

# ACADGILD

## BIG DATA HADOOP & SPARK TRAINING

### SESSION – 24

#### PROJECT

#### Music Data Analysis

#### TABLE OF CONTENTS

➤ Introduction & Data-set .....	1
➤ Look-Up Tables Files .....	3
➤ Steps to perform data analysis on the Music Data .....	5
➤ Step 1: Launch all necessary daemons .....	5
➤ Step 2: Start Job Scheduling .....	7
➤ Step 3: Populate Look-Up tables .....	8
➤ Step 4: Perform Data Formatting .....	12
➤ Step 5: Perform Data Enrichment and Cleaning .....	16
➤ Step 6: Perform Data Analysis .....	24
➤ Post Analysis .....	32

## **Introduction:**

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.

## **Data-set:**

### **Google Drive Link:**

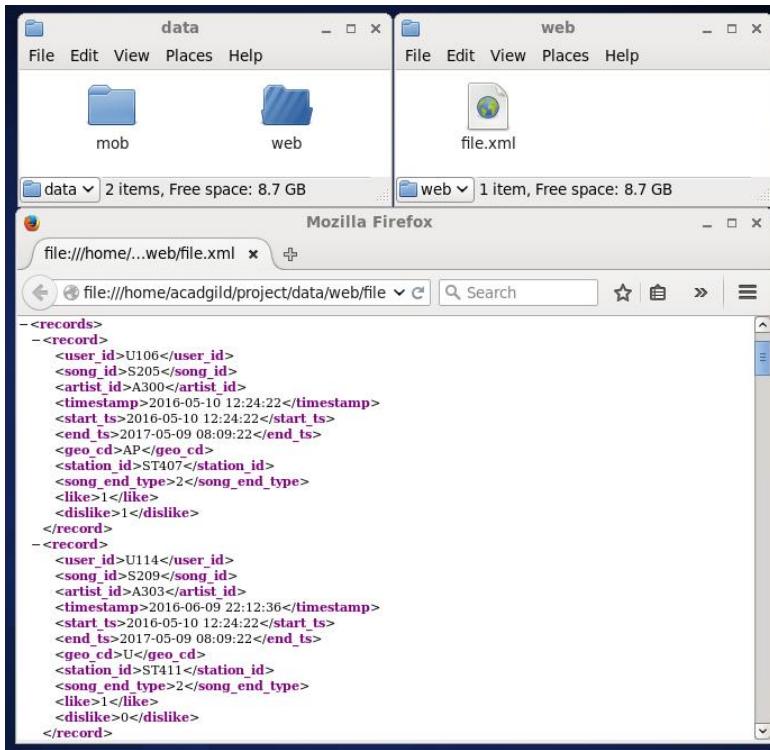
[https://drive.google.com/drive/folders/0B\\_P3pWagdlrrMjGVINsSUtbG8?usp=sharing](https://drive.google.com/drive/folders/0B_P3pWagdlrrMjGVINsSUtbG8?usp=sharing)

### **Data Description**

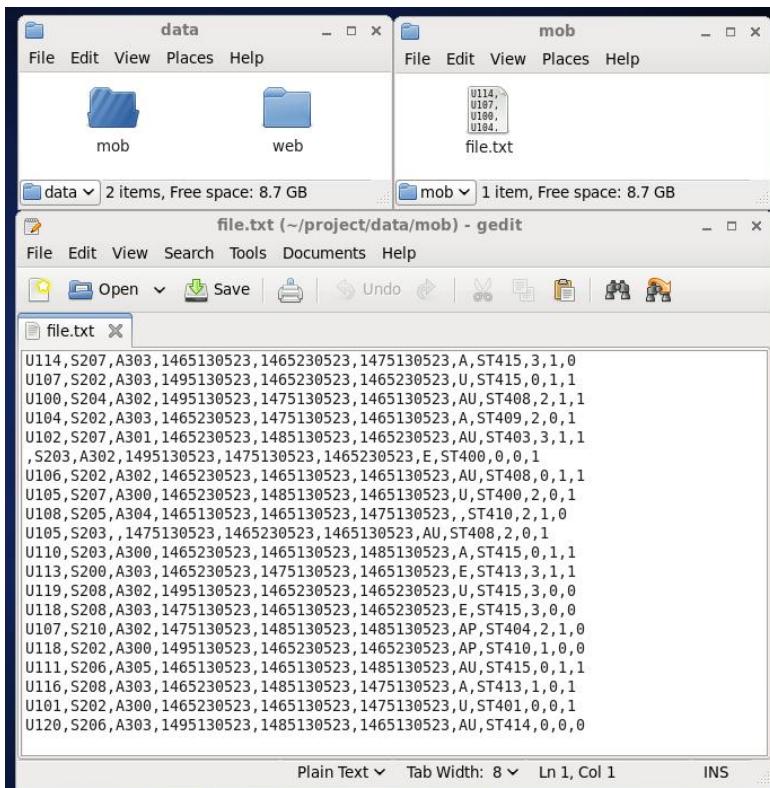
Column Name/Field Name	Column Description/Field Description
User_id	Unique identifier of every user
Song_id	Unique identifier of every song
Artist_id	Unique identifier of the lead artist of the song
Timestamp	Timestamp when the record was generated
Start_ts	Start timestamp when the song started to play
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 song 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

## Data Files:

Below is the data coming from web applications, that reside in /data/web and has **xml** format.



Below is a sample of the data coming from mobile applications, that reside in /data/mob and has **csv** format.



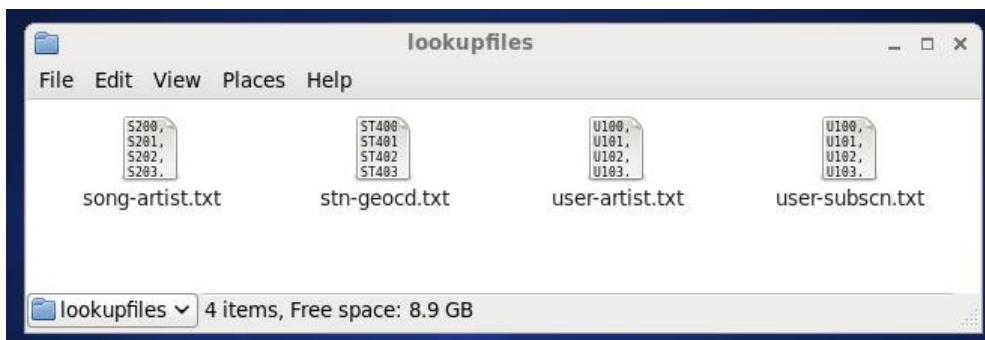
## Look-Up Tables Files:

There are some existing lookup tables present in NoSQL Databases that play an important role in data enrichment and analysis.

This data is present in lookup directory and loaded in HBase.

Table Name	Description
Station_Geo_Map	Contains mapping of a 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 alongwith royalty associated with each play of the song
User_Artist_Map	Contains an array of artist_id(s) followed by a user_id

Below are the data to be present in the lookup tables. There are 4 tables:



### song-artist

Columns: **song\_id, artist\_id**

The screenshot shows a terminal window with the title "song-artist.txt". The window displays a list of 10 entries, each consisting of a song ID followed by a comma and an artist ID. The entries are: S200,A300, S201,A301, S202,A302, S203,A303, S204,A304, S205,A301, S206,A302, S207,A303, S208,A304, and S209,A305.

song_id	artist_id
S200	A300
S201	A301
S202	A302
S203	A303
S204	A304
S205	A301
S206	A302
S207	A303
S208	A304
S209	A305

## stn-geocd

Columns: **station\_id, geo\_cd**

```
stn-geocd.txt X
ST400,A
ST401,AU
ST402,AP
ST403,J
ST404,E
ST405,A
ST406,AU
ST407,AP
ST408,E
ST409,E
ST410,A
ST411,A
ST412,AP
ST413,J
ST414,E|
```

## user-artist

Columns: **user\_id, artists\_array**

```
user-artist.txt X
U100,A300&A301&A302
U101,A301&A302
U102,A302
U103,A303&A301&A302
U104,A304&A301
U105,A305&A301&A302
U106,A301&A302
U107,A302
U108,A300&A303&A304
U109,A301&A303
U110,A302&A301
U111,A303&A301
U112,A304&A301
U113,A305&A302
U114,A300&A301&A302|
```

## user-subscn

Columns: **user\_id, subscn\_start\_dt, subscn\_end\_dt**

```
user-subscn.txt X
U100,1465230523,1465130523
U101,1465230523,1475130523
U102,1465230523,1475130523
U103,1465230523,1475130523
U104,1465230523,1475130523
U105,1465230523,1475130523
U106,1465230523,1485130523
U107,1465230523,1455130523
U108,1465230523,1465230623
U109,1465230523,1475130523
U110,1465230523,1475130523
U111,1465230523,1475130523
U112,1465230523,1475130523
U113,1465230523,1485130523
U114,1465230523,1468130523|
```

## **Steps to perform data analysis on the Music Data:**

**Step 1:** Launch all necessary daemons

**Step 2:** Start Job Scheduling (using Crontab)

**Step 3:** Populate Look-Up tables (i.e. Load all data to HBase)

**Step 4:** Perform Data Formatting (using Pig and Hive)

**Step 5:** Perform Data Enrichment and Cleaning (using Hive)

**Step 6:** Perform Data Analysis (using Spark)

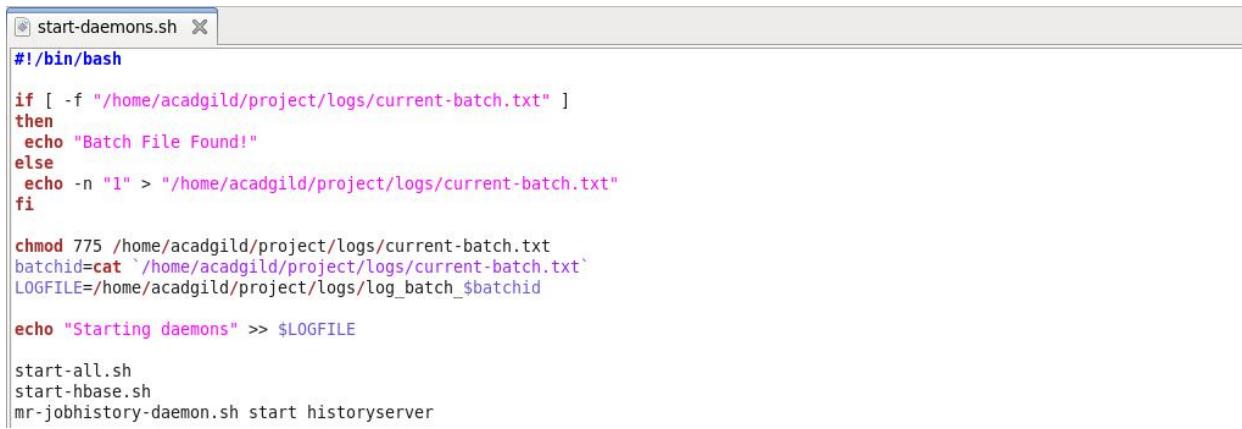
### **Step 1:**

Launch all necessary daemons

- Launch the Mysql Service (needed for Hive)
- Give permissions to scripts folder in project, so we are able to run scripts from the bash shell.
- Run the shell script `start-daemons.sh`

```
acadgild@localhost:~$ sudo service mysqld start
[sudo] password for acadgild:
Starting mysqld: [ OK ]
[acadgild@localhost ~]$ chmod 774 /home/acadgild/project/scripts/*
[acadgild@localhost ~]$
[acadgild@localhost ~]$ sh /home/acadgild/project/scripts/start-daemons.sh
/home/acadgild/project/logs/current-batch.txt: line 1: 1: command not found
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
17/10/06 00:41:11 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/hadoop-2.6.0/logs/hadoop-acadgild-namenode-localhost.localdomain.out
localhost: starting datanode, logging to /usr/local/hadoop-2.6.0/logs/hadoop-acadgild-datanode-localhost.localdomain.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: secondarynamenode running as process 3047. Stop it first.
17/10/06 00:41:29 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop-2.6.0/logs/yarn-acadgild-resourcemanager-localhost.localdomain.out
localhost: starting nodemanager, logging to /usr/local/hadoop-2.6.0/logs/yarn-acadgild-nodemanager-localhost.localdomain.out
starting master, logging to /usr/local/hbase/logs/hbase-acadgild-master-localhost.localdomain.out
historyserver running as process 3648. Stop it first.
[acadgild@localhost ~]$
```

In the shell script `start-daemons.sh` used above, we perform the following operations:



```
#!/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

start-all.sh
start-hbase.sh
mr-jobhistory-daemon.sh start historyserver
```

- Check if a file **current-batch.txt** has been created or not,  
If already created, print **Batch File Found!** else create the file and add 1 to it to signify batch 1.



- Give permissions to the file, so that we are able to modify it on the run.
- Get the batch id number from the batch file created above and create a **Log File** for the batch using the batch id. This will be **log\_batch\_1**.

Through out the course of the analysis process this log file will document the tasks that are performed for the Music Data Analysis.

- Add a log to the Log File signifying that the all necessary daemons have been started.



- Start the dfs, yarn, hbase and jobhistory daemons.

## Step 2:

### Start Job Scheduling

- Open the crontab file and insert the statement:

```
* */3 * * * /home/acadgild/project/scripts/wrapper.sh
```

Crontab is used for Job Scheduling. In the -e mode, Crontab schedules execution of commands by a regular user.

The statement above runs the wrapper.sh shell script every 3 hours.

The figure consists of three vertically stacked screenshots of a terminal window. The top screenshot shows the command `sudo crontab -e` being run, followed by a password prompt. The middle screenshot shows the crontab editor with the scheduled command `* */3 * * * /home/acadgild/project/scripts/wrapper.sh` entered, with the status `-- INSERT --` displayed. The bottom screenshot shows the command `sudo crontab -e` again, followed by a confirmation message `crontab: installing new crontab`.

In the shell script `wrapper.sh` used above, all the processes needed to perform analysis on the Music Data is called once every 3 hours thereby creating a new batch. This is the job scheduling.

The figure shows a code editor window titled "wrapper.sh" containing the following shell script:

```
#!/bin/bash
sh /home/acadgild/project/scripts/start-daemons.sh
sh /home/acadgild/project/scripts/populate-lookup.sh
sh /home/acadgild/project/scripts/dataformatting.sh
sh /home/acadgild/project/scripts/data_enrichment_filtering_schema.sh
sh /home/acadgild/project/scripts/data_enrichment.sh
sh /home/acadgild/project/scripts/data analysis.sh
```

## Step 3:

### Populate Look-Up tables

Below is the shell script [populate-lookup.sh](#) that is used to load the data for the lookup tables into HBase tables.

The following operations are performed:

- Get the batch id number from the batch file and get the **Log File** for the batch using the batch id. This will be **log\_batch\_1**
- Add logs to the Log File signifying that the lookup tables are being created and populated
- Create the HBase tables for the lookup data files: **song-artist**, **stn-geocd** and **user-subscn** with their column families
- For every lookup data file, read each line, extract the columns (comma separated) and add the data as rows to the corresponding HBase tables created above
- Run the hive script [user-artist.hql](#). This will populate a hive table with the data in the lookup data file **user-artist**. This is because this file has an array column that is difficult to populate in HBase.

```
# populate-lookup.sh
#!/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
```

- After the data in **user-artist** is loaded in the Hive Table **users-artists**, it is then saved as a text file as below (for data analysis using spark)

```

CREATE DATABASE IF NOT EXISTS project;
USE project;
CREATE TABLE users_artists
(
user_id STRING,
artists_array ARRAY<STRING>
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
COLLECTION ITEMS TERMINATED BY '&';
LOAD DATA LOCAL INPATH '/home/acadgild/project/lookupfiles/user-artist.txt'
OVERWRITE INTO TABLE users_artists;

INSERT OVERWRITE LOCAL DIRECTORY '/home/acadgild/project/exporteddata/userartists'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
SELECT user_id,artists FROM users_artists LATERAL VIEW explode(artists_array) a AS artists;

```

Below is a view of the execution of the above:

```

acadgild@localhost:~
File Edit View Search Terminal Help
[acadgild@localhost ~]$ sh /home/acadgild/project/scripts/populate-lookup.sh
2017-10-06 00:51:14,811 INFO [main] Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.98.14-hadoop2, r4e4aabb93b52f1b0fef6b66edd06ec8923014dec, Tue Aug 25 22:35:44 PDT 2015

create 'station-geo-map', 'geo'
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hbase/lib/slf4j-log4j12-1.6.4.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.
2017-10-06 00:51:19,344 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
0 row(s) in 2.9460 seconds

Hbase::Table - station-geo-map
2017-10-06 00:51:27,628 INFO [main] Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell: enter 'help<RETURN>' for list of supported commands.
OK
Time taken: 1.817 seconds
OK
Time taken: 0.051 seconds
OK
Time taken: 0.588 seconds
Loading data to table project.users_artists
Table project.users_artists stats: [numFiles=1, numRows=0, totalSize=240, rawDataSize=0]
OK
Time taken: 1.29 seconds
Query ID = acadgild_20171006010303_790650ac-df39-45ab-beab-e321b588edcb
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1507230696879_0001, Tracking URL = http://localhost:8088/proxy/application_1507230696879_0001/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job -kill job_1507230696879_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2017-10-06 01:04:16,293 Stage-1 map = 0%, reduce = 0%
2017-10-06 01:04:28,224 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.41 sec
MapReduce Total cumulative CPU time: 1 seconds 410 msec
Ended Job = job_1507230696879_0001
Copying data to local directory /home/acadgild/project/exporteddata/userartists
Copying data to local directory /home/acadgild/project/exporteddata/userartists
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 1.41 sec HDFS Read: 476 HDFS Write: 330 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 410 msec
OK
Time taken: 36.569 seconds
[acadgild@localhost ~]$
```

## Output of the above (HBase):

```
acadgild@localhost:~
```

```
File Edit View Search Terminal Help
[acadgild@localhost ~]$ hbase shell
2017-10-06 01:22:51,099 INFO [main] Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.98.14-hadoop2, r4e4aabb93b52f1b0fef6b66edd06ec8923014dec, Tue Aug 25 22:35:44 PDT 2015

hbase(main):001:0> list
TABLE
song-artist-map
station-geo-map
subscribed-users
3 row(s) in 2.1830 seconds

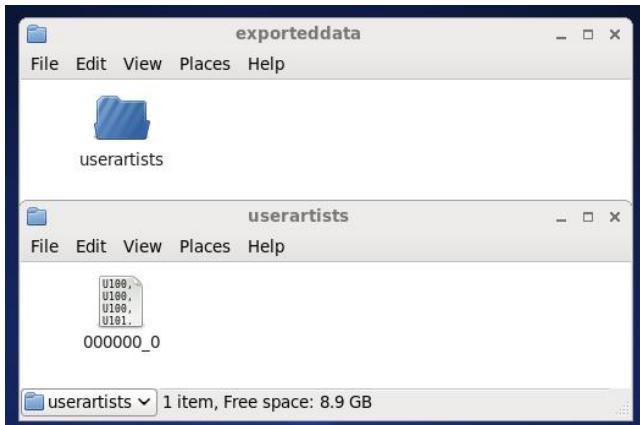
=> ["song-artist-map", "station-geo-map", "subscribed-users"]
hbase(main):002:0> scan 'song-artist-map'
ROW                                COLUMN+CELL
S200                               column=artist:artistid, timestamp=1507231503923, value=A300
S201                               column=artist:artistid, timestamp=1507231516620, value=A301
S202                               column=artist:artistid, timestamp=1507231529749, value=A302
S203                               column=artist:artistid, timestamp=1507231542525, value=A303
S204                               column=artist:artistid, timestamp=1507231555248, value=A304
S205                               column=artist:artistid, timestamp=1507231567875, value=A301
S206                               column=artist:artistid, timestamp=1507231580630, value=A302
S207                               column=artist:artistid, timestamp=1507231593034, value=A303
S208                               column=artist:artistid, timestamp=1507231605732, value=A304
S209                               column=artist:artistid, timestamp=1507231618736, value=A305
10 row(s) in 0.2150 seconds

hbase(main):003:0> scan 'station-geo-map'
ROW                                COLUMN+CELL
ST400                              column=geo:geo_cd, timestamp=1507231317923, value=A
ST401                              column=geo:geo_cd, timestamp=1507231330364, value=AU
ST402                              column=geo:geo_cd, timestamp=1507231342791, value=AP
ST403                              column=geo:geo_cd, timestamp=1507231354787, value=J
ST404                              column=geo:geo_cd, timestamp=1507231367658, value=E
ST405                              column=geo:geo_cd, timestamp=1507231380119, value=A
ST406                              column=geo:geo_cd, timestamp=1507231392585, value=AU
ST407                              column=geo:geo_cd, timestamp=1507231405210, value=AP
ST408                              column=geo:geo_cd, timestamp=1507231417589, value=E
ST409                              column=geo:geo_cd, timestamp=1507231429942, value=E
ST410                              column=geo:geo_cd, timestamp=1507231442135, value=A
ST411                              column=geo:geo_cd, timestamp=1507231454143, value=A
ST412                              column=geo:geo_cd, timestamp=1507231466386, value=AP
ST413                              column=geo:geo_cd, timestamp=1507231478812, value=J
ST414                              column=geo:geo_cd, timestamp=1507231491420, value=E
15 row(s) in 0.1420 seconds

hbase(main):004:0> scan 'subscribed-users'
ROW                                COLUMN+CELL
U100                               column=subscn:enddt, timestamp=1507231644509, value=1465130523
U100                               column=subscn:startdt, timestamp=1507231631720, value=1465230523
U101                               column=subscn:enddt, timestamp=1507231672165, value=1475130523
U101                               column=subscn:startdt, timestamp=1507231658717, value=1465230523
U102                               column=subscn:enddt, timestamp=1507231697657, value=1475130523
U102                               column=subscn:startdt, timestamp=1507231684928, value=1465230523
U103                               column=subscn:enddt, timestamp=1507231724152, value=1475130523
U103                               column=subscn:startdt, timestamp=1507231711008, value=1465230523
U104                               column=subscn:enddt, timestamp=1507231750013, value=1475130523
U104                               column=subscn:startdt, timestamp=1507231736885, value=1465230523
U105                               column=subscn:enddt, timestamp=1507231776615, value=1475130523
U105                               column=subscn:startdt, timestamp=1507231763146, value=1465230523
U106                               column=subscn:enddt, timestamp=1507231803327, value=1485130523
U106                               column=subscn:startdt, timestamp=1507231790177, value=1465230523
U107                               column=subscn:enddt, timestamp=1507231829462, value=1455130523
U107                               column=subscn:startdt, timestamp=1507231816627, value=1465230523
U108                               column=subscn:enddt, timestamp=1507231855404, value=1465230623
U108                               column=subscn:startdt, timestamp=1507231842490, value=1465230523
U109                               column=subscn:enddt, timestamp=1507231882001, value=1475130523
U109                               column=subscn:startdt, timestamp=1507231868908, value=1465230523
U110                               column=subscn:enddt, timestamp=1507231908613, value=1475130523
U110                               column=subscn:startdt, timestamp=1507231895355, value=1465230523
U111                               column=subscn:enddt, timestamp=1507231935200, value=1475130523
U111                               column=subscn:startdt, timestamp=1507231922281, value=1465230523
U112                               column=subscn:enddt, timestamp=1507231961593, value=1475130523
U112                               column=subscn:startdt, timestamp=1507231948389, value=1465230523
U113                               column=subscn:enddt, timestamp=1507231987722, value=1485130523
U113                               column=subscn:startdt, timestamp=1507231974658, value=1465230523
U114                               column=subscn:enddt, timestamp=1507232014165, value=1468130523
U114                               column=subscn:startdt, timestamp=1507232000831, value=1465230523
15 row(s) in 0.2770 seconds
```

Output of the above (Hive):

The data in the hive table (exploded) that was saved:



A screenshot of a text editor window titled "000000\_0". The window displays a large amount of text, which is a list of pairs separated by commas. The list starts with "U100,A300" and continues through many other entries, ending with "U114,A302".

```
U100,A300
U100,A301
U100,A302
U100,A302
U101,A301
U101,A302
U102,A302
U103,A303
U103,A301
U103,A302
U104,A304
U104,A301
U105,A305
U105,A301
U105,A302
U106,A301
U106,A302
U107,A302
U108,A300
U108,A303
U108,A304
U109,A301
U109,A303
U110,A302
U110,A301
U111,A303
U111,A301
U112,A304
U112,A301
U113,A305
U113,A302
U114,A300
U114,A301
U114,A302
```

## Step 4:

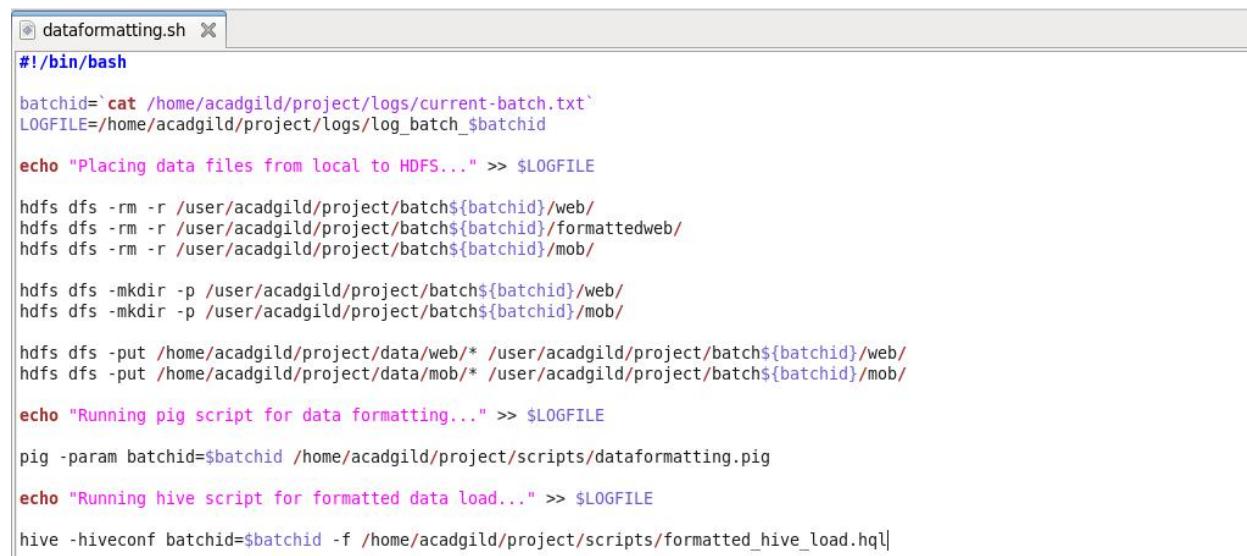
### Perform Data Formatting

Below is the shell script `dataformatting.sh` that is used to:

- Format the web xml data using **Pig** to a csv fomat and
- Load the 2 data files, **mob** and **web** (formatted by Pig), to a Hive Table for data enrichment

The following operations are performed:

- Get the batch id number from the batch file and get the **Log File** for the batch using the batch id. This will be **log\_batch\_1**
- Add logs to the Log File signifying that the data is placed in the HDFS and the running of the Pig and Hive scripts for data formatting and loading respectively.
- Delete, if they exist, folders for the mob, web and formattedweb. This is done in-case any old data remains because of execution failure.
- Create the above folders web and mob that were deleted above and move the data from the Local FS to the HDFS. The formattedweb folder is created in the Pig Script.
- Run the pig script `dataformatting.pig`. This will format the web data (stored in the web folder in the HDFS) in xml format to csv format and store it in the HDFS in the folder formattedweb.
- Run the hive script `formatted_hive_load.hql`. This will load the data in the mob folder and formattedweb folder in the HDFS to a table `formatted_input` in Hive which will be used for data enrichment later.



```
#!/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

hdfs dfs -rm -r /user/acadgild/project/batch${batchid}/web/
hdfs dfs -rm -r /user/acadgild/project/batch${batchid}/formattedweb/
hdfs dfs -rm -r /user/acadgild/project/batch${batchid}/mob/

hdfs dfs -mkdir -p /user/acadgild/project/batch${batchid}/web/
hdfs dfs -mkdir -p /user/acadgild/project/batch${batchid}/mob/

hdfs dfs -put /home/acadgild/project/data/web/* /user/acadgild/project/batch${batchid}/web/
hdfs dfs -put /home/acadgild/project/data/mob/* /user/acadgild/project/batch${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
```

## **dataformatting.pig**

Stores the formatted data to a folder in the HDFS called formattedweb.

```
dataformatting.pig X
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,
    ToUnixTimeToDate(TRIM(XPath(x, 'record/timestamp')), 'yyyy-MM-dd HH:mm:ss') AS timestamp,
    ToUnixTimeToDate(TRIM(XPath(x, 'record/start_ts')), 'yyyy-MM-dd HH:mm:ss') AS start_ts,
    ToUnixTimeToDate(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(',');
```

## **formatted\_hive\_load.hql**

Combines the data from mob and formattedweb to make one data-set and stores it partitioned by batchid.

```
formatted_hive_load.hql X
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});
```

Below is a view of the execution of the above:

```
acadgild@localhost:~ - □
File Edit View Search Terminal Help
[acadgild@localhost ~]$ sh /home/acadgild/project/scripts/dataformatting.sh
17/10/06 01:12:12 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
rm: `/user/acadgild/project/batch1/web/': No such file or directory
17/10/06 01:12:15 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
rm: `/user/acadgild/project/batch1/formattedweb/': No such file or directory
17/10/06 01:12:19 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
rm: `/user/acadgild/project/batch1/mob/': No such file or directory
2017-10-06 01:12:35,446 INFO [main] pig.ExecTypeProvider: Trying ExecType : LOCAL
2017-10-06 01:12:35,453 INFO [main] pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
2017-10-06 01:12:35,453 INFO [main] pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2017-10-06 01:12:35,596 [main] INFO org.apache.pig.Main - Apache Pig version 0.14.0 (r1640057) compiled Nov 16 2014, 18:02:00
```

Below is a look into the HDFS:

```
acadgild@localhost:~
```

```
File Edit View Search Terminal Help
[acadgild@localhost ~]$ hdfs dfs -ls
17/10/06 01:11:56 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 2015-11-20 11:46 Pictures
drwxr-xr-x - acadgild supergroup          0 2015-11-17 02:03 oozie-acad
drwxr-xr-x - acadgild supergroup          0 2015-11-17 02:00 share
[acadgild@localhost ~]$
[acadgild@localhost ~]$ hdfs dfs -ls
17/10/06 01:16:28 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 4 items
drwxr-xr-x - acadgild supergroup          0 2015-11-20 11:46 Pictures
drwxr-xr-x - acadgild supergroup          0 2015-11-17 02:03 oozie-acad
drwxr-xr-x - acadgild supergroup          0 2017-10-06 01:12 project
drwxr-xr-x - acadgild supergroup          0 2015-11-17 02:00 share
[acadgild@localhost ~]$
[acadgild@localhost ~]$ hdfs dfs -ls project
17/10/06 01:16:37 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 1 items
drwxr-xr-x - acadgild supergroup          0 2017-10-06 01:12 project/batch1
[acadgild@localhost ~]$
[acadgild@localhost ~]$ hdfs dfs -ls project/batch1
17/10/06 01:16:45 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 2017-10-06 01:13 project/batch1/formattedweb
drwxr-xr-x - acadgild supergroup          0 2017-10-06 01:12 project/batch1/mob
drwxr-xr-x - acadgild supergroup          0 2017-10-06 01:12 project/batch1/web
[acadgild@localhost ~]$
[acadgild@localhost ~]$ hdfs dfs -ls project/batch1/mob
17/10/06 01:18:33 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 1 items
-rw-r--r-- 1 acadgild supergroup      1239 2017-10-06 01:12 project/batch1/mob/file.txt
[acadgild@localhost ~]$
[acadgild@localhost ~]$ hdfs dfs -ls project/batch1/web
17/10/06 01:18:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 1 items
-rw-r--r-- 1 acadgild supergroup      6716 2017-10-06 01:12 project/batch1/web/file.xml
[acadgild@localhost ~]$
[acadgild@localhost ~]$ hdfs dfs -ls project/batch1/formattedweb
17/10/06 01:20:30 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r-- 1 acadgild supergroup      0 2017-10-06 01:13 project/batch1/formattedweb/_SUCCESS
-rw-r--r-- 1 acadgild supergroup      1236 2017-10-06 01:13 project/batch1/formattedweb/part-m-00000
```

Below is a look at the data in the table **formatted\_input**:

```
acadgild@localhost:~
```

```
File Edit View Search Terminal Help
[acadgild@localhost ~]$ hive

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]
hive> SHOW DATABASES;
OK
b1
default
project
Time taken: 0.033 seconds, Fetched: 3 row(s)
```

```

hive> select * from formatted_input;
OK
U114 S207 A303 1465130523 1465230523 1475130523 A ST415 3 1 0 1
U107 S202 A303 1495130523 1465230523 1465130523 U ST415 0 1 1 1
U100 S204 A302 1495130523 1475130523 1465130523 AU ST408 2 1 1 1
U104 S202 A303 1465230523 1475130523 1465130523 A ST409 2 0 1 1
U102 S207 A301 1465230523 1485130523 1465230523 AU ST403 3 1 1 1
S203 A302 1495130523 1475130523 1465230523 E ST400 0 0 1 1
U106 S202 A302 1465230523 1465130523 1465130523 AU ST408 0 1 1 1
U105 S207 A300 1465230523 1485130523 1465130523 U ST400 2 0 1 1
U108 S205 A304 1465130523 1465130523 1475130523 ST410 2 1 0 1
U105 S203 1475130523 1465230523 1465130523 AU ST408 2 0 1 1
U110 S203 A300 1465230523 1465130523 1485130523 A ST415 0 1 1 1
U113 S200 A303 1465230523 1475130523 1465130523 E ST413 3 1 1 1
U119 S208 A302 1495130523 1465230523 1465230523 U ST415 3 0 0 1
U118 S208 A303 1475130523 1465130523 1465230523 E ST415 3 0 0 1
U107 S210 A302 1475130523 1485130523 1485130523 AP ST404 2 1 0 1
U118 S202 A300 1495130523 1465230523 1465230523 AP ST410 1 0 0 1
U111 S206 A305 1465130523 1465130523 1485130523 AU ST415 0 1 1 1
U116 S208 A303 1465230523 1485130523 1475130523 A ST413 1 0 1 1
U101 S202 A300 1465230523 1465130523 1475130523 U ST401 0 0 1 1
U120 S206 A303 1495130523 1485130523 1465130523 AU ST414 0 0 0 1
U106 S205 A300 1462863262 1462863262 1494297562 AP ST407 2 1 1 1
U114 S209 A303 1465490556 1462863262 1494297562 U ST411 2 1 0 1
U113 S203 A304 1465490556 1465490556 1462863262 U ST405 0 0 1 1
U108 S200 A302 1468094889 1462863262 1468094889 U ST414 0 0 1 1
U102 S203 A305 1465490556 1465490556 1494297562 U ST404 2 0 0 1
S208 A300 1465490556 1494297562 1465490556 U ST411 1 0 1 1
U115 S200 A300 1465490556 1494297562 1465490556 AU ST404 3 0 0 1
U111 S204 A300 1465490556 1468094889 1468094889 U ST410 3 1 1 1
U120 S201 A300 1494297562 1465490556 1468094889 ST410 3 0 1 1
U113 S203 1465490556 1465490556 1465490556 A ST402 1 1 0 1
U109 S203 A304 1462863262 1494297562 1468094889 E ST405 1 1 1 1
U110 S202 A303 1494297562 1494297562 1468094889 AU ST402 2 1 0 1
U100 S200 A301 1494297562 1494297562 1494297562 AP ST410 3 1 1 1
U101 S208 A300 1462863262 1468094889 1462863262 E ST408 0 1 1 1
U106 S206 A300 1494297562 1465490556 1462863262 A ST405 3 1 0 1
U107 S202 A304 1494297562 1468094889 1462863262 U ST409 0 0 0 1
U103 S204 A300 1468094889 1494297562 1465490556 AU ST411 2 1 0 1
U103 S202 A300 1465490556 1465490556 1465490556 A ST415 2 1 1 1
U113 S203 A303 1462863262 1468094889 1494297562 U ST408 2 0 0 1
U113 S204 A301 1494297562 1494297562 1465490556 E ST415 3 0 1 1
Time taken: 0.089 seconds, Fetched: 40 row(s)

hive> SHOW TABLES;
OK
formatted_input
users_artists
Time taken: 0.052 seconds, Fetched: 2 row(s)

```

Below is a look at the data in the table **formatted\_input** in the hive warehouse:

```

acadgild@localhost:~$ hdfs dfs -ls /user/hive/warehouse/project.db/formatted_input/batchid=1
17/10/06 01:31:20 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r-- 1 acadgild supergroup 1239 2017-10-06 01:12 /user/hive/warehouse/project.db/formatted_input/batchid=1/file.txt
-rw-r--r-- 1 acadgild supergroup 1236 2017-10-06 01:13 /user/hive/warehouse/project.db/formatted_input/batchid=1/part-m-00000

```

## Step 5:

### Perform Data Enrichment and Cleaning

The data enrichment is carried out in two steps:

- Create lookup tables in Hive and import the data from the HBase lookup tables to them. This is done by shell script `data_enrichment_filtering_schema.sh`
- Perform the data enrichment to the data in `formatted_input` using the lookup tables. This is done by shell script `data_enrichment.sh`

#### 1) `data_enrichment_filtering_schema.sh`

Below is the shell script `data_enrichment_filtering_schema.sh` where the following operations are performed:

- Get the batch id number from the batch file and get the **Log File** for the batch using the batch id. This will be `log_batch_1`
- Add logs to the Log File signifying that the Hive lookup tables are created from the HBase lookup tables.
- Run the hive script `create_hive_hbase_lookup.hql`. This will create the lookup tables in Hive and import the data from the HBase lookup tables to the Hive lookup 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
```

#### `create_hive_hbase_lookup.hql`

Create Hive lookup tables and save lookup table **subscribed\_users** to Local FS.

```
create_hive_hbase_lookup.hql
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");

INSERT OVERWRITE LOCAL DIRECTORY '/home/acadgild/project/exporteddata/subscribeduser'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
SELECT * FROM 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");

```

Below is a view of the execution of the above:

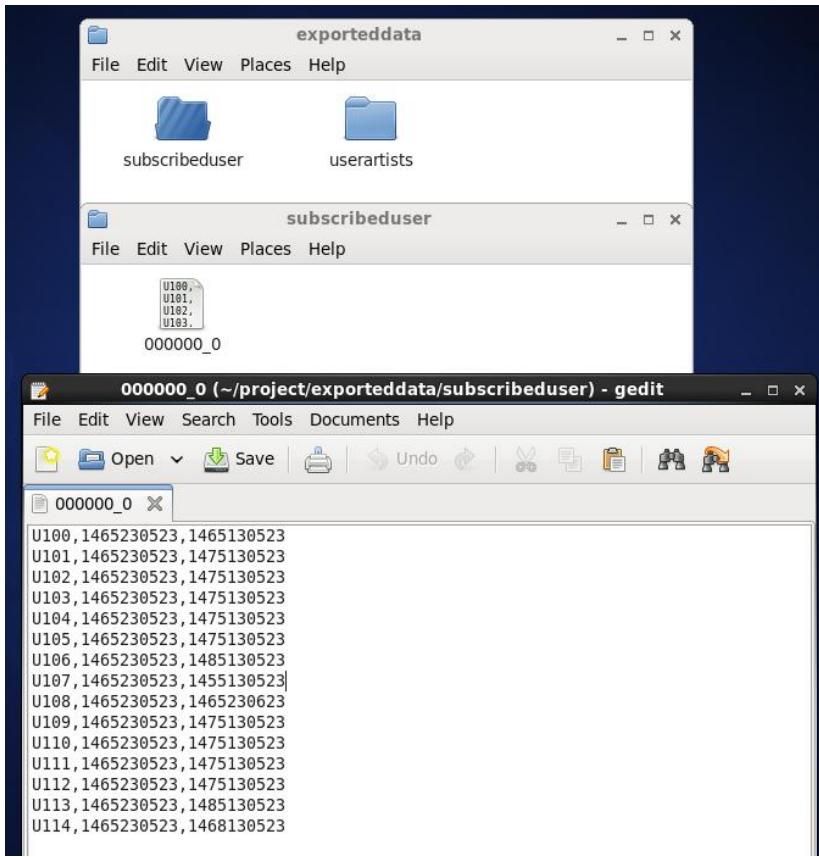
```

acadgild@localhost:~
File Edit View Search Terminal Help
[acadgild@localhost ~]$ sh /home/acadgild/project/scripts/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: 1.466 seconds
OK
Time taken: 2.198 seconds
OK
Time taken: 0.297 seconds
Query ID = acadgild_20171006013838_b459e5fd-ce6e-44a4-b846-3a962417819a
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1507230696879_0003, Tracking URL = http://localhost:8088/proxy/application_1507230696879_0003/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job -kill job_1507230696879_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2017-10-06 01:38:18,978 Stage-1 map = 0%, reduce = 0%
2017-10-06 01:38:29,735 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.35 sec
MapReduce Total cumulative CPU time: 2 seconds 350 msec
Ended Job = job_1507230696879_0003
Copying data to local directory /home/acadgild/project/exporteddata/subscribeduser
Copying data to local directory /home/acadgild/project/exporteddata/subscribeduser
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 2.35 sec HDFS Read: 276 HDFS Write: 405 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 350 msec
OK
Time taken: 28.021 seconds
OK
Time taken: 0.208 seconds

```

## Output of the saved table subscribed\_users in the Local FS:



## Output in Hive:

The tables were created and populated as intended.

```
acadgild@localhost:~$ File Edit View Search Terminal Help
hive> SHOW DATABASES;
OK
b1
default
project
Time taken: 0.027 seconds, Fetched: 3 row(s)
hive> USE project;
OK
Time taken: 0.025 seconds
hive> SHOW TABLES;
OK
formatted_input
users_artists
Time taken: 0.052 seconds, Fetched: 2 row(s)
hive> SHOW TABLES;
OK
formatted_input
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 0.037 seconds, Fetched: 5 row(s)
```

```
hive> SHOW TABLES;
OK
formatted_input
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 0.037 seconds, Fetched: 5 row(s)
hive> select * from song_artist_map;
OK
S200    A300
S201    A301
S202    A302
S203    A303
S204    A304
S205    A301
S206    A302
S207    A303
S208    A304
S209    A305
Time taken: 0.292 seconds, Fetched: 10 row(s)
hive> select * from station_geo_map;
OK
ST400    A
ST401    AU
ST402    AP
ST403    J
ST404    E
ST405    A
ST406    AU
ST407    AP
ST408    E
ST409    E
ST410    A
ST411    A
ST412    AP
ST413    J
ST414    E
Time taken: 0.194 seconds, Fetched: 15 row(s)
hive> select * from subscribed_users;
OK
U100    1465230523    1465130523
U101    1465230523    1475130523
U102    1465230523    1475130523
U103    1465230523    1475130523
U104    1465230523    1475130523
U105    1465230523    1475130523
U106    1465230523    1485130523
U107    1465230523    1455130523
U108    1465230523    1465230623
U109    1465230523    1475130523
U110    1465230523    1475130523
U111    1465230523    1475130523
U112    1465230523    1475130523
U113    1465230523    1485130523
U114    1465230523    1468130523
Time taken: 0.26 seconds, Fetched: 15 row(s)
```

## 2) data\_enrichment.sh

Below is the shell script `data_enrichment.sh` where the following operations are performed:

- Get the batch id number from the batch file and get the **Log File** for the batch using the batch id. This will be **log\_batch\_1**.
- Add logs to the Log File signifying that the data enrichment has begun.
- Run the hive script `data_enrichment.hql`. This will create a Hive table **enriched\_data** that will hold the data that is enriched and partitioned based on given rules as pass or fail (status) and batchid.
- Add logs to the Log File signifying that the valid and invalid outputs are being recorded in their respective folders.
- Copy the data from the pass and fail folders (valid & invalid) in the Hive warehouse to the Local FS.

```
#!/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

hdfs dfs -get /user/hive/warehouse/project.db/enriched_data/batchid=$batchid/status=pass/* $VALIDDIR
hdfs dfs -get /user/hive/warehouse/project.db/enriched_data/batchid=$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 {} \;
```

## data\_enrichment.hql

### 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.

NULL or absent field	Look up field	Look up table (Table from which record can be updated)
Geo_cd	Station_id	Station_Geo_Map
Artist_id	Song_id	Song_Artist_Map

For the data enrichment, a table **enriched\_data** is created and the table is overwritten with the result of the below operations:

- The data in the **formatted\_input** table is joined with the lookup tables **station\_geo\_map** and **song\_artist\_map** to fill in the data gaps that can be obtained by said tables.  
*and*
- The same data is then filtered by the rules given above and partitioned by status (pass or fail) & batchid.

The data of the enriched\_data table is then stored in a folder in the Local FS.

```
data_enrichment.hql X
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,
sa.artist_id,
i.timestamp,
i.start_ts,
i.end_ts,
sg.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,
IFI((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.geo_cd IS NULL
OR i.user_id=''
OR i.song_id=''
OR i.timestamp=''
OR i.start_ts=''
OR i.end_ts=''
OR i.geo_cd=''
OR sg.geo_cd IS NULL
OR sg.geo_cd=''
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};
```

```

INSERT OVERWRITE LOCAL DIRECTORY '/home/acadgild/project/exporteddata/enricheddata'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
SELECT * FROM enriched_data;

```

## Output in Hive :

```

acadgild@localhost:~
File Edit View Search Terminal Help
hive> SHOW TABLES;
OK
enriched_data
formatted_input
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 0.035 seconds, Fetched: 6 row(s)
hive> SELECT * FROM enriched_data;
OK
U113      S200      A300      1465230523      1475130523      J      ST413      3      1      1      1      1      fail
U100      S200      A300      1494297562      1494297562      A      ST410      3      1      1      1      1      fail
U120      S201      A301      1494297562      1465490556      1468094889      A      ST410      3      0      1      1      1      fail
U107      S202      A302      1495130523      1465230523      NULL      ST415      0      1      1      1      1      fail
U103      S202      A302      1465490556      1465490556      1465490556      NULL      ST415      2      1      1      1      1      fail
U106      S202      A302      1465230523      1465130523      1465130523      E      ST408      0      1      1      1      1      fail
U109      S203      A303      1462863262      1494297562      1468094889      A      ST405      1      1      1      1      1      fail
U203      A303      1495130523      1475130523      1465230523      A      ST400      0      0      1      1      1      fail
U110      S203      A303      1465230523      1485130523      1465130523      NULL      ST415      0      1      1      1      1      fail
U111      S204      A304      1465490556      1465490556      1468094889      A      ST410      3      1      1      1      1      fail
U113      S204      A304      1494297562      1494297562      1465490556      NULL      ST415      3      0      1      1      1      fail
U100      S204      A304      1495130523      1475130523      1465130523      E      ST408      2      1      1      1      1      fail
U106      S205      A301      1462863262      1462863262      1494297562      AP      ST407      2      1      1      1      1      fail
U108      S205      A301      1465130523      1475130523      1465130523      A      ST410      2      1      0      1      1      fail
U111      S206      A302      1465130523      1465130523      1485130523      NULL      ST415      0      1      1      1      1      fail
U114      S207      A303      1465130523      1465230523      1475130523      NULL      ST415      3      1      0      1      1      fail
U102      S207      A303      1465230523      1485130523      1465230523      J      ST403      3      1      1      1      1      fail
U208      S208      A304      1465490556      1494297562      1465490556      A      ST411      1      0      1      1      1      fail
U118      S208      A304      1475130523      1465130523      1465230523      NULL      ST415      3      0      0      0      1      fail
U119      S208      A304      1495130523      1465230523      1465230523      NULL      ST415      3      0      0      0      1      fail
U101      S208      A304      1462863262      1468094889      1462863262      E      ST408      0      1      1      1      1      fail
U107      S210      NULL      1475130523      1485130523      1485130523      E      ST404      2      1      0      1      1      fail
U115      S200      A300      1465490556      1494297562      1465490556      E      ST404      3      0      0      0      1      pass
U108      S200      A300      1468094889      1462863262      1468094889      E      ST414      0      0      1      1      1      pass
U107      S202      A302      1494297562      1468094889      1462863262      E      ST409      0      0      0      0      1      pass
U101      S202      A302      1465230523      1465130523      1475130523      AU      ST401      0      0      1      1      1      pass
U110      S202      A302      1494297562      1468094889      1468094889      AP      ST402      2      1      0      1      1      pass
U118      S202      A302      1495130523      1465230523      1465230523      A      ST410      1      0      0      0      1      pass
U104      S202      A302      1465230523      1475130523      1465130523      E      ST409      2      0      1      1      1      pass
U102      S203      A303      1465490556      1465490556      1494297562      E      ST404      2      0      0      0      1      pass
U113      S203      A303      1465490556      1465490556      1462863262      A      ST405      0      0      1      1      1      pass
U113      S203      A303      1462863262      1468094889      1494297562      E      ST408      2      0      0      0      1      pass
U105      S203      A303      1475130523      1465230523      1465130523      E      ST408      2      0      1      1      1      pass
U113      S203      A303      1465490556      1465490556      1465490556      AP      ST402      1      1      0      1      1      pass
U103      S204      A304      1468094889      1494297562      1465490556      A      ST411      2      1      0      1      1      pass
U106      S206      A302      1494297562      1465490556      1462863262      A      ST405      3      1      0      1      1      pass
U120      S206      A302      1495130523      1485130523      1465130523      E      ST414      0      0      0      0      1      pass
U105      S207      A303      1465230523      1485130523      1465130523      A      ST400      2      0      1      1      1      pass
U116      S208      A304      1465230523      1485130523      1475130523      J      ST413      1      0      1      1      1      pass
U114      S209      A305      1465490556      1462863262      1494297562      A      ST411      2      1      0      0      1      pass
Time taken: 0.076 seconds. Fetched: 40 row(s)

```

## Output in Hive Warehouse:

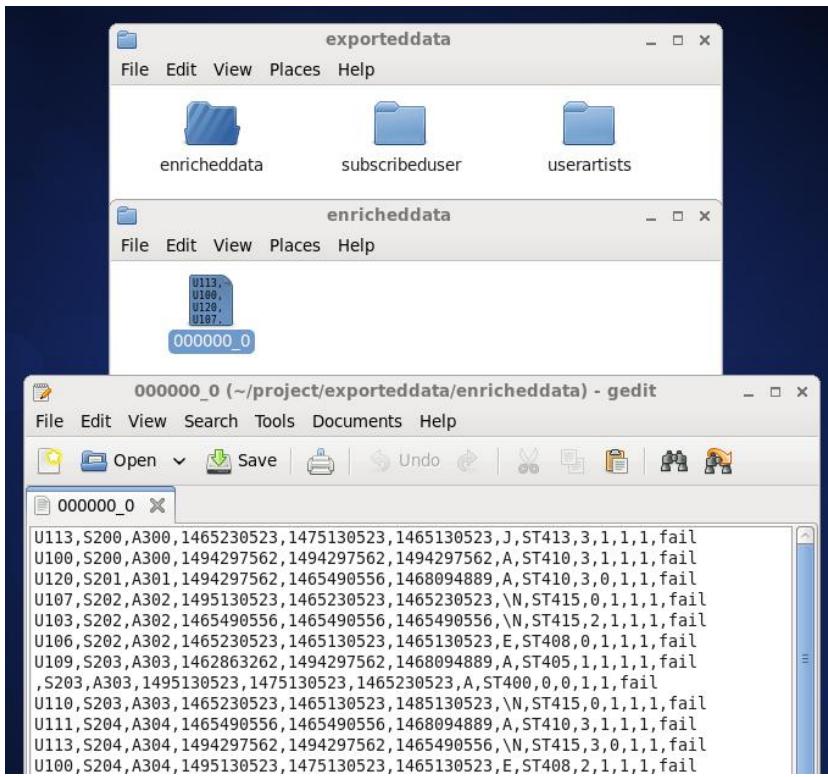
Below is a view of the enriched\_data in the Hive warehouse, partitioned by status and batchid

```

[acadgild@localhost ~]$ hdfs dfs -ls /user/hive/warehouse/project.db/enriched_data/batchid=1
17/10/06 01:55:06 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
drwxr-xr-x  - acadgild supergroup          0 2017-10-06 01:51 /user/hive/warehouse/project.db/enriched_data/batchid=1/status
=fail
drwxr-xr-x  - acadgild supergroup          0 2017-10-06 01:51 /user/hive/warehouse/project.db/enriched_data/batchid=1/status
=pass
[acadgild@localhost ~]$

```

## Output saved in the Local FS:

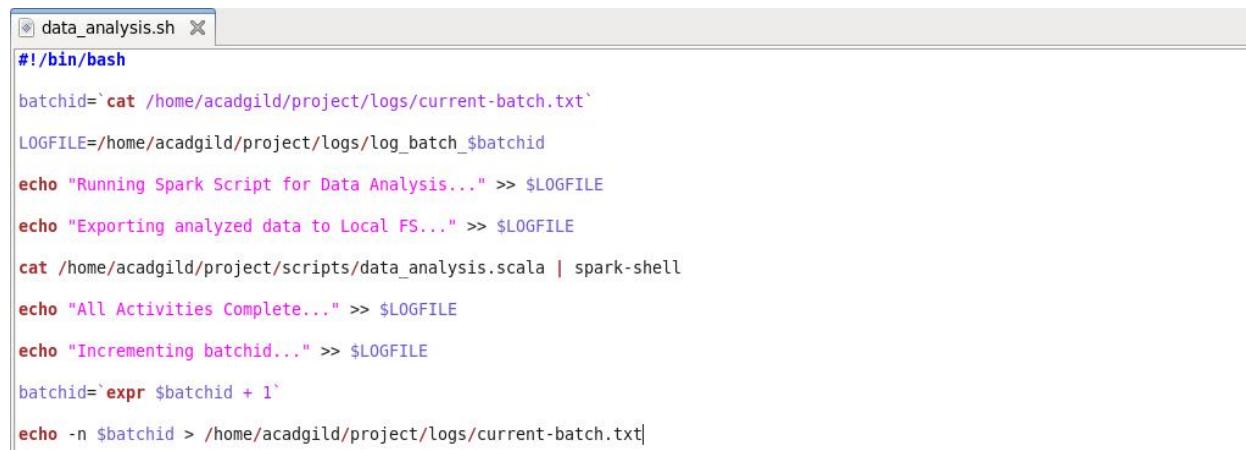


## Step 6:

### Perform Data Analysis

Below is the shell script `data_analysis.sh` where the following operations are performed:

- Get the batch id number from the batch file and get the **Log File** for the batch using the batch id. This will be **log\_batch\_1**
- Add logs to the Log File signifying that the data analysis is being performed using Spark and that the result is being exported to the Local FS.
- Run the spark script `data_analysis.scala`. This will perform the data analysis required in the problem statement given and save the result to the Local FS.
- Add logs to the Log File signifying that the data analysis has completed and that the batch is being incremented. Here from 1 to 2
- Get batchid number from batch file and increment the batchid by 1.



```
#!/bin/bash
batchid=`cat /home/acadgild/project/logs/current-batch.txt`
LOGFILE=/home/acadgild/project/logs/log_batch_$batchid
echo "Running Spark Script for Data Analysis..." >> $LOGFILE
echo "Exporting analyzed data to Local FS..." >> $LOGFILE
cat /home/acadgild/project/scripts/data_analysis.scala | spark-shell
echo "All Activities Complete..." >> $LOGFILE
echo "Incrementing batchid..." >> $LOGFILE
batchid=`expr $batchid + 1`
echo -n $batchid > /home/acadgild/project/logs/current-batch.txt
```

Below is the file **data\_analysis.scala** that will perform the data analysis for the given problem statements on the `enriched_data` that was saved to the Local FS.

#### Initialization:

- Import Row, DataFrame, Structure type and function dependencies needed to perform analysis.
- Get the batchid from the batch file and store it in the variable **batid**
- Get the data that was exported and saved in the Local FS from the steps above i.e. `enriched_data`, `subscribed_user` and `user_artists` and perform the foll. on each of them:

- Create the schema for the data
- Create a DataFrame from the schema and data
- Create a temporary table from the DataFrame created

```

data_analysis.scala

import org.apache.spark.sql.Row
import org.apache.spark.sql.DataFrame
import org.apache.spark.sql.types.{StructType, StructField, StringType, NumericType, IntegerType, ArrayType}

import org.apache.spark.sql.functions._

val batid = sc.textFile("/home/acadgild/project/logs/current-batch.txt").map(x => x.toInt).toDF().first.getInt(0)

//Music Data
val data = sc.textFile("/home/acadgild/project/exporteddata/enricheddata/000000_0")

val MDSchemaString =
"user_id:string,song_id:string,artist_id:string,timestamp:string,start_ts:string,end_ts:string,geo_cd:string,station_id:string"

val MDdataSchema = StructType(MDSchemaString.split(",").map(fieldInfo => StructField(fieldInfo.split(":")(0), if (fieldInfo.split(":")(1).equals("string")) StringType else IntegerType, true)))

val MDRowRDD = data.map(_.split(",")).map(r => Row(r(0), r(1), r(2), r(3), r(4), r(5), r(6), r(7), r(8).toInt, r(9).toInt, r(10).toInt, r(11).toInt, r(12)))

val MusicDataDF = spark.createDataFrame(MDRowRDD, MDdataSchema)

MusicDataDF.registerTempTable("Music_Data")

//Subscribed Users
val data = sc.textFile("/home/acadgild/project/exporteddata/subscribeduser/000000_0")

val SUSchemaString = "user_id:string,start_dt:string,end_dt:string"

val SUdataSchema = StructType(SUSchemaString.split(",").map(fieldInfo => StructField(fieldInfo.split(":")(0), if (fieldInfo.split(":")(1).equals("string")) StringType else IntegerType, true)))

val SUrowRDD = data.map(_.split(",")).map(r => Row(r(0), r(1), r(2)))

val SubscribedUsersDF = spark.createDataFrame(SUrowRDD, SUdataSchema)

SubscribedUsersDF.registerTempTable("Music_SubscribedUsers")

//User Artists
val data = sc.textFile("/home/acadgild/project/exporteddata/userartists/000000_0")

val UASchemaString = "user_id:string,artists:string"

val UAdataSchema = StructType(UASchemaString.split(",").map(fieldInfo => StructField(fieldInfo.split(":")(0), if (fieldInfo.split(":")(1).equals("string")) StringType else IntegerType, true)))

val UARowRDD = data.map(_.split(",")).map(r => Row(r(0), r(1)))

val UserArtistsDF = spark.createDataFrame(UARowRDD, UAdataSchema)

UserArtistsDF.registerTempTable("Music_UserArtists")

val Top10Stations = spark.sql(s"SELECT station_id, COUNT(DISTINCT song_id) AS total_distinct_songs_played, COUNT(DISTINCT user_id) AS distinct_user_count, batchid FROM Music_Data WHERE status='pass' AND batchid=$batid AND like=1 GROUP BY station_id,batchid ORDER BY total_distinct_songs_played DESC LIMIT 10");

Top10Stations.rdd.saveAsTextFile("/home/acadgild/project/output/top_10_stations")

val users_behavior = spark.sql(s"SELECT CASE WHEN (subusers.user_id IS NULL OR CAST(music.timestamp AS DECIMAL(20,0)) > CAST(subusers.end_dt AS DECIMAL(20,0))) THEN 'UNSUBSCRIBED' WHEN (subusers.user_id IS NOT NULL AND CAST(music.timestamp AS DECIMAL(20,0)) <= CAST(subusers.end_dt AS DECIMAL(20,0))) THEN 'SUBSCRIBED' END AS user_type, SUM(ABS(CAST(music.end_ts AS DECIMAL(20,0))-CAST(music.start_ts AS DECIMAL(20,0)))) AS duration, batchid FROM Music_Data music LEFT OUTER JOIN Music_SubscribedUsers subusers ON music.user_id=subusers.user_id WHERE music.status='pass' AND music.batchid=$batid GROUP BY CASE WHEN (subusers.user_id IS NULL OR CAST(music.timestamp AS DECIMAL(20,0)) > CAST(subusers.end_dt AS DECIMAL(20,0))) THEN 'UNSUBSCRIBED' WHEN (subusers.user_id IS NOT NULL AND CAST(music.timestamp AS DECIMAL(20,0)) <= CAST(subusers.end_dt AS DECIMAL(20,0))) THEN 'SUBSCRIBED' END,batchid")

users_behavior.rdd.saveAsTextFile("/home/acadgild/project/output/users_behavior")

```

```

val connected_artists = spark.sql(s"SELECT ua.artists, COUNT(DISTINCT ua.user_id) AS user_count, md.batchid FROM Music_UserArtists ua INNER JOIN ( SELECT artist_id, song_id, user_id, batchid FROM Music_Data WHERE status='pass' AND batchid=$batchid ) md ON ua.artists=md.artist_id AND ua.user_id=md.user_id GROUP BY ua.artists,batchid ORDER BY user_count DESC LIMIT 10")
connected_artists.rdd.saveAsTextFile("/home/acadgild/project/output/connected_artists")

val top_10_royalty_songs = spark.sql(s"SELECT song_id, SUM(ABS(CAST(end_ts AS DECIMAL(20,0))-CAST(start_ts AS DECIMAL(20,0))) AS duration, batchid FROM Music_Data WHERE status='pass' AND batchid=$batchid AND (like=1 OR song_end_type=0) GROUP BY song_id,batchid ORDER BY duration DESC LIMIT 10")
top_10_royalty_songs.rdd.saveAsTextFile("/home/acadgild/project/output/top_10_royalty_songs")

val top_10_unsubscribed_users = spark.sql(s"SELECT md.user_id, SUM(ABS(CAST(md.end_ts AS DECIMAL(20,0))-CAST(md.start_ts AS DECIMAL(20,0))) AS duration FROM Music_Data md LEFT OUTER JOIN Music_SubscribedUsers su ON md.user_id=su.user_id WHERE md.status='pass' AND md.batchid=$batchid AND (su.user_id IS NULL OR (CAST(md.timestamp AS DECIMAL(20,0)) > CAST(su.end_dt AS DECIMAL(20,0)))) GROUP BY md.user_id ORDER BY duration DESC LIMIT 10")
top_10_unsubscribed_users.rdd.saveAsTextFile("/home/acadgild/project/output/top_10_unsubscribed_users")

```

## Problem Statements:

- Determine top 10 station\_id(s) where maximum number of songs were played, which were liked by unique users.
- Determine total duration of songs played by each type of user, where type of user can be '**subscribed**' or '**unsubscribed**'. An unsubscribed user is the one whose record is either not present in **Subscribed\_users** lookup table or has *subscription\_end\_date* earlier than the *timestamp* of the song played by him.
- Determine top 10 connected artists. Connected artists are those whose songs are most listened by the unique users who follow them.
- Determine top 10 songs who have generated the maximum revenue. Royalty applies to a song only if it was *liked* or was *completed successfully* or both.
- Determine top 10 unsubscribed users who listened to the songs for the longest duration.

## Problem Statement 1:

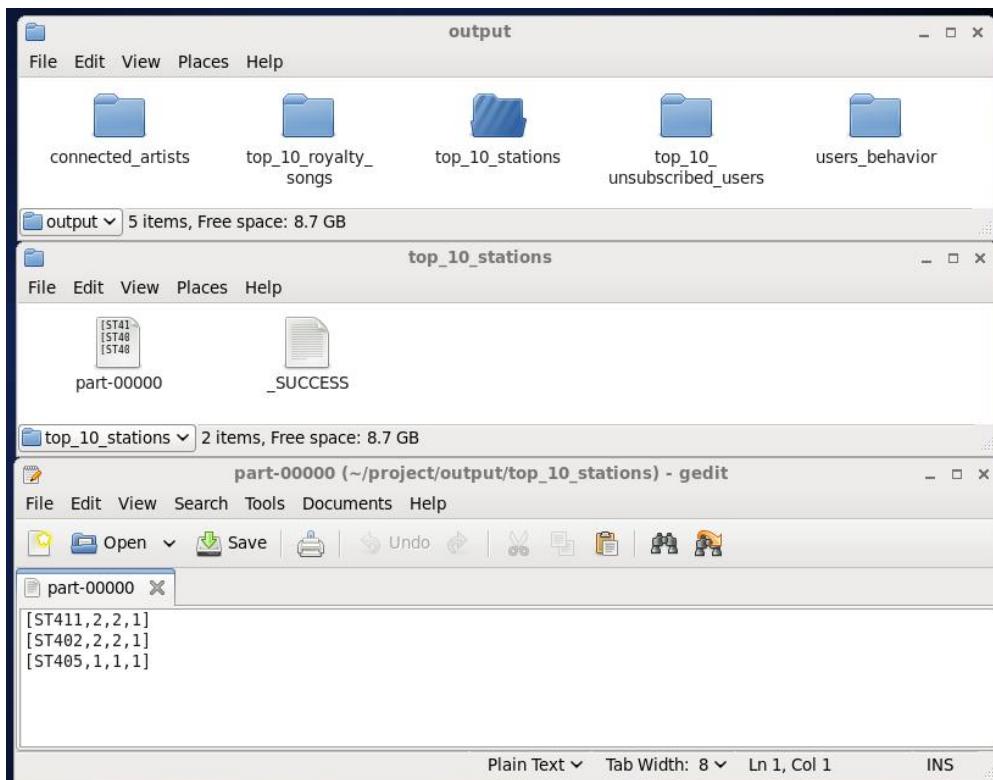
Determine top 10 station\_id(s) where maximum number of songs were played, which were liked by unique users.

### Code:

```
val Top10Stations = spark.sql(s"SELECT station_id, COUNT(DISTINCT song_id) AS total_distinct_songs_played, COUNT(DISTINCT user_id) AS distinct_user_count, batchid FROM Music_Data WHERE status='pass' AND batchid=$batchid AND like=1 GROUP BY station_id,batchid ORDER BY total_distinct_songs_played DESC LIMIT 10");

Top10Stations.rdd.saveAsTextFile("/home/acadgild/project/output/top_10_stations")
```

### Output:



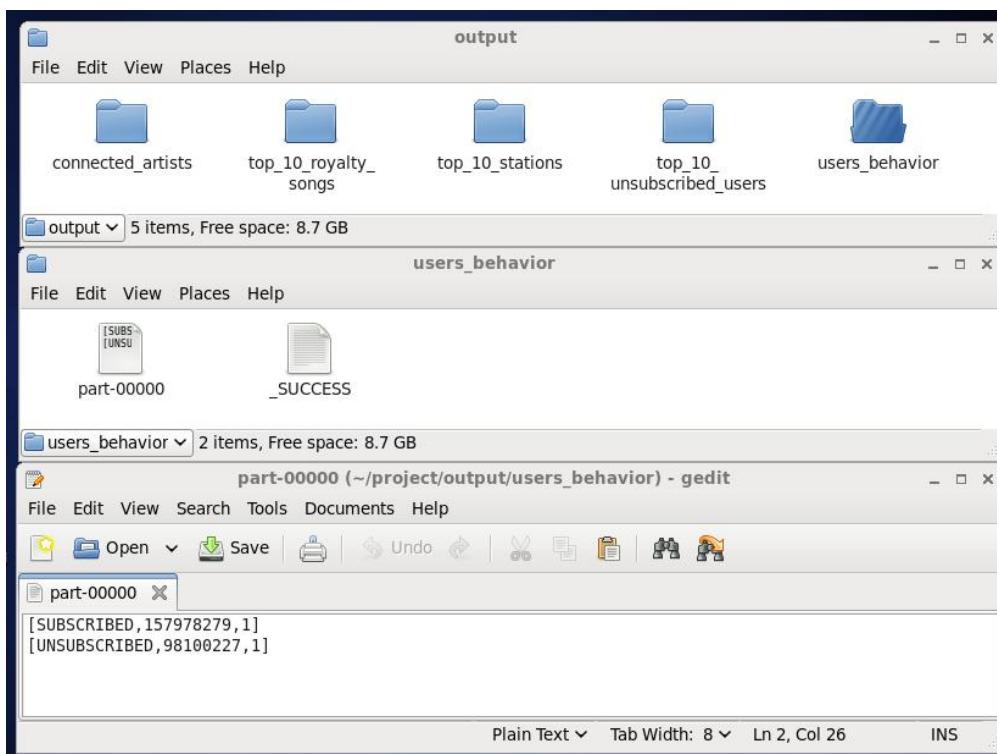
## Problem Statement 2:

Determine total duration of songs played by each type of user, where type of user can be '**subscribed**' or '**unsubscribed**'. An unsubscribed user is the one whose record is either not present in **Subscribed\_users** lookup table or has *subscription\_end\_date* earlier than the *timestamp* of the song played by him.

### Code:

```
val users_behavior = spark.sql(s"SELECT CASE WHEN (subusers.user_id IS NULL OR CAST(music.timestamp AS DECIMAL(20,0)) > CAST(subusers.end_dt AS DECIMAL(20,0))) THEN 'UNSUBSCRIBED' WHEN (subusers.user_id IS NOT NULL AND CAST(music.timestamp AS DECIMAL(20,0)) <= CAST(subusers.end_dt AS DECIMAL(20,0))) THEN 'SUBSCRIBED' END AS user_type, SUM(ABS(CAST(music.end_ts AS DECIMAL(20,0))-CAST(music.start_ts AS DECIMAL(20,0)))) AS duration, batchid FROM Music_Data music LEFT OUTER JOIN Music_SubscribedUsers subusers ON music.user_id=subusers.user_id WHERE music.status='pass' AND music.batchid=$batchid GROUP BY CASE WHEN (subusers.user_id IS NULL OR CAST(music.timestamp AS DECIMAL(20,0)) > CAST(subusers.end_dt AS DECIMAL(20,0))) THEN 'UNSUBSCRIBED' WHEN (subusers.user_id IS NOT NULL AND CAST(music.timestamp AS DECIMAL(20,0)) <= CAST(subusers.end_dt AS DECIMAL(20,0))) THEN 'SUBSCRIBED' END,batchid")  
users_behavior.rdd.saveAsTextFile("/home/acadgild/project/output/users_behavior")
```

### Output:



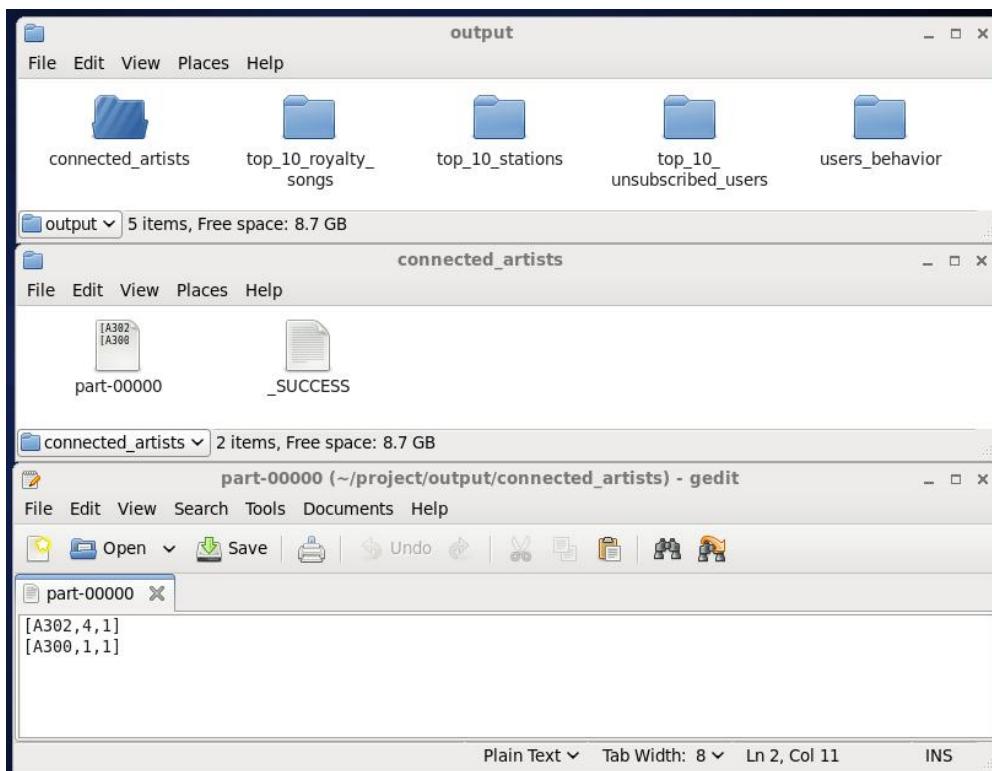
### Problem Statement 3:

Determine top 10 connected artists. Connected artists are those whose songs are most listened by the unique users who follow them.

#### Code:

```
val connected_artists = spark.sql(s"SELECT ua.artists, COUNT(DISTINCT ua.user_id) AS user_count, md.batchid FROM Music_UserArtists ua INNER JOIN ( SELECT artist_id, song_id, user_id, batchid FROM Music_Data WHERE status='pass' AND batchid=$batchid ) md ON ua.artists=md.artist_id AND ua.user_id=md.user_id GROUP BY ua.artists,batchid ORDER BY user_count DESC LIMIT 10")  
connected_artists.rdd.saveAsTextFile("/home/acadgild/project/output/connected_artists")
```

#### Output:



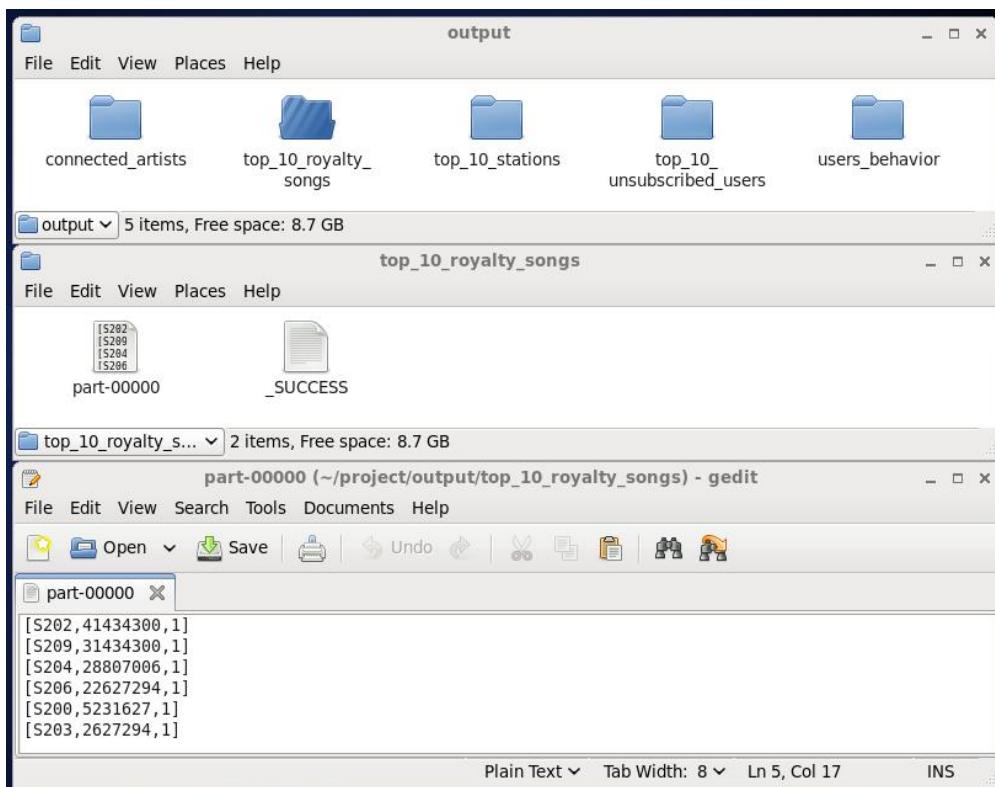
#### Problem Statement 4:

Determine top 10 songs who have generated the maximum revenue. Royalty applies to a song only if it was *liked* or was *completed successfully* or both.

#### Code:

```
val top_10_royalty_songs = spark.sql(s"SELECT song_id, SUM(ABS(CAST(end_ts AS DECIMAL(20,0))-CAST(start_ts AS DECIMAL(20,0)))) AS duration, batchid FROM Music_Data WHERE status='pass' AND batchid=$batchid AND (like=1 OR song_end_type=0) GROUP BY song_id,batchid ORDER BY duration DESC LIMIT 10")  
top_10_royalty_songs.rdd.saveAsTextFile("/home/acadgild/project/output/top_10_royalty_songs")
```

#### Output:



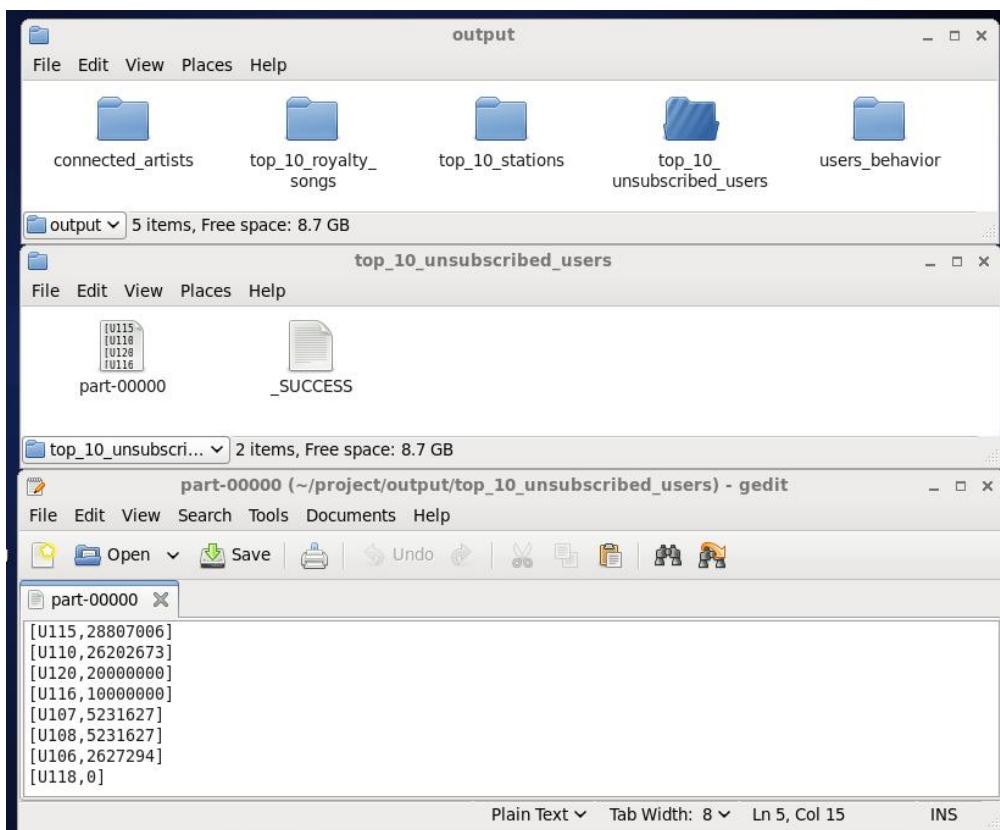
## Problem Statement 5:

Determine top 10 unsubscribed users who listened to the songs for the longest duration.

### Code:

```
val top_10_unsubscribed_users = spark.sql(s"SELECT md.user_id, SUM(ABS(CAST(md.end_ts AS DECIMAL(20,0))-CAST(md.start_ts AS DECIMAL(20,0)))) AS duration FROM Music_Data md LEFT OUTER JOIN Music_SubscribedUsers su ON md.user_id=su.user_id WHERE md.status='pass' AND md.batchid=$batchid AND (su.user_id IS NULL OR (CAST(md.timestamp AS DECIMAL(20,0)) > CAST(su.end_dt AS DECIMAL(20,0)))) GROUP BY md.user_id ORDER BY duration DESC LIMIT 10")  
top_10_unsubscribed_users.rdd.saveAsTextFile("/home/acadgild/project/output/top_10_unsubscribed_users")
```

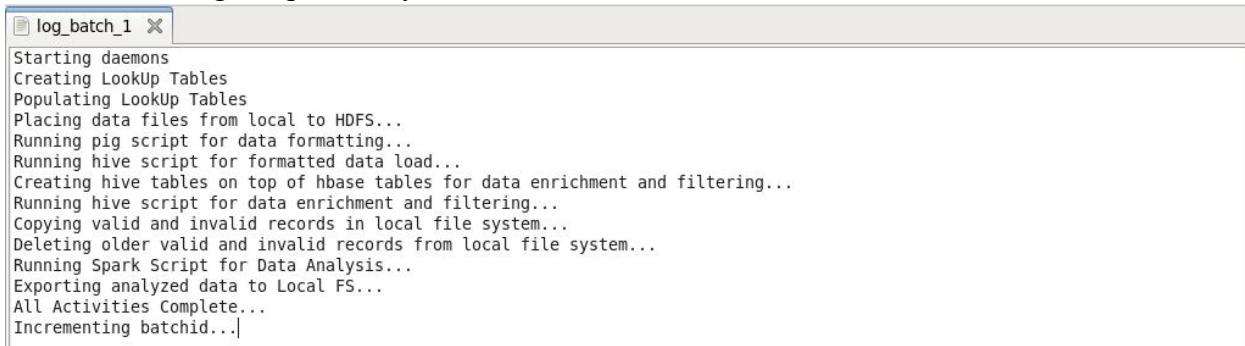
### Output:



Below is a view of the execution of the above:

## Post Analysis:

A view of the log file post analysis.



```
Starting daemons
Creating LookUp Tables
Populating LookUp Tables
Placing data files from local to HDFS...
Running pig script for data formatting...
Running hive script for formatted data load...
Creating hive tables on top of hbase tables for data enrichment and filtering...
Running hive script for data enrichment and filtering...
Copying valid and invalid records in local file system...
Deleting older valid and invalid records from local file system...
Running Spark Script for Data Analysis...
Exporting analyzed data to Local FS...
All Activities Complete...
Incrementing batchid...|
```

The batchid is incremented from 1 to 2:



A view of the log folder:

