

## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

#### 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

## Big Data- Hadoop\_Final Project

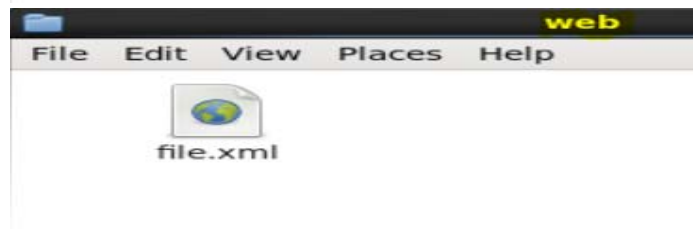
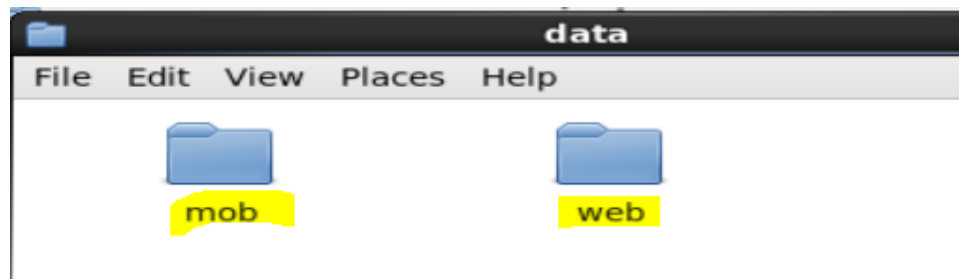
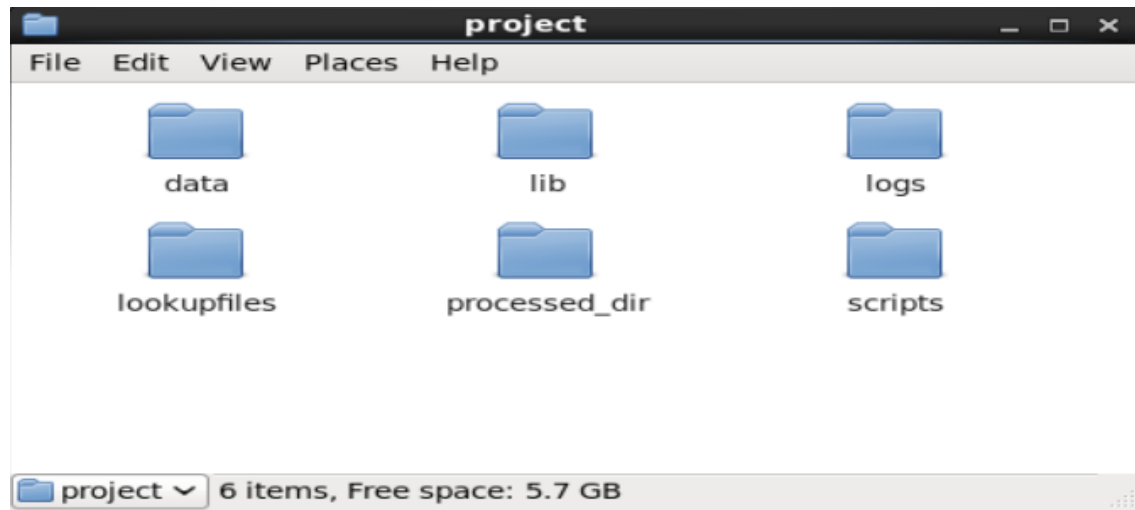
### Music Data Analysis using Hadoop

#### 1.3 DATASET

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

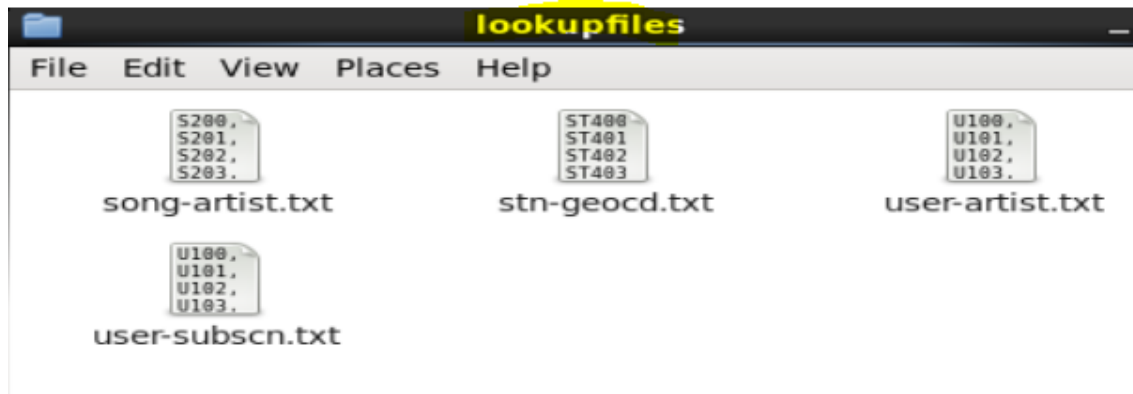
Below is the link for same.

[https://drive.google.com/drive/folders/0B\\_P3pWagdIrrMjJGVlNsSUEtbG8?usp=sharing](https://drive.google.com/drive/folders/0B_P3pWagdIrrMjJGVlNsSUEtbG8?usp=sharing)



## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop



#### 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)

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.

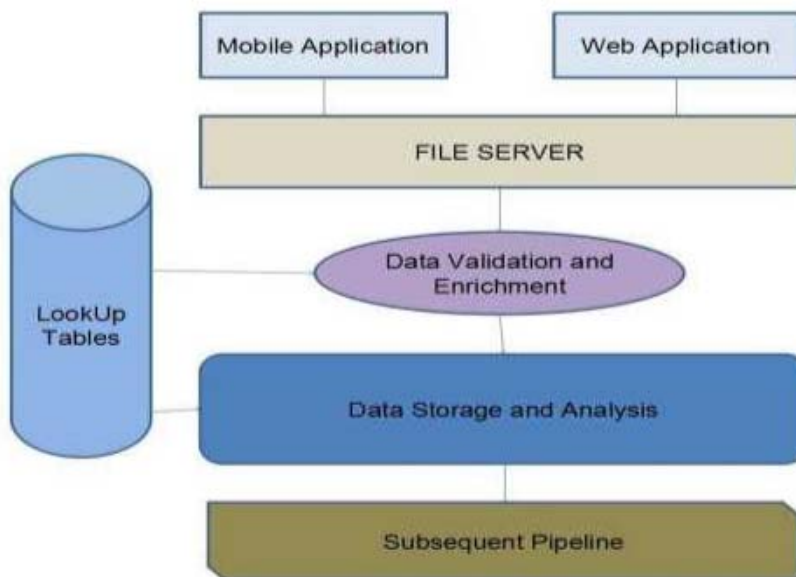
## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

2. Try to make joins as less expensive as possible.
3. Data Cleaning, Validation, Enrichment, Analysis and Post Analysis have to be automated. Try using schedulers.
4. Appropriate logs have to maintain to track the behaviour and overcome failures in the pipeline.

#### 1.7 Flow of operations

A schematic flow of operations is shown below



**Section -2 –**

**Design of the Project 2.1 Low Level Design**

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

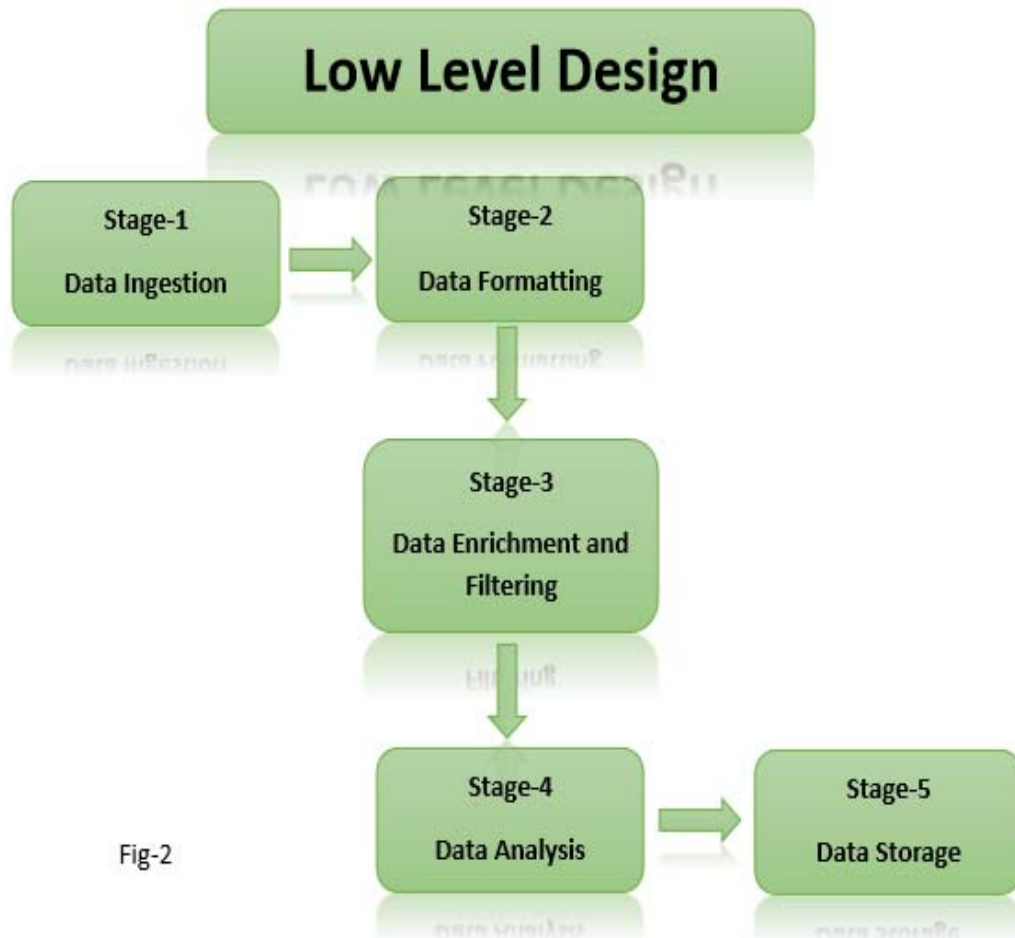


Fig-2

2.2 High Level Design

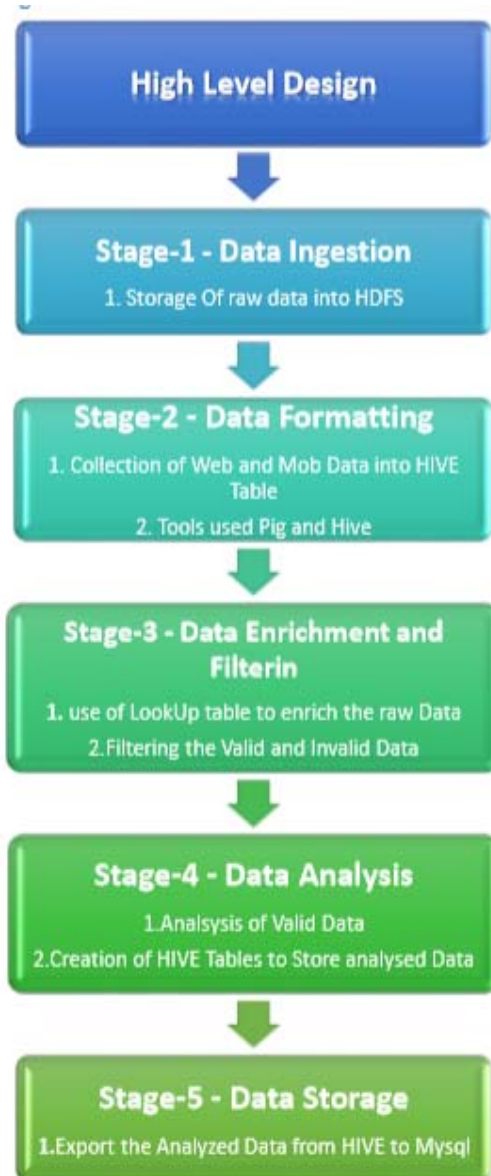


Fig-3

## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

#### Section-3-Hadoop Eco-System Implementation

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

Batch file script,

```
start-daemons.sh X
#!/bin/bash

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

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

echo "Starting daemons" >> $LOGFILE

# To Start 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 ~]$ sh /home/acadgild/project/scripts/start-daemons.sh
Batch File Found!
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
18/09/08 23:48:57 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/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-namenode-localhost.localdomain.out
localhost: starting datanode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-datanode-localhost.localdomain.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-secondarynamenode-localhost.localdomain.out
18/09/08 23:49:22 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/install/hadoop/hadoop-2.6.5/logs/yarn-acadgild-resourcemanager-localhost.localdomain.out
localhost: starting nodemanager, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/yarn-acadgild-nodemanager-localhost.localdomain.out
localhost: starting zookeeper, logging to /home/acadgild/install/hbase/hbase-1.2.6/logs/hbase-acadgild-zookeeper-localhost.localdomain.out
starting master, logging to /home/acadgild/install/hbase/hbase-1.2.6/logs/hbase-
```



## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

```
starting historyserver, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/lo
gs/mapred-acadgild-historyserver-localhost.localdomain.out
Starting mysqld: [ OK ]
2018-09-08 23:50:06: Starting Hive Metastore Server
/home/acadgild/install/hive/apache-hive-2.3.2-bin/bin/ext/metastore.sh: line 29:
export: ` -Dproc_metastore -Dlog4j.configurationFile=hive-log4j2.properties -
Djava.util.logging.config.file=/home/acadgild/install/hive/apache-hive-2.3.2-bin
/conf/parquet-logging.properties `: not a valid identifier
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-
bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/sha
re/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.
class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
2018-09-08T23:50:20,587 INFO [main] org.apache.hadoop.hive.conf.HiveConf - Found
configuration file file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/conf/
hive-site.xml
2018-09-08T23:50:27,057 INFO [main] org.apache.hadoop.hive.metastore.HiveMetaSto
re - STARTUP_MSG:
/*****
STARTUP_MSG: Starting HiveMetaStore
STARTUP_MSG: host = localhost/127.0.0.1
```

2. 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,

```
[acadgild@localhost ~]$ jps
3649 NodeManager
4561 RunJar
4267 HMaster
4171 HQuorumPeer
3211 DataNode
3403 SecondaryNameNode
3547 ResourceManager
5148 Jps
3087 NameNode
4495 JobHistoryServer
4383 HRegionServer
```



hdfs\_



metastore\_db



metastore\_db.tmp

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**.



## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop



#### 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 <b>geo_cd</b> with <b>station_id</b>	stn-geocd.txt
2	subscribed-users	Contains <b>user_id</b> , <b>subscription_start_date</b> and <b>subscription_end_date</b> . Contains details only for subscribed users	user-subscn.txt
3	song-artist-map	Contains mapping of <b>song_id</b> with <b>artist_id</b> Along with royalty associated with each play of the song	song-artist.txt
4	user-artist-map	Contains an array of <b>artist_id(s)</b> followed by a <b>user_id</b>	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.

## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

```
populate-lookup.sh X
#!/bin/bash

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

echo "Creating LookUp Tables" >> $LOGFILE

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

echo "Populating LookUp Tables" >> $LOGFILE

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

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

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

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

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

Run the script: ./populate-lookup.sh

```
[acadgild@localhost ~]$ sh /home/acadgild/project/scripts/populate-lookup.sh
2018-09-08 23:54:30,125 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

create 'station-geo-map', 'geo'
0 row(s) in 3.1530 seconds

Hbase::Table - station-geo-map
2018-09-08 23:54:45,239 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
SLF4J: Class path contains multiple SLF4J bindings.
```

## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

```
create 'subscribed-users', 'subscn'  
0 row(s) in 1.7980 seconds
```

```
Hbase::Table - subscribed-users  
2018-09-08 23:54:58,773 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes instead
```

```
create 'song-artist-map', 'artist'  
0 row(s) in 1.8870 seconds
```

```
Hbase::Table - song-artist-map  
2018-09-08 23:55:13,171 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes instead
```

```
put 'station-geo-map', 'ST400', 'geo:geo_cd', 'A'  
0 row(s) in 0.9730 seconds
```

```
put 'song-artist-map', 'S202', 'artist:artistid', 'A302'  
0 row(s) in 0.6970 seconds
```

```
put 'subscribed-users', 'U100', 'subscn:startdt', '1465230523'  
0 row(s) in 0.5150 seconds
```

```
Logging initialized using configuration in jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hive-common-2.3.2.jar!/hive-log4j2.properties Async: true  
OK  
Time taken: 21.748 seconds  
OK  
Time taken: 0.071 seconds  
OK  
Time taken: 5.026 seconds  
Loading data to table project.users_artists  
OK  
Time taken: 8.278 seconds
```

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

Lookup Tables in the hbase shell,

```
hbase(main):001:0> list  
TABLE  
bulktable  
clicks  
clicks1  
plants  
song-artist-map  
station-geo-map  
subscribed-users  
7 row(s) in 1.0390 seconds  
=> ["bulktable", "clicks", "clicks1", "plants", "song-artist-map", "station-geo-map", "subscribed-users"]
```

## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

The values loaded in the Lookup tables are shown below,

#### song-artist-map

```
hbase(main):002:0> scan 'song-artist-map'
ROW          COLUMN+CELL
S200         column=artist:artistid, timestamp=1536431307664, value=A30
0
S201         column=artist:artistid, timestamp=1536431320389, value=A30
1
S202         column=artist:artistid, timestamp=1536431333004, value=A30
2
S203         column=artist:artistid, timestamp=1536431345899, value=A30
3
S204         column=artist:artistid, timestamp=1536431358653, value=A30
4
S205         column=artist:artistid, timestamp=1536431371190, value=A30
1
S206         column=artist:artistid, timestamp=1536431384035, value=A30
2
S207         column=artist:artistid, timestamp=1536431396771, value=A30
3
S208         column=artist:artistid, timestamp=1536431409984, value=A30
4
S209         column=artist:artistid, timestamp=1536431422326, value=A30
5
10 row(s) in 0.4220 seconds
```

#### station-geo-map

```
hbase(main):003:0> scan 'station-geo-map'
ROW          COLUMN+CELL
ST400        column=geo:geo_cd, timestamp=1536431116927, value=A
ST401        column=geo:geo_cd, timestamp=1536431129347, value=AU
ST402        column=geo:geo_cd, timestamp=1536431141865, value=AP
ST403        column=geo:geo_cd, timestamp=1536431154611, value=J
ST404        column=geo:geo_cd, timestamp=1536431168157, value=E
ST405        column=geo:geo_cd, timestamp=1536431180666, value=A
ST406        column=geo:geo_cd, timestamp=1536431192822, value=AU
ST407        column=geo:geo_cd, timestamp=1536431206290, value=AP
ST408        column=geo:geo_cd, timestamp=1536431218499, value=E
ST409        column=geo:geo_cd, timestamp=1536431231455, value=E
ST410        column=geo:geo_cd, timestamp=1536431243954, value=A
ST411        column=geo:geo_cd, timestamp=1536431256698, value=A
ST412        column=geo:geo_cd, timestamp=1536431268827, value=AP
ST413        column=geo:geo_cd, timestamp=1536431281830, value=J
ST414        column=geo:geo_cd, timestamp=1536431294734, value=E
15 row(s) in 0.1340 seconds
```

## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

#### subscribed-users

```
hbase(main):004:0> scan 'subscribed-users'
ROW COLUMN+CELL
U100 column=subscn:enddt, timestamp=1536431446966, value=1465130523
U100 column=subscn:startdt, timestamp=1536431434484, value=1465230523
U101 column=subscn:enddt, timestamp=1536431473347, value=1475130523
U101 column=subscn:startdt, timestamp=1536431459505, value=1465230523
U102 column=subscn:enddt, timestamp=1536431498428, value=1475130523
U102 column=subscn:startdt, timestamp=1536431486043, value=1465230523
U103 column=subscn:enddt, timestamp=1536431523614, value=1475130523
U103 column=subscn:startdt, timestamp=1536431510669, value=1465230523
U104 column=subscn:enddt, timestamp=1536431549489, value=1475130523
U104 column=subscn:startdt, timestamp=1536431536755, value=1465230523
U105 column=subscn:enddt, timestamp=1536431575466, value=1475130523
U105 column=subscn:startdt, timestamp=1536431562508, value=1465230523
U106 column=subscn:enddt, timestamp=1536431602204, value=1485130523
U106 column=subscn:startdt, timestamp=1536431588815, value=1465230523
U107 column=subscn:enddt, timestamp=1536431628433, value=1455130523
U107 column=subscn:startdt, timestamp=1536431615180, value=1465230523
U108 column=subscn:enddt, timestamp=1536431657411, value=1465230623
U108 column=subscn:startdt, timestamp=1536431643225, value=1465230523
U109 column=subscn:enddt, timestamp=1536431684886, value=1475130523
U109 column=subscn:startdt, timestamp=1536431671469, value=1465230523
U110 column=subscn:enddt, timestamp=1536431713667, value=1475130523
```

## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

```
U110      column=subscn:enddt, timestamp=1536431713667, value=147
          5130523
U110      column=subscn:startdt, timestamp=1536431698933, value=1
          465230523
U111      column=subscn:enddt, timestamp=1536431741647, value=147
          5130523
U111      column=subscn:startdt, timestamp=1536431727804, value=1
          465230523
U112      column=subscn:enddt, timestamp=1536431774030, value=147
          5130523
U112      column=subscn:startdt, timestamp=1536431756332, value=1
          465230523
U113      column=subscn:enddt, timestamp=1536431805069, value=148
          5130523
U113      column=subscn:startdt, timestamp=1536431789462, value=1
          465230523
U114      column=subscn:enddt, timestamp=1536431834210, value=146
          8130523
U114      column=subscn:startdt, timestamp=1536431819885, value=1
          465230523
15 row(s) in 0.2580 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.

```
Logging initialized using configuration in jar:file:/home/acadgild/install/hive/
apache-hive-2.3.2-bin/lib/hive-common-2.3.2.jar!/hive-log4j2.properties Async: t
rue
OK
Time taken: 21.748 seconds
OK
Time taken: 0.071 seconds
OK
Time taken: 5.026 seconds
Loading data to table project.users_artists
OK
Time taken: 8.278 seconds
You have new mail in /var/spool/mail/acadgild
acadgild@localhost ~$ hbase
```

```
sing Hive 1.X releases.
hive> show databases;
OK
custom
default
project
Time taken: 40.762 seconds, Fetched: 3 row(s)
hive>
hive> use project;
OK
Time taken: 0.145 seconds
hive> show tables;
```

## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

hive> Select \* From users\_artists;

```
hive> show tables;
OK
users_artists
Time taken: 0.161 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: 9.456 seconds, Fetched: 15 row(s)
```

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,**

```
#!/bin/bash

batchid=`cat /home/acadgild/project/logs/current_batch.txt`
LOGFILE=/home/acadgild/project/logs/log_batch_${batchid}

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

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

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.

**create\_hive\_hbase\_lookup.hql**



## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

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

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

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

The below screenshot we can see tables getting created in hive by running the "data\_enrichement\_filtering\_schema.sh file"

## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

```
[acadgild@localhost ~]$ sh /home/acadgild/project/scripts/data_enrichment_filtering_schema.sh
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hive-common-2.3.2.jar!/hive-log4j2.properties Async: true
OK
Time taken: 32.075 seconds
OK
Time taken: 16.2 seconds
OK
Time taken: 0.529 seconds
OK
Time taken: 0.474 seconds
You have new mail in /var/spool/mail/acadgild
```

Hive>Show Tables;

```
hive> show tables;
OK
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 0.871 seconds, Fetched: 4 row(s)
```

hive>Select \* From song\_artist\_map

```
hive> select * from song_artist_map;
OK
S200    A300
S201    A301
S202    A302
S203    A303
S204    A304
S205    A301
S206    A302
S207    A303
S208    A304
S209    A305
Time taken: 15.159 seconds, Fetched: 10 row(s)
```

## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

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: 0.898 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: 1.079 seconds, Fetched: 15 row(s)
```

**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`

## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

```
#!/bin/bash

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

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

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

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

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

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

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

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

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

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

1. **Dataformatting.pig**
2. **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

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

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

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

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

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

#### formatted\_hive\_load.hql

## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

```
set hive.support.sql11.reserved.keywords=false;
USE project;

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

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

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

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

```
[acadgild@localhost ~]$ sh /home/acadgild/project/scripts/dataformatting.sh
18/09/09 02:15:09 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
rm: `/user/acadgild/project/batch2/web/': No such file or directory
18/09/09 02:15:16 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
rm: `/user/acadgild/project/batch2/formattedweb/': No such file or director
y
18/09/09 02:15:20 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
rm: `/user/acadgild/project/batch2/mob/': No such file or directory
18/09/09 02:15:24 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
18/09/09 02:15:28 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
18/09/09 02:15:34 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
18/09/09 02:16:21 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
18/09/09 02:16:31 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
18/09/09 02:16:31 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
```

## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

```
HadoopVersion  PigVersion  UserId  StartedAt      FinishedAt      Fea
tures
2.6.5    0.16.0  acadgild  2018-09-09 02:16:46  2018-09-09 02:20:37
UNKNOWN

Success!

Job Stats (time in seconds):
JobId  Maps  Reduces MaxMapTime  MinMapTime  AvgMapTime  MedianMapTime  MaxReduceTime  MinReduceTime  AvgReduceTime  MedianReduceTime  ElapsedTime  Alias  Feature  Outputs
job_1536430769011_0001  1  0  68  68  68  68  0  0  /user/acadgild/project/batch2/formattedweb,
ttedweb,

Input(s):
Successfully read 20 records (7105 bytes) from: "/user/acadgild/project/batch2/web"

Output(s):
Successfully stored 20 records (1235 bytes) in: "/user/acadgild/project/batch2/formattedweb"
```

In the above screenshot we can see the **dataformatting.pig** along with the **formatted\_hive\_load.hql** executed successfully.

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

```
[acadgild@localhost ~]$ hadoop fs -ls /user/acadgild/project
18/09/10 22:57:15 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 1 items
drwxr-xr-x - acadgild supergroup 0 2018-09-10 22:46 /user/acadgild/project/batch2
You have new mail in /var/spool/mail/acadgild
```

```
[acadgild@localhost ~]$ hadoop fs -ls /user/acadgild/project/batch2
18/09/10 22:58:46 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 3 items
drwxr-xr-x - acadgild supergroup 0 2018-09-10 22:46 /user/acadgild/project/batch2/formattedweb
drwxr-xr-x - acadgild supergroup 0 2018-09-10 22:45 /user/acadgild/project/batch2/mob
drwxr-xr-x - acadgild supergroup 0 2018-09-10 22:45 /user/acadgild/project/batch2/web
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ hadoop fs -ls /user/acadgild/project/batch2/formattedweb
18/09/10 22:59:11 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r-- 1 acadgild supergroup 0 2018-09-10 22:46 /user/acadgild/project/batch2/formattedweb/_SUCCESS
-rw-r--r-- 1 acadgild supergroup 1235 2018-09-10 22:46 /user/acadgild/project/batch2/formattedweb/part-m-00000
```



## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

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 ~]$ hadoop fs -cat /user/acadgild/project/batch2/formattedweb/*
18/09/10 23:01:49 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
U110,S206,A302,1462863262,1462863262,1494297562,E,ST410,0,1,1
U106,S207,A301,1494297562,1494297562,1465490556,AP,ST409,3,0,1
U100,S210,A303,1462863262,1468094889,1465490556,AP,ST405,3,1,0
U118,S203,A300,1465490556,1462863262,1468094889,E,ST411,0,1,1
U119,S205,A305,1462863262,1494297562,1462863262,E,ST403,3,1,0
,S209,A303,1462863262,1494297562,1462863262,A,ST401,2,0,1
U107,S204,A302,1462863262,1465490556,1494297562,AP,ST404,3,1,0
U104,S207,A305,1462863262,1465490556,1465490556,AU,ST407,0,0,0
U114,S209,A304,1462863262,1465490556,1462863262,,ST401,0,1,0
U100,S201,,1465490556,1462863262,1462863262,E,ST413,3,0,1
U101,S202,A300,1462863262,1462863262,1494297562,A,ST411,0,1,0
U116,S207,A305,1468094889,1468094889,1494297562,A,ST411,2,0,1
U111,S205,A302,1465490556,1494297562,1468094889,U,ST402,2,0,0
U119,S210,A303,1494297562,1494297562,1468094889,U,ST401,0,1,0
U110,S206,A305,1462863262,1465490556,1462863262,E,ST404,0,0,0
U119,S205,A305,1468094889,1462863262,1465490556,A,ST403,1,0,0
U119,S209,A303,1494297562,1468094889,1462863262,U,ST404,2,1,1
U103,S208,A303,1462863262,1465490556,1468094889,A,ST403,3,1,0
U116,S208,A305,1465490556,1494297562,1465490556,E,ST406,3,0,1
U111,S200,A303,1462863262,1494297562,1465490556,E,ST402,1,1,0
```

The new Tables has been created and show below,

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

DataFormatting.sh output in hive terminal, **hive> select \* from formatted\_input;**



## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

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

- In the above screenshot we can see the formatted input data with some null values in **user\_id**, **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**.

#### 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,

- If any of like or dislike is **NULL** or **absent**, consider it as **0**.
- If fields like **Geo\_cd** and **Artist\_id** are **NULL** or **absent**, consult the lookup tables for fields **Station\_id** and **Song\_id** respectively to get the values of **Geo\_cd** and **Artist\_id**.
- If corresponding lookup entry is not found, consider that **record** to be **invalid**

## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

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

#### data\_enrichment.sh

```
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 {} \;
```

#### data\_enrichment.hql

```
set hive.support.sql11.reserved.keywords=false;
SET hive.auto.convert.join=false;
SET hive.exec.dynamic.partition.mode=nonstrict;

USE project;

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

## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

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

```
[acadgild@localhost scripts]$ sh /home/acadgild/project/scripts/data_enrichment.sh
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hive-common-2.3.2-
```

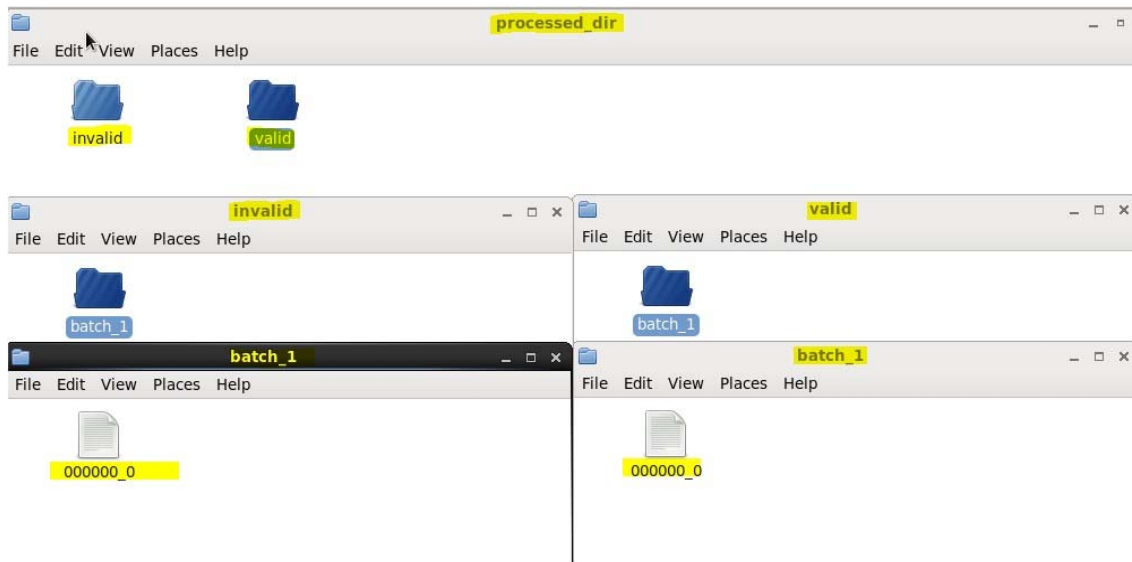
```
No Stats for project@formatted_input, Columns: start_ts, song_id, user_id, end_ts, dislike, station_id, timestamp_t, geo_cd,
No Stats for project@station_geo_map, Columns: station_id, geo_cd
No Stats for project@song_artist_map, Columns: song_id, artist_id
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
Query ID = acadgild_20180917002421_d252ae6f-956b-451f-b34e-69d255a60c86
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_1537110040827_0003, Tracking URL = http://localhost:8088/proxy/application_1537110040827_0003/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kill job_1537110040827_0003
Hadoop job information for Stage-1: number of mappers: 3; number of reducers: 1
2018-09-17 00:24:52,092 Stage-1 map = 0%, reduce = 0%
2018-09-17 00:25:42,124 Stage-1 map = 67%, reduce = 0%, Cumulative CPU 5.89 sec
2018-09-17 00:25:47,337 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 11.27 sec
2018-09-17 00:26:01,075 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 14.31 sec
MapReduce Total cumulative CPU time: 14 seconds 310 msec
Ended Job = job_1537110040827_0003
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_1537110040827_0004, Tracking URL = http://localhost:8088/proxy/application_1537110040827_0004/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kill job_1537110040827_0004
Hadoop job information for Stage-2: number of mappers: 2; number of reducers: 1
2018-09-17 00:26:27,373 Stage-2 map = 0%, reduce = 0%
2018-09-17 00:26:51,888 Stage-2 map = 50%, reduce = 0%, Cumulative CPU 2.98 sec
2018-09-17 00:26:54,052 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 7.2 sec
2018-09-17 00:27:08,007 Stage-2 map = 100%, reduce = 68%, Cumulative CPU 12.01 sec
2018-09-17 00:27:09,306 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 13.15 sec
MapReduce Total cumulative CPU time: 13 seconds 150 msec
```

## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

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.

```
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ cd project/processed_dir
[acadgild@localhost processed_dir]$ ls -l invalid
total 8
drwxrwxr-x. 2 acadgild acadgild 4096 Sep 16 22:54 batch_1
drwxrwxr-x. 2 acadgild acadgild 4096 Sep 17 00:40 batch_2
[acadgild@localhost processed_dir]$ ls -l valid
total 8
drwxrwxr-x. 2 acadgild acadgild 4096 Sep 16 22:54 batch_1
drwxrwxr-x. 2 acadgild acadgild 4096 Sep 17 00:40 batch_2
[acadgild@localhost processed_dir]$
```



In the below screenshot we have data for **enriched\_data** table where we filled the null values of **artist\_id** and **geo\_cd** of formatted input with the help of lookup tables,

```
hive>select * From enriched_data;
```

## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

```
hive> select * from enriched_data;
OK
U101 S201 A305 1465130523 1475130523 1465130523 AP ST405 3 1 1 2 fail
U118 S204 A305 1475130523 1485130523 1485130523 E ST412 0 1 1 2 fail
U119 S204 A304 1462863262 1468094889 1462863262 A ST402 0 1 1 2 fail
U100 S204 A301 1462863262 1494297562 1494297562 AP ST408 2 1 1 2 fail
U110 S204 A304 1468094889 1494297562 1494297562 E ST406 3 1 1 2 fail
S205 A303 1465490556 1468094889 1468094889 U ST403 3 1 0 2 fail
U119 S206 A301 1468094889 1494297562 1468094889 AP ST400 3 1 1 2 fail
U100 S207 A303 1462863262 1465490556 1494297562 A ST415 3 0 0 2 fail
U108 S208 A301 1468094889 1465490556 1468094889 E ST415 1 0 1 2 fail
U103 S208 A303 1494297562 1494297562 1494297562 AP ST404 1 1 1 2 fail
S209 A301 1475130523 1465130523 1485130523 U ST404 0 0 1 2 fail
U113 S209 A300 1465130523 1485130523 1465130523 U ST415 1 0 0 2 fail
U109 S210 NULL 1465130523 1475130523 1465130523 AP ST410 0 1 0 2 fail
U104 S210 A305 1462863262 1494297562 1494297562 AU ST401 1 1 0 2 fail
U113 S210 A304 1465130523 1485130523 1475130523 AP ST400 0 0 1 2 fail
U109 S200 A300 1475130523 1475130523 1465230523 U ST407 3 1 0 2 pass
U117 S201 A305 1465130523 1475130523 1475130523 E ST411 2 0 1 2 pass
U118 S202 A304 1495130523 1465130523 1465230523 E ST407 2 1 0 2 pass
U100 S202 A304 1475130523 1465230523 1465230523 A ST407 3 1 0 2 pass
U101 S202 A302 1465130523 1465230523 1475130523 J ST413 3 0 1 2 pass
U110 S202 A305 1465130523 1475130523 1485130523 AP ST404 3 0 0 2 pass
U120 S204 A305 1465230523 1485130523 1475130523 AU ST404 2 0 0 2 pass
U106 S204 A305 1494297562 1465490556 1468094889 AP ST407 0 0 0 2 pass
U115 S205 A305 1495130523 1475130523 1465230523 AU ST400 1 0 0 2 pass
U112 S205 A302 1462863262 1465490556 1465490556 U ST406 1 1 0 2 pass
U103 S206 A305 1468094889 1462863262 1465490556 U ST407 3 0 0 2 pass
U118 S206 A304 1465130523 1485130523 1465230523 E ST402 0 0 0 2 pass
U108 S206 A303 1494297562 1462863262 1468094889 AP ST405 3 0 1 2 pass
U110 S207 A300 1495130523 1485130523 1465130523 A ST404 0 0 1 2 pass
U107 S208 A303 1465130523 1475130523 1465130523 AU ST400 1 0 1 2 pass
U117 S208 A303 1495130523 1485130523 1465130523 AU ST411 2 1 0 2 pass
U108 S208 A305 1465130523 1485130523 1485130523 U ST410 3 0 1 2 pass
U109 S208 A303 1494297562 1468094889 1465490556 AP ST406 2 0 0 2 pass
U113 S208 A302 1465490556 1494297562 1494297562 A ST405 1 0 0 2 pass
U100 S208 A300 1462863262 1465490556 1462863262 AP ST405 0 1 0 2 pass
U120 S208 A304 1494297562 1462863262 1462863262 U ST402 1 0 0 2 pass
U120 S209 A304 1494297562 1465490556 1462863262 AU ST402 3 1 0 2 pass
U104 S209 A300 1465490556 1494297562 1465490556 A ST411 2 0 1 2 pass
U115 S209 A305 1465490556 1465490556 1462863262 U ST414 0 1 0 2 pass
U109 S209 A304 1465230523 1475130523 1485130523 E ST402 2 1 0 2 pass
Time taken: 0.35 seconds, Fetched: 40 row(s)
```

#### 4.4 Stage – 4 - Data Analysis

##### DataAnalysis.sh

```
#!/bin/bash

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

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

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

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

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

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



# Big Data- Hadoop\_Final Project

## Music Data Analysis using Hadoop

### DataAnalysis.hql

```
set hive.support.sql11.reserved.keywords=false;
SET hive.auto.convert.join=false;
USE project;

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

INSERT OVERWRITE TABLE top_10_stations
PARTITION(batchid=${hiveconf:batchid})
SELECT
  station_id,
  COUNT(DISTINCT song_id) AS total_distinct_songs_played,
  COUNT(DISTINCT user_id) AS distinct_user_count
FROM enriched_data
WHERE status='pass'
AND batchid=${hiveconf:batchid}
AND like_s=1
GROUP BY station_id
ORDER BY total_distinct_songs_played DESC
LIMIT 10;
```

### 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
```

## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

```
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: 0.366 seconds, Fetched: 11 row(s)
hive> select * from connected_artists;
OK
A303      2      2
A302      2      2
A300      1      2
Time taken: 0.507 seconds, Fetched: 3 row(s)
hive> select * from top_10_stations;
OK
ST407      2      3      2
ST414      1      1      2
ST411      1      1      2
ST402      1      2      2
ST406      1      1      2
ST405      1      1      2
Time taken: 0.397 seconds, Fetched: 6 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: 0.347 seconds, Fetched: 15 row(s)
hive>
```

```
hive> select * from top_10_royalty_songs;
OK
S208      22627294      2
S207      20000000      2
S206      19900000      2
S209      15254588      2
S200      9900000 2
S204      2604333 2
S202      100000 2
S205      0 2
Time taken: 0.342 seconds, Fetched: 8 row(s)
hive> select * from top_10_unsubscribed_users;
OK
U117      20000000      2
U118      20000000      2
U110      20000000      2
U120      12627294      2
U115      12527294      2
U107      10000000      2
U108      5231627 2
U109      2604333 2
U106      2604333 2
U100      0 2
Time taken: 0.402 seconds, Fetched: 10 row(s)
```

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



## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

```
hive> select station_id from top_10_stations;
OK
ST407
ST414
ST411
ST402
ST406
ST405
Time taken: 0.269 seconds, Fetched: 6 row(s)
```

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

```
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

```
OK
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

```
S208
S207
S206
S209
S200
S204
S202
S205
```

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

## Big Data- Hadoop\_Final Project

### Music Data Analysis using Hadoop

```
U117
U118
U110
U120
U115
U107
U108
U109
U106
U100
```

### Job Scheduling:

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

```
[acadgild@localhost project]$ cd logs
[acadgild@localhost logs]$ ls -l
total 36
-rwxrwxr-x. 1 acadgild acadgild  1 Sep 17 01:27 current-batch.txt
-rw-rw-r--. 1 acadgild acadgild 679 Jan 24 2018 derby.log
drwxrwxr-x. 3 acadgild acadgild 4096 Jan 24 2018 hdfs:
-rw-rw-r--. 1 acadgild acadgild  77 Jan 24 2018 log_batch_1
-rw-rw-r--. 1 acadgild acadgild 1265 Sep 17 01:27 log_batch_2
-rw-rw-r--. 1 acadgild acadgild  77 Sep 17 01:27 log_batch_2???
-rw-rw-r--. 1 acadgild acadgild  34 Sep 19 09:34 log_batch_3
-rw-rw-r--. 1 acadgild acadgild 154 Sep 19 12:49 log_batch_3???
drwxrwxr-x. 5 acadgild acadgild 4096 Jan 24 2018 metastore_db
[acadgild@localhost logs]$ cat current-batch.txt
3[acadgild@localhost logs]$
```

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

```
3[acadgild@localhost logs]$ cat log_batch_2
Starting daemons
Creating LookUp Tables
Populating LookUp Tables
Starting daemons
Creating LookUp Tables
Populating LookUp Tables
Starting daemons
Starting daemons
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...
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 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 enrichment and filtering...
Copying valid and invalid records in local file system...
Running hive script for data analysis...
Incrementing batchid...
```

## Big Data- Hadoop\_Final Project

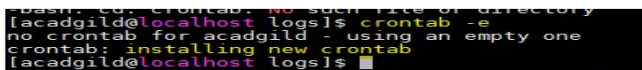
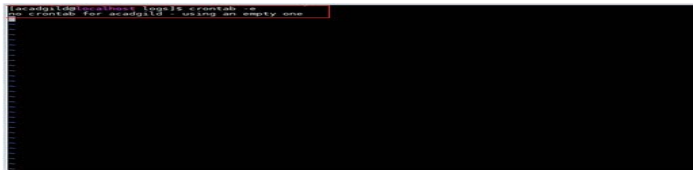
### Music Data Analysis using Hadoop

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.

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



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

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 ~]$ cd
[acadgild@localhost ~]$ crontab -l
#do this for every 3 hours
* */3 * * * date>>/home/acadgild/project/scripts/wrapper.sh >> /home/acadgild/project/scripts/jobsheduling.log
[acadgild@localhost ~]$
[acadgild@localhost ~]$
[acadgild@localhost ~]$
```