Naya College

Cloud Data Engineering

Apache Spark

Lab2: Data Ingestion with PySpark

NAYA College

Contents

Pre-requisites

Make sure the relevant files are copied into the S3 directories specified.
The directories would be:

- **/data/raw/flights/**,
- **/data/raw/flights_raw/**, and
- **/data/raw/airports/**.

Task1: Airpots

Objective:

In this exercise, you will leverage the power of PySpark for data manipulation .
our primary objective is to read raw data related to airports, process it by selecting certain fields ,
and then save the cleaned data in the efficient Parquet format.

Instructions:

Setting Up Your Workspace in PyCharm:

- Open PyCharm: Launch the PyCharm IDE on your computer.
- Create a New Folder: Right-click on the left project pane within your project. Select New > Directory and name it exercises_two.
- Create a Python Script: Inside the exercises_two folder, right-click and select New > Python File. Name this script data_parser_airports.

Import the Required Libraries:

1. At the start of the data_parser_airports.py script, you need to import the necessary libraries for:
   - PySpark SQL functionalities (including SparkSession).
   - PySpark SQL functions and types.

Initialize the SparkSession:

- Create Spark session using the following code:

```
spark=SparkSession.builder.master("local").appName('ex2_airports').getOrCreate()
```

Read the Raw Data from S3:

- Using the read.csv method, fetch the airport data from the S3 path:

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

Data Transformation:

• Select Relevant Columns: Choose only the columns of interest from the raw data. Specifically:

- Convert airport_id to integer type and keep its name as airport_id.
- Use the city column as it is.
- Use the state column as it is.
- Use the name column as it is.

Save Processed Data:

• Write the transformed dataframe airports_df into Parquet format using the following code:

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

Expected Outcome:

After executing the script, the raw airport data will be parsed, transformed, and stored in Parquet format in the specified S3 directory.

Skills Practiced:

- Using SparkSession to read and write data.
- Manipulating data frames with PySpark SQL functions.
- Casting data types.
- Writing data to S3 in Parquet format.

After following these instructions, you will have successfully parsed the raw airport data and saved it in an organized, efficient format.

Task2: Flights

Objective:

In this exercise, you will dive deeper into PySpark's capabilities for data wrangling and manipulation. Your main goal is to read raw data related to flights, process it by selecting and transforming certain fields, and then save the processed data in the efficient Parquet format.

Instructions:

Setting Up Your Workspace in PyCharm:

- **Open PyCharm:** Launch the PyCharm IDE on your computer.
- **Create a New Folder:** Right-click on the left project pane within your project. Select **New** > **Directory** and name it **exercises_two**.
- **Create a Python Script:** Inside the **exercises_two** folder, right-click and select **New** > **Python File**. Name this script **data_parser_flights**.

Import the Required Libraries:

At the start of the data_parser_flights.py script, you need to import necessary libraries:

- PySpark SQL functionalities (including SparkSession).
- PySpark SQL functions and types.
-

Initialize the SparkSession:

Create or retrieve an existing Spark session:

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

Read the Raw Data from S3

Using the read.csv method, fetch the flight data from the given S3 path:

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

Data Transformation:

- Select Relevant Columns: From the raw data, you want to select only specific columns of interest.
- Rename Columns: This makes the dataset more readable and aids in better understanding. Renamed columns will have a new alias to ensure clarity in your final dataframe.

- Cast Specific Columns: This step involves converting or specifying certain data types for your columns. The right data types help in efficient processing and ensuring data integrity.
  Specifically:
  - Convert DayofMonth to integer type and rename it to day_of_month.
  - Convert DayOfWeek to integer type and rename it to day_of_week.
  - Use the Carrier column as it is but rename it to carrier.
  - Convert OriginAirportID to integer type and rename it to origin_airport_id.
  - Convert DestAirportID to integer type and rename it to dest_airport_id.
  - Convert DepDelay to integer type and rename it to dep_delay.
  - Convert ArrDelay to integer type and rename it to arr_delay.

Save Processed Data

Write the transformed data frame **flight_df** into Parquet format.
Parquet is a popular columnar storage file format, ensuring efficiency and performance.
The data will be saved in the specified S3 directory: **/data/source/flights/**

Expected Outcome:

After running the script, the raw flight data will be parsed, transformed, and saved in Parquet format in the specified S3 directory.

Skills Practiced:

Using SparkSession to read and write data.
Manipulating data frames using PySpark SQL functions.
Casting data types and renaming columns.
Writing data to the S3 in Parquet format.

Task3: Flights Raw Data Parsing

Objective:

In this exercise, you will explore PySpark's capabilities for data wrangling. Your primary task is to read raw data related to flights, select and transform particular fields, and then save the cleaned data in the efficient Parquet format.

Instructions:

Setting Up Your Workspace in PyCharm:

• **Open PyCharm**: Launch the PyCharm IDE on your computer.
• **Create a New Folder**: Right-click on the left project pane within your project.
Select **New > Directory** and name it **exercises_two**.
• **Create a Python Script**: Inside the **exercises_two** folder, right-click and select **New > Python File**. Name this script **flights_raw_parser**.

Import the Required Libraries:

• At the beginning of the **flights_raw_parser.py** script, import the essential libraries:
- PySpark SQL functionalities (including SparkSession).
- PySpark SQL functions and types.

Initialize the SparkSession:

• Instantiate or get an existing Spark session using the following code:

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

Read the Raw Data from S3:

• Fetch the flight data from the given S3 path using the **read.csv** method:

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

Data Transformation:

• **Select Relevant Columns**: Extract specific columns from the raw data to focus on the details of interest. Specifically:
- Convert **DayofMonth** to integer type and rename it to **day_of_month**.
- Convert **DayOfWeek** to integer type and rename it to **day_of_week**.
- Retain the **Carrier** column but rename it to **carrier**.
- Convert **OriginAirportID** to integer type and rename it to **origin_airport_id**.
- Convert **DestAirportID** to integer type and rename it to **dest_airport_id**.
- Convert **DepDelay** to integer type and rename it to **dep_delay**.
- Convert **ArrDelay** to integer type and rename it to **arr_delay**.

Save Processed Data:

• Store the processed dataframe **flight_df** into Parquet format:

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

Expected Outcome:

• After executing the script, the raw flight data will be processed, transformed, and saved in Parquet format in the specified S3 directory.

Skills Practiced:

- Using SparkSession to read and write data.
- Manipulating data frames using PySpark SQL functions.
- Casting data types and renaming columns.
- Storing data to the S3 in Parquet format.

By following these steps, you will successfully transform the raw flight data and store it in a structured, efficient format.