Install Spark

↳ 3 cells hidden

Import SparkSession and SparkSQL

[ ] ↳ 1 cell hidden

Mount Google drive, we will be using GDrive for processing

Files will be present in "/content/drive/My Drive".

[ ] ↳ 1 cell hidden

Set-up helper functions

[ ] ↳ 3 cells hidden

Big Data Platforms Assignment 6

Kelsey Liu

You have two CSV available in Google Cloud Storage, representing the air travel information for 2007 and 2008. The header column for each file is fairly self-explanatory:

https://storage.googleapis.com/msca-bdp-data-open/airlines/2007.csv
https://storage.googleapis.com/msca-bdp-data-open/airlines/2008.csv

Your goal is to determine:

1. Which locations (Origin and Dest pairs) had the worst delays for both arrivals (ArrDelay) and departures (DepDelay) - for each year
2. Which locations had fewest delays.
3. Do you see any significant seasonality effects for delays?
4. Do you see any increase or decrease in delays on weekends?
5. Are flights equally distributed throughout the day?
6. Plot the distribution of DepTime, ArrTime (actual departure and arrival time)
7. Do you see the worst delays at any certain times of the day? Compare DepTime, ArrTime with CRSDepTime, CRSArrTime (scheduled arrival and departure time; CRS is the Computer Reservation System)

Your final output should look like a chart (i.e. bar chart, line chart, etc.), the chart can also be supplemented by a table as needed.

Rules and requirements:

1. You cannot download these files (or file samples) to your local computer, all processing and data exploration must take place exclusively on Colab
2. If you need to look at the file layouts, you must do so using Jupyter Notebook – no file downloads!
3. All data processing must take place in Spark, however you can use any Spark modules (i.e. PySpark, SparkDF, etc.)
4. You must build your charts directly in Jupyter Notebook
5. The assignment must be submitted as Jupyter Notebook (as ipynb file)

Pull source data from GCS into Colab

```
def get_gcs_data (bucket_name, folder_name, file_name, path_gdrive):
    url = 'https://storage.googleapis.com/' + bucket_name + '/' + folder_name + '/'
    r = requests.get(url)
    open(path_gdrive + '/' + file_name , 'wb').write(r.content)


bucket_name = 'msca-bdp-data-open'
folder_name = 'airlines'
file_name = ['2007.csv', '2008.csv']
path_gdrive = '/content/drive/My Drive/Colab Datasets/BDP/airlines'

os.makedirs(path_gdrive, exist_ok=True)

for file in file_name:
    get_gcs_data (bucket_name = bucket_name,
                  folder_name = folder_name,
                  file_name = file,
                  path_gdrive = path_gdrive)
    print('Downloaded: ' + file)
```

```
Downloaded: 2007.csv
Downloaded: 2008.csv
```

```
list_files(path_gdrive)
```

```
2007.csv  --> 702878193
2008.csv  --> 689413344
```

Import Airline Data as Spark DataFrame

```
# Enable repl.eagerEval
# This will output the results of DataFrames in each step without the need to use d
spark.conf.set("spark.sql.repl.eagerEval.enabled",True)
```

```
airlines_df = spark.read.csv(path_gdrive, header='true', inferSchema='true', sep=',
airlines_df.limit(5)
```

Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarrie
2007	1	1	1	1232	1225	1341	1340	WN
2007	1	1	1	1918	1905	2043	2035	WN
2007	1	1	1	2206	2130	2334	2300	WN
2007	1	1	1	1230	1200	1356	1330	WN
2007	1	1	1	831	830	957	1000	WN

```
airlines_df.printSchema()
```

```
root
 |-- Year: integer (nullable = true)
 |-- Month: integer (nullable = true)
 |-- DayofMonth: integer (nullable = true)
 |-- DayOfWeek: integer (nullable = true)
 |-- DepTime: string (nullable = true)
 |-- CRSDepTime: integer (nullable = true)
 |-- ArrTime: string (nullable = true)
 |-- CRSArrTime: integer (nullable = true)
 |-- UniqueCarrier: string (nullable = true)
 |-- FlightNum: integer (nullable = true)
 |-- TailNum: string (nullable = true)
 |-- ActualElapsedTime: string (nullable = true)
 |-- CRSElapsedTime: string (nullable = true)
 |-- AirTime: string (nullable = true)
 |-- ArrDelay: string (nullable = true)
 |-- DepDelay: string (nullable = true)
 |-- Origin: string (nullable = true)
 |-- Dest: string (nullable = true)
 |-- Distance: integer (nullable = true)
 |-- TaxiIn: string (nullable = true)
 |-- TaxiOut: string (nullable = true)
 |-- Cancelled: integer (nullable = true)
 |-- CancellationCode: string (nullable = true)
 |-- Diverted: integer (nullable = true)
 |-- CarrierDelay: string (nullable = true)
 |-- WeatherDelay: string (nullable = true)
 |-- NASDelay: string (nullable = true)
 |-- SecurityDelay: string (nullable = true)
 |-- LateAircraftDelay: string (nullable = true)
```

```
airlines_df = airlines_df.\
withColumn("DepTime", airlines_df["DepTime"].cast(IntegerType())).\
withColumn("ArrTime", airlines_df["ArrTime"].cast(IntegerType())).\
withColumn("ArrDelay", airlines_df["ArrDelay"].cast(IntegerType())).\
withColumn("DepDelay", airlines_df["DepDelay"].cast(IntegerType()))
```

```
columns = ["Year", "Month", "DayOfWeek", "Origin", "Dest", "DepTime", "CRSDepTime",

airlines_df_feat = airlines_df.select(columns)
airlines_df_feat.cache()
```

Year	Month	DayOfWeek	Origin	Dest	DepTime	CRSDepTime	ArrTime	CRSArrTime	ArrDelay	Dep
2007	1	1	SMF	ONT	1232	1225	1341	1340	1	7
2007	1	1	SMF	PDX	1918	1905	2043	2035	8	13
2007	1	1	SMF	PDX	2206	2130	2334	2300	34	36
2007	1	1	SMF	PDX	1230	1200	1356	1330	26	30
2007	1	1	SMF	PDX	831	830	957	1000	-3	1
2007	1	1	SMF	PDX	1430	1420	1553	1550	3	10
2007	1	1	SMF	PHX	1936	1840	2217	2130	47	56
2007	1	1	SMF	PHX	944	935	1223	1225	-2	9
2007	1	1	SMF	PHX	1537	1450	1819	1735	44	47
2007	1	1	SMF	PHX	1318	1315	1603	1610	-7	3
2007	1	1	SMF	PHX	836	835	1119	1130	-11	1
2007	1	1	SMF	PHX	2047	1955	2332	2240	52	52
2007	1	1	SMF	SAN	2128	2035	2245	2200	45	53
2007	1	1	SMF	SAN	935	940	1048	1105	-17	-5
2007	1	1	SMF	SAN	1251	1245	1405	1410	-5	6
2007	1	1	SMF	SAN	1729	1645	1843	1810	33	44
2007	1	1	SMF	SAN	825	825	941	950	-9	0
2007	1	1	SMF	SAN	1042	1040	1158	1205	-7	2
2007	1	1	SMF	SAN	1726	1725	1839	1850	-11	1
2007	1	1	SMF	SAN	1849	1820	2016	1940	36	29

only showing top 20 rows

```
airlines_df_feat.printSchema()
```

```
root
 |-- Year: integer (nullable = true)
 |-- Month: integer (nullable = true)
 |-- DayOfWeek: integer (nullable = true)
 |-- Origin: string (nullable = true)
 |-- Dest: string (nullable = true)
 |-- DepTime: integer (nullable = true)
 |-- CRSDepTime: integer (nullable = true)
 |-- ArrTime: integer (nullable = true)
 |-- CRSArrTime: integer (nullable = true)
 |-- ArrDelay: integer (nullable = true)
 |-- DepDelay: integer (nullable = true)
```

```
airlines_df_feat.createOrReplaceTempView("airlines")
```

Question 1.

Which locations (Origin and Dest pairs) had the worst delays for both arrivals (ArrDelay) and departures (DepDelay) - for each year

Assumptions made:

1. Here I define 'worst delay' as hightest value of average delay, that is, `avg(ArrDelay)` and `avg(DepDelay)`.
2. Note that for early departure/ arrival flights, the ArrDelay/ DepDelay value is negative. To only take those delayed flights into account, I exclude the negative values.
3. I don't drop any row with NA just yet, because one column with NA doesn't mean there is NA value in both ArrDelay and DepDelay. Therefore, I only filter out `DepDelay` is NA when I calculate `avg(DepDelay)` and sames goes for `avg(ArrDelay)`.
4. The goal is to find the location pair that has both `avg(ArrDelay)` and `avg(DepDelay)` being highest of all, however, since I can't be sure if it exists, I examine these two figures separately.

```
pd_ArrDelay = spark.sql(\
                        "SELECT Year, concat(Origin,'-',Dest) AS Location_pair, avg
                        FROM airlines\
                        WHERE ArrDelay IS NOT NULL AND ArrDelay > 0\
                        GROUP BY Location_pair, Year\
                        ORDER BY ArrDelay_mean DESC"
                        ).toPandas()
pd_DepDelay = spark.sql(\
                        "SELECT Year, concat(Origin,'-',Dest) AS Location_pair, avg
                        FROM airlines\
                        WHERE DepDelay IS NOT NULL AND DepDelay > 0\
                        GROUP BY Location_pair, Year\
                        ORDER BY DepDelay_mean DESC"
                        ).toPandas()
```

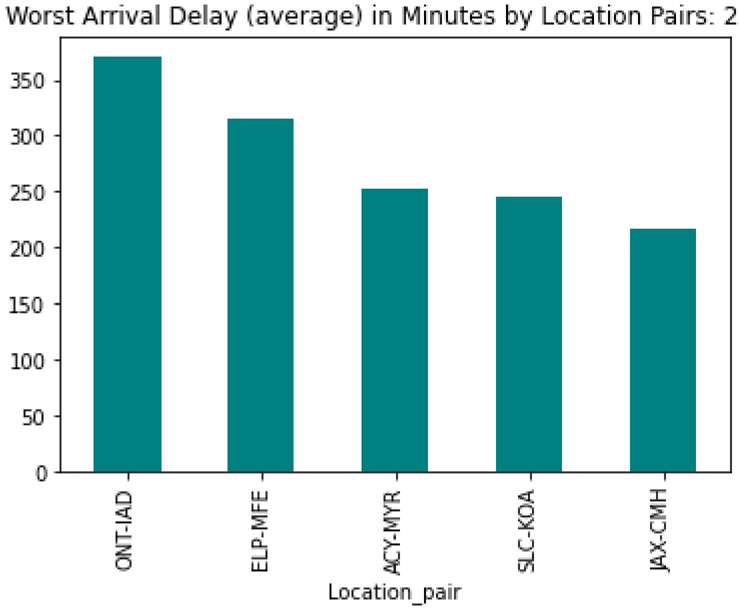
```
print(f'Worst Arrival Delay (average) in Minutes: \n{pd_ArrDelay.head()}\n')
print(f'Worst Departure Delay (average) in Minutes: \n{pd_DepDelay.head()}')
```

```
Worst Arrival Delay (average) in Minutes:
   Year Location_pair  ArrDelay_mean
0  2008       CMI-SPI          575.0
1  2007       ONT-IAD          370.0
2  2007       ELP-MFE          316.0
3  2008       ONT-SAN          257.0
4  2007       ACY-MYR          252.0

Worst Departure Delay (average) in Minutes:
   Year Location_pair  DepDelay_mean
```

```
0  2008    CMI-SPI    587.0
1  2007    ONT-IAD    386.0
2  2007    ABQ-GJT    366.0
3  2007    RAP-TWF    347.0
4  2008    SDF-SPI    329.0
```

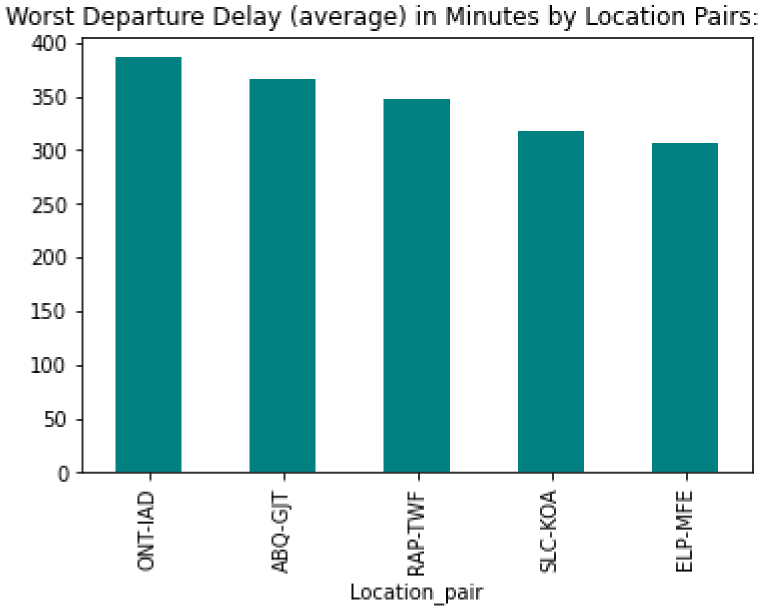
```
pd_ArrDelay[pd_ArrDelay['Year']==2007].head().\
plot(kind='bar',x='Location_pair', y='ArrDelay_mean', color='teal', legend=None, ti
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fad5ea42130>
```



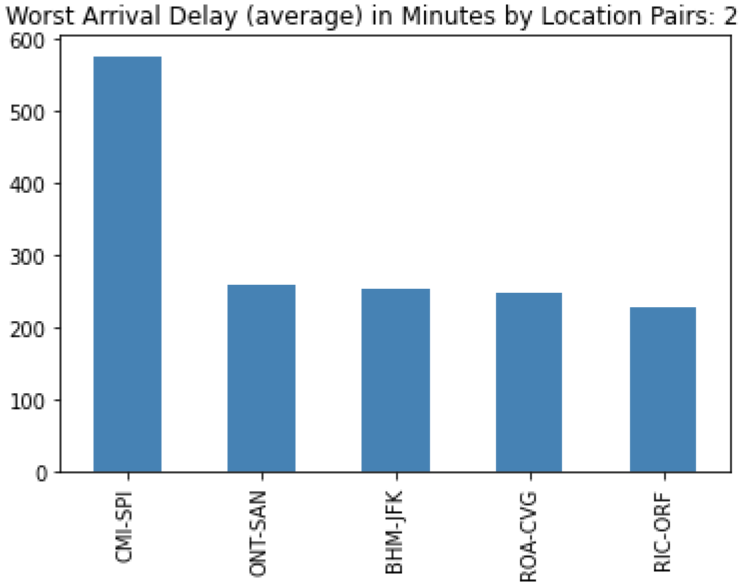

```
pd_DepDelay[pd_DepDelay['Year']==2007].head().\
plot(kind='bar',x='Location_pair', y='DepDelay_mean', color='teal', legend=None, ti
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fad5ebf13d0>
```



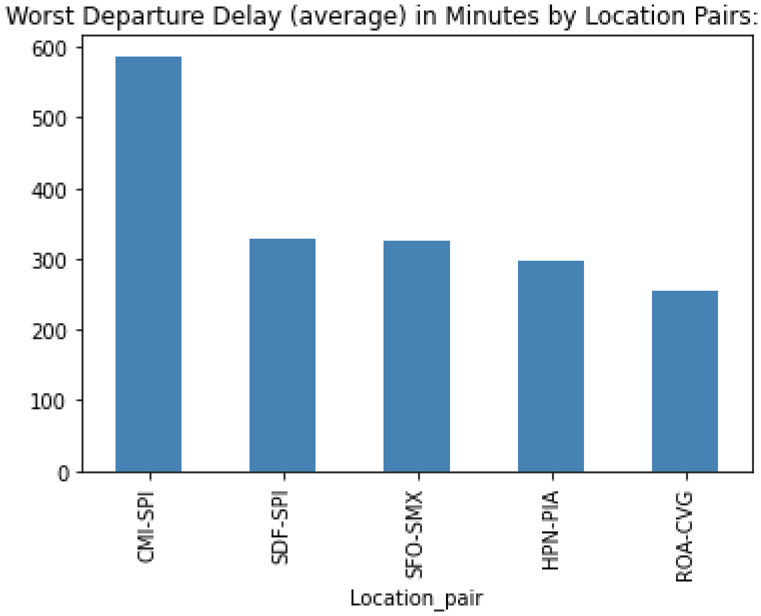

```
pd_ArrDelay[pd_ArrDelay['Year']==2008].head().\
plot(kind='bar',x='Location_pair', y='ArrDelay_mean', color='steelblue', legend=Non
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fad5ebebee0>
```




```
pd_DepDelay[pd_DepDelay['Year']==2008].head().\
plot(kind='bar',x='Location_pair', y='DepDelay_mean', color='steelblue', legend=Non
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fad5edf4430>
```



Answer 1.

In 2007, *ONT-IAD* location pair had the worst delay for both arrivals and departures, with an average of 370.0 minutes ArrDelay and an average of 386.0 minutes DepDelay;

In 2008, *CMI-SPI* location pair had the worst delay for both arrivals and departures, with an average of 575.0 minutes ArrDelay and an average of 587.0 minutes DepDelay.

Question 2.

Which locations (Origin and Dest pairs) had the fewest delays for both arrivals (ArrDelay) and departures (DepDelay) - for each year

Assumptions made:

1. I define 'fewest delay' as lowest value of average delay, that is, `avg(ArrDelay)` and `avg(DepDelay)`.
2. Note that for early departure/ arrival flights, the ArrDelay/ DepDelay value is negative. To only take those delayed flights into account, we exclude the negative values.
3. The goal is to find the location pair that has both `avg(ArrDelay)` and `avg(DepDelay)` being lowest of all, however, since I can't be sure if it exists, here I examine these two figures separately. However, since the location pairs with lowest value for `avg(ArrDelay)` and `avg(DepDelay)` are different, we combine these two figure as one `Total_Delay_mean` to pick a single location pair with fewest delay.

```
pd_ArrDelay = pd_ArrDelay.sort_values('ArrDelay_mean')
pd_DepDelay = pd_DepDelay.sort_values('DepDelay_mean')


print(f'Fewest Arrival Delay (average) in Minutes: \n{pd_ArrDelay.head(5)}\n')
print(f'Fewest Departure Delay (average) in Minutes: \n{pd_DepDelay.head(5)}')
```

```
Fewest Arrival Delay (average) in Minutes:
       Year Location_pair  ArrDelay_mean
10165  2008       PIR-MSP            1.0
10160  2007       RIC-BWI            1.0
10161  2008       CLE-TUL            1.0
10164  2008       DCA-PLN            1.0
10163  2008       BUR-PMD            1.0

Fewest Departure Delay (average) in Minutes:
       Year Location_pair  DepDelay_mean
10392  2008       BOI-ATL            1.0
10381  2007       SGF-COS            1.0
10382  2007       FCA-PIH            1.0
10383  2007       MKE-MKC            1.0
10384  2007       SLC-SUX            1.0
```

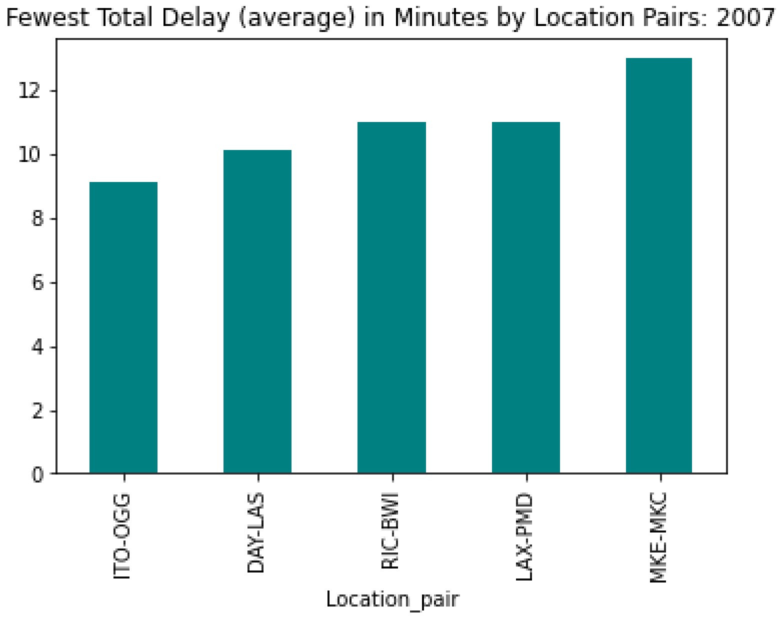
If we compare the fewest ArrDelay and fewest DepDelay separately, we can see a number of location pairs with the same lowest (yet positive) DepDelay/ ArrDelay value, which is 1.0. Therefore, to get one location pair with the fewest delay for both DepDelay and ArrDelay, we add these 2 figures together as total delay to further compare.

```
pd_Total_Delay = pd_ArrDelay.merge(pd_DepDelay, left_on=('Year','Location_pair'),ri


pd_Total_Delay['Total_Delay_mean'] = pd_Total_Delay['ArrDelay_mean'] + pd_Total_Del
pd_Total_Delay = pd_Total_Delay.sort_values('Total_Delay_mean')


print(f'Fewest Total Delay (average) in Minutes: \n{pd_Total_Delay.head(10)}')
```

```
Fewest Total Delay (average) in Minutes:
    Year Location_pair  ArrDelay_mean  DepDelay_mean  Total_Delay_mean
3   2008       BUR-PMD       1.000000       1.000000          2.000000
11  2008       MCO-OKC       4.000000       1.000000          5.000000
2   2008       CLE-TUL       1.000000       5.000000          6.000000
4   2008       CMH-PBI       1.000000       7.000000          8.000000
17  2008       ACY-JFK       5.000000       3.000000          8.000000
0   2008       PIR-MSP       1.000000       7.500000          8.500000
26  2008       MSY-IND       7.000000       1.666667          8.666667
23  2008       PIT-PBI       6.500000       2.333333          8.833333
6   2008       IAH-AGS       3.000000       6.000000          9.000000
13  2007       ITO-OGG       4.151515       5.000000          9.151515
```

```
pd_Total_Delay[pd_Total_Delay['Year']==2007].head().\
plot(kind='bar',x='Location_pair', y='Total_Delay_mean', color='teal', legend=None,
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fad5ecab460>
```




```
pd_Total_Delay[pd_Total_Delay['Year']==2008].head().\
plot(kind='bar',x='Location_pair', y='Total_Delay_mean', color='steelblue', legend=
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fad5d7551c0>
```

Answer 2.

In 2007, *BUR-PMD* location pair had the fewest total delays (average arrival delay plus average departure delay), 2.0 minutes;
In 2008, *ITO-OGG* location pair had the fewest total delays (average arrival delay plus average departure delay), 9.15 minutes.

Question 3.

Do you see any significant seasonality effects for delays?

Assumptions made:

1. To observe the variation of delays, there are 2 measurements I am interested in:
   (1)Number_of_Delay: Delayed flights count
   (2)Average_Delay: The average of [ArrDelay + DepDelay].
2. I compare 2007 data with 2008 data to see if there is a clear pattern in different months.

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
pd_Delay_by_month = spark.sql(\
                        "SELECT Year, Month, count(*) AS Number_of_Delay, avg((ArrD
                        FROM airlines\
                        WHERE ArrDelay IS NOT NULL AND ArrDelay > 0 AND DepDelay IS
                        GROUP BY Year, Month\
                        ORDER BY Month ASC"
                        ).toPandas()
```

```
pd_Delay_by_month.head()
```

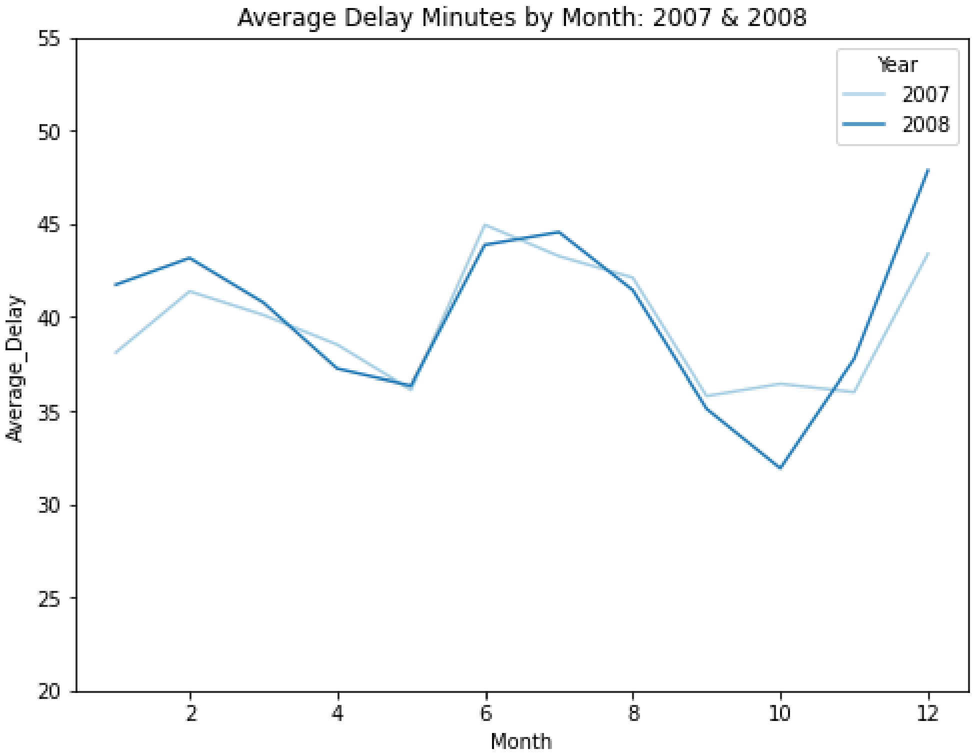
	Year	Month	Number_of_Delay	Average_Delay
0	2007	1	197311	38.100727
1	2008	1	192977	41.736365
2	2007	2	207335	41.379492
3	2008	2	200339	43.168120
4	2007	3	210924	40.112801

```
fig, ax = plt.subplots(figsize=(8, 6))
```

```
ax = sns.lineplot(data = pd_Delay_by_month, x='Month', y='Average_Delay', hue='Year
```

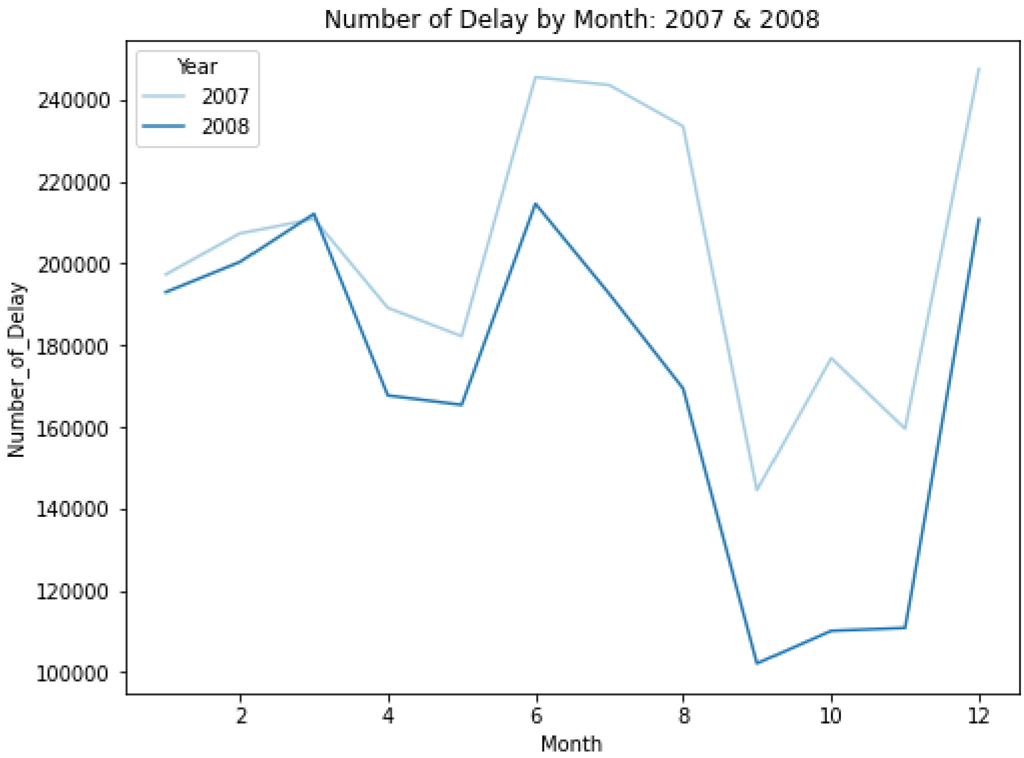
```
ax.set_title("Average Delay Minutes by Month: 2007 & 2008")
ax.set(ylim=(20,55))
```

```
[(20.0, 55.0)]
```



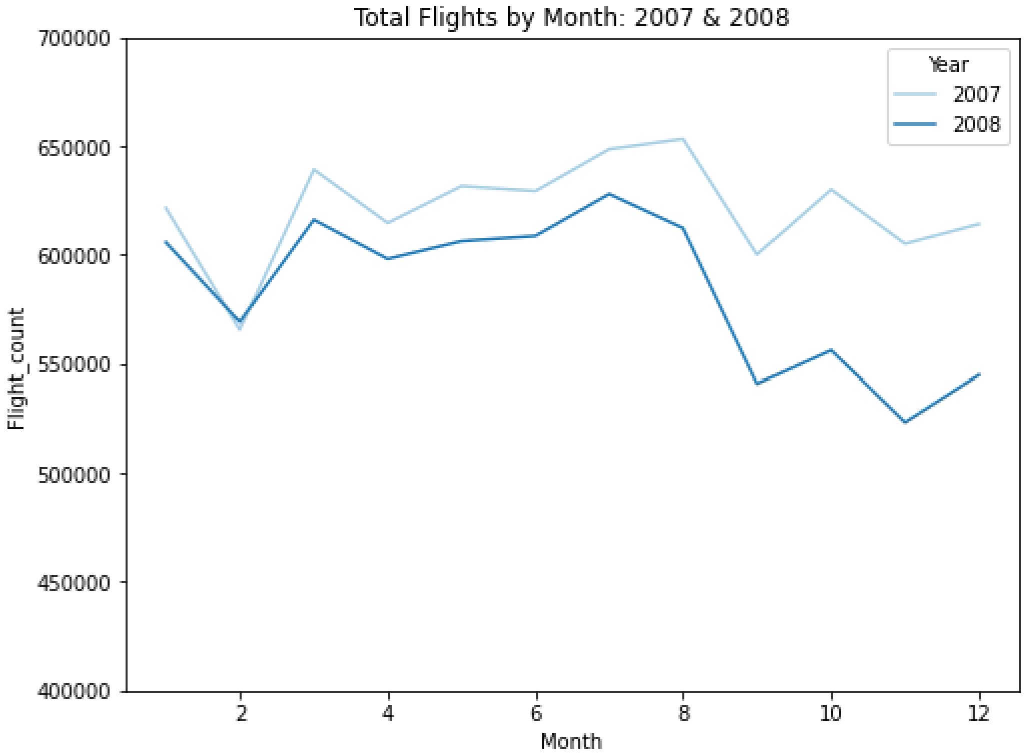

```
fig, ax = plt.subplots(figsize=(8, 6))

ax = sns.lineplot(data = pd_Delay_by_month, x='Month', y='Number_of_Delay', hue='Ye
ax.set_title("Number of Delay by Month: 2007 & 2008")
```

```
Text(0.5, 1.0, 'Number of Delay by Month: 2007 & 2008')
```



```
pd_all_flights = spark.sql('SELECT Year, Month, count(*) AS Flight_count FROM airli

ig, ax = plt.subplots(figsize=(8, 6))

ax = sns.lineplot(data = pd_all_flights, x='Month', y='Flight_count', hue='Year', p
ax.set_title("Total Flights by Month: 2007 & 2008")
ax.set(ylim=(400_000, 700_000))
```

```
[(400000.0, 700000.0)]
```



Answer 3.

Based on above 3 plots, I can see the seasonality effect playing an important role in the fluctuation of delay over time.

- First, in "Average Delay Minutes by Month: 2007 & 2008" plot, it is clear that the average delay time is relatively higher in June-August as well as in December, which are all considered travelling season in US. When taking a closer look at 2007 data comparing with 2008 data, it shows a similar pattern of fluctuation in different month, indicating the seasonality effect.
- Second, in "Number of Delay by Month: 2007 & 2008", I examine the number of flights that were delayed, and again it's showing a similar pattern of fluctuation in different month, with traveling season reaching the peak. And to take a step further, I also plot the "Total Flights by Month: 2007 & 2008" to see if the number of delayed flights increases just because the total number of flights also increases. Take December for example, there was not a significant increase in total flights (in both 2007 & 2008) but there indeed was a significant increase in number of delayed flights.

To sum up, there is significant seasonality effects for delays.

Question 4.

Do you see any increase or decrease in delays on weekends?

1. To observe the variation of delays, we have 2 measurements:
   (1)Number_of_Delay: Delayed flights count
   (2)Average_Delay: The average of [ArrDelay + DepDelay]. Both figures only take flights that are delayed into account.
2. In `DayOfWeek`, 1-5 are weekdays, 6-7 are weekends, so the focus is on comparing the difference between 1-5 and 6-7.

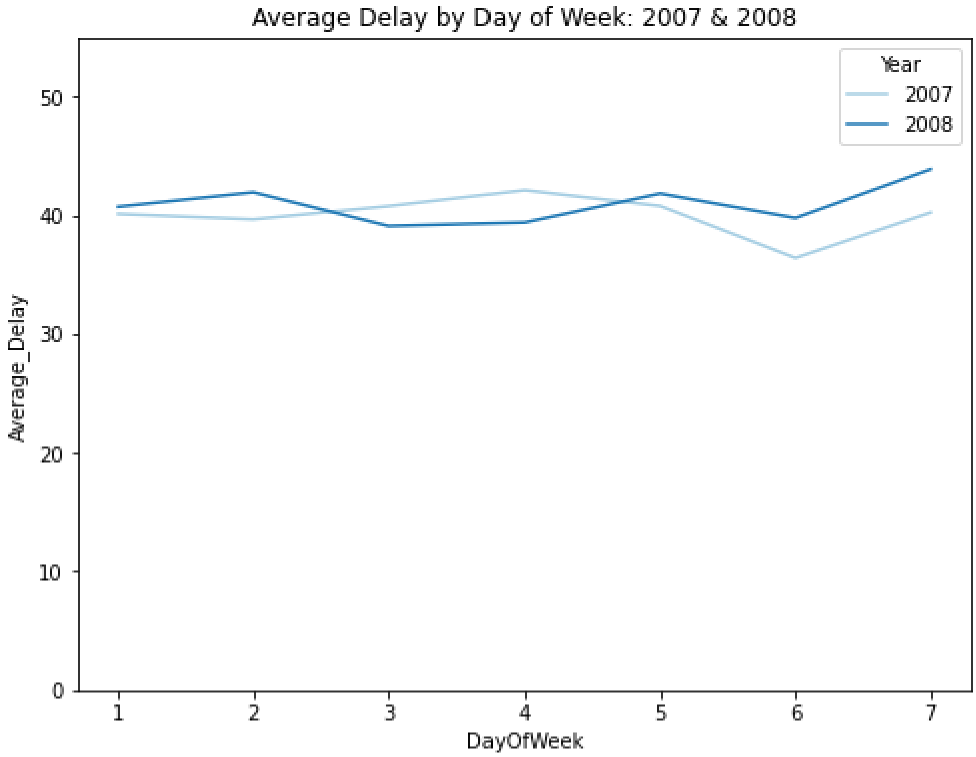
```
pd_Delay_by_dow = spark.sql(\
                        "SELECT Year, DayOfWeek, count(*) AS Number_of_Delay, avg((
                        FROM airlines\
                        WHERE ArrDelay IS NOT NULL AND ArrDelay > 0 AND DepDelay IS
                        GROUP BY Year, DayOfWeek\
                        ORDER BY DayOfWeek ASC"
                        ).toPandas()
```

```
pd_Delay_by_dow.head(15)
```

	Year	DayOfWeek	Number_of_Delay	Average_Delay
0	2007	1	374552	40.104210
1	2008	1	308296	40.715715
2	2007	2	311982	39.650130
3	2008	2	277855	41.926186
4	2007	3	338080	40.766119
5	2008	3	279970	39.088570
6	2007	4	383123	42.101003
7	2008	4	309810	39.389322
8	2007	5	407843	40.777391
9	2008	5	343405	41.822872
10	2007	6	270615	36.398134
11	2008	6	230479	39.766525
12	2007	7	351517	40.225154
13	2008	7	299185	43.867696

```
fig, ax = plt.subplots(figsize=(8, 6))

ax = sns.lineplot(data = pd_Delay_by_dow, x='DayOfWeek', y='Average_Delay', hue='Ye
ax.set_title("Average Delay by Day of Week: 2007 & 2008")
ax.set(ylim=(0, 55))
```

```
[(0.0, 55.0)]
```



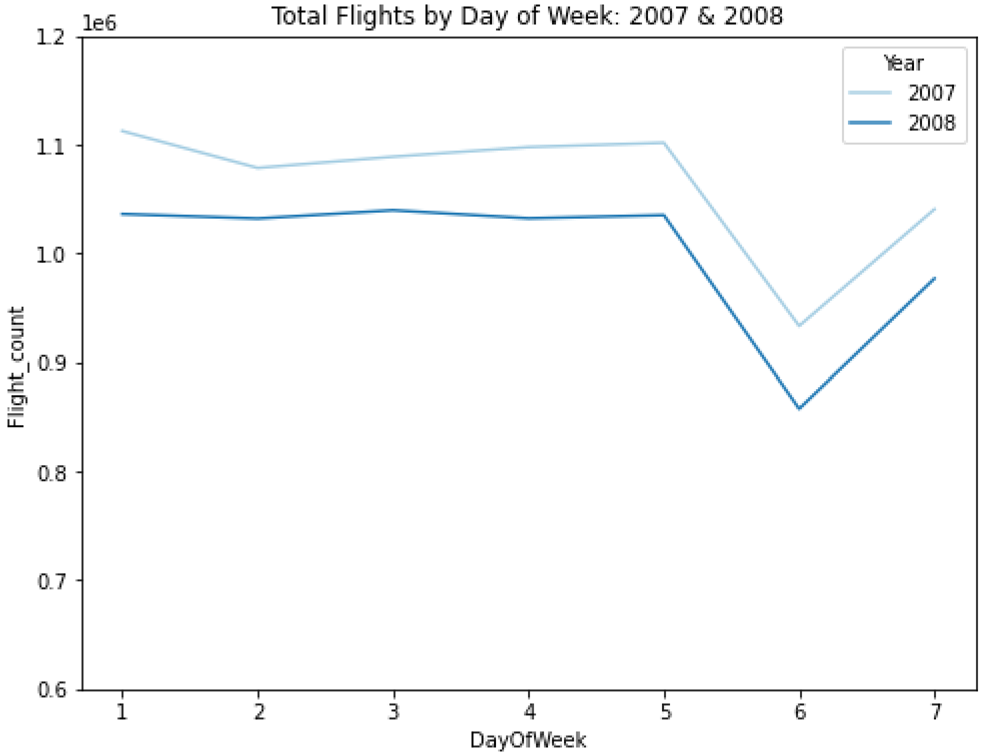

```
fig, ax = plt.subplots(figsize=(8, 6))

ax = sns.lineplot(data = pd_Delay_by_dow, x='DayOfWeek', y='Number_of_Delay', hue='
ax.set_title("Number of Delay by Day of Week: 2007 & 2008")
ax.set(ylim=(0, 500_000))
```

```
[(0.0, 500000.0)]
```

Number of Delay by Day of Week: 2007 & 2008

```
pd_all_flights = spark.sql('SELECT Year, DayOfWeek, count(*) AS Flight_count FROM a

ig, ax = plt.subplots(figsize=(8, 6))

ax = sns.lineplot(data = pd_all_flights, x='DayOfWeek', y='Flight_count', hue='Year
ax.set_title("Total Flights by Day of Week: 2007 & 2008")
ax.set(ylim=(600_000, 1_200_000))
```

```
[(600000.0, 1200000.0)]
```



Answer 4.

Based on above 3 plots, it is difficult to conclude that there is a significant increase or decrease in delays on weekends.

- First, in "Average Delay by Day of Week: 2007 & 2008" plot, the average delayed time is very similar in weekends and weekdays (both in 2007 and 2008), showing no clear increase or decrease between weekdays and weekends. If any, there is a slight decrease from Friday to Saturday in both 2007 data and 2008 data.
- Second, in "Number of Delay by Day of Week: 2007 & 2008", although this time the drop from Friday to Saturday becomes much clearer in both 2007 and 2008, if we factor in the fact that Saturday in general just had much fewer flights, then it explains the drop in number of delayed flight on Saturday. Similarly, we can see the flctuation pattern showing in "Number of Delay by Day of Week" roughly matches the flctuation pattern showing in "Total Flights by Day of Week: 2007 & 2008".

To sum up, if we exclude the factor of total number of flights being different, there is no clear increase or decrease in delays on weekends.

Question 5.

Are flights equally distributed throughout the day?
Plot the distribution of DepTime, ArrTime (actual departure and arrival time)

```
airlines_df_Time = airlines_df_feat.\
filter(airlines_df_feat.DepTime.isNotNull()).\
filter(airlines_df_feat.ArrTime.isNotNull()).\
filter(airlines_df_feat.CRSDepTime.isNotNull()).\
filter(airlines_df_feat.CRSArrTime.isNotNull()).\
select('DepTime','ArrTime','CRSDepTime','CRSArrTime', 'Year')
```

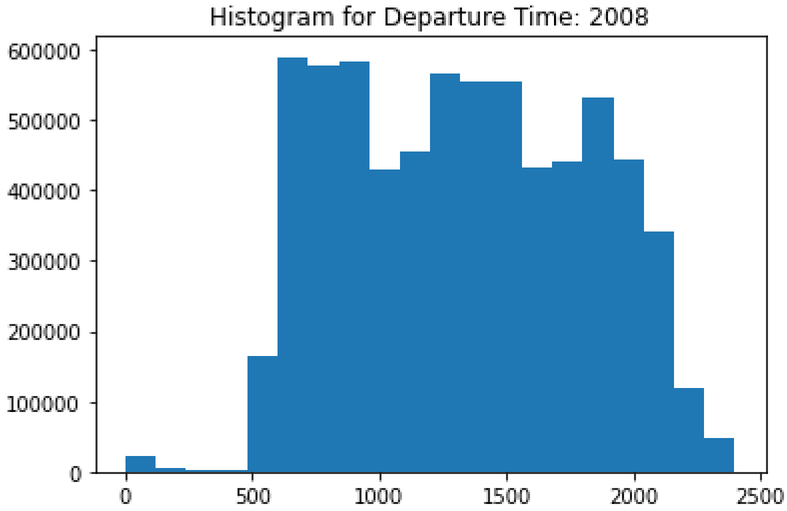
```
airlines_df_Time.printSchema()
```

```
root
 |-- DepTime: integer (nullable = true)
 |-- ArrTime: integer (nullable = true)
 |-- CRSDepTime: integer (nullable = true)
 |-- CRSArrTime: integer (nullable = true)
 |-- Year: integer (nullable = true)
```

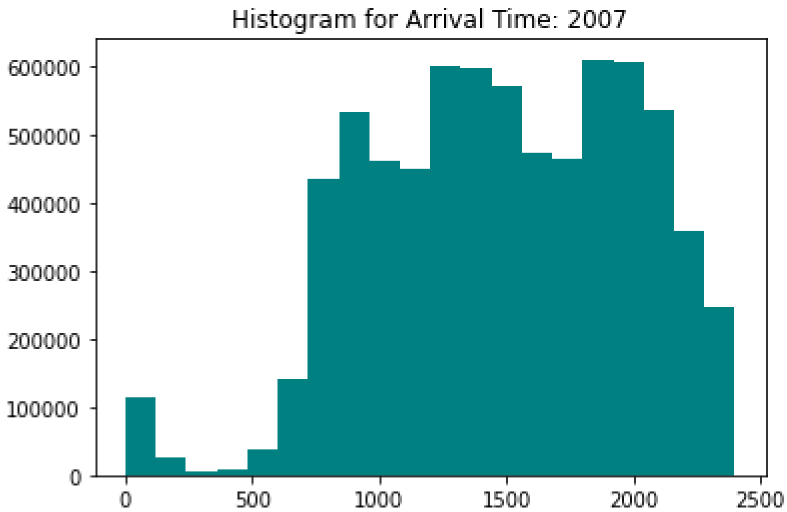
Since the data is too big to convert it to pandas for plotting distribution. Instead, it's easier to just plot the distribution on spark dataframe.
Source: https://stackoverflow.com/questions/36043256/making-histogram-with-spark-dataframe-column

```
# Plot histogram on spark dataframe

bins, counts = airlines_df_Time.filter(airlines_df_Time.Year == 2007).select('DepTi

plt.hist(bins[:-1], bins=bins, weights=counts, color='teal')
plt.title('Histogram for Departure Time: 2007')
plt.show()
```


```
# Plot histogram on spark dataframe

bins, counts = airlines_df_Time.filter(airlines_df_Time.Year == 2008).select('DepTi

plt.hist(bins[:-1], bins=bins, weights=counts)
plt.title('Histogram for Departure Time: 2008')
plt.show()
```

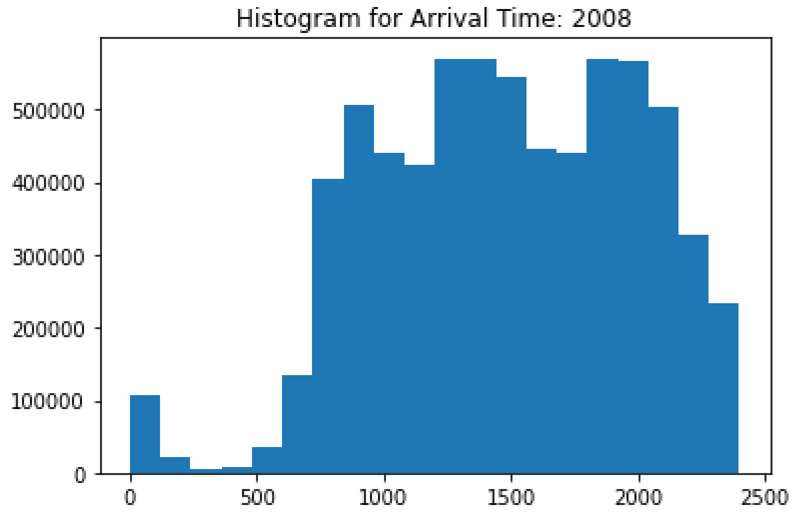



```
# Plot histogram on spark dataframe

bins, counts = airlines_df_Time.filter(airlines_df_Time.Year == 2007).select('ArrTi

plt.hist(bins[:-1], bins=bins, weights=counts, color='teal')
plt.title('Histogram for Arrival Time: 2007')
plt.show()
```




```
# Plot histogram on spark dataframe
bins, counts = airlines_df_Time.filter(airlines_df_Time.Year == 2008).select('ArrTi
```

```
plt.hist(bins[:-1], bins=bins, weights=counts)
plt.title('Histogram for Arrival Time: 2008')
plt.show()
```



Answer 5.

All 4 histogram plots above show that the flights weren't equally distributed throughout the day. It is most obvious that from midnight to early morning (around 5 am), there were very few flights, supposedly due to aviation regulation (curfew).

Question 6.

Do you see the worst delays at any certain times of the day? Compare DepTime, ArrTime with CRSDepTime, CRSArrTime (scheduled arrival and departure time; CRS is the Computer Reservation System)

1. To observe the variation of delays, there are 2 measurements I am interested in:
   (1)Number_of_Delay: Delayed flights count.
   (2)Delay_mean: The average of delayed time (minutes) per hour.
2. Compare the DepDelay and ArrDelay separatly using above 2 measurements.

```
df_Delay_Time = airlines_df_feat.\
select('CRSDepTime','CRSArrTime', 'DepDelay', 'ArrDelay')
# df_Delay_Time.show()
```

```
pd_CRSDep = df_Delay_Time.filter((df_Delay_Time.DepDelay >0)&(df_Delay_Time.DepDela
groupby('CRSDepTime').agg(count('*').alias('Num_of_DepDelay'), avg('DepDelay').alia
# pd_CRSDep.head()
```

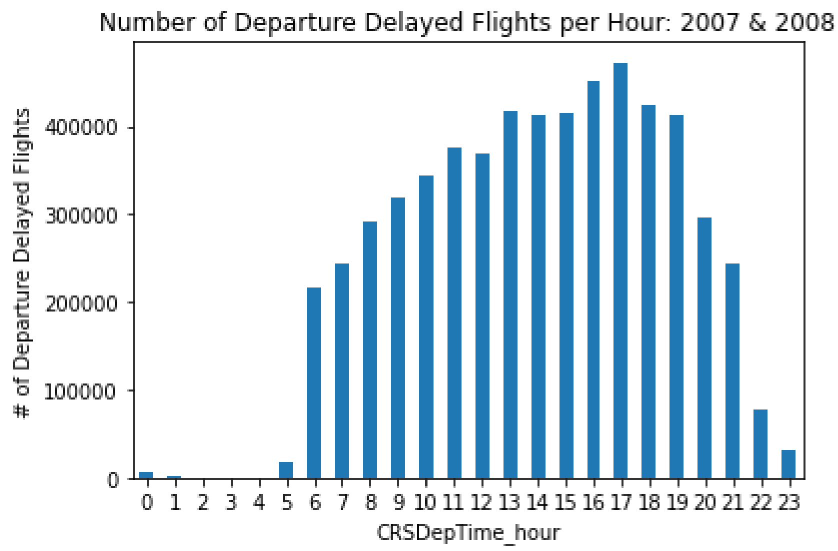
```
CRSDepTime_hour = []

for i in pd_CRSDep['CRSDepTime']:
  if i >= 1000:
    x = int(str(i)[:2])
  elif i >= 100:
    x = int(str(i)[:1])
  else:
    x = 0
  CRSDepTime_hour.append(x)

pd_CRSDep['CRSDepTime_hour'] = CRSDepTime_hour
pd_CRSDep = pd_CRSDep.groupby('CRSDepTime_hour', as_index =False).sum()

# pd_CRSDep.head()
```

```
pd_CRSDep.head()
```

	CRSDepTime_hour	CRSDepTime	Num_of_DepDelay	DepDelay_mean
0	0	1221	6448	1296.338696
1	1	3222	2361	942.511514
2	2	3154	538	486.798535
3	3	2257	278	678.705278
4	4	5602	398	727.567093

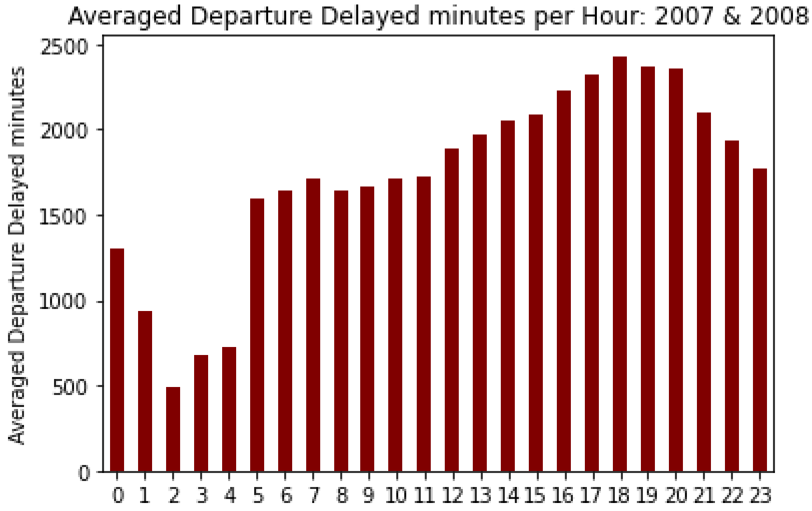
```
pd_CRSDep.plot(kind='bar', x='CRSDepTime_hour', y='Num_of_DepDelay', legend=None, r
               title="Number of Departure Delayed Flights per Hour: 2007 & 2008", y
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fad55e70d30>




```
pd_CRSDep.plot(kind='bar', x='CRSDepTime_hour', y='DepDelay_mean', legend=None, rot
               title="Averaged Departure Delayed minutes per Hour: 2007 & 2008", yl
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fad55e727f0>
```

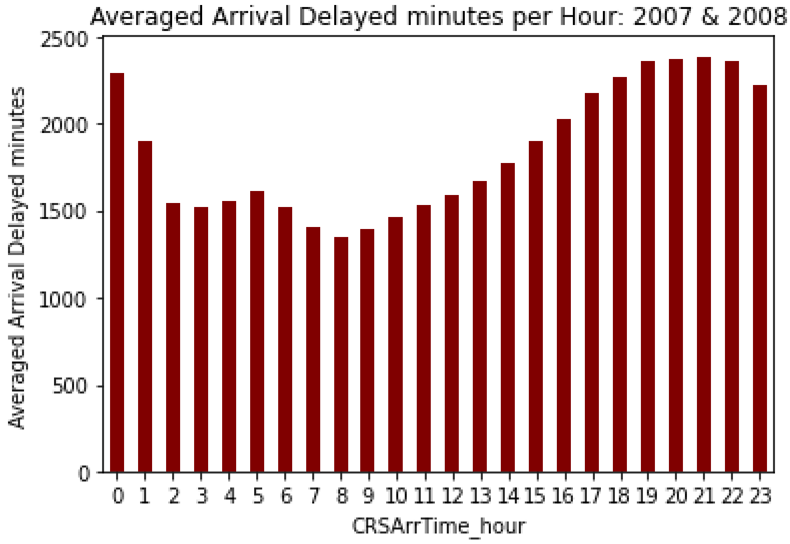



```
pd_CRSArr = df_Delay_Time.filter((df_Delay_Time.ArrDelay >0)&(df_Delay_Time.ArrDela
groupby('CRSArrTime').agg(count('*').alias('Num_of_ArrDelay'), avg('ArrDelay').alia
# pd_CRSArr.head()


CRSArrTime_hour = []

for i in pd_CRSArr['CRSArrTime']:
  if i >= 1000:
    x = int(str(i)[:2])
  elif i >= 100:
    x = int(str(i)[:1])
  else:
    x = 0
  CRSArrTime_hour.append(x)

pd_CRSArr['CRSArrTime_hour'] = CRSArrTime_hour
pd_CRSArr = pd_CRSArr.groupby('CRSArrTime_hour', as_index =False).sum()

# pd_CRSArr.head()


pd_CRSArr.plot(kind='bar', x='CRSArrTime_hour', y='Num_of_ArrDelay', legend=None, r
                    title="Number of Arrival Delayed Flights per Hour: 2007 & 2008"
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fad55b40940>
```

```
pd_CRSArr.plot(kind='bar', x='CRSArrTime_hour', y='ArrDelay_mean', legend=None, rot
               title="Averaged Arrival Delayed minutes per Hour: 2007 & 2008", ylab
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fad55d87fa0>
```



Answer 6.

1. Departure Delay:
   (1) Number of delayed flights: relatively high from 1pm (13) to 7pm (19), reaching peak around late afternoon, 4pm (16) - 5pm (17), possibly resulting from busy hour; significantly low from 0am to 5am, supposedly because very few flights were scheduled to take-off at this time interval.
   (2) Average delayed time (minutes): similarly, the peak happens during busy hour; however, during the time interval when there were very few flights, the delayed was usually quite significant.
2. Arrival Delay:
   (1) Number of delayed flights: relatively high from 4pm (16) to 8pm (20), reaching peak at 8pm, possibly resulting from busy hour; significantly low from 1am to 5am, supposedly because very few flights were scheduled to arrive at this time interval.
   (2) Average delayed time (minutes): quite different from other 3 plots, this plot is more evenly distributed, with relatively high value from 7pm (19) to 10pm (22), and relatively low value in the early morning, from 2am to 8am.