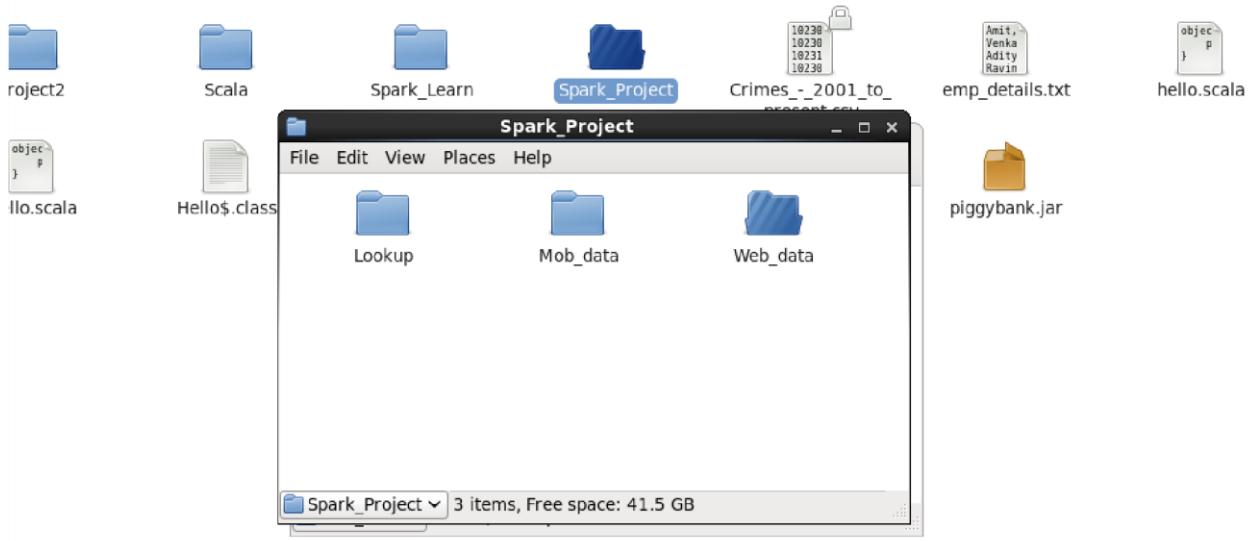
Project : Music Data Analysis Using Spark

Data Ingestion

We will download our web and mobile data and also the lookup data inside our project directory.



First we have to do data validation and enrichment. For that we have to take help of the lookup tables which we will be creating as a NoSql database using Hbase. We will create these hbase tables as

```
hbase(main):011:0> create 'song_artist_id','id','song-id','artist-id'
0 row(s) in 2.2770 seconds
```

Now using importtsv we will bulk load data from hdfs First

we have to place the lookup textfiles in hdfs.

We will use the command :
hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator=, -Dimporttsv.columns="HBASE_ROW_KEY,id" song-artist-id /usr/cloudera/songartist.txt

mapreduce job starts executing

```
                HDFS: Number of write operations=0
        Job Counters
                Launched map tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=16378
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=16378
                Total vcore-milliseconds taken by all map tasks=16378
                Total megabyte-milliseconds taken by all map tasks=16771072
        Map-Reduce Framework
                Map input records=10
                Map output records=10
                Input split bytes=126
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=281
                CPU time spent (ms)=3030
                Physical memory (bytes) snapshot=135897088
                Virtual memory (bytes) snapshot=1509367808
                Total committed heap usage (bytes)=60751872
        ImportTsv
                Bad Lines=0
        File Input Format Counters
                Bytes Read=100
        File Output Format Counters
                Bytes Written=0
```

After execution we can check the values using scan in hbase shell

```
=> Hbase::Table - song_artist_id
hbase(main):012:0> scan 'song_artist_id', {RAW => true, VERSIONS => 10}
ROW                                     COLUMN+CELL
 S200                                   column=id:, timestamp=1511593891438, value=A300
 S201                                   column=id:, timestamp=1511593891438, value=A301
 S202                                   column=id:, timestamp=1511593891438, value=A302
 S203                                   column=id:, timestamp=1511593891438, value=A303
 S204                                   column=id:, timestamp=1511593891438, value=A304
 S205                                   column=id:, timestamp=1511593891438, value=A301
 S206                                   column=id:, timestamp=1511593891438, value=A302
 S207                                   column=id:, timestamp=1511593891438, value=A303
 S208                                   column=id:, timestamp=1511593891438, value=A304
 S209                                   column=id:, timestamp=1511593891438, value=A305
10 row(s) in 0.1140 seconds

hbase(main):013:0>
```

```
hbase(main):040:0> scan 'stn_geo_id'
ROW                                    COLUMN+CELL
 ST400                                 column=cf1:geo_id, timestamp=1511713998711, value=A
 ST401                                 column=cf1:geo_id, timestamp=1511714010235, value=AU
 ST402                                 column=cf1:geo_id, timestamp=1511714028221, value=AP
 ST403                                 column=cf1:geo_id, timestamp=1511714043199, value=J
 ST404                                 column=cf1:geo_id, timestamp=1511714056446, value=E
 ST405                                 column=cf1:geo_id, timestamp=1511714071650, value=A
 ST406                                 column=cf1:geo_id, timestamp=1511714086099, value=AU
 ST407                                 column=cf1:geo_id, timestamp=1511714101654, value=AP
 ST408                                 column=cf1:geo_id, timestamp=1511714119784, value=E
 ST409                                 column=cf1:geo_id, timestamp=1511714131399, value=E
 ST410                                 column=cf1:geo_id, timestamp=1511714145457, value=A
 ST411                                 column=cf1:geo_id, timestamp=1511714156335, value=A
 ST412                                 column=cf1:geo_id, timestamp=1511714170010, value=AP
 ST413                                 column=cf1:geo_id, timestamp=1511714184525, value=J
 ST414                                 column=cf1:geo_id, timestamp=1511714198139, value=E
15 row(s) in 0.1780 seconds

hbase(main):041:0>
```

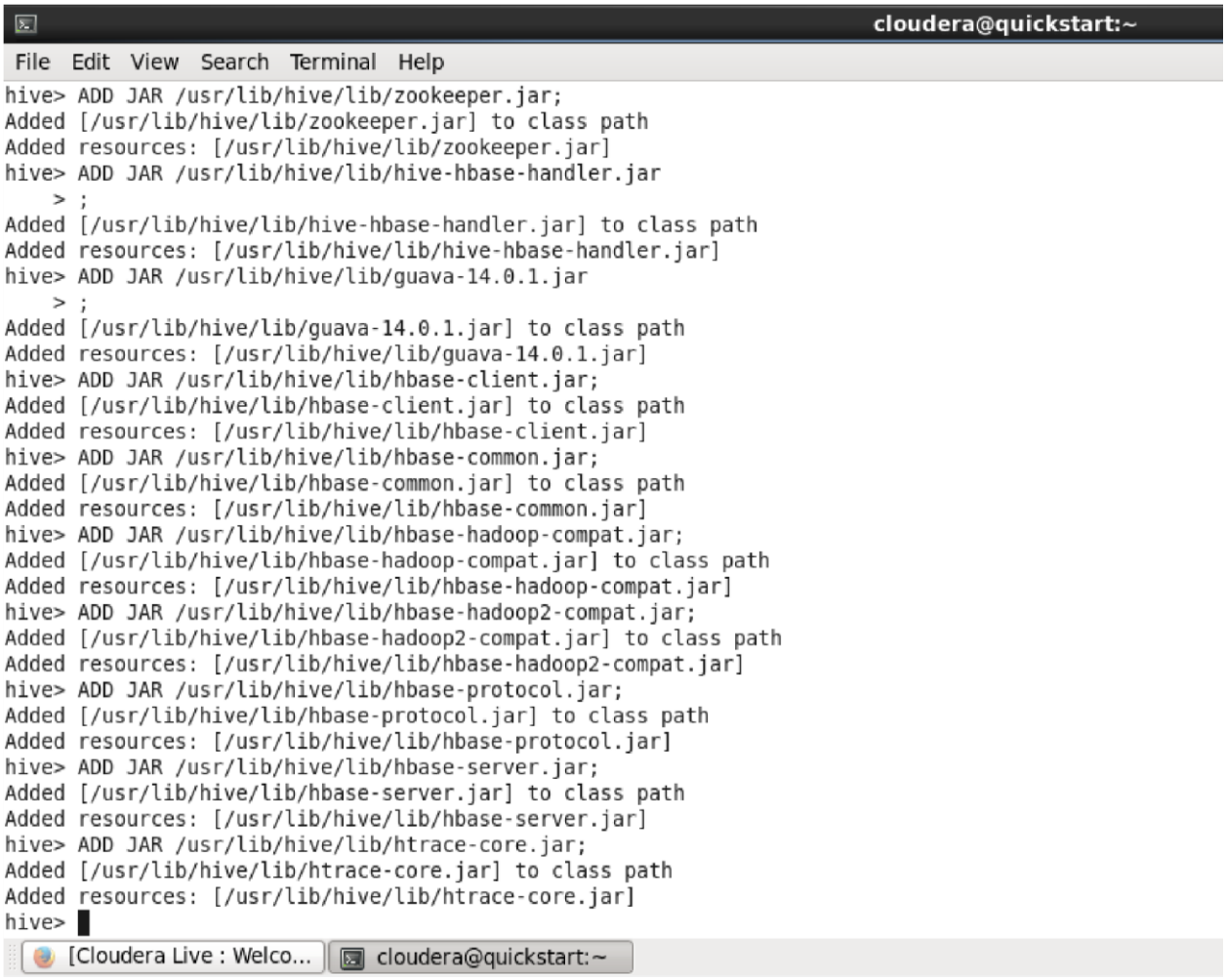
Similarly we have to create the remaining lookup tables also.

```
hbase(main):057:0> scan 'user_artist'
ROW                                    COLUMN+CELL
 U100                                  column=cf1:artist_id, timestamp=1511714516366, value=A300&A301&A302
 U101                                  column=cf1:artist_id, timestamp=1511714543025, value=A301&A302
 U102                                  column=cf1:artist_id, timestamp=1511714562502, value=A302
 U103                                  column=cf1:artist_id, timestamp=1511714589906, value=A303&A301&A302
 U104                                  column=cf1:artist_id, timestamp=1511714616149, value=A304&A301
 U105                                  column=cf1:artist_id, timestamp=1511714639761, value=A305&A301&A302
 U106                                  column=cf1:artist_id, timestamp=1511714662568, value=A301&A302
 U107                                  column=cf1:artist_id, timestamp=1511714684546, value=A302
 U108                                  column=cf1:artist_id, timestamp=1511714705664, value=A300&A303&A304
 U109                                  column=cf1:artist_id, timestamp=1511714726685, value=A301&A303
 U110                                  column=cf1:artist_id, timestamp=1511714747444, value=A302&A301
 U111                                  column=cf1:artist_id, timestamp=1511714766665, value=A303&A301
 U112                                  column=cf1:artist_id, timestamp=1511714786074, value=A304&A301
 U113                                  column=cf1:artist_id, timestamp=1511714807044, value=A305&A302
 U114                                  column=cf1:artist_id, timestamp=1511714827016, value=A300&A301&A302
15 row(s) in 0.3130 seconds

hbase(main):058:0>
```

```
hbase(main):017:0> create 'subcribed_user','sub_start_date','sub_end_date'
0 row(s) in 1.3070 seconds

=> Hbase::Table - subcribed_user
hbase(main):018:0> scan 'subcribed_user', {RAW => true, VERSIONS => 10}
ROW                                          COLUMN+CELL
 U100                                        column=sub_end_date:, timestamp=1511596652955, value=1465130523
 U100                                        column=sub_start_date:, timestamp=1511596652955, value=1465230523
 U101                                        column=sub_end_date:, timestamp=1511596652955, value=1475130523
 U101                                        column=sub_start_date:, timestamp=1511596652955, value=1465230523
 U102                                        column=sub_end_date:, timestamp=1511596652955, value=1475130523
 U102                                        column=sub_start_date:, timestamp=1511596652955, value=1465230523
 U103                                        column=sub_end_date:, timestamp=1511596652955, value=1475130523
 U103                                        column=sub_start_date:, timestamp=1511596652955, value=1465230523
 U104                                        column=sub_end_date:, timestamp=1511596652955, value=1475130523
 U104                                        column=sub_start_date:, timestamp=1511596652955, value=1465230523
 U105                                        column=sub_end_date:, timestamp=1511596652955, value=1475130523
 U105                                        column=sub_start_date:, timestamp=1511596652955, value=1465230523
 U106                                        column=sub_end_date:, timestamp=1511596652955, value=1485130523
 U106                                        column=sub_start_date:, timestamp=1511596652955, value=1465230523
 U107                                        column=sub_end_date:, timestamp=1511596652955, value=1455130523
 U107                                        column=sub_start_date:, timestamp=1511596652955, value=1465230523
 U108                                        column=sub_end_date:, timestamp=1511596652955, value=1465230623
 U108                                        column=sub_start_date:, timestamp=1511596652955, value=1465230523
 U109                                        column=sub_end_date:, timestamp=1511596652955, value=1475130523
 U109                                        column=sub_start_date:, timestamp=1511596652955, value=1465230523
 U110                                        column=sub_end_date:, timestamp=1511596652955, value=1475130523
 U110                                        column=sub_start_date:, timestamp=1511596652955, value=1465230523
 U111                                        column=sub_end_date:, timestamp=1511596652955, value=1475130523
 U111                                        column=sub_start_date:, timestamp=1511596652955, value=1465230523
 U112                                        column=sub_end_date:, timestamp=1511596652955, value=1475130523
 U109                                        column=sub_end_date:, timestamp=1511596652955, value=1475130523
 U109                                        column=sub_start_date:, timestamp=1511596652955, value=1465230523
 U110                                        column=sub_end_date:, timestamp=1511596652955, value=1475130523
 U110                                        column=sub_start_date:, timestamp=1511596652955, value=1465230523
 U111                                        column=sub_end_date:, timestamp=1511596652955, value=1475130523
 U111                                        column=sub_start_date:, timestamp=1511596652955, value=1465230523
 U112                                        column=sub_end_date:, timestamp=1511596652955, value=1475130523
 U112                                        column=sub_start_date:, timestamp=1511596652955, value=1465230523
 U113                                        column=sub_end_date:, timestamp=1511596652955, value=1485130523
 U113                                        column=sub_start_date:, timestamp=1511596652955, value=1465230523
 U114                                        column=sub_end_date:, timestamp=1511596652955, value=1468130523
 U114                                        column=sub_start_date:, timestamp=1511596652955, value=1465230523
15 row(s) in 0.1110 seconds
```

In this way we have done creating our lookup tables in hbase and this ends our data ingestion step.

Data Cleaning, Validation And Enrichment



We have to add these jars to hive classpath to integrate hbase with hive.

We can use the hbase tables that we have created earlier in hbase.

Doing this becomes easy to query the data using hql for data cleaning and validation using joins.

```
hive> use project;
OK
Time taken: 0.062 seconds
hive> set hive.cli.print.current.db=true;
hive (project)> CREATE EXTERNAL TABLE subscribed_user(user_id string,sub_start_date string,sub_end_date string) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHand
ler'  with serdeproperties ("hbase.columns.mapping"=":key,sub_date:start,sub_date:end")  tblproperties("hbase.table.name"="subcribed-user");
OK
Time taken: 7.516 seconds
hive (project)> desc subscribed_user;
OK
user_id                 string                  from deserializer
sub_start_date          string                  from deserializer
sub_end_date            string                  from deserializer
Time taken: 0.364 seconds, Fetched: 3 row(s)
hive (project)> select * from subscribed_user;
OK
U100    1465230523      1465130523
U101    1465230523      1475130523
U102    1465230523      1475130523
U103    1465230523      1475130523
U104    1465230523      1475130523
U105    1465230523      1475130523
U106    1465230523      1485130523
U107    1465230523      1455130523
U108    1465230523      1465230623
U109    1465230523      1475130523
U110    1465230523      1475130523
U111    1465230523      1475130523
U112    1465230523      1475130523
U113    1465230523      1485130523
U114    1465230523      1468130523
Time taken: 1.18 seconds, Fetched: 15 row(s)
hive (project)>
[Cloudera Live : Welco...  cloudera@quickstart:~
```

As shown in the above screenshot, we imported all the data of subscribed_user table in hbase to hive. Similarly we have to do this for all the other hbase lookup tables. We can also update, delete and insert new values to these created hive external tables.

```
hive (project)> CREATE EXTERNAL TABLE lookup1 (song_id string,artist_id string) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'  with serdeproperties ("hba
se.columns.mapping"=":key,cf1:artist_id ") tblproperties ("hbase.table.name"="song_artistid");
OK
Time taken: 1.167 seconds
hive (project)> desc lookup1;
OK
song_id                 string                  from deserializer
artist_id               string                  from deserializer
Time taken: 1.643 seconds, Fetched: 2 row(s)
hive (project)> select * from lookup1;
OK
S200    A300
S201    A301
S202    A302
S203    A303
S204    A304
S205    A301
S206    A302
S207    A303
S208    A304
S209    A305
Time taken: 5.125 seconds, Fetched: 10 row(s)
hive (project)>
```

The above screenshot is of table lookup1 consisting mapping of song_id and artist_id

```
hive (project)> CREATE EXTERNAL TABLE lookup2 (stn_id string,geo_id string) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'  with serdeproperties ("hbase.c
olumns.mapping"=":key,cf1:geo_id ") tblproperties ("hbase.table.name"="stn_geo_id");
OK
Time taken: 0.792 seconds
hive (project)> desc lookup2;
OK
stn_id                  string                  from deserializer
geo_id                  string                  from deserializer
Time taken: 0.341 seconds, Fetched: 2 row(s)
hive (project)> select * from lookup2;
OK
ST400   A
ST401   AU
ST402   AP
ST403   J
ST404   E
ST405   A
ST406   AU
ST407   AP
ST408   E
ST409   E
ST410   A
ST411   A
ST412   AP
ST413   J
ST414   E
Time taken: 0.588 seconds, Fetched: 15 row(s)
hive (project)>
```

The above table is lookup 2 which is mapping of station id and geo id.

```
hive (project)> CREATE EXTERNAL TABLE lookup3 (user_id string,artist_id string) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'  with serdeproperties ("hba
se.columns.mapping"=":key,cf1:artist_id ") tblproperties ("hbase.table.name"="user_artist");
OK
Time taken: 0.696 seconds
hive (project)> desc lookup3;
OK
user_id                 string                  from deserializer
artist_id               string                  from deserializer
Time taken: 0.301 seconds, Fetched: 2 row(s)
hive (project)> select * from lookup3;
OK
U100    A300&A301&A302
U101    A301&A302
U102    A302
U103    A303&A301&A302
U104    A304&A301
U105    A305&A301&A302
U106    A301&A302
U107    A302
U108    A300&A303&A304
U109    A301&A303
U110    A302&A301
U111    A303&A301
U112    A304&A301
U113    A305&A302
U114    A300&A301&A302
Time taken: 0.574 seconds, Fetched: 15 row(s)
hive (project)>
```

Above table is lookup3 which is mapping of user_id and artist_id.

Now we have all four lookup tables in hive which we will be using for data validation.

Next step is to load the mobile and web data in the form of hive tables so that it becomes easy to perform joins on them based on the lookup tables.

First we will create the web_data table

```
[cloudera@quickstart Web_data]$ cat file.xml | tr -d '&' | tr '\n' ' ' | tr '\r' ' ' | sed 's|</record>|</record>\n|g' | grep -v '^\s*$' > web_data.xml
[cloudera@quickstart Web_data]$ cat web_data.xml
<records> <record> <user_id>U106</user_id> <song_id>S205</song_id> <artist_id>A300</artist_id> <timestamp>2016-05-10 12:24:22</timestamp> <start_ts>2016-05-10 12:24:22<
/start_ts> <end_ts>2017-05-09 08:09:22</end_ts> <geo_cd>AP</geo_cd> <station_id>ST407</station_id> <song_end_type>2</song_end_type> <like>1</like> <dislike>1</dislike>
</record>
 <record> <user_id>U114</user_id> <song_id>S209</song_id> <artist_id>A303</artist_id> <timestamp>2016-06-09 22:12:36</timestamp> <start_ts>2016-05-10 12:24:22</start_ts
> <end_ts>2017-05-09 08:09:22</end_ts> <geo_cd>U</geo_cd> <station_id>ST411</station_id> <song_end_type>2</song_end_type> <like>1</like> <dislike>0</dislike> </record>
 <record> <user_id>U113</user_id> <song_id>S203</song_id> <artist_id>A304</artist_id> <timestamp>2016-06-09 22:12:36</timestamp> <start_ts>2016-06-09 22:12:36</start_ts
> <end_ts>2016-05-10 12:24:22</end_ts> <geo_cd>U</geo_cd> <station_id>ST405</station_id> <song_end_type>0</song_end_type> <like>0</like> <dislike>1</dislike> </record>
 <record> <user_id>U108</user_id> <song_id>S200</song_id> <artist_id>A302</artist_id> <timestamp>2016-07-10 01:38:09</timestamp> <start_ts>2016-05-10 12:24:22</start_ts
> <end_ts>2016-07-10 01:38:09</end_ts> <geo_cd>U</geo_cd> <station_id>ST414</station_id> <song_end_type>0</song_end_type> <like>0</like> <dislike>1</dislike> </record>
 <record> <user_id>U102</user_id> <song_id>S203</song_id> <artist_id>A305</artist_id> <timestamp>2016-06-09 22:12:36</timestamp> <start_ts>2016-06-09 22:12:36</start_ts
> <end_ts>2017-05-09 08:09:22</end_ts> <geo_cd>U</geo_cd> <station_id>ST404</station_id> <song_end_type>2</song_end_type> <like>0</like> <dislike>0</dislike> </record>
 <record> <user_id></user_id> <song_id>S208</song_id> <artist_id>A300</artist_id> <timestamp>2016-06-09 22:12:36</timestamp> <start_ts>2017-05-09 08:09:22</start_ts> <e
nd_ts>2016-06-09 22:12:36</end_ts> <geo_cd>U</geo_cd> <station_id>ST411</station_id> <song_end_type>1</song_end_type> <like>0</like> <dislike>1</dislike> </record>
 <record> <user_id>U115</user_id> <song_id>S200</song_id> <artist_id>A300</artist_id> <timestamp>2016-06-09 22:12:36</timestamp> <start_ts>2017-05-09 08:09:22</start_ts
> <end_ts>2016-06-09 22:12:36</end_ts> <geo_cd>AU</geo_cd> <station_id>ST404</station_id> <song_end_type>3</song_end_type> <like>0</like> <dislike>0</dislike> </record>
 <record> <user_id>U111</user_id> <song_id>S204</song_id> <artist_id>A300</artist_id> <timestamp>2016-06-09 22:12:36</timestamp> <start_ts>2016-06-09 22:12:36</start_ts
> <end_ts>2016-07-10 01:38:09</end_ts> <geo_cd>U</geo_cd> <station_id>ST410</station_id> <song_end_type>3</song_end_type> <like>1</like> <dislike>1</dislike> </record>
 <record> <user_id>U120</user_id> <song_id>S201</song_id> <artist_id>A300</artist_id> <timestamp>2017-05-09 08:09:22</timestamp> <start_ts>2016-06-09 22:12:36</start_ts
> <end_ts>2016-07-10 01:38:09</end_ts> <geo_cd></geo_cd> <station_id>ST410</station_id> <song_end_type>3</song_end_type> <like>0</like> <dislike>1</dislike> </record>
 <record> <user_id>U113</user_id> <song_id>S203</song_id> <artist_id></artist_id> <timestamp>2016-06-09 22:12:36</timestamp> <start_ts>2016-06-09 22:12:36</start_ts> <e
nd_ts>2016-06-09 22:12:36</end_ts> <geo_cd>A</geo_cd> <station_id>ST402</station_id> <song_end_type>1</song_end_type> <like>1</like> <dislike>0</dislike> </record>
 <record> <user_id>U109</user_id> <song_id>S203</song_id> <artist_id>A304</artist_id> <timestamp>2016-05-10 12:24:22</timestamp> <start_ts>2017-05-09 08:09:22</start_ts
> <end_ts>2016-07-10 01:38:09</end_ts> <geo_cd>E</geo_cd> <station_id>ST405</station_id> <song_end_type>1</song_end_type> <like>1</like> <dislike>1</dislike> </record>
```

We displayed the file horizontally using above command and saved it as web_data

Now move this data into hdfs

```
[cloudera@quickstart Web_data]$ ls
file.xml  web_data.xml
[cloudera@quickstart Web_data]$ hadoop fs -put web_data.xml /user/cloudera/web_data.xml
[cloudera@quickstart Web_data]$
```

```
hive (project)> ADD JAR /home/cloudera/Downloads/hivexmlserde-1.0.5.3.jar;
Added [/home/cloudera/Downloads/hivexmlserde-1.0.5.3.jar] to class path
Added resources: [/home/cloudera/Downloads/hivexmlserde-1.0.5.3.jar]
hive (project)> CREATE EXTERNAL TABLE web_data_xml (user_id string,song_id string,artist_id string,timestamp string,start_ts string,end_ts string,geo_cd string,station_
id string,song_end_type int,like int,dislike int)
              >  ROW FORMAT SERDE 'com.ibm.spss.hive.serde2.xml.XmlSerDe'
              > WITH SERDEPROPERTIES ("column.xpath.user_id"="/record/user_id/text()",
              > "column.xpath.song_id"="/record/song_id/text()",
              > "column.xpath.artist_id"="/record/artist_id/text()",
              > "column.xpath.timestamp"="/record/timestamp/text()",
              > "column.xpath.start_ts"="/record/start_ts/text()",
              > "column.xpath.end_ts"="/record/end_ts/text()",
              > "column.xpath.geo_cd"="/record/geo_cd/text()",
              > "column.xpath.station_id"="/record/station_id/text()",
              > "column.xpath.song_end_type"="/record/song_end_type/text()",
              > "column.xpath.like"="/record/like/text()",
              > "column.xpath.dislike"="/record/dislike/text()")
              > STORED AS INPUTFORMAT 'com.ibm.spss.hive.serde2.xml.XmlInputFormat'
              > OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
              > LOCATION '/user/cloudera/xml_input/'
              > TBLPROPERTIES ("xmlinput.start"="<record","xmlinput.end"="</record>");
OK
Time taken: 1.762 seconds
hive (project)>
```

```
hive (project)> select * from web_data;
OK
U106    S205    A300    2016-05-10 12:24:22     2016-05-10 12:24:22     2017-05-09 08:09:22     AP      ST407   2       1       1
U114    S209    A303    2016-06-09 22:12:36     2016-05-10 12:24:22     2017-05-09 08:09:22     U       ST411   2       1       0
U113    S203    A304    2016-06-09 22:12:36     2016-06-09 22:12:36     2016-05-10 12:24:22     U       ST405   0       0       1
U108    S200    A302    2016-07-10 01:38:09     2016-05-10 12:24:22     2016-07-10 01:38:09     U       ST414   0       0       1
U102    S203    A305    2016-06-09 22:12:36     2016-06-09 22:12:36     2017-05-09 08:09:22     U       ST404   2       0       0
        S208    A300    2016-06-09 22:12:36     2017-05-09 08:09:22     2016-06-09 22:12:36     U       ST411   1       0       1
U115    S200    A300    2016-06-09 22:12:36     2017-05-09 08:09:22     2016-06-09 22:12:36     AU      ST404   3       0       0
U111    S204    A300    2016-06-09 22:12:36     2016-06-09 22:12:36     2016-07-10 01:38:09     U       ST410   3       1       1
U120    S201    A300    2017-05-09 08:09:22     2016-06-09 22:12:36     2016-07-10 01:38:09             ST410   3       0       1
U113    S203            2016-06-09 22:12:36     2016-06-09 22:12:36     2016-06-09 22:12:36     A       ST402   1       1       0
U109    S203    A304    2016-05-10 12:24:22     2017-05-09 08:09:22     2016-07-10 01:38:09     E       ST405   1       1       1
U110    S202    A303    2017-05-09 08:09:22     2017-05-09 08:09:22     2016-07-10 01:38:09     AU      ST402   2       1       0
U100    S200    A301    2017-05-09 08:09:22     2017-05-09 08:09:22     2017-05-09 08:09:22     AP      ST410   3       1       1
U101    S208    A300    2016-05-10 12:24:22     2016-07-10 01:38:09     2016-05-10 12:24:22     E       ST408   0       1       1
U106    S206    A300    2017-05-09 08:09:22     2016-06-09 22:12:36     2016-05-10 12:24:22     A       ST405   3       1       0
U107    S202    A304    2017-05-09 08:09:22     2016-07-10 01:38:09     2016-05-10 12:24:22     U       ST409   0       0       0
U103    S204    A300    2016-07-10 01:38:09     2017-05-09 08:09:22     2016-06-09 22:12:36     AU      ST411   2       1       0
U103    S202    A300    2016-06-09 22:12:36     2016-06-09 22:12:36     2016-06-09 22:12:36     A       ST415   2       1       1
U113    S203    A303    2016-05-10 12:24:22     2016-07-10 01:38:09     2017-05-09 08:09:22     U       ST408   2       0       0
U113    S204    A301    2017-05-09 08:09:22     2017-05-09 08:09:22     2016-06-09 22:12:36     E       ST415   3       0       1
        NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL
Time taken: 0.279 seconds, Fetched: 21 row(s)
hive (project)>
```

Next step is to append the mob_data data

```
hive (project)> LOAD DATA LOCAL INPATH '/home/cloudera/sayantan/Spark_Project/Web_data/file_web_data' INTO TABLE mob_web_data;
Loading data to table project.mob_web_data
Table project.mob_web_data stats: [numFiles=2, totalSize=3016]
OK
Time taken: 0.475 seconds
hive (project)> select * from mob_web_data;
OK
U114    S207    A303    1465130523      1465230523      1475130523      A       ST415   3       1       0
U107    S202    A303    1495130523      1465230523      1465230523      U       ST415   0       1       1
U100    S204    A302    1495130523      1475130523      1465130523      AU      ST408   2       1       1
U104    S202    A303    1465230523      1475130523      1465130523      A       ST409   2       0       1
U102    S207    A301    1465230523      1485130523      1465230523      AU      ST403   3       1       1
        S203    A302    1495130523      1475130523      1465230523      E       ST400   0       0       1
U106    S202    A302    1465230523      1465130523      1465130523      AU      ST408   0       1       1
U105    S207    A300    1465230523      1485130523      1465130523      U       ST400   2       0       1
U108    S205    A304    1465130523      1465130523      1475130523              ST410   2       1       0
U105    S203            1475130523      1465230523      1465130523      AU      ST408   2       0       1
U110    S203    A300    1465230523      1465130523      1485130523      A       ST415   0       1       1
U113    S200    A303    1465230523      1475130523      1465130523      E       ST413   3       1       1
U119    S208    A302    1495130523      1465230523      1465230523      U       ST415   3       0       0
U118    S208    A303    1475130523      1465130523      1465230523      E       ST415   3       0       0
U107    S210    A302    1475130523      1485130523      1485130523      AP      ST404   2       1       0
U118    S202    A300    1495130523      1465230523      1465230523      AP      ST410   1       0       0
U111    S206    A305    1465130523      1465130523      1485130523      AU      ST415   0       1       1
U116    S208    A303    1465230523      1485130523      1475130523      A       ST413   1       0       1
U101    S202    A300    1465230523      1465130523      1475130523      U       ST401   0       0       1
U120    S206    A303    1495130523      1485130523      1465130523      AU      ST414   0       0       0
U106    S205    A300    2016-05-10 12:24:22     2016-05-10 12:24:22     2017-05-09 08:09:22     AP      ST407   2       1       1
U114    S209    A303    2016-06-09 22:12:36     2016-05-10 12:24:22     2017-05-09 08:09:22     U       ST411   2       1       0
U113    S203    A304    2016-06-09 22:12:36     2016-06-09 22:12:36     2016-05-10 12:24:22     U       ST405   0       0       1
```

```
U116    S208    A303    1465230523      1485130523      1475130523      A       ST413   1       0       1
U101    S202    A300    1465230523      1465130523      1475130523      U       ST401   0       0       1
U120    S206    A303    1495130523      1485130523      1465130523      AU      ST414   0       0       0
U106    S205    A300    2016-05-10 12:24:22     2016-05-10 12:24:22     2017-05-09 08:09:22     AP      ST407   2       1       1
U114    S209    A303    2016-06-09 22:12:36     2016-05-10 12:24:22     2017-05-09 08:09:22     U       ST411   2       1       0
U113    S203    A304    2016-06-09 22:12:36     2016-06-09 22:12:36     2016-05-10 12:24:22     U       ST405   0       0       1
U108    S200    A302    2016-07-10 01:38:09     2016-05-10 12:24:22     2016-07-10 01:38:09     U       ST414   0       0       1
U102    S203    A305    2016-06-09 22:12:36     2016-06-09 22:12:36     2017-05-09 08:09:22     U       ST404   2       0       0
        S208    A300    2016-06-09 22:12:36     2017-05-09 08:09:22     2016-06-09 22:12:36     U       ST411   1       0       1
U115    S200    A300    2016-06-09 22:12:36     2017-05-09 08:09:22     2016-06-09 22:12:36     AU      ST404   3       0       0
U111    S204    A300    2016-06-09 22:12:36     2016-06-09 22:12:36     2016-07-10 01:38:09     U       ST410   3       1       1
U120    S201    A300    2017-05-09 08:09:22     2016-06-09 22:12:36     2016-07-10 01:38:09             ST410   3       0       1
U113    S203            2016-06-09 22:12:36     2016-06-09 22:12:36     2016-06-09 22:12:36     A       ST402   1       1       0
U109    S203    A304    2016-05-10 12:24:22     2017-05-09 08:09:22     2016-07-10 01:38:09     E       ST405   1       1       1
U110    S202    A303    2017-05-09 08:09:22     2017-05-09 08:09:22     2016-07-10 01:38:09     AU      ST402   2       1       0
U100    S200    A301    2017-05-09 08:09:22     2017-05-09 08:09:22     2017-05-09 08:09:22     AP      ST410   3       1       1
U101    S208    A300    2016-05-10 12:24:22     2016-07-10 01:38:09     2016-05-10 12:24:22     E       ST408   0       1       1
U106    S206    A300    2017-05-09 08:09:22     2016-06-09 22:12:36     2016-05-10 12:24:22     A       ST405   3       1       0
U107    S202    A304    2017-05-09 08:09:22     2016-07-10 01:38:09     2016-05-10 12:24:22     U       ST409   0       0       0
U103    S204    A300    2016-07-10 01:38:09     2017-05-09 08:09:22     2016-06-09 22:12:36     AU      ST411   2       1       0
U103    S202    A300    2016-06-09 22:12:36     2016-06-09 22:12:36     2016-06-09 22:12:36     A       ST415   2       1       1
U113    S203    A303    2016-05-10 12:24:22     2016-07-10 01:38:09     2017-05-09 08:09:22     U       ST408   2       0       0
U113    S204    A301    2017-05-09 08:09:22     2017-05-09 08:09:22     2016-06-09 22:12:36     E       ST415   3       0       1
        NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL
Time taken: 0.169 seconds, Fetched: 41 row(s)
hive (project)>
```

Now we have all the required tables in hive for data validation.

We have to join the mob_web_data with lookup1 table to get the valid song id and artist id,we can do this as

```
hive (project)> CREATE TABLE valid_data AS select u.user_id,u.song_id,u.artist_id,u.timestamp,u.start_ts,u.end_ts,u.geo_cd,u.station_id,u.song_end_type,u.like,u.dislike
 from mob_web_data u
            > LEFT JOIN lookup1 l
            > ON u.song_id = l.song_id
            > AND u.artist_id = l.artist_id;
Query ID = cloudera_20171129132121_e06b4c0f-30f0-46a9-a0e5-3a3f99e3a690
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Defaulting to jobconf value of: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
```

```
hive (project)> select * from valid_data;
OK
        NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL
U115    S200    A300    2016-06-09 22:12:36     2017-05-09 08:09:22     2016-06-09 22:12:36     AU      ST404   3       0       0
U100    S200    A301    2017-05-09 08:09:22     2017-05-09 08:09:22     2017-05-09 08:09:22     AP      ST410   3       1       1
U108    S200    A302    2016-07-10 01:38:09     2016-05-10 12:24:22     2016-07-10 01:38:09     U       ST414   0       0       1
U113    S200    A303    1465230523      1475130523      1465130523      E       ST413   3       1       1
U120    S201    A300    2017-05-09 08:09:22     2016-06-09 22:12:36     2016-07-10 01:38:09             ST410   3       0       1
U101    S202    A300    1465230523      1465130523      1475130523      U       ST401   0       0       1
U118    S202    A300    1495130523      1465230523      1465230523      AP      ST410   1       0       0
U103    S202    A300    2016-06-09 22:12:36     2016-06-09 22:12:36     2016-06-09 22:12:36     A       ST415   2       1       1
U106    S202    A302    1465230523      1465130523      1465130523      AU      ST408   0       1       1
U107    S202    A303    1495130523      1465230523      1465230523      U       ST415   0       1       1
U104    S202    A303    1465230523      1475130523      1465130523      A       ST409   2       0       1
U110    S202    A303    2017-05-09 08:09:22     2017-05-09 08:09:22     2016-07-10 01:38:09     AU      ST402   2       1       0
U107    S202    A304    2017-05-09 08:09:22     2016-07-10 01:38:09     2016-05-10 12:24:22     U       ST409   0       0       0
U113    S203            2016-06-09 22:12:36     2016-06-09 22:12:36     2016-06-09 22:12:36     A       ST402   1       1       0
U105    S203            1475130523      1465230523      1465130523      AU      ST408   2       0       1
U110    S203    A300    1465230523      1465130523      1485130523      A       ST415   0       1       1
        S203    A302    1495130523      1475130523      1465230523      E       ST400   0       0       1
U113    S203    A303    2016-05-10 12:24:22     2016-07-10 01:38:09     2017-05-09 08:09:22     U       ST408   2       0       0
U109    S203    A304    2016-05-10 12:24:22     2017-05-09 08:09:22     2016-07-10 01:38:09     E       ST405   1       1       1
U113    S203    A304    2016-06-09 22:12:36     2016-06-09 22:12:36     2016-05-10 12:24:22     U       ST405   0       0       1
U102    S203    A305    2016-06-09 22:12:36     2016-06-09 22:12:36     2017-05-09 08:09:22     U       ST404   2       0       0
U111    S204    A300    2016-06-09 22:12:36     2016-06-09 22:12:36     2016-07-10 01:38:09     U       ST410   3       1       1
U103    S204    A300    2016-07-10 01:38:09     2017-05-09 08:09:22     2016-06-09 22:12:36     AU      ST411   2       1       0
U113    S204    A301    2017-05-09 08:09:22     2017-05-09 08:09:22     2016-06-09 22:12:36     E       ST415   3       0       1
U100    S204    A302    1495130523      1475130523      1465130523      AU      ST408   2       1       1
U106    S205    A300    2016-05-10 12:24:22     2016-05-10 12:24:22     2017-05-09 08:09:22     AP      ST407   2       1       1
U108    S205    A304    1465130523      1465130523      1475130523              ST410   2       1       0
U106    S206    A300    2017-05-09 08:09:22     2016-06-09 22:12:36     2016-05-10 12:24:22     A       ST405   3       1       0
```

Now we will join this valid_data table with lookup2 table to eliminate null geo_cd values

```
hive (project)> SET hive.auto.convert.join=false;
hive (project)> CREATE TABLE valid_data1 AS select u.user_id,u.song_id,u.artist_id,u.timestamp,u.start_ts,u.end_ts,u.geo_cd,u.station_id,u.song_end_type,u.like,u.dislik
e from valid_data u
            > LEFT JOIN lookup2 l
            > ON u.station_id = l.stn_id
            > AND u.geo_cd = l.geo_id;
Query ID = cloudera_20171129154141_8a034b02-ce87-4f98-b0d1-e7bbca6c5ed6
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
```

```
hive (project)> select * from valid_data1;
OK
        NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL
        S203    A302    1495130523      1475130523      1465230523      E       ST400   0       0       1
U105    S207    A300    1465230523      1485130523      1465130523      U       ST400   2       0       1
U101    S202    A300    1465230523      1465130523      1475130523      U       ST401   0       0       1
U113    S203            2016-06-09 22:12:36     2016-06-09 22:12:36     2016-06-09 22:12:36     A       ST402   1       1       0
U110    S202    A303    2017-05-09 08:09:22     2017-05-09 08:09:22     2016-07-10 01:38:09     AU      ST402   2       1       0
U102    S207    A301    1465230523      1485130523      1465230523      AU      ST403   3       1       1
U107    S210    A302    1475130523      1485130523      1485130523      AP      ST404   2       1       0
U115    S200    A300    2016-06-09 22:12:36     2017-05-09 08:09:22     2016-06-09 22:12:36     AU      ST404   3       0       0
U102    S203    A305    2016-06-09 22:12:36     2016-06-09 22:12:36     2017-05-09 08:09:22     U       ST404   2       0       0
U106    S206    A300    2017-05-09 08:09:22     2016-06-09 22:12:36     2016-05-10 12:24:22     A       ST405   3       1       0
U109    S203    A304    2016-05-10 12:24:22     2017-05-09 08:09:22     2016-07-10 01:38:09     E       ST405   1       1       1
U113    S203    A304    2016-06-09 22:12:36     2016-06-09 22:12:36     2016-05-10 12:24:22     U       ST405   0       0       1
U106    S205    A300    2016-05-10 12:24:22     2016-05-10 12:24:22     2017-05-09 08:09:22     AP      ST407   2       1       1
U105    S203            1475130523      1465230523      1465130523      AU      ST408   2       0       1
U100    S204    A302    1495130523      1475130523      1465130523      AU      ST408   2       1       1
U106    S202    A302    1465230523      1465130523      1465130523      AU      ST408   0       1       1
U101    S208    A300    2016-05-10 12:24:22     2016-07-10 01:38:09     2016-05-10 12:24:22     E       ST408   0       1       1
U113    S203    A303    2016-05-10 12:24:22     2016-07-10 01:38:09     2017-05-09 08:09:22     U       ST408   2       0       0
U104    S202    A303    1465230523      1475130523      1465130523      A       ST409   2       0       1
U107    S202    A304    2017-05-09 08:09:22     2016-07-10 01:38:09     2016-05-10 12:24:22     U       ST409   0       0       0
U108    S205    A304    1465130523      1465130523      1475130523              ST410   2       1       0
U120    S201    A300    2017-05-09 08:09:22     2016-06-09 22:12:36     2016-07-10 01:38:09             ST410   3       0       1
U118    S202    A300    1495130523      1465230523      1465230523      AP      ST410   1       0       0
U100    S200    A301    2017-05-09 08:09:22     2017-05-09 08:09:22     2017-05-09 08:09:22     AP      ST410   3       1       1
U111    S204    A300    2016-06-09 22:12:36     2016-06-09 22:12:36     2016-07-10 01:38:09     U       ST410   3       1       1
U103    S204    A300    2016-07-10 01:38:09     2017-05-09 08:09:22     2016-06-09 22:12:36     AU      ST411   2       1       0
U114    S209    A303    2016-06-09 22:12:36     2016-05-10 12:24:22     2017-05-09 08:09:22     U       ST411   2       1       0
        S208    A300    2016-06-09 22:12:36     2017-05-09 08:09:22     2016-06-09 22:12:36     U       ST411   1       0       1
```

Now to validate the records we will create a table valid_data_final by filtering all the null and absent values we could not fill using the lookup tables.

```
hive (project)> CREATE TABLE valid_data_final AS select * from valid_data1 where LENGTH(user_id) > 0 AND LENGTH(artist_id) > 0 AND LENGTH(geo_cd) > 0;
Query ID = cloudera_20171129160202_745be43c-03e4-4b6e-8e13-d7cdf39df0f0
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1511949492079_0002, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1511949492079_0002/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1511949492079_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2017-11-29 16:02:46,956 Stage-1 map = 0%,  reduce = 0%
2017-11-29 16:03:15,326 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.07 sec
MapReduce Total cumulative CPU time: 3 seconds 70 msec
Ended Job = job_1511949492079_0002
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
```

```
hive (project)> select * from valid_data_final;
OK
U105    S207    A300    1465230523      1485130523      1465130523      U       ST400   2       0       1
U101    S202    A300    1465230523      1465130523      1475130523      U       ST401   0       0       1
U110    S202    A303    2017-05-09 08:09:22     2017-05-09 08:09:22     2016-07-10 01:38:09     AU      ST402   2       1       0
U102    S207    A301    1465230523      1485130523      1465230523      AU      ST403   3       1       1
U107    S210    A302    1475130523      1485130523      1485130523      AP      ST404   2       1       0
U115    S200    A300    2016-06-09 22:12:36     2017-05-09 08:09:22     2016-06-09 22:12:36     AU      ST404   3       0       0
U102    S203    A305    2016-06-09 22:12:36     2016-06-09 22:12:36     2017-05-09 08:09:22     U       ST404   2       0       0
U106    S206    A300    2017-05-09 08:09:22     2016-06-09 22:12:36     2016-05-10 12:24:22     A       ST405   3       1       0
U109    S203    A304    2016-05-10 12:24:22     2017-05-09 08:09:22     2016-07-10 01:38:09     E       ST405   1       1       1
U113    S203    A304    2016-06-09 22:12:36     2016-06-09 22:12:36     2016-05-10 12:24:22     U       ST405   0       0       1
U106    S205    A300    2016-05-10 12:24:22     2016-05-10 12:24:22     2017-05-09 08:09:22     AP      ST407   2       1       1
U100    S204    A302    1495130523      1475130523      1465130523      AU      ST408   2       1       1
U106    S202    A302    1465230523      1465130523      1465130523      AU      ST408   0       1       1
U101    S208    A300    2016-05-10 12:24:22     2016-07-10 01:38:09     2016-05-10 12:24:22     E       ST408   0       1       1
U113    S203    A303    2016-05-10 12:24:22     2016-07-10 01:38:09     2017-05-09 08:09:22     U       ST408   2       0       0
U104    S202    A303    1465230523      1475130523      1465130523      A       ST409   2       0       1
U107    S202    A304    2017-05-09 08:09:22     2016-07-10 01:38:09     2016-05-10 12:24:22     U       ST409   0       0       0
U118    S202    A300    1495130523      1465230523      1465230523      AP      ST410   1       0       0
U100    S200    A301    2017-05-09 08:09:22     2017-05-09 08:09:22     2017-05-09 08:09:22     AP      ST410   3       1       1
U111    S204    A300    2016-06-09 22:12:36     2016-06-09 22:12:36     2016-07-10 01:38:09     U       ST410   3       1       1
U103    S204    A300    2016-07-10 01:38:09     2017-05-09 08:09:22     2016-06-09 22:12:36     AU      ST411   2       1       0
U114    S209    A303    2016-06-09 22:12:36     2016-05-10 12:24:22     2017-05-09 08:09:22     U       ST411   2       1       0
U116    S208    A303    1465230523      1485130523      1475130523      A       ST413   1       0       1
U113    S200    A303    1465230523      1475130523      1465130523      E       ST413   3       1       1
U120    S206    A303    1495130523      1485130523      1465130523      AU      ST414   0       0       0
U108    S200    A302    2016-07-10 01:38:09     2016-05-10 12:24:22     2016-07-10 01:38:09     U       ST414   0       0       1
U114    S207    A303    1465130523      1465230523      1475130523      A       ST415   3       1       0
U103    S202    A300    2016-06-09 22:12:36     2016-06-09 22:12:36     2016-06-09 22:12:36     A       ST415   2       1       1
U110    S203    A300    1465230523      1465130523      1485130523      A       ST415   0       1       1
U111    S206    A305    1465130523      1465130523      1485130523      AU      ST415   0       1       1
```

This is the final enriched and validated data we have with ourselves for analysis.

```
scala> val hvdata = hc.sql("show databases").collect()
hvdata: Array[org.apache.spark.sql.Row] = Array([db5], [default], [project])

scala> val hvdata1 = hc.sql("show tables").collect()
hvdata1: Array[org.apache.spark.sql.Row] = Array([lookup1,false], [lookup2,false], [lookup3,false], [mob_data,false], [mob_web_data,false], [subscribed_user,false], [va
lid_data,false], [valid_data1,false], [valid_data_final,false], [web_data,false])

scala> val hvdata2 = hc.sql("select * from valid_data_final").collect()
hvdata2: Array[org.apache.spark.sql.Row] = Array([U105,S207,A300,1465230523,1485130523,1465130523,U,ST400,2,0,1], [U101,S202,A300,1465230523,1465130523,1475130523,U,ST4
01,0,0,1], [U110,S202,A303,2017-05-09 08:09:22,2017-05-09 08:09:22,2016-07-10 01:38:09,AU,ST402,2,1,0], [U102,S207,A301,1465230523,1485130523,1465230523,AU,ST403,3,1,1]
, [U107,S210,A302,1475130523,1485130523,1485130523,AP,ST404,2,1,0], [U115,S200,A300,2016-06-09 22:12:36,2017-05-09 08:09:22,2016-06-09 22:12:36,AU,ST404,3,0,0], [U102,S
203,A305,2016-06-09 22:12:36,2016-06-09 22:12:36,2017-05-09 08:09:22,U,ST404,2,0,0], [U106,S206,A300,2017-05-09 08:09:22,2016-06-09 22:12:36,2016-05-10 12:24:22,A,ST405
,3,1,0], [U109,S203,A304,2016-05-10 12:24:22,2017-05-09 08:09:22,2016-07-10 01:38:09,E,ST405,1,1,1], [U113,S203,A304,2016-06-...
scala>
```

As we can see from above screenshot we have imported the final data for processing in spark by just creating a hive context object.

It’s easy to do processing in spark by running spark SQL commands which is fetching schema created in hive and returning us row objects.

Data Analysis Using Spark SQL

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

```
scala> val data = hc.sql("select * from valid_data_final").show()
+-------+-------+---------+-------------------+-------------------+-------------------+------+----------+-------------+----+-------+
|user_id|song_id|artist_id|          timestamp|           start_ts|             end_ts|geo_cd|station_id|song_end_type|like|dislike|
+-------+-------+---------+-------------------+-------------------+-------------------+------+----------+-------------+----+-------+
|   U105|   S207|     A300|         1465230523|         1485130523|         1465130523|     U|     ST400|            2|   0|      1|
|   U101|   S202|     A300|         1465230523|         1465130523|         1475130523|     U|     ST401|            0|   0|      1|
|   U110|   S202|     A303|2017-05-09 08:09:22|2017-05-09 08:09:22|2016-07-10 01:38:09|    AU|     ST402|            2|   1|      0|
|   U102|   S207|     A301|         1465230523|         1485130523|         1465230523|    AU|     ST403|            3|   1|      1|
|   U107|   S210|     A302|         1475130523|         1485130523|         1485130523|    AP|     ST404|            2|   1|      0|
|   U115|   S200|     A300|2016-06-09 22:12:36|2017-05-09 08:09:22|2016-06-09 22:12:36|    AU|     ST404|            3|   0|      0|
|   U102|   S203|     A305|2016-06-09 22:12:36|2016-06-09 22:12:36|2017-05-09 08:09:22|     U|     ST404|            2|   0|      0|
|   U106|   S206|     A300|2017-05-09 08:09:22|2016-06-09 22:12:36|2016-05-10 12:24:22|     A|     ST405|            3|   1|      0|
|   U109|   S203|     A304|2016-05-10 12:24:22|2017-05-09 08:09:22|2016-07-10 01:38:09|     E|     ST405|            1|   1|      1|
|   U113|   S203|     A304|2016-06-09 22:12:36|2016-06-09 22:12:36|2016-05-10 12:24:22|     U|     ST405|            0|   0|      1|
|   U106|   S205|     A300|2016-05-10 12:24:22|2016-05-10 12:24:22|2017-05-09 08:09:22|    AP|     ST407|            2|   1|      1|
|   U100|   S204|     A302|         1495130523|         1475130523|         1465130523|    AU|     ST408|            2|   1|      1|
|   U106|   S202|     A302|         1465230523|         1465130523|         1465130523|    AU|     ST408|            0|   1|      1|
|   U101|   S208|     A300|2016-05-10 12:24:22|2016-07-10 01:38:09|2016-05-10 12:24:22|     E|     ST408|            0|   1|      1|
|   U113|   S203|     A303|2016-05-10 12:24:22|2016-07-10 01:38:09|2017-05-09 08:09:22|     U|     ST408|            2|   0|      0|
|   U104|   S202|     A303|         1465230523|         1475130523|         1465130523|     A|     ST409|            2|   0|      1|
|   U107|   S202|     A304|2017-05-09 08:09:22|2016-07-10 01:38:09|2016-05-10 12:24:22|     U|     ST409|            0|   0|      0|
|   U118|   S202|     A300|         1495130523|         1465230523|         1465230523|    AP|     ST410|            1|   0|      0|
|   U100|   S200|     A301|2017-05-09 08:09:22|2017-05-09 08:09:22|2017-05-09 08:09:22|    AP|     ST410|            3|   1|      1|
|   U111|   S204|     A300|2016-06-09 22:12:36|2016-06-09 22:12:36|2016-07-10 01:38:09|     U|     ST410|            3|   1|      1|
+-------+-------+---------+-------------------+-------------------+-------------------+------+----------+-------------+----+-------+
only showing top 20 rows

data: Unit = ()

scala>
```

We run count on song_id for those records where like is 1 and order it descending.

```
scala> val problem_1 = hc.sql("select station_id,COUNT(song_id) as count from valid_data_final where like=1 group by station_id order by count DESC LIMIT 10").show()
+----------+-----+
|station_id|count|
+----------+-----+
|     ST415|    5|
|     ST408|    3|
|     ST410|    2|
|     ST411|    2|
|     ST405|    2|
|     ST402|    1|
|     ST403|    1|
|     ST404|    1|
|     ST413|    1|
|     ST407|    1|
+----------+-----+

problem_1: Unit = ()

scala>
```

Determine total duration of songs played by each type of user, where type of user can be 'subscribed' or 'unsubscribed'. An unsubscribed user is the one whose record is either not present in Subscribed_users lookup table or has subscription_end_date earlier than the timestamp of the song played by him.

As we can see from our valid_final_data table the timestamp,start_ts and end_ts are not consistent, we have to first make them consistent.

We use **unix_timestamp** udf to convert all values into epoch time as we can see And

convert it into a dataframe.

```
scala> val test = hc.sql("select user_id,song_id,unix_timestamp(timestamp) as new_timestamp,unix_timestamp(start_ts) as new_startts,unix_timestamp(end_ts) as new_endts
from valid_data_final where LENGTH(timestamp) > 10 OR LENGTH(start_ts) > 10 OR LENGTH(end_ts) > 10").toDF
test: org.apache.spark.sql.DataFrame = [user_id: string, song_id: string, new_timestamp: bigint, new_startts: bigint, new_endts: bigint]

scala> test.show()
+-------+-------+-------------+-----------+----------+
|user_id|song_id|new_timestamp|new_startts| new_endts|
+-------+-------+-------------+-----------+----------+
|   U110|   S202|   1494297562| 1494297562|1468094889|
|   U115|   S200|   1465490556| 1494297562|1465490556|
|   U102|   S203|   1465490556| 1465490556|1494297562|
|   U106|   S206|   1494297562| 1465490556|1462863262|
|   U109|   S203|   1462863262| 1494297562|1468094889|
|   U113|   S203|   1465490556| 1465490556|1462863262|
|   U106|   S205|   1462863262| 1462863262|1494297562|
|   U101|   S208|   1462863262| 1468094889|1462863262|
|   U113|   S203|   1462863262| 1468094889|1494297562|
|   U107|   S202|   1494297562| 1468094889|1462863262|
|   U100|   S200|   1494297562| 1494297562|1494297562|
|   U111|   S204|   1465490556| 1465490556|1468094889|
|   U103|   S204|   1468094889| 1494297562|1465490556|
|   U114|   S209|   1465490556| 1462863262|1494297562|
|   U108|   S200|   1468094889| 1462863262|1468094889|
|   U103|   S202|   1465490556| 1465490556|1465490556|
|   U113|   S204|   1494297562| 1494297562|1465490556|
+-------+-------+-------------+-----------+----------+


scala>
```

Now to get the unsubscribed users we can apply filter.

By running simple transformations and **withcolumn** function we reach our result.

```
scala> test.filter($"new_endts" < $"new_timestamp").show()
+-------+-------+-------------+-----------+----------+
|user_id|song_id|new_timestamp|new_startts| new_endts|
+-------+-------+-------------+-----------+----------+
|   U110|   S202|   1494297562| 1494297562|1468094889|
|   U106|   S206|   1494297562| 1465490556|1462863262|
|   U113|   S203|   1465490556| 1465490556|1462863262|
|   U107|   S202|   1494297562| 1468094889|1462863262|
|   U103|   S204|   1468094889| 1494297562|1465490556|
|   U113|   S204|   1494297562| 1494297562|1465490556|
+-------+-------+-------------+-----------+----------+

scala>
```

To get the song duration we have to subtract end timestamp from start timestamp and divide it by 3600 to get time in hours.

```
scala> test.filter($"new_endts" < $"new_timestamp").withColumn("song_duration",(test("new_startts")-test("new_endts"))/3600).show()
+-------+-------+-------------+-----------+----------+-----------------+
|user_id|song_id|new_timestamp|new_startts| new_endts|    song_duration|
+-------+-------+-------------+-----------+----------+-----------------+
|   U110|   S202|   1494297562| 1494297562|1468094889| 7278.520277777778|
|   U106|   S206|   1494297562| 1465490556|1462863262| 729.8038888888889|
|   U113|   S203|   1465490556| 1465490556|1462863262| 729.8038888888889|
|   U107|   S202|   1494297562| 1468094889|1462863262|1453.2297222222223|
|   U103|   S204|   1468094889| 1494297562|1465490556| 8001.946111111111|
|   U113|   S204|   1494297562| 1494297562|1465490556| 8001.946111111111|
+-------+-------+-------------+-----------+----------+-----------------+

scala>
```

For subscribed we do the following

```
scala> test.filter($"new_endts" > $"new_timestamp").withColumn("song_duration",(test("new_startts")-test("new_endts"))/3600).show()
+-------+-------+-------------+-----------+----------+------------------+
|user_id|song_id|new_timestamp|new_startts| new_endts|     song_duration|
+-------+-------+-------------+-----------+----------+------------------+
|   U102|   S203|   1465490556| 1465490556|1494297562|-8001.946111111111|
|   U109|   S203|   1462863262| 1494297562|1468094889| 7278.520277777778|
|   U106|   S205|   1462863262| 1462863262|1494297562|          -8731.75|
|   U113|   S203|   1462863262| 1468094889|1494297562|-7278.520277777778|
|   U111|   S204|   1465490556| 1465490556|1468094889|-723.4258333333333|
|   U114|   S209|   1465490556| 1462863262|1494297562|          -8731.75|
+-------+-------+-------------+-----------+----------+------------------+

scala>
```

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

Similarly to the 1st problem statement we run count on song_id by providing condition like=1 and song end type=0.

```
scala> val problem_3 = hc.sql("select artist_id,COUNT(song_id) as most_listened from valid_data_final where like=1 AND song_end_type=0 group by artist_id order by most_
listened DESC LIMIT 10").show()
+---------+------------+
|artist_id|most_listened|
+---------+------------+
|     A300|           2|
|     A302|           1|
|     A305|           1|
|     A303|           1|
+---------+------------+

problem_3: Unit = ()

scala>
```

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

Similarly we can filter out top 10 the songs that were liked and completed where like=1 and song end type=0.

```
scala> val problem_4 = hc.sql("select song_id from valid_data_final where like=1 OR song_end_type=0 LIMIT 10").show()
+-------+
|song_id|
+-------+
|   S202|
|   S202|
|   S207|
|   S210|
|   S206|
|   S203|
|   S203|
|   S205|
|   S204|
|   S202|
+-------+

problem_4: Unit = ()

scala>
```

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

We can just run **orderBy** command to the one which we used in problem statement 2 to arrive the result.

```
scala> import org.apache.spark.sql.functions._
import org.apache.spark.sql.functions._

scala> test.filter($"new_endts" < $"new_timestamp").withColumn("song_duration",(test("new_startts")-test("new_endts"))/3600).orderBy($"song_duration".desc).show(10)
+-------+-------+-------------+----------+----------+-----------------+
|user_id|song_id|new_timestamp|new_startts| new_endts|    song_duration|
+-------+-------+-------------+----------+----------+-----------------+
|   U113|   S204|   1494297562| 1494297562|1465490556| 8001.946111111111|
|   U103|   S204|   1468094889| 1494297562|1465490556| 8001.946111111111|
|   U110|   S202|   1494297562| 1494297562|1468094889| 7278.520277777778|
|   U107|   S202|   1494297562| 1468094889|1462863262|1453.2297222222223|
|   U106|   S206|   1494297562| 1465490556|1462863262| 729.8038888888889|
|   U113|   S203|   1465490556| 1465490556|1462863262| 729.8038888888889|
+-------+-------+-------------+----------+----------+-----------------+


scala>
```

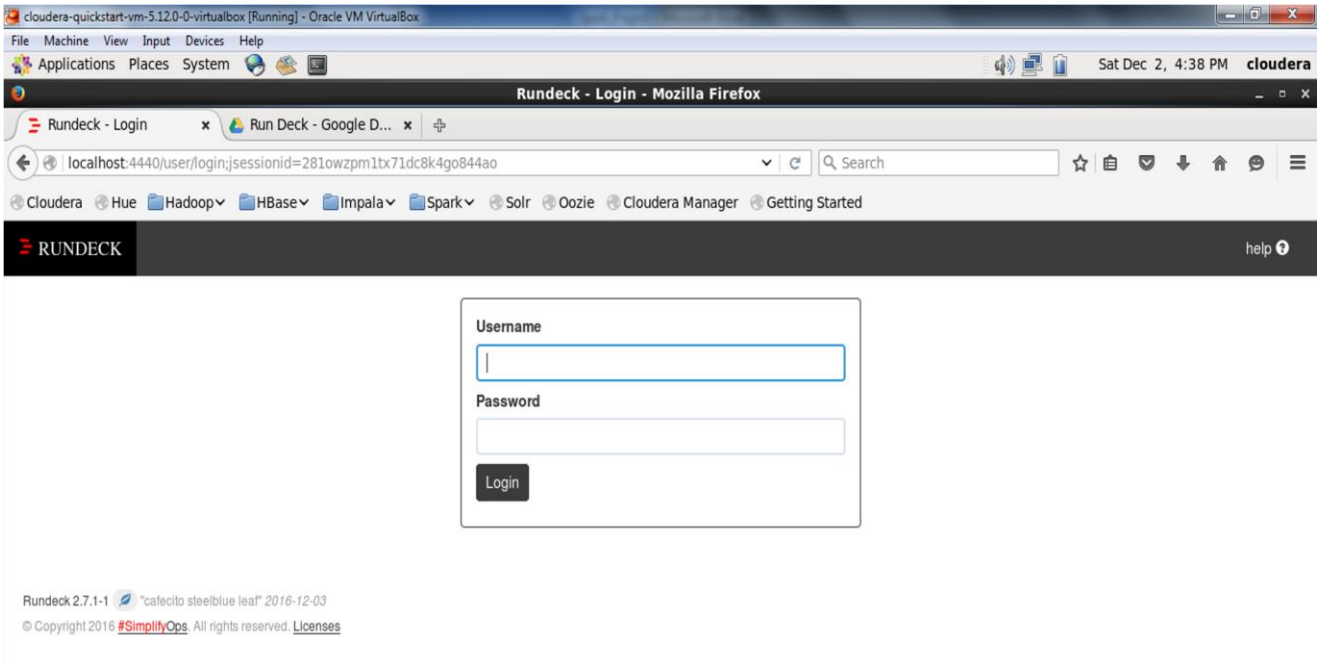
Project Automation

In real time deployment scenarios we cannot run queries or do analysis repeatedly, instead we can schedule our tasks to run on a daily or hourly basis based on our requirement and data availability. In our case we have the data coming from mobile and web every 3 hours. So we have to schedule our data enrichment and analysis tasks to run every 3 hours.

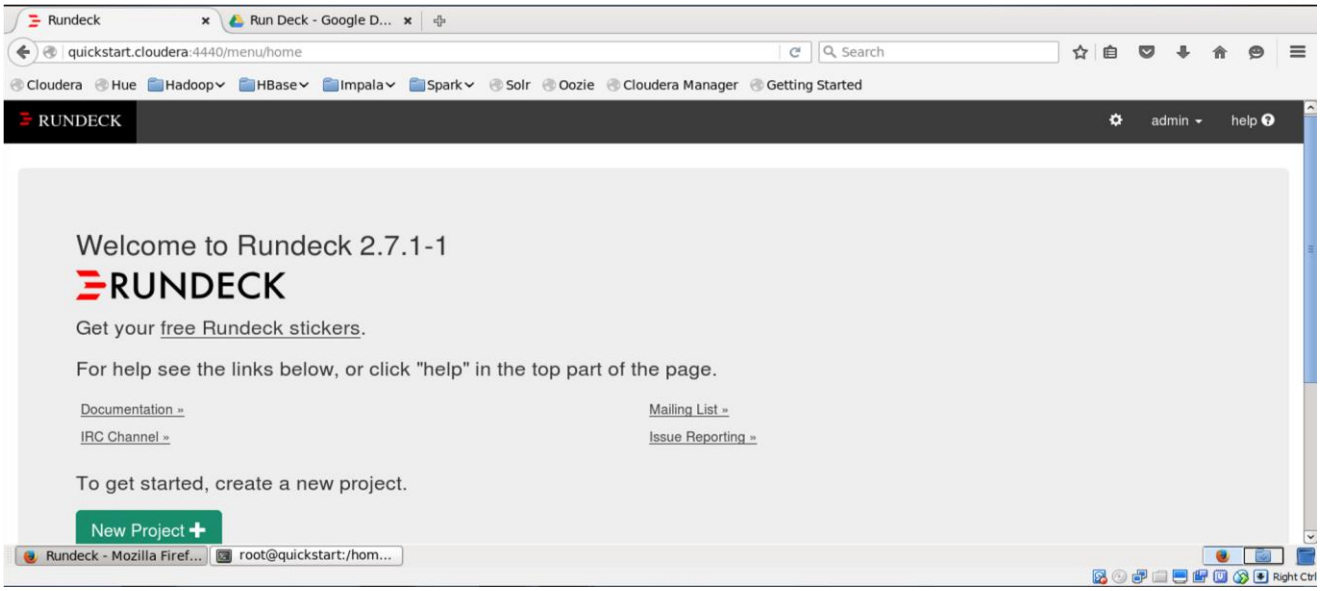
For our first step which is data cleaning and enrichment we will be using a scheduler known as Rundeck. Every time the data is available it will be cleaned using this scheduler and further analysis step takes hold which will also be automated further.

RunDeck is an open source software that is used to automate ad-hoc and routine jobs in the data center or cloud environments. RunDeck allows you to run jobs on distributed environment, here you can select the nodes to run your job. RunDeck also includes other features that make it easy to scale up your scripting efforts including access control, workflow building, scheduling, logging, and integration with external sources for node and options data.

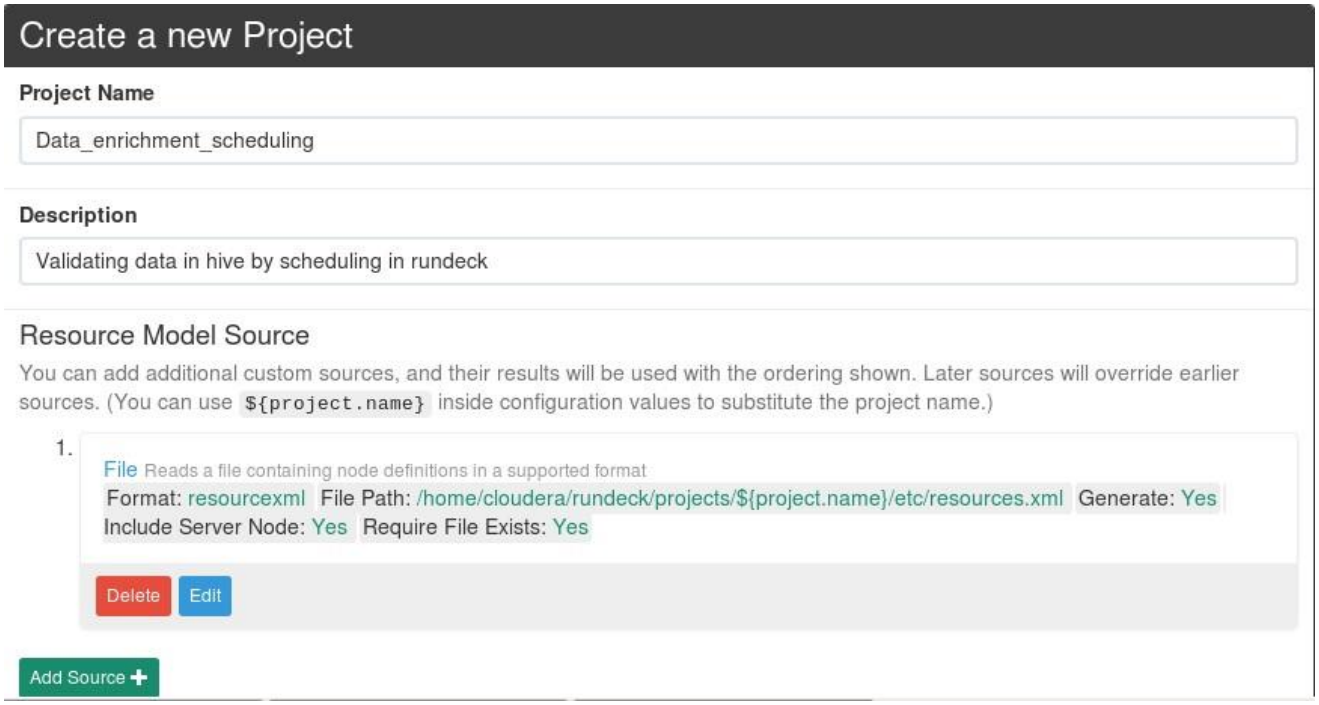
For starting rundeck we have to just run the rundeck jar file and we can see from below screenshot it is working on port 4440.



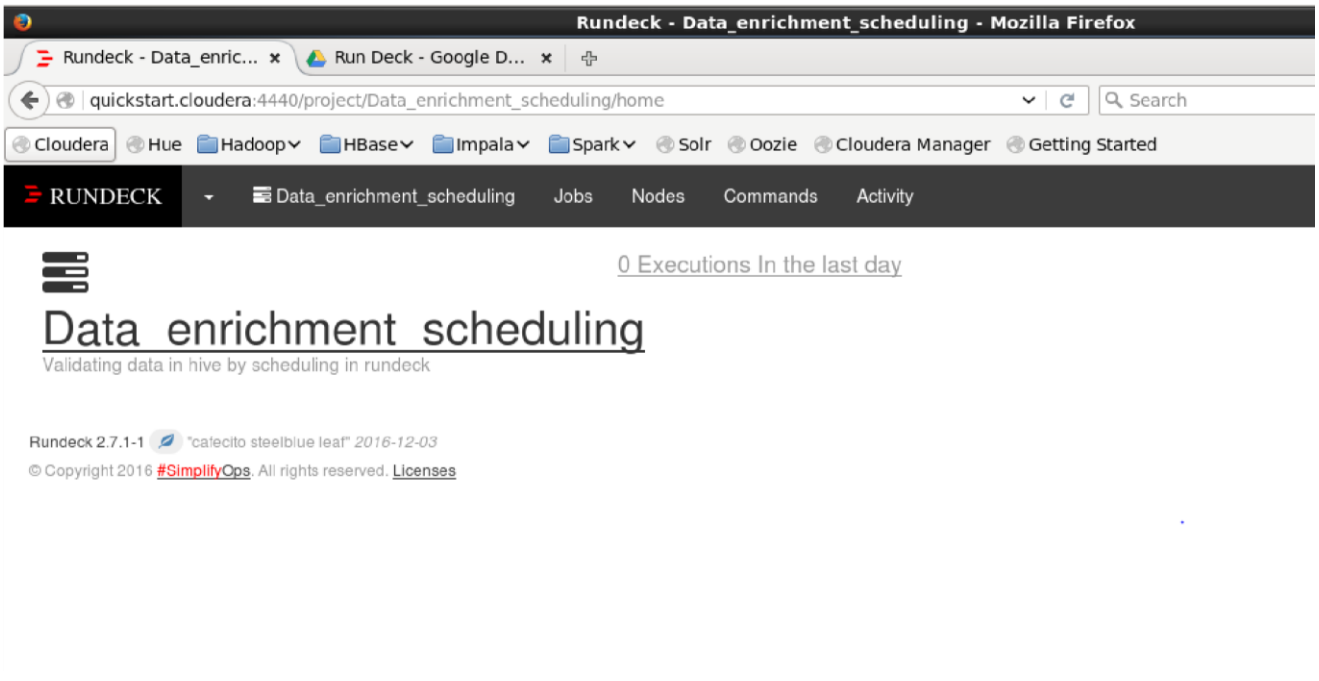
Installed rundeck in my standalone system and started it on port 4440. We will login using default username and password to access its features.



We will start a new project that is running our hive queries.

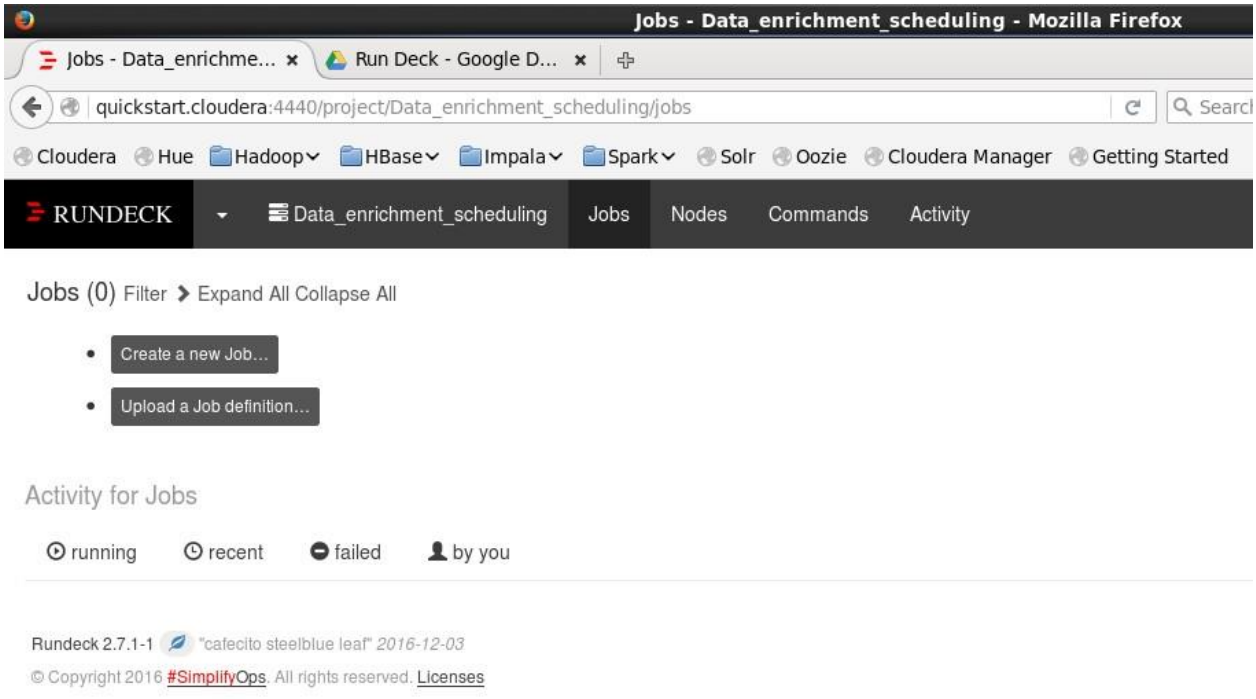


If you are executing on any other remote machine, you need to provide the ssh details below. Here we are running rundeck on a single node cluster.

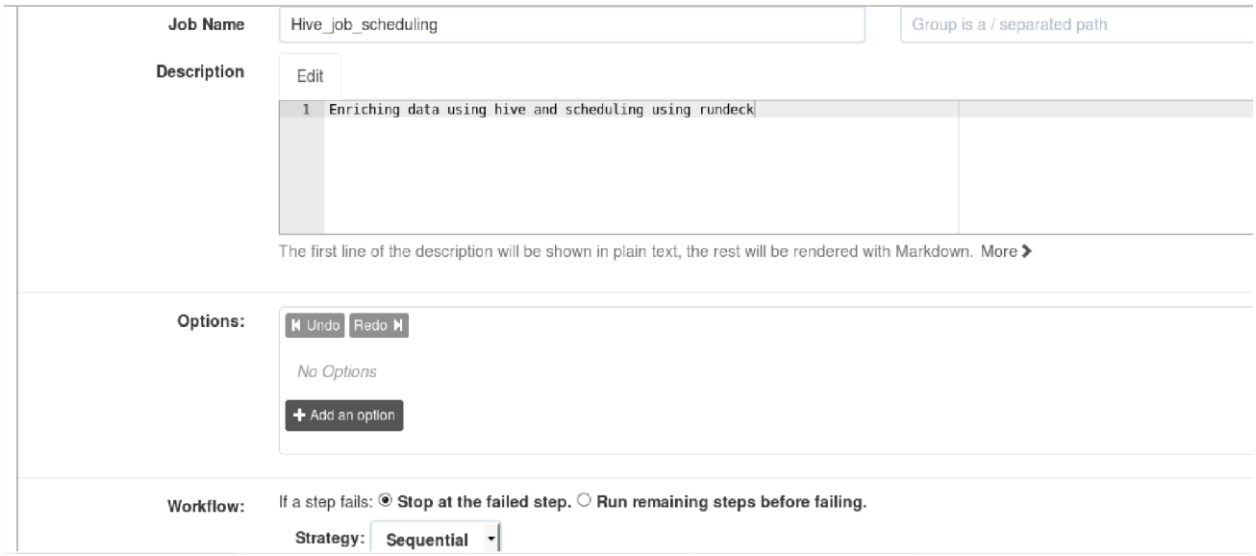


In the next screen, you can see a screen as shown below with different options.

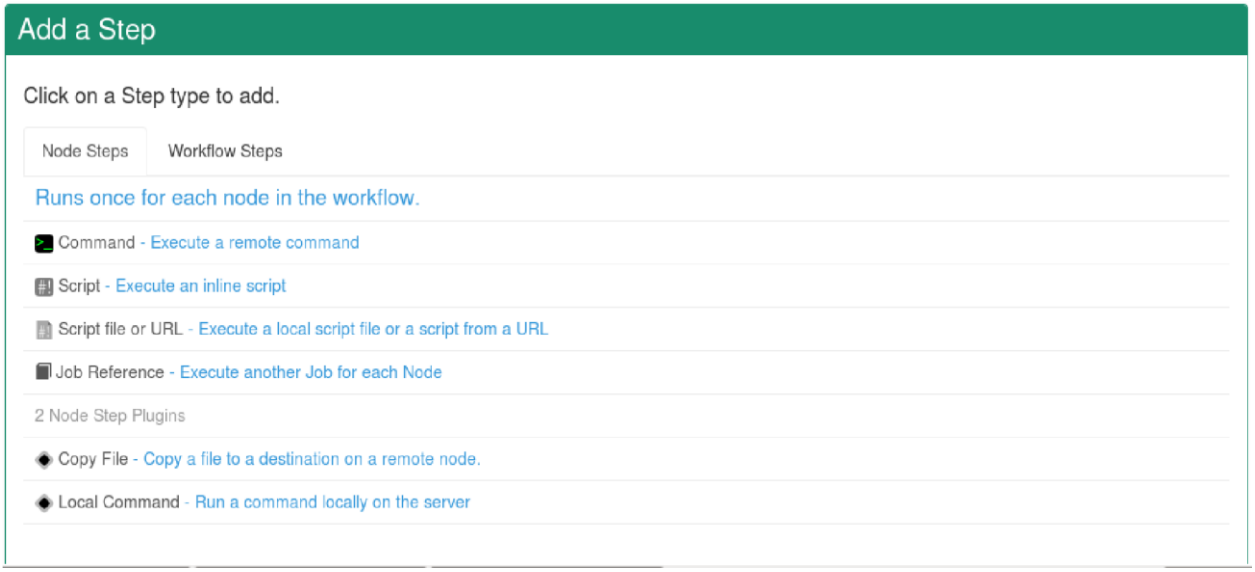
Click on Jobs and click on Create a new job.



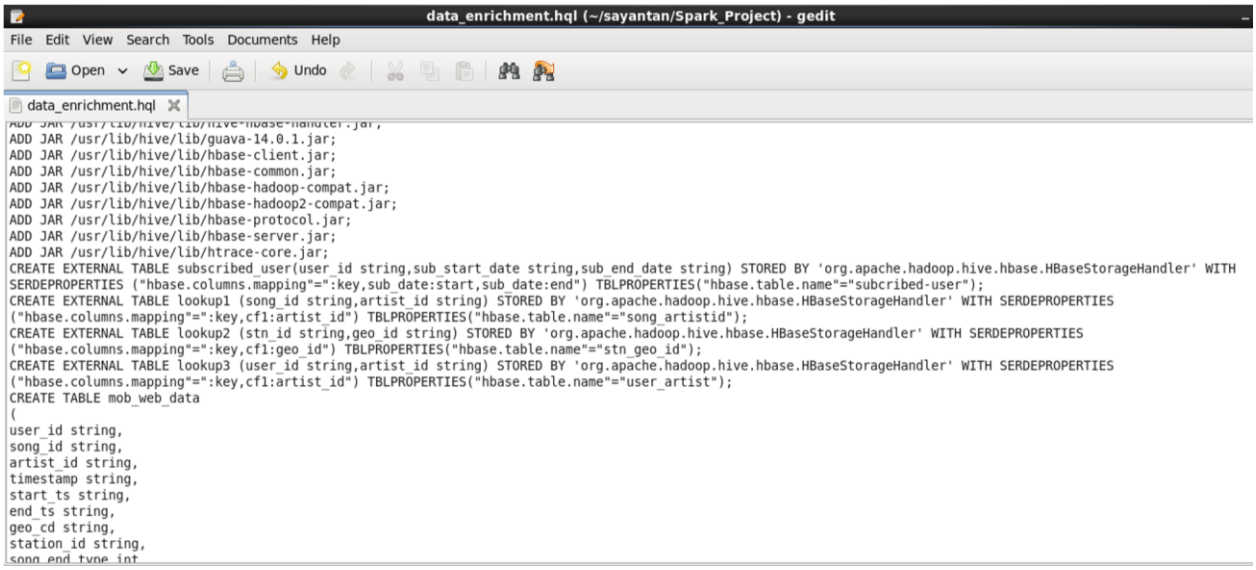
In the next screen provide the necessary details of your job like job name, job description, workflow steps and all.



To provide a job or a query, go the Add step section, and select the option Command.



Here, you need to provide the Hive query file containing a set of Hive queries. Below is our hive query.

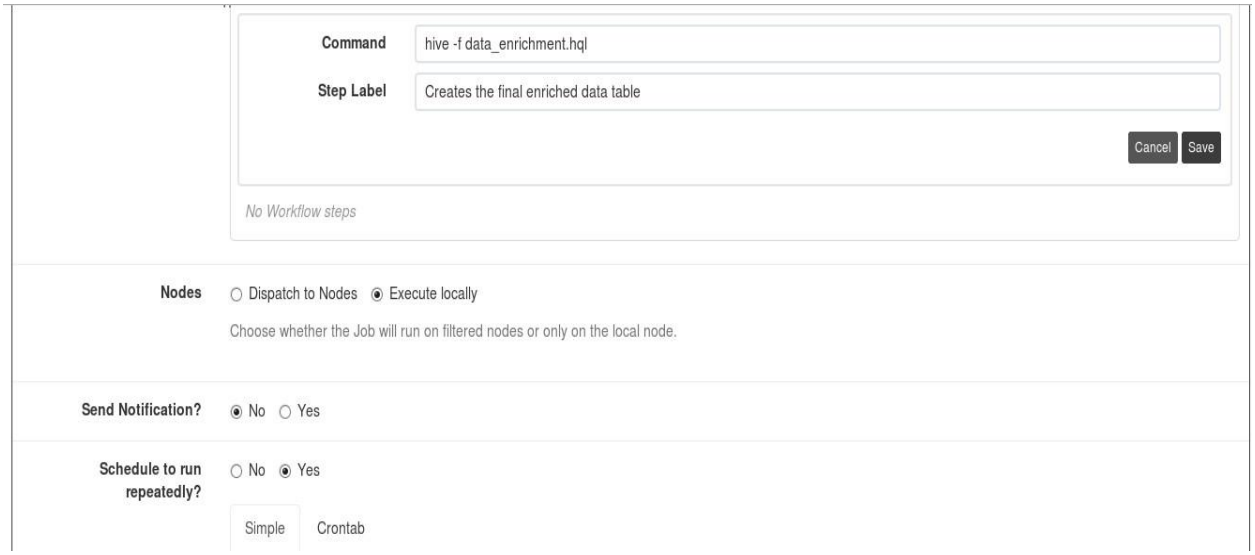


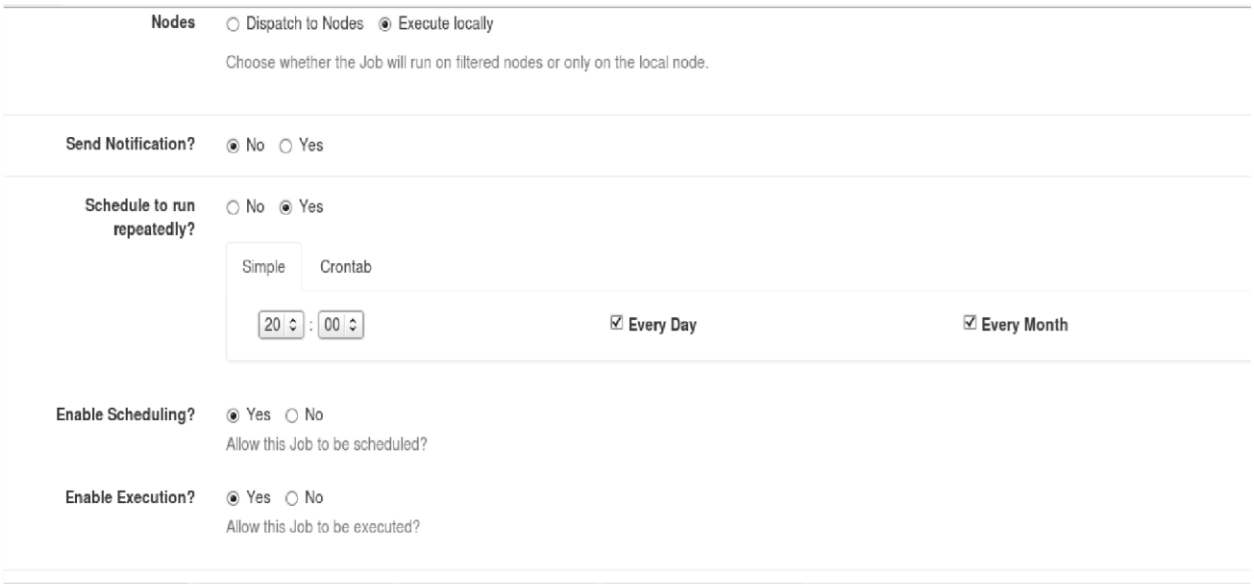
data_enrichment.hql

```
artist_id string,
timestamp string,
start_ts string,
end_ts string,
geo_cd string,
station_id string,
song_end_type int,
like int,
dislike int
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
LOAD DATA LOCAL INPATH '/home/cloudera/sayantan/Spark_Project/Web_data/file_web_data' INTO TABLE mob_web_data;
LOAD DATA LOCAL INPATH '/home/cloudera/sayantan/Spark_Project/Mob_data/file_mob.txt' INTO TABLE mob_web_data;
set hive.auto.convert.join=false;
CREATE TABLE valid_data AS SELECT u.user_id,u.song_id,u.artist_id,u.timestamp,u.start_ts,u.end_ts,u.geo_cd,u.station_id,u.song_end_type,u.like,u.dislike from
mob_web_data u
LEFT JOIN lookup1 l
ON u.song_id = l.song_id
AND u.artist_id = l.artist_id;
CREATE TABLE valid_data1 AS SELECT u.user_id,u.song_id,u.artist_id,u.timestamp,u.start_ts,u.end_ts,u.geo_cd,u.station_id,u.song_end_type,u.like,u.dislike from
valid_data u
LEFT JOIN lookup2 l
ON u.station_id = l.stn_id
AND u.geo_cd = l.geo_id;
CREATE TABLE valid_data_final AS SELECT * FROM valid_data1 WHERE LENGTH(user_id) > 0 AND LENGTH(artist_id) > 0 AND LENGTH(geo_cd) > 0;
SELECT * FROM valid_data_final;
```

The command used to run this script in the command line is shown below:

Note that the file has to reside in hdfs as well as rundeck directory.

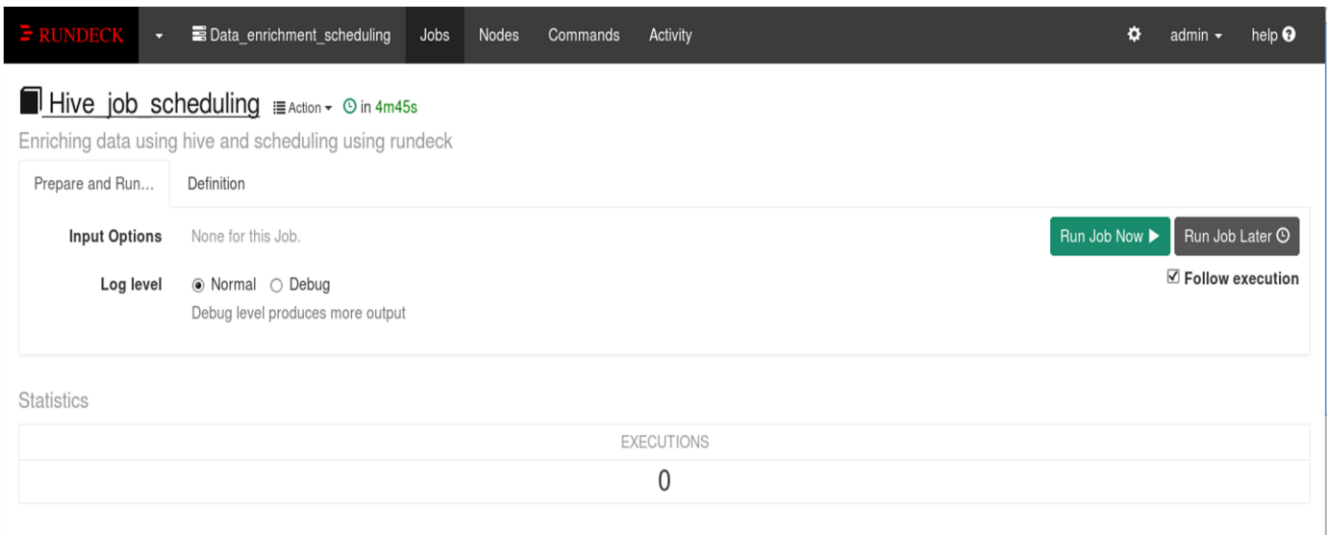


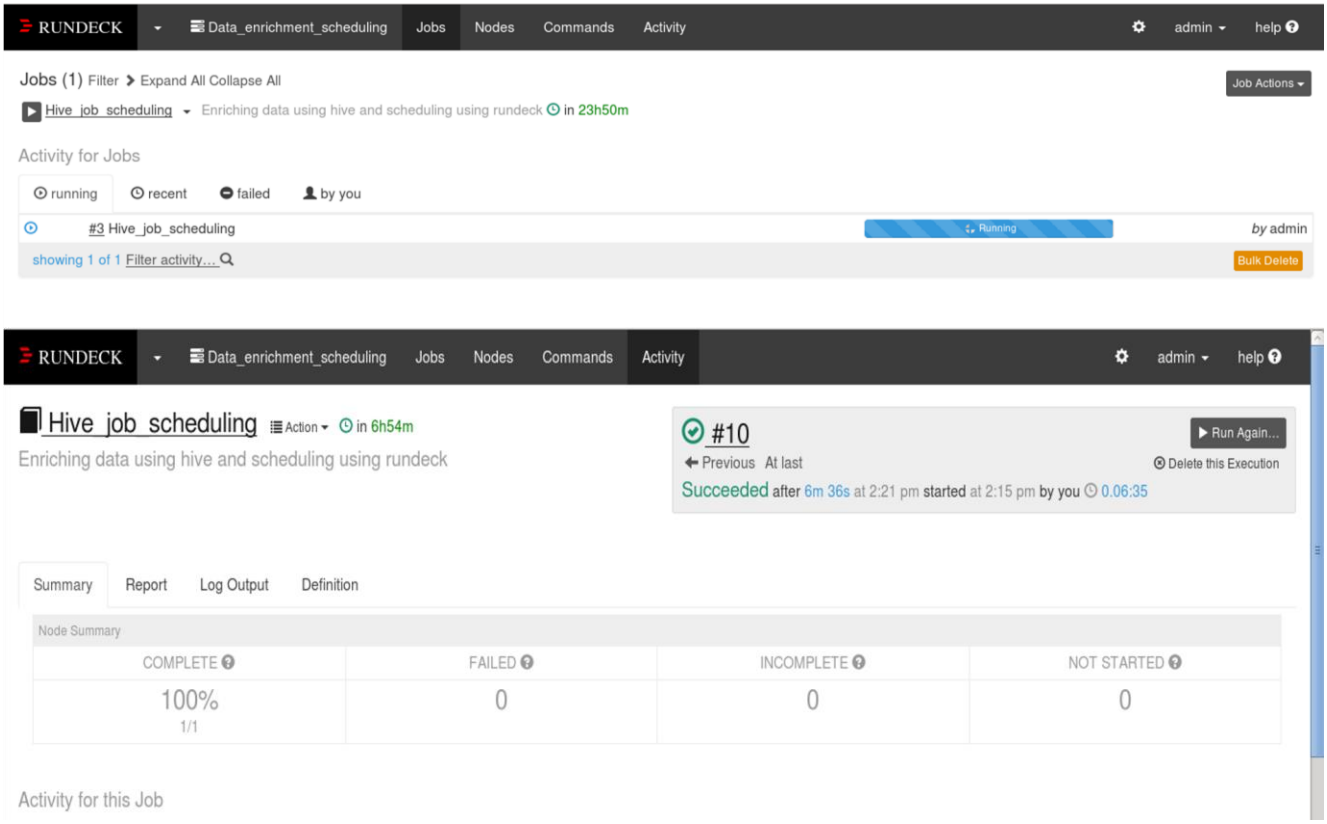


After entering the command, click on save.

You will get two kinds of automation: one is simple and the other is using the Unix crontab. After selecting the necessary option, scroll to the last and click on Create. After clicking on Create, you will be redirected to the Job page. Beside your job, you can see the countdown left to run it. You can also see your job definition in the Definition tab as shown below:

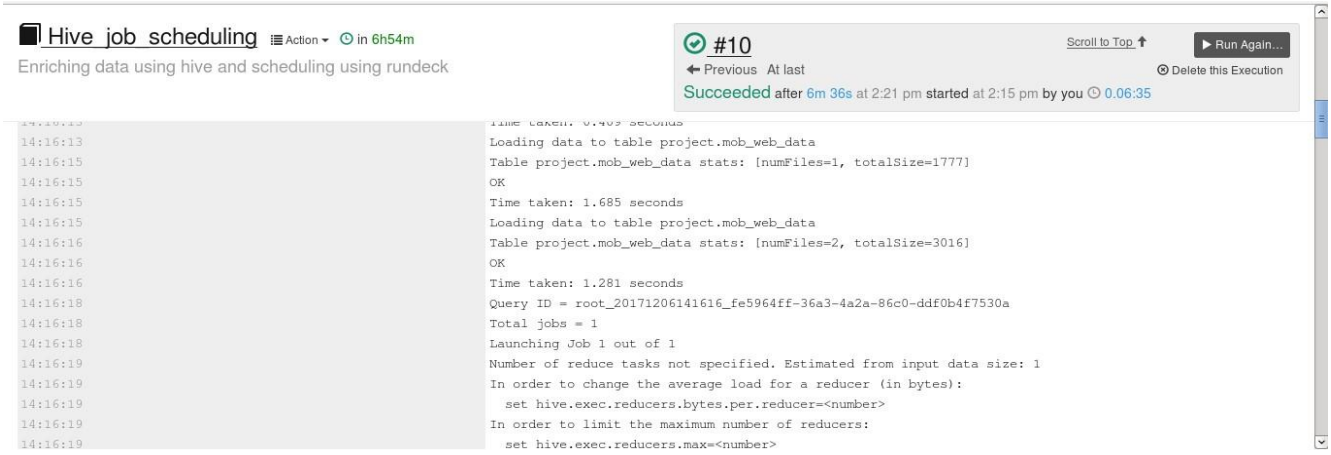
After clicking on Create, you will be redirected to the Job page. Beside your job, you can see the countdown left to run it.
You can also see your job definition in the Definition tab as shown below:





We can see the job is running and deployment number as 10.We can see that our job has run successfully. We can check for the output in Log Output tab. Here, you can see the console output for the jobs. Once the job runs successfully we can see the next deployment number countdown.

```
21:25:44  Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
21:25:59  OK
21:25:59  Time taken: 3.755 seconds
21:25:59  Added [/usr/lib/hive/lib/zookeeper.jar] to class path
21:25:59  Added resources: [/usr/lib/hive/lib/zookeeper.jar]
21:25:59  Added [/usr/lib/hive/lib/hive-hbase-handler.jar] to class path
21:25:59  Added resources: [/usr/lib/hive/lib/hive-hbase-handler.jar]
21:25:59  Added [/usr/lib/hive/lib/guava-14.0.1.jar] to class path
21:25:59  Added resources: [/usr/lib/hive/lib/guava-14.0.1.jar]
21:25:59  Added [/usr/lib/hive/lib/hbase-client.jar] to class path
21:25:59  Added resources: [/usr/lib/hive/lib/hbase-client.jar]
21:25:59  Added [/usr/lib/hive/lib/hbase-common.jar] to class path
21:25:59  Added resources: [/usr/lib/hive/lib/hbase-common.jar]
21:25:59  Added [/usr/lib/hive/lib/hbase-hadoop-compat.jar] to class path
21:25:59  Added resources: [/usr/lib/hive/lib/hbase-hadoop-compat.jar]
21:25:59  Added [/usr/lib/hive/lib/hbase-hadoop2-compat.jar] to class path
```



As from above screenshot we can see that our final validated records have been created.

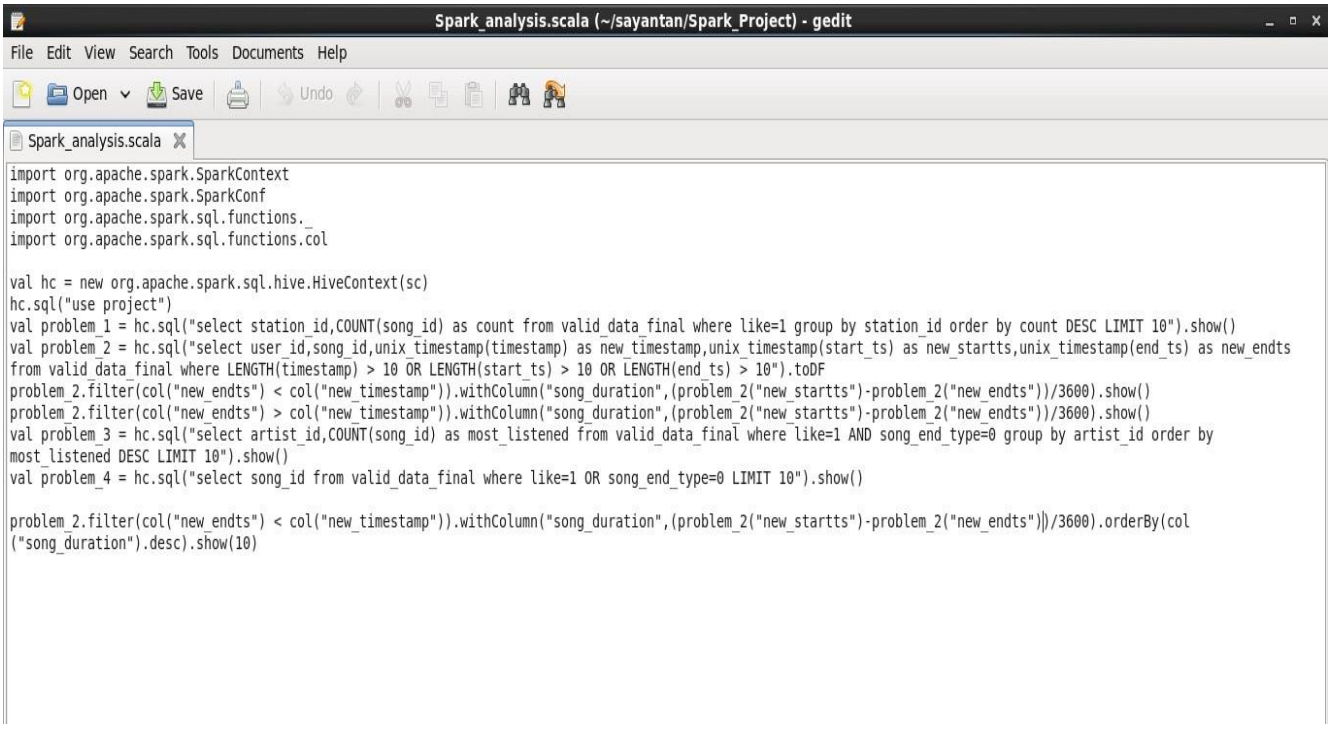
In this way we have automated our data validation and enrichment process.

The next and important step is to automate the analysis which is done is spark.

I am running my scala script in rundeck to automate my analysis in spark.

Command: **spark-shell –i file.scala**

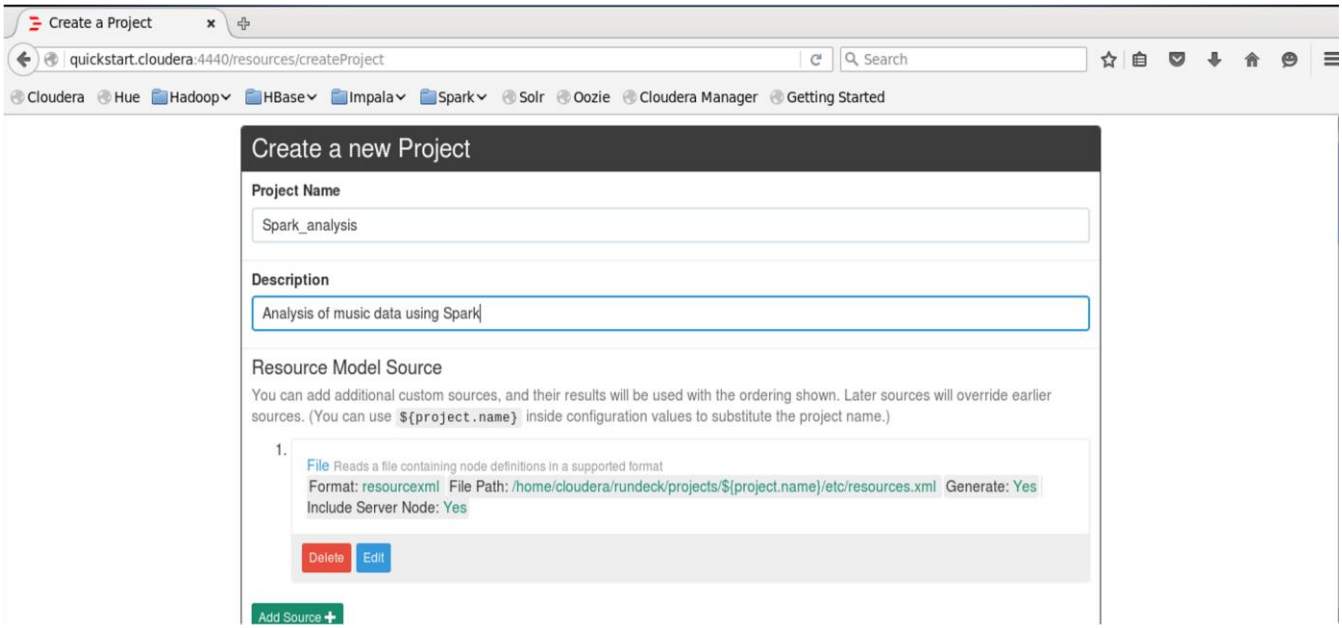
Here it will run the scala file inside a spark shell.
The scala file is as follows:

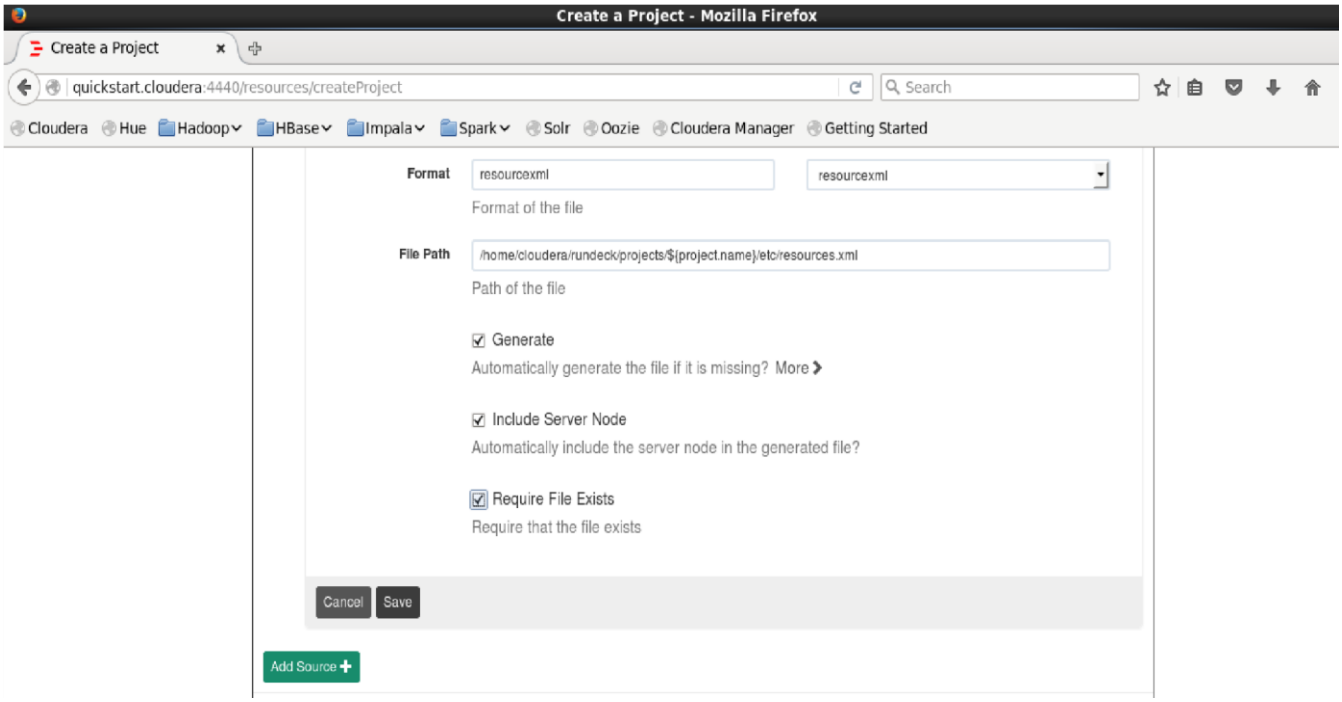


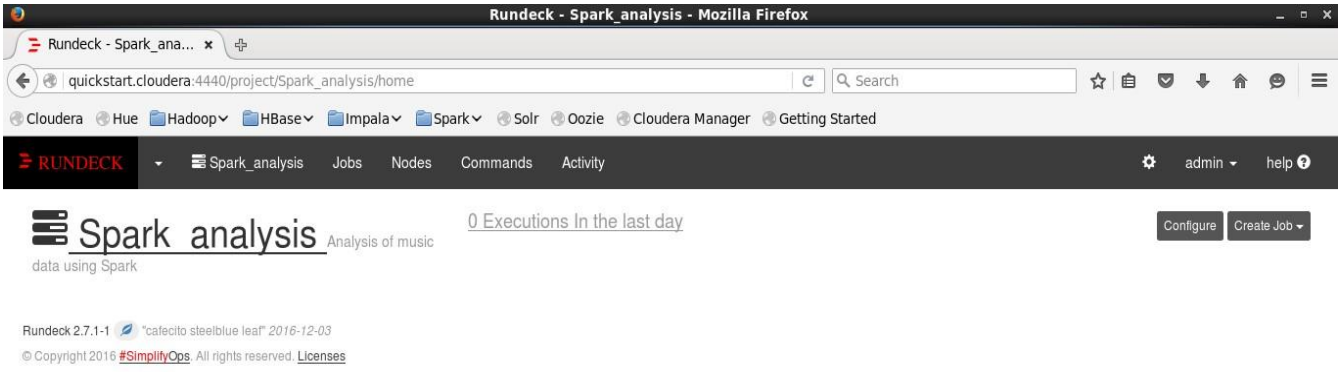
Inside this scala file I'm having all the spark commands that will be used for analysis.

We will run this scala script using the command **spark-shell –i file.scala**.

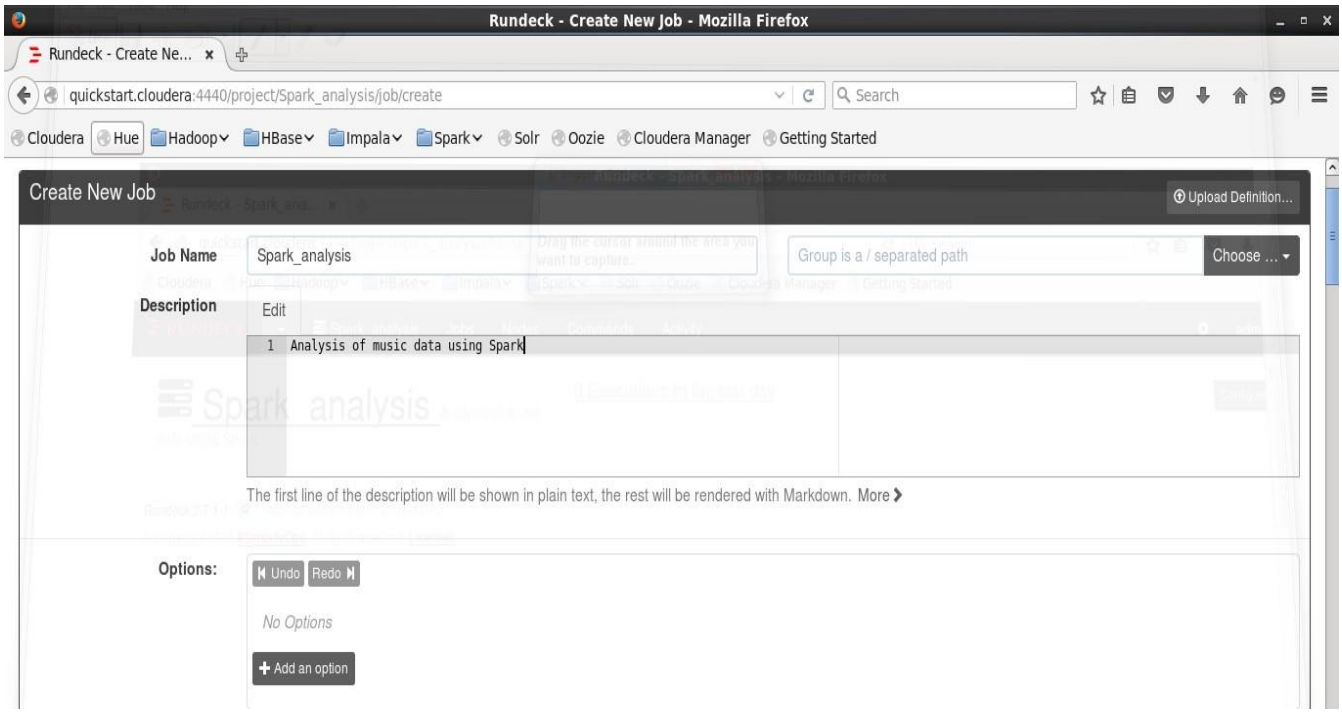
So now we will start rundeck a create a new project as we did in data enrichment process.



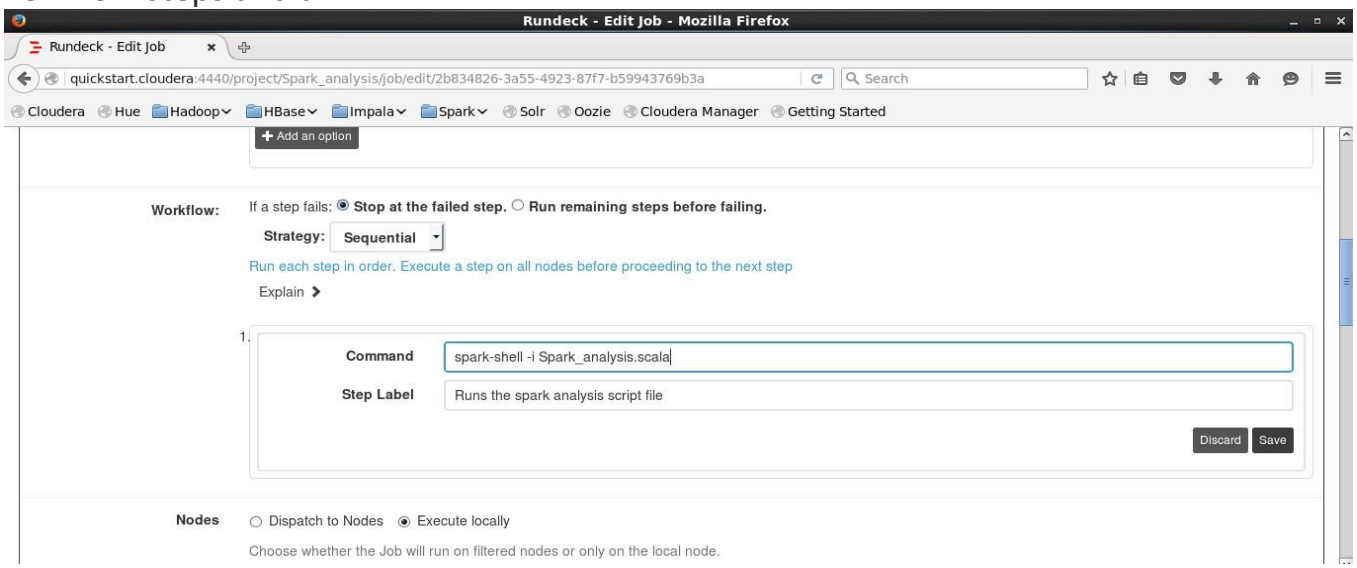


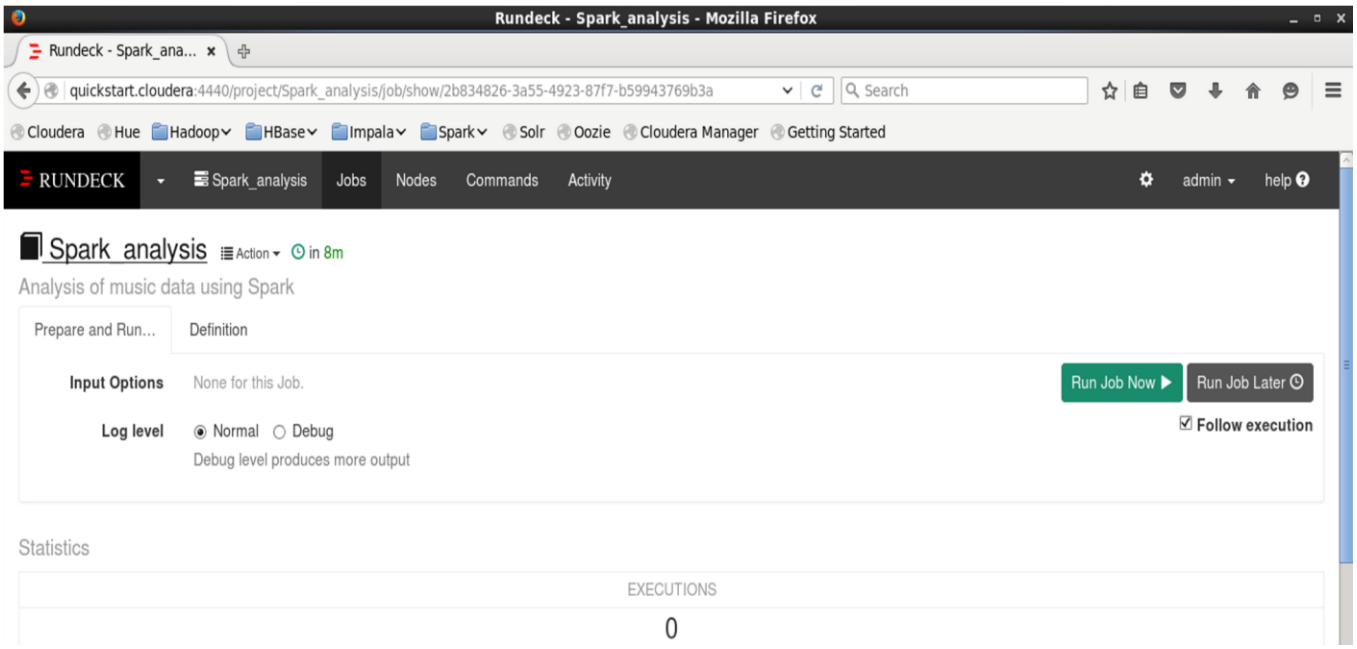


Click on Jobs and click on Create a new job



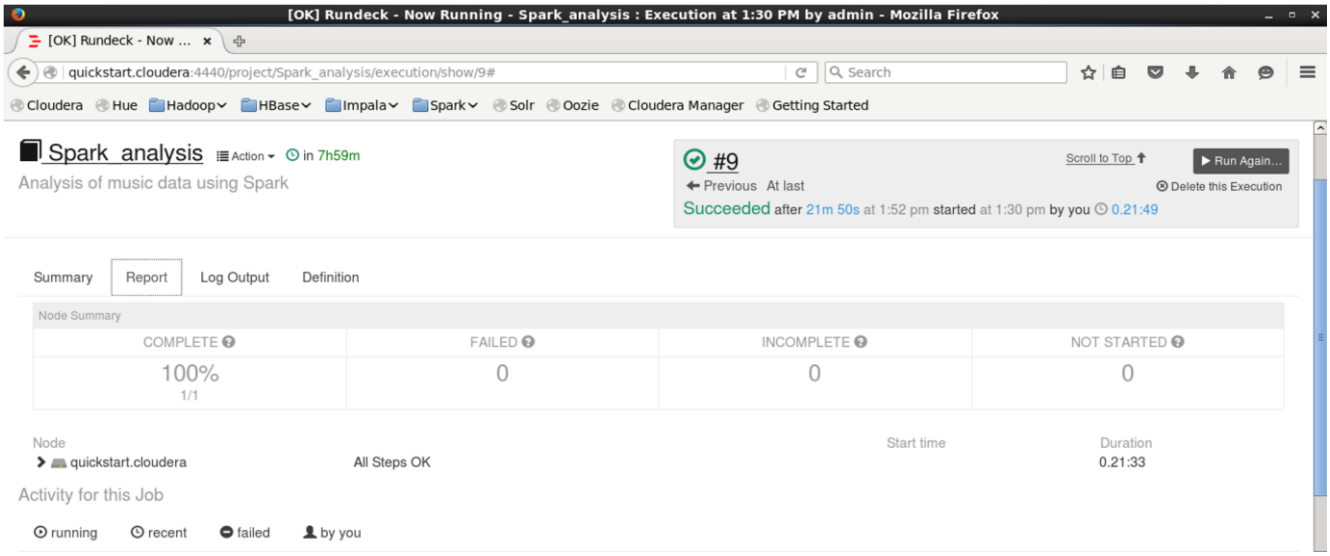
In the next screen provide the necessary details of your job like job name, job description, workflow steps and all.



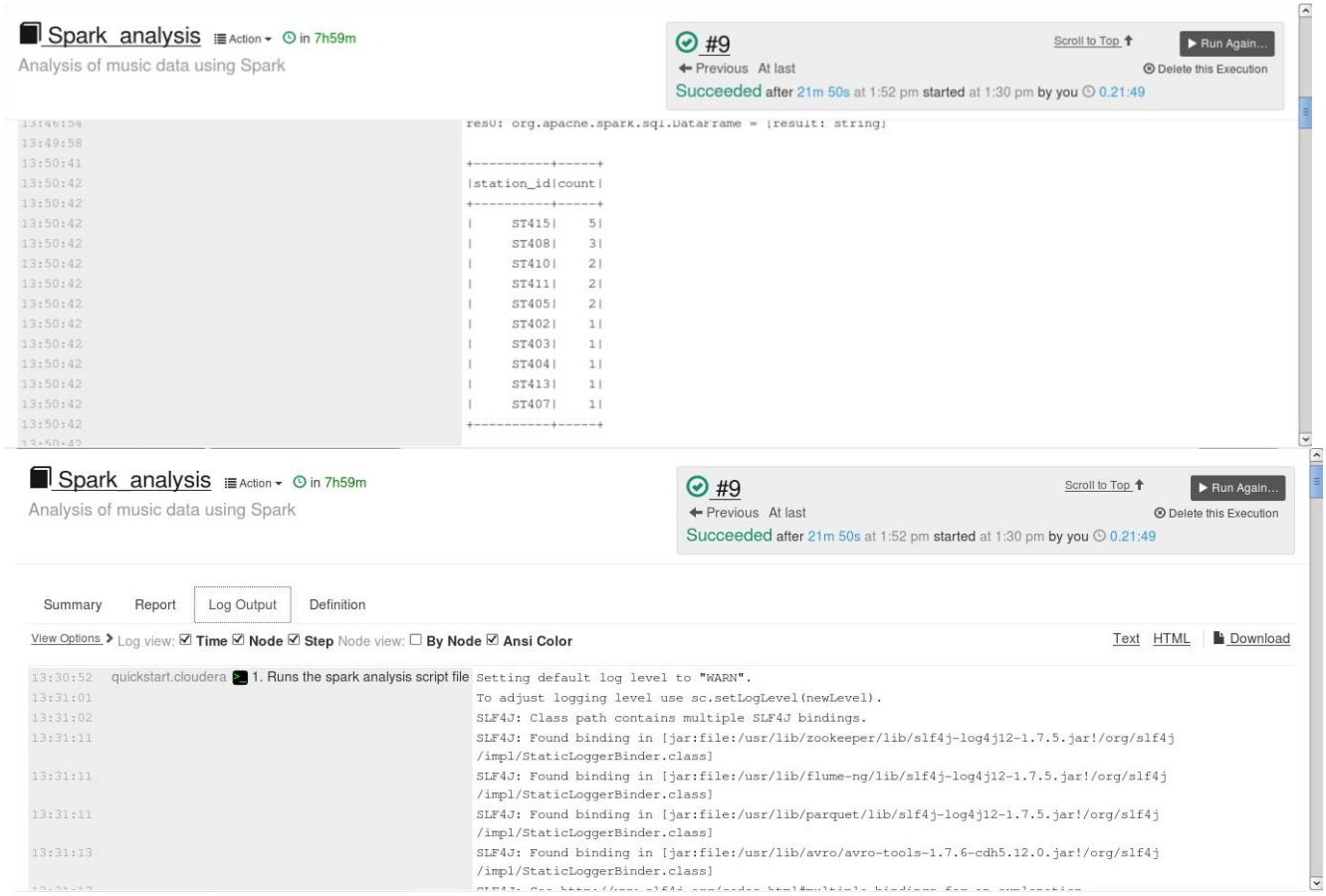


On the Right side, there are two options Run Job Now, Run Job Later. If you want to schedule this job to run after some time, you can set the time by clicking on Run Job Later. The job will run after 30mins of your scheduled time. Here we are clicking on Run Job Now.

After Running the job, you will get several options to monitor the job as shown below

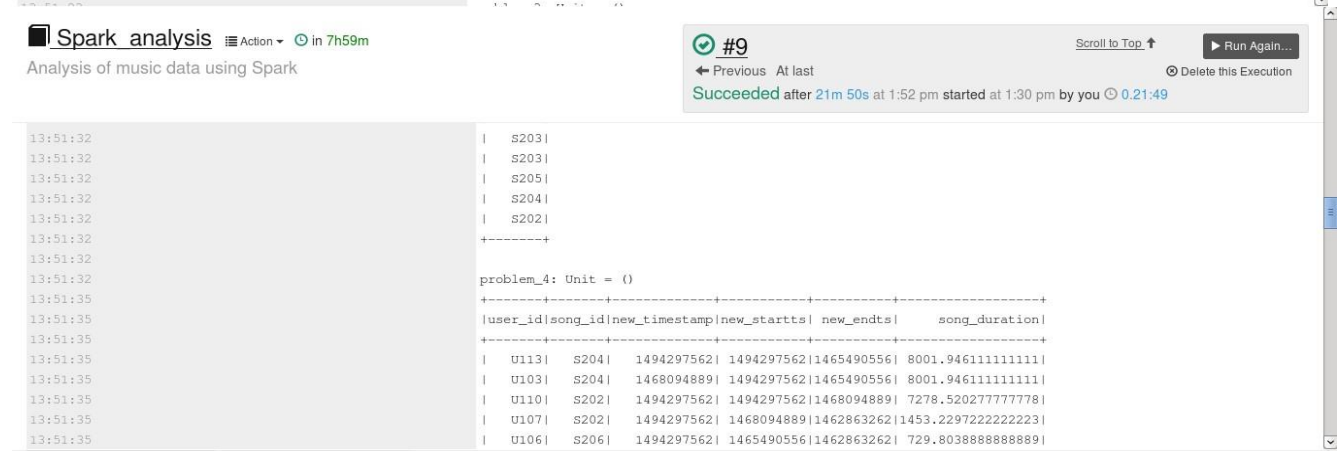


We can see that our job has run successfully. Click on Monitor to check the status of the job. After successful completion of the job, you can see a Report option over there. Here you can see the status of the job and the output of the console as shown in the below screenshots.




Spark_analysis
Action
in 7h59m
Analysis of music data using Spark
#9
Scroll to Top
Run Again...
Previous At last
Delete this Execution
Succeeded after 21m 50s at 1:52 pm started at 1:30 pm by you 0.21:49
new_startts: bigint, new_endts: bigint]
13:50:56
13:50:56 |user_id|song_id|new_timestamp|new_startts| new_endts| song_duration|
13:50:56
13:50:56 | U110| S202| 1494297562| 1494297562|1468094889| 7278.520277777778|
13:50:56 | U106| S206| 1494297562| 1465490556|1462863262| 729.8038888888889|
13:50:56 | U113| S203| 1465490556| 1465490556|1462863262| 729.8038888888889|
13:50:56 | U107| S202| 1494297562| 1468094889|1462863262|1453.2297222222223|
13:50:56 | U103| S204| 1468094889| 1494297562|1465490556| 8001.946111111111|
13:50:56 | U113| S204| 1494297562| 1494297562|1465490556| 8001.946111111111|
13:50:56
13:50:56
13:51:02
13:51:03 |user_id|song_id|new_timestamp|new_startts| new_endts| song_duration|
13:51:03
13:51:03 | U102| S203| 1465490556| 1465490556|1494297562|-8001.946111111111|


Spark_analysis
Action
in 7h59m
Analysis of music data using Spark
#9
Scroll to Top
Run Again...
Previous At last
Delete this Execution
Succeeded after 21m 50s at 1:52 pm started at 1:30 pm by you 0.21:49
13:51:03 | U109| S203| 1462863262| 1494297562|1468094889| 7278.520277777778|
13:51:03 | U106| S205| 1462863262| 1462863262|1494297562| -8731.75|
13:51:03 | U113| S203| 1462863262| 1468094889|1494297562|-7278.520277777778|
13:51:03 | U111| S204| 1465490556| 1465490556|1468094889|-723.4258333333333|
13:51:03 | U114| S209| 1465490556| 1462863262|1494297562| -8731.75|
13:51:03
13:51:22
13:51:22 |artist_id|most_listened|
13:51:23
13:51:23 | A300| 2|
13:51:23 | A302| 1|
13:51:23 | A305| 1|
13:51:23 | A303| 1|
13:51:23
13:51:23


Spark_analysis
Action
in 7h59m
Analysis of music data using Spark
#9
Scroll to Top
Run Again...
Previous At last
Delete this Execution
Succeeded after 21m 50s at 1:52 pm started at 1:30 pm by you 0.21:49
13:51:32 | S203|
13:51:32 | S203|
13:51:32 | S205|
13:51:32 | S204|
13:51:32 | S202|
13:51:32
13:51:32
13:51:32 problem_4: Unit = ()
13:51:35
13:51:35 |user_id|song_id|new_timestamp|new_startts| new_endts| song_duration|
13:51:35
13:51:35 | U113| S204| 1494297562| 1494297562|1465490556| 8001.946111111111|
13:51:35 | U103| S204| 1468094889| 1494297562|1465490556| 8001.946111111111|
13:51:35 | U110| S202| 1494297562| 1494297562|1468094889| 7278.520277777778|
13:51:35 | U107| S202| 1494297562| 1468094889|1462863262|1453.2297222222223|
13:51:35 | U106| S206| 1494297562| 1465490556|1462863262| 729.8038888888889|


Spark_analysis
Action
in 7h59m
Analysis of music data using Spark
#9
Scroll to Top
Run Again...
Previous At last
Delete this Execution
Succeeded after 21m 50s at 1:52 pm started at 1:30 pm by you 0.21:49
13:51:32
13:51:32
13:51:35
13:51:35
13:51:35
13:51:35
13:51:35
13:51:35
13:51:35
13:51:35
13:51:35
13:51:35
13:51:36
13:51:36
13:51:36
problem_4: Unit = ()
+-------+-------+-------------+-----------+----------+-----------------+
|user_id|song_id|new_timestamp|new_startts| new_endts| song_duration|
+-------+-------+-------------+-----------+----------+-----------------+
U113	S204	1494297562	1494297562	1465490556	8001.9461111111111
U103	S204	1468094889	1494297562	1465490556	8001.9461111111111
U110	S202	1494297562	1494297562	1468094889	7278.5202777777778
U107	S202	1494297562	1468094889	1462863262	1453.2297222222223
U106	S206	1494297562	1465490556	1462863262	729.8038888888889
U113	S203	1465490556	1465490556	1462863262	729.8038888888889
+-------+-------+-------------+-----------+----------+-----------------+
scala> Stopping spark context.