ACADGILD

Music Data Analysis

Contents

Section – 1 - Project Overview

1.1 Fields present in the data files 2
1.2 Lookup Tables 3
1.3 DATASET 3
1.4 Data Enrichment 5
1.5 Data Analysis 5
1.6 Challenges and Optimizations.....6
1.7 Flow of operations 6

Section -2 – Design of the Project

2.1 Rough/Low Level Design.....7
2.2 High Level Design 8

Section-3-Hadoop Eco-System Implementation 9

Section-4 –Data Ingestion, Formatting, Enrichment and Filtering

4.1 Stage – 1 – Data Ingestion 12
4.2 Stage – 2 - Data Formatting 22
4.3 Stage – 3 - Data Enrichment & Filtering 28
4.4 Stage – 4 - Data Analysis using Spark 33
4.5 Stage – 5 – Data Storage in MYSQL 42

Job Scheduling 45

Project End Conclusion 48

Section – 1 - Project Overview

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

1.1 Fields present in the data files

Data files contain below fields.

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

1.2 LookUp Tables

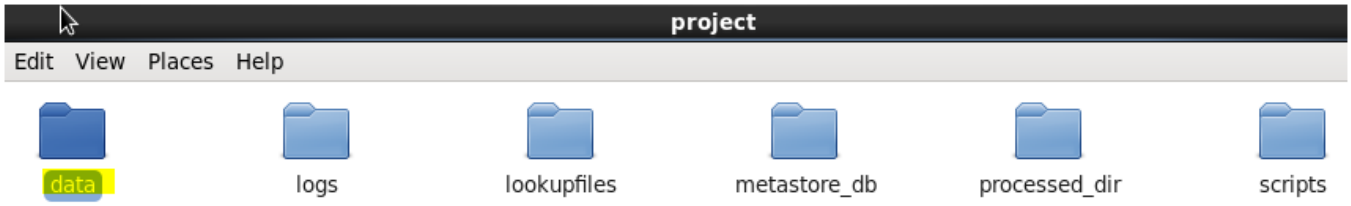
There are some existing looks up tables present in **NoSQL** databases. They play an important role in data enrichment and analysis.

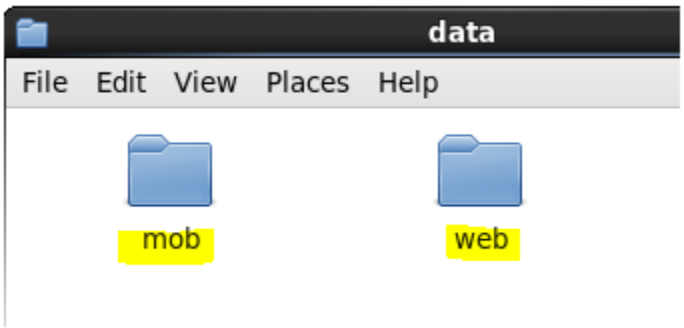
Table Name	**Decription**
Station_Geo_Map	Contains mapping of geo_cd with station_id
Subscribed_Users	Contains user_id,subscription_start_date and subscription_end_date.Contains details only for subscribed users
Song_Artist_Map	Contains mapping of song_id with artist_id along with royalty associated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

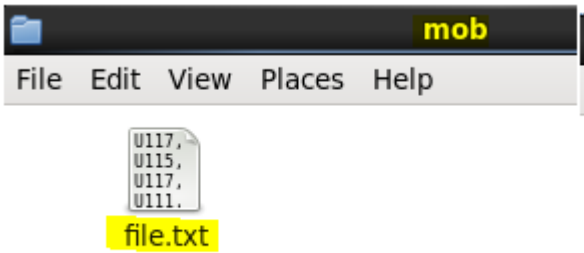
1.3 DATASET

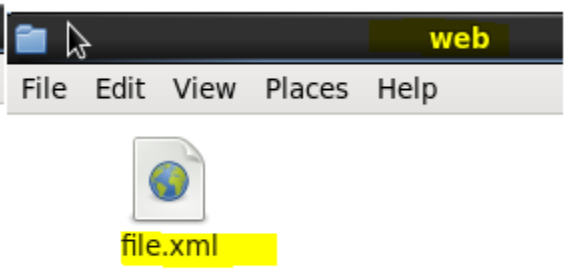
1. Data coming from web applications reside in /data/web and has xml format.
2. Data coming from mobile applications reside in /data/mob and has csv format.
3. Data present in lookup directory should be used in HBase.

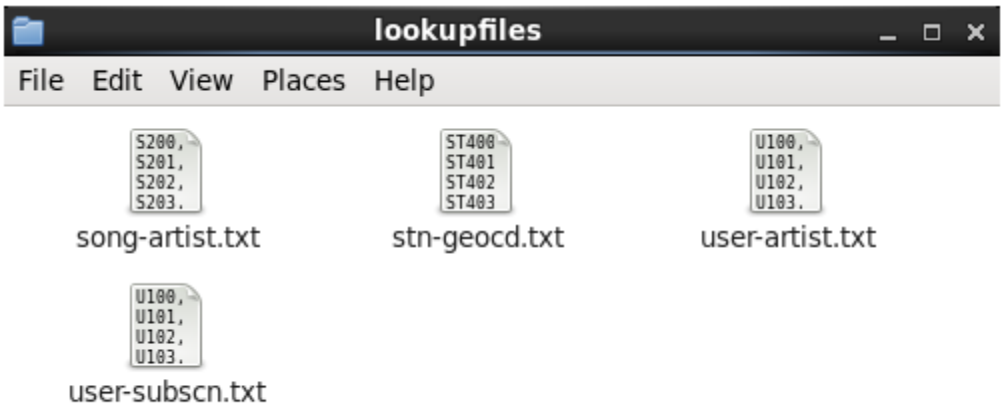
Below is the link for same.
https://drive.google.com/drive/folders/0B_P3pWagdIrrMjJGVlNsSUEtbG8?usp=sharing




data
File Edit View Places Help
mob
web


mob
File Edit View Places Help
U117,
U115,
U117,
U111.
file.txt


web
File Edit View Places Help
file.xml


lookupfiles
File Edit View Places Help
S200,
S201,
S202,
S203.
song-artist.txt
ST400
ST401
ST402
ST403
stn-geocd.txt
U100,
U101,
U102,
U103.
user-artist.txt
U100,
U101,
U102,
U103.
user-subscn.txt

1.4 Data Enrichment

Rules for data enrichment,

- If any of like or dislike is NULL or absent, consider it as 0.
- If fields like **Geo_cd** and **Artist_id** are NULL or absent, consult the lookup tables for fields **Station_id** and **Song_id** respectively to get the values of **Geo_cd** and **Artist_id**.
- If corresponding lookup entry is not found, consider that record to be invalid.

NULL or absent field	Look up Field	Look up Table(Table from which record can be updated)
Geo_cd	Station_id	Station_Geo_Map
Artist_id	Song_id	Song_Artist_Map

1.5 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.

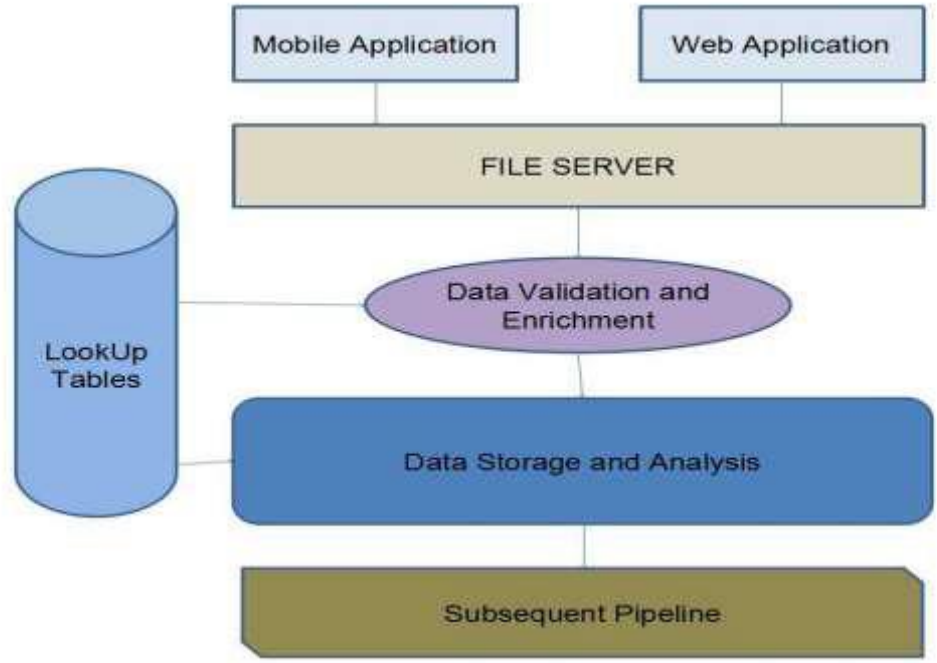
1. Determine top 10 station_id(s) where maximum number of songs were played, which were liked by unique users.
2. Determine total duration of songs played by each type of user, where type of user can be 'subscribed' or 'unsubscribed'. An unsubscribed user is the one whose record is either not present 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.

1.6 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 maintain to track the behavior and overcome failures in the pipeline.

1.7 Flow of operations

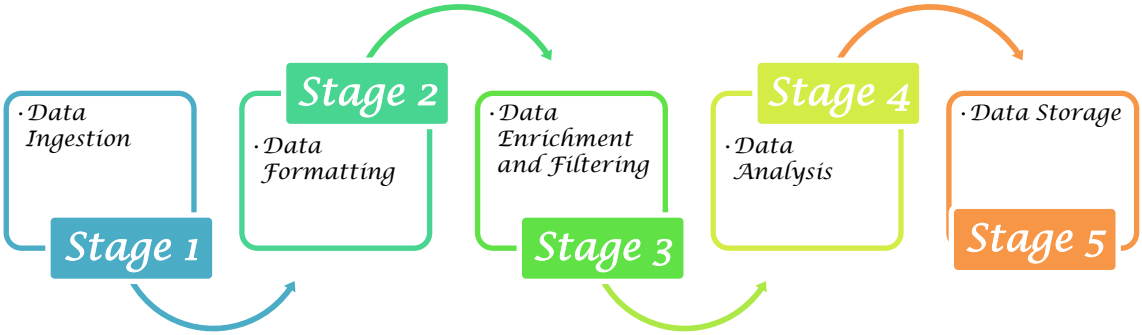
A schematic flow of operations is shown below,



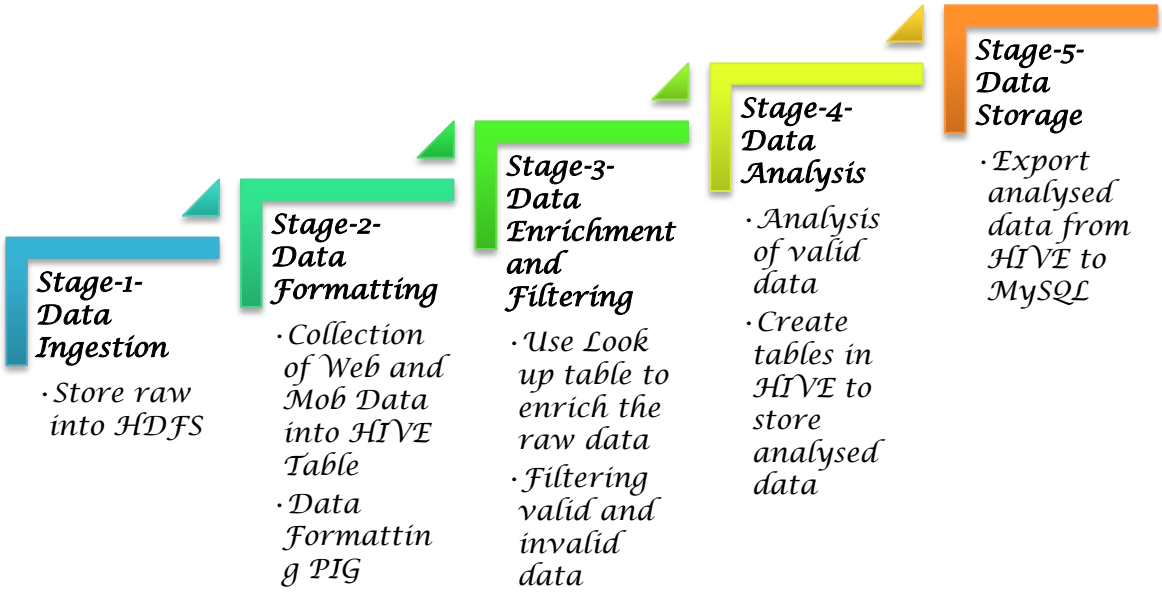
Section -2 – Design of the Project

2.1 Rough/Low Level Design

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



2.2 High Level Design



Section-3-Hadoop Eco-System Implementation

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

Batch file script,

start-daemon.sh | data_export.sh | DataAnalysis.sh | wrapper.sh | log_batch_1 | log_batch_2

```
#!/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

# To start the hadoop daemons:
/usr/local/hadoop-2.6.0/sbin/start-all.sh

# To start the Hmaster service:
/usr/local/hbase/bin/start-hbase.sh

# To start the Jobhistory server service:
mr-jobhistory-daemon.sh start historyserver

# To start mysql service:
sudo service mysqld start

# To start hive metastore:
hive --service metastore
```

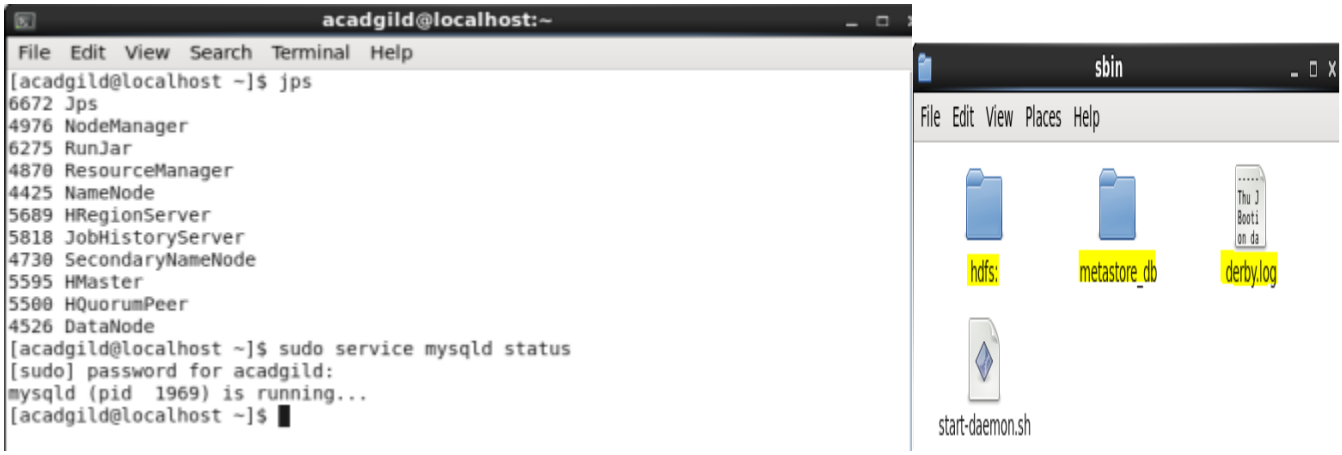
2. Starting all daemons,

- *sh start-daemon.sh*

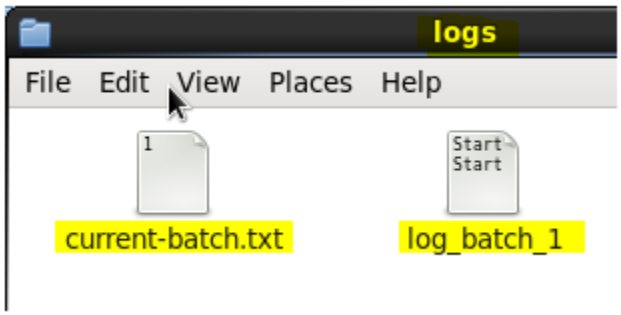
As per the batch file script all the hadoop daemons and the Hive, MySql and Hive daemons are started shown in the below screen shot,

```
acadgild@localhost:~/project/scripts
File  Edit  View  Search  Terminal  Help
[acadgild@localhost scripts]$ sh start-daemons.sh
Batch File Found!
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
18/03/05 17:20:03 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/acadgild/install/hadoop/hadoop-2.
6.5/logs/hadoop-acadgild-namenode-localhost.localdomain.out
localhost: starting datanode, logging to /home/acadgild/install/hadoop/hadoop-2.
6.5/logs/hadoop-acadgild-datanode-localhost.localdomain.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-secondaryname
node-localhost.localdomain.out
18/03/05 17:20:35 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
starting yarn daemons
starting resourcemanager, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/yarn-acadgild-resourcemanager-localhost.
localdomain.out
localhost: starting nodemanager, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/yarn-acadgild-nodemanager-localho
st.localdomain.out
localhost: starting zookeeper, logging to /home/acadgild/install/hbase/hbase-1.2.6/logs/hbase-acadgild-zookeeper-localhost.lo
caldomain.out
starting master, logging to /home/acadgild/install/hbase/hbase-1.2.6/logs/hbase-acadgild-master-localhost.localdomain.out
starting regionserver, logging to /home/acadgild/install/hbase/hbase-1.2.6/logs/hbase-acadgild-1-regionserver-localhost.local
domain.out
starting historyserver, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/mapred-acadgild-historyserver-localhost.lo
caldomain.out
[sudo] password for acadgild:
Sorry, try again.
[sudo] password for acadgild:
Starting mysqld:                                           [  OK  ]
2018-03-05 17:22:11: Starting Hive Metastore Server
/home/acadgild/install/hive/apache-hive-2.3.2-bin/bin/ext/metastore.sh: line 29: export: ` -Dproc_metastore  -Dlog4j.configur
ationFile=hive-log4j2.properties  -Djava.util.logging.config.file=/home/acadgild/install/hive/apache-hive-2.3.2-bin/conf/parq
uet-logging.properties  ': not a valid identifier
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/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!
```

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



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



Section-4 –Data Ingestion, Formatting, Enrichment and Filtering

4.1 Stage – 1 – Data Ingestion

By using the "**populate-lookup.sh**" script we will create lookup tables in **Hbase**. These tables have to be used in,
Data formatting,
Data enrichment and
Analysis stage

Lookup Tables

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

"populate-lookup.sh" script

The "populate-lookup.sh" shell script creates the above 4 lookup tables in the Hbase and populate the data into the lookup tables from the dataset files.
In the below screen shots, we can see the create-lookup.sh scripts and the following screen shots shows the tables creation and population of the data in the Hbase. Also,

the values loaded into the Hbase Tables are also shown, please see the below screen shots.

populate-lookup.sh

```
#!/bin/bash

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

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

echo "Creating LookUp Tables" >> $LOGFILE

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


echo "Populating LookUp Tables" >> $LOGFILE

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


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


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

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

```

Run the script: ./populate-lookup.sh

```
File  Edit  View  Search  Terminal  Help
[acadgild@localhost scripts]$ ./populate-lookup.sh
2018-03-05 20:23:19,516 WARN  [main] util.NativeCodeLoader: Unable to load nativ
e-hadoop library for your platform... using builtin-java classes where applicabl
e
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/s
lf4j-log4j12-1.7.5.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.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

create 'station-geo-map', 'geo'
0 row(s) in 3.4270 seconds

Hbase::Table - station-geo-map
2018-03-05 20:23:33,986 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using b
uiltin-java classes where applicable
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Static
LoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!
/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017
```

```
File  Edit  View  Search  Terminal  Help
create 'subscribed-users', 'subscn'
0 row(s) in 2.0260 seconds

Hbase::Table - subscribed-users
2018-03-05 20:23:56,656 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using b
uiltin-java classes where applicable
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Static
LoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!
/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

create 'song-artist-map', 'artist'
0 row(s) in 5.5420 seconds

Hbase::Table - song-artist-map
2018-03-05 20:24:19,037 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using b
uiltin-java classes where applicable
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Static
LoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!
/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017
```

```
put 'station-geo-map', 'ST400', 'geo:geo_cd', 'A'
0 row(s) in 0.9130 seconds

2018-03-05 20:24:35,222 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using b
uiltin-java classes where applicable
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Static
LoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!
/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

put 'station-geo-map', 'ST401', 'geo:geo_cd', 'AU'
0 row(s) in 0.8790 seconds

2018-03-05 20:24:51,542 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using b
uiltin-java classes where applicable
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Static
LoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!
/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

put 'station-geo-map', 'ST402', 'geo:geo_cd', 'AP'
0 row(s) in 0.9070 seconds

2018-03-05 20:25:08,054 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using b
uiltin-java classes where applicable
SLF4J: Class path contains multiple SLF4J bindings.
```

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

Lookup Tables in the hbase shell,

```
hbase(main):001:0> list
TABLE
clicks
song-artist-map
station-geo-map
studentAcad
subscribed-users
5 row(s) in 0.5390 seconds

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

The values loaded in the Lookup tables are shown below,

song-artist-map

```
hbase(main):002:0> scan 'song-artist-map'
ROW                                    COLUMN+CELL
 S200                                  column=artist:artistid, timestamp=1520261911286, value=A300
 S201                                  column=artist:artistid, timestamp=1520261927506, value=A301
 S202                                  column=artist:artistid, timestamp=1520261943257, value=A302
 S203                                  column=artist:artistid, timestamp=1520261959083, value=A303
 S204                                  column=artist:artistid, timestamp=1520261976127, value=A304
 S205                                  column=artist:artistid, timestamp=1520261991610, value=A301
 S206                                  column=artist:artistid, timestamp=1520262007398, value=A302
 S207                                  column=artist:artistid, timestamp=1520262024510, value=A303
 S208                                  column=artist:artistid, timestamp=1520262040377, value=A304
 S209                                  column=artist:artistid, timestamp=1520262055750, value=A305
10 row(s) in 0.4970 seconds

hbase(main):003:0>
```

station-geo-map

```
hbase(main):003:0> scan 'station-geo-map'
ROW                                    COLUMN+CELL
 ST400                                 column=geo:geo_cd, timestamp=1520261663743, value=A
 ST401                                 column=geo:geo_cd, timestamp=1520261679587, value=AU
 ST402                                 column=geo:geo_cd, timestamp=1520261696381, value=AP
 ST403                                 column=geo:geo_cd, timestamp=1520261712710, value=J
 ST404                                 column=geo:geo_cd, timestamp=1520261729609, value=E
 ST405                                 column=geo:geo_cd, timestamp=1520261745302, value=A
 ST406                                 column=geo:geo_cd, timestamp=1520261762950, value=AU
 ST407                                 column=geo:geo_cd, timestamp=1520261780201, value=AP
 ST408                                 column=geo:geo_cd, timestamp=1520261796768, value=E
 ST409                                 column=geo:geo_cd, timestamp=1520261813316, value=E
 ST410                                 column=geo:geo_cd, timestamp=1520261830471, value=A
 ST411                                 column=geo:geo_cd, timestamp=1520261846217, value=A
 ST412                                 column=geo:geo_cd, timestamp=1520261862431, value=AP
 ST413                                 column=geo:geo_cd, timestamp=1520261878344, value=J
 ST414                                 column=geo:geo_cd, timestamp=1520261895922, value=E
15 row(s) in 0.2530 seconds
```

subscribed-users

```
hbase(main):004:0> scan 'subscribed-users'
ROW                                    COLUMN+CELL
 U100                                  column=subscn:enddt, timestamp=1520262087042, value=1465130523
 U100                                  column=subscn:startdt, timestamp=1520262071579, value=1465230523
 U101                                  column=subscn:enddt, timestamp=1520262118498, value=1475130523
 U101                                  column=subscn:startdt, timestamp=1520262102826, value=1465230523
 U102                                  column=subscn:enddt, timestamp=1520262149727, value=1475130523
 U102                                  column=subscn:startdt, timestamp=1520262134055, value=1465230523
 U103                                  column=subscn:enddt, timestamp=1520262181725, value=1475130523
 U103                                  column=subscn:startdt, timestamp=1520262166544, value=1465230523
 U104                                  column=subscn:enddt, timestamp=1520262213739, value=1475130523
 U104                                  column=subscn:startdt, timestamp=1520262197974, value=1465230523
 U105                                  column=subscn:enddt, timestamp=1520262245188, value=1475130523
 U105                                  column=subscn:startdt, timestamp=1520262229388, value=1465230523
 U106                                  column=subscn:enddt, timestamp=1520262276542, value=1485130523
 U106                                  column=subscn:startdt, timestamp=1520262260792, value=1465230523
 U107                                  column=subscn:enddt, timestamp=1520262308694, value=1455130523
 U107                                  column=subscn:startdt, timestamp=1520262293070, value=1465230523
 U108                                  column=subscn:enddt, timestamp=1520262340310, value=1465230623
 U108                                  column=subscn:startdt, timestamp=1520262324074, value=1465230523
 U109                                  column=subscn:enddt, timestamp=1520262374828, value=1475130523
 U109                                  column=subscn:startdt, timestamp=1520262358062, value=1465230523
 U110                                  column=subscn:enddt, timestamp=1520262405605, value=1475130523
 U110                                  column=subscn:startdt, timestamp=1520262390207, value=1465230523
 U111                                  column=subscn:enddt, timestamp=1520262436440, value=1475130523
 U111                                  column=subscn:startdt, timestamp=1520262420983, value=1465230523
 U112                                  column=subscn:enddt, timestamp=1520262467520, value=1475130523
 U112                                  column=subscn:startdt, timestamp=1520262452064, value=1465230523
 U113                                  column=subscn:enddt, timestamp=1520262498104, value=1485130523
 U113                                  column=subscn:startdt, timestamp=1520262482841, value=1465230523
 U114                                  column=subscn:enddt, timestamp=1520262530025, value=1468130523
 U114                                  column=subscn:startdt, timestamp=1520262514364, value=1465230523
15 row(s) in 0.3530 seconds

hbase(main):005:0>
```

We have successfully created the lookup tables in the Hbase.

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

```
hive> show databases;
OK
default
emp_details
project
Time taken: 12.558 seconds, Fetched: 3 row(s)
hive> use project;
OK
Time taken: 0.099 seconds
hive> show tables;
OK
users_artists
Time taken: 0.177 seconds, Fetched: 1 row(s)
hive> select * from users_artists;
OK
U100    ["A300","A301","A302"]
U101    ["A301","A302"]
U102    ["A302"]
U103    ["A303","A301","A302"]
U104    ["A304","A301"]
U105    ["A305","A301","A302"]
U106    ["A301","A302"]
U107    ["A302"]
U108    ["A300","A303","A304"]
U109    ["A301","A303"]
U110    ["A302","A301"]
U111    ["A303","A301"]
U112    ["A304","A301"]
U113    ["A305","A302"]
U114    ["A300","A301","A302"]
Time taken: 5.316 seconds, Fetched: 15 row(s)
hive>
```

*Now we need to link theses lookup tables in hive using the Hbase Storage Handler. With the help of **"data_enrichment_filtering_schema.sh"** file we will create hive tables on the top of Hbase tables using **"create_hive_hbase_lookup.hql"**.*

Creating Hive Tables on the top of Hbase:

In this section with the help of Hbase storage handler & SerDe properties we are creating the hive external tables by matching the columns of Hbase tables to hive tables.

Run the script: ./data_enrichment_filtering_schema.sh,

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

```bash
#!/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

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

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

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

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

```
acadgild@localhost:~/project/scripts
File  Edit  View  Search  Terminal  Help
[acadgild@localhost scripts]$ ./data_enrichment_filtering_schema.sh
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/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!
/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.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.j
ar!/hive-log4j2.properties Async: true
OK
Time taken: 13.269 seconds
OK
Time taken: 6.838 seconds
OK
Time taken: 0.518 seconds
OK
Time taken: 0.438 seconds
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost scripts]$
```

Check in HIVE

```
hive> use project
    > ;
OK
Time taken: 10.119 seconds
hive> show tables;
OK
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 0.474 seconds, Fetched: 4 row(s)
```

```
hive> select * from users_artists;
OK
U100    ["A300","A301","A302"]
U101    ["A301","A302"]
U102    ["A302"]
U103    ["A303","A301","A302"]
U104    ["A304","A301"]
U105    ["A305","A301","A302"]
U106    ["A301","A302"]
U107    ["A302"]
U108    ["A300","A303","A304"]
U109    ["A301","A303"]
U110    ["A302","A301"]
U111    ["A303","A301"]
U112    ["A304","A301"]
U113    ["A305","A302"]
U114    ["A300","A301","A302"]
Time taken: 4.23 seconds, Fetched: 15 row(s)
hive>
```

```
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: 1.056 seconds, Fetched: 10 row(s)
hive>
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: 0.826 seconds, Fetched: 15 row(s)
hive>
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: 0.813 seconds, Fetched: 15 row(s)
hive>
```

We can see from above pictures all the tables are created and inserted with data.

4.2 Stage – 2 - 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: ./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
```

```
acadgild@localhost:~/project
File  Edit  View  Search  Terminal  Help
[acadgild@localhost project]$ sh /home/acadgild/project/scripts/dataformatting.sh
18/03/08 21:55:38 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
rm: `/user/acadgild/project/batch1/web/': No such file or directory
18/03/08 21:55:44 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
rm: `/user/acadgild/project/batch1/formattedweb/': No such file or directory
18/03/08 21:55:49 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
rm: `/user/acadgild/project/batch1/mob/': No such file or directory
18/03/08 21:55:53 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
18/03/08 21:55:59 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
18/03/08 21:56:05 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
18/03/08 21:56:11 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
18/03/08 21:56:20 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
18/03/08 21:56:20 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
18/03/08 21:56:20 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2018-03-08 21:56:20,730 [main] INFO  org.apache.pig.Main - Apache Pig version 0.16.0 (r1746530) compiled Jun 01 2016, 23:10:4
9
2018-03-08 21:56:20,731 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/acadgild/project/pig_152052638072
6.log
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!
/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Static
LoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2018-03-08 21:56:21,894 [main] WARN  org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop library for your
platform... using builtin-java classes where applicable
2018-03-08 21:56:22,549 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file /home/acadgild/.pigbootup not found
2018-03-08 21:56:22,948 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Ins
tead, use mapreduce.jobtracker.address
2018-03-08 21:56:22,948 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instea
d, use fs.defaultFS
```

We are running two scripts to format the data. They are:
Dataformatting.pig
Formatted_hive_load.hql

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

Dataformatting.pig

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

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

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

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

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

formatted_hive_load.hql

```
USE project;

CREATE TABLE IF NOT EXISTS formatted_input
(
user_id STRING,
song_id STRING,
artist_id STRING,
timestp STRING,
start_ts STRING,
end_ts STRING,
geo_cd STRING,
station_id STRING,
song_end_type INT,
like INT,
dislike INT
)
PARTITIONED BY
(batchid INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

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

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

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

Pig script successful completion,

```
acadgild@localhost:~/project
File  Edit  View  Search  Terminal  Help

HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
2.6.5   0.16.0  acadgild        2018-03-08 21:56:30     2018-03-08 21:59:07     UNKNOWN

Success!

Job Stats (time in seconds):
JobId   Maps    Reduces MaxMapTime      MinMapTime      AvgMapTime      MedianMapTime   MaxReduceTime   MinReduceTime   AvgRe
duceTime        MedianReducetime        Alias   Feature Outputs
job_1520520526738_0001  1       0       34      34      34      34      0       0       0       0       A,B     MAP_ONLY    /
user/acadgild/project/batch1/formattedweb,

Input(s):
Successfully read 20 records (7105 bytes) from: "/user/acadgild/project/batch1/web"

Output(s):
Successfully stored 20 records (1235 bytes) in: "/user/acadgild/project/batch1/formattedweb"

Counters:
Total records written : 20
Total bytes written : 1235
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1520520526738_0001
```

Hive script successfully load the data into hive terminal,

```
Logging initialized using configuration in jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hive-common-2.3.2.j
ar!/hive-log4j2.properties Async: true
OK
Time taken: 15.758 seconds
OK
Time taken: 1.257 seconds
Loading data to table project.formatted_input partition (batchid=1)
OK
Time taken: 4.585 seconds
Loading data to table project.formatted_input partition (batchid=1)
OK
Time taken: 2.562 seconds
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost project]$
```

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

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

```
[acadgild@localhost ~]$ hadoop fs -ls /user/acadgild/project/batch1/formattedweb/
18/03/11 23:05:54 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 2 items
-rw-r--r--   1 acadgild supergroup          0 2018-03-08 17:07 /user/acadgild/project/batch1/formattedweb/_SUCCESS
-rw-r--r--   1 acadgild supergroup       1235 2018-03-08 17:07 /user/acadgild/project/batch1/formattedweb/part-m-00000
You have new mail in /var/spool/mail/acadgild
```

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

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

```
drwxr-xr-x   - acadgild supergroup          0 2018-03-08 17:06 /user/acadgild/project/batch1/web
[acadgild@localhost ~]$ hadoop fs -cat /user/acadgild/project/batch1/formattedweb/*
18/03/11 22:35:37 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
U104,S202,A301,1494297562,1465490556,1465490556,AU,ST404,1,1,0
U113,S205,A301,1468094889,1462863262,1468094889,E,ST414,3,0,0
U106,S209,A300,1462863262,1468094889,1465490556,AU,ST413,1,0,1
U117,S208,A300,1465490556,1494297562,1468094889,U,ST402,1,0,0
U116,S201,A304,1468094889,1465490556,1465490556,E,ST407,3,0,0
,S201,A304,1494297562,1465490556,1465490556,U,ST405,0,1,1
U102,S209,A301,1494297562,1468094889,1494297562,E,ST410,0,1,0
U102,S204,A305,1462863262,1462863262,1462863262,A,ST403,1,0,0
U112,S203,A304,1468094889,1462863262,1468094889,,ST402,1,1,0
U108,S210,,1462863262,1468094889,1494297562,U,ST401,0,0,1
U103,S201,A300,1494297562,1494297562,1494297562,U,ST405,1,1,0
U111,S205,A300,1462863262,1462863262,1494297562,E,ST407,1,0,0
U102,S209,A304,1468094889,1468094889,1468094889,E,ST414,3,0,1
U117,S207,A304,1468094889,1465490556,1494297562,A,ST409,2,1,0
U105,S201,A304,1465490556,1465490556,1494297562,A,ST412,1,0,1
U119,S204,A303,1462863262,1462863262,1462863262,A,ST415,0,0,0
U113,S209,A303,1465490556,1462863262,1494297562,A,ST400,1,0,1
U120,S202,A303,1465490556,1462863262,1465490556,A,ST415,2,1,0
U105,S207,A301,1462863262,1465490556,1462863262,AP,ST414,1,0,1
U115,S201,A303,1465490556,1465490556,1468094889,AU,ST413,0,0,1
[acadgild@localhost ~]$
```

The new Tables has been created and show below,

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

DataFormatting.sh output in hive terminal,

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

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

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

4.3 Stage – 3 - Data Enrichment & Filtering

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

Rules for data enrichment,

1. If any of like or dislike is **NULL or absent**, consider it as 0.
2. If fields like **Geo_cd** and **Artist_id** are **NULL or absent**, consult the lookup tables for fields **Station_id** and **Song_id** respectively to get the values of **Geo_cd** and **Artist_id**.
3. If corresponding lookup entry is not found, consider that record to be invalid

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

data_enrichment.sh

```
#!/bin/bash

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

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

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

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

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

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

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

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

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

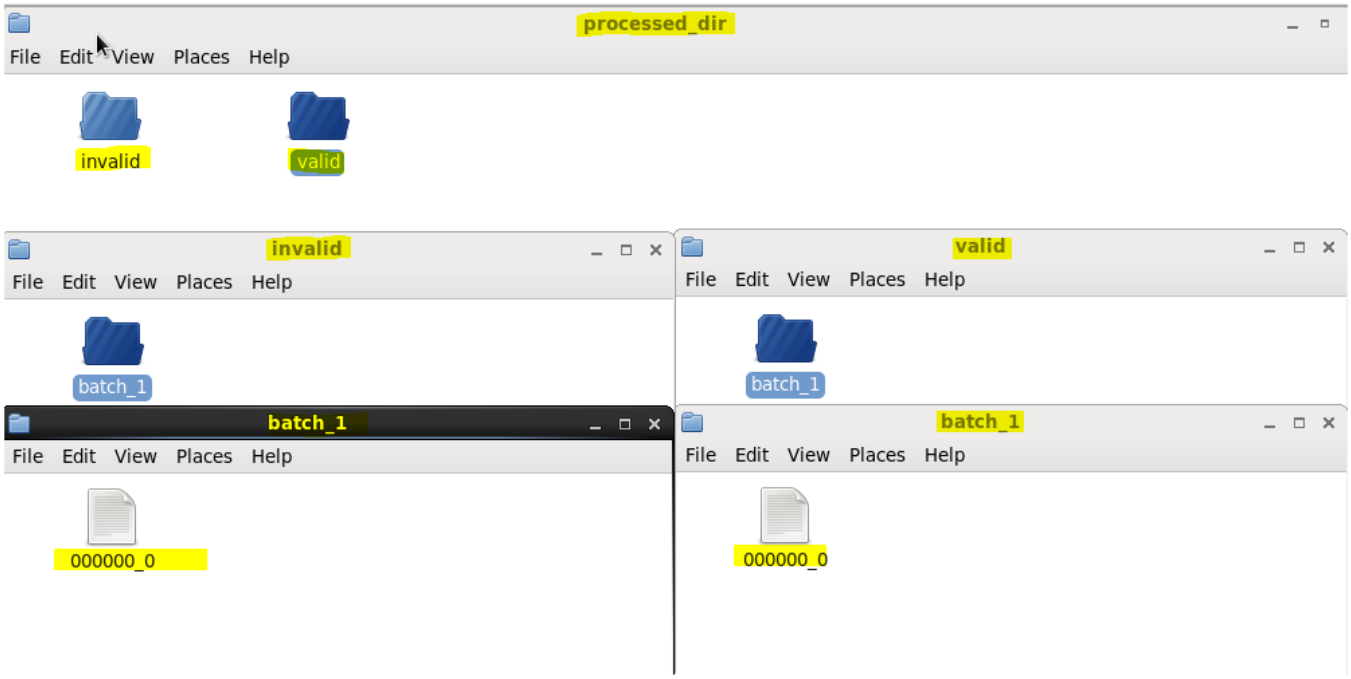
data_enrichment.hql

```
USE project;

CREATE TABLE IF NOT EXISTS enriched_data
(
user_id STRING,
song_id STRING,
artist_id STRING,
timestp STRING,
start_ts STRING,
end_ts STRING,
geo_cd STRING,
station_id STRING,
song_end_type INT,
like INT,
dislike INT
)
PARTITIONED BY
(batchid INT,
status STRING)
STORED AS ORC;

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

```
acadgild@localhost:~/project
File  Edit  View  Search  Terminal  Help
[acadgild@localhost project]$ sh /home/acadgild/project/scripts/data_enrichment.sh
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/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!
/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.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.j
ar!/hive-log4j2.properties Async: true
OK
Time taken: 79.575 seconds
OK
Time taken: 3.131 seconds
No Stats for project@formatted_input, Columns: start_ts, song_id, like_id, time_stamp, user_id, end_ts, dislike, station_id,
geo_cd, song_end_type
No Stats for project@station_geo_map, Columns: station_id, geo_cd
No Stats for project@song_artist_map, Columns: song_id, artist_id
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_20180309005812_138a1f4a-ea40-4da6-ab6d-dda87704d057
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_1520520526738_0003, Tracking URL = http://localhost:8088/proxy/application_1520520526738_0003/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1520520526738_0003
Hadoop job information for Stage-1: number of mappers: 3; number of reducers: 1
2018-03-09 01:00:08,431 Stage-1 map = 0%,  reduce = 0%
2018-03-09 01:01:08,679 Stage-1 map = 0%,  reduce = 0%
2018-03-09 01:02:09,607 Stage-1 map = 0%,  reduce = 0%
2018-03-09 01:02:29,011 Stage-1 map = 67%,  reduce = 0%, Cumulative CPU 7.3 sec
2018-03-09 01:03:12,485 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 12.34 sec
```

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

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



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

*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> show tables;
OK
enriched_data
formatted_input
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 0.263 seconds, Fetched: 6 row(s)
hive> select * from enriched_data;
OK
        S200    A300    1494297562      1465490556      1462863262      J       ST413   3       0       1       1       fail
U103    S202    A302    1494297562      1462863262      1494297562      A       ST410   3       1       1       1       fail
U115    S202    A302    1468094889      1462863262      1468094889      AP      ST402   1       0       0       1       fail
U120    S203    A303    1465490556      1468094889      1494297562      NULL    ST415   0       0       0       1       fail
U117    S203    A303    1465230523      1485130523      1485130523      NULL    ST415   1       1       0       1       fail
U115    S203    A303    1468094889      1468094889      1465490556      A       ST410   0       1       1       1       fail
U113    S204    A304    1462863262      1462863262      1462863262      J       ST413   3       1       1       1       fail
U116    S205    A301    1465490556      1494297562      1494297562      NULL    ST415   3       1       1       1       fail
U110    S205    A301    1465130523      1485130523      1465130523      A       ST411   2       1       1       1       fail
U105    S205    A301    1475130523      1475130523      1465230523      AP      ST407   2       1       1       1       fail
U108    S206    A302    1494297562      1462863262      1465490556      J       ST403   3       1       1       1       fail
        S208    A304    1465230523      1465230523      1485130523      A       ST410   2       1       1       1       fail
U114    S210    NULL    1475130523      1485130523      1485130523      E       ST409   3       1       0       1       fail
U117    S210    NULL    1475130523      1465130523      1475130523      AP      ST407   0       1       0       1       fail
U117    S210    NULL    1465490556      1462863262      1465490556      E       ST414   0       0       0       1       fail
U104    S210    NULL    1494297562      1468094889      1462863262      J       ST413   2       1       1       1       fail
U112    S210    NULL    1495130523      1475130523      1485130523      A       ST411   3       1       1       1       fail
U103    S210    NULL    1494297562      1462863262      1465490556      E       ST408   0       0       1       1       fail
U103    S210    NULL    1475130523      1475130523      1475130523      E       ST414   1       0       1       1       fail
U108    S200    A300    1462863262      1465490556      1462863262      AP      ST402   3       1       0       1       pass
U119    S200    A300    1462863262      1468094889      1465490556      E       ST414   2       0       1       1       pass
U109    S200    A300    1465230523      1465130523      1485130523      AU      ST406   1       0       1       1       pass
U100    S200    A300    1475130523      1475130523      1465130523      A       ST400   3       1       0       1       pass
U112    S201    A301    1494297562      1465490556      1468094889      E       ST408   0       0       0       1       pass
U103    S201    A301    1475130523      1465130523      1485130523      E       ST409   3       0       0       1       pass
U113    S202    A302    1465230523      1465230523      1465230523      A       ST410   3       0       0       1       pass
U113    S202    A302    1465230523      1465130523      1485130523      AU      ST401   1       0       0       1       pass
U115    S203    A303    1495130523      1465230523      1465230523      E       ST404   3       0       1       1       pass
```

4.4 Stage – 4 - Data Analysis using Spark

In this stage we will do analysis on enriched data using Spark SQL and run the program using Spark Submit command.
Before running the spark-submit command we have to zip -d command to remove the bad manifests in created spark project jar file to avoid the invalid Signature exception.
We used two spark-submits for analysis.
a. Spark_analysis for creating tables for each query/problem statement.
b. Spark_analysis_2 for displaying results for each query in terminal.

DataAnalysis.sh

```
#!/bin/bash

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

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

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

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

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

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

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

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

Spark_analysis.scala

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

object Spark_analysis {

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

    val batchId = args(0)

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

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

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

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

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

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

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

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


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

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



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


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

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

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


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

val create_hive_table_top_10_unsubscribed_users = sparkSession.sqlContext.sql("CREATE TABLE IF NOT EXISTS project.top_10_unsubscribed
  "("+
  " user_id STRING,"+
  " song_id STRING,"+
  " artist_id STRING,"+
  " total_duration_in_minutes DOUBLE"+
  ")"+
  " PARTITIONED BY (batchid INT)"+
  " ROW FORMAT DELIMITED"+
  " FIELDS TERMINATED BY ','"+
  " STORED AS TEXTFILE")


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

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


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

Spark_analysis_2.scala

```
package sparkanalysis

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

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

```
acadgild@localhost:~/project
File  Edit  View  Search  Terminal  Help
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost project]$ sh /home/acadgild/project/scripts/data_analysis.sh
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/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!
/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.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.j
ar!/hive-log4j2.properties Async: true
OK
Time taken: 255.355 seconds
OK
Time taken: 8.952 seconds
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_20180309212015_092eede3-90d5-4b83-8c5a-23602e9885b5
Total jobs = 2
Launching Job 1 out of 2
```

```
MapReduce Total cumulative CPU time: 11 seconds 630 msec
Ended Job = job_1520520526738_0006
Loading data to table project.top_10_stations partition (batchid=1)
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 9.62 sec   HDFS Read: 12914 HDFS Write: 271 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 11.63 sec   HDFS Read: 7452 HDFS Write: 159 SUCCESS
Total MapReduce CPU Time Spent: 21 seconds 250 msec
OK
Time taken: 698.648 seconds
OK
Time taken: 0.373 seconds
2018-03-09 21:38:15,040 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 7.04 sec
MapReduce Total cumulative CPU time: 7 seconds 40 msec
Ended Job = job_1520520526738_0008
Loading data to table project.users_behaviour partition (batchid=1)
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 1   Cumulative CPU: 10.09 sec   HDFS Read: 36076 HDFS Write: 166 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 7.04 sec   HDFS Read: 6751 HDFS Write: 133 SUCCESS
Total MapReduce CPU Time Spent: 23 seconds 730 msec
OK
Time taken: 384.024 seconds
OK
Time taken: 0.612 seconds
MapReduce Total cumulative CPU time: 6 seconds 280 msec
Ended Job = job_1520520526738_0011
Loading data to table project.connected_artists partition (batchid=1)
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 1   Cumulative CPU: 13.88 sec   HDFS Read: 26295 HDFS Write: 292 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 5.07 sec   HDFS Read: 5263 HDFS Write: 165 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 6.28 sec   HDFS Read: 6632 HDFS Write: 112 SUCCESS
Total MapReduce CPU Time Spent: 25 seconds 230 msec
OK
Time taken: 410.307 seconds
OK
Time taken: 0.369 seconds
```

```
2018-03-09 21:47:37,807 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 7.66 sec
MapReduce Total cumulative CPU time: 7 seconds 660 msec
Ended Job = job_1520520526738_0013
Loading data to table project.top_10_royalty_songs partition (batchid=1)
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 9.48 sec   HDFS Read: 13556 HDFS Write: 289 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 7.66 sec   HDFS Read: 6917 HDFS Write: 188 SUCCESS
Total MapReduce CPU Time Spent: 17 seconds 140 msec
OK
Time taken: 207.263 seconds
OK
Time taken: 0.342 seconds
```

The tables have also been created in the Hive,

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

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

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

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

```
Time taken: 0.237 seconds, Fetched: 8 row(s)
hive> Select * From top_10_stations;
OK
top_10_stations.station_id      top_10_stations.total_distinct_songs_played     top_10_stations.distinct_user_count     top_10_stations.batchid
ST407   2       3       1
ST414   1       1       1
ST411   1       1       1
ST402   1       2       1
ST406   1       1       1
ST405   1       1       1
Time taken: 0.336 seconds, Fetched: 6 row(s)
```

***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> Select * From users_behaviour;
OK
users_behaviour.user_type       users_behaviour.duration        users_behaviour.batchid
SUBSCRIBED      93861594        1
UNSUBSCRIBED    105594881       1
Time taken: 0.274 seconds, Fetched: 2 row(s)
hive>
    >
    >
```

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

```
Time taken: 0.097 seconds, Fetched: 11 row(s)
hive> Select * From connected_artists;
OK
connected_artists.artist_id     connected_artists.user_count    connected_artists.batchid
A303    2       1
A302    2       1
A300    1       1
Time taken: 0.225 seconds, Fetched: 3 row(s)
```

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> Select * From top_10_royalty_songs;
OK
top_10_royalty_songs.song_id    top_10_royalty_songs.duration   top_10_royalty_songs.batchid
S208    22627294        1
S207    20000000        1
S206    19900000        1
S209    15254588        1
S200    9900000 1
S204    2604333 1
S202    100000  1
S205    0       1
Time taken: 0.237 seconds, Fetched: 8 row(s)
```

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

```
hive> Select * From top_10_unsubscribed_users;
OK
top_10_unsubscribed_users.user_id       top_10_unsubscribed_users.duration      top_10_unsubscribed_users.batchid
U117    20000000        1
U118    20000000        1
U110    20000000        1
U120    12627294        1
U115    12527294        1
U107    10000000        1
U108    5231627 1
U109    2604333 1
U106    2604333 1
U100    0       1
Time taken: 0.275 seconds, Fetched: 10 row(s)
```

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

4.5 Stage – 5 – Data Storage in MYSQL

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

data_export.sh

```bash
#!/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 -p Root@123 < /home/acadgild/project/scripts/create_schema.sql

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

sqoop export -m 1 --connect jdbc:mysql://localhost/project --username 'root' --password 'Root@123' --table top_10_stations
--export-dir hdfs://localhost:9000/user/hive/warehouse/project.db/top_10_stations/batchid=$batchid --input-fields-terminated-by ',' \


sqoop export  -m 1 --connect jdbc:mysql://localhost/project --username 'root' --password 'Root@123' --table users_behaviour
--export-dir hdfs://localhost:9000/user/hive/warehouse/project.db/users_behaviour/batchid=$batchid --input-fields-terminated-by ',' \


sqoop export -m 1 --connect jdbc:mysql://localhost/project --username 'root' --password 'Root@123' --table connected_artists
--export-dir hdfs://localhost:9000/user/hive/warehouse/project.db/connected_artists/batchid=$batchid --input-fields-terminated-by ',' \


sqoop export -m 1 --connect jdbc:mysql://localhost/project --username 'root' --password 'Root@123' --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 ',' \


sqoop export -m 1 --connect jdbc:mysql://localhost/project --username 'root' --password 'Root@123' --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 ',' \

```

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

```sql
CREATE DATABASE IF NOT EXISTS project;

USE project;

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

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

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

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

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

commit;
```

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

```
[acadgild@localhost ~]$ sqoop export --connect jdbc:mysql://localhost/project --username root --password acadgild --table top 10 stations --ex
ort-dir hdfs://localhost:9000/user/hive/warehouse/project.db/top_10_stations/batchid=1 --input-fields-terminated-by ',' -m 1
Warning: /home/acadgild/sqoop-1.4.6.bin__hadoop-2.0.4-alpha/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /home/acadgild/sqoop-1.4.6.bin__hadoop-2.0.4-alpha/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /home/acadgild/sqoop-1.4.6.bin__hadoop-2.0.4-alpha/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
```

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

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

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

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

top_10_royalty_songs,

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

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***

```
[acadgild@localhost logs]$ cat current-batch.txt
2[acadgild@localhost logs]$
[acadgild@localhost logs]$
[acadgild@localhost logs]$
```

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

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

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

wrapper.sh

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

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

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

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

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

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

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

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

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

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

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

```
#do this for every 3 hours
* */3 * * * date>>/home/acadgild/project/scripts/wrapper.sh >> /home/acadgild/project/scripts/jobsheduling.log
:wq!
```

```
-bash: cd: crontab: No such file or directory
[acadgild@localhost logs]$ crontab -e
no crontab for acadgild - using an empty one
crontab: installing new crontab
[acadgild@localhost logs]$
```

*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 ~]$
[acadgild@localhost ~]$
```

Highlights of the Project

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

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*