

1. Design

To solve the Music Data Analysis Requirement the following modules are used

- i. **LoadHBaseTables** – This is used for loading the lookup data from csv files to the relevant tables in Hbase. Hbase tables used are as below:
 - **StationIdGeoCd** – This HBase table is used for storing mapping between stationId and GeoCd
 - **SongArtist** - This HBase table is used for storing mapping between Song and Artist
 - **UserArtist** – This HBase table is used for storing mapping between userId and List of artists he follows
 - **UserSubscription**- This HBase table is used for storing mapping between userId and subscription start timestamp and end timestamp
- ii. **MusicDataProcessorApp** - This is the main application for processing music data. It in turn calls multiple submodules as below
- iii. **WebMusicDataProcessor** – This is the class for processing music data stored in /data/web/file-1.xml folder and use processData method to return dataframes
- iv. **MobileMusicDataProcessor** - This is the class for processing music data stored in /data/mob/file.txt folder and use processData method to return dataframes
- v. **MusicDataEnricher** - This is the used for enriching and validating the datasets. New columns are added to dataframe for the processed data. The new columns added to temporary table MusicDataDetailed are as below:
 - a. **modified_Geo_cd** – This field is used to populate Null/blank values in Geo_cd column. If Geo_cd column is not blank, the value will copied as it is to modified_Geo_cd. If Geo_cd column is blank then consults the look table StationIdGeoCd based on stationId, If found populate it, else put value Invalid
 - b. **modified_Artist_id** – This field is used to populate Null/blank values in Artist_id column. If Artist_id column is not blank, the value will copied as it is to modified_Artist_id. If Artist_id column is blank then consults the look table SongArtist table based on songId, If found populate it, else put value Invalid
 - c. **follower** – This field is used when the user follows the artist. For each record, for userId consult UserArtist table to find the list of artists user is following. If it is blank, then follower will be 0. If it is not blank and the corresponding artistId for the record is present in artist List from the map, then set follower to 1
 - d. **subscribed** – This filed is used to know if the user was a subscriber when he played the music. This is done by getting the userId for the record and consulting **UserSubscription** table to get the tuple (start_ts,end_ts). If it does not exist then subscribed field will be 0. If it exists, If the start_ts of record is in between (start_ts,end_ts) of UserSubscriptpion then subscribed field will be 1 else it will be 0

e. isValid- This field is used to know if the record is valid or not. IsValid if valid will have value 1 else 0. The following validations are done:

If User_id is blank then (0)

If Song_id is blank then 0

If modified_Geo_cd is Invalid then 0

If modified_Arist_id is Invalid then 0

If timestamp is 0 then 0

If Start_ts is 0 then 0

If Start_ts > End_ts then 0

- vi. **MusicDataPopulateMapFromLookupTables** - This is used for populating maps from HBase tables used for lookup
- vii. **MusicDataAnalyzer** - This is used for analyzing the data in dataframes and store the results in HDFS file:

Top10Stations_<Timestamp> - Top 10 stations where maximum number of songs played which is liked by unique user

MusicDurationByUserType_<Timestamp> - Music Duration by user category of user:
Subscribed, Unsubscribed

Top10SongsHavingMaximumRevenue_<Timestamp> - Top 10 songs having maximum revenue

Top10ConnectedArtists_<Timestamp> - Top 10 connected artists

Top10UnsubscribedUsers_<Timestamp> - Top 10 unsubscribed users

2. Implementation

MODULE1: LoadHBaseTables: Loading HBase Tables with Lookup Data

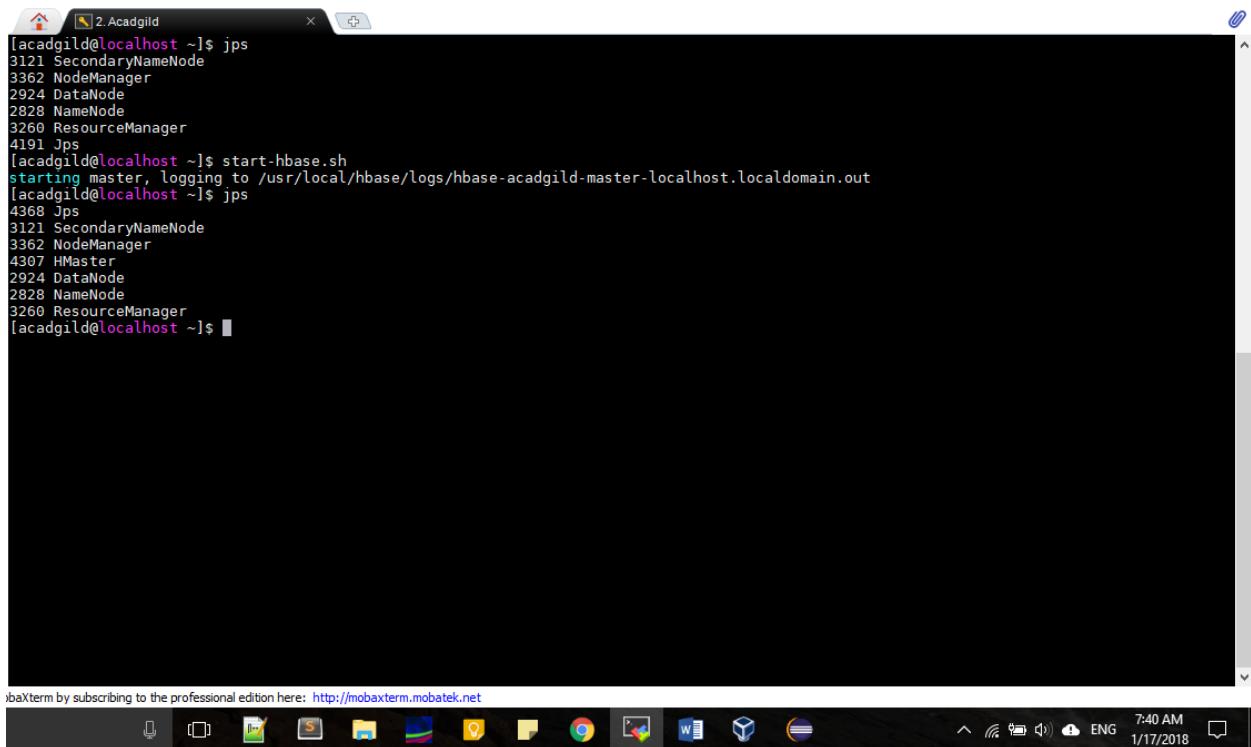
Step1: Start all the services

Start the services using start-all.sh

Start HBase using:

start-hbase.sh

Confirm that services are up using command: jps



```
[acadgild@localhost ~]$ jps
3121 SecondaryNameNode
3362 NodeManager
2924 DataNode
2828 NameNode
3260 ResourceManager
4191 Jps
[acadgild@localhost ~]$ start-hbase.sh
starting master, logging to /usr/local/hbase/logs/hbase-acadgild-master-localhost.localdomain.out
[acadgild@localhost ~]$ jps
4368 Jps
3121 SecondaryNameNode
3362 NodeManager
4307 HMaster
2924 DataNode
2828 NameNode
3260 ResourceManager
[acadgild@localhost ~]$
```

Step2: Create class for loading Hbase tables

Create a project load-hbase-tables and create a scala class LoadHBaseTables which is explained below:

- Import all the dependent packages and create the package

```
package final_project
```

```

import org.apache.log4j.{Level, LogManager, PropertyConfigurator}
import org.apache.spark._
import org.apache.spark.sql.SQLContext
import org.apache.hadoop.hbase.util.Bytes
import org.apache.hadoop.hbase.client.{ Put, HTable }
import org.apache.hadoop.hbase.HBaseConfiguration
import org.apache.hadoop.hbase.HTableDescriptor
import org.apache.hadoop.hbase.HColumnDescriptor
import org.apache.hadoop.hbase.client.HBaseAdmin
import org.apache.hadoop.hbase.client.HTable
import org.apache.spark.rdd.RDD
import org.apache.hadoop.hbase.mapreduce.TableInputFormat
import org.apache.log4j.{Level, LogManager, PropertyConfigurator}

```

- Create object LoadHBaseTables and define main method and logs
-

```

object LoadHBaseTables {

  def main(args: Array[String]) {
    val log = LogManager.getLogger
    log.setLevel(Level.INFO)
  }
}

```

- Get all the arguments, argument 1 is for filePath of lookup file, argument 2 is table name, argument 3 is composite fields of separated fields of Column Family and Column, which can be multiple

```

val filePath = args(0)
val tablename = args(1)
val columnFamilyField = args(2)

```

```

val columnFamilyFieldList = columnFamilyField.split(",")
val columnFamilyFieldListLength = columnFamilyFieldList.length
    - Create configuration, Spark Context, SQL Context
val conf = new SparkConf().setAppName("LoadHbaseTable").setMaster("local[2]")
val sc = new SparkContext(conf)
sc.setLogLevel("ERROR")
val sqlContext = new SQLContext(sc)

    - Create HBase configuration
val hconf = HBaseConfiguration.create
hconf.set(TableInputFormat.INPUT_TABLE, tablename)
val admin = new HBaseAdmin(hconf)
    - Check if tables is already created or not . If not created, create table and its column family
if (!admin.isTableAvailable(tablename)) {

    log.info("Creating table " + tablename)
    val tableDescription = new HTableDescriptor(tablename)
    tableDescription.addFamily(new HColumnDescriptor(columnFamilyFieldList(0).getBytes()))
    admin.createTable(tableDescription)
    if (admin.isTableAvailable(tablename)) {
        log.info("Table " + tablename + " is created successfully")
    }
} else {
    log.warn("Table " + tablename + " already exists")
}

```

- Get the table and

```
val table = new HTable(hconf, tablename)
```

- Process the csv file which has lookup data and create tuples using map with key and value. If there are multiple fields given, value is taken as a tuple

```

val file = sc.textFile("file://" + filePath)

var records1:Array[(String,String)] = null
var records2:Array[(String,String,String)] = null

if (columnFamilyFieldListLength == 2) {
    records1 = file.map(_.split(",")).map(x => (x(0), x(1))).collect
} else if (columnFamilyFieldListLength == 4) {
    records2 = file.map(_.split(",")).map(x => (x(0), x(1), x(2))).collect
}

- Save the columns

if (columnFamilyFieldListLength == 2) {
    for(record <- records1) {
        var p = new Put(new String(record._1).getBytes())
        p.add(columnFamilyFieldList(0).getBytes(), columnFamilyFieldList(1).getBytes(), new
String(record._2).getBytes())
        table.put(p)
    }
} else if (columnFamilyFieldListLength == 4) {
    for(record <- records2) {
        var p = new Put(new String(record._1).getBytes())
        p.add(columnFamilyFieldList(0).getBytes(), columnFamilyFieldList(1).getBytes(), new
String(record._2).getBytes())
        table.put(p)
        var q = new Put(new String(record._1).getBytes())
        q.add(columnFamilyFieldList(2).getBytes(), columnFamilyFieldList(3).getBytes(), new
String(record._3).getBytes())
    }
}

```

```

        table.put(q)

    }

}

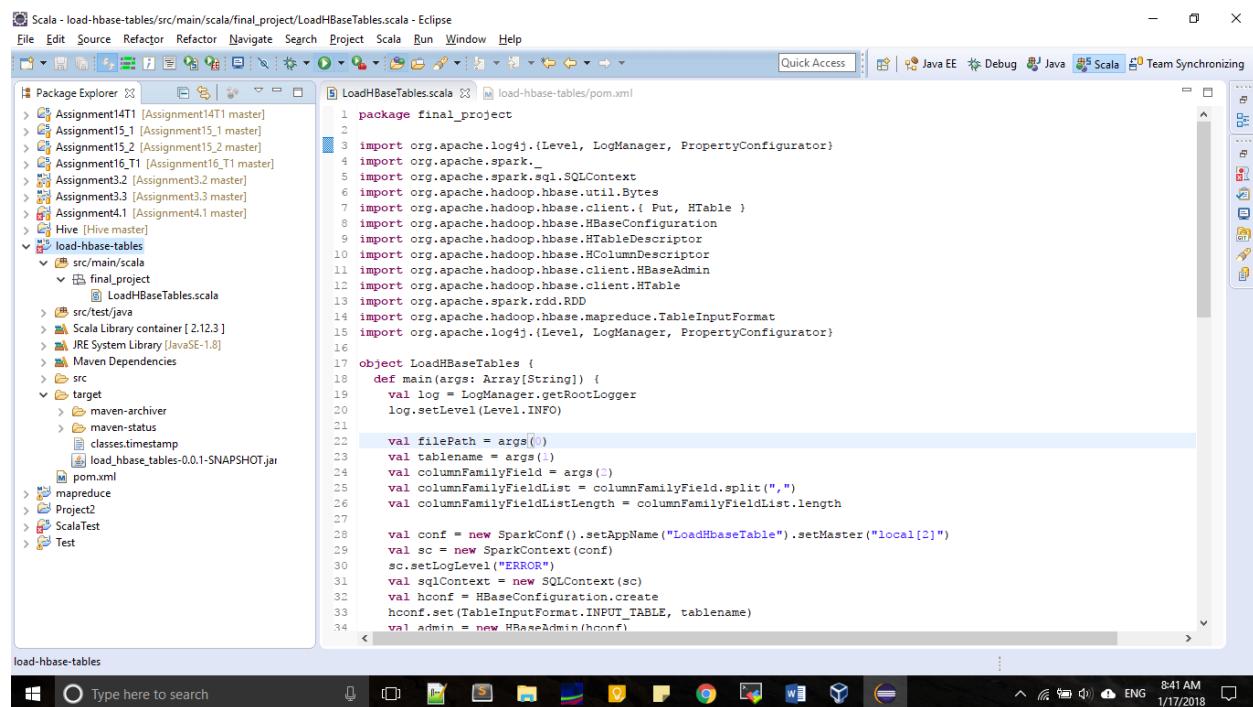
table.flushCommits()

}

}

```

Screenshots are as below:



The screenshot shows the Eclipse IDE interface with the following details:

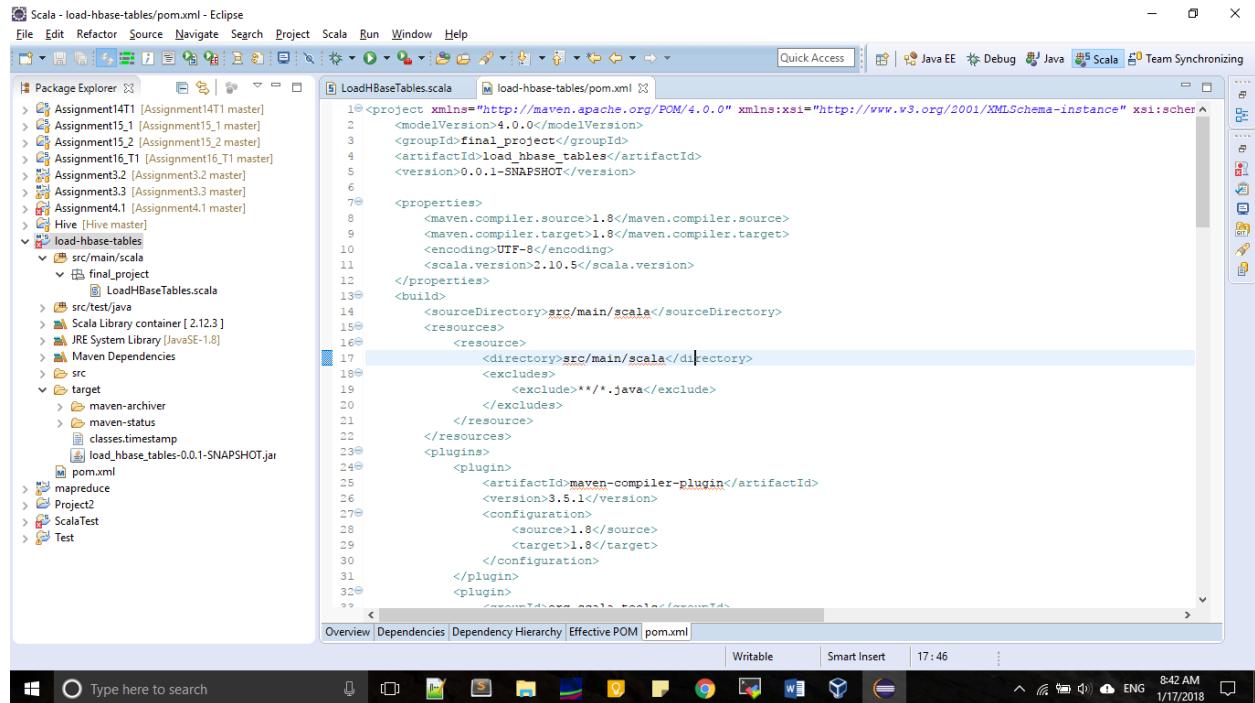
- Title Bar:** Scala - load-hbase-tables/src/main/scala/final_project/LoadHBaseTables.scala - Eclipse
- Menu Bar:** File, Edit, Refactor, Navigate, Project, Scala, Run, Window, Help
- Toolbars:** Quick Access, Java EE, Debug, Java, Scala, Team Synchronizing
- Left Sidebar (Package Explorer):** Lists various Scala projects and files, including Assignment14T1, Assignment15_1, Assignment15_2, Assignment16_T1, Assignment16_2, Assignment3_3, Assignment4_1, Hive, load-hbase-tables, src/main/scala, final_project, LoadHBaseTables.scala, src/test/java, Scala Library container [2.12.3], JRE System Library [JavaSE-1.8], Maven Dependencies, src, target, and a local Maven project.
- Central Editor:** Displays the Scala code for `LoadHBaseTables.scala`. The code initializes a SparkContext, creates an HBaseAdmin object, and then checks if a table exists. If it doesn't, it creates the table. Finally, it reads data from a file and writes it to the HBase table.
- Bottom Status Bar:** Shows Writable, Smart Insert, 22:27, and a date/time stamp of 1/17/2018 8:41 AM.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with several Scala files (Assignment14T1, Assignment15_1, Assignment15_2, Assignment16_T1, Assignment3_2, Assignment3_3, Assignment4_1) and a Hive master.
- LoadHbaseTables.scala:** The current file being edited, located at `src/main/scala/load-hbase-tables`. It contains Scala code for reading from a file and writing to HBase. The code uses `sc.textFile` to read from a file, `map` and `collect` operations to process records, and `Put` objects to add data to HBase tables.
- Toolbar:** Standard Eclipse toolbar with icons for file operations, search, and project navigation.
- Top Bar:** Shows the title "Scala - load-hbase-tables/src/main-scala/final_project/LoadHBaseTables.scala - Eclipse" and the menu bar: File, Edit, Refactor, Navigate, Search, Project, Scala, Run, Window, Help.
- Right Side:** Eclipse's "Synchronizing" view, which displays a list of synchronized projects and their status.

Step3: Compile the file using eclipse and run the code

- Define the pom file

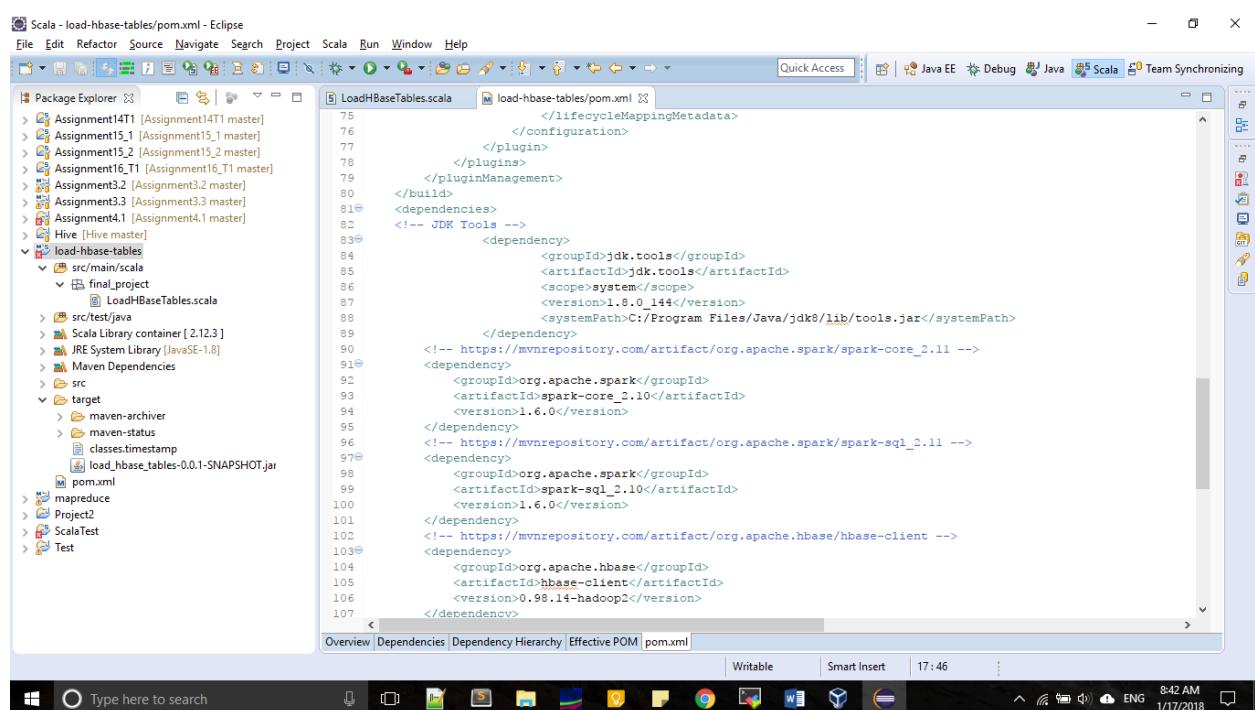


The screenshot shows the Eclipse IDE interface with the 'Scala - load-hbase-tables/pom.xml - Eclipse' window active. The left side displays the 'Package Explorer' containing various Maven projects and dependencies. The right side shows the XML code for the 'pom.xml' file, specifically focusing on the build configuration section.

```

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>final_project</groupId>
  <artifactId>load_hbase_tables</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
    <encoding>UTF-8</encoding>
    <scala.version>2.10.5</scala.version>
  </properties>
  <build>
    <sourceDirectory>src/main/scala</sourceDirectory>
    <resources>
      <resource>
        <directory>src/main/scala</directory>
        <excludes>
          <exclude>**/*.java</exclude>
        </excludes>
      </resource>
    </resources>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.5.1</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>

```



This screenshot shows the same Eclipse IDE environment, but the focus is on the dependency management section of the 'pom.xml' file. It lists several dependencies, including JDK tools, Apache Spark components, and HBase client.

```

<dependency>
  <groupId>jdk.tools</groupId>
  <artifactId>jdk.tools</artifactId>
  <scope>system</scope>
  <version>1.8.0_141</version>
  <systemPath>C:/Program Files/Java/jdk8/lib/tools.jar</systemPath>
</dependency>
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-core_2.11</artifactId>
  <version>1.6.0</version>
</dependency>
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-sql_2.11</artifactId>
  <version>1.6.0</version>
</dependency>
<dependency>
  <groupId>org.apache.hbase</groupId>
  <artifactId>hbase-client</artifactId>
  <version>0.98.14-hadoop2</version>
</dependency>

```

- Compile using maven install, screenshot is as below:

The screenshot shows the Eclipse IDE interface with the following details:

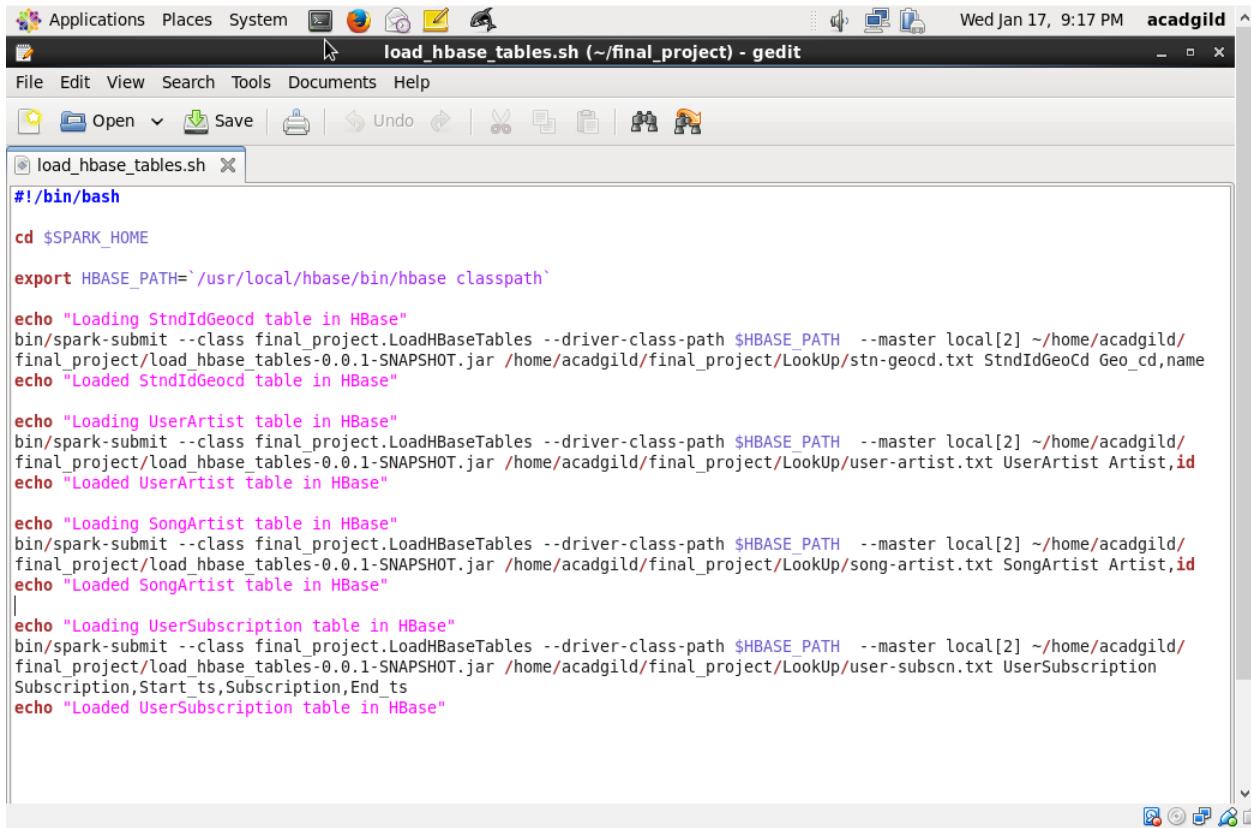
- Project Explorer:** Shows the project structure with several assignment and test modules.
- Maven Dependencies:** Shows the dependency tree for the project.
- Pom.xml:** The Maven configuration file is open, showing dependencies for Apache HBase and Hadoop.
- Console:** Displays the output of the Maven build command, showing the compilation of the project and the creation of the 'load_hbase_tables-0.0.1-SNAPSHOT.jar' file.
- Taskbar:** Shows the Windows taskbar with various application icons and the system clock indicating 8:43 AM on 1/17/2018.

```

<!-- https://mvnrepository.com/artifact/org.apache.hbase/hbase-common -->
<dependency>
    <groupId>org.apache.hbase</groupId>
    <artifactId>hbase-common</artifactId>
    <version>0.98.14-hadoop2</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.apache.hbase/hbase-protocol -->
<dependency>
    <groupId>org.apache.hbase</groupId>
    <artifactId>hbase-protocol</artifactId>
    <version>0.98.14-hadoop2</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.apache.hbase/hbase-hadoop2-compat -->
<dependency>
    <groupId>org.apache.hbase</groupId>
    <artifactId>hbase-hadoop2-compat</artifactId>
    <version>0.98.14-hadoop2</version>
</dependency>

```

- Write a script to load all the tables:



The screenshot shows a Gedit text editor window titled "load_hbase_tables.sh (~/final_project) - gedit". The status bar indicates the date and time as "Wed Jan 17, 9:17 PM" and the user as "acadgild". The file menu includes "File", "Edit", "View", "Search", "Tools", "Documents", and "Help". The toolbar contains icons for "Open", "Save", "Undo", and "Redo". The main text area contains a bash script for loading HBase tables:

```
#!/bin/bash

cd $SPARK_HOME

export HBASE_PATH=/usr/local/hbase/bin/hbase classpath

echo "Loading StndIdGeocd table in HBase"
bin/spark-submit --class final_project.LoadHBaseTables --driver-class-path $HBASE_PATH --master local[2] ~/home/acadgild/final_project/load_hbase_tables-0.0.1-SNAPSHOT.jar /home/acadgild/final_project/LookUp/stn-geocd.txt StndIdGeoCd Geo_cd,name
echo "Loaded StndIdGeocd table in HBase"

echo "Loading UserArtist table in HBase"
bin/spark-submit --class final_project.LoadHBaseTables --driver-class-path $HBASE_PATH --master local[2] ~/home/acadgild/final_project/load_hbase_tables-0.0.1-SNAPSHOT.jar /home/acadgild/final_project/LookUp/user-artist.txt UserArtist Artist,id
echo "Loaded UserArtist table in HBase"

echo "Loading SongArtist table in HBase"
bin/spark-submit --class final_project.LoadHBaseTables --driver-class-path $HBASE_PATH --master local[2] ~/home/acadgild/final_project/load_hbase_tables-0.0.1-SNAPSHOT.jar /home/acadgild/final_project/LookUp/song-artist.txt SongArtist Artist,id
echo "Loaded SongArtist table in HBase"

echo "Loading UserSubscription table in HBase"
bin/spark-submit --class final_project.LoadHBaseTables --driver-class-path $HBASE_PATH --master local[2] ~/home/acadgild/final_project/load_hbase_tables-0.0.1-SNAPSHOT.jar /home/acadgild/final_project/LookUp/user-subscn.txt UserSubscription Subscription,Start_ts,Subscription,End_ts
echo "Loaded UserSubscription table in HBase"
```

- Run the script to load all the tables

Applications Places System acadgild@localhost:~/project_music_data_analysis

File Edit View Search Terminal Tabs Help

```
acadgild@localhost:~/project_music_data_analysis [acadgild@localhost project music analysis]$ sh load_hbase_tables.sh
Loading StndIdGeocd table in HBase
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hbase/lib/slf4j-log4j12-1.6.4.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/spark/spark-1.6.0-bin-hadoop2.6/lib/spark-assembly-1.6.0-hadoop2.6.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
2018-01-03 19:54:19,862 INFO [main] spark.SparkContext: Running Spark version 1.6.0
2018-01-03 19:54:20,957 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in-java classes where applicable
2018-01-03 19:54:21,240 WARN [main] util.Utils: Your hostname, localhost.localdomain resolves to a loopback address: 127.0.0.1, but we couldn't find any external IP address!
2018-01-03 19:54:21,241 WARN [main] util.Utils: Set SPARK_LOCAL_IP if you need to bind to another address
2018-01-03 19:54:21,368 INFO [main] spark.SecurityManager: Changing view acls to: acadgild
2018-01-03 19:54:21,370 INFO [main] spark.SecurityManager: Changing modify acls to: acadgild
2018-01-03 19:54:21,373 INFO [main] spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(acadgild); users with modify permissions: Set(acadgild)
2018-01-03 19:54:23,205 INFO [main] util.Utils: Successfully started service 'sparkDriver' on port 46415.
2018-01-03 19:54:24,582 INFO [sparkDriverActorSystem-akka.actor.default-dispatcher-5] slf4j.Slf4jLogger: Slf4jLogger started
2018-01-03 19:54:24,791 INFO [sparkDriverActorSystem-akka.actor.default-dispatcher-5] Remoting: Starting remoting
2018-01-03 19:54:25,576 INFO [sparkDriverActorSystem-akka.actor.default-dispatcher-5] Remoting: Remoting started; listening on addresses :[akka.tcp://sparkDriverActorSystem@127.0.0.1:38453]
2018-01-03 19:54:25,609 INFO [main] util.Utils: Successfully started service 'sparkDriverActorSystem' on port 38453.
2018-01-03 19:54:25,684 INFO [main] spark.SparkEnv: Registering MapOutputTracker
2018-01-03 19:54:25,783 INFO [main] spark.SparkEnv: Registering BlockManagerMaster
2018-01-03 19:54:25,830 INFO [main] storage.DiskBlockManager: Created local directory at /tmp/blockmgr-c3ce1fb-12f7-4d68-b5a3-1f7c4c405a4d
2018-01-03 19:54:25,911 INFO [main] storage.MemoryStore: MemoryStore started with capacity 517.4 MB
2018-01-03 19:54:26,214 INFO [main] spark.SparkEnv: Registering OutputCommitCoordinator
2018-01-03 19:54:27,192 INFO [main] server.Server: jetty-8.y.z-SNAPSHOT
2018-01-03 19:54:27,377 INFO [main] server.AbstractConnector: Started SelectChannelConnector@0.0.0.0:4040
2018-01-03 19:54:27,377 INFO [main] util.Utils: Successfully started service 'SparkUI' on port 4040.
2018-01-03 19:54:27,390 INFO [main] ui.SparkUI: Started SparkUI at http://127.0.0.1:4040
2018-01-03 19:54:27,484 INFO [main] spark.HttpFileServer: HTTP File server directory is /tmp/spark-5c586a27-e5e5-49c6-aae1-85ff2d0e4ed/httpd-d920e5d3-d8a1-4af4-8e60-01d1a2a6b929
```

The screenshot shows a terminal window titled "acadgild@localhost:~/project_music_data_analysis". The window contains a log of HBase startup messages. The log includes details about ZooKeeper connections, session establishment, and the loading of UserSubscription tables. The terminal window has a standard Linux desktop interface with icons at the top and a scroll bar on the right.

```
2018-01-03 19:55:42,399 INFO [main] zookeeper.ZooKeeper: Client environment:java.io.tmpdir=/tmp
2018-01-03 19:55:42,399 INFO [main] zookeeper.ZooKeeper: Client environment:java.compiler=<NA>
2018-01-03 19:55:42,400 INFO [main] zookeeper.ZooKeeper: Client environment:os.name=Linux
2018-01-03 19:55:42,400 INFO [main] zookeeper.ZooKeeper: Client environment:os.arch=amd64
2018-01-03 19:55:42,400 INFO [main] zookeeper.ZooKeeper: Client environment:os.version=2.6.32-573.el6.x86_64
2018-01-03 19:55:42,400 INFO [main] zookeeper.ZooKeeper: Client environment:user.name=acadgild
2018-01-03 19:55:42,400 INFO [main] zookeeper.ZooKeeper: Client environment:user.home=/home/acadgild
2018-01-03 19:55:42,400 INFO [main] zookeeper.ZooKeeper: Client environment:user.dir=/usr/local/spark/spark-1.6.0-bin-hadoop2.6
2018-01-03 19:55:42,405 INFO [main] zookeeper.ZooKeeper: Initiating client connection, connectString=localhost:2181 sessionTimeout=90000 watcher=hconnection-0x2b10ace90x0, quorum=localhost:2181, baseZNode=/hbase
2018-01-03 19:55:42,516 INFO [main-SendThread[localhost:2181]] zookeeper.ClientCnxn: Opening socket connection to server localhost/127.0.0.1:2181. Will not attempt to authenticate using SASL (unknown error)
2018-01-03 19:55:42,560 INFO [main-SendThread[localhost:2181]] zookeeper.ClientCnxn: Socket connection established to localhost/127.0.0.1:2181, initiating session
2018-01-03 19:55:42,583 INFO [main-SendThread[localhost:2181]] zookeeper.ClientCnxn: Session establishment complete on server localhost/127.0.0.1:2181, sessionid = 0x160bc642461000c, negotiated timeout = 40000
2018-01-03 19:55:46,966 INFO [main] zookeeper.RecoverableZooKeeper: Process identifier=catalogtracker-on-hconnection-0x2b10ace9 connecting to ZooKeeper ensemble=localhost:2181
2018-01-03 19:55:46,966 INFO [main] zookeeper.ZooKeeper: Initiating client connection, connectString=localhost:2181 sessionTimeout=90000 watcher=catalogtracker-on-hconnection-0x2b10ace90x0, quorum=localhost:2181, baseZNode=/hbase
2018-01-03 19:55:46,984 DEBUG [main] catalog.CatalogTracker: Starting catalog tracker org.apache.hadoop.hbase.catalog.CatalogTracker@5ad1904f
2018-01-03 19:55:46,986 INFO [main-SendThread[localhost:2181]] zookeeper.ClientCnxn: Opening socket connection to server localhost/127.0.0.1:2181. Will not attempt to authenticate using SASL (unknown error)
2018-01-03 19:55:46,987 INFO [main-SendThread[localhost:2181]] zookeeper.ClientCnxn: Socket connection established to localhost/127.0.0.1:2181, initiating session
2018-01-03 19:55:46,988 INFO [main-SendThread[localhost:2181]] zookeeper.ClientCnxn: Session establishment complete on server localhost/127.0.0.1:2181, sessionid = 0x160bc642461000d, negotiated timeout = 40000
2018-01-03 19:55:47,013 DEBUG [main] catalog.CatalogTracker: Stopping catalog tracker org.apache.hadoop.hbase.catalog.CatalogTracker@5ad1904f
2018-01-03 19:55:47,016 INFO [main-EventThread] zookeeper.ClientCnxn: EventThread shut down
2018-01-03 19:55:47,018 INFO [main] zookeeper.ZooKeeper: Session: 0x160bc642461000d closed
2018-01-03 19:55:51,431 DEBUG [main] client.ClientSmallScanner: Finished with small scan at {ENCODED => 1588230740, NAME => 'hbase:meta,,1', STARTKEY => '', ENDKEY => ''}
Loaded UserSubscription table in HBase
[acadgild@localhost project_music_data_analysis]$
```

Step3: Verify that all the lookup tables are populated correctly

- Verify all the tables are populated correctly first login to HBase using hbase shell
- Table StnIdGeoCd is verified using

scan 'StnIdGeoCd'

Screenshot is as below:

```
[acadgild@localhost ~]$ hbase shell
2018-01-03 19:57:27,210 INFO  [main] Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.98.14-hadoop2, r4e4aabb93b52f1b0fef6b66edd06ec8923014dec, Tue Aug 25 22:35:44 PDT 2015

hbase(main):001:0> scan 'StndIdGeoCd'
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hbase/lib/slf4j-log4j12-1.6.4.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
2018-01-03 19:58:03,937 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using b
uiltin-java classes where applicable
ROW
  ST400          COLUMN+CELL
  ST401          column=Geo_cd:name, timestamp=1514989478961, value=A
  ST402          column=Geo_cd:name, timestamp=1514989478997, value=AU
  ST403          column=Geo_cd:name, timestamp=1514989479007, value=AP
  ST404          column=Geo_cd:name, timestamp=1514989479010, value=J
  ST405          column=Geo_cd:name, timestamp=1514989479016, value=E
  ST406          column=Geo_cd:name, timestamp=1514989479020, value=A
  ST407          column=Geo_cd:name, timestamp=1514989479028, value=AU
  ST408          column=Geo_cd:name, timestamp=1514989479029, value=AP
  ST409          column=Geo_cd:name, timestamp=1514989479039, value=E
  ST410          column=Geo_cd:name, timestamp=1514989479048, value=E
  ST411          column=Geo_cd:name, timestamp=1514989479053, value=A
  ST412          column=Geo_cd:name, timestamp=1514989479057, value=A
  ST413          column=Geo_cd:name, timestamp=1514989479061, value=AP
  ST414          column=Geo_cd:name, timestamp=1514989479064, value=J
  ST414          column=Geo_cd:name, timestamp=1514989479068, value=E

15 row(s) in 1.1780 seconds

hbase(main):002:0>
```

- Table UserArtist is verified using

scan 'UserArtist'

Screenshot is as below

```
Applications Places System File Edit View Search Terminal Tabs Help acadgild@localhost:~ acadgild@localhost:~/project_music_data_analysis acadgild@localhost:~ hbase(main):035:0* hbase(main):036:0* hbase(main):037:0* hbase(main):038:0* hbase(main):039:0* hbase(main):040:0* hbase(main):041:0* hbase(main):042:0* hbase(main):043:0* hbase(main):044:0* hbase(main):045:0* hbase(main):046:0* scan 'UserArtist' ROW COLUMN+CELL U100 column=Artist:id, timestamp=1514989502573, value=A300&A301&A302 U101 column=Artist:id, timestamp=1514989502616, value=A301&A302 U102 column=Artist:id, timestamp=1514989502621, value=A302 U103 column=Artist:id, timestamp=1514989502625, value=A303&A301&A302 U104 column=Artist:id, timestamp=1514989502629, value=A304&A301 U105 column=Artist:id, timestamp=1514989502632, value=A305&A301&A302 U106 column=Artist:id, timestamp=1514989502638, value=A301&A302 U107 column=Artist:id, timestamp=1514989502643, value=A302 U108 column=Artist:id, timestamp=1514989502647, value=A300&A303&A304 U109 column=Artist:id, timestamp=1514989502651, value=A301&A303 U110 column=Artist:id, timestamp=1514989502654, value=A302&A301 U111 column=Artist:id, timestamp=1514989502658, value=A303&A301 U112 column=Artist:id, timestamp=1514989502670, value=A304&A301 U113 column=Artist:id, timestamp=1514989502682, value=A305&A302 U114 column=Artist:id, timestamp=1514989502686, value=A300&A301&A302 15 row(s) in 0.2200 seconds hbase(main):047:0>
```

- Table SongArtist is verified using

scan 'SongArtist'

Screenshot is as below:

The screenshot shows a terminal window titled "acadgild@localhost:~". The window contains the following HBase command output:

```
acadgild@localhost:~/project_music_data_analysis
```

hbase(main):095:0*

hbase(main):096:0*

hbase(main):097:0* scan 'SongArtist'

ROW	COLUMN+CELL
S200	column=Artist:id, timestamp=1514989525967, value=A300
S201	column=Artist:id, timestamp=1514989526015, value=A301
S202	column=Artist:id, timestamp=1514989526021, value=A302
S203	column=Artist:id, timestamp=1514989526026, value=A303
S204	column=Artist:id, timestamp=1514989526036, value=A304
S205	column=Artist:id, timestamp=1514989526041, value=A301
S206	column=Artist:id, timestamp=1514989526047, value=A302
S207	column=Artist:id, timestamp=1514989526053, value=A303
S208	column=Artist:id, timestamp=1514989526057, value=A304
S209	column=Artist:id, timestamp=1514989526061, value=A305

10 row(s) in 0.1050 seconds

hbase(main):098:0*

hbase(main):099:0*

hbase(main):100:0*

hbase(main):101:0*

hbase(main):102:0*

hbase(main):103:0*

hbase(main):104:0*

hbase(main):105:0*

hbase(main):106:0*

hbase(main):107:0*

hbase(main):108:0*

hbase(main):109:0*

hbase(main):110:0*

hbase(main):111:0*

hbase(main):112:0*

hbase(main):113:0*

hbase(main):114:0*

hbase(main):115:0*

hbase(main):116:0*

hbase(main):117:0*

hbase(main):118:0*

- Table UserSubscription is verified using

scan 'UserSubscription'

Screenshot is as below:

```

Applications Places System acadgild@localhost:~ Wed Jan 3, 8:01 PM acadgild
File Edit View Search Terminal Tabs Help
acadgild@localhost:~/project_music_data_analysis acadgild@localhost:~
hbase(main):153:0*
hbase(main):154:0*
hbase(main):155:0* scan 'UserSubscription'
ROW COLUMN+CELL
U100 column=Subscription:End_ts, timestamp=1514989551637, value=1465130523
U100 column=Subscription:Start_ts, timestamp=1514989551608, value=1465230523
U101 column=Subscription:End_ts, timestamp=1514989551652, value=1475130523
U101 column=Subscription:Start_ts, timestamp=1514989551645, value=1465230523
U102 column=Subscription:End_ts, timestamp=1514989551662, value=1475130523
U102 column=Subscription:Start_ts, timestamp=1514989551657, value=1465230523
U103 column=Subscription:End_ts, timestamp=1514989551669, value=1475130523
U103 column=Subscription:Start_ts, timestamp=1514989551666, value=1465230523
U104 column=Subscription:End_ts, timestamp=1514989551679, value=1475130523
U104 column=Subscription:Start_ts, timestamp=1514989551673, value=1465230523
U105 column=Subscription:End_ts, timestamp=1514989551685, value=1475130523
U105 column=Subscription:Start_ts, timestamp=1514989551683, value=1465230523
U106 column=Subscription:End_ts, timestamp=1514989551692, value=1485130523
U106 column=Subscription:Start_ts, timestamp=1514989551687, value=1465230523
U107 column=Subscription:End_ts, timestamp=1514989551699, value=1455130523
U107 column=Subscription:Start_ts, timestamp=1514989551696, value=1465230523
U108 column=Subscription:End_ts, timestamp=1514989551708, value=1465230623
U108 column=Subscription:Start_ts, timestamp=1514989551702, value=1465230523
U109 column=Subscription:End_ts, timestamp=1514989551714, value=1475130523
U109 column=Subscription:Start_ts, timestamp=1514989551712, value=1465230523
U110 column=Subscription:End_ts, timestamp=1514989551719, value=1475130523
U110 column=Subscription:Start_ts, timestamp=1514989551717, value=1465230523
U111 column=Subscription:End_ts, timestamp=1514989551727, value=1475130523
U111 column=Subscription:Start_ts, timestamp=1514989551725, value=1465230523
U112 column=Subscription:End_ts, timestamp=1514989551741, value=1475130523
U112 column=Subscription:Start_ts, timestamp=1514989551732, value=1465230523
U113 column=Subscription:End_ts, timestamp=1514989551749, value=1485130523
U113 column=Subscription:Start_ts, timestamp=1514989551746, value=1465230523
U114 column=Subscription:End_ts, timestamp=1514989551756, value=1468130523
U114 column=Subscription:Start_ts, timestamp=1514989551753, value=1465230523
15 row(s) in 0.2330 seconds

hbase(main):156:0>

```

MODULE2: Handling Web and Mobile Data

Step1: Write a scala class for handling Web data

- Write class WebMusicDataProcessor to process web music data stored in /data/web/file-1.xml and store it as dataframe

-Import dependent packages

```
package final_project
```

```
import org.apache.spark._
```

```
import scala.xml.XML
```

```
import org.apache.spark.sql.DataFrame
import scala.collection.mutable.ListBuffer
import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.types._
import org.apache.log4j.{ Level, LogManager, PropertyConfigurator }
import org.apache.spark.broadcast.Broadcast
import java.text.SimpleDateFormat
import scala.collection.mutable.HashMap
```

```
// Define CustomException this is to solve the continue
```

```
case class CustomException(message:String) extends Exception(message)
```

```
// Define the class with WebMusicDataProcessor with parametes
class WebMusicDataProcessor(param: String, context: SparkContext, sqc: SQLContext) extends Serializable {
    val filePath: String = param
    val sc: SparkContext = context
    val sqlContext:SQLContext = sqc
```

```
// Define the method which does the processing of data and return as dataframe
```

```
def processData(): DataFrame = {
    val log = LogManager.getRootLogger
    log.setLevel(Level.INFO)
    // val sqlContext = new SQLContext(sc)
    import sqlContext.implicits._
```

```

// Define dateFormat as yyyy-MM-dd HH:mm:ss and recordListBuffer which will act as buffer
for storing data

val dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")

val recordListBuffer = new ListBuffer[(String, String, String, Long, Long, Long, String, String, String, String)]()

// Load data from XML path

val xml = XML.loadFile(filePath)

- Process all the fields user_id, song_id, artist_id, timestamp, start_ts,end_ts. Handle all null
conditions

for (tag <- xml.child) {

  val userId = (tag \ "user_id").text
  val songId = (tag \ "song_id").text
  var artistId = (tag \ "artist_id").text

  val timestamp = (tag \ "timestamp").text
  var timestampLong:Long = 0
  if (!timestamp.equals("")) {
    timestampLong = dateFormat.parse(timestamp).getTime()
  }

  val startTs = (tag \ "start_ts").text
  var startTsLong:Long = 0
  if (!startTs.equals("")) {
    startTsLong = dateFormat.parse(startTs).getTime()
  }

  val endTs = (tag \ "end_ts").text
  var endTsLong:Long = 0
  if (!endTs.equals("")) {

```

```

endTsLong = dateFormat.parse(endTs).getTime()

}

var geoCd = (tag \ "geo_cd").text

val stationId = (tag \ "station_id").text

val songEndType = (tag \ "song_end_type").text

var like = (tag \ "like").text

if (like.equals("")) like = "0"

var dislike = (tag \ "dislike").text

if (dislike.equals("")) dislike = "0"

try {

// Continue with the record in case fileds are blank

if (userId.equals("") && songId.equals("") && artistId.equals("")) {

throw CustomException("Record is blank")

} else {

recordListBuffer += ((userId, songId, artistId, timestampLong, startTsLong, endTsLong, geoCd,
stationId, songEndType, like, dislike))

}

} catch {

case CustomException(msg) => msg

}

}

// From the buffer convert to dataFrame recordDF having fields User_id, Songs_id, Artist_id, Timestamp,
// Start_ts, End_ts, Geo_cd, Station_id, Song_end_type, Likes, Dislikes

val recordList = recordListBuffer.toList

```

```

val recordRDD = sc.parallelize(recordList)

val recordDF = recordRDD.toDF("User_id", "Songs_id", "Artist_id", "Timestamp",
    "Start_ts", "End_ts", "Geo_cd", "Station_id", "Song_end_type",
    "Likes", "Dislikes")

log.info("Number of records =" + recordDF.count)

log.info("Showing records for Web Music Data ")

recordDF.show

return recordDF
}

}

```

Screenshot is as below:

```

Scala - process-music-data/src/scala/final_project/WebMusicDataProcessor.scala - Eclipse
File Edit Source Refactor Navigate Search Project Scala Run Window Help
Quick Access Java EE Debug J Java Scala Team Synchronizing
Package Explorer WebMusicDataProcessor.scala
1 package final_project
2 import org.apache.spark...
12
13 case class CustomException(message:String) extends Exception(message)
14
15 class WebMusicDataProcessor(param: String, context: SparkContext, sqc: SQLContext) extends Serializable {
16   val filePath: String = param
17   val sc: SparkContext = context
18   val sqlContext:SQLContext = sqc
19
20   def processData(): DataFrame = {
21     val log = LogManager.getRootLogger
22     log.setLevel(Level.INFO)
23     // val sqlContext = new SQLContext(sc)
24     import sqlContext.implicits._
25     val dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")
26     val recordListBuffer = new ListBuffer[(String, String, String, Long, Long, Long, String, String, String, String)]
27
28     val xml = XML.loadFile(filePath)
29
30     for (tag <- xml.child) {
31       val userId = (tag \ "user_id").text
32       val songId = (tag \ "song_id").text
33       var artistId = (tag \ "artist_id").text
34
35       val timestamp = (tag \ "timestamp").text
36       var timestampLong:Long = ...
37       if (!timestamp.equals("")) {
38         timestampLong = dateFormat.parse(timestamp).getTime()
39       }
40     }
41     val startTs = (tag \ "start_ts").text
42     var startTsLong:Long = ...
43     if (!startTs.equals("")) {
44
}

```

```

Scala - process-music-data/src/scala/final_project/WebMusicDataProcessor.scala - Eclipse
File Edit Refactor Navigate Search Project Scala Run Window Help
Quick Access Java EE Debug Java Scala Team Synchronizing
Package Explorer WebMusicDataProcessor.scala
Assignment14T1 [Assignment14T1 master]
Assignment15_1 [Assignment15_1 master]
Assignment15_2 [Assignment15_2 master]
Assignment16_T1 [Assignment16_T1 master]
Assignment3_2 [Assignment3_2 master]
Assignment3_3 [Assignment3_3 master]
Assignment4_1 [Assignment4_1 master]
Hive [Hive master]
load-base-tables
mapreduce
process-music-data
src/scala
final_project
MobileMusicDataProcessor.scala
MusicDataAnalyzer.scala
MusicDataEnricher.scala
MusicDataPopulateMapFromLookupTables.s
MusicDataProcessorApp.scala
WebMusicDataProcessor.scala
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
pom.xml
Project2
ScalaTest
Test

```

```

41     val startTs = (tag \ "start_ts").text
42     var startTsLong:Long = 0
43     if (!startTs.equals("")) {
44       startTsLong = dateFormat.parse(startTs).getTime()
45     }
46     val endTs = (tag \ "end_ts").text
47     var endTsLong:Long = 0
48     if (!endTs.equals("")) {
49       endTsLong = dateFormat.parse(endTs).getTime()
50     }
51     var geoCd = (tag \ "geo_cd").text
52
53     val stationId = (tag \ "station_id").text
54
55     val songEndType = (tag \ "song_end_type").text
56     var like = (tag \ "like").text
57     if (like.equals("")) like = "0"
58     var dislike = (tag \ "dislike").text
59     if (dislike.equals("")) dislike = "0"
60
61     try {
62       if (userId.equals("") && songId.equals("") && artistId.equals("")) {
63         throw CustomException("Record is blank")
64       } else {
65         recordListBuffer += ((userId, songId, artistId, timestampLong, startTsLong, endTsLong, geoCd, stationId,
66           )
67       }
68     } catch {
69     case CustomException(msg) => msg
70   }
71     val recordList = recordListBuffer.toList
72
73     val recordRDD = sc.parallelize(recordList)

```

Step2: define class for processing Mobile data

- Define a class MobileMusicDataProcessor for processing music data from /data/mob/file.txt

```

// define package as final_project and Import al the dependent packages

package final_project

import org.apache.spark._

import org.apache.spark.sql.DataFrame

import scala.collection.mutable.ListBuffer

import org.apache.spark.sql.SQLContext

import org.apache.spark.sql.types._

import org.apache.log4j.{ Level, LogManager, PropertyConfigurator }

import org.apache.spark.broadcast.Broadcast

import scala.collection.mutable.HashMap

```

```

// Create a case class MusicData with all the fields

case class MusicData(User_id:String, Songs_id:String, Artist_id:String, Timestamp:Long,
Start_ts:Long, End_ts:Long, Geo_cd:String, Station_id:String, Song_end_type:String,
Likes:String, Dislikes:String)

// Define class MobileMusicDataProcessor with all the parameters

class MobileMusicDataProcessor(param: String, context: SparkContext, sqc:SQLContext) extends
Serializable {

  val filePath: String = param
  val sc: SparkContext = context
  val sqlContext:SQLContext = sqc

  // Define method processData
  def processData(): DataFrame = {
    val log = LogManager.getRootLogger
    log.setLevel(Level.INFO)

    import sqlContext.implicits._

    // Load dataset from mobeile file path /data/mob/file.txt into RDD and slipt the fields
    val recordRDD = sc.textFile(filePath)
    val recordFieldsRDD = recordRDD.map(x => x.split(",")).filter(x=> x.length ==11)

    // Convert the RDD to dataframe with case class MusicData
    val recordDF = recordFieldsRDD.map(x => MusicData(x(0),
      x(1),
      x(2),

```

```

    if (x(3).equals("")) 0 else x(3).toLong,
    if (x(4).equals("")) 0 else x(4).toLong,
    if (x(5).equals("")) 0 else x(5).toLong,
    x(6),
    x(7),
    x(8),
    if (x(9).equals("")) "0" else x(9),
    if (x(10).equals("")) "0" else x(10))).toDF

log.info("Number of records for Mobile Music Data =" + recordDF.count)
log.info("Showing records for Mobile Music Data ")
recordDF.show

return recordDF

}

}

```

MODULE3: Enrich and validate data

Step1: Define a class MusicDataEnricher which does the enrichment and validation of fields in dataframe using UDF

```

// Define package final_project and import all the dependent packages
package final_project

import org.apache.spark._
import org.apache.spark.sql.DataFrame
import scala.collection.mutable.ListBuffer
import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.types._
import org.apache.log4j.{ Level, LogManager, PropertyConfigurator }
import org.apache.spark.broadcast.Broadcast
import org.apache.spark.sql.functions.udf
import scala.collection.mutable.HashMap

// Define class MusicDataEnricher with the parameters of all SparkContext, DataFrame and all the
// broadcast maps which has lookup data

class MusicDataEnricher( allDataFrameParam: DataFrame,
broadcastStndIdGeoCdMapParam:Broadcast[Map[String, String]],
broadcastSongArtistMapParam:Broadcast[Map[String, String]], broadcastUserArtistMapParam:
Broadcast[Map[String, String]], broadcastUserSubscriptionParam: Broadcast[Map[String, (Long, Long)]] )
extends Serializable {

    val allDataFrame: DataFrame =allDataFrameParam
    // val stndIdGeoCdMap:HashMap[String, String] = stndIdGeoCdMapParam
    val broadcastStndIdGeoCdMap:Broadcast[Map[String, String]] =
broadcastStndIdGeoCdMapParam
    val broadcastSongArtistMap:Broadcast[Map[String, String]] = broadcastSongArtistMapParam
    val broadcastUserArtistMap:Broadcast[Map[String, String]] = broadcastUserArtistMapParam
    // The UDF method fillNullValueGeoCd will take stationId and geoCd as parameter and if geoCd is not
    // blank then it will take as it is. If geoCd is blank it will use the lookup map broadcastStndIdGeoCdMap
    // using stationId, get the geoCd. If it is not there record will be marked as Invalid
}

```

```

// A new field modified_Geo_cd will be added to the dataframe

def fillNullValueGeoCd = udf((stationId: String, geoCd: String) => {

    if (!geoCd.equals("")) geoCd
    else {
        // val geoCdVal = stndIdGeoCdMap.get(stationId).getOrElse("Invalid")
        val geoCdVal = broadcastStndIdGeoCdMap.value.get(stationId).getOrElse("Invalid")
        geoCdVal
    }
})

// The UDF method fillNullValueArtistId will take songId and artistId as parameter and if artistId is not
// blank then it will take as it is. If artistId is blank it will use the lookup map broadcastSongArtistMap
// using songId, get the artistId. If it is not there record will be marked as Invalid

// A new field modified_Artist_id will be added to the dataframe

def fillNullValueArtistId = udf((songId: String, artistId: String) => {

    if (!artistId.equals("")) artistId
    else {
        val artistIdVal = broadcastSongArtistMap.value.get(songId).getOrElse("Invalid")
        artistIdVal
    }
})

// The UDF method findFollowers will take userId and artistId as parameter.
// Based on usedId from the map broadcastUserArtistMap, artistList is retrieved. If artistId list is blank
// then 0 will be returned. If artistList is not blank then it will be split based on & and create a array
// artistArray. If it contains using artistId, 1 will be returned, else 0 will be returned

// A new field follower will be added to the dataframe

```

```

def findFollowers = udf((userId: String, artistId: String) => {
    val artistList = broadcastUserArtistMap.value.get(userId).getOrElse("")
    if (artistList.equals("")) "0"
    else {
        var artistArray = artistList.split("&")
        if (artistArray contains artistId) "1"
        else "0"
    }
})

// The UDF method findSubscribers will take userId and starts as parameter. Based on userId
// lookup is done on broadcastUserSubscription.value.get(userId). If subscription does not exist, it
// returns 0. If it exists and starts is in between subscription start time and end time return 1 else
// return 0. A new field subscribed is added to the dataframe

def findSubscribers = udf((userId: String, startTs: Long) => {
    val subscriptionTuple = broadcastUserSubscription.value.get(userId).getOrElse((0L, 0L))
    if (subscriptionTuple._1 == 0 && subscriptionTuple._2 == 0) "0"
    else if (startTs >= subscriptionTuple._1 && startTs <= subscriptionTuple._2) "1"
    else "0"
})

```

```

// Using UDF validate records validation is done. If userId, songId are blank return 0. If modifiedArtistId
// or modifiedGeoCdi si Invalid return 0,. If timestamp or start_ts is 0 then 0. If end_ts is less than //
start_ts return 0. Else return 1. Add a new field isValid to the dataframe

def validateRecords = udf((userId: String, songId: String, modifiedArtistId: String, modifiedGeoCd: String,
timestamp:Long, start_ts:Long, end_ts:Long) => {

    if (userId.equals("")) "0"
        else if (songId.equals("")) "0"
            else if (modifiedArtistId.equals("Invalid")) "0"
            else if (modifiedGeoCd.equals("Invalid")) "0"
            else if (timestamp == 0) "0"
                else if(start_ts == 0) "0"
                    else if (end_ts < start_ts) "0"
                        else "1"
})

```

```

// The method enrichData will execute all the UDFs defined above and enrich dataframe with new fields

def enrichData():DataFrame = {

    var newDataFrame = allDataFrame.withColumn("modified_Geo_cd",
fillNullValueGeoCd(allDataFrame("Station_id"), allDataFrame("Geo_cd")))

    newDataFrame = newDataFrame.withColumn("modified_Artist_id",
fillNullValueArtistId(newDataFrame("Songs_id"), newDataFrame("Artist_id")))

    newDataFrame = newDataFrame.withColumn("follower", findFollowers(newDataFrame("User_id"),
newDataFrame("modified_Artist_id")))

    newDataFrame = newDataFrame.withColumn("subscribed", findSubscribers(newDataFrame("User_id"),
newDataFrame("Start_ts")))

    newDataFrame = newDataFrame.withColumn("isValid", validateRecords (newDataFrame("User_id"),
newDataFrame("Songs_id"), newDataFrame("modified_Artist_id"),

```

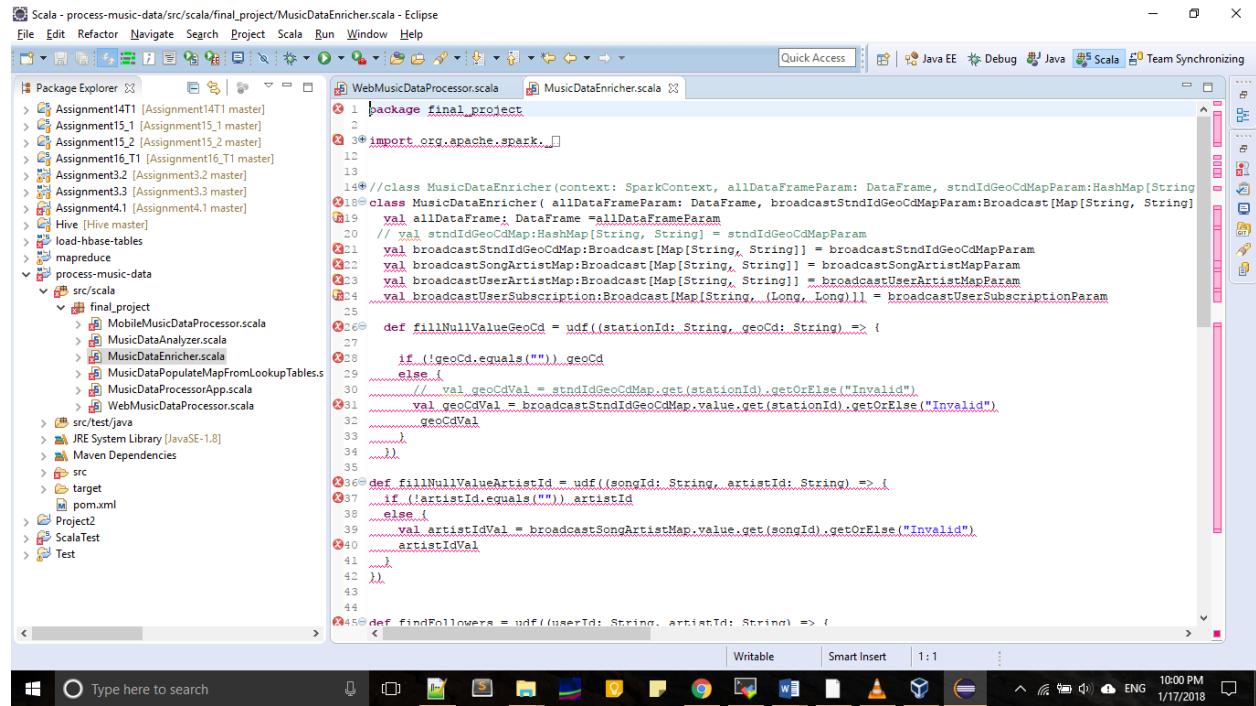
```
newDataFrame("modified_Geo_cd"), newDataFrame("Timestamp"), newDataFrame("Start_ts"),
newDataFrame("End_ts")))

return newDataFrame

}

}
```

Screenshot is as below:



Scala - process-music-data/src/scala/final_project/MusicDataEnricher.scala - Eclipse

File Edit Refactor Navigate Search Project Scala Run Window Help

Quick Access Java EE Debug Java Scala Team Synchronizing

Package Explorer

```

44 def findFollowers = udf((userId: String, artistId: String) => {
45   val artistList = broadcastUserArtistMap.value.get(userId).getOrElse("")
46   if(artistList.equals("")) "0"
47   else {
48     var artistArray = artistList.split(",")
49     if(artistArray.contains(artistId)) "1"
50     else "0"
51   }
52 }
53 )
54
55 })
56
57 def findSubscribers = udf((userId: String, startTs: Long) => {
58
59   val subscriptionTuple= broadcastUserSubscription.value.get(userId).getOrElse((0L, 0L))
60   if(subscriptionTuple._1 == 0 & subscriptionTuple._2 == 0) "0"
61   else if (startTs >= subscriptionTuple._1 && startTs <= subscriptionTuple._2) "1"
62   else "0"
63 }
64 )
65
66 def validateRecords = udf((userId: String, songId: String, modifiedArtistId: String, modifiedGeoCd: String, times: Int) => {
67   if(userId.equals("")) "0"
68   else if (songId.equals("")) "0"
69   else if (modifiedArtistId.equals("Invalid")) "0"
70   else if (modifiedGeoCd.equals("Invalid")) "0"
71   else if (timestamp == null) "0"
72   else if (start_ts == null) "0"
73   else if (end_ts < start_ts) "0"
74   else "1"
75 }
76
77
78
79 def enrichData():DataFrame = {
80
81   var newDataFrame = allDataFrame.withColumn("modified_Geo_cd", fillNullValueGeoCd(allDataFrame("Station_id"), allDataFrame("Song_id")))
82   newDataFrame = newDataFrame.withColumn("modified_Artist_id", fillNullValueArtistId(newDataFrame("Songs_id"), newDataFrame("Artist_id")))
83   newDataFrame = newDataFrame.withColumn("follower", findFollowers(newDataFrame("User_id"), newDataFrame("modified_Geo_cd")))
84   newDataFrame = newDataFrame.withColumn("subscribed", findSubscribers(newDataFrame("User_id"), newDataFrame("modified_Geo_cd")))
85   newDataFrame = newDataFrame.withColumn("isValid", validateRecords (newDataFrame("User_id"), newDataFrame("Songs_id")))
86
87   return newDataFrame
88 }
89
90 }

```

Writable Smart Insert 1:1

Type here to search

Scala - process-music-data/src/scala/final_project/MusicDataEnricher.scala - Eclipse

File Edit Refactor Navigate Search Project Scala Run Window Help

Quick Access Java EE Debug Java Scala Team Synchronizing

Package Explorer

```

57 def findSubscribers = udf((userId: String, startTs: Long) => {
58
59   val subscriptionTuple= broadcastUserSubscription.value.get(userId).getOrElse((0L, 0L))
60   if(subscriptionTuple._1 == 0 & subscriptionTuple._2 == 0) "0"
61   else if (startTs >= subscriptionTuple._1 && startTs <= subscriptionTuple._2) "1"
62   else "0"
63 }
64 )
65
66 def validateRecords = udf((userId: String, songId: String, modifiedArtistId: String, modifiedGeoCd: String, times: Int) => {
67   if(userId.equals("")) "0"
68   else if (songId.equals("")) "0"
69   else if (modifiedArtistId.equals("Invalid")) "0"
70   else if (modifiedGeoCd.equals("Invalid")) "0"
71   else if (timestamp == null) "0"
72   else if (start_ts == null) "0"
73   else if (end_ts < start_ts) "0"
74   else "1"
75 }
76
77
78
79 def enrichData():DataFrame = {
80
81   var newDataFrame = allDataFrame.withColumn("modified_Geo_cd", fillNullValueGeoCd(allDataFrame("Station_id"), allDataFrame("Song_id")))
82   newDataFrame = newDataFrame.withColumn("modified_Artist_id", fillNullValueArtistId(newDataFrame("Songs_id"), newDataFrame("Artist_id")))
83   newDataFrame = newDataFrame.withColumn("follower", findFollowers(newDataFrame("User_id"), newDataFrame("modified_Geo_cd")))
84   newDataFrame = newDataFrame.withColumn("subscribed", findSubscribers(newDataFrame("User_id"), newDataFrame("modified_Geo_cd")))
85   newDataFrame = newDataFrame.withColumn("isValid", validateRecords (newDataFrame("User_id"), newDataFrame("Songs_id")))
86
87   return newDataFrame
88 }
89
90 }

```

Writable Smart Insert 1:1

Type here to search

MODULE4: Analyze the data

- Analyze the data using defining MusicDataAnalyzer class, which will execute sql queries and the result will be stored as reports in HDFS

```
// Define the package final_project and import all the dependent packages
```

```
package final_project

import org.apache.log4j.{ Level, LogManager, PropertyConfigurator }

import org.apache.spark.sql.AnalysisException

import org.apache.spark._

import org.apache.spark.sql.SQLContext

import org.apache.spark.sql.DataFrame

import scala.collection.mutable.HashMap
```

```
// Define the class MusicDataAnalyzer with parameters SparkContext, SQLContext and DataFrame
```

```
class MusicDataAnalyzer(context: SparkContext, sqc: SQLContext, musicDataDFParam: DataFrame)
extends Serializable {
```

```
    val musicDataDF: DataFrame = musicDataDFParam

    val sc = context

    val sqlContext = sqc
```

```
// Define logger and reportBasePath and currentTimestamp
```

```
    val log = LogManager.getRootLogger

    log.setLevel(Level.INFO)

    val reportbasePath = "/user/acadgild/project_music_data_analysis/reports/"

    val currentTimestamp = System.currentTimeMillis().toString
```

```
// Define method analyze which will in run call all the methods for analysis
```

```
// While calling method, each one is put in try catch block so that failing one does not impact others

def analyze() = {
    import sqlContext.implicits._

    // Define temporary table MusicDataDetailed on the dataframe
    musicDataDF.registerTempTable("MusicDataDetailed")
    log.info("Before calling getAllRecords()")

    try {
        getAllRecords()
    } catch {
        case e: Exception => log.error("Exception got while calling getAllRecords: " + e)
    }
    log.info("Before calling getTop10Stations()")

    try {
        getTop10Stations()
    } catch {
        case e: Exception => log.error("Exception got while calling getTop10Stations: " + e)
    }
    log.info("Before calling getMusicDuritionByUser()")

    try {
        getMusicDuritionByUserType()
    } catch {
        case e: Exception => log.error("Exception got while calling getTop10Stations: " + e)
    }
    log.info("Before calling getTo10ConnectedArtists()")
}
```

```
try {
    getTop10ConnectedArtists()
} catch {
    case e: Exception => log.error("Exception got while calling getTop10ConnectedArtists: " + e)
}

log.info("Before calling getTop10UnsubscribedUsers()")

try {
    getTop10UnsubscribedUsers()
} catch {
    case e: Exception => log.error("Exception got while calling getTop10UnsubscribedUsers(): " + e)
}

log.info("Before calling getTo10SongsHavingMaximumRevenue()")

try {
    getTop10SongsHavingMaximumRevenue()
} catch {
    case e: Exception => log.error("Exception got while calling
getTop10SongsHavingMaximumRevenue(): " + e)
}

}

// Method getAllRecords will select all the records from MusicDataDetailed

def getAllRecords() {
```

```

log.info(" Get All records")

// Run the query to get the result

val df = sqlContext.sql("SELECT * FROM MusicDataDetailed ")

df.show()

// Store the result as a single file in HDFS with reportPath and report name MusicDataAllRecords
// concatenated with current timestamp

val df1 = df.repartition(1)

df1.write.format("com.databricks.spark.csv").option("header", "true").save(reportBasePath +
"/MusicDataAllRecords_" + currentTimestamp)

}

// The method getTop10Stations will return top 10 stations were maximum songs are played which are
// liked by unique user

def getTop10Stations() {

    log.info(" Top 10 Music Stations where maximum numbers of songs played which are liked by unique
users")

    // Execute query to get Station_id User_id and count of music played Group by Station_id, User_id and
    // Likes is 1 and isValid is 1

    sqlContext.sql("SELECT Station_id, User_id, count(*) AS music_count FROM MusicDataDetailed "
        + " WHERE Likes='1' AND isValid='1' GROUP BY Station_id, User_id")
        .registerTempTable("MusicCountByStation")

    // Execute query to get unique unique count for user liked, so music_count is greater than 1, it is
    // considered 1

```

```

sqlContext.sql("SELECT Station_id, User_id, CASE WHEN music_count> 1 THEN 1 ELSE music_count
END "
+ " AS unique_music_count FROM MusicCountByStation")
.registerTempTable("UniqueMusicCountByStation")

// Using sum method of SQL, aggregate the total music count group by Station_id and order
// total_music_count desceding. Take first 10 records

val df = sqlContext.sql("SELECT Station_id, sum(unique_music_count) AS total_music_count "
+ " FROM UniqueMusicCountByStation GROUP BY Station_id ORDER BY total_music_count "
+ " DESC LIMIT 10 ")
df.show()

// Store the query output to HDFS as a csv report Top10Stations concatenated with timestamp in
// reportBasePath
val df1 = df.repartition(1)
df1.write.format("com.databricks.spark.csv").option("header", "true").save(reportbasePath +
"/Top10Stations_" + currentTimestamp)

}

// The method getMusicDuritionByUserType returns total lke music by each category of user Subscribed
// or Unsubscribed
def getMusicDuritionByUserType() {
  log.info(" Total duration of Songs played by Subscibed and Unsubscribed Users")

  sqlContext.sql("SELECT CASE WHEN subscribed='1' THEN 'Subscribed' ELSE 'Unsubscribed' END AS
User_type, (End_ts -Start_ts) AS duration "
+ " FROM MusicDataDetailed WHERE isValid='1'"")
.registerTempTable("UserTypeDuration")

```

```

    val df = sqlContext.sql("SELECT User_type, SUM(duration) AS total_duration_milliseconds FROM
UserTypeDuration "
    + " GROUP BY User_type ORDER BY total_duration_milliseconds DESC")
df.show()
val df1 = df.repartition(1)

df1.write.format("com.databricks.spark.csv").option("header", "true").save(reportBasePath +
"/MusicDurationByUserType_" + currentTimestamp)

}

// The method getTop10ConnectedArtists return top 10 connected artists who are followed by user
def getTop10ConnectedArtists() {
    log.info(" Top 10 Connected Artists")
    sqlContext.sql("SELECT Artist_id, User_id, count(*) AS music_count FROM MusicDataDetailed "
    + " WHERE follower='1' AND isValid='1' GROUP BY Artist_id, User_id")
    .registerTempTable("MusicCountByArtist")
    sqlContext.sql("SELECT Artist_id, User_id, CASE WHEN music_count> 1 THEN 1 ELSE music_count END
"
    + " AS unique_music_count FROM MusicCountByArtist")
    .registerTempTable("UniqueMusicCountByUser")
    val df = sqlContext.sql("SELECT Artist_id, sum(unique_music_count) AS total_music_count "
    + " FROM UniqueMusicCountByUser GROUP BY Artist_id ORDER BY total_music_count "
    + " DESC LIMIT 10 ")
    df.show()
    val df1 = df.repartition(1)
    df1.write.format("com.databricks.spark.csv").option("header", "true").save(reportBasePath +
"/Top10ConnectedArtists_" + currentTimestamp)

```

```

}

// The method getTop10SongsHavingMaximumRevenue returns top 10 songs having maximum royalty r
//revenue

def getTop10SongsHavingMaximumRevenue() {
    log.info(" Top 10 Songs Having maximum revenue")

    sqlContext.sql("SELECT Songs_id, CASE WHEN End_ts is NOT NULL AND Start_ts is NOT NULL and
End_ts > Start_ts THEN End_ts - Start_ts ELSE 0 END AS duration FROM MusicDataDetailed "
        + " WHERE (Likes='1' OR Song_end_type = '0') AND isValid='1' ")
    .registerTempTable("SongDuration")

    val df = sqlContext.sql("SELECT Songs_id, SUM(duration) AS total_duration_milliseconds FROM
SongDuration "
        + " GROUP BY Songs_id ORDER BY total_duration_milliseconds DESC LIMIT 10")
    df.show()

    val df1 = df.repartition(1)
    df1.write.format("com.databricks.spark.csv").option("header", "true").save(reportBasePath +
"/Top10SongsHavignMaximumRevenue_" + currentTimestamp)

}

```

```

// The method getTop10UnsubscribedUsers will retrun top 10 unsubscribed users

def getTop10UnsubscribedUsers() {
    log.info(" Top 10 Unsubscribed Users")

    sqlContext.sql("SELECT User_id,CASE WHEN End_ts is NOT NULL AND Start_ts is NOT NULL and End_ts
> Start_ts THEN End_ts - Start_ts ELSE 0 END AS duration FROM MusicDataDetailed "
        + " WHERE subscribed='0' AND isValid='1'"")
    .registerTempTable("SongDuration")

```

```

    val df = sqlContext.sql("SELECT User_id, SUM(duration) AS total_duration_milliseconds FROM
SongDuration "
+ " GROUP BY User_id ORDER BY total_duration_milliseconds DESC LIMIT 10")

df.show()

val df1 = df.repartition(1)

df1.write.format("com.databricks.spark.csv").option("header", "true").save(reportBasePath +
"/Top10UnsubscribedUsers_" + currentTimestamp)

}

}

```

```

Scala - process-music-data/src/scala/final_project/MusicDataAnalyzer.scala - Eclipse
File Edit Refactor Navigate Search Project Scala Run Window Help
Quick Access Java EE Debug Java Scala Team Synchronizing
Package Explorer WebMusicDataProcessor.scala MusicDataEnrichers.scala MusicDataAnalyzer.scala
1 package final_project
2 import org.apache.log4j.{ Level, LogManager, PropertyConfigurator }
3
4 class MusicDataAnalyzer(context: SparkContext, sgc: SQLContext, musicDataDFFParam: DataFrame) extends Serializable
5
6   val musicDataDF: DataFrame = musicDataDFFParam
7   val sc = context
8   val sqlContext = sgc
9
10  val log = LogManager.getRootLogger
11  log.setLevel(Level.INFO)
12  val reportBasePath = "/user/acadgild/project_music_data_analysis/reports/"
13  val currentTimestamp = System.currentTimeMillis().toString
14
15  def analyze() = {
16    import sqlContext.implicits._
17    musicDataDF.registerTempTable("MusicDataDetailed")
18    log.info("Before calling getAllRecords()")
19
20    try {
21      getAllRecords()
22    } catch {
23      case e: Exception => log.error("Exception got while calling getAllRecords: " + e)
24    }
25  }
26
27  try {
28    getTop10Stations()
29  } catch {
30    case e: Exception => log.error("Exception got while calling getTop10Stations: " + e)
31  }
32  log.info("Before calling getMusicDurationByUser()")
33
34  try {
35    getMusicDurationByUser()
36  } catch {
37    case e: Exception => log.error("Exception got while calling getMusicDurationByUser: " + e)
38  }
39

```

Scala - process-music-data/src/scala/final_project/MusicDataAnalyzer.scala - Eclipse

```
File Edit Refactor Navigate Search Project Scala Run Window Help
```

Quick Access Java EE Debug Java Scala Team Synchronizing

Package Explorer

```
Assignment14T1 [Assignment14T1 master]
Assignment15_1 [Assignment15_1 master]
Assignment15_2 [Assignment15_2 master]
Assignment16_T1 [Assignment16_T1 master]
Assignment3_2 [Assignment3_2 master]
Assignment3_3 [Assignment3_3 master]
Assignment4_1 [Assignment4_1 master]
Hive [Hive master]
load-base-tables
mapreduce
process-music-data
  src-scala
    final_project
      MobileMusicDataProcessor.scala
      MusicDataAnalyzer.scala
      MusicDataEnricher.scala
      MusicDataPopulateMapFromLookupTables.scala
      MusicDataProcessorApp.scala
      WebMusicDataProcessor.scala
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
  pom.xml
Project2
ScalaTest
Test
```

WebMusicDataProcessor.scala MusicDataAnalyzer.scala

```
39   try {
40     getMusicDurationByUserType()
41   } catch {
42     case e: Exception => log.error("Exception got while calling getTop10Stations: " + e)
43   }
44   log.info("Before calling getTop10ConnectedArtists()")
45
46   try {
47     getTop10ConnectedArtists()
48   } catch {
49     case e: Exception => log.error("Exception got while calling getTop10ConnectedArtists: " + e)
50   }
51
52   log.info("Before calling getTop10UnsubscribedUsers()")
53
54   try {
55     getTop10UnsubscribedUsers()
56   } catch {
57     case e: Exception => log.error("Exception got while calling getTop10UnsubscribedUsers(): " + e)
58   }
59
60   log.info("Before calling getTop10SongsHavingMaximumRevenue()")
61   try {
62     getTop10SongsHavingMaximumRevenue()
63   } catch {
64     case e: Exception => log.error("Exception got while calling getTop10SongsHavingMaximumRevenue(): " + e)
65   }
66
67
68
69   def getAllRecords() {
70     log.info(" Get All records")
71     val df = sqlContext.sql("SELECT * FROM MusicDataDetailed ")
72
73     df.show()
74     val df1 = df.repartition(1)
75     df1.write.format("com.databricks.spark.csv").option("header", "true").save(reportBasePath + "/MusicDataAllRe
76
77
78
79
80   }
81
82
83   def getTop10Stations() {
84     log.info(" Top 10 Music Stations where maximum numbers of songs played which are liked by unique users")
85     sqlContext.sql("SELECT Station_id, User_id, count(*) AS music_count FROM MusicDataDetailed "
86       + " WHERE Likes='1' AND isValid='1' GROUP BY Station_id, User_id")
87     .registerTempTable("MusicCountByStation")
88     sqlContext.sql("SELECT Station_id, User_id, CASE WHEN music_count> 1 THEN 1 ELSE music_count END "
89       + " AS unique_music_count FROM MusicCountByStation")
90     .registerTempTable("UniqueMusicCountByStation")
91     val df = sqlContext.sql("SELECT Station_id, sum(unique_music_count) AS total_music_count "
92       + " FROM UniqueMusicCountByStation GROUP BY Station_id ORDER BY total_music_count "
93       + " DESC LIMIT 10 ")
94     df.show()
95     val df1 = df.repartition(1)
96     df1.write.format("com.databricks.spark.csv").option("header", "true").save(reportBasePath + "/Top10Stations_
97
98   }
99
100  def getMusicDurationByUserType() {
101    log.info(" Total duration of Songs played by Subscribed and Unsubscribed Users")
102  }
```

Writable Smart Insert 1:1

Type here to search

Scala - process-music-data/src/scala/final_project/MusicDataAnalyzer.scala - Eclipse

```
File Edit Refactor Navigate Search Project Scala Run Window Help
```

Quick Access Java EE Debug Java Scala Team Synchronizing

Package Explorer

```
Assignment14T1 [Assignment14T1 master]
Assignment15_1 [Assignment15_1 master]
Assignment15_2 [Assignment15_2 master]
Assignment16_T1 [Assignment16_T1 master]
Assignment3_2 [Assignment3_2 master]
Assignment3_3 [Assignment3_3 master]
Assignment4_1 [Assignment4_1 master]
Hive [Hive master]
load-base-tables
mapreduce
process-music-data
  src-scala
    final_project
      MobileMusicDataProcessor.scala
      MusicDataAnalyzer.scala
      MusicDataEnricher.scala
      MusicDataPopulateMapFromLookupTables.scala
      MusicDataProcessorApp.scala
      WebMusicDataProcessor.scala
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
  pom.xml
Project2
ScalaTest
Test
```

WebMusicDataProcessor.scala MusicDataAnalyzer.scala

```
69
70   def getAllRecords() {
71     log.info(" Get All records")
72     val df = sqlContext.sql("SELECT * FROM MusicDataDetailed ")
73
74     df.show()
75     val df1 = df.repartition(1)
76     df1.write.format("com.databricks.spark.csv").option("header", "true").save(reportBasePath + "/MusicDataAllRe
77
78
79
80   }
81
82
83   def getTop10Stations() {
84     log.info(" Top 10 Music Stations where maximum numbers of songs played which are liked by unique users")
85     sqlContext.sql("SELECT Station_id, User_id, count(*) AS music_count FROM MusicDataDetailed "
86       + " WHERE Likes='1' AND isValid='1' GROUP BY Station_id, User_id")
87     .registerTempTable("MusicCountByStation")
88     sqlContext.sql("SELECT Station_id, User_id, CASE WHEN music_count> 1 THEN 1 ELSE music_count END "
89       + " AS unique_music_count FROM MusicCountByStation")
90     .registerTempTable("UniqueMusicCountByStation")
91     val df = sqlContext.sql("SELECT Station_id, sum(unique_music_count) AS total_music_count "
92       + " FROM UniqueMusicCountByStation GROUP BY Station_id ORDER BY total_music_count "
93       + " DESC LIMIT 10 ")
94     df.show()
95     val df1 = df.repartition(1)
96     df1.write.format("com.databricks.spark.csv").option("header", "true").save(reportBasePath + "/Top10Stations_
97
98   }
99
100  def getMusicDurationByUserType() {
101    log.info(" Total duration of Songs played by Subscribed and Unsubscribed Users")
102  }
```

Writable Smart Insert 1:1

Type here to search

```

99
100 def getMusicDurationByUserType() {
101     log.info(" Total duration of Songs played by Subscribed and Unsubscribed Users").
102     sqlContext.sql("SELECT CASE WHEN subscribed='1' THEN 'Subscribed' ELSE 'Unsubscribed' END AS User_type, (End
103         + " FROM MusicDataDetailed WHERE isValid='1')").registerTempTable("UserTypeDuration")
104     val df = sqlContext.sql("SELECT User_type, SUM(duration) AS total_duration_milliseconds FROM UserTypeDuration
105         + " GROUP BY User_type ORDER BY total_duration_milliseconds DESC")
106     df.show()
107     val dfl = df.repartition()
108
109     dfl.write.format("com.databricks.spark.csv").option("header", "true").save(reportBasePath + "/MusicDurationB
110
111     }
112
113
114
115
116
117 def getTop10ConnectedArtists() {
118     log.info(" Top 10 Connected Artists")
119     sqlContext.sql("SELECT Artist_id, User_id, count(*) AS music_count FROM MusicDataDetailed "
120         + " WHERE follower='1' AND isValid='1' GROUP BY Artist_id, User_id")
121         .registerTempTable("MusicCountByArtist")
122     sqlContext.sql("SELECT Artist_id, User_id, CASE WHEN music_count > 1 THEN 1 ELSE music_count END "
123         + " AS unique_music_count FROM MusicCountByArtist")
124         .registerTempTable("UniqueMusicCountByUser")
125     val df = sqlContext.sql("SELECT Artist_id, sum(unique_music_count) AS total_music_count "
126         + " FROM UniqueMusicCountByUser GROUP BY Artist_id ORDER BY total_music_count "
127         + " DESC LIMIT 10")
128     df.show()
129     val dfl = df.repartition()
130     dfl.write.format("com.databricks.spark.csv").option("header", "true").save(reportBasePath + "/Top10Connected
131
132

```

```

133
134
135 def getTop10SongsHavingMaximumRevenue() {
136     log.info(" Top 10 Songs Having maximum revenue")
137
138     sqlContext.sql("SELECT Songs_id, CASE WHEN End_ts IS NOT NULL AND Start_ts IS NOT NULL AND End_ts > Start_ts
139         + " WHERE Likes='1' OR Song_end_type = '0' AND isValid='1' ")
140         .registerTempTable("SongDuration")
141     val df = sqlContext.sql("SELECT Songs_id, SUM(duration) AS total_duration_milliseconds FROM SongDuration "
142         + " GROUP BY Songs_id ORDER BY total_duration_milliseconds DESC LIMIT 10")
143     df.show()
144     val dfl = df.repartition()
145     dfl.write.format("com.databricks.spark.csv").option("header", "true").save(reportBasePath + "/Top10SongsHavi
146
147
148
149
150 def getTop10UnsubscribedUsers() {
151     log.info(" Top 10 Unsubscribed Users")
152
153     sqlContext.sql("SELECT User_id, CASE WHEN End_ts IS NOT NULL AND Start_ts IS NOT NULL AND End_ts > Start_ts T
154         + " WHERE subscribed='0' AND isValid='1' ")
155         .registerTempTable("SongDuration")
156     val df = sqlContext.sql("SELECT User_id, SUM(duration) AS total_duration_milliseconds FROM SongDuration "
157         + " GROUP BY User_id ORDER BY total_duration_milliseconds DESC LIMIT 10")
158
159     df.show()
160     val dfl = df.repartition()
161     dfl.write.format("com.databricks.spark.csv").option("header", "true").save(reportBasePath + "/Top10Unsubscri
162
163
164
165
166

```

Module5: MusicDataPopulateMapFromLookupTables : Load the maps from HBase tables

To load the HBase table, I used the class MusicDataPopulateMapFromLookupTables

```

// Import the dependent packages
package final_project

import org.apache.spark._
import org.apache.spark.sql.SQLContext
import org.apache.hadoop.hbase.util.Bytes
import org.apache.hadoop.hbase.client.{Put, HTable}
import org.apache.hadoop.hbase.HBaseConfiguration
import org.apache.hadoop.hbase.HTableDescriptor
import org.apache.hadoop.hbase.HColumnDescriptor
import org.apache.hadoop.hbase.client.HBaseAdmin
import org.apache.hadoop.hbase.client.HTable
import org.apache.hadoop.hbase.mapreduce.TableInputFormat
import org.apache.log4j.{ Level, LogManager, PropertyConfigurator }

// Create class MusicDataPopulateMapFromLookupTables with parameters
class MusicDataPopulateMapFromLookupTables(context: SparkContext) {

    val sc:SparkContext = context
    val hconf = HBaseConfiguration.create
    val admin = new HBaseAdmin(hconf)
    val log = LogManager.getRootLogger
    log.setLevel(Level.INFO)

    // Create map from HBase table with key and value as column value
    def getCommonLookupMap(tablename:String) = {
        if (!admin.isTableAvailable(tablename)) {
            log.warn("HBase Table " + tablename + " does not exists")
        } else {
            log.info("Table " + tablename + " exists")
        }
    }
}

```

```

    val hbaseRDD = sc.newAPIHadoopRDD(hconf, classOf[TableInputFormat],
    classOf[org.apache.hadoop.hbase.io.ImmutableBytesWritable],
    classOf[org.apache.hadoop.hbase.client.Result])

    val resultRDD = hbaseRDD.map(tuple=>tuple._2)

    val keyValueRDD = resultRDD.map(result => (Bytes.toString(result.getRow()).split(" ")(0),
    Bytes.toString(result.value)))

    val map = keyValueRDD.collectAsMap
    scala.collection.immutable.Map(map.toSeq:_*)

}

// Populate Map with key StationId and GeoCd from table StndIdGeoCd

def getStationIdGeoCdMap() = {

    val tablename = "StndIdGeoCd"

    hconf.set(TableInputFormat.INPUT_TABLE, tablename)
    getCommonLookupMap(tablename)

}

// Populate Map with key songId and Artist from table SongArtist

def getSongArtistMap() = {

    val tablename = "SongArtist"

    hconf.set(TableInputFormat.INPUT_TABLE, tablename)
    getCommonLookupMap(tablename)

}

// Populate Map with key userId and Artist List from table UserArtist

```

```

def getUserArtistMap() = {
    val tablename = "UserArtist"
    hconf.set(TableInputFormat.INPUT_TABLE, tablename)
    getCommonLookupMap(tablename)
}

// Populate map with Key as UserId and Value as tuple (start_ts, end_ts)

def getUserSubscriptionMap():scala.collection.immutable.Map[String, (Long, Long)] = {
    val tablename = "UserSubscription"
    hconf.set(TableInputFormat.INPUT_TABLE, tablename)
    if (!admin.isTableAvailable(tablename)) {
        log.warn("HBase Table " + tablename + " does not exists")
    } else {
        log.info("Table " + tablename + " exists")
    }
}

val hbaseRDD = sc.newAPIHadoopRDD(hconf, classOf[TableInputFormat],
    classOf[org.apache.hadoop.hbase.io.ImmutableBytesWritable],
    classOf[org.apache.hadoop.hbase.client.Result])

val resultRDD = hbaseRDD.map(tuple=>tuple._2)

val keyValueRDD = resultRDD.map(result => (Bytes.toString(result.getRow()).split(" ")(0), (
    if (Bytes.toString(result.getValue(new String("Subscription").getBytes(), new
    String("Start_ts").getBytes())).equals("")) 0 else Bytes.toString(result.getValue(new
    String("Subscription").getBytes(), new String("Start_ts").getBytes())).toLong,
    if (Bytes.toString(result.getValue(new String("Subscription").getBytes(), new
    String("End_ts").getBytes())).equals("")) 0 else Bytes.toString(result.getValue(new
    String("Subscription").getBytes(), new String("End_ts").getBytes())).toLong
))

```

```
)))
```

```
val map = keyValueRDD.collectAsMap  
return scala.collection.immutable.Map(map.toSeq:_*)
```

```
}
```

```
}
```

Screenshot is as below:

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Scala - process-music-data/src/scala/final_project/MusicDataPopulateMapFromLookupTables.scala - Eclipse
- Menu Bar:** File Edit Refactor Navigate Search Project Scala Run Window Help
- Toolbar:** Standard Eclipse toolbar items.
- Editor Area:** Displays the code for `MusicDataPopulateMapFromLookupTables.scala`. The code is as follows:

```
1 package final_project
2
3 import org.apache.spark._
4
5 class MusicDataPopulateMapFromLookupTables(context: SparkContext) {
6   val sc:SparkContext = context
7   val hconf = HBaseConfiguration.create()
8   val admin = new HBaseAdmin(hconf)
9   val log = LogManager.getLogger
10  log.setLevel(Level.INFO)
11
12  def getCommonLookupMap(tablename:String) = {
13    if (!admin.isTableAvailable(tablename)) {
14      log.warn("HBase Table " + tablename + " does not exists")
15    } else {
16      log.info("Table " + tablename + " exists")
17    }
18
19    val hbaseRDD = sc.newAPIHadoopRDD(hconf, classOf[TableInputFormat], classOf[org.apache.hadoop.hbase.io.ImmutableBytesWritable])
20    val resultRDD = hbaseRDD.map(tuple=>tuple._2)
21    val keyValueRDD = resultRDD.map(result => (Bytes.toString(result.getRow()).split("\t").(0), Bytes.toString(result.getFamily())))
22
23    val map = keyValueRDD.collectAsMap
24    scala.collection.immutable.Map(map.toSeq:_*)
25
26  }
27
28  def getStationIdGeoCdMap() = {
29    val tablename = "StndIdGeoCd"
30    hconf.set(TableInputFormat.INPUT_TABLE, tablename)
31    getCommonLookupMap(tablename)
32
33}
```

- Package Explorer:** Shows the project structure with files like Assignment14T1, Assignment15_1, Assignment15_2, Assignment16_1, Assignment3_2, Assignment3_3, Assignment4_1, Hive, load-hbase-tables, mapreduce, process-music-data, and various Scala files.
- Bottom Status Bar:** Shows the search bar, system icons, and the date/time (10:04 PM 1/17/2018).

Scala - process-music-data/src/scala/final_project/MusicDataPopulateMapFromLookupTables.scala - Eclipse

```

39
40 def getStationIdGeoCdm() = {
41   val tablename = "StndIdGeoCdm"
42   hconf.set(TableInputFormat.INPUT_TABLE, tablename)
43   getCommonLookupMap(tablename)
44
45 }
46
47
48 def getSongsArtistMap() = {
49   val tablename = "SongArtist"
50   hconf.set(TableInputFormat.INPUT_TABLE, tablename)
51   getCommonLookupMap(tablename)
52 }
53
54
55 def getUserArtistMap() = {
56   val tablename = "UserArtist"
57   hconf.set(TableInputFormat.INPUT_TABLE, tablename)
58   getCommonLookupMap(tablename)
59 }
60
61
62 def getUserSubscriptionMap():scala.collection.immutable.Map[String,(Long, Long)] = {
63   val tablename = "UserSubscription"
64   hconf.set(TableInputFormat.INPUT_TABLE, tablename)
65   if (!Admin.isTableAvailable(tablename)) {
66     log.warn("HBase Table " + tablename + " does not exists")
67   } else {
68     log.info("Table " + tablename + " exists")
69   }
70
71   val hbaseRDD = sc.newAPIHadoopRDD(hconf, classOf[TableInputFormat], classOf[org.apache.hadoop.hbase.io.Immuta
72

```

Scala - process-music-data/src/scala/final_project/MusicDataPopulateMapFromLookupTables.scala - Eclipse

```

57   hconf.set(TableInputFormat.INPUT_TABLE, tablename)
58   getCommonLookupMap(tablename)
59
60 }
61
62 def getUserSubscriptionMap():scala.collection.immutable.Map[String,(Long, Long)] = {
63   val tablename = "UserSubscription"
64   hconf.set(TableInputFormat.INPUT_TABLE, tablename)
65   if (!Admin.isTableAvailable(tablename)) {
66     log.warn("HBase Table " + tablename + " does not exists")
67   } else {
68     log.info("Table " + tablename + " exists")
69   }
70
71   val hbaseRDD = sc.newAPIHadoopRDD(hconf, classOf[TableInputFormat], classOf[org.apache.hadoop.hbase.io.Immuta
72
73   val resultRDD = hbaseRDD.map(tuple=>tuple._2)
74
75
76   val keyValueRDD = resultRDD.map(result => (Bytes.toString(result.getRow()).split(" "), (
77     if (Bytes.toString(result.getValue(new String("Subscription")).getBytes(), new String("Start_ts")).getByt
78     if (Bytes.toString(result.getValue(new String("Subscription")).getBytes(), new String("End_ts")).getByt
79   )))
80
81   val map = keyValueRDD.collectAsMap
82   return scala.collection.immutable.Map(map.toSeq:_*)
83
84
85 }
86
87
88 }
89

```

MODULE6: Main object MusicDataProcessorApp

```

// Import all the dependent packages

package final_project

import org.apache.log4j.{ Level, LogManager, PropertyConfigurator }

import org.apache.spark._

import org.apache.spark.sql.SQLContext

import org.apache.spark.sql.DataFrame

import scala.collection.mutable.HashMap

// define object MusicDataProcessorApp

object MusicDataProcessorApp {

    // define method main

    def main(args: Array[String]) {

        val web_file_path = "/data/web/file-1.xml"

        val mobile_file_path = "file:///data/mob/file.txt"

        val log = LogManager.getRootLogger

        log.setLevel(Level.INFO)

    }

    // Define all the contexts

    val conf = new SparkConf().setAppName("MusicDataProcessApp").setMaster("local[2]")

    val sc = new SparkContext(conf)

    val sqlContext = new SQLContext(sc)

    import sqlContext.implicits._

    sc.setLogLevel("ERROR")

    // Populate the map from StationId GeoCd and create a broadcast object
}

```

```

val populateMusicDataMap = new MusicDataPopulateMapFromLookupTables(sc)

val stndIdGeoCdMap:Map[String, String] = populateMusicDataMap.getStationIdGeoCdMap()

val broadcastStndIdGeoCdMap = sc.broadcast(stndIdGeoCdMap)

// Populate the map songArtistMap from songId Aritist and create a broadcast object

val songArtistMap = populateMusicDataMap.getSongArtistMap()

val broadcastSongArtistMap = sc.broadcast(songArtistMap)

// Populate the map userArtistMap from userId and Aritist and create a broadcast object

val userArtistMap = populateMusicDataMap.getUserArtistMap()

val broadcastUserArtistMap = sc.broadcast(userArtistMap)

// Populate the map userSubscription from userId and subscription tuple (start_ts, end_ts) and
// create a broadcast object

val userSubscription:Map[String, (Long, Long)] = populateMusicDataMap.getUserSubscriptionMap()

val broadcastuserSubscription = sc.broadcast(userSubscription)

// Call the processData method on WebMusicDataProcessor and get the dataframe webDataFrame

val webMusicDataProcessor = new WebMusicDataProcessor(web_file_path, sc, sqlContext )

log.info("Before calling webMusicDataProcessor.processData()")

val webDataFrame:DataFrame = webMusicDataProcessor.processData()

log.info("After calling webMusicDataProcessor.processData()")

// Call the processData method on MobileMusicDataProcessor and get the dataframe mobileDataFrame

val mobileMusicDataProcessor = new MobileMusicDataProcessor(mobile_file_path, sc, sqlContext)

log.info("Before calling mobileMusicDataProcessor.processData()")

```

```

val mobileDataFrame:DataFrame = mobileMusicDataProcessor.processData()
log.info("After calling webMusicDataProcessor.processData()")

// Combine webDataFrame and mobileDataFrame into one allDataFrame
val allDataFrame:DataFrame = webDataFrame.unionAll(mobileDataFrame)
log.info("allDataFrame count =" + allDataFrame.count)

// Call the enrichData method on MusicDataEnricher and enhance the data
val musicDataEnricher = new MusicDataEnricher(allDataFrame, broadcastStndIdGeoCdMap,
broadcastSongArtistMap, broadcastUserArtistMap, broadcastUserSubscription)

val enrichedAllDataFrame:DataFrame = musicDataEnricher.enrichData()
log.info("Showing dataframe after enrichment")
enrichedAllDataFrame.show(100)
enrichedAllDataFrame.registerTempTable("DetailedMusicData")

// Call the analyze method on musicDataAnalyzer
val musicDataAnalyzer = new MusicDataAnalyzer(sc, sqlContext, enrichedAllDataFrame)
musicDataAnalyzer.analyze()

}

}

```

COMPILATION: pom.xml file is as below:

```

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

```

```
<groupId>final_project</groupId>
<artifactId>process_music_data</artifactId>
<version>0.0.1-SNAPSHOT</version>
<properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
    <encoding>UTF-8</encoding>
    <scala.version>2.10.5</scala.version>
</properties>
<build>
    <sourceDirectory>src/main/scala</sourceDirectory>
    <resources>
        <resource>
            <directory>src/main/scala</directory>
            <excludes>
                <exclude>**/*.java</exclude>
            </excludes>
        </resource>
    </resources>
    <plugins>
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.5.1</version>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
            </configuration>
        </plugin>
        <plugin>
```

```
<groupId>org.scala-tools</groupId>
<artifactId>maven-scala-plugin</artifactId>
<version>2.15.2</version>
<executions>
  <execution>
    <goals>
      <goal>compile</goal>
    </goals>
  </execution>
</executions>
</plugin>
</plugins>
</build>
<dependencies>
  <!-- https://mvnrepository.com/artifact/org.apache.spark/spark-core_2.11 -->
  <dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-core_2.10</artifactId>
    <version>1.6.0</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.apache.spark/spark-sql_2.11 -->
  <dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-sql_2.10</artifactId>
    <version>1.6.0</version>
  </dependency>
  <dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-hive_2.10</artifactId>
```

```
<version>1.6.0</version>
<scope>provided</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.hbase/hbase-client -->
<dependency>
    <groupId>org.apache.hbase</groupId>
    <artifactId>hbase-client</artifactId>
    <version>0.98.14-hadoop2</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.hbase/hbase-common -->
<dependency>
    <groupId>org.apache.hbase</groupId>
    <artifactId>hbase-common</artifactId>
    <version>0.98.14-hadoop2</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.hbase/hbase-protocol -->
<dependency>
    <groupId>org.apache.hbase</groupId>
    <artifactId>hbase-protocol</artifactId>
    <version>0.98.14-hadoop2</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.hbase/hbase-hadoop2-compat -->
<dependency>
    <groupId>org.apache.hbase</groupId>
    <artifactId>hbase-hadoop2-compat</artifactId>
    <version>0.98.14-hadoop2</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.hbase/hbase-annotations -->
```

```

<dependency>
    <groupId>org.apache.hbase</groupId>
    <artifactId>hbase-annotations</artifactId>
    <version>0.98.14-hadoop2</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.hbase/hbase-server -->

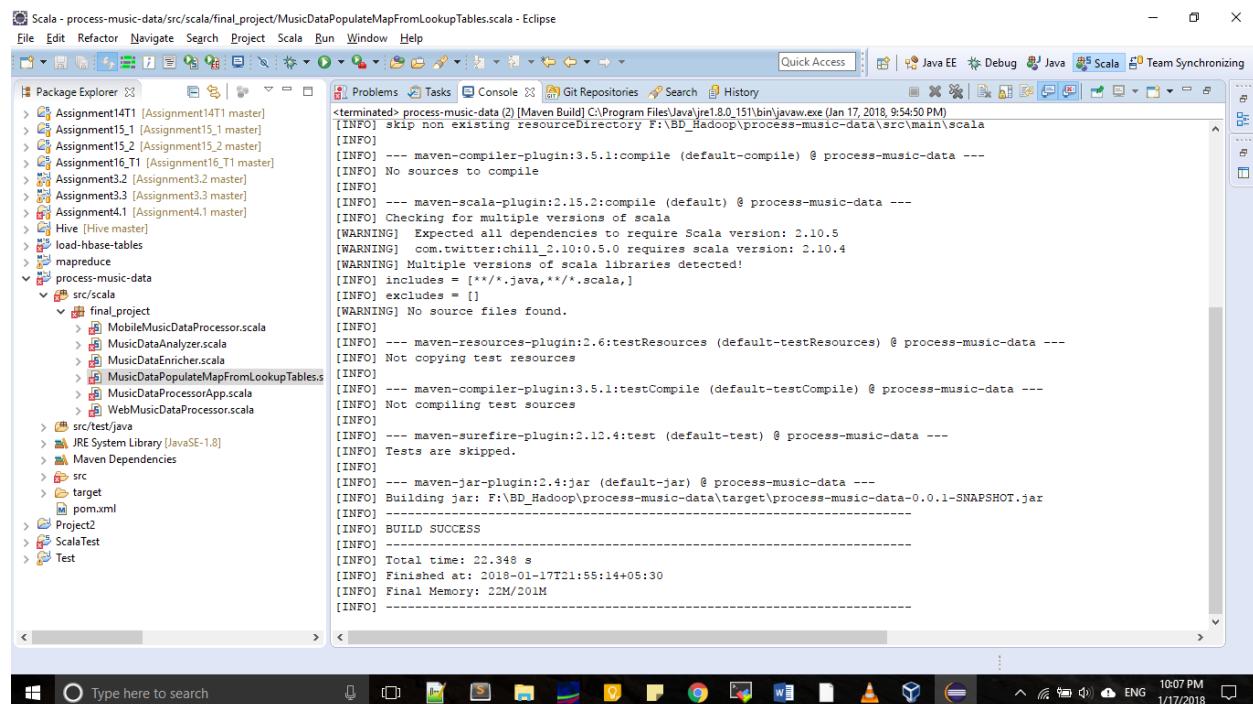
<dependency>
    <groupId>org.apache.hbase</groupId>
    <artifactId>hbase-server</artifactId>
    <version>0.98.14-hadoop2</version>
</dependency>

</dependencies>

</project>

```

Compilation using Run -> Maven Install, screenshot is as below:



OUTPUT:

All the folders are in HDFS are as below:

All Records – This will contain all the records in MusicDataDetailed

The screenshot shows a terminal window titled "acadgild@localhost:~". The window contains several tabs, with the active tab displaying the following command and its output:

```
lRecords 1515000420756/part-00000
[acadgild@localhost ~]$ clear
[acadgild@localhost ~]$ hadoop fs -cat /user/acadgild/project_music_data_analysis/reports/MusicDataAllRecords_1515000420756/part-00000
18/01/04 00:39:36 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
User_id,Songs_id,Artist_id,Timestamp,Start_ts,End_ts,Geo_cd,Station_id,Song_end_type,Likes,Dislikes,modified_Geo_cd,modified_Artist_id,follower_subscribed,isValid
U106,S205,A300,1462863262000,1462863262000,1494297562000,AP,ST407,2,1,1,AP,A300,0,0,1
U114,S209,A303,1465490556000,1462863262000,1494297562000,U,ST411,2,1,0,U,A303,0,0,1
U113,S203,A304,1465490556000,1465490556000,1462863262000,U,ST405,0,0,1,U,A304,0,0,0
U108,S200,A302,1468094889000,1462863262000,1468094889000,U,ST414,0,0,1,U,A302,0,0,1
U102,S203,A305,1465490556000,1465490556000,1494297562000,U,ST404,2,0,0,U,A305,0,0,1
",S208,A300,1465490556000,1494297562000,1465490556000,U,ST411,1,0,1,U,A300,0,0,0
U115,S200,A300,1465490556000,1494297562000,1465490556000,AU,ST404,3,0,0,AU,A300,0,0,0
U111,S204,A300,1465490556000,1465490556000,1468094889000,U,ST410,3,1,1,U,A300,0,0,1
U120,S201,A300,1494297562000,1465490556000,1468094889000,,ST410,3,0,1,A,A300,0,0,1
U113,S203,,1465490556000,1465490556000,1465490556000,A,ST402,1,1,0,A,A303,0,0,1
U109,S203,A304,1462863262000,1494297562000,1468094889000,E,ST405,1,1,1,E,A304,0,0,0
U110,S202,A303,1494297562000,1494297562000,1468094889000,AU,ST402,2,1,0,AU,A303,0,0,0
U100,S200,A301,1494297562000,1494297562000,1494297562000,AP,ST410,3,1,1,AP,A301,1,0,1
U101,S208,A300,1462863262000,1468094889000,1462863262000,E,ST408,0,1,1,E,A300,0,0,0
U106,S206,A300,1494297562000,1465490556000,1462863262000,A,ST405,3,1,0,A,A300,0,0,0
U107,S202,A304,1494297562000,1468094889000,1462863262000,U,ST409,0,0,0,U,A304,0,0,0
U103,S204,A300,1468094889000,1494297562000,1465490556000,AU,ST411,2,1,0,AU,A300,0,0,0
U103,S202,A300,1465490556000,1465490556000,1465490556000,A,ST415,2,1,1,A,A300,0,0,1
U113,S203,A303,1462863262000,1468094889000,1494297562000,U,ST408,2,0,0,U,A303,0,0,1
U113,S204,A301,1494297562000,1494297562000,1465490556000,E,ST415,3,0,1,E,A301,0,0,0
U114,S207,A303,1465130523,1465230523,1475130523,A,ST415,3,1,0,A,A303,0,1,1
U107,S202,A303,1495130523,1465230523,1465230523,U,ST415,0,1,1,U,A303,0,0,1
U100,S204,A302,1495130523,1475130523,1465130523,AU,ST408,2,1,1,AU,A302,1,0,0
U104,S202,A303,1465230523,1475130523,1465130523,A,ST409,2,0,1,A,A303,0,1,0
U102,S207,A301,1465230523,1485130523,1465230523,AU,ST403,3,1,1,AU,A301,0,0,0
",S203,A302,1495130523,1475130523,1465230523,E,ST400,0,0,1,E,A302,0,0,0
U106,S202,A302,1465230523,1465130523,1465130523,AU,ST408,0,1,1,AU,A302,1,0,1
U105,S207,A300,1465230523,1485130523,1465130523,U,ST400,2,0,1,U,A300,0,0,0
U108,S205,A304,1465130523,1465130523,1475130523,,ST410,2,1,0,A,A304,1,0,1
```

Top10Stations – This will top 10 stations having maximum music played which are liked by unique users

A screenshot of a Linux desktop environment. At the top, there is a panel with icons for Applications, Places, System, and a few others. The date and time are shown as Thu Jan 4, 12:41 AM. The user's name, acadgild, is displayed twice. Below the panel is a terminal window titled "acadgild@localhost:~". The terminal shows the following command and its output:

```
[acadgild@localhost ~]$ hadoop fs -cat /user/acadgild/project_music_data_analysis/reports/Top10Stations_1515000420756/part-0000  
18/01/04 00:41:53 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable  
Station_id,total_music_count  
ST415,5  
ST410,3  
ST402,1  
ST404,1  
ST407,1  
ST408,1  
ST411,1  
[acadgild@localhost ~]$
```

The terminal window has tabs at the bottom labeled "acadgild@localhost:~" (active), "workspace - process...", and "BF69-1207".

DurationByCategory – This will report total duration of music played by each time of user (unsubscribed/unsubscribed)

A screenshot of a Linux desktop environment. At the top, there is a panel with icons for Applications, Places, System, and a few others. The date and time are shown as Thu Jan 4, 12:44 AM. The terminal window title is "acadgild" and the current directory is "acadgild@localhost:~". The terminal window contains the following command and its output:

```
[acadgild@localhost ~]$ hadoop fs -cat /user/acadgild/project_music_data_analysis/reports/MusicDurationByUserType_15150004207/part-00000
18/01/04 00:44:34 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
User type,total duration milliseconds
Unsubscribed,128378672000
Subscribed,9900000
[acadgild@localhost ~]$
```

DurationByCategory – This will report total duration of music played by each time of user (unsubscribed/unsubscribed)

Applications Places System Thu Jan 4, 12:39 AM acadgild

acadgild@localhost:~

File Edit View Search Terminal Tabs Help

acadgild@localhost:/usr/local/s... acadgild@localhost:~/project_... acadgild@localhost:~ acadgild@localhost:~

```
lRecords 1515000420756/part-00000
[acadgild@localhost ~]$ clear
[acadgild@localhost ~]$ hadoop fs -cat /user/acadgild/project_music_data_analysis/reports/MusicDataAllRecords_1515000420756/part-00000
18/01/04 00:39:36 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
User_id,Songs_id,Artist_id,Timestamp,Start_ts,End_ts,Geo_cd,Station_id,Song_end_type,Likes,Dislikes,modified_Geo_cd,modified_Artist_id,follower,subscribed,isValid
U106,S205,A300,1462863262000,1462863262000,1494297562000,AP,ST407,2,1,1,AP,A300,0,0,1
U114,S209,A303,1465490556000,1462863262000,1494297562000,U,ST411,2,1,0,U,A303,0,0,1
U113,S203,A304,1465490556000,1465490556000,1462863262000,U,ST405,0,0,1,U,A304,0,0,0
U108,S200,A302,1468094889000,1462863262000,1468094889000,U,ST414,0,0,1,U,A302,0,0,1
U102,S203,A305,1465490556000,1465490556000,1494297562000,U,ST404,2,0,0,U,A305,0,0,1
",S208,A300,1465490556000,1494297562000,1465490556000,U,ST411,1,0,1,U,A300,0,0,0
U115,S200,A300,1465490556000,1494297562000,1465490556000,AU,ST404,3,0,0,AU,A300,0,0,0
U111,S204,A300,1465490556000,1465490556000,1468094889000,U,ST410,3,1,1,U,A300,0,0,1
U120,S201,A300,1494297562000,1465490556000,1468094889000,,ST410,3,0,1,A,A300,0,0,1
U113,S203,,1465490556000,1465490556000,1465490556000,A,ST402,1,1,0,A,A303,0,0,1
U109,S203,A304,1462863262000,1494297562000,1468094889000,E,ST405,1,1,1,E,A304,0,0,0
U110,S202,A303,1494297562000,1494297562000,1468094889000,AU,ST402,2,1,0,AU,A303,0,0,0
U100,S200,A301,1494297562000,1494297562000,1494297562000,AP,ST410,3,1,1,AP,A301,1,0,1
U101,S208,A300,1462863262000,1468094889000,1462863262000,E,ST408,0,1,1,E,A300,0,0,0
U106,S206,A300,1494297562000,1465490556000,1462863262000,A,ST405,3,1,0,A,A300,0,0,0
U107,S202,A304,1494297562000,1468094889000,1462863262000,U,ST409,0,0,0,U,A304,0,0,0
U103,S204,A300,1468094889000,1494297562000,1465490556000,AU,ST411,2,1,0,AU,A300,0,0,0
U103,S202,A300,1465490556000,1465490556000,1465490556000,A,ST415,2,1,1,A,A300,0,0,1
U113,S203,A303,1462863262000,1468094889000,1494297562000,U,ST408,2,0,0,U,A303,0,0,1
U113,S204,A301,1494297562000,1494297562000,1465490556000,E,ST415,3,0,1,E,A301,0,0,0
U114,S207,A303,1465130523,1465230523,1475130523,A,ST415,3,1,0,A,A303,0,1,1
U107,S202,A303,1495130523,1465230523,1465230523,U,ST415,0,1,1,U,A303,0,0,1
U100,S204,A302,1495130523,1475130523,1465130523,AU,ST408,2,1,1,AU,A302,1,0,0
U104,S202,A303,1465230523,1475130523,1465130523,A,ST409,2,0,1,A,A303,0,1,0
U102,S207,A301,1465230523,1485130523,1465230523,AU,ST403,3,1,1,AU,A301,0,0,0
",S203,A302,1495130523,1475130523,1465230523,E,ST400,0,0,1,E,A302,0,0,0
U106,S202,A302,1465230523,1465130523,1465130523,AU,ST408,0,1,1,AU,A302,1,0,1
U105,S207,A300,1465230523,1485130523,1465130523,U,ST400,2,0,1,U,A300,0,0,0
U108,S205,A304,1465130523,1465130523,1475130523,,ST410,2,1,0,A,A304,1,0,1
```

Top10ConnectedArtists : Top 10 Connected Artists

A screenshot of a Linux desktop environment. At the top, there is a panel with icons for Applications, Places, System, and a few others. The date and time are shown as Thu Jan 4, 12:46 AM. The user's name, acadgild, is displayed twice. Below the panel is a terminal window titled "acadgild@localhost:~". The terminal has tabs labeled "acadgild@localhost:/usr/local/s...", "acadgild@localhost:~/project_...", "acadgild@localhost:~", and "acadgild@localhost:~". The main pane of the terminal shows the following command and its output:

```
[acadgild@localhost ~]$ hadoop fs -cat /user/acadgild/project_music_data_analysis/reports/Top10ConnectedArtists_1515000420756/part-00000
18/01/04 00:46:36 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Artist_id,total_music_count
A302,2
A301,1
A304,1
[acadgild@localhost ~]$
```

Top10SongsHavingMaximumRevenue: Top 10 songs have maximum revenue

Applications Places System Thu Jan 4, 12:52 AM acadgild

acadgild@localhost:~

File Edit View Search Terminal Tabs Help

```
acadgild@localhost:/usr/local/s... acadgild@localhost:~/project_... acadgild@localhost:~ acadgild@localhost:~  
[acadgild@localhost ~]$ hadoop fs -cat /user/acadgild/project_music_data_analysis/reports/Top10SongsHavignMaximumRevenue_1515  
000420756/part-00000  
18/01/04 00:52:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl  
asses where applicable  
Songs_id,total_duration_milliseconds  
S205,31444300000  
S209,31434300000  
S200,5231627000  
S204,2604333000  
S206,20000000  
S203,20000000  
S202,10000000  
S207,9900000  
S210,0  
[acadgild@localhost ~]$
```

acadgild@localhost:~ workspace - process-... BF69-1207

A screenshot of a Linux desktop environment. At the top, there is a panel with icons for Applications, Places, System, and a few others. The date and time are shown as Thu Jan 4, 12:52 AM. The user name acadgild is displayed. Below the panel is a terminal window titled "acadgild@localhost:~". The terminal shows the following command and its output:

```
[acadgild@localhost ~]$ hadoop fs -cat /user/acadgild/project_music_data_analysis/reports/Top10SongsHavignMaximumRevenue_1515000420756/part-00000
18/01/04 00:52:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Songs_id,total_duration_milliseconds
S205,31444300000
S209,31434300000
S200,5231627000
S204,2604333000
S206,20000000
S203,20000000
S202,10000000
S207,9900000
S210,0
[acadgild@localhost ~]$
```

The terminal window has tabs at the bottom labeled "acadgild@localhost:~" (active), "workspace - process...", and "BF69-1207".

Top10UnsubscribedUsers – Top 10 unsubscribed users who listened to the music for maximum duration

Applications Places System Thu Jan 4, 12:54 AM acadgild

acadgild@localhost:~\$ hadoop fs -cat /user/acadgild/project_music_data_analysis/reports/Top10UnsubscribedUsers_1515000420756/part-00000

```
18/01/04 00:54:35 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
User_id,total_duration_milliseconds
U106,31434300000
U114,31434300000
U102,28807006000
U113,26202673000
U108,5241627000
U111,2624333000
U120,2604333000
U110,20000000
U101,10000000
U118,100000
[acadgild@localhost ~]$
```

acadgild@localhost:~ workspace - process... BF69-1207

Scala - process-music-data/src-scala/final_project/MusicDataProcessorApp.scala - Eclipse

```
File Edit Refactor Navigate Search Project Scala Run Window Help
```

Quick Access Java EE Debug Java Scala Team Synchronizing

Package Explorer

- Assignment14T1 [Assignment14T1 master]
- Assignment15_1 [Assignment15_1 master]
- Assignment15_2 [Assignment15_2 master]
- Assignment16_T1 [Assignment16_T1 master]
- Assignment3_2 [Assignment3_2 master]
- Assignment3_3 [Assignment3_3 master]
- Assignment4_1 [Assignment4_1 master]
- Hive [Hive master]
- load-base-tables
- mapreduce
- process-music-data
 - src-scala
 - final_project
 > MobileMusicDataProcessor.scala
 > MusicDataAnalyzer.scala
 > MusicDataEnricher.scala
 > MusicDataPopulateMapFromLookupTables.scala
 > MusicDataProcessorApp.scala
 > WebMusicDataProcessor.scala
 - src/test/java
 - JRE System Library [JavaSE-1.8]
 - Maven Dependencies
 - src
 - target
 - pom.xml
- Project2
- ScalaTest
- Test

```
1 package final_project
2 import org.apache.log4j.{ Level, LogManager, PropertyConfigurator }
3
4 object MusicDataProcessorApp {
5   def main(args: Array[String]) {
6     val web_file_path = "/data/web/file-1.xml"
7     val mobile_file_path = "file:///data/mob/file.txt"
8     val log = LogManager.getRootLogger
9     log.setLevel(Level.INFO)
10
11     val conf = new SparkConf().setAppName("MusicDataProcessApp").setMaster("local[2]")
12     val sc = new SparkContext(conf)
13     val sqlContext = new SQLContext(sc)
14     import sqlContext.implicits._
15
16     sc.setLogLevel("ERROR")
17
18     val populateMusicDataMap = new MusicDataPopulateMapFromLookupTables(sc)
19
20     val stndIdGeoCdMap:Map[String, String] = populateMusicDataMap.getStationIdGeoCdMap()
21     val broadcastStndIdGeoCdMap = sc.broadcast(stndIdGeoCdMap)
22
23     val songArtistMap = populateMusicDataMap.getSongArtistMap()
24     val broadcastSongArtistMap = sc.broadcast(songArtistMap)
25
26     val userArtistMap = populateMusicDataMap.getUserArtistMap()
27     val broadcastUserArtistMap = sc.broadcast(userArtistMap)
28
29     val userSubscription:Map[String, (Long, Long)] = populateMusicDataMap.getUserSubscriptionMap()
30     val broadcastUserSubscription = sc.broadcast(userSubscription)
31
32
33
34
35
36
37
38 }
```

Type here to search

Scala - process-music-data/src-scala/final_project/MusicDataProcessorApp.scala - Eclipse

```

import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._

object MusicDataProcessorApp {
    def main(args: Array[String]): Unit = {
        val spark = SparkSession.builder()
            .appName("Music Data Processor")
            .getOrCreate()

        // Read data from CSV files
        val songsDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("songs.csv")

        val artistsDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("artists.csv")

        val usersDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("users.csv")

        val subscriptionsDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("subscriptions.csv")

        // Process data
        val songArtistMap = songsDF
            .join(artistsDF, songsDF("artist_id") === artistsDF("id"))
            .select(songsDF("song_id"), artistsDF("name"))

        val userArtistMap = usersDF
            .join(artistsDF, usersDF("artist_id") === artistsDF("id"))
            .select(usersDF("user_id"), artistsDF("name"))

        val userSubscriptionMap = subscriptionsDF
            .select(subscriptionsDF("user_id"), subscriptionsDF("subscription_type"))

        // Broadcast data
        val songArtistBroadcast = songArtistMap
            .rdd
            .mapPartitions(partition => partition.map(row => (row.getString(0), row.getString(1))))
            .broadcast()

        val userArtistBroadcast = userArtistMap
            .rdd
            .mapPartitions(partition => partition.map(row => (row.getString(0), row.getString(1))))
            .broadcast()

        val userSubscriptionBroadcast = userSubscriptionMap
            .rdd
            .mapPartitions(partition => partition.map(row => (row.getString(0), row.getString(1))))
            .broadcast()

        // Load base tables
        val songsTable = spark.createDataFrame(songsDF)
        val artistsTable = spark.createDataFrame(artistsDF)
        val usersTable = spark.createDataFrame(usersDF)
        val subscriptionsTable = spark.createDataFrame(subscriptionsDF)

        // Process web data
        val webSongsDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("web_songs.csv")

        val webArtistsDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("web_artists.csv")

        val webUsersDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("web_users.csv")

        val webSubscriptionsDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("web_subscriptions.csv")

        // Process mobile data
        val mobileSongsDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("mobile_songs.csv")

        val mobileArtistsDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("mobile_artists.csv")

        val mobileUsersDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("mobile_users.csv")

        val mobileSubscriptionsDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("mobile_subscriptions.csv")

        // Enrich data
        val enrichedAllDataDF = songsTable
            .join(songArtistBroadcast, songsTable("song_id") === songArtistBroadcast.key)
            .join(artistsTable, songArtistBroadcast.value === artistsTable("name"))
            .join(usersTable, usersTable("user_id") === songArtistBroadcast.value)
            .join(userSubscriptionBroadcast, usersTable("user_id") === userSubscriptionBroadcast.key)
            .select(songsTable("song_id"), artistsTable("name"), usersTable("user_id"), userSubscriptionBroadcast.value)

        val enrichedMobileDataDF = mobileSongsDF
            .join(mobileSongArtistBroadcast, mobileSongsDF("song_id") === mobileSongArtistBroadcast.key)
            .join(mobileArtistsTable, mobileSongArtistBroadcast.value === mobileArtistsTable("name"))
            .join(mobileUsersTable, mobileUsersTable("user_id") === mobileSongArtistBroadcast.value)
            .join(mobileUserSubscriptionBroadcast, mobileUsersTable("user_id") === mobileUserSubscriptionBroadcast.key)
            .select(mobileSongsDF("song_id"), mobileArtistsTable("name"), mobileUsersTable("user_id"), mobileUserSubscriptionBroadcast.value)

        // Analyze data
        val musicDataAnalyzer = new MusicDataAnalyzer(spark, sqlContext, enrichedAllDataDF)
        val analysisResults = musicDataAnalyzer.analyze()

        // Output results
        analysisResults.show(100)
    }
}

```

Scala - process-music-data/src-scala/final_project/MusicDataProcessorApp.scala - Eclipse

```

import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._

object MusicDataProcessorApp {
    def main(args: Array[String]): Unit = {
        val spark = SparkSession.builder()
            .appName("Music Data Processor")
            .getOrCreate()

        // Read data from CSV files
        val songsDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("songs.csv")

        val artistsDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("artists.csv")

        val usersDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("users.csv")

        val subscriptionsDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("subscriptions.csv")

        // Process data
        val songArtistMap = songsDF
            .join(artistsDF, songsDF("artist_id") === artistsDF("id"))
            .select(songsDF("song_id"), artistsDF("name"))

        val userArtistMap = usersDF
            .join(artistsDF, usersDF("artist_id") === artistsDF("id"))
            .select(usersDF("user_id"), artistsDF("name"))

        val userSubscriptionMap = subscriptionsDF
            .select(subscriptionsDF("user_id"), subscriptionsDF("subscription_type"))

        // Broadcast data
        val songArtistBroadcast = songArtistMap
            .rdd
            .mapPartitions(partition => partition.map(row => (row.getString(0), row.getString(1))))
            .broadcast()

        val userArtistBroadcast = userArtistMap
            .rdd
            .mapPartitions(partition => partition.map(row => (row.getString(0), row.getString(1))))
            .broadcast()

        val userSubscriptionBroadcast = userSubscriptionMap
            .rdd
            .mapPartitions(partition => partition.map(row => (row.getString(0), row.getString(1))))
            .broadcast()

        // Load base tables
        val songsTable = spark.createDataFrame(songsDF)
        val artistsTable = spark.createDataFrame(artistsDF)
        val usersTable = spark.createDataFrame(usersDF)
        val subscriptionsTable = spark.createDataFrame(subscriptionsDF)

        // Process web data
        val webSongsDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("web_songs.csv")

        val webArtistsDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("web_artists.csv")

        val webUsersDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("web_users.csv")

        val webSubscriptionsDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("web_subscriptions.csv")

        // Process mobile data
        val mobileSongsDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("mobile_songs.csv")

        val mobileArtistsDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("mobile_artists.csv")

        val mobileUsersDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("mobile_users.csv")

        val mobileSubscriptionsDF = spark.read.format("csv")
            .option("header", "true")
            .option("sep", ",")
            .load("mobile_subscriptions.csv")

        // Enrich data
        val enrichedAllDataDF = songsTable
            .join(songArtistBroadcast, songsTable("song_id") === songArtistBroadcast.key)
            .join(artistsTable, songArtistBroadcast.value === artistsTable("name"))
            .join(usersTable, usersTable("user_id") === songArtistBroadcast.value)
            .join(userSubscriptionBroadcast, usersTable("user_id") === userSubscriptionBroadcast.key)
            .select(songsTable("song_id"), artistsTable("name"), usersTable("user_id"), userSubscriptionBroadcast.value)

        val enrichedMobileDataDF = mobileSongsDF
            .join(mobileSongArtistBroadcast, mobileSongsDF("song_id") === mobileSongArtistBroadcast.key)
            .join(mobileArtistsTable, mobileSongArtistBroadcast.value === mobileArtistsTable("name"))
            .join(mobileUsersTable, mobileUsersTable("user_id") === mobileSongArtistBroadcast.value)
            .join(mobileUserSubscriptionBroadcast, mobileUsersTable("user_id") === mobileUserSubscriptionBroadcast.key)
            .select(mobileSongsDF("song_id"), mobileArtistsTable("name"), mobileUsersTable("user_id"), mobileUserSubscriptionBroadcast.value)

        // Analyze data
        val musicDataAnalyzer = new MusicDataAnalyzer(spark, sqlContext, enrichedAllDataDF)
        val analysisResults = musicDataAnalyzer.analyze()

        // Output results
        analysisResults.show(100)
    }
}

```

Module7: Script for running the Music Data App

A bash script music_data_spark_app.sh is created and its content is as below:

```
#!/bin/bash

export HBASE_PATH=`/usr/local/hbase/bin/hbase classpath`
cd $SPARK_HOME

echo "Starting the Music Data Application"
bin/spark-submit --class final_project.MusicDataProcessorApp --driver-class-path $HBASE_PATH --
master local[2] --packages com.databricks:spark-csv_2.10:1.4.0 ~/scala_eclipse/workspace/process-
music-data/target/process_music_data-0.0.1-SNAPSHOT.jar > output.log 2>&1

echo "Completed the Music Data Application"
```

Screenshot is as below:

Applications Places System Wed Jan 3, 10:51 PM acadgild

acadgild@localhost:~/project_music_data_analysis

File Edit View Search Terminal Tabs Help

```
#!/bin/bash

export HBASE_PATH=`/usr/local/hbase/bin/hbase classpath`
cd $SPARK_HOME

echo "Starting the Music Data Application"
bin/spark-submit --class final_project.MusicDataProcessorApp --driver-class-path $HBASE_PATH --master local[2] --packages com.databricks:spark-csv_2.10:1.4.0 ~/scala_eclipse/workspace/process-music-data/target/process_music_data-0.0.1-SNAPSHOT.jar > output.log 2>&1

echo "Completed the Music Data Application"
~
```

:wq!

acadgild@localhost:~/... workspace - process-...

The screenshot shows a terminal window titled "acadgild" with three tabs open. The current tab displays the command "sh ./music_data_spark_app.sh" being run, which starts and completes a "Music Data Application". The terminal is located on a desktop environment with a menu bar at the top.

```
acadgild@localhost:~/project_music_data_analysis$ sh ./music_data_spark_app.sh
Starting the Music Data Application
Completed the Music Data Application
[acadgild@localhost project_music_data_analysis]$
```

Next, a cron job is created using “crontab –e” so that the script is run every 3 hours

Content is as below

```
0 */4 * * * /home/acadgild/project_music_data_analysis/music_data_spark_app.sh
```

The screenshot is as below:

The screenshot shows a terminal window titled "acadgild" with three tabs open. The current tab displays a crontab entry:

```
*/4 * * * * /home/acadgild/project_music_data_analysis/music_data_spark_app.sh
```

The terminal window also shows the path "/tmp/crontab.xcvewp" and the file size "1L, 79C".

3. Highlights of the project

- i. No join of query is used while analysis. Data is already enriched with new fields and using broadcast maps on Lookup tables so as to avoid any join
- ii. Logger has been used in all over the code to allow

4. Issues Faced and how to resolve it

While doing project, I face following issues and how I resolved it

- i. XML processing – I found the databricks provides he XML processing APIs in spark. But it was not working. The main issue I face

I was trying the process the xml dataset (file-1.xml) to covert to spark dataframe as given for the final project. I have created a scala Object ReadXMLMusicData.scala based on the steps given in

<https://github.com/databricks/spark-xml>

Created maven project with pom.xml attached. Using mvn install, I am able to generate jar file read_xml_music_data-0.0.1-SNAPSHOT.jar . However when I try to deploy to spark using spark-submit on Acadgild VM

```
bin/spark-submit --class ReadXMLMusicData --master local[2] --driver-class-path  
/home/acadgild/.m2/repository/com/databricks/spark-xml_2.10/0.4.1/spark-xml_2.11-  
0.4.1.jar ~/scala_eclipse/workspace/read_xml_music_data/target/read_xml_music_data-0.0.1-  
SNAPSHOT.jar
```

```
Exception in thread "main" java.lang.NoSuchMethodError:  
org.apache.spark.sql.DataFrameReader.load(Ljava/lang/String;)Lorg/apache/spark/sql/Dataset;  
at ReadXMLMusicData$.main(ReadXMLMusicData.scala:12)  
at ReadXMLMusicData.main(ReadXMLMusicData.scala)  
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)  
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)  
at java.lang.reflect.Method.invoke(Method.java:497)  
at  
org.apache.spark.deploy.SparkSubmit$.org$apache$spark$deploy$SparkSubmit$$runMain(Spark  
Submit.scala:731)  
at org.apache.spark.deploy.SparkSubmit$.doRunMain$1(SparkSubmit.scala:181)  
at org.apache.spark.deploy.SparkSubmit$.submit(SparkSubmit.scala:206)  
at org.apache.spark.deploy.SparkSubmit$.main(SparkSubmit.scala:121)  
at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
```

I tried all the options

1. --packages com.databricks:spark-xml_2.10:0.4.1
2. --packages com.databricks:spark-xml_2.11:0.4.1
3. --driver-class-path /home/acadgild/.m2/repository/com/databricks/spark-xml_2.10/0.4.1/spark-xml_2.10-0.4.1.jar
4. --driver-class-path /home/acadgild/.m2/repository/com/databricks/spark-xml_2.10/0.4.1/spark-xml_2.11-0.4.1.jar

Acadgild gave the following solution:

DataFrame() was supplanted in Spark 2 by Dataset(). You'll need to import org.apache.spark.sql.Dataset and use that if you're running a Spark 1.6 client with a Spark 2.1 server-side. it would be a lot better off using at least Spark 2.1.+ dependencies.

Follow the below link to download newer version of spark.

[Spark Download](#)

But I solved this I used a xml APIs from scala.xml.XML. Relevant links are as below:

<https://alvinalexander.com/scala/how-to-extract-data-from-xml-nodes-in-scala>

<https://hadoopist.wordpress.com/2016/01/08/parsing-a-basic-xml-using-hadoop-and-spark-core-apis/>

<http://www.scala-lang.org/api/2.11.1/scala-xml>

ii. **Task Not Serializable Exception:**

I am encountering "org.apache.spark.SparkException: Task Not Serializable" In the code below when Artist_id is blank I try to get it using Song_Id from broadcastSongArtistMap. The dataset used is the mobile dataset. Basically exception is thrown in the line below:

```
if (x(2).equals("")) broadcastSongArtistMap.value.get(x(1)).getOrElse("Invalid") else x(2),
```

The job is submitted using spark-submit. I would very much appreciate, if you can give me a solution on this (I tried lot of things, but could not make it work)

Code snippet is as below:

```
val songArtistMap = Map("S200" -> "A300",
    "S201" -> "A301",
    "S202" -> "A302",
    "S203" -> "A303",
    "S204" -> "A304",
    "S205" -> "A301",
    "S206" -> "A302",
    "S207" -> "A303",
    "S208" -> "A304",
    "S209" -> "A305"
)
val broadcastSongArtistMap = sc.broadcast(songArtistMap)

case class MusicData(User_id:String, Song_id:String, Artist_id:String, Timestamp:Long,
    Start_ts:Long, End_ts:Long, Geo_cd:String, Station_id:String, Song_end_type:String,
    Like:String, Dislike:String)

val recordRDD = sc.textFile("/home/acadgild/final_project/file.txt")
val recordFieldsRDD = recordRDD.map(x => x.split(",")).filter(x=> x.length ==11)

val recordDF = recordFieldsRDD.map(x => MusicData(x(0),
    x(1),
    if (x(2).equals("")) broadcastSongArtistMap.value.get(x(1)).getOrElse("Invalid") else x(2),
```

```

x(3).toLong,
x(4).toLong,
x(5).toLong,
x(6),
x(7),
x(8),
if (x(9).equals("")) "0" else x(9),
if (x(10).equals("")) "0" else x(10))).toDF

recordDF.show

```

Solution: The problem was that SparkContext as parameter to class MobileMusicDataProcessor was not Serializable. When I removed the SparkContext, it worked

iii. Continue does not work for scala:

There is no support for continue in scala. To use it I created a case class
CustomException(message:String) extends Exception(message)

Then used

iv. HBase support with Spark

I used the following link:

<https://acadgild.com/blog/spark-on-hbase/>

<https://acadgild.com/blog/apache-hbase-beginners-guide/>

Even using that the multiple fields were not working. To solve this, I used the following code:

```

val hbaseRDD = sc.newAPIHadoopRDD(hconf, classOf[TableInputFormat],
classOf[org.apache.hadoop.hbase.io.ImmutableBytesWritable],
classOf[org.apache.hadoop.hbase.client.Result])

val resultRDD = hbaseRDD.map(tuple=>tuple._2)

val keyValueRDD = resultRDD.map(result => (Bytes.toString(result.getRow()).split(" ")(0), (
  if (Bytes.toString(result.getValue(new String("Subscription").getBytes(), new
String("Start_ts").getBytes())).equals("")) 0 else Bytes.toString(result.getValue(new
String("Subscription").getBytes(), new String("Start_ts").getBytes())).toLong,
  if (Bytes.toString(result.getValue(new String("Subscription").getBytes(), new
String("End_ts").getBytes())).equals("")) 0 else Bytes.toString(result.getValue(new
String("Subscription").getBytes(), new String("End_ts").getBytes())).toLong
)))

```

v. Map was giving compilation error

There are two types of Map Mutable and Immutable

I needed to convert to Immutable map to make the compilation work using the following:

```

val map = keyValueRDD.collectAsMap
return scala.collection.immutable.Map(map.toSeq:_*)

```

vi. toDF does not work and giving compilation error
To solve this I needed to add:

```
import sqlContext.implicits._
```

To every class

vii. The following command was giving exception:

```
sqlContext.registerTempTable("MusicDataDetailed")
```

The problem was that I was creating sqlContext in every class and dataframe was created in two different classes.
To solve this I had to use one sqlContext created at MusicDataProcessorApp

```
try {
    if (userId.equals("") && songId.equals("") && artistId.equals("")) {
        throw CustomException("Record is blank")
    } else {
        recordListBuffer += ((userId, songId, artistId, timestampLong, startTsLong, endTsLong, geoCd, stationId,
        songEndType, like, dislike))
    }
} catch {
    case CustomException(msg) => msg
}
```

viii. Not able to store query result to file..

To use this I used databricks csv API:

```
val df = sqlContext.sql("SELECT Songs_id, SUM(duration) AS total_duration_milliseconds FROM
SongDuration "
+ " GROUP BY Songs_id ORDER BY total_duration_milliseconds DESC LIMIT 10")
df.show()
val df1 = df.repartition(1)
df1.write.format("com.databricks.spark.csv").option("header", "true").save(reportBasePath +
"/Top10SongsHavingMaximumRevenue_" + currentTimestamp)
```

```
bin/spark-submit --class final_project.MusicDataProcessorApp --master local[2] --packages
com.databricks:spark-csv_2.10:1.4.0 --driver-java-options "-Dlog4j.configuration=file:/usr/local/spark/spark-
1.6.0-bin-hadoop2.6/conf/log4j.properties" ~/scala_eclipse/workspace/process-music-
data/target/process_music_data-0.0.1-SNAPSHOT.jar > output.log 2>&1
```

ix. AnalysisException whenever I used a field which does not exist because of case. For example Song_Id, but the schema it is Song_id. Even if I restart or whatever I do not work. I needed to change the schema to Songs_id to solve this

x.

