FINAL PROJECT: MUSIC DATA ANALYSIS

Project Description: A leading music-catering company is planning to analyse large amount of data received from varieties of sources, namely mobile app and website to track the behaviour of users, classify users, calculate royalties associated with the song and make appropriate business strategies.

The file server receives data files periodically after every 3 hours.

Dataset Description: Data coming from the two sources, through web and mobile. Data coming from mobile is in **CSV** format and from web is in **XML** format.

Fields Present in the data files are as follows:

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

There are four existing lookup tables present in NoSQL databases, which will be used for data enrichment and analysis.

Lookup Tables:

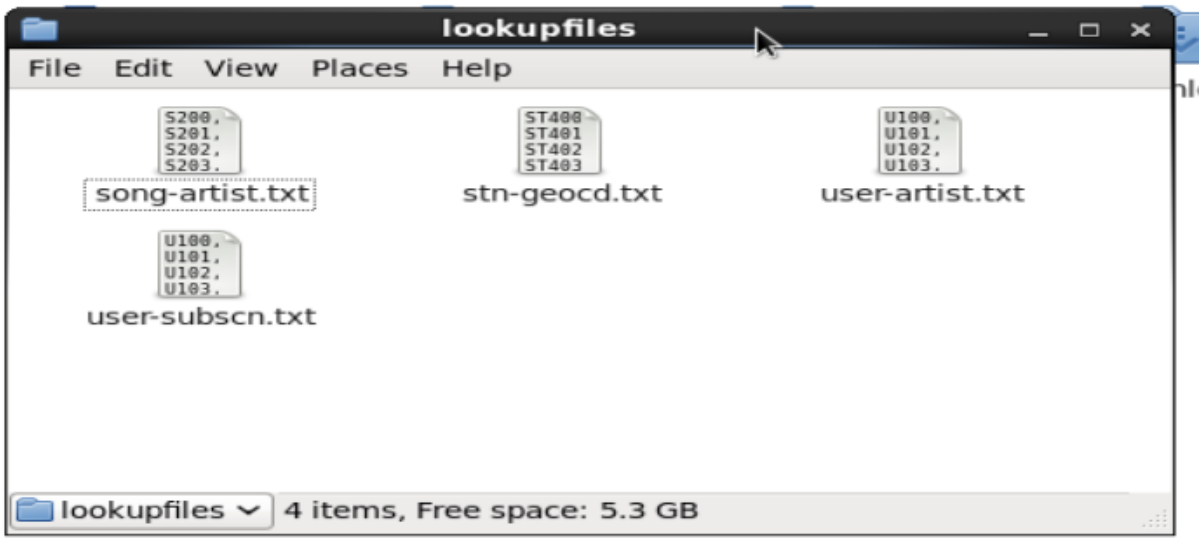
a) Station_Geo_Map: Contains mapping of a **geo_cd** with **station_id**

b)Subscribed_Users: Contains **user_id, subscription_start_date** and **subscription_end_date**. Contains details only for subscribed users

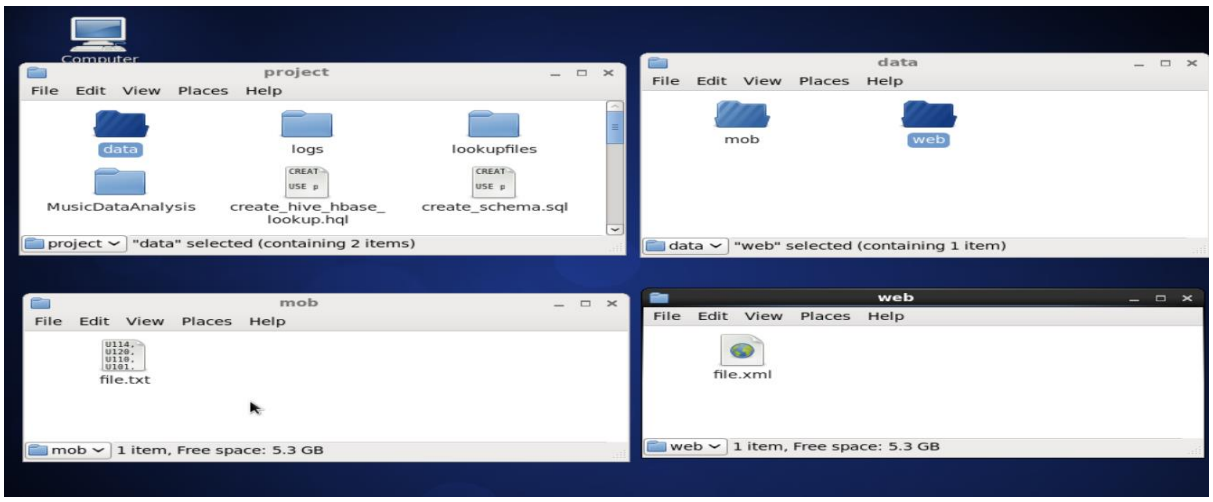
c) Song_Artist_Map: Contains mapping of **song_id** with **artist_id** along with royalty, associated with each play of the song

d) User_Artist_Map: Contains an array of **artist_id(s)** followed by a **user_id**

Below screen, shot shows the lookup dataset present in **lookupfiles** directory.



In our case, data coming from web server is present in the /bhaskar/project/data/web and from mobile app is present in /bhaskar/project/data/mob as shown below in the screenshots.



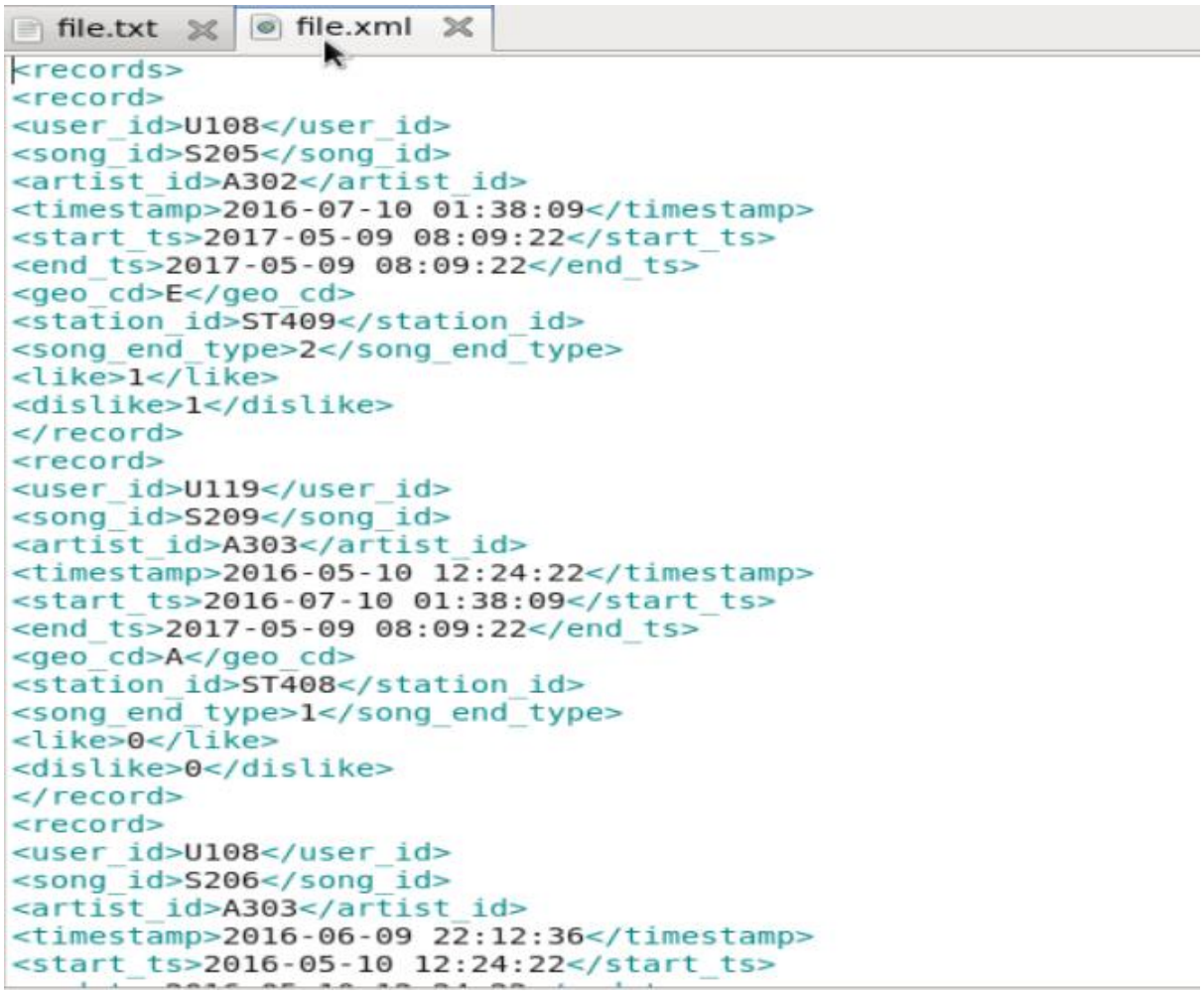
Dataset file present in Mob and Web directory are **file.txt** and **file.xml, which** consists of datasets of 500 records in each of them.

Below screenshot shows the sample data present in file.txt dataset file

file.txt | file.xml

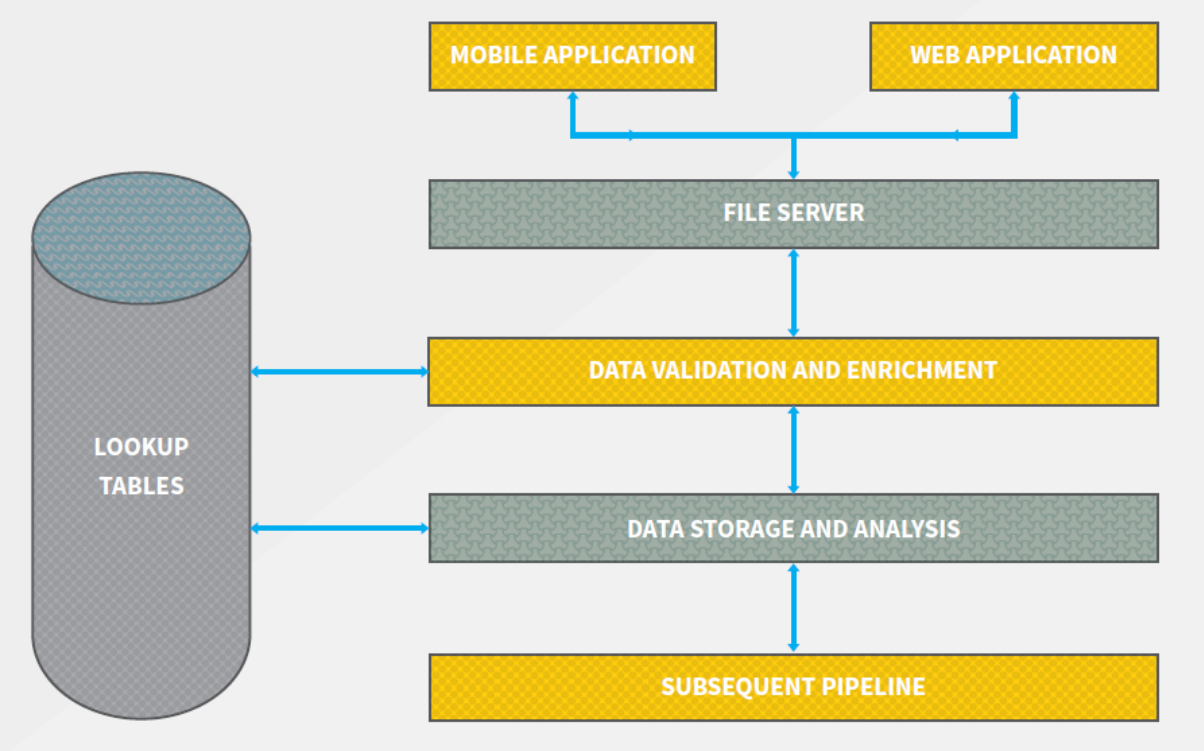
```
U111,S201,A304,1495130523,1465230523,1465130523,AP,ST405,1,1,0
U107,S205,A303,1465230523,1485130523,1465130523,A,ST410,3,0,0
U105,S201,A300,1475130523,1465130523,1485130523,A,ST405,0,0,1
U106,S209,A303,1465230523,1465230523,1465230523,E,ST401,1,1,1
U111,S206,A304,1475130523,1485130523,1485130523,A,ST400,1,0,0
,S201,,1465130523,1475130523,1475130523,AP,ST411,1,1,1
U114,S209,A301,1475130523,1465230523,1475130523,A,ST414,2,0,1
U106,S203,A303,1475130523,1475130523,1485130523,U,ST413,3,0,1
U118,S210,A300,1465230523,1485130523,1465130523,,ST408,1,1,0
U105,S201,A301,1475130523,1485130523,1465230523,A,ST403,1,0,1
U112,S201,A302,1475130523,1465230523,1475130523,AP,ST406,2,0,0
U118,S204,A305,1475130523,1465230523,1485130523,E,ST400,0,0,0
U120,S202,A305,1465130523,1485130523,1465130523,E,ST410,1,0,1
U113,S204,A305,1495130523,1465130523,1465130523,AU,ST411,2,1,0
U102,S206,A300,1495130523,1475130523,1475130523,AP,ST404,1,0,1
U114,S205,A305,1465230523,1465130523,1465230523,E,ST413,1,1,0
U117,S205,,1465230523,1475130523,1465230523,U,ST401,3,1,1
U115,S207,A300,1495130523,1465130523,1465130523,E,ST407,3,0,0
U115,S205,A303,1495130523,1485130523,1465130523,U,ST414,0,0,1
U117,S202,A300,1475130523,1485130523,1465130523,AU,ST400,1,0,0
,S206,A304,1465230523,1465130523,1465230523,,ST400,1,1,1
U117,S201,A302,1495130523,1485130523,1465130523,E,ST414,0,0,0
U117,S202,A301,1465130523,1465130523,1465130523,U,ST411,0,0,0
U117,S202,A303,1465230523,1465130523,1485130523,AP,ST404,0,0,1
U111,S202,A303,1465130523,1465230523,1465230523,E,ST401,1,1,1
U117,S210,A303,1495130523,1475130523,1475130523,AP,ST413,0,1,1
U109,S203,A303,1495130523,1475130523,1465230523,E,ST413,0,0,1
U116,S206,,1465230523,1475130523,1465130523,AP,ST412,0,1,1
U114,S207,A303,1465130523,1475130523,1485130523,AP,ST411,0,1,0
U110,S200,A301,1465230523,1475130523,1475130523,E,ST406,0,1,0
U111,S203,A300,1495130523,1475130523,1465130523,U,ST412,2,1,0
U116,S207,A301,1465130523,1465230523,1465230523,U,ST414,3,1,0
U114,S200,A302,1465230523,1485130523,1465130523,,ST406,1,0,1
```

Below screenshot shows the sample data present in file.xml dataset file



Flow of Operations:

Below screenshot shows the implementation diagram for the project



Our project is divided into five stages, which is described as follow:

Data Ingestion: Storage of raw data to HDFS.

Data Formatting: In this stage, Hive Table will collect web and mob data. Tools used for this process is HIVE and SPARK.

Data Enrichment: With the help of the lookup tables present in the Hbase Table will enrich the raw data present in the Hive Tables. Segregation of the valid and invalid data.

Data Analysis: Analysis of valid data in Spark and create the hive tables to store analysed data.

Data Storage: Export the analysed data from hive to MySQL database.Tool Used for this process is Sqoop.

IMPLEMENTATION:

1. Starting all daemons in Hadoop

We have a script **start-daemon.sh**, we will using this script to start all the daemons that are required to start the services for hive, hbase and MySQL as shown below in screenshot.

start-daemons.sh

```
#!/bin/bash

rm -r /home/acadgild/bhaskar/project/logs
mkdir -p /home/acadgild/bhaskar/project/logs

if [ -f "/home/acadgild/bhaskar/project/logs/current-batch.txt" ]
then
 echo "Batch File Found!"
else
 echo -n "1" > "/home/acadgild/bhaskar/project/logs/current-batch.txt"
fi

chmod 775 /home/acadgild/bhaskar/project/logs/current-batch.txt
echo "After chmod"
batchid=`cat /home/acadgild/bhaskar/project/logs/current-batch.txt`
echo "After batchid-->> "$batchid
LOGFILE=/home/acadgild/bhaskar/project/logs/log_batch_$batchid

echo "Starting daemons" >> $LOGFILE

start-all.sh
start-hbase.sh
mr-jobhistory-daemon.sh start historyserver
service mysqld start

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

So following commands in the script are used:

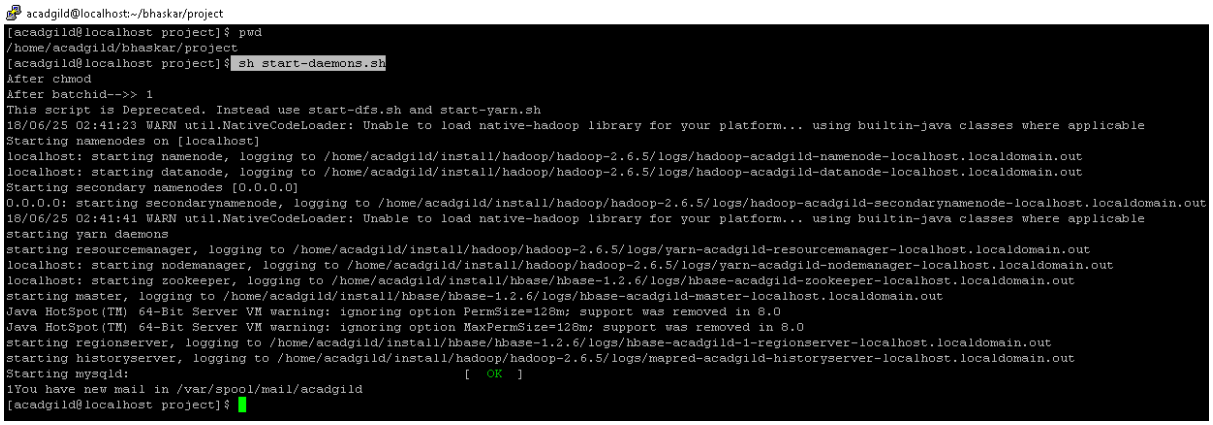
start-all.sh to start all the Hadoop daemons

start-hbase.sh to start all the Hbase daemons

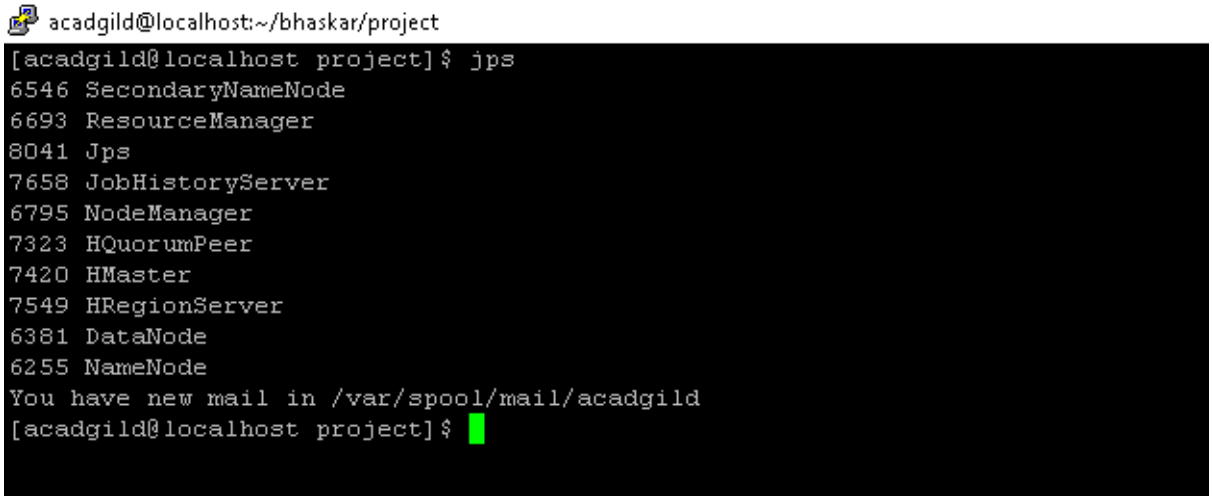
mr-jobhistory-daemon.sh start historyserver to start historyserver

service mysqld start to start the MySQL service.

Below screenshot shows the start process of **start-daemons.sh**



Below screenshot shows all the daemon services by **jps** command



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**.

Now we will ingest the mobile and web datasets from local file system to the HDFS location bhaskar/project/data.

We use the following commands to ingest the data in hdfs location.

hdfs dfs -put /home/acadgild/bhaskar/project/data/* /bhaskar/project/data

Below screenshot shows, the above command used to transfer data from local file system to hdfs location

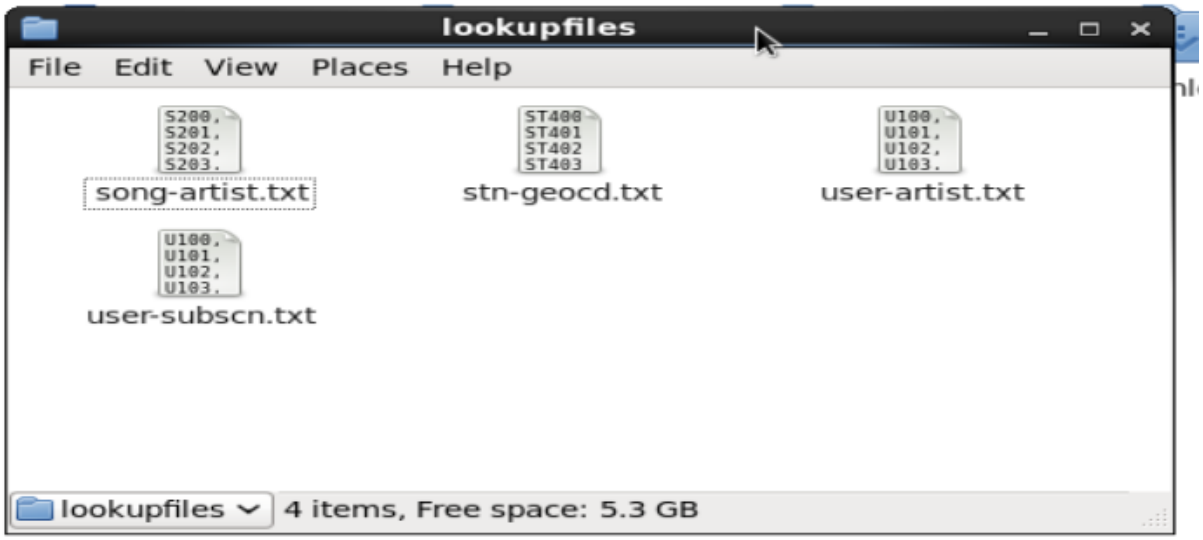
```
[acadgild@localhost ~]$
[acadgild@localhost ~]$ hdfs dfs -put /bhaskar/project/data/* /bhaskar/project/data
18/06/26 12:36:09 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
put: `/bhaskar/project/data/*': No such file or directory
[acadgild@localhost ~]$
[acadgild@localhost ~]$ hdfs dfs -put /home/acadgild/bhaskar/project/data/* /bhaskar/project/data
18/06/26 12:36:35 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
[acadgild@localhost ~]$ hdfs dfs -ls /bhaskar/project/data
18/06/26 12:36:51 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Found 2 items
drwxr-xr-x   - acadgild supergroup          0 2018-06-26 12:36 /bhaskar/project/data/mob
drwxr-xr-x   - acadgild supergroup          0 2018-06-26 12:36 /bhaskar/project/data/web
[acadgild@localhost ~]$
```

acadgild@localhost:~

```
[acadgild@localhost ~]$ hdfs dfs -ls /bhaskar/project/data/mob
18/06/26 12:41:12 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platfor
Found 1 items
-rw-r--r--   1 acadgild supergroup      30831 2018-06-26 12:36 /bhaskar/project/data/mob/file.txt
[acadgild@localhost ~]$ hdfs dfs -ls /bhaskar/project/data/web
18/06/26 12:41:17 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platfor
Found 1 items
-rw-r--r--   1 acadgild supergroup     167356 2018-06-26 12:36 /bhaskar/project/data/web/file.xml
[acadgild@localhost ~]$
```

2. Populating Lookup Tables in Hbase

As described above in data description we have four-lookup dataset present in /bhaskar/project/lookupfiles. Below screenshot shows the all four-lookup files in **lookupfiles** directory.



Now we use these four lookup files to create lookup tables in Hbase using **populate-lookup.sh** script.

Below screenshot shows the **populate-lookup.sh** script

populate-lookup.sh

```
#!/bin/bash

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

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

echo "Creating LookUp Tables" >> $LOGFILE

echo "disable 'station-geo-map'" | hbase shell
echo "drop 'station-geo-map'" | hbase shell
echo "disable 'subscribed-users'" | hbase shell
echo "drop 'subscribed-users'" | hbase shell
echo "disable 'song-artist-map'" | hbase shell
echo "drop 'song-artist-map'" | hbase shell
echo "disable 'user-artist-map'" | hbase shell
echo "drop 'user-artist-map'" | hbase shell


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


echo "Populating LookUp Tables" >> $LOGFILE
```

populate-lookup.sh

```
file="/home/acadgild/bhaskar/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/bhaskar/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/bhaskar/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"
```

```
done <"$file"


cp /home/acadgild/bhaskar/project/lookupfiles/user-artist.txt /home/acadgild/bhaskar/project/lookupfiles/user-artist1.txt

awk '$1=$1' FS="&" OFS=" " /home/acadgild/bhaskar/project/lookupfiles/user-artist1.txt > /home/acadgild/bhaskar/project/
lookupfiles/user-artist.txt

file="/home/acadgild/bhaskar/project/lookupfiles/user-artist.txt"

num=1
while IFS= read -r line
do
 userid=`echo $line | cut -d',' -f1`
 artists=`echo $line | cut -d',' -f2`
for row in $artists
 do
echo "put 'user-artist-map', '$userid', 'artists:artist$num', '$row'" | hbase shell
let "num=num+1"
done
num=1
done <"$file"
```

We will run the above **populate-lookup.sh** shell script and it will create the 4-lookup tables in hbase.

```
[acadgild@localhost project]$ sh populate-lookup.sh
2018-06-25 02:53:46,507 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Stati
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

disable 'station-geo-map'

ERROR: Table station-geo-map does not exist.

Here is some help for this command:
Start disable of named table:
  hbase> disable 't1'
  hbase> disable 'ns1:t1'


2018-06-25 02:53:54,843 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Stati
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017
```

acadgild@localhost:~/bhaskar/project

```
create 'station-geo-map', 'geo'
0 row(s) in 2.8450 seconds

Hbase::Table - station-geo-map
2018-06-25 02:55:01,898 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

create 'subscribed-users', 'subscn'
0 row(s) in 1.6300 seconds

Hbase::Table - subscribed-users
2018-06-25 02:55:11,010 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017
```

```
create 'song-artist-map', 'artist'
0 row(s) in 1.7140 seconds

Hbase::Table - song-artist-map
2018-06-25 02:55:20,492 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... usin
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Sta
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.j
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

create 'user-artist-map', 'artists'
0 row(s) in 1.6020 seconds

Hbase::Table - user-artist-map
2018-06-25 02:55:29,574 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... usin
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Sta
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.j
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017
```

To very that all the lookup tables has been created in hbase correctly we have to open hbase in other terminal by using command **hbase shell** and **list** the tables as show below in the screenshot.

acadgild@localhost:~

```
[acadgild@localhost ~]$ hbase shell
2018-06-25 03:15:02,158 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

hbase(main):001:0> list
TABLE
song-artist-map
station-geo-map
subscribed-users
user-artist-map
4 row(s) in 0.3350 seconds

=> ["song-artist-map", "station-geo-map", "subscribed-users", "user-artist-map"]
hbase(main):002:0>
```

Below screen shows the **scan** result of **song-artist-map**.

acadgild@localhost:~

```
hbase(main):003:0> scan "song-artist-map"
ROW                                   COLUMN+CELL
 S200                                 column=artist:artistid, timestamp=1529875660954, value=A300
 S201                                 column=artist:artistid, timestamp=1529875670019, value=A301
 S202                                 column=artist:artistid, timestamp=1529875678980, value=A302
 S203                                 column=artist:artistid, timestamp=1529875687661, value=A303
 S204                                 column=artist:artistid, timestamp=1529875697096, value=A304
 S205                                 column=artist:artistid, timestamp=1529875705787, value=A301
 S206                                 column=artist:artistid, timestamp=1529875714583, value=A302
 S207                                 column=artist:artistid, timestamp=1529875723457, value=A303
 S208                                 column=artist:artistid, timestamp=1529875732058, value=A304
 S209                                 column=artist:artistid, timestamp=1529875740769, value=A305
10 row(s) in 0.2150 seconds
```

Below screen shows the **scan** result of **station-geo-map**

```
hbase(main):004:0> scan "station-geo-map"
ROW                                    COLUMN+CELL
 ST400                                 column=geo:geo_cd, timestamp=1529875531717, value=A
 ST401                                 column=geo:geo_cd, timestamp=1529875539670, value=AU
 ST402                                 column=geo:geo_cd, timestamp=1529875547937, value=AP
 ST403                                 column=geo:geo_cd, timestamp=1529875555935, value=J
 ST404                                 column=geo:geo_cd, timestamp=1529875563920, value=E
 ST405                                 column=geo:geo_cd, timestamp=1529875571519, value=A
 ST406                                 column=geo:geo_cd, timestamp=1529875579802, value=AU
 ST407                                 column=geo:geo_cd, timestamp=1529875588752, value=AP
 ST408                                 column=geo:geo_cd, timestamp=1529875597689, value=E
 ST409                                 column=geo:geo_cd, timestamp=1529875606592, value=E
 ST410                                 column=geo:geo_cd, timestamp=1529875614890, value=A
 ST411                                 column=geo:geo_cd, timestamp=1529875623862, value=A
 ST412                                 column=geo:geo_cd, timestamp=1529875632380, value=AP
 ST413                                 column=geo:geo_cd, timestamp=1529875641305, value=J
 ST414                                 column=geo:geo_cd, timestamp=1529875651412, value=E
15 row(s) in 0.0980 seconds
```

Below screen shows the **scan** result of **subscribed-users**

acadgild@localhost:~

```
hbase(main):005:0> scan "subscribed-users"
ROW                                    COLUMN+CELL
 U100                                  column=subscn:enddt, timestamp=1529875758685, value=1465130523
 U100                                  column=subscn:startdt, timestamp=1529875749870, value=1465230523
 U101                                  column=subscn:enddt, timestamp=1529875775901, value=1475130523
 U101                                  column=subscn:startdt, timestamp=1529875767788, value=1465230523
 U102                                  column=subscn:enddt, timestamp=1529875793077, value=1475130523
 U102                                  column=subscn:startdt, timestamp=1529875784118, value=1465230523
 U103                                  column=subscn:enddt, timestamp=1529875811608, value=1475130523
 U103                                  column=subscn:startdt, timestamp=1529875802208, value=1465230523
 U104                                  column=subscn:enddt, timestamp=1529875830292, value=1475130523
 U104                                  column=subscn:startdt, timestamp=1529875821688, value=1465230523
 U105                                  column=subscn:enddt, timestamp=1529875849025, value=1475130523
 U105                                  column=subscn:startdt, timestamp=1529875839155, value=1465230523
 U106                                  column=subscn:enddt, timestamp=1529875867351, value=1485130523
 U106                                  column=subscn:startdt, timestamp=1529875858435, value=1465230523
 U107                                  column=subscn:enddt, timestamp=1529875885687, value=1455130523
 U107                                  column=subscn:startdt, timestamp=1529875876089, value=1465230523
 U108                                  column=subscn:enddt, timestamp=1529875904331, value=1465230623
 U108                                  column=subscn:startdt, timestamp=1529875895123, value=1465230523
 U109                                  column=subscn:enddt, timestamp=1529875923226, value=1475130523
 U109                                  column=subscn:startdt, timestamp=1529875913982, value=1465230523
 U110                                  column=subscn:enddt, timestamp=1529875942218, value=1475130523
 U110                                  column=subscn:startdt, timestamp=1529875932622, value=1465230523
 U111                                  column=subscn:enddt, timestamp=1529875960477, value=1475130523
 U111                                  column=subscn:startdt, timestamp=1529875951597, value=1465230523
 U112                                  column=subscn:enddt, timestamp=1529875979218, value=1475130523
 U112                                  column=subscn:startdt, timestamp=1529875969416, value=1465230523
 U113                                  column=subscn:enddt, timestamp=1529875999419, value=1485130523
 U113                                  column=subscn:startdt, timestamp=1529875988467, value=1465230523
 U114                                  column=subscn:enddt, timestamp=1529876017359, value=1468130523
 U114                                  column=subscn:startdt, timestamp=1529876008637, value=1465230523
15 row(s) in 0.1760 seconds
```

Below screen shows the **scan** result of **user-artist-map**

acadgild@localhost:~

```
hbase(main):018:0> scan "user-artist-map"
ROW                                   COLUMN+CELL
 U100                                 column=artists:artist1, timestamp=1529876026453, value=A300
 U100                                 column=artists:artist2, timestamp=1529876036040, value=A301
 U100                                 column=artists:artist3, timestamp=1529876044612, value=A302
 U101                                 column=artists:artist1, timestamp=1529876054135, value=A301
 U101                                 column=artists:artist2, timestamp=1529876063194, value=A302
 U102                                 column=artists:artist1, timestamp=1529876071784, value=A302
 U103                                 column=artists:artist1, timestamp=1529876081799, value=A303
 U103                                 column=artists:artist2, timestamp=1529876090741, value=A301
 U103                                 column=artists:artist3, timestamp=1529876099925, value=A302
 U104                                 column=artists:artist1, timestamp=1529876109103, value=A304
 U104                                 column=artists:artist2, timestamp=1529876118842, value=A301
 U105                                 column=artists:artist1, timestamp=1529876128073, value=A305
 U105                                 column=artists:artist2, timestamp=1529876137019, value=A301
 U105                                 column=artists:artist3, timestamp=1529876145760, value=A302
 U106                                 column=artists:artist1, timestamp=1529876155248, value=A301
 U106                                 column=artists:artist2, timestamp=1529876164602, value=A302
 U107                                 column=artists:artist1, timestamp=1529876174447, value=A302
 U108                                 column=artists:artist1, timestamp=1529876183905, value=A300
 U108                                 column=artists:artist2, timestamp=1529876193483, value=A303
 U108                                 column=artists:artist3, timestamp=1529876204028, value=A304
 U109                                 column=artists:artist1, timestamp=1529876213248, value=A301
 U109                                 column=artists:artist2, timestamp=1529876223055, value=A303
 U110                                 column=artists:artist1, timestamp=1529876232825, value=A302
 U110                                 column=artists:artist2, timestamp=1529876242359, value=A301
 U111                                 column=artists:artist1, timestamp=1529876251384, value=A303
 U111                                 column=artists:artist2, timestamp=1529876260046, value=A301
 U112                                 column=artists:artist1, timestamp=1529876269111, value=A304
 U112                                 column=artists:artist2, timestamp=1529876279082, value=A301
 U113                                 column=artists:artist1, timestamp=1529876288569, value=A305
 U113                                 column=artists:artist2, timestamp=1529876298262, value=A302
 U114                                 column=artists:artist1, timestamp=1529876307921, value=A300
 U114                                 column=artists:artist2, timestamp=1529876318041, value=A301
 U114                                 column=artists:artist3, timestamp=1529876327748, value=A302
15 row(s) in 0.0960 seconds
```

3. Creating Hive Tables on the top of Hbase:

After creation of lookup tables in hbase, we need to link these tables in Hive using **hbase storage handler.**

Now with the help of hbase storage handler and SerDe properties we are creating the hive external tables by matching the columns of hbase tables to hive tables.

For this, we have hive script **create_hive_hbase_lookup.hql** that consist of hive query to create external tables as shown below in screenshots.

This .hql file consist of four hive queries which creates the tables **station_geo_map,subscribed_users,song_atrtist-map** and **user_artists.**

create_hive_hbase_lookup.hql | data_enrichment_filtering_schema

```
CREATE DATABASE IF NOT EXISTS project;

USE project;

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

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

create external table if not exists song_artist_map
(
song_id STRING,
artist_id STRING
)
```

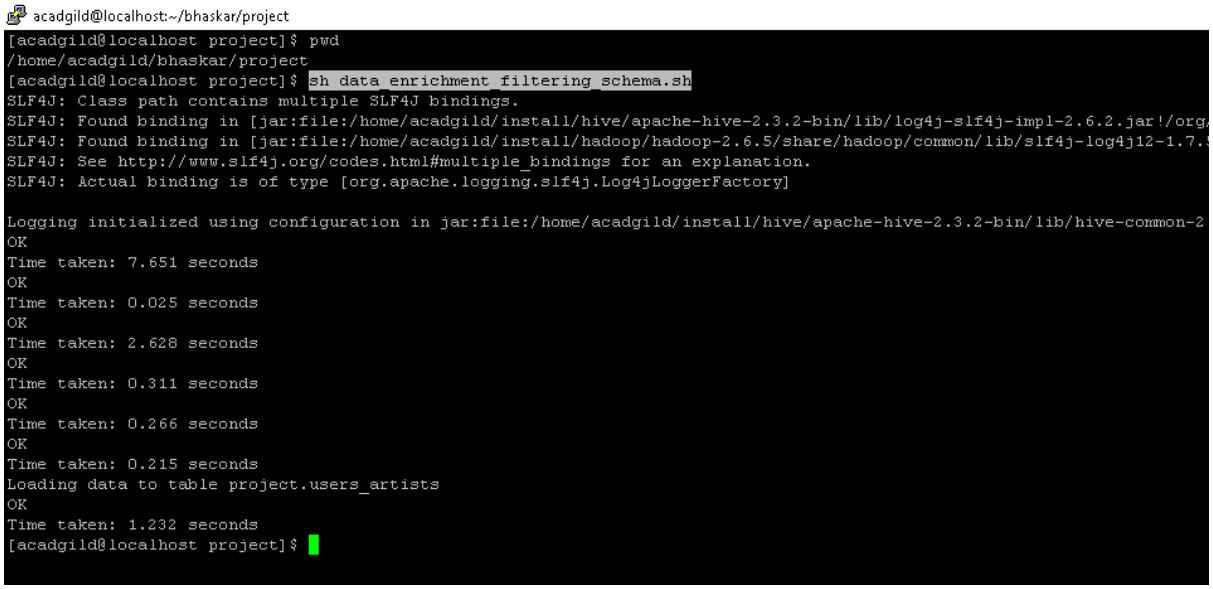
```
create external table if not exists song_artist_map
(
song_id STRING,
artist_id STRING
)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
with serdeproperties
("hbase.columns.mapping"=":key,artist:artistid")
tblproperties("hbase.table.name"="song-artist-map");


CREATE TABLE users_artists
(
user_id STRING,
artists_array ARRAY<STRING>
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
COLLECTION ITEMS TERMINATED BY '&';

LOAD DATA LOCAL INPATH '/home/acadgild/bhaskar/project/lookupfiles/user-artist.txt'
OVERWRITE INTO TABLE users_artists;
```

Now to run the hive .hql file we have **data_enrichment_filtering_schema.sh** script, which run the **create_hive_hbase_lookup.hql** by using "**hive –f** "command in the script as shown below in the screenshot.

data_enrichment_filtering_schema.sh

```
#!/bin/bash

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

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

hive -f /home/acadgild/bhaskar/project/create_hive_hbase_lookup.hql
```

Below screenshot shows that tables getting created in hive by running the **data_enrichement_filtering_schema.sh** file



To verify the creation of all tables in hive we open the hive in another terminal and check the tables as shown below in **screenshot**.

4. Data Formatting

Now we are merging the data coming from both web applications and mobile applications by inserting them in a common table for analysing purpose and create partitioned data based on batchid, as we have to run this script for every 3 hours.

For this, we create a spark application **DataFormatting.scala** that will create spark **sqlcontext.**

Spark SQL allows relational queries expressed in SQL, HiveQL, or Scala to be executed using Spark.

When working with Hive we must construct a HiveContext, which inherits from SQLContext, and adds support for finding tables in in the Metastore and writing queries using HiveQL.

We are going to use spark HiveContext to create a hive table **formatted_input** which portioned by batchid.

In this spark application, we inserted the mobile data from local file system directory to the table and partitioned it by batchid.

For xml data coming from web data directory, we used **com.databricks.spark.xml** format to read and format the xml data.

This data is first loaded into temporary table **web_data** and then inserted into formatted_input table and partitioned it by batchid.

DataFormatting.scala | build.sbt

```scala
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.sql

object DataFormatting {
  def main(args: Array[String]): Unit = {
    val conf = new SparkConf().setAppName("Data Formatting")
    val sc = new SparkContext(conf)
    val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)
    val batchId = args(0)
    val create_hive_table = """CREATE TABLE IF NOT EXISTS project.formatted_input
                              (
                              User_id STRING,
                              Song_id STRING,
                              Artist_id STRING,
                              Timestamp STRING,
                              Start_ts STRING,
                              End_ts STRING,
                              Geo_cd STRING,
                              Station_id STRING,
                              Song_end_type INT,
                              Like INT,
                              Dislike INT
                              )
                              PARTITIONED BY
                              (batchid INT)
                              ROW FORMAT DELIMITED
                              FIELDS TERMINATED BY ','
                              """
                              ROW FORMAT DELIMITED
                              FIELDS TERMINATED BY ','
                              """

    val load_mob_data = s"""LOAD DATA LOCAL INPATH '/home/acadgild/bhaskar/project/data/mob/file.txt'
                              INTO TABLE project.formatted_input PARTITION (batchid='$batchId')"""

    val load_web_data = s"""INSERT INTO project.formatted_input
                              PARTITION(batchid='$batchId')
                              SELECT user_id,
                              song_id,
                              artist_id,
                              unix_timestamp(timestamp,'yyyy-MM-dd HH:mm:ss') AS timestamp,
                              unix_timestamp(start_ts,'yyyy-MM-dd HH:mm:ss') AS start_ts,
                              unix_timestamp(end_ts,'yyyy-MM-dd HH:mm:ss') AS end_ts,
                              geo_cd,
                              station_id,
                              song_end_type,
                              like,
                              dislike
                              FROM web_data
                              """
```

```
49
50      try {
51          val xmlData = sqlContext.read.format( source = "com.databricks.spark.xml")
52                                       .option("rowTag", "record")
53                                       .load( path = "file:///home/acadgild/bhaskar/project/data/web/file.xml")
54
55          xmlData.createOrReplaceTempView( viewName = "web_data")
56
57          sqlContext.sql(create_hive_table)
58          sqlContext.sql(load_mob_data)
59          sqlContext.sql(load_web_data)
60        }
61       catch{
62        case e: Exception=>e.printStackTrace()
63       }
64  }
65  }
```

Code for above spark application:

```
import org.apache.spark.{SparkConf, SparkContext}

import org.apache.spark.sql


object DataFormatting {
  def main(args: Array[String]): Unit = {
    val conf = new SparkConf().setAppName("Data Formatting")
    val sc = new SparkContext(conf)
    val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)
    val batchId = args(0)
    val create_hive_table = """CREATE TABLE IF NOT EXISTS project.formatted_input
                            (
                            User_id STRING,
                            Song_id STRING,
                            Artist_id STRING,
                            Timestamp STRING,
                            Start_ts STRING,
                            End_ts STRING,
                            Geo_cd STRING,
                            Station_id STRING,
                            Song_end_type INT,
                            Like INT,
```

```
                Dislike INT
                )
                PARTITIONED BY
                (batchid INT)
                ROW FORMAT DELIMITED
                FIELDS TERMINATED BY ','
                """

  val load_mob_data = s"""LOAD DATA LOCAL INPATH '/home/acadgild/bhaskar/project/data/mob/file.txt'
                INTO TABLE project.formatted_input PARTITION (batchid='$batchId')"""

  val load_web_data = s"""INSERT INTO project.formatted_input
                PARTITION(batchid='$batchId')
                SELECT user_id,
                song_id,
                artist_id,
                unix_timestamp(timestamp,'yyyy-MM-dd HH:mm:ss') AS timestamp,
                unix_timestamp(start_ts,'yyyy-MM-dd HH:mm:ss') AS start_ts,
                unix_timestamp(end_ts,'yyyy-MM-dd HH:mm:ss') AS end_ts,
                geo_cd,
                station_id,
                song_end_type,
                like,
                dislike
                FROM web_data
                """

  try {
```

```
        val xmlData = sqlContext.read.format("com.databricks.spark.xml")
                                    .option("rowTag", "record")
.load("file:///home/acadgild/bhaskar/project/data/web/file.xml")
        xmlData.createOrReplaceTempView("web_data")

        sqlContext.sql(create_hive_table)
        sqlContext.sql(load_mob_data)
        sqlContext.sql(load_web_data)
      }
     catch{
      case e: Exception=>e.printStackTrace()
     }
 }
}
```

To run the above spark application we need to package them alongside with our application containing other code (which we will using in further process like data enrichment and data analysis part) and its dependencies in order to distribute the code to a spark cluster.

The above spark application written in **IDEA INTELLIJ** and its required dependencies added in **build.sbt.** I put the complete application in /bhaskar/project/ directory.

The complete application consist of three Scala files **DataFormatting.scala**, **DataEnrichment.scala** and **DataAnalysis.scala** files. DataEnrichment.scala and DataAnalysis.scala will used in next stages.

Now, we will use **sbt –v package** command in root folder of the spark application as shown below in the screenshot to bundle up the assembly jars, list Spark and Hadoop as provided dependencies. Once we have an assembled jar we can call the bin/spark-submit in script

Below screen shot shows that jar file created successfully

acadgild@localhost:~/bhaskar/project/MusicDataAnalysis

```
[acadgild@localhost project]$
[acadgild@localhost project]$ cd MusicDataAnalysis
[acadgild@localhost MusicDataAnalysis]$ sbt -v package
[process_args] java_version = '1.8'
# Executing command line:
java
-Xms1024m
-Xmx1024m
-XX:ReservedCodeCacheSize=128m
-XX:MaxMetaspaceSize=256m
-jar
/usr/share/sbt/bin/sbt-launch.jar
package

Getting org.scala-sbt sbt 1.0.4  (this may take some time)...
downloading https://repo1.maven.org/maven2/org/scala-sbt/sbt/1.0.4/sbt-1.0.4.jar ...
        [SUCCESSFUL ] org.scala-sbt#sbt;1.0.4!sbt.jar (888ms)
downloading https://repo1.maven.org/maven2/org/scala-lang/scala-library/2.12.4/scala-library-2.12.4.jar ...
        [SUCCESSFUL ] org.scala-lang#scala-library;2.12.4!scala-library.jar (10140ms)
downloading https://repo1.maven.org/maven2/org/scala-sbt/main_2.12/1.0.4/main_2.12-1.0.4.jar ...
        [SUCCESSFUL ] org.scala-sbt#main_2.12;1.0.4!main_2.12.jar (1611ms)
downloading https://repo1.maven.org/maven2/org/scala-sbt/logic_2.12/1.0.4/logic_2.12-1.0.4.jar ...
        [SUCCESSFUL ] org.scala-sbt#logic_2.12;1.0.4!logic_2.12.jar (670ms)
downloading https://repo1.maven.org/maven2/org/scala-sbt/actions_2.12/1.0.4/actions_2.12-1.0.4.jar ...
```

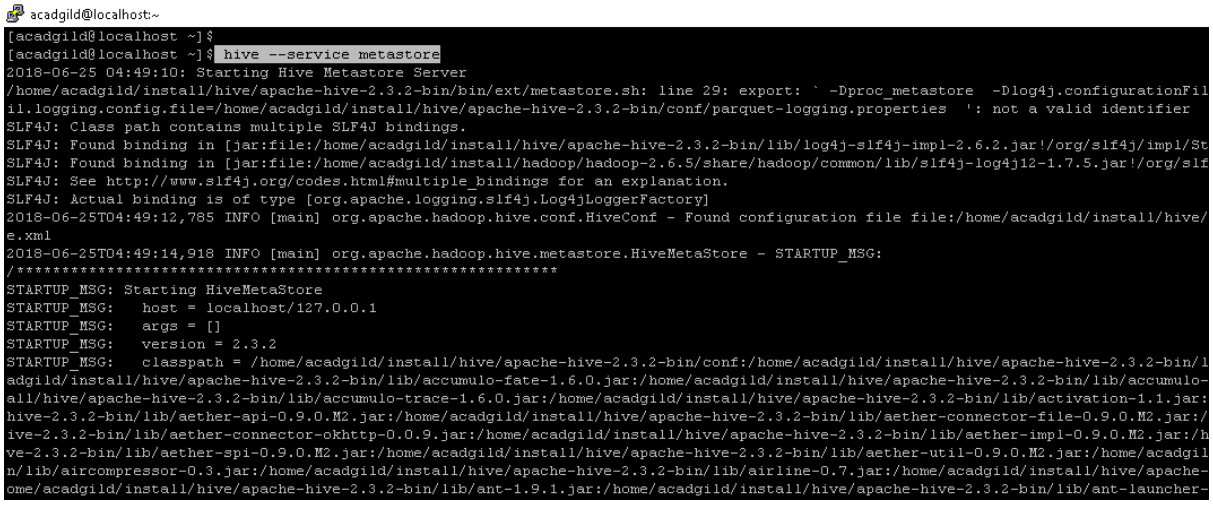
```
[info] Loading project definition from /home/acadgild/bhaskar/project/MusicDataAnalysis/project
[info] Loading settings from build.sbt ...
[info] Set current project to MusicDataAnalysis (in build file:/home/acadgild/bhaskar/project/MusicDataAnalysis/)
[info] Updating {file:/home/acadgild/bhaskar/project/MusicDataAnalysis/}musicdataanalysis...
[info] downloading https://repo1.maven.org/maven2/org/apache/avro/avro/1.7.7/avro-1.7.7.jar ...
[info] downloading https://repo1.maven.org/maven2/org/apache/hadoop/hadoop-mapreduce-client-common/2.6.5/hadoop-mapreduce-client-common-2.6.5.jar
[info]  [SUCCESSFUL ] org.apache.avro#avro;1.7.7!avro.jar (1445ms)
[info]  [SUCCESSFUL ] org.apache.hadoop#hadoop-mapreduce-client-common;2.6.5!hadoop-mapreduce-client-common.jar (2570ms)
[info] Done updating.
[info] Compiling 3 Scala sources to /home/acadgild/bhaskar/project/MusicDataAnalysis/target/scala-2.11/classes ...
[info] Non-compiled module 'compiler-bridge_2.11' for Scala 2.11.8. Compiling...
[info]   Compilation completed in 18.505s.
[warn] there were three deprecation warnings; re-run with -deprecation for details
[warn] one warning found
[info] Done compiling.
[warn] Multiple main classes detected.  Run 'show discoveredMainClasses' to see the list
[info] Packaging /home/acadgild/bhaskar/project/MusicDataAnalysis/target/scala-2.11/musicdataanalysis_2.11-1.0.jar ...
[info] Done packaging.
[success] Total time: 49 s, completed Jun 25, 2018 4:41:16 AM
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost MusicDataAnalysis]$
```

Above spark-application will be run by using **spark-submit** script in **dataformatting.sh** script as shown below in screenshot.

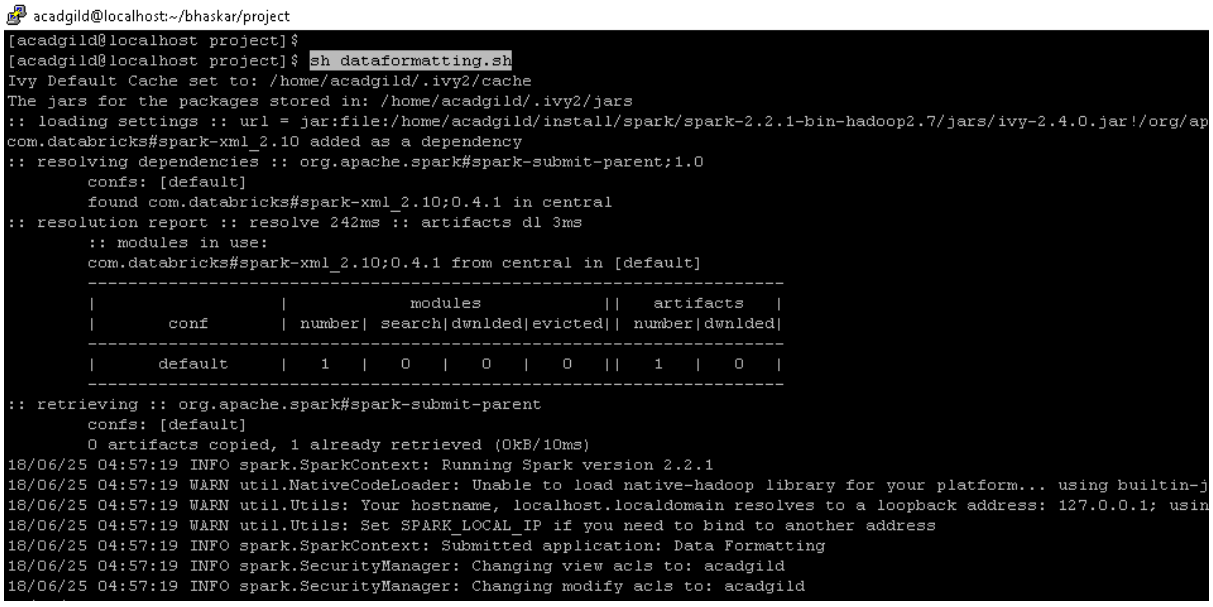
dataformatting.sh

```
#!/bin/bash

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


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

spark-submit --packages com.databricks:spark-xml_2.10:0.4.1 \
--class DataFormatting \
--master local[2] \
/home/acadgild/bhaskar/project/MusicDataAnalysis/target/scala-2.11/musicdataanalysis_2.11-1.0.jar $batchid
```

To run **dataformatting.sh** first we have to start the hive metastore service by using command **hive --service metastore** as shown below in screenshot.



Now we will run the **dataformatting.sh** script



Screenshot shows the creation of the formatted input table

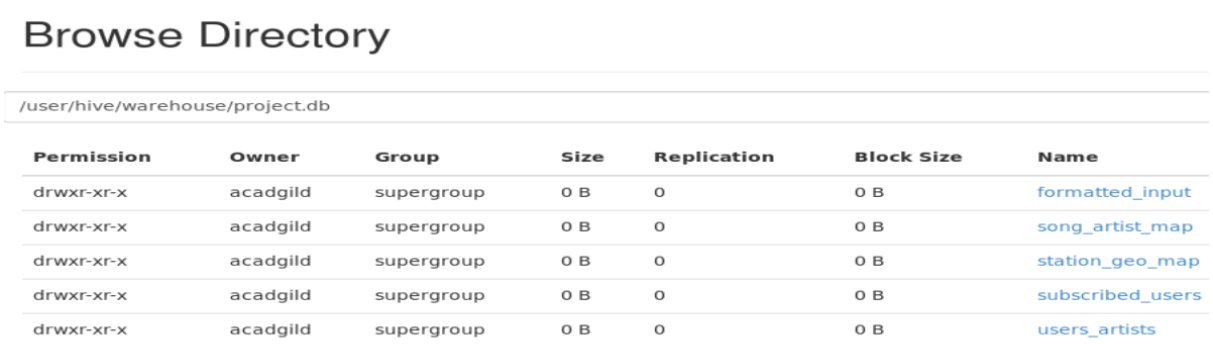
```
18/06/25 04:57:27 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 1197 ms on localhost (exec
18/06/25 04:57:27 INFO scheduler.DAGScheduler: ResultStage 0 (treeAggregate at InferSchema.scala:109) finished in 1.2
18/06/25 04:57:27 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
18/06/25 04:57:27 INFO scheduler.DAGScheduler: Job 0 finished: treeAggregate at InferSchema.scala:109, took 1.573228
18/06/25 04:57:30 INFO execution.SparkSqlParser: Parsing command: web_data
18/06/25 04:57:31 INFO storage.BlockManagerInfo: Removed broadcast_1_piece0 on 192.168.0.26:38946 in memory (size: 2.
18/06/25 04:57:31 INFO execution.SparkSqlParser: Parsing command: CREATE TABLE IF NOT EXISTS project.formatted_input
                          (
                          User_id STRING,
                          Song_id STRING,
                          Artist_id STRING,
                          Timestamp STRING,
                          Start_ts STRING,
                          End_ts STRING,
                          Geo_cd STRING,
                          Station_id STRING,
                          Song_end_type INT,
                          Like INT,
                          Dislike INT
                          )
                          PARTITIONED BY
                          (batchid INT)
                          ROW FORMAT DELIMITED
                          FIELDS TERMINATED BY ','

18/06/25 04:57:32 INFO parser.CatalystSqlParser: Parsing command: array<string>
18/06/25 04:57:33 INFO execution.SparkSqlParser: Parsing command: LOAD DATA LOCAL INPATH '/home/acadgild/bhaskar/proj
                          INTO TABLE project.formatted_input PARTITION (batchid='1')
18/06/25 04:57:34 INFO parser.CatalystSqlParser: Parsing command: int
18/06/25 04:57:34 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 04:57:34 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 04:57:34 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 04:57:34 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 04:57:34 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 04:57:34 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 04:57:34 INFO parser.CatalystSqlParser: Parsing command: string
```

Below screen shot shows the insertion data from temporary table **web_data**

```
18/06/25 04:57:34 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 04:57:34 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 04:57:34 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 04:57:34 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 04:57:34 INFO parser.CatalystSqlParser: Parsing command: int
18/06/25 04:57:34 INFO parser.CatalystSqlParser: Parsing command: int
18/06/25 04:57:34 INFO parser.CatalystSqlParser: Parsing command: int
18/06/25 04:57:35 INFO common.FileUtils: Creating directory if it doesn't exist: hdfs://localhost:8020/user/hive/warehouse/proj
18/06/25 04:57:35 ERROR hdfs.KeyProviderCache: Could not find uri with key [dfs.encryption.key.provider.uri] to create a keyPro
18/06/25 04:57:35 INFO metadata.Hive: Renaming src: file:/home/acadgild/bhaskar/project/data/mob/file.txt, dest: hdfs://localho
rmatted_input/batchid=1/file.txt, Status:true
18/06/25 04:57:35 INFO execution.SparkSqlParser: Parsing command: INSERT INTO project.formatted_input
                          PARTITION(batchid='1')
                          SELECT user_id,
                          song_id,
                          artist_id,
                          unix_timestamp(timestamp,'yyyy-MM-dd HH:mm:ss') AS timestamp,
                          unix_timestamp(start_ts,'yyyy-MM-dd HH:mm:ss') AS start_ts,
                          unix_timestamp(end_ts,'yyyy-MM-dd HH:mm:ss') AS end_ts,
                          geo_cd,
                          station_id,
                          song_end_type,
                          like,
                          dislike
                          FROM web_data

18/06/25 04:57:36 INFO parser.CatalystSqlParser: Parsing command: int
18/06/25 04:57:36 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 04:57:36 INFO parser.CatalystSqlParser: Parsing command: string
```

```
18/06/25 04:57:38 INFO metadata.Hive: Renaming src: hdfs://localhost:8020/user/hive/warehouse/project.db/formatted_input/.hive-staging_hive_2018-06-25_04-57-36_304_5944
4927876530073316-1/-ext-10000/part-00000-ddf2d66e-dfd6-4d5f-b071-ba0678e1da80-c000, dest: hdfs://localhost:8020/user/hive/warehouse/project.db/formatted_input/batchid=1/
part-00000-ddf2d66e-dfd6-4d5f-b071-ba0678e1da80-c000, Status:true
18/06/25 04:57:39 INFO execution.SparkSqlParser: Parsing command: `project`.`formatted_input`
18/06/25 04:57:39 INFO parser.CatalystSqlParser: Parsing command: int
18/06/25 04:57:39 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 04:57:39 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 04:57:39 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 04:57:39 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 04:57:39 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 04:57:39 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 04:57:39 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 04:57:39 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 04:57:39 INFO parser.CatalystSqlParser: Parsing command: int
18/06/25 04:57:39 INFO parser.CatalystSqlParser: Parsing command: int
18/06/25 04:57:39 INFO parser.CatalystSqlParser: Parsing command: int
18/06/25 04:57:39 INFO spark.SparkContext: Invoking stop() from shutdown hook
18/06/25 04:57:39 INFO server.AbstractConnector: Stopped Spark@57ba8097{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}
18/06/25 04:57:39 INFO ui.SparkUI: Stopped Spark web UI at http://192.168.0.26:4040
18/06/25 04:57:39 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/06/25 04:57:39 INFO memory.MemoryStore: MemoryStore cleared
18/06/25 04:57:39 INFO storage.BlockManager: BlockManager stopped
18/06/25 04:57:39 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
18/06/25 04:57:39 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/06/25 04:57:39 INFO spark.SparkContext: Successfully stopped SparkContext
18/06/25 04:57:39 INFO util.ShutdownHookManager: Shutdown hook called
18/06/25 04:57:39 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-8994be7c-f8e3-45bb-a8ff-ae421c1a4286
```

The above script will produce formatted output in hdfs folder as shown below.

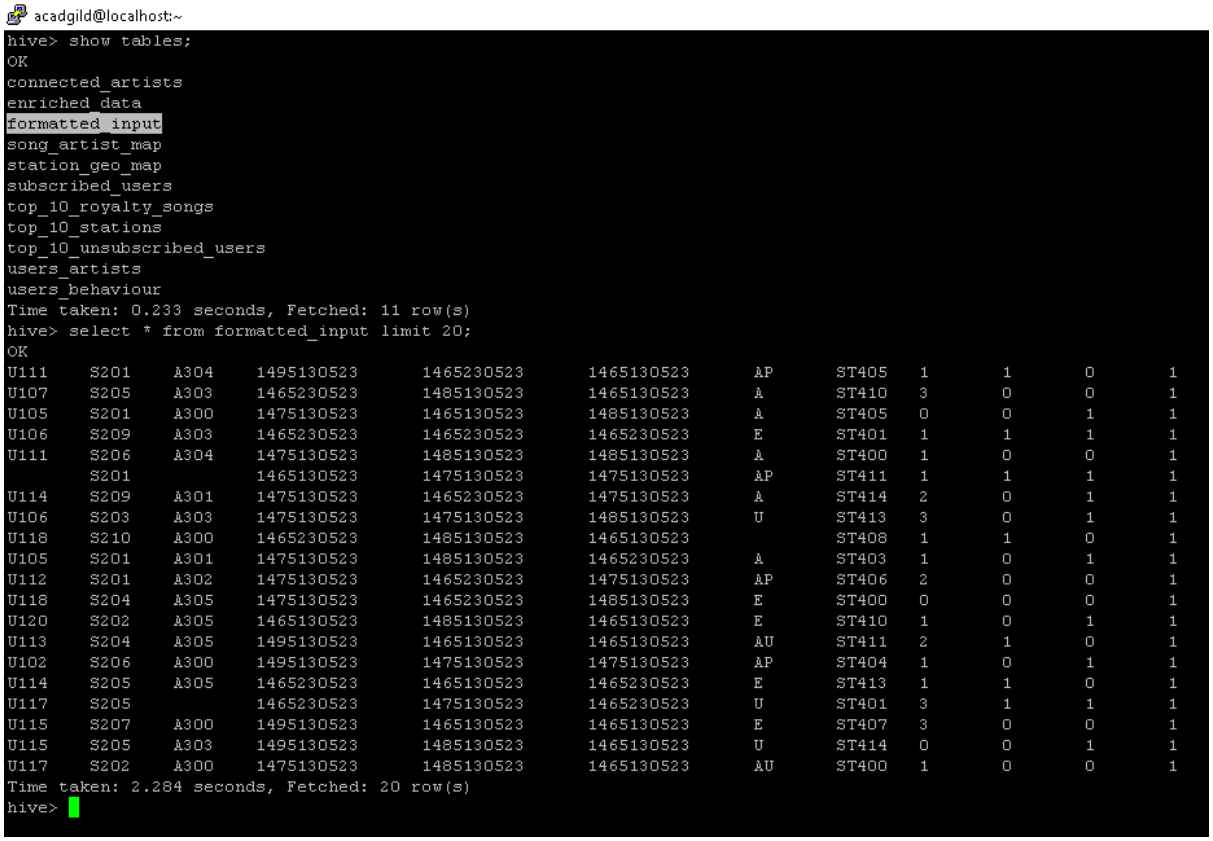


Browse Directory

/user/hive/warehouse/project.db

Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	acadgild	supergroup	0 B	0	0 B	formatted_input
drwxr-xr-x	acadgild	supergroup	0 B	0	0 B	song_artist_map
drwxr-xr-x	acadgild	supergroup	0 B	0	0 B	station_geo_map
drwxr-xr-x	acadgild	supergroup	0 B	0	0 B	subscribed_users
drwxr-xr-x	acadgild	supergroup	0 B	0	0 B	users_artists

To verify the **formatted_input table,** check the tables in hive terminal as show below screenshot.



```
acadgild@localhost:~
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: 0.233 seconds, Fetched: 11 row(s)
hive> select * from formatted_input limit 20;
OK
U111    S201    A304    1495130523      1465230523      1465130523      AP      ST405   1       1       0       1
U107    S205    A303    1465230523      1485130523      1465130523      A       ST410   3       0       0       1
U105    S201    A300    1475130523      1465130523      1485130523      A       ST405   0       0       1       1
U106    S209    A303    1465230523      1465230523      1465230523      E       ST401   1       1       1       1
U111    S206    A304    1475130523      1485130523      1485130523      A       ST400   1       0       0       1
        S201            1465130523      1475130523      1475130523      AP      ST411   1       1       1       1
U114    S209    A301    1475130523      1465230523      1475130523      A       ST414   2       0       1       1
U106    S203    A303    1475130523      1475130523      1485130523      U       ST413   3       0       1       1
U118    S210    A300    1465230523      1485130523      1465130523              ST408   1       1       0       1
U105    S201    A301    1475130523      1485130523      1465230523      A       ST403   1       0       1       1
U112    S201    A302    1475130523      1465230523      1475130523      AP      ST406   2       0       0       1
U118    S204    A305    1475130523      1465230523      1485130523      E       ST400   0       0       0       1
U120    S202    A305    1465130523      1485130523      1465130523      E       ST410   1       0       1       1
U113    S204    A305    1495130523      1465130523      1465130523      AU      ST411   2       1       0       1
U102    S206    A300    1495130523      1475130523      1475130523      AP      ST404   1       0       1       1
U114    S205    A305    1465230523      1465130523      1465230523      E       ST413   1       1       0       1
U117    S205            1465230523      1475130523      1465230523      U       ST401   3       1       1       1
U115    S207    A300    1495130523      1465130523      1465130523      E       ST407   3       0       0       1
U115    S205    A303    1495130523      1485130523      1465130523      U       ST414   0       0       1       1
U117    S202    A300    1475130523      1485130523      1465130523      AU      ST400   1       0       0       1
Time taken: 2.284 seconds, Fetched: 20 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 did not mentioned anything about **user_id** for enrichment purpose.

Data formatting phase executed successfully by loading both mobile, web data, and partitioned based on batchid.

5. Data Enrichment:

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

As per the requirement of music-catering company 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 not found, consider that record 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.

For all the above requirements of data enrichment rules, we created spark application **DataEnrichment.scala.**

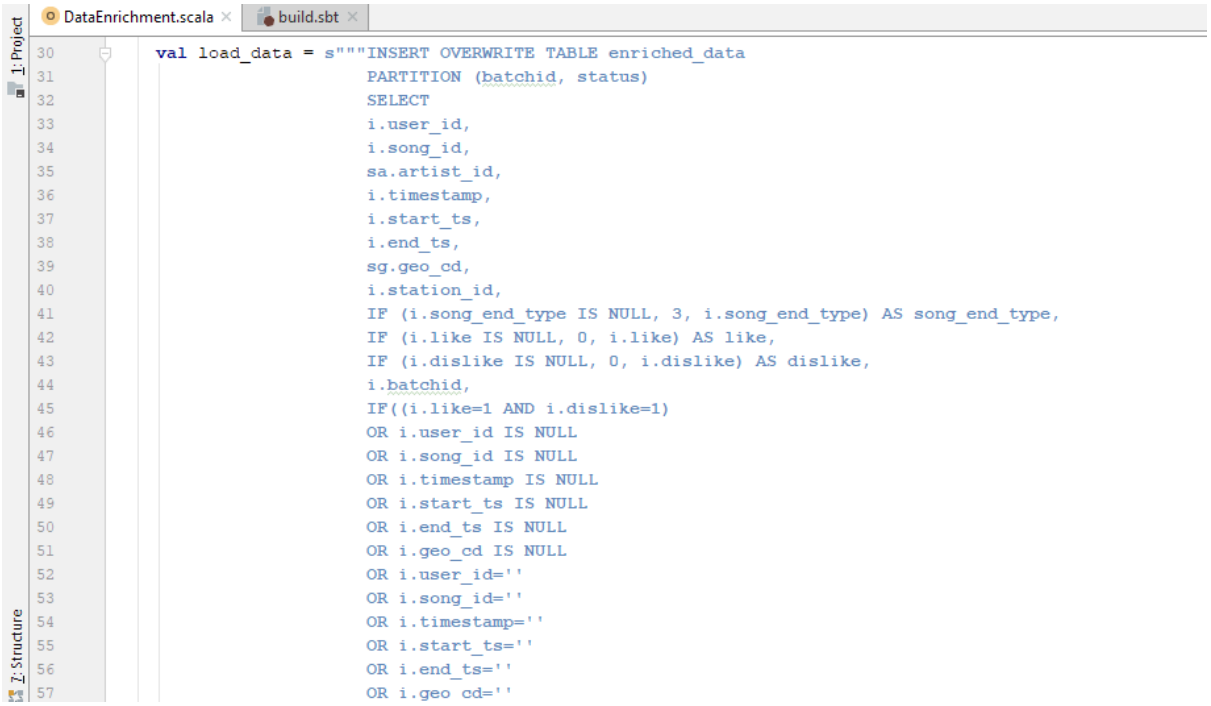
This spark application will first create the Hive table **enrichment_data** will all the required columns.

Table is partitioned by batchid and status. Stored the tables as **ORC** format.

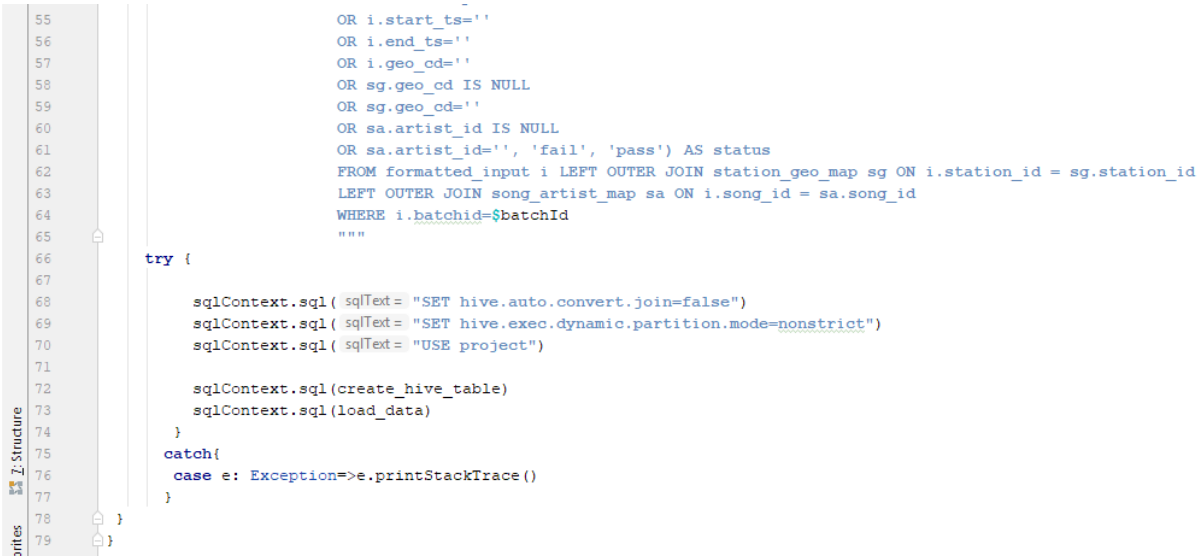
Inserted data from **formatted_input** table by applying all the above data enrichment rules.

Below screenshot shows the spark-application DataEnrichment.scala

```scala
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.sql

object DataEnrichment {
  def main(args: Array[String]): Unit = {
    val conf = new SparkConf().setAppName("Data Formatting")
    val sc = new SparkContext(conf)
    val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)
    val batchId = args(0)
    val create_hive_table = """CREATE TABLE IF NOT EXISTS enriched_data
                              (
                              User_id STRING,
                              Song_id STRING,
                              Artist_id STRING,
                              Timestamp STRING,
                              Start_ts STRING,
                              End_ts STRING,
                              Geo_cd STRING,
                              Station_id STRING,
                              Song_end_type INT,
                              Like INT,
                              Dislike INT
                              )
                              PARTITIONED BY
                              (batchid INT,
                              status STRING)
                              STORED AS ORC
                              """
```



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



```
                    OR i.start_ts=''
                    OR i.end_ts=''
                    OR i.geo_cd=''
                    OR sg.geo_cd IS NULL
                    OR sg.geo_cd=''
                    OR sa.artist_id IS NULL
                    OR sa.artist_id='', 'fail', 'pass') AS status
                    FROM formatted_input i LEFT OUTER JOIN station_geo_map sg ON i.station_id = sg.station_id
                    LEFT OUTER JOIN song_artist_map sa ON i.song_id = sa.song_id
                    WHERE i.batchid=$batchId
                    """
try {

    sqlContext.sql( sqlText = "SET hive.auto.convert.join=false")
    sqlContext.sql( sqlText = "SET hive.exec.dynamic.partition.mode=nonstrict")
    sqlContext.sql( sqlText = "USE project")

    sqlContext.sql(create_hive_table)
    sqlContext.sql(load_data)
  }
  catch{
   case e: Exception=>e.printStackTrace()
  }
}
}
```

Code for the above spark application

```
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.sql


object DataEnrichment {
  def main(args: Array[String]): Unit = {
    val conf = new SparkConf().setAppName("Data Formatting")
    val sc = new SparkContext(conf)
    val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)
    val batchId = args(0)
    val create_hive_table = """CREATE TABLE IF NOT EXISTS enriched_data
                            (
                            User_id STRING,
                            Song_id STRING,
                            Artist_id STRING,
                            Timestamp STRING,
                            Start_ts STRING,
                            End_ts STRING,
                            Geo_cd STRING,
```

```
    Station_id STRING,
    Song_end_type INT,
    Like INT,
    Dislike INT
    )
    PARTITIONED BY
    (batchid INT,
    status STRING)
    STORED AS ORC
    """

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

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

    sqlContext.sql("SET hive.auto.convert.join=false")
    sqlContext.sql("SET hive.exec.dynamic.partition.mode=nonstrict")
    sqlContext.sql("USE project")

    sqlContext.sql(create_hive_table)
    sqlContext.sql(load_data)
   }
  catch{
   case e: Exception=>e.printStackTrace()
```

```
    }
 }
}
```

As we already created the **sbt** package of the above, spark application and other applications, which we are using in this project. So we will directly run the spark-submit script by using data**_enrichment.sh** script as shown below in screenshot.

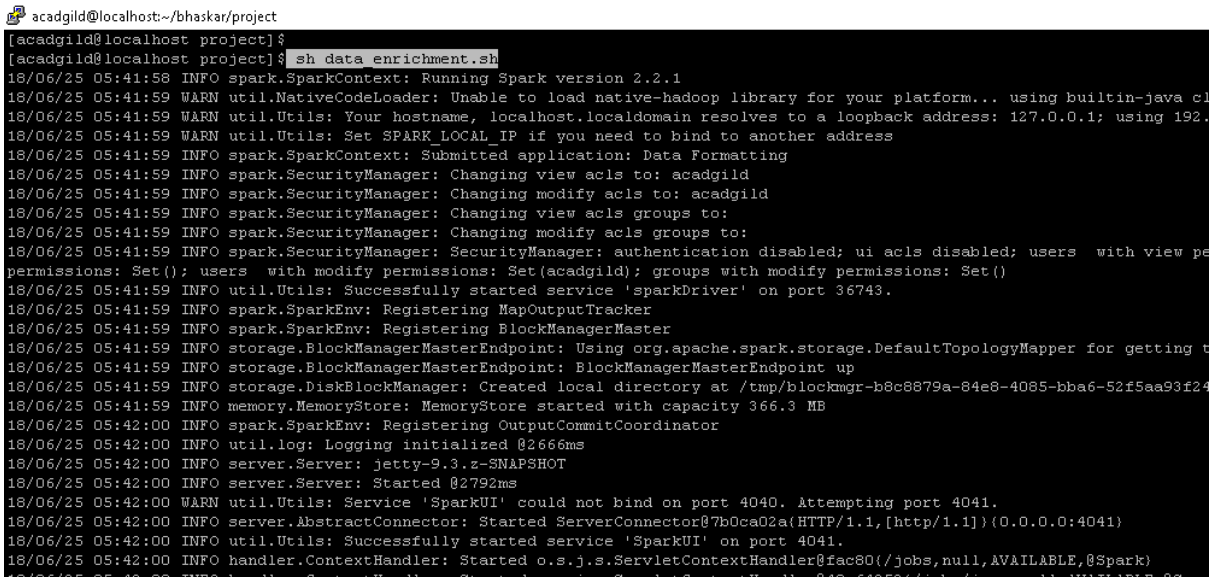
data_enrichment.sh

```
#!/bin/bash

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

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

spark-submit --class DataEnrichment --master local[2] --jars /home/acadgild/install/hive/apache-
hive-2.3.2-bin/lib/hive-hbase-handler-2.3.2.jar,/home/acadgild/install/hive/apache-hive-2.3.2-
bin/lib/hbase-client-1.1.1.jar,/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hbase-
common-1.1.1.jar,/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hbase-hadoop-
compat-1.1.1.jar,/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hbase-server-1.1.1.jar,/
home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hbase-protocol-1.1.1.jar,/home/acadgild/
install/hive/apache-hive-2.3.2-bin/lib/zookeeper-3.4.6.jar,/home/acadgild/install/hive/apache-
hive-2.3.2-bin/lib/guava-14.0.1.jar,/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/
htrace-core-3.1.0-incubating.jar /home/acadgild/bhaskar/project/MusicDataAnalysis/target/
scala-2.11/musicdataanalysis_2.11-1.0.jar $batchid

if [ ! -d "$VALIDDIR" ]
then
```

```
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/bhaskar/project/processed_dir/ -mtime +7 -exec rm {} \;
```

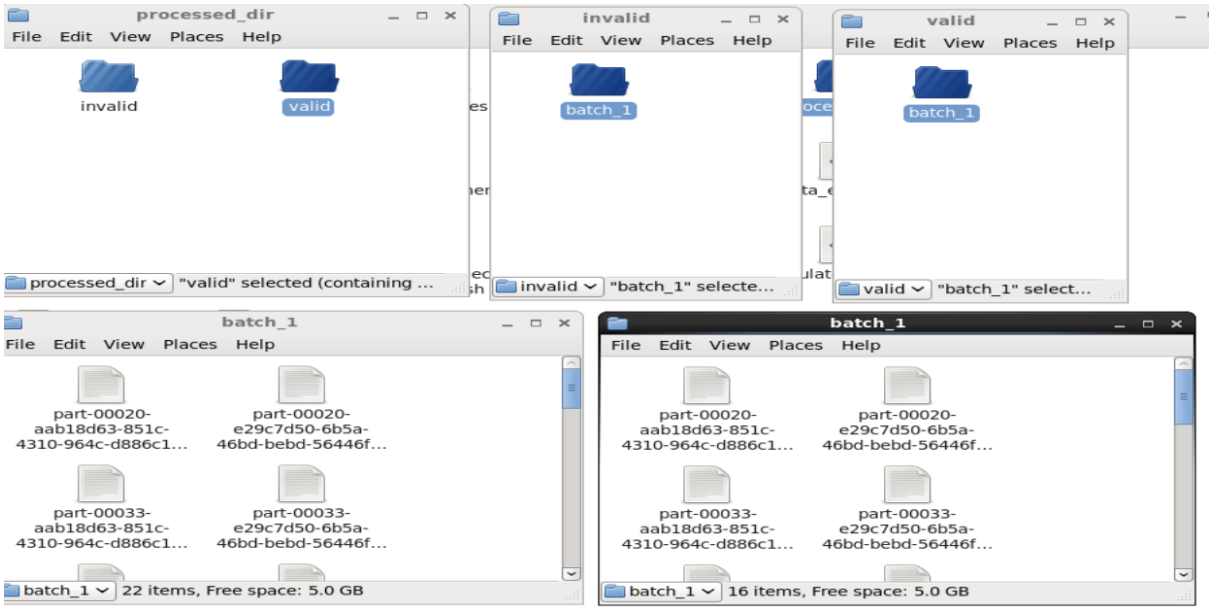
data_enrichment.sh script will create the **VALIDDIR** and **INVALIDDIR** for valid and invalid data respectively **processed_dir** in local file system and it will clear the records which are older than 7 days.

Below screenshot shows the running process of data enrichment.sh script

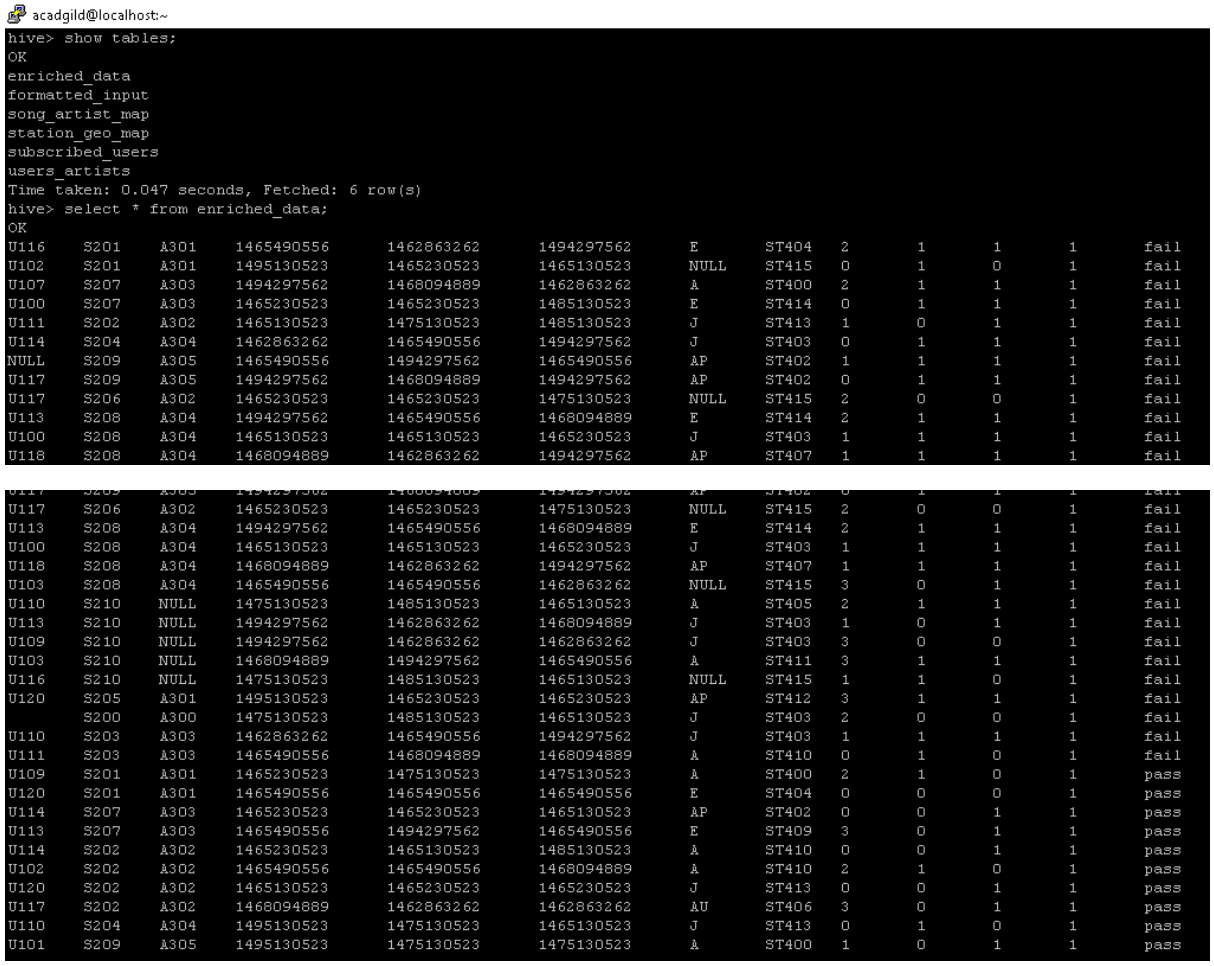


```
18/06/25 05:37:11 INFO scheduler.TaskSetManager: Finished task 82.0 in stage 4.0 (TID 283) in 56 ms on localhost (executor
18/06/25 05:37:11 INFO executor.Executor: Running task 84.0 in stage 4.0 (TID 285)
18/06/25 05:37:11 INFO storage.ShuffleBlockFetcherIterator: Getting 0 non-empty blocks out of 200 blocks
18/06/25 05:37:11 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 1 ms
18/06/25 05:37:11 INFO storage.ShuffleBlockFetcherIterator: Getting 0 non-empty blocks out of 1 blocks
18/06/25 05:37:11 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
18/06/25 05:37:11 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
18/06/25 05:37:11 INFO datasources.SQLHadoopMapReduceCommitProtocol: Using output committer class org.apache.hadoop.mapre
18/06/25 05:37:11 INFO mapred.SparkHadoopMapRedUtil: No need to commit output of task because needsTaskCommit=false: atte
18/06/25 05:37:11 INFO executor.Executor: Finished task 83.0 in stage 4.0 (TID 284). 4224 bytes result sent to driver
18/06/25 05:37:11 INFO scheduler.TaskSetManager: Starting task 85.0 in stage 4.0 (TID 286, localhost, executor driver, pa
18/06/25 05:37:11 INFO scheduler.TaskSetManager: Finished task 83.0 in stage 4.0 (TID 284) in 60 ms on localhost (executor
18/06/25 05:37:11 INFO executor.Executor: Running task 85.0 in stage 4.0 (TID 286)
18/06/25 05:37:11 INFO storage.ShuffleBlockFetcherIterator: Getting 0 non-empty blocks out of 200 blocks
18/06/25 05:37:11 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
18/06/25 05:37:11 INFO storage.ShuffleBlockFetcherIterator: Getting 0 non-empty blocks out of 1 blocks
18/06/25 05:37:11 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
18/06/25 05:37:11 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
18/06/25 05:37:11 INFO datasources.SQLHadoopMapReduceCommitProtocol: Using output committer class org.apache.hadoop.mapre
18/06/25 05:37:11 INFO storage.ShuffleBlockFetcherIterator: Getting 0 non-empty blocks out of 200 blocks
18/06/25 05:37:11 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 1 ms
18/06/25 05:37:11 INFO storage.ShuffleBlockFetcherIterator: Getting 0 non-empty blocks out of 1 blocks
18/06/25 05:37:11 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 1 ms
18/06/25 05:37:11 INFO mapred.SparkHadoopMapRedUtil: No need to commit output of task because needsTaskCommit=false: atte
18/06/25 05:37:11 INFO executor.Executor: Finished task 84.0 in stage 4.0 (TID 285). 4267 bytes result sent to driver
18/06/25 05:37:11 INFO scheduler.TaskSetManager: Starting task 86.0 in stage 4.0 (TID 287, localhost, executor driver, pa
18/06/25 05:37:11 INFO executor.Executor: Running task 86.0 in stage 4.0 (TID 287)
18/06/25 05:37:11 INFO scheduler.TaskSetManager: Finished task 84.0 in stage 4.0 (TID 285) in 95 ms on localhost (executor
18/06/25 05:37:11 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
18/06/25 05:37:11 INFO datasources.SQLHadoopMapReduceCommitProtocol: Using output committer class org.apache.hadoop.mapre
18/06/25 05:37:11 INFO mapred.SparkHadoopMapRedUtil: No need to commit output of task because needsTaskCommit=false: atte
18/06/25 05:37:11 INFO executor.Executor: Finished task 85.0 in stage 4.0 (TID 286). 4267 bytes result sent to driver
18/06/25 05:37:11 INFO scheduler.TaskSetManager: Starting task 88.0 in stage 4.0 (TID 288, localhost, executor driver, pa
18/06/25 05:37:11 INFO scheduler.TaskSetManager: Finished task 85.0 in stage 4.0 (TID 286) in 94 ms on localhost (executor
```

At the end of the above script, records will automatically divided based on pass and fail status and will dumped into processed_dir folder with the **VALIDDIR** and **INVALIDDIR** folders as shown below in screenshot.



To verify the **data_enrichment table,** check the tables in hive terminal as show below screenshot.



```
acadgild@localhost:~
hive> show tables;
OK
enriched_data
formatted_input
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 0.047 seconds, Fetched: 6 row(s)
hive> select * from enriched_data;
OK
U116    S201    A301    1465490556      1462863262      1494297562      E       ST404   2       1       1       1       fail
U102    S201    A301    1495130523      1465230523      1465130523      NULL    ST415   0       1       0       1       fail
U107    S207    A303    1494297562      1468094889      1462863262      A       ST400   2       1       1       1       fail
U100    S207    A303    1465230523      1465230523      1485130523      E       ST414   0       1       1       1       fail
U111    S202    A302    1465130523      1475130523      1485130523      J       ST413   1       0       1       1       fail
U114    S204    A304    1462863262      1465490556      1494297562      J       ST403   0       1       1       1       fail
NULL    S209    A305    1465490556      1494297562      1465490556      AP      ST402   1       1       1       1       fail
U117    S209    A305    1494297562      1468094889      1494297562      AP      ST402   0       1       1       1       fail
U117    S206    A302    1465230523      1465230523      1475130523      NULL    ST415   2       0       0       1       fail
U113    S208    A304    1494297562      1465490556      1468094889      E       ST414   2       1       1       1       fail
U100    S208    A304    1465130523      1465130523      1465230523      J       ST403   1       1       1       1       fail
U118    S208    A304    1468094889      1462863262      1494297562      AP      ST407   1       1       1       1       fail
U117    S206    A302    1465230523      1465230523      1475130523      NULL    ST415   2       0       0       1       fail
U113    S208    A304    1494297562      1465490556      1468094889      E       ST414   2       1       1       1       fail
U100    S208    A304    1465130523      1465130523      1465230523      J       ST403   1       1       1       1       fail
U118    S208    A304    1468094889      1462863262      1494297562      AP      ST407   1       1       1       1       fail
U103    S208    A304    1465490556      1465490556      1462863262      NULL    ST415   3       0       1       1       fail
U110    S210    NULL    1475130523      1485130523      1465130523      A       ST405   2       1       1       1       fail
U113    S210    NULL    1494297562      1462863262      1468094889      J       ST403   1       0       1       1       fail
U109    S210    NULL    1494297562      1462863262      1462863262      J       ST403   3       0       0       1       fail
U103    S210    NULL    1468094889      1494297562      1465490556      A       ST411   3       1       1       1       fail
U116    S210    NULL    1475130523      1485130523      1465130523      NULL    ST415   1       1       0       1       fail
U120    S205    A301    1495130523      1465230523      1465230523      AP      ST412   3       1       1       1       fail
        S200    A300    1475130523      1485130523      1465130523      J       ST403   2       0       0       1       fail
U110    S203    A303    1462863262      1465490556      1494297562      J       ST403   1       1       1       1       fail
U111    S203    A303    1465490556      1468094889      1468094889      A       ST410   0       1       0       1       fail
U109    S201    A301    1465230523      1475130523      1475130523      A       ST400   2       1       0       1       pass
U120    S201    A301    1465490556      1465490556      1465490556      E       ST404   0       0       0       1       pass
U114    S207    A303    1465230523      1465230523      1465130523      AP      ST402   0       0       1       1       pass
U113    S207    A303    1465490556      1494297562      1465490556      E       ST409   3       0       1       1       pass
U114    S202    A302    1465230523      1465130523      1485130523      A       ST410   0       0       1       1       pass
U102    S202    A302    1465490556      1465490556      1468094889      A       ST410   2       1       0       1       pass
U120    S202    A302    1465130523      1465230523      1465230523      J       ST413   0       0       1       1       pass
U117    S202    A302    1468094889      1462863262      1462863262      AU      ST406   3       0       1       1       pass
U110    S204    A304    1495130523      1475130523      1465130523      J       ST413   0       1       0       1       pass
U101    S209    A305    1495130523      1475130523      1475130523      A       ST400   1       0       1       1       pass
```

Above screenshot shows data of **enrichment_data,** which shows that**,** the enrichment rules applied on the data. We will able to see that one extra column added in the table.

This extra column divides the data into fail and pass category.

The data having **pass** category will copy to the **VALIDDIR** and **fail** category data copied to **INVALIDDIR** directory of local file system.

This copy will maintained for 7 days and then the same script will remove it.

By this enrichment part is successfully executed by applying all the enrichment rules.

6. Data Analysis:

After completing the enrichment of data now, we have to analyse the data. So we have following requirement by the company which will be filled by the analysing it hive queries.

A) Determine **top 10 station_id(s)** where maximum number of songs played, which unique users liked.

B) 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.

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

D) 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.

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

To solve the above problem we have a spark application **DataAnalysis.scala** which will create the hive tables as per the above five problems. Hive table will created with the help of **sparkcontext** object.

Below screenshots, shows the spark-application **DataAnalysis.scala**

DataAnalysis.scala | build.sbt

```
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.sql

object DataAnalysis {
  def main(args: Array[String]): Unit = {
    val conf = new SparkConf().setAppName("Data Analysis")
    val sc = new SparkContext(conf)
    val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)
    val batchId = args(0)


val create_top_10_stations = """CREATE TABLE IF NOT EXISTS 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 load_top_10_stations = s"""INSERT OVERWRITE TABLE top_10_stations
PARTITION(batchid='$batchId')
SELECT
station_id,
COUNT(DISTINCT song_id) AS total_distinct_songs_played,
COUNT(DISTINCT user_id) AS distinct_user_count
FROM enriched_data
WHERE status='pass'
AND batchid='$batchId'
AND like=1
GROUP BY station_id
ORDER BY total_distinct_songs_played DESC
LIMIT 10"""


val create_users_behaviour = """CREATE TABLE IF NOT EXISTS users_behaviour
(
user_type STRING,
duration INT
)
PARTITIONED BY (batchid INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE"""
```

```
val load_users_behaviour = s"""INSERT OVERWRITE TABLE users_behaviour
PARTITION(batchid='$batchId')
SELECT
CASE WHEN (su.user_id IS NULL OR CAST(ed.timestamp AS DECIMAL(20,0)) > CAST(su.subscn_end_dt AS DECIMAL(20,0))) THEN 'UNSUBSCRIBED'
WHEN (su.user_id IS NOT NULL AND CAST(ed.timestamp AS DECIMAL(20,0)) <= CAST(su.subscn_end_dt AS DECIMAL(20,0))) THEN 'SUBSCRIBED'
END AS user_type,
SUM(ABS(CAST(ed.end_ts AS DECIMAL(20,0))-CAST(ed.start_ts AS DECIMAL(20,0)))) AS duration
FROM enriched_data ed
LEFT OUTER JOIN subscribed_users su
ON ed.user_id=su.user_id
WHERE ed.status='pass'
AND ed.batchid='$batchId'
GROUP BY CASE WHEN (su.user_id IS NULL OR CAST(ed.timestamp AS DECIMAL(20,0)) > CAST(su.subscn_end_dt AS DECIMAL(20,0))) THEN 'UNSUBSCRIBED'
WHEN (su.user_id IS NOT NULL AND CAST(ed.timestamp AS DECIMAL(20,0)) <= CAST(su.subscn_end_dt AS DECIMAL(20,0))) THEN 'SUBSCRIBED' END"""


val create_connected_artists = """CREATE TABLE IF NOT EXISTS connected_artists
(
artist_id STRING,
user_count INT
)
PARTITIONED BY (batchid INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE"""
```

```
val load_connected_artists = s"""INSERT OVERWRITE TABLE connected_artists
PARTITION(batchid='$batchId')
SELECT
ua.artist_id,
COUNT(DISTINCT ua.user_id) AS user_count
FROM
(
SELECT user_id, artist_id FROM users_artists
LATERAL VIEW explode(artists_array) artists AS artist_id
) ua
INNER JOIN
(
SELECT artist_id, song_id, user_id
FROM enriched_data
WHERE status='pass'
AND batchid='$batchId'
) ed
ON ua.artist_id=ed.artist_id
AND ua.user_id=ed.user_id
GROUP BY ua.artist_id
ORDER BY user_count DESC
LIMIT 10"""
```

```
val create_top_10_royalty_songs = """CREATE TABLE IF NOT EXISTS top_10_royalty_songs
(
song_id STRING,
duration INT
)
PARTITIONED BY (batchid INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE"""

val load_top_10_royalty_songs = s"""INSERT OVERWRITE TABLE top_10_royalty_songs
PARTITION(batchid='$batchId')
SELECT song_id,
SUM(ABS(CAST(end_ts AS DECIMAL(20,0))-CAST(start_ts AS DECIMAL(20,0)))) AS duration
FROM enriched_data
WHERE status='pass'
AND batchid='$batchId'
AND (like=1 OR song_end_type=0)
GROUP BY song_id
ORDER BY duration DESC
LIMIT 10"""
```

```
120
121  val create_top_10_unsubscribed_users = """CREATE TABLE IF NOT EXISTS top_10_unsubscribed_users
122  (
123  user_id STRING,
124  duration INT
125  )
126  PARTITIONED BY (batchid INT)
127  ROW FORMAT DELIMITED
128  FIELDS TERMINATED BY ','
129  STORED AS TEXTFILE"""
130
131  val load_top_10_unsubscribed_users = s"""INSERT OVERWRITE TABLE top_10_unsubscribed_users
132  PARTITION(batchid='$batchId')
133  SELECT
134  ed.user_id,
135  SUM(ABS(CAST(ed.end_ts AS DECIMAL(20,0))-CAST(ed.start_ts AS DECIMAL(20,0)))) AS duration
136  FROM enriched_data ed
137  LEFT OUTER JOIN subscribed_users su
138  ON ed.user_id=su.user_id
139  WHERE ed.status='pass'
140  AND ed.batchid='$batchId'
141  AND (su.user_id IS NULL OR (CAST(ed.timestamp AS DECIMAL(20,0)) > CAST(su.subscn_end_dt AS DECIMAL(20,0))))
142  GROUP BY ed.user_id
143  ORDER BY duration DESC
144  LIMIT 10"""
145
```

```
144  LIMIT 10"""
145
146
147      try {
148           sqlContext.sql( sqlText = "SET hive.auto.convert.join=false")
149           sqlContext.sql( sqlText = "USE project")
150           sqlContext.sql(create_top_10_stations)
151           sqlContext.sql(load_top_10_stations)
152           sqlContext.sql(create_users_behaviour)
153           sqlContext.sql(load_users_behaviour)
154           sqlContext.sql(create_connected_artists)
155           sqlContext.sql(load_connected_artists)
156           sqlContext.sql(create_top_10_royalty_songs)
157           sqlContext.sql(load_top_10_royalty_songs)
158           sqlContext.sql(create_top_10_unsubscribed_users)
159           sqlContext.sql(load_top_10_unsubscribed_users)
160         }
161        catch{
162         case e: Exception=>e.printStackTrace()
163        }
164   }
165  }
```

Code for above spark-application

```
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.sql

object DataAnalysis {
  def main(args: Array[String]): Unit = {
    val conf = new SparkConf().setAppName("Data Analysis")
```

```
    val sc = new SparkContext(conf)
    val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)
    val batchId = args(0)



val create_top_10_stations = """CREATE TABLE IF NOT EXISTS 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 load_top_10_stations = s"""INSERT OVERWRITE TABLE top_10_stations
PARTITION(batchid='$batchId')
SELECT
station_id,
COUNT(DISTINCT song_id) AS total_distinct_songs_played,
COUNT(DISTINCT user_id) AS distinct_user_count
FROM enriched_data
WHERE status='pass'
AND batchid='$batchId'
AND like=1
GROUP BY station_id
ORDER BY total_distinct_songs_played DESC
LIMIT 10"""
```

```
val create_users_behaviour = """CREATE TABLE IF NOT EXISTS users_behaviour
(
user_type STRING,
duration INT
)
PARTITIONED BY (batchid INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE"""

val load_users_behaviour = s"""INSERT OVERWRITE TABLE users_behaviour
PARTITION(batchid='$batchId')
SELECT
CASE WHEN (su.user_id IS NULL OR CAST(ed.timestamp AS DECIMAL(20,0)) > CAST(su.subscn_end_dt AS DECIMAL(20,0))) THEN 'UNSUBSCRIBED'
WHEN (su.user_id IS NOT NULL AND CAST(ed.timestamp AS DECIMAL(20,0)) <= CAST(su.subscn_end_dt AS DECIMAL(20,0))) THEN 'SUBSCRIBED'
END AS user_type,
SUM(ABS(CAST(ed.end_ts AS DECIMAL(20,0))-CAST(ed.start_ts AS DECIMAL(20,0)))) AS duration
FROM enriched_data ed
LEFT OUTER JOIN subscribed_users su
ON ed.user_id=su.user_id
WHERE ed.status='pass'
AND ed.batchid='$batchId'
GROUP BY CASE WHEN (su.user_id IS NULL OR CAST(ed.timestamp AS DECIMAL(20,0)) > CAST(su.subscn_end_dt AS DECIMAL(20,0))) THEN 'UNSUBSCRIBED'
WHEN (su.user_id IS NOT NULL AND CAST(ed.timestamp AS DECIMAL(20,0)) <= CAST(su.subscn_end_dt AS DECIMAL(20,0))) THEN 'SUBSCRIBED' END"""
```

```
val create_connected_artists = """CREATE TABLE IF NOT EXISTS connected_artists
(
artist_id STRING,
user_count INT
)
PARTITIONED BY (batchid INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE"""

val load_connected_artists = s"""INSERT OVERWRITE TABLE connected_artists
PARTITION(batchid='$batchId')
SELECT
ua.artist_id,
COUNT(DISTINCT ua.user_id) AS user_count
FROM
(
SELECT user_id, artist_id FROM users_artists
LATERAL VIEW explode(artists_array) artists AS artist_id
) ua
INNER JOIN
(
SELECT artist_id, song_id, user_id
FROM enriched_data
WHERE status='pass'
AND batchid='$batchId'
) ed
ON ua.artist_id=ed.artist_id
AND ua.user_id=ed.user_id
```

```
GROUP BY ua.artist_id
ORDER BY user_count DESC
LIMIT 10"""

val create_top_10_royalty_songs = """CREATE TABLE IF NOT EXISTS top_10_royalty_songs
(
song_id STRING,
duration INT
)
PARTITIONED BY (batchid INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE"""

val load_top_10_royalty_songs = s"""INSERT OVERWRITE TABLE top_10_royalty_songs
PARTITION(batchid='$batchId')
SELECT song_id,
SUM(ABS(CAST(end_ts AS DECIMAL(20,0))-CAST(start_ts AS DECIMAL(20,0)))) AS duration
FROM enriched_data
WHERE status='pass'
AND batchid='$batchId'
AND (like=1 OR song_end_type=0)
GROUP BY song_id
ORDER BY duration DESC
LIMIT 10"""

val create_top_10_unsubscribed_users = """CREATE TABLE IF NOT EXISTS
top_10_unsubscribed_users
```

```
(
user_id STRING,
duration INT
)
PARTITIONED BY (batchid INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE"""

val load_top_10_unsubscribed_users = s"""INSERT OVERWRITE TABLE top_10_unsubscribed_users
PARTITION(batchid='$batchId')
SELECT
ed.user_id,
SUM(ABS(CAST(ed.end_ts AS DECIMAL(20,0))-CAST(ed.start_ts AS DECIMAL(20,0)))) AS duration
FROM enriched_data ed
LEFT OUTER JOIN subscribed_users su
ON ed.user_id=su.user_id
WHERE ed.status='pass'
AND ed.batchid='$batchId'
AND (su.user_id IS NULL OR (CAST(ed.timestamp AS DECIMAL(20,0)) > CAST(su.subscn_end_dt AS DECIMAL(20,0))))
GROUP BY ed.user_id
ORDER BY duration DESC
LIMIT 10"""
    try {
        sqlContext.sql("SET hive.auto.convert.join=false")
        sqlContext.sql("USE project")
        sqlContext.sql(create_top_10_stations)
```

```
sqlContext.sql(load_top_10_stations)
sqlContext.sql(create_users_behaviour)
sqlContext.sql(load_users_behaviour)
sqlContext.sql(create_connected_artists)
sqlContext.sql(load_connected_artists)
sqlContext.sql(create_top_10_royalty_songs)
sqlContext.sql(load_top_10_royalty_songs)
sqlContext.sql(create_top_10_unsubscribed_users)
sqlContext.sql(load_top_10_unsubscribed_users)



sqlContext.sql("SELECT station_id FROM top_10_stations").show()
sqlContext.sql("SELECT user_type,duration FROM users_behaviour").show()
sqlContext.sql("SELECT artist_id FROM connected_artists").show()
sqlContext.sql("SELECT song_id FROM top_10_royalty_songs").show()
sqlContext.sql("SELECT user_id FROM top_10_unsubscribed_users ").show()
}
catch{
case e: Exception=>e.printStackTrace()
}}}
```

To run the above spark application we have **data_analysis.sh** script which will run the spark-submit job as shown below in the screenshot.

data_analysis.sh

```
#!/bin/bash

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

echo "Running hive script for data analysis..." >> $LOGFILE

spark-submit --class DataAnalysis --master local[2] --jars /home/acadgild/install/hive/apache-
hive-2.3.2-bin/lib/hive-hbase-handler-2.3.2.jar,/home/acadgild/install/hive/apache-hive-2.3.2-bin/
lib/hbase-client-1.1.1.jar,/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hbase-
common-1.1.1.jar,/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hbase-hadoop-
compat-1.1.1.jar,/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hbase-server-1.1.1.jar,/
home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hbase-protocol-1.1.1.jar,/home/acadgild/
install/hive/apache-hive-2.3.2-bin/lib/zookeeper-3.4.6.jar,/home/acadgild/install/hive/apache-
hive-2.3.2-bin/lib/guava-14.0.1.jar,/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/htrace-
core-3.1.0-incubating.jar /home/acadgild/bhaskar/project/MusicDataAnalysis/target/scala-2.11/
musicdataanalysis_2.11-1.0.jar $batchid

echo "Incrementing batchid..." >> $LOGFILE
```

Now run the above shell script file.

```
atchid=1/part-00000-132d05c7-4860-4b6f-bb1f-31482d2ec353-c000, Status:true
18/06/25 06:55:23 INFO execution.SparkSqlParser: Parsing command: `project`.`top_10_royalty_songs`
18/06/25 06:55:23 INFO parser.CatalystSqlParser: Parsing command: int
18/06/25 06:55:23 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 06:55:23 INFO parser.CatalystSqlParser: Parsing command: int
18/06/25 06:55:23 INFO execution.SparkSqlParser: Parsing command: CREATE TABLE IF NOT EXISTS top_10_unsubscribed_users
(
user_id STRING,
duration INT
)
PARTITIONED BY (batchid INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
18/06/25 06:55:24 INFO execution.SparkSqlParser: Parsing command: INSERT OVERWRITE TABLE top_10_unsubscribed_users
PARTITION(batchid='1')
SELECT
ed.user_id,
SUM(ABS(CAST(ed.end_ts AS DECIMAL(20,0))-CAST(ed.start_ts AS DECIMAL(20,0)))) AS duration
FROM enriched_data ed
LEFT OUTER JOIN subscribed_users su
ON ed.user_id=su.user_id
WHERE ed.status='pass'
AND ed.batchid='1'
AND (su.user_id IS NULL OR (CAST(ed.timestamp AS DECIMAL(20,0)) > CAST(su.subscn_end_dt AS DECIMAL(20,0))))
GROUP BY ed.user_id
ORDER BY duration DESC
LIMIT 10
18/06/25 06:55:24 INFO parser.CatalystSqlParser: Parsing command: int
```

acadgild@localhost:~/bhaskar/project

```
distinct_user_count INT
)
PARTITIONED BY (batchid INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
18/06/25 10:08:04 INFO parser.CatalystSqlParser: Parsing command: array<string>
18/06/25 10:08:05 INFO execution.SparkSqlParser: Parsing command: INSERT OVERWRITE TABLE top_10_stations
PARTITION(batchid='1')
SELECT
station_id,
COUNT(DISTINCT song_id) AS total_distinct_songs_played,
COUNT(DISTINCT user_id) AS distinct_user_count
FROM enriched_data
WHERE status='pass'
AND batchid='1'
AND like=1
GROUP BY station_id
ORDER BY total_distinct_songs_played DESC
LIMIT 10
18/06/25 10:08:05 INFO parser.CatalystSqlParser: Parsing command: int
18/06/25 10:08:05 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 10:08:05 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 10:08:05 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 10:08:05 INFO parser.CatalystSqlParser: Parsing command: string
```

```
18/06/25 06:55:28 INFO execution.SparkSqlParser: Parsing command: `project`.`top_10_unsubscribed_users`
18/06/25 06:55:28 INFO parser.CatalystSqlParser: Parsing command: int
18/06/25 06:55:28 INFO parser.CatalystSqlParser: Parsing command: string
18/06/25 06:55:28 INFO parser.CatalystSqlParser: Parsing command: int
18/06/25 06:55:28 INFO spark.SparkContext: Invoking stop() from shutdown hook
18/06/25 06:55:28 INFO server.AbstractConnector: Stopped Spark@453aa2fa{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}
18/06/25 06:55:28 INFO ui.SparkUI: Stopped Spark web UI at http://192.168.0.26:4040
18/06/25 06:55:28 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/06/25 06:55:28 INFO memory.MemoryStore: MemoryStore cleared
18/06/25 06:55:28 INFO storage.BlockManager: BlockManager stopped
18/06/25 06:55:28 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
18/06/25 06:55:28 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/06/25 06:55:28 INFO spark.SparkContext: Successfully stopped SparkContext
18/06/25 06:55:28 INFO util.ShutdownHookManager: Shutdown hook called
18/06/25 06:55:28 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-87444ee0-8eaa-4f05-b84d-3965a11eb374
```

a) Top **station_id** where maximum number of songs played, which unique users liked.

```
18/06/25 10:08:44 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 20.0 (TID 1647) in 10 ms on localhost
18/06/25 10:08:44 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 20.0, whose tasks have all completed, from pool
18/06/25 10:08:44 INFO scheduler.DAGScheduler: ResultStage 20 (show at DataAnalysis.scala:161) finished in 0.007 s
18/06/25 10:08:44 INFO scheduler.DAGScheduler: Job 6 finished: show at DataAnalysis.scala:161, took 0.016440 s
+----------+
|station_id|
+----------+
|     ST413|
|     ST400|
|     ST414|
|     ST410|
|     ST411|
|     ST406|
+----------+

18/06/25 10:08:44 INFO execution.SparkSqlParser: Parsing command: SELECT user_type,duration FROM users_behaviour
```

b) Total duration of songs played by each type of user, where type of user can be '**subscribed**' or '**unsubscribed**'.

```
18/06/26 10:00:29 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 34.0, whose tasks have all completed, fro
18/06/26 10:00:29 INFO scheduler.DAGScheduler: ResultStage 34 (show at DataAnalysis.scala:168) finished in 0.
18/06/26 10:00:29 INFO scheduler.DAGScheduler: Job 20 finished: show at DataAnalysis.scala:168, took 0.490992
+------------+-----------+
|   user_type|   duration|
+------------+-----------+
|UNSUBSCRIBED| -766123647|
|  SUBSCRIBED|-1693340827|
+------------+-----------+

18/06/26 10:00:29 INFO execution.SparkSqlParser: Parsing command: SELECT artist_id FROM connected_artists
```

c) Top **connected artists** whose songs are most listened by the unique users who follow them.

```
18/06/25 10:08:45 INFO scheduler.DAGScheduler: ResultStage 26 (show at DataAnalysis.scala:163) finished in 0.005 s
18/06/25 10:08:45 INFO scheduler.DAGScheduler: Job 12 finished: show at DataAnalysis.scala:163, took 0.024587 s
+---------+
|artist_id|
+---------+
|     A302|
+---------+

18/06/25 10:08:45 INFO execution.SparkSqlParser: Parsing command: SELECT song_id FROM top_10_royalty_songs
```

d) Top songs who gave **maximum revenues** and **Royalty** applies on it.

```
18/06/26 10:00:30 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 36.0, whose tasks have all completed, from po
18/06/26 10:00:30 INFO scheduler.DAGScheduler: ResultStage 36 (show at DataAnalysis.scala:170) finished in 0.012
18/06/26 10:00:30 INFO scheduler.DAGScheduler: Job 22 finished: show at DataAnalysis.scala:170, took 0.027356 s
+-------+
|song_id|
+-------+
|   S200|
|   S209|
|   S203|
|   S206|
|   S202|
|   S205|
|   S201|
|   S208|
|   S207|
|   S204|
+-------+

18/06/26 10:00:30 INFO execution.SparkSqlParser: Parsing command: SELECT user_id FROM top_10_unsubscribed_users
```

e) Top **unsubscribed users** who listened to the songs for the **longest duration.**

```
18/06/26 10:00:30 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 37.0, whose tasks h
18/06/26 10:00:30 INFO scheduler.DAGScheduler: ResultStage 37 (show at DataAnalysis.sca
18/06/26 10:00:30 INFO scheduler.DAGScheduler: Job 23 finished: show at DataAnalysis.sc
+-------+
|user_id|
+-------+
|   U108|
|   U107|
|   U116|
|   U117|
|   U120|
|   U118|
|   U115|
|   U119|
|   U100|
|   U110|
+-------+

18/06/26 10:00:30 INFO spark.SparkContext: Invoking stop() from shutdown hook
```

Below screenshot shows the output of the each spark query in tables as per required queries for data analysis.

```
acadgild@localhost:~
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: 0.037 seconds, Fetched: 11 row(s)
hive>
```

Now let see the output of the spark analysis in hive table as shown below in the screen shot.

```
hive>
hive> select * from top_10_stations;
OK
ST405   10      11      1
ST404   9       12      1
ST410   9       12      1
ST408   9       10      1
ST411   8       12      1
ST402   7       8       1
ST406   7       11      1
ST414   6       9       1
ST409   6       8       1
ST401   6       8       1
Time taken: 0.232 seconds, Fetched: 10 row(s)
hive> select * from connected_artists;
OK
A302    2       1
Time taken: 0.239 seconds, Fetched: 1 row(s)
hive> select * from
hive> select * from top_10_royalty_songs;
OK
S200    346549552       1
S209    324396065       1
S203    322950199       1
S206    317089845       1
S202    296001467       1
S205    267393448       1
S201    261721528       1
S208    251839873       1
S207    241198567       1
S204    210082612       1
Time taken: 0.187 seconds, Fetched: 10 row(s)
```

```
Time taken: 0.187 seconds, Fetched: 10 row(s)
hive>
hive> select * from top_10_unsubscribed_users;
OK
U108    398480620       1
U107    335283122       1
U116    333537427       1
U117    328382839       1
U120    316684272       1
U118    292852225       1
U115    280746879       1
U119    256725578       1
U100    212832867       1
U110    105395894       1
Time taken: 0.237 seconds, Fetched: 10 row(s)
hive> select * from users_behaviour;
OK
UNSUBSCRIBED    -766123647      1
SUBSCRIBED      -1693340827     1
Time taken: 0.176 seconds, Fetched: 2 row(s)
hive>
```

From above screenshot, we have seen all the spark queries creating the tables for each query. So Data Analysis using Spark executed successfully.

7. Data Exportation:

In this stage, we have to export the result of analysis in RDBMS for data storage and quick retrieval. These data will then form visualisations on top of analysed data and send data to Data Science or machine learning pipelines for further forecast.

Therefore, we have to export all the spark query tables from HIVE warehouse directory to MySQL database using **SQOOP** scripts as shown below in **data_export_toMySQL.sh** script.

Data_export_toMySQL.sh will first create the mysql tables or schema of the tables required for analysed data in hive table by using **create_schema.sql** script.

create_schema.sql is general sql script has all the five tables creation queries which create the table in database project in MySQL database.

Below screenshot, show the **export_data_toMySQL.sh** script.

data_export_toMySQL.sh

```
#!/bin/bash

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

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

mysql -u root -p  < /home/acadgild/bhaskar/project/create_schema.sql

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

sqoop export \
--connect jdbc:mysql://localhost/project \
--username 'root' \
--password-file file:///home/acadgild/bhaskar/project/mysql_password \
--table top_10_stations \
--export-dir /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-file file:///home/acadgild/bhaskar/project/mysql_password \
--table users_behaviour \
--export-dir /user/hive/warehouse/project.db/users_behaviour/batchid=$batchid
\
--input-fields-terminated-by ',' \
-m 1
```

data_export_toMySQL.sh

```
--export-dir /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-file file:///home/acadgild/bhaskar/project/mysql_password \
--table connected_artists \
--export-dir /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-file file:///home/acadgild/bhaskar/project/mysql_password \
--table top_10_royalty_songs \
--export-dir /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-file file:///home/acadgild/bhaskar/project/mysql_password \
--table top_10_unsubscribed_users \
--export-dir /user/hive/warehouse/project.db/top_10_unsubscribed_users/batchid=$batchid \
--input-fields-terminated-by ',' \
-m 1

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

Sqoop commands to export the tables

Export command to transfer top 10 stations to similar table schema top 10 stations in MySQL

sqoop export \

--connect jdbc:mysql://localhost/project \

--username 'root' \

--password-file file:///home/acadgild/bhaskar/project/mysql_password \

--table top_10_stations \

--export-dir /user/hive/warehouse/project.db/top_10_stations/batchid=$batchid \

--input-fields-terminated-by ',' \

-m 1

Export command to transfer users behaviour to similar table schema users behaviour in MySQL

sqoop export \

--connect jdbc:mysql://localhost/project \

--username 'root' \

```
--password-file file:///home/acadgild/bhaskar/project/mysql_password \
--table users_behaviour \
--export-dir /user/hive/warehouse/project.db/users_behaviour/batchid=$batchid \
--input-fields-terminated-by ',' \
-m 1
```

Export command to transfer connected_artists to similar table schema connected_artists in MySQL

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

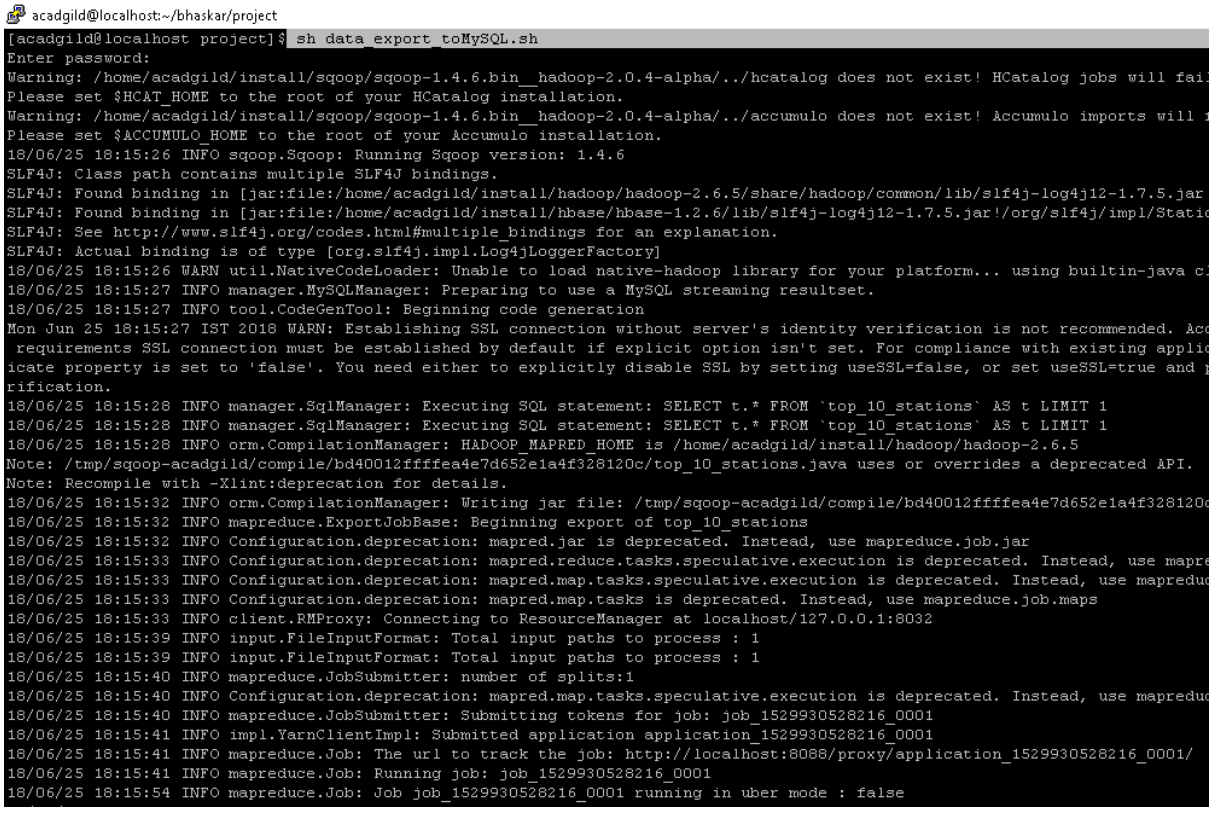
Export command to transfer table top_10_royalty_songs to similar table schema table top_10_royalty_songs in MySQL

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

Export command to transfer table table top_10_unsubscribed_users to similar table schema table table top_10_unsubscribed_users in MySQL

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

Now let's run the **export_data_toMySQL.sh** script as shown below in the screenshot



```
18/06/25 18:16:02 INFO mapreduce.Job: Job job_1529930528216_0001 completed successfully
18/06/25 18:16:02 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=127642
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=276
                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)=4947
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=4947
                Total vcore-milliseconds taken by all map tasks=4947
                Total megabyte-milliseconds taken by all map tasks=5065728
        Map-Reduce Framework
                Map input records=6
                Map output records=6
                Input split bytes=213
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=113
                CPU time spent (ms)=1830
                Physical memory (bytes) snapshot=169279488
                Virtual memory (bytes) snapshot=2084868096
                Total committed heap usage (bytes)=119013376
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=0
18/06/25 18:16:02 INFO mapreduce.ExportJobBase: Transferred 276 bytes in 29.0395 seconds (9.5043 bytes/sec)
18/06/25 18:16:02 INFO mapreduce.ExportJobBase: Exported 6 records.
```

acadgild@localhost:~/bhaskar/project

```
18/06/25 18:16:31 INFO mapreduce.Job:  map 50% reduce 0%
18/06/25 18:16:33 INFO mapreduce.Job:  map 100% reduce 0%
18/06/25 18:16:33 INFO mapreduce.Job: Job job_1529930528216_0002 completed successfully
18/06/25 18:16:33 INFO mapreduce.Job: Counters: 31
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=255204
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=29976
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=602
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=0
        Job Counters
                Launched map tasks=2
                Other local map tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=17797
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=17797
                Total vcore-milliseconds taken by all map tasks=17797
                Total megabyte-milliseconds taken by all map tasks=18224128
        Map-Reduce Framework
                Map input records=2
                Map output records=2
                Input split bytes=29928
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=293
                CPU time spent (ms)=4790
                Physical memory (bytes) snapshot=340824064
                Virtual memory (bytes) snapshot=4169449472
                Total committed heap usage (bytes)=238551040
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=0
18/06/25 18:16:33 INFO mapreduce.ExportJobBase: Transferred 29.2734 KB in 25.0316 seconds (1.1695 KB/sec)
```

acadgild@localhost:~/bhaskar/project

```
18/06/25 18:17:00 INFO mapreduce.Job: Job job_1529930528216_0003 completed successfully
18/06/25 18:17:00 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=127612
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=225
                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)=4342
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=4342
                Total vcore-milliseconds taken by all map tasks=4342
                Total megabyte-milliseconds taken by all map tasks=4446208
        Map-Reduce Framework
                Map input records=1
                Map output records=1
                Input split bytes=215
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=82
                CPU time spent (ms)=1800
                Physical memory (bytes) snapshot=169697280
                Virtual memory (bytes) snapshot=2084958208
                Total committed heap usage (bytes)=122159104
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=0
18/06/25 18:17:00 INFO mapreduce.ExportJobBase: Transferred 225 bytes in 20.6111 seconds (10.9165 bytes/sec)
18/06/25 18:17:00 INFO mapreduce.ExportJobBase: Exported 1 records.
```

acadgild@localhost:~/bhaskar/project

```
18/06/25 18:17:26 INFO mapreduce.Job:  map 100% reduce 0%
18/06/25 18:17:26 INFO mapreduce.Job: Job job_1529930528216_0004 completed successfully
18/06/25 18:17:26 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=127620
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=310
                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)=4441
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=4441
                Total vcore-milliseconds taken by all map tasks=4441
                Total megabyte-milliseconds taken by all map tasks=4547584
        Map-Reduce Framework
                Map input records=7
                Map output records=7
                Input split bytes=218
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=95
                CPU time spent (ms)=1740
                Physical memory (bytes) snapshot=168808448
                Virtual memory (bytes) snapshot=2085560320
                Total committed heap usage (bytes)=108003328
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=0
18/06/25 18:17:26 INFO mapreduce.ExportJobBase: Transferred 310 bytes in 20.6233 seconds (15.0315 bytes/sec)
18/06/25 18:17:26 INFO mapreduce.ExportJobBase: Exported 7 records.
```

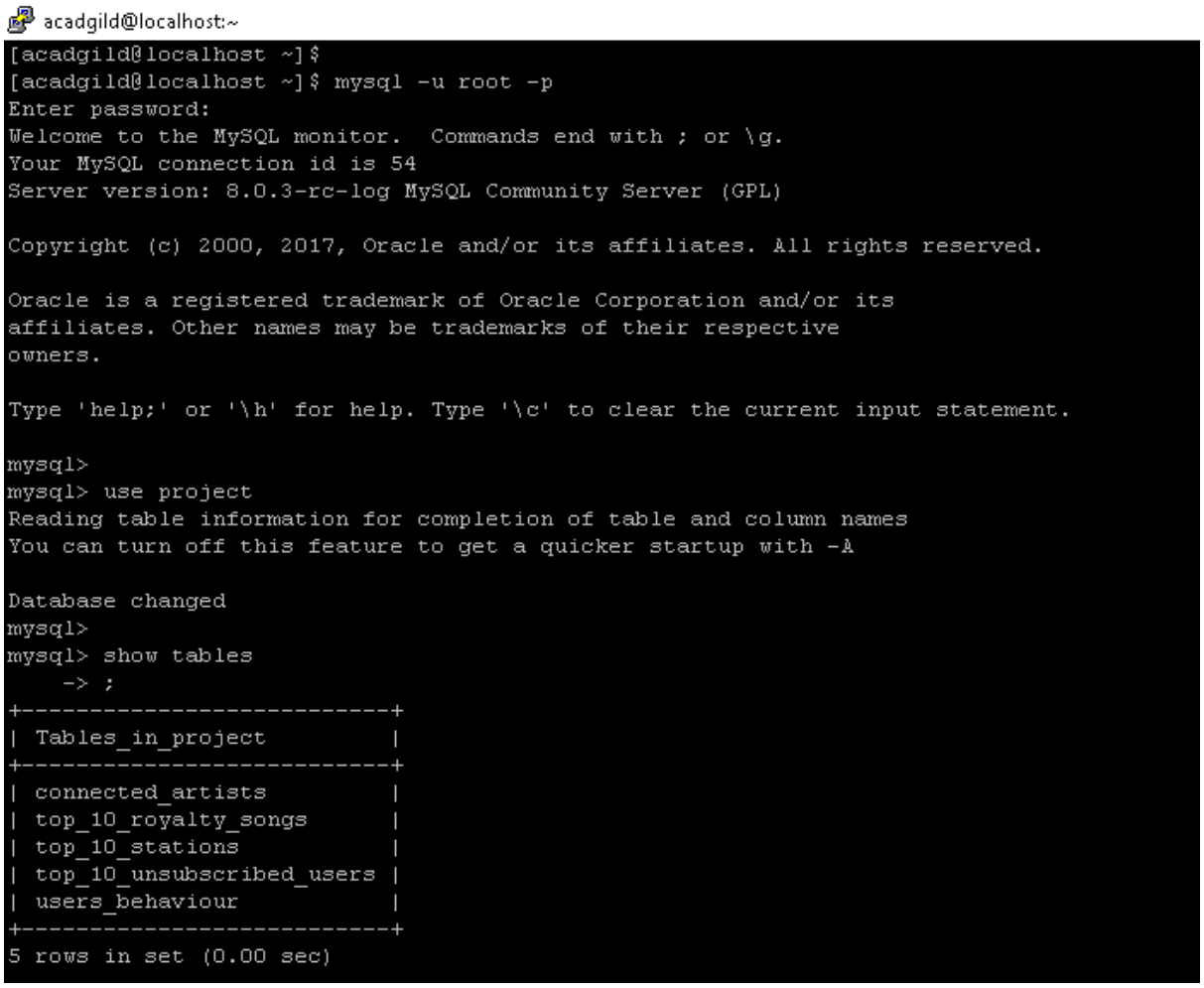
```
18/06/25 18:17:46 INFO mapreduce.Job:  map 0% reduce 0%
18/06/25 18:17:53 INFO mapreduce.Job:  map 100% reduce 0%
18/06/25 18:17:53 INFO mapreduce.Job: Job job_1529930528216_0005 completed successfully
18/06/25 18:17:53 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=127640
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=309
                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)=4108
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=4108
                Total vcore-milliseconds taken by all map tasks=4108
                Total megabyte-milliseconds taken by all map tasks=4206592
        Map-Reduce Framework
                Map input records=7
                Map output records=7
                Input split bytes=223
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=85
                CPU time spent (ms)=1770
                Physical memory (bytes) snapshot=167895040
                Virtual memory (bytes) snapshot=2084478976
                Total committed heap usage (bytes)=106430464
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=0
18/06/25 18:17:53 INFO mapreduce.ExportJobBase: Transferred 309 bytes in 20.5319 seconds (15.0497 bytes/sec)
18/06/25 18:17:53 INFO mapreduce.ExportJobBase: Exported 7 records.
```

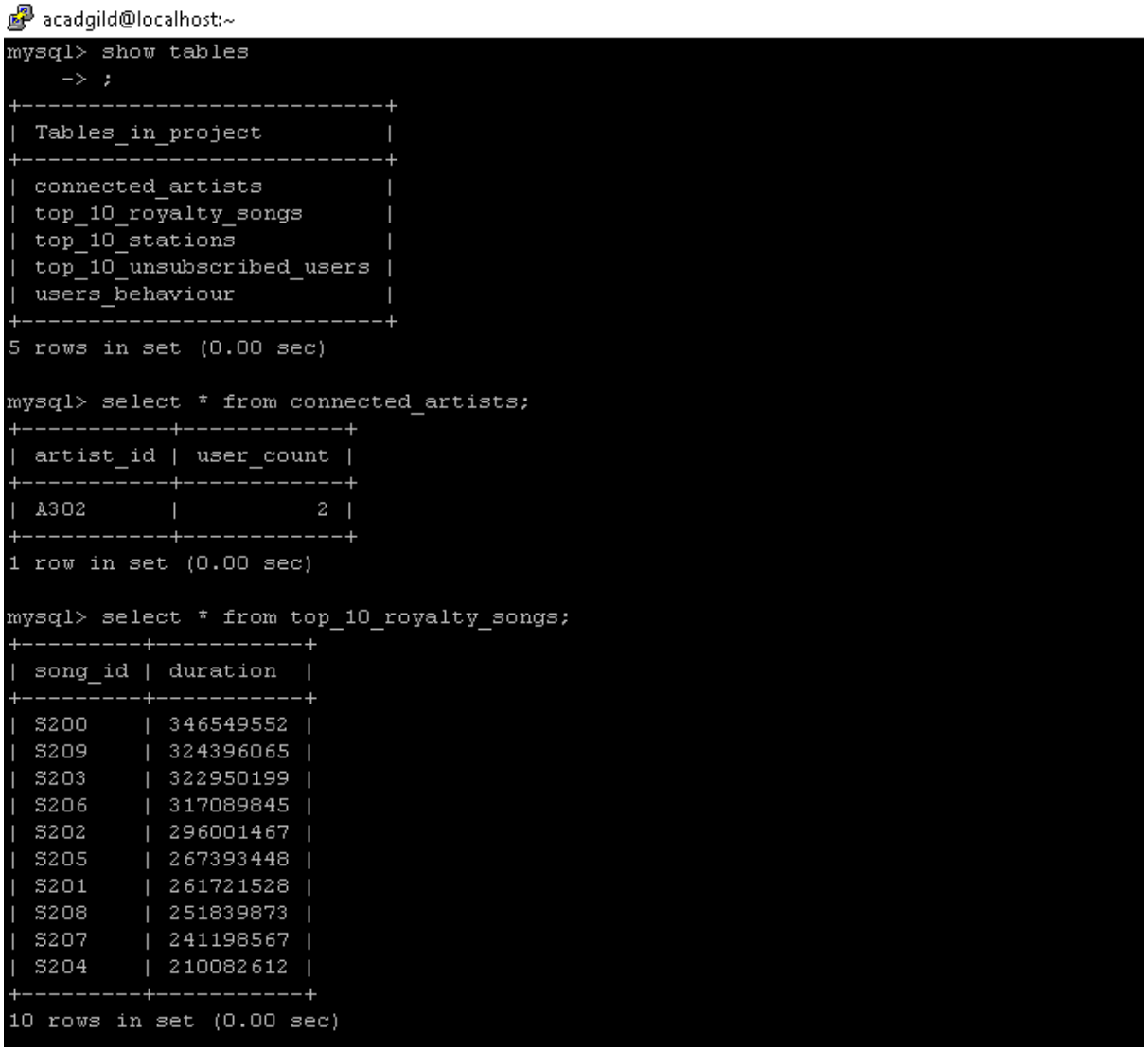
To verify whether data is exported successfully in MySQL database, let's open the MySql database in another terminal by using the below command and first check the required schema of the tables.

Mysql –u root –p [password**]**

use project;

Below screenshot shows the schema of all tables for analysed data created under the database project.

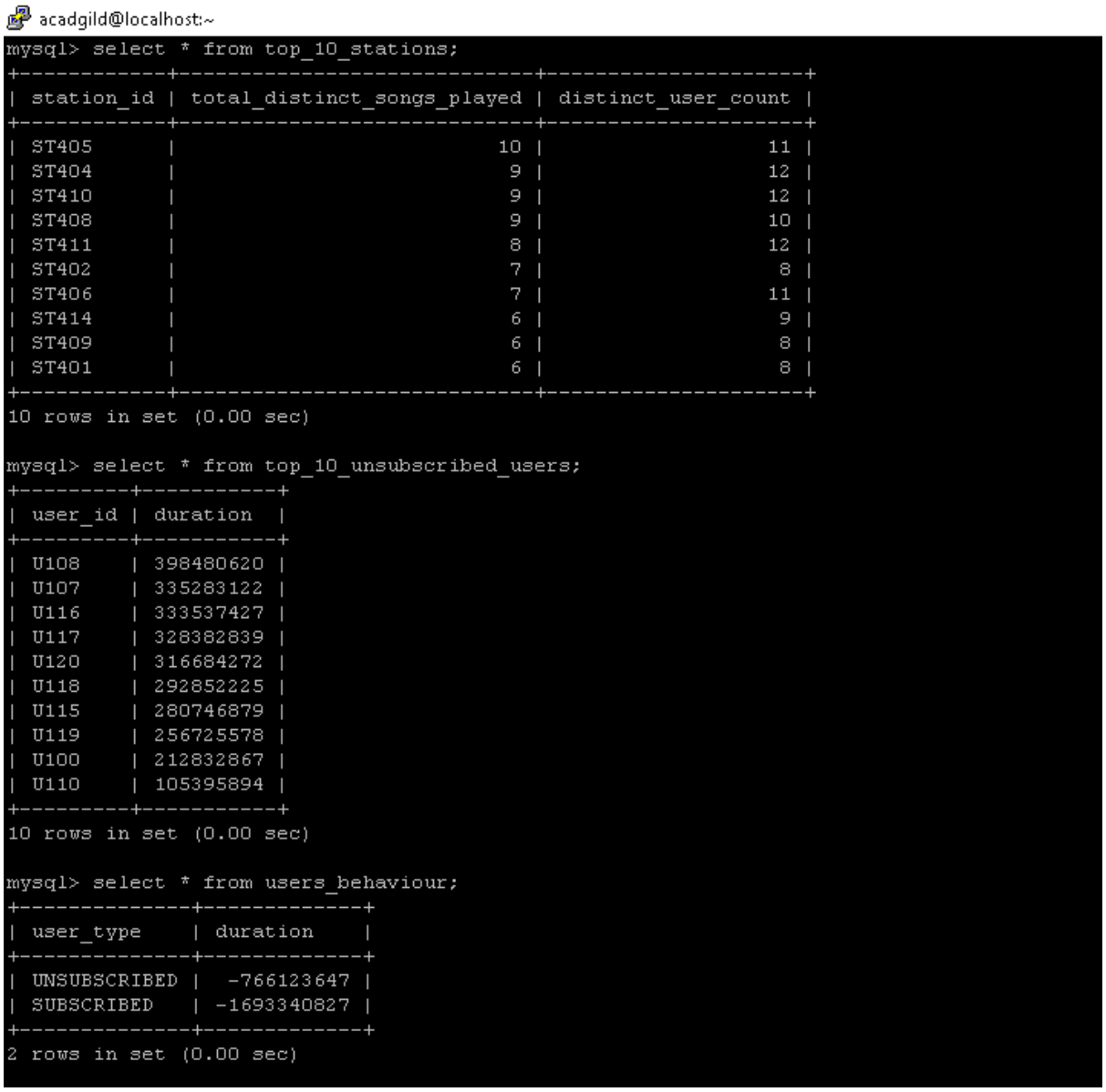




Above screen shot shows the table exported form hive table into MySQL databases.

It shows records of table connect_users and top_10_royality_songs.

Below screenshot shows the data imported top 10 stations, top 10 unsubscribed users and users-behaviour



Now after exporting data into MySQL, the following command in **data_export_toMySQL.sh**

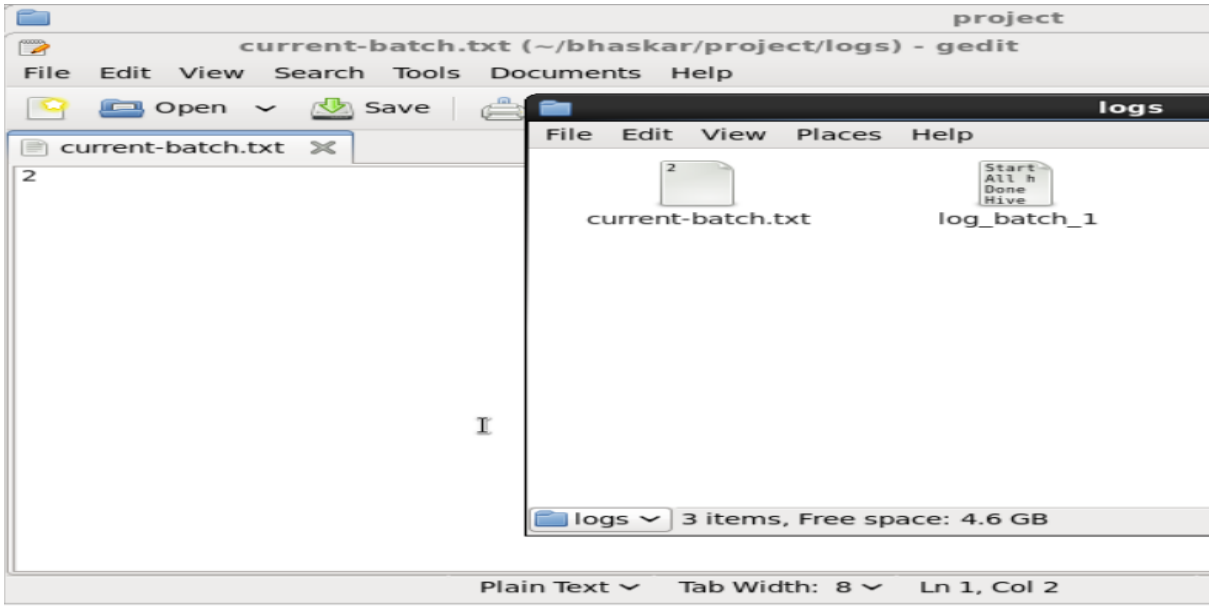
batchid = 'expr **batchid + 1'**

will show the batchid value '2' means one batch of data operations is successfully completed and new batch of data will be loaded for the analysis after every 3 hours.

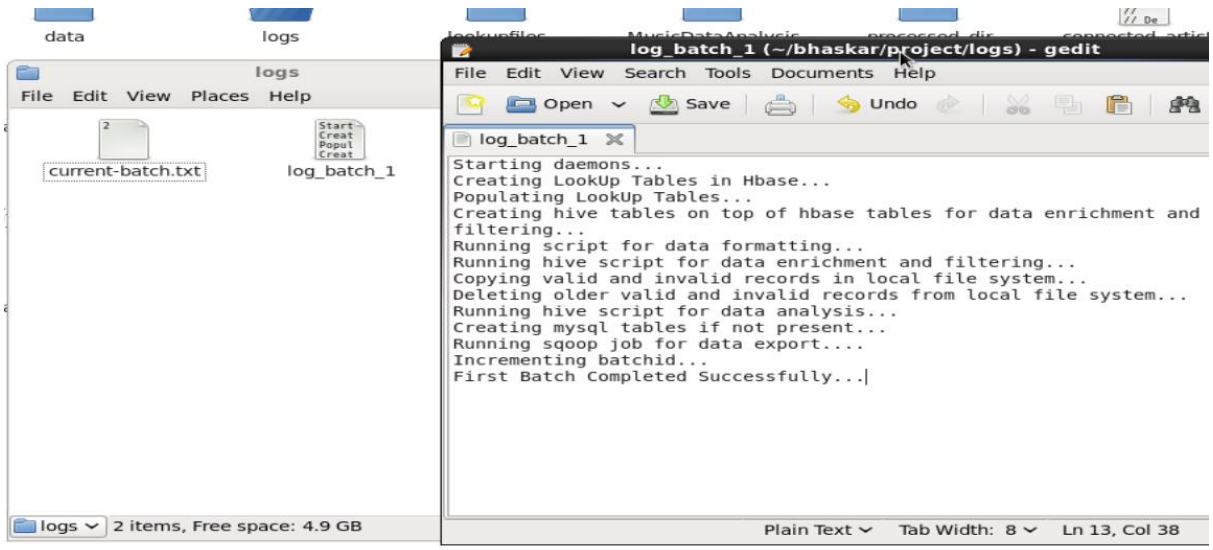
In addition, our complete **logs** maintained in the logs directory.

We can check logs to track the behaviour of the operations we have done on the data and overcome failures in the pipeline.

we can see the batchid incremented value 2 in current-batch.txt as shown below in the screenshot.



From below screenshot, we can check that all running process details from **log from log_batch_1 file**



8. Job Scheduling:

This is the last stage of the project in which we will wrapping up the all the above scripts in a single script file **music_data_processing.sh** and scheduling this script file to run periodically every three hours of time.

Below screenshot shows the **music_data_processing.sh** script file.

music_data_processing.sh

```
echo "Starting the daemons...."
sh start-daemons.sh

#Run jps command to confirm all the daemons

jps
echo "All hadoop daemons started....."

echo "Upload the look up tables now in Hbase..."
sh populate-lookup.sh

echo "Done with data population in hbase lookup tables..."

echo "Creating hive tables on top of hbase tables for data enrichment and filtering..."
sh data_enrichment_filtering_schema.sh

echo "Hive table with Hbase Mapping Complete..."

echo "Start the data formatting...."
sh dataformatting.sh

echo "data formatting complete..."

echo "Start data enrichment as per the requirement..."
sh data_enrichment.sh
```

```
echo "Start data enrichment as per the requirement..."
sh data_enrichment.sh

echo "Data Enrichment Complete..."

echo "Run Data Analysis script for some use cases..."
sh data_analysis.sh

echo "Run export command to transfer table to mysql..."
sh data_export_toMySQL.sh

echo "Table export part complete..."

echo "First Batch Complete Successfully!!"
```

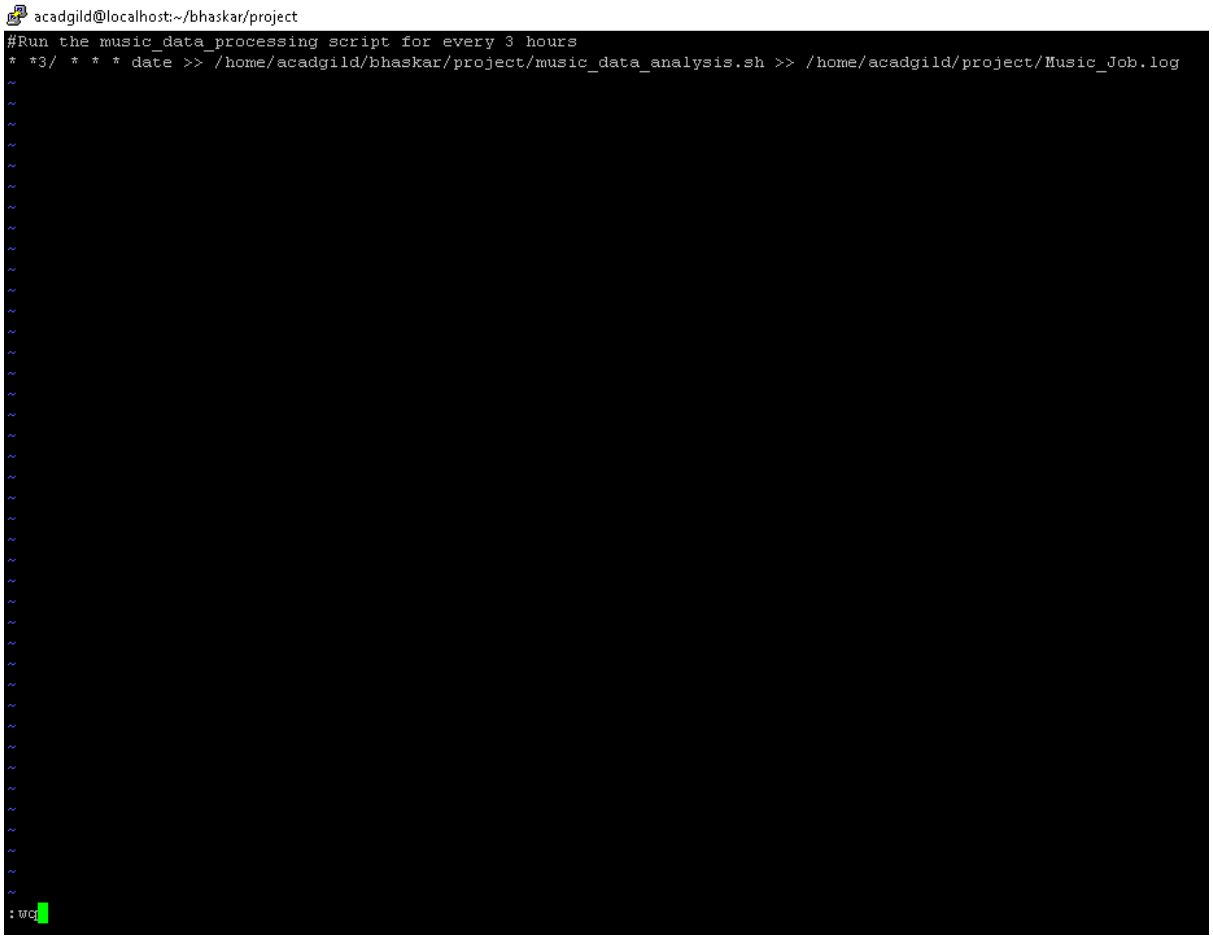
We schedule our script by creating a **crontab** to run it for every 3 hours.

To schedule job from crontab first we change the directory to our project directory and then we will launch crontab editor by using command **crontab –e** as shown below in the screenshot

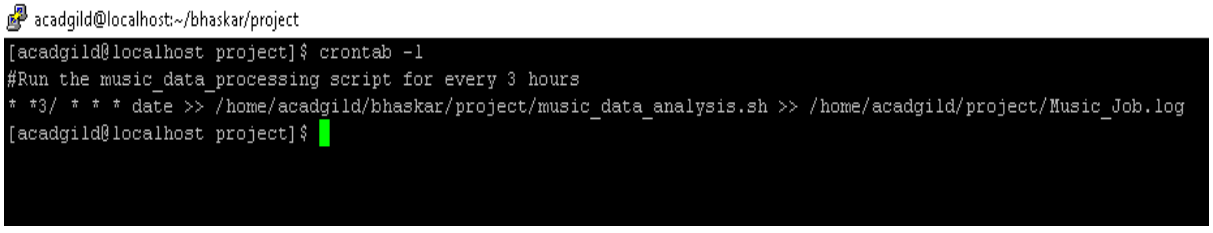
acadgild@localhost:~/bhaskar/project

```
[acadgild@localhost ~]$ cd bhaskar/project/
[acadgild@localhost project]$ crontab -e
* * * * * /home/acadgild/install/data/dfs/simple/update-acadgildvm.sh
~
~
~
~
~
~
~
~
~
~
~
```

Below screenshot shows the creation of job in the crontab for **music_data_analysis.sh** script



We can check our scheduled job by using the command **crontab –l** (to list our job) as shown below in the screen shot.



Issues or errors occur during creation and compilation of the scripts.

During creation of the script, I faced some issue to run the spark job from the script.

Below are the main error messages from the log generated during the running the script.

ERROR: metadata.Table: Unable to get field from serde: org.apache.hadoop.hive.hbase.HBaseSerDe java.lang.RuntimeException: MetaException(message:java.lang.ClassNotFoundException Class org.apache.hadoop.hive.hbase.HBaseSerDe not found)

Solution: To solve this issue, I already given the path to hive libraries where all the jars are present, but we need to provide the exact path with hive-hbase storage handler path with version.

ERROR: Caused by: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 0.0 failed 1 times, most recent failure: Lost task 0.0 in stage 0.0 (TID 0, localhost, executor driver): java.io.IOException:

Solution: To solve this error we can kill the other spark-submit process if completed or we have to use the below properties

set by SparkConf: conf.set("spark.driver.maxResultSize", "3g")

set by spark-defaults.conf: spark.driver.maxResultSize 3g

set when calling spark-submit: --conf spark.driver.maxResultSize=3g

ERROR: INFO mapred.JobClient: map 0% reduce 0%
INFO mapred.JobClient: Task Id : attempt_201304210944_0692_m_000001_0, Status : FAILED
java.lang.NumberFormatException: For input string: "U102"
at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)

Solutions: I tried to stop this script and restart the mysqld server, and run the script again.

SUBMITTED BY
BHASKAR