

A Project Report On
“Analysis on Banking Industry”

Submitted in partial fulfillment of requirements of the course

“Principles of Big Data Management”

For the degree of
Master's in Computer Science

By

Nandanamudi, Jyothikiran	(jnyc8@mail.umkc.edu)	(16225795)
Swargam, Gopi Krishna	(gs446@mail.umkc.edu)	(16222026)
Junaid, Sidrah	(sjhv6@mail.umkc.edu)	(16225745)

Under the esteemed guidance of

Dr. Praveen Rao



Abstract:

The banking industry has been a revolution over a period of time. Banks today are growing swiftly and are being adaptable to inevitable changes being occurred across technology, customer nature and regulation. Nowadays, modern banking sector is shifting towards to technological trends to reach more number of customers from day to day. As banking system becomes more user friendly, customers still covet of human connection. Today, customers are able to quickly raise their issues through social media network which is acting as a social interaction medium" which puts greater pressure on the banks to adapt with the new solutions and new business models.

Even Banks are using social media to a greater extent to monitor their brand identity, testing new products like credit cards, debit cards, flexi loans, personal loans, vehicle loans and many other products are in line as well banks are using social media to get feedback about this products. Banks use social media for conversations and building trust. Even, employment opportunities are being provided using social media.

There are billions of active users of social media networks like Twitter, Facebook etc., worldwide, who are frequently active and connected by means of smartphones, computer and tablets. The posts and blogs which are being posted daily might be structured or unstructured data. These big data are waiting to be explored and as we see the social media embodies the leading and biggest source of consumer data. With the good management of Big Data it can lead to many great insights; we can find the root cause of the problems and failure that affects the revenues of the business.

Twitter is a fabulous source of information. Whenever something is happening people around the world start tweeting away. Twitter's kind of a feasible place to look for big data. It's an example of the ubiquitous services used by consumers and businesses alike that helps to generate this huge amount of data in the first place.

There are a number of types of software tools for analyzing unstructured data found in tweets. [Apache Spark™](#) is a fast and general engine for large-scale data processing. Spark offers a wide variety such as 80 high level operators that make it easy to build parallel operating apps. And you can use it interactively from the Scala, Python and R shells and you can write applications quickly in Java, Scala, Python, R.

It is an open source big data accessing framework built around speed, ease of use, and sophisticated analytics. Spark enables applications in Hadoop clusters to run up to 100 times faster in memory and 10 times speed even when running on disk. In addition to Map and Reduce functions, it supports queries written in SQL, streaming data, machine learning and graph data processing.

Table of Contents

- I. Introduction
 - i. Analytical procedure
 - ii. Software interface
- II. Architectural Design
 - i. System Architecture
- III. Project Implementation
 - i. Source Code
 - ii. Analytical Queries and Visualization Results
 - a. Analytical Queries
 - b. Visualization Results.
 - iii. Other pages
- IV. Summary of the project
- V. Conclusion and links
- VI. References

I. Introduction:

Development of Analytical application for analyzing the twitter data collected for better understanding and for better decision making. We chose “Banking and other financial sectors” as our project concept for analyzing.

Banks find Twitter an indispensable tool for communications. Our project proposes eight different analytical queries on the Banking Industry on the data collected from twitter. One of the main feature of spark is speed and it has ability to run computations and operations in memory, but the system is also more efficient than MapReduce for complex applications running on disk.

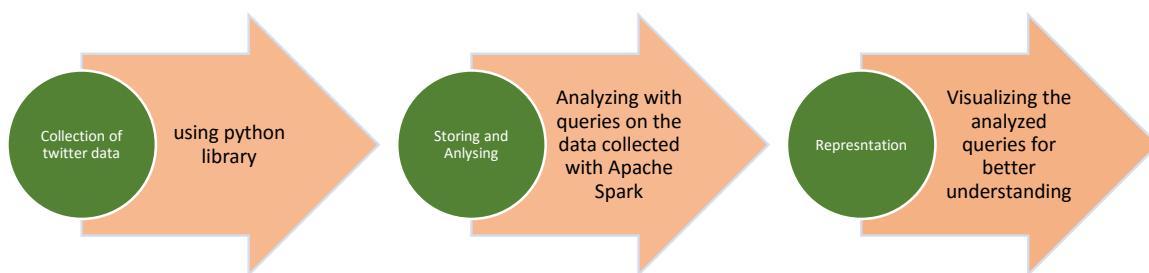
Spark SQL is Spark’s package made available for working with structured data. Spark SQL allows developers to mix SQL queries with the programmatic data manipulations which are supported by RDDs in Python, Java, and Scala, all within a single application, thus combining SQL with complex analytics. We used Spark SQL to write analytical queries in Java.

Big Data is something of a buzzword, but the data can be so complex and it is incomprehensible. Data visualization makes it easy to understand as it distill the lots of information into human understandable form typically focusing on the aggregates of the data. We made use of d3.js for visualization of our proposed analytical queries.

Our project draws few insights from the available data of the twitter based on the tweets posted by banks, their users and by common people who wants to share their opinion about the banking products and banking industry.

i. Analytical procedure

The sequence followed in analyzing the data is as shown below,



We used python streaming library to collect data being posted on banks in valid JSON format and is stored into a text file. We have collected almost 1200Mb of data for our analysis. We used spark locally in our Java work space for writing analytical queries on the collected data and has been visualized with d3.js.

ii. Software interface

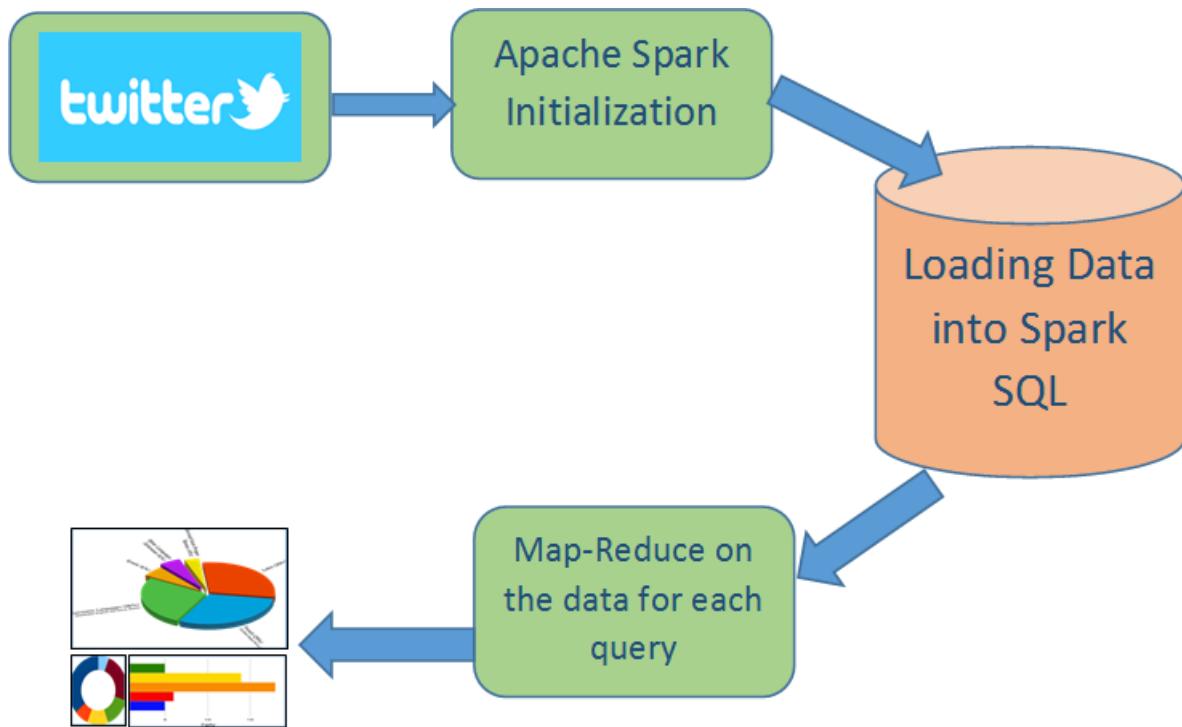
1. Client side: HTML, HTML5, JavaScript, d3.js, high charts
2. Framework: Apache Spark, Eclipse modelling framework.
3. Date base: Spark SQL

II. Architectural Design

The overall design of our project is shown in steps.

- Collection of data
- Load our input data.
- Parse our input into words.
- Reduce our words into a tuple pair that contains the word and the count of occurrences.
- Save our results.
- Visualizing the results saved with charts and graphs.

The above steps is implemented as in the below architecture diagram.



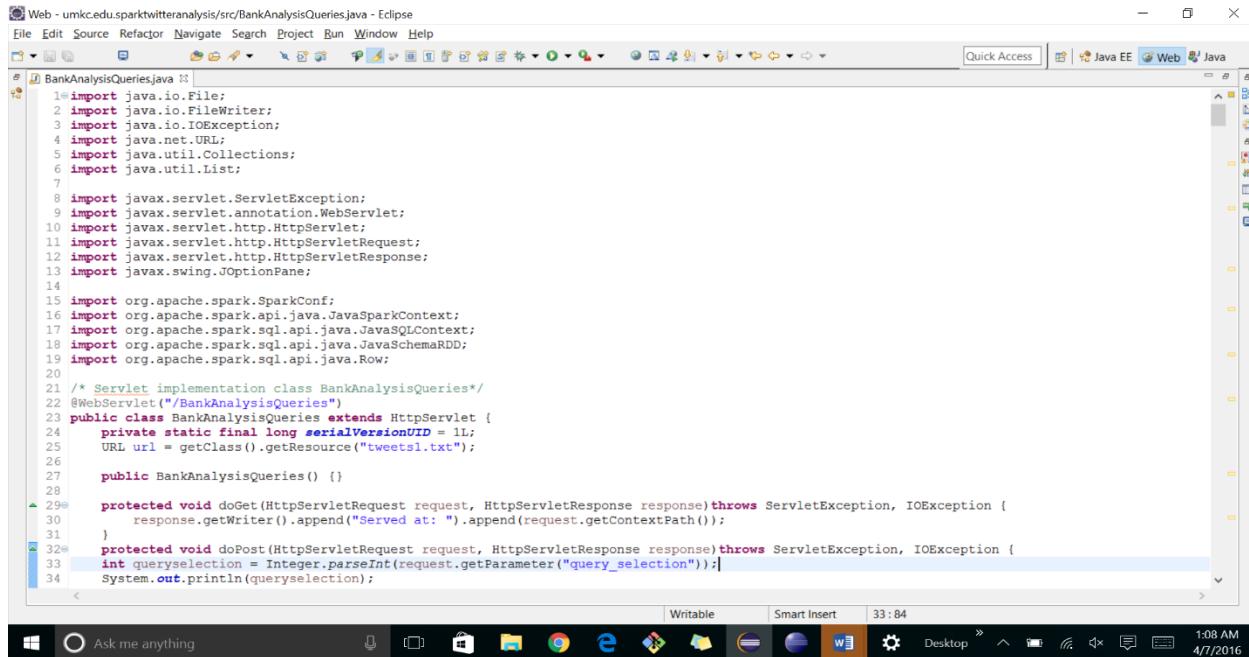
As represented above, tweets collected will be loaded into Apache Spark and then stored into Spark SQL. We will run queries which will execute map-reduce on the twitter data and will display the result.

III. Project Implementation

As spark supports different programming languages Java is our choice. We used eclipse framework for implementing the Analytics using Apache Spark.

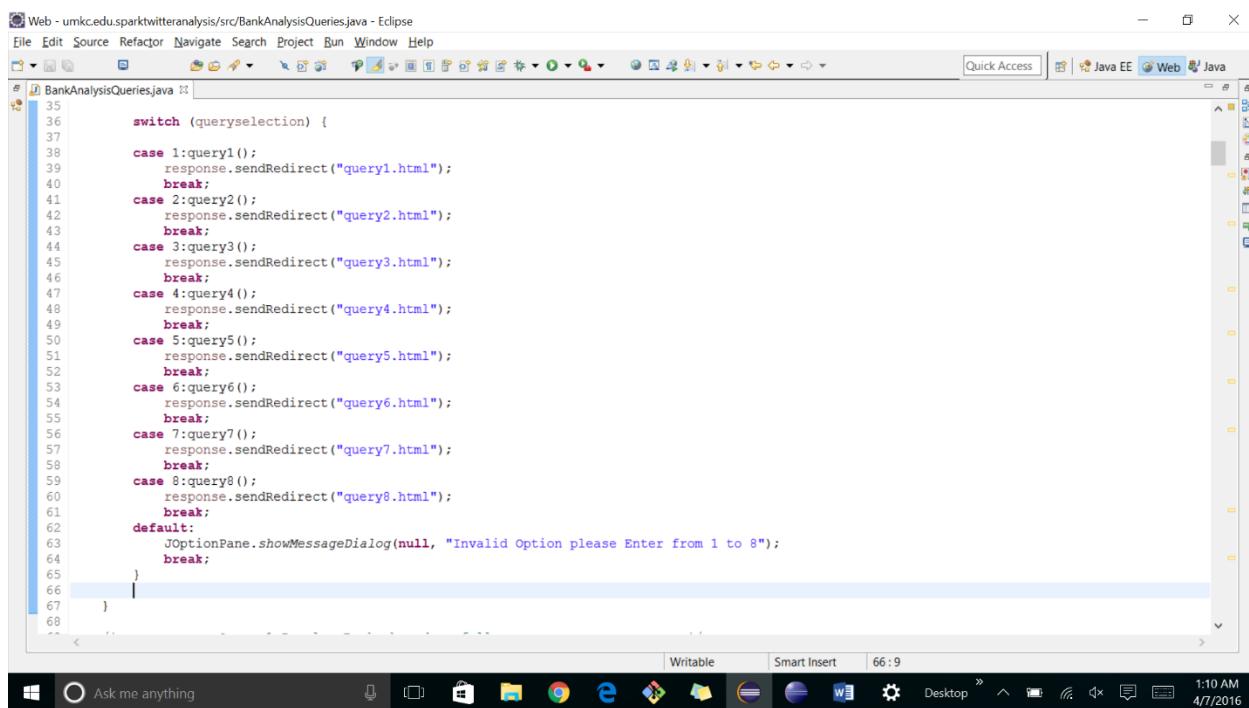
We used Spark locally in our Java process space. We created SparkConf object which points to our Spark instance as in this case 'local' and in order to interact with Spark we need a SparkContext instance, so we created JavaSparkContext. We stored the tweets in Spark SQL database under tweetable and sql query is run on this table for selection of required data for analytics.

a. Source Code



Web - umkc.edu.sparktwitteranalysis/src/BankAnalysisQueries.java - Eclipse

```
1 import java.io.File;
2 import java.io.FileWriter;
3 import java.io.IOException;
4 import java.net.URL;
5 import java.util.Collections;
6 import java.util.List;
7
8 import javax.servlet.ServletException;
9 import javax.servlet.annotation.WebServlet;
10 import javax.servlet.http.HttpServlet;
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13 import javax.swing.JOptionPane;
14
15 import org.apache.spark.SparkConf;
16 import org.apache.spark.api.java.JavaSparkContext;
17 import org.apache.spark.sql.api.java.JavaSQLContext;
18 import org.apache.spark.sql.api.java.JavaSchemaRDD;
19 import org.apache.spark.sql.api.java.Row;
20
21 /* Servlet implementation class BankAnalysisQueries*/
22 @WebServlet("/BankAnalysisQueries")
23 public class BankAnalysisQueries extends HttpServlet {
24     private static final long serialVersionUID = 1L;
25     URL url = getClass().getResource("tweets1.txt");
26
27     public BankAnalysisQueries() {}
28
29     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
30         response.getWriter().append("Served at: ").append(request.getContextPath());
31     }
32     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
33         int queryselection = Integer.parseInt(request.getParameter("query_selection"));
34         System.out.println(queryselection);
35     }
36
37     switch (queryselection) {
38
39         case 1:query1();
40             response.sendRedirect("query1.html");
41             break;
42         case 2:query2();
43             response.sendRedirect("query2.html");
44             break;
45         case 3:query3();
46             response.sendRedirect("query3.html");
47             break;
48         case 4:query4();
49             response.sendRedirect("query4.html");
50             break;
51         case 5:query5();
52             response.sendRedirect("query5.html");
53             break;
54         case 6:query6();
55             response.sendRedirect("query6.html");
56             break;
57         case 7:query7();
58             response.sendRedirect("query7.html");
59             break;
60         case 8:query8();
61             response.sendRedirect("query8.html");
62             break;
63         default:
64             JOptionPane.showMessageDialog(null, "Invalid Option please Enter from 1 to 8");
65             break;
66     }
67 }
```



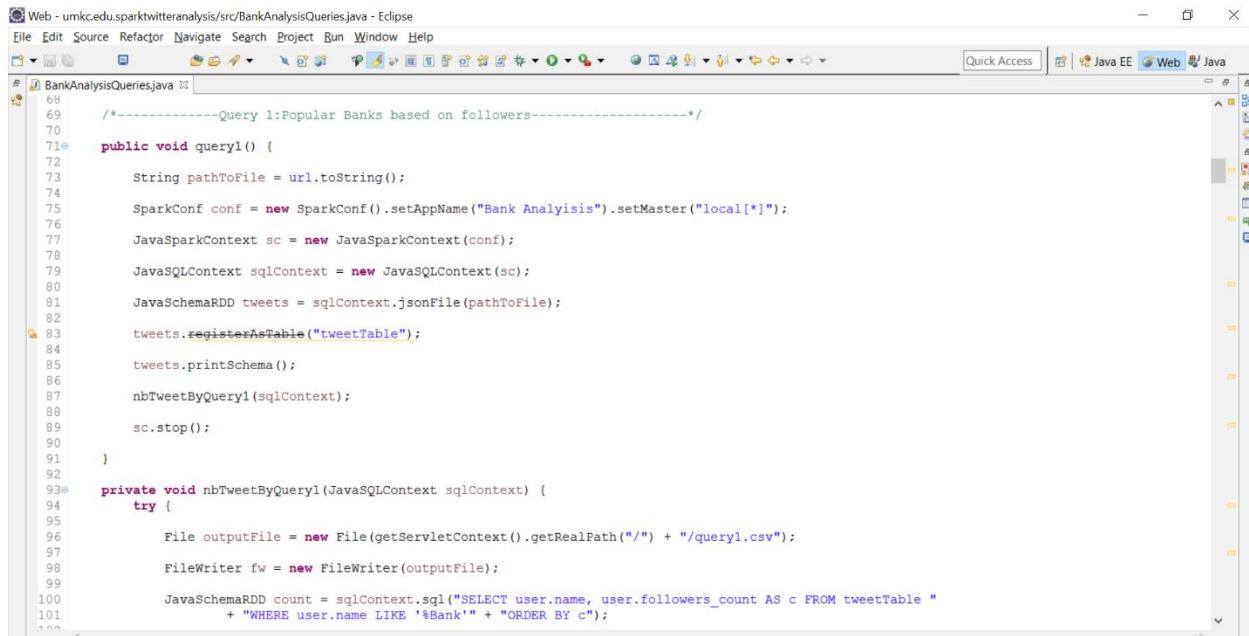
Web - umkc.edu.sparktwitteranalysis/src/BankAnalysisQueries.java - Eclipse

```
35
36     switch (queryselection) {
37
38         case 1:query1();
39             response.sendRedirect("query1.html");
40             break;
41         case 2:query2();
42             response.sendRedirect("query2.html");
43             break;
44         case 3:query3();
45             response.sendRedirect("query3.html");
46             break;
47         case 4:query4();
48             response.sendRedirect("query4.html");
49             break;
50         case 5:query5();
51             response.sendRedirect("query5.html");
52             break;
53         case 6:query6();
54             response.sendRedirect("query6.html");
55             break;
56         case 7:query7();
57             response.sendRedirect("query7.html");
58             break;
59         case 8:query8();
60             response.sendRedirect("query8.html");
61             break;
62         default:
63             JOptionPane.showMessageDialog(null, "Invalid Option please Enter from 1 to 8");
64             break;
65     }
66 }
```

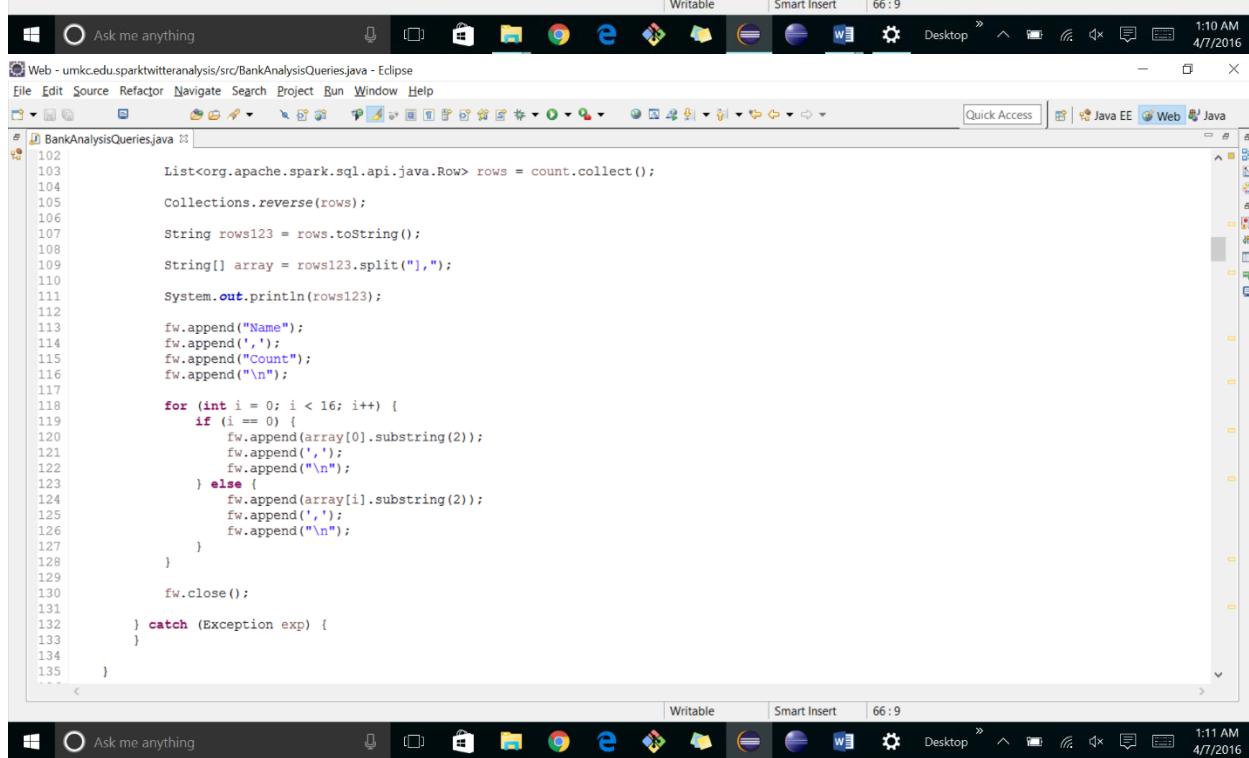
a. Analytical Queries:

Query 1: Popular Banks internationally based on their followers

As there are lot of banks all over the world, people often would like to know the updates of some particular banks and often comment on them and usually follow some banks. So we did analysis based on this followers count and selected top banks.



```
Web - umkc.edu.sparkwitteranalysis/src/BankAnalysisQueries.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java EE Web Java
BankAnalysisQueries.java
68  /*
69   *-----Query 1:Popular Banks based on followers-----
70
71  public void query1() {
72
73      String pathToFile = url.toString();
74
75      SparkConf conf = new SparkConf().setAppName("Bank Analysis").setMaster("local[*]");
76
77      JavaSparkContext sc = new JavaSparkContext(conf);
78
79      JavaSQLContext sqlContext = new JavaSQLContext(sc);
80
81      JavaSchemaRDD tweets = sqlContext.jsonFile(pathToFile);
82
83      tweets.registerAsTable("tweetTable");
84
85      tweets.printSchema();
86
87      nbTweetByQuery1(sqlContext);
88
89      sc.stop();
90
91  }
92
93  private void nbTweetByQuery1(JavaSQLContext sqlContext) {
94      try {
95
96          File outputFile = new File(getServletContext().getRealPath("/") + "/query1.csv");
97
98          FileWriter fw = new FileWriter(outputFile);
99
100         JavaSchemaRDD count = sqlContext.sql("SELECT user.name, user.followers_count AS c FROM tweetTable "
101             + "WHERE user.name LIKE '%Bank'" + "ORDER BY c");
102
103         List<org.apache.spark.sql.api.java.Row> rows = count.collect();
104
105         Collections.reverse(rows);
106
107         String rows123 = rows.toString();
108
109         String[] array = rows123.split(",\"");
110
111         System.out.println(rows123);
112
113         fw.append("Name");
114         fw.append(",");
115         fw.append("count");
116         fw.append("\n");
117
118         for (int i = 0; i < 16; i++) {
119             if (i == 0) {
120                 fw.append(array[0].substring(2));
121                 fw.append(',');
122                 fw.append("\n");
123             } else {
124                 fw.append(array[i].substring(2));
125                 fw.append(',');
126                 fw.append("\n");
127             }
128         }
129
130         fw.close();
131
132     } catch (Exception exp) {
133     }
134
135 }
```



```
Web - umkc.edu.sparkwitteranalysis/src/BankAnalysisQueries.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java EE Web Java
BankAnalysisQueries.java
102
103     List<org.apache.spark.sql.api.java.Row> rows = count.collect();
104
105     Collections.reverse(rows);
106
107     String rows123 = rows.toString();
108
109     String[] array = rows123.split(",\"");
110
111     System.out.println(rows123);
112
113     fw.append("Name");
114     fw.append(",");
115     fw.append("count");
116     fw.append("\n");
117
118     for (int i = 0; i < 16; i++) {
119         if (i == 0) {
120             fw.append(array[0].substring(2));
121             fw.append(',');
122             fw.append("\n");
123         } else {
124             fw.append(array[i].substring(2));
125             fw.append(',');
126             fw.append("\n");
127         }
128     }
129
130     fw.close();
131
132 } catch (Exception exp) {
133 }
134
135 }
```

Query 2: Most interested type of loan

As we know banks are the sources of financial resources in terms of providing loans for the people when there is a need and often provides different offers on these loans. People do often tweet about these loans and want to know about these loans. So we did analysis on the most popular loans offered by the banks and what type of loans are most popular over the twitter.

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

- Title Bar:** Web - umkc.edu.sparktwitteranalysis/src/BankAnalysisQueries.java - Eclipse
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Includes icons for file operations, search, and navigation.
- Quick Access:** A button for quick access to frequently used files.
- Java EE:** A button for Java Enterprise Edition tools.
- Web:** A button for web development tools.
- Java:** A button for Java development tools.
- Code Editor:** Displays the Java code for `BankAnalysisQueries.java`. The code implements a `query2()` method to count tweets containing specific loan types. It uses `SparkConf`, `JavaSparkContext`, and `JavaSQLContext` to process JSON data from a file and register it as a table named `tweetTable`. It then performs a query to count rows where the text contains "%car loan%" and writes the result to a CSV file named `query2.csv`.
- Right-hand Side:** Shows the Java EE perspective with various toolbars and views.

```
137  /*-----Query 2:Most Interested Type of Loan-----*/
138
139
140 public void query2() {
141
142     String pathToFile = url.toString();
143
144     SparkConf conf = new SparkConf().setAppName("Bank Analysis").setMaster("local[*]");
145
146     JavaSparkContext sc = new JavaSparkContext(conf);
147
148     JavaSQLContext sqlContext = new JavaSQLContext(sc);
149
150     JavaSchemaRDD tweets = sqlContext.jsonFile(pathToFile);
151
152     tweets.registerAsTable("tweetTable");
153
154     tweets.printSchema();
155
156     nbTweetByQuery2(sqlContext);
157
158     sc.stop();
159
160 }
161
162 private void nbTweetByQuery2(JavaSQLContext sqlContext) {
163
164     try {
165         File outputFile = new File(getServletContext().getRealPath("/") + "/query2.csv");
166
167         FileWriter fw = new FileWriter(outputFile);
168
169         JavaSchemaRDD count = sqlContext.sql("SELECT COUNT(*) AS c FROM tweetTable " + "WHERE text LIKE '%car loan%'");
170     }
171 }
```

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

- Title Bar:** "Web - umkc.edu.sparkwitteranalysis/src/BankAnalysisQueries.java - Eclipse".
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Toolbar:** Includes icons for New, Open, Save, Cut, Copy, Paste, Find, Replace, and various project management tools.
- Quick Access Bar:** Contains icons for Java EE, Web, and Java.
- Code Editor:** Displays the Java code for `BankAnalysisQueries.java`. The code uses Scala-like syntax to perform SQL queries on a DataFrame and extract specific columns for different loan types (home, education, personal, mortgages).
- Right-hand View:** Shows the Project Explorer, Package Explorer, and Navigator.
- Bottom Status Bar:** Shows "Writable", "Smart Insert", and the current time "171:13".

```
L/0
171     JavaSchemaRDD count1 = sqlContext
172         .sql("SELECT COUNT(*) AS c FROM tweetTable " + "WHERE text LIKE '%home loan%'");
173     JavaSchemaRDD count2 = sqlContext
174         .sql("SELECT COUNT(*) AS c FROM tweetTable " + "WHERE text LIKE '%education loan%'");
175     JavaSchemaRDD count3 = sqlContext
176         .sql("SELECT COUNT(*) AS c FROM tweetTable " + "WHERE text LIKE '%personal loan%'");
177     JavaSchemaRDD count4 = sqlContext
178         .sql("SELECT COUNT(*) AS c FROM tweetTable " + "WHERE text LIKE '%mortgages%'");
179
180     List<Row> car = count1.collect();
181     String car12 = car.toString();
182     String car1 = car12.substring(car12.indexOf("[") + 2, car12.indexOf("]")));
183
184     List<Row> home = count2.collect();
185     String home12 = home.toString();
186     String home1 = home12.substring(home12.indexOf("[") + 2, home12.indexOf("]"));
187
188     List<Row> education = count3.collect();
189     String education12 = education.toString();
190     String education1 = education12.substring(education12.indexOf("[") + 2, education12.indexOf("]"));
191
192     List<Row> personal = count4.collect();
193     String personal12 = personal.toString();
194     String personal1 = personal12.substring(personal12.indexOf("[") + 2, personal12.indexOf("]"));
195
196     List<Row> mortgages = mortgages.collect();
197     String mortgages12 = mortgages.toString();
198     String mortgages1 = mortgages12.substring(mortgages12.indexOf("[") + 2, mortgages12.indexOf("]"));
199
200     fw.append("LoanType");
201     fw.append(',');
202     fw.append("Count");
203     fw.append("\n");
2..
```

Web - umkc.edu.sparktwitteranalysis/src/BankAnalysisQueries.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access Java EE Web Java

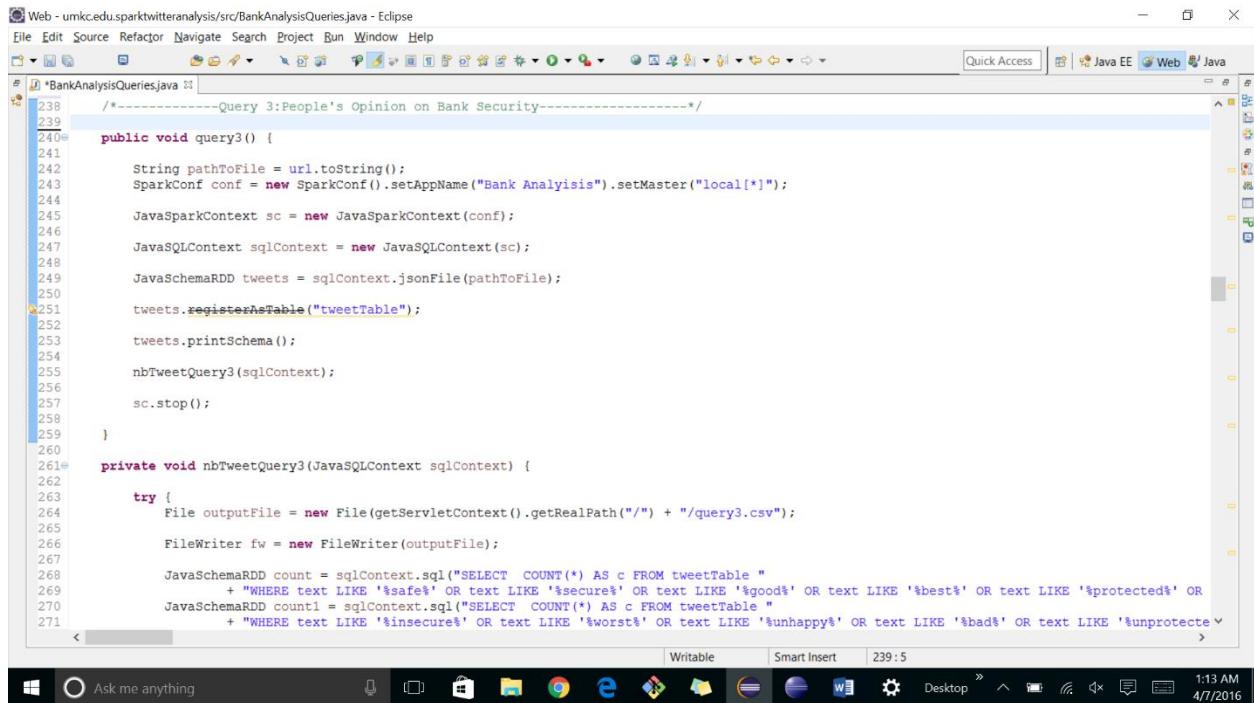
```
204     fw.append("car");
205     fw.append(',');
206     fw.append(car1);
207     fw.append("\n");
208     fw.append("home");
209     fw.append(',');
210     fw.append(home1);
211     fw.append("\n");
212     fw.append("education");
213     fw.append(',');
214     fw.append(education1);
215     fw.append("\n");
216     fw.append("personal");
217     fw.append(',');
218     fw.append(personal1);
219     fw.append("\n");
220     fw.append("mortgages");
221     fw.append(',');
222     fw.append(mortgages1);
223     fw.append("\n");
224
225     fw.close();
226
227 } catch (Exception exp) {
228 }
229
230 }
231
232
233
234
235
236
237 <
```

Writable Smart Insert 236:5

Ask me anything 1:12 AM 4/7/2016

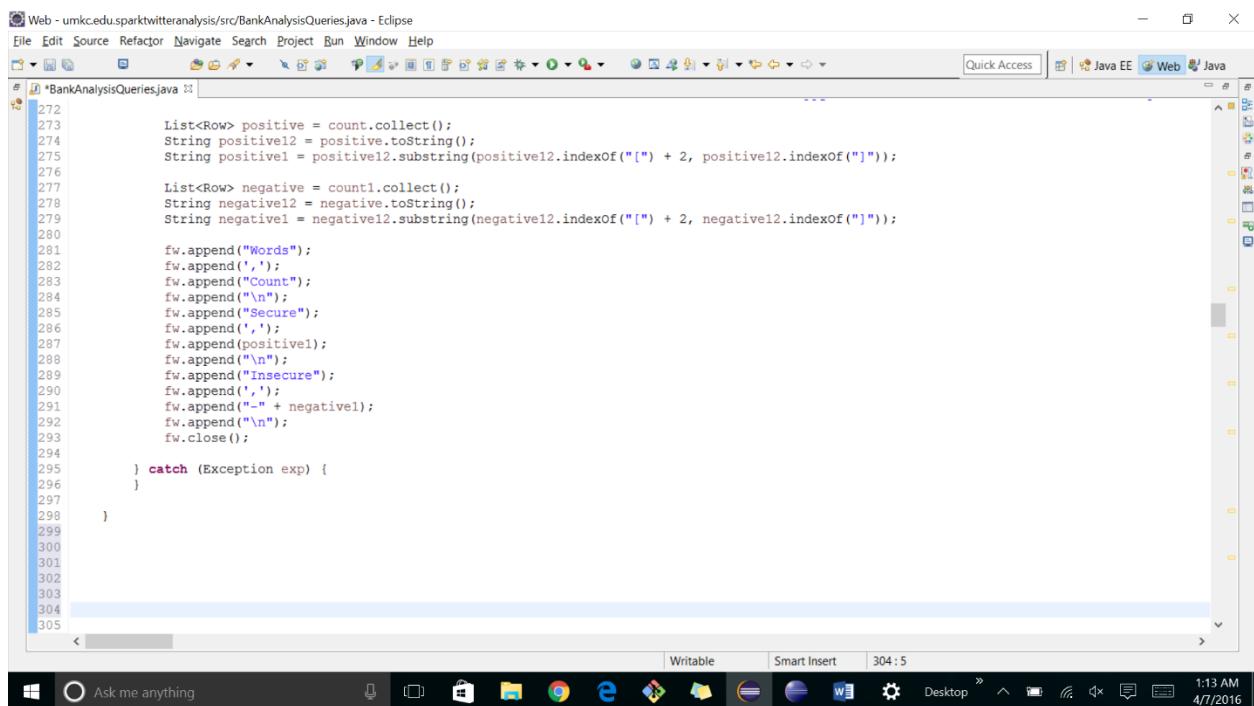
Query 3: People's Opinion on Bank Security

As there is a mixed opinion on banks when comes to security, service etc., from the customers as well as people. When we heard of robbery or fraud at a bank people would like to share and express their feelings. And banks do often reach their customers to ensure their relation. So, we did sentimental analysis based on the tweets collected i.e., what people think most about banks.



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

- Title Bar:** Web - umkc.edu.sparktwitteranalysis/src/BankAnalysisQueries.java - Eclipse
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar items.
- Quick Access:** Shows Java EE, Web, Java icons.
- Code Editor:** Displays Java code for "BankAnalysisQueries.java". The code implements a query to analyze bank security opinions from tweets. It uses JavaSparkContext and JavaSQLContext to interact with a tweetTable, counts words like 'safe', 'secure', 'good', 'best', 'protected', 'worst', 'unprotect', 'insecure', and 'unhappy', and then prints the results to a CSV file.
- Bottom Status Bar:** Shows Writable, Smart Insert, and the current line number (239:5).
- Taskbar:** Shows the Windows taskbar with various pinned icons and the system tray.
- System Tray:** Shows the date and time (4/7/2016, 1:13 AM).

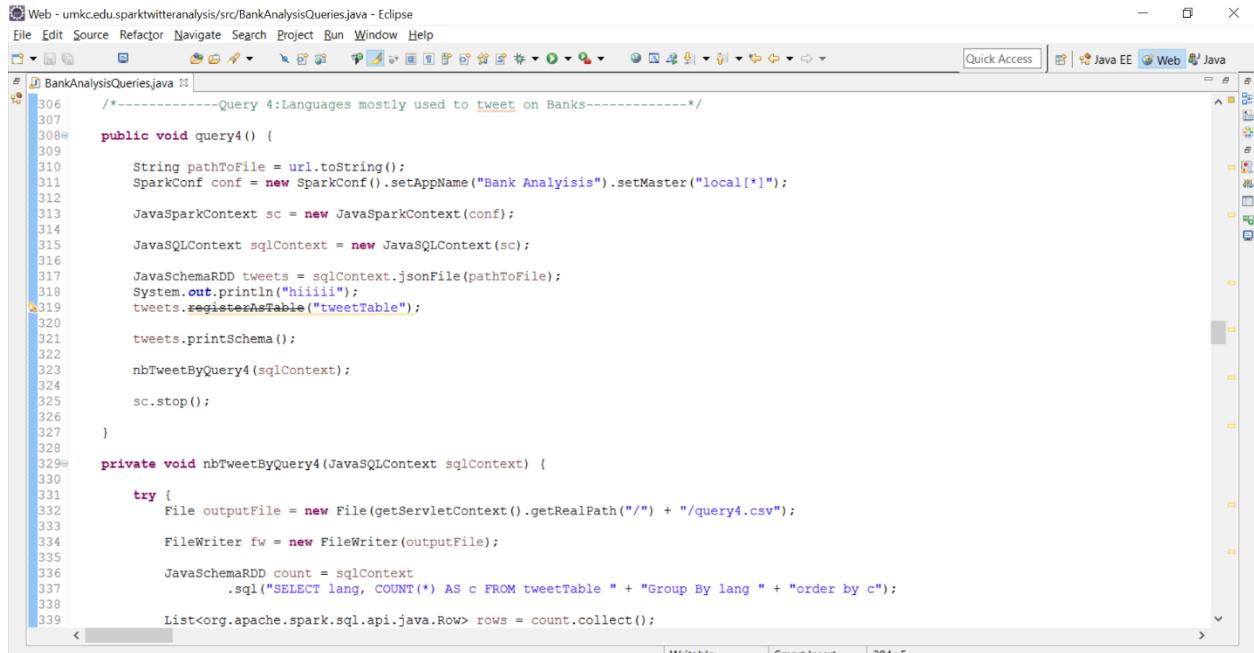


The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Web - umkc.edu.sparktwitteranalysis/src/BankAnalysisQueries.java - Eclipse
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar items.
- Quick Access:** Shows Java EE, Web, Java icons.
- Code Editor:** Displays Java code for "BankAnalysisQueries.java". This part of the code handles the output of the sentiment analysis. It reads the collected positive and negative counts from the previous code, formats them into a CSV-like string with headers 'Words', 'count', and 'positive/negative', and then writes this data to a file named 'query3.csv' using a FileWriter.
- Bottom Status Bar:** Shows Writable, Smart Insert, and the current line number (304:5).
- Taskbar:** Shows the Windows taskbar with various pinned icons and the system tray.
- System Tray:** Shows the date and time (4/7/2016, 1:13 AM).

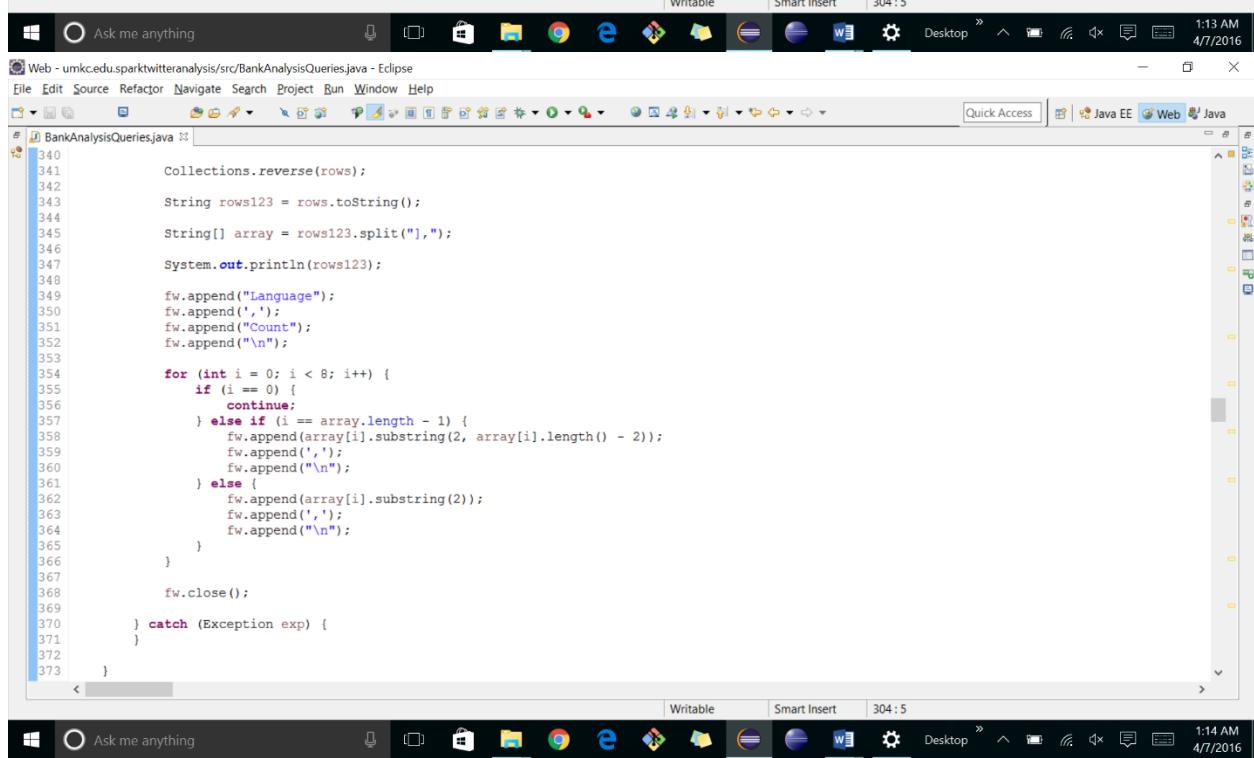
Query 4: Languages mostly used to tweet about banks

Twitter is social network which is being used almost worldwide. So, people of different areas uses different languages to tweet about banks. So we did analysis based on the language for finding the most used languages to tweet by the people all over the world.



The screenshot shows the Eclipse IDE interface with the code editor open. The file is named 'BankAnalysisQueries.java'. The code implements a query to find the most used languages for tweets about banks. It uses SparkContext, JavaSQLContext, and JavaSchemaRDD. The code includes a main method 'query4' and a private helper method 'nbTweetByQuery4'. The code is annotated with line numbers and comments explaining the logic.

```
306     /*-----Query 4:Languages mostly used to tweet on Banks-----*/
307
308     public void query4() {
309
310         String pathToFile = url.toString();
311         SparkConf conf = new SparkConf().setAppName("Bank Analysis").setMaster("local[*]");
312
313         JavaSparkContext sc = new JavaSparkContext(conf);
314
315         JavaSQLContext sqlContext = new JavaSQLContext(sc);
316
317         JavaSchemaRDD tweets = sqlContext.jsonFile(pathToFile);
318         System.out.println("Hiiiii");
319         tweets.registerAsTable("tweetTable");
320
321         tweets.printSchema();
322
323         nbTweetByQuery4(sqlContext);
324
325         sc.stop();
326
327     }
328
329     private void nbTweetByQuery4(JavaSQLContext sqlContext) {
330
331         try {
332             File outputFile = new File(getServletContext().getRealPath("/") + "/query4.csv");
333
334             FileWriter fw = new FileWriter(outputFile);
335
336             JavaSchemaRDD count = sqlContext
337                 .sql("SELECT lang, COUNT(*) AS c FROM tweetTable " + "Group By lang " + "order by c");
338
339             List<org.apache.spark.sql.api.java.Row> rows = count.collect();
340
341             Collections.reverse(rows);
342
343             String rows123 = rows.toString();
344
345             String[] array = rows123.split("[", "]");
346
347             System.out.println(rows123);
348
349             fw.append("Language");
350             fw.append(",");
351             fw.append("count");
352             fw.append("\n");
353
354             for (int i = 0; i < 8; i++) {
355                 if (i == 0) {
356                     continue;
357                 } else if (i == array.length - 1) {
358                     fw.append(array[i].substring(2, array[i].length() - 2));
359                     fw.append(',');
360                     fw.append("\n");
361                 } else {
362                     fw.append(array[i].substring(2));
363                     fw.append(',');
364                     fw.append("\n");
365                 }
366             }
367
368             fw.close();
369
370         } catch (Exception exp) {
371
372     }
373 }
```

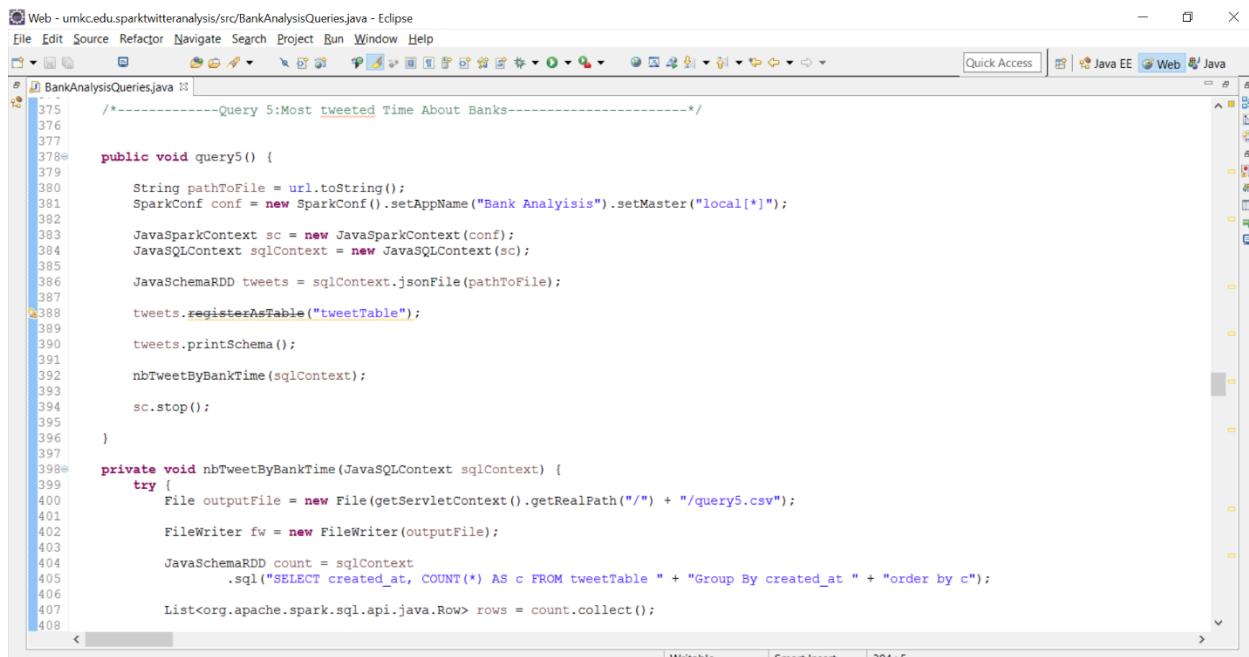


The screenshot shows the Eclipse IDE interface with the code editor open. The file is named 'BankAnalysisQueries.java'. The code implements a query to find the most used languages for tweets about banks. It uses SparkContext, JavaSQLContext, and JavaSchemaRDD. The code includes a main method 'query4' and a private helper method 'nbTweetByQuery4'. The code is annotated with line numbers and comments explaining the logic.

```
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
```

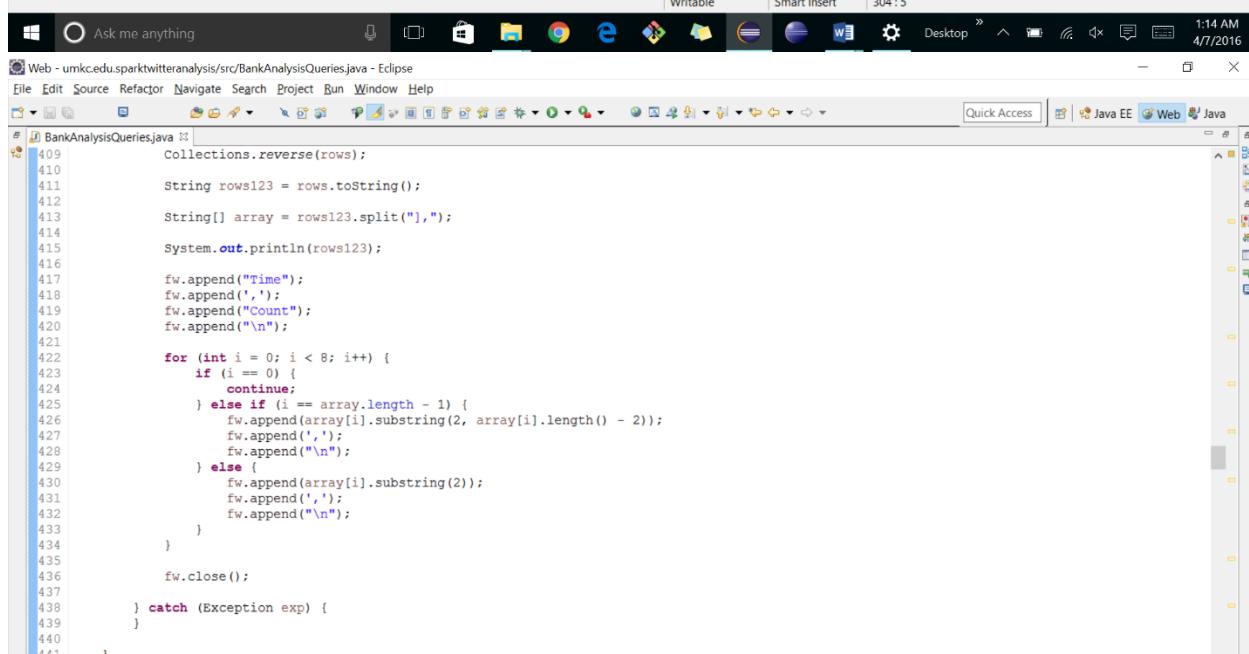
Query 5: Most tweeted specific time at a particular instant about banks

As we all see, when there is some interesting thing is happening regarding banks or well something unexpected happens then people wants to share with their friends and family. So, based on this we did analysis on at what specific time many tweets were recorded about banks.



The screenshot shows the Eclipse IDE interface with the Java code for 'BankAnalysisQueries.java'. The code implements a query to find the most tweeted time about banks. It uses SparkContext and JavaSQLContext to read JSON files, register them as tables, and execute a SQL query to group by created_at and count the tweets. The output is then written to a CSV file named 'query5.csv'.

```
375  *-----Query 5:Most tweeted Time About Banks-----*/
376
377
378 public void query5() {
379     String pathToFile = url.toString();
380     SparkConf conf = new SparkConf().setAppName("Bank Analysis").setMaster("local[*]");
381
382     JavaSparkContext sc = new JavaSparkContext(conf);
383     JavaSQLContext sqlContext = new JavaSQLContext(sc);
384
385     JavaSchemaRDD tweets = sqlContext.jsonFile(pathToFile);
386
387     tweets.registerAsTable("tweetTable");
388
389     tweets.printSchema();
390
391     nbTweetByBankTime(sqlContext);
392
393     sc.stop();
394
395 }
396
397 private void nbTweetByBankTime(JavaSQLContext sqlContext) {
398     try {
399         File outputFile = new File(getServletContext().getRealPath("/") + "/query5.csv");
400
401         FileWriter fw = new FileWriter(outputFile);
402
403         JavaSchemaRDD count = sqlContext
404             .sql("SELECT created_at, COUNT(*) AS c FROM tweetTable " + "Group By created_at " + "order by c");
405
406         List<org.apache.spark.sql.api.java.Row> rows = count.collect();
407
408     }
409
410     Collections.reverse(rows);
411
412     String rows123 = rows.toString();
413
414     String[] array = rows123.split("[", "]");
415
416     System.out.println(rows123);
417
418     fw.append("Time");
419     fw.append(",");
420     fw.append("Count");
421     fw.append("\n");
422
423     for (int i = 0; i < 8; i++) {
424         if (i == 0) {
425             continue;
426         } else if (i == array.length - 1) {
427             fw.append(array[i].substring(2, array[i].length() - 2));
428             fw.append(",");
429             fw.append("\n");
430         } else {
431             fw.append(array[i].substring(2));
432             fw.append(",");
433             fw.append("\n");
434         }
435     }
436
437     fw.close();
438
439     } catch (Exception exp) {
440     }
441 }
442 }
```



The screenshot shows the continuation of the Java code for 'BankAnalysisQueries.java'. The code handles the exception from the previous part, closes the file writer, and ends the main method. The code then proceeds to implement the 'nbTweetByBankTime' method, which reads the collected rows, reverses them, and writes them to a CSV file named 'query5.csv'.

```
409
410     Collections.reverse(rows);
411
412     String rows123 = rows.toString();
413
414     String[] array = rows123.split("[", "]");
415
416     System.out.println(rows123);
417
418     fw.append("Time");
419     fw.append(",");
420     fw.append("Count");
421     fw.append("\n");
422
423     for (int i = 0; i < 8; i++) {
424         if (i == 0) {
425             continue;
426         } else if (i == array.length - 1) {
427             fw.append(array[i].substring(2, array[i].length() - 2));
428             fw.append(",");
429             fw.append("\n");
430         } else {
431             fw.append(array[i].substring(2));
432             fw.append(",");
433             fw.append("\n");
434         }
435     }
436
437     fw.close();
438
439     } catch (Exception exp) {
440     }
441 }
442 }
```



The screenshot shows the final part of the Java code for 'BankAnalysisQueries.java'. It handles the exception from the previous part, closes the file writer, and ends the main method. The code then proceeds to implement the 'nbTweetByBankTime' method, which reads the collected rows, reverses them, and writes them to a CSV file named 'query5.csv'.

```
409
410     Collections.reverse(rows);
411
412     String rows123 = rows.toString();
413
414     String[] array = rows123.split("[", "]");
415
416     System.out.println(rows123);
417
418     fw.append("Time");
419     fw.append(",");
420     fw.append("Count");
421     fw.append("\n");
422
423     for (int i = 0; i < 8; i++) {
424         if (i == 0) {
425             continue;
426         } else if (i == array.length - 1) {
427             fw.append(array[i].substring(2, array[i].length() - 2));
428             fw.append(",");
429             fw.append("\n");
430         } else {
431             fw.append(array[i].substring(2));
432             fw.append(",");
433             fw.append("\n");
434         }
435     }
436
437     fw.close();
438
439     } catch (Exception exp) {
440     }
441 }
442 }
```

Query 6: Type of investment people interested in most

People do often saves money they earned in banks. Banks will invest these money in different investment sectors like real estate, stocks, business etc. and people do often talk about different type of investment and wants to gain some prior knowledge before investing. So, they do often tweet about these different investments. We did analysis based on this to find what type of investment people want to know about or what interests them most.

The screenshot shows the Eclipse IDE interface with the title bar "Web - umkc.edu.sparktwitteranalysis/src/BankAnalysisQueries.java - Eclipse". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, Find, etc. The quick access bar on the right has Java EE, Web, Java options. The code editor displays Java code for "BankAnalysisQueries.java". The code uses SparkContext and JavaSQLContext to process tweets from a JSON file, register a table named "tweetTable", and count tweets by investment type. A try-catch block is used to write the results to a CSV file named "query6.csv".

```
443     /*-----Query 6:Type of Investment People Interested in-----*/
444
445@ public void query6() {
446
447     String pathToFile = url.toString();
448
449     SparkConf conf = new SparkConf().setAppName("Bank Analysis").setMaster("local[*]");
450
451     JavaSparkContext sc = new JavaSparkContext(conf);
452
453     JavaSQLContext sqlContext = new JavaSQLContext(sc);
454
455     JavaSchemaRDD tweets = sqlContext.jsonFile(pathToFile);
456
457     tweets.registerAsTable("tweetTable");
458
459     tweets.printSchema();
460
461     nbTweetByInvestment(sqlContext);
462
463     sc.stop();
464
465 }
466
467@ private void nbTweetByInvestment(JavaSQLContext sqlContext)
468
469 {
470     try {
471
472         File outputFile = new File(getServletContext().getRealPath("/") + "/query6.csv");
473
474         FileWriter fw = new FileWriter(outputFile);
475
476         JavaSchemaRDD count = sqlContext
```

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

- Title Bar:** Ask me anything
- Toolbar:** Standard Eclipse toolbar with icons for File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Quick Access:** Shows Java EE, Web, Java.
- Code Editor:** Displays the file `BankAnalysisQueries.java`. The code contains several `sql` statements and variable assignments for counting words like "stock", "real estate", "Land", etc., from a `tweetTable`. It also includes logic to extract substrings from the results.
- Right-hand View:** Shows the Java EE perspective with various toolbars and views.

```
477     .sql("SELECT COUNT(*) AS c FROM tweetTable " + "WHERE text LIKE '%stock%'");
478     JavaSchemaRDD count1 = sqlContext.sql(
479         "SELECT COUNT(*) AS c FROM tweetTable " + "WHERE text LIKE '%real estate%' OR text LIKE '%Land%'");
480     JavaSchemaRDD count2 = sqlContext
481         .sql("SELECT COUNT(*) AS c FROM tweetTable " + "WHERE text LIKE '%business%'");
482     JavaSchemaRDD count3 = sqlContext
483         .sql("SELECT COUNT(*) AS c FROM tweetTable " + "WHERE text LIKE '%bonds%'");
484     JavaSchemaRDD count4 = sqlContext.sql("SELECT COUNT(*) AS c FROM tweetTable "
485         + "WHERE text LIKE '%precious objects%' OR text LIKE '%gold%' OR text LIKE '%diamonds%' OR text LIKE '%silver%' OR text LIKE '%commodities%' OR text LIKE '%natural gas%' OR text LIKE '%gasoline%' OR text LIKE '%crude oil%'");
486     JavaSchemaRDD count5 = sqlContext.sql("SELECT COUNT(*) AS c FROM tweetTable "
487         + "WHERE text LIKE '%gold%'");
488     JavaSchemaRDD count6 = sqlContext
489         .sql("SELECT COUNT(*) AS c FROM tweetTable " + "WHERE text LIKE '%funds%'");
490
491     List<Row> stocks = count.collect();
492     String stocks12 = stocks.toString();
493     String stocks1 = stocks12.substring(stocks12.indexOf("[") + 2, stocks12.indexOf("]"));
494
495     List<Row> realestate = count1.collect();
496     String realestate12 = realestate.toString();
497     String realestate1 = realestate12.substring(realestate12.indexOf("[") + 2, realestate12.indexOf("]"));
498
499     List<Row> business = count2.collect();
500     String business12 = business.toString();
501     String business1 = business12.substring(business12.indexOf("[") + 2, business12.indexOf("]"));
502
503     List<Row> bonds = count3.collect();
504     String bonds12 = bonds.toString();
505     String bonds1 = bonds12.substring(bonds12.indexOf("[") + 2, bonds12.indexOf("]"));
506
507     List<Row> gold = count4.collect();
508     String gold12 = gold.toString();
509     String gold1 = gold12.substring(gold12.indexOf("[") + 2, gold12.indexOf("]"));
```

Web - umkc.edu.sparktwitteranalysis/src/BankAnalysisQueries.java - Eclipse

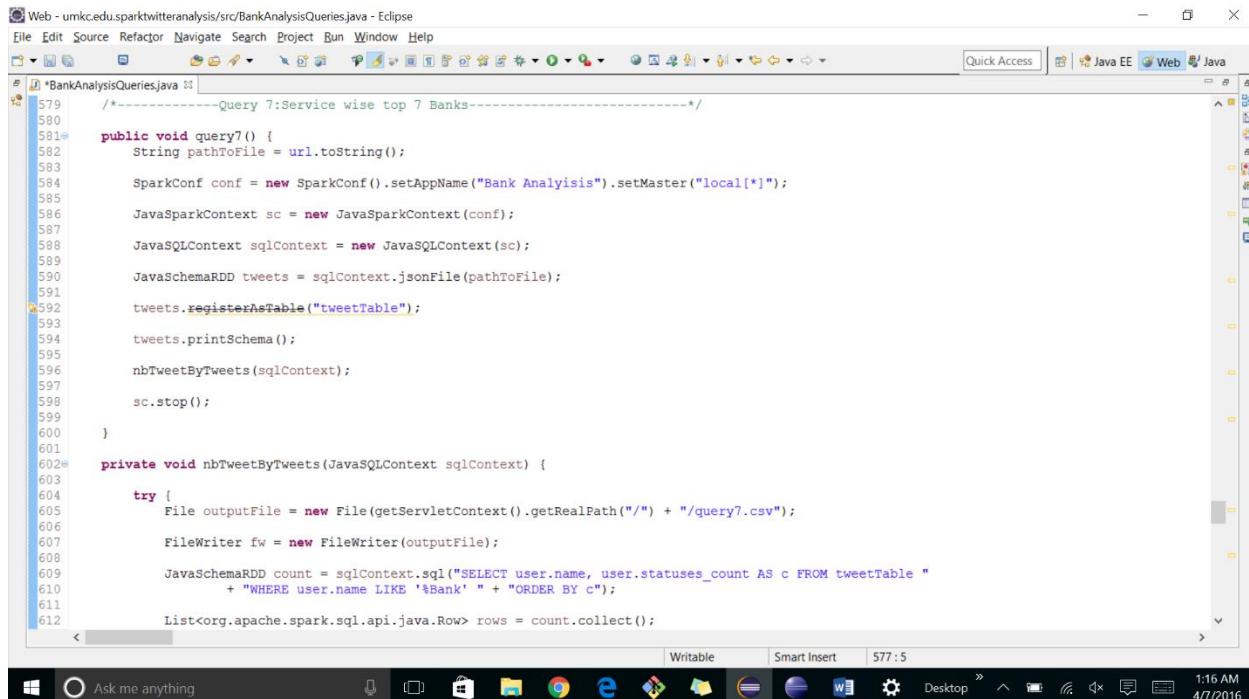
```
511     List<Row> commodities = count5.collect();
512     String commodities12 = commodities.toString();
513     String commodities1 = commodities12.substring(commodities12.indexOf("[") + 2, commodities12.indexOf("]"));
514
515     List<Row> funds = count6.collect();
516     String funds12 = funds.toString();
517     String funds1 = funds12.substring(funds12.indexOf("[") + 2, funds12.indexOf("]"));
518
519     fw.append("InvestmentType");
520     fw.append(",");
521     fw.append("count");
522     fw.append("\n");
523     fw.append("stocks");
524     fw.append(",");
525     fw.append(stocks1);
526     fw.append("\n");
527     fw.append("realestate");
528     fw.append(",");
529     fw.append(realestate1);
530     fw.append("\n");
531     fw.append("business");
532     fw.append(",");
533     fw.append(business1);
534     fw.append("\n");
535     fw.append("bonds");
536     fw.append(",");
537     fw.append(bonds1);
538     fw.append("\n");
539     fw.append("preciousObjects");
540     fw.append(",");
541     fw.append(gold1);
542     fw.append("\n");
543     fw.append("commodities");
544     fw.append(",");
```

Web - umkc.edu.sparktwitteranalysis/src/BankAnalysisQueries.java - Eclipse

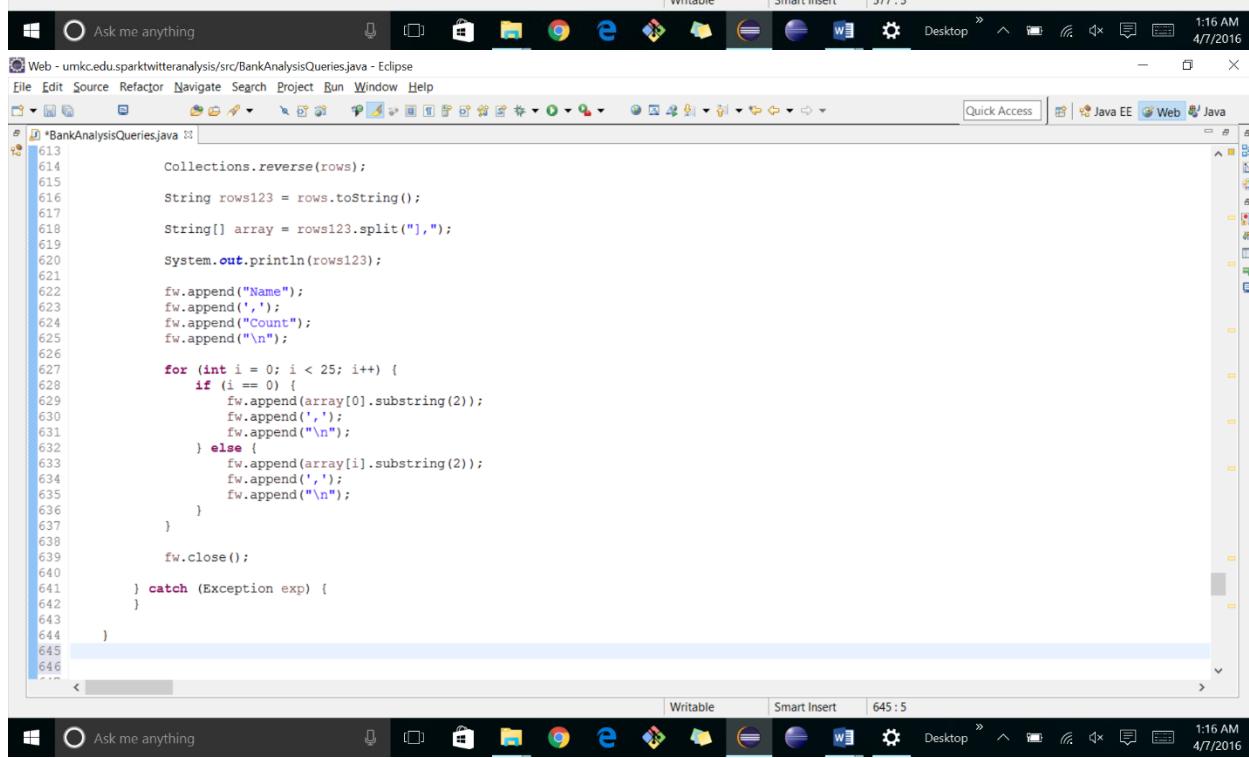
```
545     fw.append(commodities1);
546     fw.append("\n");
547     fw.append("funds");
548     fw.append(",");
549     fw.append(funds1);
550     fw.append("\n");
551
552     fw.close();
553
554 } catch (Exception exp) {
555 }
556
557 }
```

Query 7: Service wise top banks

Nowadays banks are using twitter as source for reaching out people and wants to know the opinion about their products to ensure quality. They even provide customer support through this social networking site. So, we did analysis on banks to find out which banks is in touch with their customers the most i.e., which banks is providing more service to their customers through twitter.



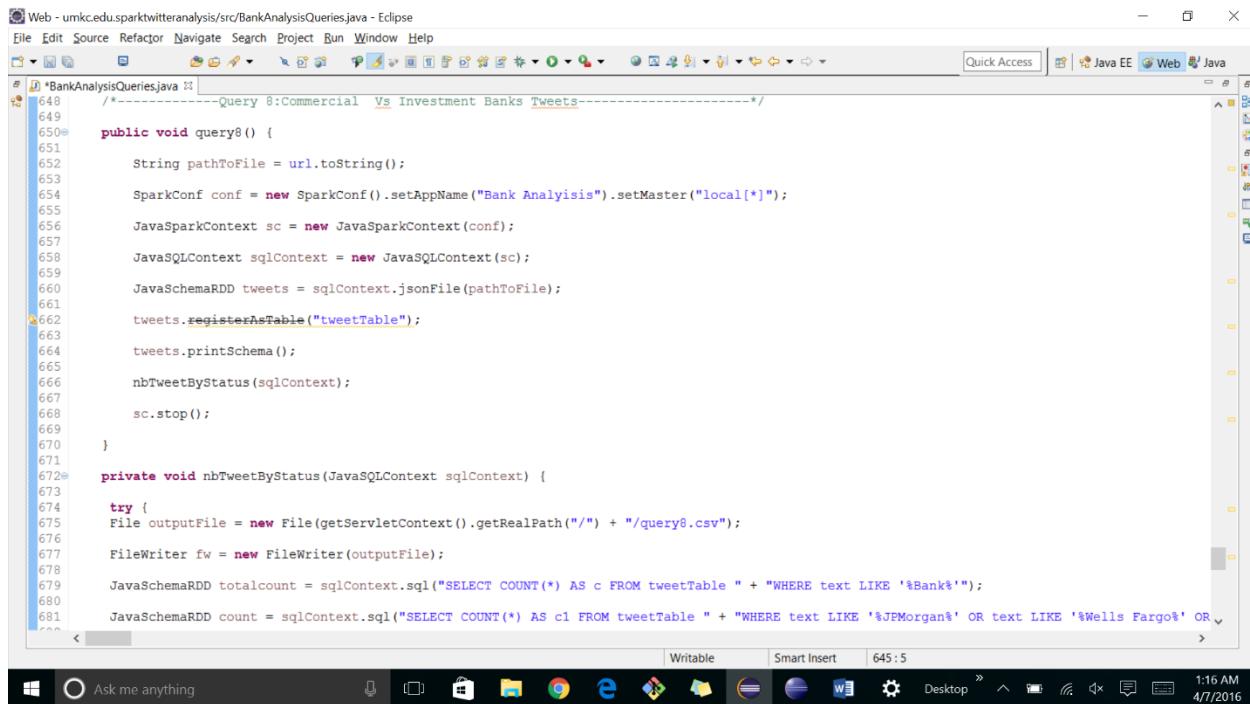
```
579  *-----Query 7:Service wise top 7 Banks-----*/
580
581  public void query7() {
582      String pathToFile = url.toString();
583
584      SparkConf conf = new SparkConf().setAppName("Bank Analysis").setMaster("local[*]");
585
586      JavaSparkContext sc = new JavaSparkContext(conf);
587
588      JavaSQLContext sqlContext = new JavaSQLContext(sc);
589
590      JavaSchemaRDD tweets = sqlContext.jsonFile(pathToFile);
591
592      tweets.registerAsTable("tweetTable");
593
594      tweets.printSchema();
595
596      nbTweetByTweets(sqlContext);
597
598      sc.stop();
599
600  }
601
602  private void nbTweetByTweets(JavaSQLContext sqlContext) {
603
604      try {
605          File outputFile = new File(getServletContext().getRealPath("/") + "/query7.csv");
606
607          FileWriter fw = new FileWriter(outputFile);
608
609          JavaSchemaRDD count = sqlContext.sql("SELECT user.name, user.statuses_count AS c FROM tweetTable "
610              + "WHERE user.name LIKE '%Bank' " + "ORDER BY c");
611
612          List<org.apache.spark.sql.api.java.Row> rows = count.collect();
613
614          Collections.reverse(rows);
615
616          String rows123 = rows.toString();
617
618          String[] array = rows123.split("[", "]");
619
620          System.out.println(rows123);
621
622          fw.append("Name");
623          fw.append(",");
624          fw.append("Count");
625          fw.append("\n");
626
627          for (int i = 0; i < 25; i++) {
628              if (i == 0) {
629                  fw.append(array[0].substring(2));
630                  fw.append(',');
631                  fw.append("\n");
632              } else {
633                  fw.append(array[i].substring(2));
634                  fw.append(',');
635                  fw.append("\n");
636              }
637          }
638
639          fw.close();
640
641      } catch (Exception exp) {
642      }
643
644  }
```



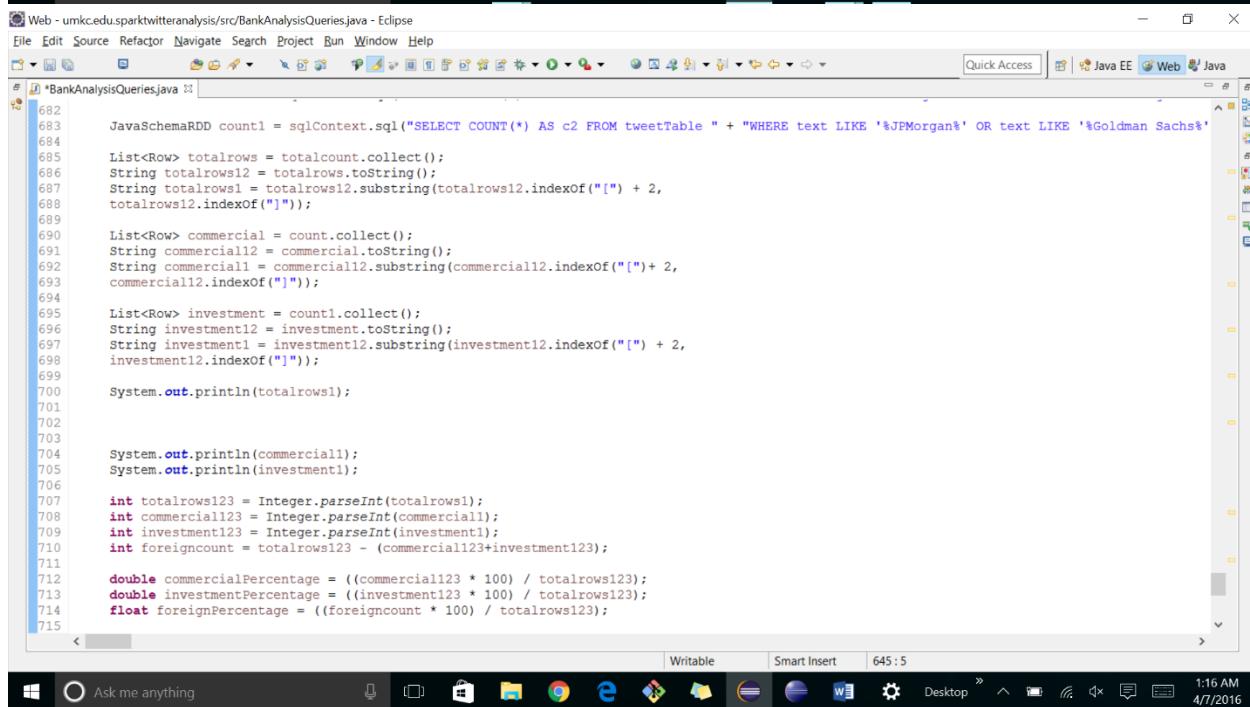
```
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2489
2490
2491
2492
2493
2494
2495
2496
2497
2497
2498
2499
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2598
2599
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2698
2699
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
```

Query 8: Commercial Vs investment banks of USA

Banks are often categorized into Commercial and Investment Banks. The U.S finance industry has grown about 50% to 60% more compared to other countries over the past 2 decades. Bank regulation is very fragmented in USA when compared to other Group of Ten (economics) countries. So, we did analysis on finding out the percentage of tweets often tweeted about commercial banks and investment banks and tweets regarding to banks of all other countries.



```
/*Query 8:Commercial Vs Investment Banks Tweets*/
public void query8() {
    String pathToFile = url.toString();
    SparkConf conf = new SparkConf().setAppName("Bank Analysis").setMaster("local[*]");
    JavaSparkContext sc = new JavaSparkContext(conf);
    JavaSQLContext sqlContext = new JavaSQLContext(sc);
    JavaSchemaRDD tweets = sqlContext.jsonFile(pathToFile);
    tweets.registerAsTable("tweetTable");
    tweets.printSchema();
    nbTweetByStatus(sqlContext);
    sc.stop();
}
private void nbTweetByStatus(JavaSQLContext sqlContext) {
    try {
        File outputFile = new File(getServletContext().getRealPath("/") + "/query8.csv");
        FileWriter fw = new FileWriter(outputFile);
        JavaSchemaRDD totalcount = sqlContext.sql("SELECT COUNT(*) AS c FROM tweetTable " + "WHERE text LIKE '%Bank%'");
        JavaSchemaRDD count = sqlContext.sql("SELECT COUNT(*) AS c1 FROM tweetTable " + "WHERE text LIKE '%JPMorgan%' OR text LIKE '%Wells Fargo%' OR text LIKE '%Bank%'");
        fw.write(totalcount.first().c + "," + count.first().c1);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```



```
JavaSchemaRDD count1 = sqlContext.sql("SELECT COUNT(*) AS c2 FROM tweetTable " + "WHERE text LIKE '%JPMorgan%' OR text LIKE '%Goldman Sachs%'");
List<Row> totalrows = totalcount.collect();
String totalrows12 = totalrows.toString();
String totalrows1 = totalrows12.substring(totalrows12.indexOf("[") + 2, totalrows12.indexOf("]"));
List<Row> commercial = count1.collect();
String commercial12 = commercial.toString();
String commercial1 = commercial12.substring(commercial12.indexOf("[") + 2, commercial12.indexOf("]"));
List<Row> investment = count1.collect();
String investment12 = investment.toString();
String investment1 = investment12.substring(investment12.indexOf("[") + 2, investment12.indexOf("]"));
System.out.println(totalrows1);
System.out.println(commercial1);
System.out.println(investment1);
int totalrows123 = Integer.parseInt(totalrows1);
int commercial123 = Integer.parseInt(commercial1);
int investment123 = Integer.parseInt(investment1);
int foreigncount = totalrows123 - (commercial123+investment123);
double commercialPercentage = ((commercial123 * 100) / totalrows123);
double investmentPercentage = ((investment123 * 100) / totalrows123);
float foreignPercentage = ((foreigncount * 100) / totalrows123);
```

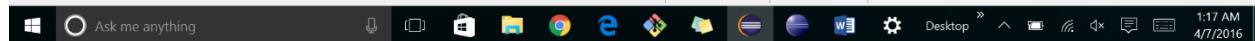
Web - umkc.edu.sparktwitteranalysis/src/BankAnalysisQueries.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access Java EE Web Java

```
BankAnalysisQueries.java
```

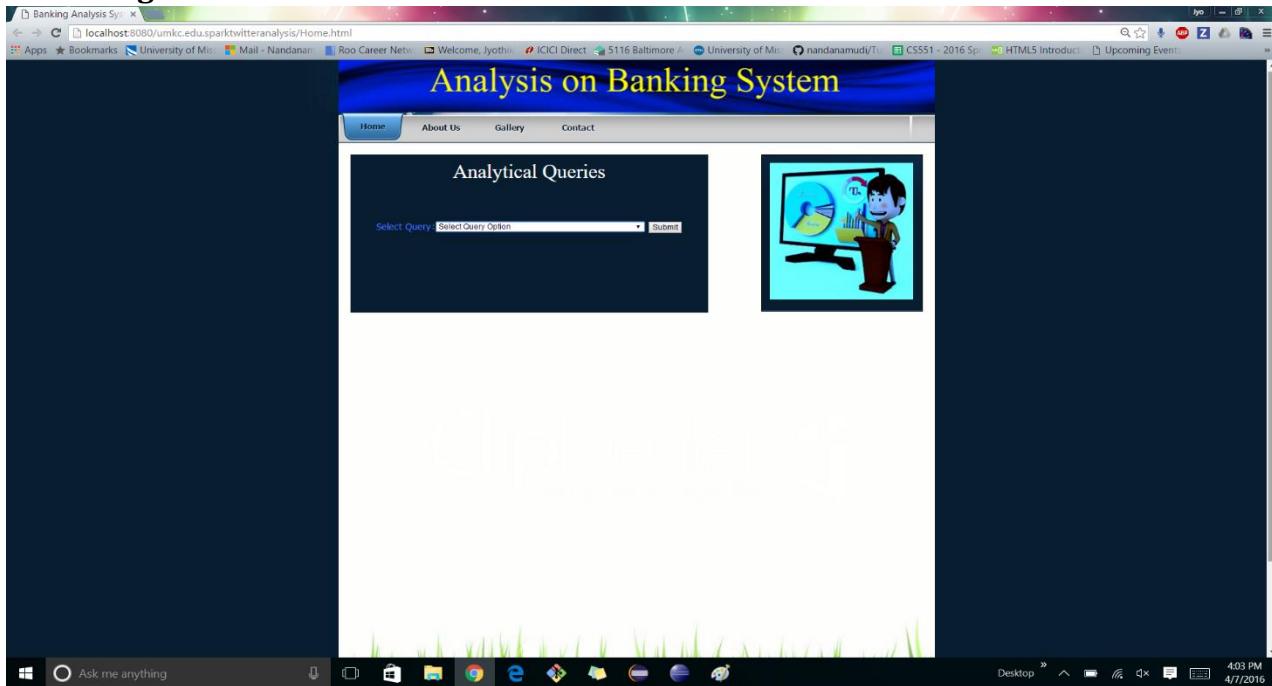
```
716
717
718
719     System.out.println(commercialPercentage);
720     System.out.println(investmentPercentage);
721     System.out.println(foreignPercentage);
722
723     String commercialPercentage1 = Double.toString(commercialPercentage);
724     String investmentPercentage1 = Double.toString(investmentPercentage);
725     String foreignPercentage1 = Float.toString(foreignPercentage);
726
727     fw.append("TweetStatus");
728     fw.append(',');
729     fw.append("Percentage");
730     fw.append("\n");
731     fw.append("CommercialBanks%");
732     fw.append(',');
733     fw.append(commercialPercentage1);
734     fw.append("\n");
735     fw.append("InvestmentBanks%");
736     fw.append(',');
737     fw.append(investmentPercentage1);
738     fw.append("\n");
739     fw.append("Total%");
740     fw.append(',');
741     fw.append(foreignPercentage1);
742     fw.append("\n");
743
744     fw.close();
745
746     } catch (Exception exp) {
747     }
748 }
749 }
```



b. Visualization results

On client HTML, HTML5, CSS and Javascript were used to represent the analytical queries in attracting way. For pictorial representation we chose d3.js library.

Home Page:



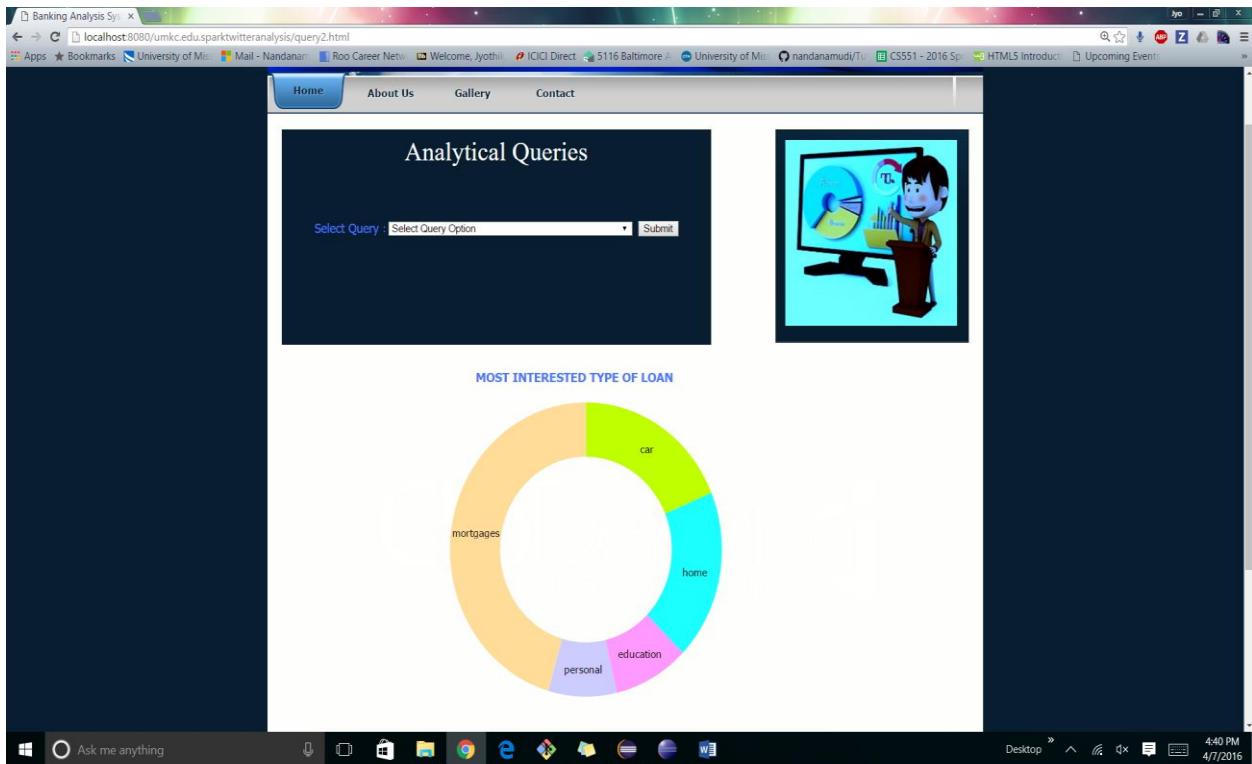
Query 1 Output:



The query 1 is about finding the popular bank across the social networking site: twitter

Our analysis shows that World Bank being most popular as it is being followed by 1712829 followers all over the world. Guaranty Trust Bank in the second, Jodrell Bank in third place, Kotak Mahindra Bank in fourth place, Diamond bank in fifth place and ICICI Bank in sixth place respectively with 527567, 230042, 113469, 87193, 83472 respectively.

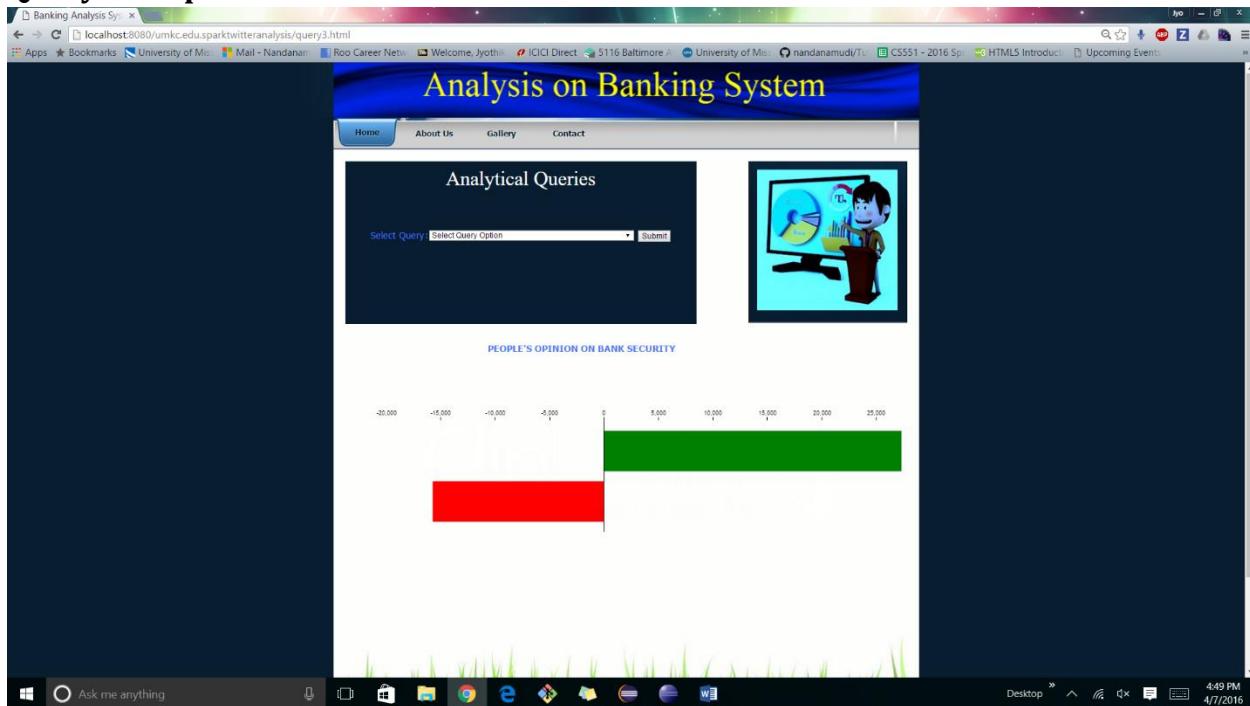
Query 2 Output:



The query 2 is about finding the most interested type of loan.

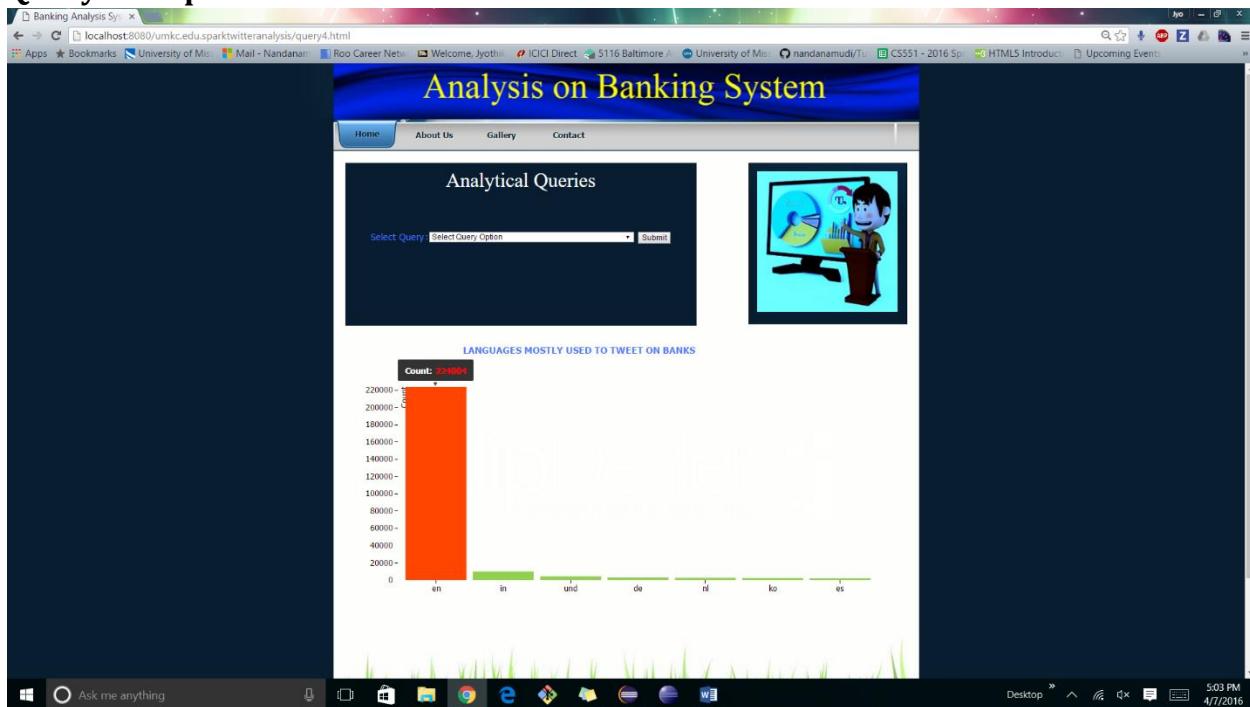
Our analysis shows that Mortgages is the type of loan people talk about loan and vehicle, home, education and personal takes the next level of interest in order respectively.

Query 3 Output:



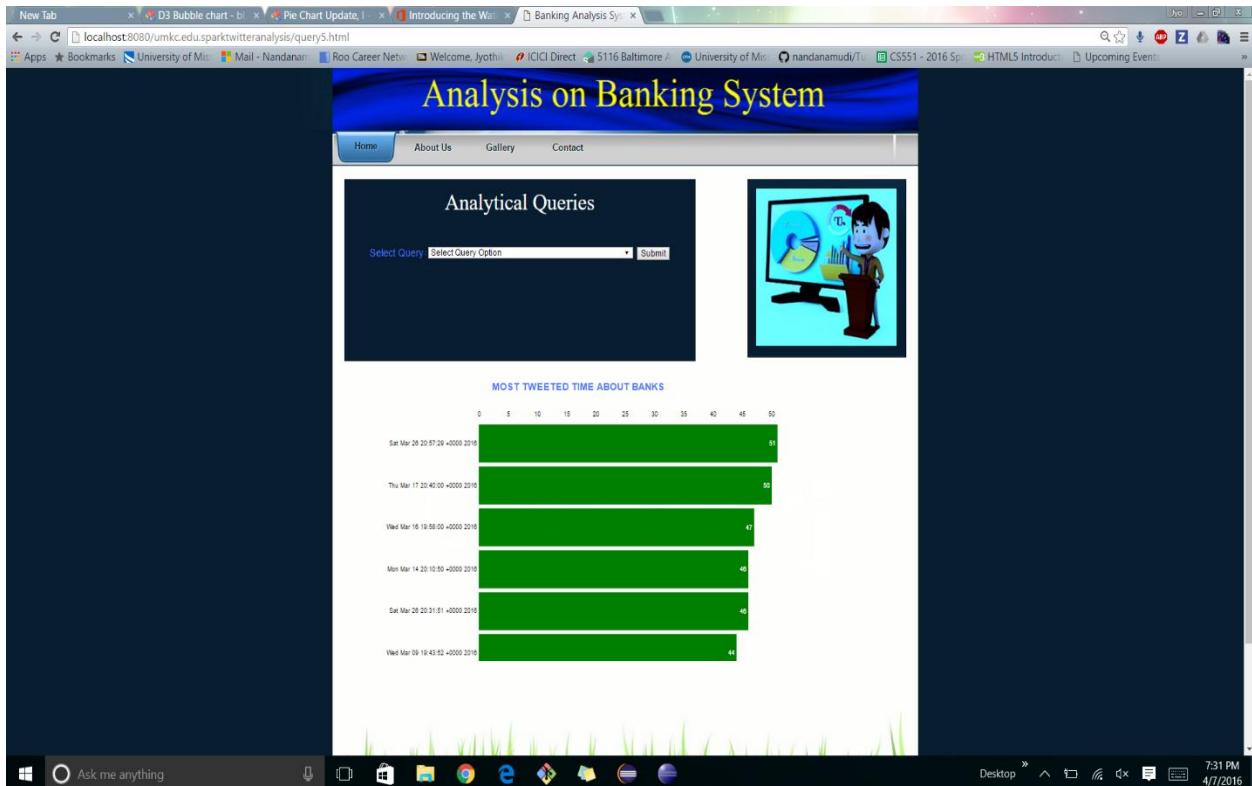
The query 3 is about analyzing the data to find the opinion of people on bank security. We come to conclusion people are with good opinion about banks as you can see in the snapshot the green color indicates positive side opinion and the red colored bar indicates the negative opinion.

Query 4 Output:



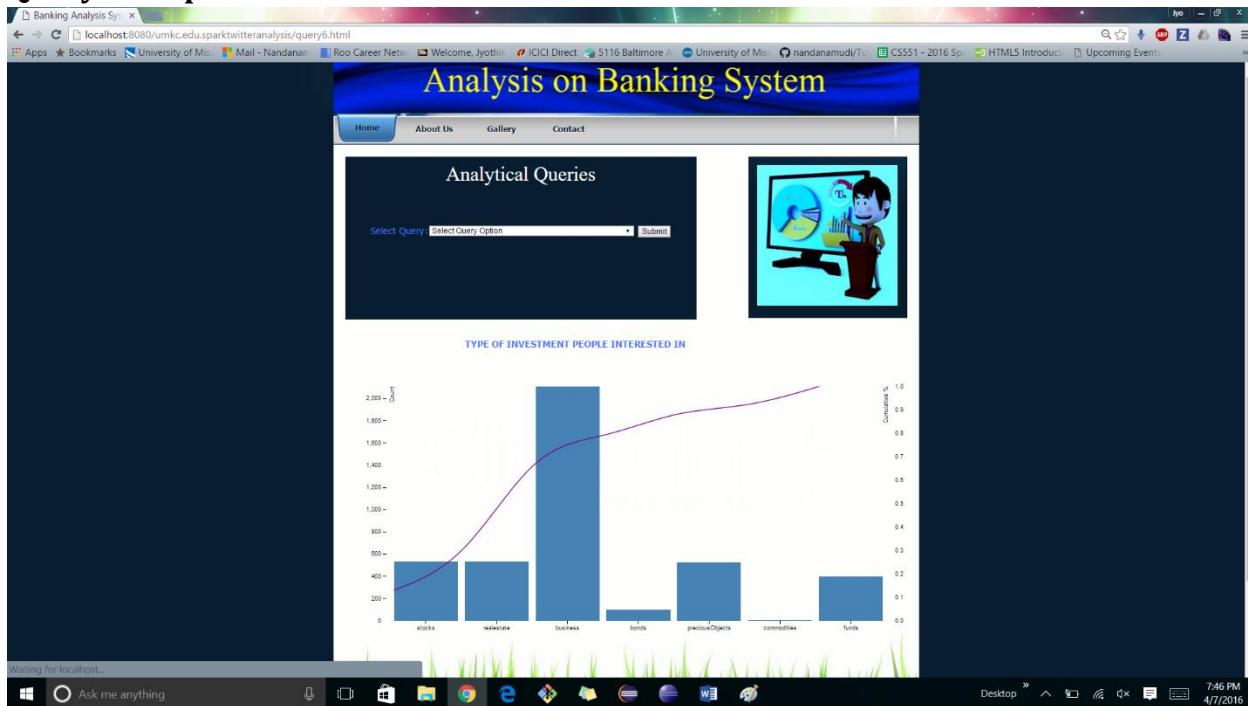
The query 4 is finding the most communicative language about banks across twitter. The most used language is English as we all know with a total count of 224000 tweets of all the tweets collected.

Query 5 Output:



The query 5 is about finding the time at which there are many tweets about banks. For our analysis on the collected data the output of this query is on Mar 26, 2016 Saturday at 20:57:29, 51 tweets has been recognized. And in the second place on Mar 17, 2016 Thursday at 20:40:00, 50 tweets has been recognized.

Query 6 Output:



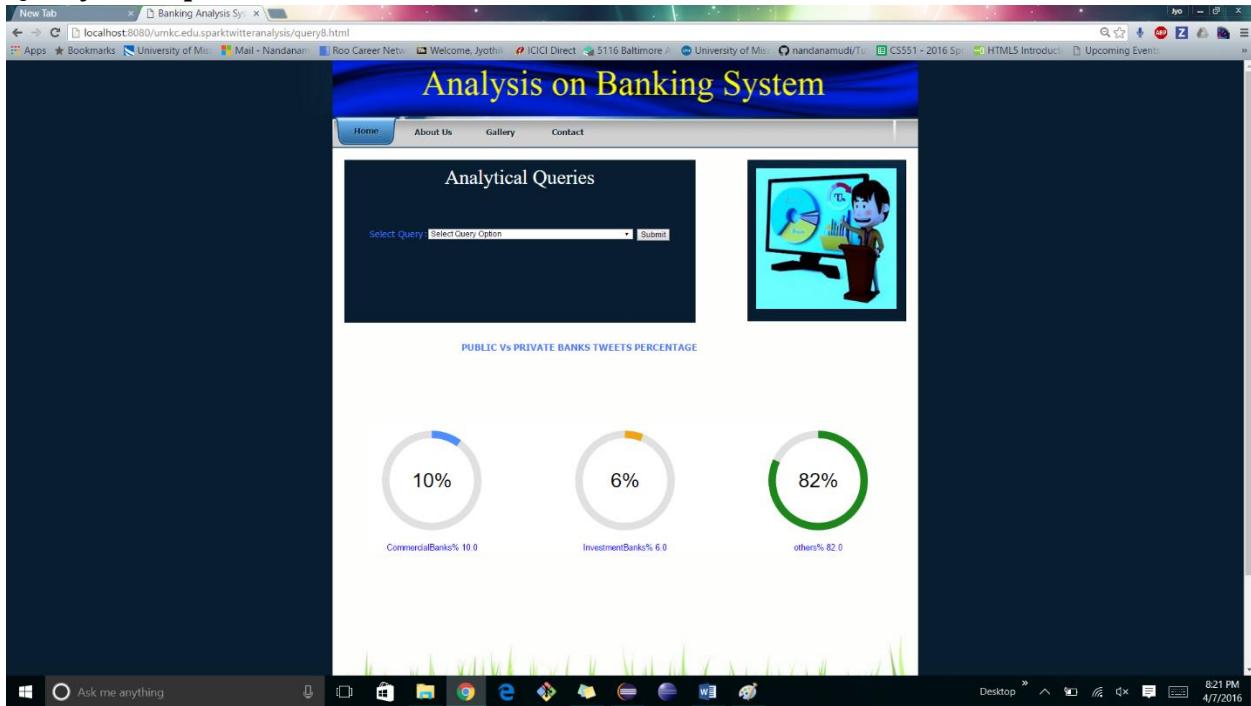
The query 6 is about finding the most interested type of investment people interested in most. Our analysis shows that Business takes the most interested type of investment when compared to others. In the second place precious objects like gold, silver, diamonds, platinum etc., takes place and people are interested in stocks and real estate almost equally.

Query 7 Output:



The query 7 is about finding the service oriented banks that are using twitter the most to reach out the people and their customers. Here we chose the status count of bank for comparison. Our analytical results shows that Lloyds Banks stays in top position in service wise that uses twitter has medium for reaching the people. Guaranty Bank, HDFC Bank, TD Bank and Next Bank occupies the next positions respectively.

Query 8 Output:



The query 8 is to find the percentage of commercial banks, investment banks of USA of all the tweets available. Our analysis has shown that commercial banks occupy about 10% and investment banks is of 6% and tweets related to all other banks worldwide is of 82%.

iii. Other Pages:

About Us Page:



Here we mentioned what the project is about and what all the steps involved in designing the project. We also mentioned few extra links like Facebook share, like buttons, twitter share tag, google following and share tags etc.

Recently IBM has done analysis on Banking Industry and you can find the links below at the bottom of the page which is an added reference to our project.

On the right side bar there is a search engine, where you can search all the information you want to and calendar plugin is an extra added quality of this page.

Gallery Page:

The screenshot shows a web browser displaying the "Analysis on Banking System" gallery page. The page features a navigation bar with links for Home, About Us, Gallery (which is currently selected), and Contact. Below the navigation bar, there are five distinct chart displays, each with a title, a brief description, and a "View in Detail" button.

- Discrete Bar Chart**: A bar chart showing the frequency of bank names. The Y-axis is labeled "Number of banks [100]" and ranges from 0 to 15. The X-axis categories are "Federated Banking", "Public Sector", "Private Sector", and "Other". The data is approximately: Federated Banking (~14), Public Sector (~11), Private Sector (~7), Other (~8).
- Pie Chart Update**: A pie chart illustrating the proportion of different types of loans. The segments are colored orange, green, blue, and white.
- Barchart with neg values**: A horizontal bar chart showing negative values. The X-axis has numerical ticks from -10 to 10. The bars are orange and blue, representing values like -10, -5, 0, 5, 10.
- Radial Progress Bar**: A circular progress bar with a dark blue ring and a light blue center, set against a background of green grass.
- Horizontal Bar Chart**: A horizontal bar chart showing the time at which people tweeted the most. The X-axis represents hours of the day, and the Y-axis represents the count of tweets. The data is highly skewed with a single dominant peak around 10 AM.

At the bottom of the page, there are "Quick Links" for Home, Gallery, and Contact Us, along with social sharing icons for Facebook, Twitter, and Email. There are also "Other Sites" links to IBM Analytics and IBM BigData Blog. The browser's status bar indicates the date and time: 9:11 PM 4/7/2016.

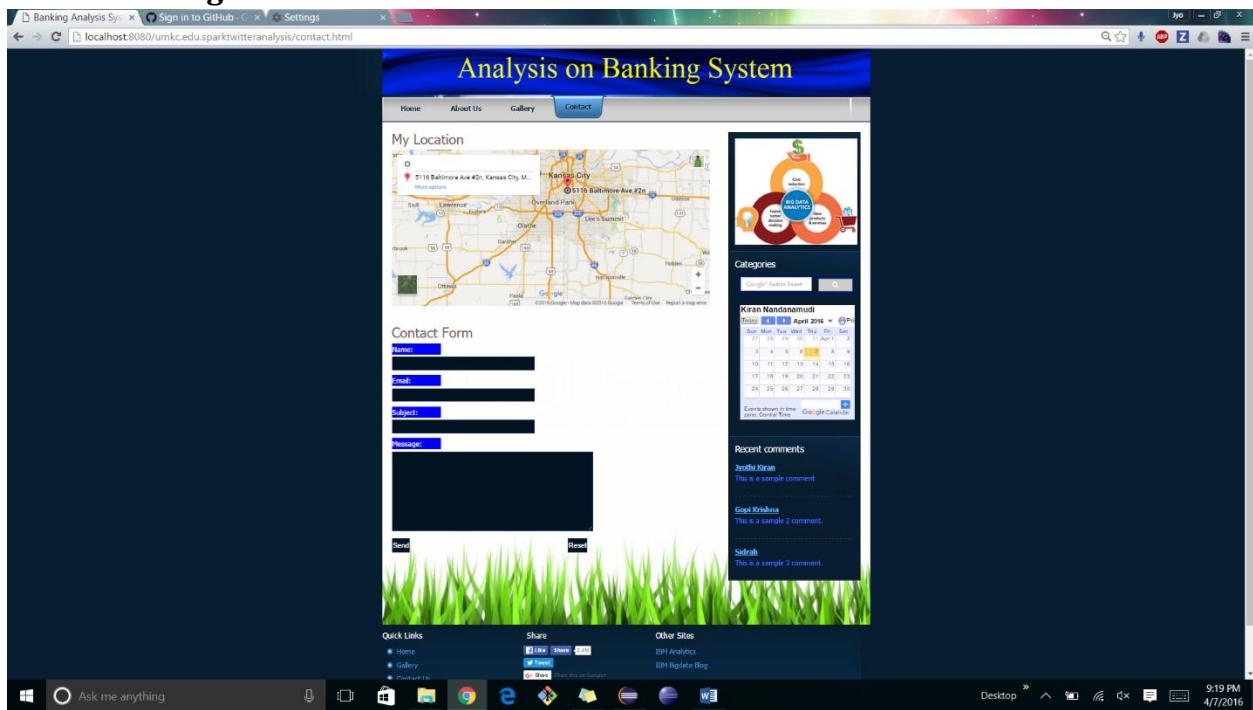
This screenshot shows the same gallery page with a different set of charts. The navigation bar and overall layout remain the same.

- Bar Chart with tool tips**: A bar chart showing the frequency of language usage by people to tweet about banks. The Y-axis has numerical ticks from 0 to 10. The bars are yellow, with one prominent red bar reaching nearly 10.
- Line and Bar Chart**: A dual-axis chart showing a line graph (blue) and a bar chart (blue). The X-axis represents time periods, and the Y-axis represents a quantitative value. The line shows a sharp peak corresponding to the bar chart's highest point.
- Bar graph**: A bar chart showing which bank is most service-oriented based on the status count of their tweets. The Y-axis represents the status count, ranging from 0 to 10. The bars are purple, with the first bar being the tallest (~10).

The bottom section of the page includes "Quick Links" for Home, Gallery, and Contact Us, social sharing icons, and "Other Sites" links. The browser's status bar indicates the date and time: 11:17 PM 4/7/2016.

This page shows all the visualization charts and graphs we used in this project for analyzing. A brief information is shown explaining why we used that graph and links has been provided to show the complete details of the graphs and charts for complete information of the visualization charts.

Contact Us Page:



This page just shows our location to contact us as well as contact from to post if there is anything there is need for clarification.

IV. Summary of the Project

The summary of our analysis on the banking sector is available in the below table to give outlook of whole project.

Query	Result
1. Most Popular Bank	World Bank
2. Most Interested type of loan	Mortgages
3. People Opinion on Bank Security	Secure
4. Most used language to tweet about banks	English (en)
5. Time at which people tweeted mostly on banks	March 26, 2016 at 20:57:29
6. Type of investment people often talk about	Business
7. Service wise top bank which uses twitter	Lloyd Bank of London

V. Conclusion and links:

Our analysis on banks has given some insights on the data collected from the twitter and gives some idea upon the banks. In overall this project can be extended to a greater extents for analyzing huge amount of data.

Link to the Project

<https://github.com/nandanamudi/PB Phase 2-Project>

VI. References:

<http://stackoverflow.com/questions/34236112/java-lang-noclassdeffounderror-org-apache-spark-sparkconf>

<https://m.oschina.net/blog/467423>

<https://www.mapr.com/ebooks/spark/03-apache-spark-architecture-overview.html>

<http://www.ibm.com/analytics/us/en/industry/banking/>

<http://www.ibmbigdatahub.com/blog/analytics-banking-services>

<http://www.taywils.me/2013/11/05/javasparkframeworktutorial.html>

<https://github.com/mbostock/d3/wiki/Gallery>