# CS 340 README Template

## About the Project/Project Title

The project is called ‘Austin Animal Center Database Manipulation with Python and MongoDB.’ It uses CRUD and MongoDB, to work with the database and find/manipulate information.

## Motivation

The motivation and goal behind this project is to first familiarize the user with how to use and setup MongoDB. Learning how to set up and use MongoDB and learning commands like ‘find(), insert().’ As well as learning CRUD (Create, update, read, delete) a database using python.

**Functionality/ Purpose of CRUD**

The purpose of a CRUD module is to make it easier to access and perform functions needed in the database. It makes it a lot easier to reuse the functions and allows for better scalability for a bigger project. It also follows one of OOP principles of encapsulations. Furthermore, it looks cleaner, testing is a lot easier because you only need to call update(),delete(),read(), or create().

**CREATE** - create is the first pillar of CRUD, it takes a dictionary and it add the data into the current database. EX: shelter.create(data)

**READ -** read function take a query and search for that information inside the database, it will return an empty list if the query doesn’t exist. EX: shelter.read(query)

**UPDATE** - update will query for the specific data before updating the requested information in that data, user can choose if they want to update all or just one, the function will return the amount of docs that it updated EX: shelter.update(query, new\_data,multi=True/False)

**DELETE -** delete will query the request and delete all/one data depending on the users’ request. It will return the number of documents that was deleted. EX: shelter.update(query,multi=True/False)

**CHALLENGES**

* One of the most difficult challenges that I faced while writing the code was that my code was working fine when it was on reset. But if I navigated to any of the other query like water or disaster, the query would not show up. I can tell that the query is being pulled correctly in terms of the MongoDB, but it would not show up on my dashboard. Finally I figured it out that I need this line:  **if '\_id' in df.columns: df = df.drop(columns=['\_id']).** This help me fixed the problem but it was very frustrating before because didn’t know why my query wasn’t showing up at first

## Getting Started

1. Set up the database with MongoDB. Use the command mongoimport to import the files into the AAC database into the animals collection. Provided is the command. Please provide your user, password, port, and host.

**mongoimport --username="${MONGO\_USER}" --password="${MONGO\_PASS}" --port="${MONGO\_PORT}" --host="${MONGO\_HOST}" --db AAC --collection animals \**

**--authenticationDatabase admin --type csv --headerline --drop \**

**/usr/local/datasets/aac\_shelter\_outcomes.csv**

1. Create a simple and complex index for queries purposes

**Simple - db.animals.createIndex({ breed: 1 })**

**Complex - db.animals.createIndex({ breed: 1, outcome\_type: 1 })**

1. Create a username and password from the admin database with access to the AAC database with read/write permission

**use admin**

**db.createUser({ user: "aacuser",pwd: passwordPrompt(),roles: [{ role: "readWrite", db: "AAC" }]})**

1. Download the python file that has all the CRUD commands and save it ‘aac\_crud.py’
2. Import aac\_crud to Jupyter Notebook to get access to MongoDB and run the code

## Installation

* Python (pymongo)
  + **Pymongo** was the driver that was chosen for this project because it’s an easy to use API that is compatible with Python that can also access the MongoDB to perform simple CRUD functions. It supports a lot of the needed functions for this project while also being easy to navigate and well-supported in the community.
* Jupyter Notebook run the code
* Python (dash)
  + **Dash** was selected for this project because it works with plotly, the graph that we are using for the dashboard, it’s also very lightweight and function well with python so it’s the perfect framework for this project.
* MongoDB to import the AAC files as well as running all the python commands
  + **MongoDB** is decided for this project because it has PyMongo which helps with interacting with Python. Through python dictionary, it makes it easy to use MongoDB to insert, read, or delete any data. It’s also efficient at handling data, which helps when there is a high amount of data.

## Usage

### Code Example

After following the instructions from ‘Getting Started.’ User can now access and search within the database using commands like ‘ create() or read()’. Load up Jupyter Notebook and run this command. User should be able to now access the database and perform all the queries.

***from acc\_crud import AnimalShelter***

***shelter = AnimalShelter()***

***query = {‘name’ : ‘Buddy’}***

***shelter.create(data)***

***shelter.read(query)***

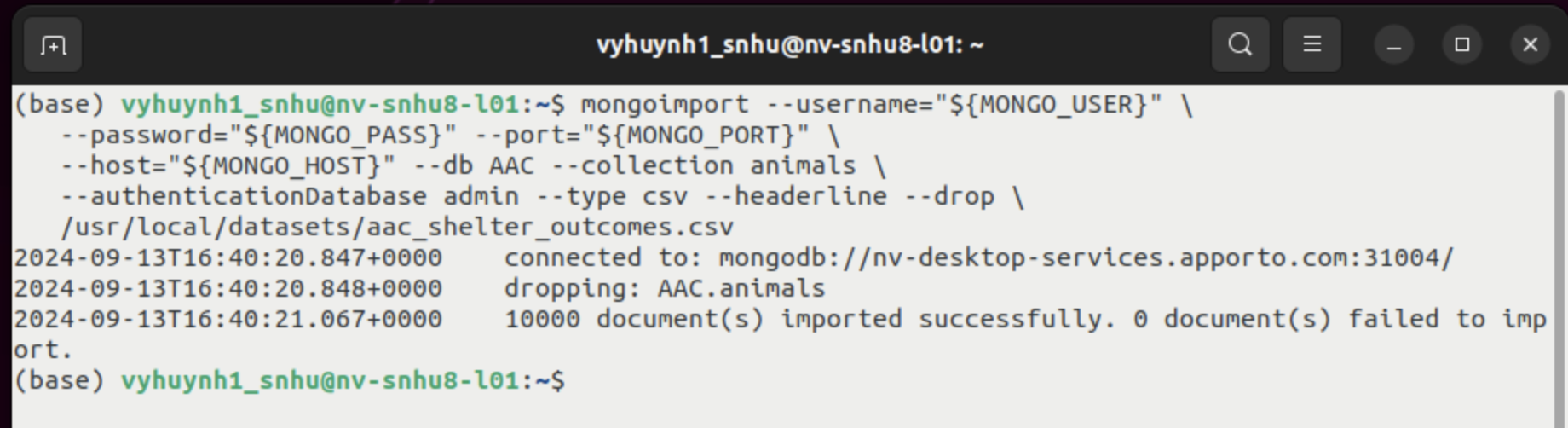
***shelter.update(data,new\_query,multi=True/False)***

***shelter.delete(data,multi=True/False)***

### Tests

The database was tested to make sure it doesn’t accept data that isn’t in dictionary format, or trying to query as an int “{0: ‘hi’}”. Furthermore, it uses ‘try/except’ to make sure that if there is an error, the user knows the exact problem.

1. Insert the AAC database in MongoDB

****

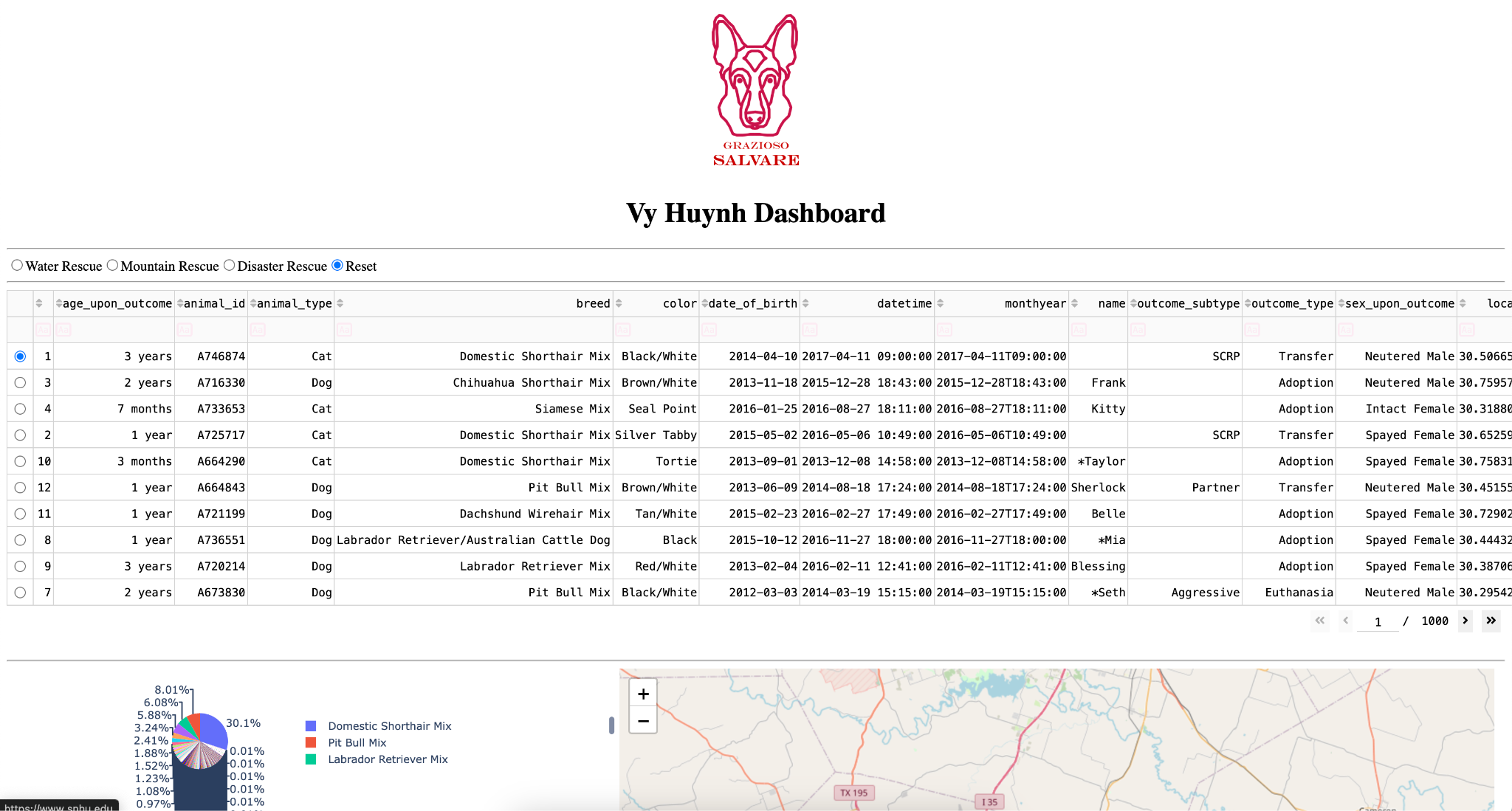
1. Ensures User Authentication

****

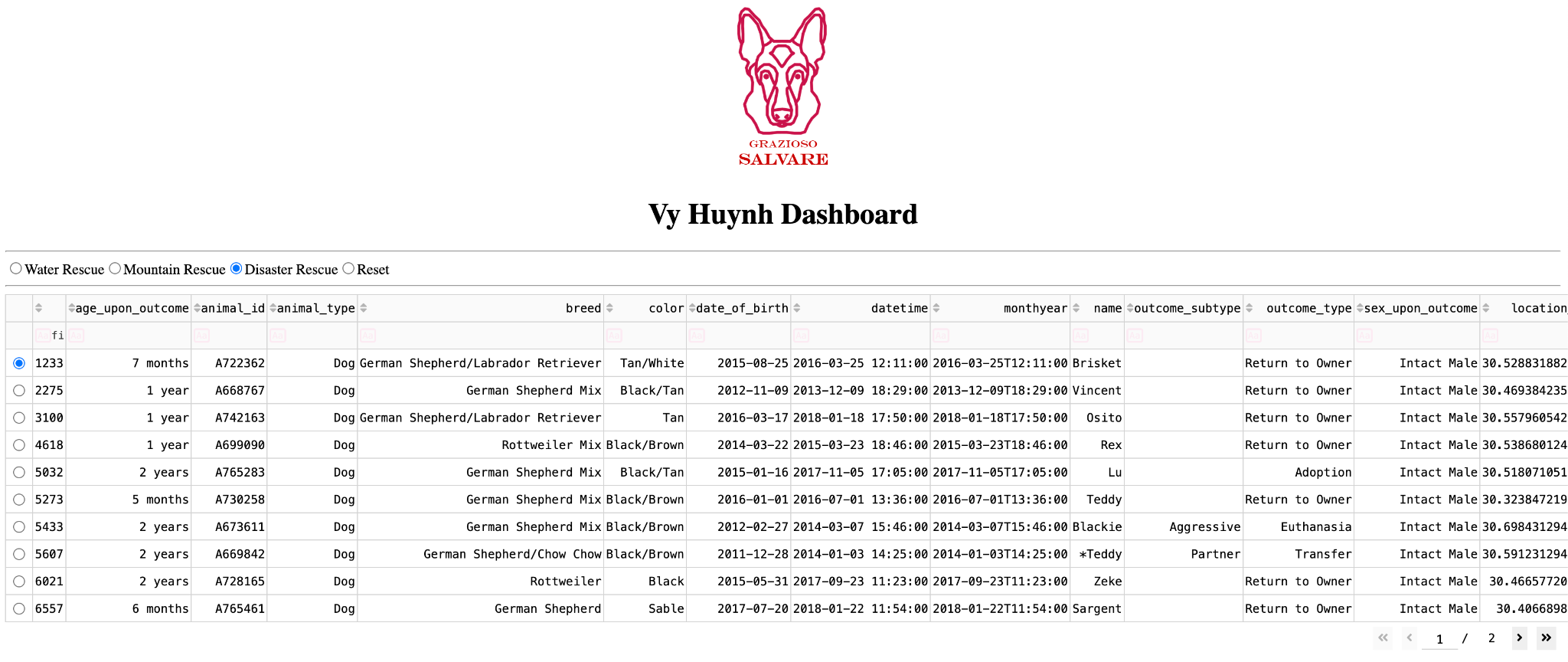
1. Load up Jupyter Notebook and run the code, click on the link to open the dashboard

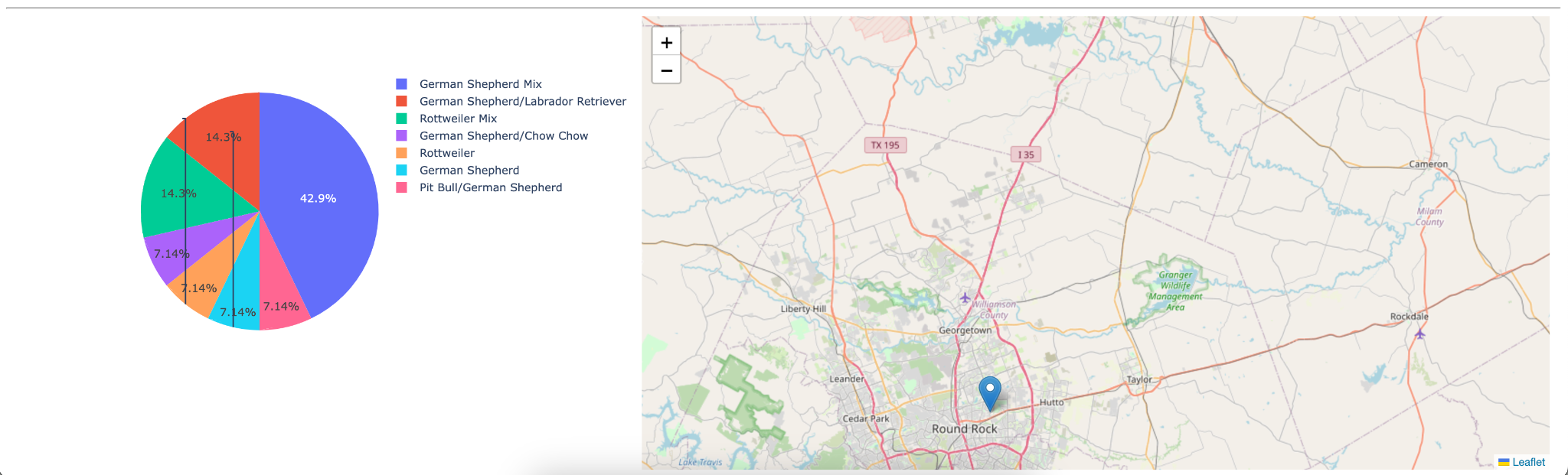


1. Dashboard first launched

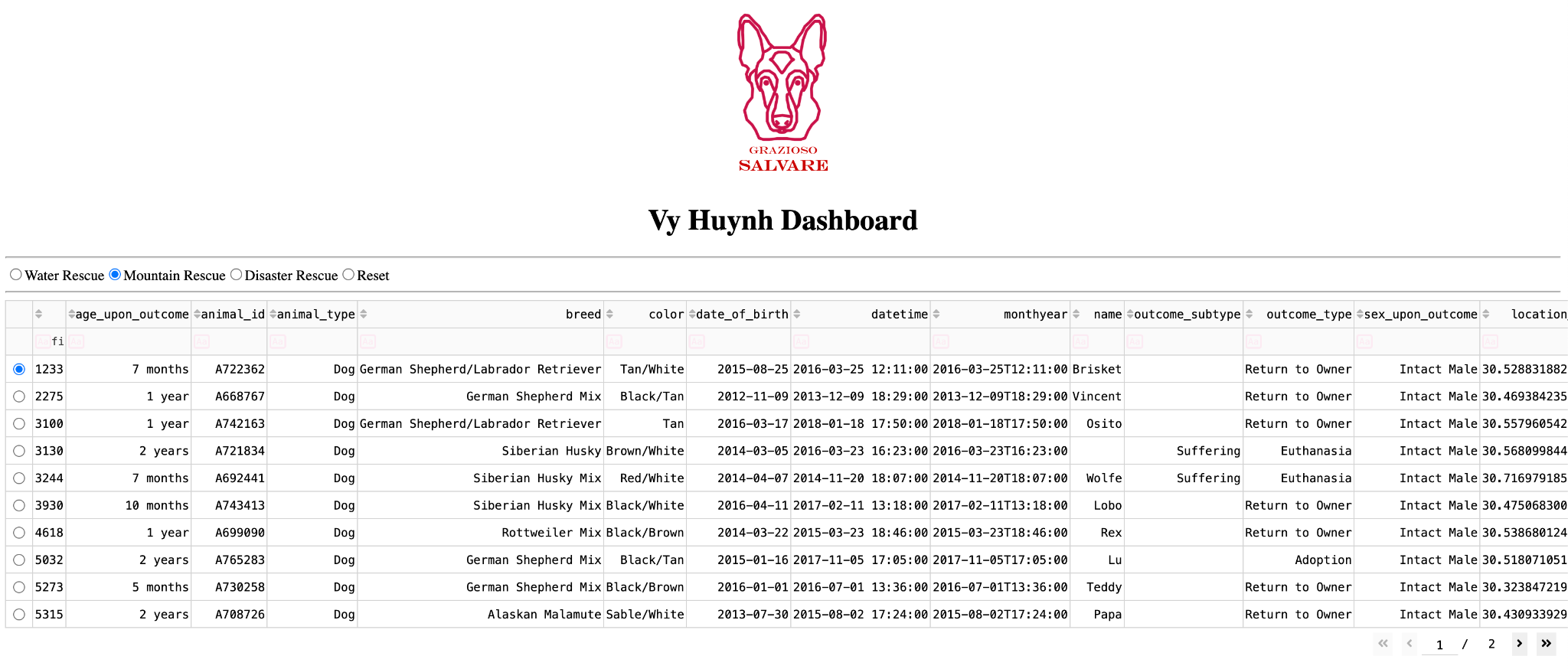


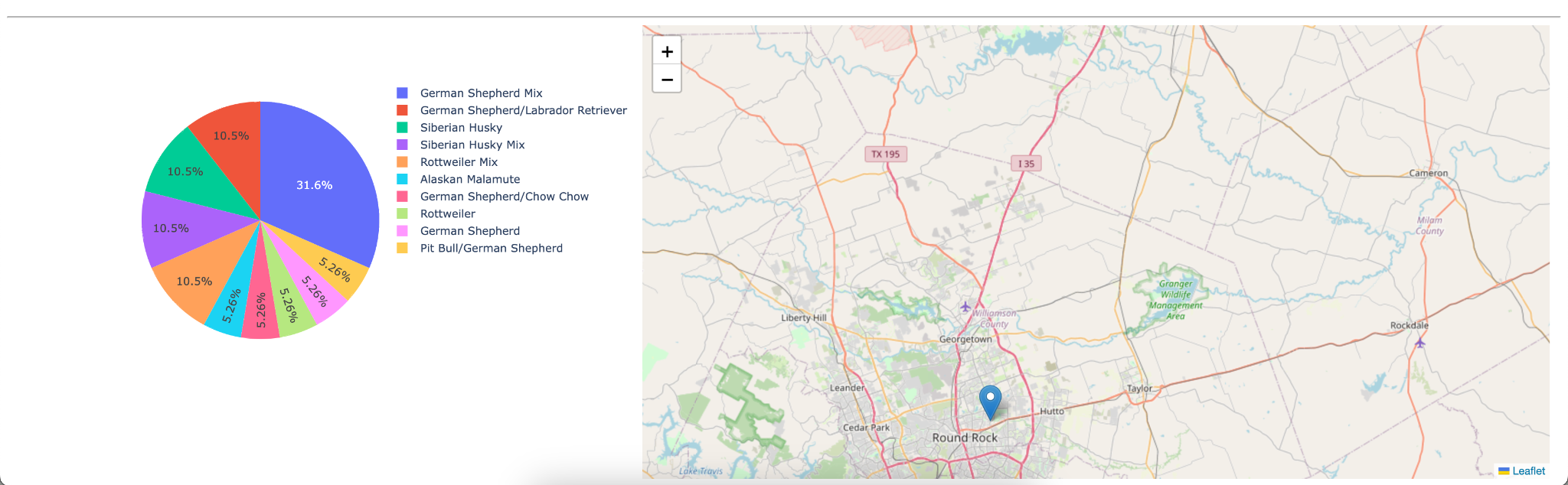
1. Disaster Rescue Set



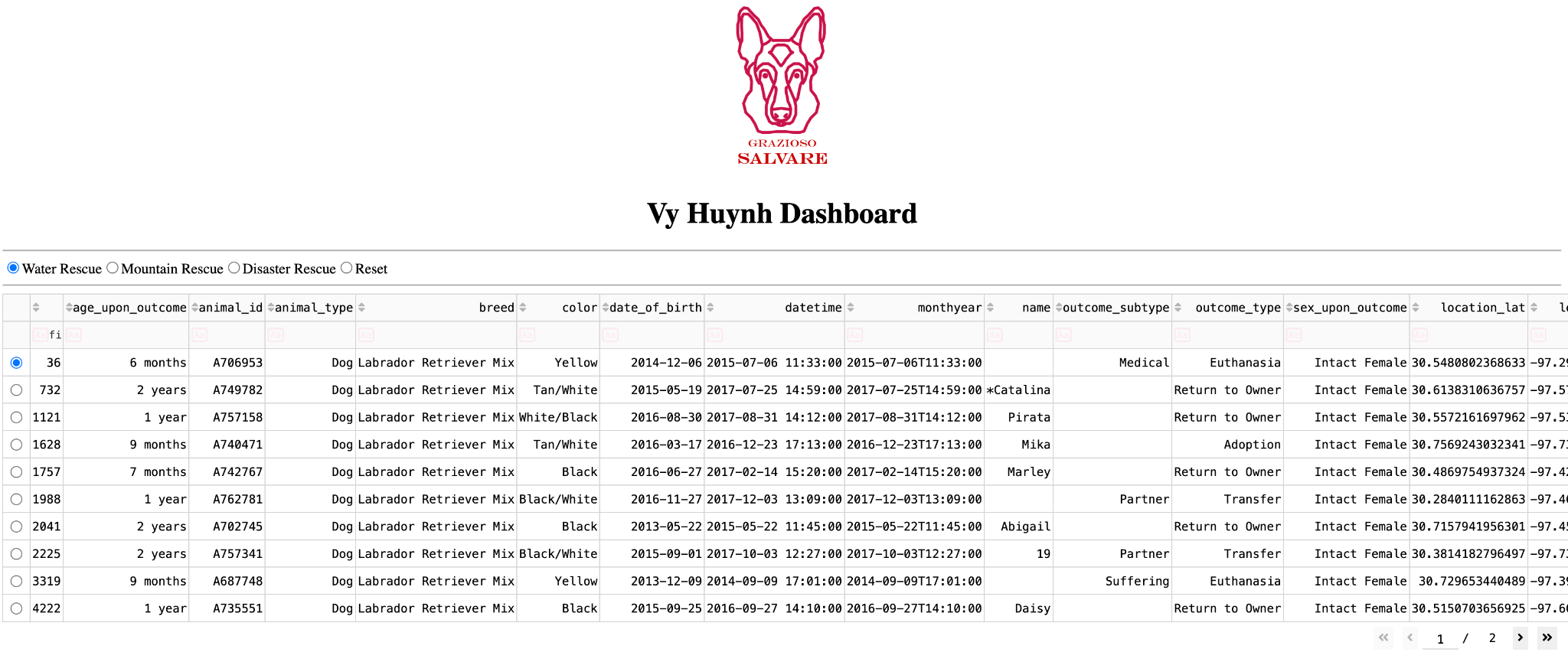


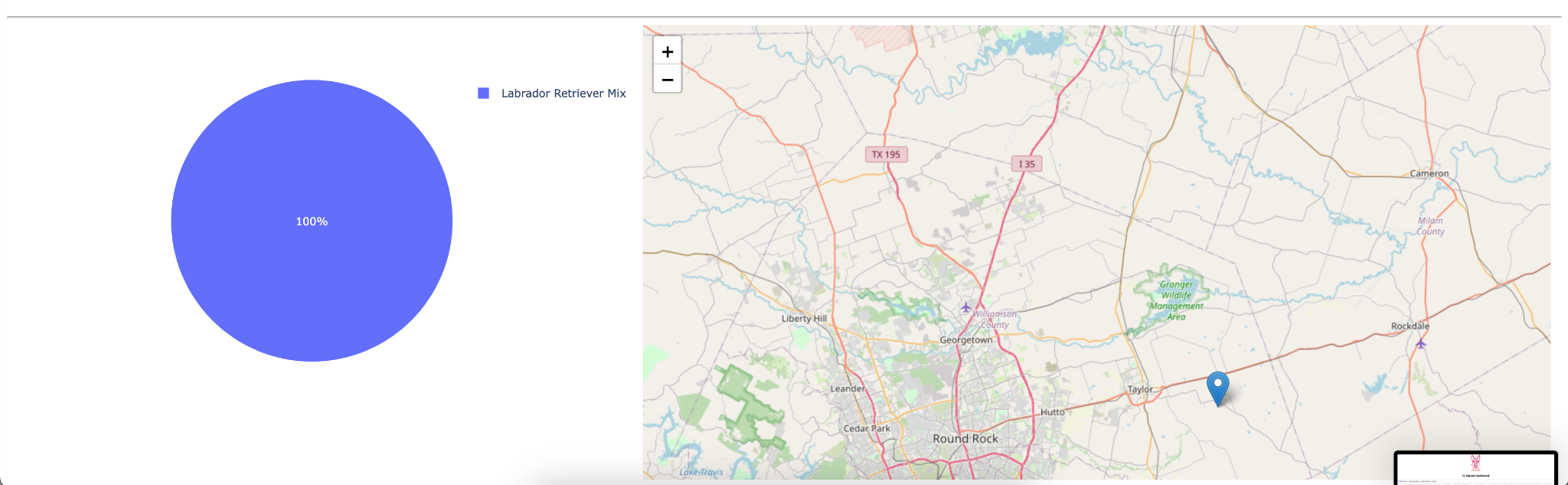
1. Mountain Rescue



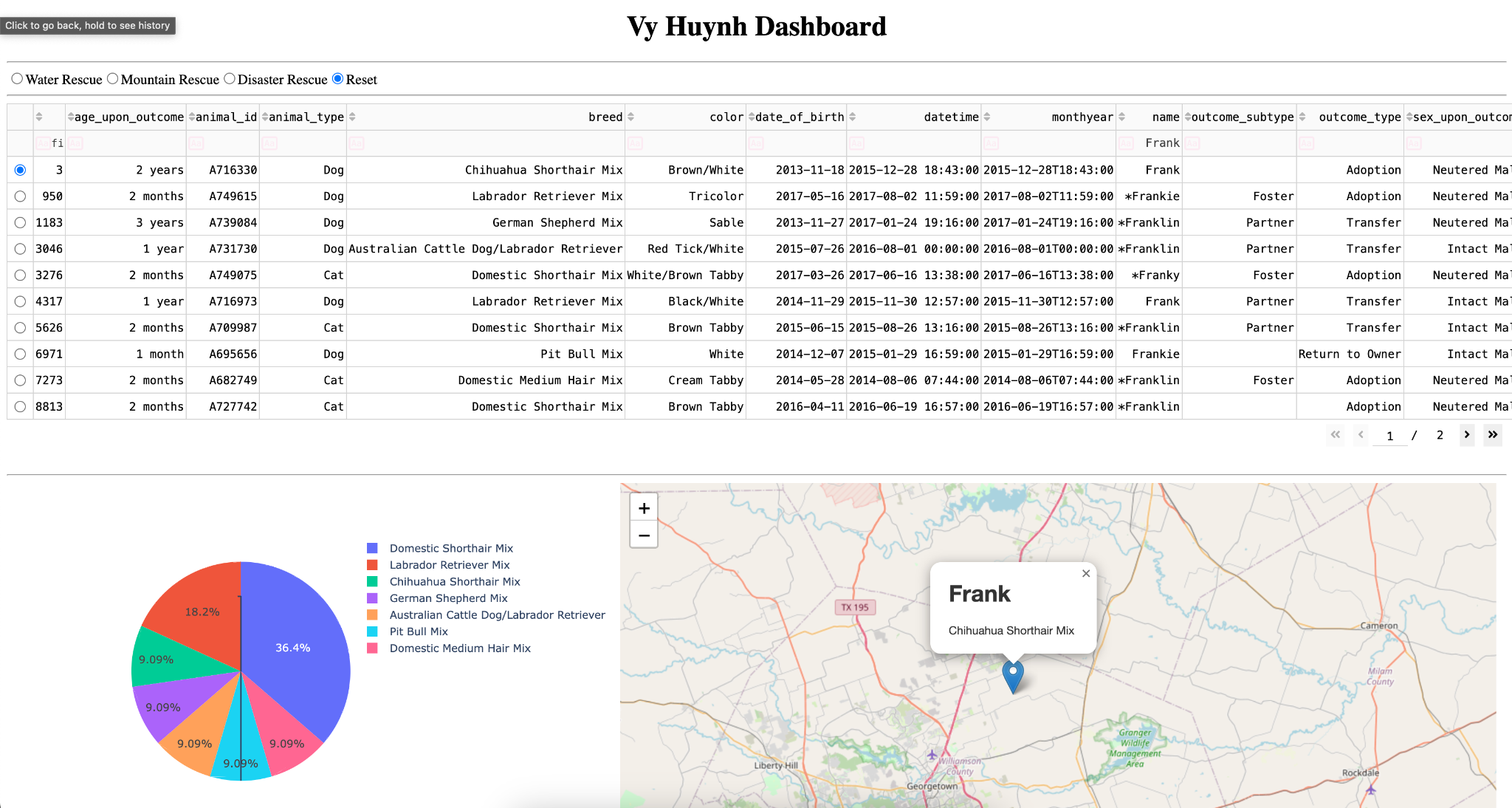


1. Water Rescue

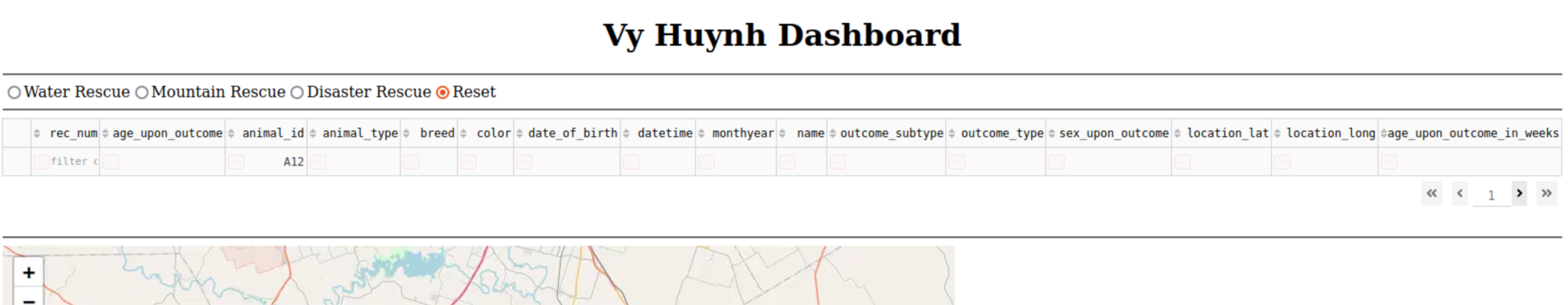




1. Searching for all animal names Frank



1. Searching for an invalid query



## Contact

Your name: Vy Huynh