Application Challenge Documentation

1. Counting Vitae

Tools Used:

Coding Lanaguage: Python

Operating System: Windows 10

Coding Pad: iPython Notebook (Jupyter)

Solution:

I first approached this problem by copying all the contents from my resume to a notepad (.txt format). Then I decided to count the frequency of distinct characters, numbers from my text document using Python functional programming. I plotted the graph using matplotlib.pyplot library. The code that I have written to calculate the character count and plot the corresponding graph based on the count are given below:

Python Code:

```
import matplotlib.pyplot as pp
resume = open('Manikandan_Ganesh_Resume_PlainText.txt', 'r')
wordlist = resume.readlines()
character_frequency = {}
i=0
for line in wordlist:
    i=0
    while(i<len(line)):
        if(line[i:i+1] not in (':, \x95\x96\n/-\t@().|')):
                if(line[i:i+1] not in character_frequency):
                    character_frequency[line[i:i+1]] = 0
                character_frequency[line[i:i+1]] += 1
            i+=1
    y_axis = []
    x_axis = []
    x_axis_labels = character_frequency.keys()
    x_axis_labels.sort()
    x_axis_value = 1
    for key in x_axis_labels:
        y_axis.append(character_frequency[key])
        x_axis.append(x_axis_value)
        x_axis_value+=1
    pp.bar(x_axis,y_axis,align='center', color='blue')
    pp.xticks(x_axis,x_axis_labels)
    pp.show()
```

Result:

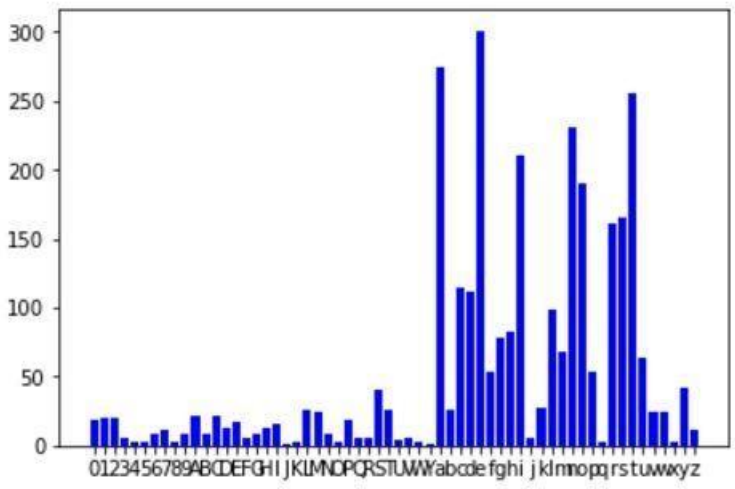

Figure 1: Character Count from plain text resume

2. Disaster Planning

Tools Used:

Scripting Language: Bash, MySQL scripts

Datastore Used: MySQL

Operating System: trusty64 Linux box using vagrant, Windows10

Coding Pad: Linux Command Line

Data Visualization (Graphs/Charts): Microsoft Excel

Solution (1):

As given in the problem statement, I took the following URL Link (https://www1.ncdc.noaa.gov/pub/data/swdi/stormevents/csvfiles/) as a single shell parameter and stored all the .gzip links in an array (targets). Then, I used **wget** command inside the shell script to download all the gzip files and extract csv's from the gzip using **gunzip** command.

After successfully obtaining the csv files, the problem statement required to combine all the Storm Event Details into a single csv. I could witness two types of csv files in the given set. They were:

1. StormEvents_details file and
2. StormEvents_fatalities file

So, I decided to combine all the files started with the name StormEvents_details into a single file. One more complexity was that every csv file had the same header statement including the names of all the columns. So, my approach was to include the header statement from the first file alone and neglect the

header statement from the rest of the files. So, I used **head** and **tail** commands to include the header and content in the shell script.

Once I created the csv, my next step was dumping the contents of the csv into a MySQL table. The vital steps that I took into consideration were:

- Understanding the schema and create the same number of columns in my table as present in the csv file.
- Ignore the first line (header statement) from the csv file while dumping the contents into the MySQL data store.
- Maintaining the **data consistency** by using the right data types for every column.
- Creating an auto_incremented primary key column as the final column in my table.

Taking these factors into consideration, I could successfully dumped the csv contents into the MySQL table.

The commands that I wrote in my shell script to complete the first step in the problem statement are given below.

Shell Script Commands:

```
#!/bin/bash
targets=($(wget -O - https://www1.ncdc.noaa.gov/pub/data/swdi/stormevents/csvfiles/ |  grep -o '<a
href=["'"'"'][^"'"'"']*["'"'"']' | sed -e 's/^<a href=["'"'"']//' -e 's/["'"'"']$//' | grep 'csv.gz'))
for i in "${targets[@]}"
do
      wget 'https://www1.ncdc.noaa.gov/pub/data/swdi/stormevents/csvfiles/'$i
done
gunzip *.gz
head -1 ~/pythonpractice/StormEvents_details-ftp_v1.0_d1950_c20170120.csv > StormEvents.csv
tail -n +2 -q StormEvents_details*.csv >> StormEvents.csv
mysql -u root -p -e "DROP DATABASE IF EXISTS STORMEVENTS;
CREATE DATABASE STORMEVENTS;
USE STORMEVENTS;
DROP TABLE IF EXISTS Event_Records
CREATE TABLE Event_Records(
Begin_YearMonth int,
Begin_Day int,
Begin_Time int,
End_YearMonth int,
End_Day int,
End_Time int,
Episode_Id int,
Event_ID int,
State varchar(30),
State_Fips int,
Year int,
Month_Name varchar(20),
Event_Type varchar(50),
CZ_Type varchar(10),
CZ_Fips int,
```

```
CZ_Name varchar(50), WFO
varchar(50), Begin_Date_Time
varchar(50), CZ_TimeZone
varchar(10), End_Date_Time
varchar(50), Injuries_Direct
varchar (20), Injuries_Indirect
varchar(20), Deaths_Direct
varchar(20), Deaths_Indirect
varchar(20), Damage_Property
varchar(20), Damage_Crops
varchar(20), Source
varchar(20),
Magnitude varchar(10),
Magnitude_Type varchar(20),
Flood_Cause varchar(50),
Category varchar(50),
Tor_F_Scale varchar(10),
Tor_Length varchar(20),
Tor_Width varchar(20),
Tor_Other_Wfo varchar(20),
Tor_Other_CZ_State varchar(20),
Tor_Other_CZ_Fips varchar(20),
Top_Other_CZ_Name varchar(20),
Begin_Range varchar(20),
Begin_Azimuth varchar(20),
Begin_Location varchar(20),
End_Range varchar(20),
End_Azimuth varchar(20),
End_Location varchar(20),
Begin_Lat varchar(50),
Begin_Lon varchar(50),
End_Lat varchar(50),
End_Lon varchar(50),
Episode_Narrative varchar(250),
Event_Narrative varchar(250),
Data_Source varchar(20),
Id int not null auto_increment
PRIMARY KEY(Id));

LOAD DATA INFILE "~/pythonpractice/StormEvents.csv
INTO TABLE STORMEVENTS.Event_Records
COLUMNS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```

Results:

```
mysql> use STORMEVENTS;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> DESCRIBE Event_Records;
+--------------------+--------------+------+-----+---------+----------------+
| Field              | Type         | Null | Key | Default | Extra          |
+--------------------+--------------+------+-----+---------+----------------+
| Begin_YearMonth    | int(11)      | YES  |     | NULL    |                |
| Begin_Day          | int(11)      | YES  |     | NULL    |                |
| Begin_Time         | int(11)      | YES  |     | NULL    |                |
| End_YearMonth      | int(11)      | YES  |     | NULL    |                |
| End_Day            | int(11)      | YES  |     | NULL    |                |
| End_Time           | int(11)      | YES  |     | NULL    |                |
| Episode_Id         | int(11)      | YES  |     | NULL    |                |
| Event_ID           | int(11)      | YES  |     | NULL    |                |
| State              | varchar(30)  | YES  |     | NULL    |                |
| State_Fips         | int(11)      | YES  |     | NULL    |                |
| Year               | int(11)      | YES  |     | NULL    |                |
| Month_Name         | varchar(20)  | YES  |     | NULL    |                |
| Event_Type         | varchar(50)  | YES  |     | NULL    |                |
| CZ_Type            | varchar(10)  | YES  |     | NULL    |                |
| CZ_Fips            | int(11)      | YES  |     | NULL    |                |
| CZ_Name            | varchar(50)  | YES  |     | NULL    |                |
| WFO                | varchar(50)  | YES  |     | NULL    |                |
| Begin_Date_Time    | varchar(50)  | YES  |     | NULL    |                |
| CZ_TimeZone        | varchar(10)  | YES  |     | NULL    |                |
| End_Date_Time      | varchar(50)  | YES  |     | NULL    |                |
| Injuries_Direct    | varchar(20)  | YES  |     | NULL    |                |
| Injuries_Indirect  | varchar(20)  | YES  |     | NULL    |                |
| Deaths_Direct      | varchar(20)  | YES  |     | NULL    |                |
| Deaths_Indirect    | varchar(20)  | YES  |     | NULL    |                |
| Damage_Property    | varchar(20)  | YES  |     | NULL    |                |
| Damage_Crops       | varchar(20)  | YES  |     | NULL    |                |
| Source             | varchar(20)  | YES  |     | NULL    |                |
| Magnitude          | varchar(10)  | YES  |     | NULL    |                |
| Magnitude_Type     | varchar(20)  | YES  |     | NULL    |                |
| Flood_Cause        | varchar(50)  | YES  |     | NULL    |                |
| Category           | varchar(50)  | YES  |     | NULL    |                |
| Tor_F_Scale        | varchar(10)  | YES  |     | NULL    |                |
| Tor_Length         | varchar(20)  | YES  |     | NULL    |                |
| Tor_Width          | varchar(20)  | YES  |     | NULL    |                |
| Tor_Other_Wfo      | varchar(20)  | YES  |     | NULL    |                |
| Tor_Other_CZ_State | varchar(20)  | YES  |     | NULL    |                |
| Tor_Other_CZ_Fips  | varchar(20)  | YES  |     | NULL    |                |
| Top_Other_CZ_Name  | varchar(20)  | YES  |     | NULL    |                |
| Begin_Range        | varchar(20)  | YES  |     | NULL    |                |
| Begin_Azimuth      | varchar(20)  | YES  |     | NULL    |                |
| Begin_Location     | varchar(20)  | YES  |     | NULL    |                |
| End_Range          | varchar(20)  | YES  |     | NULL    |                |
| End_Azimuth        | varchar(20)  | YES  |     | NULL    |                |
| End_Location       | varchar(20)  | YES  |     | NULL    |                |
| Begin_Lat          | varchar(50)  | YES  |     | NULL    |                |
| Begin_Lon          | varchar(50)  | YES  |     | NULL    |                |
| End_Lat            | varchar(50)  | YES  |     | NULL    |                |
| End_Lon            | varchar(50)  | YES  |     | NULL    |                |
| Episode_Narrative  | varchar(250) | YES  |     | NULL    |                |
| Event_Narrative    | varchar(250) | YES  |     | NULL    |                |
| Data_Source        | varchar(20)  | YES  |     | NULL    |                |
| Id                 | int(11)      | NO   | PRI | NULL    | auto_increment |
+--------------------+--------------+------+-----+---------+----------------+
52 rows in set (0.00 sec)
```

Figure 2a: Table structure describing the schema and the data types used

```
mysql> select count(*)
    -> from Event_Records;
+----------+
| count(*) |
+----------+
|  1423556 |
+----------+
1 row in set (2.51 sec)
```

Figure 2b: Number of records present inside the table

Solution for Property Damage by Year:

To summarize the property damage by year and to store the result in a table, I used a simple approach of writing a SQL query to calculate the total property damage by year taking two columns – **Year** and **Damage_Property** into consideration. The following SQL queries helped me to obtain the result.

```
drop table if exists damage_byyear;
create table damage_byyear as (
select Year,
concat(sum(case
when Damage_Property like '%K' then Damage_Property*1000
when Damage_Property like '%M' then Damage_Property*1000000
end)/1000000, 'M') as Total_Damage_Property
```

from Event_Records
group by year);

Results:

YEAR	TOTAL PROPERTY DAMAGE (IN USD)
1950	34.48165M
1951	65.50599M
1952	94.30224M
1953	596.1047M
1954	85.80532M
1955	82.66063M
1956	116.91235M
1957	224.38889M
1958	128.99461M
1959	87.45304M
1960	118.22424M
1961	179.38873M
1962	67.20073M
1963	87.23058M
1964	165.99082M
1965	1762.46901M
1966	476.62936M
1967	577.01922M
1968	210.07991M
1969	113.27423M
1970	625.20598M
1971	202.12055M
1972	151.17257M
1973	2063.04156M
1974	1941.01778M
1975	826.72379M
1976	308.49719M
1977	342.02423M
1978	616.06816M
1979	946.24124M
1980	2152.81435M
1981	765.88831M
1982	1429.193M
1983	765.85559M
1984	2131.17139M
1985	1608.09721M
1986	1034.91234M
1987	407.83795M
1988	1124.21465M
1989	1760.05105M
1990	1560.56807M
1991	1152.29643M
1992	1406.6889M
1993	1605.51425M

1994	1660.23125M
1995	1748.6365M
1996	3625.98415M
1997	4410.75874M
1998	11590.18393M
1999	5720.83205M
2000	4121.40484M
2001	4996.79767M
2002	4100.86225M
2003	8214.50893M
2004	9350.15574M
2005	15743.85761M
2006	7072.29889M
2007	7357.27066M
2008	16106.82072M
2009	6773.37663M
2010	6253.84169M
2011	12815.22929M
2012	11292.58641M
2013	6817.38805M
2014	4411.72039M
2015	3446.67889M
2016	10749.34873M
2017	1476.76859M

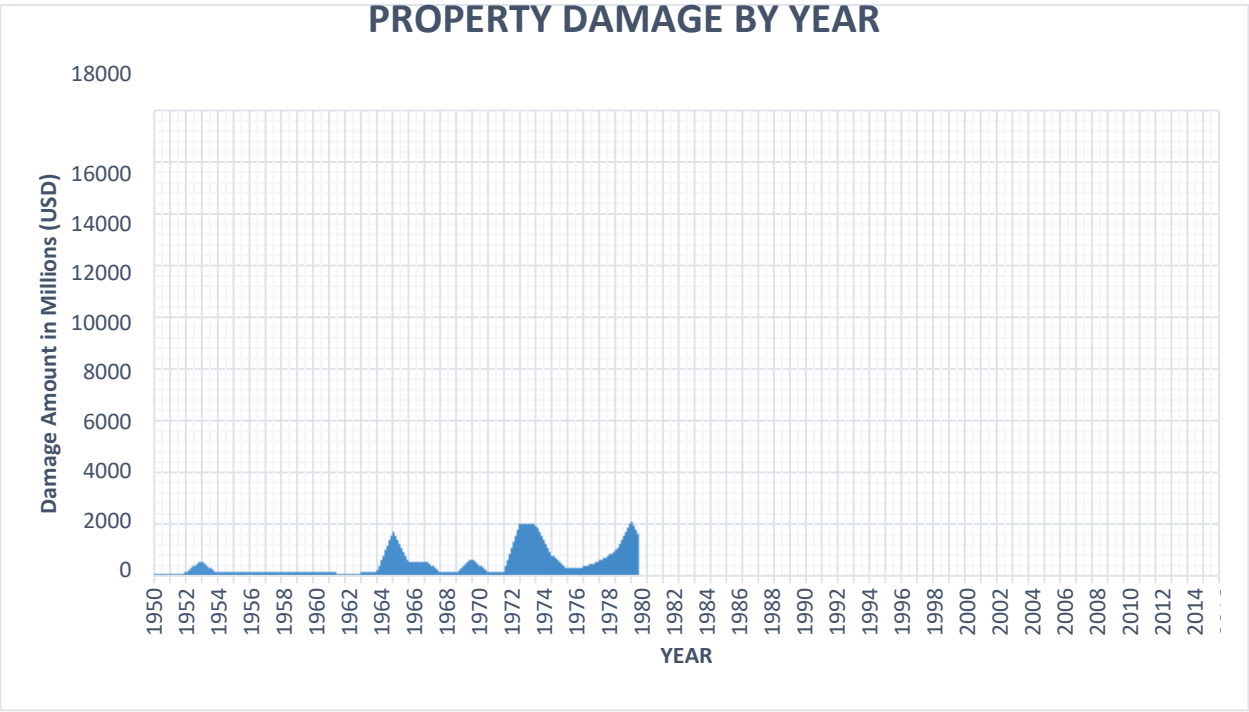

Figure 2c: Grid Chart describing the property damage caused every year

Summary (By Year):

To summarize my result findings for property damage every year, I can see that the year **2008** had the maximum damage and the year **1962** had the minimum damage. With respect to the data, the year 2008 had $16.1 billion damage as it marks the year when category 4 major hurricane **Ike** swept through portions of North America in September 2008.

(Source: https://en.wikipedia.org/wiki/Hurricane_Ike)

Solution for Property Damage by State:

The total property damage amount in every state is calculated using the same approach as I did for property damage by year, except I took the column **State** into consideration instead of **Year**. The SQL scripts and the results obtained are given below:

SQL Scripts for creating the table and dump values into it from Event_Records table:

```
drop table if exists damage_bystate;
create table damage_bystate as (
select State,
concat(sum(case
when Damage_Property like '%K' then Damage_Property*1000
when Damage_Property like '%M' then Damage_Property*1000000
end)/1000000, 'M') as Total_Damage_Property
from Event_Records
group by State
having State Not in (" "));
```

Results obtained:

STATE	TOTAL PROPERTY DAMAGE (IN USD)
ALABAMA	5026.40592M
ALASKA	362.6155M
AMERICAN SAMOA	209.255M
ARIZONA	2037.41688M
ARKANSAS	5291.00777M
ATLANTIC NORTH	0.316M
ATLANTIC SOUTH	5.7069M
CALIFORNIA	7336.1549M
COLORADO	4968.90229M
CONNECTICUT	770.12472M
DELAWARE	158.19907M
DISTRICT OF COLUMBIA	157.6981M
E PACIFIC	0.10201M
FLORIDA	12832.33646M
GEORGIA	5328.12966M
GUAM	970.823M
GULF OF ALASKA	0.031M
GULF OF MEXICO	19.19214M
HAWAII	193.75726M

HAWAII WATERS	0M
IDAHO	808.58511M
ILLINOIS	5419.95953M
INDIANA	4223.56389M
IOWA	5746.94753M
KANSAS	5071.86666M
KENTUCKY	3101.85668M
LAKE ERIE	0.03M
LAKE HURON	0M
LAKE MICHIGAN	2.8078M
LAKE ONTARIO	0.075M
LAKE ST CLAIR	0.1M
LAKE SUPERIOR	0.4M
LOUISIANA	14701.44454M
MAINE	534.4196M
MARYLAND	1245.43769M
MASSACHUSETTS	1134.34161M
MICHIGAN	3518.14005M
MINNESOTA	4999.00406M
MISSISSIPPI	12096.61529M
MISSOURI	5688.55143M
MONTANA	316.751M
NEBRASKA	3883.74003M
NEVADA	396.42966M
NEW HAMPSHIRE	252.40283M
NEW JERSEY	6294.70848M
NEW MEXICO	841.38069M
NEW YORK	4853.0268M
NORTH CAROLINA	5371.16751M
NORTH DAKOTA	2065.92307M
OHIO	7476.96674M
OKLAHOMA	6696.41582M
OREGON	739.43633M
PENNSYLVANIA	4561.05145M
PUERTO RICO	2221.58804M
RHODE ISLAND	129.7727M
SOUTH CAROLINA	1391.70181M
SOUTH DAKOTA	812.1215M
ST LAWRENCE R	0.015M
TENNESSEE	3147.17417M
TEXAS	25947.23355M
UTAH	873.21447M
VERMONT	1523.3073M
VIRGIN ISLANDS	39.0909M
VIRGINIA	2185.27231M
WASHINGTON	1250.10327M
WEST VIRGINIA	1232.31972M
WISCONSIN	3345.71878M
WYOMING	248.52041M

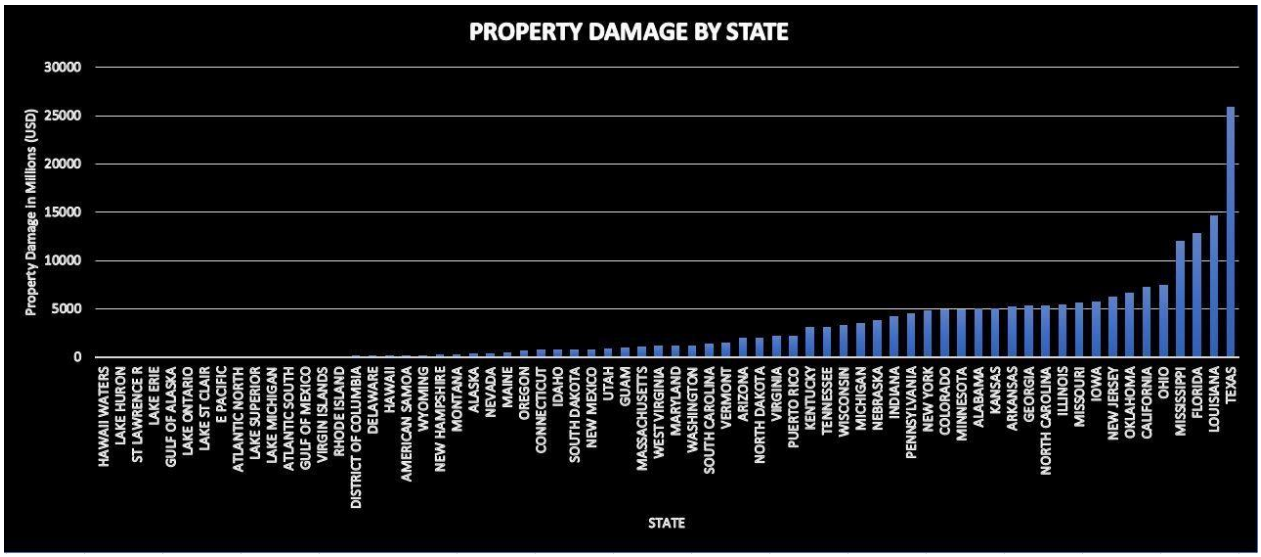

Figure 2d: Bar Chart displaying the property damage by state

Summary (By Year):

To summarize my result findings for property damage by State over the period of 67 years, I can see that Texas has the maximum damage till date with the total property damage amounting to over $2 billion. Louisiana has an equal amount of Property Damage amounting to over $1.4 billion. One of the prime reason for the huge amount of property damage in both Texas and Louisiana is hurricane **Ike** in 2008, which caused devastation from Louisiana coastline all the way till Corpus Christi in Texas. In addition, Ike also caused significant damages to **Mississippi** coastline and **Florida** panhandle.

(Source: https://en.wikipedia.org/wiki/Hurricane_Ike)

Optional Embellishments:

Static Visualization of result findings:

Tools Used: MySQL, Microsoft Excel

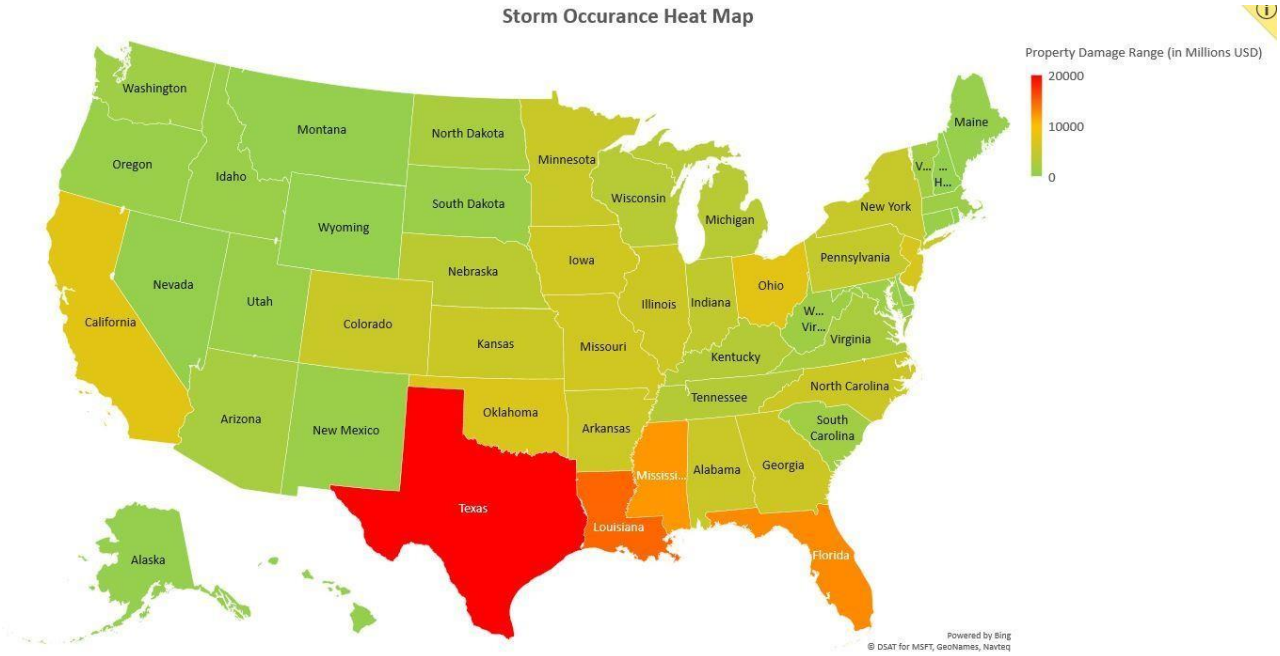

Figure 2e: Heat Map of property damage by state

My take on building a house to minimize property damage

If I want to consider a place in building a house this year to minimize property damage, I would choose **Colorado** and I have a reason for it.

I ran the following SQL script and I came to the decision of building a house in **Colorado**.

The following SQL script takes the top 10 values of the places that has minimum property damage for the past five years. I eliminated state values like **"HAWAII WATERS","LAKE HURON","ST LAWRENCE R","LAKE ERIE","GULF OF ALASKA","LAKE ONTARIO","LAKE ST CLAIR","E PACIFIC","ATLANTIC NORTH","LAKE SUPERIOR","LAKE MICHIGAN","ATLANTIC SOUTH","ALASKA","AMERICAN SAMOA"** where human survival is difficult due to cold weather conditions and other environment aspects. Once I get the top 10 values of states, I could notice **Colorado** having less property damage among all the states for the past five years. So, I identified **Colorado** as the place where I want to build a house.

```
select State, MAX(State_Count) AS Minimum_Storm_Occurence
from
(select State, count(State) as State_Count
from
((select Year, State, Total_Damage_Property
from
(select Year, State,
concat(sum(case
when Damage_Property like '%K' then Damage_Property*1000
when Damage_Property like '%M' then Damage_Property*1000000
end)/1000000, 'M') as Total_Damage_Property
from Event_Records
where Year IN (2013, 2014, 2015, 2016, 2017)
group by State, Year
having State Not in (" ","HAWAII WATERS","LAKE HURON","ST LAWRENCE R","LAKE ERIE","GULF OF
ALASKA","LAKE ONTARIO","LAKE ST CLAIR","E PACIFIC","ATLANTIC NORTH","LAKE SUPERIOR","LAKE
MICHIGAN","ATLANTIC SOUTH","ALASKA","AMERICAN SAMOA")) A
where Year = 2013
group by State
order by Total_Damage_Property
LIMIT 10)
UNION ALL
(select Year, State, Total_Damage_Property
from
(select Year, State,
concat(sum(case
when Damage_Property like '%K' then Damage_Property*1000
when Damage_Property like '%M' then Damage_Property*1000000
end)/1000000, 'M') as Total_Damage_Property
from Event_Records
where Year IN (2013, 2014, 2015, 2016, 2017)
group by State, Year
having State Not in (" ","HAWAII WATERS","LAKE HURON","ST LAWRENCE R","LAKE ERIE","GULF OF
ALASKA","LAKE ONTARIO","LAKE ST CLAIR","E PACIFIC","ATLANTIC NORTH","LAKE SUPERIOR","LAKE
MICHIGAN","ATLANTIC SOUTH","ALASKA","AMERICAN SAMOA")) A
where Year = 2014
group by State
order by Total_Damage_Property
LIMIT 10)
UNION ALL
(select Year, State, Total_Damage_Property
from
(select Year, State,
concat(sum(case
when Damage_Property like '%K' then Damage_Property*1000
when Damage_Property like '%M' then Damage_Property*1000000
end)/1000000, 'M') as Total_Damage_Property
```

```
from Event_Records
where Year IN (2013, 2014, 2015, 2016, 2017)
group by State, Year
having State Not in (" ","HAWAII WATERS","LAKE HURON","ST LAWRENCE R","LAKE ERIE","GULF OF
ALASKA","LAKE ONTARIO","LAKE ST CLAIR","E PACIFIC","ATLANTIC NORTH","LAKE SUPERIOR","LAKE
MICHIGAN","ATLANTIC SOUTH","ALASKA","AMERICAN SAMOA")) A
where Year = 2015
group by State
order by Total_Damage_Property
LIMIT 10)
UNION ALL
(select Year, State, Total_Damage_Property
from
(select Year, State,
concat(sum(case
when Damage_Property like '%K' then Damage_Property*1000
when Damage_Property like '%M' then Damage_Property*1000000
end)/1000000, 'M') as Total_Damage_Property
from Event_Records
where Year IN (2013, 2014, 2015, 2016, 2017)
group by State, Year
having State Not in (" ","HAWAII WATERS","LAKE HURON","ST LAWRENCE R","LAKE ERIE","GULF OF
ALASKA","LAKE ONTARIO","LAKE ST CLAIR","E PACIFIC","ATLANTIC NORTH","LAKE SUPERIOR","LAKE
MICHIGAN","ATLANTIC SOUTH","ALASKA","AMERICAN SAMOA")) A
where Year = 2016
group by State
order by Total_Damage_Property
LIMIT 10)
UNION ALL
(select Year, State, Total_Damage_Property
from
(select Year, State,
concat(sum(case
when Damage_Property like '%K' then Damage_Property*1000
when Damage_Property like '%M' then Damage_Property*1000000
end)/1000000, 'M') as Total_Damage_Property
from Event_Records
where Year IN (2013, 2014, 2015, 2016, 2017)
group by State, Year
having State Not in (" ","HAWAII WATERS","LAKE HURON","ST LAWRENCE R","LAKE ERIE","GULF OF
ALASKA","LAKE ONTARIO","LAKE ST CLAIR","E PACIFIC","ATLANTIC NORTH","LAKE SUPERIOR","LAKE
MICHIGAN","ATLANTIC SOUTH","ALASKA","AMERICAN SAMOA")) A
where Year = 2017
group by State
order by Total_Damage_Property
LIMIT 10)) B
group by State) C;
```

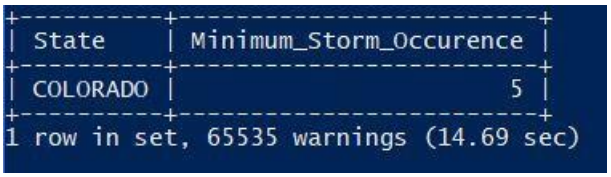

Figure 2f: Minimum Storm Occurrence result

Links of my other works:

- My current project work is framing custom ETL operations for integrating attendance records from three different data sets (with varying degrees of success) into a single, clean, highly accurate and flexible system. The project work can be accessed through this link - https://github.com/manikandanganesh2791/NewAttendanceSystem
- I like writing about new technologies that I learn. My write up on Cassandra data store can viewed through this link - https://github.com/manikandanganesh2791/CassandraProjects/blob/master/Cassandra.pdf
- My understanding of Hadoop MapReduce, Sqoop and Hive can be seen through my work on processing weblog files on HDFS platform. The project work can be seen through this link - https://github.com/manikandanganesh2791/Hadoop-Tools/