Week 12 - Final Project

Introduction

Airline cancellations or delays are one of the major causes for passenger inconvenience. With the publicly available dataset (huge datasets with around 16 million flights flown annually), using datascience I am hoping to gain meaningful insights into the best performing airlines and understanding the causes for delays and cancellations across different airline carriers. For the final project I would like to analyze airline data to identify different factors and their effects on a carrier's performance. As a performance measure, we would be exploring on-time arrivals, number of cancellations by carrier and also explore different reasons for a carrier delay. Data Science can help identify the major causes of delay and cancellations per carrier. Based on the outcome, carriers can take necessary actions to focus on the problem areas.

DATA SOURCE: Department of Transportation(DOT) -
https://catalog.data.gov/dataset/airline-on-time-performance-and-causes-of-flight-delays/

With this analysis, I am hoping to address a few questions such as -

1. Are small carriers reliable in terms of lesser cancellations and delays?
2. Are the delays seasonal? If yes, which regions are most affected?
3. Does the time of day have any significance on delays?
4. Which carrier has the best on-time performance.
5. Which carrier has the least on-time performance.
6. Identifying the most common cancellation reason for all carriers.
7. Which carrier has the most number of cancellations.
8. Which carrier has the most number of delays.
9. What is the percentage of delays by reason.

Problem statement addressed:

This study would benefit airlines by comparing airline performances and predicting possibilities of delay based on aircraft/origin/destination and apply corrective measures to reduce cancellations and delays and to improve on-time performance.

Research Questions: Following are the topics I would like to focus on as part of this project.

1. Are small carriers reliable in terms of lesser cancellations and delays?
2. Are the delays seasonal? If yes, which regions are most affected?
3. Does the time of day have any significance on delays?
4. Which carrier has the best on-time performance.

5. Which carrier has the least on-time performance.
6. Identifying the most common cancellation reason for all carriers.
7. Which carrier has the most number of cancellations.
8. Which carrier has the most number of delays.
9. What is the percentage of delays by reason.

Approach: I will be performing the following steps:

1. Data analysis - Gathering and understanding different datasets.
2. Data Cleaning and Transforming
3. Merge transformed/cleansed datasets
4. Data visualization/plotting

Addressing the problem: Based on the outcomes from data analysis and visualization, I would like to identify the following:

- Which carriers are more likely to cause delays or cancellations.
- Which carriers are more reliable in terms of on-time performance.

Datasets:

Below data submitted by major carriers to department of transportation (DOT).

- Flights.csv
- UniqueCarriers.csv
- Airports.csv

Data was collected by DOT's Bureau of Transportation Statistics for the year 2022. The purpose of this data is to analyze airline on-time performance reported by carriers. The datasets has around 40 fields in total of which I will be considering between 15 to 25 columns for analysis.

Analysis

In [1]:

```
from os.path import basename, exists


def download(url):
    filename = basename(url)
    if not exists(filename):
        from urllib.request import urlretrieve

        local, _ = urlretrieve(url, filename)
        print("Downloaded " + local)
```

In [2]:

```
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/density.py")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/first.py")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/nsfg.py")
```

```
# Import libraries
import glob
import pandas as pd
import os
import sys
import numpy as np
import thinkstats2
from pyspark.sql.functions import col
from pandas.core.common import SettingWithCopyWarning
import warnings

import thinkplot
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
#from plotly.offline import init_notebook_mode, iplot
#from plotly.graph_objs import *
#import plotly.io as pio

import scipy
import sklearn.linear_model as slm
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression

#if not os.path.exists("images"):
#    os.mkdir("images")
```

In [3]:
```
#init_notebook_mode(connected=True)
#pio.kaleido.scope.default_format="png"
```

In [4]:
```
# Load Reference Data
sys.path.append(os.getcwd() + '/Data')
reference_data_path = 'C:/Users/aarti/ThinkStats2-Code/Assignments/Week12/Data'
#reference_data_path
```

In [5]:
```
#Airport - Reference Data
airport_data_file = reference_data_path+'/'+'L_AIRPORT.csv'
airport_data_df = pd.read_csv(airport_data_file)
airport_data_df.head(3)
```

Out[5]:

	Code	Description
0	01A	Afognak Lake, AK: Afognak Lake Airport
1	03A	Granite Mountain, AK: Bear Creek Mining Strip
2	04A	Lik, AK: Lik Mining Camp

In [6]:
```
#Cancellation - Reference Data
cancellation_data_file = reference_data_path+'/'+'L_CANCELLATION.csv'
cancellation_data_df = pd.read_csv(cancellation_data_file)
cancellation_data_df
```

Out[6]:

	Code	Description
0	A	Carrier
1	B	Weather
2	C	National Air System
3	D	Security

In [7]:
```
#Unique Carriers - Reference Data
unique_carrier_data_file = reference_data_path+'/'+'L_UNIQUE_CARRIERS.csv'
unique_carrier_data_df = pd.read_csv(unique_carrier_data_file)
unique_carrier_data_df.head(3)
```

Out[7]:

	Code	Description
0	02Q	Titan Airways
1	04Q	Tradewind Aviation
2	05Q	Comlux Aviation, AG

In [8]:
```
#Flight Data - Concatinate flight data for the year 2022 - Jan through Nov.
sys.path.append(os.getcwd() + '/Data/FlightData')
os.getcwd()
flight_data_path = 'C:/Users/aarti/ThinkStats2-Code/Assignments/Week12/Data/FlightD
flight_data_csv_files = glob.glob(flight_data_path + "/*.csv")
#flight_data_csv_files
```

In [9]:
```
#This creates a list of dataframes
data_df = (pd.read_csv(file) for file in flight_data_csv_files)

#Concatenate all files into a DataFrames
flight_data_df = pd.concat(data_df, ignore_index=False)
print(len(flight_data_df))
flight_data_df.head(3)
```

6435187

Out[9]:

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	FL_DATE	MKT_UNIQUE_CARRIER
0	2022	2	4	1	5	4/1/2022 12:00:00 AM	AA
1	2022	2	4	1	5	4/1/2022 12:00:00 AM	AA
2	2022	2	4	1	5	4/1/2022 12:00:00 AM	AA

3 rows × 39 columns

For the research, I would like to consider the following columns.

OP_UNIQUE_CARRIER - Operating Carrier Airline Code

CANCELLATION_CODE - Specifies The Reason For Cancellation

DIVERTED - A flight that is required to land at a destination other than the original scheduled destination for reasons beyond the control of the pilot/company.

DISTANCE - Distance between airports (miles)

ARR_DELAY - Difference in minutes between scheduled and actual arrival time. Early arrivals show negative numbers.

DEP_DELAY - Difference in minutes between scheduled and actual departure time.

CARRIER_DELAY - Carrier Delay, in Minutes

WEATHER_DELAY - Weather Delay, in Minutes

NAS_DELAY - National Air System Delay, in Minutes

SECURITY_DELAY - Security Delay, in Minutes

LATE_AIRCRAFT_DELAY - Late Aircraft Delay, in Minutes

Data Transformation

In [10]:

```
#Carrier codes in flight dataset are represented as 2 character airline carrier cod
#Looking up the carrier code against the unique carrier dataset and updating the
#code by carrier name in the flight dataframe for both operating and marketing carr
flight_data_df=pd.merge(flight_data_df, unique_carrier_data_df, how='left', left_on
flight_data_df.rename(columns={'Description':'MKT_UNIQUE_CARRIER_NAME'}, inplace=Tr
del flight_data_df['Code']

#Add Carrier Name for operating carrier
flight_data_df=pd.merge(flight_data_df, unique_carrier_data_df, how='left', left_on
flight_data_df.rename(columns={'Description':'OP_UNIQUE_CARRIER_NAME'}, inplace=Tru
del flight_data_df['Code']
```

In [11]:

```
#Cancellation reason in the flight dataset is represented as A, B, C and D.
#Looking up the cancellation code against the cancellation dataset and adding
#cancellation description to the flight dataframe.
flight_data_df=pd.merge(flight_data_df, cancellation_data_df, how='left', left_on='
flight_data_df.rename(columns={'Description':'CANCELLATION_REASON'}, inplace=True)
del flight_data_df['Code']

flight_data_df.groupby(['CANCELLATION_REASON'])['CANCELLATION_REASON'].count().sort
```

Out[11]:
```
CANCELLATION_REASON
Carrier                   54128
National Air System       15387
Security                   1057
Weather                   88279
Name: CANCELLATION_REASON, dtype: int64
```

In [12]:
```
# Drop null rows if any
flight_data_df.dropna()
#Update null values to 0
flight_data_df.DISTANCE = flight_data_df.DISTANCE.fillna(0)
flight_data_df.DEP_DELAY = flight_data_df.DEP_DELAY.fillna(0)
flight_data_df.ARR_DELAY = flight_data_df.ARR_DELAY.fillna(0)
flight_data_df.CARRIER_DELAY = flight_data_df.CARRIER_DELAY.fillna(0)
flight_data_df.WEATHER_DELAY = flight_data_df.WEATHER_DELAY.fillna(0)
flight_data_df.NAS_DELAY = flight_data_df.NAS_DELAY.fillna(0)
flight_data_df.SECURITY_DELAY = flight_data_df.SECURITY_DELAY.fillna(0)
flight_data_df.LATE_AIRCRAFT_DELAY = flight_data_df.LATE_AIRCRAFT_DELAY.fillna(0)
```

In [13]:
```
#Update Day of week from Number to Day
flight_data_df.DAY_OF_WEEK = np.where(flight_data_df.DAY_OF_WEEK==1, 'Monday',
              np.where(flight_data_df.DAY_OF_WEEK==2, 'Tuesday',
              np.where(flight_data_df.DAY_OF_WEEK==3, 'Wednesday',
              np.where(flight_data_df.DAY_OF_WEEK==4, 'Thursday',
              np.where(flight_data_df.DAY_OF_WEEK==5, 'Friday',
              np.where(flight_data_df.DAY_OF_WEEK==6, 'Saturday',
              np.where(flight_data_df.DAY_OF_WEEK==7, 'Sunday','')))))))
```

In [14]:
```
 # Add a new column for performance status
flight_data_df['STATUS'] = ''

flight_data_df.STATUS = np.where(flight_data_df.CANCELLED==1, 'Cancelled',
                                 np.where(flight_data_df.DIVERTED==1, 'Diverted',
                                          np.where(flight_data_df.ARR_DELAY<=15, 'O
                                                   np.where(flight_data_df.ARR_DELA
flight_data_df.groupby(['STATUS'])['STATUS'].count().sort_index()
```

Out[14]:
```
STATUS
Cancelled     158851
Delayed      1236619
Diverted       15297
On-Time      5024420
Name: STATUS, dtype: int64
```

In [15]:
```
# Creating a flag for delayed flights
flight_data_df.loc[(flight_data_df['ARR_DELAY']>15), 'DELAYED'] = True
flight_data_df.loc[(flight_data_df['ARR_DELAY']<=15), 'DELAYED'] = False

flight_data_df.groupby(['DELAYED'])['DELAYED'].count().sort_index()
```

Out[15]:
```
DELAYED
False    5198568
True     1236619
Name: DELAYED, dtype: int64
```

In [16]:
```
flight_data_df['DELAY_REASON'] = np.where(((flight_data_df.DELAYED==True) & (flight
                                          np.where(((flight_data_df.DELAYED==True)
                                                   np.where(((flight_data_df.DELAYE
                                                            np.where(((flight_data_
                                                                     np.where(((fli

flight_data_df.groupby(['DELAY_REASON'])['DELAY_REASON'].count().sort_index()
```

Out[16]:
```
DELAY_REASON
                5198569
Carrier          751660
LateAircraft     265082
NAS              175812
Security           1651
Weather           42413
Name: DELAY_REASON, dtype: int64
```

In [17]:
```
#Since the number of rows are very high (over 6 million), we'll narrow the research

#Filtering ORIGIN airports
flight_data_df = flight_data_df.loc[(flight_data_df.ORIGIN == "ORD") | (flight_data
                                    (flight_data_df.ORIGIN == "DFW") | (flight_data
                                    (flight_data_df.ORIGIN == "EWR") | (flight_data
                                    (flight_data_df.ORIGIN == "IAH") | (flight_data
                                    (flight_data_df.ORIGIN == "DTW") | (flight_data
                                    (flight_data_df.ORIGIN == "LAS") | (flight_data
                                    (flight_data_df.ORIGIN == "ORD") | (flight_data
                                    (flight_data_df.ORIGIN == "CLT") | (flight_data
                                    (flight_data_df.ORIGIN == "MCO") | (flight_data
                                    (flight_data_df.ORIGIN == "BOS") | (flight_data
```

In [18]:
```
#Filtering DESTINATION airports
print(len(flight_data_df))
flight_data_df = flight_data_df.loc[(flight_data_df.DEST == "ORD") | (flight_data_d
                                    (flight_data_df.DEST == "DFW") | (flight_data_d
                                    (flight_data_df.DEST == "EWR") | (flight_data_d
                                    (flight_data_df.DEST == "IAH") | (flight_data_d
                                    (flight_data_df.DEST == "DTW") | (flight_data_d
                                    (flight_data_df.DEST == "LAS") | (flight_data_d
                                    (flight_data_df.DEST == "ORD") | (flight_data_d
                                    (flight_data_df.DEST == "CLT") | (flight_data_d
                                    (flight_data_df.DEST == "MCO") | (flight_data_d
                                    (flight_data_df.DEST == "BOS") | (flight_data_d
print(len(flight_data_df))
```

```
3016994
1073457
```

In [19]:
```
# Selecting relevant columns from flights data
flight_data_df =  flight_data_df[["YEAR","QUARTER","MONTH","DAY_OF_MONTH","DAY_OF_W
                                  "FL_DATE","MKT_UNIQUE_CARRIER","OP_UNIQUE_CARRIER
                                  "MKT_UNIQUE_CARRIER_NAME","ORIGIN","ORIGIN_CITY_N
                                  "ORIGIN_STATE_NM","DEST","DEST_CITY_NAME","DEST_S
                                  "DEST_STATE_NM","DEP_DELAY","TAXI_OUT","TAXI_IN",
                                  "CANCELLED","CANCELLATION_CODE","CANCELLATION_REA
```

```
                                  "CARRIER_DELAY","WEATHER_DELAY","NAS_DELAY","SECU
                                  "DELAYED" ,"DELAY_REASON","STATUS"]]
```

In [20]:
```
#Validating transformed data
print('Total number of rows',len(flight_data_df))
print('\n',flight_data_df.groupby(['DELAY_REASON'])['DELAY_REASON'].count().sort_in
print('\n',flight_data_df.groupby(['DELAYED'])['DELAYED'].count().sort_index())
print('\n',flight_data_df.groupby(['STATUS'])['STATUS'].count().sort_index())
print('\n',flight_data_df.groupby(['CANCELLATION_REASON'])['CANCELLATION_REASON'].c
```

```
Total number of rows 1073457

 DELAY_REASON
                857935
Carrier         132924
LateAircraft     37109
NAS              38438
Security           243
Weather           6808
Name: DELAY_REASON, dtype: int64

 DELAYED
False    857935
True     215522
Name: DELAYED, dtype: int64

 STATUS
Cancelled     27655
Delayed      215522
Diverted       2408
On-Time      827872
Name: STATUS, dtype: int64

 CANCELLATION_REASON
Carrier                10632
National Air System     2724
Security                 264
Weather                14035
Name: CANCELLATION_REASON, dtype: int64
```

For the purposes of this analysis, we are considering flights with arrival time less than 15 minutes as on-time.

In [21]:
```
delayed_arrival = flight_data_df[flight_data_df.DELAYED==True]
on_time_arrival = flight_data_df[flight_data_df.DELAYED==False]

carrier_delay_df = flight_data_df[flight_data_df.DELAY_REASON =='Carrier']
late_aircraft_delay_df = flight_data_df[flight_data_df.DELAY_REASON =='LateAircraft
nas_delay_df = flight_data_df[flight_data_df.DELAY_REASON =='NAS']
security_delay_df = flight_data_df[flight_data_df.DELAY_REASON =='Security']
weather_delay_df = flight_data_df[flight_data_df.DELAY_REASON =='Weather']

print('Delayed : ',len(delayed_arrival))
print('On-Time : ',len(on_time_arrival))
print('CarrierDelays : ',len(carrier_delay_df))
print('LateAircraftDelays : ',len(late_aircraft_delay_df))
```

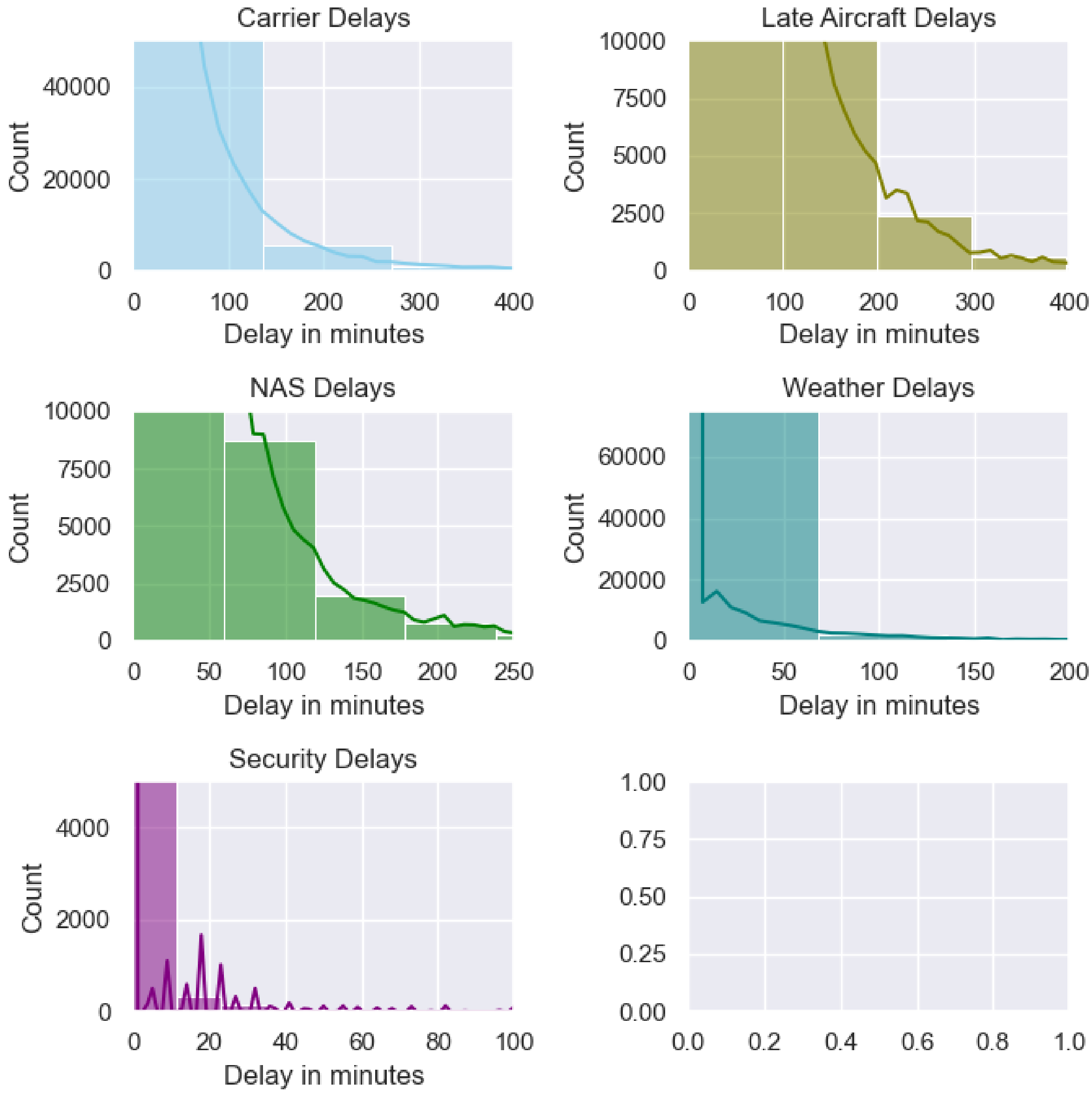
```
print('NasDelays : ',len(nas_delay_df))
print('SecurityDelays : ',len(security_delay_df))
print('WeatherDelays : ',len(weather_delay_df))

cancelled_df = flight_data_df[flight_data_df.CANCELLED == 1]
diverted_df = flight_data_df[flight_data_df.DIVERTED == 1]

print('Cancelled : ',len(cancelled_df))
print('Diverted : ',len(diverted_df))
```

```
Delayed :  215522
On-Time :  857935
CarrierDelays :  132924
LateAircraftDelays :  37109
NasDelays :  38438
SecurityDelays :  243
WeatherDelays :  6808
Cancelled :  27655
Diverted :  2408
```

Histogram

In [23]:

```
# set a grey background (use sns.set_theme() if seaborn version 0.11.0 or above)
sns.set(style="darkgrid")
fig, ((ax0,ax1),(ax2,ax3),(ax4,ax5)) = plt.subplots(3, 2, figsize=(7, 7))

sns.histplot(data=flight_data_df, x="CARRIER_DELAY", kde=True, color="skyblue", ax=
ax0.set_xlim([0, 400])
ax0.set_ylim([0, 50000])
ax0.set(xlabel='Delay in minutes',ylabel='Count',title='Carrier Delays')

sns.histplot(data=flight_data_df, x="LATE_AIRCRAFT_DELAY", kde=True, color="olive",
ax1.set_xlim([0, 400])
ax1.set_ylim([0, 10000])
ax1.set(xlabel='Delay in minutes',ylabel='Count',title='Late Aircraft Delays')


sns.histplot(data=flight_data_df, x="NAS_DELAY", kde=True, color="green", ax=ax2)
ax2.set_xlim([0, 250])
ax2.set_ylim([0, 10000])
ax2.set(xlabel='Delay in minutes',ylabel='Count',title='NAS Delays')

sns.histplot(data=flight_data_df, x="WEATHER_DELAY", kde=True, color="teal", ax=ax3
ax3.set_xlim([0, 200])
ax3.set_ylim([0, 75000])
ax3.set(xlabel='Delay in minutes',ylabel='Count',title='Weather Delays')

sns.histplot(data=flight_data_df, x="SECURITY_DELAY", kde=True, color="purple", ax=
ax4.set_xlim([0, 100])
ax4.set_ylim([0, 5000])
ax4.set(xlabel='Delay in minutes',ylabel='Count',title='Security Delays')

fig.tight_layout()
```



From the plots we can see that most delays are caused by Carriers.

Descriptive Statistics

In [24]:
```
flight_data_df.describe()
```

Out[24]:

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DEP_DELAY	TAXI_OUT	
count	1073457.0	1.073457e+06	1.073457e+06	1.073457e+06	1.073457e+06	1.045855e+06	1.0
mean	2022.0	2.398995e+00	6.106110e+00	1.572306e+01	1.326509e+01	1.864172e+01	9.4
std	0.0	1.063062e+00	3.134588e+00	8.769813e+00	5.237481e+01	9.926912e+00	7.0
min	2022.0	1.000000e+00	1.000000e+00	1.000000e+00	-7.800000e+01	1.000000e+00	1.0
25%	2022.0	1.000000e+00	3.000000e+00	8.000000e+00	-5.000000e+00	1.300000e+01	6.0
50%	2022.0	2.000000e+00	6.000000e+00	1.600000e+01	-1.000000e+00	1.600000e+01	8.0
75%	2022.0	3.000000e+00	9.000000e+00	2.300000e+01	1.000000e+01	2.100000e+01	1.1
max	2022.0	4.000000e+00	1.100000e+01	3.100000e+01	2.991000e+03	1.970000e+02	2.5

In [25]:

```
print('MEDIAN','\n')
print('DISTANCE : ', flight_data_df.DISTANCE.median())
print('DEPARTURE DELAY : ',flight_data_df.DEP_DELAY.median())
print('ARRIVAL DELAY : ',flight_data_df.ARR_DELAY.median())
print('CARRIER DELAY : ',flight_data_df.CARRIER_DELAY.median())
print('WEATHER DELAY : ',flight_data_df.WEATHER_DELAY.median())
print('NAS DELAY : ',flight_data_df.NAS_DELAY.median())
print('SECURITY DELAY : ',flight_data_df.SECURITY_DELAY.median())
print('LATE AIRCRAFT DELAY : ',flight_data_df.LATE_AIRCRAFT_DELAY.median())

print('\n','\n','MODE','\n')
print('DISTANCE : ', flight_data_df.DISTANCE.mode())
print('DEPARTURE DELAY : ',flight_data_df.DEP_DELAY.mode())
print('ARRIVAL DELAY : ',flight_data_df.ARR_DELAY.mode())
print('CARRIER DELAY : ',flight_data_df.CARRIER_DELAY.mode())
print('WEATHER DELAY : ',flight_data_df.WEATHER_DELAY.mode())
print('NAS DELAY : ',flight_data_df.NAS_DELAY.mode())
print('SECURITY DELAY : ',flight_data_df.SECURITY_DELAY.mode())
print('LATE AIRCRAFT DELAY : ',flight_data_df.LATE_AIRCRAFT_DELAY.mode())
```

```
MEDIAN

DISTANCE :  907.0
DEPARTURE DELAY :  -1.0
ARRIVAL DELAY :  -6.0
CARRIER DELAY :  0.0
WEATHER DELAY :  0.0
NAS DELAY :  0.0
SECURITY DELAY :  0.0
LATE AIRCRAFT DELAY :  0.0


 MODE

DISTANCE :  0    733.0
Name: DISTANCE, dtype: float64
DEPARTURE DELAY :  0    0.0
Name: DEP_DELAY, dtype: float64
ARRIVAL DELAY :  0    0.0
Name: ARR_DELAY, dtype: float64
CARRIER DELAY :  0    0.0
Name: CARRIER_DELAY, dtype: float64
WEATHER DELAY :  0    0.0
Name: WEATHER_DELAY, dtype: float64
NAS DELAY :  0    0.0
Name: NAS_DELAY, dtype: float64
SECURITY DELAY :  0    0.0
Name: SECURITY_DELAY, dtype: float64
LATE AIRCRAFT DELAY :  0    0.0
Name: LATE_AIRCRAFT_DELAY, dtype: float64
```

The average arrival delay is only around 6 minutes. We can see that the median value is -6 minutes, suggesting the majority of flights actually arrive earlier than their expected time of arrival.

In [26]:
```
#Tail for flight data
flight_data_df.tail()
```

Out[26]:

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	FL_DATE	MKT_UNIQUE_C/
6434666	2022	3	9	30	Friday	9/30/2022 12:00:00 AM	
6434667	2022	3	9	30	Friday	9/30/2022 12:00:00 AM	
6434668	2022	3	9	30	Friday	9/30/2022 12:00:00 AM	
6434669	2022	3	9	30	Friday	9/30/2022 12:00:00 AM	
6434670	2022	3	9	30	Friday	9/30/2022 12:00:00 AM	

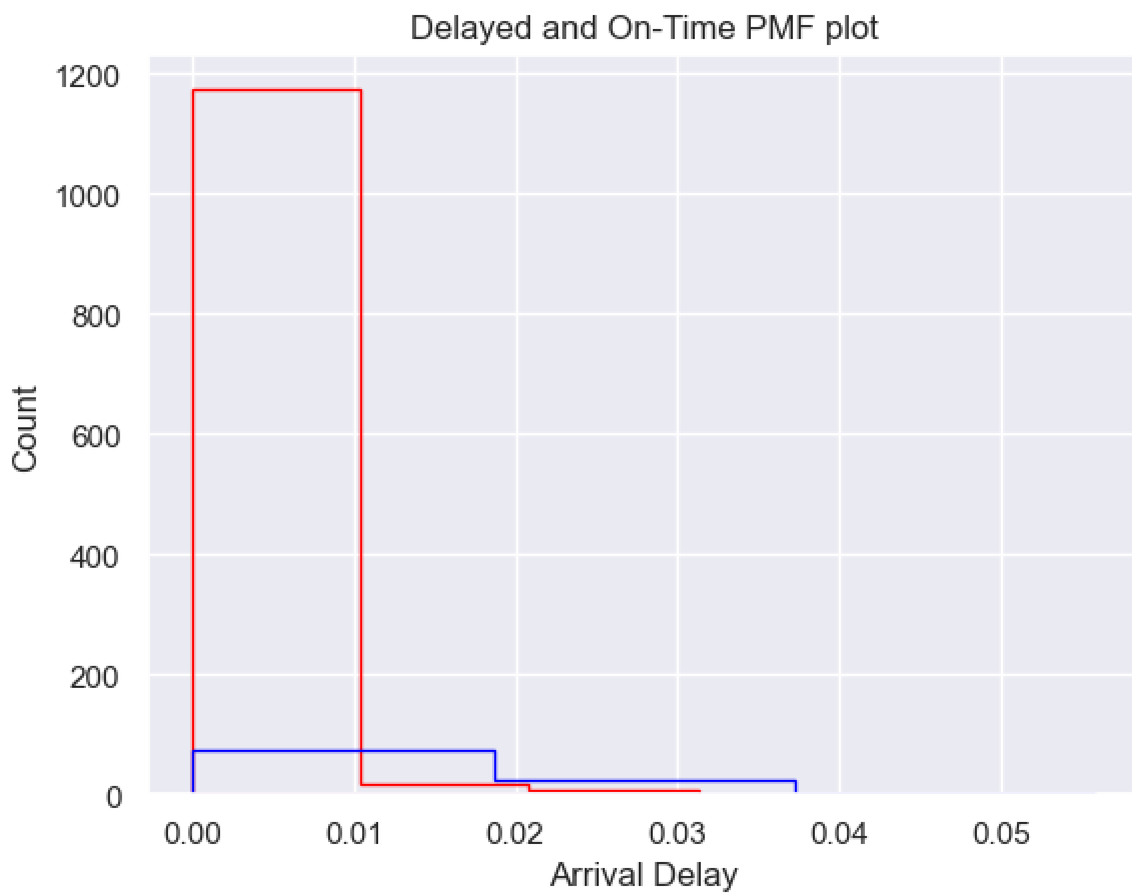
5 rows × 35 columns

PMF

In [27]:

```
delay_pmf = delayed_arrival.ARR_DELAY.value_counts().sort_index() / len(delayed_arr
on_time_pmf = on_time_arrival.ARR_DELAY.value_counts().sort_index() / len(on_time_a


plt.hist(delay_pmf,  histtype='stepfilled', facecolor='none', edgecolor='red',bins=
plt.hist(on_time_pmf,  histtype='stepfilled', facecolor='none', edgecolor='blue',bi
plt.title('Delayed and On-Time PMF plot')
plt.xlabel('Arrival Delay')
plt.ylabel('Count')
```

Out[27]: Text(0, 0.5, 'Count')

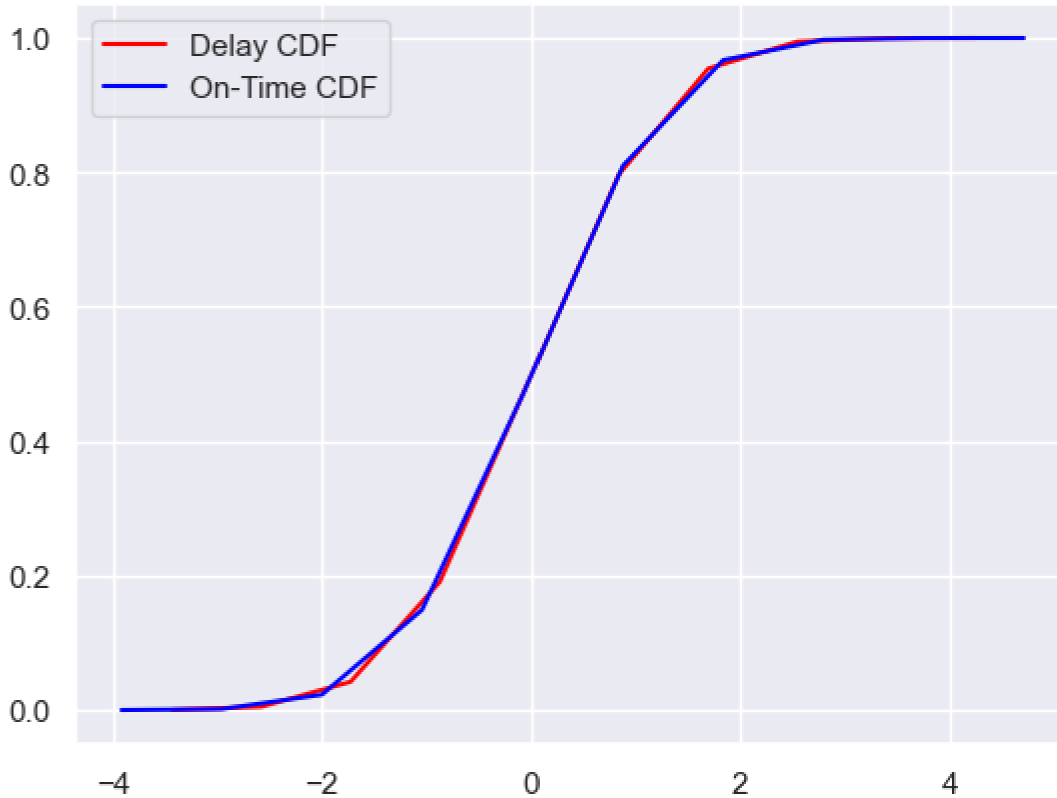


CDF

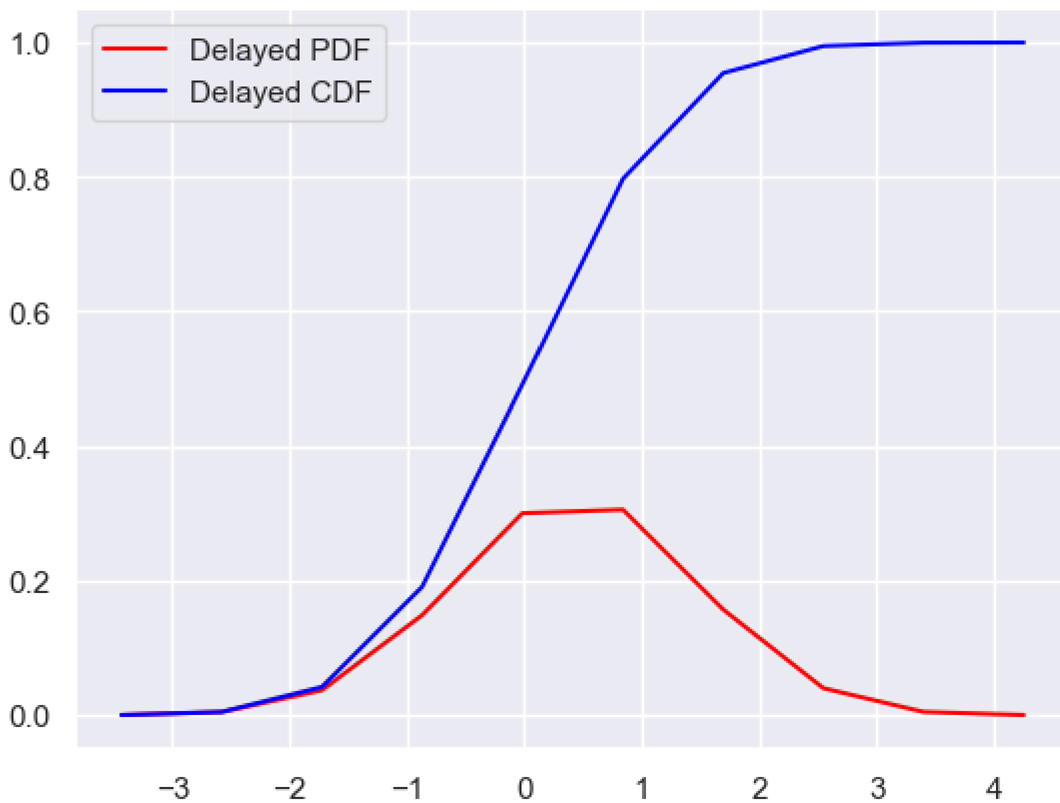
In [28]:

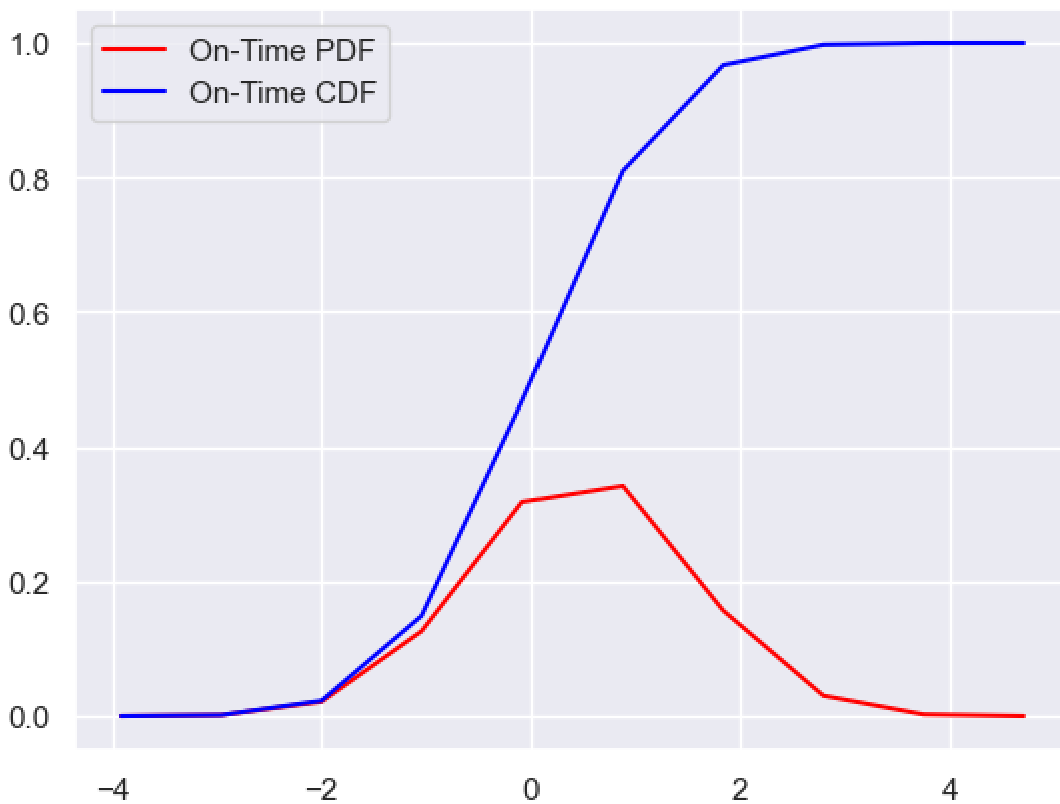
```
# No of Data points
d_N = len(delayed_arrival)
# initializing random values
d_data = np.random.randn(d_N)
# getting data of the histogram
d_count, d_bins_count = np.histogram(d_data, bins=10)
# finding the PDF of the histogram using count values
delayed_pdf = d_count / sum(d_count)
# using numpy np.cumsum to calculate the CDF
# We can also find using the PDF values by looping and adding
delayed_cdf = np.cumsum(delayed_pdf)

# No of Data points
o_N = len(on_time_arrival)
# initializing random values
o_data = np.random.randn(o_N)
# getting data of the histogram
o_count, o_bins_count = np.histogram(o_data, bins=10)
# finding the PDF of the histogram using count values
on_time_pdf = o_count / sum(o_count)
# using numpy np.cumsum to calculate the CDF
# We can also find using the PDF values by looping and adding
on_time_cdf = np.cumsum(on_time_pdf)
```

```
plt.plot(d_bins_count[1:], delayed_cdf, label="Delay CDF", color="red")
plt.plot(o_bins_count[1:], on_time_cdf, label="On-Time CDF", color="blue")
plt.legend()
plt.show()
```



In [29]:
```
plt.plot(d_bins_count[1:], delayed_pdf, label="Delayed PDF", color="red")
plt.plot(d_bins_count[1:], delayed_cdf, label="Delayed CDF", color="blue")
plt.legend()
plt.show()

plt.plot(o_bins_count[1:], on_time_pdf, label="On-Time PDF", color="red")
plt.plot(o_bins_count[1:], on_time_cdf, label="On-Time CDF", color="blue")
plt.legend()
plt.show()
```



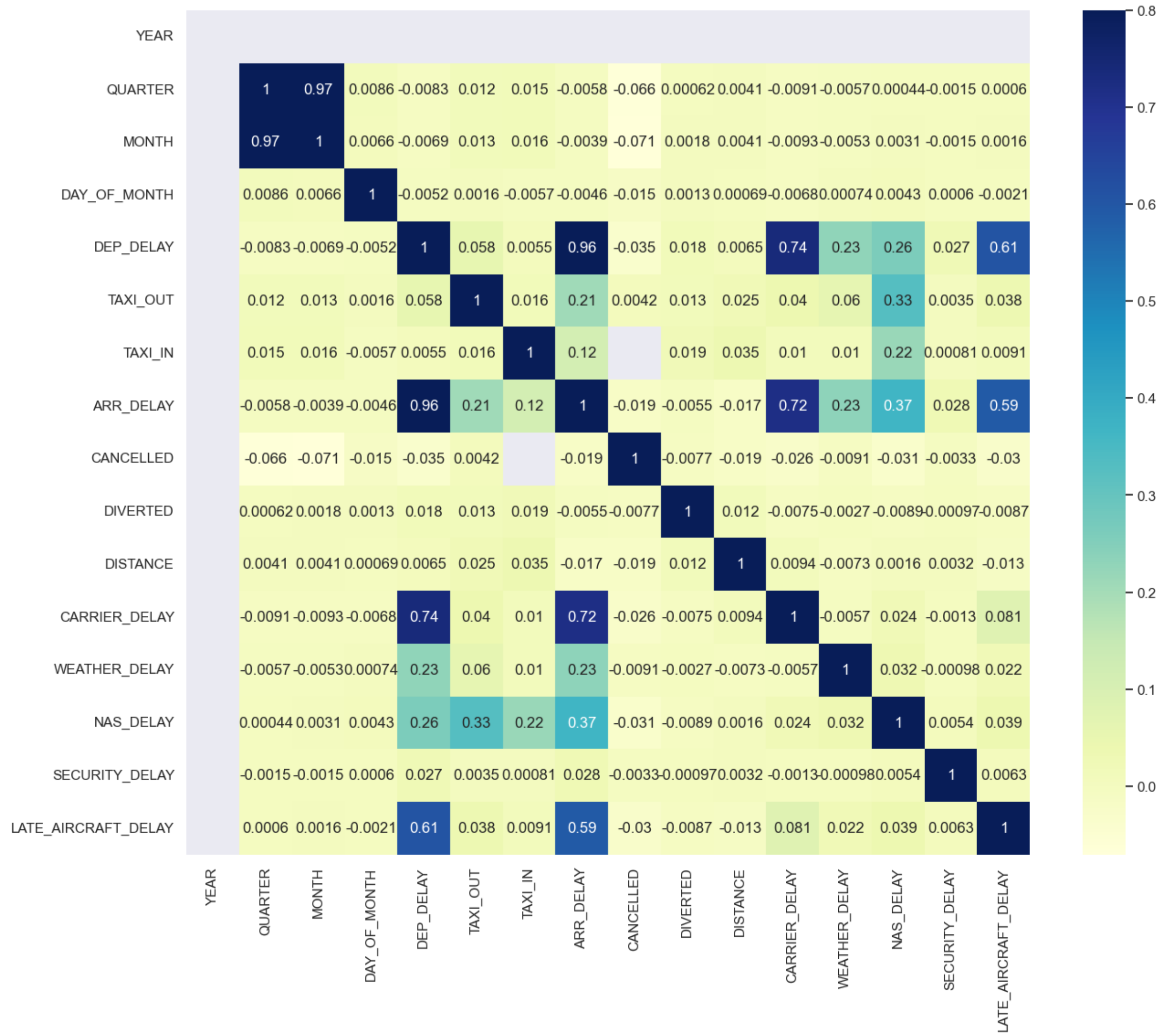


Analytical Distribution

In [28]:
```
corr_df = flight_data_df[["YEAR","QUARTER","MONTH","DAY_OF_MONTH",
                          "FL_DATE","DEP_DELAY","TAXI_OUT","TAXI_IN","ARR_D
```

```
                                   "CANCELLED","CANCELLATION_CODE","DIVERTED","DISTA
                                   "CARRIER_DELAY","WEATHER_DELAY","NAS_DELAY","SECU
```

In [31]:
```python
corrmat = corr_df.corr()
f, ax = plt.subplots(figsize=(15, 12))
sns.heatmap(corrmat, vmax=.8, square=True,annot=True,cmap='YlGnBu');
plt.show()
```



In [32]:
```python
corrmat
```

Out[32]:

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DEP_DELAY	TAXI_OUT	TA
YEAR	NaN	NaN	NaN	NaN	NaN	NaN	
QUARTER	NaN	1.000000	0.968575	0.008605	-0.008267	0.011999	0.0
MONTH	NaN	0.968575	1.000000	0.006644	-0.006852	0.012857	0.0
DAY_OF_MONTH	NaN	0.008605	0.006644	1.000000	-0.005231	0.001552	-0.0
DEP_DELAY	NaN	-0.008267	-0.006852	-0.005231	1.000000	0.058371	0.0
TAXI_OUT	NaN	0.011999	0.012857	0.001552	0.058371	1.000000	0.0
TAXI_IN	NaN	0.015220	0.016422	-0.005678	0.005461	0.015587	1.0
ARR_DELAY	NaN	-0.005770	-0.003884	-0.004600	0.956878	0.205379	0.1
CANCELLED	NaN	-0.065984	-0.070686	-0.014633	-0.034835	0.004208	
DIVERTED	NaN	0.000615	0.001756	0.001320	0.017735	0.013404	0.0
DISTANCE	NaN	0.004118	0.004064	0.000691	0.006531	0.024519	0.0
CARRIER_DELAY	NaN	-0.009069	-0.009271	-0.006799	0.737808	0.039696	0.0
WEATHER_DELAY	NaN	-0.005721	-0.005319	0.000738	0.228622	0.059951	0.0
NAS_DELAY	NaN	0.000436	0.003050	0.004301	0.262007	0.330432	0.2
SECURITY_DELAY	NaN	-0.001478	-0.001512	0.000602	0.027430	0.003491	0.0
LATE_AIRCRAFT_DELAY	NaN	0.000601	0.001572	-0.002087	0.607649	0.037839	0.0

Departure delay has a close correlation with carrier delay and late aircraft delay. These 2 delay reasons could be contributing to departure delays.

Arival Delay has a close correlation with departure delay, carrier delay and late aircraft delay.

Scatter plots comparing two variables

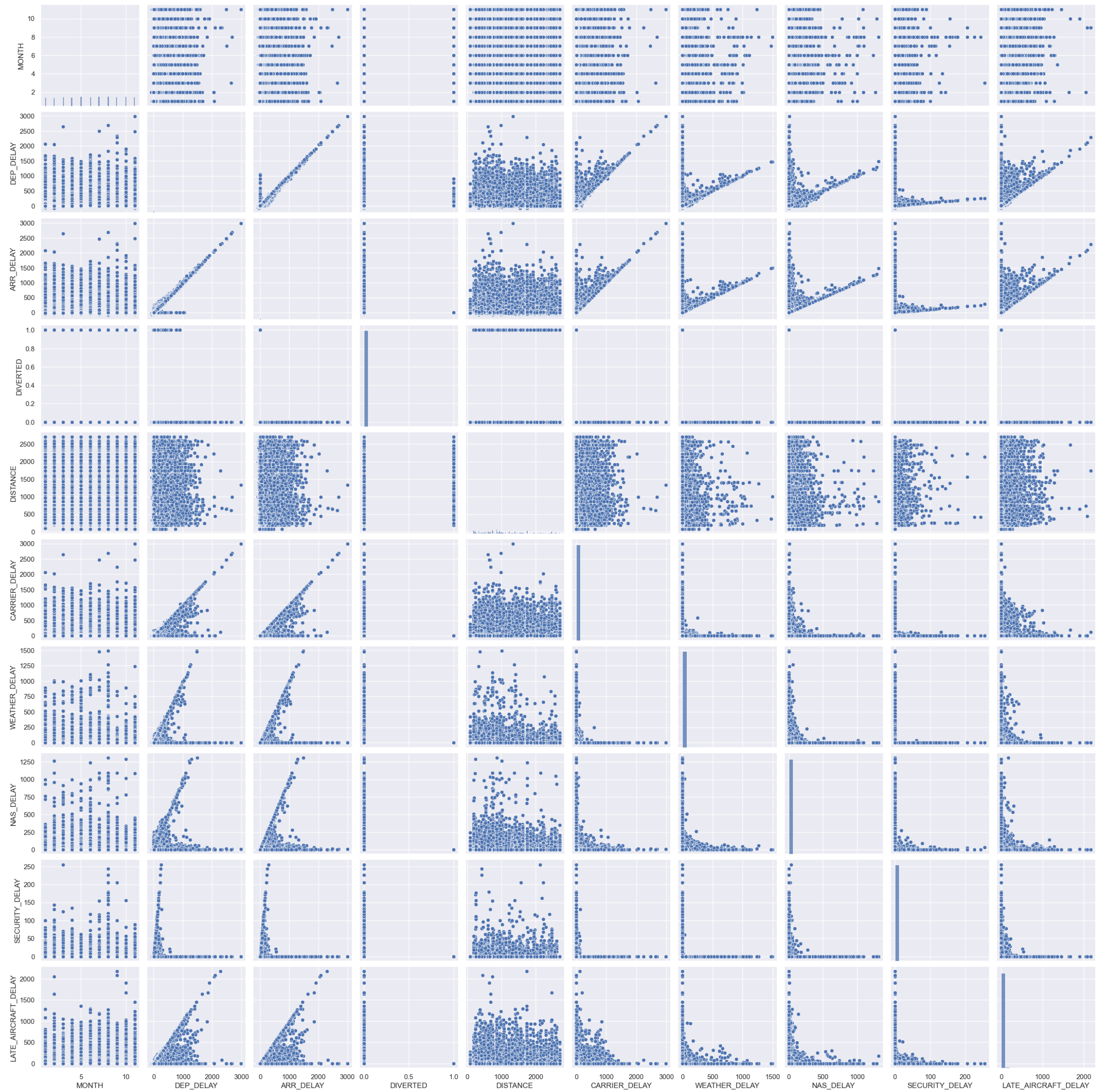
Covariance

Pearson’s correlation, and

Non-Linear Relationships

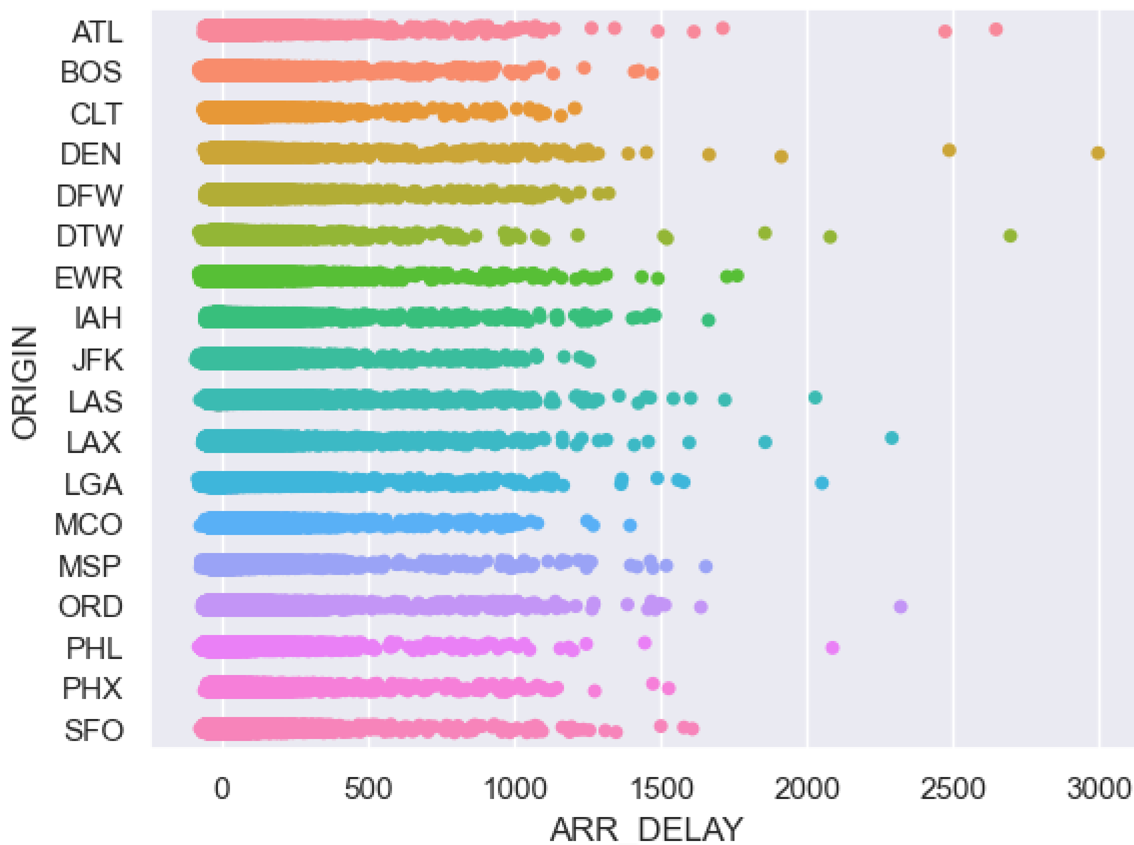
In [32]:

```
corr_df = flight_data_df[["MONTH","DEP_DELAY","ARR_DELAY","CANCELLATION_CODE","DIVE

sns.pairplot(corr_df)
plt.show()
```



In [33]:

```
#axis = plt.subplots(figsize=(10,14))
sns.despine(bottom=True, left=True)
# Observations with Scatter Plot
sns.stripplot(x="ARR_DELAY", y="ORIGIN",data = flight_data_df, dodge=True, jitter=T
plt.show()
```



In [34]:

```
print('\n','\n','SKEWNESS','\n')
print('DISTANCE : ', scipy.stats.skew(flight_data_df.DISTANCE))
print('DEPARTURE DELAY : ',scipy.stats.skew(flight_data_df.DEP_DELAY))
print('ARRIVAL DELAY : ',scipy.stats.skew(flight_data_df.ARR_DELAY))
print('CARRIER DELAY : ',scipy.stats.skew(flight_data_df.CARRIER_DELAY))
print('WEATHER DELAY : ',scipy.stats.skew(flight_data_df.WEATHER_DELAY))
print('NAS DELAY : ',scipy.stats.skew(flight_data_df.NAS_DELAY))
print('SECURITY DELAY : ',scipy.stats.skew(flight_data_df.SECURITY_DELAY))
print('LATE AIRCRAFT DELAY : ',scipy.stats.skew(flight_data_df.LATE_AIRCRAFT_DELAY)

print('\n','\n','KURTOSIS','\n')
print('DISTANCE : ', scipy.stats.kurtosis(flight_data_df.DISTANCE))
print('DEPARTURE DELAY : ',scipy.stats.kurtosis(flight_data_df.DEP_DELAY))
print('ARRIVAL DELAY : ',scipy.stats.kurtosis(flight_data_df.ARR_DELAY))
print('CARRIER DELAY : ',scipy.stats.kurtosis(flight_data_df.CARRIER_DELAY))
print('WEATHER DELAY : ',scipy.stats.kurtosis(flight_data_df.WEATHER_DELAY))
print('NAS DELAY : ',scipy.stats.kurtosis(flight_data_df.NAS_DELAY))
print('SECURITY DELAY : ',scipy.stats.kurtosis(flight_data_df.SECURITY_DELAY))
print('LATE AIRCRAFT DELAY : ',scipy.stats.kurtosis(flight_data_df.LATE_AIRCRAFT_DE
```

```
 SKEWNESS

DISTANCE :  0.7921339541032948
DEPARTURE DELAY :  10.674804748347697
ARRIVAL DELAY :  9.536490658683023
CARRIER DELAY :  20.049523097873497
WEATHER DELAY :  45.46099194885411
NAS DELAY :  17.872705271312853
SECURITY DELAY :  92.90007389949665
LATE AIRCRAFT DELAY :  14.879549252215236


 KURTOSIS

DISTANCE :  -0.21837907395493783
DEPARTURE DELAY :  209.72530841384085
ARRIVAL DELAY :  179.55865054436103
CARRIER DELAY :  646.4560643730676
WEATHER DELAY :  3239.851325493505
NAS DELAY :  717.2507844906166
SECURITY DELAY :  12459.326647890168
LATE AIRCRAFT DELAY :  454.49686950998534
```

In [22]:

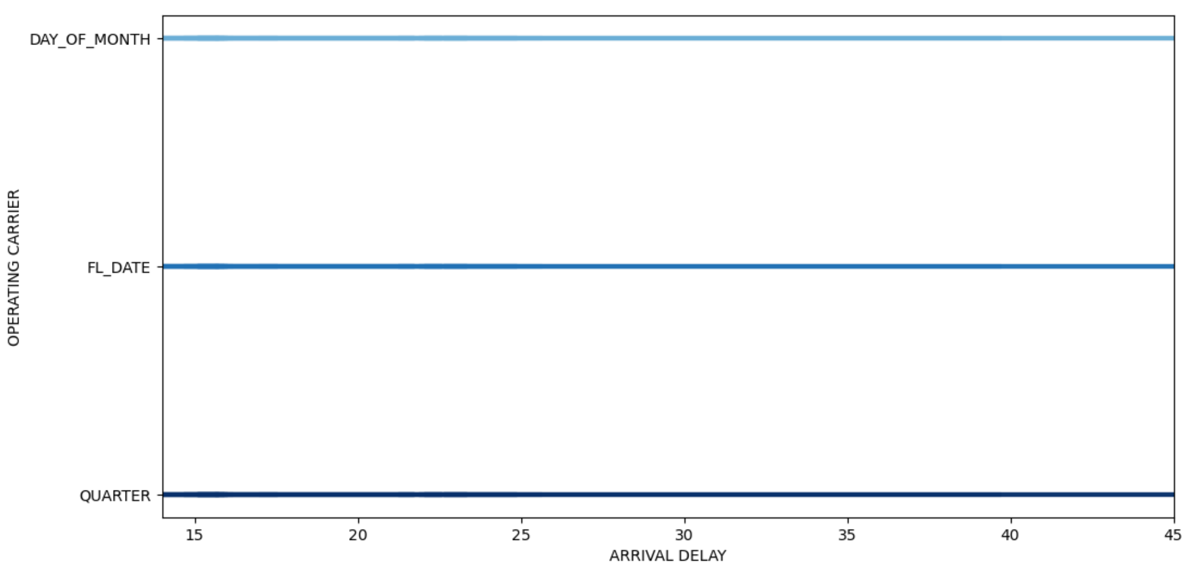
```
def Corr(xs, ys):
    xs = np.asarray(xs)
    ys = np.asarray(ys)

    meanx, varx = thinkstats2.MeanVar(xs)
    meany, vary = thinkstats2.MeanVar(ys)

    corr = Cov(xs, ys, meanx, meany) / np.sqrt(varx * vary)
    return corr

def Cov(xs, ys, meanx=None, meany=None):
    xs = np.asarray(xs)
    ys = np.asarray(ys)

    if meanx is None:
        meanx = np.mean(xs)
    if meany is None:
        meany = np.mean(ys)

    cov = np.dot(xs-meanx, ys-meany) / len(xs)
    return cov

def SpearmanCorr(xs, ys):
    xranks = pd.Series(xs).rank()
    yranks = pd.Series(ys).rank()
    return Corr(xranks, yranks)

def BinPercentiles(df):
    bins=np.arange(10,48,3)
    indices=np.digitize(df['ARR_DELAY'],bins)
    #print('INDICES :',indices)
    groups=df.groupby(indices)
```

```
    #print('GROUPS :',groups)

    gp=[group.mean() for i, group in groups]
    cdfs=[thinkstats2.Cdf(group) for i, group in groups]
    #print('CDFs:',cdfs)

    thinkplot.PrePlot(3)
    for percent in [75,50,25]:
        cd=[cdf.Percentile(percent) for cdf in cdfs]
        #print('CD:',cd)
        label='%dth' % percent
        thinkplot.Plot(gp,cd)
        thinkplot.Config(xlabel="ARRIVAL DELAY",ylabel="OPERATING CARRIER",xlim=[14
```

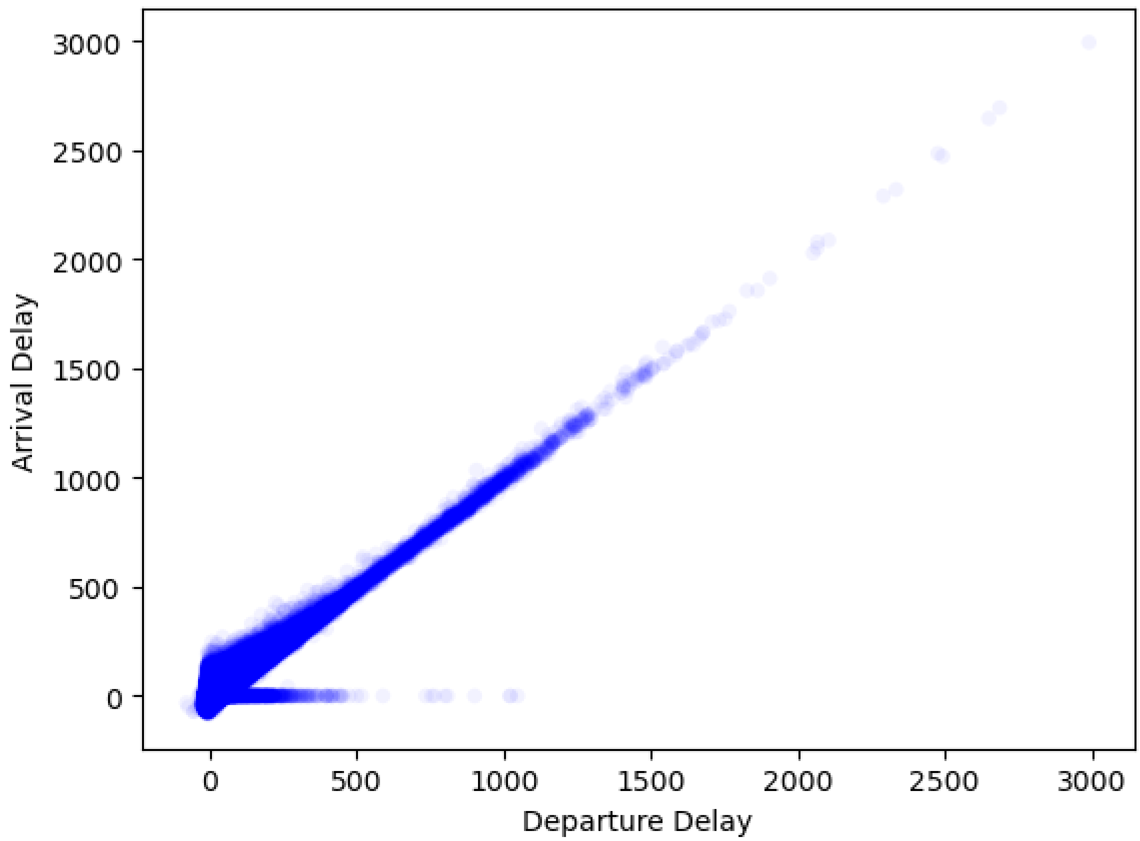
In [26]:
```
dep_delay = flight_data_df.DEP_DELAY
arr_delay = flight_data_df.ARR_DELAY

print(len(dep_delay))
print(len(arr_delay))

print('Correlation',Corr(arr_delay,dep_delay))
print("Spearman's Correlation",SpearmanCorr(arr_delay,dep_delay))
```

```
1073457
1073457
Correlation 0.9568775802514493
Spearman's Correlation 0.6727587861415726
```

In [29]:
```
warnings.simplefilter(action='ignore', category=FutureWarning)
fig, ax = plt.subplots(figsize=(12,6))

corr_df_sample = corr_df.drop(['CANCELLATION_CODE','SECURITY_DELAY'],axis=1)
sample = thinkstats2.SampleRows(corr_df_sample, 10000)

BinPercentiles(sample)
```



In [30]:
```
thinkplot.Scatter(flight_data_df.DEP_DELAY,flight_data_df.ARR_DELAY,alpha=0.05)
thinkplot.Config(xlabel="Departure Delay",ylabel="Arrival Delay")
```



Tables for plots

In [31]:
```
flight_totals = flight_data_df.value_counts(subset=['OP_UNIQUE_CARRIER','OP_UNIQUE_
flight_totals_df = pd.DataFrame(flight_totals)
flight_totals_df.columns = ['OP_UNIQUE_CARRIER','OP_UNIQUE_CARRIER_NAME','TOTAL']
flight_totals_df['PERCENTAGE'] = round(flight_totals_df.TOTAL/flight_totals_df.TOTA

flight_totals_df = flight_totals_df.sort_values('PERCENTAGE',ascending=False)
flight_totals_df.head(5)
```

Out[31]:

	OP_UNIQUE_CARRIER	OP_UNIQUE_CARRIER_NAME	TOTAL	PERCENTAGE
0	AA	American Airlines Inc.	256452	23.89
1	DL	Delta Air Lines Inc.	228512	21.29
2	UA	United Air Lines Inc.	208725	19.44
3	B6	JetBlue Airways	76435	7.12
4	WN	Southwest Airlines Co.	75171	7.00

In [32]:
```
flight_stats = flight_data_df.value_counts(subset=['OP_UNIQUE_CARRIER','OP_UNIQUE_C
flight_stats_df = pd.DataFrame(flight_stats)
flight_stats_df.columns = ['OP_UNIQUE_CARRIER','OP_UNIQUE_CARRIER_NAME','DELAY_REAS
flight_stats_df = flight_stats_df.sort_values('OP_UNIQUE_CARRIER')

flight_stats_df['PERCENTAGE'] = ''
```

```
for index, row in flight_stats_df.iterrows():
    tot = flight_totals.loc[flight_totals.OP_UNIQUE_CARRIER==row.OP_UNIQUE_CARRIER]
    val = (row.COUNT/tot * 100)
    flight_stats_df.at[index,'PERCENTAGE'] = round(val[0].astype(float),2)

flight_stats_df.head(10)
```

Out[32]:

	OP_UNIQUE_CARRIER	OP_UNIQUE_CARRIER_NAME	DELAY_REASON	COUNT	PERCENTAGE
52	9E	Endeavor Air Inc.	NAS	559	4.45
46	9E	Endeavor Air Inc.	Carrier	829	6.59
100	9E	Endeavor Air Inc.	Security	1	0.01
53	9E	Endeavor Air Inc.	LateAircraft	556	4.42
73	9E	Endeavor Air Inc.	Weather	77	0.61
15	9E	Endeavor Air Inc.		10553	83.92
36	AA	American Airlines Inc.	Weather	1886	0.74
74	AA	American Airlines Inc.	Security	70	0.03
21	AA	American Airlines Inc.	NAS	7621	2.97
14	AA	American Airlines Inc.	LateAircraft	10606	4.14

In [33]:

```
flight_status = flight_data_df.value_counts(subset=['OP_UNIQUE_CARRIER','OP_UNIQUE_
flight_status_df = pd.DataFrame(flight_status)
flight_status_df.columns = ['OP_UNIQUE_CARRIER','OP_UNIQUE_CARRIER_NAME','STATUS',
flight_status_df = flight_status_df.sort_values('OP_UNIQUE_CARRIER')

flight_status_df['PERCENTAGE'] = ''

for index, row in flight_status_df.iterrows():
    tot = flight_totals.loc[flight_totals.OP_UNIQUE_CARRIER==row.OP_UNIQUE_CARRIER]
    val = (row.COUNT/tot * 100)
    flight_status_df.at[index,'PERCENTAGE'] = round(val[0].astype(float),2)

flight_status_df.head(10)
```

Out[33]:

	OP_UNIQUE_CARRIER	OP_UNIQUE_CARRIER_NAME	STATUS	COUNT	PERCENTAGE
29	9E	Endeavor Air Inc.	Delayed	2022	16.08
59	9E	Endeavor Air Inc.	Diverted	29	0.23
16	9E	Endeavor Air Inc.	On-Time	9893	78.67
39	9E	Endeavor Air Inc.	Cancelled	631	5.02
0	AA	American Airlines Inc.	On-Time	197045	76.84
38	AA	American Airlines Inc.	Diverted	648	0.25
5	AA	American Airlines Inc.	Delayed	50919	19.86
18	AA	American Airlines Inc.	Cancelled	7840	3.06
61	AS	Alaska Airlines Inc.	Diverted	19	0.15
17	AS	Alaska Airlines Inc.	On-Time	9400	74.45

In [34]:

```
airline_on_time_performance = flight_status_df[flight_status_df.STATUS == 'On-Time'
airline_on_time_performance.head(10)
```

Out[34]:

	OP_UNIQUE_CARRIER	OP_UNIQUE_CARRIER_NAME	STATUS	COUNT	PERCENTAGE
50	PT	Piedmont Airlines	On-Time	127	82.47
10	OO	SkyWest Airlines Inc.	On-Time	31690	81.92
25	OH	PSA Airlines Inc.	On-Time	3855	81.28
1	DL	Delta Air Lines Inc.	On-Time	185561	81.2
23	MQ	Envoy Air	On-Time	4107	80.89
2	UA	United Air Lines Inc.	On-Time	167229	80.12
68	G4	Allegiant Air	On-Time	4	80.0
19	YV	Mesa Airlines Inc.	On-Time	7664	79.12
16	9E	Endeavor Air Inc.	On-Time	9893	78.67
9	YX	Republic Airline	On-Time	32953	76.85

In [35]:

```
status_percentage = flight_data_df.value_counts(subset=['STATUS']).reset_index()
status_percentage_df = pd.DataFrame(status_percentage)
status_percentage_df.columns = ['STATUS', 'COUNT']

status_percentage_df['PERCENTAGE'] = ''
tot = status_percentage_df.COUNT.sum()

for index, row in status_percentage_df.iterrows():

    val = (row.COUNT/tot * 100)
    status_percentage_df.at[index,'PERCENTAGE'] = round(val.astype(float),2)

status_percentage_df
```

Out[35]:

	STATUS	COUNT	PERCENTAGE
0	On-Time	827872	77.12
1	Delayed	215522	20.08
2	Cancelled	27655	2.58
3	Diverted	2408	0.22

In [36]:

```
flight_cancel = flight_data_df.value_counts(subset=['OP_UNIQUE_CARRIER','OP_UNIQUE_
flight_cancel_df = pd.DataFrame(flight_cancel)
flight_cancel_df.columns = ['OP_UNIQUE_CARRIER','OP_UNIQUE_CARRIER_NAME','CANCELLAT
flight_cancel_df = flight_cancel_df.sort_values('OP_UNIQUE_CARRIER')

flight_cancel_df['PERCENTAGE'] = ''
flight_cancel_df


for index, row in flight_cancel_df.iterrows():
    tot = flight_totals.loc[flight_totals.OP_UNIQUE_CARRIER==row.OP_UNIQUE_CARRIER]
    val = (row.COUNT/tot * 100)
    flight_cancel_df.at[index,'PERCENTAGE'] = round(val[0].astype(float),2)

flight_cancel_df.head(10)
```

Out[36]:

	OP_UNIQUE_CARRIER	OP_UNIQUE_CARRIER_NAME	CANCELLATION_REASON	COUNT	PERCENT
30	9E	Endeavor Air Inc.	Carrier	118	
24	9E	Endeavor Air Inc.	Weather	210	
21	9E	Endeavor Air Inc.	National Air System	303	
0	AA	American Airlines Inc.	Weather	4813	
16	AA	American Airlines Inc.	National Air System	442	
1	AA	American Airlines Inc.	Carrier	2585	
19	AS	Alaska Airlines Inc.	Carrier	357	
41	AS	Alaska Airlines Inc.	Weather	17	
50	AS	Alaska Airlines Inc.	National Air System	3	
8	B6	JetBlue Airways	Carrier	1143	

In [37]:

```
delayed_performance = flight_status_df[flight_status_df.STATUS == 'Delayed'].sort_v
delayed_performance.head(10)
```

Out[37]:

	OP_UNIQUE_CARRIER	OP_UNIQUE_CARRIER_NAME	STATUS	COUNT	PERCENTAGE
15	F9	Frontier Airlines Inc.	Delayed	11831	30.35
12	B6	JetBlue Airways	Delayed	21284	27.85
13	WN	Southwest Airlines Co.	Delayed	19469	25.9
43	G7	GoJet Airlines LLC d/b/a United Express	Delayed	463	25.4
14	NK	Spirit Air Lines	Delayed	14103	23.52
46	QX	Horizon Air	Delayed	209	22.94
26	AS	Alaska Airlines Inc.	Delayed	2830	22.41
64	ZW	Air Wisconsin Airlines Corp	Delayed	9	20.0
71	G4	Allegiant Air	Delayed	1	20.0
5	AA	American Airlines Inc.	Delayed	50919	19.86

In [38]:

```
flight_origin_totals = flight_data_df.value_counts(subset=['ORIGIN']).reset_index()
flight_origin_totals_df = pd.DataFrame(flight_origin_totals)
flight_origin_totals_df.columns = ['ORIGIN','TOTAL']
flight_origin_totals_df['PERCENTAGE'] = round(flight_origin_totals_df.TOTAL/flight_

cancelled_status = flight_data_df.value_counts(subset=['ORIGIN','CANCELLATION_REASO
cancelled_status_df = pd.DataFrame(cancelled_status)
cancelled_status_df.columns = ['ORIGIN','CANCELLATION_REASON','STATUS', 'COUNT']
cancelled_status_df = cancelled_status_df.sort_values('ORIGIN')
cancelled_status_df['PERCENTAGE'] = ''

print(cancelled_status_df.head(10))
for index, row in cancelled_status_df.iterrows():
    tot = flight_origin_totals.loc[flight_origin_totals.ORIGIN==row.ORIGIN].TOTAL.v
    val = (row.COUNT/tot * 100)
    cancelled_status_df.at[index,'PERCENTAGE'] = round(val[0].astype(float),2)

cancelled_status_df.head(10)
cancelled_status_df = cancelled_status_df.sort_values('PERCENTAGE',ascending=False)

cancelled_status_df=pd.merge(cancelled_status_df, airport_data_df, how='left', left
cancelled_status_df.rename(columns={'Description':'ORIGIN_AIRPORT_NAME'}, inplace=T
del cancelled_status_df['Code']

new = cancelled_status_df.ORIGIN_AIRPORT_NAME.str.split(":", n = 1, expand = True)
cancelled_status_df["ORIGIN_AIRPORT_NAME"] =  new[1]
cancelled_status_df[cancelled_status_df.STATUS=='Cancelled']
```

```
   ORIGIN  CANCELLATION_REASON     STATUS  COUNT PERCENTAGE
43    ATL  National Air System  Cancelled    159
64    ATL             Security  Cancelled      8
15    ATL              Carrier  Cancelled    643
14    ATL              Weather  Cancelled    655
34    BOS  National Air System  Cancelled    317
60    BOS             Security  Cancelled     16
5     BOS              Weather  Cancelled   1141
12    BOS              Carrier  Cancelled    700
42    CLT  National Air System  Cancelled    167
70    CLT             Security  Cancelled      2
```

Out[38]:

	ORIGIN	CANCELLATION_REASON	STATUS	COUNT	PERCENTAGE	ORIGIN_AIRPORT_NAME
0	LGA	Weather	Cancelled	1299	2.3	LaGuardia
1	EWR	Weather	Cancelled	1218	2.26	Newark Liberty International
2	DFW	Weather	Cancelled	1442	2.15	Dallas/Fort Worth International
3	MCO	Weather	Cancelled	1210	1.97	Orlando International
4	CLT	Weather	Cancelled	974	1.86	Charlotte Douglas International
...	...	...	...	...	...	...
66	PHL	Security	Cancelled	5	0.01	Philadelphia International
67	DFW	Security	Cancelled	9	0.01	Dallas/Fort Worth International
68	LGA	Security	Cancelled	6	0.01	LaGuardia
69	MSP	Security	Cancelled	3	0.01	Minneapolis-St Paul International
70	CLT	Security	Cancelled	2	0.0	Charlotte Douglas International

71 rows × 6 columns

In [39]:

```
delayed_status = flight_data_df.value_counts(subset=['ORIGIN','STATUS']).reset_inde
delayed_status_df = pd.DataFrame(delayed_status)
delayed_status_df.columns = ['ORIGIN','STATUS', 'COUNT']
delayed_status_df = delayed_status_df.sort_values('ORIGIN')
delayed_status_df['PERCENTAGE'] = ''

for index, row in delayed_status_df.iterrows():
    tot = flight_origin_totals.loc[flight_origin_totals.ORIGIN==row.ORIGIN].TOTAL.v
    val = (row.COUNT/tot * 100)
    delayed_status_df.at[index,'PERCENTAGE'] = round(val[0].astype(float),2)

delayed_status_df.head(10)
delayed_status_df = delayed_status_df.sort_values('PERCENTAGE',ascending=False)

delayed_status_df=pd.merge(delayed_status_df, airport_data_df, how='left', left_on=
delayed_status_df.rename(columns={'Description':'ORIGIN_AIRPORT_NAME'}, inplace=Tru
```

```
del delayed_status_df['Code']

new = delayed_status_df.ORIGIN_AIRPORT_NAME.str.split(":", n = 1, expand = True)
delayed_status_df["ORIGIN_AIRPORT_NAME"] =  new[1]

delayed_status_df
```

Out[39]:

	ORIGIN	STATUS	COUNT	PERCENTAGE	ORIGIN_AIRPORT_NAME
0	SFO	On-Time	46291	83.19	San Francisco International
1	DTW	On-Time	37182	81.14	Detroit Metro Wayne County
2	MSP	On-Time	33335	80.62	Minneapolis-St Paul International
3	LAX	On-Time	66889	80.13	Los Angeles International
4	IAH	On-Time	41878	80.09	George Bush Intercontinental/Houston
...	...	...	...	...	...
67	BOS	Diverted	136	0.21	Logan International
68	ORD	Diverted	174	0.2	Chicago O'Hare International
69	SFO	Diverted	113	0.2	San Francisco International
70	DTW	Diverted	93	0.2	Detroit Metro Wayne County
71	EWR	Diverted	87	0.16	Newark Liberty International

72 rows × 5 columns

In [40]:

```
delayed_status_reason = flight_data_df.value_counts(subset=['ORIGIN','DELAY_REASON'
delayed_status_reason_df = pd.DataFrame(delayed_status_reason)
delayed_status_reason_df.columns = ['ORIGIN','DELAY_REASON','STATUS', 'COUNT']
delayed_status_reason_df = delayed_status_reason_df.sort_values('ORIGIN')
delayed_status_reason_df['PERCENTAGE'] = ''

for index, row in delayed_status_reason_df.iterrows():
    tot = flight_origin_totals.loc[flight_origin_totals.ORIGIN==row.ORIGIN].TOTAL.v
    val = (row.COUNT/tot * 100)
    delayed_status_reason_df.at[index,'PERCENTAGE'] = round(val[0].astype(float),2)

delayed_status_reason_df.head(10)

delayed_status_reason_df = delayed_status_reason_df.sort_values('PERCENTAGE',ascend

delayed_status_reason_df=pd.merge(delayed_status_reason_df, airport_data_df, how='l
delayed_status_reason_df.rename(columns={'Description':'ORIGIN_AIRPORT_NAME'}, inpl
del delayed_status_reason_df['Code']
delayed_status_reason_df

new = delayed_status_reason_df.ORIGIN_AIRPORT_NAME.str.split(":", n = 1, expand = T
delayed_status_reason_df["ORIGIN_AIRPORT_NAME"] =  new[1]

delayed_status_reason_df
```

Out[40]:

	ORIGIN	DELAY_REASON	STATUS	COUNT	PERCENTAGE	ORIGIN_AIRPORT_NAME
0	SFO		On-Time	46291	83.19	San Francisco International
1	DTW		On-Time	37182	81.14	Detroit Metro Wayne County
2	MSP		On-Time	33335	80.62	Minneapolis-St Paul International
3	LAX		On-Time	66889	80.13	Los Angeles International
4	IAH		On-Time	41878	80.09	George Bush Intercontinental/Houston
...	...	...	...	...	...	...
139	PHL	Security	Delayed	5	0.01	Philadelphia International
140	BOS	Security	Delayed	5	0.01	Logan International
141	SFO	Security	Delayed	7	0.01	San Francisco International
142	DTW	Security	Delayed	5	0.01	Detroit Metro Wayne County
143	ORD	Security	Delayed	7	0.01	Chicago O'Hare International

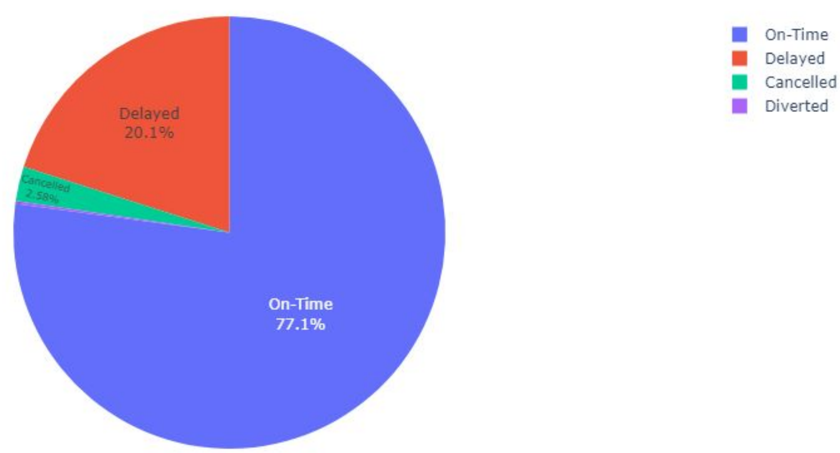
144 rows × 6 columns

Airline Performance

In [61]:

```
fig = px.pie(status_percentage_df, values='PERCENTAGE', names='STATUS', title='Over
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.show()
#fig.write_image("Overall Airline Performance for 2022/fig1.pdf",engine='kaleido')
```

Overall Airline Performance for 2022

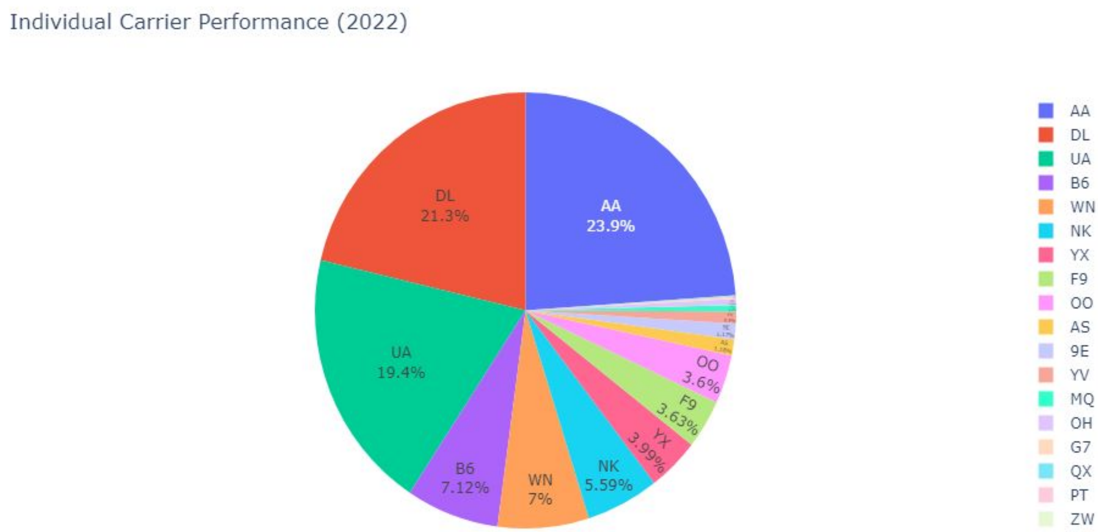


In [138…

```
#plt.pie(status_percentage_df.PERCENTAGE, labels = status_percentage_df.STATUS,   a
#plt.title("Overall Airline Performance for 2022")
#plt.show()
```

In [64]:

```
fig = px.pie(flight_totals_df, values='PERCENTAGE', names='OP_UNIQUE_CARRIER', titl
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.show()
#fig.write_image("Individual Carrier Performance (2022)/fig2.pdf",engine='kaleido')
```



In [137…

```
#plt.pie(flight_totals_df.PERCENTAGE, labels = flight_totals_df.OP_UNIQUE_CARRIER,
#plt.title("Individual Carrier Performance (2022")
#plt.figure(figsize=(10,6))
#plt.show()
```

Flight Stats by Operating Carrier

In [48]:

```
fig=px.bar(airline_on_time_performance, x=airline_on_time_performance.OP_UNIQUE_CAR
fig.update_xaxes(tickangle=45)
fig.update_layout(autosize=False,width=900, height=700)
#fig.write_image("Airline On-Time Performance.pdf",engine='kaleido')
```

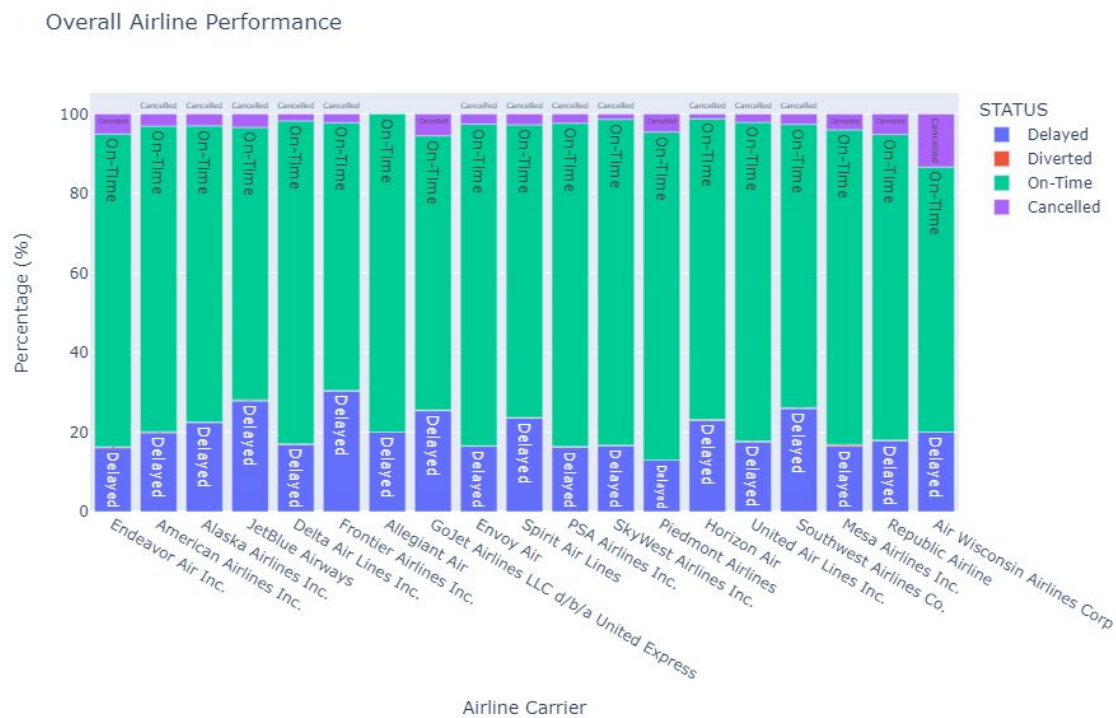
In [136…

```
#ax = sns.barplot(x='OP_UNIQUE_CARRIER', y='PERCENTAGE', data=airline_on_time_perfo
#sns.set(rc={'figure.figsize':(12,5)})

#for i in ax.containers:
#    ax.bar_label(i,)

#plt.xlabel("Predicted Values")
#plt.ylabel("Actual Values")
#plt.title("Individual Carrier Performance (2022")
```

In [50]:

```
fig = px.bar(flight_status_df, x="OP_UNIQUE_CARRIER_NAME", y="PERCENTAGE", title="O
             labels=dict(OP_UNIQUE_CARRIER_NAME="Airline Carrier", PERCENTAGE="Perce
fig.update_layout(autosize=False,width=900, height=600)
```

```
fig.show()
#fig.write_image("Overall Airline Performance.pdf",engine='kaleido')
```

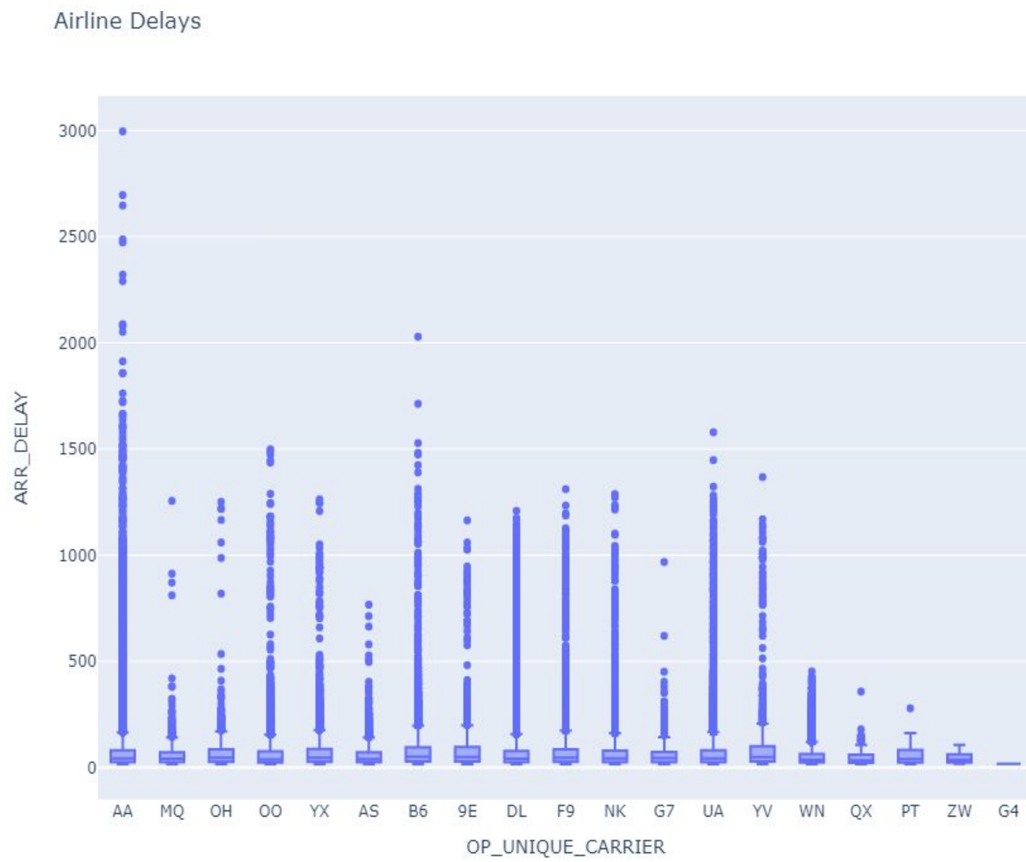


Delays

Overall Delays per carrier

In [111…

```
fig = px.box(delayed_arrival, x="OP_UNIQUE_CARRIER", y="ARR_DELAY", title="Airline
             labels=dict(OP_UNIQUE_CARRIER_NAME="Arrival Delay in minutes", PERCENT
fig.update_layout(autosize=False,width=900, height=700)
fig.show()
#fig.write_image("Airline Delays.pdf",engine='kaleido')
```

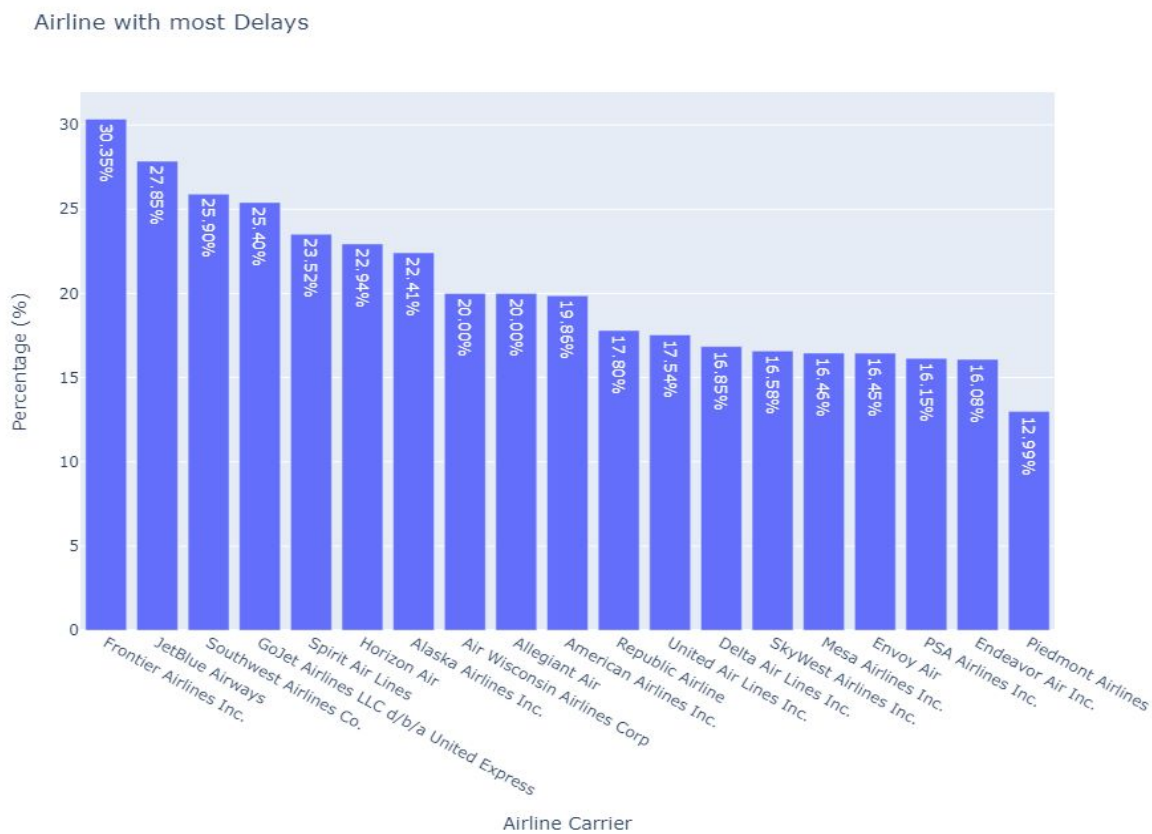


Which carrier has the most number of delays?

Carrier with most delays

In [112…

```
fig = px.bar(delayed_performance, x="OP_UNIQUE_CARRIER_NAME", y="PERCENTAGE", title
labels=dict(OP_UNIQUE_CARRIER_NAME="Airline Carrier", PERCENTAGE="Percentage (%)"))
fig.update_layout(autosize=False,width=900, height=700)
fig.show()
#fig.write_image("Airline with most Delays.pdf",engine='kaleido')
```

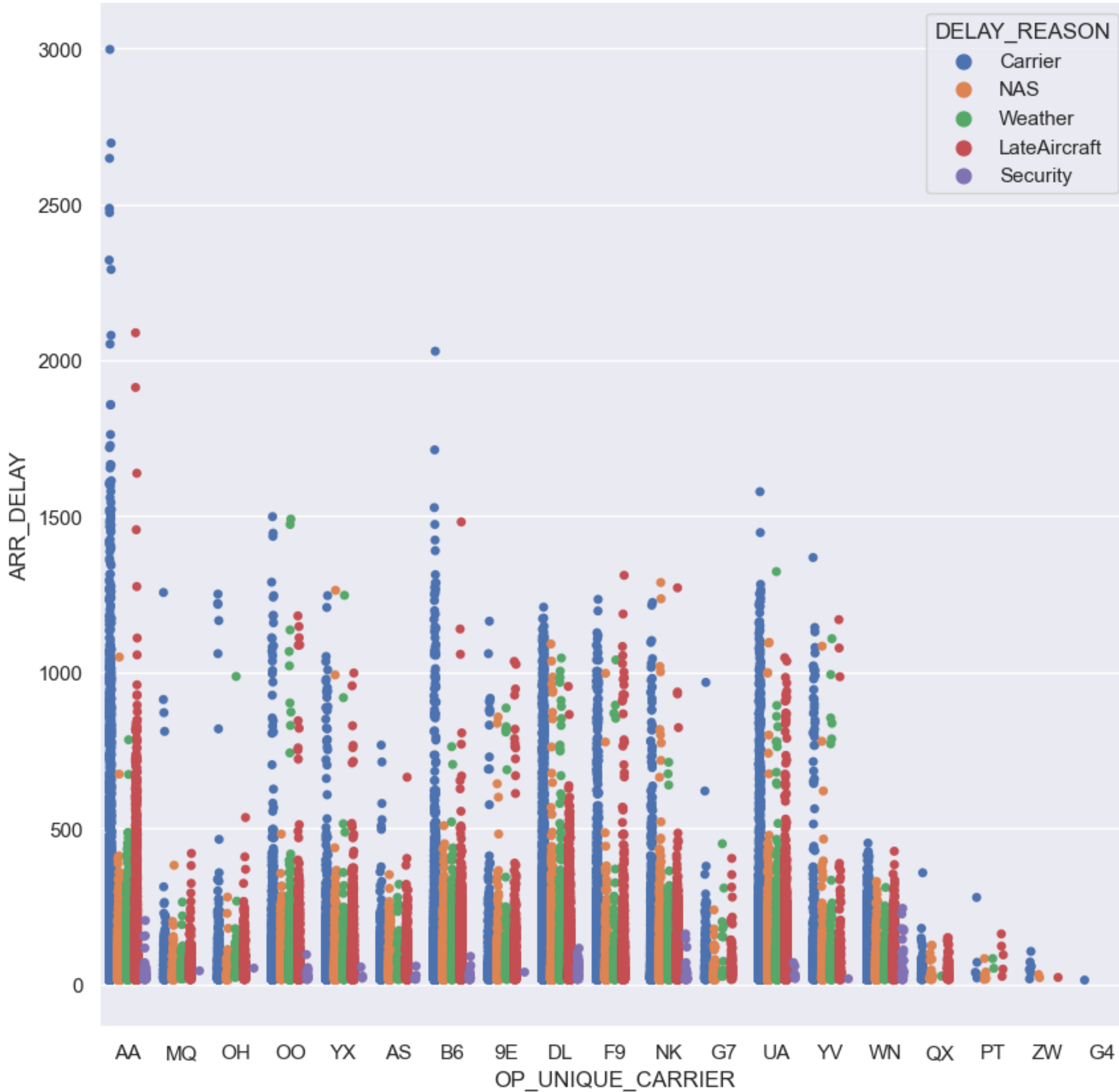


Frontier Airlines has the most number of delays, followed by JetBlue Airways. Piedmont and Endeavor air have the least delays.

Carriers vs Delay Reasons

In [113…

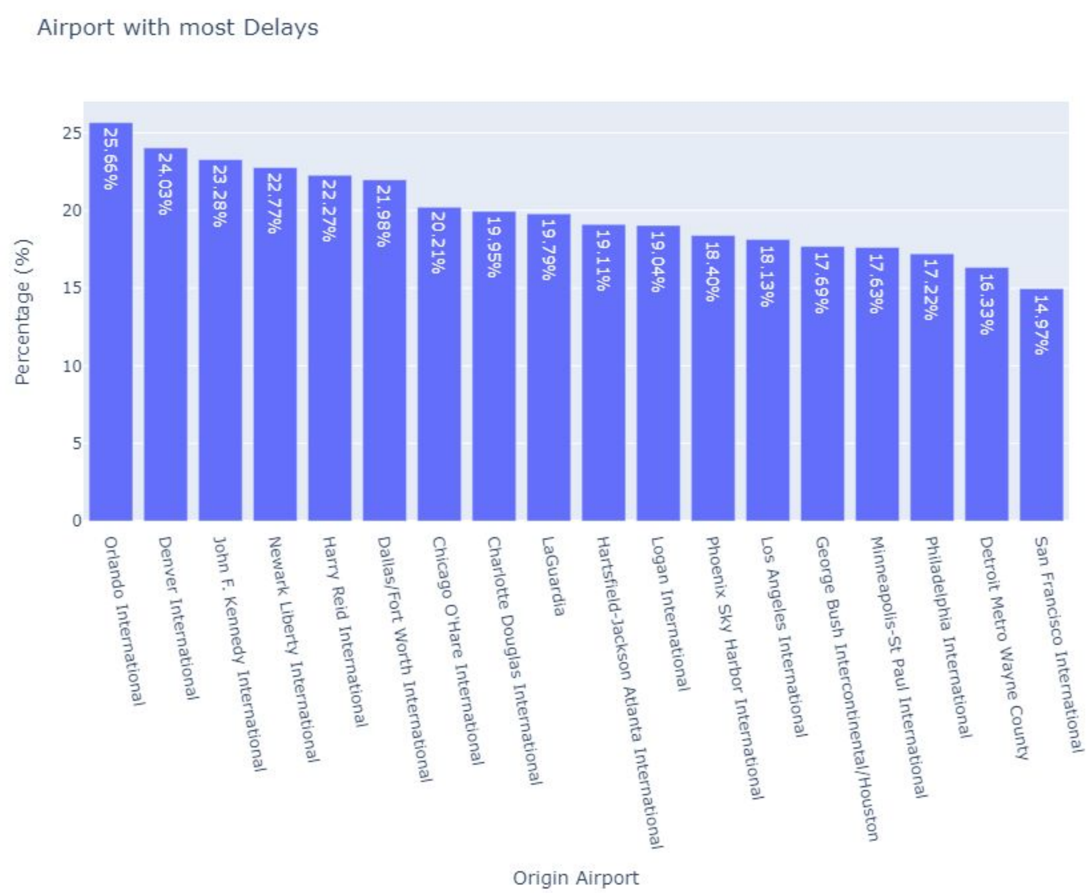
```
f, ax = plt.subplots(figsize=(10, 10))
sns.despine(bottom=True, left=True)
# Observations with Scatter Plot
sns.stripplot(x=delayed_arrival.OP_UNIQUE_CARRIER,y=delayed_arrival.ARR_DELAY,
              hue=delayed_arrival.DELAY_REASON,data = delayed_arrival, dodge=True,
plt.show()
```



Origin Airport vs Arrival Delays

In [114…

```
fig = px.bar(delayed_status_df[delayed_status_df.STATUS=="Delayed"], x="ORIGIN_AIRP
             title="Airport with most Delays",
             text=delayed_status_df[delayed_status_df.STATUS=="Delayed"].PERCENTAGE
             labels=dict(ORIGIN_AIRPORT_NAME="Origin Airport", PERCENTAGE="Percenta
fig.update_xaxes(tickangle=80)
fig.update_layout(autosize=False,width=900, height=700)
fig.show()
#fig.write_image("Airport with most Delays.pdf",engine='kaleido')
```

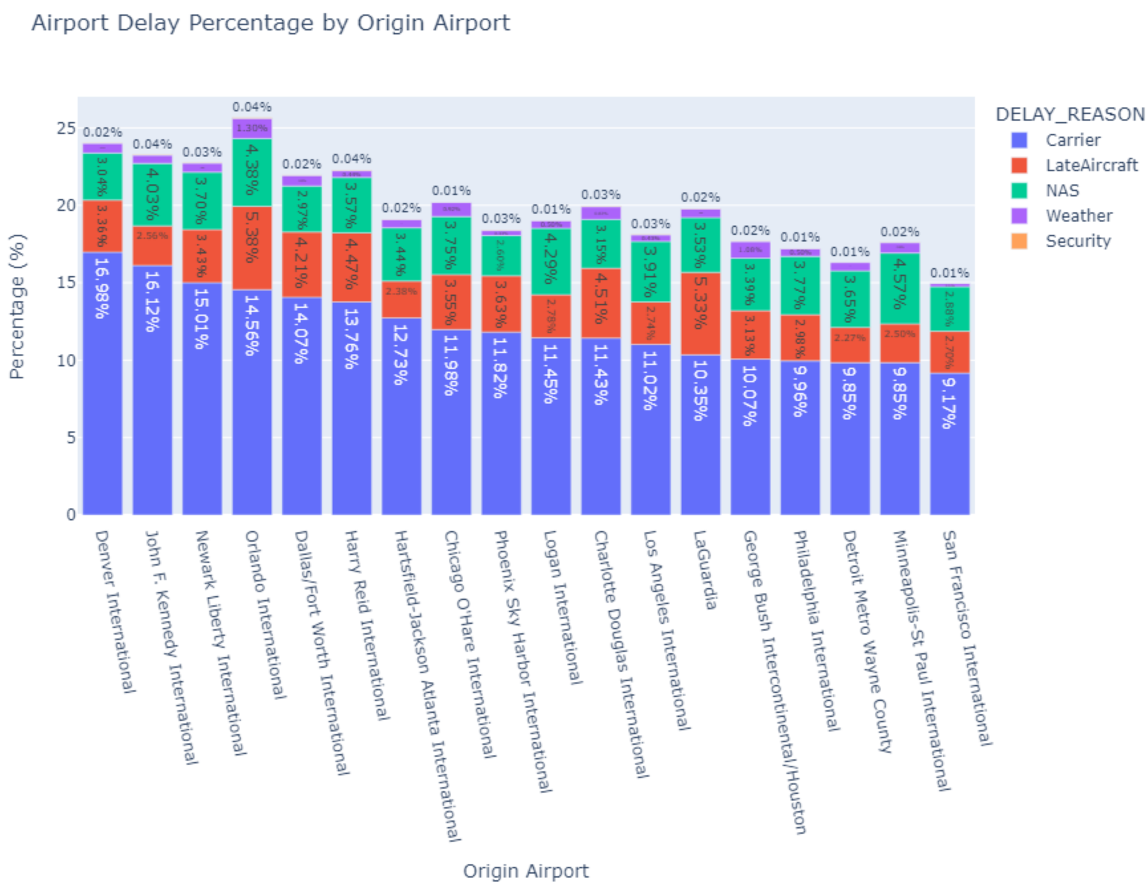


Orlando International has the most delays.

Origin Airport vs Delay Reasons

In [115…

```
fig = px.bar(delayed_status_reason_df[delayed_status_reason_df.STATUS=="Delayed"],
             color="DELAY_REASON",title="Airport Delay Percentage by Origin Airport
             text=delayed_status_reason_df[delayed_status_reason_df.STATUS=="Delaye
             labels=dict(ORIGIN_AIRPORT_NAME="Origin Airport", PERCENTAGE="Percenta
fig.update_xaxes(tickangle=80)
fig.update_layout(autosize=False,width=900, height=700)
fig.show()
#fig.write_image("Airport Delay Percentage by Origin Airport.pdf",engine='kaleido')
```

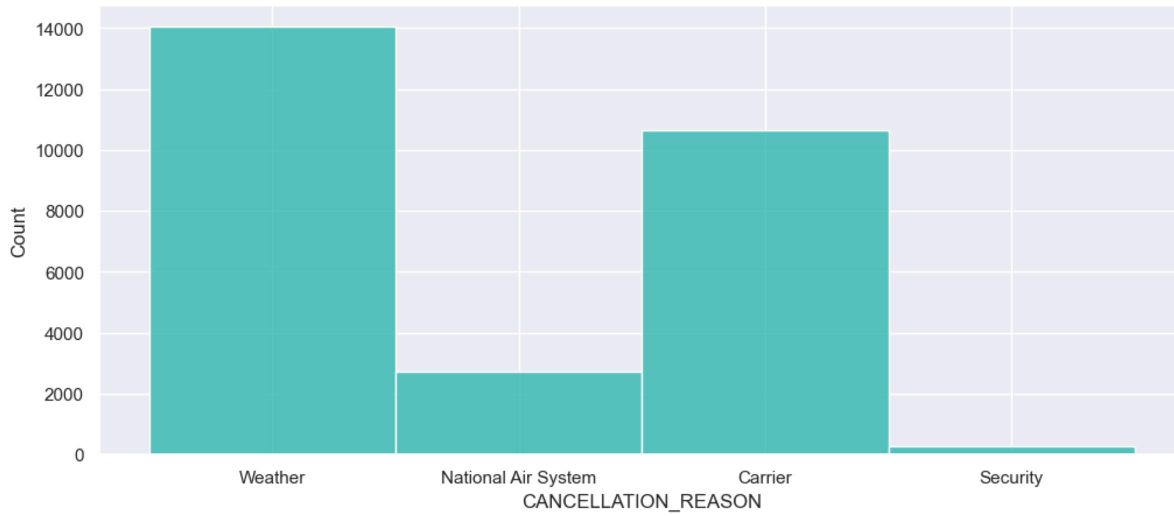


Cancellations

Overall cancellations

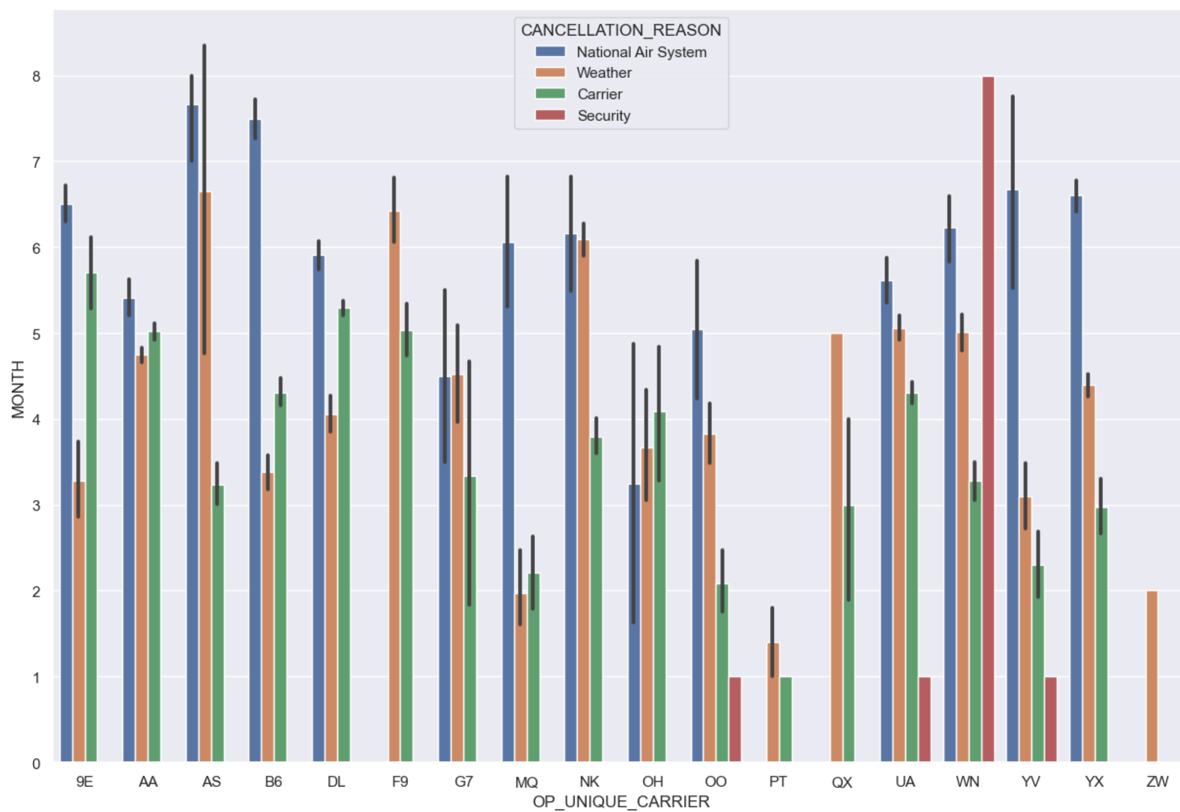
In [116…

```
sns.histplot(data=flight_data_df, x="CANCELLATION_REASON",color='lightseagreen')
plt.show()
```



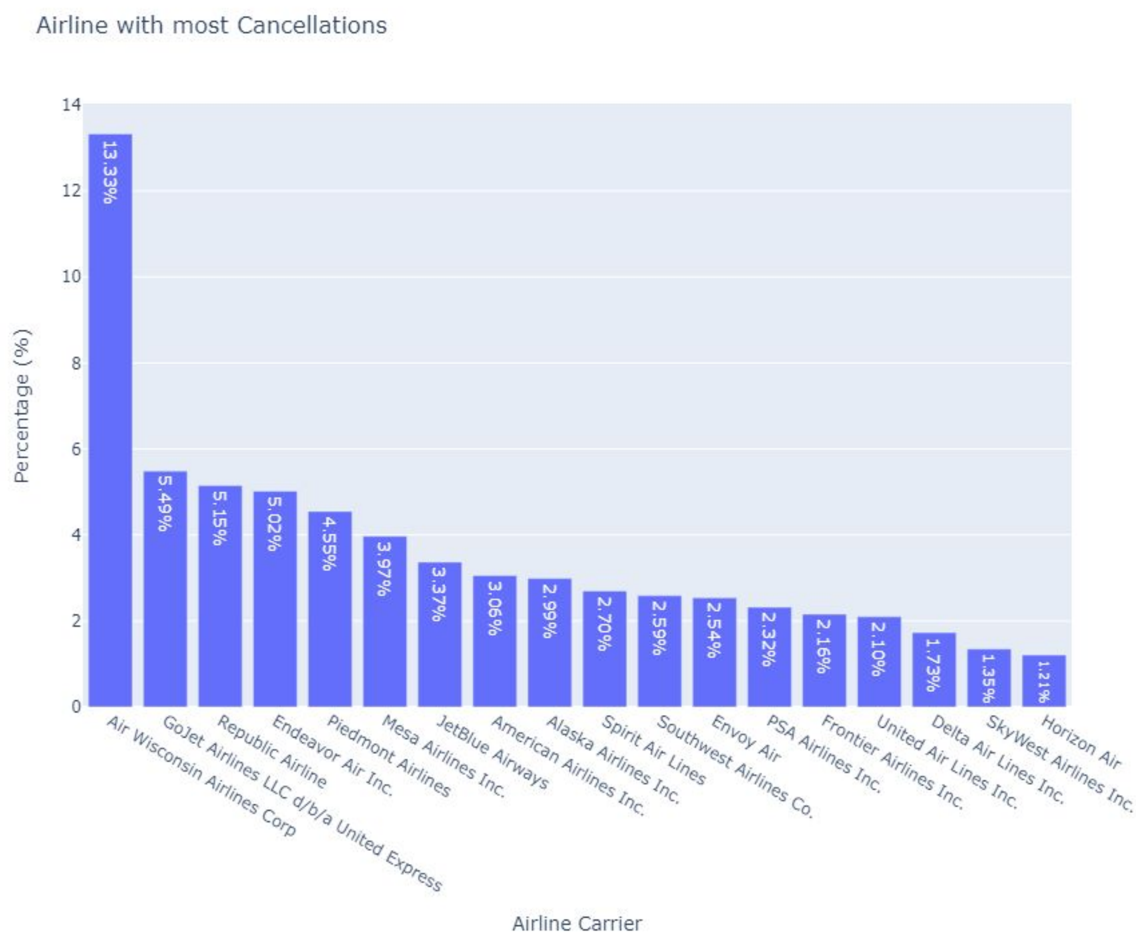
Carriers vs Cancellation Reasons

In [117…
```
cancelled_df = cancelled_df.sort_values('OP_UNIQUE_CARRIER',ascending=True)
f, ax = plt.subplots(figsize=(15, 10))
#sns.despine(bottom=True, left=True)
# Observations with Scatter Plot
sns.barplot(data=cancelled_df, y="MONTH", x="OP_UNIQUE_CARRIER", hue="CANCELLATION_
plt.show()
```



In [118…
```
cancelled_performance = flight_status_df[flight_status_df.STATUS == 'Cancelled'].so
cancelled_performance = cancelled_performance.sort_values('PERCENTAGE',ascending=Fa


fig = px.bar(cancelled_performance, x="OP_UNIQUE_CARRIER_NAME", y="PERCENTAGE", tit
```

```
labels=dict(OP_UNIQUE_CARRIER_NAME="Airline Carrier", PERCENTAGE="Percentage (%)"))
fig.update_layout(autosize=False,width=900, height=700)
fig.show()
#fig.write_image("Airline with most Cancellations.pdf",engine='kaleido')
```



Hypothesis Test

In [125…

```
def check_normality(data):
    test_stat_normality, p_value_normality=stats.shapiro(data)
    print("p value:%.4f" % p_value_normality)
    if p_value_normality <0.05:
        print("Reject null hypothesis >> The data is not normally distributed")
```

```
    else:
        print("Fail to reject null hypothesis >> The data is normally distributed")
```

In [126…

```
n = len(flight_data_df)
cnt=0
print('PEARSONS TEST')
iters = 10000
for _ in range(3):
    sample = thinkstats2.SampleRows(flight_data_df, n)

    testA = scipy.stats.pearsonr(sample.ARR_DELAY, sample.DEP_DELAY)
    print('****  ARR and DEP DELAY  ****','\n')
    print('Correlation Coefficient : ',testA.statistic)
    print('P VALUE :',testA.pvalue)
    print('CONFIDENCE :',testA.confidence_interval(confidence_level=0.99))

    testB = scipy.stats.pearsonr(sample.ARR_DELAY, sample.CARRIER_DELAY)
    print('\n','****  ARR and CARRIER DELAY  ****','\n')
    print('Correlation Coefficient :',testB.statistic)
    print('P VALUE :',testB.pvalue)
    print('CONFIDENCE :',testB.confidence_interval(confidence_level=0.99))

    n //= 2
```

```
PEARSONS TEST
****  ARR and DEP DELAY  ****

Correlation Coefficient :  0.95687758025145
P VALUE : 0.0
CONFIDENCE : ConfidenceInterval(low=0.9566672868572089, high=0.9570868754756202)

 ****  ARR and CARRIER DELAY  ****

Correlation Coefficient : 0.7179582983327268
P VALUE : 0.0
CONFIDENCE : ConfidenceInterval(low=0.7167515233685442, high=0.7191607729303732)
****  ARR and DEP DELAY  ****

Correlation Coefficient :  0.9585303515046257
P VALUE : 0.0
CONFIDENCE : ConfidenceInterval(low=0.9582438241210035, high=0.9588149541121659)

 ****  ARR and CARRIER DELAY  ****

Correlation Coefficient : 0.7156386570413412
P VALUE : 0.0
CONFIDENCE : ConfidenceInterval(low=0.7139190473808986, high=0.7173496348957548)
****  ARR and DEP DELAY  ****

Correlation Coefficient :  0.9553963130363297
P VALUE : 0.0
CONFIDENCE : ConfidenceInterval(low=0.9549605730921444, high=0.9558279326077482)

 ****  ARR and CARRIER DELAY  ****

Correlation Coefficient : 0.7073747246418528
P VALUE : 0.0
CONFIDENCE : ConfidenceInterval(low=0.7048817107511103, high=0.7098502628885948)
```

For both pairs - p value is 0 for all samples. The null hypothesis is rejected and your test is statistically significant. Correlation Coefficient is positive and closer to 1. There is a good relationship between the variables.

Regression Analysis

Predicting Carrier Delays when there is an Arrival Delay

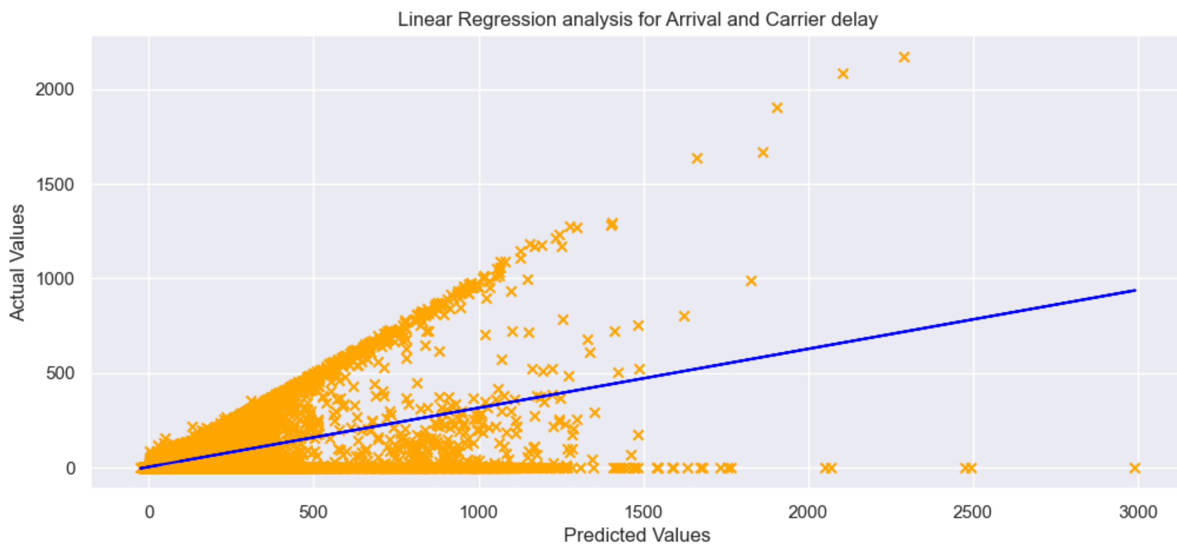
In [132…
```
delayed_arrival.loc[pd.isna(delayed_arrival.DEP_DELAY),"ARR_DELAY"] = 0
delayed_arrival.loc[pd.isna(delayed_arrival.LATE_AIRCRAFT_DELAY),"CARRIER_DELAY"] =


x = np.array(delayed_arrival.DEP_DELAY).reshape(-1, 1)
y = np.array(delayed_arrival.LATE_AIRCRAFT_DELAY).reshape(-1, 1)
#Input data into a single call for splitting (and optionally subsampling) data into
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3,random_st
```

In [133…
```
#Probability density functions of the target before and after applying the logarith
```

```
lr = slm.LinearRegression()
lr.fit(x_train,y_train)
predictions = lr.predict(x_test)
```

In [134…

```
plt.scatter(x_train,y_train, color="orange", marker='x')
plt.plot(x_train, lr.predict(x_train), color="Blue")
plt.xlabel("Predicted Values")
plt.ylabel("Actual Values")
plt.title("Linear Regression analysis for Arrival and Carrier delay")
plt.show()
```



In [135…

```
import sklearn.exceptions as sklexceptions

warnings.simplefilter("ignore", category=sklexceptions.DataConversionWarning)
warnings.simplefilter("ignore", category=sklexceptions.ConvergenceWarning)
warnings.simplefilter("ignore", category=sklexceptions.UndefinedMetricWarning)

logmodel = slm.LogisticRegression()
logmodel.fit(x_train,y_train)
predictions = logmodel.predict(x_test)
```

In [139…

```
print(classification_report(y_test,predictions))
```

	precision	recall	f1-score	support
0.0	0.55	0.99	0.71	35811
1.0	0.00	0.00	0.00	413
2.0	0.00	0.00	0.00	391
3.0	0.00	0.00	0.00	378
4.0	0.00	0.00	0.00	421
5.0	0.00	0.00	0.00	391
6.0	0.00	0.00	0.00	414
7.0	0.00	0.00	0.00	473
8.0	0.00	0.00	0.00	446
9.0	0.00	0.00	0.00	426
10.0	0.00	0.00	0.00	445
11.0	0.00	0.00	0.00	426
12.0	0.00	0.00	0.00	444
13.0	0.00	0.00	0.00	471
14.0	0.00	0.00	0.00	493
15.0	0.00	0.00	0.00	506
16.0	0.00	0.00	0.00	589
17.0	0.00	0.00	0.00	586
18.0	0.00	0.00	0.00	546
19.0	0.00	0.00	0.00	522
20.0	0.00	0.00	0.00	493
21.0	0.00	0.00	0.00	504
22.0	0.00	0.00	0.00	479
23.0	0.00	0.00	0.00	484
24.0	0.00	0.00	0.00	433
25.0	0.00	0.00	0.00	440
26.0	0.00	0.00	0.00	417
27.0	0.00	0.00	0.00	410
28.0	0.00	0.00	0.00	367
29.0	0.00	0.00	0.00	377
30.0	0.00	0.00	0.00	359
31.0	0.00	0.00	0.00	357
32.0	0.00	0.00	0.00	320
33.0	0.00	0.00	0.00	360
34.0	0.00	0.00	0.00	271
35.0	0.00	0.00	0.00	290
36.0	0.00	0.00	0.00	302
37.0	0.00	0.00	0.00	280
38.0	0.00	0.00	0.00	291
39.0	0.00	0.00	0.00	257
40.0	0.00	0.00	0.00	254
41.0	0.00	0.00	0.00	263
42.0	0.00	0.00	0.00	253
43.0	0.00	0.00	0.00	231
44.0	0.00	0.00	0.00	246
45.0	0.00	0.00	0.00	235
46.0	0.00	0.00	0.00	247
47.0	0.00	0.00	0.00	198
48.0	0.00	0.00	0.00	226
49.0	0.00	0.00	0.00	196
50.0	0.00	0.00	0.00	210
51.0	0.00	0.00	0.00	199
52.0	0.00	0.00	0.00	193
53.0	0.00	0.00	0.00	163

54.0	0.00	0.00	0.00	169
55.0	0.00	0.00	0.00	158
56.0	0.00	0.00	0.00	183
57.0	0.00	0.00	0.00	194
58.0	0.00	0.00	0.00	157
59.0	0.00	0.00	0.00	162
60.0	0.00	0.00	0.00	154
61.0	0.00	0.00	0.00	159
62.0	0.00	0.00	0.00	135
63.0	0.00	0.00	0.00	144
64.0	0.00	0.00	0.00	145
65.0	0.00	0.00	0.00	127
66.0	0.00	0.00	0.00	129
67.0	0.00	0.00	0.00	140
68.0	0.00	0.00	0.00	115
69.0	0.00	0.00	0.00	134
70.0	0.00	0.00	0.00	139
71.0	0.00	0.00	0.00	120
72.0	0.00	0.00	0.00	122
73.0	0.00	0.00	0.00	124
74.0	0.00	0.00	0.00	103
75.0	0.00	0.00	0.00	120
76.0	0.00	0.00	0.00	126
77.0	0.00	0.00	0.00	115
78.0	0.00	0.00	0.00	110
79.0	0.00	0.00	0.00	108
80.0	0.00	0.00	0.00	102
81.0	0.00	0.00	0.00	96
82.0	0.00	0.00	0.00	97
83.0	0.00	0.00	0.00	89
84.0	0.00	0.00	0.00	95
85.0	0.00	0.00	0.00	95
86.0	0.00	0.00	0.00	100
87.0	0.00	0.00	0.00	90
88.0	0.00	0.00	0.00	90
89.0	0.00	0.00	0.00	71
90.0	0.00	0.00	0.00	83
91.0	0.00	0.00	0.00	72
92.0	0.00	0.00	0.00	76
93.0	0.00	0.00	0.00	83
94.0	0.00	0.00	0.00	55
95.0	0.00	0.00	0.00	80
96.0	0.00	0.00	0.00	54
97.0	0.00	0.00	0.00	89
98.0	0.00	0.00	0.00	75
99.0	0.00	0.00	0.00	76
100.0	0.00	0.00	0.00	70
101.0	0.00	0.00	0.00	65
102.0	0.00	0.00	0.00	63
103.0	0.00	0.00	0.00	80
104.0	0.00	0.00	0.00	55
105.0	0.00	0.00	0.00	70
106.0	0.00	0.00	0.00	77
107.0	0.00	0.00	0.00	53
108.0	0.00	0.00	0.00	58
109.0	0.00	0.00	0.00	51

110.0	0.00	0.00	0.00	60
111.0	0.00	0.00	0.00	50
112.0	0.00	0.00	0.00	52
113.0	0.00	0.00	0.00	44
114.0	0.00	0.00	0.00	47
115.0	0.00	0.00	0.00	44
116.0	0.00	0.00	0.00	42
117.0	0.00	0.00	0.00	50
118.0	0.00	0.00	0.00	45
119.0	0.00	0.00	0.00	50
120.0	0.00	0.00	0.00	43
121.0	0.00	0.00	0.00	58
122.0	0.00	0.00	0.00	38
123.0	0.00	0.00	0.00	45
124.0	0.00	0.00	0.00	44
125.0	0.00	0.00	0.00	40
126.0	0.00	0.00	0.00	37
127.0	0.00	0.00	0.00	34
128.0	0.00	0.00	0.00	37
129.0	0.00	0.00	0.00	40
130.0	0.00	0.00	0.00	29
131.0	0.00	0.00	0.00	33
132.0	0.00	0.00	0.00	42
133.0	0.00	0.00	0.00	44
134.0	0.00	0.00	0.00	36
135.0	0.00	0.00	0.00	31
136.0	0.00	0.00	0.00	35
137.0	0.00	0.00	0.00	28
138.0	0.00	0.00	0.00	35
139.0	0.00	0.00	0.00	37
140.0	0.00	0.00	0.00	27
141.0	0.00	0.00	0.00	27
142.0	0.00	0.00	0.00	27
143.0	0.00	0.00	0.00	44
144.0	0.00	0.00	0.00	26
145.0	0.00	0.00	0.00	27
146.0	0.00	0.00	0.00	44
147.0	0.00	0.00	0.00	39
148.0	0.00	0.00	0.00	28
149.0	0.00	0.00	0.00	24
150.0	0.00	0.00	0.00	27
151.0	0.00	0.00	0.00	28
152.0	0.00	0.00	0.00	18
153.0	0.00	0.00	0.00	21
154.0	0.00	0.00	0.00	23
155.0	0.00	0.00	0.00	24
156.0	0.00	0.00	0.00	30
157.0	0.00	0.00	0.00	25
158.0	0.00	0.00	0.00	18
159.0	0.00	0.00	0.00	27
160.0	0.00	0.00	0.00	19
161.0	0.00	0.00	0.00	25
162.0	0.00	0.00	0.00	28
163.0	0.00	0.00	0.00	25
164.0	0.00	0.00	0.00	14
165.0	0.00	0.00	0.00	22

166.0	0.00	0.00	0.00	18
167.0	0.00	0.00	0.00	19
168.0	0.00	0.00	0.00	20
169.0	0.00	0.00	0.00	19
170.0	0.00	0.00	0.00	24
171.0	0.00	0.00	0.00	21
172.0	0.00	0.00	0.00	19
173.0	0.00	0.00	0.00	18
174.0	0.00	0.00	0.00	11
175.0	0.00	0.00	0.00	24
176.0	0.00	0.00	0.00	12
177.0	0.00	0.00	0.00	15
178.0	0.00	0.00	0.00	17
179.0	0.00	0.00	0.00	13
180.0	0.00	0.00	0.00	16
181.0	0.00	0.00	0.00	21
182.0	0.00	0.00	0.00	21
183.0	0.00	0.00	0.00	18
184.0	0.00	0.00	0.00	24
185.0	0.00	0.00	0.00	19
186.0	0.00	0.00	0.00	20
187.0	0.00	0.00	0.00	17
188.0	0.00	0.00	0.00	15
189.0	0.00	0.00	0.00	20
190.0	0.00	0.00	0.00	15
191.0	0.00	0.00	0.00	9
192.0	0.00	0.00	0.00	21
193.0	0.00	0.00	0.00	13
194.0	0.00	0.00	0.00	15
195.0	0.00	0.00	0.00	11
196.0	0.00	0.00	0.00	18
197.0	0.00	0.00	0.00	17
198.0	0.00	0.00	0.00	16
199.0	0.00	0.00	0.00	12
200.0	0.00	0.00	0.00	16
201.0	0.00	0.00	0.00	14
202.0	0.00	0.00	0.00	11
203.0	0.00	0.00	0.00	16
204.0	0.00	0.00	0.00	15
205.0	0.00	0.00	0.00	14
206.0	0.00	0.00	0.00	7
207.0	0.00	0.00	0.00	9
208.0	0.00	0.00	0.00	9
209.0	0.00	0.00	0.00	7
210.0	0.00	0.00	0.00	13
211.0	0.00	0.00	0.00	11
212.0	0.00	0.00	0.00	13
213.0	0.00	0.00	0.00	17
214.0	0.00	0.00	0.00	12
215.0	0.00	0.00	0.00	8
216.0	0.00	0.00	0.00	5
217.0	0.00	0.00	0.00	15
218.0	0.00	0.00	0.00	7
219.0	0.00	0.00	0.00	9
220.0	0.00	0.00	0.00	6
221.0	0.00	0.00	0.00	6

222.0	0.00	0.00	0.00	9
223.0	0.00	0.00	0.00	9
224.0	0.00	0.00	0.00	11
225.0	0.00	0.00	0.00	13
226.0	0.00	0.00	0.00	9
227.0	0.00	0.00	0.00	11
228.0	0.00	0.00	0.00	12
229.0	0.00	0.00	0.00	5
230.0	0.00	0.00	0.00	10
231.0	0.00	0.00	0.00	11
232.0	0.00	0.00	0.00	9
233.0	0.00	0.00	0.00	8
234.0	0.00	0.00	0.00	9
235.0	0.00	0.00	0.00	12
236.0	0.00	0.00	0.00	8
237.0	0.00	0.00	0.00	11
238.0	0.00	0.00	0.00	3
239.0	0.00	0.00	0.00	5
240.0	0.00	0.00	0.00	8
241.0	0.00	0.00	0.00	14
242.0	0.00	0.00	0.00	5
243.0	0.00	0.00	0.00	4
244.0	0.00	0.00	0.00	5
245.0	0.00	0.00	0.00	8
246.0	0.00	0.00	0.00	4
247.0	0.00	0.00	0.00	7
248.0	0.00	0.00	0.00	5
249.0	0.00	0.00	0.00	3
250.0	0.00	0.00	0.00	11
251.0	0.00	0.00	0.00	7
252.0	0.00	0.00	0.00	8
253.0	0.00	0.00	0.00	5
254.0	0.00	0.00	0.00	7
255.0	0.00	0.00	0.00	5
256.0	0.00	0.00	0.00	8
257.0	0.00	0.00	0.00	2
258.0	0.00	0.00	0.00	7
259.0	0.00	0.00	0.00	3
260.0	0.00	0.00	0.00	5
261.0	0.00	0.00	0.00	9
262.0	0.00	0.00	0.00	6
263.0	0.00	0.00	0.00	1
264.0	0.00	0.00	0.00	4
265.0	0.00	0.00	0.00	5
266.0	0.00	0.00	0.00	3
267.0	0.00	0.00	0.00	5
268.0	0.00	0.00	0.00	1
269.0	0.00	0.00	0.00	3
270.0	0.00	0.00	0.00	5
271.0	0.00	0.00	0.00	1
272.0	0.00	0.00	0.00	4
273.0	0.00	0.00	0.00	6
274.0	0.00	0.00	0.00	2
275.0	0.00	0.00	0.00	5
276.0	0.00	0.00	0.00	8
277.0	0.00	0.00	0.00	6

278.0	0.00	0.00	0.00	5
279.0	0.00	0.00	0.00	3
280.0	0.00	0.00	0.00	5
282.0	0.00	0.00	0.00	2
283.0	0.00	0.00	0.00	1
284.0	0.00	0.00	0.00	4
285.0	0.00	0.00	0.00	6
286.0	0.00	0.00	0.00	3
287.0	0.00	0.00	0.00	6
288.0	0.00	0.00	0.00	4
289.0	0.00	0.00	0.00	1
290.0	0.00	0.00	0.00	7
292.0	0.00	0.00	0.00	4
293.0	0.00	0.00	0.00	3
295.0	0.00	0.00	0.00	1
296.0	0.00	0.00	0.00	2
297.0	0.00	0.00	0.00	6
298.0	0.00	0.00	0.00	8
299.0	0.00	0.00	0.00	1
300.0	0.00	0.00	0.00	3
301.0	0.00	0.00	0.00	4
302.0	0.00	0.00	0.00	4
303.0	0.00	0.00	0.00	4
304.0	0.00	0.00	0.00	4
305.0	0.00	0.00	0.00	1
306.0	0.00	0.00	0.00	3
307.0	0.00	0.00	0.00	2
308.0	0.00	0.00	0.00	4
310.0	0.00	0.00	0.00	5
311.0	0.00	0.00	0.00	6
312.0	0.00	0.00	0.00	1
313.0	0.00	0.00	0.00	1
314.0	0.00	0.00	0.00	3
315.0	0.00	0.00	0.00	1
316.0	0.00	0.00	0.00	4
317.0	0.00	0.00	0.00	2
318.0	0.00	0.00	0.00	3
319.0	0.00	0.00	0.00	1
320.0	0.00	0.00	0.00	3
321.0	0.00	0.00	0.00	3
322.0	0.00	0.00	0.00	2
323.0	0.00	0.00	0.00	2
324.0	0.00	0.00	0.00	2
325.0	0.00	0.00	0.00	3
326.0	0.00	0.00	0.00	5
330.0	0.00	0.00	0.00	2
332.0	0.00	0.00	0.00	1
334.0	0.00	0.00	0.00	3
335.0	0.00	0.00	0.00	3
336.0	0.00	0.00	0.00	1
337.0	0.00	0.00	0.00	2
338.0	0.00	0.00	0.00	1
339.0	0.00	0.00	0.00	1
340.0	0.00	0.00	0.00	1
343.0	0.00	0.00	0.00	2
344.0	0.00	0.00	0.00	3

346.0	0.00	0.00	0.00	3
347.0	0.00	0.00	0.00	1
348.0	0.00	0.00	0.00	3
349.0	0.00	0.00	0.00	2
350.0	0.00	0.00	0.00	2
351.0	0.00	0.00	0.00	1
352.0	0.00	0.00	0.00	2
355.0	0.00	0.00	0.00	3
357.0	0.00	0.00	0.00	4
358.0	0.00	0.00	0.00	1
359.0	0.00	0.00	0.00	2
360.0	0.00	0.00	0.00	1
364.0	0.00	0.00	0.00	3
366.0	0.00	0.00	0.00	1
367.0	0.00	0.00	0.00	3
368.0	0.00	0.00	0.00	4
369.0	0.00	0.00	0.00	3
371.0	0.00	0.00	0.00	2
372.0	0.00	0.00	0.00	2
373.0	0.00	0.00	0.00	3
374.0	0.00	0.00	0.00	2
376.0	0.00	0.00	0.00	2
378.0	0.00	0.00	0.00	1
379.0	0.00	0.00	0.00	4
380.0	0.00	0.00	0.00	1
382.0	0.00	0.00	0.00	1
384.0	0.00	0.00	0.00	2
385.0	0.00	0.00	0.00	4
386.0	0.00	0.00	0.00	1
387.0	0.00	0.00	0.00	3
388.0	0.00	0.00	0.00	2
389.0	0.00	0.00	0.00	1
390.0	0.00	0.00	0.00	1
391.0	0.00	0.00	0.00	1
392.0	0.00	0.00	0.00	2
394.0	0.00	0.00	0.00	1
395.0	0.00	0.00	0.00	1
396.0	0.00	0.00	0.00	1
397.0	0.00	0.00	0.00	1
398.0	0.00	0.00	0.00	1
399.0	0.00	0.00	0.00	1
400.0	0.00	0.00	0.00	1
401.0	0.00	0.00	0.00	4
404.0	0.00	0.00	0.00	1
405.0	0.00	0.00	0.00	1
406.0	0.00	0.00	0.00	1
411.0	0.00	0.00	0.00	1
412.0	0.00	0.00	0.00	1
414.0	0.00	0.00	0.00	1
415.0	0.00	0.00	0.00	1
417.0	0.00	0.00	0.00	1
418.0	0.00	0.00	0.00	1
419.0	0.00	0.00	0.00	1
420.0	0.00	0.00	0.00	1
421.0	0.00	0.00	0.00	1
424.0	0.00	0.00	0.00	1

426.0	0.00	0.00	0.00	2
431.0	0.00	0.00	0.00	1
433.0	0.00	0.00	0.00	1
434.0	0.00	0.00	0.00	1
435.0	0.00	0.00	0.00	1
440.0	0.00	0.00	0.00	2
442.0	0.00	0.00	0.00	3
443.0	0.00	0.00	0.00	1
444.0	0.00	0.00	0.00	1
446.0	0.00	0.00	0.00	2
448.0	0.00	0.00	0.00	1
449.0	0.00	0.00	0.00	1
450.0	0.00	0.00	0.00	1
459.0	0.00	0.00	0.00	1
463.0	0.00	0.00	0.00	1
467.0	0.00	0.00	0.00	2
469.0	0.00	0.00	0.00	1
470.0	0.00	0.00	0.00	1
474.0	0.00	0.00	0.00	1
475.0	0.00	0.00	0.00	1
476.0	0.00	0.00	0.00	1
482.0	0.00	0.00	0.00	2
483.0	0.00	0.00	0.00	1
492.0	0.00	0.00	0.00	1
493.0	0.00	0.00	0.00	1
495.0	0.00	0.00	0.00	1
496.0	0.00	0.00	0.00	1
497.0	0.00	0.00	0.00	1
498.0	0.00	0.00	0.00	1
501.0	0.00	0.00	0.00	1
503.0	0.00	0.00	0.00	1
509.0	0.00	0.00	0.00	2
512.0	0.00	0.00	0.00	1
516.0	0.00	0.00	0.00	1
520.0	0.00	0.00	0.00	1
522.0	0.00	0.00	0.00	1
528.0	0.00	0.00	0.00	1
530.0	0.00	0.00	0.00	1
531.0	0.00	0.00	0.00	1
534.0	0.00	0.00	0.00	1
536.0	0.00	0.00	0.00	1
538.0	0.00	0.00	0.00	1
540.0	0.00	0.00	0.00	2
542.0	0.00	0.00	0.00	2
543.0	0.00	0.00	0.00	2
544.0	0.00	0.00	0.00	2
545.0	0.00	0.00	0.00	1
554.0	0.00	0.00	0.00	2
556.0	0.00	0.00	0.00	1
557.0	0.00	0.00	0.00	1
562.0	0.00	0.00	0.00	1
563.0	0.00	0.00	0.00	1
568.0	0.00	0.00	0.00	1
569.0	0.00	0.00	0.00	1
572.0	0.00	0.00	0.00	2
575.0	0.00	0.00	0.00	2

577.0	0.00	0.00	0.00	1
581.0	0.00	0.00	0.00	1
582.0	0.00	0.00	0.00	1
587.0	0.00	0.00	0.00	1
598.0	0.00	0.00	0.00	1
602.0	0.00	0.00	0.00	3
605.0	0.00	0.00	0.00	1
609.0	0.00	0.00	0.00	1
612.0	0.00	0.00	0.00	1
619.0	0.00	0.00	0.00	1
620.0	0.00	0.00	0.00	1
627.0	0.00	0.00	0.00	1
637.0	0.00	0.00	0.00	1
638.0	0.00	0.00	0.00	1
641.0	0.00	0.00	0.00	1
645.0	0.00	0.00	0.00	1
647.0	0.00	0.00	0.00	1
649.0	0.00	0.00	0.00	1
650.0	0.00	0.00	0.00	1
652.0	0.00	0.00	0.00	1
655.0	0.00	0.00	0.00	1
657.0	0.00	0.00	0.00	1
658.0	0.00	0.00	0.00	2
662.0	0.00	0.00	0.00	1
665.0	0.00	0.00	0.00	1
667.0	0.00	0.00	0.00	1
671.0	0.00	0.00	0.00	1
679.0	0.00	0.00	0.00	1
680.0	0.00	0.00	0.00	1
702.0	0.00	0.00	0.00	1
711.0	0.00	0.00	0.00	1
717.0	0.00	0.00	0.00	1
719.0	0.00	0.00	0.00	1
722.0	0.00	0.00	0.00	1
724.0	0.00	0.00	0.00	1
733.0	0.00	0.00	0.00	1
755.0	0.00	0.00	0.00	1
758.0	0.00	0.00	0.00	2
759.0	0.00	0.00	0.00	1
761.0	0.00	0.00	0.00	2
769.0	0.00	0.00	0.00	1
773.0	0.00	0.00	0.00	1
778.0	0.00	0.00	0.00	1
780.0	0.00	0.00	0.00	1
784.0	0.00	0.00	0.00	1
799.0	0.00	0.00	0.00	1
800.0	0.00	0.00	0.00	1
804.0	0.00	0.00	0.00	1
806.0	0.00	0.00	0.00	1
812.0	0.00	0.00	0.00	1
822.0	0.00	0.00	0.00	2
824.0	0.00	0.00	0.00	1
836.0	0.00	0.00	0.00	1
843.0	0.00	0.00	0.00	1
848.0	0.00	0.00	0.00	1
849.0	0.00	0.00	0.00	1

```
       861.0       0.00      0.00      0.00         1
       883.0       0.00      0.00      0.00         1
       886.0       0.00      0.00      0.00         1
       902.0       0.00      0.00      0.00         1
       905.0       0.00      0.00      0.00         1
       908.0       0.00      0.00      0.00         1
       914.0       0.00      0.00      0.00         1
       919.0       0.00      0.00      0.00         1
       928.0       0.00      0.00      0.00         1
       940.0       0.00      0.00      0.00         1
       941.0       0.00      0.00      0.00         1
       950.0       0.00      0.00      0.00         1
       998.0       0.00      0.00      0.00         1
      1004.0       0.00      0.00      0.00         1
      1018.0       0.00      0.00      0.00         1
      1026.0       0.00      0.00      0.00         1
      1028.0       0.00      0.00      0.00         1
      1078.0       0.00      0.00      0.00         1
      1088.0       0.00      0.00      0.00         1
      1107.0       0.00      0.00      0.00         1
      1187.0       0.00      0.00      0.00         1
      1362.0       0.00      0.00      0.00         1
      1454.0       0.00      0.00      0.00         1
      2050.0       0.00      0.00      0.00         1

    accuracy                           0.55     64657
   macro avg       0.00      0.00      0.00     64657
weighted avg       0.31      0.55      0.39     64657
```

Precision - What percent of your predictions were correct?

Recall — What percent of the positive cases did you catch?

F1 score — What percent of positive predictions were correct?

Support - Support is the number of actual occurrences of the class in the specified dataset.

OUTCOMES

1. Are small carriers reliable in terms of lesser cancellations and delays?

   Answer: Frontier has the maximum number of delays whereas Piedmont has the least delays. It is unclear if small carriers are more reliable.

2. Which carrier has the best on-time performance?

   Answer: American Airlines Inc, Delta Airlines, and United Airlines have the best performance.

3. Which carrier has the least on-time performance?

   Answer: Allegiant Air, Air Wisconsin Airlines Corp, Piedmont Airlines, Horizon Air , and GoJet Airlines LLC have the least on-time performance

4. Identifying the most common cancellation reason for all carriers.

   Answer: Based on the 1 million rows of data, weather cancellations are the most common.

5. Which carrier has the most cancellations?

   Answer: Air Wisconsin has the most cancellations.

6. Which carrier has the most number of delays?

   Answer: Frontier Airlines has the most delays.

LIMITATIONS

1. The dataset used for this analysis has around 6 million rows. For purposes of analysis, I stripped data to 1 million rows. The outcomes mentioned could change with more data. Restricting the analysis to major airports could be omitting
   many performance aspects of airlines. It would be nice to run the analysis with years of data to average the findings.

2. The huge size of the dataset made the process extremely slow with multiple application crashes.

3. Moreover, another inherent challenge of the dataset was that there were limited variables that could be used. Many columns were inapplicable to the analysis (i.e. TAXI_OUT, TAXI_IN, AIR_TIME, etc.), so the analysis was done on limited variables. Additional information such as weather, NAS issue, etc., could open more areas for analysis.

QUESTIONS

It is unclear if I would be able to recommend the right area of focus for better performance, to the airlines. Delays are high

For example: If the majority of delays are due to NAS - National Air System Delay, it could mean there was an issue in one or more areas such as mechanical, crew, airport operations, etc. I would need to identify another dataset that logs the maintenance or operational issues by the carrier. This information could be hard to get as it is carrier specific and probably not allowed to be made public.

CONCLUSION

Analyzing this dataset was a very interesting project for me. I found myself surprised in several instances. I assumed most cancellations would be because of weather but on adding more parameters in the process of data cleaning, I noticed that most cancellations are actually due to Carriers and not weather. I wasn't able to show this in the analysis due to data size restrictions.

It was a great experience in understanding how to work with datasets and understanding the significance of each step. As next steps, I would like to calculate the delay percentage of flights at each interval of arrival delay, such as (0-15, <15, >15 - <30, >30) to validate the average delay time.