Word Count using Apache Spark and BigQuery on Google Cloud Dataproc

Done by: Malleeswari D

Step 1: Create Dataproc Cluster

Go to GCP Console > Dataproc and create a new cluster.
- Region: us-central1 (or your preferred region)
- Enable Component Gateway
- Ensure roles for BigQuery and Cloud Storage access are set

Step 2: Create GCS Bucket

Navigate to Cloud Storage and create a bucket.
- Example bucket name: malleeswari-temp-bucket
- Use Standard storage in the same region as your cluster

Step 3: Create BigQuery Dataset

Open BigQuery Console and create a dataset named: wordcount_ts_dataset_1

Step 4: Connect to Dataproc Master via SSH

In your cluster dashboard, click SSH on the master node to open the terminal

Step 5: Launch Spark Shell

Run the following command:

```
spark-shell
```

Wait for the Spark shell (scala>) prompt to appear.

Step 6: Spark + BigQuery Code

Paste the following Scala code block line-by-line into spark-shell:

```
import org.apache.spark.sql.SparkSession

val spark = SparkSession.builder()
  .appName("spark-bigquery-demo")
  .getOrCreate()
```

Then continue through the full script (set GCS bucket, read from BigQuery, etc.).

Step 7: Set Temporary GCS Bucket

```
spark.conf.set("temporaryGcsBucket", "datasib999")
```

Step 8: Read BigQuery Data (Shakespeare dataset)

```
val wordsDF = (spark.read.format("bigquery")
  .option("table", "bigquery-public-data:samples.shakespeare")
  .load()
  .cache())
```

Step 9: Create Temp View

```
wordsDF.createOrReplaceTempView("words")
```

Step 10: Perform Word Count

```
val wordCountDF = spark.sql(
  "SELECT word, SUM(word_count) AS word_count FROM words GROUP BY word")
wordCountDF.show()
wordCountDF.printSchema()
```

Step 11: Save to BigQuery

```
wordCountDF.write.format("bigquery")
  .option("table", "wordcount_ts_dataset_1.wordcount_output")
  .save()
```

Step 12: Read Shakespeare Data from BigQuery

```
val wordsDF = (spark.read.format("bigquery")
  .option("table", "bigquery-public-data:samples.shakespeare")
  .load()
  .cache())
```

Wait for it to load. Once it's done, continue.

Step 13: Create a Temporary View

```
wordsDF.createOrReplaceTempView("words")
```

This allows you to use SQL queries on the data.

Step 14: Run Word Count SQL Query

```
val wordCountDF = spark.sql(
  "SELECT word, SUM(word_count) AS word_count FROM words GROUP BY word")
```

This creates a new DataFrame wordCountDF containing each word and its total count.

Step 15: Display the Results (Optional)

```
wordCountDF.show()
```

And check the structure of the DataFrame:

```
wordCountDF.printSchema()
```

Step 16: Write the Results to BigQuery

```
wordCountDF.write.format("bigquery")
  .option("table", "wordcount_ts_dataset_1.wordcount_output")
  .save()
```

Suppose the table is not visible follow the below steps to complete the process

Create a bucket:

- Name: malleeswari-temp-bucket
- Location: Same as your Dataproc cluster (e.g., us-central1)
- Storage class: Standard

Step 17: Set Your New Bucket in Spark

```
spark.conf.set("temporaryGcsBucket", "malleeswari-temp-bucket")
```

Step 18: Save the DataFrame to BigQuery Again

```
wordCountDF.write.format("bigquery")
  .option("table", "wordcount_ts_dataset_1.wordcount_output")
  .save()
```

Example for word checking:

```
scala> spark.sql("""
     |   SELECT word, SUM(word_count) AS word_count
     |   FROM words
     |   WHERE word IN ('love', 'death', 'king')
     |   GROUP BY word
     | """).show()
+-----+----------+
| word|word_count|
+-----+----------+
| love|      2135|
|death|       887|
| king|      1191|
+-----+----------+
```

```
scala> spark.sql("""
     |   SELECT word, SUM(word_count) AS word_count
     |   FROM words
     |   WHERE word IN ('life', 'blood', 'night')
     |   GROUP BY word
     | """).show()
+-----+----------+
| word|word_count|
+-----+----------+
|night|       902|
| life|       892|
|blood|       696|
+-----+----------+
```

```
scala> spark.sql("""
     |   SELECT word, SUM(word_count) AS word_count
     |   FROM words
     |   WHERE word = 'revenge'
     |   GROUP BY word
     | """).show()
+-------+----------+
|   word|word_count|
+-------+----------+
|revenge|       135|
+-------+----------+
```

It is shown in shell but not shown in table because in these command we give only .show() so If we want the output should be in table also we have to give the following commands after give the prompt

```
filteredDF.write.format("bigquery")
  .option("table", "wordcount_ts_dataset_1.filtered_words_output")
  .save()
```