Capstone Project

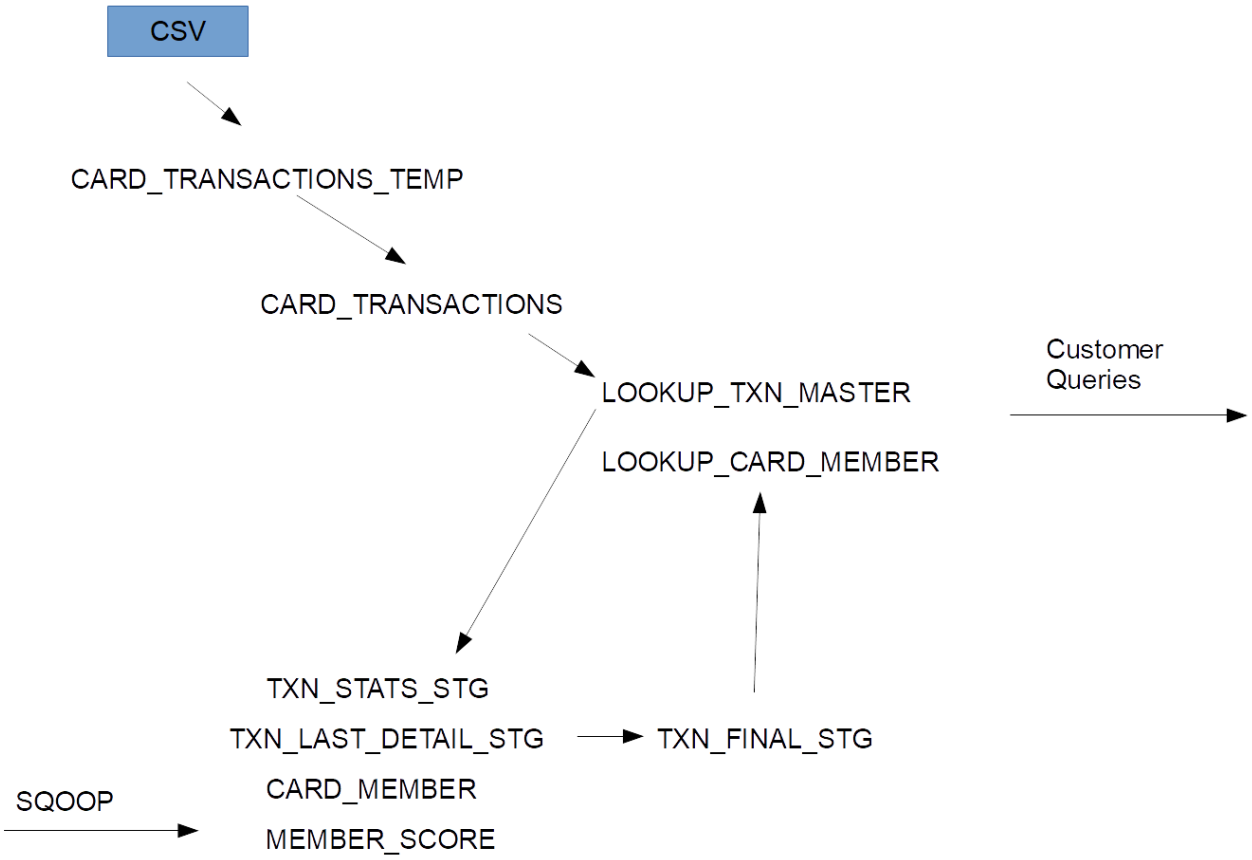
Mid Submission

Logic – Batch Layer Problem

By Venkatesh Jagannathan

Table Design

Following are the tables needed with their purpose indicated: -



LookUp

LOOKUP_CARD_MEMBER(HBASE+HIVE)

card_details:card_id (key)
last_transaction_details:ucl
last_transaction_details:postcode
last_transaction_details:transaction_dt
last_transaction_details:score
member_details:member_id
member_details:member_joining_dt
member_details:card_purchase_dt
member_details:country
member_details:city

This is the primary lookup table to hold all card wise details like ucl of last 10 transactions, last transaction details & card member details. Keeping card and member details in same table makes it efficient for customer queries and they are 1:1 relation with the card id.

LOOKUP_TXN_MASTER(HBASE+HIVE)

key(card_id~timestamp)
card_details:card_id
member_details:member_id
transaction_details:amount
transaction_details:postcode
transaction_details:pos_id
transaction_details:transaction_dt
transaction_details:status
transaction_details:txrank (Ranking of transaction in descending order of time of occurrence)

This table holds last 10 “genuine” transactions along with ranking of each transaction to obtain the last transaction easily. This table along with LOOKUP_CARD_MEMBER becomes our lookup table for all the customer queries, efficiently without need to query transaction table.

INTERMEDIATE TABLES (HIVE)

TXN_STATS_STG
card_id
moving_average
std_dev

Stores statistics of each card id

TXN_LAST_DETAIL_STG
card_id
postcode
transaction_dt

Stores last transaction detail

TXN_FINAL_STG
card_id
moving_average
std_dev
postcode
transaction_dt
score
member_id
member_joining_dt

card_purchase_dt
country string
city string

CARD_TRANSACTIONS_TEMP
card_id
member_id
amount
postcode
pos_id
transaction_d
status

Intermediate table for CSV load

Stores consolidated information of statistics, last transaction details, score, card details & member score to be used as source to update lookup table back

The other tables are as per mentioned in assignment: -
CARD_TRANSACTIONS to hold incoming transactions & initial csv upload.
CARD_MEMBER to receive card member details ingested from RDS.
MEMBER_SCORE to receive card member score data ingested from RDS.

Initial Load

In this stage, the csv is loaded into CARD_TRANSACTIONS with help of intermediate table to convert the transaction_dt to timestamp(bigint). Also at this stage, top ten genuine transactions are loaded to LOOKUP_TXN_MASTER to facilitate processing. The top ten transactions are **“ranked”** using a rank column in descending order of occurrence (latest one on top) to help us identify the order of transaction later.

Data Import

In this stage a stored scoop job is run to import data from amazon RDS database. Creating this as a sqoop job facilitates incremental imports without need to pass last value of check column manually to support increments. The import is implemented as “hive import” to directly import it as hive table. CARD_MEMBER & MEMBER_SCORE tables get created/updated by ingestion process. The password in assignment has been given on command line when job runs but an encrypted password file and sqoop’s **CryptoFileLoader** class will be used in production for ensuring security.

Calculation of UCL, Passing on last transaction details & score

The moving average and standard deviation are calculated on last ten ranked transaction in LOOKUP_TXN_MASTER table which holds last ranked transactions. If the transactions for a card is less than 10, the average and standard deviation consideration the available number of records. Last transaction post code & transaction date details are extracted. These along with card member details and member score is combined to load LOOKUP_CARD_MEMBER hbase look up table.

Scheduled Job (Every 4 hours)

The assignment solution contains output from a simple batch program that runs the hbase, hive and sqoop commands in a loop in the interval of every 4 hours. Oozie Scheduler is perfect choice for managing this workflow of different types of actions but could not be made possible in time due to infrastructure issues. The provided solution will work perfectly fine in a system if no infrastructure issues.