
Project - Music Data Analysis
ACADGILD

ACADGILD

PROJECT

Music Data Analysis

Student Name: P Akash Menon

Course: BigData Engineering with Hadoop & Spark


Project - Music Data Analysis
ACADGILD

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.

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


Project - Music Data Analysis
ACADGILD

Look-Up 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

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.

Data present in lookup directory should be used in HBase.


Project - Music Data Analysis
ACADGILD

Dataset Creation:

Script: **Mobile Data**

generate_mob_data.py

```
from random import randint
from random import choice

file = open("/home/acadgild/project/data/mob/file.txt", "w")
count = 20

while (count > 0):
        geo_cd_list=["A", "E", "AU", "AP", "U"]
        song_end_type_list=["0","1","2","3"]
        timestamp_list=["1465230523", "1465130523", "1475130523", "1495130523"]
        start_ts_list=["1465230523", "1465130523", "1475130523", "1485130523"]
        end_ts_list=["1465230523", "1465130523", "1475130523", "1485130523"]

        if (count%15 == 0):
                user_id = ""
        else:
                user_id = "U" + str(randint(100,120))

        song_id = "S" + str(randint(200,210))

        if (count%11 == 0):
                artist_id = ""
        else:
                artist_id = "A" + str(randint(300,305))

        timestamp = choice(timestamp_list)
        start_ts = choice(start_ts_list)
        end_ts = choice(end_ts_list)

        if (count%12 == 0):
                geo_cd = ""
        else:
                geo_cd = choice(geo_cd_list)

        station_id = "ST" + str(randint(400,415))
        song_end_type = choice(song_end_type_list)
        like = str(randint(0,1))
        dislike = str(randint(0,1))

        file.write("%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s\n" % (user_id, song_id, artist_id, timestamp, start_ts, end_ts, geo_cd, station_id, song_end_type, like,
dislike))

        count = count-1

file.close()
```

Run script: **"python /home/acadgild/project/scripts/generate_mob_data.py"**

Project - Music Data Analysis

ACADGILD

Script: **Web Data**

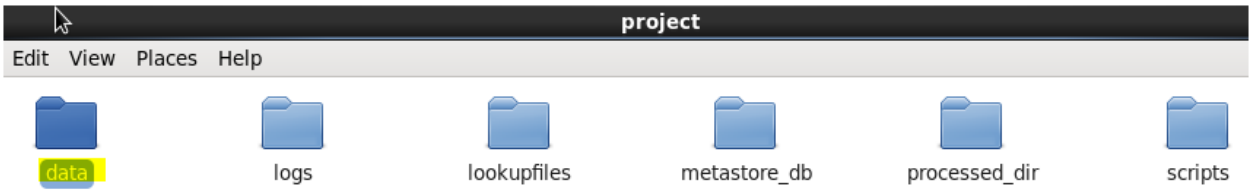
generate_web_data.py

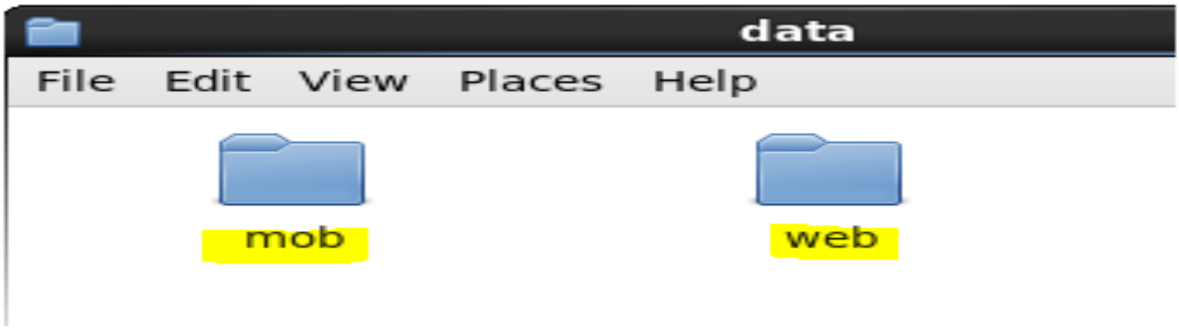
```
from random import randint
from random import choice

file = open("/home/acadgild/project/data/web/file.xml", "w")
count = 20

file.write("<records>\n")

while (count > 0):
        geo_cd_list=["A", "E", "AU", "AP", "U"]
        song_end_type_list=["0","1","2","3"]
        timestamp_list=["2016-05-10 12:24:22", "2016-06-09 22:12:36", "2016-07-10 01:38:09", "2017-05-09 08:09:22"]
        start_ts_list=["2016-05-10 12:24:22", "2016-06-09 22:12:36", "2016-07-10 01:38:09", "2017-05-09 08:09:22"]
        end_ts_list=["2016-05-10 12:24:22", "2016-06-09 22:12:36", "2016-07-10 01:38:09", "2017-05-09 08:09:22"]

        if (count%15 == 0):
                user_id = ""
        else:
                user_id = "U" + str(randint(100,120))

        song_id = "S" + str(randint(200,210))

        if (count%11 == 0):
                artist_id = ""
        else:
                artist_id = "A" + str(randint(300,305))

        timestamp = choice(timestamp_list)
        start_ts = choice(start_ts_list)
        end_ts = choice(end_ts_list)

        if (count%12 == 0):
                geo_cd = ""
        else:
                geo_cd = choice(geo_cd_list)

          station_id = "ST" + str(randint(400,415))
          song_end_type = choice(song_end_type_list)
          like = str(randint(0,1))
          dislike = str(randint(0,1))
          file.write("<record>\n")
          file.write("<user_id>%s</user_id>\n" % (user_id))
          file.write("<song_id>%s</song_id>\n" % (song_id))
          file.write("<artist_id>%s</artist_id>\n" % (artist_id))
          file.write("<timestamp>%s</timestamp>\n" % (timestamp))
          file.write("<start_ts>%s</start_ts>\n" % (start_ts))
          file.write("<end_ts>%s</end_ts>\n" % (end_ts))
          file.write("<geo_cd>%s</geo_cd>\n" % (geo_cd))
          file.write("<station_id>%s</station_id>\n" % (station_id))
          file.write("<song_end_type>%s</song_end_type>\n" % (song_end_type))
          file.write("<like>%s</like>\n" % (like))
          file.write("<dislike>%s</dislike>\n" % (dislike))
          file.write("</record>\n")

          count = count-1

file.write("</records>")
file.close()
```

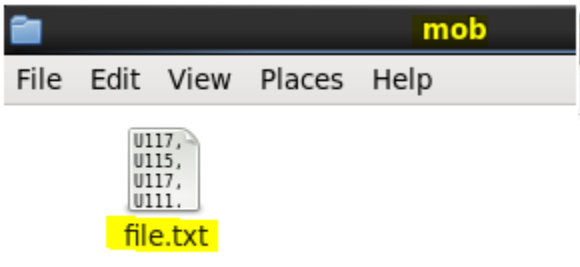
Run script : **"python /home/acadgild/project/scripts/generate_mob_data.py"**

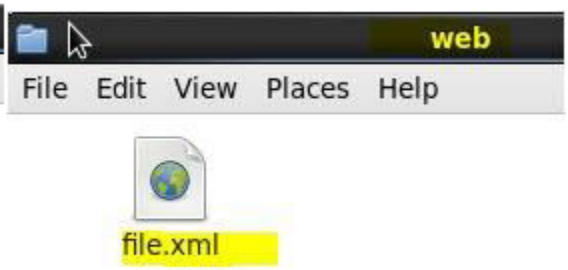
Project - Music Data Analysis

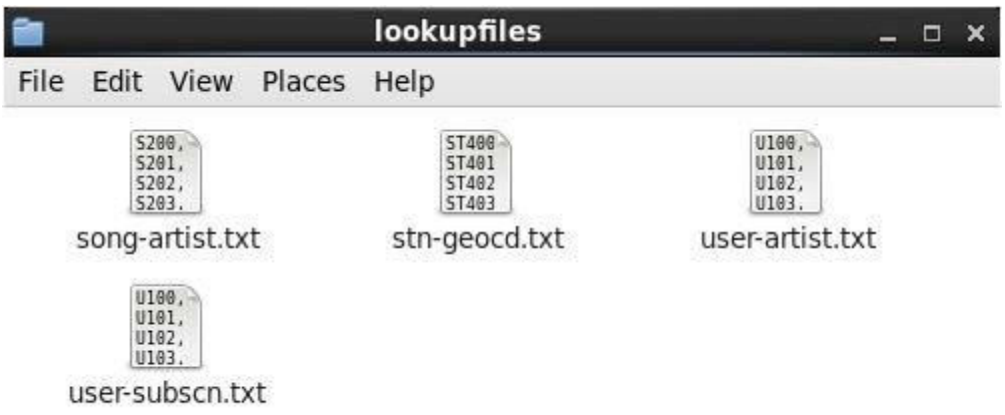
ACADGILD











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

Data Analysis

It is not only the data which is important, rather it is the insight it can be used to generate important. Once we have made the data ready for analysis, we have to perform below analysis on a daily basis.

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

2. Determine total duration of songs played by each type of user, where type of user can be 'subscribed' or 'unsubscribed'. An unsubscribed user is the one whose record is either not 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.


Project - Music Data Analysis
ACADGILD

Challenges and Optimizations

1. Look-Up tables are in NoSQL databases. Integrate them with the actual data flow.
2. Try to make joins as less expensive as possible.
3. Data Cleaning, Validation, Enrichment, Analysis and Post Analysis have to be automated. Try using schedulers.
4. Appropriate logs have to maintain to track the behavior and overcome failures in the pipeline.

A schematic flow of operations is shown below,

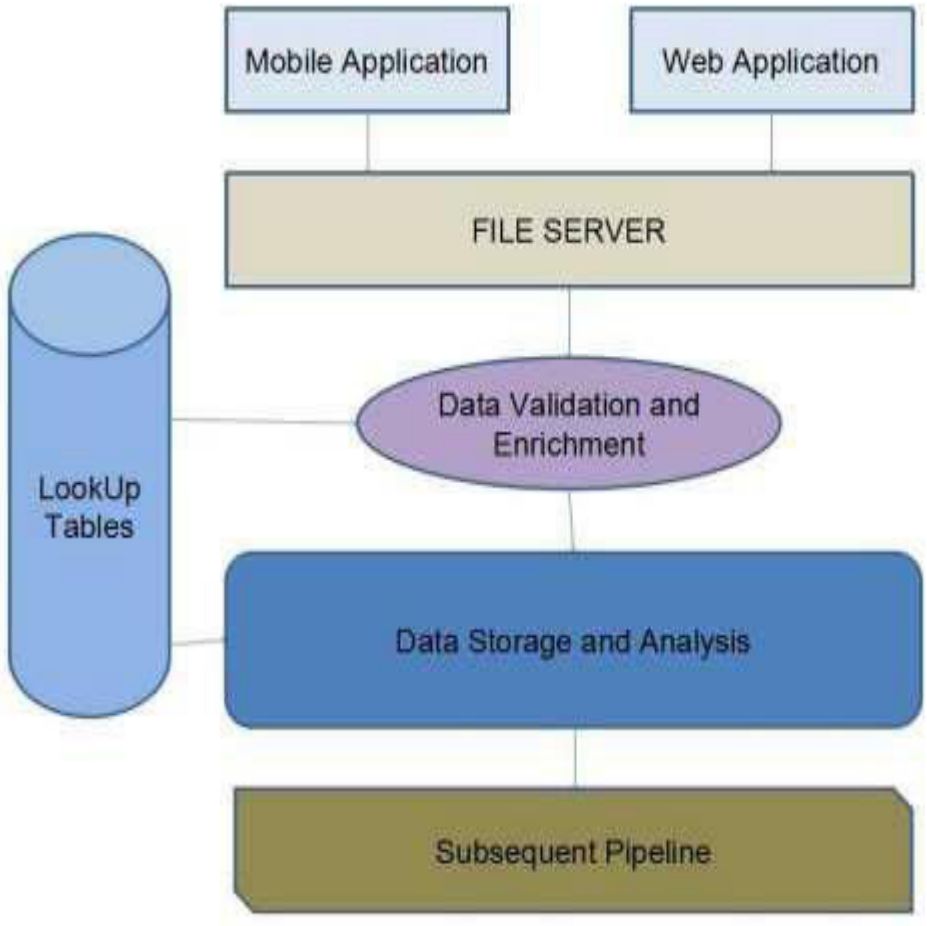


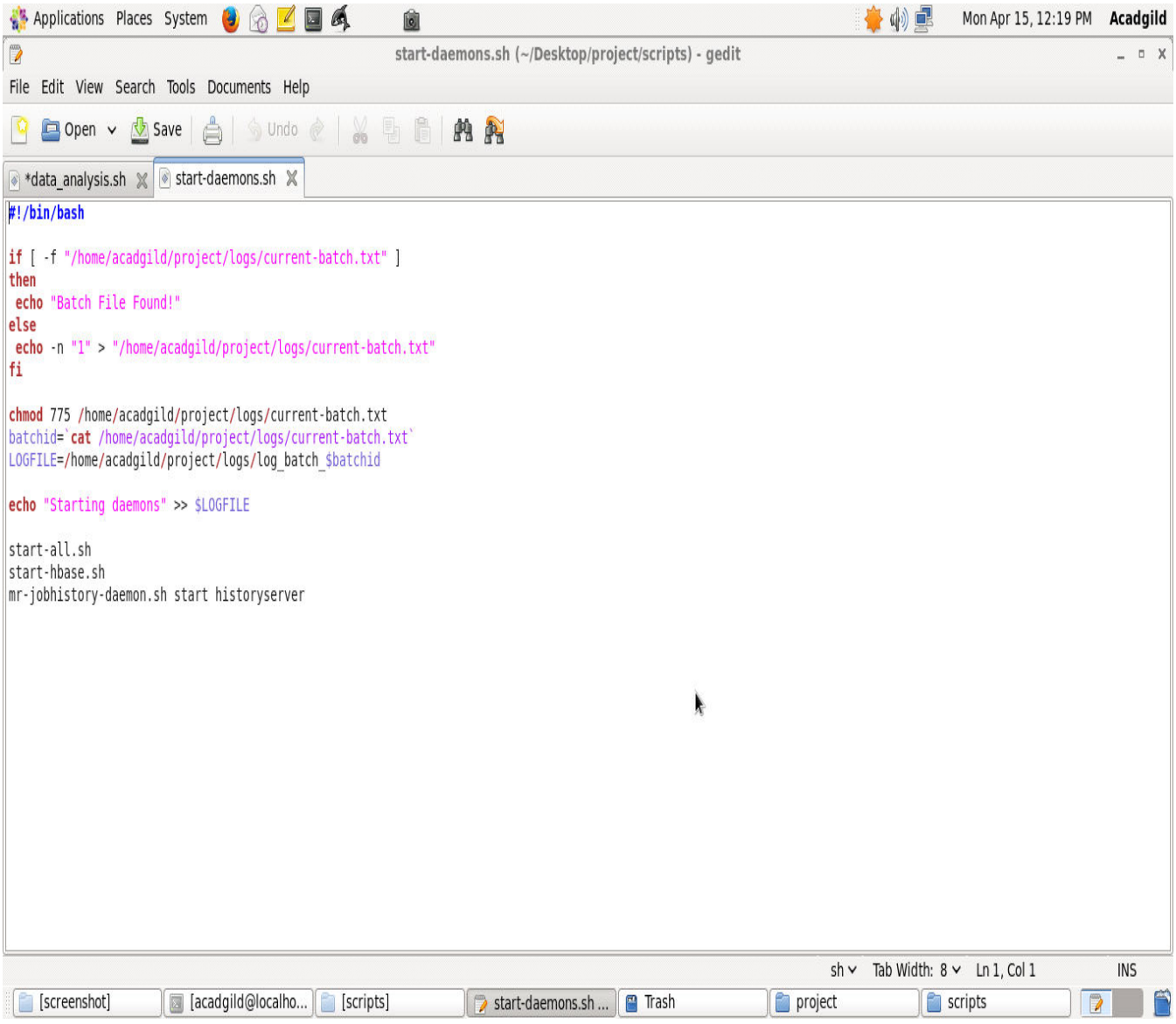
Fig – In the following sections, we are going to see the Music Data Analysis as per the above rules.


Project - Music Data Analysis
ACADGILD

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,




Project - Music Data Analysis
ACADGILD

2. Starting all daemons

Run Script: **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,

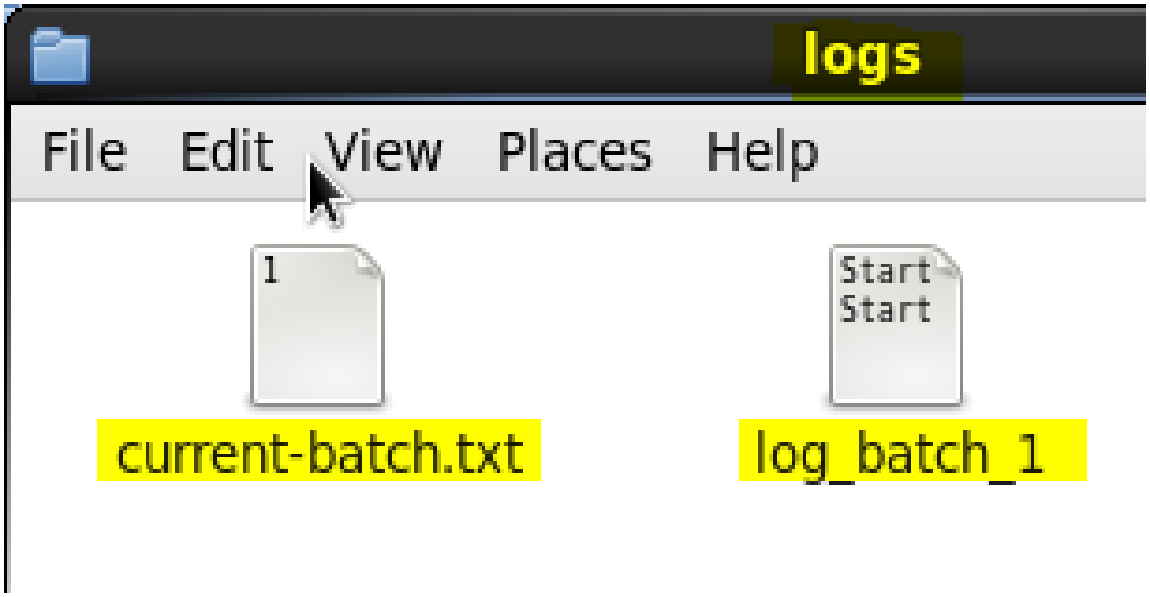
```
[acadgild@localhost scripts]$ sh start-daemons.sh
Batch File Found!
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
19/04/15 10:51:58 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-namenode-localhost.localdomain.out
localhost: starting datanode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-datanode-localhost.localdomain.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-secondarynamenode-localhost.localdomain.out
19/04/15 10:52:44 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
starting yarn daemons
starting resourcemanager, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/yarn-acadgild-resourcemanager-localhost.localdomain.out
localhost: starting nodemanager, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/yarn-acadgild-nodemanager-localhost.localdomain.out
localhost: starting zookeeper, logging to /home/acadgild/install/hbase/hbase-1.2.6/logs/hbase-acadgild-zookeeper-localhost.localdomain.out
starting master, logging to /home/acadgild/install/hbase/hbase-1.2.6/logs/hbase-acadgild-master-localhost.localdomain.out
starting regionserver, logging to /home/acadgild/install/hbase/hbase-1.2.6/logs/hbase-acadgild-1-regionserver-localhost.localdomain.out
starting historyserver, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/mapred-acadgild-historyserver-localhost.localdomain.out
```


Project - Music Data Analysis
ACADGILD

3. We can see the list active services using the ***jps*** command, see below screen shot and also Starting the hive metastore created a metastore_db in the location where we desired,

```
[acadgild@localhost ~]$ jps
3506 DataNode
3955 NodeManager
3382 NameNode
3687 SecondaryNameNode
6072 Jps
4601 HMaster
4795 JobHistoryServer
3852 ResourceManager
4701 HRegionServer
4511 HQuorumPeer
```

3. 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**.




Project - Music Data Analysis
ACADGILD

Data Ingestion, Formatting, Enrichment and Filtering

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
- 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

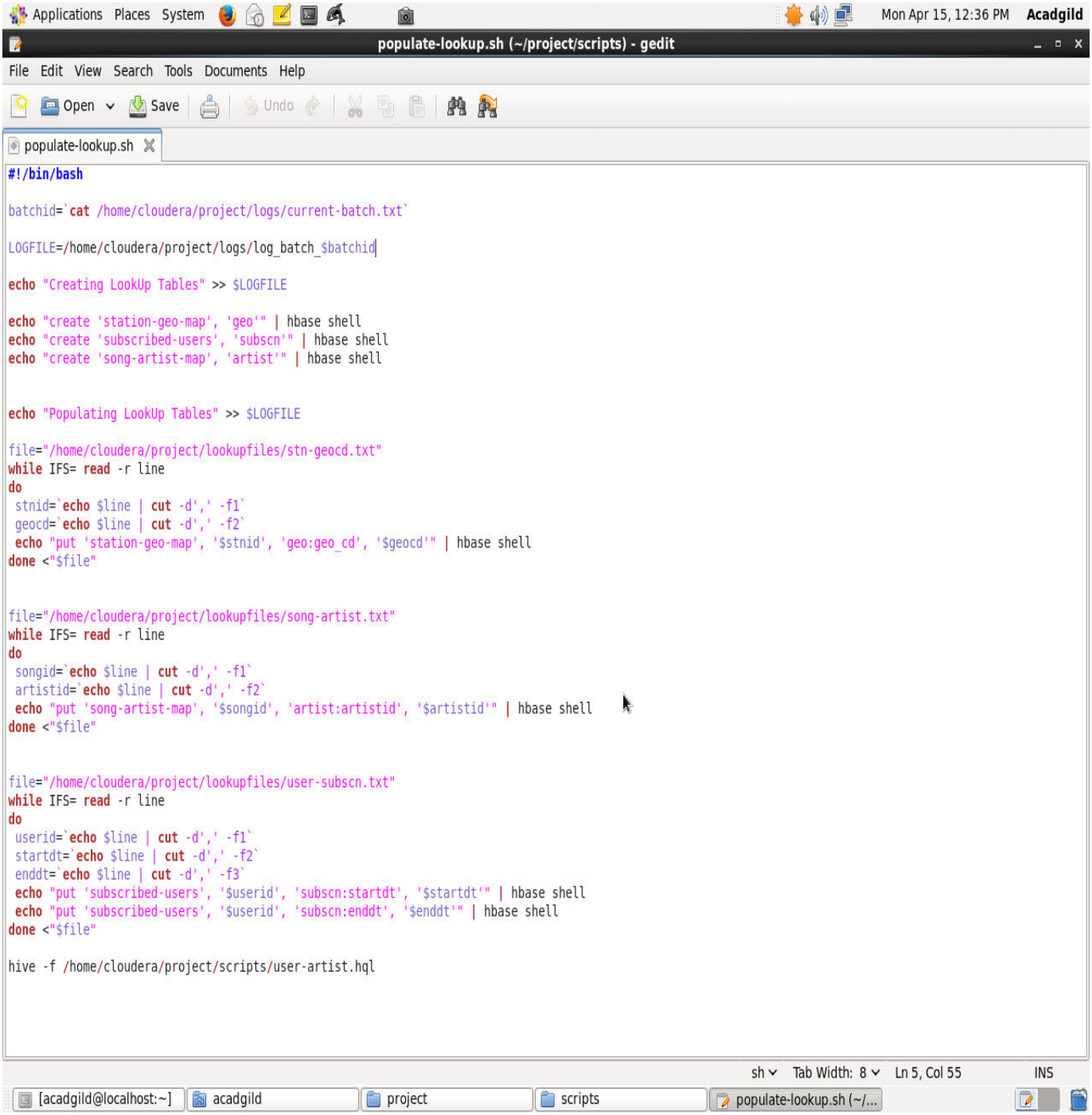
The "**populate-lookup.sh"** shell script creates the above 4 lookup tables in the Hbase and populate the data into the lookup tables from the dataset files.

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.

Project - Music Data Analysis

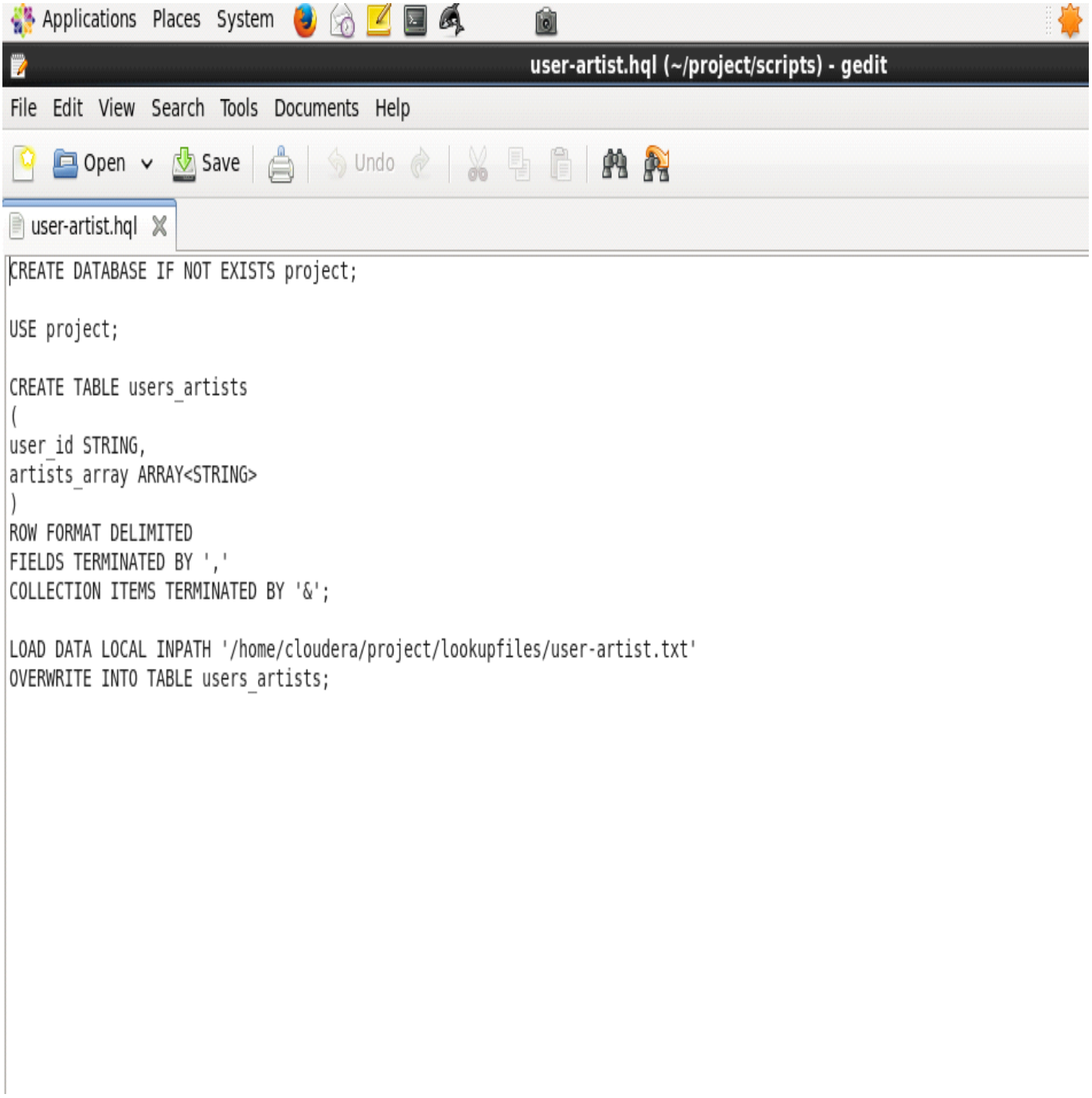
ACADGILD

Script file: "sh populate-lookup.sh




Project - Music Data Analysis
ACADGILD

Script: **users_artists table**

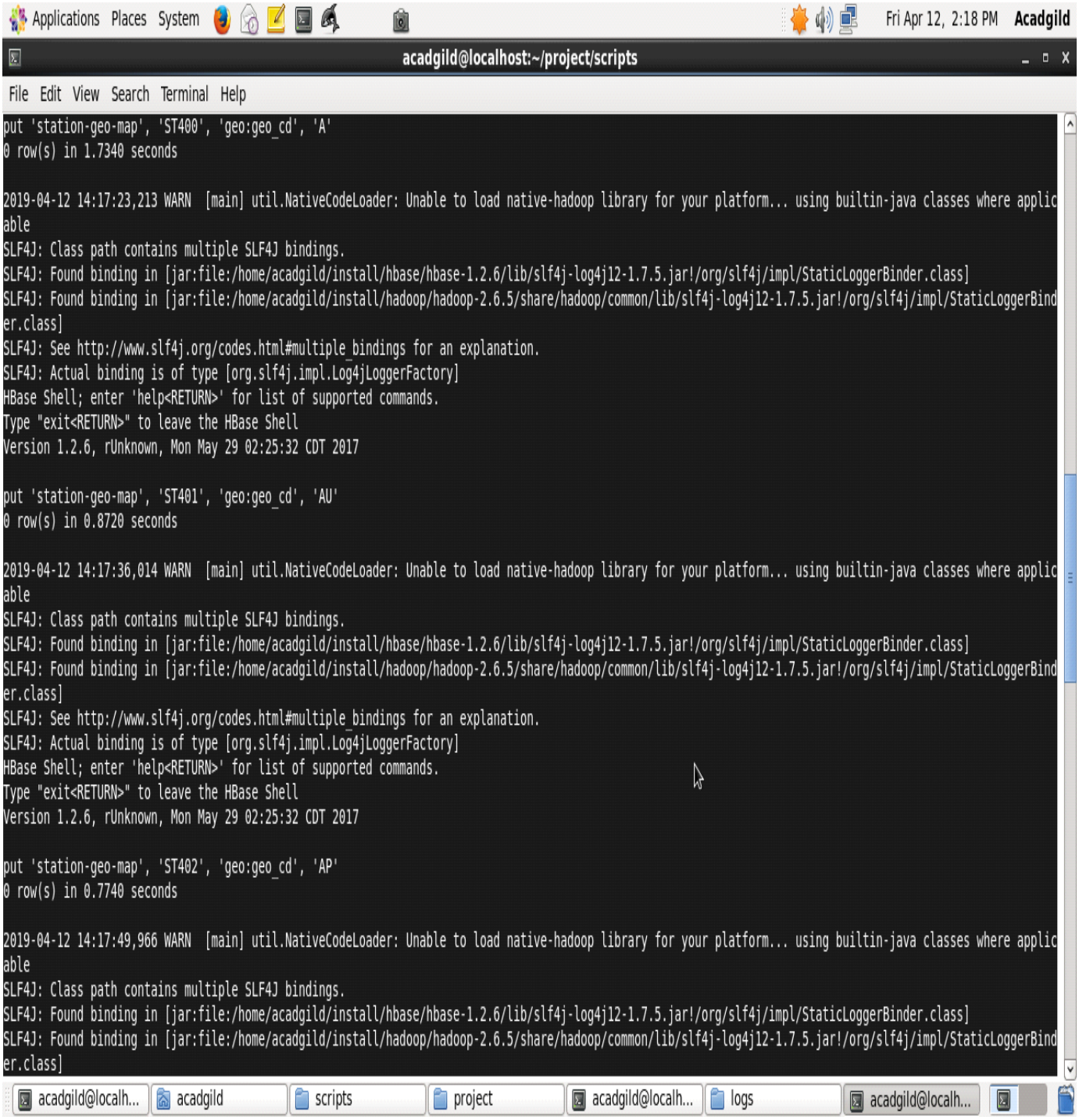



```
CREATE DATABASE IF NOT EXISTS project;

USE project;

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

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

Project - Music Data Analysis

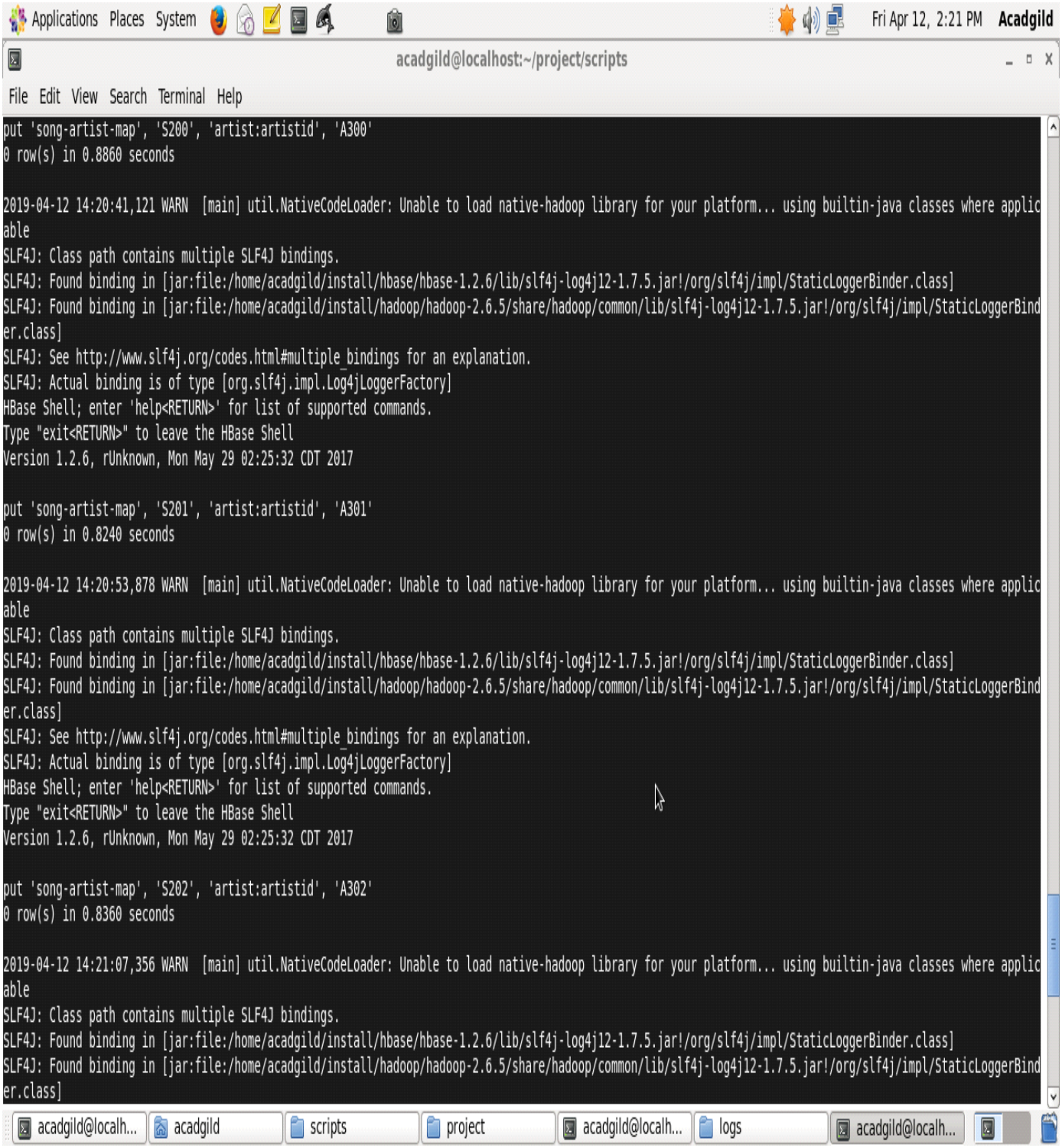
ACADGILD

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



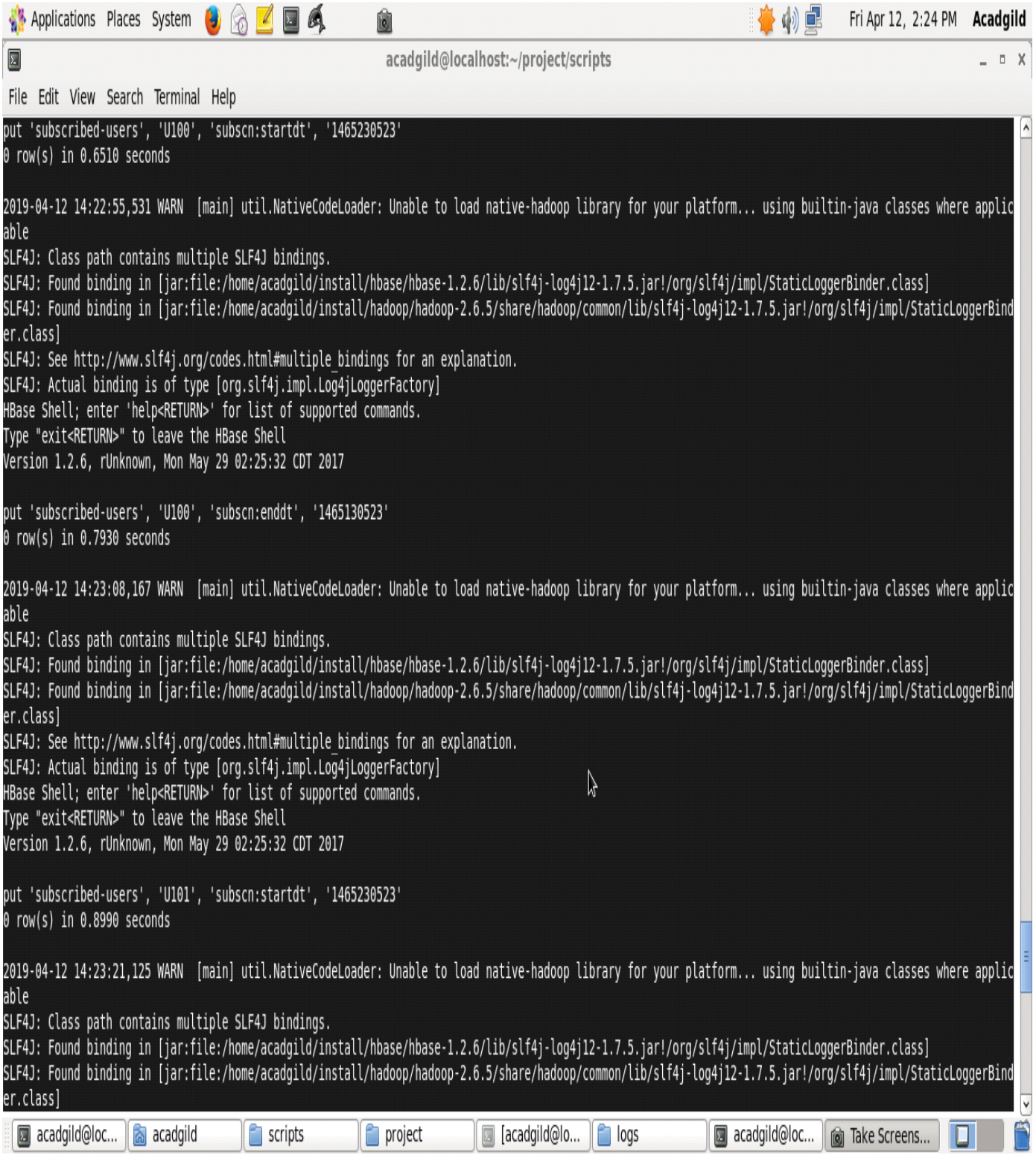
Project - Music Data Analysis

ACADGILD



Project - Music Data Analysis

ACADGILD




Project - Music Data Analysis
ACADGILD

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):002:0> list
TABLE
song-artist-map
station-geo-map
subscribed-users
3 row(s) in 0.8530 seconds
```

The values loaded as Lookup tables in hbase are **"song-artist-map" , "station-geo-map" and "subscribed-users"** shown below

```
=> ["song-artist-map", "station-geo-map", "subscribed-users"]
hbase(main):003:0> scan 'song-artist-map'
ROW                                   COLUMN+CELL
 S200                                 column=artist:artistid, timestamp=1555125203147, value=A300
 S201                                 column=artist:artistid, timestamp=1555125212597, value=A301
 S202                                 column=artist:artistid, timestamp=1555125222189, value=A302
 S203                                 column=artist:artistid, timestamp=1555125231607, value=A303
 S204                                 column=artist:artistid, timestamp=1555125241031, value=A304
 S205                                 column=artist:artistid, timestamp=1555125250575, value=A301
 S206                                 column=artist:artistid, timestamp=1555125260208, value=A302
 S207                                 column=artist:artistid, timestamp=1555125269663, value=A303
 S208                                 column=artist:artistid, timestamp=1555125279148, value=A304
 S209                                 column=artist:artistid, timestamp=1555125288688, value=A305
10 row(s) in 1.4450 seconds

hbase(main):004:0> scan 'station-geo-map'
ROW                                   COLUMN+CELL
 ST400                                column=geo:geo_cd, timestamp=1555125058914, value=A
 ST401                                column=geo:geo_cd, timestamp=1555125068558, value=AU
 ST402                                column=geo:geo_cd, timestamp=1555125077956, value=AP
 ST403                                column=geo:geo_cd, timestamp=1555125087503, value=J
 ST404                                column=geo:geo_cd, timestamp=1555125097320, value=E
 ST405                                column=geo:geo_cd, timestamp=1555125107351, value=A
 ST406                                column=geo:geo_cd, timestamp=1555125117329, value=AU
 ST407                                column=geo:geo_cd, timestamp=1555125126856, value=AP
 ST408                                column=geo:geo_cd, timestamp=1555125136387, value=E
 ST409                                column=geo:geo_cd, timestamp=1555125145832, value=E
 ST410                                column=geo:geo_cd, timestamp=1555125155195, value=A
 ST411                                column=geo:geo_cd, timestamp=1555125164751, value=A
 ST412                                column=geo:geo_cd, timestamp=1555125174268, value=AP
 ST413                                column=geo:geo_cd, timestamp=1555125183974, value=J
 ST414                                column=geo:geo_cd, timestamp=1555125193425, value=E
15 row(s) in 0.3040 seconds
```

Project - Music Data Analysis

ACADGILD

```
hbase(main):005:0> scan 'subscribed-users'
ROW                           COLUMN+CELL
 U100                         column=subscn:enddt, timestamp=1555125307930, value=1465130523
 U100                         column=subscn:startdt, timestamp=1555125298229, value=1465230523
 U101                         column=subscn:enddt, timestamp=1555125328199, value=1475130523
 U101                         column=subscn:startdt, timestamp=1555125317859, value=1465230523
 U102                         column=subscn:enddt, timestamp=1555125347628, value=1475130523
 U102                         column=subscn:startdt, timestamp=1555125337724, value=1465230523
 U103                         column=subscn:enddt, timestamp=1555125367208, value=1475130523
 U103                         column=subscn:startdt, timestamp=1555125357257, value=1465230523
 U104                         column=subscn:enddt, timestamp=1555125386657, value=1475130523
 U104                         column=subscn:startdt, timestamp=1555125376871, value=1465230523
 U105                         column=subscn:enddt, timestamp=1555125406178, value=1475130523
 U105                         column=subscn:startdt, timestamp=1555125396329, value=1465230523
 U106                         column=subscn:enddt, timestamp=1555125425544, value=1485130523
 U106                         column=subscn:startdt, timestamp=1555125415838, value=1465230523
 U107                         column=subscn:enddt, timestamp=1555125445179, value=1455130523
 U107                         column=subscn:startdt, timestamp=1555125435352, value=1465230523
 U108                         column=subscn:enddt, timestamp=1555125464926, value=1465230623
 U108                         column=subscn:startdt, timestamp=1555125455190, value=1465230523
 U109                         column=subscn:enddt, timestamp=1555125484400, value=1475130523
 U109                         column=subscn:startdt, timestamp=1555125474720, value=1465230523
 U110                         column=subscn:enddt, timestamp=1555125503313, value=1475130523
 U110                         column=subscn:startdt, timestamp=1555125493883, value=1465230523
 U111                         column=subscn:enddt, timestamp=1555125522384, value=1475130523
 U111                         column=subscn:startdt, timestamp=1555125512944, value=1465230523
 U112                         column=subscn:enddt, timestamp=1555125541514, value=1475130523
 U112                         column=subscn:startdt, timestamp=1555125531928, value=1465230523
 U113                         column=subscn:enddt, timestamp=1555125560773, value=1485130523
 U113                         column=subscn:startdt, timestamp=1555125551314, value=1465230523
 U114                         column=subscn:enddt, timestamp=1555125579964, value=1468130523
 U114                         column=subscn:startdt, timestamp=1555125570465, value=1465230523
15 row(s) in 0.3880 seconds
```


Project - Music Data Analysis
ACADGILD

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
: true
OK
Time taken: 12.317 seconds
OK
Time taken: 0.103 seconds
OK
Time taken: 1.406 seconds
Loading data to table project.users_artists
OK
Time taken: 2.204 seconds
```

hive> select * from users_artists;

```
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: 6.06 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".**

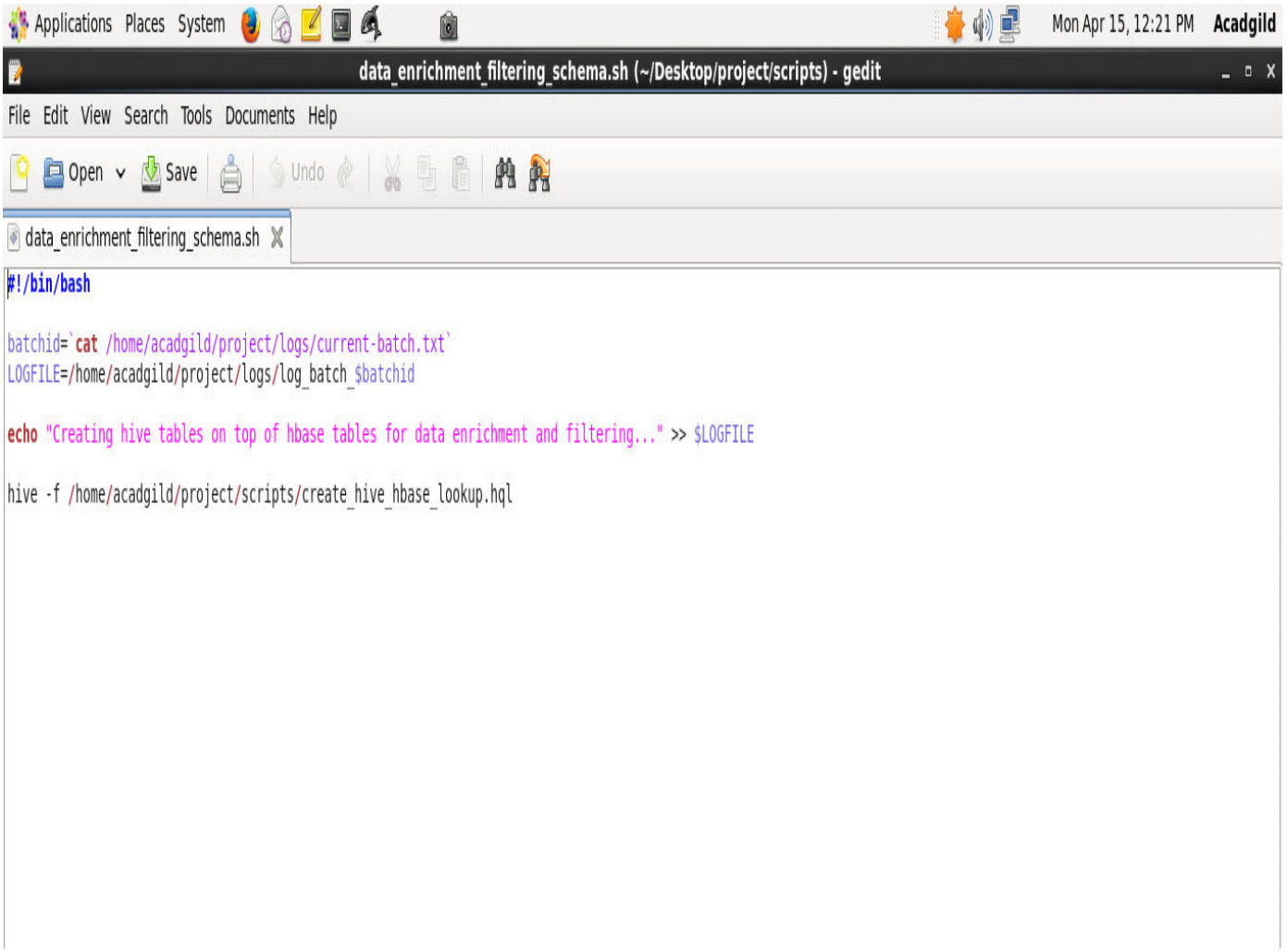

Project - Music Data Analysis
ACADGILD

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.

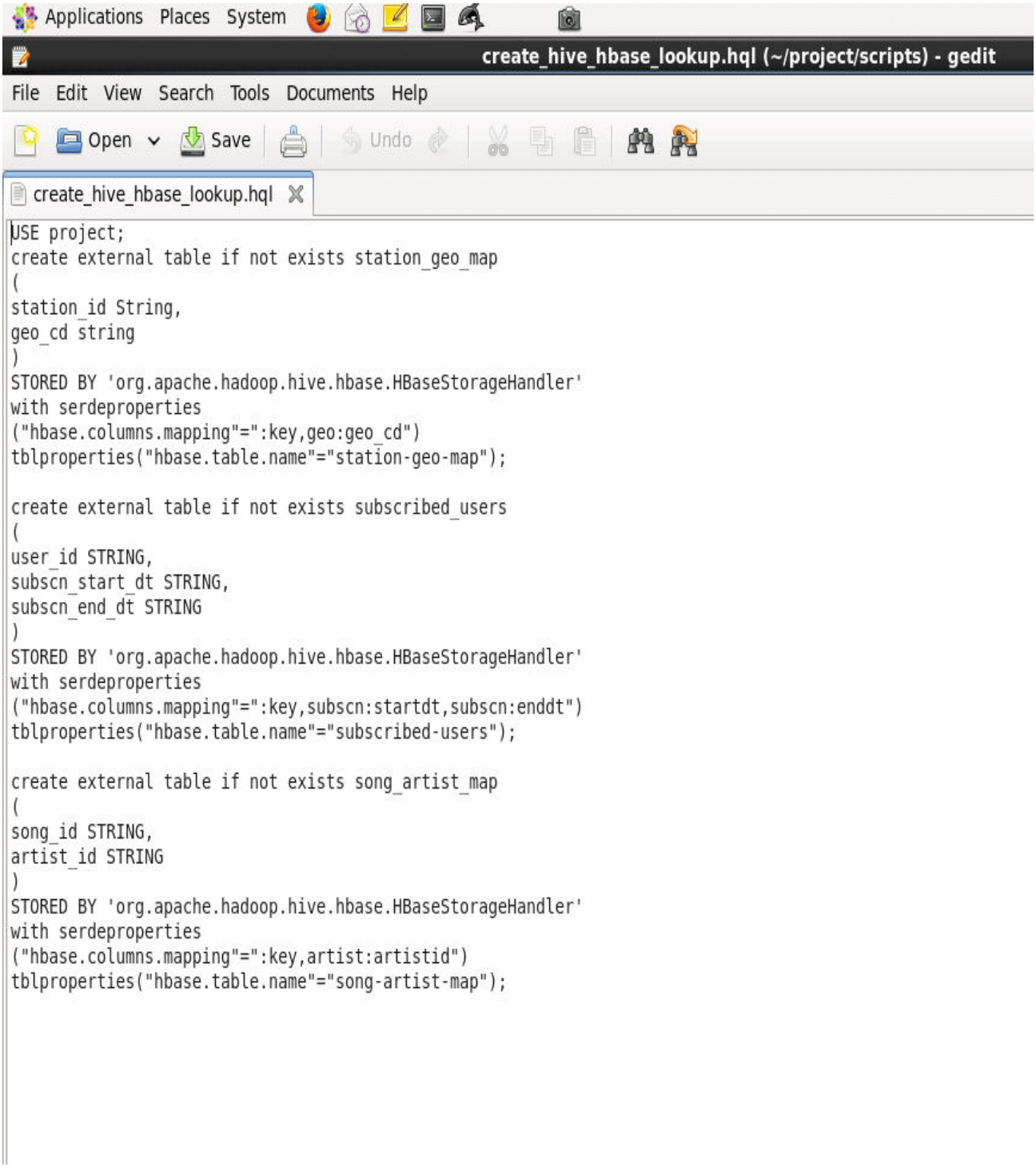
Run the script: **sh data_enrichment_filtering_schema.sh**

The script will run the **"create_hive_hbase_lookup.hql"** which will create the HIVE external tables with the help of **Hbase storage handler & SerDe properties**. The hive external tables will match the columns of **Hbase** tables to **HIVE** tables.




Project - Music Data Analysis
ACADGILD

create_hive_hbase_lookup.hql




```
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");
```


Project - Music Data Analysis
ACADGILD

The below screenshot we can see tables getting created in hive by running the hive script:

Run script : **"sh data_enrichement_filtering_schema.sh "**

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

Logging initialized using configuration in jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hive-common-2.3.2.jar!/hive-log4j2.properties Async: true
OK
Time taken: 7.561 seconds
OK
Time taken: 3.694 seconds
OK
Time taken: 0.308 seconds
OK
Time taken: 0.289 seconds
```

hive> show tables;

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


Project - Music Data Analysis
ACADGILD

hive> select * from song_artist_map;

hive> select * from station_geo_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: 8.654 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: 0.626 seconds, Fetched: 15 row(s)
```


Project - Music Data Analysis
ACADGILD

hive> select * from subscribed_users;

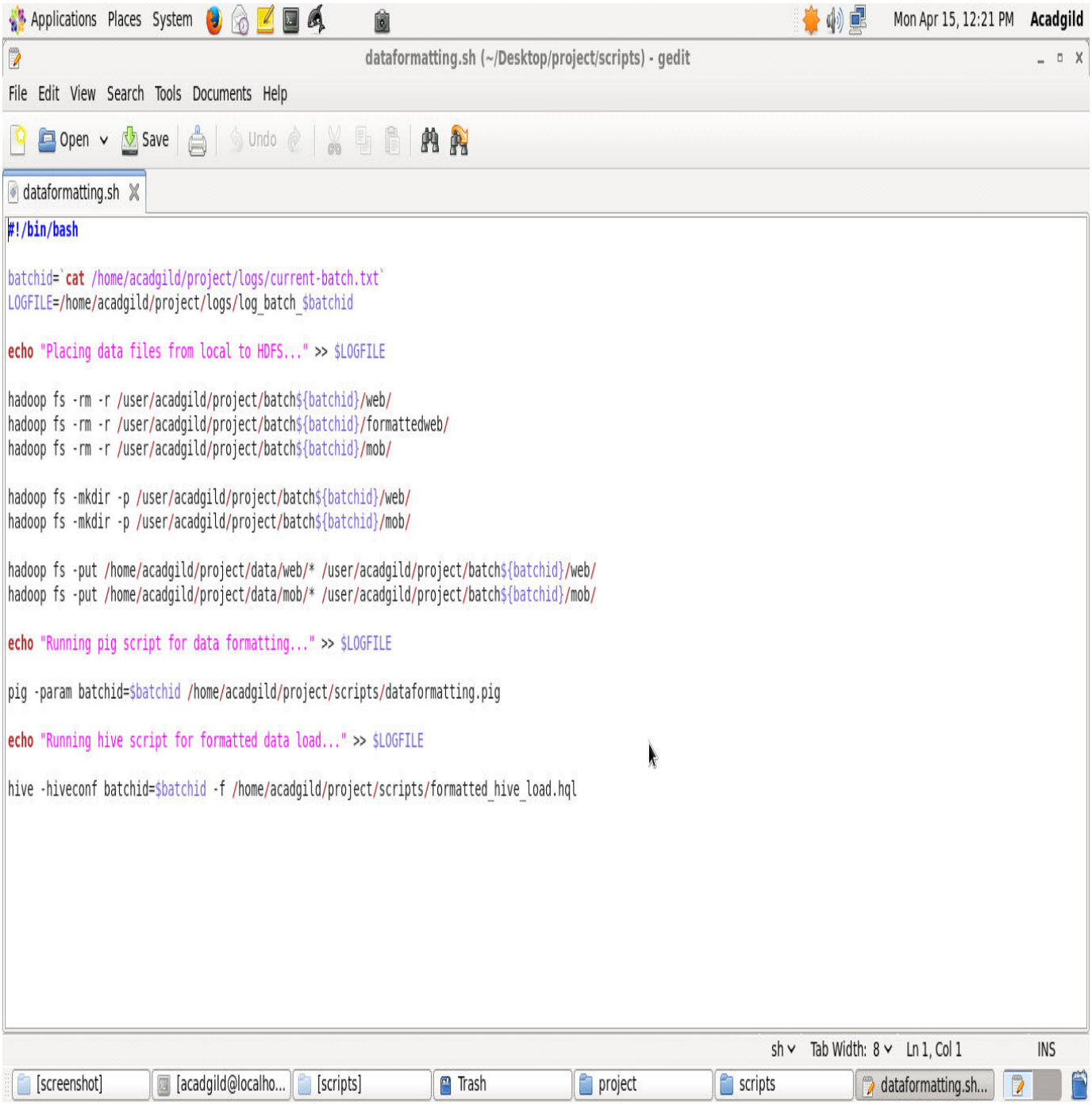
```
hive> select * from subscribed_users;
OK
U100    1465230523      1465130523
U101    1465230523      1475130523
U102    1465230523      1475130523
U103    1465230523      1475130523
U104    1465230523      1475130523
U105    1465230523      1475130523
U106    1465230523      1485130523
U107    1465230523      1455130523
U108    1465230523      1465230623
U109    1465230523      1475130523
U110    1465230523      1475130523
U111    1465230523      1475130523
U112    1465230523      1475130523
U113    1465230523      1485130523
U114    1465230523      1468130523
Time taken: 0.467 seconds, Fetched: 15 row(s)
```

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

Project - Music Data Analysis

ACADGILD

Run the script: **sh dataformatting.sh**




Project - Music Data Analysis
ACADGILD

Output:

```
[acadgild@localhost scripts]$ sh dataformatting.sh
19/04/12 14:43:25 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
rm: `/user/acadgild/project/batch1/web/': No such file or directory
19/04/12 14:43:28 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
rm: `/user/acadgild/project/batch1/formattedweb/': No such file or directory
19/04/12 14:43:30 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
rm: `/user/acadgild/project/batch1/mob/': No such file or directory
19/04/12 14:43:32 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
19/04/12 14:43:34 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
19/04/12 14:43:37 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
19/04/12 14:43:39 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
19/04/12 14:43:44 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
19/04/12 14:43:44 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
19/04/12 14:43:44 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2019-04-12 14:43:44,969 [main] INFO  org.apache.pig.Main - Apache Pig version 0.16.0 (r1746530) compiled Jun 01 2016, 23:10:49
2019-04-12 14:43:44,969 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/acadgild/project/scripts/pig_1555060424967.log
SLF4J: Class path contains multiple SLF4J bindings.
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/StaticLoggerBind
er.class]
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: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2019-04-12 14:43:45,625 [main] WARN  org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop library for your platform... using builtin-java c
lasses where applicable
2019-04-12 14:43:45,978 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file /home/acadgild/.pigbootup not found
2019-04-12 14:43:46,216 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.a
ddress
2019-04-12 14:43:46,216 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2019-04-12 14:43:46,216 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://localhost:8
020
2019-04-12 14:43:47,119 [main] INFO  org.apache.pig.PigServer - Pig Script ID for the session: PIG-dataformatting.pig-8a3832fa-dd3b-4da7-94a1-15cd77d6c9a9
2019-04-12 14:43:47,119 [main] WARN  org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
2019-04-12 14:43:47,274 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2019-04-12 14:43:48,186 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2019-04-12 14:43:48,454 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2019-04-12 14:43:48,583 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.textoutputformat.separator is deprecated. Instead, use mapredu
ce.output.textoutputformat.separator
2019-04-12 14:43:48,635 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: UNKNOWN
2019-04-12 14:43:48,709 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
```

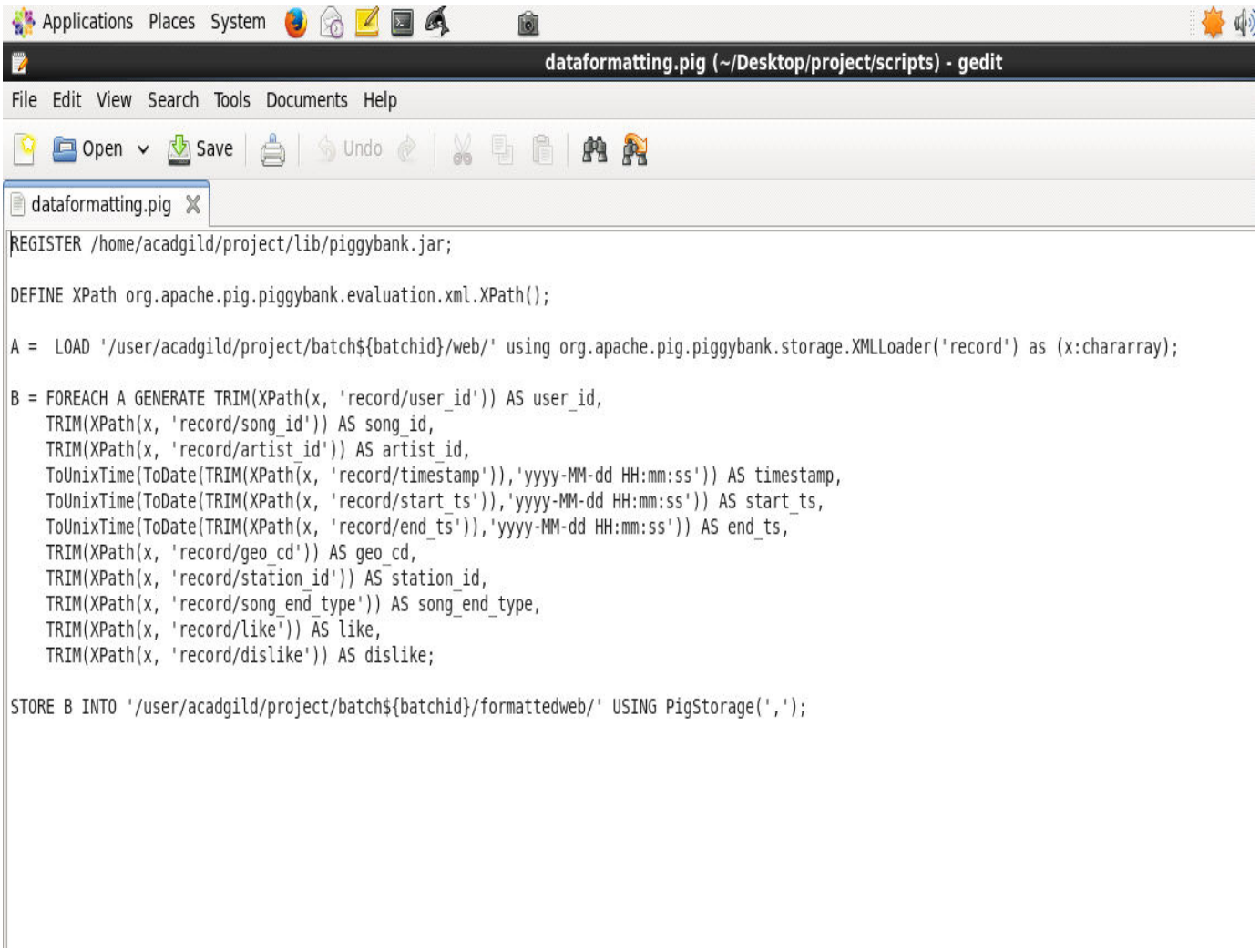

Project - Music Data Analysis
ACADGILD

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

- **Dataformatting.pig**
- **Formatted_hive_load.hql**

Pig script to parse the data from coming from **web_data.xml** to **csv** format and partition both web and
mob data based on based on batch ID's

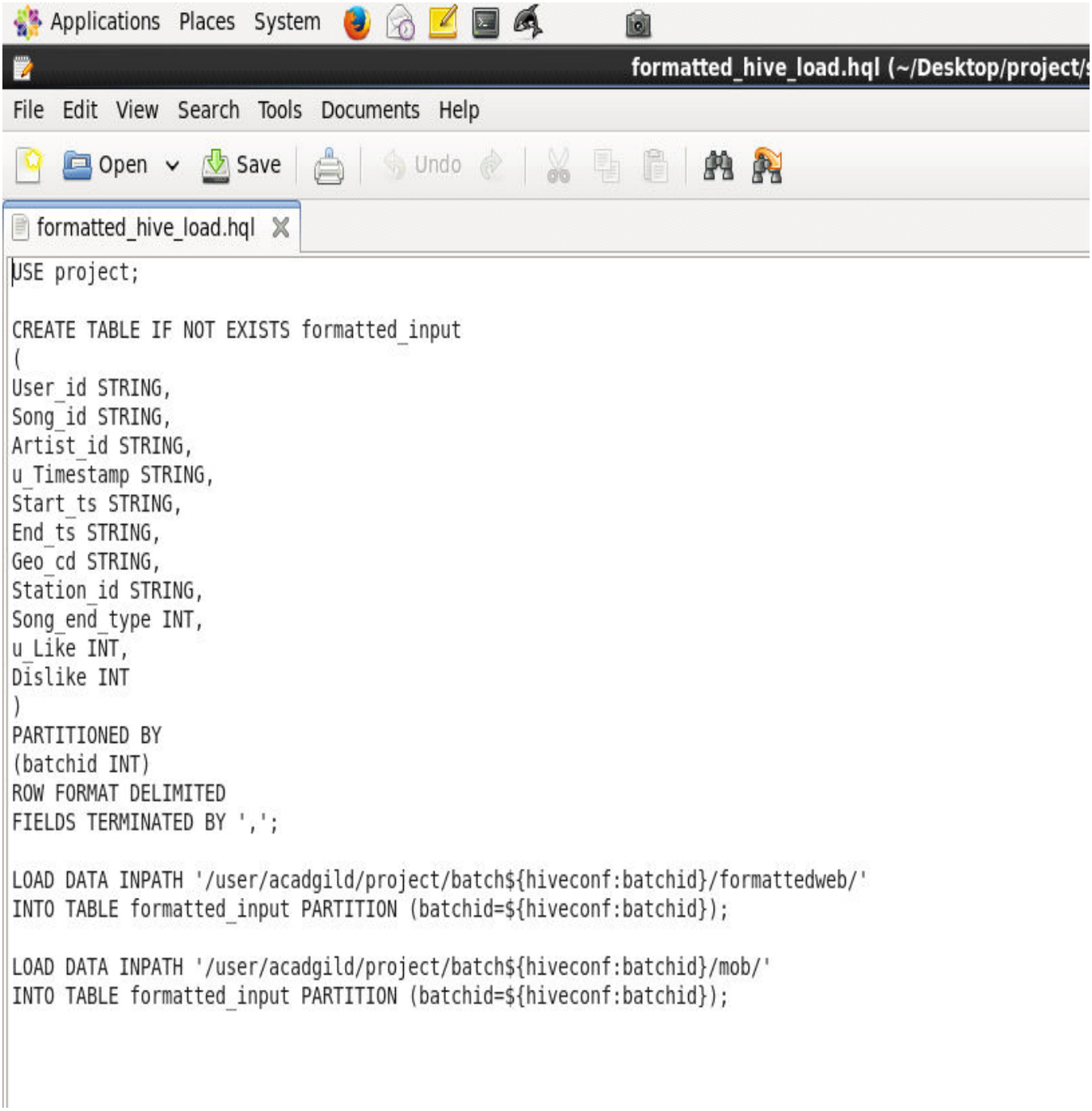
Script : **dataformatting.pig script**




```
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(',');
```


Project - Music Data Analysis
ACADGILD

formatted_hive_load.hql



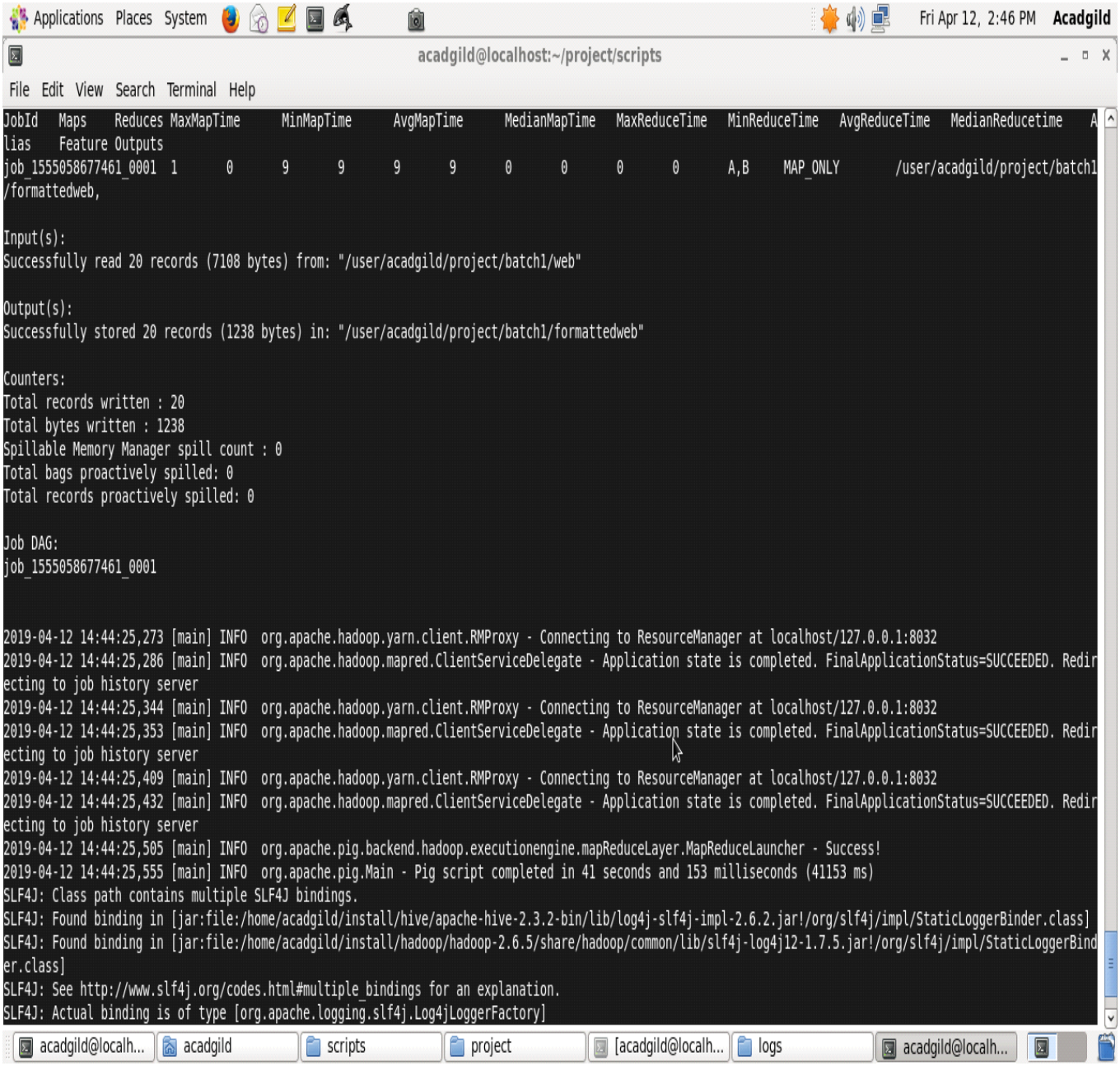

```
USE project;

CREATE TABLE IF NOT EXISTS formatted_input
(
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)
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});
```

Project - Music Data Analysis

ACADGILD

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

Pig script successful completion,




Project - Music Data Analysis
ACADGILD

Hive script successfully load the data into hive terminal,

```
Logging initialized using configuration in jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hive-common-2.3.2.jar!/hive-log4j2.properties Async
: true
OK
Time taken: 9.362 seconds
OK
Time taken: 0.456 seconds
Loading data to table project.formatted_input partition (batchid=1)
OK
Time taken: 2.91 seconds
Loading data to table project.formatted_input partition (batchid=1)
OK
Time taken: 1.582 seconds
```

In the above screenshot we can see the "**dataformatting.pig script**" along with the "**formatted_hive_load.hql**" executed successfully.

Output for formatted_hive_load.hql can be seen in hive as shown in fig:

hive> use project;

hive> show tables;

```
hive> use project;
OK
Time taken: 1.467 seconds
hive> show tables;
OK
formatted_input
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 0.719 seconds, Fetched: 5 row(s)
```


Project - Music Data Analysis
ACADGILD

hive> select * from formatted_input;

```
hive> select * from formatted_input;
OK
U107    S209    A305    1495130523      1485130523      1465130523      AU      ST412   1       1       0       1
U111    S203    A303    1465230523      1465230523      1485130523      E       ST405   3       0       0       1
U106    S210    A303    1475130523      1485130523      1475130523      AU      ST403   1       1       1       1
U106    S203    A301    1465130523      1475130523      1485130523      AU      ST409   3       0       1       1
U111    S204    A301    1495130523      1465230523      1485130523      AP      ST409   0       0       0       1
        S203    A303    1495130523      1465230523      1475130523      E       ST414   1       0       1       1
U112    S209    A305    1465130523      1465230523      1475130523      U       ST403   3       0       0       1
U113    S208    A301    1465230523      1465230523      1485130523      A       ST413   3       1       1       1
U112    S210    A300    1465230523      1465230523      1465130523              ST411   2       0       0       1
U103    S203            1465130523      1485130523      1485130523      AU      ST413   2       1       0       1
U108    S200    A301    1465230523      1465230523      1465130523      AU      ST404   0       0       0       1
U106    S201    A301    1495130523      1475130523      1465130523      AP      ST404   3       0       0       1
U107    S205    A303    1465230523      1485130523      1465230523      E       ST410   2       1       1       1
U102    S207    A300    1475130523      1465230523      1465230523      AU      ST404   2       0       1       1
U113    S208    A303    1475130523      1485130523      1475130523      E       ST400   1       1       1       1
U110    S206    A301    1465230523      1485130523      1485130523      A       ST404   1       1       1       1
U109    S208    A300    1465130523      1465130523      1465130523      E       ST415   3       0       0       1
U114    S205    A305    1475130523      1465130523      1475130523      E       ST408   3       0       0       1
U107    S203    A303    1465130523      1475130523      1475130523      AP      ST400   1       1       0       1
U114    S204    A305    1465130523      1465230523      1465230523      E       ST405   0       1       1       1
U118    S208    A301    1462908262      1465535556      1462908262      U       ST406   2       1       0       1
U103    S210    A303    1465535556      1462908262      1468139889      A       ST407   1       0       0       1
U112    S206    A301    1465535556      1468139889      1462908262      A       ST414   1       1       1       1
U104    S201    A301    1462908262      1468139889      1462908262      A       ST407   3       0       1       1
U101    S202    A305    1494342562      1462908262      1468139889      U       ST414   3       1       1       1
        S206    A304    1494342562      1465535556      1462908262      AP      ST412   3       0       1       1
U108    S200    A303    1462908262      1465535556      1468139889      AP      ST401   3       0       1       1
U115    S206    A302    1465535556      1465535556      1465535556      E       ST414   1       1       1       1
U110    S210    A303    1465535556      1465535556      1465535556              ST412   0       0       1       1
U110    S209            1494342562      1468139889      1465535556      AP      ST406   0       1       0       1
U110    S209    A302    1462908262      1465535556      1462908262      AP      ST400   3       0       1       1
U108    S204    A305    1462908262      1494342562      1465535556      AU      ST402   1       1       1       1
U103    S209    A302    1494342562      1462908262      1462908262      AU      ST405   1       0       1       1
U118    S205    A301    1494342562      1468139889      1468139889      AP      ST402   3       1       0       1
U112    S203    A304    1468139889      1468139889      1494342562      E       ST415   2       1       1       1
U115    S209    A300    1462908262      1462908262      1462908262      AU      ST412   1       0       1       1
U112    S207    A300    1462908262      1468139889      1465535556      E       ST409   2       1       1       1
U103    S201    A302    1468139889      1465535556      1468139889      AP      ST412   2       1       0       1
U103    S209    A300    1462908262      1462908262      1494342562      AU      ST408   1       1       1       1
U110    S204    A303    1494342562      1494342562      1462908262      U       ST406   2       0       1       1
Time taken: 0.209 seconds, Fetched: 40 row(s)
```

- In the above screenshot we can see the formatted input data with some null values in **user_id, aritist_id** and **geo_cd** columns which we will fill the enrichment script based on rules of enrichment for **artist_id** and **geo_cd** only. We will get neglect **user_id** because they didn't mentioned anything about **user_id** for enrichment purpose.
- Data formatting phase is executed successfully by loading both **mobile** and **web** data and partitioned based on **batchid**.

In this stage, we will enrich the data coming from **web** and **mobile** applications using the look-up table stored in **Hbase** and divide the records based on the enrichment rules into '**pass'** and '**fail'** records.

Rules for Data Enrichment

1. If any of like or dislike is **NULL** or **absent**, consider it as **0**.

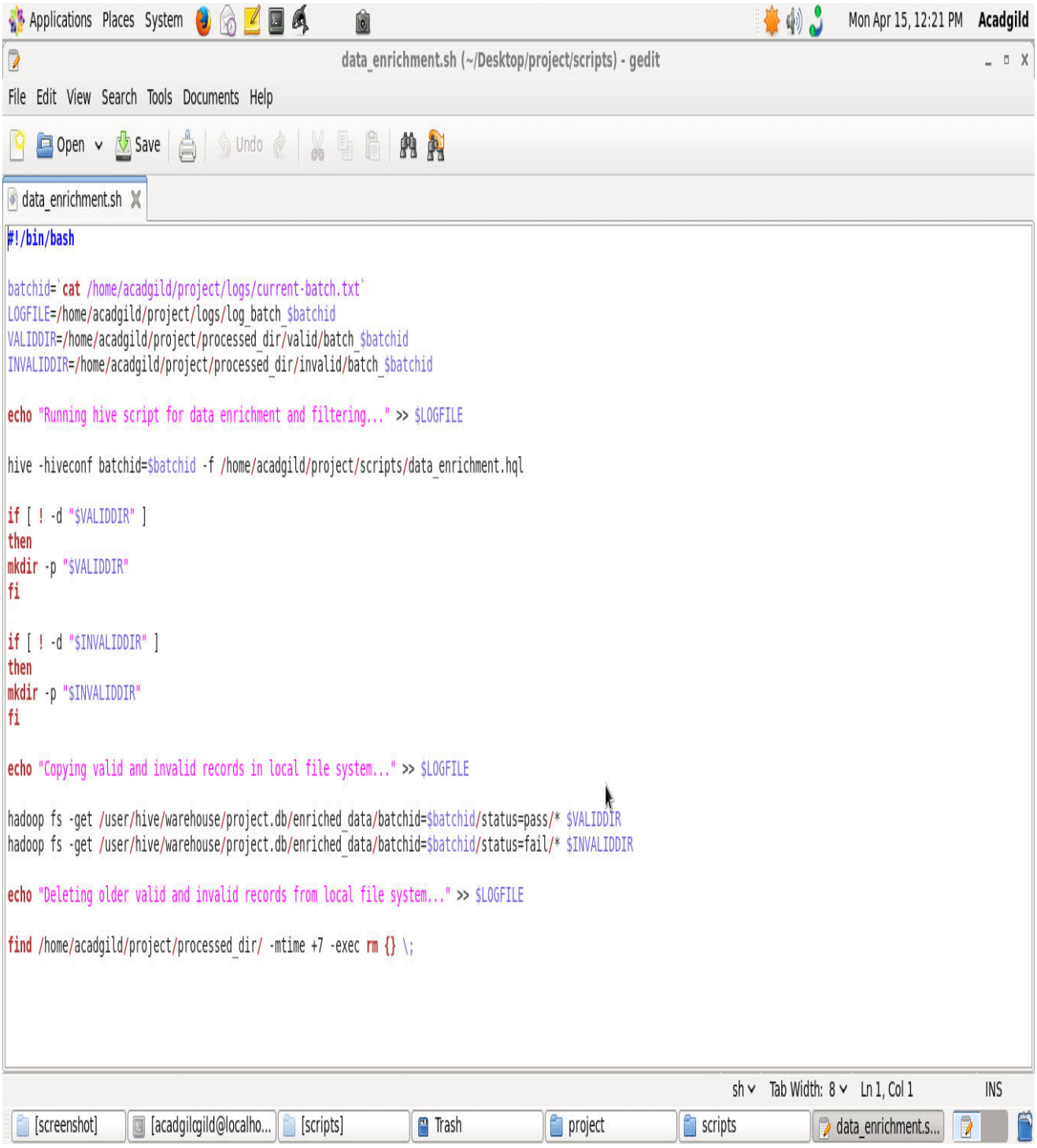
2. If fields like **Geo_cd** and **Artist_id** are NULL or absent, consult the lookup tables for fields **Station_id** and **Song_id** respectively to get the values of **Geo_cd** and **Artist_id.**

3. If corresponding lookup entry is not found, consider that **record** to be **invalid**

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

Project - Music Data Analysis
ACADGILD

Script: **"sh data_enrichment.sh"**



Project - Music Data Analysis

ACADGILD

Hive table: **"data_enrichment.hql"**

Applications Places System

data_enrichment.hql (~/project/scripts) - gedit

File Edit View Search Tools Documents Help

Open Save Undo

data_enrichment.hql

```
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.u_dislike IS NULL, 0, i.u_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};
```


Project - Music Data Analysis
ACADGILD

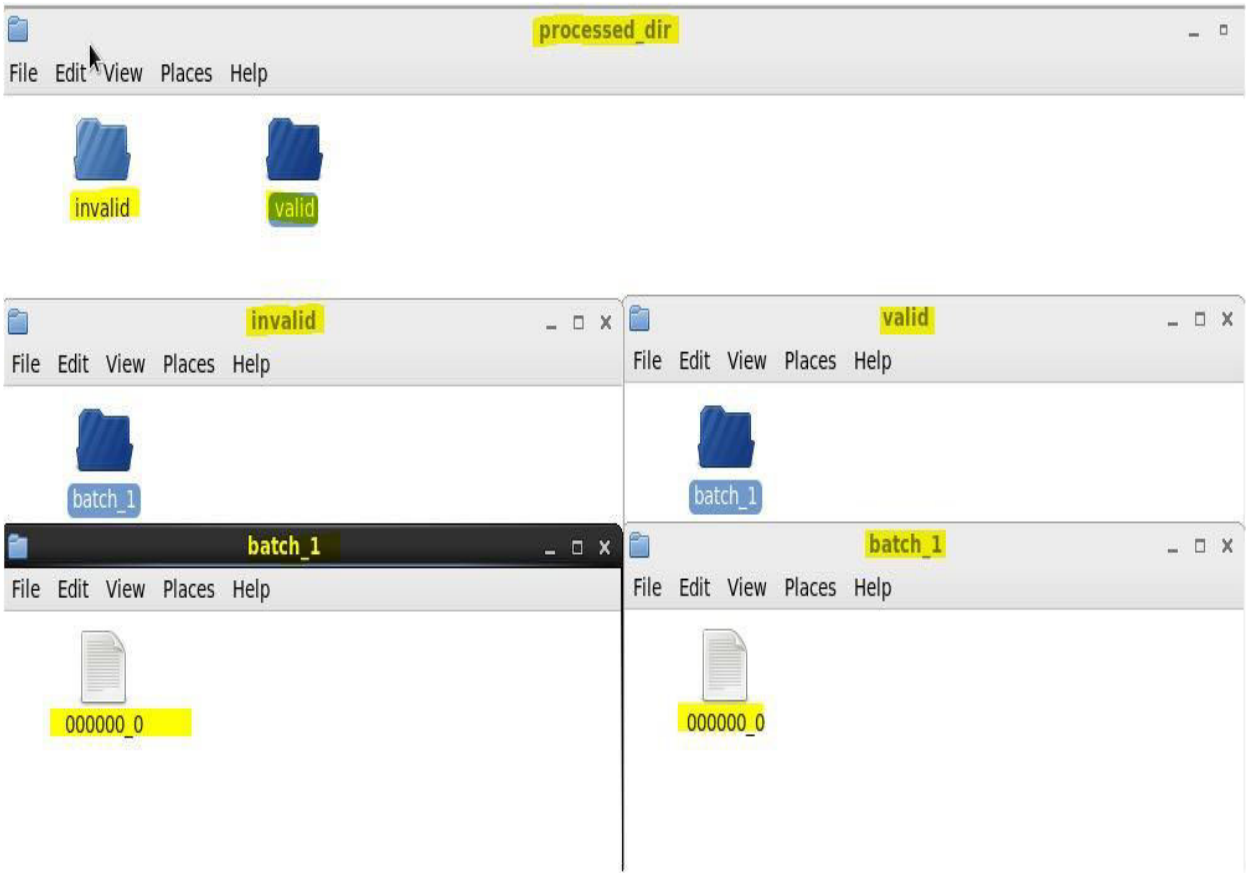
Run script : **"sh data_enrichment.sh**

Project - Music Data Analysis

ACADGILD

At the end script will automatically divide the records based on status **pass** & **fail** and dump the result into **"processed_dir"** folder with valid and invalid folders.

Project - Music Data Analysis
ACADGILD



Now we can check whether the data properly loaded in the hive terminal or not.

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

In the below screenshot we have data for **enriched_data** table where we filled the null values of **artist_id** and **geo_cd** of formatted input with the help of lookup tables,


Project - Music Data Analysis

ACADGILD

hive> select * from enriched_data;

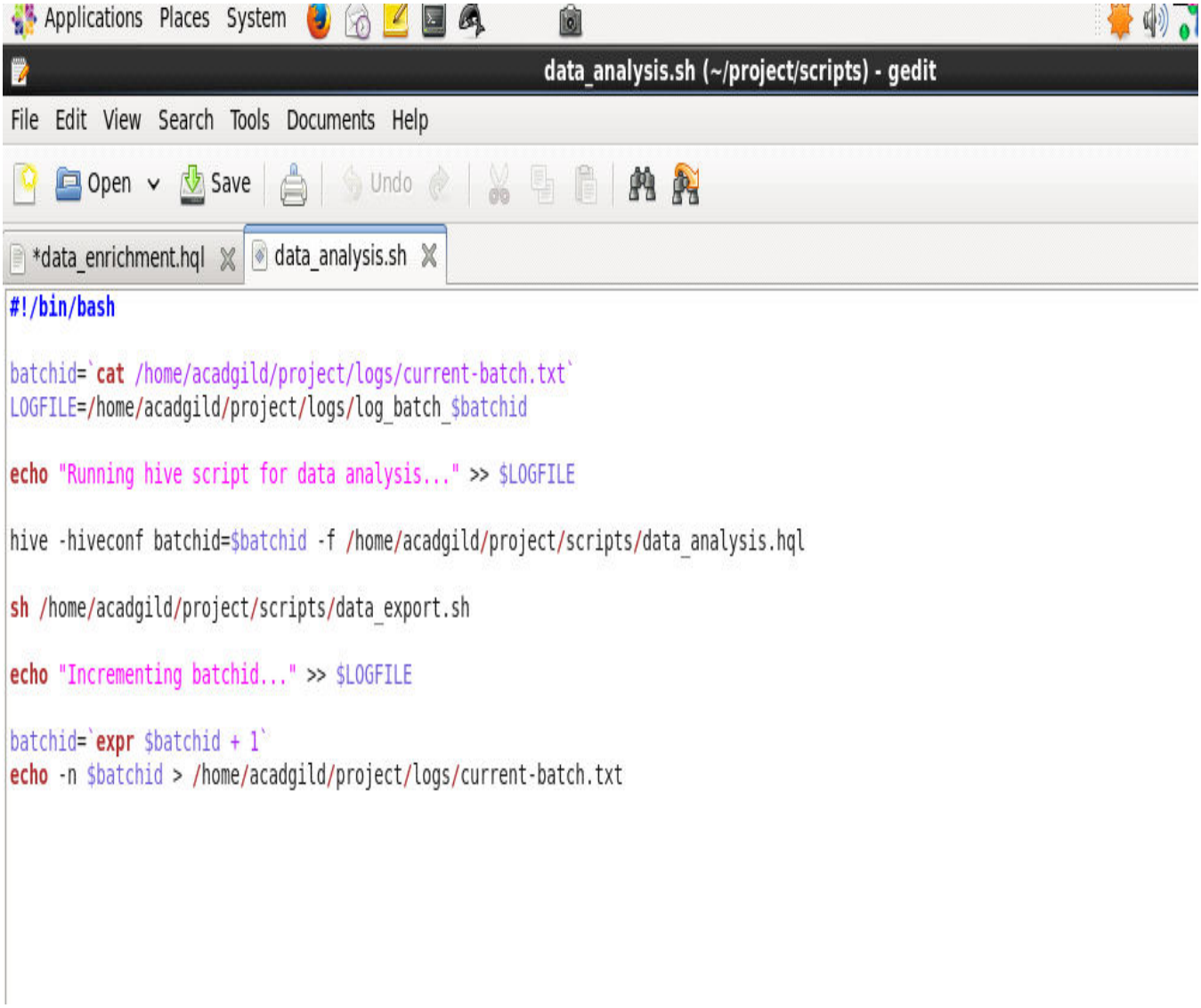
```
hive> select * from enriched_data;
OK
U101    S202    A302    1494342562      1462908262      1468139889      E       ST414   3       1       1       1       fail
        S203    A303    1495130523      1465230523      1475130523      E       ST414   1       0       1       1       fail
U112    S203    A303    1468139889      1468139889      1494342562      NULL    ST415   2       1       1       1       fail
U114    S204    A304    1465130523      1465230523      1465230523      A       ST405   0       1       1       1       fail
U108    S204    A304    1462908262      1494342562      1465535556      AP      ST402   1       1       1       1       fail
U107    S205    A301    1465230523      1485130523      1465230523      A       ST410   2       1       1       1       fail
U110    S206    A302    1465230523      1485130523      1485130523      E       ST404   1       1       1       1       fail
        S206    A302    1494342562      1465535556      1462908262      AP      ST412   3       0       1       1       fail
U115    S206    A302    1465535556      1465535556      1465535556      E       ST414   1       1       1       1       fail
U112    S206    A302    1465535556      1468139889      1462908262      E       ST414   1       1       1       1       fail
U112    S207    A303    1462908262      1468139889      1465535556      E       ST409   2       1       1       1       fail
U109    S208    A304    1465130523      1465130523      1465130523      NULL    ST415   3       0       0       1       fail
U113    S208    A304    1465230523      1465230523      1485130523      J       ST413   3       1       1       1       fail
U113    S208    A304    1475130523      1485130523      1475130523      A       ST400   1       1       1       1       fail
U103    S209    A305    1462908262      1462908262      1494342562      E       ST408   1       1       1       1       fail
U103    S210    NULL    1465535556      1462908262      1468139889      AP      ST407   1       0       0       1       fail
U106    S210    NULL    1475130523      1485130523      1475130523      J       ST403   1       1       1       1       fail
U112    S210    NULL    1465230523      1465230523      1465130523      A       ST411   2       0       0       1       fail
U110    S210    NULL    1465535556      1465535556      1465535556      AP      ST412   0       0       1       1       fail
U108    S200    A300    1465230523      1465230523      1465130523      E       ST404   0       0       0       1       pass
U108    S200    A300    1462908262      1465535556      1468139889      AU      ST401   3       0       1       1       pass
U106    S201    A301    1495130523      1475130523      1465130523      E       ST404   3       0       0       1       pass
U104    S201    A301    1462908262      1468139889      1462908262      AP      ST407   3       0       1       1       pass
U103    S201    A301    1468139889      1465535556      1468139889      AP      ST412   2       1       0       1       pass
U111    S203    A303    1465230523      1465230523      1485130523      A       ST405   3       0       0       1       pass
U106    S203    A303    1465130523      1475130523      1485130523      E       ST409   3       0       1       1       pass
U103    S203    A303    1465130523      1485130523      1485130523      J       ST413   2       1       0       1       pass
U107    S203    A303    1465130523      1475130523      1475130523      A       ST400   1       1       0       1       pass
U110    S204    A304    1494342562      1494342562      1462908262      AU      ST406   2       0       1       1       pass
U111    S204    A304    1495130523      1465230523      1485130523      E       ST409   0       0       0       1       pass
U114    S205    A301    1475130523      1465130523      1475130523      E       ST408   3       0       0       1       pass
U118    S205    A301    1494342562      1468139889      1468139889      AP      ST402   3       1       0       1       pass
U102    S207    A303    1475130523      1465230523      1465230523      E       ST404   2       0       1       1       pass
U118    S208    A304    1462908262      1465535556      1462908262      AU      ST406   2       1       0       1       pass
U110    S209    A305    1494342562      1468139889      1465535556      AU      ST406   0       1       0       1       pass
U112    S209    A305    1465130523      1465230523      1475130523      J       ST403   3       0       0       1       pass
U107    S209    A305    1495130523      1485130523      1465130523      AP      ST412   1       1       0       1       pass
U110    S209    A305    1462908262      1465535556      1462908262      A       ST400   3       0       1       1       pass
U115    S209    A305    1462908262      1462908262      1462908262      AP      ST412   1       0       1       1       pass
U103    S209    A305    1494342562      1462908262      1462908262      A       ST405   1       0       1       1       pass
Time taken: 0.235 seconds, Fetched: 40 row(s)
```

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

In this stage we will do analysis on enriched data using Hive QL and run the program using Hive file **"sh data_analysis.hql"** which is a script line in script file **"data_analysis.sh"** command.


Project - Music Data Analysis
ACADGILD

Script : **"sh data_analysis.sh"**




```
#!/bin/bash

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

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

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

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

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

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

Here the batch file will first edit the log file **“log_batch_1”** and add entry to it as **“Running hive script for data analysis…”**. Then execute the hive file **“data_analysis.hql”** in which analysis on all the Five Queries will start one by one. The Map-Reduce program will show “Success” message after each hive table analysis is finished for all the 5 Queries. Later then it will start the to execute the script file “data_export.sh” in which all the hive tables are exported to **“MySql Database”** using **“Sqoop”.** This script execution is shown r in the later part, refer the screenshots after the data_analysis screenshot are seen. Then the batchid will get incremented by 1 (i.e batchid will become 2).

Project - Music Data Analysis

ACADGILD

Hive script: **"data_analysis.hql"**

data_analysis.hql (~/Downloads/project1/scripts) - gedit

File Edit View Search Tools Documents Help

Open Save Undo

data_analysis.hql

```
SET hive.auto.convert.join=false;
USE project;

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

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


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

INSERT OVERWRITE TABLE users_behaviour
PARTITION(batchid=${hiveconf:batchid})
SELECT
CASE WHEN (su.user_id IS NULL OR CAST(ed.u_timestamp AS DECIMAL(20,0)) > CAST(su.subscn_end_dt AS DECIMAL(20,0))) THEN 'UNSUBSCRIBED'
WHEN (su.user_id IS NOT NULL AND CAST(ed.u_timestamp AS DECIMAL(20,0)) <= CAST(su.subscn_end_dt AS DECIMAL(20,0))) THEN 'SUBSCRIBED'
END AS user_type,
SUM(ABS(CAST(ed.end_ts AS DECIMAL(20,0))-CAST(ed.start_ts AS DECIMAL(20,0)))) AS duration
FROM enriched_data ed
LEFT OUTER JOIN subscribed users su
```

Project - Music Data Analysis

ACADGILD

data_analysis.hql (~/Downloads/project1/scripts) - gedit

File Edit View Search Tools Documents Help

Open Save Undo

data_analysis.hql

```
ON ed.user_id=su.user_id
WHERE ed.status='pass'
AND ed.batchid=${hiveconf:batchid}
GROUP BY CASE WHEN (su.user_id IS NULL OR CAST(ed.u_timestamp AS DECIMAL(20,0)) > CAST(su.subscn_end_dt AS DECIMAL(20,0))) THEN 'UNSUBSCRIBED'
WHEN (su.user_id IS NOT NULL AND CAST(ed.u_timestamp AS DECIMAL(20,0)) <= CAST(su.subscn_end_dt AS DECIMAL(20,0))) THEN 'SUBSCRIBED' END;


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

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


CREATE TABLE IF NOT EXISTS top_10_royalty_songs
(
song_id STRING,
duration INT
)
PARTITIONED BY (batchid INT)
ROW FORMAT DELIMITED
```


Project - Music Data Analysis

ACADGILD

```
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

INSERT OVERWRITE TABLE top_10_royalty_songs
PARTITION(batchid=${hiveconf:batchid})
SELECT song_id,
SUM(ABS(CAST(end_ts AS DECIMAL(20,0))-CAST(start_ts AS DECIMAL(20,0)))) AS duration
FROM enriched_data
WHERE status='pass'
AND batchid=${hiveconf:batchid}
AND (u_like=1 OR song_end_type=0)
GROUP BY song_id
ORDER BY duration DESC
LIMIT 10;


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

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


Project - Music Data Analysis
ACADGILD

Output:

Project - Music Data Analysis

ACADGILD


Project - Music Data Analysis

ACADGILD

The script will run till the analysis of table **"top_10_unsubscribed_users"** is finished and will show a "**success"** message as shown in below

```
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1555173863193_0015, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1555173863193_0015/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1555173863193_0015
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2019-04-13 10:54:22,686 Stage-2 map = 0%,  reduce = 0%
2019-04-13 10:54:38,244 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 2.32 sec
2019-04-13 10:54:56,200 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 5.21 sec
MapReduce Total cumulative CPU time: 5 seconds 210 msec
Ended Job = job_1555173863193_0015
Launching Job 3 out of 3
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1555173863193_0016, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1555173863193_0016/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1555173863193_0016
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 1
2019-04-13 10:55:16,864 Stage-3 map = 0%,  reduce = 0%
2019-04-13 10:55:32,540 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 2.18 sec
2019-04-13 10:55:51,548 Stage-3 map = 100%,  reduce = 100%, Cumulative CPU 6.25 sec
MapReduce Total cumulative CPU time: 6 seconds 250 msec
Ended Job = job_1555173863193_0016
Loading data to table project.top_10_unsubscribed_users partition (batchid=1)
Partition project.top_10_unsubscribed_users{batchid=1} stats: [numFiles=1, numRows=8, totalSize=97, rawDataSize=89]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 1   Cumulative CPU: 12.65 sec   HDFS Read: 21705 HDFS Write: 313 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 5.21 sec   HDFS Read: 4834 HDFS Write: 313 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 6.25 sec   HDFS Read: 6639 HDFS Write: 196 SUCCESS
Total MapReduce CPU Time Spent: 24 seconds 110 msec
OK
Time taken: 191.576 seconds
WARN: The method class org.apache.commons.logging.impl.SLF4JLogFactory#release() was invoked.
```


Project - Music Data Analysis
ACADGILD

Query-1:

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

Query:

```
hive> select * from top_10_stations;
```

Query-2:

Determine total duration of songs played by each type of user, where type of user can be **'subscribed'** or **'unsubscribed'**. An unsubscribed user is the one whose record is either not present in subscribed_users lookup table or has subscription_end_date earlier than the timestamp of the song played by him.

Query:

```
hive> select * users_behaviour;
```

Query-3:

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

Query:

```
hive> select * from connected_artists;
```

Query-4:

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

Query:

```
hive> select * from top_10_royalty_songs;
```

Query-5:

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

Query:

```
hive> select * from top_10_unsubscribed_users;
```


Project - Music Data Analysis
ACADGILD

We can see the hive tables has been created successfully:

hive> show tables;

```
hive> use project;
OK
Time taken: 13.083 seconds
hive> show tables;
OK
connected_artists
enriched_data
formatted_input
song_artist_map
station_geo_map
subscribed_users
top_10_royalty_songs
top_10_stations
top_10_unsubscribed_users
users_artists
users_behaviour
Time taken: 0.701 seconds, Fetched: 11 row(s)
```

The solution for all the queries is shown below :-

```
hive> select * from top_10_stations;
OK
ST412    2         2         1
ST406    2         2         1
ST413    1         1         1
ST402    1         1         1
ST400    1         1         1
Time taken: 0.125 seconds, Fetched: 5 row(s)
hive> select * from users_behaviour;
OK
SUBSCRIBED        52967587          1
UNSUBSCRIBED      96565927          1
Time taken: 0.113 seconds, Fetched: 2 row(s)
hive> select * from connected_artists;
OK
A301     4         1
A303     2         1
A300     1         1
Time taken: 0.117 seconds, Fetched: 3 row(s)
hive> select * from top_10_royalty_songs;
OK
S209     22604333          1
S204     19900000          1
S208     2627294  1
S201     2604333  1
S200     100000   1
S205     0         1
S203     0         1
Time taken: 0.173 seconds, Fetched: 7 row(s)
hive> select * from top_10_unsubscribed_users;
OK
U110     34038633          1
U107     20000000          1
U111     19900000          1
U114     10000000          1
U106     10000000          1
U118     2627294  1
U115     0         1
U103     0         1
Time taken: 0.157 seconds, Fetched: 8 row(s)
```


Project - Music Data Analysis
ACADGILD

Thus data analysis result is successfully created in the above Hive tables in the screen shot.

Now, we need to export all the data to the **MYSQL** using sqoop,

Using the bash file shown below, **data_export.sh** we are going to export the data from the hive tables into mysql using **Sqoop** export.

Script: data_export.sh

Open Save Undo

data_export.sh

```
#!/bin/bash

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

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

mysql -u root < /home/acadgild/project/scripts/create_schema.sql

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

sqoop export \
--connect jdbc:mysql://localhost/project \
--username root -P \
--table top_10_stations \
--export-dir hdfs://localhost:8020/user/hive/warehouse/project.db/top_10_stations/batchid=$batchid \
--input-fields-terminated-by ',' -m 1

sqoop export \
--connect jdbc:mysql://localhost/project \
--username root -P \
--table users_behaviour \
--export-dir hdfs://localhost:8020/user/hive/warehouse/project.db/users_behaviour/batchid=$batchid \
--input-fields-terminated-by ',' -m 1

sqoop export \
--connect jdbc:mysql://localhost/project \
--username root -P \
--table connected_artists \
--export-dir hdfs://localhost:8020/user/hive/warehouse/project.db/connected_artists/batchid=$batchid \
--input-fields-terminated-by ',' -m 1

sqoop export \
--connect jdbc:mysql://localhost/project \
--username root -P \
--table top_10_royalty_songs \
--export-dir hdfs://localhost:8020/user/hive/warehouse/project.db/top_10_royalty_songs/batchid=$batchid \
--input-fields-terminated-by ',' -m 1

sqoop export \
--connect jdbc:mysql://localhost/project \
--username root -P \
--table top_10_unsubscribed_users \
--export-dir hdfs://localhost:8020/user/hive/warehouse/project.db/top_10_unsubscribed_users/batchid=$batchid \
--input-fields-terminated-by ',' -m 1
```


Project - Music Data Analysis
ACADGILD

create_schema.sql

create_schema.sql

```
CREATE DATABASE IF NOT EXISTS project;

USE project;

CREATE TABLE IF NOT EXISTS top_10_stations
(
station_id VARCHAR(50),
total_distinct_songs_played INT,
distinct_user_count INT
);

CREATE TABLE IF NOT EXISTS users_behaviour
(
user_type VARCHAR(50),
duration BIGINT
);

CREATE TABLE IF NOT EXISTS connected_artists
(
artist_id VARCHAR(50),
user_count INT
);

CREATE TABLE IF NOT EXISTS top_10_royalty_songs
(
song_id VARCHAR(50),
duration BIGINT
);

CREATE TABLE IF NOT EXISTS top_10_unsubscribed_users
(
user_id VARCHAR(50),
duration BIGINT
);

commit;
```

Before running the script "sh data_export.sh" make sure that you logged in to MySql. The below schema will create the database and tables in the MySQl.


Project - Music Data Analysis

ACADGILD

Run script : **"sh data_export.sh"**

```
[cloudera@quickstart ~]$ batchid=`cat /home/cloudera/project/logs/current-batch.txt`
[cloudera@quickstart ~]$ LOGFILE=/home/cloudera/project/logs/log_batch_$batchid
[cloudera@quickstart ~]$ sqoop export --connect jdbc:mysql://localhost/project --username root -P  --table top_10_stations --export-dir hdfs://localhost:8020/user/hive/
warehouse/project.db/top_10_stations/batchid=$batchid --input-fields-terminated-by ',' -m 1
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
19/04/13 11:12:53 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
Enter password:
19/04/13 11:13:01 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
19/04/13 11:13:01 INFO tool.CodeGenTool: Beginning code generation
19/04/13 11:13:03 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `top_10_stations` AS t LIMIT 1
19/04/13 11:13:03 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `top_10_stations` AS t LIMIT 1
19/04/13 11:13:03 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-mapreduce
Note: /tmp/sqoop-cloudera/compile/4aa389e2d9617667046d4589a74c2d06/top_10_stations.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
19/04/13 11:13:10 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-cloudera/compile/4aa389e2d9617667046d4589a74c2d06/top_10_stations.jar
19/04/13 11:13:10 INFO mapreduce.ExportJobBase: Beginning export of top_10_stations
19/04/13 11:13:10 INFO Configuration.deprecation: mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
19/04/13 11:13:11 INFO Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job.jar
19/04/13 11:13:15 INFO Configuration.deprecation: mapred.reduce.tasks.speculative.execution is deprecated. Instead, use mapreduce.reduce.speculative
19/04/13 11:13:15 INFO Configuration.deprecation: mapred.map.tasks.speculative.execution is deprecated. Instead, use mapreduce.map.speculative
19/04/13 11:13:15 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Instead, use mapreduce.job.maps
19/04/13 11:13:15 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032

19/04/13 11:13:20 INFO input.FileInputFormat: Total input paths to process : 1
19/04/13 11:13:20 INFO input.FileInputFormat: Total input paths to process : 1
19/04/13 11:13:20 INFO mapreduce.JobSubmitter: number of splits:1
19/04/13 11:13:20 INFO Configuration.deprecation: mapred.map.tasks.speculative.execution is deprecated. Instead, use mapreduce.map.speculative
19/04/13 11:13:21 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1555173863193_0017
19/04/13 11:13:22 INFO impl.YarnClientImpl: Submitted application application_1555173863193_0017
19/04/13 11:13:22 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1555173863193_0017/
19/04/13 11:13:22 INFO mapreduce.Job: Running job: job_1555173863193_0017
19/04/13 11:13:42 INFO mapreduce.Job: Job job_1555173863193_0017 running in uber mode : false
19/04/13 11:13:42 INFO mapreduce.Job:  map 0% reduce 0%
19/04/13 11:13:57 INFO mapreduce.Job:  map 100% reduce 0%
19/04/13 11:13:58 INFO mapreduce.Job: Job job_1555173863193_0017 completed successfully
19/04/13 11:13:59 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=171058
                FILE: Number of read operations=0
```


Project - Music Data Analysis
ACADGILD

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

```
                FILE: Number of bytes read=0
                FILE: Number of bytes written=171058
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=221
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=4
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=0
        Job Counters
                Launched map tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=13052
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=13052
                Total vcore-milliseconds taken by all map tasks=13052
                Total megabyte-milliseconds taken by all map tasks=13365248
        Map-Reduce Framework
                Map input records=5
                Map output records=5
                Input split bytes=168
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=149
                CPU time spent (ms)=1580
                Physical memory (bytes) snapshot=129011712
                Virtual memory (bytes) snapshot=1509097472
                Total committed heap usage (bytes)=60882944
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=0
19/04/13 11:13:59 INFO mapreduce.ExportJobBase: Transferred 221 bytes in 44.1175 seconds (5.0093 bytes/sec)
19/04/13 11:13:59 INFO mapreduce.ExportJobBase: Exported 5 records.
```


Project - Music Data Analysis
ACADGILD

The same output will be shown for other 5 queries and the output will be exported from hdfs to mysql database.

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

The database "**project"** had been exported from the hive and shown in below screenshot

```
mysql> select * from  top_10_stations;
+------------+---------------------------+--------------------+
| station_id | total_distinct_songs_played | distinct_user_count |
+------------+---------------------------+--------------------+
| ST412      |                         2 |                  2 |
| ST406      |                         2 |                  2 |
| ST413      |                         1 |                  1 |
| ST402      |                         1 |                  1 |
| ST400      |                         1 |                  1 |
+------------+---------------------------+--------------------+
5 rows in set (0.00 sec)

mysql> select * from users_behaviour;
+--------------+----------+
| user_type    | duration |
+--------------+----------+
| SUBSCRIBED   | 52967587 |
| UNSUBSCRIBED | 96565927 |
+--------------+----------+
2 rows in set (0.00 sec)

mysql> select * from connected_artists;
+-----------+------------+
| artist_id | user_count |
+-----------+------------+
| A301      |          4 |
| A303      |          2 |
| A300      |          1 |
+-----------+------------+
3 rows in set (0.02 sec)

mysql> select * from top_10_royalty_songs;
+---------+----------+
| song_id | duration |
+---------+----------+
| S209    | 22604333 |
| S204    | 19900000 |
| S208    |  2627294 |
| S201    |  2604333 |
| S200    |   100000 |
| S205    |        0 |
| S203    |        0 |
+---------+----------+
7 rows in set (0.00 sec)
```


```
mysql> select * from top_10_unsubscribed_users;
+---------+----------+
| user_id | duration |
+---------+----------+
| U110    | 34038633 |
| U107    | 20000000 |
| U111    | 19900000 |
| U114    | 10000000 |
| U106    | 10000000 |
| U118    |  2627294 |
| U115    |        0 |
| U103    |        0 |
+---------+----------+
8 rows in set (0.00 sec)
```

Now after exporting data into MySQL **batchid** will be incremented to additional 1 means one batch of data operations is successfully completed and new batch of data will be loaded for the analysis after every 3 hours.

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

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

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


Project - Music Data Analysis
ACADGILD

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

```
[acadgild@localhost logs]$ cat log_batch_1
Starting daemons
Creating LookUp Tables
Populating LookUp Tables
Placing data files from local to HDFS...
Running pig script for data formatting...
Running hive script for formatted data load...
Creating hive tables on top of hbase tables for data enrichment and filtering...
Running hive script for data enrichment and filtering...
Copying valid and invalid records in local file system...
Deleting older valid and invalid records from local file system...
Running hive script for data analysis...
Incrementing batchid...
```

log_batch_1

```
Starting daemons
Creating LookUp Tables
Populating LookUp Tables
Placing data files from local to HDFS...
Running pig script for data formatting...
Running hive script for formatted data load...
Creating hive tables on top of hbase tables for data enrichment and filtering...
Running hive script for data enrichment and filtering...
Copying valid and invalid records in local file system...
Deleting older valid and invalid records from local file system...
Running hive script for data analysis...
Incrementing batchid...
```


Project - Music Data Analysis
ACADGILD

Wrapping all the scripts inside the single script file and scheduling this file to run at the periodic interval of every 3 hours.

Script: **wrapper.sh**

wrapper.sh (~/Downloads/project1/scripts) - gedit

File Edit View Search Tools Documents Help

Open Save Undo

wrapper.sh

```
#!/bin/bash

python /home/cloudera/project/scripts/generate_web_data.py

python /home/cloudera/project/scripts/generate_mob_data.py

sh /home/cloudera/project/scripts/start-daemons.sh

sh /home/cloudera/project/scripts/populate-lookup.sh

sh /home/cloudera/project/scripts/dataformatting.sh

sh /home/cloudera/project/scripts/data_enrichment.sh

sh /home/cloudera/project/scripts/data_analysis.sh
```

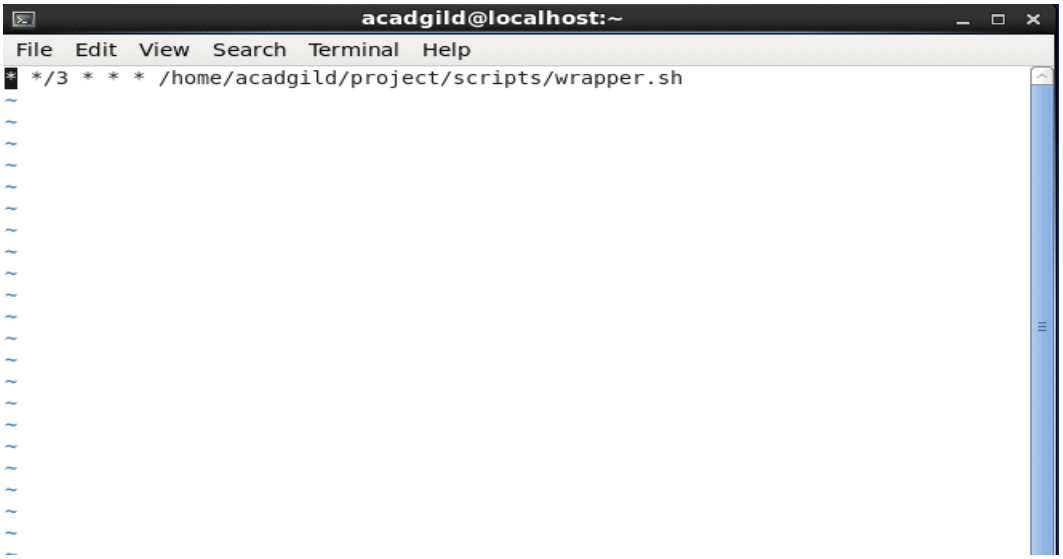
The wrapper.sh will be running for every 3 hours as per the job scheduling done below, as per the above order the wrapper.sh will run the scripts.


Project - Music Data Analysis
ACADGILD

Job Scheduling using Crontab

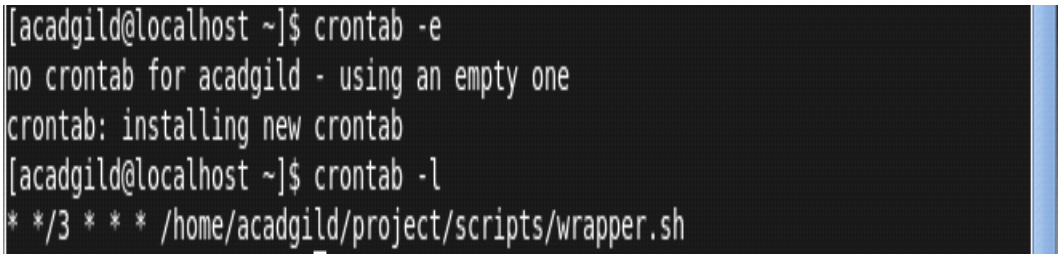
Creating **Crontab** to schedule the wrapper.sh script to run for every 3 hour interval.
So using the below commands as shown in fig we can schedule our script "wrapper.sh"

Code: crontab -e



As soon as we enter the above code we need to enter the below code in the script and save the crontab. As soon as the crontab script is saved it will start installing the new crontab as shown in the fig

Code: * */3 * * * /home/acadgild/project/scripts/wrapper.sh



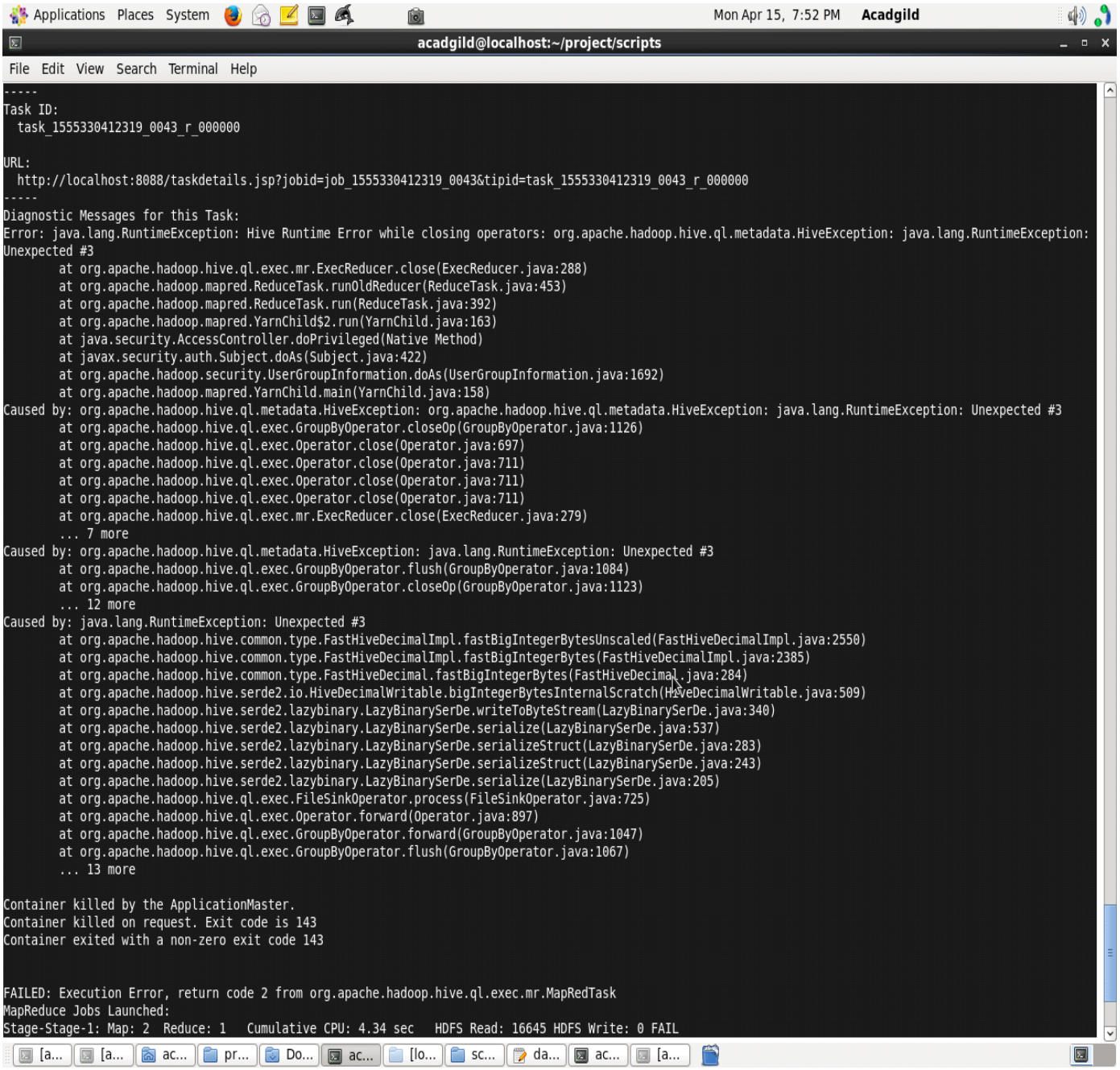
The **crontab** job scheduler will run the **wrappr.sh** every 3 hours and for every 3 hours we will get incremental batch ID's. **Hence, as per the request this job scheduling has been done**


Project - Music Data Analysis
ACADGILD

Problems Faced During Project Execution:

While running the script file "sh data_analysis.sh" got error while getting result for the analysis of table creation for the query **"top_10_unsubscribed_users"**

Error:-




Project - Music Data Analysis
ACADGILD

This error arrived because of container failure or space issue. To solve this error we need to clear all the temp files from the directory and unwanted logs. Also delete the non required files which will give more space for the memory allocation of the above script to run successfully or else restart the services and do all the scripts from the beginning. In my case I still got the same error. So I used the "Cloudera Distributions" to run the whole project script. Also I got the same error while using the script file
"sh data_export.sh" .

After using the Cloudera Distributions I got the script running successfully as shown below:

```
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1555173863193_0015, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1555173863193_0015/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1555173863193_0015
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2019-04-13 10:54:22,686 Stage-2 map = 0%,  reduce = 0%
2019-04-13 10:54:38,244 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 2.32 sec
2019-04-13 10:54:56,200 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 5.21 sec
MapReduce Total cumulative CPU time: 5 seconds 210 msec
Ended Job = job_1555173863193_0015
Launching Job 3 out of 3
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1555173863193_0016, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1555173863193_0016/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1555173863193_0016
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 1
2019-04-13 10:55:16,864 Stage-3 map = 0%,  reduce = 0%
2019-04-13 10:55:32,540 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 2.18 sec
2019-04-13 10:55:51,548 Stage-3 map = 100%,  reduce = 100%, Cumulative CPU 6.25 sec
MapReduce Total cumulative CPU time: 6 seconds 250 msec
Ended Job = job_1555173863193_0016
Loading data to table project.top_10_unsubscribed_users partition (batchid=1)
Partition project.top_10_unsubscribed_users{batchid=1} stats: [numFiles=1, numRows=8, totalSize=97, rawDataSize=89]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 1   Cumulative CPU: 12.65 sec   HDFS Read: 21705 HDFS Write: 313 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 5.21 sec   HDFS Read: 4834 HDFS Write: 313 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 6.25 sec   HDFS Read: 6639 HDFS Write: 196 SUCCESS
Total MapReduce CPU Time Spent: 24 seconds 110 msec
OK
Time taken: 191.576 seconds
WARN: The method class org.apache.commons.logging.impl.SLF4JLogFactory#release() was invoked.
```


Project - Music Data Analysis
ACADGILD

Highlights of the Project

- No join of query is used while analysis. Data is already enriched with new fields and using broadcast maps on Lookup tables so as to avoid any join.
- We used full automated bash scripts from start to end.

Project End Conclusion

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