

MUSIC

DATA ANALYSIS USING

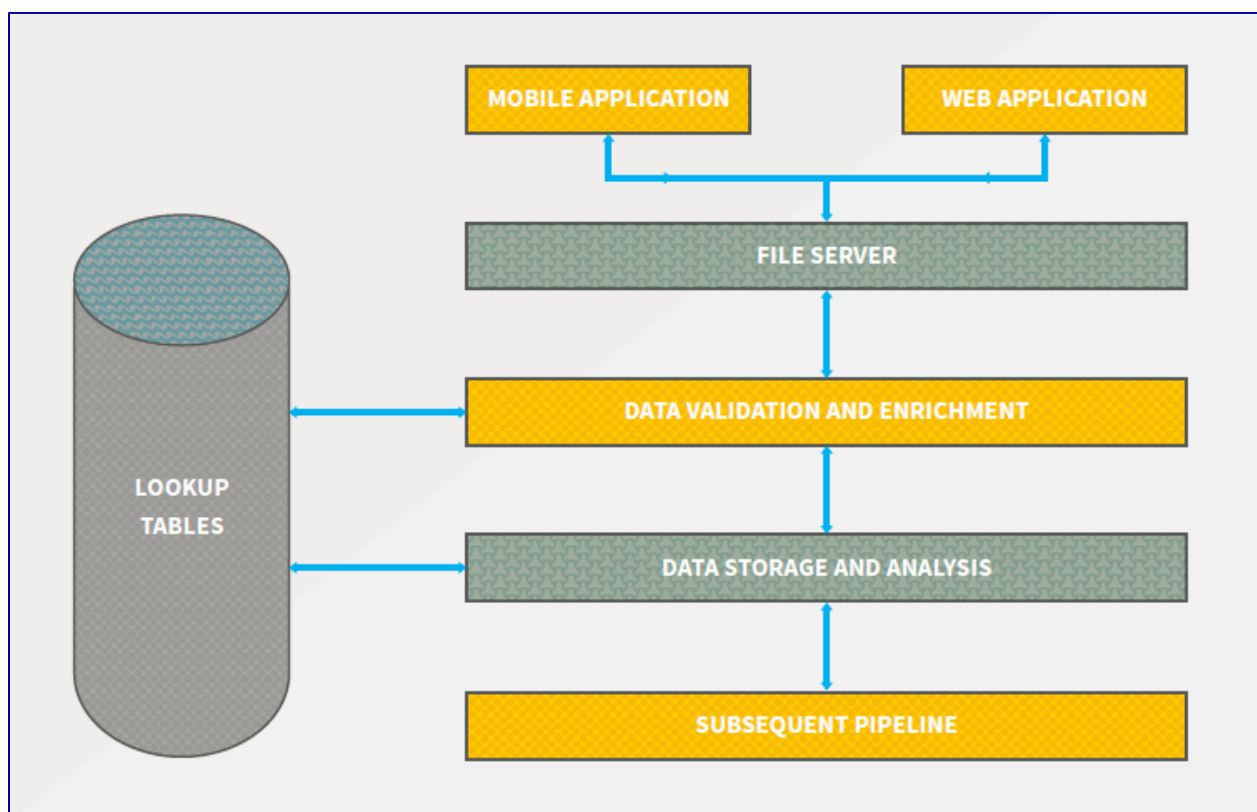
HADOOP

i. Project Description

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 aer every 3 hours.

ii. Architecture:

The below diagram provides a high level architecture of music data analysis using Hadoop platform,



iii. Database Definition:

The main data table will have the definition as below,

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
Time_stamp	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.
Song_Like	0 means song was not liked song was played 1 means song was liked
Song_Dislike	0 means song was not disliked 1 means song was disliked

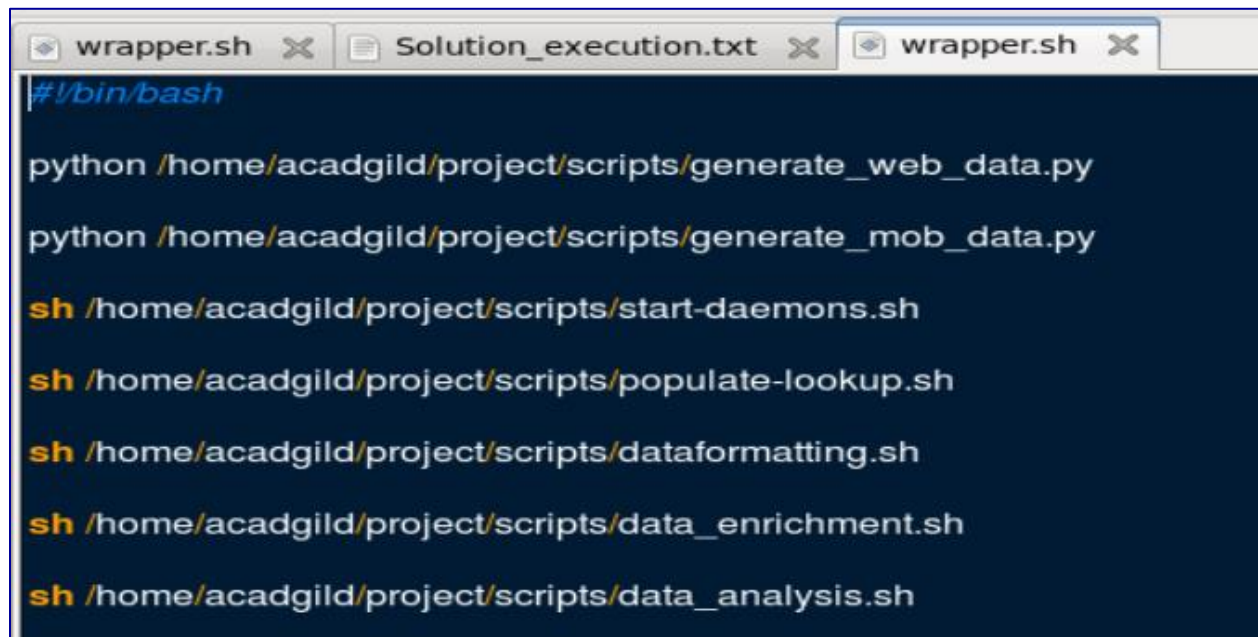
There is some existing look up tables present in NoSQL databases. They play an important role in data enrichment and analysis.

Table Name	Table 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 along with royalty associated with each play of the song
User_Artist_Map	Contains an array of artist_id(s) followed by a user_id

iv. Execution

The data analysis is performed through various stages. In order to execute all the stages in sequence, a master bash script is created and that will be executed, which will trigger all the processes in sequential manner,

File: wrapper.sh



```
#!/bin/bash

python /home/acadgild/project/scripts/generate_web_data.py
python /home/acadgild/project/scripts/generate_mob_data.py
sh /home/acadgild/project/scripts/start-daemons.sh
sh /home/acadgild/project/scripts/populate-lookup.sh
sh /home/acadgild/project/scripts/dataformatting.sh
sh /home/acadgild/project/scripts/data_enrichment.sh
sh /home/acadgild/project/scripts/data_analysis.sh
```

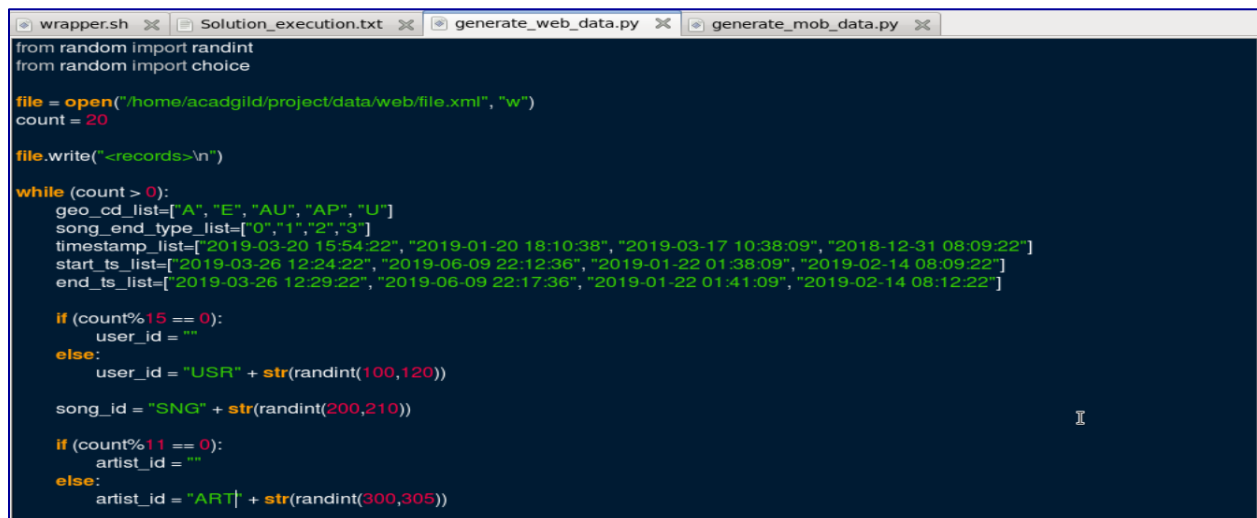
v. Data Ingestion & Initial Validation:

The data for the analysis are ingested from web and mobile based applications. Below are the rules for initial data ingestion and data filtering,

- Data coming from web applications reside in /data/web and has xml format.
- Data coming from mobile applications reside in /data/mob and has csv format.
- Data files come every 3 hours.
- All the timestamp fields in data coming from web application is of the format YYYY-MM-DD HH:MM:SS.
- All the timestamp fields in data coming from mobile application is a long integer interpreted as UNIX timestamps.
- Finally, all timestamps must have the format of a long integer to be interpreted as UNIX timestamps.
- If both like and dislike are 1, consider that record to be invalid.
- If any of the fields from User_id, Song_id, Timestamp, Start_ts, End_ts, Geo_cd is NULL or absent, consider that record to be invalid.
- If Song_end_type is NULL or absent, treat it to be 3
- Create a temporary identifier for all the data files received in the last 3 hours (may be an integer batch_id which is auto incremented or a string obtained after combining current date and current hour, to keep track of valid and invalid records per batch).

For test data ingestion, the data is dynamically created with the help of python scripts for both web and mobile applications in the appropriate formats.

Web Data: generate_web_data.py

A screenshot of a code editor window with multiple tabs. The active tab is 'generate_web_data.py'. The code is written in Python and generates XML data. It imports 'randint' and 'choice' from the 'random' module. It opens a file at '/home/acadgild/project/data/web/file.xml' in write mode. It sets a 'count' variable to 20 and writes an opening XML tag '<records>'. It then enters a 'while' loop that runs as long as 'count' is greater than 0. Inside the loop, it defines lists for 'geo_cd_list', 'song_end_type_list', 'timestamp_list', 'start_ts_list', and 'end_ts_list'. It then uses 'if' statements to generate 'user_id', 'song_id', and 'artist_id' by concatenating a prefix with a random integer. The code is as follows:

```
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=["2019-03-20 15:54:22", "2019-01-20 18:10:38", "2019-03-17 10:38:09", "2018-12-31 08:09:22"]
    start_ts_list=["2019-03-26 12:24:22", "2019-06-09 22:12:36", "2019-01-22 01:38:09", "2019-02-14 08:09:22"]
    end_ts_list=["2019-03-26 12:29:22", "2019-06-09 22:17:36", "2019-01-22 01:41:09", "2019-02-14 08:12:22"]

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

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

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

    count = count - 1
```

```

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

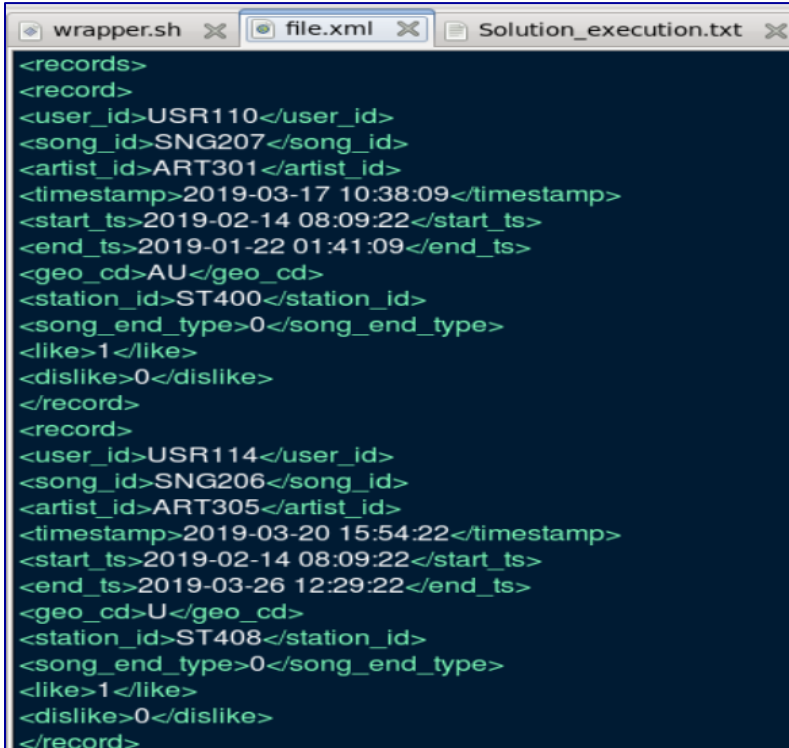
station_id = "ST" + str(randint(400,415))
song_end_type = choice(song_end_type_list)
like = str(randint(0,1))
dislike = str(randint(0,1))
file.write("<record>\n")
file.write("<user_id>%s</user_id>\n" % (user_id))
file.write("<song_id>%s</song_id>\n" % (song_id))
file.write("<artist_id>%s</artist_id>\n"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

```

Data is generated successfully in XML format as expected,



```

<records>
<record>
<user_id>USR110</user_id>
<song_id>SNG207</song_id>
<artist_id>ART301</artist_id>
<timestamp>2019-03-17 10:38:09</timestamp>
<start_ts>2019-02-14 08:09:22</start_ts>
<end_ts>2019-01-22 01:41:09</end_ts>
<geo_cd>AU</geo_cd>
<station_id>ST400</station_id>
<song_end_type>0</song_end_type>
<like>1</like>
<dislike>0</dislike>
</record>
<record>
<user_id>USR114</user_id>
<song_id>SNG206</song_id>
<artist_id>ART305</artist_id>
<timestamp>2019-03-20 15:54:22</timestamp>
<start_ts>2019-02-14 08:09:22</start_ts>
<end_ts>2019-03-26 12:29:22</end_ts>
<geo_cd>U</geo_cd>
<station_id>ST408</station_id>
<song_end_type>0</song_end_type>
<like>1</like>
<dislike>0</dislike>
</record>

```

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=["1547115260", "1549071270", "1545308110", "1546161506"]
    start_ts_list=["1548982861", "1549188000", "1545457500", "1546299600"]
    end_ts_list=["1548983100", "1549188300", "1545457800", "1546300200"]

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

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

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

    timestamp = choice(timestamp_list)
    start_ts = choice(start_ts_list)
```

```
        artist_id = ""
    else:
        artist_id = "ART" + 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)

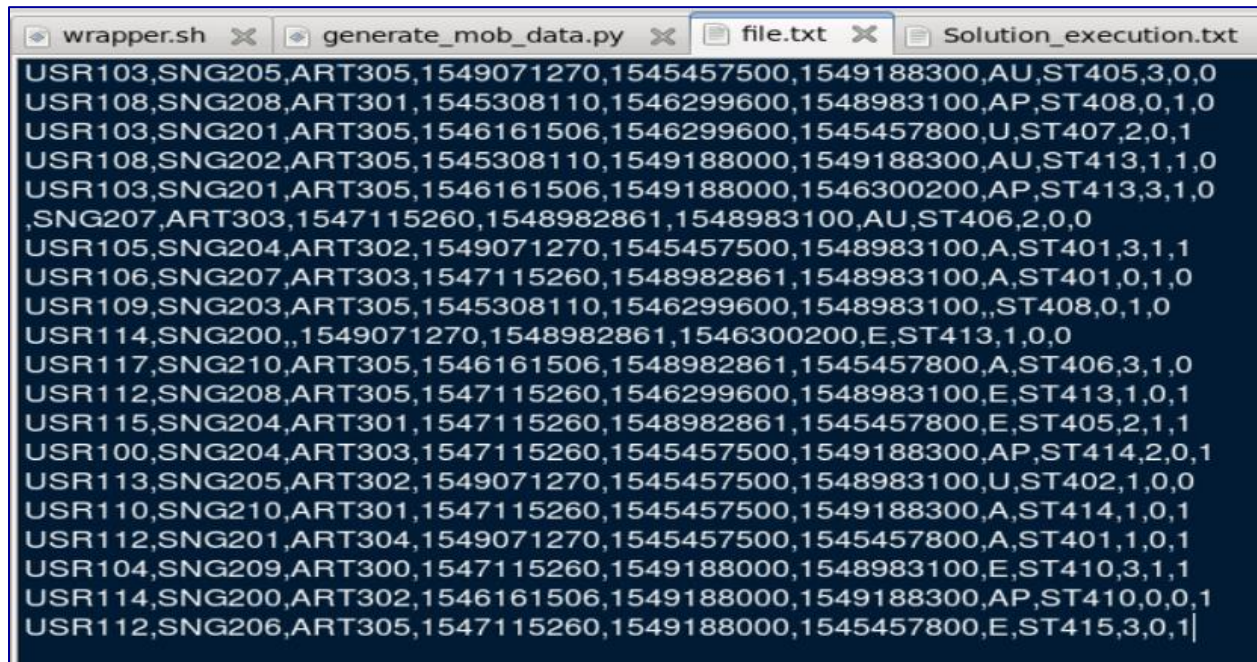
    station_id = "ST" + str(randint(400,415))
    song_end_type = choice(song_end_type_list)
    like = str(randint(0,1))
    dislike = str(randint(0,1))

    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()
```


Data file is created successfully with comma (,) delimited as expected,

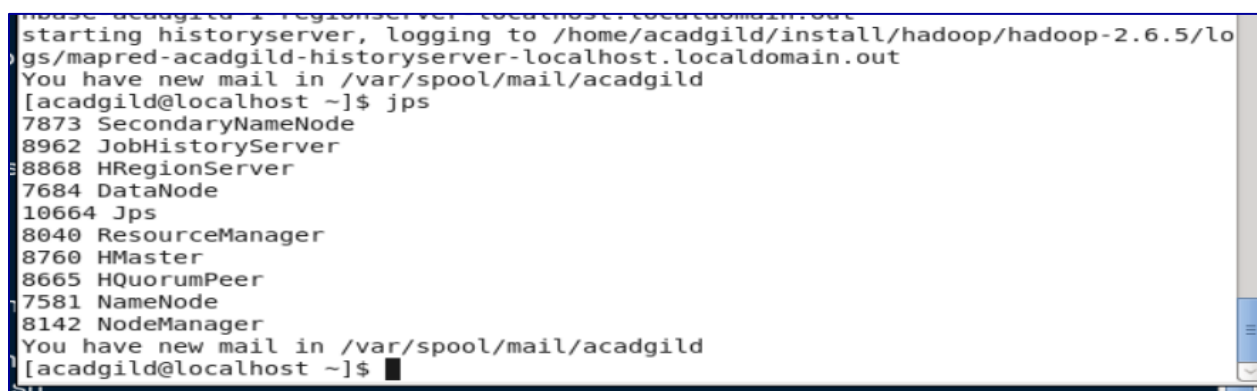


```
wrapper.sh generate_mob_data.py file.txt Solution_execution.txt
USR103,SNG205,ART305,1549071270,1545457500,1549188300,AU,ST405,3,0,0
USR108,SNG208,ART301,1545308110,1546299600,1548983100,AP,ST408,0,1,0
USR103,SNG201,ART305,1546161506,1546299600,1545457800,U,ST407,2,0,1
USR108,SNG202,ART305,1545308110,1549188000,1549188300,AU,ST413,1,1,0
USR103,SNG201,ART305,1546161506,1549188000,1546300200,AP,ST413,3,1,0
,SNG207,ART303,1547115260,1548982861,1548983100,AU,ST406,2,0,0
USR105,SNG204,ART302,1549071270,1545457500,1548983100,A,ST401,3,1,1
USR106,SNG207,ART303,1547115260,1548982861,1548983100,A,ST401,0,1,0
USR109,SNG203,ART305,1545308110,1546299600,1548983100,,ST408,0,1,0
USR114,SNG200,,1549071270,1548982861,1546300200,E,ST413,1,0,0
USR117,SNG210,ART305,1546161506,1548982861,1545457800,A,ST406,3,1,0
USR112,SNG208,ART305,1547115260,1546299600,1548983100,E,ST413,1,0,1
USR115,SNG204,ART301,1547115260,1548982861,1545457800,E,ST405,2,1,1
USR100,SNG204,ART303,1547115260,1545457500,1549188300,AP,ST414,2,0,1
USR113,SNG205,ART302,1549071270,1545457500,1548983100,U,ST402,1,0,0
USR110,SNG210,ART301,1547115260,1545457500,1549188300,A,ST414,1,0,1
USR112,SNG201,ART304,1549071270,1545457500,1545457800,A,ST401,1,0,1
USR104,SNG209,ART300,1547115260,1549188000,1548983100,E,ST410,3,1,1
USR114,SNG200,ART302,1546161506,1549188000,1549188300,AP,ST410,0,0,1
USR112,SNG206,ART305,1547115260,1549188000,1545457800,E,ST415,3,0,1
```

vi. Start Daemons:

Script: start_daemons.sh

The script will start HDFS name node, data node, resource manager, history server and HBase instances,



```
starting historyserver, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/lo
gs/mapred-acadgild-historyserver-localhost.localdomain.out
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ jps
7873 SecondaryNameNode
8962 JobHistoryServer
8868 HRegionServer
7684 DataNode
10664 Jps
8040 ResourceManager
8760 HMaster
8665 HQuorumPeer
7581 NameNode
8142 NodeManager
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$
```


vii. Populate Lookup Data:

Script: populate-lookup.sh

The script populate all necessary lookup tables in HBASE and HIVE databases,

```
#!/bin/bash

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

LOGFILE=/home/acadgild/project/logs/log_batch_${batchid}

echo "Creating LookUp Tables" >> $LOGFILE

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

echo "Populating LookUp Tables" >> $LOGFILE

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

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


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

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

Upon the execution of this script, lookup tables are created successfully in both HBase and Hive,

Tables song-artist-map, station-geo-map, subscribed-users are created in HBase database,

```
hbase(main):008:0> list
TABLE
EMPLOYEES
bulktable
clicks
song-artist-map
station-geo-map
subscribed-users
6 row(s) in 0.0170 seconds
```



User and Artists mapping data has been loaded in to Hive table "users_artists"

```
Time taken: 0.058 seconds
OK
Time taken: 0.994 seconds
Loading data to table project.users_artists
OK
Time taken: 1.986 seconds
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$
```

HBase shell> scan 'song-artist-map'

```
hbase(main):010:0> scan 'song-artist-map'
ROW
COLUMN+CELL
SNG200 column=artist:artistid, timestamp=1553432540669, value=ART300
SNG201 column=artist:artistid, timestamp=1553432554353, value=ART301
SNG202 column=artist:artistid, timestamp=1553432576290, value=ART302
SNG203 column=artist:artistid, timestamp=1553432589289, value=ART303
SNG204 column=artist:artistid, timestamp=1553432602375, value=ART304
SNG205 column=artist:artistid, timestamp=1553432615566, value=ART301
SNG206 column=artist:artistid, timestamp=1553432627751, value=ART302
SNG207 column=artist:artistid, timestamp=1553432639910, value=ART303
SNG208 column=artist:artistid, timestamp=1553432652487, value=ART304
SNG209 column=artist:artistid, timestamp=1553432664737, value=ART305
10 row(s) in 0.0440 seconds
hbase(main):011:0>
```

HBase shell> scan 'station-geo-map'

```

hbase(main):011:0> scan 'station-geo-map'
ROW COLUMN+CELL
ST400 column=geo:geo_cd, timestamp=1553432354315, value=A
ST401 column=geo:geo_cd, timestamp=1553432365881, value=AU
ST402 column=geo:geo_cd, timestamp=1553432377457, value=AP
ST403 column=geo:geo_cd, timestamp=1553432389733, value=J
ST404 column=geo:geo_cd, timestamp=1553432403011, value=E
ST405 column=geo:geo_cd, timestamp=1553432417249, value=A
ST406 column=geo:geo_cd, timestamp=1553432431854, value=AU
ST407 column=geo:geo_cd, timestamp=1553432443471, value=AP
ST408 column=geo:geo_cd, timestamp=1553432455363, value=E
ST409 column=geo:geo_cd, timestamp=1553432467547, value=E
ST410 column=geo:geo_cd, timestamp=1553432479706, value=A
ST411 column=geo:geo_cd, timestamp=1553432492212, value=A
ST412 column=geo:geo_cd, timestamp=1553432504577, value=AP
ST413 column=geo:geo_cd, timestamp=1553432516150, value=J
ST414 column=geo:geo_cd, timestamp=1553432528459, value=E
15 row(s) in 0.0700 seconds

hbase(main):012:0> █

```

HBase shell> scan 'subscribed-users'

```

acadgild@localhost:~/install/hbase/hbase-1.2.6/bin
File Edit View Search Terminal Help
hbase(main):013:0> scan 'subscribed-users'
ROW COLUMN+CELL
USR100 column=subscn:enddt, timestamp=1553432689859, value=1553432023
USR100 column=subscn:startdt, timestamp=1553432676880, value=1553432023
USR101 column=subscn:enddt, timestamp=1553432722578, value=1553432023
USR101 column=subscn:startdt, timestamp=1553432702781, value=1553432023
USR102 column=subscn:enddt, timestamp=1553432749156, value=1553432023
USR102 column=subscn:startdt, timestamp=1553432736857, value=1553432023
USR103 column=subscn:enddt, timestamp=1553432773915, value=1553432023
USR103 column=subscn:startdt, timestamp=1553432761389, value=1553432023
USR104 column=subscn:enddt, timestamp=1553432799210, value=1553432023
USR104 column=subscn:startdt, timestamp=1553432786845, value=1553432023
USR105 column=subscn:enddt, timestamp=1553432825485, value=1553432023
USR105 column=subscn:startdt, timestamp=1553432812576, value=1553432023
USR106 column=subscn:enddt, timestamp=1553432852027, value=1553432023
USR106 column=subscn:startdt, timestamp=1553432838905, value=1553432023
USR107 column=subscn:enddt, timestamp=1553432878438, value=1553432023
USR107 column=subscn:startdt, timestamp=1553432864695, value=1553432023
USR108 column=subscn:enddt, timestamp=1553432902473, value=1553432023
USR108 column=subscn:startdt, timestamp=1553432890536, value=1553432023
USR109 column=subscn:enddt, timestamp=1553432926594, value=1553432023
USR109 column=subscn:startdt, timestamp=1553432914491, value=1553432023
USR110 column=subscn:enddt, timestamp=1553432950190, value=1553432023

```

viii. Data Formatting:

Script: dataformatting.sh

Once lookup tables are populated, data validation is part of populate-lookup bash script and validate, data enrichment (formatting) will be performed based on below rules,

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

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
JobId      Maps      Reduces  MaxMapTime  MinMapTime  AvgMapTime  MedianMapTime  Alias  
pTime      MaxReduceTime  MinReduceTime  AvgReduceTime  MedianReducetime  A  
lias      Feature  Outputs  
job_1553429169058_0001 1      0      9      9      9      9      0      0  
0      0      A,B      MAP_ONLY      /user/acadgild/project/batch1/formattedweb,  
eb,  
  
Input(s):  
Successfully read 20 records (7227 bytes) from: "/user/acadgild/project/batch1/web"  
  
Output(s):  
Successfully stored 20 records (1357 bytes) in: "/user/acadgild/project/batch1/formattedweb"  
  
Counters:  
Total records written : 20  
Total bytes written : 1357  
Spillable Memory Manager spill count : 0  
Total bags proactively spilled: 0  
Total records proactively spilled: 0  
  
Job DAG:  
job_1553429169058_0001
```

```
Time taken: 0.565 seconds, Fetched: 4 row(s)  
(hive> use project;  
OK  
Time taken: 0.033 seconds  
(hive> show tables;  
OK  
users_artists  
Time taken: 0.083 seconds, Fetched: 1 row(s)  
(hive> show tables;  
OK  
formatted input  
users_artists  
Time taken: 0.074 seconds, Fetched: 2 row(s)
```

```
acadgild@localhost:~/install/hive/apache-hive-2.3.2-bin/bin  
File Edit View Search Terminal Help  
Time taken: 0.482 seconds, Fetched: 40 row(s)  
(hive> select * from formatted_input;  
OK  
USR103  SNG205  ART305  1549071270  1545457500  1549188300  AU  ST405  3  0  0  1  
USR108  SNG208  ART301  1545308110  1546299600  1548983100  AP  ST408  0  1  0  1  
USR103  SNG201  ART305  1546161506  1546299600  1545457800  U  ST407  2  0  1  1  
USR108  SNG202  ART305  1545308110  1549188000  1549188300  AU  ST413  1  1  0  1  
USR103  SNG201  ART305  1546161506  1549188000  1546300200  AP  ST413  3  1  0  1  
SNG207  ART303  1547115260  1548982861  1548983100  AU  ST406  2  0  0  1  
USR105  SNG204  ART302  1549071270  1545457500  1548983100  A  ST401  3  1  1  1  
USR106  SNG207  ART303  1547115260  1548982861  1548983100  A  ST401  0  1  0  1  
USR109  SNG203  ART305  1545308110  1546299600  1548983100  A  ST408  0  1  0  1  
SNG200  ART301  1549071270  1548982861  1546300200  E  ST413  1  0  0  1  
USR117  SNG210  ART305  1546161506  1548982861  1545457800  A  ST406  3  1  0  1  
USR112  SNG208  ART305  1547115260  1546299600  1548983100  E  ST413  1  0  1  1  
USR115  SNG204  ART301  1547115260  1548982861  1545457800  E  ST405  2  1  1  1  
USR100  SNG204  ART303  1547115260  1545457500  1549188300  AP  ST414  2  0  1  1  
USR113  SNG205  ART302  1549071270  1545457500  1548983100  U  ST402  1  0  0  1  
USR110  SNG210  ART301  1547115260  1545457500  1549188300  A  ST414  1  0  1  1  
USR112  SNG201  ART304  1549071270  1545457500  1545457800  A  ST401  1  0  1  1  
USR104  SNG209  ART300  1547115260  1549188000  1548983100  E  ST410  3  1  1  1  
USR114  SNG200  ART302  1546161506  1549188000  1549188300  AP  ST410  0  0  1  1  
USR112  SNG206  ART305  1547115260  1549188000  1545457800  E  ST415  3  0  1  1  
USR110  SNG207  ART301  1552799289  1550111962  1548101469  AU  ST400  0  1  0  1  
USR114  SNG206  ART305  1553077462  1550111962  1553583562  U  ST408  0  1  0  1  
USR116  SNG208  ART304  1552799289  1560098556  1548101469  AU  ST404  3  1  0  1  
USR113  SNG209  ART302  1553077462  1548101289  1548101469  E  ST408  2  0  1  1  
USR105  SNG207  ART300  1552799289  1560098556  1548101469  AU  ST413  0  0  0  1  
SNG201  ART304  1552799289  1550111962  1560098856  E  ST410  2  1  0  1  
USR120  SNG207  ART302  1547988038  1553583262  1548101469  AU  ST400  3  1  0  1  
USR104  SNG202  ART300  1547988038  1550111962  1550112142  AU  ST409  3  1  0  1  
USR115  SNG204  ART301  1553077462  1560098556  1550112142  AP  ST402  2  0  0  1
```

Load lookup tables from HBase to Hive database "project"

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

```
Logging initialized using configuration in jar:file:/
apache-hive-2.3.2-bin/lib/hive-common-2.3.2.jar!/hive
rue
t OK
Time taken: 8.878 seconds
OK
Time taken: 3.533 seconds
OK
Time taken: 0.397 seconds
OK
Time taken: 0.341 seconds
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$
```

Lookup tables are loaded successfully in to Hive database,

```
Time taken: 0.552 seconds, Fetched: 40 row(s)
hive> show tables;
OK
formatted_input
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 0.098 seconds, Fetched: 5 row(s)
```

ix. Data Enrichment:

Script: *data_enrichment.sh*

```
wrapper.sh x Solution_execution.txt x data_enrichment.sh x *Unsaved Document 1 x
#!/bin/bash

batchid=`cat /home/acadgild/project/logs/current-batch.txt`
LOGFILE=/home/acadgild/project/logs/log_batch_${batchid}
VALIDDIR=/home/acadgild/project/processed_dir/valid/batch_${batchid}
INVALIDDIR=/home/acadgild/project/processed_dir/invalid/batch_${batchid}

echo "Running hive script for data enrichment and filtering..." >> $LOGFILE

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

if [ ! -d "$VALIDDIR" ]
then
mkdir -p "$VALIDDIR"
fi

if [ ! -d "$INVALIDDIR" ]
then
mkdir -p "$INVALIDDIR"
fi

echo "Copying valid and invalid records in local file system..." >> $LOGFILE

hadoop fs -get /user/hive/warehouse/project.db/enriched_data/batchid=$batchid/status=pass/* $VALIDDIR
hadoop fs -get /user/hive/warehouse/project.db/enriched_data/batchid=$batchid/status=fail/* $INVALIDDIR

echo "Deleting older valid and invalid records from local file system..." >> $LOGFILE
```

Internally it calls data_enrichment.hql hive script to perform data enrichment activity,

Hive script: *data_enrichment.hql*

```
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,
Time_stamp STRING,
Start_ts STRING,
End_ts STRING,
Geo_cd STRING,
Station_id STRING,
Song_end_type INT,
Song_Like INT,
Song_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.time_stamp,
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.song_like IS NULL, 0, i.song_like) AS song_like,
IF (i.song_dislike IS NULL, 0, i.song_dislike) AS song_dislike,
i.batchid,
IF((i.song_like=1 AND i.song_dislike=1)
OR i.user_id IS NULL
OR i.song_id IS NULL
OR i.time_stamp 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.time_stamp=""
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};

```

The task has completed successfully,

```

File Edit View Search Terminal Help
Ended Job = job_1553429169058_0003
Loading data to table project.enriched_data partition (batchid=null, status=null)

Time taken to load dynamic partitions: 1.048 seconds
Time taken for adding to write entity : 0.006 seconds
MapReduce Jobs Launched:
Stage-Stage-1: Map: 3 Reduce: 1 Cumulative CPU: 8.31 sec HDFS Read: 49901 H
DFS Write: 3249 SUCCESS
Stage-Stage-2: Map: 2 Reduce: 1 Cumulative CPU: 8.5 sec HDFS Read: 24380 HD
FS Write: 3287 SUCCESS
Total MapReduce CPU Time Spent: 16 seconds 810 msec
OK
Time taken: 115.777 seconds
19/03/24 19:55:28 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
get: '/home/acadgild/project/processed_dir/valid/batch_1/000000_0': File exists
19/03/24 19:55:31 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable

```


The table "enriched_data" is also created successfully,

```
Time taken: 0.098 seconds, Fetched: 5 row(s)
hive> show tables;
OK
enriched_data
formatted_input
song_artist_map
station_geo_map
subscribed_users
users_artists
Time taken: 0.494 seconds, Fetched: 6 row(s)
hive>
```

Select the table data to validate the enriched data exist or not,

acadgild@localhost:~/install/hive/apache-hive-2.3.2-bin/bin

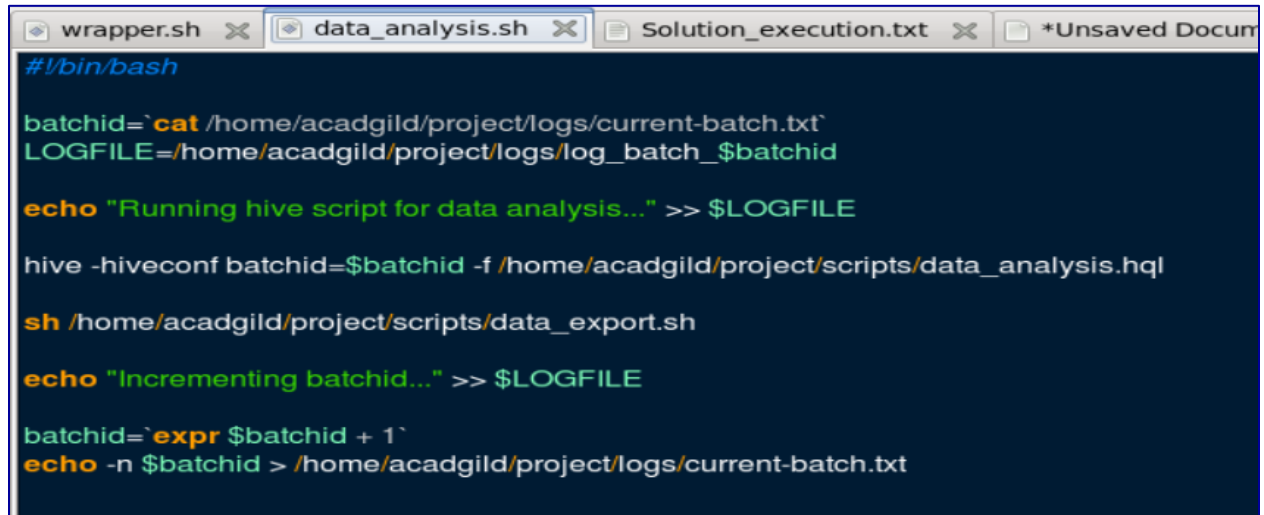
File Edit View Search Terminal Help

hive> select * from enriched_data;
OK

USR104	SNG200	ART300	1553077462	1560098556	1560098856	AP	ST407	1	1	1	1	1	1	1	1	fail
	SNG201	ART301	1552799289	1550111962	1560098856	A	ST410	2	1	0	1	0	1	1	1	fail
USR109	SNG203	ART303	1545308110	1546299600	1548983100	E	ST408	0	1	0	1	0	1	1	1	fail
USR115	SNG204	ART304	1547115260	1548982861	1545457800	A	ST405	2	1	1	1	1	1	1	1	fail
USR103	SNG204	ART304	1546223962	1550111962	1553583562	NULL	ST415	1	0	1	1	1	1	1	1	fail
USR105	SNG204	ART304	1549071270	1545457500	1548983100	AU	ST401	3	1	1	1	1	1	1	1	fail
USR115	SNG204	ART304	1553077462	1560098556	1550112142	AP	ST402	2	0	0	0	1	1	1	1	fail
USR112	SNG206	ART302	1547115260	1549188000	1545457800	NULL	ST415	3	0	1	1	1	1	1	1	fail
	SNG207	ART303	1547115260	1548982861	1548983100	AU	ST406	2	0	0	0	1	1	1	1	fail
USR104	SNG209	ART305	1547115260	1549188000	1548983100	A	ST410	3	1	1	1	1	1	1	1	fail
USR110	SNG210	NULL	1547115260	1545457500	1549188300	E	ST414	1	0	1	1	1	1	1	1	fail
USR117	SNG210	NULL	1546161506	1548982861	1545457800	AU	ST406	3	1	0	1	0	1	1	1	fail
USR102	SNG200	ART300	1546223962	1550111962	1548101469	AP	ST412	2	0	0	0	1	1	1	1	pass
USR114	SNG200	ART300	1546161506	1549188000	1549188300	A	ST410	0	0	1	1	1	1	1	1	pass
USR105	SNG200	ART300	1552799289	1548101289	1548101469	E	ST404	2	1	0	1	0	1	1	1	pass
USR114	SNG200	ART300	1549071270	1548982861	1546300200	J	ST413	1	0	0	1	0	1	1	1	pass
USR103	SNG201	ART301	1546161506	1546299600	1545457800	AP	ST407	2	0	1	1	1	1	1	1	pass
USR112	SNG201	ART301	1549071270	1545457500	1545457800	AU	ST401	1	0	1	1	1	1	1	1	pass
USR103	SNG201	ART301	1546161506	1549188000	1546300200	J	ST413	3	1	0	1	0	1	1	1	pass
USR103	SNG202	ART302	1547988038	1553583262	1560098856	A	ST400	1	0	1	1	1	1	1	1	pass
USR108	SNG202	ART302	1545308110	1549188000	1549188300	J	ST413	1	1	0	1	0	1	1	1	pass
USR104	SNG202	ART302	1547988038	1550111962	1550112142	E	ST409	3	1	0	1	0	1	1	1	pass
USR116	SNG202	ART302	1552799289	1560098556	1553583562	AP	ST407	2	0	0	0	1	1	1	1	pass
USR119	SNG202	ART302	1552799289	1550111962	1560098856	E	ST404	0	0	0	0	1	1	1	1	pass
USR100	SNG204	ART304	1547115260	1545457500	1549188300	E	ST414	2	0	1	1	1	1	1	1	pass
USR116	SNG204	ART304	1553077462	1553583262	1548101469	AU	ST401	2	1	0	1	0	1	1	1	pass
USR103	SNG205	ART301	1549071270	1545457500	1549188300	A	ST405	3	0	0	0	1	1	1	1	pass
USR113	SNG205	ART301	1549071270	1545457500	1548983100	AP	ST402	1	0	0	0	1	1	1	1	pass
USR114	SNG206	ART302	1553077462	1550111962	1553583562	E	ST408	0	1	0	1	0	1	1	1	pass
USR110	SNG207	ART303	1552799289	1550111962	1548101469	A	ST400	0	1	0	1	0	1	1	1	pass

x. Data Analysis:

Script: *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

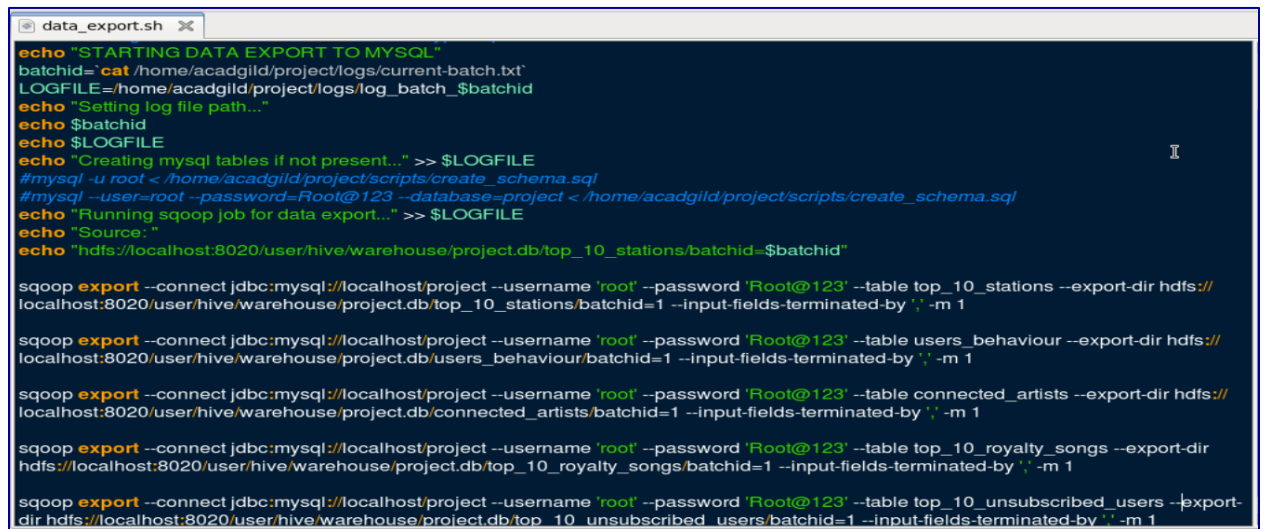
sh /home/acadgild/project/scripts/data_export.sh

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

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

xi. Data Export:

Script: *data_export.sh*



```
echo "STARTING DATA EXPORT TO MYSQL"
batchid=`cat /home/acadgild/project/logs/current-batch.txt`
LOGFILE=/home/acadgild/project/logs/log_batch_${batchid}
echo "Setting log file path..."
echo $batchid
echo $LOGFILE
echo "Creating mysql tables if not present..." >> $LOGFILE
#mysql -u root < /home/acadgild/project/scripts/create_schema.sql
#mysql --user=root --password=Root@123 --database=project < /home/acadgild/project/scripts/create_schema.sql
echo "Running sqoop job for data export..." >> $LOGFILE
echo "Source: "
echo "hdfs://localhost:8020/user/hive/warehouse/project.db/top_10_stations/batchid=${batchid}"

sqoop export --connect jdbc:mysql://localhost/project --username 'root' --password 'Root@123' --table top_10_stations --export-dir hdfs://localhost:8020/user/hive/warehouse/project.db/top_10_stations/batchid=1 --input-fields-terminated-by ';' -m 1

sqoop export --connect jdbc:mysql://localhost/project --username 'root' --password 'Root@123' --table users_behaviour --export-dir hdfs://localhost:8020/user/hive/warehouse/project.db/users_behaviour/batchid=1 --input-fields-terminated-by ';' -m 1

sqoop export --connect jdbc:mysql://localhost/project --username 'root' --password 'Root@123' --table connected_artists --export-dir hdfs://localhost:8020/user/hive/warehouse/project.db/connected_artists/batchid=1 --input-fields-terminated-by ';' -m 1

sqoop export --connect jdbc:mysql://localhost/project --username 'root' --password 'Root@123' --table top_10_royalty_songs --export-dir hdfs://localhost:8020/user/hive/warehouse/project.db/top_10_royalty_songs/batchid=1 --input-fields-terminated-by ';' -m 1

sqoop export --connect jdbc:mysql://localhost/project --username 'root' --password 'Root@123' --table top_10_unsubscribed_users --export-dir hdfs://localhost:8020/user/hive/warehouse/project.db/top_10_unsubscribed_users/batchid=1 --input-fields-terminated-by ';' -m 1
```

xii. Final Result:

After executing the data export script, final set of tables have been created successfully in target MySQL. Below is the screenshot of target tables created in MySQL.

```
mysql> use project;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_project |
+-----+
| connected_artists |
| top_10_royalty_songs |
| top_10_stations |
| top_10_unsubscribed_users |
| users_behaviour |
+-----+
5 rows in set (0.00 sec)
```

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

```
mysql> select * from top_10_stations;
+-----+-----+-----+
| station_id | total_distinct_songs_played | distinct_user_count |
+-----+-----+-----+
| ST413 | 2 | 2 |
| ST408 | 2 | 2 |
| ST404 | 2 | 2 |
| ST401 | 2 | 2 |
| ST409 | 1 | 1 |
| ST402 | 1 | 1 |
| ST400 | 1 | 2 |
+-----+-----+-----+
7 rows in set (0.01 sec)
```

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.

```
mysql> select * from users_behaviour;
+-----+-----+
| user_type | duration |
+-----+-----+
| SUBSCRIBED | 56266233 |
| UNSUBSCRIBED | 39462561 |
+-----+-----+
2 rows in set (0.02 sec)
```

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

```
mysql> select * from connected_artists;
+-----+-----+
| artist_id | user_count |
+-----+-----+
| ART302 | 2 |
| ART304 | 2 |
| ART301 | 2 |
| ART305 | 1 |
| ART300 | 1 |
+-----+-----+
5 rows in set (0.03 sec)
```

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.

```
mysql> select * from top_10_royalty_songs;
```

song_id	duration
SNG207	19489612
SNG208	14680587
SNG202	9987374
SNG204	5481793
SNG206	3471600
SNG201	2887800
SNG200	480
SNG209	180

8 rows in set (0.01 sec)

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

```
mysql> select * from top_10_unsubscribed_users;
```

user_id	duration
USR116	23993874
USR119	9986894
USR120	5481793

3 rows in set (0.00 sec)

xiii. Scheduling:

The music data analysis application has been scheduled to execute for every 3 hours by scheduling the program in crontab.

```
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ crontab -l
* * * * * /home/acadgild/install/data/dfs/simple/update-acadgildvm.sh
[acadgild@localhost ~]$ crontab -e
crontab: installing new crontab
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ crontab -l
* * * * * /home/acadgild/install/data/dfs/simple/update-acadgildvm.sh
0 0,3,6,9,12,15,18,21 * * * /home/acadgild/project/scripts/wrapper.sh
[acadgild@localhost ~]$
```