San Diego Vacation Safety

By: Matthew Vale, Sara Bagherikashkooli, & Alia Tirado

Project Theme

Data Wrangling Techniques

- Jupyter Notebook and imported Pandas to help clean the data

- Created a Path to the folder holding our datasets

```
In [1]: import csv
        import json
        import pandas as pd

        from pathlib import Path

In [2]: file = Path('../Resources/Original Datasets/Airbnb Listings/listings.csv')

In [3]: df = pd.read_csv(file)

In [4]: df.head(15)
```

Out[4]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	lon
0	6	Home in San Diego · ★4.81 · 3 bedrooms · 3 bed...	29	Sara	NaN	North Hills	32.75522	-117.
1	29967	Bungalow in San Diego · ★4.78 · 2 bedrooms · 3...	129123	Michael	NaN	Pacific Beach	32.80751	-117.
2	54001	Guesthouse in La Jolla · ★4.94 · 2 bedrooms · ...	252692	Marsha	NaN	La Jolla	32.81301	-117.
		Guesthouse						

Data Wrangling Techniques

- Displayed a list of column names

- Chose column names with relevant data for our visualization

- Dropped columns that didn't have the room_type we were looking for

In [5]:
```
df.columns
```

Out[5]:
```
Index(['id', 'name', 'host_id', 'host_name', 'neighbourhood_group',
       'neighbourhood', 'latitude', 'longitude', 'room_type', 'price',
       'minimum_nights', 'number_of_reviews', 'last_review',
       'reviews_per_month', 'calculated_host_listings_count',
       'availability_365', 'number_of_reviews_ltm', 'license'],
      dtype='object')
```

In [6]:
```
airbnb_df = df[["neighbourhood", "latitude", "longitude", "room_type", "price"]]
airbnb_df
```

Out[6]:

	neighbourhood	latitude	longitude	room_type	price
0	North Hills	32.755220	-117.128730	Entire home/apt	349
1	Pacific Beach	32.807510	-117.257600	Entire home/apt	286
2	La Jolla	32.813010	-117.268560	Entire home/apt	196
3	Pacific Beach	32.805830	-117.242440	Entire home/apt	99
4	Pacific Beach	32.806210	-117.233720	Entire home/apt	114
...	...	...	...	...	...
12196	Core	32.718861	-117.158751	Entire home/apt	152
12197	Clairemont Mesa	32.806893	-117.168541	Entire home/apt	186
12198	Pacific Beach	32.800856	-117.255359	Entire home/apt	140
12199	Allied Gardens	32.797100	-117.094438	Entire home/apt	408
12200	East Village	32.720777	-117.155181	Entire home/apt	130

12201 rows × 5 columns

In [7]:
```
unique = airbnb_df["room_type"].unique()
unique
```

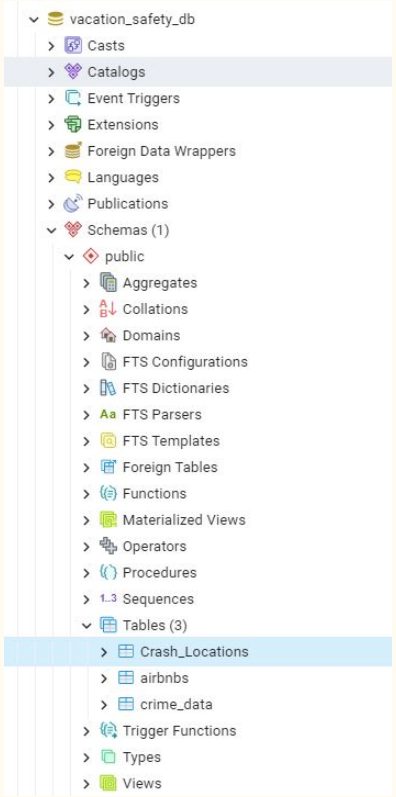
Out[7]:
```
array(['Entire home/apt', 'Private room', 'Shared room', 'Hotel room'],
      dtype=object)
```

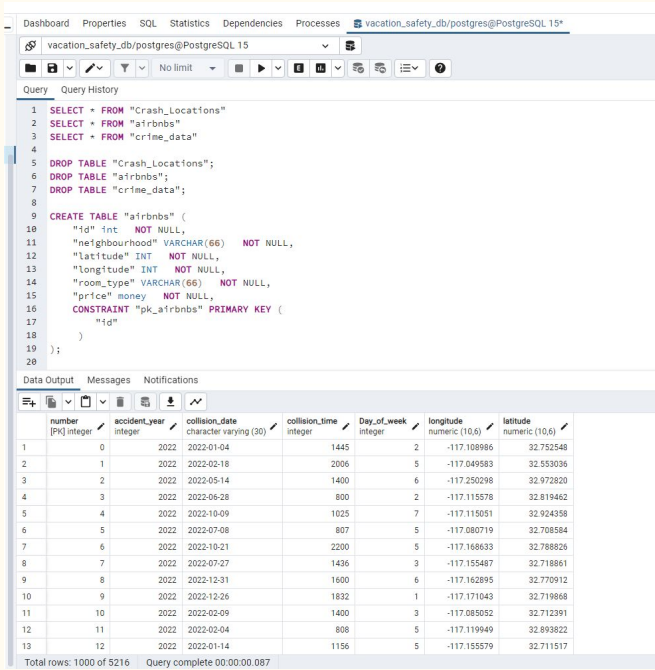
In [8]:
```
entire_home_df = airbnb_df.drop(airbnb_df[airbnb_df['room_type'] != 'Entire home
entire_home_df
```

Postgresql

- Created the schema for each table we wanted for the database

- Imported the csvs into each data table and used SELECT * FROM to make sure each table imported correctly





Flask API

- Uses SQLAlchemy
- Creates routes to specific tables in our PostgreSQL database
- Returns in JSON format

```python
@app.route('/')
def index():
    return render_template('index.html')

# Defines a route to get the airbnb data
@app.route('/airbnb')
def airbnb_data():

    try:
        # Query data from the airbnb table in the database
        airbnb_data = Airbnb.query.all()

        # Puts the data into a list of dictionaries
        results = [{'neighbourhood': data.neighbourhood, 'latitude': data.latitude, 'longitude': data.longitude, '
        print(results)

        return jsonify(results)
    except Exception as e:
        return jsonify({'error': str(e)}), 400
```

Java Script

- Leaflet map and OpenStreetMap positioned at San Diego's coordinates

- Fetch the data from the API Endpoint from our Flask API /PostgreSQL database for the markers

- Adds pop ups when user hovers over markers

```
// Gets a Leaflet map and points it towards San Diego
var map = L.map('map').setView([32.7157, -117.1611], 10);
var markers = L.markerClusterGroup();

// Gets OpenStreeMap for base layer
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
  attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
}).addTo(map);

// Defines API endpoint for Airbnb data
const apiEndpoint = '/airbnb';


// Function that fetches the airbnb table data and creates markers
function fetchDataAndCreateMarkers() {
  fetch(apiEndpoint)
    .then((response) => response.json())
    .then((data) => {
      data.forEach(item => {
        // Creates markers
        var marker = L.marker([item.latitude, item.longitude]);

        // Adds things to the pop up, such as price and neighbourhood name then adds to mark cluster group
        marker.bindPopup(`<strong>${item.price}</strong><br>Neighbourhood: ${item.neighbourhood}`);
        markers.addLayer(marker);

      });

      // Add the marker cluster layer to the map
      map.addLayer(markers);

      console.log(data);
    })
    .catch((error) => {
      console.error('Error fetching data:', error);
    });
```

Java Script

- User selects a column and event listener is used to fetch the data from that selected column

- Bar chart is created with plotly based on the data from the selected column

```
// Gets the column select element from DOM
const columnSelect = document.getElementById('column-select');

// Event listner
columnSelect.addEventListener('change', (event) => {
  const selectedColumn = event.target.value;
  fetchCrimeDataAndCreateBarChart(selectedColumn);
});

// Function that fetches the crime_data table data for the user selected column
function fetchCrimeDataAndCreateBarChart(selectedColumn) {

  // Defines this API endpoint that will fetch the data from the selected column
  const secondapiEndpoint = `/crime?column=${selectedColumn}`;
  fetch(secondapiEndpoint)
    .then((response) => response.json())
    .then((data) => {
      const labels = data.map((item) => item.Crime);
      const values = data.map((item) => item[selectedColumn]);

      const barData = [{
        x: labels,
        y: values,
        type: 'bar',
      }];

      const layout = {
        title: `Crime Data for ${selectedColumn} from July 2022 to July 2023`,
        xaxis: { title: 'Crime Type'},
        yaxis: { title: selectedColumn}
      };

      // Use Plotly to create the bar graph
      Plotly.newPlot('bar-chart', barData, layout);
    })
    .catch((error) => {
      console.error('Error fetching crime data:', error);
    });
}
```

JS Library: Google Charts

- Pie data is fetched. Labels are created and the occurrence of the days of the week are counted

- Loads Google Charts library, and with the counted occurrences and label titles, it draws the Pie Chart

```
// Function that fetches the data from the crime_data table
function fetchPieDataCreatePieChart() {
  // Defines API endpoint for the Crash_Locations table data
  const piechartEndpoint = '/car_crash';
  fetch(piechartEndpoint)
    .then((response) => response.json())
    .then((data) => {

      // Creates the headers for the pie chart
      const chartData = [['Day', 'Count']];
      // Needed for the sidebar to show what each number represents and to count the number of times the day is found in the column
      const dayLabels = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"];
      const dayCounts = {
        Monday: 0,
        Tuesday: 0,
        Wednesday: 0,
        Thursday: 0,
        Friday: 0,
        Saturday: 0,
        Sunday: 0,
      };

      // Iterates through the column to count the amount of times the number that represents the day of the week appears
      data.forEach((item) => {
        const dayOfWeek = dayLabels[item.Day_of_week - 1];
        dayCounts[dayOfWeek]++
      });

      // Turns dayCounts into an array and appends to chartData
      Object.entries(dayCounts).forEach(([dayOfWeek, count]) => {
        chartData.push([dayOfWeek, count]);
      });

      // Used to load Google Charts js library and calls the GooglePieChart function
      google.charts.load('current', {'packages': ['corechart']});
      google.charts.setOnLoadCallback(() => GooglePieChart(chartData));
    })
    .catch((error) => {
      console.error('Error fetching pie chart data:', error);
    });
}
```

HTML

- Imports Plotly.js and Google Chart.js, Leaflet CSS, Leaflet.markercluster CSS, etc.

- Body holds the divs for the map, bar chart, and pie chart

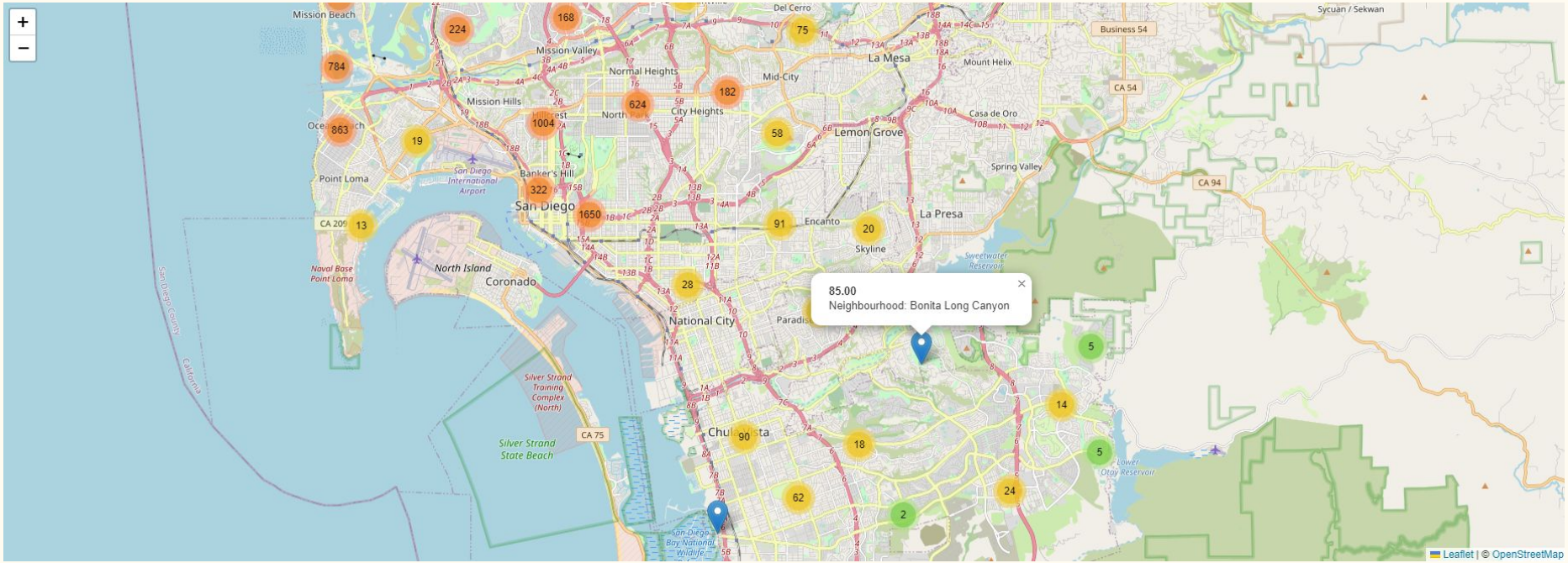
```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>San Diego Safety Trip</title>
  <!-- For the Plotly.js library -->
  <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>

  <!-- For the Google Chart.js library -->
  <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>

  <!-- Leaflet CSS -->
  <link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css"
    integrity="sha256-p4NxAoJBhIIN+hmNHrzRCf9tD/miZyoHS5obTRR9BMY="
    crossorigin="" />

  <!-- Leaflet.markercluster CSS -->
  <link rel="stylesheet" type="text/css" href="https://unpkg.com/leaflet.markercluster@1.0.3/dist/MarkerCluster.css">
  <link rel="stylesheet" type="text/css" href="https://unpkg.com/leaflet.markercluster@1.0.3/dist/MarkerCluster.Default.css">

  <!-- Our CSS -->
  <link rel="stylesheet" type="text/css" href="static/css/style.css">
</head>

<body>

  <!-- The div that holds our map -->
  <div id="map"></div>
```

CSS

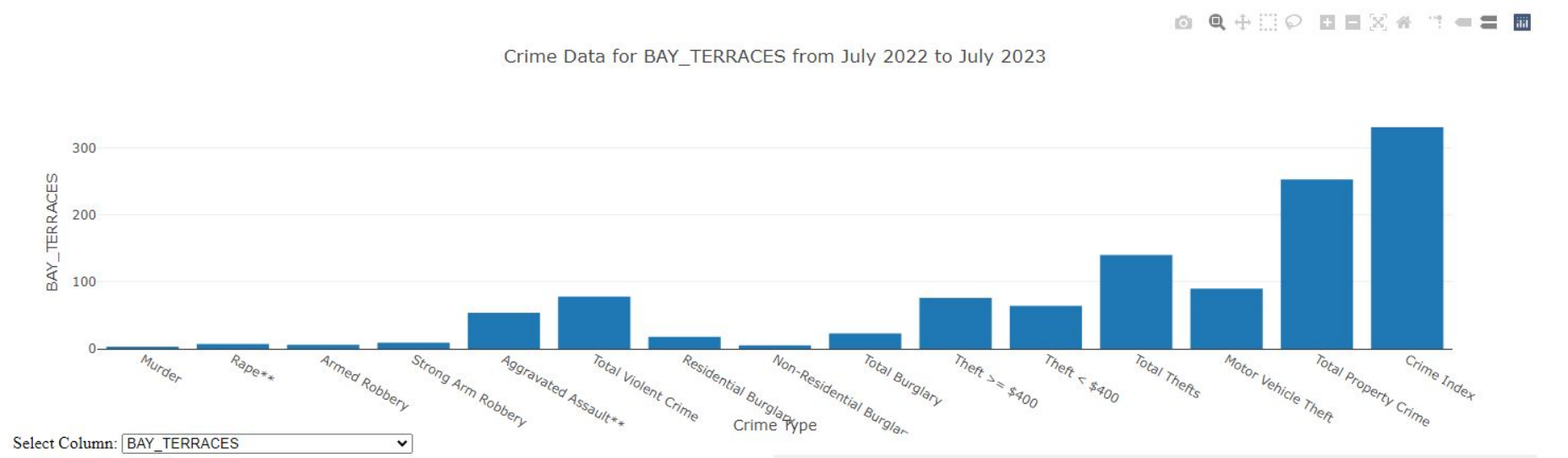
- Visual styles for the map, bar chart, and pie chart

```
body {
  padding: 20;
  margin: 20;
  }
  html, body, #map {
  height: 90%;
  width: 100vw;
  }

#bar-chart{
  width: 75%;
  height: 400px;
  margin: 20px auto;
  background-color: #f0f0f0;
}

#google-pie-chart {
  width: 75%;
  height: 400px;
  margin: 20px auto;
  background-color: #f0f0f0;
}
```

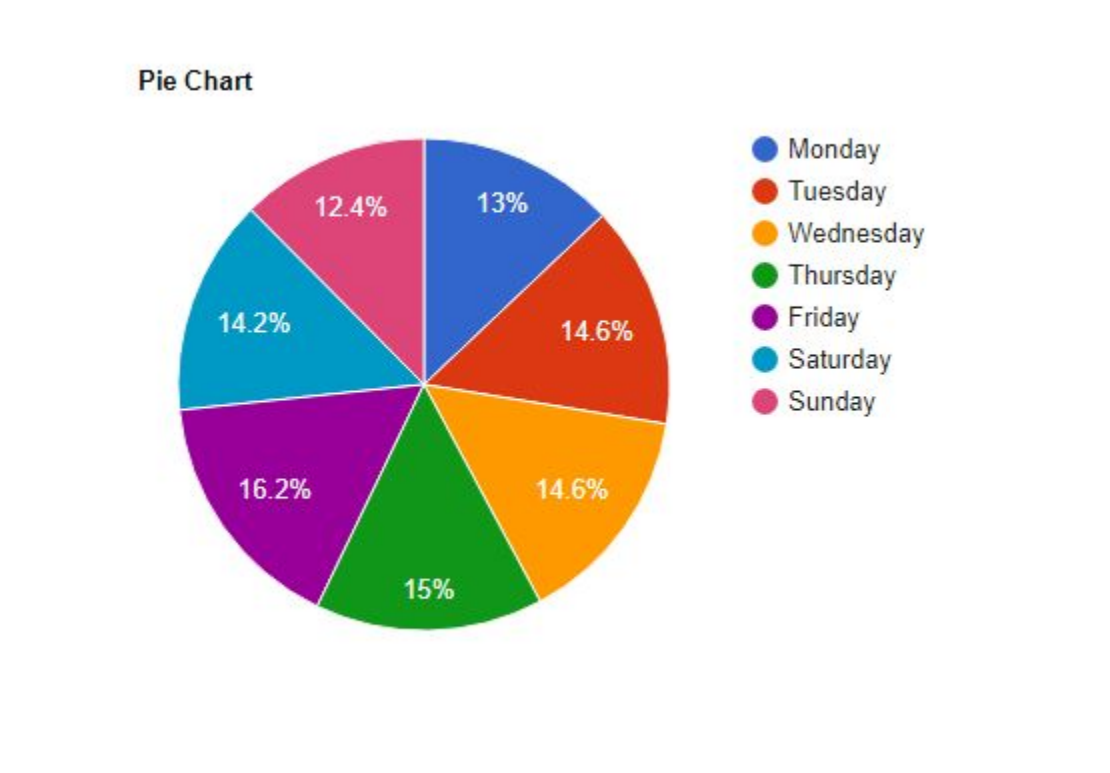
Final Visualization Demo



Final Visualization Demo



Final Visualization Demo



Questions?