Gun Violence 2015-2018

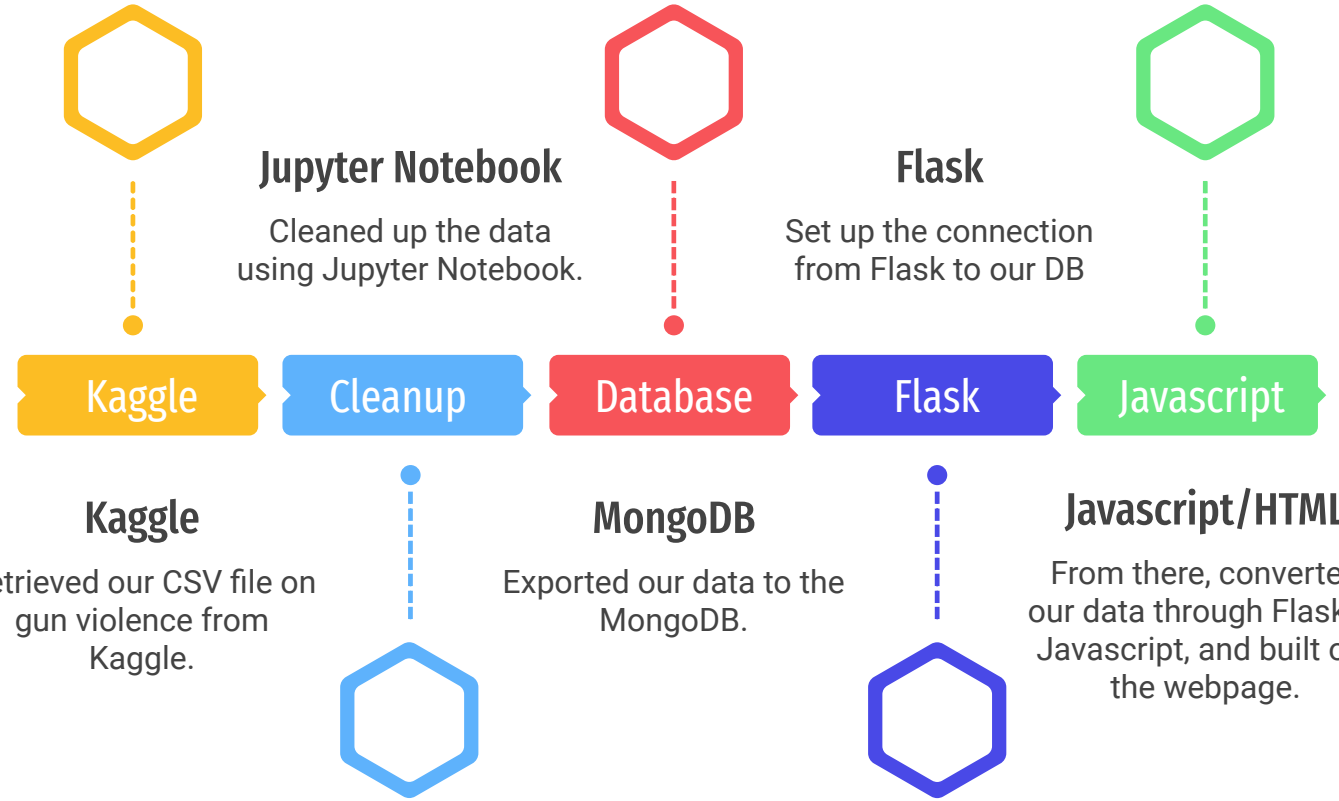
Presentation by Sara Edgar, Jessica Johnson, Amanda Vital, and Natalie Lassen

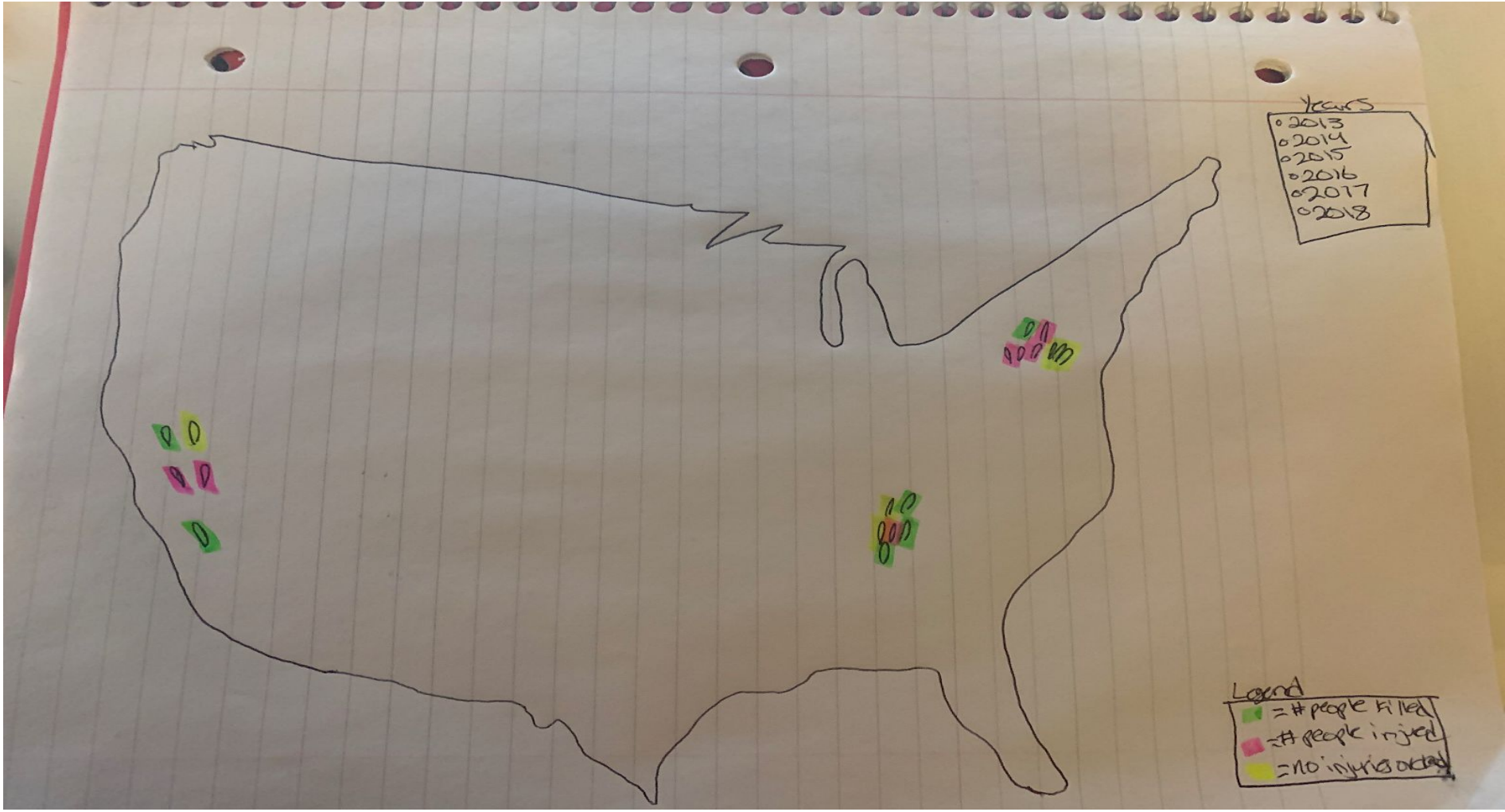
Theme

Gun Violence in America

- 20,000 people died from gun violence in 2020
  - Almost a 25% increase from 2019
- Using the number of fatalities due to shootings
  - PA: East Coast
  - CA: West Coast
  - TN: South
- Comparing the years of 2015, 2016, 2017 and 2018

The Process


Jupyter Notebook
Cleaned up the data using Jupyter Notebook.
Flask
Set up the connection from Flask to our DB
Kaggle
Cleanup
Database
Flask
Javascript
Kaggle
Retrieved our CSV file on gun violence from Kaggle.
MongoDB
Exported our data to the MongoDB.
Javascript/HTML
From there, converted our data through Flask to Javascript, and built out the webpage.


Years
2013
2014
2015
2016
2017
2018
Legend
=# people killed
=# people injured
=no injuries reported

```
In [1]: import pandas as pd
        import numpy as np
        import pymongo

In [2]: df=pd.read_csv("gun-violence-data_01-2013_03-2018.csv.zip")

In [3]: df.head()

Out[3]:
```

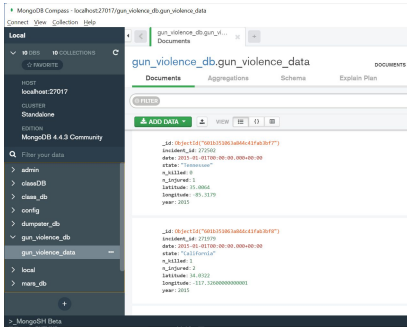
	incident_id	date	state	city_or_county	address	n_killed	n_injured	incident_url	source_url
0	461105	2013-01-01	Pennsylvania	Mckeesport	1506 Versailles Avenue and Counsin Street	0	4	http://www.gunviolencearchive.org/incident/461105	http://www.post-gazette.com/local/south/2013
1	460726	2013-01-01	California	Hawthorne	13500 block of Cerise Avenue	1	3	http://www.gunviolencearchive.org/incident/460726	http://www.dailybulletin.com/
2	478855	2013-01-01	Ohio	Lorain	1776 East 28th Street	1	3	http://www.gunviolencearchive.org/incident/478855	http://chronicle.northcoastnow
3	478925	2013-01-05	Colorado	Aurora	16000 block of East Ithaca Place	4	0	http://www.gunviolencearchive.org/incident/478925	http://www.dailydemocrat.com/s...
4	478959	2013-01-07	North Carolina	Greensboro	307 Mourning Dove Terrace	2	2	http://www.gunviolencearchive.org/incident/478959	http://www.journalnow.com/n

5 rows × 29 columns

```
In [4]: conn = 'mongodb://localhost:27017'
        client = pymongo.MongoClient(conn)
        # Declare the database
```

Jupyter Notebook

MongoDb




```
17  // Create the map object with center, zoom level and default layer.
18  // Create the map with our layers
19  var map = L.map("mapid", {
20    center: [39.8283, -98.5795],
21    zoom: 5,
22    layers: [
23      layers.fifteen,
24      layers.sixteen,
25      layers.seventeen,
26      layers.eighteen,
27    ],
28    onClick: function (e, legendItem) {
29      var index = legendItem.datasetIndex;
30      var ci = this.chart;
31      var alreadyHidden = (ci.getDatasetMeta(index).hidden === null) ? false : ci.getDatasetMeta(index).hidden;
32      ci.data.datasets.forEach(function (e, i) {
33        var meta = ci.getDatasetMeta(i);
34        if (i !== index) {
35          if (!alreadyHidden) {
36            meta.hidden = meta.hidden === null ? !meta.hidden : null;
37          } else if (meta.hidden === null) {
38            meta.hidden = true;
39          }
40        } else if (i === index) {
41          meta.hidden = null;
42        }
43      });
44      ci.update();
45    }
46  });
```

Java Script

```
1  from flask import Flask, render_template, jsonify
2  from pymongo import MongoClient
3  from flask_pymongo import pymongo
4  import json
5
6  app = Flask(__name__)
7
8  conn = 'mongodb://localhost:27017'
9  client = pymongo.MongoClient(conn)
10 db = client['guns_db']
11 collection = db['guns']
12
13 @app.route('/')
14 def return_all():
15     returned_data = {}
16     results = collection.find({}, {"_id":False})
17     for result in results:
18         year = result['year']
19         if year in returned_data:
20             returned_data[year].append(result)
21         else:
22             returned_data[year] = [result]
23     return render_template("index.html", event=returned_data)
24
25
26 if __name__ == "__main__":
27     app.run(debug=True)
```

Flask

```
conn = 'mongodb://localhost:27017'
client = pymongo.MongoClient(conn)
# Declare the database
db = client.guns_db
# Declare the collection
guns = db.guns
```

```
client.list_database_names()
```

```
['admin',
 'classDB',
 'config',
 'craigslist_db',
 'dumpster_db',
 'guns_db2',
 'local',
 'mars_db',
 'store_inventory',
 'team_db',
 'travel_db']
```

```
df.columns
```

```
Index(['incident_id', 'date', 'state', 'city_or_county', 'address', 'n_killed',
       'n_injured', 'incident_url', 'source_url',
       'incident_url_fields_missing', 'congressional_district', 'gun_stolen',
       'gun_type', 'incident_characteristics', 'latitude',
       'location_description', 'longitude', 'n_guns_involved', 'notes',
       'participant_age', 'participant_age_group', 'participant_gender',
       'participant_name', 'participant_relationship', 'participant_status',
       'participant_type', 'sources', 'state_house_district',
       'state_senate_district'],
      dtype='object')
```

```
dropped_columns=df.drop(columns=['city_or_county','address','incident_url','source_url','incident_ur
essional_district','gun_stolen','gun_type','incident_characteristics','location_description','n_guns
icipant_age','participant_age_group','participant_name','participant_relationship','participant_stat
e','sources','state_house_district','state_senate_district','participant_gender'])
```

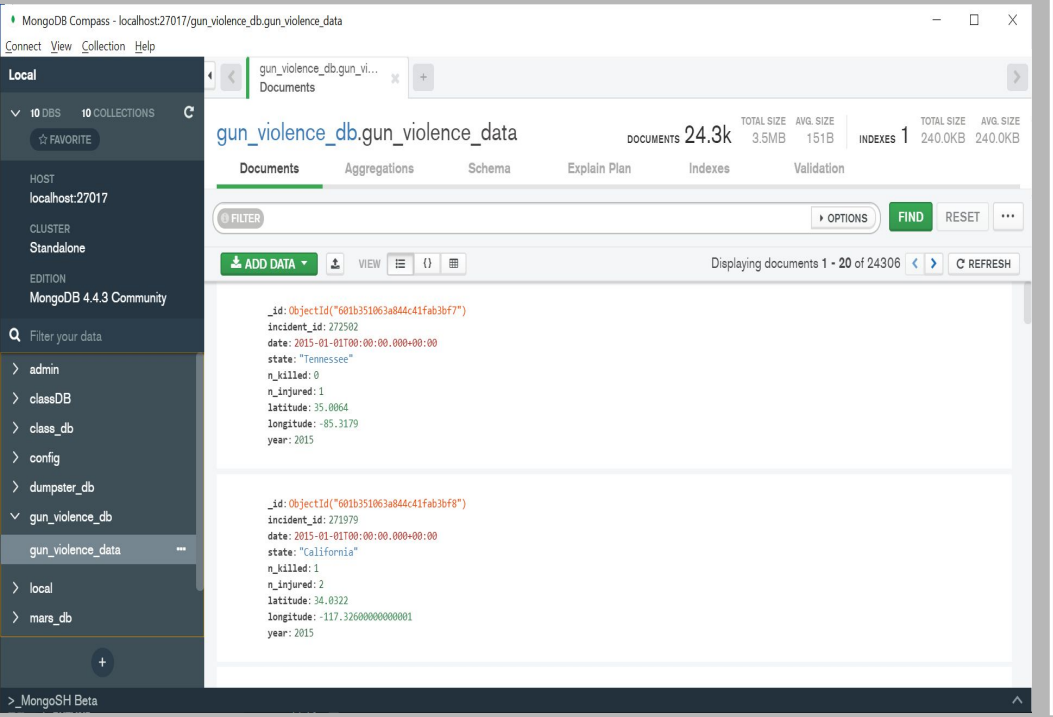
Jupyter Notebook

```
clean = dropped_columns.loc[(dropped_columns['state'] == "Pennsylvania") | (dropped_columns['state'] == "Tennessee") | (drop
ped_columns['state'] == "California")]
```

```
clean
```

	incident_id	date	state	n_killed	n_injured	latitude	longitude
0	461105	2013-01-01	Pennsylvania	0	4	40.3467	-79.8559
1	460726	2013-01-01	California	1	3	33.9090	-118.3330
8	479389	2013-01-21	California	0	4	37.9656	-121.7180
10	491674	2013-01-23	Tennessee	1	3	35.0221	-85.2697
15	479573	2013-02-02	Tennessee	0	5	35.0803	-89.8871
...	...	...	...	...	...	...	...
239655	1082089	2018-03-31	California	0	0	32.8936	-117.1360
239657	1082394	2018-03-31	California	0	0	35.5019	-119.2830
239658	1082392	2018-03-31	California	1	0	NaN	NaN
239660	1082091	2018-03-31	California	2	0	37.9478	-121.3140
239667	1082234	2018-03-31	Tennessee	0	1	35.2045	-89.9872

32861 rows × 7 columns


MongoDB Compass - localhost:27017/gun_violence_db.gun_violence_data
Connect View Collection Help
Local
10 DBS 10 COLLECTIONS
FAVORITE
HOST
localhost:27017
CLUSTER
Standalone
EDITION
MongoDB 4.4.3 Community
Filter your data
admin
classDB
class_db
config
dumpster_db
gun_violence_db
gun_violence_data
local
mars_db
gun_violence_db.gun_vi...
Documents
gun_violence_db.gun_violence_data
DOCUMENTS 24.3k
TOTAL SIZE 3.5MB
AVG. SIZE 151B
INDEXES 1
TOTAL SIZE 240.0KB
AVG. SIZE 240.0KB
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
FILTER
OPTIONS
FIND
RESET
ADD DATA
VIEW
Displaying documents 1 - 20 of 24306
REFRESH
_id: ObjectId("601b351063a844c41fab3bf7")
incident_id: 272502
date: 2015-01-01T00:00:00.000+00:00
state: "Tennessee"
n_killed: 0
n_injured: 1
latitude: 35.0064
longitude: -85.3179
year: 2015
_id: ObjectId("601b351063a844c41fab3bf8")
incident_id: 271979
date: 2015-01-01T00:00:00.000+00:00
state: "California"
n_killed: 1
n_injured: 2
latitude: 34.0322
longitude: -117.32600000000001
year: 2015
>_MongoSH Beta

MongoDB

Flask

app.py | style.css | logic2.js | index.html | logic.js

Homework > Visualization_Project > app.py > ...

```python
from flask import Flask, render_template, jsonify
from pymongo import MongoClient
from flask_pymongo import pymongo
import json

app = Flask(__name__)

conn = 'mongodb://localhost:27017'
client = pymongo.MongoClient(conn)
db = client['guns_db']
collection = db['guns']

@app.route('/')
def return_all():
    returned_data = {}
    results = collection.find({}, {"_id":False})
    for result in results:
        year = result['year']
        if year in returned_data:
            returned_data[year].append(result)
        else:
            returned_data[year] = [result]
    return render_template("index.html", event=returned_data)


if __name__ == "__main__":
    app.run(debug=True)
```

app.py | style.css | logic2.js | index.html

Homework > Visualization_Project > static > js > logic2.js > ...

```
// We create the tile layer that will be the background of our map.
var lightmap = L.tileLayer("https://api.mapbox.com/styles/v1/mapbox/{id}/tiles/{z}/{x}/{y}?access_token={accessToken}", {
  attribution: "Map data &copy; <a href=\"https://www.openstreetmap.org/\">OpenStreetMap</a> contributors, <a href=\"https://creativecommons.org/l
  maxZoom: 18,
  id: "light-v10",
  accessToken: API_KEY
});
// Check that we are pulling our data correctly by console logging
console.log(user);
// Initialize all of the LayerGroups we'll be using
var layers = {
  fifteen: new L.LayerGroup(),
  sixteen: new L.LayerGroup(),
  seventeen: new L.LayerGroup(),
  eighteen: new L.LayerGroup(),
};
```

Java Script

```
// Create the map object with center, zoom level and default layer.
// Create the map with our layers
var map = L.map("mapid", {
  center: [39.8283, -98.5795],
  zoom: 5,
  layers: [
    layers.fifteen,
    layers.sixteen,
    layers.seventeen,
    layers.eighteen,
  ],
  onClick: function (e, legendItem) {
    var index = legendItem.datasetIndex;
    var ci = this.chart;
    var alreadyHidden = (ci.getDatasetMeta(index).hidden === null) ? false : ci.getDatasetMeta(index).hidden;
    ci.data.datasets.forEach(function (e, i) {
      var meta = ci.getDatasetMeta(i);
      if (i !== index) {
        if (!alreadyHidden) {
          meta.hidden = meta.hidden === null ? !meta.hidden : null;
        } else if (meta.hidden === null) {
          meta.hidden = true;
        }
      } else if (i === index) {
        meta.hidden = null;
      }
    });
    ci.update();
  }
});
```

```
// Create a base layer that holds all three maps.
lightmap.addTo(map);
// Add an extra layer group for combined years
let allYears = new L.LayerGroup();
// Add overlays for each year
var overlays = {
    "2015": layers.fifteen,
    "2016": layers.sixteen,
    "2017": layers.seventeen,
    "2018": layers.eighteen,
  };
// Then we add a control to the map that will allow the user to change which
// layers are visible.
L.control.layers(null, overlays).addTo(map);
```

```
// Here we create a legend control object.
var legend = L.control({
position: "bottomright"
});
// Then add all the details for the legend
legend.onAdd = function () {
let div = L.DomUtil.create("div", "info legend");
const deaths = [0, 1, 2, 3, 4];
const colors = [
    "red",
    "yellow",
    "blue",
    "green",
    "white"
];
// Looping through our intervals to generate a label with a colored square for each interval.
for (var i = 0; i < deaths.length; i++) {
    div.innerHTML +=
    "<i style='background: " + colors[i] + "'></i> " +
    deaths[i]+ (deaths[i + 1] ? "&ndash;" + deaths[i + 1] + "<br>" : "+");
}
return div;
};
// Finally, we add our legend to the map.
legend.addTo(map);
```

Lessons Learned the Hard Way

- MongoDB
  - Recreating database
- Flask
  - Building out flask app route (we had 5 different ways at one point)
- Flask -> Index -> JavaScript
  - Reading of data from flask to html to JS
- JavaScript
  - Reading JSON data into JS
  - Fixing coordinates in GeoJson (when we were using GeoJson..)
  - Difference between JS libraries and frameworks

The Final Product.....

Take it away Natalie!