Assignment - PYSPARK

Affan Mohammed N Marikar
281911

Questions:

a) Create a new Spark Session with new SparkConfig

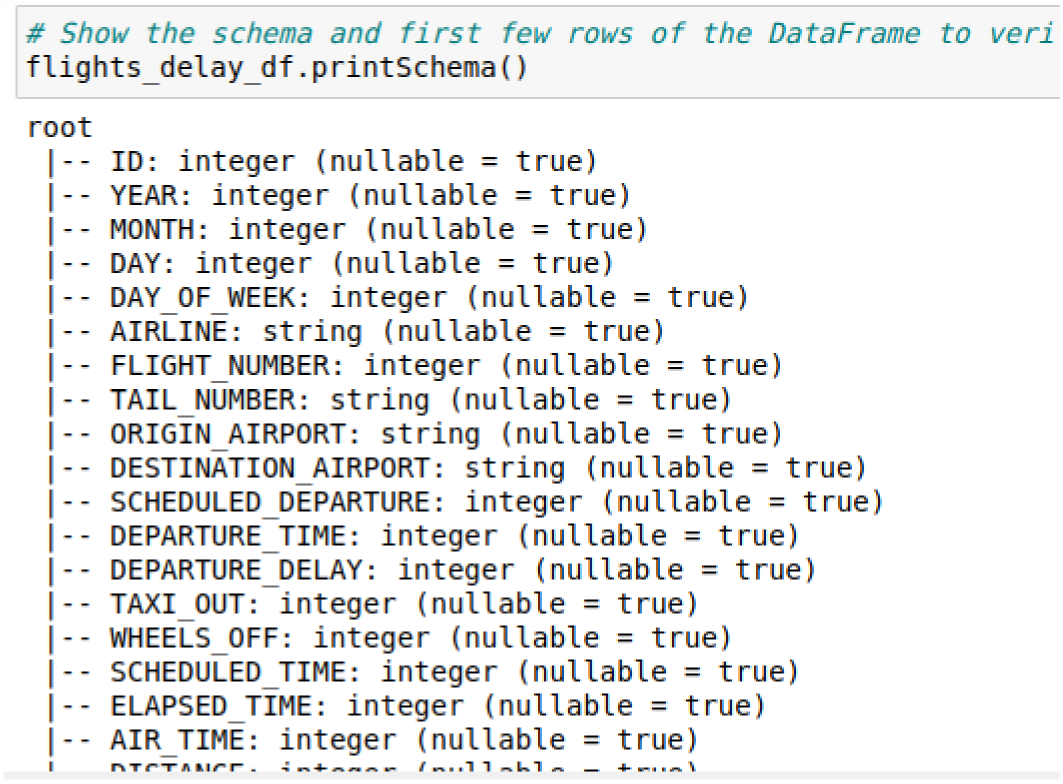
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession

config = SparkConf().setAppName("test_Session").setMaster("local[2]")
sc = SparkContext(conf = config)
spark = SparkSession.builder.appName('PySparkSession').getOrCreate()

sc
spark

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

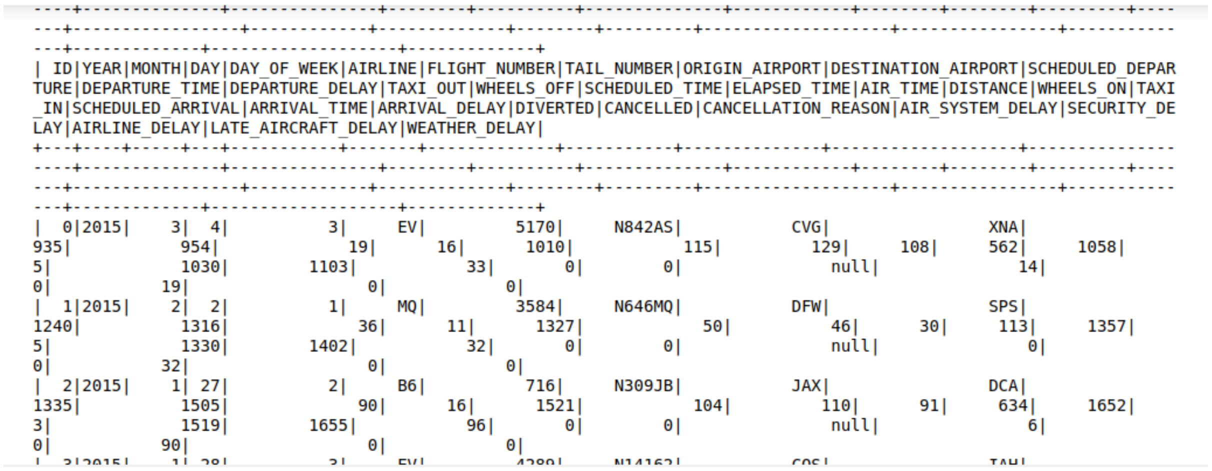
# Read the CSV file into a DataFrame
flights_delay_df = spark.read.csv("file:///home/hadoop/Downloads/Flights_Delay.csv", header=True, inferSchema=True)
Show the schema and first few rows of the DataFrame to verify
flights_delay_df.printSchema()



```
# Show the schema and first few rows of the DataFrame to veri
flights_delay_df.printSchema()
```

```
root
 |-- ID: integer (nullable = true)
 |-- YEAR: integer (nullable = true)
 |-- MONTH: integer (nullable = true)
 |-- DAY: integer (nullable = true)
 |-- DAY_OF_WEEK: integer (nullable = true)
 |-- AIRLINE: string (nullable = true)
 |-- FLIGHT_NUMBER: integer (nullable = true)
 |-- TAIL_NUMBER: string (nullable = true)
 |-- ORIGIN_AIRPORT: string (nullable = true)
 |-- DESTINATION_AIRPORT: string (nullable = true)
 |-- SCHEDULED_DEPARTURE: integer (nullable = true)
 |-- DEPARTURE_TIME: integer (nullable = true)
 |-- DEPARTURE_DELAY: integer (nullable = true)
 |-- TAXI_OUT: integer (nullable = true)
 |-- WHEELS_OFF: integer (nullable = true)
 |-- SCHEDULED_TIME: integer (nullable = true)
 |-- ELAPSED_TIME: integer (nullable = true)
 |-- AIR_TIME: integer (nullable = true)
 |-- DISTANCE: integer (nullable = true)
```

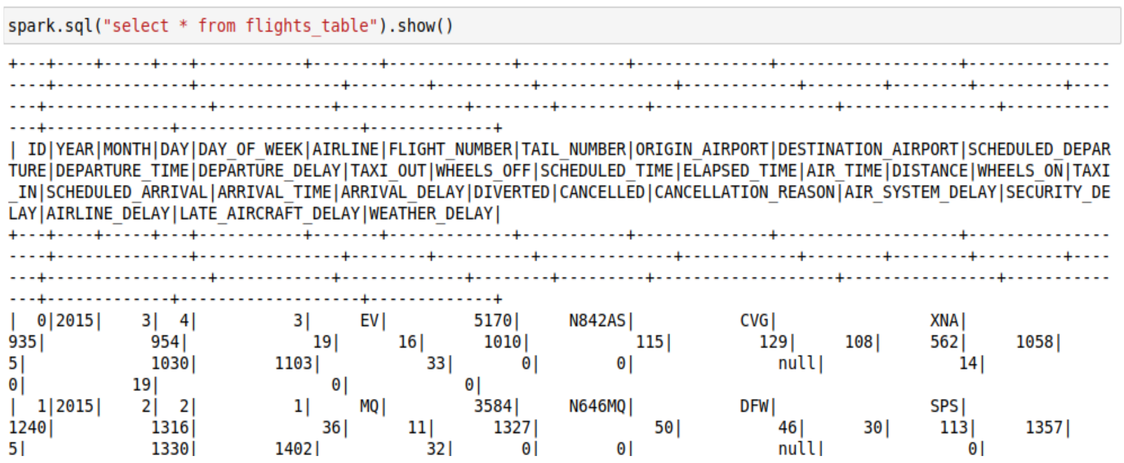
flights_delay_df.show(5)



```
+---+----+-----+---+-----------+-------+-------------+-----------+--------------+-------------------+---------------
----+--------------+---------------+--------+----------+--------------+------------+--------+--------+---------+----
---+-----------------+------------+-------------+--------+---------+-------------------+----------------+-----------
---+-------------+-------------------+-------------+
| ID|YEAR|MONTH|DAY|DAY_OF_WEEK|AIRLINE|FLIGHT_NUMBER|TAIL_NUMBER|ORIGIN_AIRPORT|DESTINATION_AIRPORT|SCHEDULED_DEPAR
TURE|DEPARTURE_TIME|DEPARTURE_DELAY|TAXI_OUT|WHEELS_OFF|SCHEDULED_TIME|ELAPSED_TIME|AIR_TIME|DISTANCE|WHEELS_ON|TAXI
_IN|SCHEDULED_ARRIVAL|ARRIVAL_TIME|ARRIVAL_DELAY|DIVERTED|CANCELLED|CANCELLATION_REASON|AIR_SYSTEM_DELAY|SECURITY_DE
LAY|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|
```

c) Create table Spark HIVE table flights_table

flights_delay_updated_df.createOrReplaceTempView("flights_table")

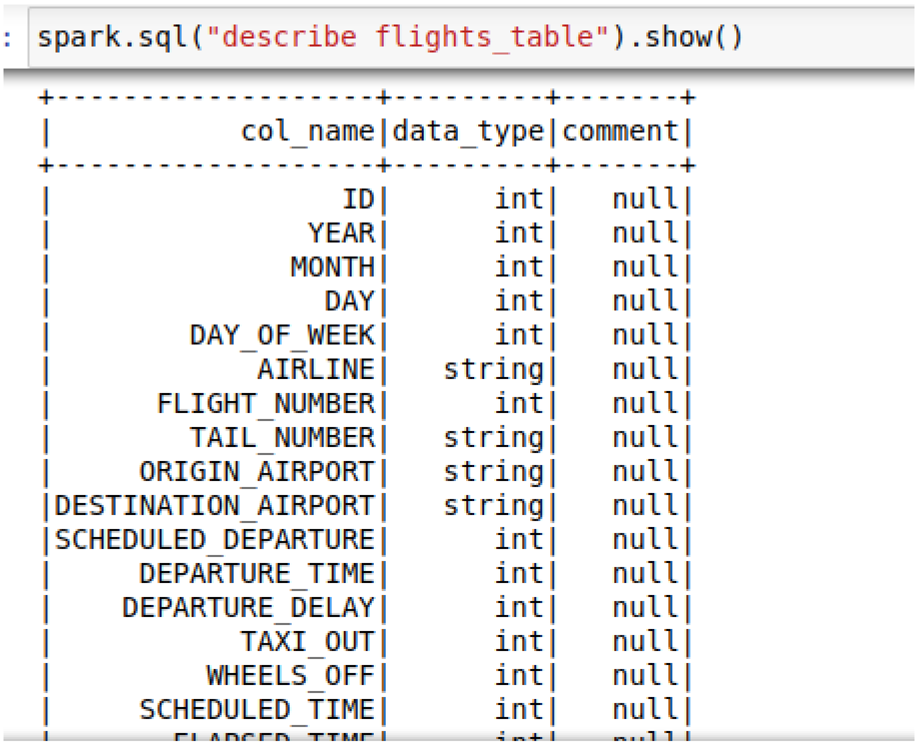


```
spark.sql("select * from flights_table").show()
```

```
+---+----+-----+---+-----------+-------+-------------+-----------+--------------+-------------------+---------------
----+--------------+---------------+--------+----------+--------------+------------+--------+--------+---------+----
---+-----------------+------------+-------------+--------+---------+-------------------+----------------+-----------
---+-------------+-------------------+-------------+
| ID|YEAR|MONTH|DAY|DAY_OF_WEEK|AIRLINE|FLIGHT_NUMBER|TAIL_NUMBER|ORIGIN_AIRPORT|DESTINATION_AIRPORT|SCHEDULED_DEPAR
TURE|DEPARTURE_TIME|DEPARTURE_DELAY|TAXI_OUT|WHEELS_OFF|SCHEDULED_TIME|ELAPSED_TIME|AIR_TIME|DISTANCE|WHEELS_ON|TAXI
_IN|SCHEDULED_ARRIVAL|ARRIVAL_TIME|ARRIVAL_DELAY|DIVERTED|CANCELLED|CANCELLATION_REASON|AIR_SYSTEM_DELAY|SECURITY_DE
LAY|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|
```

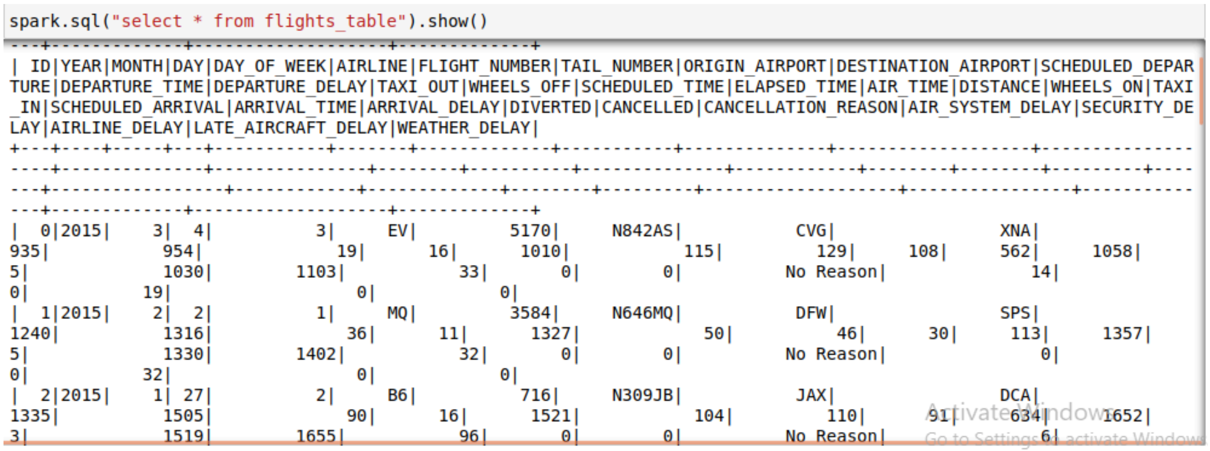
d) Describe the table schema & show top 10 rows of Dataset

spark.sql("describe flights_table").show()



```
: 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|
```

spark.sql("select * from flights_table limit 10").show()



```
spark.sql("select * from flights_table").show()

---+-------------+------------------+-------------+
| ID|YEAR|MONTH|DAY|DAY_OF_WEEK|AIRLINE|FLIGHT_NUMBER|TAIL_NUMBER|ORIGIN_AIRPORT|DESTINATION_AIRPORT|SCHEDULED_DEPAR
TURE|DEPARTURE_TIME|DEPARTURE_DELAY|TAXI_OUT|WHEELS_OFF|SCHEDULED_TIME|ELAPSED_TIME|AIR_TIME|DISTANCE|WHEELS_ON|TAXI
_IN|SCHEDULED_ARRIVAL|ARRIVAL_TIME|ARRIVAL_DELAY|DIVERTED|CANCELLED|CANCELLATION_REASON|AIR_SYSTEM_DELAY|SECURITY_DE
LAY|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|         No Reason|              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|         No Reason|               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|         No Reason|               6|
```

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

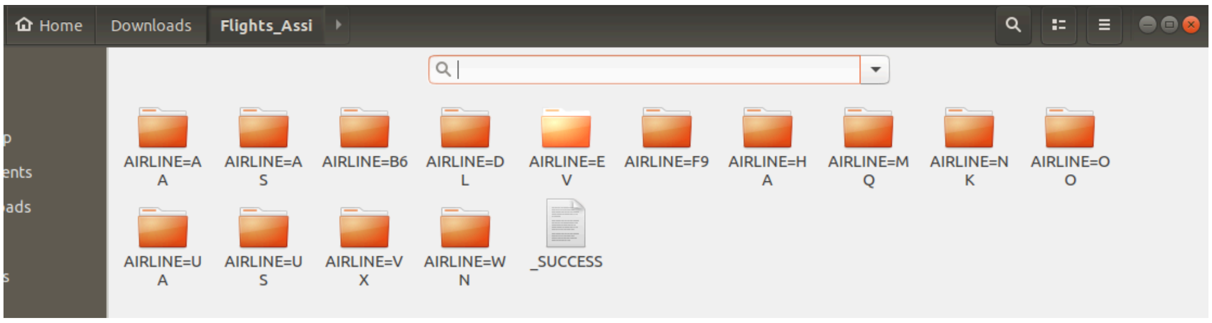
# Optimise by partitioning the DataFrame by a specific column (here; Airline)

partitioned_df = flights_delay_updated_df.repartition("AIRLINE")

# Save the DataFrame as Parquet files for better compression and faster queries

parquet_path = "file:///home/hadoop/Downloads/Flights_Assi"

partitioned_df.write.mode("overwrite").partitionBy("AIRLINE").parquet(parquet_path)



# Load the Parquet files into a DataFrame

flight_parq_df = spark.read.parquet(parquet_path)

# Create a Hive table using the optimised Parquet data

flight_parq_df.createOrReplaceTempView("flight_parq_hivetable")

spark.sql("CREATE DATABASE IF NOT EXISTS flights_db")

spark.sql("USE flights_db")

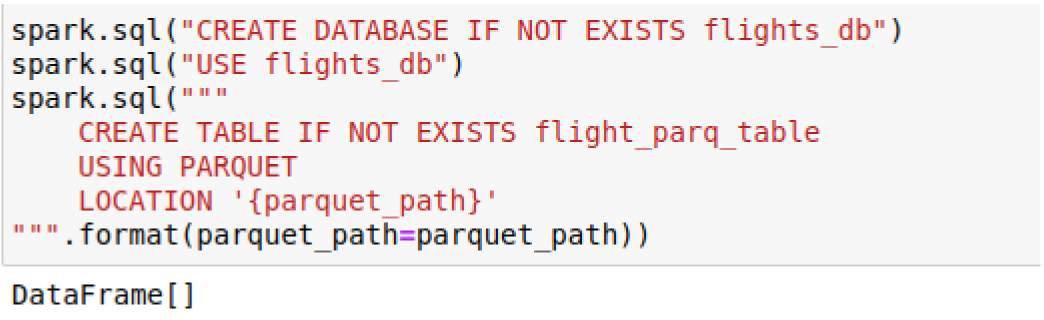
spark.sql("""

CREATE TABLE IF NOT EXISTS flight_parq_table

USING PARQUET

LOCATION '{parquet_path}'

""".format(parquet_path=parquet_path))

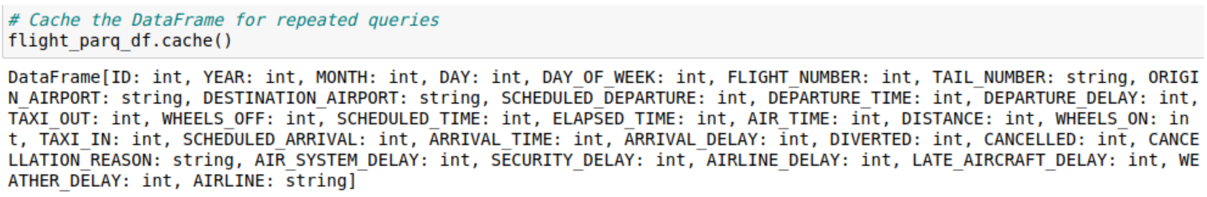


```
spark.sql("CREATE DATABASE IF NOT EXISTS flights_db")
spark.sql("USE flights_db")
spark.sql("""
    CREATE TABLE IF NOT EXISTS flight_parq_table
    USING PARQUET
    LOCATION '{parquet_path}'
""".format(parquet_path=parquet_path))
```

```
DataFrame[]
```

Cache the DataFrame for repeated queries

flight_parq_df.cache()



```
# Cache the DataFrame for repeated queries
flight_parq_df.cache()
```

```
DataFrame[ID: int, YEAR: int, MONTH: int, DAY: int, DAY_OF_WEEK: int, FLIGHT_NUMBER: int, TAIL_NUMBER: string, ORIGI
N_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, DISTANCE: int, WHEELS_ON: in
t, TAXI_IN: int, SCHEDULED_ARRIVAL: int, ARRIVAL_TIME: int, ARRIVAL_DELAY: int, DIVERTED: int, CANCELLED: int, CANCE
LLATION_REASON: string, AIR_SYSTEM_DELAY: int, SECURITY_DELAY: int, AIRLINE_DELAY: int, LATE_AIRCRAFT_DELAY: int, WE
ATHER_DELAY: int, AIRLINE: string]
```

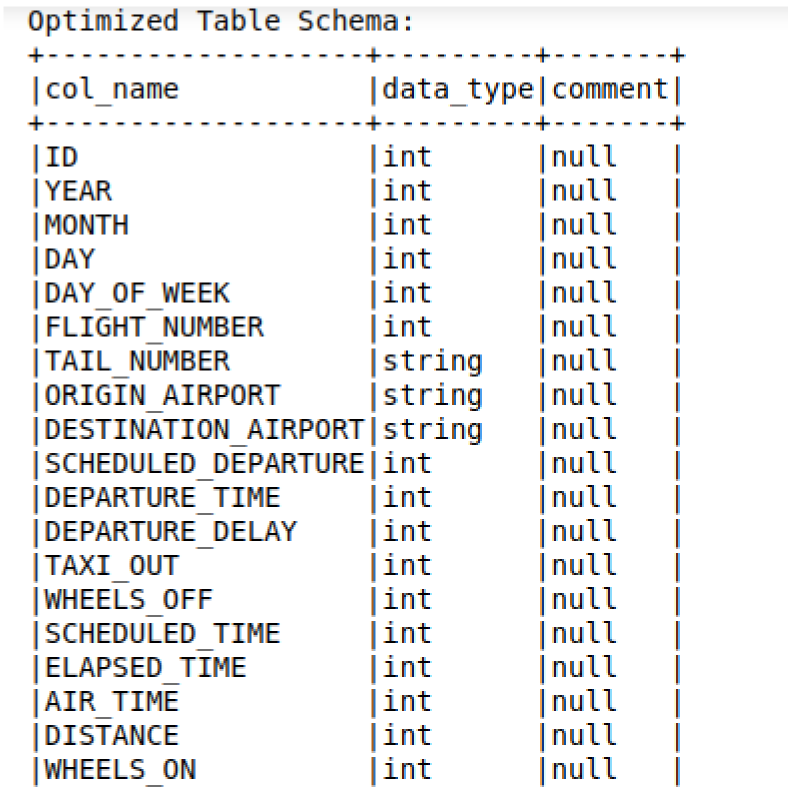
# Enable predicate pushdown by filtering the DataFrame by filter data for ORIGIN_AIRPORT

filtered_df = flight_parq_df.filter("ORIGIN_AIRPORT = 'CVG'")

# Describe the optimised table schema

print("Optimised Table Schema:")

spark.sql("DESCRIBE flight_parq_table").show(truncate=False)

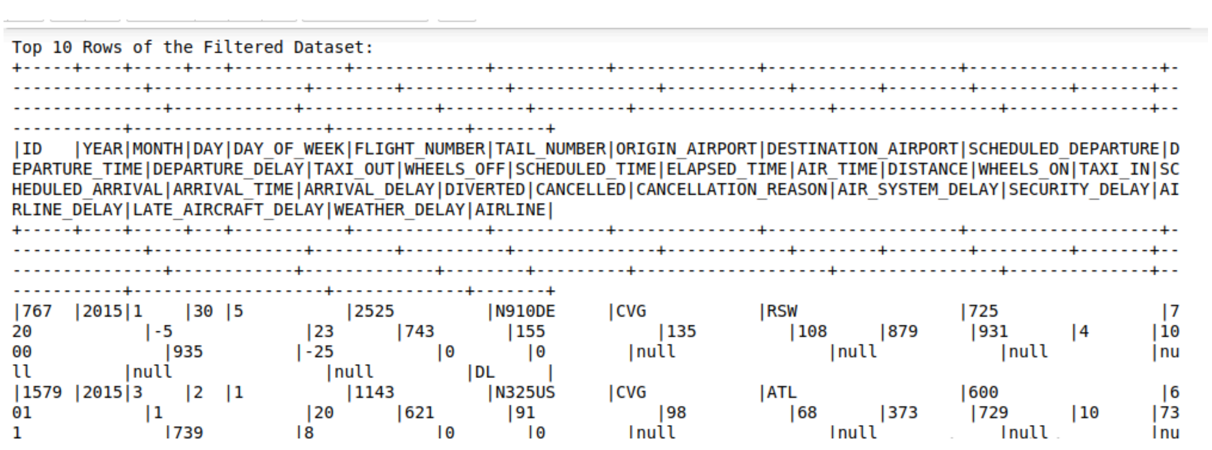


```
Optimized Table Schema:
+-------------------+---------+-------+
|col_name           |data_type|comment|
+-------------------+---------+-------+
|ID                 |int      |null   |
|YEAR               |int      |null   |
|MONTH              |int      |null   |
|DAY                |int      |null   |
|DAY_OF_WEEK        |int      |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   |
```

# Show the top 10 rows of the filtered dataset

print("Top 10 Rows of the Filtered Dataset:")

filtered_df.show(10, truncate=False)



```
Top 10 Rows of the Filtered Dataset:
+-----+----+-----+---+-----------+-------------+-----------+--------------+-------------------+-------------------+-
------------+---------------+--------+----------+--------------+------------+--------+--------+---------+-------+--
---------------+------------+-------------+--------+---------+--------------------+-----------------+--------------+--
----------+-------------------+-------------+-------+
|ID   |YEAR|MONTH|DAY|DAY_OF_WEEK|FLIGHT_NUMBER|TAIL_NUMBER|ORIGIN_AIRPORT|DESTINATION_AIRPORT|SCHEDULED_DEPARTURE|D
EPARTURE_TIME|DEPARTURE_DELAY|TAXI_OUT|WHEELS_OFF|SCHEDULED_TIME|ELAPSED_TIME|AIR_TIME|DISTANCE|WHEELS_ON|TAXI_IN|SC
HEDULED_ARRIVAL|ARRIVAL_TIME|ARRIVAL_DELAY|DIVERTED|CANCELLED|CANCELLATION_REASON|AIR_SYSTEM_DELAY|SECURITY_DELAY|AI
RLINE_DELAY|LATE_AIRCRAFT_DELAY|WEATHER_DELAY|AIRLINE|
+-----+----+-----+---+-----------+-------------+-----------+--------------+-------------------+-------------------+-
------------+---------------+--------+----------+--------------+------------+--------+--------+---------+-------+--
---------------+------------+-------------+--------+---------+--------------------+-----------------+--------------+--
----------+-------------------+-------------+-------+
|767  |2015|1    |30 |5          |2525         |N910DE     |CVG           |RSW                |725                |7
20          |-5              |23      |743       |155           |135          |108     |879     |931      |4       |10
00             |935          |-25          |0       |0         |null                |null             |null          |nu
ll        |null                |null          |DL     |
|1579 |2015|3    |2  |1          |1143         |N325US     |CVG           |ATL                |600                |6
01          |1               |20      |621       |91            |98           |68      |373     |729      |10      |73
1              |739          |8            |0       |0         |null                |null             |null          |nu
```

f) Average arrival delay caused by airlines

avg_arrival_delay_airlines = spark.sql("""

SELECT AIRLINE, round(AVG(CASE WHEN ARRIVAL_DELAY > 0 THEN ARRIVAL_DELAY ELSE NULL END),2) AS avg_arrival_delay

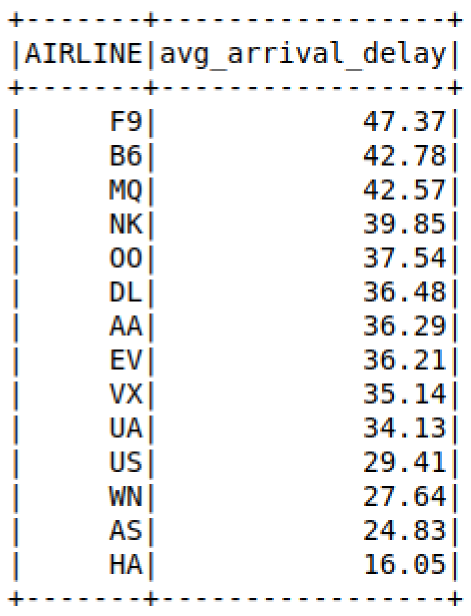
FROM flights_table

GROUP BY AIRLINE

ORDER BY avg_arrival_delay DESC

""")

avg_arrival_delay_airlines.show()

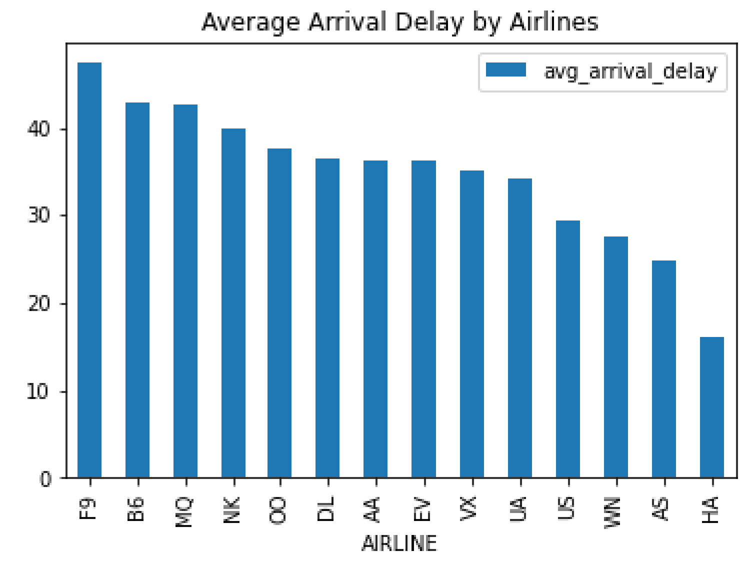


```
+-------+-----------------+
|AIRLINE|avg_arrival_delay|
+-------+-----------------+
|     F9|            47.37|
|     B6|            42.78|
|     MQ|            42.57|
|     NK|            39.85|
|     OO|            37.54|
|     DL|            36.48|
|     AA|            36.29|
|     EV|            36.21|
|     VX|            35.14|
|     UA|            34.13|
|     US|            29.41|
|     WN|            27.64|
|     AS|            24.83|
|     HA|            16.05|
+-------+-----------------+
```

# Visualisation

avg_arrival_delay_pandas = avg_arrival_delay_airlines.toPandas()

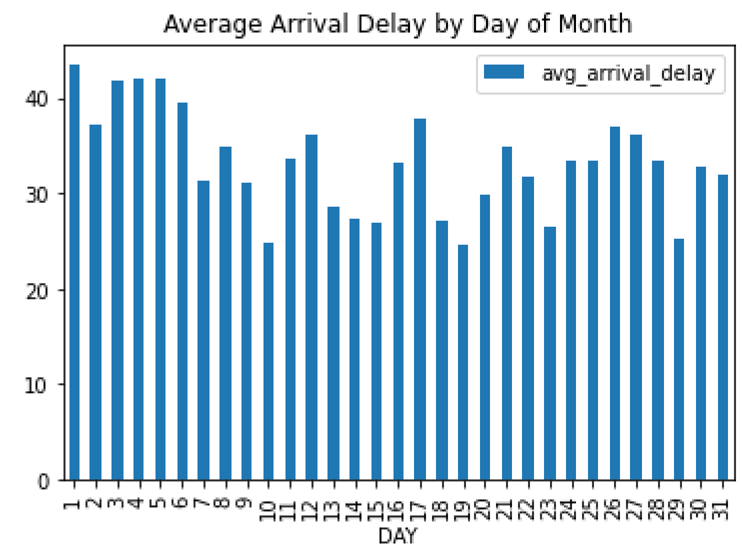
avg_arrival_delay_pandas.plot(kind='bar', x='AIRLINE', y='avg_arrival_delay', title='Average Arrival Delay by Airlines')



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

avg_arrival_delay_days = spark.sql("""
SELECT DAY, round(AVG(CASE WHEN ARRIVAL_DELAY > 0 THEN ARRIVAL_DELAY ELSE NULL END),2) AS avg_arrival_delay
FROM flights_table
GROUP BY DAY
ORDER BY DAY
""")
avg_arrival_delay_days.show(31)

```
+---+-----------------+
|DAY|avg_arrival_delay|
+---+-----------------+
|  1|            43.55|| 17|            37.84|
|  2|            37.13|| 18|            27.11|
|  3|            41.83|| 19|             24.6|
|  4|            42.05|| 20|            29.92|
|  5|            42.09|| 21|            34.86|
|  6|            39.52|| 22|            31.69|
|  7|            31.25|| 23|            26.55|
|  8|            34.96|| 24|            33.48|
|  9|            31.23|| 25|            33.48|
| 10|            24.73|| 26|            37.02|
| 11|            33.65|| 27|            36.13|
| 12|            36.25|| 28|            33.38|
| 13|            28.53|| 29|            25.16|
| 14|            27.39|| 30|            32.82|
| 15|            26.97|| 31|            31.97|
| 16|            33.15|+---+-----------------+
# Visualisation
avg_arrival_delay_days_df = avg_arrival_delay_days.toPandas()
avg_arrival_delay_days_df.plot(kind='bar', x='DAY', y='avg_arrival_delay', title='Average
Arrival Delay by Day of Month')
```



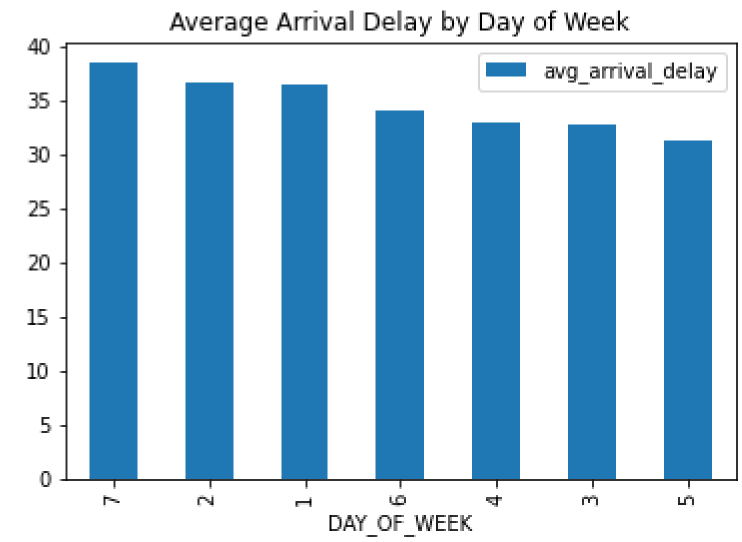
h) Arrange weekdays with respect to the average arrival delays caused

```
avg_arrival_delay_weekdays = spark.sql("""
            SELECT DAY_OF_WEEK, round(AVG(CASE WHEN ARRIVAL_DELAY > 0 THEN
ARRIVAL_DELAY ELSE NULL END),2) AS avg_arrival_delay
            FROM flights_table
            GROUP BY DAY_OF_WEEK
            ORDER BY avg_arrival_delay DESC
""")
avg_arrival_delay_weekdays.show()
```

```
+-----------+-----------------+
|DAY_OF_WEEK|avg_arrival_delay|
+-----------+-----------------+
|          7|            38.42|
|          2|            36.64|
|          1|            36.38|
|          6|            34.05|
|          4|            32.85|
|          3|            32.78|
|          5|            31.19|
+-----------+-----------------+
```

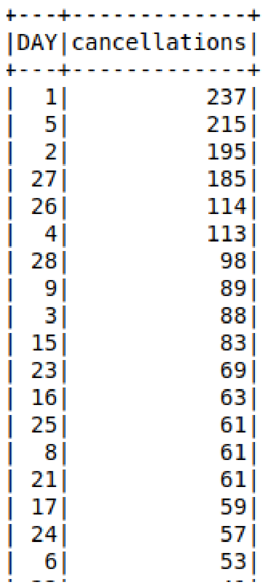
```
# Visualisation
avg_delay_weekdays_df = avg_arrival_delay_weekdays.toPandas()
```

avg_delay_weekdays_df.plot(kind='bar', x='DAY_OF_WEEK', y='avg_arrival_delay', title='Average Arrival Delay by Day of Week')

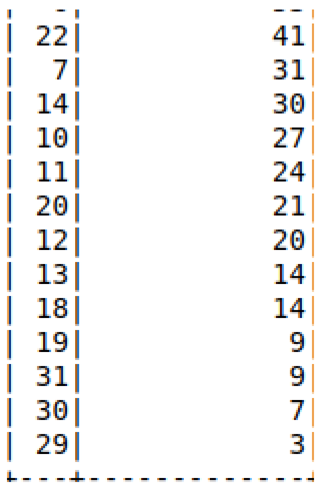


i) Arrange Days of month as per cancellations done in Descending

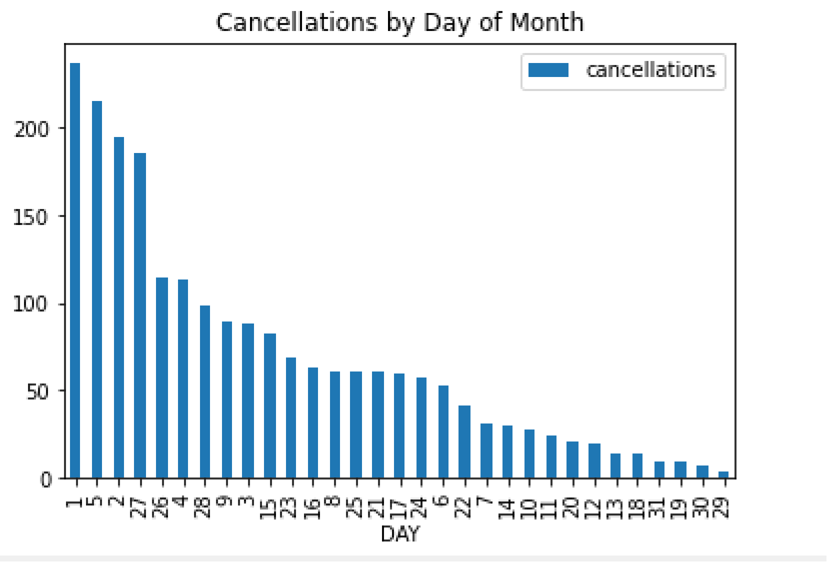
```
cancellations_by_day = spark.sql("""
        SELECT DAY, COUNT(*) AS cancellations
        FROM flights_table
        WHERE CANCELLED = 1
        GROUP BY DAY
        ORDER BY cancellations DESC
""")
cancellations_by_day.show(31)
```



```
+---+-------------+
|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|
```

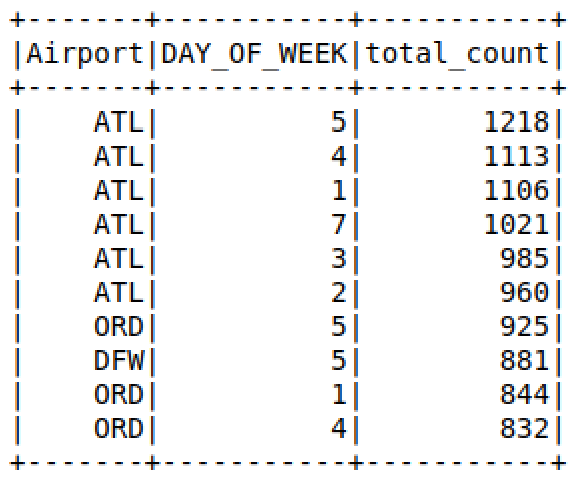


```
| 22|           41|
|  7|           31|
| 14|           30|
| 10|           27|
| 11|           24|
| 20|           21|
| 12|           20|
| 13|           14|
| 18|           14|
| 19|            9|
| 31|            9|
| 30|            7|
| 29|            3|
+---+-------------+
# Visualisation
cancellations_by_day_df = cancellations_by_day.toPandas()
cancellations_by_day_df.plot(kind='bar', x='DAY', y='cancellations', title='Cancellations by
Day of Month')
```



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

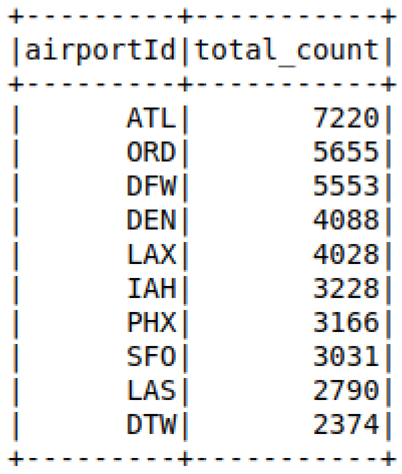
```
busiest_airports_by_day = spark.sql("""
SELECT Airport, DAY_OF_WEEK, SUM(flight_count) AS total_count FROM
(  SELECT ORIGIN_AIRPORT AS Airport, DAY_OF_WEEK, COUNT(*) AS flight_count
   FROM flights_table GROUP BY DAY_OF_WEEK, ORIGIN_AIRPORT
   UNION ALL
   SELECT DESTINATION_AIRPORT AS Airport, DAY_OF_WEEK, COUNT(*) AS flight_count
   FROM flights_table GROUP BY DAY_OF_WEEK, DESTINATION_AIRPORT
)
 AS combined GROUP BY Airport, DAY_OF_WEEK ORDER BY total_count DESC LIMIT 10
""")
busiest_airports_by_day.show()
```



```
+-------+-----------+-----------+
|Airport|DAY_OF_WEEK|total_count|
+-------+-----------+-----------+
|    ATL|          5|       1218|
|    ATL|          4|       1113|
|    ATL|          1|       1106|
|    ATL|          7|       1021|
|    ATL|          3|        985|
|    ATL|          2|        960|
|    ORD|          5|        925|
|    DFW|          5|        881|
|    ORD|          1|        844|
|    ORD|          4|        832|
+-------+-----------+-----------+
```

#top 10 busiest airports:-

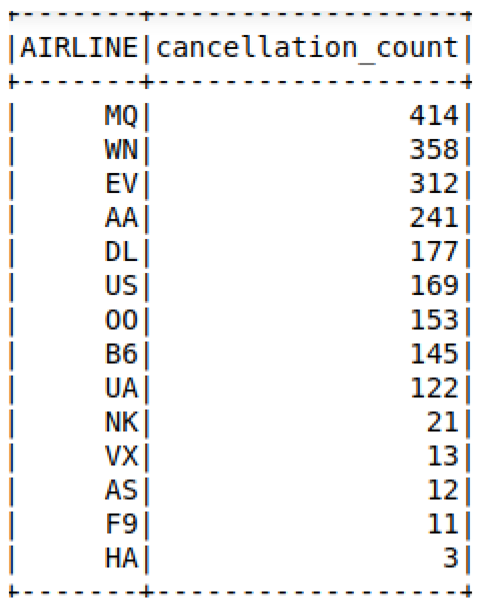
spark.sql("””select airportId, sum(total) as total_count from ((select ORIGIN_AIRPORT as airportId, count(*) as total from flights_table group by ORIGIN_AIRPORT) UNION ALL (select DESTINATION_AIRPORT as airportId, count(*) as total from flights_table group by DESTINATION_AIRPORT)) as combined group by airportId order by total_count DESC limit 10"””).show()



```
+---------+-----------+
|airportId|total_count|
+---------+-----------+
|      ATL|       7220|
|      ORD|       5655|
|      DFW|       5553|
|      DEN|       4088|
|      LAX|       4028|
|      IAH|       3228|
|      PHX|       3166|
|      SFO|       3031|
|      LAS|       2790|
|      DTW|       2374|
+---------+-----------+
```

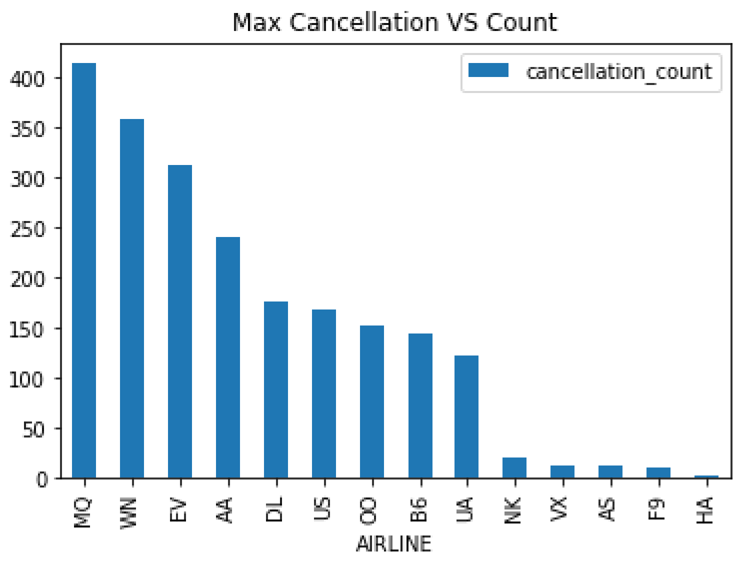
k) Finding airlines that make the maximum number of cancellations

```
max_cancellations = spark.sql("""
        SELECT AIRLINE, COUNT(*) AS cancellation_count
        FROM flights_table
        WHERE CANCELLED = 1
        GROUP BY AIRLINE
        ORDER BY cancellation_count DESC
""")
max_cancellations.show()
```



```
+-------+------------------+
|AIRLINE|cancellation_count|
+-------+------------------+
|     MQ|               414|
|     WN|               358|
|     EV|               312|
|     AA|               241|
|     DL|               177|
|     US|               169|
|     OO|               153|
|     B6|               145|
|     UA|               122|
|     NK|                21|
|     VX|                13|
|     AS|                12|
|     F9|                11|
|     HA|                 3|
+-------+------------------+
```

# Visualisation
max_cancellations_df = max_cancellations.toPandas()
max_cancellations_df.plot(kind='bar', x='AIRLINE', y='cancellation_count', title='Max Cancellation VS Count')

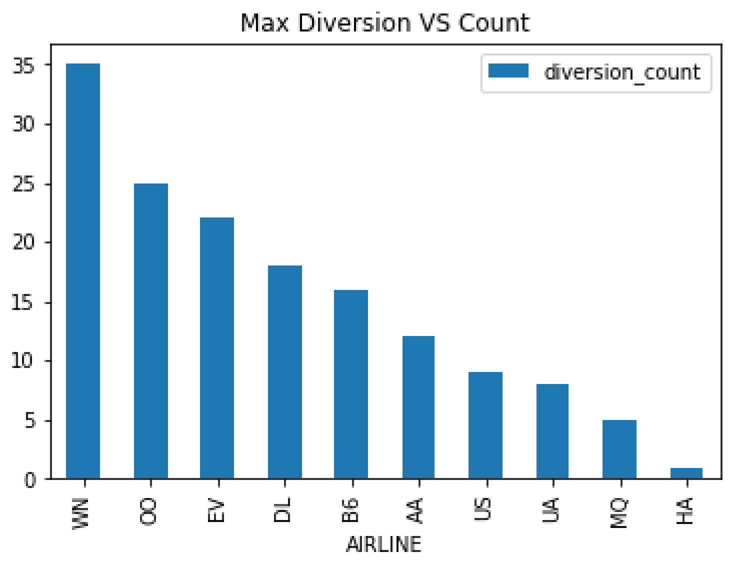


l) Find and order airlines in descending that make the most number of diversions

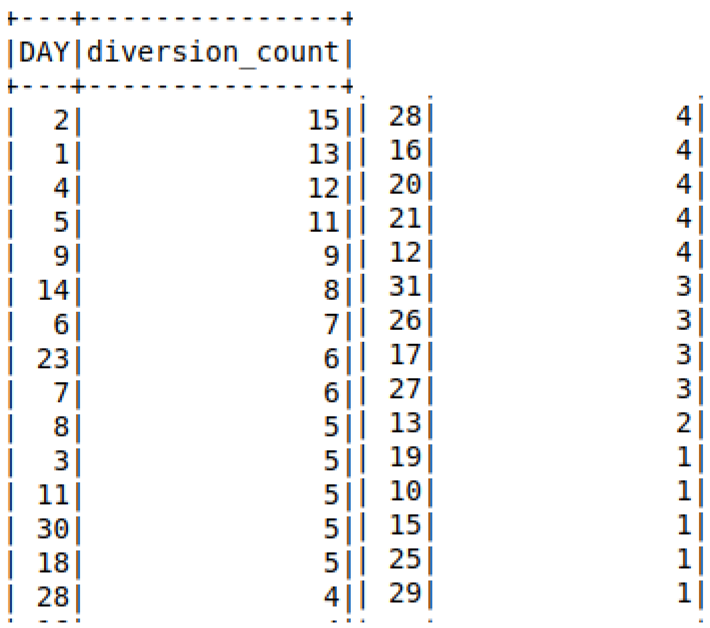
```
max_diversions_airlines = spark.sql("""
        SELECT AIRLINE, COUNT(*) AS diversion_count
        FROM flights_table
        WHERE DIVERTED = 1
        GROUP BY AIRLINE
        ORDER BY diversion_count DESC
""")
max_diversions_airlines.show()
```

```
+-------+---------------+
|AIRLINE|diversion_count|
+-------+---------------+
|     WN|             35|
|     OO|             25|
|     EV|             22|
|     DL|             18|
|     B6|             16|
|     AA|             12|
|     US|              9|
|     UA|              8|
|     MQ|              5|
|     HA|              1|
+-------+---------------+
```

# Visualisation
max_diversions_df = max_diversions_airlines.toPandas()
max_diversions_df.plot(kind='bar', x='AIRLINE', y='diversion_count', title='Max Diversion VS Count')

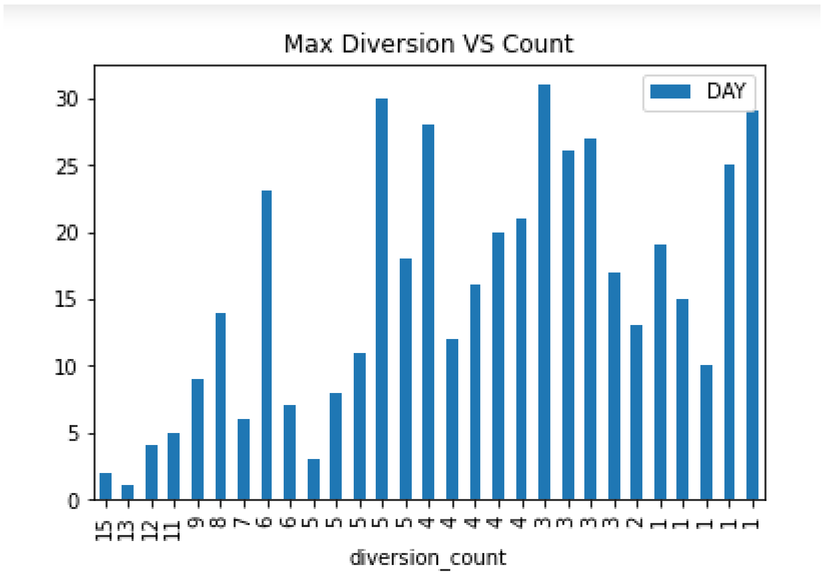


m) Finding days of month that see the most number of diversion
max_diversions_days = spark.sql("""
SELECT DAY, COUNT(*) AS diversion_count
FROM flights_table
WHERE DIVERTED = 1
GROUP BY DAY
ORDER BY diversion_count DESC
""")
max_diversions_days.show()



```
+---+---------------+
|DAY|diversion_count|
+---+---------------+
|  2|             15|| 28|               4|
|  1|             13|| 16|               4|
|  4|             12|| 20|               4|
|  5|             11|| 21|               4|
|  9|              9|| 12|               4|
| 14|              8|| 31|               3|
|  6|              7|| 26|               3|
| 23|              6|| 17|               3|
|  7|              6|| 27|               3|
|  8|              5|| 13|               2|
|  3|              5|| 19|               1|
| 11|              5|| 10|               1|
| 30|              5|| 15|               1|
| 18|              5|| 25|               1|
| 28|              4|| 29|               1|
```

# Visualisation
max_diversions_days_df = max_diversions_days.toPandas()
max_diversions_days_df.plot(kind='bar', y='DAY', x='diversion_count', title='Max Diversion VS Count')

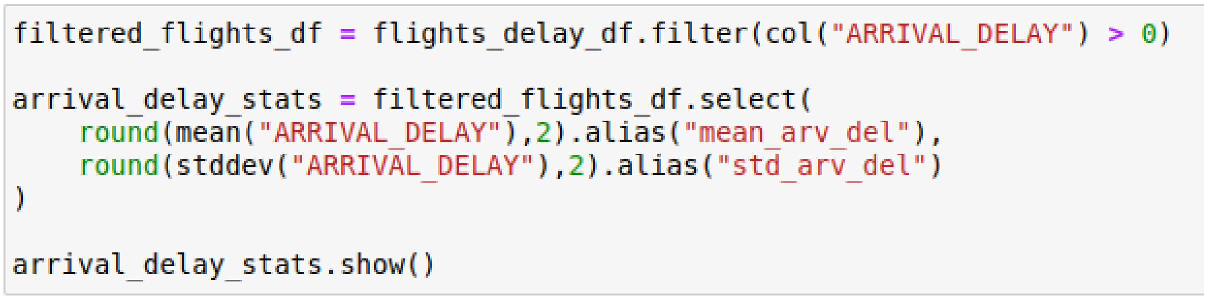


n) Calculating mean and standard deviation of departure delay for all flights in minutes

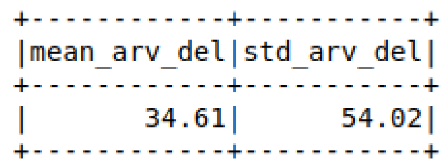
```
filtered_flights_df = flights_delay_df.filter(col("DEPARTURE_DELAY") > 0)

departure_delay_stats = filtered_flights_df.select(
    round(mean("DEPARTURE_DELAY"),2).alias("mean_dep_del"),
    round(stddev("DEPARTURE_DELAY"),2).alias("std_dep_del")
)

departure_delay_stats.show()
```

```
+------------+-----------+
|mean_dep_del|std_dep_del|
+------------+-----------+
|       33.92|      54.23|
+------------+-----------+
```

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



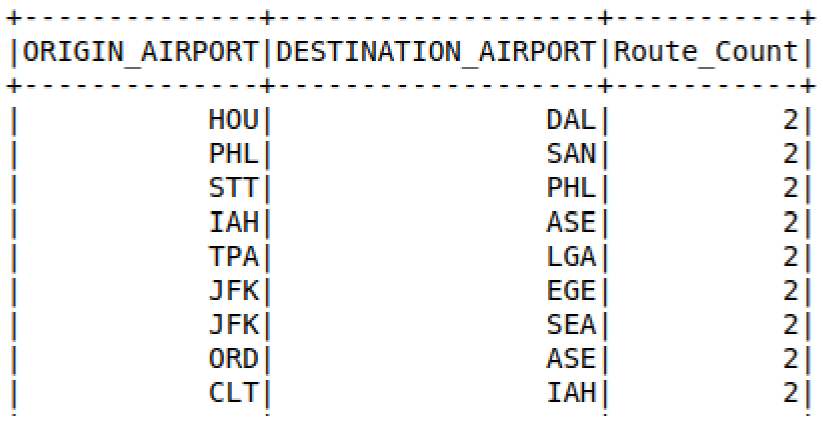
```
filtered_flights_df = flights_delay_df.filter(col("ARRIVAL_DELAY") > 0)

arrival_delay_stats = filtered_flights_df.select(
    round(mean("ARRIVAL_DELAY"),2).alias("mean_arv_del"),
    round(stddev("ARRIVAL_DELAY"),2).alias("std_arv_del")
)

arrival_delay_stats.show()
```



```
+------------+-----------+
|mean_arv_del|std_arv_del|
+------------+-----------+
|       34.61|      54.02|
+------------+-----------+
```

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

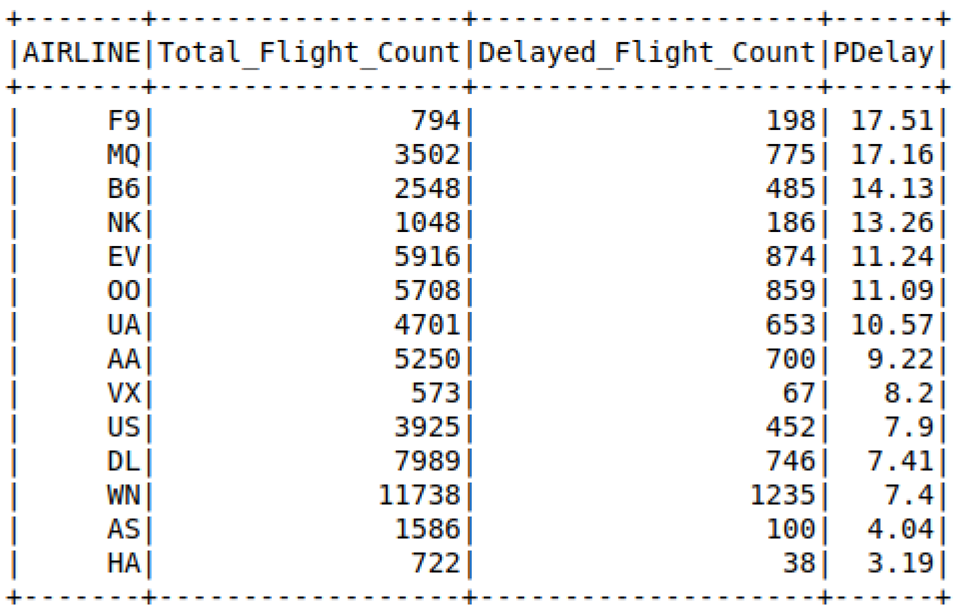
spark.sql("""select ORIGIN_AIRPORT, DESTINATION_AIRPORT, COUNT(*) as Route_Count from flights_table where DIVERTED = 1 group by ORIGIN_AIRPORT, DESTINATION_AIRPORT ORDER BY Route_Count DESC""").show()



```
+--------------+-------------------+-----------+
|ORIGIN_AIRPORT|DESTINATION_AIRPORT|Route_Count|
+--------------+-------------------+-----------+
|           HOU|                DAL|          2|
|           PHL|                SAN|          2|
|           STT|                PHL|          2|
|           IAH|                ASE|          2|
|           TPA|                LGA|          2|
|           JFK|                EGE|          2|
|           JFK|                SEA|          2|
|           ORD|                ASE|          2|
|           CLT|                IAH|          2|
```

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.

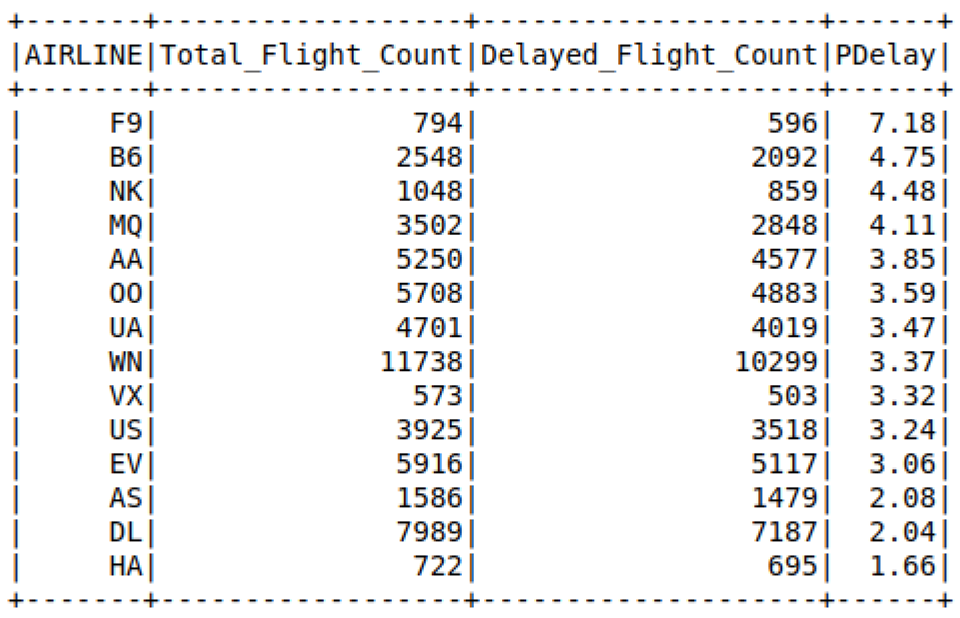
spark.sql("""select AIRLINE, count(*) as Total_Flight_Count,
sum(case when ARRIVAL_DELAY > 30 then 1 else 0 end) as Delayed_Flight_Count,
round(100 * sum(case when ARRIVAL_DELAY > 30 and DAY_OF_WEEK not in (6,7) then 1 else 0 end)/count(*),2) as PDelay
FROM flights_table
WHERE AIRLINE NOT IN ('AK', 'HI', '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.

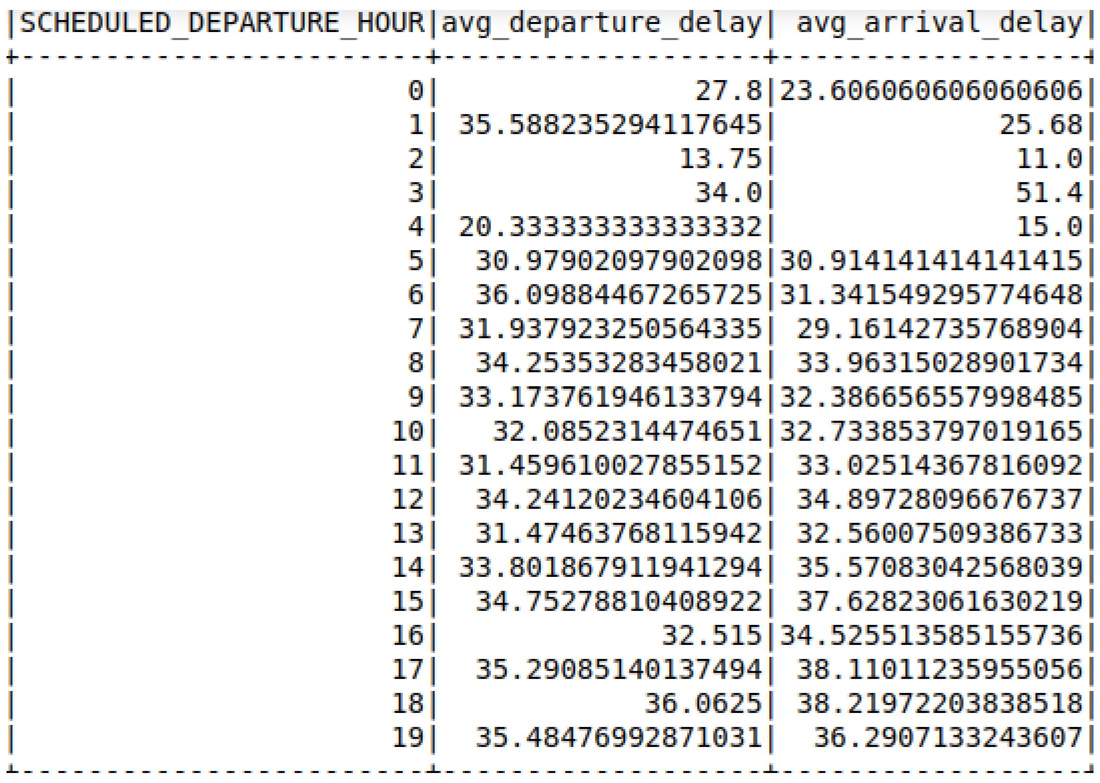
```
spark.sql("""select AIRLINE, count(*) as Total_Flight_Count,
        sum(case when DEPARTURE_DELAY < 30 then 1 else 0 end) as
Delayed_Flight_Count,
        round(100 * sum(case when  DEPARTURE_DELAY > 30 and DAY_OF_WEEK >5 then 1
else 0 end)/count(*),2) as PDelay
        FROM flights_table
        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|                596|  7.18|
|     B6|             2548|               2092|  4.75|
|     NK|             1048|                859|  4.48|
|     MQ|             3502|               2848|  4.11|
|     AA|             5250|               4577|  3.85|
|     OO|             5708|               4883|  3.59|
|     UA|             4701|               4019|  3.47|
|     WN|            11738|              10299|  3.37|
|     VX|              573|                503|  3.32|
|     US|             3925|               3518|  3.24|
|     EV|             5916|               5117|  3.06|
|     AS|             1586|               1479|  2.08|
|     DL|             7989|               7187|  2.04|
|     HA|              722|                695|  1.66|
+-------+-----------------+-------------------+------+
```

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

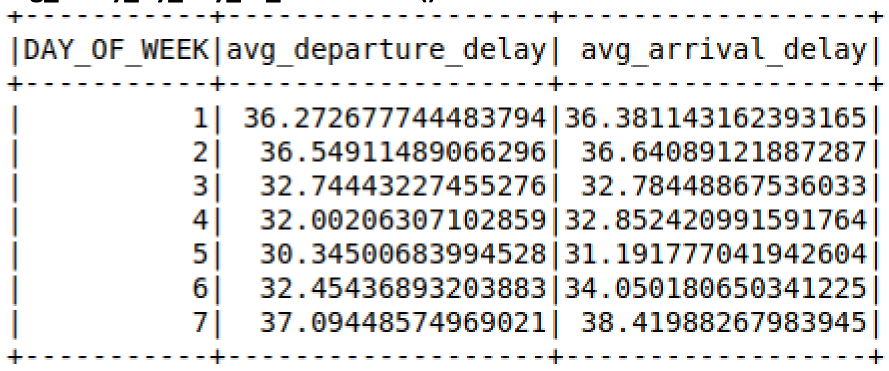
```
from pyspark.sql.functions import hour, minute, col, when, avg

# Convert SCHEDULED_DEPARTURE and SCHEDULED_ARRIVAL to hour of the day
flights_delay_updated_df =
flights_delay_updated_df.withColumn("SCHEDULED_DEPARTURE_HOUR",
(flights_delay_updated_df["SCHEDULED_DEPARTURE"] / 100).cast("int")) \
                        .withColumn("SCHEDULED_ARRIVAL_HOUR",
(flights_delay_updated_df["SCHEDULED_ARRIVAL"] / 100).cast("int"))



# Calculate average departure and arrival delay by hour of the day
avg_delay_by_hour =
flights_delay_updated_df.groupBy("SCHEDULED_DEPARTURE_HOUR").agg(
            avg(when(col("DEPARTURE_DELAY") > 0,
col("DEPARTURE_DELAY"))).alias("avg_departure_delay"),
            avg(when(col("ARRIVAL_DELAY") > 0,
col("ARRIVAL_DELAY"))).alias("avg_arrival_delay")
).orderBy("SCHEDULED_DEPARTURE_HOUR")

# Show the results
avg_delay_by_hour.show()
```



```
|SCHEDULED_DEPARTURE_HOUR|avg_departure_delay| avg_arrival_delay|
+------------------------+-------------------+------------------+
|                       0|               27.8|23.606060606060606|
|                       1| 35.588235294117645|             25.68|
|                       2|              13.75|              11.0|
|                       3|               34.0|              51.4|
|                       4| 20.333333333333332|              15.0|
|                       5|  30.97902097902098|30.914141414141415|
|                       6|  36.09884467265725|31.341549295774648|
|                       7| 31.937923250564335| 29.16142735768904|
|                       8|  34.25353283458021| 33.96315028901734|
|                       9| 33.173761946133794|32.386655557998485|
|                      10|   32.0852314474651|32.733853797019165|
|                      11| 31.459610027855152| 33.02514367816092|
|                      12|  34.24120234604106| 34.89728096676737|
|                      13|  31.47463768115942| 32.56007509386733|
|                      14| 33.801867911941294| 35.57083042568039|
|                      15|  34.75278810408922| 37.62823061630219|
|                      16|             32.515|34.525513585155736|
|                      17|  35.29085140137494| 38.11011235955056|
|                      18|            36.0625| 38.21972203838518|
|                      19|  35.48476992871031|  36.2907133243607|
+------------------------+-------------------+------------------+
```

# Calculate average departure and arrival delay by day of the week
avg_delay_by_day_of_week = flights_delay_updated_df.groupBy("DAY_OF_WEEK").agg(
avg(when(col("DEPARTURE_DELAY") > 0,
col("DEPARTURE_DELAY"))).alias("avg_departure_delay"),
avg(when(col("ARRIVAL_DELAY") > 0,
col("ARRIVAL_DELAY"))).alias("avg_arrival_delay")
).orderBy("DAY_OF_WEEK")

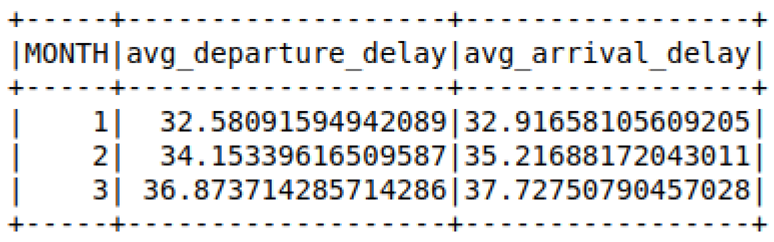
# Show the results
avg_delay_by_day_of_week.show()



```
+-----------+-------------------+------------------+
|DAY_OF_WEEK|avg_departure_delay| avg_arrival_delay|
+-----------+-------------------+------------------+
|          1| 36.272677744483794|36.381143162393165|
|          2|  36.54911489066296| 36.64089121887287|
|          3|  32.74443227455276| 32.78448867536033|
|          4|  32.00206307102859|32.852420991591764|
|          5|  30.34500683994528|31.191777041942604|
|          6|  32.45436893203883|34.050180650341225|
|          7|  37.09448574969021| 38.41988267983945|
+-----------+-------------------+------------------+
```

# Calculate average departure and arrival delay by month
avg_delay_by_month = flights_delay_updated_df.groupBy("MONTH").agg(
avg(when(col("DEPARTURE_DELAY") > 0,
col("DEPARTURE_DELAY"))).alias("avg_departure_delay"),

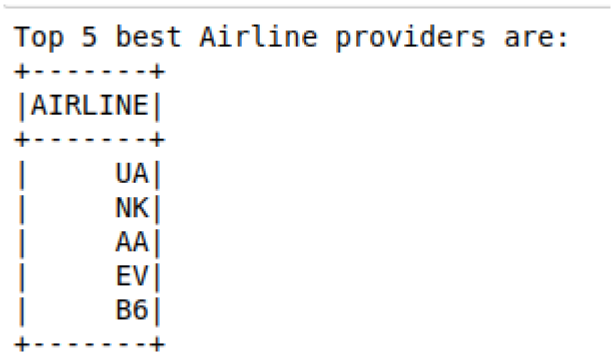
```
        avg(when(col("ARRIVAL_DELAY") > 0,
col("ARRIVAL_DELAY"))).alias("avg_arrival_delay")
).orderBy("MONTH")

# Show the results
avg_delay_by_month.show()
```



```
+-----+------------------+-----------------+
|MONTH|avg_departure_delay|avg_arrival_delay|
+-----+------------------+-----------------+
|    1|  32.58091594942089|32.91658105609205|
|    2|  34.15339616509587|35.21688172043011|
|    3| 36.873714285714286|37.72750790457028|
+-----+------------------+-----------------+
```

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

```
from pyspark.sql.functions import col, avg, sum

airline_performance = spark.sql("""

        SELECT

        AIRLINE,

        COUNT(*) AS total_flights,


        SUM(CANCELLED) AS total_cancellations,

        AVG(CASE WHEN 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 > 0 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

""")


# Top 5 Airline providers

print("Top 5 best Airline providers are: ")

airline_performance.select('AIRLINE').show()
```



```
Top 5 best Airline providers are:
+-------+
|AIRLINE|
+-------+
|     UA|
|     NK|
|     AA|
|     EV|
|     B6|
+-------+
```