ACAD**GILD**

MUSIC
DATA ANALYSIS USING HADOOP

Project Implementation Document

Sr. No.	Version	Created Date	Changed Date	Author
1	Initial Version	07/22/2018	07/22/2018	Ankith M

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 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

1.2 LookUp Tables

There is 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

1.3 Data Ingestion and Initial Validation

Below is the link for datasets.

https://drive.google.com/drive/folders/0B_P3pWagdIrrMjJGVlNsSUEtbG8

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 files come every 3 hours.
4. All the timestamp fields in data coming from web application is of the format YYYY-MM-DD HH:MM:SS.
5. All the timestamp fields in data coming from mobile application is a long integer interpreted as UNIX timestamps.
6. Finally, all timestamps must have the format of a long integer to be interpreted as UNIX timestamps.
7. If both like and dislike are 1, consider that record to be invalid.
8. If any of the fields from User_id, Song_id, Timestamp, Start_ts, End_ts, Geo_cd is NULL or absent, consider that record to be invalid.
9. If Song_end_type is NULL or absent, treat it to be 3.
10. Create a temporary identfier for all the data files received in the last 3 hours (may be an integer batch_id which is auto incremented or a string obtained after combining current date and current hour, to keep track of valid and invalid records per batch).

1.4 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

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 presents 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 that 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 behaviour and overcome failures in the pipeline.

1.7 Flow of Operations

The diagram shows the basic steps of how music data will be processed on different stages.

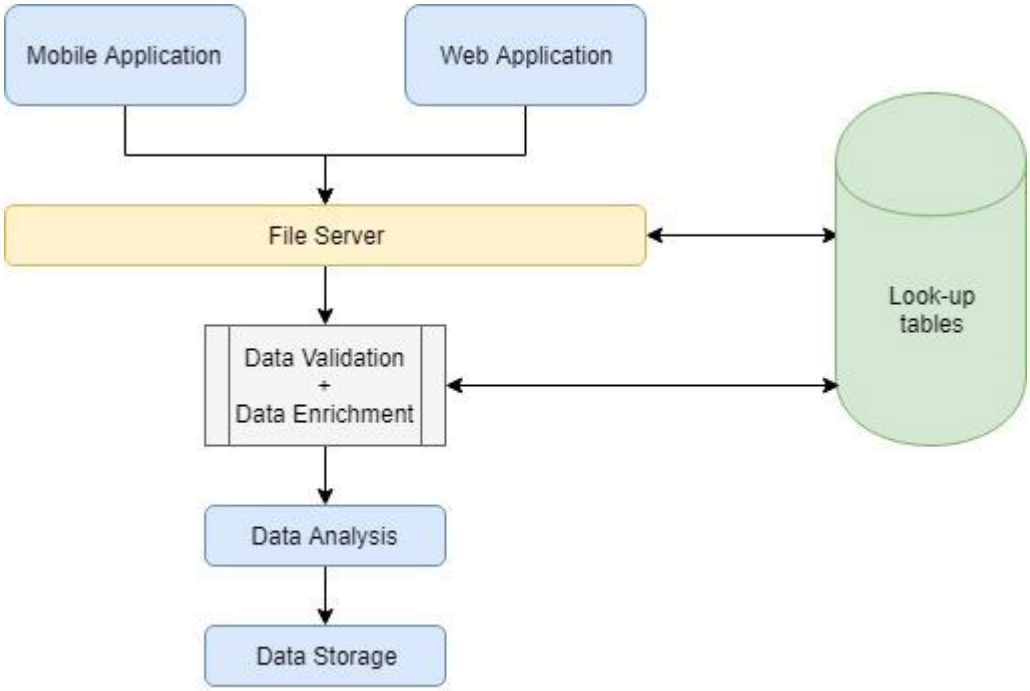


Fig -1 High Level Flow Music Data Analysis

2 Hadoop Implementation

We have a master file which executes each phase of the project which will be finally scheduled to be executed in every three hours when there is a data ingestion done for this analysis.

* music_project_m...

```
# Create data
echo "Preparing to execute python scripts to generate data..."
rm -r /home/acadgild/examples/music/data/web
rm -r /home/acadgild/examples/music/data/mob
mkdir -p /home/acadgild/examples/music/data/web
mkdir -p /home/acadgild/examples/music/data/mob
python /home/acadgild/examples/music/generate_web_data.py
python /home/acadgild/examples/music/generate_mob_data.py
echo "Data Generated Successfully !"

# Call Stop start daemon scripts to start hadoop daemons
echo "Starting the daemons...."
sh start-daemons.sh
# run jps commands to check the daemons
jps
echo "All hadoop daemons started !"

echo "Upload the look up tables now in Hbase..."
#sh populate-lookup.sh
echo "Done with data population in look up tables !"

echo "Creating hive tables on top of hbase tables for data enrichment and filtering..."
#sh data_enrichment_filtering_schema.sh
echo "Hive table with Hbase Mapping Complete !"

echo "Lets do some data formatting now...."
#sh dataformatting.sh
echo "data formatting complete !"

echo "Let us do data enrichment as per the requirement..."
#sh data_enrichment.sh
echo "Data Enrichment Complete"

echo "Lets run some use cases now..."
#sh data_analysis.sh
echo "USE CASES COMPLETE !!"
```

1. Initially we could see that the data is generated using the Python files – this step generates the input files for both web applications (file.xml) and mobile applications (file.txt).
   Web Application input file –

file.xml

```xml
<records>
<record>
<user_id>U104</user_id>
<song_id>S205</song_id>
<artist_id>A304</artist_id>
<timestamp>2016-06-09 22:12:36</timestamp>
<start_ts>2016-05-10 12:24:22</start_ts>
<end_ts>2016-06-09 22:12:36</end_ts>
<geo_cd>A</geo_cd>
<station_id>ST403</station_id>
<song_end_type>3</song_end_type>
<like>1</like>
<dislike>1</dislike>
</record>
```

Mobile Application Input file –

file.txt

```
U114,S202,A300,1465230523,1465130523,1475130523,E,ST405,1,1,1
U114,S209,A305,1465230523,1485130523,1485130523,AP,ST415,0,1,0
U107,S200,A300,1495130523,1465230523,1465130523,AP,ST409,0,1,1
U104,S205,A305,1465230523,1475130523,1465230523,U,ST414,0,0,0
U107,S200,A304,1465130523,1485130523,1475130523,AP,ST404,0,1,0
,S202,A303,1465230523,1475130523,1465130523,AP,ST404,0,0,1
```

2. Once the input files are ready, start all the hadoop daemons using the **start-daemons.sh** file.

start-daemons.sh

```
#!/bin/bash

rm -r /home/acadgild/examples/music/logs
mkdir -p /home/acadgild/examples/music/logs

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

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

echo "Starting daemons" >> $LOGFILE

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

cat /home/acadgild/examples/music/logs/current-batch.txt
```

```
[acadgild@localhost music]$ ./music_project_master.sh
Preparing to execute python scripts to generate data...
Data Generated Successfully !
Starting the daemons....
After chmod
After batchid-->> 1
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
18/07/22 19:53:20 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes wh
ere applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-namenode-localhost.localdoma
in.out
localhost: starting datanode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-datanode-localhost.localdoma
in.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-secondarynamenode-loc
alhost.localdomain.out
18/07/22 19:53:39 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes wh
ere applicable
starting yarn daemons
starting resourcemanager, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/yarn-acadgild-resourcemanager-localhost.localdom
ain.out
localhost: starting nodemanager, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/yarn-acadgild-nodemanager-localhost.local
domain.out
localhost: starting zookeeper, logging to /home/acadgild/install/hbase/hbase-1.2.6/logs/hbase-acadgild-zookeeper-localhost.localdomai
n.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.localdomain.o
ut
starting historyserver, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/mapred-acadgild-historyserver-localhost.localdomai
n.out
```

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.

3. We can see the list of active daemons running in the machine.

```
118769 NodeManager
19588 JobHistoryServer
19494 HRegionServer
19385 HMaster
19290 HQuorumPeer
18363 DataNode
18668 ResourceManager
18525 SecondaryNameNode
18238 NameNode
19647 Jps
All hadoop daemons started !
```

3 Data Ingestion, Formatting, Enrichment and Filtering

3.1 Data Ingestion

Using ***populate-lookup.sh*** script we would get the look-up tables created in HBase. These look-up tables will be further used in – Data Formatting, Data Enrichment and Data Analysis.

Look up tables and related files.

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

Table-1

populate-lookup.sh script

* populate-lookup.sh

```
#!/bin/bash

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

echo "Creating LookUp Tables" >> $LOGFILE

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

echo "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/examples/music/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/examples/music/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"
```

```
37
38
39 file="/home/acadgild/examples/music/lookupfiles/user-subscn.txt"
40 while IFS= read -r line
41 do
42  userid=`echo $line | cut -d',' -f1`
43  startdt=`echo $line | cut -d',' -f2`
44  enddt=`echo $line | cut -d',' -f3`
45  echo "put 'subscribed-users', '$userid', 'subscn:startdt', '$startdt'" | hbase shell
46  echo "put 'subscribed-users', '$userid', 'subscn:enddt', '$enddt'" | hbase shell
47 done <"$file"
48
49 hive -f /home/acadgild/examples/music/user-artist.hql
```

In the below screenshots we can see the HBase tables getting created and the values are populated into them.

```
create 'station-geo-map', 'geo'
0 row(s) in 1.9930 seconds

Hbase::Table - station-geo-map
2018-07-22 21:57:55,631 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your pla
ava 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/s
nder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-lo
4j/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 'subscribed-users', 'subscn'
0 row(s) in 1.6560 seconds

Hbase::Table - subscribed-users
2018-07-22 21:58:06,842 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platfor
ava 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
nder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j1
4j/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 1.8060 seconds

Hbase::Table - song-artist-map
2018-07-22 21:58:18,571 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platf
ava 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/slf
nder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4
4j/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
```

Inserting values into tables:

```
put 'station-geo-map', 'ST400', 'geo:geo_cd', 'A'
0 row(s) in 0.8260 seconds

2018-07-22 21:58:29,142 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your plat
ava 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/sl
nder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log
4j/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
```

Let's check the look-up tables in HBase shell.

```
hbase(main):001:0> list
TABLE
song-artist-map
station-geo-map
subscribed-users
3 row(s) in 0.4250 seconds

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

Please find the tables with the values stored by script - populate-lookup.sh

```
hbase(main):002:0> scan 'song-artist-map'
ROW                             COLUMN+CELL
 S200                           column=artist:artistid, timestamp=1532277052808, value=A300
 S201                           column=artist:artistid, timestamp=1532277063975, value=A301
 S202                           column=artist:artistid, timestamp=1532277074927, value=A302
 S203                           column=artist:artistid, timestamp=1532277085940, value=A303
 S204                           column=artist:artistid, timestamp=1532277096508, value=A304
 S205                           column=artist:artistid, timestamp=1532277107380, value=A301
 S206                           column=artist:artistid, timestamp=1532277117916, value=A302
 S207                           column=artist:artistid, timestamp=1532277128708, value=A303
 S208                           column=artist:artistid, timestamp=1532277139626, value=A304
9 row(s) in 0.2440 seconds
```

```
hbase(main):003:0> scan 'station-geo-map'
ROW                              COLUMN+CELL
 ST400                           column=geo:geo_cd, timestamp=1532276901571, value=A
 ST401                           column=geo:geo_cd, timestamp=1532276912108, value=AU
 ST402                           column=geo:geo_cd, timestamp=1532276922831, value=AP
 ST403                           column=geo:geo_cd, timestamp=1532276933380, value=J
 ST404                           column=geo:geo_cd, timestamp=1532276944269, value=E
 ST405                           column=geo:geo_cd, timestamp=1532276954714, value=A
 ST406                           column=geo:geo_cd, timestamp=1532276966054, value=AU
 ST407                           column=geo:geo_cd, timestamp=1532276976538, value=AP
 ST408                           column=geo:geo_cd, timestamp=1532276987193, value=E
 ST409                           column=geo:geo_cd, timestamp=1532276998216, value=E
 ST410                           column=geo:geo_cd, timestamp=1532277009161, value=A
 ST411                           column=geo:geo_cd, timestamp=1532277020083, value=A
 ST412                           column=geo:geo_cd, timestamp=1532277030853, value=AP
 ST413                           column=geo:geo_cd, timestamp=1532277041902, value=J
14 row(s) in 0.1260 seconds
```

```
hbase(main):004:0> scan 'subscribed-users'
ROW                                COLUMN+CELL
 U100                              column=subscn:enddt, timestamp=1532277161513, value=1465130523
 U100                              column=subscn:startdt, timestamp=1532277150572, value=1465230523
 U101                              column=subscn:enddt, timestamp=1532277183558, value=1475130523
 U101                              column=subscn:startdt, timestamp=1532277172591, value=1465230523
 U102                              column=subscn:enddt, timestamp=1532277205001, value=1475130523
 U102                              column=subscn:startdt, timestamp=1532277194398, value=1465230523
 U103                              column=subscn:enddt, timestamp=1532277226044, value=1475130523
 U103                              column=subscn:startdt, timestamp=1532277215490, value=1465230523
 U104                              column=subscn:enddt, timestamp=1532277248517, value=1475130523
 U104                              column=subscn:startdt, timestamp=1532277237099, value=1465230523
 U105                              column=subscn:enddt, timestamp=1532277270534, value=1475130523
 U105                              column=subscn:startdt, timestamp=1532277259547, value=1465230523
 U106                              column=subscn:enddt, timestamp=1532277292198, value=1485130523
 U106                              column=subscn:startdt, timestamp=1532277281420, value=1465230523
 U107                              column=subscn:enddt, timestamp=1532277313425, value=1455130523
 U107                              column=subscn:startdt, timestamp=1532277302798, value=1465230523
 U108                              column=subscn:enddt, timestamp=1532277334643, value=1465230623
 U108                              column=subscn:startdt, timestamp=1532277323818, value=1465230523
 U109                              column=subscn:enddt, timestamp=1532277356273, value=1475130523
 U109                              column=subscn:startdt, timestamp=1532277345393, value=1465230523
 U110                              column=subscn:enddt, timestamp=1532277378160, value=1475130523
 U110                              column=subscn:startdt, timestamp=1532277367228, value=1465230523
 U111                              column=subscn:enddt, timestamp=1532277399844, value=1475130523
 U111                              column=subscn:startdt, timestamp=1532277389179, value=1465230523
 U112                              column=subscn:enddt, timestamp=1532277421027, value=1475130523
 U112                              column=subscn:startdt, timestamp=1532277410399, value=1465230523
 U113                              column=subscn:enddt, timestamp=1532277442510, value=1485130523
 U113                              column=subscn:startdt, timestamp=1532277431728, value=1465230523
14 row(s) in 0.1340 seconds
```

We can see, we have successfully completed the table creation 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.

```
-log4j2.properties Async: true
OK
Time taken: 8.277 seconds
OK
Time taken: 0.041 seconds
OK
Time taken: 1.229 seconds
Loading data to table project.users_artists
OK
Time taken: 1.723 seconds
Done with data population in look up tables !
```

Checking the Hive table as mentioned below.

```
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: 3.029 seconds, Fetched: 15 row(s)
hive>
```

Creating Hive Tables on the top of Hbase:

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.

data_enrichment_filtering_schema.sh

data_enrichment...

```
1 #!/bin/bash
2
3 batchid=`cat /home/acadgild/examples/music/logs/current-batch.txt`
4 LOGFILE=/home/acadgild/examples/music/logs/log_batch_$batchid
5
6 echo "Creating hive tables on top of hbase tables for data enrichment and filtering..." >> $LOGFILE
7
8 hive -f /home/acaidgild/examples/music/create_hive_hbase_lookup.hql
```

create_hive_hbase_lookup.hql

create_hive_hbase...

```
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 screenshpt shows how it is executed in the terminal

```
Creating hive tables on top of hbase tables for data enrichment and filtering...
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.jar!/hive-log4j2.properties Async: true
OK
Time taken: 7.786 seconds
OK
Time taken: 4.043 seconds
OK
Time taken: 0.517 seconds
OK
Time taken: 0.36 seconds
Hive table with Hbase Mapping Complete !
```

The next set of screenshots cross verifies that the tables are created in HIVE.

```
hive> use project;
OK
Time taken: 5.868 seconds
hive> show tables;
OK
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 0.329 seconds, Fetched: 4 row(s)
```

```
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
Time taken: 4.118 seconds, Fetched: 9 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
Time taken: 0.604 seconds, Fetched: 14 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
Time taken: 0.447 seconds, Fetched: 14 row(s)
```

3.2 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 **batched**.

dataformatting.sh

dataformatting.sh

```
#!/bin/bash

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


echo "Running script for Data Formatting...AnkithTest" >> $LOGFILE

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

Here we can see DataFormatting class will be submitted by Spark.

But prior to execution of above – get all the dependent jars downloaded for DataFormatting.scala using **sbt –v package** command (might take 30mins). This command downloads all the JARs required for the dependencies mentioned in the **buid.sbt** file.

```
[acadgild@localhost music]$ cd /home/acadgild/examples/music/MusicDataAnalysis/
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost MusicDataAnalysis]$ sbt -v package
[process_args] java_version = '1.8'
# Executing command line:
java
-Xms1024m
-Xmx1024m
-XX:ReservedCodeCacheSize=128m
-XX:MaxMetaspaceSize=256m
-jar
/usr/share/sbt/bin/sbt-launch.jar
package

Getting org.scala-sbt sbt 1.0.4  (this may take some time)...
```

```
[info] Done updating.
[info] Compiling 3 Scala sources to /home/acadgild/examples/music/MusicDataAnalysis/target/s
[info] Non-compiled module 'compiler-bridge_2.11' for Scala 2.11.8. Compiling...
[info]   Compilation completed in 23.006s.
[warn] there were three deprecation warnings; re-run with -deprecation for details
[warn] one warning found
[info] Done compiling.
[warn] Multiple main classes detected.  Run 'show discoveredMainClasses' to see the list
[info] Packaging /home/acadgild/examples/music/MusicDataAnalysis/target/scala-2.11/musicdata
[info] Done packaging.
[success] Total time: 1785 s, completed Jul 24, 2018 8:04:06 PM
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost MusicDataAnalysis]$
```

dataFormatting.scala

DataFormatting.scala

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

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

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

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

```
48
49
50      try {
51          val xmlData = sqlContext.read.format("com.databricks.spark.xml").option("rowTag", "record").load("file:///home/acadgild/examples/music/data/web/file.xml"
 )
52          xmlData.createOrReplaceTempView("web_data")
53
54          sqlContext.sql(create_hive_table)
55          sqlContext.sql(load_mob_data)
56          sqlContext.sql(load_web_data)
57        }
58       catch{
59        case e: Exception=>e.printStackTrace()
60       }
61  }
62 }
```

Make sure that Hive metastore service (use command – "***hive –service metastore***") is running and then we are executing ***dataformatting.sh***
The below screenshots represent the flow in the terminal.

```
[acadgild@localhost music]$ ./dataformatting.sh
Ivy Default Cache set to: /home/acadgild/.ivy2/cache
The jars for the packages stored in: /home/acadgild/.ivy2/jars
:: loading settings :: url = jar:file:/home/acadgild/install/spark/spark-2.2.1-bin-
settings/ivysettings.xml
com.databricks#spark-xml_2.10 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent;1.0
        confs: [default]
        found com.databricks#spark-xml_2.10;0.4.1 in central
:: resolution report :: resolve 348ms :: artifacts dl 21ms
        :: modules in use:
        com.databricks#spark-xml_2.10;0.4.1 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   1   |   0   |   0   |   0   ||   1   |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent
        confs: [default]
        0 artifacts copied, 1 already retrieved (0kB/23ms)
18/07/24 21:11:13 INFO spark.SparkContext: Running Spark version 2.2.1
18/07/24 21:11:14 WARN util.NativeCodeLoader: Unable to load native-hadoop library
```

```
18/07/24 21:11:42 INFO execution.SparkSqlParser: Parsing command: CREATE TABLE IF NOT EXISTS project.formatted_input
                        (
                        User_id STRING,
                        Song_id STRING,
                        Artist_id STRING,
                        Timestamp STRING,
                        Start_ts STRING,
                        End_ts STRING,
                        Geo_cd STRING,
                        Station_id STRING,
                        Song_end_type INT,
                        Like INT,
                        Dislike INT
                        )
                        PARTITIONED BY
                        (batchid INT)
                        ROW FORMAT DELIMITED
                        FIELDS TERMINATED BY ','

18/07/24 21:11:43 INFO parser.CatalystSqlParser: Parsing command: array<string>
18/07/24 21:11:45 INFO execution.SparkSqlParser: Parsing command: LOAD DATA LOCAL INPATH 'file:///home/acadgild/examples/music/data/m
ob/file.txt'
                        INTO TABLE project.formatted_input PARTITION (batchid='1')
18/07/24 21:11:46 INFO parser.CatalystSqlParser: Parsing command: int
```

We can check that a new table is created in Hive shell.

```
hive> show tables;
OK
formatted_input
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 0.08 seconds, Fetched: 5 row(s)
hive>
```

```
hive> select * from formatted_input;
OK
U103    S201    A302    1465230523      1485130523      1485130523      AP      ST411   0       0       1       1
U102    S200    A300    1475130523      1465130523      1485130523      AP      ST412   3       0       1       1
U112    S202    A305    1495130523      1465230523      1465130523      AU      ST404   1       1       0       1
U117    S204    A300    1475130523      1475130523      1465130523      U       ST407   1       0       0       1
U108    S208    A300    1465230523      1475130523      1465130523      U       ST401   2       0       1       1
        S202    A305    1475130523      1465230523      1465130523      AP      ST410   0       0       0       1
U116    S203    A305    1465130523      1465230523      1485130523      E       ST414   1       0       0       1
U118    S210    A301    1465130523      1485130523      1475130523      E       ST413   2       1       0       1
U112    S205    A301    1465230523      1485130523      1475130523              ST401   2       1       1       1
U109    S207            1465230523      1485130523      1465130523      E       ST413   3       1       1       1
U112    S201    A303    1495130523      1485130523      1465230523      AP      ST411   1       0       1       1
U113    S206    A302    1495130523      1485130523      1465130523      E       ST400   0       1       0       1
U112    S200    A300    1475130523      1465230523      1465230523      A       ST414   2       0       0       1
U114    S210    A300    1465130523      1475130523      1475130523      E       ST409   3       0       0       1
U109    S207    A303    1465130523      1485130523      1475130523      E       ST402   1       0       0       1
U115    S209    A301    1495130523      1485130523      1475130523      AP      ST406   2       0       1       1
U112    S203    A302    1465230523      1465130523      1465130523      AP      ST413   1       1       1       1
U116    S206    A300    1465230523      1485130523      1465230523      AU      ST410   3       0       0       1
U107    S210    A302    1465130523      1465230523      1465130523      A       ST405   3       0       0       1
U103    S206    A305    1465130523      1465130523      1475130523      AU      ST407   3       1       0       1
U114    S206    A304    1462863262      1468094889      1462863262      AP      ST403   1       1       0       1
U104    S205    A305    1462863262      1462863262      1468094889      AU      ST405   2       1       1       1
U104    S206    A302    1462863262      1465490556      1494297562      AU      ST400   3       0       0       1
U111    S201    A305    1494297562      1465490556      1494297562      AP      ST405   3       0       0       1
U118    S208    A302    1468094889      1462863262      1468094889      U       ST407   0       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 mention anything about **user_id** for enrichment purpose.

3.3 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:

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

data_enrichment.sh

```
#!/bin/bash

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

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

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

```
14
15 if [ ! -d "$VALIDDIR" ]
16 then
17 mkdir -p "$VALIDDIR"
18 fi
19
20 if [ ! -d "$INVALIDDIR" ]
21 then
22 mkdir -p "$INVALIDDIR"
23 fi
24
25 echo "Copying valid and invalid records in local file system...AnkithTest" >> $LOGFILE
26
27 hadoop fs -get /user/hive/warehouse/project.db/enriched_data/batchid=$batchid/status=pass/* $VALIDDIR
28 hadoop fs -get /user/hive/warehouse/project.db/enriched_data/batchid=$batchid/status=fail/* $INVALIDDIR
29
30 echo "Deleting older valid and invalid records from local file system... AnkithTest" >> $LOGFILE
31
32 find /home/acadgild/examples/music/processed_dir/ -mtime +7 -exec rm {} \;
```

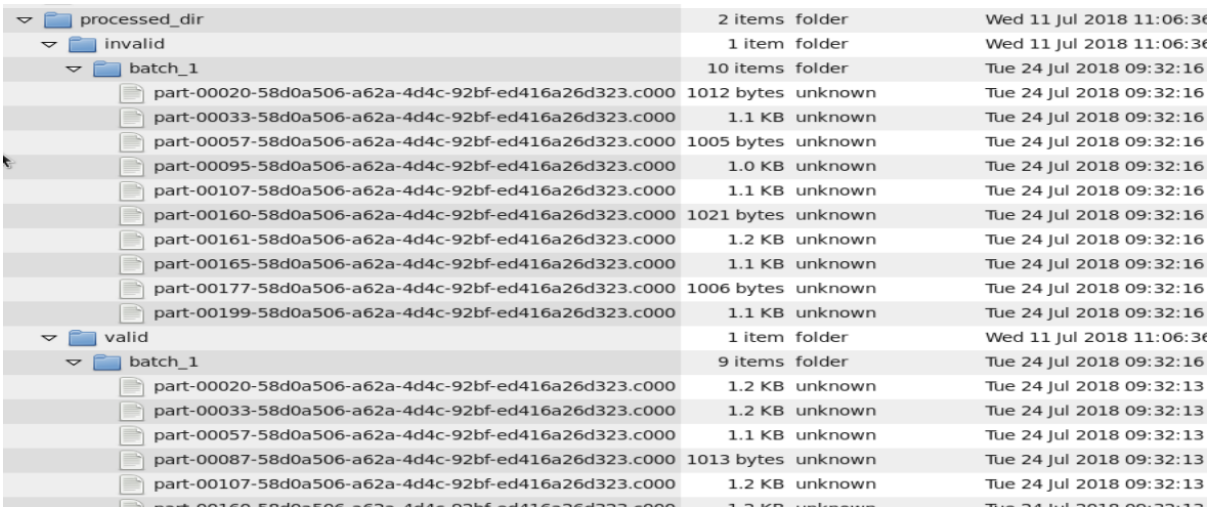
Once running the above script in terminal we would get a new table created in Hive. Please find the below screenshots for the Hive screens.

```
hive> show tables;
OK
enriched_data
formatted_input
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 0.059 seconds, Fetched: 6 row(s)
```

```
hive> select * from enriched_data;
OK
U113    S201    A301    1462863262      1465490556      1465490556      NULL    ST415   1       0       1       1       fail
NULL    S207    A303    1462863262      1494297562      1468094889      NULL    ST414   0       0       0       1       fail
U109    S207    A303    1465230523      1485130523      1465130523      J       ST413   3       1       1       1       fail
        S202    A302    1475130523      1465230523      1465130523      A       ST410   0       0       0       1       fail
U115    S209    NULL    1495130523      1485130523      1475130523      AU      ST406   2       0       1       1       fail
U108    S206    A302    1465490556      1462863262      1462863262      J       ST403   1       1       1       1       fail
U114    S206    A302    1468094889      1468094889      1494297562      E       ST409   3       0       0       1       fail
U101    S208    A304    1494297562      1468094889      1494297562      NULL    ST415   1       1       0       1       fail
U107    S210    NULL    1465130523      1465230523      1465130523      A       ST405   3       0       0       1       fail
U114    S210    NULL    1465130523      1475130523      1475130523      E       ST409   3       0       0       1       fail
U119    S210    NULL    1494297562      1465490556      1465490556      A       ST410   1       0       1       1       fail
U118    S210    NULL    1465130523      1485130523      1475130523      J       ST413   2       1       0       1       fail
U108    S210    NULL    1468094889      1465490556      1462863262      J       ST413   3       0       0       1       fail
U104    S205    A301    1462863262      1462863262      1468094889      A       ST405   2       1       1       1       fail
U112    S205    A301    1465230523      1485130523      1475130523      AU      ST401   2       1       1       1       fail
U112    S200    A300    1475130523      1465230523      1465230523      NULL    ST414   2       0       0       1       fail
U116    S203    A303    1465130523      1465230523      1485130523      NULL    ST414   1       0       0       1       fail
U112    S203    A303    1465230523      1465130523      1465130523      J       ST413   1       1       1       1       fail
U111    S201    A301    1494297562      1465490556      1494297562      A       ST405   3       0       0       1       pass
U103    S201    A301    1465230523      1485130523      1485130523      A       ST411   0       0       1       1       pass
U112    S201    A301    1495130523      1485130523      1465230523      A       ST411   1       0       1       1       pass
U120    S201    A301    1494297562      1462863262      1465490556      J       ST413   1       1       0       1       pass
U109    S207    A303    1465130523      1485130523      1475130523      AP      ST402   1       0       0       1       pass
U119    S207    A303    1468094889      1494297562      1468094889      J       ST403   1       0       0       1       pass
U100    S207    A303    1462863262      1462863262      1465490556      A       ST410   0       1       0       1       pass
U112    S202    A302    1495130523      1465230523      1465130523      E       ST404   1       1       0       1       pass
U117    S202    A302    1465490556      1465490556      1465490556      E       ST409   0       0       0       1       pass
U117    S204    A304    1475130523      1475130523      1465130523      AP      ST407   1       0       0       1       pass
U113    S206    A302    1495130523      1485130523      1465130523      A       ST400   0       1       0       1       pass
U104    S206    A302    1462863262      1465490556      1494297562      A       ST400   3       0       0       1       pass
U114    S206    A302    1462863262      1468094889      1462863262      J       ST403   1       1       0       1       pass
```

In the above screenshot we can see the records are categorised as “pass” or “fail”.

All the **“pass”** records are accumulated in **“processed_dir/valid/batch_1”** and “**fail”** records are accumulated in **“processed_dir/invalid/batch_1” (**refer below screenshot**)**



processed_dir	2 items	folder	Wed 11 Jul 2018 11:06:36
invalid	1 item	folder	Wed 11 Jul 2018 11:06:36
batch_1	10 items	folder	Tue 24 Jul 2018 09:32:16
part-00020-58d0a506-a62a-4d4c-92bf-ed416a26d323.c000	1012 bytes	unknown	Tue 24 Jul 2018 09:32:16
part-00033-58d0a506-a62a-4d4c-92bf-ed416a26d323.c000	1.1 KB	unknown	Tue 24 Jul 2018 09:32:16
part-00057-58d0a506-a62a-4d4c-92bf-ed416a26d323.c000	1005 bytes	unknown	Tue 24 Jul 2018 09:32:16
part-00095-58d0a506-a62a-4d4c-92bf-ed416a26d323.c000	1.0 KB	unknown	Tue 24 Jul 2018 09:32:16
part-00107-58d0a506-a62a-4d4c-92bf-ed416a26d323.c000	1.1 KB	unknown	Tue 24 Jul 2018 09:32:16
part-00160-58d0a506-a62a-4d4c-92bf-ed416a26d323.c000	1021 bytes	unknown	Tue 24 Jul 2018 09:32:16
part-00161-58d0a506-a62a-4d4c-92bf-ed416a26d323.c000	1.2 KB	unknown	Tue 24 Jul 2018 09:32:16
part-00165-58d0a506-a62a-4d4c-92bf-ed416a26d323.c000	1.1 KB	unknown	Tue 24 Jul 2018 09:32:16
part-00177-58d0a506-a62a-4d4c-92bf-ed416a26d323.c000	1006 bytes	unknown	Tue 24 Jul 2018 09:32:16
part-00199-58d0a506-a62a-4d4c-92bf-ed416a26d323.c000	1.1 KB	unknown	Tue 24 Jul 2018 09:32:16
valid	1 item	folder	Wed 11 Jul 2018 11:06:36
batch_1	9 items	folder	Tue 24 Jul 2018 09:32:16
part-00020-58d0a506-a62a-4d4c-92bf-ed416a26d323.c000	1.2 KB	unknown	Tue 24 Jul 2018 09:32:13
part-00033-58d0a506-a62a-4d4c-92bf-ed416a26d323.c000	1.2 KB	unknown	Tue 24 Jul 2018 09:32:13
part-00057-58d0a506-a62a-4d4c-92bf-ed416a26d323.c000	1.1 KB	unknown	Tue 24 Jul 2018 09:32:13
part-00087-58d0a506-a62a-4d4c-92bf-ed416a26d323.c000	1013 bytes	unknown	Tue 24 Jul 2018 09:32:13
part-00107-58d0a506-a62a-4d4c-92bf-ed416a26d323.c000	1.2 KB	unknown	Tue 24 Jul 2018 09:32:13

3.4 Data Analysis

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

All the Spark SQL are captured in ***DataAnalysis.scala*** which will be internally triggered by ***data_analysis.sh.***

data_analysis.sh

data_analysis.sh

```
#!/bin/bash

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

echo "Running script for data analysis for different use cases... AnkithTest"

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

echo "Running script for data analysis for different use cases... AnkithTest-Completed"

```

DataAnalysis.scala

DataAnalysis.scala

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

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


val create_top_10_stations = """CREATE TABLE IF NOT EXISTS top_10_stations
(
station_id STRING,
total_distinct_songs_played INT,
distinct_user_count INT
)
PARTITIONED BY (batchid INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE"""


val load_top_10_stations = s"""INSERT OVERWRITE TABLE top_10_stations
PARTITION(batchid='$batchId')
SELECT
station_id,
COUNT(DISTINCT song_id) AS total_distinct_songs_played,
COUNT(DISTINCT user_id) AS distinct_user_count
FROM enriched_data
WHERE status='pass'
AND batchid='$batchId'
AND like=1
GROUP BY station_id
ORDER BY total_distinct_songs_played DESC
LIMIT 10"""


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

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

```

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

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

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

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

val create_top_10_unsubscribed_users = """CREATE TABLE IF NOT EXISTS top_10_unsubscribed_users
(
user_id STRING,
duration INT
)
PARTITIONED BY (batchid INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE"""
```

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


    try {
         sqlContext.sql("SET hive.auto.convert.join=false")
         sqlContext.sql("USE project")
         sqlContext.sql(create_top_10_stations)
         sqlContext.sql(load_top_10_stations)
         sqlContext.sql(create_users_behaviour)
         sqlContext.sql(load_users_behaviour)
         sqlContext.sql(create_connected_artists)
         sqlContext.sql(load_connected_artists)
         sqlContext.sql(create_top_10_royalty_songs)
         sqlContext.sql(load_top_10_royalty_songs)
         sqlContext.sql(create_top_10_unsubscribed_users)
         sqlContext.sql(load_top_10_unsubscribed_users)
        }
       catch{
        case e: Exception=>e.printStackTrace()
       }
 }
}

```

After executing the above script, we would get the below

```
18/07/24 22:00:50 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator st
18/07/24 22:00:50 INFO spark.SparkContext: Successfully stopped SparkContext
18/07/24 22:00:50 INFO util.ShutdownHookManager: Shutdown hook called
18/07/24 22:00:50 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-3c8ddd16-fb74-4688-adf2-0288d2e0da92
Running script for data analysis for different use cases... AnkithTest-Completed
USE CASES COMPLETE !!
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost music]$
```

And for each of the data analysis we have ingested the final output into the tables in HIVE.

Please find the new tables available in HIVE.

```
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: 0.06 seconds, Fetched: 11 row(s)
hive>
```

```
hive> select * from connected_artists;
OK
A301    4       1
A302    3       1
A303    1       1
A304    1       1
Time taken: 0.341 seconds, Fetched: 4 row(s)
```

```
hive> select * from top_10_royalty_songs;
OK
S206    35231627        1
S208    5231627 1
S205    5231627 1
S201    2627294 1
S207    2627294 1
S200    2627294 1
S202    100000  1
Time taken: 0.389 seconds, Fetched: 7 row(s)
```

```
hive> select * from top_10_stations;
OK
ST400   2       2       1
ST404   1       1       1
ST403   1       1       1
ST410   1       1       1
ST407   1       1       1
ST413   1       1       1
Time taken: 0.435 seconds, Fetched: 6 row(s)
```

```
hive> select * from top_10_unsubscribed_users;
OK
U111    28807006        1
U107    26202673        1
U119    26202673        1
U112    20000000        1
U113    20000000        1
U116    19900000        1
U117    10000000        1
U100    7835960 1
U114    5231627 1
U118    5231627 1
Time taken: 0.294 seconds, Fetched: 10 row(s)
```

```
hive> select * from  users_behaviour;
OK
UNSUBSCRIBED    174666154       1
SUBSCRIBED      81434300        1
Time taken: 0.424 seconds, Fetched: 2 row(s)
hive>
```

So we have successfully completed the data analysis for the given dataset. But now – we look forward to store the results. This is explained in the next section – we have implemented it by using the “export” concept of sqoop as mentioned in the script - ***data_export.sh***.

3.5 Data Export

In **data_export.sh** we are going to export the data from the hive tables into mysql using **Sqoop** export.

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/examples/music/logs/current-batch.txt`
LOGFILE=/home/acadgild/examples/music/logs/log_batch_$batchid

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

mysql -u "root" "-pRoot@123" < /home/acadgild/examples/music/create_schema.sql

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

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

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

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

sqoop export --connect jdbc:mysql://localhost/project --username root --password Root@123 --table top_10_royalty_songs --export-dir /user/hive/warehouse/project.db/
/top_10_royalty_songs/batchid=$batchid --input-fields-terminated-by ',' -m 1

sqoop export --connect jdbc:mysql://localhost/project --username root --password Root@123 --table top_10_unsubscribed_users --export-dir /user/hive/warehouse/
/project.db/top_10_unsubscribed_users/batchid=$batchid --input-fields-terminated-by ',' -m 1
```

create_schema.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 export using export sqoop is in-progress.

```
Let us export the final data....
mysql: [Warning] Using a password on the command line interface can be insecure.
Warning: /home/acadgild/install/sqoop/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/install/sqoop/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.
18/07/25 21:34:03 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6
18/07/25 21:34:03 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
18/07/25 21:34:03 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
18/07/25 21:34:03 INFO tool.CodeGenTool: Beginning code generation
Wed Jul 25 21:34:03 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. According
ySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set. For compli
 with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either to explicitly disa
SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
18/07/25 21:34:04 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `top_10_stations` AS t LIMIT 1
18/07/25 21:34:04 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `top_10_stations` AS t LIMIT 1
18/07/25 21:34:04 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /home/acadgild/install/hadoop/hadoop-2.6.5
Note: /tmp/sqoop-acadgild/compile/b03f92d636d323d60b5f4f15a9a82d55/top_10_stations.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
18/07/25 21:34:07 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-acadgild/compile/b03f92d636d323d60b5f4f15a9a82d55/top_10
tions.jar
18/07/25 21:34:07 INFO mapreduce.ExportJobBase: Beginning export of top_10_stations
```

```
18/07/25 21:34:11 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1532530385941_0002
18/07/25 21:34:12 INFO impl.YarnClientImpl: Submitted application application_1532530385941_0002
18/07/25 21:34:12 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_15325303
18/07/25 21:34:12 INFO mapreduce.Job: Running job: job_1532530385941_0002
18/07/25 21:34:21 INFO mapreduce.Job: Job job_1532530385941_0002 running in uber mode : false
18/07/25 21:34:21 INFO mapreduce.Job:  map 0% reduce 0%
18/07/25 21:34:30 INFO mapreduce.Job:  map 100% reduce 0%
18/07/25 21:34:30 INFO mapreduce.Job: Job job_1532530385941_0002 completed successfully
18/07/25 21:34:30 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=127642
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=276
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=4
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=0
        Job Counters
                Launched map tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=5658
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=5658
                Total vcore-milliseconds taken by all map tasks=5658
                Total megabyte-milliseconds taken by all map tasks=5793792
        Map-Reduce Framework
                Map input records=6
                Map output records=6
                Input split bytes=213
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=63
                CPU time spent (ms)=1450
```

The sqoop export command exported the tables from the hive and it stored in the Mysql.

The data stored in the Mysql is shown in the successive screen shots –

“project” database

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| metastore          |
| mysql              |
| oozie              |
| performance_schema |
| project            |
| simplidb           |
| sys                |
+--------------------+
8 rows in set (0.00 sec)
```

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

Database changed
mysql> show tables;
+---------------------------+
| Tables_in_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 connected_artists;
+-----------+------------+
| artist_id | user_count |
+-----------+------------+
| A301      |          4 |
| A302      |          3 |
| A303      |          1 |
| A304      |          1 |
+-----------+------------+
4 rows in set (0.00 sec)
```

```
mysql> select * from top_10_royalty_songs;
+---------+----------+
| song_id | duration |
+---------+----------+
| S206    | 35231627 |
| S208    |  5231627 |
| S205    |  5231627 |
| S201    |  2627294 |
| S207    |  2627294 |
| S200    |  2627294 |
| S202    |   100000 |
+---------+----------+
7 rows in set (0.00 sec)
```

```
mysql> select * from top_10_stations;
+------------+-------------------------------+----------------------+
| station_id | total_distinct_songs_played | distinct_user_count |
+------------+-------------------------------+----------------------+
| ST400      |                             2 |                    2 |
| ST404      |                             1 |                    1 |
| ST403      |                             1 |                    1 |
| ST410      |                             1 |                    1 |
| ST407      |                             1 |                    1 |
| ST413      |                             1 |                    1 |
+------------+-------------------------------+----------------------+
6 rows in set (0.00 sec)
```

```
mysql> select * from top_10_unsubscribed_users;
+---------+----------+
| user_id | duration |
+---------+----------+
| U111    | 28807006 |
| U107    | 26202673 |
| U119    | 26202673 |
| U112    | 20000000 |
| U113    | 20000000 |
| U116    | 19900000 |
| U117    | 10000000 |
| U100    |  7835960 |
| U114    |  5231627 |
| U118    |  5231627 |
+---------+----------+
10 rows in set (0.00 sec)
```

```
mysql> select * from users_behaviour;
+--------------+-----------+
| user_type    | duration  |
+--------------+-----------+
| UNSUBSCRIBED | 174666154 |
| SUBSCRIBED   |  81434300 |
+--------------+-----------+
2 rows in set (0.00 sec)
```

Now having a look at the log file generated.

```
[acadgild@localhost music]$ cat /home/acadgild/examples/music/logs/log_batch_1
Starting daemons
Creating LookUp Tables
Populating LookUp Tables
Creating hive tables on top of hbase tables for data enrichment and filtering...
Running script for Data Formatting...AnkithTest
Running script for data enrichment and filtering... AnkithTest
Copying valid and invalid records in local file system...AnkithTest
Deleting older valid and invalid records from local file system... AnkithTest
Creating mysql tables if not present...AnkithTest
Running sqoop job for data export...AnkithTest
```

3.6 Job Scheduling

For scheduling, the master file - music_project_master.sh, which holds all the scripts used in the Project, should run in a period of every 3 hours.

This scheduling can usually be done using the crontab. As of now, we are not going to schedule the file since this project was a self-learning activity where we had only one input batch files.

3.7 Problems faced during project installation and how it resolved

1. The frequent error was the metastore service was not connected due to which I had trouble on creation of hbase tables.
2. I was explicitly mentioning localhost address for HDFS in the sqoop export command which was not required.