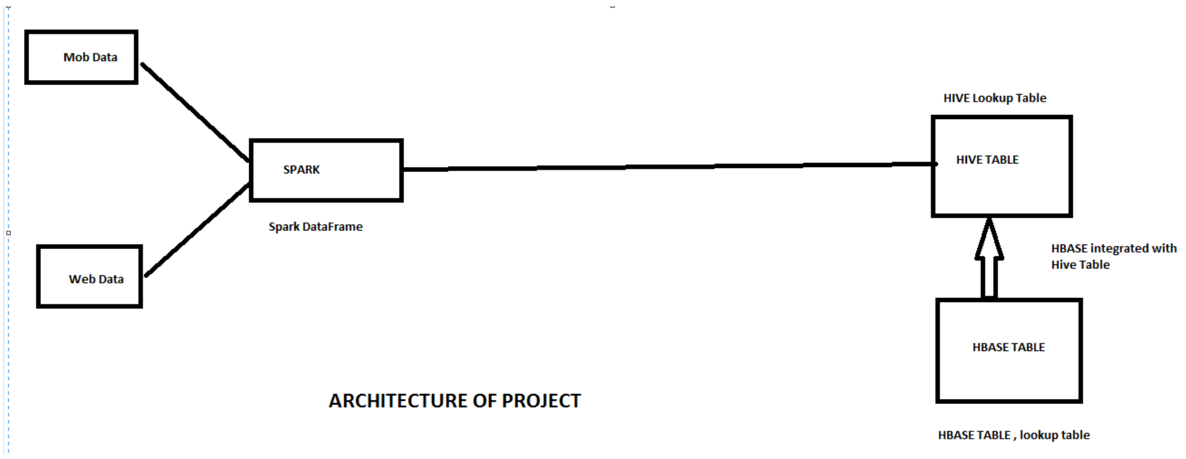
Project - Music Data Analysis

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



ARCHITECTURE OF PROJECT

LookUp Tables

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

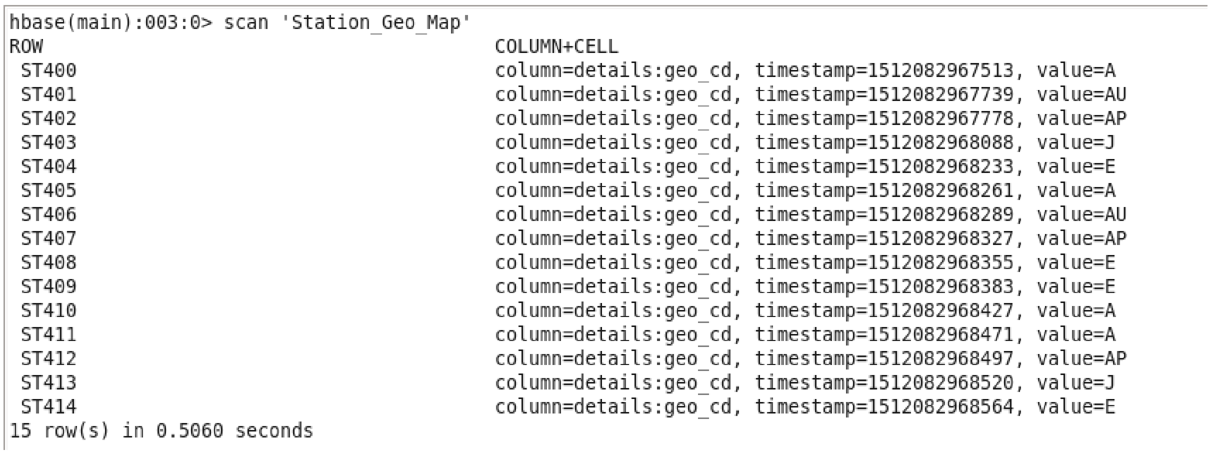
Table Name	Description
Station_Geo_Map	Contains mapping of a geo_cd with station_id
Subscribed_Users	Contains user_id, subscription_start_date and subscription_end_date. Contains details only for subscribed users
Song_Artist_Map	Contains mapping of song_id with artist_id alongwith royalty associated with each play of the song
User_Artist_Map	Contains an array of artist_id(s) followed by a user_id

CREATING HBASE (LOOKUP TABLES)

CODE:

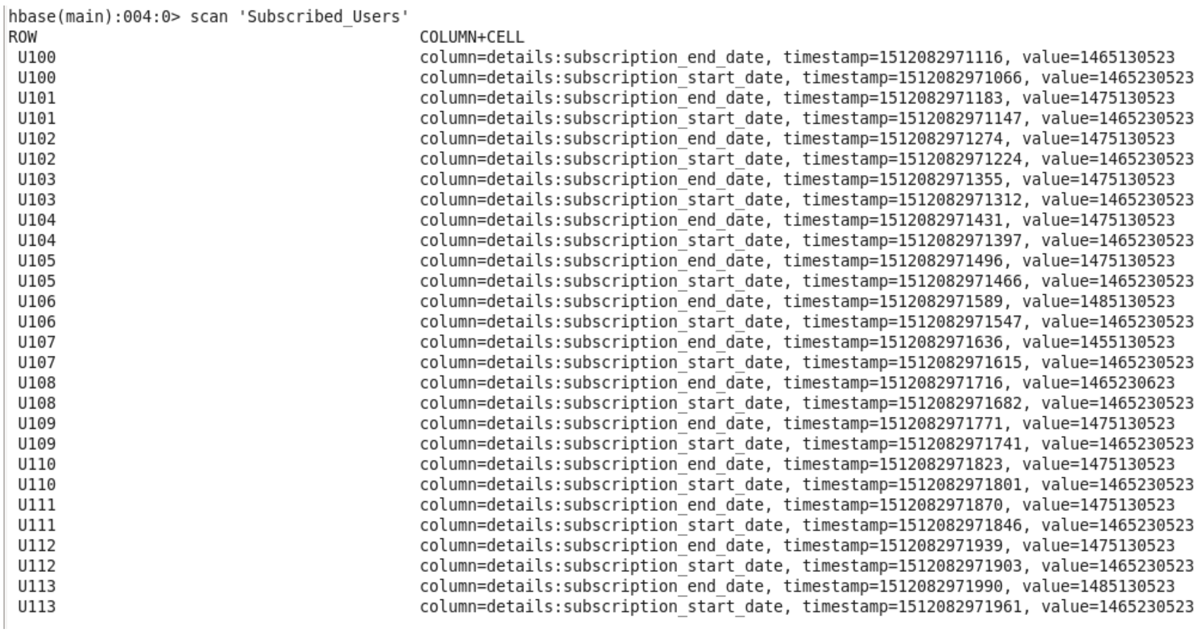
```
create 'Station_Geo_Map','details'

put 'Station_Geo_Map','ST400','details:geo_cd','A'
put 'Station_Geo_Map','ST401','details:geo_cd','AU'
put 'Station_Geo_Map','ST402','details:geo_cd','AP'
put 'Station_Geo_Map','ST403','details:geo_cd','J'
put 'Station_Geo_Map','ST404','details:geo_cd','E'
put 'Station_Geo_Map','ST405','details:geo_cd','A'
put 'Station_Geo_Map','ST406','details:geo_cd','AU'
put 'Station_Geo_Map','ST407','details:geo_cd','AP'
put 'Station_Geo_Map','ST408','details:geo_cd','E'
put 'Station_Geo_Map','ST409','details:geo_cd','E'
put 'Station_Geo_Map','ST410','details:geo_cd','A'
put 'Station_Geo_Map','ST411','details:geo_cd','A'
put 'Station_Geo_Map','ST412','details:geo_cd','AP'
put 'Station_Geo_Map','ST413','details:geo_cd','J'
put 'Station_Geo_Map','ST414','details:geo_cd','E'
```

OUTPUT:



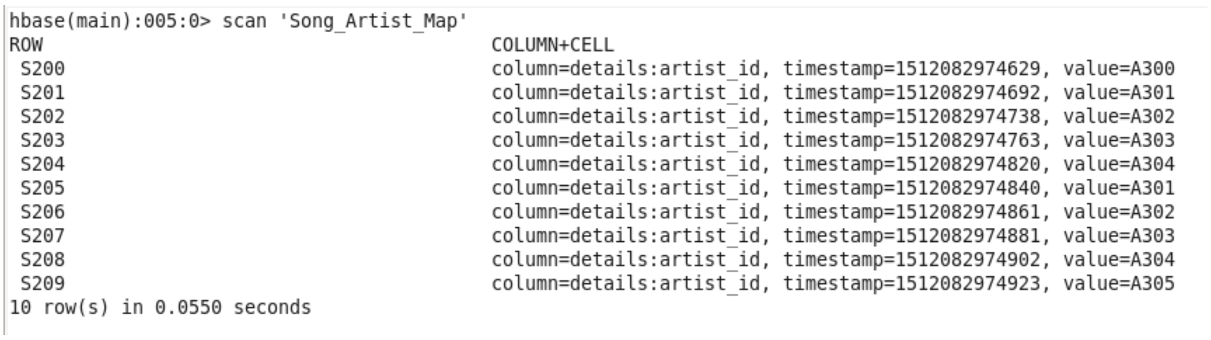
```
hbase(main):003:0> scan 'Station_Geo_Map'
ROW                                        COLUMN+CELL
 ST400                                     column=details:geo_cd, timestamp=1512082967513, value=A
 ST401                                     column=details:geo_cd, timestamp=1512082967739, value=AU
 ST402                                     column=details:geo_cd, timestamp=1512082967778, value=AP
 ST403                                     column=details:geo_cd, timestamp=1512082968088, value=J
 ST404                                     column=details:geo_cd, timestamp=1512082968233, value=E
 ST405                                     column=details:geo_cd, timestamp=1512082968261, value=A
 ST406                                     column=details:geo_cd, timestamp=1512082968289, value=AU
 ST407                                     column=details:geo_cd, timestamp=1512082968327, value=AP
 ST408                                     column=details:geo_cd, timestamp=1512082968355, value=E
 ST409                                     column=details:geo_cd, timestamp=1512082968383, value=E
 ST410                                     column=details:geo_cd, timestamp=1512082968427, value=A
 ST411                                     column=details:geo_cd, timestamp=1512082968471, value=A
 ST412                                     column=details:geo_cd, timestamp=1512082968497, value=AP
 ST413                                     column=details:geo_cd, timestamp=1512082968520, value=J
 ST414                                     column=details:geo_cd, timestamp=1512082968564, value=E
15 row(s) in 0.5060 seconds
```

```
create 'Subscribed_Users','details'

put 'Subscribed_Users','U100','details:subscription_start_date','1465230523'
put 'Subscribed_Users','U100','details:subscription_end_date','1465130523'
put 'Subscribed_Users','U101','details:subscription_start_date','1465230523'
put 'Subscribed_Users','U101','details:subscription_end_date','1475130523'
put 'Subscribed_Users','U102','details:subscription_start_date','1465230523'
put 'Subscribed_Users','U102','details:subscription_end_date','1475130523'
put 'Subscribed_Users','U103','details:subscription_start_date','1465230523'
put 'Subscribed_Users','U103','details:subscription_end_date','1475130523'
put 'Subscribed_Users','U104','details:subscription_start_date','1465230523'
put 'Subscribed_Users','U104','details:subscription_end_date','1475130523'
put 'Subscribed_Users','U105','details:subscription_start_date','1465230523'
put 'Subscribed_Users','U105','details:subscription_end_date','1475130523'
put 'Subscribed_Users','U106','details:subscription_start_date','1465230523'
put 'Subscribed_Users','U106','details:subscription_end_date','1485130523'
put 'Subscribed_Users','U107','details:subscription_start_date','1465230523'
put 'Subscribed_Users','U107','details:subscription_end_date','1455130523'
put 'Subscribed_Users','U108','details:subscription_start_date','1465230523'
put 'Subscribed_Users','U108','details:subscription_end_date','1465230623'
put 'Subscribed_Users','U109','details:subscription_start_date','1465230523'
put 'Subscribed_Users','U109','details:subscription_end_date','1475130523'
put 'Subscribed_Users','U110','details:subscription_start_date','1465230523'
put 'Subscribed_Users','U110','details:subscription_end_date','1475130523'
put 'Subscribed_Users','U111','details:subscription_start_date','1465230523'
put 'Subscribed_Users','U111','details:subscription_end_date','1475130523'
put 'Subscribed_Users','U112','details:subscription_start_date','1465230523'
put 'Subscribed_Users','U112','details:subscription_end_date','1475130523'
put 'Subscribed_Users','U113','details:subscription_start_date','1465230523'
put 'Subscribed_Users','U113','details:subscription_end_date','1485130523'
put 'Subscribed_Users','U114','details:subscription_start_date','1465230523'
put 'Subscribed_Users','U114','details:subscription_end_date','1468130523'
```



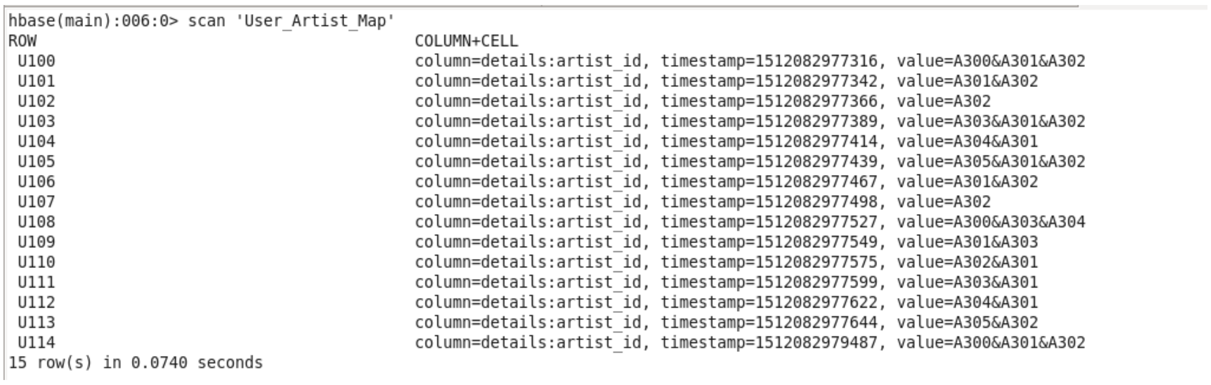
```
hbase(main):004:0> scan 'Subscribed_Users'
ROW                                       COLUMN+CELL
 U100                                     column=details:subscription_end_date, timestamp=1512082971116, value=1465130523
 U100                                     column=details:subscription_start_date, timestamp=1512082971066, value=1465230523
 U101                                     column=details:subscription_end_date, timestamp=1512082971183, value=1475130523
 U101                                     column=details:subscription_start_date, timestamp=1512082971147, value=1465230523
 U102                                     column=details:subscription_end_date, timestamp=1512082971274, value=1475130523
 U102                                     column=details:subscription_start_date, timestamp=1512082971224, value=1465230523
 U103                                     column=details:subscription_end_date, timestamp=1512082971355, value=1475130523
 U103                                     column=details:subscription_start_date, timestamp=1512082971312, value=1465230523
 U104                                     column=details:subscription_end_date, timestamp=1512082971431, value=1475130523
 U104                                     column=details:subscription_start_date, timestamp=1512082971397, value=1465230523
 U105                                     column=details:subscription_end_date, timestamp=1512082971496, value=1475130523
 U105                                     column=details:subscription_start_date, timestamp=1512082971466, value=1465230523
 U106                                     column=details:subscription_end_date, timestamp=1512082971589, value=1485130523
 U106                                     column=details:subscription_start_date, timestamp=1512082971547, value=1465230523
 U107                                     column=details:subscription_end_date, timestamp=1512082971636, value=1455130523
 U107                                     column=details:subscription_start_date, timestamp=1512082971615, value=1465230523
 U108                                     column=details:subscription_end_date, timestamp=1512082971716, value=1465230623
 U108                                     column=details:subscription_start_date, timestamp=1512082971682, value=1465230523
 U109                                     column=details:subscription_end_date, timestamp=1512082971771, value=1475130523
 U109                                     column=details:subscription_start_date, timestamp=1512082971741, value=1465230523
 U110                                     column=details:subscription_end_date, timestamp=1512082971823, value=1475130523
 U110                                     column=details:subscription_start_date, timestamp=1512082971801, value=1465230523
 U111                                     column=details:subscription_end_date, timestamp=1512082971870, value=1475130523
 U111                                     column=details:subscription_start_date, timestamp=1512082971846, value=1465230523
 U112                                     column=details:subscription_end_date, timestamp=1512082971939, value=1475130523
 U112                                     column=details:subscription_start_date, timestamp=1512082971903, value=1465230523
 U113                                     column=details:subscription_end_date, timestamp=1512082971990, value=1485130523
 U113                                     column=details:subscription_start_date, timestamp=1512082971961, value=1465230523
```

```
create 'Song_Artist_Map','details'

put 'Song_Artist_Map','S200','details:artist_id','A300'
put 'Song_Artist_Map','S201','details:artist_id','A301'
put 'Song_Artist_Map','S202','details:artist_id','A302'
put 'Song_Artist_Map','S203','details:artist_id','A303'
put 'Song_Artist_Map','S204','details:artist_id','A304'
put 'Song_Artist_Map','S205','details:artist_id','A301'
put 'Song_Artist_Map','S206','details:artist_id','A302'
put 'Song_Artist_Map','S207','details:artist_id','A303'
put 'Song_Artist_Map','S208','details:artist_id','A304'
put 'Song_Artist_Map','S209','details:artist_id','A305'
```



```
hbase(main):005:0> scan 'Song_Artist_Map'
ROW                                        COLUMN+CELL
 S200                                      column=details:artist_id, timestamp=1512082974629, value=A300
 S201                                      column=details:artist_id, timestamp=1512082974692, value=A301
 S202                                      column=details:artist_id, timestamp=1512082974738, value=A302
 S203                                      column=details:artist_id, timestamp=1512082974763, value=A303
 S204                                      column=details:artist_id, timestamp=1512082974820, value=A304
 S205                                      column=details:artist_id, timestamp=1512082974840, value=A301
 S206                                      column=details:artist_id, timestamp=1512082974861, value=A302
 S207                                      column=details:artist_id, timestamp=1512082974881, value=A303
 S208                                      column=details:artist_id, timestamp=1512082974902, value=A304
 S209                                      column=details:artist_id, timestamp=1512082974923, value=A305
10 row(s) in 0.0550 seconds
```

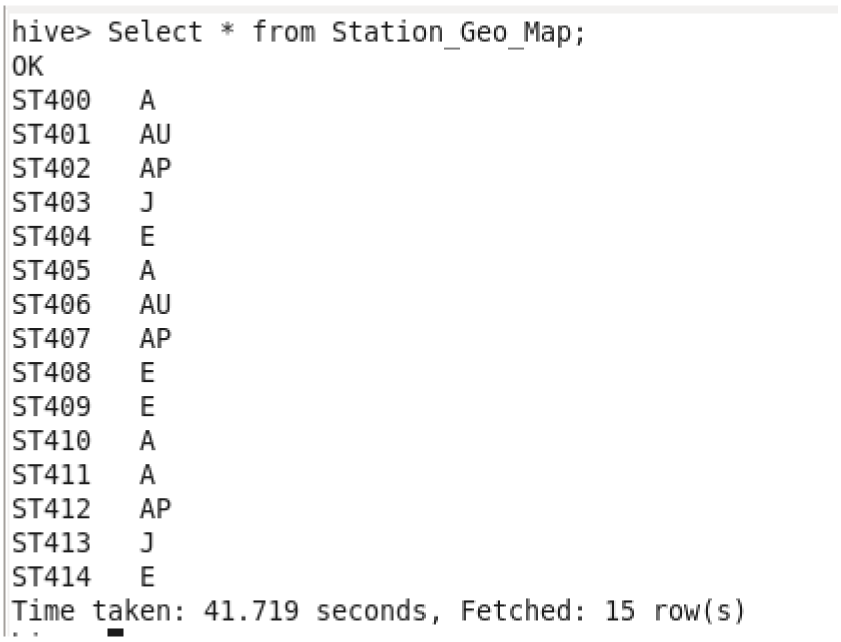
```
create 'User_Artist_Map','details'

put 'User_Artist_Map','U100','details:artist_id','A300&A301&A302'
put 'User_Artist_Map','U101','details:artist_id','A301&A302'
put 'User_Artist_Map','U102','details:artist_id','A302'
put 'User_Artist_Map','U103','details:artist_id','A303&A301&A302'
put 'User_Artist_Map','U104','details:artist_id','A304&A301'
put 'User_Artist_Map','U105','details:artist_id','A305&A301&A302'
put 'User_Artist_Map','U106','details:artist_id','A301&A302'
put 'User_Artist_Map','U107','details:artist_id','A302'
put 'User_Artist_Map','U108','details:artist_id','A300&A303&A304'
put 'User_Artist_Map','U109','details:artist_id','A301&A303'
put 'User_Artist_Map','U110','details:artist_id','A302&A301'
put 'User_Artist_Map','U111','details:artist_id','A303&A301'
put 'User_Artist_Map','U112','details:artist_id','A304&A301'
put 'User_Artist_Map','U113','details:artist_id','A305&A302'
put 'User_Artist_Map','U114','details:artist_id','A300&A301&A302'
```



```
hbase(main):006:0> scan 'User_Artist_Map'
ROW                                      COLUMN+CELL
 U100                                    column=details:artist_id, timestamp=1512082977316, value=A300&A301&A302
 U101                                    column=details:artist_id, timestamp=1512082977342, value=A301&A302
 U102                                    column=details:artist_id, timestamp=1512082977366, value=A302
 U103                                    column=details:artist_id, timestamp=1512082977389, value=A303&A301&A302
 U104                                    column=details:artist_id, timestamp=1512082977414, value=A304&A301
 U105                                    column=details:artist_id, timestamp=1512082977439, value=A305&A301&A302
 U106                                    column=details:artist_id, timestamp=1512082977467, value=A301&A302
 U107                                    column=details:artist_id, timestamp=1512082977498, value=A302
 U108                                    column=details:artist_id, timestamp=1512082977527, value=A300&A303&A304
 U109                                    column=details:artist_id, timestamp=1512082977549, value=A301&A303
 U110                                    column=details:artist_id, timestamp=1512082977575, value=A302&A301
 U111                                    column=details:artist_id, timestamp=1512082977599, value=A303&A301
 U112                                    column=details:artist_id, timestamp=1512082977622, value=A304&A301
 U113                                    column=details:artist_id, timestamp=1512082977644, value=A305&A302
 U114                                    column=details:artist_id, timestamp=1512082979487, value=A300&A301&A302
15 row(s) in 0.0740 seconds
```

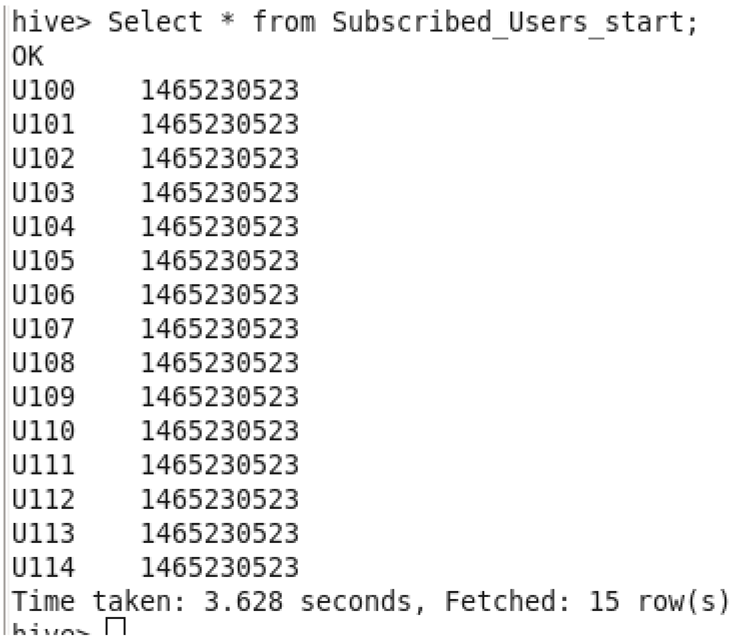
CREATING HIVE TABLES FROM THESE HBASE LOOK UP TABLES (Integrating hbase with hive)

```
create external table Station_Geo_Map(stationid String,geo_cd string)

STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'

with serdeproperties ("hbase.columns.mapping"=":key,details:geo_cd")

tblproperties("hbase.table.name"="Station_Geo_Map");
```



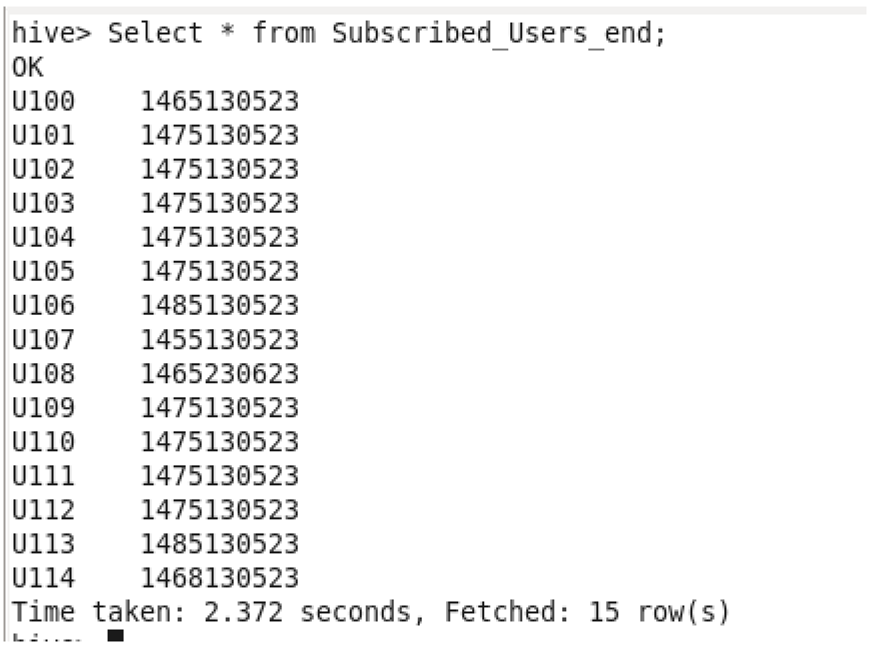
```
hive> Select * from Station_Geo_Map;
OK
ST400   A
ST401   AU
ST402   AP
ST403   J
ST404   E
ST405   A
ST406   AU
ST407   AP
ST408   E
ST409   E
ST410   A
ST411   A
ST412   AP
ST413   J
ST414   E
Time taken: 41.719 seconds, Fetched: 15 row(s)
```

```
create external table Subscribed_Users_start(user_id String,subscription_start_date string)

STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'

with serdeproperties ("hbase.columns.mapping"=":key,details:subscription_start_date")

tblproperties("hbase.table.name"="Subscribed_Users");
```

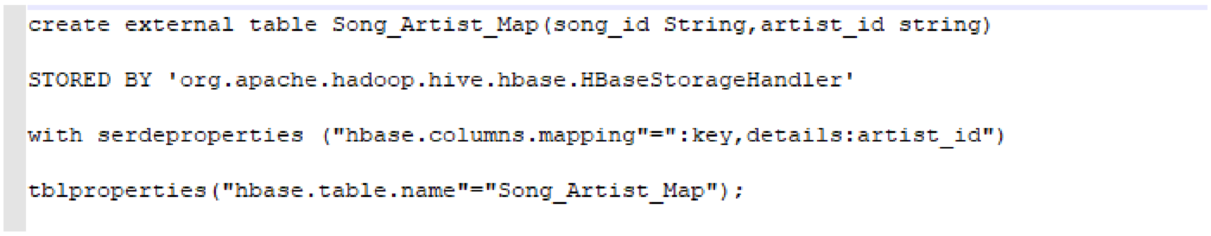


```
hive> Select * from Subscribed_Users_start;
OK
U100    1465230523
U101    1465230523
U102    1465230523
U103    1465230523
U104    1465230523
U105    1465230523
U106    1465230523
U107    1465230523
U108    1465230523
U109    1465230523
U110    1465230523
U111    1465230523
U112    1465230523
U113    1465230523
U114    1465230523
Time taken: 3.628 seconds, Fetched: 15 row(s)
hive>
```

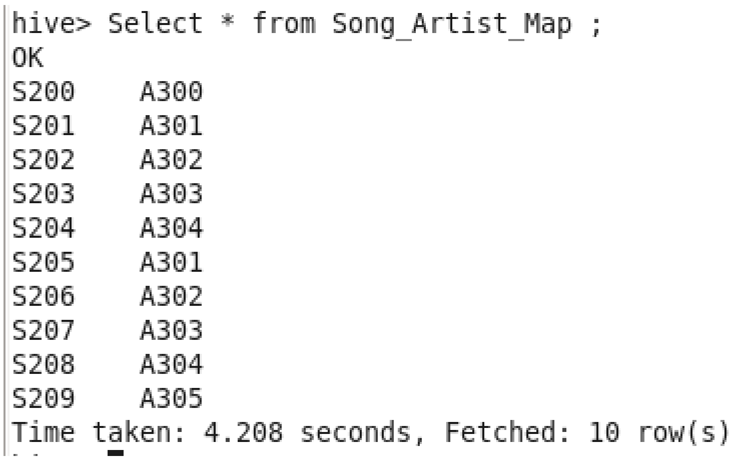
```
create external table Subscribed_Users_end(user_id String,subscription_end_date string)

STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'

with serdeproperties ("hbase.columns.mapping"=":key,details:subscription_end_date")

tblproperties("hbase.table.name"="Subscribed_Users");
```



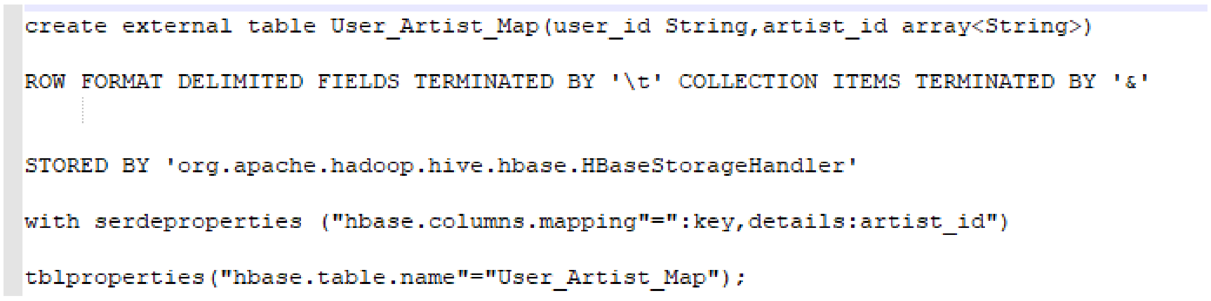
```
hive> Select * from Subscribed_Users_end;
OK
U100    1465130523
U101    1475130523
U102    1475130523
U103    1475130523
U104    1475130523
U105    1475130523
U106    1485130523
U107    1455130523
U108    1465230623
U109    1475130523
U110    1475130523
U111    1475130523
U112    1475130523
U113    1485130523
U114    1468130523
Time taken: 2.372 seconds, Fetched: 15 row(s)
hive>
```



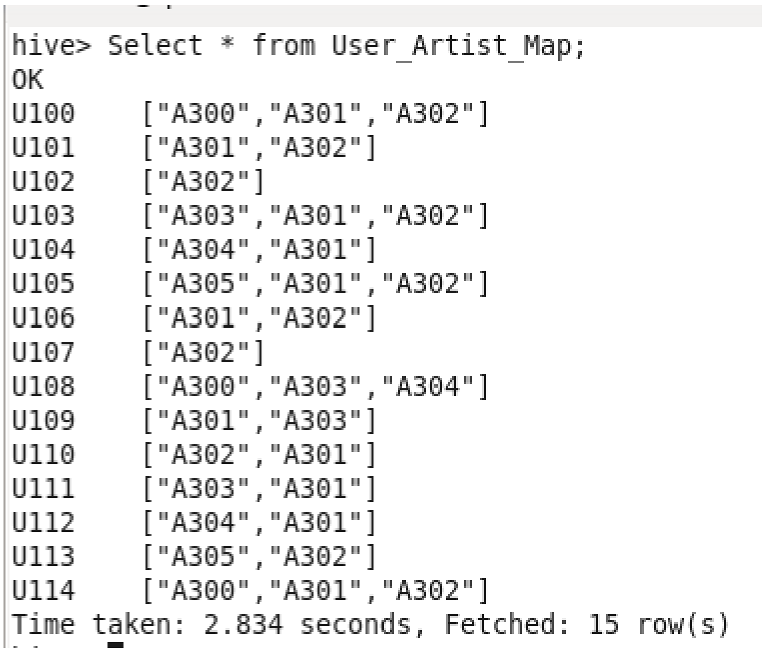
```
create external table Song_Artist_Map(song_id String,artist_id string)

STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'

with serdeproperties ("hbase.columns.mapping"=":key,details:artist_id")

tblproperties("hbase.table.name"="Song_Artist_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: 4.208 seconds, Fetched: 10 row(s)
```



```
create external table User_Artist_Map(user_id String,artist_id array<String>)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' COLLECTION ITEMS TERMINATED BY '&'


STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'

with serdeproperties ("hbase.columns.mapping"=":key,details:artist_id")

tblproperties("hbase.table.name"="User_Artist_Map");
```

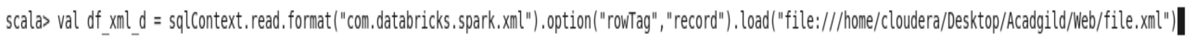


```
hive> Select * from User_Artist_Map;
OK
U100    ["A300","A301","A302"]
U101    ["A301","A302"]
U102    ["A302"]
U103    ["A303","A301","A302"]
U104    ["A304","A301"]
U105    ["A305","A301","A302"]
U106    ["A301","A302"]
U107    ["A302"]
U108    ["A300","A303","A304"]
U109    ["A301","A303"]
U110    ["A302","A301"]
U111    ["A303","A301"]
U112    ["A304","A301"]
U113    ["A305","A302"]
U114    ["A300","A301","A302"]
Time taken: 2.834 seconds, Fetched: 15 row(s)
```

Loading packages for xml and csv, opening spark shell

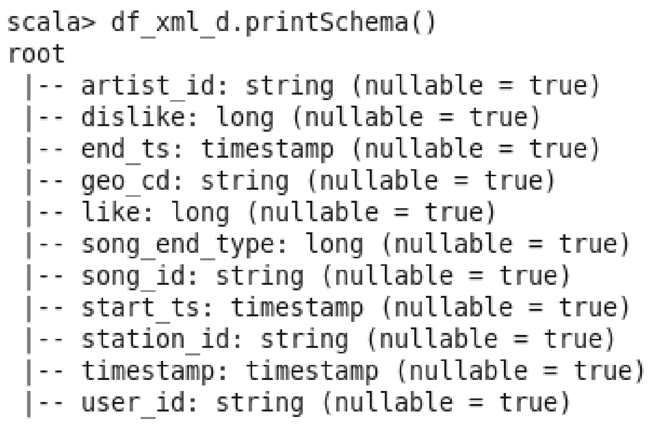
spark-shell --packages com.databricks:spark-xml_2.10:0.4.1,com.databricks:spark-csv_2.10:1.5.0

Import XML Data file



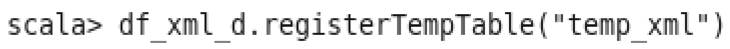
```
scala> val df_xml_d = sqlContext.read.format("com.databricks.spark.xml").option("rowTag","record").load("file:///home/cloudera/Desktop/Acadgild/Web/file.xml")
```

Show schema of the file



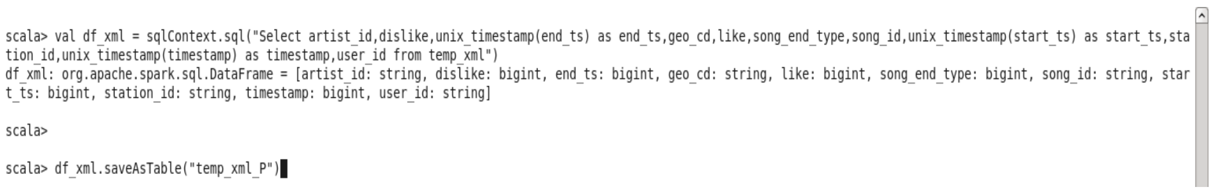
```
scala> df_xml_d.printSchema()
root
 |-- artist_id: string (nullable = true)
 |-- dislike: long (nullable = true)
 |-- end_ts: timestamp (nullable = true)
 |-- geo_cd: string (nullable = true)
 |-- like: long (nullable = true)
 |-- song_end_type: long (nullable = true)
 |-- song_id: string (nullable = true)
 |-- start_ts: timestamp (nullable = true)
 |-- station_id: string (nullable = true)
 |-- timestamp: timestamp (nullable = true)
 |-- user_id: string (nullable = true)
```

Register temporary table for execution

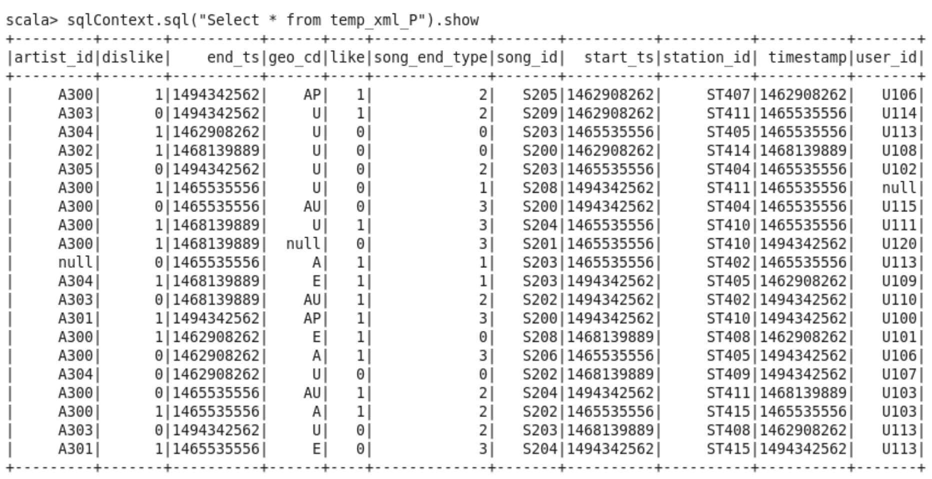


```
scala> df_xml_d.registerTempTable("temp_xml")
```

Changing time to timestamp



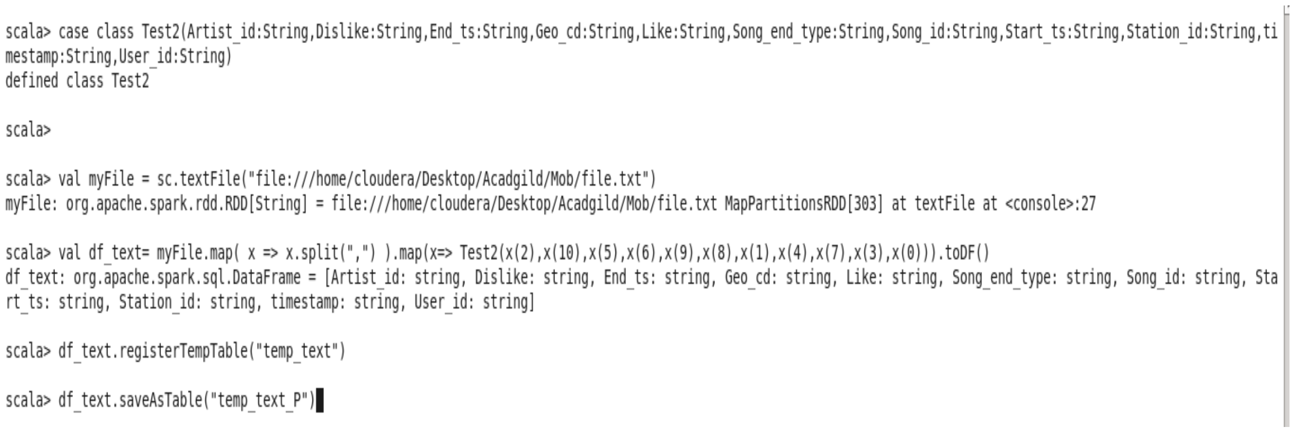
```
scala> val df_xml = sqlContext.sql("Select artist_id,dislike,unix_timestamp(end_ts) as end_ts,geo_cd,like,song_end_type,song_id,unix_timestamp(start_ts) as start_ts,sta
tion_id,unix_timestamp(timestamp) as timestamp,user_id from temp_xml")
df_xml: org.apache.spark.sql.DataFrame = [artist_id: string, dislike: bigint, end_ts: bigint, geo_cd: string, like: bigint, song_end_type: bigint, song_id: string, star
t_ts: bigint, station_id: string, timestamp: bigint, user_id: string]

scala>

scala> df_xml.saveAsTable("temp_xml_P")
```



```
scala> sqlContext.sql("Select * from temp_xml_P").show
+---------+-------+----------+------+----+-------------+-------+----------+----------+----------+-------+
|artist_id|dislike|    end_ts|geo_cd|like|song_end_type|song_id|  start_ts|station_id| timestamp|user_id|
+---------+-------+----------+------+----+-------------+-------+----------+----------+----------+-------+
|     A300|      1|1494342562|    AP|   1|            2|   S205|1462908262|     ST407|1462908262|   U106|
|     A303|      0|1494342562|     U|   1|            2|   S209|1462908262|     ST411|1465535556|   U114|
|     A304|      1|1462908262|     U|   0|            0|   S203|1465535556|     ST405|1465535556|   U113|
|     A302|      1|1468139889|     U|   0|            0|   S200|1462908262|     ST414|1468139889|   U108|
|     A305|      0|1494342562|     U|   0|            2|   S203|1465535556|     ST404|1465535556|   U102|
|     A300|      1|1465535556|     U|   0|            1|   S208|1494342562|     ST411|1465535556|   null|
|     A300|      0|1465535556|    AU|   0|            3|   S200|1494342562|     ST404|1465535556|   U115|
|     A300|      1|1468139889|     U|   1|            3|   S204|1465535556|     ST410|1465535556|   U111|
|     A300|      1|1468139889|  null|   0|            3|   S201|1465535556|     ST410|1494342562|   U120|
|     null|      0|1465535556|     A|   1|            1|   S203|1465535556|     ST402|1465535556|   U113|
|     A304|      1|1468139889|     E|   1|            1|   S203|1494342562|     ST405|1462908262|   U109|
|     A303|      0|1468139889|    AU|   1|            2|   S202|1494342562|     ST402|1494342562|   U110|
|     A301|      1|1494342562|    AP|   1|            3|   S200|1494342562|     ST410|1494342562|   U100|
|     A300|      1|1462908262|     E|   1|            0|   S208|1468139889|     ST408|1462908262|   U101|
|     A300|      0|1462908262|     A|   1|            3|   S206|1465535556|     ST405|1494342562|   U106|
|     A304|      0|1462908262|     U|   0|            0|   S202|1468139889|     ST409|1494342562|   U107|
|     A300|      0|1465535556|    AU|   1|            2|   S204|1494342562|     ST411|1468139889|   U103|
|     A300|      1|1465535556|     A|   1|            2|   S202|1465535556|     ST415|1465535556|   U103|
|     A303|      0|1494342562|     U|   0|            2|   S203|1468139889|     ST408|1462908262|   U113|
|     A301|      1|1465535556|     E|   0|            3|   S204|1494342562|     ST415|1494342562|   U113|
+---------+-------+----------+------+----+-------------+-------+----------+----------+----------+-------+
```

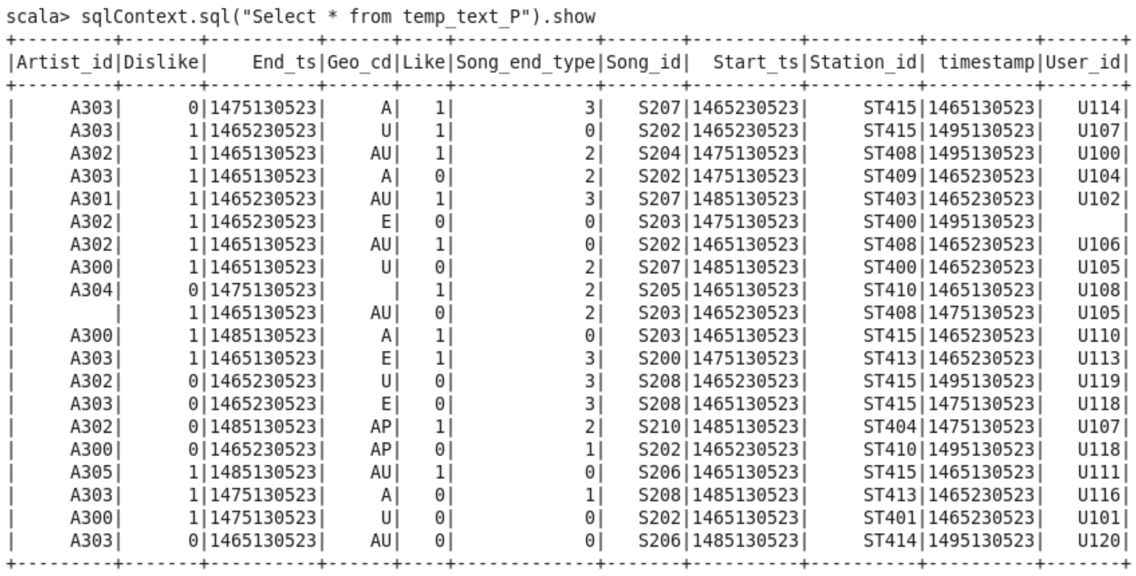
Loading data from textfile

--Creating case class , and loading the textfile



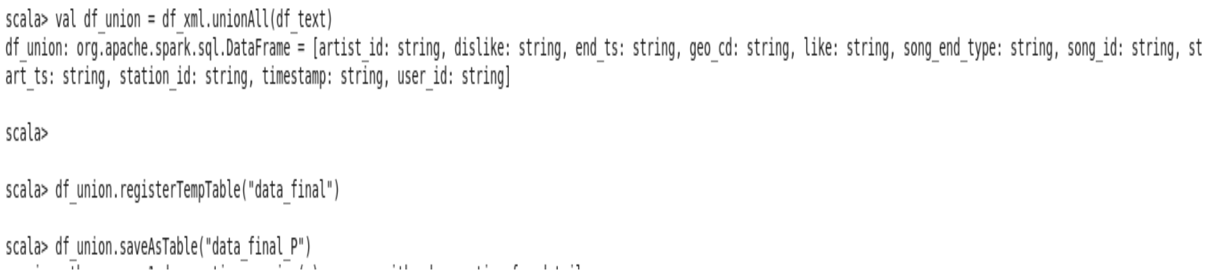
```
scala> case class Test2(Artist_id:String,Dislike:String,End_ts:String,Geo_cd:String,Like:String,Song_end_type:String,Song_id:String,Start_ts:String,Station_id:String,ti
mestamp:String,User_id:String)
defined class Test2

scala>

scala> val myFile = sc.textFile("file:///home/cloudera/Desktop/Acadgild/Mob/file.txt")
myFile: org.apache.spark.rdd.RDD[String] = file:///home/cloudera/Desktop/Acadgild/Mob/file.txt MapPartitionsRDD[303] at textFile at <console>:27

scala> val df_text= myFile.map( x => x.split(",") ).map(x=> Test2(x(2),x(10),x(5),x(6),x(9),x(8),x(1),x(4),x(7),x(3),x(0))).toDF()
df_text: org.apache.spark.sql.DataFrame = [Artist_id: string, Dislike: string, End_ts: string, Geo_cd: string, Like: string, Song_end_type: string, Song_id: string, Sta
rt_ts: string, Station_id: string, timestamp: string, User_id: string]

scala> df_text.registerTempTable("temp_text")

scala> df_text.saveAsTable("temp_text_P")
```

Show the table data



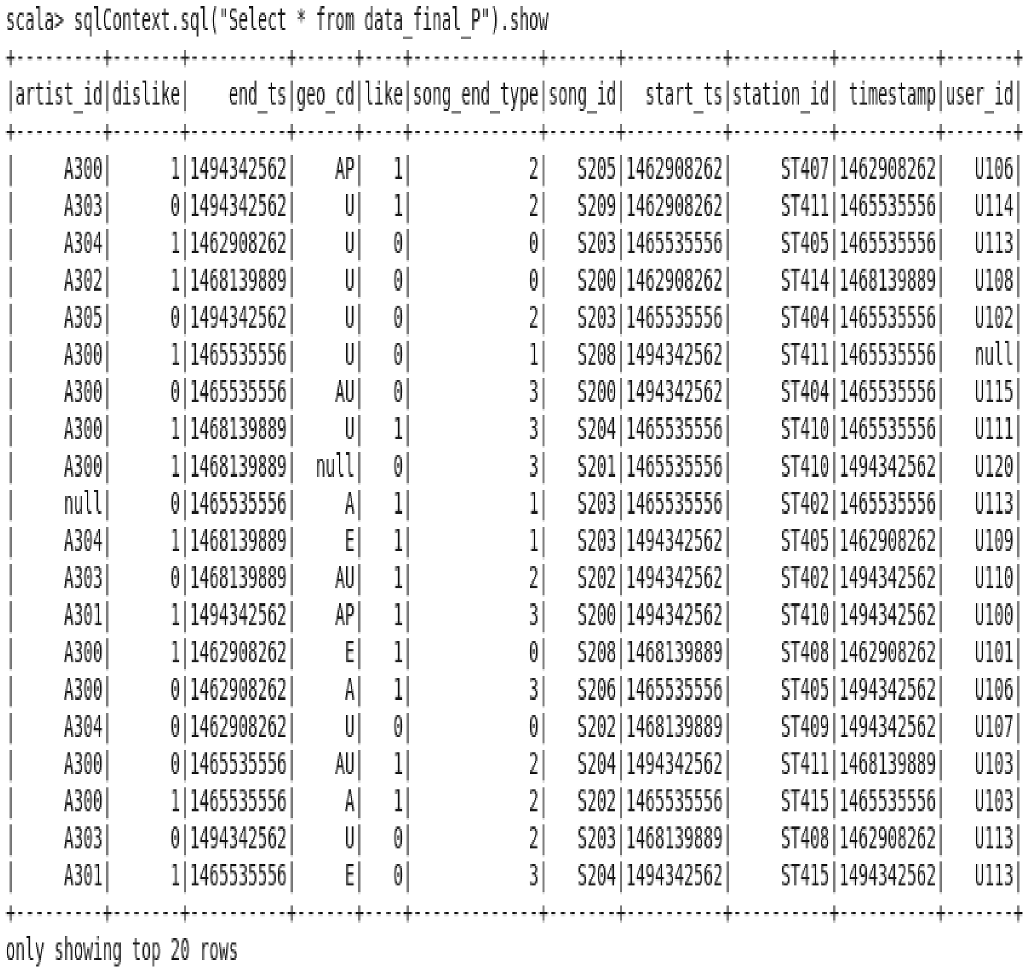
```
scala> sqlContext.sql("Select * from temp_text_P").show
+---------+-------+----------+------+----+-------------+-------+----------+----------+----------+-------+
|Artist_id|Dislike|    End_ts|Geo_cd|Like|Song_end_type|Song_id|  Start_ts|Station_id| timestamp|User_id|
+---------+-------+----------+------+----+-------------+-------+----------+----------+----------+-------+
|     A303|      0|1475130523|     A|   1|            3|   S207|1465230523|     ST415|1465130523|   U114|
|     A303|      1|1465230523|     U|   1|            0|   S202|1465230523|     ST415|1495130523|   U107|
|     A302|      1|1465130523|    AU|   1|            2|   S204|1475130523|     ST408|1495130523|   U100|
|     A303|      1|1465130523|     A|   0|            2|   S202|1475130523|     ST409|1465230523|   U104|
|     A301|      1|1465230523|    AU|   1|            3|   S207|1485130523|     ST403|1465230523|   U102|
|     A302|      1|1465230523|     E|   0|            0|   S203|1475130523|     ST400|1495130523|       |
|     A302|      1|1465130523|    AU|   1|            0|   S202|1465130523|     ST408|1465230523|   U106|
|     A300|      1|1465130523|     U|   0|            2|   S207|1485130523|     ST400|1465230523|   U105|
|     A304|      0|1475130523|      |   1|            2|   S205|1465130523|     ST410|1465130523|   U108|
|         |      1|1465130523|    AU|   0|            2|   S203|1465230523|     ST408|1475130523|   U105|
|     A300|      1|1485130523|     A|   1|            0|   S203|1465130523|     ST415|1465230523|   U110|
|     A303|      1|1465130523|     E|   1|            3|   S200|1475130523|     ST413|1465230523|   U113|
|     A302|      0|1465230523|     U|   0|            3|   S208|1465230523|     ST415|1495130523|   U119|
|     A303|      0|1465230523|     E|   0|            3|   S208|1465130523|     ST415|1475130523|   U118|
|     A302|      0|1485130523|    AP|   1|            2|   S210|1485130523|     ST404|1475130523|   U107|
|     A300|      0|1465230523|    AP|   0|            1|   S202|1465230523|     ST410|1495130523|   U118|
|     A305|      1|1485130523|    AU|   1|            0|   S206|1465130523|     ST415|1465130523|   U111|
|     A303|      1|1475130523|     A|   0|            1|   S208|1485130523|     ST413|1465230523|   U116|
|     A300|      1|1475130523|     U|   0|            0|   S202|1465130523|     ST401|1465230523|   U101|
|     A303|      0|1465130523|    AU|   0|            0|   S206|1485130523|     ST414|1495130523|   U120|
+---------+-------+----------+------+----+-------------+-------+----------+----------+----------+-------+
```

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



```
scala> val df_union = df_xml.unionAll(df_text)
df_union: org.apache.spark.sql.DataFrame = [artist_id: string, dislike: string, end_ts: string, geo_cd: string, like: string, song_end_type: string, song_id: string, st
art_ts: string, station_id: string, timestamp: string, user_id: string]

scala>

scala> df_union.registerTempTable("data_final")

scala> df_union.saveAsTable("data_final_P")
```

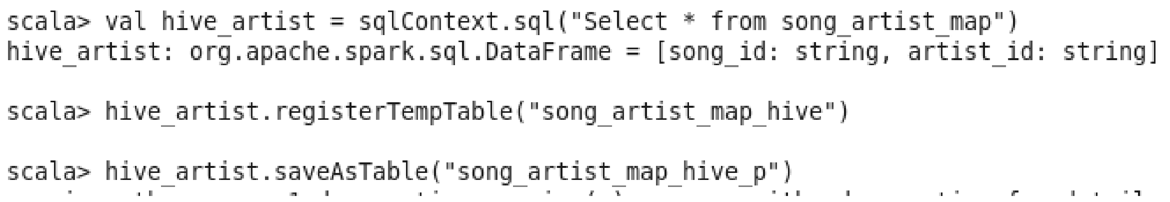
SHOWING THE DATA



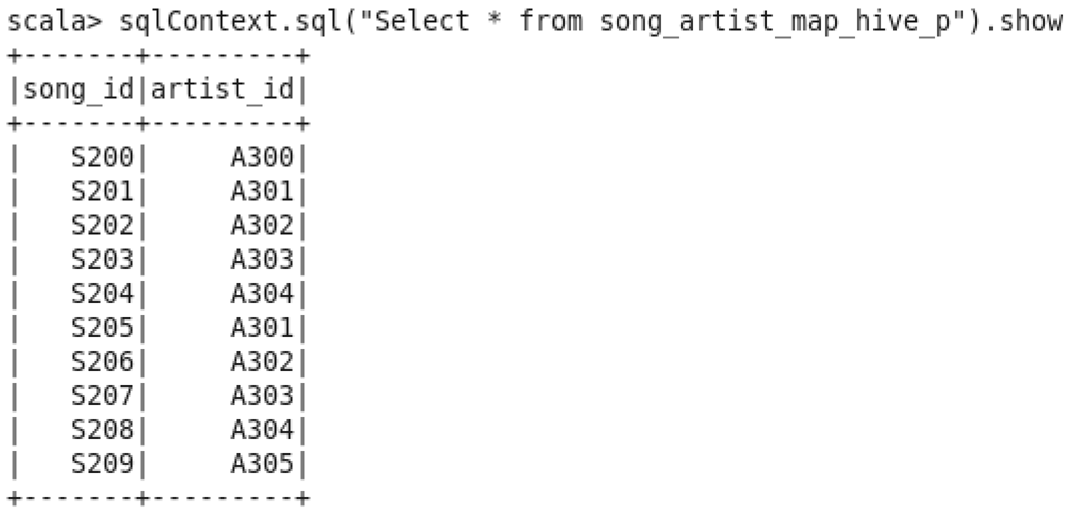
```
scala> sqlContext.sql("Select * from data_final_P").show
+---------+-------+----------+------+----+-------------+-------+----------+----------+----------+-------+
|artist_id|dislike|    end_ts|geo_cd|like|song_end_type|song_id|  start_ts|station_id| timestamp|user_id|
+---------+-------+----------+------+----+-------------+-------+----------+----------+----------+-------+
|     A300|      1|1494342562|    AP|   1|            2|   S205|1462908262|     ST407|1462908262|   U106|
|     A303|      0|1494342562|     U|   1|            2|   S209|1462908262|     ST411|1465535556|   U114|
|     A304|      1|1462908262|     U|   0|            0|   S203|1465535556|     ST405|1465535556|   U113|
|     A302|      1|1468139889|     U|   0|            0|   S200|1462908262|     ST414|1468139889|   U108|
|     A305|      0|1494342562|     U|   0|            2|   S203|1465535556|     ST404|1465535556|   U102|
|     A300|      1|1465535556|     U|   0|            1|   S208|1494342562|     ST411|1465535556|   null|
|     A300|      0|1465535556|    AU|   0|            3|   S200|1494342562|     ST404|1465535556|   U115|
|     A300|      1|1468139889|     U|   1|            3|   S204|1465535556|     ST410|1465535556|   U111|
|     A300|      1|1468139889|  null|   0|            3|   S201|1465535556|     ST410|1494342562|   U120|
|     null|      0|1465535556|     A|   1|            1|   S203|1465535556|     ST402|1465535556|   U113|
|     A304|      1|1468139889|     E|   1|            1|   S203|1494342562|     ST405|1462908262|   U109|
|     A303|      0|1468139889|    AU|   1|            2|   S202|1494342562|     ST402|1494342562|   U110|
|     A301|      1|1494342562|    AP|   1|            3|   S200|1494342562|     ST410|1494342562|   U100|
|     A300|      1|1462908262|     E|   1|            0|   S208|1468139889|     ST408|1462908262|   U101|
|     A300|      0|1462908262|     A|   1|            3|   S206|1465535556|     ST405|1494342562|   U106|
|     A304|      0|1462908262|     U|   0|            0|   S202|1468139889|     ST409|1494342562|   U107|
|     A300|      0|1465535556|    AU|   1|            2|   S204|1494342562|     ST411|1468139889|   U103|
|     A300|      1|1465535556|     A|   1|            2|   S202|1465535556|     ST415|1465535556|   U103|
|     A303|      0|1494342562|     U|   0|            2|   S203|1468139889|     ST408|1462908262|   U113|
|     A301|      1|1465535556|     E|   0|            3|   S204|1494342562|     ST415|1494342562|   U113|
+---------+-------+----------+------+----+-------------+-------+----------+----------+----------+-------+
only showing top 20 rows
```

-

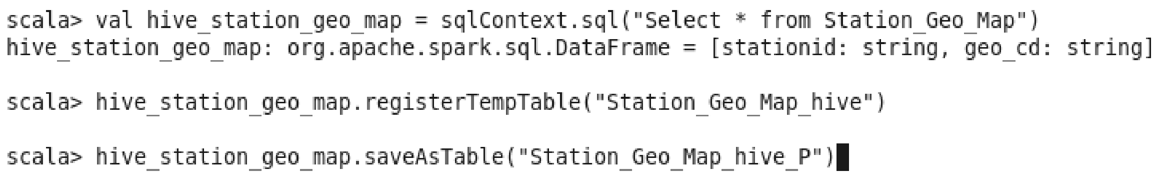
LOADING DATA FROM HIVE TABLE IN SPARK SQL , HIVE TABLE ARE LOOKUP TABLES WHICH WE INTEGRATED FROM HBASE



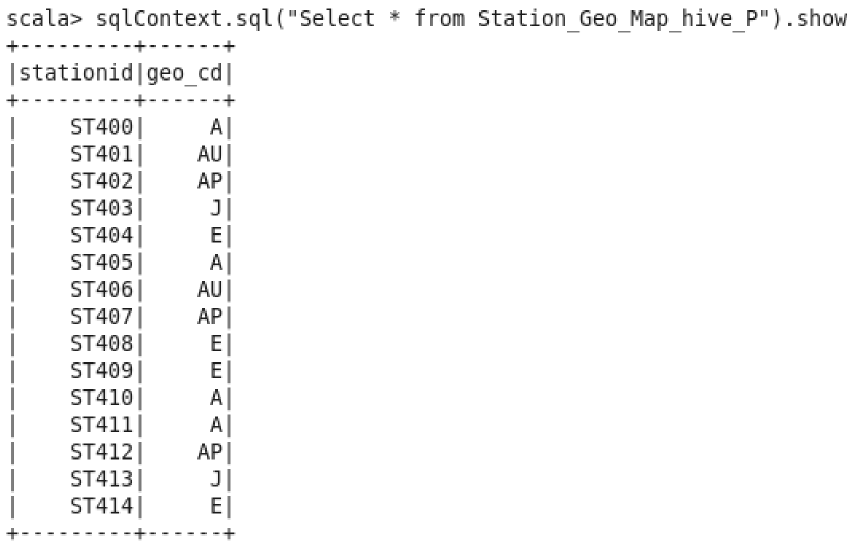
```
scala> val hive_artist = sqlContext.sql("Select * from song_artist_map")
hive_artist: org.apache.spark.sql.DataFrame = [song_id: string, artist_id: string]

scala> hive_artist.registerTempTable("song_artist_map_hive")

scala> hive_artist.saveAsTable("song_artist_map_hive_p")
```



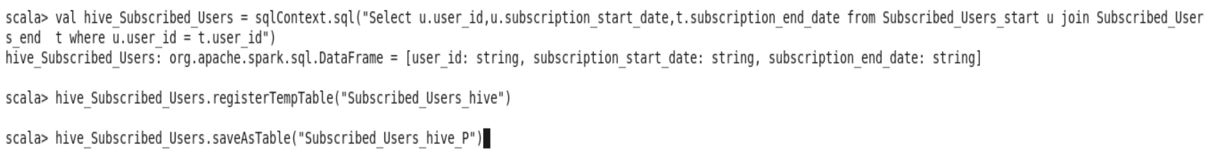
```
scala> sqlContext.sql("Select * from song_artist_map_hive_p").show
+-------+---------+
|song_id|artist_id|
+-------+---------+
|   S200|     A300|
|   S201|     A301|
|   S202|     A302|
|   S203|     A303|
|   S204|     A304|
|   S205|     A301|
|   S206|     A302|
|   S207|     A303|
|   S208|     A304|
|   S209|     A305|
+-------+---------+
```



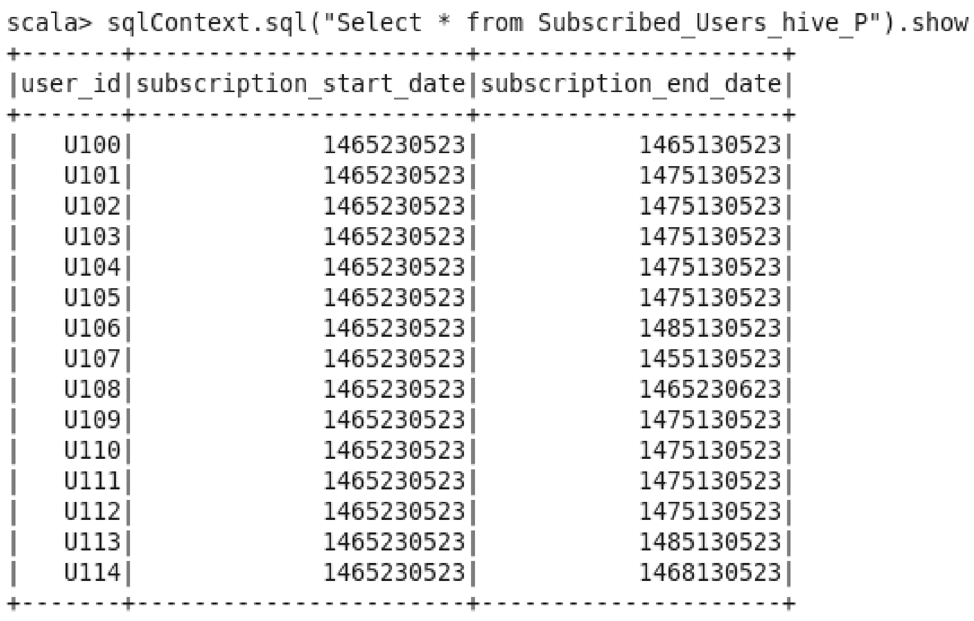
```
scala> val hive_station_geo_map = sqlContext.sql("Select * from Station_Geo_Map")
hive_station_geo_map: org.apache.spark.sql.DataFrame = [stationid: string, geo_cd: string]

scala> hive_station_geo_map.registerTempTable("Station_Geo_Map_hive")

scala> hive_station_geo_map.saveAsTable("Station_Geo_Map_hive_P")
```



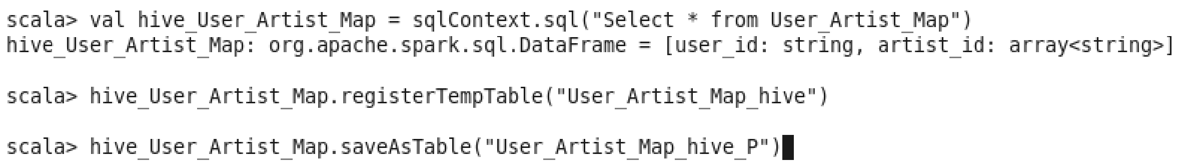
```
scala> sqlContext.sql("Select * from Station_Geo_Map_hive_P").show
+---------+------+
|stationid|geo_cd|
+---------+------+
|    ST400|     A|
|    ST401|    AU|
|    ST402|    AP|
|    ST403|     J|
|    ST404|     E|
|    ST405|     A|
|    ST406|    AU|
|    ST407|    AP|
|    ST408|     E|
|    ST409|     E|
|    ST410|     A|
|    ST411|     A|
|    ST412|    AP|
|    ST413|     J|
|    ST414|     E|
+---------+------+
```



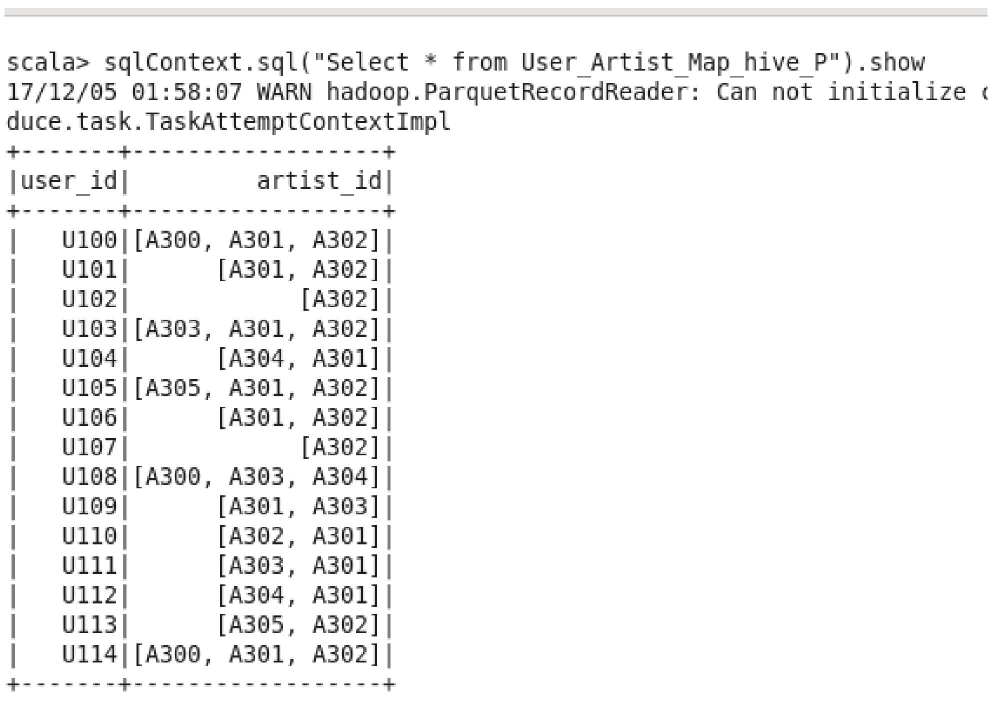
```
scala> val hive_Subscribed_Users = sqlContext.sql("Select u.user_id,u.subscription_start_date,t.subscription_end_date from Subscribed_Users_start u join Subscribed_User
s_end  t where u.user_id = t.user_id")
hive_Subscribed_Users: org.apache.spark.sql.DataFrame = [user_id: string, subscription_start_date: string, subscription_end_date: string]

scala> hive_Subscribed_Users.registerTempTable("Subscribed_Users_hive")

scala> hive_Subscribed_Users.saveAsTable("Subscribed_Users_hive_P")
```



```
scala> sqlContext.sql("Select * from Subscribed_Users_hive_P").show
+-------+-----------------------+---------------------+
|user_id|subscription_start_date|subscription_end_date|
+-------+-----------------------+---------------------+
|   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|
+-------+-----------------------+---------------------+
```



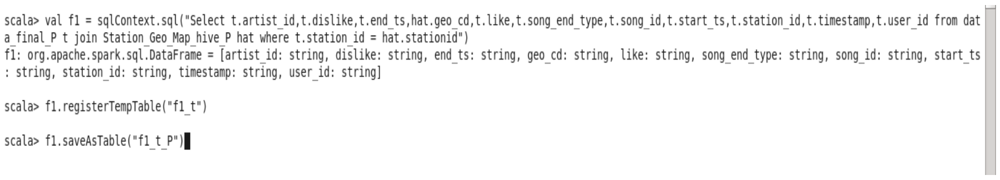
```
scala> val hive_User_Artist_Map = sqlContext.sql("Select * from User_Artist_Map")
hive_User_Artist_Map: org.apache.spark.sql.DataFrame = [user_id: string, artist_id: array<string>]

scala> hive_User_Artist_Map.registerTempTable("User_Artist_Map_hive")

scala> hive_User_Artist_Map.saveAsTable("User_Artist_Map_hive_P")
```



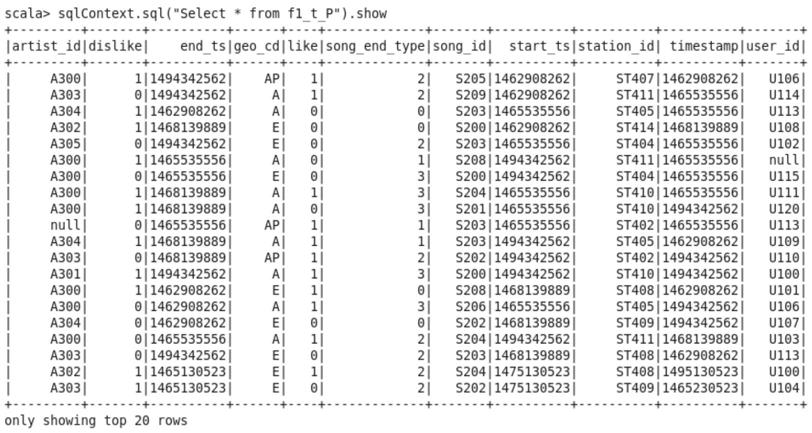
```
scala> sqlContext.sql("Select * from User_Artist_Map_hive_P").show
17/12/05 01:58:07 WARN hadoop.ParquetRecordReader: Can not initialize c
duce.task.TaskAttemptContextImpl
+-------+------------------+
|user_id|         artist_id|
+-------+------------------+
|   U100|[A300, A301, A302]|
|   U101|      [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]|
+-------+------------------+
```

--DATA ENRICHMENT--

Removing null or absent values from Geo_cd using lookup field Station_id from Station_Geo_Map

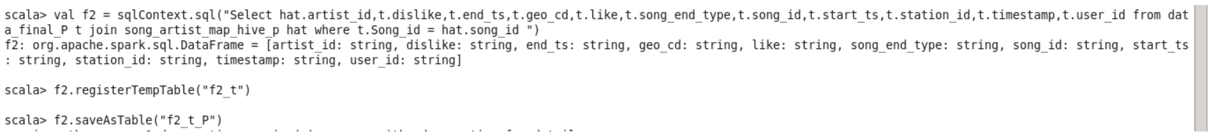


```
scala> val f1 = sqlContext.sql("Select t.artist_id,t.dislike,t.end_ts,hat.geo_cd,t.like,t.song_end_type,t.song_id,t.start_ts,t.station_id,t.timestamp,t.user_id from dat
a_final_P t join Station_Geo_Map_hive_P hat where t.station_id = hat.stationid")
f1: org.apache.spark.sql.DataFrame = [artist_id: string, dislike: string, end_ts: string, geo_cd: string, like: string, song_end_type: string, song_id: string, start_ts
: string, station_id: string, timestamp: string, user_id: string]

scala> f1.registerTempTable("f1_t")

scala> f1.saveAsTable("f1_t_P")
```

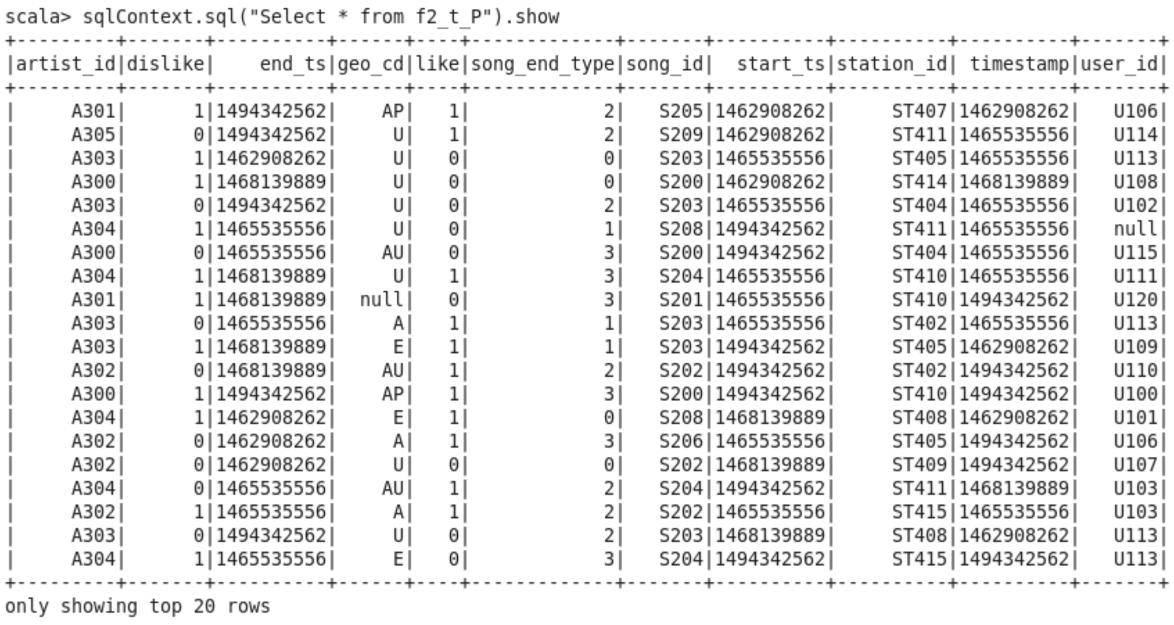


```
scala> sqlContext.sql("Select * from f1_t_P").show
+---------+-------+----------+------+----+-------------+-------+----------+----------+----------+-------+
|artist_id|dislike|    end_ts|geo_cd|like|song_end_type|song_id|  start_ts|station_id| timestamp|user_id|
+---------+-------+----------+------+----+-------------+-------+----------+----------+----------+-------+
|     A300|      1|1494342562|    AP|   1|            2|   S205|1462908262|     ST407|1462908262|   U106|
|     A303|      0|1494342562|     A|   1|            2|   S209|1462908262|     ST411|1465535556|   U114|
|     A304|      1|1462908262|     A|   0|            0|   S203|1465535556|     ST405|1465535556|   U113|
|     A302|      1|1468139889|     E|   0|            0|   S200|1462908262|     ST414|1468139889|   U108|
|     A305|      0|1494342562|     E|   0|            2|   S203|1465535556|     ST404|1465535556|   U102|
|     A300|      1|1465535556|     A|   0|            1|   S208|1494342562|     ST411|1465535556|   null|
|     A300|      0|1465535556|     E|   0|            3|   S200|1494342562|     ST404|1465535556|   U115|
|     A300|      1|1468139889|     A|   1|            3|   S204|1465535556|     ST410|1465535556|   U111|
|     A300|      1|1468139889|     A|   0|            3|   S201|1465535556|     ST410|1494342562|   U120|
|     null|      0|1465535556|    AP|   1|            1|   S203|1465535556|     ST402|1465535556|   U113|
|     A304|      1|1468139889|     A|   1|            1|   S203|1494342562|     ST405|1462908262|   U109|
|     A303|      0|1468139889|    AP|   1|            2|   S202|1494342562|     ST402|1494342562|   U110|
|     A301|      1|1494342562|     A|   1|            3|   S200|1494342562|     ST410|1494342562|   U100|
|     A300|      1|1462908262|     E|   1|            0|   S208|1468139889|     ST408|1462908262|   U101|
|     A300|      0|1462908262|     A|   1|            3|   S206|1465535556|     ST405|1494342562|   U106|
|     A304|      0|1462908262|     E|   0|            0|   S202|1468139889|     ST409|1494342562|   U107|
|     A300|      0|1465535556|     A|   1|            2|   S204|1494342562|     ST411|1468139889|   U103|
|     A303|      0|1494342562|     E|   0|            2|   S203|1468139889|     ST408|1462908262|   U113|
|     A302|      1|1465130523|     E|   1|            2|   S204|1475130523|     ST408|1495130523|   U100|
|     A303|      1|1465130523|     E|   0|            2|   S202|1475130523|     ST409|1465230523|   U104|
+---------+-------+----------+------+----+-------------+-------+----------+----------+----------+-------+
only showing top 20 rows
```

Removing null or absent values from Artist using lookup field Song_id from song_artist_map

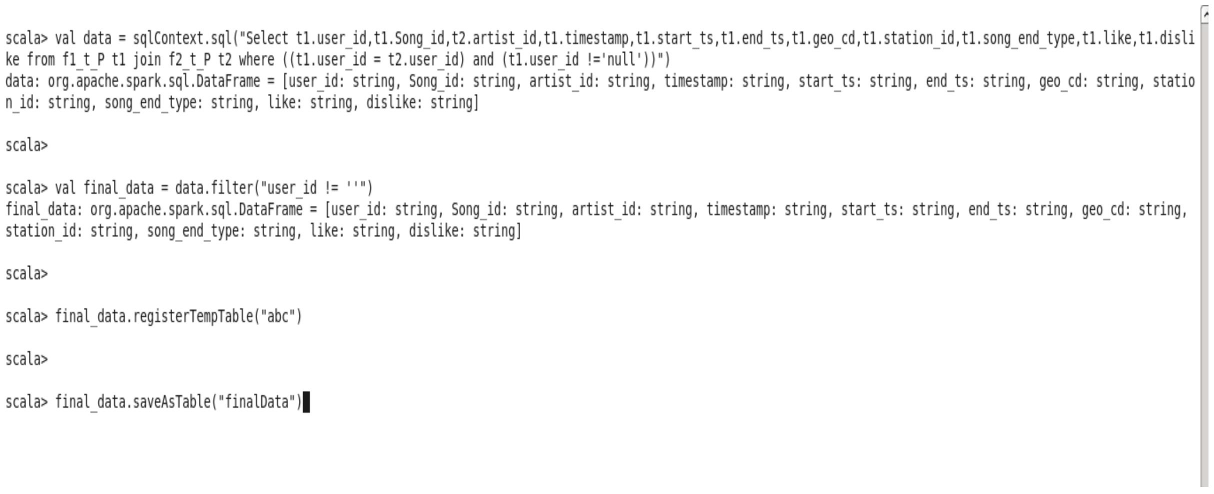


```
scala> val f2 = sqlContext.sql("Select hat.artist_id,t.dislike,t.end_ts,t.geo_cd,t.like,t.song_end_type,t.song_id,t.start_ts,t.station_id,t.timestamp,t.user_id from dat
a_final_P t join song_artist_map_hive_p hat where t.Song_id = hat.song_id ")
f2: org.apache.spark.sql.DataFrame = [artist_id: string, dislike: string, end_ts: string, geo_cd: string, like: string, song_end_type: string, song_id: string, start_ts
: string, station_id: string, timestamp: string, user_id: string]

scala> f2.registerTempTable("f2_t")

scala> f2.saveAsTable("f2_t_P")
```

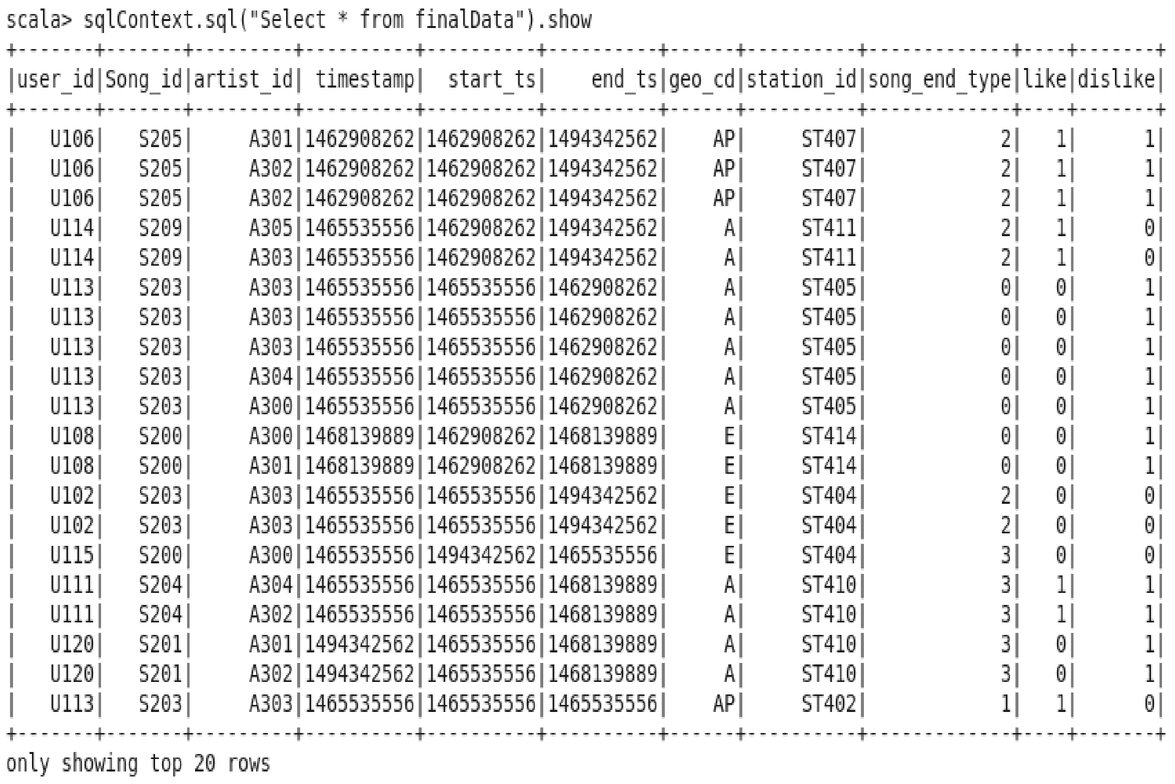


```
scala> sqlContext.sql("Select * from f2_t_P").show
+---------+-------+----------+------+----+-------------+-------+----------+----------+----------+-------+
|artist_id|dislike|    end_ts|geo_cd|like|song_end_type|song_id|  start_ts|station_id| timestamp|user_id|
+---------+-------+----------+------+----+-------------+-------+----------+----------+----------+-------+
|     A301|      1|1494342562|    AP|   1|            2|   S205|1462908262|     ST407|1462908262|   U106|
|     A305|      0|1494342562|     U|   1|            2|   S209|1462908262|     ST411|1465535556|   U114|
|     A303|      1|1462908262|     U|   0|            0|   S203|1465535556|     ST405|1465535556|   U113|
|     A300|      1|1468139889|     U|   0|            0|   S200|1462908262|     ST414|1468139889|   U108|
|     A303|      0|1494342562|     U|   0|            2|   S203|1465535556|     ST404|1465535556|   U102|
|     A304|      1|1465535556|     U|   0|            1|   S208|1494342562|     ST411|1465535556|   null|
|     A300|      0|1465535556|    AU|   0|            3|   S200|1494342562|     ST404|1465535556|   U115|
|     A304|      1|1468139889|     U|   1|            3|   S204|1465535556|     ST410|1465535556|   U111|
|     A301|      1|1468139889|  null|   0|            3|   S201|1465535556|     ST410|1494342562|   U120|
|     A303|      0|1465535556|     A|   1|            1|   S203|1465535556|     ST402|1465535556|   U113|
|     A303|      1|1468139889|     E|   1|            1|   S203|1494342562|     ST405|1462908262|   U109|
|     A302|      0|1468139889|    AU|   1|            2|   S202|1494342562|     ST402|1494342562|   U110|
|     A300|      1|1494342562|    AP|   1|            3|   S200|1494342562|     ST410|1494342562|   U100|
|     A304|      1|1462908262|     E|   1|            0|   S208|1468139889|     ST408|1462908262|   U101|
|     A302|      0|1462908262|     A|   1|            3|   S206|1465535556|     ST405|1494342562|   U106|
|     A302|      0|1462908262|     U|   0|            0|   S202|1468139889|     ST409|1494342562|   U107|
|     A304|      0|1465535556|    AU|   1|            2|   S204|1494342562|     ST411|1468139889|   U103|
|     A302|      1|1465535556|     A|   1|            2|   S202|1465535556|     ST415|1465535556|   U103|
|     A303|      0|1494342562|     U|   0|            2|   S203|1468139889|     ST408|1462908262|   U113|
|     A304|      1|1465535556|     E|   0|            3|   S204|1494342562|     ST415|1494342562|   U113|
+---------+-------+----------+------+----+-------------+-------+----------+----------+----------+-------+
only showing top 20 rows
```

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



```
scala> val data = sqlContext.sql("Select t1.user_id,t1.Song_id,t2.artist_id,t1.timestamp,t1.start_ts,t1.end_ts,t1.geo_cd,t1.station_id,t1.song_end_type,t1.like,t1.disli
ke from f1_t_P t1 join f2_t_P t2 where ((t1.user_id = t2.user_id) and (t1.user_id !='null'))")
data: org.apache.spark.sql.DataFrame = [user_id: string, Song_id: string, artist_id: string, timestamp: string, start_ts: string, end_ts: string, geo_cd: string, statio
n_id: string, song_end_type: string, like: string, dislike: string]

scala>

scala> val final_data = data.filter("user_id != ''")
final_data: org.apache.spark.sql.DataFrame = [user_id: string, Song_id: string, artist_id: string, timestamp: string, start_ts: string, end_ts: string, geo_cd: string,
station_id: string, song_end_type: string, like: string, dislike: string]

scala>

scala> final_data.registerTempTable("abc")

scala>

scala> final_data.saveAsTable("finalData")
```

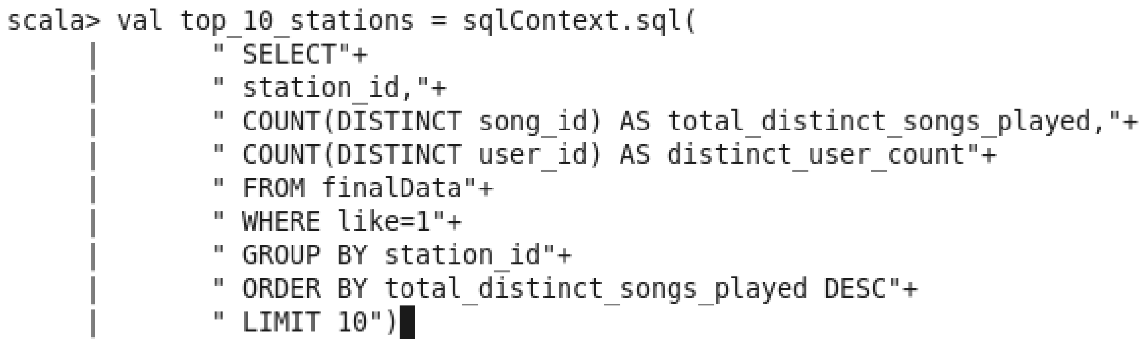


```
scala> sqlContext.sql("Select * from finalData").show
+-------+-------+---------+----------+----------+----------+------+----------+-------------+----+-------+
|user_id|Song_id|artist_id| timestamp|  start_ts|    end_ts|geo_cd|station_id|song_end_type|like|dislike|
+-------+-------+---------+----------+----------+----------+------+----------+-------------+----+-------+
|   U106|   S205|     A301|1462908262|1462908262|1494342562|    AP|     ST407|            2|   1|      1|
|   U106|   S205|     A302|1462908262|1462908262|1494342562|    AP|     ST407|            2|   1|      1|
|   U106|   S205|     A302|1462908262|1462908262|1494342562|    AP|     ST407|            2|   1|      1|
|   U114|   S209|     A305|1465535556|1462908262|1494342562|     A|     ST411|            2|   1|      0|
|   U114|   S209|     A303|1465535556|1462908262|1494342562|     A|     ST411|            2|   1|      0|
|   U113|   S203|     A303|1465535556|1465535556|1462908262|     A|     ST405|            0|   0|      1|
|   U113|   S203|     A303|1465535556|1465535556|1462908262|     A|     ST405|            0|   0|      1|
|   U113|   S203|     A303|1465535556|1465535556|1462908262|     A|     ST405|            0|   0|      1|
|   U113|   S203|     A304|1465535556|1465535556|1462908262|     A|     ST405|            0|   0|      1|
|   U113|   S203|     A300|1465535556|1465535556|1462908262|     A|     ST405|            0|   0|      1|
|   U108|   S200|     A300|1468139889|1462908262|1468139889|     E|     ST414|            0|   0|      1|
|   U108|   S200|     A301|1468139889|1462908262|1468139889|     E|     ST414|            0|   0|      1|
|   U102|   S203|     A303|1465535556|1465535556|1494342562|     E|     ST404|            2|   0|      0|
|   U102|   S203|     A303|1465535556|1465535556|1494342562|     E|     ST404|            2|   0|      0|
|   U115|   S200|     A300|1465535556|1494342562|1465535556|     E|     ST404|            3|   0|      0|
|   U111|   S204|     A304|1465535556|1465535556|1468139889|     A|     ST410|            3|   1|      1|
|   U111|   S204|     A302|1465535556|1465535556|1468139889|     A|     ST410|            3|   1|      1|
|   U120|   S201|     A301|1494342562|1465535556|1468139889|     A|     ST410|            3|   0|      1|
|   U120|   S201|     A302|1494342562|1465535556|1468139889|     A|     ST410|            3|   0|      1|
|   U113|   S203|     A303|1465535556|1465535556|1465535556|    AP|     ST402|            1|   1|      0|
+-------+-------+---------+----------+----------+----------+------+----------+-------------+----+-------+
only showing top 20 rows
```

-----------------------------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.

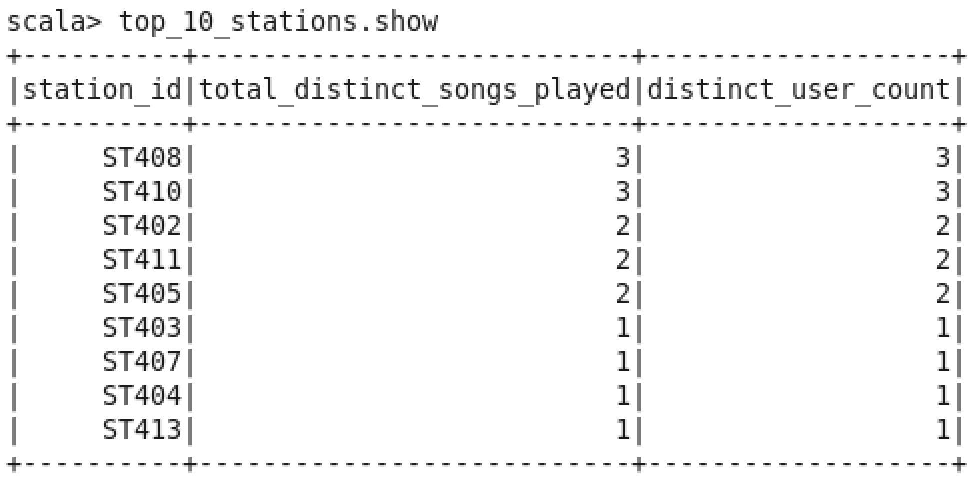
CODE:



```
scala> val top_10_stations = sqlContext.sql(
     |        " SELECT"+
     |        " station_id,"+
     |        " COUNT(DISTINCT song_id) AS total_distinct_songs_played,"+
     |        " COUNT(DISTINCT user_id) AS distinct_user_count"+
     |        " FROM finalData"+
     |        " WHERE like=1"+
     |        " GROUP BY station_id"+
     |        " ORDER BY total_distinct_songs_played DESC"+
     |        " LIMIT 10")
```

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

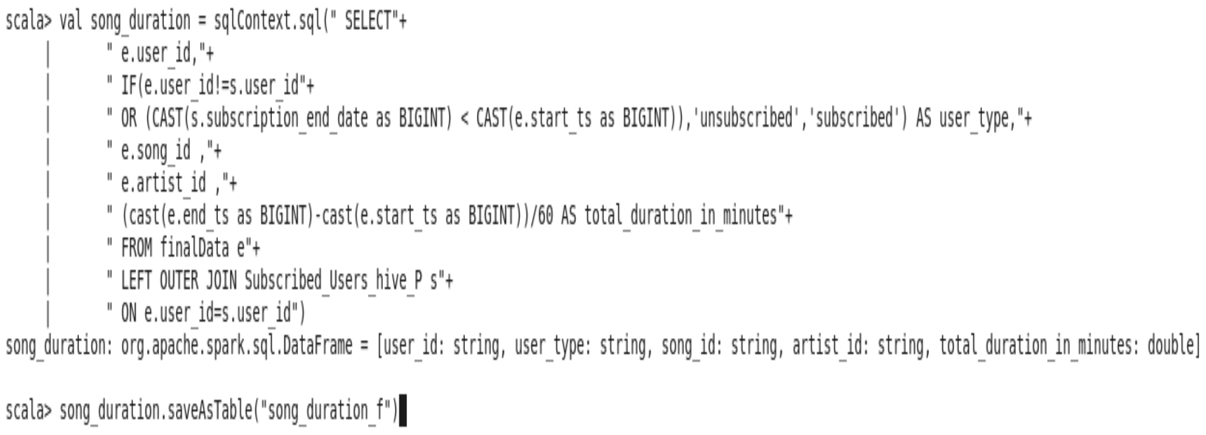
OUTPUT



```
scala> top_10_stations.show
+----------+---------------------------+-------------------+
|station_id|total_distinct_songs_played|distinct_user_count|
+----------+---------------------------+-------------------+
|     ST408|                          3|                  3|
|     ST410|                          3|                  3|
|     ST402|                          2|                  2|
|     ST411|                          2|                  2|
|     ST405|                          2|                  2|
|     ST403|                          1|                  1|
|     ST407|                          1|                  1|
|     ST404|                          1|                  1|
|     ST413|                          1|                  1|
+----------+---------------------------+-------------------+
```

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.

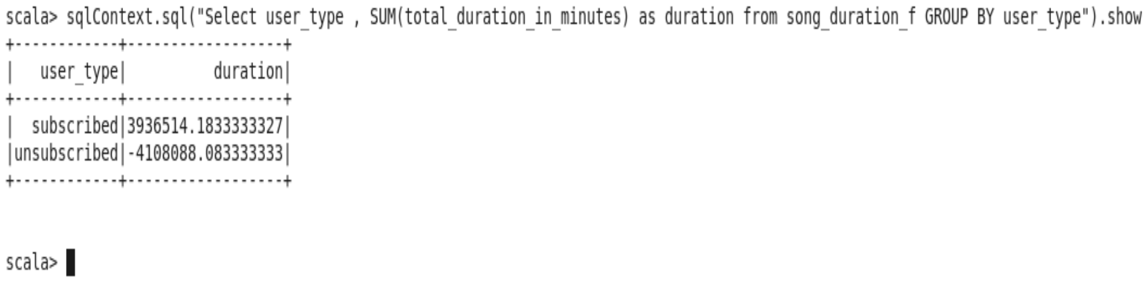
CODE:



```
scala> val song_duration = sqlContext.sql(" SELECT"+
     |       " e.user_id,"+
     |       " IF(e.user_id!=s.user_id"+
     |       " OR (CAST(s.subscription_end_date as BIGINT) < CAST(e.start_ts as BIGINT)),'unsubscribed','subscribed') AS user_type,"+
     |       " e.song_id ,"+
     |       " e.artist_id ,"+
     |       " (cast(e.end_ts as BIGINT)-cast(e.start_ts as BIGINT))/60 AS total_duration_in_minutes"+
     |       " FROM finalData e"+
     |       " LEFT OUTER JOIN Subscribed_Users_hive_P s"+
     |       " ON e.user_id=s.user_id")
song_duration: org.apache.spark.sql.DataFrame = [user_id: string, user_type: string, song_id: string, artist_id: string, total_duration_in_minutes: double]

scala> song_duration.saveAsTable("song_duration_f")
```

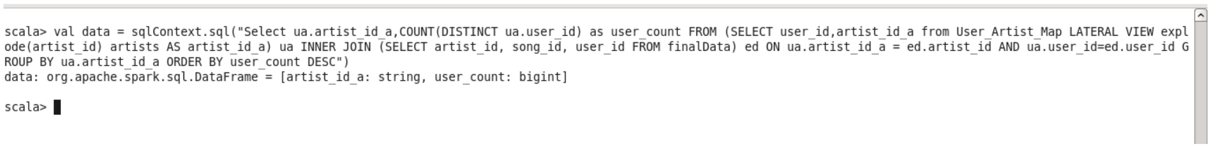
We are using lookup tables for the analysis , we have cast the subs end data and start date to big int

And joining with the lookup table for necessary analysis .



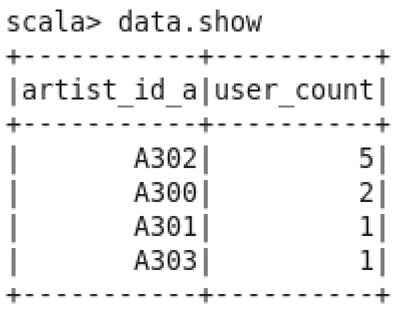
```
scala> sqlContext.sql("Select user_type , SUM(total_duration_in_minutes) as duration from song_duration_f GROUP BY user_type").show
+------------+------------------+
|   user_type|          duration|
+------------+------------------+
|  subscribed|3936514.1833333327|
|unsubscribed|-4108088.083333333|
+------------+------------------+


scala> 
```

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



```
scala> val data = sqlContext.sql("Select ua.artist_id_a,COUNT(DISTINCT ua.user_id) as user_count FROM (SELECT user_id,artist_id_a from User_Artist_Map LATERAL VIEW expl
ode(artist_id) artists AS artist_id_a) ua INNER JOIN (SELECT artist_id, song_id, user_id FROM finalData) ed ON ua.artist_id_a = ed.artist_id AND ua.user_id=ed.user_id G
ROUP BY ua.artist_id_a ORDER BY user_count DESC")
data: org.apache.spark.sql.DataFrame = [artist_id_a: string, user_count: bigint]

scala> 
```

OUTPUT

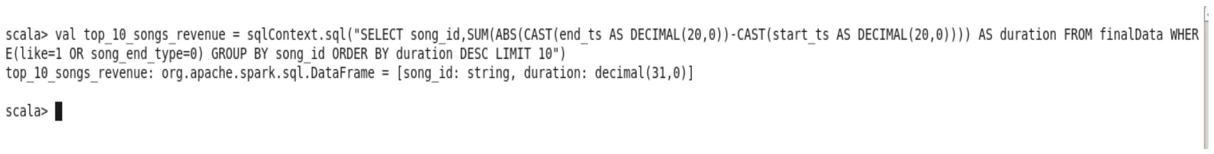


```
scala> data.show
+-----------+----------+
|artist_id_a|user_count|
+-----------+----------+
|       A302|         5|
|       A300|         2|
|       A301|         1|
|       A303|         1|
+-----------+----------+
```

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

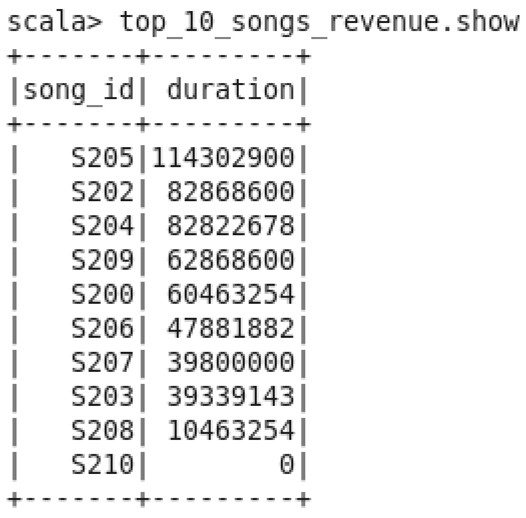
4) Determine top 10 songs who have generated the maximum revenue. Royalty applies to a song only if it was liked or was completed successfully or both.

CODE:



```
scala> val top_10_songs_revenue = sqlContext.sql("SELECT song_id,SUM(ABS(CAST(end_ts AS DECIMAL(20,0))-CAST(start_ts AS DECIMAL(20,0)))) AS duration FROM finalData WHER
E(like=1 OR song_end_type=0) GROUP BY song_id ORDER BY duration DESC LIMIT 10")
top_10_songs_revenue: org.apache.spark.sql.DataFrame = [song_id: string, duration: decimal(31,0)]

scala> 
```

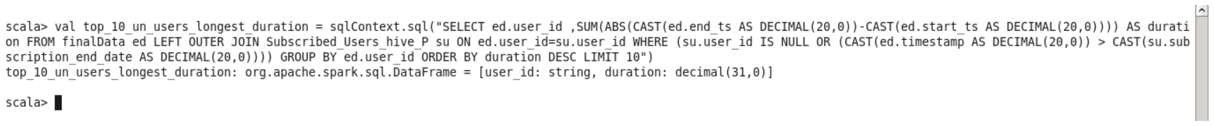
OUTPUT:



```
scala> top_10_songs_revenue.show
+-------+---------+
|song_id| duration|
+-------+---------+
|   S205|114302900|
|   S202| 82868600|
|   S204| 82822678|
|   S209| 62868600|
|   S200| 60463254|
|   S206| 47881882|
|   S207| 39800000|
|   S203| 39339143|
|   S208| 10463254|
|   S210|        0|
+-------+---------+
```

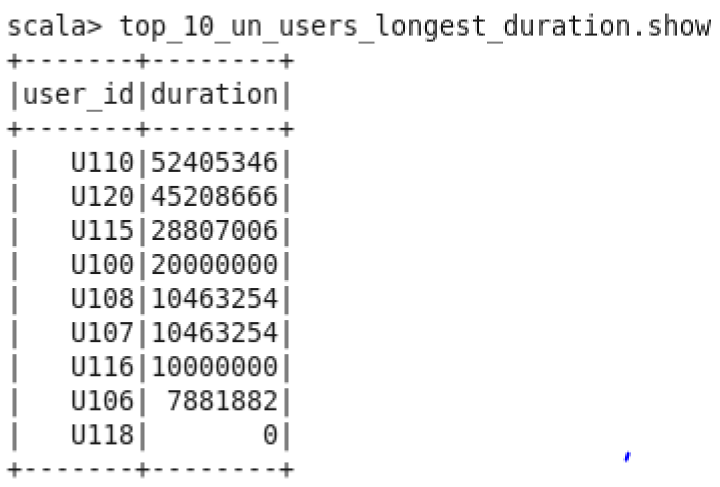
5) Determine top 10 unsubscribed users who listened to the songs for the longest duration.

CODE:



```
scala> val top_10_un_users_longest_duration = sqlContext.sql("SELECT ed.user_id ,SUM(ABS(CAST(ed.end_ts AS DECIMAL(20,0))-CAST(ed.start_ts AS DECIMAL(20,0)))) AS durati
on FROM finalData ed LEFT OUTER JOIN Subscribed_Users_hive_P su ON ed.user_id=su.user_id WHERE (su.user_id IS NULL OR (CAST(ed.timestamp AS DECIMAL(20,0)) > CAST(su.sub
scription_end_date AS DECIMAL(20,0)))) GROUP BY ed.user_id ORDER BY duration DESC LIMIT 10")
top_10_un_users_longest_duration: org.apache.spark.sql.DataFrame = [user_id: string, duration: decimal(31,0)]

scala> 
```

OUTPUT:



```
scala> top_10_un_users_longest_duration.show
+-------+--------+
|user_id|duration|
+-------+--------+
|   U110|52405346|
|   U120|45208666|
|   U115|28807006|
|   U100|20000000|
|   U108|10463254|
|   U107|10463254|
|   U116|10000000|
|   U106| 7881882|
|   U118|       0|
+-------+--------+
```