a) Create a new Spark Session with new SparkConfig

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
        .appName("pyspark-hive-integration") \
        .config('spark.sql.warehouse.dir', '/user/hive/warehouse/') \
        .enableHiveSupport() \
        .getOrCreate()
```

Output:

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

```
flight_df = spark.read.csv("file:///home/hadoop/Downloads/Flights_Delay.csv",inferSchema = True,header = True)
df_selected = flight_df.select(
                "ID", "YEAR", "MONTH", "DAY", "DAY_OF_WEEK", "AIRLINE",
        "FLIGHT_NUMBER", "TAIL_NUMBER",
                "ORIGIN_AIRPORT", "DESTINATION_AIRPORT", "SCHEDULED_DEPARTURE",
        "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"
        )

df_selected.createOrReplaceTempView("flights")
```

Output:

```
In [7]:
flight_df = spark.read.csv("file:///home/hadoop/Downloads/Flights_Delay.csv",inferSchema = True,header = True)
df_selected = flight_df.select(
    "ID", "YEAR", "MONTH", "DAY", "DAY_OF_WEEK", "AIRLINE", "FLIGHT_NUMBER", "TAIL_NUMBER",
    "ORIGIN_AIRPORT", "DESTINATION_AIRPORT", "SCHEDULED_DEPARTURE", "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"
)

df_selected.createOrReplaceTempView("flights")
```

c) Create table Spark HIVE table flights_table

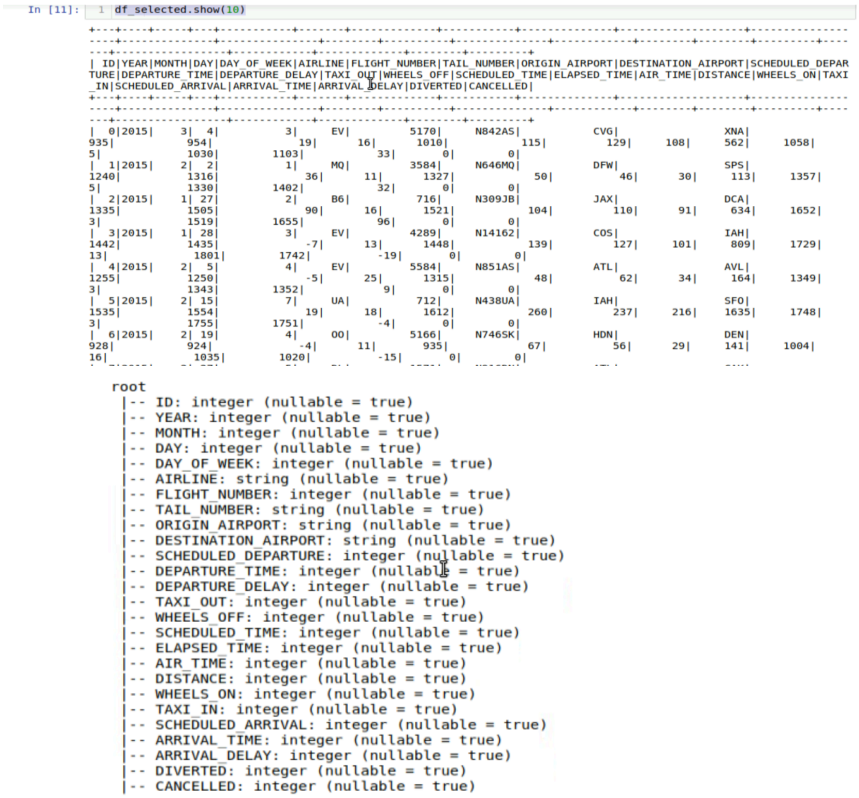
```
spark.sql("create database if not exists flight_db ").show()
spark.sql("use flight_db")



spark.sql("""
CREATE TABLE IF NOT EXISTS flights_table (
        ID INT,
        YEAR INT,
        MONTH INT,
        DAY INT,
        DAY_OF_WEEK INT,
        AIRLINE STRING,
        FLIGHT_NUMBER STRING,
        TAIL_NUMBER 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,
        DISTANCE INT,
        WHEELS_ON INT,
        TAXI_IN INT,
        SCHEDULED_ARRIVAL INT,
        ARRIVAL_TIME INT,
        ARRIVAL_DELAY INT,
        DIVERTED INT,
        CANCELLED INT
)
ROW FORMAT DELIMITED
```

```
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
tblProperties("skip.header.line.count" = 1)
""")
```

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

```
df_selected.printSchema()
df_selected.show(10)
```

Output:



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

```
df = df_selected.repartition("MONTH")
df.write.parquet("flights_parquet")
df.cache()
df_filtered = df.filter(df["ARRIVAL_DELAY"] > 0)
```

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

f) Average arrival delay caused by airlines

Query:

```
spark.sql("SELECT AIRLINE,AVG(ARRIVAL_DELAY) as average_delay FROM flights GROUP BY AIRLINE").show()
```

Output:

```
+-------+------------------+
|AIRLINE|     average_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|
+-------+------------------+
```

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

Query:

```
spark.sql("SELECT DAY,AVG(ARRIVAL_DELAY) as average_delay FROM flights GROUP BY DAY").show()
```

Output:

```
+---+------------------+
|DAY|     average_delay|
+---+------------------+
| 31|-1.196594427244582|
| 28| 3.257425742574257|
| 27| 4.706711409395973|
| 26| 11.96778269109286|
| 12| 11.24892703862661|
| 22| 6.550920245398773|
|  1|14.807807807807809|
| 13|3.3769751693002257|
|  6|10.608832807570979|
| 16| 9.124321062160531|
|  3|18.141541038525965|
| 20|3.8770149253731345|
|  5| 16.23061262014208|
| 19|1.6344282238442822|
| 15| 2.966753585397653|
|  9| 4.421887390959556|
| 17| 8.761435608726249|
|  4|17.157790927021697|
|  8| 5.232349165596919|
| 23| 4.207086133170434|
+---+------------------+
only showing top 20 rows
```

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

Query:

spark.sql("SELECT DAY_OF_WEEK,AVG(ARRIVAL_DELAY) AS AVERAGE_DELAY FROM flights GROUP BY DAY_OF_WEEK ORDER BY AVERAGE_DELAY").show()

Output:

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

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

Query:

spark.sql("SELECT DAY FROM flights WHERE CANCELLED = 1 GROUP BY DAY ORDER BY DAY DESC").show()

Output:

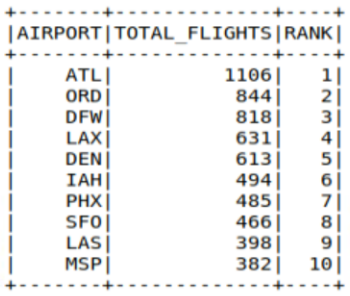
```
+---+
|DAY|
+---+
| 31|
| 30|
| 29|
| 28|
| 27|
| 26|
| 25|
| 24|
| 23|
| 22|
| 21|
| 20|
| 19|
| 18|
| 17|
| 16|
| 15|
| 14|
| 13|
| 12|
+---+
```

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

Query:

```
spark.sql("""
WITH flight_counts AS (
        SELECT ORIGIN_AIRPORT AS AIRPORT,DAY_OF_WEEK, COUNT(*) AS
NUM_FLIGHTS
        FROM flights
        GROUP BY ORIGIN_AIRPORT,DAY_OF_WEEK
        UNION ALL
        SELECT DESTINATION_AIRPORT AS AIRPORT,DAY_OF_WEEK,COUNT(*)
AS NUM_FLIGHTS
        FROM flights
        GROUP BY DESTINATION_AIRPORT,DAY_OF_WEEK
),
airport_totals AS (
        SELECT
        AIRPORT,
        DAY_OF_WEEK,
        SUM(NUM_FLIGHTS) AS TOTAL_FLIGHTS
        FROM flight_counts
        GROUP BY AIRPORT,DAY_OF_WEEK
),
ranked_airports AS (
        SELECT
        AIRPORT,
        DAY_OF_WEEK,
        TOTAL_FLIGHTS,
        ROW_NUMBER() OVER (PARTITION BY DAY_OF_WEEK ORDER BY
TOTAL_FLIGHTS DESC) AS RANK
        FROM airport_totals
)
SELECT
        AIRPORT,
        TOTAL_FLIGHTS,
        RANK
FROM ranked_airports
WHERE RANK <= 10
ORDER BY DAY_OF_WEEK,RANK
LIMIT 10
""").show()
```

Output:



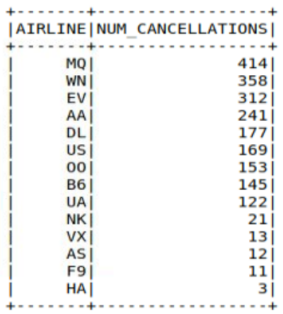
AIRPORT	TOTAL_FLIGHTS	RANK
ATL	1106	1
ORD	844	2
DFW	818	3
LAX	631	4
DEN	613	5
IAH	494	6
PHX	485	7
SFO	466	8
LAS	398	9
MSP	382	10

k) Finding airlines that make the maximum number of cancellations

Query:

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

Output:



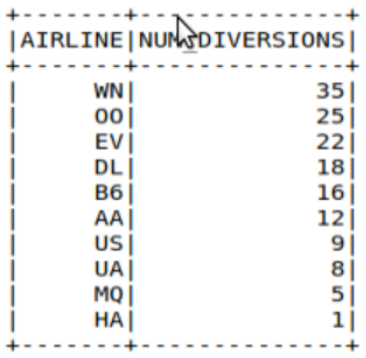
AIRLINE	NUM_CANCELLATIONS
MQ	414
WN	358
EV	312
AA	241
DL	177
US	169
OO	153
B6	145
UA	122
NK	21
VX	13
AS	12
F9	11
HA	3

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

Query:

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

Output:



AIRLINE	NUM_DIVERSIONS
WN	35
OO	25
EV	22
DL	18
B6	16
AA	12
US	9
UA	8
MQ	5
HA	1

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

Query:

```
spark.sql("""
WITH diversion_count as(
        SELECT MONTH,DAY,COUNT(*) AS NUM_DIVERSIONS
        FROM flights
        WHERE DIVERTED = 1
        GROUP BY MONTH,DAY
),
max_days as(
        SELECT MONTH,MAX(DAY) AS MAX_DAYS
        FROM diversion_count
        GROUP BY MONTH
)
SELECT d.MONTH,d.DAY AS MAX_DAYS,d.NUM_DIVERSIONS
FROM diversion_count d
JOIN max_days m
ON d.MONTH = m.MONTH AND d.DAY = m.MAX_DAYS
ORDER BY d.MONTH
""").show()
```

Output:

```
+-----+--------+--------------+
|MONTH|MAX_DAYS|NUM_DIVERSIONS|
+-----+--------+--------------+
|    1|      31|             3|
|    2|      28|             3|
|    3|       9|             2|
+-----+--------+--------------+
```

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

Query:

```
from pyspark.sql.functions import (col, floor)

df_with_delay_conversions = df.withColumn(
    "DEPARTURE_DELAY_MINUTES",
    col("DEPARTURE_DELAY") % 60
).withColumn(
    "ARRIVAL_DELAY_MINUTES",
    col("ARRIVAL_DELAY") % 60
)
df_with_delay_conversions.createOrReplaceTempView("flights_with_delay_conversions")
spark.sql("SELECT MEAN(DEPARTURE_DELAY_MINUTES) AS MEAN_DELAY,STDDEV(DEPARTURE_DELAY_MINUTES) AS STDDEV_DELAY from flights_with_delay_conversions").show()
```

Output:

```
+-----------------+------------------+
|       MEAN_DELAY|      STDDEV_DELAY|
+-----------------+------------------+
|4.76619112108788|15.059861202574904|
+-----------------+------------------+
```

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

Query:

```
spark.sql("SELECT MEAN(ARRIVAL_DELAY_MINUTES) AS MEAN_ARRIVAL_DELAY,STDDEV(ARRIVAL_DELAY_MINUTES) AS STDDEV_ARRIVAL_DELAY from flights_with_delay_conversions").show()
```

Output:

```
+------------------+-------------------+
|MEAN_ARRIVAL_DELAY|STDDEV_ARRIVAL_DELAY|
+------------------+-------------------+
|0.7675518641290179|   19.19434655049972|
+------------------+-------------------+
```

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

Query:

```
spark.sql("""
with DIVERTED_ROUTES AS(
        SELECT ORIGIN_AIRPORT,DESTINATION_AIRPORT,
        COUNT(*) AS DIVERSION_COUNT
        FROM flights
        WHERE DIVERTED = 1
        GROUP BY ORIGIN_AIRPORT,DESTINATION_AIRPORT
),
MAX_DIVERTED_ROUTE AS(
        SELECT ORIGIN_AIRPORT,DESTINATION_AIRPORT,DIVERSION_COUNT
        FROM DIVERTED_ROUTES
        ORDER BY DIVERSION_COUNT DESC
        LIMIT 1
)
SELECT dr.ORIGIN_AIRPORT,dr.DESTINATION_AIRPORT,dr.DIVERSION_COUNT,
CASE WHEN dr.ORIGIN_AIRPORT = mdr.ORIGIN_AIRPORT and
dr.DESTINATION_AIRPORT = mdr.DESTINATION_AIRPORT
THEN 'Most Diverted Route' ELSE '' END AS RouteType
FROM DIVERTED_ROUTES dr
LEFT JOIN MAX_DIVERTED_ROUTE mdr
ON dr.ORIGIN_AIRPORT = mdr.ORIGIN_AIRPORT and dr.DESTINATION_AIRPORT =
mdr.DESTINATION_AIRPORT
ORDER BY dr.DIVERSION_COUNT DESC
""").show()
```

Output:

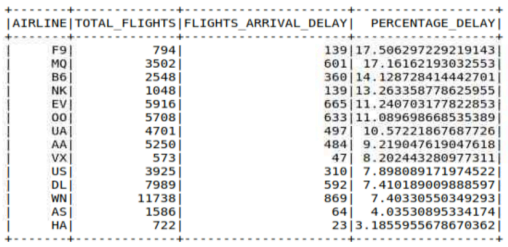
ORIGIN_AIRPORT	DESTINATION_AIRPORT	DIVERSION_COUNT	RouteType
HOU	DAL	2	
PHL	SAN	2	
STT	PHL	2	
TPA	LGA	2	Most Diverted Route
IAH	ASE	2	
JFK	EGE	2	
JFK	SEA	2	
ORD	ASE	2	
CLT	IAH	2	
EWR	STL	1	
SBP	SFO	1	
FLL	PVD	1	
SLC	RDM	1	
SLC	SUN	1	
CLT	MIA	1	
ATL	GTR	1	
ATL	LGA	1	
IAH	ISN	1	
LAX	ASE	1	
SNA	SFO	1	

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.

Query:

```
spark.sql("""
WITH FILTERED_FLIGHTS AS (
        SELECT
        AIRLINE,
        COUNT(*) AS TOTAL_FLIGHTS,
        SUM(CASE WHEN ARRIVAL_DELAY > 30 AND DAY_OF_WEEK NOT IN (6,
7) THEN 1 ELSE 0 END) AS FLIGHTS_ARRIVAL_DELAY
        FROM FLIGHTS
        WHERE AIRLINE NOT IN ('AK', 'HI', 'PR', 'VI')
        GROUP BY AIRLINE
        HAVING TOTAL_FLIGHTS > 10
),
DELAY_PERCENTAGE AS (
        SELECT
        AIRLINE,
        TOTAL_FLIGHTS,
        FLIGHTS_ARRIVAL_DELAY,
        (FLIGHTS_ARRIVAL_DELAY / TOTAL_FLIGHTS) * 100 AS
PERCENTAGE_DELAY
        FROM FILTERED_FLIGHTS
)
SELECT
        AIRLINE,
        TOTAL_FLIGHTS,
        FLIGHTS_ARRIVAL_DELAY,
        PERCENTAGE_DELAY
FROM DELAY_PERCENTAGE
ORDER BY PERCENTAGE_DELAY DESC
""").show()
```

Output:



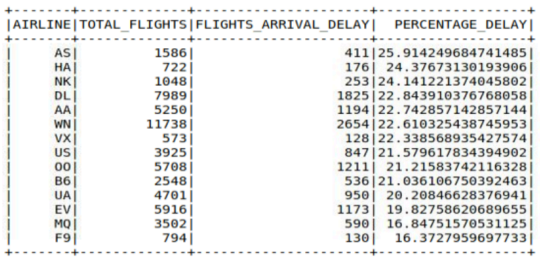
AIRLINE	TOTAL_FLIGHTS	FLIGHTS_ARRIVAL_DELAY	PERCENTAGE_DELAY
F9	794	139	17.506297229219143
MQ	3502	601	17.16162193032553
B6	2548	360	14.128728414442701
NK	1048	139	13.263358778625955
EV	5916	665	11.240703177822853
OO	5708	633	11.089698668535389
UA	4701	497	10.572218676877726
AA	5250	484	9.219047619047618
VX	573	47	8.202443280977311
US	3925	310	7.898089171974522
DL	7989	592	7.410189009888597
WN	11738	869	7.40330550349293
AS	1586	64	4.03530895334174
HA	722	23	3.1855955678670362

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.

Query:

```
spark.sql("""
WITH FILTERED_FLIGHTS AS (
        SELECT
        AIRLINE,
        COUNT(*) AS TOTAL_FLIGHTS,
        SUM(CASE WHEN ARRIVAL_DELAY < 30 AND DAY_OF_WEEK IN (6, 7)
THEN 1 ELSE 0 END) AS FLIGHTS_ARRIVAL_DELAY
        FROM FLIGHTS
        WHERE AIRLINE NOT IN ('AK', 'HI', 'PR', 'VI')
        GROUP BY AIRLINE
        HAVING TOTAL_FLIGHTS > 10
),
DELAY_PERCENTAGE AS (
        SELECT
        AIRLINE,
        TOTAL_FLIGHTS,
        FLIGHTS_ARRIVAL_DELAY,
        (FLIGHTS_ARRIVAL_DELAY / TOTAL_FLIGHTS) * 100 AS
PERCENTAGE_DELAY
        FROM FILTERED_FLIGHTS
)
SELECT
        AIRLINE,
        TOTAL_FLIGHTS,
        FLIGHTS_ARRIVAL_DELAY,
        PERCENTAGE_DELAY
FROM DELAY_PERCENTAGE
ORDER BY PERCENTAGE_DELAY DESC
""").show()
```

Output:



AIRLINE	TOTAL_FLIGHTS	FLIGHTS_ARRIVAL_DELAY	PERCENTAGE_DELAY
AS	1586	411	25.914249684741485
HA	722	176	24.37673130193906
NK	1048	253	24.141221374045802
DL	7989	1825	22.843910376768058
AA	5250	1194	22.742857142857144
WN	11738	2654	22.610325438745953
VX	573	128	22.338568935427574
US	3925	847	21.579617834394902
OO	5708	1211	21.21583742116328
B6	2548	536	21.036106750392463
UA	4701	950	20.20846628376941
EV	5916	1173	19.82758620689655
MQ	3502	590	16.84751570531125
F9	794	130	16.3727959697733

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

Query:

```
spark.sql("""
SELECT DAY_OF_WEEK,AVG(ARRIVAL_DELAY) AS avg_arrival_delay
FROM
flights
GROUP BY DAY_OF_WEEK
ORDER BY avg_arrival_delay ASC
LIMIT 1
""").show()
```

Output:

```
+-----------+-----------------+
|DAY_OF_WEEK|avg_arrival_delay|
+-----------+-----------------+
|          6|4.888689138576779|
+-----------+-----------------+
```

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

Query:

```
spark.sql("""
    WITH AIRLINE_DATA AS(
    SELECT
        AIRLINE,COUNT(*) AS TOTAL_FLIGHTS,
        SUM(CASE WHEN CANCELLED=1 THEN 1 ELSE 0 END) AS TOTAL_CANCELLED,
        SUM(CASE WHEN ARRIVAL_DELAY IS NOT NULL THEN ARRIVAL_DELAY ELSE 0
    END) AS AVG_ARRIVAL_DELAY,
        SUM(CASE WHEN DEPARTURE_DELAY IS NOT NULL THEN DEPARTURE_DELAY
    ELSE 0 END) AS AVG_DEPARTURE_DELAY
    FROM flights
    GROUP BY AIRLINE
    ),
    RANKED_AIRLINE AS(
    SELECT
        AIRLINE,
        TOTAL_FLIGHTS,
        TOTAL_CANCELLED,
        AVG_ARRIVAL_DELAY,
        AVG_DEPARTURE_DELAY,
        RANK() OVER (ORDER BY TOTAL_CANCELLED ASC,AVG_ARRIVAL_DELAY
    ASC,AVG_DEPARTURE_DELAY ASC) AS RANK
    FROM
        AIRLINE_DATA
    )
```

```
SELECT
        AIRLINE,
        TOTAL_FLIGHTS,
        TOTAL_CANCELLED,
        AVG_ARRIVAL_DELAY,
        AVG_DEPARTURE_DELAY
FROM
        RANKED_AIRLINE
ORDER BY
        RANK ASC
LIMIT 10
""").show()
```

Output:

AIRLINE	TOTAL_FLIGHTS	TOTAL_CANCELLED	AVG_ARRIVAL_DELAY	AVG_DEPARTURE_DELAY
HA	722	3	2924	851
F9	794	11	18873	18412
AS	1586	12	-2411	3680
VX	573	13	2872	5520
NK	1048	21	14590	16017
UA	4701	122	30613	65534
B6	2548	145	33319	38613
OO	5708	153	56156	64516
US	3925	169	22397	29375
DL	7989	177	21936	77642