Big Data- Hadoop_Final Project

Music Data Analysis using Hadoop

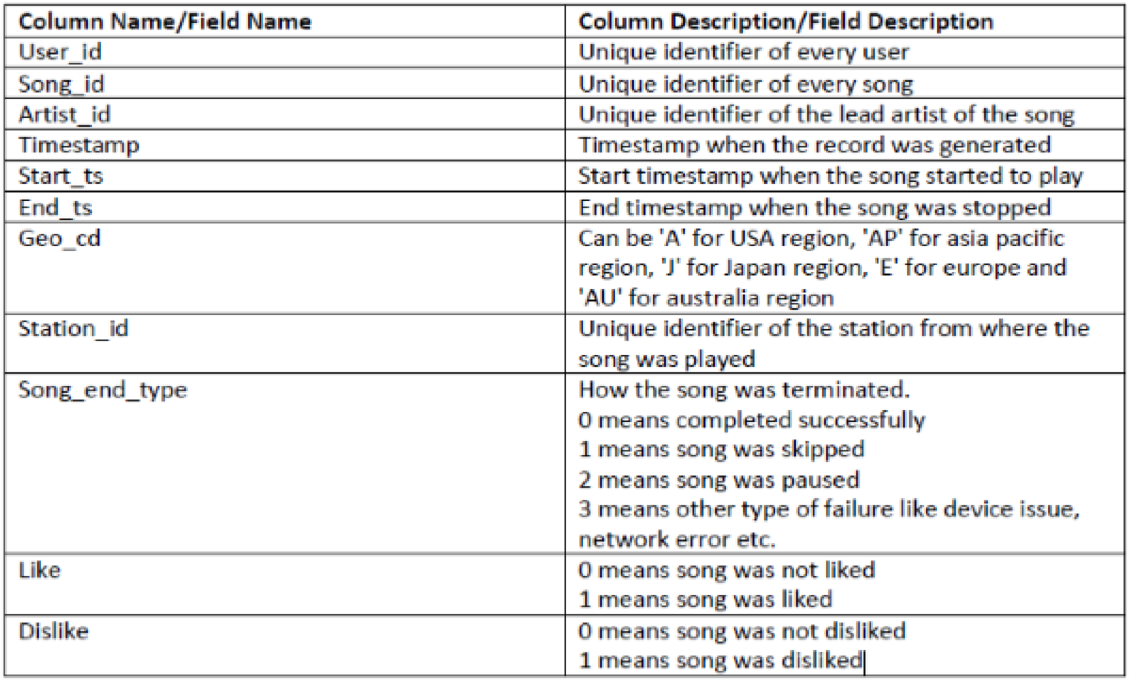
Section – 1 –

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.

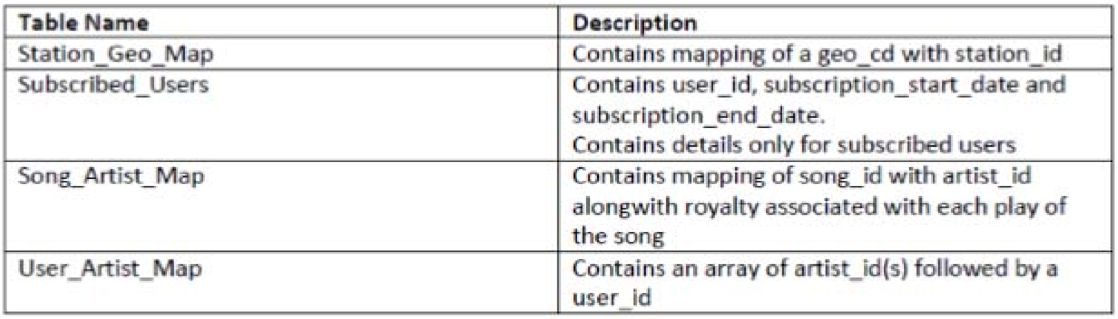


Table Name	Description
Station_Geo_Map	Contains mapping of a geo_cd with station_id
Subscribed_Users	Contains user_id, subscription_start_date and subscription_end_date. Contains details only for subscribed users
Song_Artist_Map	Contains mapping of song_id with artist_id alongwith royalty associated 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

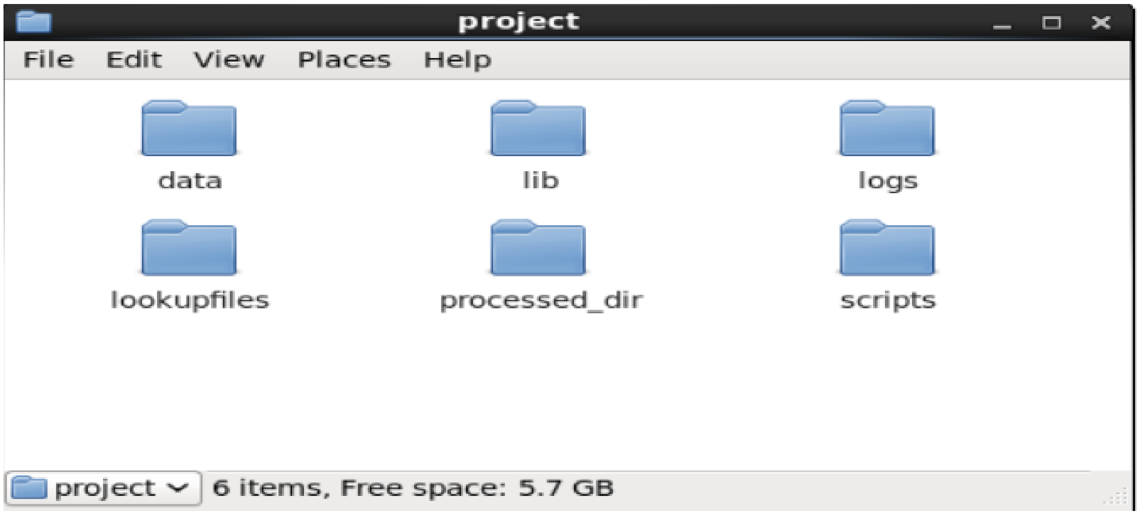
1. Data coming from web applications reside in /data/web and has xml format.

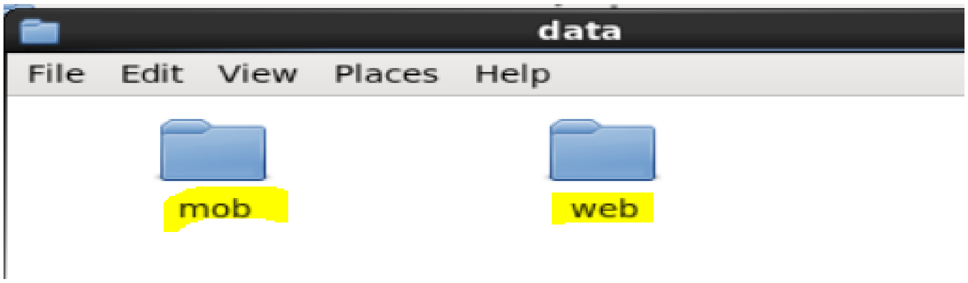
2. Data coming from mobile applications reside in /data/mob and has csv format.

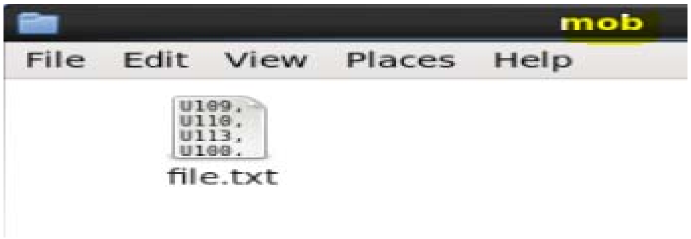
3. Data present in lookup directory should be used in HBase.

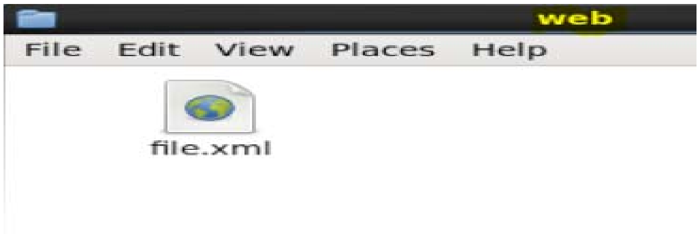
Below is the link for same.

https://drive.google.com/drive/folders/0B_P3pWagdIrrMjJGVlNsSUEtbG8?usp=sharing



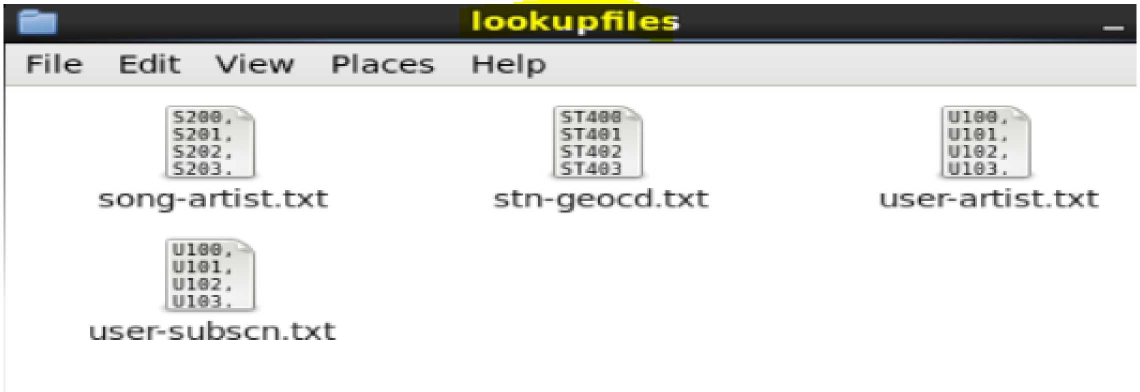






Big Data- Hadoop_Final Project

Music Data Analysis using Hadoop

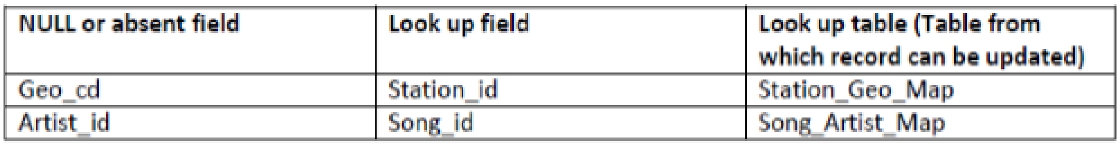


1.4 Data Enrichment Rules for data enrichment,

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 IMPLEMETED 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.

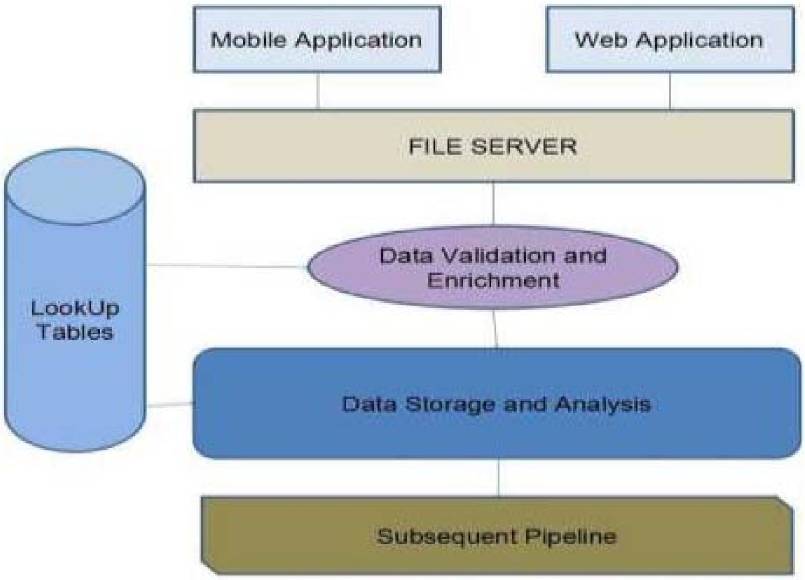
2. Try to make joins as less expensive as possible.

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,

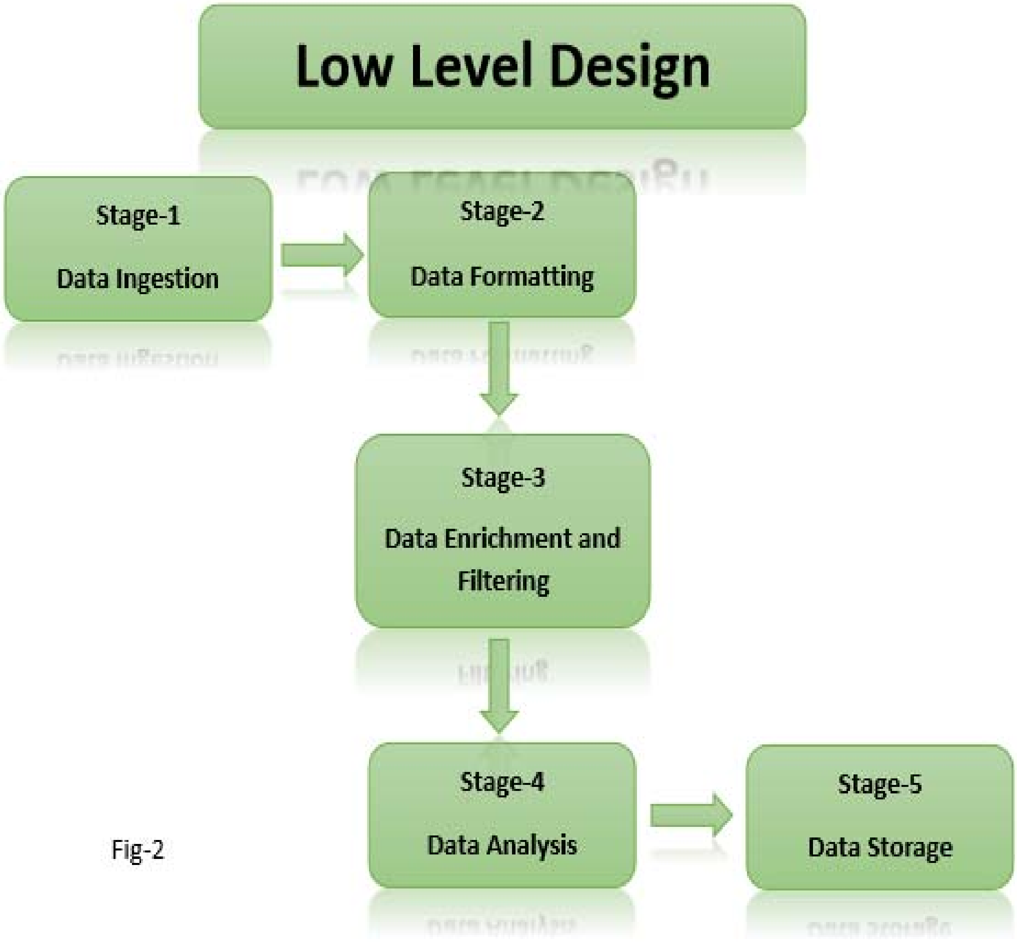


Fig-2

2.2 High Level Design

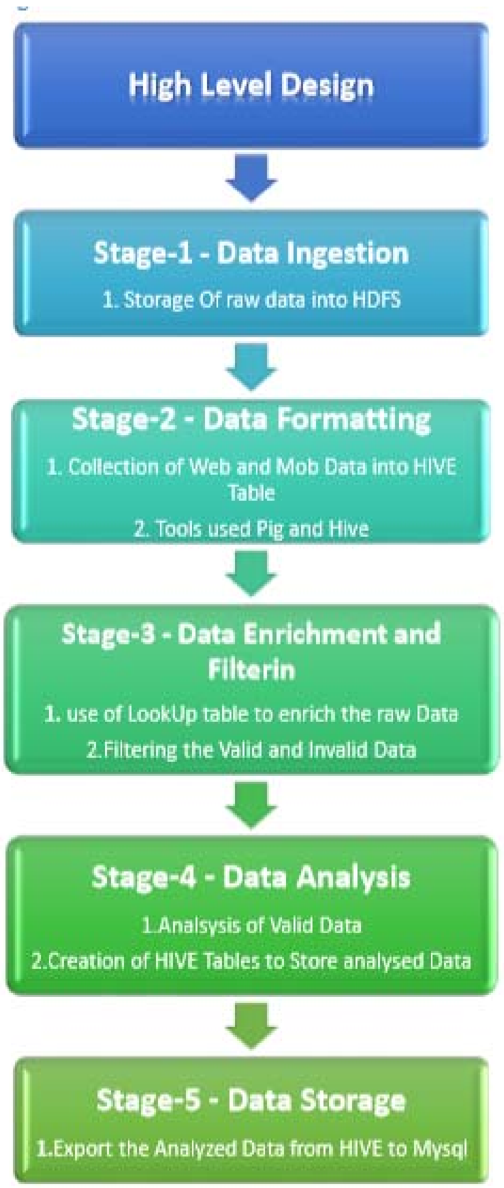

High Level Design
Stage-1 - Data Ingestion
1. Storage Of raw data into HDFS
Stage-2 - Data Formatting
1. Collection of Web and Mob Data into HIVE Table
2. Tools used Pig and Hive
Stage-3 - Data Enrichment and Filterin
1. use of LookUp table to enrich the raw Data
2.Filtering the Valid and Invalid Data
Stage-4 - Data Analysis
1.Analsysis of Valid Data
2.Creation of HIVE Tables to Store analysed Data
Stage-5 - Data Storage
1.Export the Analyzed Data from HIVE to Mysql

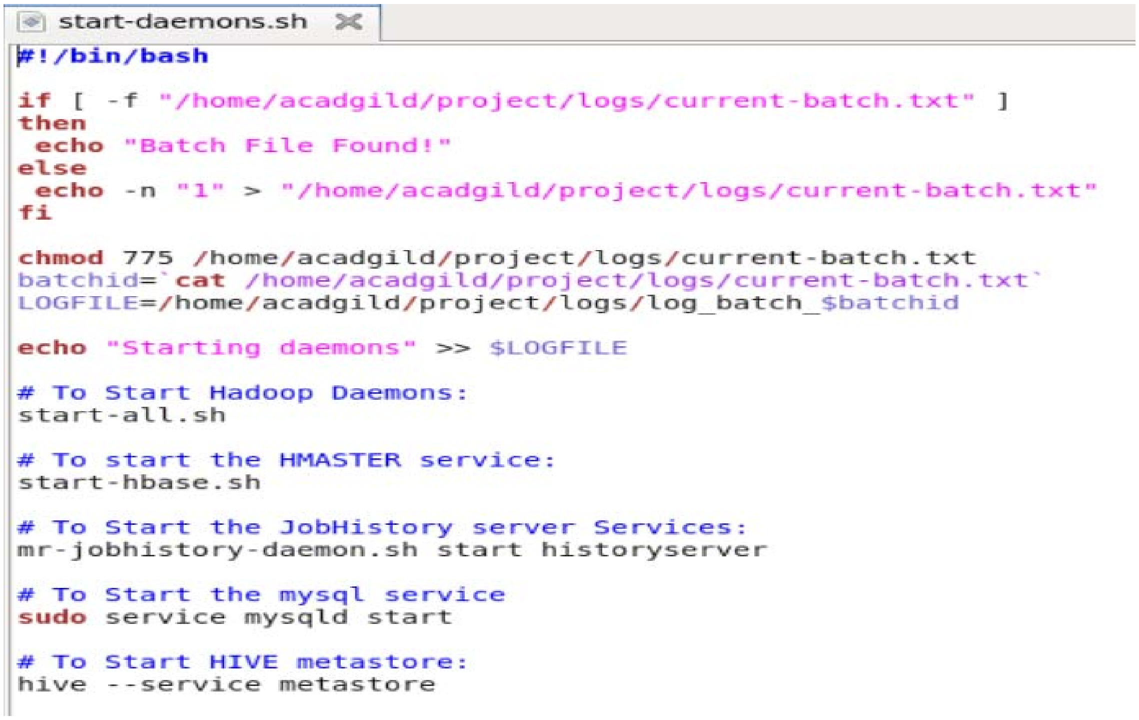
Fig-3

Big Data- Hadoop_Final Project

Music Data Analysis using Hadoop

Section-3-Hadoop Eco-System Implementation

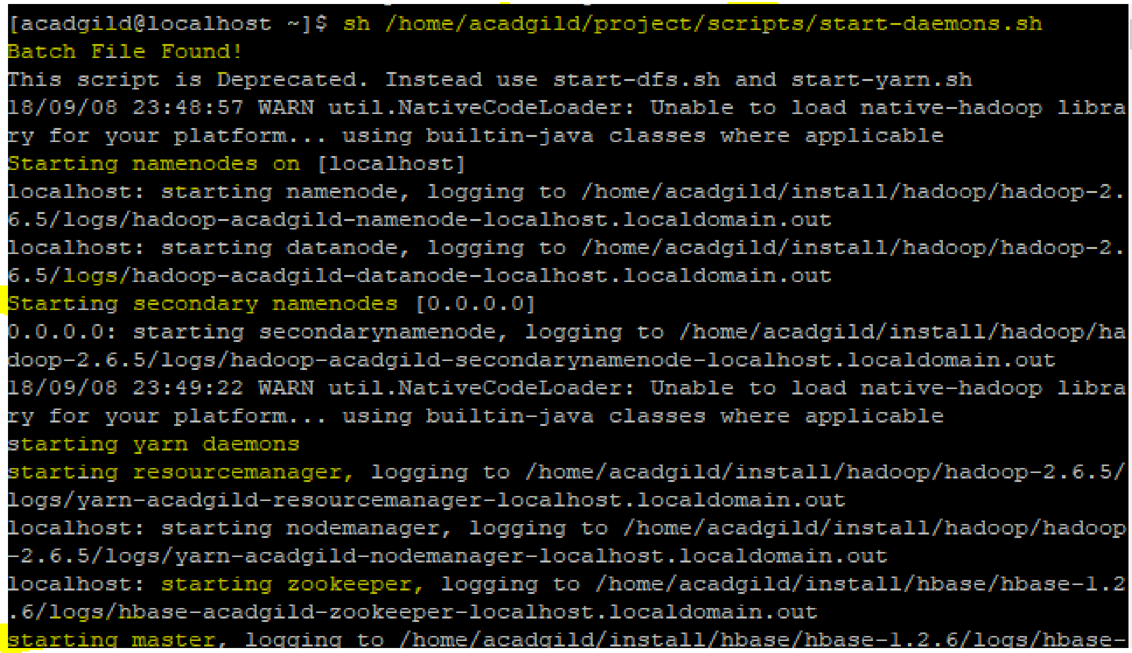
1. We have created a batch file **“start-daemon.sh”** which starts the daemons such as hive, hbase, Mysql and rest of the all hadoop daemons.
   Batch file script,



```
#!/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**

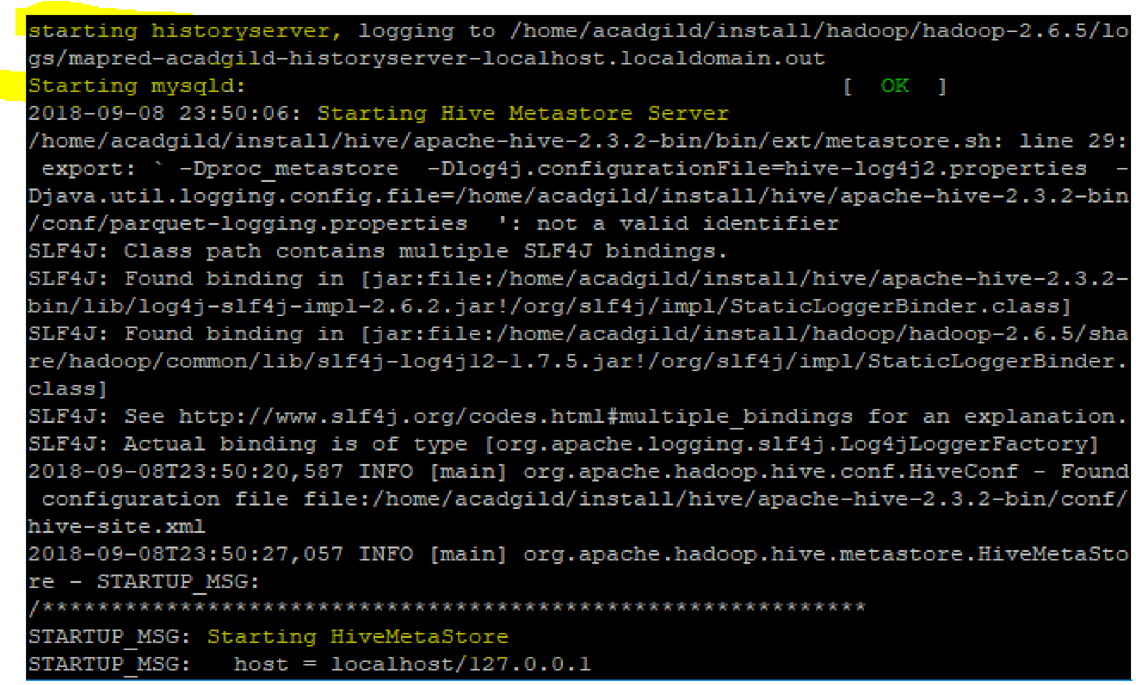
As per the batch file script all the hadoop daemons and the Hive, MySql and Hive daemons are started shown in the below screen shot,



```
[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 libra
ry for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/acadgild/install/hadoop/hadoop-2.
6.5/logs/hadoop-acadgild-namenode-localhost.localdomain.out
localhost: starting datanode, logging to /home/acadgild/install/hadoop/hadoop-2.
6.5/logs/hadoop-acadgild-datanode-localhost.localdomain.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/acadgild/install/hadoop/ha
doop-2.6.5/logs/hadoop-acadgild-secondarynamenode-localhost.localdomain.out
18/09/08 23:49:22 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry 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-
```

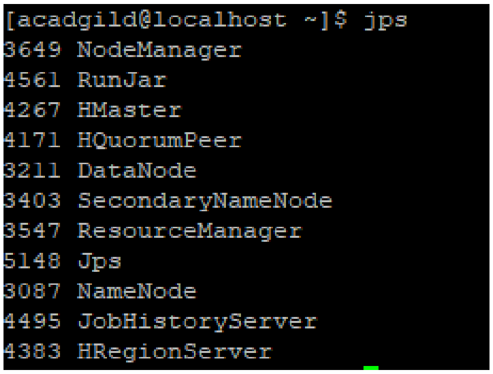
Big Data- Hadoop_Final Project

Music Data Analysis using Hadoop

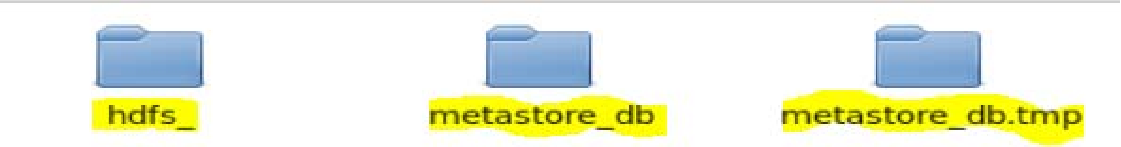


```
starting historyserver, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/lo
gs/mapred-acadgild-historyserver-localhost.localdomain.out
Starting mysqld:                                           [  OK  ]
2018-09-08 23:50:06: Starting Hive Metastore Server
/home/acadgild/install/hive/apache-hive-2.3.2-bin/bin/ext/metastore.sh: 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
```



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.**

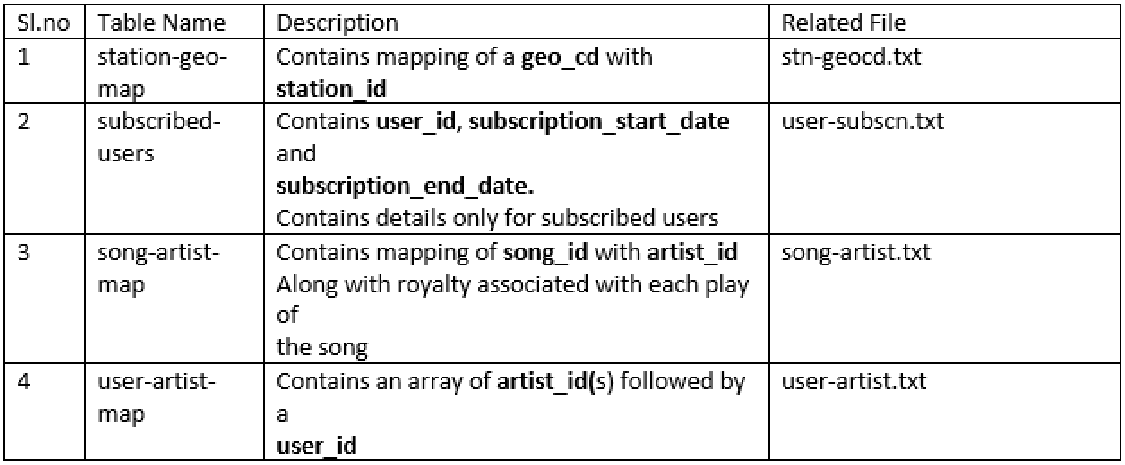
Big Data- Hadoop_Final Project

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 **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

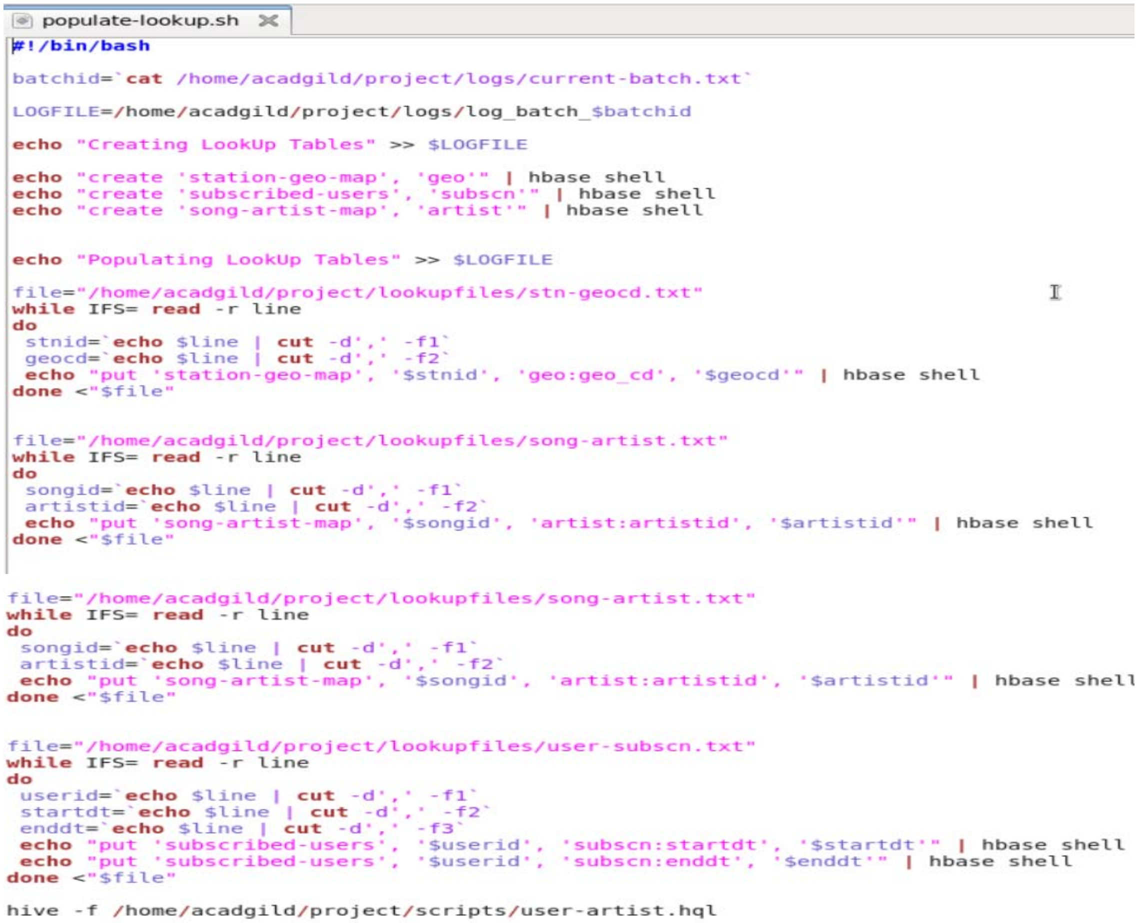
"populate-lookup.sh" script

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.

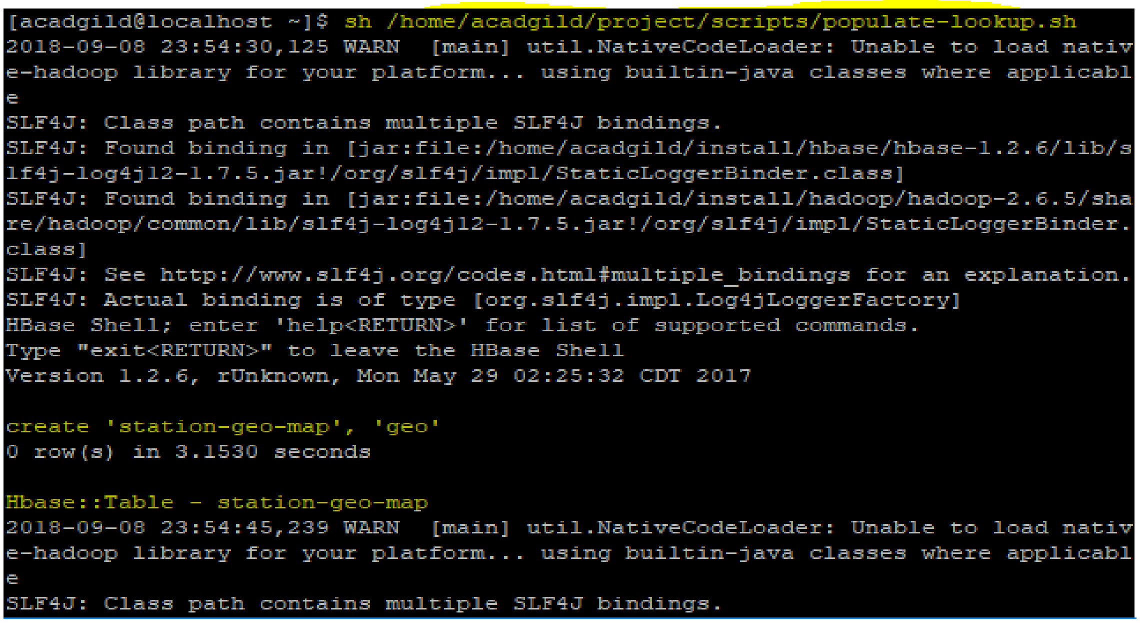
Big Data- Hadoop_Final Project

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 "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
```

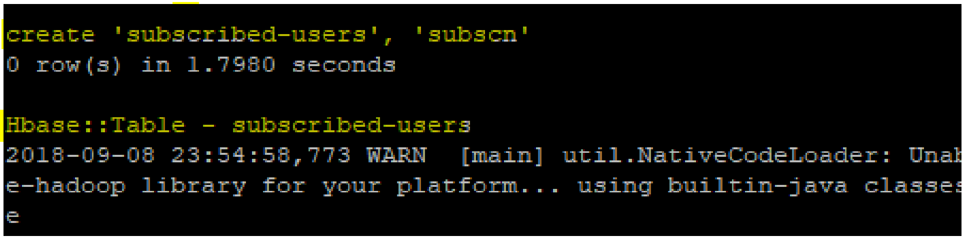
Run the script: ./populate-lookup.sh



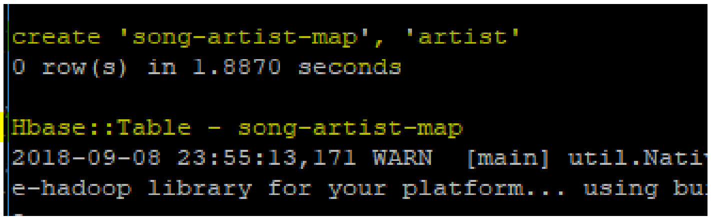
```
[acadgild@localhost ~]$ sh /home/acadgild/project/scripts/populate-lookup.sh
2018-09-08 23:54:30,125 WARN  [main] util.NativeCodeLoader: Unable to load nativ
e-hadoop library for your platform... using builtin-java classes where applicabl
e
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/s
lf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/sha
re/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.
class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

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

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

Big Data- Hadoop_Final Project

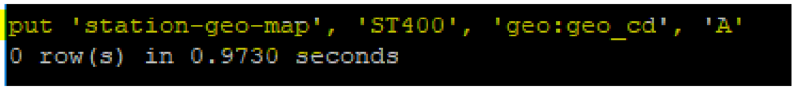
Music Data Analysis using Hadoop



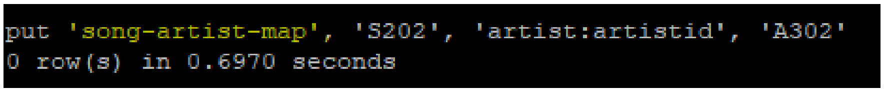
```
create 'subscribed-users', 'subscn'
0 row(s) in 1.7980 seconds

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



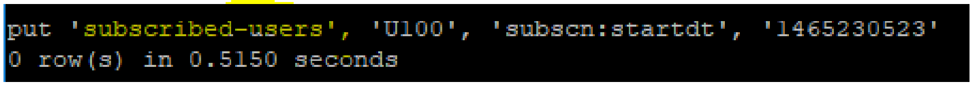
```
create 'song-artist-map', 'artist'
0 row(s) in 1.8870 seconds

Hbase::Table - song-artist-map
2018-09-08 23:55:13,171 WARN  [main] util.Nati
e-hadoop library for your platform... using bu
```



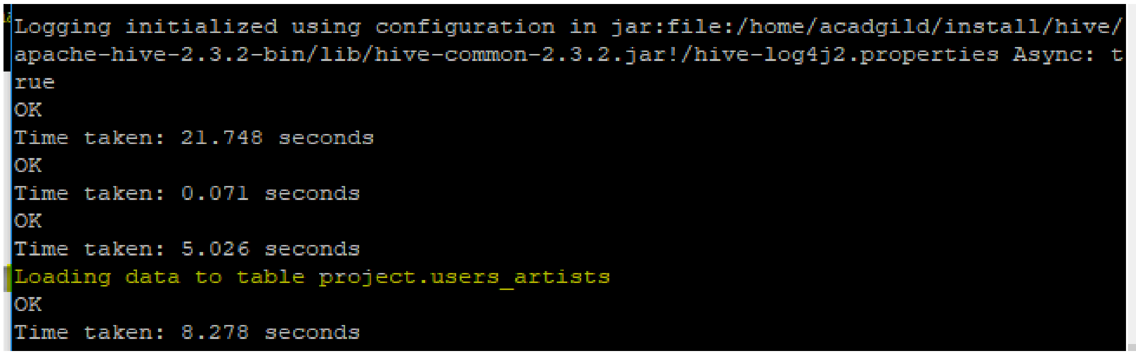
```
put 'station-geo-map', 'ST400', 'geo:geo_cd', 'A'
0 row(s) in 0.9730 seconds
```



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



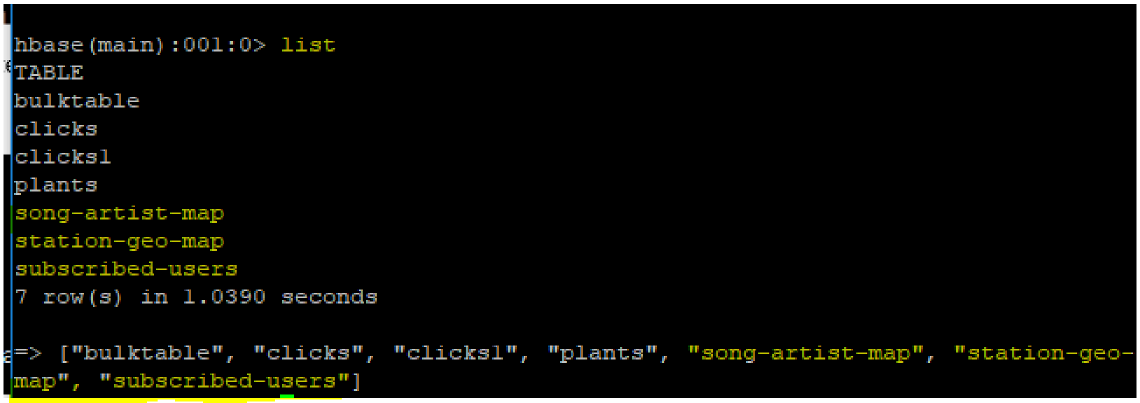
```
put 'subscribed-users', 'U100', 'subscn:startdt', '1465230523'
0 row(s) in 0.5150 seconds
```



```
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: 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
```

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

Lookup Tables in the hbase shell,



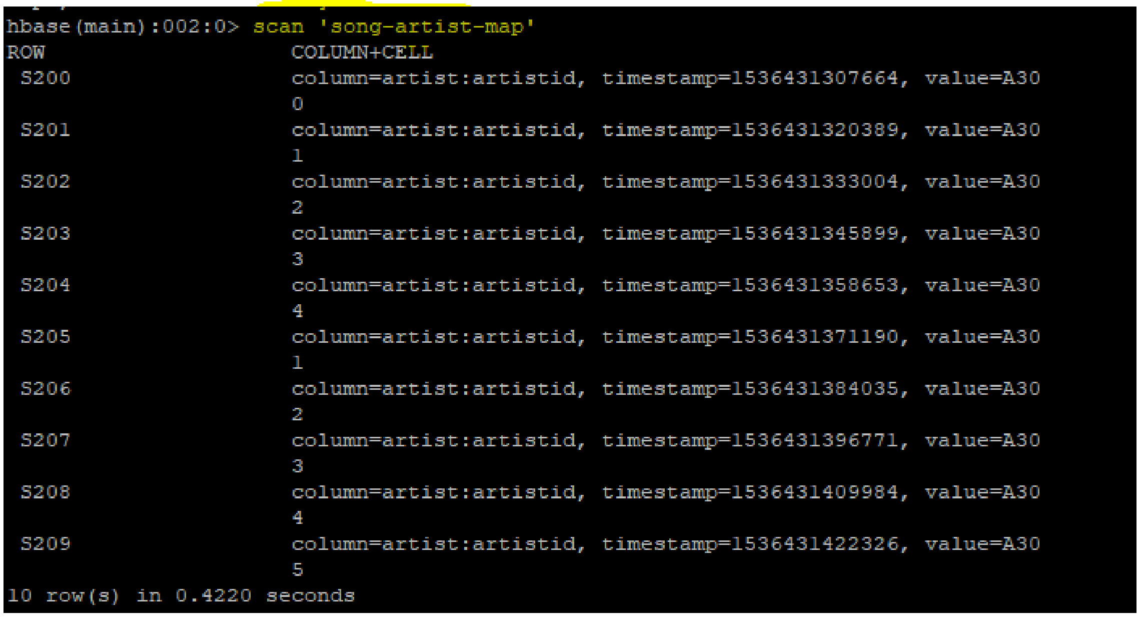
```
hbase(main):001:0> list
TABLE
bulktable
clicks
clicks1
plants
song-artist-map
station-geo-map
subscribed-users
7 row(s) in 1.0390 seconds

=> ["bulktable", "clicks", "clicks1", "plants", "song-artist-map", "station-geo-
map", "subscribed-users"]
```

Big Data- Hadoop_Final Project

Music Data Analysis using Hadoop

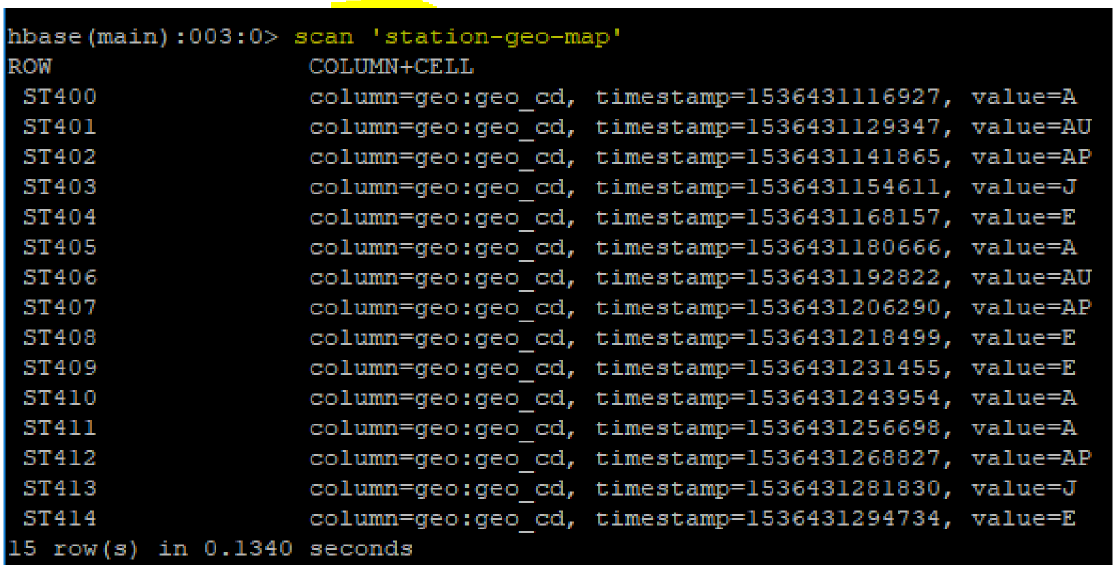
The values loaded in the Lookup tables are shown below,

song-artist-map



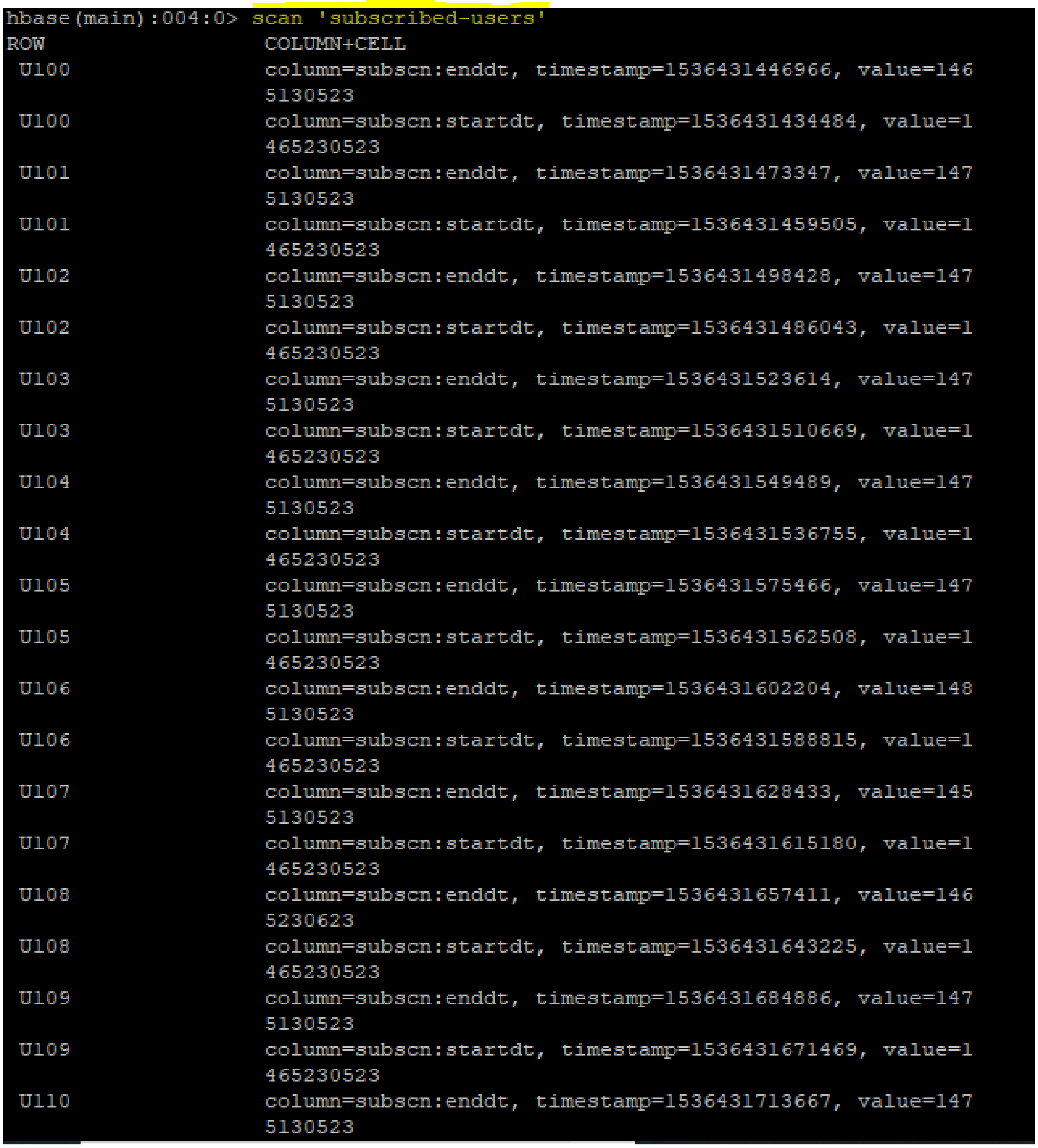
```
hbase(main):002:0> scan 'song-artist-map'
ROW                  COLUMN+CELL
 S200                column=artist:artistid, timestamp=1536431307664, value=A30
                     0
 S201                column=artist:artistid, timestamp=1536431320389, value=A30
                     1
 S202                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
```

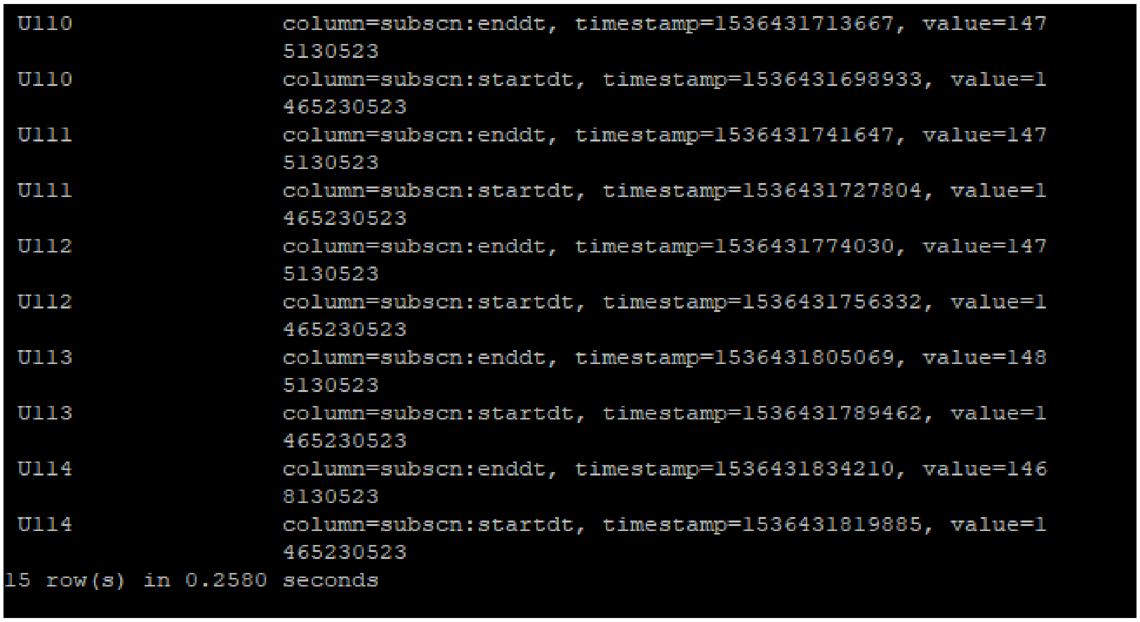
subscribed-users



```
hbase(main):004:0> scan 'subscribed-users'
ROW                   COLUMN+CELL
 U100                 column=subscn:enddt, timestamp=1536431446966, value=146
                      5130523
 U100                 column=subscn:startdt, timestamp=1536431434484, value=1
                      465230523
 U101                 column=subscn:enddt, timestamp=1536431473347, value=147
                      5130523
 U101                 column=subscn:startdt, timestamp=1536431459505, value=1
                      465230523
 U102                 column=subscn:enddt, timestamp=1536431498428, value=147
                      5130523
 U102                 column=subscn:startdt, timestamp=1536431486043, value=1
                      465230523
 U103                 column=subscn:enddt, timestamp=1536431523614, value=147
                      5130523
 U103                 column=subscn:startdt, timestamp=1536431510669, value=1
                      465230523
 U104                 column=subscn:enddt, timestamp=1536431549489, value=147
                      5130523
 U104                 column=subscn:startdt, timestamp=1536431536755, value=1
                      465230523
 U105                 column=subscn:enddt, timestamp=1536431575466, value=147
                      5130523
 U105                 column=subscn:startdt, timestamp=1536431562508, value=1
                      465230523
 U106                 column=subscn:enddt, timestamp=1536431602204, value=148
                      5130523
 U106                 column=subscn:startdt, timestamp=1536431588815, value=1
                      465230523
 U107                 column=subscn:enddt, timestamp=1536431628433, value=145
                      5130523
 U107                 column=subscn:startdt, timestamp=1536431615180, value=1
                      465230523
 U108                 column=subscn:enddt, timestamp=1536431657411, value=146
                      5230623
 U108                 column=subscn:startdt, timestamp=1536431643225, value=1
                      465230523
 U109                 column=subscn:enddt, timestamp=1536431684886, value=147
                      5130523
 U109                 column=subscn:startdt, timestamp=1536431671469, value=1
                      465230523
 U110                 column=subscn:enddt, timestamp=1536431713667, value=147
                      5130523
```

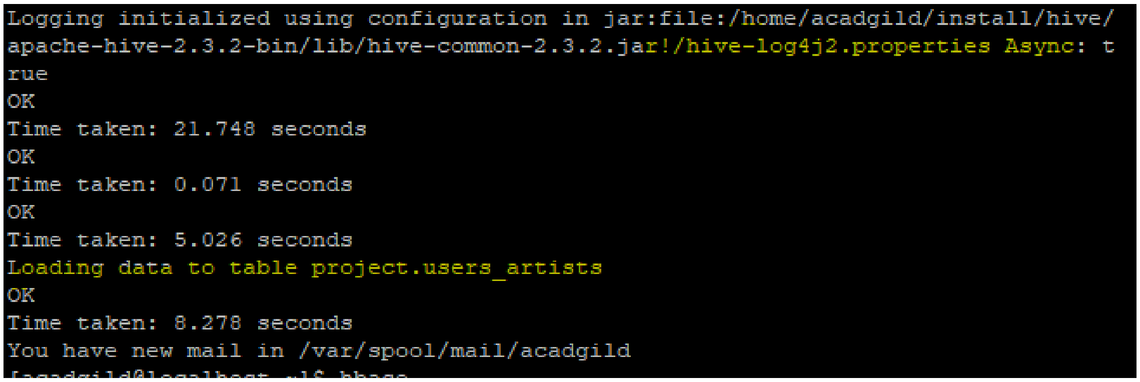
Big Data- Hadoop_Final Project

Music Data Analysis using Hadoop

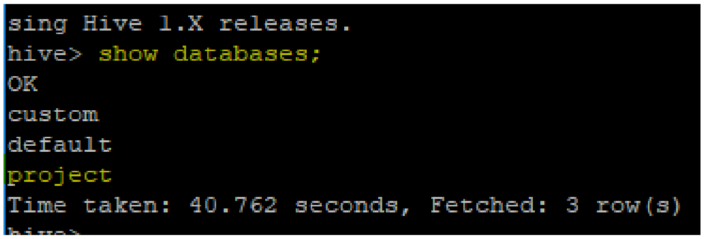


```
 U110                 column=subscn:enddt, timestamp=1536431713667, value=147
                      5130523
 U110                 column=subscn:startdt, timestamp=1536431698933, value=1
                      465230523
 U111                 column=subscn:enddt, timestamp=1536431741647, value=147
                      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
```

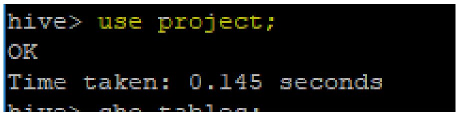
We have successfully created the lookup tables in the Hbase.
The populate-lookup.sh also creates a lookup table **“users_artists”** in the HIVE, loading the data from the **user-artist.txt,** the below screen shot shows that the table has been created in the HIVE.



```
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>
```

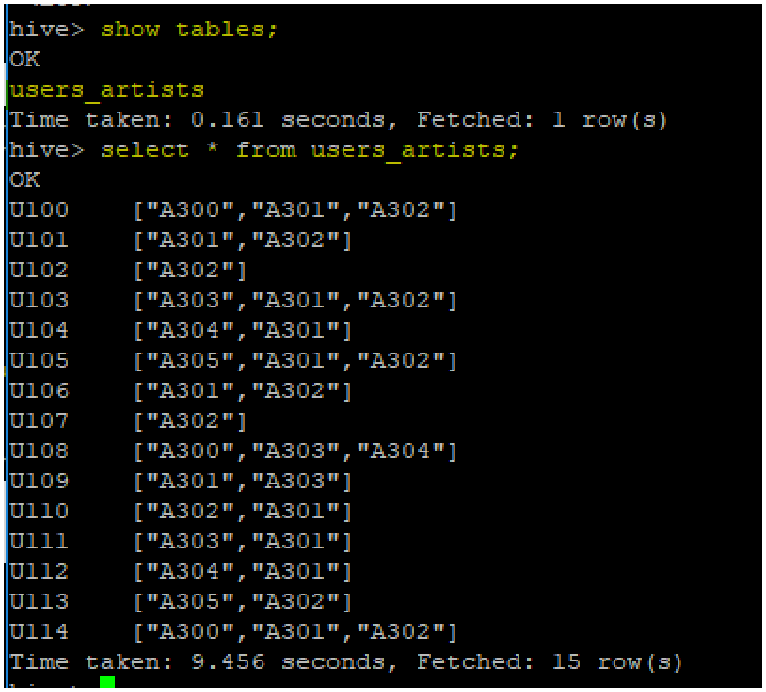


```
hive> use project;
OK
Time taken: 0.145 seconds
hive> sho tables;
```

Big Data- Hadoop_Final Project

Music Data Analysis using Hadoop

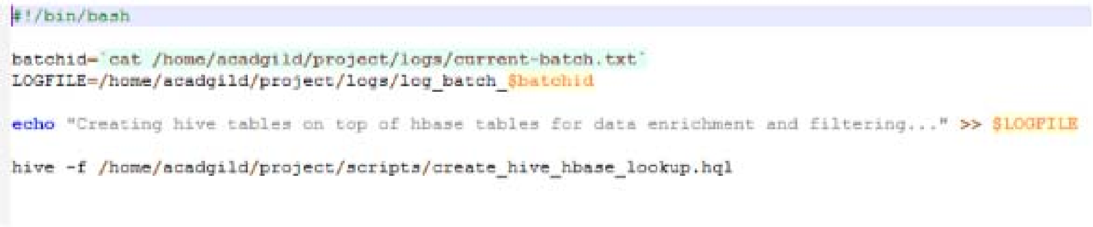
hive> Select * From users_artists;



```
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 theses lookup tables in hive using the Hbase Storage Handler.
With the help of **“data_enrichment_filtering_schema.sh”** file we will create hive tables on the top of Hbase tables using **“create_hive_hbase_lookup.hql”**

Creating Hive Tables on the top of Hbase:
In this section with the help of Hbase storage handler & SerDe properties we are creating the hive external tables by matching the columns of Hbase tables to hive tables.
Run the script: ./data_enrichment_filtering_schema.sh,



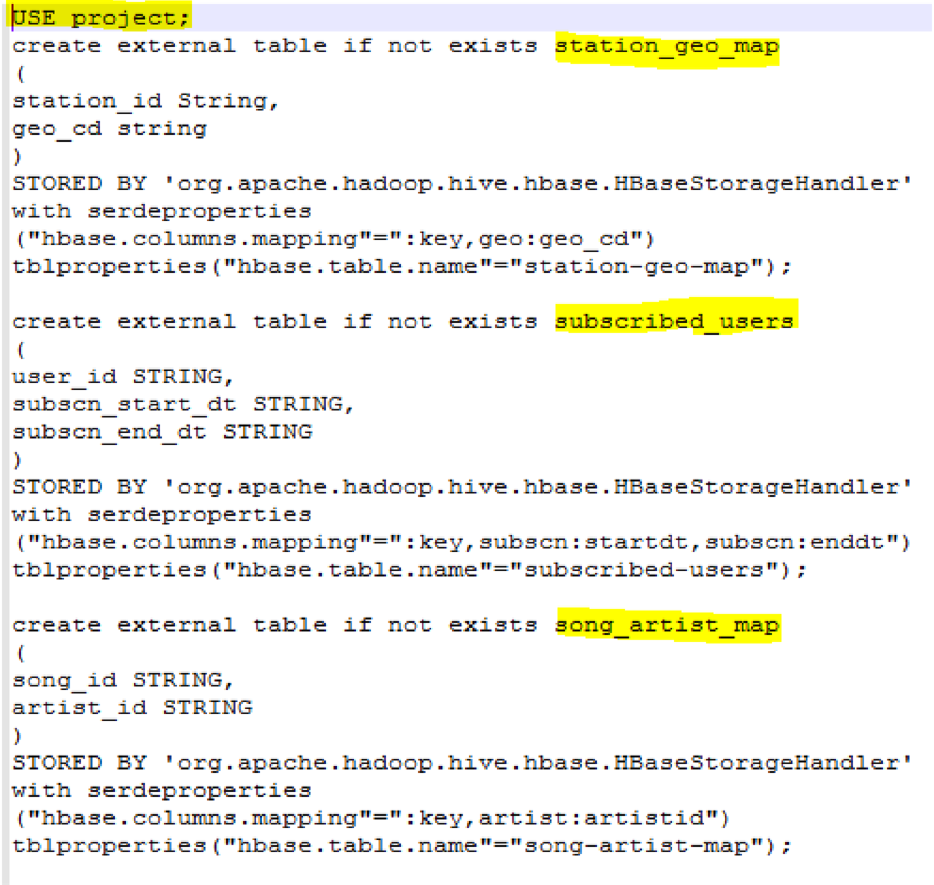
```
#!/bin/bash

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

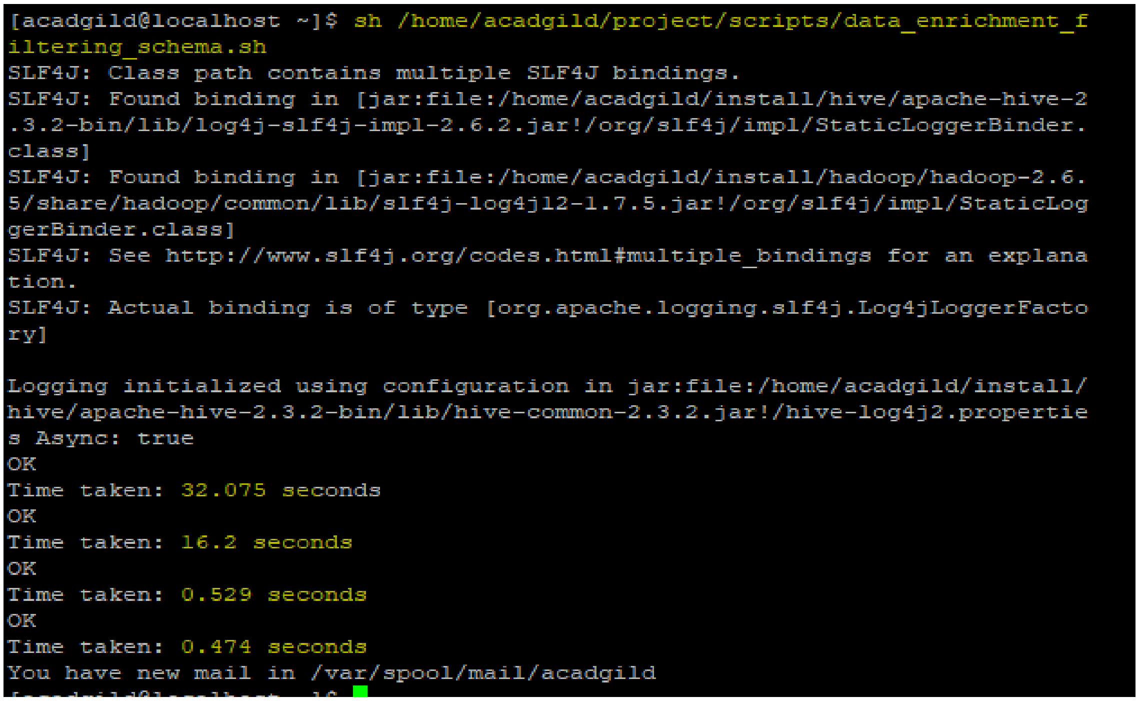


```
USE project;
create external table if not exists station_geo_map
(
station_id String,
geo_cd string
)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
with serdeproperties
("hbase.columns.mapping"=":key,geo:geo_cd")
tblproperties("hbase.table.name"="station-geo-map");

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”**

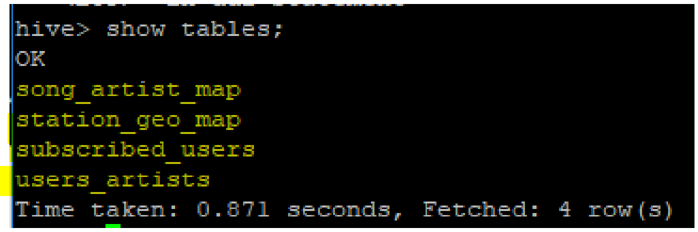
Big Data- Hadoop_Final Project

Music Data Analysis using Hadoop



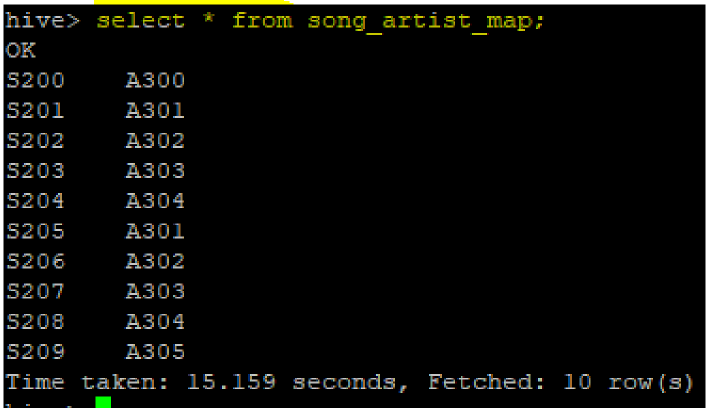
```
[acadgild@localhost ~]$ sh /home/acadgild/project/scripts/data_enrichment_f
iltering_schema.sh
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hive/apache-hive-2
.3.2-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.
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/StaticLog
gerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explana
tion.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFacto
ry]

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.propertie
s 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

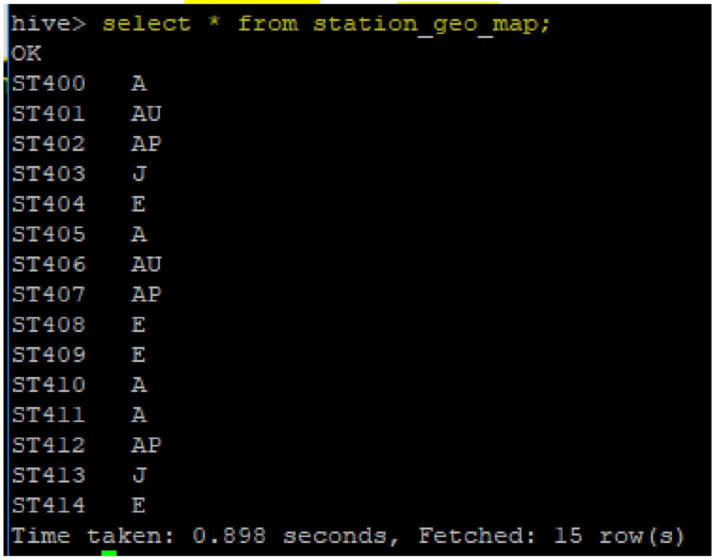


```
hive> select * from song_artist_map;
OK
S200    A300
S201    A301
S202    A302
S203    A303
S204    A304
S205    A301
S206    A302
S207    A303
S208    A304
S209    A305
Time taken: 15.159 seconds, Fetched: 10 row(s)
```

Big Data- Hadoop_Final Project

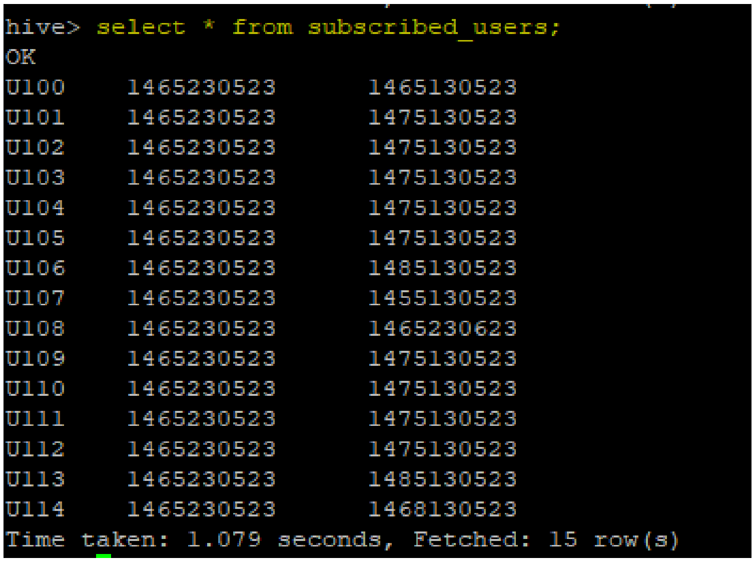
Music Data Analysis using Hadoop

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

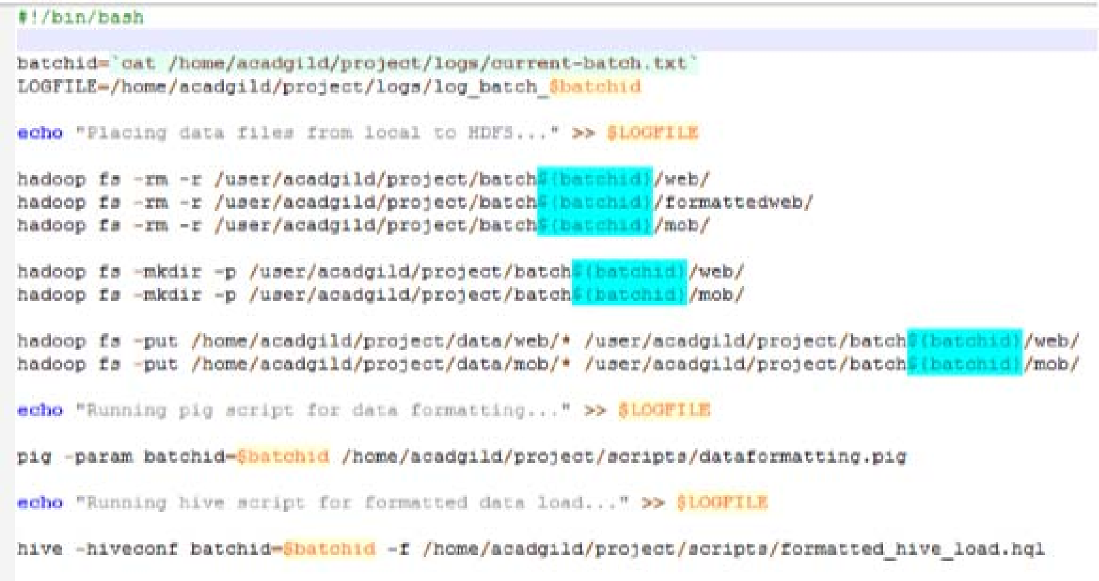


```
hive> select * from subscribed_users;
OK
U100    1465230523      1465130523
U101    1465230523      1475130523
U102    1465230523      1475130523
U103    1465230523      1475130523
U104    1465230523      1475130523
U105    1465230523      1475130523
U106    1465230523      1485130523
U107    1465230523      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



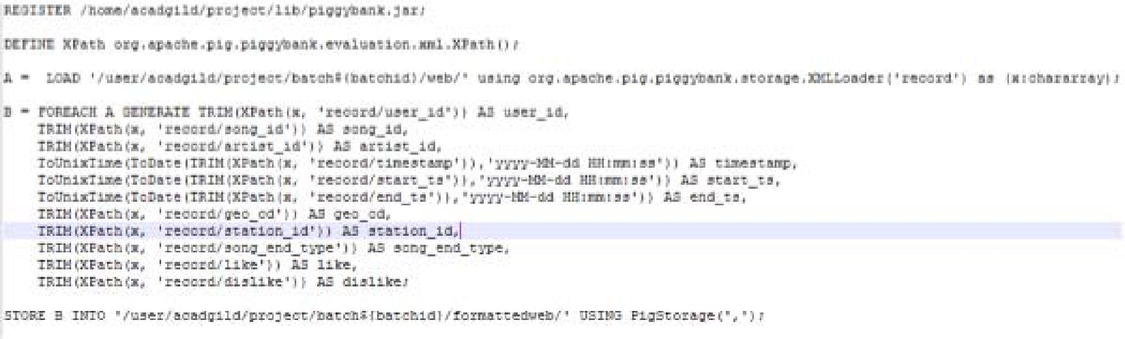
```
#!/bin/bash

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 based on batch ID's

Dataformatting.pig

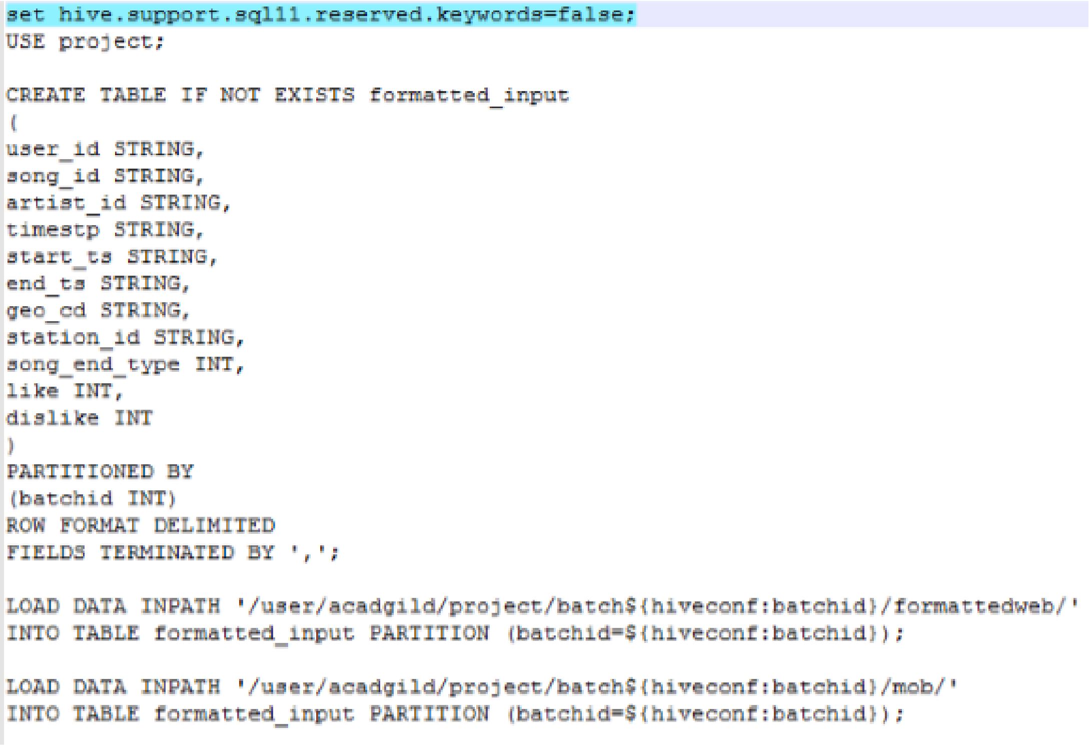


```
REGISTER /home/acadgild/project/lib/piggybank.jar;

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

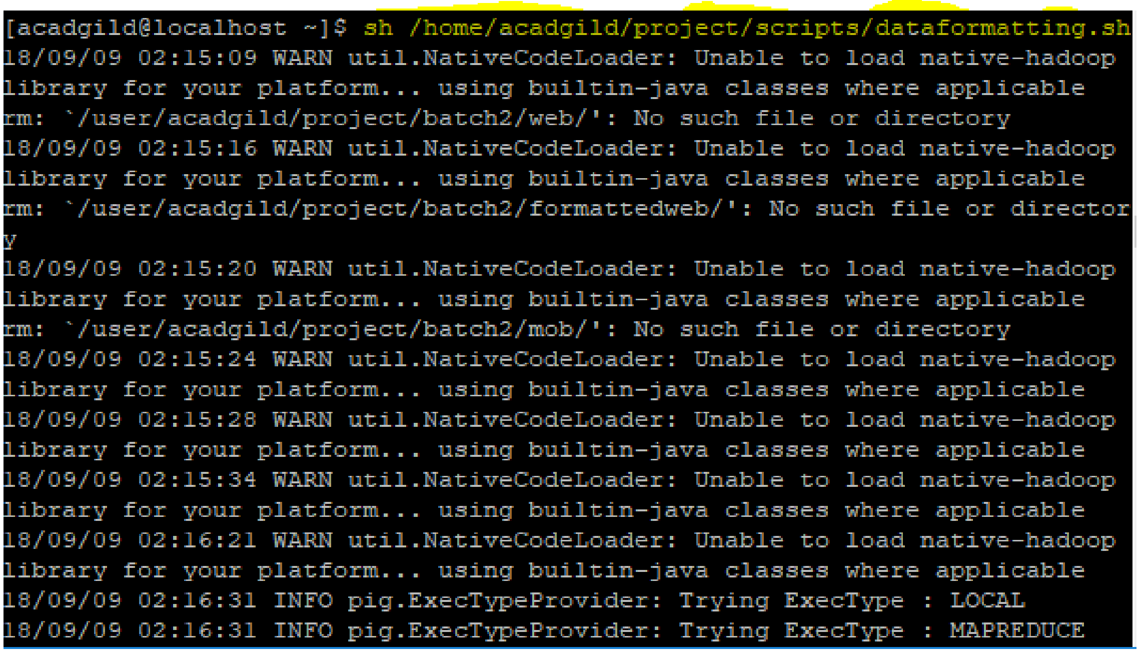
Big Data- Hadoop_Final Project

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,
timestp STRING,
start_ts STRING,
end_ts STRING,
geo_cd STRING,
station_id STRING,
song_end_type INT,
like INT,
dislike INT
)
PARTITIONED BY
(batchid INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

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

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

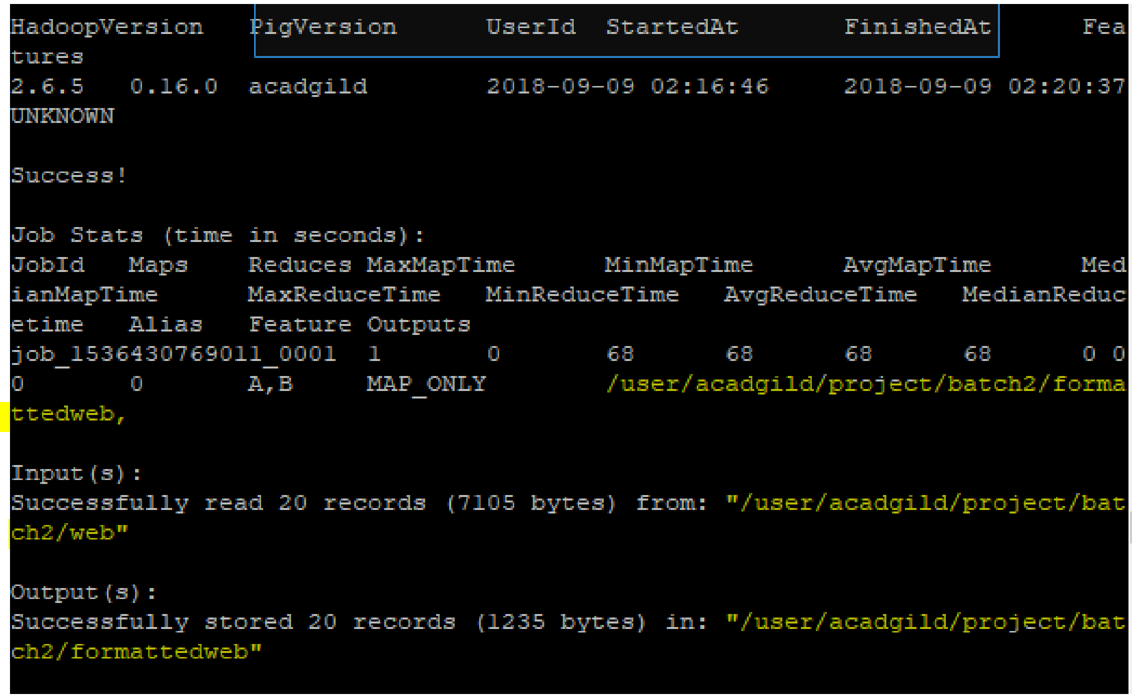
In the below screenshot we can see the data both the scripts in action, first pig script will parse the data and then hive script will load the data into hive terminal successfully.
Pig script successful completion,



```
[acadgild@localhost ~]$ 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

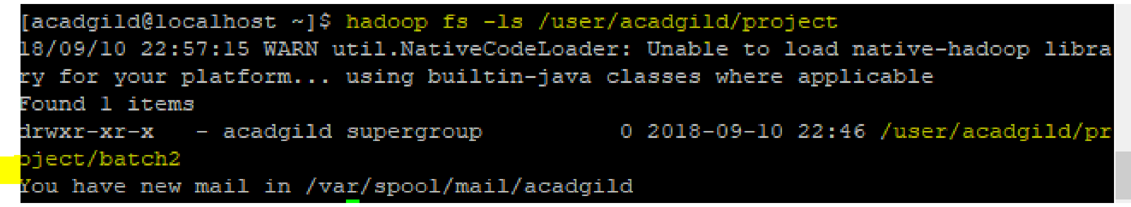
Music Data Analysis using Hadoop



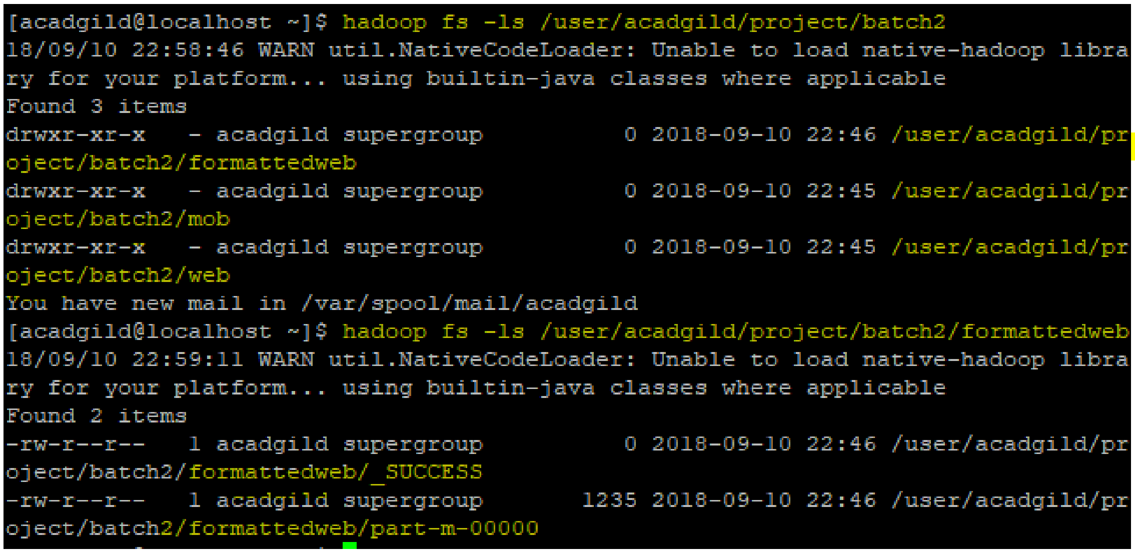
```
HadoopVersion    PigVersion      UserId   StartedAt           FinishedAt          Fea
tures
2.6.5    0.16.0   acadgild        2018-09-09 02:16:46     2018-09-09 02:20:37
UNKNOWN

Success!

Job Stats (time in seconds):
JobId    Maps     Reduces MaxMapTime      MinMapTime      AvgMapTime      Med
ianMapTime       MaxReduceTime    MinReduceTime    AvgReduceTime    MedianReduc
etime    Alias    Feature Outputs
job_1536430769011_0001  1       0       68      68      68      68      0 0
0        0        A,B     MAP_ONLY        /user/acadgild/project/batch2/forma
ttedweb,

Input(s):
Successfully read 20 records (7105 bytes) from: "/user/acadgild/project/bat
ch2/web"

Output(s):
Successfully stored 20 records (1235 bytes) in: "/user/acadgild/project/bat
ch2/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 libra
ry 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/pr
oject/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 libra
ry 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/pr
oject/batch2/formattedweb
drwxr-xr-x   - acadgild supergroup          0 2018-09-10 22:45 /user/acadgild/pr
oject/batch2/mob
drwxr-xr-x   - acadgild supergroup          0 2018-09-10 22:45 /user/acadgild/pr
oject/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 libra
ry 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/pr
oject/batch2/formattedweb/_SUCCESS
-rw-r--r--   1 acadgild supergroup       1235 2018-09-10 22:46 /user/acadgild/pr
oject/batch2/formattedweb/part-m-00000
```

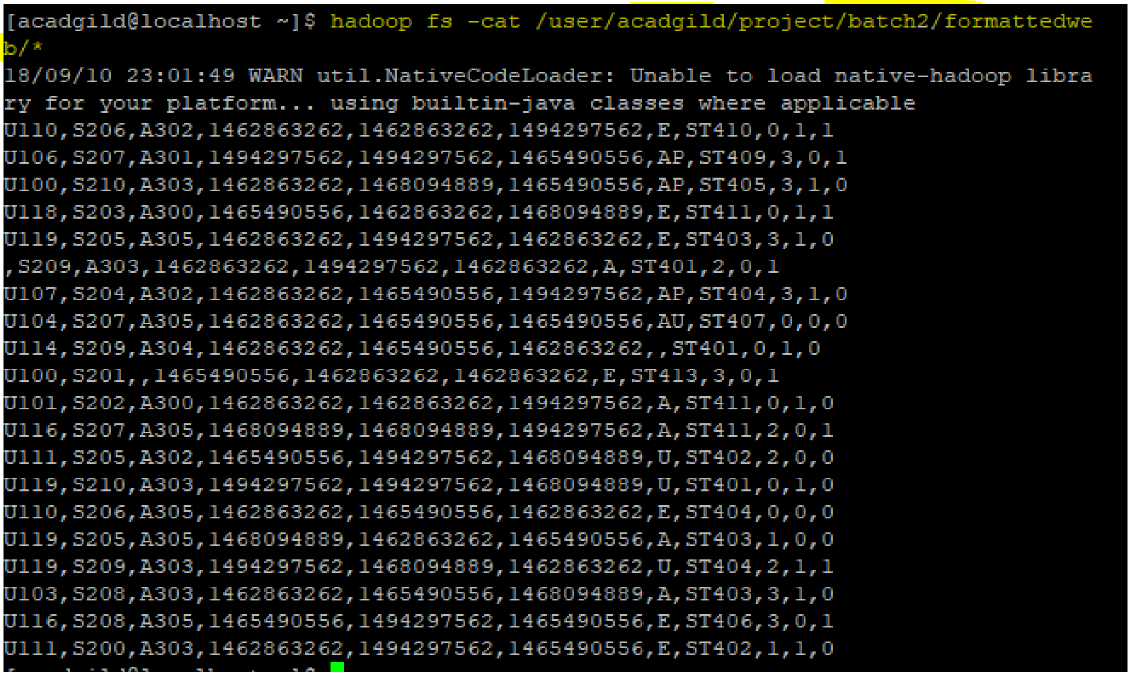
Big Data- Hadoop_Final Project

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/formattedwe
b/*
18/09/10 23:01:49 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry 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
```