Assignment – Capstone Project
Mid submission

Credit Card Fraud Detection System - LogicMid.pdf

By: Shubhra Sinha

Table of Contents

Batch Processing Explanation

This activity involves initial loading of bulk data, data ingestion from the RDS, writing the necessary hive and hbase scripts to load data into NoSQL database (HBase) and create the lookup table depending on UCL (Upper Control Limit)

This document is arranged to explain:

1. Section I - Task-wise explanation in the further sections
2. Section II - List of all the used tables in hive/hbase
3. Section III - All the scripts used in the tasks (along with the basic commands for initial setup)

Section 1 - Task-wise explanation

Task 1

a) Winscp **card_transactions.csv** to EC2
b) Create the internal card_transactions_tmp temporary table on hive
c) Load given card_transactions.csv to card_transactions_tmp table
d) Create HBase empty table card_transactions with table name and column family. This is required to create the external table in hive. External table is needed so that data is not removed even if hive table is dropped.
e) Create HIVE-HBase integrated external card_transaction table in hive
f) Insert data into external table card_transaction (HIVE-Hbase integrated table) from card_transactions_tmp temporary table
g) Verify if the data is loaded successfully or not in hbase-hive integrated table or not.

Task 2

a) Create the sqoop job to load data for card_member table from RDS. Since all the fields are of type string except member_joining_dt hence it is used for the check-column condition for incremental load. The hive table is created at the same location as of target-directory since it will load the data automatically from the location and load data is not required. (Load data command is although explicitly mentioned to do so).
b) Verify the created sqoop job for card_member table and verify the configuration.
c) Execute the job for the initial load.
d) Verify the data in card_member table if load is successful.
e) Create the sqoop job to ingest member_score. There is no field that can be used for incremental load hence the complete data needed to be imported and refresh in every 4 hours.
f) Verify the created sqoop job for member_score table and verify the configuration.
g) Execute the job for the initial load.
h) Verify the data in member_score table if load is successful.

Task 3

This task involves creation of **lookup table**, with various values including card_id specific UCL (upper control limit) based on the GENUINE card_transaction details

a) UCL has to be calculated as: (Moving Average)+3×(Standard Deviation)
    a. Calculate moving average and standard deviation, by selecting latest 10 GENUINE card_transactions for each of the cards (card_id), and then derive UCL by partitioning by card_id
    b. rank GENUINE transactions by row_number, in the descending order of the transaction date
    c. pick <= 10 records – if any card has less than 10 GENUINE transactions at a given time, it shall pick available GENUINE transactions
b) Get postcode, and transaction_dt from the last GENUINE card transaction
c) Get score value for the member_id from member_score for the corresponding card_id
d) Insert data into NoSQL (Hive – HBase integrated) table with card_id,UCL, postcode, transaction_dt, score

Task 4

Refer to section “Scripts and Commands”

Section II - List of all the used tables in hive/hbase

Name	Description	Database
card_transactions_tmp	Temporary table to load the card_transactions.csv	Hive
card_transaction	Table in hive to hold data for hive-hbase integrated table	Hive
card_transactions	HBase (NoSQL) table to hold the final card_transactions data	HBase
card_lookup	Table in hive to hold the lookup data for hive-hbase integrated table	Hive
card_lookup	HBase (NoSQL) entity of the Hive-HBase integrated tables to hold the lookup data	HBase
card_member	To hold RDS imported data	Hive
member_score	To hold RDS imported data	Hive
card_lookup_raw	Temporary table to hold the computed card_id specific UCL details	Hive
card_lookup_tmp	Temporary table to hold the card_member, member_score and UCL details	Hive
lookup_tmp_trans	Staging (intermediate) table to hold the last (latest) GENUINE card_transactions’s transaction date and postal code details	Hive

Section III - All the scripts used in the tasks (along with the basic commands for initial setup)

Scripts

Task 1:

Write a script to load the transactions history data (card_transactions.csv) in a NoSQL database. Write a script to create a look-up table with columns specified earlier in the problem statement.

Scripts:

1. Script to create (INTERNAL) card_transactions temporary hive table to load the given CSV

```
hive> CREATE TABLE card_transactions_tmp (card_id string, member_id string,
amount double, postcode string, pos_id bigint, transaction_dt string, status
string)  ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION
'/user/root/creditcard/temporary/card_transactions_tmp' TBLPROPERTIES
("skip.header.line.count"="1");
```

2. Script to load initially card_transactions data from csv

```
hive> load data local inpath '/home/ec2-user/card_transactions.csv'
overwrite into table card_transactions_tmp;
```

3. Check the data loaded

```
hive> select * from card_transactions_tmp limit 10;
```

4. Create card_transactions table (HIVE-HABSE integrated table)

```
[root@ip-10-0-0-201 ~]# hbase shell
hbase(main):001:0> create 'card_transactions','cardtransactions';
hbase(main):002:0* create 'card_lookup','lookup';
```

NOTE: For creating HIVE-HBASE integrated table, we need to first create a HBase table and then integrate with hive table, HIVE external creation works only on existing HBase table.

Creating external table in hive

```
hive> CREATE EXTERNAL TABLE card_transaction (key struct<member_id:string,
transaction_dt:string, amount:double>, card_id string,  postcode string,
pos_id bigint,  status string)  ROW FORMAT DELIMITED COLLECTION ITEMS
TERMINATED BY '~'  STORED BY
'org.apache.hadoop.hive.hbase.HBaseStorageHandler'  WITH SERDEPROPERTIES
("hbase.columns.mapping" = ":key, cardtransactions:card_id,
cardtransactions:postcode, cardtransactions:pos_id,
cardtransactions:status") TBLPROPERTIES("hbase.table.name" =
"card_transactions","hbase.mapred.output.outputtable" =
"card_transactions");
```

5. Insert data from card_transactions_tmp table to hive-hbase table

```
hive> INSERT OVERWRITE TABLE card_transaction SELECT
named_struct('member_id',ct.member_id,'transaction_dt',ct.transaction_dt,'am
ount',ct.amount), ct.card_id, ct.postcode, ct.pos_id, ct.status  FROM
card_transactions_tmp ct;
```

6. Check if the data is inserted in hive

```
hive> select * from card_transaction limit 10;
```

7. Check if the data is inserted in hbase

```
hbase(main):002:0> scan 'card_transactions', {'LIMIT' => 5}
```

8. Script to create final lookup table (HIVE-HABSE integrated)

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS card_lookup (
    > card_id string,
    > UCL decimal,
    > postcode string,
    > transaction_dt string,
    > score bigint)
    > STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
    > WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,
    > lookup:UCL,
    > lookup:postcode,
    > lookup:transaction_dt,
    > lookup:score")
    > TBLPROPERTIES("hbase.table.name" = "card_lookup" ,
    > "hbase.mapred.output.outputtable" = "card_lookup" );
```

Task 2

Write a script to ingest the relevant data from AWS RDS to Hadoop.

Scripts:

1. Switching to hdfs user

```
#> sudo -i su - hdfs
```

2. Create Sqoop job to import card_member data incrementally from AWS RDS

```
#> set sqoop.metastore.client.record.password=true;
#> sqoop job --create job_card_member_incremental -- import --connect
jdbc:mysql://upgradawsrds.cpclxrkdvwmz.us-east-
1.rds.amazonaws.com/cred_financials_data -username upgraduser --password upgraduser
--table card_member --incremental append --check-column member_joining_dt --last-
value "2000-01-01 00:00:00" --target-dir /user/root/creditcard/final/card_member
```

3. Look for the listed Sqoop job

```
#> sqoop job --list
```

4. View the configuration of the job

```
#> sqoop job --show job_card_member_incremental
```

5. Execute the job to load initial data

```
#> sqoop job --exec job_card_member_incremental
```

6. List contents of /user/root/creditcard/card_member, to check if the import is working

```
#> hadoop fs -ls /user/root/creditcard/final/card_member
```

7. Create card_member table

```
#> CREATE EXTERNAL TABLE IF NOT EXISTS card_member ( card_id string, member_id
string, member_joining_dt timestamp, card_purchase_dt string, country string, city
string) row format delimited fields terminated by ',' location
'/user/root/creditcard/final/card_member';
```

8. Check if the data is inserted (Data automatically loaded into table)

```
#> select * from card_member limit 10;
```

9. Script to insert/load data from hadoop into card_member table (Load data is not required)

```
#> load data inpath '/user/root/creditcard/final/card_member/part*' into table
card_member;
```

10. Create Sqoop job to import member_score data from AWS RDS (It need to be refreshed in 4 hours)

```
#> set sqoop.metastore.client.record.password=true;
#> sqoop job --create job_member_score_complete -- import --connect
jdbc:mysql://upgradawsrds.cpclxrkdvwmz.us-east-
1.rds.amazonaws.com/cred_financials_data -username upgraduser --password upgraduser
--table member_score --target-dir /user/root/creditcard/temporary/member_score
```

11. Execute to load the initial data

```
#> sqoop job --exec job_member_score_complete
```

12. Create member_score table

```
#> CREATE EXTERNAL TABLE IF NOT EXISTS member_score ( member_id string, score
bigint) row format delimited fields terminated by ',' location
'/user/root/creditcard/final/member_score';
```

13. Load member_score table

```
#> load data inpath '/user/root/creditcard/temporary/member_score/part*' overwrite
into table member_score;
```

14. Check if the data is inserted

```
#> select * from member_score  limit 10;
```

Task 3:

Write a script to calculate the moving average and standard deviation of the last 10 transactions for each card_id for the data present in Hadoop and NoSQL database. If the total number of transactions for a particular card_id is less than 10, then calculate the parameters based on the total number of records available for that card_id. The script should be able to extract and feed the other relevant data ('postcode', 'transaction_dt', 'score', etc.) for the look-up table along with card_id and UCL.

Scripts:

1. Create raw lookup table

```
#> CREATE TABLE IF NOT EXISTS card_lookup_raw ( card_id string, member_id string,
ucl decimal) LOCATION '/user/root/creditcard/temporary/card_lookup_raw';
```

2. Insert data into card_lookup_raw table

```
#> INSERT OVERWRITE TABLE card_lookup_raw SELECT card_id,  member_id, (AVG(amount)
+ (3 * STDDEV_POP(amount))) as ucl  FROM (SELECT card_id,  key.member_id as
member_id,  key.amount as amount, row_number() OVER (PARTITION BY card_id order by
UNIX_TIMESTAMP(key.transaction_dt, 'dd-MM-yyyy HH:mm:ss') desc) as rank FROM
card_transaction WHERE status = "GENUINE") a WHERE rank <= 10 GROUP BY card_id,
member_id;
```

3. Check if the data is inserted

```
#> select * from card_lookup_raw limit 10;
```

4. Create temporary lookup table

```
#> CREATE TABLE IF NOT EXISTS card_lookup_tmp ( card_id string, member_id string,
ucl decimal, score bigint) LOCATION
'/user/root/creditcard/temporary/card_lookup_tmp';
```

5. Insert UCL, score data into card_lookup_tmp by joining member_score and card_lookup_raw tables

```
#> INSERT OVERWRITE TABLE card_lookup_tmp SELECT r.card_id, r.member_id, r.ucl,
m.score FROM card_lookup_raw r JOIN member_score m ON (r.member_id = m.member_id);
```

6. Check if the data is inserted

```
#> select * from card_lookup_tmp limit 10;
```

7. Create lookup temporary table for card transaction

```
#> CREATE TABLE IF NOT EXISTS lookup_tmp_trans ( card_id string, postcode string,
transaction_dt string) LOCATION '/user/root/creditcard/temporary/
lookup_tmp_trans';
```

8. Insert into lookup_tmp_trans, a record with latest GENUINE transaction date and postcode

```
#> INSERT OVERWRITE TABLE lookup_tmp_trans SELECT card_id,  postcode,
transaction_dt FROM (SELECT card_id , postcode, key.transaction_dt as
transaction_dt, row_number() OVER (PARTITION BY card_id  ORDER BY
UNIX_TIMESTAMP(key.transaction_dt, 'dd-MM-yyyy HH:mm:ss') desc) as rank FROM
card_transaction WHERE status = 'GENUINE') a WHERE rank == 1;
```

9. Check if the data is inserted correctly

```
#> select * from lookup_tmp_trans limit 10;
```

10. Insert into final lookup table (hive-hbase integrated)

```
#> INSERT OVERWRITE TABLE card_lookup SELECT l.card_id, l.ucl, t.postcode,
t.transaction_dt, l.score FROM card_lookup_tmp l  JOIN lookup_tmp_trans t ON
l.card_id = t.card_id;
```

11. Check if the data is inserted correctly

```
#> select * from card_lookup limit 10;
```

Basic Commands

Following are some of the basic commands required:

```
Switching to root and hdfs user
#> sudo -i
#> su - hdfs

Creating the HDFS directory
#> hadoop fs -mkdir /user/root/creditcard
#> hadoop fs -mkdir /user/root/creditcard/temporary
#> hadoop fs -mkdir /user/root/creditcard/final
#> hadoop fs -chown root /user/root/creditcard/temporary
#> hadoop fs -chmod 777 /user/root/creditcard/temporary
#> hadoop fs -chown root /user/root/creditcard/final
#> hadoop fs -chmod 777 /user/root/creditcard/final

Delete all data
#> hadoop fs -rm -r /user/root/creditcard
```

```
Remove imported data
#> hadoop fs -rm -r /user/root/cc/stg/card_member
#> hadoop fs -rm -r /user/root/cc/prod/member_score

To remove ACCUMULO warning, before running SQOOP
#> sudo mkdir /var/lib/accumulo
#> ACCUMULO_HOME='/var/lib/accumulo'
#> export ACCUMULO_HOME
```