



Project - Music Data Analysis

Contents

<u>Section – 1 - Project Overview</u>	<u>2</u>
<u>1.1 Fields present in the data files</u>	<u>2</u>
<u>1.2 LookUp Tables</u>	<u>2</u>
<u>1.3 DATASET</u>	<u>3</u>
<u>1.4 Data Enrichment</u>	<u>4</u>
<u>1.5 Data Analysis (SHOULD BE IMPLMETED IN SPARK)</u>	<u>4</u>
<u>1.6 Challenges and Optimizations:</u>	<u>4</u>
<u>1.7 Flow of operations</u>	<u>5</u>
<u>Section -2 - Design of the Project</u>	<u>6</u>
<u>2.1 Low Level Design</u>	<u>6</u>
<u>2.2 High Level Design</u>	<u>7</u>
<u>Section-3-Hadoop Eco-System Implementation</u>	<u>8</u>
<u>Section-4 -Data Ingestion, Formatting, Enrichment and Filtering</u>	
<u>10</u>	
<u>4.1 Stage - 1 - Data Ingestion</u>	
<u>10</u>	
<u>4.2 Stage - 2 - Data Formatting</u>	
<u>19</u>	
<u>4.3 Stage - 3 - Data Enrichment & Filtering</u>	
<u>24</u>	
<u>4.4 Stage - 4 - Data Analysis using Spark</u>	
<u>29</u>	
<u>4.5 Stage - 5 - Data Storage in MYSQL</u>	
<u>37</u>	
<u>Job Scheduling:</u>	
<u>40</u>	
<u>Problems faced during project installation and how it resolved</u>	
<u>43</u>	

Highlights of the Project

44

Project End Conclusion:

44



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 liked 1 means song was liked
Dislike	0 means song was not disliked 1 means song was disliked

1.2 LookUp Tables

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

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



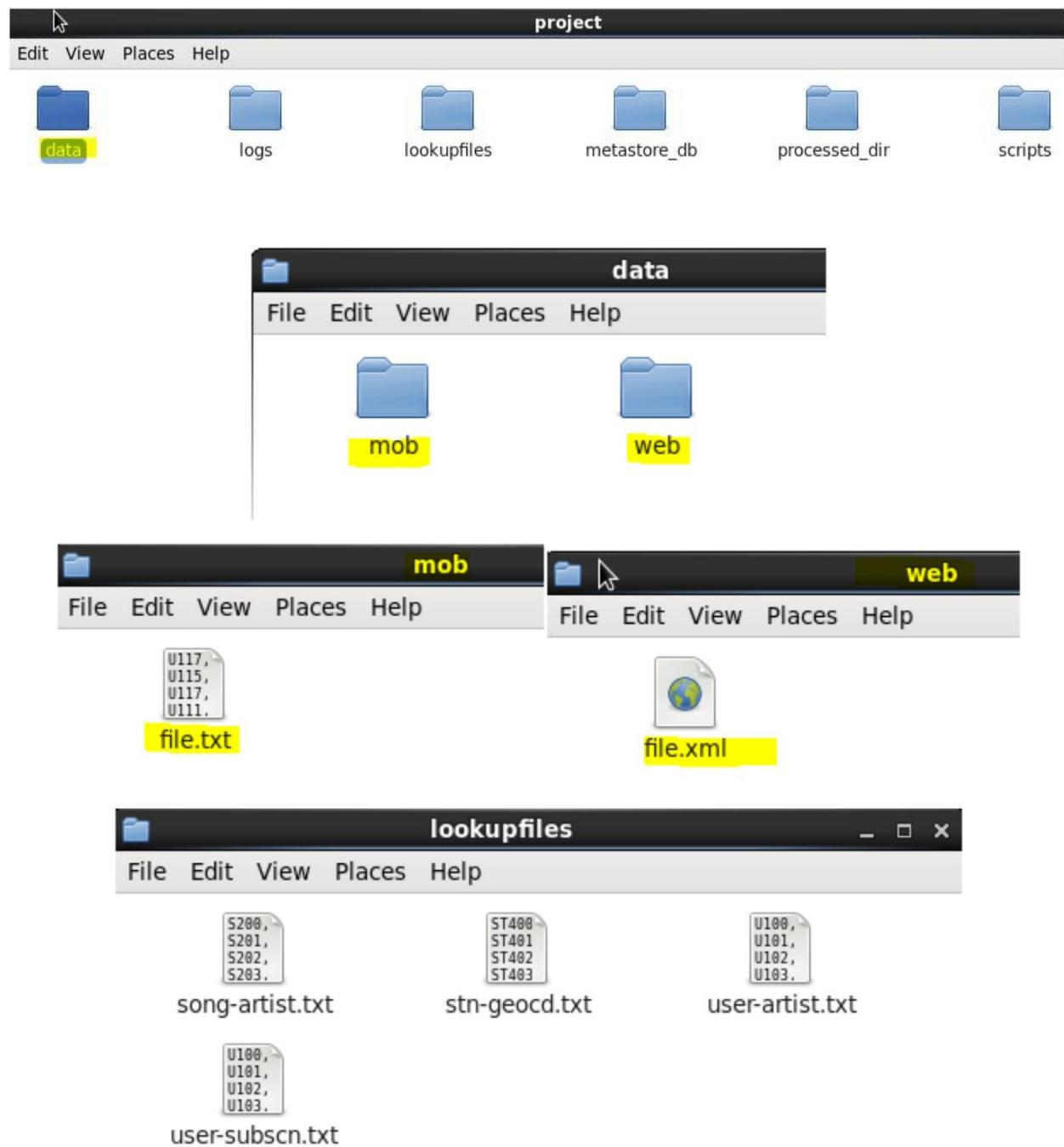
ACADGILD

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_P3pWagdIrrMjjGVINsSUEtbG8?usp=sharing





ACADGILD

1.4 Data Enrichment

Rules for data enrichment,

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

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

1.5 Data Analysis (SHOULD BE IMPLEMENTED IN SPARK)

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.



ACADGILD

1.7 Flow of operations

A schematic flow of operations is shown below,

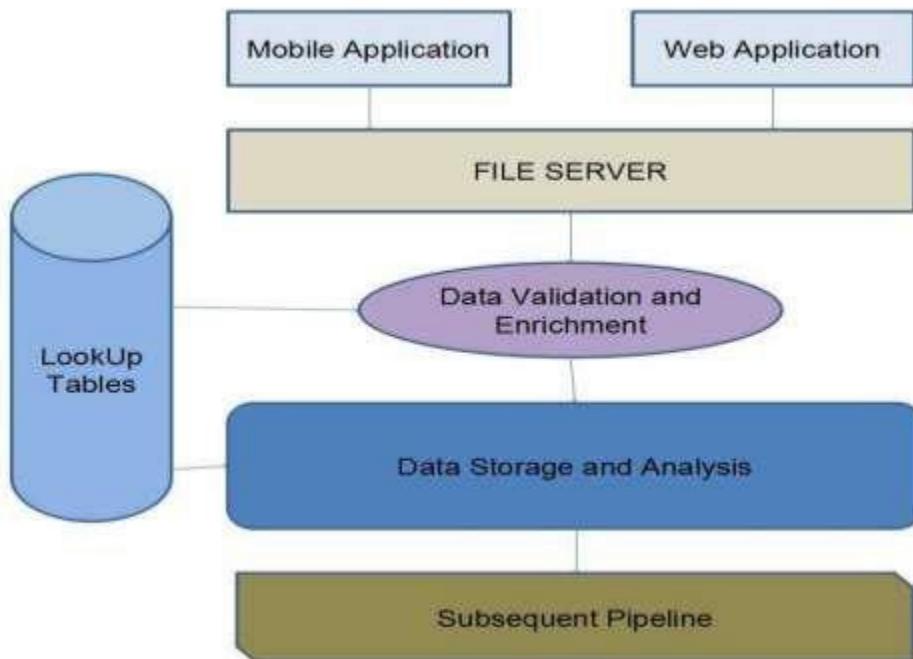


Fig-1

In the following sections, we are going to see the Music Data Analysis as per the above rules.



Section -2 – Design of the Project

2.1 Low Level Design

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

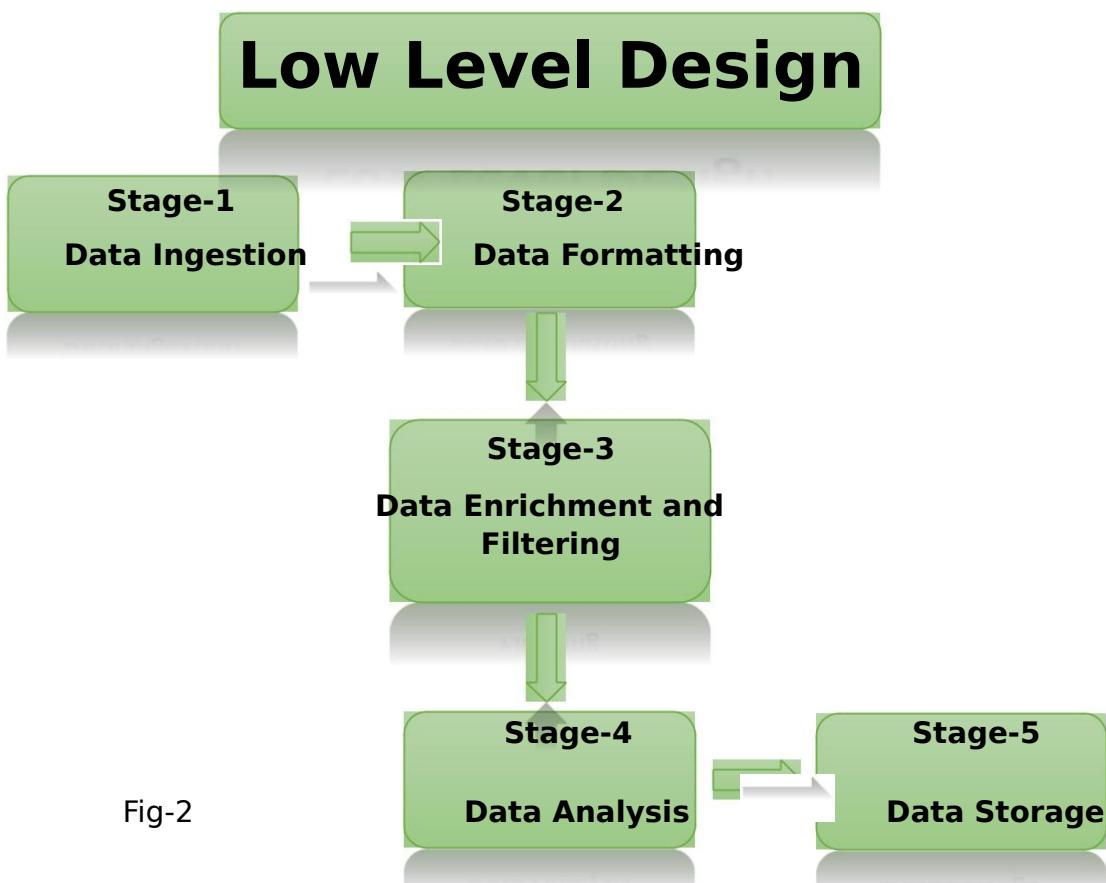


Fig-2



2.2 High Level Design

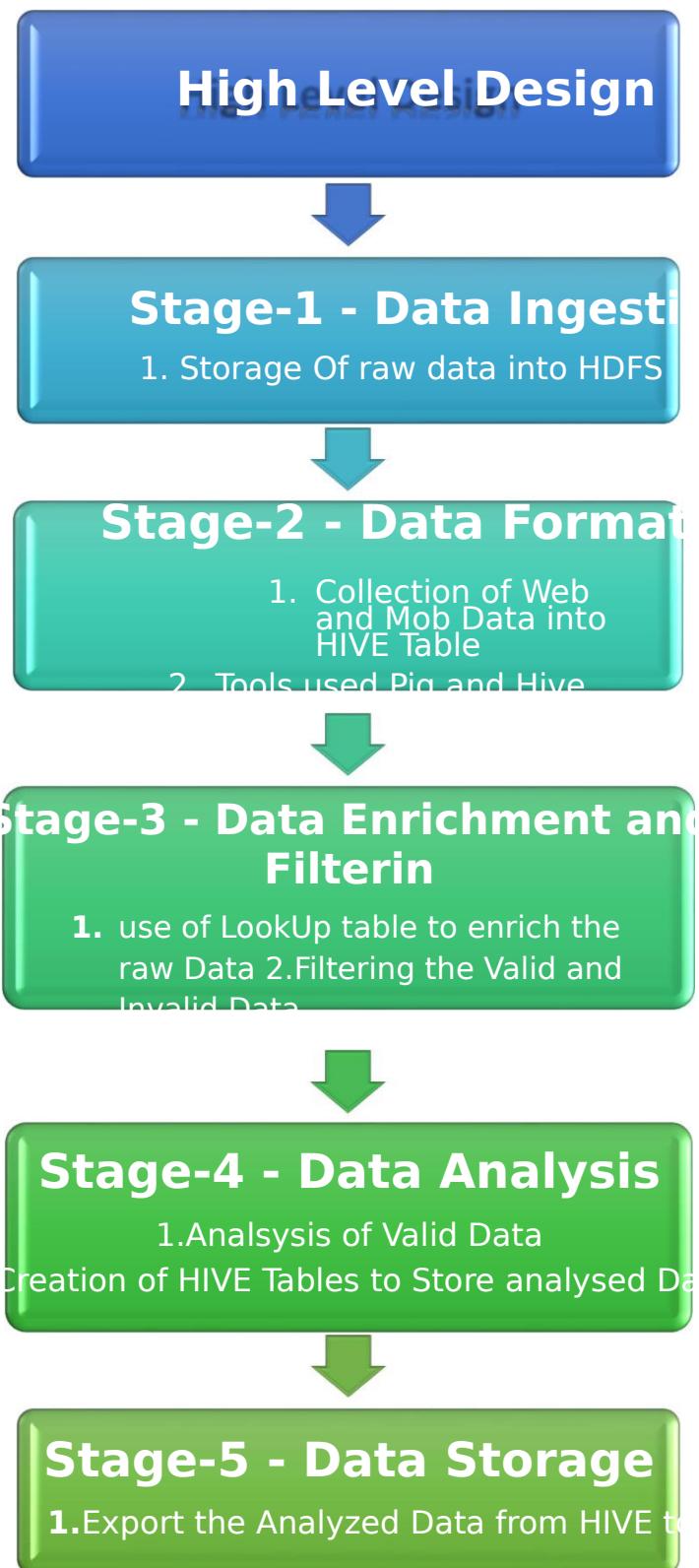


Fig-3



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,

```
#!/bin/bash

if [ -f "/home/acadgild/Project_2_Music_Data_Analysis/logs/current-batch.txt" ]
then
echo "Batch File Found!"
else
echo -n "1" > "/home/acadgild/Project_2_Music_Data_Analysis/logs/current-batch.txt"
fi

chmod 775 /home/acadgild/Project_2_Music_Data_Analysis/logs/current-batch.txt
batchid=`cat /home/acadgild/Project_2_Music_Data_Analysis/logs/current-batch.txt`
LOGFILE=/home/acadgild/Project_2_Music_Data_Analysis/logs/log_batch_$batchid

echo "Starting daemons" >> $LOGFILE

# To Start Hadoop Daemons:
start-all.sh

# To start the HMASTER service:
start-hbase.sh

# To Start the JobHistory server Services:
mr-jobhistory-daemon.sh start historyserver

# To Start the mysql service
sudo service mysqld start

# To Start HIVE metastore:
hive --service metastore
```

2. Starting all daemons,

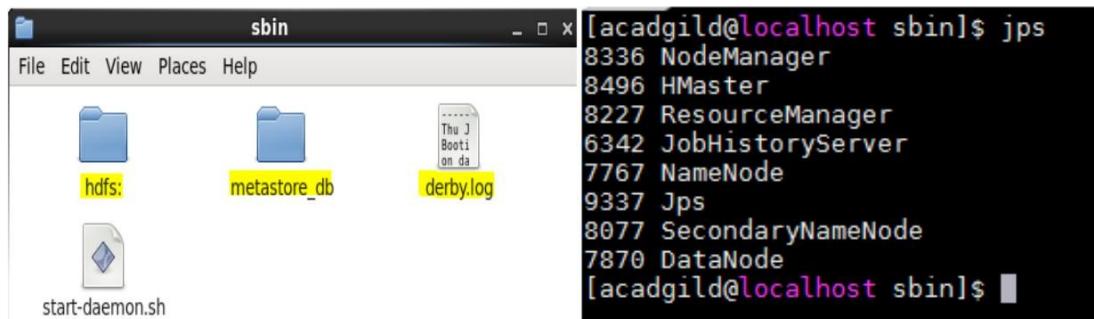
► **sh start-daemon.sh**

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,

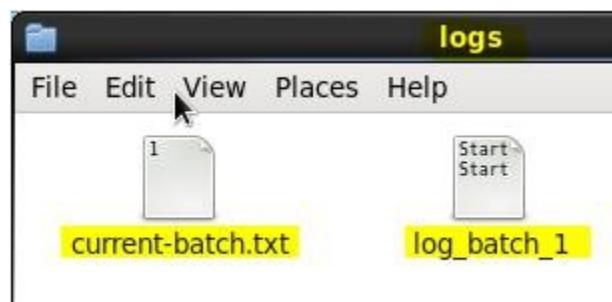


```
[acadgild@localhost sbin]$ sh start-daemon.sh
Batch File Found
This script is deprecated. Instead use start-dfs.sh and start-yarn.sh
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which might have disabled stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/18 11:40:39 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/acadgild/hadoop-2.7.2/logs/hadoop-acadgild-namenode-localhost.localdomain.out
localhost: starting datanode, logging to /home/acadgild/hadoop-2.7.2/logs/hadoop-acadgild-datanode-localhost.localdomain.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/acadgild/hadoop-2.7.2/logs/hadoop-acadgild-secondarynamenode-localhost.localdomain.out
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which might have disabled stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/18 11:42:06 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
starting yarn daemons
starting resourcemanager, logging to /home/acadgild/hadoop-2.7.2/logs/yarn-acadgild-resourcemanager-localhost.localdomain.out
localhost: starting nodemanager, logging to /home/acadgild/hadoop-2.7.2/logs/yarn-acadgild-nodemanager-localhost.localdomain.out
starting master, logging to /home/acadgild/hbase-1.0.3/logs/hbase-acadgild-master-localhost.localdomain.out
historyserver running as process 6342. Stop it first.
[sudo] password for acadgild:
Starting mysqld: [ OK ]
Starting Hive Metastore Server
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/apache-hive-2.1.0-bin/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/hadoop-2.7.2/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]
javax.jdo.JDODataStoreException: Required table missing : "VERSION" in Catalog "" Schema "".
DataNucleus requires this table to perform its persistence operations. Either your MetaData is incorrect, or you need to enable "datanucleus.schema.autoCreateTables"
    at org.datanucleus.api.jdo.NucleusJDOHelper.getJDOExceptionForNucleusException(NucleusJDOHelper.java:553)
    at org.datanucleus.api.jdo.JDOPersistenceManager._doMakePersistent(JDOPersistenceManager.java:720)
```

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

Table-1

“**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.

1
0Project 2 -Music Data Analysis
*populate-lookup.sh*ACADGI
LD

```
1  #!/bin/bash
2
3 batchid=`cat /home/acadgild/project/logs/current-batch.txt`
4
5 LOGFILE=/home/acadgild/project/logs/log_batch_$batchid
6
7 echo "Creating LookUp Tables" >> $LOGFILE
8
9 echo "create 'station-geo-map', 'geo'" | hbase shell
10 echo "create 'subscribed-users', 'subscn'" | hbase shell
11 echo "create 'song-artist-map', 'artist'" | hbase shell
12
13
14 echo "Populating LookUp Tables" >> $LOGFILE
15
16 file="/home/acadgild/project/lookupfiles/stn-geocd.txt"
17 while IFS= read -r line
18 do
19     stnid=`echo $line | cut -d',' -f1`
20     geocd=`echo $line | cut -d',' -f2`
21     echo "put 'station-geo-map', '$stnid', 'geo:geo_cd', '$geocd'" | hbase shell
22 done <"$file"
23
24
25 file="/home/acadgild/project/lookupfiles/song-artist.txt"
26 while IFS= read -r line
27 do
28     songid=`echo $line | cut -d',' -f1`
29     artistid=`echo $line | cut -d',' -f2`
30     echo "put 'song-artist-map', '$songid', 'artist:artistid', '$artistid'" | hbase shell
31 done <"$file"
32
33
34 file="/home/acadgild/project/lookupfiles/user-subscn.txt"
35 while IFS= read -r line
36 do
37     userid=`echo $line | cut -d',' -f1`
38     startdt=`echo $line | cut -d',' -f2`
39     enddt=`echo $line | cut -d',' -f3`
40     echo "put 'subscribed-users', '$userid', 'subscn:startdt', '$startdt'" | hbase shell
41     echo "put 'subscribed-users', '$userid', 'subscn:enddt', '$enddt'" | hbase shell
42 done <"$file"
43
44 hive -f /home/acadgild/project/scripts/user-artist.hql
45
```

Run the script: *./populate-lookup.sh*



```
-rwxrwxr--. 1 acadgild acadgild 412 Jan 19 22:14 wrapper.sh
[acadgild@localhost scripts]$ ./populate-lookup.sh
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/hbase-1.0.3/lib/slf4j-log4j1
SLF4J: Found binding in [jar:file:/home/acadgild/hadoop-2.7.2/share/hadoop/core.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/libhadoop.so which might conflict with your current native libhadoop.so version.
It's highly recommended that you fix the library with 'execstack -c <libfile>' or 'ldconfig'.
2018-01-19 22:42:24,744 WARN  [main] util.NativeCodeLoader: Unable to load native library for your platform; try setting 'java.library.path' to a directory containing your native library, or set 'native.library.path' to search paths (see java.lang.Runtime.availableProcessors())
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.0.3, rfle1312f9790a7c40f6a4b5a1bab2ealdd559890, Tue Jan 19 19:24:24 UTC 2018
create 'station-geo-map', 'geo'
0 row(s) in 1.3100 seconds

Hbase::Table - station-geo-map
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/hbase-1.0.3/lib/slf4j-log4j1
SLF4J: Found binding in [jar:file:/home/acadgild/hadoop-2.7.2/share/hadoop/core.class]
```

```
create 'subscribed-users', 'subscn'
0 row(s) in 1.7040 seconds

Hbase::Table - subscribed-users
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/hbase-1.0.3/lib/slf4j-log4j1
SLF4J: Found binding in [jar:file:/home/acadgild/hadoop-2.7.2/share/hadoop/core.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/libhadoop.so which might conflict with your current native libhadoop.so version.
It's highly recommended that you fix the library with 'execstack -c <libfile>' or 'ldconfig'.
2018-01-19 22:43:17,551 WARN  [main] util.NativeCodeLoader: Unable to load native library for your platform; try setting 'java.library.path' to a directory containing your native library, or set 'native.library.path' to search paths (see java.lang.Runtime.availableProcessors())
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.0.3, rfle1312f9790a7c40f6a4b5a1bab2ealdd559890, Tue Jan 19 19:26:53

create 'song-artist-map', 'artist'
0 row(s) in 1.4620 seconds

Hbase::Table - song-artist-map
```




```
Type "exit<RETURN>" to leave the HBase Shell
Version 1.0.3, rf1e1312f9790a7c40f6a4b5a1bab2ealdd559890, Tue Jan 19 19:26:53

put 'subscribed-users', 'U114', 'subscn:enddt', '1468130523'
0 row(s) in 1.1740 seconds

SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/apache-hive-2.1.0-bin/lib/log4j-over-slf4j-2.1.0.jar!/org.slf4j.impl.Log4jLoggerFactory]
SLF4J: Found binding in [jar:file:/home/acadgild/hadoop-2.7.2/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org.slf4j.impl.Log4jLoggerFactory]
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/apache-hive-2.1.0-bin/lib/log4j-over-slf4j-2.1.0.jar!/org.slf4j.impl.Log4jLoggerFactory
c: true
java.lang.ClassNotFoundException: org.apache.hadoop.hbase.HBaseConfiguration
java.net.FactoryURLStreamHandler
```

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):040:0> list
TABLE
song-artist-map
station-geo-map
subscribed-users
3 row(s) in 0.1250 seconds

=> ["song-artist-map", "station-geo-map", "subscribed-users"]
hbase(main):041:0>
```

The values loaded in the Lookup tables are

shown
below, *song-*
artist-map

```
hbase(main):061:0* scan 'song-artist-map'
ROW                                     COLUMN+CELL
S200                                     column=artist:artistid, timestamp=1516341421411, value=A300
S201                                     column=artist:artistid, timestamp=1516341435130, value=A301
S202                                     column=artist:artistid, timestamp=1516341449406, value=A302
S203                                     column=artist:artistid, timestamp=1516341464984, value=A303
S204                                     column=artist:artistid, timestamp=1516341479845, value=A304
S205                                     column=artist:artistid, timestamp=1516341494365, value=A301
S206                                     column=artist:artistid, timestamp=1516341509536, value=A302
S207                                     column=artist:artistid, timestamp=1516341524259, value=A303
S208                                     column=artist:artistid, timestamp=1516341537840, value=A304
S209                                     column=artist:artistid, timestamp=1516341551721, value=A305
10 row(s) in 0.1250 seconds
```




ACADGILD

station-geo-map

```
hbase(main):062:0> scan 'station-geo-map'
ROW                                     COLUMN+CELL
ST400                                    column=geo:geo_cd, timestamp=1516341188768, value=A
ST401                                    column=geo:geo_cd, timestamp=1516341208229, value=AU
ST402                                    column=geo:geo_cd, timestamp=1516341225914, value=AP
ST403                                    column=geo:geo_cd, timestamp=1516341247762, value=J
ST404                                    column=geo:geo_cd, timestamp=1516341264812, value=E
ST405                                    column=geo:geo_cd, timestamp=1516341278706, value=A
ST406                                    column=geo:geo_cd, timestamp=1516341293480, value=AU
ST407                                    column=geo:geo_cd, timestamp=1516341308185, value=AP
ST408                                    column=geo:geo_cd, timestamp=1516341322088, value=E
ST409                                    column=geo:geo_cd, timestamp=1516341337723, value=E
ST410                                    column=geo:geo_cd, timestamp=1516341351596, value=A
ST411                                    column=geo:geo_cd, timestamp=1516341365274, value=A
ST412                                    column=geo:geo_cd, timestamp=1516341379574, value=AP
ST413                                    column=geo:geo_cd, timestamp=1516341393291, value=J
ST414                                    column=geo:geo_cd, timestamp=1516341407388, value=E
15 row(s) in 0.0830 seconds
```

subscribed-users

```
hbase(main):063:0> scan 'subscribed-users'
ROW                                     COLUMN+CELL
U100                                    column=subscn:enddt, timestamp=1516341581655, value=1465130523
U100                                    column=subscn:startdt, timestamp=1516341566016, value=1465230523
U101                                    column=subscn:enddt, timestamp=1516341609966, value=1475130523
U101                                    column=subscn:startdt, timestamp=1516341596203, value=1465230523
U102                                    column=subscn:enddt, timestamp=1516341639844, value=1475130523
U102                                    column=subscn:startdt, timestamp=1516341625162, value=1465230523
U103                                    column=subscn:enddt, timestamp=1516341668849, value=1475130523
U103                                    column=subscn:startdt, timestamp=1516341654569, value=1465230523
U104                                    column=subscn:enddt, timestamp=1516341698838, value=1475130523
U104                                    column=subscn:startdt, timestamp=1516341684423, value=1465230523
U105                                    column=subscn:enddt, timestamp=1516341726878, value=1475130523
U105                                    column=subscn:startdt, timestamp=1516341713257, value=1465230523
U106                                    column=subscn:enddt, timestamp=1516341756353, value=1485130523
U106                                    column=subscn:startdt, timestamp=1516341740927, value=1465230523
U107                                    column=subscn:enddt, timestamp=1516341785496, value=1455130523
U107                                    column=subscn:startdt, timestamp=1516341770793, value=1465230523
U108                                    column=subscn:enddt, timestamp=1516341815102, value=1465230623
U108                                    column=subscn:startdt, timestamp=1516341800313, value=1465230523
U109                                    column=subscn:enddt, timestamp=1516341843290, value=1475130523
U109                                    column=subscn:startdt, timestamp=1516341829227, value=1465230523
U110                                    column=subscn:enddt, timestamp=1516341871578, value=1475130523
U110                                    column=subscn:startdt, timestamp=1516341857362, value=1465230523
U111                                    column=subscn:enddt, timestamp=1516341900490, value=1475130523
U111                                    column=subscn:startdt, timestamp=1516341886141, value=1465230523
U112                                    column=subscn:enddt, timestamp=1516341929297, value=1475130523
U112                                    column=subscn:startdt, timestamp=1516341914639, value=1465230523
U113                                    column=subscn:enddt, timestamp=1516341958696, value=1485130523
U113                                    column=subscn:startdt, timestamp=1516341944389, value=1465230523
U114                                    column=subscn:enddt, timestamp=1516341988193, value=1468130523
U114                                    column=subscn:startdt, timestamp=1516341973580, value=1465230523
15 row(s) in 0.1170 seconds
```

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.



```
c: true
ACADGILD Java HotSpot(TM) Client VM warning: You have
           stack guard. The VM will try to fix the stack
           It's highly recommended that you fix the lib
           OK
           Time taken: 2.705 seconds
           OK
           Time taken: 0.089 seconds
           OK
           Time taken: 1.689 seconds
           Loading data to table project.users_artists
           OK
           Time taken: 2.168 seconds
[acadgild@localhost scripts]$
```

hive> Select * From users_artists;

```
z) or using Hive 1.X releases.
hive> Show Databases;
OK
default
project
Time taken: 2.468 seconds, Fetched: 2 row(s)
hive> Use project;
OK
Time taken: 0.042 seconds
hive> Show Tables;
OK
users_artists
Time taken: 0.178 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: 3.591 seconds, Fetched: 15 row(s)
hive>
>
```

Now we need to link these 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.

```
1 #!/bin/bash
2
3 batchid=`cat /home/acadgild/project/logs/current-batch.txt`
4 LOGFILE=/home/acadgild/project/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/acadgild/project/scripts/create_hive_hbase_lookup.hql
9
10
```

create_hive_hbase_lookup.hql

```
1 USE project;
2 create external table if not exists station_geo_map
3 (
4   station_id String,
5   geo_cd string
6 )
7 STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
8 with serdeproperties
9 ("hbase.columns.mapping"=:key,geo:geo_cd")
10 tblproperties("hbase.table.name"="station-geo-map");
11
12 create external table if not exists subscribed_users
13 (
14   user_id STRING,
15   subscn_start_dt STRING,
16   subscn_end_dt STRING
17 )
18 STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
19 with serdeproperties
20 ("hbase.columns.mapping"=:key,subscn:startdt,subscn:enddt")
21 tblproperties("hbase.table.name"="subscribed-users");
22
23 create external table if not exists song_artist_map
24 (
25   song_id STRING,
26   artist_id STRING
27 )
28 STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
29 with serdeproperties
30 ("hbase.columns.mapping"=:key,artist:artistid")
31 tblproperties("hbase.table.name"="song-artist-map");
```




Project 2 Music Data Analysis ACADGILD The below screenshot we can see tables getting created in hive by running the **“data_enrichement_filtering_schema.sh file”**

```
[acadgild@localhost scripts]$ ./data_enrichment_filtering_schema.sh
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/apache-hive-2.1.0-bin/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/Slf4jLoggerFactory.class]
SLF4J: Found binding in [jar:file:/home/acadgild/hadoop-2.7.2/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/Slf4jLoggerFactory.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/apache-hive-2.1.0-bin/lib/hive-common-2.1.0.jar!/hive-site.xml: true
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which might corrupt memory. StackGuard detected and will try to fix it now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
OK
Time taken: 1.77 seconds
OK
Time taken: 3.301 seconds
OK
Time taken: 0.305 seconds
OK
Time taken: 0.271 seconds
[acadgild@localhost scripts]$ hive
```

Hive>Show Tables;

```
hive> show tables;
Time taken: 1.485 seconds
hive> show tables;
OK
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 0.513 seconds, Fetched: 4 row(s)
hive>
```

hive>Select * From song_artist_map

```
Time taken: 0.103 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
S209      A305
Time taken: 0.421 seconds, Fetched: 10 row(s)
```



ACADGILD

hive>Select * From station_geo_map

```
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: 0.542 seconds, Fetched: 15 row(s)
```

hive>Select * From Subscribed_users

```
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: 0.643 seconds, Fetched: 15 row(s)
```



ACADGILD

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*

```
1  #!/bin/bash
2
3  batchid=`cat /home/acadgild/project/logs/current-batch.txt`
4  LOGFILE=/home/acadgild/project/logs/log_batch_${batchid}
5
6  echo "Placing data files from local to HDFS..." >> $LOGFILE
7
8  hadoop fs -rm -r /user/acadgild/project/batch${batchid}/web/
9  hadoop fs -rm -r /user/acadgild/project/batch${batchid}/formattedweb/
10 hadoop fs -rm -r /user/acadgild/project/batch${batchid}/mob/
11
12 hadoop fs -mkdir -p /user/acadgild/project/batch${batchid}/web/
13 hadoop fs -mkdir -p /user/acadgild/project/batch${batchid}/mob/
14
15 hadoop fs -put /home/acadgild/project/data/web/* /user/acadgild/project/batch${batchid}/web/
16 hadoop fs -put /home/acadgild/project/data/mob/* /user/acadgild/project/batch${batchid}/mob/
17
18 echo "Running pig script for data formatting..." >> $LOGFILE
19
20 pig -param batchid=${batchid} /home/acadgild/project/scripts/dataformatting.pig
21
22 echo "Running hive script for formatted data load..." >> $LOGFILE
23
24 hive -hiveconf batchid=${batchid} -f /home/acadgild/project/scripts/formatted_hive_load.hql
25
```

```
-rwxrwxr-- 1 acadgild acadgild 412 Jan 19 22:14 wrapper.sh
[acadgild@localhost scripts]$ ./dataformatting.sh
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which might have disabled stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/20 17:58:29 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
18/01/20 17:58:30 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0 minutes, Emptier interval = 0 minutes.
Deleted /user/acadgild/project/batch1/web
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which might have disabled stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/20 17:58:34 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
rm: '/user/acadgild/project/batch1/formattedweb/': No such file or directory
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which might have disabled stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/20 17:58:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
18/01/20 17:58:42 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0 minutes, Emptier interval = 0 minutes.
Deleted /user/acadgild/project/batch1/mob
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which might have disabled stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/20 17:58:46 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```



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

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 batch ID's

Dataformatting.pig

```
1 REGISTER /home/acadgild/project/lib/piggybank.jar;
2
3 DEFINE XPath org.apache.pig.piggybank.evaluation.xml.XPath();
4
5 A = LOAD '/user/acadgild/project/batch${batchid}/web/' using org.apache.pig.piggybank.storage.XMLLoader('record') as (x:chararray);
6
7 B = FOREACH A GENERATE TRIM(XPath(x, 'record/user_id')) AS user_id,
8     TRIM(XPath(x, 'record/song_id')) AS song_id,
9     TRIM(XPath(x, 'record/artist_id')) AS artist_id,
10    ToUnixTime(ToDate(TRIM(XPath(x, 'record/timestamp')),'yyyy-MM-dd HH:mm:ss')) AS timestamp,
11    ToUnixTime(ToDate(TRIM(XPath(x, 'record/start_ts')),'yyyy-MM-dd HH:mm:ss')) AS start_ts,
12    ToUnixTime(ToDate(TRIM(XPath(x, 'record/end_ts')),'yyyy-MM-dd HH:mm:ss')) AS end_ts,
13    TRIM(XPath(x, 'record/geo_cd')) AS geo_cd,
14    TRIM(XPath(x, 'record/station_id')) AS station_id,
15    TRIM(XPath(x, 'record/song_end_type')) AS song_end_type,
16    TRIM(XPath(x, 'record/like')) AS like,
17    TRIM(XPath(x, 'record/dislike')) AS dislike;
18
19 STORE B INTO '/user/acadgild/project/batch${batchid}/formattedweb/' USING PigStorage(',');
20
```

formatted_hive_load.hql

```
1 set hive.support.sql11.reserved.keywords=false;
2 USE project;
3
4 CREATE TABLE IF NOT EXISTS formatted_input
5 (
6     user_id STRING,
7     song_id STRING,
8     artist_id STRING,
9     timestamp STRING,
10    start_ts STRING,
11    end_ts STRING,
12    geo_cd STRING,
13    station_id STRING,
14    song_end_type INT,
15    like INT,
16    dislike INT
17 )
18 PARTITIONED BY
19     (batchid INT)
20 ROW FORMAT DELIMITED
21 FIELDS TERMINATED BY ',';
22
23 LOAD DATA INPATH '/user/acadgild/project/batch${hiveconf:batchid}/formattedweb/'
24 INTO TABLE formatted_input PARTITION (batchid=${hiveconf:batchid});
25
26 LOAD DATA INPATH '/user/acadgild/project/batch${hiveconf:batchid}/mob/';
27 INTO TABLE formatted_input PARTITION (batchid=${hiveconf:batchid});
28
```



Project 2 M-Music Data Analysis ACADGILD 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,

```
2018-01-20 18:00:32,460 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.reduce.tasks is deprecated. Instead, use mapreduce.job.reduces
2018-01-20 18:00:32,463 [main] INFO org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2018-01-20 18:00:32,478 [main] INFO org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCESS
2018-01-20 18:00:32,478 [main] INFO org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCESS
2018-01-20 18:00:32,820 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 100% complete
2018-01-20 18:00:32,829 [main] INFO org.apache.pig.tools.pigstats.mapreduce.SimplePigStats - Script Statistics:
HadoopVersion PigVersion UserId StartedAt FinishedAt Features
2.5.1 0.16.0 acadgild 2018-01-20 17:59:26 2018-01-20 18:00:32 UNKNOWN

Success!
Job Stats (time in seconds):
JobId Maps Reduces MaxMapTime MinMapTime AvgMapTime MedianMapTime MaxReduceTime MinReduceTime AvgReduceTime MedianR
edutime Alias Outputs
job_1516450284102_0001 1 0 18 18 18 18 0 0 0 A,B MAP_ONLY /user/acadgild/project/batch1/formattedweb,
Input(s):
Successfully read 20 records (7111 bytes) from: "/user/acadgild/project/batch1/web"
Output(s):
Successfully stored 20 records (1241 bytes) in: "/user/acadgild/project/batch1/formattedweb"
Counters:
Total records written : 20
Total bytes written : 1241
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0
Job DAG:
job_1516450284102_0001

2018-01-20 18:00:32,833 [main] INFO org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2018-01-20 18:00:32,847 [main] INFO org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCESS
2018-01-20 18:00:32,847 [main] INFO org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCESS
2018-01-20 18:00:32,847 [main] INFO org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCESS
2018-01-20 18:00:32,847 [main] INFO org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCESS
```

Hive script successfully load the data into hive terminal,

```
er.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/apache-hive-
c: true
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hado
stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>',
OK
Time taken: 2.627 seconds
OK
Time taken: 2.714 seconds
Loading data to table project.formatted_input partition (batchid=1)
OK
Time taken: 4.048 seconds
Loading data to table project.formatted_input partition (batchid=1)
OK
Time taken: 1.662 seconds
[acadgild@localhost scripts]$
```

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:



```
drwxr-xr-x - acadgild supergroup          0 2018-01-20 16:29 project
[acadgild@localhost ~]$ hadoop fs -ls /user/acadgild/project
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which
stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/20 19:05:34 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-jav
able
Found 1 items
drwxr-xr-x - acadgild supergroup          0 2018-01-20 18:12 /user/acadgild/project/batch1
[acadgild@localhost ~]$ hadoop fs -ls /user/acadgild/project/batch1
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which
stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/20 19:05:47 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-jav
able
Found 3 items
drwxr-xr-x - acadgild supergroup          0 2018-01-20 18:12 /user/acadgild/project/batch1/formattedweb
drwxr-xr-x - acadgild supergroup          0 2018-01-20 18:12 /user/acadgild/project/batch1/mob
drwxr-xr-x - acadgild supergroup          0 2018-01-20 18:11 /user/acadgild/project/batch1/web
[acadgild@localhost ~]$
[acadgild@localhost ~]$
[acadgild@localhost ~]$
[acadgild@localhost ~]$
[acadgild@localhost ~]$ hadoop fs -ls /user/acadgild/project/batch1/formattedweb
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which
stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/20 19:07:25 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-jav
able
Found 2 items
-rw-r--r-- 1 acadgild supergroup          0 2018-01-20 18:12 /user/acadgild/project/batch1/formattedweb/_SUCCESS
-rw-r--r-- 1 acadgild supergroup        1241 2018-01-20 18:12 /user/acadgild/project/batch1/formattedweb/part-m-00000
[acadgild@localhost ~]$
```

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/*

```
[acadgild@localhost ~]$
[acadgild@localhost ~]$ hadoop fs -cat /user/acadgild/project/batch1/formattedweb/*
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7
stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or li
18/01/20 19:09:24 WARN util.NativeCodeLoader: Unable to load native-hadoop library fo
able
U113,S205,A305,1462863262,1465490556,1462863262,AP,ST407,3,0,1
U102,S200,A301,1494297562,1465490556,1465490556,A,ST400,1,0,1
U115,S207,A301,1494297562,1468094889,1465490556,AU,ST406,2,1,1
U110,S201,A300,1468094889,1462863262,1468094889,AU,ST413,2,0,1
U102,S203,A305,1465490556,1494297562,1465490556,A,ST414,2,0,0
,S209,A304,1465490556,1462863262,1465490556,E,ST412,0,0,1
U105,S203,A300,1462863262,1468094889,1468094889,U,ST407,2,1,1
U113,S205,A303,1462863262,1468094889,1468094889,E,ST415,2,0,1
U120,S205,A302,1494297562,1494297562,1494297562,,ST400,0,1,0
U105,S210,,1468094889,1462863262,1494297562,E,ST410,1,0,1
U117,S206,A300,1468094889,1468094889,1465490556,A,ST414,2,0,0
U114,S200,A301,1462863262,1468094889,1462863262,AP,ST408,1,1,1
U110,S208,A303,1494297562,1468094889,1468094889,E,ST405,1,0,1
U115,S201,A303,1465490556,1465490556,1494297562,AU,ST407,2,1,1
U103,S209,A305,1465490556,1468094889,1468094889,AU,ST408,3,0,1
U112,S210,A303,1494297562,1494297562,1462863262,AU,ST408,2,1,0
U118,S202,A301,1468094889,1465490556,1468094889,AP,ST414,0,0,1
U100,S200,A301,1462863262,1494297562,1494297562,AU,ST408,2,0,0
U113,S210,A304,1468094889,1465490556,1494297562,E,ST403,2,0,1
U104,S203,A300,1468094889,1468094889,1468094889,1494297562,AU,ST406,1,0,1
[acadgild@localhost ~]$
[acadgild@localhost ~]$
```



Project 2 Music Data Analysis

The new Tables has been created and show below,

```
/* 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)
```

ACADGID
LD

DataFormatting.sh output in hive terminal,

hive> select * from formatted_input;

```
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
U209  S201   A301   1465230523   1465230523   1465130523   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
U209  S204   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   1462863262   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**, **artist_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**.



ACADGILD

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 corresponding lookup values in **song-artist-map** and **station-geo-map**

artist_id values with the help of tables in **Hive-Hbase** tables.

data_enrichment.sh

```
1  #!/bin/bash
2
3  batchid=`cat /home/acadgild/project/logs/current-batch.txt`
4  LOGFILE=/home/acadgild/project/logs/log_batch_$batchid
5  VALIDDIR=/home/acadgild/project/processed_dir/valid/batch_$batchid
6  INVALIDDIR=/home/acadgild/project/processed_dir/invalid/batch_$batchid
7
8  echo "Running hive script for data enrichment and filtering..." >> $LOGFILE
9
10 hive -hiveconf batchid=$batchid -f /home/acadgild/project/scripts/data_enrichment.hql
11
12 if [ ! -d "$VALIDDIR" ]
13 then
14     mkdir -p "$VALIDDIR"
15 fi
16
17 if [ ! -d "$INVALIDDIR" ]
18 then
19     mkdir -p "$INVALIDDIR"
20 fi
21
22 echo "Copying valid and invalid records in local file system..." >> $LOGFILE
23
24 hadoop fs -get /user/hive/warehouse/project.db/enriched_data/batchid=$batchid/status=pass/* $VALIDDIR
25 hadoop fs -get /user/hive/warehouse/project.db/enriched_data/batchid=$batchid/status=fail/* $INVALIDDIR
26
27 echo "Deleting older valid and invalid records from local file system..." >> $LOGFILE
28
29 find /home/acadgild/project/processed_dir/ -mtime +7 -exec rm {} \;
```




ACADGILD

data_enrichment.hql

```
1  set hive.support.sql11.reserved.keywords=false;
2  SET hive.auto.convert.join=false;
3  SET hive.exec.dynamic.partition.mode=nonstrict;
4
5  USE project;
6
7  CREATE TABLE IF NOT EXISTS enriched_data
8  (
9    user_id STRING,
10   song_id STRING,
11   artist_id STRING,
12   timestamp STRING,
13   start_ts STRING,
14   end_ts STRING,
15   geo_cd STRING,
16   station_id STRING,
17   song_end_type INT,
18   like INT,
19   dislike INT
20 )
21 PARTITIONED BY
22   (batchid INT,
23    status STRING)
24 STORED AS ORC;
25
26 INSERT OVERWRITE TABLE enriched_data
27 PARTITION (batchid,status)
28 SELECT
29   i.user_id,
30   i.song_id,
31   IF(i.artist_id is NULL OR i.artist_id='',sa.artist_id,i.artist_id) AS artist_id,
32   i.timestamp,
33   i.start_ts,
34   i.end_ts,
35   IF(i.geo_cd is NULL OR i.geo_cd='',sg.geo_cd,i.geo_cd) AS geo_cd,
36   i.station_id,
37   IF (i.song_end_type IS NULL,3,i.song_end_type) AS song_end_type,
38   IF (i.like IS NULL,0,i.like) AS like,
39   IF (i.dislike IS NULL,0,i.dislike) AS dislike,
40   i.batchid,
41   IF((i.like=1 AND i.dislike=1)
42   OR i.user_id IS NULL
43   OR i.song_id IS NULL
44   OR i.timestamp IS NULL
45   OR i.start_ts IS NULL
46   OR i.end_ts IS NULL
47   OR i.user_id=''
48   OR i.song_id=''
49   OR i.timestamp=''
50   OR i.start_ts=''
51   OR i.end_ts=''
52   OR sg.geo_cd=''
53   OR sg.geo_cd IS NULL
54   OR sa.artist_id IS NULL
55   OR sa.artist_id='', 'fail', 'pass') AS status
56   FROM formatted_input i
57   LEFT OUTER JOIN station_geo_map sg ON i.station_id = sg.station_id
58   LEFT OUTER JOIN song_artist_map sa ON i.song_id = sa.song_id
59   WHERE i.batchid=${hiveconf:batchid};
```



```
[acadgild@localhost scripts]$ ./data_enrichment.sh
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/apache-hive-2.1.0-bin/lib/log4j-slf4j-impl
SLF4J: Found binding in [jar:file:/home/acadgild/hadoop-2.7.2/share/hadoop/common/lib/slf4j
er.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/apache-hive-2.1.0-bin/li
c: true
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib
stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it
OK
Time taken: 2.344 seconds
OK
Time taken: 1.592 seconds
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions
spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180121050629_4da8c068-b197-457a-8f78-6cdle80c34b7
Total jobs = 2
```

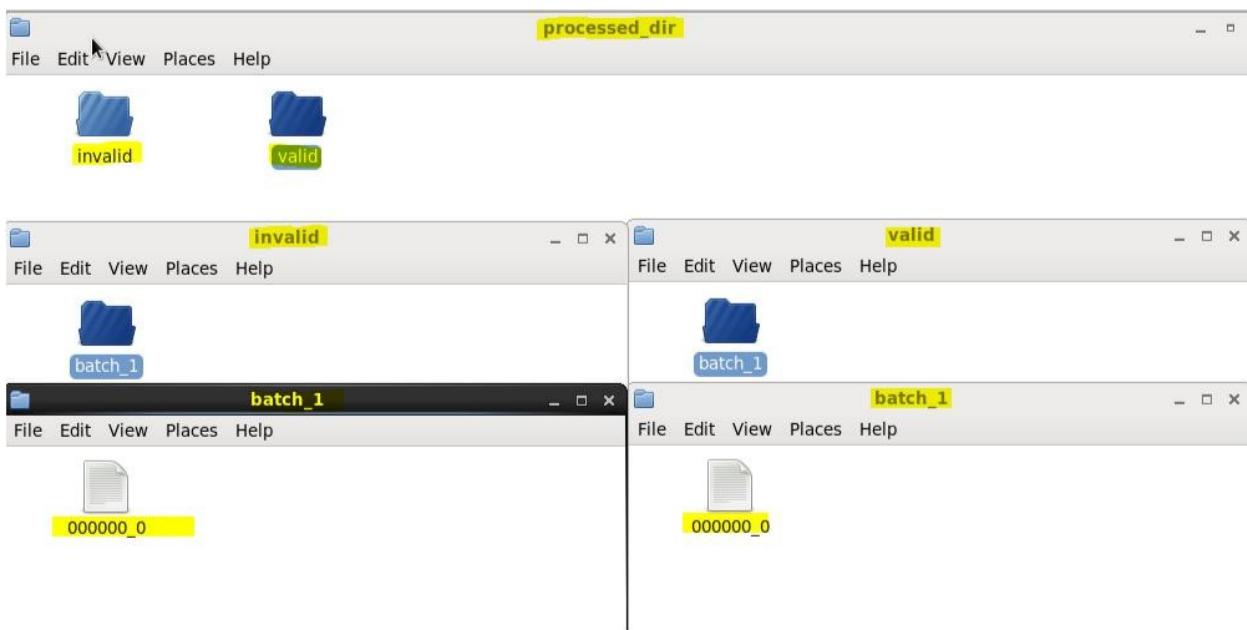
```
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1516485910189_0006, Tracking URL = http://localhost:8088/proxy/application_1516485910189_0006/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job -kill job_1516485910189_0006
Hadoop job information for Stage-2: number of mappers: 2; number of Reducers: 1
2018-01-21 05:08:49,177 Stage-2 map = 0%, reduce = 0%
2018-01-21 05:09:12,849 Stage-2 map = 50%, reduce = 0%, Cumulative CPU 2.2 sec
2018-01-21 05:09:15,182 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 5.74 sec
2018-01-21 05:09:31,414 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 9.86 sec
MapReduce Total cumulative CPU time: 9 seconds 860 msec
Ended Job = job_1516485910189_0006
Loading data to table project.enriched_data partition (batchid=null, status=null)

Loaded : 2/2 partitions.
  Time taken to load dynamic partitions: 1.231 seconds
  Time taken for adding to write entity : 0.004 seconds
MapReduce Jobs Launched:
Stage-Stage-1: Map: 3  Reduce: 1  Cumulative CPU: 14.16 sec  HDFS Read: 50542 HDFS Write: 3280 SUCCESS
Stage-Stage-2: Map: 2  Reduce: 1  Cumulative CPU: 9.86 sec  HDFS Read: 25154 HDFS Write: 3177 SUCCESS
Total MapReduce CPU Time Spent: 24 seconds 20 msec
OK
Time taken: 182.412 seconds
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 wh
stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/21 05:09:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-j
able
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 wh
stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/21 05:09:45 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-j
able
```

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
ACADGILD
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.

```
hive> use project;
OK
Time taken: 2.773 seconds
hive> show tables;
OK
enriched_data
formatted_input
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 1.291 seconds, Fetched: 6 row(s)
hive> select * from enriched_data;
```

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,



ACADGILD

hive>select * From enriched_data;

OK																
U114	S200	A301	1462863262	1468094889	1462863262	AP	ST408	1	1	1	1	1	1	1	1	fail
U118	S201	A304	1465230523	1475130523	1485130523	A	ST408	2	1	1	1	1	1	1	1	fail
U115	S201	A303	1465490556	1465490556	1494297562	AU	ST407	2	1	1	1	1	1	1	1	fail
U113	S202	A300	1465130523	1475130523	1475130523	U	ST415	1	1	0	1	1	1	1	1	fail
U105	S203	A300	1462863262	1468094889	1468094889	U	ST407	2	1	1	1	1	1	1	1	fail
U113	S205	A303	1462863262	1468094889	1468094889	E	ST415	2	0	1	1	1	1	1	1	fail
U101	S206	A305	1465130523	1465230523	1465230523	AP	ST415	3	0	0	0	0	1	1	1	fail
U104	S206	A303	1495130523	1465130523	1475130523	U	ST401	1	1	1	1	1	1	1	1	fail
U112	S207	A302	1465230523	1465230523	1475130523	AU	ST410	0	1	1	1	1	1	1	1	fail
U103	S207	A303	1465230523	1465130523	1465130523	A	ST400	1	1	1	1	1	1	1	1	fail
U119	S207	A301	1465230523	1475130523	1485130523	AU	ST408	1	1	1	1	1	1	1	1	fail
U115	S207	A301	1494297562	1468094889	1465490556	AU	ST406	2	1	1	1	1	1	1	1	fail
U118	S208	A304	1465130523	1475130523	1465130523	E	ST410	0	1	1	1	1	1	1	1	fail
S209	A301	1465230523	1465230523	1485130523	U	ST411	3	0	1	1	1	1	1	1	1	fail
S209	A304	1465490556	1462863262	1465490556	E	ST412	0	0	1	1	1	1	1	1	1	fail
U113	S210	A304	1468094889	1465490556	1494297562	E	ST403	2	0	1	1	1	1	1	1	fail
U105	S210	NULL	1468094889	1462863262	1494297562	E	ST410	1	0	1	1	1	1	1	1	fail
U112	S210	A303	1494297562	1494297562	1462863262	AU	ST408	2	1	0	1	1	1	1	1	fail
U102	S200	A301	1494297562	1465490556	1465490556	A	ST400	1	0	1	1	1	1	1	1	pass
U108	S200	A300	1495130523	1475130523	1465230523	U	ST400	1	0	1	1	1	1	1	1	pass
U100	S200	A301	1462863262	1494297562	1494297562	AU	ST408	2	0	0	0	1	1	1	1	pass
U110	S201	A300	1468094889	1462863262	1468094889	AU	ST413	2	0	1	1	1	1	1	1	pass
U110	S202	A303	1495130523	1465130523	1465130523	AP	ST413	0	0	1	1	1	1	1	1	pass
U118	S202	A301	1468094889	1465490556	1468094889	AP	ST414	0	0	1	1	1	1	1	1	pass
U118	S203	A304	1475130523	1465130523	1465230523	U	ST403	0	0	0	0	1	1	1	1	pass
U104	S203	A300	1468094889	1468094889	1494297562	AU	ST406	1	0	1	1	1	1	1	1	pass
U115	S203	A305	1465230523	1465130523	1475130523	AP	ST409	0	1	0	1	1	1	1	1	pass
U102	S203	A305	1465490556	1494297562	1465490556	A	ST414	2	0	0	0	1	1	1	1	pass
U101	S204	A301	1475130523	1485130523	1485130523	A	ST411	2	0	1	1	1	1	1	1	pass
U117	S204	A301	1495130523	1465130523	1475130523	A	ST402	0	1	0	1	1	1	1	1	pass
U113	S205	A305	1462863262	1465490556	1462863262	AP	ST407	3	0	1	1	1	1	1	1	pass
U120	S205	A302	1494297562	1494297562	1494297562	A	ST400	0	1	0	1	1	1	1	1	pass
U117	S206	A300	1468094889	1468094889	1465490556	A	ST414	2	0	0	0	1	1	1	1	pass
U111	S206	A303	1465230523	1485130523	1465130523	U	ST414	1	0	0	0	1	1	1	1	pass
U113	S207	A305	1495130523	1465130523	1485130523	AU	ST402	0	0	1	1	1	1	1	1	pass
U117	S208	A305	1465130523	1465130523	1465130523	AP	ST407	3	0	1	1	1	1	1	1	pass
U110	S208	A303	1494297562	1468094889	1468094889	E	ST405	1	0	1	1	1	1	1	1	pass
U105	S208	A300	1465130523	1475130523	1465230523	AU	ST410	1	0	0	0	1	1	1	1	pass
U103	S209	A305	1465490556	1468094889	1468094889	AU	ST408	3	0	1	1	1	1	1	1	pass
U118	S209	A305	1475130523	1465230523	1465230523	E	ST400	0	0	0	0	1	1	1	1	pass

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



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.

- Spark_analysis for creating tables for each query/problem statement.
- Spark_analysis_2 for displaying results for each query in terminal.

DataAnalysis.sh

```

1 #!/bin/bash
2
3 batchid=`cat /home/acadgild/project/logs/current-batch.txt`
4
5 LOGFILE=/home/acadgild/project/logs/log_batch_$batchid
6
7 echo "Running script for data analysis using spark..." >> $LOGFILE
8 chmod 775 /home/acadgild/project/lib/sparkanalysis.jar
9
10 zip -d /home/acadgild/project/lib/sparkanalysis.jar META-INF/*.DSA META-INF/*.RSA META-INF/*.SF
11
12 /home/acadgild/spark-2.2.1-bin-hadoop2.7/bin/spark-submit \
13 --class Spark_analysis \
14 --master local[2] \
15 --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/* \
16 /home/acadgild/project/lib/sparkanalysis.jar $batchid
17
18 /home/acadgild/spark-2.2.1-bin-hadoop2.7/bin/spark-submit \
19 --class Spark_analysis_2 \
20 --master local[2] \
21 --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/* \
22 /home/acadgild/project/lib/sparkanalysis.jar $batchid
23
24 echo "Exporting data to MySQL using sqoop export..." >> $LOGFILE
25 sh /home/acadgild/project/scripts/data_export.sh
26
27 echo "Incrementing batchid..." >> $LOGFILE
28 batchid=`expr $batchid + 1`
29 echo -n $batchid > /home/acadgild/project/logs/current-batch.txt
30
31
32

```

Spark_analysis.scala

```

1 import org.apache.hadoop.hive.serde2.`lazy`.LazySimpleSerDe
2 import org.apache.spark.sql.SparkSession
3
4 object Spark_analysis {
5
6   def main(args: Array[String]): Unit = {
7     val sparkSession = SparkSession.builder()
8       .master("local[2]")
9       .appName("Data Analysis Main_1")
10      .config("spark.sql.warehouse.dir","/user/hive/warehouse")
11      .config("hive.metastore.uris","thrift://127.0.0.1:9083")
12      .enableHiveSupport()
13      .getOrCreate()
14
15   val batchId = args(0)
16
17   //<<<<<----- PROBLEM 1 - Creation of table and Insertion of data ----->>>>>>>
18   //Determine top 10 station_id(s) where maximum number of songs were played, which were liked by unique users.
19
20   val set_properties = sparkSession.sqlContext.sql("set hive.auto.convert.join=false")
21
22   val use_project_database = sparkSession.sqlContext.sql("USE project")
23
24   val create_hive_table_top_10_stations = sparkSession.sqlContext.sql("CREATE TABLE IF NOT EXISTS project.top_10_stations"+
25   "("+
26   " station_id STRING,"+
27   " total_distinct_songs_played INT,"+
28   " distinct_user_count INT"+
29   ")" +
30   " PARTITIONED BY (batchid INT)" +
31   " ROW FORMAT DELIMITED" +
32   " FIELDS TERMINATED BY ','" +
33   " STORED AS TEXTFILE")

```



Project 2 Music Data Analysis

ACADGILD

```
36  
37 val insert_into_top_10_stations = sparkSession.sqlContext.sql("INSERT OVERWRITE TABLE project.top_10_stations"+  
38 " PARTITION (batchid=$batchId)+"  
39 " SELECT"+  
40 " station_id,"+  
41 " COUNT(DISTINCT song_id) AS total_distinct_songs_played,"+  
42 " COUNT(DISTINCT user_id) AS distinct_user_count"+  
43 " FROM project.enriched_data"+  
44 " WHERE status='pass'"+  
45 " AND (batchid=$batchId)+"  
46 " AND like=1"+  
47 " GROUP BY station_id"+  
48 " ORDER BY total_distinct_songs_played DESC"+  
49 " LIMIT 10")  
50  
51 //<<<<<----- PROBLEM 2 - Creation of table and Insertion of data ----->>>>>>>>>  
52 /*Determine total duration of songs played by each type of user, where type of user can be 'subscribed' or 'unsubscribed'.  
53 An unsubscribed user is the one whose record is either not present in Subscribed_users lookup table or has subscription_end_date  
54 earlier than the timestamp of the song played by him.*/  
55  
56 val create_hive_table_song_duration = sparkSession.sqlContext.sql("CREATE TABLE IF NOT EXISTS project.song_duration"+  
57 " ("+  
58 " user_id STRING,"+  
59 " user_type STRING,"+  
60 " song_id STRING,"+  
61 " artist_id STRING,"+  
62 " total_duration_in_minutes DOUBLE"+  
63 " )"+  
64 " PARTITIONED BY (batchid INT)" +  
65 " ROW FORMAT DELIMITED"+  
66 " FIELDS TERMINATED BY ','"+  
67 " STORED AS TEXTFILE")  
68  
69  
70 val insert_into_song_duration = sparkSession.sqlContext.sql("INSERT OVERWRITE TABLE project.song_duration"+  
71 " s" PARTITION (batchid=$batchId)+"  
72 " SELECT"+  
73 " e.user_id STRING,"+  
74 " IF(e.user_id!=s.user_id"+  
75 " OR (CAST(s.subscription_end_dt as BIGINT) < CAST(e.start_ts as BIGINT)), 'unsubscribed', 'subscribed') AS user_type,"+  
76 " e.song_id STRING,"+  
77 " e.artist_id STRING,"+  
78 " (cast(e.end_ts as BIGINT)-cast(e.start_ts as BIGINT))/60 AS total_duration_in_minutes"+  
79 " FROM project.enriched_data e"+  
80 " LEFT OUTER JOIN project.subscribed_users s"+  
81 " ON e.user_id=s.user_id"+  
82 " WHERE e.status='pass'"+  
83 " AND (batchid=$batchId)")  
84  
85 //<<<<<----- PROBLEM 3 - Creation of table and Insertion of data ----->>>>>>>>  
86 //Determine top 10 connected artists.  
87 //Connected artists are those whose songs are most listened by the unique users who follow them.  
88  
89 val create_hive_table_top_10_connected_artists = sparkSession.sqlContext.sql("CREATE TABLE IF NOT EXISTS project.connected_artists"+  
90 " ("+  
91 " artist_id STRING,"+  
92 " total_distinct_songs INT,"+  
93 " unique_followers INT"+  
94 " )"+  
95 " PARTITIONED BY (batchid INT)" +  
96 " ROW FORMAT DELIMITED"+  
97 " FIELDS TERMINATED BY ','"+  
98 " STORED AS TEXTFILE")  
99  
100  
101 val insert_into_top_10_connected_artists = sparkSession.sqlContext.sql("INSERT OVERWRITE TABLE project.connected_artists"+  
102 " s" PARTITION (batchid=$batchId)+"  
103 " SELECT"+  
104 " artist_id,"+  
105 " COUNT(DISTINCT song_id) AS total_distinct_songs,"+  
106 " COUNT(DISTINCT user_id) AS unique_followers"+  
107 " FROM project.enriched_data"+  
108 " WHERE status='pass'"+  
109 " AND (batchid=$batchId)+"  
110 " GROUP BY artist_id"+  
111 " ORDER BY unique_followers desc,total_distinct_songs desc"+  
112 " LIMIT 10")  
113  
114  
115 //<<<<<----- PROBLEM 4 - Creation of table and Insertion of data ----->>>>>>>  
116 //Determine top 10 songs who have generated the maximum revenue.  
117 //NOTE: Royalty applies to a song only if it was liked or was completed successfully or both.  
118  
119 val create_hive_table_top_10_songs_maxrevenue = sparkSession.sqlContext.sql("CREATE TABLE IF NOT EXISTS project.top_10_songs_maxrevenue"+  
120 " ("+  
121 " song_id STRING,"+  
122 " artist_id STRING,"+  
123 " total_duration_in_minutes DOUBLE"+  
124 " )"+  
125 " PARTITIONED BY (batchid INT)" +  
126 " ROW FORMAT DELIMITED"+  
127 " FIELDS TERMINATED BY ','"+  
128 " STORED AS TEXTFILE")  
129
```



```
131  
132     s" PARTITION (batchid=$batchId)" +  
133     " SELECT"  
134     " song_id," +  
135     " artist_id," +  
136     " (cast(end_ts as BIGINT)-cast(start_ts as BIGINT))/60 AS total_duration_in_minutes" +  
137     " FROM project.enriched_data" +  
138     " WHERE status='pass'" +  
139     " AND (batchid=$batchId)" +  
140     " AND (like=1 OR song_end_type=0 OR (like=1 and song_end_type=0))" +  
141     " ORDER BY total_duration_in_minutes desc" +  
142     " LIMIT 10")  
143  
144  
145 //<<<<<----- PROBLEM 5 - Creation of table and Insertion of data ----->>>>>  
146 //Determine top 10 unsubscribed users who listened to the songs for the longest duration.  
147  
148 val create_hive_table_top_10_unsubscribed_users = sparkSession.sqlContext.sql("CREATE TABLE IF NOT EXISTS project.top_10_unsubscribed_users"+  
149     "("+  
150     " user_id STRING," +  
151     " song_id STRING," +  
152     " artist_id STRING," +  
153     " total_duration_in_minutes DOUBLE" +  
154     ")" +  
155     " PARTITIONED BY (batchid INT)" +  
156     " ROW FORMAT DELIMITED" +  
157     " FIELDS TERMINATED BY ','" +  
158     " STORED AS TEXTFILE")  
159
```

```
145 //<<<<<----- PROBLEM 5 - Creation of table and Insertion of data ----->>>>>  
146 //Determine top 10 unsubscribed users who listened to the songs for the longest duration.  
147  
148 val create_hive_table_top_10_unsubscribed_users = sparkSession.sqlContext.sql("CREATE TABLE IF NOT EXISTS project.top_10_unsubscribed_users"+  
149     "("+  
150     " user_id STRING," +  
151     " song_id STRING," +  
152     " artist_id STRING," +  
153     " total_duration_in_minutes DOUBLE" +  
154     ")" +  
155     " PARTITIONED BY (batchid INT)" +  
156     " ROW FORMAT DELIMITED" +  
157     " FIELDS TERMINATED BY ','" +  
158     " STORED AS TEXTFILE")  
159  
160  
161 val insert_into_unsubscribed_users = sparkSession.sqlContext.sql("INSERT OVERWRITE TABLE project.top_10_unsubscribed_users"+  
162     s" PARTITION (batchid=$batchId)" +  
163     " SELECT" +  
164     " user_id," +  
165     " song_id," +  
166     " artist_id," +  
167     " total_duration_in_minutes" +  
168     " FROM project.song_duration" +  
169     " WHERE user_type='unsubscribed'" +  
170     " AND total_duration_in_minutes>=0" +  
171     s" AND (batchid=$batchId)" +  
172     " ORDER BY total_duration_in_minutes desc" +  
173     " LIMIT 10")  
174  
175  
176 }  
177  
178 }
```

Spark_analysis_2.scala



```
1 package sparkanalysis
2 ACADGILD
3 import org.apache.spark.{SparkConf, SparkContext}
4 import org.apache.spark.sql.SparkSession
5 object Spark_analysis_2 {
6   def main(args: Array[String]): Unit = {
7     val sparkSession = SparkSession.builder.master("local").appName("Spark Session example")
8       .config("spark.sql.warehouse.dir", "/user/hive/warehouse")
9       .config("hive.metastore.uris", "thrift://localhost:9083")
10      .enableHiveSupport().getOrCreate()
11      val batchId = args(0)
12
13      sparkSession.sqlContext.sql("USE project")
14      sparkSession.sqlContext.sql("SELECT station_id from top_10_stations").show()
15      sparkSession.sqlContext.sql("SELECT user_type, total_duration_in_minutes from song_duration").show()
16      sparkSession.sqlContext.sql("SELECT artist_id from connected_artists").show()
17      sparkSession.sqlContext.sql("SELECT song_id from top_10_songs_maxrevenue").show()
18      sparkSession.sqlContext.sql("SELECT user_id from top_10_unsubscribed_users").show()
19
20
21   }
22 }
```

```
[acadgild@localhost scripts]$ sh DataAnalysis.sh
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/hbase-1.0.3/lib/slf4j-log4j12-1.7.7.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/spark-2.2.1-bin-hadoop2.7/jars/slf4j-log4j12-1.7.16.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
18/01/22 11:38:36 INFO SparkContext: Running Spark version 2.2.1
18/01/22 11:38:37 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
18/01/22 11:38:38 WARN Utils: Your hostname, localhost.localdomain resolves to a loopback address: 127.0.0.1; using 10.0.2.15 instead (on interface eth0)
18/01/22 11:38:38 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
18/01/22 11:38:38 INFO SparkContext: Submitted application: Data Analysis Main_1
18/01/22 11:38:38 INFO SecurityManager: Changing view acls to: acadgild
18/01/22 11:38:38 INFO SecurityManager: Changing modify acls to: acadgild
18/01/22 11:38:38 INFO SecurityManager: Changing view acls groups to:
18/01/22 11:38:38 INFO SecurityManager: Changing modify acls groups to:
18/01/22 11:38:38 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(acadgild); groups with view permissions: Set(); users with modify permissions: Set(acadgild); groups with modify permissions: Set()
18/01/22 11:38:39 INFO Utils: Successfully started service 'sparkDriver' on port 52909.
18/01/22 11:38:39 INFO SparkEnv: Registering MapOutputTracker
18/01/22 11:38:39 INFO SparkEnv: Registering BlockManagerMaster
18/01/22 11:38:39 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
18/01/22 11:38:39 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
18/01/22 11:38:39 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-0343769b-ba6b-4405-b875-d4935ce4d38d
18/01/22 11:38:40 INFO MemoryStore: MemoryStore started with capacity 413.9 MB
18/01/22 11:38:40 INFO SparkEnv: Registering OutputCommitCoordinator
18/01/22 11:38:40 INFO Utils: Successfully started service 'SparkUI' on port 4040.
18/01/22 11:38:41 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://10.0.2.15:4040
18/01/22 11:38:41 INFO SparkContext: Added JAR file:/home/acadgild/project/lib/sparkanalysis.jar at spark://10.0.2.15:52909/jars/sparkanalysis.jar with timestamp 1516601321290
18/01/22 11:38:41 INFO Executor: Starting executor ID driver on host localhost
18/01/22 11:38:41 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 60285.
```




```
Time taken: 1.729 seconds
OK
Time taken: 13.799 seconds
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180122102731_b67f3e6a-6470-44f7-bc68-6e98ef39b68d
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_1516485910189_0019, Tracking URL = http://localhost:8088/proxy/application_1516485910189_0019/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job -kill job_1516485910189_0019
Hadoop job information for Stage-1: number of mappers: 0; number of Reducers: 1
2018-01-22 10:28:08,714 Stage-1 map = 0%, reduce = 0%
2018-01-22 10:28:24,894 Stage-1 map = 0%, reduce = 100%, Cumulative CPU 2.05 sec
MapReduce Total cumulative CPU time: 2 seconds 50 msec
Ended Job = job_1516485910189_0019
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1516485910189_0020, Tracking URL = http://localhost:8088/proxy/application_1516485910189_0020/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job -kill job_1516485910189_0020
Hadoop job information for Stage-2: number of mappers: 1; number of Reducers: 1
2018-01-22 10:28:49,750 Stage-2 map = 0%, reduce = 0%
2018-01-22 10:29:08,673 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 3.85 sec
2018-01-22 10:29:29,196 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 7.65 sec
MapReduce Total cumulative CPU time: 7 seconds 650 msec
Ended Job = job_1516485910189_0020
Loading data to table project.top_10_stations partition (batchid=7)
```

```
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_1516485910189_0021, Tracking URL = http://localhost:8088/proxy/application_1516485910189_0021/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job -kill job_1516485910189_0021
Hadoop job information for Stage-1: number of mappers: 1; number of Reducers: 1
2018-01-22 10:30:06,890 Stage-1 map = 0%, reduce = 0%
2018-01-22 10:30:30,036 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 5.88 sec
2018-01-22 10:30:47,380 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 9.78 sec
MapReduce Total cumulative CPU time: 9 seconds 780 msec
Ended Job = job_1516485910189_0021
Launching Job 2 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1516485910189_0022, Tracking URL = http://localhost:8088/proxy/application_1516485910189_0022/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job -kill job_1516485910189_0022
Hadoop job information for Stage-2: number of mappers: 1; number of Reducers: 1
2018-01-22 10:31:15,265 Stage-2 map = 0%, reduce = 0%
2018-01-22 10:31:26,602 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 1.92 sec
2018-01-22 10:31:42,859 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 5.33 sec
MapReduce Total cumulative CPU time: 5 seconds 330 msec
Ended Job = job_1516485910189_0022
Loading data to table project.users_behaviour partition (batchid=7)
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1  Cumulative CPU: 9.78 sec  HDFS Read: 21286 HDFS Write: 96 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1  Cumulative CPU: 5.33 sec  HDFS Read: 6760 HDFS Write: 59 SUCCESS
Total MapReduce CPU Time Spent: 15 seconds 110 msec
OK
Time taken: 132.863 seconds
OK
Time taken: 0.138 seconds
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e.
```

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

```
scala> val result
+-----+
|station_id|
+-----+
|    ST407|
|    ST414|
|    ST411|
|    ST402|
|    ST406|
|    ST405|
+-----+
```



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.

user_type	duration
SUBSCRIBED	93861594
UNSUBSCRIBED	105594881

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

artist_id
A303
A302
A300

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

song_id
S208
S207
S206
S209
S200
S204
S202
S205

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

user_id
U117
U118
U110
U120
U115
U107
U108
U109
U106
U100



Table Creation in HIVE and Data analysis using HIVE,

```
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_1516485910189_0029, Tracking URL = http://localhost:8088/proxy/application_1516485910189_0029/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job -kill job_1516485910189_0029
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-01-22 10:37:36,483 Stage-2 map = 0%,  reduce = 0%
2018-01-22 10:37:50,781 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 2.26 sec
2018-01-22 10:38:06,474 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 4.34 sec
MapReduce Total cumulative CPU time: 4 seconds 340 msec
Ended Job = job_1516485910189_0029
Launching Job 3 out of 3
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1516485910189_0030, Tracking URL = http://localhost:8088/proxy/application_1516485910189_0030/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job -kill job_1516485910189_0030
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 1
2018-01-22 10:38:30,610 Stage-3 map = 0%,  reduce = 0%
2018-01-22 10:38:41,945 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 1.8 sec
2018-01-22 10:38:56,880 Stage-3 map = 100%,  reduce = 100%, Cumulative CPU 4.73 sec
MapReduce Total cumulative CPU time: 4 seconds 730 msec
Ended Job = job_1516485910189_0030
Loading data to table project.top_10_unsubscribed_users partition (batchid=7)
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1   Cumulative CPU: 9.36 sec   HDFS Read: 14798 HDFS Write: 96 SUCCESS
Stage-Stage-2: Map: 1 Reduce: 1   Cumulative CPU: 4.34 sec   HDFS Read: 5072 HDFS Write: 96 SUCCESS
Stage-Stage-3: Map: 1 Reduce: 1   Cumulative CPU: 4.73 sec   HDFS Read: 6582 HDFS Write: 69 SUCCESS
Total MapReduce CPU Time Spent: 18 seconds 430 msec
OK
Time taken: 177.031 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,



ACADGILD

Output from, connected_artists, top_10_royalty_songs, top_10_stations.

```
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)
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    99000000        1
S204    2604333         1
S202    1000000         1
S205    0          1
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)
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
```

Output from top_10_unsubscribed_users, usersBehaviour.

```
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)
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
U109    0          1
Time taken: 0.275 seconds, Fetched: 10 row(s)
hive> Select * From usersBehaviour;
OK
usersBehaviour.user_type      usersBehaviour.duration      usersBehaviour.batchid
SUBSCRIBED      93861594        1
UNSUBSCRIBED    105594881       1
Time taken: 0.274 seconds, Fetched: 2 row(s)
hive>
>
>
```

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.

```
1 #!/bin/bash
2
3 #This script is not working.
4 #Either change table to text or use STRING as type of partitioned column
5
6 batchid=`cat /home/acadgild/project/logs/current-batch.txt`
7 LOGFILE=/home/acadgild/project/logs/log_batch_$batchid
8
9 echo "Creating mysql tables if not present..." >> $LOGFILE
10
11 mysql < /home/acadgild/project/scripts/create_schema.sql
12
13 echo "Running sqoop job for data export..." >> $LOGFILE
14
15 sqoop export --connect jdbc:mysql://localhost/project --username root --password acadgild --table top_10_stations --export-dir
16 hdfs://localhost:9000/user/hive/warehouse/project.db/top_10_stations/batchid=$batchid --input-fields-terminated-by ',' -m 1
17
18 sqoop export --connect jdbc:mysql://localhost/project --username root --password acadgild --table users_behaviour --export-dir
19 hdfs://localhost:9000/user/hive/warehouse/project.db/users_behaviour/batchid=$batchid --input-fields-terminated-by ',' -m 1
20
21 sqoop export --connect jdbc:mysql://localhost/project --username root --password acadgild --table connected_artists --export-dir
22 hdfs://localhost:9000/user/hive/warehouse/project.db/connected_artists/batchid=$batchid --input-fields-terminated-by ',' -m 1
23
24 sqoop export --connect jdbc:mysql://localhost/project --username root --password acadgild --table top_10_royalty_songs --export-dir
25 hdfs://localhost:9000/user/hive/warehouse/project.db/top_10_royalty_songs/batchid=$batchid --input-fields-terminated-by ',' -m 1
26
27 sqoop export --connect jdbc:mysql://localhost/project --username root --password acadgild --table top_10_unsubscribed_users --export-dir
28 hdfs://localhost:9000/user/hive/warehouse/project.db/top_10_unsubscribed_users/batchid=$batchid --input-fields-terminated-by ',' -m 1
```

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

```
1 CREATE DATABASE IF NOT EXISTS project;
2
3 USE project;
4
5 CREATE TABLE IF NOT EXISTS top_10_stations
6 (
7     station_id VARCHAR(50),
8     total_distinct_songs_played INT,
9     distinct_user_count INT
10 );
11
12 CREATE TABLE IF NOT EXISTS users_behaviour
13 (
14     user_type VARCHAR(50),
15     duration BIGINT
16 );
17
18 CREATE TABLE IF NOT EXISTS connected_artists
19 (
20     artist_id VARCHAR(50),
21     user_count INT
22 );
23
24 CREATE TABLE IF NOT EXISTS top_10_royalty_songs
25 (
26     song_id VARCHAR(50),
27     duration BIGINT
28 );
29
30 CREATE TABLE IF NOT EXISTS top_10_unsubscribed_users
31 (
32     user_id VARCHAR(50),
33     duration BIGINT
34 );
35
36 commit;
```



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

```
stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
18/01/24 09:57:24 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 1 items
drwxrwxr-x    - acadgild supergroup          0 2018-01-24 09:34 hdfs://localhost:9000/user/hive/warehouse/project.db/top_10_stations/batchid=1
[acadgild@localhost ~]$ sqoop export --connect jdbc:mysql://localhost/project --username root --password acadgild --table top_10_stations --exp
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! Zookeeper imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
2018-01-24 09:58:23,657 INFO  [main] sqoop.Sqoop: Running Sqoop version: 1.4.6
2018-01-24 09:58:23,694 WARN  [main] tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
2018-01-24 09:58:24,084 INFO  [main] manager.MySQLManager: Preparing to use a MySQL streaming resultset.
2018-01-24 09:58:24,085 INFO  [main] tool.CodeGenTool: Beginning code generation
2018-01-24 09:58:24,696 INFO  [main] manager.SqlManager: Executing SQL statement: SELECT t.* FROM `top_10_stations` AS t LIMIT 1
2018-01-24 09:58:24,767 INFO  [main] manager.SqlManager: Executing SQL statement: SELECT t.* FROM `top_10_stations` AS t LIMIT 1
2018-01-24 09:58:24,810 INFO  [main] orm.CompilationManager: HADOOP_MAPRED_HOME is /home/acadgild/hadoop-2.7.2
Note: /tmp/sqoop-acadgild/compile/flae2653abc7121116a39d1b97e8735b/top_10_stations.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
2018-01-24 09:58:29,735 INFO  [main] orm.CompilationManager: Writing jar file: /tmp/sqoop-acadgild/compile/flae2653abc7121116a39d1b97e8735b/top_10_stations.jar
2018-01-24 09:58:29,763 INFO  [main] mapreduce.ExportJobBase: Beginning export of top_10_stations
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/hbase-1.0.3/lib/slf4j-log4j12-1.7.7.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/hadoop-2.7.2/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
Java HotSpot(TM) Client VM warning: You have loaded library /home/acadgild/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0 which might have disabled
stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
2018-01-24 09:58:30,180 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classe
s where applicable
2018-01-24 09:58:30,199 INFO  [main] Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job.jar
2018-01-24 09:58:32,079 INFO  [main] Configuration.deprecation: mapred.reduce.tasks.speculative.execution is deprecated. Instead, use mapreduce
```

The sqoop export command exported the tables from the hive and it stored in the Mysql. The below screen shot show the successful Sqoop export from hive to mysql. The data stored in the Mysql is shown in the successive screen shots,

```
2018-01-24 10:06:14,238 INFO  [main] input.FileInputFormat: Total input paths to process : 1
2018-01-24 10:06:14,435 INFO  [main] mapreduce.JobSubmitter: number of splits:1
2018-01-24 10:06:14,464 INFO  [main] Configuration.deprecation: mapred.map.tasks.speculative.execution is deprecated. Instead, use mapreduce.ma
p.speculative
2018-01-24 10:06:14,791 INFO  [main] mapreduce.JobSubmitter: Submitting tokens for job: job_1516764714140_0018
2018-01-24 10:06:16,293 INFO  [main] impl.YarnClientImpl: Submitted application application_1516764714140_0018
2018-01-24 10:06:16,420 INFO  [main] mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1516764714140_0018/
2018-01-24 10:06:16,424 INFO  [main] mapreduce.Job: Running job: job_1516764714140_0018
2018-01-24 10:06:45,186 INFO  [main] mapreduce.Job: Job job_1516764714140_0018 running in uber mode : false
2018-01-24 10:06:45,191 INFO  [main] mapreduce.Job: map 0% reduce 0%
2018-01-24 10:07:01,634 INFO  [main] mapreduce.Job: map 100% reduce 0%
2018-01-24 10:07:02,707 INFO  [main] mapreduce.Job: Job job_1516764714140_0018 completed successfully
2018-01-24 10:07:03,246 INFO  [main] mapreduce.Job: Counters: 30
File System Counters
  FILE: Number of bytes read=0
  FILE: Number of bytes written=136426
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=311
  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)=12452
  Total time spent by all reduces in occupied slots (ms)=0
  Total time spent by all map tasks (ms)=12452
  Total vcore-seconds taken by all map tasks=12452
  Total megabyte-seconds taken by all map tasks=12750848
Map-Reduce Framework
  Map input records=10
  Map output records=10
  Input split bytes=178
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0
```



Project 2 Music Data Analysis ACADGILD 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)
```



ACADGILD

Output from **top_10_unsubscribed_users** and **usersBehaviour**

```
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 usersBehaviour;
+-----+-----+
| 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.

```
21   ----- *
22   --driver-class-path /home/acadgild/apache-hive-2.1.0-bin/lib/hive-hbase-handler-
23   /home/acadgild/project/lib/sparkanalysis.jar $batchid
24
25   echo "Exporting data to MYSQL using sqoop export..." >> $LOGFILE
26   sh /home/acadgild/project/scripts/data_export.sh
27
28   echo "Incrementing batchid..." >> $LOGFILE
29   batchid=`expr $batchid + 1`
30   echo -n $batchid > /home/acadgild/project/logs/current-batch.txt
31
```

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**

4
0



```
[acadgild@localhost project]$ cd logs  
[acadgild@localhost logs]$ ls -l  
total 24  
-rwxrwxr-x. 1 acadgild acadgild 1 Jan 24 09:44 current-batch.txt  
-rw-rw-r--. 1 acadgild acadgild 679 Jan 24 09:03 derby.log  
drwxrwxr-x. 3 acadgild acadgild 4096 Jan 24 09:02 hdfs:  
-rw-rw-r--. 1 acadgild acadgild 523 Jan 24 09:44 log_batch_1  
-rw-rw-r--. 1 acadgild acadgild 77 Jan 24 09:44 log_batch_1???  
drwxrwxr-x. 5 acadgild acadgild 4096 Jan 24 09:03 metastore_db  
[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

```
1 #!/bin/bash  
2 #All the below scripts will work based on the data provided by acadgild as data/web/file.xml and data/mob/file.txt  
3  
4 python /home/acadgild/project/scripts/generate_web_data.py  
5  
6 python /home/acadgild/project/scripts/generate_mob_data.py  
7  
8 sh /home/acadgild/project/scripts/start-daemons.sh  
9  
10 sh /home/acadgild/project/scripts/populate-lookup.sh  
11  
12 sh /home/acadgild/project/scripts/dataformatting.sh  
13  
14 sh /home/acadgild/project/scripts/data_enrichment.sh  
15  
16 sh /home/acadgild/project/scripts/data_analysis.sh  
17
```

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.



```
[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
```

```
-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]$
```

*Installing the **crontab** in the vm,*



Project 2 Music Data Analysis ACADGILD The **crontab** job scheduler will run the **wrapper.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
* */* * * date>>/home/acadgild/project/scripts/wrapper.sh >> /home/acadgild/project/scripts/jobsheduling.log
[acadgild@localhost ~]$
[acadgild@localhost ~]$
[acadgild@localhost ~]$
```

Problems faced during project installation and how it resolved

1. The hive tables cannot be created when we run the **data_enrichment_filtering_schema.sh**

```
Error: Exception in thread "main" java.lang.RuntimeException: Hive metastore database is not initialized. Please use schematool (e.g. ./schematool -initSchema -dbType ...) to create the schema. If needed, don't forget to include the option to auto-create the underlying database in your JDBC connection string (e.g. ?createDatabaseIfNotExist=true for mysql)
```

Solution:

The metastore already existed, but not in complete form.

Before you run hive for the first time, run the below commands where the **metastore_db** is located,

```
+ schematool -initSchema -dbType derby
+ mv metastore_db metastore_db.tmp
+ schematool -initSchema -dbType derby
```

2. The hive cannot be created and receiving the below errors,

```
{useSQL11ReservedKeywordsForIdentifier()}?} at
org.apache.hadoop.hive.ql.parse.HiveParser_IdentifiersParser.identifier(HiveParser_IdentifiersParser.java:11644) at
org.apache.hadoop.hive.ql.parse.HiveParser.tabTypeExpr(HiveParser.java:15574)
```

Solution

Setting **set hive.support.sql11.reserved.keywords=false;** to false resolved the issue.

3. The spark submit cannot connect to the Hive,

Solution

Move **hive-site.xml** from **\$HIVE_HOME/conf/hive-site.xml** to **\$SPARK_HOME/conf/hive-site.xml**. Make an entry regarding hive metastore uris in this file. The entry will look like this:

4
3



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

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

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

```
-rwxrwxr-x. 1 acadgild acadgild 3764 Nov 25 05:01 spark-env.sh.template
[acadgild@localhost conf]$ ln -s /home/acadgild/project/scripts/hive-site.xml
[acadgild@localhost conf]$ ls -l
total 32
-rw-rw-r--. 1 acadgild acadgild  996 Nov 25 05:01 docker.properties.template
-rw-rw-r--. 1 acadgild acadgild 1105 Nov 25 05:01 fairscheduler.xml.template
lrwxrwxrwx. 1 acadgild acadgild    44 Jan 25 00:44 hive-site.xml -> /home/acadgild/project/scripts/hive-site.xml
-rw-rw-r--. 1 acadgild acadgild 2025 Nov 25 05:01 log4j.properties.template
-rw-rw-r--. 1 acadgild acadgild 7313 Nov 25 05:01 metrics.properties.template
-rw-rw-r--. 1 acadgild acadgild   865 Nov 25 05:01 slaves.template
-rw-rw-r--. 1 acadgild acadgild 1292 Nov 25 05:01 spark-defaults.conf.template
-rwxrwxr-x. 1 acadgild acadgild 3764 Nov 25 05:01 spark-env.sh.template
[acadgild@localhost conf]$ cat hive-site.xml
<configuration>
```

Source: <https://acadgild.com/blog/how-to-access-hive-tables-to-spark-sql/>

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.

Project End Conclusion:

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