

Music Data Analysis

Project:

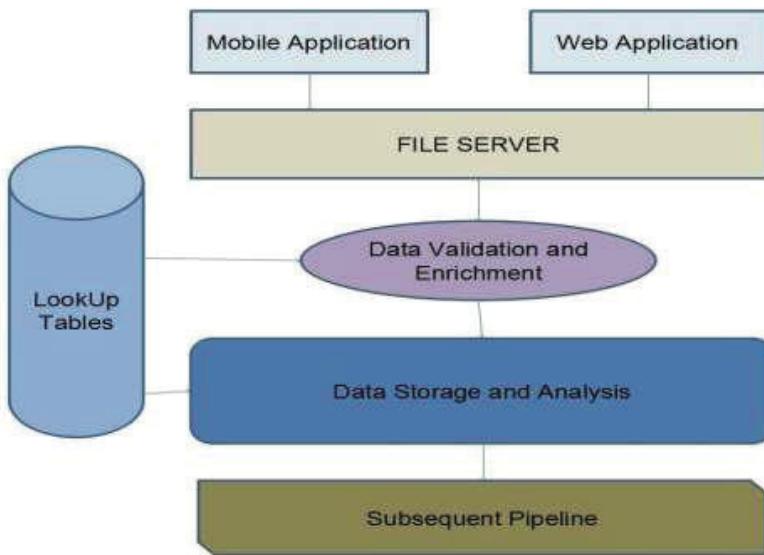
A leading music-catering company is planning to analyze large amount of data received from varieties of sources, namely mobile app and website to track the behavior of users, classify users, calculate royalties associated with the song and make appropriate business strategies. The file server receives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

DATASET:

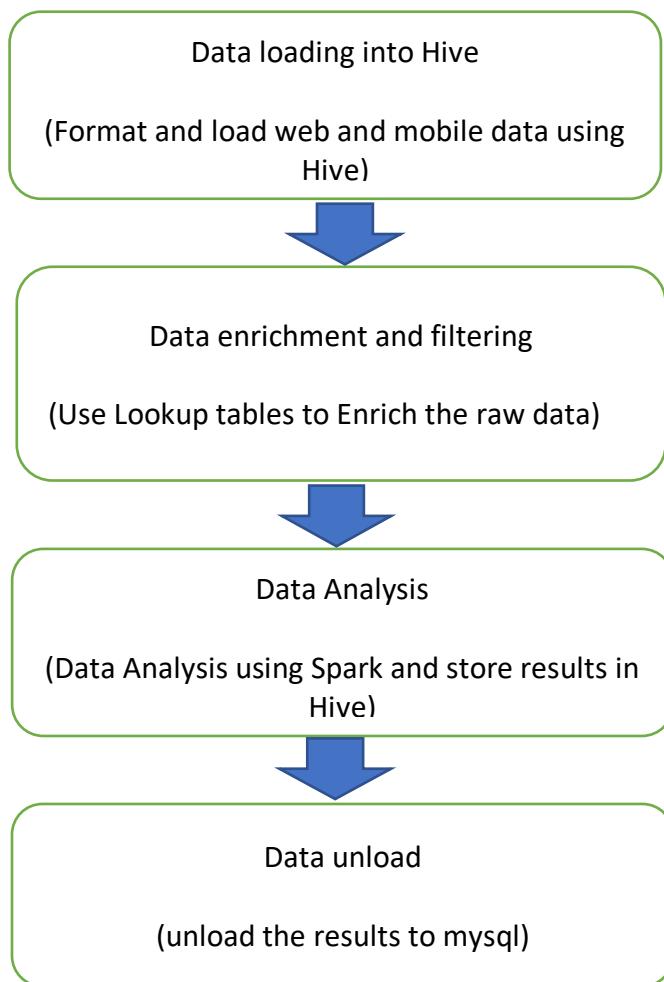
1. Data coming from web applications reside in /data/web and has xml format.
2. Data coming from mobile applications reside in /data/mob and has csv format.
3. Data present in lookup directory should be used in HBase.



Data Flow:



Design and Approach:



Softwares

OS: Centos

Hadoop 2.6

Hive 1.2.1

Sqoop 1.4.5

Mysql 5.1.73

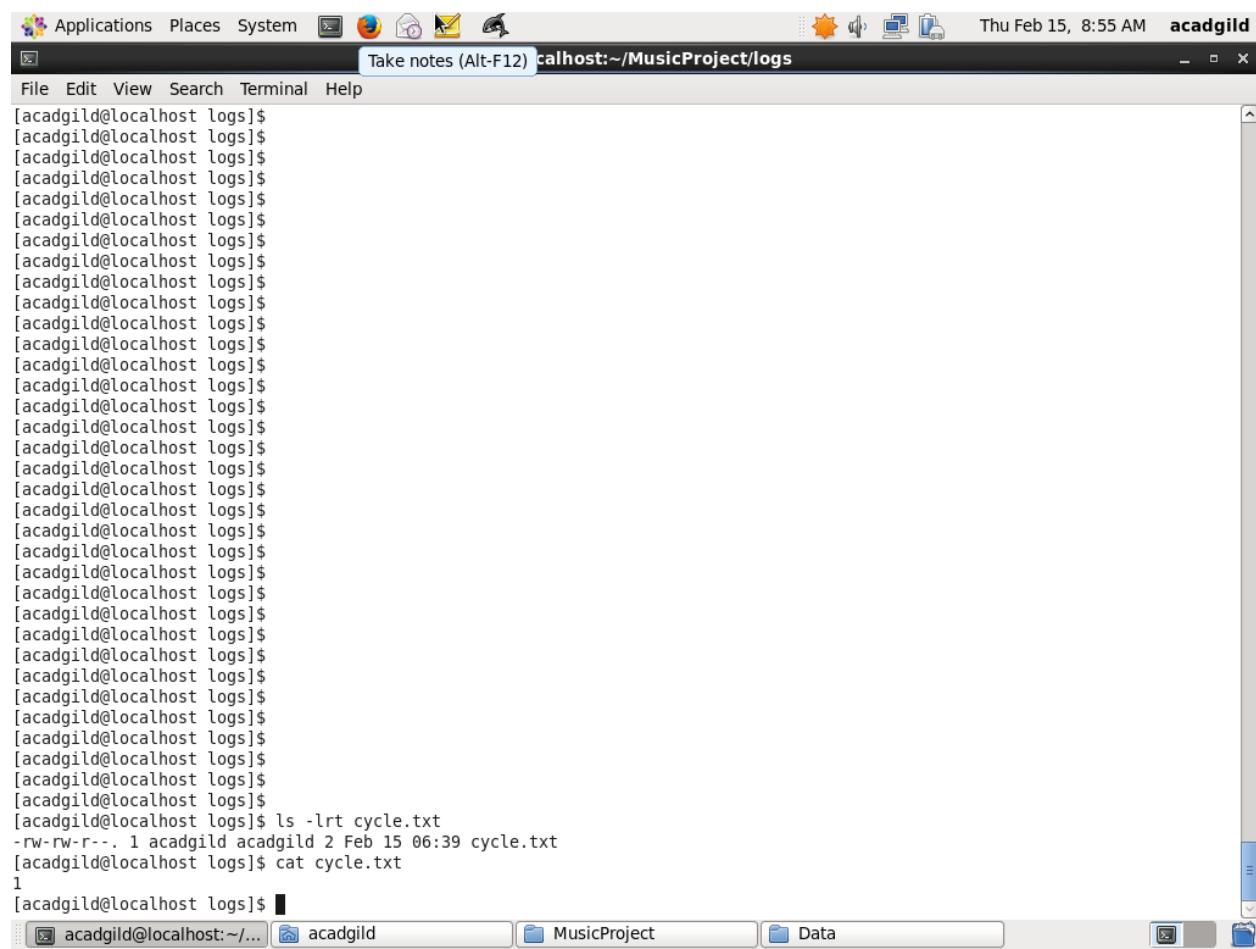
Hbase 0.99.14-hadoop2

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



The screenshot shows a terminal window titled "Take notes (Alt-F12) calhost:~/MusicProject/logs". The terminal is running on a Centos system, as indicated by the desktop environment icons at the top. The command history shows the user navigating through a directory of logs and then executing several commands to create and update a file named "cycle.txt". The terminal output is as follows:

```
[acadgild@localhost logs]$  
[acadgild@localhost logs]$ ls -lrt cycle.txt  
-rw-rw-r--. 1 acadgild acadgild 2 Feb 15 06:39 cycle.txt  
[acadgild@localhost logs]$ cat cycle.txt  
1  
[acadgild@localhost logs]$
```

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

The screenshot shows a Linux desktop environment with a terminal window open. The terminal window has a title bar "Change desktop appearance and behavior, get help, or log out calhost:~/MusicProject/Scripts". The window contains a shell script being run. The script reads from several files (song-artist.txt, user-subscn.txt, user-artist.txt) and writes to a file (user-artist1.txt). It uses awk, cut, echo, and hbase shell commands. The terminal window is part of a desktop interface with icons for Applications, Places, System, and various system status indicators.

```
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 $line | cut -d',' -f3`
    echo "put 'subscribed_users', '$userid', 'subscn:startdt', '$startdt'" | hbase shell
    echo "put 'subscribed_users', '$userid', 'subscn:enddt', '$enddt'" | hbase shell
done <"$file"

file="/home/acadgild/MusicProject/Data/LookUp/user-artist.txt"
touch "/home/acadgild/MusicProject/Data/LookUp/user-artist1.txt"
chmod 775 "/home/acadgild/MusicProject/Data/LookUp/user-artist1.txt"
file1="/home/acadgild/MusicProject/Data/LookUp/user-artist1.txt"
awk '$1=$1' FS="&" OFS=" " $file > $file1
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', 'user:artist_$num', '$row'" | hbase shell
        let "num=num+1"
    done
    num=1
done <"$file1"
rm "/home/acadgild/MusicProject/Data/LookUp/user-artist1.txt"
```

Logs after running the script

Applications Places System Firefox Web Browser acadgild@localhost:~/MusicProject/Scripts - x

File Edit View Search Term Browse the Web

available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.98.14-hadoop2, r4e4aab93b52f1b0fef6b66edd06ec8923014dec, Tue Aug 25 22:35:44 PDT 2015

```
put 'user_artist_map', 'U113', 'user:artist_2', 'A302'
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 07:25:01,938 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable
0 row(s) in 0.5980 seconds

2018-02-15 07:25:08,354 INFO [main] Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.98.14-hadoop2, r4e4aab93b52f1b0fef6b66edd06ec8923014dec, Tue Aug 25 22:35:44 PDT 2015

put 'user_artist_map', 'U114', 'user:artist_1', 'A300'
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 07:25:11,623 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable
0 row(s) in 0.5970 seconds

2018-02-15 07:25:18,139 INFO [main] Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.98.14-hadoop2, r4e4aab93b52f1b0fef6b66edd06ec8923014dec, Tue Aug 25 22:35:44 PDT 2015

put 'user_artist_map', 'U114', 'user:artist_2', 'A301
SLF4J: Class path contains multiple SLF4J bindings.
```

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 b
uiltin-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
S200          COLUMN+CELL
S201          column=artist:artistid, timestamp=1518658921151, value=A300
S202          column=artist:artistid, timestamp=1518658933836, value=A301
S203          column=artist:artistid, timestamp=1518658943228, value=A302
S204          column=artist:artistid, timestamp=1518658952787, value=A303
S205          column=artist:artistid, timestamp=1518658962804, value=A304
S206          column=artist:artistid, timestamp=1518658973712, value=A301
S207          column=artist:artistid, timestamp=1518658984386, value=A302
S208          column=artist:artistid, timestamp=1518658995630, value=A303
S209          column=artist:artistid, timestamp=1518659006569, value=A304
                           column=artist:artistid, timestamp=1518659017215, value=A305
10 row(s) in 0.4060 seconds

hbase(main):008:0> scan 'station_geo_map'
ROW
ST400         COLUMN+CELL
ST401         column=geo:geo_cd, timestamp=1518658759193, value=A
ST402         column=geo:geo_cd, timestamp=1518658769757, value=AU
ST403         column=geo:geo_cd, timestamp=1518658781201, value=AP
ST404         column=geo:geo_cd, timestamp=1518658792119, value=J
ST405         column=geo:geo_cd, timestamp=1518658803345, value=E
ST406         column=geo:geo_cd, timestamp=1518658814863, value=A
ST407         column=geo:geo_cd, timestamp=1518658826254, value=AU
ST408         column=geo:geo_cd, timestamp=1518658836769, value=AP
ST409         column=geo:geo_cd, timestamp=1518658847318, value=E
ST410         column=geo:geo_cd, timestamp=1518658857438, value=E
ST411         column=geo:geo_cd, timestamp=1518658868713, value=A
ST412         column=geo:geo_cd, timestamp=1518658878990, value=A
ST413         column=geo:geo_cd, timestamp=151865889106, value=AP
ST414         column=geo:geo_cd, timestamp=1518658900033, value=J
                           column=geo:geo_cd, timestamp=1518658910958, value=E
15 row(s) in 0.2760 seconds

hbase(main):009:0> scan 'subscribed_users'
ROW
U100          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 will 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

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.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
OK
Time taken: 1.235 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

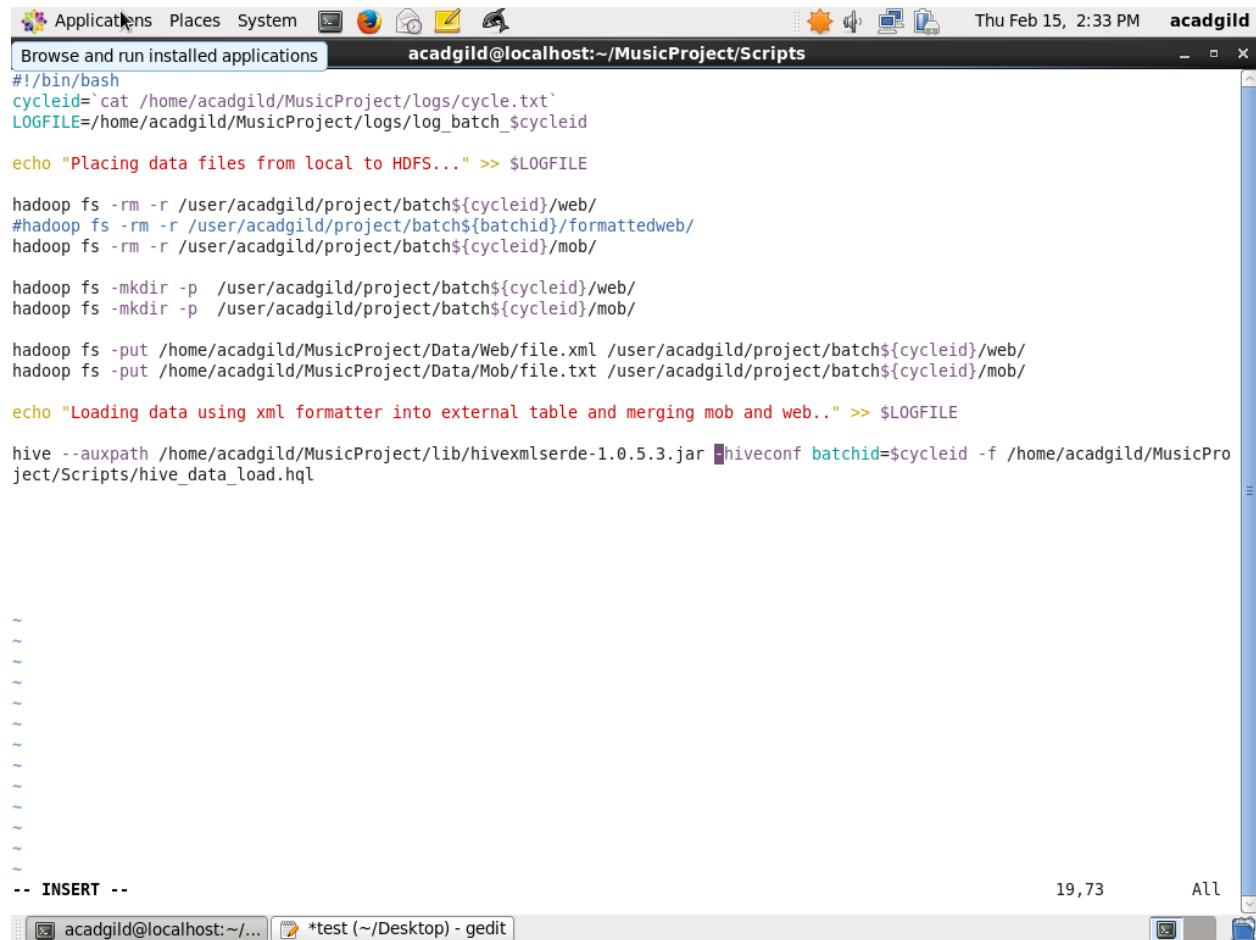
Logging initialized using configuration in jar:file:/usr/local/hive/lib/hive-common-0.14.0.jar!/hive-log4j.properties
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/hive-jdbc-0.14.0-standalone.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
hive> show databases;
OK
b1
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



The screenshot shows a terminal window titled "acadgild@localhost:~/MusicProject/Scripts". The window contains the following command-line session:

```
#!/bin/bash
cycleid=`cat /home/acadgild/MusicProject/logs/cycle.txt`
LOGFILE=/home/acadgild/MusicProject/logs/log_batch_${cycleid}

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

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

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

hadoop fs -put /home/acadgild/MusicProject/Data/Web/file.xml /user/acadgild/project/batch${cycleid}/web/
hadoop fs -put /home/acadgild/MusicProject/Data/Mob/file.txt /user/acadgild/project/batch${cycleid}/mob/

echo "Loading data using xml formatter into external table and merging mob and web.." >> $LOGFILE

hive --auxpath /home/acadgild/MusicProject/lib/hivexmlserde-1.0.5.3.jar --hiveconf batchid=$cycleid -f /home/acadgild/MusicProject/Scripts/hive_data_load.hql
```

The terminal window also displays the file contents of "hive_data_load.hql" at the bottom:

```
-- INSERT --
```

```

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"="","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});

```

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/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
OK
Time taken: 1.091 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]$
```



```

Applications Places System Terminal adgild@localhost:~/MusicProject/Scripts
Thu Feb 15, 2:52 PM acadgild
hive> show tables;
OK
merged_data
song_artist_map
station_geo_map
subscribed_users
user_artist_map
web_formatted_data
Time taken: 0.036 seconds, Fetched: 6 row(s)
hive> select * from merged_data;
OK
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| U106 | S205 | A300 | 1451197800 | 1451197800 | 1483237800 | AP | ST407 | 2 | 1 | 1 | 1 |
| U114 | S209 | A303 | 1451233800 | 1451197800 | 1483237800 | U | ST411 | 2 | 1 | 0 | 1 |
| U113 | S203 | A304 | 1451233800 | 1451233800 | 1451197800 | U | ST405 | 0 | 0 | 1 | 1 |
| U108 | S200 | A302 | 1451158200 | 1451197800 | 1451158200 | U | ST414 | 0 | 0 | 1 | 1 |
| U102 | S203 | A305 | 1451233800 | 1451233800 | 1483237800 | U | ST404 | 2 | 0 | 0 | 1 |
| NULL | S208 | A300 | 1451233800 | 1483237800 | 1451233800 | U | ST411 | 1 | 0 | 1 | 1 |
| U115 | S200 | A300 | 1451233800 | 1483237800 | 1451233800 | AU | ST404 | 3 | 0 | 0 | 1 |
| U111 | S204 | A300 | 1451233800 | 1451233800 | 1451158200 | U | ST410 | 3 | 1 | 1 | 1 |
| U120 | S201 | A300 | 1483237800 | 1451233800 | 1451158200 | NULL | ST410 | 3 | 0 | 1 | 1 |
| U113 | S203 | NULL | 1451233800 | 1451233800 | 1451233800 | A | ST402 | 1 | 1 | 0 | 1 |
| U109 | S203 | A304 | 1451197800 | 1483237800 | 1451158200 | E | ST405 | 1 | 1 | 1 | 1 |
| U110 | S202 | A303 | 1483237800 | 1483237800 | 1451158200 | AU | ST402 | 2 | 1 | 0 | 1 |
| U100 | S200 | A301 | 1483237800 | 1483237800 | 1483237800 | AP | ST410 | 3 | 1 | 1 | 1 |
| U101 | S208 | A300 | 1451197800 | 1451158200 | 1451197800 | E | ST408 | 0 | 1 | 1 | 1 |
| U106 | S206 | A300 | 1483237800 | 1451233800 | 1451197800 | A | ST405 | 3 | 1 | 0 | 1 |
| U107 | S202 | A304 | 1483237800 | 1451158200 | 1451197800 | U | ST409 | 0 | 0 | 0 | 1 |
| U103 | S204 | A300 | 1451158200 | 1483237800 | 1451233800 | AU | ST411 | 2 | 1 | 0 | 1 |
| U103 | S202 | A300 | 1451233800 | 1451233800 | 1451233800 | A | ST415 | 2 | 1 | 1 | 1 |
| U113 | S203 | A303 | 1451197800 | 1451158200 | 1483237800 | U | ST408 | 2 | 0 | 0 | 1 |
| U113 | S204 | A301 | 1483237800 | 1483237800 | 1451233800 | E | ST415 | 3 | 0 | 1 | 1 |
| U114 | S207 | A303 | 1465130523 | 1465230523 | 1475130523 | A | ST415 | 3 | 1 | 0 | 1 |
| U107 | S202 | A303 | 1495130523 | 1465230523 | 1465230523 | U | ST415 | 0 | 1 | 1 | 1 |
| U100 | S204 | A302 | 1495130523 | 1475130523 | 1465130523 | AU | ST408 | 2 | 1 | 1 | 1 |
| U104 | S202 | A303 | 1465230523 | 1475130523 | 1465130523 | A | ST409 | 2 | 0 | 1 | 1 |
| U102 | S207 | A301 | 1465230523 | 1485130523 | 1465230523 | AU | ST403 | 3 | 1 | 1 | 1 |
| S203 | A302 | 1495130523 | 1475130523 | 1465230523 | E | ST400 | 0 | 0 | 1 | 1 |
| U106 | S202 | A302 | 1465230523 | 1465130523 | 1465130523 | AU | ST408 | 0 | 1 | 1 | 1 |
| U105 | S207 | A300 | 1465230523 | 1485130523 | 1465130523 | U | ST400 | 2 | 0 | 1 | 1 |
| U108 | S205 | A304 | 1465130523 | 1465130523 | 1475130523 | ST410 | 2 | 1 | 0 | 1 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

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

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

The screenshot shows a terminal window titled "acadgild@localhost:~/MusicProject/Scripts". The terminal contains a shell script with the following content:

```

#!/bin/bash

batchid=`cat /home/acadgild/MusicProject/logs/cycle.txt`
LOGFILE=/home/acadgild/MusicProject/logs/log_batch_$batchid
VALIDDIR=/home/acadgild/MusicProject/processed_dir/validated/batch_$batchid
INVALIDDIR=/home/acadgild/MusicProject/processed_dir/invalid/batch_$batchid

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

hive -hiveconf batchid=$batchid -f /home/acadgild/MusicProject/Scripts/data_enrichment.hql

if [ ! -d "$VALIDDIR" ]; then
    mkdir -p "$VALIDDIR"
else
    rm -rf "$VALIDDIR"
    mkdir -p "$VALIDDIR"
fi

if [ ! -d "$INVALIDDIR" ]; then
    mkdir -p "$INVALIDDIR"
else
    rm -rf "$INVALIDDIR"
    mkdir -p "$INVALIDDIR"
fi

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

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

echo "Copying data to HDFS processed_dir"

hadoop fs -rm -r /hadoop/processed_dir/
hadoop fs -mkdir -p /hadoop/processed_dir/
hadoop fs -cp /user/hive/warehouse/musicproject.db/enriched_data/batchid=$batchid/status=pass/* /hadoop/processed_dir/

echo "Deleting older valid and invalid records from local file system..." >> $LOGFILE
"data_enrichment.sh" 41L, 1370C

```

The terminal window has a title bar with icons for Applications, Places, System, and a browser. The status bar at the bottom shows "Thu Feb 15, 4:50 PM" and the user "acadgild". The bottom right corner of the terminal window shows "24,1" and "Top". The bottom of the window shows the command line prompt "acadgild@localhost:~/...".

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,

```

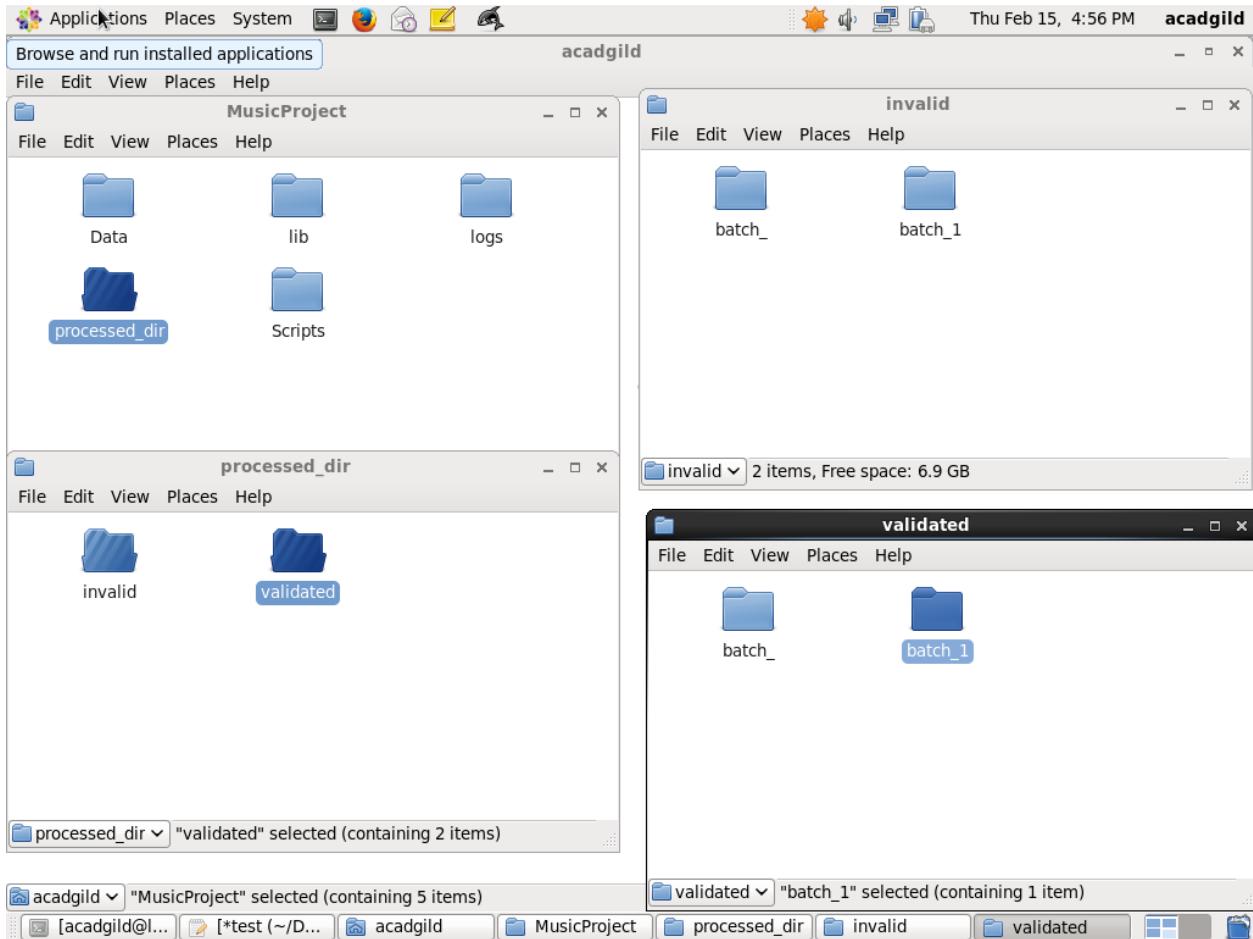
57,1 64%

After running the scripts below are the screenshots

```
Applications Places System Thu Feb 15, 4:49 PM acadgild
Change desktop appearance and behavior, get help, or log out calhost:~/MusicProject/Scripts
[acadgild@localhost Scripts]$ ./data_enrichment.sh

Logging initialized using configuration in jar:file:/usr/local/hive/lib/hive-common-0.14.0.jar!/hive-log4j.properties
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/hive-jdbc-0.14.0-standalone.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
OK
Time taken: 1.028 seconds
OK
Time taken: 0.358 seconds
OK
Time taken: 0.073 seconds
Query ID = acadgild_20180215164545_59319c30-c743-46c8-92df-f628e251cc18
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1518654102046_0026, Tracking URL = http://localhost:8088/proxy/application_1518654102046_0026/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job -kill job_1518654102046_0026
Hadoop job information for Stage-1: number of mappers: 3; number of reducers: 1
2018-02-15 16:45:34,840 Stage-1 map = 0%, reduce = 0%
2018-02-15 16:46:26,704 Stage-1 map = 67%, reduce = 0%, Cumulative CPU 3.78 sec
2018-02-15 16:46:29,405 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 8.98 sec
2018-02-15 16:46:39,375 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 10.85 sec
MapReduce Total cumulative CPU time: 10 seconds 850 msec
Ended Job = job_1518654102046_0026
Launching Job 2 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
```

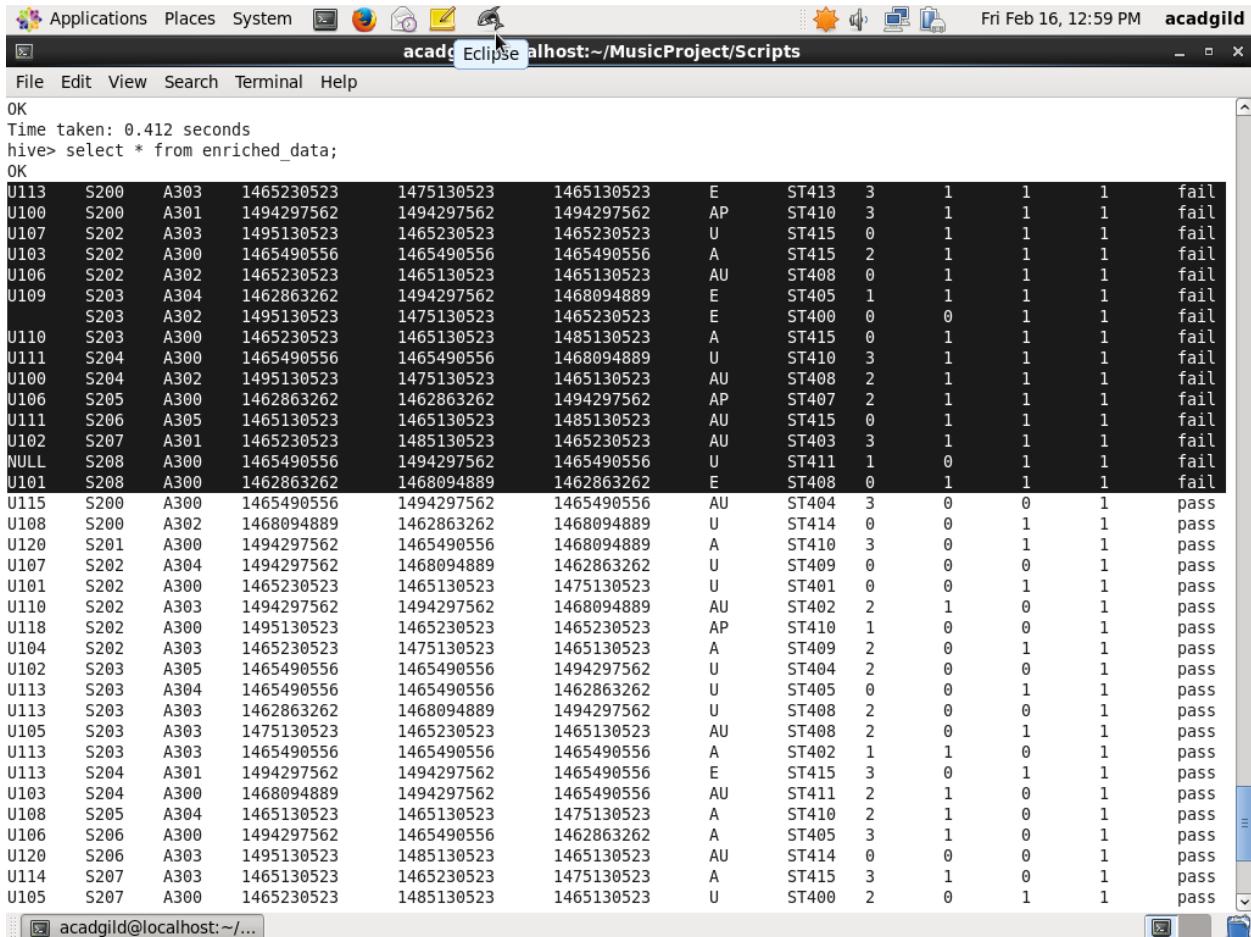
After scripts are validated and invalidated records are copied to local and only 7 days data is kept



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 classes 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.



The screenshot shows a terminal window titled "acadgild Eclipse localhost:~/MusicProject/Scripts". The window contains the following text:

```
OK
Time taken: 0.412 seconds
hive> select * from enriched_data;
OK
U113 S200 A303 1465230523 1475130523 1465130523 E ST413 3 1 1 1 1 fail
U100 S200 A301 1494297562 1494297562 1494297562 AP ST410 3 1 1 1 1 fail
U107 S202 A303 1495130523 1465230523 1465230523 U ST415 0 1 1 1 1 fail
U103 S202 A300 1465490556 1465490556 1465490556 A ST415 2 1 1 1 1 fail
U106 S202 A302 1465230523 1465130523 1465130523 AU ST408 0 1 1 1 1 fail
U109 S203 A304 1462863262 1494297562 1468094889 E ST405 1 1 1 1 1 fail
S203 A302 1495130523 1475130523 1465230523 E ST400 0 0 1 1 1 fail
U110 S203 A300 1465230523 1465130523 1485130523 A ST415 0 1 1 1 1 fail
U111 S204 A300 1465490556 1465490556 1468094889 U ST410 3 1 1 1 1 fail
U100 S204 A302 1495130523 1475130523 1465130523 AU ST408 2 1 1 1 1 fail
U106 S205 A300 1462863262 1462863262 1494297562 AP ST407 2 1 1 1 1 fail
U111 S206 A305 1465130523 1485130523 AU ST415 0 1 1 1 1 fail
U102 S207 A301 1465230523 1485130523 1465230523 AU ST403 3 1 1 1 1 fail
NULL S208 A300 1465490556 1494297562 1465490556 U ST411 1 0 1 1 1 fail
U101 S208 A300 1462863262 1468094889 1462863262 E ST408 0 1 1 1 1 fail
U115 S200 A300 1465490556 1494297562 1465490556 AU ST404 3 0 0 1 1 pass
U108 S200 A302 1468094889 1462863262 1468094889 U ST414 0 0 1 1 1 pass
U120 S201 A300 1494297562 1465490556 1468094889 A ST410 3 0 1 1 1 pass
U107 S202 A304 1494297562 1468094889 1462863262 U ST409 0 0 0 1 1 pass
U101 S202 A300 1465230523 1465130523 1475130523 U ST401 0 0 1 1 1 pass
U110 S202 A303 1494297562 1495130523 1468094889 AU ST402 2 1 0 1 1 pass
U118 S202 A300 1495130523 1465230523 1465230523 AP ST410 1 0 0 1 1 pass
U104 S202 A303 1465230523 1475130523 1465130523 A ST409 2 0 1 1 1 pass
U102 S203 A305 1465490556 1465490556 1494297562 U ST404 2 0 0 1 1 pass
U113 S203 A304 1465490556 1465490556 1462863262 U ST405 0 0 1 1 1 pass
U113 S203 A303 1462863262 1468094889 1494297562 U ST408 2 0 0 1 1 pass
U105 S203 A303 1475130523 1465230523 1465130523 AU ST408 2 0 1 1 1 pass
U113 S203 A303 1465490556 1465490556 1465490556 A ST402 1 1 0 1 1 pass
U113 S204 A301 1494297562 1494297562 1465490556 E ST415 3 0 1 1 1 pass
U103 S204 A300 1468094889 1494297562 1465490556 AU ST411 2 1 0 1 1 pass
U108 S205 A304 1465130523 1475130523 1465130523 A ST410 2 1 0 1 1 pass
U106 S206 A300 1494297562 1465490556 1462863262 A ST405 3 1 0 1 1 pass
U120 S206 A303 1495130523 1485130523 1465130523 AU ST414 0 0 0 1 1 pass
U114 S207 A303 1465130523 1465230523 1475130523 A ST415 3 1 0 1 1 pass
U105 S207 A300 1465230523 1485130523 1465130523 U ST400 2 0 1 1 1 pass
```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** workspace - Scala - Test/src/main/scala/com/musicanalysis/exec/MusicAnalysis.scala - Eclipse
- Menu Bar:** File Edit Refactor Navigate Search Project Scala Run Window Help
- Toolbar:** Standard Eclipse toolbar with icons for file operations, search, and project management.
- Quick Access:** A panel on the right containing icons for various tools and perspectives.
- Code Editor:** The main window displays the Scala code for `MusicAnalysis.scala`. The code uses Apache Hadoop and Spark libraries to analyze music data stored in Hive. It includes imports for `java.io.File`, `org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe`, `org.apache.spark.sql.SparkSession`, and `org.apache.spark.SparkException`. The `main` method sets up a `SparkSession` with `Data Analysis` as the application name, configures the warehouse location, and enables Hive support. It then performs a query to find the top 10 stations based on unique user likes. The code is annotated with comments explaining its purpose.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** workspace - Scala - Test/src/main/scala/com/musicanalysis/exec/MusicAnalysis.scala - Eclipse
- Menu Bar:** File Edit Refactor Navigate Search Project Scala Run Window Help
- Toolbar:** Includes icons for New, Open, Save, Cut, Copy, Paste, Find, Replace, Undo, Redo, and others.
- Quick Access:** A panel on the right containing icons for Java, Scala, Python, and other languages.
- Code Editor:** The main window displays Scala code for `MusicAnalysis.scala`. The code includes operations like `spark.sql` for inserting into tables and creating temporary tables, as well as `SELECT`, `FROM`, and `WHERE` clauses. It also handles partitioning by batch ID and ordering by song duration.
- Status Bar:** Shows "Writable" and "Smart Insert" status indicators, along with the current time "16:48".

workspace - Scala - Test/src/main/scala/com/musicanalysis/exec/MusicAnalysis.scala - Eclipse

File Edit Refactor Navigate Search Project Scala Run Window Help

Quick Access

```
103
104
105 //Determine top 10 connected artists.
106 //Connected artists are those whose songs are most listened by the unique users who follow them.
107
108 val create_hive_table_top_10_connected_artists = spark.sql("""CREATE TABLE IF NOT EXISTS musicproject.connected_artists
109 (
110     artist_id STRING,
111     unique_followers INT
112 )
113 PARTITIONED BY (batchid INT)
114 ROW FORMAT DELIMITED
115 FIELDS TERMINATED BY ','
116 STORED AS TEXTFILE""")
117
118
119 val insert_into_top_10_connected_artists = spark.sql(s"""INSERT OVERWRITE TABLE musicproject.connected_artists
120 PARTITION (batchid=$batchId)
121 SELECT
122     f.artist_id,
123     COUNT(DISTINCT f.user_id) AS unique_followers
124 FROM musicproject.enriched_data f join (select user_id, artist_id["artist_1"] as artist_id from musicproject.user_artist_map union all select user_id, artist_id["art
125 on i.user_id=f.user_id
126 and (f.artist_id=i.artist_id )
127 WHERE
128     status='pass'
129     AND (batchid=$batchId)
130     GROUP BY f.artist_id
131     ORDER BY unique_followers desc
132     LIMIT 10""")
133
134
135
136 //Determine top 10 songs who have generated the maximum revenue.
137 //NOTE: Royalty applies to a song only if it was liked or was completed successfully or both.
138
139 val create_hive_table_top_10_songs_maxrevenue = spark.sql("""CREATE TABLE IF NOT EXISTS musicproject.top_10_songs_maxrevenue
140 (
141     ... 13 columns
```

The screenshot shows the Eclipse IDE interface with the title bar "workspace - Scala - Test/src/main/scala/com/musicanalysis/exec/MusicAnalysis.scala - Eclipse". The menu bar includes File, Edit, Refactor, Navigate, Search, Project, Scala, Run, Window, Help. The toolbar has various icons for file operations like Open, Save, Find, and Run. The Quick Access panel is visible on the right. The code editor displays Scala code for music analysis, specifically for creating Hive tables and inserting data into them. The code includes imports for spark.sql, DataFrame, and Row. It defines two main functions: one for creating a top 10 songs table based on maximum revenue and another for creating a top 10 unsubscribed users table based on total duration played.

```
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

130
131
132
133 //Determine top 10 songs who have generated the maximum revenue.
134 //NOTE: Royalty applies to a song only if it was liked or was completed successfully or both.
135
136 val create_hive_table_top_10_songs_maxrevenue = spark.sql("""CREATE TABLE IF NOT EXISTS musicproject.top_10_songs_maxrevenue
137   (
138     song_id STRING,
139     total_duration_played_in_minutes DOUBLE
140   )
141   PARTITIONED BY (batchid INT)
142   ROW FORMAT DELIMITED
143   FIELDS TERMINATED BY ','
144   STORED AS TEXTFILE""")
145
146
147 val insert_into_top_10_songs_maxrevenue = spark.sql(s"""INSERT OVERWRITE TABLE musicproject.top_10_songs_maxrevenue
148   PARTITION (batchid=$batchId)
149   SELECT
150     song_id,
151     SUM(cast(end_ts as BIGINT)-cast(start_ts as BIGINT))/60 AS total_duration_played_in_minutes
152   FROM musicproject.enriched_data
153   WHERE status='pass'
154   AND (batchid=$batchId)
155   AND (like=1 OR song_end_type=0)
156   GROUP BY song_id
157   ORDER BY total_duration_played_in_minutes desc
158   LIMIT 10""")
159
160
161
162 //Determine top 10 unsubscribed users who listened to the songs for the longest duration.
163
164 val create_hive_table_top_10_unsubscribed_users = spark.sql("""CREATE TABLE IF NOT EXISTS musicproject.top_10_unsubscribed_users
165   (
166     user_id STRING,
167     total_duration_played DOUBLE
168   )
169   PARTITIONED BY (batchid INT)
170   ROW FORMAT DELIMITED
171   FIELDS TERMINATED BY ','
172   STORED AS TEXTFILE""")
```

The screenshot shows the Eclipse IDE interface with the title bar "workspace - Scala - Test/src/main/scala/com/musicanalysis/exec/MusicAnalysis.scala - Eclipse". The code editor displays the following Scala code:

```
161 //Determine top 10 unsubscribed users who listened to the songs for the longest duration.
162
163 val create_hive_table_top_10_unsubscribed_users = spark.sql("""CREATE TABLE IF NOT EXISTS musicproject.top_10_unsubscribed_users
164 (
165     user_id STRING,
166     total_duration_played DOUBLE
167 )
168 PARTITIONED BY (batchid INT)
169 ROW FORMAT DELIMITED
170 FIELDS TERMINATED BY ','
171 STORED AS TEXTFILE""")
172
173
174 val insert_into_unsubscribed_users = spark.sql(s"""INSERT OVERWRITE TABLE musicproject.top_10_unsubscribed_users
175 PARTITION (batchid$batchId)
176 SELECT
177     user_id,
178     SUM(total_duration_in_minutes) as total_duration_played
179     FROM musicproject.user_song_duration
180     WHERE user_type='unsubscribed'
181     AND total_duration_in_minutes>=0
182     AND (batchid$batchId)
183     GROUP BY user_id
184     ORDER BY total_duration_played desc
185     LIMIT 10""")
186
187
188 catch {
189     case ex2: Exception => ex2.printStackTrace()
190     throw new SparkException("Sql Exception")
191 }
192
193
194 finally{
195     spark.stop()
196 }
197
198
```

SBT for building dependency.

The screenshot shows the Eclipse IDE interface with the title bar "workspace - Scala - Test/build.sbt - Eclipse". The code editor displays the following SBT build configuration:

```
1 name := "MusicAnalysis"
2
3 version := "0.1"
4
5 scalaVersion := "2.11.8"
6 libraryDependencies ++= Seq(
7   "org.apache.spark" % "spark-core_2.11" % "2.1.1",
8   "org.apache.spark" % "spark-sql_2.11" % "2.1.1",
9   "org.apache.spark" % "spark-hive_2.11" % "2.1.1",
10  "org.apache.spark" % "spark-yarn_2.11" % "2.1.1",
11  )
12
13 scalacOptions ++= Seq(
14   "-deprecation",
15   "-feature",
16   //other options
17 )
```

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

Spark analysis output

```
[acadgild@localhost logs]$ cd ..Scripts/
[acadgild@localhost Scripts]$ ./data_analysis.sh
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/spark/jars/slf4j-log4j12-1.7.16.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://Logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
18/02/17 09:31:50 INFO SparkContext: Running Spark version 2.0.0
18/02/17 09:31:51 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
18/02/17 09:31:51 WARN Utils: Your hostname, localhost.localdomain resolves to a loopback address: 127.0.0.1; using 192.168.0.121 instead (d'oh interface eth0)
18/02/17 09:31:51 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
18/02/17 09:31:51 INFO SecurityManager: Changing view acls to: acadgild
18/02/17 09:31:51 INFO SecurityManager: Changing modify acls to: acadgild
18/02/17 09:31:51 INFO SecurityManager: Changing view acls groups to:
18/02/17 09:31:51 INFO SecurityManager: Changing modify acls groups to:
18/02/17 09:31:51 INFO SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(acadgild); groups with view permissions: Set(); users with modify permissions: Set(acadgild); groups with modify permissions: Set()
18/02/17 09:31:52 INFO Utils: Successfully started service 'sparkDriver' on port 52259.
18/02/17 09:31:52 INFO SparkEnv: Registering MapOutputTracker
18/02/17 09:31:52 INFO SparkEnv: Registering BlockManagerMaster
18/02/17 09:31:52 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-d313d16e-e76d-464b-a268-657c3fac7312
18/02/17 09:31:52 INFO MemoryStore: MemoryStore started with capacity 413.9 MB
18/02/17 09:31:52 INFO SparkEnv: Registering OutputCommitCoordinator
18/02/17 09:31:54 INFO Utils: Successfully started service 'SparkUI' on port 4040.
18/02/17 09:31:54 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://192.168.0.121:4040
18/02/17 09:31:54 INFO SparkContext: Added JAR file:/home/acadgild/MusicProject/spark/musicanalysis_2.11-0.1.jar at spark://192.168.0.121:52259/jars/musicanalysis_2.11-0.1.jar with timestamp 1518840114104
18/02/17 09:31:54 INFO Executor: Starting executor ID driver on host localhost
18/02/17 09:31:54 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 50536.
18/02/17 09:31:54 INFO NettyBlockTransferService: Server created on 192.168.0.121:50536
18/02/17 09:31:54 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 192.168.0.121, 50536)
18/02/17 09:31:54 INFO BlockManagerMasterEndpoint: Registering block manager 192.168.0.121:50536 with 413.9 MB RAM, BlockManagerId(driver, 192.168.0.121, 50536)
```

```
2 192.168.0.121 * 4 192.168.0.121 cmd=cmd>
[acadgild@localhost ~]$ ./data_analysis.sh
18/02/17 09:32:24 INFO audit: ugi=acadgild ip=unknown-ip-addr cmd=get_database: musicproject
18/02/17 09:32:24 INFO HiveMetaStore: 0: get_database: musicproject
18/02/17 09:32:24 INFO audit: ugi=acadgild ip=unknown-ip-addr cmd=create_table: top_10_stations
18/02/17 09:32:25 INFO HiveMetaStore: 0: create_table Table(tableName=top_10_stations, dbName:musicproject, owner:acadgild, createTime:1518840144, lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:station_id, type:string, comment:null)], FieldsSchema(total_locations:1), numBuckets:1, serdeInfo:org.apache.hadoop.mapred.TextInputFormat, outputFormat:org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat, compressed:false, bucketCols:[], sortCols:[], parameters:{}, skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLocationMaps:{}), partitionKeys:[FieldSchema(name:batchid, type:int, comment:null)], parameters:{}, viewOriginalText:null, viewExpandedText:null, tableType:MANAGED_TABLE, privileges:PrincipalPrivilegeSet(userPrivileges:{}, groupPrivileges:null, rolePrivileges:null)})
18/02/17 09:32:25 INFO audit: ugi=acadgild ip=unknown-ip-addr cmd=create_table: Table(tableName:top_10_stations, dbName:musicproject, owner:acadgild, createTime:1518840144, lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:station_id, type:string, comment:null)], FieldsSchema(total_locations:1), numBuckets:1, serdeInfo:org.apache.hadoop.mapred.TextInputFormat, outputFormat:org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat, compressed:false, bucketCols:[], sortCols:[], parameters:{}, skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLocationMaps:{}), partitionKeys:[FieldSchema(name:batchid, type:int, comment:null)], parameters:{}, viewOriginalText:null, viewExpandedText:null, tableType:MANAGED_TABLE, privileges:PrincipalPrivilegeSet(userPrivileges:{}, groupPrivileges:null, rolePrivileges:null)))
18/02/17 09:32:25 INFO SparkSqlParser: Parsing command: INSERT OVERWRITE TABLE musicproject.top_10_stations
    PARTITION (batchid=1)
        SELECT
            station_id,
            COUNT(DISTINCT user_id,song_id) AS total_songs_played_liked_by_unique_user
        FROM musicproject.enriched_data
        WHERE status='pass'
        AND (batchid=1)
        AND like=1
        GROUP BY station_id
        ORDER BY total_songs_played_liked_by_unique_user DESC
        LIMIT 10
18/02/17 09:32:25 INFO HiveMetaStore: 0: get_database: musicproject
18/02/17 09:32:25 INFO audit: ugi=acadgild ip=unknown-ip-addr cmd=get_database: musicproject
18/02/17 09:32:25 INFO Persistence: Request to load fields "comment,name,type" of class org.apache.hadoop.hive.metastore.model.MFields
18/02/17 09:32:25 INFO Persistence: Request to load fields "comment,name,type" of class org.apache.hadoop.hive.metastore.model.MFields
18/02/17 09:32:25 INFO Persistence: Request to load fields "comment,name,type" of class org.apache.hadoop.hive.metastore.model.MFields
18/02/17 09:32:25 INFO Persistence: Request to load fields "comment,name,type" of class org.apache.hadoop.hive.metastore.model.MFields
```

```
2. 192.168.0.121 x 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  
ime:1518840166, lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:user_type, type:string, comment:null), Fields  
chema(name:total_duration_played, type:double, comment:null)], location:null, inputFormat:org.apache.hadoop.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:,, serialization.format:}, bucketCols:[], sortCols:[], skewedInfo:  
[SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLocationMaps:{})]), partitionKeys:[FieldSchema(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:musicproj  
ect, 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.  
adop.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  
ation.format:}, bucketCols:[], sortCols:[], skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLocati  
onMaps:{}), 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.MFieldS  
chema 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.MFieldS  
chema 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.MFieldS  
chema but object is embedded, so ignored  
  
(term by subscribing to the professional edition here: http://mobaxterm.mobatek.net
```

```
2. 192.168.0.121 x 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, createT  
ime:1518840169, lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:artist_id, type:string, comment:null), Fields  
chema(name:unique_followers, type:int, comment:null)], location:null, inputFormat:org.apache.hadoop.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:,, serialization.format:}, bucketCols:[], sortCols:[], skewedInfo:  
[SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLocationMaps:{})], partitionKeys:[FieldSchema(name:batchid, type:int, comment:  
null)], parameters:{}, viewOriginalText:null, viewExpandedText:null, tableType:MANAGED_TABLE, privileges:PrincipalPrivilegeSet(userPrivileges:  
{}), groupPrivileges:null, rolePrivileges:null)  
18/02/17 09:32:49 INFO audit: ugi=acadgild ip=unknown-ip-addr cmd=create_table: Table(tableName:connected_artists, dbName:musicproj  
ect, owner:acadgild, createTime:1518840169, lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:artist_id,  
type:string, comment:null), Fieldschema(name:unique_followers, type:int, comment:null)], location:null, inputFormat:org.apache.hadoop.  
mapred.TextInputFormat, outputFormat:org.apache.hadoop.hive ql.io.HiveIgnoreKeyTextOutputFormat, compressed:false, numBuckets:-1, serdeIn  
fo:SerdeInfo(name:null, serializationLib:org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, parameters:{field.delim:,, serializati  
on.format:}, bucketCols:[], sortCols:[], skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLocati  
onMaps:{}), partitionKeys:[FieldSchema(name:batchid, type:int, comment:null)], parameters:{}, viewOriginalText:null, viewExpandedText:  
null, tableType:MANAGED_TABLE, privileges:PrincipalPrivilegeSet(userPrivileges:{}), groupPrivileges:null, rolePrivileges:  
null)  
18/02/17 09:32:49 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.metastore.model.MFieldS  
chema 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  
chema 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  
chema 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  
chema but object is embedded, so ignored
```

```
2. 192.168.0.121 x 4. 192.168.0.121
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
18/02/17 09:32:52 INFO HiveMetaStore: 0: get_database: musicproject
18/02/17 09:32:52 INFO audit: ugi=acadgild ip=unknown-ip-addr cmd=get_database: musicproject
18/02/17 09:32:52 INFO HiveMetaStore: 0: get_database: musicproject
18/02/17 09:32:52 INFO audit: ugi=acadgild ip=unknown-ip-addr cmd=get_database: musicproject
18/02/17 09:32:52 INFO HiveMetaStore: 0: create_table: Table(tableName=top_10_songs_maxrevenue, dbName:musicproject, owner:acadgild, createTime:1518840172, lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:song_id, type:string, comment:null)], fieldschema(name:total_duration_played_in_minutes, type:double, comment:null), location:null, inputFormat:org.apache.hadoop.mapred.TextInputFormat, outputFormat:org.apache.hadoop.hive.io.HiveIgnoreKeyTextOutputFormat, compressed:false, numBuckets:1, serdeInfo:SerdeInfo(name:null, serializationlib:org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, parameters:{field.delim=, serialization.format=}), bucketCols:[], sortCols:[], parameters:{}, skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLocationMaps:{}), partitionKeys:[FieldSchema(name:batchid, type:int, comment:null)], parameters:{}, viewOriginalText:null, viewExpandedText:null, tableType:MANAGED_TABLE, privileges:PrincipalPrivilegeSet(userPrivileges:{}, groupPrivileges:null, rolePrivileges:null)})
18/02/17 09:32:52 INFO audit: ugi=acadgild ip=unknown-ip-addr cmd=create_table: Table(tableName=top_10_songs_maxrevenue, dbName:musicproject, owner:acadgild, createTime:1518840172, lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:song_id, type:string, comment:null)], fieldschema(name:total_duration_played_in_minutes, type:double, comment:null), location:null, inputFormat:org.apache.hadoop.mapred.TextInputFormat, outputFormat:org.apache.hadoop.hive.io.HiveIgnoreKeyTextOutputFormat, compressed:false, numBuckets:1, serdeInfo:SerdeInfo(name:null, serializationlib:org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, parameters:{field.delim=, serialization.format=}), bucketCols:[], sortCols:[], parameters:{}, skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLocationMaps:{}), partitionKeys:[FieldSchema(name:batchid, type:int, comment:null)], parameters:{}, viewOriginalText:null, viewExpandedText:null, tableType:MANAGED_TABLE, privileges:PrincipalPrivilegeSet(userPrivileges:{}, groupPrivileges:null, rolePrivileges:null)})
18/02/17 09:32:52 INFO SparkSqlParser: Parsing command: INSERT OVERWRITE TABLE musicproject.top_10_songs_maxrevenue
PARTITION (batchid=1)
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=1)
AND (like1 OR song_end_type=0)
GROUP BY song_id
ORDER BY total_duration_played_in_minutes desc
LIMIT 10
18/02/17 09:32:52 INFO HiveMetaStore: 0: get_database: musicproject
18/02/17 09:32:52 INFO audit: ugi=acadgild ip=unknown-ip-addr cmd=get_database: musicproject
18/02/17 09:32:52 INFO Persistence: Request to load fields "comment,name,type" of class org.apache.hadoop.hive.metastore.model.MFieldS
```

```
2. 192.168.0.121 x 4. 192.168.0.121
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
18/02/17 09:32:52 INFO HiveMetaStore: 0: get_database: musicproject
18/02/17 09:32:52 INFO audit: ugi=acadgild ip=unknown-ip-addr cmd=get_database: musicproject
18/02/17 09:32:52 INFO HiveMetaStore: 0: get_database: musicproject
18/02/17 09:32:52 INFO audit: ugi=acadgild ip=unknown-ip-addr cmd=get_database: musicproject
18/02/17 09:32:52 INFO HiveMetaStore: 0: create_table: Table(tableName=top_10_songs_maxrevenue, dbName:musicproject, owner:acadgild, createTime:1518840172, lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:song_id, type:string, comment:null)], fieldschema(name:total_duration_played_in_minutes, type:double, comment:null), location:null, inputFormat:org.apache.hadoop.mapred.TextInputFormat, outputFormat:org.apache.hadoop.hive.io.HiveIgnoreKeyTextOutputFormat, compressed:false, numBuckets:1, serdeInfo:SerdeInfo(name:null, serializationlib:org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, parameters:{field.delim=, serialization.format=}), bucketCols:[], sortCols:[], parameters:{}, skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLocationMaps:{}), partitionKeys:[FieldSchema(name:batchid, type:int, comment:null)], parameters:{}, viewOriginalText:null, viewExpandedText:null, tableType:MANAGED_TABLE, privileges:PrincipalPrivilegeSet(userPrivileges:{}, groupPrivileges:null, rolePrivileges:null)})
18/02/17 09:32:52 INFO audit: ugi=acadgild ip=unknown-ip-addr cmd=create_table: Table(tableName=top_10_songs_maxrevenue, dbName:musicproject, owner:acadgild, createTime:1518840172, lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:song_id, type:string, comment:null)], fieldschema(name:total_duration_played_in_minutes, type:double, comment:null), location:null, inputFormat:org.apache.hadoop.mapred.TextInputFormat, outputFormat:org.apache.hadoop.hive.io.HiveIgnoreKeyTextOutputFormat, compressed:false, numBuckets:1, serdeInfo:SerdeInfo(name:null, serializationlib:org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, parameters:{field.delim=, serialization.format=}), bucketCols:[], sortCols:[], parameters:{}, skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLocationMaps:{}), partitionKeys:[FieldSchema(name:batchid, type:int, comment:null)], parameters:{}, viewOriginalText:null, viewExpandedText:null, tableType:MANAGED_TABLE, privileges:PrincipalPrivilegeSet(userPrivileges:{}, groupPrivileges:null, rolePrivileges:null)})
18/02/17 09:32:52 INFO SparkSqlParser: Parsing command: INSERT OVERWRITE TABLE musicproject.top_10_songs_maxrevenue
PARTITION (batchid=1)
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=1)
AND (like1 OR song_end_type=0)
GROUP BY song_id
ORDER BY total_duration_played_in_minutes desc
LIMIT 10
18/02/17 09:32:52 INFO HiveMetaStore: 0: get_database: musicproject
18/02/17 09:32:52 INFO audit: ugi=acadgild ip=unknown-ip-addr cmd=get_database: musicproject
18/02/17 09:32:52 INFO Persistence: Request to load fields "comment,name,type" of class org.apache.hadoop.hive.metastore.model.MFieldS
```

Xterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

We can now see the results in Hive.

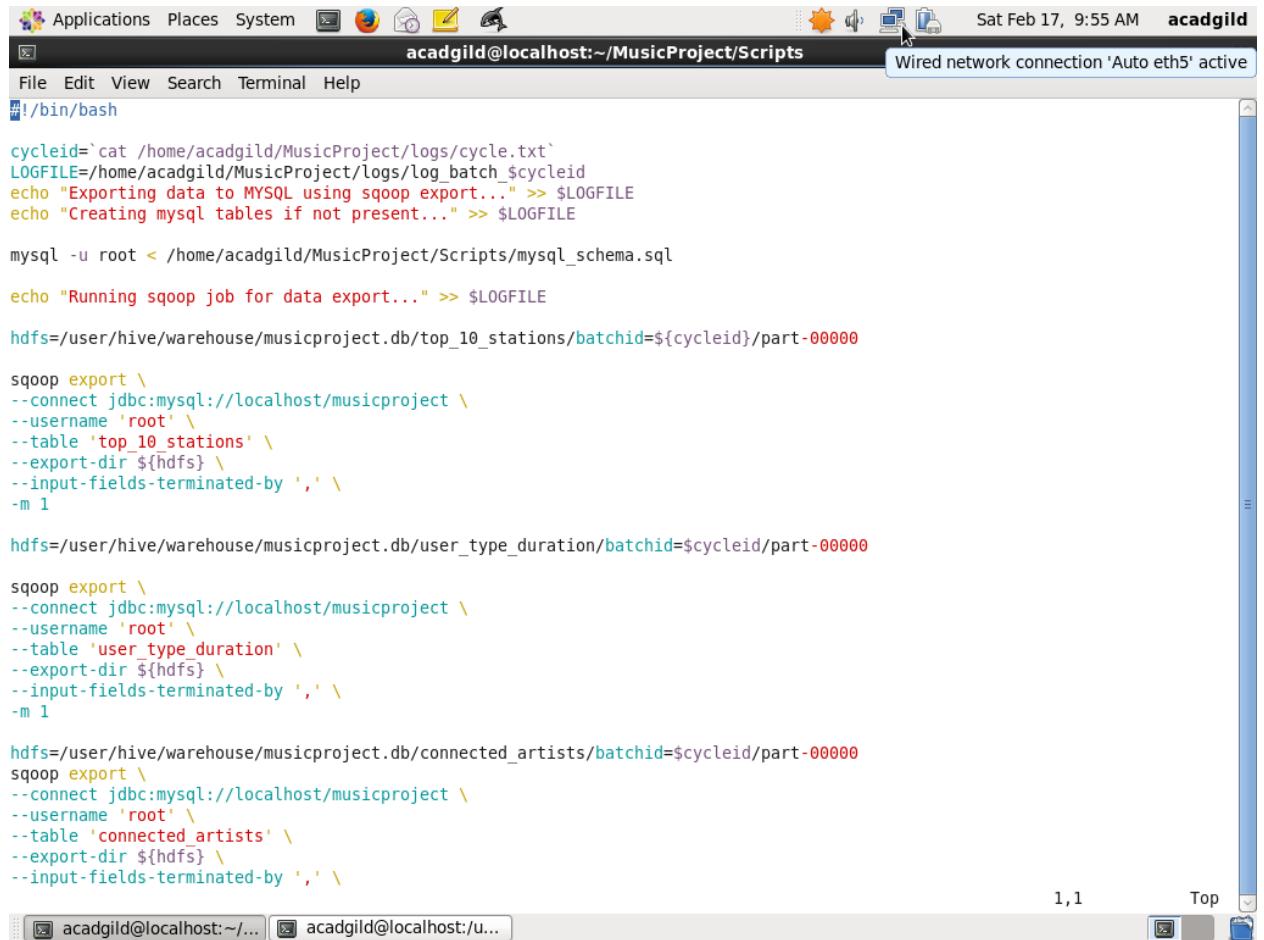
```
hive> use musicproject
      > ;
OK
Time taken: 1.101 seconds
hive> show tables;
OK
connected_artists
enriched_data
merged_data
song_artist_map
station_geo_map
subscribed_users
temp_enriched_data
top_10_songs_maxrevenue
top_10_stations
top_10_unsubscribed_users
user_artist_map
user_song_duration
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
ST409    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    166666.6666666666   1
U102    subscribed    480116.766666666666   1
U103    unsubscribed -480116.766666666666   1
U104    subscribed    -166666.6666666666   1
U105    subscribed    -1666.666666666667   1
U106    unsubscribed -333333.3333333333   1
U108    subscribed    -43788.2333333333   1
U107    unsubscribed  87193.78333333334   1
U107    unsubscribed  0.0     1
U108    subscribed    87193.78333333334   1
U109    subscribed    166666.6666666666   1
U110    unsubscribed -436711.2166666667   1
U113    subscribed    -43788.2333333333   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    -166666.6666666666   1
U118    subscribed    0.0     1
U118    subscribed    1666.666666666667   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    166666.6666666666   1
S207    165000.0     1
S200    87193.78333333334   1
S210    0.0     1
S203    -43788.2333333333   1
S202    -357238.3333333333   1
S206    -377121.5666666665   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
A303    8      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



```
acadgild@localhost:~/MusicProject/Scripts$ ./data_export.sh
acadgild@localhost:~/MusicProject/Scripts$ cat /home/acadgild/MusicProject/logs/cycle.txt
LOGFILE=/home/acadgild/MusicProject/logs/log_batch_${cycleid}
echo "Exporting data to MYSQL using sqoop export..." >> $LOGFILE
echo "Creating mysql tables if not present..." >> $LOGFILE

mysql -u root < /home/acadgild/MusicProject/Scripts/mysql_schema.sql

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

hdfs=/user/hive/warehouse/musicproject.db/top_10_stations/batchid=${cycleid}/part-00000

sqoop export \
--connect jdbc:mysql://localhost/musicproject \
--username 'root' \
--table 'top_10_stations' \
--export-dir ${hdfs} \
--input-fields-terminated-by ',' \
-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
```

```
Applications Places System Firefox Nautilus Konsole Terminal Help Sat Feb 17, 9:55 AM acadgild
Browse and run installed applications acadgild@localhost:~/MusicProject/Scripts - X
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.name
2018-02-17 10:01:49,904 INFO [main] Configuration.deprecation: mapred.cache.files.timestamps is deprecated. Instead, use mapreduce.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 mapreduce.job.inputformat.class
2018-02-17 10:01:49,912 INFO [main] Configuration.deprecation: mapreduce.outputformat.class is deprecated. Instead, use mapreduce.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 mapreduce.map.output.value.class
2018-02-17 10:01:49,914 INFO [main] Configuration.deprecation: mapred.mapoutput.key.class is deprecated. Instead, use mapreduce.map.output.key.class
2018-02-17 10:01:49,914 INFO [main] Configuration.deprecation: mapred.job.classpath.files is deprecated. Instead, use mapreduce.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 mapreduce.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 ResourceManager 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> █

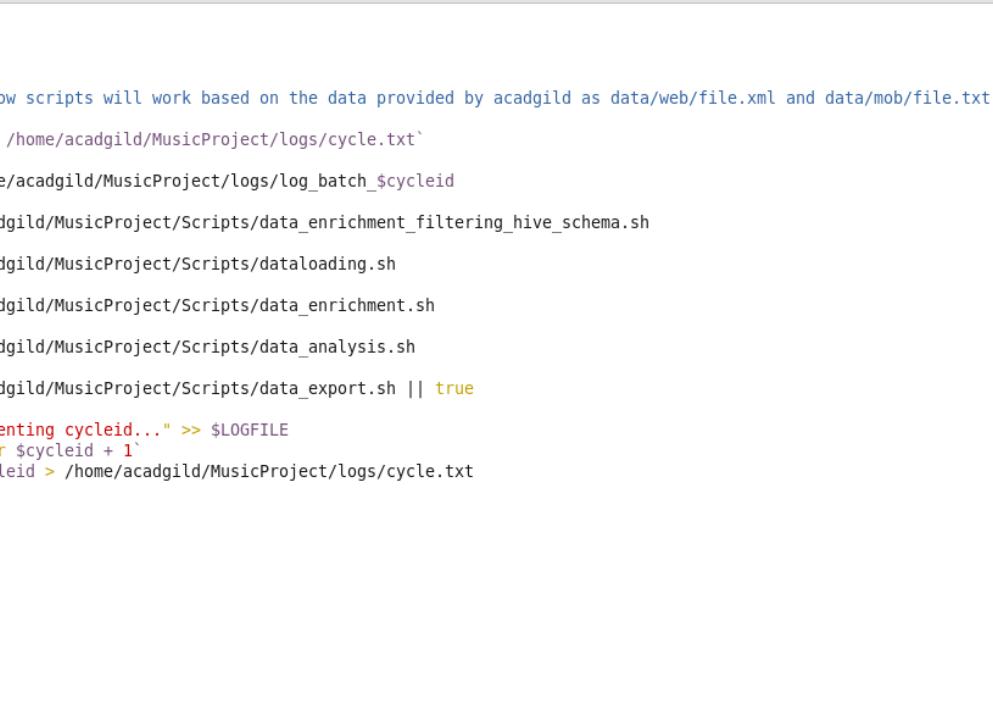
- Determine top 10 songs who have generated the maximum revenue. Royalty applies to a song only if it was liked or was completed successfully or both.

```
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     | -357238.3333333333 |
| 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.



The screenshot shows a Linux desktop environment with an open terminal window titled "acadgild@localhost:~/MusicProject/Scripts". The terminal contains a shell script for a music project. The script starts with "#!/bin/bash" and includes several commands to process data files, update a log file, and increment a cycle ID. The terminal window is part of a desktop interface with icons for Applications, Places, System, and various system status indicators.

```
##!/bin/bash

set -e

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

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

LOGFILE=/home/acadgild/MusicProject/logs/log_batch_$cycleid

sh /home/acadgild/MusicProject/Scripts/data_enrichment_filtering_hive_schema.sh

sh /home/acadgild/MusicProject/Scripts/dataloading.sh

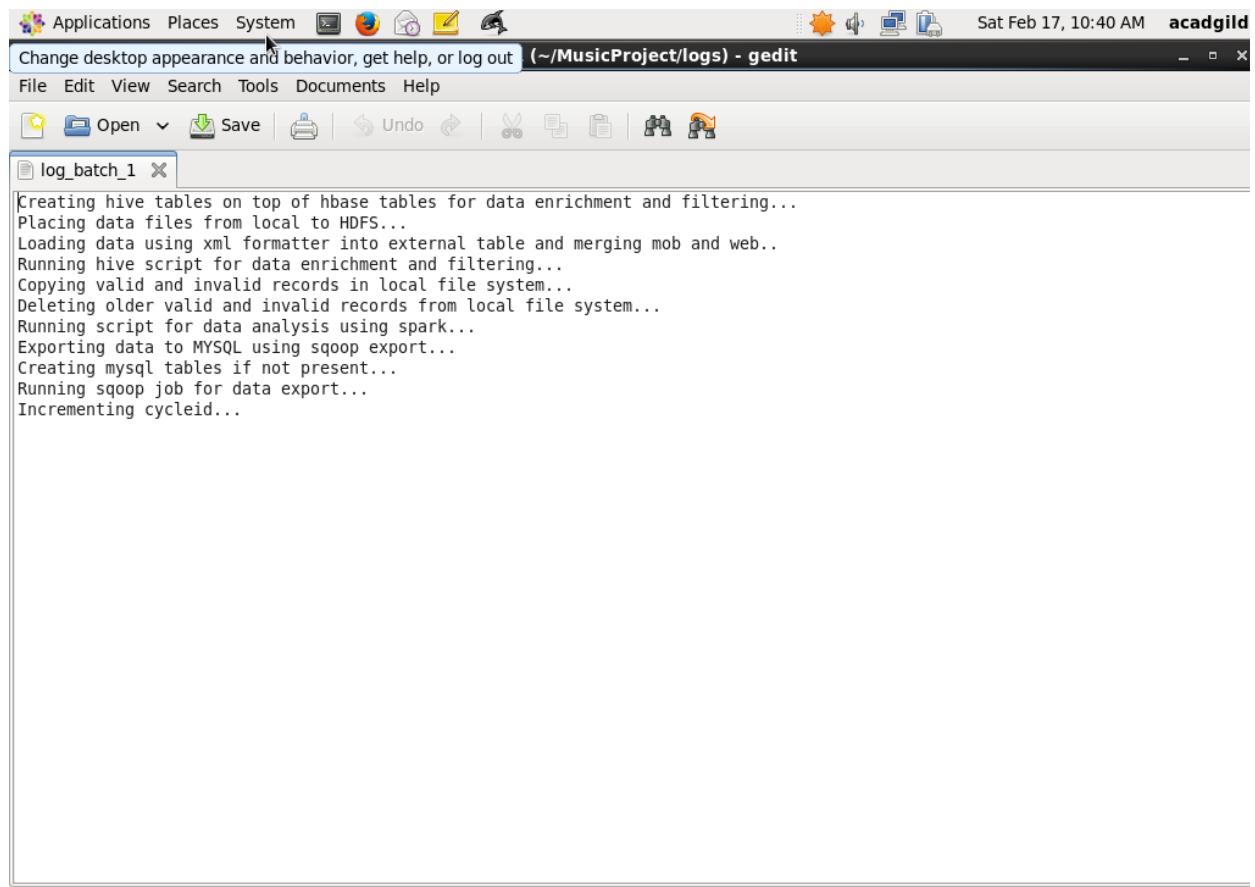
sh /home/acadgild/MusicProject/Scripts/data_enrichment.sh

sh /home/acadgild/MusicProject/Scripts/data_analysis.sh

sh /home/acadgild/MusicProject/Scripts/data_export.sh || true

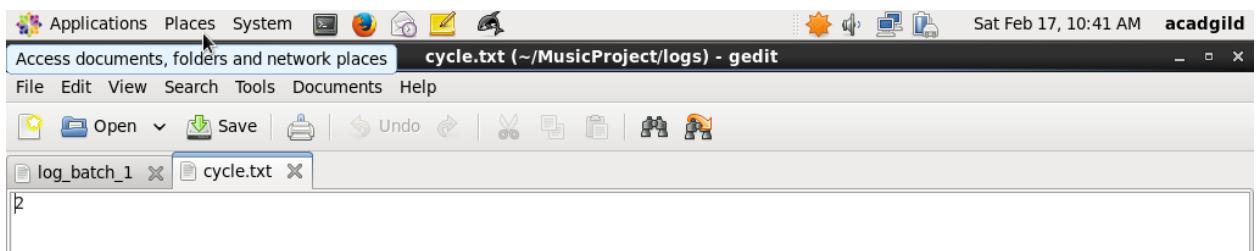
echo "Incrementing cycleid..." >> $LOGFILE
cycleid=`expr $cycleid + 1`
echo -n $cycleid > /home/acadgild/MusicProject/logs/cycle.txt
~
```

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



Creating hive tables on top of hbase tables for data enrichment and filtering...
Placing data files from local to HDFS...
Loading data using xml formatter into external table and merging mob and web..
Running hive script for data enrichment and filtering...
Copying valid and invalid records in local file system...
Deleting older valid and invalid records from local file system...
Running script for data analysis using spark...
Exporting data to MYSQL using sqoop export...
Creating mysql tables if not present...
Running sqoop job for data export...
Incrementing cycleid...

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



Access documents, folders and network places cycle.txt (~/MusicProject/logs) - gedit
File Edit View Search Tools Documents Help
Open Save Undo |
log_batch_1 cycle.txt
2

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/MusicAnalysisSchedule.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.