New York Flights data analysis & Visualization

SRAVANI RAVULAPARTHI & JASPREET SINGH BHATIA

The New York flights dataset is a collection of data pertaining to different airlines flying from 3 different airports in NYC to IAD, BWI and, also capturing flights, planes, and weather specific details during the year of 2013.

In the data we see there are 2 variables that relate to the delay that we need to consider for finding the worst day to fly if we hate delays:

1. arr_delay: This is the arrival delay of the flight for that trip
2. dep_delay: This is the departure delay of the flight for that trip.

In both the above variables, the positive values are delayed flights while negative values are flights that arrived or departed early. The flights dataset also contains information about the carrier, talinum, air_time, distance, year, month, day, hour, and minute.

Now, let's perform some EDA of the data. For that we need to import libraries first, so importing all the necessary libraries:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.pyplot import bar
import seaborn as sns
import statsmodels.formula.api as smf
import statsmodels.api as sm
from scipy import stats
import math
from statsmodels.formula.api import ols
from statsmodels.stats.anova import AnovaRM
from statistics import mean
from scipy.stats import ttest_ind

# sklearn stuff
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_squared_error
```

Reading csv files from GitHub using pandas read_csv:

```
flights = pd.read_csv('https://raw.githubusercontent.com/SravaniRVS/DATA601/main/Project%202/flights.csv', parse_dates= {"Date" : ["year","month","day"]},ke
flights.drop(columns=['Unnamed: 0'], inplace=True)
flights = flights[(flights['dest'] == 'BWI') | (flights['dest']== 'DCA') | (flights['dest'] == 'IAD')]
planes = pd.read_csv('https://raw.githubusercontent.com/SravaniRVS/DATA601/main/Project%202/planes.csv')
airlines = pd.read_csv('https://raw.githubusercontent.com/SravaniRVS/DATA601/main/Project%202/airlines.csv')
airports = pd.read_csv('https://raw.githubusercontent.com/SravaniRVS/DATA601/main/Project%202/airports.csv')
md_daily = pd.read_excel('https://raw.githubusercontent.com/SravaniRVS/DATA601/main/Project%202/weatherMDdaily.xlsx')
ny_daily = pd.read_excel('https://raw.githubusercontent.com/SravaniRVS/DATA601/main/Project%202/weatherNYdaily.xlsx')
ny_hourly = pd.read_csv('https://raw.githubusercontent.com/SravaniRVS/DATA601/main/Project%202/weatherNYhourly.csv', parse_dates= {"Date" : ["year","month",
```

Let's check the shape, info, and description of the flights dataset.

```
print("Shape of the dataset is:", flights.shape, "\n\n")
print("Information about the dataset : \n\n")
flights.info()
print("\n\n Description of the dataset : \n")
flights.describe()
```

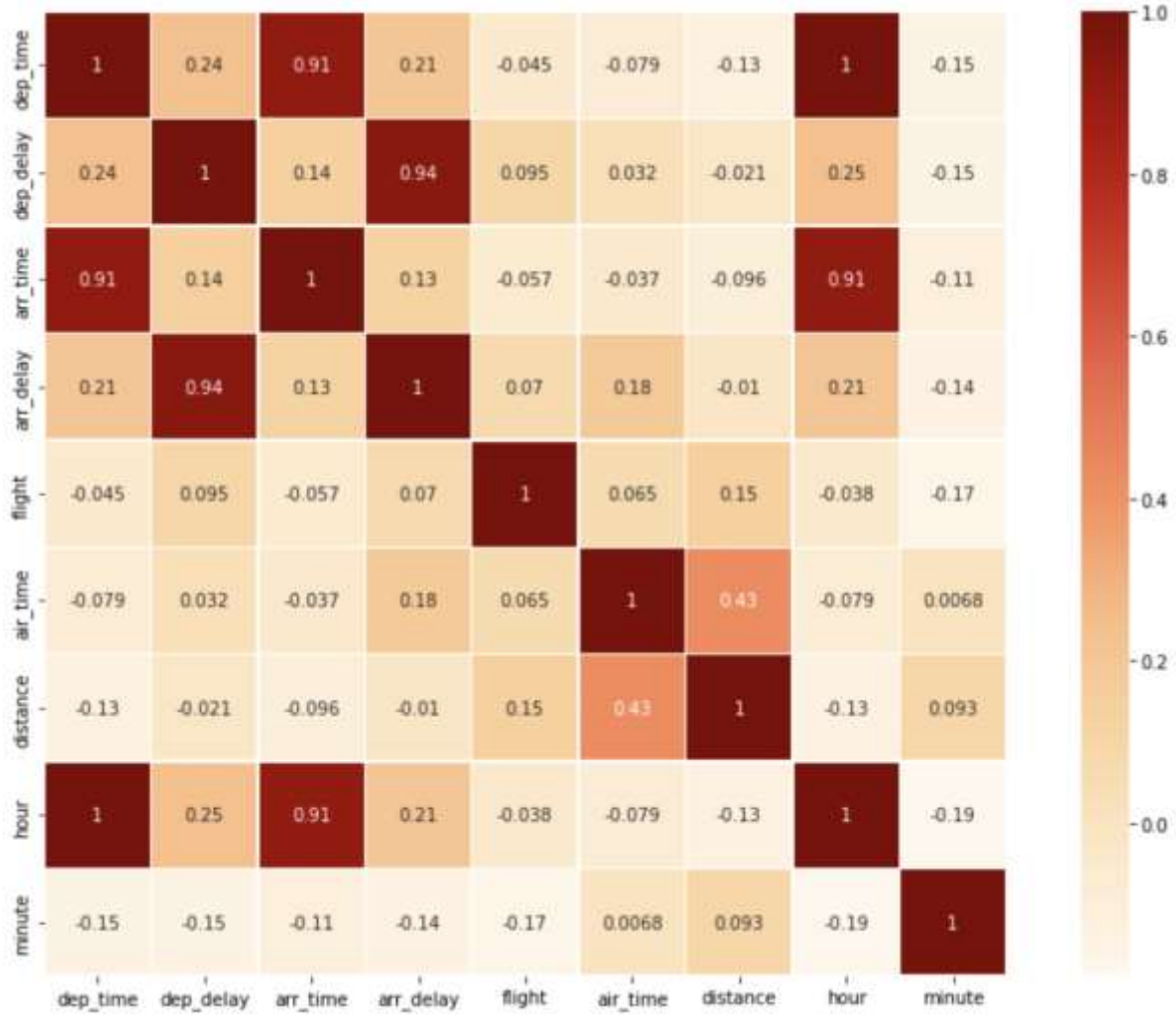
```
Shape of the dataset is: (17186, 17)


Information about the dataset :


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17186 entries, 0 to 17185
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       17186 non-null  datetime64[ns]
 1   year       17186 non-null  object
 2   month      17186 non-null  object
 3   day        17186 non-null  object
 4   dep_time   16244 non-null  float64
 5   dep_delay  16244 non-null  float64
 6   arr_time   16197 non-null  float64
 7   arr_delay  16181 non-null  float64
 8   carrier    17186 non-null  object
 9   tailnum    16811 non-null  object
 10  flight     17186 non-null  int64
 11  origin     17186 non-null  object
 12  dest       17186 non-null  object
 13  air_time   16181 non-null  float64
 14  distance   17186 non-null  int64
 15  hour       16244 non-null  float64
 16  minute     16244 non-null  float64
dtypes: datetime64[ns](1), float64(7), int64(2), object(7)
memory usage: 2.2+ MB
```

Description of the dataset :

	dep_time	dep_delay	arr_time	arr_delay	flight	air_time	distance	hour	minute
count	16244.000000	16244.000000	16197.000000	16181.000000	17186.000000	16181.000000	17186.000000	16244.000000	16244.000000
mean	1384.065747	13.150517	1497.865839	10.835919	3667.399162	45.745319	212.323170	13.490396	35.026102
std	491.080973	41.683282	501.327183	45.396630	1390.788852	6.552948	14.248791	4.943026	19.032372
min	2.000000	-32.000000	1.000000	-62.000000	63.000000	31.000000	169.000000	0.000000	0.000000
25%	956.000000	-6.000000	1109.000000	-13.000000	2187.000000	41.000000	212.000000	9.000000	19.000000
50%	1435.000000	-3.000000	1543.000000	-3.000000	3761.000000	45.000000	214.000000	14.000000	38.000000
75%	1819.000000	11.000000	1922.000000	16.000000	4418.000000	49.000000	228.000000	18.000000	53.000000
max	2400.000000	853.000000	2400.000000	851.000000	6181.000000	131.000000	229.000000	24.000000	59.000000

Finding the correlation matrix of the data:

```
plt.figure(figsize = (15,10))
df_corr = flights.drop(['year',], axis=1)
sns.heatmap(df_corr.corr(), annot=True,square=True,cmap='OrRd', linewidth=0.5, annot_kws = {'size':10}, fmt='.2g' )
```



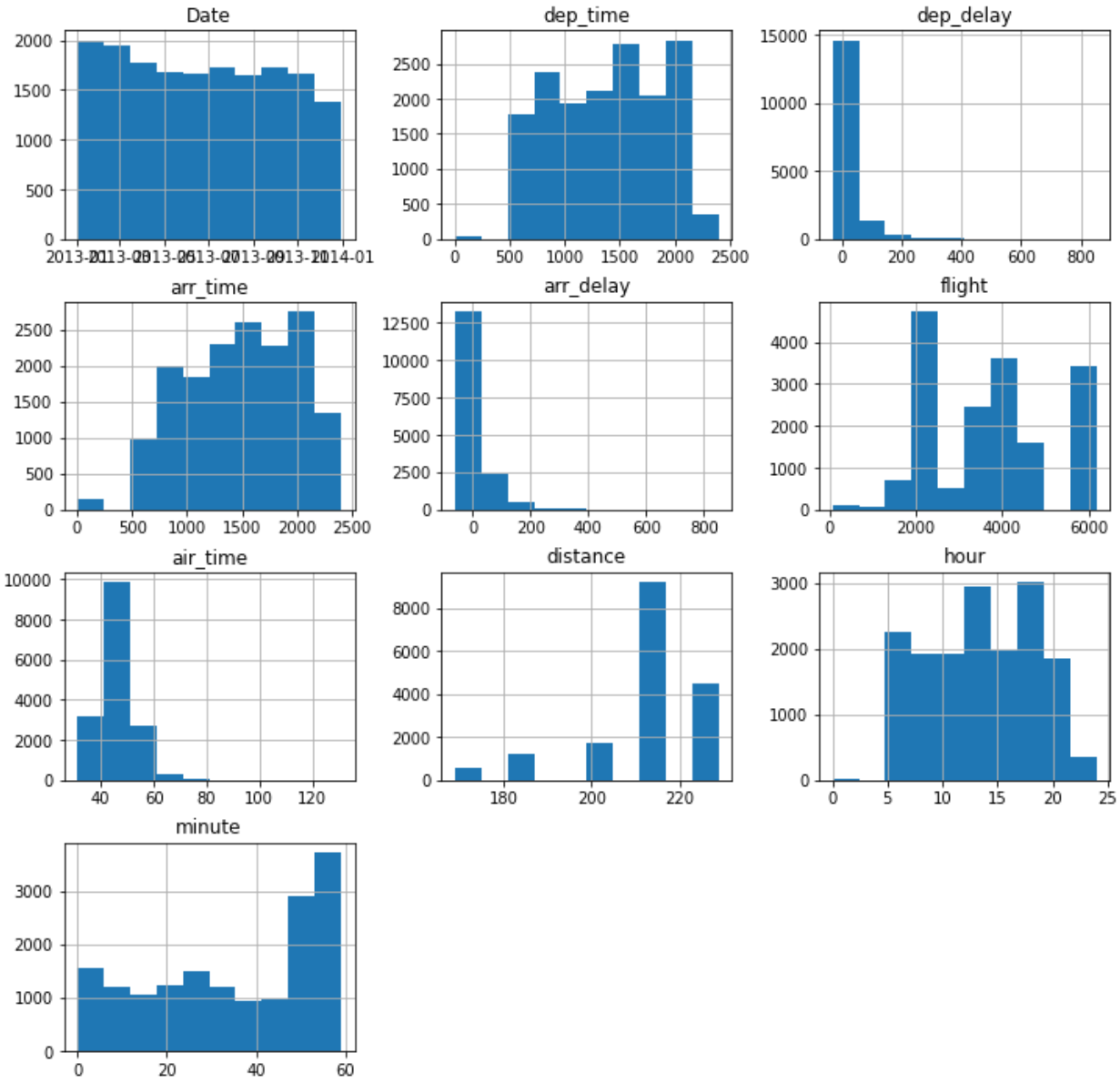
Checking for null values:

```
flights.isnull().sum()
```

```
Date            0
year            0
month           0
day             0
dep_time      942
dep_delay     942
arr_time      989
arr_delay    1005
carrier         0
tailnum       375
flight          0
origin          0
dest            0
air_time     1005
distance        0
hour          942
minute        942
dtype: int64
```

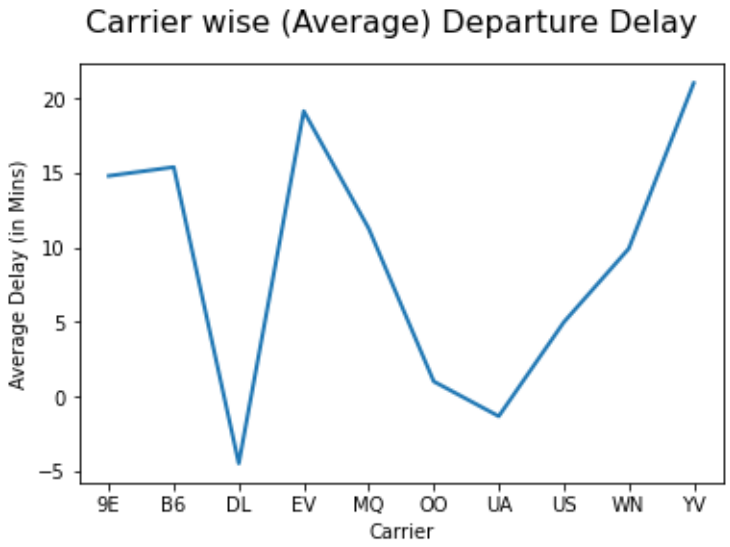
We can see here that dep_delay is strongly correlated with arr_delay, which means higher dep_delay leads to higher arr_delay.

Histogram of the data:



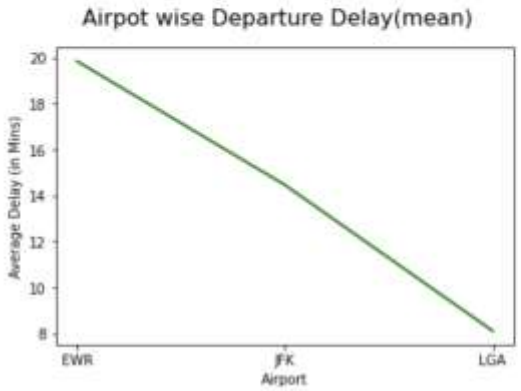
Checking the departure delay of flights carrier wise.

```
#departure  delay carrier wise
b=flights[['dep_delay','carrier']].groupby('carrier').mean()

plt.plot(b, linewidth=2.0)
plt.xlabel('Carrier')
plt.ylabel('Average Delay (in Mins)')
plt.suptitle('Carrier wise (Average) Departure Delay',fontsize=16)

plt.show()
```



YV has the worst dep_delay among the carriers, and DL has the best dep_delay. Now checking the dep_delay airports wise:

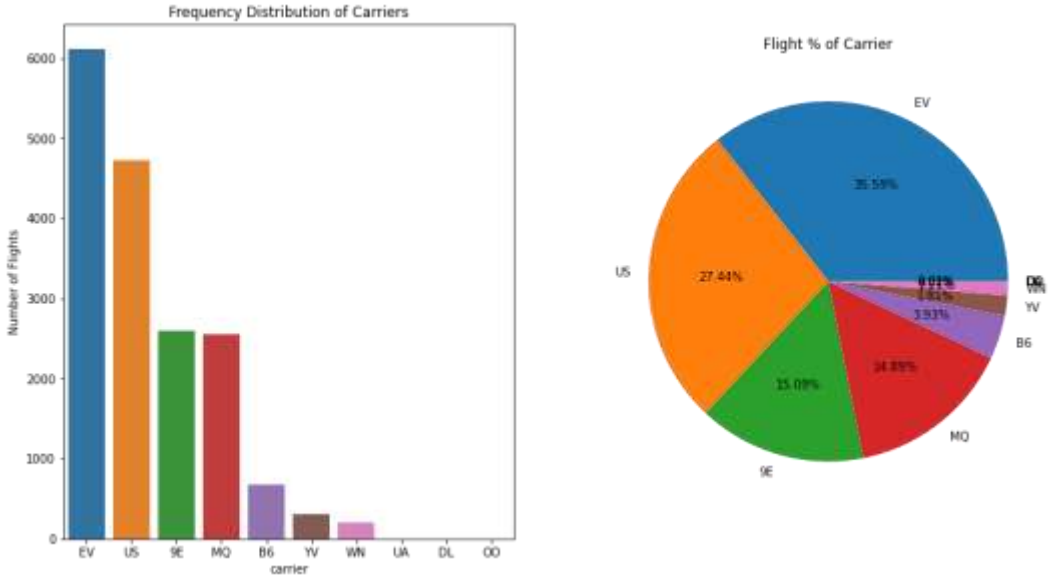
```
#departure delay airport wise

c=flights[['dep_delay','origin']].groupby('origin').mean()

plt.plot(c, linewidth=2.0,color='green')
plt.xlabel('Airport')
plt.ylabel('Average Delay (in Mins)')
plt.suptitle('Airpot wise Departure Delay(mean)',fontsize=16)

plt.show()
```



LGA airport in NY has the least dep_delay, and EWR has the highest dep_delay.

Graphical representation of carriers scheduled flights in numbers and %:

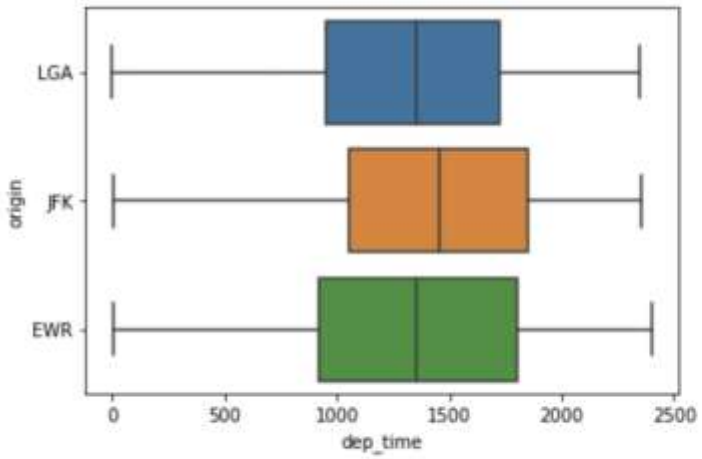
```
f,ax=plt.subplots(1,2,figsize=(15,8))
flights['carrier'].value_counts().plot.pie(autopct='%1.2f%%',ax=ax[1],shadow=False)
ax[1].set_title('Flight % of Carrier')
ax[1].set_ylabel('')
sns.countplot('carrier',order = flights['carrier'].value_counts().index, data=flights,ax=ax[0])
ax[0].set_title('Frequency Distribution of Carriers')
ax[0].set_ylabel('Number of Flights')
plt.show()
```



EV has the highest percentage of all flights, & UA, DL, and OO has negligible percent of all flights.

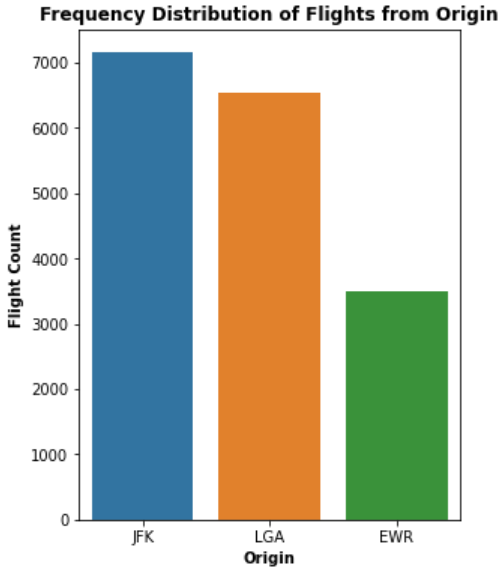
Visualizing dep_time and origin:

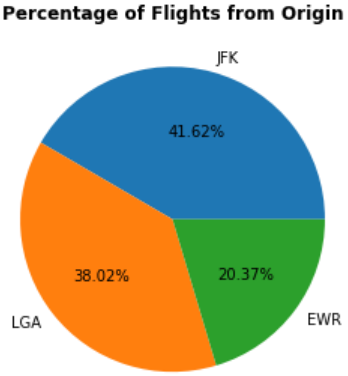
```
sns.boxplot('dep_time','origin',data = flights)

plt.show()
```



The highest IQ range (900 to 1800) is for JFK, where 75% flights are falling under the departure time of 1800 and middle data point for departure is near 1500

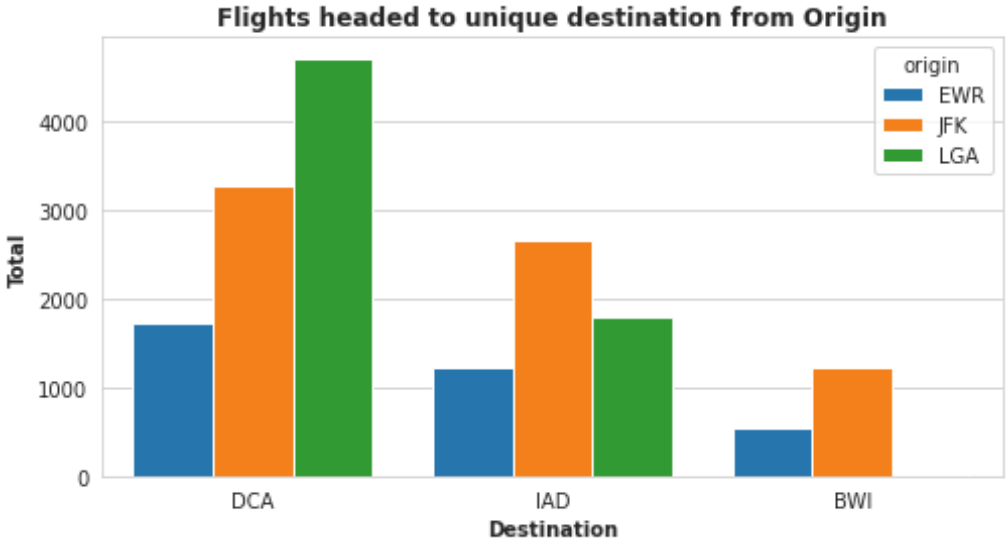
Plotting the number of flights from origin and their percentages:





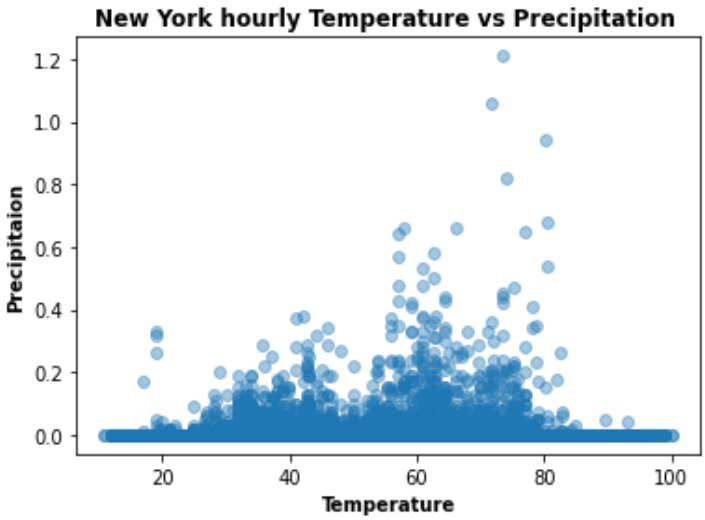
We can understand that JFK is the airport operating with most flights and EWR is the airport with least flights.

Finding the Maximum number of flights headed to unique destination from Origin.



Flights from LGA to DCA are greatest in amount, while flights from EWR to BWI are the least.

Plotting New York hourly temperature vs precipitation data from the ny_hourly dataset:



Time to explore and find answers to some questions

Question 1

a. Calculate the total number of seats for all the planned flights for each destination separately?

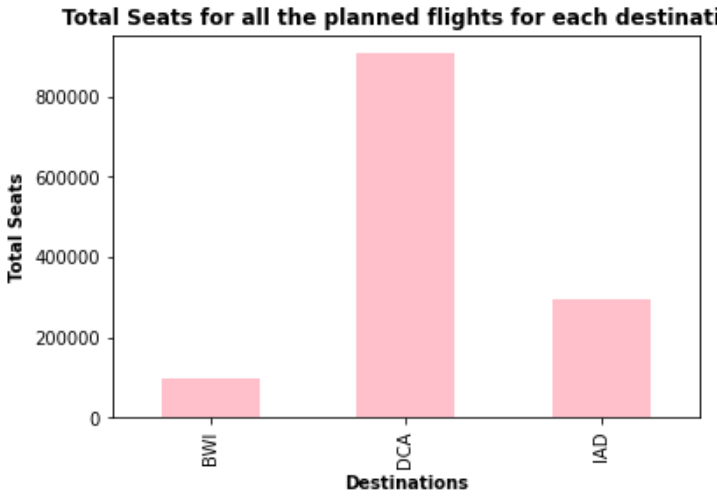
To answer this, we're going to merge the flights and planes datasets on 'tailnum' to find out the exact number of seats per destination.

```
merged_df = pd.merge(flights, planes.loc[:,planes.columns !='year'], on='tailnum')
merged_df.head()
```

```
tot_seats = merged_df.groupby('dest')['seats'].sum()
tot_seats.reset_index(name='Total')
```

	dest	Total
0	BWI	96135
1	DCA	906985
2	IAD	296499

We can see here that, flights to destination DCA has the most number of seats, with 906985 seats, and BWI has the least with 96135 seats. Plotting this for visualization:

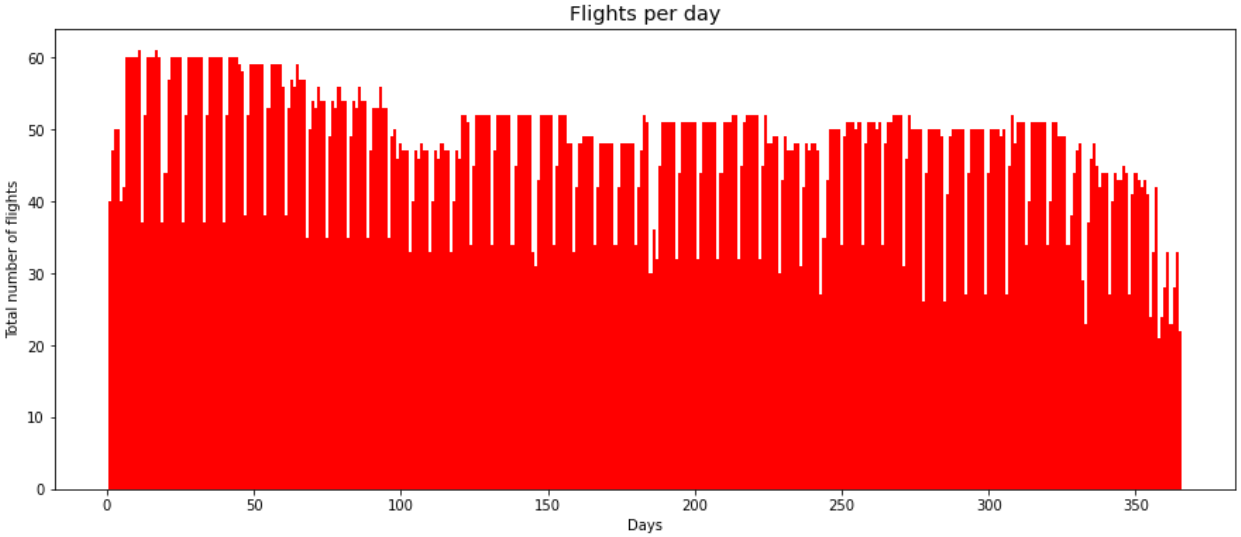


b. What is the day of the year with the highest number of flights?

```
x = flights.groupby(['Date']).Date.size().reset_index(name='Total')
y=x.Total.max()
z=x[x['Total']==y]
print("The day of the year having highest number of flights" '\n\n' ,z)
```

```
The day of the year having highest number of flights

          Date  Total
10 2013-01-11     61
16 2013-01-17     61
```

11th January 2013, & 17th January 2013 are the days with highest number of flights. Will be visualizing total flights on each day of the year.



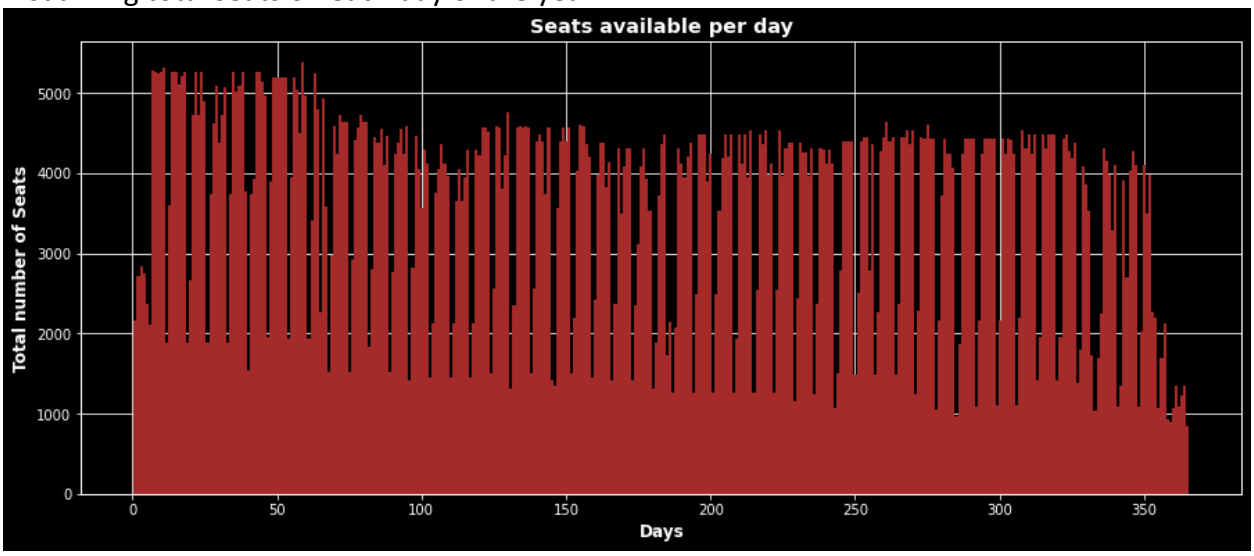
We can see that the frequency of flights is more in the start of the year and low towards the end.

c. What is the day of the year with the highest number of seats available on that day?

```
x = merged_df.groupby(['Date']).seats.sum().reset_index(name='Total')
y=x.Total.max()
z=x[x['Total']==y]
print("The day and year with the highest number of seats available" '\n\n' ,z)
```

```
The day and year with the highest number of seats available

          Date  Total
58 2013-02-28   5379
```

28th Feb 2013 is the day with highest number of available seats, with a total of 5379 seats.
Visualizing total seats on each day of the year.



Question 2

a. What day of the year most cancellations happened?

```
x=cancelled.groupby(['year', 'month','day']).day.count().reset_index(name='Total')
y=x.Total.max()
z=x[x['Total']==y]
print("The day of the year most cancellations happened" '\n\n' ,z)
```

```
The day of the year most cancellations happened

      year month day  Total
112   2013     3   6     46
```

6th March 2013 is day where most cancellations happened, with a total of 46 cancellations.

b. Is there any relationship between the weather datasets and cancellations? If yes, describe it and justify with a statistical model (hypothesis testing).

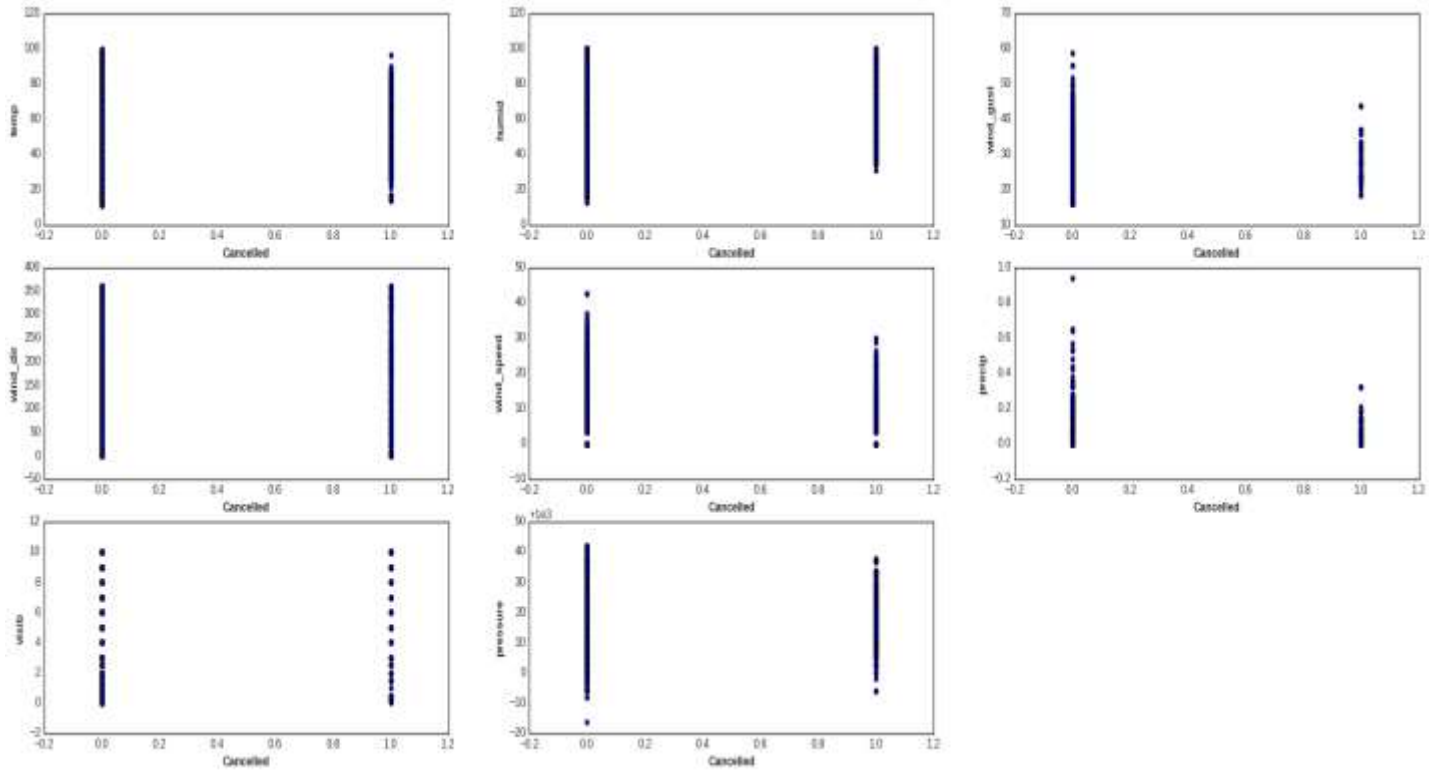
For this we're going to first merger the ny_hourly and ny_daily datasets and creating ny_weather dataframe. And then merging the flights dataframe and ny_weather dataframe to create cancel_ny_weather dataframe.

```
ny_weather = pd.merge(ny_hourly, ny_daily, on='Date')
```

```
flights['date_time'] = pd.to_datetime(flights[['year', 'month', 'day', 'hour']])
ny_weather['date_time'] = pd.to_datetime(ny_weather[['year', 'month', 'day', 'hour']])
```

```
cancel_ny_weather = flights.merge(ny_weather, on=['year','month','day','origin','date_time','Date','hour'],how='inner')
cancel_ny_weather.head()
```

Plotting the weather with respect to cancellations. Cancelled = 0 is false, meaning the flight is not cancelled, cancelled = 1 is true, means the flight is cancelled.



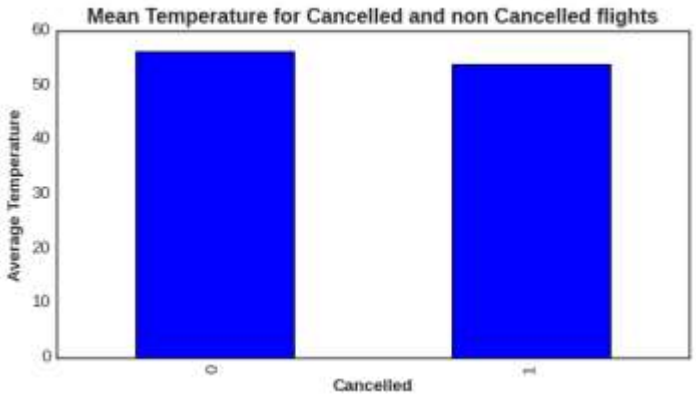
We'll answer this question in a step-by-step process. First let's try to find if temperature has anything to do with cancellations.

Temperature VS Cancellations

```
can_temp = cancel_ny_weather.groupby(['Cancelled']).temp.mean()
print(can_temp)
can_temp.plot(kind='bar', figsize=(8,4))
plt.title('\nMean Temperature for Cancelled and non Cancelled flights', fontweight='bold')
plt.xlabel('Cancelled', fontweight='bold')
plt.ylabel('Average Temperature', fontweight='bold')
```

Answer to this code is: Cancelled 0 - 56.055374, Cancelled 1 - 53.733333

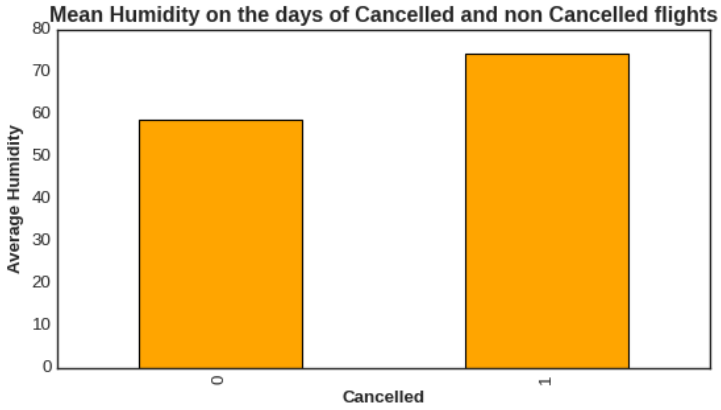
There isn't a significant difference of mean temperatures on cancelled days and non-cancelled days. We can say that temperature has no effect on cancellations.



Humidity VS Cancellations

```
can_hum = cancel_ny_weather.groupby(['Cancelled']).humid.mean()
print(can_hum)
can_hum.plot(kind='bar', figsize=(8,4), color='orange')
plt.title('\nMean Humidity on the days of Cancelled and non Cancelled flights', fontweight='bold')
plt.xlabel('Cancelled', fontweight='bold')
plt.ylabel('Average Humidity', fontweight='bold')
```

```
Cancelled
0    58.710149
1    74.249530
```



We can analyze from the picture that there is a strong relation between humidity and cancellations. Let's perform a hypothesis testing to check if that's true or not.

Null hypothesis: There is no significant relation between Humidity and Cancellation

```
non_canl = cancel_ny_weather[cancel_ny_weather.Cancelled == 0].humid
cancl = cancel_ny_weather[cancel_ny_weather.Cancelled == 1].humid
tstas, pvalue = stats.ttest_ind(non_canl, cancl, equal_var=False)
print("P-value is : {} ".format(pvalue))
```

```
P-value is : 1.1678775702583998e-135
```

Since the p-value is less than 0.05 we reject the null-hypothesis. Which means there is a significant relation between Humidity and Cancellations

Wind Direction vs Cancellations

```
can_win_dir = cancel_ny_weather.groupby(['Cancelled']).wind_dir.mean()
print(can_win_dir)
can_win_dir.plot(kind='pie', figsize=(6,4))
plt.title('\nMean Wind Direction on the days of Cancelled and non Cancelled flights', fontweight='bold')
plt.xlabel('Cancelled', fontweight='bold')
plt.ylabel('Average Wind Direction', fontweight='bold')
```

```
Cancelled
0    204.307254
1    175.636911
```

We can see here that there is a significant relation between Wind Direction and Cancellations.

Null hypothesis: There is no significant relation between Wind Direction and Cancellation

```
non_canl = cancel_ny_weather[cancel_ny_weather.Cancelled == 0].wind_dir.dropna()
cancl = cancel_ny_weather[cancel_ny_weather.Cancelled == 1].wind_dir.dropna()
tstas, pvalue = stats.ttest_ind(non_canl, cancl, equal_var=False)
print("P-value is : {} ".format(pvalue))
```

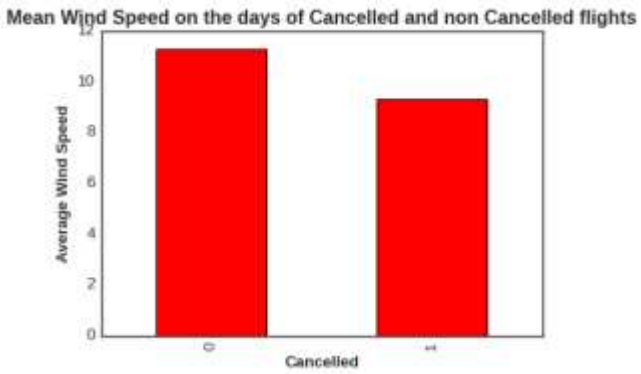
```
P-value is : 2.073726447147906e-17
```

Since the p-value is less than 0.05 we reject the null-hypothesis. Which means there is a significant relation between Wind Direction and Cancellations

Wind Speed VS Cancellations

```
can_win_sp = cancel_ny_weather.groupby(['Cancelled']).wind_speed.mean()
print(can_win_sp)
can_win_sp.plot(kind='bar', figsize=(6,4), color ='red')
plt.title('\nMean Wind Speed on the days of Cancelled and non Cancelled flights', fontweight='bold')
plt.xlabel('Cancelled', fontweight='bold')
plt.ylabel('Average Wind Speed', fontweight='bold')
```

```
Cancelled
0    11.279686
1     9.313370
```

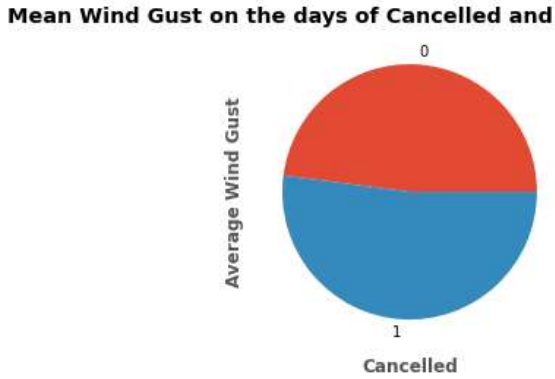


The difference between mean wind speed on the day of cancelled and non-cancelled flights is around 2. We can infer that there is no significant relation between wind speed and cancellations.

Wind Gust VS Cancellations

```
can_win_gus = cancel_ny_weather.groupby(['Cancelled']).wind_gust.mean()
print(can_win_gus)
can_win_gus.plot(kind='pie', figsize=(6,4))
plt.title('\nMean Wind Gust on the days of Cancelled and non Cancelled flights', fontweight='bold')
plt.xlabel('Cancelled', fontweight='bold')
plt.ylabel('Average Wind Gust', fontweight='bold')
```

```
Cancelled
0    25.425206
1    27.607547
```



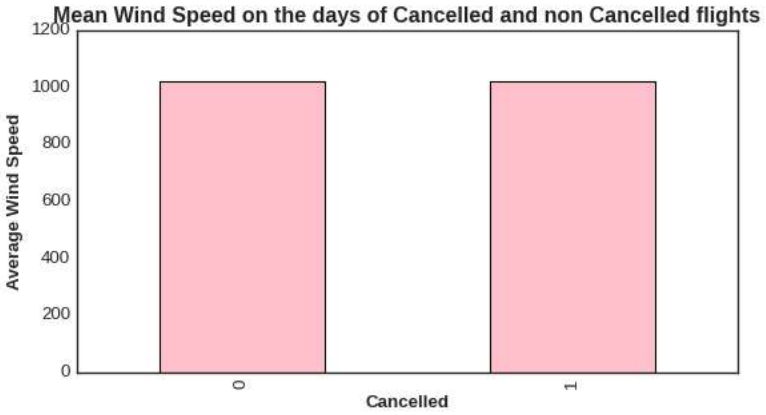
We can see here in the plot that there is no significant relation between Wind Gust and Cancellations. As wind gust is almost similar on both cancelled days and non-cancelled days.

Pressure VS Cancellations

```
can_pre = cancel_ny_weather.groupby(['Cancelled']).pressure.mean()
print(can_pre)
can_pre.plot(kind='bar', figsize=(6,4), color ='pink')
plt.title('\nMean Wind Speed on the days of Cancelled and non Cancelled flights', fontweight='bold')
plt.xlabel('Cancelled', fontweight='bold')
plt.ylabel('Average Wind Speed', fontweight='bold')
```

```
Cancelled
0    1017.927545
1    1017.201313
```

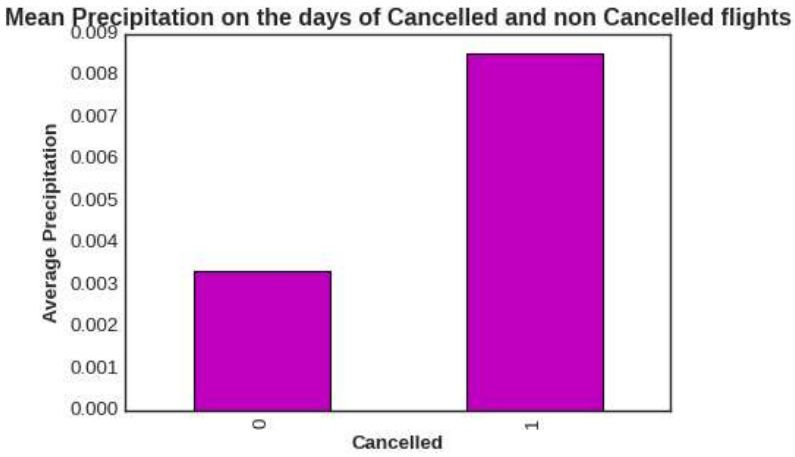
As we can see the difference between pressure on cancelled days and non-cancelled days is negligible. It means there is significant relation between Pressure and Cancellations.



Precipitation VS Cancellations

```
can_precip = cancel_ny_weather.groupby(['Cancelled']).precip.mean()
print(can_precip)
can_precip.plot(kind='bar', figsize=(6,4), color ='m')
plt.title('\nMean Precipitation on the days of Cancelled and non Cancelled flights', fontweight='bold')
plt.xlabel('Cancelled', fontweight='bold')
plt.ylabel('Average Precipitation', fontweight='bold')
```

```
Cancelled
0    0.003330
1    0.008539
```



We can see that there is a significant relation between Precipitation and Cancellations. Let's use hypothesis testing to prove it.

Null Hypothesis: There is no significant relation between precipitation and cancellations

```
non_canl = cancel_ny_weather[cancel_ny_weather.Cancelled == 0].precip
cancl = cancel_ny_weather[cancel_ny_weather.Cancelled == 1].precip
tstas, pvalue = stats.ttest_ind(non_canl, cancl, equal_var=False)
print("P-value is : {} ".format(pvalue))
```

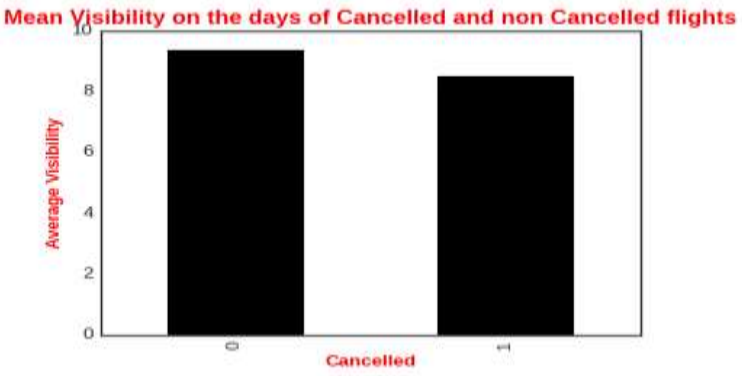
```
P-value is : 8.272943295614399e-07
```

Since the p-value is less than 0.05, we must reject the null-hypothesis. Which means there is a significant relation between precipitation and cancellations. It means, more precipitation leads to more cancellations.

Visibility VS Cancellations

```
can_visib = cancel_ny_weather.groupby(['Cancelled']).visib.mean()
print(can_visib)
can_visib.plot(kind='bar', figsize=(6,4), color ='black')
plt.title('\nMean Visibility on the days of Cancelled and non Cancelled flights', fontweight='bold',color='r')
plt.xlabel('Cancelled', fontweight='bold',color='r')
plt.ylabel('Average Visibility', fontweight='bold',color='r')
```

```
Cancelled
0    9.368435
1    8.493984
```



We can see that mean visibility is less on cancelled days, so there is a relation between these two.

Null Hypothesis: No significant relation between Visibility and Cancellations

Since p-value is less than 0.05 we must reject the null hypothesis. Which means there is a relation between these two. But we can see that there is negative correlation, meaning lesser visibility leads to more cancellations.

```
non_canl = cancel_ny_weather[cancel_ny_weather.Cancelled == 0].visib
cancl = cancel_ny_weather[cancel_ny_weather.Cancelled == 1].visib
tstas, pvalue = stats.ttest_ind(non_canl, cancl, equal_var=False)
print("P-value is : {} ".format(pvalue))
```

```
P-value is : 1.021281099487865e-19
```

c. Is there any relationship between the Federal Holiday Schedule and cancellations? If yes, describe it and justify with a statistical model (hypothesis testing).

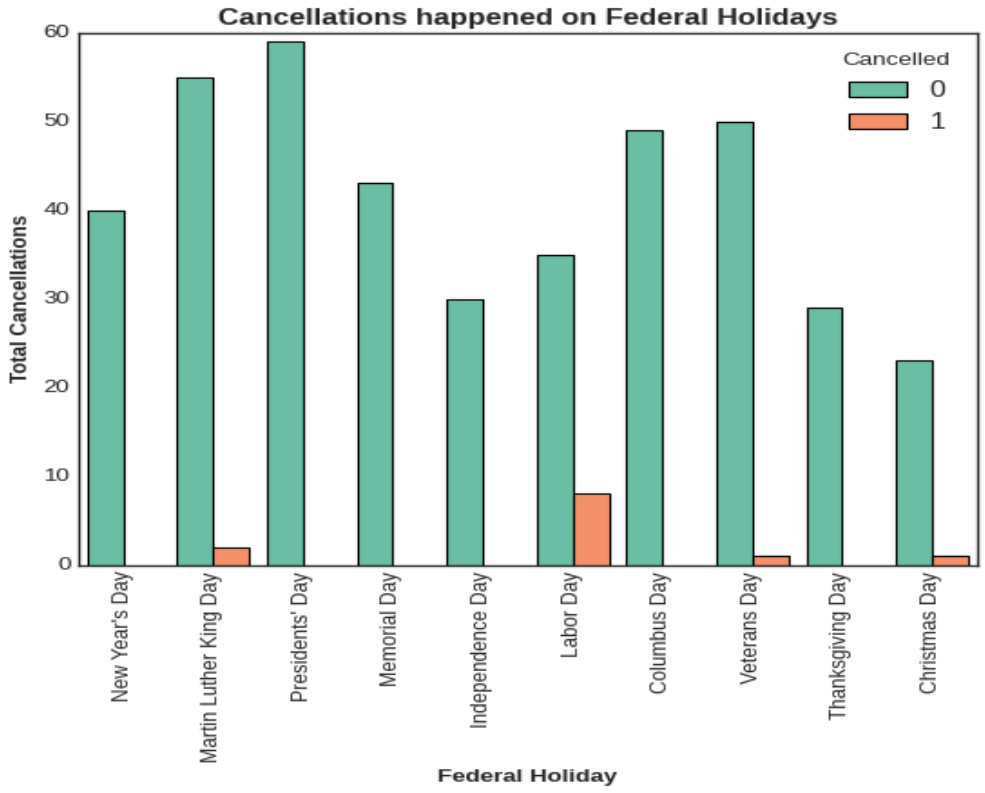
```
fed_holi = pd.read_excel('https://raw.githubusercontent.com/SravaniRVS/DATA601/main/Project%202/federal-holidays-2013.xlsx',skiprows=1,nrows=10)
fed_holi['Date'] = pd.to_datetime(fed_holi['Date'])
fed_holi
```

```
fed_cncl = flights.groupby('Date').Cancelled.value_counts().reset_index(name='Total_Cancelled')
fed_cncl_df = fed_holi.merge(fed_cncl,on='Date')
fed_cncl_df
```

```
fed_non_cnclll = fed_cncl_df[fed_cncl_df['Cancelled']==0].Total_Cancelled.sum()
fed_cnclll = fed_cncl_df[fed_cncl_df['Cancelled']==1].Total_Cancelled.sum()
print('Total non-cancelled flights on Federal Holidays : {}'.format(fed_non_cnclll),'\n')
print('Total cancelled flights on Federal Holidays : {}'.format(fed_cnclll))
```

```
Total non-cancelled flights on Federal Holidays : 413

Total cancelled flights on Federal Holidays : 12
```

First reading the excel files containing the federal holiday data using pandas read_excel option. Then merging the flights dataframe with the fed_holi dataframe on 'Date' to create a new dataframe. After running the above-mentioned code, we got the answer that, there are only 12 flights that got cancelled on federal holidays, while 413 flights were not cancelled in the year 2013. Visualizing to depict just how much the cancelled and non-cancelled flights differ in number.



We can observe that the number of non-cancelled flights on federal holidays is much higher, at 413 flights, than the number of cancelled flights which are a mere 12.

This means there is no significant relation between Federal Holidays & Cancellations.

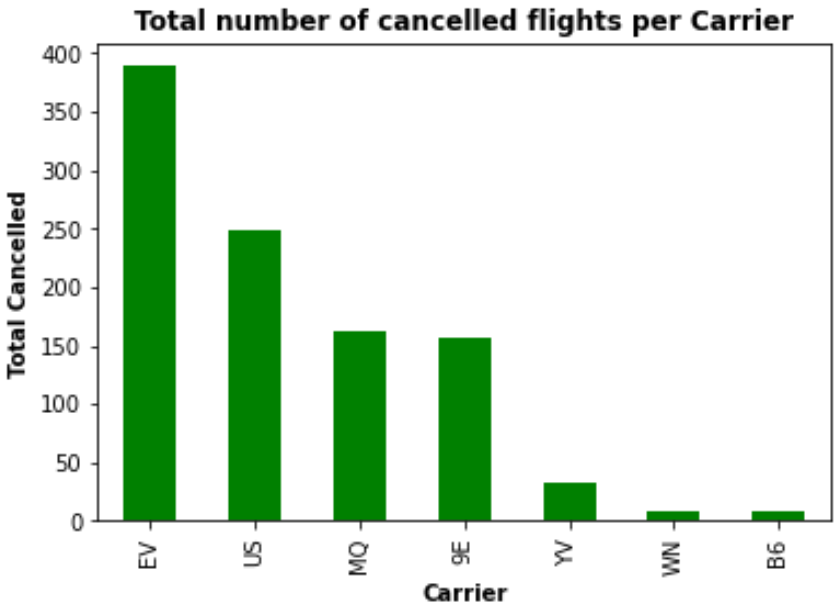
d. What is the total number of seats for the cancelled flights? If we assume the average flight price of $50, what is the total economic loss?

```
tot = cancelled.seats.sum()
print("Total number of seats for the cancelled flights are : ", tot,"\n")
sum_can_fli = cancelled.carrier.value_counts().sum()
print('Total number of flights cancelled : {}'.format(sum_can_fli),'\n')
print("Total Economic loss : ${}".format(sum_can_fli*50))
```

```
Total number of seats for the cancelled flights are :  29157.0

Total number of flights cancelled : 1005

Total Economic loss : $50250
```

After merging the flights and planes dataframes before for one of our questions. I've created another dataframe called 'cancelled' where merged_df[merged_df['Cancelled'] ==1]. It only has the data of the cancelled flights. From that we can calculate the total number of seats on those flights. Which are 29,157. The total number of flights cancelled are 1005, if the average flight price is $50 then the total economic loss would be average price * total cancelled flights, which is: 50 * 1005 = $50,250.

Plotting the number of cancelled flights per carrier:



e. Determine the ratio of cancelled flights/planned flights for each airline company, list it, and determine the most and least reliable airline company (most reliable = the one that has the smallest ratio of cancelled/planned)

```
cancel = cancelled.groupby(['carrier']).carrier.size().reset_index(name='Cancelled')
print('Total cancelled flights per carrier:\n',cancel,'\n')

planned = flights.groupby(['carrier']).carrier.size().reset_index(name='Planned')
print('Total planned flights per carrier:\n',planned,'\n')

temp_df = planned.merge(cancel,on='carrier',how='left').fillna(0)


def ratio(numerator,denominator):
      return str(numerator)+'/'+str(denominator)
def value(c,p):
      return c/p
temp_df['Ratio'] = temp_df.apply(lambda x: ratio(x['Cancelled'], x['Planned']),axis=1)
temp_df['Ratio_Value'] = temp_df.apply(lambda x: value(x['Cancelled'], x['Planned']), axis=1)
temp_df.sort_values(by='Ratio_Value',ascending=True)
```

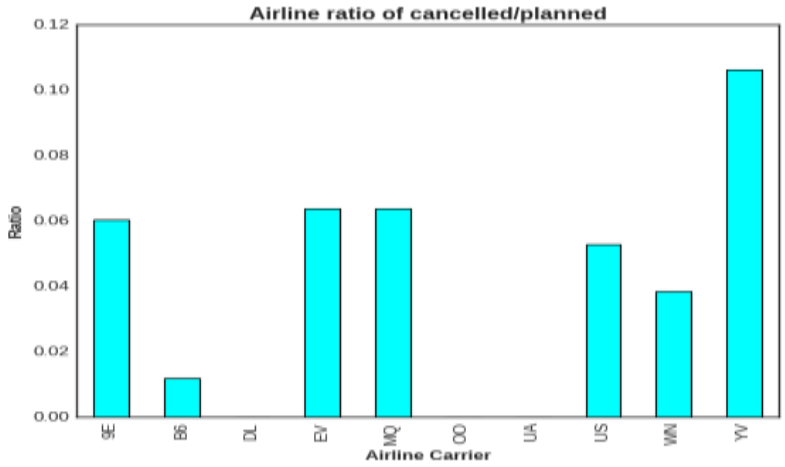
If we execute this code, the output will be:

```
Total cancelled flights per carrier:
   carrier  Cancelled
0       9E        156
1       B6          8
2       EV        389
3       MQ        163
4       US        248
5       WN          8
6       YV         33

Total planned flights per carrier:
   carrier  Planned
0       9E     2594
1       B6      675
2       DL        2
3       EV     6117
4       MQ     2559
5       OO        1
6       UA        3
7       US     4716
8       WN      208
9       YV      311
```

	carrier	Planned	Cancelled	Ratio	Ratio_Value
2	DL	2	0.0	0.0/2	0.000000
5	OO	1	0.0	0.0/1	0.000000
6	UA	3	0.0	0.0/3	0.000000
1	B6	675	8.0	8.0/675	0.011852
8	WN	208	8.0	8.0/208	0.038462
7	US	4716	248.0	248.0/4716	0.052587
0	9E	2594	156.0	156.0/2594	0.060139
3	EV	6117	389.0	389.0/6117	0.063593
4	MQ	2559	163.0	163.0/2559	0.063697
9	YV	311	33.0	33.0/311	0.106109

- **DL**, **OO**, **UA**, are the most reliable carries with **0** cancellations.
- If we only consider carriers with at least one cancellation, then US is the most reliable carrier with cancellation ratio of **0.002757**
- **YV** is the least reliable carrier with a cancellation ratio of **0.106109**

Visualizing the results:



Question/Task 3

a. Calculate the average arrival delay for all the flights that took place in the same day and plot it (x = 1:365, y = daily average delay). On the same plot, please mark the Federal Holidays from the federal-holidays-2013.xlsx dataset.

Calculating the average arrival delay:

```
avg_arr_delay = flights.groupby('Date')['arr_delay'].mean().reset_index(name='Avg_Arr_Delay')
avg_arr_delay
```

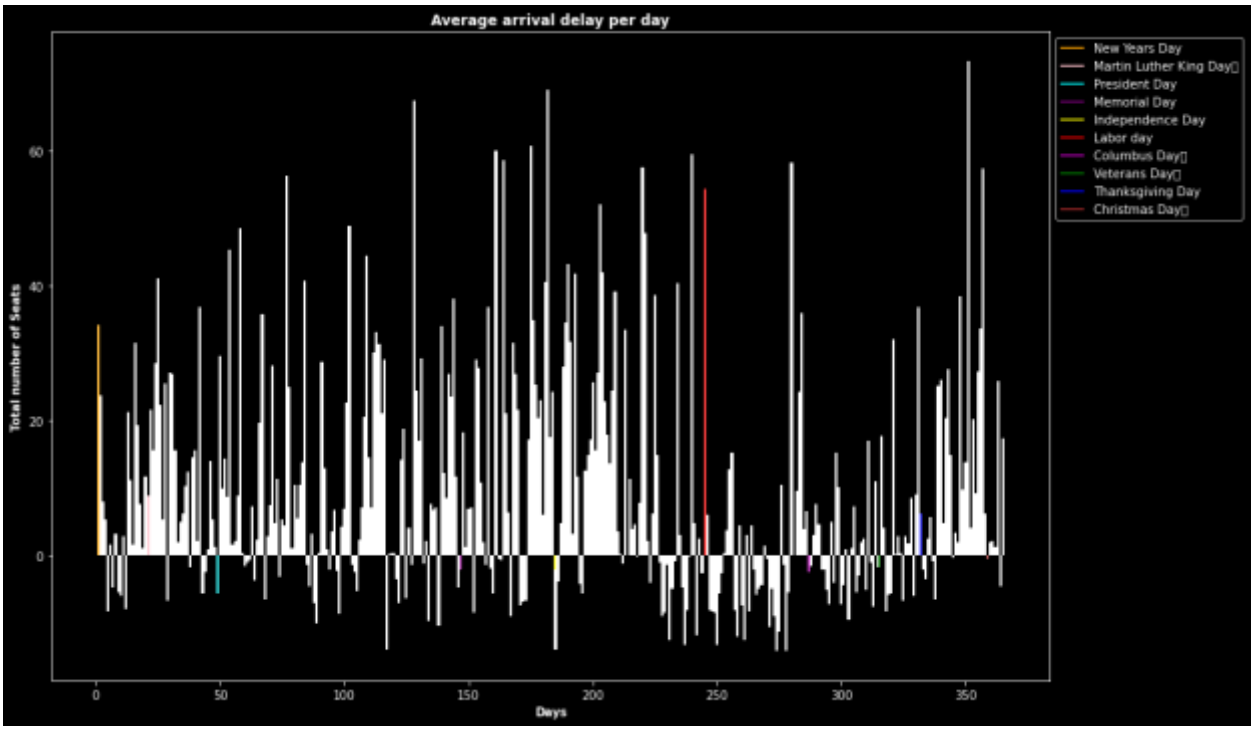
	Date	Avg_Arr_Delay
0	2013-01-01	34.075000
1	2013-01-02	23.702128
2	2013-01-03	7.880000
3	2013-01-04	5.220000
4	2013-01-05	-8.325000
...	...	...
360	2013-12-27	2.090909
361	2013-12-28	1.217391
362	2013-12-29	25.785714
363	2013-12-30	-4.636364
364	2013-12-31	17.318182

Finding the day of federal holidays:

```
import datetime
fed_holi_arr_del = fed_holi.merge(avg_arr_delay, on='Date',how='inner')
fed_holi_arr_del['Day_of_the_year'] = fed_holi_arr_del['Date'].dt.strftime('%j')
fed_holi_arr_del
```

	Date	Federal holiday	Day of the week	Avg_Arr_Delay	Day_of_the_year
0	2013-01-01	New Year's Day	Tuesday	34.075000	001
1	2013-01-21	Martin Luther King Day	Monday	8.771930	021
2	2013-02-18	Presidents' Day	Monday	-5.745763	049
3	2013-05-27	Memorial Day	Monday	-2.232558	147
4	2013-07-04	Independence Day	Thursday	-14.033333	185
5	2013-09-02	Labor Day	Monday	54.279070	245
6	2013-10-14	Columbus Day	Monday	-2.428571	287
7	2013-11-11	Veterans Day	Monday	-1.784314	315
8	2013-11-28	Thanksgiving Day	Thursday	6.103448	332
9	2013-12-25	Christmas Day	Wednesday	-0.500000	359

Plotting these values in a single plot:



b. Is there a correlation between the weather datasets and daily average arrival delay? Justify your answer with a statistical model (hypothesis testing)

For this we're going to first merge the ny_weather and flights datasets and creating weather_flights dataframe.

```
weather_flights = flights.merge(ny_weather, on=['Date','origin','year','month','day'])
```

We'll answer this question in a step-by-step process. First let's try to find if temperature has anything to do with average arrival delays.

Here we plot the heatmap to see if there is any correlation between the variables.

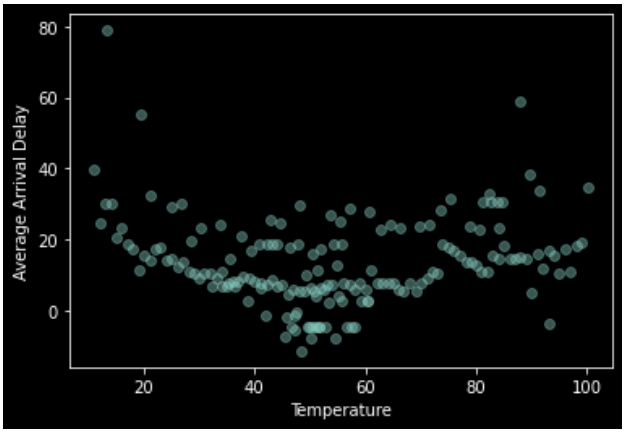
Temperature vs. Delay

```
plt.figure(figsize = (15,10))
variables = ['temp','humid','precip','visib','wind_gust','wind_speed','wind_dir','pressure','arr_delay']
weather_flights[variables].corr(method="pearson")
correlation = weather_flights[variables].corr(method="pearson")
sns.heatmap(correlation,xticklabels = correlation.columns,yticklabels = correlation.columns, annot=True)
```

This is the heatmap showing the correlation

Here we try to plot the scatter plot between the temperature and average arrival delay.

```
# aggregate average departure delay by temperature
temp_delay = weather_flights.groupby('temp', as_index = False)['arr_delay'].mean()
temp_delay
# plot a scatter plot that takes temperature as predictor, and departure delay as response
plt.xlabel("Temperature")
plt.ylabel("Average Arrival Delay")
plt.scatter(temp_delay.temp, temp_delay.arr_delay, alpha=0.4)
```

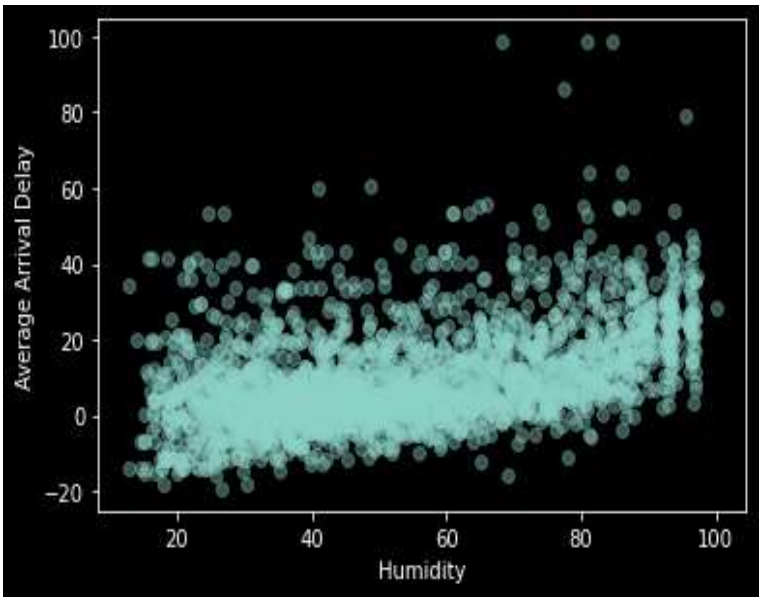


We can see here that temperature has just 0.021 correlation with arrival delay. We can infer from that, that there is no significant effect of temperature on arrival delays.

Now we try to find relation between Humidity and average arrival delay.

We try to plot the scatter plot between the humidity and average arrival delay.

```
# aggregate average departure delay by humidity
humid_delay = weather_flights.groupby('humid', as_index = False)['arr_delay'].mean()

# plot a scatter plot that takes humidity as predictor, and departure delay as response
plt.xlabel("Humidity")
plt.ylabel("Average Arrival Delay")
plt.scatter(humid_delay.humid, humid_delay.arr_delay, alpha=0.4)
```



We can see here in the plot that as humidity increases arrival delay is also increasing. That means it has a relation. Try to find out by performing a null hypothesis.

Null Hypothesis: "There is no significant relation between humidity and arrival delay. "is correct or not by checking the p value.

```
non_canl = humid_delay.arr_delay
cancl = humid_delay.humid
tstas, pvalue = stats.ttest_ind(non_canl, cancl, equal_var=False)
print("P-value is : {} ".format(pvalue))
```

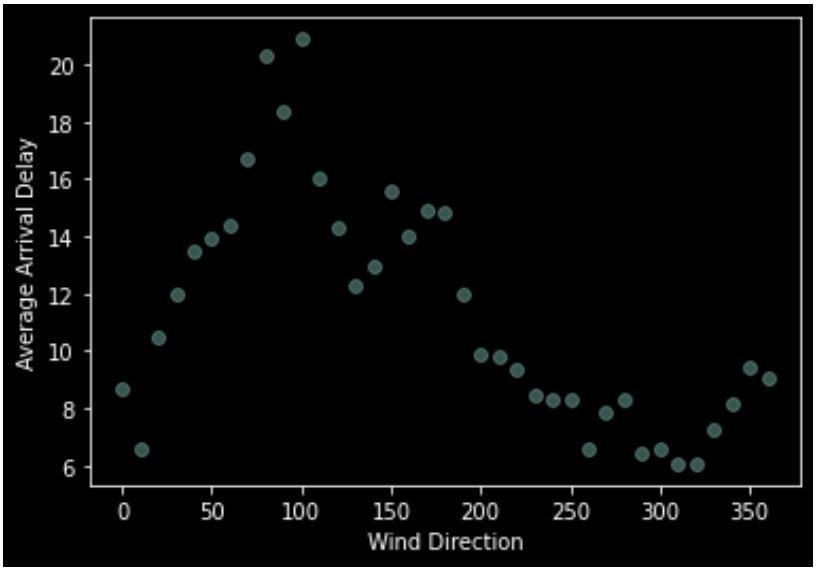
```
P-value is : 0.0
```

Since the p-value is less than 0.05. We have to reject the null hypothesis, which means there is a significant relation between humidity and arrival delays.

Now we try to find relation between Wind Direction and average arrival delay.

We try to plot the scatter plot between the Wind direction and average arrival delay.

```
# aggregate average departure delay by wind direction
dir_delay = weather_flights.groupby('wind_dir', as_index = False)['arr_delay'].mean()

# plot a scatter plot that takes wind direction as predictor, and departure delay as response
plt.xlabel("Wind Direction")
plt.ylabel("Average Arrival Delay")
plt.scatter(dir_delay.wind_dir, dir_delay.arr_delay, alpha=0.4)
```



We can observe that there is a negative correlation between wind direction and average arrival delay. Let's try to find out by performing null hypothesis.

Null Hypothesis: There is no significant relation between wind direction and arrival delay.

```
non_canl = dir_delay.arr_delay
cancl = dir_delay.wind_dir
tstas, pvalue = stats.ttest_ind(non_canl, cancl, equal_var=False)
print("P-value is : {} ".format(pvalue))
```

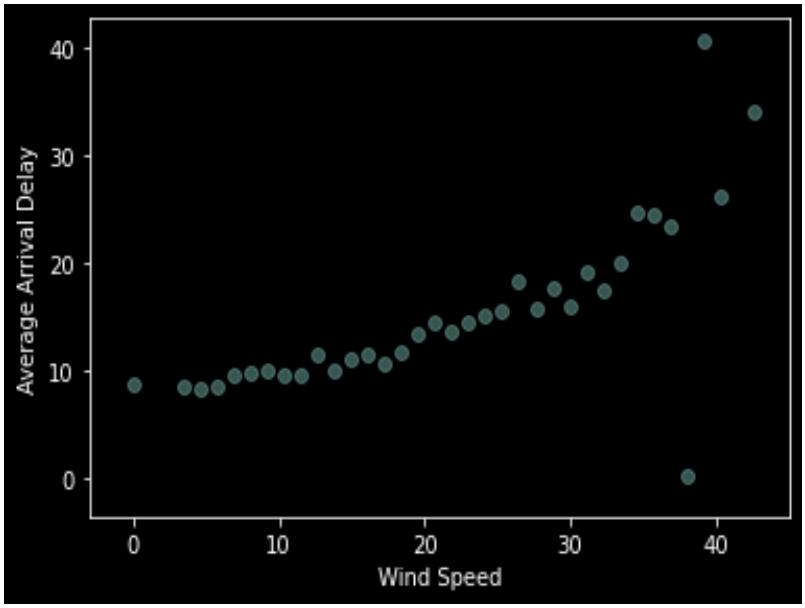
```
P-value is : 2.514571499799725e-11
```

Since the p-value is less than 0.05 we have to reject the null hypothesis, that means there is a significant relation between wind direction and arrival delay.

Now we try to find relation between Wind Speed and average arrival delay.

We try to plot the scatter plot between the Wind Speed and average arrival delay.

```
# aggregate average departure delay by wind speed
speed_delay = weather_flights.groupby('wind_speed', as_index = False)['arr_delay'].mean()

# plot a scatter plot that takes wind speed as predictor, and departure delay as response
plt.xlabel("Wind Speed")
plt.ylabel("Average Arrival Delay")
plt.scatter(speed_delay.wind_speed, speed_delay.arr_delay, alpha=0.4)

# filter out out-liers
plt.xlim(-3,45)
```



Wind speed has a correlation value of 0.02 with arrival delay. That is not a significant amount. But let's perform hypothesis testing anyway to find this out.

Null hypothesis: There is no significant relation between wind direction and arrival delay.

```
non_canl = speed_delay.arr_delay
cancl = speed_delay.wind_speed
tstas, pvalue = stats.ttest_ind(non_canl, cancl, equal_var=False)
print("P-value is : {} ".format(pvalue))
```

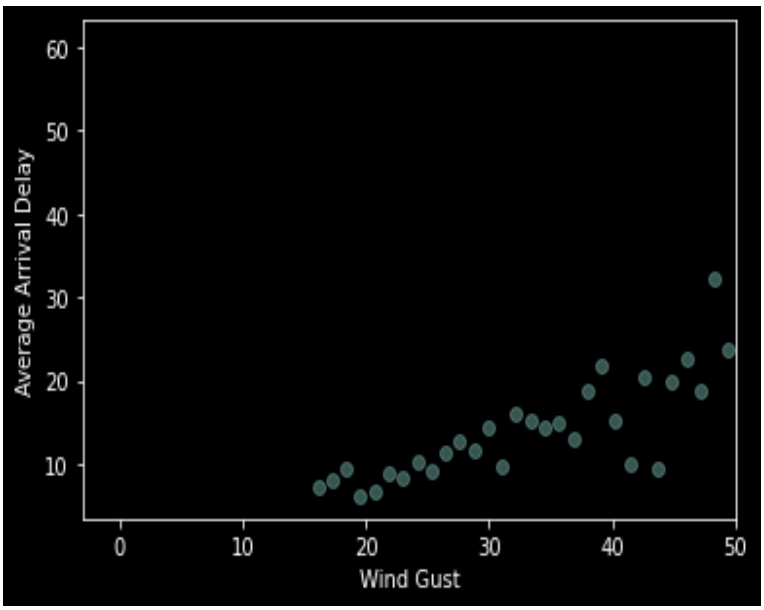
```
P-value is : 0.22166683321789096
```

Since the p-value is more than 0.05, we have to accept the null hypothesis, that means there is no significant relation between wind speed and arrival delay.

Now we try to find relation between Wind Gust and average arrival delay.

We try to plot the scatter plot between the Wind Gust and average arrival delay.

```
# aggregate average departure delay by wind gust
gust_delay = weather_flights.groupby('wind_gust', as_index = False)['arr_delay'].mean()

# plot a scatter plot that takes wind gust as predictor, and departure delay as response
plt.xlabel("Wind Gust")
plt.ylabel("Average Arrival Delay")
plt.scatter(gust_delay.wind_gust, gust_delay.arr_delay, alpha=0.4)

# filter out out-liers
plt.xlim(-3,50)
```



We can see in this plot that arrival delay increases when wind gust increases. Let's use hypothesis testing to find out more.

Null hypothesis: There is no significant relation between wind gust and arrival delay.

```
non_canl = gust_delay.arr_delay
cancl = gust_delay.wind_gust
tstas, pvalue = stats.ttest_ind(non_canl, cancl, equal_var=False)
print("P-value is : {} ".format(pvalue))
```

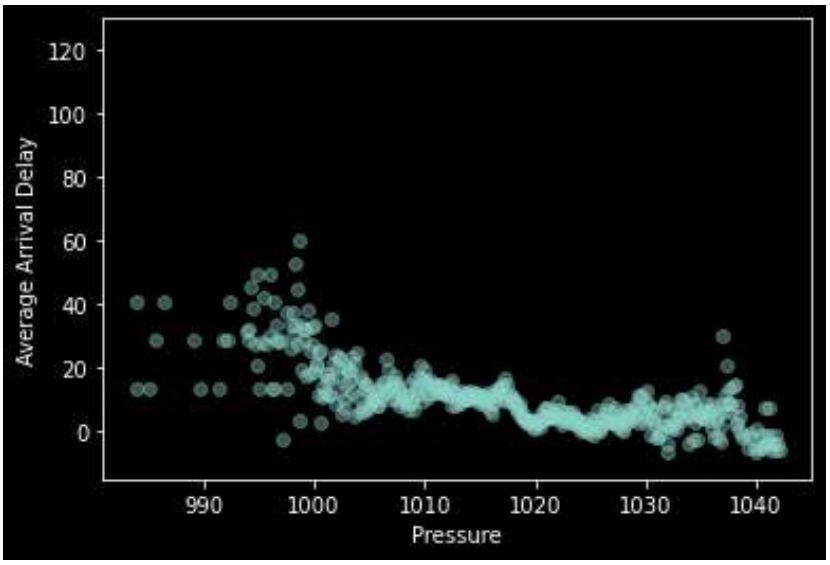
```
P-value is : 8.363215537002272e-10
```

Since, the p-value is less than 0.05 we have to reject the null hypothesis, that means there is a significant relation between wind gust and arrival delay.

Now we try to find relation between Pressure and average arrival delay.

We try to plot the scatter plot between the Pressure and average arrival delay.

```
# aggregate average departure delay by pressure
pre_delay = weather_flights.groupby('pressure', as_index = False)['arr_delay'].mean()

# plot a scatter plot that takes the pressure as predictor, and departure delay as response
plt.xlabel("Pressure")
plt.ylabel("Average Arrival Delay")
plt.scatter(pre_delay.pressure, pre_delay.arr_delay, alpha=0.4)

# filter out out-liers
plt.ylim(-15,130)
```



Null hypothesis: There is no significant relation between pressure and arrival delay.

```
non_canl = pre_delay.arr_delay
cancl = pre_delay.pressure
tstas, pvalue = stats.ttest_ind(non_canl, cancl, equal_var=False)
print("P-value is : {} ".format(pvalue))
```

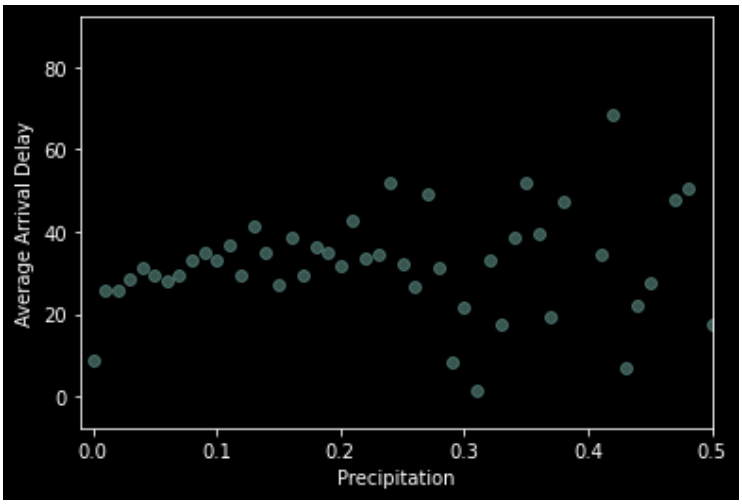
```
P-value is : 0.0
```

Since, the p-value is less than 0.05 we have to reject the null hypothesis, that means there is a significant relation between pressure and arrival delay.

Now we try to find relation between Precipitation and average arrival delay.

We try to plot the scatter plot between the Precipitation and average arrival delay.

```
# aggregate average departure delay by precipitation
precip_delay = weather_flights.groupby('precip', as_index = False)['arr_delay'].mean()

# plot a scatter plot that takes the precipitation as predictor, and departure delay as response
plt.xlabel("Precipitation")
plt.ylabel("Average Arrival Delay")
plt.scatter(precip_delay.precip, precip_delay.arr_delay, alpha=0.4)

# filter out out-liers
plt.xlim(-0.01,0.5)
```

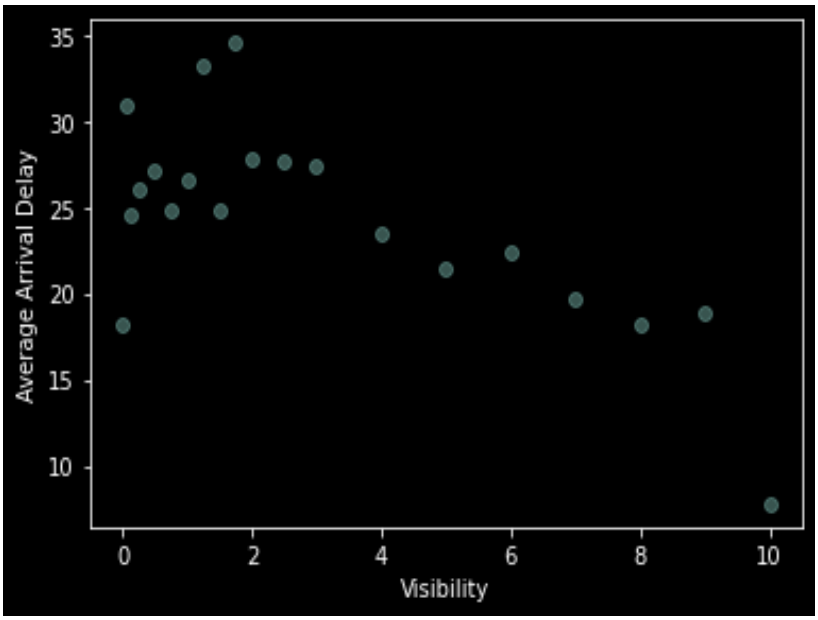


The correlation value of precipitation and arrival delay is just 0.07. That means there is no significant relation between these two. We can also see in the plot that arrival delay remains mostly similar even when the precipitation increases.

Now we try to find relation between Visibility and average arrival delay.

We try to plot the scatter plot between the Visibility and average arrival delay.

```
# aggregate average departure delay by visibility
visib_delay = weather_flights.groupby('visib', as_index = False)['arr_delay'].mean()

# plot a scatter plot that takes visibility as predictor, and departure delay as response
plt.xlabel("Visibility")
plt.ylabel("Average Arrival Delay")
plt.scatter(visib_delay.visib, visib_delay.arr_delay, alpha=0.4)
```



Visibility has a correlation value of -0.12 with arrival delay, that means it has a negative correlation. Here we can see that if the level of visibily decreases arrival delay increases. Using hypothesis testing to find out more.

Null Hypothesis: There is no significant relation between visibility and arrival delay.

```
non_canl = visib_delay.arr_delay
cancl = visib_delay.visib
tstas, pvalue = stats.ttest_ind(non_canl, cancl, equal_var=False)
print("P-value is : {} ".format(pvalue))
```

```
P-value is : 2.6847187671254106e-14
```

Since the p-value is less than 0.05, we have to reject the null hypothesis, that means there is relation between visibility and arrival delay.

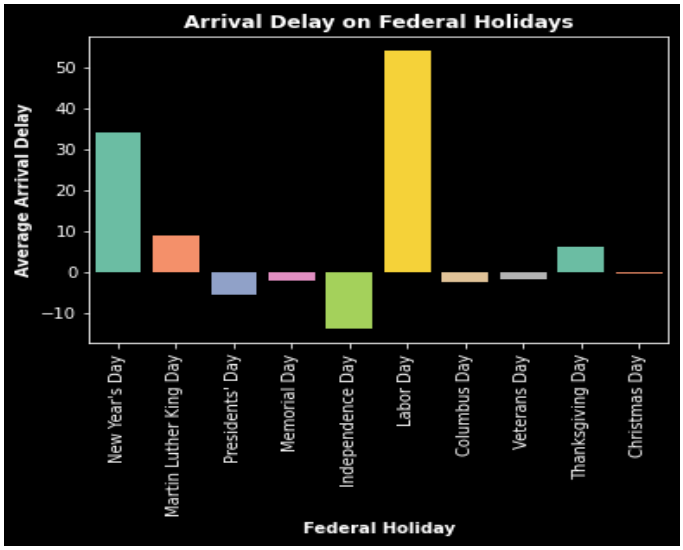
c. Is there a correlation between the Federal Holiday Schedule and daily average arrival delay?

```
avg_arr_delay = flights.groupby('Date')['arr_delay'].mean().reset_index(name='Avg_Arr_Delay')
avg_arr_delay
import datetime

fed_holi_arr_del = fed_holi.merge(avg_arr_delay, on='Date',how='inner')
fed_holi_arr_del
```

	Date	Federal holiday	Day of the week	Avg_Arr_Delay
0	2013-01-01	New Year's Day	Tuesday	34.075000
1	2013-01-21	Martin Luther King Day	Monday	8.771930
2	2013-02-18	Presidents' Day	Monday	-5.745763
3	2013-05-27	Memorial Day	Monday	-2.232558
4	2013-07-04	Independence Day	Thursday	-14.033333
5	2013-09-02	Labor Day	Monday	54.279070
6	2013-10-14	Columbus Day	Monday	-2.428571
7	2013-11-11	Veterans Day	Monday	-1.784314

Plotting these values:

```
sns.barplot(x='Federal holiday',y='Avg_Arr_Delay', data=fed_holi_arr_del,palette='Set2',saturation=0.9 )
plt.xticks(rotation=90)
plt.title('Arrival Delay on Federal Holidays', fontweight='bold')
plt.xlabel('Federal Holiday', fontweight='bold')
plt.ylabel('Average Arrival Delay', fontweight='bold')
```



As we can see from the above figure that the arrival delay is negative for 5 of these days, thus there is no significant relation.

d. Calculate the average arrival delay for all the flights for each arrival airport (e.g. IAD, DCA, and BWI) and determine most and least reliable (most reliable = the one that has the shortest average delay)

```
dest_avg_arr = flights.groupby('dest').arr_delay.mean().reset_index(name='Average_Arrival_Delay')
dest_avg_arr
```

	dest	Average_Arrival_Delay
0	BWI	10.160584
1	DCA	8.512004
2	IAD	13.093158

We can understand from this that **DCA** is the most reliable airport with least average arrival delay of **8.512004**. And **IAD** is the least reliable with average arrival delay of **13.093158.** We can plot this like:

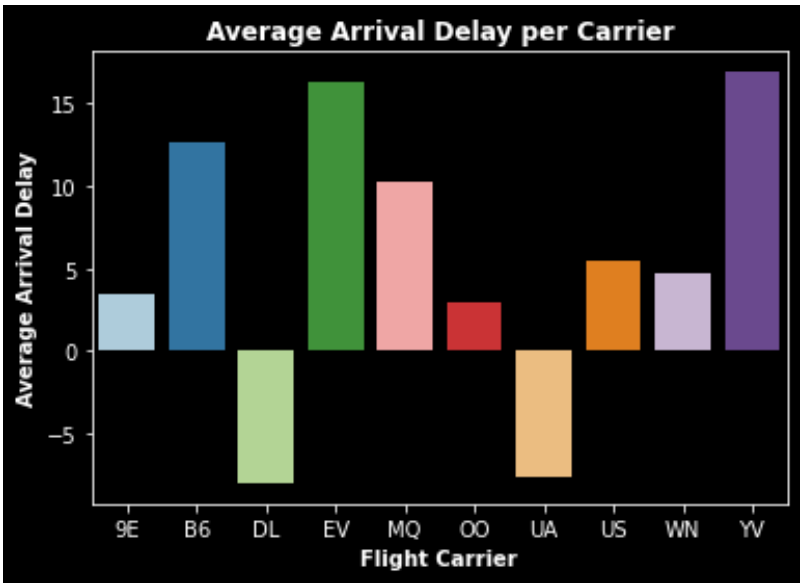
e. Calculate the average arrival delay for all the airlines and determine most and least reliable (most reliable = the one that has the shortest average delay)

```
carrier_arr_delay = flights.groupby('carrier').arr_delay.mean().reset_index(name='Average_Arrival_Delay')
carrier_arr_delay
```

	carrier	Average_Arrival_Delay
0	9E	3.409792
1	B6	12.653333
2	DL	-8.000000
3	EV	16.248651
4	MQ	10.263775
5	OO	3.000000
6	UA	-7.666667
7	US	5.531383
8	WN	4.725962
9	YV	16.909968

DL is the most reliable airline carrier with **-8.00000** average arrival delay. It means the flights arrive 8 minutes early than the scheduled arrival time.

And **YV** is least reliable with **16.909968** average arrival delay. That means YV flights arrive 16 minutes after the scheduled time.



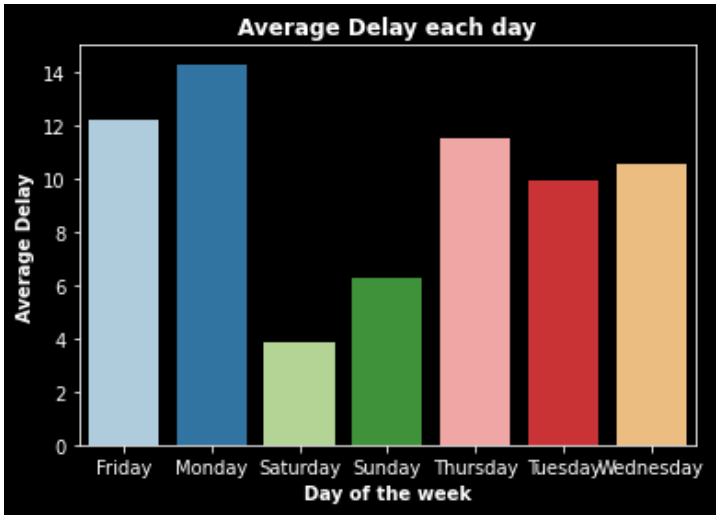
f. What day of the week we had the highest average delay?

```
temp_df1 = flights
temp_df1['Weekday'] = temp_df1['Date'].dt.strftime('%A')
```

```
week_delay = temp_df1.groupby('Weekday').arr_delay.mean().reset_index(name='Average_Delay')
week_delay
```

	Weekday	Average_Delay
0	Friday	12.212620
1	Monday	14.321970
2	Saturday	3.834410
3	Sunday	6.270552
4	Thursday	11.530760
5	Tuesday	9.940732
6	Wednesday	10.565152

Monday has the highest average delay of **14.321970**. And **Saturday** has the least average delay with **6.270552**.



g. Which one had a higher average delay: flights that took off in the morning (6 am to 10 am), noon (11 am to 2 pm), afternoon (3 pm to 5pm), or evening (6 pm to 10 pm)?

```
def f(x):
    if (x >=6) and (x <= 10):
        return 'Morning'
    elif (x >= 11) and (x <= 14 ):
        return 'Noon'
    elif (x >= 15) and (x <= 17):
        return'After Noon'
    elif (x >= 18) and (x <= 22) :
        return 'Evening'
    elif (x >= 23) and (x <= 24):
        return'Night'
temp_df1['session'] =temp_df1['hour'].apply(f)
#temp_df1
```

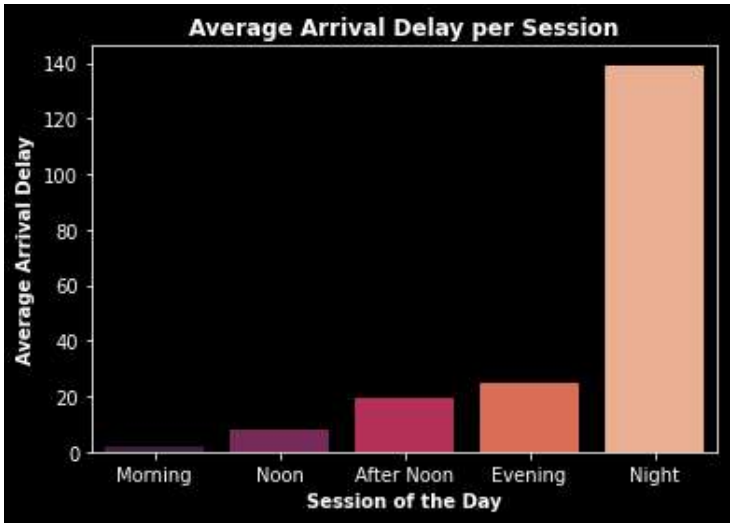
Using this code, we're assigning values based on the 'hour'. Now we can calculate the necessary answer:

```
ses_avg_gelay = temp_df1.groupby('session').dep_delay.mean().reset_index(name='Average_Arrival_Delay').sort_values(by='Average_Arrival_Delay')
ses_avg_gelay
```

	session	Average_Arrival_Delay
2	Morning	1.838044
4	Noon	8.064988
0	After Noon	19.405076
1	Evening	24.620250
3	Night	139.206897

Since we need flights that took off we need to calculate departure delay, and if we only consider morning, noon, afternoon and evening, then the highest average delay is in the **Evening** with **24.620250** minutes of delay.

And flights leaving in the **Morning** has the least average delay of **1.838044**. But if we also include **Night**, then it has the highest average delay of **139.206897** minutes.

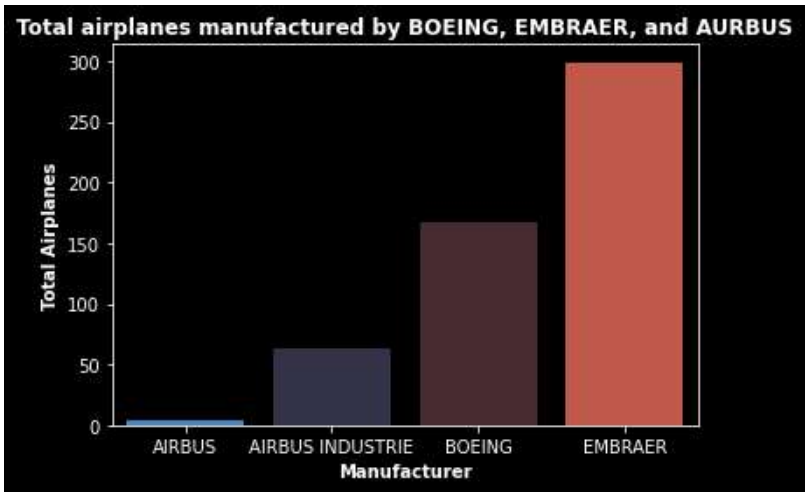


h. Determine the number of airplanes used in these flights manufactured by BOEING, EMBRAER, and AIRBUS separately.

```
manu_tot = merged_df.groupby('manufacturer').tailnum.nunique().reset_index(name='Total')
manu_tot = manu_tot[(manu_tot['manufacturer']=='BOEING')  | (manu_tot['manufacturer']=='EMBRAER') | (manu_tot['manufacturer']=='AIRBUS INDUSTRIE') | (m
manu_tot
```

	manufacturer	Total
0	AIRBUS	4
1	AIRBUS INDUSTRIE	63
2	BOEING	167
7	EMBRAER	299

1. The number of airplanes used in these flights manufactured by BOEING are: 167.
2. The number of airplanes used in these flights manufactured by EMBRAER are: 299
3. The number of airplanes used in these flights manufactured by AIRBUS are: 4
4. The number of airplanes used in these flights manufactured by AIRBUS INDUSTRIE are: 63



Question 4:

Build a linear regression model to estimate the arrival delay of the flights given in "fligts_test_data.xlsx". Note that you have the full authority to decide what columns, what datasets (among the given datasets) to work with. In your report, please explain how you build the LR model and elaborate on its accuracy.

For this question we first load all the libraries required for the Linear Regression model and visualization. We also read the dataset “flights_test_data.csv” that is the required test data.

```
In [103]: from sklearn.model_selection import train_test_split
          from sklearn.linear_model import LogisticRegression
          from sklearn.feature_extraction import DictVectorizer
          from sklearn.feature_extraction.text import TfidfTransformer
          from sklearn import preprocessing
          from sklearn.linear_model import LinearRegression
          from sklearn.metrics import confusion_matrix, accuracy_score

          test_data1 = pd.read_excel('https://raw.githubusercontent.com/SravaniRVS/DATA601/main/Project%202/flights-test-data.xlsx')

          test_data1
```

Now we would train the model. The features that we have chosen here are the best features for the “arr_delay”. After selecting X and Y we divide the data into train and test data for training the model.

```
Y=flights["arr_delay"].values.reshape(-1,1)
X=flights[["dep_delay","hour","dep_time","arr_time"]]
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=17)
linreg=LinearRegression()
linreg.fit(X_train,Y_train)
```

We calculate the accuracy of the model now.

```
print('Model Accuracy: ',linreg.score(X_test,Y_test))
```

```
Model Accuracy:  0.8418528059613568
```

If we execute the above code, we would get this output:

	year	month	day	carrier	origin	dest	distance	prediction
0	2013	1	6	MQ	JFK	DCA	213	2095.981474
1	2013	1	25	EV	LGA	IAD	229	2095.629629
2	2013	2	11	MQ	JFK	DCA	213	2097.329798
3	2013	4	14	US	LGA	DCA	214	2100.227719
4	2013	4	29	EV	LGA	IAD	229	2099.974558
5	2013	5	26	MQ	JFK	DCA	213	2101.374769
6	2013	7	14	MQ	JFK	BWI	184	2104.369979
7	2013	7	16	US	LGA	DCA	214	2104.627184
8	2013	8	1	EV	EWR	IAD	212	2106.500102
9	2013	9	29	US	LGA	DCA	214	2107.242025
10	2013	10	10	9E	JFK	IAD	228	2109.390281
11	2013	10	27	MQ	JFK	DCA	213	2108.770839
12	2013	11	13	US	LGA	DCA	214	2110.647662
13	2013	11	21	9E	LGA	IAD	229	2110.585383
14	2013	12	2	EV	EWR	IAD	212	2112.411504
15	2013	12	4	US	LGA	DCA	214	2112.377750
16	2013	12	8	EV	LGA	IAD	229	2112.424546
17	2013	12	20	EV	EWR	IAD	212	2111.920665
18	2013	12	21	US	LGA	DCA	214	2111.914179
19	2013	12	30	EV	EWR	IAD	212	2111.647977

The test data provided here contains 7 columns out of which 3 columns "carrier", "origin" and "dest" is irrelevant for the Linear regression model because they are string type. Out of the four remaining columns the "year", "month" and "day" features does not provide a remarkable contribution in the prediction because they are not unique to provide a strong prediction. Moreover, the test data does not have ample amount of data to provide proper prediction. We tried to train data with some particular features so that we can get a respectful accuracy.

Question 5:

Build a logistic regression model to guess the 3 cancelled flights given in "fligts_test_data.xlsx". Again, you have the full authority to decide what columns, what datasets (among the given datasets) to work with. In your report, please explain how you build the LogReg model and elaborate on its accuracy.

For this question we're going to use the flights dataset and merge with the ny_weather dataset. First reading the test data:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction import DictVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn import preprocessing
from sklearn.linear_model import LinearRegression
from sklearn.metrics import confusion_matrix, accuracy_score

test_data = pd.read_excel('https://raw.githubusercontent.com/SravaniRVS/DATA601/main/Project%202/flights-test-data.xlsx')

test_data
```

```
df = pd.merge(flights, ny_weather,on=['Date', 'year','month','day','hour','date_time','origin'],  how='inner')
df['Cancelled'] = df['Cancelled'].astype('int')
df
```

Now since we're at liberty use the columns we wish, we'll be deleting some columns and create a new dataset:

```
new_df = df.drop(columns=['tailnum','date_time','Precipitation','Date','time_hour', 'dewp',
  'Snow Depth' ,'Snowfall', 'dep_delay','arr_delay'])

new_df = new_df.reindex(columns=['year', 'month', 'day', 'dep_time', 'arr_time',
       'carrier', 'flight', 'origin', 'dest', 'air_time',
       'distance', 'hour', 'minute', 'temp', 'humid',
       'wind_dir', 'wind_speed', 'wind_gust', 'precip','pressure', 'visib','Max Temp','Min Temp','Cancelled'])
new_df.fillna(0,inplace=True)
```

Now let's split the data into training data and testing data to fit into the logistic regression model.

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
for column_name in new_df.columns:
      if new_df[column_name].dtype == object:
          new_df[column_name] = le.fit_transform(new_df[column_name])
      else:
          pass

X=new_df.iloc[:,:-1]
y= new_df.iloc[:,-1].values.reshape(-1,1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

Find out the model accuracy:

```
print('Model Accuracy: ',logreg.score(X_test,y_test))
```

```
Model Accuracy:  1.0
```

That's some unbelievable accuracy to achieve. Let's find out if it gives the required result.

We'll now be using SelectKBest to find out the best features that we'll be using to predict the cancelled flights.

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
print("Feature data dimension: ", X.shape)

select = SelectKBest(score_func=chi2, k=7)
z = select.fit_transform(X, y)
print("After selecting best k features:", z.shape)


filter = select.get_support()
features = np.array(X.columns.tolist())

print("\nSelected best features:")
print(features[filter])
```

```
Feature data dimension:  (17120, 23)
After selecting best k features: (17120, 7)

Selected best features:
['dep_time' 'arr_time' 'flight' 'air_time' 'hour' 'minute' 'pressure']
```

From all the features we've used in new_df we got 'dep_time', 'arr_time' 'flight' 'air_time' 'hour' 'minute' 'pressure' as the best features we can use. So now let's fit these features in our model.

```
x_filtered = X.iloc[:,filter]
X_train, X_test, y_train, y_test = train_test_split(x_filtered.values, y, test_size=0.2, random_state=1)

logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

```
print('Model Accuracy:',logreg.score(X_test,y_test))
```

```
Model Accuracy: 1.0
```

We still got the model accuracy of 1.0. Let's do the final prediction.

```
from sklearn import preprocessing

class MultiColumnLabelEncoder:

    def __init__(self, columns=None):
        self.columns = columns # array of column names to encode

    def fit(self, X, y=None):
        self.encoders = {}
        columns = X.columns if self.columns is None else self.columns
        for col in columns:
            self.encoders[col] = preprocessing.LabelEncoder().fit(X[col])
        return self

    def transform(self, X):
        output = X.copy()
        columns = X.columns if self.columns is None else self.columns
        for col in columns:
            output[col] = self.encoders[col].transform(X[col])
        return output

    def fit_transform(self, X, y=None):
        return self.fit(X,y).transform(X)

    def inverse_transform(self, X):
        output = X.copy()
        columns = X.columns if self.columns is None else self.columns
        for col in columns:
            output[col] = self.encoders[col].inverse_transform(X[col])
        return output
```

This is the code that is needed to transform out test data contains strings in 'carrier', 'dest', and 'origin' columns and after predicting it will inverse_transform to display the original string in the final result.

```
encoded = MultiColumnLabelEncoder(columns=['carrier','origin','dest'])
a = encoded.fit_transform(test_data)

a['Prediction'] = logreg.predict(a)
final_result = encoded.inverse_transform(a)
final_result
```

If we run this code, we would get the following result:

	year	month	day	carrier	origin	dest	distance	Prediction
0	2013	1	6	MQ	JFK	DCA	213	0
1	2013	1	25	EV	LGA	IAD	229	0
2	2013	2	11	MQ	JFK	DCA	213	0
3	2013	4	14	US	LGA	DCA	214	0
4	2013	4	29	EV	LGA	IAD	229	0
5	2013	5	26	MQ	JFK	DCA	213	0
6	2013	7	14	MQ	JFK	BWI	184	0
7	2013	7	16	US	LGA	DCA	214	0
8	2013	8	1	EV	EWR	IAD	212	0
9	2013	9	29	US	LGA	DCA	214	0
10	2013	10	10	9E	JFK	IAD	228	1
11	2013	10	27	MQ	JFK	DCA	213	0
12	2013	11	13	US	LGA	DCA	214	0
13	2013	11	21	9E	LGA	IAD	229	1
14	2013	12	2	EV	EWR	IAD	212	0
15	2013	12	4	US	LGA	DCA	214	0
16	2013	12	8	EV	LGA	IAD	229	0
17	2013	12	20	EV	EWR	IAD	212	0
18	2013	12	21	US	LGA	DCA	214	0
19	2013	12	30	EV	EWR	IAD	212	0

We can see that the flight from JFK to IAD carried by 9E on the date 10/10/2013 is predicted to be cancelled based on our model. And the flight from LGA to IAD carried by 9E on the date 11/21/2013 is also predicted to get cancelled by our logistic regression model. Even though we need to predict 3, we could only predict 2 cancelled flights. Maybe because the test data we have does not have ample amount of data to provide prediction. Or it could just simply mean that there is some kind

of an error in our model. But after learning everything we could about the logistic regression techniques this is the best prediction that we could provide.