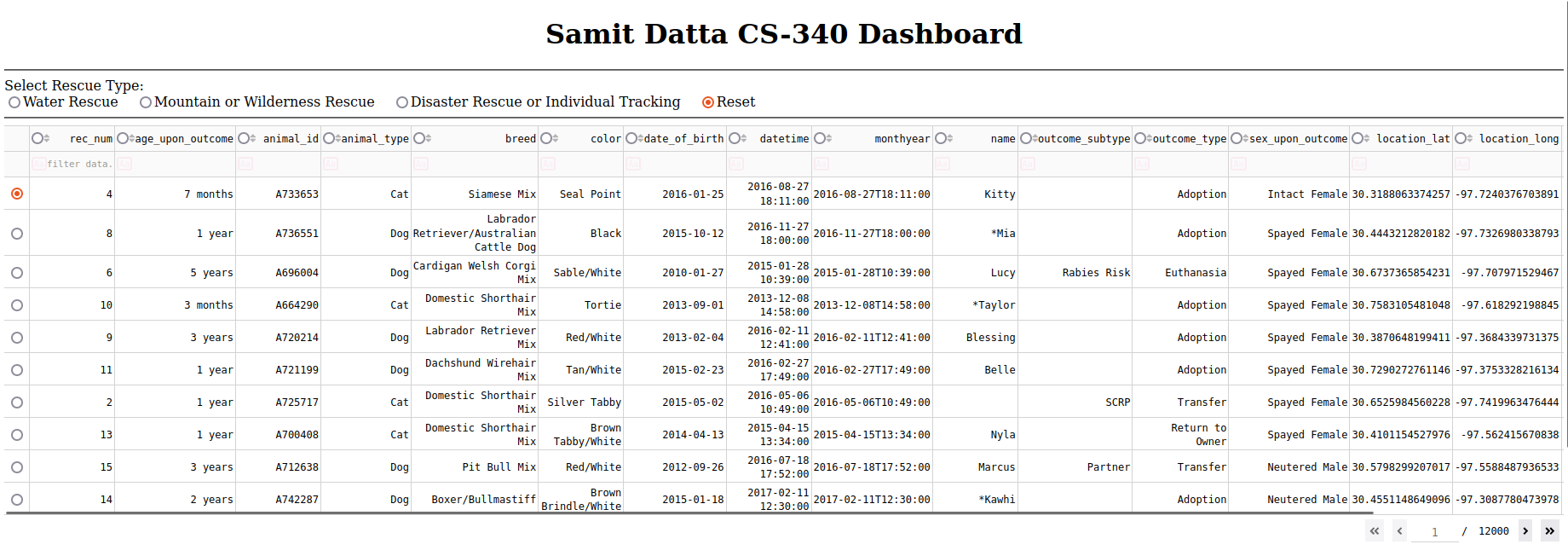
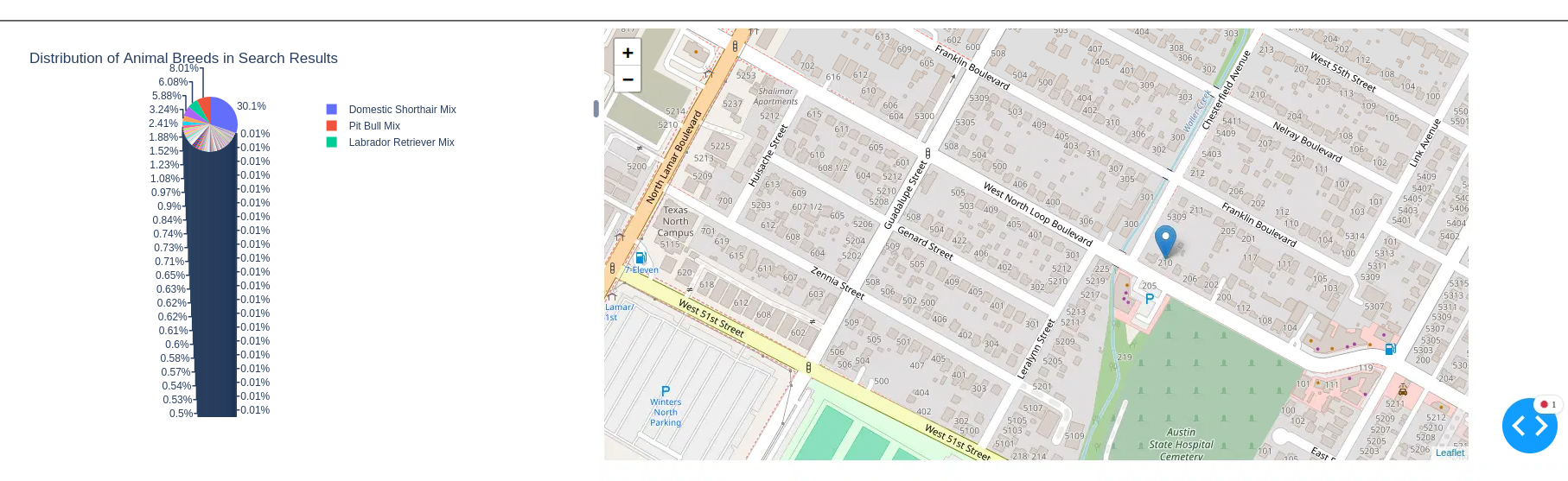
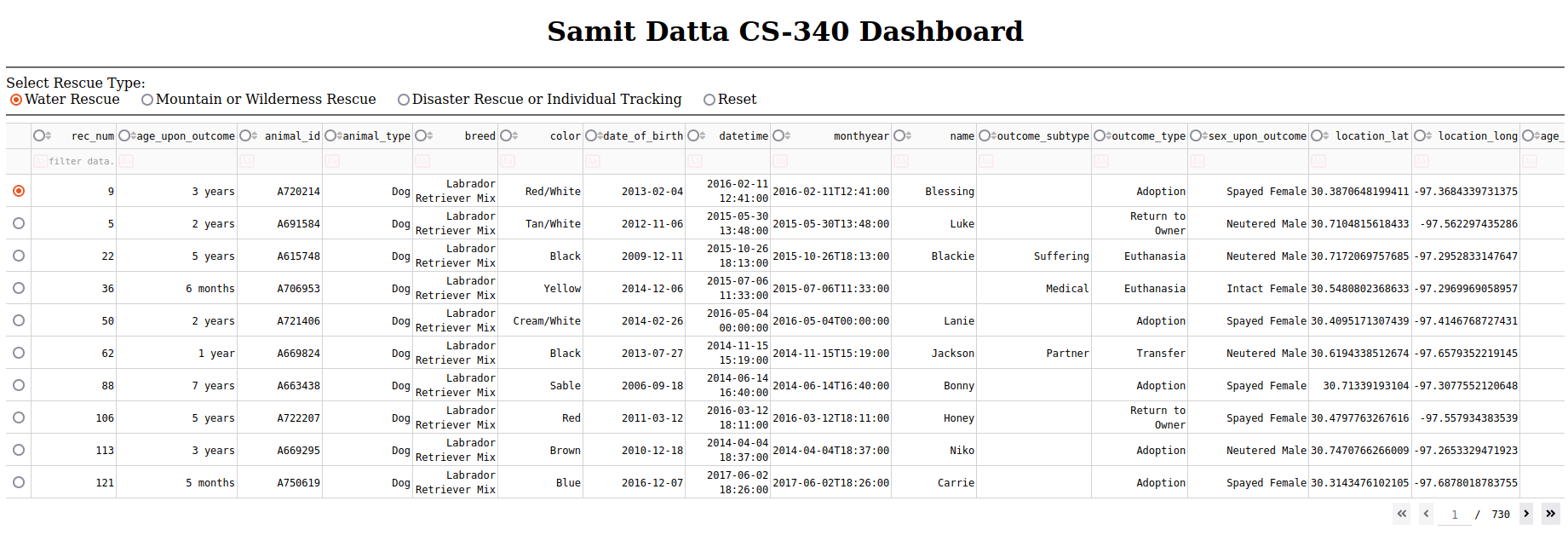
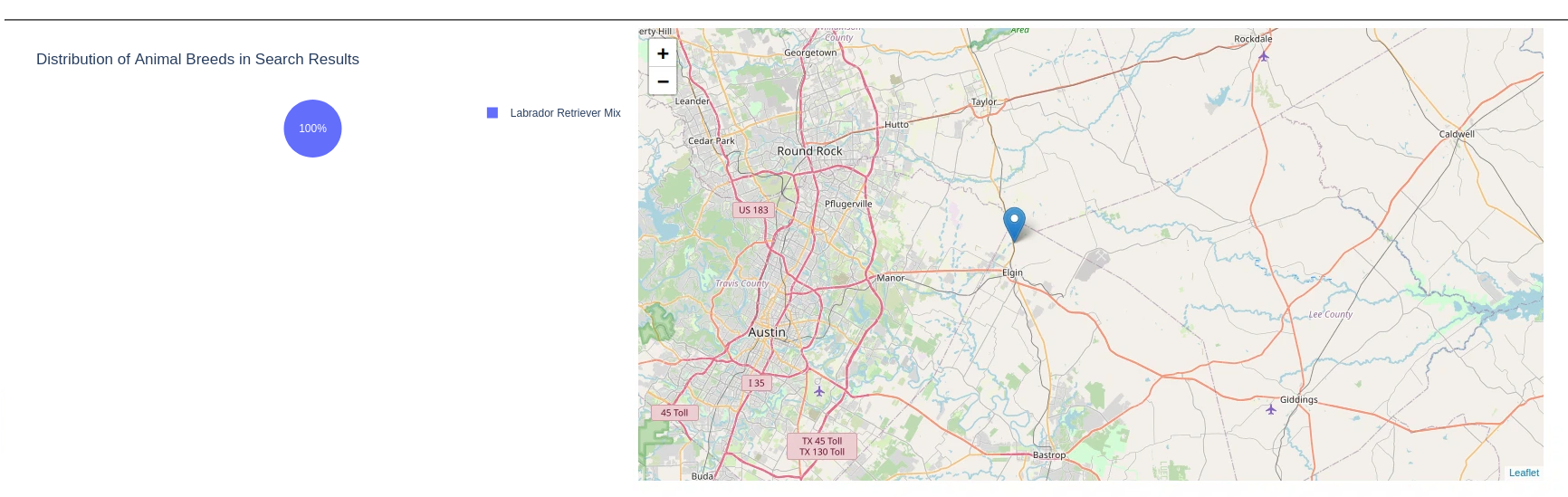
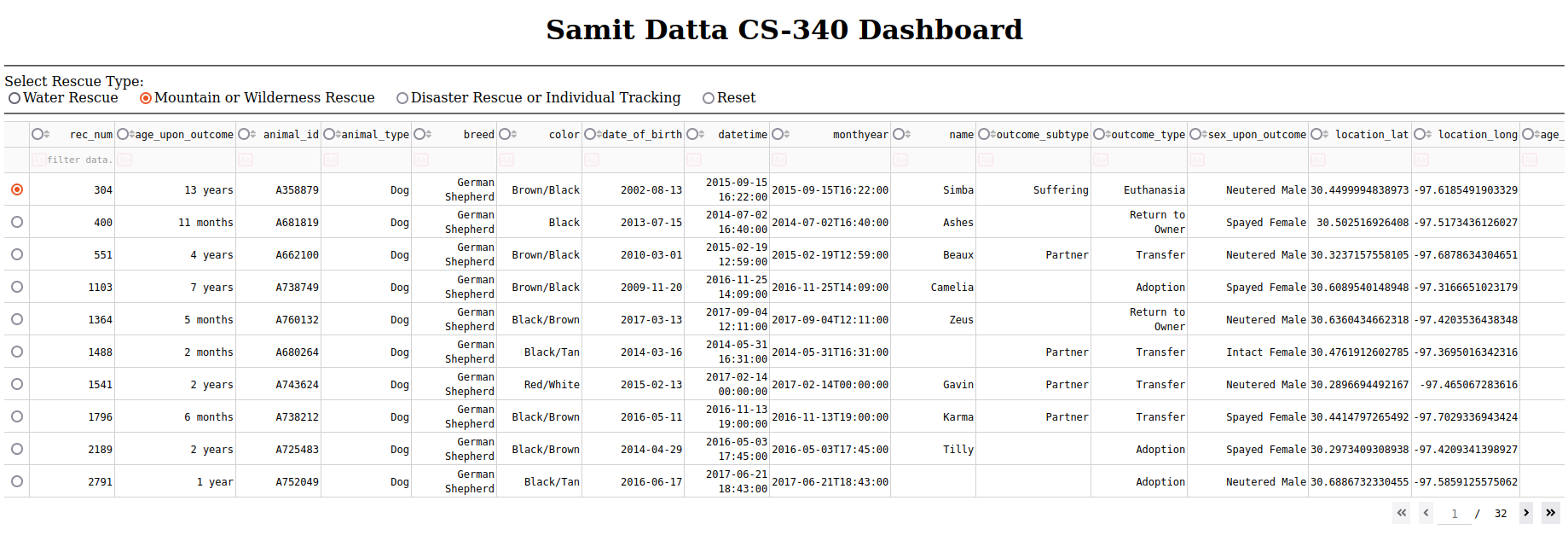
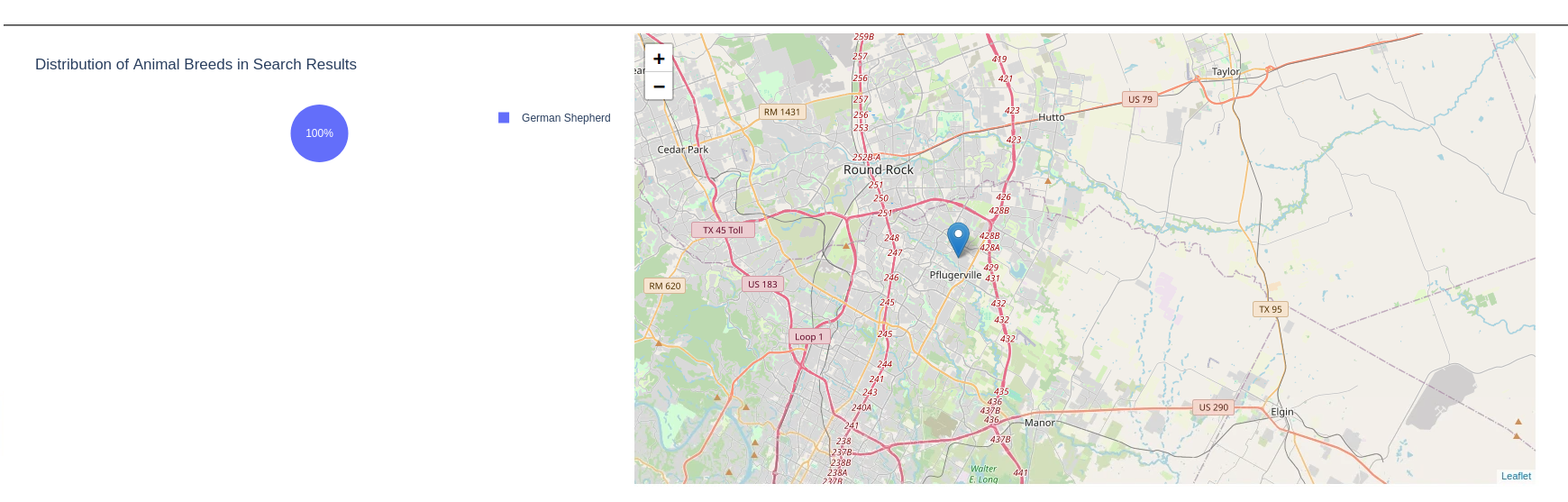
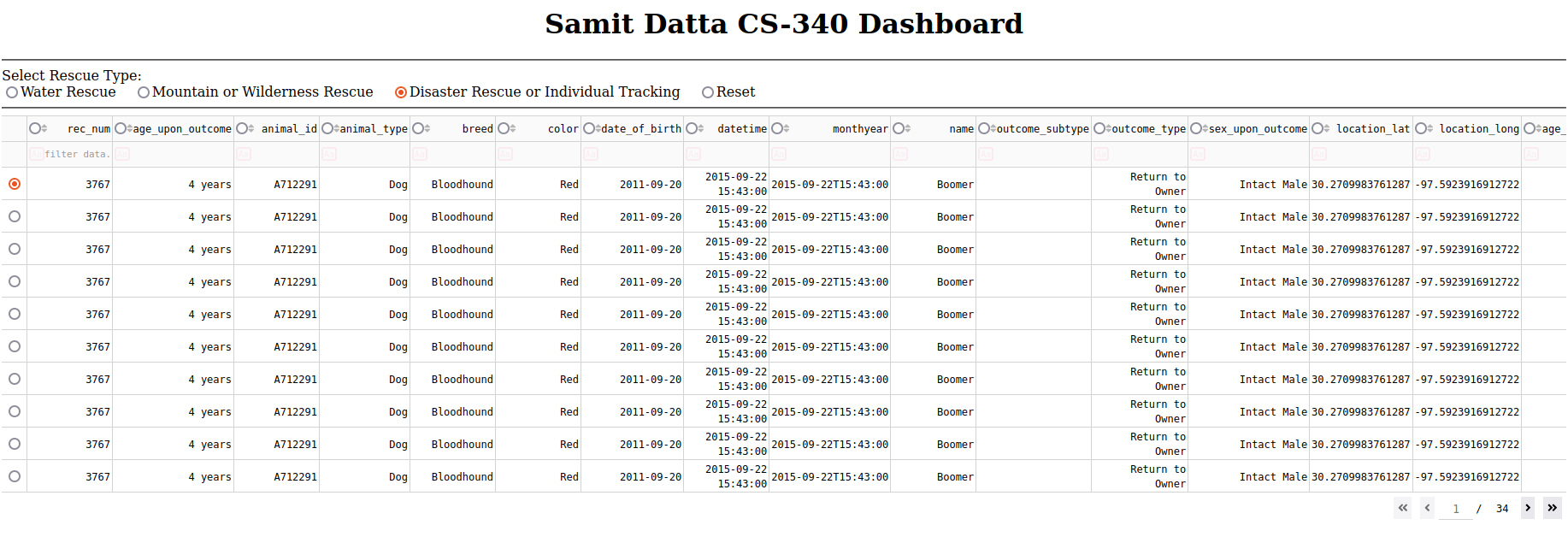
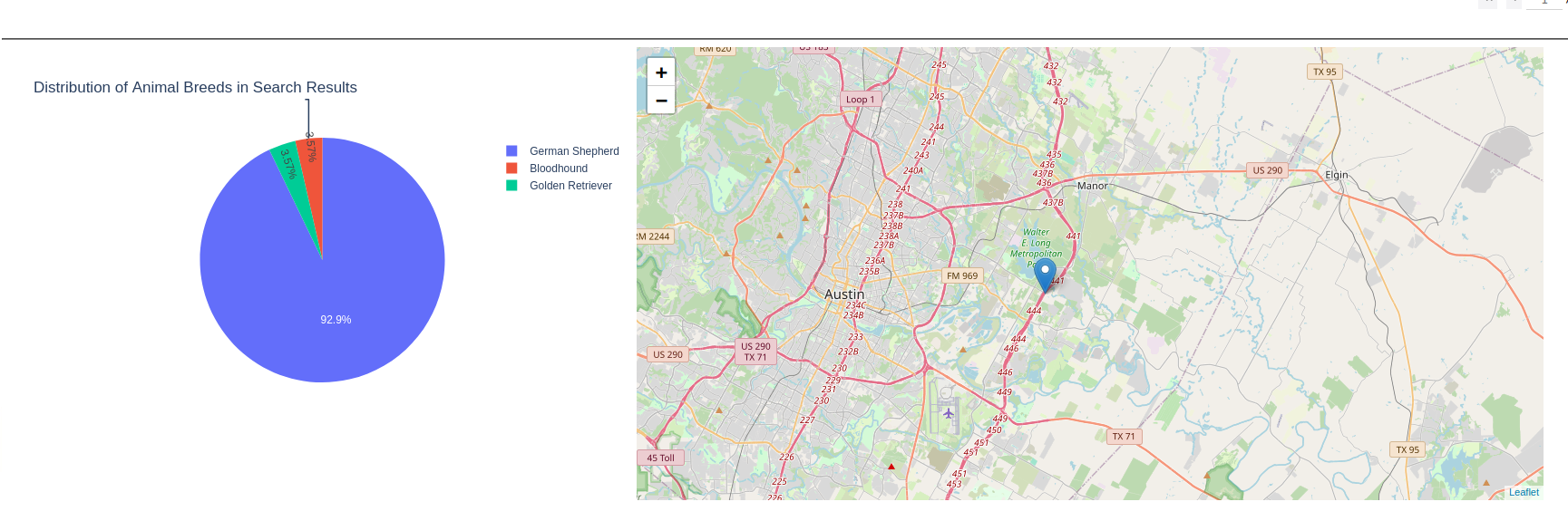
**Describe the required functionality of the project. Include the screenshots or screencast taken while testing and deploying your dashboard (Step 6) as proof that you have achieved the required functionality.** This project is a web-based dashboard for Grazioso Salvage to identify and categorize rescue dog candidates based on data from regional animal shelters. The dashboard provides interactive filtering and dynamic visualizations to support decision-making for search-and-rescue training.  
  
Reset Data

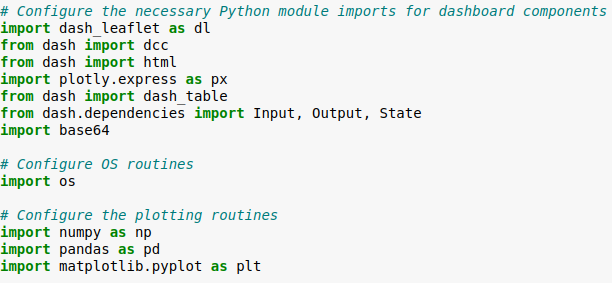
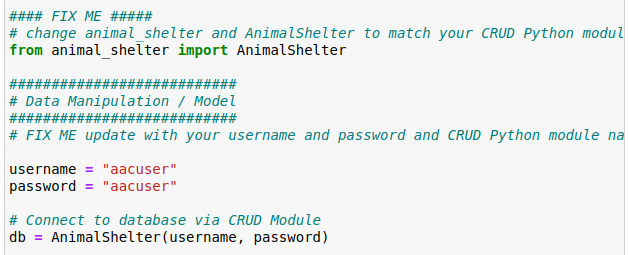
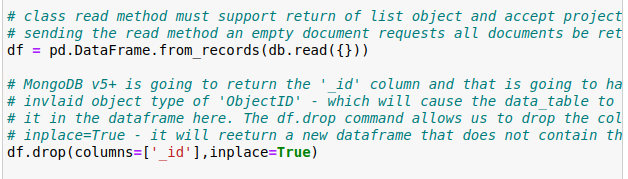
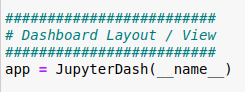
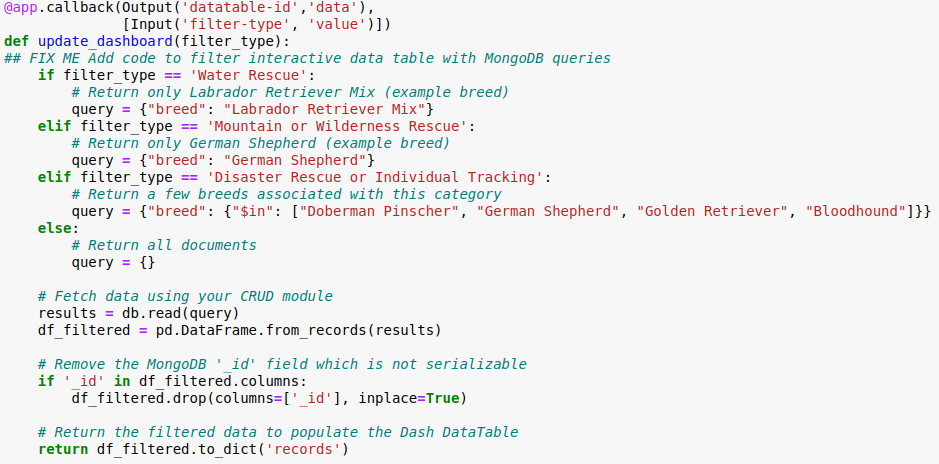
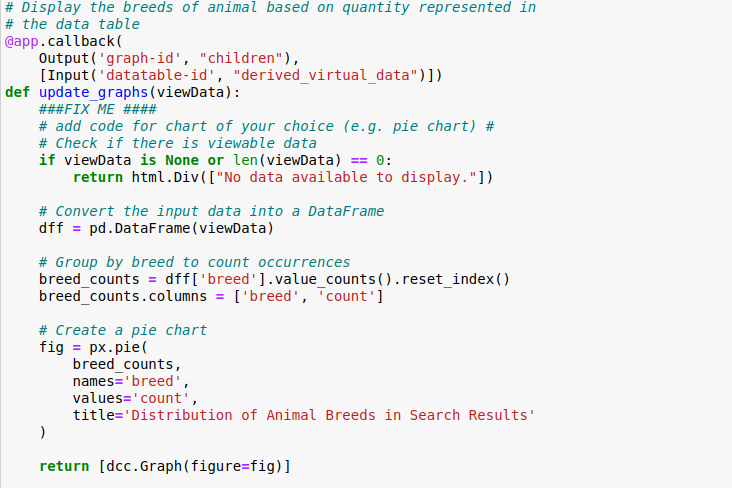
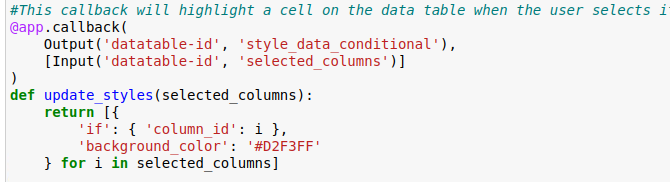
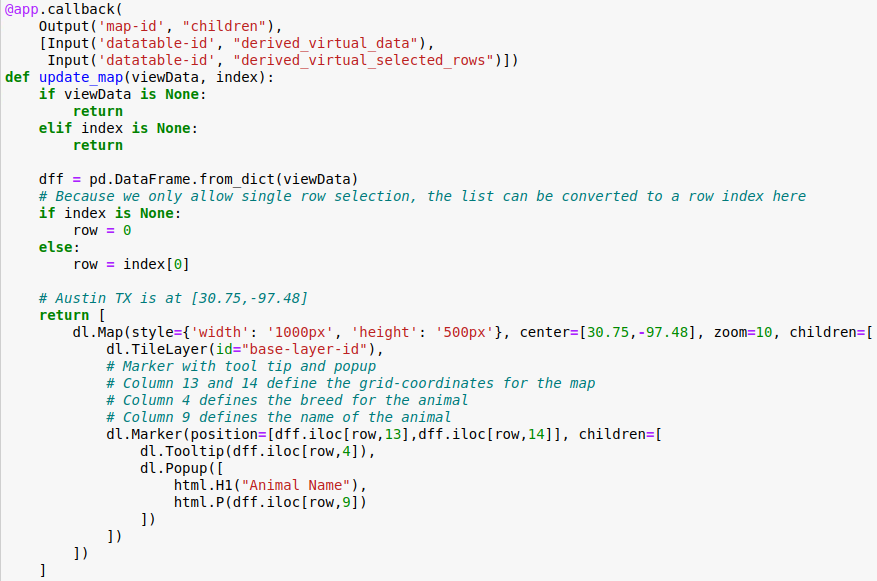
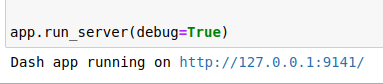
  
  
  
  
  
  
  
  
Water Rescue  
  
****  
  
  
Mountain or Wilderness Rescue  
  


  
  
Disaster Rescue or Individual Tracking  
  
  


**Describe the tools used to achieve this functionality and a rationale for why these tools were used.**

I have used MongoDB and Dash. MongoDB’s document-based structure allows flexible storage of the diverse animal data, including breeds, ages, and locations. Its integration with Python through PyMongo provides seamless query capabilities and dynamic data retrieval that powers the dashboard. Dash offers a Python-first approach to building interactive web apps without requiring JavaScript. It provides native components for tables, charts, and user inputs, which integrate smoothly into a clean MVC-like architecture.

**Explain the steps that were taken to complete the project.**Prerequisite: A created MongoDB database and CRUD module

1. Importing necessary libraries:  
     
     
     
     
     
     
     
     
     
     
     
   
2. Connecting to the MongoDB using the CRUD module:  
   
3. Retrieve and clean the initial dataset:  
   
4. Setting up the dashboard layout:  
   
5. Configuring the interactive data table:  
     
   
6. Create callbacks for interactivity:  
     
     
     
     
   
7. Running the server:  
     
   

**Identify any challenges that were encountered and explain how those challenges were overcome.**

The hardest challenge I have encounter was working on the FIXME about filtering the data table with MongoDB queries, The reason it was hard because when the user picks a rescue type, the app has to send the right query to the MongoDB, retrieve the data with the dogs that matches the rescue type base on breed and age, and then update the table, pie chart and map with the filtered data. I was able to solve this by allowing the user to select a filter to trigger a function. This function checks which rescue type is chosen and then queries for the specific breed for that type, fetches the data from MongoDB using “db.read(query)” to match the query and then convert the data in a DataFrame and removes the “\_id” column as Dash DataTable can’t handle MongoDB’s ObjectID type. Finally, it updates the filtered data back to the DataTable to display to the user.