Data Exploration and Preparation

REPORT

1A. INITIAL DATA EXPLORATION

1. Attributes that include codes and names are of the OBJECT datatype since these are characters with digits that include the attributes:
   O_AIRPORT_IATA_CODE
   O_AIRPORT
   O_CITY
   O_STATE
   O_COUNTRY
   D_AIRPORT_IATA_CODE
   D_AIRPORT
   D_CITY
   D_STATE
   D_COUNTRY
   CANCELLATION_REASON
   AIRLINE_NAME
   There is a DATE attribute that also is listed as an OBJECT datatype however it can be converted to DATETIME datatype. This helps to easily perform visualizations on the data.

   There are attributes of the 'FLOAT' datatype:
   O_LATITUDE
   O_LONGITUDE
   D_LATITUDE
   D_LONGITUDE
   DEPARTURE_TIME
   DEPARTURE_DELAY
   TAXI_OUT
   WHEELS_OFF
   SCHEDULED_TIME
   ELAPSED_TIME
   AIR_TIME
   WHEELS_ON
   TAXI_IN
   ARRIVAL_TIME
   ARRIVAL_DELAY
   AIR_SYSTEM_DELAY
   SECURITY_DELAY
   AIRLINE_DELAY
   LATE_AIRCRAFT_DELAY
   WEATHER_DELAY
   The first four attributes indicate the latitude and longitude of the origin and destination of the flights whereas the rest of the attributes indicate time in minutes which might either be departure or arrival time, or delay caused during arrival, departure or due to the weather, security etc.

   There also are attributes of the INTEGER datatype:

SCHEDULED_DEPARTURE
DISTANCE
SCHEDULED_ARRIVAL
There are two other attributes DIVERTED and CANCELLED which are also listed as INTEGER however they can be converted to CATEGORY values since they take values 0 or 1 which represent two categories and they have no order to them.

2. A snapshot from Colab indicating the frequency, location and spread of a few attributes is shown below:

```
] pd.DataFrame(flight_df.describe())
```

	O_LATITUDE	O_LONGITUDE	D_LATITUDE	D_LONGITUDE	SCHEDULED_DEPARTURE	DEPARTURE_TIME	DEPARTURE_DELAY	TAXI_OUT	WHEELS_OFF
count	2276.000000	2276.000000	2278.000000	2278.000000	2500.000000	2471.000000	2471.000000	2471.000000	2471.000000
mean	36.636260	-95.396896	36.676713	-95.513602	1330.982400	1335.202752	9.043707	15.946985	1360.797653
std	5.440588	16.306328	5.427860	16.341909	478.292825	489.419678	35.056018	8.601103	490.152049
min	17.701890	-157.922410	18.337310	-157.922410	10.000000	3.000000	-40.000000	3.000000	7.000000
25%	33.434170	-111.977770	33.434170	-111.977770	923.750000	925.000000	-4.500000	11.000000	939.000000
50%	36.475210	-92.548560	36.198370	-93.216920	1325.000000	1328.000000	-1.000000	14.000000	1342.000000
75%	40.788390	-83.348840	40.777240	-82.891880	1720.000000	1728.000000	7.000000	19.000000	1743.500000
max	61.174320	-64.798560	61.174320	-64.973360	2359.000000	2400.000000	540.000000	149.000000	2359.000000

8 rows × 23 columns

```
pd.DataFrame(flight_df.describe())
```

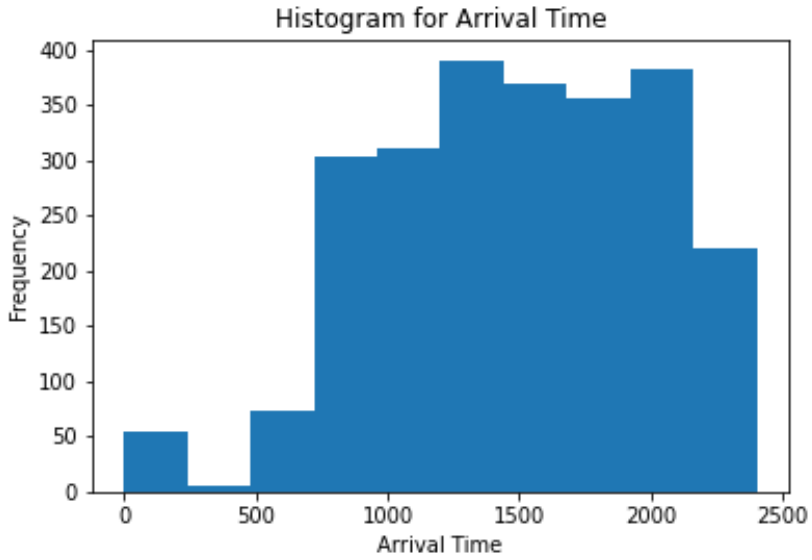
	TAXI_IN	SCHEDULED_ARRIVAL	ARRIVAL_TIME	ARRIVAL_DELAY	AIR_SYSTEM_DELAY	SECURITY_DELAY	AIRLINE_DELAY	LATE_AIRCRAFT_DELAY	WEATHER_DELAY
)	2469.000000	2500.000000	2469.000000	2463.000000	429.000000	429.000000	429.000000	429.000000	429.000000
'	7.384771	1503.012000	1485.708384	3.057653	10.757576	0.118881	17.347319	25.967366	2.589744
'	5.510285	499.455351	517.378421	36.523107	21.295692	1.739412	39.979704	47.806794	16.169213
)	2.000000	5.000000	1.000000	-55.000000	0.000000	0.000000	0.000000	0.000000	0.000000
)	4.000000	1115.000000	1108.000000	-13.000000	0.000000	0.000000	0.000000	0.000000	0.000000
)	6.000000	1525.500000	1524.000000	-5.000000	0.000000	0.000000	3.000000	9.000000	0.000000
)	8.000000	1915.000000	1916.000000	7.000000	16.000000	0.000000	20.000000	34.000000	0.000000
)	95.000000	2359.000000	2400.000000	527.000000	154.000000	26.000000	527.000000	464.000000	241.000000

From the above snapshot, the following properties are brought to light:

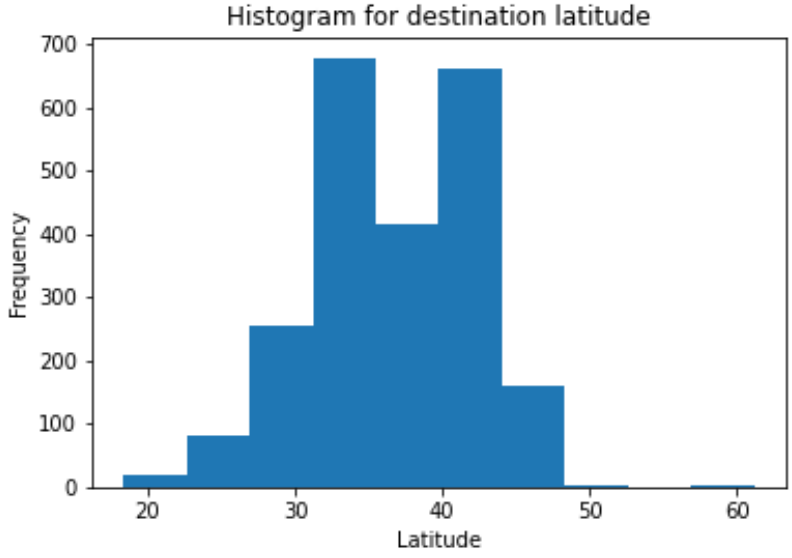
- Count: It shows the number of non-empty values. The dataset has 2500 rows and 38 attributes. From the snapshot, it is clear that many attributes have missing values since their count is less than 2500. For eg. Attribute O_LATITUDE has count 2276 which means there are 224 missing values.

- Mean: It indicates the average value of that particular attribute. Its values vary depending on the values each attribute has.

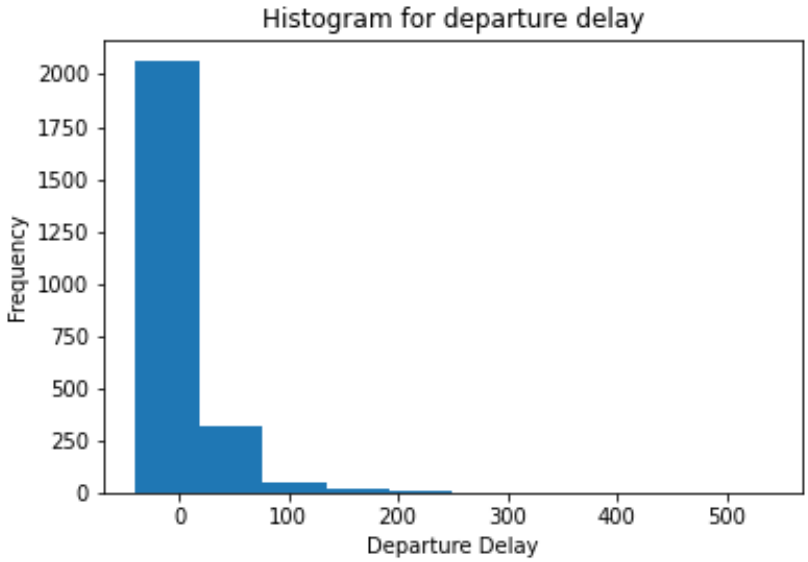
- Std: It means standard deviation that indicates how much is the maximum value close to the mean. If an attribute has high standard deviation, it means that its values vary greatly showing high variations. For eg. The histogram for the attribute ARRIVAL_TIME is shown below. A histogram can be used to efficiently indicate the variability of the data which is its standard deviation and also provides us information about the type of distribution.



From the histogram, we can see that it is a left skewed distribution where the long tail is situated in the left and majority of the values are clustered at the right. This also helps in detecting outliers. In an instant it can be spotted that there are unusual values between 0-250 since all other points are clustered towards the right.
Similarly plotting the distributions of a few more attributes:



The above figure shows the histogram for the D_LATITUDE attribute. This histogram clearly depicts the distribution of the data and it also very clearly shows the outliers at 60. Here the data does not have much variation since all the datapoints are clustered almost around the same space with very few outliers.



The above figure shows the distribution of the DEPARTURE_DELAY attribute. This diagram clearly depicts that the data has very little variation since the data points are clustered around the same values. The distribution in the histogram is not spread out as seen in the previous histogram distributions of other attributes. Thus, it will also have a very low standard deviation value.

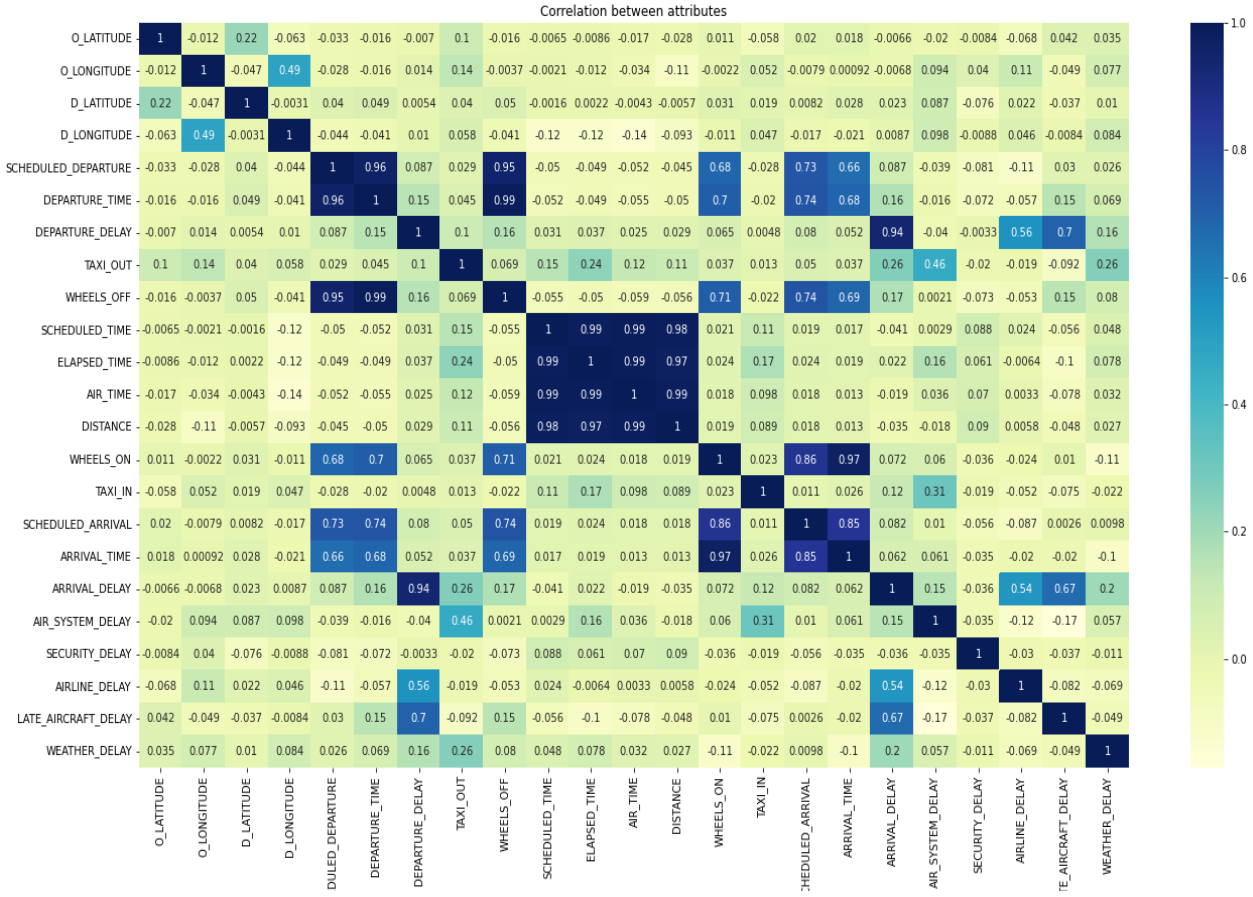
- Min and Max: These indicate the minimum and maximum values of each attribute respectively.

- 25 percentile: It indicated that 25% of the data for that particular attribute is less than a given value. For example for attribute O_LATITUDE, the 25 percentile is 33.434 which means that 25% of the data for the attribute O_LATITUDE is less than 33.434.

- 50 percentile: It shows that 50% of the data for that particular attribute is less than a given value as explained above.

- 75 percentile: Indicates that 75% of the data is less than a given value. For eg. For the attribute O_LATITUDE, 75% of the data (i.e. ¾ of the data) is less than 40.788.

3. Performing **Correlation Analysis** between various numerical attributes, the following heat map is generated. The dark blue grids indicate very high correlation between the attributes which leads to the problem of multicollinearity. For example The attributes ARRIVAL_TIME and WHEELS_ON have a very high correlation of 0.97. This means any change in any one of the two attributes will affect the other attribute. Thus, we need to maintain a set of independent variables where there is minimal correlation among the attributes. Thus, for the above pair of attributes, one of the attributes can be eliminated.



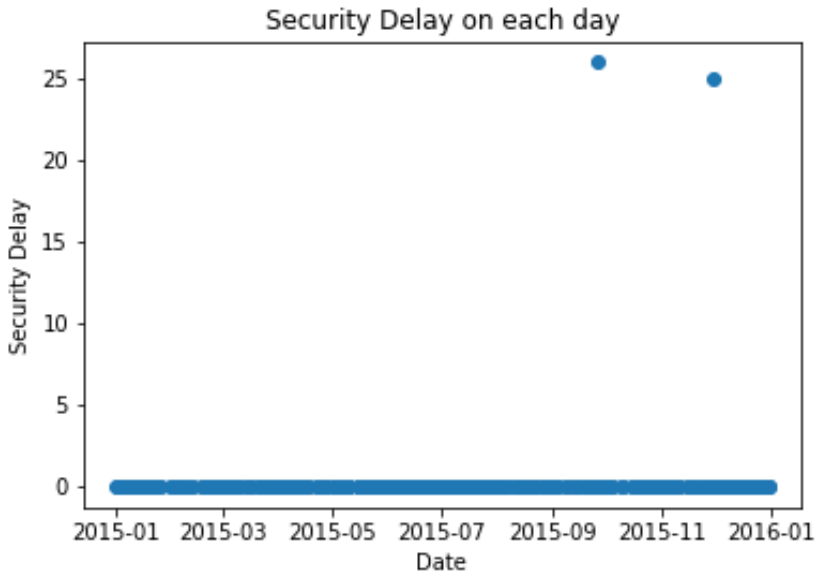
Similarly, other pair of attributes having very high correlation between them are:

- DEPARTURE_TIME and SCHEDULED DEPARTURE: 0.96
- WHEELS_OFF and DEPARTURE_TIME: 0.99
- WHEELS_OFF and SCHEDULED DEPARTURE: 0.95

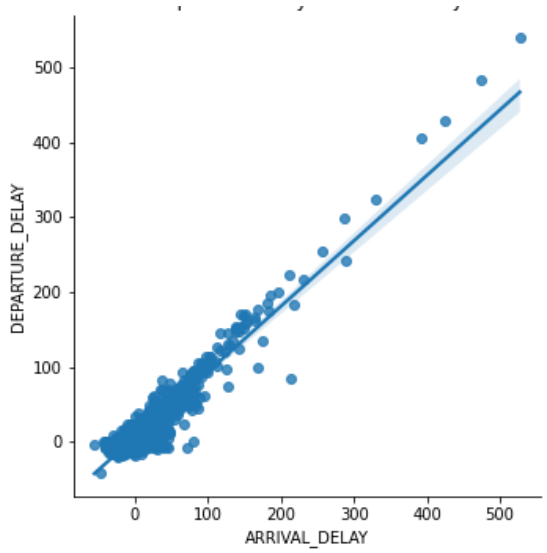
In the above attributes, since departure_time and scheduled_departure is highly correlated with each other, the attribute wheels_off if correlated with departure_time will also be highly correlated with scheduled_departure. Thus, it is necessary to eliminate any one of these attributes to avoid the problem of multicollinearity.

Similarly, there are other pairs of attributes that are highly correlated to each other like DEPARTURE_DELAY and ARRIVAL_DELAY and so on. Thus, it is advisable to eliminate one of these attributes from each pair to avoid the problem of multicollinearity.

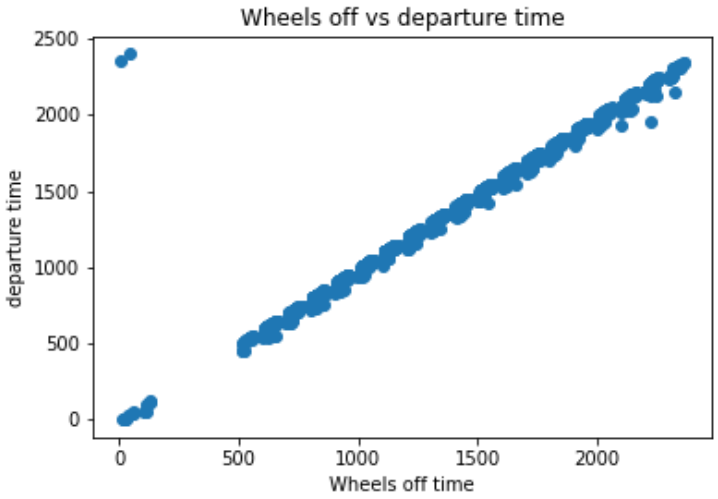
Next, we notice the attribute **SECURITY_DELAY** has 2071 missing values out of 2500. That means about 82.84% of this attribute has missing values. We also plot the SECURITY_DELAY for each day.



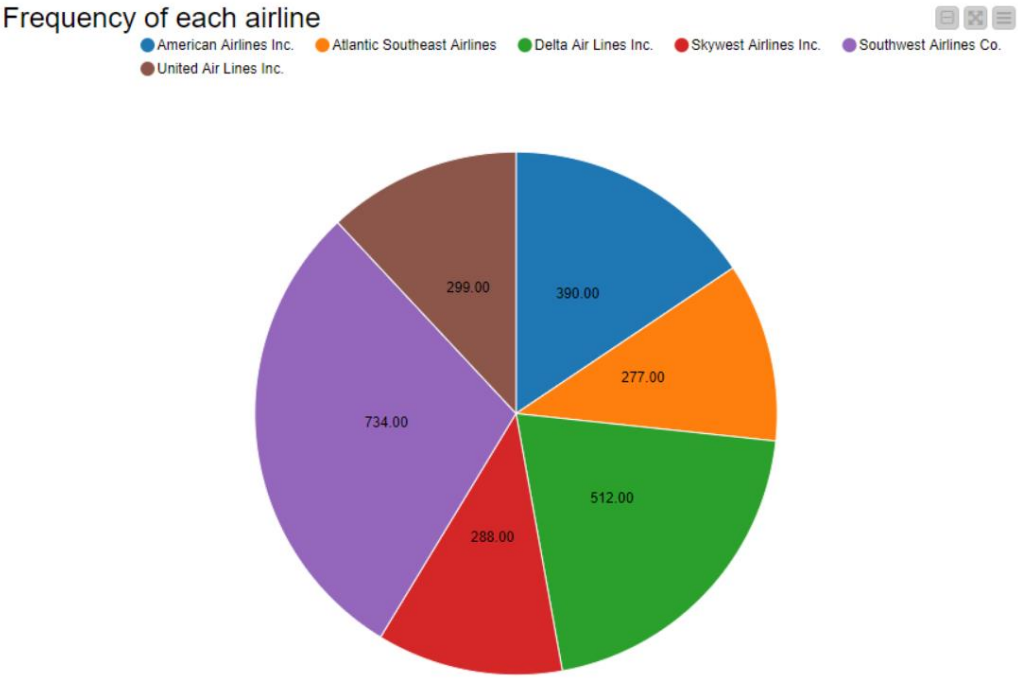
From the above graph, it is clearly evident that the non-null values for this attribute are mostly zero except for two points where the SECURITY_DELAY value is 24 and 25. These points can be considered as **outliers** since they vary a lot from the normal distribution of the data. Thus, we can drop this attribute since it has a large number of null values and the non-null values are mostly zero. This indicates that the delay caused by the security is very rare and mostly does not happen.



The above graph is an lmplot which shows two attributes: DEPARTURE_DELAY and ARRIVAL_DELAY. From the plot, it is evident that most of the points are such where ARRIVAL_DELAY is equivalent to DEPARTURE_DELAY. Thus, we can drop one of these attributes and also earlier during correlation analysis, these attributes had high correlation of 0.94 between them. Hence one of them can be dropped.



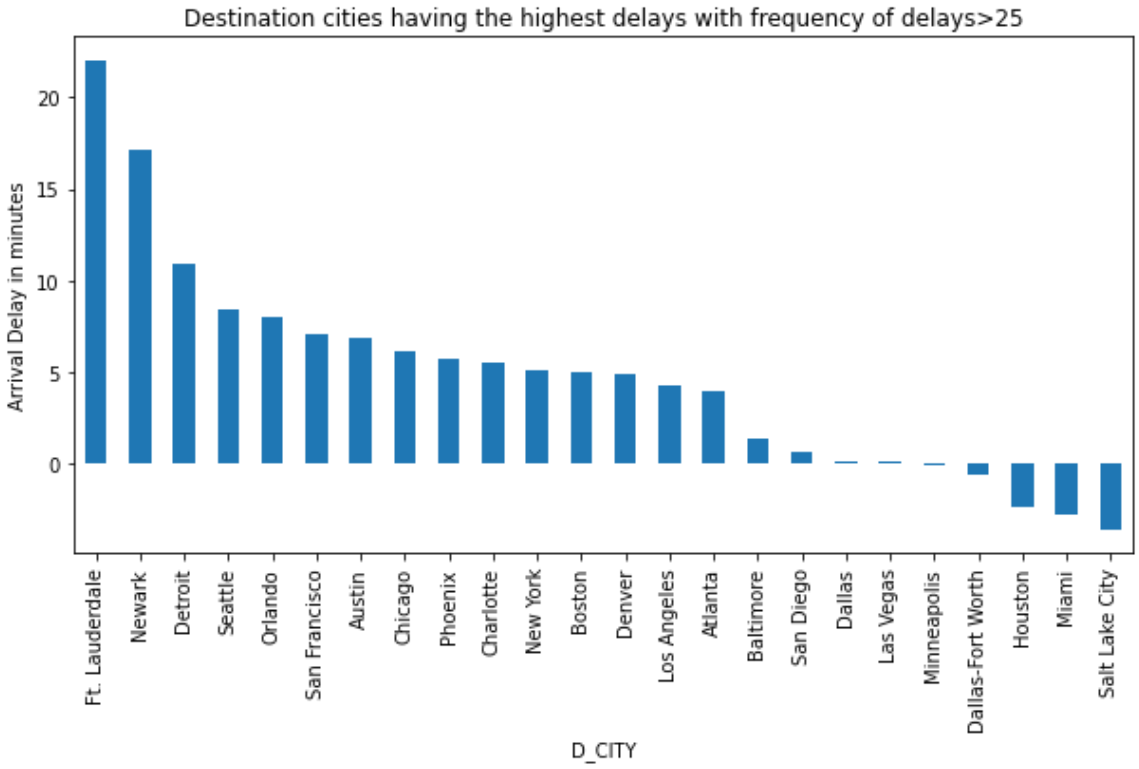
Similarly, the above conclusion can also be applied to the above graph where each value of WHEELS_OFF is almost similar to DEPARTURE_TIME except for a very few values that differ which can be considered as **outliers**. One of these attributes can be dropped since they also have a high correlation of 0.99 between them.



The above pie chart shows the **frequency of each airline**. Thus, from the above pie chart, we can write these airlines in the decreasing order of their frequency:

a) Southwest Airlines Co. : 734
b) Delta Airlines Inc : 512
c) American Airlines Inc: 399
d) United Air Lines Inc: 299
e) Skywest Airlines Inc: 288
f) Atlantic Southeast Airlines: 277

From this we understand that there are six different types of airlines and hence they can be converted into categorical values.

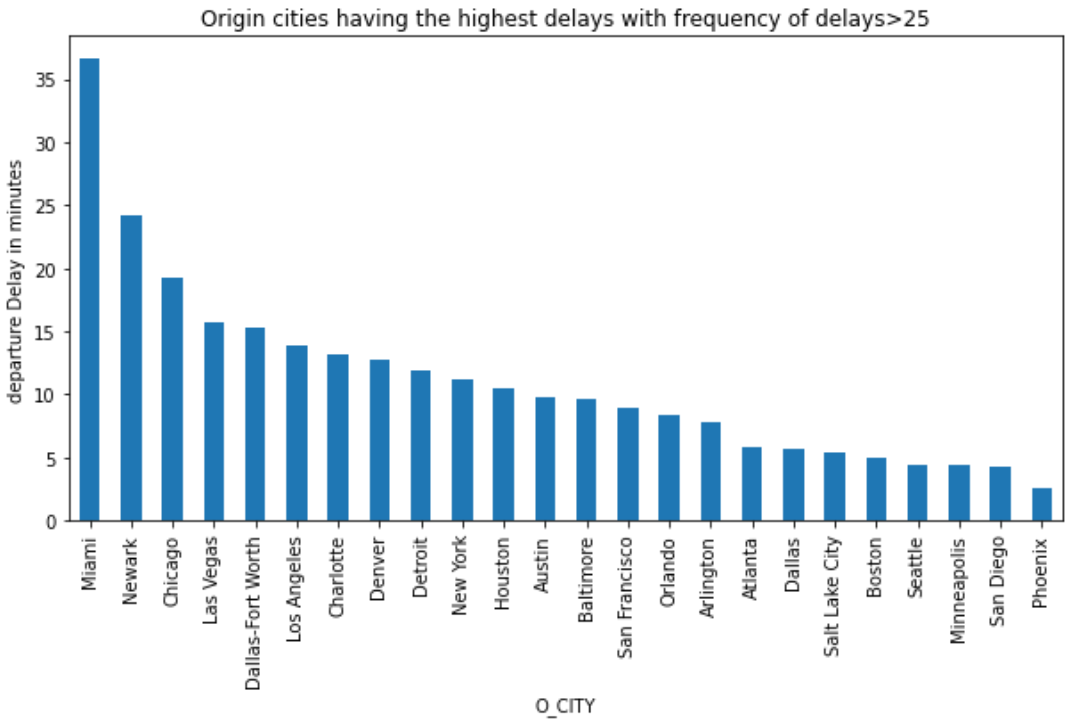


The above graph shows the destination **cities having the highest arrival delays** in minutes with frequency of delays greater than 25. For this, GROUP BY is performed on D_CITY and taking its ARRIVAL_DELAY attribute's mean along with its frequency. From this, filter out the cities having frequency of their delays greater than 25. The value 25 is chosen because it can be considered as a considerable frequency of delays that can be set as threshold as the frequency of these airlines vary from 1-178. After performing the above computations, the table for the first five values looks as shown below:

	ARRIVAL_DELAY	
	mean	size
D_CITY		
Ft. Lauderdale	22.035714	28
Newark	17.145455	58
Detroit	10.888889	36
Seattle	8.448276	29
Orlando	8.028571	36

From the above table, it is understood that Ft. Lauderdale destination city has the highest arrival delay(mean) with frequency of 28 delays followed by Newark with a mean slightly lower however having higher delay frequencies of 58. The fifth city in the table, Orlando has a very low arrival delay of 8.02 however it's a city where delays occur frequently(frequency=36).

A similar plot can be made for **origin cities with their departure delays**:



From the plot, we can see that Miami has the highest departure delay with frequently occurring delays greater than 25 followed by Newark which has relatively lower departure delay and so on. The computations mentioned for destination cities with arrival delays are applied over here too and the following table is generated (showing for the first five origin cities).

The table shows Miami having a departure delay(mean) of 36.594 which is highest along with a frequency of 38 delays followed by Newark which has a comparatively lesser delay of 24.26 however delays occur frequently here (size=48). Size indicates the frequency of delayed flights at that particular origin city.

	DEPARTURE_DELAY	
	mean	size
O_CITY		
Miami	36.594595	38
Newark	24.260870	48
Chicago	19.221477	150
Las Vegas	15.769231	52
Dallas-Fort Worth	15.306818	91

Now plotting the **total delay for each of the six airlines**:

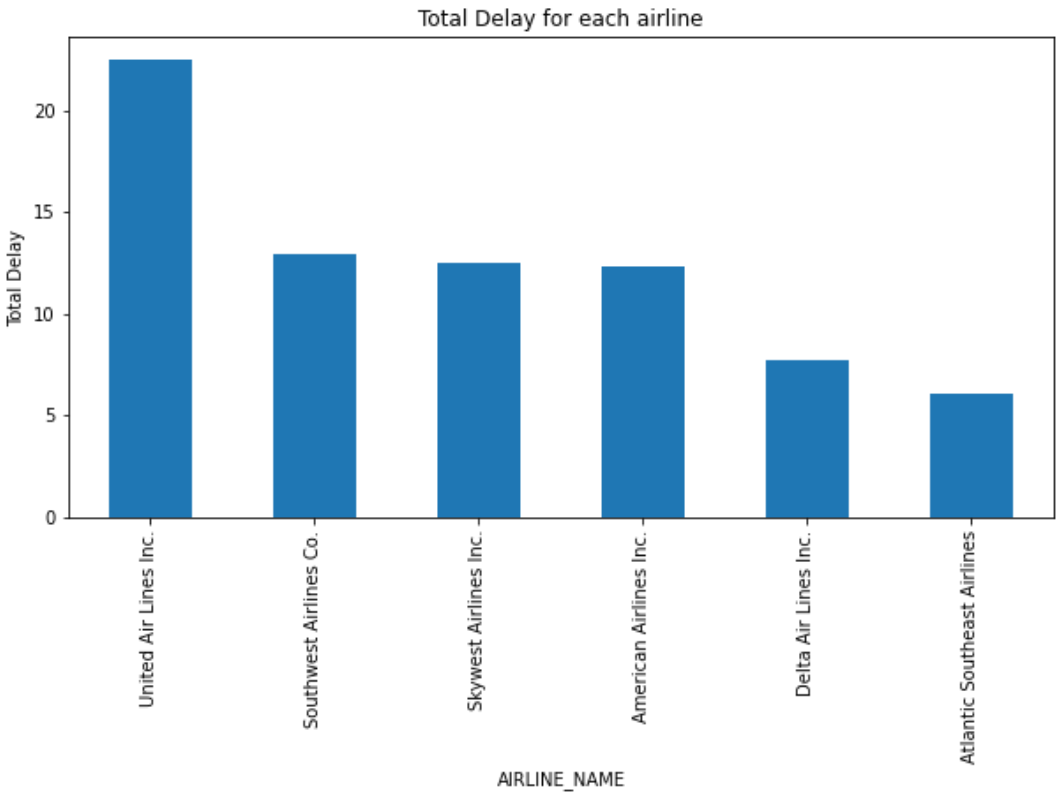
This can be done by grouping the data based on airline names along with the aggregate of their ARRIVAL_DELAY (mean and size) and DEPARTURE_DELAY (mean) and adding these two attributes to get the TOTAL DELAY which is represented in the table below:

	DEPARTURE_DELAY	ARRIVAL_DELAY		total_mean_delay
	mean	mean	size	
AIRLINE_NAME				
United Air Lines Inc.	15.637584	6.861953	299	22.499537
Southwest Airlines Co.	9.928473	2.977901	734	12.906374
Skywest Airlines Inc.	7.130282	5.354610	288	12.484892
American Airlines Inc.	9.080729	3.281984	390	12.362714
Delta Air Lines Inc.	7.824803	-0.151874	512	7.672929
Atlantic Southeast Airlines	3.637037	2.396296	277	6.033333

This can also be plotted in a graph as shown below.

Thus, from the bar plot, it is evident that United Air Lines Inc has the highest delay with a frequency of 299 arrival delays (total number of flights for this airline is 299) which suggests that all the flights of this airline are delayed. This is followed by

Southwest Airlines Co. having the next highest delay with a frequency of 734 arrival delays (total number of flights for Southwest Airlines Co. is 734) indicating all flights are delayed and thus it comes lower to United Air Lines and so on with the minimum number of delays for Atlantic Southeast Airlines.



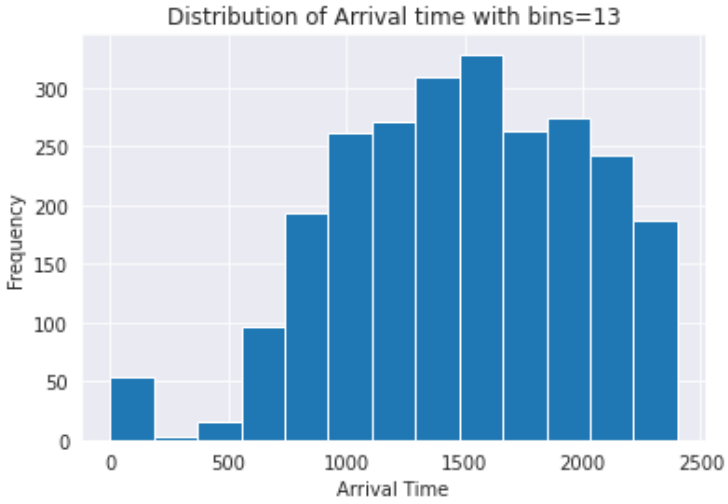
1B.DATA PREPROCESSING

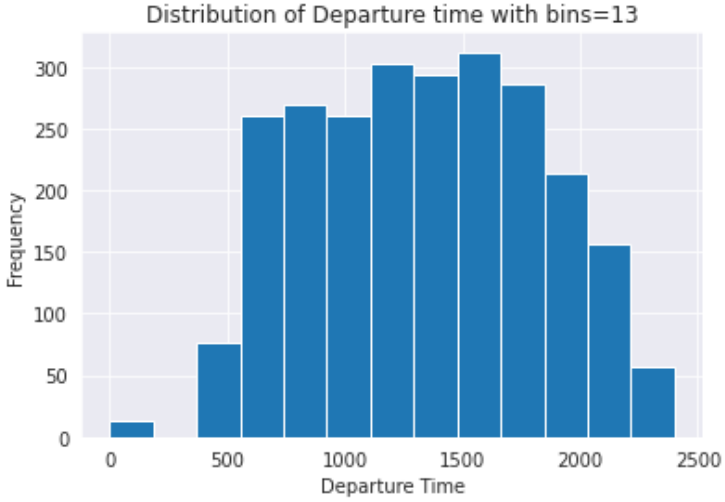
1. Binning on ARRIVAL_TIME and DEPARTURE_TIME:
   In order to apply binning, first the number of bins into which the data should be categorized should be decided.
   Since the dataset consists of 2500 points, we cannot choose very small number of bins since it will not represent the data properly. In order to decide the number of bins, the Sturge's rule can be used which takes into account the size of the data to decide the number of bins. Th formula is shown below:

$$bins = 1 + ceil(log_2(n))$$

   Where n is the sample size (2500 in this case) and thus we get bins=13.

   The data is distributed well for bins=13 and increasing the number of bins does not change the shape of the distribution. The distribution for bins=13 is shown as:





From the above figures, we can see that the data has been distributed nicely and each bin has enough number of values in it.

i. Equi-width Binning: Here, the continuous variable is divided into several categories of the same width.

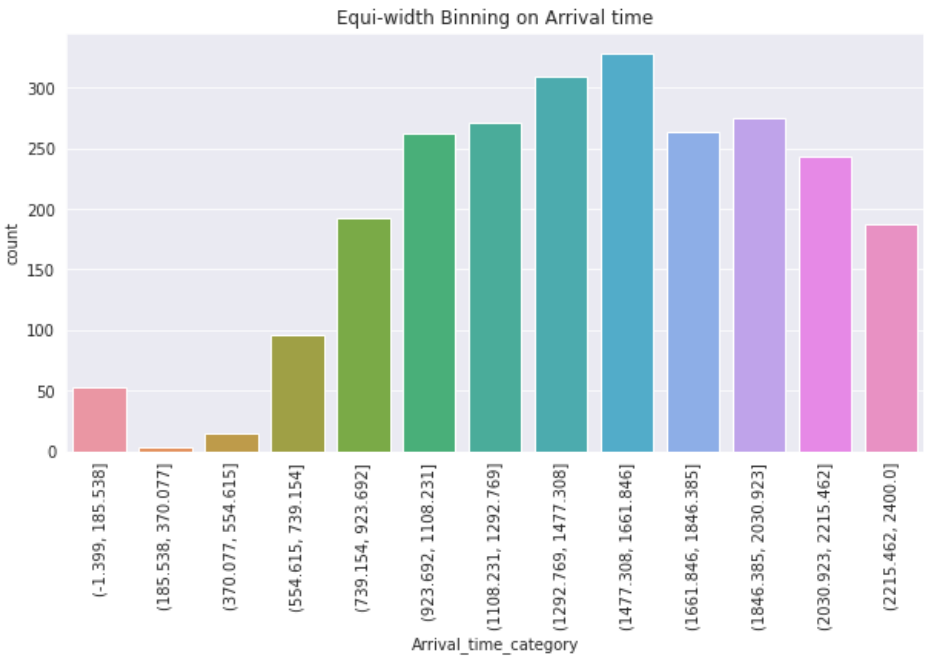
$$w = \left\lfloor \frac{max - min}{x} \right\rfloor$$

Where, x=number of categories(bins), in this case, x=13 and w=width of a category.

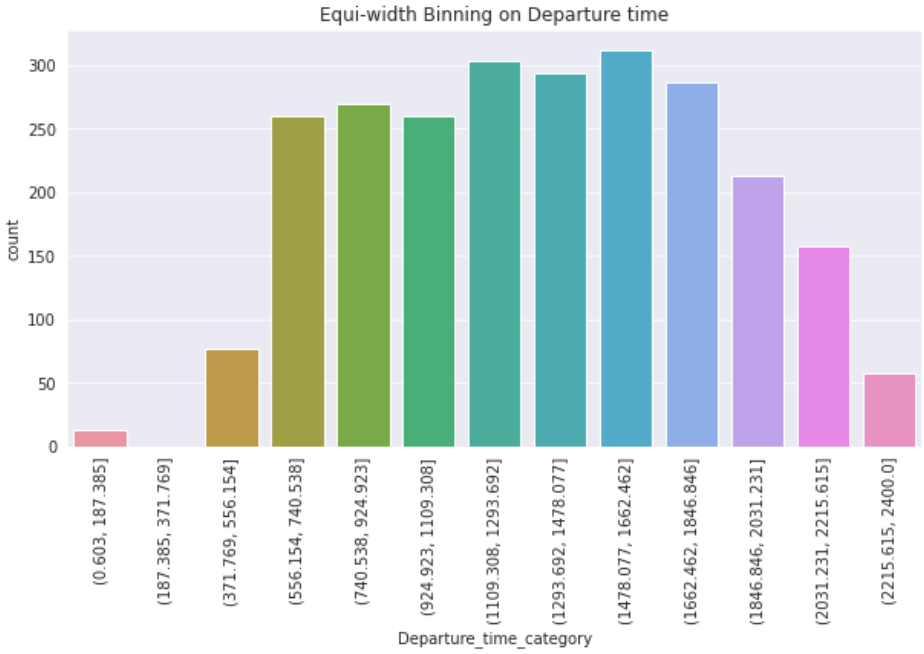
This can be automatically done using the cut function of pandas to convert the numerical values into 13 categorical values. We need to specify the number of bins (13) which is shown in the code below for ARRIVAL_TIME and DEPARTURE_TIME:

```
flight_df['Arrival_time_category']=pd.cut(flight_df['ARRIVAL_TIME'],bins=13)

flight_df['Departure_time_category']=pd.cut(flight_df['DEPARTURE_TIME'],bins=13)
```

Thus the data is divided into 13 bins of equal sizes. This is shown in the graph below for the attribute ARRIVAL_TIME:



The graph for DEPARTURE_TIME is also similar and is shown below:



The x-axis for each of these graphs represents the 13 bins into which the continuous values are categorized into.

ii. Equi-depth Binning: Here, it calculates the size of each bin so that each bin consists of almost the same number of observations, but the bin range will vary. The values of data are distributed equally into formed categories.

$$freq = \frac{n}{x}$$

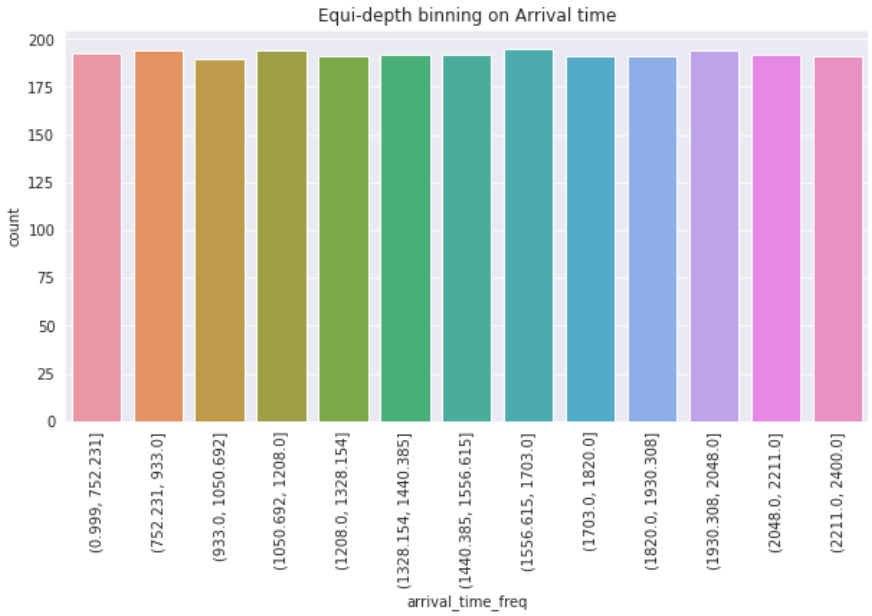
Here, x represents number of categories or bins, freq represents the frequency of a category and n represents the number of values in the data.

The qcut function of pandas defines the bins using percentiles based on the distribution of the data, not the actual numeric edge of bins.

The code below is used to bin the data using qcut:

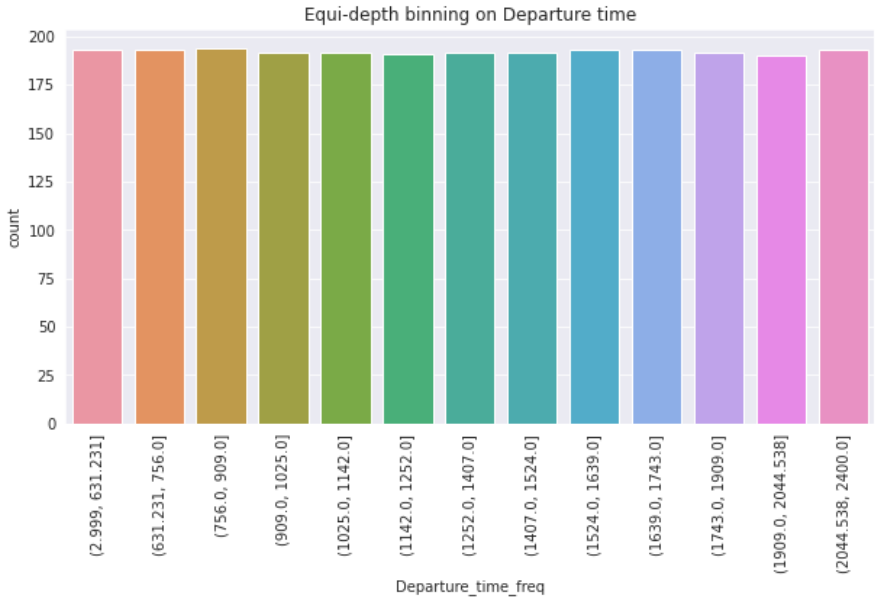
```
flight_df['arrival_time_freq']=pd.qcut(flight_df['ARRIVAL_TIME'],q=13)

flight_df['Departure_time_freq']=pd.qcut(flight_df['DEPARTURE_TIME'],q=13)
```

The graph for the equi-depth binning on ARRIVAL_TIME is shown below:



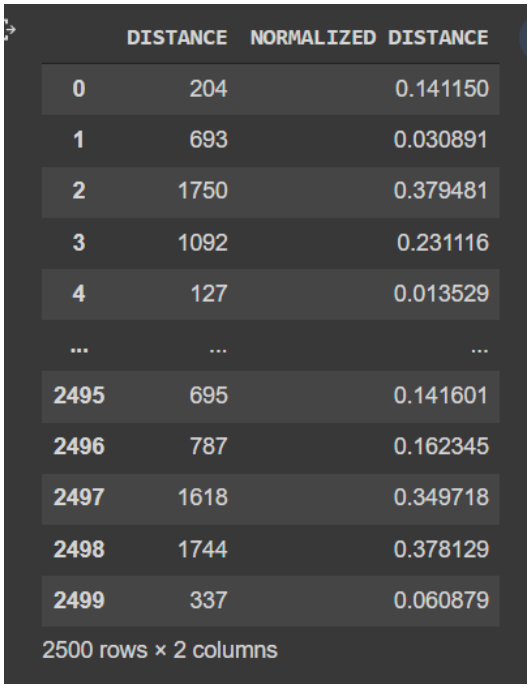
The above graph shows the data being divided into 13 bins however the bins do not have the same width but all the bins have almost the same number of values in them.

Similarly, the equi-depth binning on DEPARTURE_TIME can also be shown in the graph below:



The x-axis of these two graphs indicates the intervals (bins) into which the data is divided into and the y-axis indicates the frequency of each bin. Since it is an equi-depth binning, all the intervals have almost the same frequency.

2. Normalization is done using two methods as specified in the question:
    i. Min-Max Normalization: A snapshot of the result after normalizing the DISTANCE attribute is given below:



	DISTANCE	NORMALIZED DISTANCE
0	204	0.141150
1	693	0.030891
2	1750	0.379481
3	1092	0.231116
4	127	0.013529
...	...	...
2495	695	0.141601
2496	787	0.162345
2497	1618	0.349718
2498	1744	0.378129
2499	337	0.060879

2500 rows × 2 columns

This method rescales the values to a fixed range of [0.0,1.0] by subtracting the minimum value of the feature and then dividing it by the range.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

MinMaxScalar available in the Scikit-learn library is used for this purpose. First a scalar object is created. Then, we fit the scalar parameters, meaning we calculate the minimum and maximum value of the feature. Finally, we transform the data using those parameters.

The code for the above method is shown as below:

```
[97] from sklearn.preprocessing import MinMaxScaler
     scalar=MinMaxScaler()
     scalar.fit_transform(np.array(flight_df['DISTANCE']).reshape(-1,1))
```

The reshape method is used to reshape it to the required format else it shows error and normalization does not take place. Thus, we convert it into the acceptable format.

ii. Z-score Normalization: It transforms the data into a distribution with a mean of 0 and a standard deviation of 1.
A snapshot of the result is given below:

	DISTANCE	Z-SCORE DISTANCE
0	204	-0.217464
1	693	-1.048564
2	1750	1.579003
3	1092	0.460673
4	127	-1.179432
...	...	...
2495	695	-0.214065
2496	787	-0.057702
2497	1618	1.354657
2498	1744	1.568806
2499	337	-0.822518

2500 rows × 2 columns

Each standardized value is computed by subtracting the mean(μ) of the corresponding feature and then dividing by the standard deviation(σ).

$$x_{std} = \frac{x-\mu}{\sigma}$$

Z-score does not rescale to a fixed range. Here the StandardScaler class from Scikit-learn library to perform the z-score. First, we create a standard_scalar object. Then, we calculate the parameters of the transformation (mean and standard deviation) using .fit() method. Next, we call the .transform() method to apply standardization to the data frame. The .transform() method uses the parameters generated from the .fit() method to perform the z-score. To simplify the code, .fit_transform() can be directly used.

The code for the above method is shown below:

```
[103] from sklearn.preprocessing import StandardScaler
      standard_scalar=StandardScaler()
      standard_scalar.fit_transform(np.array(flight_df['DISTANCE']).reshape(-1,1))
```

The mean and median of the transformed data is shown below:

	DISTANCE
count	2.500000e+03
mean	4.964917e-17
std	1.000200e+00
min	-1.281408e+00
25%	-7.307403e-01
50%	-2.735504e-01
75%	3.790923e-01
max	6.256276e+00

From the above table it is clear that after applying z-score normalization, the mean of the data is reduced to nearly zero and standard deviation becomes one,

3. To discretize the ARRIVAL_DELAY attribute, binning technique is used. According to the question, four bins can be selected which can be mapped to labels: 'Early Arrival', 'Negligible Arrival Delay', 'Minor Arrival Delay', 'Major Arrival Delay'.

   The bins chosen are: [-1000,-15,15,60,1000]

   -1000 and 1000 are just random high values which are chosen just to give the bins a limit since any delay below -15 is grouped to 'Early Arrival' and any delay above 60 is considered as a 'Major Arrival Delay'. Delays within 15 minutes (early or delayed) is considered as a 'Negligible Arrival Delay' whereas between 15 and 60 minutes is considered as 'Minor Arrival Delay'.

   Here, the cut function of pandas is used to convert the numerical values of the ARRIVAL_DELAY attribute to categorical values. The interval range is calculated as the difference between the minimum and maximum value and then this interval is split into four parts one for each group. We need to specify the bins and the labels as shown below:

```
labels=['Early Arrival','Negligible Arrival Delay','Minor Arrival Delay','Major Arrival Delay']
bins=[-1000,-15,15,60,1000]
flight_df['ARRIVAL_STATUS']=pd.cut(flight_df['ARRIVAL_DELAY'],
                                   bins=bins, labels=labels,
                                   include_lowest=True)
```

   The result of applying these operations is:

	ARRIVAL_DELAY	ARRIVAL_STATUS
1	14.0	Negligible Arrival Delay
0	-0.5	Negligible Arrival Delay
2	-15.0	Early Arrival
3	1.0	Negligible Arrival Delay
4	-9.0	Negligible Arrival Delay
...	...	...
2495	0.0	Negligible Arrival Delay
2496	-12.0	Negligible Arrival Delay
2497	-20.0	Early Arrival
2498	-41.0	Early Arrival
2499	-3.0	Negligible Arrival Delay

2500 rows × 2 columns

From the above result, we can see that the attribute has been discretized according to the specified requirements in the question.

The frequency of each class from ARRIVAL_STATUS (new column) is shown below:

```
for i in labels:
  print(i,len(flight_df[flight_df['ARRIVAL_STATUS']==i]))
```

```
Early Arrival 559
Negligible Arrival Delay 1507
Minor Arrival Delay 314
Major Arrival Delay 120
```

Thus, from the above snapshot, we can see that 1507 i.e. most of the flights have 'Negligible Arrival Delay' whereas 559 flights have 'Early Arrival' which means they arrived 15 or more minutes early. Very small number of flights (120) had a 'Major Arrival Delay' with a delay of more than an hour whereas the remaining 314 flights had a 'Minor Arrival Delay'.

4. In order to binarize the AIRLINE_NAMES attribute with values 0 or 1 the get_dummies() function of pandas can be used which converts the categorical columns into indicator columns (columns of 0s and 1s).

For each distinct value in the AIRLINE_NAMES column, a new column with an indicator of 0 or 1 is created as shown below.

In the table given below, it is seen that there are six attributes created each corresponding to an airline. Thus the airline which is present is marked as 1 whereas the rest of the columns representing the other airlines are marked as zero.
This can be written as:

pd.get_dummies(flight_df['AIRLINE_NAME'])

	AIRLINE_NAME	American Airlines Inc._binary	Atlantic Southeast Airlines_binary	Delta Air Lines Inc._binary	Skywest Airlines Inc._binary	Southwest Airlines Co._binary	United Air Lines Inc._binary
1	Atlantic Southeast Airlines	0	1	0	0	0	0
0	American Airlines Inc.	1	0	0	0	0	0
2	Southwest Airlines Co.	0	0	0	0	1	0
3	Atlantic Southeast Airlines	0	1	0	0	0	0
4	Atlantic Southeast Airlines	0	1	0	0	0	0
...	...	...	...	...	...	...	...
2495	Skywest Airlines Inc.	0	0	0	1	0	0
2496	United Air Lines Inc.	0	0	0	0	0	1
2497	Southwest Airlines Co.	0	0	0	0	1	0
2498	United Air Lines Inc.	0	0	0	0	0	1

1C. SUMMARY

By examining the attributes, it is found that many of them are categorical variables. Further exploratory data analysis on these attributes might reveal more useful information. The dataset also has many variables which are empty which can be filled. There are certain attributes that do not contribute much information about the dataset like SECURITY_DELAY which can be dropped since most of the times it is zero and majority of the values are empty. On examining the correlation between the attributes, it is also found out that a large number of attributes are highly correlated or dependent on each other which may lead to problems of multicollinearity. Thus, such attributes must be examined and dropped accordingly.

There are a few attributes which are very similar to each other having almost equal values like WHEELS_OFF and DEPARTURE_TIME which is evidently depicted in the graph in the pages above. There AIRLINE_NAMES are categorical attributes which are binarized and instead of writing all the six columns corresponding to each airline name, one of them can be dropped. It is also observed that majority of the flights are from the Southwest Airlines Co. and still has lower number of delays compared to United Air Lines Inc. which even though has small number of flights, has much higher delays. It is also found that Ft. Lauderdale destination city has the highest arrival delay whereas Miami origin city has the highest departure delay. Further exploration on these can help to give more accurate details.