Data Visualization

Data Analysis and Interactive Visualization

INTRODUCTION

Exploratory Data Analysis (EDA), also known as Data Exploration, is a step in the Data Analysis Process, where a number of techniques are used to better understand the dataset being used. 'Understanding the dataset' can refer to a number of things including but not limited to Extracting important variables and leaving behind useless variables Identifying outliers, missing values, or human error. Understanding the relationship(s), or lack of, between variables

Ultimately, maximizing your insights of a dataset and minimizing potential error that may occur later in the process.

- **Loading and Summary of the Dataset**

```
df=pd.read_csv('Flights dataset.csv')
```

```
[ ] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 182533 entries, 0 to 182532
Data columns (total 35 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   YEAR                 182533 non-null  int64
 1   MONTH                182533 non-null  int64
 2   DAY_OF_MONTH         182533 non-null  int64
 3   DAY_OF_WEEK          182533 non-null  int64
 4   UNIQUE_CARRIER       182533 non-null  object
 5   AIRLINE_ID           182533 non-null  int64
 6   CARRIER              182533 non-null  object
 7   TAIL_NUM             181618 non-null  object
 8   FL_NUM               182532 non-null  float64
 9   ORIGIN               182532 non-null  object
 10  ORIGIN_CITY_NAME     182532 non-null  object
 11  ORIGIN_STATE_ABR     182532 non-null  object
 12  ORIGIN_STATE_NM      182532 non-null  object
 13  ORIGIN_WAC           182532 non-null  float64
 14  DEST                 182532 non-null  object
 15  DEST_CITY_NAME       182532 non-null  object
 16  DEST_STATE_ABR       182532 non-null  object
 17  DEST_STATE_NM        182532 non-null  object
 18  DEST_WAC             182532 non-null  float64
 19  CRS_DEP_TIME         182532 non-null  float64
```

```
[ ]  df.describe()
```

	YEAR	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	AIRLINE_ID	FL_NUM	ORIGIN_WAC
count	182533.0	182533.0	182533.000000	182533.000000	182533.000000	182532.000000	182532.000000
mean	2009.0	12.0	15.926205	3.847545	20016.251067	1902.455175	49.966669
std	0.0	0.0	8.956029	1.946194	290.027110	1666.628891	25.447615
min	2009.0	12.0	1.000000	1.000000	19704.000000	1.000000	1.000000
25%	2009.0	12.0	8.000000	2.000000	19790.000000	623.000000	33.000000
50%	2009.0	12.0	16.000000	4.000000	19805.000000	1401.000000	41.000000
75%	2009.0	12.0	24.000000	5.000000	20366.000000	2350.000000	74.000000
max	2009.0	12.0	31.000000	7.000000	20437.000000	5710.000000	93.000000

8 rows × 23 columns

```
[ ]  df.isnull().sum()
```

```
YEAR                    0
MONTH                   0
DAY_OF_MONTH            0
DAY_OF_WEEK             0
UNIQUE_CARRIER          0
AIRLINE_ID              0
CARRIER                 0
TAIL_NUM              915
FL_NUM                  1
ORIGIN                  1
ORIGIN_CITY_NAME        1
ORIGIN_STATE_ABR        1
ORIGIN_STATE_NM         1
ORIGIN_WAC              1
DEST                    1
DEST_CITY_NAME          1
DEST_STATE_ABR          1
DEST_STATE_NM           1
DEST_WAC                1
CRS_DEP_TIME            1
DEP_TIME             3882
DEP_DELAY            3882
CRS_ARR_TIME            1
ARR_TIME             4134
ARR_DELAY            4504
```

- **Data Preparation and Wrangling**

This step is performed after the data gathering procedure. Since the dataset was already provided, we did not have to do much in terms of gathering. Once the data is collected, the data preparation and wrangling stage begins. This stage involves two important tasks: cleansing and preprocessing, respectively.

1. Data Preparation/Cleaning/Preprocessing

This is the initial and most common task in data preparation that is performed on raw data. Data cleansing is the process of examining, identifying, and mitigating errors in raw data. Normally, the raw data are neither sufficiently complete nor sufficiently clean to directly train the ML model. Manually entered data can have incomplete, duplicated, erroneous, or inaccurate values.

2. Data Wrangling

This task performs transformations and critical processing steps on the cleansed data to make the data ready for ML model training. Raw data most commonly are not present in the appropriate format for model consumption. After the cleansing step, data need to be processed by dealing with outliers, extracting useful variables from existing data points, and scaling the data.

List of Features & Their Unique Values

```
[ ]  df.nunique()
```

```
YEAR                    1
MONTH                   1
DAY_OF_MONTH           31
DAY_OF_WEEK             7
UNIQUE_CARRIER          9
AIRLINE_ID              9
CARRIER                 9
TAIL_NUM             2106
FL_NUM               3357
ORIGIN                213
ORIGIN_CITY_NAME      209
ORIGIN_STATE_ABR       52
ORIGIN_STATE_NM        52
ORIGIN_WAC             52
DEST                  213
DEST_CITY_NAME        209
DEST_STATE_ABR         52
DEST_STATE_NM          52
DEST_WAC               52
CRS_DEP_TIME          945
DEP_TIME             1376
```

1- **Remove 6 last variable since they were really sparse.**

```
[ ]  #drop the last 6 variable since they are really sparse

[ ]  df1=df.drop(['CANCELLATION_CODE','CARRIER_DELAY','WEATHER_DELAY','NAS_DELAY','SECURITY_DELAY','LATE_AIRCRAFT_DELAY'],axis=1)
```

2- Filling missing values by mean

fill missing value by mean

```
[ ]  for i in df1.columns:
       if df1[i].dtypes=='float':
         df1[i]=df1[i].fillna(df1[i].mean())
```

```
[ ]  df1.select_dtypes(include='float').isnull().sum()
```

```
FL_NUM          0
ORIGIN_WAC      0
DEST_WAC        0
CRS_DEP_TIME    0
DEP_TIME        0
DEP_DELAY       0
CRS_ARR_TIME    0
ARR_TIME        0
ARR_DELAY       0
CANCELLED       0
DIVERTED        0
AIR_TIME        0
DISTANCE        0
dtype: int64
```

.

3- data Normalization

normalize with MinMaxScaler

```
[ ]  sub_df=df1.select_dtypes(include=['float','int'])
     from sklearn.preprocessing import StandardScaler
     from sklearn.preprocessing import MinMaxScaler

     ss = MinMaxScaler()
     scaled_df = ss.fit_transform(sub_df)
     scaled_df = pd.DataFrame(scaled_df, columns=sub_df.columns)
     scaled_df = pd.concat([scaled_df, df1['CARRIER']], axis=1)
     scaled_df.head()
```

	YEAR	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	AIRLINE_ID	FL_NUM	ORIGIN_WAC	DEST_WAC	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	CRS_ARR_TIME	ARR_TIME	ARR_DELAY	CANCELLED
0	0.0	0.0	0.033333	0.333333	0.899045	0.148713	0.358696	0.380435	0.742869	0.751146	0.050075	0.817642	0.809504	0.077372	0.0
1	0.0	0.0	0.066667	0.500000	0.899045	0.148713	0.358696	0.380435	0.742869	0.729887	0.041854	0.817642	0.801584	0.063504	0.0
2	0.0	0.0	0.100000	0.666667	0.899045	0.148713	0.358696	0.380435	0.742869	0.730721	0.043348	0.817642	0.799917	0.060584	0.0
3	0.0	0.0	0.166667	1.000000	0.899045	0.148713	0.358696	0.380435	0.742869	0.729054	0.040359	0.817642	0.798666	0.058394	0.0
4	0.0	0.0	0.200000	0.000000	0.899045	0.148713	0.358696	0.380435	0.742869	0.729887	0.041854	0.817642	0.804085	0.067883	0.0

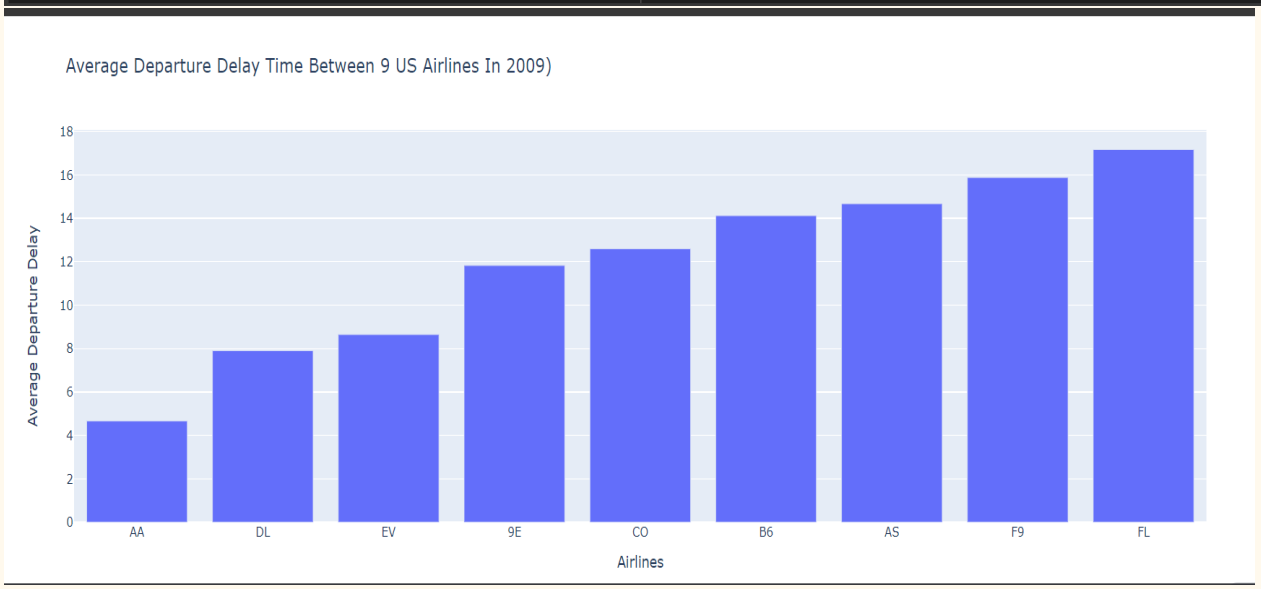
- Task Answers

which airline dose the most departure delay? comparing Average departure delay time between US airlines

qestion2: which airline dose the most departure delay?comparing Average departure delay time between US airlines

```
[ ] DELAY=df1.groupby('CARRIER').agg({'DEP_DELAY':np.mean}).sort_values(by='DEP_DELAY')
    Airlines=list(df1['CARRIER'].value_counts().index)
    DELAY1=list(DELAY['DEP_DELAY'])

    fig = px.bar(x=Airlines, y=DELAY1,
                 labels=dict(x="Airlines", y="Average Departure Delay"), title='Average Departure Delay Time Between 9 US Airlines In 2009')
    fig.show()
```



In this chart, by displaying the average delay time of departure flights among American Airlines, we have identified which airlines had the most delays.

Based on the data displayed on the chart, we can conclude that FL airline had the highest delay rate among all US Airlines. In contrast, AA airline had the minimum delay time in departure flights in 2009.

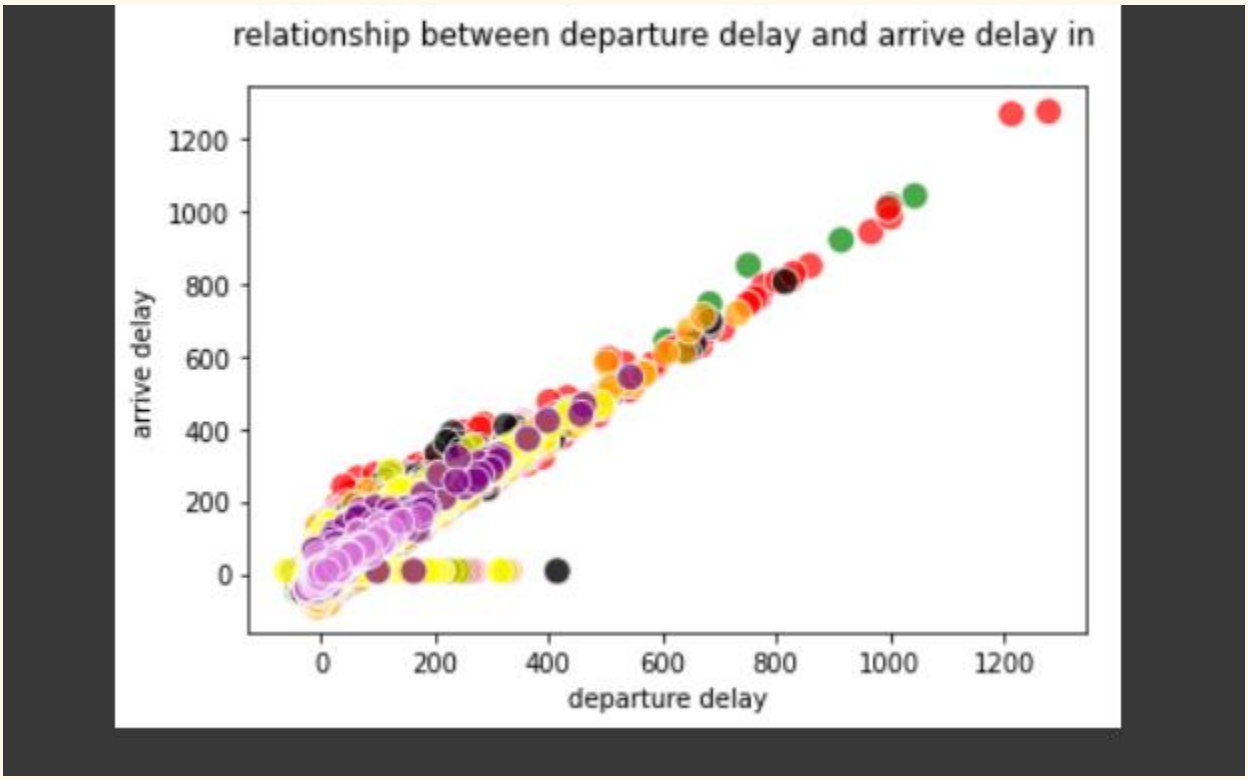
Is there any relationship between Departure Delay and Arrive Delay in different airline?

qestion3: Are there any relationship between Departure Delay and Arrive Delay in different airline?

```
[ ]  colors = ['red' if i=='AA' else 'yellow' if i =='EV' else 'green'
               if i == '9E' else 'black' if i=='CO' else 'pink' if
               i=='B6' else 'grey' if i=='AS'  else 'purple'
               if i=='F9' else 'orange' if i=='DL' else 'orchid' for i in list(df1['CARRIER'])]

     plt.scatter(df1['DEP_DELAY'], df1['ARR_DELAY'], c=colors,s=120,
                 alpha=0.7, edgecolors='w')

     plt.xlabel('departure delay')
     plt.ylabel('arrive delay')
     plt.title('relationship between departure delay and arrive delay ',y=1.05)
```



Answer: as shown by the scatterplot, the relationship between arrival and departure flight delays is determined by airline.

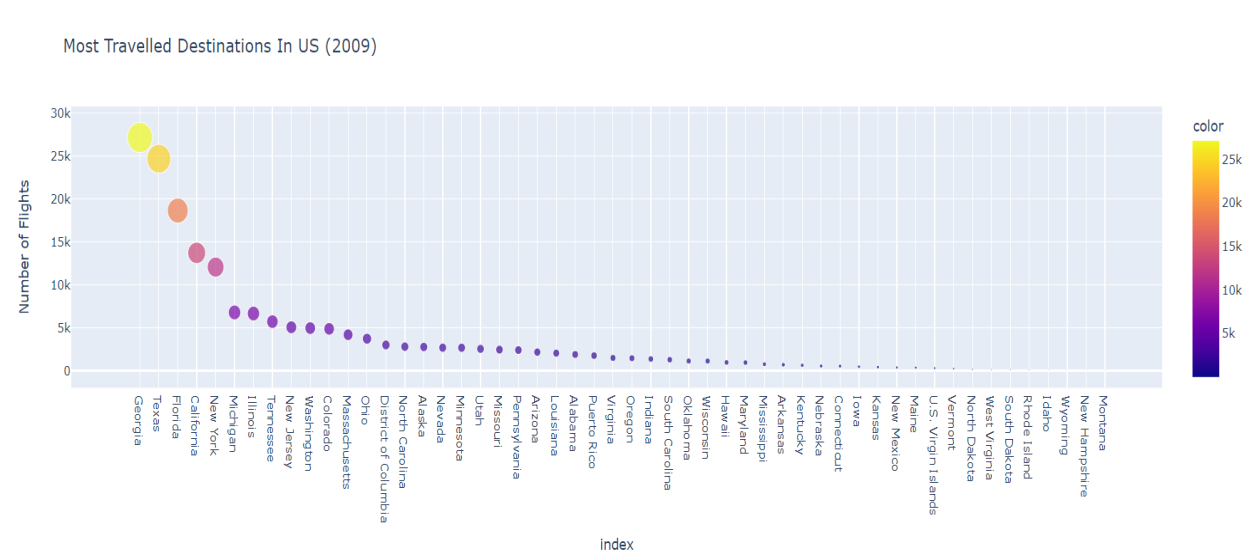
Based on the data displayed on the plot, it can be concluded that the amount of delay in departure flights is directly related to the amount of delay in arrival flights. Therefore, it can be said that they have direct relationship.

compare destinations travelled of US airlines in 2009.

question4:compare destinations travelled of US airlines in 2009.

```
scat = df1['DEST_STATE_NM'].value_counts()

fig = px.scatter(scat, x=scat.index, y=scat, color=scat,
                 size=scat, labels=dict(x="Destinations", y="Number of Flights"), title='Most Travelled Destinations In US (2009)')
fig.show()
```



As we can see the most popular destination belonged to the Georgia and Texas which around 27k and 24k flights in the year of 2009, respectively. In addition, there was dramatically reduction in the number of flights to around 18k for Florida. The number of flights has been reduced to got 11 which was belonged to the Montana.

compare weekly average delay in Departure and Arrival in 2009

question5:compare weekly average delay in Departure and Arrival in 2009.

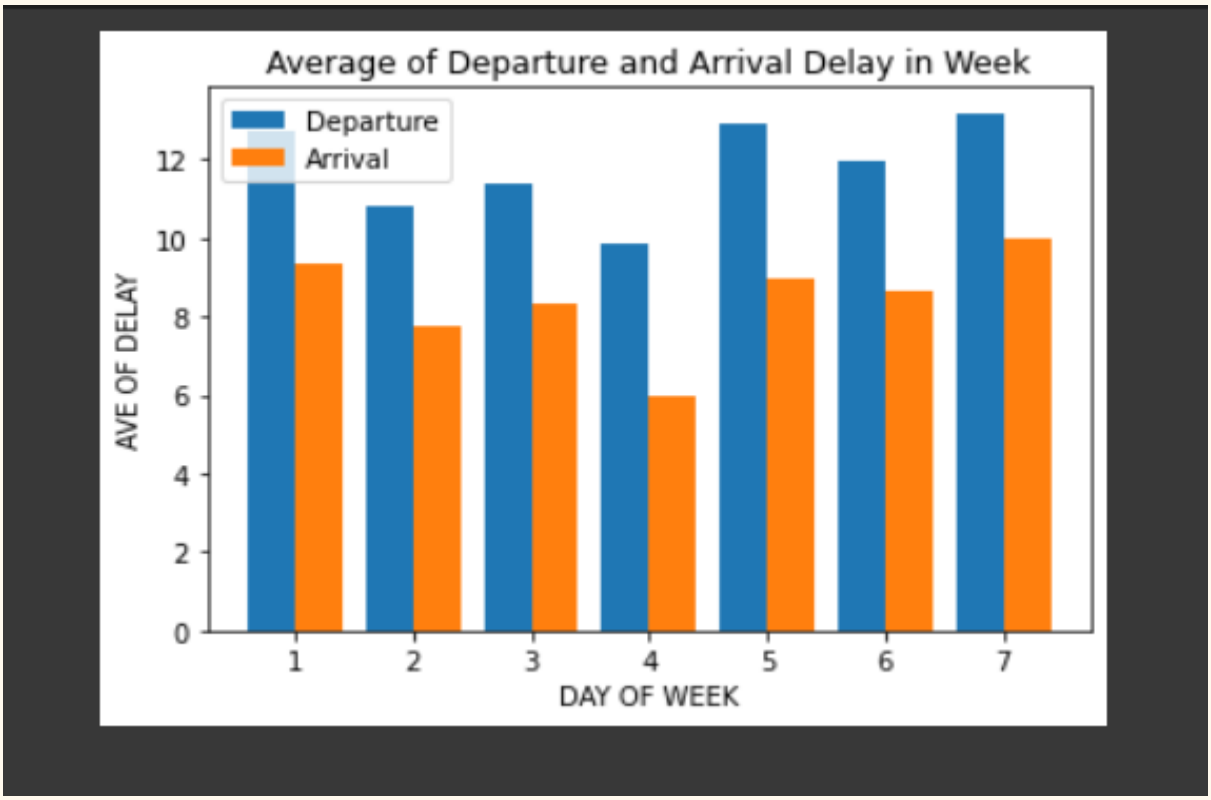
```
[ ]
      # Design a chart for weekly average delay in Departure and Arrival.


    X_axis = np.arange(1,8)

    DEP_DELAY = df1.groupby(['DAY_OF_WEEK'])['DEP_DELAY'].mean()
    ARR_DELAY = df1.groupby(['DAY_OF_WEEK'])['ARR_DELAY'].mean()

    plt.bar(X_axis - 0.2, DEP_DELAY, 0.4, label = 'Departure')
    plt.bar(X_axis + 0.2, ARR_DELAY, 0.4, label = 'Arrival')

    plt.xticks(X_axis, X_axis )
    plt.xlabel("DAY OF WEEK")
    plt.ylabel("AVE OF DELAY")
    plt.title("Average of Departure and Arrival Delay in Week")
    plt.legend()
    plt.show()
```



On all days of the week, delay on departure was higher than arrival. The highest time of delay is around 14 which is belonged to the first and last day of the week for departure flights while the lowest time is around 6 which is belonged to the 4th day of the week for arrival flights.

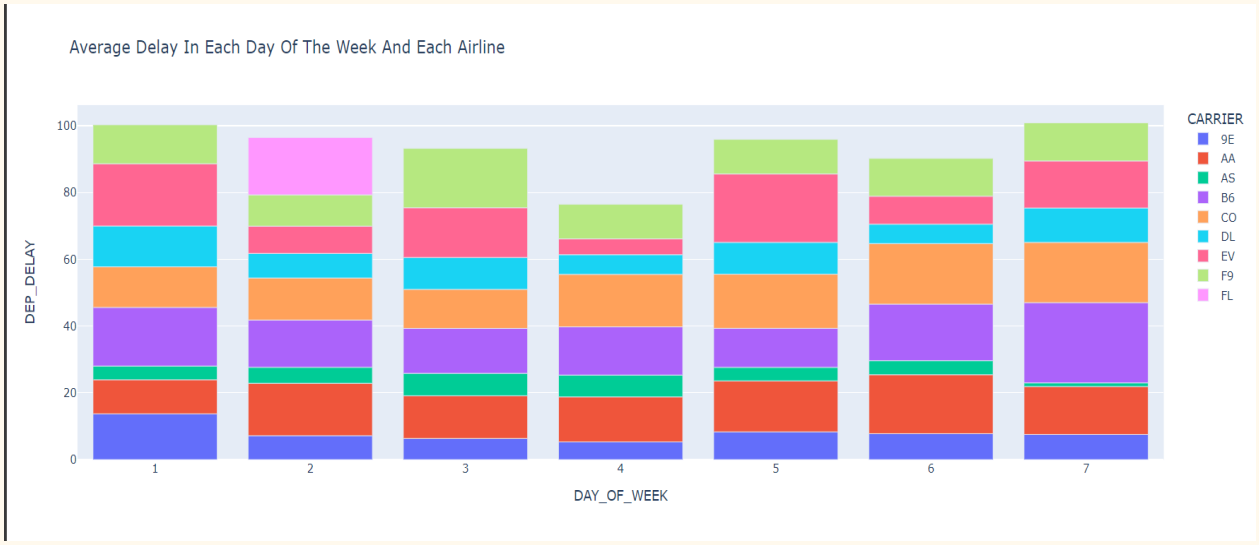
Which day of week and which airline have the high Delay?

question6:which Day Of Week And wich Airline have the high Delay?

```
WEEK = list(np.arange(1,8))

DELAY = df1.groupby(['DAY_OF_WEEK','CARRIER']).agg({'DEP_DELAY':np.mean})
DELAY=DELAY.reset_index()

fig = px.bar(DELAY,x='DAY_OF_WEEK', y='DEP_DELAY' ,color='CARRIER', labels=dict(x="DAY OF WEEK", y="Number of Flights"), title="Average Delay In Each Day Of The Week And Each Airline'
fig.show()
```



As can be seen the most delay belong to the first and last day of the week. Although the lowest time of delay belong to the 4th day of the week. As can be seen in the plot, on the 1st and 5th day of the week, EV airline had the highest amount of delay. Also, the CO airline had the most delays on the 4th and 6th days of the week. In addition, on the second and third days

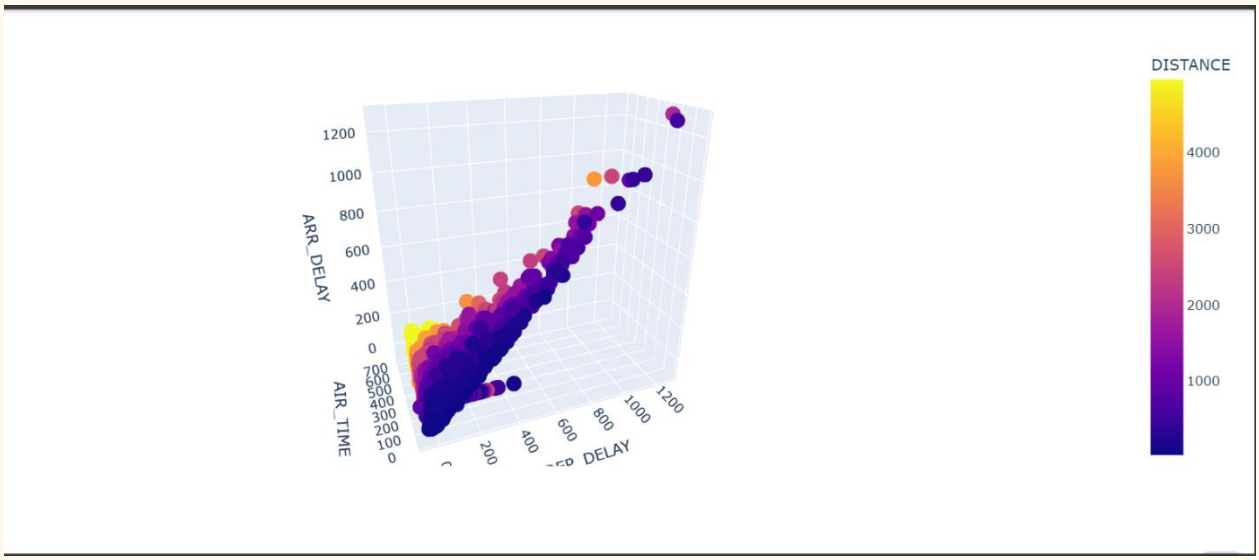
of the week, FI and F9 airlines had the highest delay, respectively. FI airline was delayed only on the second day of the week. Finally, on the 7th day, B6 airline had the highest amount of delay, which is the highest amount of delay in the week.

Investigate relationship between departure delay, arrive delay, airtime, and destination?

question7:investigat relationship between departure delay, arrive delay, air time and destination?

```
data = px.scatter_3d(df1, x='DEP_DELAY', y='AIR_TIME', z='ARR_DELAY',
                     color='DISTANCE')

layout1 = go.Layout(
    margin=dict(
        l=0,
        r=0,
        b=0,
        t=0
    )
)
fig = go.Figure(data=data, layout=layout1)
fig.show()
```



As you can see, the amount of delay in departure flights is directly related to the amount of delay in arrival flights. This means that how much the departure flight is delayed, the arrival

flight will also be delayed, which can be seen from the linearity of the scatter plot. Also, the amount of distance between the origin and destination has an inverse relationship with the amount of delay. Which means, the shorter the distance between the origin and destination, the greater the delay in the arrival and departure flight, and a longer distance, the lower the delay.

—---------- END ----------