Phase II Report

Sasya Malireddy

Vijay Kumar Tummala

Praveen Kumar Surapaneni

Introduction:

We intend to develop a Web Application that will dynamically analyze tweets on IPL 2016 that is going to be held in India and generate interesting findings on this data.

Design:

Every line of code written in our project is in JAVA and Spark is the framework we have used for running big data analysis. Sparkcontext is the DB we have used for storing temporary data. All the graphs were developed using D3.js.

Language used: JAVA

Frameworks: SPARK, D3.js

DB: SparkDB

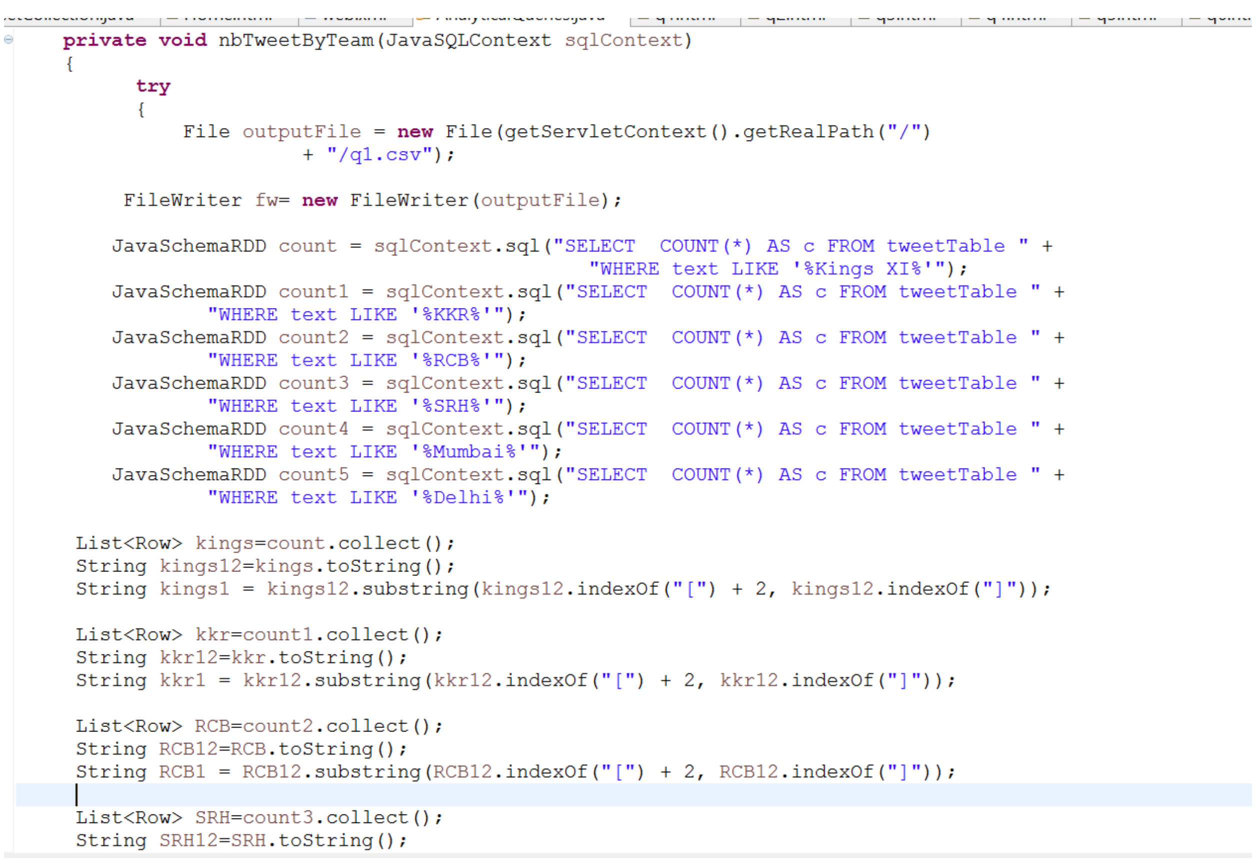
UI: HTML, CSS

Home page consists links for the available queries and by clicking on each link we will process an html request in the backend. For each request there will be different servlet which will handle the incoming requests. For each incoming request we will instantiate a servlet which will run a query on the spark db. Once the results are received we will create a spark context which will read the tweet data and generate the output for that particular query. We designed eight queries which will generate eight different analysis.

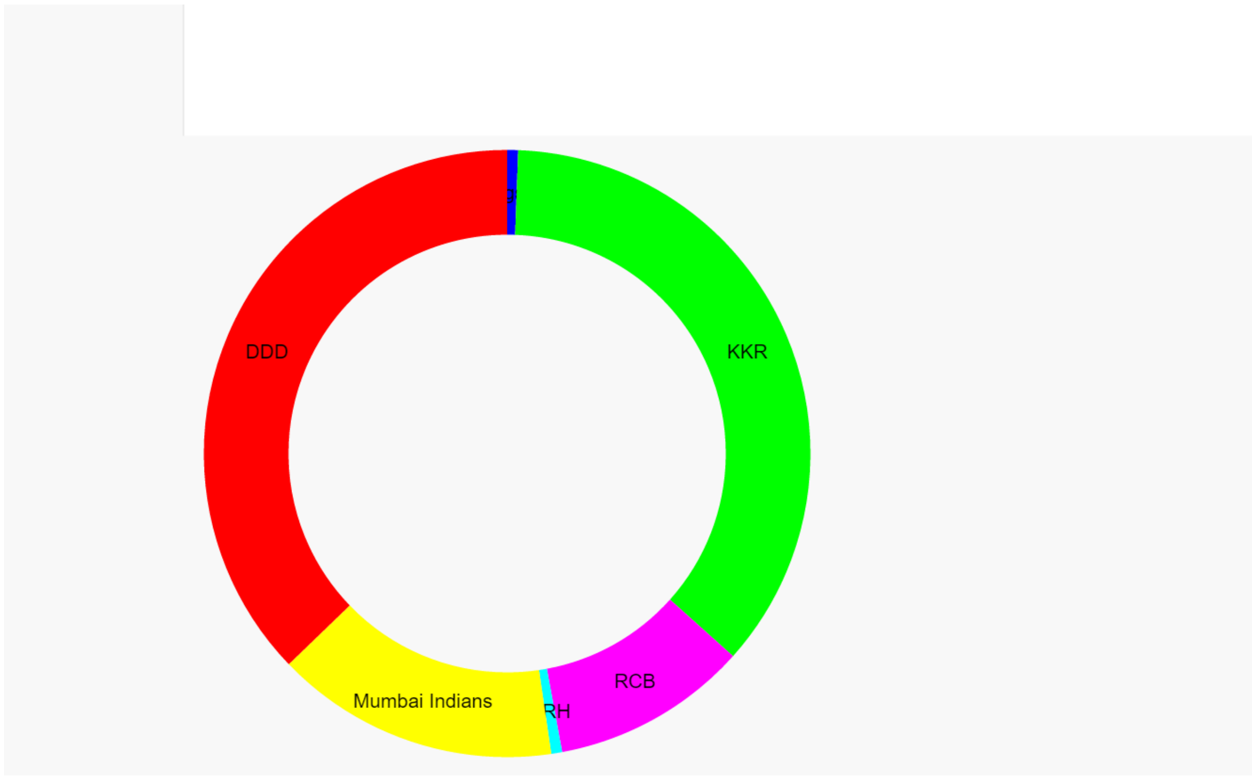
Queries:

Query 1: Most Popular IPL teams

The popularity for the each IPL team is analyzed by collecting the count of tweets on each team. We visualize the following of all IPL teams using a Pie chart.



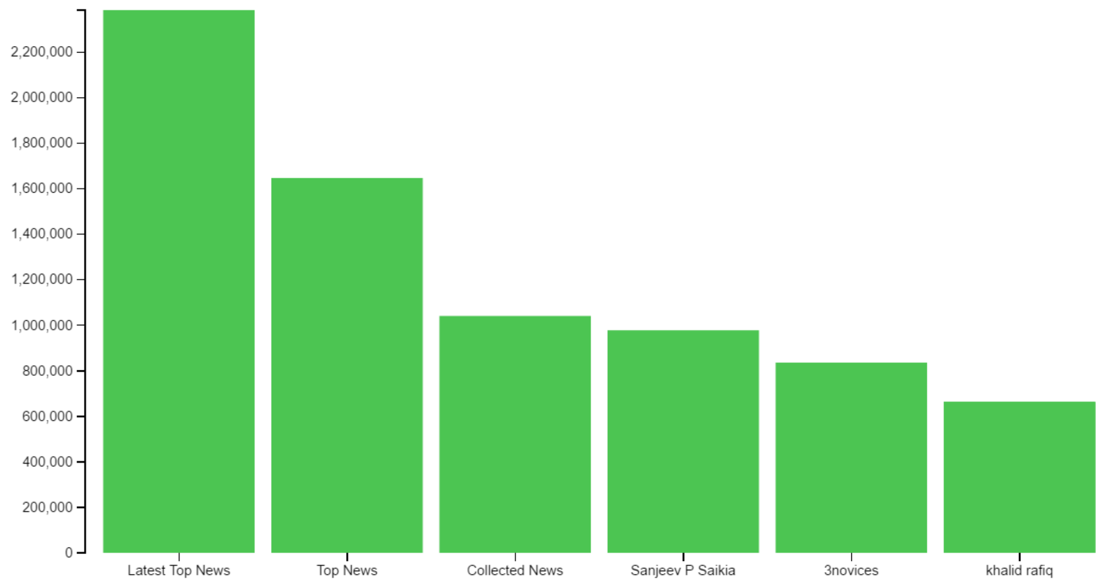
```
    private void nbTweetByTeam(JavaSQLContext sqlContext)
    {
          try
          {
              File outputFile = new File(getServletContext().getRealPath("/")
                        + "/q1.csv");

         FileWriter fw= new FileWriter(outputFile);

        JavaSchemaRDD count = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable " +
                                              "WHERE text LIKE '%Kings XI%'");
        JavaSchemaRDD count1 = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable " +
                "WHERE text LIKE '%KKR%'");
        JavaSchemaRDD count2 = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable " +
                "WHERE text LIKE '%RCB%'");
        JavaSchemaRDD count3 = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable " +
                "WHERE text LIKE '%SRH%'");
        JavaSchemaRDD count4 = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable " +
                "WHERE text LIKE '%Mumbai%'");
        JavaSchemaRDD count5 = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable " +
                "WHERE text LIKE '%Delhi%'");

     List<Row> kings=count.collect();
     String kings12=kings.toString();
     String kings1 = kings12.substring(kings12.indexOf("[") + 2, kings12.indexOf("]"));

     List<Row> kkr=count1.collect();
     String kkr12=kkr.toString();
     String kkr1 = kkr12.substring(kkr12.indexOf("[") + 2, kkr12.indexOf("]"));

     List<Row> RCB=count2.collect();
     String RCB12=RCB.toString();
     String RCB1 = RCB12.substring(RCB12.indexOf("[") + 2, RCB12.indexOf("]"));

     List<Row> SRH=count3.collect();
     String SRH12=SRH.toString();
```



Query 2: Top Most Users Tweeting about IPL

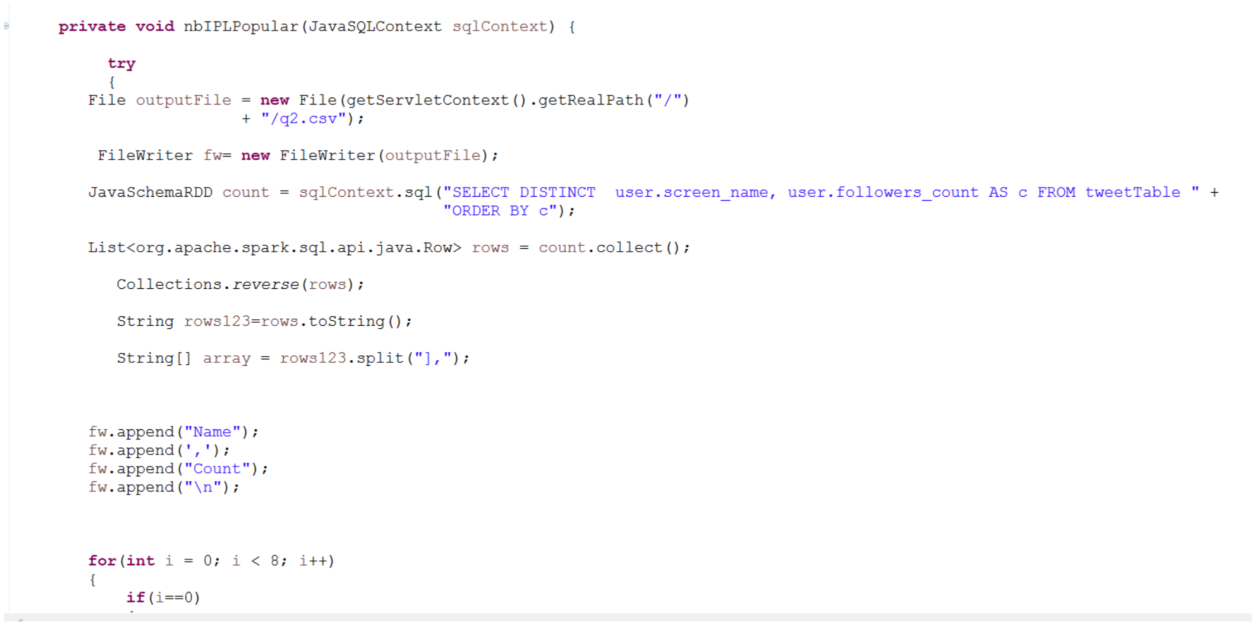
The top users who tweet more about IPL on twitter are found using the following query. The Bar Graphs are used to represent these top users and their tweet counts.

```
private void nbmosttwittingUser(JavaSQLContext sqlContext)
{
     try
     {

         File outputFile = new File(getServletContext().getRealPath("/")
                    + "/q3.csv");

     FileWriter fw= new FileWriter(outputFile);

    JavaSchemaRDD count = sqlContext.sql("SELECT user.name,user.statuses_count AS c FROM tweetTable " +
                                        "ORDER BY c");


   List<org.apache.spark.sql.api.java.Row> rows = count.collect();

   Collections.reverse(rows);

    String rows123=rows.toString();


   String[] array = rows123.split("],");

    System.out.println(rows123);

    fw.append("Name");
    fw.append(',');
    fw.append("Count");
    fw.append("\n");
```

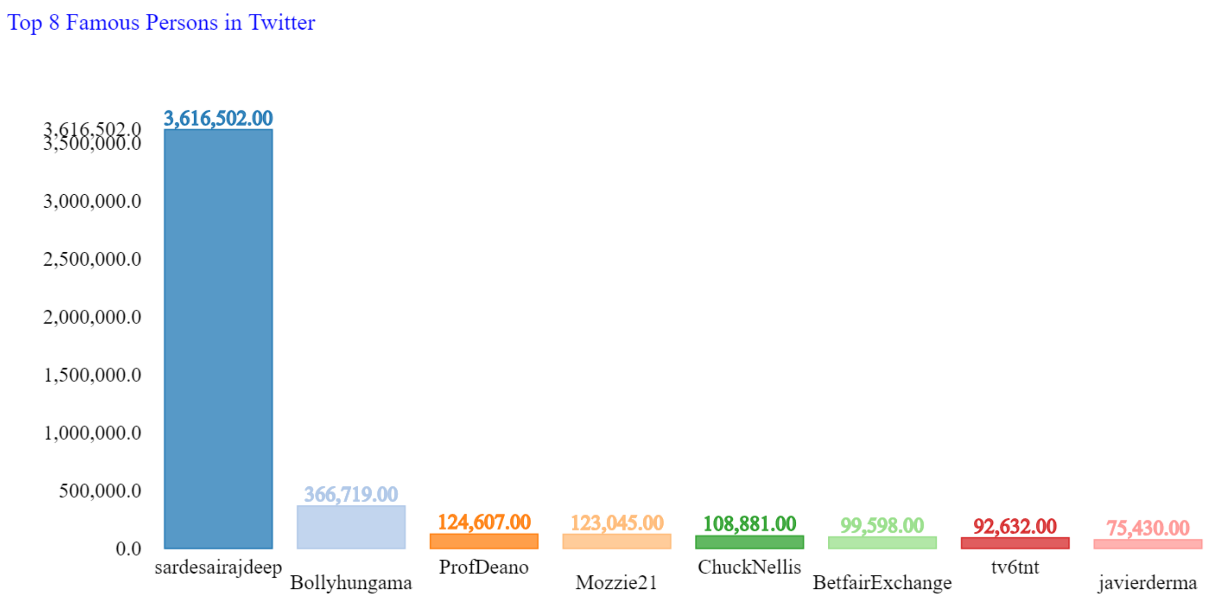


Query 3: Users with high follower count tweeting about IPL

Users on twitter with highest follower count and tweeting about IPL are traced using below query. We use bar graphs for visualizing these users and there follower count statistics.

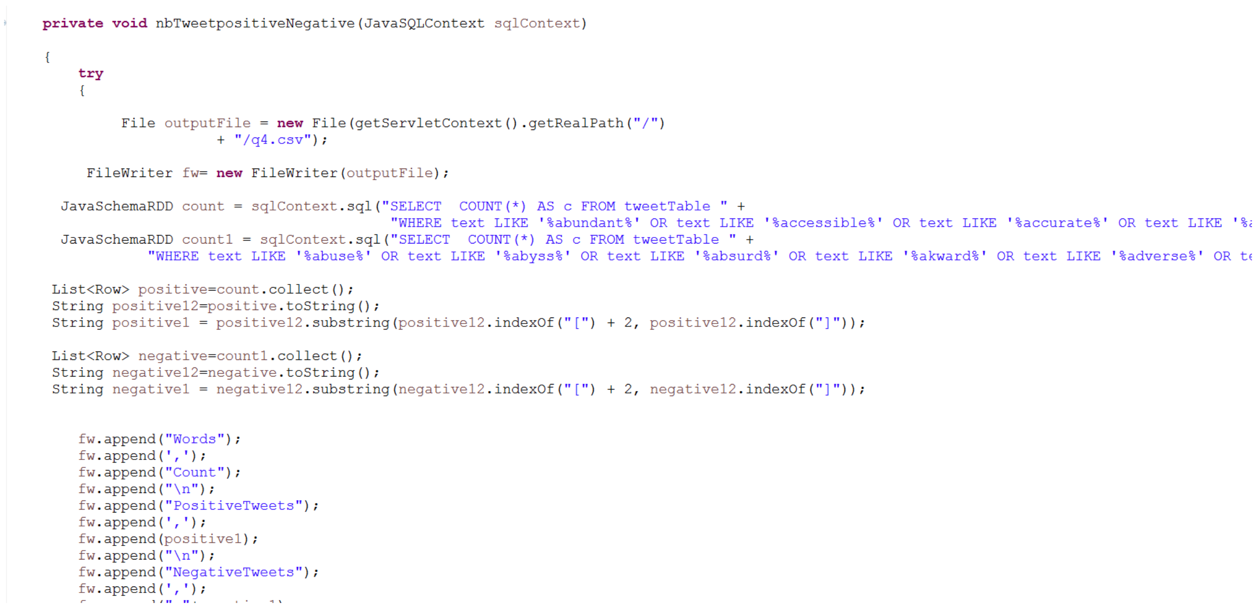


```
    private void nbIPLPopular(JavaSQLContext sqlContext) {

        try
        {
      File outputFile = new File(getServletContext().getRealPath("/")
                      + "/q2.csv");

       FileWriter fw= new FileWriter(outputFile);

      JavaSchemaRDD count = sqlContext.sql("SELECT DISTINCT  user.screen_name, user.followers_count AS c FROM tweetTable " +
                                          "ORDER BY c");

      List<org.apache.spark.sql.api.java.Row> rows = count.collect();

         Collections.reverse(rows);

         String rows123=rows.toString();

         String[] array = rows123.split("],");


      fw.append("Name");
      fw.append(',');
      fw.append("Count");
      fw.append("\n");


      for(int i = 0; i < 8; i++)
      {
          if(i==0)
```

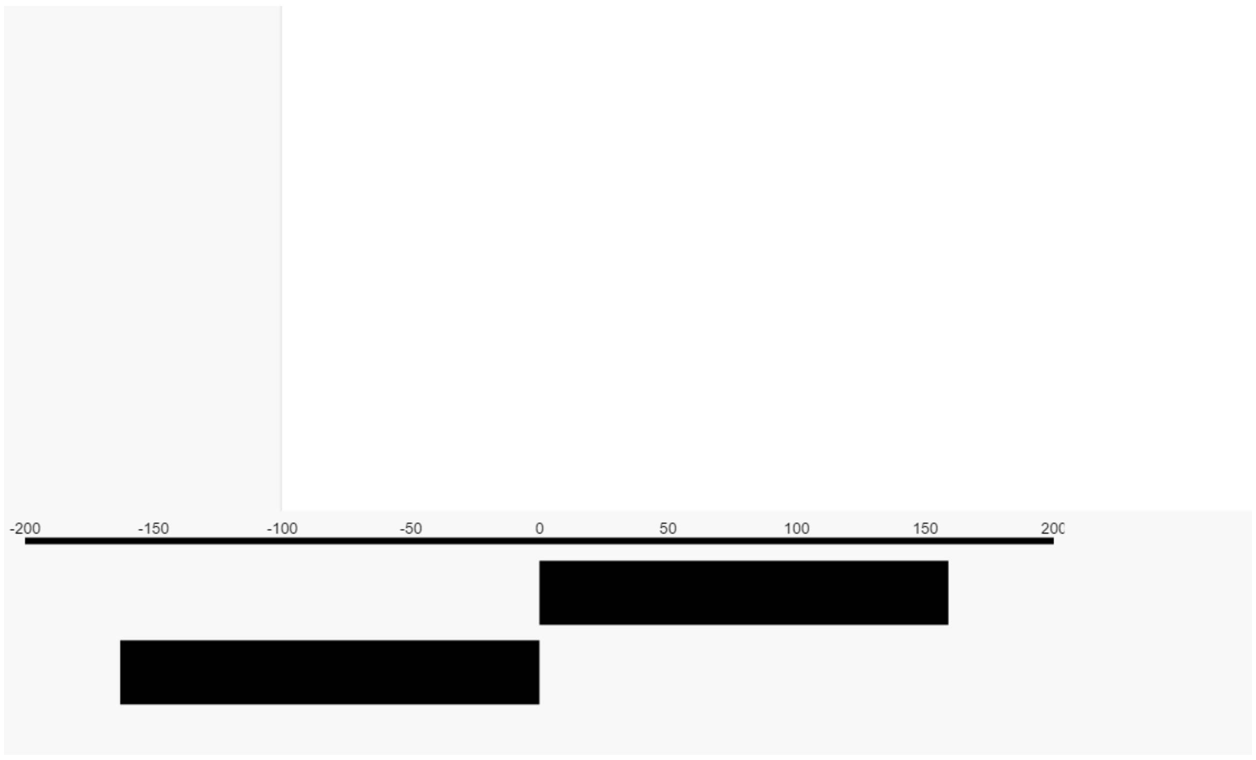


Query 4: Positive and Negative Tweets stats about IPL on Twitter

The comparison of positive and negative tweets is done using sentimental analysis on twitter data by some key words which exhibit the feeling. The comparative bar graph is used to visualize these stats.

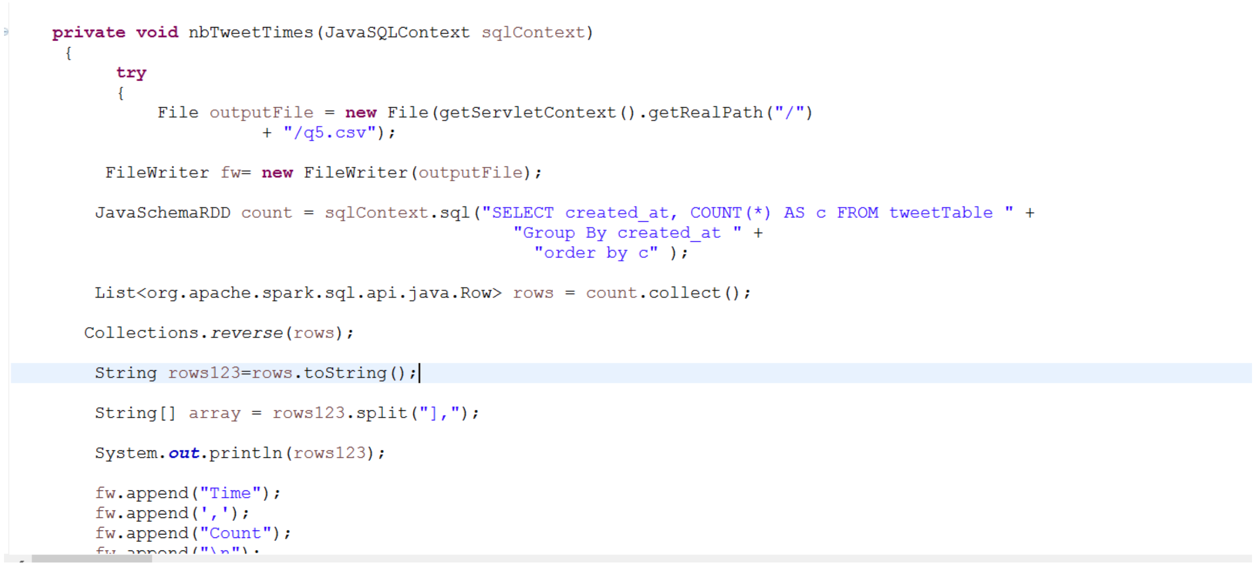


```
    private void nbTweetpositiveNegative(JavaSQLContext sqlContext)

    {
        try
        {

             File outputFile = new File(getServletContext().getRealPath("/")
                         + "/q4.csv");

         FileWriter fw= new FileWriter(outputFile);

      JavaSchemaRDD count = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable " +
                                         "WHERE text LIKE '%abundant%' OR text LIKE '%accessible%' OR text LIKE '%accurate%' OR text LIKE '%a
      JavaSchemaRDD count1 = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable " +
                "WHERE text LIKE '%abuse%' OR text LIKE '%abyss%' OR text LIKE '%absurd%' OR text LIKE '%akward%' OR text LIKE '%adverse%' OR te

     List<Row> positive=count.collect();
     String positive12=positive.toString();
     String positive1 = positive12.substring(positive12.indexOf("[") + 2, positive12.indexOf("]"));

     List<Row> negative=count1.collect();
     String negative12=negative.toString();
     String negative1 = negative12.substring(negative12.indexOf("[") + 2, negative12.indexOf("]"));


        fw.append("Words");
        fw.append(',');
        fw.append("Count");
        fw.append("\n");
        fw.append("PositiveTweets");
        fw.append(',');
        fw.append(positive1);
        fw.append("\n");
        fw.append("NegativeTweets");
        fw.append(',');
```



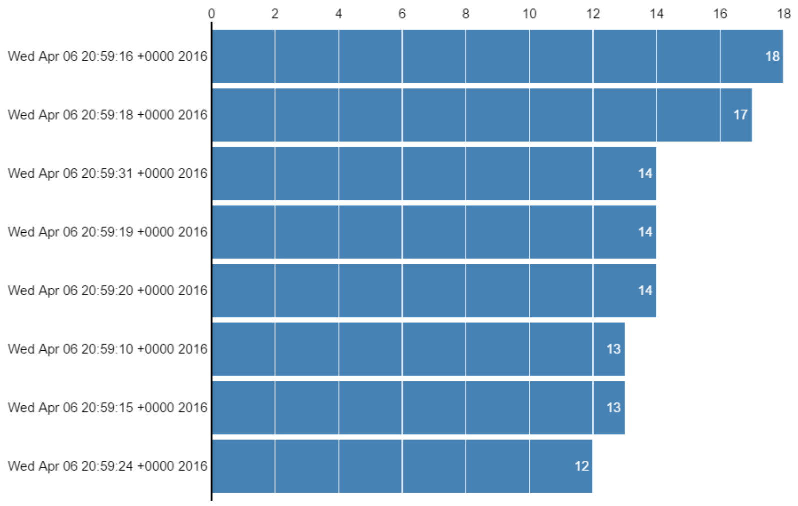
Query 5: Most Common timings of users tweeting about IPL on twitter

The time around clock when the users are more actively tweeting about IPL can be traced using the below query. The Horizontal bar graphs are used to visualize these stats.



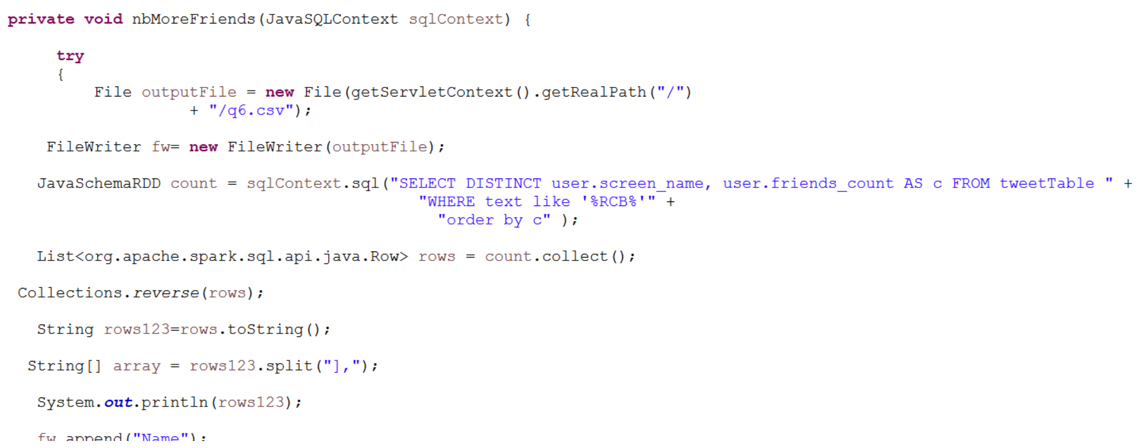
```
private void nbTweetTimes(JavaSQLContext sqlContext)
 {
        try
        {
            File outputFile = new File(getServletContext().getRealPath("/")
                      + "/q5.csv");

        FileWriter fw= new FileWriter(outputFile);

       JavaSchemaRDD count = sqlContext.sql("SELECT created_at, COUNT(*) AS c FROM tweetTable " +
                                           "Group By created_at " +
                                             "order by c" );

       List<org.apache.spark.sql.api.java.Row> rows = count.collect();

      Collections.reverse(rows);

       String rows123=rows.toString();

       String[] array = rows123.split("],");

       System.out.println(rows123);

       fw.append("Time");
       fw.append(',');
       fw.append("Count");
```

Top 8 Most Twitting Times



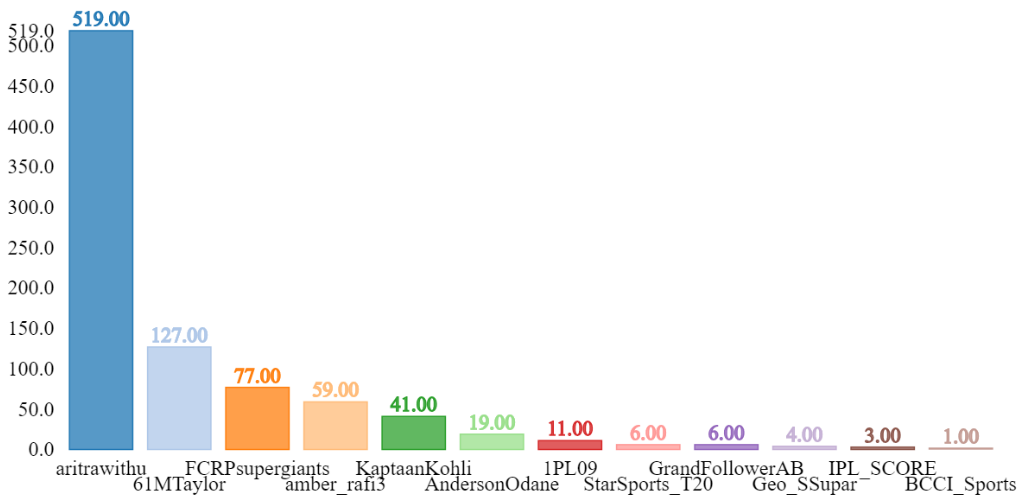
Query 6: Users with Highest follower count supporting a single IPL Team

Users with highest follower count who are tweeting about a particular IPL can be traced using this query. We used Bar graphs to visualize these stats.



```
private void nbMoreFriends(JavaSQLContext sqlContext) {

     try
     {
         File outputFile = new File(getServletContext().getRealPath("/")
                   + "/q6.csv");

     FileWriter fw= new FileWriter(outputFile);

    JavaSchemaRDD count = sqlContext.sql("SELECT DISTINCT user.screen_name, user.friends_count AS c FROM tweetTable " +
                                         "WHERE text like '%RCB%'" +
                                           "order by c" );

    List<org.apache.spark.sql.api.java.Row> rows = count.collect();

  Collections.reverse(rows);

    String rows123=rows.toString();

   String[] array = rows123.split("],");

    System.out.println(rows123);

    fw.append("Name");
```

The Users having Friends Greater than 300000

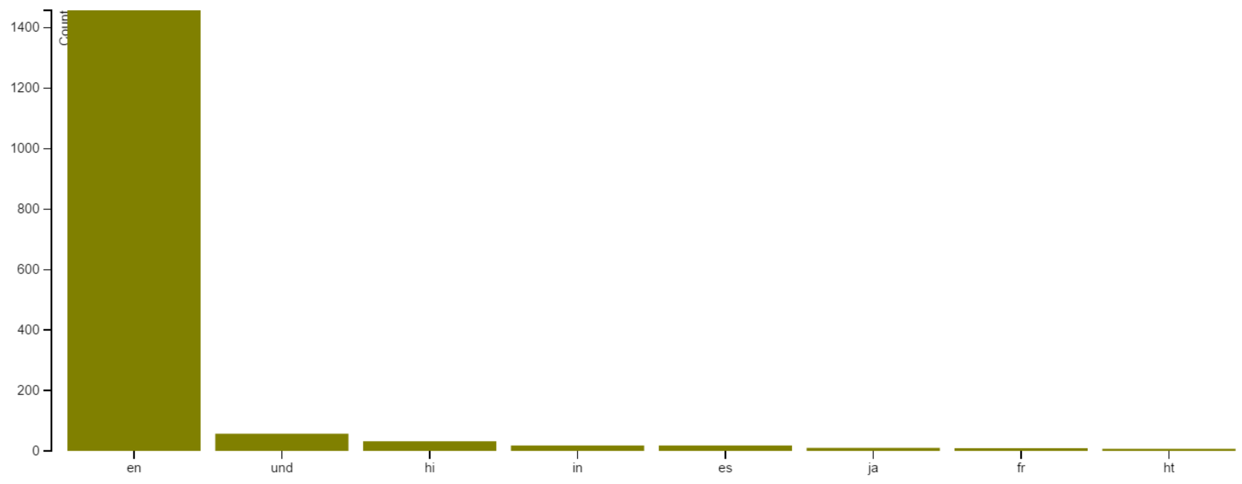


Query 7: Language which has highest tweets about IPL

We can trace out the languages which are high on IPL tweets using the below query. These are visualized using Bar Graphs.

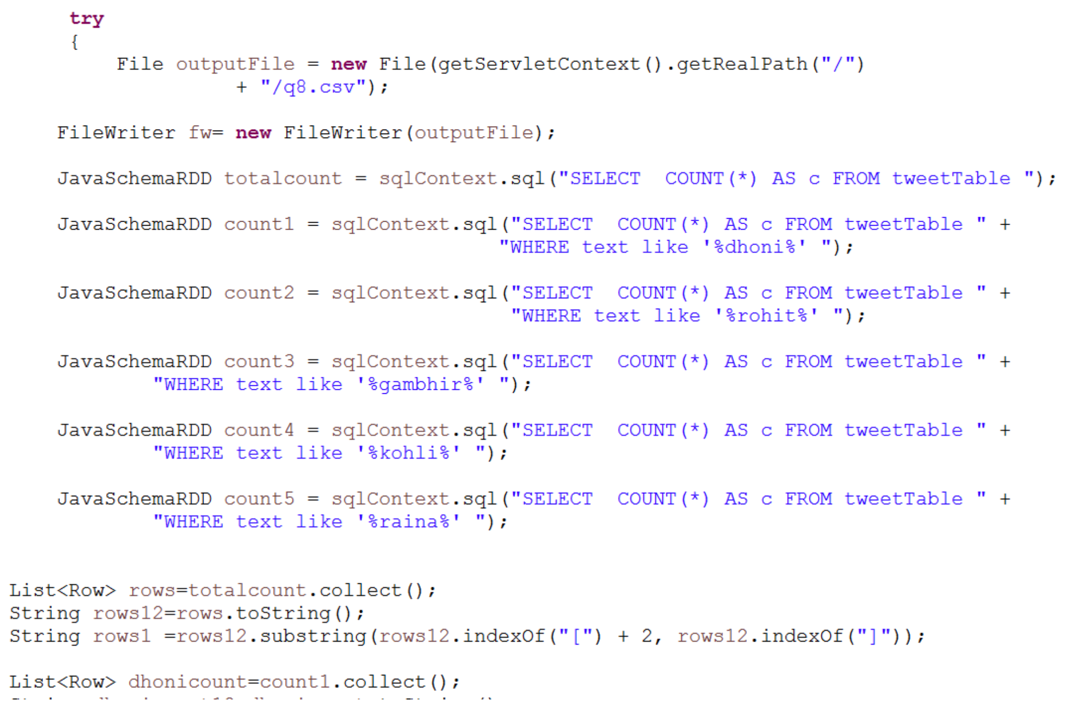
```
private void nbLanguage(JavaSQLContext sqlContext) {

     try
     {
         File outputFile = new File(getServletContext().getRealPath("/")
                   + "/q7.csv");

     FileWriter fw= new FileWriter(outputFile);

    JavaSchemaRDD count = sqlContext.sql("SELECT lang, COUNT(*) AS c FROM tweetTable " +
                                        "Group By lang " +
                                          "order by c");

    List<org.apache.spark.sql.api.java.Row> rows = count.collect();

Collections.reverse(rows);

    String rows123=rows.toString();

   String[] array = rows123.split("],");

    System.out.println(rows123);

    fw.append("Language");
    fw.append(',');
    fw.append("Count");
    fw.append("\n");


    for(int i = 0; i < 9; i++)
```

Top 8 Twitting Languages



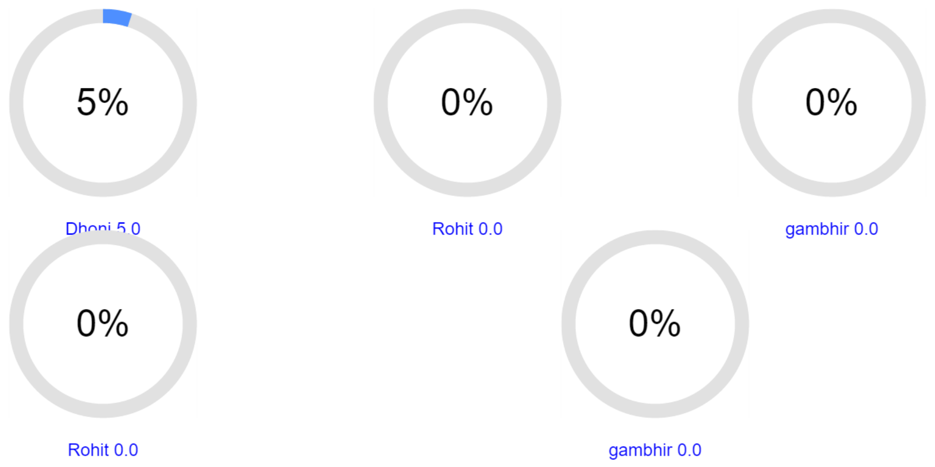
Query 8: Team Captains and there following on twitter

The tweet count for each IPL team captain are retrieved and are compared using bar graph. These are visualized using circular graphs.



```
     try
     {
         File outputFile = new File(getServletContext().getRealPath("/")
                 + "/q8.csv");

    FileWriter fw= new FileWriter(outputFile);

    JavaSchemaRDD totalcount = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable ");

    JavaSchemaRDD count1 = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable " +
                                       "WHERE text like '%dhoni%' ");

    JavaSchemaRDD count2 = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable " +
                                        "WHERE text like '%rohit%' ");

    JavaSchemaRDD count3 = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable " +
            "WHERE text like '%gambhir%' ");

    JavaSchemaRDD count4 = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable " +
            "WHERE text like '%kohli%' ");

    JavaSchemaRDD count5 = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable " +
            "WHERE text like '%raina%' ");


List<Row> rows=totalcount.collect();
String rows12=rows.toString();
String rows1 =rows12.substring(rows12.indexOf("[") + 2, rows12.indexOf("]"));

List<Row> dhonicount=count1.collect();
```

Total Tweets: 264590



References:

https://github.com/AshokYaganti/PBBlumixCode

https://github.com/stefani75/workspace/tree/b90a63f2f3028fb358b28a77ac416b223d37a52a/projet1/Hands-On-Spark-java-solution/src/main/java/com/duchessfr/spark