Lab3-Task3: Guided solution for Aggregation analysis

Overview

In this exercise, we use Spark to perform various aggregate analyses on a dataset containing information about flights and airports. The analyses include finding the top 10 airports by number of departures and arrivals, the top 10 most common flight tracks, the top 10 delays by various categories, and identifying flights that are getting shorter over time

1. We load the transformed flights and airports data from their respective S3 directories.
2. We examine the schema of these datasets for better understanding.
3. Various aggregate analyses are performed using Spark's DataFrame API.

Step 1: Initialization and Data Loading

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as F

spark=SparkSession.builder.master("local").appName('ex3_aggregations').getOrCreate()

flights_df = spark.read.parquet('s3a://spark/data/transformed/flights/')
airports_df = spark.read.parquet('s3a://spark/data/source/airports/')

flights_df.printSchema()
airports_df.printSchema()
```

Breakdown:

- **Initialize Spark Session.**
- **Data Loading:** spark.read.parquet() reads a Parquet file into a DataFrame. We do this twice, once for each dataset (flights_df and airports_df).
- **Schema Inspection:** printSchema() is used to inspect the schema of the loaded DataFrames. This is often useful for understanding the data types and structure of your tables.

Step 2: Top 10 Airports by Number of Departures

```
flights_df \
    .groupBy(F.col('origin_airport_id').alias('airport_id')) \
    .agg(F.count(F.lit(1)).alias('number_of_departures')) \
    .join(airports_df.select(F.col('airport_id'), F.col('name').alias('airport_name')),
['airport_id']) \
    .orderBy(F.col('number_of_departures').desc()) \
    .show(10, False)
```

Breakdown:

- **groupBy():** We start by grouping the data by the **origin_airport_id**. We use an alias to rename it to **airport_id**.
- **agg():** We then aggregate each group by counting the number of rows using **F.count(F.lit(1))**. This is simply counting the number of rows in each group and aliasing it as **number_of_departures**.
- **join():** We then join this aggregated DataFrame with **airports_df** to get the airport names.
- **orderBy():** The results are ordered by **number_of_departures** in descending order.
- **show():** Finally, the top 10 results are displayed.

Step 3: Top 10 Airports by Number of Arrivals

This part is nearly identical to the "number of departures" query, except that we group by **dest_airport_id** (destination airport) instead of **origin_airport_id**.

```
flights_df \
   .groupBy(F.col('dest_airport_id').alias('airport_id')) \
   .agg(F.count(F.lit(1)).alias('number_of_arrivals')) \
   .join(airports_df.select(F.col('airport_id'), F.col('name').alias('airport_name')),
['airport_id']) \
   .orderBy(F.col('number_of_arrivals').desc()) \
   .show(10, False)
```

Step 4: Top 10 Most Common Flight Tracks

```
flights_df \
  .groupBy(F.col('origin_airport_id').alias('source_airport_id'), F.col('dest_airport_id').alias('dest_airport_id')) \
  .agg(F.count(F.lit(1)).alias('number_of_tracks')) \
  .join(airports_df.select(F.col('airport_id').alias('source_airport_id'),
                    F.col('name').alias('source_airport_name')),
     ['source_airport_id']) \
  .join(airports_df.select(F.col('airport_id').alias('dest_airport_id'),
                    F.col('name').alias('dest_airport_name')),
     ['dest_airport_id']) \
  .select(F.concat(F.col('source_airport_name'), F.lit(' -> '), F.col('dest_airport_name')).alias('track'),
       F.col('number_of_tracks')) \
  .orderBy(F.col('number_of_tracks').desc()) \
  .show(10, False)
```

Breakdown

- **groupBy():** Groups the flights_df DataFrame by two columns - origin_airport_id and dest_airport_id.
- **F.col():** Selects a column in a DataFrame (Spark function).
- **alias():** Renames the column to a more descriptive name for later reference (Spark function).
- **agg():** Aggregates the groups created by groupBy().
- F.**count():** Counts the number of rows in each group (Spark function).
- F.**lit(1):** Creates a literal column with value 1 for each row, which is then counted (Spark function).

- **Join with Source Airport Names**

```
.join(airports_df.select(F.col('airport_id').alias('source_airport_id'),
F.col('name').alias('source_airport_name')), ['source_airport_id'])
```

1. **join():** Joins the aggregated DataFrame with airports_df to get the names of the source airports.
2. **select():** Selects specific columns from airports_df (Spark function).

- Join with Destination Airport Names

```
.join(airports_df.select(F.col('airport_id').alias('dest_airport_id'),
F.col('name').alias('dest_airport_name')), ['dest_airport_id'])
```

Similar to the previous join but for destination airports.

- Generate Full Track Names and Sort

```
.select(F.concat(F.col('source_airport_name'), F.lit(' -> '), F.col('dest_airport_name')).alias('track'), F.col('number_of_tracks'))

.orderBy(F.col('number_of_tracks').desc())
```

- select(): Creates a new DataFrame with specific columns.
- F.concat(): Concatenates multiple columns or values into a single column (Spark function).
- orderBy(): Sorts the DataFrame by the column number_of_tracks in descending order (Spark function).

Step 5: Full Code Solution

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as F

spark = SparkSession.builder.master("local").appName('ex3_aggregations').getOrCreate()

flights_df = spark.read.parquet('s3a://spark/data/transformed/flights/')
airports_df = spark.read.parquet(' s3a://spark/data/source/airports/')

flights_df.printSchema()
airports_df.printSchema()

flights_df \
    .groupBy(F.col('origin_airport_id').alias('airport_id')) \
    .agg(F.count(F.lit(1)).alias('number_of_departures')) \
    .join(airports_df.select(F.col('airport_id'), F.col('name').alias('airport_name')), ['airport_id']) \
    .orderBy(F.col('number_of_departures').desc()) \
    .show(10, False)

flights_df \
    .groupBy(F.col('dest_airport_id').alias('airport_id')) \
    .agg(F.count(F.lit(1)).alias('number_of_arrivals')) \
    .join(airports_df.select(F.col('airport_id'), F.col('name').alias('airport_name')), ['airport_id']) \
    .orderBy(F.col('number_of_arrivals').desc()) \
    .show(10, False)

flights_df \
    .groupBy(F.col('origin_airport_id').alias('source_airport_id'), F.col('dest_airport_id').alias('dest_airport_id')) \
    .agg(F.count(F.lit(1)).alias('number_of_tracks')) \
    .join(airports_df.select(F.col('airport_id').alias('source_airport_id'),
                             F.col('name').alias('source_airport_name')),
          ['source_airport_id']) \
    .join(airports_df.select(F.col('airport_id').alias('dest_airport_id'),
                             F.col('name').alias('dest_airport_name')),
          ['dest_airport_id']) \
    .select(F.concat(F.col('source_airport_name'), F.lit(' -> '), F.col('dest_airport_name')).alias('track'),
            F.col('number_of_tracks')) \
    .orderBy(F.col('number_of_tracks').desc()) \
    .show(10, False)

spark.stop()
```