**SREERANJANI THEVANGUDIRAJANGAM**

HU Extension School E-63 Big Data Analytics

**BIG DATA CASE STUDY INTERNET OF THINGS**

**STUDY OF NATIONAL CLIMATIC DATA**

PROBLEM STATEMENT

Analyze the storm events database and find the total number of storm episodes happened across all the states in United States in year 2015. Find how many deaths and injuries occurred and find the minimum, maximum and average property damages caused by the various types of storm.

Join the storm events database with storm fatality and storm locations to identify the exact location of the events and fetch its fatality information.

OVERVIEW OF TECHNOLOGIES

|  |  |
| --- | --- |
| Technology /Tools | Description |
| Platform | ClouderaVM CDH 5.5 on VMware Fusion |
| Data analysis and Computations | Spark RDD, Spark Dataframes, SparkSQL with Python apis |
| Visualization | Python 2.7 |
| Spark | Spark 1.5, HDFS, spark-csv |

This project uses the version of Spark 1.5 , it utilizes the new feature of **automatic schema extraction** defined in Spark 1.5 . It utilizes the **spark-csv** external package provided by Databricks to load the dataset into HDFS.

We create lots of spark dataframes through out this project and execute the spark-sql quieries for our analysis and computation. The data visualization is done with histogram, piechart and scatterplot using python scripts.

DATASETS:

<http://www1.ncdc.noaa.gov/pub/data/swdi/stormevents/csvfiles/>

This database contains 3 different types of files (details, fatalities, locations) from year 1951 to 2016. This project uses the data for the year 2015.

Renamed files for readability:

StormEvents1\_details-2015.csv - Size (50MB)

StormEvents\_fatalities-2015.csv - Size (57KB)

StormEvents\_locations-2015.csv - Size (4.6MB)

YOU TUBE URLs:

National Centers for Environmental Information (NCEI) formerly called National climatic data center (NCDC) preserves the nation’s treasure of climate and access to historical weather data and information. The repository contains 3 different types of files (details, fatalities, locations) from year 1951 to 2016. This case study analyzes the Storm events database provided by NCDC and does the following computations.

Problem Statement:

1. How many total storm events happened in 2015?

2. How many deaths occurred in 2015 due to the storm events?

3. How many injuries occurred in 2015 due to the storm events?

4. What was the maximum, minimum and average property damage caused by storm in 2015?

5. What was the total count of storm episodes across all states in US in the year 2015?

6. What was the total damage to the properties due to the different types of storm events across all states in US in 2015?

7. Join the stormdetails with stormlocations data to get the exact location with lat, long attributes.

8. Join Stormfatalities table on event\_id and get the fatality location.

9. Save the result of dataframe from the last join query into a CSV file.

Visualization:

10. Visualize the states against the sum of property damages as frequency in a bar graph.

11. Visualize the count of different types of storm episodes in a pie chart.

12. Visualize and create a scatter plot of month of the year against the storm episode.

Solution:

This case study is done on cloudera quickstart VM. pyspark api, spark dataframes, spark sql programming are used to address the use cases.

Platform: Cloudera VM CDH 5.5

Spark 1.5.0

Python 2.6.6

Datasets:

StormEvents1\_details-2015.csv - Size (50MB)

StormEvents\_fatalities-2015.csv - Size (57KB)

StormEvents\_locations-2015.csv - Size (4.6MB)

Script File: stormeventsprojectpy.py

Step 1:

* Since we use Spark 1.5, we can utilize the spark-csv external package provided by Databricks to load our dataset.
* Used the spark-csv and referred this documentation. https://github.com/databricks/spark-csv
* In order to include the spark-csv package, we must start pyspark with the folowing argument:
* $pyspark --packages com.databricks:spark-csv\_2.10:1.2.0 (or) to run a script
* $spark-submit --packages com.databricks:spark-csv\_2.10:1.2.0 stormprojectpy.py

Step 2:

* The spark-csv does the AUTOMATIC SCHEMA EXTRACTION for us.
* The command for loading the CSV file StormEvents1\_details-2015.csv is
* df = sqlContext.read.format('com.databricks.spark.csv').options(header='true').load('final\_project/StormEvents1\_details-2015.csv')

Let us verify the automatic schema extraction:

df.dtypes



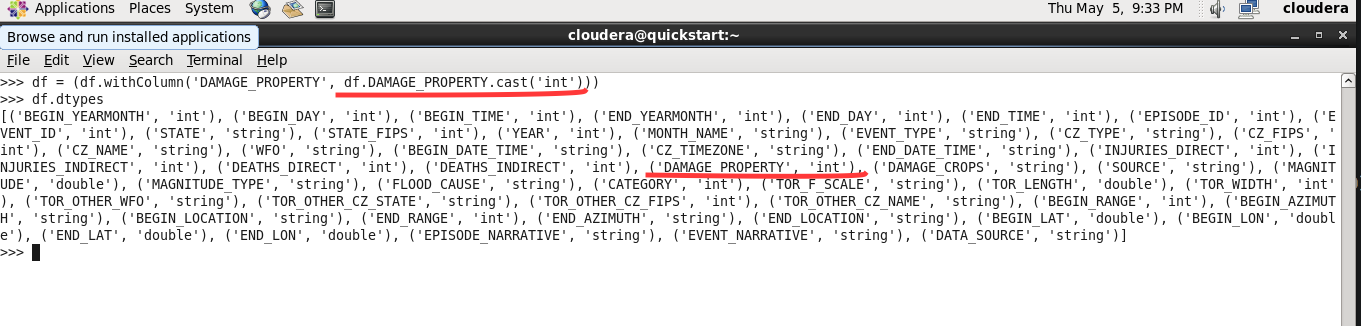
Step 3:

Most of the column types are recognized correctly, I need the DAMAGE\_PROPERTY as int, but it was recognized as String. Its datatype is converted to int as follows:

df = (df.withColumn('DAMAGE\_PROPERTY', df.DAMAGE\_PROPERTY.cast('int')))

Now verify its type.

df.dtypes

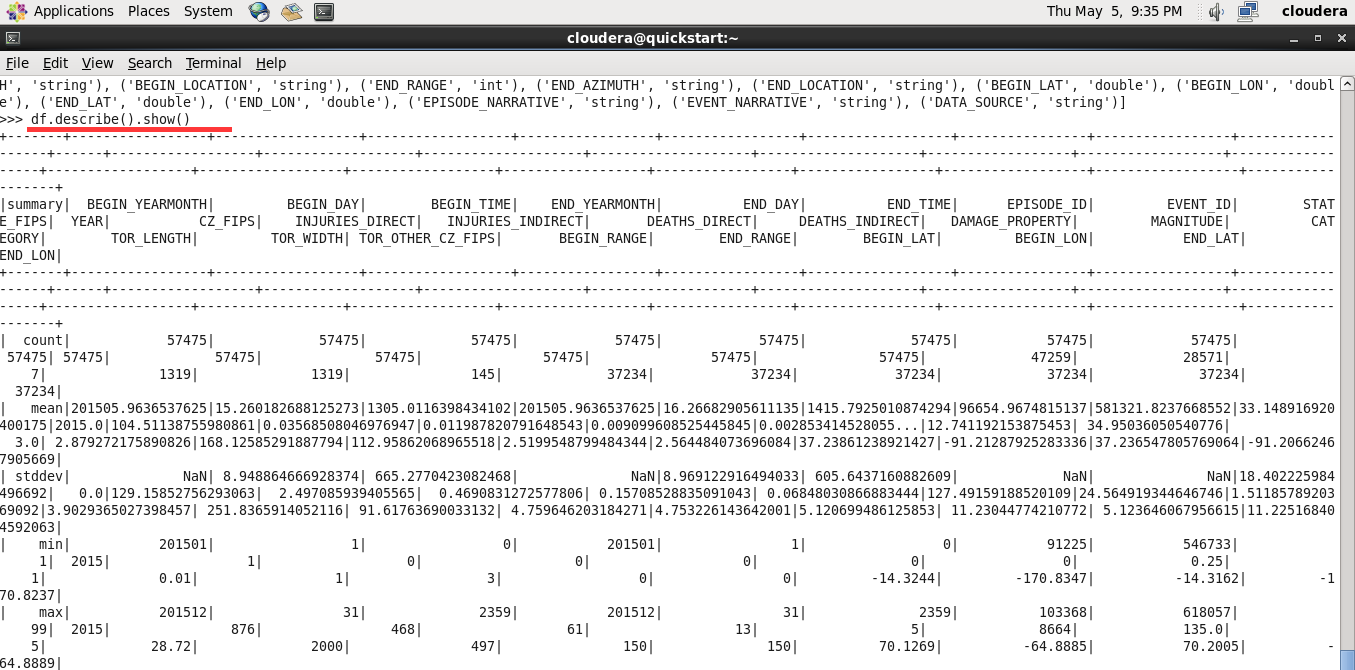


Step 4:

SUMMARY STATISTICS:

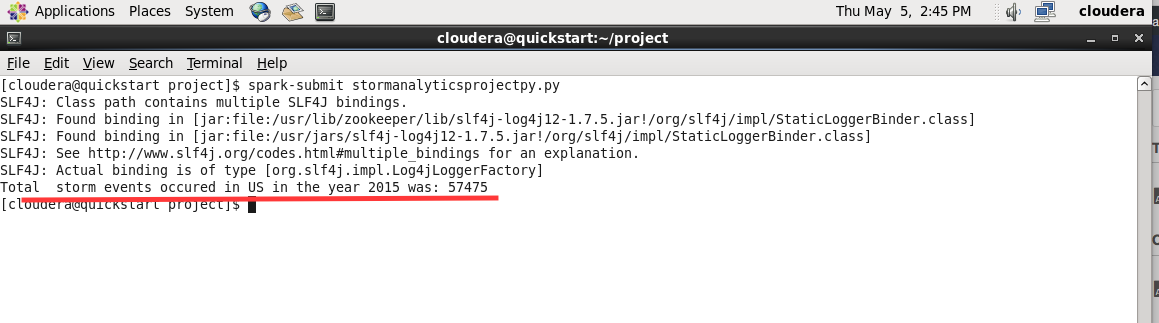
Spark-csv gives the neat summary statistics of all columns with count, min, max, mean, stddev using the following command.

df.describe().show()



Problem 1: How many total storm events happened in 2015?

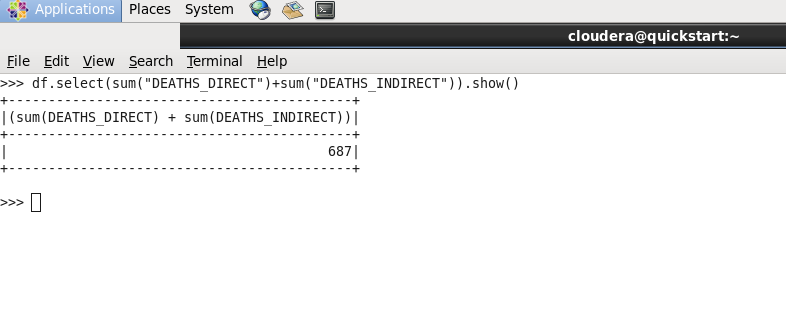
df.count()



Problem 2: How many deaths occurred in 2015 due to the storm events?

df.select(sum("DEATHS\_DIRECT")+sum("DEATHS\_INDIRECT")).show()

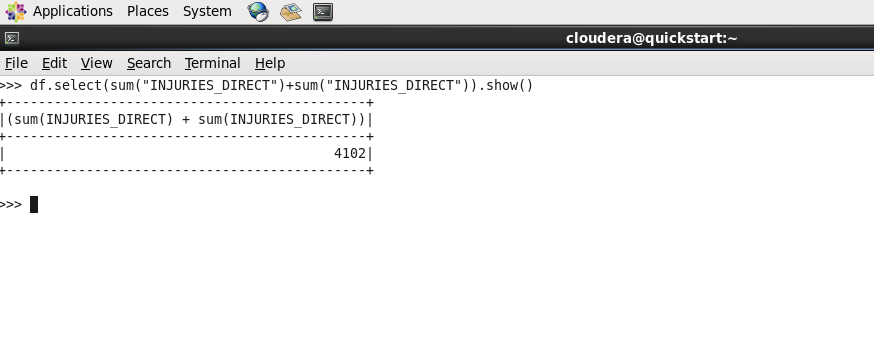
687



Problem 3: How many injuries occurred in 2015 due to the storm events?

df.select(sum("INJURIES\_DIRECT")+sum("INJURIES\_DIRECT")).show()

4102

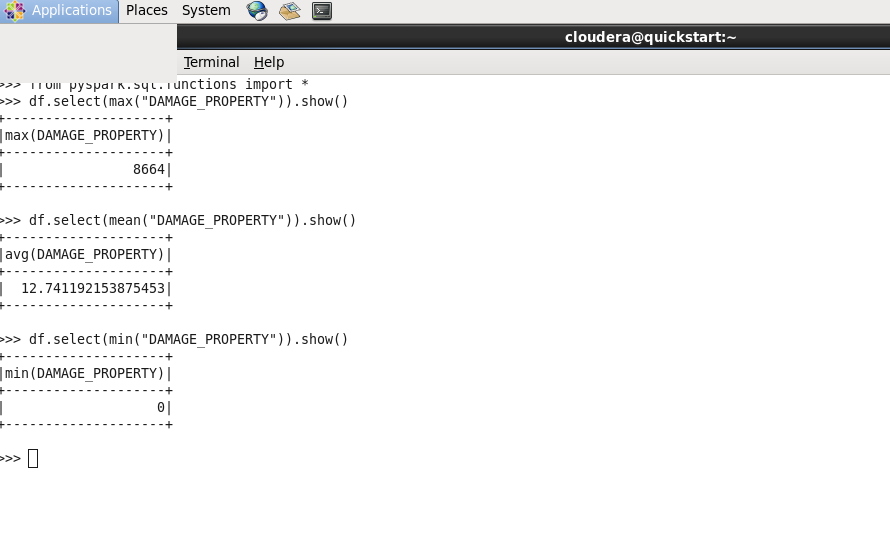


Problem 4: What was the maximum, minimum and average property damage caused by storm in 2015?

df.select(max("DAMAGE\_PROPERTY")).show()

df.select(min("DAMAGE\_PROPERTY")).show()

df.select(mean("DAMAGE\_PROPERTY")).show()

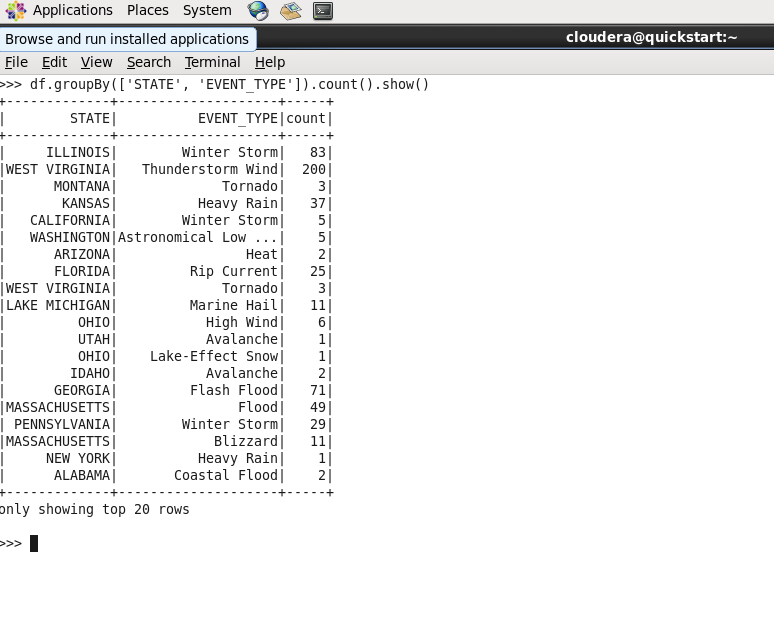


Problem 5: What was the total count of storm episodes across all states in US in the year 2015?

stateeventsdf = df.groupBy(['STATE', 'EVENT\_TYPE']).count()

stateeventsdf.show(50)

The dataframe is grouped by state and event type and the count is calculated.

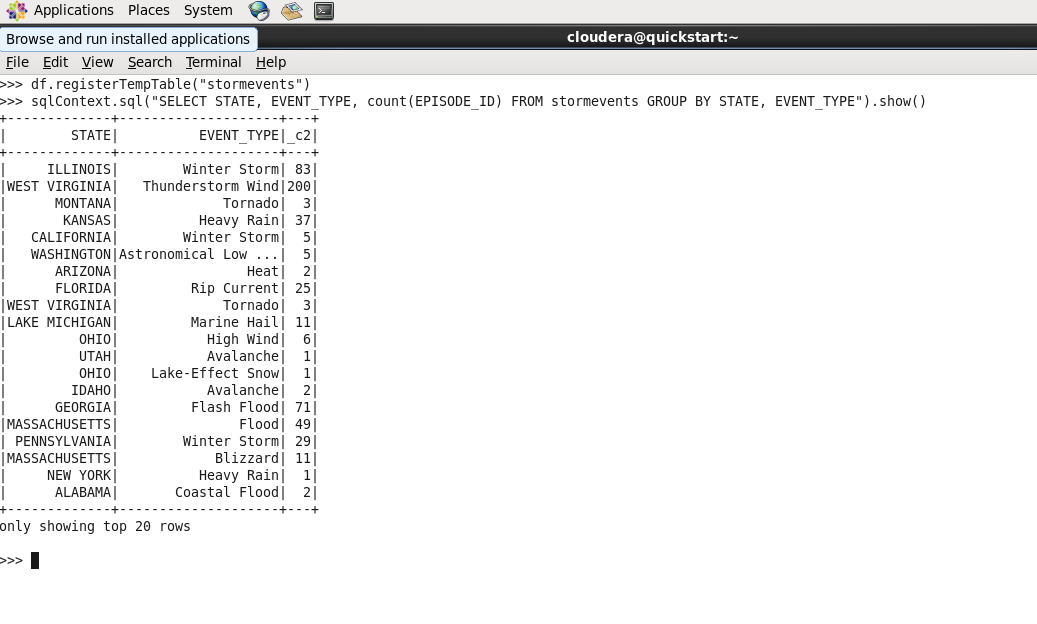


We can register the dataframe as table and query the table using SQL as follows.

df.registerTempTable("stormevents")

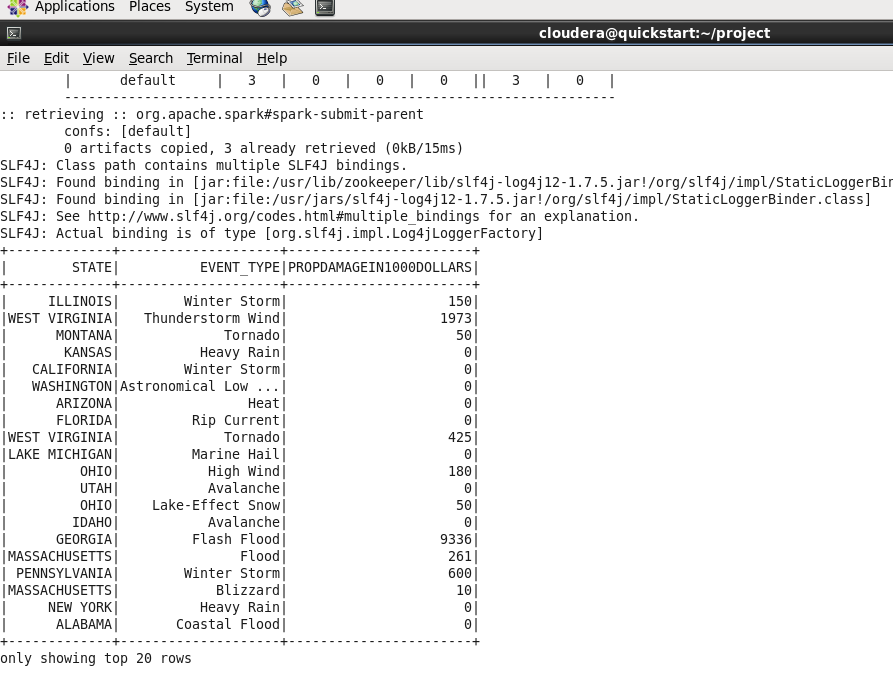
sqlContext.sql("SELECT STATE, EVENT\_TYPE, count(EPISODE\_ID) FROM stormevents GROUP BY STATE, EVENT\_TYPE").show()

Verify the last dataframe output with this output and both are same.



Problem 6: What was the total damage to the properties due to the different types of storm events across all states in US in 2015?

sqlContext.sql("SELECT STATE, EVENT\_TYPE, SUM(DAMAGE\_PROPERTY) AS PROPDAMAGEIN1000DOLLARS FROM ` GROUP BY STATE, EVENT\_TYPE").show()



Problem 7. Join the stormdetails with stormlocations data to get the exact location with lat, long attributes

We create a dataframe df4 for the StormEvents\_locations2015.csv, register the df4 as a table stormlocations and join this table with stormevents table to get the exact location, latitude, longitude for the different event ids.

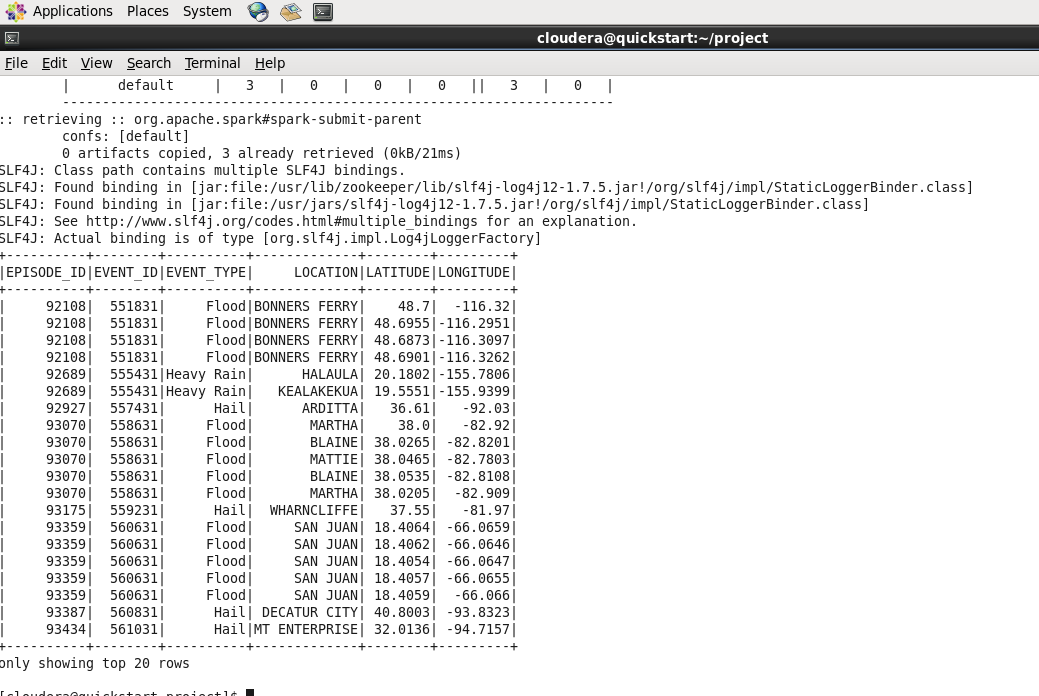
df4 = sqlContext.read.format('com.databricks.spark.csv').options(header='true', inferschema='true').load('final\_project/StormEvents\_locations2015.csv')

df4.registerTempTable("stormlocations")

df5 = sqlContext.sql("SELECT se.EPISODE\_ID, sl.EVENT\_ID, se.EVENT\_TYPE, sl.LOCATION, sl.LATITUDE, sl.LONGITUDE FROM stormevents se, stormlocations sl WHERE se.EVENT\_ID = sl.EVENT\_ID")

df5.show(20)

Below is the output on the console for top 20 rows.



Problem 8: Join StormEventsdetails data with Stormfatalities data and get the fatality location.

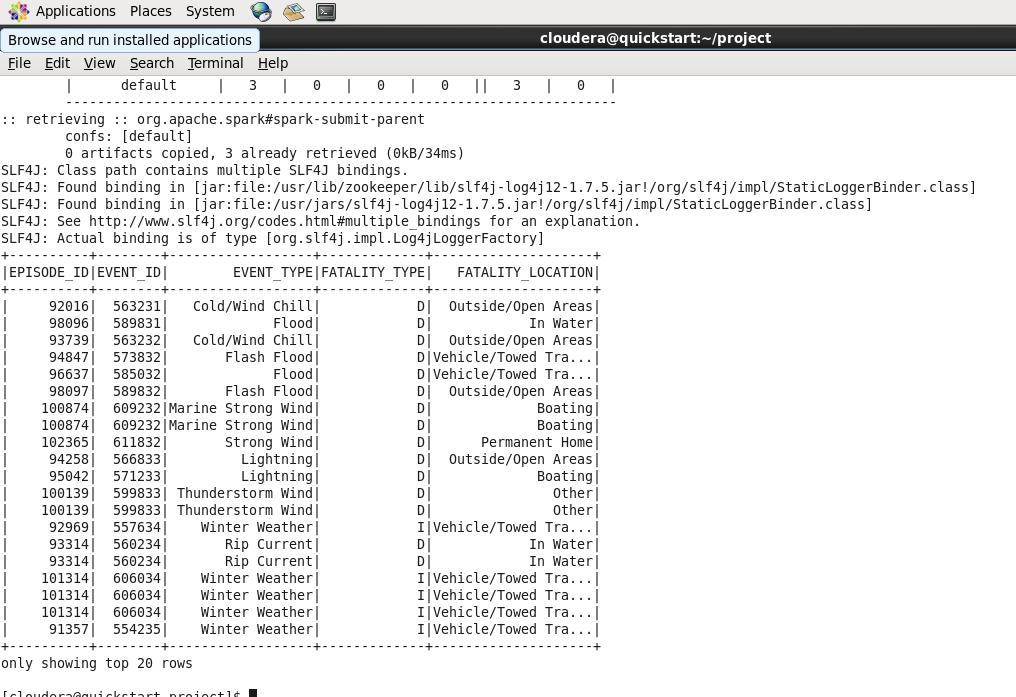
We create another dataframe df2 for the stormfatalities data as

df2 = sqlContext.read.format('com.databricks.spark.csv').options(header='true', inferschema='true').load('final\_project/StormEvents\_fatalities2015.csv')

df2.registerTempTable("stormfatalities")

df3 = sqlContext.sql("SELECT se.EPISODE\_ID, sf.EVENT\_ID, se.EVENT\_TYPE, sf.FATALITY\_TYPE, sf.FATALITY\_LOCATION FROM stormevents se, stormfatalities sf WHERE se.EVENT\_ID = sf.EVENT\_ID")

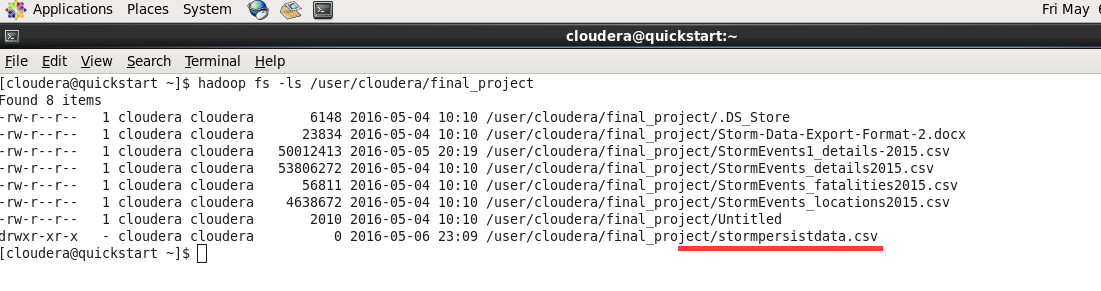
df3.show(20)

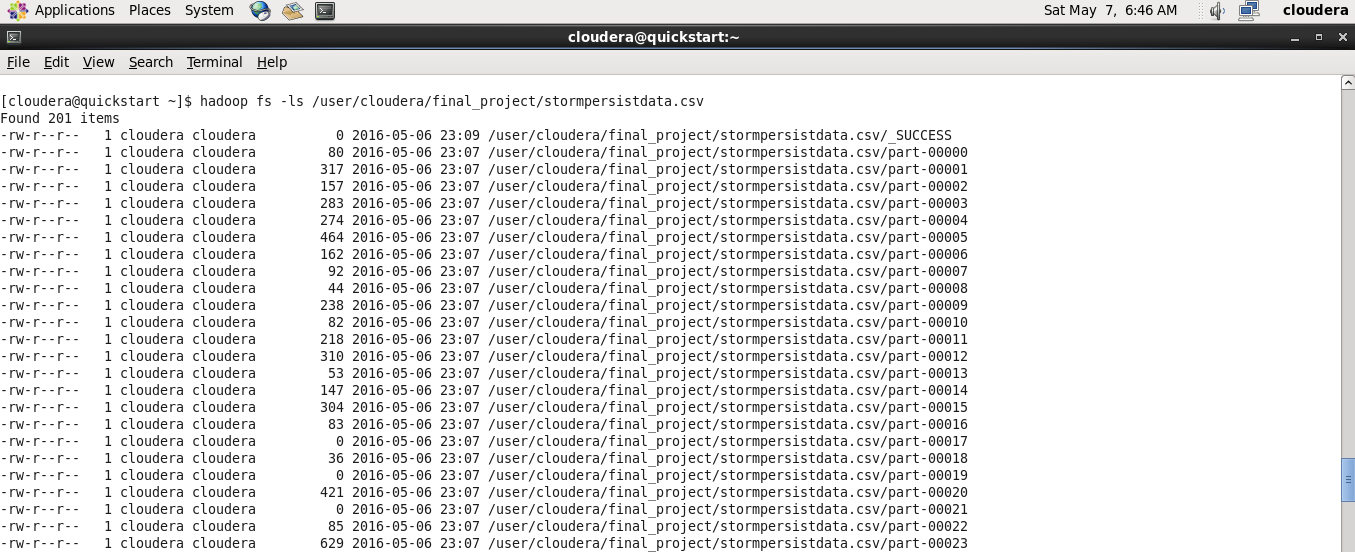


Problem 9: Save the result of dataframe from the last join query into a CSV file.

df3.write.format('com.databricks.spark.csv').save('final\_project/stormpersistdata.csv')

The result dataframe df3 is saved in HDFS as stormpersistdata.csv.

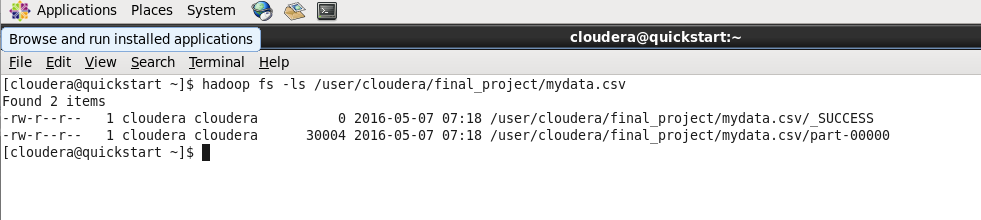




We can repartition the dataframe to write into a single CSV file using the following command.

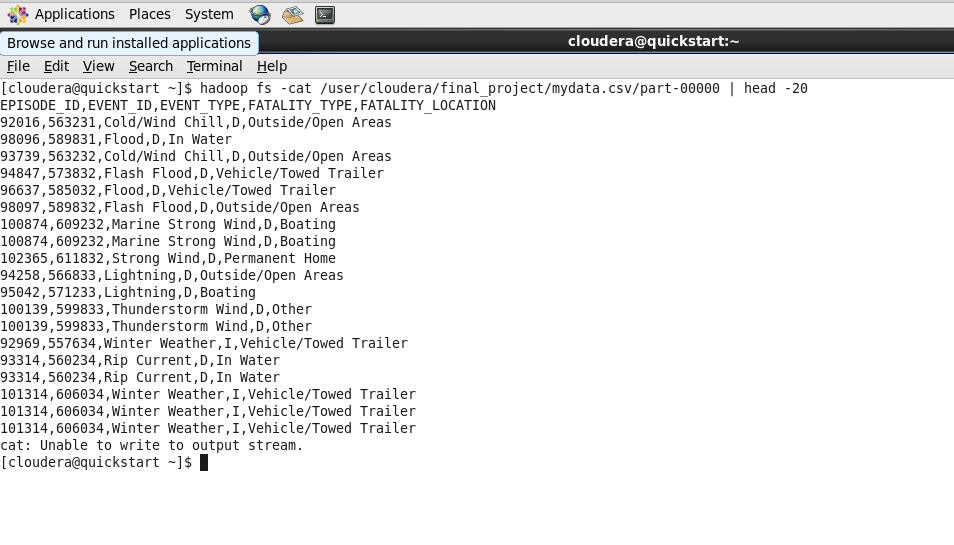
df3.repartition(1).write.format("com.databricks.spark.csv").option("header", "true").save("final\_project/mydata.csv")

Now instead of huge multiple part-xxxxx files, we get the result in a single CSV file mydata.csv



Display the result of CSV file.(top 20 rows)

hadoop fs -cat /user/cloudera/final\_project/mydata.csv/part-00000 | head -20



VISUALIZATION

10. Visualize the states against the sum of property damages as frequency in a bar graph.

Dataset: stateproperty.csv

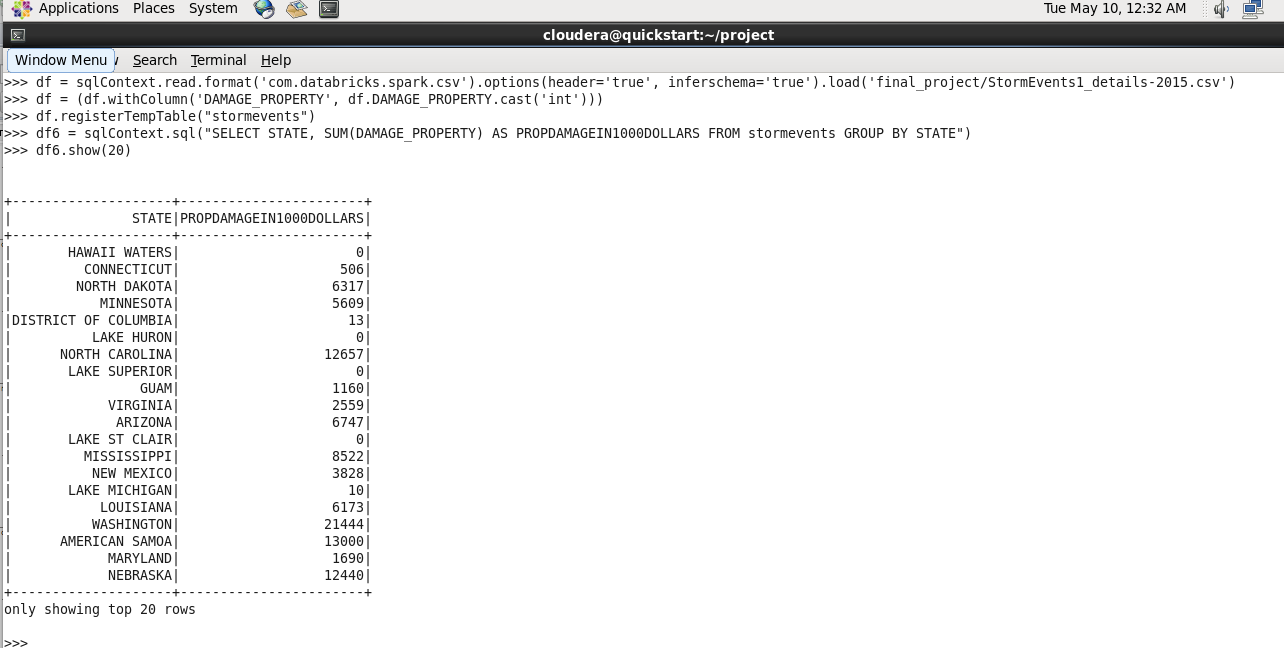
Script: histpy.py

Steps:

* We calculate the dataframe for sum of property damages for all the states and save the resulting dataframe as a csv file.
* Here is the command
* df6 = sqlContext.sql("SELECT STATE, SUM(DAMAGE\_PROPERTY) AS PROPDAMAGEIN1000DOLLARS FROM stormevents GROUP BY STATE")
* df6.repartition(1).write.format("com.databricks.spark.csv").option("header", "true").save("final\_project/stateproperty")

We created the stateproperty.csv data using the above query and we visualize it with bar graph.

Here is the python script to create the bar graph of states with total property damage.



import matplotlib.pyplot as plt

import pandas as pd

#Load the csv, specify the columns to load, skip the header and create the Dataframe using pandas

data = pd.read\_csv('stateproperty.csv', sep=',',names=['STATE','PROPERTY DAMAGE'], skiprows=1, index\_col =0)

data.plot(kind='bar')

#Label X, Y axes and add a title to the plot

plt.ylabel('propertydamageinthousands')

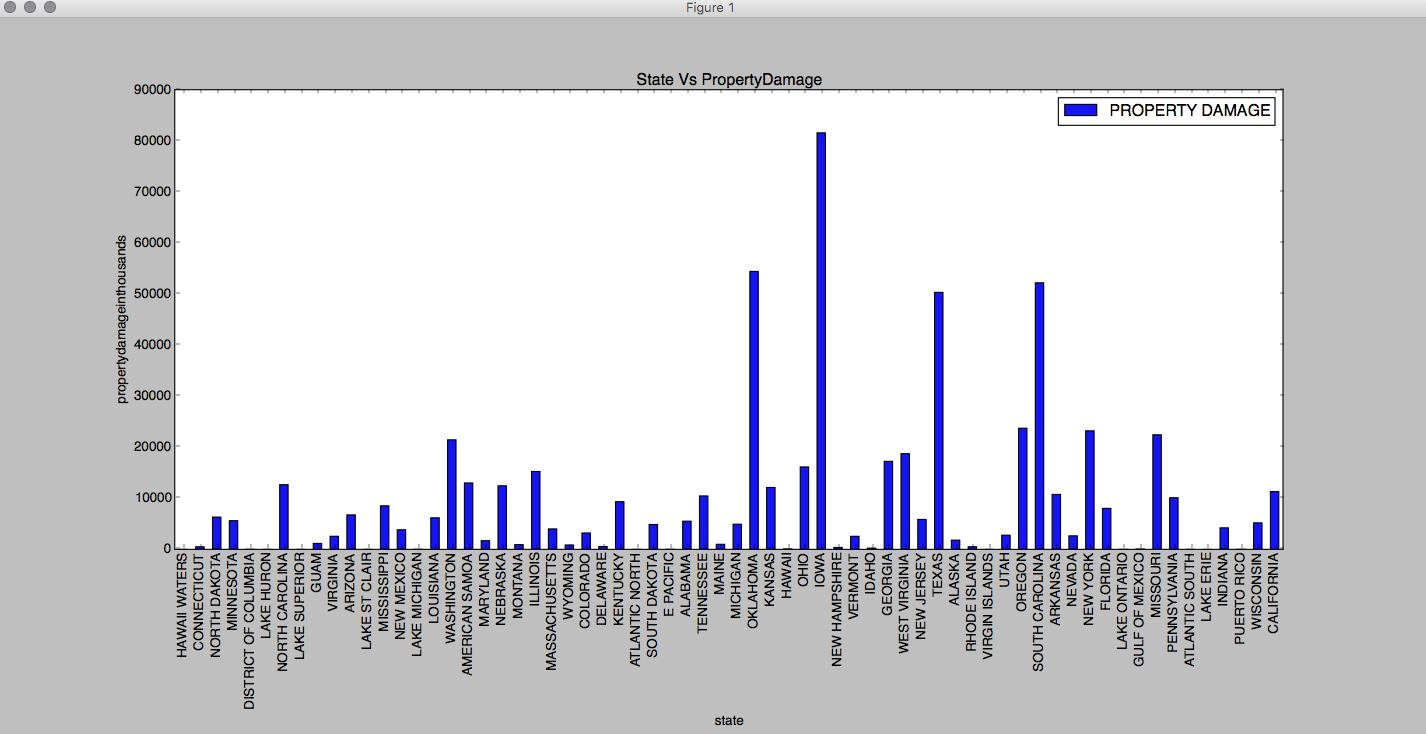
plt.xlabel('state')

plt.title('State Vs PropertyDamage')

plt.show()

Code Description:

* This python script uses matplotlib package and pandas.
* Import the packages and load the external CSV file to create a dataframe in Pandas.
* The CSV uses the columns STATE, PROPERTY DAMAGE, skip the header.
* Plot the data using the plot function of dataframe.
* Add X label, Y label and title.
* Call the show method to display the canvas.



On visualizing we found that the total damage to the property is higher for Iowa, whereas Oklahoma stood second with Texas and South Carolina stood third.

11. Visualize the count of different types of storm episodes in a pie chart.

Dataset: stormtypecounts.csv

Script: piechartpy.py

Steps:

* Like the previous problem , for this the dataset stormtypecounts.csv is derived from executing the SQL commands below.
* df = sqlContext.read.format('com.databricks.spark.csv').options(header='true').load('final\_project/StormEvents1\_details-2015.csv')
* df6 = sqlContext.sql("SELECT EVENT\_TYPE, COUNT(EPISODE\_ID) AS STORMCOUNTS FROM stormevents GROUP BY EVENT\_TYPE ")
* df6.repartition(1).write.format("com.databricks.spark.csv").option("header", "true").save("final\_project/stormtype")
* After renaming the file part-00000 to stormtypecounts.csv, we visualize the data in a Pie chart

The following script is used to generate it.

import matplotlib.pyplot as plt

import pandas as pd

#load the CSV, specify the column names, skip the header and create the dataframe using pandas

data = pd.read\_csv('stormtypecounts.csv', sep=',', names=['EVENT\_TYPE','STORMCOUNTS'], skiprows=1, index\_col=None)

#get the columns and save it in variables for loading the data

x = data[data.columns[0]]

y = data[data.columns[1]]

colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue', 'pink', '#f7fcb9', '#c994c7', '#bcbddc','#2ca25f','#fc9272']

# Used legends to display the labels separately as adding the labels causes ovrlapping in the pie chart

percent = 100.\*y/y.sum()

patches, texts = plt.pie(y, colors=colors, shadow=True, startangle=90, radius=1.2)

labels = ['{0} - {1:1.2f} %'.format(i,j) for i,j in zip(x, percent)]

sort\_legend = True

if sort\_legend:

patches, labels, dummy = zip(\*sorted(zip(patches, labels, y),

key=lambda x: x[2],

reverse=True))

plt.legend(patches, labels, loc='best', bbox\_to\_anchor=(-0.1, 1.),

fontsize=9)

plt.show()

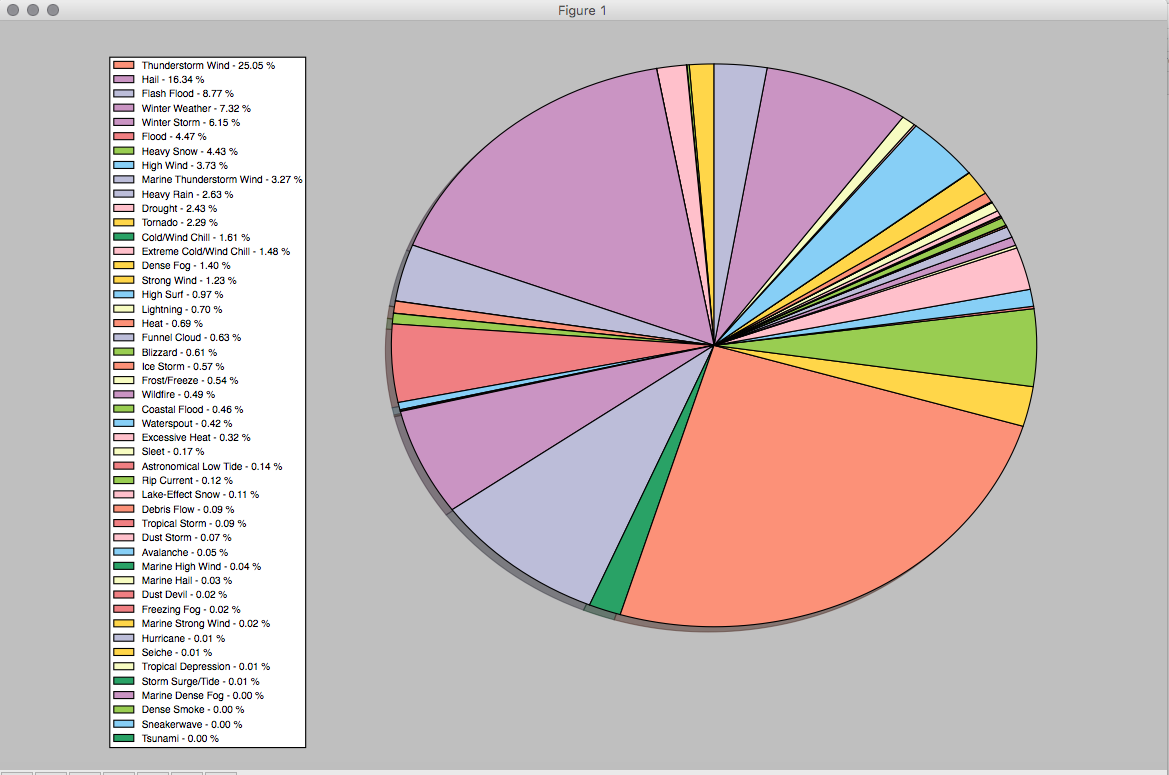
Code Description:

* This python script uses matplotlib package and pandas.
* Import the packages and load the external CSV file to create a dataframe in Pandas.
* The CSV uses the columns 'EVENT\_TYPE, 'STORMCOUNTS', skip the header.
* X ,Y holds the column 1 and column 2 values.
* Plot the data using the pie function in plot of dataframe.
* Adding the labels to the piechart causes overlapping, so we separately create the legends to display the labels.

Reused this code for displaying the legends:

http://stackoverflow.com/questions/23577505/how-to-avoid-overlapping-of-labels-autopct-in-a-matplotlib-pie-chart

* Call the show method to display the canvas.



Of all the storm types Thunderstorm is more with 25%, followed by Hail 16.3% and flashflood with 8.7%

12. Visualize and create a scatter plot of month of the year against the storm episode.

Dataset : stormtypedamage.csv

Script: scatterplotpy.py

We used the yearmonth and episode\_id columns from the StormEvents1\_details-2015.csv and created stormtypedamage.csv file.

Here is the scatterplot script

import matplotlib.pyplot as plt

import pandas as pd

#Load the csv, specify the columns to load, skip the header and create the Dataframe using pandas

data = pd.read\_csv('stormtypedamage.csv', sep=',', names=['YEARMONTH','EPISODE\_ID'], skiprows=1, index\_col=None)

col1 = data[data.columns[0]]

col2 = data[data.columns[1]]

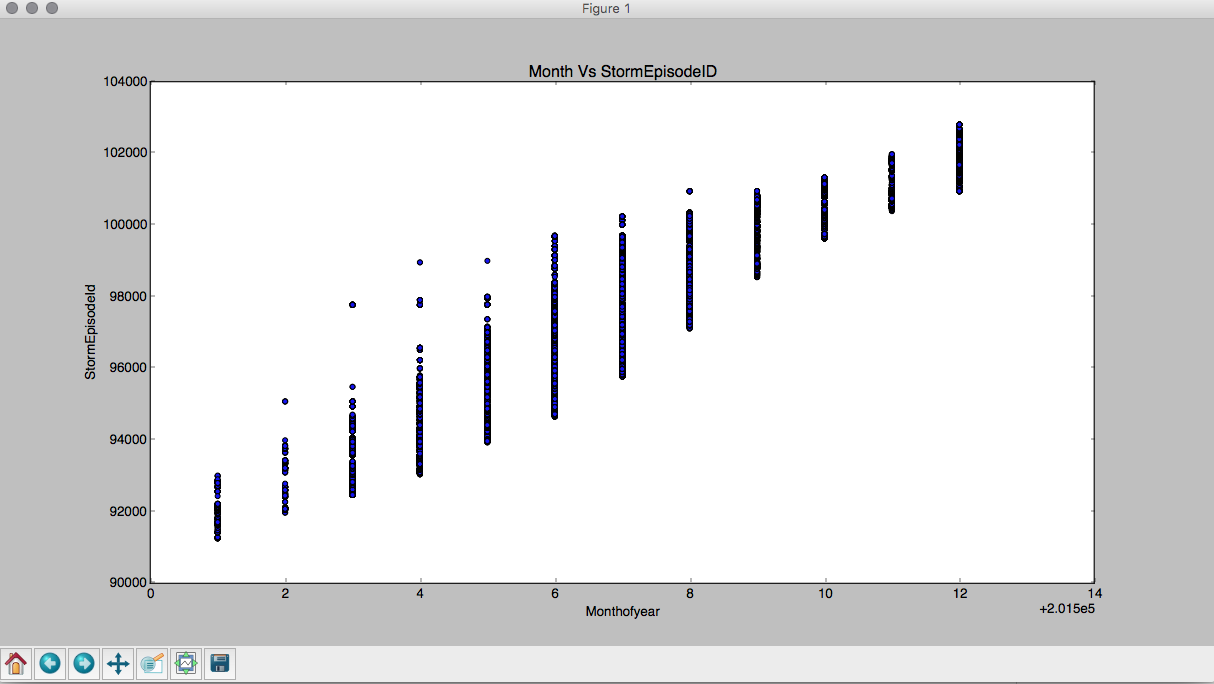
plt.scatter(col1, col2)

plt.ylabel('StormEpisodeId')

plt.xlabel('Monthofyear')

plt.title('Month Vs StormEpisodeID')

plt.show()



CONCLUSION:

This project gives a detailed analysis of the storm events occurred in 2015. We did the case study using SPARK datafarames and SparkSQL and visualized the data using python apis.

Benefits of the study:

* Using Spark we achieve great performance and fast computations. As the intermediate results are in memory it is great for querying the same data multiple times. Spark is 100x faster than Mapreduce.
* Using the study we analyzed and identified the total storm events occurred across all states in U.S, how much damage it caused to the property.
* We also analyzed about how many deaths and injuries occurred because of them.
* Which states it affected badly and during which months they were severe.

Challenges or Future Improvements:

* The study can be improvised by analyzing the last 10 year or 5 year data and we can create a 5 years /10 years trend analysis. This will give a broader picture of weather information and analysis.
* This project used Cloudera VM and had limitations for memory.

You Tube URLs:

15minutes you tube video: https://youtu.be/TPEfgi-Dpts

2minutes you tube video: https://youtu.be/beH3qgULVXY