Project A – Working with National Weather Service Data

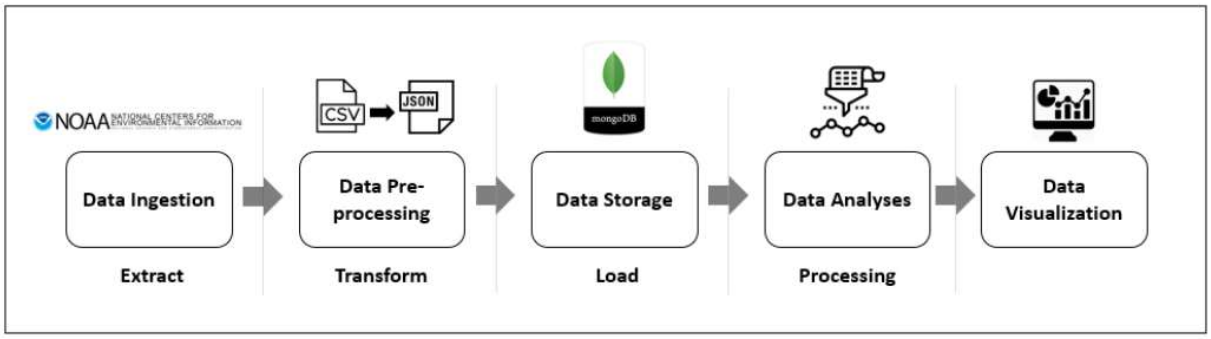
Introduction:

In this project, I worked with data from NOAA's National Weather Service (NWS) and trying to answer questions on storm events that are predominant in different states and reason for it. As mentioned in the assignment, the data was taken from the Storm Events Database (Links to an external site.), which contained data about the weather-related events in US from January 1950 to May 2019. I took data from 2010 to 2018 for my project work.

Data pipeline followed in this project

A data pipeline provides organized access to reliable and well-structured datasets for analytics. Automating the movement and transformation of **data** allows the consolidation of **data** from multiple sources so that it **can** be used strategically.

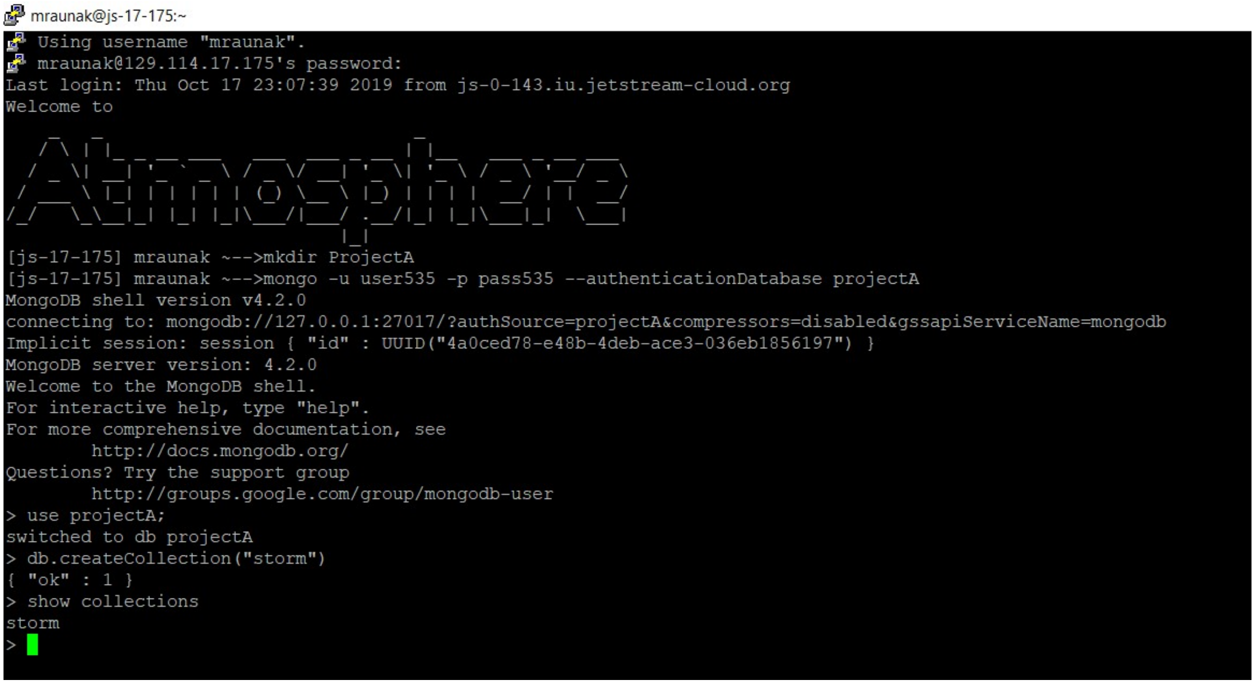
The diagram below provides an overview of your data pipeline:



Starting with the first step, which is creating a database to store, access and manipulate the data.

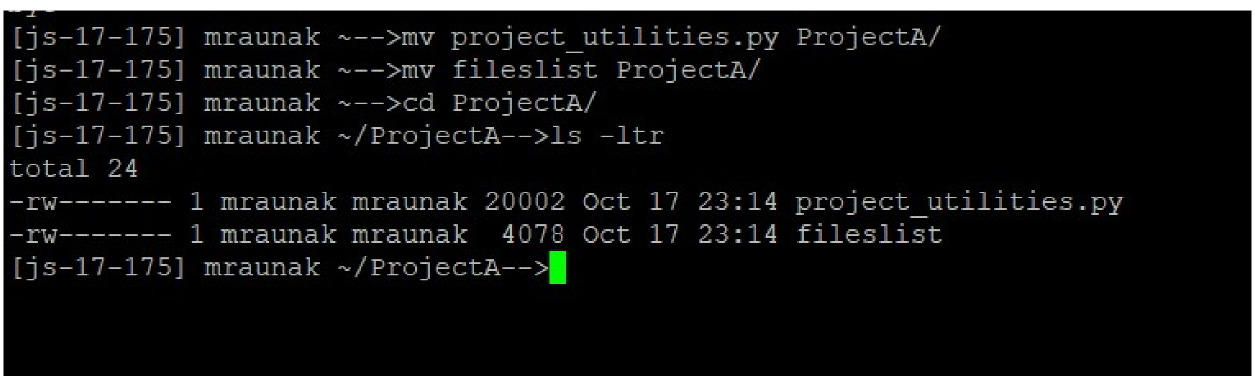
Database used for working is MongoDB which belongs to the family of NoSQL and it is a document-oriented database model which supports various forms of data. Created a Collection name ‘ storm’ using MongoDB. Data is stored in MongoDB in JSON format. It is easy to scale, enables faster access to data and it is widely used in Big Data.

Below is the screenshot of the creation of collection’ storm’ and confirmed that the collection has been created



```
mraunak@js-17-175:~
Using username "mraunak".
mraunak@129.114.17.175's password:
Last login: Thu Oct 17 23:07:39 2019 from js-0-143.iu.jetstream-cloud.org
Welcome to
[js-17-175] mraunak ~-->mkdir ProjectA
[js-17-175] mraunak ~-->mongo -u user535 -p pass535 --authenticationDatabase projectA
MongoDB shell version v4.2.0
connecting to: mongodb://127.0.0.1:27017/?authSource=projectA&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("4a0ced78-e48b-4deb-ace3-036eb1856197") }
MongoDB server version: 4.2.0
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
> use projectA;
switched to db projectA
> db.createCollection("storm")
{ "ok" : 1 }
> show collections
storm
>
```

Created a project directory and copied files to the project directory



```
[js-17-175] mraunak ~-->mv project_utilities.py ProjectA/
[js-17-175] mraunak ~-->mv fileslist ProjectA/
[js-17-175] mraunak ~-->cd ProjectA/
[js-17-175] mraunak ~/ProjectA-->ls -ltr
total 24
-rw------- 1 mraunak mraunak 20002 Oct 17 23:14 project_utilities.py
-rw------- 1 mraunak mraunak  4078 Oct 17 23:14 fileslist
[js-17-175] mraunak ~/ProjectA-->
```

DATA INGESTION—

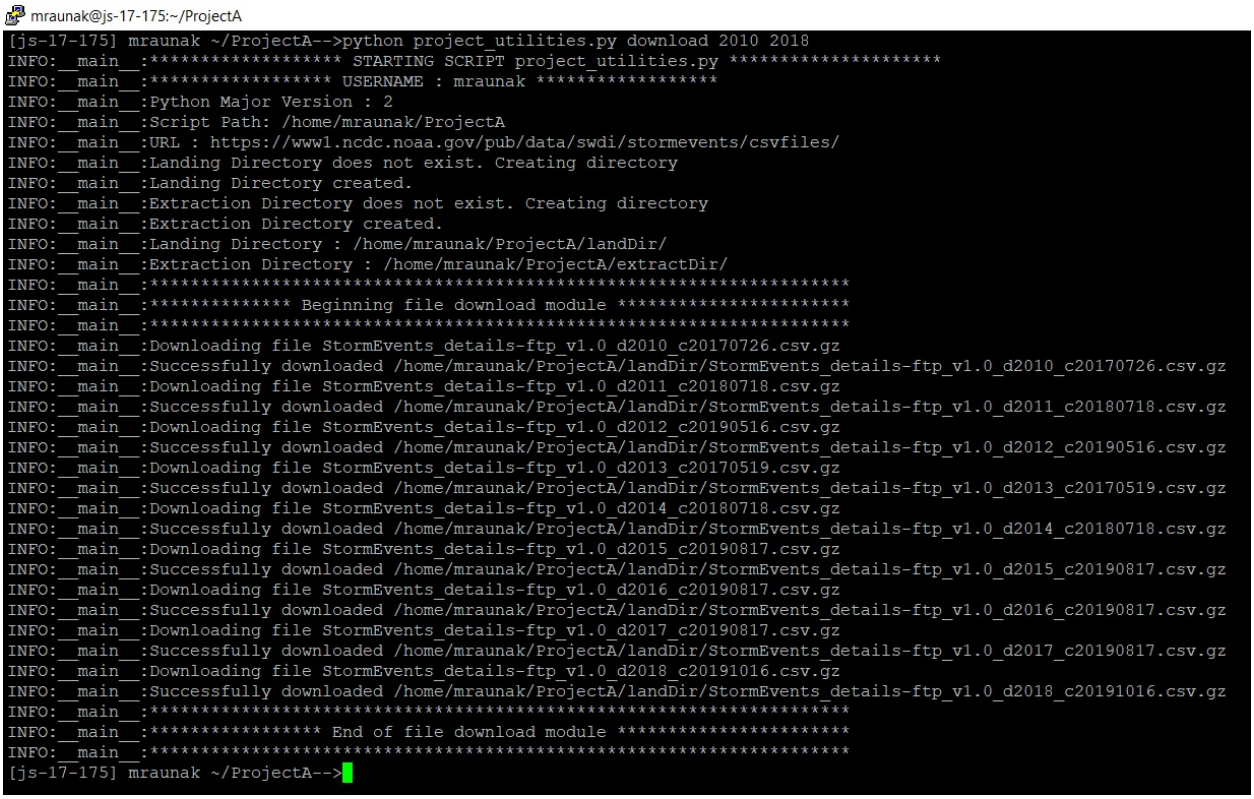
Data ingestion is the process of obtaining and importing data for immediate use or storage in a database. This is the first stage of the

Website: https://www1.ncdc.noaa.gov/pub/data/swdi/stormevents/csvfiles/ .

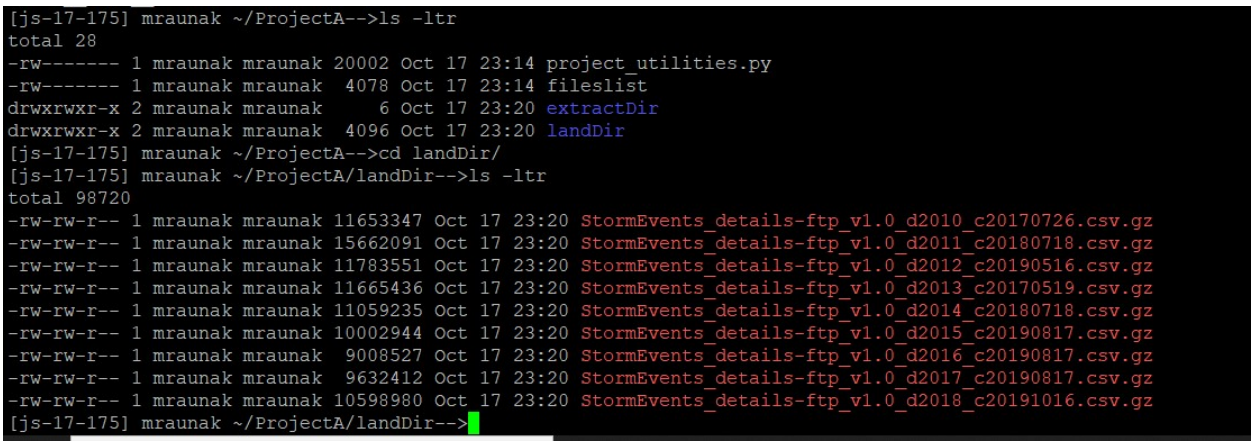
Imported and download the data from the above website   This data was downloaded to a landing directory in the VM. The downloaded data is in GZ compressed format.

For my analysis, I have used the data from the year 2010 to year 2018.

Downloaded the data



Below is screenshot of the downloaded files and extracted files. Here the files were in ‘gz compressed’ format as you can see in the above screenshot, file extension being(csv.gz). Extracted the files as csv files for further working.



DATA PREPROCESSING

Data preprocessing is an important and often required component in data analytics. Data preprocessing becomes even more important when consuming unstructured text data generated from multiple different sources. Data preprocessing steps include operations such as cleaning the data, extracting important features from data, removing duplicate items from the datasets, converting data formats.

Transform: To store the data in MongoDB, we need to convert the files from CSV to JSON format. For this the Pandas library was used , which allowed us to read the files and select the columns we need.

EXTRACT

mraunak@js-17-175:~/ProjectA

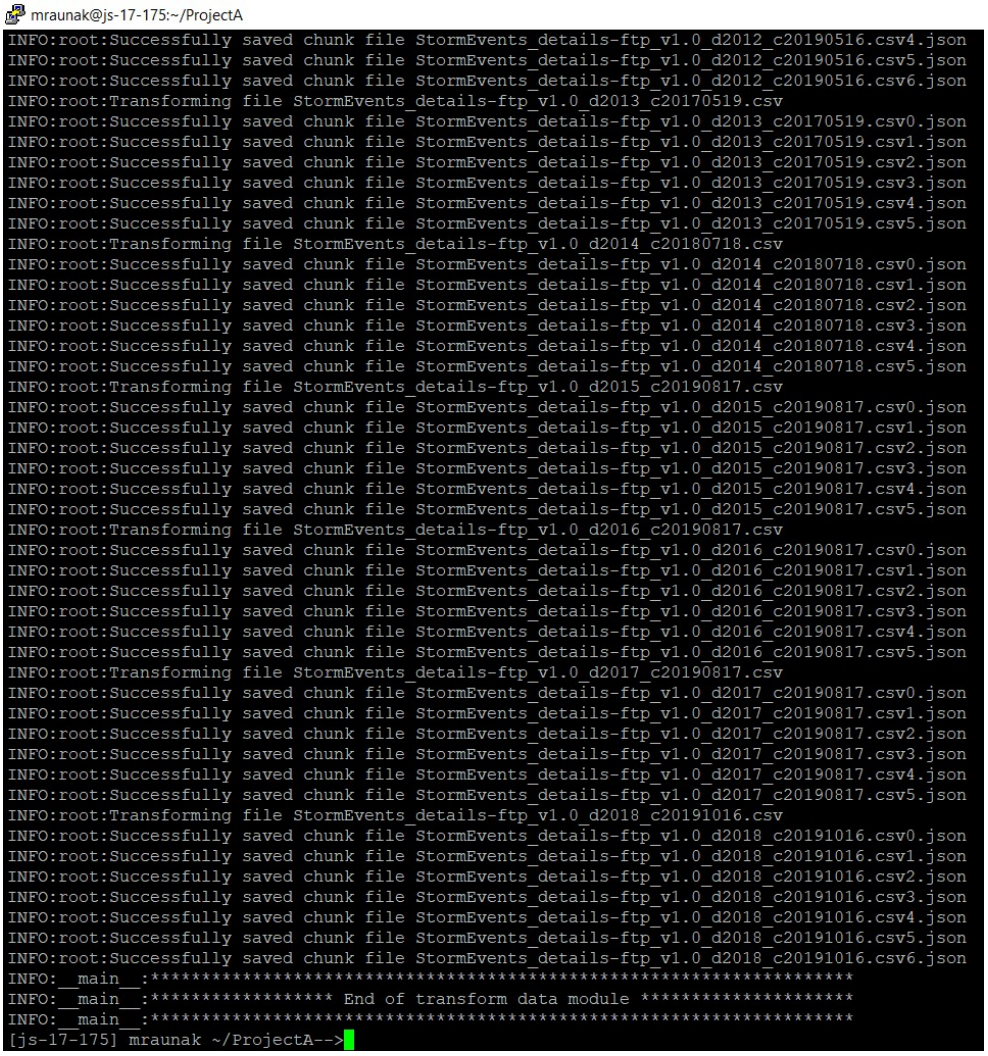
```
[js-17-175] mraunak ~/ProjectA-->python project_utilities.py extract
INFO:__main__:****************** STARTING SCRIPT project_utilities.py *********************
INFO:__main__:****************** USERNAME : mraunak ******************
INFO:__main__:Python Major Version : 2
INFO:__main__:Script Path: /home/mraunak/ProjectA
INFO:__main__:URL : https://www1.ncdc.noaa.gov/pub/data/swdi/stormevents/csvfiles/
INFO:__main__:Landing Directory : /home/mraunak/ProjectA/landDir/
INFO:__main__:Extraction Directory : /home/mraunak/ProjectA/extractDir/
INFO:__main__:**********************************************************************
INFO:__main__:*************** Beginning file extract module ***********************
INFO:__main__:**********************************************************************
INFO:__main__:Extracting file /home/mraunak/ProjectA/landDir/StormEvents_details-ftp_v1.0_d2010_c20170726.csv.gz
INFO:__main__:Successfully extracted file StormEvents_details-ftp_v1.0_d2010_c20170726.csv.gz
INFO:__main__:Extracting file /home/mraunak/ProjectA/landDir/StormEvents_details-ftp_v1.0_d2011_c20180718.csv.gz
INFO:__main__:Successfully extracted file StormEvents_details-ftp_v1.0_d2011_c20180718.csv.gz
INFO:__main__:Extracting file /home/mraunak/ProjectA/landDir/StormEvents_details-ftp_v1.0_d2012_c20190516.csv.gz
INFO:__main__:Successfully extracted file StormEvents_details-ftp_v1.0_d2012_c20190516.csv.gz
INFO:__main__:Extracting file /home/mraunak/ProjectA/landDir/StormEvents_details-ftp_v1.0_d2013_c20170519.csv.gz
INFO:__main__:Successfully extracted file StormEvents_details-ftp_v1.0_d2013_c20170519.csv.gz
INFO:__main__:Extracting file /home/mraunak/ProjectA/landDir/StormEvents_details-ftp_v1.0_d2014_c20180718.csv.gz
INFO:__main__:Successfully extracted file StormEvents_details-ftp_v1.0_d2014_c20180718.csv.gz
INFO:__main__:Extracting file /home/mraunak/ProjectA/landDir/StormEvents_details-ftp_v1.0_d2015_c20190817.csv.gz
INFO:__main__:Successfully extracted file StormEvents_details-ftp_v1.0_d2015_c20190817.csv.gz
INFO:__main__:Extracting file /home/mraunak/ProjectA/landDir/StormEvents_details-ftp_v1.0_d2016_c20190817.csv.gz
INFO:__main__:Successfully extracted file StormEvents_details-ftp_v1.0_d2016_c20190817.csv.gz
INFO:__main__:Extracting file /home/mraunak/ProjectA/landDir/StormEvents_details-ftp_v1.0_d2017_c20190817.csv.gz
INFO:__main__:Successfully extracted file StormEvents_details-ftp_v1.0_d2017_c20190817.csv.gz
INFO:__main__:Extracting file /home/mraunak/ProjectA/landDir/StormEvents_details-ftp_v1.0_d2018_c20191016.csv.gz
INFO:__main__:Successfully extracted file StormEvents_details-ftp_v1.0_d2018_c20191016.csv.gz
INFO:__main__:**********************************************************************
INFO:__main__:****************** End of file extract module ***********************
INFO:__main__:**********************************************************************
[js-17-175] mraunak ~/ProjectA-->
```

Extracted Files

```
[js-17-175] mraunak ~/ProjectA-->cd extractDir/
[js-17-175] mraunak ~/ProjectA/extractDir-->ls -ltr
total 532568
-rw-rw-r-- 1 mraunak mraunak 62205183 Oct 17 23:34 StormEvents_details-ftp_v1.0_d2010_c20170726.csv
-rw-rw-r-- 1 mraunak mraunak 83899123 Oct 17 23:34 StormEvents_details-ftp_v1.0_d2011_c20180718.csv
-rw-rw-r-- 1 mraunak mraunak 64921112 Oct 17 23:34 StormEvents_details-ftp_v1.0_d2012_c20190516.csv
-rw-rw-r-- 1 mraunak mraunak 61782395 Oct 17 23:34 StormEvents_details-ftp_v1.0_d2013_c20170519.csv
-rw-rw-r-- 1 mraunak mraunak 58639893 Oct 17 23:34 StormEvents_details-ftp_v1.0_d2014_c20180718.csv
-rw-rw-r-- 1 mraunak mraunak 54110624 Oct 17 23:34 StormEvents_details-ftp_v1.0_d2015_c20190817.csv
-rw-rw-r-- 1 mraunak mraunak 51014265 Oct 17 23:34 StormEvents_details-ftp_v1.0_d2016_c20190817.csv
-rw-rw-r-- 1 mraunak mraunak 51886043 Oct 17 23:34 StormEvents_details-ftp_v1.0_d2017_c20190817.csv
-rw-rw-r-- 1 mraunak mraunak 56875407 Oct 17 23:34 StormEvents_details-ftp_v1.0_d2018_c20191016.csv
[js-17-175] mraunak ~/ProjectA/extractDir-->
```

TRANSFORM (part of data preprocessing)

Selected the relevant columns from the data files and converted the file type to json. This can be seen in screenshot below



DATA DTORAGE---

Load/Data Storage: Used PyMongo to work with MongoDB. It is a Python distribution that contains tools for working with and is the recommended way to work with MongoDB when you rely on Python . Once the data files are converted to JSON with required columns, we stored them in MongoDB.

We loaded file in MongoDB data base in json. MongoDB stores data in json format, hence converting into json format.

```
INFO:__main__:MongoDB Username : user535
INFO:__main__:MongoDB Hostname : localhost
INFO:__main__:MongoDB Port : 27017
INFO:__main__:MongoDB Database : projectA
INFO:__main__:MongoDB Collection : storm
INFO:__main__:Creating MongoDB connection.
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2010_c20170726.csv0.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2010_c20170726.csv0.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2010_c20170726.csv1.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2010_c20170726.csv1.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2010_c20170726.csv2.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2010_c20170726.csv2.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2010_c20170726.csv3.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2010_c20170726.csv3.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2010_c20170726.csv4.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2010_c20170726.csv4.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2010_c20170726.csv5.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2010_c20170726.csv5.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2010_c20170726.csv6.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2010_c20170726.csv6.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2011_c20180718.csv0.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2011_c20180718.csv0.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2011_c20180718.csv1.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2011_c20180718.csv1.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2011_c20180718.csv2.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2011_c20180718.csv2.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2011_c20180718.csv3.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2011_c20180718.csv3.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2011_c20180718.csv4.json into MongoDB
```

```
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2016_c20190817.csv5.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2017_c20190817.csv0.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2017_c20190817.csv0.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2017_c20190817.csv1.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2017_c20190817.csv1.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2017_c20190817.csv2.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2017_c20190817.csv2.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2017_c20190817.csv3.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2017_c20190817.csv3.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2017_c20190817.csv4.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2017_c20190817.csv4.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2017_c20190817.csv5.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2017_c20190817.csv5.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2018_c20191016.csv0.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2018_c20191016.csv0.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2018_c20191016.csv1.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2018_c20191016.csv1.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2018_c20191016.csv2.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2018_c20191016.csv2.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2018_c20191016.csv3.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2018_c20191016.csv3.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2018_c20191016.csv4.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2018_c20191016.csv4.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2018_c20191016.csv5.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2018_c20191016.csv5.json into MongoDB
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2018_c20191016.csv6.json into MongoDB
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2018_c20191016.csv6.json into MongoDB
INFO:__main__:TOTAL NUMBER OF RECORDS LOADED IN MONGODB : 559014
INFO:__main__:************************************************************************
INFO:__main__:********************* End of load data module ************************
INFO:__main__:************************************************************************
[js-17-175] mraunak ~/ProjectA-->
```

Verified that, all records have been loaded to mongo dB as the number of records loaded are equal (559014) in both the places.

```
[js-17-175] mraunak ~/ProjectA-->mongo -u user535 -p pass535 --authenticationDatabase projectA
MongoDB shell version v4.2.0
connecting to: mongodb://127.0.0.1:27017/?authSource=projectA&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("df850357-33d8-4fe7-9473-7902c05cbcd5") }
MongoDB server version: 4.2.0
> use projectA;
switched to db projectA
> db.storm.find().count()
559014
>
```

Analytics

Data Analytics is the often complex process of examining large and varied data sets, or big data, and find hidden patterns, unknown correlations, that can help organizations make informed business decisions.

In this stage, basic operations were performed on data such as insert a new record, update a record, delete a record and some complex and informative queries were done on the data.

The results of the different queries help us to get the in depth knowledge about the data.

Pretty function used get the better format for the output.


```
> db.storm.find().pretty()
{
        "_id" : ObjectId("5da94903937de2c54cac181a"),
        "DEATHS_INDIRECT" : 0,
        "DEATHS_DIRECT" : 0,
        "INJURIES_INDIRECT" : 0,
        "INJURIES_DIRECT" : 0,
        "EVENT_TYPE" : "Heat",
        "SOURCE" : "AWOS",
        "EVENT_ID" : 254780,
        "DAMAGE_CROPS" : "0.00K",
        "MONTH_NAME" : "July",
        "EPISODE_ID" : 43850,
        "CZ_TYPE" : "Z",
        "STATE" : "NEW HAMPSHIRE",
        "END_DAY" : 7,
        "YEAR" : 2010,
        "EVENT_NARRATIVE" : "Heat index values at the Nashua Boire Field (KASH) Automated Weather Observing System were 100 to 104 degrees.",
        "EPISODE_NARRATIVE" : "A strong ridge built into Southern New England resulting in temperatures nearing 100 with high humidity. Heat index values ranged from 100 to 106 for most of
Southern New England on the 6th and again on the 7th in a more limited area, generally the Connecticut River Valley.",
        "DAMAGE_PROPERTY" : "0.00K",
        "BEGIN_DAY" : 7
}
{
        "_id" : ObjectId("5da94903937de2c54cac181b"),
        "DEATHS_INDIRECT" : 0,
        "DEATHS_DIRECT" : 0,
        "INJURIES_INDIRECT" : 0,
        "INJURIES_DIRECT" : 0,
        "EVENT_TYPE" : "Heavy Snow",
        "SOURCE" : "CoCoRaHS",
        "EVENT_ID" : 211550,
        "DAMAGE_CROPS" : "0.00K",
        "MONTH_NAME" : "January",
        "EPISODE_ID" : 36500,
        "CZ_TYPE" : "Z",
        "STATE" : "NEW HAMPSHIRE",
        "END_DAY" : 18,
        "YEAR" : 2010,
        "EVENT_NARRATIVE" : "Four to eight inches fell across eastern Hillsborough County.",
        "EPISODE_NARRATIVE" : "A coastal storm passing southern New England just southeast of Nantucket resulted in a period of snow across northern Massachusetts and southwest New Hampshir
e.",
        "DAMAGE_PROPERTY" : "0.00K",
        "BEGIN_DAY" : 17
}
{
        "_id" : ObjectId("5da94903937de2c54cac181c"),
        "DEATHS_INDIRECT" : 0,
```


Querying

I reduced the number of fields while uploading as it doesn't consider specific schema as we can see it was loaded successfully. It is a NoSQL database which is schema less

some operations were performed such as verification and modification of the data:

Query1: Insert a record into collection: Inserted a record using the db.collection.insert(<document or array of documents>) command

```
> db.storm.insert( {
... BEGIN_DAY: 20,
... END_DAY: 20,
... EPISODE_ID: 132265438,
... EVENT_ID:432432412,
... STATE: "Indiana",
... YEAR: 2029,
... MONTH_NAME: "April",
... EVENT_TYPE: "Storm",
... CZ_TYPE: "Z",
... INJURIES_DIRECT:0,
... INJURIES_INDIRECT:0,
... DEATHS_DIRECT:0,
... DEATHS_INDIRECT:0,
... DAMAGE_PROPERTY:"1K",
... SOURCE: "AWOS,ASOS,MESONET,ETC",
... MAGNITUDE: 60,
... BEGIN_LAT: 37.968777,
... BEGIN_LON: -87.549725,
... END_LAT : 39.084419,
... END_LON : -87.204043,
... EVENT_NARRATIVE: "A F2 Tornado touched down near open field and moved north eastward. "
... } )
WriteResult({ "nInserted" : 1 })
>
```

Used db.collection.find() command to view your inserted record (e.g., by specifying its EVENT_ID). Hence, the record was inserted successfully which can be seen in the below screenshot.

```
> db.storm.find({EVENT_ID:432432412}).pretty()
{
        "_id" : ObjectId("5da94c91684bcacdc47bdba4"),
        "BEGIN_DAY" : 20,
        "END_DAY" : 20,
        "EPISODE_ID" : 132265438,
        "EVENT_ID" : 432432412,
        "STATE" : "Indiana",
        "YEAR" : 2019,
        "MONTH_NAME" : "April",
        "EVENT_TYPE" : "Storm",
        "CZ_TYPE" : "Z",
        "INJURIES_DIRECT" : 0,
        "INJURIES_INDIRECT" : 0,
        "DEATHS_DIRECT" : 0,
        "DEATHS_INDIRECT" : 0,
        "DAMAGE_PROPERTY" : "1K",
        "SOURCE" : "AWOS,ASOS,MESONET,ETC",
        "MAGNITUDE" : 60,
        "BEGIN_LAT" : 37.968777,
        "BEGIN_LON" : -87.549725,
        "END_LAT" : 39.084419,
        "END_LON" : -87.204043,
        "EVENT_NARRATIVE" : "A F2 Tornado touched down near open field and moved north eastward. ",
        "EPISODE_NARRATIVE" : "A huge tornado was reported and sightings were confirmed."
}
>
```

Query2 :Update records in collection: The collection was updated with db.collection.update() command. The year was successfully updated from 2029 to 2019 and text was added to the EPISODE_NARRATIVE variable.

```
> db.storm.update({EVENT_ID:432432412, YEAR:2029},
... {
... $set: { YEAR : 2019, EPISODE_NARRATIVE : "A huge tornado was reported and sightings were confirmed." }
... }
... )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Query 3 : Delete records from collection: To remove a record, used db.collection.delete*() command. one can remove one or many records at once:

```
> db.storm.deleteOne( { EVENT_ID: 432432412 } );
{ "acknowledged" : true, "deletedCount" : 1 }
> db.storm.find({EVENT_ID:432432412}).pretty()
>
>
```

We can see that the EVENT_ID:432432412 has been deleted, as the command find couldn't find the mentioned EVENT_ID

Query 4 : Find the distinct records from collection: To the distinct records from collection, used db.collection.delete*() command db.storm.distinct("EVENT_TYPE")

```
> db.storm.distinct( "EVENT_TYPE" )
[
        "Heat",
        "Heavy Snow",
        "Strong Wind",
        "Winter Storm",
        "High Wind",
        "Flash Flood",
        "Flood",
        "Hail",
        "Thunderstorm Wind",
        "Marine Thunderstorm Wind",
        "Drought",
        "Excessive Heat",
        "Dense Fog",
        "Lake-Effect Snow",
        "Tornado",
        "Winter Weather",
        "Lightning",
        "Frost/Freeze",
        "Funnel Cloud",
        "Blizzard",
        "Ice Storm",
        "Heavy Rain",
        "Wildfire",
        "Cold/Wind Chill",
        "Extreme Cold/Wind Chill",
        "Waterspout",
        "Coastal Flood",
        "Rip Current",
        "Marine Strong Wind",
        "High Surf",
        "Marine Hail",
        "Avalanche",
        "Dust Storm",
```

Query 5 : Find top 10 events between 2010 and 2018

```
> db.storm.aggregate(
...     [
...        {
...          "$group" :
...                 {
...                    _id:"$EVENT_TYPE",
...                    count:{$sum:1}
...                 }
...        },
...        {
...          $sort:
...                 { count: -1 }
...        },
...        {
...          "$limit": 10
...        }
...     ])
{ "_id" : "Thunderstorm Wind", "count" : 142665 }
{ "_id" : "Hail", "count" : 98289 }
{ "_id" : "Winter Weather", "count" : 34496 }
{ "_id" : "Flash Flood", "count" : 34158 }
{ "_id" : "Drought", "count" : 30433 }
{ "_id" : "Winter Storm", "count" : 27769 }
{ "_id" : "High Wind", "count" : 27038 }
{ "_id" : "Flood", "count" : 23028 }
{ "_id" : "Heavy Snow", "count" : 20481 }
{ "_id" : "Marine Thunderstorm Wind", "count" : 15878 }
>
```

Some Complex Queries

Query 6 : Top 10 events that caused the greatest number of direct and indirect deaths for a specific period.

```
> db.storm.aggregate(
...
...    [
...
...      {
...
...        $group:
...
...          {
...
...            _id: "$EVENT_TYPE",
...
...            totalDeaths: { $sum: { $sum: [ "$DEATHS_DIRECT", "$DEATHS_INDIRECT" ] } }
...
...          }
...
...      },
...
...          { $sort: { totalDeaths: -1 } },
...
...          { "$limit": 10 }
...
...    ]
...
... )
{ "_id" : "Tornado", "totalDeaths" : 892 }
{ "_id" : "Flash Flood", "totalDeaths" : 691 }
{ "_id" : "Excessive Heat", "totalDeaths" : 605 }
{ "_id" : "Heat", "totalDeaths" : 590 }
{ "_id" : "Winter Weather", "totalDeaths" : 498 }
{ "_id" : "Rip Current", "totalDeaths" : 438 }
{ "_id" : "Thunderstorm Wind", "totalDeaths" : 340 }
{ "_id" : "Flood", "totalDeaths" : 285 }
{ "_id" : "Lightning", "totalDeaths" : 265 }
{ "_id" : "Cold/Wind Chill", "totalDeaths" : 259 }
>
```

Query 7:Top 10 states with maximum number of direct and indirect deaths

```
> db.storm.aggregate(     [       {         $group:          {            _id: "$STATE",           totalDeaths: { $sum: { $sum: [ "$DEATHS_DIRECT", "$DEATHS_INDIRECT" ] } }
     },          { $sort: { totalDeaths: -1 } },          { "$limit": 10 }     ]  )
{ "_id" : "TEXAS", "totalDeaths" : 602 }
{ "_id" : "NEVADA", "totalDeaths" : 488 }
{ "_id" : "FLORIDA", "totalDeaths" : 381 }
{ "_id" : "CALIFORNIA", "totalDeaths" : 375 }
{ "_id" : "ALABAMA", "totalDeaths" : 366 }
{ "_id" : "MISSOURI", "totalDeaths" : 352 }
{ "_id" : "ILLINOIS", "totalDeaths" : 308 }
{ "_id" : "PENNSYLVANIA", "totalDeaths" : 242 }
{ "_id" : "NORTH CAROLINA", "totalDeaths" : 229 }
{ "_id" : "NEW YORK", "totalDeaths" : 211 }
```

Query 8: How many times Tornado has occurred in different states between the year 2010 to 2018

```
mraunak@js-17-175:~/ProjectA
>
> db.storm.aggregate(
...     {$match: {EVENT_TYPE: "Tornado"}},
...     {$group: {_id: "$STATE", number: {$sum: 1}}}
... )
{ "_id" : "HAWAII", "number" : 3 }
{ "_id" : "MONTANA", "number" : 55 }
{ "_id" : "NEW YORK", "number" : 96 }
{ "_id" : "LOUISIANA", "number" : 458 }
{ "_id" : "ARKANSAS", "number" : 366 }
{ "_id" : "SOUTH CAROLINA", "number" : 153 }
{ "_id" : "FLORIDA", "number" : 354 }
{ "_id" : "KANSAS", "number" : 737 }
{ "_id" : "WISCONSIN", "number" : 255 }
{ "_id" : "MAINE", "number" : 28 }
{ "_id" : "WEST VIRGINIA", "number" : 33 }
{ "_id" : "OHIO", "number" : 259 }
{ "_id" : "TEXAS", "number" : 1083 }
{ "_id" : "VERMONT", "number" : 4 }
{ "_id" : "OREGON", "number" : 17 }
{ "_id" : "NEW MEXICO", "number" : 75 }
{ "_id" : "UTAH", "number" : 14 }
{ "_id" : "NEVADA", "number" : 16 }
{ "_id" : "PENNSYLVANIA", "number" : 166 }
{ "_id" : "RHODE ISLAND", "number" : 2 }
Type "it" for more
>
```

Query 9 : How many times the ‘tornado event’ have occurred in Indiana state in different years from 2010 to 2018

```
> db.storm.aggregate(
... ...     {$match: {EVENT_TYPE: "Tornado", STATE: "Indiana"}},
... ...     {$group: {_id: "$YEAR", number: {$sum: 1}}}
... ... )
> db.storm.aggregate(     {$match: {EVENT_TYPE: "Tornado", STATE: "INDIANA"}},     {$group: {_id: "$YEAR", number: {$sum: 1}}} )
{ "_id" : 2012, "number" : 21 }
{ "_id" : 2013, "number" : 53 }
{ "_id" : 2014, "number" : 31 }
{ "_id" : 2015, "number" : 24 }
{ "_id" : 2016, "number" : 43 }
{ "_id" : 2018, "number" : 16 }
{ "_id" : 2017, "number" : 37 }
{ "_id" : 2010, "number" : 31 }
{ "_id" : 2011, "number" : 85 }
>
```

Visualization

Data visualization is a technique that helps people to understand the significance of data by placing it in a visual context. Patterns, trends and correlations that might go undetected in text-based data can be exposed and interpreted easily with data visualization software.

There are different types of Big Data Visualization Categories such as Temporal, Hierarchical, Network, Multidimensional and Geospatial. As I wanted to perform visualization on multiple events, so I removed the filter event type tornado

```
mraunak@js-17-175:~/ProjectA
[js-17-175] mraunak ~/ProjectA-->mongoexport --username user535 --password pass535 --authenticationDatabase projectA \
and...>   --host localhost --port 27017 --db projectA --collection storm \
and...>   --fields EVENT_ID,YEAR,MONTH_NAME,TOR_F_SCALE,BEGIN_LAT,BEGIN_LON,END_LAT,END_LON,\
and...> INJURIES_DIRECT,INJURIES_INDIRECT,DEATHS_DIRECT,DEATHS_INDIRECT,DAMAGE_PROPERTY,STATE \
and...>   --type csv -o report.csv
2019-10-18T01:52:34.135-0500    connected to: mongodb://localhost:27017/
2019-10-18T01:52:35.145-0500    [#.......................]  projectA.storm  32000/559014  (5.7%)
2019-10-18T01:52:36.136-0500    [###.....................]  projectA.storm  80000/559014  (14.3%)
2019-10-18T01:52:37.153-0500    [#####...................]  projectA.storm  136000/559014  (24.3%)
2019-10-18T01:52:38.150-0500    [########................]  projectA.storm  200000/559014  (35.8%)
2019-10-18T01:52:39.141-0500    [#########...............]  projectA.storm  232000/559014  (41.5%)
2019-10-18T01:52:40.142-0500    [############............]  projectA.storm  280000/559014  (50.1%)
2019-10-18T01:52:41.143-0500    [##############..........]  projectA.storm  336000/559014  (60.1%)
2019-10-18T01:52:42.143-0500    [################........]  projectA.storm  392000/559014  (70.1%)
2019-10-18T01:52:43.149-0500    [##################......]  projectA.storm  424000/559014  (75.8%)
2019-10-18T01:52:44.145-0500    [####################....]  projectA.storm  472000/559014  (84.4%)
2019-10-18T01:52:45.159-0500    [#####################.]  projectA.storm  536000/559014  (95.9%)
2019-10-18T01:52:45.487-0500    [########################]  projectA.storm  559014/559014  (100.0%)
2019-10-18T01:52:45.487-0500    exported 559014 records
[js-17-175] mraunak ~/ProjectA-->
```

I used Tableau for performing the Geospatial Visualization Below is the screenshot of my first Visualization.

Geospatial or spatial data visualizations relate to real life physical locations, overlaying familiar maps with different data points. These types of data visualizations are commonly used to display weather reports, Weather related events such as Hail, freezing fog, high wind, flash flood etc. and even in other domains such as sales or acquisitions over time, and can be most recognizable for their use in political campaigns.

Maps are an amazing visualization to add to the dashboard as organizing data geographically tells an important story for the weather events. For example, if your dashboard is looking at the weather events in different states , it could be extremely useful to see the geographic locations of the event.

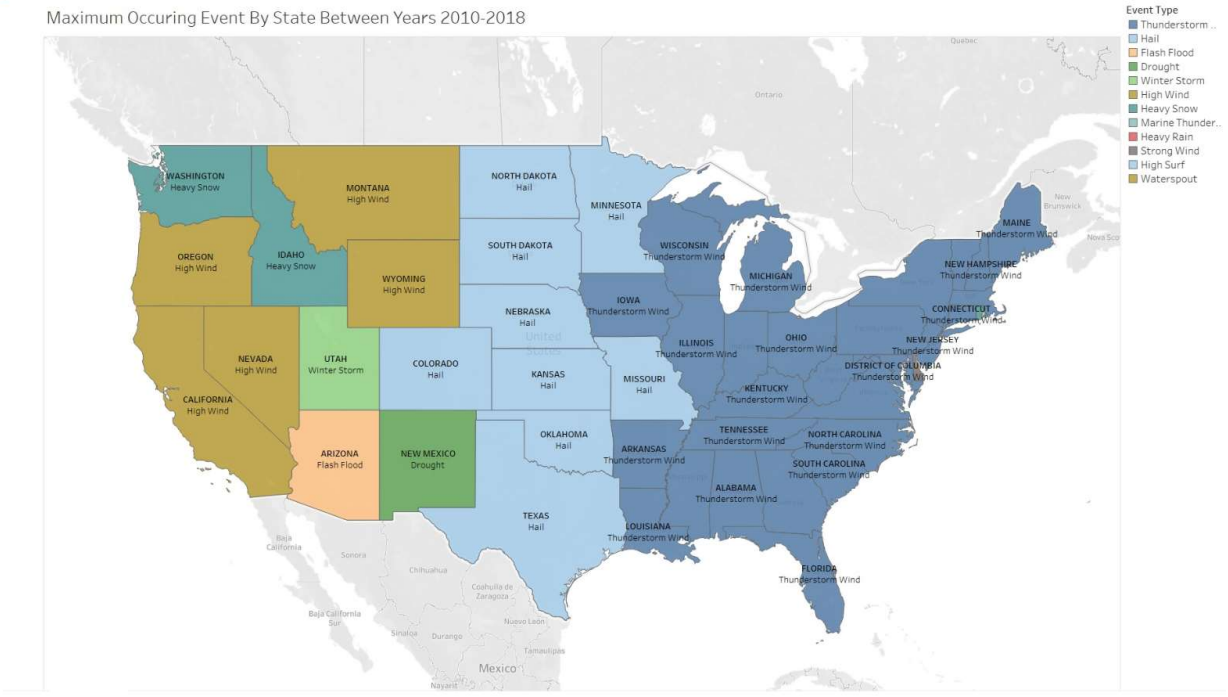


Fig: Event that has occurred maximum number of times in different states between 2010-2018

From the above graph we can see that the weather event 'Thunderstorm wind' is prevalent in the eastern, north eastern and south eastern states of USA such as Connecticut, North Carolina, District of Columbia, Michigan, Florida, Alabama. After thunderstorm, Hail is the second most occurring event in the USA. Mostly central states of USA are affected by hail such as Colorado, Kansas, Texas, South and North Dakota. The reasons of such weather event is the Rocky Mountains just the west of these states provides a source of very dry, unstable air aloft, that mixes with what low-level moisture we have near the surface which causes storm and also the rising sun starts to bake the east side of the Rockies, and that causes air to start flowing in from the east, which is up slope. That's another favorable condition to get storms forming.

High wind is frequent in the states such as Nevada, California, Oregon, Montana. Wind is caused by differences in the atmospheric pressure. When a difference in atmospheric pressure exists, air moves from the higher to the lower pressure area, resulting in winds of various speeds. The Santa Ana winds are strong, extremely dry downslope winds that originate inland and affect coastal Southern California and northern Baja California. High winds are dangerous and caused wildfires They originate from cool, dry high-pressure air masses in the Great Basin.

We can also see that heavy snow is common in Washington, Idaho.

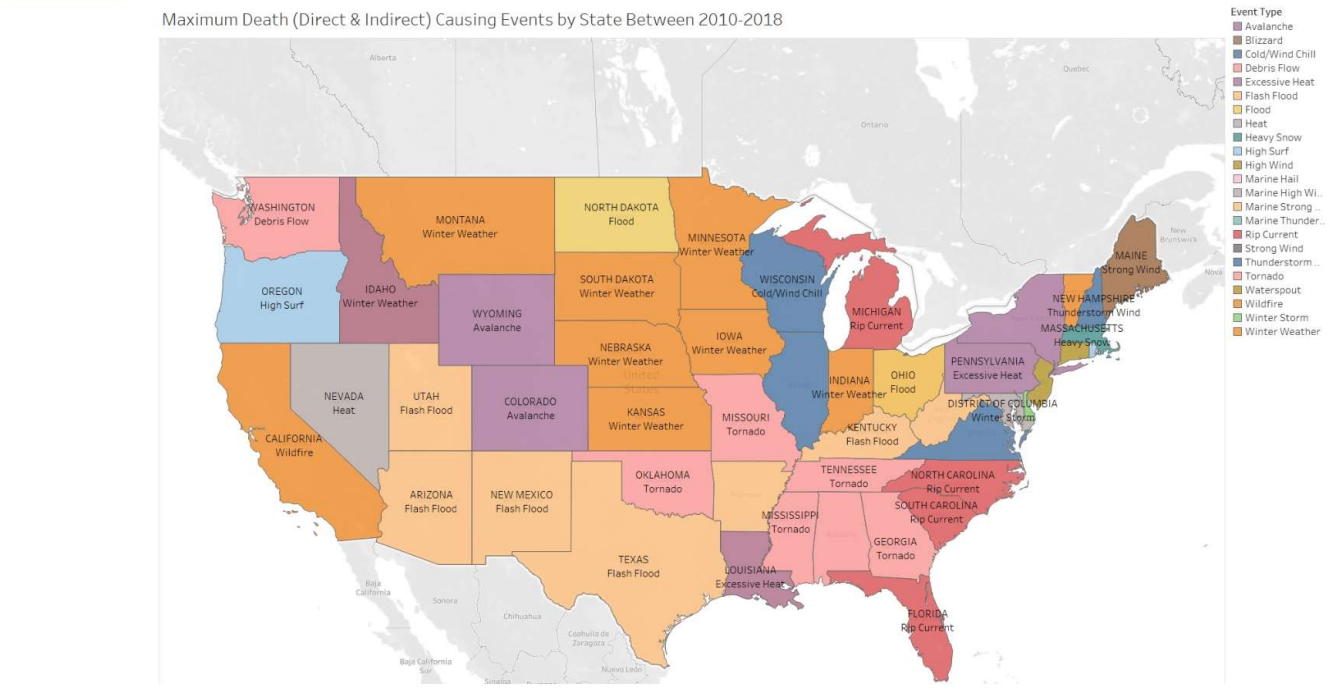


Fig: Maximum Death(Direct & Indirect) causing Events by State between 2010-2018

From this map, we get an idea about the events which cause maximum death in the different states of the USA. People in Indiana mostly gets affected by the Winter Weather, in states like Montana, South Dakota,Kansas, winter weather causes maximum number of deaths. However, the states like Arizona, New Mexico, Texas flash floods have been proved to be disastrous. In California wildfire caused by the High winds causes maximum deaths.

With the help of above visualization , we can be prepared for the weather event and storm events and can mitigate the loss and damages caused by these events in states

Challenges faced during the project :

During the transform step, I faced some issues with the script failing to transform the files. This was happening as the exrtactDir contained old files which were in json format already. I overcame this challenge by removing those files and cleaning up the extractDir using the python script.

Conclusion and Improvements in pipeline:

Overall, this project helped me to get familiar with NoSQL database MongoDB, and visualization tools like Tableau. From the visualization, we got a fair idea about the different weather events as well as storm events in different states of USA and their impact on human fatalities. These visualizations can help government as well as the people to be proactive and avoid the disastrous effect of the events. Some improvements in the pipeline are listed below.

1. Instead of using the file list with hardcoded filenames, we can create a matching pattern to directly get the filename from the website itself for the corresponding years.
2. In the python script, an automated function can be added which will check the extractDir if it contains json files already during the transform step.

References

https://www.oreilly.com/library/view/hadoop-mapreduce-v2/9781783285471/ch10s02.html

https://www.klipfolio.com/resources/articles/what-is-data-visualization

https://en.m.wikipedia.org/wiki/Santa_Ana_winds

https://www.5280.com/2019/06/why-does-hail-pummel-colorado-and-how-can-we-prepare-for-it/

https://www.nssl.noaa.gov/education/svrwx101/thunderstorms/

https://www1.ncdc.noaa.gov/pub/data/swdi/stormevents/csvfiles/