**COMP-SCI 5540 PB Twitter tweets Analysis using Apache Spark and Bluemix**

**Sri Harsha Chennavajjala (SC9V9)**

**Teja Garidepally (TGWW4)**

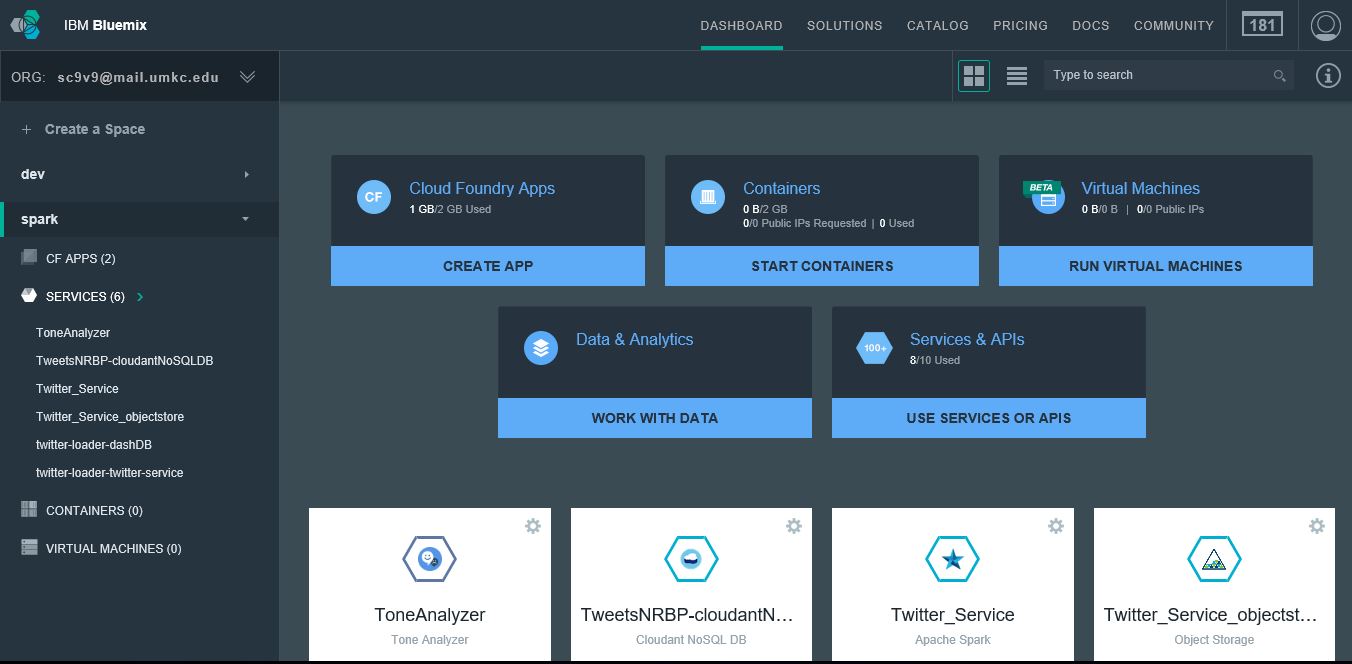
**Raj Kiran Reddy Munnangi (RMYB9)**

Below are the screenshots of our Bluemix environment:

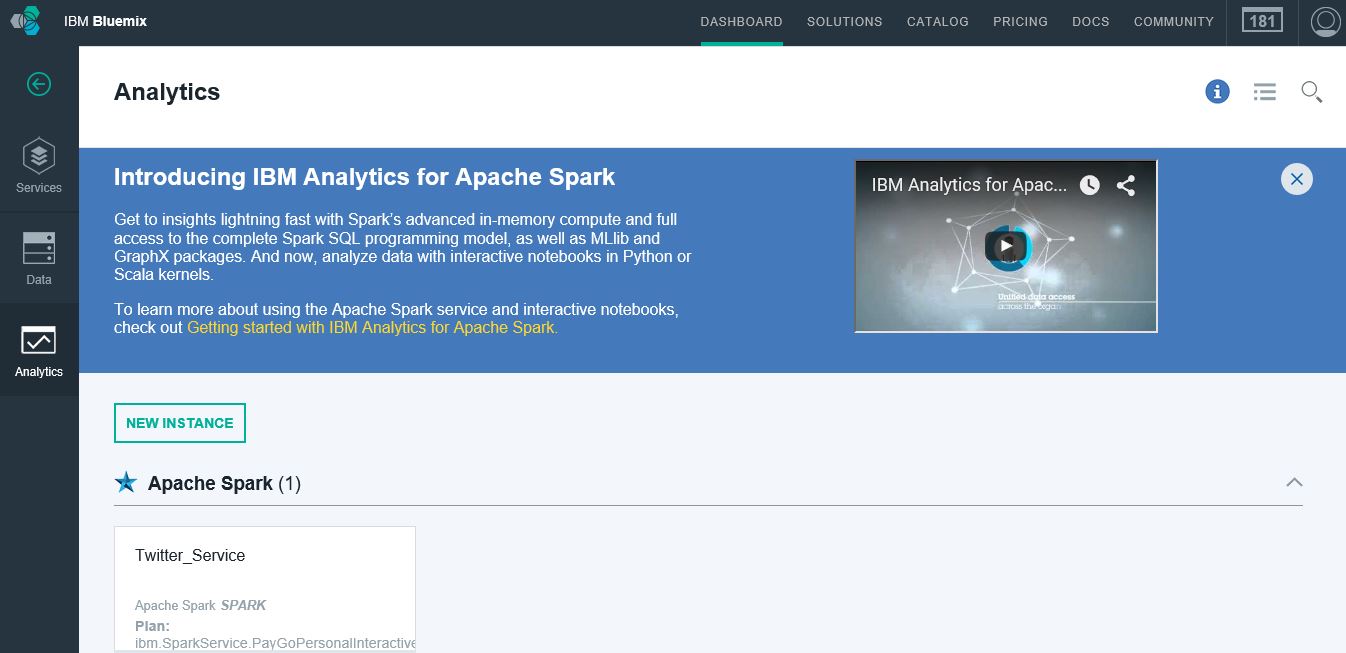
Along with the document we are adding IPython notebooks for your reference.

**Tweets collection application URL:** <http://TweetsNRBP.mybluemix.net>

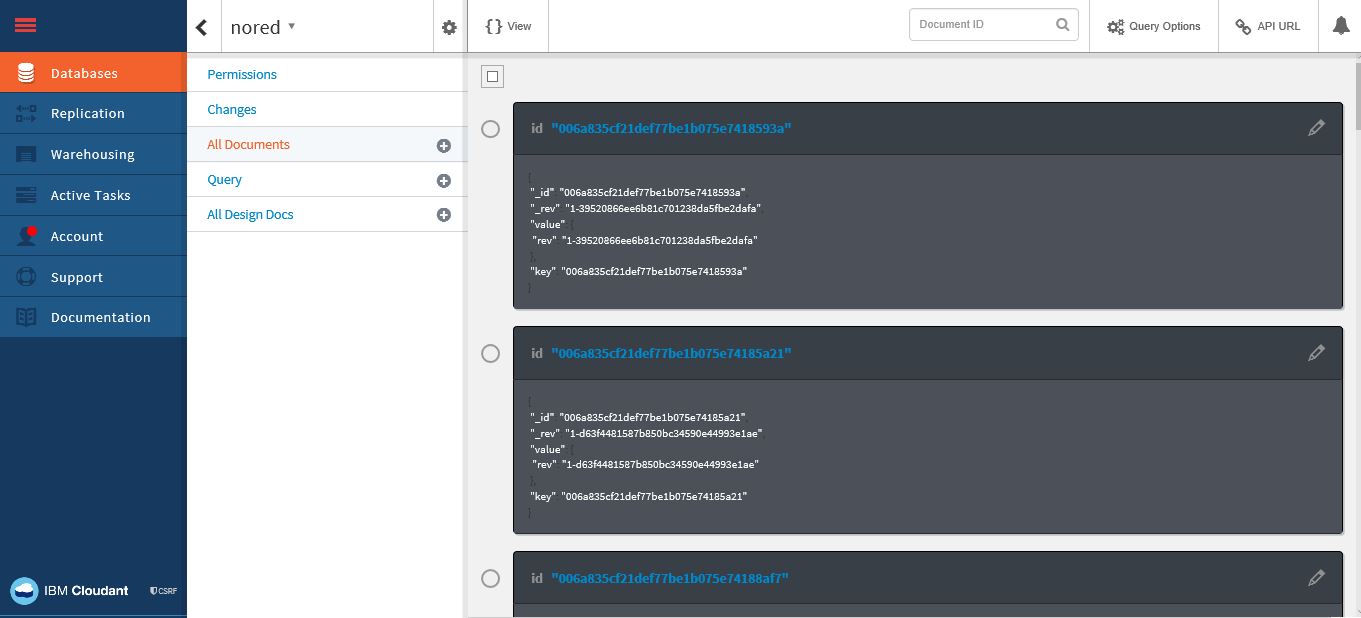
**Dashboard of Bluemix account, which consists of services we have created:**



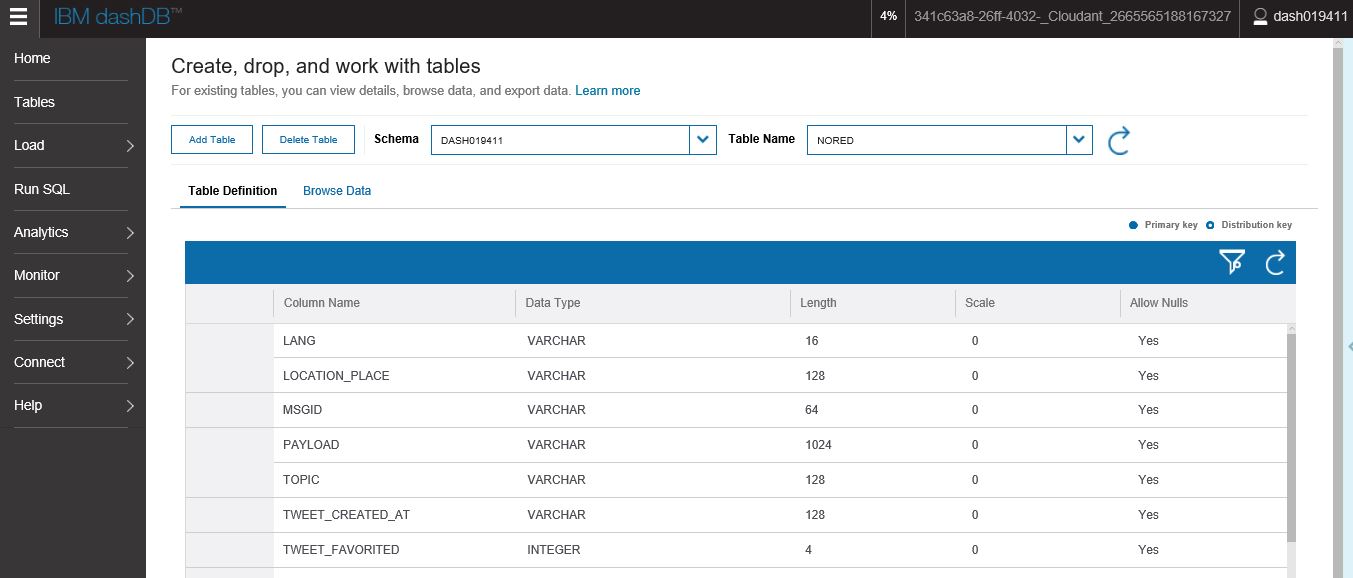
**Apache Spark service for Analysis:**



**Cloudant for storing tweets in JSON format:**

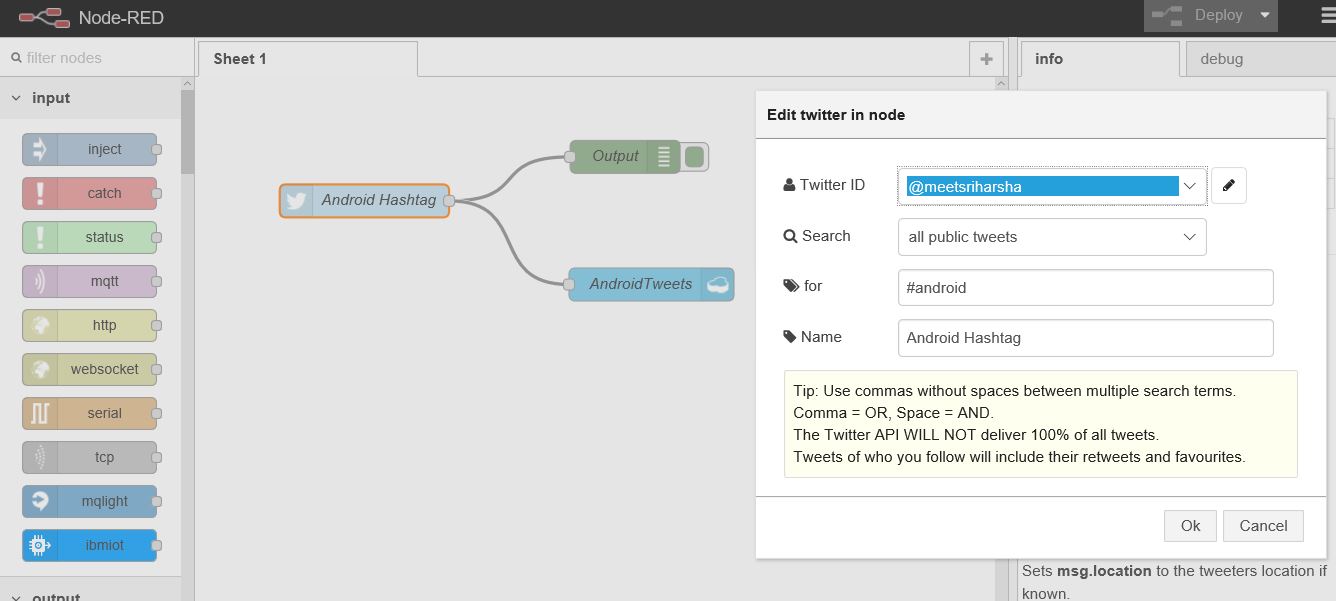


**DashDB to store tweets in tabular format:**

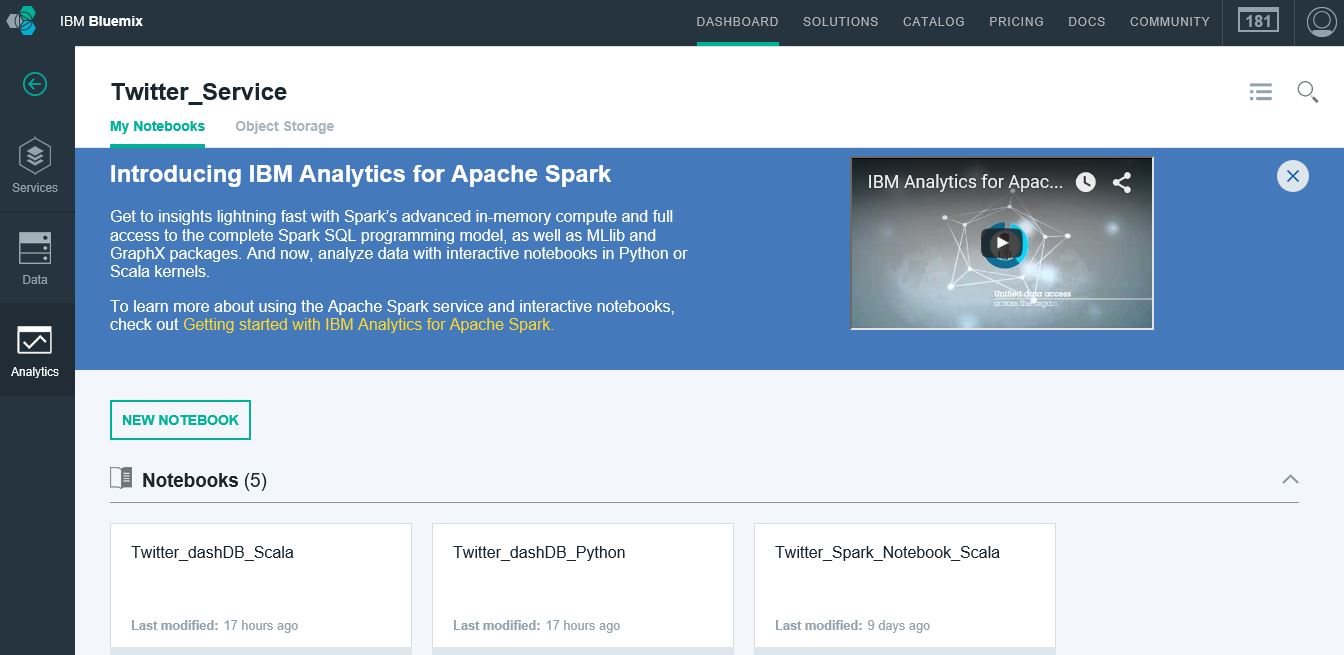


**Tweets collection app based on Node.js:**

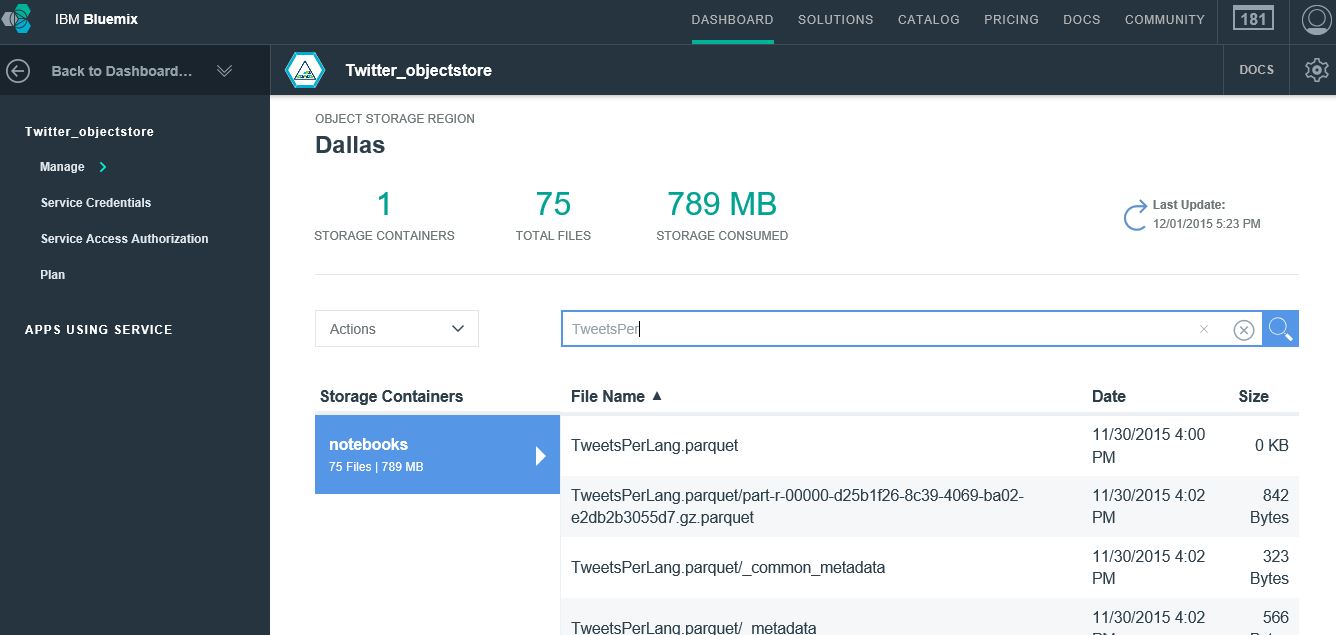
**Application Link:** <http://TweetsNRBP.mybluemix.net>



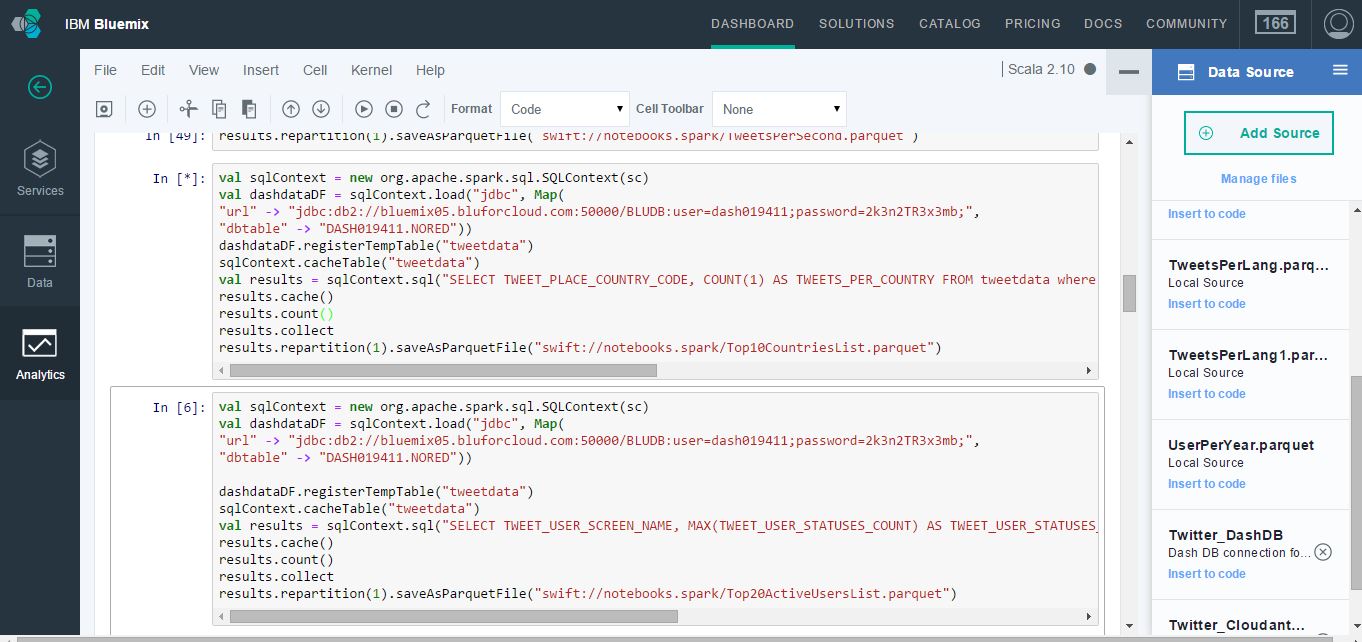
**Notebooks created for tweets analysis and visualization:**



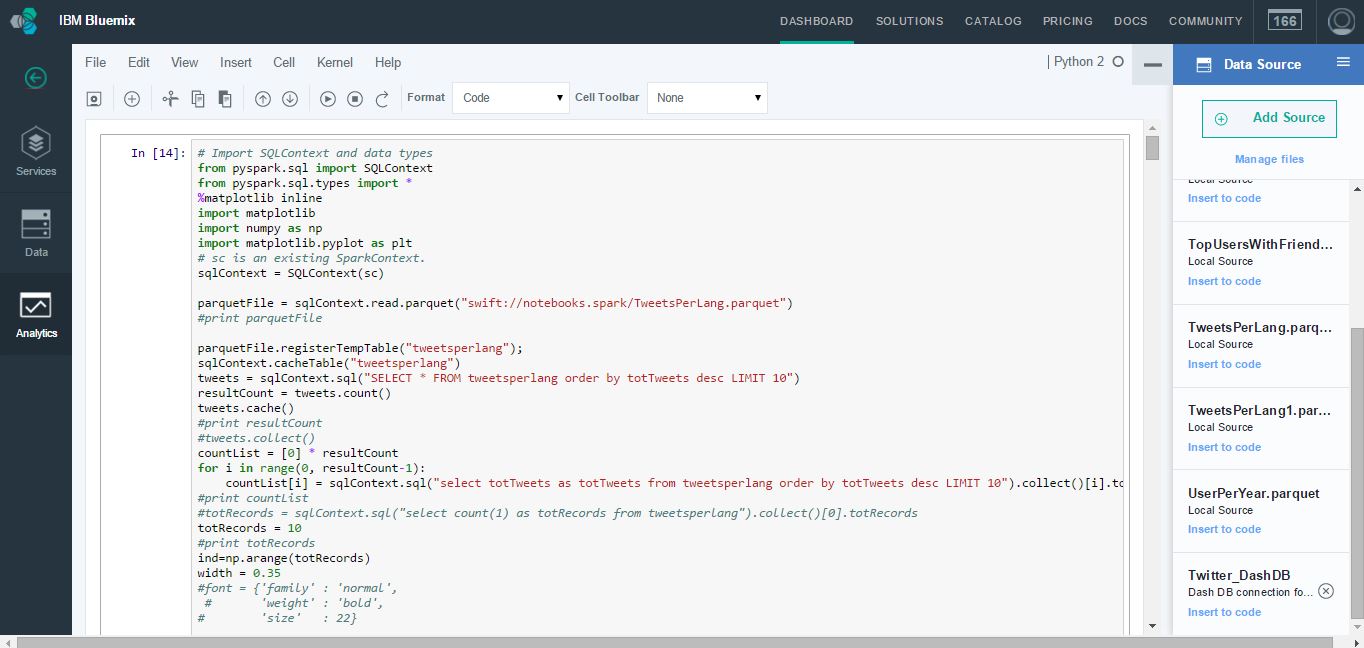
**Object storage for storing output generated from scala analytical queries:**



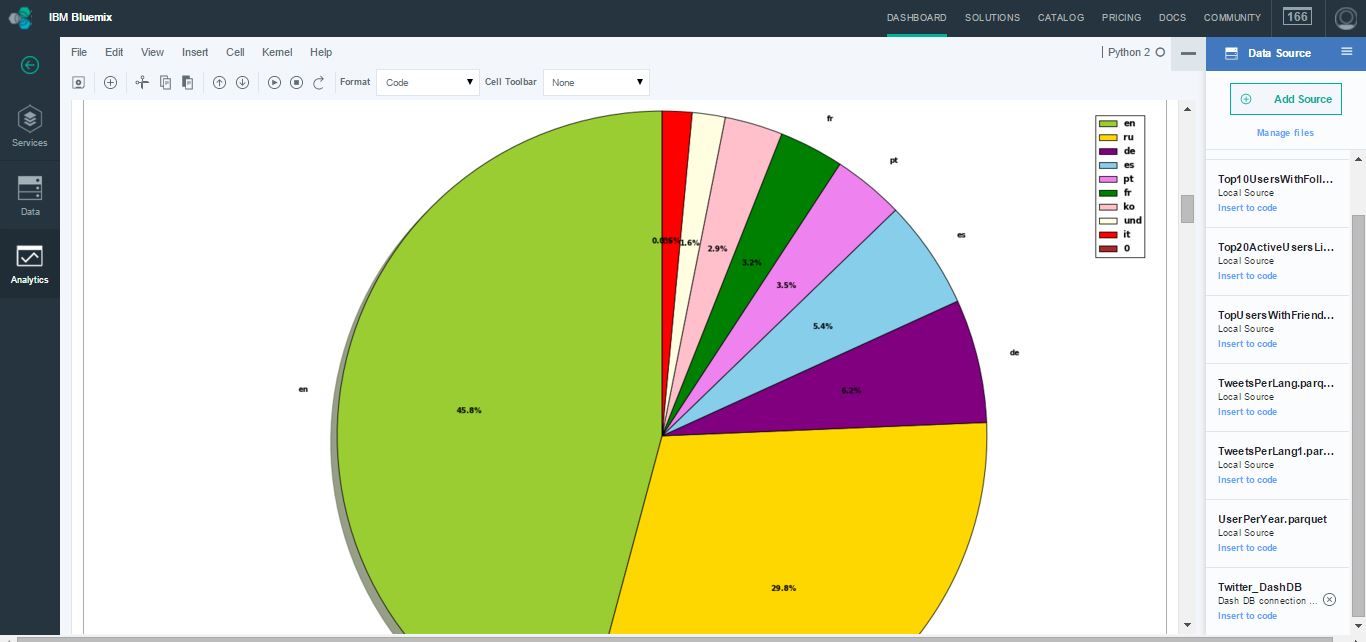
**Sample scala notebook screenshot of analytical queries:**



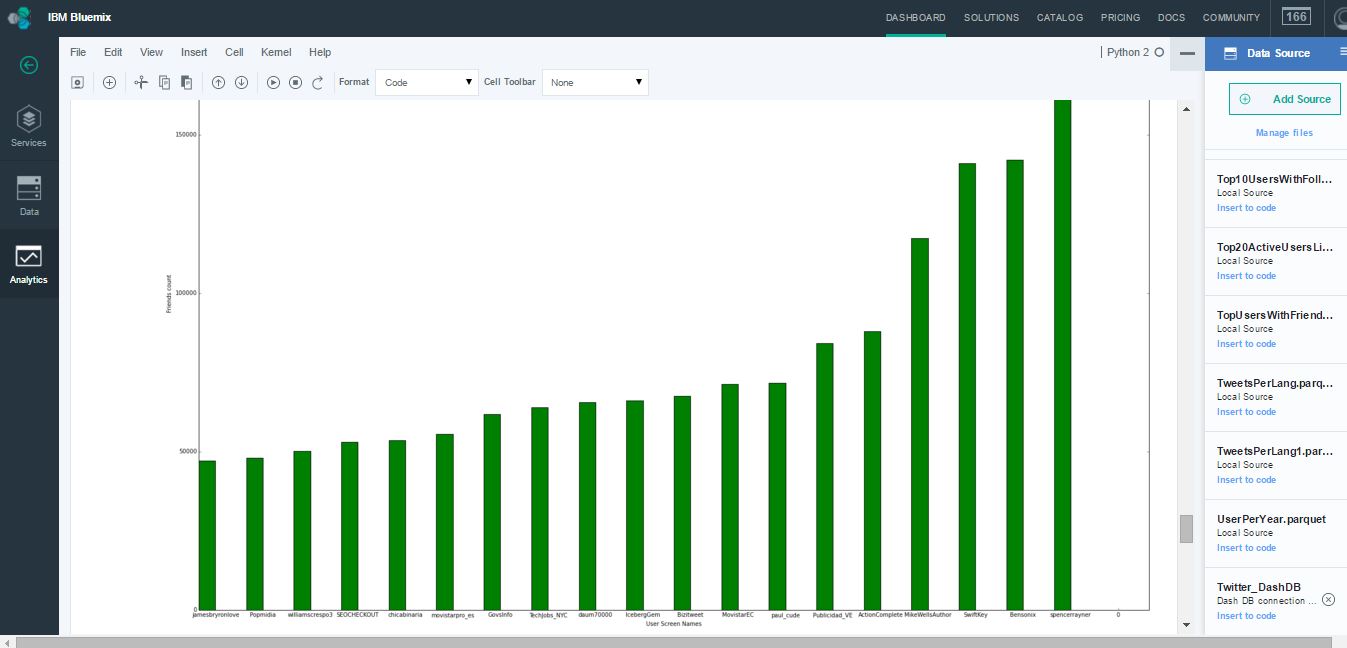
**Python code in IPhython notebook for visualization:**



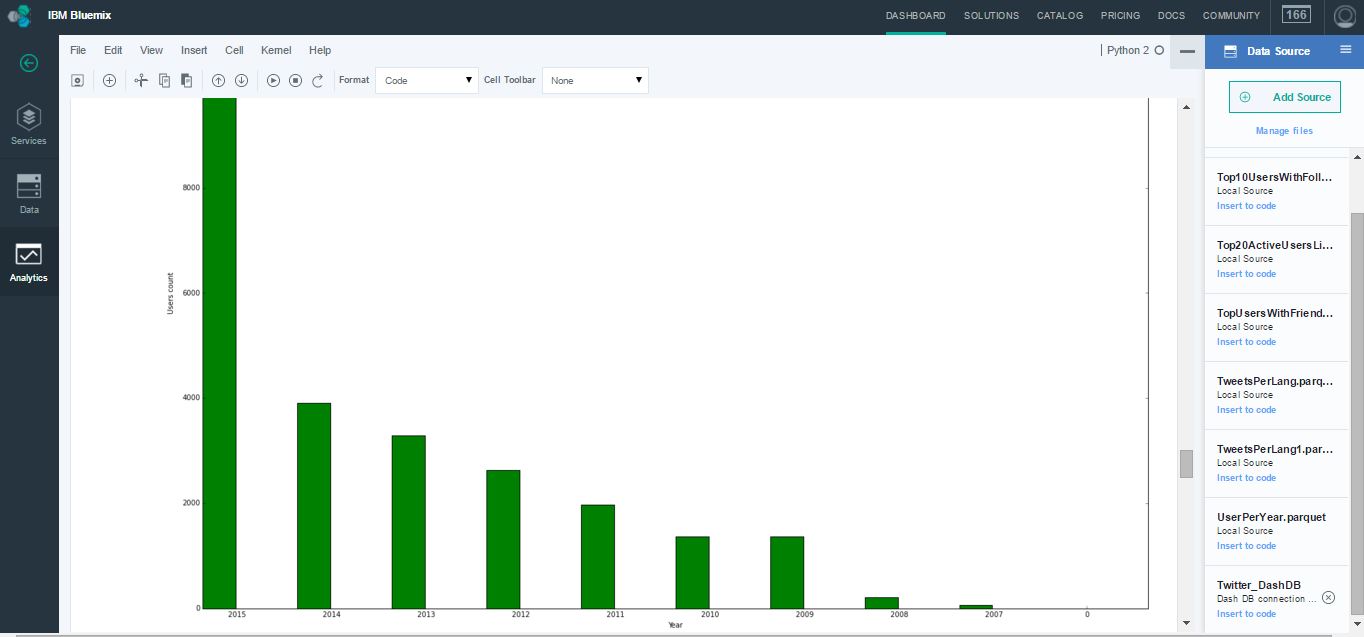
**Visualization of tweets per language in IPython notebook:**



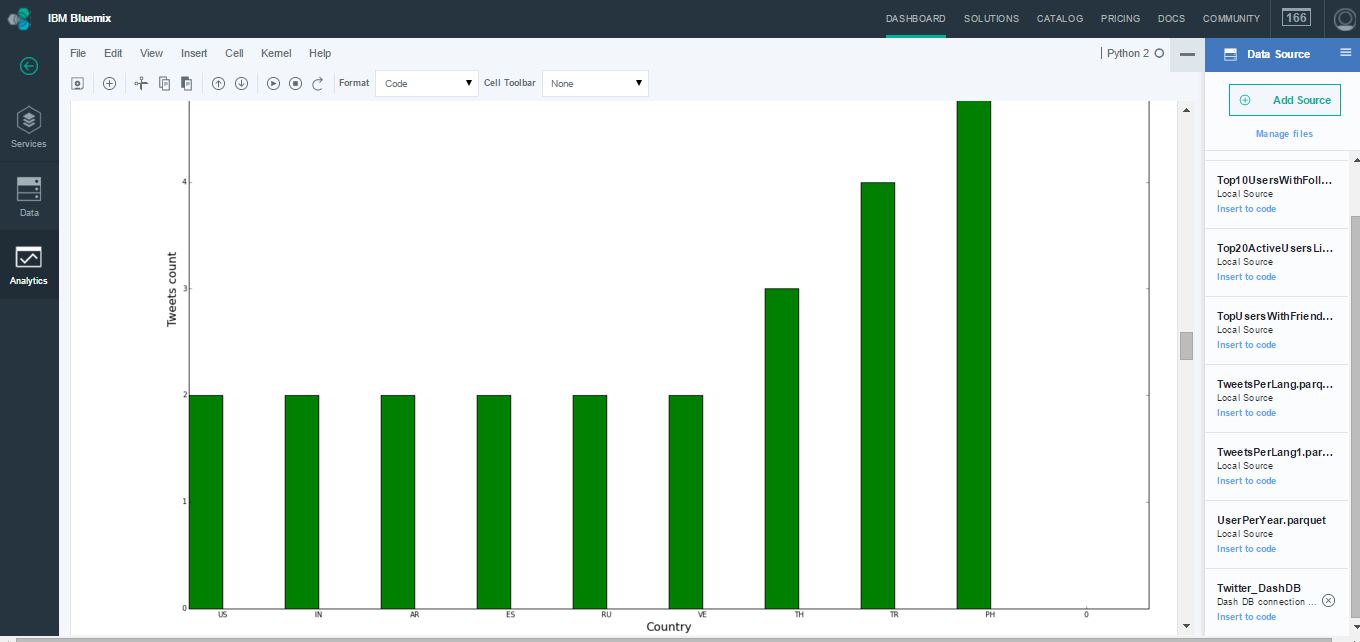
**Top 20 users with highest number of friends:**



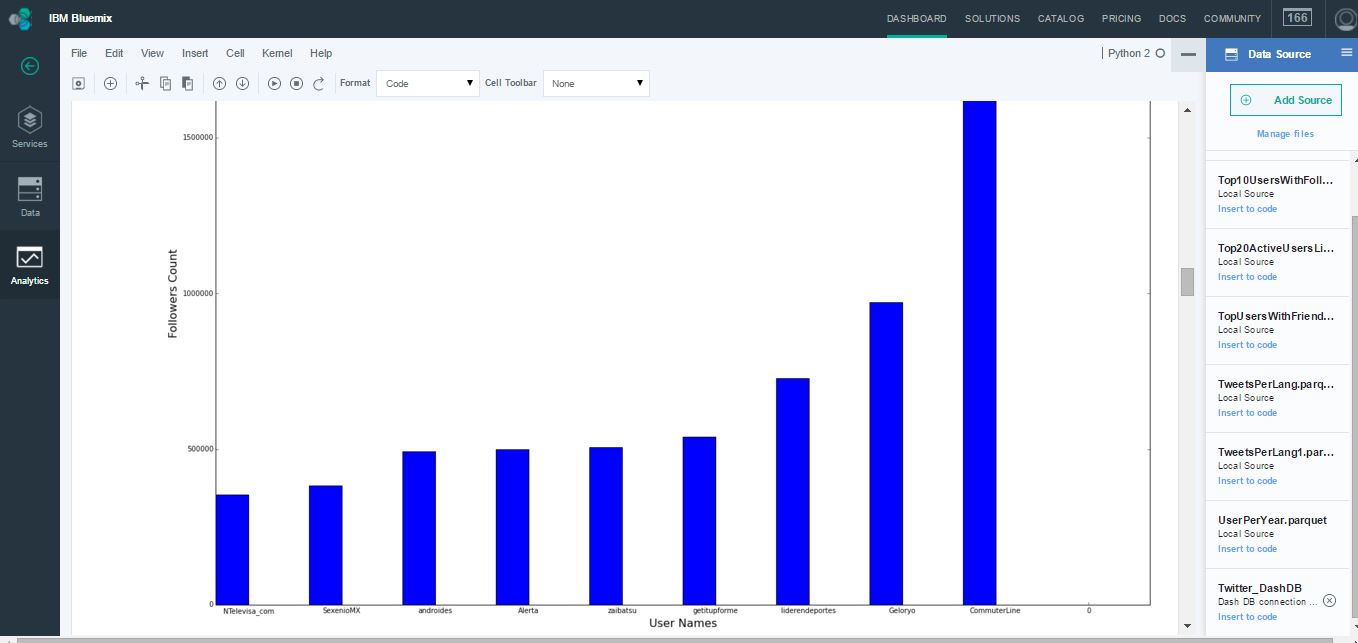
**Number of Users created per year:**



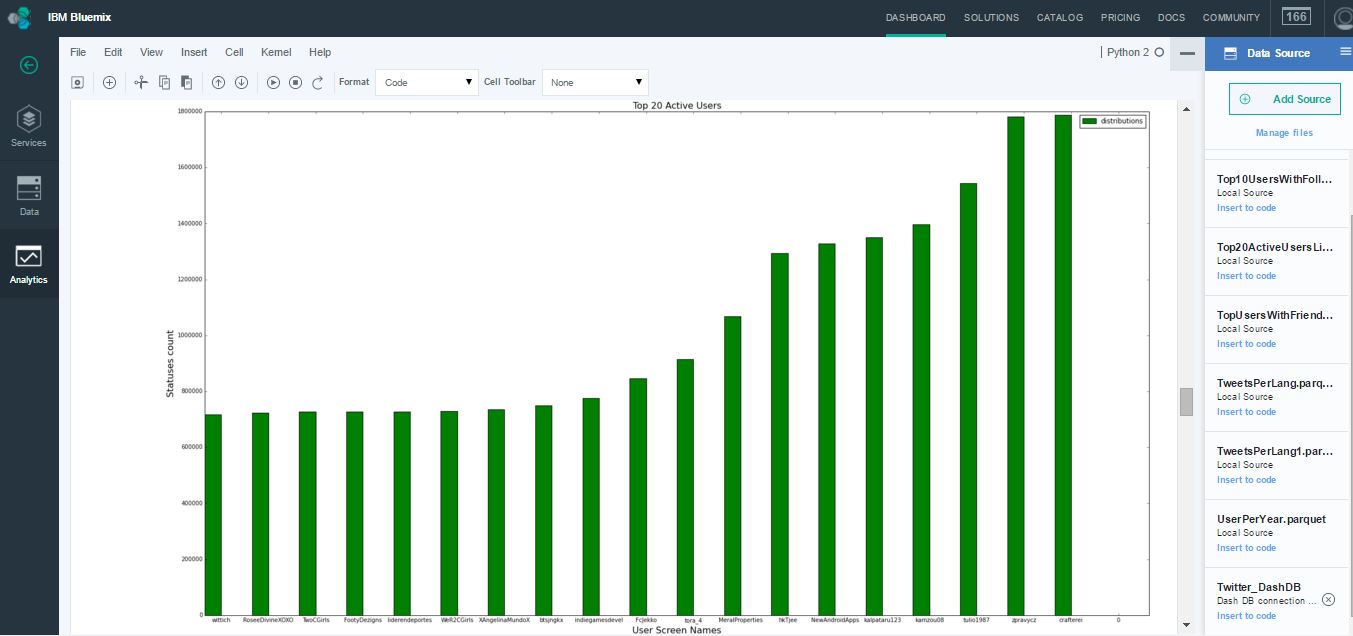
**Number of tweets per country:**



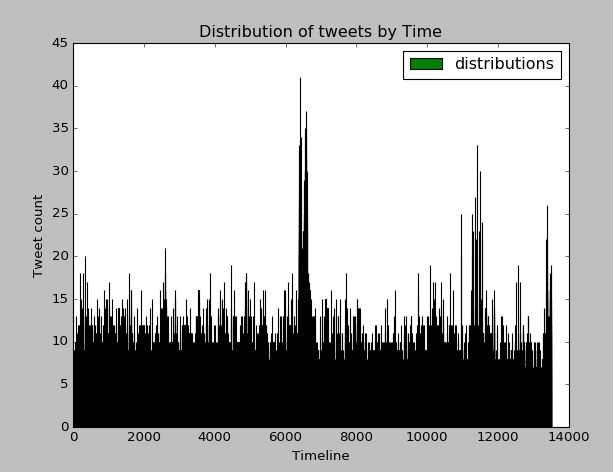
**Top 10 users with highest followers count in dataset:**



**Top 20 active users in dataset:**



**Tweets generated per second:**



**Queries:**

**Most Active users:**

####################################################

# Top 20 Active Users ###############################

# Bar Chart ############# ##########################

####################################################

#Scala Notebook Logic

####################################################

val sqlContext **=** new org**.**apache**.**spark**.**sql**.**SQLContext**(**sc**)**

val dashdataDF **=** sqlContext**.**load**(**"jdbc"**,** Map**(**

"url" **->** "jdbc:db2://bluemix05.bluforcloud.com:50000/BLUDB:user=dash019411;password=2k3n2TR3x3mb;"**,**

"dbtable" **->** "DASH019411.NORED"**))**

dashdataDF**.**registerTempTable**(**"tweetdata"**)**

sqlContext**.**cacheTable**(**"tweetdata"**)**

val results **=** sqlContext**.**sql**(**"SELECT TWEET\_USER\_SCREEN\_NAME, MAX(TWEET\_USER\_STATUSES\_COUNT) AS TWEET\_USER\_STATUSES\_COUNT FROM tweetdata group by TWEET\_USER\_SCREEN\_NAME order by TWEET\_USER\_STATUSES\_COUNT DESC LIMIT 20"**)**

results**.**cache**()**

results**.**count**()**

results**.**collect

results**.**repartition**(**1**).**saveAsParquetFile**(**"swift://notebooks.spark/Top20ActiveUsersList.parquet"**)**

####################################################

#Python Notebook Logic

####################################################

# Import SQLContext and data types

**from** pyspark**.**sql **import** SQLContext

**from** pyspark**.**sql**.**types **import** **\***

**%**matplotlib inline

**import** matplotlib

**import** numpy **as** np

**import** matplotlib**.**pyplot **as** plt

# sc is an existing SparkContext.

sqlContext **=** SQLContext**(**sc**)**

parquetFile **=** sqlContext**.**read**.**parquet**(**"swift://notebooks.spark/Top20ActiveUsersList.parquet"**)**

#print parquetFile

parquetFile**.**registerTempTable**(**"UserStatus"**);**

sqlContext**.**cacheTable**(**"UserStatus"**)**

tweets **=** sqlContext**.**sql**(**"SELECT \* FROM UserStatus order by TWEET\_USER\_STATUSES\_COUNT desc"**)**

resultCount **=** tweets**.**count**()**

tweets**.**cache**()**

**print** resultCount

#tweets.collect()

SCountList **=** **[**0**]** **\*** resultCount

**for** i **in** range**(**0**,** resultCount**-**1**):**

SCountList**[**i**]** **=** sqlContext**.**sql**(**"select TWEET\_USER\_STATUSES\_COUNT as statusCount from UserStatus order by TWEET\_USER\_STATUSES\_COUNT"**).**collect**()[**i**].**statusCount

**print** TCountList

UserList **=** **[**0**]** **\*** resultCount

**for** i **in** range**(**0**,** resultCount**-**1**):**

UserList**[**i**]** **=** sqlContext**.**sql**(**"select TWEET\_USER\_SCREEN\_NAME as userName from UserStatus order by TWEET\_USER\_STATUSES\_COUNT"**).**collect**()[**i**].**userName

#totRecords = sqlContext.sql("select count(1) as totRecords from tweetsperlang").collect()[0].totRecords

totRecords **=** 20

**print** totRecords

ind**=**np**.**arange**(**totRecords**)**

width **=** 0.35

fig **=** plt**.**figure**()**

plt**.**bar**(**ind**,** SCountList**,** width**,** color**=**'g'**,** label **=** "distributions"**)**

params **=** plt**.**gcf**()**

plSize **=** params**.**get\_size\_inches**()**

params**.**set\_size\_inches**(** **(**plSize**[**0**]\***5**,** plSize**[**1**]\***5**)** **)**

plt**.**ylabel**(**'Statuses count'**)**

plt**.**xlabel**(**'User Screen Names'**)**

plt**.**title**(**'Top 20 Active Users'**)**

plt**.**xticks**(**ind**+**width**,** UserList**)**

plt**.**legend**()**

plt**.**show**()**

**Users created per year:**

####################################################

# Top 20 Active Users ###############################

# Bar Chart ############# ##########################

####################################################

#Scala Notebook Logic

####################################################

val sqlContext **=** new org**.**apache**.**spark**.**sql**.**SQLContext**(**sc**)**

val dashdataDF **=** sqlContext**.**load**(**"jdbc"**,** Map**(**

"url" **->** "jdbc:db2://bluemix05.bluforcloud.com:50000/BLUDB:user=dash019411;password=2k3n2TR3x3mb;"**,**

"dbtable" **->** "DASH019411.NORED"**))**

dashdataDF**.**registerTempTable**(**"tweetdata"**)**

sqlContext**.**cacheTable**(**"tweetdata"**)**

val results **=** sqlContext**.**sql**(**"SELECT t1.YR as YR, count(1) as CNT FROM (select DISTINCT TWEET\_USER\_ID, substring(TWEET\_USER\_CREATED\_AT,26) AS YR FROM tweetdata where TWEET\_USER\_CREATED\_AT IS NOT NULL) t1 group by YR order by YR"**)**

results**.**cache**()**

results**.**count**()**

results**.**collect

results**.**repartition**(**1**).**saveAsParquetFile**(**"swift://notebooks.spark/UsersPerYear.parquet"**)**

####################################################

#Python Notebook Logic

####################################################

# Import SQLContext and data types

**from** pyspark**.**sql **import** SQLContext

**from** pyspark**.**sql**.**types **import** **\***

**%**matplotlib inline

**import** matplotlib

**import** numpy **as** np

**import** matplotlib**.**pyplot **as** plt

# sc is an existing SparkContext.

sqlContext **=** SQLContext**(**sc**)**

parquetFile **=** sqlContext**.**read**.**parquet**(**"swift://notebooks.spark/UserYear.parquet"**)**

#print parquetFile

parquetFile**.**registerTempTable**(**"UsersPerYear"**);**

sqlContext**.**cacheTable**(**"UsersPerYear"**)**

tweets **=** sqlContext**.**sql**(**"SELECT \* FROM UsersPerYear order by CNT desc"**)**

resultCount **=** tweets**.**count**()**

tweets**.**cache**()**

**print** resultCount

#tweets.collect()

userCountList **=** **[**0**]** **\*** resultCount

**for** i **in** range**(**0**,** resultCount**-**1**):**

userCountList**[**i**]** **=** sqlContext**.**sql**(**"select CNT as userCount from UsersPerYear order by CNT desc"**).**collect**()[**i**].**userCount

**print** userCountList

yearList **=** **[**0**]** **\*** resultCount

**for** i **in** range**(**0**,** resultCount**-**1**):**

yearList**[**i**]** **=** sqlContext**.**sql**(**"select YR as yer from UsersPerYear order by CNT desc"**).**collect**()[**i**].**yer

#totRecords = sqlContext.sql("select count(1) as totRecords from tweetsperlang").collect()[0].totRecords

ind**=**np**.**arange**(**len**(**yearList**))**

width **=** 0.35

fig **=** plt**.**figure**()**

plt**.**bar**(**ind**,** userCountList**,** width**,** color**=**'g'**,** label **=** "distributions"**)**

params **=** plt**.**gcf**()**

plSize **=** params**.**get\_size\_inches**()**

params**.**set\_size\_inches**(** **(**plSize**[**0**]\***5**,** plSize**[**1**]\***5**)** **)**

plt**.**ylabel**(**'Users count'**)**

plt**.**xlabel**(**'Year'**)**

plt**.**title**(**'Users created per each year'**)**

plt**.**xticks**(**ind**+**width**,** yearList**)**

plt**.**legend**()**

plt**.**show**()**

**Tweets per second:**

####################################################

# Tweets per Second ###############################

# Bar Chart ############# ##########################

####################################################

#Scala Notebook Logic

####################################################

**import** org**.**apache**.**spark**.**SparkContext**.**\_

**import** org**.**apache**.**spark**.{**SparkConf**,** SparkContext**}**

**import** org**.**apache**.**spark**.**sql**.**SQLContext

**import** org**.**apache**.**spark**.**SparkContext

**import** scala**.**collection**.**immutable**.**ListMap

object TweetCount **{**

**def** main**(**args**:** Array**[**String**])** **{**

val conf **=** new SparkConf**().**setAppName**(**"TweetCount"**)**

conf**.**setMaster**(**"local"**)**

val sc **=** new SparkContext**(**conf**)**

val sqlContext **=** new SQLContext**(**sc**)**

val tweets **=** sc**.**textFile**(**"D:/UMKC/Docs/Subjects/PBDM/PB\_Project/iphone6s.json"**)**

**//.**map**(**gson**.**toJson**(**\_**))**

**//** println**(**tweets**.**first**())**

**//**tweets**.**collect**()**

**//**val userMap **=** tweets**.**map**(**x **=>** **((**extractUserId**(**x**),**extractLang**(**x**)),** 1**))**

val userTweetCnt **=** tweets**.**filter**(**\_**.**nonEmpty**).**map**(**x **=>** **(**extractUserId**(**x**),** 1**))**

val langTweetCnt **=** tweets**.**filter**(**\_**.**nonEmpty**).**map**(**x **=>** **(**extractLang**(**x**),** 1**))**

val tweetCreatedDt **=** tweets**.**filter**(**\_**.**nonEmpty**).**map**(**x **=>** **(**extractTweetDate**(**x**),** 1**))**

val tweetCreatedDtLang **=** tweets**.**filter**(**\_**.**nonEmpty**).**map**(**x **=>** **((**extractTweetDate**(**x**),**extractLang**(**x**)),** 1**))**

val numTweetsPerUser **=** userTweetCnt**.**reduceByKey**((**a**,** b**)** **=>** a **+** b**)**

val numTweetsPerLang **=** langTweetCnt**.**reduceByKey**((**a**,** b**)** **=>** a **+** b**)**

val tweetCreatedDtCnt **=** tweetCreatedDt**.**reduceByKey**((**a**,** b**)** **=>** a **+** b**)**

val tweetCreatedDtLangCnt **=** tweetCreatedDtLang**.**reduceByKey**((**a**,** b**)** **=>** a **+** b**)**

val numTweetsPerUserSorted **=**numTweetsPerUser**.**sortBy**(**\_**.**\_2**)**

val numTweetsPerLangSorted **=**numTweetsPerLang**.**sortBy**(**\_**.**\_2**)**

numTweetsPerUser**.**repartition**(**1**).**saveAsTextFile**(**"D:/UMKC/Docs/Subjects/PBDM/PB\_Project/TweetsPerUser"**)**

numTweetsPerLang**.**repartition**(**1**).**saveAsTextFile**(**"D:/UMKC/Docs/Subjects/PBDM/PB\_Project/TweetsPerLang"**)**

tweetCreatedDtCnt**.**repartition**(**1**).**saveAsTextFile**(**"D:/UMKC/Docs/Subjects/PBDM/PB\_Project/TweetsPerTime"**)**

tweetCreatedDtLangCnt**.**repartition**(**1**).**saveAsTextFile**(**"D:/UMKC/Docs/Subjects/PBDM/PB\_Project/TweetsPerTimeLang"**)**

**//** val tweet **=** sqlContext**.**load**(**"D:/UMKC/Docs/Subjects/PBDM/PB\_Project/iphone6s.json"**,** "json"**)**

**//** tweet**.**printSchema**()**

**//**println**(**tweet**.**select**(**"quoted\_status.user.id\_str"**).**rdd**)**

**//**val userMp **=** tweet**.**map **{** x **=>** **(** **}**

**}**

**def** extractUserId**(**x**:** String**):** String **=** **{**

val userIdPattern **=** "\"user\"\\:\\{\"id\"\\:([^,]+)"**.**r

val userId **=** userIdPattern**.**findFirstMatchIn**(**x**).**map**(**x **=>** x**.**group**(**1**)).**getOrElse**(**"na"**)**

userId

**}**

**def** extractLang**(**x**:** String**):** String **=** **{**

**//**val userIdPattern **=** "\"user\".\*\"id\"\\:([^,]+)"**.**r

**//**val userIdPattern **=** "\"user\"\\:\\{\"id\"\\:([^,]+)"**.**r

val langPattern **=** "\"user\"\\:\\{\"id\".\*\"lang\"\\:([^,]+)"**.**r

val lang **=** langPattern**.**findFirstMatchIn**(**x**).**map**(**x **=>** x**.**group**(**1**)).**getOrElse**(**"na"**)**

lang

**}**

**def** extractFollowerCnt**(**x**:** String**):** String **=** **{**

**//**val userIdPattern **=** "\"user\".\*\"id\"\\:([^,]+)"**.**r

**//**val userIdPattern **=** "\"user\"\\:\\{\"id\"\\:([^,]+)"**.**r

val followerCntPattern **=** "\"user\"\\:\\{\"id\".\*\"followers\_count\"\\:([^,]+)"**.**r

val followerCnt **=** followerCntPattern**.**findFirstMatchIn**(**x**).**map**(**x **=>** x**.**group**(**1**)).**getOrElse**(**"na"**)**

followerCnt

**}**

**def** extractTweetDate**(**x**:** String**):** String **=** **{**

**//**val userIdPattern **=** "\"user\".\*\"id\"\\:([^,]+)"**.**r

**//**val userIdPattern **=** "\"user\"\\:\\{\"id\"\\:([^,]+)"**.**r

val tweetDatePattern **=** "\"created\_at\"\\:([^,]+)"**.**r

val tweetDate **=** tweetDatePattern**.**findFirstMatchIn**(**x**).**map**(**x **=>** x**.**group**(**1**)).**getOrElse**(**"na"**)**

tweetDate

**}**

**}**

####################################################

#Python Notebook Logic

####################################################

**import** matplotlib

**import** numpy **as** np

**import** matplotlib**.**pyplot **as** plt

**import** re

**import** time

**from** dateutil**.**parser **import** parse

dt **=** **[]**

count **=** **[]**

f **=** open**(**'D:/UMKC/Docs/Subjects/PBDM/PB\_Project/TweetsPerTime/part-00000'**,** 'rU'**)** #open the file in read universal mode

**for** line **in** f**:**

cells **=** **(**re**.**sub**(**'[\(\)\"\\n]'**,** ''**,** line**)).**split**(**","**)**

dt**.**append**(**cells**[**0**])**

count**.**append**(**cells**[**1**])**

f**.**close**()**

epochdt **=** **[**0**]** **\*** len**(**dt**)**

**for** i **in** range**(**0**,** len**(**dt**)):**

**try:**

epochdtTmp **=** parse**(**dt**[**i**])**

**except** ValueError**:**

epochdtTmp **=** parse**(**'Sun Nov 23 01:44:10 +0000 2015'**)**

epochdt**[**i**]** **=** time**.**mktime**(**epochdtTmp**.**timetuple**())**

finalList **=** **[**0**]** **\*** len**(**count**)**

**for** i **in** range**(**0**,** len**(**count**)):**

finalList**[**i**]** **=** **[**epochdt**[**i**],** count**[**i**]]**

finalListSorted **=** sorted**(**finalList**,** key**=lambda** x**:** x**[**0**],** reverse**=False)**

#print finalListSorted

countFinal **=** **[**0**]** **\*** len**(**finalListSorted**)**

countFinal **=** **[**x**[**1**]** **for** x **in** finalListSorted**]**

#print countFinal

width **=** 0.5

ind **=** np**.**arange**(**len**(**countFinal**))**

plt**.**bar**(**ind**,** countFinal**,** width**,** color**=**'g'**,** label**=**"distributions"**)**

params **=** plt**.**gcf**()**

plSize **=** params**.**get\_size\_inches**()**

params**.**set\_size\_inches**((**plSize**[**0**]\***5**,** plSize**[**1**]\***5**))**

plt**.**ylabel**(**'Tweet count'**)**

plt**.**xlabel**(**'Timeline'**)**

plt**.**title**(**'Distribution of tweets by Time'**)**

#plt.xticks(ind+width, finalListSorted[:0])

plt**.**legend**(**loc**=**'best'**)**

plt**.**show**()**

**Tweets per Language:**

####################################################

#Tweets per Language Bar Chart and Pie Chart###

####################################################

#Scala Notebook Logic

####################################################

val sqlContext **=** new org**.**apache**.**spark**.**sql**.**SQLContext**(**sc**)**

val dashdataDF **=** sqlContext**.**load**(**"jdbc"**,** Map**(**

"url" **->** "jdbc:db2://bluemix05.bluforcloud.com:50000/BLUDB:user=dash019411;password=2k3n2TR3x3mb;"**,**

"dbtable" **->** "DASH019411.NORED"**))**

dashdataDF**.**registerTempTable**(**"tweetdata"**)**

sqlContext**.**cacheTable**(**"tweetdata"**)**

val results **=** sqlContext**.**sql**(**"SELECT TWEET\_LANG, count(1) as totTweets from tweetdata group by TWEET\_LANG order by totTweets"**)**

results**.**cache**()**

results**.**count**()**

results**.**collect

results**.**repartition**(**1**).**saveAsParquetFile**(**"swift://notebooks.spark/TweetsPerLang.parquet"**)**

####################################################

#Python Notebook Logic

####################################################

# Import SQLContext and data types

**from** pyspark**.**sql **import** SQLContext

**from** pyspark**.**sql**.**types **import** **\***

**%**matplotlib inline

**import** matplotlib

**import** numpy **as** np

**import** matplotlib**.**pyplot **as** plt

# sc is an existing SparkContext.

sqlContext **=** SQLContext**(**sc**)**

parquetFile **=** sqlContext**.**read**.**parquet**(**"swift://notebooks.spark/TweetsPerLang.parquet"**)**

#print parquetFile

parquetFile**.**registerTempTable**(**"tweetsperlang"**);**

sqlContext**.**cacheTable**(**"tweetsperlang"**)**

tweets **=** sqlContext**.**sql**(**"SELECT \* FROM tweetsperlang order by totTweets desc LIMIT 10"**)**

resultCount **=** tweets**.**count**()**

tweets**.**cache**()**

**print** resultCount

#tweets.collect()

countList **=** **[**0**]** **\*** resultCount

**for** i **in** range**(**0**,** resultCount**-**1**):**

countList**[**i**]** **=** sqlContext**.**sql**(**"select totTweets as totTweets from tweetsperlang order by totTweets desc LIMIT 10"**).**collect**()[**i**].**totTweets

**print** countList

langList **=** **[**0**]** **\*** resultCount

**for** i **in** range**(**0**,** resultCount**-**1**):**

langList**[**i**]** **=** sqlContext**.**sql**(**"select TWEET\_LANG as tweetLang from tweetsperlang order by totTweets desc LIMIT 10"**).**collect**()[**i**].**tweetLang

#totRecords = sqlContext.sql("select count(1) as totRecords from tweetsperlang").collect()[0].totRecords

totRecords **=** 10

**print** totRecords

ind**=**np**.**arange**(**totRecords**)**

width **=** 0.35

fig **=** plt**.**figure**()**

plt**.**bar**(**ind**,** countList**,** width**,** color**=**'g'**,** label **=** "distributions"**)**

params **=** plt**.**gcf**()**

plSize **=** params**.**get\_size\_inches**()**

params**.**set\_size\_inches**(** **(**plSize**[**0**]\***5**,** plSize**[**1**]\***5**)** **)**

plt**.**ylabel**(**'Tweet count'**)**

plt**.**xlabel**(**'Language'**)**

plt**.**title**(**'Distribution of tweets by Language'**)**

plt**.**xticks**(**ind**+**width**,** langList**)**

plt**.**legend**()**

plt**.**show**()**

#Plotting Pie chart for tweets per language

colorArray **=** **[**"yellowgreen"**,** "gold"**,** "purple"**,** "skyblue"**,** "violet"**,** "green"**,** "pink"**,** "lightyellow"**,** "red"**,**"brown"**]**

f **=** plt**.**figure**()**

params **=** plt**.**gcf**()**

plSize **=** params**.**get\_size\_inches**()**

params**.**set\_size\_inches**((**plSize**[**0**]\***5**,** plSize**[**1**]\***5**))**

plt**.**pie**(**countList**,** labels**=**langList**,** colors**=**colorArray**,** autopct**=**'%1.1f%%'**,** shadow**=True,** startangle**=**90**)**

plt**.**legend**(**topLang**,** loc**=**'best'**,** fontsize**=**12**)**

plt**.**axis**(**'equal'**)**

plt**.**show**()**

**Tweets per Country:**

####################################################

# Tweets per Country ###############################

# Bar Chart ############# ##########################

####################################################

#Scala Notebook Logic

####################################################

val sqlContext **=** new org**.**apache**.**spark**.**sql**.**SQLContext**(**sc**)**

val dashdataDF **=** sqlContext**.**load**(**"jdbc"**,** Map**(**

"url" **->** "jdbc:db2://bluemix05.bluforcloud.com:50000/BLUDB:user=dash019411;password=2k3n2TR3x3mb;"**,**

"dbtable" **->** "DASH019411.NORED"**))**

dashdataDF**.**registerTempTable**(**"tweetdata"**)**

sqlContext**.**cacheTable**(**"tweetdata"**)**

val results **=** sqlContext**.**sql**(**"SELECT TWEET\_PLACE\_COUNTRY\_CODE, COUNT(1) AS TWEETS\_PER\_COUNTRY FROM tweetdata group by TWEET\_PLACE\_COUNTRY\_CODE order by TWEETS\_PER\_COUNTRY DESC LIMIT 10"**)**

results**.**cache**()**

results**.**count**()**

results**.**collect

results**.**repartition**(**1**).**saveAsParquetFile**(**"swift://notebooks.spark/Top10CountriesList.parquet"**)**

####################################################

#Python Notebook Logic

####################################################

# Import SQLContext and data types

**from** pyspark**.**sql **import** SQLContext

**from** pyspark**.**sql**.**types **import** **\***

**%**matplotlib inline

**import** matplotlib

**import** numpy **as** np

**import** matplotlib**.**pyplot **as** plt

# sc is an existing SparkContext.

sqlContext **=** SQLContext**(**sc**)**

parquetFile **=** sqlContext**.**read**.**parquet**(**"swift://notebooks.spark/Top10CountriesList.parquet"**)**

#print parquetFile

parquetFile**.**registerTempTable**(**"TweetsPerCountry"**);**

sqlContext**.**cacheTable**(**"TweetsPerCountry"**)**

tweets **=** sqlContext**.**sql**(**"SELECT \* FROM TweetsPerCountry order by TWEETS\_PER\_COUNTRY desc"**)**

resultCount **=** tweets**.**count**()**

tweets**.**cache**()**

**print** resultCount

#tweets.collect()

TCountList **=** **[**0**]** **\*** resultCount

**for** i **in** range**(**0**,** resultCount**-**1**):**

TCountList**[**i**]** **=** sqlContext**.**sql**(**"select TWEETS\_PER\_COUNTRY as tweetsCount from TweetsPerCountry order by TWEETS\_PER\_COUNTRY"**).**collect**()[**i**].**tweetsCount

**print** TCountList

CountryList **=** **[**0**]** **\*** resultCount

**for** i **in** range**(**0**,** resultCount**-**1**):**

CountryList**[**i**]** **=** sqlContext**.**sql**(**"select TWEET\_PLACE\_COUNTRY\_CODE as countryName from TweetsPerCountry"**).**collect**()[**i**].**countryName

#totRecords = sqlContext.sql("select count(1) as totRecords from tweetsperlang").collect()[0].totRecords

totRecords **=** 10

**print** totRecords

ind**=**np**.**arange**(**totRecords**)**

width **=** 0.35

fig **=** plt**.**figure**()**

plt**.**bar**(**ind**,** TCountList**,** width**,** color**=**'g'**,** label **=** "distributions"**)**

params **=** plt**.**gcf**()**

plSize **=** params**.**get\_size\_inches**()**

params**.**set\_size\_inches**(** **(**plSize**[**0**]\***5**,** plSize**[**1**]\***5**)** **)**

plt**.**ylabel**(**'Tweets count'**)**

plt**.**xlabel**(**'Country'**)**

plt**.**title**(**'Distribution of tweets by Country'**)**

plt**.**xticks**(**ind**+**width**,** CountryList**)**

plt**.**legend**()**

plt**.**show**()**

**Top users with highest number of friends:**

####################################################

# Top 20 Active Users ###############################

# Bar Chart ############# ##########################

####################################################

#Scala Notebook Logic

####################################################

val sqlContext **=** new org**.**apache**.**spark**.**sql**.**SQLContext**(**sc**)**

val dashdataDF **=** sqlContext**.**load**(**"jdbc"**,** Map**(**

"url" **->** "jdbc:db2://bluemix05.bluforcloud.com:50000/BLUDB:user=dash019411;password=2k3n2TR3x3mb;"**,**

"dbtable" **->** "DASH019411.NORED"**))**

dashdataDF**.**registerTempTable**(**"tweetdata"**)**

sqlContext**.**cacheTable**(**"tweetdata"**)**

val results **=** sqlContext**.**sql**(**"SELECT MAX(TWEET\_USER\_FRIENDS\_COUNT) AS FRIENDS\_COUNT, TWEET\_USER\_SCREEN\_NAME FROM tweetdata GROUP BY TWEET\_USER\_SCREEN\_NAME ORDER BY FRIENDS\_COUNT DESC LIMIT 20"**);**

")

results**.**cache**()**

results**.**count**()**

results**.**collect

results**.**repartition**(**1**).**saveAsParquetFile**(**"swift://notebooks.spark/Top20ActiveUsersList.parquet"**)**

####################################################

#Python Notebook Logic

####################################################

# Import SQLContext and data types

**from** pyspark**.**sql **import** SQLContext

**from** pyspark**.**sql**.**types **import** **\***

**%**matplotlib inline

**import** matplotlib

**import** numpy **as** np

**import** matplotlib**.**pyplot **as** plt

# sc is an existing SparkContext.

sqlContext **=** SQLContext**(**sc**)**

parquetFile **=** sqlContext**.**read**.**parquet**(**"swift://notebooks.spark/TopUsersWithFriends.parquet"**)**

#print parquetFile

parquetFile**.**registerTempTable**(**"UserStatus"**);**

sqlContext**.**cacheTable**(**"UserStatus"**)**

tweets **=** sqlContext**.**sql**(**"SELECT \* FROM UserStatus order by FRIENDS\_COUNT desc"**)**

resultCount **=** tweets**.**count**()**

tweets**.**cache**()**

**print** resultCount

#tweets.collect()

SCountList **=** **[**0**]** **\*** resultCount

**for** i **in** range**(**0**,** resultCount**-**1**):**

SCountList**[**i**]** **=** sqlContext**.**sql**(**"select FRIENDS\_COUNT as statusCount from UserStatus order by FRIENDS\_COUNT"**).**collect**()[**i**].**statusCount

**print** TCountList

UserList **=** **[**0**]** **\*** resultCount

**for** i **in** range**(**0**,** resultCount**-**1**):**

UserList**[**i**]** **=** sqlContext**.**sql**(**"select TWEET\_USER\_SCREEN\_NAME as userName from UserStatus order by FRIENDS\_COUNT"**).**collect**()[**i**].**userName

#totRecords = sqlContext.sql("select count(1) as totRecords from tweetsperlang").collect()[0].totRecords

totRecords **=** 20

**print** totRecords

ind**=**np**.**arange**(**len**(**UserList**)**

width **=** 0.35

fig **=** plt**.**figure**()**

plt**.**bar**(**ind**,** SCountList**,** width**,** color**=**'g'**,** label **=** "distributions"**)**

params **=** plt**.**gcf**()**

plSize **=** params**.**get\_size\_inches**()**

params**.**set\_size\_inches**(** **(**plSize**[**0**]\***5**,** plSize**[**1**]\***5**)** **)**

plt**.**ylabel**(**'Friends count'**)**

plt**.**xlabel**(**'User Screen Names'**)**

plt**.**title**(**'Top Users With Highest Friends'**)**

plt**.**xticks**(**ind**+**width**,** UserList**)**

plt**.**legend**()**

plt**.**show**()**

**Top users with highest number of followers:**

####################################################

#Top 10 users with highest followers count##########

# Bar Chart and Pie Chart ##########################

####################################################

#Scala Notebook Logic

####################################################

val sqlContext **=** new org**.**apache**.**spark**.**sql**.**SQLContext**(**sc**)**

val dashdataDF **=** sqlContext**.**load**(**"jdbc"**,** Map**(**

"url" **->** "jdbc:db2://bluemix05.bluforcloud.com:50000/BLUDB:user=dash019411;password=2k3n2TR3x3mb;"**,**

"dbtable" **->** "DASH019411.NORED"**))**

dashdataDF**.**registerTempTable**(**"tweetdata"**)**

sqlContext**.**cacheTable**(**"tweetdata"**)**

val results **=** sqlContext**.**sql**(**"SELECT TWEET\_USER\_SCREEN\_NAME, MAX(TWEET\_USER\_FOLLOWERS\_COUNT) AS TWEET\_USER\_FOLLOWERS\_COUNT FROM tweetdata group by TWEET\_USER\_SCREEN\_NAME order by TWEET\_USER\_FOLLOWERS\_COUNT DESC LIMIT 10"**)**

results**.**cache**()**

results**.**count**()**

results**.**collect

results**.**repartition**(**1**).**saveAsParquetFile**(**"swift://notebooks.spark/Top10UsersWithFollowers.parquet"**)**

####################################################

#Python Notebook Logic

####################################################

# Import SQLContext and data types

**from** pyspark**.**sql **import** SQLContext

**from** pyspark**.**sql**.**types **import** **\***

**%**matplotlib inline

**import** matplotlib

**import** numpy **as** np

**import** matplotlib**.**pyplot **as** plt

# sc is an existing SparkContext.

sqlContext **=** SQLContext**(**sc**)**

parquetFile **=** sqlContext**.**read**.**parquet**(**"swift://notebooks.spark/Top10UsersWithFollowers.parquet"**)**

#print parquetFile

parquetFile**.**registerTempTable**(**"FCountPerUser"**);**

sqlContext**.**cacheTable**(**"FCountPerUser"**)**

tweets **=** sqlContext**.**sql**(**"SELECT \* FROM FCountPerUser order by TWEET\_USER\_FOLLOWERS\_COUNT desc"**)**

resultCount **=** tweets**.**count**()**

tweets**.**cache**()**

**print** resultCount

#tweets.collect()

FCountList **=** **[**0**]** **\*** resultCount

**for** i **in** range**(**0**,** resultCount**-**1**):**

FCountList**[**i**]** **=** sqlContext**.**sql**(**"select TWEET\_USER\_FOLLOWERS\_COUNT as followersCount from FCountPerUser order by followersCount"**).**collect**()[**i**].**followersCount

**print** FCountList

UsersList **=** **[**0**]** **\*** resultCount

**for** i **in** range**(**0**,** resultCount**-**1**):**

UsersList**[**i**]** **=** sqlContext**.**sql**(**"select TWEET\_USER\_SCREEN\_NAME as UserName from FCountPerUser"**).**collect**()[**i**].**UserName

#totRecords = sqlContext.sql("select count(1) as totRecords from tweetsperlang").collect()[0].totRecords

totRecords **=** 10

**print** totRecords

ind**=**np**.**arange**(**totRecords**)**

width **=** 0.35

fig **=** plt**.**figure**()**

plt**.**bar**(**ind**,** FCountList**,** width**,** color**=**'g'**,** label **=** "distributions"**)**

params **=** plt**.**gcf**()**

plSize **=** params**.**get\_size\_inches**()**

params**.**set\_size\_inches**(** **(**plSize**[**0**]\***5**,** plSize**[**1**]\***5**)** **)**

plt**.**ylabel**(**'Followers count'**)**

plt**.**xlabel**(**'Users'**)**

plt**.**title**(**'Distribution of tweets by Language'**)**

plt**.**xticks**(**ind**+**width**,** UsersList**)**

plt**.**legend**()**

plt**.**show**()**