ACADGILD

Big Data Engineering with Hadoop & Spark

PROJECT REPORT

MUSIC DATA ANALYSIS

PREPARED BY

AMAN TURATE

TABLE OF CONTENTS

INTRODUCTION

A leading music-catering company is planning to analyse large amount of data received from varieties of sources, namely mobile app and website to track the behaviour 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.

DATASET:

Below is the link for same.

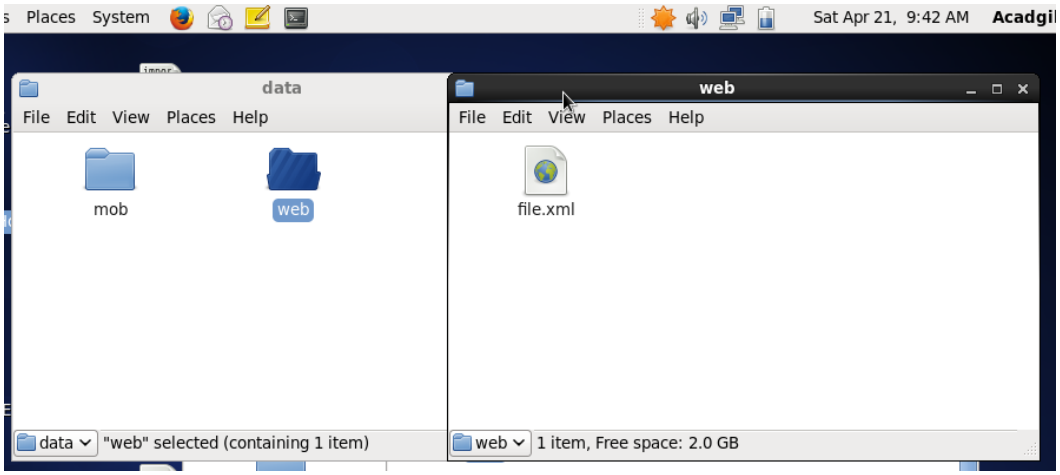
https://drive.google.com/drive/folders/0B_P3pWagdIrrMjJGVlNsSUEtbG8?usp=sharing

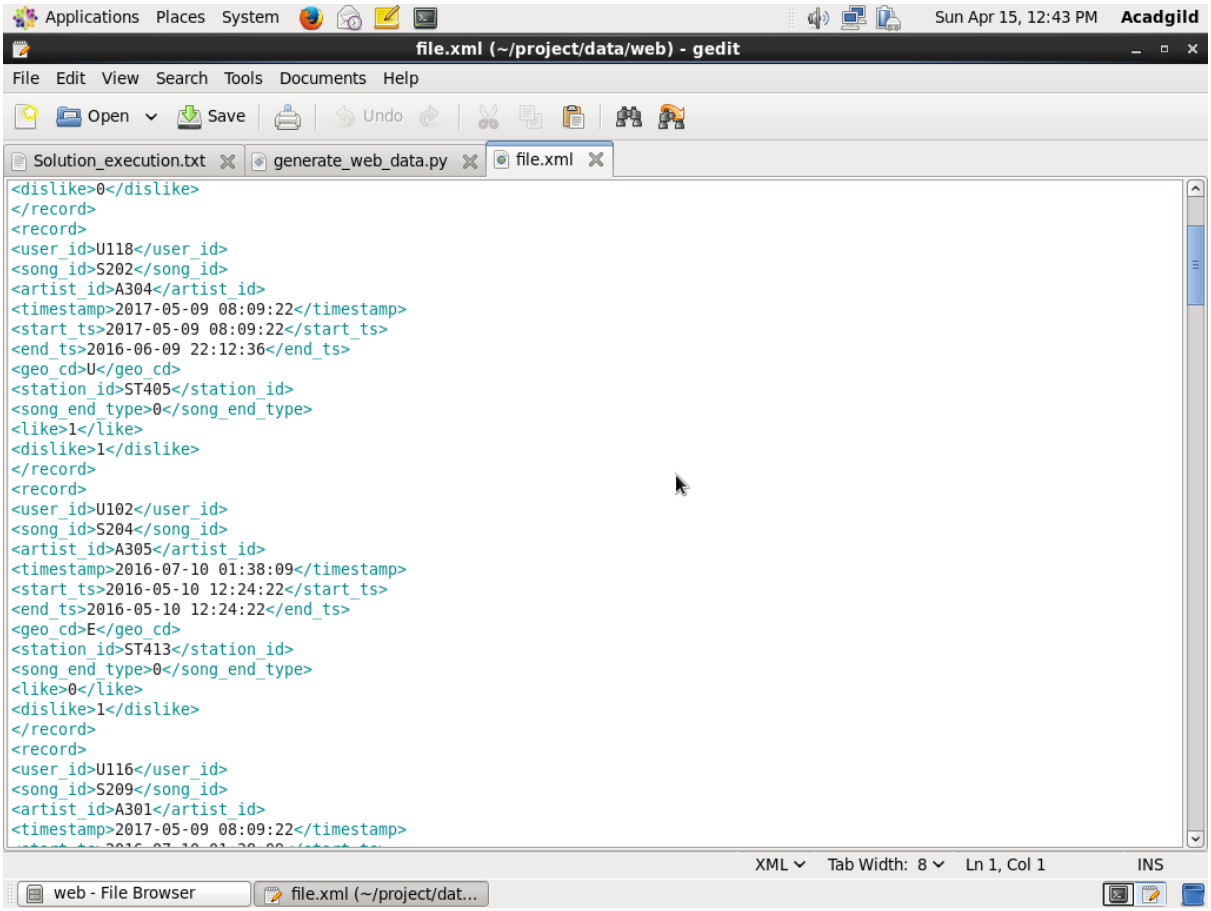
Data Description

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

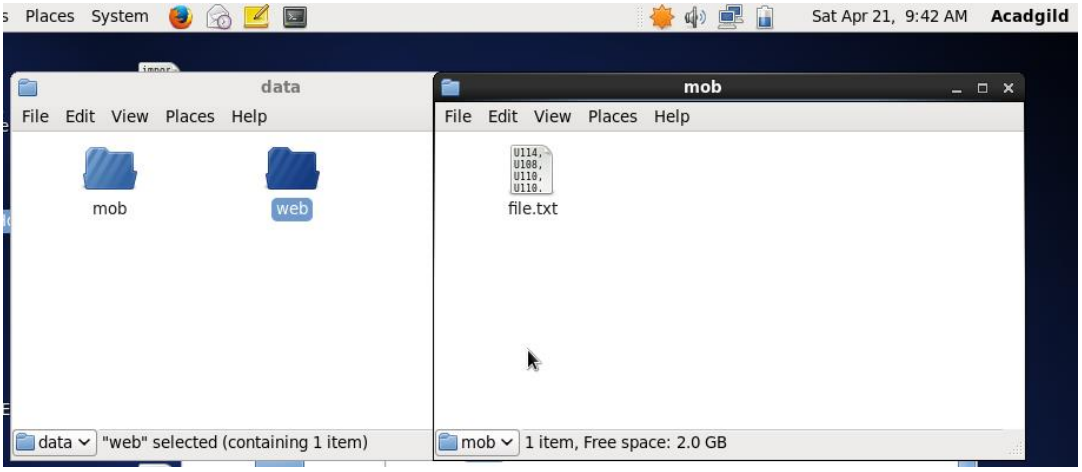
Data Files

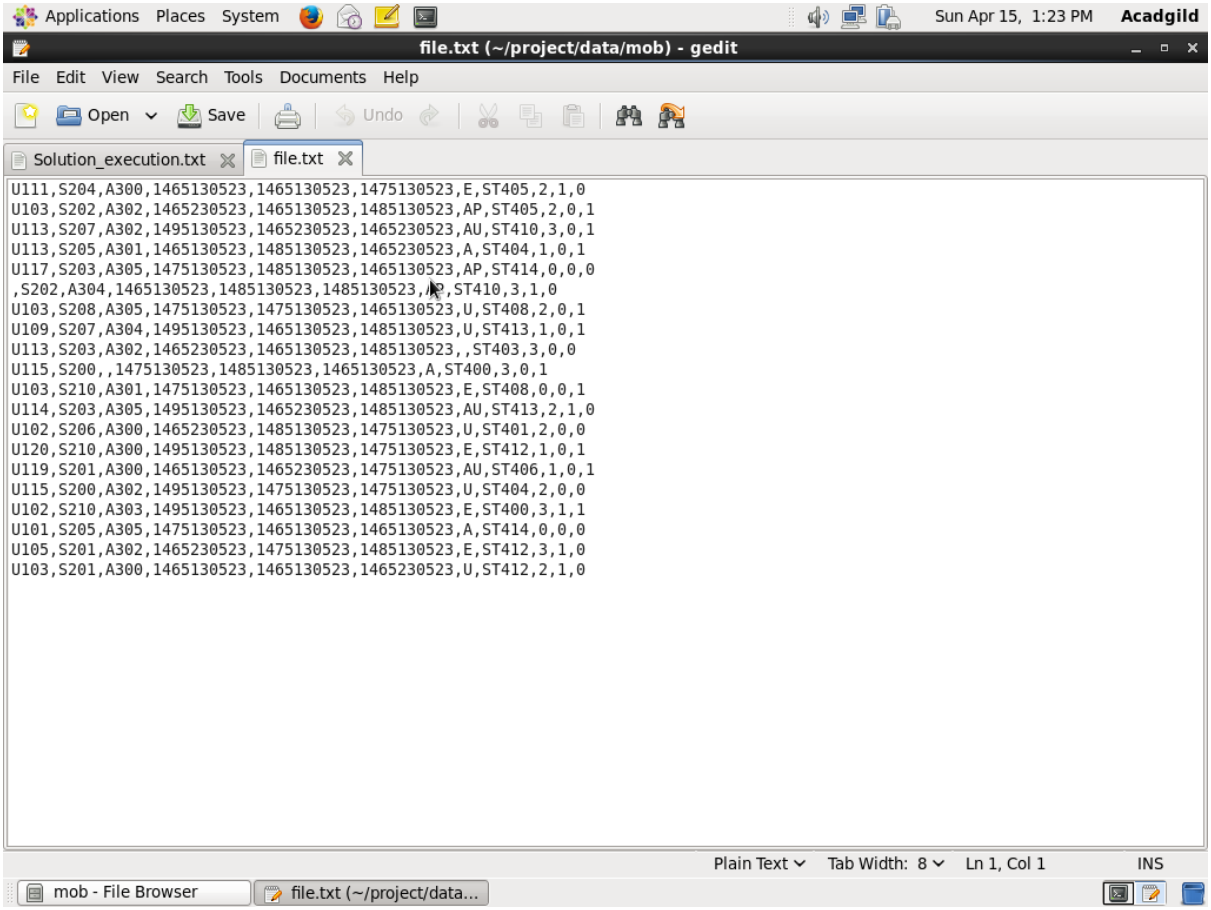
- ✓ Sample Data coming from web applications reside in /data/web and has xml format.





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



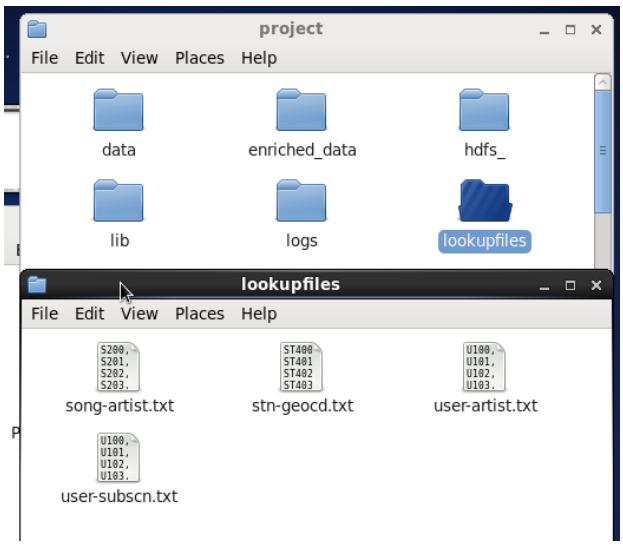


LookUp Tables

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

- ✓ Sample Data present in lookup directory is being used in HBase.



Song-artist.txt

Columns: song_id, artist_id

```
S200,A300
S201,A301
S202,A302
S203,A303
S204,A304
S205,A301
S206,A302
S207,A303
S208,A304
S209,A305
```

user-subscn.txt

user_id,subscn_start_dt, subscn_end_dt

```
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
```

stn-geocd.txt

Columns: station_id, geo_cd

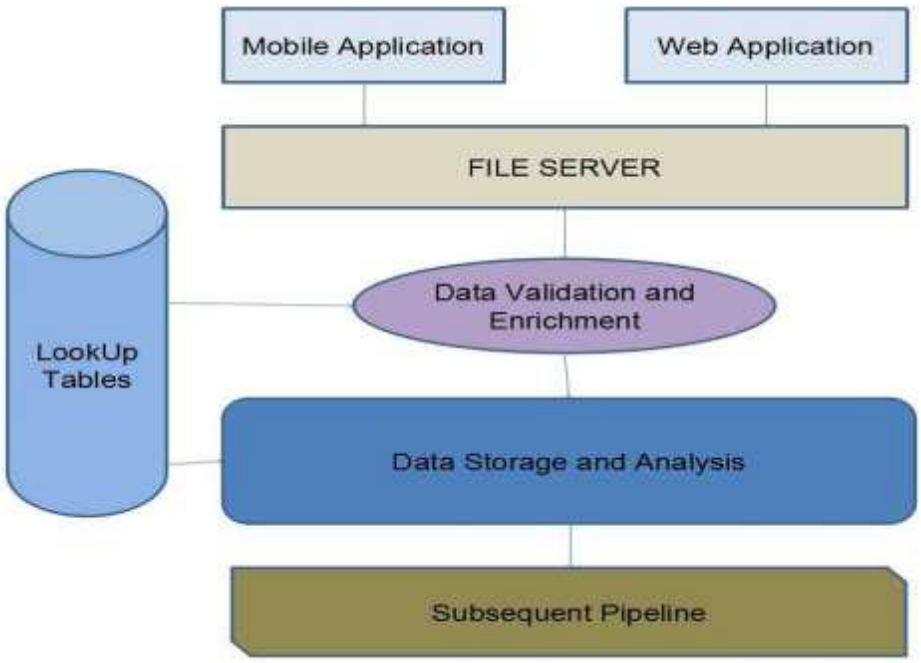
```
ST400,A
ST401,AU
ST402,AP
ST403,J
ST404,E
ST405,A
ST406,AU
ST407,AP
ST408,E
ST409,E
ST410,A
ST411,A
ST412,AP
ST413,J
ST414,E
```

user-artist.txt

user_id, artists_array

```
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
```

FLOW OF OPERATIONS :



Steps to perform Music Data Analysis :

1) Scheduling Crontab job
2) Launch all necessary daemons.
3) Populate look up tables into HBase.
4) Perform Data Formatting.
5) Perform Data Enrichment and Cleaning.
6) Perform Data Analysis.

1) Scheduling Crontab job :-

✓ Open Crontab File using following command -

```
sudo crontab -e
```

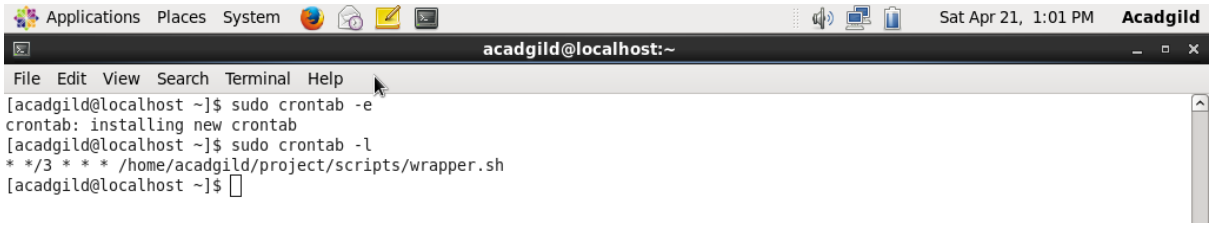
✓ Press I to enter into insert mode.

✓ Insert the following command -

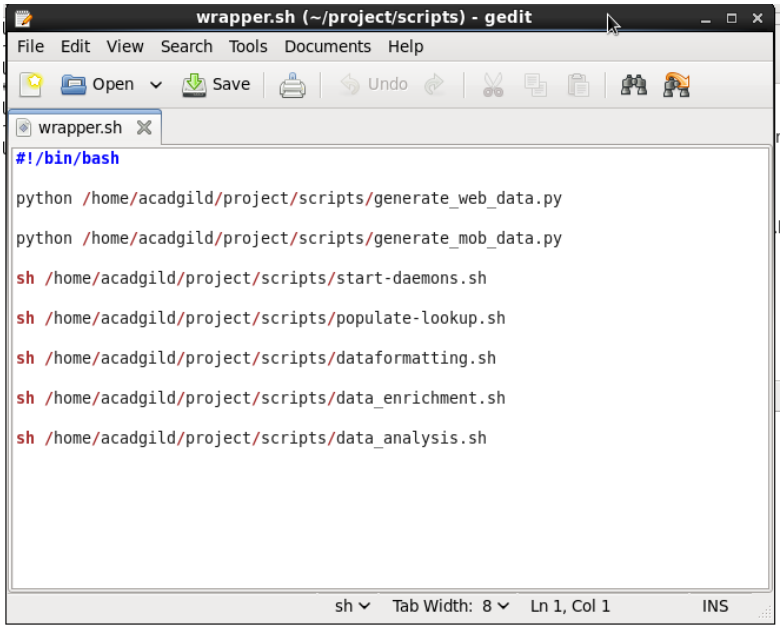
```
* */3 * * * /home/acadgild/project/scripts/wrapper.sh
```

Explaination : Crontab is used for Job Scheduling. In the -e mode, Crontab schedules execution of commands by a regular user.

The statement above runs the **wrapper.sh** shell script every 3 hours.



In the shell script **wrapper.sh** used above, all the processes needed to perform analysis on the Music Data is called once every 3 hours thereby creating a new batch. This is the job scheduling.

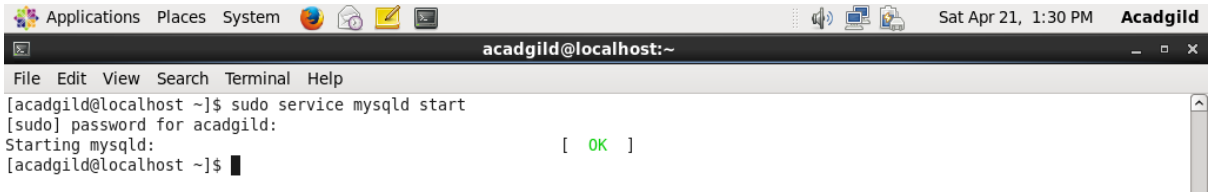


2) Launch all necessary daemons :-

Execution Script - start-daemons.sh

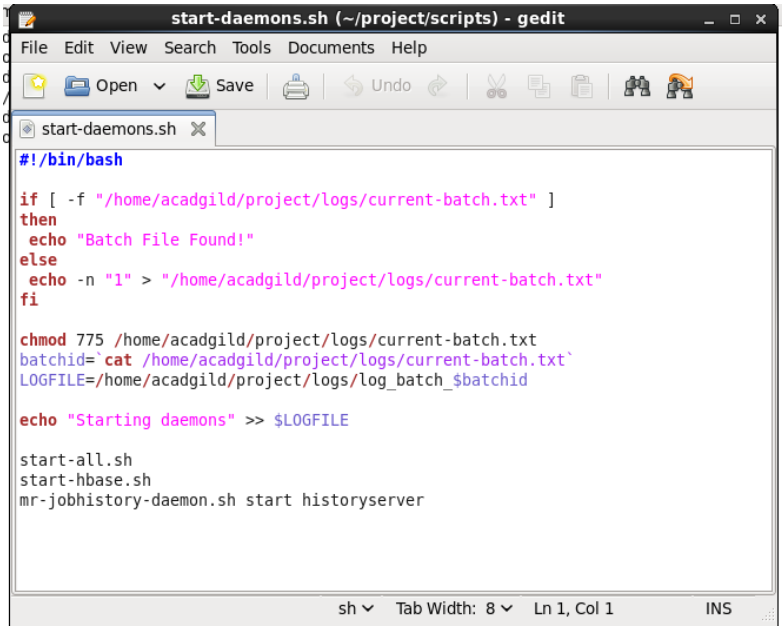
Launch Mysql service

```
sudo service mysqld start
```



Start-daemons.sh is the file that contains command to start all the daemons required for this project.

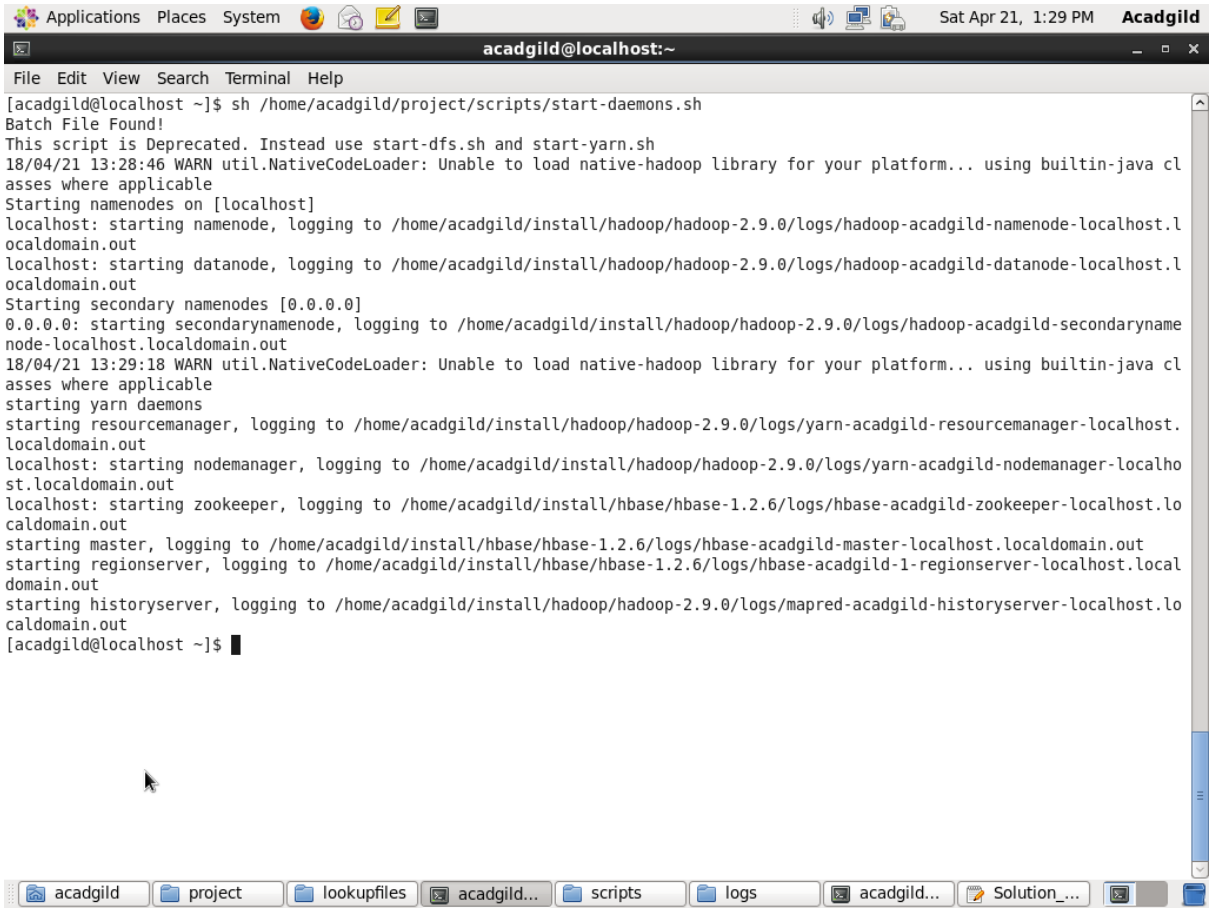
Daemons include - dfs, yarn, hbase and jobhistory daemons.

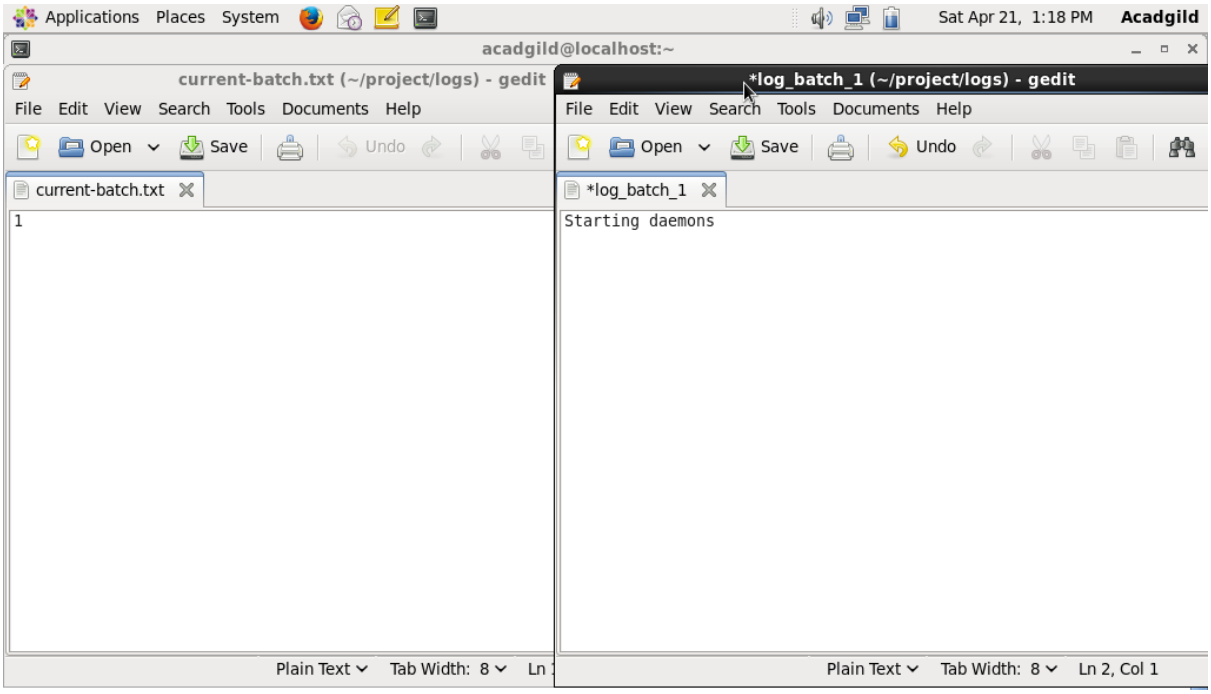


Current-batch.sh is the file which contains batch id.

Chmod 775 provides all the necessary permissions to current-batch.sh file.

Echo command writes the string to the Logfile.





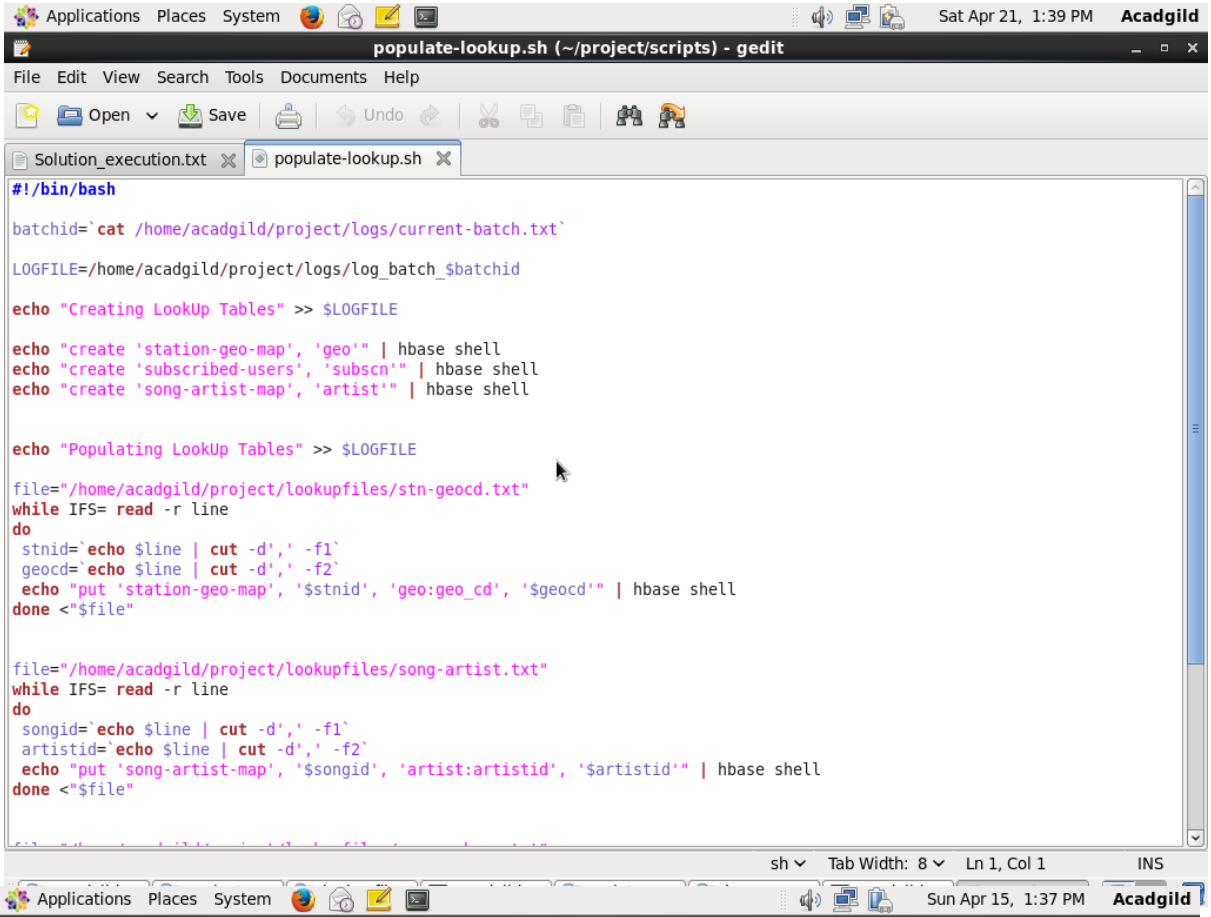
3) Populate look up tables into HBase :-

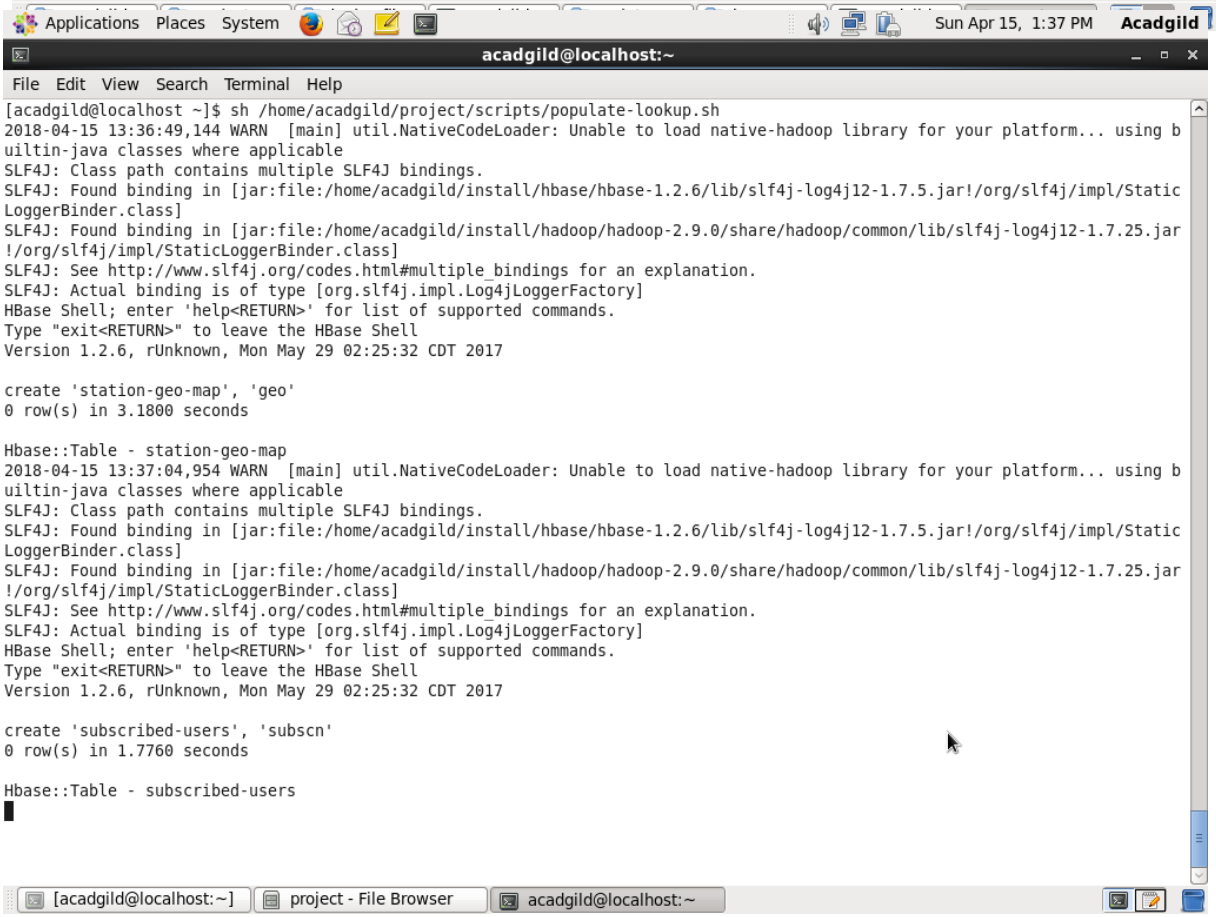
Execution script - populate-lookup.sh

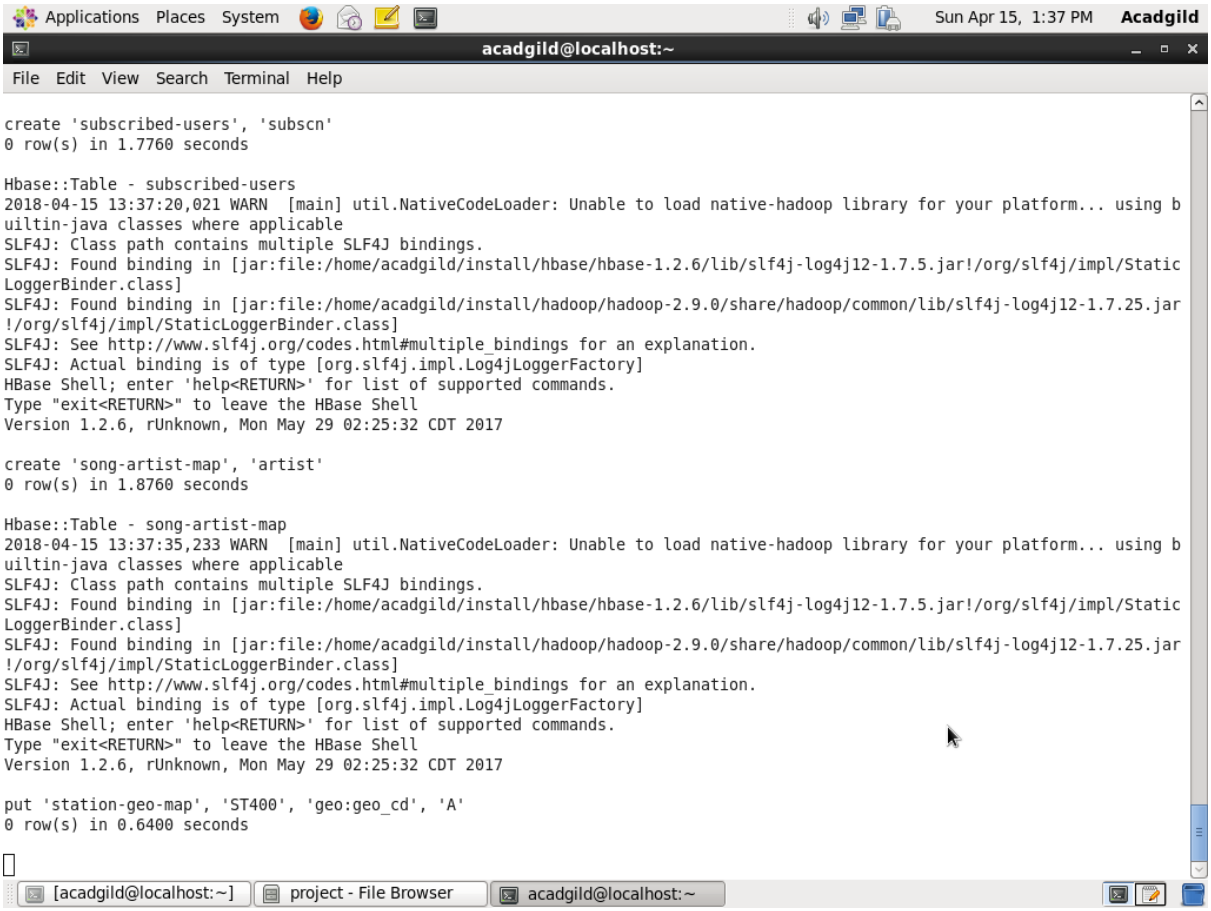
Below is the shell script populate-lookup.sh that is used to load the data for the lookup tables into HBase tables.

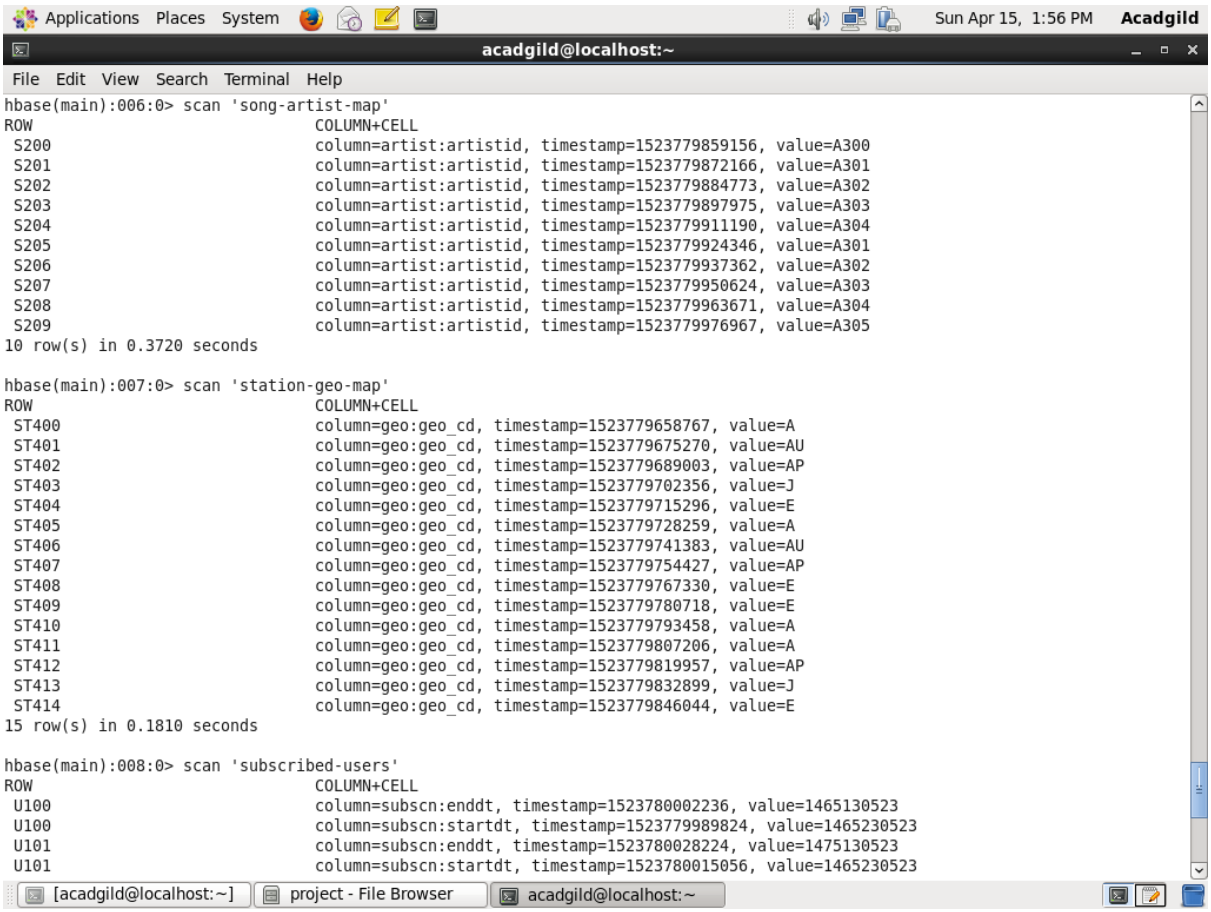
The following operations are performed:

- ✓ Get the batch id number from the batch file and get the Log File for the batch using the batch id.
- ✓ Create the HBase tables for the lookup data files: **song-artist**, **stn-geocd** and **user-subscn** with their column families.
- ✓ For every lookup data file, read each line, extract the columns (comma separated) and add the data as rows to the corresponding HBase tables.
- ✓ Run the hive script **user-artist.hql**. This will populate a hive table with the data in the lookup data file user-artist. This is because this file has an array column that is difficult to populate in HBase.


Applications Places System
Sat Apr 21, 1:39 PM Acadgild
populate-lookup.sh (~/project/scripts) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
Solution_execution.txt
populate-lookup.sh
#!/bin/bash
batchid=`cat /home/acadgild/project/logs/current-batch.txt`
LOGFILE=/home/acadgild/project/logs/log_batch_$batchid
echo "Creating LookUp Tables" >> $LOGFILE
echo "create 'station-geo-map', 'geo'" | hbase shell
echo "create 'subscribed-users', 'subscn'" | hbase shell
echo "create 'song-artist-map', 'artist'" | hbase shell
echo "Populating LookUp Tables" >> $LOGFILE
file="/home/acadgild/project/lookupfiles/stn-geocd.txt"
while IFS= read -r line
do
stnid=`echo $line | cut -d',' -f1`
geocd=`echo $line | cut -d',' -f2`
echo "put 'station-geo-map', '$stnid', 'geo:geo_cd', '$geocd'" | hbase shell
done <"$file"
file="/home/acadgild/project/lookupfiles/song-artist.txt"
while IFS= read -r line
do
songid=`echo $line | cut -d',' -f1`
artistid=`echo $line | cut -d',' -f2`
echo "put 'song-artist-map', '$songid', 'artist:artistid', '$artistid'" | hbase shell
done <"$file"
sh Tab Width: 8 Ln 1, Col 1 INS

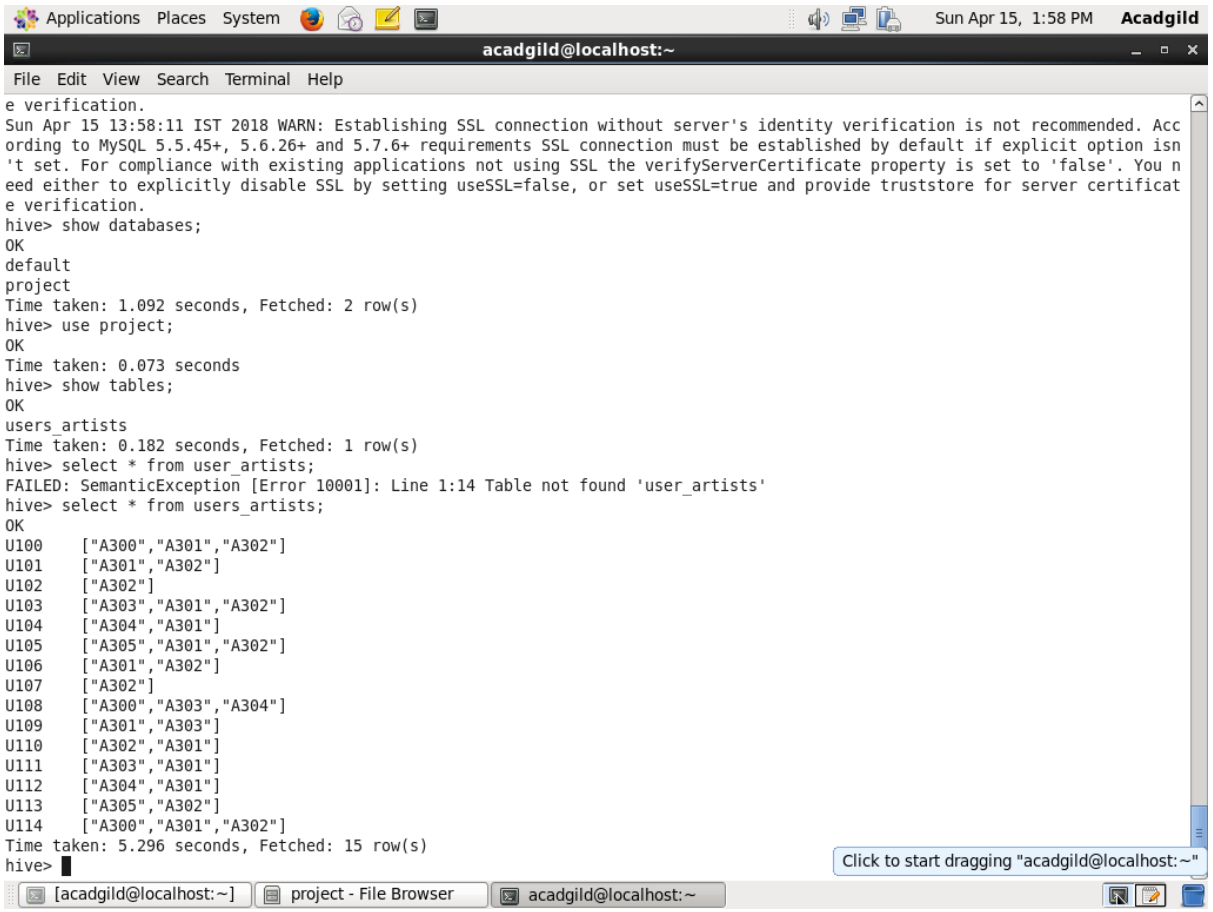

Applications Places System
Sun Apr 15, 1:37 PM Acadgild
acadgild@localhost:~
File Edit View Search Terminal Help
[acadgild@localhost ~]$ sh /home/acadgild/project/scripts/populate-lookup.sh
2018-04-15 13:36:49,144 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... 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-2.9.0/share/hadoop/common/lib/slf4j-log4j12-1.7.25.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 3.1800 seconds
Hbase::Table - station-geo-map
2018-04-15 13:37:04,954 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-2.9.0/share/hadoop/common/lib/slf4j-log4j12-1.7.25.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 'subscribed-users', 'subscn'
0 row(s) in 1.7760 seconds
Hbase::Table - subscribed-users
[acadgild@localhost:~] project - File Browser acadgild@localhost:~


Applications Places System
Sun Apr 15, 1:37 PM Acadgild
acadgild@localhost:~
File Edit View Search Terminal Help
create 'subscribed-users', 'subscn'
0 row(s) in 1.7760 seconds
Hbase::Table - subscribed-users
2018-04-15 13:37:20,021 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using b
uiltin-java classes where applicable
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Static
LoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.9.0/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar
!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017
create 'song-artist-map', 'artist'
0 row(s) in 1.8760 seconds
Hbase::Table - song-artist-map
2018-04-15 13:37:35,233 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using b
uiltin-java classes where applicable
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Static
LoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.9.0/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar
!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017
put 'station-geo-map', 'ST400', 'geo:geo_cd', 'A'
0 row(s) in 0.6400 seconds
[acadgild@localhost:~]
project - File Browser
acadgild@localhost:~

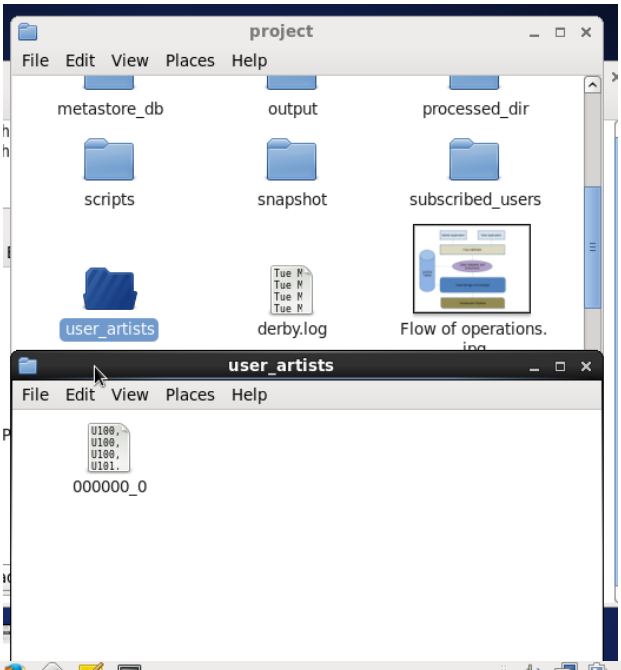

Applications Places System
Sun Apr 15, 1:56 PM Acadgild
acadgild@localhost:~
File Edit View Search Terminal Help
hbase(main):006:0> scan 'song-artist-map'
ROW COLUMN+CELL
S200 column=artist:artistid, timestamp=1523779859156, value=A300
S201 column=artist:artistid, timestamp=1523779872166, value=A301
S202 column=artist:artistid, timestamp=1523779884773, value=A302
S203 column=artist:artistid, timestamp=1523779897975, value=A303
S204 column=artist:artistid, timestamp=1523779911190, value=A304
S205 column=artist:artistid, timestamp=1523779924346, value=A301
S206 column=artist:artistid, timestamp=1523779937362, value=A302
S207 column=artist:artistid, timestamp=1523779950624, value=A303
S208 column=artist:artistid, timestamp=1523779963671, value=A304
S209 column=artist:artistid, timestamp=1523779976967, value=A305
10 row(s) in 0.3720 seconds
hbase(main):007:0> scan 'station-geo-map'
ROW COLUMN+CELL
ST400 column=geo:geo_cd, timestamp=1523779658767, value=A
ST401 column=geo:geo_cd, timestamp=1523779675270, value=AU
ST402 column=geo:geo_cd, timestamp=1523779689003, value=AP
ST403 column=geo:geo_cd, timestamp=1523779702356, value=J
ST404 column=geo:geo_cd, timestamp=1523779715296, value=E
ST405 column=geo:geo_cd, timestamp=1523779728259, value=A
ST406 column=geo:geo_cd, timestamp=1523779741383, value=AU
ST407 column=geo:geo_cd, timestamp=1523779754427, value=AP
ST408 column=geo:geo_cd, timestamp=1523779767330, value=E
ST409 column=geo:geo_cd, timestamp=1523779780718, value=E
ST410 column=geo:geo_cd, timestamp=1523779793458, value=A
ST411 column=geo:geo_cd, timestamp=1523779807206, value=A
ST412 column=geo:geo_cd, timestamp=1523779819957, value=AP
ST413 column=geo:geo_cd, timestamp=1523779832899, value=J
ST414 column=geo:geo_cd, timestamp=1523779846044, value=E
15 row(s) in 0.1810 seconds
hbase(main):008:0> scan 'subscribed-users'
ROW COLUMN+CELL
U100 column=subscn:enddt, timestamp=1523780002236, value=1465130523
U100 column=subscn:startdt, timestamp=1523779989824, value=1465230523
U101 column=subscn:enddt, timestamp=1523780028224, value=1475130523
U101 column=subscn:startdt, timestamp=1523780015056, value=1465230523
[acadgild@localhost:~]
project - File Browser
acadgild@localhost:~

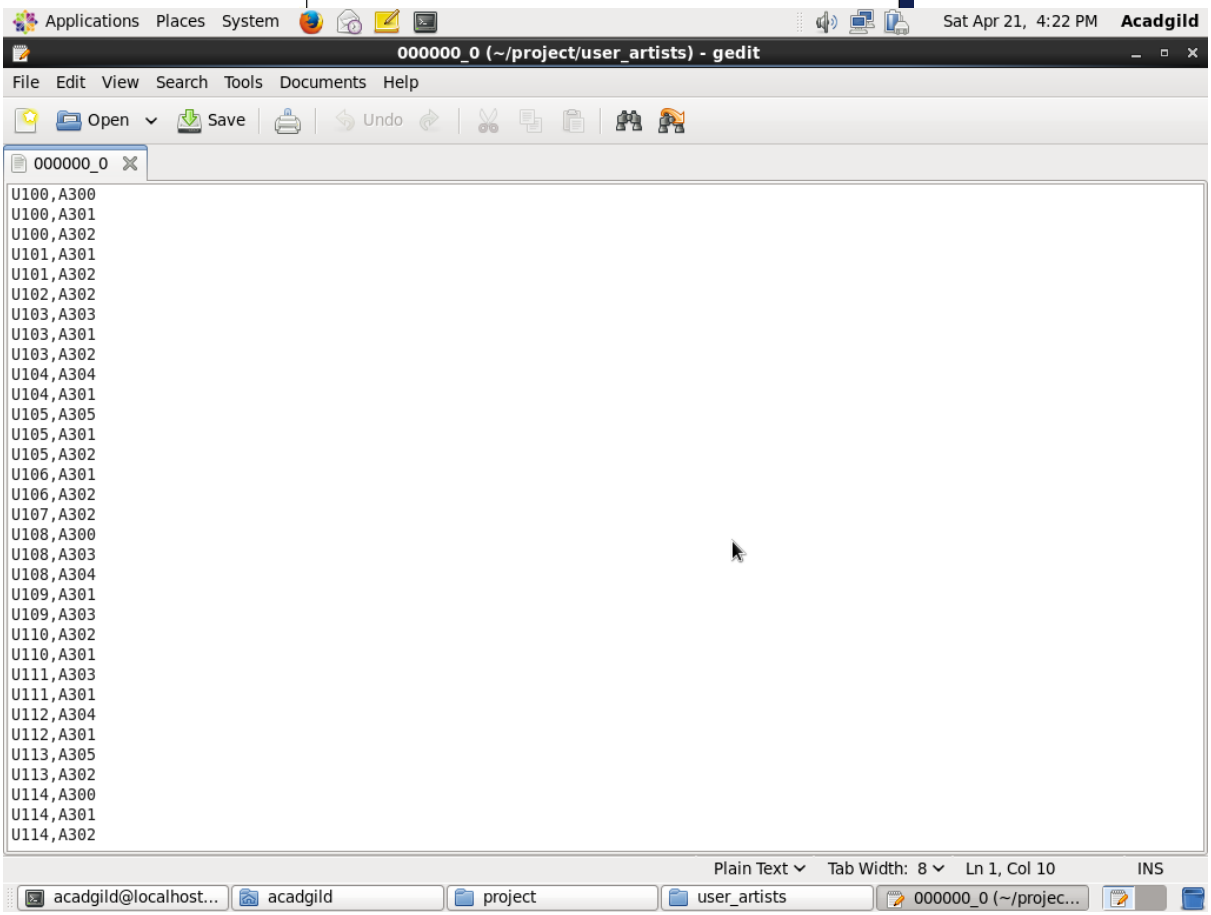
Content of user-artist.hql :

```
CREATE DATABASE IF NOT EXISTS project;
USE project;

CREATE TABLE
(
user_id STRING,
artists_array ARRAY<STRING>
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
COLLECTION ITEMS TERMINATED BY '&';

LOAD DATA LOCAL INPATH '/home/acadgild/project/lookupfiles/user-artist.txt'
OVERWRITE INTO TABLE users_artists;

//A lateral view with explode() can be used to convert artists_array into separate rows using the query

INSERT OVERWRITE LOCAL DIRECTORY '/home/acadgild/project/user_artists/'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
SELECT user_id, artists FROM users_artists LATERAL VIEW explode(artists_array) a AS artists;
```



Output of Hive explode :





4) Perform Data Formatting :-

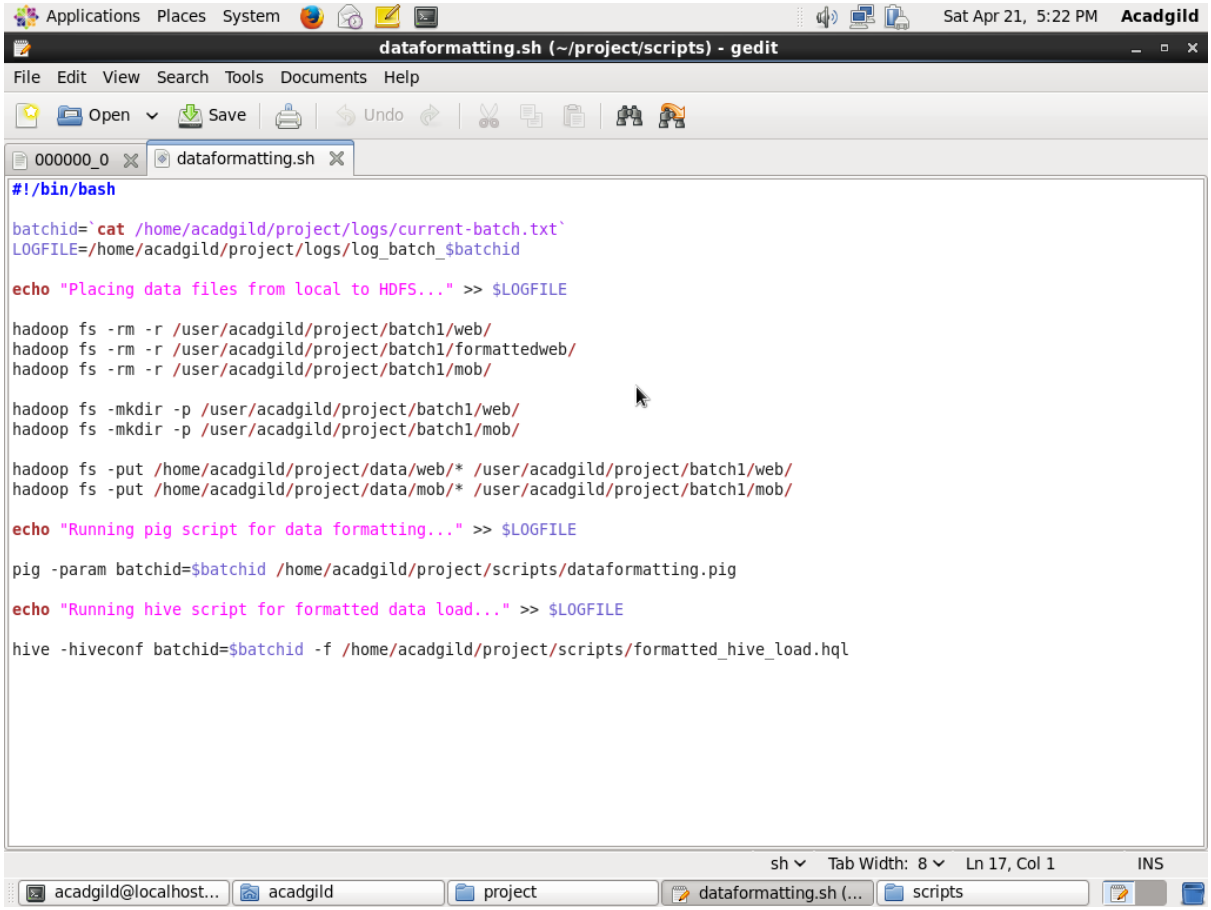
Execution Script - dataformatting.sh

- ✓ Firstly we will be executing **dataformatting.pig** which will format web xml data using pig to CSV.
- ✓ Then Load mob and formatted web data to hive using **formatted_hive_load.hql**

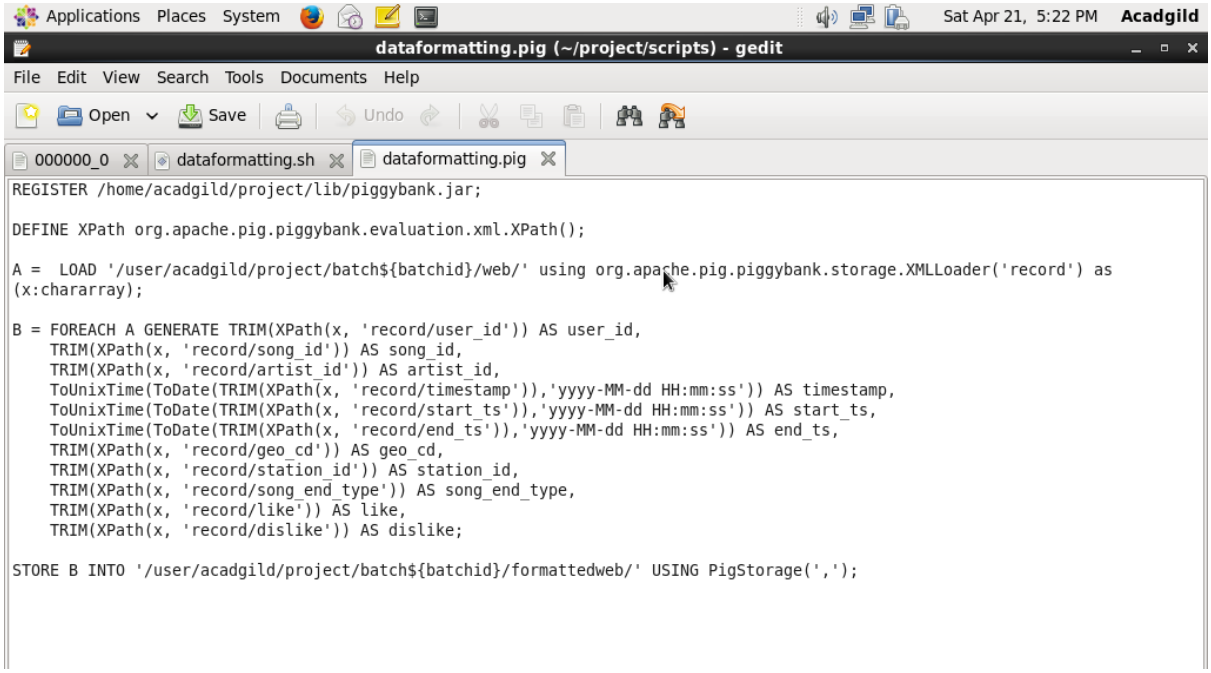
Following operations are performed in **dataformatting.sh** :

- ✓ Fetching batch id from the current_batch file.
- ✓ Writing to the log file about process start.
- ✓ Checking if the directory is present in HDFS if yes, then deleting it and creating new blank directory.
- ✓ Copying mob and formatted web data to HDFS from local.
- ✓ Run pig script **dataformatting.pig**, to format web data stored in HDFS as xml to csv format and store formatted file to formattedweb directory.
- ✓ Run hive script **formatted_hive_load.hql** to load data from mob and formattedweb directory to hive tables.

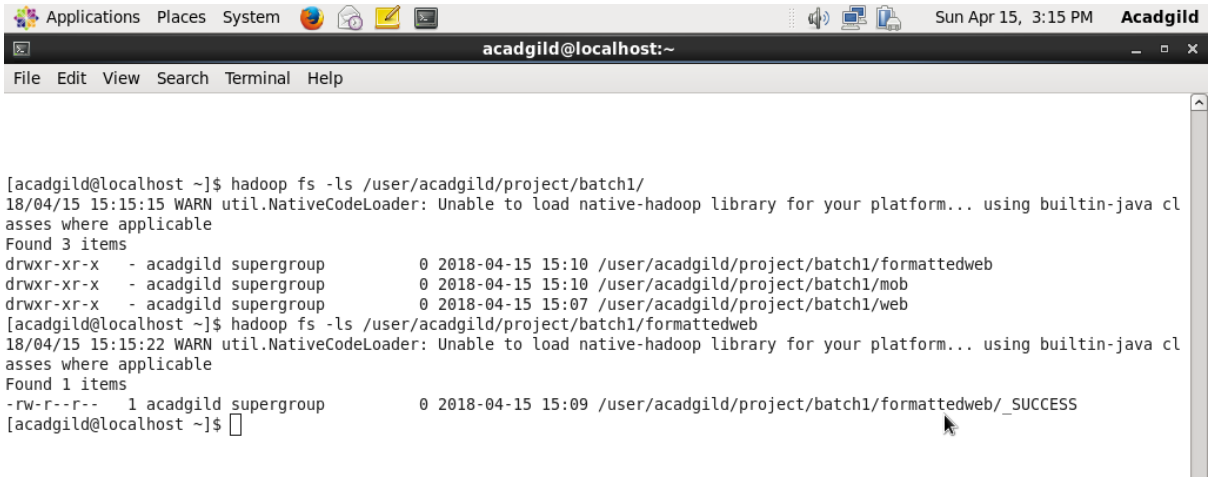
dataformatting.sh



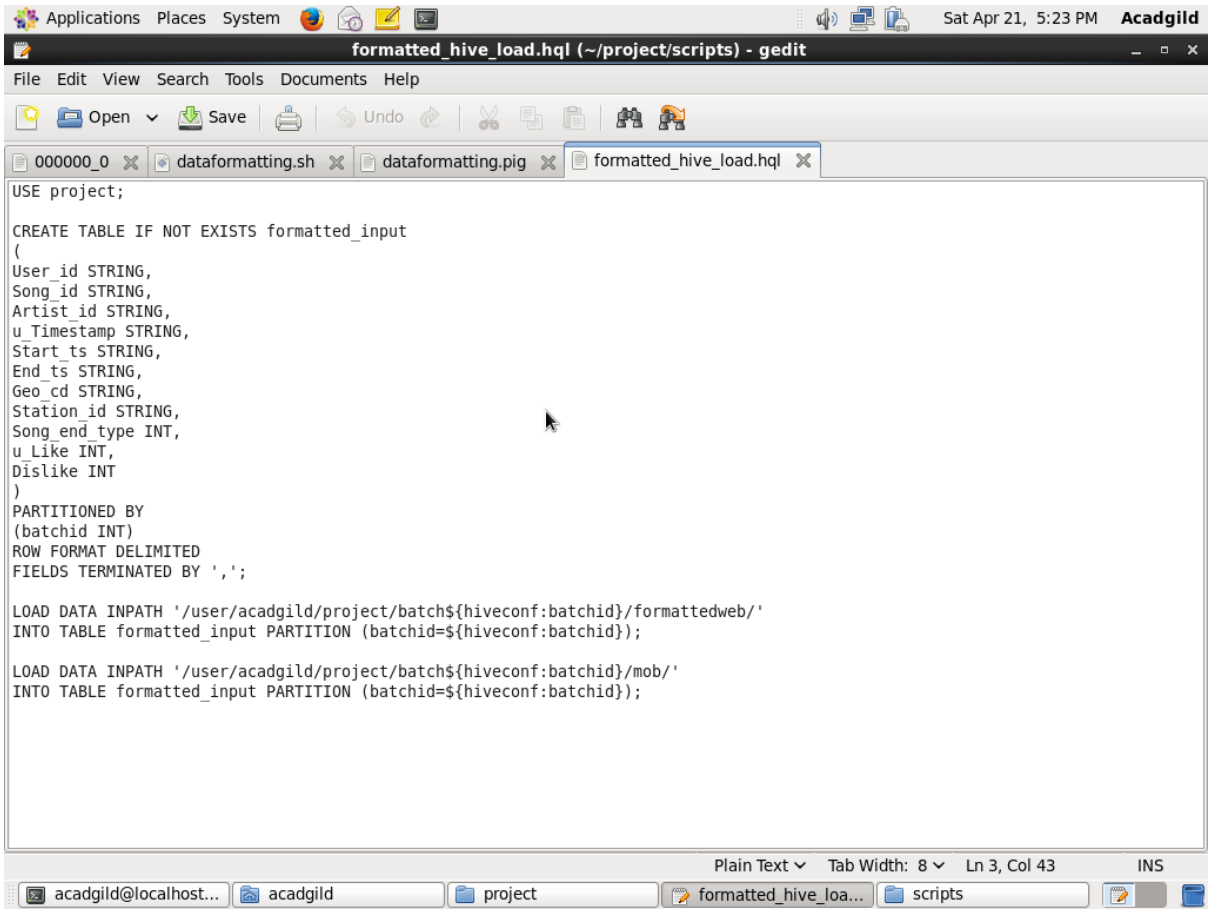
dataformatting.pig

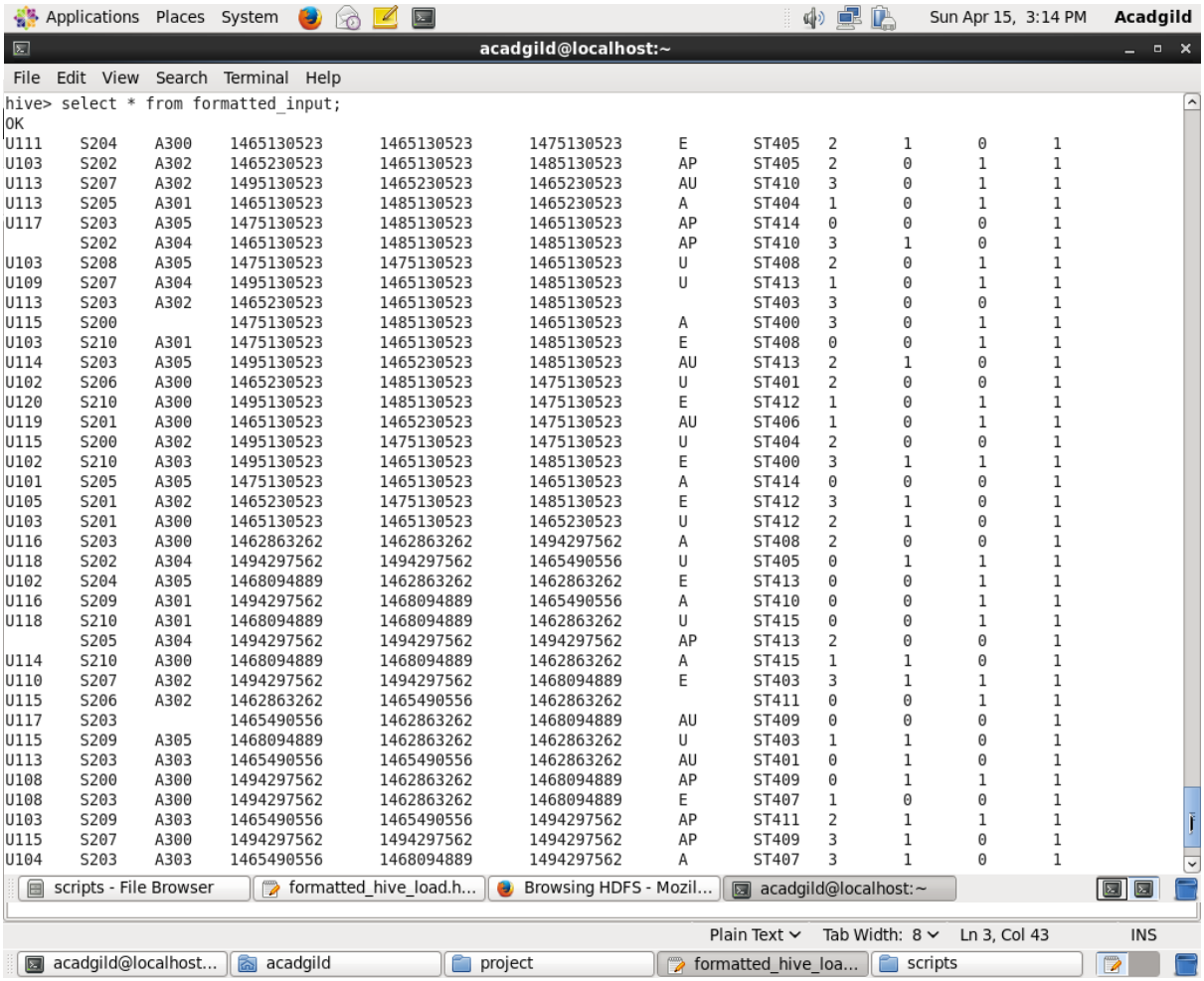


Output :



formatted_hive_load.hql





5) Perform Data Enrichment and Cleaning :-

The data enrichment is carried out in two steps:

- ✓ Create lookup tables in Hive and import the data from the HBase lookup tables to them. This is done by shell script **data_enrichment_filtering_schema.sh**
- ✓ Perform the data enrichment to the data in formatted_input using the lookup tables. This is done by shell script **data_enrichment.sh**

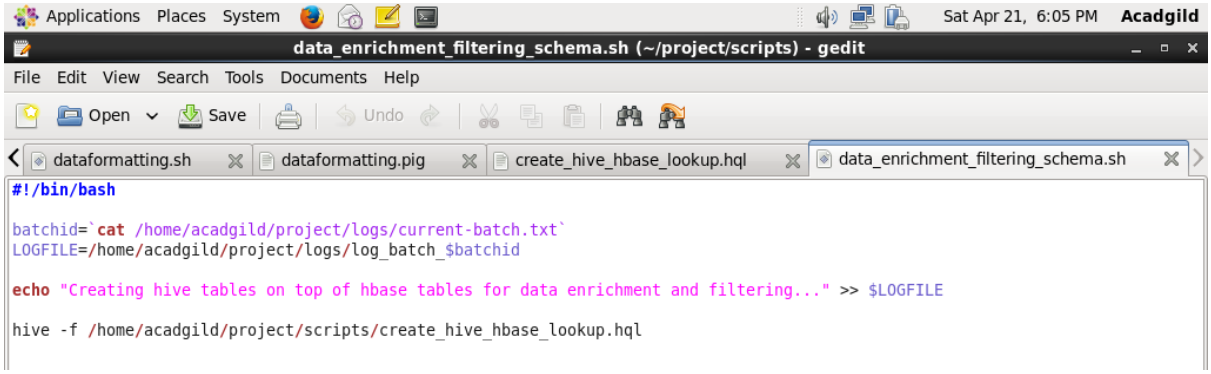
data_enrichment_filtering_schema.sh

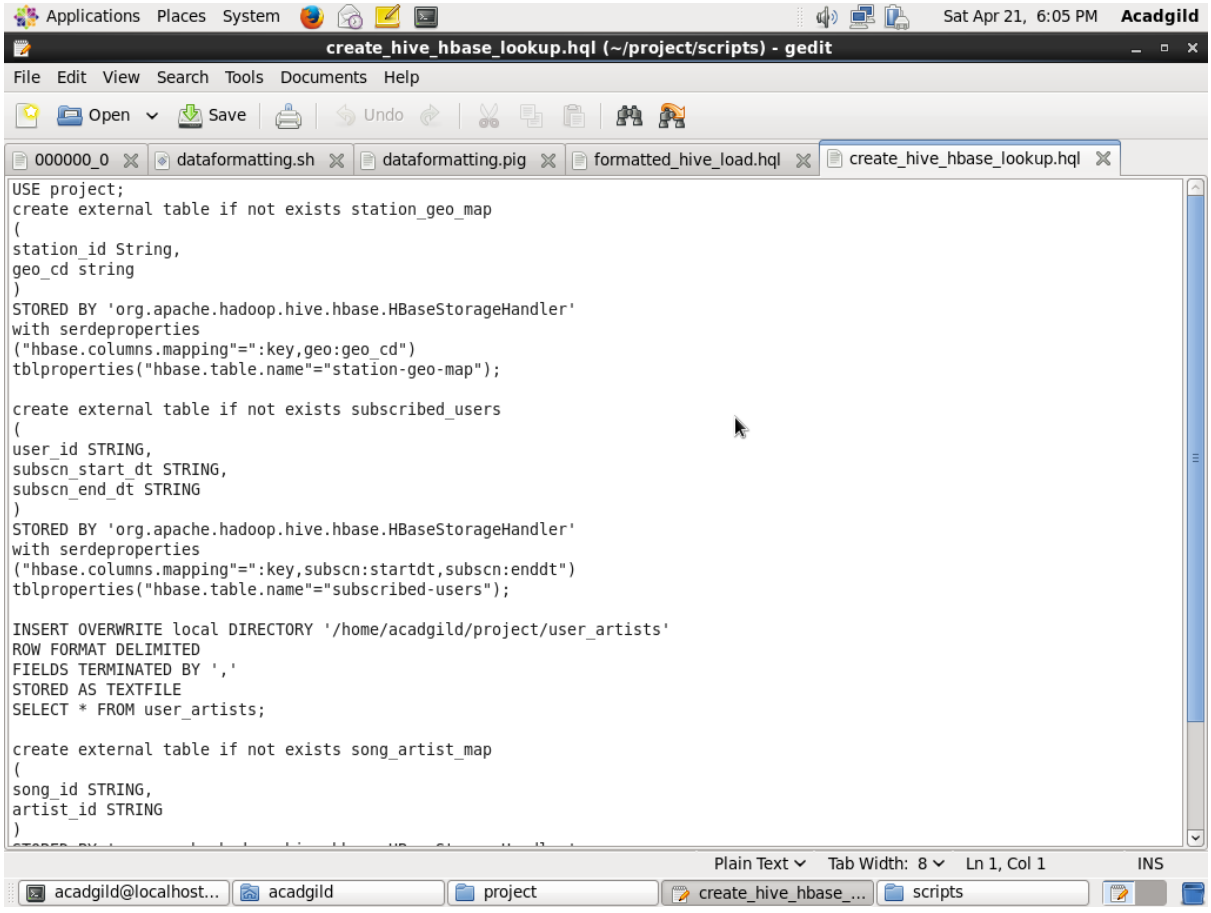
Below is the shell script **data_enrichment_filtering_schema.sh** where the following operations are performed:

Get the batch id number from the batch file and get the Log File for the batch using the batch id.

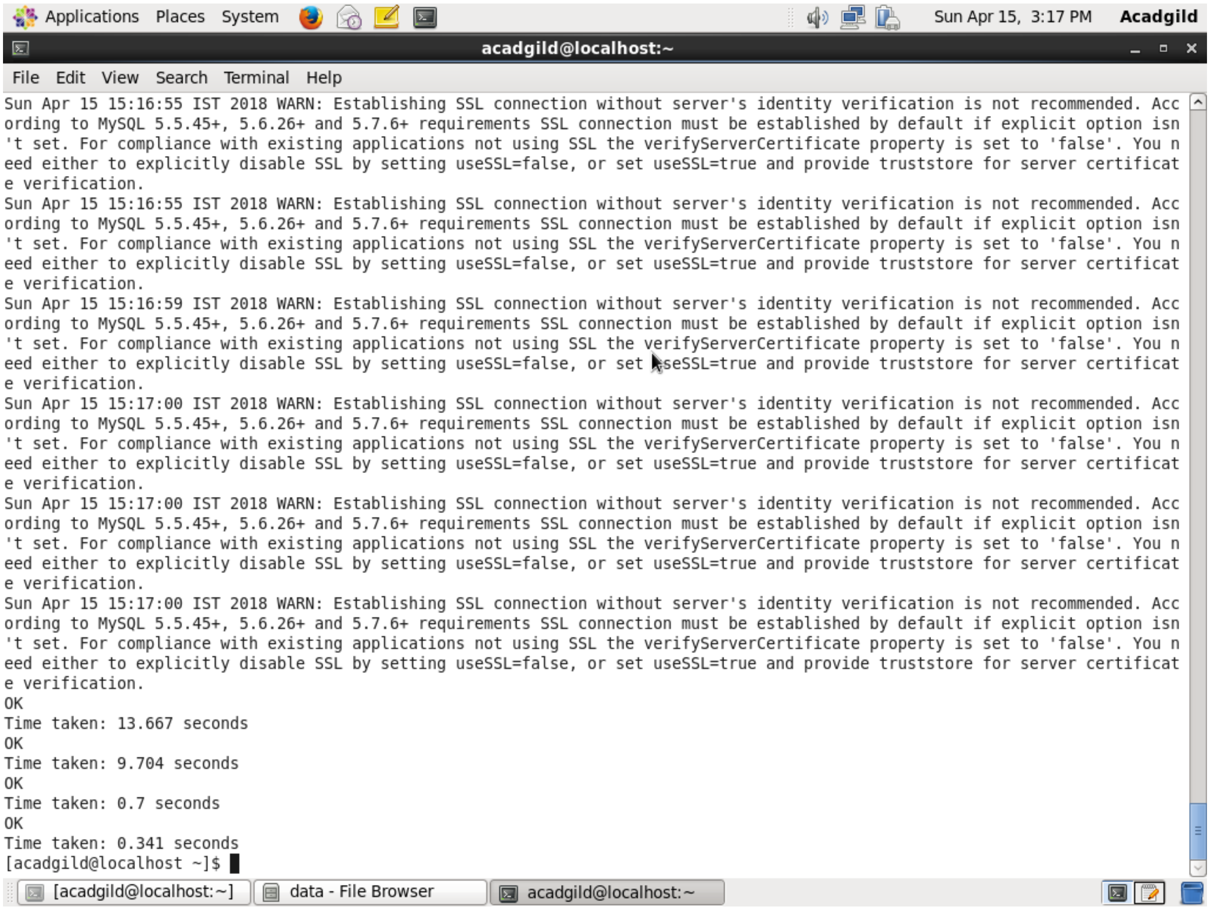
Add logs to the Log File signifying that the Hive lookup tables are created from the HBase lookup tables.

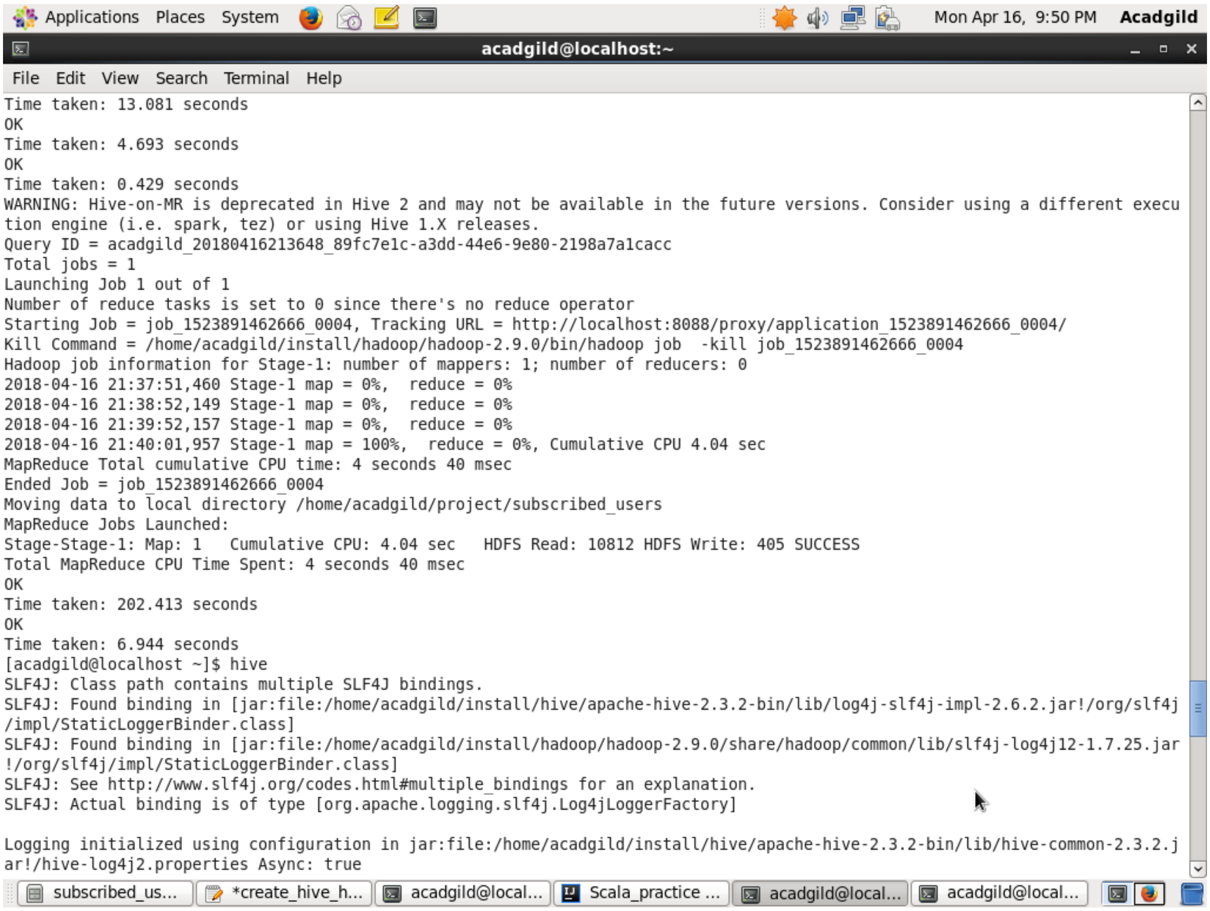
Run the hive script **create_hive_hbase_lookup.hql**. This will create the lookup tables in Hive and import the data from the HBase lookup tables to the Hive lookup tables.

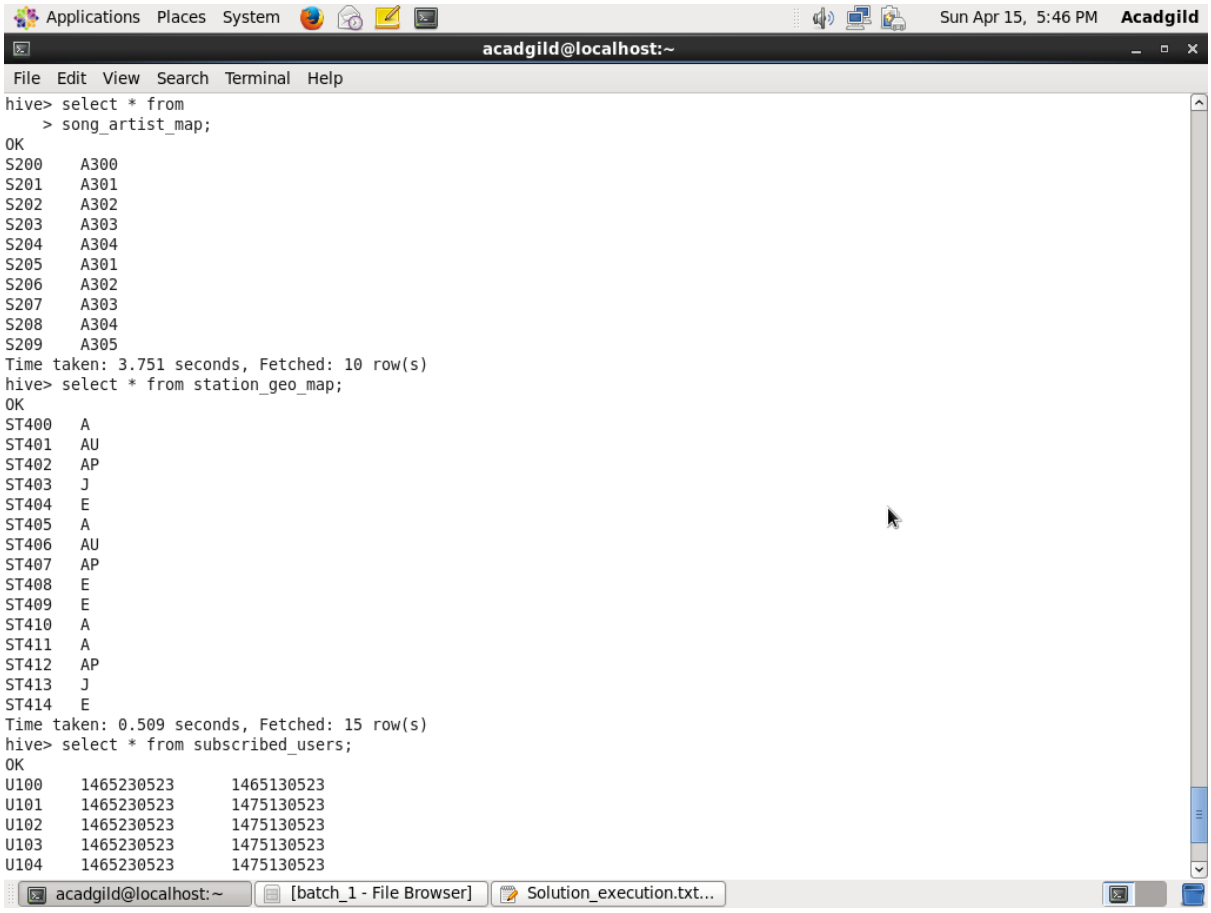


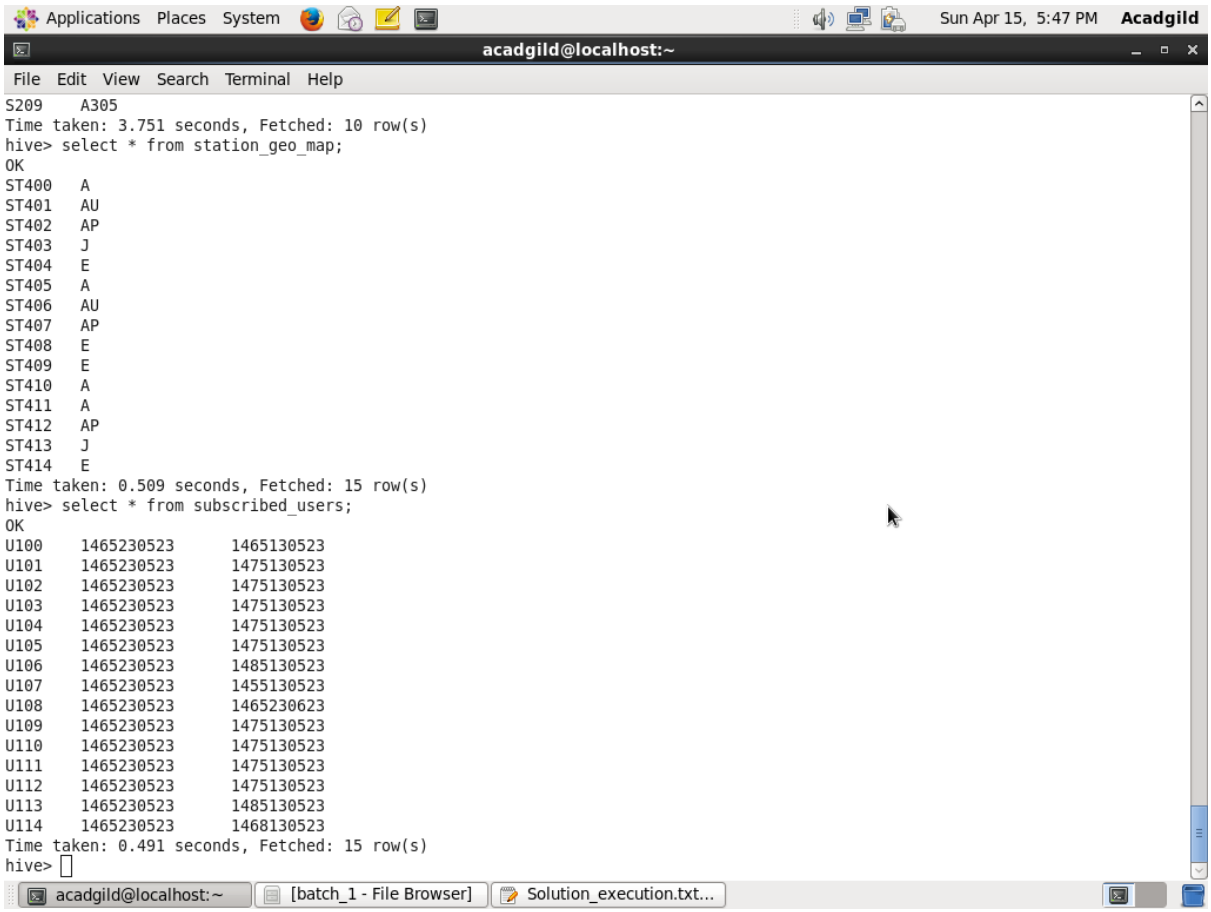


Output :


Applications Places System
Sun Apr 15, 3:17 PM Acadgild
acadgild@localhost:~
File Edit View Search Terminal Help
Sun Apr 15 15:16:55 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. Acc
ording to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn
't set. For compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You n
eed either to explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificat
e verification.
Sun Apr 15 15:16:55 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. Acc
ording to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn
't set. For compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You n
eed either to explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificat
e verification.
Sun Apr 15 15:16:59 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. Acc
ording to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn
't set. For compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You n
eed either to explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificat
e verification.
Sun Apr 15 15:17:00 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. Acc
ording to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn
't set. For compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You n
eed either to explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificat
e verification.
Sun Apr 15 15:17:00 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. Acc
ording to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn
't set. For compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You n
eed either to explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificat
e verification.
Sun Apr 15 15:17:00 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. Acc
ording to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn
't set. For compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You n
eed either to explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificat
e verification.
OK
Time taken: 13.667 seconds
OK
Time taken: 9.704 seconds
OK
Time taken: 0.7 seconds
OK
Time taken: 0.341 seconds
[acadgild@localhost ~]$
[acadgild@localhost:~]
data - File Browser
acadgild@localhost:~


Applications Places System
Mon Apr 16, 9:50 PM Acadgild
acadgild@localhost:~
File Edit View Search Terminal Help
Time taken: 13.081 seconds
OK
Time taken: 4.693 seconds
OK
Time taken: 0.429 seconds
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180416213648_89fc7e1c-a3dd-44e6-9e80-2198a7a1cacc
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1523891462666_0004, Tracking URL = http://localhost:8088/proxy/application_1523891462666_0004/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.9.0/bin/hadoop job -kill job_1523891462666_0004
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2018-04-16 21:37:51,460 Stage-1 map = 0%, reduce = 0%
2018-04-16 21:38:52,149 Stage-1 map = 0%, reduce = 0%
2018-04-16 21:39:52,157 Stage-1 map = 0%, reduce = 0%
2018-04-16 21:40:01,957 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.04 sec
MapReduce Total cumulative CPU time: 4 seconds 40 msec
Ended Job = job_1523891462666_0004
Moving data to local directory /home/acadgild/project/subscribed_users
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 4.04 sec HDFS Read: 10812 HDFS Write: 405 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 40 msec
OK
Time taken: 202.413 seconds
OK
Time taken: 6.944 seconds
[acadgild@localhost ~]$ hive
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-2.9.0/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar
!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Logging initialized using configuration in jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hive-common-2.3.2.j
ar!/hive-log4j2.properties Async: true
subscribed_us...
*create_hive_h...
acadgild@local...
Scala_practice ...
acadgild@local...
acadgild@local...


Applications Places System
Sun Apr 15, 5:46 PM Acadgild
acadgild@localhost:~
File Edit View Search Terminal Help
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: 3.751 seconds, Fetched: 10 row(s)
hive> select * from station_geo_map;
OK
ST400 A
ST401 AU
ST402 AP
ST403 J
ST404 E
ST405 A
ST406 AU
ST407 AP
ST408 E
ST409 E
ST410 A
ST411 A
ST412 AP
ST413 J
ST414 E
Time taken: 0.509 seconds, Fetched: 15 row(s)
hive> select * from subscribed_users;
OK
U100 1465230523 1465130523
U101 1465230523 1475130523
U102 1465230523 1475130523
U103 1465230523 1475130523
U104 1465230523 1475130523
acadgild@localhost:~
[batch_1 - File Browser]
Solution_execution.txt...


Applications Places System
Sun Apr 15, 5:47 PM Acadgild
acadgild@localhost:~
File Edit View Search Terminal Help
S209 A305
Time taken: 3.751 seconds, Fetched: 10 row(s)
hive> select * from station_geo_map;
OK
ST400 A
ST401 AU
ST402 AP
ST403 J
ST404 E
ST405 A
ST406 AU
ST407 AP
ST408 E
ST409 E
ST410 A
ST411 A
ST412 AP
ST413 J
ST414 E
Time taken: 0.509 seconds, Fetched: 15 row(s)
hive> select * from subscribed_users;
OK
U100 1465230523 1465130523
U101 1465230523 1475130523
U102 1465230523 1475130523
U103 1465230523 1475130523
U104 1465230523 1475130523
U105 1465230523 1475130523
U106 1465230523 1485130523
U107 1465230523 1455130523
U108 1465230523 1465230623
U109 1465230523 1475130523
U110 1465230523 1475130523
U111 1465230523 1475130523
U112 1465230523 1475130523
U113 1465230523 1485130523
U114 1465230523 1468130523
Time taken: 0.491 seconds, Fetched: 15 row(s)
hive>
acadgild@localhost:~
[batch_1 - File Browser]
Solution_execution.txt...

data_enrichment.sh

The following operations are performed:

- ✓ Get the batch id number from the batch file and get the Log File for the batch using the batch id.
- ✓ Add logs to the Log File signifying that the data enrichment has begun.
- ✓ Run the hive script **data_enrichment.hql**. This will create a Hive table enriched_data that will hold the data that is partitioned based on given rules as pass or fail (status) and batchid.
- ✓ Copy the data from the pass and fail folders (valid & invalid) in the Hive warehouse to the Local FS.

data_enrichment.hql :

Data Enrichment

Rules for data enrichment

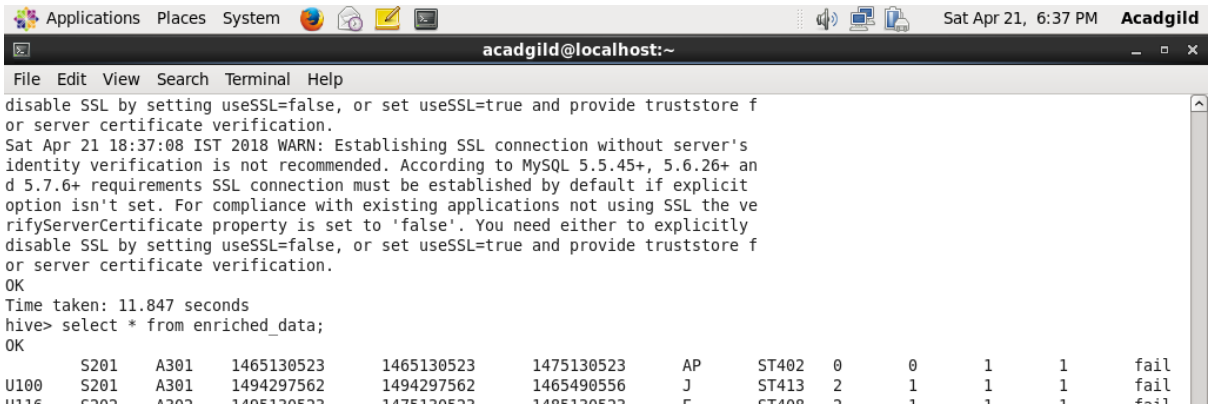
1. If any of *like* or *dislike* is **NULL** or *absent*, consider it as 0.
2. If fields like *Geo_cd* and *Artist_id* are **NULL** or *absent*, consult the lookup tables for fields *Station_id* and *Song_id* respectively to get the values of *Geo_cd* and *Artist_id*.
3. If corresponding lookup entry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

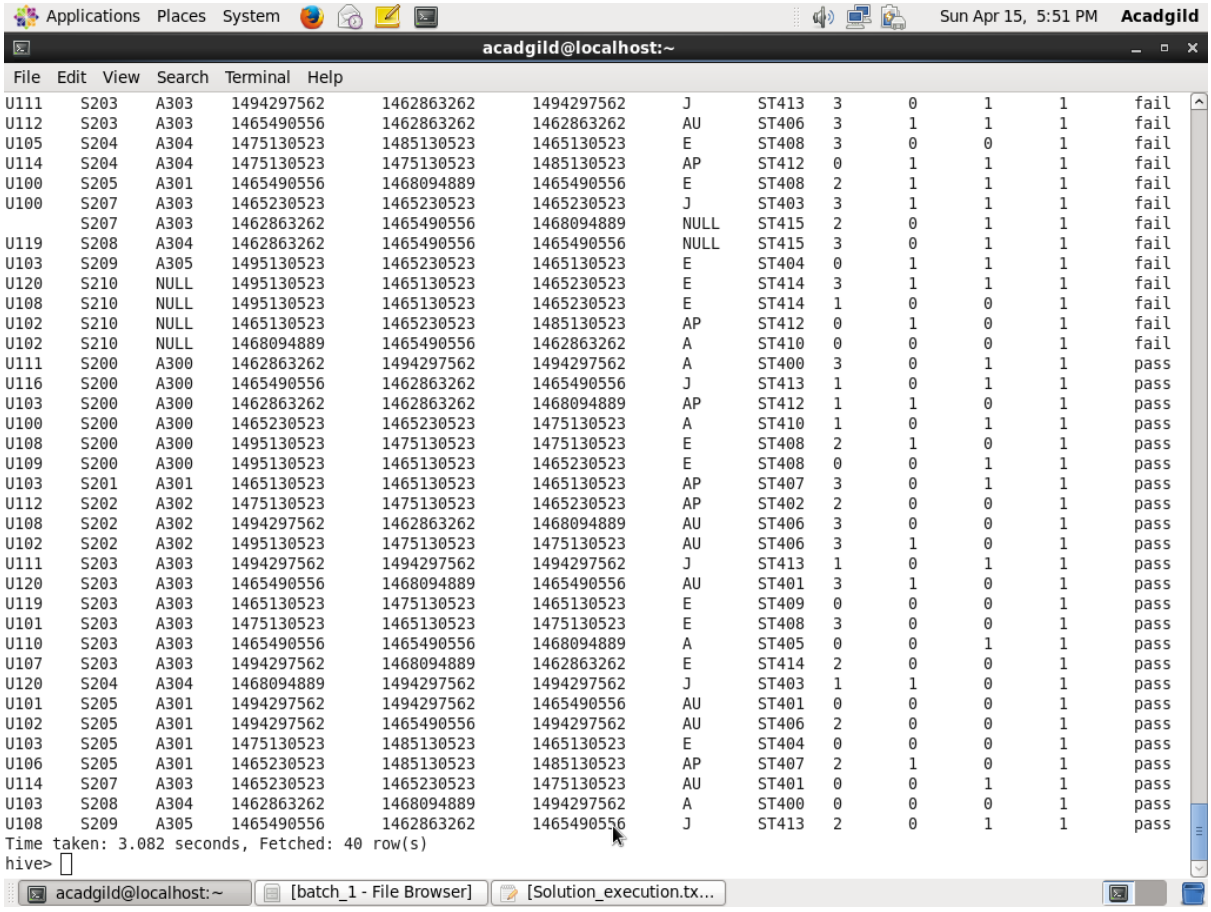
NULL or absent field	Look up field	Look up table (Table from which record can be updated)
Geo_cd	Station_id	Station_Geo_Map
Artist_id	Song_id	Song_Artist_Map

```
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,
u_Timestamp STRING,
Start_ts STRING,
End_ts STRING,
Geo_cd STRING,
Station_id STRING,
Song_end_type INT,
u_Like INT,
Dislike INT
)
PARTITIONED BY
(batchid INT,
status STRING)
STORED AS ORC;
```

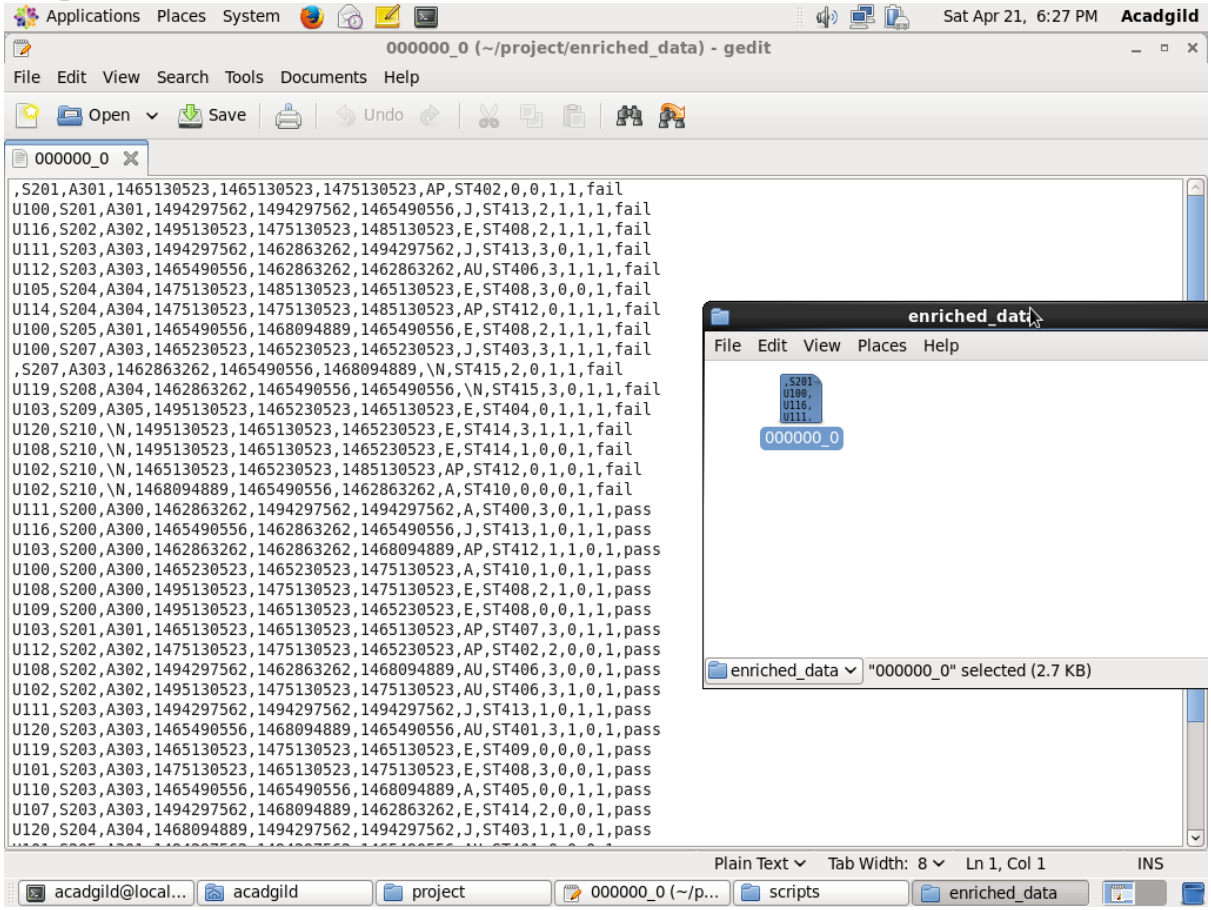
```
INSERT OVERWRITE TABLE enriched_data
PARTITION (batchid, status)
SELECT
i.user_id,
i.song_id,
sa.artist_id,
i.u_timestamp,
i.start_ts,
i.end_ts,
sg.geo_cd,
i.station_id,
IF (i.song_end_type IS NULL, 3, i.song_end_type) AS song_end_type,
IF (i.u_like IS NULL, 0, i.u_like) AS u_like,
IF (i.dislike IS NULL, 0, i.dislike) AS dislike,
i.batchid,
IF((i.u_like=1 AND i.dislike=1)
OR i.user_id IS NULL
OR i.song_id IS NULL
OR i.u_timestamp IS NULL
OR i.start_ts IS NULL
OR i.end_ts IS NULL
OR i.geo_cd IS NULL
OR i.user_id=''
OR i.song_id=''
OR i.u_timestamp=''
OR i.start_ts=''
OR i.end_ts=''
OR i.geo_cd=''
OR sg.geo_cd IS NULL
OR sg.geo_cd=''
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};

INSERT OVERWRITE local DIRECTORY '/home/acadgild/project/enriched_data'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE

SELECT * FROM enriched_data;
```

Output in hive :





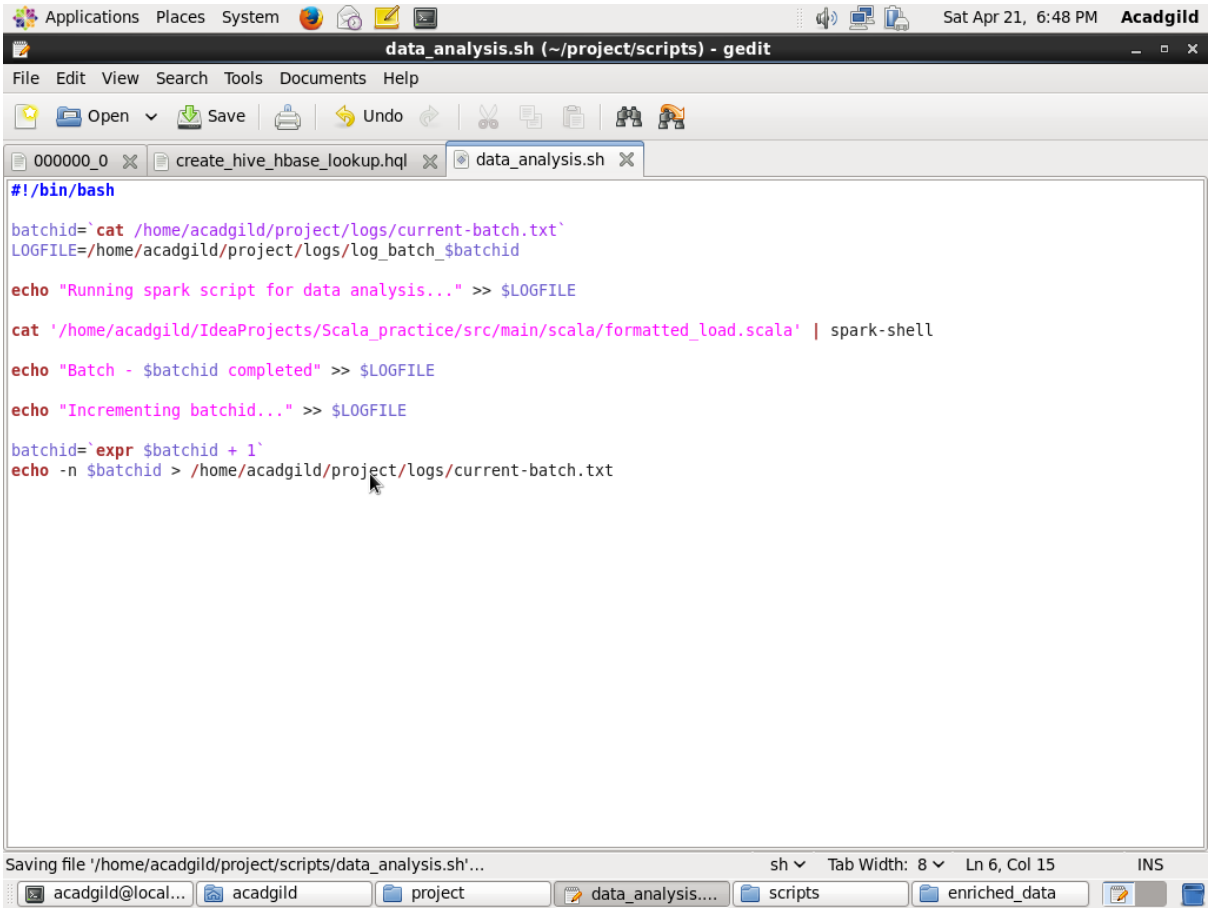
Output in local :



6) Perform Data Analysis :-

Below is the shell script data_analysis.sh where the following operations are performed:

- ✓ Get the batch id number from the batch file and get the Log File for the batch using the batch id.
- ✓ Add logs to the Log File signifying that the data analysis is being performed using Spark and that the result is being exported to the Local FS.
- ✓ Run the spark script **formatted_load.scala**. This will perform the data analysis required in the problem statement given and save the result to the Local FS.
- ✓ Add logs to the Log File signifying that the data analysis has completed and that the batch is being incremented. Here from 1 to 2.



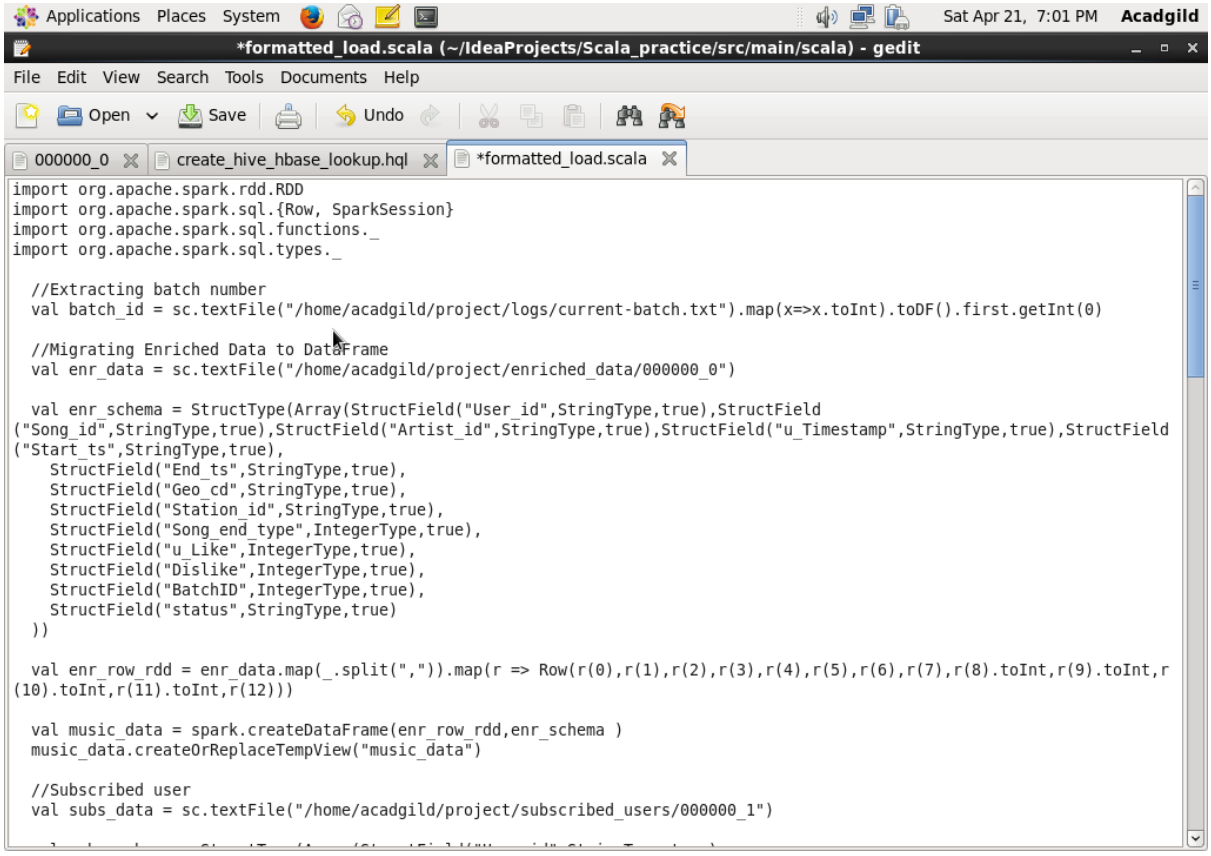
In the file **formatted_load.scala** that will perform data analysis and store each result into local drive by creating a new directory with every new batch id.

Steps :

- ✓ Importing row, Sparksession, RDD, functions and types which are need for analysis.
- ✓ Then, we will retrive the batch id from current_batch file.
- ✓ We will now import data stored in local FS i.e. enriched_data, subscribed_user and user_artists and perform the following operations for each :
  - Create the schema for the data
  - Create dataframe
  - Create temperory table

Now we are ready for analysis.

formatted_load.scala



```
   prob_1.write.format("csv").option("header","true").save(s"/home/acadgild/project/output/top_10_station/batch_$batch_id")

  //*************************************************************************
  //                            PROBLEM 2
  //*************************************************************************

  val prob_2 = spark.sql("select CASE WHEN ( subusr.user_id is null or " +
    "cast(music.u_timestamp as decimal(20,0)) > cast(subusr.end_dt as decimal(20,0)) " +
    ") then 'UNSUBSCRIBED' WHEN (subusr.user_id is NOT null or " +
    "cast(music.u_timestamp as decimal(20,0)) <= cast(subusr.end_dt as decimal(20,0)) " +
    ") then 'SUBSCRIBED' END AS USER_TYPE, " +
    "SUM(ABS(CAST(music.end_ts as decimal(20,0)) - CAST(music.start_ts as decimal(20,0))))" +
    " as duration ,batchid" +
    " from music_data music left outer join " +
    "subscribed_user subusr on music.user_id = subusr.user_id " +
    s"where music.status = 'pass' and music.batchid = $batch_id " +
    s"group by music.batchid, CASE WHEN ( subusr.user_id is null or " +
    "cast(music.u_timestamp as decimal(20,0)) > cast(subusr.end_dt as decimal(20,0)) " +
    ") then 'UNSUBSCRIBED' WHEN (subusr.user_id is NOT null or " +
    "cast(music.u_timestamp as decimal(20,0)) <= cast(subusr.end_dt as decimal(20,0)) " +
    ") then 'SUBSCRIBED' END " ).toDF()

  prob_2.repartition(1).write.format("csv").option("header","true").save(s"/home/acadgild/project/output/user_response/
batch_$batch_id")

  //*************************************************************************
  //                            PROBLEM 3
  //*************************************************************************
```

```
  val prob_3 = spark.sql("select ua.artists, count(distinct ua.user_id) as count_of_users, " +
    "md.batchid from users_artists ua inner join (select artist_id,song_id,user_id,batchid" +
    s" from music_data where status = 'pass' and batchid = $batch_id)md " +
    s" on ua.artists = md.artist_id and ua.user_id = md.user_id group by ua.artists," +
    s" batchid order by count_of_users DESC limit 10").toDF("user_id","count_of_users","batchid")

    prob_3.write.format("csv").option("header","true").save(s"/home/acadgild/project/output/connected_artist/batch_
$batch_id")

  //*************************************************************************
  //                            PROBLEM 4
  //*************************************************************************

  val prob_4 = spark.sql("select song_id, " +
    "SUM(ABS(CAST(end_ts as decimal(20,0)) - " +
    " CAST(start_ts as decimal(20,0)))) as duration,batchid" +
    s" from music_data where status = 'pass' and batchid = $batch_id " +
    "and (u_like= 1 or song_end_type = 0) group by song_id, batchid " +
    "order by duration desc limit 10").toDF("song_id","duration","batch_id")

  prob_4.write.format("csv").option("header","true").save(s"/home/acadgild/project/output/royalty_songs/batch_$batch_id")
```

```
//*************************************************************************
//                            PROBLEM 5
//*************************************************************************

val prob_5 = spark.sql("select md.user_id, " +
  "SUM(ABS(CAST(end_ts as decimal(20,0)) - " +
  " CAST(start_ts as decimal(20,0)))) as duration,batchid" +
  s" from music_data md LEFT OUTER JOIN subscribed_user su " +
  s"on md.user_id = su.user_id where md.status = 'pass' and md.batchid = $batch_id " +
  "and (md.user_id IS NULL or (CAST(md.u_timestamp as DECIMAL(20,0)) > " +
  " cast(su.end_dt as DECIMAL(20,0)))" +
  " group by md.user_id, batchid " +
  "order by duration desc limit 10").toDF()

prob_5.write.format("csv").option("header","true").save(s"/home/acadgild/project/output/unsubscribed_users/batch_$batch_id")
```

Problem Statement :-

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.

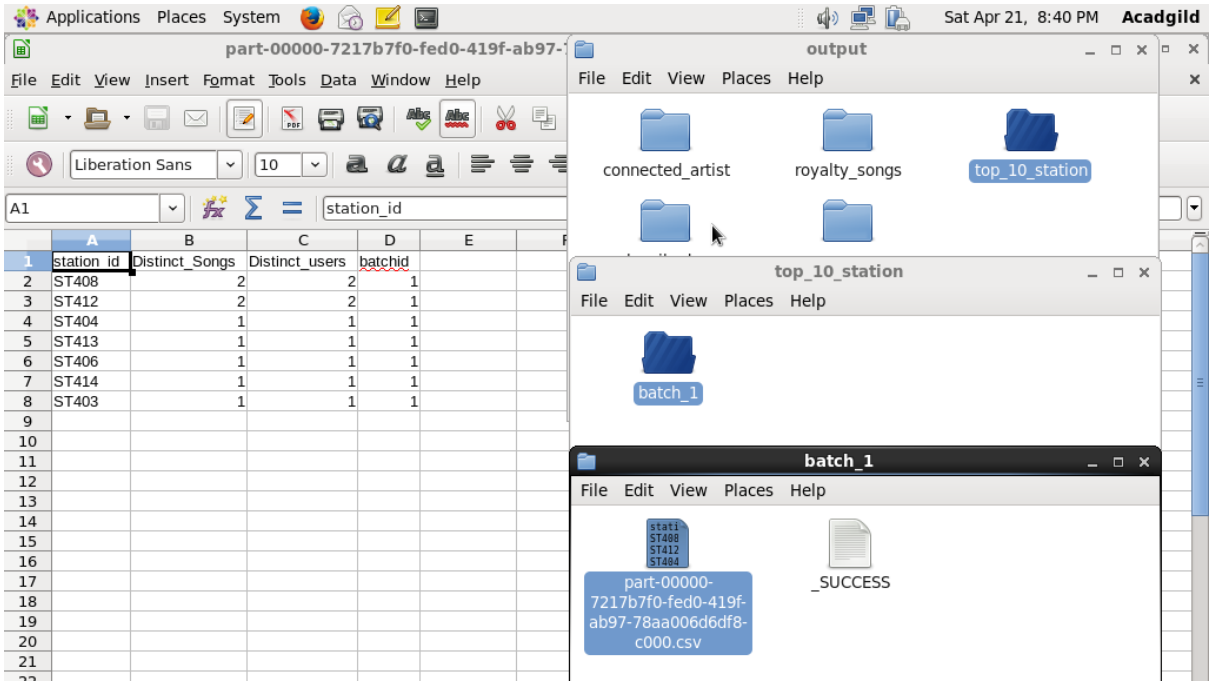
4. Determine top 10 songs who have generated the maximum revenue. Royalty applies to a song only if it was liked or was completed successfully or both.

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

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

Solution :

```
//*******************************************************************************
//                                PROBLEM 1
//*******************************************************************************

val prob_1  = spark.sql("select station_id, count(distinct song_id) Distinct_Songs," +
  " count(distinct user_id) Distinct_users, batchid from music_data " +
  s"where status='fail' and batchid = $batch_id " +
  s"and u_like=1 group by station_id,batchid order by Distinct_Songs " +
  s"DESC limit 10").toDF("station_id","Distinct_Songs","Distinct_users","batchid")

  prob_1.write.format("csv").option("header","true").save(s"/home/acadgild/project/output/top_10_station/batch_$batch_id")
```

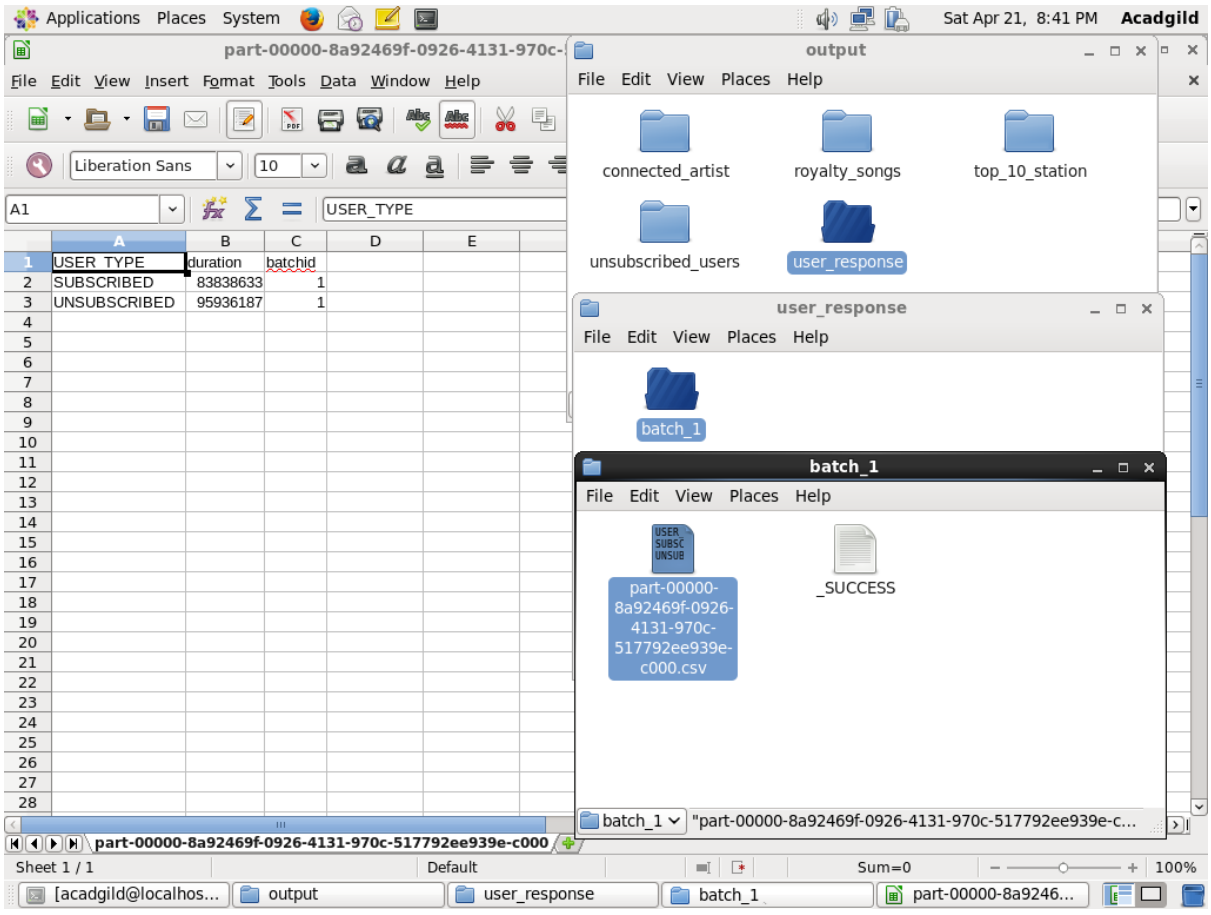


	A	B	C	D	E
1	station_id	Distinct_Songs	Distinct_users	batchid	
2	ST408	2	2	1	
3	ST412	2	2	1	
4	ST404	1	1	1	
5	ST413	1	1	1	
6	ST406	1	1	1	
7	ST414	1	1	1	
8	ST403	1	1	1	

Problem 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.

Solution :

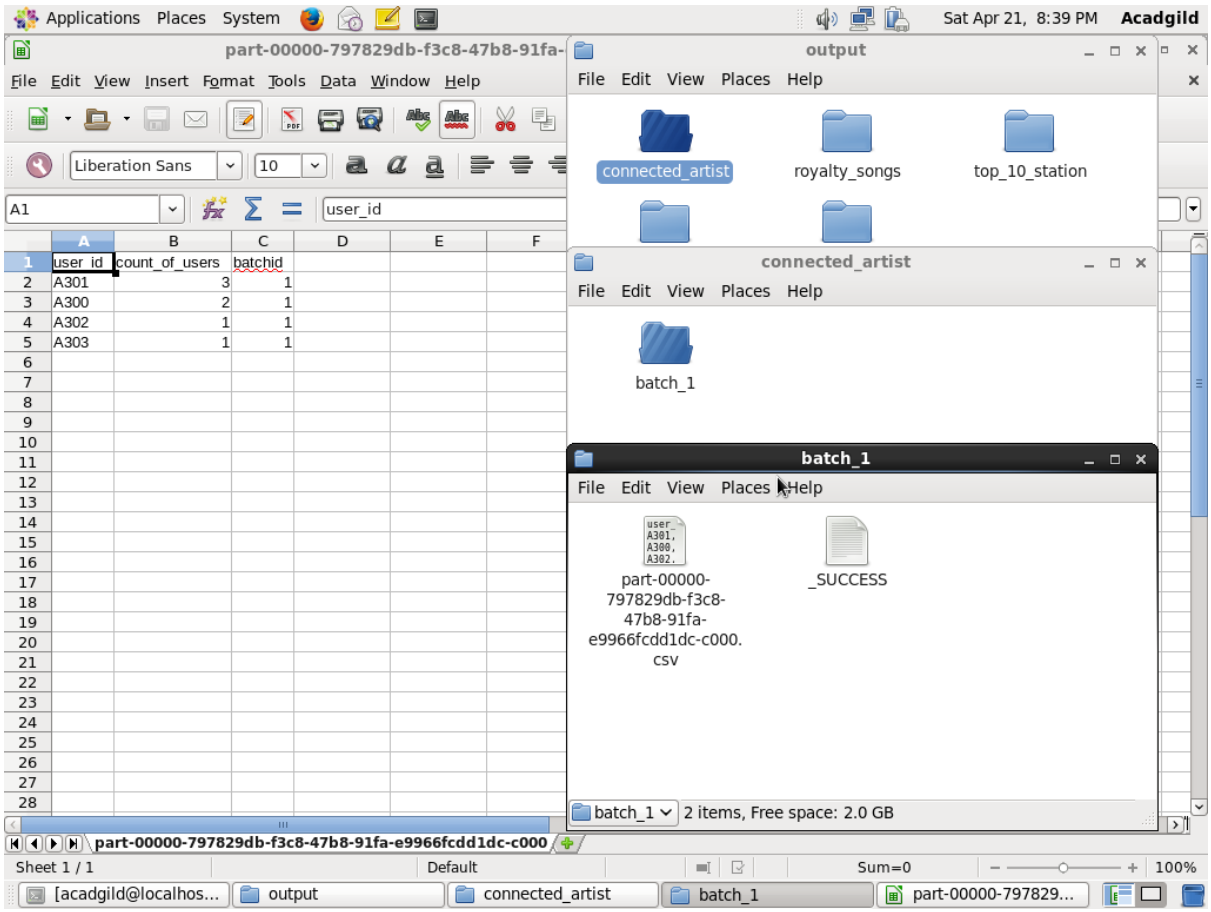
```
//*******************************************************************************
//                                PROBLEM 2
//*******************************************************************************

val prob_2 = spark.sql("select CASE WHEN ( subusr.user_id is null or " +
  "cast(music.u_timestamp as decimal(20,0)) > cast(subusr.end_dt as decimal(20,0)) " +
  ") then 'UNSUBSCRIBED' WHEN (subusr.user_id is NOT null or " +
  "cast(music.u_timestamp as decimal(20,0)) <= cast(subusr.end_dt as decimal(20,0)) " +
  ") then 'SUBSCRIBED' END AS USER_TYPE, " +
  "SUM(ABS(CAST(music.end_ts as decimal(20,0)) - CAST(music.start_ts as decimal(20,0))))" +
  " as duration ,batchid" +
  " from music_data music left outer join " +
  "subscribed_user subusr on music.user_id = subusr.user_id " +
  s"where music.status = 'pass' and music.batchid = $batch_id " +
  s"group by music.batchid, CASE WHEN ( subusr.user_id is null or " +
  "cast(music.u_timestamp as decimal(20,0)) > cast(subusr.end_dt as decimal(20,0)) " +
  ") then 'UNSUBSCRIBED' WHEN (subusr.user_id is NOT null or " +
  "cast(music.u_timestamp as decimal(20,0)) <= cast(subusr.end_dt as decimal(20,0)) " +
  ") then 'SUBSCRIBED' END " ).toDF()

prob_2.repartition(1).write.format("csv").option("header","true").save(s"/home/acadgild/project/output/user_response/
batch_$batch_id")
```



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

Solution :

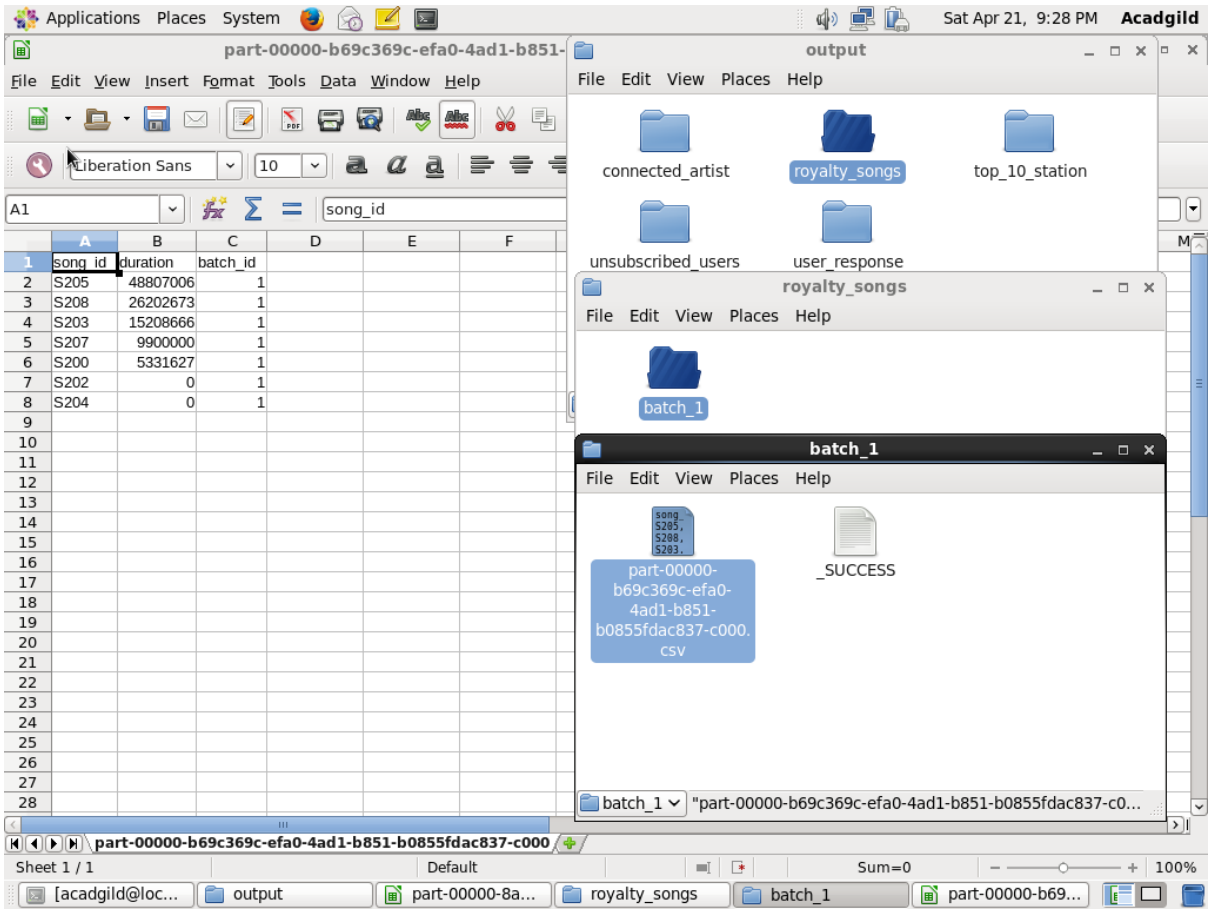
```
  //*******************************************************************************
  //                                PROBLEM 3
  //*******************************************************************************

  val prob_3 = spark.sql("select ua.artists, count(distinct ua.user_id) as count_of_users, " +
    "md.batchid from users_artists ua inner join (select artist_id,song_id,user_id,batchid" +
    s" from music_data where status = 'pass' and batchid = $batch_id)md " +
    s" on ua.artists = md.artist_id and ua.user_id = md.user_id group by ua.artists," +
    s" batchid order by count_of_users DESC limit 10").toDF("user_id","count_of_users","batchid")

    prob_3.write.format("csv").option("header","true").save(s"/home/acadgild/project/output/connected_artist/batch_
$batch_id")
```



Problem 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.

Solution :

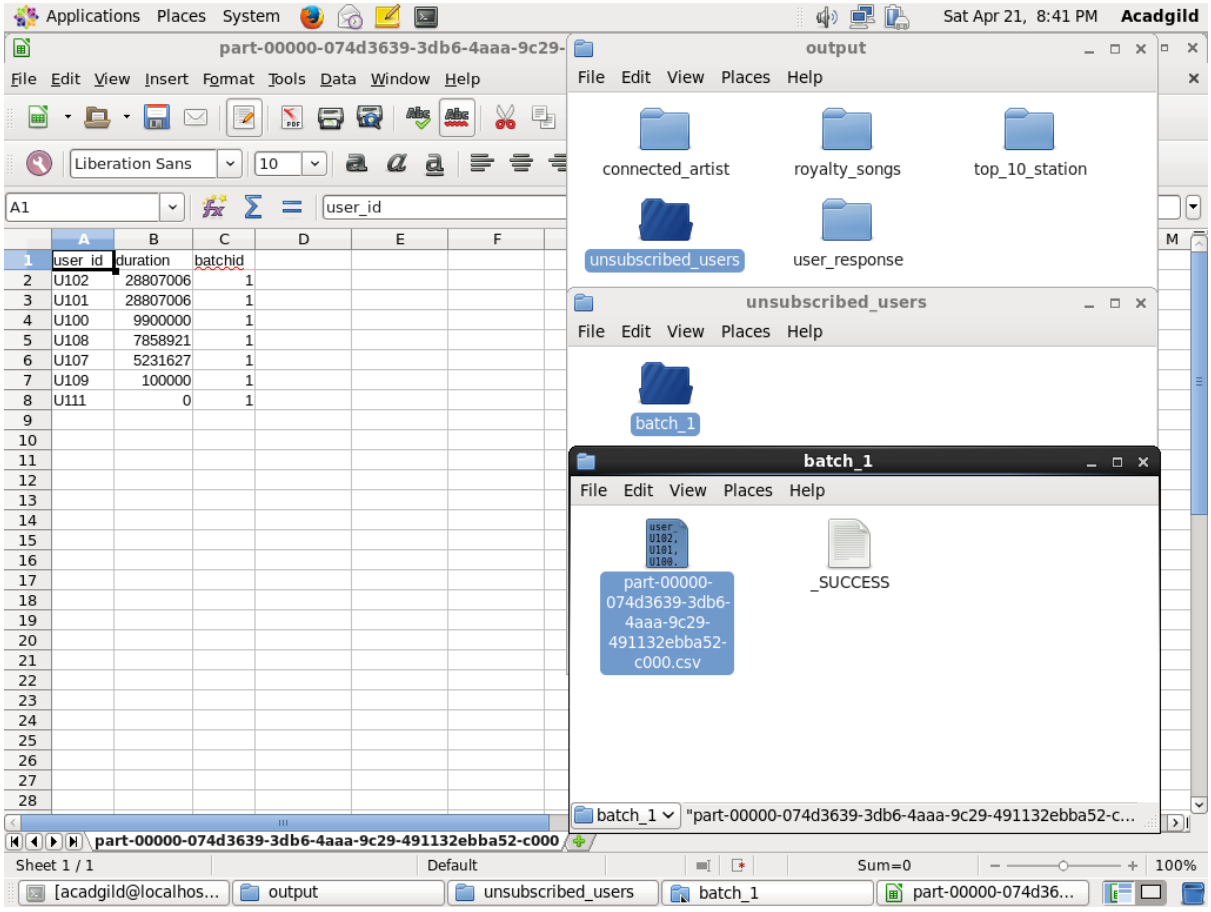
```
//*******************************************************************************
//                                  PROBLEM 4
//*******************************************************************************

val prob_4 = spark.sql("select song_id, " +
  "SUM(ABS(CAST(end_ts as decimal(20,0)) - " +
  " CAST(start_ts as decimal(20,0)))) as duration,batchid" +
  s" from music_data where status = 'pass' and batchid = $batch_id " +
  "and (u_like= 1 or song_end_type = 0) group by song_id, batchid " +
  "order by duration desc limit 10").toDF("song_id","duration","batch_id")

prob_4.write.format("csv").option("header","true").save(s"/home/acadgild/project/output/royalty_songs/batch_$batch_id")
```



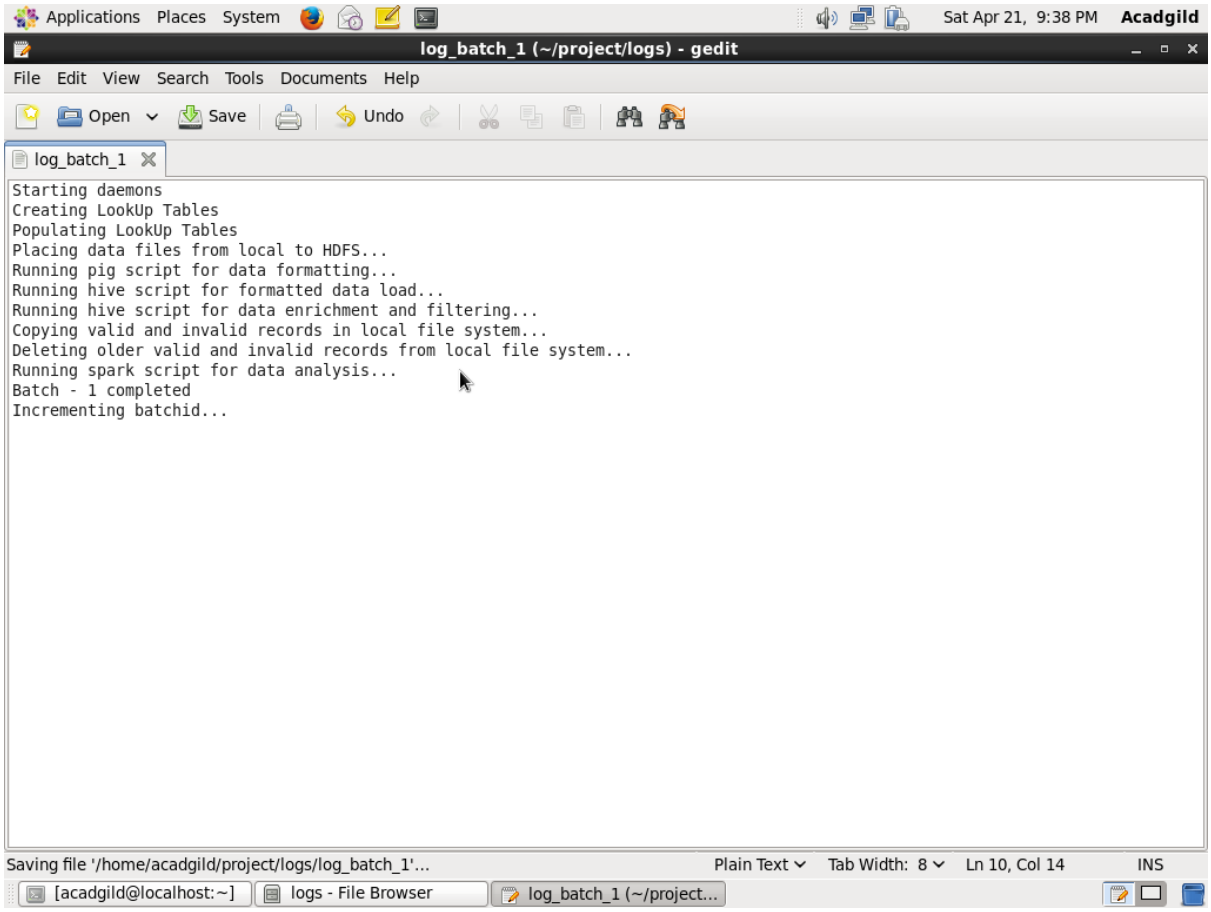
Problem 5 : Determine top 10 unsubscribed users who listened to the songs for the longest duration.

Solution :

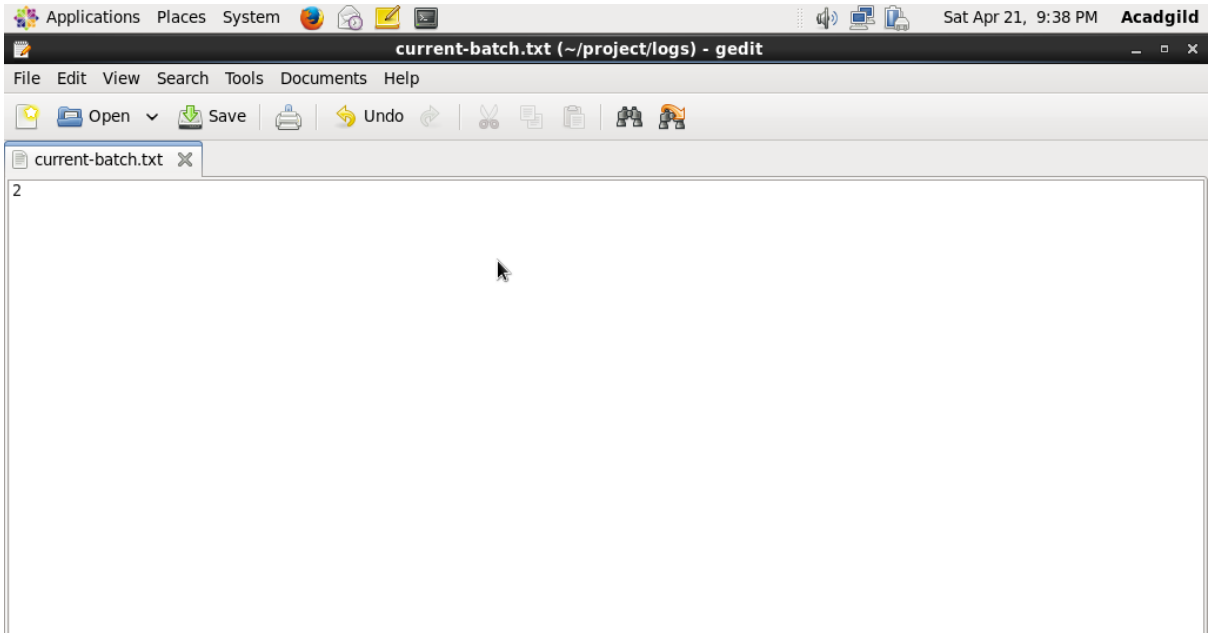
```
//*******************************************************************
//                              PROBLEM 5
//*******************************************************************

val prob_5 = spark.sql("select md.user_id, " +
  "SUM(ABS(CAST(end_ts as decimal(20,0)) - " +
  " CAST(start_ts as decimal(20,0)))) as duration,batchid" +
  s" from music_data md LEFT OUTER JOIN subscribed_user su " +
  s"on md.user_id = su.user_id where md.status = 'pass' and md.batchid = $batch_id " +
  "and (md.user_id IS NULL or (CAST(md.u_timestamp as DECIMAL(20,0)) > " +
  " cast(su.end_dt as DECIMAL(20,0))))" +
  " group by md.user_id, batchid " +
  "order by duration desc limit 10").toDF()

prob_5.write.format("csv").option("header","true").save(s"/home/acadgild/project/output/unsubscribed_users/batch_$batch_id")
```



View of Log File after Data Analysis :



Incremented Batch id to 2



****