Music Data Analysis

1.0: Project Overview

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

2.0: Data Files

Data set consists of user information , song details like song_id , Artist_id, and number of likes and dislikes received for each song.

2.1: Project Description

Column Name/Field Name Column	Description/Field Description
User_id	Unique identifier of every user
Song_id	Unique identifier of every song
Artist_id	Unique identifier of the lead artist of the song
Timestamp	Timestamp when the record was generated
Start_ts	Start timestamp when the song started to play
End_ts	End timestamp when the song was stopped
Geo_cd	Can be 'A' for USA region, 'AP' for asia pacific region,'J' for Japan region, 'E' for europe and 'AU' for australia region
Station_id	Unique identifier of the station from where the song was played
Song_end_type	How the song was terminated. 0 means completed successfully 1 means song was skipped 2 means song was paused 3 means other type of failure like device issue, network error etc.
Like	0 means song was not likedsong was played 1 means song was liked
Dislike	0 means song was not disliked 1 means song was disliked

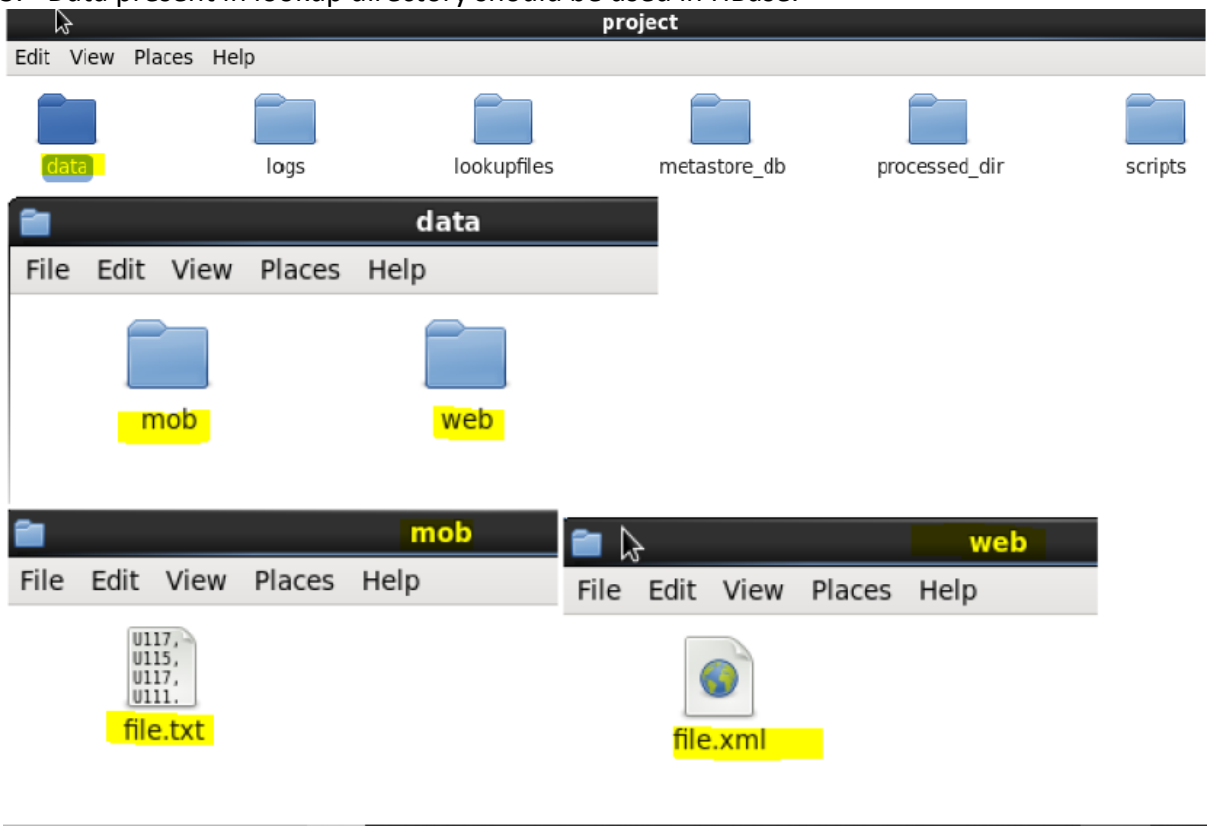
2.2 LookUp Tables:

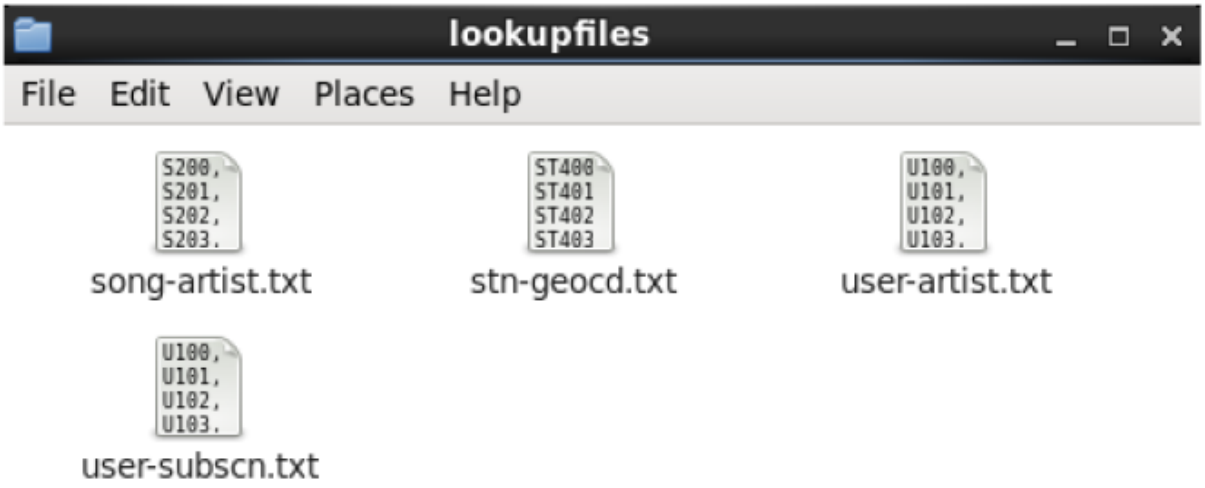
There are some existing look up tables present in NoSQL databases. They play an important role in data enrichment and analysis.

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

3.0 Data Ingestion and Initial Validaion

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





4.0 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)**
Geo_cd	Station_id	Station_Geo_Map
Artist_id	Song_id	Song_Artist_Map

5.0 Data Analysis

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

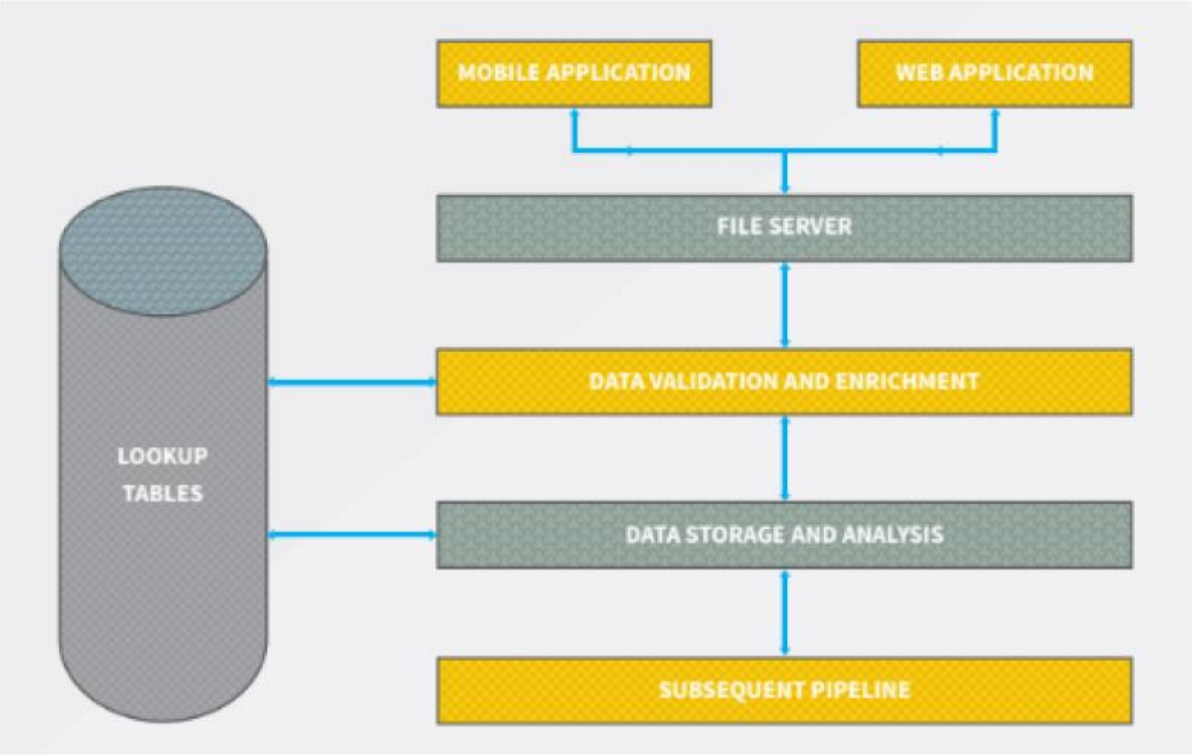
1. Determine top 10 station_id(s) where maximum number of songs were played, which were liked by unique users
2. Determine total duration of songs played by each type of user, where type of user can be 'subscribed' or ' unsubscribed'. An unsubscribed user is the one whose record is either not present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

5.1 Challenges and Optimizations

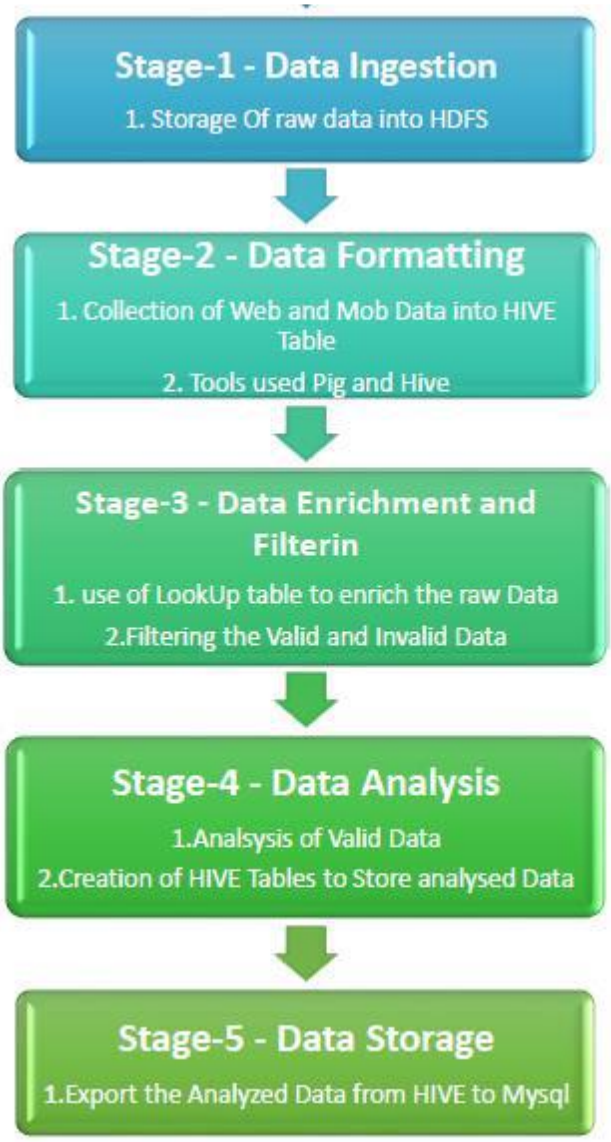
1. LookUp tables are in NoSQL databases. Integrate them with the actual data flow.
2. Try to make joins as less expensive as possible.
3. Data Cleaning, Validation, Enrichment, Analysis and Post Analysis have to be automated. Try using schedulers.
4. Appropriate logs have to maintained to track the behaviour and overcome failures in the pipeline.

6.0 Flow of Operations

A schematic flow of operations is shown below



Section II- Designing of the Project:



Section III- Hadoop eco-system implementation

1. Making sure that all scripts in /home/acadgild/project/scripts have 774 permissions

```
[acadgild@localhost ~]$ chmod 774 /home/acadgild/project/scripts/*
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$
```

2. We have created a batch file "**start-daemons.sh**" which will start all the daemons such as **hive**, **hbase**, **Mysql** and rest of all **hadoop** daemons.

Solution_execution.txt | start-daemons.sh

```
#!/bin/bash

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

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

echo "Starting daemons" >> $LOGFILE

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

- Using command “sh/home/acadgild/project/scripts/start-daemons.sh”
- We can also see all list active services using the **jps** command, see the below screenshot and also starting the hive metastore and created a metastore_db in the location where we desired.

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

```
File  Edit  View  Search  Terminal  Help
[acadgild@localhost ~]$ cd project/logs
[acadgild@localhost logs]$ ls
current-batch.txt  log_batch_  log_batch_1
[acadgild@localhost logs]$ cat current-batch.txt
[acadgild@localhost logs]$ cat current-batch.txt
1[acadgild@localhost logs]$
```

Section IV- Data Generation

I. Web Data generation

- Generate_web_data.py → generates some random data coming from web application python /home/acadgild/project/scripts/generate_web_data.py
- We have generated the web data from python script below.

Solution_execution.txt | generate_web_data.py

```
from random import randint
from random import choice

file = open("/home/acadgild/project/data/web/file.xml", "w")
count = 20

file.write("<records>\n")

while (count > 0):
        geo_cd_list=["A", "E", "AU", "AP", "U"]
        song_end_type_list=["0","1","2","3"]
        timestamp_list=["2016-05-10 12:24:22", "2016-06-09 22:12:36", "2016-07-10 01:38:09", "2017-05-09 08:09:22"]
        start_ts_list=["2016-05-10 12:24:22", "2016-06-09 22:12:36", "2016-07-10 01:38:09", "2017-05-09 08:09:22"]
        end_ts_list=["2016-05-10 12:24:22", "2016-06-09 22:12:36", "2016-07-10 01:38:09", "2017-05-09 08:09:22"]

        if (count%15 == 0):
                user_id = ""
        else:
                user_id = "U" + str(randint(100,120))

        song_id = "S" + str(randint(200,210))

        if (count%11 == 0):
                artist_id = ""
        else:
                artist_id = "A" + str(randint(300,305))

        timestamp = choice(timestamp_list)
        start_ts = choice(start_ts_list)
        end_ts = choice(end_ts_list)

        if (count%12 == 0):
                geo_cd = ""
```

- Details of the steps below.
  - In step 1 we are running the script name **generate_web_date.py** using command “python /home/acadgild/project/scripts/generate_web_data-py”
  - In step 2 we check whether the file is generated in the directory.
  - In step 3 we are checking data generated from above scripts in “file.xml”
  - Using command “cat file.xml”

```
acadgild@localhost:~/project/data/web
File  Edit  View  Search  Terminal  Help
[acadgild@localhost ~]$ cd project
[acadgild@localhost project]$ ls
data                    hdfs_          metastore_db
derby.log               lib            processed_dir
Flow of operations.jpg  logs           reading_line_by_line_in_unix~
Flow of operations.pptx lookupfiles    scripts
[acadgild@localhost project]$ python /home/acadgild/project/scripts/generate_web_data.py
[acadgild@localhost project]$ cd data
[acadgild@localhost data]$ ls
mob  web
[acadgild@localhost data]$ cd web
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost web]$ ls
file.xml
[acadgild@localhost web]$ cat file.xml
<records>
<record>
<user_id>U102</user_id>
<song_id>S208</song_id>
<artist_id>A305</artist_id>
<timestamp>2016-05-10 12:24:22</timestamp>
<start_ts>2016-05-10 12:24:22</start_ts>
<end_ts>2017-05-09 08:09:22</end_ts>
<geo_cd>AU</geo_cd>
<station_id>ST411</station_id>
<song_end_type>1</song_end_type>
<like>1</like>
<dislike>0</dislike>
</record>
<record>
<user_id>U118</user_id>
<song_id>S202</song_id>
<artist_id>A304</artist_id>
<timestamp>2017-05-09 08:09:22</timestamp>
<start_ts>2016-06-09 22:12:36</start_ts>
<end_ts>2016-05-10 12:24:22</end_ts>
<geo_cd>E</geo_cd>
<station_id>ST400</station_id>
<song_end_type>1</song_end_type>
```

II. Mobile Data Generation

- generate_mob_data.py → generates some random data coming from mobile application
- we have generated the mobile data from python script below.

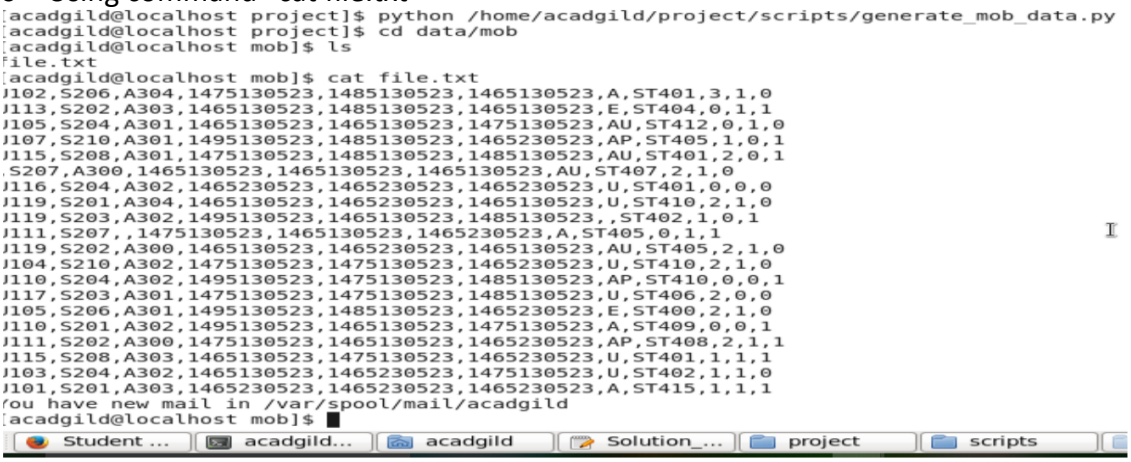
Solution_execution.txt | generate_mob_data.py

```
from random import randint
from random import choice

file = open("/home/acadgild/project/data/mob/file.txt", "w")
count = 20

while (count > 0):
        geo_cd_list=["A", "E", "AU", "AP", "U"]
        song_end_type_list=["0","1","2","3"]
        timestamp_list=["1465230523", "1465130523", "1475130523", "1495130523"]
        start_ts_list=["1465230523", "1465130523", "1475130523", "1485130523"]
        end_ts_list=["1465230523", "1465130523", "1475130523", "1485130523"]

        if (count%15 == 0):
                user_id = ""
        else:
                user_id = "U" + str(randint(100,120))

        song_id = "S" + str(randint(200,210))

        if (count%11 == 0):
                artist_id = ""
        else:
                artist_id = "A" + str(randint(300,305))

        timestamp = choice(timestamp_list)
        start_ts = choice(start_ts_list)
        end_ts = choice(end_ts_list)

        if (count%12 == 0):
                geo_cd = ""
        else:
                geo_cd = choice(geo_cd_list)
```

Pytho

- Details of the steps below.
    - In step 1 we are running the script name **generate_mob_date.py** using command “python /home/acadgild/project/scripts/generate_mob_data-py”

- In step 2 we check whether the file is generated in the directory.
- In step 3 we are checking data generated from above scripts in "file.txt"
- Using command "cat file.txt"



Section V: Data Ingestion, Formatting, Enrichment and Filtering

I. Data Ingestion

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

1) Data formatting
2) Data enrichment and
3) Analysis stage

"***populate-lookup.sh***" script:

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

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

Open Save Undo

Solution_execution.txt | populate-lookup.sh

```
#!/bin/bash

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

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

echo "Creating LookUp Tables" >> $LOGFILE

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


echo "Populating LookUp Tables" >> $LOGFILE

file="/home/acadgild/project/lookupfiles/stn-geocd.txt"
while IFS= read -r line
do
 stnid=`echo $line | cut -d',' -f1`
 geocd=`echo $line | cut -d',' -f2`
 echo "put 'station-geo-map', '$stnid', 'geo:geo_cd', '$geocd'" | hbase shell
done <"$file"


file="/home/acadgild/project/lookupfiles/song-artist.txt"
while IFS= read -r line
do
 songid=`echo $line | cut -d',' -f1`
 artistid=`echo $line | cut -d',' -f2`
 echo "put 'song-artist-map', '$songid', 'artist:artistid', '$artistid'" | hbase shell
done <"$file"


file="/home/acadgild/project/lookupfiles/user-subscn.txt"
while IFS= read -r line
do
 userid=`echo $line | cut -d',' -f1`
 startdt=`echo $line | cut -d',' -f2`
 enddt=`echo $line | cut -d',' -f3`
 echo "put 'subscribed-users', '$userid', 'subscn:startdt', '$startdt'" | hbase shell
 echo "put 'subscribed-users', '$userid', 'subscn:enddt', '$enddt'" | hbase shell
done <"$file"

hive -f /home/acadgild/project/scripts/user-artist.hql
```

- We can see that lookup tables in HBase are created above from **"populate-lookup.sh"**

```
Logging initialized using configuration in jar:file:/home/aca
ar!/hive-log4j2.properties Async: true
OK
Time taken: 14.086 seconds
OK
Time taken: 0.051 seconds
OK
Time taken: 1.847 seconds
Loading data to table project.users_artists
OK
Time taken: 2.68 seconds
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost project]$
```

```
hbase(main):001:0> list
TABLE
TRANSACTIONS
bulktable
clicks
clicks1
employee
song-artist-map
station-geo-map
subscribed-users
8 row(s) in 1.4310 seconds
```

- The values loaded in the lookup tables are shown below.
- In table ‘song-artist-map’

```
hbase(main):002:0> scan 'song-artist-map'
ROW                   COLUMN+CELL
 S200                 column=artist:artistid, timestamp=1543046433302, value=A30
                      0
 S201                 column=artist:artistid, timestamp=1543046449327, value=A30
                      1
 S202                 column=artist:artistid, timestamp=1543046465361, value=A30
                      2
 S203                 column=artist:artistid, timestamp=1543046484947, value=A30
                      3
 S204                 column=artist:artistid, timestamp=1543046505998, value=A30
                      4
 S205                 column=artist:artistid, timestamp=1543046523106, value=A30
                      1
 S206                 column=artist:artistid, timestamp=1543046540961, value=A30
                      2
 S207                 column=artist:artistid, timestamp=1543046559088, value=A30
                      3
 S208                 column=artist:artistid, timestamp=1543046576964, value=A30
                      4
 S209                 column=artist:artistid, timestamp=1543046595714, value=A30
                      5
```

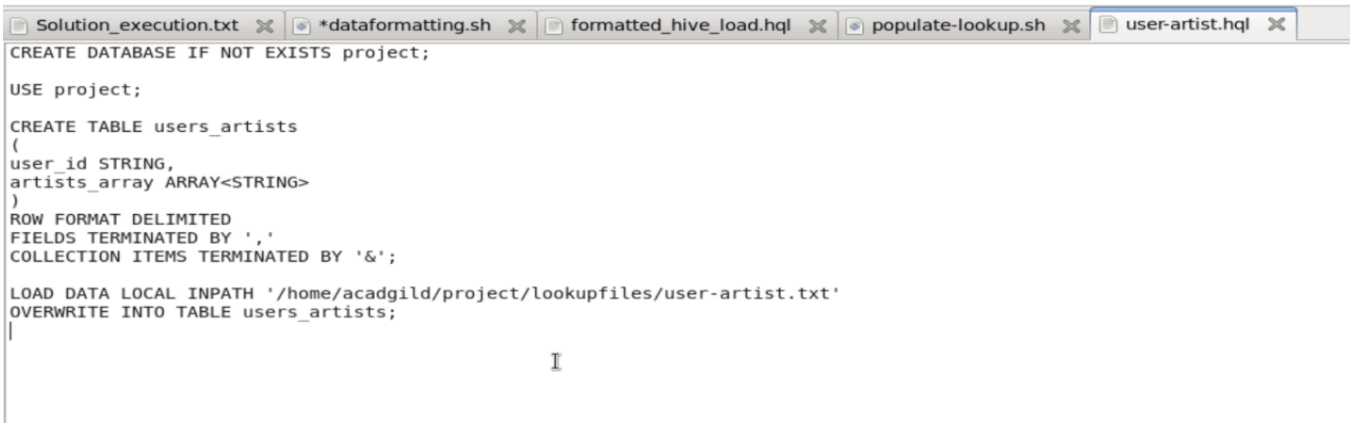
- In table ‘station-geo-map’

```
hbase(main):003:0> scan 'station-geo-map'
ROW                           COLUMN+CELL
 ST400                        column=geo:geo_cd, timestamp=1543046164584, value=A
 ST401                        column=geo:geo_cd, timestamp=1543046183119, value=AU
 ST402                        column=geo:geo_cd, timestamp=1543046201648, value=AP
 ST403                        column=geo:geo_cd, timestamp=1543046219927, value=J
 ST404                        column=geo:geo_cd, timestamp=1543046237815, value=E
 ST405                        column=geo:geo_cd, timestamp=1543046255967, value=A
 ST406                        column=geo:geo_cd, timestamp=1543046274169, value=AU
 ST407                        column=geo:geo_cd, timestamp=1543046293473, value=AP
 ST408                        column=geo:geo_cd, timestamp=1543046314537, value=E
 ST409                        column=geo:geo_cd, timestamp=1543046332490, value=E
 ST410                        column=geo:geo_cd, timestamp=1543046350752, value=A
 ST411                        column=geo:geo_cd, timestamp=1543046367003, value=A
 ST412                        column=geo:geo_cd, timestamp=1543046382939, value=AP
 ST413                        column=geo:geo_cd, timestamp=1543046399256, value=J
 ST414                        column=geo:geo_cd, timestamp=1543046416047, value=E
15 row(s) in 0.5970 seconds
```

- In table ‘subscribed-users’

```
hbase(main):004:0> scan 'subscribed-users'
ROW                                   COLUMN+CELL
 U100                                 column=subscn:enddt, timestamp=1543046631163, value=1465130523
 U100                                 column=subscn:startdt, timestamp=1543046612809, value=1465230523
 U101                                 column=subscn:enddt, timestamp=1543046666115, value=1475130523
 U101                                 column=subscn:startdt, timestamp=1543046648755, value=1465230523
 U102                                 column=subscn:enddt, timestamp=1543046701486, value=1475130523
 U102                                 column=subscn:startdt, timestamp=1543046683614, value=1465230523
 U103                                 column=subscn:enddt, timestamp=1543046738081, value=1475130523
 U103                                 column=subscn:startdt, timestamp=1543046719810, value=1465230523
 U104                                 column=subscn:enddt, timestamp=1543046773064, value=1475130523
 U104                                 column=subscn:startdt, timestamp=1543046755939, value=1465230523
 U105                                 column=subscn:enddt, timestamp=1543046810098, value=1475130523
 U105                                 column=subscn:startdt, timestamp=1543046789969, value=1465230523
 U106                                 column=subscn:enddt, timestamp=1543046846074, value=1485130523
 U106                                 column=subscn:startdt, timestamp=1543046828117, value=1465230523
 U107                                 column=subscn:enddt, timestamp=1543046880422, value=1455130523
 U107                                 column=subscn:startdt, timestamp=1543046864118, value=1465230523
 U108                                 column=subscn:enddt, timestamp=1543046916941, value=1465230623
 U108                                 column=subscn:startdt, timestamp=1543046899093, value=1465230523
 U109                                 column=subscn:enddt, timestamp=1543046951055, value=1475130523
 U109                                 column=subscn:startdt, timestamp=1543046933959, value=1465230523
 U110                                 column=subscn:enddt, timestamp=1543046986559, value=1475130523
 U110                                 column=subscn:startdt, timestamp=1543046967848, value=1465230523
 U111                                 column=subscn:enddt, timestamp=1543047020607, value=1475130523
 U111                                 column=subscn:startdt, timestamp=1543047003408, value=1465230523
 U112                                 column=subscn:enddt, timestamp=1543047054804, value=1475130523
 U112                                 column=subscn:startdt, timestamp=1543047037783, value=1465230523
 U113                                 column=subscn:enddt, timestamp=1543047088953, value=1485130523
 U113                                 column=subscn:startdt, timestamp=1543047071902, value=1465230523
 U114                                 column=subscn:enddt, timestamp=1543047122252, value=1468130523
 U114                                 column=subscn:startdt, timestamp=1543047105943, value=1465230523
15 row(s) in 0.5410 seconds
```

We have successfully created the lookup tables in HBase.
The populate-lookup.sh also creates a lookup table "**users_artists**" in the HIVE, loading the data from the **user-artist.txt**, the below screen shot shows that the table has been created in the HIVE.



We have entered in HIVE shell to check data in hive table

```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Help
[acadgild@localhost ~]$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-
bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/sha
re/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.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/acadgild/install/hive/
apache-hive-2.3.2-bin/lib/hive-common-2.3.2.jar!/hive-log4j2.properties Async: t
rue
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versio
ns. Consider using a different execution engine (i.e. spark, tez) or using Hive
1.X releases.
hive> show databases;
OK
acadgilddb
default
oly
project
Time taken: 10.007 seconds, Fetched: 4 row(s)
hive> use project;
OK
Time taken: 0.054 seconds
hive> show tables;
OK
users_artists
Time taken: 0.1 seconds, Fetched: 1 row(s)
hive> select * from users_artists;
OK
U100    ["A300","A301","A302"]
U101    ["A301","A302"]
U102    ["A302"]
U103    ["A303","A301","A302"]
U104    ["A304","A301"]
U105    ["A305","A301","A302"]
U106    ["A301","A302"]
U107    ["A302"]
U108    ["A300","A303","A304"]
U109    ["A301","A303"]
U110    ["A302","A301"]
U111    ["A303","A301"]
U112    ["A304","A301"]
U113    ["A305","A302"]
U114    ["A300","A301","A302"]
```

From above screenshot we can conclude that we have table in HIVE named as "**user_artists**" with columns USER_ID and array of artists id's
Now we need to link theses lookup tables in hive using the Hbase Storage Handler.
With the help of "**data_enrichment_filtering_schema.sh**" file we will create hive tables on the top of Hbase tables using "**create_hive_hbase_lookup.hql"**.

<u>Creating Hive tables on top of HBase:</u>
In this section with the help of Hbase storage handler &SerDe properties we are creating the hive external tables by matching the columns of Hbase tables to hive tables.

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

Solution_execution.txt | create_hive_hbase_lookup.hql | data_enrichment_filtering_schema.sh

```
#!/bin/bash

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

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

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

create_hive_hbase_lookup.hql

Solution_execution.txt | create_hive_hbase_lookup.hql | data_enrichment_filtering_s

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

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

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

```
acadgild@localhost:~/project
File  Edit  View  Search  Terminal  Help
[acadgild@localhost ~]$ cd project
[acadgild@localhost project]$ hive -f /home/acadgild/project/scripts/create_hive
_hbase_lookup.hql
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-
bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/sha
re/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.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/acadgild/install/hive/
apache-hive-2.3.2-bin/lib/hive-common-2.3.2.jar!/hive-log4j2.properties Async: t
rue
OK
Time taken: 22.677 seconds
OK
Time taken: 10.691 seconds
OK
Time taken: 1.03 seconds
OK
Time taken: 1.804 seconds
You have new mail in /var/spool/mail/acadgild
```

hive> show tables;

```
hive> show tables;
OK
enriched_data
formatted_input
formatted_input1
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 0.255 seconds, Fetched: 7 row(s)
hive>
```

hive> select * from song artist_map;

```
hive> select * from song_artist_map;
OK
S200    A300
S201    A301
S202    A302
S203    A303
S204    A304
S205    A301
S206    A302
S207    A303
S208    A304
S209    A305
Time taken: 1.235 seconds, Fetched: 10 row(s)
hive>
```

hive> select * from station_geo_map;

&

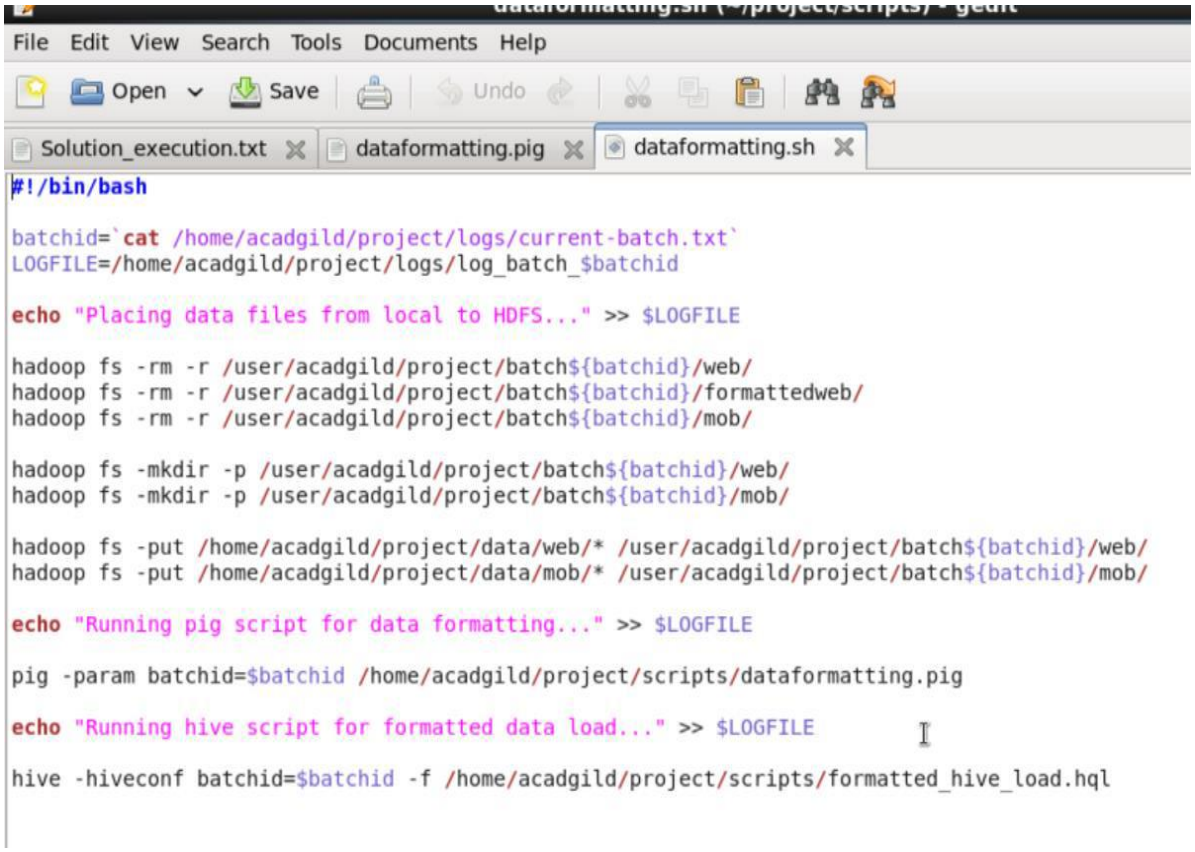
hive> select * from subscribed_users;

```
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 0.255 seconds, Fetched: 7 row(s)
hive> select * from station_geo_map;
OK
ST400   A
ST401   AU
ST402   AP
ST403   J
ST404   E
ST405   A
ST406   AU
ST407   AP
ST408   E
ST409   E
ST410   A
ST411   A
ST412   AP
ST413   J
ST414   E
Time taken: 6.611 seconds, Fetched: 15 row(s)
hive> select * from subscribed_users;
OK
U100    1465230523      1465130523
U101    1465230523      1475130523
U102    1465230523      1475130523
U103    1465230523      1475130523
U104    1465230523      1475130523
U105    1465230523      1475130523
U106    1465230523      1485130523
U107    1465230523      1455130523
U108    1465230523      1465230623
U109    1465230523      1475130523
U110    1465230523      1475130523
U111    1465230523      1475130523
U112    1465230523      1475130523
U113    1465230523      1485130523
U114    1465230523      1468130523
Time taken: 1.375 seconds, Fetched: 15 row(s)
```

II. Data Formatting:

In this stage we are merging the data coming from both **web** applications and **mobile** applications and create a common table for analyzing purpose and create partitioned data based on **batchid**, since we are running this scripts for every 3 hours.

Run the Script: *sh /home/acadgild/project/scripts/dataformatting.sh*



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

- Dataformatting.pig
- Formatted_hive_load.hql

Pig script to parse the data from coming from web_data.xml to csv format and partition both web and mob data based on based on batch ID's

- In pig script we are removing spaces using TRIM and converting data in desire time stamp using ToUnixTime
- Then storing data in formatted web using "STORE INTO using PigStorage (,) " command

Dataformatting.pig

File Edit View Search Tools Documents Help

Open Save Undo

Solution_execution.txt | dataformatting.pig

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

DEFINE XPath org.apache.pig.piggybank.evaluation.xml.XPath();

A =  LOAD '/user/acadgild/project/batch${batchid}/web/' using org.apache.pig.piggybank.storage.XMLLoader('record') as
(x:chararray);

B = FOREACH A GENERATE TRIM(XPath(x, 'record/user_id')) AS user_id,
    TRIM(XPath(x, 'record/song_id')) AS song_id,
    TRIM(XPath(x, 'record/artist_id')) AS artist_id,
    ToUnixTime(ToDate(TRIM(XPath(x, 'record/timestamp')),'yyyy-MM-dd HH:mm:ss')) AS timestamp,
    ToUnixTime(ToDate(TRIM(XPath(x, 'record/start_ts')),'yyyy-MM-dd HH:mm:ss')) AS start_ts,
    ToUnixTime(ToDate(TRIM(XPath(x, 'record/end_ts')),'yyyy-MM-dd HH:mm:ss')) AS end_ts,
    TRIM(XPath(x, 'record/geo_cd')) AS geo_cd,
    TRIM(XPath(x, 'record/station_id')) AS station_id,
    TRIM(XPath(x, 'record/song_end_type')) AS song_end_type,
    TRIM(XPath(x, 'record/like')) AS like,
    TRIM(XPath(x, 'record/dislike')) AS dislike;

STORE B INTO '/user/acadgild/project/batch${batchid}/formattedweb/' USING PigStorage(',');
```

formatted_hive_load.hql

Solution_execution.txt | dataformatting.pig | dataformatting.sh | formatted_hive_load.hql

```
USE project;

CREATE TABLE IF NOT EXISTS formatted_input
(
User_id STRING,
Song_id STRING,
Artist_id STRING,
Timestamp STRING,
Start_ts STRING,
End_ts STRING,
Geo_cd STRING,
Station_id STRING,
Song_end_type INT,
Like INT,
Dislike INT
)
PARTITIONED BY
(batchid INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

LOAD DATA INPATH '/user/acadgild/project/batch${hiveconf:batchid}/formattedweb/'
INTO TABLE formatted_input PARTITION (batchid=${hiveconf:batchid});

LOAD DATA INPATH '/user/acadgild/project/batch${hiveconf:batchid}/mob/'
INTO TABLE formatted_input PARTITION (batchid=${hiveconf:batchid});
```

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

```
OK
formatted_input
formatted_input1
users_artists
Time taken: 0.446 seconds, Fetched: 3 row(s)
hive> select * from formatted_input;
OK
U102    S206    A304    1475130523      1485130523      1465130523      A       ST401   3       1       0       1
U113    S202    A303    1465130523      1485130523      1465130523      E       ST404   0       1       1       1
U105    S204    A301    1465130523      1465130523      1475130523      AU      ST412   0       1       0       1
U107    S210    A301    1495130523      1485130523      1465230523      AP      ST405   1       0       1       1
U115    S208    A301    1475130523      1485130523      1485130523      AU      ST401   2       0       1       1
        S207    A300    1465130523      1465130523      1465130523      AU      ST407   2       1       0       1
U116    S204    A302    1465230523      1465230523      1465230523      U       ST401   0       0       0       1
U119    S201    A304    1465130523      1465230523      1465130523      U       ST410   2       1       0       1
U119    S203    A302    1495130523      1465130523      1485130523              ST402   1       0       1       1
U111    S207            1475130523      1465130523      1465230523      A       ST405   0       1       1       1
U119    S202    A300    1465130523      1465230523      1465130523      AU      ST405   2       1       0       1
U104    S210    A302    1475130523      1475130523      1465230523      U       ST410   2       1       0       1
U110    S204    A302    1495130523      1475130523      1485130523      AP      ST410   0       0       1       1
U117    S203    A301    1475130523      1475130523      1485130523      U       ST406   2       0       0       1
U105    S206    A301    1495130523      1485130523      1465230523      E       ST400   2       1       0       1
U110    S201    A302    1495130523      1465130523      1475130523      A       ST409   0       0       1       1
U111    S202    A300    1475130523      1465130523      1465230523      AP      ST408   2       1       1       1
U115    S208    A303    1465130523      1475130523      1465230523      U       ST401   1       1       1       1
U103    S204    A302    1465130523      1465230523      1475130523      U       ST402   1       1       0       1
U101    S201    A303    1465230523      1465230523      1465230523      A       ST415   1       1       1       1
U102    S208    A305    1462863262      1462863262      1494297562      AU      ST411   1       1       0       1
U118    S202    A304    1494297562      1465490556      1462863262      E       ST400   1       0       0       1
U118    S202    A304    1465490556      1465490556      1465490556      AP      ST404   0       1       0       1
U118    S205    A300    1462863262      1468094889      1465490556      U       ST407   2       1       1       1
U119    S207    A300    1462863262      1465490556      1494297562      E       ST409   3       0       0       1
        S201    A303    1494297562      1494297562      1462863262      A       ST411   3       0       1       1
U108    S203    A301    1494297562      1494297562      1462863262      U       ST405   3       0       1       1
U111    S210    A302    1494297562      1468094889      1494297562      E       ST406   1       1       1       1
U119    S209    A304    1468094889      1468094889      1468094889              ST413   1       1       0       1
U103    S209            1468094889      1494297562      1468094889      AP      ST414   0       0       1       1
U111    S207    A301    1462863262      1462863262      1468094889      E       ST406   2       1       1       1
U102    S202    A305    1468094889      1465490556      1494297562      A       ST413   1       0       1       1
U116    S209    A305    1468094889      1465490556      1465490556      AP      ST401   0       0       0       1
U103    S203    A301    1468094889      1465490556      1468094889      A       ST400   2       1       1       1
U113    S203    A300    1462863262      1462863262      1465490556      AU      ST414   2       0       1       1
U106    S200    A305    1494297562      1494297562      1494297562      U       ST402   2       0       1       1
U119    S210    A301    1462863262      1465490556      1465490556      E       ST402   0       1       1       1
U112    S207    A301    1462863262      1462863262      1494297562      E       ST405   0       0       0       1
U115    S202    A301    1468094889      1494297562      1465490556      AP      ST402   1       1       0       1
U105    S209    A302    1494297562      1462863262      1468094889      E       ST408   2       0       1       1
```

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

<u>Data Formatting Stage summary</u>

In this stage we are merging the data coming from both web applications and mobile applications and create a common table for analyzing purpose and create partitioned data based on **batchid**, since we are running this scripts for every 3 hours.

Run this script: ***sh /home/acadgild/project/scripts/dataformatting.sh***

Solution_execution.txt | dataformatting.pig | dataformatting.sh

```
#!/bin/bash

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

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

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

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

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

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

pig -param batchid=$batchid /home/acadgild/project/scripts/dataformatting.pig

echo "Running hive script for formatted data load..." >> $LOGFILE

hive -hiveconf batchid=$batchid -f /home/acadgild/project/scripts/formatted_hive_load.hql
```

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

- Dataformatting.pig
- Formatted_hive_load.hql

Pig script to parse the data from coming from web_data.xml to csv format and partition both web and mob data based on based on batch ID's

- In pig script we are removing spaces using TRIM and converting data in desire time stamp using ToUnixTime
- Then storing data in formatted web using "STORE INTO using PigStorage (,) " command

Dataformatting.pig

Solution_execution.txt | dataformatting.pig

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

DEFINE XPath org.apache.pig.piggybank.evaluation.xml.XPath();

A =  LOAD '/user/acadgild/project/batch${batchid}/web/' using org.apache.pig.piggybank.storage.XMLLoader('record') as
(x:chararray);

B = FOREACH A GENERATE TRIM(XPath(x, 'record/user_id')) AS user_id,
    TRIM(XPath(x, 'record/song_id')) AS song_id,
    TRIM(XPath(x, 'record/artist_id')) AS artist_id,
    ToUnixTime(ToDate(TRIM(XPath(x, 'record/timestamp')),'yyyy-MM-dd HH:mm:ss')) AS timestamp,
    ToUnixTime(ToDate(TRIM(XPath(x, 'record/start_ts')),'yyyy-MM-dd HH:mm:ss')) AS start_ts,
    ToUnixTime(ToDate(TRIM(XPath(x, 'record/end_ts')),'yyyy-MM-dd HH:mm:ss')) AS end_ts,
    TRIM(XPath(x, 'record/geo_cd')) AS geo_cd,
    TRIM(XPath(x, 'record/station_id')) AS station_id,
    TRIM(XPath(x, 'record/song_end_type')) AS song_end_type,
    TRIM(XPath(x, 'record/like')) AS like,
    TRIM(XPath(x, 'record/dislike')) AS dislike;

STORE B INTO '/user/acadgild/project/batch${batchid}/formattedweb/' USING PigStorage(',');
```

formatted_hive_load.hql

Solution_execution.txt | dataformatting.pig | dataformatting.sh | formatted_hive_load.hql

```
USE project;

CREATE TABLE IF NOT EXISTS formatted_input
(
User_id STRING,
Song_id STRING,
Artist_id STRING,
Timestamp STRING,
Start_ts STRING,
End_ts STRING,
Geo_cd STRING,
Station_id STRING,
Song_end_type INT,
Like INT,
Dislike INT
)
PARTITIONED BY
(batchid INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

LOAD DATA INPATH '/user/acadgild/project/batch${hiveconf:batchid}/formattedweb/'
INTO TABLE formatted_input PARTITION (batchid=${hiveconf:batchid});

LOAD DATA INPATH '/user/acadgild/project/batch${hiveconf:batchid}/mob/'
INTO TABLE formatted_input PARTITION (batchid=${hiveconf:batchid});
```

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

```
formatted_input
formatted_input1
users_artists
Time taken: 0.446 seconds, Fetched: 3 row(s)
hive> select * from formatted_input;
OK
U102    S206    A304    1475130523      1485130523      1465130523      A       ST401   3       1       0       1
U113    S202    A303    1465130523      1485130523      1465130523      E       ST404   0       1       1       1
U105    S204    A301    1465130523      1465130523      1475130523      AU      ST412   0       1       0       1
U107    S210    A301    1495130523      1485130523      1465230523      AP      ST405   1       0       1       1
U115    S208    A301    1475130523      1485130523      1485130523      AU      ST401   2       0       1       1
        S207    A300    1465130523      1465130523      1465130523      AU      ST407   2       1       0       1
U116    S204    A302    1465230523      1465230523      1465230523      U       ST401   0       0       0       1
U119    S201    A304    1465130523      1465230523      1465130523      U       ST410   2       1       0       1
U119    S203    A302    1495130523      1465130523      1485130523              ST402   1       0       1       1
U111    S207            1475130523      1465130523      1465230523      A       ST405   0       1       1       1
U119    S202    A300    1465130523      1465230523      1465130523      AU      ST405   2       1       0       1
U104    S210    A302    1475130523      1475130523      1465230523      U       ST410   2       1       0       1
U110    S204    A302    1495130523      1475130523      1485130523      AP      ST410   0       0       1       1
U117    S203    A301    1475130523      1475130523      1485130523      U       ST406   2       0       0       1
U105    S206    A301    1495130523      1485130523      1465230523      E       ST400   2       1       0       1
U110    S201    A302    1495130523      1465130523      1475130523      A       ST409   0       0       1       1
U111    S202    A300    1475130523      1465130523      1465230523      AP      ST408   2       1       1       1
U115    S208    A303    1465130523      1475130523      1465230523      U       ST401   1       1       1       1
U103    S204    A302    1465130523      1465230523      1475130523      U       ST402   1       1       0       1
U101    S201    A303    1465230523      1465230523      1465230523      A       ST415   1       1       1       1
U102    S208    A305    1462863262      1462863262      1494297562      AU      ST411   1       1       0       1
U118    S202    A304    1494297562      1465490556      1462863262      E       ST400   1       0       0       1
U118    S202    A304    1465490556      1465490556      1465490556      AP      ST404   0       1       0       1
U118    S205    A300    1462863262      1468094889      1465490556      U       ST407   2       1       1       1
U119    S207    A300    1462863262      1465490556      1494297562      E       ST409   3       0       0       1
        S201    A303    1494297562      1494297562      1462863262      A       ST411   3       0       1       1
U108    S203    A301    1494297562      1494297562      1462863262      U       ST405   3       0       1       1
U111    S210    A302    1494297562      1468094889      1494297562      E       ST406   1       1       1       1
U119    S209    A304    1468094889      1468094889      1468094889              ST413   1       1       0       1
U103    S209            1468094889      1494297562      1468094889      AP      ST414   0       0       1       1
U111    S207    A301    1462863262      1462863262      1468094889      E       ST406   2       1       1       1
U102    S202    A305    1468094889      1465490556      1494297562      A       ST413   1       0       1       1
U116    S209    A305    1468094889      1465490556      1465490556      AP      ST401   0       0       0       1
U103    S203    A301    1468094889      1465490556      1468094889      A       ST400   2       1       1       1
U113    S203    A300    1462863262      1462863262      1465490556      AU      ST414   2       0       1       1
U106    S200    A305    1494297562      1494297562      1494297562      U       ST402   2       0       1       1
U119    S210    A301    1462863262      1465490556      1465490556      E       ST402   0       1       1       1
U112    S207    A301    1462863262      1462863262      1494297562      E       ST405   0       0       0       1
U115    S202    A301    1468094889      1494297562      1465490556      AP      ST402   1       1       0       1
U105    S209    A302    1494297562      1462863262      1468094889      E       ST408   2       0       1       1
```

- In the above screenshot we can see the formatted input data with some null values in **user_id**, **aritist_id** and **geo_cd** columns which we will fill the enrichment script based on rules of enrichment for artist_id and geo_cd only. We will get neglect user_id because they didn't mentioned anything about user_id for enrichment purpose.

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

III. Data Enrichment and Filtering

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

Rules for data enrichment,

a. If any of like or dislike is **NULL** or **absent**, consider it as 0.
b. 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**.
c. If corresponding lookup entry is not found, consider that record to be invalid

So based on the enrichment rules we will fill the null **geo_cd** and **artist_id** values with the help of corresponding lookup values in **song-artist-map** and **station-geo-map** tables in Hive-Hbase tables.

data_enrichment.sh

Solution_execution.txt | data_enrichment.sh | data_enrichment.hql

```
#!/bin/bash

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

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

hive -hiveconf batchid=$batchid -f /home/acadgild/project/scripts/data_enrichment.hql

if [ ! -d "$VALIDDIR" ]
then
mkdir -p "$VALIDDIR"
fi

if [ ! -d "$INVALIDDIR" ]
then
mkdir -p "$INVALIDDIR"
fi

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

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

echo "Deleting older valid and invalid records from local file system..." >> $LOGFILE

find /home/acadgild/project/processed_dir/ -mtime +7 -exec rm {} \;
```

dataenrichment.hql

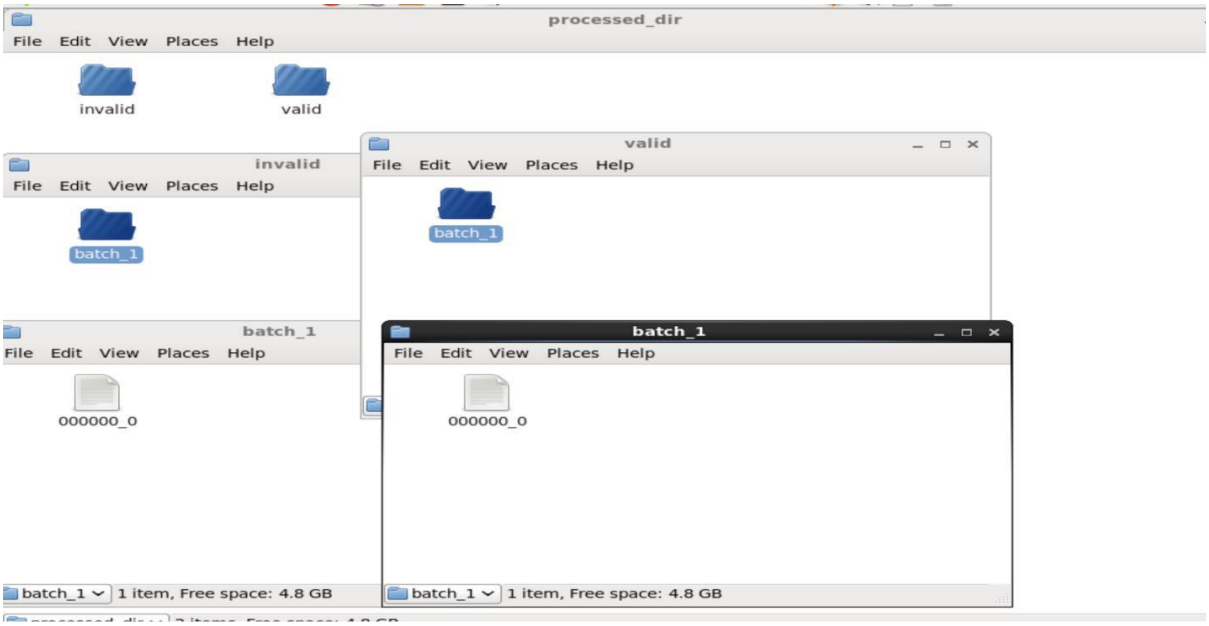
Solution_execution.txt | data_enrichment.hql | data_enrichment.s

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

USE project;

CREATE TABLE IF NOT EXISTS enriched_data
(
User_id STRING,
Song_id STRING,
Artist_id STRING,
Timestamp STRING,
Start_ts STRING,
End_ts STRING,
Geo_cd STRING,
Station_id STRING,
Song_end_type INT,
Like INT,
Dislike INT
)
PARTITIONED BY
(batchid INT,
status STRING)
STORED AS ORC;
```

```
INSERT OVERWRITE TABLE enriched_data
PARTITION (batchid, status)
SELECT
i.user_id,
i.song_id,
sa.artist_id,
i.timestamp,
i.start_ts,
i.end_ts,
sg.geo_cd,
i.station_id,
IF (i.song_end_type IS NULL, 3, i.song_end_type) AS song_end_type,
IF (i.like IS NULL, 0, i.like) AS like,
IF (i.dislike IS NULL, 0, i.dislike) AS dislike,
i.batchid,
IF((i.like=1 AND i.dislike=1)
OR i.user_id IS NULL
OR i.song_id IS NULL
OR i.timestamp IS NULL
OR i.start_ts IS NULL
OR i.end_ts IS NULL
OR i.geo_cd IS NULL
OR i.user_id=''
OR i.song_id=''
OR i.timestamp=''
OR i.start_ts=''
OR i.end_ts=''
OR i.geo_cd=''
OR sg.geo_cd IS NULL
OR sg.geo_cd=''
OR sa.artist_id IS NULL
OR sa.artist_id='', 'fail', 'pass') AS status
FROM formatted_input i
LEFT OUTER JOIN station_geo_map sg ON i.station_id = sg.station_id
LEFT OUTER JOIN song_artist_map sa ON i.song_id = sa.song_id
WHERE i.batchid=${hiveconf:batchid};
```

We have run the data enrichment script below:

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



Now we can check whether the data properly loaded in the hive terminal or not.
In the below screenshot we have data for **enriched_data** table where we filled the null values of **artist_id** and **geo_cd** of formatted input with the help of lookup tables,

hive>select * From enriched_data;

```
hive> select * from enriched_data;
OK
U101    S201    A303    1465230523      1465230523      1465230523      A       ST415   1       1       1       1       fail
        S201    A303    1494297562      1494297562      1462863262      A       ST411   3       0       1       1       fail
U111    S202    A300    1475130523      1465130523      1465230523      AP      ST408   2       1       1       1       fail
U113    S202    A303    1465130523      1485130523      1465130523      E       ST404   0       1       1       1       fail
U103    S203    A301    1468094889      1465490556      1468094889      A       ST400   2       1       1       1       fail
U118    S205    A300    1462863262      1468094889      1465490556      U       ST407   2       1       1       1       fail
U111    S207    A301    1462863262      1462863262      1468094889      E       ST406   2       1       1       1       fail
        S207    A300    1465130523      1465130523      1465130523      AU      ST407   2       1       0       1       fail
U111    S207    A303    1475130523      1465130523      1465230523      A       ST405   0       1       1       1       fail
U115    S208    A303    1465130523      1475130523      1465230523      U       ST401   1       1       1       1       fail
U107    S210    A301    1495130523      1485130523      1465230523      AP      ST405   1       0       1       1       fail
U111    S210    A302    1494297562      1468094889      1494297562      E       ST406   1       1       1       1       fail
U104    S210    A302    1475130523      1475130523      1465230523      U       ST410   2       1       0       1       fail
U119    S210    A301    1462863262      1465490556      1465490556      E       ST402   0       1       1       1       fail
U106    S200    A305    1494297562      1494297562      1494297562      U       ST402   2       0       1       1       pass
U119    S201    A304    1465130523      1465230523      1465130523      U       ST410   2       1       0       1       pass
U110    S201    A302    1495130523      1465130523      1475130523      A       ST409   0       0       1       1       pass
U118    S202    A304    1494297562      1465490556      1462863262      E       ST400   1       0       0       1       pass
U102    S202    A305    1468094889      1465490556      1494297562      A       ST413   1       0       1       1       pass
U115    S202    A301    1468094889      1494297562      1465490556      AP      ST402   1       1       0       1       pass
U118    S202    A304    1465490556      1465490556      1465490556      AP      ST404   0       1       0       1       pass
U119    S202    A300    1465130523      1465230523      1465130523      AU      ST405   2       1       0       1       pass
U113    S203    A300    1462863262      1462863262      1465490556      AU      ST414   2       0       1       1       pass
U117    S203    A301    1475130523      1475130523      1485130523      U       ST406   2       0       0       1       pass
U108    S203    A301    1494297562      1494297562      1462863262      U       ST405   3       0       1       1       pass
U119    S203    A302    1495130523      1465130523      1485130523      AP      ST402   1       0       1       1       pass
U110    S204    A302    1495130523      1475130523      1485130523      AP      ST410   0       0       1       1       pass
U116    S204    A302    1465230523      1465230523      1465230523      U       ST401   0       0       0       1       pass
U105    S204    A301    1465130523      1465130523      1475130523      AU      ST412   0       1       0       1       pass
U103    S204    A302    1465130523      1465230523      1475130523      U       ST402   1       1       0       1       pass
U102    S206    A304    1475130523      1485130523      1465130523      A       ST401   3       1       0       1       pass
U105    S206    A301    1495130523      1485130523      1465230523      E       ST400   2       1       0       1       pass
U112    S207    A301    1462863262      1462863262      1494297562      E       ST405   0       0       0       1       pass
U119    S207    A300    1462863262      1465490556      1494297562      E       ST409   3       0       0       1       pass
U115    S208    A301    1475130523      1485130523      1485130523      AU      ST401   2       0       1       1       pass
U102    S208    A305    1462863262      1462863262      1494297562      AU      ST411   1       1       0       1       pass
U119    S209    A304    1468094889      1468094889      1468094889      J       ST413   1       1       0       1       pass
```

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

IV. Data Analysis using spark

In this stage we will do analysis on enriched data data_analysis.

*Solution_execution.txt | data_analysis.sh | data_analysis.hql

```
#!/bin/bash

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

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

hive -hiveconf batchid=$batchid -f /home/acadgild/project/scripts/data_analysis.hql

sh /home/acadgild/project/scripts/data_export.sh

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

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

dataanalysis.hql

data_analysis.hql

```
1  set hive.support.sql11.reserved.keywords=false;
2  SET hive.auto.convert.join=false;
3  USE project;
4
5  CREATE TABLE IF NOT EXISTS top_10_stations
6  (
7  station_id STRING,
8  total_distinct_songs_played INT,
9  distinct_user_count INT
10 )
11 PARTITIONED BY (batchid INT)
12 ROW FORMAT DELIMITED
13 FIELDS TERMINATED BY ','
14 STORED AS TEXTFILE;
15
16 INSERT OVERWRITE TABLE top_10_stations
17 PARTITION(batchid=${hiveconf:batchid})
18 SELECT
19 station_id,
20 COUNT(DISTINCT song_id) AS total_distinct_songs_played,
21 COUNT(DISTINCT user_id) AS distinct_user_count
22 FROM enriched_data
23 WHERE status='pass'
24 AND batchid=${hiveconf:batchid}
25 AND like=1
26 GROUP BY station_id
27 ORDER BY total_distinct_songs_played DESC
28 LIMIT 10;
29
```

data_analysis.hql

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

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

data_analysis.hql

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

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

data_analysis.hql

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

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



CREATE TABLE IF NOT EXISTS top_10_unsubscribed_users
(
user_id STRING,
duration INT
)
PARTITIONED BY (batchid INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

INSERT OVERWRITE TABLE top_10_unsubscribed_users
PARTITION(batchid=${hiveconf:batchid})
SELECT
ed.user_id,
SUM(ABS(CAST(ed.end_ts AS DECIMAL(20,0))-CAST(ed.start_ts AS DECIMAL(20,0)))) AS duration
FROM enriched_data ed
LEFT OUTER JOIN subscribed_users su
ON ed.user_id=su.user_id
WHERE ed.status='pass'
AND ed.batchid=${hiveconf:batchid}
AND (su.user_id IS NULL OR (CAST(ed.timestamp AS DECIMAL(20,0)) > CAST(su.subscn_end_dt AS DECIMAL(20,0))))
GROUP BY ed.user_id
ORDER BY duration DESC
LIMIT 10;

```

We ran data_analysis.hql for below data analysis query,once query completed successfully then,we checked tables created from above query with
select * table_name;
at end of query execution.

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

```
hive> CREATE TABLE IF NOT EXISTS top_10_stations
    > (
    > station_id STRING,
    > total_distinct_songs_played INT,
    > distinct_user_count INT
    > )
    > PARTITIONED BY (batchid INT)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE;
OK
Time taken: 1.342 seconds
hive>
    > INSERT OVERWRITE TABLE top_10_stations
    > PARTITION(batchid=1)
    > SELECT
    > station_id,
    > COUNT(DISTINCT song_id) AS total_distinct_songs_played,
    > COUNT(DISTINCT user_id) AS distinct_user_count
    > FROM enriched_data
    > WHERE status='pass'
    > AND batchid=1
    > AND like1=1
    > GROUP BY station_id
    > ORDER BY total_distinct_songs_played DESC
    > LIMIT 10;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20181124202807_8bdde27d-e07d-4a24-a2a5-06566ce3f723
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>
2018-11-24 20:29:44,567 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 11.07 sec
MapReduce Total cumulative CPU time: 11 seconds 70 msec
Ended Job = job_1543044437289_0009
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1543044437289_0010, Tracking URL = http://localhost:8088/proxy/application_1543044437289_0010/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1543044437289_0010
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-11-24 20:30:08,448 Stage-2 map = 0%,  reduce = 0%
2018-11-24 20:30:22,184 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 2.18 sec
2018-11-24 20:30:39,037 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 7.09 sec
MapReduce Total cumulative CPU time: 7 seconds 630 msec
Ended Job = job_1543044437289_0010
Loading data to table project.top_10_stations partition (batchid=1)
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 11.07 sec   HDFS Read: 12938 HDFS Write: 321 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 7.63 sec   HDFS Read: 7508 HDFS Write: 179 SUCCESS
Total MapReduce CPU Time Spent: 18 seconds 700 msec
OK
Time taken: 157.229 seconds
hive> select * from top_10_stations;
OK
ST402   2       2       1
ST413   1       1       1
ST412   1       1       1
ST411   1       1       1
ST410   1       1       1
ST405   1       1       1
ST404   1       1       1
ST400   1       1       1
ST401   1       1       1
Time taken: 0.806 seconds, Fetched: 9 row(s)
hive>
```

Query-2: Determine total duration of songs played by each type of user, where type of user can be **'subscribed'** or **'unsubscribed'**. An unsubscribed user is the one whose record is either not present in Subscribed_users lookup table or has subscription_end_date earlier than the timestamp of the song played by him.

```
hive>
    > CREATE TABLE IF NOT EXISTS users_behaviour
    > (
    > user_type STRING,
    > duration INT
    > )
    > PARTITIONED BY (batchid INT)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE;
```

```
hive> INSERT OVERWRITE TABLE users_behaviour
    > PARTITION(batchid=1)
    > SELECT
    > CASE WHEN (su.user_id IS NULL OR CAST(ed.timestmp AS DECIMAL(20,0)) > CAST(su.subscn_end_dt AS DECIMAL(20,0))) THEN 'UN
SUBSCRIBED'
    > WHEN (su.user_id IS NOT NULL AND CAST(ed.timestmp AS DECIMAL(20,0)) <= CAST(su.subscn_end_dt AS DECIMAL(20,0))) THEN 'S
UBSCRIBED'
    > END AS user_type,
    > SUM(ABS(CAST(ed.end_ts AS DECIMAL(20,0))-CAST(ed.start_ts AS DECIMAL(20,0)))) AS duration
    > FROM enriched_data ed
    > LEFT OUTER JOIN subscribed_users su
    > ON ed.user_id=su.user_id
    > WHERE ed.status='pass'
    > AND ed.batchid=1
    > GROUP BY CASE WHEN (su.user_id IS NULL OR CAST(ed.timestmp AS DECIMAL(20,0)) > CAST(su.subscn_end_dt AS DECIMAL(20,0)))
 THEN 'UNSUBSCRIBED'
    > WHEN (su.user_id IS NOT NULL AND CAST(ed.timestmp AS DECIMAL(20,0)) <= CAST(su.subscn_end_dt AS DECIMAL(20,0))) THEN 'S
UBSCRIBED' END;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20181124203740_3b8061cc-9b46-4399-8181-d920f6f45974
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_1543044437289_0011, Tracking URL = http://localhost:8088/proxy/application_1543044437289_0011/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1543044437289_0011
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1
2018-11-24 20:38:10,500 Stage-1 map = 0%,  reduce = 0%
2018-11-24 20:38:41,637 Stage-1 map = 50%,  reduce = 0%, Cumulative CPU 3.63 sec
2018-11-24 20:38:44,098 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 9.87 sec
2018-11-24 20:39:03,069 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 14.84 sec
hive> INSERT OVERWRITE TABLE users_behaviour
    > PARTITION(batchid=1)
    > SELECT
    > CASE WHEN (su.user_id IS NULL OR CAST(ed.timestmp AS DECIMAL(20,0)) > CAST(su.subscn_end_dt AS DECIMAL(20,0))) THEN 'UN
SUBSCRIBED'
    > WHEN (su.user_id IS NOT NULL AND CAST(ed.timestmp AS DECIMAL(20,0)) <= CAST(su.subscn_end_dt AS DECIMAL(20,0))) THEN 'S
UBSCRIBED'
    > END AS user_type,
    > SUM(ABS(CAST(ed.end_ts AS DECIMAL(20,0))-CAST(ed.start_ts AS DECIMAL(20,0)))) AS duration
    > FROM enriched_data ed
    > LEFT OUTER JOIN subscribed_users su
    > ON ed.user_id=su.user_id
    > WHERE ed.status='pass'
    > AND ed.batchid=1
    > GROUP BY CASE WHEN (su.user_id IS NULL OR CAST(ed.timestmp AS DECIMAL(20,0)) > CAST(su.subscn_end_dt AS DECIMAL(20,0)))
 THEN 'UNSUBSCRIBED'
    > WHEN (su.user_id IS NOT NULL AND CAST(ed.timestmp AS DECIMAL(20,0)) <= CAST(su.subscn_end_dt AS DECIMAL(20,0))) THEN 'S
UBSCRIBED' END;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20181124203740_3b8061cc-9b46-4399-8181-d920f6f45974
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_1543044437289_0011, Tracking URL = http://localhost:8088/proxy/application_1543044437289_0011/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1543044437289_0011
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1
2018-11-24 20:38:10,500 Stage-1 map = 0%,  reduce = 0%
2018-11-24 20:38:41,637 Stage-1 map = 50%,  reduce = 0%, Cumulative CPU 3.63 sec
2018-11-24 20:38:44,098 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 9.87 sec
2018-11-24 20:39:03,069 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 14.84 sec
2018-11-24 20:39:04,289 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 16.18 sec
MapReduce Total cumulative CPU time: 16 seconds 180 msec
Ended Job = job_1543044437289_0011
Launching Job 2 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1543044437289_0012, Tracking URL = http://localhost:8088/proxy/application_1543044437289_0012/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1543044437289_0012
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-11-24 20:39:26,977 Stage-2 map = 0%,  reduce = 0%
2018-11-24 20:39:40,150 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 2.35 sec
2018-11-24 20:39:56,515 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 7.47 sec
MapReduce Total cumulative CPU time: 7 seconds 610 msec
Ended Job = job_1543044437289_0012
Loading data to table project.users_behaviour partition (batchid=1)
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 1   Cumulative CPU: 16.18 sec   HDFS Read: 36086 HDFS Write: 166 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 7.61 sec   HDFS Read: 6751 HDFS Write: 133 SUCCESS
Total MapReduce CPU Time Spent: 23 seconds 790 msec
OK
Time taken: 140.022 seconds
hive>
```

```
hive> select * from users_behaviour;
OK
SUBSCRIBED      160405573       1
UNSUBSCRIBED    167007233       1
Time taken: 0.617 seconds, Fetched: 2 row(s)
hive>
```

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

```
hive> CREATE TABLE IF NOT EXISTS connected_artists
    > (
    > artist_id STRING,
    > user_count INT
    > )
    > PARTITIONED BY (batchid INT)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE;
OK
Time taken: 0.281 seconds
hive> INSERT OVERWRITE TABLE connected_artists
    > PARTITION(batchid=1)
    > SELECT
    > ua.artist_id,
    > COUNT(DISTINCT ua.user_id) AS user_count
    > FROM
    > (
    > SELECT user_id, artist_id FROM users_artists
    > LATERAL VIEW explode(artists_array) artists AS artist_id
    > ) ua
    > INNER JOIN
    > (
    > SELECT artist_id, song_id, user_id
    > FROM enriched_data
    > WHERE status='pass'
    > AND batchid=1
    > ) ed
    > ON ua.artist_id=ed.artist_id
    > AND ua.user_id=ed.user_id
    > GROUP BY ua.artist_id
    > ORDER BY user_count DESC
    > LIMIT 10:
```

```
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20181124204548_0de62fe4-91b9-484c-8f3e-86f64845e3b0
Total jobs = 3
Launching Job 1 out of 3
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_1543044437289_0013, Tracking URL = http://localhost:8088/proxy/application_1543044437289_0013/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1543044437289_0013
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1
2018-11-24 20:46:09,009 Stage-1 map = 0%,  reduce = 0%
2018-11-24 20:46:40,962 Stage-1 map = 50%,  reduce = 0%, Cumulative CPU 5.12 sec
2018-11-24 20:46:42,695 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 10.53 sec
2018-11-24 20:46:59,244 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 14.28 sec
MapReduce Total cumulative CPU time: 14 seconds 280 msec
Ended Job = job_1543044437289_0013
Launching Job 2 out of 3
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_1543044437289_0014, Tracking URL = http://localhost:8088/proxy/application_1543044437289_0014/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1543044437289_0014
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-11-24 20:47:23,072 Stage-2 map = 0%,  reduce = 0%
2018-11-24 20:47:35,131 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 2.2 sec
2018-11-24 20:47:50,494 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 5.38 sec
MapReduce Total cumulative CPU time: 5 seconds 380 msec
Ended Job = job_1543044437289_0014
Launching Job 3 out of 3
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1543044437289_0015, Tracking URL = http://localhost:8088/proxy/application_1543044437289_0015/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1543044437289_0015
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 1
2018-11-24 20:48:12,623 Stage-3 map = 0%,  reduce = 0%
2018-11-24 20:48:26,008 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 2.26 sec
2018-11-24 20:48:43,842 Stage-3 map = 100%,  reduce = 100%, Cumulative CPU 7.7 sec
MapReduce Total cumulative CPU time: 7 seconds 700 msec
Ended Job = job_1543044437289_0015
Loading data to table project.connected_artists partition (batchid=1)
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 1   Cumulative CPU: 14.28 sec   HDFS Read: 26296 HDFS Write: 236 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 5.38 sec   HDFS Read: 5203 HDFS Write: 142 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 7.7 sec   HDFS Read: 6603 HDFS Write: 105 SUCCESS
Total MapReduce CPU Time Spent: 27 seconds 360 msec
OK
Time taken: 180.663 seconds
hive> select * from connected_artists;
OK
A302    3       1
A301    2       1
Time taken: 1.805 seconds, Fetched: 2 row(s)
hive>
```

Query-4: Determine top 10 songs who have generated the maximum revenue. Royalty applies to a song only if it was liked or was completed successfully or both

```
hive> CREATE TABLE IF NOT EXISTS top_10_royalty_songs
    > (
    > song_id STRING,
    > duration INT
    > )
    > PARTITIONED BY (batchid INT)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE;
OK
Time taken: 0.388 seconds
hive>
    > INSERT OVERWRITE TABLE top_10_royalty_songs
    > PARTITION(batchid=1)
    > SELECT song_id,
    > SUM(ABS(CAST(end_ts AS DECIMAL(20,0))-CAST(start_ts AS DECIMAL(20,0)))) AS duration
    > FROM enriched_data
    > WHERE status='pass'
    > AND batchid=1
    > AND (like1=1 OR song_end_type=0)
    > GROUP BY song_id
    > ORDER BY duration DESC
    > LIMIT 10;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20181124205521_396af61b-e3f3-4b24-af7d-4133c21da9e4
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_1543044437289_0016, Tracking URL = http://localhost:8088/proxy/application_1543044437289_0016/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1543044437289_0016
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-11-24 20:55:39,478 Stage-1 map = 0%,  reduce = 0%
2018-11-24 20:55:57,078 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 5.62 sec
2018-11-24 20:56:14,971 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 9.21 sec
MapReduce Total cumulative CPU time: 9 seconds 210 msec
Ended Job = job_1543044437289_0016
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1543044437289_0017, Tracking URL = http://localhost:8088/proxy/application_1543044437289_0017/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1543044437289_0017
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-11-24 20:56:38,817 Stage-2 map = 0%,  reduce = 0%
2018-11-24 20:56:51,862 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 2.55 sec
2018-11-24 20:57:09,484 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 7.22 sec
MapReduce Total cumulative CPU time: 7 seconds 790 msec
Ended Job = job_1543044437289_0017
Loading data to table project.top_10_royalty_songs partition (batchid=1)
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 9.21 sec   HDFS Read: 13589 HDFS Write: 292 SUCCESS
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 9.21 sec   HDFS Read: 13589 HDFS Write: 292 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 7.79 sec   HDFS Read: 6920 HDFS Write: 192 SUCCESS
Total MapReduce CPU Time Spent: 17 seconds 0 msec
OK
Time taken: 111.812 seconds
hive> select * from top_10_royalty_songs;
OK
S206    39900000        1
S208    31434300        1
S207    31434300        1
S204    29900000        1
S202    28907006        1
S209    26202673        1
S201    10100000        1
Time taken: 0.523 seconds, Fetched: 7 row(s)
hive>
```

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

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

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

a. Spark_analysis for creating tables for each query/problem statement.

b. Spark_analysis_2 for displaying results for each query in terminal.

```
#!/bin/bash

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

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

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

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

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

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

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

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

Spark_analysis.scala

Project Explorer | Console | spark_analysis.scala | spark_analysis_2.scala

```
package spark_analysis

import org.apache.spark.sql.SparkSession
import org.apache.spark.SparkContext._
import org.apache.spark._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
//import org.apache.spark.streaming.streaming.flume._
import org.apache.spark.sql._
import org.apache.hadoop.hive._
object spark_analysis {
  def main(args: Array[String]): Unit = {
    val sparkSession = SparkSession.builder()
      .master("local[2]")
      .appName("Data Analysis Main_1")
      .config("spark.sql.warehouse.dir","/user/hive/warehouse")
      .config("hive.metastore.uris","thrift://127.0.0.1:9083")
      .enableHiveSupport()
      .getOrCreate()

val batchId = args(0)
  val set_properties = sparkSession.sqlContext.sql("set hive.auto.convert.join=false")

val use_project_database = sparkSession.sqlContext.sql("USE project")
   //<<<<<<<<<<---------- PROBLEM 5 - Creation of table and Insertion of data ------------>>>>>>>>>>>>
    //Determine top 10 unsubscribed users who listened to the songs for the longest duration.
```

Project Explorer | Console | spark_analysis.scala | spark_analysis_2.scala

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


val insert_into_unsubscribed_users = sparkSession.sqlContext.sql("INSERT OVERWRITE TABLE project.top_10_unsubscri
  s" PARTITION (batchid=$batchId)"+
  " SELECT"+
  " user_id,"+
  " song_id,"+
  " artist_id,"+
  " total_duration_in_minutes"+
  " FROM project.song_duration"+
  " WHERE user_type='unsubscribed'"+
  " AND total_duration_in_minutes>=0"+
  s" AND (batchid=$batchId)"+
  " ORDER BY total_duration_in_minutes desc"+
  " LIMIT 10")
```

Spark_analysis2.scala

Project Explorer | Console | spark_analysis.scala | spark_analysis_2.scala

```scala
package spark_analysis

import org.apache.spark.sql.SparkSession
import org.apache.spark.SparkContext._
import org.apache.spark._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
//import org.apache.spark.streaming.streaming.flume._
import org.apache.spark.sql._
import org.apache.hadoop.hive._

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

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

Tables created in HIVE

```
hive> show tables;
OK
connected_artists
enriched_data
enriched_data1
formatted_input
formatted_input1
song_artist_map
station_geo_map
subscribed_users
top_10_royalty_songs
top_10_stations
top_10_unsubscribed_users
users_artists
users_behaviour
Time taken: 32.729 seconds, Fetched: 13 row(s)
hive>
```

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

V. Data Storage in MySQL

*Solution_execution.txt | data_export.sh

```
#!/bin/bash

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

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

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

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

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

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

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

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

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

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

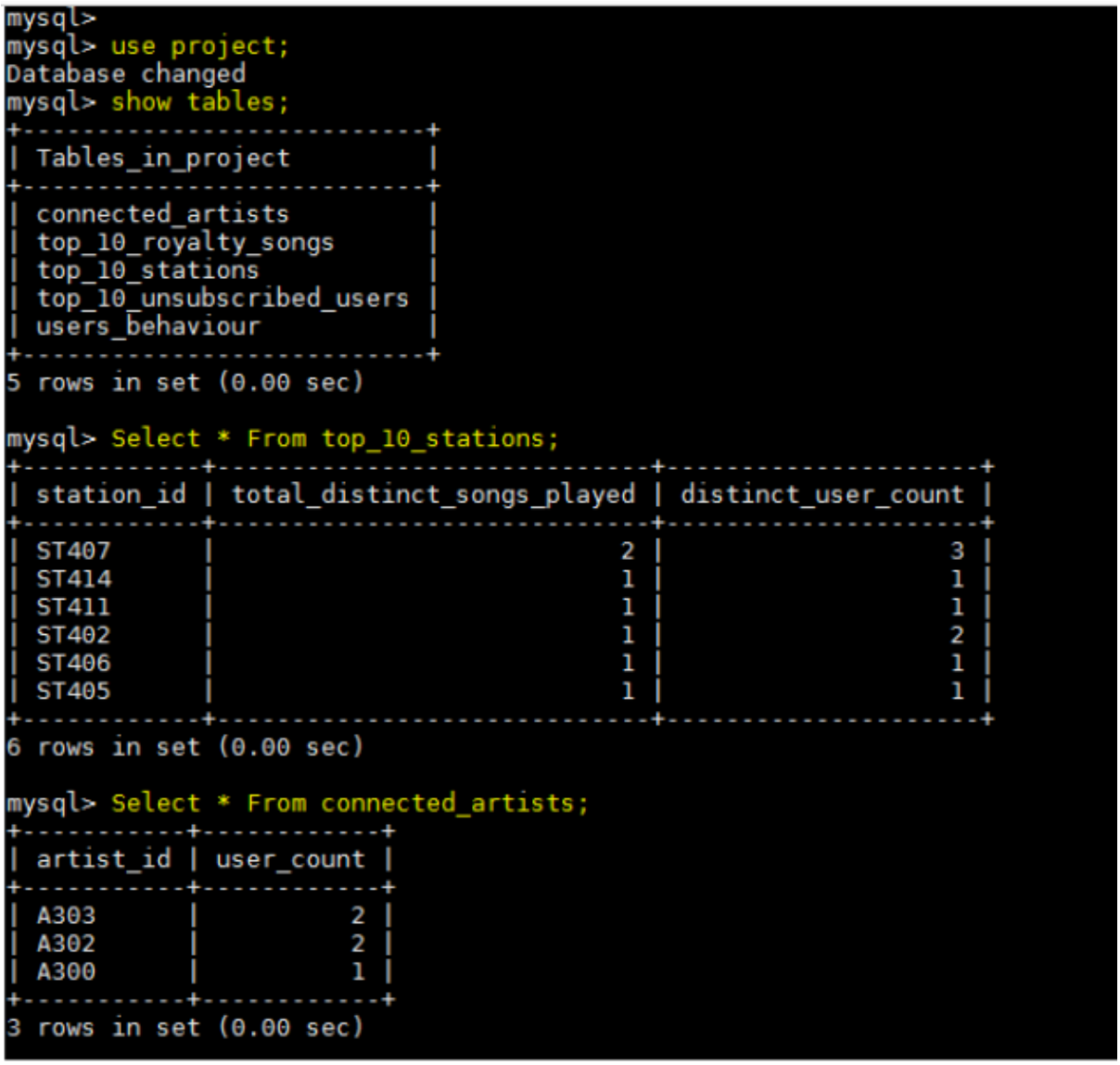
create_schema.sql – Make sure that you logged in to MySql. The below schema will create the database and tables in the MySQL.

*Solution_execution.txt | create_schema.sql | data_export.sh

```
CREATE DATABASE IF NOT EXISTS project;

USE project;

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

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

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

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

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

commit;
```

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



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

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

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

top_10_rayalty_songs,

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

Output from **top_10_unsubscribed_users** and **users_behaviour**

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

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

Job Scheduling

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

```
--driver-class-path /home/acadgild/apache-hive-2.1.0-bin/lib/hive-hbase-handler-
/home/acadgild/project/lib/sparkanalysis.jar $batchid

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

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

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

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

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

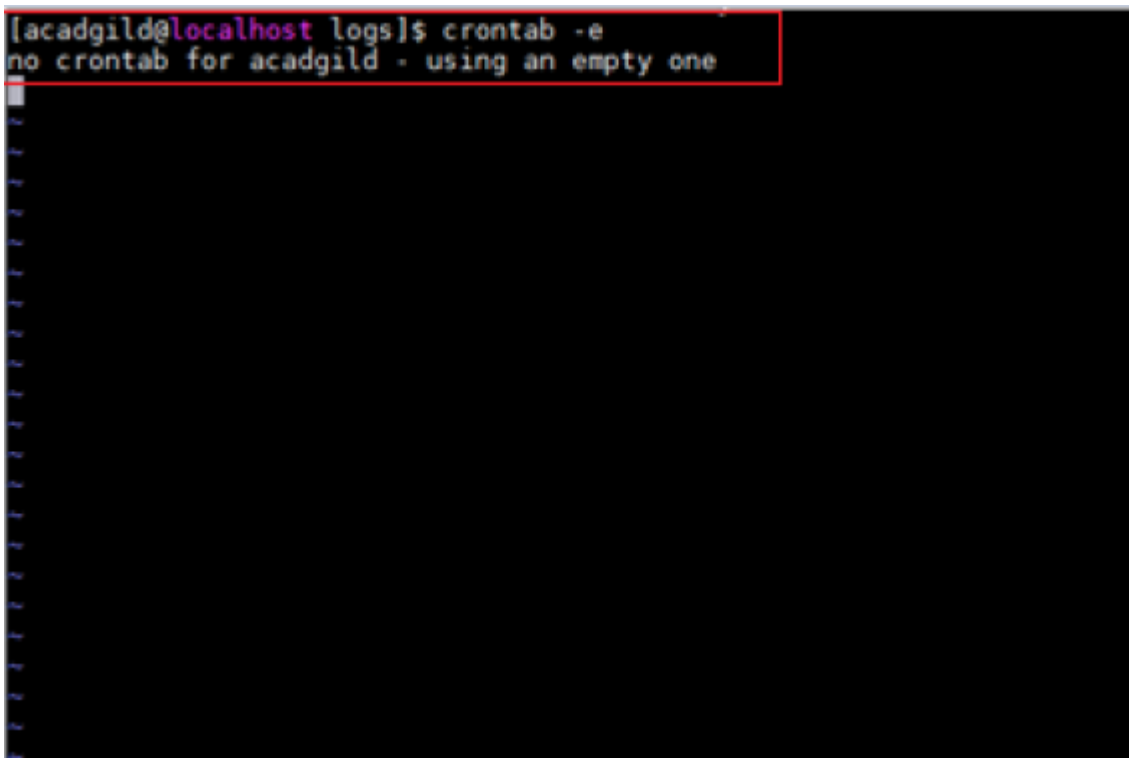
Wrapping all the scripts inside the single script file and scheduling this file to run at the periodic interval of every 3 hours.

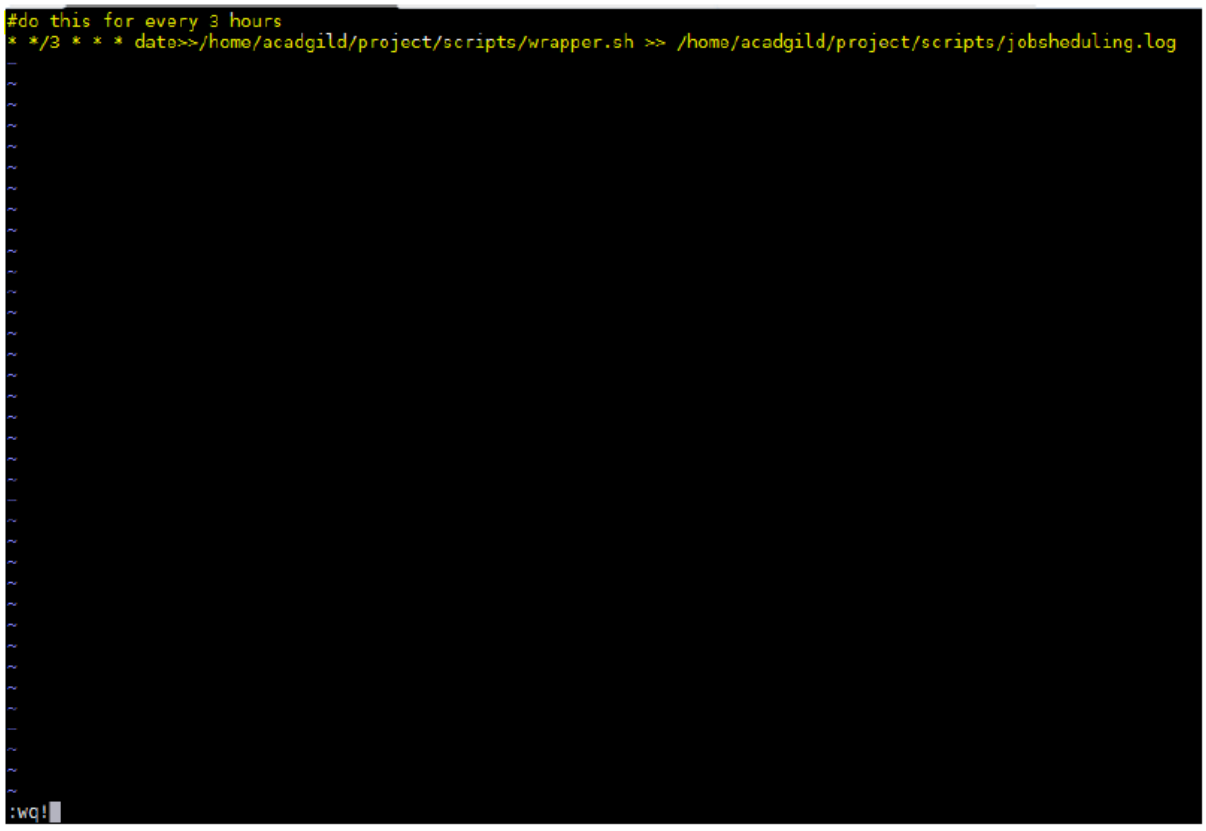
*Solution_execution.txt | data_export.sh | data_analysis.sh | wrapper.sh

```
#!/bin/bash

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

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

sh /home/acadgild/project/scripts/start-daemons.sh

sh /home/acadgild/project/scripts/populate-lookup.sh

sh /home/acadgild/project/scripts/dataformatting.sh

sh /home/acadgild/project/scripts/data_enrichment.sh

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

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

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





```
[acadgild@localhost logs]$ crontab -e
no crontab for acadgild - using an empty one
crontab: installing new crontab
[acadgild@localhost logs]$
```

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

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

Problems faced during project installation and how it resolved

1. The hive cannot be created and received the below errors
   Error:FailedPredicateException(identifier,{useSQL11ReservedKeywordsForIdentifier()}?) at org.apache.hadoop.hive.ql.parse.HiveParser_IdentifiersParser.identifier(HiveParser_Identifie rsParser.jav a:11644) at org.apache.hadoop.hive.ql.parse.HiveParser.identifier(HiveParser.java:45920) at org.apache.hadoop.hive.ql.parse.HiveParser.tabTypeExpr(HiveParser.java:15574)

2. The Spark submit cannot connect to the Hive
   Solution: Move **hive-site.xml** from $HIVE_HOME/conf/hive-site.xml to $SPARK_HOME/conf/hive-site.xml. Make an entry regarding hive metastore uris in this file. The entry will look like this:

```
<property>
  <name>hive.metastore.warehouse.dir</name>
  <value>${test.build.data}/sqoop/warehouse</value>
</property>
<property>
  <name>hive.metastore.uris</name>
  <value>thrift://localhost:9083</value>
      <description>URI for clien to connect to metastore</description>
</property>
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:derby:;databaseName=${test.build.data}/sqoop/metastore_db;create=true</value>
</property>
<property>
```

   The above modified **hive-site.xml** site is linked to the $SPARK_HOME/conf using below command

$SPARK_HOME/conf]$ ln –s /home/acadgild/project/scripts/hive-site.xml

Project End Conclusion:

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