Retail Data Analysis

Logic for Python Script ‘spark-streaming.py’:

- ### Importing important libraries

Below lib are required to run spark application.

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
```

- ### Initializing the spark session

To start any spark application we need to initialize the Spark session with getOrCreate method.

```
spark = SparkSession.builder.appName("spark-streaming-RetailDataAnalysis").getOrCreate()
```

- ### Writing the Python functions, which contain the logic for the UDFs

- **Total Items UDF** - To calculate the number of products in every invoice I added the quantity ordered ofeach product in that invoice

```
# calculate total items in order
def total_item_count(items):
   if items is not None:
      item_count =0
      for item in items:
         item_count = item_count + item['quantity']
      return item_count
```

- **Total Cost UDF** - To calculate the total income from every invoice I needed to calculate the income from sale of each product, so I multiplied the unit price of the product with the quantity of the product purchased. The sum of this cost across the products in that invoice gives me the total cost of the order. I made sure that if the transaction is a return transaction, the total cost is negative.

```
# calculate total cost of items from order
def total_cost(items,type):
    if items is not None:
        total_cost =0
        item_price =0
    for item in items:
        item_price = (item['quantity']*item['unit_price'])
        total_cost = total_cost+ item_price
        item_price=0

    if type  == 'RETURN':
        return total_cost *-1
    else:
        return total_cost
```

- **Is Order UDF -** To determine if invoice is for an order or not I used an if-else statement

```
#check if the order is new or not
def is_a_order(type):
   return 1 if type == 'ORDER' else 0
```

- **Is Return UDF -** To determine if invoice is for a return or not I used an if-else statement

```
#check if order is return type
def is_a_return(type):
   return 1 if type == 'RETURN' else 0
```

- **Reading input data from Kafka mentioning the details of the Kafka broker, such as bootstrap server, port and topic name**

```
# Reading input data from Kafka
orderRawData = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers","18.211.252.152:9092") \
    .option("subscribe","real-time-project") \
    .option("startingOffsets", "latest")  \
    .load()
```

- **Defining JSON schema of each order, using appropriate datatypes and StructField in the case of the item attributes**

```
# Define Schema
JSON_Schema = StructType() \
    .add("invoice_no", LongType()) \
    .add("country",StringType()) \
    .add("timestamp", TimestampType()) \
    .add("type", StringType()) \
    .add("items", ArrayType(StructType([
        StructField("SKU", StringType()),
        StructField("title", StringType()),
        StructField("unit_price", FloatType()),
        StructField("quantity", IntegerType())
        ])))
```

- **Reading the raw JSON data from Kafka as 'order stream' by casting it to string and storing itinto the alias 'data'**

```
orderStream = orderRawData.select(from_json(col("value").cast("string"),
             JSON_Schema).alias("orderdata")).select("orderdata.*")
```

- **Defining the UDFs by Converting the Python functions I defined earlier, and assigning the appropriate return datatype:**

```
# calling the user defined functions as defined above
agg_isOrder = udf(is_a_order, IntegerType())
agg_isReturn = udf(is_a_return, IntegerType())
agg_total_item_count = udf(total_item_count, IntegerType())
agg_total_order_cost = udf(total_cost, FloatType())
```

- **Derive the additional columns according to the required input values:**

```
# deriving additional columns
expandedOrderStream = orderStream \
   .withColumn("total_cost",
agg_total_order_cost(orderStream.items,orderStream.type)) \
   .withColumn("total_items", agg_total_item_count(orderStream.items)) \
   .withColumn("is_order", agg_isOrder(orderStream.type)) \
   .withColumn("is_return", agg_isReturn(orderStream.type))
```

- **Writing the summarized input values to console, using ‘append’ output method and applying truncate as false and setting the processing time to 1 minute:**

```
# appending the calculated input values to console
extendedOrderQuery = expandedOrderStream \
    .select("invoice_no", "country",
"timestamp","total_cost","total_items","is_order","is_return") \
    .writeStream \
    .outputMode("append") \
    .format("console") \
    .option("truncate", "false") \
    .option("path", "Console_output") \
    .option("checkpointLocation", "Console_output_checkpoints") \
    .trigger(processingTime="1 minute") \
    .start()
```

- **Calculating time-based KPIs (Total sale volume, OPM, Rate of return, Average transaction size) having tumbling window of one minute and watermark of one minute:**

```
# Calculating time based KPI values
aggStreamByTime = expandedOrderStream \
     .withWatermark("timestamp","1 minutes") \
     .groupby(window("timestamp", "1 minute")) \
     .agg(sum("total_cost").alias("total_volume_of_sales"),
         avg("total_cost").alias("average_transaction_size"),
         count("invoice_no").alias("OPM"),
         avg("is_Return").alias("rate_of_return")) \
     .select("window.start","window.end","OPM","total_volume_of_sales","average_tr
ansaction_size","rate_of_return")
```

- **Writing the time-based KPIs data to HDFS - HDFS into JSON files for each one-minute window, using 'append' output mode, setting truncate as false, and specifying the HDFS output path for both the KPI files and for their checkpoints. Ten 1-minute window batches were taken:**

```
# Writing to the Console and hadoop location for Time based KPI values
queryByTime = aggStreamByTime.writeStream \
    .format("json") \
    .outputMode("append") \
    .option("truncate", "false") \
    .option("path", "timeKPIvalue") \
    .option("checkpointLocation", "timeKPIvalue_checkpoints") \
    .trigger(processingTime="1 minutes") \
    .start()
```

- **Calculating and writing time-and-country-based KPIs (Total sale volume, OPM, Rate of return) having tumbling window of one minute and watermark of one minute. Here I grouped by window and country both:**

```
# Calculating Time and country based KPIs values
aggStreamByCountry = expandedOrderStream \
    .withWatermark("timestamp", "1 minutes") \
    .groupBy(window("timestamp", "1 minutes"), "country") \
    .agg(sum("total_cost").alias("total_volume_of_sales"),
        count("invoice_no").alias("OPM"),
        avg("is_Return").alias("rate_of_return")) \
    .select("window.start","window.end","country",
"OPM","total_volume_of_sales","rate_of_return")
```

```
# WWriting to the Console and hadoop location for Time and country based KPI
values
ByTime_country = aggStreamByCountry.writeStream \
    .format("json") \
    .outputMode("append") \
    .option("truncate", "false") \
    .option("path", "time_countryKPIvalue") \
    .option("checkpointLocation", "time_countryKPIvalue_checkpoints") \
    .trigger(processingTime="1 minutes") \
    .start()
```

- **Indicating Spark to await termination**

```
# Spark to await for termination
extendedOrderQuery.awaitTermination()
queryByTime.awaitTermination()
queryByCountry.awaitTermination()
```

Console Commands

- Started by logging into the emr instance as 'hadoop' user (Used emr-5.30.1)

- Create the 'spark-streaming.py' file having the code discussed above
  `vi spark-streaming.py`

- set the Kafka Version using the following command
  `export SPARK_KAFKA_VERSION=0.10`

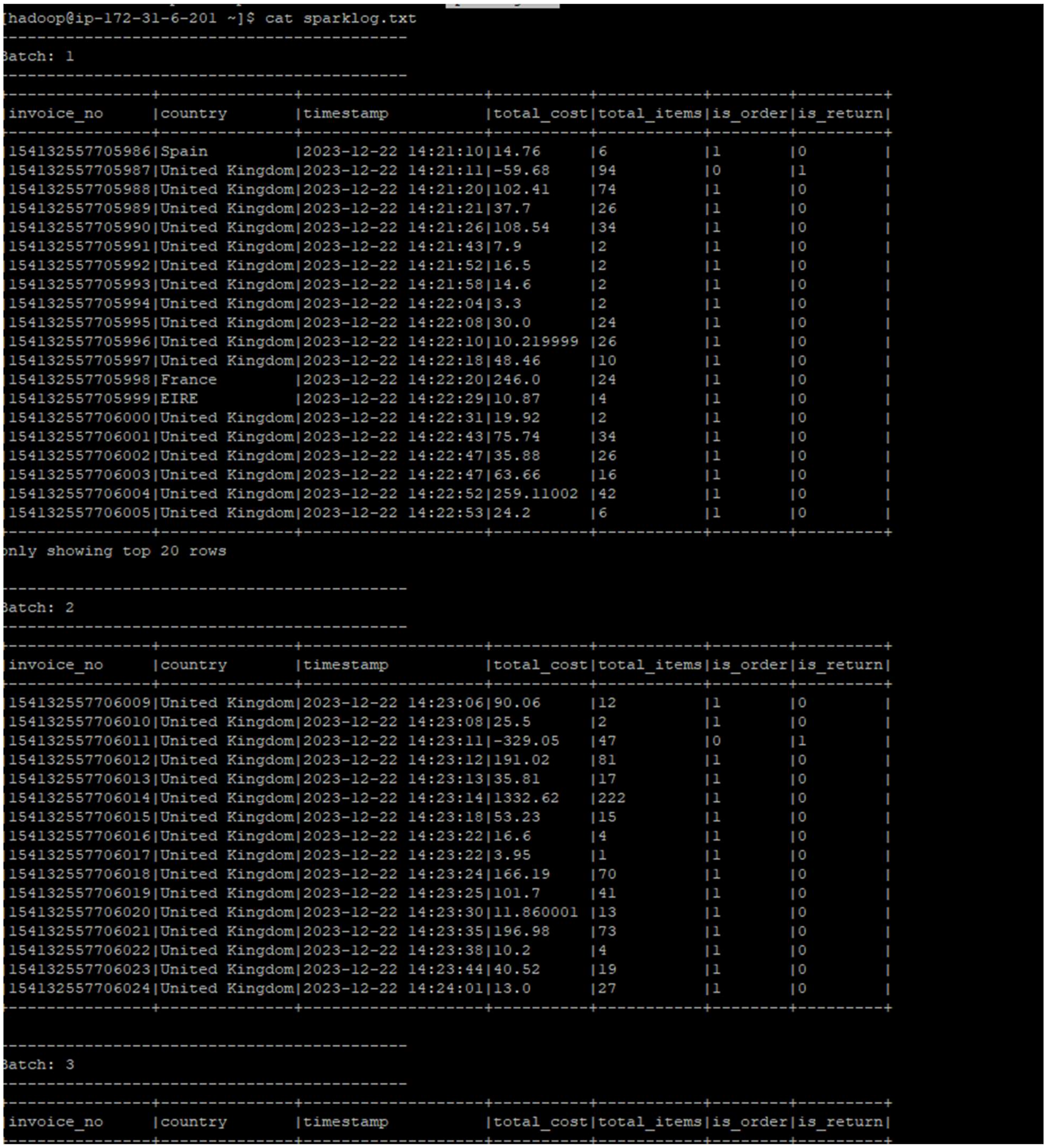
- Run the spark-submit command, specifying the jar and python file
  spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 spark-streaming.py

If you are using latest EMR (emr-6.*) and Spark you may face the error (py4j.protocol.Py4JJavaError: An error occurred while calling o69.load.), to resolve it follow the below steps.

- Download the Spark-SQL-Kafka jar file. This jar is used to run the Spark Streaming-Kafka codes
  wget https://ds-spark-sql-kafka-jar.s3.amazonaws.com/spark-sql-kafka-0-10_2.11-2.3.0.jar

- Ran the spark2-submit command, specifying the jar and python file
  spark2-submit --jars spark-sql-kafka-0-10_2.11-2.3.0.jar spark-streaming.py

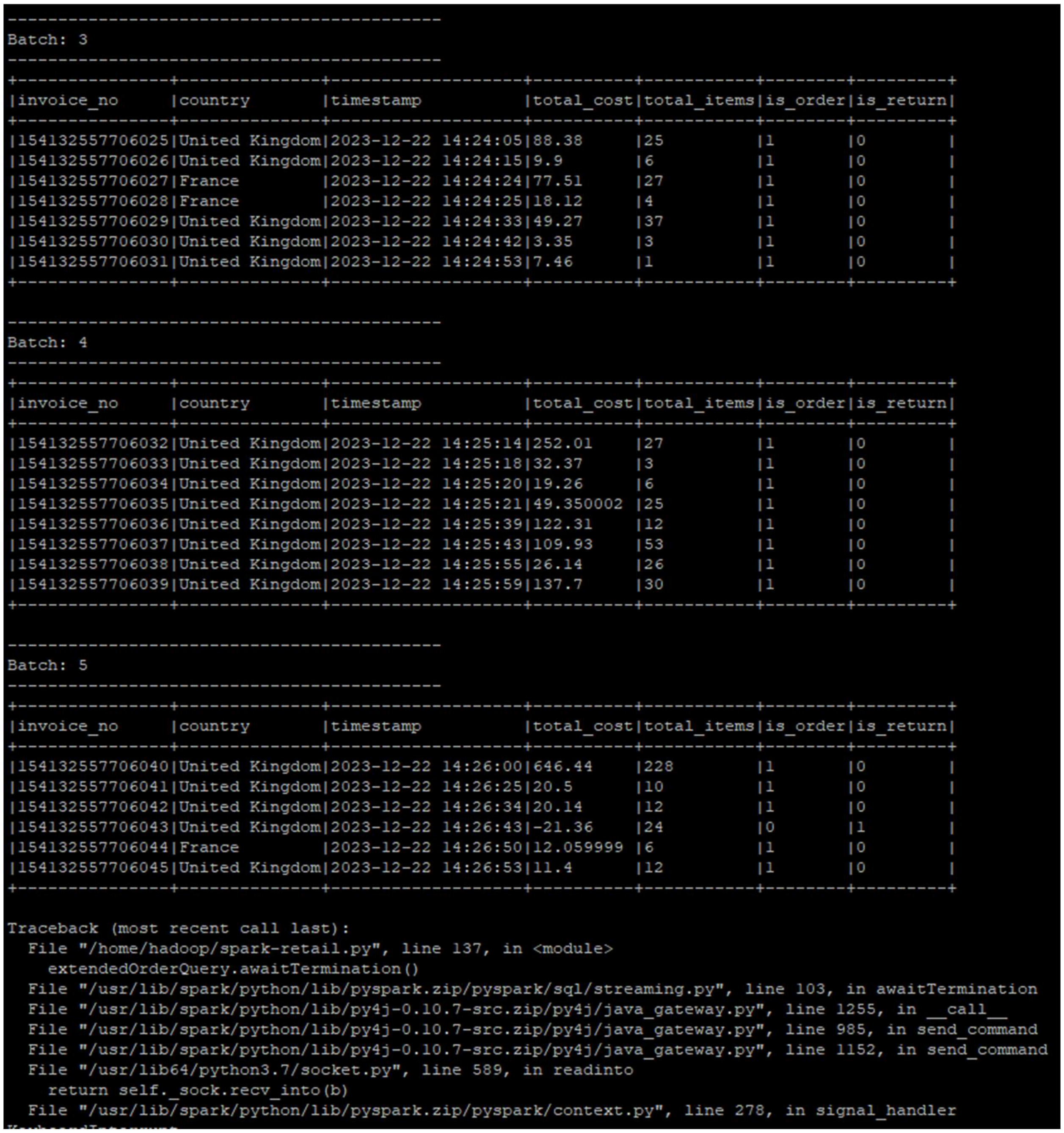
Output:



```
[hadoop@ip-172-31-6-201 ~]$ cat sparklog.txt
-------------------------------------------
Batch: 1
-------------------------------------------
+---------------+--------------+-------------------+----------+-----------+--------+---------+
|invoice_no     |country       |timestamp          |total_cost|total_items|is_order|is_return|
+---------------+--------------+-------------------+----------+-----------+--------+---------+
|154132557705986|Spain         |2023-12-22 14:21:10|14.76     |6          |1       |0        |
|154132557705987|United Kingdom|2023-12-22 14:21:11|-59.68    |94         |0       |1        |
|154132557705988|United Kingdom|2023-12-22 14:21:20|102.41    |74         |1       |0        |
|154132557705989|United Kingdom|2023-12-22 14:21:21|37.7      |26         |1       |0        |
|154132557705990|United Kingdom|2023-12-22 14:21:26|108.54    |34         |1       |0        |
|154132557705991|United Kingdom|2023-12-22 14:21:43|7.9       |2          |1       |0        |
|154132557705992|United Kingdom|2023-12-22 14:21:52|16.5      |2          |1       |0        |
|154132557705993|United Kingdom|2023-12-22 14:21:58|14.6      |2          |1       |0        |
|154132557705994|United Kingdom|2023-12-22 14:22:04|3.3       |2          |1       |0        |
|154132557705995|United Kingdom|2023-12-22 14:22:08|30.0      |24         |1       |0        |
|154132557705996|United Kingdom|2023-12-22 14:22:10|10.219999 |26         |1       |0        |
|154132557705997|United Kingdom|2023-12-22 14:22:18|48.46     |10         |1       |0        |
|154132557705998|France        |2023-12-22 14:22:20|246.0     |24         |1       |0        |
|154132557705999|EIRE          |2023-12-22 14:22:29|10.87     |4          |1       |0        |
|154132557706000|United Kingdom|2023-12-22 14:22:31|19.92     |2          |1       |0        |
|154132557706001|United Kingdom|2023-12-22 14:22:43|75.74     |34         |1       |0        |
|154132557706002|United Kingdom|2023-12-22 14:22:47|35.88     |26         |1       |0        |
|154132557706003|United Kingdom|2023-12-22 14:22:47|63.66     |16         |1       |0        |
|154132557706004|United Kingdom|2023-12-22 14:22:52|259.11002 |42         |1       |0        |
|154132557706005|United Kingdom|2023-12-22 14:22:53|24.2      |6          |1       |0        |
+---------------+--------------+-------------------+----------+-----------+--------+---------+
only showing top 20 rows

-------------------------------------------
Batch: 2
-------------------------------------------
+---------------+--------------+-------------------+----------+-----------+--------+---------+
|invoice_no     |country       |timestamp          |total_cost|total_items|is_order|is_return|
+---------------+--------------+-------------------+----------+-----------+--------+---------+
|154132557706009|United Kingdom|2023-12-22 14:23:06|90.06     |12         |1       |0        |
|154132557706010|United Kingdom|2023-12-22 14:23:08|25.5      |2          |1       |0        |
|154132557706011|United Kingdom|2023-12-22 14:23:11|-329.05   |47         |0       |1        |
|154132557706012|United Kingdom|2023-12-22 14:23:12|191.02    |81         |1       |0        |
|154132557706013|United Kingdom|2023-12-22 14:23:13|35.81     |17         |1       |0        |
|154132557706014|United Kingdom|2023-12-22 14:23:14|1332.62   |222        |1       |0        |
|154132557706015|United Kingdom|2023-12-22 14:23:18|53.23     |15         |1       |0        |
|154132557706016|United Kingdom|2023-12-22 14:23:22|16.6      |4          |1       |0        |
|154132557706017|United Kingdom|2023-12-22 14:23:22|3.95      |1          |1       |0        |
|154132557706018|United Kingdom|2023-12-22 14:23:24|166.19    |70         |1       |0        |
|154132557706019|United Kingdom|2023-12-22 14:23:25|101.7     |41         |1       |0        |
|154132557706020|United Kingdom|2023-12-22 14:23:30|11.860001 |13         |1       |0        |
|154132557706021|United Kingdom|2023-12-22 14:23:35|196.98    |73         |1       |0        |
|154132557706022|United Kingdom|2023-12-22 14:23:38|10.2      |4          |1       |0        |
|154132557706023|United Kingdom|2023-12-22 14:23:44|40.52     |19         |1       |0        |
|154132557706024|United Kingdom|2023-12-22 14:24:01|13.0      |27         |1       |0        |
+---------------+--------------+-------------------+----------+-----------+--------+---------+

-------------------------------------------
Batch: 3
-------------------------------------------
+---------------+--------------+-------------------+----------+-----------+--------+---------+
|invoice_no     |country       |timestamp          |total_cost|total_items|is_order|is_return|
+---------------+--------------+-------------------+----------+-----------+--------+---------+
```



```
-------------------------------------------
Batch: 3
-------------------------------------------
+---------------+--------------+-------------------+----------+-----------+--------+---------+
|invoice_no     |country       |timestamp          |total_cost|total_items|is_order|is_return|
+---------------+--------------+-------------------+----------+-----------+--------+---------+
|154132557706025|United Kingdom|2023-12-22 14:24:05|88.38     |25         |1       |0        |
|154132557706026|United Kingdom|2023-12-22 14:24:15|9.9       |6          |1       |0        |
|154132557706027|France        |2023-12-22 14:24:24|77.51     |27         |1       |0        |
|154132557706028|France        |2023-12-22 14:24:25|18.12     |4          |1       |0        |
|154132557706029|United Kingdom|2023-12-22 14:24:33|49.27     |37         |1       |0        |
|154132557706030|United Kingdom|2023-12-22 14:24:42|3.35      |3          |1       |0        |
|154132557706031|United Kingdom|2023-12-22 14:24:53|7.46      |1          |1       |0        |
+---------------+--------------+-------------------+----------+-----------+--------+---------+

-------------------------------------------
Batch: 4
-------------------------------------------
+---------------+--------------+-------------------+----------+-----------+--------+---------+
|invoice_no     |country       |timestamp          |total_cost|total_items|is_order|is_return|
+---------------+--------------+-------------------+----------+-----------+--------+---------+
|154132557706032|United Kingdom|2023-12-22 14:25:14|252.01    |27         |1       |0        |
|154132557706033|United Kingdom|2023-12-22 14:25:18|32.37     |3          |1       |0        |
|154132557706034|United Kingdom|2023-12-22 14:25:20|19.26     |6          |1       |0        |
|154132557706035|United Kingdom|2023-12-22 14:25:21|49.350002 |25         |1       |0        |
|154132557706036|United Kingdom|2023-12-22 14:25:39|122.31    |12         |1       |0        |
|154132557706037|United Kingdom|2023-12-22 14:25:43|109.93    |53         |1       |0        |
|154132557706038|United Kingdom|2023-12-22 14:25:55|26.14     |26         |1       |0        |
|154132557706039|United Kingdom|2023-12-22 14:25:59|137.7     |30         |1       |0        |
+---------------+--------------+-------------------+----------+-----------+--------+---------+

-------------------------------------------
Batch: 5
-------------------------------------------
+---------------+--------------+-------------------+----------+-----------+--------+---------+
|invoice_no     |country       |timestamp          |total_cost|total_items|is_order|is_return|
+---------------+--------------+-------------------+----------+-----------+--------+---------+
|154132557706040|United Kingdom|2023-12-22 14:26:00|646.44    |228        |1       |0        |
|154132557706041|United Kingdom|2023-12-22 14:26:25|20.5      |10         |1       |0        |
|154132557706042|United Kingdom|2023-12-22 14:26:34|20.14     |12         |1       |0        |
|154132557706043|United Kingdom|2023-12-22 14:26:43|-21.36    |24         |0       |1        |
|154132557706044|France        |2023-12-22 14:26:50|12.059999 |6          |1       |0        |
|154132557706045|United Kingdom|2023-12-22 14:26:53|11.4      |12         |1       |0        |
+---------------+--------------+-------------------+----------+-----------+--------+---------+

Traceback (most recent call last):
  File "/home/hadoop/spark-retail.py", line 137, in <module>
    extendedOrderQuery.awaitTermination()
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/sql/streaming.py", line 103, in awaitTermination
  File "/usr/lib/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.py", line 1255, in __call__
  File "/usr/lib/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.py", line 985, in send_command
  File "/usr/lib/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.py", line 1152, in send_command
  File "/usr/lib64/python3.7/socket.py", line 589, in readinto
    return self._sock.recv_into(b)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/context.py", line 278, in signal_handler
```

- Checked HDFS to make sure the KPI files were present using below command:

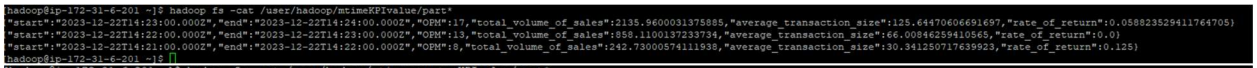
```
hadoop fs -ls /user/hadoop
```

- Checked the folders to see the JSON files using below commands:

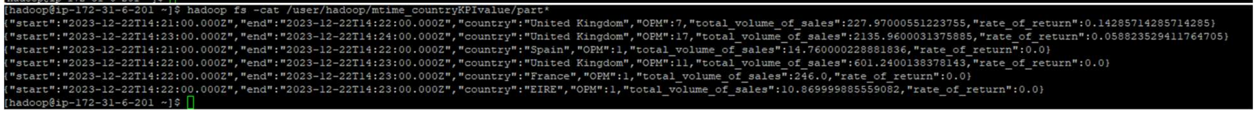
```
hadoop fs -ls /user/hadoop/timeKPIvalue
hadoop fs -ls /user/hadoop/time_countryKPIvalue
```

- And used ‘cat’ command to take a look at the data using below commands:

```
hadoop fs -cat /user/hadoop/timeKPIvalue/part*
hadoop fs -cat /user/hadoop/time_countryKPIvalue/part*
```



```
[hadoop@ip-172-31-6-201 ~]$ hadoop fs -cat /user/hadoop/mtimeKPIvalue/part*
{"start":"2023-12-22T14:23:00.000Z","end":"2023-12-22T14:24:00.000Z","OPM":17,"total_volume_of_sales":2135.9600031375885,"average_transaction_size":125.64470606691697,"rate_of_return":0.058823529411764705}
{"start":"2023-12-22T14:22:00.000Z","end":"2023-12-22T14:23:00.000Z","OPM":13,"total_volume_of_sales":858.1100137233734,"average_transaction_size":66.00846259410565,"rate_of_return":0.0}
{"start":"2023-12-22T14:21:00.000Z","end":"2023-12-22T14:22:00.000Z","OPM":8,"total_volume_of_sales":242.73000574111938,"average_transaction_size":30.341250717639923,"rate_of_return":0.125}
[hadoop@ip-172-31-6-201 ~]$
```



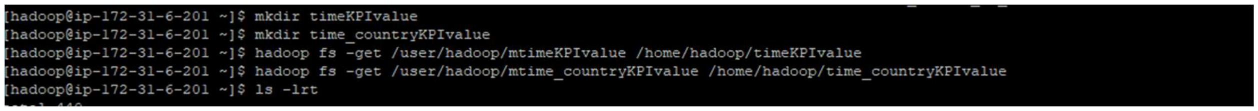
```
[hadoop@ip-172-31-6-201 ~]$ hadoop fs -cat /user/hadoop/mtime_countryKPIvalue/part*
{"start":"2023-12-22T14:21:00.000Z","end":"2023-12-22T14:22:00.000Z","country":"United Kingdom","OPM":7,"total_volume_of_sales":227.97000551223755,"rate_of_return":0.14285714285714285}
{"start":"2023-12-22T14:23:00.000Z","end":"2023-12-22T14:24:00.000Z","country":"United Kingdom","OPM":17,"total_volume_of_sales":2135.9600031375885,"rate_of_return":0.058823529411764705}
{"start":"2023-12-22T14:21:00.000Z","end":"2023-12-22T14:22:00.000Z","country":"Spain","OPM":1,"total_volume_of_sales":14.760000228881836,"rate_of_return":0.0}
{"start":"2023-12-22T14:22:00.000Z","end":"2023-12-22T14:23:00.000Z","country":"United Kingdom","OPM":11,"total_volume_of_sales":601.2400138378143,"rate_of_return":0.0}
{"start":"2023-12-22T14:22:00.000Z","end":"2023-12-22T14:23:00.000Z","country":"France","OPM":1,"total_volume_of_sales":246.0,"rate_of_return":0.0}
{"start":"2023-12-22T14:22:00.000Z","end":"2023-12-22T14:23:00.000Z","country":"EIRE","OPM":1,"total_volume_of_sales":10.869999885559082,"rate_of_return":0.0}
[hadoop@ip-172-31-6-201 ~]$
```

First, I needed to transfer the JSON files from HDFS into the EC2 system

I created directories for time-based and then time-and-country-based KPIs as ec2-user. Usingthe ‘get’ command I copied the contents of the output folders into the EC2 system.

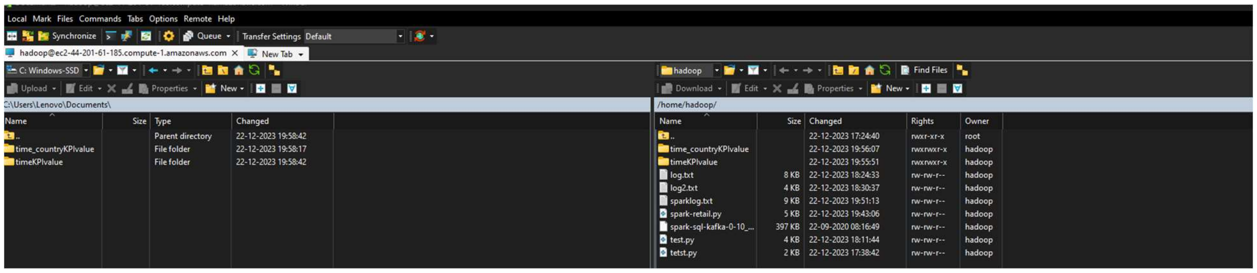
```
mkdir timeKPIvalue
hadoop fs -get /user/hadoop/timeKPIvalue /home/hadoop/timeKPIvalue


mkdir time_countryKPIvalue
hadoop fs -get /user/hadoop/time_countryKPIvalue /home/hadoop/time_countryKPIvalue
```



```
[hadoop@ip-172-31-6-201 ~]$ mkdir timeKPIvalue
[hadoop@ip-172-31-6-201 ~]$ mkdir time_countryKPIvalue
[hadoop@ip-172-31-6-201 ~]$ hadoop fs -get /user/hadoop/mtimeKPIvalue /home/hadoop/timeKPIvalue
[hadoop@ip-172-31-6-201 ~]$ hadoop fs -get /user/hadoop/mtime_countryKPIvalue /home/hadoop/time_countryKPIvalue
[hadoop@ip-172-31-6-201 ~]$ ls -lrt
```

Thereafter I used WinSCP to establish a connection between the EC2 instance and my local file system to transfer all the required files into my system.


Local Mark Files Commands Tabs Options Remote Help
Synchronize
Queue
Transfer Settings Default
hadoop@ec2-44-201-61-185.compute-1.amazonaws.com
New Tab
C: Windows-SSD
Upload
Edit
Properties
New
hadoop
Find Files
Download

C:\Users\Lenovo\Documents\

Name	Size	Type	Changed
..		Parent directory	22-12-2023 19:58:42
time_countryKPIvalue		File folder	22-12-2023 19:58:17
timeKPIvalue		File folder	22-12-2023 19:58:42

/home/hadoop/

Name	Size	Changed	Rights	Owner
..		22-12-2023 17:24:40	rwxr-xr-x	root
time_countryKPIvalue		22-12-2023 19:56:07	rwxrwxr-x	hadoop
timeKPIvalue		22-12-2023 19:55:51	rwxrwxr-x	hadoop
log.txt	8 KB	22-12-2023 18:24:33	rw-rw-r--	hadoop
log2.txt	4 KB	22-12-2023 18:30:37	rw-rw-r--	hadoop
sparklog.txt	9 KB	22-12-2023 19:51:13	rw-rw-r--	hadoop
spark-retail.py	5 KB	22-12-2023 19:43:06	rw-rw-r--	hadoop
spark-sql-kafka-0-10_...	397 KB	22-09-2020 08:16:49	rw-rw-r--	hadoop
test.py	4 KB	22-12-2023 18:11:44	rw-rw-r--	hadoop
tetst.py	2 KB	22-12-2023 17:38:42	rw-rw-r--	hadoop