Music Data Analysis

A leading music-catering company is planning to analyse large amount of data received from

varieties of sources, namely mobile app and website to track the behaviour of users, classify users,

calculate royalties associated with the song and make appropriate business strategies. The file server

receives data files periodically after every 3 hours.

Fields present in the data file:

Column Name	Column Description
User_id	Unique identifier of every user
Song_id	Unique identifier of every song
Artist_id	Unique identifier of the lead artist of the song
Timestamp	Timestamp when the record was generated
Start_ts	Start timestamp when the song started to play
End_ts	End timestamp when the song was stopped
Geo_cd	Can be A for USA region, AP for Asia pacific region, J for Japan region, E for Europe and AU for Australia region
Station_id	Unique identifier of the station from where the song was played
Song_end_type	How the was terminated 0 means completed successfully 1 means song was skipped 2 means song was paused 3 means other type of failure like device issue, network error, etc.
Like	0 means song was not liked 1 means song was liked
Dislike	0 means song was not disliked 1 means song was disliked

Fields in LookUp Table

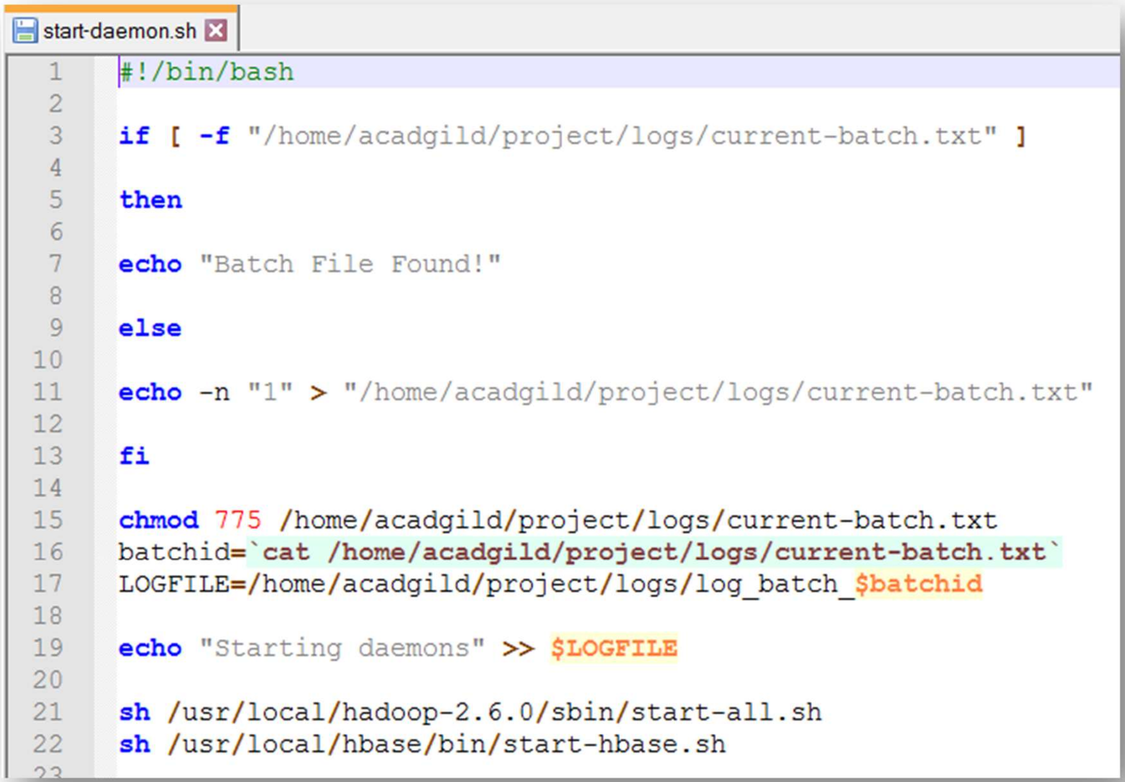
Table Name	Description
Station_Geo_Map	Contains mapping of geo_cd with station_id
Subscribed_Users	Contains user_id, subscription_start_date and subscription_end_date. Contains details only for subscribed users
Song_Artist_Map	Contains mapping of song_id with artist_id along with royalty associated with each play of the song
User_Artist_Map	Contains an array of artist_id(s) followed by a user_id

Datasets:

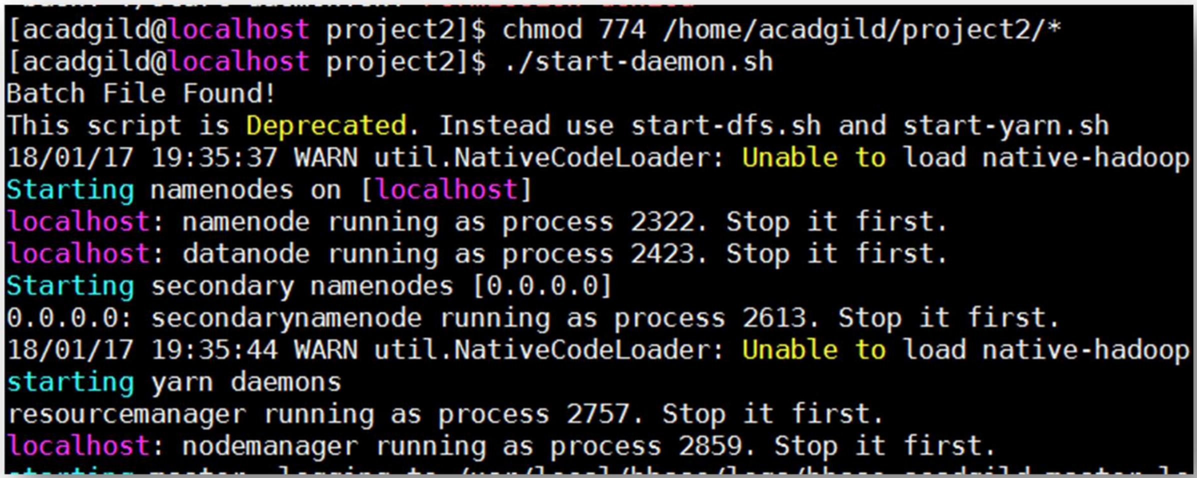
1. Data coming from web application reside in /web and has **XML** format
2. Data coming from mobile applications reside in /mob and has **csv** format
3. Data present in lookup directory should be used in HBase.

Following are few scripts and commands that we shall run.

First we will start the daemons:

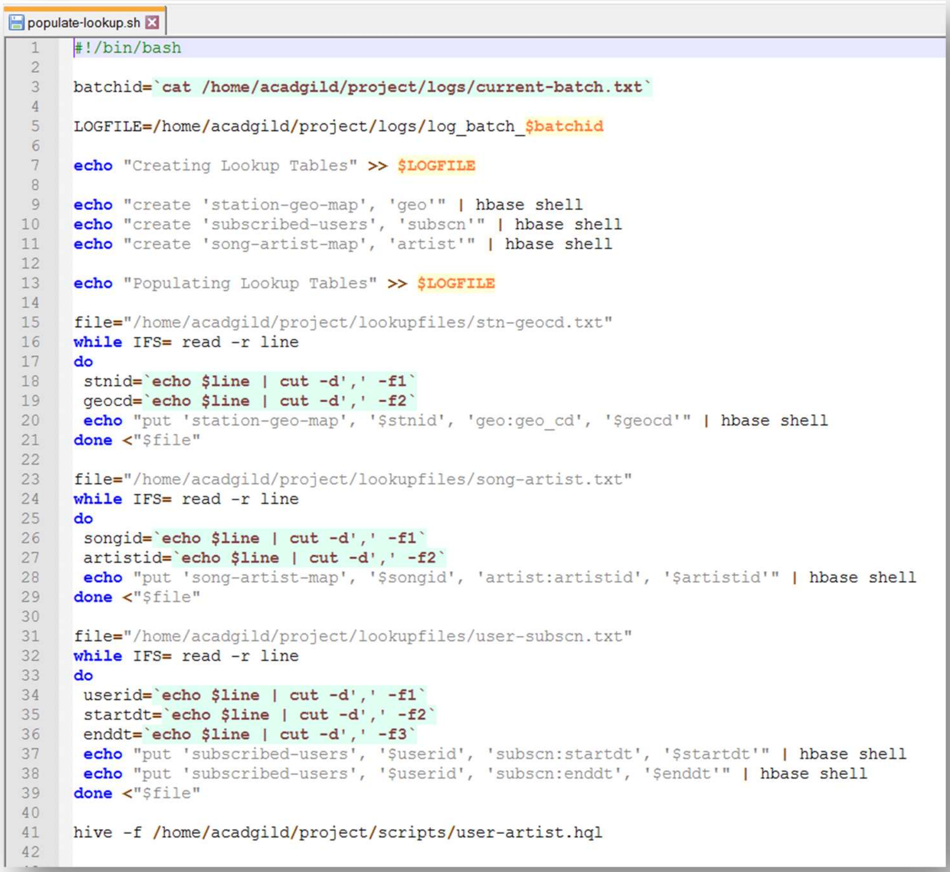


```
#!/bin/bash

if [ -f "/home/acadgild/project/logs/current-batch.txt" ]

then

echo "Batch File Found!"

else

echo -n "1" > "/home/acadgild/project/logs/current-batch.txt"

fi

chmod 775 /home/acadgild/project/logs/current-batch.txt
batchid=`cat /home/acadgild/project/logs/current-batch.txt`
LOGFILE=/home/acadgild/project/logs/log_batch_$batchid

echo "Starting daemons" >> $LOGFILE

sh /usr/local/hadoop-2.6.0/sbin/start-all.sh
sh /usr/local/hbase/bin/start-hbase.sh
```



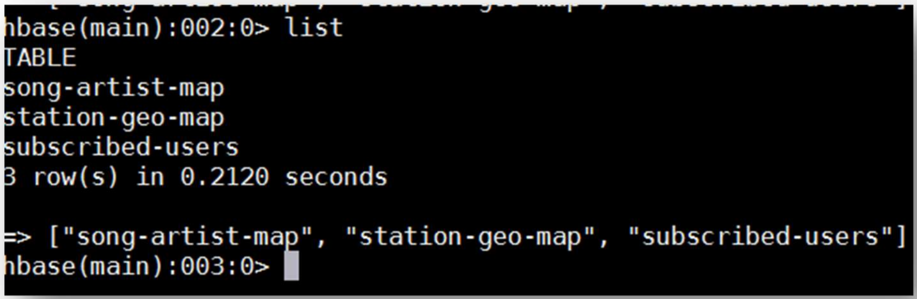
```
[acadgild@localhost project2]$ chmod 774 /home/acadgild/project2/*
[acadgild@localhost project2]$ ./start-daemon.sh
Batch File Found!
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
18/01/17 19:35:37 WARN util.NativeCodeLoader: Unable to load native-hadoop
Starting namenodes on [localhost]
localhost: namenode running as process 2322. Stop it first.
localhost: datanode running as process 2423. Stop it first.
Starting secondary namenodes [0.0.0.0]
0.0.0.0: secondarynamenode running as process 2613. Stop it first.
18/01/17 19:35:44 WARN util.NativeCodeLoader: Unable to load native-hadoop
starting yarn daemons
resourcemanager running as process 2757. Stop it first.
localhost: nodemanager running as process 2859. Stop it first.
```

Then, create the lookup tables:



```bash
#!/bin/bash

batchid=`cat /home/acadgild/project/logs/current-batch.txt`

LOGFILE=/home/acadgild/project/logs/log_batch_$batchid

echo "Creating Lookup Tables" >> $LOGFILE

echo "create 'station-geo-map', 'geo'" | hbase shell
echo "create 'subscribed-users', 'subscn'" | hbase shell
echo "create 'song-artist-map', 'artist'" | hbase shell

echo "Populating Lookup Tables" >> $LOGFILE

file="/home/acadgild/project/lookupfiles/stn-geocd.txt"
while IFS= read -r line
do
 stnid=`echo $line | cut -d',' -f1`
 geocd=`echo $line | cut -d',' -f2`
 echo "put 'station-geo-map', '$stnid', 'geo:geo_cd', '$geocd'" | hbase shell
done <"$file"

file="/home/acadgild/project/lookupfiles/song-artist.txt"
while IFS= read -r line
do
 songid=`echo $line | cut -d',' -f1`
 artistid=`echo $line | cut -d',' -f2`
 echo "put 'song-artist-map', '$songid', 'artist:artistid', '$artistid'" | hbase shell
done <"$file"

file="/home/acadgild/project/lookupfiles/user-subscn.txt"
while IFS= read -r line
do
 userid=`echo $line | cut -d',' -f1`
 startdt=`echo $line | cut -d',' -f2`
 enddt=`echo $line | cut -d',' -f3`
 echo "put 'subscribed-users', '$userid', 'subscn:startdt', '$startdt'" | hbase shell
 echo "put 'subscribed-users', '$userid', 'subscn:enddt', '$enddt'" | hbase shell
done <"$file"

hive -f /home/acadgild/project/scripts/user-artist.hql

```

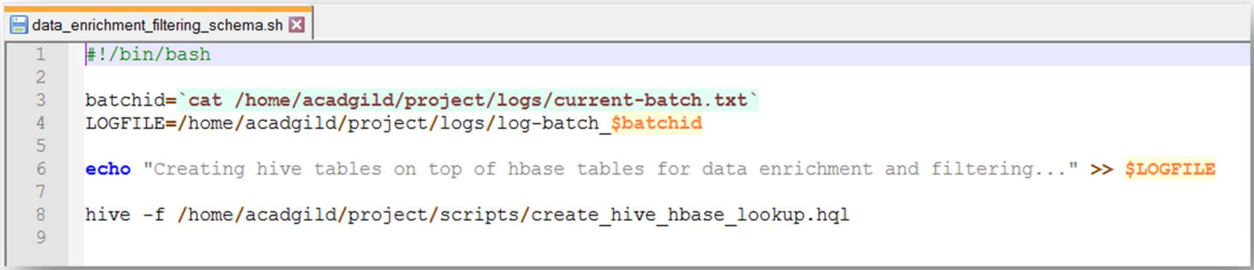
By using the above script, we will create tables in HBase:



```
hbase(main):002:0> list
TABLE
song-artist-map
station-geo-map
subscribed-users
3 row(s) in 0.2120 seconds

=> ["song-artist-map", "station-geo-map", "subscribed-users"]
hbase(main):003:0>
```

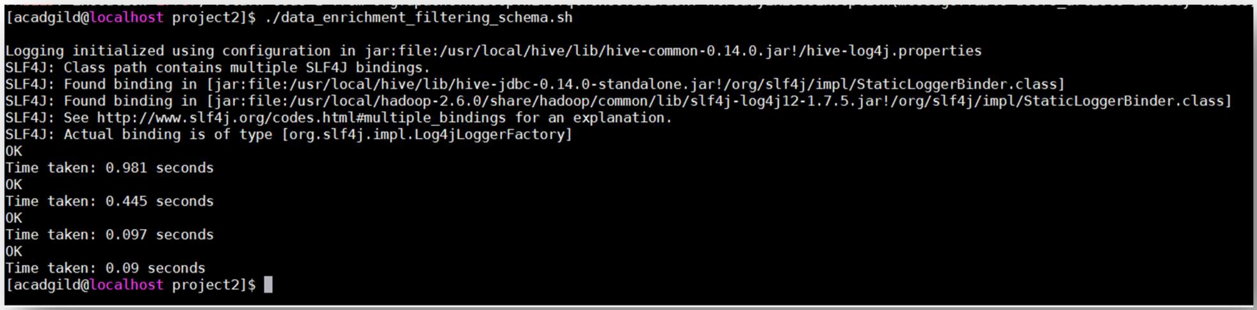
Now, we will create HIVE table over HBase tables:

In this stage with the help of Hbase storage handler & SerDe properties we are creating the hive external tables by matching the
columns of Hbase tables to hive tables.

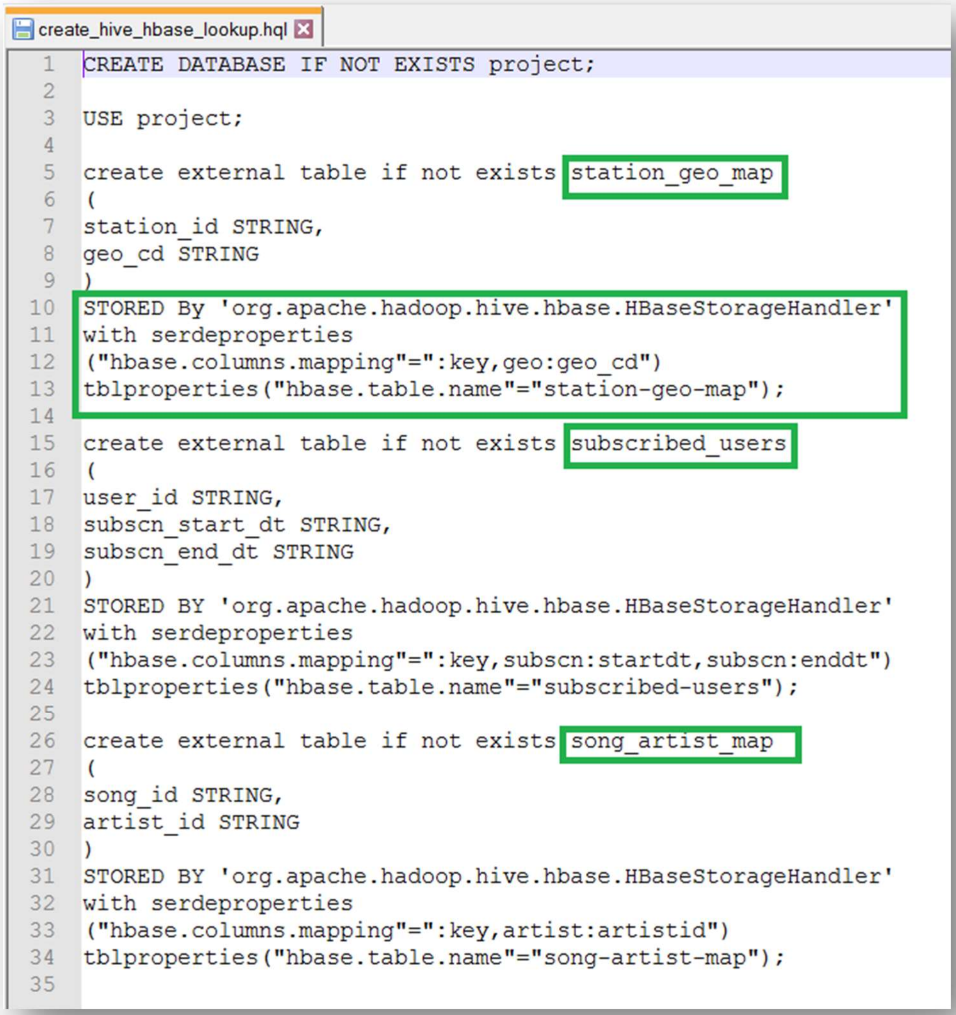


data_enrichment_filtering_schema.sh

```
#!/bin/bash

batchid=`cat /home/acadgild/project/logs/current-batch.txt`
LOGFILE=/home/acadgild/project/logs/log-batch_$batchid

echo "Creating hive tables on top of hbase tables for data enrichment and filtering..." >> $LOGFILE

hive -f /home/acadgild/project/scripts/create_hive_hbase_lookup.hql

```

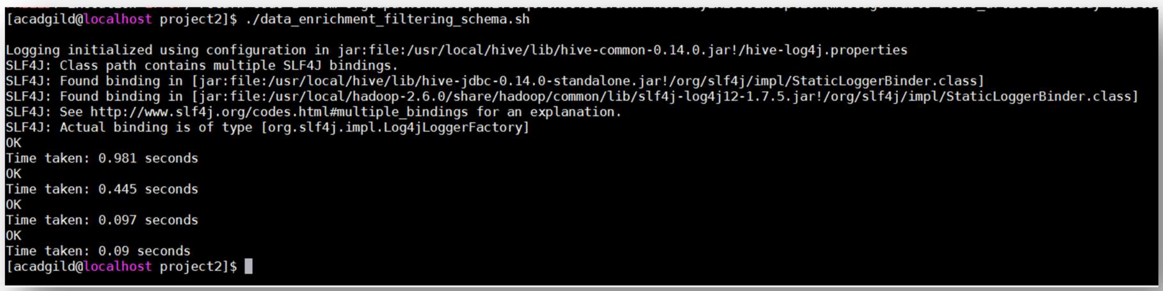


```
[acadgild@localhost project2]$ ./data_enrichment_filtering_schema.sh

Logging initialized using configuration in jar:file:/usr/local/hive/lib/hive-common-0.14.0.jar!/hive-log4j.properties
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/hive-jdbc-0.14.0-standalone.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
OK
Time taken: 0.981 seconds
OK
Time taken: 0.445 seconds
OK
Time taken: 0.097 seconds
OK
Time taken: 0.09 seconds
[acadgild@localhost project2]$
```


create_hive_hbase_lookup.hql

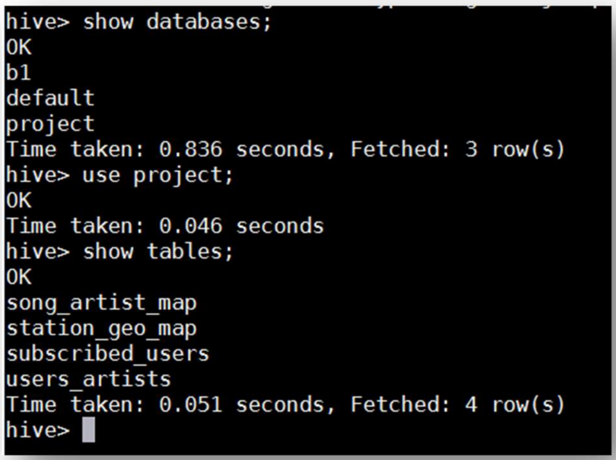
```
CREATE DATABASE IF NOT EXISTS project;

USE project;

create external table if not exists station_geo_map
(
station_id STRING,
geo_cd STRING
)
STORED By 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
with serdeproperties
("hbase.columns.mapping"=":key,geo:geo_cd")
tblproperties("hbase.table.name"="station-geo-map");

create external table if not exists subscribed_users
(
user_id STRING,
subscn_start_dt STRING,
subscn_end_dt STRING
)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
with serdeproperties
("hbase.columns.mapping"=":key,subscn:startdt,subscn:enddt")
tblproperties("hbase.table.name"="subscribed-users");

create external table if not exists song_artist_map
(
song_id STRING,
artist_id STRING
)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
with serdeproperties
("hbase.columns.mapping"=":key,artist:artistid")
tblproperties("hbase.table.name"="song-artist-map");
```

In the following screen shot we can see the tables getting created by running the scrip data_enrichemnt_filtering_schema.sh



```
[acadgild@localhost project2]$ ./data_enrichment_filtering_schema.sh

Logging initialized using configuration in jar:file:/usr/local/hive/lib/hive-common-0.14.0.jar!/hive-log4j.properties
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/hive-jdbc-0.14.0-standalone.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
OK
Time taken: 0.981 seconds
OK
Time taken: 0.445 seconds
OK
Time taken: 0.097 seconds
OK
Time taken: 0.09 seconds
[acadgild@localhost project2]$
```

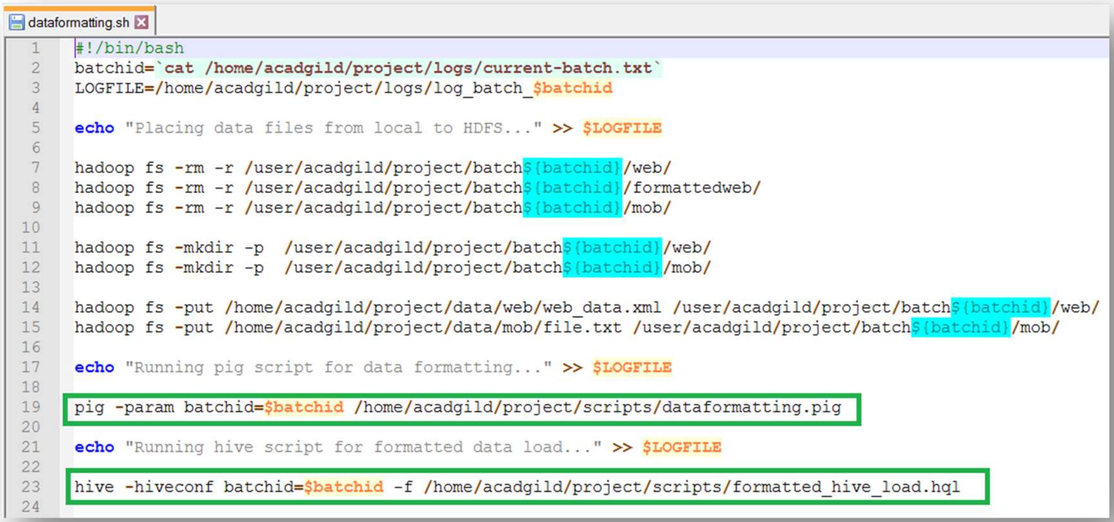
HIVE tables:



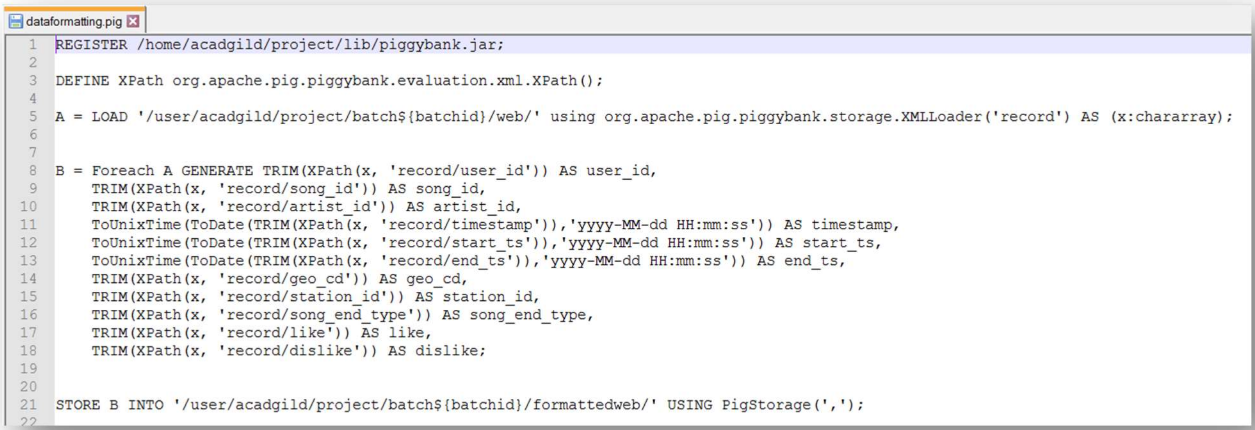
```
hive> show databases;
OK
b1
default
project
Time taken: 0.836 seconds, Fetched: 3 row(s)
hive> use project;
OK
Time taken: 0.046 seconds
hive> show tables;
OK
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 0.051 seconds, Fetched: 4 row(s)
hive>
```

DATA FORMATTING

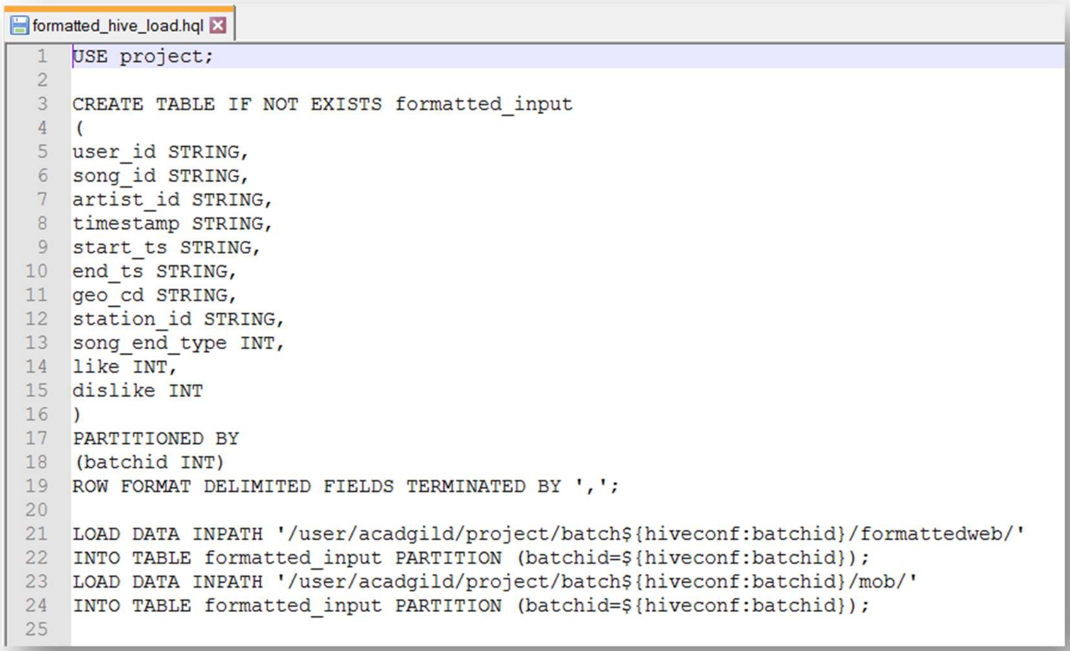
In this stage we are merging the data coming from both web applications and mobile applications and create a common table for analyzing
purpose and create partitioned data based on batchid, since we are running this scripts for every 3 hours.



```bash
#!/bin/bash
batchid=`cat /home/acadgild/project/logs/current-batch.txt`
LOGFILE=/home/acadgild/project/logs/log_batch_$batchid

echo "Placing data files from local to HDFS..." >> $LOGFILE

hadoop fs -rm -r /user/acadgild/project/batch${batchid}/web/
hadoop fs -rm -r /user/acadgild/project/batch${batchid}/formattedweb/
hadoop fs -rm -r /user/acadgild/project/batch${batchid}/mob/

hadoop fs -mkdir -p  /user/acadgild/project/batch${batchid}/web/
hadoop fs -mkdir -p  /user/acadgild/project/batch${batchid}/mob/

hadoop fs -put /home/acadgild/project/data/web/web_data.xml /user/acadgild/project/batch${batchid}/web/
hadoop fs -put /home/acadgild/project/data/mob/file.txt /user/acadgild/project/batch${batchid}/mob/

echo "Running pig script for data formatting..." >> $LOGFILE

pig -param batchid=$batchid /home/acadgild/project/scripts/dataformatting.pig

echo "Running hive script for formatted data load..." >> $LOGFILE

hive -hiveconf batchid=$batchid -f /home/acadgild/project/scripts/formatted_hive_load.hql

```



```
REGISTER /home/acadgild/project/lib/piggybank.jar;

DEFINE XPath org.apache.pig.piggybank.evaluation.xml.XPath();

A = LOAD '/user/acadgild/project/batch${batchid}/web/' using org.apache.pig.piggybank.storage.XMLLoader('record') AS (x:chararray);


B = Foreach A GENERATE TRIM(XPath(x, 'record/user_id')) AS user_id,
    TRIM(XPath(x, 'record/song_id')) AS song_id,
    TRIM(XPath(x, 'record/artist_id')) AS artist_id,
    ToUnixTime(ToDate(TRIM(XPath(x, 'record/timestamp')),'yyyy-MM-dd HH:mm:ss')) AS timestamp,
    ToUnixTime(ToDate(TRIM(XPath(x, 'record/start_ts')),'yyyy-MM-dd HH:mm:ss')) AS start_ts,
    ToUnixTime(ToDate(TRIM(XPath(x, 'record/end_ts')),'yyyy-MM-dd HH:mm:ss')) AS end_ts,
    TRIM(XPath(x, 'record/geo_cd')) AS geo_cd,
    TRIM(XPath(x, 'record/station_id')) AS station_id,
    TRIM(XPath(x, 'record/song_end_type')) AS song_end_type,
    TRIM(XPath(x, 'record/like')) AS like,
    TRIM(XPath(x, 'record/dislike')) AS dislike;


STORE B INTO '/user/acadgild/project/batch${batchid}/formattedweb/' USING PigStorage(',');
```



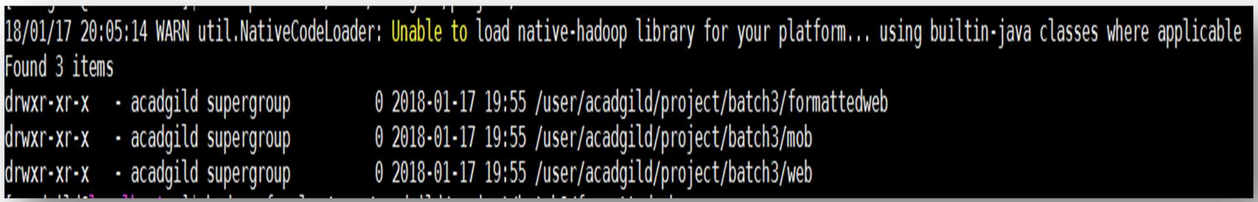
```
USE project;

CREATE TABLE IF NOT EXISTS formatted_input
(
user_id STRING,
song_id STRING,
artist_id STRING,
timestamp STRING,
start_ts STRING,
end_ts STRING,
geo_cd STRING,
station_id STRING,
song_end_type INT,
like INT,
dislike INT
)
PARTITIONED BY
(batchid INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

LOAD DATA INPATH '/user/acadgild/project/batch${hiveconf:batchid}/formattedweb/'
INTO TABLE formatted_input PARTITION (batchid=${hiveconf:batchid});
LOAD DATA INPATH '/user/acadgild/project/batch${hiveconf:batchid}/mob/'
INTO TABLE formatted_input PARTITION (batchid=${hiveconf:batchid});

```

We are running two scripts to format the data. They are:

1) Dataformatting.pig
2) Formatted_hive_load.hql

Pig script to parse the data from coming from web_data.xml to csv format and partition both web and mob data based on based on batch ID's

In the below screen shot you will find the files in HDFS folder



```
18/01/17 20:05:14 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 3 items
drwxr-xr-x   - acadgild supergroup          0 2018-01-17 19:55 /user/acadgild/project/batch3/formattedweb
drwxr-xr-x   - acadgild supergroup          0 2018-01-17 19:55 /user/acadgild/project/batch3/mob
drwxr-xr-x   - acadgild supergroup          0 2018-01-17 19:55 /user/acadgild/project/batch3/web
```

DATA ENRICHMENT

In this phase we will enrich the data coming from web and mobile applications using the lookup table stored in Hbase and divide the records based on the enrichment rules into ‘pass’ and ‘fail’ records.

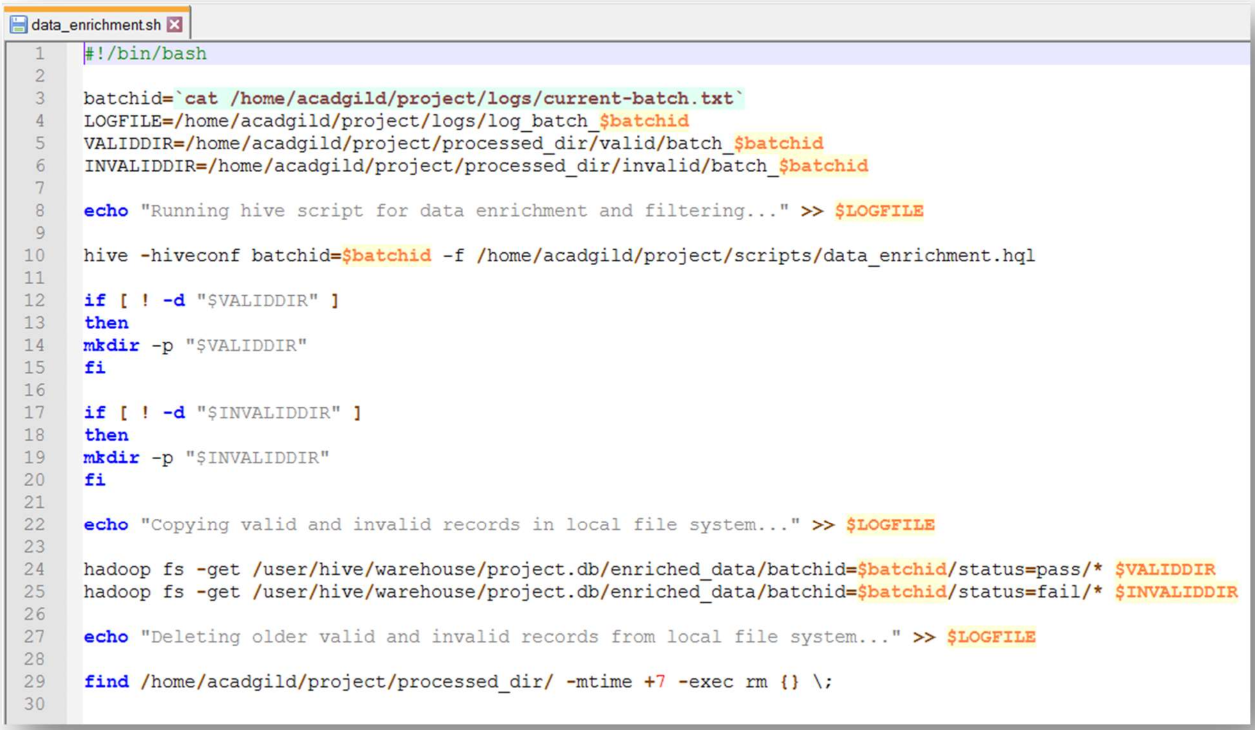
Rules for data enrichment

1. If any of like or dislike is NULL or absent, consider it as 0.

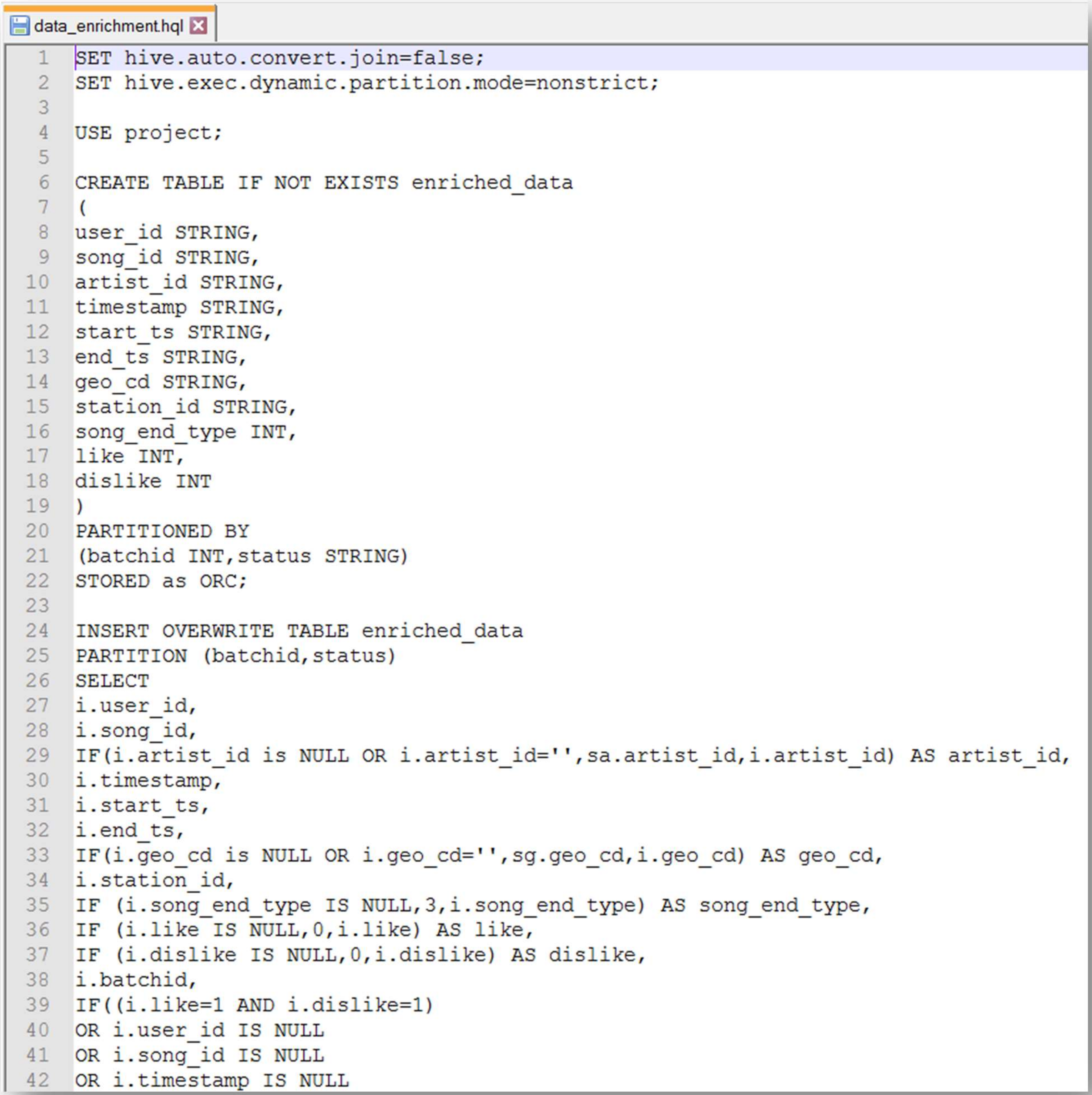
2. If fields like Geo_cd and Artist_id are NULL or absent, consult the lookup tables for fields Station_id and Song_id respectively to get the values of Geo_cd and Artist_id.

3. If corresponding lookup entry is not found, consider that record to be invalid

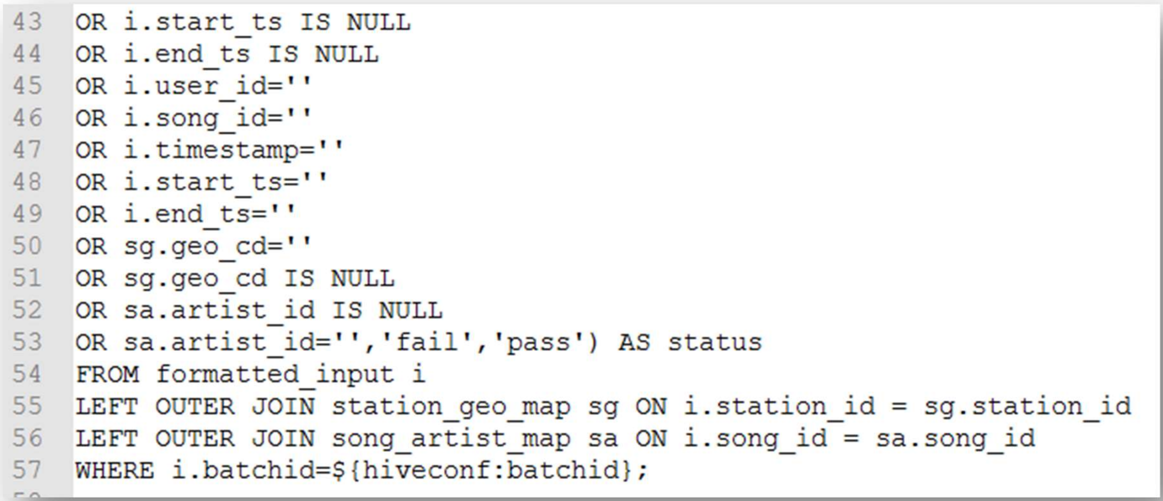
So based on the enrichment rules we will fill the null geo_cd and artist_id values with the help of corresponding lookup values in song-artist-map and station-geo-map tables in Hive-Hbase tables.



```
#!/bin/bash

batchid=`cat /home/acadgild/project/logs/current-batch.txt`
LOGFILE=/home/acadgild/project/logs/log_batch_$batchid
VALIDDIR=/home/acadgild/project/processed_dir/valid/batch_$batchid
INVALIDDIR=/home/acadgild/project/processed_dir/invalid/batch_$batchid

echo "Running hive script for data enrichment and filtering..." >> $LOGFILE

hive -hiveconf batchid=$batchid -f /home/acadgild/project/scripts/data_enrichment.hql

if [ ! -d "$VALIDDIR" ]
then
mkdir -p "$VALIDDIR"
fi

if [ ! -d "$INVALIDDIR" ]
then
mkdir -p "$INVALIDDIR"
fi

echo "Copying valid and invalid records in local file system..." >> $LOGFILE

hadoop fs -get /user/hive/warehouse/project.db/enriched_data/batchid=$batchid/status=pass/* $VALIDDIR
hadoop fs -get /user/hive/warehouse/project.db/enriched_data/batchid=$batchid/status=fail/* $INVALIDDIR

echo "Deleting older valid and invalid records from local file system..." >> $LOGFILE

find /home/acadgild/project/processed_dir/ -mtime +7 -exec rm {} \;

```

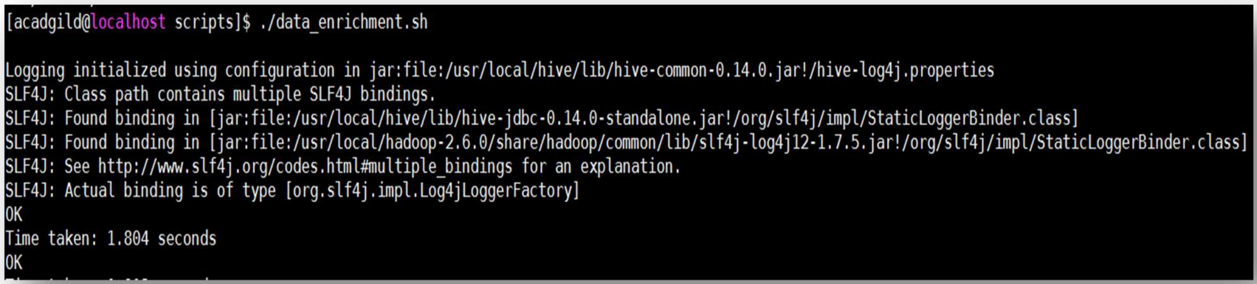


```
SET hive.auto.convert.join=false;
SET hive.exec.dynamic.partition.mode=nonstrict;

USE project;

CREATE TABLE IF NOT EXISTS enriched_data
(
user_id STRING,
song_id STRING,
artist_id STRING,
timestamp STRING,
start_ts STRING,
end_ts STRING,
geo_cd STRING,
station_id STRING,
song_end_type INT,
like INT,
dislike INT
)
PARTITIONED BY
(batchid INT,status STRING)
STORED as ORC;

INSERT OVERWRITE TABLE enriched_data
PARTITION (batchid,status)
SELECT
i.user_id,
i.song_id,
IF(i.artist_id is NULL OR i.artist_id='',sa.artist_id,i.artist_id) AS artist_id,
i.timestamp,
i.start_ts,
i.end_ts,
IF(i.geo_cd is NULL OR i.geo_cd='',sg.geo_cd,i.geo_cd) AS geo_cd,
i.station_id,
IF (i.song_end_type IS NULL,3,i.song_end_type) AS song_end_type,
IF (i.like IS NULL,0,i.like) AS like,
IF (i.dislike IS NULL,0,i.dislike) AS dislike,
i.batchid,
IF((i.like=1 AND i.dislike=1)
OR i.user_id IS NULL
OR i.song_id IS NULL
OR i.timestamp IS NULL
```



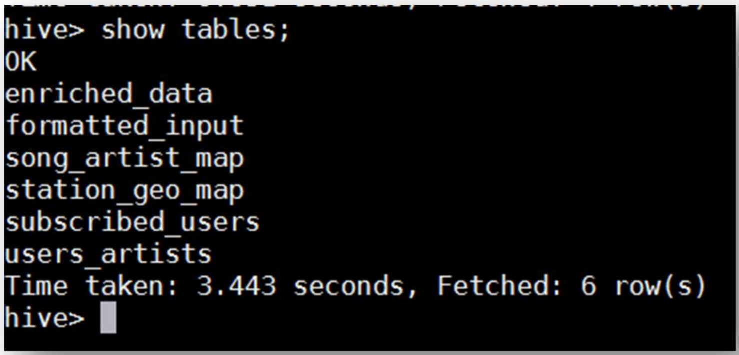
```
OR i.start_ts IS NULL
OR i.end_ts IS NULL
OR i.user_id=''
OR i.song_id=''
OR i.timestamp=''
OR i.start_ts=''
OR i.end_ts=''
OR sg.geo_cd=''
OR sg.geo_cd IS NULL
OR sa.artist_id IS NULL
OR sa.artist_id='','fail','pass') AS status
FROM formatted_input i
LEFT OUTER JOIN station_geo_map sg ON i.station_id = sg.station_id
LEFT OUTER JOIN song_artist_map sa ON i.song_id = sa.song_id
WHERE i.batchid=${hiveconf:batchid};
```

Running data enrichment script



```
[acadgild@localhost scripts]$ ./data_enrichment.sh

Logging initialized using configuration in jar:file:/usr/local/hive/lib/hive-common-0.14.0.jar!/hive-log4j.properties
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/hive-jdbc-0.14.0-standalone.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
OK
Time taken: 1.804 seconds
OK
```

HIVE tables:



```
hive> show tables;
OK
enriched_data
formatted_input
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 3.443 seconds, Fetched: 6 row(s)
hive>
```