**Grazioso Salvare MongoDB Dashboard**

**Project Overview**

This project is part of the development of a dashboard for Grazioso Salvare, an international animal rescue training company. The dashboard integrates with a MongoDB database containing animal shelter data and allows users to filter, visualize, and map rescue dogs based on specific traits that match their training profiles. This dashboard was developed using the Dash framework for the front end, and MongoDB for data management.

**Features**

* **Interactive Filter Options**: Filter the data table and charts by rescue type (Water Rescue, Mountain/Wilderness Rescue, or Disaster/Individual Tracking).
* **Interactive Data Table**: Displays detailed information about animals based on the selected filter, including name, breed, age, and more.
* **Pie Chart Visualization**: Displays the distribution of breeds in the dataset. Breeds accounting for less than 1% are grouped under an "Other" category.
* **Geolocation Map**: Displays the location of animals on the map. Selecting a row in the table updates the map to show the animal’s location, name, and breed.

**Tools and Technologies Used**

* **Dash**: A Python framework for building analytical web applications.
* **MongoDB**: Used as the back-end database for storing animal data.
* **Plotly Express**: For generating interactive data visualizations (the pie chart).
* **Dash Leaflet**: For creating the interactive map.
* **JupyterDash**: Used for running Dash applications in a Jupyter Notebook environment.
* **Pandas**: For handling and manipulating data within Python.

**Why MongoDB?**

MongoDB is a flexible, NoSQL database well-suited for working with semi-structured data like the shelter outcomes dataset. It allows us to quickly store and retrieve large amounts of data, with easy-to-implement CRUD operations. Its compatibility with Python through pymongo makes it a great choice for this application.

**Why Dash Framework?**

Dash is an open-source framework built on top of Flask, Plotly, and React.js, making it ideal for building data visualizations that integrate seamlessly with data from the MongoDB database. It allows for the rapid creation of interactive, data-driven applications.

**Dashboard Features and Usage**

The dashboard consists of three key components:

1. **Interactive Filter Options**: The user can select from one of three rescue types or reset the filter to view the entire dataset. The available filters are:
   * Water Rescue
   * Mountain/Wilderness Rescue
   * Disaster/Individual Tracking
   * Reset
2. **Interactive Data Table**: The data table dynamically updates based on the selected filter and displays detailed information about the animals, including:
   * Animal name
   * Breed
   * Location (latitude and longitude)
   * Age, and more.
3. **Pie Chart**: Displays the distribution of animal breeds in the dataset. Breeds with less than 1% of the population are grouped under "Other" to prevent cluttering the chart. The tooltip on each slice of the pie displays the breed name and the total number of animals in that breed.
4. **Geolocation Map**: A map shows the location of the selected animal from the data table. The marker displays the animal's name and breed.

**Steps to Use the Dashboard**

1. **Launch the Application**:
   * Run the JupyterDash app in a Jupyter Notebook or deploy it as a web application.
   * The dashboard will load with the full dataset displayed by default.
2. **Apply Filters**:
   * Select one of the rescue types (Water, Mountain/Wilderness, or Disaster) to filter the data table and charts accordingly. The map will update to show the location of the first animal in the filtered list.
3. **Explore the Data**:
   * Hover over the pie chart to see the breed distribution, including the total number of animals in each breed.
   * Click on a row in the data table to update the map and view the selected animal's location, name, and breed.

**Screenshots**

1. **Initial View of the Dashboard (Reset)**:
   * Screenshot of the dashboard in its unfiltered state.
   * **A screenshot of a map

     Description automatically generated**Should display the full dataset in the table, the pie chart with all breeds, and the map centered on a default location.
2. **Filtered View (Water Rescue)**:
   * Screenshot after selecting "Water Rescue" in the filter options.
   * Should display only breeds related to water rescue, with the pie chart updating to reflect the filtered breeds.
   * **A screenshot of a computer

     Description automatically generated**The map should update to show the first animal in the filtered list.
3. **Filtered View (Mountain/Wilderness Rescue)**:
   * Screenshot after selecting "Mountain/Wilderness Rescue" in the filter options.
   * Should display only breeds related to mountain or wilderness rescue, with the pie chart updating to reflect the filtered breeds.
   * **A close-up of a map

     Description automatically generated**The map should update to show the first animal in the filtered list.
4. **Filtered View (Disaster/Individual Tracking)**:
   * Screenshot after selecting "Disaster/Individual Tracking" in the filter options.
   * Should display only breeds related to disaster or individual tracking, with the pie chart updating to reflect the filtered breeds.
   * **A screenshot of a map

     Description automatically generated**The map should update to show the first animal in the filtered list.

**How to Reproduce the Project**

**Prerequisites**

* **Python 3.x**
* **MongoDB** (You can install locally or use a cloud instance)
* **Libraries**: Install required libraries using the following command:

pip install dash jupyter\_dash dash\_leaflet pandas plotly pymongo

**Steps**

1. **Set Up MongoDB**:
   * Import the provided dataset into your MongoDB database using the mongoimport tool or directly through the MongoDB shell. Ensure you create the correct user credentials for accessing the database.
   * Example import command:

mongoimport --db AAC --collection animals --file aac\_shelter\_outcomes.csv --jsonArray

1. **Set Up the Python Environment**:
   * Ensure the pymongo driver is installed to allow Python to interact with MongoDB.
   * Run the provided Python script in a Jupyter Notebook or as a standalone Dash web application.
2. **Run the Application**:
   * Execute the app, and the dashboard will be available for use. You can test the filter functionality, explore the dataset, and visualize the breed distribution and animal locations.

**Challenges Faced**

**Layout Adjustments**

Fine-tuning the layout of the dashboard to ensure the pie chart, data table, and map displayed correctly together was a key challenge. Several iterations were made to optimize the spacing and alignment.

**Filtering Logic**

The filtering logic needed to account for not only specific breeds but also grouping smaller categories into an "Other" category for clarity. This required dynamic aggregation of breed counts below a certain threshold.