**Project - Music Data Analysis**

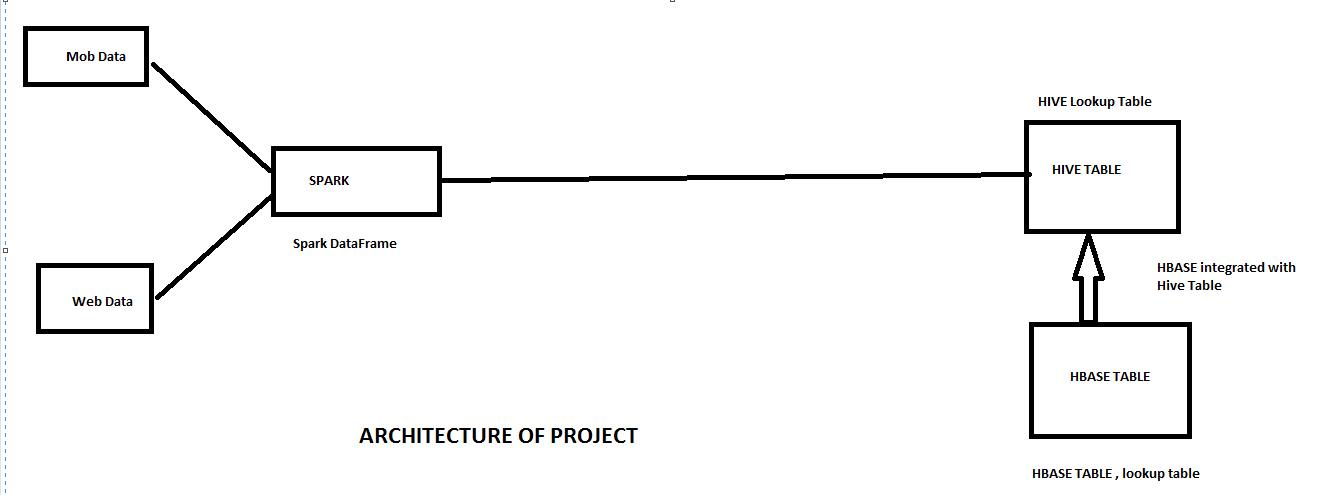
A leading music-catering company is planning to analyse large amount of data received from

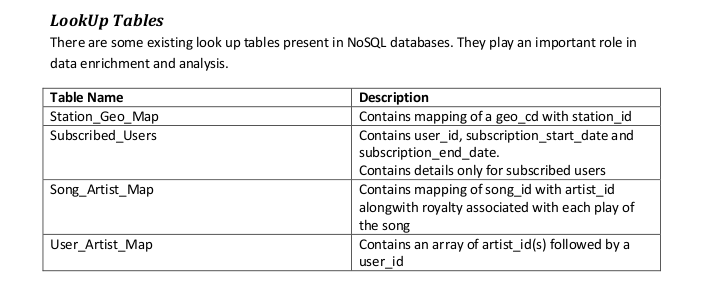
varieties of sources, namely mobile app and website to track the behaviour of users, classify users,

calculate royalties associated with the song and make appropriate business strategies. The file server

receives data files periodically after every 3 hours.

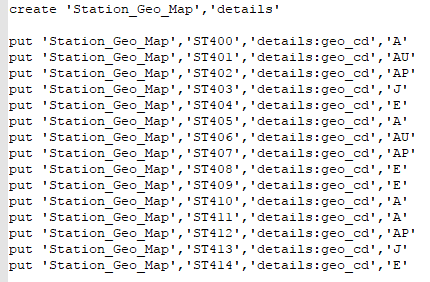
**ARCHITECTURE OF THE PROJECT**

****

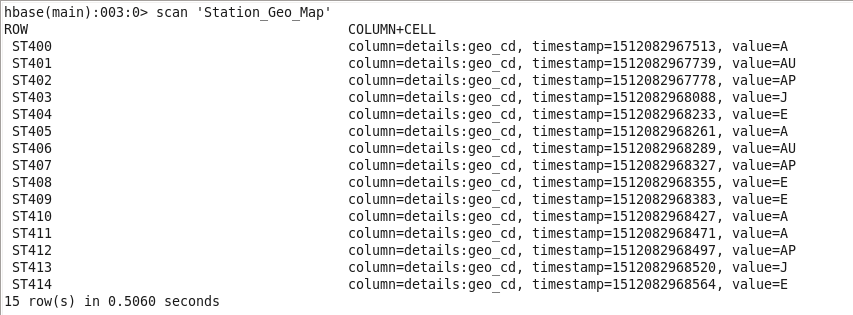


**CREATING HBASE (LOOKUP TABLES)**

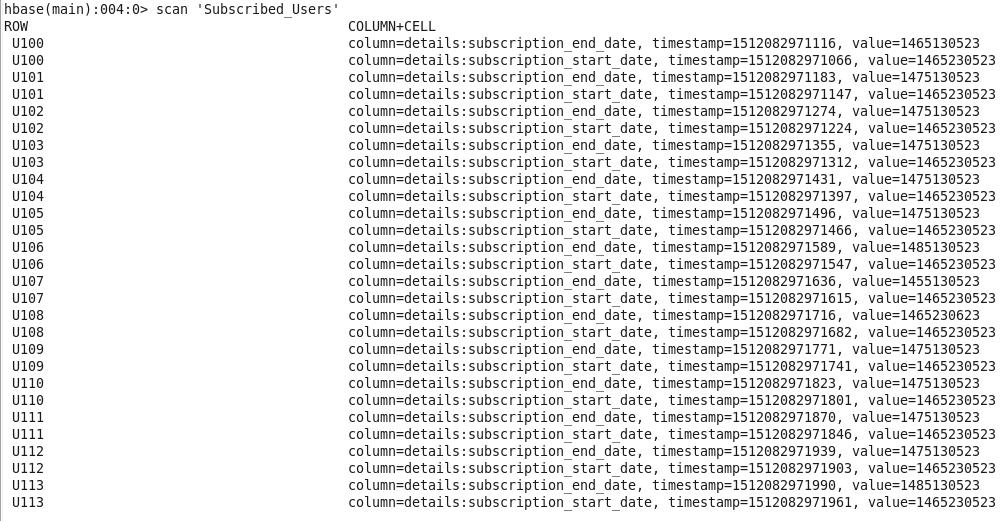
**CODE:**

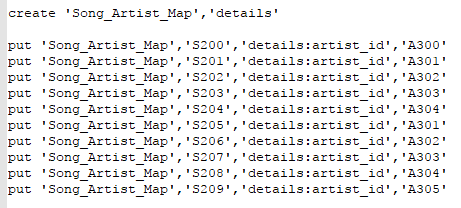


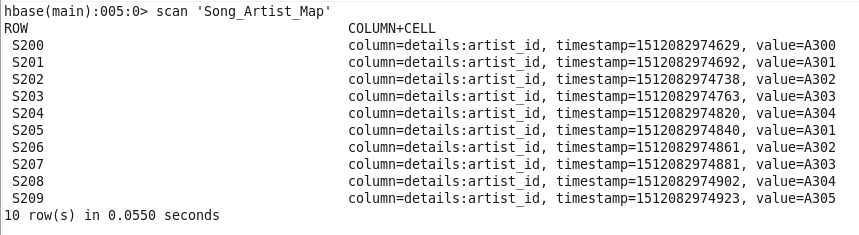
**OUTPUT:**

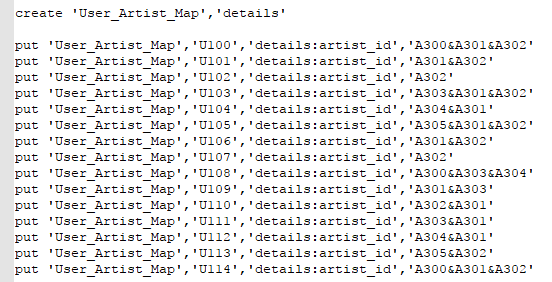


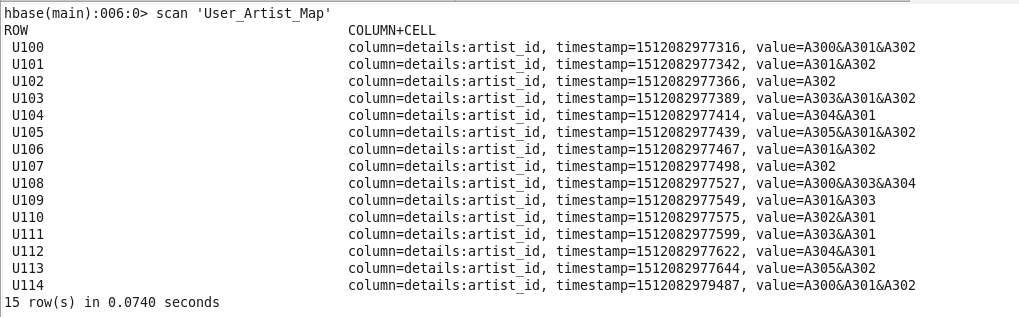




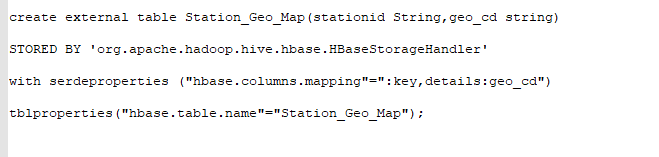


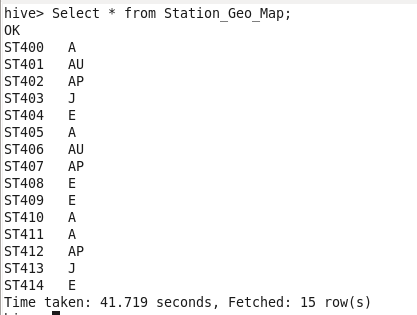


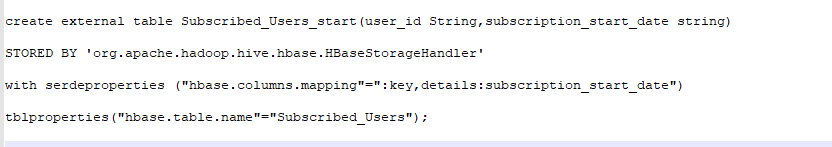


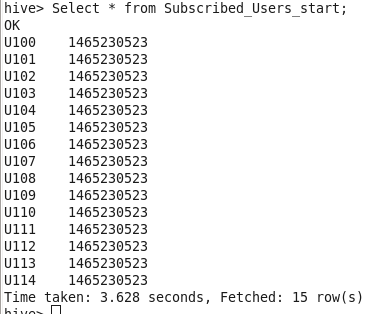


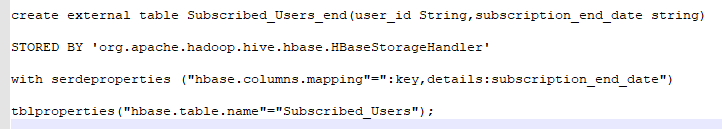
**CREATING HIVE TABLES FROM THESE HBASE LOOK UP TABLES (Integrating hbase with hive)**

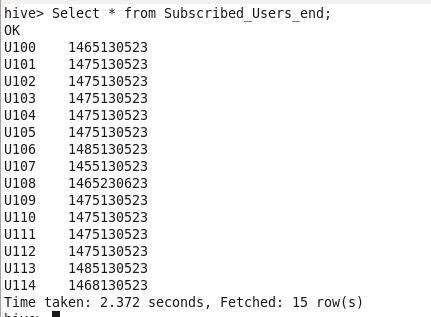


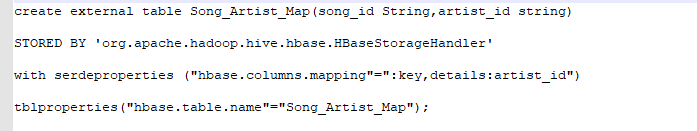


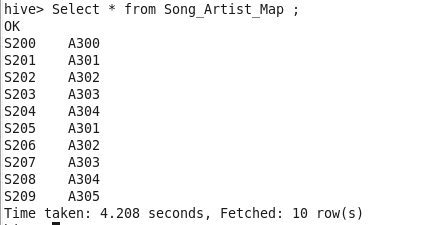


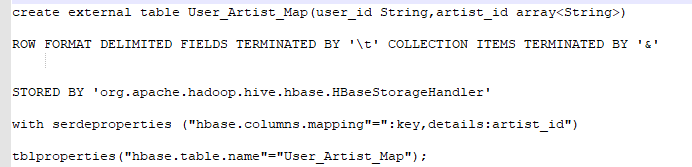


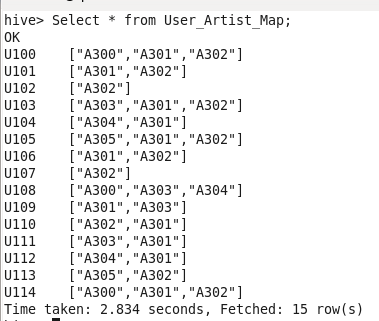












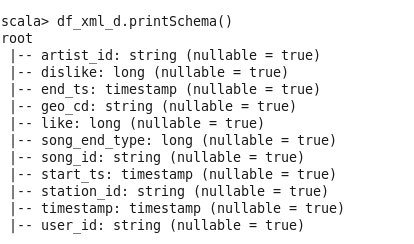
**Loading packages for xml and csv, opening spark shell**

spark-shell --packages com.databricks:spark-xml\_2.10:0.4.1,com.databricks:spark-csv\_2.10:1.5.0

**Import XML Data file**



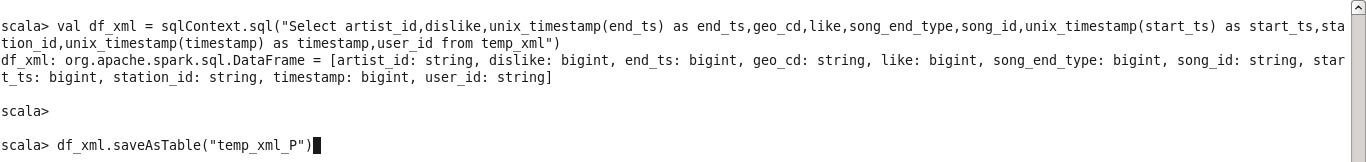
**Show schema of the file**

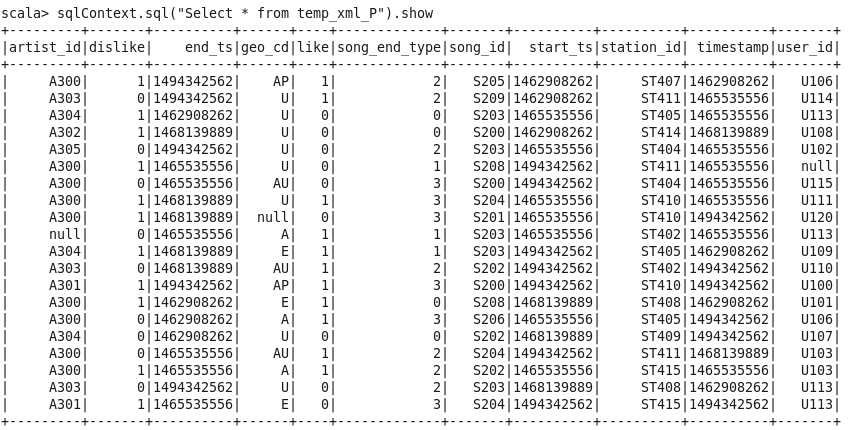


**Register temporary table for execution**



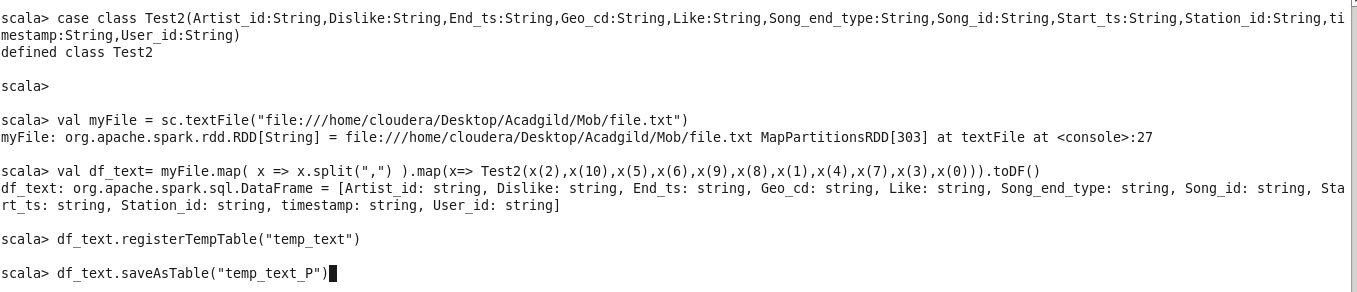
**Changing time to timestamp**



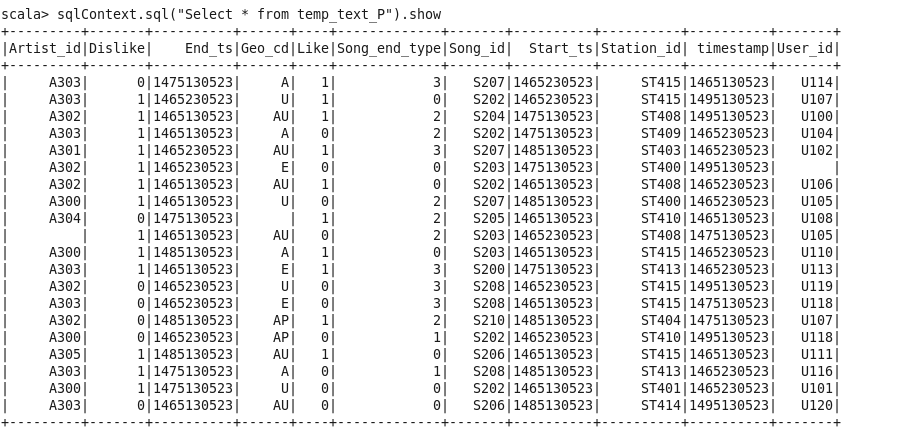


**Loading data from textfile**

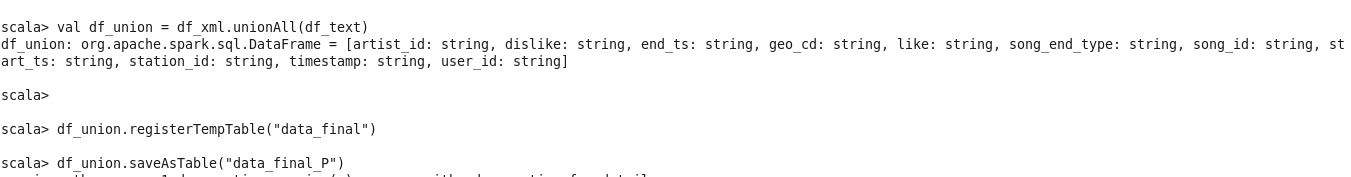
**--Creating case class , and loading the textfile**



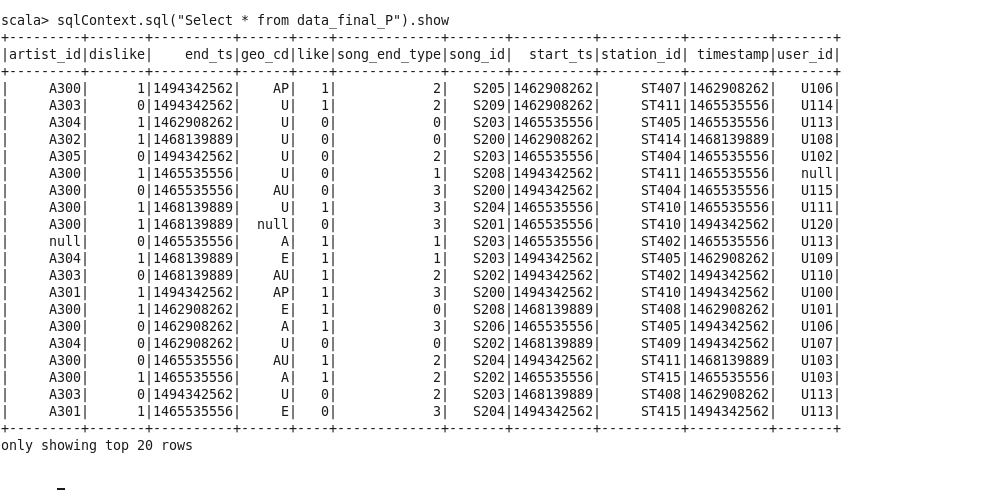
**Show the table data**



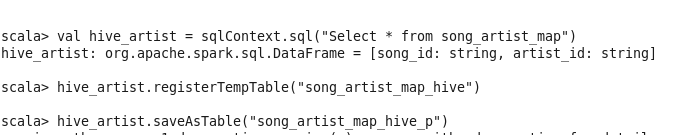
**UNIOIN our data the data from MOB AND WEB using union all on the datafranme**

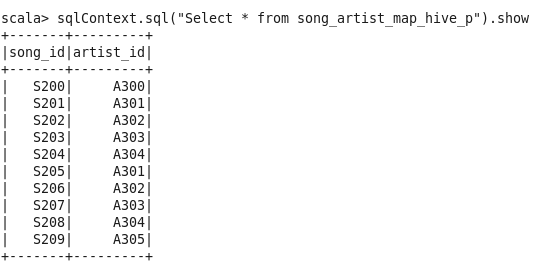


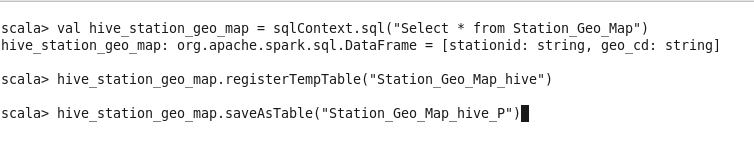
**SHOWING THE DATA**

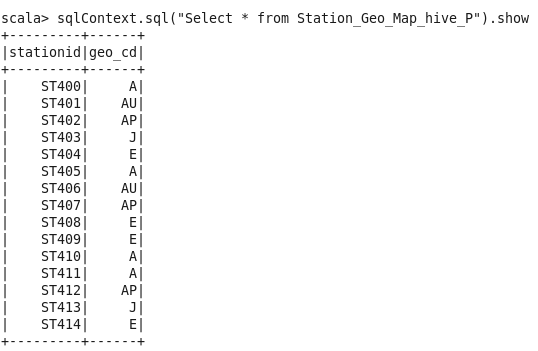


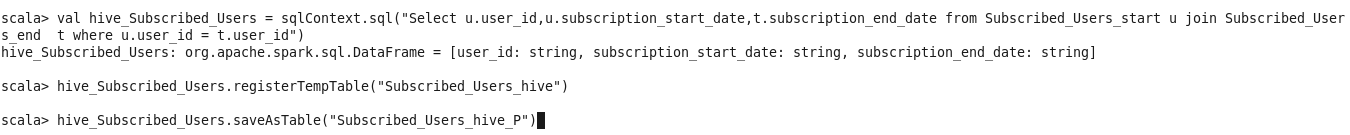
**LOADING DATA FROM HIVE TABLE IN SPARK SQL , HIVE TABLE ARE LOOKUP TABLES WHICH WE INTEGRATED FROM HBASE**

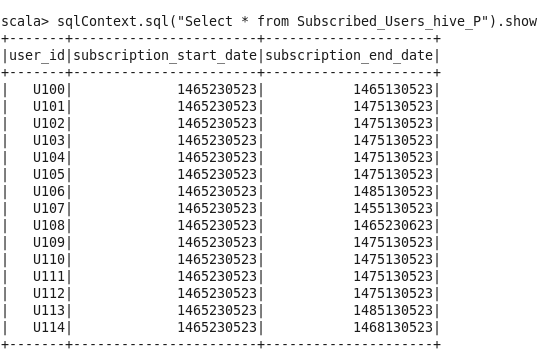


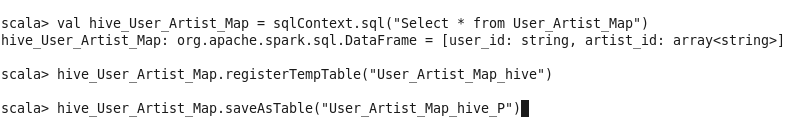


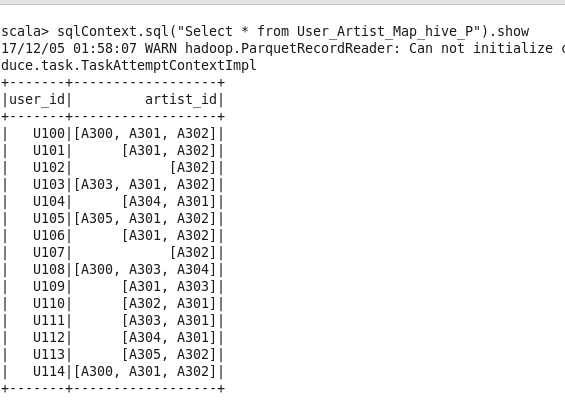






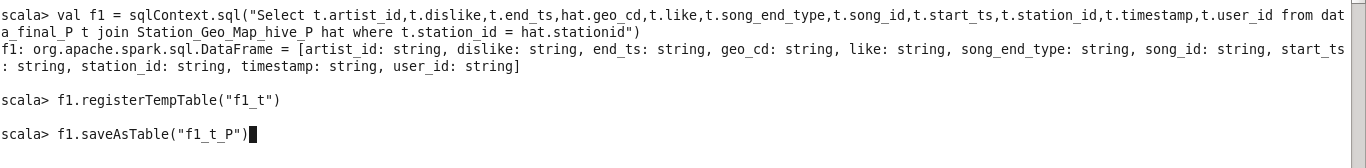


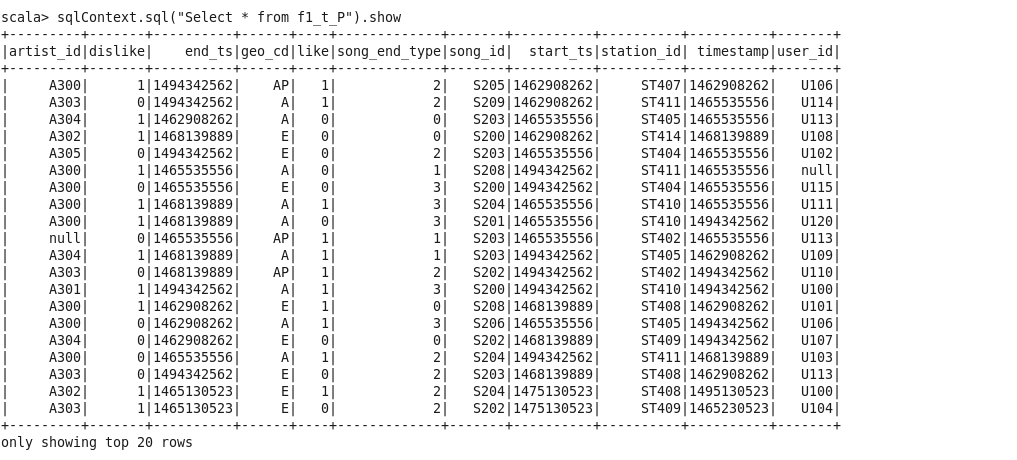




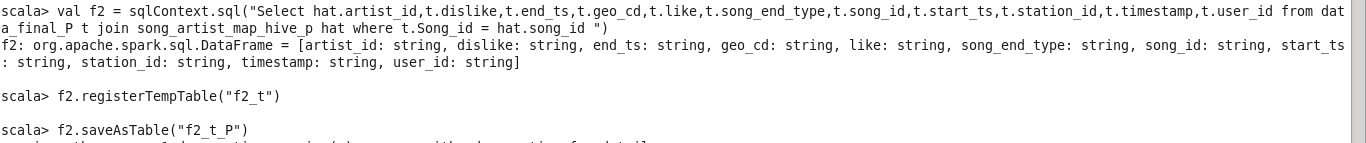
**---------------------------------------------------------DATA ENRICHMENT--------------------------------------------------**

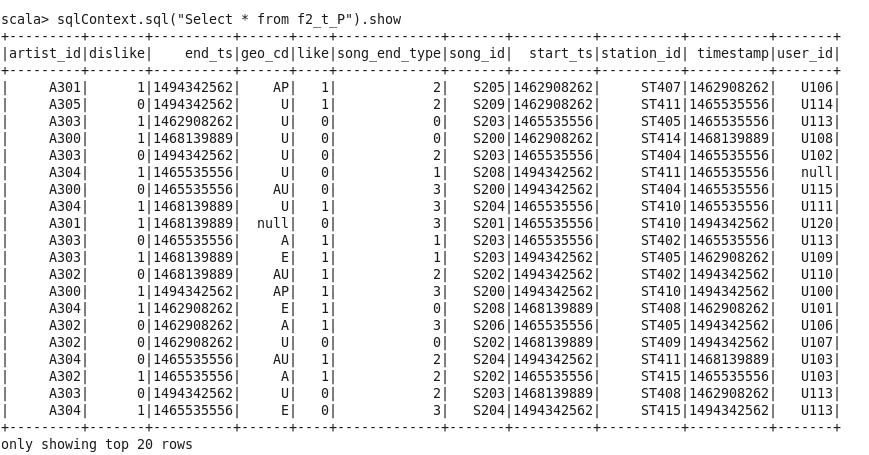
**Removing null or absent values from Geo\_cd using lookup field Station\_id from Station\_Geo\_Map**



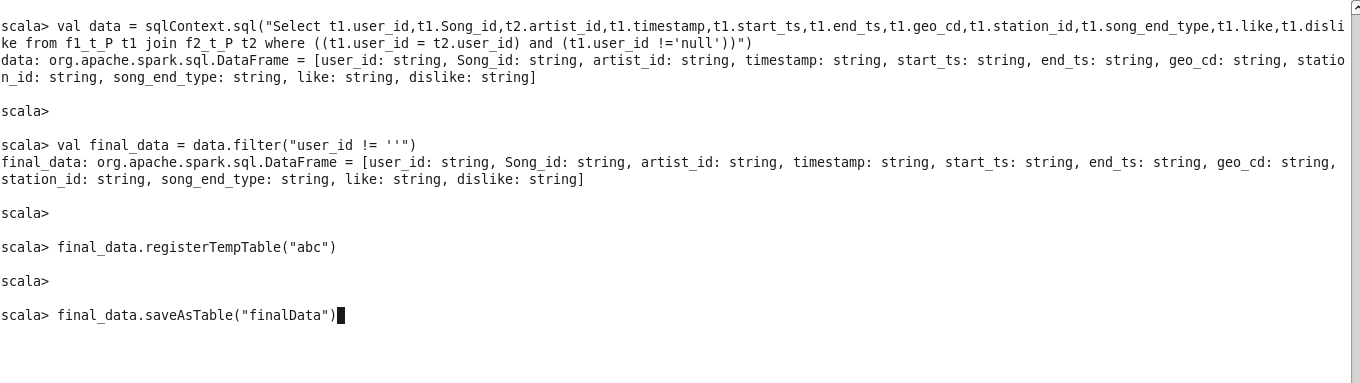


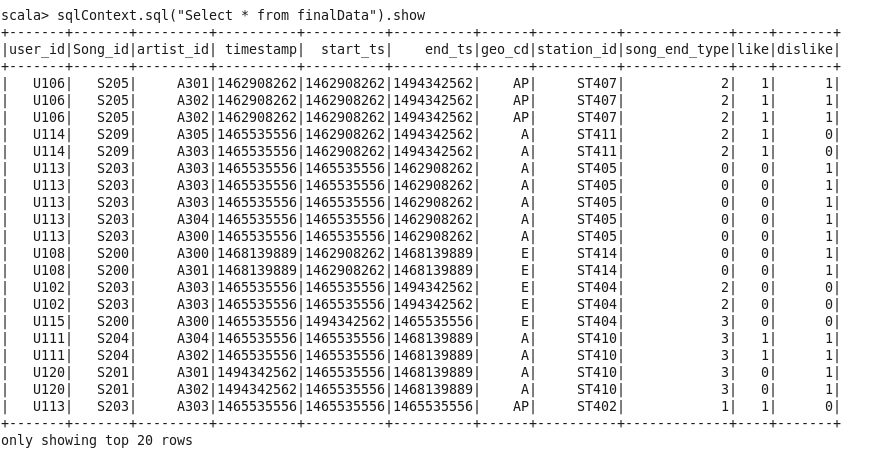
**Removing null or absent values from Artist using lookup field Song\_id from song\_artist\_map**





**--------------------------FINAL DATA JOINING AND FILTERING INVALID RECORDS--------------------------------**

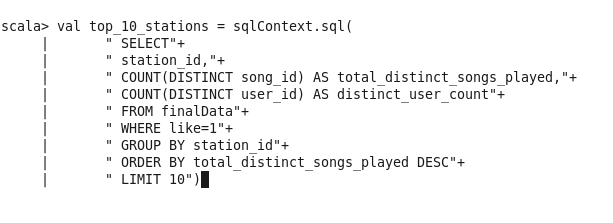




**------------------------------Data Analysis (SHOULD BE IMPLEMETED IN SPARK)----------------------------------**

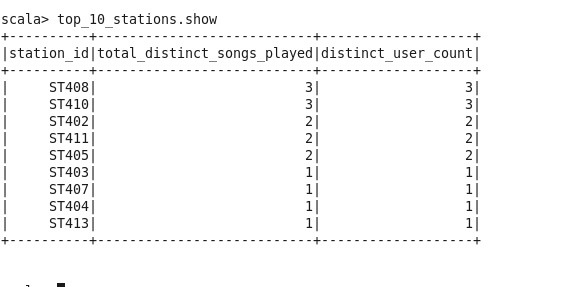
**1) Determine top 10 station\_id(s) where maximum number of songs were played, which were liked by unique users.**

**CODE:**



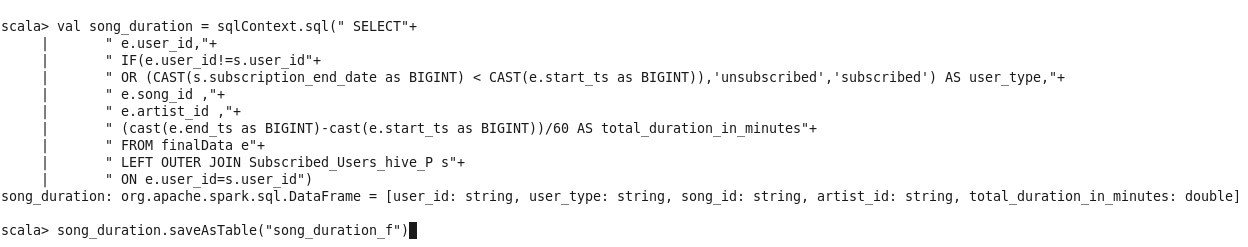
CODE: We are using Spark Sql for analysis , so we have used group by and order by for necessary analysis. Count is used for counting songs and users count

**OUTPUT**



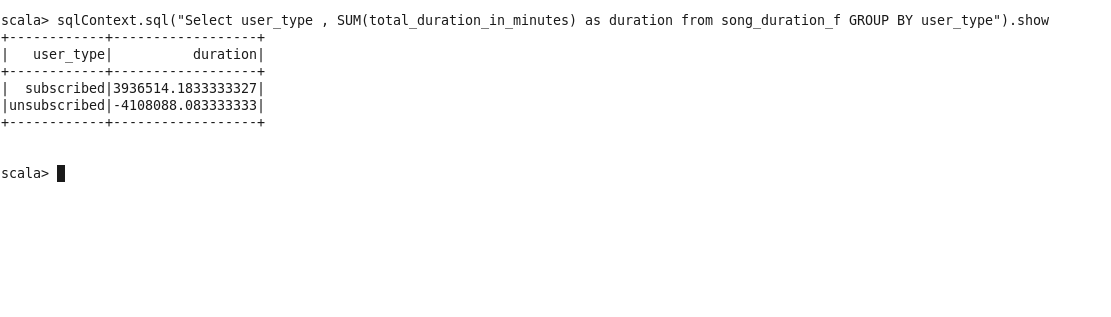
**2) Determine total duration of songs played by each type of user, where type of user can be 'subscribed' or 'unsubscribed'. An unsubscribed user is the one whose record is either not present in Subscribed\_users lookup table or has subscription\_end\_date earlier than the timestamp of the song played by him.**

**CODE:**

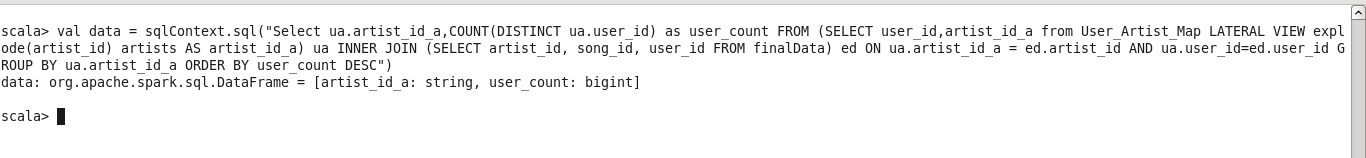


We are using lookup tables for the analysis , we have cast the subs end data and start date to big int

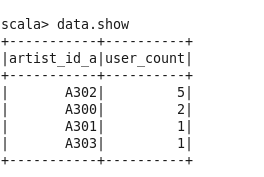
And joining with the lookup table for necessary analysis .



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



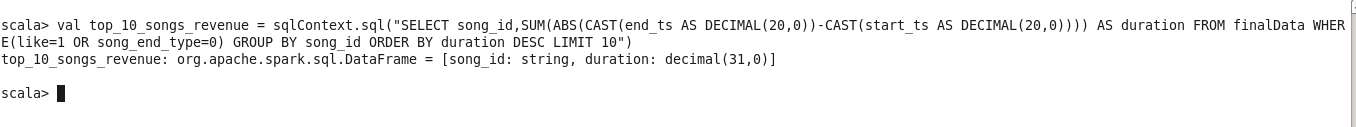
**OUTPUT**



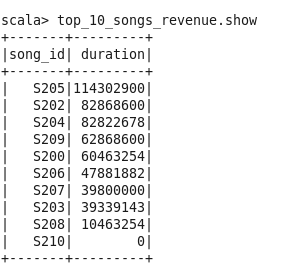
We got artist id which is highest connected, in this we have use lateral view and lookup table and using Join we have got final output

**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.**

**CODE:**

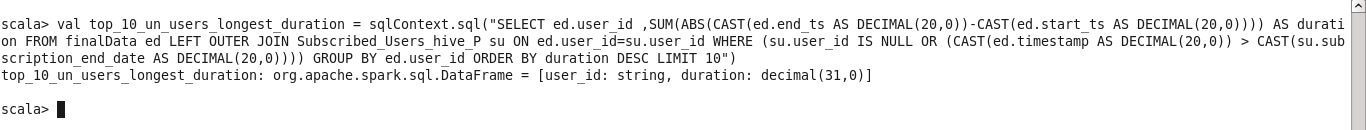


**OUTPUT:**



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

**CODE:**



**OUTPUT:**

