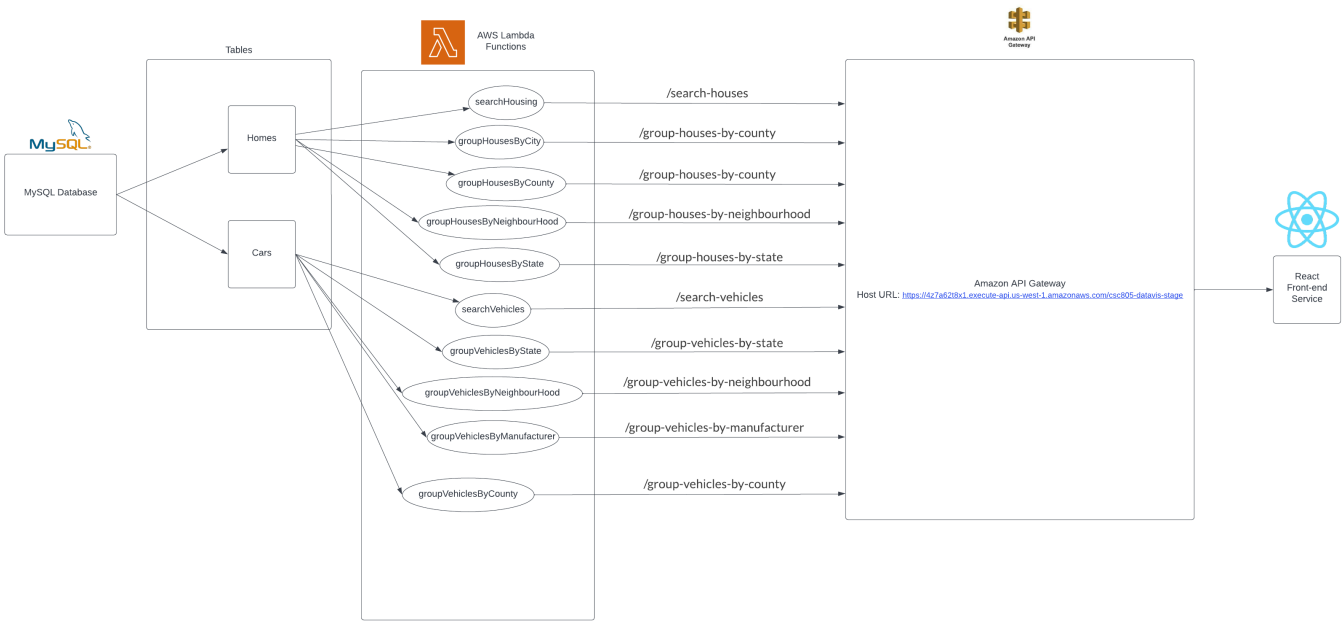
Project Report

Member 1: Kshitiz Sareen
Member 2: Aarshil Patel

System Architecture



Dataset Description

There are two datasets used in this application. The first dataset is a housing dataset called USA Housing Listings, and the second is called the Used Cars Dataset.

Here are the descriptions of the datasets:

1. **USA Housing Listings**: This dataset consists of all the houses for rent posted on Craigslist. There are 400,000 records in this dataset, and each record consists of 22 columns. Most of these columns don't provide any meaning to the data. However, some columns are required to give sense to this data. The relevant columns are:
    1. Price: Integer
    2. Housing Type (Categorical Data) : [String] = ["Apartment", "Condo", "House", "Duplex", "Townhouse", "Loft", "Manufactured", "Cottage/Cabin", "Flat", "In-Law", "Land", "Assisted Living"]
    3. Square Feet: Integer
    4. No. Of Beds: Integer
    5. No. Of Baths: Integer
    6. Cats Allowed: Boolean
    7. Dogs Allowed: Boolean
    8. Smoking Allowed: Boolean
    9. Wheelchair Access: Boolean
    10. Electric Vehicle Charge: Boolean
    11. Comes Furnished: Boolean

12. Latitude: Double
13. Longitude: Double

This dataset was cleaned by dropping all values which had an empty or null value in the columns of price, housing type, square feet, and No. Of beds, No. Of baths, latitude, and longitude.

After cleaning the dataset, we generated the address from the longitude and latitude using Google Maps API. We extracted the formatted address, neighborhood, city, county, and state of the record in the housing data. We created extra columns in the dataset to easily visualize our data. We created these additional columns in the dataset:

1. Address: String (This column stores the formatted address of each record in the dataset)
2. Neigborhood: String (This column stores the neighborhood of each record in the dataset)
3. City: String (This column stores the city of each record in the dataset)
4. County: String (This column stores the county of each record in the dataset)
5. State: String (This column holds the state of each record in the dataset)

2. **USA Vehicle Listings:** This dataset consists of data on all the vehicles for sale posted on Craigslist. There are 500,000 records in this dataset, and each record consists of 26 columns. Most of these columns don't provide any meaning to the data. However, some columns are required to give sense to this data. The relevant columns are:
    1. Price: Integer
    2. Year: Integer
    3. Mileage: Integer
    4. Manufacturer: String (The Manufacturer of the Vehicle)
    5. Fuel Type : [String] (Categorical values of the Fuel Type of the Vehicle) = ["Gas", "Diesel", "Hybrid", "Electric", "Other"]
    6. Vehicle Types : [String] (Categorical values of the Vehicle Type of the Vehicle) = ["Truck", "Pickup", "Bus", "Coupe", "Mini-van", "SUV", "Sedan", "Offroad", "Van", "Convertible", "Hatchback", "Wagon", "Other"]
    7. Latitude: Double
    8. Longitude: Double

    This dataset was cleaned by dropping all values with an empty or null value in the columns of price, year, mileage, manufacturer, fuel type, vehicle type, latitude, and longitude.

    After cleaning the dataset, we generated the address from the longitude and latitude using Google Maps API. We extracted the formatted address, neighborhood, city, county, and state of the record in the housing data. We created extra columns in the dataset to easily visualize our data. We created these additional columns in the dataset:

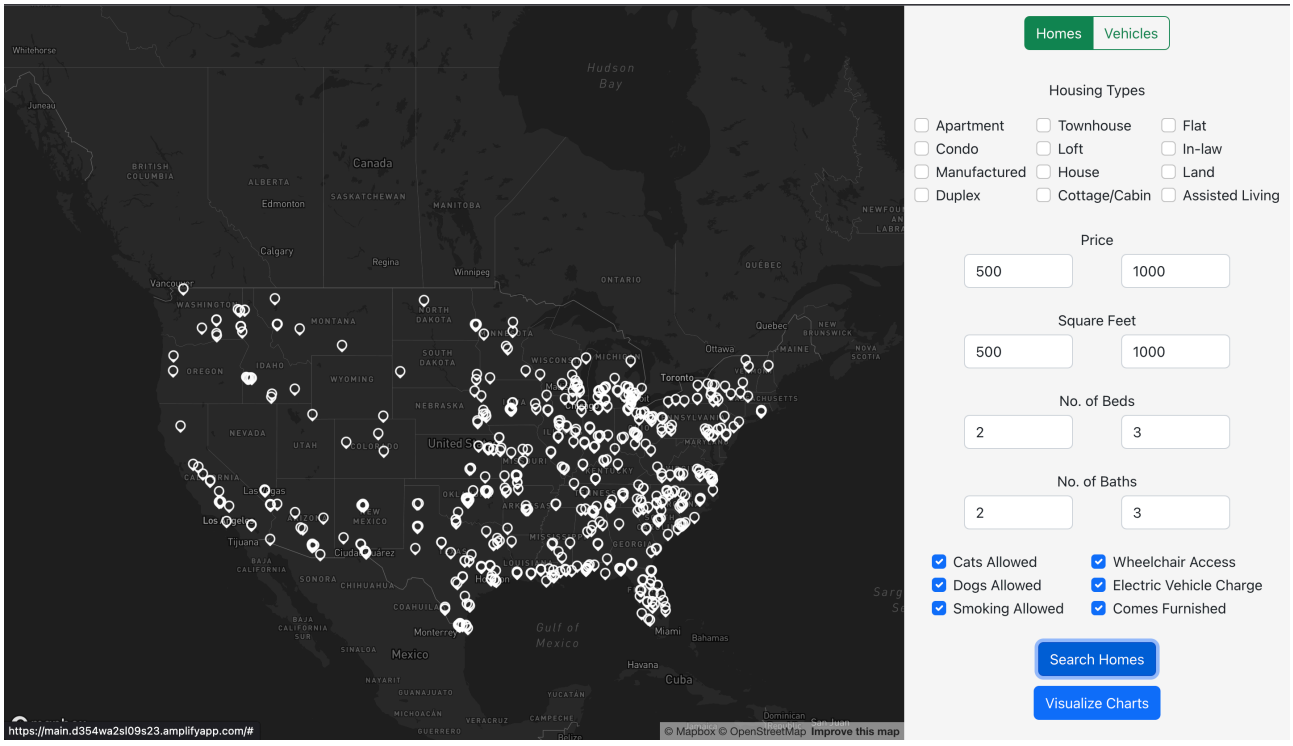
    1. Address: String (This column stores the formatted address of each record in the dataset)
    2. Neigborhood: String (This column stores the neighborhood of each record in the dataset)
    3. City: String (This column holds the city of each record in the dataset)
    4. County: String (This column stores the county of each record in the dataset)
    5. State: String (This column holds the state of each record in the dataset)

System Description

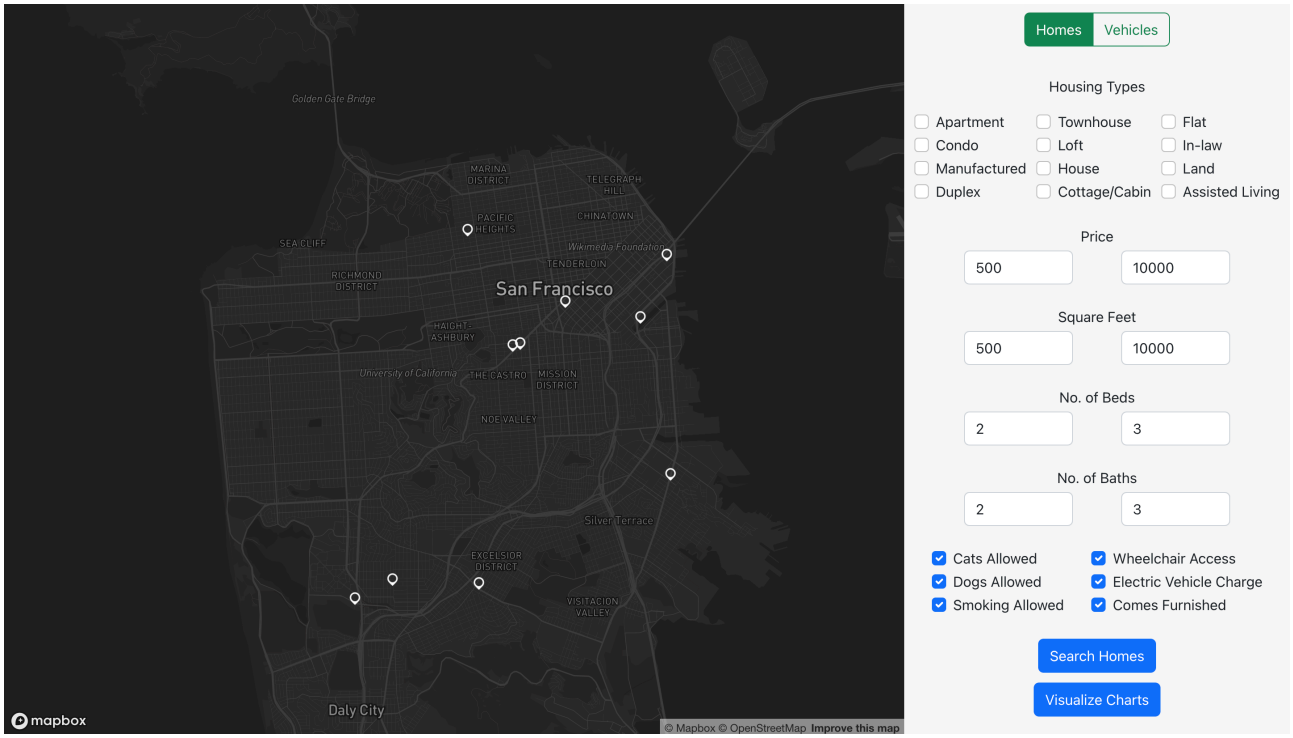
1. MySQL Database: - We created a MySQL database that consists of two tables:
    1. Houses: This table contains all the housing data records.
    2. Vehicles: This table contains all the data for vehicle listings.
2. AWS Lambda: - We created ten lambda functions for the backend service. We made the lambda functions using the Go programming language. The lambda functions are listed as follows:
    1. searchHousing: This lambda function searches through all the houses in the Housing table of the MySQL database.
    2. groupHousesByCity: This lambda function is used to group houses by the city specified in their address.
    3. groupHousesByCounty: This lambda function is used to group houses by the County specified in their address.
    4. groupHousesByNeighbourHood: This lambda function is used to group houses by the neighborhood specified in their address.
    5. groupHousesByState: This lambda function is used to group houses by the state they specified in their address
    6. searchVehicles: This lambda function searches through all the vehicles in the vehicles table of the MySQL database.
    7. groupVehiclesByCity: This lambda function is used to group Vehicles by the city specified in their address
    8. groupVehiclesByCounty: This lambda function is used to group Vehicles by the County specified in their address.
    9. groupVehiclesByNeighbourHood: This lambda function is used to group Vehicles by the neighborhood specified in their address.
    10. groupVehiclesByState: This lambda function is used to group Vehicles by the state specified in their address.
3. AWS API Gateway: - We created an API with the following base: https://4z7a62t8x1.execute-api.us-west-1.amazonaws.com/csc805-datavis-stage. This API acts as a trigger for our AWS lambda functions. Here are the routes for each AWS lambda function.
    1. /search-houses: searchHousing.
    2. /group-houses-by-city: groupHousesByCity.
    3. /group-houses-by-county: groupHousesByCounty.
    4. /group-houses-by-neighbourhood: groupHousesByNeighbourHood
    5. /group-houses-by-state: groupHousesByState
    6. /search-vehicles: searchVehicles.
    7. /group-vehicles-by-state: groupVehiclesByState
    8. /group-vehicles-by-neighbourhood: groupVehiclesByNeighbourHood.
    9. /group-vehicles-by-Manufacturer: groupVehcilesByManufacturer.
    10. /group-vehicles-by-county: groupVehiclesByCounty.
4. React: We created our front end using React. Our React application interacts with our backend service using the API we created using Amazon API Gateway. We made our visualizations using D3.js.

Link to Video Demo:

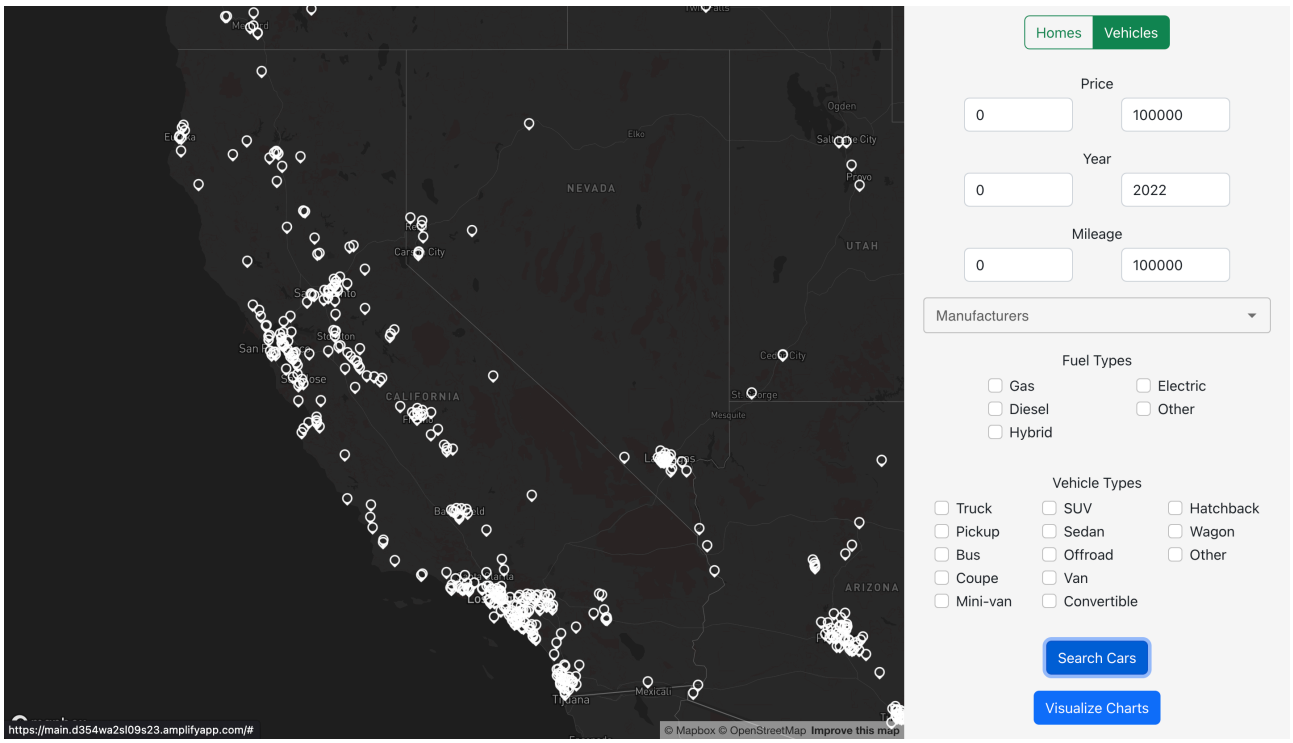
Screenshots



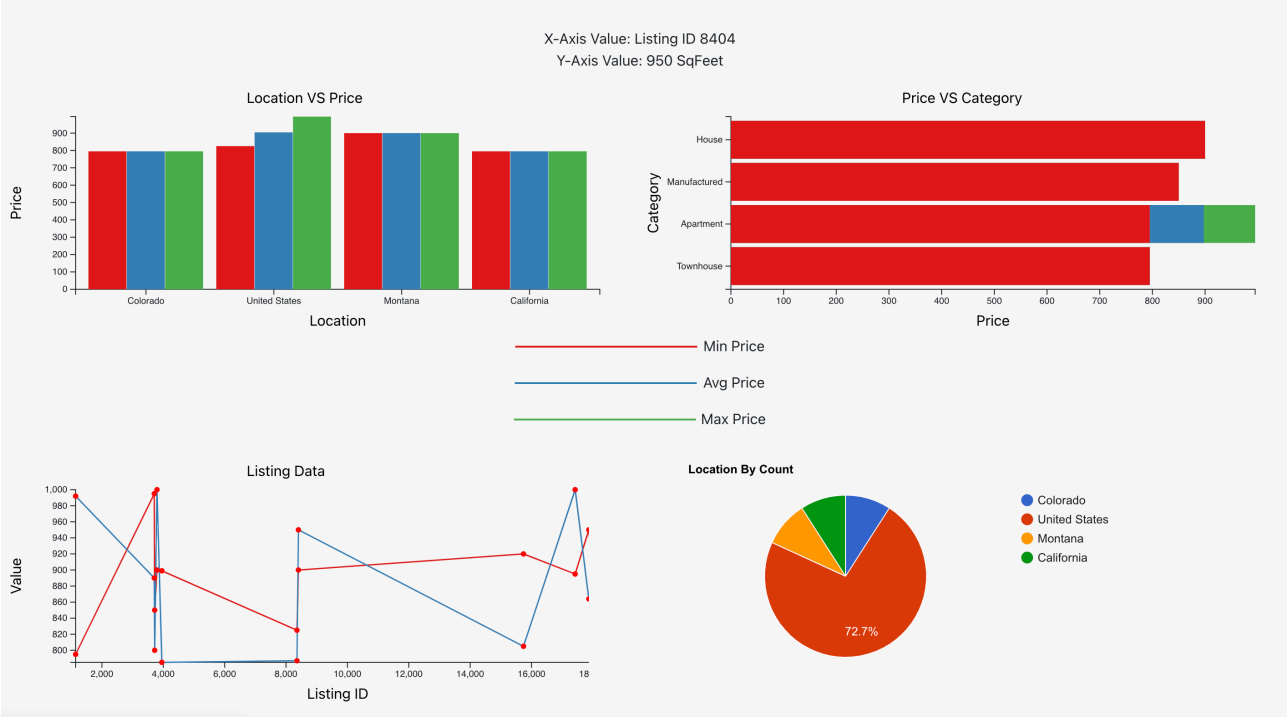
In this screenshot, we can see all the homes that satisfy all the conditions specified in the right panel. In this case, we can see all the homes represented in the map have a price between 500 and 1000, square feet between 500 and 1000, no. of beds between 2 and 3, no. of baths between 2 and 3.



All homes for rent in San Francisco.



All cars for sale in California.

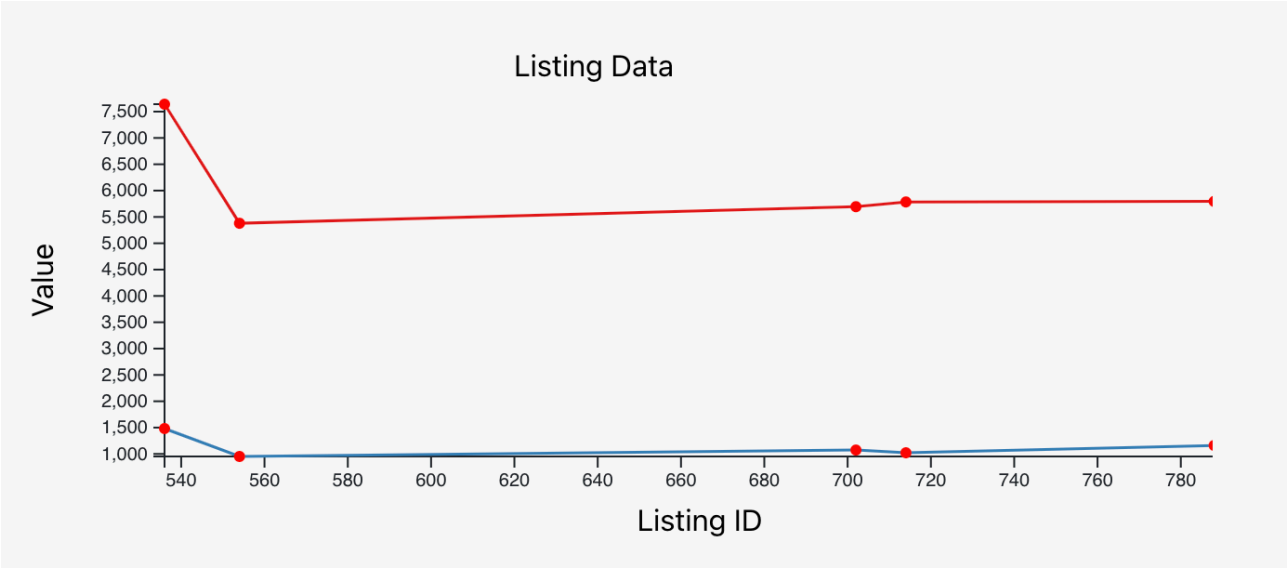


In this screenshot, we can see four different types of charts:

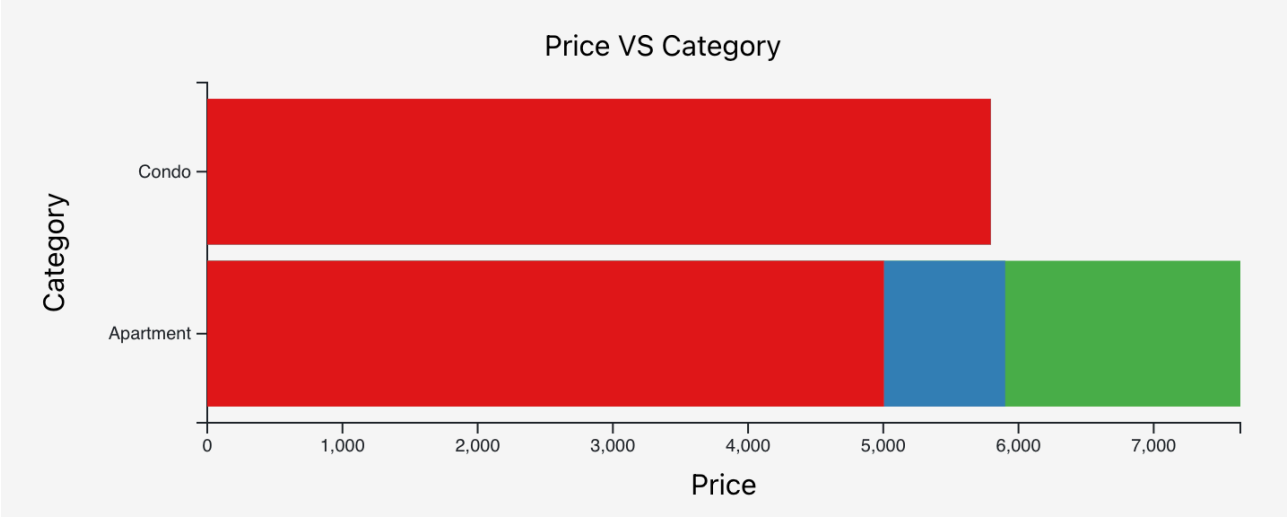
1. Horizontal Bar Chart
2. Stacked Bar Chart
3. Multi-series Line Chart
4. Pie Chart

In this chart we can see the average price, minimum price, maximum price vs the location of the homes. The red bar represents the minimum price, the blue bar represents the average price, and the green price represents the maximum price of the homes represented on the map.

The red bar represents the minimum price, the blue bar represents the average price, and the green price represents the maximum price of the homes represented on the map.

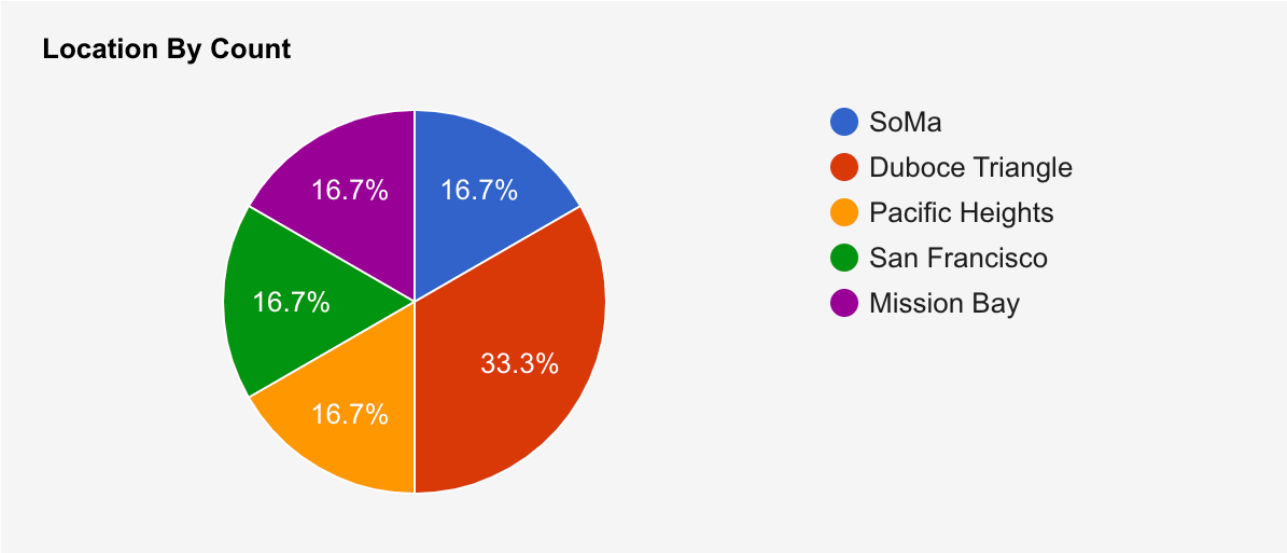


In this chart, we can see the individual listings of each house that is represented on the map. In this case the blue line represents the square feet, and the redline represents the price.



In this chart we can see the average price, minimum price, maximum price vs the type of the homes. The red bar represents the minimum price, the blue bar represents the average price, and the green price represents the maximum price of the homes represented on the map.

The red bar represents the minimum price, the blue bar represents the average price, and the green price represents the maximum price of the homes represented on the map.



In this chart, we can see the number of homes in each state.

Github Link

https://github.com/KshitizSareen/CSC-805-Data-Visualization-Project