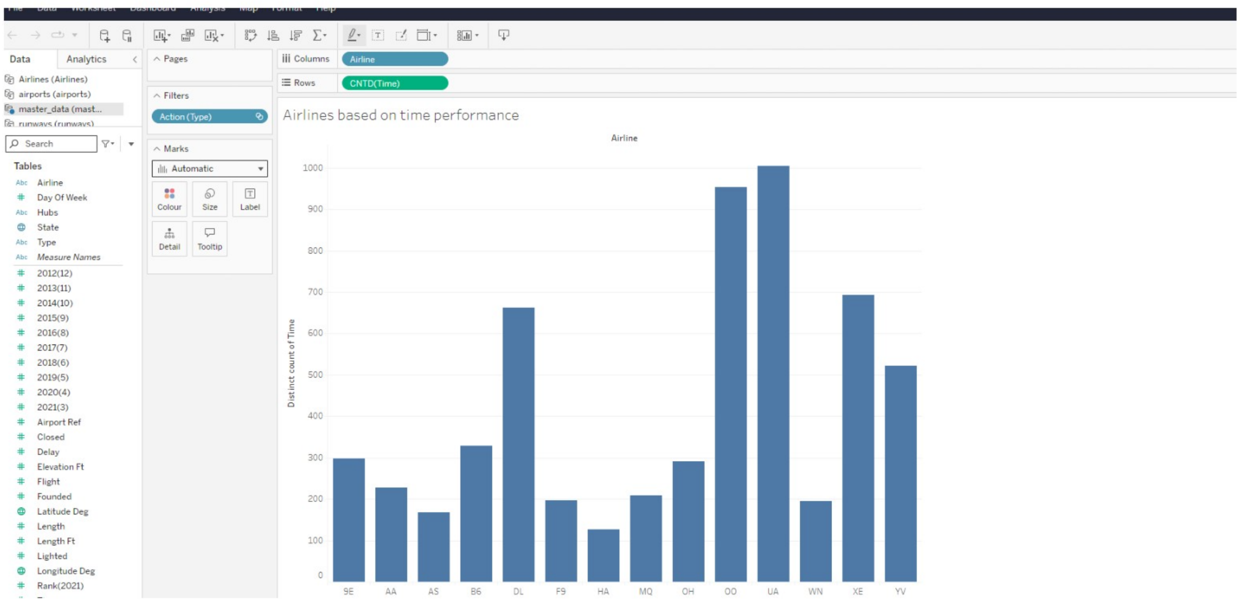


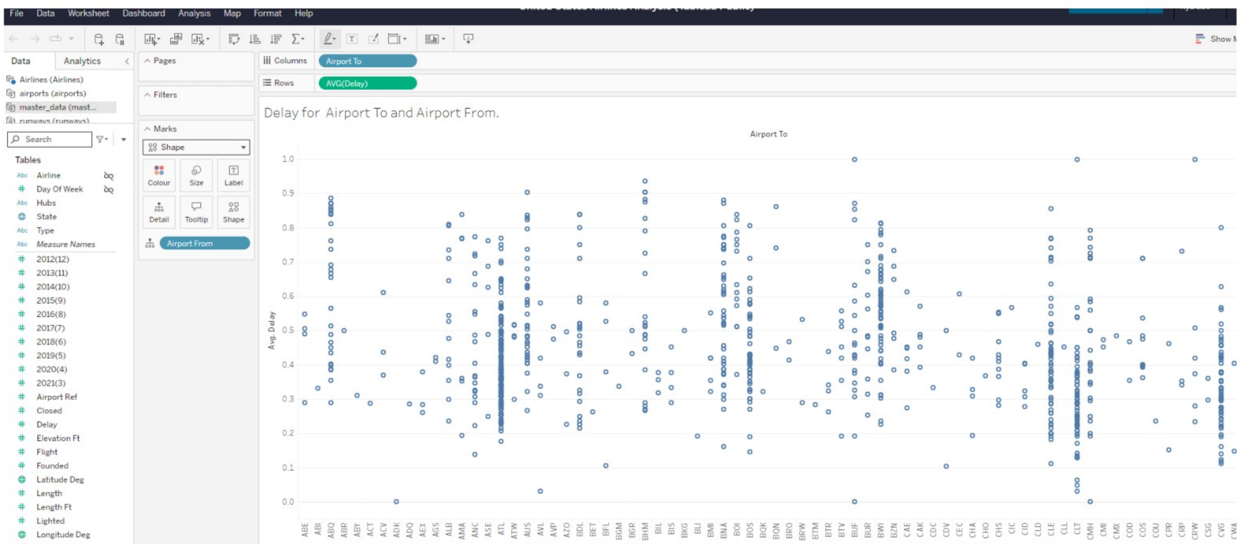
Count of delayed and on-time flights
Hubs
Large Hub
Medium Hub
Avg. Delay
Avg. Time
Day Of Week
Measure Names
Columns
Rows
Filters
Action (Type)
Marks
Close

Story
This is the Dashboard.
© 2023 Mapbox © OpenStreetMap
Airline
Distinct count of Time
48.34%
Medium Hub
47.44%
Large Hub
Airport To
Avg. Delay
Day Of Week
Hubs
Avg. Time


Columns Longitude (generated)
Rows Latitude (generated)
Map to show types
Type
balloonport
closed
heliport
large_airport
medium_airport
seaplane_base
small_airport
© 2023 Mapbox © OpenStreetMap


Columns Airline
Rows CNTD(Time)
Filters Action (Type)
Airlines based on time performance
Airline
Distinct count of Time
0
100
200
300
400
500
600
700
800
900
1000
9E
AA
AS
B6
DL
F9
HA
MQ
OH
OO
UA
WN
XE
YV


# Tableau Insights and Findings

# Airlines Data:

**Airline Distribution:** There are a total of 17 unique airlines in the dataset. The airline with the most flights is "WN" (Southwest Airlines), which appears 94,097 times.

**Flight Length:** The mean flight length is approximately 2,499.38 miles, with a minimum of 1 mile and a maximum of 7,814 miles.

**Day of the Week:** Flights are distributed over the days of the week, with an average of 3.93 flights per day. The highest number of flights occurs on Day 7.

**Time of Day:** The mean flight time is approximately 801.51 minutes, ranging from a minimum of 10 minutes to a maximum of 1,439 minutes.

**Flight Length:** The mean flight length is approximately 132.22 miles, with values ranging from 0 to 655 miles.

**Flight Delay:** The mean flight delay is approximately 0.45, indicating that, on average, flights are not delayed. This is further supported by the fact that 75% of flights have a delay value of 1 or less.

## Airports Data:

**Airport Types:** There are 7 unique types of airports. The most common type is "small_airport," which appears 37,676 times.

**Elevation Distribution:** Airport elevations range from -1,266 feet (possibly below sea level) to 17,372 feet, indicating that airports are situated at varying altitudes.

**Continent Distribution:** Airports are located on 6 different continents. The most common continent is Asia (AS), with 10,138 airports.

**Country Distribution:** Airports are located in 243 unique countries, with the United States (US) having the highest number of airports (28,510).

**Municipality Distribution:** Airports are located in 32,444 unique municipalities, with "Osaka" being the most common, appearing 402 times.

**Scheduled Service:** Most airports in the dataset (67,034 out of 71,076) do not have scheduled service.

**GPS Code Distribution:** There are 40,446 unique GPS codes, with some appearing multiple times. The most common GPS code is "SDPS," which appears 3 times.

**IATA Code Distribution:** There are 8,820 unique IATA codes, with each code representing a different airport.

**Local Code Distribution:** There are 30,277 unique local codes, with some appearing multiple times.

**Home Link Distribution:** There are 3,360 unique home links, with some appearing multiple times. The most common home link appears 4 times.

**Wikipedia Link Distribution:** There are 10,264 unique Wikipedia links, with some appearing multiple times. The most common Wikipedia link appears 14 times.

**Keywords Distribution:** There are 13,140 unique keywords, with some appearing multiple times. The most common keyword is "Mukho," which appears 47 times.

## Airport Details Data:

**Airport Reference Distribution:** Airport references range from 2 to 430,661.

**Airport Identifier Distribution:** There are 36,025 unique airport identifiers, with "KORD" being the most common, appearing 11 times.

**Runway Length Distribution:** Runway lengths range from 0 to 30,000 feet.

**Runway Width Distribution:** Runway widths range from 0 to 9,000 feet.

**Surface Type Distribution:** There are 592 unique surface types, with "ASP" being the most common, appearing 10,702 times.

**Lighted Runway Distribution:** Most runways (42,390 out of 42,390) are lighted.

**Closed Runway Distribution:** Only a small percentage of runways (16.70%) are marked as closed.

**Left Runway Identifier Distribution:** There are 266 unique left runway identifiers, with "H1" being the most common, appearing 5,958 times.

**Left Runway Latitude Distribution:** Left runway latitudes range from -75.60 to 82.51 degrees.

**Left Runway Longitude Distribution:** Left runway longitudes range from -178.30 to 179.34 degrees.

**Left Runway Elevation Distribution:** Left runway elevations range from -1,246 to 13,202 feet.

**Left Runway Heading Distribution:** Left runway headings range from 0 to 360 degrees.

# Detailed Insights and Findings

## Airline Distribution

- There are a total of 17 unique airlines in the dataset.
- The airline with the most flights is "WN" (Southwest Airlines), which appears 94,097 times.

## Flight Length

- The mean flight length is approximately 2,499.38 miles, with a minimum of 1 mile and a maximum of 7,814 miles.

## Day of the Week

- Flights are distributed over the days of the week, with an average of 3.93 flights per day.
- The highest number of flights occurs on Day 7.

## Time of Day

- The mean flight time is approximately 801.51 minutes, ranging from a minimum of 10 minutes to a maximum of 1,439 minutes.

## Flight Length (Again)

- The mean flight length is approximately 132.22 miles, with values ranging from 0 to 655 miles.

## Flight Delay

- The mean flight delay is approximately 0.45, indicating that, on average, flights are not delayed.
- 75% of flights have a delay value of 1 or less.

# Airports Data Overview

## Airport Types

- There are 7 unique types of airports.
- The most common type is "small_airport," which appears 37,676 times.

## Elevation Distribution

- Airport elevations range from -1,266 feet (possibly below sea level) to 17,372 feet, indicating that airports are situated at varying altitudes.

## Continent Distribution

- Airports are located on 6 different continents.
- The most common continent is Asia (AS), with 10,138 airports.

## Country Distribution

- Airports are located in 243 unique countries.
- The United States (US) has the highest number of airports (28,510).

## Municipality Distribution

- Airports are located in 32,444 unique municipalities.
- "Osaka" is the most common, appearing 402 times.

## Scheduled Service

- Most airports in the dataset (67,034 out of 71,076) do not have scheduled service.

## GPS Code Distribution

- There are 40,446 unique GPS codes, with some appearing multiple times.
- The most common GPS code is "SDPS," which appears 3 times.

## IATA Code Distribution

- There are 8,820 unique IATA codes, with each code representing a different airport.

## Local Code Distribution

- There are 30,277 unique local codes, with some appearing multiple times.

## Home Link Distribution

- There are 3,360 unique home links, with some appearing multiple times.
- The most common home link appears 4 times.

## Wikipedia Link Distribution

- There are 10,264 unique Wikipedia links, with some appearing multiple times.
- The most common Wikipedia link appears 14 times.

## Keywords Distribution

- There are 13,140 unique keywords, with some appearing multiple times.
- The most common keyword is "Mukho," which appears 47 times.

# Airport Details Data Overview

## Airport Reference Distribution

- Airport references range from 2 to 430,661.

## Airport Identifier Distribution

- There are 36,025 unique airport identifiers.
- "KORD" is the most common, appearing 11 times.

## Runway Length Distribution

- Runway lengths range from 0 to 30,000 feet.

## Runway Width Distribution

- Runway widths range from 0 to 9,000 feet.

## Surface Type Distribution

- There are 592 unique surface types.
- "ASP" is the most common, appearing 10,702 times.

## Lighted Runway Distribution

- Most runways (42,390 out of 42,390) are lighted.

## Closed Runway Distribution

- Only a small percentage of runways (16.70%) are marked as closed.

## Left Runway Identifier Distribution

- There are 266 unique left runway identifiers.
- "H1" is the most common, appearing 5,958 times.

## Left Runway Latitude Distribution

- Left runway latitudes range from -75.60 to 82.51 degrees.

## Left Runway Longitude Distribution

- Left runway longitudes range from -178.30 to 179.34 degrees.

## Left Runway Elevation Distribution

- Left runway elevations range from -1,246 to 13,202 feet.

## Left Runway Heading Distribution

- Left runway headings range from 0 to 360 degrees.

## Report on Counts in the Dataset

### Count of Airlines:

The dataset contains information about different airlines. Here is a breakdown of the counts for each airline:

- WN (Southwest Airlines) has the highest count with 94,097 flights.
- DL (Delta Air Lines) follows closely with 60,940 flights.
- OO (SkyWest Airlines) has 50,254 flights.
- AA (American Airlines) has 45,656 flights.
- MQ (Envoy Air) has 36,605 flights.
- US (US Airways) has 34,500 flights.
- XE (ExpressJet Airlines) has 31,126 flights.
- EV (ExpressJet Airlines) has 27,983 flights.
- UA (United Airlines) has 27,619 flights.
- CO (Continental Airlines) has 21,118 flights.
- 9E (Endeavor Air) has 20,686 flights.
- B6 (JetBlue Airways) has 18,112 flights.
- YV (Mesa Airlines) has 13,725 flights.
- OH (Comair) has 12,630 flights.
- AS (Alaska Airlines) has 11,471 flights.
- F9 (Frontier Airlines) has 6,456 flights.
- HA (Hawaiian Airlines) has 5,578 flights.

This information provides an overview of the distribution of flights among different airlines in the dataset.

### Report on Count of AirportFrom and AirportTo:

The dataset includes information about departure and arrival airports. Here are the counts for both departure and arrival airports:

**For the 'AirportFrom' column:**

- ATL (Hartsfield-Jackson Atlanta International Airport) is the most common departure airport with 28,827 flights.
- ORD (Chicago O'Hare International Airport) follows with 24,822 flights.
- DFW (Dallas/Fort Worth International Airport) has 21,900 flights.
- DEN (Denver International Airport) has 19,720 flights.
- LAX (Los Angeles International Airport) has 16,490 flights.
- There are a total of 291 unique departure airports in the dataset.

**For the 'AirportTo' column:**

- ATL (Hartsfield-Jackson Atlanta International Airport) is also the most common arrival airport with 28,825 flights.
- ORD (Chicago O'Hare International Airport) follows with 24,871 flights.
- DFW (Dallas/Fort Worth International Airport) has 21,899 flights.
- DEN (Denver International Airport) has 19,725 flights.
- LAX (Los Angeles International Airport) has 16,491 flights.
- There are also 291 unique arrival airports in the dataset.

These counts provide insights into the popularity of various airports for both departures and arrivals.

## Report on Count of DayOfWeek:

The dataset records flights on different days of the week. Here is the count of flights for each day of the week:

- Day 4 (Thursday) has the highest count with 87,988 flights.
- Day 3 (Wednesday) follows closely with 86,478 flights.
- Day 5 (Friday) has 81,797 flights.
- Day 1 (Sunday) has 70,008 flights.
- Day 2 (Monday) has 68,721 flights.
- Day 7 (Saturday) has 67,210 flights.
- Day 6 (Sunday) has 56,354 flights.

These counts reveal the distribution of flights across different days of the week.

## Correlation Matrix:

The correlation matrix provides insights into the relationships between numerical columns in the dataset. Here are the correlations between various columns:

- The 'Delay' column has a positive correlation with the 'Time' column (0.15), indicating that longer flights tend to have more delays.
- The 'Length' column has a negative correlation with the 'Flight' column (-0.35), suggesting that shorter flights are associated with higher flight numbers.
- There is a weak positive correlation (0.14) between the 'Delay' and 'id' columns.

It's important to note that correlation does not imply causation, but these correlations can provide valuable information for further analysis and modeling.

## Descriptive Statistics for Numerical Columns:

- **latitude_deg:**
  - Count: 73,805
  - Mean: 25.786389
  - Standard Deviation: 26.232686
  - Minimum: -90.000000
  - 25th Percentile: 12.536100
  - Median (50th Percentile): 35.160179
  - 75th Percentile: 42.720901
  - Maximum: 82.750000
- **longitude_deg:**
  - Count: 73,805
  - Mean: -28.880235
  - Standard Deviation: 86.121515
  - Minimum: -179.876999
  - 25th Percentile: -94.170097
  - Median (50th Percentile): -69.893898
  - 75th Percentile: 23.934668
  - Maximum: 179.975700
- **elevation_ft:**
  - Count: 59,683
  - Mean: 1299.934370
  - Standard Deviation: 1672.759483
  - Minimum: -1266.000000
  - 25th Percentile: 205.000000
  - Median (50th Percentile): 730.000000
  - 75th Percentile: 1608.000000
  - Maximum: 17,372.000000

## Statistics Grouped by Continent:

- Descriptive statistics for latitude_deg, longitude_deg, and elevation_ft for each continent are provided. For example, in Africa (AF), the mean latitude is -5.791058, the mean longitude is 23.599012, and the mean elevation is 2524.691277.

## Statistics Grouped by ISO Country:

- Descriptive statistics for latitude_deg, longitude_deg, and elevation_ft are provided for each ISO country code. For example, for AD (Andorra), the mean latitude is 42.534156, the mean longitude is 1.501690, and the mean elevation is 3450.000000.

## Statistics Grouped by Type:

- Descriptive statistics for latitude_deg, longitude_deg, and elevation_ft are provided for each airport type. For example, for "heliport," the mean latitude is 28.711213, the mean longitude is -8.883333, and the mean elevation is 1247.456442.

These statistics give you insights into the distribution and variation of geographical and elevation data in your dataset, both overall and within specific groups based on continent, ISO country code, and airport type.

## Latitude and Longitude Distribution:

- The `latitude_deg` column ranges from -90.000000 (representing the South Pole) to 82.750000 (near the North Pole).
- The `longitude_deg` column spans from -179.876999 (near the International Date Line) to 179.975700 (opposite side of the globe).
- This wide distribution of latitude and longitude values suggests that the dataset contains airport locations from around the world.

## Elevation Variation:

- The `elevation_ft` column shows a wide variation in elevation levels, ranging from -1266.000000 (possibly below sea level) to 17,372.000000 feet.
- This indicates that airports are situated at varying altitudes, from below sea level to high-altitude locations.

## Continent-wise Analysis:

- The data is grouped by continent, revealing significant differences in latitude, longitude, and elevation among continents.
- For example, airports in Africa (AF) tend to have lower latitudes and longitudes, while those in Asia (AS) have a wider range of longitudes.
- Elevation levels also vary widely by continent, with airports in Africa having relatively higher elevations compared to other continents.

## Country-wise Analysis:

- Descriptive statistics for latitude, longitude, and elevation are provided for individual ISO countries.
- Some countries, like Andorra (AD), have relatively consistent values for latitude, longitude, and elevation, while others, like the United States (US), exhibit significant variations.

## Airport Type Variation:

- Different types of airports, such as "heliport" and "small_airport," have distinct characteristics.
- Heliports tend to have higher mean elevations, possibly due to their locations in mountainous or elevated areas.

- Closed airports, on average, have lower elevations compared to other types, indicating that they may be located in lower-lying regions.

## Data Completeness:

- It's important to note that the count of `elevation_ft` is lower than the total count of records. This suggests that some airport entries do not have elevation data.
- Data completeness and quality checks may be required to ensure accurate analysis, especially when considering elevation-related insights.

## Geographical Diversity:

- The dataset includes airport locations from various parts of the world, reflecting the global reach of aviation.
- Researchers or analysts working with this dataset can explore geographical patterns and relationships based on these coordinates and elevation data.

### Excel Analysis