Task #1

# Spark SQL

## Query #1

#Prepare Spark

from pyspark.context import SparkContext

from pyspark.sql.session import SparkSession

sc = SparkContext('local')

spark = SparkSession(sc)

from pyspark.sql.functions import \*

import datetime

#Create a view to analyze

df = spark.read.csv("flights.csv", header=True)

df.createOrReplaceTempView("flights")

#Convert time UDF

def ConvertTime(s):

if s is None:

return "00:00:00"

return s[0:2] + ":" + s[2:4] + ":00"

spark.udf.register("OS\_ConvertTime", ConvertTime)

ConvertTime\_udf = udf(ConvertTime)

#Convert time difference UDF

def minute\_diff(start,end):

if end is None:

return None

try:

startdate = datetime.datetime.strptime(start,'%H:%M:%S')

enddate = datetime.datetime.strptime(end,'%H:%M:%S')

except ValueError:

return None

timediff = enddate-startdate

diff\_in\_min = int(timediff.total\_seconds()//60)

if diff\_in\_min < 0:

diff\_in\_min += 1440

return(diff\_in\_min)

spark.udf.register("OS\_minute\_diff", minute\_diff)

minute\_diff\_udf = udf(minute\_diff)

# Create a dataset and persist the results

result = spark.sql(f"""

WITH cnt as (

SELECT COUNT(\*) as airline\_daily\_flights\_count,YEAR,MONTH,DAY,AIRLINE from Flights group by YEAR,MONTH,DAY,AIRLINE

)

select date\_format(concat(f.YEAR,'-',f.MONTH,'-',f.DAY),'yyyy-MM-dd') as departure\_date,

OS\_ConvertTime(f.DEPARTURE\_TIME) as departure\_time,

f.AIRLINE as airline,

f.FLIGHT\_NUMBER as flight\_number,

ROW\_NUMBER() OVER (PARTITION BY date\_format(concat(f.YEAR,'-',f.MONTH,'-',f.DAY),'yyyy-MM-dd') order by OS\_ConvertTime(f.DEPARTURE\_TIME)) as daily\_flight\_serial\_number,

c.airline\_daily\_flights\_count,

OS\_minute\_diff((Lag(OS\_ConvertTime(f.DEPARTURE\_TIME),1,0) OVER(PARTITION BY f.AIRLINE ORDER by f.YEAR,f.MONTH,f.DAy,f.DEPARTURE\_TIME )),OS\_ConvertTime(f.DEPARTURE\_TIME)) AS time\_since\_previous\_departure

from flights f

join cnt c on f.YEAR=c.YEAR and f.MONTH=c.MONTH and f.DAY=c.DAY and f.AIRLINE=c.AIRLINE

""")

OS\_airlines = result.persist()

#Read the golden dataset

Reference\_airlines = spark.read.parquet("Query\_1")

# Compare the results and write the difference to parquet file

Comp\_test4=OS\_airlines.subtract(Reference\_airlines)

Comp\_test4.write.parquet('SQL\_Airlines\_diff')

#Make an archive of parquet files to download

CLI:

C:\Users\Olga\_Seregina>docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

b93170698b60 jupyter/all-spark-notebook "tini -g -- start-no…" 6 days ago Up 6 days 0.0.0.0:4040->4040/tcp, 0.0.0.0:8888->8888/tcp loving\_taussig

C:\Users\Olga\_Seregina> docker exec b93170698b60 tar czvf SQL\_airlines\_diff.tar.gz SQL\_Airlines\_diff

#Write the difference to excel file

import pandas

pd\_obj =pandas.read\_parquet('SQL\_Airlines\_diff ', engine='pyarrow')

pd\_obj.to\_excel(r' SQL\_Airlines\_diff.xlsx', index = False)

Error appears:

ValueError: This sheet is too large! Your sheet size is: 5369763, 7 Max sheet size is: 1048576, 16384

Comparison result was not exported to excel file due to sheet size

Comparison results in parquet can be found in SQL\_airlines\_diff.tar.gz (attached)

## Query #2

#Read json file

df\_json = spark.read.json("airlines.json", multiLine=True)

df\_json.printSchema()

#Create a view for json data

df\_json.createOrReplaceTempView("airlines\_json\_view")

#Read csv file with airlines

df\_airl\_csv = spark.read.csv("airlines.csv", header=True)

#Create a view for csv with airlines

df\_airl\_csv.createOrReplaceTempView("airlines\_csv\_view")

#Create datasets

Result\_1 = spark.sql(f"""

Select

date\_format(concat(aj.Time.YEAR,'-',aj.Time.MONTH),'yyyy-MM') as year\_month,

aj.airport.code as airport\_code,

(aj.Statistics.Flights.Cancelled+aj.Statistics.Flights.Delayed+aj.Statistics.Flights.Diverted) as number\_of\_delays\_for\_airport,

airline\_name

from airlines\_json\_view aj LATERAL VIEW explode(split(statistics.carriers.names, ',')) cr AS airline\_name

where aj.Time.Year = 2015

""").show(10)

Result\_1.createOrReplaceTempView("Result\_1\_view")

#remove show before creating a view

Result\_2 = spark.sql(f"""

select

r1.year\_month,

r1.airport\_code,

r1.number\_of\_delays\_for\_airport,

r1.airline\_name,

IFNULL(ac.IATA\_CODE,'N/A') as airline\_iata\_code

from Result\_1\_view r1

left join

airlines\_csv\_view ac

on r1.airline\_name=ac.AIRLINE

""").show(10)

Result\_2.createOrReplaceTempView("Result\_2\_view")

#remove show before creating a view

Result\_3 = spark.sql(f"""

select

date\_format(concat(YEAR,'-',MONTH),'yyyy-MM') as year\_month,

AIRLINE,

case

when (DEPARTURE\_DELAY > 0 AND ARRIVAL\_DELAY > 0) OR (CANCELLED = true) THEN ORIGIN\_AIRPORT

when (DEPARTURE\_DELAY <= 0 AND ARRIVAL\_DELAY > 0) OR (DIVERTED = true) THEN DESTINATION\_AIRPORT

else 'N/A' end AS delayed\_airport

from flights

""").show(10)

Result\_3.createOrReplaceTempView("Result\_3\_view")

#remove show before creating a view

Result\_4 = spark.sql(f"""

select

year\_month,

AIRLINE,

delayed\_airport,

count(\*) as number\_of\_delays\_for\_airline\_in\_airport

from Result\_3\_view

group by year\_month,

AIRLINE,

delayed\_airport

""").show(10)

Result\_4.createOrReplaceTempView("Result\_4\_view")

#remove show before creating a view

Result\_5 = spark.sql(f"""

select

r2.year\_month,

r2.airport\_code,

r2.number\_of\_delays\_for\_airport,

r2.airline\_name,

r2.airline\_iata\_code,

case

when r4.number\_of\_delays\_for\_airline\_in\_airport = 0 and r2.airline\_iata\_code = 'N/A' then NULL

else r4.number\_of\_delays\_for\_airline\_in\_airport end as number\_of\_delays\_for\_airline\_in\_airport

from Result\_2\_view r2

left join Result\_4\_view r4

on r2.year\_month = r4.year\_month

and r2.airline\_iata\_code = r4.AIRLINE

and r2.airport\_code = r4.delayed\_airport

""").show(10)

#FINAL RESULT

Result\_5.createOrReplaceTempView("Result\_5\_view")

#remove show before creating a view

#Compare results with golden set 2 and write to parquet

df\_ref2 = spark.read.parquet("Query\_2")

df\_ref2.createOrReplaceTempView("reference2")

Comp\_test10=Result\_5.subtract(df\_ref2)

Comp\_test10.write.parquet('SQL\_Flights\_diff')

#Make an archive of parquet files to download

CLI:

C:\Users\Olga\_Seregina>docker exec b93170698b60 tar czvf SQL\_flights\_diff.tar.gz SQL\_Flights\_diff

#Write the difference to excel file

import pandas

pd\_obj =pandas.read\_parquet('SQL\_Flights\_diff', engine='pyarrow')

pd\_obj.to\_excel(r'SQL\_Flights\_diff.xlsx', index = False)

Comparison results in parquet can be found in SQL\_flights\_diff.tar.gz (attached)

Comparison results in excel can be found in SQL\_Flights\_diff.xlsx (attached)

# Spark Dataframe API

## Query #1

#Prepare Spark

from pyspark.context import SparkContext

from pyspark.sql.session import SparkSession

sc = SparkContext('local')

spark = SparkSession(sc)

from pyspark.sql import Window

from pyspark.sql.functions import \*

#Create UDFs

import datetime

def minute\_diff(start,end):

if start is None:

return None

if end is None:

return None

try:

startdate = datetime.datetime.strptime(start,'%H:%M:%S')

enddate = datetime.datetime.strptime(end,'%H:%M:%S')

except ValueError:

return None

timediff = enddate-startdate

diff\_in\_min = int(timediff.total\_seconds()//60)

if diff\_in\_min < 0:

diff\_in\_min += 1440

return(diff\_in\_min)

spark.udf.register("OS\_minute\_diff", minute\_diff)

OS\_minute\_diff = udf(minute\_diff)

def ConvertTime(s):

if s is None:

return "00:00:00"

return s[0:2] + ":" + s[2:4] + ":00"

spark.udf.register("OS\_ConvertTime", ConvertTime)

ConvertTime\_udf = udf(ConvertTime)

#Create dataset

API\_df\_flights=spark.read.csv("flights.csv", header=True)

join\_group = ["YEAR", "MONTH", "DAY", "AIRLINE"]

API\_df\_flights\_1 = API\_df\_flights.withColumn("departure\_date",date\_format(concat\_ws("-","YEAR","MONTH","DAY"),'yyyy-MM-dd')) \

.withColumn("departure\_time",ConvertTime\_udf("DEPARTURE\_TIME")) \

.withColumn("daily\_flight\_serial\_number",row\_number().over(Window.partitionBy("YEAR","MONTH","DAY").orderBy("DEPARTURE\_TIME"))) \

.withColumn("time\_since\_previous\_departure", OS\_minute\_diff( lag("departure\_time",1,0).over(Window.partitionBy("AIRLINE").orderBy("YEAR","MONTH","DAY","DEPARTURE\_TIME")),"departure\_time"))

API\_df\_flights\_cnt = API\_df\_flights.groupBy("YEAR","MONTH","DAY","AIRLINE").count().withColumnRenamed("count","airline\_daily\_flights\_count")

API\_df\_flights\_1 = API\_df\_flights\_1.join(API\_df\_flights\_cnt,join\_group)

API\_df\_flights\_2 = API\_df\_flights\_1.select(API\_df\_flights\_1.departure\_date,

API\_df\_flights\_1.departure\_time,

API\_df\_flights\_1.AIRLINE.alias("airline"),

API\_df\_flights\_1.FLIGHT\_NUMBER.alias("flight\_number"),

API\_df\_flights\_1.daily\_flight\_serial\_number,

API\_df\_flights\_cnt.airline\_daily\_flights\_count,

API\_df\_flights\_1.time\_since\_previous\_departure).orderBy("YEAR","MONTH","DAY","DEPARTURE\_TIME")

API\_df\_flights\_2.persist()

API\_df\_flights\_2.show(20)

#Compare the result and write them to parquet

API\_Flights\_res=API\_df\_flights\_2.subtract(Reference\_airlines)

API\_Flights\_res.write.parquet('API\_Airlines\_diff')

#Make an archive to download

C:\Users\Olga\_Seregina>docker exec b93170698b60 tar czvf API\_Airlines\_diff.tar.gz API\_Airlines\_diff

#Write to excel

import pandas

pd\_obj =pandas.read\_parquet('API\_Airlines\_diff', engine='pyarrow')

pd\_obj.to\_excel(r'API\_Airlines\_diff.xlsx', index = False)

Comparison results in parquet can be found in API\_Airlines\_diff.tar.gz (attached)

Comparison results in excel can be found in API\_Airlines\_diff.xlsx (attached)

## Query #2

#Prepare Spark

from pyspark.context import SparkContext

from pyspark.sql.session import SparkSession

sc = SparkContext('local')

spark = SparkSession(sc)

from pyspark.sql import Window

from pyspark.sql.functions import \*

#Create UDFs

def checkIATA(s):

if s is None:

return "N/A"

return s

checkIATA\_udf = udf(checkIATA)

def check\_delay(oa,da,dd,ad,c,d):

if dd is None:

return "dd N/A"

if ad is None:

return "N/A"

if c is None:

return "N/A"

if d is None:

return "N/A"

if (int(dd) > 0 and int(ad) > 0) or c == 1:

result = oa

elif (int(dd) <= 0 and int(ad) > 0 ) or d ==1:

result = da

else:

result = "N/A"

return result

check\_delay\_udf = udf(check\_delay)

def checkNumberOfDelays(delays,iata):

if delays == 0 and iata == "N/A":

return Null

return delays

checkNumberOfDelays\_udf = udf(checkNumberOfDelays)

#Read files

df\_json = spark.read.json("airlines.json", multiLine=True)

df\_airl\_csv = spark.read.csv("airlines.csv", header=True)

#Create dataset

f\_result\_1 = df\_json.withColumn("year\_month",date\_format(concat\_ws("-","Time.YEAR","Time.MONTH"),'yyyy-MM')) \

.withColumn("airport\_code",col("Airport.Code")) \

.withColumn("number\_of\_delays\_for\_airport", col("Statistics.Flights.Cancelled") + col("Statistics.Flights.Delayed") + col("Statistics.Flights.Diverted")) \

.withColumn("airline\_name",explode(split("Statistics.Carriers.Names",",")))

f\_result\_2 = f\_result\_1.filter(col("Time.YEAR") == 2015)

f\_result\_3 = f\_result\_2.join(df\_airl\_csv,f\_result\_2.airline\_name == df\_airl\_csv.AIRLINE,"left").withColumn("airline\_iata\_code", checkIATA\_udf(col("IATA\_CODE")))

df\_delayed\_airports = API\_df\_flights.withColumn("flights\_year\_month",date\_format(concat\_ws("-","YEAR","MONTH"),'yyyy-MM')) \

.withColumn("flights\_airline",col("AIRLINE")) \

.withColumn("flights\_delayed\_airport",check\_delay\_udf("ORIGIN\_AIRPORT","DESTINATION\_AIRPORT","DEPARTURE\_DELAY","ARRIVAL\_DELAY","CANCELLED","DIVERTED")) \

.select("flights\_year\_month","flights\_airline","flights\_delayed\_airport") \

.groupBy("flights\_year\_month","flights\_airline","flights\_delayed\_airport") \

.count().withColumnRenamed("count","flights\_number\_of\_delays\_for\_airline\_in\_airport")

f\_result\_4 = f\_result\_3.join(df\_delayed\_airports,(f\_result\_3.airline\_iata\_code == df\_delayed\_airports.flights\_airline) & (f\_result\_3.year\_month == df\_delayed\_airports.flights\_year\_month) & (f\_result\_3.airport\_code == df\_delayed\_airports.flights\_delayed\_airport),"left")

f\_result\_5 = f\_result\_4.withColumn("number\_of\_delays\_for\_airline\_in\_airport",checkNumberOfDelays\_udf("flights\_number\_of\_delays\_for\_airline\_in\_airport","airline\_iata\_code")).select("year\_month","airport\_code","number\_of\_delays\_for\_airport","airline\_name","airline\_iata\_code","number\_of\_delays\_for\_airline\_in\_airport")

f\_result\_5.persist().show()

#Compare the result and write them to parquet

API\_Flights\_res= f\_result\_5.subtract(df\_ref2)

API\_Flights\_res.write.parquet('API\_Flights\_diff')

#Make an archive to download

C:\Users\Olga\_Seregina>docker exec b93170698b60 tar czvf API\_Flights\_diff.tar.gz API\_Flights\_diff

#Write to excel

import pandas

pd\_obj =pandas.read\_parquet('API\_Flights\_diff', engine='pyarrow')

pd\_obj.to\_excel(r'API\_Flights\_diff.xlsx', index = False)

Comparison results in parquet can be found in API\_Flights\_diff.tar.gz (attached)

Comparison results in excel can be found in API\_Flights\_diff.xlsx (attached)