Project Documentation

First will start by importing the libraries that we need Pandas, Numpy, matplotlib & seaborn

```
import pandas as pd

import numpy as np

from pandas import DataFrame, Series

import matplotlib.pyplot as plt

from pylab import rcParams

import seaborn as sns

from sklearn.datasets import load_iris

from sklearn.neighbors import KNeighborsClassifier

from sklearn import metrics

from datetime import date

import warnings

%matplotlib inline

warnings.filterwarnings('ignore')
```

Then we need to read our data

```
df = pd.read_csv("C:\\Users\\mrnaz\\Desktop\\BootCamp\\My_Project\\WN-JAN-2020.csv")
```

let's have a look at what just we read

```
df.shape

df.head(5)
```

First, I will rename some columns

```
df.rename({'MKT_CARRIER_FL_NUM':'FL_NUM',

 'ORIGIN_CITY_NAME':'ORIGIN_CITY',

 'ORIGIN_CITY_NAME':'ORIGIN_CITY',

 'ORIGIN_STATE_NM':'ORIGIN_STATE',

 'DEST_CITY_NAME':'DEST_CITY',

'DAY_OF_MONTH':'MONTH_DAY',

 'DAY_OF_WEEK':'WEEK_DAY',

 'DEST_STATE_NM':'DEST_STATE'},
```

```
axis=1, inplace=True)
```

Second, I will remove the Time in Hourly Block and the 3 letters city code it is not familiar for most people that are not working in the aviation let's just remove it and save some space

```
df.drop(['ORIGIN','DEST','DEP_TIME_BLK','ARR_TIME_BLK'],axis=1,inplace=True)
```

we decided that we will not include canceled flights in our study

so we will remove the row that have tha value of 1 in the column CANCELLED then we will remove the columns CANCELLED and CANCELLATION_CODE

```
df = df[df.CANCELLED == 0]

df.drop(['CANCELLED','CANCELLATION_CODE'],axis=1,inplace=True)
```

let's have a look on the table after we do these changes

```
df.head(100)# or df.sample(5)
```

O.K we need to know more about our table such as number of non-null and data type for each column

```
df.corr()  # View the correlations

df.info()
```

Let's convert the column FL_DATE from object to Date

```
df["FL_DATE"] = pd.to_datetime(df["FL_DATE"])
```

how still have nulls?

```
df.isna().any()
```

we need to know number of NaN in each column

```
df.isna().sum()  # or df.isnull().sum()
```

In the columns CARRIER_DELAY, WEATHER_DELAY, NAS_DELAY, SECURITY_DELAY and LATE_AIRCRAFT_DELAY

The null means that there is no delay we will just replace the nulls with 0

```
#replac null with 0

df['CARRIER_DELAY'] = df['CARRIER_DELAY'].fillna(0)

df['WEATHER_DELAY'] = df['WEATHER_DELAY'].fillna(0)

df['NAS_DELAY'] = df['NAS_DELAY'].fillna(0)

df['SECURITY_DELAY'] = df['SECURITY_DELAY'].fillna(0)
```

df['LATE_AIRCRAFT_DELAY'] = df['LATE_AIRCRAFT_DELAY'].fillna(0)

We found nulls in columns WHEELS_ON, TAXI_IN, ARR_TIME, ACTUAL_ELAPSED_TIME and AIR_TIME

But this column should not have nulls so we will consider it un complete data and we will exclude it from the study

```
#remove raw with null

df = df[df.WHEELS_ON.notnull()]

df = df[df.TAXI_IN.notnull()]

df = df[df.ARR_TIME.notnull()]

df = df[df.ACTUAL_ELAPSED_TIME.notnull()]

df = df[df.AIR_TIME.notnull()]
```

more descriptions

```
df.describe()
```

To examine data types and look for cases of missing or potentially wrong data, the describe function helps to get an overview of the data from a statistical summary point of view. This is also useful for spotting any potential errors that may need a closer look.

```
df.shape

df.size

df.columns

df.dtypes # This will peovied data type for each column
```

I need to list all origin city name

```
Print(df.ORIGIN_CITY.unique())
```

To check if there is duplicate rows in the dataset

```
sum(df.duplicated())
```

This command tells us how many flights in each airport, it will basically calculate city name repeat number

```
print (df['ORIGIN_CITY'].value_counts())
```

We need to how many flights for each plain

```
print (df.TAIL_NUM.value_counts())
```

This will display all flight number 795

```
df[df['FL_NUM'] == 795]
```

The average of the departial delay

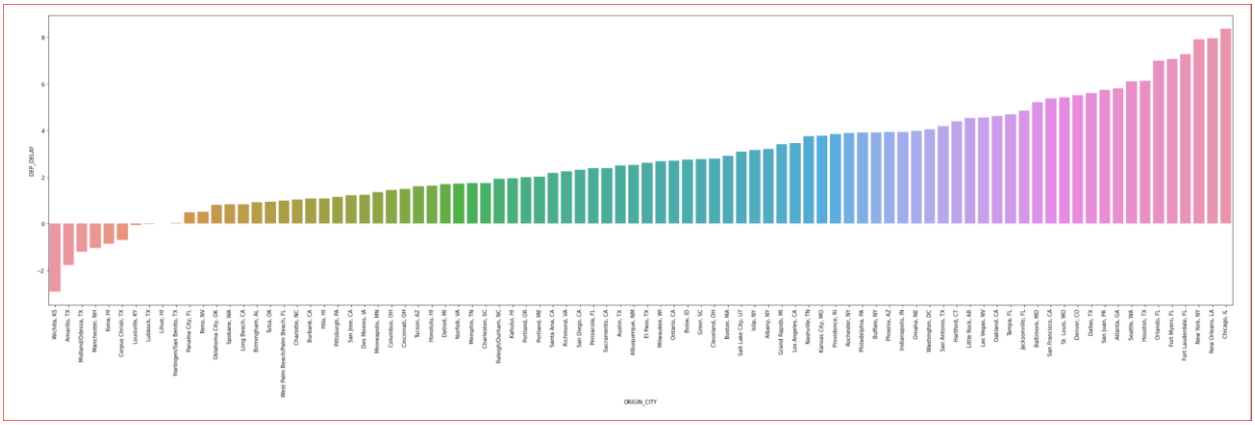
```
av_DEP_DELAY = df.DEP_DELAY.mean()
```

```
print (av_DEP_DELAY)
```

let's Display some Graphs

The below figure shows the delay departure time for each city, the minus number means early departure

```
plt.figure(figsize =(40,10))

plt.xticks(rotation = 90)

temp = df[['DEP_DELAY', 'ORIGIN_CITY']]

temp=pd.DataFrame(temp.groupby('ORIGIN_CITY')['DEP_DELAY'].agg('mean')).reset_index()
.sort_values(by= ['DEP_DELAY'])

temp

sns.barplot(x='ORIGIN_CITY', y="DEP_DELAY", data = temp)
```



Here we can get the same information in a table

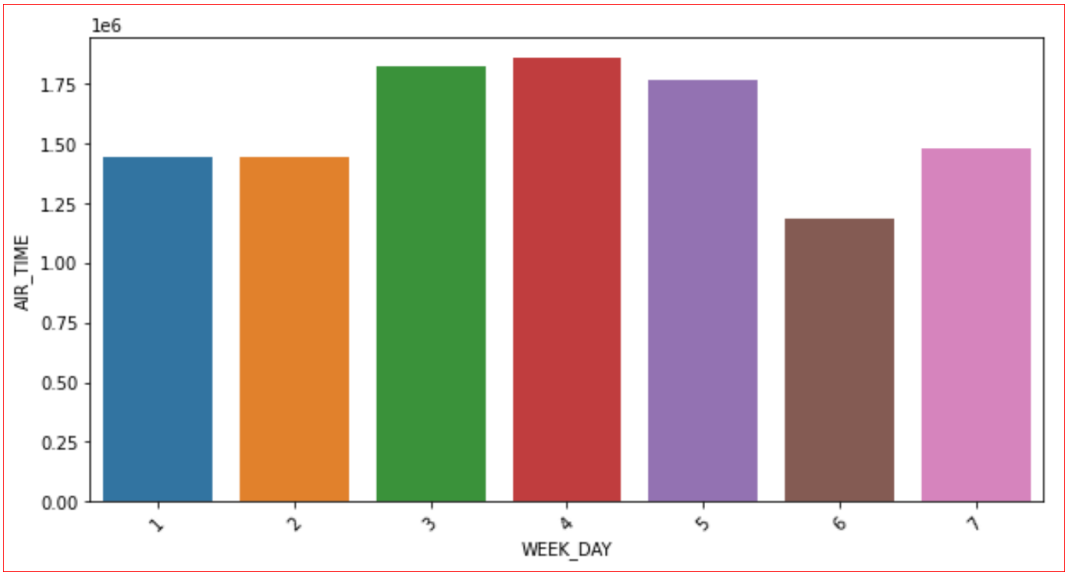
```
t1=temp

t1
```

	ORIGIN_CITY	DEP_DELAY
87	Wichita, KS	-2.923729
2	Amarillo, TX	-1.783217
47	Midland/Odessa, TX	-1.205036
45	Manchester, NH	-1.042802
37	Kona, HI	-0.872483
...	...	...
24	Fort Myers, FL	7.084225
23	Fort Lauderdale, FL	7.290528
52	New York, NY	7.927954
51	New Orleans, LA	7.960164
13	Chicago, IL	8.377028

88 rows × 2 columns

The below figure shows the number of flights time in each day of the week (Air Traffic)

```
plt.figure(figsize =(10,5))

plt.xticks(rotation = 45)

temp = df[['AIR_TIME', 'WEEK_DAY']]

temp=pd.DataFrame(temp.groupby('WEEK_DAY')['AIR_TIME'].agg('sum')).reset_index().sort
_values(by= ['AIR_TIME'])

temp

sns.barplot(x='WEEK_DAY', y="AIR_TIME", data = temp)
```



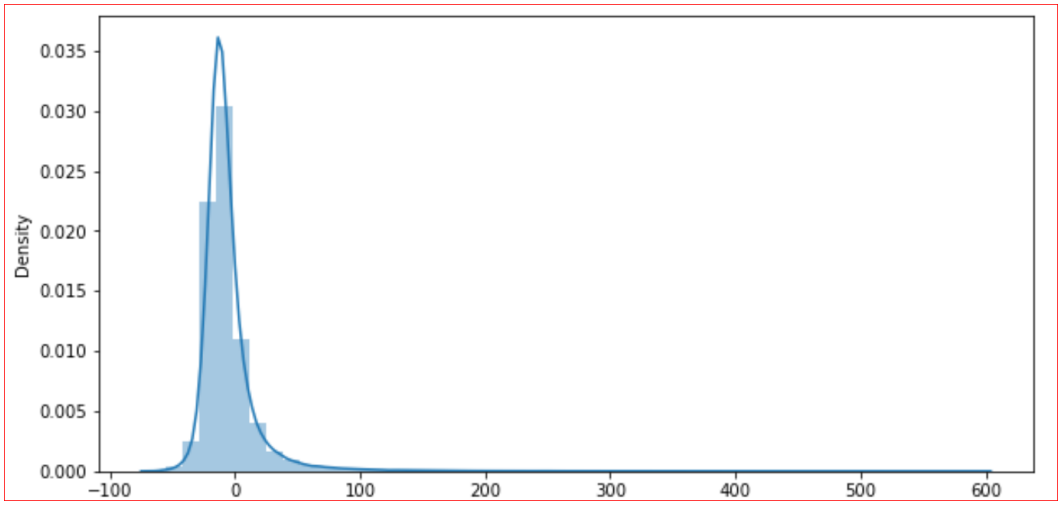
Wednesday has the highest air traffic during the week

```
t2=temp

t2
```

	WEEK_DAY	AIR_TIME
5	6	1182881.0
1	2	1440845.0
0	1	1443243.0
6	7	1478819.0
4	5	1768817.0
2	3	1822548.0
3	4	1855419.0

The rate of difference between the scheduled access time and the actual time

```
plt.figure(figsize = (10,5))

sns.distplot(x=df['ARR_DELAY']);
```



The figure below gives us an idea of the relationship between flight delays due to civil aviation and the distance between airports

```
plt.figure(figsize =(10,10))

temp = df[['NAS_DELAY', 'DISTANCE']]

temp=pd.DataFrame(temp.groupby('DISTANCE')['NAS_DELAY'].agg('max')).reset_index().sort
_values(by= ['NAS_DELAY'])

temp

sns.kdeplot(x='DISTANCE', y="NAS_DELAY", data = temp)
```