Project A – Working with National Weather Service Data

Submit Assignment

Due Oct 23 by 11:59pm **Points** 100 **Submitting** a text entry box or a file upload

1. Overview

In this project, you will be working with data from NOAA's National Weather Service (NWS) and taking it through steps that can be seen as a manually executed data pipeline. The data comes from the **Storm Events Database** (https://www.ncdc.noaa.gov/stormevents/details.jsp) , which contains data from January 1950 to May 2019. The database contains weather-related events from the US.

Using a subset of this database, you will use tools to extract the data and import it into a MongoDB database to analyze and visualize the data. You will be using a Jetstream virtual machine that has all the necessary tools installed.

Before you start working on the project, it is recommended that you familiarize yourself with the dataset by looking at some file examples and information about the fields / columns (**https://www.ncdc.noaa.gov/stormevents/ftp.jsp** (https://www.ncdc.noaa.gov/stormevents/ftp.jsp)) and look through the instructions to know what you'll need.

2. Objectives

- Learn the different stages of a data pipeline – download, extract, transform, load, analyze and visualize.
- Get basic knowledge of the MongoDB database and its *insert*, *update*, and *query* operations
- Learn to manipulate tools on the VM to answer interesting questions about the data.

3. What You Need

- Some knowledge of Linux OS. Jetstream VMs use Linux OS. If you're not familiar with Linux, refer to Linux Help resources on the **Additional Resources** page.
- Utilities script (available in **Canvas' Project A folder**)
- MongoDB 4.2 (installed on the VM)
- Python Version >= 2.7 (installed on the VM)
- Pandas & Pymongo Libraries (installed on the VM)
- Jupyter Notebook/ Plotly for visualization (available for download, see [3, 4])
- Tableau Desktop Version (optional, available on IUware)

4. Deliverables

Submit the following through Canvas:

- Comma separated value (csv) file with analyzed events
- An image file with your visualization (please use common formats, e.g., png or jpeg, not specific tools formats)
- A project report that describes the details of your work. It should include:
  - an introduction (1 paragraph)
  - a description of your pipeline (1-2 paragraphs, you can re-use the diagram we provided, but need to add details about your own execution)
  - a brief analysis of your results, both pipeline execution and visualizations (2-3 paragraphs)
  - any challenges you had and how you addressed them
  - a conclusion with suggestions for improving the pipeline (1-2 paragraphs)
  - list of resources used in your work

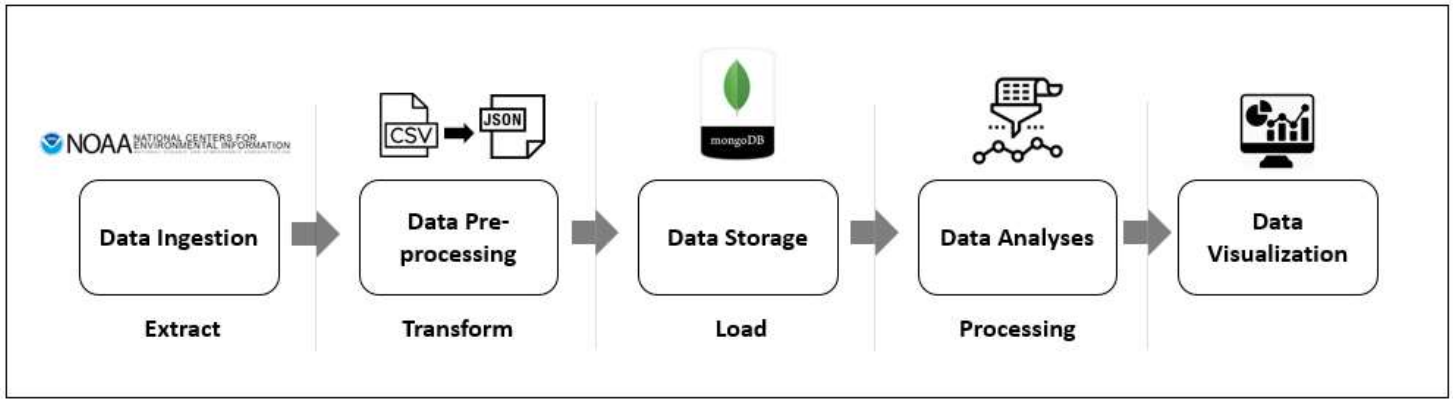
5. Grading

The assignment is graded on a 100-point scale.

- Submitting all data files with correct answers and following the instructions will give you a maximum of 75 points.
- Points between 76 and 100 will be awarded if the report goes beyond simple description of steps and shows understanding of the pipeline, your own interpretations of the results and comments on how this pipeline can be improved.
- Points will be taken off for files that do not read or are otherwise incorrect, for not listing resources you used, or for missing steps.
- Extra credit, up to 10 more points, will be awarded for going a step beyond in refinement and suggestions, i.e., for practicing the steps with your own criteria, for your own ideas of how this dataset can be analyzed, or for creating your own distinct visualization.

6. Data Pipeline Overview

The diagram below provides an overview of your data pipeline:



We are using the following stages in this data pipeline:

1. **Extract/Data Ingestion.** In this stage, we are importing/downloading the data from the data source – the NOAA website at **https://www1.ncdc.noaa.gov/pub/data/swdi/stormevents/csvfiles/ (https://www1.ncdc.noaa.gov/pub/data/swdi/stormevents/csvfiles/)** . This data will be downloaded to a landing directory in your VM. The data is downloaded in GZ compressed format, so we need to extract each CSV file. You will use a Python utility script provided to you to extract the files from the landing directory and save them in the extraction directory.

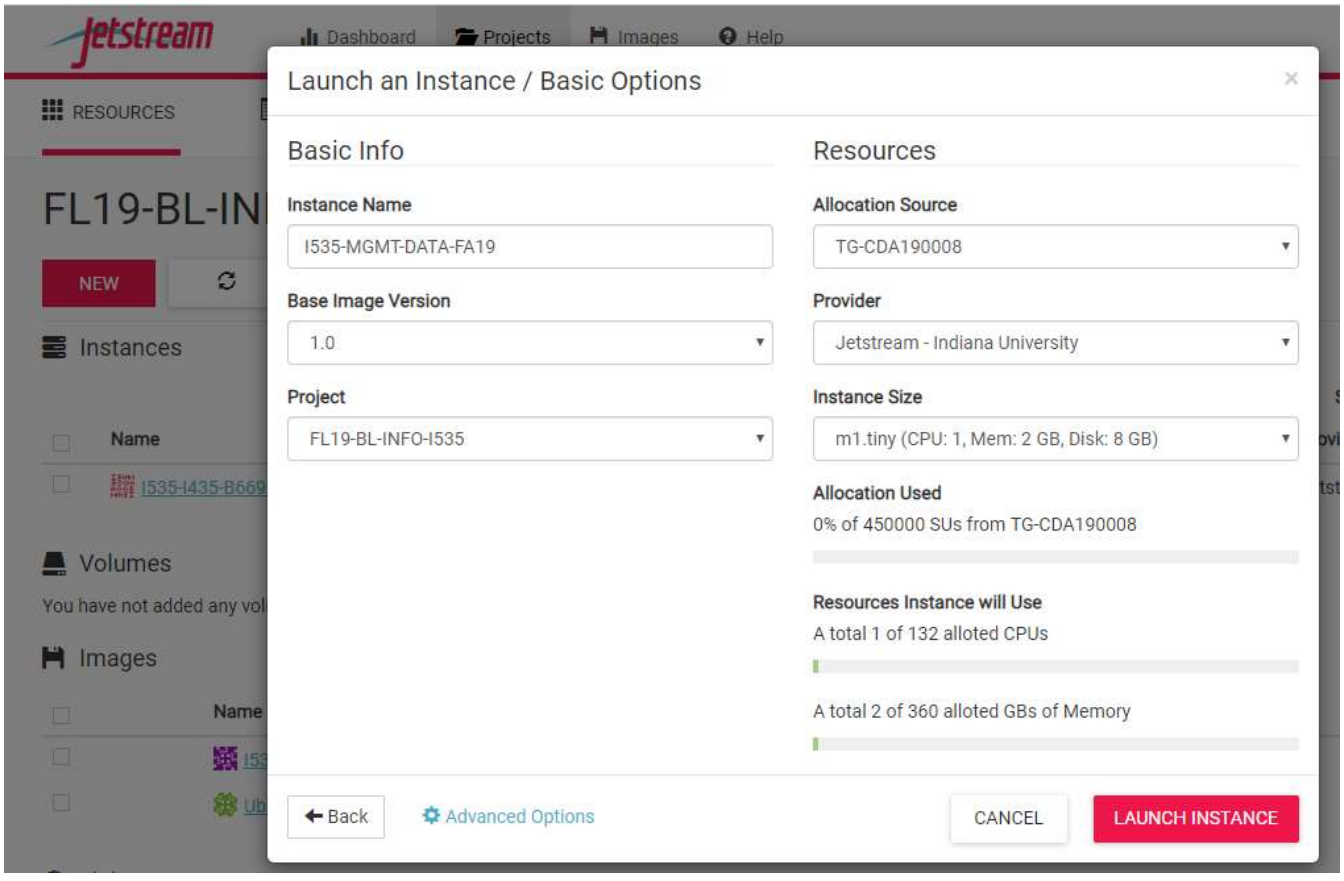
2. **Transform/Data Preprocessing.** To store the data in MongoDB, we need to convert the files from CSV to JSON format. For this we are using the Pandas library [1], which will allow us to read the files and select the columns we need.
3. **Load/Data Storage.** We will use PyMongo to work with MongoDB. It is a Python distribution that contains tools for working with **MongoDB** (http://www.mongodb.org/), and is the recommended way to work with MongoDB when you rely on Python [2]. Once the data files are converted to JSON with required columns, we will store them in MongoDB.
4. **Processing/Analyze.** In this stage, we will do basic operations on data such as insert a new record, update a record, delete a record and query the data. We will also write the result of our queries to generate a report.
5. **Visualize.** You can use a visualization tool of your choice (or use the options described above) to create visualizations that provide insights about the trends in our data.

7. Pipeline Stages

Create and Configure Virtual Machine

Use tutorials from XSEDE if needed to launch and set up your own VM instance. Rules for creating a VM:

- Configure a small VM to avoid consuming unnecessary compute resources.
- Search for the following image “I535-MGMT-DATA-FA19”; make sure you’re using the right allocation (TG-CDA190008); select provider “Jetstream – Indiana University” and instance size “m1.tiny”. Launch the instance.



- It takes some time to build your instance, wait for "Status: Active" and "Activity: N/A" before trying to log in using a Web Shell or SSH command.
- Be sure to suspend your VM when you are done for the day. Failing to do so will reduce your quota.

Create a project directory (e.g., *ProjectA*) and go to that directory:

```
--> mkdir ProjectA
--> ls
--> cd ProjectA/
```

Copy project files *fileslist* and *project_utilities.py* to this directory. You can transfer files via SCP / SFTP or the Web shell / Web desktop, see **Jetstream User Guide** (https://portal.xsede.org/jetstream), Transferring Files. Check if files have been copied.

Two more directories (landDir and extractDir) will be automatically created when you run the script.

Create a MongoDB Collection

MongoDB is already installed on your VM. In MongoDB your records (documents) are stored in collections, the collections are stored in a database. For this project we will be using **"projectA"** database in MongoDB.

Your database is protected by a password. We will be using **user535 / pass535** as our login credentials for connecting to the database ***projectA*.**

Log into MongoDB using *mongo* command as below (for more about mongo shell, see [6]:

```
--> mongo -u user535 -p pass535 --authenticationDatabase projectA
```

Now you can create a collection in MongoDB where all the files will be loaded. We name our collection "**storm**", but you can use any name.

```
> use projectA;
> db.createCollection("storm")
```

Extract and Transform

You will use **python_utilities.py** to download, extract, transform and load the data into MongoDB. Feel free to open it in a text editor or Python IDE and review it.

To see its usage, you can also use the following command:

```
--> python project_utilities.py help
```

```
[js-157-241] utkumar ~/ProjectA-->python project_utilities.py help
INFO:__main__:******************** STARTING SCRIPT project_utilities.py **********************
INFO:__main__:****************** USERNAME : utkumar ******************


Usage: python project_utilities.py {download|extract|transform|load|cleanup|help}
download <start year> <end year>                                    download storm data from NOAA's National Weather Service int
extract                                                             extract the downloaded gz files to CSV format into extractio
transform <chunksize>                                               transforms data from csv to json and selects all columns
transform <comma separated list of columns> <chunksize>             transforms data from csv to json and selects given columns
load <hostname> <port> <database> <collection> <username> <password>    load the data (json files) into MongoDB
cleanup                                                             delete all files from landing and extract directories
help                                                                display help menu
[js-157-241] utkumar ~/ProjectA-->
```

DOWNLOAD

To download the data, you need to specify a year range after the *download* keyword.

Note: Since, the data files are large, it is recommended to keep the number of years at maximum to 10. Start and end years are inclusive.

```
--> python project_utilities.py download <start year> <end year>
```

```
[js-157-241] utkumar ~/ProjectA-->python project_utilities.py download 2008 2017
INFO:__main__:******************** STARTING SCRIPT project_utilities.py **********************
INFO:__main__:****************** USERNAME : utkumar ******************
INFO:__main__:Python Major Version : 2
INFO:__main__:Script Path: /home/utkumar/ProjectA
INFO:__main__:URL : https://www1.ncdc.noaa.gov/pub/data/swdi/stormevents/csvfiles/
INFO:__main__:Landing Directory does not exist. Creating directory
INFO:__main__:Landing Directory created.
INFO:__main__:Extraction Directory does not exist. Creating directory
INFO:__main__:Extraction Directory created.
INFO:__main__:Landing Directory : /home/utkumar/ProjectA/landDir/
INFO:__main__:Extraction Directory : /home/utkumar/ProjectA/extractDir/
INFO:__main__:************************************************************************
INFO:__main__:************** Beginning file download module ***********************
INFO:__main__:************************************************************************
INFO:__main__:Downloading file StormEvents_details-ftp_v1.0_d2008_c20180718.csv.gz
INFO:__main__:Successfully downloaded /home/utkumar/ProjectA/landDir/StormEvents_details-ftp_v1.0_d2008_c20180718.csv.gz
```

Once all the files are downloaded successfully, check that files are available in the "landDir" directory using the *ls* command.

EXTRACT

Use the command below to extract the CSV files from GZ files into the "extractDir" directory:

```
--> python project_utilities.py extract
```

```
[js-157-241] utkumar ~/ProjectA-->python project_utilities.py extract
INFO:__main__:******************** STARTING SCRIPT project_utilities.py **********************
INFO:__main__:****************** USERNAME : utkumar ******************
INFO:__main__:Python Major Version : 2
INFO:__main__:Script Path: /home/utkumar/ProjectA
INFO:__main__:URL : https://www1.ncdc.noaa.gov/pub/data/swdi/stormevents/csvfiles/
INFO:__main__:Landing Directory : /home/utkumar/ProjectA/landDir/
INFO:__main__:Extraction Directory : /home/utkumar/ProjectA/extractDir/
INFO:__main__:************************************************************************
INFO:__main__:*************** Beginning file extract module **********************
INFO:__main__:************************************************************************
INFO:__main__:Extracting file /home/utkumar/ProjectA/landDir/StormEvents_details-ftp_v1.0_d2008_c20180718.csv.gz
INFO:__main__:Successfully extracted file StormEvents_details-ftp_v1.0_d2008_c20180718.csv.gz
INFO:__main__:Extracting file /home/utkumar/ProjectA/landDir/StormEvents_details-ftp_v1.0_d2009_c20180718.csv.gz
```

TRANSFORM

This data comes in the CSV format and needs to be converted to JSON to be loaded into MongoDB. We also want to only keep columns that are relevant to us and remove the null key-value pairs.

As MongoDB is a schema-less database, we don't have to specify number of columns or data types before the load.

The argument *"chunksize"* allows to stage the transformation and work with very large files. Chunk size allows to specify a number of rows that will be transformed per file in one go. For example, a file with 100,000 rows will be transformed into 4 json files if chunk size is set to 25,000.

Note: You can start with 25000 and decrease it if you face memory issues. Make sure you delete the json files from the extract directory before running *transform* command again (rm extractDir/*.json).

If you're still facing issues during loading, you'll need to clean MongoDB collection before loading again (db.storm.remove({ }))

```
--> python project_utilities.py <comma separated list of columns> <chunksize>
```

To keep all the columns, you can run the command as follows:

```
--> python project_utilities.py transform <chunksize>
```

We suggest to keep only the columns that are relevant to the storm episodes and events, using the list of columns below:

```
--> python project_utilities.py transform BEGIN_DAY,END_DAY,EPISODE_ID,EVENT_ID,STATE,YEAR,MONTH_NAME,
EVENT_TYPE,CZ_TYPE,INJURIES_DIRECT,INJURIES_INDIRECT,DEATHS_DIRECT,DEATHS_INDIRECT,DAMAGE_PROPERTY,
DAMAGE_CROPS,SOURCE,MAGNITUDE,MAGNITUDE_TYPE,FLOOD_CAUSE,TOR_F_SCALE,TOR_LENGTH,TOR_WIDTH,
BEGIN_LAT,BEGIN_LON,END_LAT,END_LON,EPISODE_NARRATIVE,EVENT_NARRATIVE 10000
```

```
[js-157-241] utkumar ~/ProjectA-->python project_utilities.py transform BEGIN_DAY,END_DAY,EPISODE_ID,EVENT_ID
HS_DIRECT,DEATHS_INDIRECT,DAMAGE_PROPERTY,DAMAGE_CROPS,SOURCE,MAGNITUDE,MAGNITUDE_TYPE,FLOOD_CAUSE,TOR_F_SCAL
INFO:__main__:******************** STARTING SCRIPT project_utilities.py **********************
INFO:__main__:****************** USERNAME : utkumar ******************
INFO:__main__:Python Major Version : 2
INFO:__main__:Script Path: /home/utkumar/ProjectA
INFO:__main__:URL : https://www1.ncdc.noaa.gov/pub/data/swdi/stormevents/csvfiles/
INFO:__main__:Landing Directory : /home/utkumar/ProjectA/landDir/
INFO:__main__:Extraction Directory : /home/utkumar/ProjectA/extractDir/
INFO:__main__:**********************************************************************
INFO:__main__:*************** Beginning transform data module ********************
INFO:__main__:**********************************************************************
INFO:root:Transforming file StormEvents_details-ftp_v1.0_d2008_c20180718.csv
INFO:root:Successfully saved chunk file StormEvents_details-ftp_v1.0_d2008_c20180718.csv0.json
```

LOAD

Once all the transformations are done, we can load the data into MongoDB using the below command:

```
--> python project_utilities.py load <hostname> <port> <database> <collection> <username> <password>
```

Use *localhost 27017* as a hostname along with our database, collection and credential details to load the data.

At the end of the log output, there will be the total number of records. Write down the number as you will verify it later by running a count in the *storm* collection.

```
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2017_c20190
INFO:__main__:Beginning to load file StormEvents_details-ftp_v1.0_d2017_c20190817.csv
INFO:__main__:Successfully loaded data file StormEvents_details-ftp_v1.0_d2017_c20190
INFO:__main__:TOTAL NUMBER OF RECORDS LOADED IN MONGODB : 625251
INFO:__main__:**********************************************************************
INFO:__main__:********************** End of load data module ***********************
INFO:__main__:**********************************************************************
[js-157-241] utkumar ~/ProjectA-->
```

CHECK

After the files are loaded into MongoDB, you can check if the data is loaded by using the query below.

```
--> mongo -u <username> -p <password> --authenticationDatabase <database name>
```

Now, you can switch to our database and spot-check the records using a MongoDB *find* command:

```
> use projectA;
> db.storm.find().pretty()
```

To check all the records were loaded, run a count of data records – the number should be the same as in your load step:

```
> db.storm.find().count()
```

Querying and Analytics

At this stage you will perform some operations that allow the analyst to verify and modify the data:

1. Insert a record into collection
2. Update records in collection
3. Delete records from collection
4. Query records from collection

INSERT A RECORD INTO COLLECTION

If you have additional records that need to be inserted or you realized a record was skipped during your load, you can insert a record using the *db.collection.insert(<document or array of documents>)* command.

Practice inserting the records, for example:

```
> db.storm.insert( {
BEGIN_DAY: 20,
END_DAY: 20,
EPISODE_ID: 132265438,
EVENT_ID:432432412,
STATE: "Indiana",
YEAR: 2029,
MONTH_NAME: "April",
EVENT_TYPE: "Tornado",
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,
TOR_F_SCALE: "F2",
TOR_LENGTH: 1,
TOR_WIDTH: 20,
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. "
} )
```

Use *db.collection.find()* command to view your inserted record (e.g., by specifying its EVENT_ID)

UPDATE A RECORD IN COLLECTION

If a record turns to be incorrect, it can be updated with *db.collection.update()* command. You can practice updating your records, e.g., changing year from 2029 to 2019 and adding text 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." }
}
)
```

REMOVE A RECORD FROM COLLECTION

To remove a record, use *db.collection.delete*()* command. You can remove one or many records at once:

```
> db.storm.deleteOne( { EVENT_ID: 432432412 } );
```

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

```
> db.storm.deleteMany( { EVENT_TYPE : "Lake-Effect Snow" } );
```

```
> db.storm.deleteMany( { EVENT_TYPE : "Lake-Effect Snow" } );
{ "acknowledged" : true, "deletedCount" : 2185 }
```

QUERY RECORDS FROM COLLECTION

To query your collection, you can use the following methods on db.collection: *find()*, *distinct()*, or *aggregate()*. The *find()* method was used above to view some documents. The last one – aggregation – is a useful tool for grouping values from multiple documents together [7]. Use examples below to practice both:

Find distinct event types

```
> db.storm.distinct( "EVENT_TYPE" )
```

Find top 10 events between 1998 and 2018

```
> db.storm.aggregate(
    [
       {
         "$group" :
                {
                   _id:"$EVENT_TYPE",
                   count:{$sum:1}
                 }
       },
       {
         $sort:
                { count: -1 }
       },
       {
         "$limit": 10
       }
```

```
    ])
```

To keep practicing querying, you can come up with more complex queries, e.g.:

- All narratives for the “Tornado” event in specific years (to use in text analytics).
- Total property damage caused by one of the events (e.g., Tornado or Flashflood) in the state of Indiana by each year.
- Top 10 events that caused the greatest number of direct and indirect deaths for a specific period.

EXPORT RESULTS

While MongoDB has its own tools to visualize data (**MongoDB Charts** (https://www.mongodb.com/products/charts)), you can export your data to use with any other visualization tools. The example below shows how to export data to visualize Tornado events in the state of Indiana from 2008 to 2017.

```
> mongoexport --username user535 --password pass535 --authenticationDatabase projectA
  --host localhost --port 27017 --db projectA --collection storm
  --fields EVENT_ID,YEAR,MONTH_NAME,TOR_F_SCALE,BEGIN_LAT,BEGIN_LON,END_LAT,END_LON,
  INJURIES_DIRECT,INJURIES_INDIRECT,DEATHS_DIRECT,DEATHS_INDIRECT,DAMAGE_PROPERTY
  -q '{ "EVENT_TYPE" : "Tornado" }' --type csv -o report.csv
```

```
[js-157-241] utkumar ~/ProjectA-->mongoexport --username user535 --password pass535 --authenticati
 EVENT_ID,YEAR,MONTH_NAME,TOR_F_SCALE,BEGIN_LAT,BEGIN_LON,END_LAT,END_LON,INJURIES_DIRECT,INJURIES_
-type csv -o report.csv
2019-09-20T16:13:45.003-0400    connected to: mongodb://localhost:27017/
2019-09-20T16:13:45.628-0400    exported 13959 records
[js-157-241] utkumar ~/ProjectA-->
```

8. Visualization

Your data were exported into a CSV file (e.g., report.csv). Now you can download it using **WinSCP** or a **SCP command on Mac** to get the file and visualize it using Python's plotly library. You can also use Tableau Desktop or any other tool of your choice to create a visualization.

The visualization instructions for plotly are available as **a Jupiter notebook in our project folder**. You can follow the instructions there or come up with your own visualization.

9. References

1. Python Data Analysis Library. Available at **https://pandas.pydata.org/** (https://pandas.pydata.org/)
2. PyMongo 3.9.0 Documentation. Available at **https://api.mongodb.com/python/current/** (https://api.mongodb.com/python/current/)
3. Jupyter notebook. Available at **https://test-jupyter.readthedocs.io/en/latest/install.html** (https://test-jupyter.readthedocs.io/en/latest/install.html)
4. Available at **https://plot.ly/python/getting-started/** (https://plot.ly/python/getting-started/)
5. MongoDB tutorial. Available at **https://docs.mongodb.com/manual/mongo/** (https://docs.mongodb.com/manual/mongo/)
6. Mongo Shell Quick Reference. Available at **https://docs.mongodb.com/manual/reference/mongo-shell/** (https://docs.mongodb.com/manual/reference/mongo-shell/)

7. MongoDB Aggregation. Available at **https://docs.mongodb.com/manual/aggregation/ (https://docs.mongodb.com/manual/aggregation/)**