**Music Data Analysis:**

**Introduction:**

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 behaviour of users, classify users, calculate royalties associated with the song and make appropriate business strategies. The file server receives data files periodically after every 3 hours.

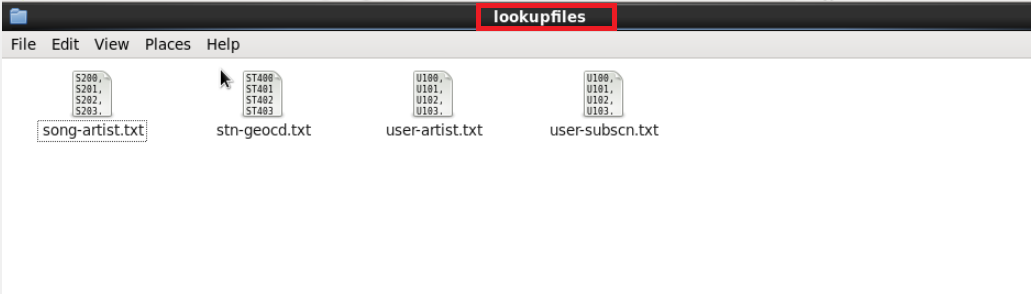
**DATASET:**

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

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

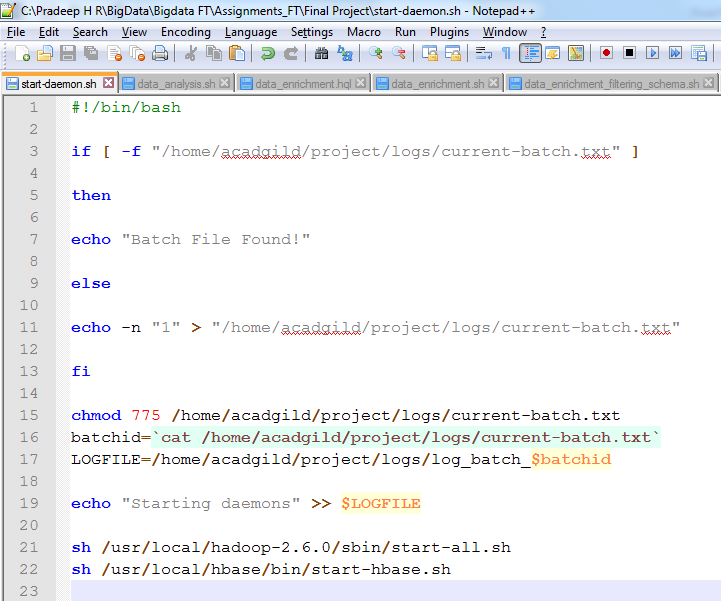
3. Data present in lookup directory should be used in HBase.



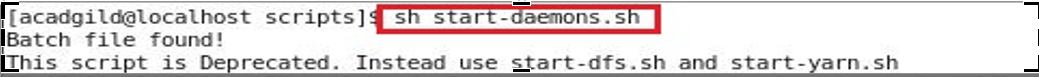


1. **Implementation:** 
   1. **Starting all daemons in Hadoop:**

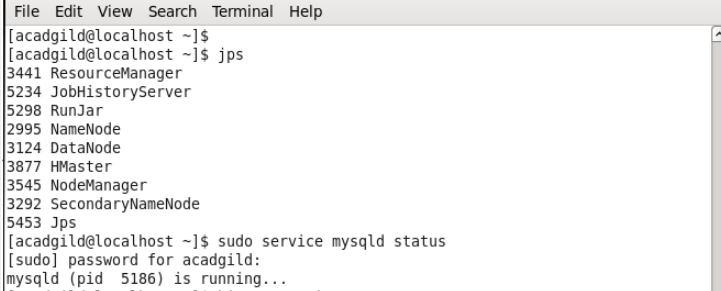
In this script we will start all the daemons that are required to start the services like hive, hbase, MySQL etc.



Terminal with all the daemons started screenshot.



With the help of **JPS** command we can list out the services that are currently in active state.



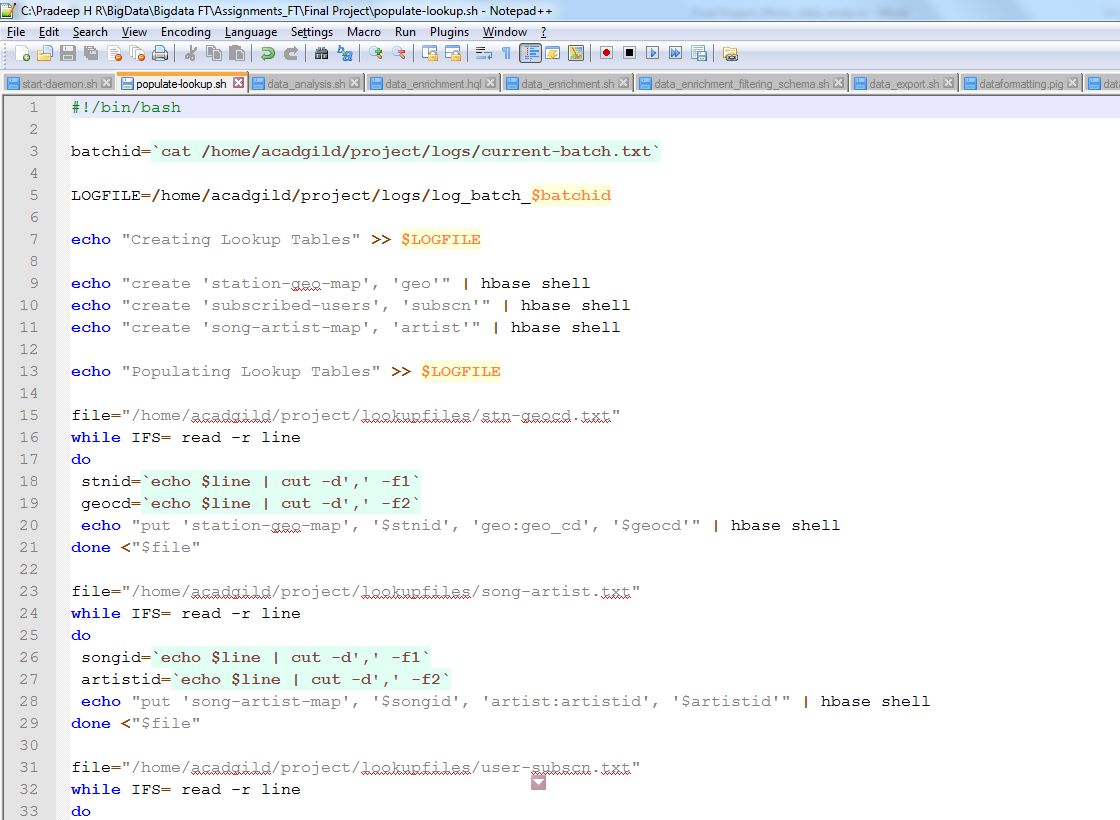
**Lookup Tables creation in HBASE:**

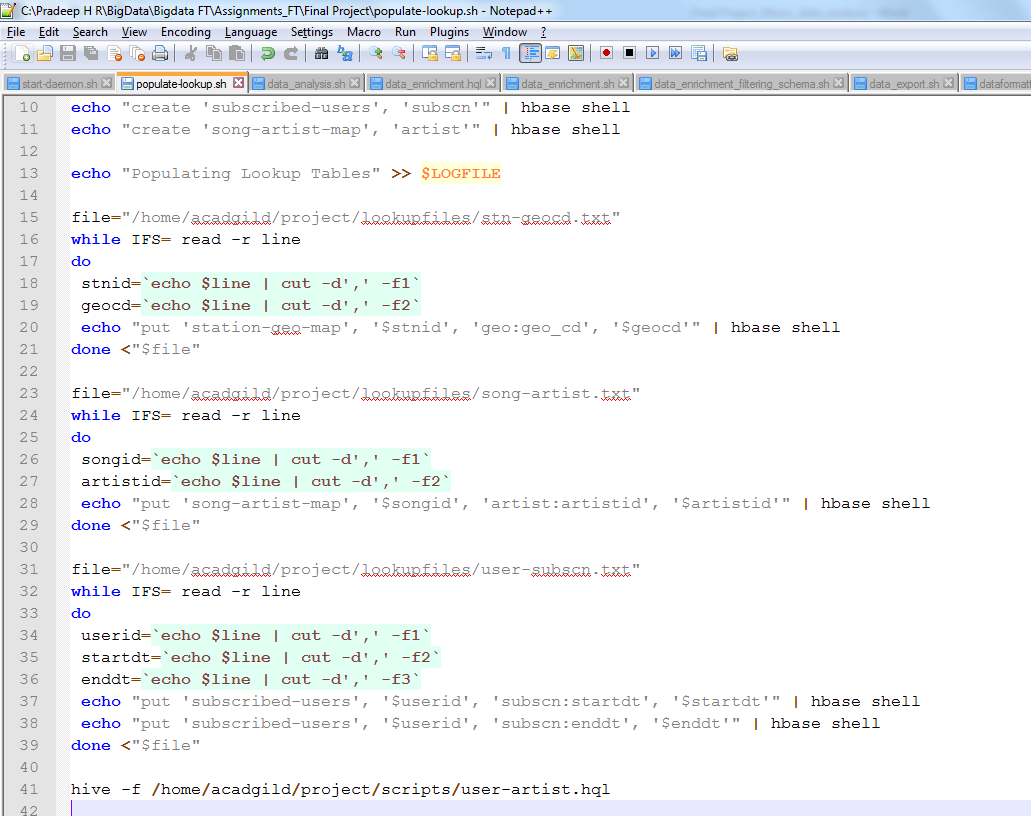
By using the populate-lookup.sh script we will create lookup tables in Hbase. These tables have to be used in data formatting, data enrichment and analysis stage.

|  |  |
| --- | --- |
| **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 |

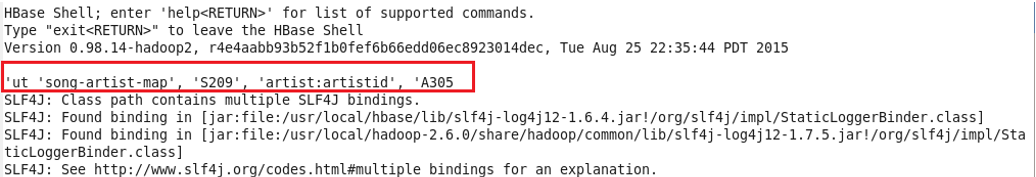
These 4 lookup files are used to create Hbase tables using populate-lookup.sh script and then with the help of data\_enrichment\_filtering\_schema.sh file we will create hive tables on the top of Hbase tables using create\_hive\_hbase\_lookup.hql

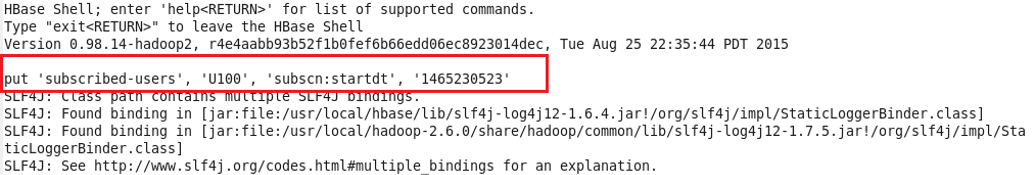
* 1. **Creating Lookup Tables :**





By using the shell scripting 4 lookup tables are created in Hbase NoSQL Database.

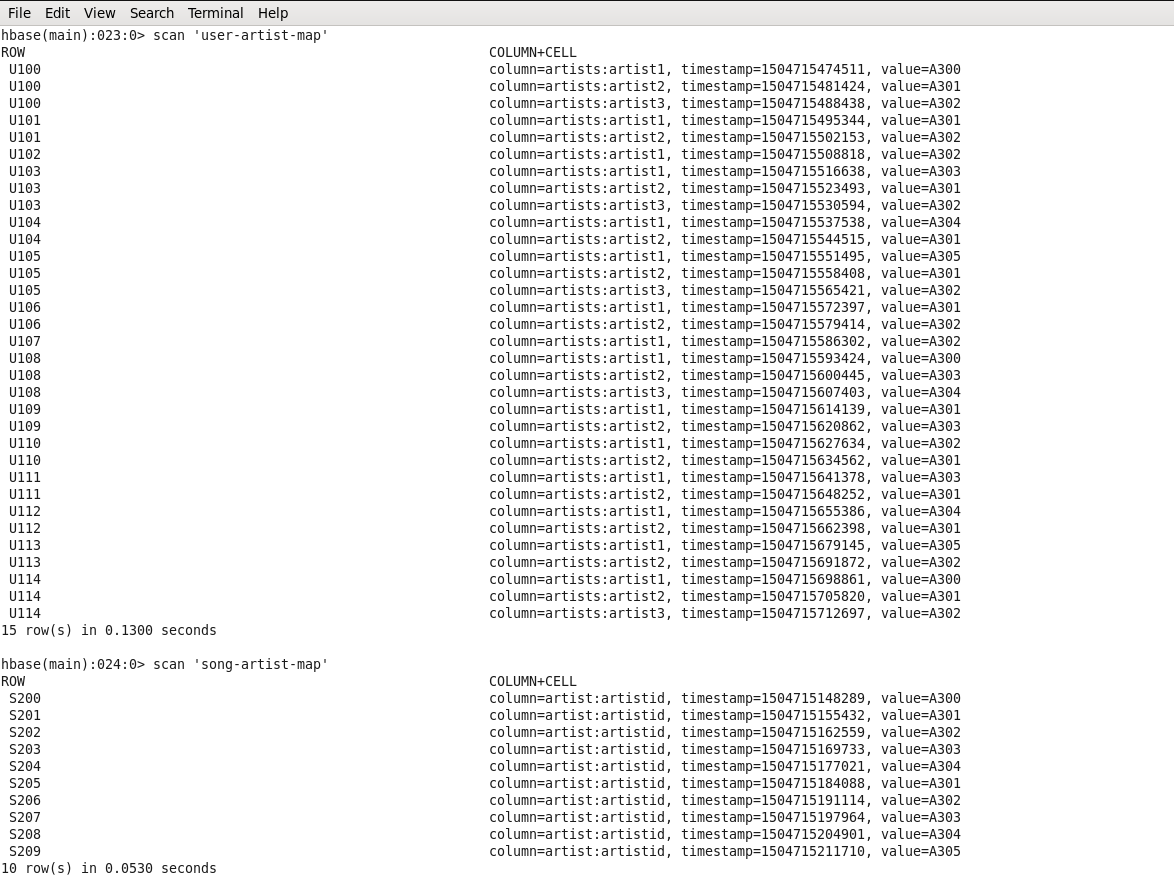


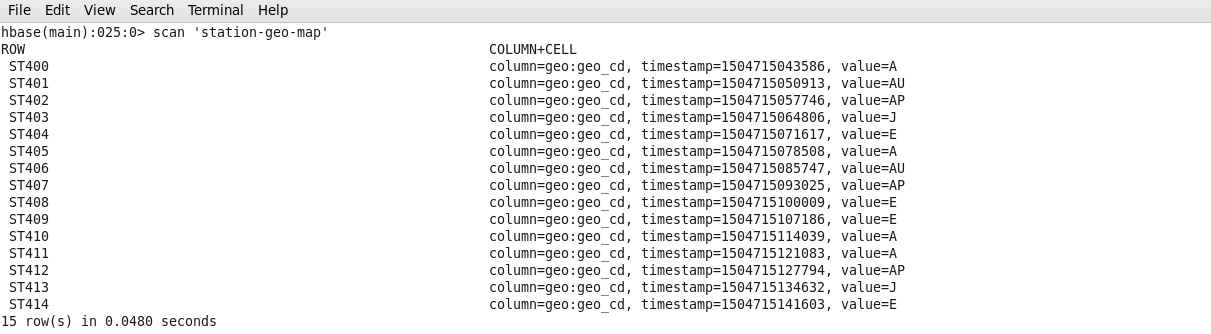


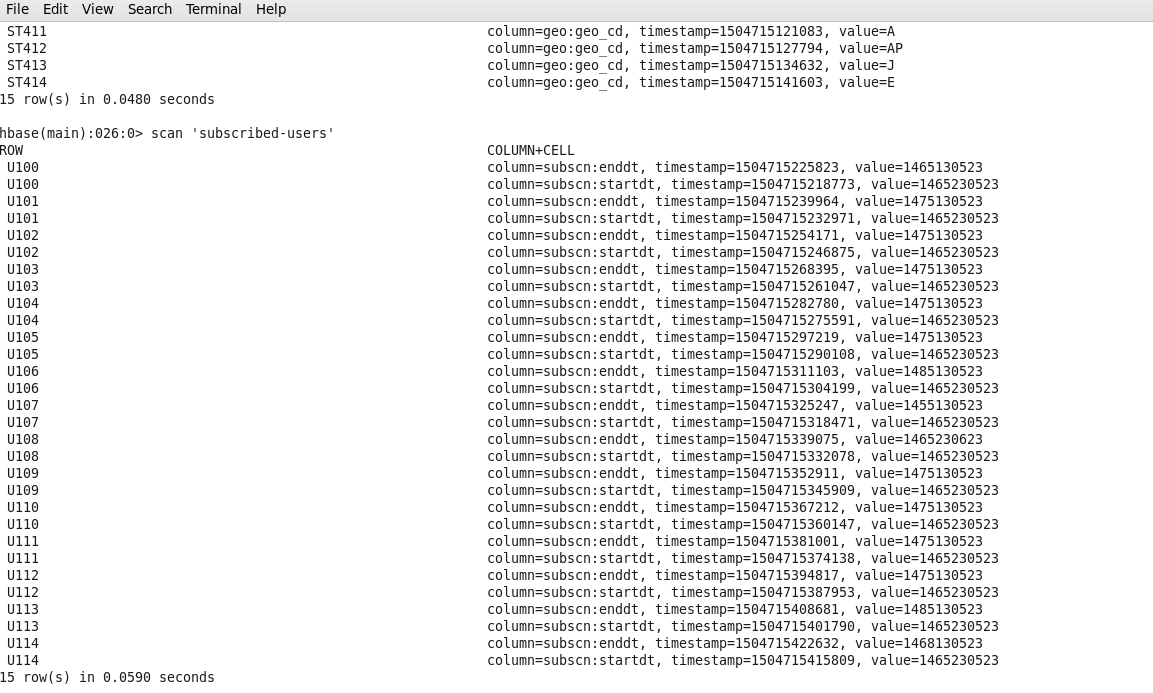


Now we can see the lookup tables in Hbase shell terminal as shown below:





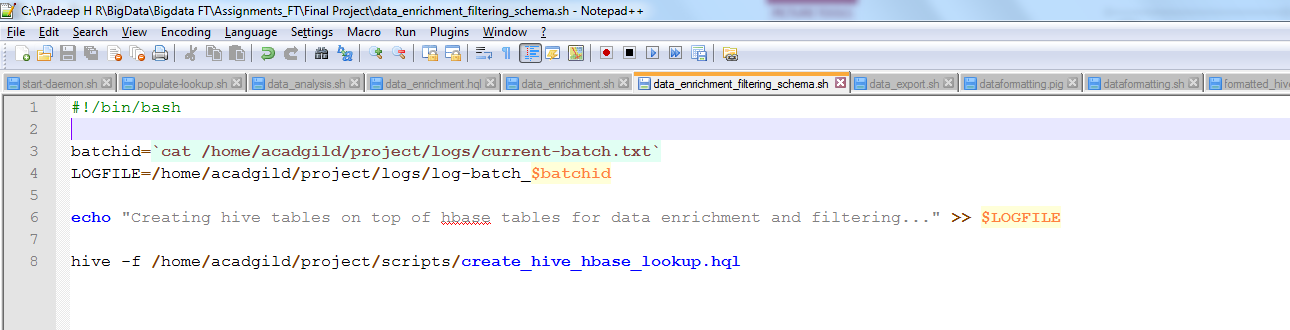


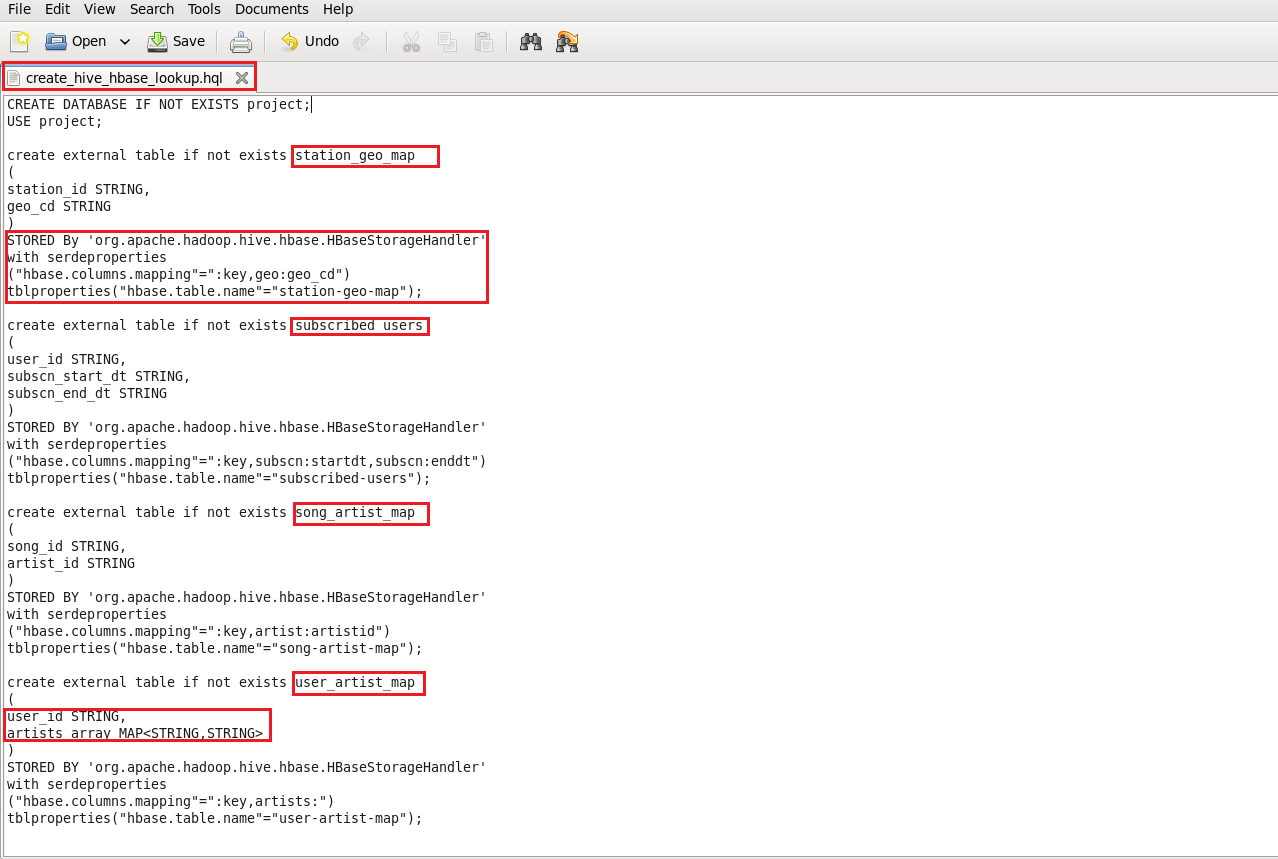


Now Lookup table creation is completed. So now we need to link theses lookup tables in hive using the Hbase Storage Handler.

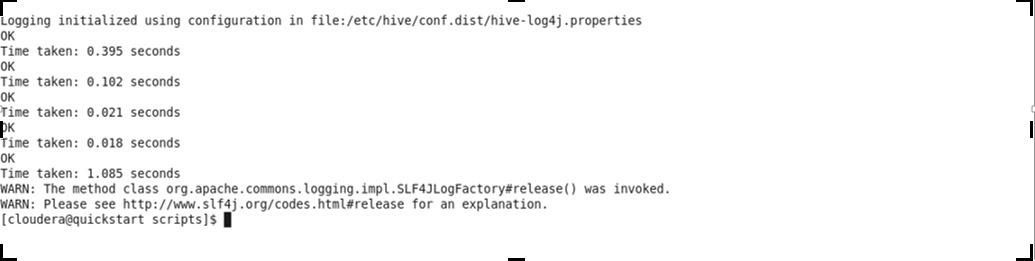
* 1. **Creating Hive Tables on the top of Hbase:**

Hive external tables are created by matching the columns of Hbase tables to hive tables.





In the below screenshot, it can been seen tables getting created in hive by running the data\_enrichement\_filtering\_schema.sh file

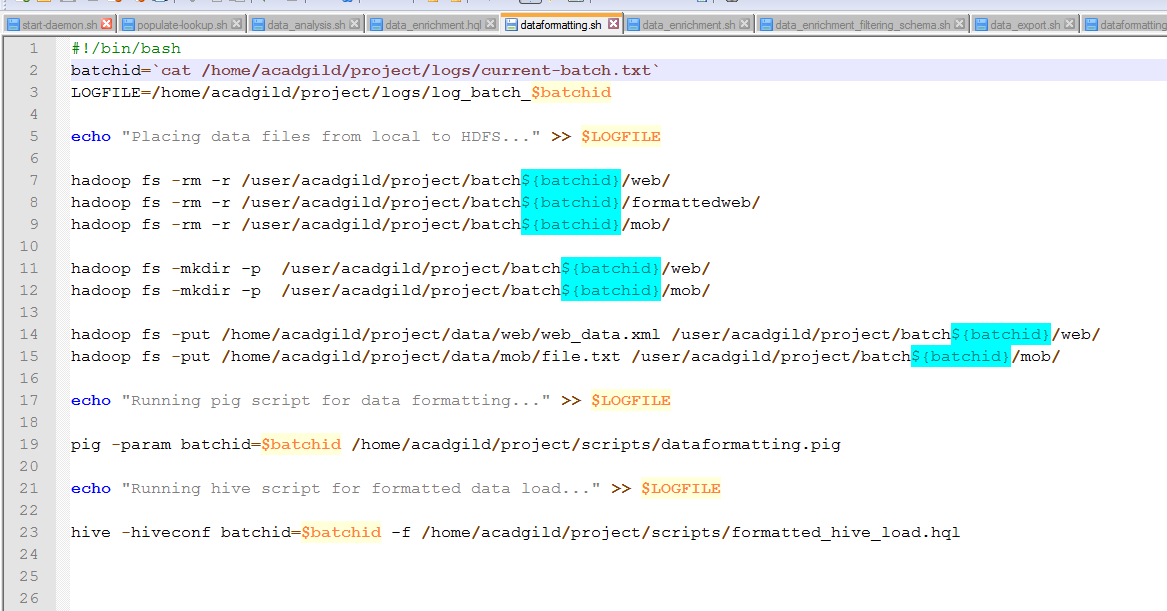




* 1. **Data Formatting:**

Data coming from both web applications and mobile applications are merged to create a common table for analyzing purpose

Partitioned data are created based on batchid, since scripts should run for every 3 hours.

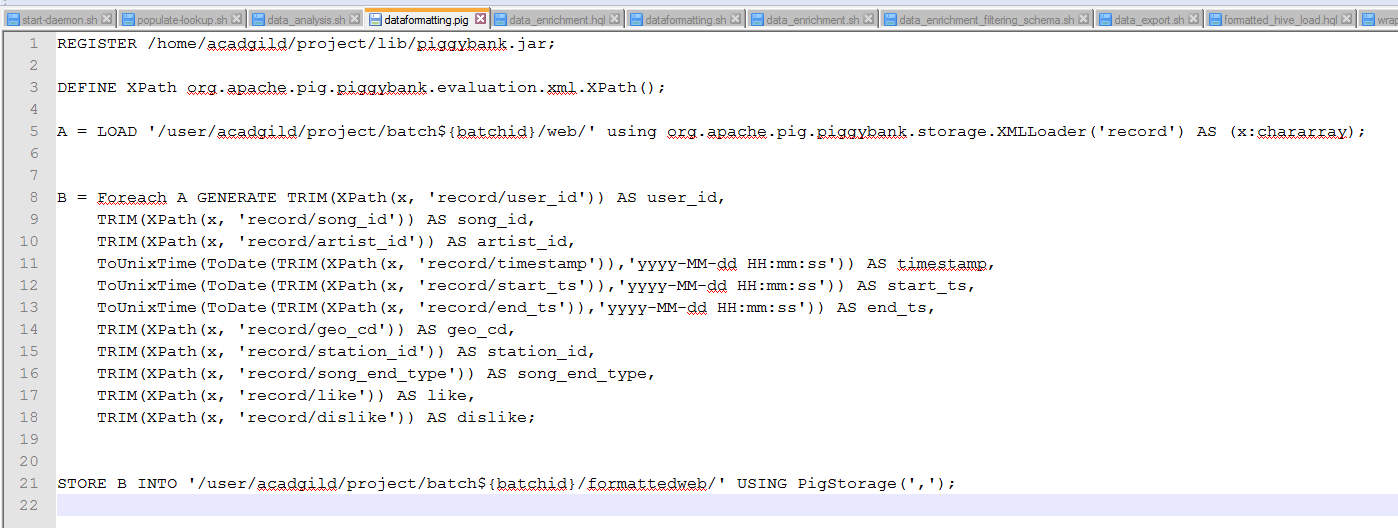


Below two scripts are run to format the data.

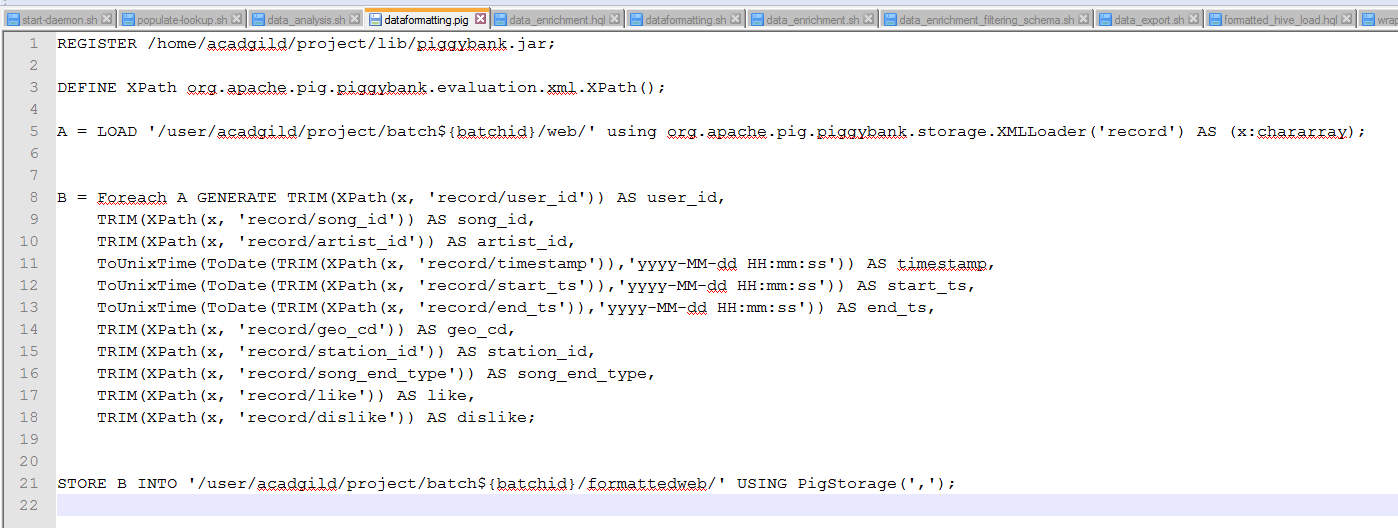
1) Dataformatting.pig

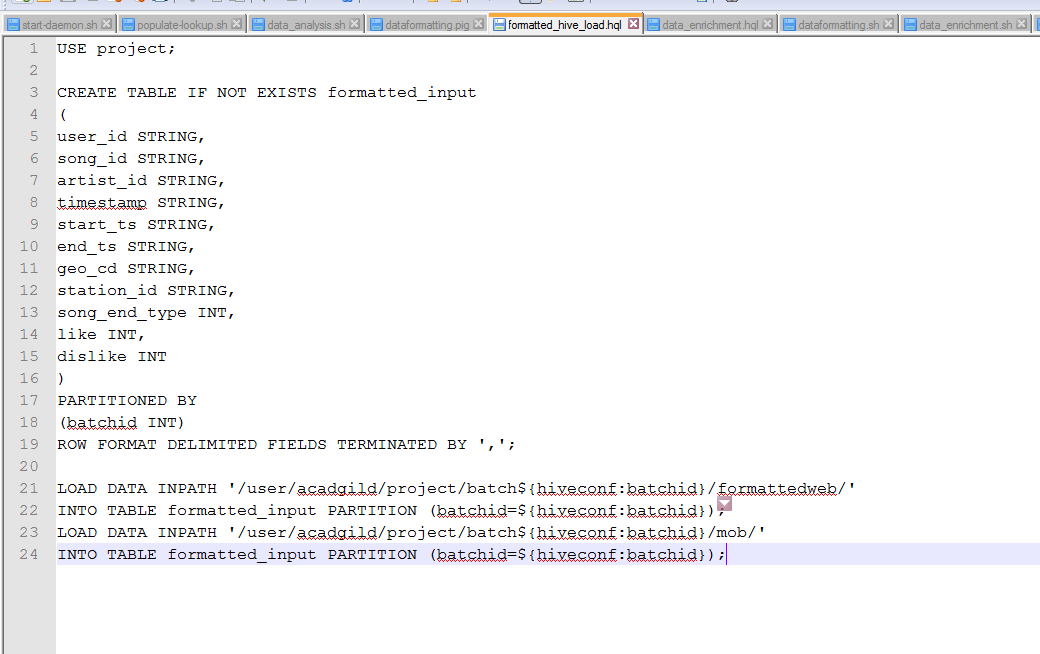
2) Formatted\_hive\_load.hql

Pig script to parse the data coming from web\_data.xml to csv format and partition both web and mob data based on based on batch ID’s



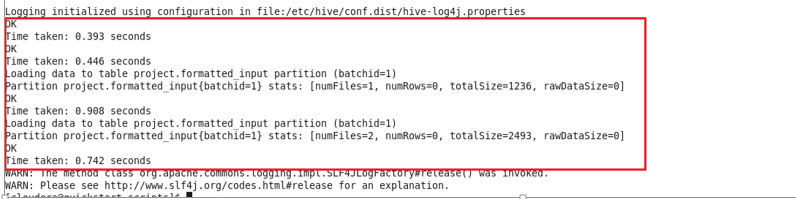
Hive script to load the parsed data from pig to hive and do analysis based on this data.





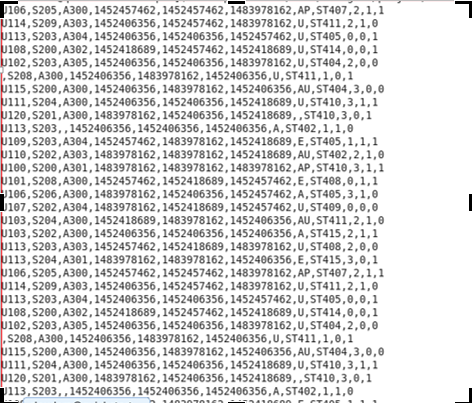
Executing **./dataformatting.sh**



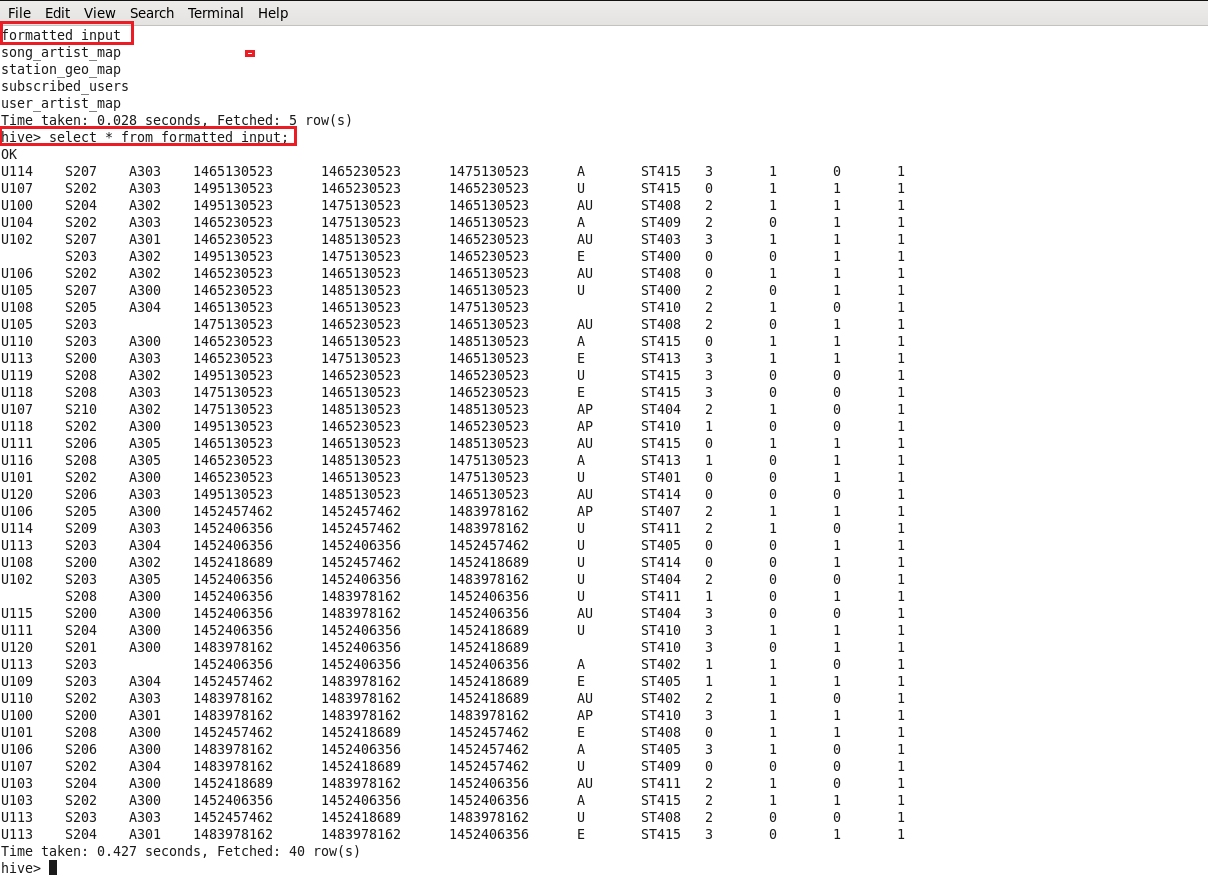


In the above screenshots it can be seen that dataformatting.pig along with the formatted\_hive\_load.hql executed successfully.

The output of dataformatting.sh script in HDFS folders:



DataFormatting.sh output in hive terminal:



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.

* 1. **Data Enrichment:**

In this phase we will enrich the data coming from web and mobile applications using the lookup table stored in Hbase and divide the records based on the enrichment rules into ‘pass’ and ‘fail’ records.

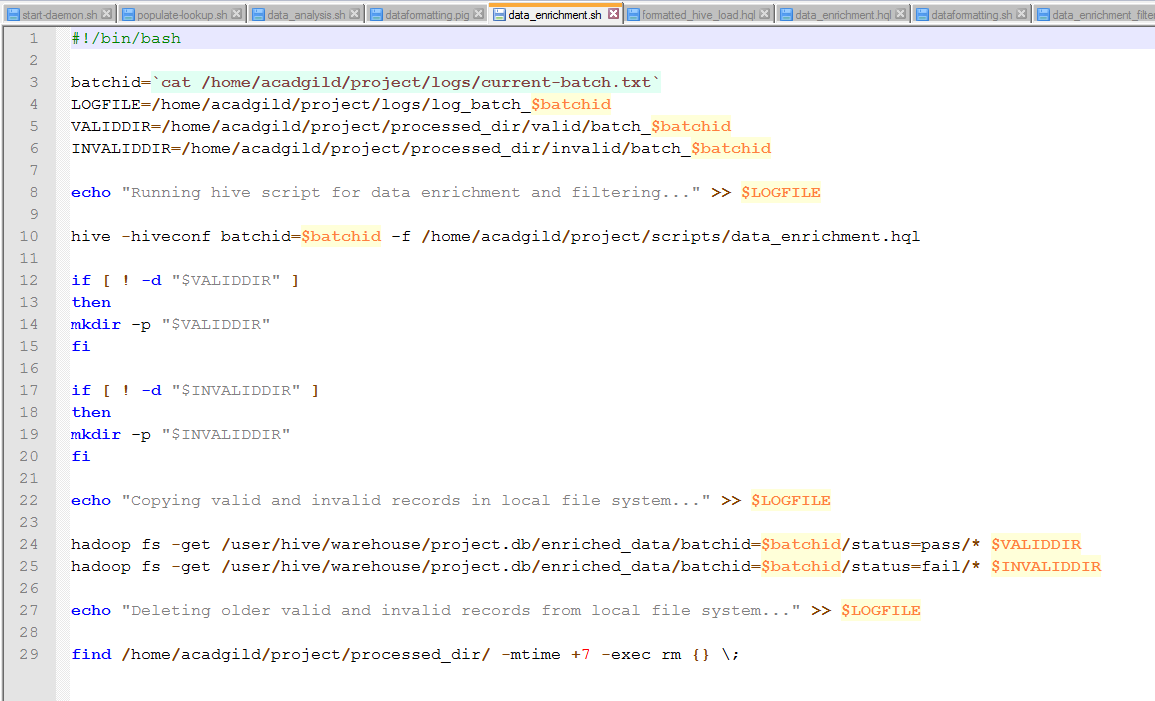
**Rules for data enrichment**

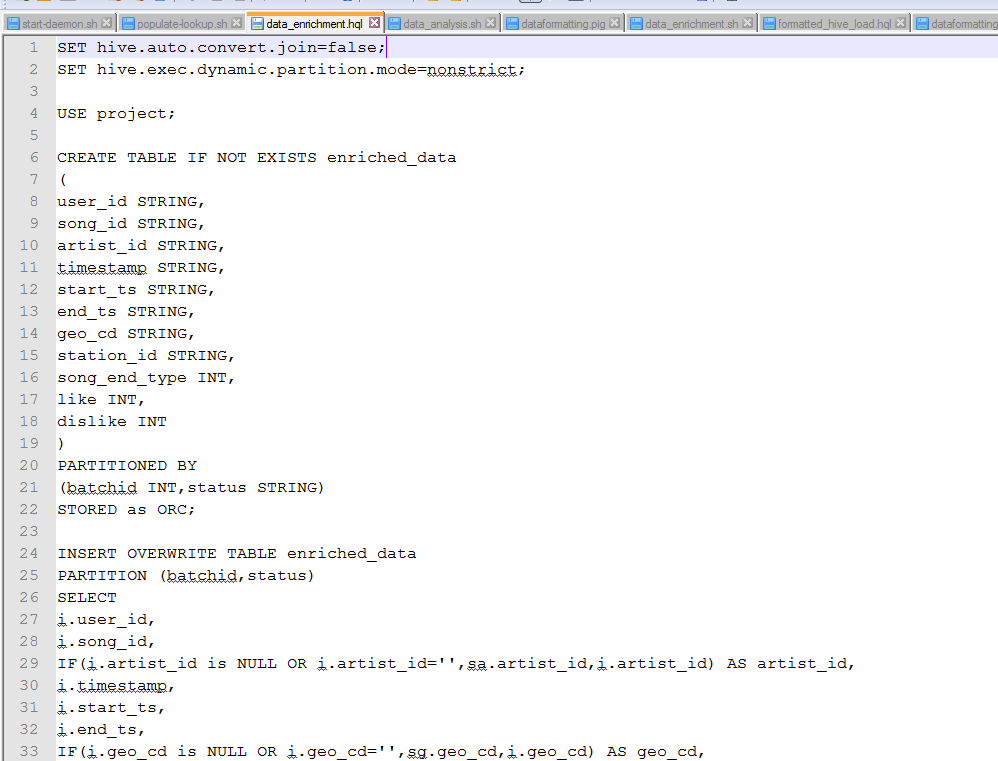
1. If any of like or dislike is NULL or absent, consider it as 0.

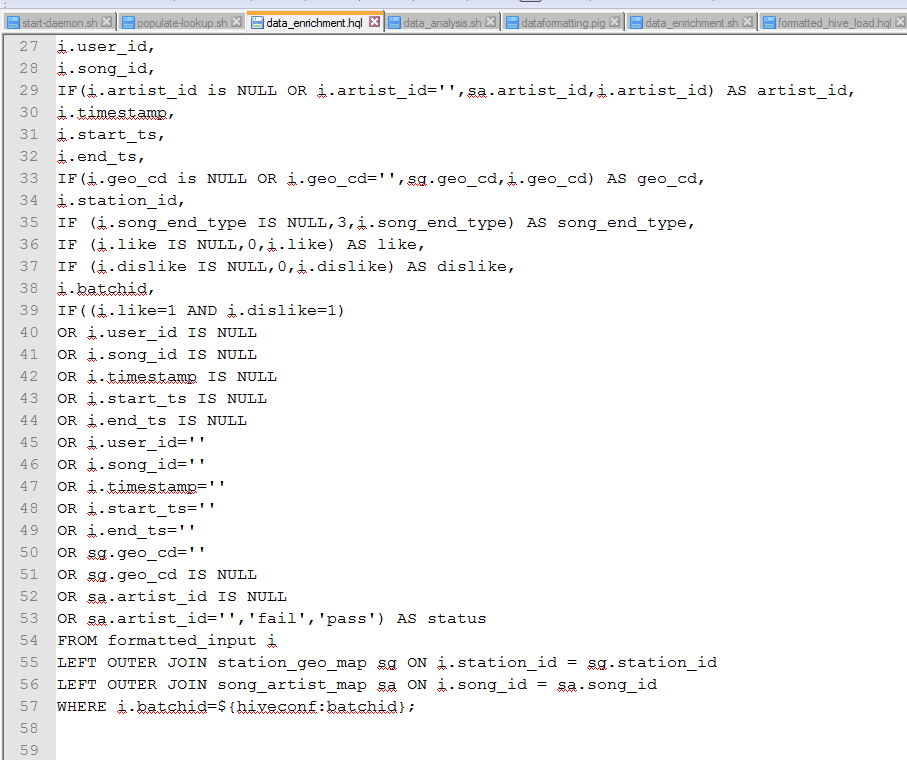
2. If fields like Geo\_cd and Artist\_id are NULL or absent, consult the lookup tables for fields Station\_id and Song\_id respectively to get the values of Geo\_cd and Artist\_id.

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

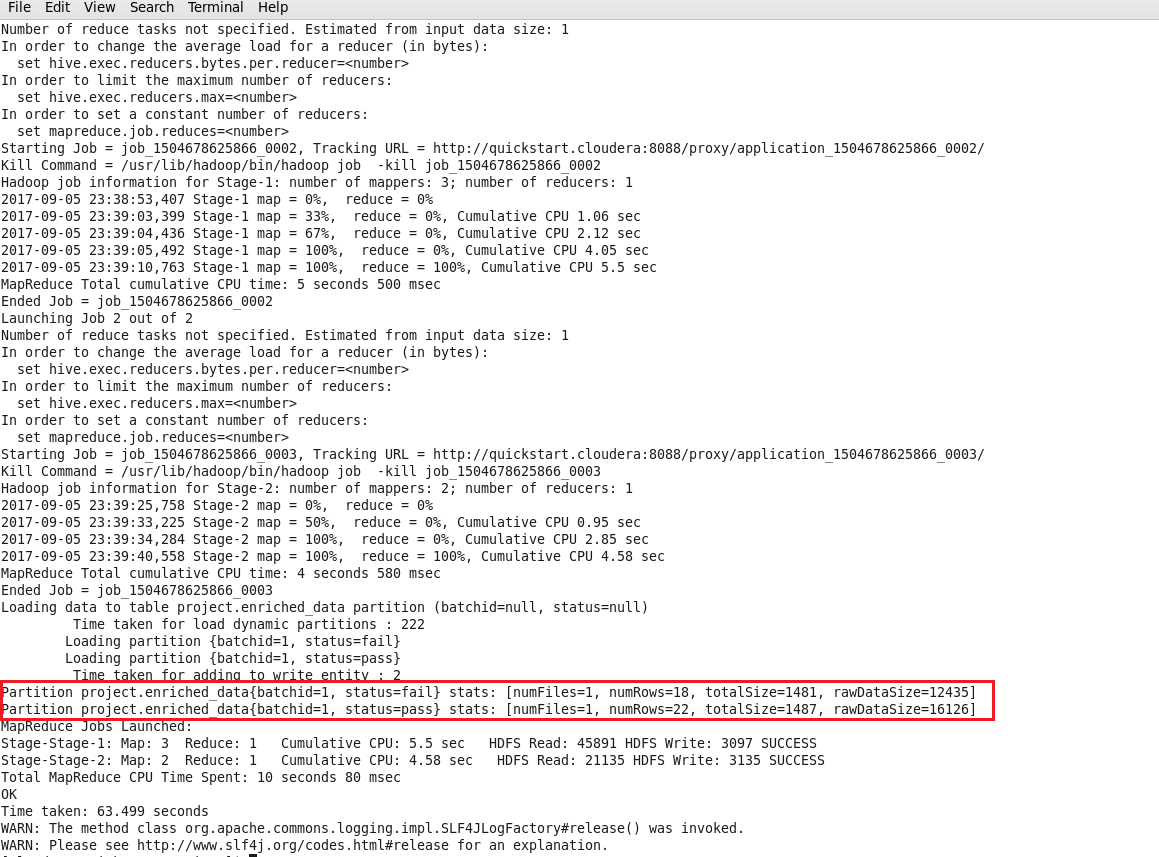
So based on the enrichment rules we will fill the null geo\_cd and artist\_id values with the help of corresponding lookup values in song-artist-map and station-geo-map tables in Hive-Hbase tables.



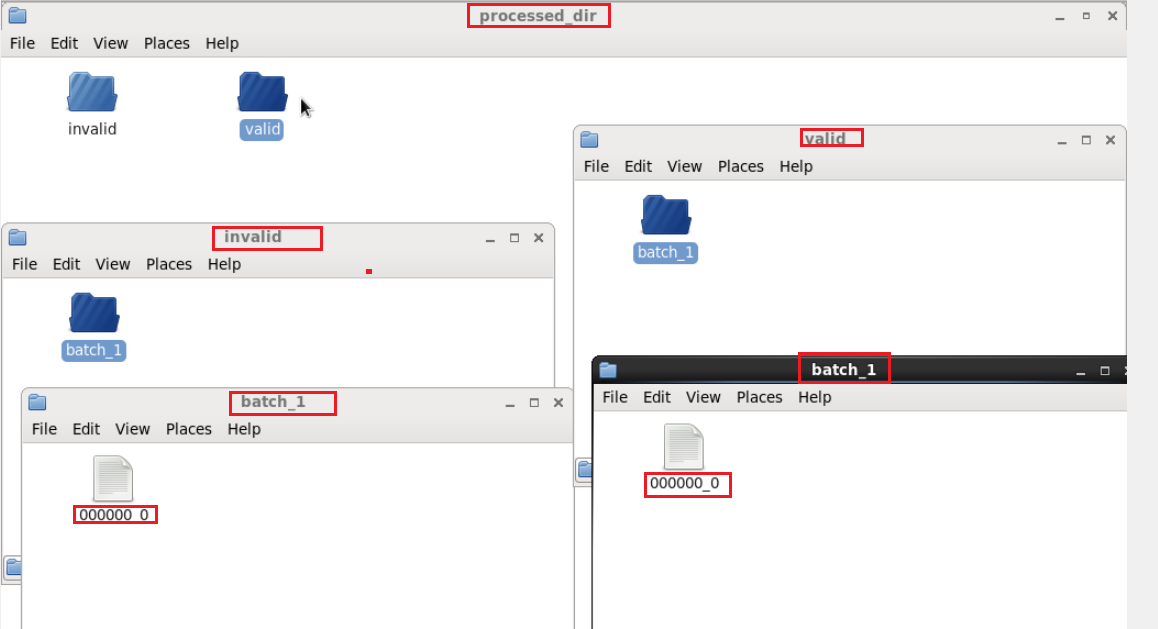




Executing **./data\_enrichment.sh**



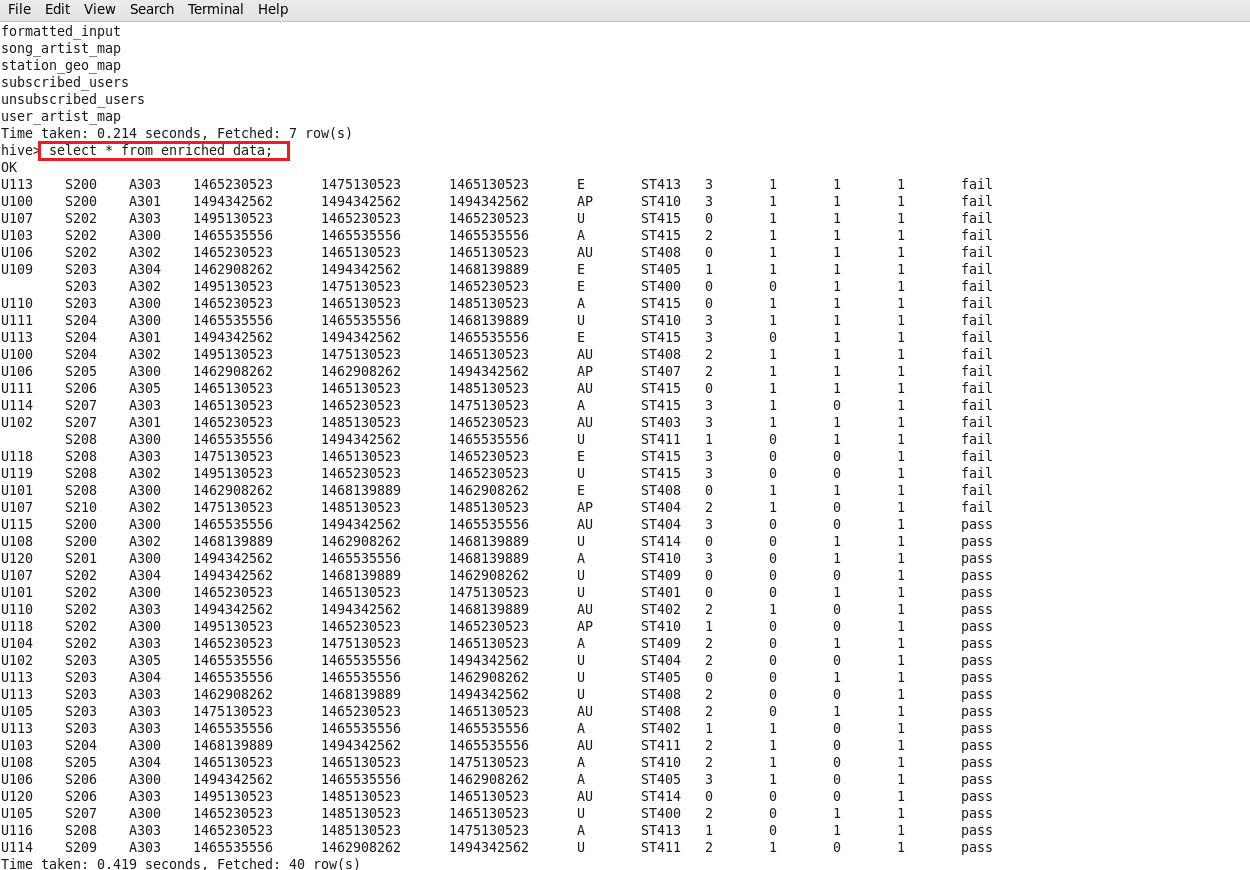
At the end script will automatically divide the records based on status pass & fail and dump the result into processed\_dir folder with valid and invalid folders.



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



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



* 1. **Data Analysis using Spark:**

In this stage we will do analysis on enriched data using Spark SQL and run the program using **Spark-Submit** command.

Here two spark-submits are used for analysis.

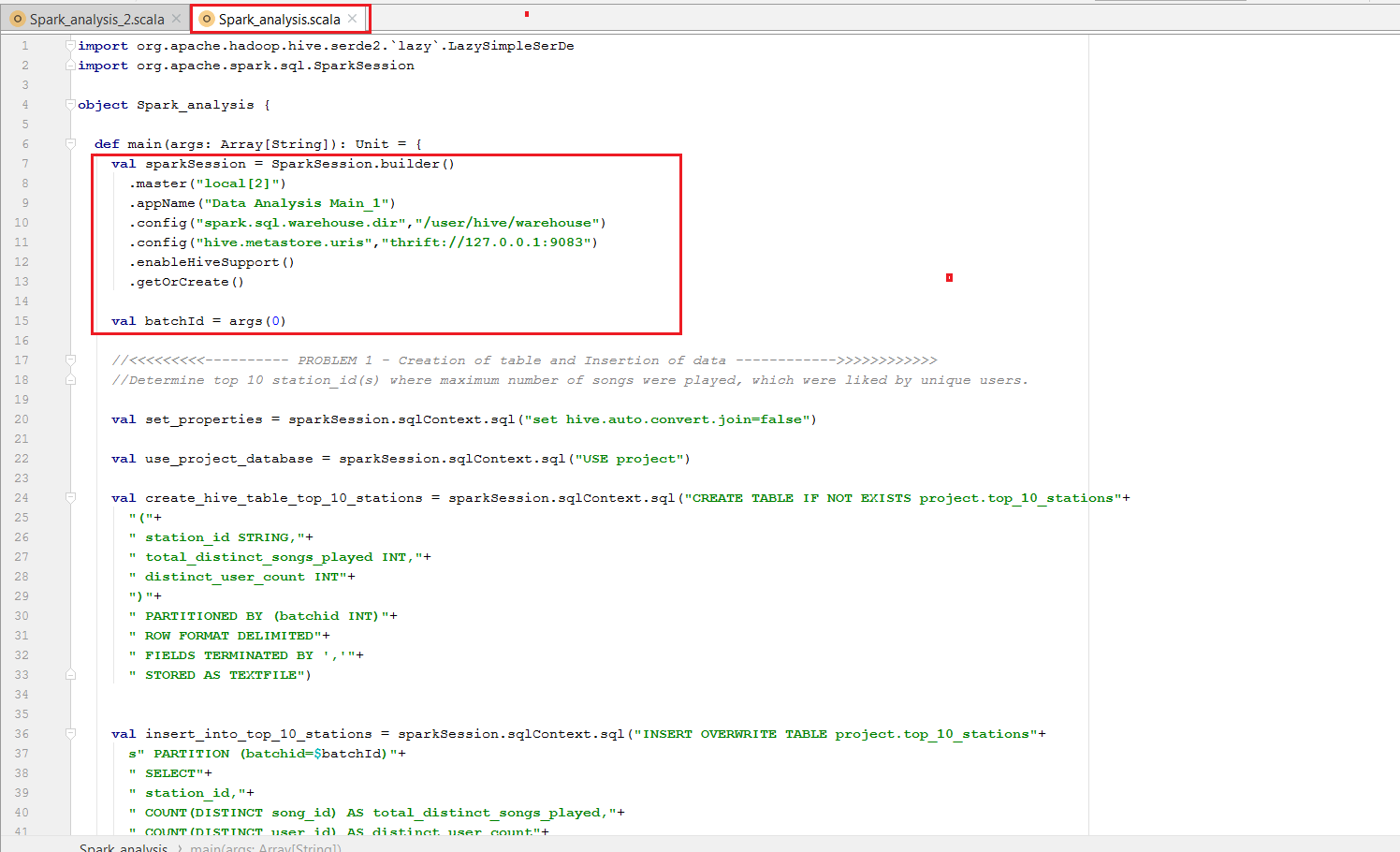
1) Spark\_analysis for creating tables for each query/problem statement.

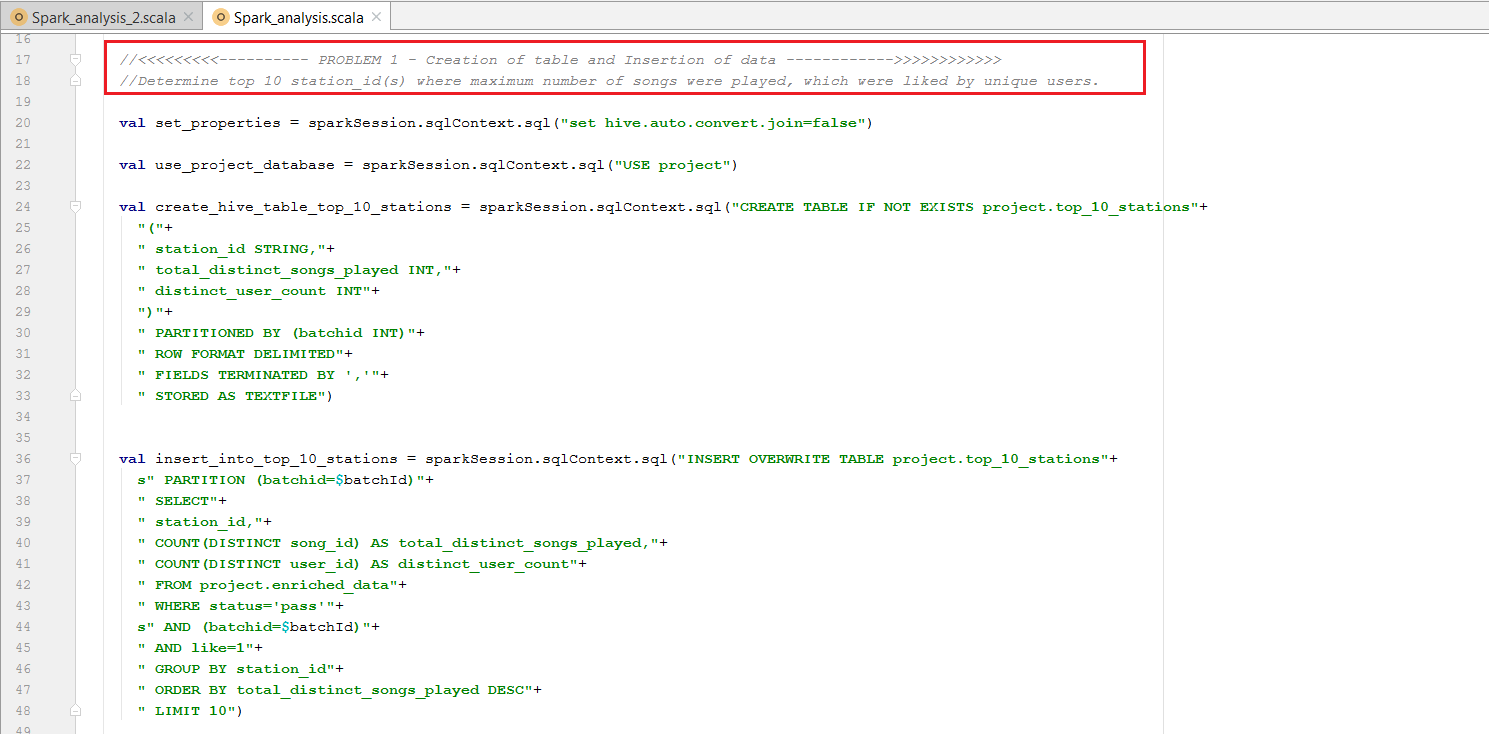
2) Spark\_analysis\_2 for displaying results for each query in terminal.

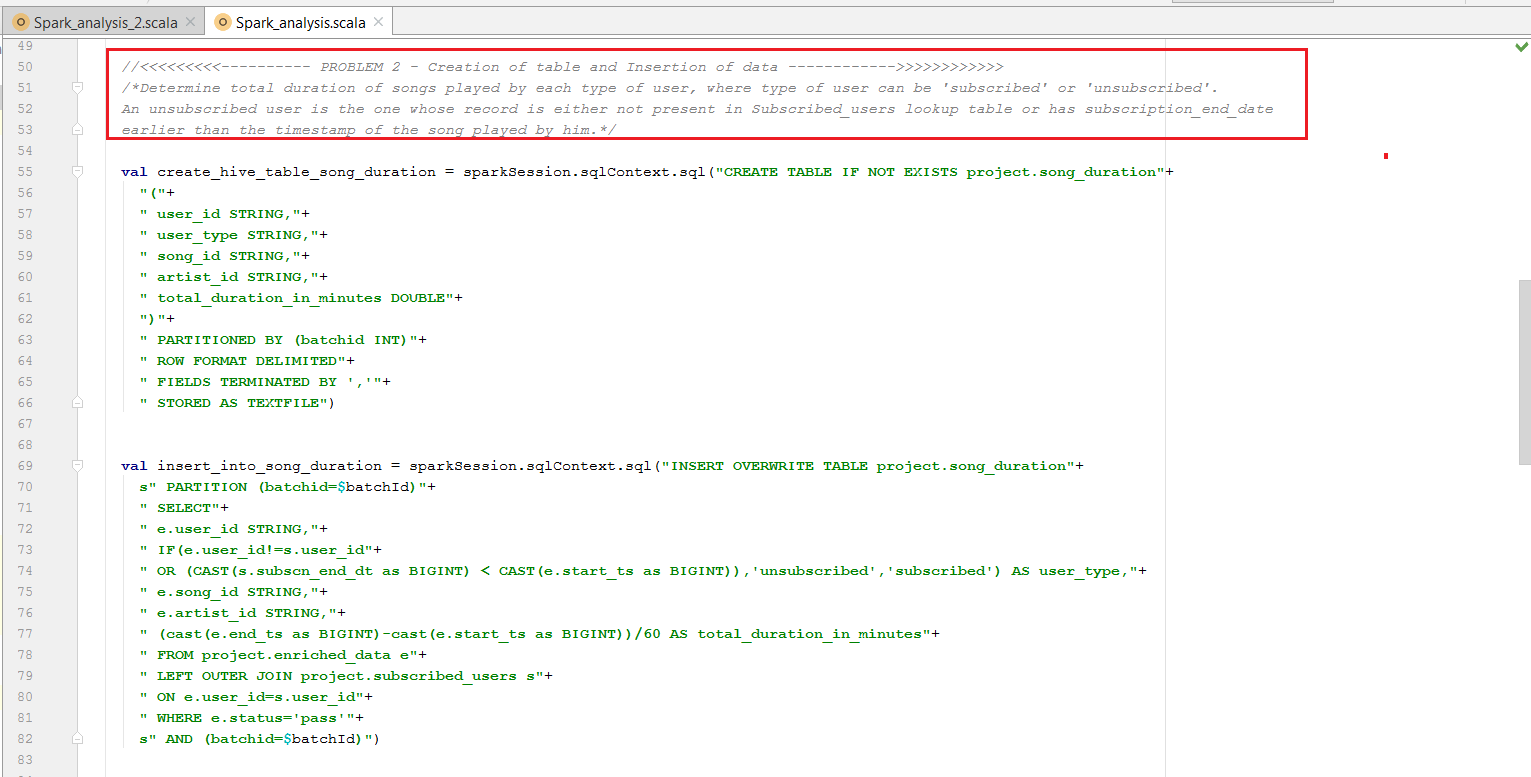


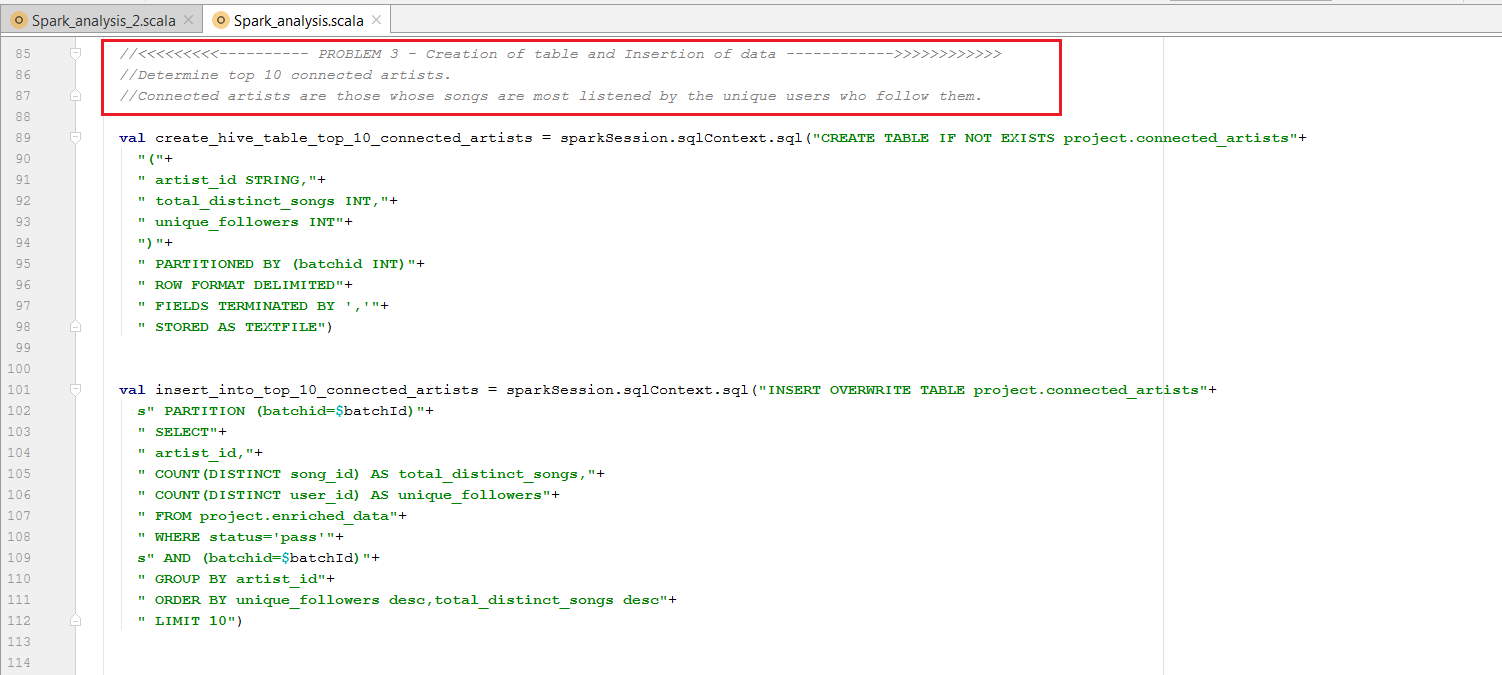
**Spark Source Code:**

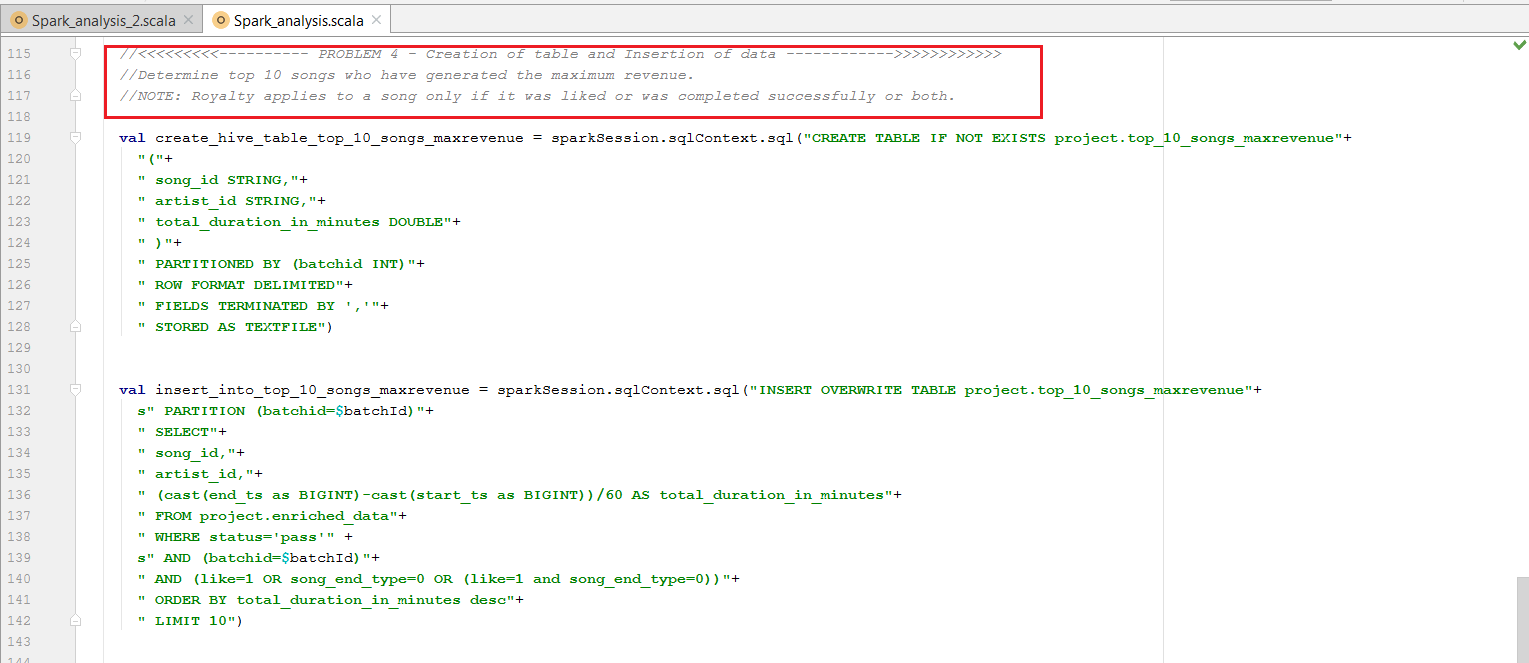
Below is the Scala file for creating tables as per query wise.

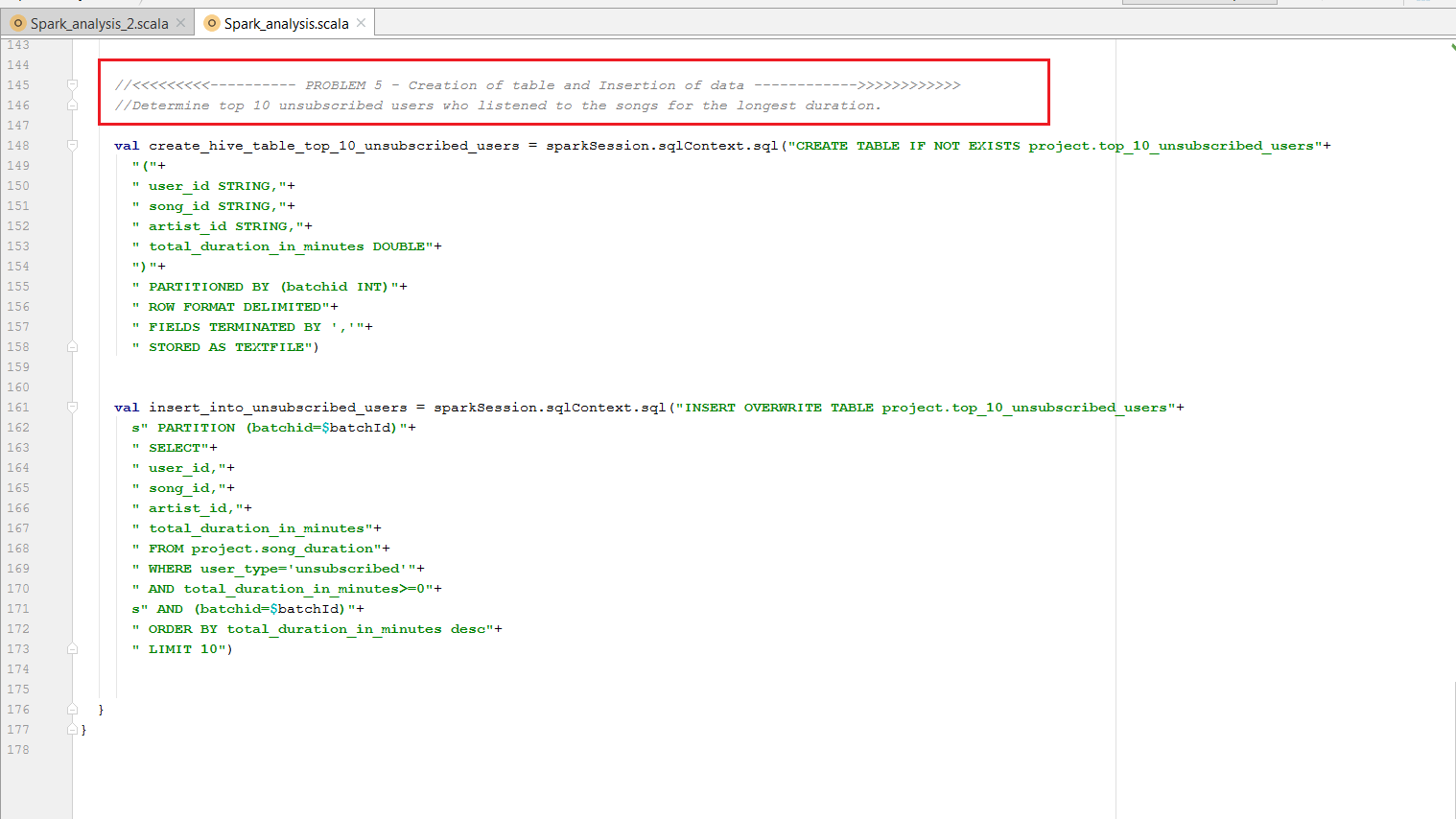




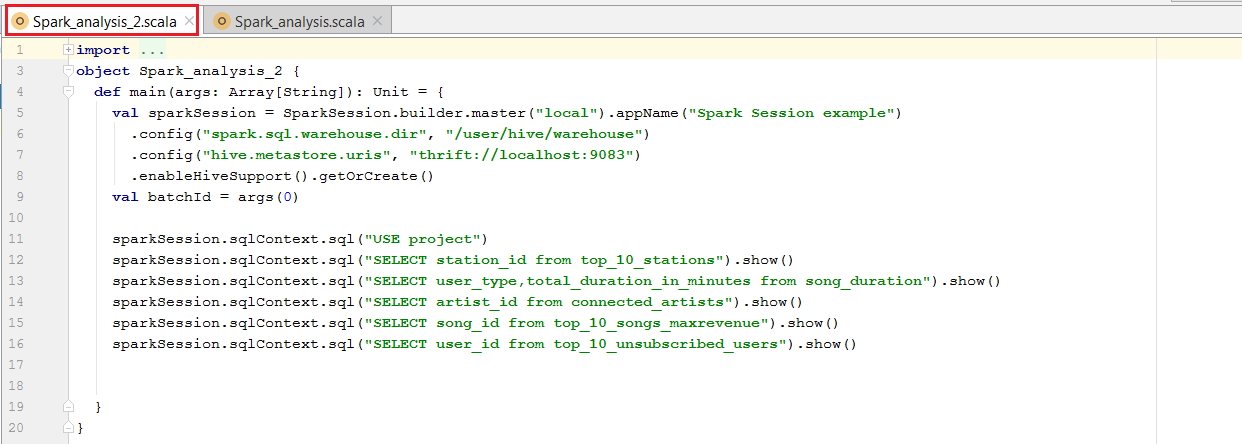








Below is the Scala file is for displaying results of each query in the terminal.

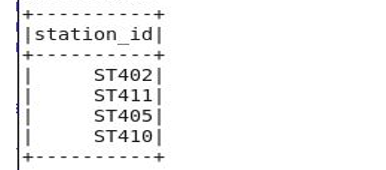


When we run the Data Analysis First it will submit Spark\_analysis file and then it will run the Spark-analysis\_2 file which displays the result for each query.

**Spark analysis\_2 output:**

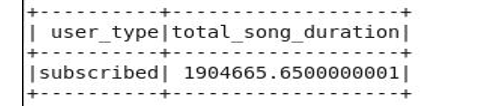
**Problem Statement 1:**

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



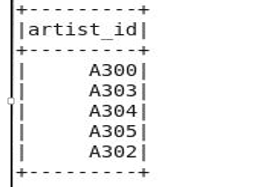
**Problem Statement 2:**

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.



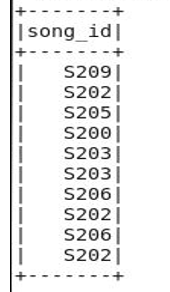
**Problem Statement 3:**

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



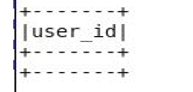
**Problem Statement 4:**

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



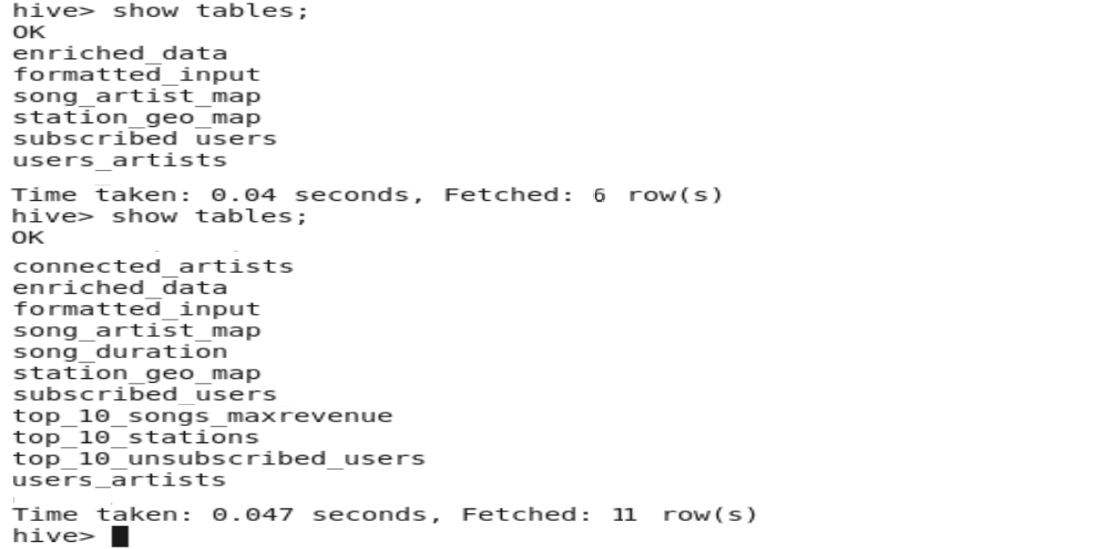
**Problem Statement 5:**

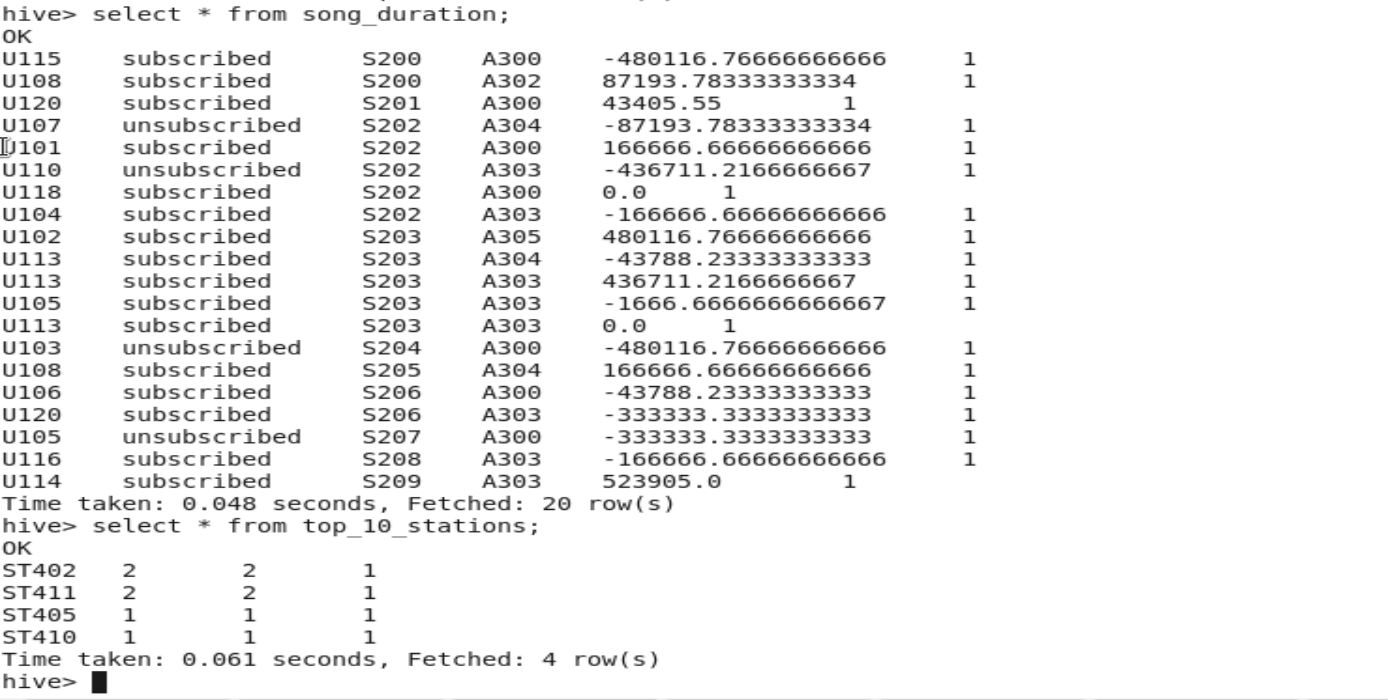
Determine top 10 unsubscribed users who listened to the songs for the longest duration.

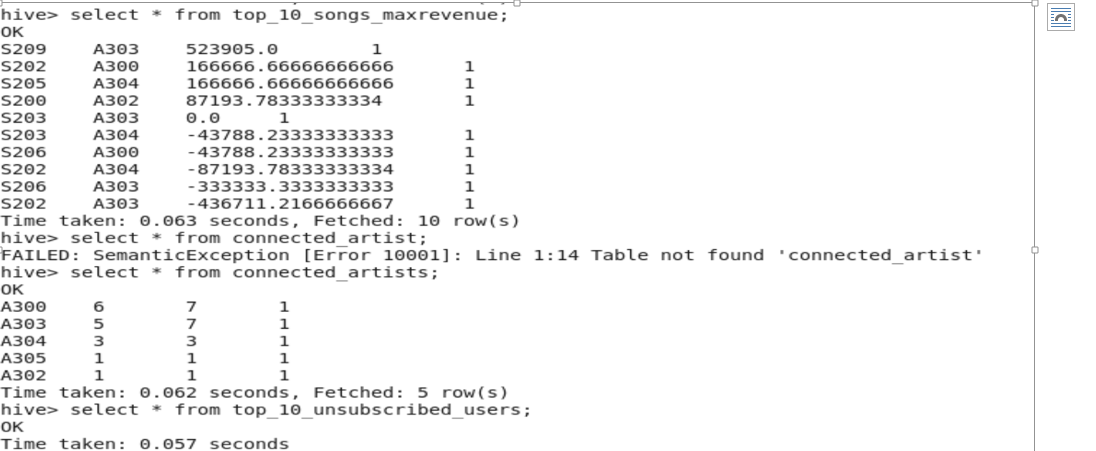


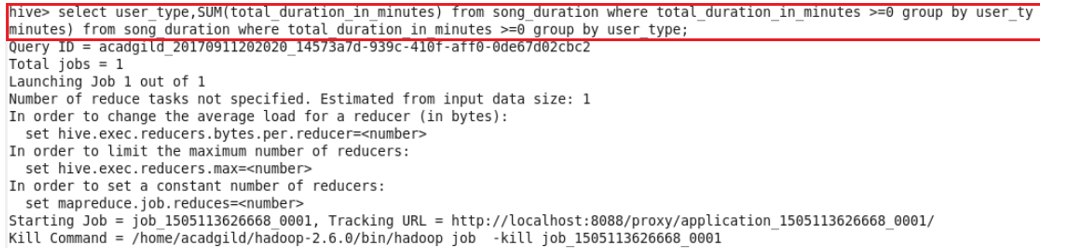
There is no unsubscriber whose has got positive total \_duration\_minutes\_in\_minutes. All the unsubscribed users got negative total\_duration means their end timestamp is less than the start time stamp of the song which is not correct.

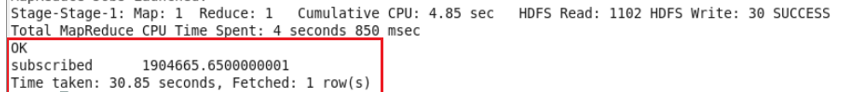
Now we can see the output of Spark analysis in hive:







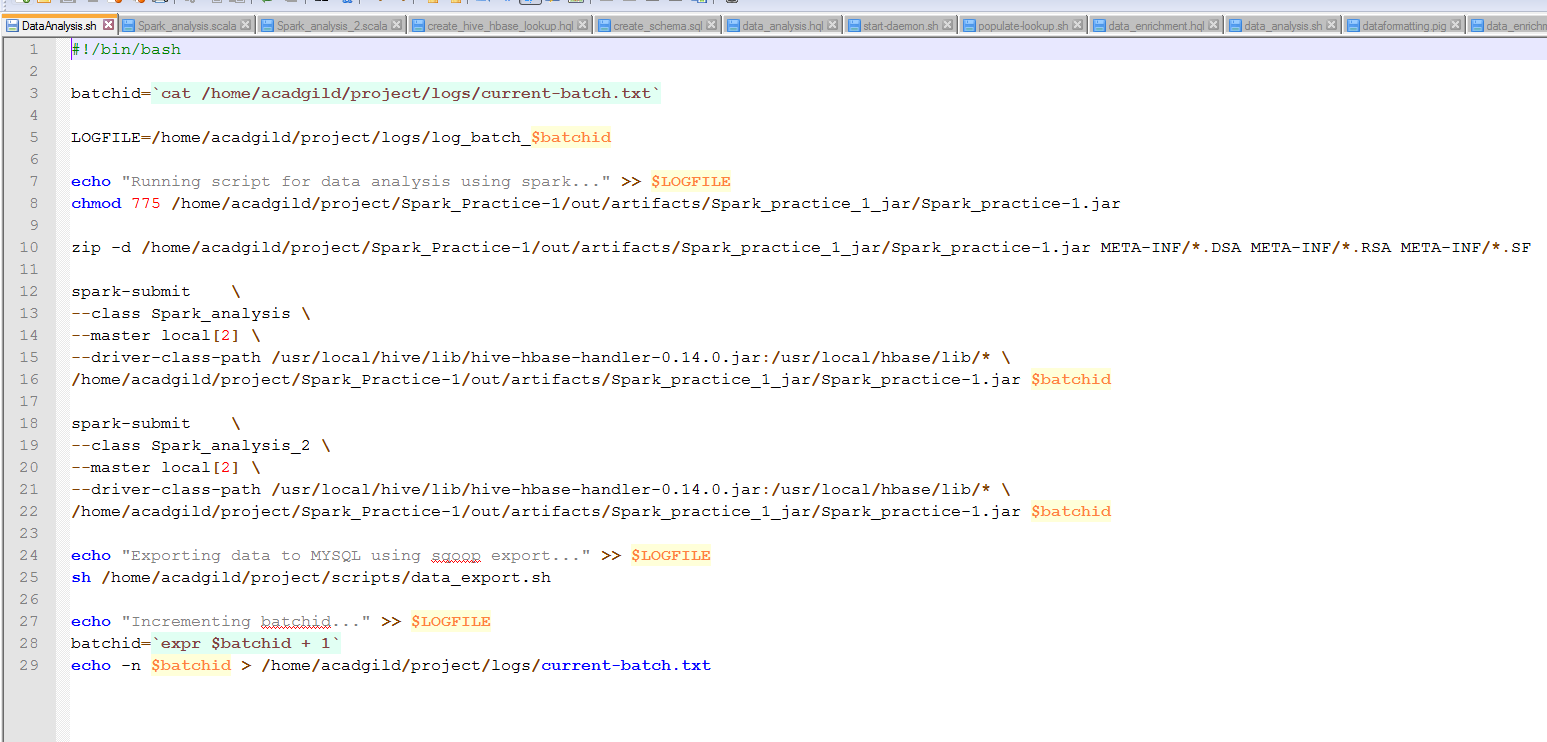


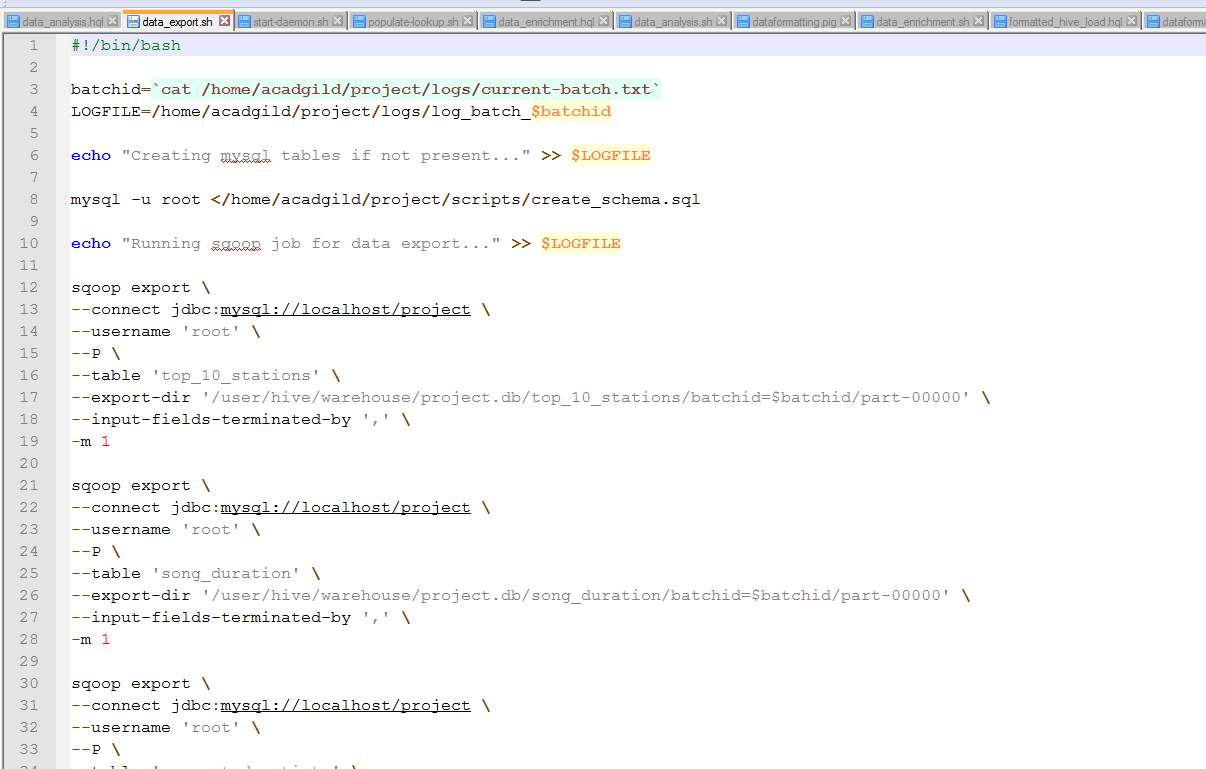


Now we have seen all the spark queries creating the tables for each query. So Data Analysis using Spark is executed successfully.

* 1. Data Export to MYSQL:

In this stage data will be exported from hive warehouse directory to MYSQL database using data\_export.sh command mentioned in the DataAnalysis.sh script.

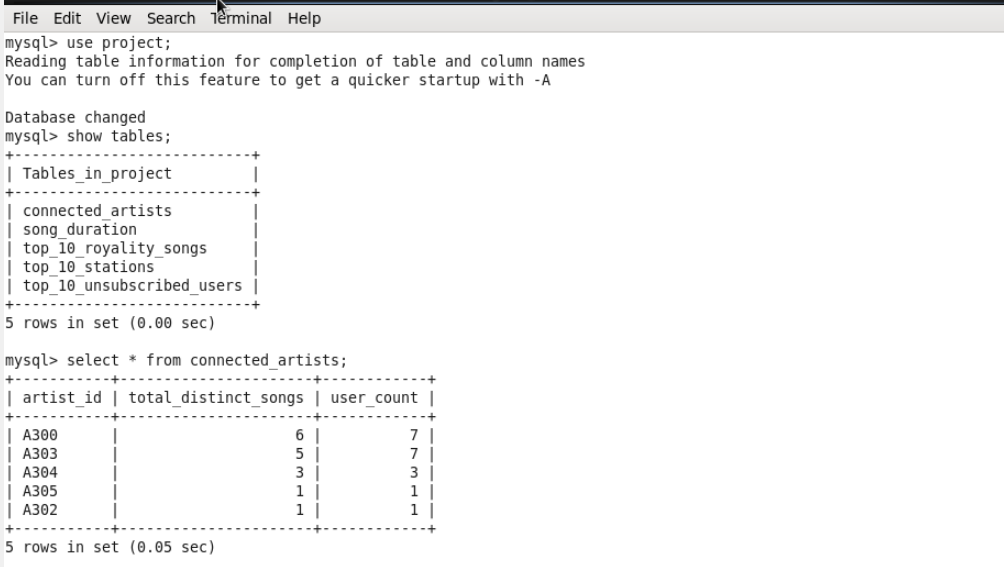


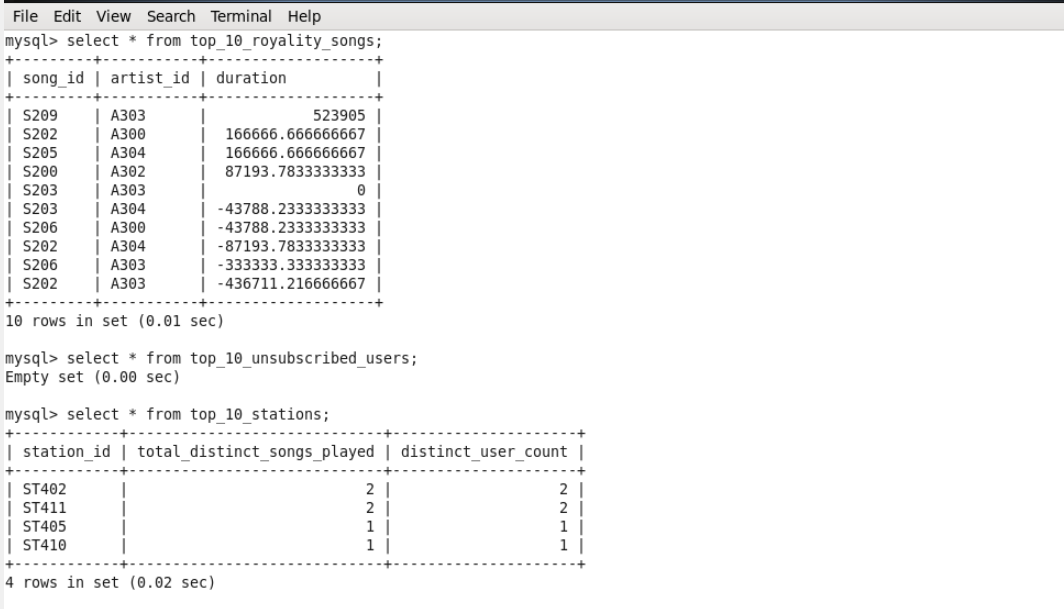


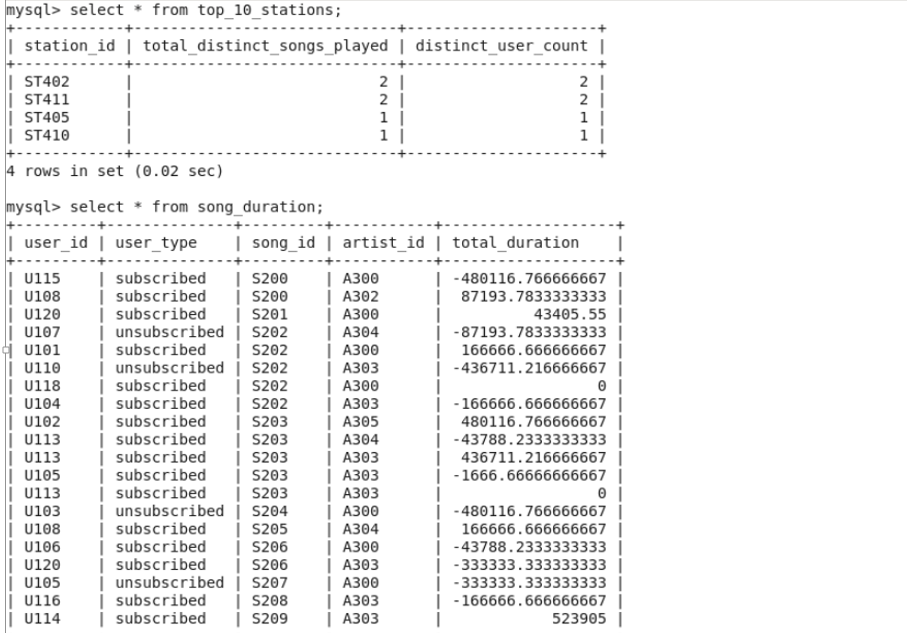


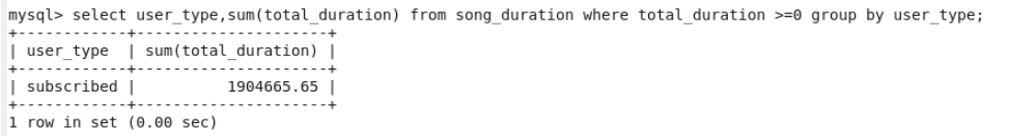
Now we can see the data exported successfully into the MYSQL Database for all the 5 queries.

**MYSQL:**





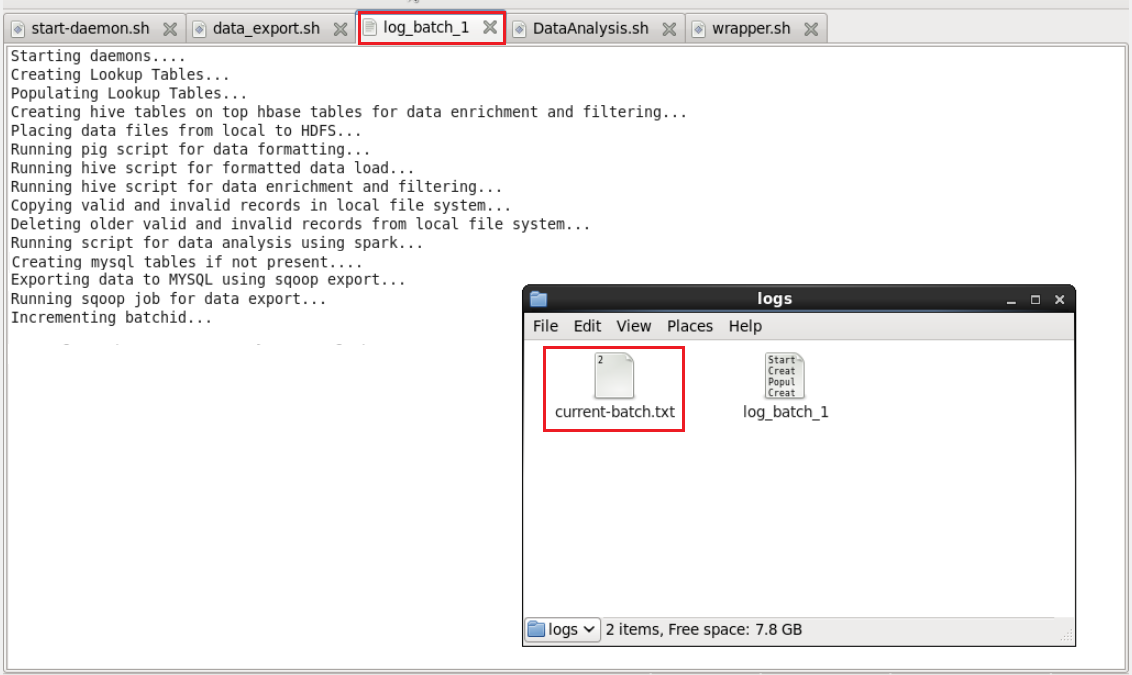




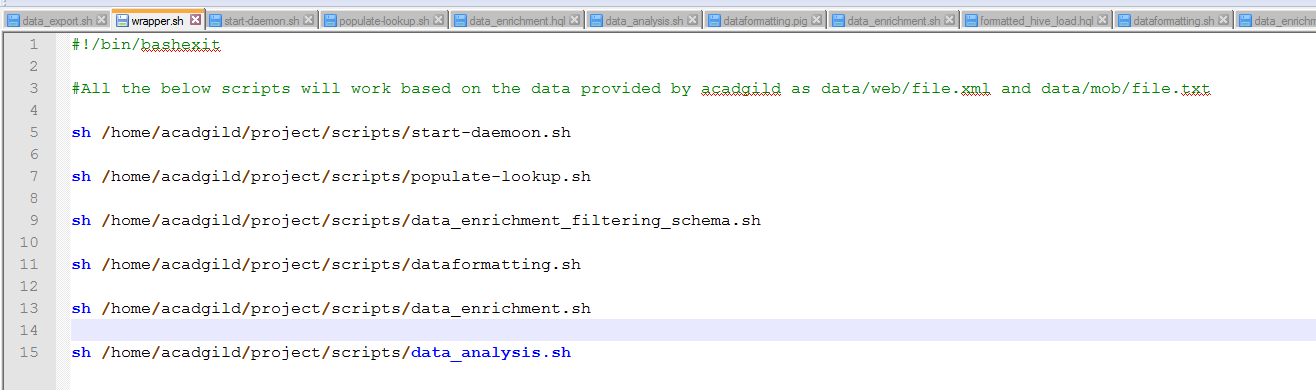
Now after exporting data into MySQL batchid will be incremented to 2 means one batch of data operations is successfully completed and new batch of data will be loaded for the analysis after every 3 hours.

C:\Users\Shanmuka.Bandi\Music\Data\data_export_9.JPG

We can check logs to track the behaviour 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.



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



* 1. **Job Scheduling:**

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

C:\Users\Shanmuka.Bandi\Downloads\crontab_1.JPG



