Lab Tutorial

By Sridhar reddy Puli (spuli@calstatela.edu), Ram Dharan Donda (rdonda@calstatela.edu), Goutham kumar Pola (gpola@calstatela.edu) and Vinay Chennupati (vchennu@calstatela.edu)

06/10/2016

Yelp Data Analysis using Spark

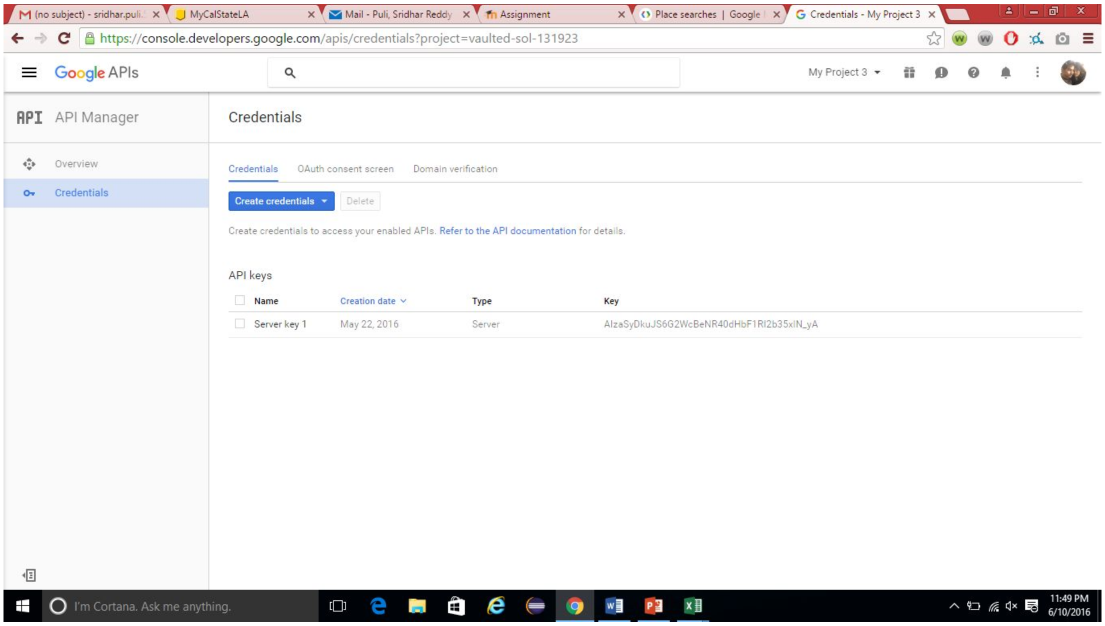
Objectives

In this hands-on lab, you will learn how to:

- Get data manually using REST API
- Create Spark cluster
- Train NLP system
- SQL commands to perform the analysis.
- Visualization

Exercise 1: Get data manually using REST API

1. Create Google API keys at https://developers.google.com/places/web-service/get-api-key



https://console.developers.google.com/apis/credentials?project=vaulted-sol-131923

Credentials

Create credentials to access your enabled APIs. Refer to the API documentation for details.

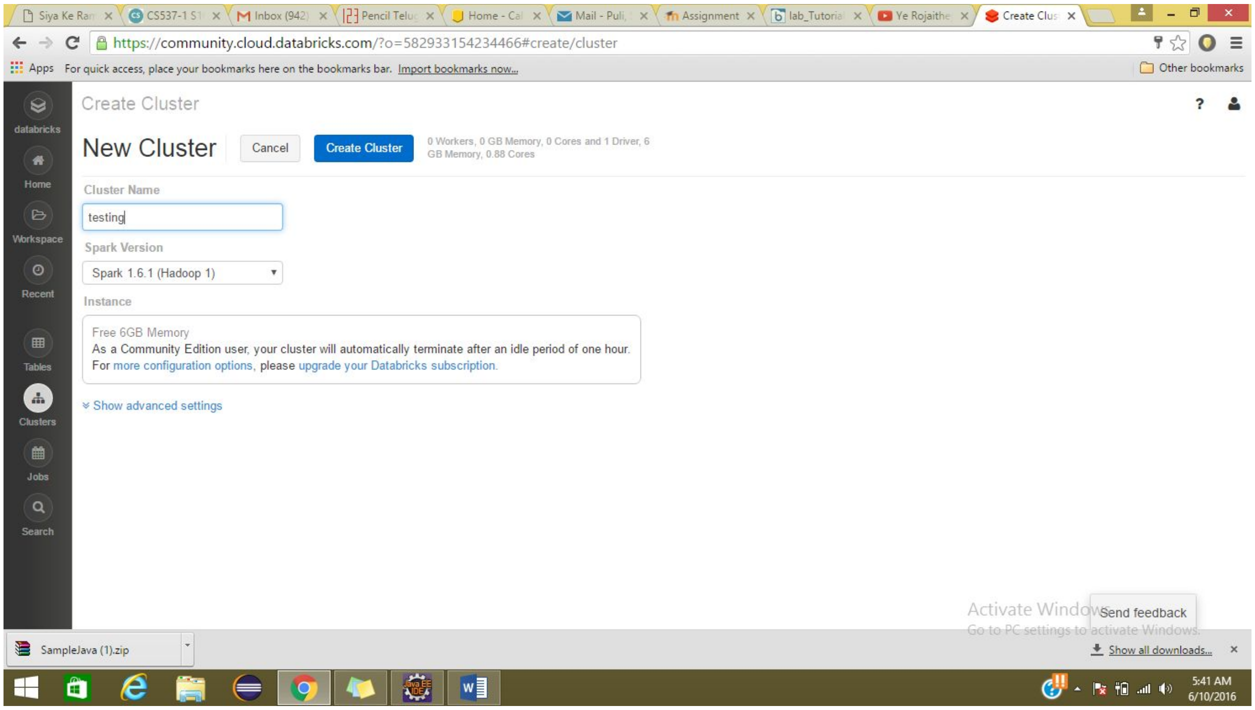
API keys

□	Name	Creation date	Type	Key
□	Server key 1	May 22, 2016	Server	AIzaSyDkuJS6G2WcBeNR40dHbF1RI2b35xIN_yA

2. Run the given Google places java code by replacing key values with your keys in the code.
3. Also run the given java code for Yelp data.
4. Result data will be stored in a file in your workspace.
5. Convert the Json data into csv using online tools and merge data into one file.

Exercise 2: Create Spark cluster and load data

1. Sign into your databricks account.
2. Go to Clusters option on the left and click on create cluster.
3. Give the cluster name and click create cluster.



Create Cluster

New Cluster

0 Workers, 0 GB Memory, 0 Cores and 1 Driver, 6 GB Memory, 0.88 Cores

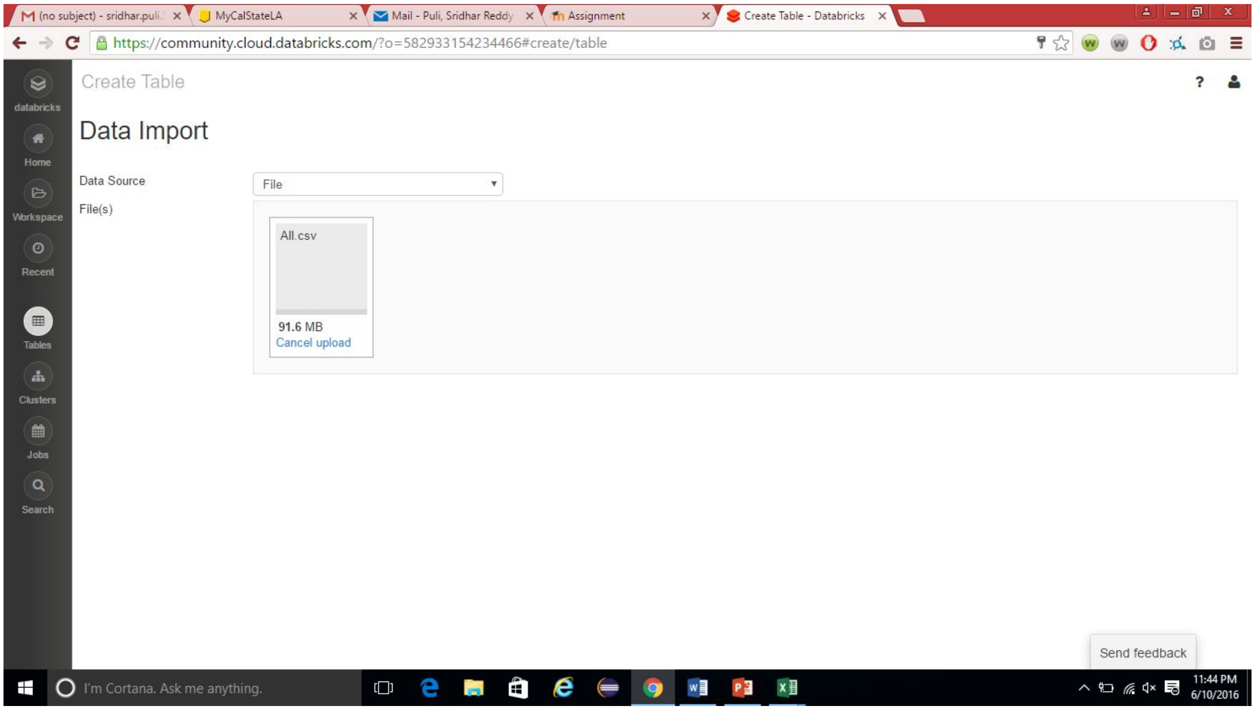
Cluster Name: testing

Spark Version: Spark 1.6.1 (Hadoop 1)

Instance

Free 6GB Memory
As a Community Edition user, your cluster will automatically terminate after an idle period of one hour.
For more configuration options, please upgrade your Databricks subscription.

4. Under tables section click on create table and select the file to upload.


(no subject) - sridhar.puli.
MyCalStateLA
Mail - Puli, Sridhar Reddy
Assignment
Create Table - Databricks
https://community.cloud.databricks.com/?o=582933154234466#create/table
databricks
Home
Workspace
Recent
Tables
Clusters
Jobs
Search
Send feedback
I'm Cortana. Ask me anything.
11:44 PM
6/10/2016

Create Table

Data Import

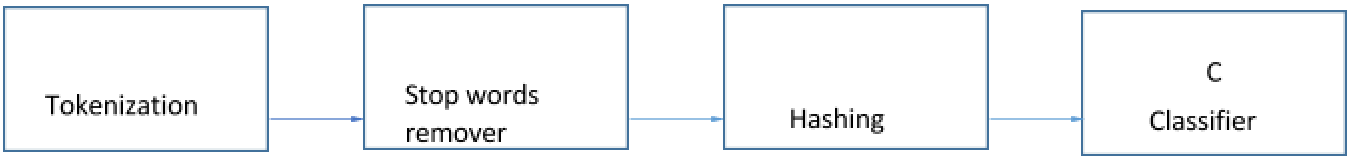
Data Source: File

File(s): All.csv 91.6 MB Cancel upload

Exercise 3: Train NLP

Pipeline stages:

Spark machine learning API makes it easier to combine multiple steps into a single workflow called as pipeline. The figure below shows the pipeline stages and data is transformed at each stage before passing it to the next.



Creating table for Natural Language processing:

i) Download the data from yelp challenge
ii) As shown above create a table by using "review.json" file and create a **Scala** notebook.
iii) Query the data from the table as shown below.

```
 val data = sqlContext.sql("SELECT CAST(stars as DOUBLE) as label, text from TABLE NAME")
   val splits = data.randomSplit(Array(0.80, 0.20), seed = 10)
```

iv) We use the split data as
splits[0]→ Training data
    splits[1]→ Test data

Don’t copy this in the notebook, this is just for understanding.

v)Use the below command to tokenize the data

```
import org.apache.spark.ml.feature.RegexTokenizer
val tokenizer = new RegexTokenizer()
  .setPattern("\\p{L}+").setMinTokenLength(3)
.setGaps(false)
  .setInputCol("text")
  .setOutputCol("words")

val tokenized_df=tokenizer.transform(splits(0))
```

vi) Use the below code to remove stop words
Run them in separate cells for better understanding

```
%sh wget http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words -O /tmp/stopwords
%fs cp file:/tmp/stopwords dbfs:/tmp/stopwords
val stopwords = sc.textFile("/tmp/stopwords").collect()

import org.apache.spark.ml.feature.StopWordsRemover
// Set params for StopWordsRemover
val remover = new StopWordsRemover()
  .setStopWords(stopwords) // This parameter is optional
  .setInputCol("words")
  .setOutputCol("filtered")

// Create new DF with Stopwords removed
val filtered_df = remover.transform(tokenized_df)
```

vii)Use the below code for hashing after removing the stop words

```
import org.apache.spark.ml.feature.{HashingTF, Tokenizer}

val hashingTF = new HashingTF()
  .setNumFeatures(1000)
  .setInputCol("filtered")
  .setOutputCol("features")
```

viii) Create a Naïve Bayes model by using below code

```
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.PipelineStage
import org.apache.spark.ml.classification.NaiveBayes
```

```
val nb = new NaiveBayes()
nb.setModelType("multinomial")
```

ix) Set the piple line stages

```
val pipeline = new Pipeline().setStages(Array(tokenizer,remover,hashingTF,nb))
```

x)Fit the training data into pipeline

```
 val lrModel = pipeline.fit(splits(0))
```

xi)Test the model , with the test data

```
val dd = lrModel.transform(splits(1))
```

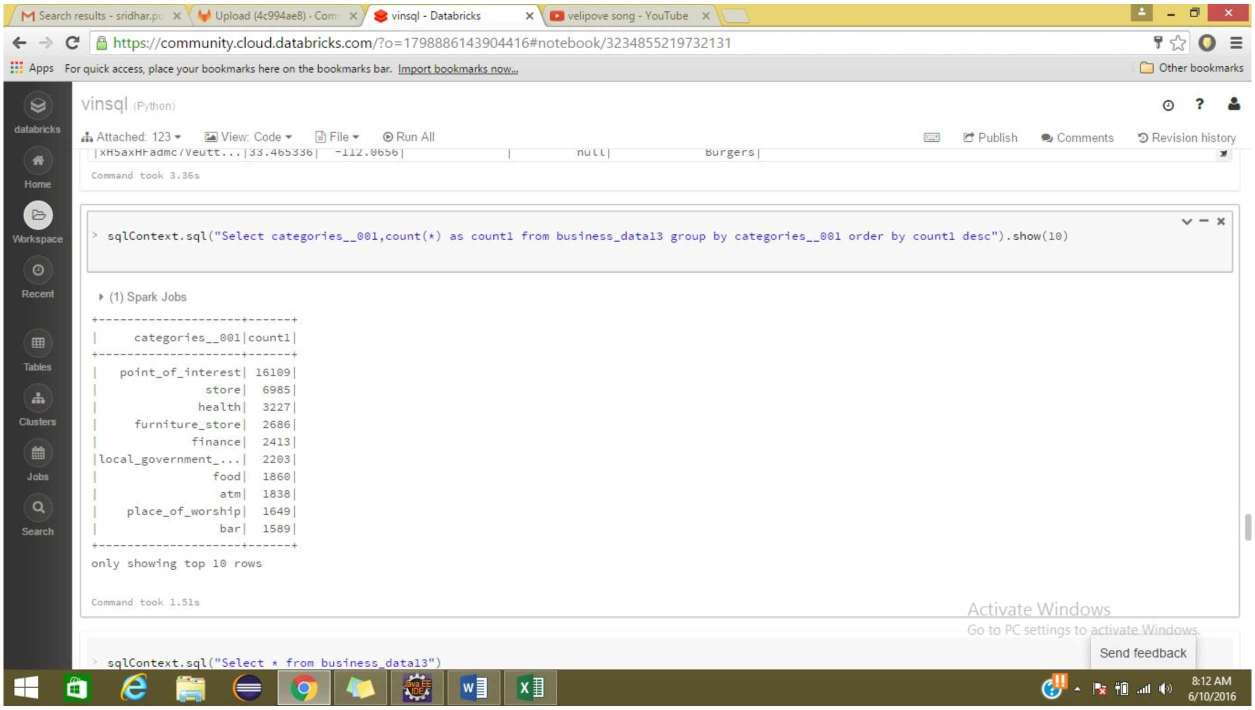
xii) To see the results of the test data

```
val res=dd.select("label","prediction")
display(res)
```

Exercise 4: SQL commands to perform the analysis

1. To show top ten categories

   ```
   sqlContext.sql("Select categories__001,count(*) as count1 from business_data13 group by
   categories__001 order by count1 desc").show(10)
   ```



```
|xH5axHFadmc7Veutt...|33.465336|  -112.0656|          |          null|          Burgers|
Command took 3.36s
```

```
> sqlContext.sql("Select categories__001,count(*) as count1 from business_data13 group by categories__001 order by count1 desc").show(10)
```

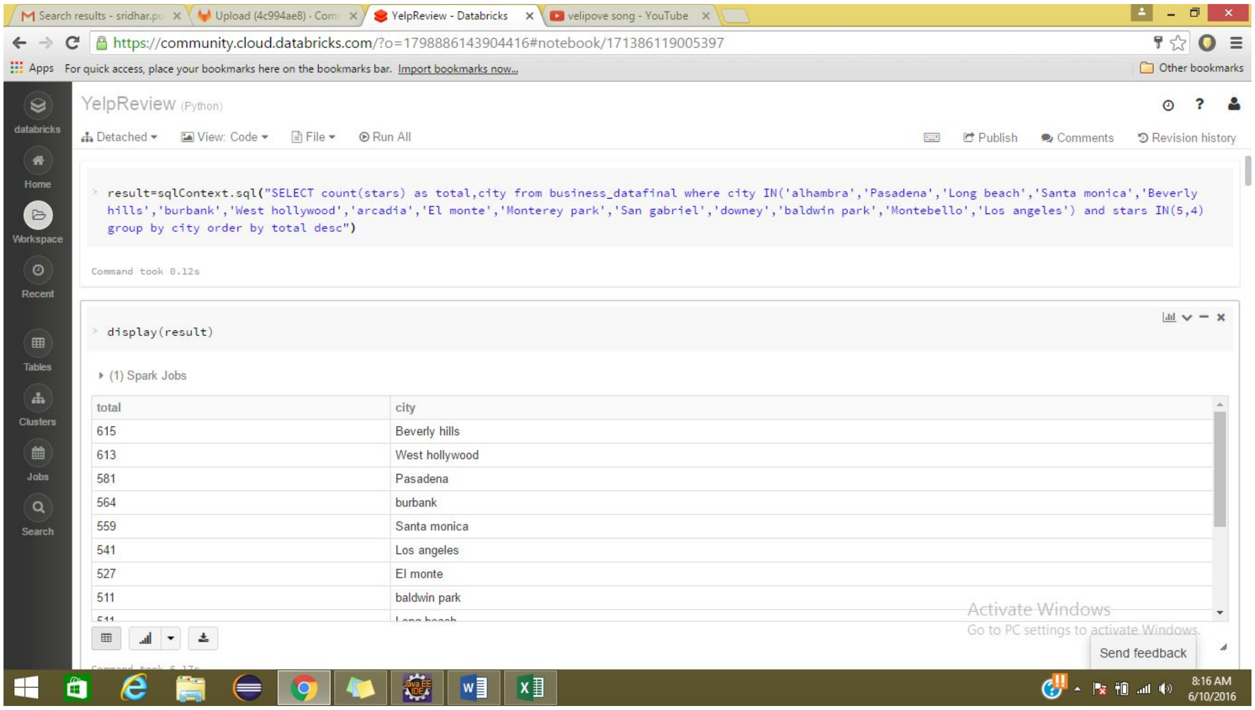
```
+--------------------+------+
|     categories__001|count1|
+--------------------+------+
|   point_of_interest| 16109|
|               store|  6985|
|              health|  3227|
|     furniture_store|  2686|
|             finance|  2413|
|local_government_...|  2203|
|                food|  1860|
|                 atm|  1838|
|    place_of_worship|  1649|
|                 bar|  1589|
+--------------------+------+
only showing top 10 rows

Command took 1.51s
```

```
> sqlContext.sql("Select * from business_data13")
```

2. For showing total number of 4 and 5 star rating businesses for every area in los angeles.

```
result=sqlContext.sql("SELECT count(stars) as total,city from business_datafinal where city IN('alhambra','Pasadena','Long beach','Santa monica','Beverly hills','burbank','West hollywood','arcadia','El monte','Monterey park','San gabriel','downey','baldwin park','Montebello','Los angeles') and stars IN(5,4) group by city order by total desc")
```



```
result=sqlContext.sql("SELECT count(stars) as total,city from business_datafinal where city IN('alhambra','Pasadena','Long beach','Santa monica','Beverly
hills','burbank','West hollywood','arcadia','El monte','Monterey park','San gabriel','downey','baldwin park','Montebello','Los angeles') and stars IN(5,4)
group by city order by total desc")

Command took 0.12s
```

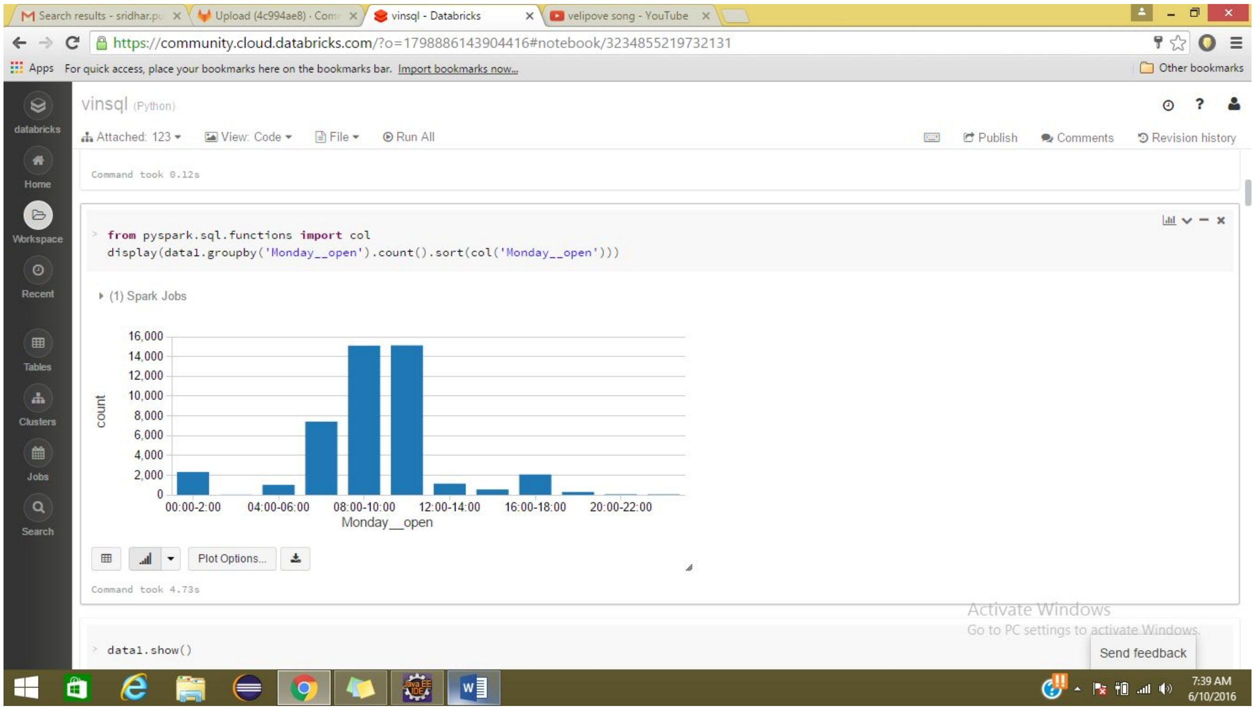
```
display(result)
```

total	city
615	Beverly hills
613	West hollywood
581	Pasadena
564	burbank
559	Santa monica
541	Los angeles
527	El monte
511	baldwin park
511	Long beach

3. Similarly copy and paste the given SQL commands to find different kinds of analysis.

Exercise 5: Visualization

1. Select the chart type below the result in databricks to instantly show the visualization.



```
Command took 0.12s
```

```
from pyspark.sql.functions import col
display(data1.groupby('Monday__open').count().sort(col('Monday__open')))
```

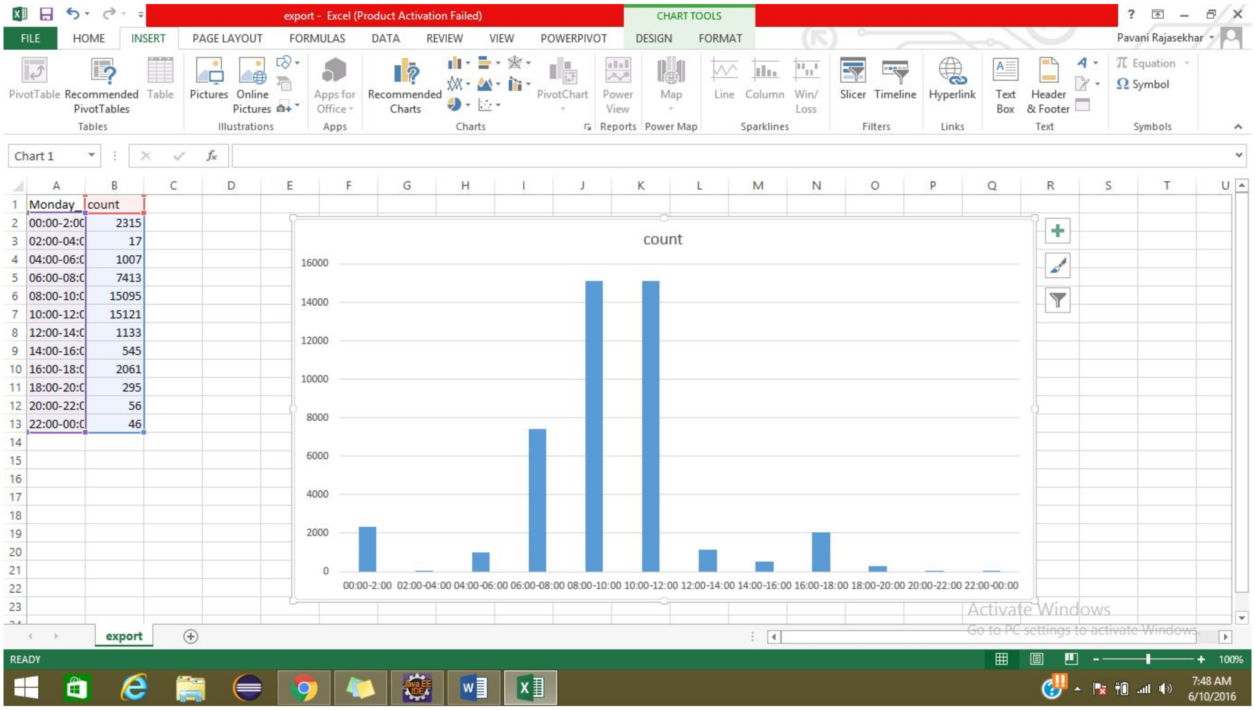
(1) Spark Jobs

```
Command took 4.73s
```

```
data1.show()
```

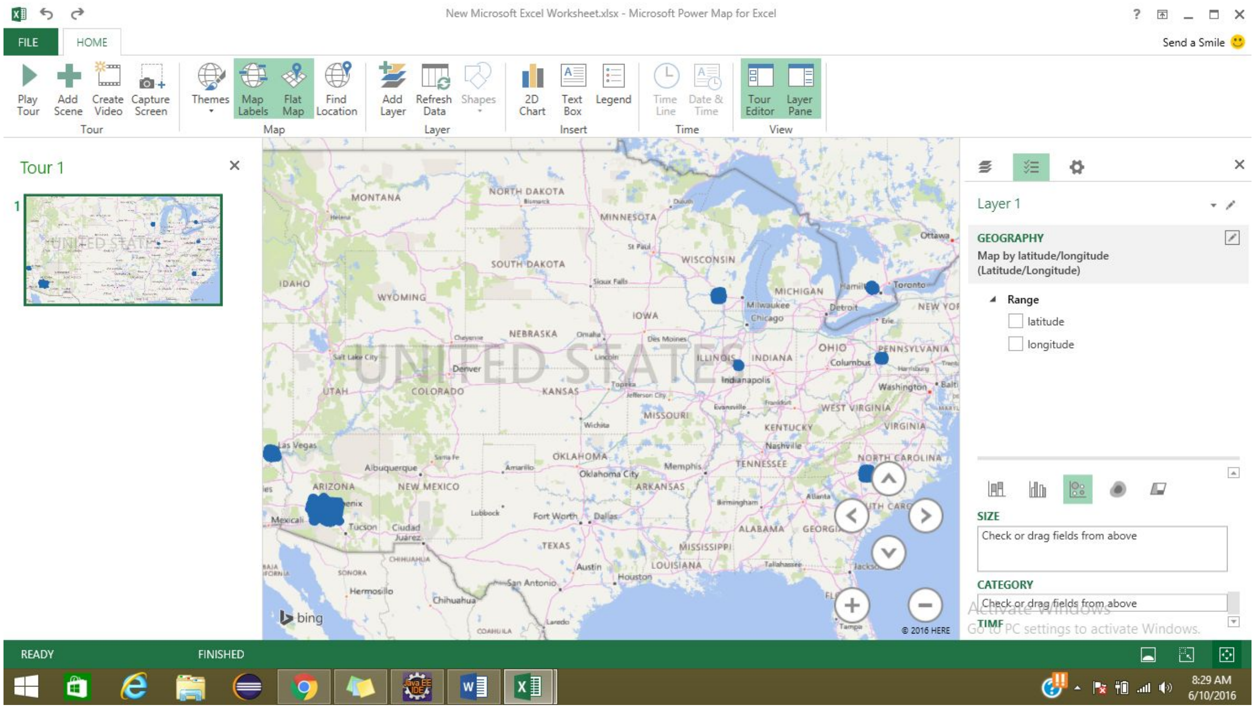
2. Or, click the download button below the result.

3. Open the downloaded file, under insert tab select the chart best suited.



	A	B
1	Monday_	count
2	00:00-2:00	2315
3	02:00-04:0	17
4	04:00-06:0	1007
5	06:00-08:0	7413
6	08:00-10:0	15095
7	10:00-12:0	15121
8	12:00-14:0	1133
9	14:00-16:0	545
10	16:00-18:0	2061
11	18:00-20:0	295
12	20:00-22:0	56
13	22:00-00:0	46

4. To visualize location type of results on map, convert csv file to excel and click on map button under insert tab.



Map by latitude/longitude (Latitude/Longitude)

□ latitude

□ longitude

SIZE: Check or drag fields from above

CATEGORY: Check or drag fields from above