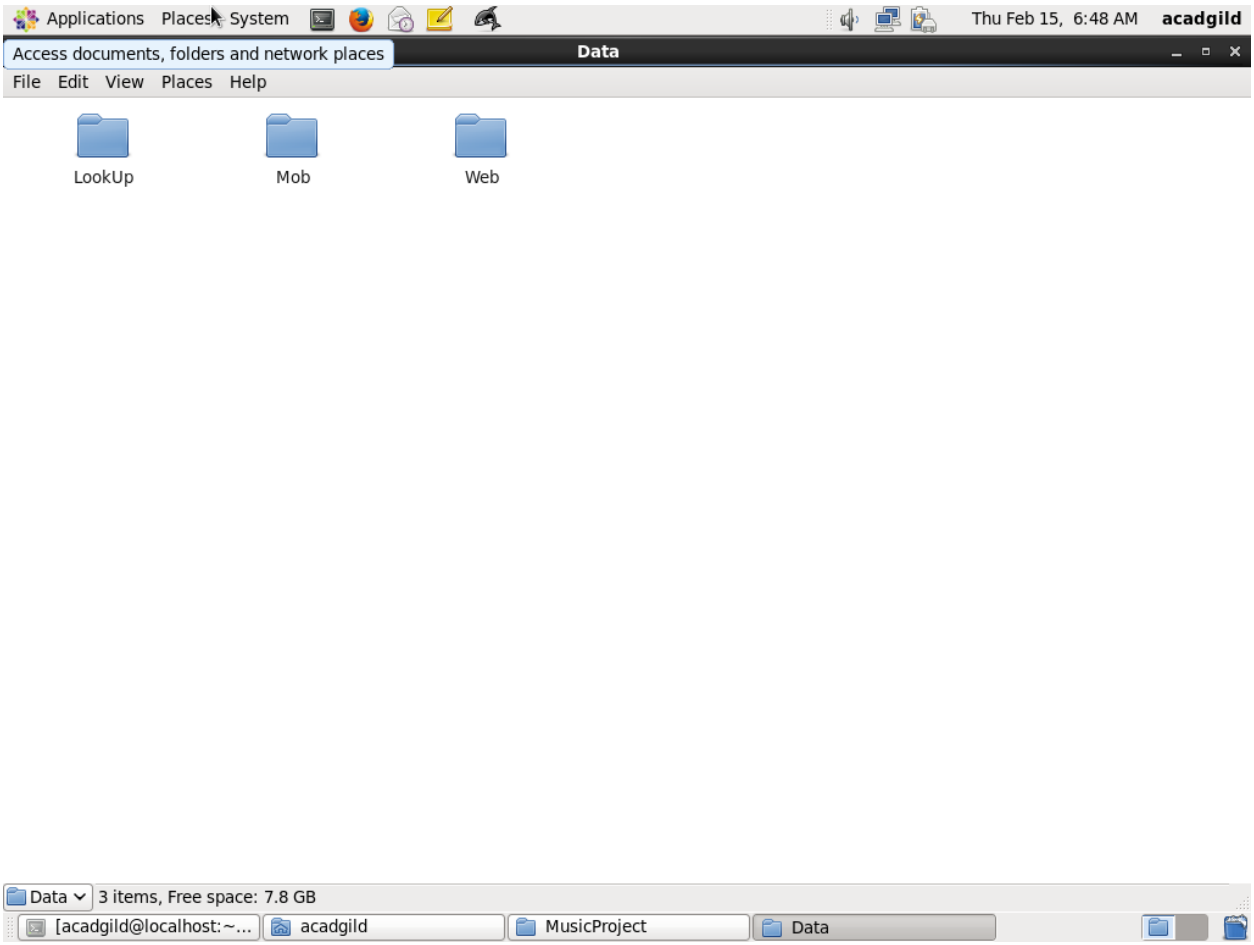
Music Data Analysis

Project:

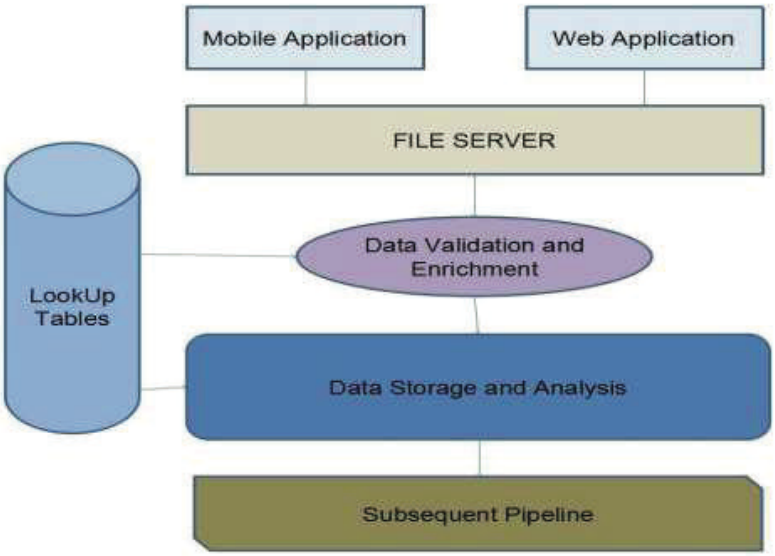
A leading music-catering company is planning to analyze large amount of data received from varieties of sources, namely mobile app and website to track the behavior of users, classify users, calculate royalties associated with the song and make appropriate business strategies. The file server receives data files periodically after every 3 hours.

DATASET:

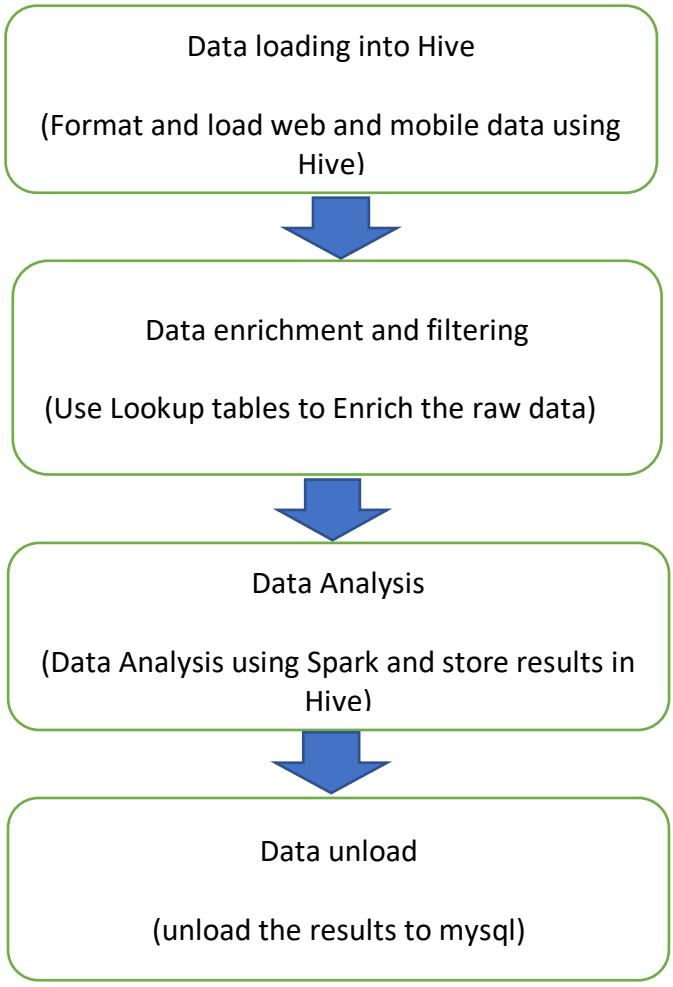
1. Data coming from web applications reside in /data/web and has xml format.
2. Data coming from mobile applications reside in /data/mob and has csv format.
3. Data present in lookup directory should be used in HBase.



Data Flow:



Design and Approach:



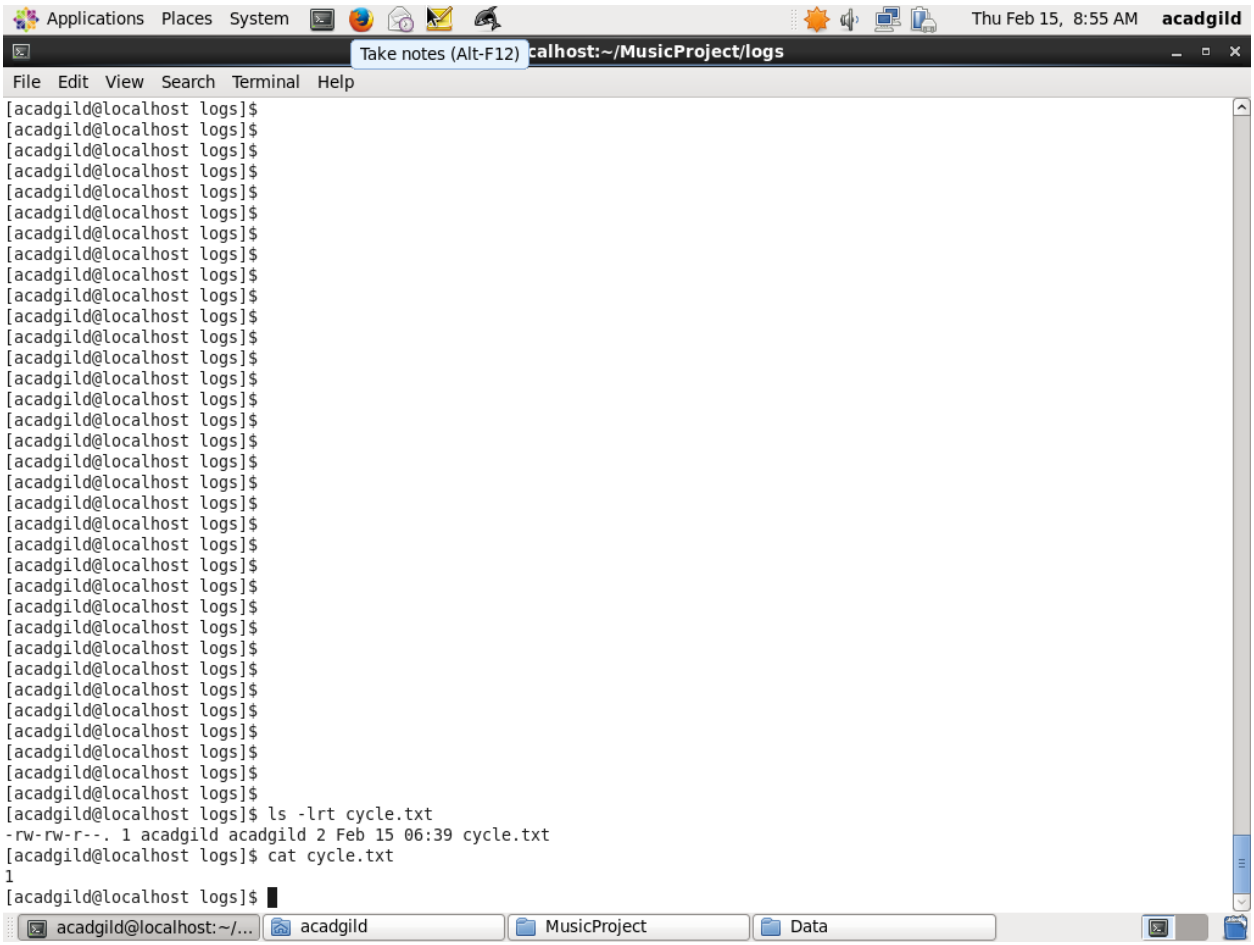
Softwares
OS: Centos

Hadoop 2.6
Hive 1.2.1
Sqoop 1.4.5
Mysql 5.1.73
Hbase 0.99.14-hadoop2
Spark 2.0.0

IDE used Eclipse, Notepad

Implementation:

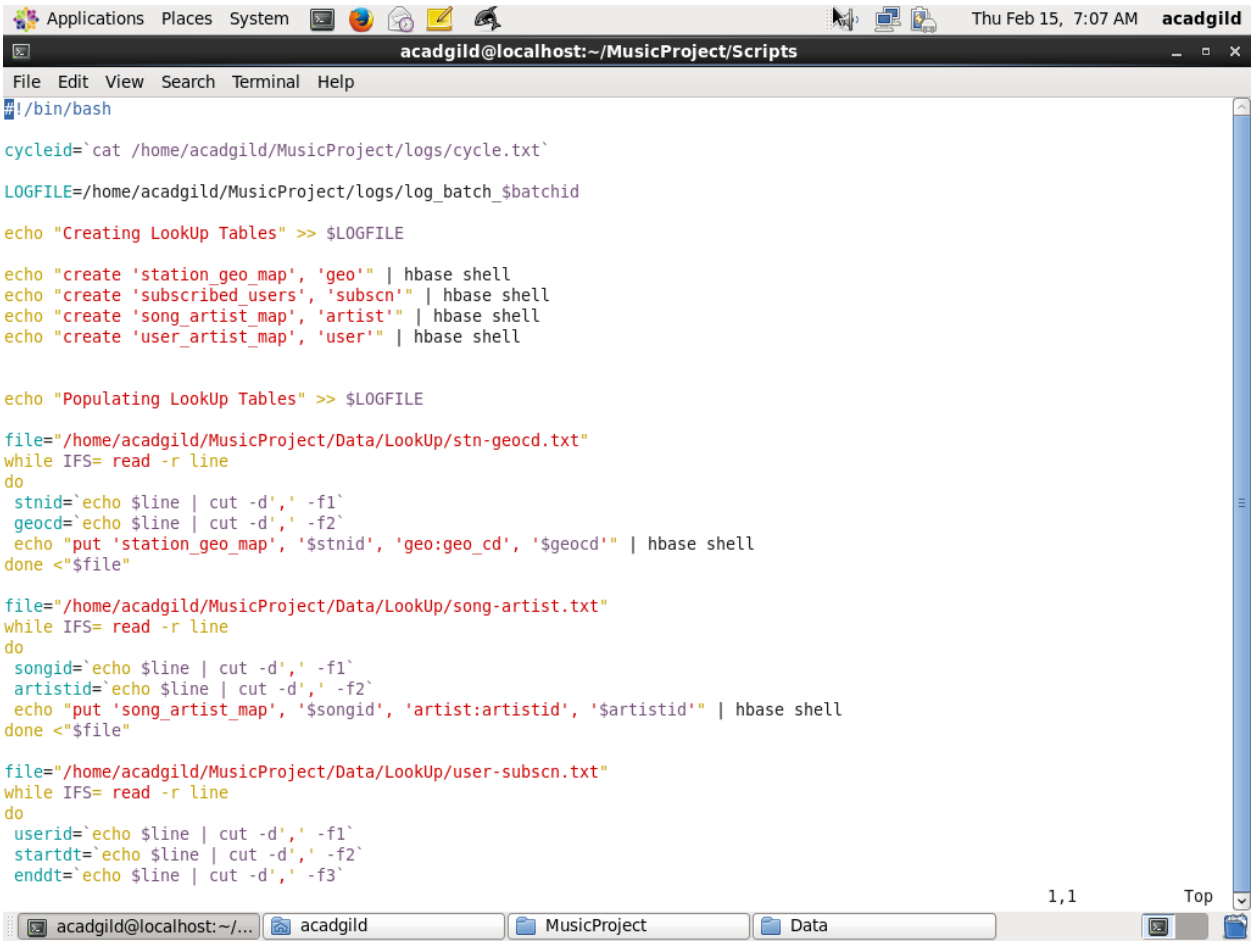
1. **Start all the hadoop demons, hbase and mysql. Create the cycle.txt to keep count of the cycle every three hours. Initial value is 1**

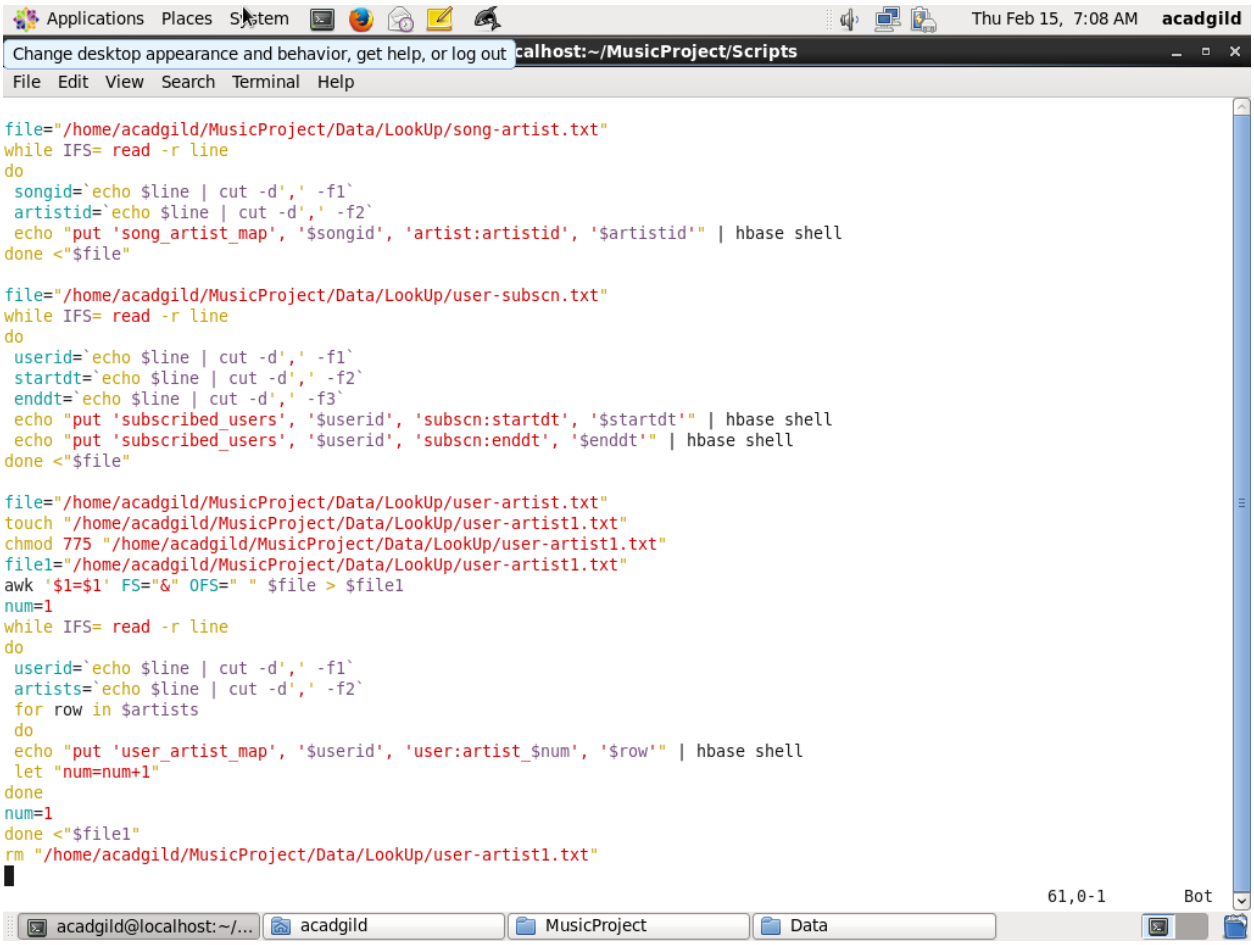


Populate the lookup table using the hbase-lookup.sh script. Below are the lookup tables to populate. This script needs to run at start to load hbase.

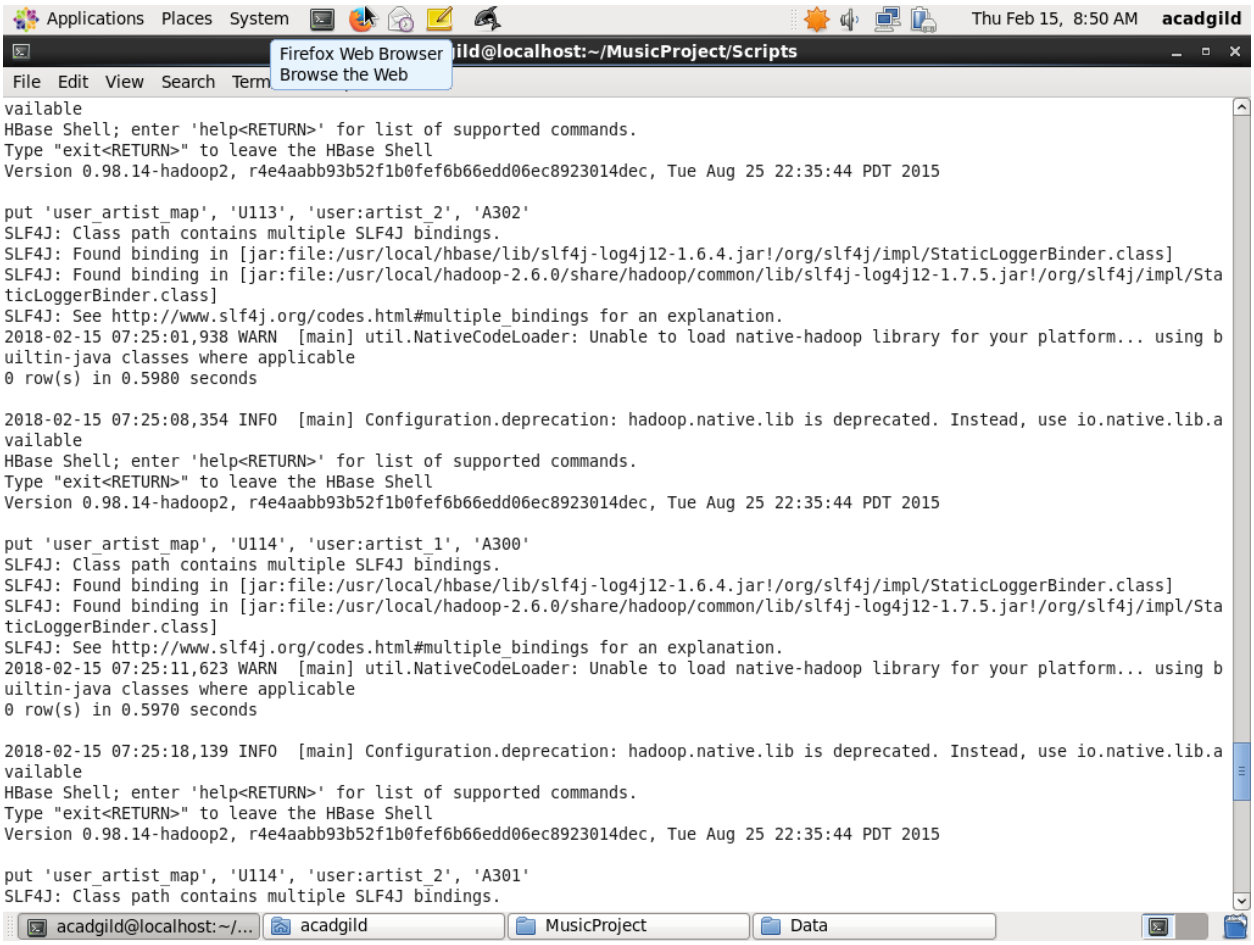
Table Name	**Description**
Station_Geo_Map	Contains mapping of a geo_cd with station_id
Subscribed_Users	Contains user_id, subscription_start_date and subscription_end_date. Contains details only for subscribed users
Song_Artist_Map	Contains mapping of song_id with artist_id alongwith royalty associated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

Script


Applications Places System
Thu Feb 15, 7:07 AM acadgild
acadgild@localhost:~/MusicProject/Scripts
File Edit View Search Terminal Help
#!/bin/bash
cycleid=`cat /home/acadgild/MusicProject/logs/cycle.txt`
LOGFILE=/home/acadgild/MusicProject/logs/log_batch_$batchid
echo "Creating LookUp Tables" >> $LOGFILE
echo "create 'station_geo_map', 'geo'" | hbase shell
echo "create 'subscribed_users', 'subscn'" | hbase shell
echo "create 'song_artist_map', 'artist'" | hbase shell
echo "create 'user_artist_map', 'user'" | hbase shell
echo "Populating LookUp Tables" >> $LOGFILE
file="/home/acadgild/MusicProject/Data/LookUp/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/MusicProject/Data/LookUp/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/MusicProject/Data/LookUp/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`
1,1 Top
acadgild@localhost:~/...
acadgild
MusicProject
Data



Logs after running the script



HBase tables and data

```
hbase(main):006:0* list
TABLE
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hbase/lib/slf4j-log4j12-1.6.4.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
2018-02-15 09:05:27,199 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
song_artist_map
station_geo_map
subscribed_users
user_artist_map
4 row(s) in 2.9140 seconds

=> ["song_artist_map", "station_geo_map", "subscribed_users", "user_artist_map"]
hbase(main):007:0>
```

```
=> ["song_artist_map", "station_geo_map", "subscribed_users", "user_artist_map"]
hbase(main):007:0> scan 'song_artist_map'
ROW                           COLUMN+CELL
 S200                         column=artist:artistid, timestamp=1518658921151, value=A300
 S201                         column=artist:artistid, timestamp=1518658933836, value=A301
 S202                         column=artist:artistid, timestamp=1518658943228, value=A302
 S203                         column=artist:artistid, timestamp=1518658952787, value=A303
 S204                         column=artist:artistid, timestamp=1518658962804, value=A304
 S205                         column=artist:artistid, timestamp=1518658973712, value=A301
 S206                         column=artist:artistid, timestamp=1518658984386, value=A302
 S207                         column=artist:artistid, timestamp=1518658995630, value=A303
 S208                         column=artist:artistid, timestamp=1518659006569, value=A304
 S209                         column=artist:artistid, timestamp=1518659017215, value=A305
10 row(s) in 0.4060 seconds

hbase(main):008:0> scan 'station_geo_map'
ROW                           COLUMN+CELL
 ST400                        column=geo:geo_cd, timestamp=1518658759193, value=A
 ST401                        column=geo:geo_cd, timestamp=1518658769757, value=AU
 ST402                        column=geo:geo_cd, timestamp=1518658781201, value=AP
 ST403                        column=geo:geo_cd, timestamp=1518658792119, value=J
 ST404                        column=geo:geo_cd, timestamp=1518658803345, value=E
 ST405                        column=geo:geo_cd, timestamp=1518658814863, value=A
 ST406                        column=geo:geo_cd, timestamp=1518658826254, value=AU
 ST407                        column=geo:geo_cd, timestamp=1518658836769, value=AP
 ST408                        column=geo:geo_cd, timestamp=1518658847318, value=E
 ST409                        column=geo:geo_cd, timestamp=1518658857438, value=E
 ST410                        column=geo:geo_cd, timestamp=1518658868713, value=A
 ST411                        column=geo:geo_cd, timestamp=1518658878990, value=A
 ST412                        column=geo:geo_cd, timestamp=1518658889106, value=AP
 ST413                        column=geo:geo_cd, timestamp=1518658900033, value=J
 ST414                        column=geo:geo_cd, timestamp=1518658910958, value=E
15 row(s) in 0.2760 seconds

hbase(main):009:0> scan 'subscribed_users'
ROW                           COLUMN+CELL
 U100                         column=subscn:enddt, timestamp=1518659041346, value=1465130523
 U100                         column=subscn:startdt, timestamp=1518659029729, value=1465230523
 U101                         column=subscn:enddt, timestamp=1518659064587, value=1475130523
 U101                         column=subscn:startdt, timestamp=1518659053883, value=1465230523
 U102                         column=subscn:enddt, timestamp=1518659087083, value=1475130523
 U102                         column=subscn:startdt, timestamp=1518659075097, value=1465230523
 U103                         column=subscn:enddt, timestamp=1518659110492, value=1475130523
 U103                         column=subscn:startdt, timestamp=1518659099086, value=1465230523
 U104                         column=subscn:enddt, timestamp=1518659132935, value=1475130523
 U104                         column=subscn:startdt, timestamp=1518659121206, value=1465230523
 U105                         column=subscn:enddt, timestamp=1518659156670, value=1475130523
 U105                         column=subscn:startdt, timestamp=1518659144955, value=1465230523
 U106                         column=subscn:enddt, timestamp=1518659179642, value=1485130523
 U106                         column=subscn:startdt, timestamp=1518659167810, value=1465230523
 U107                         column=subscn:enddt, timestamp=1518659201182, value=1455130523
 U107                         column=subscn:startdt, timestamp=1518659189787, value=1465230523
 U108                         column=subscn:enddt, timestamp=1518659223667, value=1465230623
 U108                         column=subscn:startdt, timestamp=1518659212469, value=1465230523
 U109                         column=subscn:enddt, timestamp=1518659245721, value=1475130523
 U109                         column=subscn:startdt, timestamp=1518659234176, value=1465230523
 U110                         column=subscn:enddt, timestamp=1518659269357, value=1475130523
 U110                         column=subscn:startdt, timestamp=1518659257277, value=1465230523
 U111                         column=subscn:enddt, timestamp=1518659292304, value=1475130523
 U111                         column=subscn:startdt, timestamp=1518659280383, value=1465230523
 U112                         column=subscn:enddt, timestamp=1518659315439, value=1475130523
 U112                         column=subscn:startdt, timestamp=1518659304342, value=1465230523
 U113                         column=subscn:enddt, timestamp=1518659338181, value=1485130523
 U113                         column=subscn:startdt, timestamp=1518659327089, value=1465230523
 U114                         column=subscn:enddt, timestamp=1518659360869, value=1468130523
 U114                         column=subscn:startdt, timestamp=1518659349485, value=1465230523
15 row(s) in 0.3470 seconds
```

```
hbase(main):010:0> scan 'user_artist_map'
ROW                             COLUMN+CELL
 U100                           column=user:artist_1, timestamp=1518659371867, value=A300
 U100                           column=user:artist_2, timestamp=1518659384234, value=A301
 U100                           column=user:artist_3, timestamp=1518659395871, value=A302
 U101                           column=user:artist_1, timestamp=1518659408318, value=A301
 U101                           column=user:artist_2, timestamp=1518659419986, value=A302
 U102                           column=user:artist_1, timestamp=1518659431481, value=A302
 U103                           column=user:artist_1, timestamp=1518659441733, value=A303
 U103                           column=user:artist_2, timestamp=1518659452740, value=A301
 U103                           column=user:artist_3, timestamp=1518659464285, value=A302
 U104                           column=user:artist_1, timestamp=1518659474510, value=A304
 U104                           column=user:artist_2, timestamp=1518659486241, value=A301
 U105                           column=user:artist_1, timestamp=1518659499689, value=A305
 U105                           column=user:artist_2, timestamp=1518659512151, value=A301
 U105                           column=user:artist_3, timestamp=1518659523110, value=A302
 U106                           column=user:artist_1, timestamp=1518659533861, value=A301
 U106                           column=user:artist_2, timestamp=1518659543645, value=A302
 U107                           column=user:artist_1, timestamp=1518659553629, value=A302
 U108                           column=user:artist_1, timestamp=1518659563299, value=A300
 U108                           column=user:artist_2, timestamp=1518659575921, value=A303
 U108                           column=user:artist_3, timestamp=1518659587276, value=A304
 U109                           column=user:artist_1, timestamp=1518659597880, value=A301
 U109                           column=user:artist_2, timestamp=1518659610694, value=A303
 U110                           column=user:artist_1, timestamp=1518659622659, value=A302
 U110                           column=user:artist_2, timestamp=1518659633953, value=A301
 U111                           column=user:artist_1, timestamp=1518659646106, value=A303
 U111                           column=user:artist_2, timestamp=1518659658013, value=A301
 U112                           column=user:artist_1, timestamp=1518659672217, value=A304
 U112                           column=user:artist_2, timestamp=1518659682573, value=A301
 U113                           column=user:artist_1, timestamp=1518659693661, value=A305
 U113                           column=user:artist_2, timestamp=1518659703719, value=A302
 U114                           column=user:artist_1, timestamp=1518659713331, value=A300
 U114                           column=user:artist_2, timestamp=1518659723001, value=A301
 U114                           column=user:artist_3, timestamp=1518659732523, value=A302
15 row(s) in 0.2090 seconds
```

Now, we to lookup this tables in Hive using HBase storage handlers

2. Create Hive table on top of Hbase

Using data_enrichment_filtering_hive_schema.sh, create the external hive tables on top of Hbase

File Edit View Search Terminal Help

```
#!/bin/bash

cycleid=`cat /home/acadgild/MusicProject/logs/cycle.txt`
LOGFILE=/home/acadgild/MusicProject/logs/log_$cycleid

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

hive -f /home/acadgild/MusicProject/Scripts/create_hive_hbase_lookup.hql
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"data_enrichment_filtering_hive_schema.sh" 8L, 299C                                    1,1           All
```

File Edit View Search Terminal Help

```
station_id STRING,
geo_cd STRING
)
STORED By 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
with serdeproperties
("hbase.columns.mapping"=":key,geo:geo_cd")
tblproperties("hbase.table.name"="station_geo_map");

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

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

create external table if not exists user_artist_map
(
user_id STRING,
artist_id array MAP<STRING,STRING>
)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
with serdeproperties
("hbase.columns.mapping"=":key,user:")
tblproperties("hbase.table.name"="user_artist_map");
"create_hive_hbase_lookup.hql" [dos] 44L, 1248C                                         44,1          Bot
```

Running the script

```
[acadgild@localhost Scripts]$ ls
create_hive_hbase_lookup.hql  data_enrichment_filtering_hive_schema.sh  hbase-lookup.sh
[acadgild@localhost Scripts]$ vi data_enrichment_filtering_hive_schema.sh
[acadgild@localhost Scripts]$ vi create_hive_hbase_lookup.hql
[acadgild@localhost Scripts]$ vi data_enrichment_filtering_hive_schema.sh
[acadgild@localhost Scripts]$ vi create_hive_hbase_lookup.hql
[acadgild@localhost Scripts]$ sh data_enrichment_filtering_hive_schema.sh

Logging initialized using configuration in jar:file:/usr/local/hive/lib/hive-common-0.14.0.jar!/hive-log4j.properties
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/hive-jdbc-0.14.0-standalone.jar!/org/slf4j/impl/StaticLoggerBinder.clas
s]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Sta
ticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
OK
Time taken: 1.235 seconds
OK
Time taken: 0.042 seconds
OK
Time taken: 2.148 seconds
OK
Time taken: 0.271 seconds
OK
Time taken: 0.313 seconds
```

Hive tables

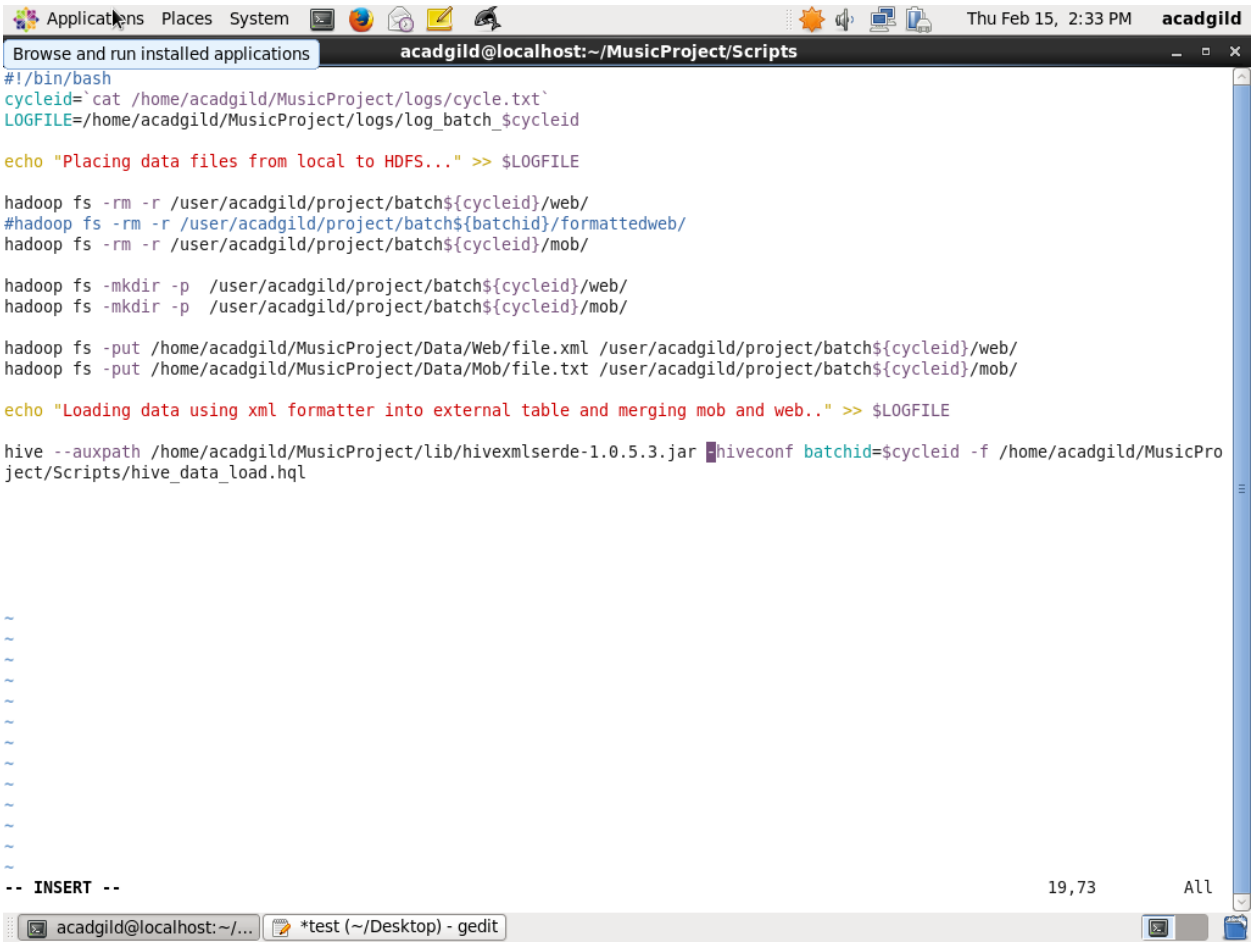
```
[acadgild@localhost Scripts]$ hive

Logging initialized using configuration in jar:file:/usr/local/hive/lib/hive-common-0.14.0.jar!/hive-log4j.properties
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/hive-jdbc-0.14.0-standalone.jar!/org/slf4j/impl/StaticLoggerBinder.clas
s]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Sta
ticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
hive> show databases;
OK
b1
custom
default
musicproject
Time taken: 0.429 seconds, Fetched: 4 row(s)
hive> use musicproject;
OK
Time taken: 0.047 seconds
hive> show tables;
OK
song_artist_map
station_geo_map
subscribed_users
user_artist_map
Time taken: 0.056 seconds, Fetched: 4 row(s)
hive>
```

3. Data Loading

In this step we are loading web xml in to hive using xml serde and finally merge the web and mobile data

First we move the data to HDFS for both mob and web data into batch partition and finally use hive script to load both web and mob to hive table called merged_data.

Script dataloading.sh, **HiveScript** hive_data_load.hql

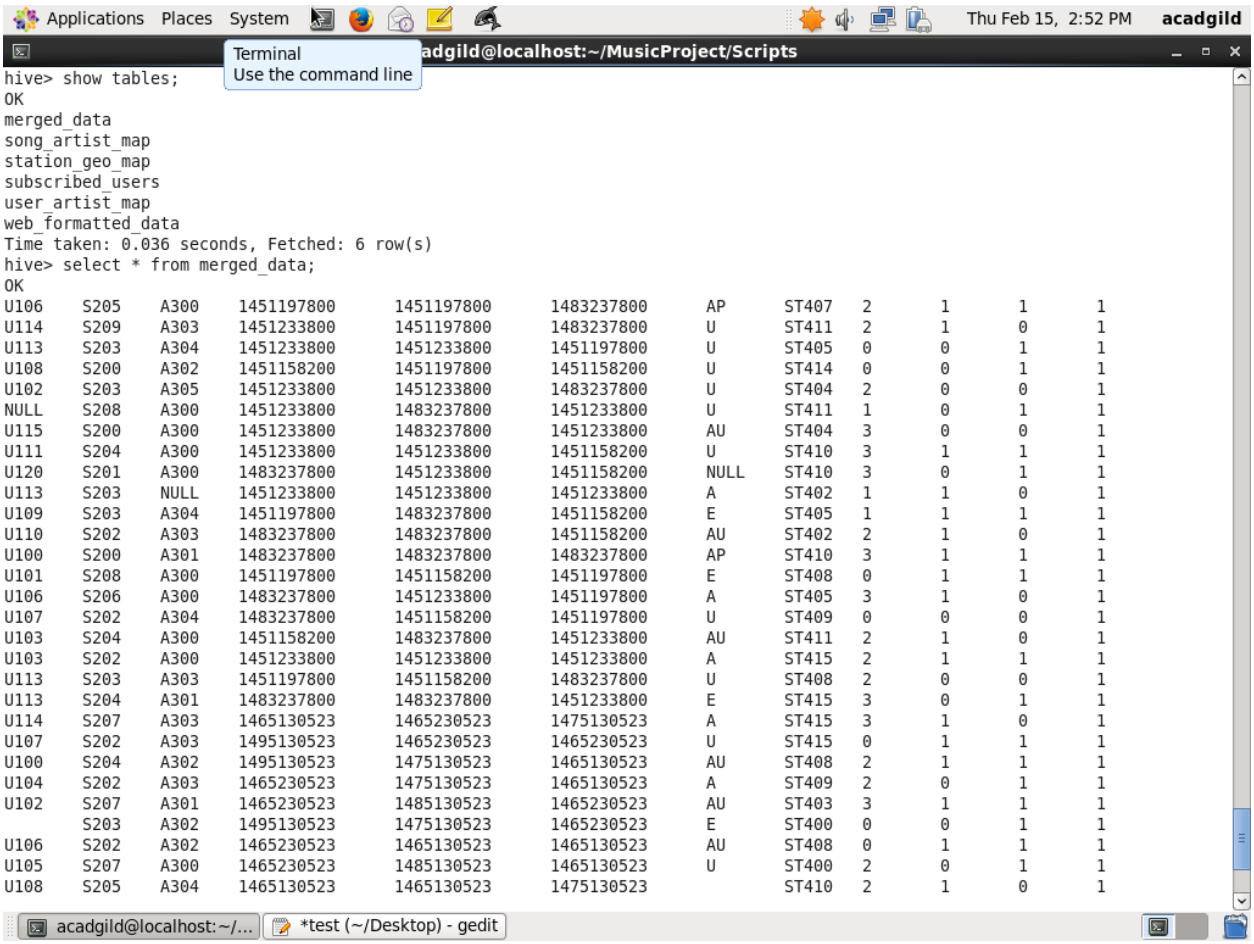


```
set hive.support.sql11.reserved.keywords=false;

USE musicproject;

DROP TABLE IF EXISTS WEB_FORMATTED_DATA;
CREATE EXTERNAL TABLE WEB_FORMATTED_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)
 ROW FORMAT SERDE 'com.ibm.spss.hive.serde2.xml.XmlSerDe'
 WITH SERDEPROPERTIES (
 "column.xpath.user_id"="/record/user_id/text()",
 "column.xpath.song_id"="/record/song_id/text()",
  "column.xpath.artist_id"="/record/artist_id/text()",
 "column.xpath.timestamp"="/record/timestamp/text()",
  "column.xpath.start_ts"="/record/start_ts/text()",
 "column.xpath.end_ts"="/record/end_ts/text()",
  "column.xpath.geo_cd"="/record/geo_cd/text()",
 "column.xpath.station_id"="/record/station_id/text()",
  "column.xpath.song_end_type"="/record/song_end_type/text()",
 "column.xpath.like"="/record/like/text()",
  "column.xpath.dislike"="/record/dislike/text()")
 STORED AS INPUTFORMAT 'com.ibm.spss.hive.serde2.xml.XmlInputFormat'
 OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
 LOCATION '/user/acadgild/project/batch${hiveconf:batchid}/web/'
 TBLPROPERTIES ("xmlinput.start"="<record>","xmlinput.end"= "</record>");
```

```
 CREATE TABLE IF NOT EXISTS MERGED_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)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

INSERT OVERWRITE TABLE MERGED_DATA
PARTITION(batchid=${hiveconf:batchid})
SELECT
user_id ,
song_id ,
artist_id ,
unix_timestamp(timestamp , 'yyyy-MM-dd HH:mm:ss') ,
unix_timestamp(start_ts , 'yyyy-MM-dd HH:mm:ss') ,
unix_timestamp(end_ts , 'yyyy-MM-dd HH:mm:ss'),
geo_cd ,
station_id ,
song_end_type ,
like ,
dislike
from WEB_FORMATTED_DATA;


LOAD DATA INPATH '/user/acadgild/project/batch${hiveconf:batchid}/mob/'
INTO TABLE merged_data PARTITION (batchid=${hiveconf:batchid});
                                                                                73,0-1        Bot
```

After loading the running the script data is loaded in merged_data

```
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Sta
ticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
OK
Time taken: 1.091 seconds
OK
Time taken: 0.752 seconds
OK
Time taken: 0.465 seconds
OK
Time taken: 0.103 seconds
Query ID = acadgild_20180215143434_bf5425bd-4872-4bda-9fe5-26efef0936a8
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1518654102046_0003, Tracking URL = http://localhost:8088/proxy/application_1518654102046_0003/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1518654102046_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2018-02-15 14:34:41,371 Stage-1 map = 0%,  reduce = 0%
2018-02-15 14:34:56,421 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.8 sec
MapReduce Total cumulative CPU time: 2 seconds 800 msec
Ended Job = job_1518654102046_0003
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://localhost:9000/tmp/hive/acadgild/1de780ee-054e-4267-a304-c222102fe87d/hive_2018-02-15_14-34-24_580_802
8894388405474641-1/-ext-10000
Loading data to table musicproject.merged_data partition (batchid=1)
Partition musicproject.merged_data{batchid=1} stats: [numFiles=1, numRows=0, totalSize=0, rawDataSize=0]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 2.8 sec   HDFS Read: 307 HDFS Write: 60 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 800 msec
OK
Time taken: 37.176 seconds
Loading data to table musicproject.merged_data partition (batchid=1)
Partition musicproject.merged_data{batchid=1} stats: [numFiles=2, numRows=0, totalSize=1239, rawDataSize=0]
OK
Time taken: 1.031 seconds
[acadgild@localhost Scripts]$
```



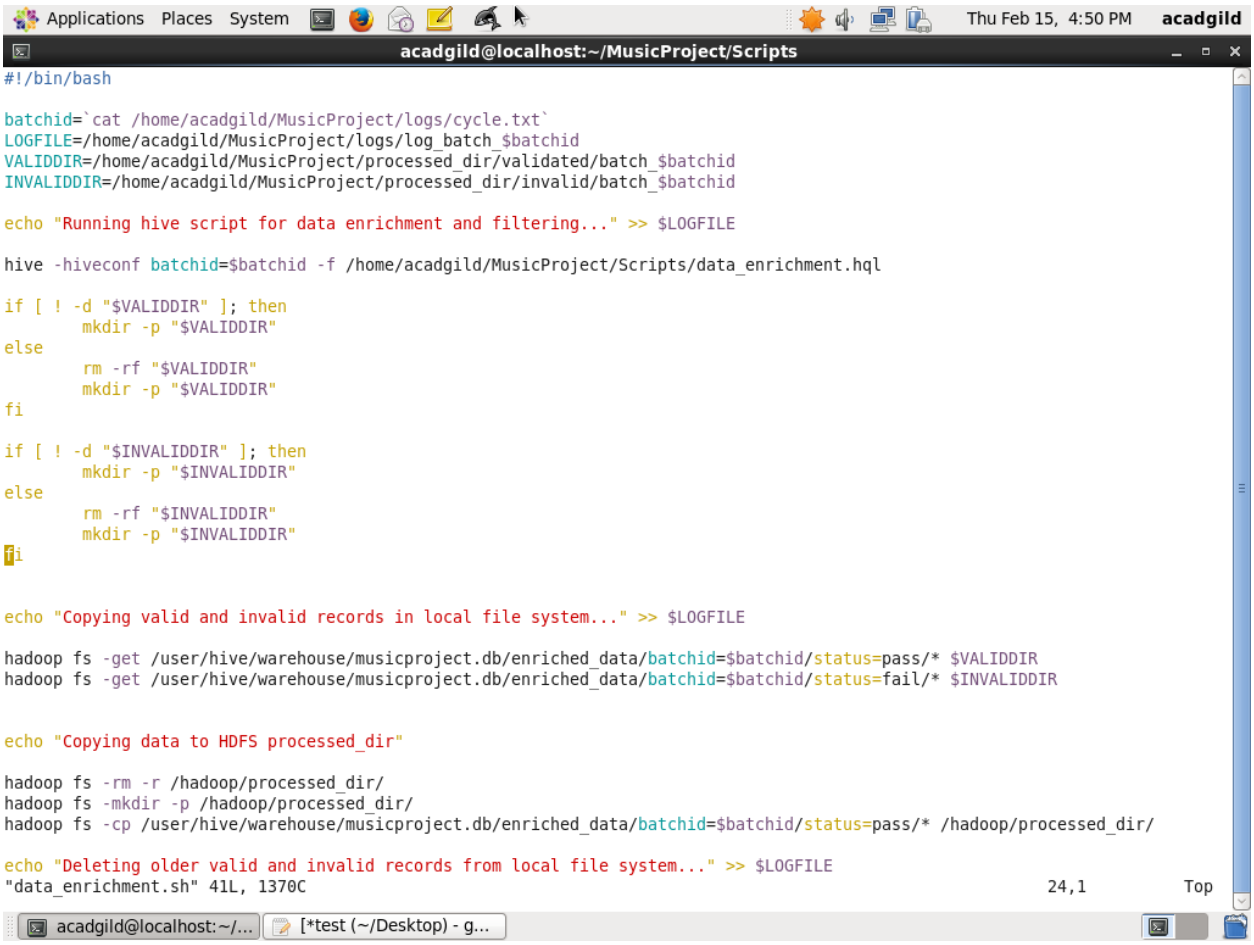
In the above screenshot, data from web is formatted and loaded into merged_data and data from mob is loaded into merged data and portioned based on batch id.

4. Data Enrichment

Rules for data enrichment
1. If any of like or dislike is NULL or absent, consider it as 0.
2. If fields like Geo_cd and Artist_id are NULL or absent, consult the lookup tables for fields Station_id and Song_id respectively to get the values of Geo_cd and Artist_id.
3. If corresponding lookup entry is not found, consider that record to be invalid.
NULL or absent field Look up field Look up table (Table from
which record can be updated)

Using data_enrichment.sh script, the data is enriched and valid, invalid records are stored in localfile , for which only 7 days data is maintained

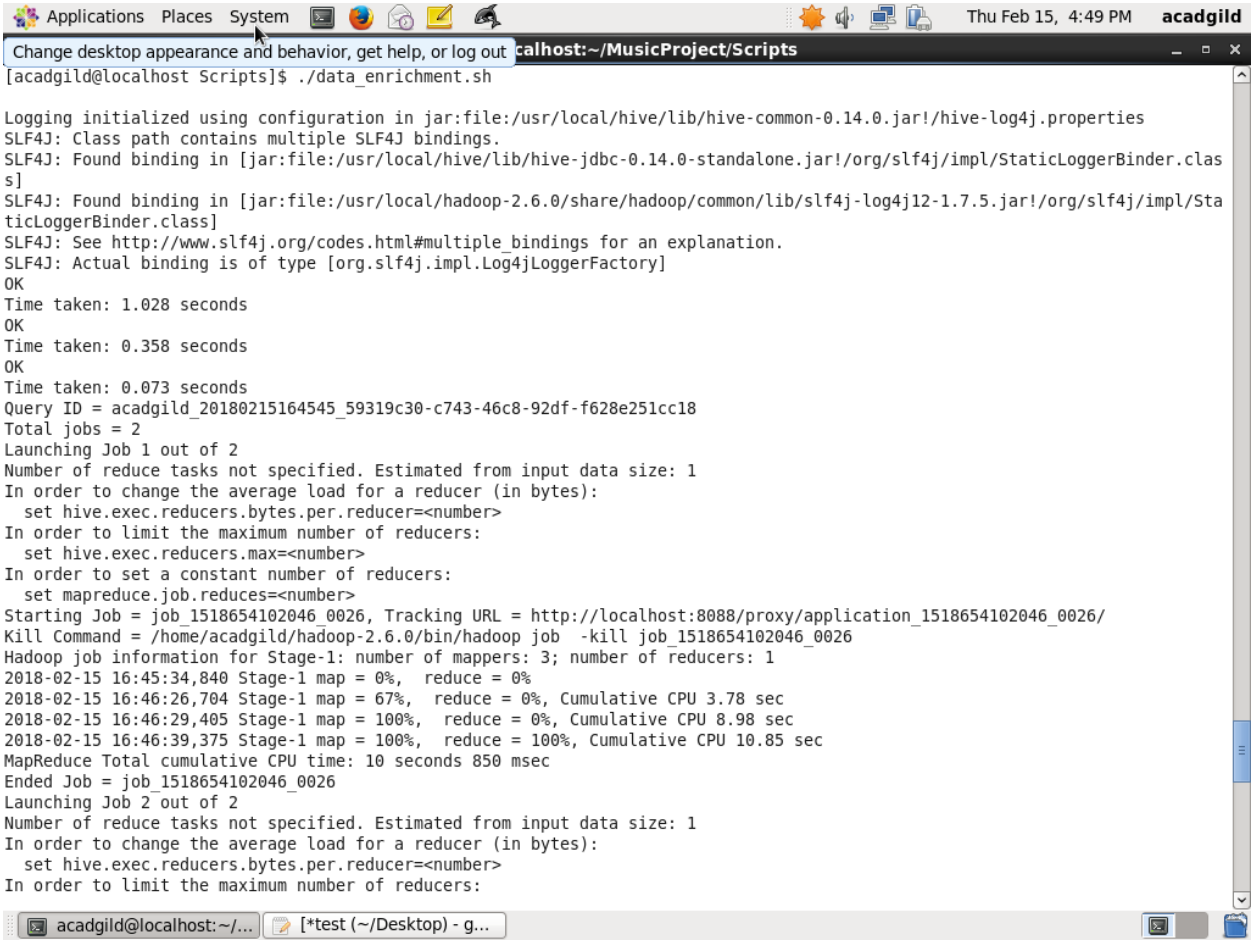
Below is the screenshot of the script



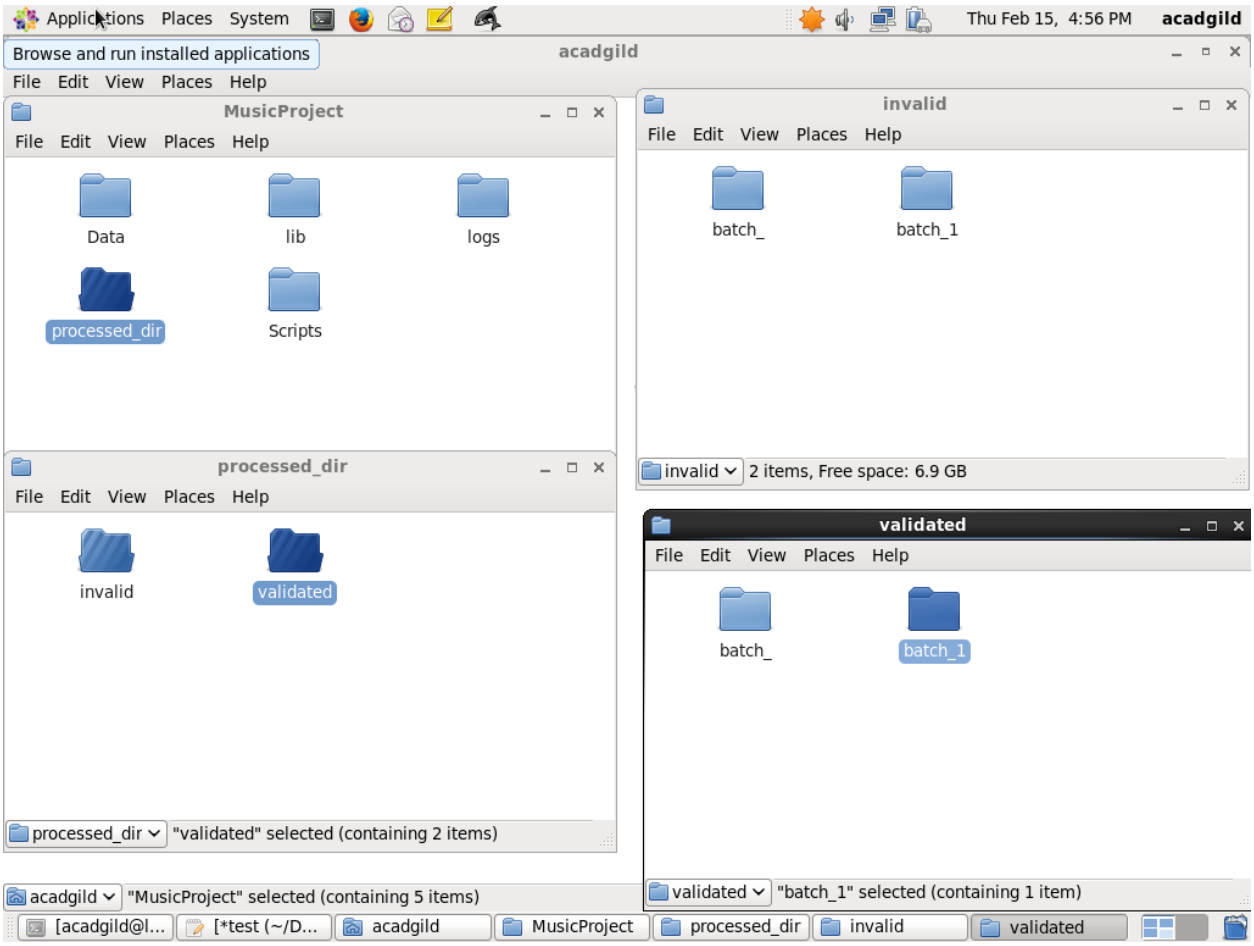
Hive queries to do the enrichment, and enriched data is stored in enriched_data table in hive

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

USE musicproject;

CREATE TABLE IF NOT EXISTS temp_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)
STORED as ORC;

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)
                                                                                1,1           Top


INSERT OVERWRITE TABLE temp_enriched_data
PARTITION (batchid)
SELECT
i.user_id,
i.song_id,
IF(i.artist_id is NULL OR i.artist_id='',sa.artist_id,i.artist_id) AS artist_id,
i.timestamp,
i.start_ts,
i.end_ts,
IF(i.geo_cd is NULL OR i.geo_cd='',sg.geo_cd,i.geo_cd) AS geo_cd,
i.station_id,
IF (i.song_end_type IS NULL,3,i.song_end_type) AS song_end_type,
IF (i.like IS NULL,0,i.like) AS like,
IF (i.dislike IS NULL,0,i.dislike) AS dislike,
i.batchid
FROM merged_data i
LEFT OUTER JOIN station_geo_map sg ON i.station_id = sg.station_id
LEFT OUTER JOIN song_artist_map sa ON i.song_id = sa.song_id
WHERE i.batchid=${hiveconf:batchid};

INSERT OVERWRITE TABLE enriched_data
PARTITION (batchid,status)
SELECT
i.user_id,
i.song_id,
i.artist_id,
i.timestamp,
i.start_ts,
i.end_ts,
i.geo_cd,
i.station_id,
i.song_end_type,
i.like,
i.dislike,
                                                                                57,1          64%
```

After running the scripts below are the screenshots



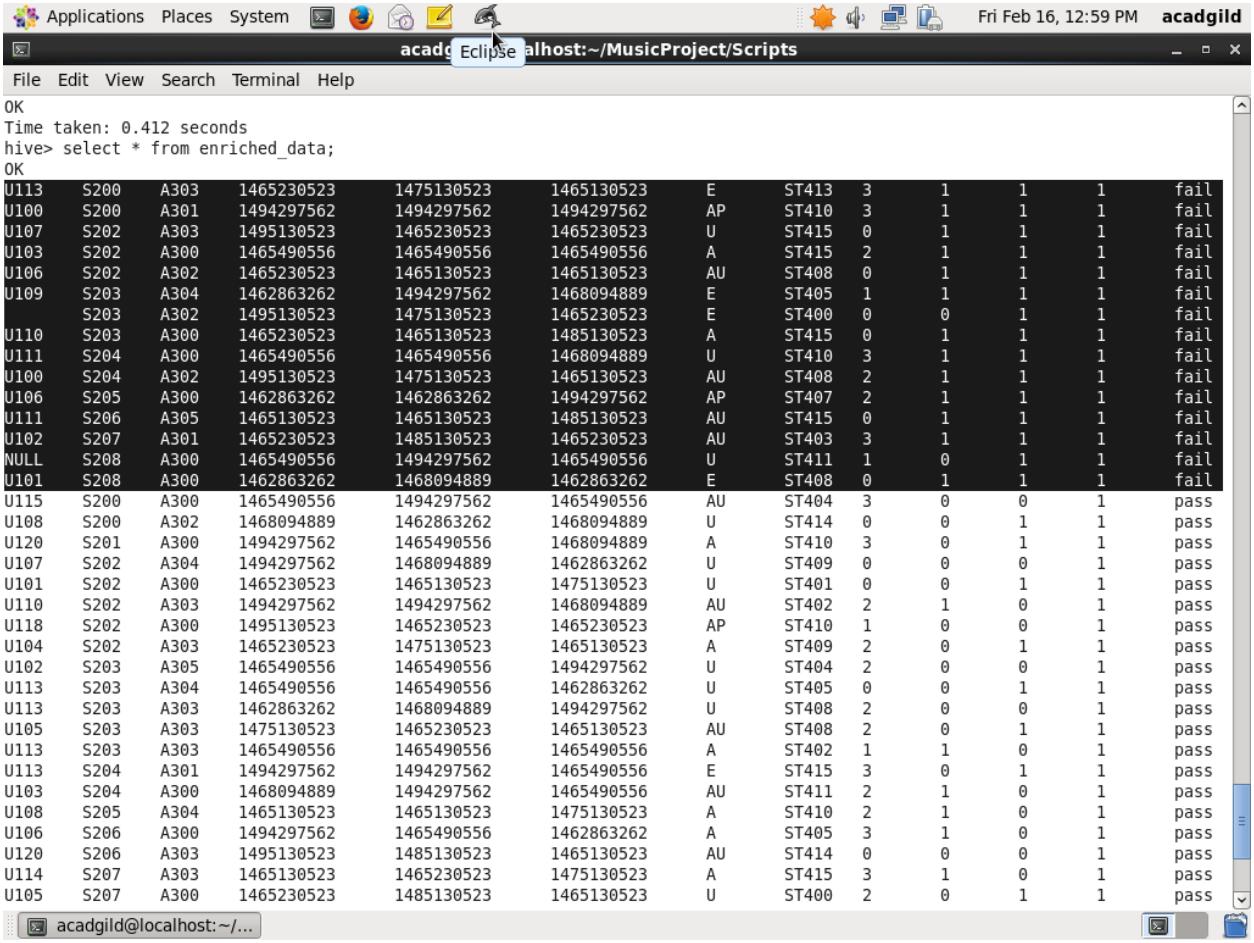
After scripts are validated and invalidated records are copied to local and only 7 days data is kept



Valid records are moved to hadoop/processed_dir/

```
[acadgild@localhost Scripts]$ hadoop fs -ls /hadoop/processed_dir/
18/02/15 16:58:30 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 1 items
-rw-r--r--   1 acadgild supergroup       1435 2018-02-15 16:48 /hadoop/processed_dir/000000_0
[acadgild@localhost Scripts]$
```

We also can check and validate the enriched data in hive. All failed records are because user_id is null or like and dislike equals 1.

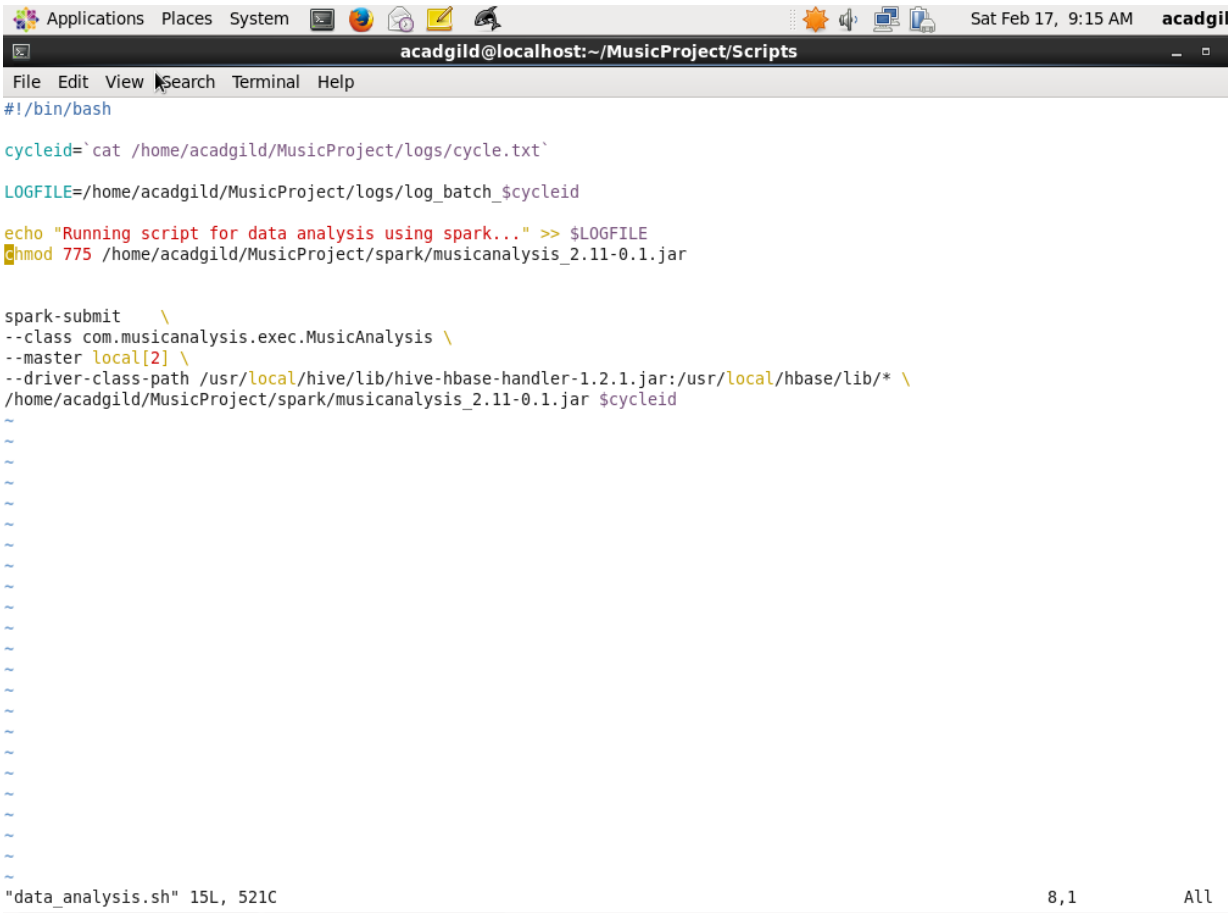


After data is enriched successfully, we start the data analysis using Spark

5. Data Analysis using Spark

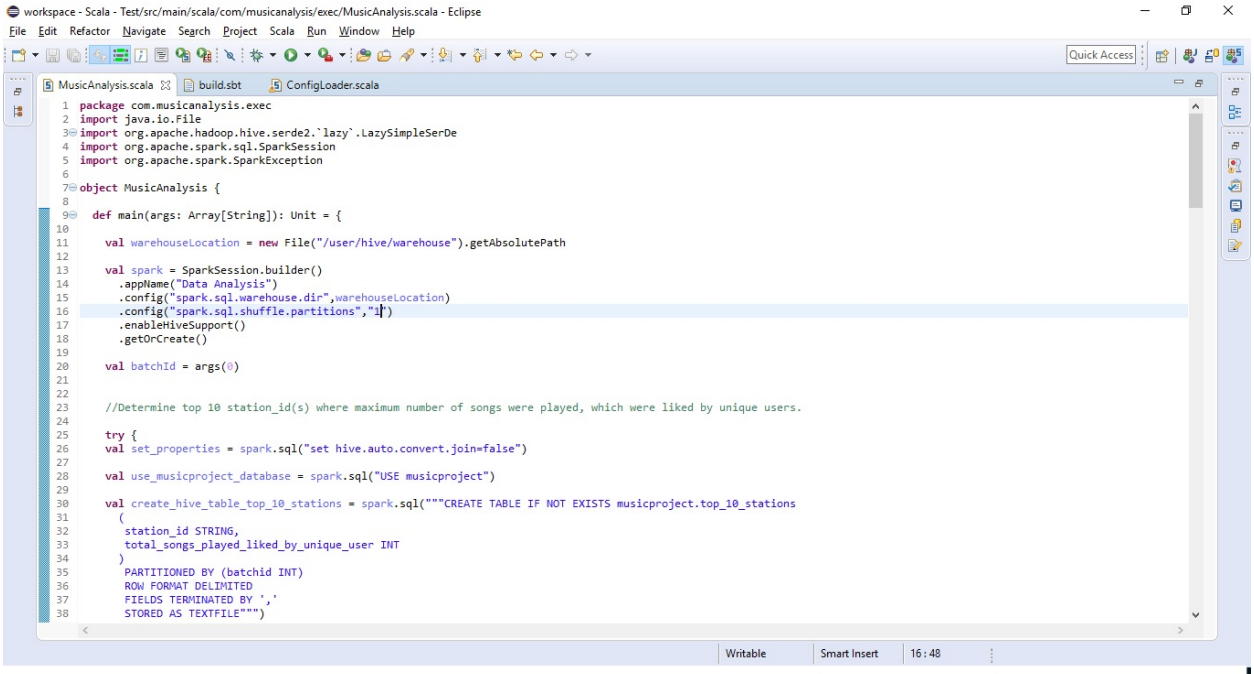
In the stage enriched data is analyzed using spark sql. Spark submit job is triggered using shell, so analysis can be done for every batch.

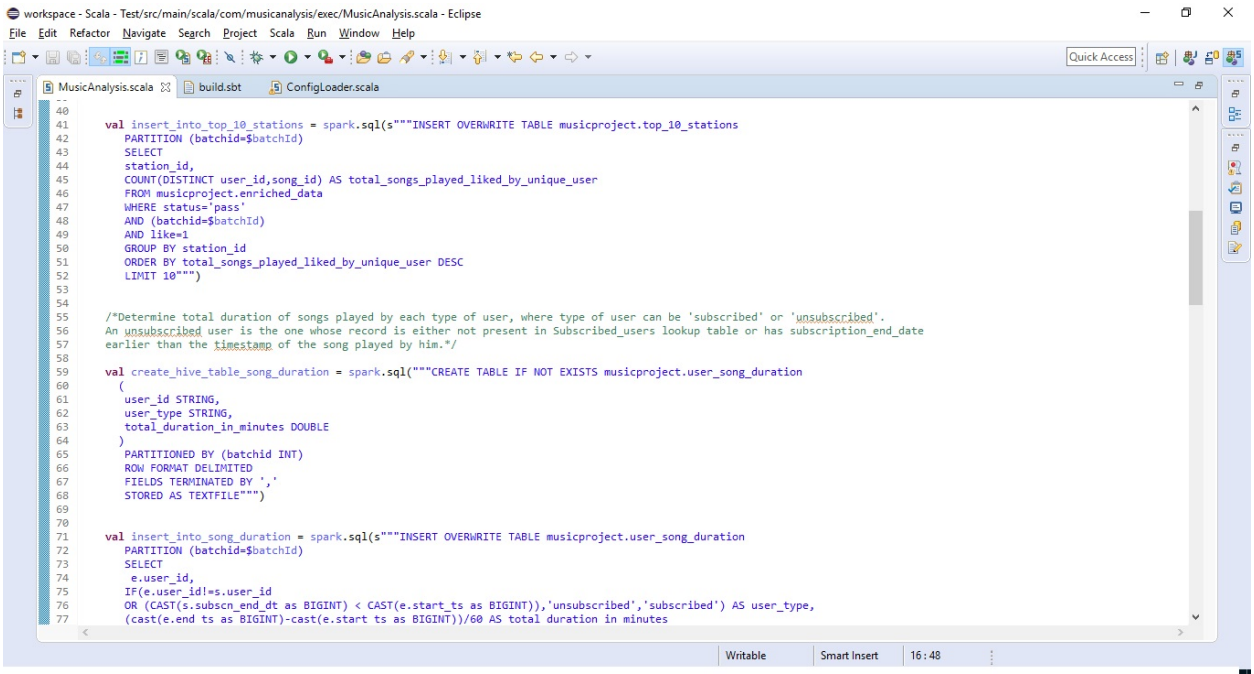
The results from analyzed data are stored in hive.

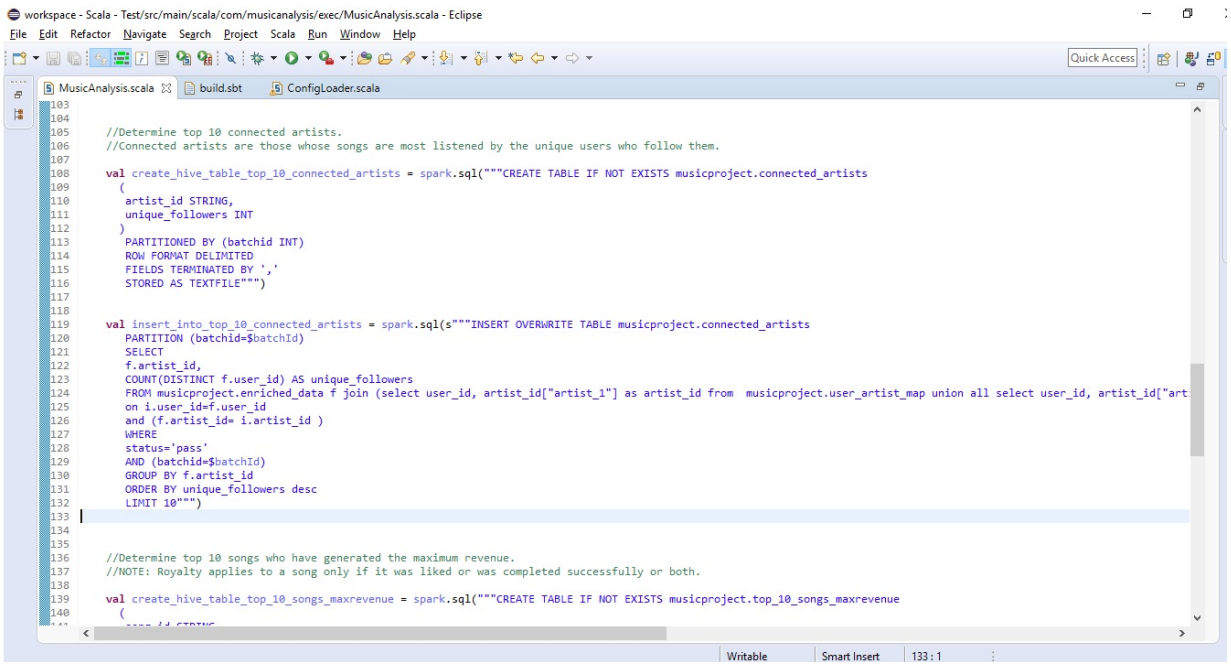


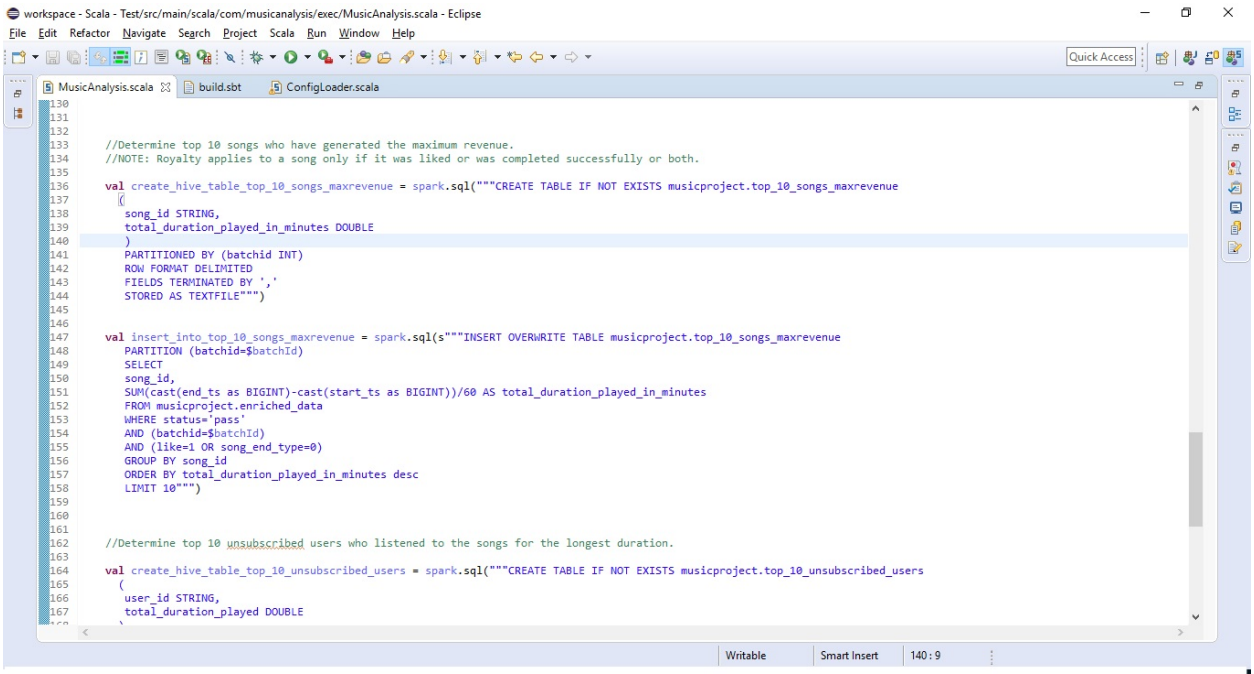
Spark code is packaged into jar using SBT, and below is source code

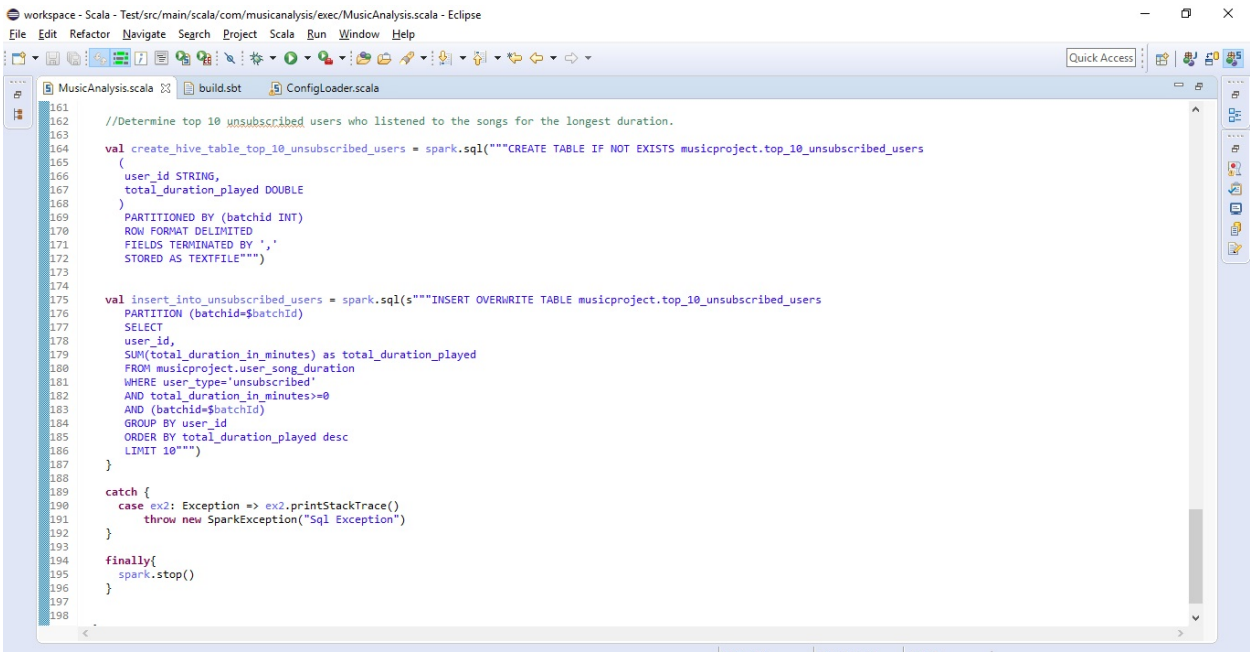
SPARK SOURCE CODE



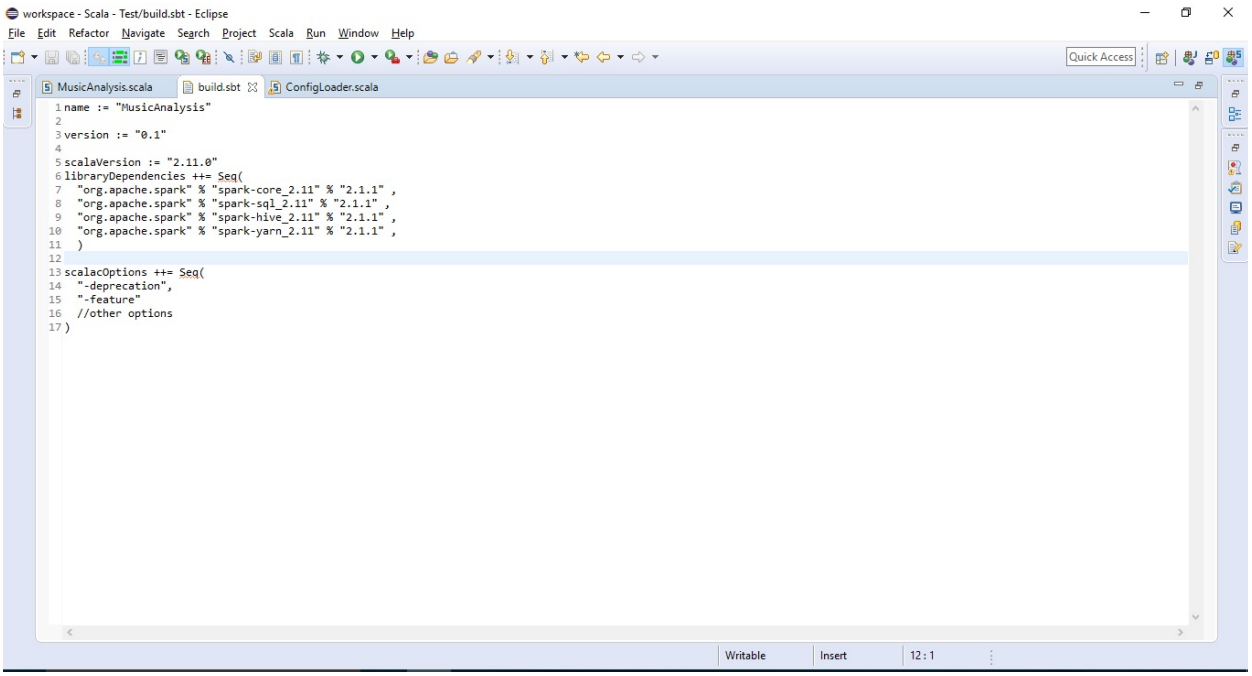



workspace - Scala - Test/src/main/scala/com/musicanalysis/exec/MusicAnalysis.scala - Eclipse
File Edit Refactor Navigate Search Project Scala Run Window Help
Quick Access
MusicAnalysis.scala
build.sbt
ConfigLoader.scala
//Determine top 10 connected artists.
//Connected artists are those whose songs are most listened by the unique users who follow them.
val create_hive_table_top_10_connected_artists = spark.sql("""CREATE TABLE IF NOT EXISTS musicproject.connected_artists
(artist_id STRING, unique_followers INT)
PARTITIONED BY (batchid INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE""")
val insert_into_top_10_connected_artists = spark.sql(s"""INSERT OVERWRITE TABLE musicproject.connected_artists
PARTITION (batchid=$batchId)
SELECT
f.artist_id,
COUNT(DISTINCT f.user_id) AS unique_followers
FROM musicproject.enriched_data f join (select user_id, artist_id["artist_1"] as artist_id from musicproject.user_artist_map union all select user_id, artist_id["art
on i.user_id=f.user_id
and (f.artist_id= i.artist_id)
WHERE
status='pass'
AND (batchid=$batchId)
GROUP BY f.artist_id
ORDER BY unique_followers desc
LIMIT 10""")
//Determine top 10 songs who have generated the maximum revenue.
//NOTE: Royalty applies to a song only if it was liked or was completed successfully or both.
val create_hive_table_top_10_songs_maxrevenue = spark.sql("""CREATE TABLE IF NOT EXISTS musicproject.top_10_songs_maxrevenue
Writable
Smart Insert
133 : 1


workspace - Scala - Test/src/main/scala/com/musicanalysis/exec/MusicAnalysis.scala - Eclipse
File Edit Refactor Navigate Search Project Scala Run Window Help
Quick Access
MusicAnalysis.scala
build.sbt
ConfigLoader.scala
//Determine top 10 songs who have generated the maximum revenue.
//NOTE: Royalty applies to a song only if it was liked or was completed successfully or both.
val create_hive_table_top_10_songs_maxrevenue = spark.sql("""CREATE TABLE IF NOT EXISTS musicproject.top_10_songs_maxrevenue
(song_id STRING, total_duration_played_in_minutes DOUBLE)
PARTITIONED BY (batchid INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE""")
val insert_into_top_10_songs_maxrevenue = spark.sql(s"""INSERT OVERWRITE TABLE musicproject.top_10_songs_maxrevenue
PARTITION (batchid=$batchId)
SELECT
song_id,
SUM(cast(end_ts as BIGINT)-cast(start_ts as BIGINT))/60 AS total_duration_played_in_minutes
FROM musicproject.enriched_data
WHERE status='pass'
AND (batchid=$batchId)
AND (like=1 OR song_end_type=0)
GROUP BY song_id
ORDER BY total_duration_played_in_minutes desc
LIMIT 10""")
//Determine top 10 unsubscribed users who listened to the songs for the longest duration.
val create_hive_table_top_10_unsubscribed_users = spark.sql("""CREATE TABLE IF NOT EXISTS musicproject.top_10_unsubscribed_users
(user_id STRING, total_duration_played DOUBLE)
Writable
Smart Insert
140 : 9

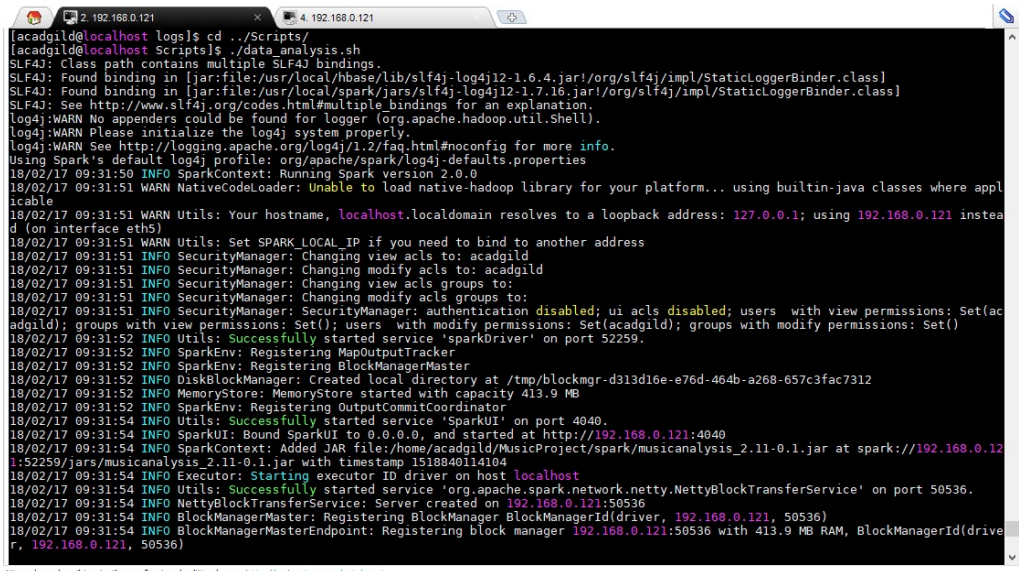


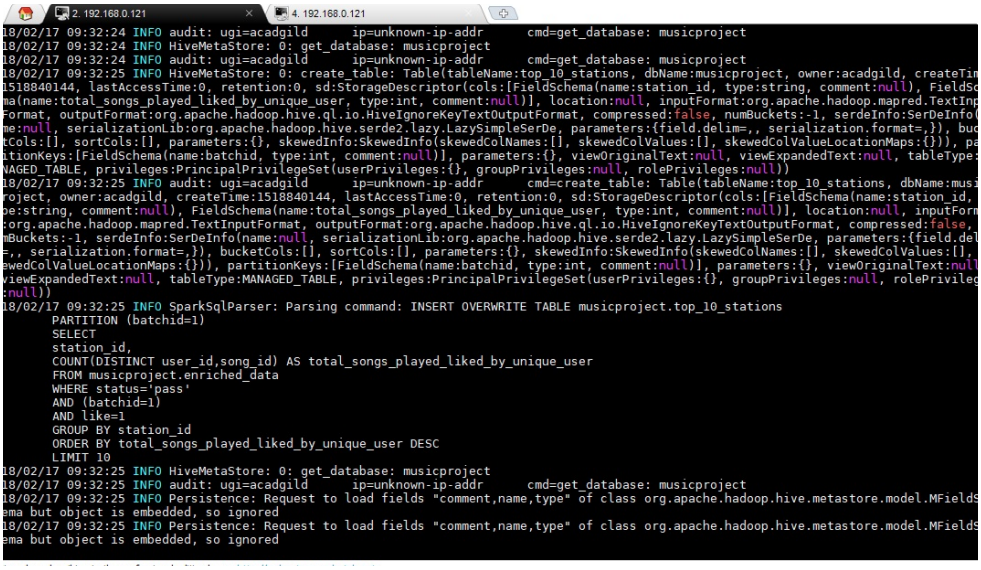
SBT for building dependency.



Jar is built from sbt package and jar is exported to vm for analysis

Spark analysis output





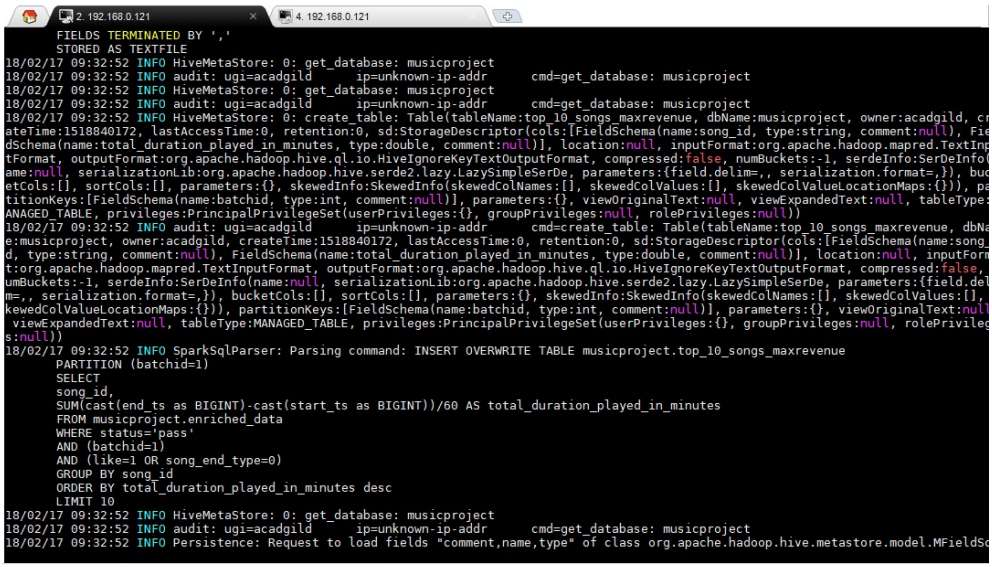
2. 192.168.0.121 | 4. 192.168.0.121

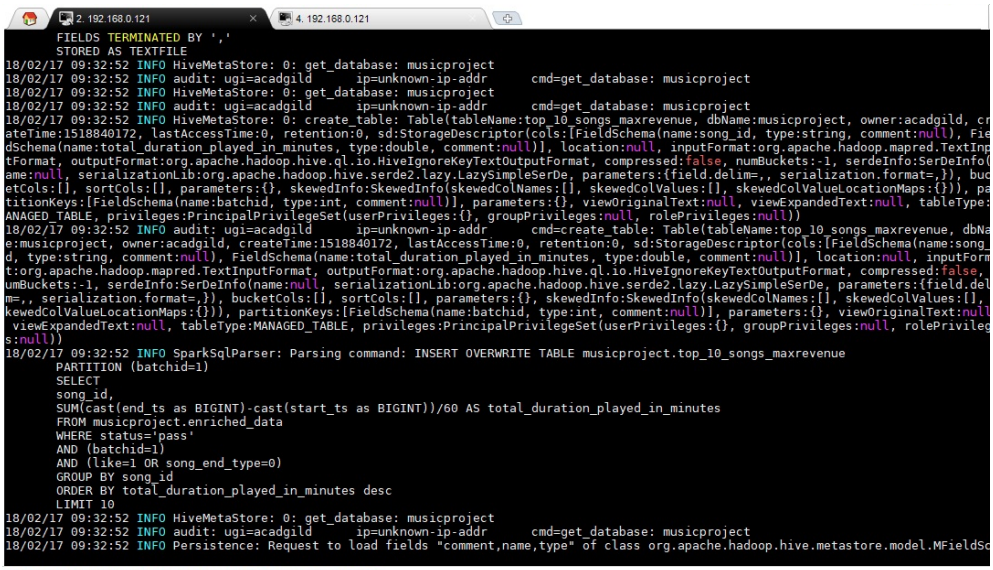
```
      FIELDS TERMINATED BY ','
      STORED AS TEXTFILE
18/02/17 09:32:46 INFO HiveMetaStore: 0: get_database: musicproject
18/02/17 09:32:46 INFO audit: ugi=acadgild      ip=unknown-ip-addr      cmd=get_database: musicproject
18/02/17 09:32:46 INFO HiveMetaStore: 0: get_database: musicproject
18/02/17 09:32:46 INFO audit: ugi=acadgild      ip=unknown-ip-addr      cmd=get_database: musicproject
18/02/17 09:32:46 INFO HiveMetaStore: 0: create_table: Table(tableName:user_type_duration, dbName:musicproject, owner:acadgild, createT
me:1518840166, lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:user_type, type:string, comment:null), FieldS
hema(name:total_duration_played, type:double, comment:null)], location:null, inputFormat:org.apache.hadoop.mapred.TextInputFormat, outp
tFormat:org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat, compressed:false, numBuckets:-1, serdeInfo:SerDeInfo(name:null, ser
alizationLib:org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, parameters:{field.delim=,, serialization.format=,}), bucketCols:[], so
tCols:[], parameters:{}, skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLocationMaps:{})), partitionKeys:[F
eldSchema(name:batchid, type:int, comment:null)], parameters:{}, viewOriginalText:null, viewExpandedText:null, tableType:MANAGED_TABLE,
privileges:PrincipalPrivilegeSet(userPrivileges:{}, groupPrivileges:null, rolePrivileges:null))
18/02/17 09:32:46 INFO audit: ugi=acadgild      ip=unknown-ip-addr      cmd=create_table: Table(tableName:user_type_duration, dbName:mu
icproject, owner:acadgild, createTime:1518840166, lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:user_type,
type:string, comment:null), FieldSchema(name:total_duration_played, type:double, comment:null)], location:null, inputFormat:org.apache.
adoop.mapred.TextInputFormat, outputFormat:org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat, compressed:false, numBuckets:-1,
serdeInfo:SerDeInfo(name:null, serializationLib:org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, parameters:{field.delim=,, serializ
tion.format=,}), bucketCols:[], sortCols:[], parameters:{}, skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValue
ocationMaps:{})), partitionKeys:[FieldSchema(name:batchid, type:int, comment:null)], parameters:{}, viewOriginalText:null, viewExpanded
ext:null, tableType:MANAGED_TABLE, privileges:PrincipalPrivilegeSet(userPrivileges:{}, groupPrivileges:null, rolePrivileges:null))
18/02/17 09:32:46 INFO SparkSqlParser: Parsing command: INSERT OVERWRITE TABLE musicproject.user_type_duration
      PARTITION (batchid=1)
      SELECT
      user_type,
      sum(total_duration_in_minutes) as total_duration_played
      FROM musicproject.user_song_duration
      WHERE (batchid=1)
       GROUP BY user_type
18/02/17 09:32:46 INFO HiveMetaStore: 0: get_database: musicproject
18/02/17 09:32:46 INFO audit: ugi=acadgild      ip=unknown-ip-addr      cmd=get_database: musicproject
18/02/17 09:32:46 INFO Persistence: Request to load fields "comment,name,type" of class org.apache.hadoop.hive.metastore.model.MFieldSc
ema but object is embedded, so ignored
18/02/17 09:32:46 INFO Persistence: Request to load fields "comment,name,type" of class org.apache.hadoop.hive.metastore.model.MFieldSc
ema but object is embedded, so ignored
18/02/17 09:32:46 INFO Persistence: Request to load fields "comment,name,type" of class org.apache.hadoop.hive.metastore.model.MFieldSc
ema but object is embedded, so ignored
```

(term by subscribing to the professional edition here: http://mobaxterm.mobatek.net

2. 192.168.0.121 | 4. 192.168.0.121

```
18/02/17 09:32:49 INFO audit: ugi=acadgild      ip=unknown-ip-addr      cmd=get_database: musicproject
18/02/17 09:32:49 INFO HiveMetaStore: 0: create_table: Table(tableName:connected_artists, dbName:musicproject, owner:acadgild, createTi
e:1518840169, lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:artist_id, type:string, comment:null), FieldS
ema(name:unique_followers, type:int, comment:null)], location:null, inputFormat:org.apache.hadoop.mapred.TextInputFormat, outputFormat
rg.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat, compressed:false, numBuckets:-1, serdeInfo:SerDeInfo(name:null, serializati
Lib:org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, parameters:{field.delim=,, serialization.format=,}), bucketCols:[], sortCols:[
 parameters:{}, skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLocationMaps:{})), partitionKeys:[FieldSche
(name:batchid, type:int, comment:null)], parameters:{}, viewOriginalText:null, viewExpandedText:null, tableType:MANAGED_TABLE, privile
s:PrincipalPrivilegeSet(userPrivileges:{}, groupPrivileges:null, rolePrivileges:null))
18/02/17 09:32:49 INFO audit: ugi=acadgild      ip=unknown-ip-addr      cmd=create_table: Table(tableName:connected_artists, dbName:mus
cproject, owner:acadgild, createTime:1518840169, lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:artist_id,
ype:string, comment:null), FieldSchema(name:unique_followers, type:int, comment:null)], location:null, inputFormat:org.apache.hadoop.ma
red.TextInputFormat, outputFormat:org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat, compressed:false, numBuckets:-1, serdeIn
:SerDeInfo(name:null, serializationLib:org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, parameters:{field.delim=,, serialization.fo
at=,}), bucketCols:[], sortCols:[], parameters:{}, skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLocation
ps:{})), partitionKeys:[FieldSchema(name:batchid, type:int, comment:null)], parameters:{}, viewOriginalText:null, viewExpandedText:nul
 tableType:MANAGED_TABLE, privileges:PrincipalPrivilegeSet(userPrivileges:{}, groupPrivileges:null, rolePrivileges:null))
18/02/17 09:32:49 INFO SparkSqlParser: Parsing command: INSERT OVERWRITE TABLE musicproject.connected_artists
      PARTITION (batchid=1)
      SELECT
      artist_id,
      COUNT(DISTINCT user_id) AS unique_followers
      FROM musicproject.enriched_data
      WHERE status='pass'
      AND (batchid=1)
      GROUP BY artist_id
      ORDER BY unique_followers desc
      LIMIT 10
18/02/17 09:32:49 INFO HiveMetaStore: 0: get_database: musicproject
18/02/17 09:32:49 INFO audit: ugi=acadgild      ip=unknown-ip-addr      cmd=get_database: musicproject
18/02/17 09:32:50 INFO Persistence: Request to load fields "comment,name,type" of class org.apache.hadoop.hive.metastore.model.MFieldS
ema but object is embedded, so ignored
18/02/17 09:32:50 INFO Persistence: Request to load fields "comment,name,type" of class org.apache.hadoop.hive.metastore.model.MFieldS
ema but object is embedded, so ignored
18/02/17 09:32:50 INFO Persistence: Request to load fields "comment,name,type" of class org.apache.hadoop.hive.metastore.model.MFieldS
ema but object is embedded, so ignored
18/02/17 09:32:50 INFO Persistence: Request to load fields "comment,name,type" of class org.apache.hadoop.hive.metastore.model.MFieldS
ema but object is embedded, so ignored
```





We can now see the results in Hive.

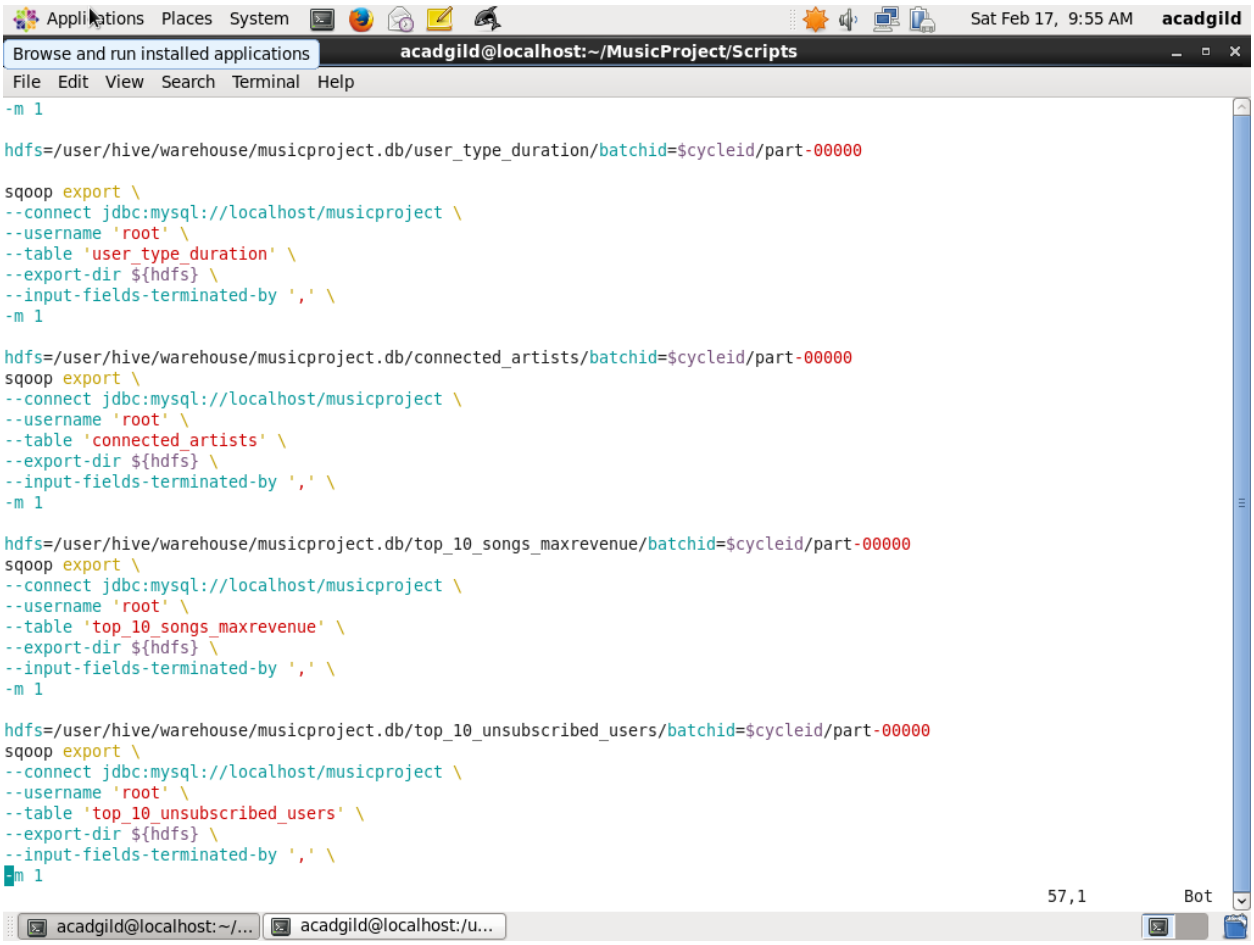
```
hive> use musicproject
    > ;
OK
Time taken: 1.101 seconds
hive> show tables;
OK
connected_artists
enriched_data
merged_data
song_artist_map
station_geo_map
subscribed_users
temp_enriched_data
top_10_songs_maxrevenue
top_10_stations
top_10_unsubscribed_users
user_artist_map
user_song_duration
user_type_duration
web_formatted_data
Time taken: 0.584 seconds, Fetched: 14 row(s)
hive> select * from top_10_stations;
OK
ST402   2       1
ST411   2       1
ST410   1       1
ST405   1       1
ST415   1       1
ST404   1       1
Time taken: 1.061 seconds, Fetched: 6 row(s)
```

```
hive> select * from user_song_duration;
OK
U101    subscribed      166666.66666666666      1
U102    subscribed      480116.76666666666      1
U103    unsubscribed    -480116.76666666666     1
U104    subscribed      -166666.66666666666     1
U105    subscribed      -1666.6666666666667     1
U105    unsubscribed    -333333.3333333333      1
U106    subscribed      -43788.23333333333      1
U107    unsubscribed    -87193.78333333334      1
U107    unsubscribed    0.0     1
U108    subscribed      87193.78333333334       1
U108    subscribed      166666.66666666666      1
U110    unsubscribed    -436711.2166666667      1
U113    subscribed      -43788.23333333333      1
U113    subscribed      436711.2166666667       1
U113    subscribed      0.0     1
U113    unsubscribed    -480116.76666666666     1
U114    subscribed      165000.0        1
U114    subscribed      523905.0        1
U115    subscribed      -480116.76666666666     1
U116    subscribed      -166666.66666666666     1
U118    subscribed      0.0     1
U118    subscribed      1666.6666666666667      1
U119    subscribed      0.0     1
U120    subscribed      43405.55        1
U120    subscribed      -333333.3333333333      1
Time taken: 0.177 seconds, Fetched: 25 row(s)
```

```
hive> select * from top_10_songs_maxrevenue;
OK
S209    523905.0        1
S205    166666.66666666666      1
S207    165000.0        1
S200    87193.78333333334       1
S210    0.0     1
S203    -43788.23333333333      1
S202    -357238.3333333333      1
S206    -377121.56666666665     1
S204    -480116.76666666666     1
Time taken: 0.149 seconds, Fetched: 9 row(s)
hive> select * from connected_artists;
OK
A303    8       1
A300    7       1
A302    3       1
A304    3       1
A305    1       1
A301    1       1
Time taken: 0.209 seconds, Fetched: 6 row(s)
hive> select * from top_10_unsubscribed_users;
OK
U107    0.0     1
Time taken: 0.158 seconds, Fetched: 1 row(s)
hive>
```

6. Export data to mysql

In this stage data will be exported from hive warehouse directory to MYSQL database using data_export.sh


Applications Places System
Sat Feb 17, 9:55 AM acadgild
Browse and run installed applications
acadgild@localhost:~/MusicProject/Scripts
File Edit View Search Terminal Help
-m 1
hdfs=/user/hive/warehouse/musicproject.db/user_type_duration/batchid=$cycleid/part-00000
sqoop export \
--connect jdbc:mysql://localhost/musicproject \
--username 'root' \
--table 'user_type_duration' \
--export-dir ${hdfs} \
--input-fields-terminated-by ',' \
-m 1
hdfs=/user/hive/warehouse/musicproject.db/connected_artists/batchid=$cycleid/part-00000
sqoop export \
--connect jdbc:mysql://localhost/musicproject \
--username 'root' \
--table 'connected_artists' \
--export-dir ${hdfs} \
--input-fields-terminated-by ',' \
-m 1
hdfs=/user/hive/warehouse/musicproject.db/top_10_songs_maxrevenue/batchid=$cycleid/part-00000
sqoop export \
--connect jdbc:mysql://localhost/musicproject \
--username 'root' \
--table 'top_10_songs_maxrevenue' \
--export-dir ${hdfs} \
--input-fields-terminated-by ',' \
-m 1
hdfs=/user/hive/warehouse/musicproject.db/top_10_unsubscribed_users/batchid=$cycleid/part-00000
sqoop export \
--connect jdbc:mysql://localhost/musicproject \
--username 'root' \
--table 'top_10_unsubscribed_users' \
--export-dir ${hdfs} \
--input-fields-terminated-by ',' \
-m 1
57,1 Bot
acadgild@localhost:~/...
acadgild@localhost:/u...

Log after running SQOOP

```
2018-02-17 10:01:49,904 INFO  [main] Configuration.deprecation: mapred.job.name is deprecated. Instead, use mapreduce.job.nam
e
2018-02-17 10:01:49,904 INFO  [main] Configuration.deprecation: mapred.cache.files.timestamps is deprecated. Instead, use map
reduce.job.cache.files.timestamps
2018-02-17 10:01:49,904 INFO  [main] Configuration.deprecation: mapreduce.map.class is deprecated. Instead, use mapreduce.job
.map.class
2018-02-17 10:01:49,905 INFO  [main] Configuration.deprecation: mapred.input.dir is deprecated. Instead, use mapreduce.input.
fileinputformat.inputdir
2018-02-17 10:01:49,905 INFO  [main] Configuration.deprecation: mapred.map.tasks.speculative.execution is deprecated. Instead
, use mapreduce.map.speculative
2018-02-17 10:01:49,911 INFO  [main] Configuration.deprecation: mapreduce.inputformat.class is deprecated. Instead, use mapre
duce.job.inputformat.class
2018-02-17 10:01:49,912 INFO  [main] Configuration.deprecation: mapreduce.outputformat.class is deprecated. Instead, use mapr
educe.job.outputformat.class
2018-02-17 10:01:49,912 INFO  [main] Configuration.deprecation: mapred.cache.files is deprecated. Instead, use mapreduce.job.
cache.files
2018-02-17 10:01:49,912 INFO  [main] Configuration.deprecation: mapred.working.dir is deprecated. Instead, use mapreduce.job.
working.dir
2018-02-17 10:01:49,913 INFO  [main] Configuration.deprecation: mapred.mapoutput.value.class is deprecated. Instead, use mapr
educe.map.output.value.class
2018-02-17 10:01:49,914 INFO  [main] Configuration.deprecation: mapred.mapoutput.key.class is deprecated. Instead, use mapred
uce.map.output.key.class
2018-02-17 10:01:49,914 INFO  [main] Configuration.deprecation: mapred.job.classpath.files is deprecated. Instead, use mapred
uce.job.classpath.files
2018-02-17 10:01:49,914 INFO  [main] Configuration.deprecation: user.name is deprecated. Instead, use mapreduce.job.user.name
2018-02-17 10:01:49,914 INFO  [main] Configuration.deprecation: mapred.reduce.tasks is deprecated. Instead, use mapreduce.job
.reduces
2018-02-17 10:01:49,914 INFO  [main] Configuration.deprecation: mapred.cache.files.filesizes is deprecated. Instead, use mapr
educe.job.cache.files.filesizes
2018-02-17 10:01:50,154 INFO  [main] mapreduce.JobSubmitter: Submitting tokens for job: job_1518765191740_0083
2018-02-17 10:01:50,654 INFO  [main] impl.YarnClientImpl: Submitted application application_1518765191740_0083 to ResourceMan
ager at /0.0.0.0:8032
2018-02-17 10:01:50,842 INFO  [main] mapreduce.Job: The url to track the job: http://http://localhost:8088/proxy/application_
1518765191740_0083/
2018-02-17 10:01:50,845 INFO  [main] mapreduce.Job: Running job: job_1518765191740_0083
2018-02-17 10:02:01,286 INFO  [main] mapreduce.Job: Job job_1518765191740_0083 running in uber mode : false
2018-02-17 10:02:01,289 INFO  [main] mapreduce.Job:  map 0% reduce 0%
2018-02-17 10:02:12,113 INFO  [main] mapreduce.Job:  map 100% reduce 0%
2018-02-17 10:02:13,197 INFO  [main] mapreduce.Job: Job job_1518765191740_0083 completed successfully
```

All the five result are exported to mysql, now we can see the results in mysql

```
[acadgild@localhost Scripts]$ mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 692
Server version: 5.1.73 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use musicproject;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+--------------------------+
| Tables_in_musicproject   |
+--------------------------+
| connected_artists        |
| top_10_songs_maxrevenue  |
| top_10_stations          |
| top_10_unsubscribed_users |
| user_type_duration       |
+--------------------------+
5 rows in set (0.00 sec)
```

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

```
mysql> select * from top_10_stations;
+------------+-----------------------------------------+
| station_id | total_songs_played_liked_by_unique_user |
+------------+-----------------------------------------+
| ST402      |                                       2 |
| ST411      |                                       2 |
| ST410      |                                       1 |
| ST405      |                                       1 |
| ST415      |                                       1 |
| ST404      |                                       1 |
+------------+-----------------------------------------+
6 rows in set (0.01 sec)
```

- Determine total duration of songs played by each type of user, where type of user can be 'subscribed' or 'unsubscribed'.

```
mysql> select * from user_type_duration;
+--------------+----------------------+
| user_type    | total_duration_played |
+--------------+----------------------+
| subscribed   |             835305.75 |
| unsubscribed |     -1817471.86666667 |
+--------------+----------------------+
2 rows in set (0.00 sec)
```

- Determine top 10 connected artists. Connected artists are those whose songs are most listened by the unique users who follow them. (use who follow them is obtained from user_artists_map)

```
mysql> select * from connected_artists;
+-----------+------------------+
| artist_id | unique_followers |
+-----------+------------------+
| A302      |                1 |
| A304      |                1 |
+-----------+------------------+
2 rows in set (0.01 sec)

mysql>
```

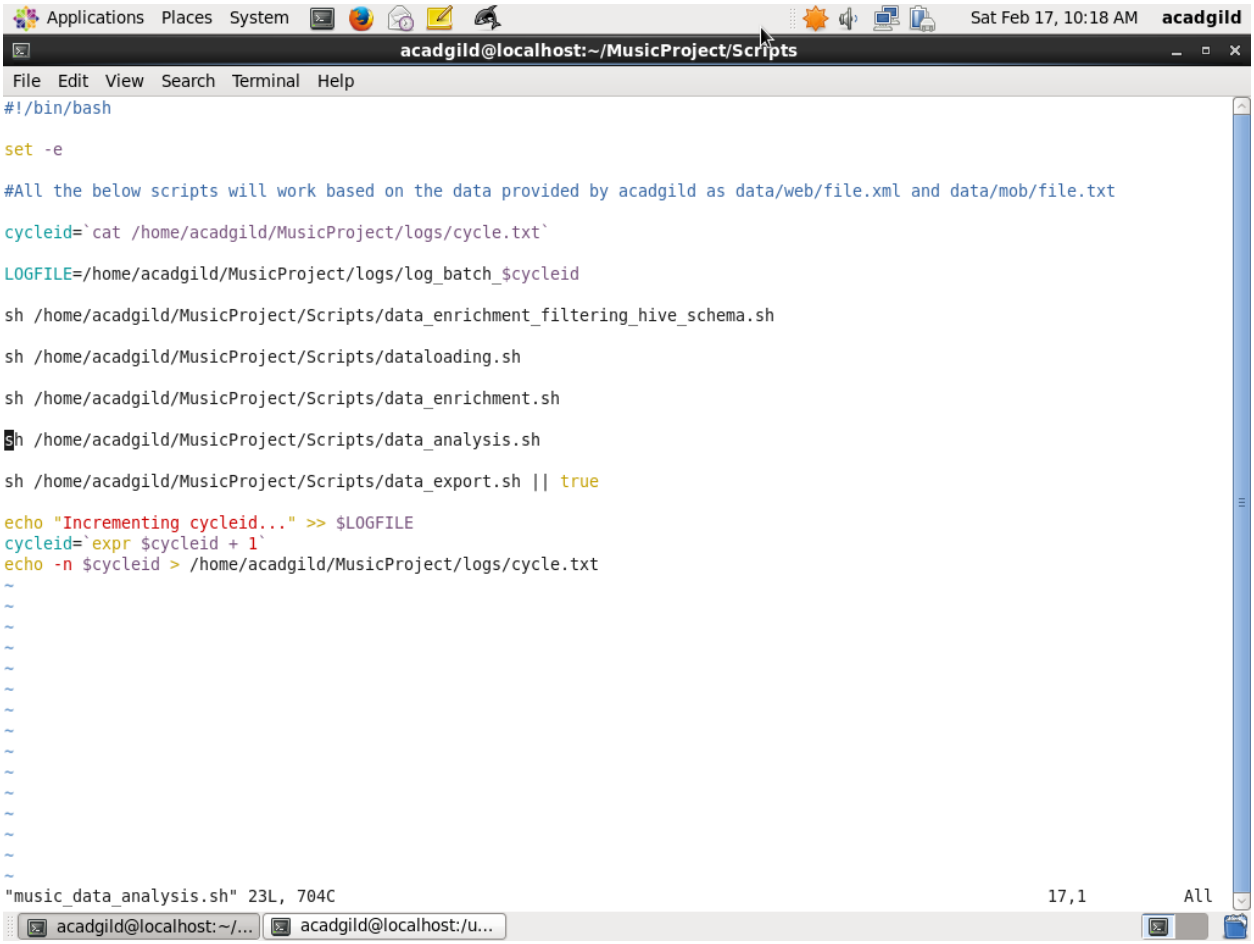
- Determine top 10 songs who have generated the maximum revenue. Royalty applies to a song only if it was liked or was completed successfully or both.

```
mysql> select * from top_10_songs_maxrevenue;
+---------+---------------------------------+
| song_id | total_duration_played_in_minutes |
+---------+---------------------------------+
| S209    |                          523905 |
| S205    |                166666.666666667 |
| S207    |                          165000 |
| S200    |                87193.7833333333 |
| S210    |                               0 |
| S203    |               -43788.2333333333 |
| S202    |               -357238.333333333 |
| S206    |               -377121.566666667 |
| S204    |               -480116.766666667 |
+---------+---------------------------------+
9 rows in set (0.00 sec)
```

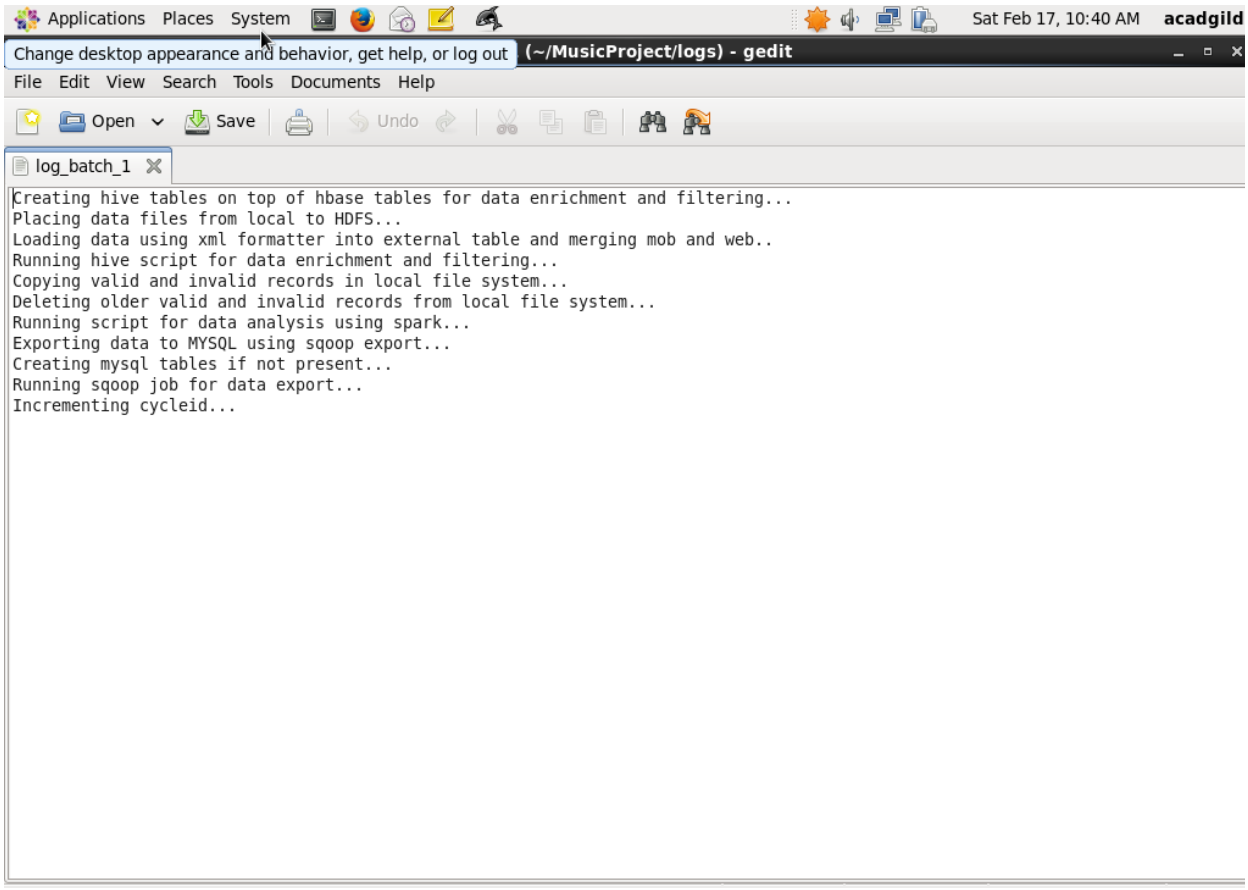
- Determine top 10 unsubscribed users who listened to the songs for the longest duration.

```
mysql> select * from top_10_unsubscribed_users;
+---------+----------------------+
| user_id | total_duration_played |
+---------+----------------------+
| U107    |                    0 |
+---------+----------------------+
1 row in set (0.00 sec)
```

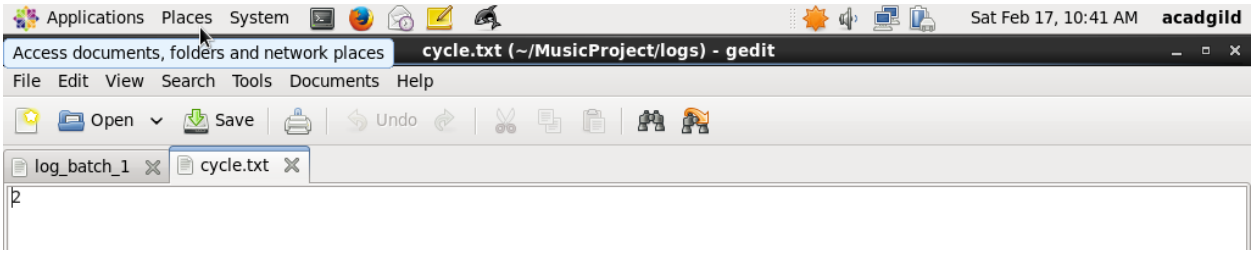
Now we can put all this scripts into one wrapper script, music_data_analysis.sh and increment the cycleid every time we run the entire process.



After running this script, below is the screenshot of the log generated



And Cycleid is incremented to 2 and ready for next batch of processing



This script can be scheduled in cron to run for every three hours to do music data analysis for the arriving files.

```
[acadgild@localhost lib]$ crontab -l
* 3 * * * /home/acadgild/MusicProject/Scripts/music_data_analysis.sh >> /home/acadgild/MusicProject/logs/MusicAnalysisSchedul
e.log
[acadgild@localhost lib]$
```

Conclusion

Music analysis is successfully performed using music_analysis_script.sh which will run every 3 hours for and provide the result using big data.