Scripts Execution

solution to the batch layer problem.

1) Solution is on Pyspark.
2) To load data which is present in RDS to HDFS using Sqoop import commands.

   Table 1 (member_score) sqoop import \ --connect jdbc:mysql://upgradawsrds1.cyaielc9bmnf.us-east 1.rds.amazonaws.com/cred_financials_data \ --table member_score \ -- username upgraduser --password upgraduser \ -- target-dir /user/root/cap_project/member_score \ -m 1

   Table 2 (card_member) sqoop import \ --connect jdbc:mysql://upgradawsrds1.cyaielc9bmnf.us-east 1.rds.amazonaws.com/cred_financials_data \ --table card_member \ -- username upgraduser --password upgraduser \ -- target-dir /user/root/cap_project/card_member \ -m 1
3) We will Load card_transactions.csv to HDFS after moving it to EC2-USER by using below command **hadoop fs -copyFromLocal / home/ec2- user/card_transaction.csv cap_project/ card_transaction.csv**
4) Connect to putty instance and load jupyter notebook from root user, by using command
   jupyter notebook --port 7861 --allow-root
5) Open a new notebook and load a spark context.
6) Start reading all 3 files namely in Pyspark notebook into predefined file schemas
   a. CARD_MEMBER,
   b. MEMBER_SCORE
   c. CARD_TRANSACTIONS.
7) Once read is successful, use below command to see data is read successfully.
   df.show()
8) Once we load all input data, next we need to join all these files and extract only relevant fields out of them that are need for our analysis.
9) First join card_member & member_score dataframes and slide credit score into card_member by using member_id field as join key .
10) With the fresh df(Dataframe), use member ID once again as common key and join with card_transaction.csv to load postcode, pos_id, status, amount & transaction date fields from history transactions.
11) To arrive at derived columns like latest_transaction date, group the combined data frame on card_id such that all transactions on same card id collate and get max(transaction date). Append this column to combined data frame.
12) Calculate the UCL value that mainly revolves around “amount” field. We all know UCL can be calculated as moving **average + 3 *(standard deviation).** Hence we open a window frame where we group input dfrows on card_id and order by transaction date to get all transactions on card in chronological order.
13) Now, once you group & order by transactions rank these chronological transactions starting from 1 till go on.
14) Pick rows only whose rank is less than 10, by which we select moving average of top 10 latest transactions done on card_id.

15) Import SQL functions library in pyspark and perform **avg()** function on top 10 rows of grouped card_id.
16) Similarly, perform **stddev()** to derive standard deviation on these top 10 rows selected by rank.
17) Perform computation as per formula given to deduce UCL value and append this to original dataframe obtained at step 11
18) Using happybase to load this dataframe into NoSQL database i.e., Hbase.
19) Create a connection to Hbase, check if table you want to create already exists and create one if it doesn't exist.
20) Batch load data from data frame to table created.