Task #1

**import org.apache.spark.sql.functions.\_**

**val flights = spark.read.option("header",true)**

**.csv("data/flights.csv")**

**val result = flights.groupBy("ORIGIN\_AIRPORT")**

**.agg(sum("DEPARTURE\_DELAY")**

**.as("TOTAL\_DELAY"))**

**.orderBy(desc("TOTAL\_DELAY"))**

**result.show(numRows=1)**

**flights.createOrReplaceTempView("flights")**

**val resultSQL = spark.sql("""SELECT ORIGIN\_AIRPORT, SUM(DEPARTURE\_DELAY) as TOTAL\_DELAY**

**FROM flights**

**GROUP BY ORIGIN\_AIRPORT**

**ORDER BY TOTAL\_DELAY DESC""")**

**resultSQL.show(numRows=1)**

Text

Description automatically generated with medium confidence

TASK#2

**val result = spark.sql("""**

**WITH temp AS(SELECT**

**TO\_DATE(CONCAT (YEAR,"-",MONTH,"-",DAY)) as departure\_date,**

**CASE WHEN ISNULL(departure\_time) THEN "00:00:00"**

**ELSE coalesce(DATE\_FORMAT(TO\_TIMESTAMP(departure\_time, "HHmm"), "HH:mm:ss"), "00:00:00")**

**END as departure\_time,**

**airline, flight\_number**

**FROM flights**

**)**

**SELECT departure\_date, departure\_time, airline, flight\_number,**

**ROW\_NUMBER() OVER(PARTITION BY departure\_date ORDER BY departure\_time) AS daily\_flight\_serial\_number,**

**COUNT(\*) OVER(PARTITION BY departure\_date, airline) AS airline\_daily\_flights\_count,**

**cast((int(to\_timestamp(departure\_time)) - int(to\_timestamp(lag(departure\_time) OVER(PARTITION BY departure\_date, airline ORDER BY departure\_time))))/(60) as int) as time\_since\_previous\_departure**

**FROM temp**

**Order BY departure\_date, departure\_time**

**""")**

**result.show(numRows=10)**

**result.count()**

**Table

Description automatically generated with low confidence**

**var golden1 = spark.read.parquet("query1")**

**golden1.createOrReplaceTempView("goldenData")**

**result.createOrReplaceTempView("kaggleData")**

**val diff = spark.sql("""**

**SELECT \* FROM(**

**SELECT \* FROM kaggleData**

**UNION ALL**

**SELECT \* FROM goldenData)**

**GROUP BY departure\_date,**

**departure\_time,**

**airline,**

**flight\_number,**

**airline\_daily\_flights\_count,**

**daily\_flight\_serial\_number,**

**time\_since\_previous\_departure**

**having count(\*) = 1**

**""")**

**Table

Description automatically generated with low confidence**

**diff.toDF()**

**diff.write.mode("overwrite").csv("logCSV/task2")**

**diff.write.mode("overwrite").parquet("logPARQ/task2")**

**TASK2 DF API**

**import org.apache.spark.sql.expressions.Window**

**import org.apache.spark.sql.functions.\_**

**val resultDF = flights.select($"airline",**

**$"flight\_number",**

**when(col("departure\_time").isNull, "00:00:00")**

**.otherwise(date\_format(to\_timestamp($"departure\_time", "HHmm"), "HH:mm:ss")).alias("departure\_time"),**

**when(col("YEAR" ).isNull || col("MONTH").isNull || col("DAY").isNull , "2015-01-01")**

**.otherwise(**

**to\_date(concat($"YEAR", lit("-"), $"MONTH", lit("-"), $"DAY"), "yyyy-M-d").as("departure\_date")).alias("departure\_date"))**

**.select($"airline",**

**$"flight\_number",**

**$"departure\_date",**

**$"departure\_time",**

**row\_number.over(Window.partitionBy($"departure\_date").orderBy($"departure\_time")).as("daily\_flight\_serial\_number"),**

**count("\*").over(Window.partitionBy($"departure\_date", $"airline")).as("airline\_daily\_flights\_count"),**

**expr("""cast((int(to\_timestamp(departure\_time)) - int(to\_timestamp(lag(departure\_time) OVER(PARTITION BY departure\_date, airline ORDER BY departure\_time))))/(60) as int) as time\_since\_previous\_departure**

**"""))**

**.orderBy("departure\_date", "daily\_flight\_serial\_number")**

**Table

Description automatically generated**

**val dfDiff = golden1.unionAll(resultDF).except(resultDF.intersect(golden1))**

**dfDiff.write.mode("overwrite").csv("logCSV/task2DF")**

**dfDiff.write.mode("overwrite").parquet("logPARQ/task2DF")**

**dfDiff.count()**

**Table

Description automatically generated with medium confidence**

**Task 3**

**flights.schema.fields**

**.filter(f => f.dataType.typeName == "string")**

**.foreach(f=>println(f.name + "\t" + f.dataType.typeName + "\t" + flights.where(col(f.name).isNull).count()))**

**Text

Description automatically generated with medium confidence**