

# **PRINCIPLES OF BIG DATA**

## **PROJECT PHASE 3- FINAL REPORT**

**(Topic: ICC World Cup 2015)**



**Submitted By:**

Pooja Agrawal  
Sathyadhari Yenugu  
Sirisha Valluri

# Introduction:

Big data analytics is the process of examining large data sets containing a variety of data types -- i.e., big data -- to uncover hidden patterns, unknown correlations, market trends, customer preferences and other useful business information. With big data analytics, data scientists and others can analyze huge volumes of data that conventional analytics and business intelligence solutions can't touch.

# Project Description:

Here in this project we have first collected the tweets on the ICC World Cup 2015 based on the game schedules and then performed some analysis on collected data using big data tool HIVE by performing analytical queries on it. Finally we have used some visualizations like User Interface page and Big sheets tools to display the output.

# Software Modules:

The implementation of this project is divided into four modules. These are:

## **1. DATA COLLECTION:**

We collected tweets from twitter and stored them in the form of text files. The collection of tweets is done using Twitter 4j API in java. The search is done base on the following keywords: ‘ World Cup 2015 ‘, ‘2015 World Cup’, ‘ WorldCup2015 ’ and ‘2015WorldCup’.

## **2. DATA MODELLING:**

We have modeled the tweet data into below mentioned fields for our analysis and stored them into 3 text files **TweetsInfo.txt**, **UserInfo.txt** and **TweetUser.txt**.

1. Tweet ID
2. Tweet Content
3. Tweeted Date& Time
4. Re tweeted count
5. Favorites count
6. User ID
7. User name
8. User location
9. User Country
10. Followers Count
11. Account type

### 3) DATA PROCESSING:

For analyzing, first we have loaded the data into the hive database. The tweets collected were loaded into three different tables into hive database:

- tweetinfo - Contains all information about tweets.
- userinfo - Contains user information.
- tweetuser - Contains tweet and user id's.

#### Queries for table creation:

Creation of above three tables with the following syntax:

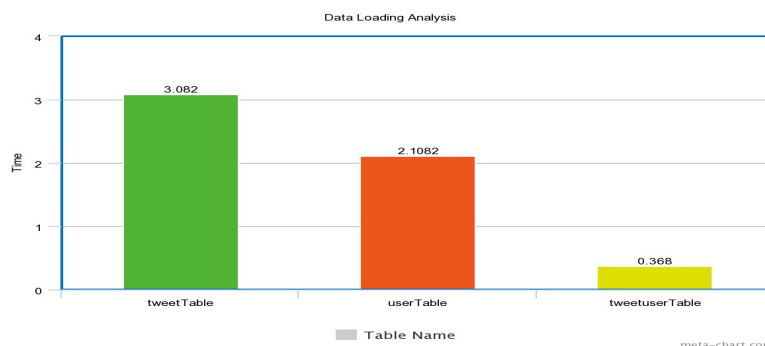
- tweetinfo table:  
Create table tweetuser(userid string,tweetid string) row format delimited fields terminated by '\t';
- Userinfo table:  
Create table userinfo(userid string,username string,country string, location string,followerscount int,isverified boolean,imageurl string) row format delimited fields terminated by '\t';
- tweetuser table:  
Create table tweetinfo(tweetid string,favouritecount int,retweetcount int,date string,tweetcontent string) row format delimited fields terminated by '\t';

#### Queries for data loading:

Loading text files into above three tables with the following syntax:

- load data local inpath '/home/biadmin/Desktop/TweetInfo.txt' into table tweetinfo;
- load data local inpath '/home/biadmin/Desktop/UserInfo.txt' into table userinfo;
- load data local inpath '/home/biadmin/Desktop/TweetUser.txt' into table tweetuser;

#### Time Analysis for data loading:



# Analytical Queries:

## Query1: - Based on Location

In this, we will find the number of tweets depending on the location that are tweeting more about ICC World Cup 2015 and making an analysis that in which are the top 5 countries in the world where cricket is more followed.

Below table explains the process of creating a table for storing the results of the query, inserting the query results in the table:

TASK	COMMAND
Creating an external table for storing query results.	<code>create external table topcountrytweet(country string,count int) row format delimited fields terminated by '\t' stored as textfile location '/home/biadmin/LOCATION/topcountrytweet';</code>
Running the query and storing the results in external table	<code>insert overwrite table topcountrytweet <i>select a.location,count(distinct b.tweetid) AS tweetcount from userinfo a join tweetuser b on a.userid=b.userid group by a.location order by tweetcount desc limit 5;</i></code>

## **Explanation:**

Here we used JOIN operator to join the tables userinfo and tweetuser to get the distinct count of tweet id's using the keyword DISTINCT, location from where the tweet is tweeted. As, we are retrieving more than one column with the aggregate function COUNT we have to use GROUP BY clause in order to get top 5 locations. Limit gives the top 5 results.

## Screenshots:

```
hive> insert overwrite table topcountrytweet select a.location,count(distinct b.tweetid) AS tweetcount from userinfo a join tweetuser b on a.userid=b.userid
group by a.location order by tweetcount desc limit 5;
Total MapReduce jobs = 2
Execution log at: /var/ibm/biginsights/hive/logs/biadmin/.log
2015-04-30 02:57:29 Starting to launch local task to process map join; maximum memory = 536870912
2015-04-30 02:57:30 Dump the side-table into file: file:/tmp/biadmin/hive_2015-04-30_02-57-25_819_1605217829351842590-1/-local-10004/HashTable-Stage-2/MapJoin-mapfile21--.hashtable
2015-04-30 02:57:31 Upload 1 File to: file:/tmp/biadmin/hive_2015-04-30_02-57-25_819_1605217829351842590-1/-local-10004/HashTable-Stage-2/MapJoin-mapfile21--.hashtable
2015-04-30 02:57:31 End of local task; Time Taken: 1.212 sec.
Execution completed successfully
Mapred Local Task Succeeded . Convert the Join into MapJoin
Mapred Local Task Succeeded . Convert the Join into MapJoin
Launching Job 1 out of 2
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:
    set mapred.reduce.tasks=<number>
Starting Job = job_201504291550_0039, Tracking URL = http://bivm.ibm.com:50030/jobdetails.jsp?jobid=job_201504291550_0039
Kill Command = /opt/ibm/biginsights/IHC/bin/hadoop job -kill job_201504291550_0039
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1

2015-04-30 02:58:46,894 Stage-3 map = 0%, reduce = 0%
2015-04-30 02:58:48,907 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 1.65 sec
2015-04-30 02:58:49,913 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 1.65 sec
2015-04-30 02:58:50,922 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 1.65 sec
2015-04-30 02:58:51,928 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 1.65 sec
2015-04-30 02:58:52,933 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 1.65 sec
2015-04-30 02:58:53,939 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 1.65 sec
2015-04-30 02:58:54,945 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 1.65 sec
2015-04-30 02:58:55,950 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 1.65 sec
2015-04-30 02:58:56,956 Stage-3 map = 100%, reduce = 33%, Cumulative CPU 1.65 sec
2015-04-30 02:58:57,962 Stage-3 map = 100%, reduce = 100%, Cumulative CPU 9.57 sec
2015-04-30 02:58:58,968 Stage-3 map = 100%, reduce = 100%, Cumulative CPU 9.57 sec
2015-04-30 02:58:59,974 Stage-3 map = 100%, reduce = 100%, Cumulative CPU 9.57 sec
MapReduce Total cumulative CPU time: 9 seconds 570 msec
Ended Job = job_201504291550_0040
Loading data to table default.topcountrytweet
rmr: DEPRECATED: Please use 'rm -r' instead.
Deleted /home/biadmin/LOCATION/topcountrytweet
MapReduce Jobs Launched:
Job 0: Map: 1 Reduce: 1 Cumulative CPU: 58.14 sec HDFS Read: 13722152 HDFS Write: 2126 SUCCESS
Job 1: Map: 1 Reduce: 1 Cumulative CPU: 9.57 sec HDFS Read: 2495 HDFS Write: 117 SUCCESS
Total MapReduce CPU Time Spent: 1 minutes 7 seconds 710 msec
OK
```

## Query2: - Based on Frequency

For a particular game, we are planning to analyze the number of tweets posted before the game, during the game and after the game. Here we are using 3 dates for our analyses: March 28<sup>th</sup>, March 29<sup>th</sup> and March 30<sup>th</sup>. (This was the final match schedule which was held on March 29<sup>th</sup>).

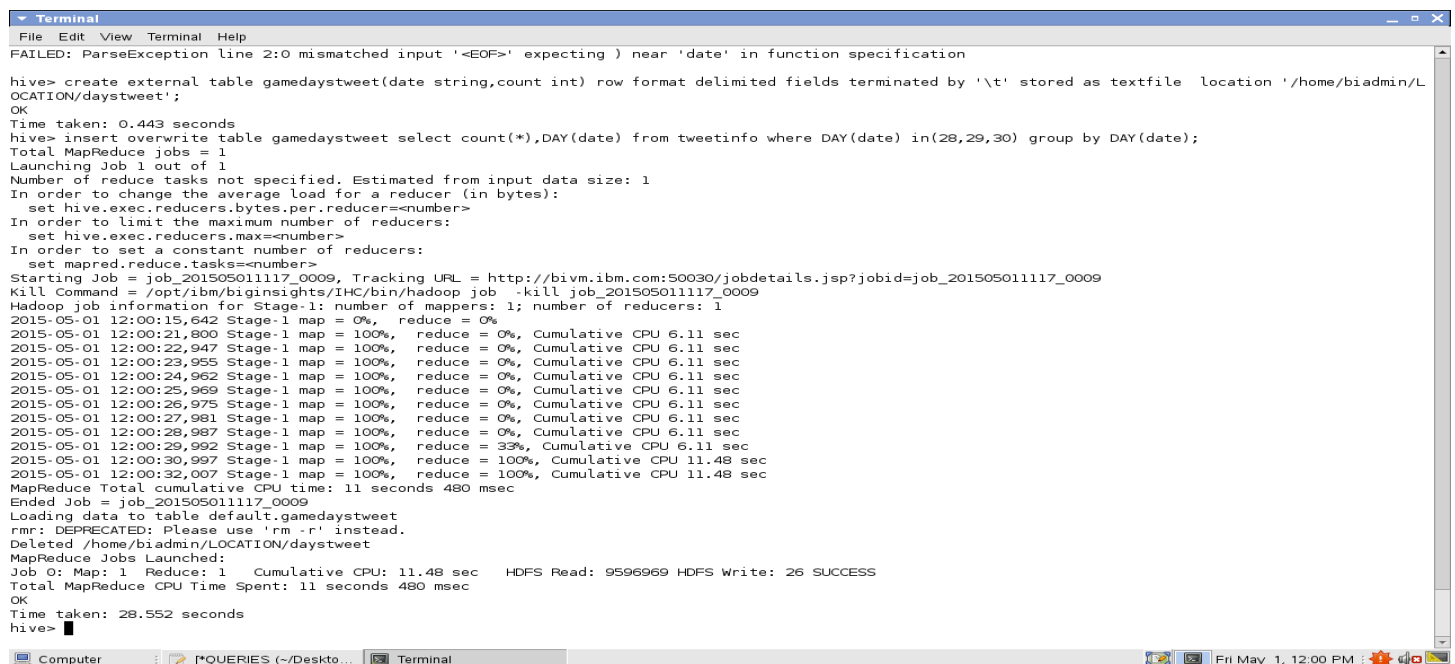
Below table explains the process of creating a table for storing the results of the query, inserting the query results in the table:

TASK	COMMAND
Creating an external table for storing query results.	create external table gamedaystweet(date string,count int) row format delimited fields terminated by '\t' stored as textfile location '/home/biadmin/LOCATION/daystweet';
Running the query and storing the results in external table	insert overwrite table gamedaystweet <i>select count(*),DAY(date) from tweetinfo where DAY(date) in(28,29,30) group by DAY(date);</i>

## Explanation:

Here we used DAY function in the query to get the date of out the complete date format and searching is done by using 3 values i.e 28, 29, 30(day before game, game day, day after the game) and finally using GROUP BY.

## Screenshots:



```
Terminal
File Edit View Terminal Help
FAILED: ParseException line 2:0 mismatched input '<EOF>' expecting ')' near 'date' in function specification

hive> create external table gamedaystweet(date string,count int) row format delimited fields terminated by '\t' stored as textfile location '/home/biadmin/LOCATION/daystweet';
OK
Time taken: 0.443 seconds
hive> insert overwrite table gamedaystweet select count(*),DAY(date) from tweetinfo where DAY(date) in(28,29,30) group by DAY(date);
Total MapReduce 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:
  set mapred.reduce.tasks=<number>
Starting Job = job_20150501117_0009, Tracking URL = http://bivm.ibm.com:50030/jobdetails.jsp?jobid=job_20150501117_0009
Kill Command = /opt/ibm/biginsights/IHC/bin/hadoop job -kill job_20150501117_0009
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2015-05-01 12:00:15,642 Stage-1 map = 0%, reduce = 0%
2015-05-01 12:00:21,800 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 6.11 sec
2015-05-01 12:00:22,947 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 6.11 sec
2015-05-01 12:00:23,955 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 6.11 sec
2015-05-01 12:00:24,962 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 6.11 sec
2015-05-01 12:00:25,969 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 6.11 sec
2015-05-01 12:00:26,975 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 6.11 sec
2015-05-01 12:00:27,981 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 6.11 sec
2015-05-01 12:00:28,987 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 6.11 sec
2015-05-01 12:00:29,992 Stage-1 map = 100%, reduce = 33%, Cumulative CPU 6.11 sec
2015-05-01 12:00:30,997 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 11.48 sec
2015-05-01 12:00:32,007 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 11.48 sec
MapReduce Total cumulative CPU time: 11 seconds 480 msec
Ended Job = job_20150501117_0009
Loading data to table default.gamedaystweet
rmr: DEPRECATED: Please use 'rm -r' instead.
Deleted /home/biadmin/LOCATION/daystweet
MapReduce Jobs Launched:
Job 0: Map: 1 Reduce: 1 Cumulative CPU: 11.48 sec HDFS Read: 9596969 HDFS Write: 26 SUCCESS
Total MapReduce CPU Time Spent: 11 seconds 480 msec
OK
Time taken: 28.552 seconds
hive>
```

### Query3: - Based on Teams

In this, we will find the number of tweets for the teams of ICC World Cup 2015 and making an analysis that which are the top 5 teams that are most popular.

Below table explains the steps how the user-defined function is created and used:

TASK	COMMAND
Create a java class	TeamCustomFunction.java is the java class containing the evaluation method.
Export this class as jar file and place at a location.	/home/biadmin/TweetJars.jar
Add this jar to HIVE	add jar /home/biadmin/TweetJars.jar;
Create User Defined Function	create temporary function team_custom_function as 'com.example3.hive.udf.TeamCustomFunction';
Create an internal table for storing output	create table teamquery(teamname string);
Store data in table	insert into table teamquery <i>select team_custom_function(tweetcontent) from tweetinfo;</i>
Creating an external table for storing query results	create external table result_teamquery(teamname string,count int) row format delimited fields terminated by '\t' stored as textfile location '/home/biadmin/LOCATION/result_teamquery';
Running the query and storing the results in external table	insert overwrite table result_teamquery <i>select teamname,count(*)as tweetcount from teamquery where teamname != 'NULL' group by teamname order by tweetcount desc limit 5 ;</i>

### Explanation:

Here in this query we are retrieving tweet count and team name from the teamquery table and using “!=” function in order to avoid any null values in the result. Finally using GROUP BY function. Limit gives the top 5 results.

## **Query4: - Based on Players**

In this, we will find the number of tweets for the player of ICC World Cup 2015 and making an analysis that which are the top 5 player that are most popular.

Below table explains the steps how the user-defined function is created and used:

<b>TASK</b>	<b>COMMAND</b>
Create a java class	PlayerCustomFunction.java is the java class containing the evaluation method.
Export this class as jar file and place at a location.	/home/biadmin/TweetJars.jar
Add this jar to HIVE	add jar /home/biadmin/TweetJars.jar;
Create User Defined Function	create temporary function player_custom_function as 'com.example2.hive.udf.PlayerCustomFunction';
Create an internal table for storing output	create table playerquery(playername string);
Store data in table	insert into table playerquery <i>select player_custom_function(tweetcontent) from tweetinfo;</i>
Creating an external table for storing query results	create external table result_playerquery(playername string,count int) row format delimited fields terminated by '\t' stored as textfile location '/home/biadmin/LOCATION/result_playerquery';
Running the query and storing the results in external table	insert overwrite table result_playerquery <i>select playername,count(*)as tweetcount from playerquery where playername != 'NULL' group by playername order by tweetcount desc limit 5 ;</i>

### **Explanation:**

Here in this query we are retrieving tweet count and player name from the playerquery table and using “!=” function in order to avoid any null values in the result. Finally using GROUP BY function. Limit gives the top 5 results.



## Query5: - Based on Time

Here we will be analyzing at what time of day we have maximum number of tweet. We will analyze the no. of tweets coming in morning, noon, evening and night.

Below table explains the steps how the user-defined function is created and used:

TASK	COMMAND
Create a java class	DateCustomFunction.java is the java class containing the evaluation method.
Export this class as jar file and place at a location.	/home/biadmin/TweetJars.jar
Add this jar to HIVE	add jar /home/biadmin/TweetJars.jar;
Create User Defined Function	create temporary function date_custom_function as 'com.example.hive.udf.DateCustomFunction';
Create an internal table for storing output	create table timequery(tweetid string,count int);
Store data in table	insert overwrite table timequery <i>select distinct tweetid,date_custom_function(date) as count from tweetinfo;</i>
Creating an external table for storing query results	create external table result_timequery(timeinterval string,count int) row format delimited fields terminated by '\t' stored as textfile location '/home/biadmin/LOCATION/result_timequery';
Running the query and storing the results in external table	insert overwrite table result_timequery <i>select count,count(*) from timequery group by count having count not in(0);</i>

### Explanation:

Here in this query, we have created a user defined function in Hive that takes Date from tweetinfo as input and after processing gives the output. The output contains the duration of the day in which the tweet was posted.

## **Query6: - Most Popular Tweet**

In this, we are determining the count of most popular tweet related to ICC World Cup 2015 and displaying it to User Interface with the User name, Tweet Content and Profile picture.

In order to get this, we have calculated the sum of favorite count and retweet count and considered that tweet as the most popular tweet.

NOTE: Since this is the query that we are displaying through UI (means we are not storing the output but fetching the output on run-time) and was taking long time to get the output. So in order to get the output fast( for DEMO feasibility) we have created another 3 tables with only 1000 records and named them as **tweetinfo\_demo**, **tweetuser\_demo**, **userinfo\_demo**.

Below is the query that we have created to achieve this:

```
select c.username, a.tweetcontent,(a.retweetcount+a.favouritecount) as sum ,c.imageurl from tweetinfo_demo  
a join tweetuser_demo b on a.tweetid=b.tweetid join userinfo_demo c on c.userid=b.userid order by sum desc  
limit 1;
```

### **Explanation:**

Here we are retrieving three columns username, tweetcontent and (favouritecount + retweetcount) by joining tables tweetinfo\_demo, tweetuser\_demo, userinfo\_demo. We have used order by desc clause which retrieves the result in descending order. Here limit 1 retrieves only one row.

## **Query7: - Celebrity Account**

In this, we are finding the count celebrity account and the non-celebrity account and displaying it to User Interface.

Below is the query that we have created to achieve this:

```
select isverified,count(*) from userinfo where isverified !='NULL' group by isverified;
```

### **Explanation:**

Here, we are calculating the number of tweets from the field isverified which has Boolean values(either TRUE or FALSE). The above query retrieves the result with the number of tweets from the profiles who have isverified column true and the number of tweets from the profiles who have isverified column false. Here True values represents the Celebrity account and False values represents the Non-Celebrity accounts.

#### 4) DATA VISUALIZATION:

For the visualization of the output we have used **Bigsheets tool**. Also for better and creative visualization of the results we have created **jsp page** and call servlet from that jsp page. From the servlet we have given connection to hive database using jdbc connection drivers.

Out of 7 queries, 5 queries(Location based, Time based, Team based, Player based and Frequency based) we have stored as Bigsheets output and remaining 2 queries(Most popular tweet and Celebrity Account) we are directly displaying result on UI through a jsp page where we are calling servlet and from the servlet we have given connection to hive database.

#### Output Screenshots

##### 0) Output of our jsp page:

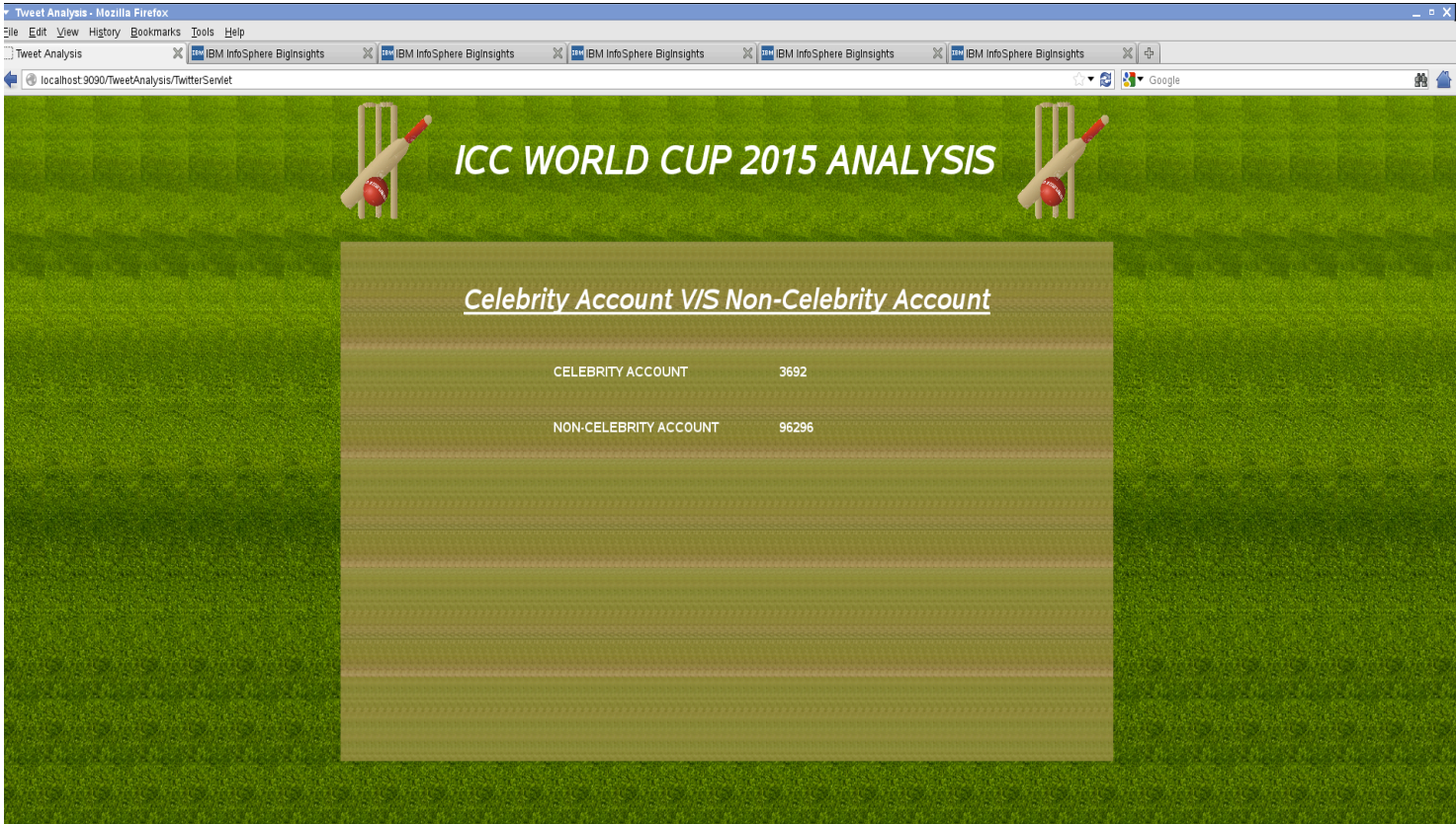




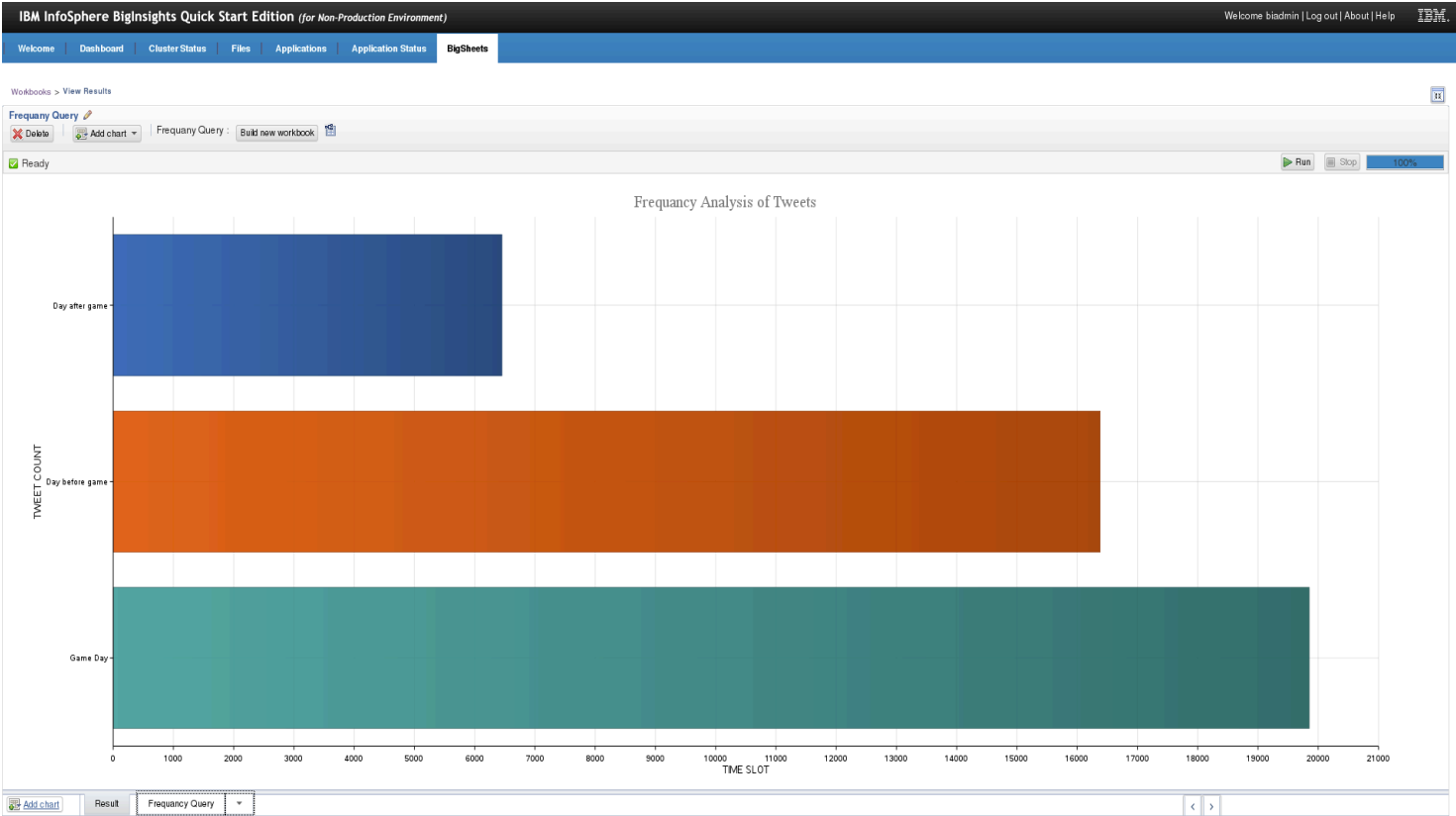
### 1) Output of Most Popular Tweet Query: (User Interface)



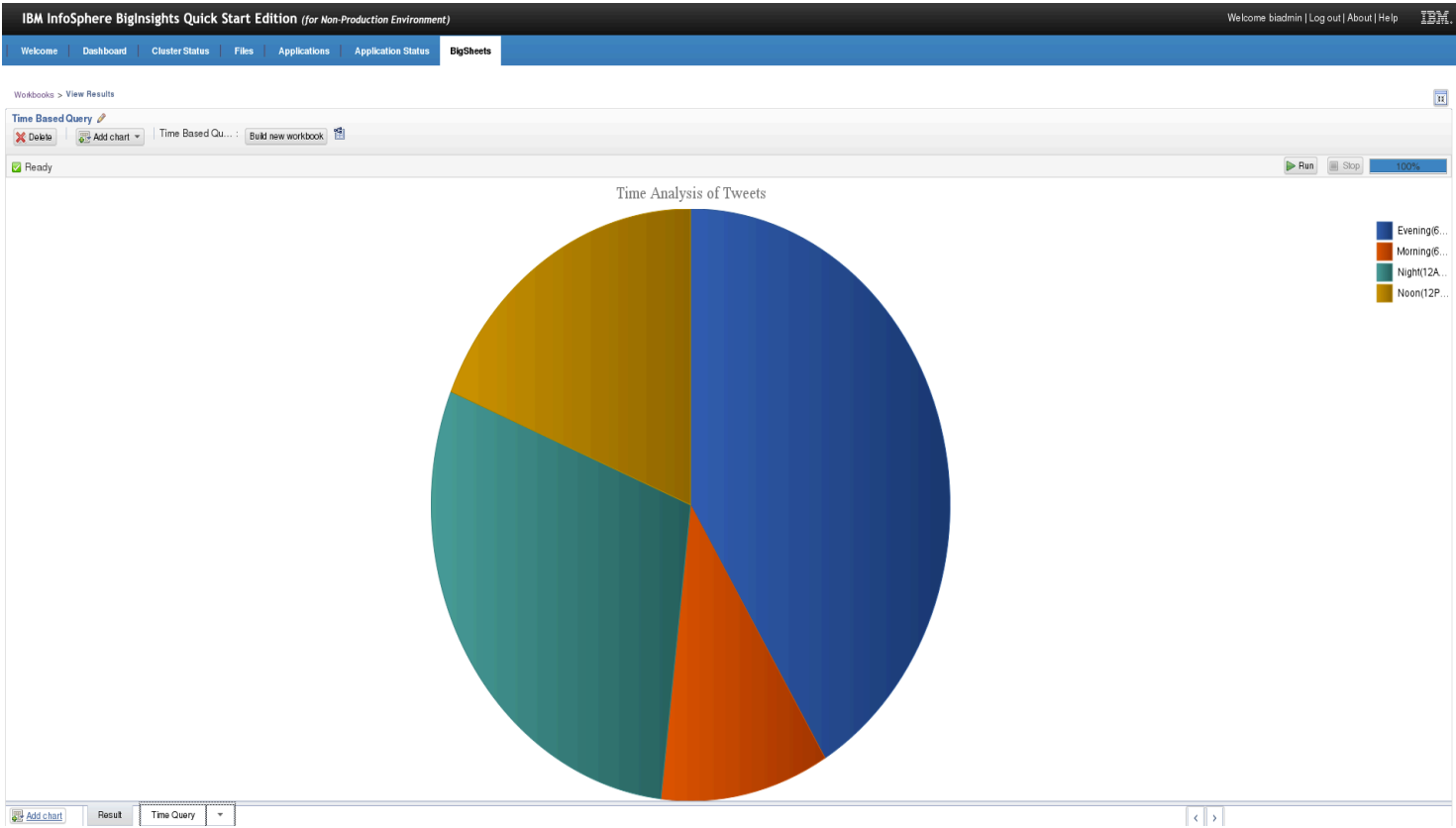
## 2) Output of Celebrity Account Query: (User Interface)



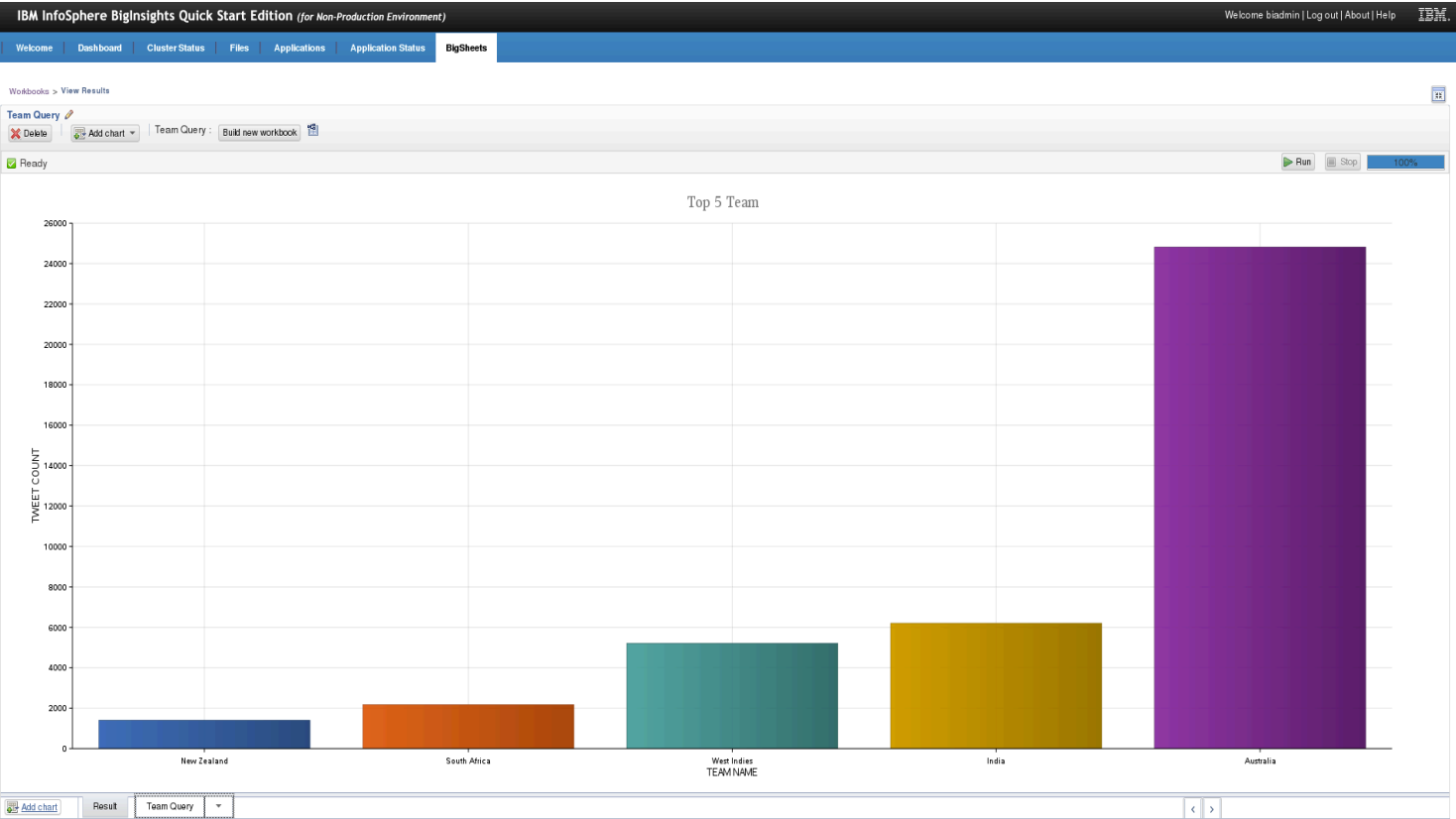
### 3) Output of Frequency Analysis Query: (Bigsheets Result)



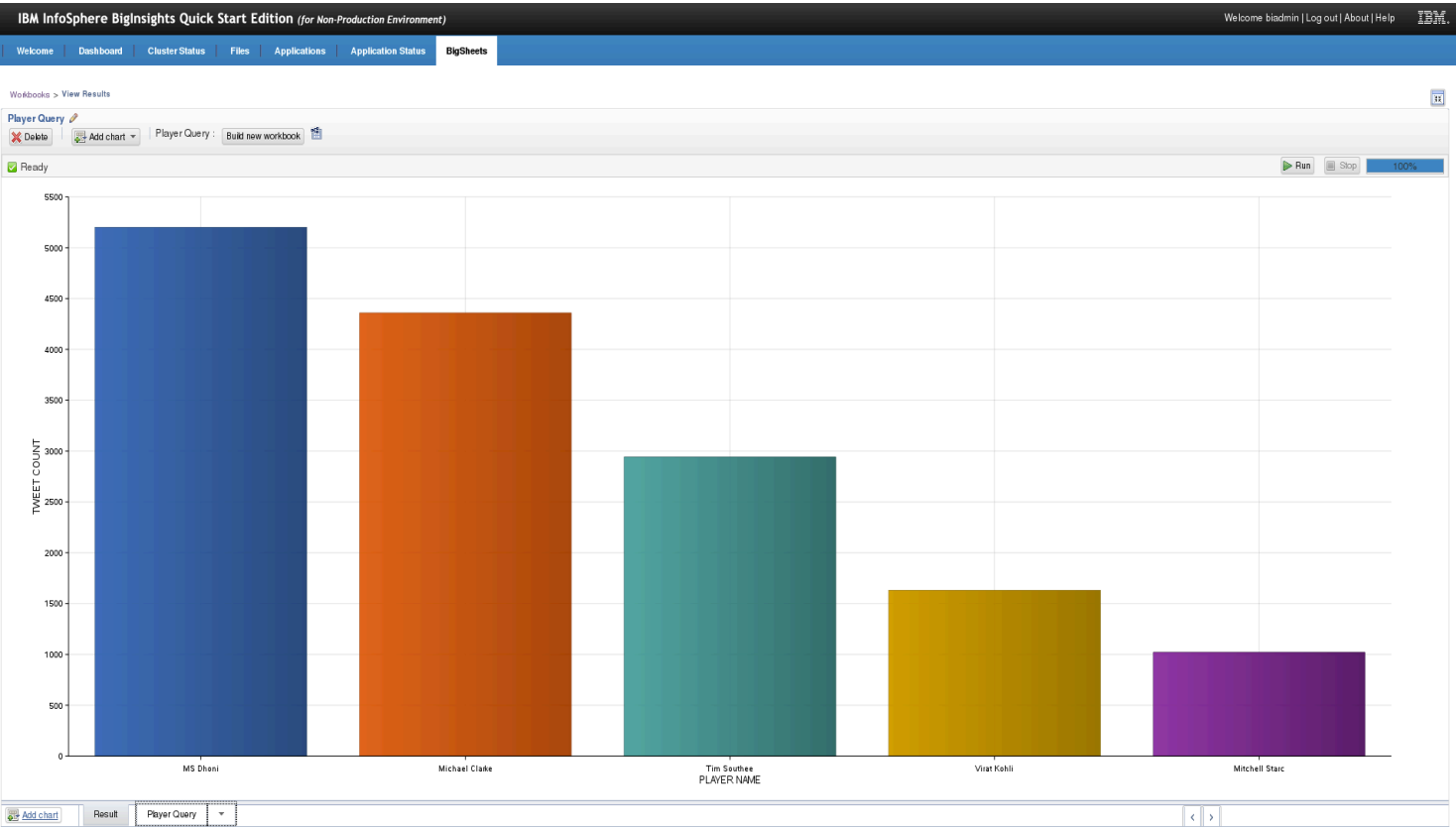
### 4) Output of Time Analysis Query: (Bigsheets Result)



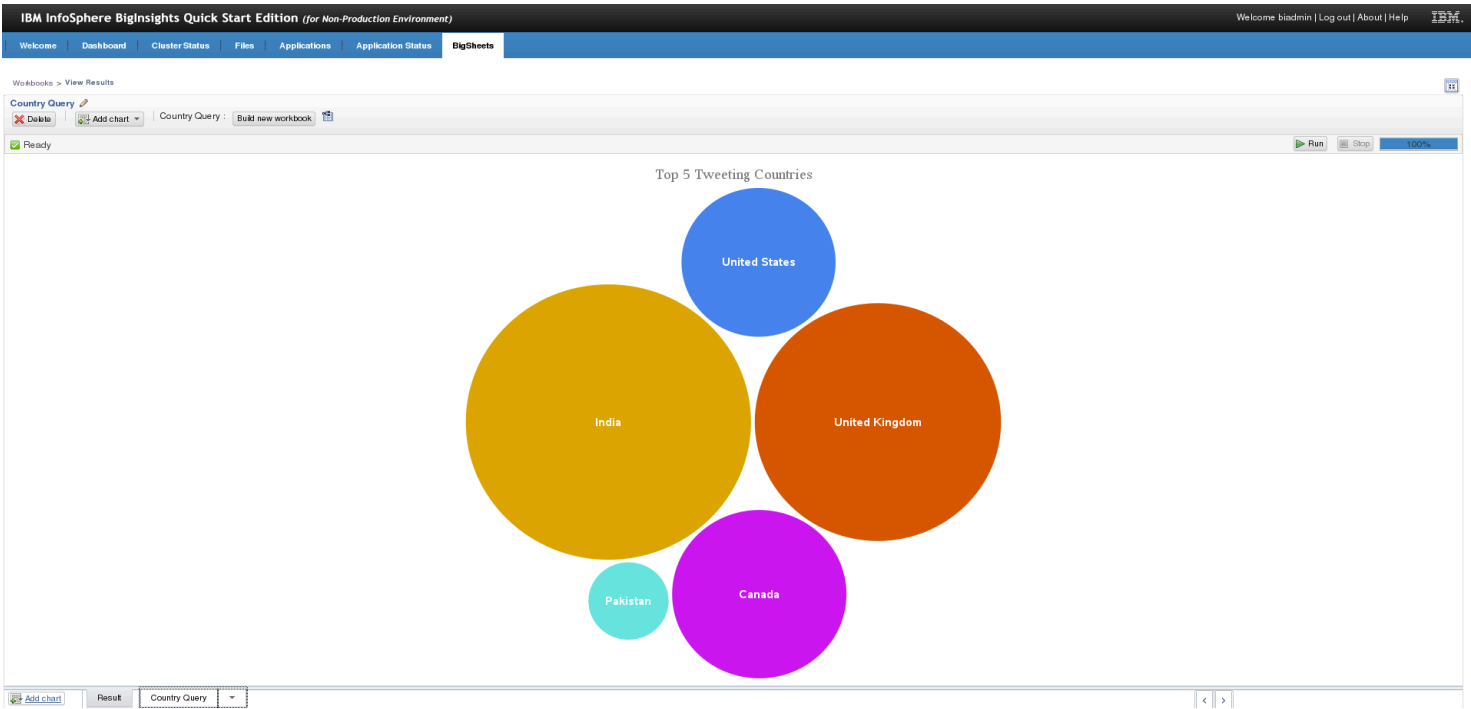
## 5) Output of Team Analysis Query: (Bigsheets Result)



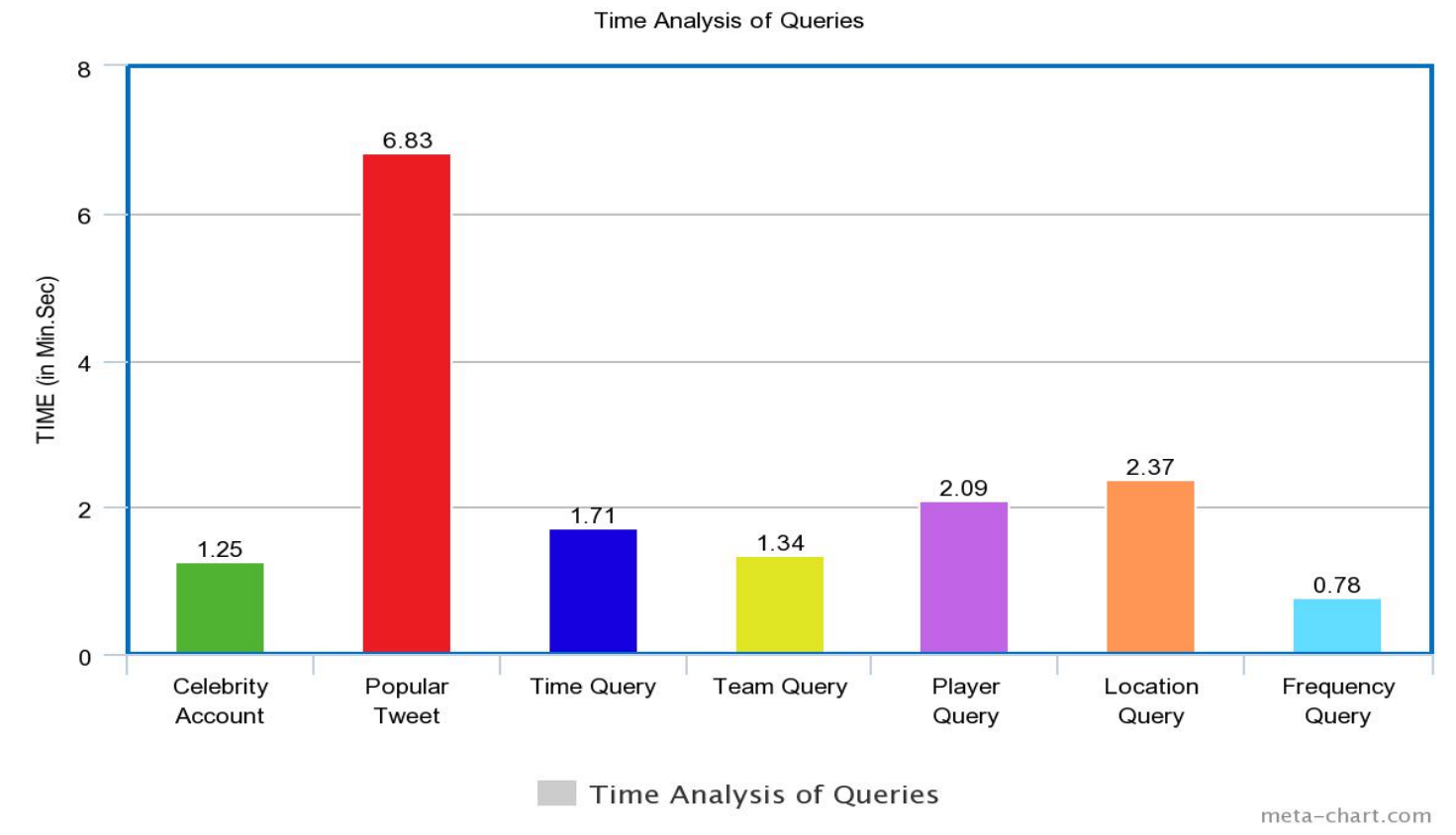
## 6) Output of Player Analysis Query: (Bigsheets Result)



## 7) Output of Location Analysis Query: (Bigsheets Result)



## Time Analysis of various queries:



## **Source Code Link:**

### **1)Main Code:**

<https://drive.google.com/file/d/0B-n05MIQozRIZnVNeU9YVHo1ems/view?usp=sharing>

### **2)Code for Tweets Downloading**

<https://drive.google.com/file/d/0B-n05MIQozRIaTFyWG82SFpnSlE/view?usp=sharing>

## **References:**

- <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Commands>
- <http://help.eclipse.org/luna/index.jsp?topic=%2Forg.eclipse.birt.doc%2Fbirt%2Fcon-HowToSpecifyConnectionInformationForHiveDataSource.html>
- <http://stackoverflow.com/questions/7677333/how-to-write-subquery-and-use-in-clause-in-hive>
- <http://www-01.ibm.com/software/ebusiness/jstart/bigsheets/>