Lab2-Task2: Guided solution for data Ingestion with PySpark: Flights

Overview

read data from a CSV file named **flights.csv**, transform the DataFrame to adhere to relational database norms, and write it back to S3 in the Parquet file format.

Step 1: Initialize the Spark Session

Create a Python script named data_parser_flights.py in a folder called exercises_two.

```
# You initiate a Spark session to use Spark SQL's DataFrame API.
# A Spark Session is a combined entry point of Spark Context and SQL Context
from pyspark.sql import functions as F
from pyspark.sql import types as T
from pyspark.sql import SparkSession

spark =SparkSession.builder\
.master("local")\
.appName('ex2_flights')\
.getOrCreate()
```

Step 2: Read the Data

```
# Utilize Spark's read.csv method to read the CSV files from the S3 location.
#We specify that the CSV has a header, so the first row will be used as column names.

flights_raw_df = spark.read.csv('s3a://spark/data/raw/flights/', header=True)
```

Step 3: Data Transformation

```
flight_df = flights_raw_df.select(
    F.col('DayofMonth').cast(T.IntegerType()).alias('day_of_month'),
    F.col('DayOfWeek').cast(T.IntegerType()).alias('day_of_week'),
    F.col('Carrier').alias('carrier'),
    F.col('OriginAirportID').cast(T.IntegerType()).alias('origin_airport_id'),
    F.col('DestAirportID').cast(T.IntegerType()).alias('dest_airport_id'),
    F.col('DepDelay').cast(T.IntegerType()).alias('dep_delay'),
    F.col('ArrDelay').cast(T.IntegerType()).alias('arr_delay'))
```

Step 4: Write Data to S3 in Parquet Format

```
# write this transformed DataFrame back to S3 in Parquet format.

flight_df.write.parquet('s3a://spark/data/source/flights/', mode='overwrite')
```

Techniques:

- Parquet is an efficient columnar storage format.
- The mode 'overwrite' is specified to replace the existing data if it exists.

Step 5: Stop the Spark Session

```
# terminate the Spark session to release its resources
# a good practice to clean up after your operations
spark.stop()
```

Upon successful completion of these steps, you will have ingested data from S3, performed basic transformations, and stored it back into S3 in a more efficient format. You've essentially built a rudimentary data pipeline!

Step 6 - Full Code Solution

Data Parsing - flights
Folder Name: exercises_two
File Name: data_parser_flights.py

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

spark = SparkSession.builder.master("local").appName('ex2_flights').getOrCreate()


flights_raw_df = spark.read.csv('s3a://spark/data/raw/flights/', header=True)


flight_df = flights_raw_df.select(
    F.col('DayofMonth').cast(T.IntegerType()).alias('day_of_month'),
    F.col('DayOfWeek').cast(T.IntegerType()).alias('day_of_week'),
    F.col('Carrier').alias('carrier'),
    F.col('OriginAirportID').cast(T.IntegerType()).alias('origin_airport_id'),
    F.col('DestAirportID').cast(T.IntegerType()).alias('dest_airport_id'),
    F.col('DepDelay').cast(T.IntegerType()).alias('dep_delay'),
    F.col('ArrDelay').cast(T.IntegerType()).alias('arr_delay'))

flight_df.write.parquet('s3a://spark/data/source/flights/', mode='overwrite')

spark.stop()
```