CS225 Spatial Computing- Project Report

Mass Shootings & Gun Violence Analysis in USA

Team 20

Prudhvi Manukonda (SID:862325916)
Avinash Reddy Kummeta (SID : 862324723)
Venkata Vamsi Krishna Mupparaju (SID : 862324952)
Vaibhavi Mutya (SID : 862324892)
Aira Shrestha(SID: 862325662)

ABSTRACT

Mass shootings are one of the main issues in society. Unfortunately, there have been tremendous shootings all over the world especially in the United States. In this project we would explore mass shooting using an application. We would then observe the mass shootings on a map depending on the user inputs.

INTRODUCTION

The period from 2009 to 2021 saw the greatest number of mass shootings. Due to easy access to firearms, the gun homicide rate in the United States is 26 times greater than that of other high-income nations. Mass shootings occur more frequently in US states with fairly strict gun regulations and higher rates of gun ownership.

Although the majority of gun deaths in the country may be attributed to mass shootings, the facts reveal a more nuanced picture. Mass shootings are actually just the beginning of this nation's problem with gun violence. The majority (more than 99%) of gun fatalities in the US occur in shootings other than mass shootings. Mass shootings are mostly avoidable with evidence-based policy interventions, just like the daily gun violence that contributes to the more than 110 gun deaths each day in the US.

In a nationwide survey, 58% of individuals said that they or a loved one had ever been the victim of gun violence. Additionally, millions more people in the US experience gun violence in the form of being shot and hurt, being threatened with a gun, or witnessing an act of gun violence. The depth and variety of the survivor experience is closely proportional to how widespread the gun violence issue is in America. Gun violence can occur in a variety of ways, including deliberate and inadvertent shootings, killings and attacks with firearms, domestic violence with firearms, school shootings, police shootings, and other instances.

However, due to America's culture of silence around gun violence, we frequently

fail to discuss or completely comprehend the impact on survivors' lives.

1.1 PROBLEM STATEMENT

Between the years 2009 to 2020 there have been **281 Mass shootings** resulting in **1584 people shot and killed and 1054 shot and wounded,** these horrific scenes of Mass shootings have haunted the nation and flooded the anger and grief at this senseless preventable violence.Gun homicides in the USA are 26 **times higher** than other high income countries. We wanted to create an application to show this data visually in the usa map and see which state has more of these incidents occurrences. This visualization will give us a quick overview of violence in country wide and state governments can take appropriate actions to prevent them from happening

RELATED WORK

Academic Work

S.No	Paper/Article/website	Link
1	Spatial Modeling and Analysis of the Determinants of Property Crime in Portugal	https://www.mdpi.com/2220-9964/10/11/731
2	Patterns in the Locations of U.S. Mass Shootings-Keenen-Avonlea	https://shareok.org/handle/11244/323404
4	Dawei Wang ·Wei Ding ·Henry Lo ·Tomasz Stepinski ·Josue Salazar ·Melissa Morabit	https://www.researchgate.net/publication/257518350_Crime_hotspot_mapping_using_the_crime_related_factors_-_A_spatial_data_mining_

5	City-Wide Firearm Violence Spikes in Minneapolis following the Murder of George Floyd: A Comparative Time-Series Analysis of Three Cities	**https://www.mdpi.com/2413-8851/6/1/16/html**
9	This paper is about predicting whether or not a county would experience a mass shooting between 2014 and 2018 using logistic regression.	**https://shareok.org/handle/11244/323404**

Non-Academic Work

S.No	**Paper/Article/website**	**Link**
1	This article summarizes characteristics of mass shootings, including offender characteristics, types of firearm(s) used, and community-level correlates.	**https://www.rand.org/research/gun-policy/analysis/essays/mass-shootings.html**
2	This website analyses a compilation of 12 years (2009 to 2020) of original data on mass shootings in the United States, sourced from media reports and official police and court records. These records enable unique insights into the circumstances of mass shootings.	**https://everytownresearch.org/maps/mass-shootings-in-america/**
3	This website is used by Crime analysts to identify crime trends and patterns and help police agencies identify problems, allocate resources, and solve crimes using mapping and analytical methods such as hot spot analysis. ArcGIS is the industry standard for crime analysis technology and is critical to implementing evidence-based, data-driven crime reduction strategies.	**https://www.esri.com/en-us/industries/law-enforcement/strategies/crime-analysis**
4	The Gun Violence Archive, a nonprofit	

	research group that tracks gun violence using police reports, news coverage and other public sources, defines a mass shooting as one in which at least four people were killed or injured.	**https://www.gunviolencearchive.org/**
5	This website by Andrew Crooks highlights and examines the latest advances in the fields of geographical information science (GIS), agent-based modeling and other things along the way loosely constrained around urban system**s.**	**https://www.gisagents.org/2020/01/**
6	The purpose of this study is to gain some insights on mass shootings by performing spatial and statistical analysis on mass shooting data.	**https://nycdatascience.com/blog/student-works/r-visualization/data-analyzing-mass-shooting-us/**

2.1 TAXONOMY

Item set	**Contents in the Dataset**	**Models Used**
1	This paper investigates the spatial distribution of crimes against property in mainland Portugal with the primary goal of determining which demographic and socioeconomic factors may be associated with crime incidence in each municipality. For this purpose, Geographic Information System (GIS) tools were used to analyze spatial patterns, and different Poisson-based regression models were investigated, namely global models, local Geographically Weighted Poisson Regression (GWPR) models, and semi-parametric GWPR models	Poisson-based regression models, Geographically Weighted Poisson Regression (GWPR)
2	This study explores mass shooting locations as unique sites marked by tragedy with the goal of revealing patterns associated with mass shooting locations.It mainly answers two questions. Firstly, how is the county associated with the	Logistic Regression

	mass shooting. Secondly, how is the firearm policy related to the mass shooting. Regression model is used to do this.	
3	This chapter looks at 68 various crime keywords to help you figure out what kind of crime you are dealing with concerning geographical and temporal data. For categorizing crime into subgroups of categories with geographical and time aspects using news feeds, the Naive Bayes classification algorithm is used. For retrieving keywords from news feeds, the Mallet package is used. The hotspots in crime hotspots are identified using the K-means method. The KDE approach is utilized to address crime density and this methodology has solved the difficulties that the current KDE algorithm has	Advanced KDE, Kernel density estimation, K means clustering, is a nonparametric technique for estimating the probability density function of a random variable. KDE is used to smooth the point density
4	In this study, we introduce a new crime hotspot mapping tool—Hotspot Optimization Tool (HOT). HOT is an application of spatial data mining to the field of hotspot mapping. The key component of HOT is the Geospatial Discriminative Patterns (GDPatterns) concept,	Hot spot optimization tool

DATA COLLECTION, CLEANING AND PROCESSING

2.1 Data Collection

Inorder to provide precise information to the user regarding gun violence we need to collect various data from different websites and to include various incidents.In order to demonstrate the incidents on map we need information related to gun violence location coordinates and exact address of the incident for better visualization. Collected 3 datasets from various sources.Collected first data from https://everytownresearch.org/maps/mass-shootings-in-america/ this website which contains date, state,city,latitude,latitude,narrative, columns. Second dataset collected from https://www.theviolenceproject.org/mass-shooter-database/ website which contains date state,city,address but it doesn't have latitude and longitude values. Third dataset https://www.gunviolencearchive.org/query/14c00d51-0b8d-4dd6-a71e-0caa61f541

55 collected from the above website which contains 2000 samples of data in which date state,city,address as columns.

2.2 Data Cleaning and Processing

2.2.1 Data Cleaning and Processing on First Dataset

- Removed rows which have null values and have missing values in the dataset.
- Modified state column to full name from state abbreviation.
- Dropped all the unnecessary columns and took desired columns like “date”, “state”, “city”, “longitude”, “latitude”.

2.2.1.1 Code snippet for Data Cleaning and Processing on First Dataset

In [ ]:
```
#Cleaning DataSet1
```

In [ ]:
```
import pandas as pd
#Reading dataset1 from csv
df1 = pd.read_csv('dataset1.csv')
df1.head()
```

In [ ]:
```
#Inorder to convert two letter state abbrevation to full name stored those mapping in dictionary
us_state_to_abbrev = {
"Alabama": "AL","Alaska": "AK","Arizona": "AZ","Arkansas": "AR","California": "CA","Colorado": "CO", "Connecticut": "CT
"Maine": "ME","Maryland": "MD","Massachusetts": "MA", "Michigan": "MI", "Minnesota": "MN","Mississippi": "MS","Missouri
    "South Carolina": "SC",
    "South Dakota": "SD",
    "Tennessee": "TN",
    "Texas": "TX",
    "Utah": "UT",
    "Vermont": "VT",
    "Virginia": "VA",
    "Washington": "WA",
    "West Virginia": "WV",
    "Wisconsin": "WI",
    "Wyoming": "WY",
    "District of Columbia": "DC",
    "American Samoa": "AS",
    "Guam": "GU",
    "Northern Mariana Islands": "MP",
    "Puerto Rico": "PR",
    "United States Minor Outlying Islands": "UM",
    "U.S. Virgin Islands": "VI",
}

# invert the dictionary
abbrev_to_us_state = dict(map(reversed, us_state_to_abbrev.items()))
```

In [ ]:
```
#Converted two leter short cut state name into full name.
df1['State'] = df1['State'].map(lambda x: abbrev_to_us_state.get(x, x))
#Took desired columns from the whole dataset
final_df1 =df1[['Date', 'City' , 'State' , 'Latitude' , 'Longitude']]
#Dropping rows with missing values
final_df1.dropna()
```

In [ ]:
```
final_df1
```

2.2.2 Data Cleaning and Processing on Second Dataset

- Renamed “IncidentDate”, “County”, columns to “Date”, “ City”.
- Adding a new column “CompAddress” which was obtained by merging “Address”, “State”, “City”, USA.
- Modifying the date in the below format “ YYYY-MM-DD” for data consistency between the sets.
- Extracting required columns for geocoding “Date”, “City” , “State” , “CompAddress”.
- Downloaded the CSV file into local and geocoded the data using google sheets.
- After geocoding, read the geocoded csv file into the pandas framework.

2.2.1.1 Code snippet for Data Cleaning and Processing on Second Dataset

```
In [ ]: #Reading datase2 from csv
        df2 = pd.read_csv('dataset2.csv')
        #Renaming columns
        df2.rename(columns = {'Incident Date':'Date', 'City Or County':'City'}, inplace = True)
        df2.head()

In [ ]: #Extracting exact address by merging the coolumns
        df2['CompAddress']=df2['Address']+", "+df2['City']+", "+df2['State']+", USA"
        df2['PartialAddress']=df2['City']+", "+df2['State']+", USA"

In [ ]: #Converting date to "YYY-MM-DD" format
        df2['Date'] = pd.to_datetime(df2['Date'])

In [ ]: df2.head()

In [ ]: #Extracting required columns for geocoding
        finaldata2=df2[['Date', 'City' , 'State' , 'CompAddress','PartialAddress']]
        df3=df3.dropna(subset=['Address'])
        finaldata2.dropna()

In [ ]: finaldata2

In [ ]: #downloading finaldata2 to local using google sheets to use geocode api and to get corresponding latitude and longitutu
        finaldata2.to_csv('finaldata2.csv')

In [ ]: #After gecoding in googlesheets downloaded the "finaldata2 - finaldata2.csv" to local file and read the file from local

In [ ]: finaldata2 = pd.read_csv('finaldata2 - finaldata2.csv')

In [ ]: #In this database weh have latitude and longitude
        finaldata2.head()
```

2.2.3 Data Cleaning and Processing on Second Dataset

- Adding a new column “CompAddress” which was obtained by merging “Address”, “State”, “City”, USA.
- Renamed “IncidentDate”, “County”, columns to “Date”, “ City”.

- Dropping rows whose address column has na values.
- Modifying the date in the below format " YYYY-MM-DD" for data consistency between the sets.
- Extracting required columns for geocoding "Date", "City" , "State" , "CompAddress".
- Downloaded the CSV file into local and geocoded the data using google sheets.
- After geocoding, read the geocoded csv file into the pandas framework.

2.2.3.1 Code snippet for Data Cleaning and Processing on Third Dataset

```
In [ ]: #Cleaning dataset3

In [ ]: #Reading dataset from local
        df3 = pd.read_csv('datset3-b8243c3bcdfe.csv')
        df3.head()

In [ ]: df3.rename(columns = {'Incident Date':'Date', 'City Or County':'City'}, inplace = True)
        df3['CompAddress']=df3['Address']+", "+df3['City']+", "+df3['State']+", USA"
        df3['Date'] = pd.to_datetime(df3['Date'])
        #Dropping rows whose adress column as na values.
        df3=df3.dropna(subset=['Address'])
        finaldata3=df3[['Date', 'City' , 'State' , 'CompAddress']]
        finaldata3.head()

In [ ]: #downloading finaldata3 to local using google sheets to use geocode api and to get corresponding latitude and longitutu
        finaldata3.to_csv('finaldata3.csv')

In [ ]: #After gecoding in googlesheets downloaded the "finaldata2 - finaldata2.csv" to local file and read the file from local
        finaldata3=pd.read_csv('finaldata3 - finaldata3.csv')

In [ ]: #Merging three databases

In [ ]: final_df2=finaldata2[['Date', 'City' , 'State','Latitude','Longitude']]
        final_df3=finaldata3[['Date', 'City' , 'State', 'Latitude','Longitude']]

In [ ]: final_df1.head()

In [ ]: final_df2.head()

In [ ]: final_df3.head()
```

2.3 Dataset Attributes

Attribute	**Description**
Date	Date is in the YYYY-MM-DD format.
State	Denoted state in which incident happened in the postal form.

City	City or county name in which incident happened.
Address	Exact Address in which incident happened.
Latitude	Latitude of the address.
Longitude	Longitude of the address

3. FrontEnd

3.1 Input form and Map display:

In frontend we used **React JS framework.** We are utilizing react node modules available for visualizing google maps in the front end.

Below Code snippet is used to display Form with Date range and region section along with map

```
return (
    <div>
        <div className="row">
        <div className="col-lg-3">
    <h1> Welcome to Spatial </h1>
    <form onSubmit = {getLatLangPoints}>
        <Form.Group className="mb-3" controlId="formBasicEmail">
            <Form.Label >Select Dates</Form.Label>
            <DatePicker onDateSelected={setDate} />
        </Form.Group>
        <Form.Group className="mb-3" controlId="formBasicPassword">
                <Form.Label></Form.Label>
                <CountryDropDown onDropDownSelect = {allValues} />
        </Form.Group>

        <Button variant="primary" type="submit" onClick = {getDataFromLocalHost}>
            Submit
        </Button>
    </form>
    </div>
        <div className="col-lg-9">
            <MapMarkerComponent locations = {latlang} />
        </div>
    </div>
    </div>
)
```

This will show as below

3.2 Fetching Filtered Data from Api:

Below code snippet with function ***getDataFromLocalHost*** is used to get the filtered shootings data from api.

```
const getDataFromLocalHost = () => {
    const URL = 'http://localhost:9191/incidents'
    console.log({
        "state" :  state.value,
        "city" : city ,
        "orginalState" : state.label,
        "startDate" : startdate,
       "endDate" : enddate
   })
    fetch(URL , {
        method: 'POST',
        mode: 'cors',
        headers : {
            'Content-Type' : 'application/json'
        },
        body: JSON.stringify({
            "state" :  state.value,
            "city" : city,
            "orginalState" : state.label,
            "startDate" : startdate,
            "endDate" : enddate
        })
    })
   .then(response => response.json())
   .then(response => { console.log(response); setlatlang(response) } )
   .catch(err => console.error(err));

}
```

Response from api is passed to **setLatlang** function , which is used to plot the data in map using variable locations={latlang}

```
<MapMarkerComponent locations = {latlang} />
```

3.3 Map Generation :

We used react map node modules to generate maps and show data as clusters

```
"@react-google-maps/api";

import { MarkerClusterer } from "@react-google-maps/api";

import CustomMarker from "./CustomMarker";
```

mapFitBounds function will fit all the data points within the current view.

Then we pass the locations data in variable *location1* to Map module which will display the locations in clustered format

```
function mapFitBounds() {
  if (!map) return;
  const bounds = new window.google.maps.LatLngBounds();
  locations1.map((loc) =>
    bounds.extend(new window.google.maps.LatLng(loc.lat, loc.lng))
  );
  map.fitBounds(bounds);
}

if (map) {
  mapFitBounds();
}

}, [map]);

return (
  <div>
    <Map setMap={setMap}>
      <MarkerClusterer>
      {(clusterer) => (
        <>
        {locations1.map((loc) => (
          <CustomMarker
          key={loc.lat}
          position={loc}
          clusterer={clusterer}
          />
        ))}
        </>
      )}
      </MarkerClusterer>
    </Map>
  </div>
)
}
```

```
const renderMap = () => {

  const loadHandler = (map) => {
    props.setMap(map);
  };

  return (
      <div className='map'>
          <GoogleMap
              id="circle-example"
              zoom={6}
              center={defaultCenter}
              options={options}
              onLoad={loadHandler}
          >
              {props.children}
          </GoogleMap>
      </div>
  );
};
```

4.BackEnd

We are storing the dataset into a **mysql database** using mysql workbench.

Written backend code in **java** and **spring frameworks**. The main backend functionality is to take the input from the user and to search on the database to retrieve latitude longitudes which will be useful to plot in the frontend maps.

Code Snippets

In the below snippet, we have implemented two api , one for getting incidents based on filtered range, other is to get all the data

```
prudhvi *
@RestController
@CrossOrigin
public class SpatialProjectController {

    2 usages
    @Autowired
    private SpatialProjectService service;



    prudhvi *
    @PostMapping("/incidents")
    public List<Location> findProducts(@RequestBody FilterRequest r) {
        return service.getResults(r);
    }

    new *
    @GetMapping("/allIncidents")
    public List<DataSet1> findAllProducts(@RequestBody FilterRequest r) {
        return service.getAllResults();
    }
```

Internally for each end point , we are calling service to perform sql queries from data base and return the output.

Below code shows how and service is calling the functions to filter the data(latitudes and longitudes) from database based on input parameters in the user form

```
        else{
            if(city!=null){
                if(state!=null){
                    li1= repository1.findIncidentsByStateAndCity(state.toLowerCase(),city.toLowerCase());
                }
                else{
                    li1= repository1.findIncidentsByCity(city.toLowerCase());
                }
            }
            else{
                if(state!=null){
                    li1=repository1.findIncidentsByState(state.toLowerCase());
                }
                else{
                    li1= repository1.findAll();
                }
            }
        }
        List<Location> l1=li1.stream().map(l->convertEntityToLocation(l)).collect(Collectors.toList());
        return l1;
    }

    1 usage   new *
    public List<DataSet1> getAllResults(){
        return repository1.findAll();
    }

    1 usage   prudhvi
    public Location convertEntityToLocation(DataSet1 set1){
        Location l=new Location(set1.getLatitude(),set1.getLongitude());

        return l;
    }
```

Below code snippet perform the all the sql queries to filter from data set

```
2 usages  prudhvi *
public interface DataSet1Repository extends JpaRepository<DataSet1,Integer> {

    1 usage  prudhvi
    @Query(value="select * from new_data_set p where p.date> :date1 and p.date < :date2", nativeQuery = true)
    List<DataSet1> findIncidentsByDate(@Param("date1")Date date1, @Param("date2")Date date2);

    1 usage  prudhvi
    @Query(value="select * from new_data_set p where lower(p.city)= :city", nativeQuery = true)
    List<DataSet1> findIncidentsByCity(@Param("city")String city);

    1 usage  prudhvi *
    @Query(value="select * from new_data_set p where lower(p.state)= :state ", nativeQuery = true)
    List<DataSet1> findIncidentsByState(@Param("state")String state);

    1 usage  prudhvi
    @Query(value="select * from new_data_set p where p.date> :date1 and p.date < :date2 and lower(p.city) = :city", nativeQuery = true)
    List<DataSet1> findIncidentsByDateAndCity(@Param("date1")Date date1, @Param("date2")Date date2, @Param("city")String city);

    1 usage  new *
    @Query(value="select * from new_data_set p where lower(p.state)= :state  and lower(p.city) = :city", nativeQuery = true)
    List<DataSet1> findIncidentsByStateAndCity(@Param("state")String state,@Param("city")String city);

    1 usage  prudhvi
    @Query(value="select * from new_data_set p where p.date> :date1 and p.date < :date2 and lower(p.state)= :state", nativeQuery = true)
    List<DataSet1> findIncidentsByStateAndDate(@Param("date1")Date date1, @Param("date2")Date date2, @Param("state")String state);
    1 usage  prudhvi
    @Query(value="select * from new_data_set p where p.date> :date1 and p.date < :date2 and lower(p.state)= :state and lower(p.city)= :city", nativeQuery = true)
    List<DataSet1> findIncidentByAll(@Param("date1")Date date1, @Param("date2")Date date2, @Param("state")String state, @Param("city")String city);
}
```

Visualization and Results

We used react-google-maps node module to for visualization

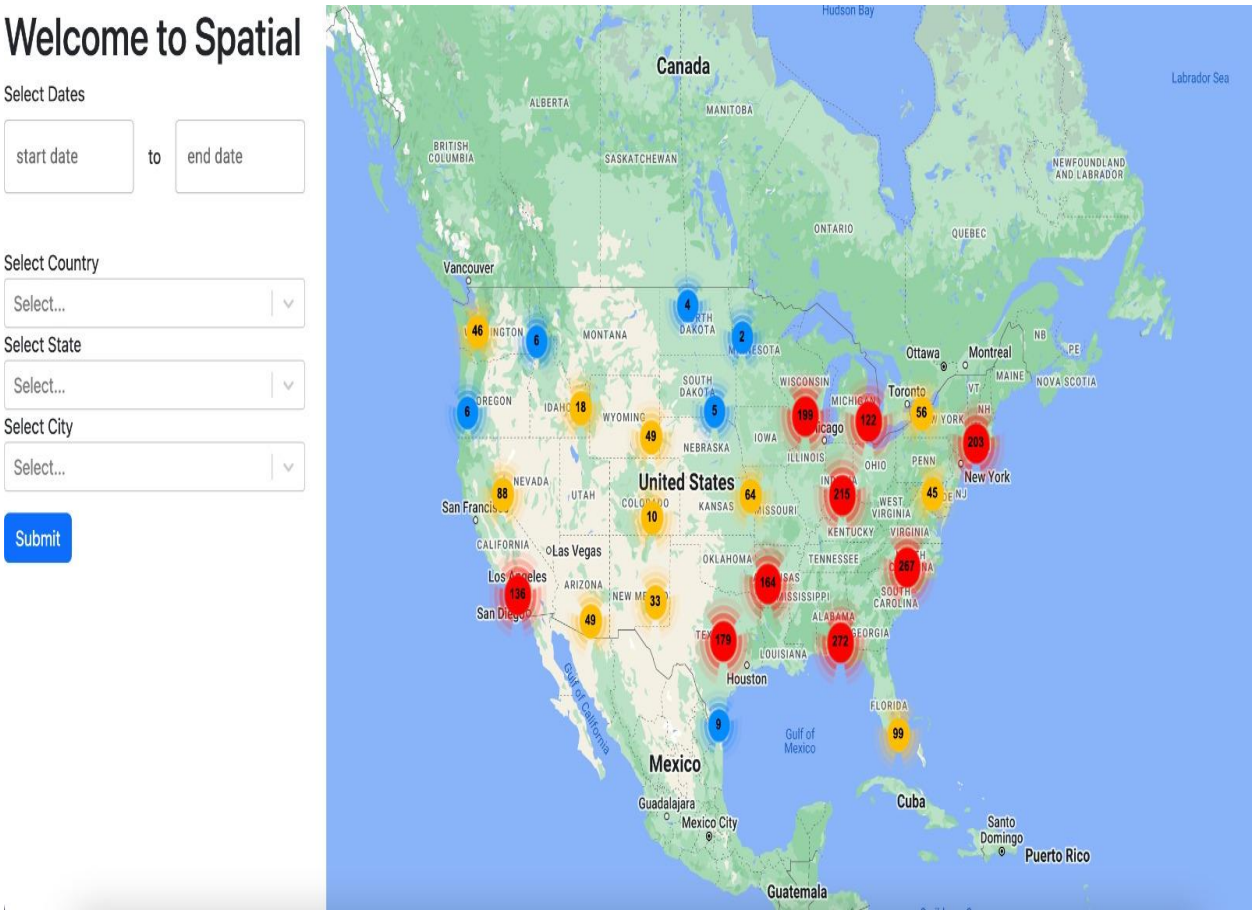
```
import {
 GoogleMap,
 GoogleMapProps,
 useLoadScript
} from "@react-google-maps/api";
```

```
{ MarkerClusterer } from "@react-google-maps/api";
```

MarkerCluster will convert the lat-long points into cluster representation in the map.

Below figure shows the data of all records from **all states in usa** in cluster representation

Case -1: all states in usa



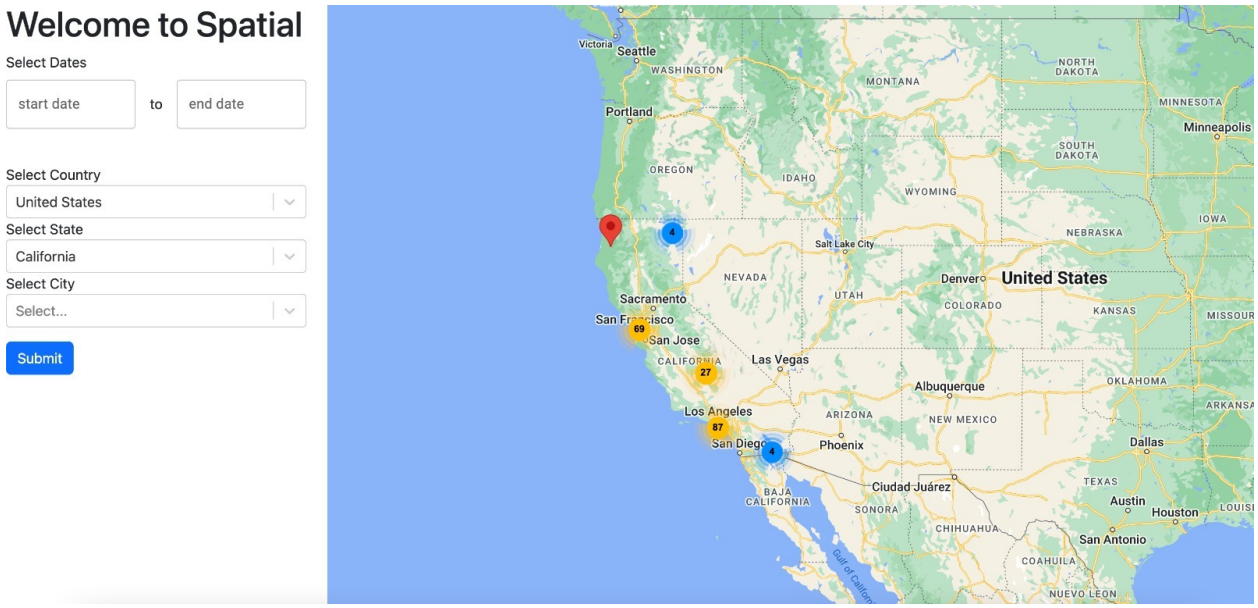
Red -Cluster -Higher number of gun crime incidents

Yellow -Cluster- Moderate number of gun crime incidents

Blue -Cluster -Lower number of gun crime incidents

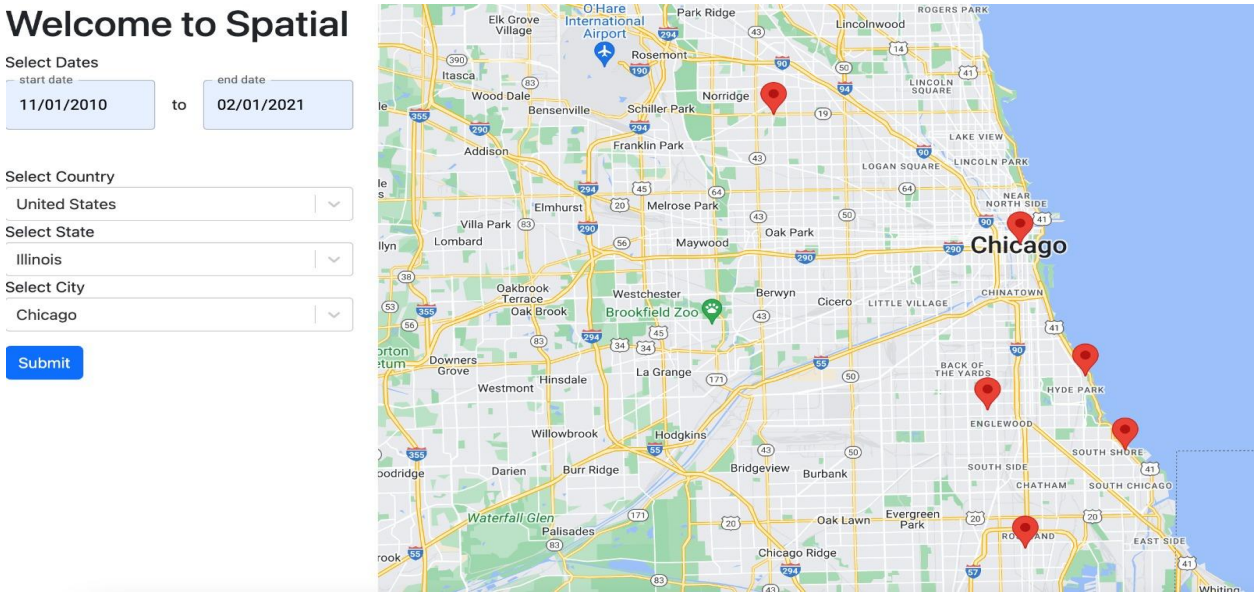
Below figure shows the results of single state (California)

Case -2: single state (California)



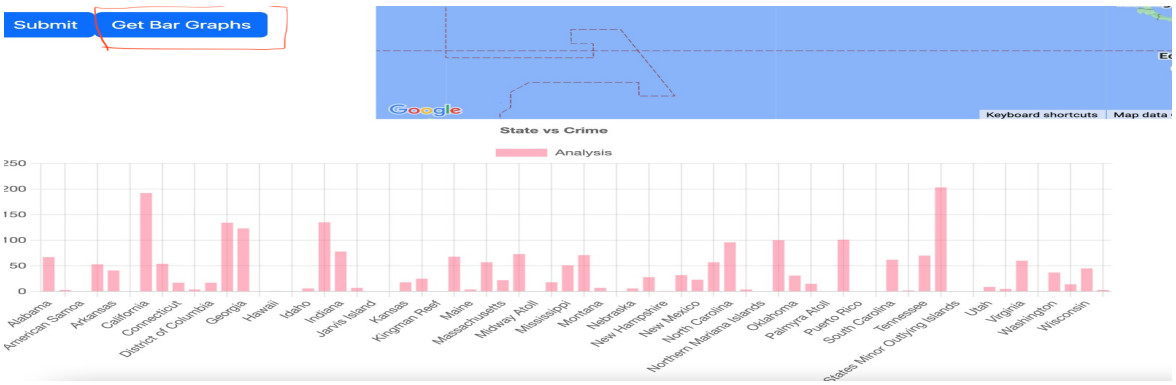
In the below figure Results shown for when search for Specific City Filtering

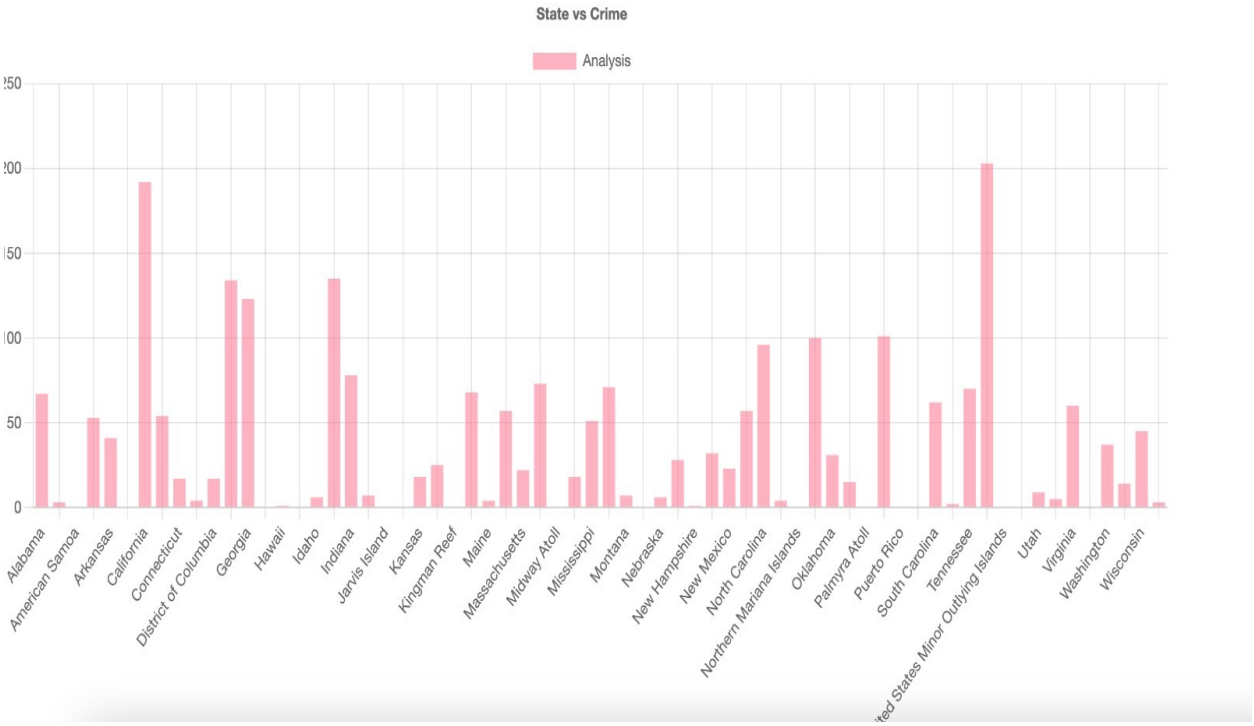
Case -3: specific city (Chicago)



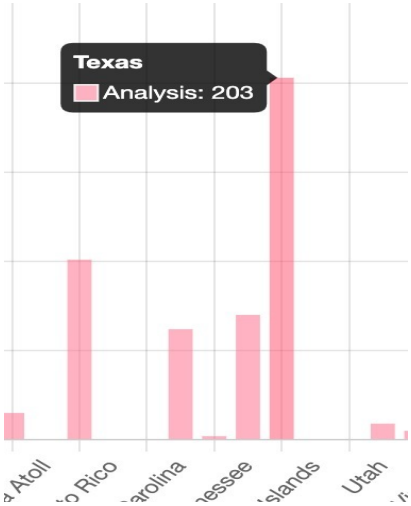
Analysis

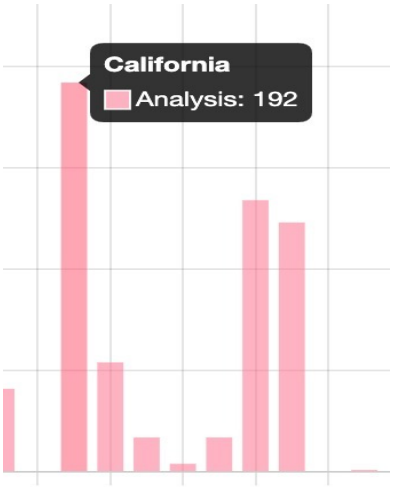
After clicking BarGraph button,





From the bar chart above it is observed that **Texas** and **California** has recorded most number of Gun violence incident in usa





Future Scope

This application can be extended to analyze huge data sets as we are using sql server in backend and writing custom Apis. Several Visualization features can be added as we are using react components , this application is very modular and easily scaled.

Currently, we work on USA data sets , but this application can extend to use world wide data sets from all countries, and also in future it can be extended to add Machine learning algorithms as separate components for analytics modules to identify patterns and underlying reasons for increased gun violence.