**Grazioso Salvare Dashboard**

**1. Project Overview**

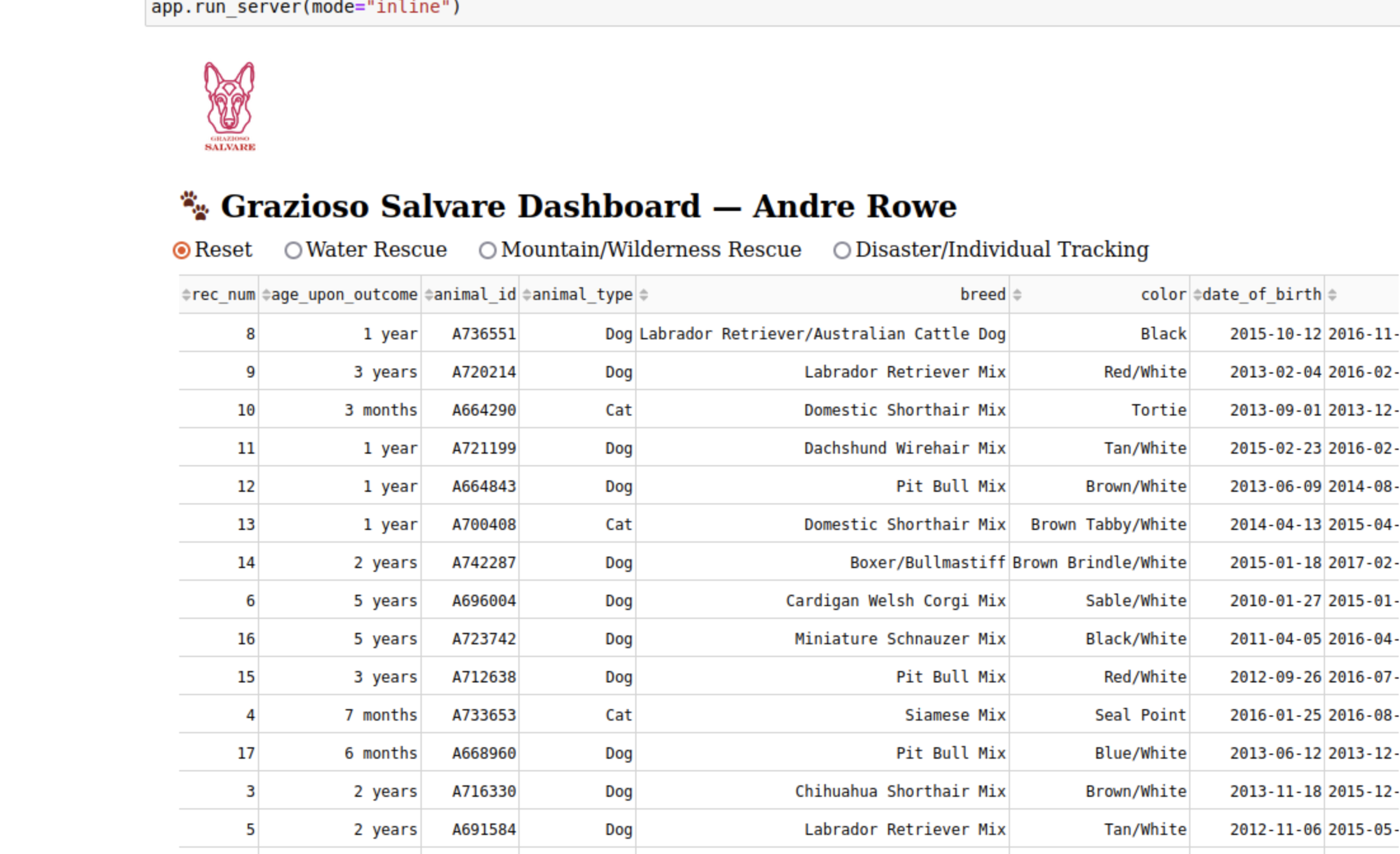
The Grazioso Salvare Dashboard is an interactive web app built in Python using Dash. It connects to a MongoDB instance of Austin Animal Center Outcomes and allows users to filter shelter animals by rescue type, view them in a paginated table, a map of their locations, and a bar chart of outcome types.

**2. Required Functionality**

**Reset (All Animals)**

**Filter**: Reset

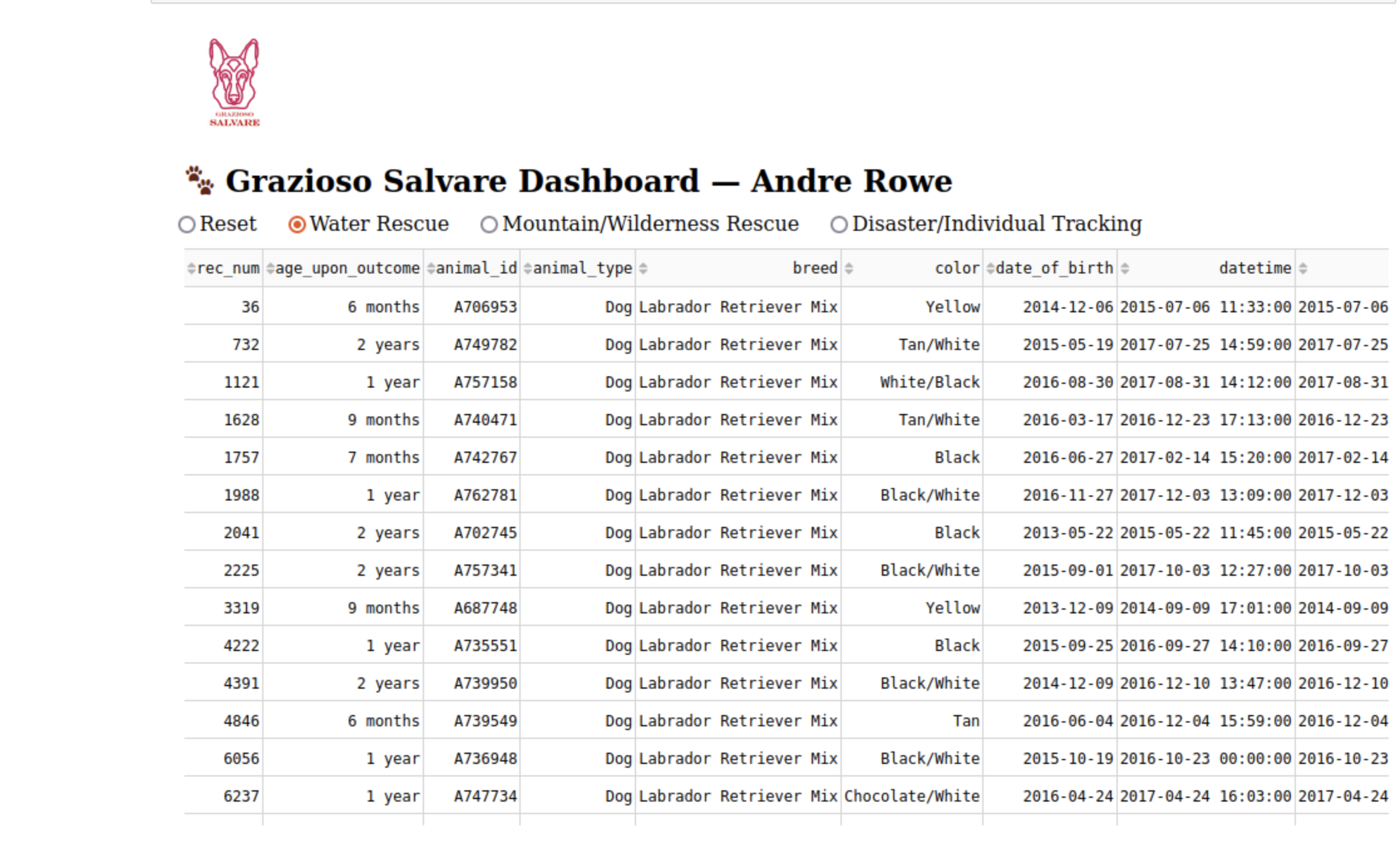
**Description**: Shows every record unfiltered.

Screenshot:

**Water Rescue**

Filter: Water Rescue

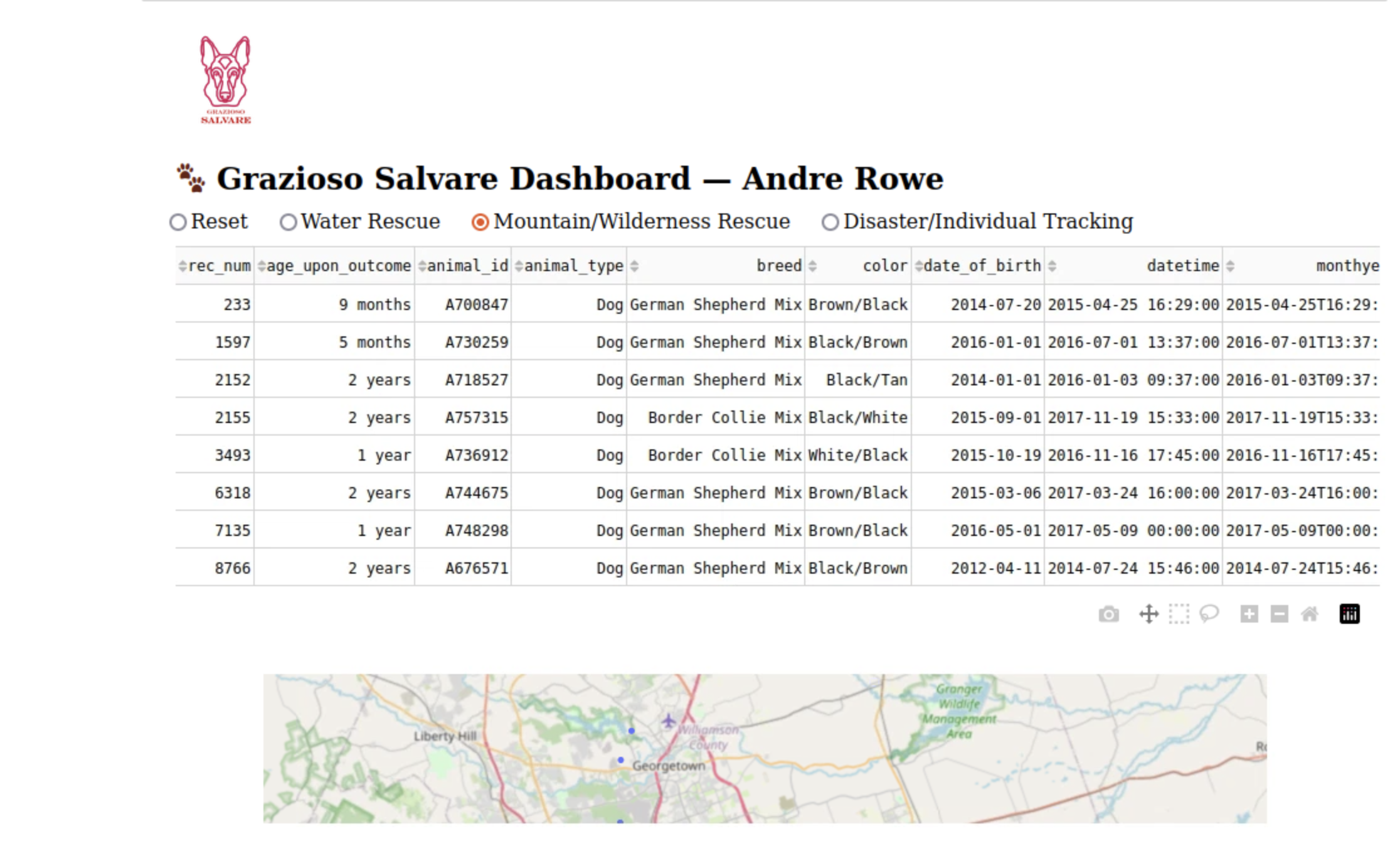
Description: Dogs breeds suited for water rescue, intact females 6 months–2 years old.

:

**Mountain/Wilderness Rescue**

Filter: Mountain/Wilderness Rescue

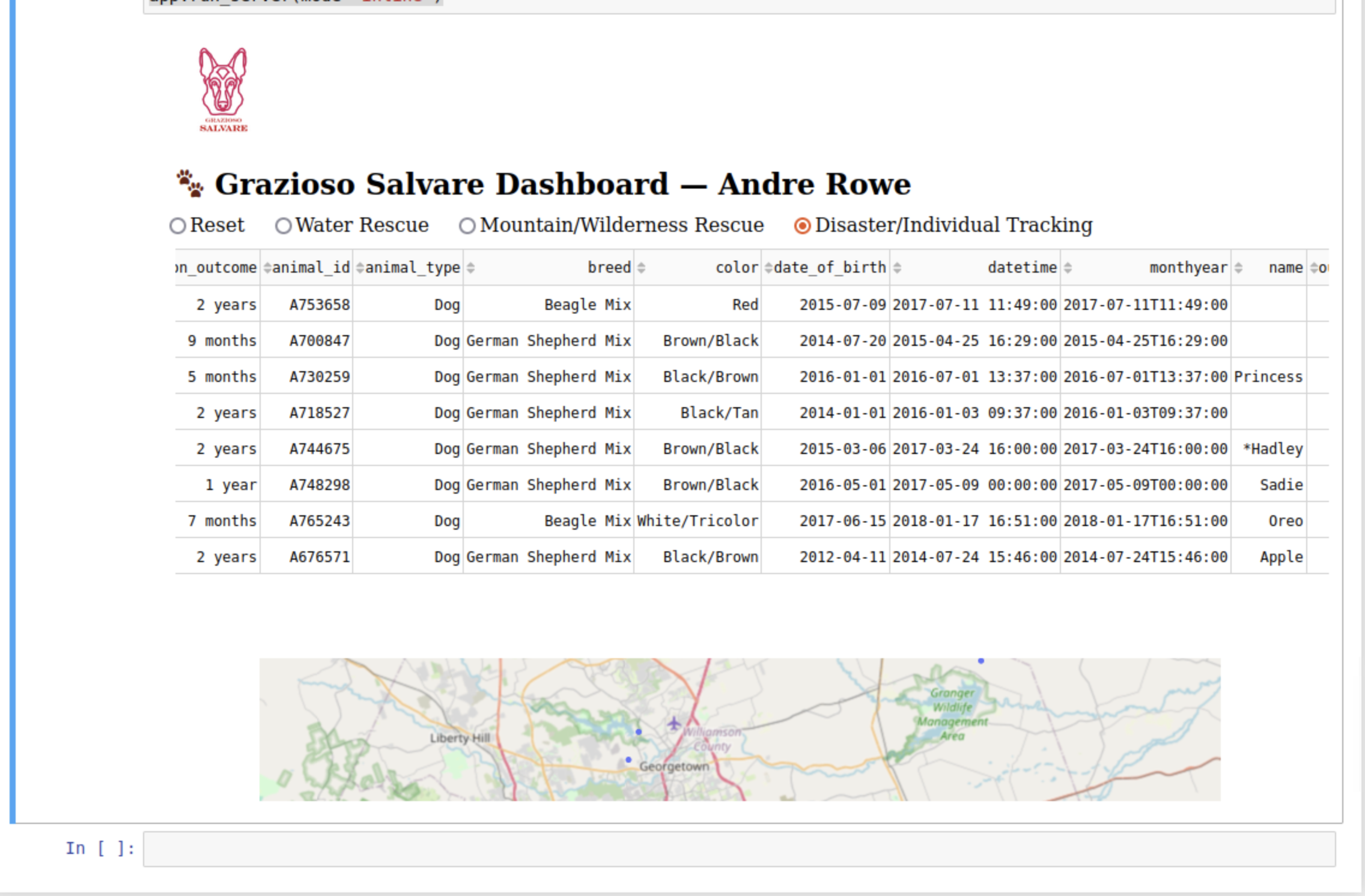
Description: Dogs breeds for mountain or wilderness rescue, intact females 6 months–2 years old.

:

**Disaster/Individual Tracking**

Filter: Disaster/Individual Tracking

Description: Dogs breeds used for disaster/individual tracking, intact females 6 months–2 years old.



**3. Tools & Rationale**

**MongoDB & PyMongo**

Chosen for its flexible JSON‑style document model and easy Python integration via the official driver.

**Dash (JupyterDash)**

Provides MVC‑style callbacks, built‑in DataTable and Plotly charts, and rapid prototyping in Jupyter.

**Pandas**

For converting Mongo query results into DataFrames for both table display and charting.

**4. Reproduction Instructions**

**Prerequisites**

Python 3.x

MongoDB instance with the AAC “animals” collection

Credentials: username “aacuser”, password “AACuser2025”

**Install dependencies**

pip install pymongo pandas dash jupyter‑dash plotly

**Clone the repo** (or download the zip) so you have:

ProjectTwoDashboard/

├── ProjectTwoDashboard.ipynb

├── assets/

│ ├── logo.png

│ └── screenshots/

│ ├── reset.png

│ ├── water.png

│ ├── mountain.png

│ └── tracking.png

Open the notebook

In JupyterLab or VS Code’s Notebook editor, launch ProjectTwoDashboard.ipynb.

Run all cells

The final cell spins up the dashboard inline—just interact with the radio buttons.

**5. Challenges & Solutions**

ObjectId serialization

* Mongo’s \_id field caused JSON errors in Dash. We drop it before passing data to the table.

Logo loading

* Asset paths and spaces in filenames required embedding via base64.

Port conflicts

* Using JupyterDash inline mode eliminated separate server‑port issues.

**6. Resources & Links**

MongoDB: https://mongodb.com

PyMongo docs: https://pymongo.readthedocs.io

Dash docs: https://dash.plotly.com

Austin Animal Center Outcomes: https://doi.org/10.26000/025.000001