Code Logic - Retail Data Analysis

Submitted By:

Manish Pramanik (pramanik.manish@gmail.com)
Nandanavanam Dinakar (itsdinakar98@gmail.com)
Abhay Kumar (abhaykumariiit@gmail.com)

Set Up:

Initiate an AWS EMR cluster with Hadoop and spark service.
Environment set up to integrate kafka with spark job,

- export SPARK_KAFKA_VERSION=0.10

Execute Stream Job:

Create the script file **spark-streaming.py** to to run a Spark Streaming job where you are using connecting with a Kafka stream
Command used for run spark streaming job:

- spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 spark-streaming.py | tee Console-output

  spark-submit to execute spark job, --package to download dependency package required to run job and | tee ConsoleOutput would save printed streaming data on console to ConsoleOutput file

Code Logic of Script **spark-streaming.py :**

1. Import required pyspark packages
2. Initiate spark session
3. Define bootstrap server and kafka-topic provided, from which we would read stream retail data
4. **orderRawStream** : To read raw stream from defined bootstrap server and kafka-topic

5. **orderSchema** :Define Spark schema to parse json format stream from raw stream of data
6. **orderStream** : Parsing of stream of json format of stream using orderRawStream schema.
7. To obtain derived fields, utility functions are required and defined. The functions are:

    - is_order: flag 1 if order type is ORDER else 0
    - is_return: flag 1 if order type is RETURN else 0
    - total_item_count: to count the number of items in each order
    - total_cost: total cost which computed from item quantity * item price for each order. Taken as negative if order type is RETURN.

8. Define User Defined function using UDFs using above created utility functions.

9. **expandedOrderStream:** Define stream with required derived columns. The stream constitutes base columns invoice_no, country, timestamp and derived columns total_cost, total_items, is_order, is_return
10. **orderStreamConsole:** Generate stream and print in console. Stream generated in append mode with 1 minute process time.
11. **aggStreamByTime:** Generate stream on computing aggregation to obtain Key Performance Indicators(KPI) based on window of time.
    An event time columns taken for window time. Here timestamp column is event time columns. Aggregation of total_sale_volume, average_transaction_size, rate_of_return calculated for windows of 1 minute and 1 minute sliding interval. To aggregate based on tumbling window, window and sliding interval need to be kept equal which is here 1 minute. Required interval 10 minutes, thus processing time kept 10 minutes.

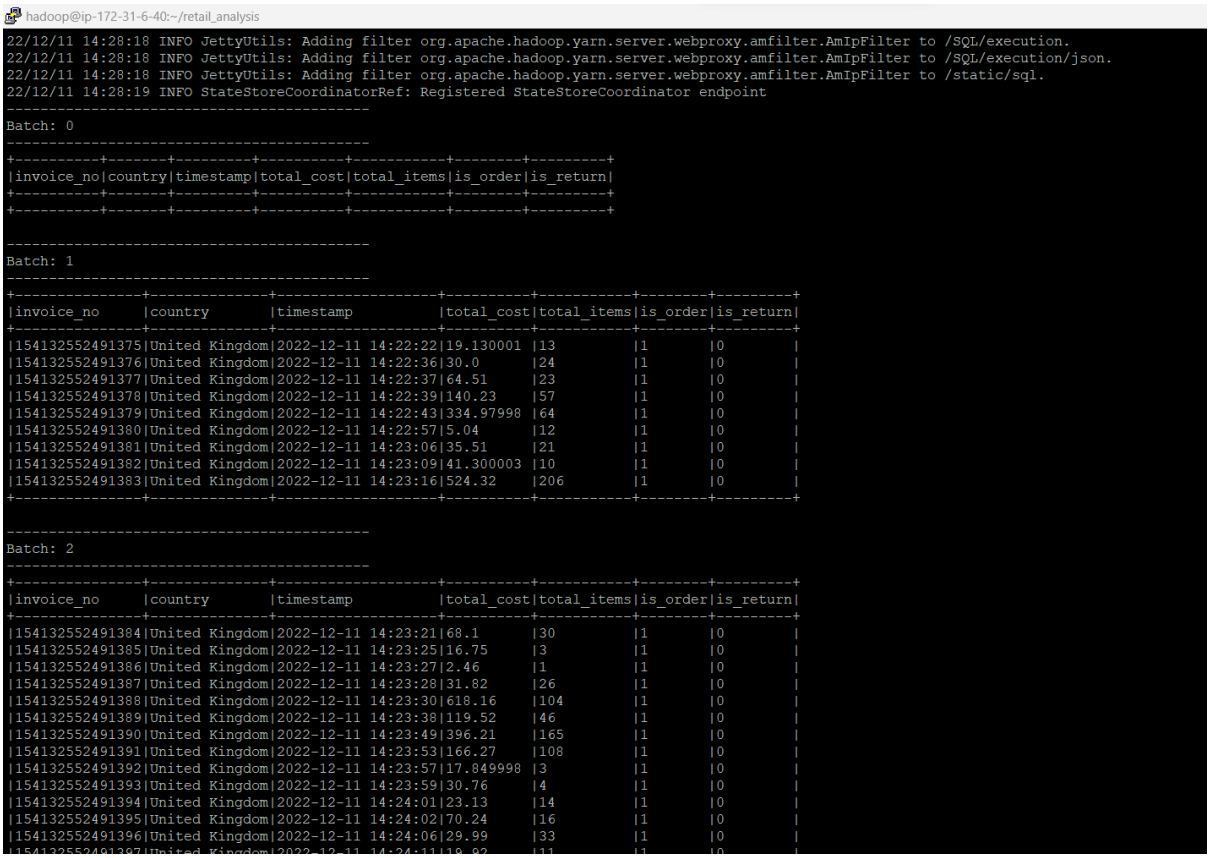
    The stream executed in json mode and stored in hdfs location ../Timebased-KPI

    To get KPI stored in json files from hdfs to local storage run
    - hadoop fs -get /user/hadoop/Timebased-KPI ./

12. **aggStreamByTimeCountry:** Country and Time based KPI computed. Here, grouping is done on window of time as well as additionally 'country' column. The stream in json format to be saved on hdfs location ../ Country-and-timebased-KPI

    To get KPI stored in json files from hdfs to local storage run
    - hadoop fs -get /user/hadoop/Country-and-timebased-KPI ./

Console Output:



Retrieval of generated data from local EMR instance:
zip -r Timebased-KPI.zip Timebased-KPI
zip -r Country-and-timebased-KPI.zip Country-and-timebased-KPI
Timebased-KPI.zip and Country-and-timebased-KPI.zip obtained using data transfer tool winscp.