Muhammad Shas K T (UID: 281775)

Analysis of Airline Delay using Spark

In [5]:
```
sc.stop()
```

a) Create a new Spark Session with new SparkConfig

In [7]:
```
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession, HiveContext
```

In [11]:
```
config = SparkConf().setAppName("airline_analysis").setMaster("local[2]")
sc = SparkContext.getOrCreate(conf=config)
```

In [13]:
```
sc
```

Out[13]: **SparkContext**

Spark UI

Version	`v2.4.8`
Master	`local[2]`
AppName	`airline_analysis`

b) Create new instance of Spark SQL session and define new DataFrame using Flights_Delay.csv dataset.

In [14]:
```
spark = SparkSession.builder.appName('airline_analysis').getOrCreate()
```

In [20]:
```
spark.sql
```

Out[20]:
```
<bound method SparkSession.sql of <pyspark.sql.session.SparkSession object at 0x7f1f7a2751d0>>
```

In [22]:
```
flight_df = spark.read.csv("file:///home/hadoop/Downloads/Flights_Delay.csv", header=True, inferSchema=True)
```

In [25]:
```
flight_df.show(5)
```

```
+---+----+-----+---+-----------+-------+-------------+-----------+--------------+-------------------+------------
-------+--------------+---------------+--------+----------+--------------+------------+--------+--------+--------
-+-------+-----------------+------------+-------------+--------+---------+-------------------+----------------+--
------------+-------------+-------------------+-------------+
| ID|YEAR|MONTH|DAY|DAY_OF_WEEK|AIRLINE|FLIGHT_NUMBER|TAIL_NUMBER|ORIGIN_AIRPORT|DESTINATION_AIRPORT|SCHEDULED_DE
PARTURE|DEPARTURE_TIME|DEPARTURE_DELAY|TAXI_OUT|WHEELS_OFF|SCHEDULED_TIME|ELAPSED_TIME|AIR_TIME|DISTANCE|WHEELS_O
N|TAXI_IN|SCHEDULED_ARRIVAL|ARRIVAL_TIME|ARRIVAL_DELAY|DIVERTED|CANCELLED|CANCELLATION_REASON|AIR_SYSTEM_DELAY|SE
CURITY_DELAY|AIRLINE_DELAY|LATE_AIRCRAFT_DELAY|WEATHER_DELAY|
+---+----+-----+---+-----------+-------+-------------+-----------+--------------+-------------------+------------
-------+--------------+---------------+--------+----------+--------------+------------+--------+--------+--------
-+-------+-----------------+------------+-------------+--------+---------+-------------------+----------------+--
------------+-------------+-------------------+-------------+
|  0|2015|    3|  4|          3|     EV|         5170|     N842AS|           CVG|                XNA|
935|           954|             19|      16|      1010|           115|         129|     108|     562|     1058|
5|             1030|        1103|           33|       0|        0|               null|              14|
0|           19|                  0|            0|
|  1|2015|    2|  2|          1|     MQ|         3584|     N646MQ|           DFW|                SPS|
1240|          1316|             36|      11|      1327|            50|          46|      30|     113|     1357|
5|             1330|        1402|           32|       0|        0|               null|               0|
0|           32|                  0|            0|
|  2|2015|    1| 27|          2|     B6|          716|     N309JB|           JAX|                DCA|
1335|          1505|             90|      16|      1521|           104|         110|      91|     634|     1652|
3|             1519|        1655|           96|       0|        0|               null|               6|
0|           90|                  0|            0|
|  3|2015|    1| 28|          3|     EV|         4289|     N14162|           COS|                IAH|
1442|          1435|             -7|      13|      1448|           139|         127|     101|     809|     1729|
13|             1801|        1742|          -19|       0|        0|               null|            null|
null|         null|               null|         null|
|  4|2015|    2|  5|          4|     EV|         5584|     N851AS|           ATL|                AVL|
1255|          1250|             -5|      25|      1315|            48|          62|      34|     164|     1349|
3|             1343|        1352|            9|        0|         0|                null|             null|
null|         null|               null|         null|
+---+----+-----+---+-----------+-------+------------+-----------+--------------+-------------------+------------
-------+--------------+---------------+--------+----------+--------------+------------+--------+--------+--------
-+-------+-----------------+------------+-------------+--------+---------+-------------------+----------------+--
------------+-------------+-------------------+-------------+
only showing top 5 rows
```

c) Create table Spark HIVE table flights_table

In [36]:
```
#Creating a database flights_db
spark.sql("create database flights_db")
spark.sql("show databases").show()
spark.sql("use flights_db")
```

```
+------------+
|databaseName|
+------------+
|  banking_db|
|     default|
|  flights_db|
+------------+
```

Out[36]:
```
DataFrame[]
```

In [42]:
```
flight_df.createOrReplaceTempView("flights_table")
spark.sql("show tables").show()
```

```
+--------+-------------+-----------+
|database|    tableName|isTemporary|
+--------+-------------+-----------+
|        |flights_table|       true|
+--------+-------------+-----------+
```

d) Describe the table schema & show top 10 rows of Dataset

In [46]:
```
spark.sql('describe flights_table').show()
```

```
+-------------------+---------+-------+
|           col_name|data_type|comment|
+-------------------+---------+-------+
|                 ID|      int|   null|
|               YEAR|      int|   null|
|              MONTH|      int|   null|
|                DAY|      int|   null|
|        DAY_OF_WEEK|      int|   null|
|            AIRLINE|   string|   null|
|      FLIGHT_NUMBER|      int|   null|
|        TAIL_NUMBER|   string|   null|
|     ORIGIN_AIRPORT|   string|   null|
|DESTINATION_AIRPORT|   string|   null|
|SCHEDULED_DEPARTURE|      int|   null|
|     DEPARTURE_TIME|      int|   null|
|    DEPARTURE_DELAY|      int|   null|
|           TAXI_OUT|      int|   null|
|         WHEELS_OFF|      int|   null|
|     SCHEDULED_TIME|      int|   null|
|       ELAPSED_TIME|      int|   null|
|           AIR_TIME|      int|   null|
|           DISTANCE|      int|   null|
|          WHEELS_ON|      int|   null|
+-------------------+---------+-------+
only showing top 20 rows
```

In [47]:
```
spark.sql('select * from flights_table').show(10)
```

```
+---+----+-----+---+-----------+-------+------------+-----------+--------------+-------------------+------------
```

```
-------+--------------+--------------+--------+----------+--------------+------------+--------+--------+--------
-+-------+-----------------+------------+------------+--------+---------+-------------------+----------------+--
------------+-------------+-------------------+-------------+
| ID|YEAR|MONTH|DAY|DAY_OF_WEEK|AIRLINE|FLIGHT_NUMBER|TAIL_NUMBER|ORIGIN_AIRPORT|DESTINATION_AIRPORT|SCHEDULED_DE
PARTURE|DEPARTURE_TIME|DEPARTURE_DELAY|TAXI_OUT|WHEELS_OFF|SCHEDULED_TIME|ELAPSED_TIME|AIR_TIME|DISTANCE|WHEELS_O
N|TAXI_IN|SCHEDULED_ARRIVAL|ARRIVAL_TIME|ARRIVAL_DELAY|DIVERTED|CANCELLED|CANCELLATION_REASON|AIR_SYSTEM_DELAY|SE
CURITY_DELAY|AIRLINE_DELAY|LATE_AIRCRAFT_DELAY|WEATHER_DELAY|
+---+----+-----+---+-----------+-------+-------------+-----------+--------------+-------------------+------------
-------+--------------+--------------+--------+----------+--------------+------------+--------+--------+--------
-+-------+-----------------+------------+------------+--------+---------+-------------------+----------------+--
------------+-------------+-------------------+-------------+
|  0|2015|    3|  4|          3|     EV|         5170|     N842AS|           CVG|                XNA|
935|            954|             19|      16|       1010|           115|         129|     108|     562|     1058|
5|            1030|        1103|           33|       0|         0|               null|              14|
0|           19|              0|            0|
|  1|2015|    2|  2|          1|     MQ|         3584|     N646MQ|           DFW|                SPS|
1240|           1316|             36|      11|       1327|            50|          46|      30|     113|     1357|
5|            1330|        1402|           32|       0|         0|               null|               0|
0|           32|              0|            0|
|  2|2015|    1| 27|          2|     B6|          716|     N309JB|           JAX|                DCA|
1335|           1505|             90|      16|       1521|           104|         110|      91|     634|     1652|
3|            1519|        1655|           96|       0|         0|               null|               6|
0|           90|              0|            0|
|  3|2015|    1| 28|          3|     EV|         4289|     N14162|           COS|                IAH|
1442|           1435|             -7|      13|       1448|           139|         127|     101|     809|     1729|
13|            1801|        1742|          -19|       0|         0|               null|            null|
null|         null|           null|         null|
|  4|2015|    2|  5|          4|     EV|         5584|     N851AS|           ATL|                AVL|
1255|           1250|             -5|      25|       1315|            48|          62|      34|     164|     1349|
3|            1343|        1352|            9|       0|         0|               null|            null|
null|         null|           null|         null|
|  5|2015|    2| 15|          7|     UA|          712|     N438UA|           IAH|                SFO|
1535|           1554|             19|      18|       1612|           260|         237|     216|    1635|     1748|
3|            1755|        1751|           -4|       0|         0|               null|            null|
null|         null|           null|         null|
|  6|2015|    2| 19|          4|     OO|         5166|     N746SK|           HDN|                DEN|
928|            924|             -4|      11|        935|            67|          56|      29|     141|     1004|
16|            1035|        1020|          -15|       0|         0|               null|            null|
null|         null|           null|         null|
|  7|2015|    2| 27|          5|     DL|         1571|     N916DN|           ATL|                CAK|
2104|           2103|             -1|      20|       2123|           106|          97|      70|     528|     2233|
7|            2250|        2240|          -10|       0|         0|               null|            null|
null|         null|           null|         null|
|  8|2015|    1| 20|          2|     WN|          518|     N405WN|           HOU|                MEM|
2140|           2150|             10|       8|       2158|            80|          79|      68|     484|     2306|
3|            2300|        2309|            9|       0|         0|               null|            null|
null|         null|           null|         null|
|  9|2015|    2|  6|          5|     WN|          336|     N663SW|           DAL|                MAF|
1750|           1748|             -2|       7|       1755|            70|          62|      52|     319|     1847|
3|            1900|        1850|          -10|       0|         0|               null|            null|
null|         null|           null|         null|
+---+----+-----+---+-----------+-------+-------------+-----------+--------------+-------------------+------------
-------+--------------+--------------+--------+----------+--------------+------------+--------+--------+--------
-+-------+-----------------+------------+------------+--------+---------+-------------------+----------------+--
------------+-------------+-------------------+-------------+
only showing top 10 rows
```

e) Apply Query performance optimization techniques like – creating Partitioning DataFrame by a specific column, parquet data, caching, predicate pushdown methods etc.

1.Partitioning

In [58]:
```
#Partitioning based on 'AIRLINE' column
partitioned_df = flight_df.repartition("AIRLINE")
```

2.Parquet data

In [62]:
```
#Saving the DataFrame in Parquet format
partitioned_df.write.partitionBy("AIRLINE").parquet("file:///home/hadoop/Downloads/flights_data")
```

In [69]:
```
#Load the parquet files into a Dataframe
parquet_df = spark.read.parquet("file:///home/hadoop/Downloads/flights_data")
```

3.Caching

In [75]:

```
#Caching the Dataframe for improving performance for repeated queries
flight_df.cache()
```

Out[75]:

```
DataFrame[ID: int, YEAR: int, MONTH: int, DAY: int, DAY_OF_WEEK: int, AIRLINE: string, FLIGHT_NUMBER: int, TAIL_N
UMBER: string, ORIGIN_AIRPORT: string, DESTINATION_AIRPORT: string, SCHEDULED_DEPARTURE: int, DEPARTURE_TIME: int
, DEPARTURE_DELAY: int, TAXI_OUT: int, WHEELS_OFF: int, SCHEDULED_TIME: int, ELAPSED_TIME: int, AIR_TIME: int, DI
STANCE: int, WHEELS_ON: int, TAXI_IN: int, SCHEDULED_ARRIVAL: int, ARRIVAL_TIME: int, ARRIVAL_DELAY: int, DIVERTE
D: int, CANCELLED: int, CANCELLATION_REASON: string, AIR_SYSTEM_DELAY: int, SECURITY_DELAY: int, AIRLINE_DELAY: i
nt, LATE_AIRCRAFT_DELAY: int, WEATHER_DELAY: int]
```

4.Predicate pushdown

In [78]:

```
#Predicate pushdown by filtering
filtered_df = partitioned_df.filter("DEPARTURE_DELAY > 0")
filtered_df.show(5)
```

```
+---+----+-----+---+-----------+-------+-------------+-----------+--------------+-------------------+------------
-------+--------------+---------------+--------+----------+--------------+------------+--------+--------+--------
-+-------+-----------------+------------+-------------+--------+---------+-------------------+----------------+--
------------+-------------+-------------------+-------------+
| ID|YEAR|MONTH|DAY|DAY_OF_WEEK|AIRLINE|FLIGHT_NUMBER|TAIL_NUMBER|ORIGIN_AIRPORT|DESTINATION_AIRPORT|SCHEDULED_DE
PARTURE|DEPARTURE_TIME|DEPARTURE_DELAY|TAXI_OUT|WHEELS_OFF|SCHEDULED_TIME|ELAPSED_TIME|AIR_TIME|DISTANCE|WHEELS_O
N|TAXI_IN|SCHEDULED_ARRIVAL|ARRIVAL_TIME|ARRIVAL_DELAY|DIVERTED|CANCELLED|CANCELLATION_REASON|AIR_SYSTEM_DELAY|SE
CURITY_DELAY|AIRLINE_DELAY|LATE_AIRCRAFT_DELAY|WEATHER_DELAY|
+---+----+-----+---+-----------+-------+-------------+-----------+--------------+-------------------+------------
-------+--------------+---------------+--------+----------+--------------+------------+--------+--------+--------
-+-------+-----------------+------------+-------------+--------+---------+-------------------+----------------+--
------------+-------------+-------------------+-------------+
|  5|2015|    2| 15|          7|     UA|          712|     N438UA|           IAH|                SFO|
1535|          1554|              19|      18|      1612|            260|         237|     216|    1635|     1748|
3|             1755|        1751|           -4|       0|        0|               null|            null|
null|         null|               null|         null|
| 24|2015|    2| 28|          6|     UA|          792|     N463UA|           ORD|                SNA|
1815|          1821|               6|      31|      1852|            266|         295|     261|    1726|     2113|
3|             2041|        2116|           35|       0|        0|               null|              29|
0|            6|                  0|            0|
| 28|2015|    1| 12|          1|     UA|          532|     N401UA|           ORD|                DCA|
1801|          1811|              10|      35|      1846|            114|         135|      97|     612|     2123|
3|             2055|        2126|           31|       0|        0|               null|              21|
0|            0|                  0|           10|
| 97|2015|    2| 19|          4|     UA|         1205|     N36476|           LAX|                LAS|
2234|          2244|              10|      12|      2256|             73|          62|      43|     236|     2339|
7|             2347|        2346|           -1|       0|        0|               null|            null|
null|         null|               null|         null|
|108|2015|    1|  3|          6|     UA|          541|     N510UA|           JFK|                SFO|
859|           916|              17|      24|       940|            398|         372|     343|    2586|     1223|
5|             1237|        1228|           -9|       0|        0|               null|            null|
null|         null|               null|         null|
+---+----+-----+---+-----------+-------+-------------+-----------+--------------+-------------------+------------
-------+--------------+---------------+--------+----------+--------------+------------+--------+--------+--------
-+-------+-----------------+------------+-------------+--------+---------+-------------------+----------------+--
------------+-------------+-------------------+-------------+
only showing top 5 rows
```

Write Spark SQL queries to show following analysis with Visualization on Databricks Community Edition.

f) Average arrival delay caused by airlines

In [92]:

```
avgdelay_df = spark.sql('select AIRLINE, AVG(ARRIVAL_DELAY) AVG_ARRIVAL_DELAY from flights_table group by AIRLINE
avgdelay_df.show()
```

```
+-------+------------------+
|AIRLINE| AVG_ARRIVAL_DELAY|
+-------+------------------+
|     UA| 6.697221614526362|
|     NK|14.206426484907498|
|     AA| 8.386631979187513|
|     EV|10.884270870655678|
|     B6| 13.95852534562212|
|     DL|2.8144726712856043|
|     OO|10.154792043399638|
|     F9|24.103448275862068|
|     US| 5.977315185481719|
|     MQ|19.231592604605904|
|     HA| 4.072423398328691|
|     AS|-1.531766200762389|
|     VX| 5.128571428571429|
|     WN| 3.697840458351697|
+-------+------------------+
```

In [87]:

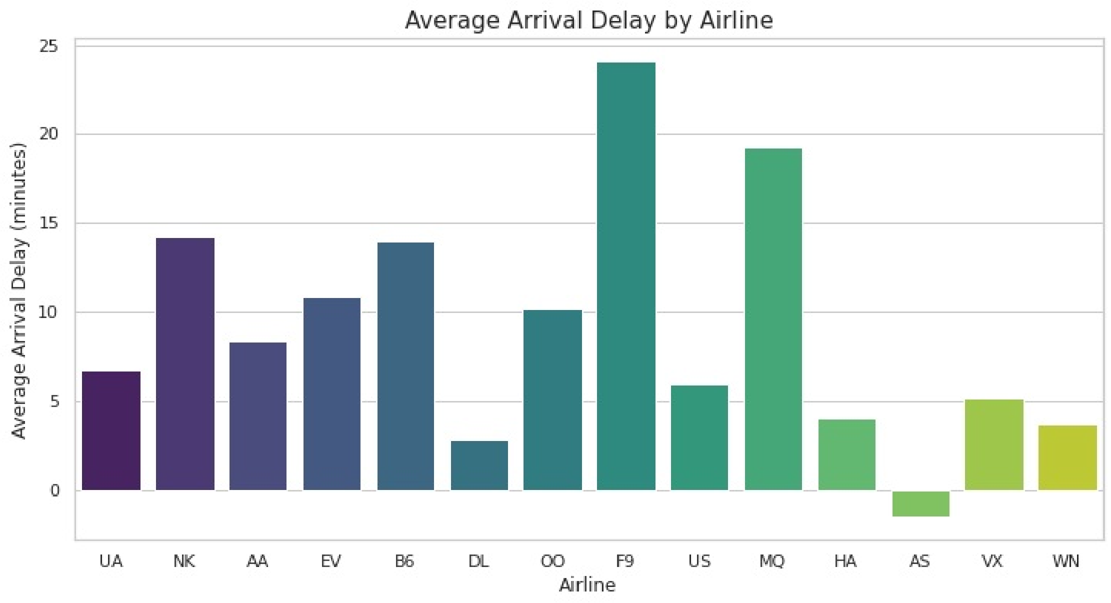
```
#Visualization
import matplotlib.pyplot as plt
import seaborn as sns

avgdelay_pd = avgdelay_df.toPandas()
sns.set(style="whitegrid")

plt.figure(figsize=(12, 6))
sns.barplot(x='AIRLINE', y='AVG_ARRIVAL_DELAY', data=avgdelay_pd, palette="viridis")

plt.title('Average Arrival Delay by Airline', fontsize=15)
plt.xlabel('Airline', fontsize=12)
plt.ylabel('Average Arrival Delay (minutes)', fontsize=12)

plt.show()
```



g) Days of months with respected to average of arrival delays

In [94]:

```
daydl_df = spark.sql('select DAY, AVG(ARRIVAL_DELAY) AVG_ARRIVAL_DELAY from flights_table group by day order by d
daydl_df.show()
```

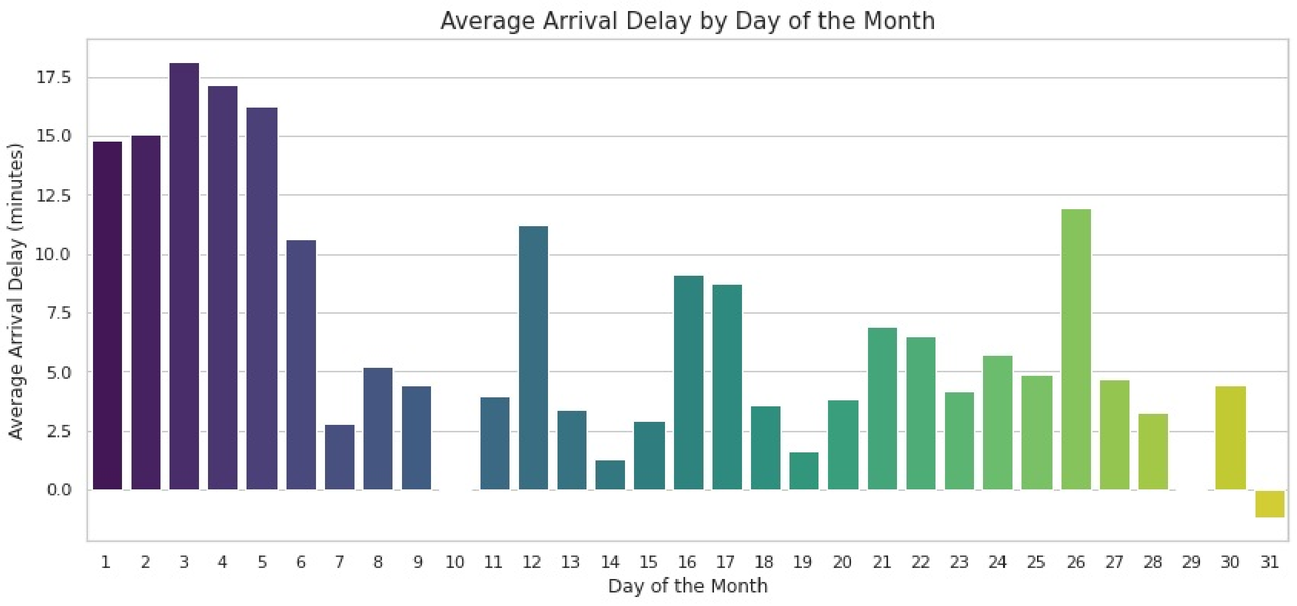
```
+---+--------------------+
|DAY|   AVG_ARRIVAL_DELAY|
+---+--------------------+
|  1|  14.807807807807809|
|  2|  15.046014790468364|
|  3|  18.141541038525965|
|  4|  17.157790927021697|
|  5|   16.23861262014208|
|  6|  10.608832807570979|
|  7|  2.8309417040358746|
|  8|   5.232349165596919|
|  9|   4.421887390959556|
| 10|-0.04705882352941...|
| 11|  3.9912935323383083|
| 12|   11.24892703862661|
| 13|  3.3769751693002257|
| 14|  1.3299319727891157|
| 15|   2.966753585397653|
| 16|   9.124321062160531|
| 17|   8.761435608726249|
| 18|  3.5693430656934306|
| 19|  1.6344282238442822|
| 20|  3.8770149253731345|
| 21|   6.919860627177701|
| 22|   6.550920245398773|
| 23|   4.207086133170434|
| 24|   5.737543859649123|
| 25|   4.903708523096942|
| 26|   11.96778269109286|
| 27|   4.706711409395973|
| 28|   3.257425742574257|
| 29| 0.07971014492753623|
| 30|   4.471478463329452|
| 31|  -1.196594427244582|
+---+--------------------+
```

In [99]:

```
#Visualization
daydl_pd = daydl_df.toPandas()
plt.figure(figsize=(14, 6))
sns.barplot(x='DAY', y='AVG_ARRIVAL_DELAY', data=daydl_pd, palette="viridis")

plt.title('Average Arrival Delay by Day of the Month', fontsize=15)
plt.xlabel('Day of the Month', fontsize=12)
plt.ylabel('Average Arrival Delay (minutes)', fontsize=12)

plt.show()
```



h) Arrange weekdays with respect to the average arrival delays caused

In [100…

```
weekdl_df = spark.sql('select DAY_OF_WEEK, AVG(ARRIVAL_DELAY) AVG_ARRIVAL_DELAY from flights_table \
                      group by day_of_week order by avg_arrival_delay desc')
weekdl_df.show()
```

```
+-----------+------------------+
|DAY_OF_WEEK| AVG_ARRIVAL_DELAY|
+-----------+------------------+
|          1|10.807447207297264|
|          7|10.110840438489646|
|          2| 8.033644102148358|
|          4| 7.174969021065675|
|          5| 6.010538373424971|
|          3| 5.587079407806191|
|          6| 4.888689138576779|
+-----------+------------------+
```

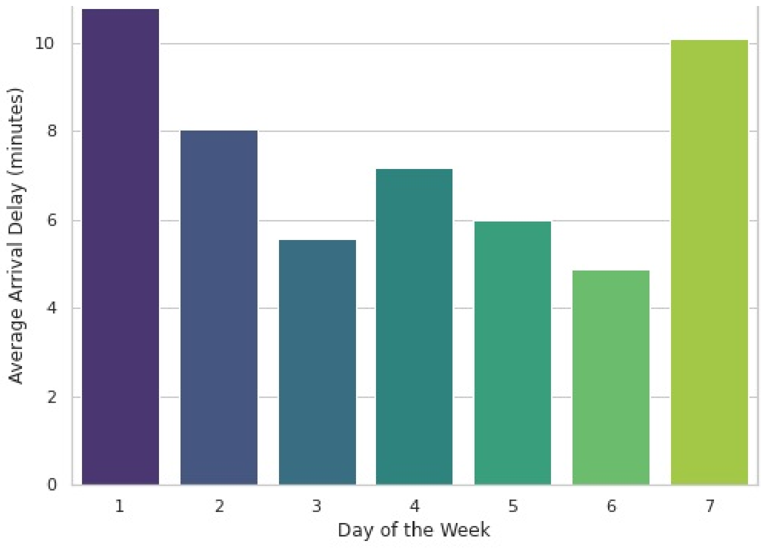
In [103…

```
#Visualization
weekdl_pd = weekdl_df.toPandas()
plt.figure(figsize=(8, 6))
sns.barplot(x='DAY_OF_WEEK', y='AVG_ARRIVAL_DELAY', data=weekdl_pd, palette="viridis")

plt.title('Average Arrival Delay by Day of the Week', fontsize=15)
plt.xlabel('Day of the Week', fontsize=12)
plt.ylabel('Average Arrival Delay (minutes)', fontsize=12)

plt.show()
```

Average Arrival Delay by Day of the Week



i) Arrange Days of month as per cancellations done in Descending

In [116…

```
cancel_df = spark.sql('select DAY, COUNT(*) CANCELLATIONS from flights_table where cancelled=1\
                       group by day order by cancellations desc')
cancel_df.show()
```

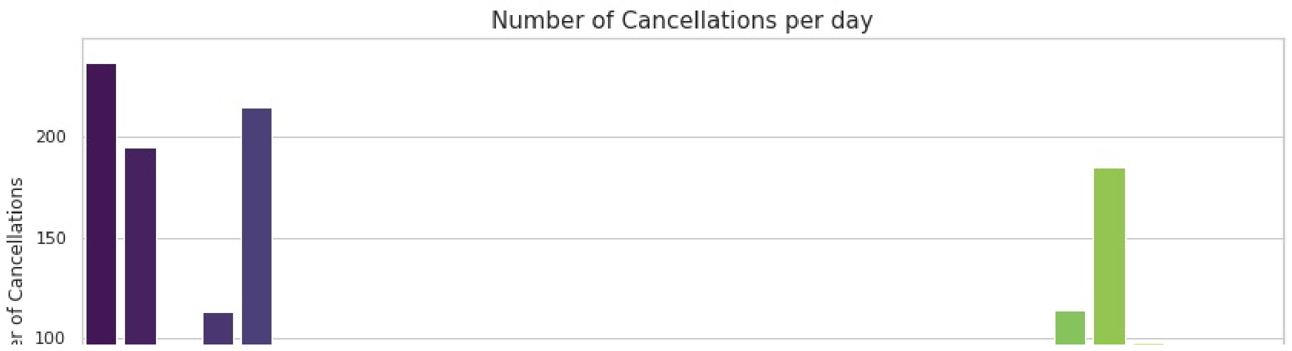
```
+---+-------------+
|DAY|CANCELLATIONS|
+---+-------------+
|  1|          237|
|  5|          215|
|  2|          195|
| 27|          185|
| 26|          114|
|  4|          113|
| 28|           98|
|  9|           89|
|  3|           88|
| 15|           83|
| 23|           69|
| 16|           63|
| 25|           61|
|  8|           61|
| 21|           61|
| 17|           59|
| 24|           57|
|  6|           53|
| 22|           41|
|  7|           31|
+---+-------------+
only showing top 20 rows
```

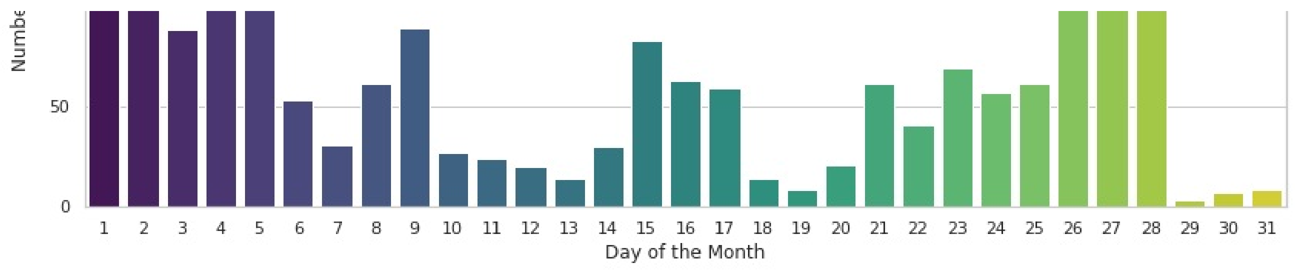
In [118…

```
#Visualization
cancel_pd = cancel_df.toPandas()
plt.figure(figsize=(14, 6))
sns.barplot(x='DAY', y='CANCELLATIONS', data=cancel_pd, palette="viridis")

plt.title('Number of Cancellations per day', fontsize=15)
plt.xlabel('Day of the Month', fontsize=12)
plt.ylabel('Number of Cancellations', fontsize=12)

plt.show()
```





j) Find Top 10 busiest airports with respect to day of week

In [121…

```
topair_df = spark.sql('select ORIGIN_AIRPORT, DAY_OF_WEEK, COUNT(*) TOTAL_FLIGHTS from flights_table \
                       group by origin_airport, day_of_week order by total_flights desc limit 10')
topair_df.show()
```

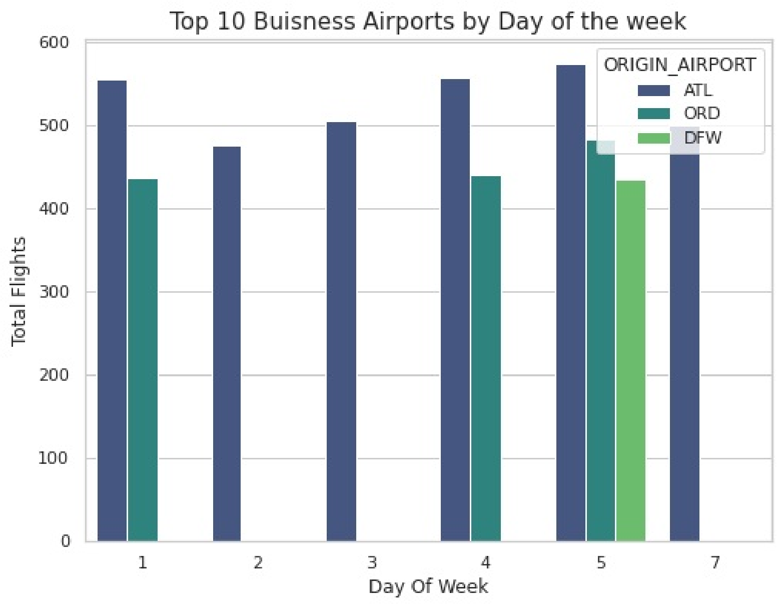
```
+--------------+-----------+-------------+
|ORIGIN_AIRPORT|DAY_OF_WEEK|TOTAL_FLIGHTS|
+--------------+-----------+-------------+
|           ATL|          5|          574|
|           ATL|          4|          556|
|           ATL|          1|          555|
|           ATL|          3|          505|
|           ATL|          7|          499|
|           ORD|          5|          483|
|           ATL|          2|          475|
|           ORD|          4|          441|
|           ORD|          1|          436|
|           DFW|          5|          434|
+--------------+-----------+-------------+
```

In [135…

```
#Visualization
topair_pd = topair_df.toPandas()
plt.figure(figsize=(8, 6))
sns.barplot(x='DAY_OF_WEEK', y='TOTAL_FLIGHTS', hue='ORIGIN_AIRPORT', data=topair_pd, palette="viridis")

plt.title('Top 10 Buisness Airports by Day of the week', fontsize=15)
plt.xlabel('Day Of Week', fontsize=12)
plt.ylabel('Total Flights', fontsize=12)

plt.show()
```



k) Finding airlines that make the maximum number of cancellations

In [140…

```
maxcancel_df = spark.sql('select AIRLINE, COUNT(*) TOTAL_CANCELLATIONS from flights_table where cancelled=1 \
                          group by AIRLINE order by count(*) desc limit 5')
maxcancel_df.show()
```

```
+-------+-------------------+
|AIRLINE|TOTAL_CANCELLATIONS|
+-------+-------------------+
|     MQ|                414|
|     WN|                358|
|     EV|                312|
|     AA|                241|
|     DL|                177|
+-------+-------------------+
```

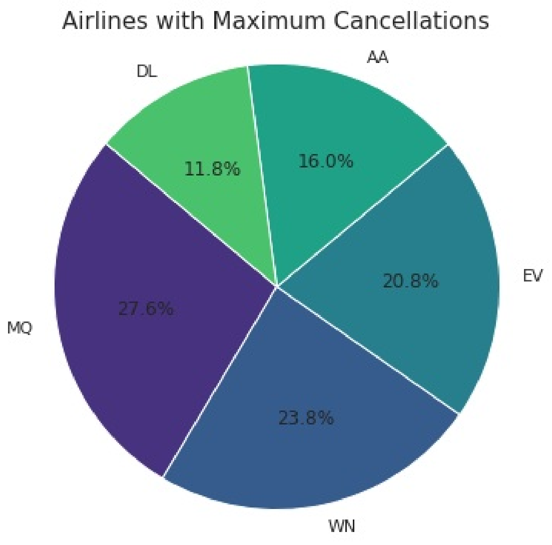
In [144…

```
#Visualization
maxcancel_pd = maxcancel_df.toPandas()

plt.figure(figsize=(8, 6))
plt.pie(maxcancel_pd['TOTAL_CANCELLATIONS'], labels=maxcancel_pd['AIRLINE'], autopct='%1.1f%%', startangle=140, c

plt.title('Airlines with Maximum Cancellations', fontsize=15)
plt.axis('equal')

plt.show()
```



I) Find and order airlines in descending that make the most number of diversions

In [145…

```
maxdiv_df = spark.sql('select AIRLINE, COUNT(*) TOTAL_DIVERSIONS from flights_table where diverted=1 \
                          group by AIRLINE order by TOTAL_DIVERSIONS desc limit 5')
maxdiv_df.show()
```

```
+-------+----------------+
|AIRLINE|TOTAL_DIVERSIONS|
+-------+----------------+
|     WN|              35|
|     OO|              25|
|     EV|              22|
|     DL|              18|
|     B6|              16|
+-------+----------------+
```

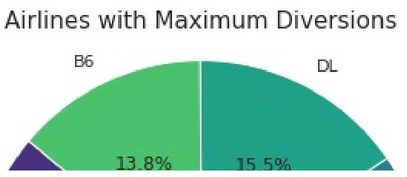
In [146…

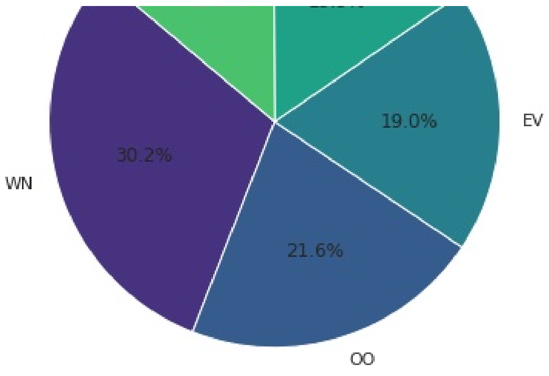
```
#Visualization
maxdiv_pd = maxdiv_df.toPandas()

plt.figure(figsize=(8, 6))
plt.pie(maxdiv_pd['TOTAL_DIVERSIONS'], labels=maxdiv_pd['AIRLINE'], autopct='%1.1f%%', startangle=140, colors=sns

plt.title('Airlines with Maximum Diversions', fontsize=15)
plt.axis('equal')

plt.show()
```





m) Finding days of month that see the most number of diversion

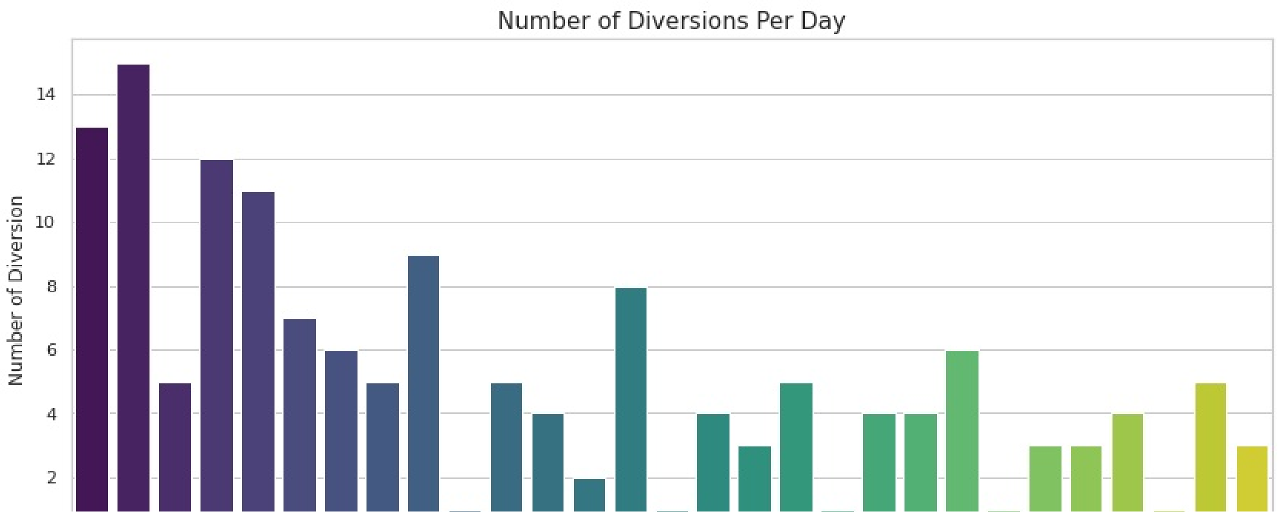
```
In [148…
daydiv_df = spark.sql('select DAY, COUNT(*) DIVERSION_COUNT from flights_table where diverted=1\
                       group by day order by count(*) desc')
daydiv_df.show()
```

```
+---+---------------+
|DAY|DIVERSION_COUNT|
+---+---------------+
|  2|             15|
|  1|             13|
|  4|             12|
|  5|             11|
|  9|              9|
| 14|              8|
|  6|              7|
| 23|              6|
|  7|              6|
|  3|              5|
|  8|              5|
| 30|              5|
| 11|              5|
| 18|              5|
| 28|              4|
| 12|              4|
| 20|              4|
| 16|              4|
| 21|              4|
| 26|              3|
+---+---------------+
only showing top 20 rows
```

```
In [149…
#Visualization
daydiv_pd = daydiv_df.toPandas()
plt.figure(figsize=(14, 6))
sns.barplot(x='DAY', y='DIVERSION_COUNT', data=daydiv_pd, palette="viridis")

plt.title('Number of Diversions Per Day', fontsize=15)
plt.xlabel('Day of the Month', fontsize=12)
plt.ylabel('Number of Diversion', fontsize=12)

plt.show()
```



n) Calculating mean and standard deviation of departure delay for all flights in minutes

In [154...

```
spark.sql('select round(mean(DEPARTURE_DELAY),2) MEAN, round(std(DEPARTURE_DELAY),2) STD_DEVIATION from \
           flights_table').show()
```

```
+-----+-------------+
| MEAN|STD_DEVIATION|
+-----+-------------+
|11.33|        39.62|
+-----+-------------+
```

o) Calculating mean and standard deviation of arrival delay for all flights in minutes

In [155...

```
spark.sql('select round(mean(ARRIVAL_DELAY),2) MEAN, round(std(ARRIVAL_DELAY),2) STD_DEVIATION from \
           flights_table').show()
```

```
+----+-------------+
|MEAN|STD_DEVIATION|
+----+-------------+
|7.55|        42.38|
+----+-------------+
```

p) Finding all diverted Route from a source to destination Airport & which route is the most diverted

In [157...

```
spark.sql('select ORIGIN_AIRPORT, DESTINATION_AIRPORT, COUNT(*) NUMBER_OF_DIVERSIONS from flights_table \
          where diverted=1 group by origin_airport, destination_airport order by count(*) desc').show()
```

```
+--------------+-------------------+--------------------+
|ORIGIN_AIRPORT|DESTINATION_AIRPORT|NUMBER_OF_DIVERSIONS|
+--------------+-------------------+--------------------+
|           HOU|                DAL|                   2|
|           PHL|                SAN|                   2|
|           STT|                PHL|                   2|
|           TPA|                LGA|                   2|
|           IAH|                ASE|                   2|
|           JFK|                EGE|                   2|
|           JFK|                SEA|                   2|
|           ORD|                ASE|                   2|
|           CLT|                IAH|                   2|
|           EWR|                STL|                   1|
|           SNA|                SFO|                   1|
|           FLL|                PVD|                   1|
|           CLT|                MIA|                   1|
|           FLL|                BWI|                   1|
|           BOS|                LAS|                   1|
|           ATL|                GTR|                   1|
|           COS|                ORD|                   1|
|           KOA|                SFO|                   1|
|           SLC|                RDM|                   1|
|           ATL|                LGA|                   1|
+--------------+-------------------+--------------------+
only showing top 20 rows
```

q) Finding AIRLINES with its total flight count, total number of flights arrival delayed by more than 30 Minutes, % of such flights delayed by more than 30 minutes when it is not Weekends with minimum count of flights from Airlines by more than 10. Also Exclude some of Airlines 'AK', 'HI', 'PR', 'VI' and arrange output in descending order by % of such count of flights.

In [163...

```
spark.sql("select AIRLINE, count(*) as Total_Flight_Count, sum(case when ARRIVAL_DELAY > 30 then 1 else 0 end) \
           as Delayed_Flight_Count, round(100 * sum(case when ARRIVAL_DELAY > 30 and DAY_OF_WEEK not in (6,7) \
           then 1 else 0 end)/count(*),2) as PDelay FROM flights_table WHERE AIRLINE NOT IN \
           ('AK', 'HI', 'PR', 'VI') GROUP BY AIRLINE HAVING COUNT(*) > 10 ORDER BY PDelay DESC").show()
```

```
+-------+------------------+--------------------+------+
|AIRLINE|Total_Flight_Count|Delayed_Flight_Count|PDelay|
+-------+------------------+--------------------+------+
|     F9|               794|                 198| 17.51|
|     MQ|              3502|                 775| 17.16|
|     B6|              2548|                 485| 14.13|
|     NK|              1048|                 186| 13.26|
|     EV|              5916|                 874| 11.24|
|     OO|              5708|                 859| 11.09|
|     UA|              4701|                 653| 10.57|
|     AA|              5250|                 700|  9.22|
|     VX|               573|                  67|   8.2|
|     US|              3925|                 452|   7.9|
|     DL|              7989|                 746|  7.41|
|     WN|             11738|                1235|   7.4|
|     AS|              1586|                 100|  4.04|
|     HA|               722|                  38|  3.19|
+-------+------------------+--------------------+------+
```

r) Finding AIRLINES with its total flight count with total number of flights departure delayed by less than 30 Minutes, % of such flights delayed by less than 30 minutes when it is Weekends with minimum count of flights from Airlines by more than 10. Also Exclude some of Airlines 'AK', 'HI', 'PR', 'VI' and arrange output in descending order by % of such count of flights.

In [165…

```
spark.sql("select AIRLINE, count(*) as Total_Flight_Count, sum(case when DEPARTURE_DELAY < 30 then 1 else 0 end)
           as Delayed_Flight_Count, round(100 * sum(case when  DEPARTURE_DELAY > 30 and DAY_OF_WEEK >5 then 1 els
           end)/count(*),2) as PDelay FROM flights_table WHERE AIRLINE NOT IN ('AK', 'HI', 'PR', 'VI') GROUP BY \
           AIRLINE HAVING COUNT(*) > 10 ORDER BY PDelay DESC").show()
```

```
+-------+------------------+--------------------+------+
|AIRLINE|Total_Flight_Count|Delayed_Flight_Count|PDelay|
+-------+------------------+--------------------+------+
|     F9|               794|                 585|  7.18|
|     B6|              2548|                1947|  4.75|
|     NK|              1048|                 839|  4.48|
|     MQ|              3502|                2443|  4.11|
|     AA|              5250|                4342|  3.85|
|     OO|              5708|                4736|  3.59|
|     UA|              4701|                3903|  3.47|
|     WN|             11738|                9945|  3.37|
|     VX|               573|                 490|  3.32|
|     US|              3925|                3356|  3.24|
|     EV|              5916|                4819|  3.06|
|     AS|              1586|                1468|  2.08|
|     DL|              7989|                7010|  2.04|
|     HA|               722|                 692|  1.66|
+-------+------------------+--------------------+------+
```

s) When is the best time of day/day of week/time of a year to fly with minimum delays?

In [182…

```
spark.sql('select DAY DAY_OF_MONTH, AVG(COALESCE(ARRIVAL_DELAY, 0) + COALESCE(DEPARTURE_DELAY, 0)) TOTAL_DELAY \
           from flights_table group by day order by total_delay LIMIT 5').show()

spark.sql('select DAY_OF_WEEK, AVG(COALESCE(ARRIVAL_DELAY, 0) + COALESCE(DEPARTURE_DELAY, 0)) TOTAL_DELAY \
           from flights_table group by day_of_week order by total_delay LIMIT 5').show()
```

```
+------------+-----------------+
|DAY_OF_MONTH|      TOTAL_DELAY|
+------------+-----------------+
|          31| 4.09726443768997|
|          10|4.924986210700497|
|          29|4.960336538461538|
|          14|6.968832891246684|
|          19|8.237001209189843|
+------------+-----------------+

+-----------+------------------+
|DAY_OF_WEEK|       TOTAL_DELAY|
+-----------+------------------+
|          6|14.154561301568855|
|          3|14.233472149921916|
|          5|15.901192887688499|
|          4|17.397617629541394|
|          2| 18.47514450867052|
+-----------+------------------+
```

t) Which airlines are best airline to travel considering number of cancellations, arrival, departure delays and all reasons affecting performance of airline industry.

In [175…

```
spark.sql("Select AIRLINE from (SELECT AIRLINE, COUNT(*) AS total_flights, SUM(CANCELLED) AS total_cancellations,
        DEPARTURE_DELAY > 0 THEN DEPARTURE_DELAY ELSE NULL END) AS avg_departure_delay, AVG(CASE WHEN \
        ARRIVAL_DELAY > 0 THEN ARRIVAL_DELAY ELSE NULL END) AS avg_arrival_delay, AVG(CASE WHEN AIR_SYSTEM_DELAY
        THEN AIR_SYSTEM_DELAY ELSE NULL END) AS avg_air_system_delay, AVG(CASE WHEN SECURITY_DELAY > 0 THEN \
        SECURITY_DELAY ELSE NULL END) AS avg_security_delay, AVG(CASE WHEN AIRLINE_DELAY > 0 THEN AIRLINE_DELAY \
        ELSE NULL END)AS avg_airline_delay, AVG(CASE WHEN LATE_AIRCRAFT_DELAY > 0 THEN LATE_AIRCRAFT_DELAY ELSE \
        NULL END) AS avg_late_aircraft_delay, AVG(CASE WHEN WEATHER_DELAY > 0 THEN WEATHER_DELAY ELSE NULL END) \
        AS avg_weather_delay FROM flights_table GROUP BY AIRLINE limit 5)").show()
```

```
+-------+
|AIRLINE|
+-------+
|     UA|
|     NK|
|     AA|
|     EV|
|     B6|
+-------+
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js