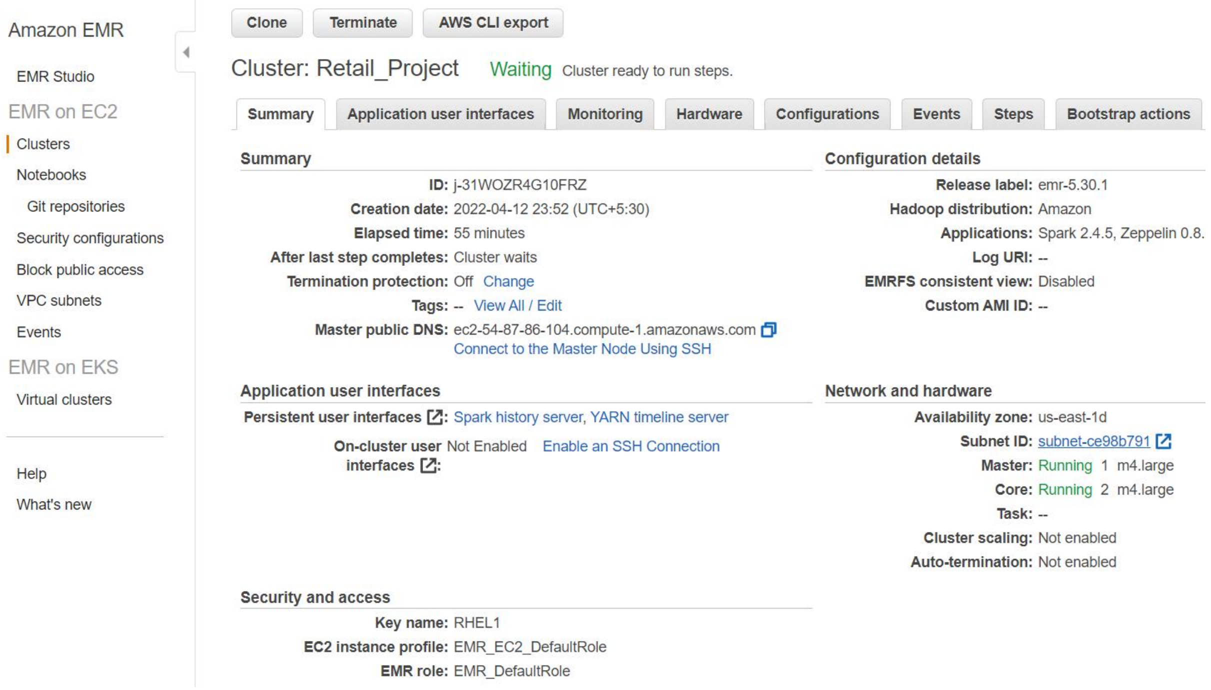
Code Logic - Retail Data Analysis

This document contains description of the code and the overall steps taken to solve the problem statement.

- **In order to proceed with solution created EMR cluster with basic spark setup**



Cluster: Retail_Project Waiting Cluster ready to run steps.

Summary

ID:	j-31WOZR4G10FRZ
Creation date:	2022-04-12 23:52 (UTC+5:30)
Elapsed time:	55 minutes
After last step completes:	Cluster waits
Termination protection:	Off Change
Tags:	-- View All / Edit
Master public DNS:	ec2-54-87-86-104.compute-1.amazonaws.com Connect to the Master Node Using SSH

Configuration details

Release label:	emr-5.30.1
Hadoop distribution:	Amazon
Applications:	Spark 2.4.5, Zeppelin 0.8.
Log URI:	--
EMRFS consistent view:	Disabled
Custom AMI ID:	--

Application user interfaces

Persistent user interfaces:	Spark history server, YARN timeline server
On-cluster user interfaces:	Not Enabled Enable an SSH Connection

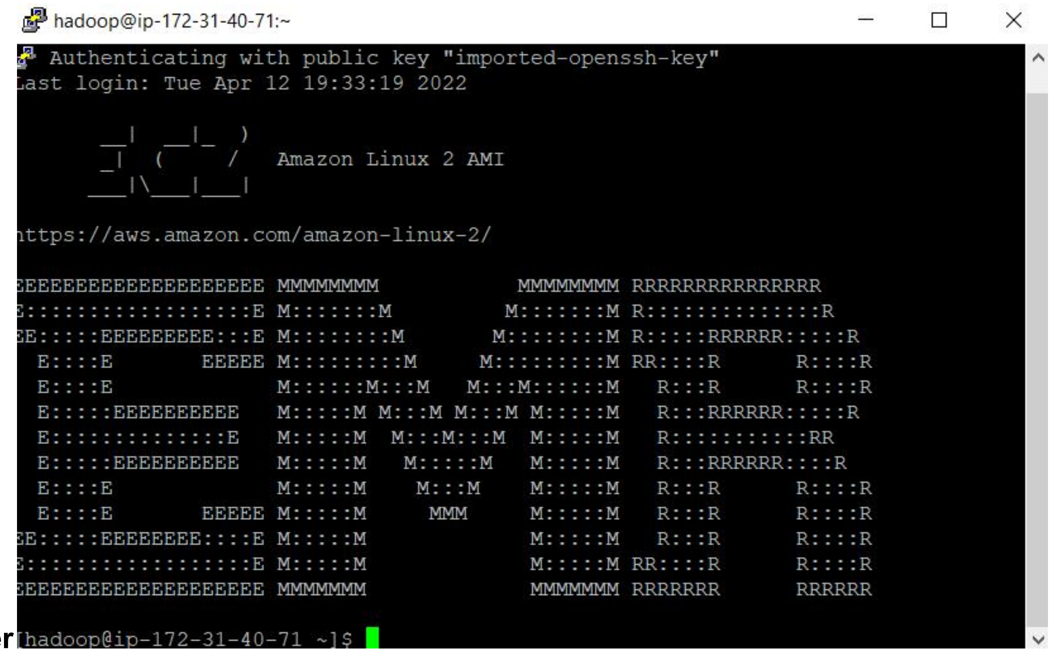
Network and hardware

Availability zone:	us-east-1d
Subnet ID:	subnet-ce98b791
Master:	Running 1 m4.large
Core:	Running 2 m4.large
Task:	--
Cluster scaling:	Not enabled
Auto-termination:	Not enabled

Security and access

Key name:	RHEL1
EC2 instance profile:	EMR_EC2_DefaultRole
EMR role:	EMR_DefaultRole

- **Logged into EMR instance as "hadoop" user**



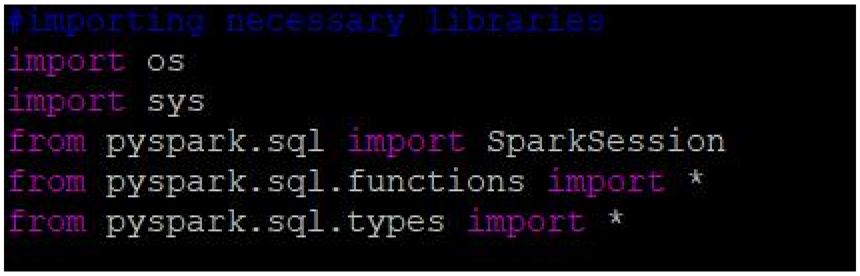
```
 Authenticating with public key "imported-openssh-key"
Last login: Tue Apr 12 19:33:19 2022

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/

EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-40-71 ~]$
```

- Switch to root user and run pip install kafka-python and then again use “sudo -i -u hadoop” to be a hadoop user

- Next, I created the ‘spark-streaming.py’ file having the following code
vi spark-streaming.py

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

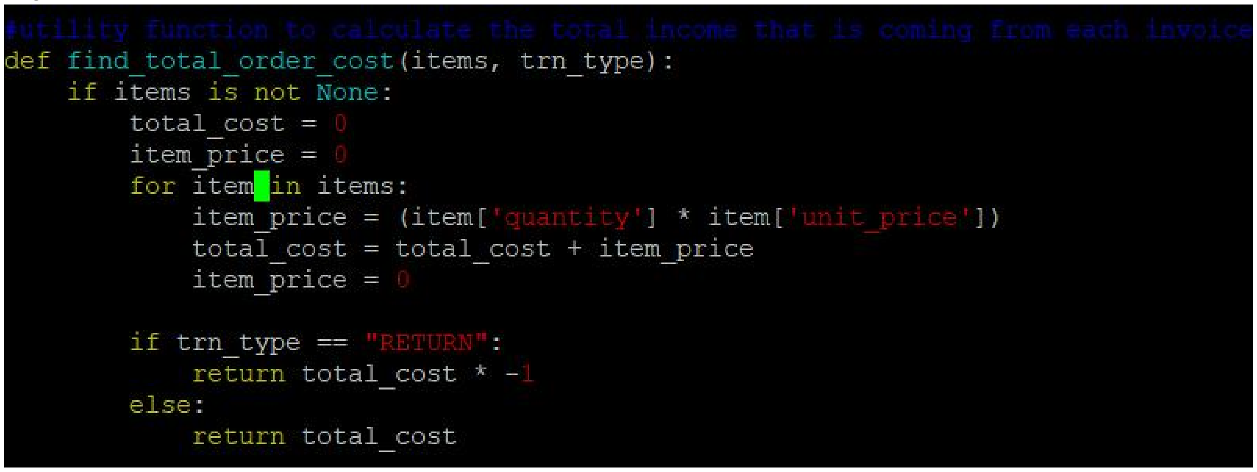
Setting up the system dependencies and importing necessary libraries and modules



```
#importing necessary libraries
import os
import sys
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
```

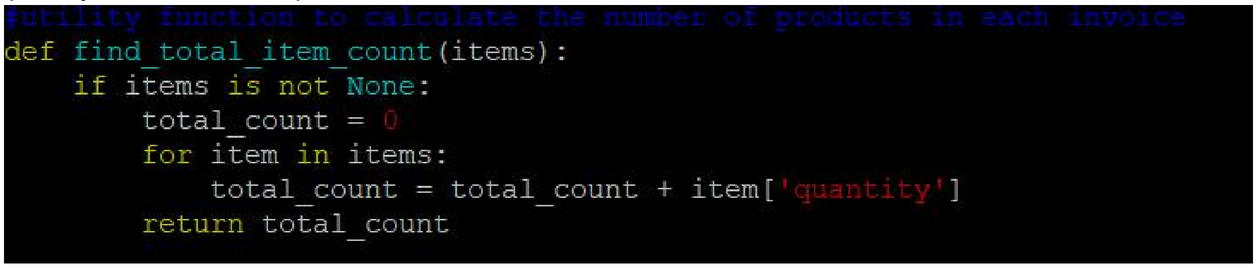
Python functions, containing the logic for the UDFs

1. Total Cost UDF - To calculate the total income from each invoice which is required to calculate the income from sale of each product, so multiplied the unit price of the product with the quantity of the product purchased. The sum of operation gives me the total cost of the order. I also made sure that if the transaction is a return transaction, then the total cost is negative.



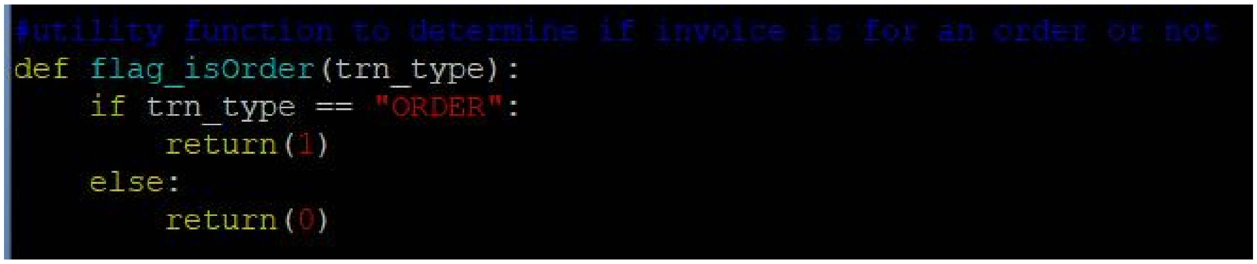
```
#utility function to calculate the total income that is coming from each invoice
def find_total_order_cost(items, trn_type):
    if items is not None:
        total_cost = 0
        item_price = 0
        for item in items:
            item_price = (item['quantity'] * item['unit_price'])
            total_cost = total_cost + item_price
            item_price = 0

        if trn_type == "RETURN":
            return total_cost * -1
        else:
            return total_cost
```

2. Total Items UDF - This calculate the number of products in each invoice by adding the quantity ordered of each product in that invoice



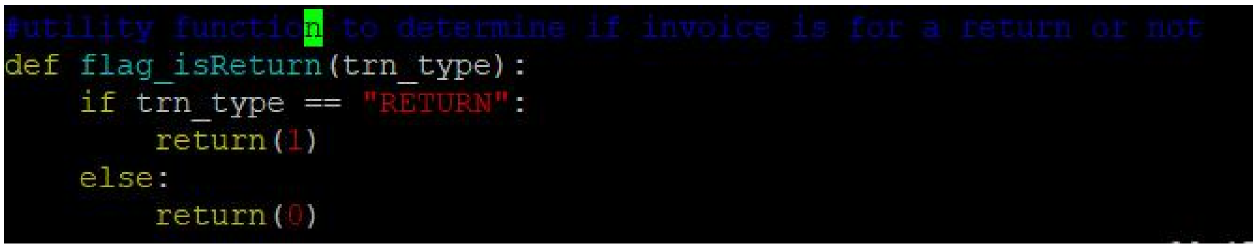
```
#utility function to calculate the number of products in each invoice
def find_total_item_count(items):
    if items is not None:
        total_count = 0
        for item in items:
            total_count = total_count + item['quantity']
        return total_count
```

3. Is Order UDF – This is to determine if invoice is for an order or not with help of an if-else statement



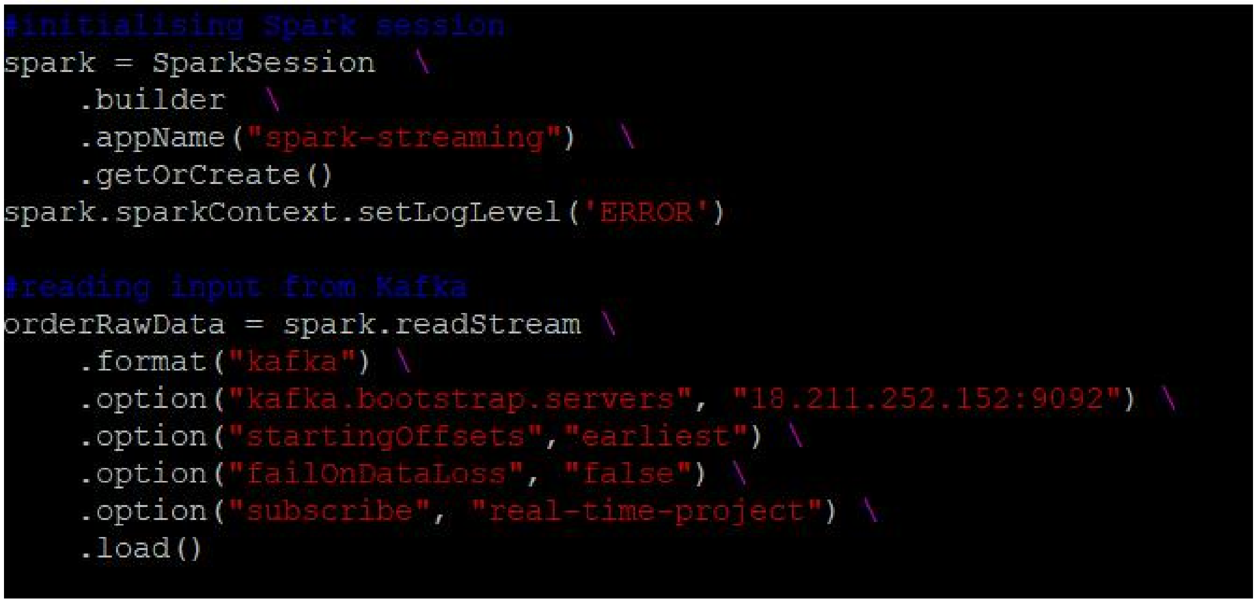
```
#utility function to determine if invoice is for an order or not
def flag_isOrder(trn_type):
    if trn_type == "ORDER":
        return(1)
    else:
        return(0)
```

4. Is Return UDF – This is to determine if invoice is for a return or not with help of an if-else statement



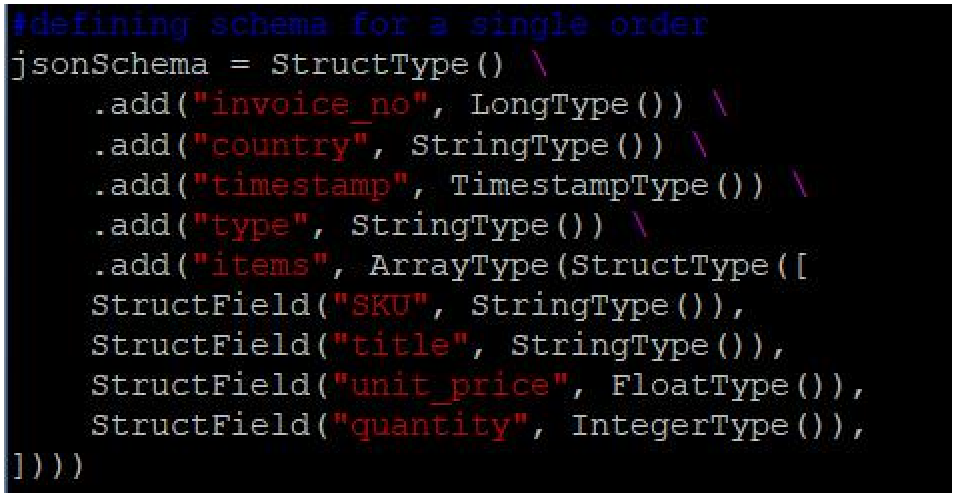
```
#utility function to determine if invoice is for a return or not
def flag_isReturn(trn_type):
    if trn_type == "RETURN":
        return(1)
    else:
        return(0)
```

- Initializing the Spark session and reading input data from Kafka mentioning the details of the Kafka broker, such as bootstrap server, port and topic name



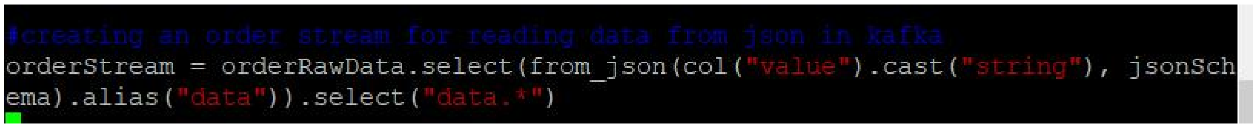
```
#initialising Spark session
spark = SparkSession  \
     .builder  \
     .appName("spark-streaming")  \
     .getOrCreate()
spark.sparkContext.setLogLevel('ERROR')

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

- Define JSON schema of each order



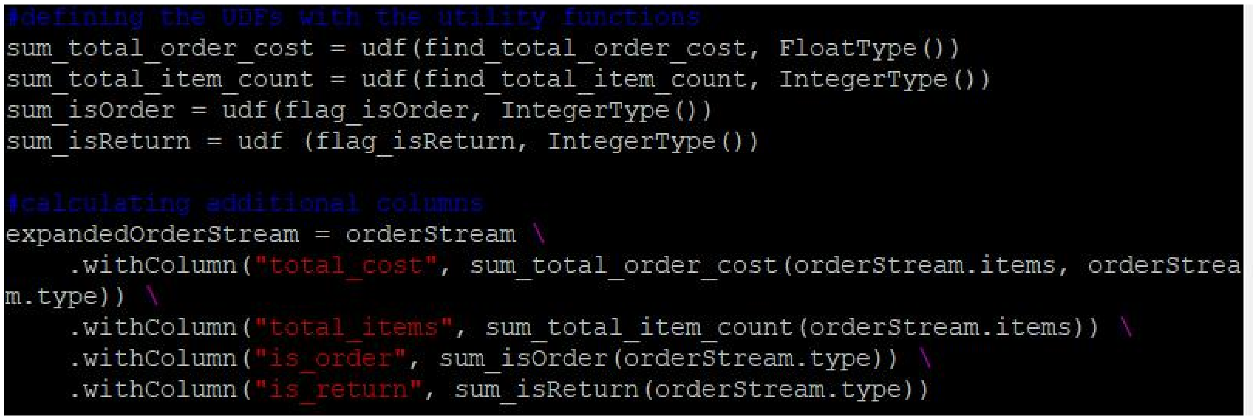
```
#defining schema for a single order
jsonSchema = StructType() \
    .add("invoice_no", LongType()) \
    .add("country", StringType()) \
    .add("timestamp", TimestampType()) \
    .add("type", StringType()) \
    .add("items", ArrayType(StructType([
    StructField("SKU", StringType()),
    StructField("title", StringType()),
    StructField("unit_price", FloatType()),
    StructField("quantity", IntegerType()),
])))
```

- Read the raw JSON data from Kafka as ‘order stream’



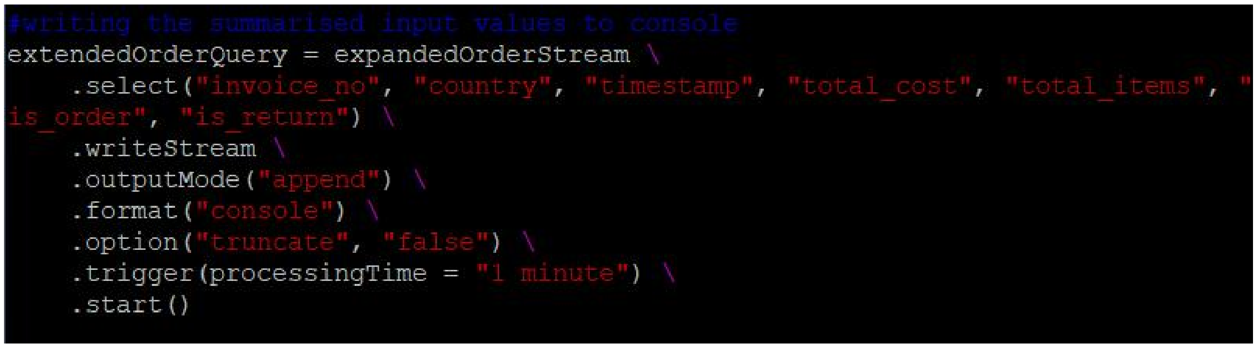
```
#creating an order stream for reading data from json in Kafka
orderStream = orderRawData.select(from_json(col("value").cast("string"), jsonSch
ema).alias("data")).select("data.*")
```

- Defining the UDFs by Converting the Python functions defined earlier, and Calculating the additional columns according to the required input values



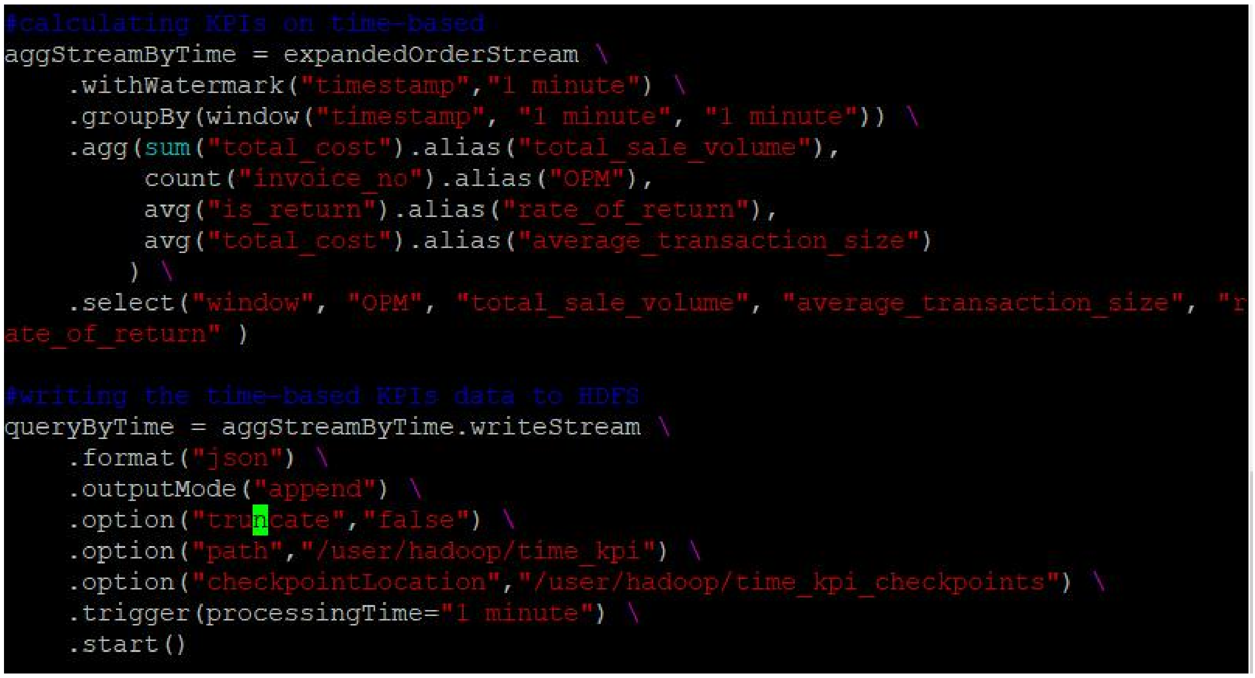
```
#defining the UDFs with the utility functions
sum_total_order_cost = udf(find_total_order_cost, FloatType())
sum_total_item_count = udf(find_total_item_count, IntegerType())
sum_isOrder = udf(flag_isOrder, IntegerType())
sum_isReturn = udf (flag_isReturn, IntegerType())

#calculating additional columns
expandedOrderStream = orderStream \
    .withColumn("total_cost", sum_total_order_cost(orderStream.items, orderStrea
m.type)) \
    .withColumn("total_items", sum_total_item_count(orderStream.items)) \
    .withColumn("is_order", sum_isOrder(orderStream.type)) \
    .withColumn("is_return", sum_isReturn(orderStream.type))
```

- Write the summarized input values to console, use ‘append’ output method , Set truncate as ‘false’ and processing time as ‘1 minute’



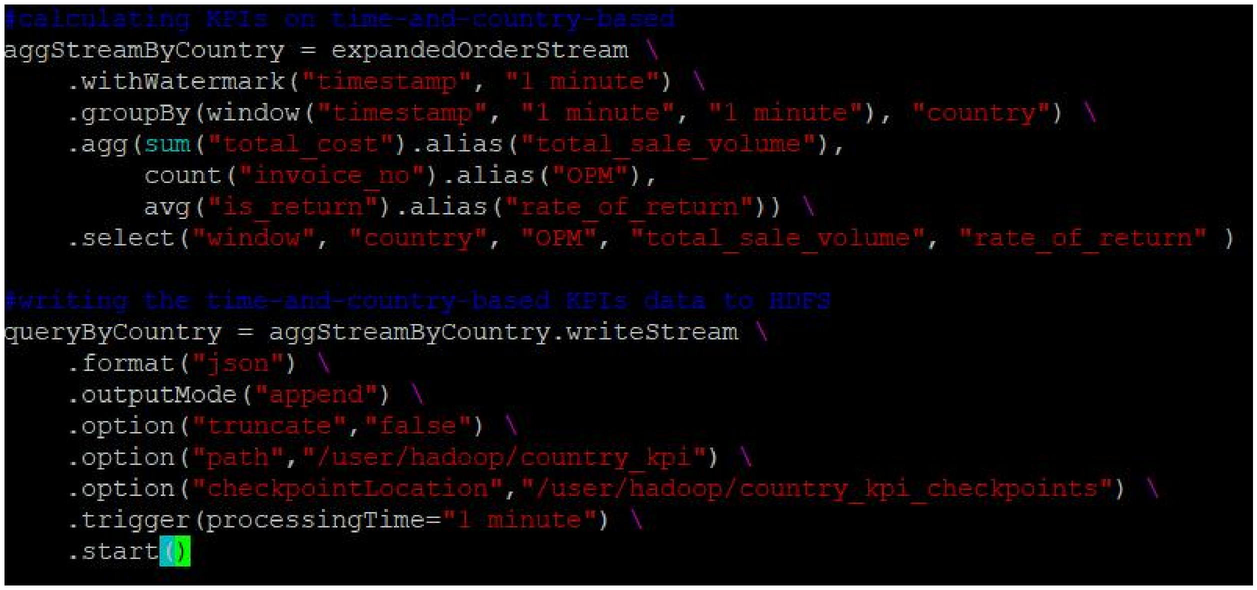
```
#writing the summarised input values to console
extendedOrderQuery = expandedOrderStream \
    .select("invoice_no", "country", "timestamp", "total_cost", "total_items", "
is_order", "is_return") \
    .writeStream \
    .outputMode("append") \
    .format("console") \
    .option("truncate", "false")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

- Calculate time-based KPIs (Total sale volume, OPM, Rate of return, Average transaction size) set tumbling window and watermark as one minute and Writing the time-based KPIs data to HDFS as JSON files



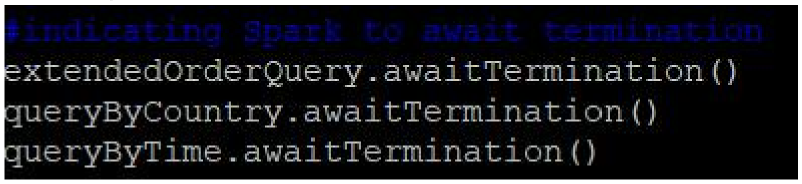
```
#calculating KPIs on time-based
aggStreamByTime = expandedOrderStream \
    .withWatermark("timestamp","1 minute") \
    .groupBy(window("timestamp", "1 minute", "1 minute")) \
    .agg(sum("total_cost").alias("total_sale_volume"),
         count("invoice_no").alias("OPM"),
         avg("is_return").alias("rate_of_return"),
         avg("total_cost").alias("average_transaction_size")
        ) \
    .select("window", "OPM", "total_sale_volume", "average_transaction_size", "r
ate_of_return" )

#writing the time-based KPIs data to HDFS
queryByTime = aggStreamByTime.writeStream \
    .format("json") \
    .outputMode("append") \
    .option("truncate","false") \
    .option("path","/user/hadoop/time_kpi") \
    .option("checkpointLocation","/user/hadoop/time_kpi_checkpoints") \
    .trigger(processingTime="1 minute") \
    .start()
```

- Calculate time-and-country-based KPIs (Total sale volume, OPM, Rate of return) having tumbling window and watermark of one minute. Use grouped by on both window and country and Writing the time-and-country-based KPIs data to HDFS as JSON files



```
#calculating KPIs on time-and-country-based
aggStreamByCountry = expandedOrderStream \
    .withWatermark("timestamp", "1 minute") \
    .groupBy(window("timestamp", "1 minute", "1 minute"), "country") \
    .agg(sum("total_cost").alias("total_sale_volume"),
         count("invoice_no").alias("OPM"),
         avg("is_return").alias("rate_of_return")) \
    .select("window", "country", "OPM", "total_sale_volume", "rate_of_return" )

#writing the time-and-country-based KPIs data to HDFS
queryByCountry = aggStreamByCountry.writeStream \
    .format("json") \
    .outputMode("append") \
    .option("truncate","false") \
    .option("path","/user/hadoop/country_kpi") \
    .option("checkpointLocation","/user/hadoop/country_kpi_checkpoints") \
    .trigger(processingTime="1 minute") \
    .start()
```

- Define spark termination

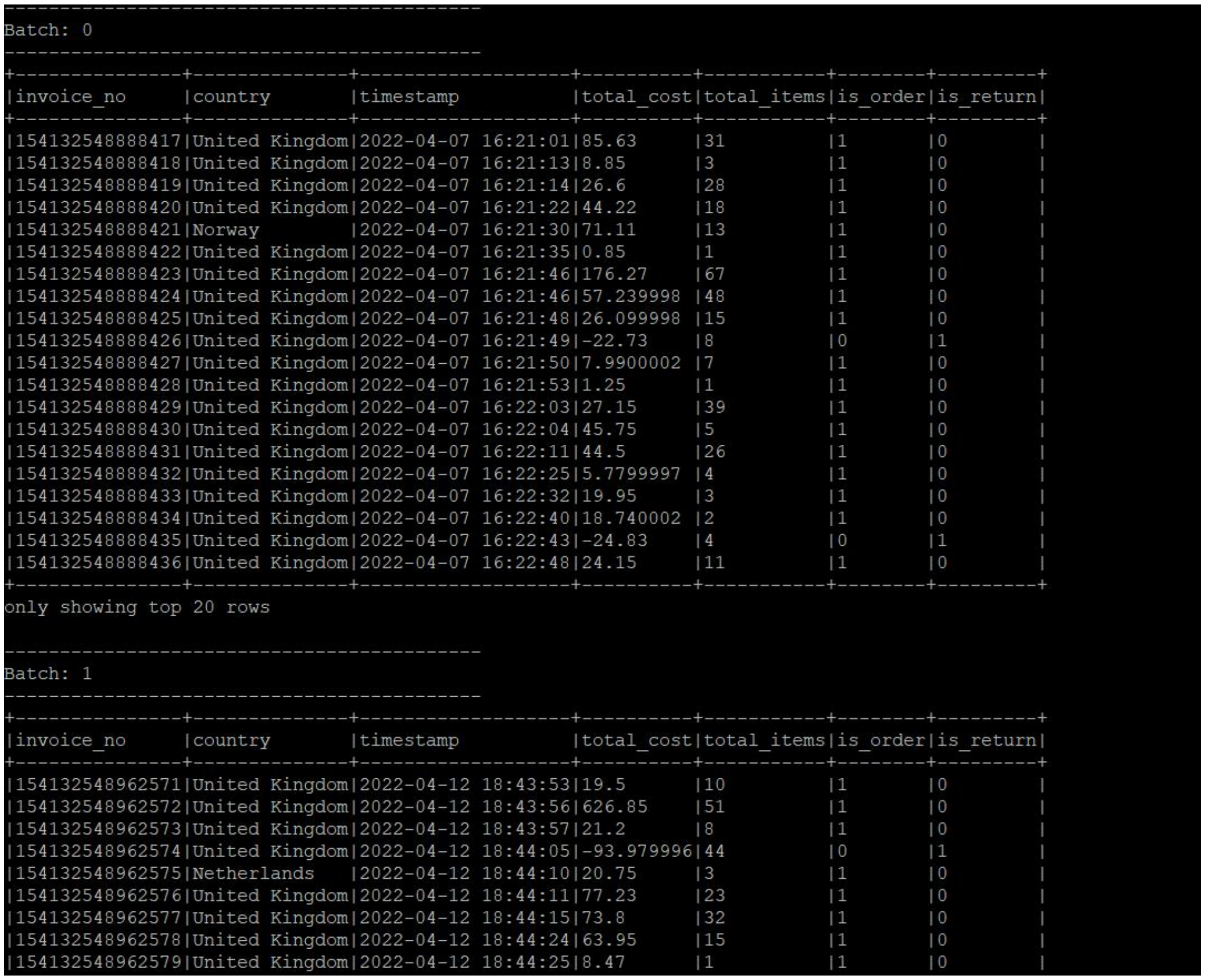


```
#indicating Spark to await termination
extendedOrderQuery.awaitTermination()
queryByCountry.awaitTermination()
queryByTime.awaitTermination()
```

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

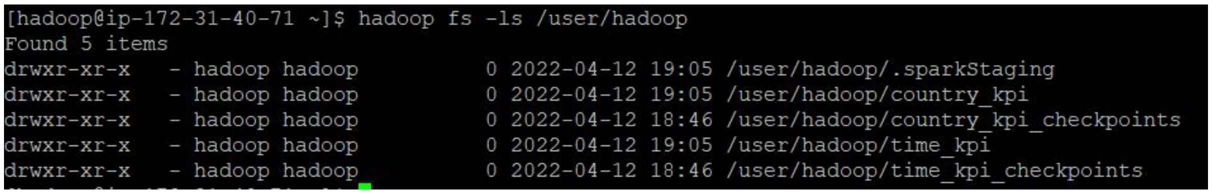
- Run the spark-submit command, specifying the Spark-SQL-Kafka package and python file
  spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 spark-streaming.py

- Streaming output in console:



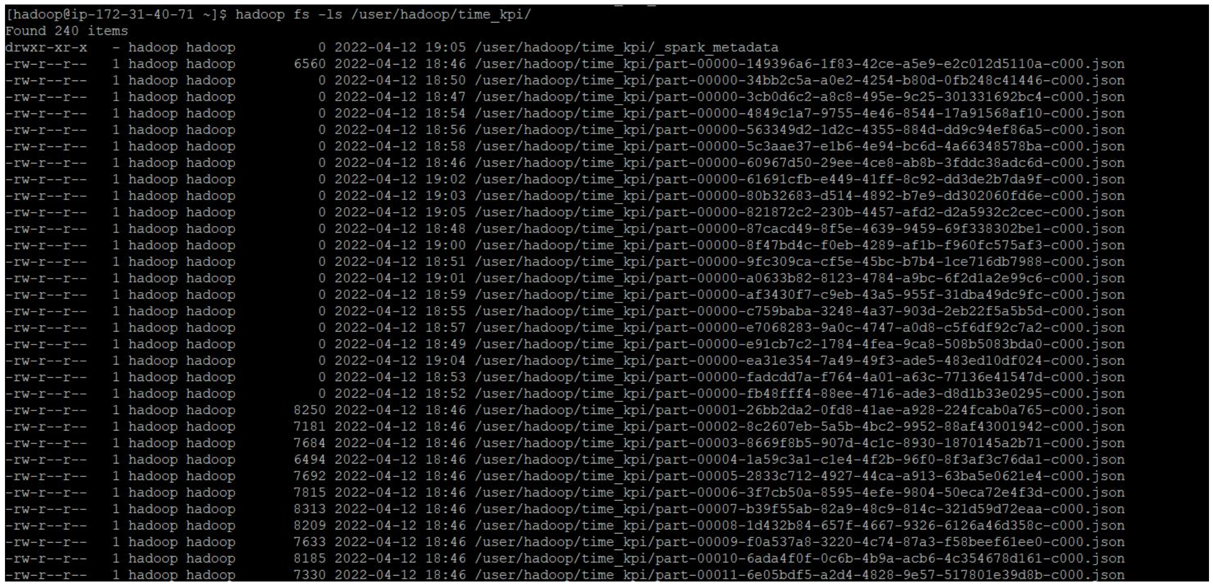
```
-------------------------------------------
Batch: 0
-------------------------------------------
+---------------+--------------+-------------------+----------+-----------+--------+---------+
|invoice_no     |country       |timestamp          |total_cost|total_items|is_order|is_return|
+---------------+--------------+-------------------+----------+-----------+--------+---------+
|154132548888417|United Kingdom|2022-04-07 16:21:01|85.63     |31         |1       |0        |
|154132548888418|United Kingdom|2022-04-07 16:21:13|8.85      |3          |1       |0        |
|154132548888419|United Kingdom|2022-04-07 16:21:14|26.6      |28         |1       |0        |
|154132548888420|United Kingdom|2022-04-07 16:21:22|44.22     |18         |1       |0        |
|154132548888421|Norway        |2022-04-07 16:21:30|71.11     |13         |1       |0        |
|154132548888422|United Kingdom|2022-04-07 16:21:35|0.85      |1          |1       |0        |
|154132548888423|United Kingdom|2022-04-07 16:21:46|176.27    |67         |1       |0        |
|154132548888424|United Kingdom|2022-04-07 16:21:46|57.239998 |48         |1       |0        |
|154132548888425|United Kingdom|2022-04-07 16:21:48|26.099998 |15         |1       |0        |
|154132548888426|United Kingdom|2022-04-07 16:21:49|-22.73    |8          |0       |1        |
|154132548888427|United Kingdom|2022-04-07 16:21:50|7.9900002 |7          |1       |0        |
|154132548888428|United Kingdom|2022-04-07 16:21:53|1.25      |1          |1       |0        |
|154132548888429|United Kingdom|2022-04-07 16:22:03|27.15     |39         |1       |0        |
|154132548888430|United Kingdom|2022-04-07 16:22:04|45.75     |5          |1       |0        |
|154132548888431|United Kingdom|2022-04-07 16:22:11|44.5      |26         |1       |0        |
|154132548888432|United Kingdom|2022-04-07 16:22:25|5.7799997 |4          |1       |0        |
|154132548888433|United Kingdom|2022-04-07 16:22:32|19.95     |3          |1       |0        |
|154132548888434|United Kingdom|2022-04-07 16:22:40|18.740002 |2          |1       |0        |
|154132548888435|United Kingdom|2022-04-07 16:22:43|-24.83    |4          |0       |1        |
|154132548888436|United Kingdom|2022-04-07 16:22:48|24.15     |11         |1       |0        |
+---------------+--------------+-------------------+----------+-----------+--------+---------+
only showing top 20 rows

-------------------------------------------
Batch: 1
-------------------------------------------
+---------------+--------------+-------------------+----------+-----------+--------+---------+
|invoice_no     |country       |timestamp          |total_cost|total_items|is_order|is_return|
+---------------+--------------+-------------------+----------+-----------+--------+---------+
|154132548962571|United Kingdom|2022-04-12 18:43:53|19.5      |10         |1       |0        |
|154132548962572|United Kingdom|2022-04-12 18:43:56|626.85    |51         |1       |0        |
|154132548962573|United Kingdom|2022-04-12 18:43:57|21.2      |8          |1       |0        |
|154132548962574|United Kingdom|2022-04-12 18:44:05|-93.979996|44         |0       |1        |
|154132548962575|Netherlands   |2022-04-12 18:44:10|20.75     |3          |1       |0        |
|154132548962576|United Kingdom|2022-04-12 18:44:11|77.23     |23         |1       |0        |
|154132548962577|United Kingdom|2022-04-12 18:44:15|73.8      |32         |1       |0        |
|154132548962578|United Kingdom|2022-04-12 18:44:24|63.95     |15         |1       |0        |
|154132548962579|United Kingdom|2022-04-12 18:44:25|8.47      |1          |1       |0        |
```

- Check HDFS to make sure the KPI files are present
  hadoop fs -ls /user/hadoop



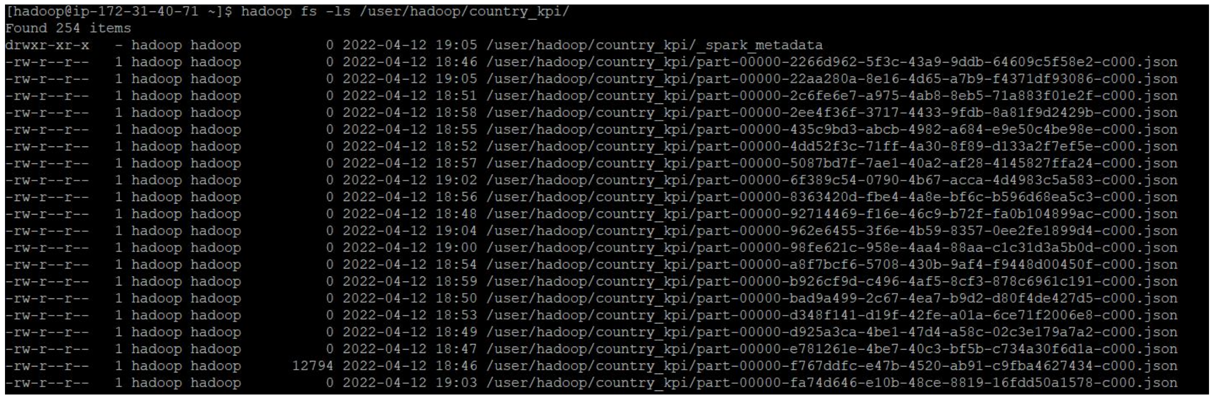
```
[hadoop@ip-172-31-40-71 ~]$ hadoop fs -ls /user/hadoop
Found 5 items
drwxr-xr-x   - hadoop hadoop          0 2022-04-12 19:05 /user/hadoop/.sparkStaging
drwxr-xr-x   - hadoop hadoop          0 2022-04-12 19:05 /user/hadoop/country_kpi
drwxr-xr-x   - hadoop hadoop          0 2022-04-12 18:46 /user/hadoop/country_kpi_checkpoints
drwxr-xr-x   - hadoop hadoop          0 2022-04-12 19:05 /user/hadoop/time_kpi
drwxr-xr-x   - hadoop hadoop          0 2022-04-12 18:46 /user/hadoop/time_kpi_checkpoints
```

- Check the specified folders to see the JSON files
  hadoop fs -ls /user/hadoop/time_kpi/



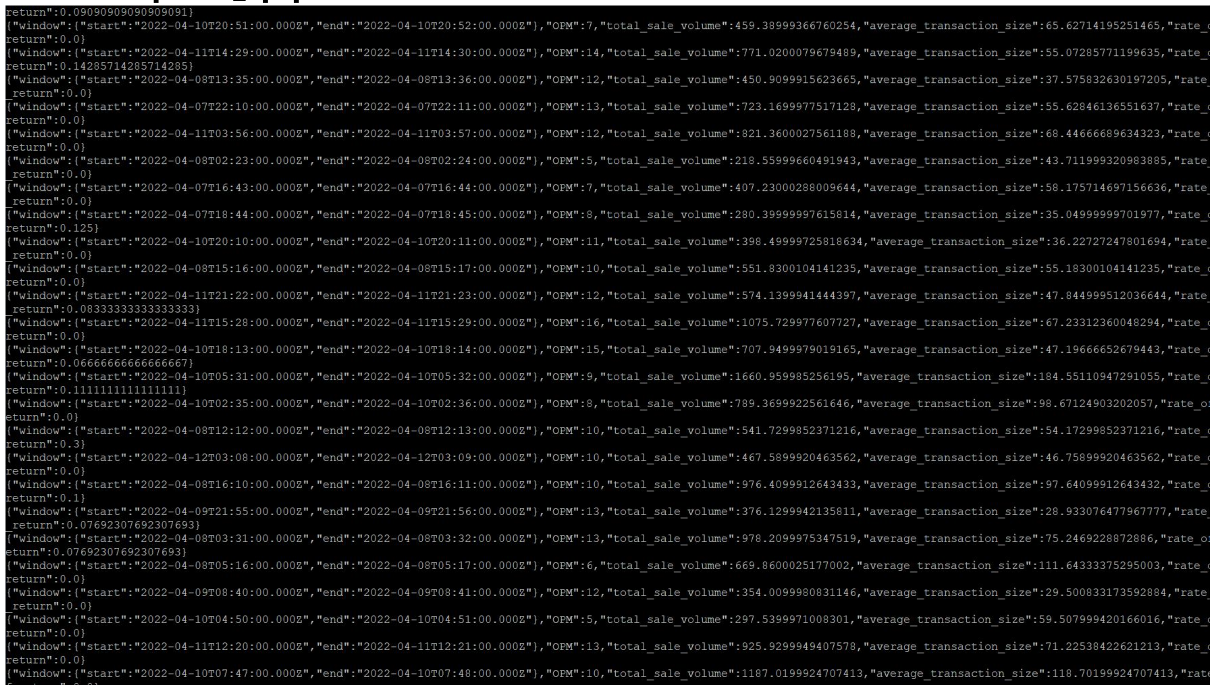
```
[hadoop@ip-172-31-40-71 ~]$ hadoop fs -ls /user/hadoop/time_kpi/
Found 240 items
drwxr-xr-x   - hadoop hadoop          0 2022-04-12 19:05 /user/hadoop/time_kpi/_spark_metadata
-rw-r--r--   1 hadoop hadoop       6560 2022-04-12 18:46 /user/hadoop/time_kpi/part-00000-149396a6-1f83-42ce-a5e9-e2c012d5110a-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:50 /user/hadoop/time_kpi/part-00000-34bb2c5a-a0e2-4254-b80d-0fb248c41446-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:47 /user/hadoop/time_kpi/part-00000-3cb0d6c2-a8c8-495e-9c25-301331692bc4-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:54 /user/hadoop/time_kpi/part-00000-4849c1a7-9755-4e46-8544-17a91568af10-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:56 /user/hadoop/time_kpi/part-00000-563349d2-1d2c-4355-884d-dd9c94ef86a5-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:58 /user/hadoop/time_kpi/part-00000-5c3aae37-e1b6-4e94-bc6d-4a66348578ba-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:46 /user/hadoop/time_kpi/part-00000-60967d50-29ee-4ce8-ab8b-3fddc38adc6d-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 19:02 /user/hadoop/time_kpi/part-00000-61691cfb-e449-41ff-8c92-dd3de2b7da9f-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 19:03 /user/hadoop/time_kpi/part-00000-80b32683-d514-4892-b7e9-dd302060fd6e-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 19:05 /user/hadoop/time_kpi/part-00000-821872c2-230b-4457-afd2-d2a5932c2cec-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:48 /user/hadoop/time_kpi/part-00000-87cacd49-8f5e-4639-9459-69f338302be1-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 19:00 /user/hadoop/time_kpi/part-00000-8f47bd4c-f0eb-4289-af1b-f960fc575af3-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:51 /user/hadoop/time_kpi/part-00000-9fc309ca-cf5e-45bc-b7b4-1ce716db7988-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 19:01 /user/hadoop/time_kpi/part-00000-a0633b82-8123-4784-a9bc-6f2d1a2e99c6-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:59 /user/hadoop/time_kpi/part-00000-af3430f7-c9eb-43a5-955f-31dba49dc9fc-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:55 /user/hadoop/time_kpi/part-00000-c759baba-3248-4a37-903d-2eb22f5a5b5d-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:57 /user/hadoop/time_kpi/part-00000-e7068283-9a0c-4747-a0d8-c5f6df92c7a2-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:49 /user/hadoop/time_kpi/part-00000-e91cb7c2-1784-4fea-9ca8-508b5083bda0-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 19:04 /user/hadoop/time_kpi/part-00000-ea31e354-7a49-49f3-ade5-483ed10df024-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:53 /user/hadoop/time_kpi/part-00000-fadcdd7a-f764-4a01-a63c-77136e41547d-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:52 /user/hadoop/time_kpi/part-00000-fb48fff4-88ee-4716-ade3-d8d1b33e0295-c000.json
-rw-r--r--   1 hadoop hadoop       8250 2022-04-12 18:46 /user/hadoop/time_kpi/part-00001-26bb2da2-0fd8-41ae-a928-224fcab0a765-c000.json
-rw-r--r--   1 hadoop hadoop       7181 2022-04-12 18:46 /user/hadoop/time_kpi/part-00002-8c2607eb-5a5b-4bc2-9952-88af43001942-c000.json
-rw-r--r--   1 hadoop hadoop       7684 2022-04-12 18:46 /user/hadoop/time_kpi/part-00003-8669f8b5-907d-4c1c-8930-1870145a2b71-c000.json
-rw-r--r--   1 hadoop hadoop       6494 2022-04-12 18:46 /user/hadoop/time_kpi/part-00004-1a59c3a1-c1e4-4f2b-96f0-8f3af3c76da1-c000.json
-rw-r--r--   1 hadoop hadoop       7692 2022-04-12 18:46 /user/hadoop/time_kpi/part-00005-2833c712-4927-44ca-a913-63ba5e0621e4-c000.json
-rw-r--r--   1 hadoop hadoop       7815 2022-04-12 18:46 /user/hadoop/time_kpi/part-00006-3f7cb50a-8595-4efe-9804-50eca72e4f3d-c000.json
-rw-r--r--   1 hadoop hadoop       8313 2022-04-12 18:46 /user/hadoop/time_kpi/part-00007-b39f55ab-82a9-48c9-814c-321d59d72eaa-c000.json
-rw-r--r--   1 hadoop hadoop       8209 2022-04-12 18:46 /user/hadoop/time_kpi/part-00008-1d432b84-657f-4667-9326-6126a46d358c-c000.json
-rw-r--r--   1 hadoop hadoop       7633 2022-04-12 18:46 /user/hadoop/time_kpi/part-00009-f0a537a8-3220-4c74-87a3-f58beef61ee0-c000.json
-rw-r--r--   1 hadoop hadoop       8185 2022-04-12 18:46 /user/hadoop/time_kpi/part-00010-6ada4f0f-0c6b-4b9a-acb6-4c354678d161-c000.json
-rw-r--r--   1 hadoop hadoop       7330 2022-04-12 18:46 /user/hadoop/time_kpi/part-00011-6e05bdf5-a2d4-4828-9e57-517801e39d8b-c000.json
```

hadoop fs -ls /user/hadoop/country_kpi/



```
[hadoop@ip-172-31-40-71 ~]$ hadoop fs -ls /user/hadoop/country_kpi/
Found 254 items
drwxr-xr-x   - hadoop hadoop          0 2022-04-12 19:05 /user/hadoop/country_kpi/_spark_metadata
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:46 /user/hadoop/country_kpi/part-00000-2266d962-5f3c-43a9-9ddb-64609c5f58e2-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 19:05 /user/hadoop/country_kpi/part-00000-22aa280a-8e16-4d65-a7b9-f4371df93086-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:51 /user/hadoop/country_kpi/part-00000-2c6fe6e7-a975-4ab8-8eb5-71a883f01e2f-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:58 /user/hadoop/country_kpi/part-00000-2ee4f36f-3717-4433-9fdb-8a81f9d2429b-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:55 /user/hadoop/country_kpi/part-00000-435c9bd3-abcb-4982-a684-e9e50c4be98e-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:52 /user/hadoop/country_kpi/part-00000-4dd52f3c-71ff-4a30-8f89-d133a2f7ef5e-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:57 /user/hadoop/country_kpi/part-00000-5087bd7f-7ae1-40a2-af28-4145827ffa24-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 19:02 /user/hadoop/country_kpi/part-00000-6f389c54-0790-4b67-acca-4d4983c5a583-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:56 /user/hadoop/country_kpi/part-00000-8363420d-fbe4-4a8e-bf6c-b596d68ea5c3-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:48 /user/hadoop/country_kpi/part-00000-92714469-f16e-46c9-b72f-fa0b104899ac-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 19:04 /user/hadoop/country_kpi/part-00000-962e6455-3f6e-4b59-8357-0ee2fe1899d4-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 19:00 /user/hadoop/country_kpi/part-00000-98fe621c-958e-4aa4-88aa-c1c31d3a5b0d-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:54 /user/hadoop/country_kpi/part-00000-a8f7bcf6-5708-430b-9af4-f9448d00450f-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:59 /user/hadoop/country_kpi/part-00000-b926cf9d-c496-4af5-8cf3-878c6961c191-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:50 /user/hadoop/country_kpi/part-00000-bad9a499-2c67-4ea7-b9d2-d80f4de427d5-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:53 /user/hadoop/country_kpi/part-00000-d348f141-d19f-42fe-a01a-6ce71f2006e8-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:49 /user/hadoop/country_kpi/part-00000-d925a3ca-4be1-47d4-a58c-02c3e179a7a2-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 18:47 /user/hadoop/country_kpi/part-00000-e781261e-4be7-40c3-bf5b-c734a30f6d1a-c000.json
-rw-r--r--   1 hadoop hadoop      12794 2022-04-12 18:46 /user/hadoop/country_kpi/part-00000-f767ddfc-e47b-4520-ab91-c9fba4627434-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-12 19:03 /user/hadoop/country_kpi/part-00000-fa74d646-e10b-48ce-8819-16fdd50a1578-c000.json
```

- Use 'cat' command to view the data **hadoop fs -cat /user/hadoop/time_kpi/part***



```
return":0.09090909090909091}
{"window":{"start":"2022-04-10T20:51:00.000Z","end":"2022-04-10T20:52:00.000Z"},"OPM":7,"total_sale_volume":459.38999366760254,"average_transaction_size":65.62714195251465,"rate_o
return":0.0}
{"window":{"start":"2022-04-11T14:29:00.000Z","end":"2022-04-11T14:30:00.000Z"},"OPM":14,"total_sale_volume":771.0200079679489,"average_transaction_size":55.07285771199635,"rate_o
return":0.14285714285714285}
{"window":{"start":"2022-04-08T13:35:00.000Z","end":"2022-04-08T13:36:00.000Z"},"OPM":12,"total_sale_volume":450.9099915623665,"average_transaction_size":37.575832630197205,"rate_
_return":0.0}
{"window":{"start":"2022-04-07T22:10:00.000Z","end":"2022-04-07T22:11:00.000Z"},"OPM":13,"total_sale_volume":723.1699977517128,"average_transaction_size":55.62846136551637,"rate_o
return":0.0}
{"window":{"start":"2022-04-11T03:56:00.000Z","end":"2022-04-11T03:57:00.000Z"},"OPM":12,"total_sale_volume":821.3600027561188,"average_transaction_size":68.4466668963432З,"rate_o
return":0.0}
{"window":{"start":"2022-04-08T02:23:00.000Z","end":"2022-04-08T02:24:00.000Z"},"OPM":5,"total_sale_volume":218.55999660491943,"average_transaction_size":43.711999320983885,"rate_
_return":0.0}
{"window":{"start":"2022-04-07T16:43:00.000Z","end":"2022-04-07T16:44:00.000Z"},"OPM":7,"total_sale_volume":407.23000288009644,"average_transaction_size":58.17571469715663б,"rate_
_return":0.0}
{"window":{"start":"2022-04-07T18:44:00.000Z","end":"2022-04-07T18:45:00.000Z"},"OPM":8,"total_sale_volume":280.39999997615814,"average_transaction_size":35.04999999701977,"rate_o
return":0.125}
{"window":{"start":"2022-04-10T20:10:00.000Z","end":"2022-04-10T20:11:00.000Z"},"OPM":11,"total_sale_volume":398.49999725818634,"average_transaction_size":36.22727247801694,"rate_
_return":0.0}
{"window":{"start":"2022-04-08T15:16:00.000Z","end":"2022-04-08T15:17:00.000Z"},"OPM":10,"total_sale_volume":551.8300104141235,"average_transaction_size":55.18300104141235,"rate_o
return":0.0}
{"window":{"start":"2022-04-11T21:22:00.000Z","end":"2022-04-11T21:23:00.000Z"},"OPM":12,"total_sale_volume":574.1399941444397,"average_transaction_size":47.844999512036644,"rate_
_return":0.08333333333333333}
{"window":{"start":"2022-04-11T15:28:00.000Z","end":"2022-04-11T15:29:00.000Z"},"OPM":16,"total_sale_volume":1075.729977607727,"average_transaction_size":67.23312360048294,"rate_o
return":0.0}
{"window":{"start":"2022-04-10T18:13:00.000Z","end":"2022-04-10T18:14:00.000Z"},"OPM":15,"total_sale_volume":707.9499979019165,"average_transaction_size":47.19666652679443,"rate_o
return":0.06666666666666667}
{"window":{"start":"2022-04-10T05:31:00.000Z","end":"2022-04-10T05:32:00.000Z"},"OPM":9,"total_sale_volume":1660.959985256195,"average_transaction_size":184.55110947291055,"rate_o
return":0.1111111111111111}
{"window":{"start":"2022-04-10T02:35:00.000Z","end":"2022-04-10T02:36:00.000Z"},"OPM":8,"total_sale_volume":789.3699922561646,"average_transaction_size":98.67124903202057,"rate_of
eturn":0.0}
{"window":{"start":"2022-04-08T12:12:00.000Z","end":"2022-04-08T12:13:00.000Z"},"OPM":10,"total_sale_volume":541.7299852371216,"average_transaction_size":54.17299852371216,"rate_o
return":0.3}
{"window":{"start":"2022-04-12T03:08:00.000Z","end":"2022-04-12T03:09:00.000Z"},"OPM":10,"total_sale_volume":467.5899920463562,"average_transaction_size":46.75899920463562,"rate_o
return":0.0}
{"window":{"start":"2022-04-08T16:10:00.000Z","end":"2022-04-08T16:11:00.000Z"},"OPM":10,"total_sale_volume":976.4099912643433,"average_transaction_size":97.64099912643432,"rate_o
return":0.1}
{"window":{"start":"2022-04-09T21:55:00.000Z","end":"2022-04-09T21:56:00.000Z"},"OPM":13,"total_sale_volume":376.1299942135811,"average_transaction_size":28.933076477967777,"rate_
_return":0.07692307692307693}
{"window":{"start":"2022-04-08T03:31:00.000Z","end":"2022-04-08T03:32:00.000Z"},"OPM":13,"total_sale_volume":978.2099975347519,"average_transaction_size":75.2469228872886,"rate_of
eturn":0.07692307692307693}
{"window":{"start":"2022-04-08T05:16:00.000Z","end":"2022-04-08T05:17:00.000Z"},"OPM":6,"total_sale_volume":669.8600025177002,"average_transaction_size":111.64333375295003,"rate_o
return":0.0}
{"window":{"start":"2022-04-09T08:40:00.000Z","end":"2022-04-09T08:41:00.000Z"},"OPM":12,"total_sale_volume":354.0099980831146,"average_transaction_size":29.500833173592884,"rate_
_return":0.0}
{"window":{"start":"2022-04-10T04:50:00.000Z","end":"2022-04-10T04:51:00.000Z"},"OPM":5,"total_sale_volume":297.5399971008301,"average_transaction_size":59.50799942016601б,"rate_o
return":0.0}
{"window":{"start":"2022-04-11T12:20:00.000Z","end":"2022-04-11T12:21:00.000Z"},"OPM":13,"total_sale_volume":925.9299949407578,"average_transaction_size":71.22538422621213,"rate_o
return":0.0}
{"window":{"start":"2022-04-10T07:47:00.000Z","end":"2022-04-10T07:48:00.000Z"},"OPM":10,"total_sale_volume":1187.0199924707413,"average_transaction_size":118.70199924707413,"rate
```

- Create directories for time-based and then time-and-country-based KPIs as hadoop-user. Using the 'get' command copy the contents of the output folders into the hadoop system.

  mkdir timebased-KPI
  hadoop fs -get /user/hadoop/time_kpi /home/hadoop/timebased-KPI

  mkdir country-with-timebased-KPI
  hadoop fs -get /user/hadoop/country_kpi /home/hadoop/country-with-timebased-KPI

- Use WinSCP to establish a connection between the EMR instance and local file system to transfer all the required files into my system.