Project Overview

A leading music-catering company is planning to analyze large amount of data received from varieties of sources, namely mobile app and website to track the behavior 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 files

Data files contain below fields.

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

LookUp Tables

There are some existing look up tables present in **NoSQL** databases. They play an important role in data enrichment and analysis.

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

DATASET

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

Data Enrichment

Rules for data enrichment,

4. If any of like or dislike is NULL or absent, consider it as 0.
5. 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**.

6. 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

Data Analysis (SHOULD BE IMPLEMETED IN SPARK)

It is not only the data which is important, rather it is the insight it can be used to generate important. Once we have made the data ready for analysis, we have to perform below analysis on a daily basis.

7. Determine top 10 station_ID(s) where maximum number of songs were played, which were liked by unique users.
8. 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.
9. Determine top 10 connected artists. Connected artists are those whose songs are most listened by the unique users who follow them.
10. 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.
11. Determine top 10 unsubscribed users who listened to the songs for the longest duration.

Challenges and Optimizations:

12. LookUp tables are in NoSQL databases. Integrate them with the actual data flow.
13. Try to make joins as less expensive as possible.
14. Data Cleaning, Validation, Enrichment, Analysis and Post Analysis have to be automated. Try using schedulers.
15. Appropriate logs have to maintain to track the behavior and overcome failures in the pipeline.

Flow of operations

A schematic flow of operations is shown below,

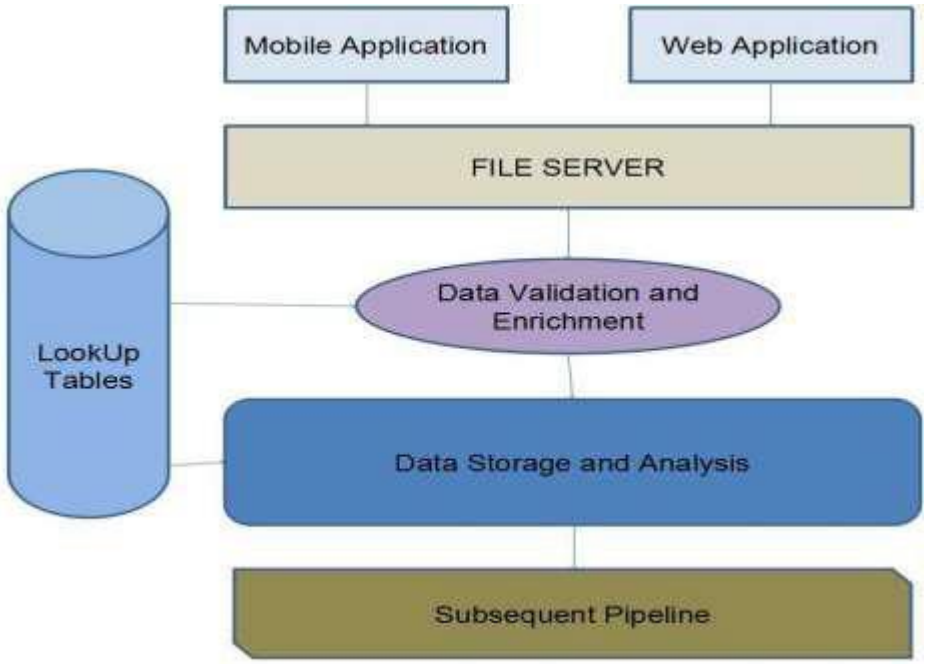


Fig-1

In the following sections, we are going to see the Music Data Analysis as per the above rules.

Low Level Design

The following flowchart shows the Low-Level design of this project,

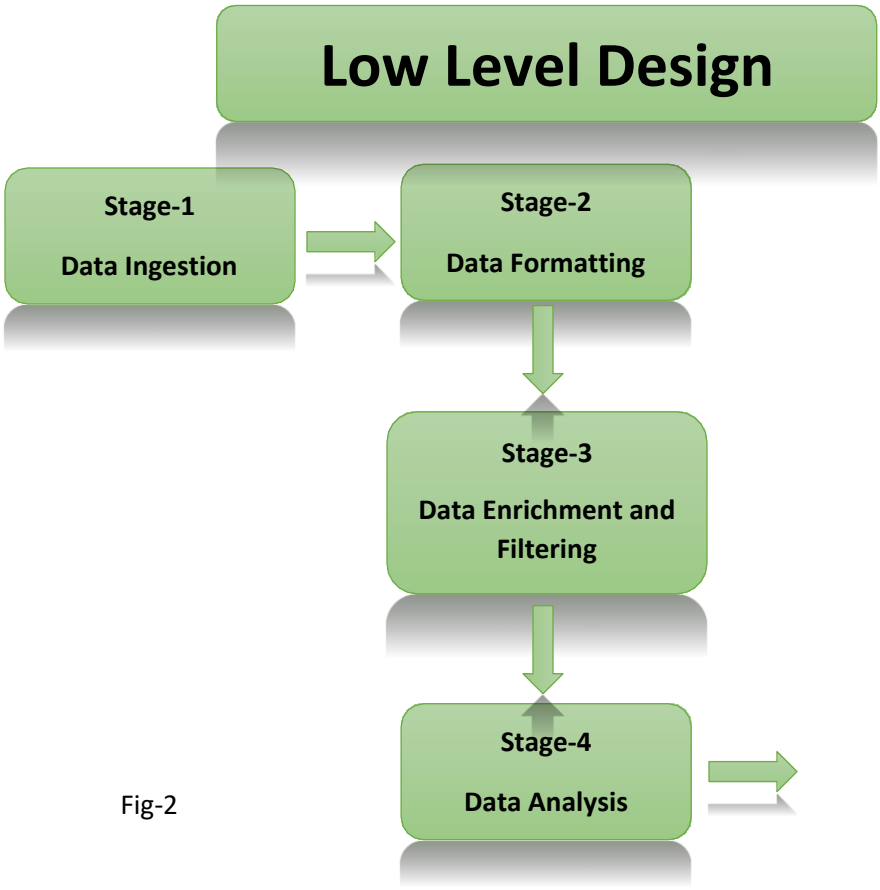


Fig-2

High Level Design

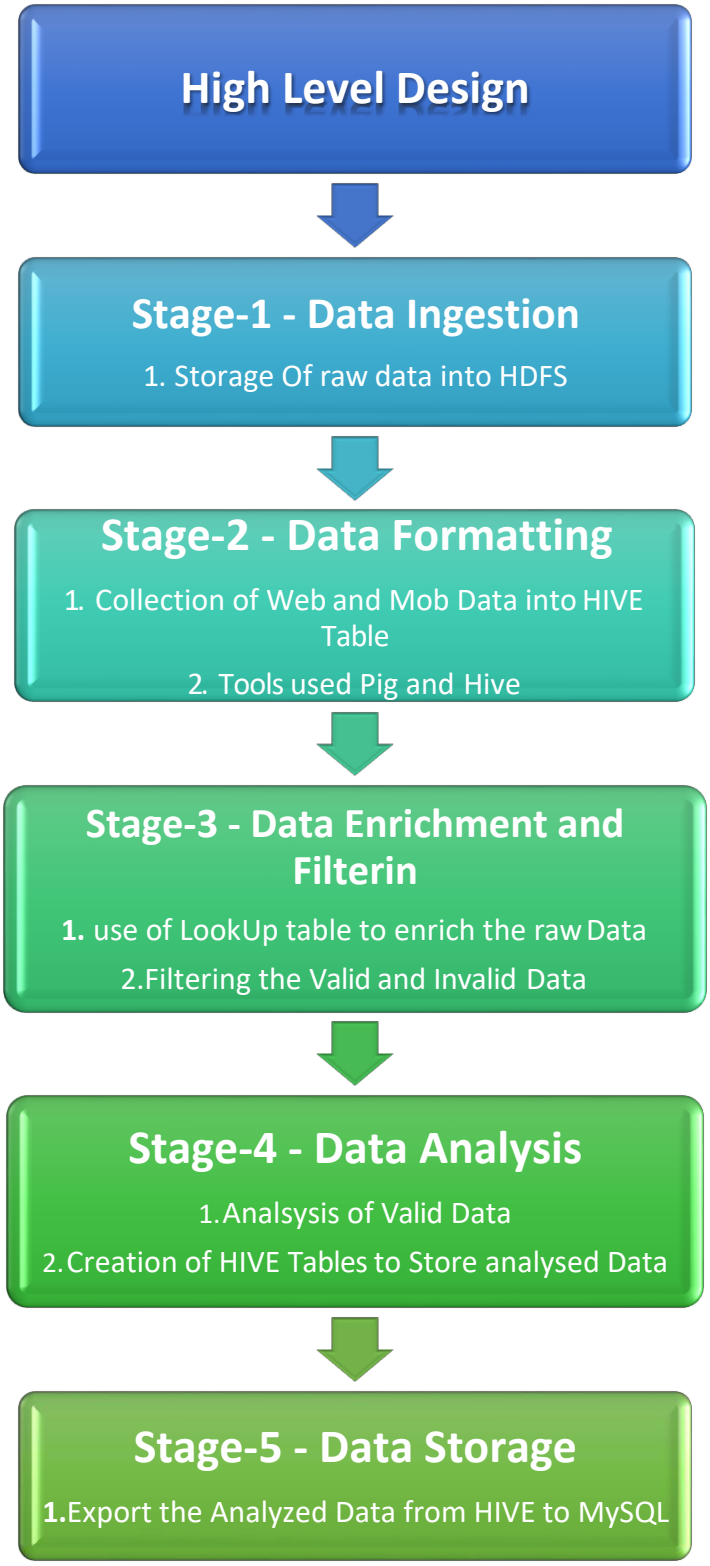


Fig-3

Hadoop Eco-System Implementation

1. We have created a batch file **"start-daemon.sh"** which starts the daemons such as **hive**, **hbase**, **MySQL** and rest of the all **hadoop** daemons.

Batch file script,

```
#!/bin/bash

if [ -f "/home/acadgild/Project_2_Music_Data_Analysis/logs/current-batch.txt" ]

then

echo "Batch File Found!"

else

echo -n "1" > "/home/acadgild/Project_2_Music_Data_Analysis/logs/current-batch.txt"

fi

chmod 775 /home/acadgild/Project_2_Music_Data_Analysis/logs/current-batch.txt
batchid=`cat /home/acadgild/Project_2_Music_Data_Analysis/logs/current-batch.txt`
LOGFILE=/home/acadgild/Project_2_Music_Data_Analysis/logs/log_batch_$batchid

echo "Starting daemons" >> $LOGFILE

# To Start Hadoop Daemons:
start-all.sh

# To start the HMASTER service:
start-hbase.sh

# To Start the JobHistory server Services:
mr-jobhistory-daemon.sh start historyserver

# To Start the mysql service
sudo service mysqld start

# To Start HIVE metastore:
hive --service metastore
```

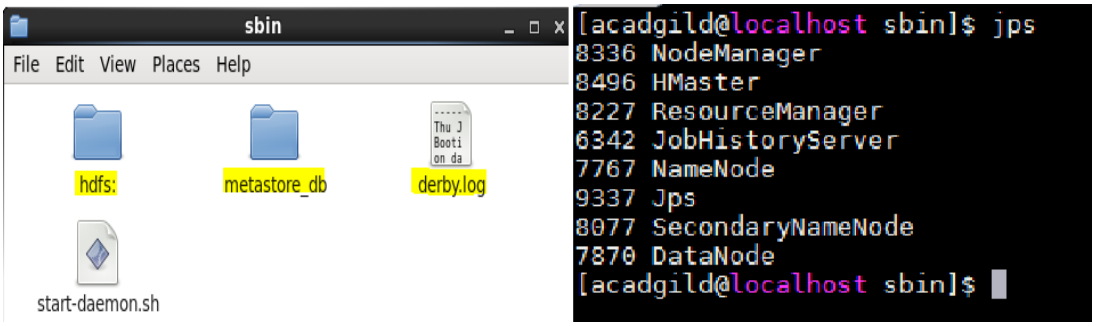
2. Starting all daemons,

- ***sh start-daemon.sh***

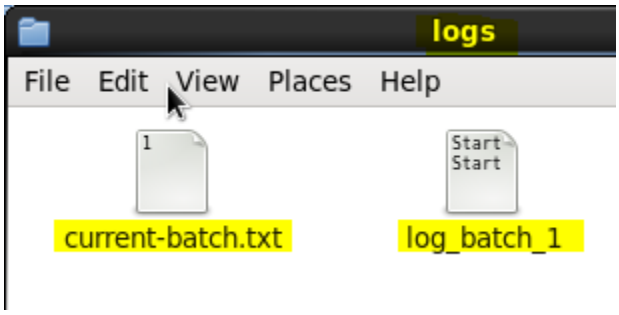
As per the batch file script all the hadoop daemons and the Hive, MySQL and Hive daemons are started shown in the below screen shot,

```
[acadgild@localhost sbin]$ sh start-daemon.sh
Batch File Found!
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which might have disabled
 stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/18 11:40:39 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/acadgild/hadoop-2.7.2/logs/hadoop-acadgild-namenode-localhost.localdomain.out
localhost: starting datanode, logging to /home/acadgild/hadoop-2.7.2/logs/hadoop-acadgild-datanode-localhost.localdomain.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/acadgild/hadoop-2.7.2/logs/hadoop-acadgild-secondarynamenode-localhost.localdomain.out
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which might have disabled
 stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/18 11:42:06 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
starting yarn daemons
starting resourcemanager, logging to /home/acadgild/hadoop-2.7.2/logs/yarn-acadgild-resourcemanager-localhost.localdomain.out
localhost: starting nodemanager, logging to /home/acadgild/hadoop-2.7.2/logs/yarn-acadgild-nodemanager-localhost.localdomain.out
starting master, logging to /home/acadgild/hbase-1.0.3/logs/hbase-acadgild-master-localhost.localdomain.out
historyserver running as process 6342. Stop it first.
[sudo] password for acadgild:
Starting mysqld:                                           [  OK  ]
Starting Hive Metastore Server
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/apache-hive-2.1.0-bin/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/hadoop-2.7.2/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBind
er.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
javax.jdo.JDODataStoreException: Required table missing : "VERSION" in Catalog "" Schema "". DataNucleus requires this table to perform its per
sistence operations. Either your MetaData is incorrect, or you need to enable "datanucleus.schema.autoCreateTables"
        at org.datanucleus.api.jdo.NucleusJDOHelper.getJDOExceptionForNucleusException(NucleusJDOHelper.java:553)
        at org.datanucleus.api.jdo.JDOPersistenceManager.jdoMakePersistent(JDOPersistenceManager.java:720)
```

3. We can see the list active services using the ***jps*** command, see below screen shot and also Starting the hive metastore created a metastore_db in the location where we desired,



4. The **start-daemon.sh** script will check whether the current-batch.txt file is available in the logs folder or not. If not it will create the file and dump value '1' in that file and create LOGFILE with the current **batchid**.



Data Ingestion, Formatting, Enrichment and Filtering

Data Ingestion

By using the "***populate-lookup.sh"*** script we will create lookup tables in **Hbase**. These tables have to be used in,

- Data formatting,
- Data enrichment and
- Analysis stage

Lookup Tables

Sl.no	Table Name	Description	Related File
1	station-geo-map	Contains mapping of a **geo_cd** with **station_id**	stn-geocd.txt
2	subscribed-users	Contains **user_id**, **subscription_start_date** and **subscription_end_date.** Contains details only for subscribed users	user-subscn.txt
3	song-artist-map	Contains mapping of **song_id** with **artist_id** Along with royalty associated with each play of the song	song-artist.txt
4	user-artist-map	Contains an array of **artist_id(**s) followed by a **user_id**	user-artist.txt

Table-1

"populate-lookup.sh" script

The "***populate-lookup.sh"*** shell script creates the above 4 lookup tables in the Hbase and populate the data into the lookup tables from the dataset files.

In the below screen shots, we can see the create-lookup.sh scripts and the following screen shots shows the tables creation and population of the data in the Hbase. Also, the values loaded into the Hbase Tables are also shown, please see the below screen shots.

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

```

Run the script: *./populate-lookup.sh*

```
-rwxrwxr--. 1 acadgild acadgild   412 Jan 19 22:14 wrapper.sh
[acadgild@localhost scripts]$ ./populate-lookup.sh
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/hbase-1.0.3/lib/slf4j-l
SLF4J: Found binding in [jar:file:/home/acadgild/hadoop-2.7.2/share/hado
er.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an expl
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgi
 stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <lib
2018-01-19 22:42:24,744 WARN  [main] util.NativeCodeLoader: Unable to lo
s where applicable
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.0.3, rf1e1312f9790a7c40f6a4b5a1bab2ea1dd559890, Tue Jan 19 19:

create 'station-geo-map', 'geo'
0 row(s) in 1.3100 seconds

Hbase::Table - station-geo-map
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/hbase-1.0.3/lib/slf4j-l
```

```
create 'subscribed-users', 'subscn'
0 row(s) in 1.7040 seconds

Hbase::Table - subscribed-users
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/hbase-1.0.3/lib/slf4j-log4j1
SLF4J: Found binding in [jar:file:/home/acadgild/hadoop-2.7.2/share/hadoop/co
er.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanati
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/ha
 stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>
2018-01-19 22:43:17,551 WARN  [main] util.NativeCodeLoader: Unable to load na
s where applicable
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.0.3, rf1e1312f9790a7c40f6a4b5a1bab2ea1dd559890, Tue Jan 19 19:26:53

create 'song-artist-map', 'artist'
0 row(s) in 1.4620 seconds

Hbase::Table - song-artist-map
```

```
Type "exit<RETURN>" to leave the HBase Shell
Version 1.0.3, rf1e1312f9790a7c40f6a4b5a1bab2ea1dd559890, Tue Jan 19 19:26:53

put 'subscribed-users', 'U114', 'subscn:enddt', '1468130523'
0 row(s) in 1.1740 seconds

SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/apache-hive-2.1.0-bin/lib/log
SLF4J: Found binding in [jar:file:/home/acadgild/hadoop-2.7.2/share/hadoop/com
er.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanatio
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/acadgild/apache-hive
c: true
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/had
```

We can see the lookup tables created using the ***"populate-lookup.sh"*** in the below screen shot,

Lookup Tables in the hbase shell,

```
hbase(main):040:0> list
TABLE
song-artist-map
station-geo-map
subscribed-users
3 row(s) in 0.1250 seconds

=> ["song-artist-map", "station-geo-map", "subscribed-users"]
hbase(main):041:0>
```

The values loaded in the Lookup tables are shown below,

song-artist-map

```
hbase(main):061:0* scan 'song-artist-map'
ROW                                   COLUMN+CELL
 S200                                 column=artist:artistid, timestamp=1516341421411, value=A300
 S201                                 column=artist:artistid, timestamp=1516341435130, value=A301
 S202                                 column=artist:artistid, timestamp=1516341449406, value=A302
 S203                                 column=artist:artistid, timestamp=1516341464984, value=A303
 S204                                 column=artist:artistid, timestamp=1516341479845, value=A304
 S205                                 column=artist:artistid, timestamp=1516341494365, value=A301
 S206                                 column=artist:artistid, timestamp=1516341509536, value=A302
 S207                                 column=artist:artistid, timestamp=1516341524259, value=A303
 S208                                 column=artist:artistid, timestamp=1516341537840, value=A304
 S209                                 column=artist:artistid, timestamp=1516341551721, value=A305
10 row(s) in 0.1250 seconds
```

station-geo-map

```
hbase(main):062:0> scan 'station-geo-map'
ROW                                   COLUMN+CELL
 ST400                                column=geo:geo_cd, timestamp=1516341188768, value=A
 ST401                                column=geo:geo_cd, timestamp=1516341208229, value=AU
 ST402                                column=geo:geo_cd, timestamp=1516341225914, value=AP
 ST403                                column=geo:geo_cd, timestamp=1516341247762, value=J
 ST404                                column=geo:geo_cd, timestamp=1516341264812, value=E
 ST405                                column=geo:geo_cd, timestamp=1516341278706, value=A
 ST406                                column=geo:geo_cd, timestamp=1516341293480, value=AU
 ST407                                column=geo:geo_cd, timestamp=1516341308185, value=AP
 ST408                                column=geo:geo_cd, timestamp=1516341322088, value=E
 ST409                                column=geo:geo_cd, timestamp=1516341337723, value=E
 ST410                                column=geo:geo_cd, timestamp=1516341351596, value=A
 ST411                                column=geo:geo_cd, timestamp=1516341365274, value=A
 ST412                                column=geo:geo_cd, timestamp=1516341379574, value=AP
 ST413                                column=geo:geo_cd, timestamp=1516341393291, value=J
 ST414                                column=geo:geo_cd, timestamp=1516341407388, value=E
15 row(s) in 0.0830 seconds
```

subscribed-users

```
hbase(main):063:0> scan 'subscribed-users'
ROW                                   COLUMN+CELL
 U100                                 column=subscn:enddt, timestamp=1516341581655, value=1465130523
 U100                                 column=subscn:startdt, timestamp=1516341566016, value=1465230523
 U101                                 column=subscn:enddt, timestamp=1516341609966, value=1475130523
 U101                                 column=subscn:startdt, timestamp=1516341596203, value=1465230523
 U102                                 column=subscn:enddt, timestamp=1516341639844, value=1475130523
 U102                                 column=subscn:startdt, timestamp=1516341625162, value=1465230523
 U103                                 column=subscn:enddt, timestamp=1516341668849, value=1475130523
 U103                                 column=subscn:startdt, timestamp=1516341654569, value=1465230523
 U104                                 column=subscn:enddt, timestamp=1516341698838, value=1475130523
 U104                                 column=subscn:startdt, timestamp=1516341684423, value=1465230523
 U105                                 column=subscn:enddt, timestamp=1516341726878, value=1475130523
 U105                                 column=subscn:startdt, timestamp=1516341713257, value=1465230523
 U106                                 column=subscn:enddt, timestamp=1516341756353, value=1485130523
 U106                                 column=subscn:startdt, timestamp=1516341740927, value=1465230523
 U107                                 column=subscn:enddt, timestamp=1516341785496, value=1455130523
 U107                                 column=subscn:startdt, timestamp=1516341770793, value=1465230523
 U108                                 column=subscn:enddt, timestamp=1516341815102, value=1465230623
 U108                                 column=subscn:startdt, timestamp=1516341800313, value=1465230523
 U109                                 column=subscn:enddt, timestamp=1516341843290, value=1475130523
 U109                                 column=subscn:startdt, timestamp=1516341829227, value=1465230523
 U110                                 column=subscn:enddt, timestamp=1516341871578, value=1475130523
 U110                                 column=subscn:startdt, timestamp=1516341857362, value=1465230523
 U111                                 column=subscn:enddt, timestamp=1516341900490, value=1475130523
 U111                                 column=subscn:startdt, timestamp=1516341886141, value=1465230523
 U112                                 column=subscn:enddt, timestamp=1516341929297, value=1475130523
 U112                                 column=subscn:startdt, timestamp=1516341914639, value=1465230523
 U113                                 column=subscn:enddt, timestamp=1516341958696, value=1485130523
 U113                                 column=subscn:startdt, timestamp=1516341944389, value=1465230523
 U114                                 column=subscn:enddt, timestamp=1516341988193, value=1468130523
 U114                                 column=subscn:startdt, timestamp=1516341973580, value=1465230523
15 row(s) in 0.1170 seconds
```

We have successfully created the lookup tables in the Hbase.

The populate-lookup.sh also creates a lookup table **“users_artists”** in the HIVE, loading the data from the **user-artist.txt,** the below screen shot shows that the table has been created in the HIVE.

```
c: true
Java HotSpot(TM) Client VM warning: You have
 stack guard. The VM will try to fix the stac
It's highly recommended that you fix the libr
OK
Time taken: 2.705 seconds
OK
Time taken: 0.089 seconds
OK
Time taken: 1.689 seconds
Loading data to table project.users_artists
OK
Time taken: 2.168 seconds
[acadgild@localhost scripts]$
```

hive> Select * From users_artists;

```
z) or using Hive 1.X releases.
hive> Show Databases;
OK
default
project
Time taken: 2.468 seconds, Fetched: 2 row(s)
hive> Use project;
OK
Time taken: 0.042 seconds
hive> Show Tables;
OK
users_artists
Time taken: 0.178 seconds, Fetched: 1 row(s)
hive> Select * From users_artists;
OK
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"]
Time taken: 3.591 seconds, Fetched: 15 row(s)
hive>
    >
```

Now we need to link theses lookup tables in hive using the Hbase Storage Handler.

With the help of **“data_enrichment_filtering_schema.sh”** file we will create hive tables on the top of Hbase tables using **“create_hive_hbase_lookup.hql”.**

Creating Hive Tables on the top of Hbase:

In this section 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.

Run the script: **./data_enrichment_filtering_schema.sh,**

The script will run the **"create_hive_hbase_lookup.hql"** which will create the HIVE external tables with the help of **Hbase storage handler & SerDe properties**. The hive external tables will match the columns of **Hbase** tables to **HIVE** tables.

```
#!/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

```
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");

```

The below screenshot we can see tables getting created in hive by running the **"data_enrichement_filtering_schema.sh file"**

```
[acadgild@localhost scripts]$ ./data_enrichment_filtering_schema.sh
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/apache-hive-2.1.0-bin/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/St
SLF4J: Found binding in [jar:file:/home/acadgild/hadoop-2.7.2/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4
er.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/acadgild/apache-hive-2.1.0-bin/lib/hive-common-2.1.0.jar!/hive
c: true
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 wh
 stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
OK
Time taken: 1.77 seconds
OK
Time taken: 3.301 seconds
OK
Time taken: 0.305 seconds
OK
Time taken: 0.271 seconds
[acadgild@localhost scripts]$ hive
```

Hive>Show Tables;

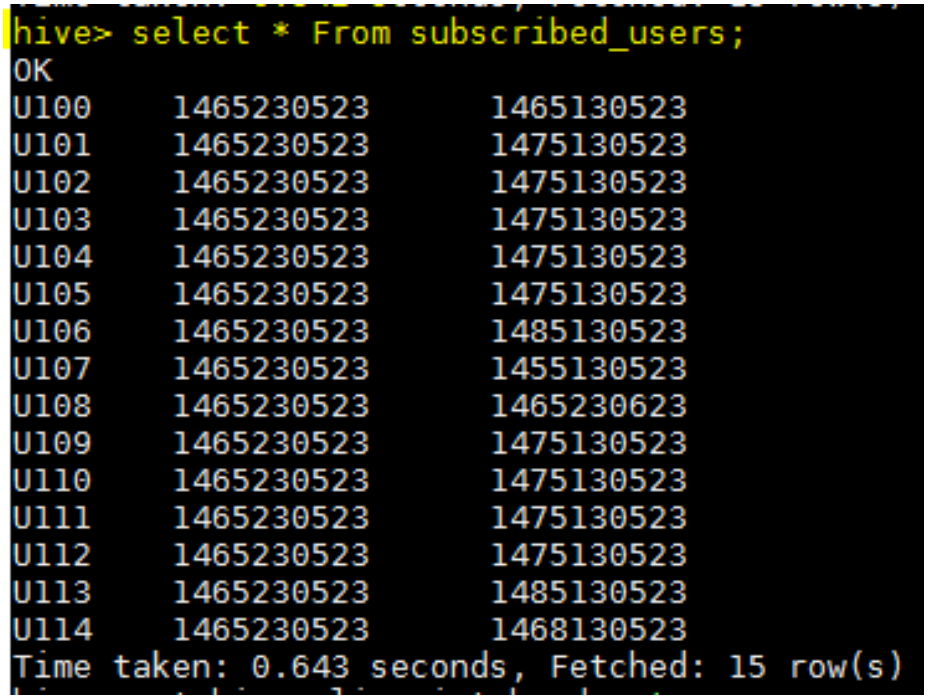
```
hive> use project;
OK
Time taken: 1.485 seconds
hive> show tables;
OK
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 0.513 seconds, Fetched: 4 row(s)
hive>
```

hive>Select * From song_artist_map

```
Time taken: 0.103 seconds, Fetched: 4 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: 0.421 seconds, Fetched: 10 row(s)
```

hive>Select * From station_geo_map

hive>Select * From Subscribed_users



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 script for every 3 hours.

Run the script: *./dataformatting.sh*

```
#!/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/* /user/acadgild/project/batch${batchid}/web/
hadoop fs -put /home/acadgild/project/data/mob/* /user/acadgild/project/batch${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

```

```
-rwxrwxr--. 1 acadgild acadgild   412 Jan 19 22:14 wrapper.sh
[acadgild@localhost scripts]$ ./dataformatting.sh
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which might have disabled
 stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/20 17:58:29 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
18/01/20 17:58:30 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0 minutes, Emptier interval = 0 minutes.
Deleted /user/acadgild/project/batch1/web
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which might have disabled
 stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/20 17:58:34 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
rm: `/user/acadgild/project/batch1/formattedweb/': No such file or directory
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which might have disabled
 stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/20 17:58:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
18/01/20 17:58:42 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0 minutes, Emptier interval = 0 minutes.
Deleted /user/acadgild/project/batch1/mob
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which might have disabled
 stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/20 17:58:46 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
```

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

- **Dataformatting.pig**
- **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

Dataformatting.pig

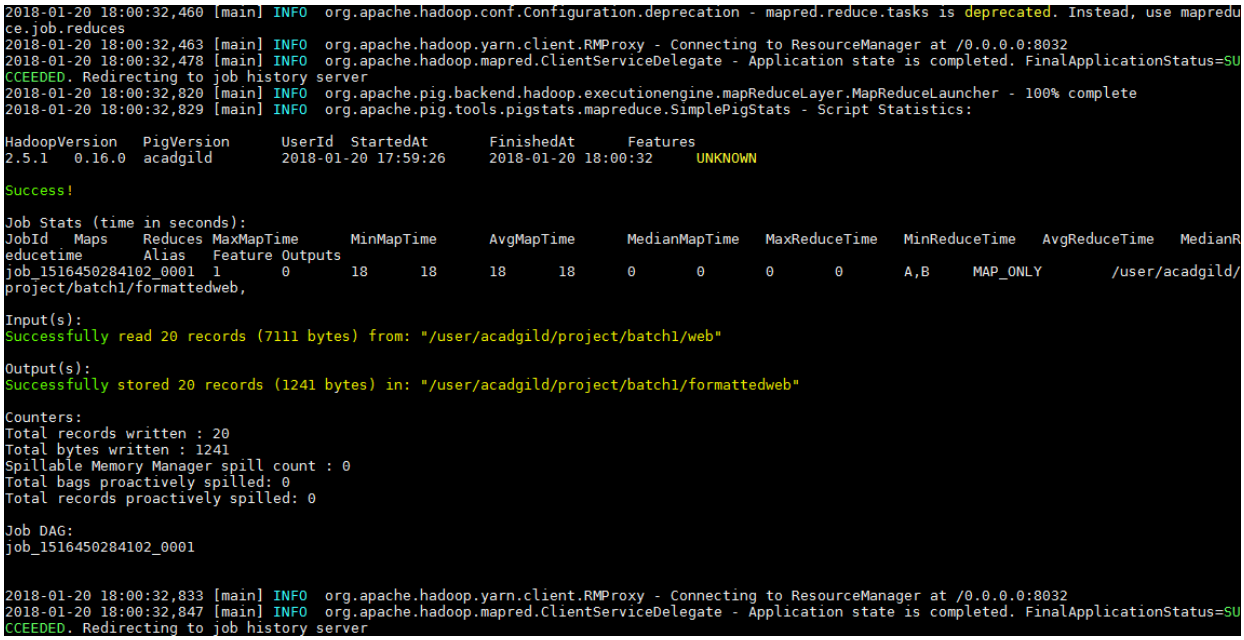
```
REGISTER /home/acadgild/project/lib/piggybank.jar;

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(',');

```

formatted_hive_load.hql

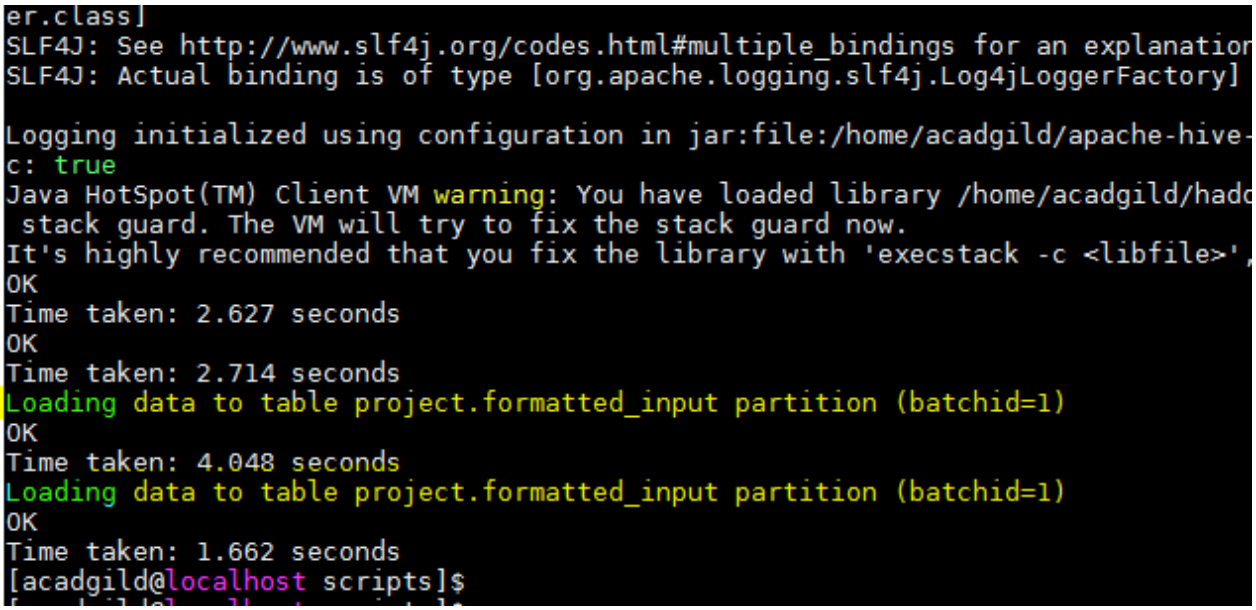
```
set hive.support.sql11.reserved.keywords=false;
USE project;

CREATE TABLE IF NOT EXISTS formatted_input
(
user_id STRING,
song_id STRING,
artist_id STRING,
timestp 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});

```

In the below screenshot we can see the data both the scripts in action, first pig script will parse the data and then hive script will load the data into hive terminal successfully.

Pig script successful completion,

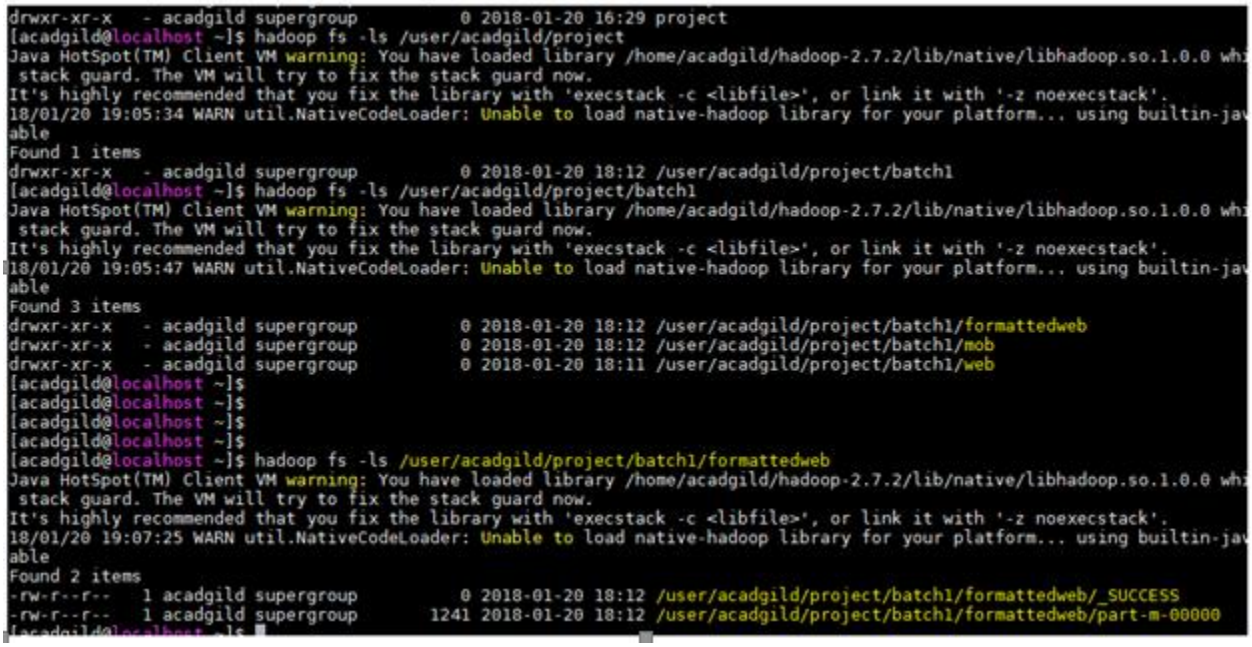


Hive script successfully load the data into hive terminal,



In the above screenshot we can see the **dataformatting.pig** along with the **formatted_hive_load.hql** executed successfully.

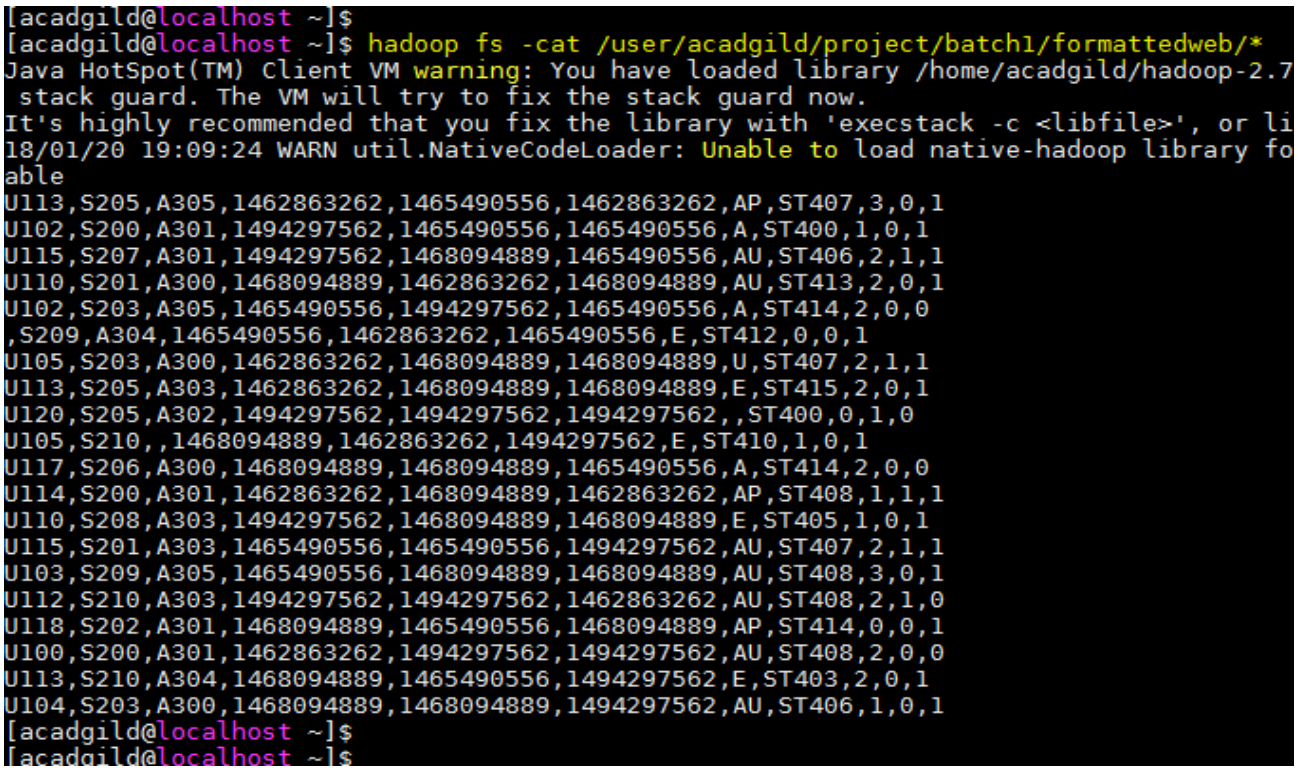
The output of **dataformatting.sh** script in HDFS folders:



The output of the **formattedweb** data obtained from the **Dataformatting.pig** is shown in the below screen shot,

Command,

hadoop fs -cat /user/acadgild/project/batch1/formattedweb/*



The new Tables has been created and show below,

```
2) or using Hive 1.X releases.
hive> use project;
OK
Time taken: 1.467 seconds
hive> show tables;
OK
formatted_input
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 0.719 seconds, Fetched: 5 row(s)
```

DataFormatting.sh output in hive terminal,

hive> select * from formatted_input;

```
Time taken: 0.719 seconds, Fetched: 5 row(s)
hive> select * from formatted_input;
OK
U117    S204    A301    1495130523      1465130523      1475130523      A       ST402   0       1       0       1
U115    S203    A305    1465230523      1465130523      1475130523      AP      ST409   0       1       0       1
U117    S208    A305    1465130523      1465130523      1465130523      AP      ST407   3       0       1       1
U111    S206    A303    1465230523      1485130523      1465130523      U       ST414   1       0       0       1
U119    S207    A301    1465230523      1475130523      1485130523      AU      ST408   1       1       1       1
        S209    A301    1465230523      1465230523      1485130523      U       ST411   3       0       1       1
U112    S207    A302    1465230523      1465230523      1475130523      AU      ST410   0       1       1       1
U118    S203    A304    1475130523      1465130523      1465230523      U       ST403   0       0       0       1
U101    S204    A301    1475130523      1485130523      1485130523              ST411   2       0       1       1
U103    S207            1465230523      1465130523      1465130523      A       ST400   1       1       1       1
U113    S202    A300    1465130523      1475130523      1475130523      U       ST415   1       1       0       1
U104    S206    A303    1495130523      1465130523      1475130523      U       ST401   1       1       1       1
U113    S207    A305    1495130523      1465130523      1485130523      AU      ST402   0       0       1       1
U101    S206    A305    1465130523      1465230523      1465230523      AP      ST415   3       0       0       1
U110    S202    A303    1495130523      1465130523      1465130523      AP      ST413   0       0       1       1
U118    S208    A304    1465130523      1475130523      1465130523      E       ST410   0       1       1       1
U118    S209    A305    1475130523      1465230523      1465230523      E       ST400   0       0       0       1
U108    S200    A300    1495130523      1475130523      1465230523      U       ST400   1       0       1       1
U105    S208    A300    1465130523      1475130523      1465230523      AU      ST410   1       0       0       1
U118    S201    A304    1465230523      1475130523      1485130523      A       ST408   2       1       1       1
U113    S205    A305    1462863262      1465490556      1462863262      AP      ST407   3       0       1       1
U102    S200    A301    1494297562      1465490556      1465490556      A       ST400   1       0       1       1
U115    S207    A301    1494297562      1468094889      1465490556      AU      ST406   2       1       1       1
U110    S201    A300    1468094889      1462863262      1468094889      AU      ST413   2       0       1       1
U102    S203    A305    1465490556      1494297562      1465490556      A       ST414   2       0       0       1
        S209    A304    1465490556      1462863262      1465490556      E       ST412   0       0       1       1
U105    S203    A300    1462863262      1468094889      1468094889      U       ST407   2       1       1       1
U113    S205    A303    1462863262      1468094889      1468094889      E       ST415   2       0       1       1
U120    S205    A302    1494297562      1494297562      1494297562              ST400   0       1       0       1
U105    S210            1468094889      1462863262      1494297562      E       ST410   1       0       1       1
U117    S206    A300    1468094889      1468094889      1465490556      A       ST414   2       0       0       1
U114    S200    A301    1462863262      1468094889      1462863262      AP      ST408   1       1       1       1
U110    S208    A303    1494297562      1468094889      1468094889      E       ST405   1       0       1       1
U115    S201    A303    1465490556      1465490556      1494297562      AU      ST407   2       1       1       1
U103    S209    A305    1465490556      1468094889      1468094889      AU      ST408   3       0       1       1
U112    S210    A303    1494297562      1494297562      1462863262      AU      ST408   2       1       0       1
U118    S202    A301    1468094889      1465490556      1468094889      AP      ST414   0       0       1       1
U100    S200    A301    1462863262      1494297562      1494297562      AU      ST408   2       0       0       1
U113    S210    A304    1468094889      1465490556      1494297562      E       ST403   2       0       1       1
U104    S203    A300    1468094889      1468094889      1494297562      AU      ST406   1       0       1       1
Time taken: 3.192 seconds, Fetched: 40 row(s)
hive>
```

- In the above screenshot we can see the formatted input data with some null values in **user_id, aritist_id** and **geo_cd** columns which we will fill the enrichment script based on rules of enrichment for **artist_id** and **geo_cd** only. We will get neglect **user_id** because they didn't mentioned anything about **user_id** for enrichment purpose.
- Data formatting phase is executed successfully by loading both **mobile** and **web** data and partitioned based on **batchid**.

Data Enrichment & Filtering

In this stage, 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.

data_enrichment.sh

```
#!/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 {} \;
```

data_enrichment.hql

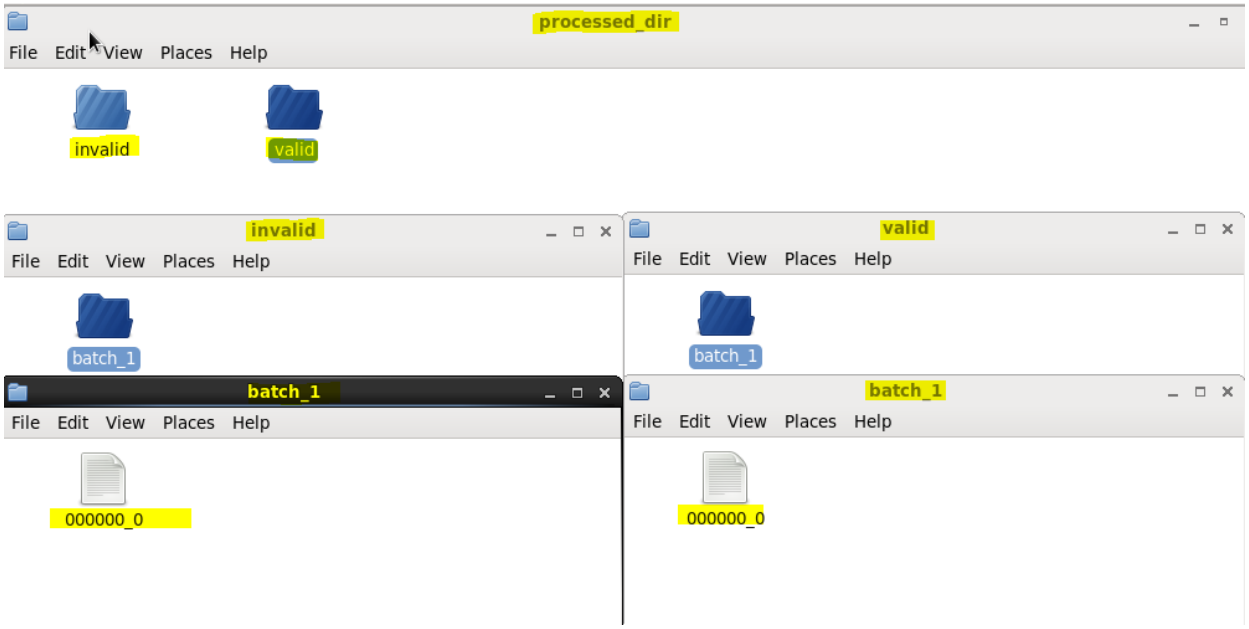
```
set hive.support.sql11.reserved.keywords=false;
SET hive.auto.convert.join=false;
SET hive.exec.dynamic.partition.mode=nonstrict;

USE project;

CREATE TABLE IF NOT EXISTS enriched_data
(
user_id STRING,
song_id STRING,
artist_id STRING,
timestp 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.timestp,
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.timestp IS NULL
OR i.start_ts IS NULL
OR i.end_ts IS NULL
OR i.user_id=''
OR i.song_id=''
OR i.timestp=''
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};
```

```
[acadgild@localhost scripts]$ ./data_enrichment.sh
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/apache-hive-2.1.0-bin/lib/log4j-slf4j-impl
SLF4J: Found binding in [jar:file:/home/acadgild/hadoop-2.7.2/share/hadoop/common/lib/slf4j
er.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/acadgild/apache-hive-2.1.0-bin/li
c: true
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib
 stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it
OK
Time taken: 2.344 seconds
OK
Time taken: 1.592 seconds
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions
spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180121050629_4da8c068-b197-457a-8f78-6cd1e80c34b7
Total jobs = 2
```

```
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1516485910189_0006, Tracking URL = http://localhost:8088/proxy/application_1516485910189_0006/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job  -kill job_1516485910189_0006
Hadoop job information for Stage-2: number of mappers: 2; number of reducers: 1
2018-01-21 05:08:49,177 Stage-2 map = 0%,  reduce = 0%
2018-01-21 05:09:12,849 Stage-2 map = 50%,  reduce = 0%, Cumulative CPU 2.2 sec
2018-01-21 05:09:15,182 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 5.74 sec
2018-01-21 05:09:31,414 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 9.86 sec
MapReduce Total cumulative CPU time: 9 seconds 860 msec
Ended Job = job_1516485910189_0006
Loading data to table project.enriched_data partition (batchid=null, status=null)

Loaded : 2/2 partitions.
         Time taken to load dynamic partitions: 1.231 seconds
         Time taken for adding to write entity : 0.004 seconds
MapReduce Jobs Launched:
Stage-Stage-1: Map: 3  Reduce: 1   Cumulative CPU: 14.16 sec   HDFS Read: 50542 HDFS Write: 3280 SUCCESS
Stage-Stage-2: Map: 2  Reduce: 1   Cumulative CPU: 9.86 sec   HDFS Read: 25154 HDFS Write: 3177 SUCCESS
Total MapReduce CPU Time Spent: 24 seconds 20 msec
OK
Time taken: 182.412 seconds
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 wh
 stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/21 05:09:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-ja
able
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 wh
 stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/21 05:09:45 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-ja
able
```

At the end script will automatically divide the records based on status **pass** & **fail** and dump the result into **processed_dir** folder with valid and invalid folders.

```
drwxrwxr-x. 3 acadgild acadgild 4096 Jan 21 05:09 invalid
drwxrwxr-x. 3 acadgild acadgild 4096 Jan 21 05:09 valid
[acadgild@localhost processed_dir]$ ls -l invalid
total 4
drwxrwxr-x. 2 acadgild acadgild 4096 Jan 21 05:09 batch_1
[acadgild@localhost processed_dir]$ ls -l invalid/batch_1
total 4
-rw-r--r--. 1 acadgild acadgild 1505 Jan 21 05:09 000000_0
[acadgild@localhost processed_dir]$
[acadgild@localhost processed_dir]$ ls -l valid/batch_1
total 4
-rw-r--r--. 1 acadgild acadgild 1507 Jan 21 05:09 000000_0
[acadgild@localhost processed_dir]$
```



Now we can check whether the data properly loaded in the hive terminal or not.

```
hive> use project;
OK
Time taken: 2.773 seconds
hive> show tables;
OK
enriched_data
formatted_input
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 1.291 seconds, Fetched: 6 row(s)
hive> select * from enriched_data;
```

In the below screenshot we have data for **enriched_data** table where we filled the null values of **artist_id** and **geo_cd** of formatted input with the help of lookup tables,

hive>select * From enriched_data;

```
    >
    > Select * From enriched_data;
OK
U114    S200    A301    1462863262      1468094889      1462863262      AP      ST408   1       1       1       1       fail
U118    S201    A304    1465230523      1475130523      1485130523      A       ST408   2       1       1       1       fail
U115    S201    A303    1465490556      1465490556      1494297562      AU      ST407   2       1       1       1       fail
U113    S202    A300    1465130523      1475130523      1475130523      U       ST415   1       1       0       1       fail
U105    S203    A300    1462863262      1468094889      1468094889      U       ST407   2       1       1       1       fail
U113    S205    A303    1462863262      1468094889      1468094889      E       ST415   2       0       1       1       fail
U101    S206    A305    1465130523      1465230523      1465230523      AP      ST415   3       0       0       1       fail
U104    S206    A303    1495130523      1465130523      1475130523      U       ST401   1       1       1       1       fail
U112    S207    A302    1465230523      1465230523      1475130523      AU      ST410   0       1       1       1       fail
U103    S207    A303    1465230523      1465130523      1465130523      A       ST400   1       1       1       1       fail
U119    S207    A301    1465230523      1475130523      1485130523      AU      ST408   1       1       1       1       fail
U115    S207    A301    1494297562      1468094889      1465490556      AU      ST406   2       1       1       1       fail
U118    S208    A304    1465130523      1475130523      1465130523      E       ST410   0       1       1       1       fail
        S209    A301    1465230523      1465230523      1485130523      U       ST411   3       0       1       1       fail
        S209    A304    1465490556      1462863262      1465490556      E       ST412   0       0       1       1       fail
U113    S210    A304    1468094889      1465490556      1494297562      E       ST403   2       0       1       1       fail
U105    S210    NULL    1468094889      1462863262      1494297562      E       ST410   1       0       1       1       fail
U112    S210    A303    1494297562      1494297562      1462863262      AU      ST408   2       1       0       1       fail
U102    S200    A301    1494297562      1465490556      1465490556      A       ST400   1       0       1       1       pass
U108    S200    A300    1495130523      1475130523      1465230523      U       ST400   1       0       1       1       pass
U100    S200    A301    1462863262      1494297562      1494297562      AU      ST408   2       0       0       1       pass
U110    S201    A300    1468094889      1462863262      1468094889      AU      ST413   2       0       1       1       pass
U110    S202    A303    1495130523      1465130523      1465130523      AP      ST413   0       0       1       1       pass
U118    S202    A301    1468094889      1465490556      1468094889      AP      ST414   0       0       1       1       pass
U118    S203    A304    1475130523      1465130523      1465230523      U       ST403   0       0       0       1       pass
U104    S203    A300    1468094889      1468094889      1494297562      AU      ST406   1       0       1       1       pass
U115    S203    A305    1465230523      1465130523      1475130523      AP      ST409   0       1       0       1       pass
U102    S203    A305    1465490556      1494297562      1465490556      A       ST414   2       0       0       1       pass
U101    S204    A301    1475130523      1485130523      1485130523      A       ST411   2       0       1       1       pass
U117    S204    A301    1495130523      1465130523      1475130523      A       ST402   0       1       0       1       pass
U113    S205    A305    1462863262      1465490556      1462863262      AP      ST407   3       0       1       1       pass
U120    S205    A302    1494297562      1494297562      1494297562      A       ST400   0       1       0       1       pass
U117    S206    A300    1468094889      1468094889      1465490556      A       ST414   2       0       0       1       pass
U111    S206    A303    1465230523      1485130523      1465130523      U       ST414   1       0       0       1       pass
U113    S207    A305    1495130523      1465130523      1485130523      AU      ST402   0       0       1       1       pass
U117    S208    A305    1465130523      1465130523      1465130523      AP      ST407   3       0       1       1       pass
U110    S208    A303    1494297562      1468094889      1468094889      E       ST405   1       0       1       1       pass
U105    S208    A300    1465130523      1475130523      1465230523      AU      ST410   1       0       0       1       pass
U103    S209    A305    1465490556      1468094889      1468094889      AU      ST408   3       0       1       1       pass
U118    S209    A305    1475130523      1465230523      1465230523      E       ST400   0       0       0       1       pass
Time taken: 3.266 seconds, Fetched: 40 row(s)
hive>
```

By applying the provided rules, we have successfully accomplished Data enrichment and Filtering stage.

Data Analysis using Spark

In this stage we will do analysis on enriched data using Spark SQL and run the program using Spark Submit command.

Before running the spark-submit command we have to zip –d command to remove the bad manifests in created spark project jar file to avoid the invalid Signature exception. We used two spark-submits for analysis.

a. Spark_analysis for creating tables for each query/problem statement.
b. Spark_analysis_2 for displaying results for each query in terminal.

DataAnalysis.sh

```
#!/bin/bash

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

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

echo "Running script for data analysis using spark..." >> $LOGFILE
chmod 775 /home/acadgild/project/lib/sparkanalysis.jar

zip -d /home/acadgild/project/lib/sparkanalysis.jar META-INF/*.DSA META-INF/*.RSA META-INF/*.SF

/home/acadgild/spark-2.2.1-bin-hadoop2.7/bin/spark-submit    \
--class Spark_analysis \
--master local[2] \
--driver-class-path /home/acadgild/apache-hive-2.1.0-bin/lib/hive-hbase-handler-2.1.0.jar:/home/acadgild/hbase-1.0.3/lib/* \
/home/acadgild/project/lib/sparkanalysis.jar $batchid

/home/acadgild/spark-2.2.1-bin-hadoop2.7/bin/spark-submit    \
--class Spark_analysis_2 \
--master local[2] \
--driver-class-path /home/acadgild/apache-hive-2.1.0-bin/lib/hive-hbase-handler-2.1.0.jar:/home/acadgild/hbase-1.0.3/lib/* \
/home/acadgild/project/lib/sparkanalysis.jar $batchid

echo "Exporting data to MYSQL using sqoop export..." >> $LOGFILE
sh /home/acadgild/project/scripts/data_export.sh

echo "Incrementing batchid..." >> $LOGFILE
batchid=`expr $batchid + 1`
echo -n $batchid > /home/acadgild/project/logs/current-batch.txt
```

Spark_analysis.scala

```
import org.apache.hadoop.hive.serde2.`lazy`.LazySimpleSerDe
import org.apache.spark.sql.SparkSession

object Spark_analysis {

  def main(args: Array[String]): Unit = {
    val sparkSession = SparkSession.builder()
      .master("local[2]")
      .appName("Data Analysis Main_1")
      .config("spark.sql.warehouse.dir","/user/hive/warehouse")
      .config("hive.metastore.uris","thrift://127.0.0.1:9083")
      .enableHiveSupport()
      .getOrCreate()

    val batchId = args(0)

    //<<<<<<<<<<---------- PROBLEM 1 - Creation of table and Insertion of data ------------>>>>>>>>>>>>
    //Determine top 10 station_id(s) where maximum number of songs were played, which were liked by unique users.

    val set_properties = sparkSession.sqlContext.sql("set hive.auto.convert.join=false")

    val use_project_database = sparkSession.sqlContext.sql("USE project")

    val create_hive_table_top_10_stations = sparkSession.sqlContext.sql("CREATE TABLE IF NOT EXISTS project.top_10_stations"+
      "("+
      " station_id STRING,"+
      " total_distinct_songs_played INT,"+
      " distinct_user_count INT"+
      ")"+
      " PARTITIONED BY (batchid INT)"+
      " ROW FORMAT DELIMITED"+
      " FIELDS TERMINATED BY ','"+
      " STORED AS TEXTFILE")
```

```
val insert_into_top_10_stations = sparkSession.sqlContext.sql("INSERT OVERWRITE TABLE project.top_10_stations"+
  s" PARTITION (batchid=$batchId)"+
  " SELECT"+
  " station_id,"+
  " COUNT(DISTINCT song_id) AS total_distinct_songs_played,"+
  " COUNT(DISTINCT user_id) AS distinct_user_count"+
  " FROM project.enriched_data"+
  " WHERE status='pass'"+
  s" AND (batchid=$batchId)"+
  " AND like=1"+
  " GROUP BY station_id"+
  " ORDER BY total_distinct_songs_played DESC"+
  " LIMIT 10")

//<<<<<<<<<<---------- PROBLEM 2 - Creation of table and Insertion of data ---------->>>>>>>>>>
/*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.*/

val create_hive_table_song_duration = sparkSession.sqlContext.sql("CREATE TABLE IF NOT EXISTS project.song_duration"+
  "("+
  " user_id STRING,"+
  " user_type STRING,"+
  " song_id STRING,"+
  " artist_id STRING,"+
  " total_duration_in_minutes DOUBLE"+
  ")"+
  " PARTITIONED BY (batchid INT)"+
  " ROW FORMAT DELIMITED"+
  " FIELDS TERMINATED BY ','"+
  " STORED AS TEXTFILE")
```

```
val insert_into_song_duration = sparkSession.sqlContext.sql("INSERT OVERWRITE TABLE project.song_duration"+
  s" PARTITION (batchid=$batchId)"+
  " SELECT"+
  " e.user_id STRING,"+
  " IF(e.user_id!=s.user_id"+
  " OR (CAST(s.subscn_end_dt as BIGINT) < CAST(e.start_ts as BIGINT)),'unsubscribed','subscribed') AS user_type,"+
  " e.song_id STRING,"+
  " e.artist_id STRING,"+
  " (cast(e.end_ts as BIGINT)-cast(e.start_ts as BIGINT))/60 AS total_duration_in_minutes"+
  " FROM project.enriched_data e"+
  " LEFT OUTER JOIN project.subscribed_users s"+
  " ON e.user_id=s.user_id"+
  " WHERE e.status='pass'"+
  s" AND (batchid=$batchId)")


//<<<<<<<<<<---------- PROBLEM 3 - Creation of table and Insertion of data ---------->>>>>>>>>>
//Determine top 10 connected artists.
//Connected artists are those whose songs are most listened by the unique users who follow them.

val create_hive_table_top_10_connected_artists = sparkSession.sqlContext.sql("CREATE TABLE IF NOT EXISTS project.connected_artists"+
  "("+
  " artist_id STRING,"+
  " total_distinct_songs INT,"+
  " unique_followers INT"+
  ")"+
  " PARTITIONED BY (batchid INT)"+
  " ROW FORMAT DELIMITED"+
  " FIELDS TERMINATED BY ','"+
  " STORED AS TEXTFILE")
```

```
val insert_into_top_10_connected_artists = sparkSession.sqlContext.sql("INSERT OVERWRITE TABLE project.connected_artists"+
  s" PARTITION (batchid=$batchId)"+
  " SELECT"+
  " artist_id,"+
  " COUNT(DISTINCT song_id) AS total_distinct_songs,"+
  " COUNT(DISTINCT user_id) AS unique_followers"+
  " FROM project.enriched_data"+
  " WHERE status='pass'"+
  s" AND (batchid=$batchId)"+
  " GROUP BY artist_id"+
  " ORDER BY unique_followers desc,total_distinct_songs desc"+
  " LIMIT 10")


//<<<<<<<<<<---------- PROBLEM 4 - Creation of table and Insertion of data ---------->>>>>>>>>>
//Determine top 10 songs who have generated the maximum revenue.
//NOTE: Royalty applies to a song only if it was liked or was completed successfully or both.

val create_hive_table_top_10_songs_maxrevenue = sparkSession.sqlContext.sql("CREATE TABLE IF NOT EXISTS project.top_10_songs_maxrevenue"+
  "("+
  " song_id STRING,"+
  " artist_id STRING,"+
  " total_duration_in_minutes DOUBLE"+
  " )"+
  " PARTITIONED BY (batchid INT)"+
  " ROW FORMAT DELIMITED"+
  " FIELDS TERMINATED BY ','"+
  " STORED AS TEXTFILE")
```

```
    val insert_into_top_10_songs_maxrevenue = sparkSession.sqlContext.sql("INSERT OVERWRITE TABLE project.top_10_songs_maxrevenue"+
      s" PARTITION (batchid=$batchId)"+
      " SELECT"+
      " song_id,"+
      " artist_id,"+
      " (cast(end_ts as BIGINT)-cast(start_ts as BIGINT))/60 AS total_duration_in_minutes"+
      " FROM project.enriched_data"+
      " WHERE status='pass'" +
      s" AND (batchid=$batchId)"+
      " AND (like=1 OR song_end_type=0 OR (like=1 and song_end_type=0))"+
      " ORDER BY total_duration_in_minutes desc"+
      " LIMIT 10")


    //<<<<<<<<<<---------- PROBLEM 5 - Creation of table and Insertion of data ------------>>>>>>>>>>>>
    //Determine top 10 unsubscribed users who listened to the songs for the longest duration.

    val create_hive_table_top_10_unsubscribed_users = sparkSession.sqlContext.sql("CREATE TABLE IF NOT EXISTS project.top_10_unsubscribed_users"+
      "("+
      " user_id STRING,"+
      " song_id STRING,"+
      " artist_id STRING,"+
      " total_duration_in_minutes DOUBLE"+
      ")"+
      " PARTITIONED BY (batchid INT)"+
      " ROW FORMAT DELIMITED"+
      " FIELDS TERMINATED BY ','"+
      " STORED AS TEXTFILE")
```

```
    //<<<<<<<<<<---------- PROBLEM 5 - Creation of table and Insertion of data ------------>>>>>>>>>>>>
    //Determine top 10 unsubscribed users who listened to the songs for the longest duration.

    val create_hive_table_top_10_unsubscribed_users = sparkSession.sqlContext.sql("CREATE TABLE IF NOT EXISTS project.top_10_unsubscribed_users"+
      "("+
      " user_id STRING,"+
      " song_id STRING,"+
      " artist_id STRING,"+
      " total_duration_in_minutes DOUBLE"+
      ")"+
      " PARTITIONED BY (batchid INT)"+
      " ROW FORMAT DELIMITED"+
      " FIELDS TERMINATED BY ','"+
      " STORED AS TEXTFILE")


    val insert_into_unsubscribed_users = sparkSession.sqlContext.sql("INSERT OVERWRITE TABLE project.top_10_unsubscribed_users"+
      s" PARTITION (batchid=$batchId)"+
      " SELECT"+
      " user_id,"+
      " song_id,"+
      " artist_id,"+
      " total_duration_in_minutes"+
      " FROM project.song_duration"+
      " WHERE user_type='unsubscribed'"+
      " AND total_duration_in_minutes>=0"+
      s" AND (batchid=$batchId)"+
      " ORDER BY total_duration_in_minutes desc"+
      " LIMIT 10")


  }
}
```

Spark_analysis_2.scala

```scala
package sparkanalysis

import org.apache.spark.{SparkConf,SparkContext}
import org.apache.spark.sql.SparkSession
object Spark_analysis_2 {
  def main(args: Array[String]): Unit = {
    val sparkSession = SparkSession.builder.master("local").appName("Spark Session example")
      .config("spark.sql.warehouse.dir", "/user/hive/warehouse")
      .config("hive.metastore.uris", "thrift://localhost:9083")
      .enableHiveSupport().getOrCreate()
    val batchId = args(0)

    sparkSession.sqlContext.sql("USE project")
    sparkSession.sqlContext.sql("SELECT station_id from top_10_stations").show()
    sparkSession.sqlContext.sql("SELECT user_type,total_duration_in_minutes from song_duration").show()
    sparkSession.sqlContext.sql("SELECT artist_id from connected_artists").show()
    sparkSession.sqlContext.sql("SELECT song_id from top_10_songs_maxrevenue").show()
    sparkSession.sqlContext.sql("SELECT user_id from top_10_unsubscribed_users").show()


  }
}
```

```
[acadgild@localhost scripts]$ sh DataAnalysis.sh
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/hbase-1.0.3/lib/slf4j-log4j12-1.7.7.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/spark-2.2.1-bin-hadoop2.7/jars/slf4j-log4j12-1.7.16.jar!/org/slf4j/impl/StaticLoggerBinder.cla
ss]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
18/01/22 11:38:36 INFO SparkContext: Running Spark version 2.2.1
18/01/22 11:38:37 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
18/01/22 11:38:38 WARN Utils: Your hostname, localhost.localdomain resolves to a loopback address: 127.0.0.1; using 10.0.2.15 instead (on inter
face eth0)
18/01/22 11:38:38 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
18/01/22 11:38:38 INFO SparkContext: Submitted application: Data Analysis Main_1
18/01/22 11:38:38 INFO SecurityManager: Changing view acls to: acadgild
18/01/22 11:38:38 INFO SecurityManager: Changing modify acls to: acadgild
18/01/22 11:38:38 INFO SecurityManager: Changing view acls groups to:
18/01/22 11:38:38 INFO SecurityManager: Changing modify acls groups to:
18/01/22 11:38:38 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissions: Set(acadgild)
; groups with view permissions: Set(); users  with modify permissions: Set(acadgild); groups with modify permissions: Set()
18/01/22 11:38:39 INFO Utils: Successfully started service 'sparkDriver' on port 52909.
18/01/22 11:38:39 INFO SparkEnv: Registering MapOutputTracker
18/01/22 11:38:39 INFO SparkEnv: Registering BlockManagerMaster
18/01/22 11:38:39 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
18/01/22 11:38:39 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
18/01/22 11:38:39 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-0343769b-ba6b-4405-b875-d4935ce4d38d
18/01/22 11:38:40 INFO MemoryStore: MemoryStore started with capacity 413.9 MB
18/01/22 11:38:40 INFO SparkEnv: Registering OutputCommitCoordinator
18/01/22 11:38:40 INFO Utils: Successfully started service 'SparkUI' on port 4040.
18/01/22 11:38:41 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://10.0.2.15:4040
18/01/22 11:38:41 INFO SparkContext: Added JAR file:/home/acadgild/project/lib/sparkanalysis.jar at spark://10.0.2.15:52909/jars/sparkanalysis.
jar with timestamp 1516601321290
18/01/22 11:38:41 INFO Executor: Starting executor ID driver on host localhost
18/01/22 11:38:41 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 60285.
```

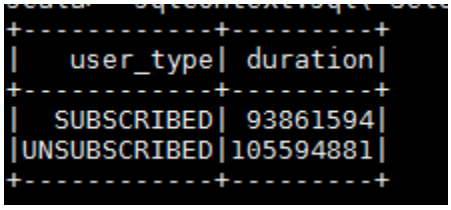
```
Time taken: 1.729 seconds
OK
Time taken: 1.437 seconds
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e.
spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180122102731_b67f3e6a-6470-44f7-bc68-6e98ef39b68d
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1516485910189_0019, Tracking URL = http://localhost:8088/proxy/application_1516485910189_0019/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job  -kill job_1516485910189_0019
Hadoop job information for Stage-1: number of mappers: 0; number of reducers: 1
2018-01-22 10:28:08,714 Stage-1 map = 0%,  reduce = 0%
2018-01-22 10:28:24,894 Stage-1 map = 0%,  reduce = 100%, Cumulative CPU 2.05 sec
MapReduce Total cumulative CPU time: 2 seconds 50 msec
Ended Job = job_1516485910189_0019
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1516485910189_0020, Tracking URL = http://localhost:8088/proxy/application_1516485910189_0020/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job  -kill job_1516485910189_0020
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-01-22 10:28:49,750 Stage-2 map = 0%,  reduce = 0%
2018-01-22 10:29:08,673 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 3.85 sec
2018-01-22 10:29:29,196 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 7.65 sec
MapReduce Total cumulative CPU time: 7 seconds 650 msec
Ended Job = job_1516485910189_0020
Loading data to table project.top_10_stations partition (batchid=7)
```

```
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1516485910189_0021, Tracking URL = http://localhost:8088/proxy/application_1516485910189_0021/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job  -kill job_1516485910189_0021
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-01-22 10:30:06,890 Stage-1 map = 0%,  reduce = 0%
2018-01-22 10:30:30,036 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 5.88 sec
2018-01-22 10:30:47,380 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 9.78 sec
MapReduce Total cumulative CPU time: 9 seconds 780 msec
Ended Job = job_1516485910189_0021
Launching Job 2 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1516485910189_0022, Tracking URL = http://localhost:8088/proxy/application_1516485910189_0022/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job  -kill job_1516485910189_0022
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-01-22 10:31:15,265 Stage-2 map = 0%,  reduce = 0%
2018-01-22 10:31:26,602 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.92 sec
2018-01-22 10:31:42,859 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 5.33 sec
MapReduce Total cumulative CPU time: 5 seconds 330 msec
Ended Job = job_1516485910189_0022
Loading data to table project.users_behaviour partition (batchid=7)
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 9.78 sec   HDFS Read: 21286 HDFS Write: 96 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 5.33 sec   HDFS Read: 6760 HDFS Write: 59 SUCCESS
Total MapReduce CPU Time Spent: 15 seconds 110 msec
OK
Time taken: 132.863 seconds
OK
Time taken: 0.138 seconds
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e.
```

Query-1: Determine top 10 **station id(s)** where maximum number of songs were played, which were liked by unique users.

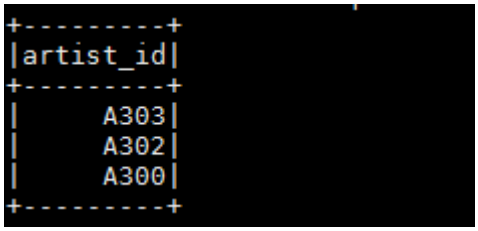
```
scala> val result
+----------+
|station_id|
+----------+
|     ST407|
|     ST414|
|     ST411|
|     ST402|
|     ST406|
|     ST405|
+----------+
```

Query-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.



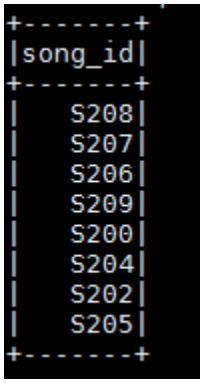
user_type	duration
SUBSCRIBED	93861594
UNSUBSCRIBED	105594881

Query-3: Determine top 10 connected artists. Connected artists are those whose songs are most listened by the unique users who follow them



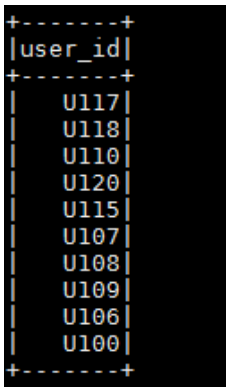
artist_id
A303
A302
A300

Query-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



song_id
S208
S207
S206
S209
S200
S204
S202
S205

Query-5: Determine top **10 unsubscribed** users who listened to the songs for the longest duration.



user_id
U117
U118
U110
U120
U115
U107
U108
U109
U106
U100

Table Creation in HIVE and Data analysis using HIVE,

```
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1516485910189_0029, Tracking URL = http://localhost:8088/proxy/application_1516485910189_0029/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job  -kill job_1516485910189_0029
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-01-22 10:37:36,483 Stage-2 map = 0%,  reduce = 0%
2018-01-22 10:37:50,781 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 2.26 sec
2018-01-22 10:38:06,474 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 4.34 sec
MapReduce Total cumulative CPU time: 4 seconds 340 msec
Ended Job = job_1516485910189_0029
Launching Job 3 out of 3
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1516485910189_0030, Tracking URL = http://localhost:8088/proxy/application_1516485910189_0030/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job  -kill job_1516485910189_0030
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 1
2018-01-22 10:38:30,610 Stage-3 map = 0%,  reduce = 0%
2018-01-22 10:38:41,945 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 1.8 sec
2018-01-22 10:38:56,880 Stage-3 map = 100%,  reduce = 100%, Cumulative CPU 4.73 sec
MapReduce Total cumulative CPU time: 4 seconds 730 msec
Ended Job = job_1516485910189_0030
Loading data to table project.top_10_unsubscribed_users partition (batchid=7)
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 9.36 sec   HDFS Read: 14798 HDFS Write: 96 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 4.34 sec   HDFS Read: 5072 HDFS Write: 96 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 4.73 sec   HDFS Read: 6582 HDFS Write: 69 SUCCESS
Total MapReduce CPU Time Spent: 18 seconds 430 msec
OK
Time taken: 177.031 seconds
```

The tables have also been created in the Hive,

```
hive>
    > use project;
OK
Time taken: 1.522 seconds
hive> show tables;
OK
connected_artists
enriched_data
formatted_input
song_artist_map
station_geo_map
subscribed_users
top_10_royalty_songs
top_10_stations
top_10_unsubscribed_users
users_artists
users_behaviour
Time taken: 0.612 seconds, Fetched: 11 row(s)
hive>
```

We have seen all the spark queries creating the tables for each query. So, Data Analysis using Spark is executed successfully.

The data analysis result is shown in the Hive tables below in the screen shot,

Output from, connected_artists, top_10_royalty_songs, top_10_stations.

```
Time taken: 0.097 seconds, Fetched: 11 row(s)
hive> Select * From connected_artists;
OK
connected_artists.artist_id     connected_artists.user_count    connected_artists.batchid
A303    2       1
A302    2       1
A300    1       1
Time taken: 0.225 seconds, Fetched: 3 row(s)
hive> Select * From top_10_royalty_songs;
OK
top_10_royalty_songs.song_id    top_10_royalty_songs.duration   top_10_royalty_songs.batchid
S208    22627294        1
S207    20000000        1
S206    19900000        1
S209    15254588        1
S200    9900000 1
S204    2604333 1
S202    100000  1
S205    0       1
Time taken: 0.237 seconds, Fetched: 8 row(s)
hive> Select * From top_10_stations;
OK
top_10_stations.station_id      top_10_stations.total_distinct_songs_played     top_10_stations.distinct_user_count     top_10_stations.batchid
ST407   2       3       1
ST414   1       1       1
ST411   1       1       1
ST402   1       2       1
ST406   1       1       1
ST405   1       1       1
Time taken: 0.336 seconds, Fetched: 6 row(s)
hive> Select * From top_10_unsubscribed_users;
OK
top_10_unsubscribed_users.user_id       top_10_unsubscribed_users.duration      top_10_unsubscribed_users.batchid
U117    20000000        1
U118    20000000        1
U110    20000000        1
U120    12627294        1
U115    12527294        1
```

Output from top_10_unsubscribed_users, users behavior.

```
Time taken: 0.237 seconds, Fetched: 8 row(s)
hive> Select * From top_10_stations;
OK
top_10_stations.station_id      top_10_stations.total_distinct_songs_played     top_10_stations.distinct_user_count     top_10_stations.batchid
ST407   2       3       1
ST414   1       1       1
ST411   1       1       1
ST402   1       2       1
ST406   1       1       1
ST405   1       1       1
Time taken: 0.336 seconds, Fetched: 6 row(s)
hive> Select * From top_10_unsubscribed_users;
OK
top_10_unsubscribed_users.user_id       top_10_unsubscribed_users.duration      top_10_unsubscribed_users.batchid
U117    20000000        1
U118    20000000        1
U110    20000000        1
U120    12627294        1
U115    12527294        1
U107    10000000        1
U108    5231627 1
U109    2604333 1
U106    2604333 1
U100    0       1
Time taken: 0.275 seconds, Fetched: 10 row(s)
hive> Select * From users_behaviour;
OK
users_behaviour.user_type       users_behaviour.duration        users_behaviour.batchid
SUBSCRIBED      93861594        1
UNSUBSCRIBED    105594881       1
Time taken: 0.274 seconds, Fetched: 2 row(s)
hive>
    >
    >
```

Now, we need to export all the data to the **MYSQL** using Sqoop, run the script **data_export.sh,**

Data Storage in MYSQL

Using the bash file shown below, **data_export.sh** we are going to export the data from the hive tables into MySQL using **Sqoop** export.

```
#!/bin/bash

#This script is not working.
#Either change table to text or use STRING as type of partitioned column

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

echo "Creating mysql tables if not present..." >> $LOGFILE

mysql < /home/acadgild/project/scripts/create_schema.sql

echo "Running sqoop job for data export..." >> $LOGFILE

sqoop export --connect jdbc:mysql://localhost/project --username root --password acadgild --table top_10_stations --export-dir
hdfs://localhost:9000/user/hive/warehouse/project.db/top_10_stations/batchid=$batchid --input-fields-terminated-by ',' -m 1

sqoop export --connect jdbc:mysql://localhost/project --username root --password acadgild --table users_behaviour  --export-dir
hdfs://localhost:9000/user/hive/warehouse/project.db/users_behaviour/batchid=$batchid --input-fields-terminated-by ',' -m 1

sqoop export --connect jdbc:mysql://localhost/project --username root --password acadgild --table connected_artists --export-dir
hdfs://localhost:9000/user/hive/warehouse/project.db/connected_artists/batchid=$batchid --input-fields-terminated-by ',' -m 1

sqoop export --connect jdbc:mysql://localhost/project --username root --password acadgild --table top_10_royalty_songs --export-dir
hdfs://localhost:9000/user/hive/warehouse/project.db/top_10_royalty_songs/batchid=$batchid --input-fields-terminated-by ',' -m 1

sqoop export --connect jdbc:mysql://localhost/project --username root --password acadgild --table top_10_unsubscribed_users --export-dir
hdfs://localhost:9000/user/hive/warehouse/project.db/top_10_unsubscribed_users/batchid=$batchid --input-fields-terminated-by ',' -m 1

```

Create schema.sql – Make sure that you logged in to MySQL. The below schema will create the database and tables in the MySQL.

```
CREATE DATABASE IF NOT EXISTS project;

USE project;

CREATE TABLE IF NOT EXISTS top_10_stations
(
station_id VARCHAR(50),
total_distinct_songs_played INT,
distinct_user_count INT
);

CREATE TABLE IF NOT EXISTS users_behaviour
(
user_type VARCHAR(50),
duration BIGINT
);

CREATE TABLE IF NOT EXISTS connected_artists
(
artist_id VARCHAR(50),
user_count INT
);

CREATE TABLE IF NOT EXISTS top_10_royalty_songs
(
song_id VARCHAR(50),
duration BIGINT
);

CREATE TABLE IF NOT EXISTS top_10_unsubscribed_users
(
user_id VARCHAR(50),
duration BIGINT
);

commit;
```

Now we can see the data exported successfully into the MYSQL Database for all the 5 queries.

```
 stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/24 09:57:24 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
Found 1 items
drwxrwxr-x   - acadgild supergroup          0 2018-01-24 09:34 hdfs://localhost:9000/user/hive/warehouse/project.db/top_10_stations/batchid=1
[acadgild@localhost ~]$ sqoop export --connect jdbc:mysql://localhost/project --username root --password acadgild --table top_10_stations --exp
ort-dir hdfs://localhost:9000/user/hive/warehouse/project.db/top_10_stations/batchid=1 --input-fields-terminated-by ',' -m 1
Warning: /home/acadgild/sqoop-1.4.6.bin__hadoop-2.0.4-alpha/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /home/acadgild/sqoop-1.4.6.bin__hadoop-2.0.4-alpha/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /home/acadgild/sqoop-1.4.6.bin__hadoop-2.0.4-alpha/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
2018-01-24 09:58:23,657 INFO  [main] sqoop.Sqoop: Running Sqoop version: 1.4.6
2018-01-24 09:58:23,694 WARN  [main] tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
2018-01-24 09:58:24,084 INFO  [main] manager.MySQLManager: Preparing to use a MySQL streaming resultset.
2018-01-24 09:58:24,085 INFO  [main] tool.CodeGenTool: Beginning code generation
2018-01-24 09:58:24,696 INFO  [main] manager.SqlManager: Executing SQL statement: SELECT t.* FROM `top_10_stations` AS t LIMIT 1
2018-01-24 09:58:24,767 INFO  [main] manager.SqlManager: Executing SQL statement: SELECT t.* FROM `top_10_stations` AS t LIMIT 1
2018-01-24 09:58:24,810 INFO  [main] orm.CompilationManager: HADOOP_MAPRED_HOME is /home/acadgild/hadoop-2.7.2
Note: /tmp/sqoop-acadgild/compile/f1ae2653abc7121116a39d1b97e8735b/top_10_stations.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
2018-01-24 09:58:29,735 INFO  [main] orm.CompilationManager: Writing jar file: /tmp/sqoop-acadgild/compile/f1ae2653abc7121116a39d1b97e8735b/top
_10_stations.jar
2018-01-24 09:58:29,763 INFO  [main] mapreduce.ExportJobBase: Beginning export of top_10_stations
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/hbase-1.0.3/lib/slf4j-log4j12-1.7.7.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/hadoop-2.7.2/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBind
er.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which might have disabled
 stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
2018-01-24 09:58:30,180 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classe
s where applicable
2018-01-24 09:58:30,199 INFO  [main] Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job.jar
2018-01-24 09:58:32,079 INFO  [main] Configuration.deprecation: mapred.reduce.tasks.speculative.execution is deprecated. Instead, use mapreduce
```

The Sqoop export command exported the tables from the hive and it stored in the MySQL. The below screen shot shows the successful Sqoop export from hive to MySQL. The data stored in the MySQL is shown in the successive screen shots,

```
2018-01-24 10:06:14,238 INFO  [main] input.FileInputFormat: Total input paths to process : 1
2018-01-24 10:06:14,435 INFO  [main] mapreduce.JobSubmitter: number of splits:1
2018-01-24 10:06:14,464 INFO  [main] Configuration.deprecation: mapred.map.tasks.speculative.execution is deprecated. Instead, use mapreduce.ma
p.speculative
2018-01-24 10:06:14,791 INFO  [main] mapreduce.JobSubmitter: Submitting tokens for job: job_1516764714140_0018
2018-01-24 10:06:16,203 INFO  [main] impl.YarnClientImpl: Submitted application application_1516764714140_0018
2018-01-24 10:06:16,420 INFO  [main] mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1516764714140_0018/
2018-01-24 10:06:16,424 INFO  [main] mapreduce.Job: Running job: job_1516764714140_0018
2018-01-24 10:06:45,186 INFO  [main] mapreduce.Job: Job job_1516764714140_0018 running in uber mode : false
2018-01-24 10:06:45,191 INFO  [main] mapreduce.Job:  map 0% reduce 0%
2018-01-24 10:07:01,634 INFO  [main] mapreduce.Job:  map 100% reduce 0%
2018-01-24 10:07:02,707 INFO  [main] mapreduce.Job: Job job_1516764714140_0018 completed successfully
2018-01-24 10:07:03,246 INFO  [main] mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=136426
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=311
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=4
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=0
        Job Counters
                Launched map tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=12452
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=12452
                Total vcore-seconds taken by all map tasks=12452
                Total megabyte-seconds taken by all map tasks=12750848
        Map-Reduce Framework
                Map input records=10
                Map output records=10
                Input split bytes=178
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
```

The data base ***project*** had been exported from the hive and the below screen shot shows the data base presence, output from **top_10_stations, connected artists** shown below,

```
mysql>
mysql> use project;
Database changed
mysql> show tables;
+---------------------------+
| Tables_in_project         |
+---------------------------+
| connected_artists         |
| top_10_royalty_songs      |
| top_10_stations           |
| top_10_unsubscribed_users |
| users_behaviour           |
+---------------------------+
5 rows in set (0.00 sec)

mysql> Select * From top_10_stations;
+------------+-----------------------------+---------------------+
| station_id | total_distinct_songs_played | distinct_user_count |
+------------+-----------------------------+---------------------+
| ST407      |                           2 |                   3 |
| ST414      |                           1 |                   1 |
| ST411      |                           1 |                   1 |
| ST402      |                           1 |                   2 |
| ST406      |                           1 |                   1 |
| ST405      |                           1 |                   1 |
+------------+-----------------------------+---------------------+
6 rows in set (0.00 sec)

mysql> Select * From connected_artists;
+-----------+------------+
| artist_id | user_count |
+-----------+------------+
| A303      |          2 |
| A302      |          2 |
| A300      |          1 |
+-----------+------------+
3 rows in set (0.00 sec)
```

top_10_royalty_songs,

```
mysql> Select * From top_10_royalty_songs;
+---------+----------+
| song_id | duration |
+---------+----------+
| S208    | 22627294 |
| S207    | 20000000 |
| S206    | 19900000 |
| S209    | 15254588 |
| S200    |  9900000 |
| S204    |  2604333 |
| S202    |   100000 |
| S205    |        0 |
+---------+----------+
8 rows in set (0.00 sec)
```

Output from **top_10_unsubscribed_users** and **users behavior**

```
mysql> Select * From top_10_unsubscribed_users;
+---------+----------+
| user_id | duration |
+---------+----------+
| U117    | 20000000 |
| U118    | 20000000 |
| U110    | 20000000 |
| U120    | 12627294 |
| U115    | 12527294 |
| U107    | 10000000 |
| U108    |  5231627 |
| U109    |  2604333 |
| U106    |  2604333 |
| U100    |        0 |
+---------+----------+
10 rows in set (0.01 sec)

mysql> Select * From users_behaviour;
+--------------+-----------+
| user_type    | duration  |
+--------------+-----------+
| SUBSCRIBED   |  93861594 |
| UNSUBSCRIBED | 105594881 |
+--------------+-----------+
2 rows in set (0.00 sec)
```

Job Scheduling:

Now after exporting data into MySQL **batchid** will be incremented to additional 1 means one batch of data operations is successfully completed and new batch of data will be loaded for the analysis after every 3 hours.

```
21  --driver-class-path /home/acadgild/apache-hive-2.1.0-bin/lib/hive-hbase-handler-
22  /home/acadgild/project/lib/sparkanalysis.jar $batchid
23
24  echo "Exporting data to MYSQL using sqoop export..." >> $LOGFILE
25  sh /home/acadgild/project/scripts/data_export.sh
26
27  echo "Incrementing batchid..." >> $LOGFILE
28  batchid=`expr $batchid + 1`
29  echo -n $batchid > /home/acadgild/project/logs/current-batch.txt
30
```

We can check logs to track the behavior of the operations we have done on the data and overcome failures in the pipeline and we can see the **batchid** incremented value in **current-batch.txt**

```
[acadgild@localhost project]$ cd logs
[acadgild@localhost logs]$ ls -l
total 24
-rwxrwxr-x. 1 acadgild acadgild    1 Jan 24 09:44 current-batch.txt
-rw-rw-r--. 1 acadgild acadgild  679 Jan 24 09:03 derby.log
drwxrwxr-x. 3 acadgild acadgild 4096 Jan 24 09:02 hdfs:
-rw-rw-r--. 1 acadgild acadgild  523 Jan 24 09:44 log_batch_1
-rw-rw-r--. 1 acadgild acadgild   77 Jan 24 09:44 log_batch_1???
drwxrwxr-x. 5 acadgild acadgild 4096 Jan 24 09:03 metastore_db
[acadgild@localhost logs]$ cat current-batch.txt
2[acadgild@localhost logs]$
[acadgild@localhost logs]$
[acadgild@localhost logs]$
```

The log file captured all the data and steps we performed so far,

```
[acadgild@localhost logs]$ cat log_batch_1
Starting daemons
Creating LookUp Tables
Populating LookUp Tables
Creating hive tables on top of hbase tables for data enrichment and filtering...
Placing data files from local to HDFS...
Running pig script for data formatting...
Running hive script for formatted data load...
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 hive script for data analysis...
Incrementing batchid...
[acadgild@localhost logs]$
```

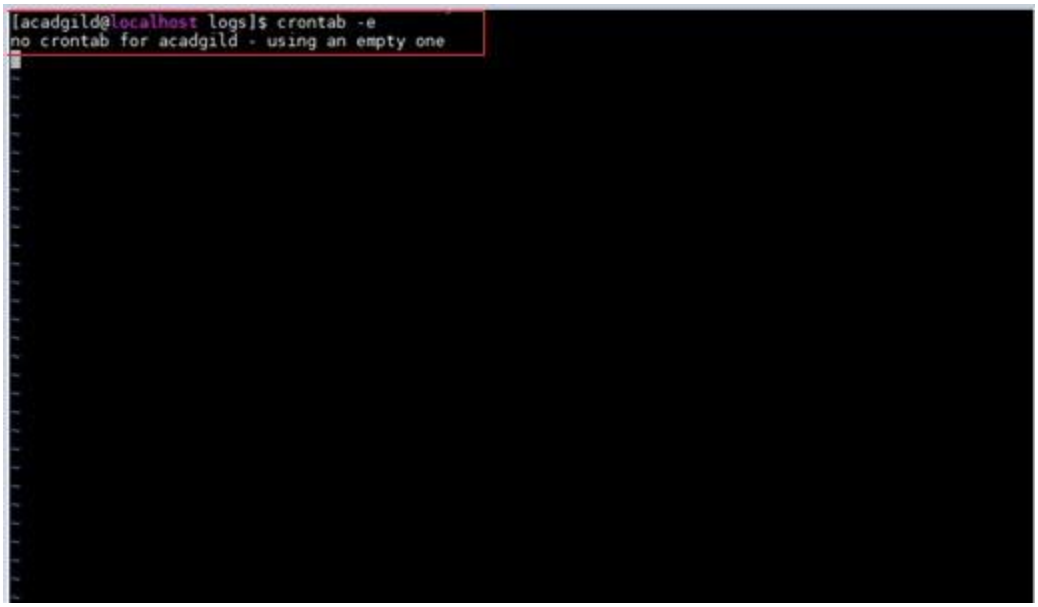
Wrapping all the scripts inside the single script file and scheduling this file to run at the periodic interval of every 3 hours.

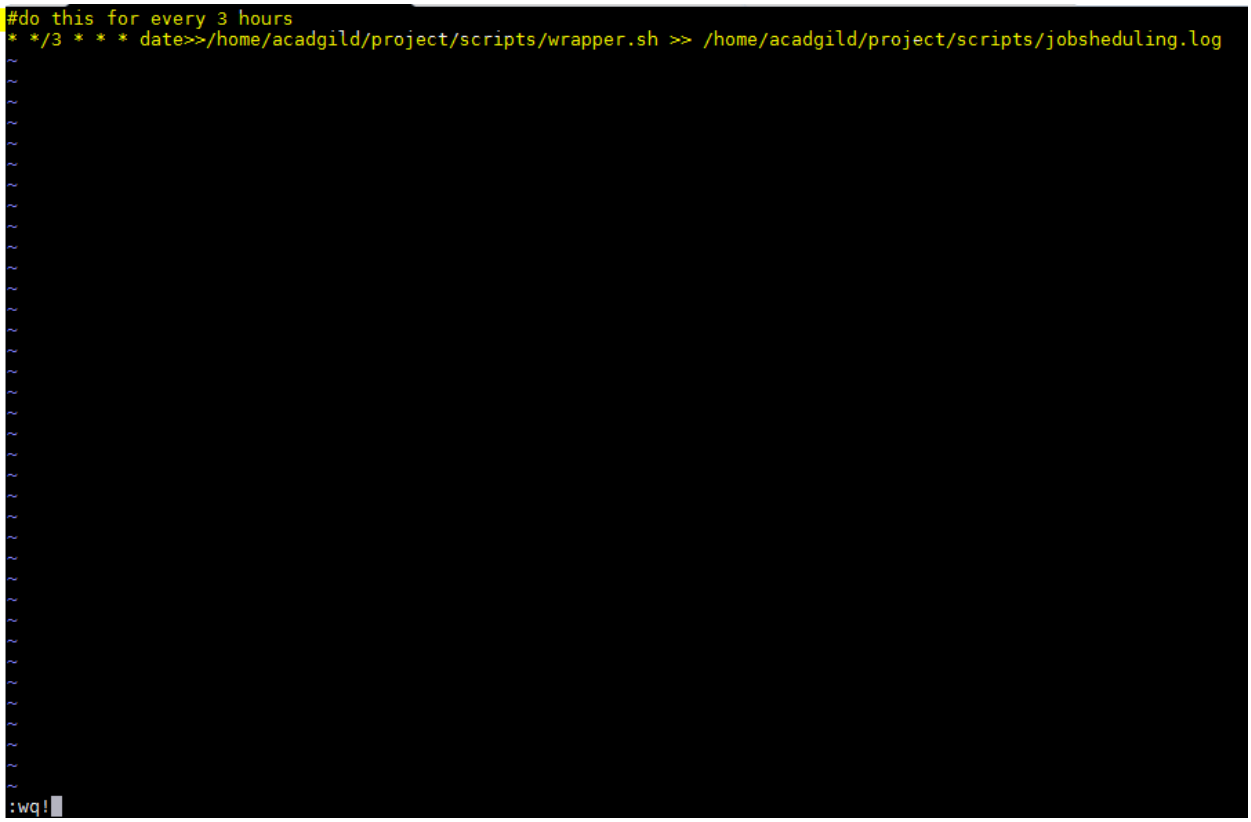
wrapper.sh

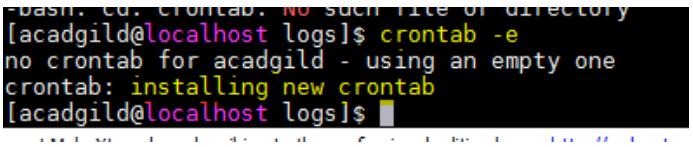
```
#!/bin/bash
#All the below scripts will work based on the data provided by acadgild as data/web/file.xml and data/mob/file.txt

python /home/acadgild/project/scripts/generate_web_data.py

python /home/acadgild/project/scripts/generate_mob_data.py

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.sh

sh /home/acadgild/project/scripts/data_analysis.sh

```

The wrapper.sh will be running for every 3 hours as per the job scheduling done below, as per the above order the wrapper.sh will run the scripts.

Creating **Crontab** to schedule the wrapper.sh script to run for every 3-hour interval.







*Installing the **crontab** in the vm,*

The **crontab** job scheduler will run the **wrappr.sh** every 3 hours and for every 3 hours we will get incremental batch ID's. **Hence, as per the request this job scheduling has been done.**

```
Deleting older valid and invalid records from local file system...
Running hive script for data analysis...
Incrementing batchid...
[acadgild@localhost logs]$ cd
[acadgild@localhost ~]$ crontab -l
#do this for every 3 hours
* */3 * * * date>>/home/acadgild/project/scripts/wrapper.sh >> /home/acadgild/project/scripts/jobsheduling.log
[acadgild@localhost ~]$
[acadgild@localhost ~]$
```

Highlights of the Project

- No join of query is used while analysis. Data is already enriched with new fields and using broadcast maps on Lookup tables so as to avoid any join.
- We used full automated bash scripts from start to end.

Project End Conclusion:

So, we performed all the data operations as per the sequence mentioned in the **wrapper.sh** file and obtained results successfully for the one of the leading music company.