Code Logic - Retail Data Analysis

In this document, you will describe the code and the overall steps taken to solve the project. Following steps were taken for the code:

1) Imported all necessary python libraries and functions for executing spark sql
2) Create spark session
3) Connect to the kafka server where data was hosted to read the source data set
4) Define schema to read columns by assigning appropriate data types.
5) As the data is stored in binary format in Kafka, the cast to string type was done
6) Used the explode function to read the nested data in source file and stored the data in a dataframe where each nested columns will now be available in a columnar structure.
7) Define UDF's. As per the requirement, UDF's were to be calculated which is done as below:

   Each order will contain returns and orders, if type is return then that amount has to be deducted hence multiplied with -1 to make amount negative else total sales will be unit price* quantity

   ```
   #UDF to determine total_sales
   def total_sales(type,unit_price,quantity):
       if (type == 'RETURN'):
           return (-1*(unit_price*quantity))
       else:
                   return (unit_price*quantity)
   ```

   this is a flag to indicate return or no return

   ```
   # UDF to determine Is_Return in case of Return.
   def Is_Return(type):
       if type == "RETURN":
           return(1)
       else:
           return(0)
   ```

   flag to indicate order or not order and not return

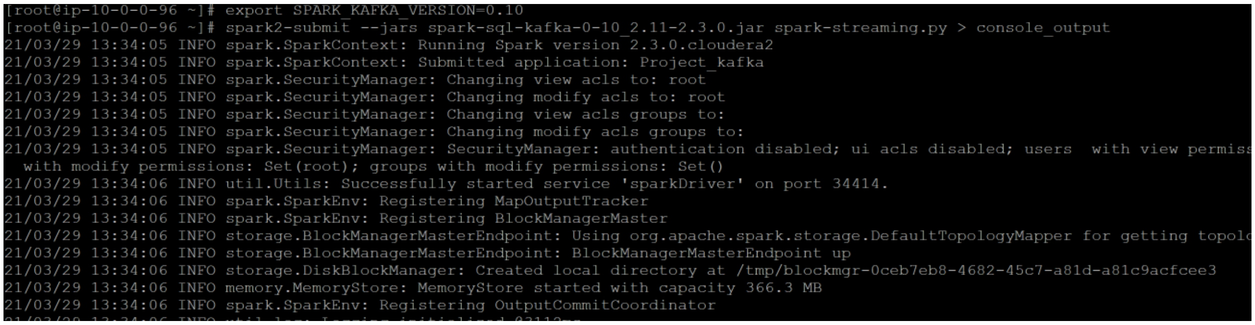
   ```
   # UDF to determine Is_Order in case of Order.
   def Is_Order(type):
       if type == "ORDER":
           return(1)
       else:
           return(0)
   ```

8) The data was then written to console for a tumbling window of one minute, with write format of append to avoid repeated data, and considering a watermark of 1 second. This can be increased or decreased based on the requirement.

9) The next step was to calculate KPI's as per the instructions in project description for a tumbling window of one min. This was done using the window function.

10) Last step was to write data in json on hdfs..

11) To run script, go to ec2 instance and login as root user and run the script as follows:



```
[root@ip-10-0-0-96 ~]# export SPARK_KAFKA_VERSION=0.10
[root@ip-10-0-0-96 ~]# spark2-submit --jars spark-sql-kafka-0-10_2.11-2.3.0.jar spark-streaming.py > console_output
21/03/29 13:34:05 INFO spark.SparkContext: Running Spark version 2.3.0.cloudera2
21/03/29 13:34:05 INFO spark.SparkContext: Submitted application: Project_kafka
21/03/29 13:34:05 INFO spark.SecurityManager: Changing view acls to: root
21/03/29 13:34:05 INFO spark.SecurityManager: Changing modify acls to: root
21/03/29 13:34:05 INFO spark.SecurityManager: Changing view acls groups to:
21/03/29 13:34:05 INFO spark.SecurityManager: Changing modify acls groups to:
21/03/29 13:34:05 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permiss
  with modify permissions: Set(root); groups with modify permissions: Set()
21/03/29 13:34:06 INFO util.Utils: Successfully started service 'sparkDriver' on port 34414.
21/03/29 13:34:06 INFO spark.SparkEnv: Registering MapOutputTracker
21/03/29 13:34:06 INFO spark.SparkEnv: Registering BlockManagerMaster
21/03/29 13:34:06 INFO storage.BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topolo
21/03/29 13:34:06 INFO storage.BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
21/03/29 13:34:06 INFO storage.DiskBlockManager: Created local directory at /tmp/blockmgr-0ceb7eb8-4682-45c7-a81d-a81c9acfcee3
21/03/29 13:34:06 INFO memory.MemoryStore: MemoryStore started with capacity 366.3 MB
21/03/29 13:34:06 INFO spark.SparkEnv: Registering OutputCommitCoordinator
```

12) Manually terminate the session from putty , until then data will be written to console_output and later to respective hdfs paths were json output is recorded.