

Case Study: Analysis of Airline Delay using Spark

Airline delay is a critical issue affecting both airlines and passengers. In this assignment, you will use

PySpark to analyze a dataset containing information about airline flights and predict flight delays.

Technology: Spark, SQL

Dataset: Flights_Delay.csv

Questions:

a) Create a new Spark Session with new SparkConfig

```
from pyspark import SparkConf, SparkContext
```

```
from pyspark.sql import SparkSession, HiveContext
```

```
config = SparkConf().setAppName("AirlineDelayAnalysis").setMaster('local[4]')
```

```
sc = SparkContext.getOrCreate(conf=config)
```

sc

```
Out[6]: SparkContext
```

```
Spark UI
```

```
Version
```

```
v2.4.8
```

```
Master
```

```
local[4]
```

```
AppName
```

```
AirlineDelayAnalysis
```

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

```
spark = (SparkSession.builder.appName("pyspark-hive-integration")\
```

```
.config('spark.sql.warehouse.dir', '/usr/hive/warehouse/')\
```

```
.enableHiveSupport().getOrCreate())
```

```

flights_df =
spark.read.csv("file:///home/hadoop/Downloads/Flights_Delay.csv",inferSchema =
True, header = True)

flights_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)
-- WHEELS_ON: integer (nullable = true)
-- TAXI_IN: integer (nullable = true)
-- SCHEDULED_ARRIVAL: integer (nullable = true)
-- ARRIVAL_TIME: integer (nullable = true)
-- ARRIVAL_DELAY: integer (nullable = true)
-- DIVERTED: integer (nullable = true)
-- CANCELLED: integer (nullable = true)
-- CANCELLATION_REASON: string (nullable = true)
-- AIR_SYSTEM_DELAY: integer (nullable = true)
-- SECURITY_DELAY: integer (nullable = true)
-- AIRLINE_DELAY: integer (nullable = true)
-- LATE_AIRCRAFT_DELAY: integer (nullable = true)
-- WEATHER_DELAY: integer (nullable = true)

```

c) Create table Spark HIVE table flights_table

```

spark.sql("create database if not exists flights_db").show()

spark.sql("show databases").show()

spark.sql("use flights_db").show()

```

```

+-----+
|databaseName|
+-----+
| banking_db|
| default|
| flights_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,
CANCELLATION_REASON STRING,
AIR_SYSTEM_DELAY INT,
SECURITY_DELAY INT,
AIRLINE_DELAY INT,
LATE_AIRCRAFT_DELAY INT,
WEATHER_DELAY INT
)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE

TBLPROPERTIES ("skip.header.line.count" = "1")

""")

```
df_selected = flights_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', 'CANCELLATION_REASON',
```

```
    'AIR_SYSTEM_DELAY', \
```

```
    'SECURITY_DELAY', 'AIRLINE_DELAY', 'LATE_AIRCRAFT_DELAY',
```

```
    'WEATHER_DELAY')
```

```
df_selected.createOrReplaceTempView("flights")
```

```
spark.sql("""
```

```
insert into table flights_table
```

```
select * from flights
```

```
""")
```

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

```
spark.sql("describe formatted flights_table").show(10)
```

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	string	null
TAIL_NUMBER	string	null
ORIGIN_AIRPORT	string	null
DESTINATION_AIRPORT	string	null

only showing top 10 rows

e) Apply Query performance optimization techniques like – creating Partitioning DataFrame by

caching

```
flights_cached = flights_df.cache()
```

flights_cached

```
Out[19]: DataFrame[ID: int, YEAR: int, MONTH: int, DAY: int, DAY OF WEEK: int, AIRLINE: string, FLIGHT NUMBER: int, 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, CANCELLATION REASON: string, AIR_SYSTEM_DELAY: int, SECURITY_DELAY: int, AIRLINE_DELAY: int, LATE_AIRCRAFT_DELAY: int, WEATHER_DELAY: int]
```

parquet

```
flights_df.write.parquet("file:///home/hadoop/Downloads/Flights/")
```

```
parquet_flight = spark.read.parquet("file:///home/hadoop/Downloads/Flights/part-00000-9b7b224a-b75c-4477-a13c-90a2734dc18b-c000.snappy.parquet")
```

parquet_flight.show()

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 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|CANCELLATION_REASON|AIR_SYSTEM_DELAY|SECURITY_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|    0|    null|    14| | | | | |
| 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|    0|    null|    0|
| 2|2015| 1| 27|         2|B6|    716|    N309JB|    JAX|    DCA|    0|    0|    0|    0|    0|    0|    0|    0|    0|    0|    0|    0|    0|    0|    0|    0|    0|    0|    0|    0|    0|    0|    0|    0|
```

partitioning

```
partitioned_flights = flights_df.repartition(3)
```

```
partitioned_flights.write.parquet("file:///home/hadoop/Downloads/Flights/Repartition")
```

```
parquet_flight = spark.read.parquet("file:///home/hadoop/Downloads/Flights/part-00000-9b7b224a-b75c-4477-a13c-90a2734dc18b-c000.snappy.parquet")
```

```
parquet_flight.show()
```

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	CANCELLATION_REASON	AIR_SYSTEM_DELAY	SECURITY_DELAY	AIRLINE_DELAY	LATE_AIRCRAFT_DELAY	WEATHER_DELAY
935	2015	3	4	3	EV	5170	N842AS	CVG	XNA	129	1058	115	129	108	562	14	1058														
5			1030	1103		16	1010	115		129	1058	115	129	108	562	14	1058														
0			19	0		0	0																								
1240	2015	2	2	1	MQ	3584	N646MQ	DFW	SPS	46	1357	50	46	30	113	0	1357														
5			1330	1402		11	1327	50		46	1357	50	46	30	113	0	1357														
0			32	0		0	0																								
1335	2015	1	27	2	B6	716	N309JB	JAX	DCA	110	1652	104	110	91	634	6	1652														
3			1519	1655		16	1521	104		110	1652	104	110	91	634	6	1652														
0			90	0		0	0																								

```
# pushdown
```

```
parquet_flight.printSchema()
```

```
parquet_flight.filter(parquet_flight["MONTH"]!=1).show()
```

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	CANCELLATION_REASON	AIR_SYSTEM_DELAY	SECURITY_DELAY	AIRLINE_DELAY	LATE_AIRCRAFT_DELAY	WEATHER_DELAY	
935	2015	3	4	3	EV	5170	N842AS	CVG	XNA	129	1058	115	129	108	562	14	1058															
5			1030	1103		16	1010	115		129	1058	115	129	108	562	14	1058															
0			19	0		0	0																									
1240	2015	2	2	1	MQ	3584	N646MQ	DFW	SPS	46	1357	50	46	30	113	0	1357															
5			1330	1402		11	1327	50		46	1357	50	46	30	113	0	1357															
0			32	0		0	0																									
1255	2015	2	5	4	EV	5584	N851AS	ATL	AVL	62	1349	48	62	34	164	1	1349															
3			1343	1352		9	1315	48		62	1349	48	62	34	164	1	1349															
11			11	11		11	11																									

f) Average arrival delay caused by airlines

```
spark.sql("""
```

```
select airline, avg(airline_delay) Average_airline_delay from flights_table group by airline
```

```
""").show()
```

airline	Average_airline_delay
UA	18.825
NK	13.272425249169435
AA	21.797598627787306
EV	21.38169014084507
B6	19.62044817927171
DL	24.849923430321592
OO	17.7892749244713
F9	14.47266881028939
US	18.03059975520196
MQ	15.768812330009066
HA	24.591836734693878
AS	13.192118226600986
VX	10.85576923076923
WN	14.513646532438479

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

```
spark.sql("""
select Month,Day,avg(arrival_delay) Average_of_arrival_delay from flights_table
group by month,day
order by month,day
""").show()
```

Month	Day	Average_of_arrival_delay
1	1	5.939130434782609
1	2	9.62860310421286
1	3	26.090561224489797
1	4	31.87239263803681
1	5	19.858809801633605
1	6	21.018915510718788
1	7	13.591939546599496
1	8	13.301115241635689
1	9	11.06872852233677
1	10	3.1073529411764707
1	11	9.544980443285528
1	12	19.91331658291457
1	13	5.15112160566706
1	14	1.0241545893719808
1	15	-0.5099882491186839
1	16	-0.5833333333333334
1	17	-3.6096
1	18	-0.7560355781448539
1	19	-2.303921568627451

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

```
spark.sql("""
select Day_of_Week,avg(arrival_delay) Average_arrival_delay
from flights_table
group by day_of_week
order by avg(arrival_delay) desc
""").show()
```


Day_of_Week	Average_arrival_delay
1	10.807447207297264
7	10.110840438489646
2	8.033644102148358
4	7.174969021065675
5	6.010538373424971
3	5.587079407806191
6	4.888689138576779

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

```
spark.sql("""
select Day,Month,count(Cancelled) Cancellations from flights_table
where cancelled = 1
group by Day,Month
order by count(Cancelled) desc
""").show()
```

Day	Month	Cancellations
5	3	348
2	2	308
27	1	298
1	2	244
1	3	180
15	2	158
26	1	152
4	3	150
23	2	134
9	2	120
16	2	120
17	2	116
21	2	108
25	2	104
28	2	102
28	1	94

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

```
spark.sql("""
SELECT rank() OVER (PARTITION BY t.day_of_week ORDER BY sum(t.flight_count) DESC)
AS rank,
t.airport,
t.day_of_week,
sum(t.flight_count) AS total_flights
FROM (
SELECT destination_airport AS airport,
```



```

        day_of_week,
        count(*) AS flight_count
FROM flights_table
GROUP BY destination_airport, day_of_week
UNION ALL
SELECT origin_airport AS airport,
        day_of_week,
        count(*) AS flight_count
FROM flights_table
GROUP BY origin_airport, day_of_week
) t
GROUP BY t.airport, t.day_of_week
ORDER BY t.day_of_week, total_flights DESC
LIMIT 10
""").show()

```

rank	airport	day_of_week	total_flights
1	ATL	1	2212
2	ORD	1	1688
3	DFW	1	1636
4	LAX	1	1262
5	DEN	1	1226
6	IAH	1	988
7	PHX	1	970
8	SFO	1	932
9	LAS	1	796
10	MSP	1	764

k) Finding airlines that make the maximum number of cancellations

```

spark.sql("""
select Airline,count(cancelled) No_Of_Cancellations from flights_table
where cancelled = 1
group by airline

```

order by count(cancelled) desc

""").show()

Airline	No_Of_Cancellations
MQ	828
WN	716
EV	624
AA	482
DL	354
US	338
OO	306
B6	290
UA	244
NK	42
VX	26
AS	24
F9	22
HA	6

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

spark.sql("""

select airline,count(diverted) as total_diversion

from flights_table

where diverted = 1

group by airline

order by total_diversion desc

""").show()

airline	total_diversion
WN	70
OO	50
EV	44
DL	36
B6	32
AA	24
US	18
UA	16
MQ	10
HA	2

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

spark.sql("""

select month,day,count(diverted) as total_diversion

from flights_table

```

where diverted = 1

group by month,day

order by total_diversion desc

""").show()

```

month	day	total_diversion
2	2	18
3	5	14
2	14	14
3	1	14
3	2	12
3	4	12
1	18	10
2	23	10
1	11	10
1	7	10
1	30	10
2	1	10
2	21	8
1	8	8
2	9	8
1	4	6
2	4	6
2	28	6

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

```

spark.sql("""

select mean(departure_delay) as Mean, std(departure_delay) as Standard_deviation

from flights_table

""").show()

```

Mean	Standard_deviation
11.329091145205275	39.62073227132917

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

```

spark.sql("""

select mean(arrival_delay) as Mean, std(arrival_delay) as Standard_deviation

from flights_table

""").show()

```

Mean	Standard_deviation
7.545457931394093	42.37818852268862

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(diverted) as No_of_Diversion
from flights_table
where diverted = 1
group by origin_airport, destination_airport
order by No_of_Diversion
""").show()
```

origin_airport	destination_airport	No_of_Diversion
SFO	BOI	2
ATL	GSP	2
SNA	SFO	2
FLL	BWI	2
BOS	LAS	2
SBP	SFO	2
SLC	RDM	2
MCO	PVD	2
FLL	PVD	2
SLC	SUN	2
CLT	MIA	2
COS	ORD	2
KOA	SFO	2
EWB	STL	2
MCO	BWI	2
ATL	ASE	2
ATL	GTR	2
CAK	LGA	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("""
WITH airline_query AS (
```

```

SELECT
    f1.airline,
    COUNT(f1.flight_number) AS flights_travelled,
    COUNT(DISTINCT f1.flight_number) AS flights_each_airline,
    SUM(CASE WHEN f1.arrival_delay > 30 THEN 1 ELSE 0 END) AS
total_flights_delayed_by_30mins
FROM flights_table f1
GROUP BY f1.airline
)

```

```

SELECT
    f.airline,
    q.flights_travelled,
    q.flights_each_airline,
    q.total_flights_delayed_by_30mins,
    SUM(
        CASE
            WHEN f.arrival_delay > 30
                AND f.day_of_week NOT IN (6, 7)
                AND q.flights_each_airline > 10
                AND f.airline NOT IN ('AK', 'HI', 'PR', 'VI')
            THEN 1
            ELSE 0
        END
    ) / q.flights_travelled * 100 AS per_of_fl_delay_30
FROM flights_table f
INNER JOIN airline_query q
    ON f.airline = q.airline
GROUP BY

```

```

f.airline,
q.flights_travelled,
q.flights_each_airline,
q.total_flights_delayed_by_30mins
ORDER BY per_of_fl_delay_30 DESC
""").show()

```

airline	flights_travelled	flights_each_airline	total_flights_delayed_by_30mins	per_of_fl_delay_30
F9	1588	294	396	17.506297229219143
MQ	7004	740	1550	17.16162193032553
B6	5096	770	970	14.128728414442701
NK	2096	261	372	13.263358778625955
EV	11832	1685	1748	11.240703177822853
OO	11416	1462	1718	11.089698668535389
UA	9402	1167	1306	10.57221867687726
AA	10500	1129	1400	9.219047619047618
VX	1146	163	134	8.202443280977311
US	7850	836	904	7.898089171974522
DL	15978	2022	1492	7.410189009888597
WN	23476	2847	2470	7.40330550349293
AS	3172	435	200	4.03530895334174
HA	1444	210	76	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.

```

spark.sql("""
WITH airline_query AS (
    SELECT
        f1.airline,
        COUNT(f1.flight_number) AS flights_travelled,
        COUNT(DISTINCT f1.flight_number) AS flights_each_airline,
        SUM(CASE WHEN f1.departure_delay < 30 THEN 1 ELSE 0 END) AS
total_flights_delayed_less_30_mins
    FROM flights_table f1
    GROUP BY f1.airline
)
SELECT

```

```

f.airline,
q.flights_travelled,
q.flights_each_airline,
q.total_flights_delayed_less_30_mins,
SUM(
    CASE
        WHEN f.departure_delay < 30
            AND f.day_of_week IN (6, 7)
            AND q.flights_each_airline > 10
            AND q.airline NOT IN ('AK', 'HI', 'PR', 'VI')
        THEN 1
        ELSE 0
    END
) / q.flights_travelled * 100 AS per_of_fl_delay_30
FROM flights_table f
INNER JOIN airline_query q
    ON f.airline = q.airline
GROUP BY
    f.airline,
    q.flights_travelled,
    q.flights_each_airline,
    q.total_flights_delayed_less_30_mins
ORDER BY per_of_fl_delay_30 DESC
""").show()

```


airline	flights_travelled	flights_each_airline	total_flights_delayed_less_30_mins	per_of_fl_delay_30
AS	3172	435	2936	25.97730138713745
HA	1444	210	1384	24.792243767313018
NK	2096	261	1678	24.141221374045802
AA	10500	1129	8684	23.123809523809523
DL	15978	2022	14020	22.70622105394918
VX	1146	163	980	22.5130890052356
WN	23476	2847	19890	22.456977338558527
US	7850	836	6712	22.089171974522294
OO	11416	1462	9472	21.79397337070778
B6	5096	770	3894	21.31083202511774
EV	11832	1685	9638	20.33468559837728
UA	9402	1167	7806	20.20846628376941
MQ	7004	740	4886	17.76127926898915
F9	1588	294	1170	16.750629722921914

Activate Windows

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

```
spark.sql("""
```

```
SELECT
```

```
    day_of_week,
```

```
    AVG(COALESCE(arrival_delay, 0) + COALESCE(departure_delay, 0)) AS total_delay
```

```
FROM flights_table
```

```
GROUP BY day_of_week
```

```
ORDER BY total_delay
```

```
""").show()
```

day_of_week	total_delay
6	14.154561301568855
3	14.233472149921916
5	15.901192887688499
4	17.397617629541394
2	18.47514450867052
7	22.593458540948
1	23.541281180466097

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

```
spark.sql("""
```

```
SELECT
```

```
    Airline,
```

```
    COUNT(flight_number) AS total_flights,
```

```
    SUM(CASE WHEN cancelled = 1 THEN 1 ELSE 0 END) AS total_cancellations,
```

```

AVG(COALESCE(arrival_delay, 0)) AS avg_arrival_delay,
AVG(COALESCE(departure_delay, 0)) AS avg_departure_delay,
SUM(COALESCE(air_system_delay, 0) +
    COALESCE(security_delay, 0) +
    COALESCE(late_aircraft_delay, 0) +
    COALESCE(airline_delay, 0) +
    COALESCE(weather_delay, 0)
) AS total_delay_reasons
FROM flights_table
GROUP BY airline
""").show()

```

Airline	total_flights	total_cancellations	avg_arrival_delay	avg_departure_delay	total_delay_reasons
UA	9402	244	6.5120187194214	13.940438204637312	120396
NK	2096	42	13.92175572519084	15.283396946564885	37624
AA	10500	482	7.9824761904761905	10.987428571428572	141970
EV	11832	624	10.269776876267748	10.95368492224476	173040
B6	5096	290	13.076530612244898	15.154238618524333	92368
DL	15978	354	2.7457754412316935	9.71861309300288	168084
OO	11416	306	9.83812193412754	11.302733006306937	166506
F9	1588	22	23.769521410579344	23.188916876574307	42370
US	7850	338	5.70624203821656	7.484076433121019	84370
MQ	7004	828	16.93061107938321	15.098515134209023	138706
HA	1444	6	4.049861495844875	1.1786703601108033	8144
AS	3172	24	-1.5201765447667086	2.320302648171501	20976
VX	1146	26	5.012216404886562	9.633507853403142	13422
WN	23476	716	3.574033055034929	9.810103935934572	216420