Southern
New Hampshire
University

CS 340 Project Two
GCZ79
12/09/2025

About the Project

This project focuses on developing the CRUD functionalities (Create, Read, Update, and Delete) of a Python module and a Dashboard for interacting with the Austin Animal Center (AAC) MongoDB database. The module enables users to connect to a MongoDB database, insert new records, query existing records using key-value pairs, update existing records, and delete records. This phase of the project extends the CRUD Python module by implementing a web-based interactive dashboard using the Dash framework. The Dashboard will allow Grazioso Salvare staff to visually explore and filter the animal shelters' data to identify dogs suitable for search-and-rescue training.
The project is divided into two files:

- CRUD_Python_Module.py (CRUD operations module)
- ProjectTwoDashboard.ipynb (Script to use the CRUD module and generate the Dashboard)

Motivation

Grazioso Salvare, an innovative international rescue-animal training company, requires a software application that enables them to interact with the existing databases of five animal shelters in Austin, Texas, to identify dogs suitable for search-and-rescue training.
Global Rain, the software engineering company that employs me, has contracted for a full-stack development application, and the present document has been produced to accompany the CRUD Python module and the dashboard in Dash.

GRAZIOSO
SALVARE

Getting Started

This interactive dashboard was developed in Jupyter using Python and is designed for dynamic data exploration and management. It implements full CRUD functionality through a dedicated Python module that handles database operations against MongoDB, enabling persistent and scalable data storage. Dash is used to create a reactive user interface with interactive charts and controls, while Leaflet is integrated to render interactive maps, allowing users to explore geospatial data directly within the dashboard. Its structure can be described as it follows.

MODEL: The Python module provides CRUD (Create, Read, Update, and Delete) functionalities to handle the data saved on the Austin Animal Center (AAC) MongoDB database. The dashboard script imports and instantiates the AnimalShelter class from the module at application startup:

```
from CRUD_Python_Module import AnimalShelter
shelter = AnimalShelter(username, password)
```

The credentials to access the database are hardcoded inside the script:
User account = aacuser
Password = NoSQLNoParty

All database interactions, like reading/filtering the results, updating maps, charts, and tables, are performed through this object. The retrieved documents are converted into Pandas DataFrames for downstream processing.

LOGIC: A helper function (get_rescue_query) defines the MongoDB query criteria for the different rescue types (Water, Mountain/Wilderness, Disaster/Tracking). This function is reused among buttons, charts, and callbacks to enforce the DRY principle (Don't Repeat Yourself).
The data is cleaned of the unsupported _id fields, and empty or failed queries are handled gracefully.
A dataset cardinality summary is generated to get more insights into the data by analyzing column uniqueness and classification.

VIEW: The dashboard UI is built using Dash HTML components, Leaflet, and Plotly visualizations.
It includes:

- Header with organization logo and title
- Rescue-type filter buttons
- Interactive DataTable with:
  - Sorting, filtering, and pagination
  - Row and column selection
- Collapsible sections for:
  - Interactive Map and summary Pie Chart
  - Dynamic column visibility controls
  - Dataset cardinality analysis

CONTROLLER: This part coordinates user actions with updates to the dashboard. It features:

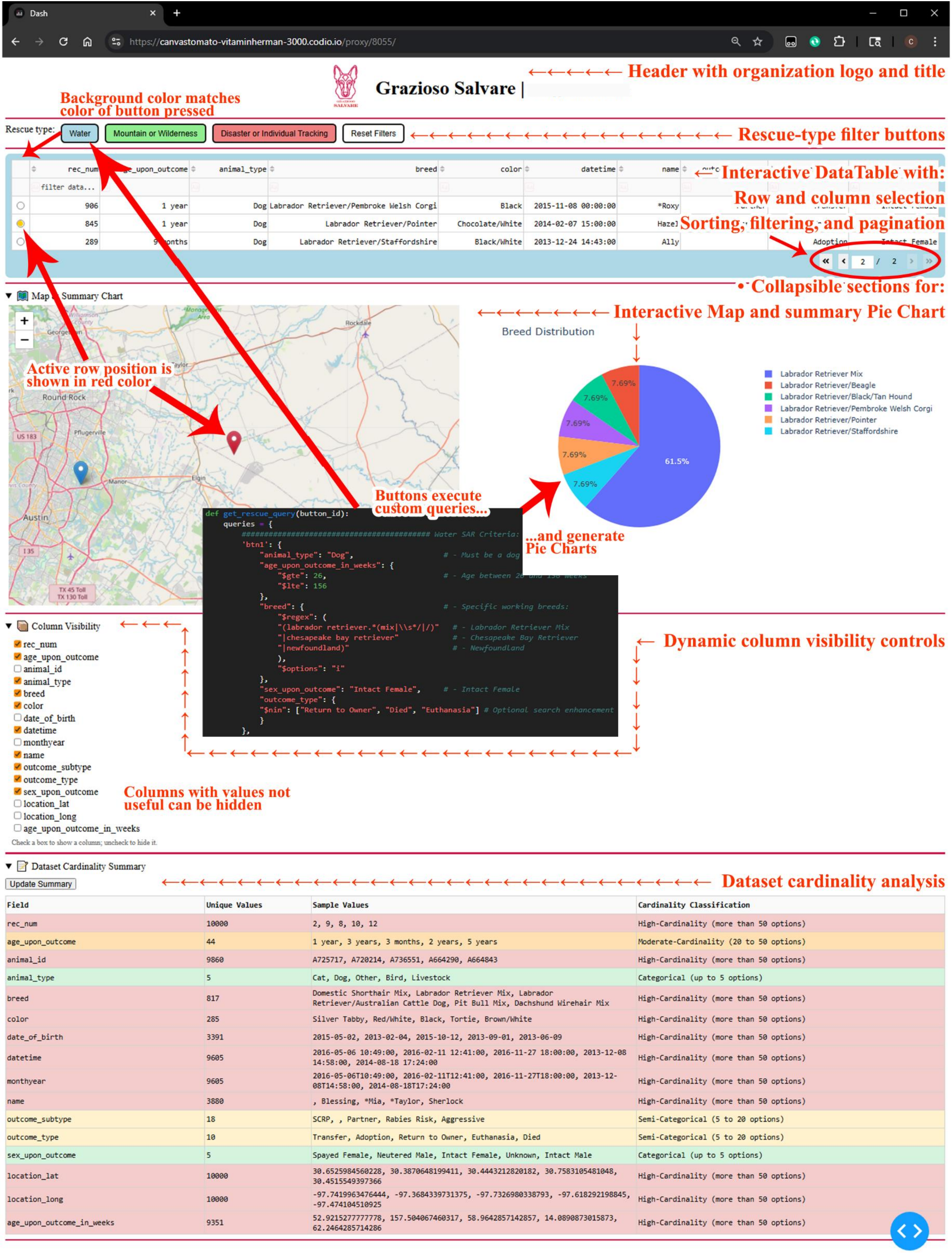
- Dash callback functions to connect UI components to the data layer
- User interaction triggers:
  - Database queries
  - Table updates
  - Map rendering
  - Chart generation
  - Dynamic styling changes
- Context-aware callbacks identify which UI control triggered the update

RUNTIME: The dashboard runs inside JupyterLab, and the application is launched locally on port 8055.

Examples of interactions:

- View: User clicks a rescue-type button
- Controller: Dash callback processes the input and determines the active filter
- Model: MongoDB is queried for matching records
- View: DataTable, map, chart, and table styling are updated with the query results

- View: The user selects a row in the DataTable representing an individual rescue record.
- Controller: A Dash callback detects the selected row event, extracts the associated record identifier and location data, and determines which map marker corresponds to the selected entry.
- Model: The underlying dataset (previously retrieved from MongoDB) is referenced to obtain the record's geographic coordinates and metadata.
- View: The map component updates to visually highlight the selected marker.



Header with organization logo and title

Grazioso Salvare |

Background color matches color of button pressed

Rescue type: Water | Mountain or Wilderness | Disaster or Individual Tracking | Reset Filters

Rescue-type filter buttons

Interactive DataTable with:

Row and column selection

Sorting, filtering, and pagination

	rec_num	age_upon_outcome	animal_type	breed	color	datetime	name	outc		
	filter data...									
○	906	1 year	Dog	Labrador Retriever/Pembroke Welsh Corgi	Black	2015-11-08 00:00:00	*Roxy			
●	845	1 year	Dog	Labrador Retriever/Pointer	Chocolate/White	2014-02-07 15:00:00	Hazel			
○	289	9 months	Dog	Labrador Retriever/Staffordshire	Black/White	2013-12-24 14:43:00	Ally		Adoption	Intact Female

« ‹ 2 / 2 › »

• Collapsible sections for:

Interactive Map and summary Pie Chart

Active row position is shown in red color

Buttons execute custom queries...

...and generate Pie Charts

```
def get_rescue_query(button_id):
    queries = {
        ########################################## Water SAR Criteria:
        'btn1': {
            "animal_type": "Dog",                  # - Must be a dog
            "age_upon_outcome_in_weeks": {
                "$gte": 26,                        # - Age between 26 and 156 weeks
                "$lte": 156
            },
            "breed": {                             # - Specific working breeds:
                "$regex": (
                "(labrador retriever.*(mix|\\s*/|/)"   # - Labrador Retriever Mix
                "|chesapeake bay retriever"            # - Chesapeake Bay Retriever
                "|newfoundland)"                       # - Newfoundland
                ),
                "$options": "i"
            },
            "sex_upon_outcome": "Intact Female",   # - Intact Female
            "outcome_type": {
            "$nin": ["Return to Owner", "Died", "Euthanasia"] # Optional search enhancement
            }
        },
```

Column Visibility

Dynamic column visibility controls

- ☑ rec_num
- ☑ age_upon_outcome
- ☐ animal_id
- ☑ animal_type
- ☑ breed
- ☑ color
- ☐ date_of_birth
- ☑ datetime
- ☐ monthyear
- ☑ name
- ☑ outcome_subtype
- ☑ outcome_type
- ☑ sex_upon_outcome
- ☐ location_lat
- ☐ location_long
- ☐ age_upon_outcome_in_weeks

Check a box to show a column; uncheck to hide it.

Columns with values not useful can be hidden

Dataset Cardinality Summary

Update Summary

Dataset cardinality analysis

Field	Unique Values	Sample Values	Cardinality Classification
rec_num	10000	2, 9, 8, 10, 12	High-Cardinality (more than 50 options)
age_upon_outcome	44	1 year, 3 years, 3 months, 2 years, 5 years	Moderate-Cardinality (20 to 50 options)
animal_id	9860	A725717, A720214, A736551, A664290, A664843	High-Cardinality (more than 50 options)
animal_type	5	Cat, Dog, Other, Bird, Livestock	Categorical (up to 5 options)
breed	817	Domestic Shorthair Mix, Labrador Retriever Mix, Labrador Retriever/Australian Cattle Dog, Pit Bull Mix, Dachshund Wirehair Mix	High-Cardinality (more than 50 options)
color	285	Silver Tabby, Red/White, Black, Tortie, Brown/White	High-Cardinality (more than 50 options)
date_of_birth	3391	2015-05-02, 2013-02-04, 2015-10-12, 2013-09-01, 2013-06-09	High-Cardinality (more than 50 options)
datetime	9605	2016-05-06 10:49:00, 2016-02-11 12:41:00, 2016-11-27 18:00:00, 2013-12-08 14:58:00, 2014-08-18 17:24:00	High-Cardinality (more than 50 options)
monthyear	9605	2016-05-06T10:49:00, 2016-02-11T12:41:00, 2016-11-27T18:00:00, 2013-12-08T14:58:00, 2014-08-18T17:24:00	High-Cardinality (more than 50 options)
name	3880	, Blessing, *Mia, *Taylor, Sherlock	High-Cardinality (more than 50 options)
outcome_subtype	18	SCRP, , Partner, Rabies Risk, Aggressive	Semi-Categorical (5 to 20 options)
outcome_type	10	Transfer, Adoption, Return to Owner, Euthanasia, Died	Semi-Categorical (5 to 20 options)
sex_upon_outcome	5	Spayed Female, Neutered Male, Intact Female, Unknown, Intact Male	Categorical (up to 5 options)
location_lat	10000	30.6525984560228, 30.3870648199411, 30.4443212820182, 30.7583105481048, 30.4515549397366	High-Cardinality (more than 50 options)
location_long	10000	-97.7419963476444, -97.3684339731375, -97.7326980338793, -97.618292198845, -97.474104510925	High-Cardinality (more than 50 options)
age_upon_outcome_in_weeks	9351	52.9215277777778, 157.504067460317, 58.9642857142857, 14.0890873015873, 62.2464285714286	High-Cardinality (more than 50 options)

Prerequisites (from Project 1)

To run the dashboard, you will need Python and MongoDB installed. Detailed setup instructions, including Python environment configuration and MongoDB initialization, are available in the README of the previous project.

- MongoDB
  MongoDB is particularly effective for this project thanks to its schema-free structure accommodating variable animal records, native PyMongo integration enabling seamless Python connectivity, and powerful querying capabilities supporting complex rescue-type filtering.
- Python 3.x
  The primary language for developing the CRUD module, it can be downloaded from here.
- Pymongo Library
  This library is used to connect Python to MongoDB, providing the MongoClient class and database/collection methods.

Installation

```
pip install dash jupyter-dash dash-leaflet pandas plotly
```

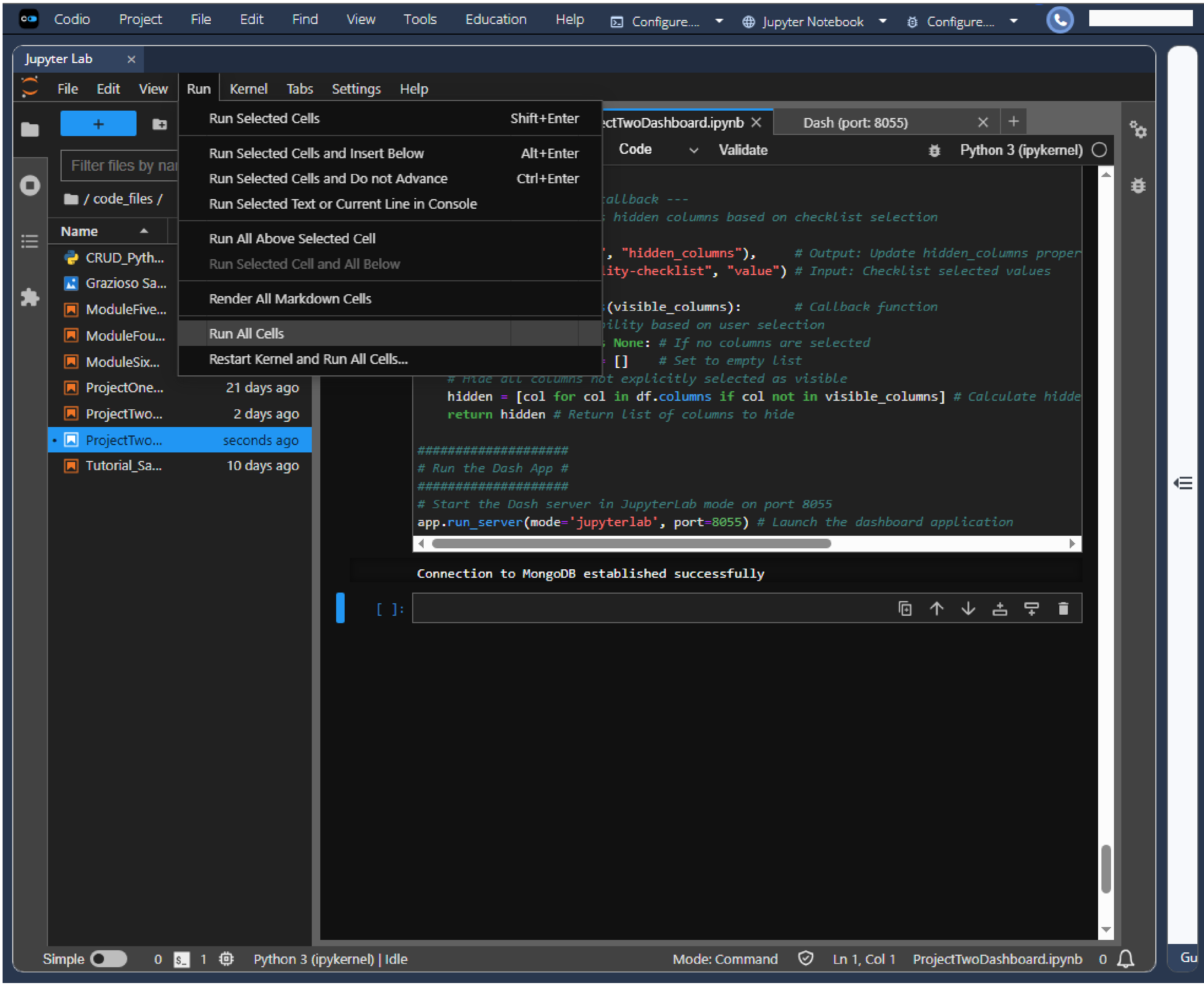
This project uses the following tools and libraries:

- JupyterLab
  A web-based interactive development environment for notebooks, code, and data. JupyterLab is particularly effective for this project thanks to its rapid prototyping of queries and callbacks, its immediate visual feedback when performing UI interactions, and the easy reproduction of the project in local or cloud-based environments. Installation instructions here.
- Dash Framework
  Dash is a Python framework for building interactive, web-based dashboards. It has been chosen for this project because it provides a reactive callback system that updates UI components, such as tables, charts, and maps, in response to user interactions, and without requiring custom JavaScript. Installation instructions here.
- Leaflet (via dash-leaflet)
  Leaflet is a lightweight mapping library used to display interactive geospatial data. Through the dash-leaflet integration, maps and markers can be dynamically updated based on user input, making it well-suited for visualizing location-based rescue data as required in this project. Installation instructions here.
- Pandas
  Pandas provides high-performance data structures that support efficient handling and analysis of relational and labeled datasets, making it a key component of the Python data analysis ecosystem. Pandas is used in this project to efficiently manipulate and analyze data retrieved from MongoDB before it is displayed in the dashboard. Installation instructions here.
- Plotly
  The Plotly Python library is an interactive, open-source plotting library that supports unique chart types covering a wide range of statistical, financial, geographic, scientific, and 3D use cases. It is used in this project to show a pie chart of the different dog breeds for the various SAR types. Installation instructions here.

Setting up the Dashboard

After installing JupyterLab and all the above required libraries, users will import into the software the files CRUD_Python_Module.py and ProjectTwoDashboard.ipynb.
From this last file, they will "**Run All Cells**" (or hit the play button ▶) to connect to MongoDB and launch the dashboard. The dashboard will load locally inside JupyterLab, but it is accessible even through an external link, in our case:

https://canvastomato-vitaminherman-3000.codio.io/proxy/8055/


Codio
Project
File
Edit
Find
View
Tools
Education
Help
Configure...
Jupyter Notebook
Configure...
Jupyter Lab
File
Edit
View
Run
Kernel
Tabs
Settings
Help
Run Selected Cells Shift+Enter
Run Selected Cells and Insert Below Alt+Enter
Run Selected Cells and Do not Advance Ctrl+Enter
Run Selected Text or Current Line in Console
Run All Above Selected Cell
Run Selected Cell and All Below
Render All Markdown Cells
Run All Cells
Restart Kernel and Run All Cells...
+
Filter files by nar
/ code_files /
Name
CRUD_Pyth...
Grazioso Sa...
ModuleFive...
ModuleFou...
ModuleSix...
ProjectOne... 21 days ago
ProjectTwo... 2 days ago
ProjectTwo... seconds ago
Tutorial_Sa... 10 days ago
ctTwoDashboard.ipynb
Dash (port: 8055)
Code
Validate
Python 3 (ipykernel)
[ ]:
Simple
0
1
Python 3 (ipykernel) | Idle
Mode: Command
Ln 1, Col 1
ProjectTwoDashboard.ipynb
0
Gu

```
allback ---
 hidden columns based on checklist selection

', "hidden_columns"),      # Output: Update hidden_columns proper
lity-checklist", "value") # Input: Checklist selected values

(visible_columns):       # Callback function
bility based on user selection
 None: # If no columns are selected
 []    # Set to empty list
    # Hide all columns not explicitly selected as visible
    hidden = [col for col in df.columns if col not in visible_columns] # Calculate hidde
    return hidden # Return list of columns to hide

####################
# Run the Dash App #
####################
# Start the Dash server in JupyterLab mode on port 8055
app.run_server(mode='jupyterlab', port=8055) # Launch the dashboard application
```

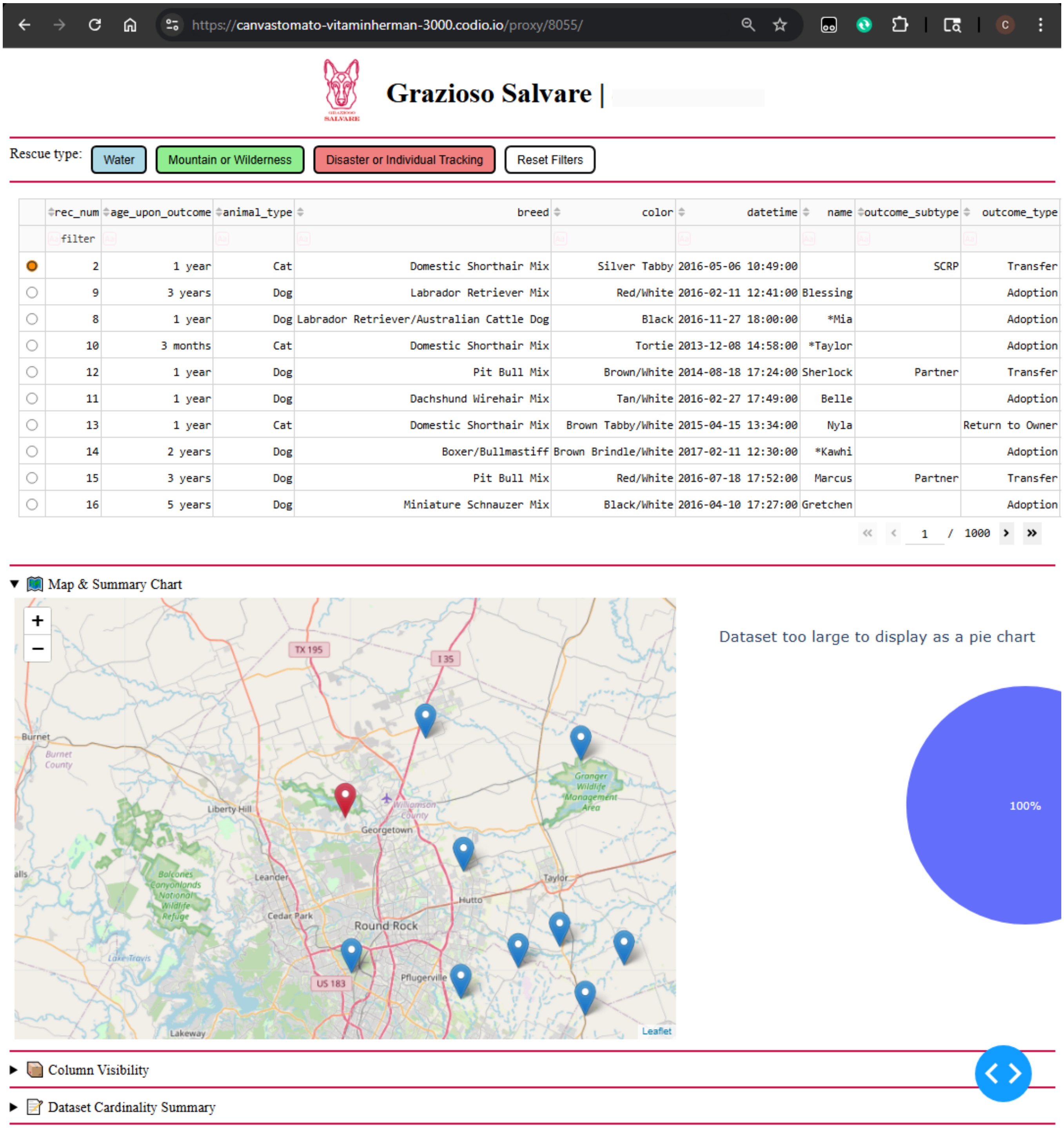
```
Connection to MongoDB established successfully
```

Using the Dashboard: Testing & Deployment Results

When the user connects to the dashboard, the table initially displays the unfiltered results from the entire database. Since the table shows 10 records per page and provides up to 1,000 pages, this indicates that approximately 9,991 to 10,000 records are available. This volume of data is too large to be meaningfully represented in a pie chart; therefore, the application displays the error message “Dataset too large to display as a pie chart.”

The map displays the geolocation of the 10 records shown on the current page, with the active entry highlighted in red (orange button, rec_num = 2) and the remaining entries shown in blue.

The two bottom rows are untoggled by default: Column Visibility and Dataset Cardinality Summary. Their functionality is described in the following sections.



https://canvastomato-vitaminherman-3000.codio.io/proxy/8055/

Grazioso Salvare |

Rescue type: Water | Mountain or Wilderness | Disaster or Individual Tracking | Reset Filters

	rec_num	age_upon_outcome	animal_type	breed	color	datetime	name	outcome_subtype	outcome_type
	filter								
●	2	1 year	Cat	Domestic Shorthair Mix	Silver Tabby	2016-05-06 10:49:00		SCRP	Transfer
○	9	3 years	Dog	Labrador Retriever Mix	Red/White	2016-02-11 12:41:00	Blessing		Adoption
○	8	1 year	Dog	Labrador Retriever/Australian Cattle Dog	Black	2016-11-27 18:00:00	*Mia		Adoption
○	10	3 months	Cat	Domestic Shorthair Mix	Tortie	2013-12-08 14:58:00	*Taylor		Adoption
○	12	1 year	Dog	Pit Bull Mix	Brown/White	2014-08-18 17:24:00	Sherlock	Partner	Transfer
○	11	1 year	Dog	Dachshund Wirehair Mix	Tan/White	2016-02-27 17:49:00	Belle		Adoption
○	13	1 year	Cat	Domestic Shorthair Mix	Brown Tabby/White	2015-04-15 13:34:00	Nyla		Return to Owner
○	14	2 years	Dog	Boxer/Bullmastiff	Brown Brindle/White	2017-02-11 12:30:00	*Kawhi		Adoption
○	15	3 years	Dog	Pit Bull Mix	Red/White	2016-07-18 17:52:00	Marcus	Partner	Transfer
○	16	5 years	Dog	Miniature Schnauzer Mix	Black/White	2016-04-10 17:27:00	Gretchen		Adoption

« ‹ 1 / 1000 › »

▼ Map & Summary Chart

Dataset too large to display as a pie chart

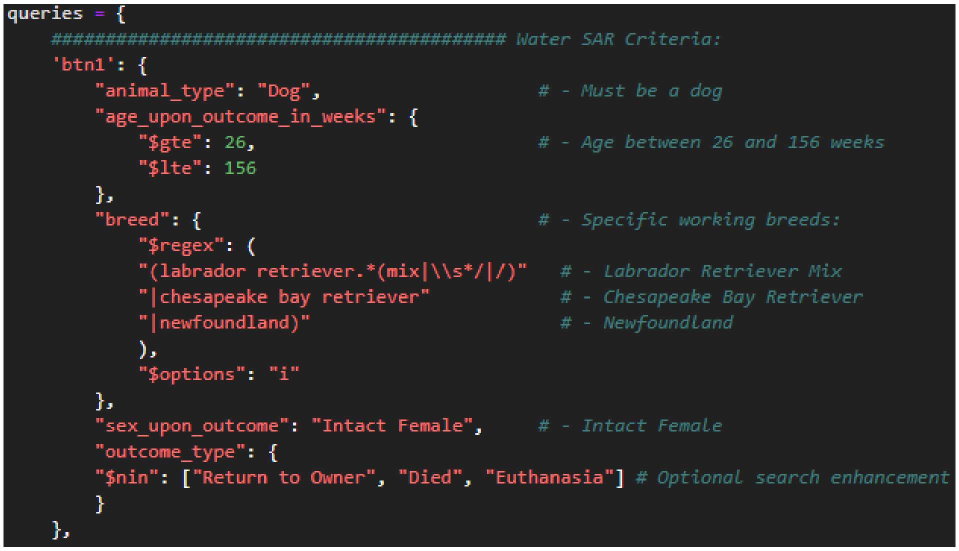
▶ Column Visibility

▶ Dataset Cardinality Summary

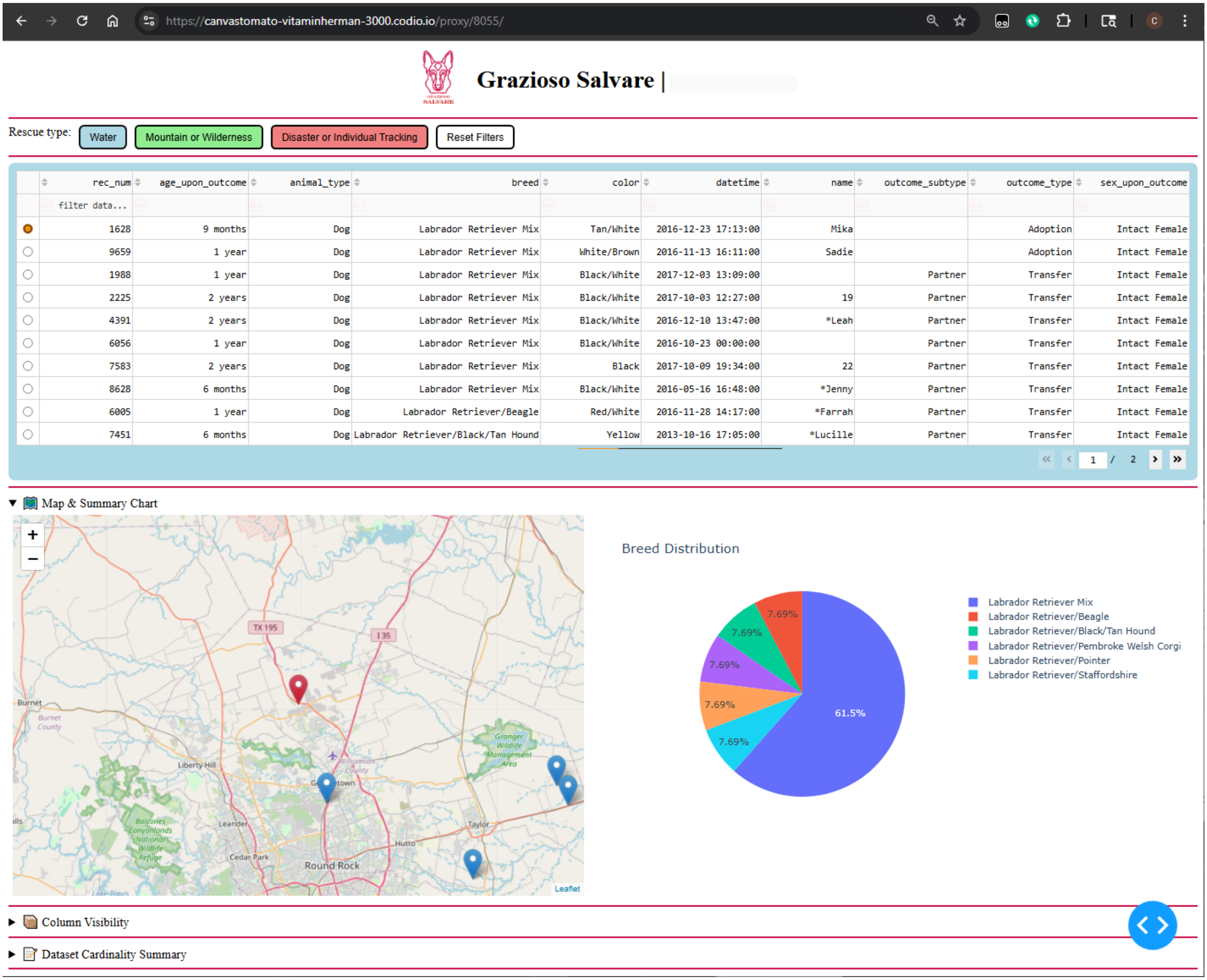
Clicking on the “Water” button will return the results of the query with the filters to select only good candidates for the Water Search and Rescue training (2 pages of records, which means up to 20 candidates).

The map shows the position of these first 10 records, and the pie chart offers a summary of the different breeds found with the search.

To note how the background of the table changes to match the blue color of the button.



```
queries = {
    ############################################ Water SAR Criteria:
    'btn1': {
        "animal_type": "Dog",                        # - Must be a dog
        "age_upon_outcome_in_weeks": {
            "$gte": 26,                              # - Age between 26 and 156 weeks
            "$lte": 156
        },
        "breed": {                                   # - Specific working breeds:
            "$regex": (
            "(labrador retriever.*(mix|\\s*/|/)"       # - Labrador Retriever Mix
            "|chesapeake bay retriever"                # - Chesapeake Bay Retriever
            "|newfoundland)"                           # - Newfoundland
            ),
            "$options": "i"
        },
        "sex_upon_outcome": "Intact Female",         # - Intact Female
        "outcome_type": {
        "$nin": ["Return to Owner", "Died", "Euthanasia"] # Optional search enhancement
        }
    },
```



https://canvastomato-vitaminherman-3000.codio.io/proxy/8055/

Grazioso Salvare |

Rescue type: Water | Mountain or Wilderness | Disaster or Individual Tracking | Reset Filters

	rec_num	age_upon_outcome	animal_type	breed	color	datetime	name	outcome_subtype	outcome_type	sex_upon_outcome
	filter data...									
●	1628	9 months	Dog	Labrador Retriever Mix	Tan/White	2016-12-23 17:13:00	Mika		Adoption	Intact Female
○	9659	1 year	Dog	Labrador Retriever Mix	White/Brown	2016-11-13 16:11:00	Sadie		Adoption	Intact Female
○	1988	1 year	Dog	Labrador Retriever Mix	Black/White	2017-12-03 13:09:00		Partner	Transfer	Intact Female
○	2225	2 years	Dog	Labrador Retriever Mix	Black/White	2017-10-03 12:27:00	19	Partner	Transfer	Intact Female
○	4391	2 years	Dog	Labrador Retriever Mix	Black/White	2016-12-10 13:47:00	*Leah	Partner	Transfer	Intact Female
○	6056	1 year	Dog	Labrador Retriever Mix	Black/White	2016-10-23 00:00:00		Partner	Transfer	Intact Female
○	7583	2 years	Dog	Labrador Retriever Mix	Black	2017-10-09 19:34:00	22	Partner	Transfer	Intact Female
○	8628	6 months	Dog	Labrador Retriever Mix	Black/White	2016-05-16 16:48:00	*Jenny	Partner	Transfer	Intact Female
○	6005	1 year	Dog	Labrador Retriever/Beagle	Red/White	2016-11-28 14:17:00	*Farrah	Partner	Transfer	Intact Female
○	7451	6 months	Dog	Labrador Retriever/Black/Tan Hound	Yellow	2013-10-16 17:05:00	*Lucille	Partner	Transfer	Intact Female

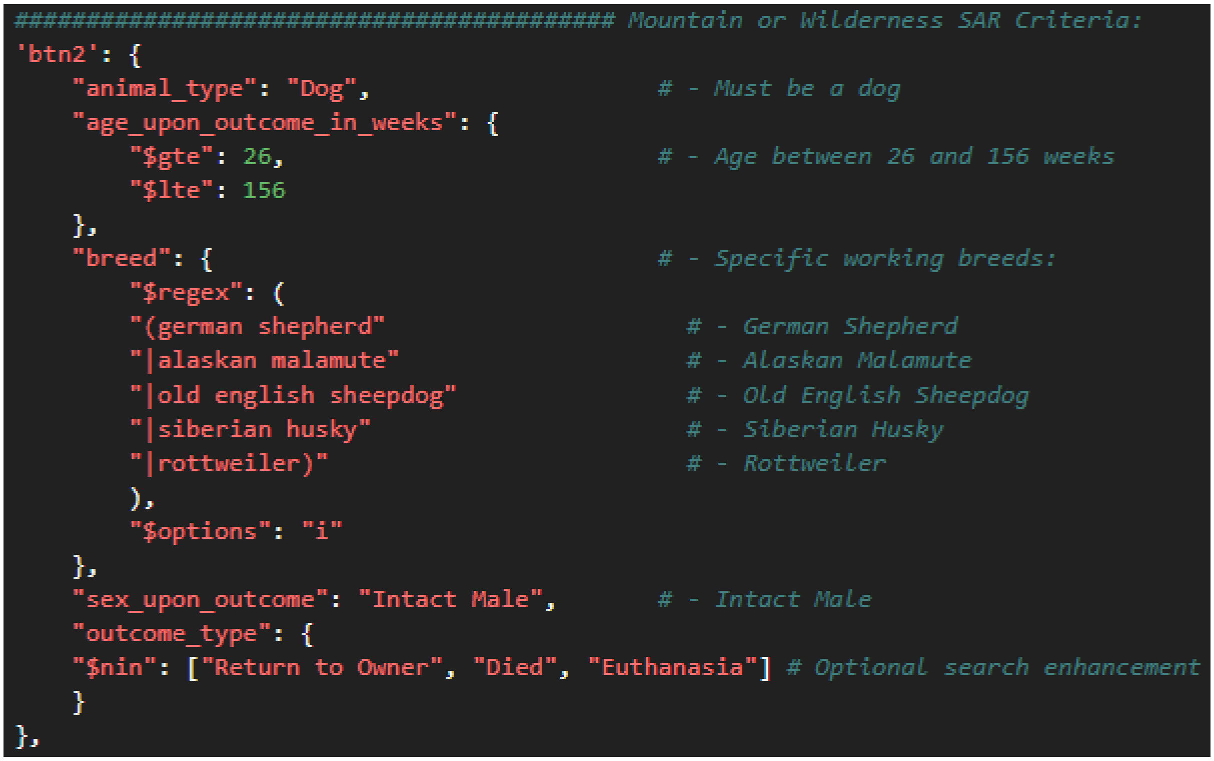
▼ Map & Summary Chart

+
−
TX 195
I 35
Burnet
Burnet County
Liberty Hill
Williamson County
Granger Wildlife Management Area
Georgetown
Leander
Taylor
Balcones Canyonlands National Wildlife Refuge
Cedar Park
Hutto
Round Rock
Lake Travis
Leaflet
Breed Distribution
7.69%
7.69%
7.69%
7.69%
7.69%
61.5%
Labrador Retriever Mix
Labrador Retriever/Beagle
Labrador Retriever/Black/Tan Hound
Labrador Retriever/Pembroke Welsh Corgi
Labrador Retriever/Pointer
Labrador Retriever/Staffordshire

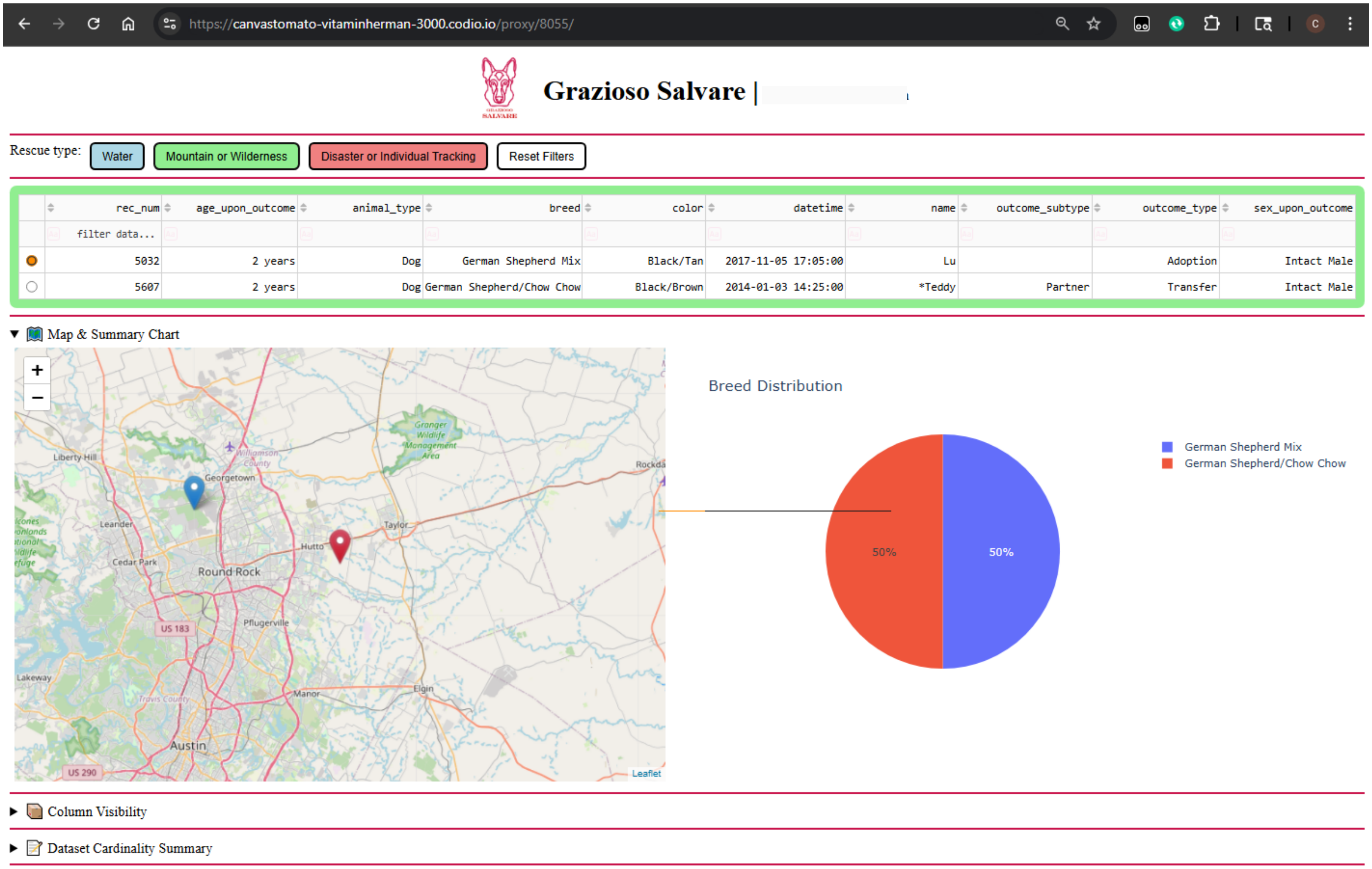
▶ Column Visibility

▶ Dataset Cardinality Summary

The following screenshots show the results for the “Mountain or Wilderness” SAR search, the “Disaster or Individual Tracking” one, and the results after pushing the “Reset Filters” button.

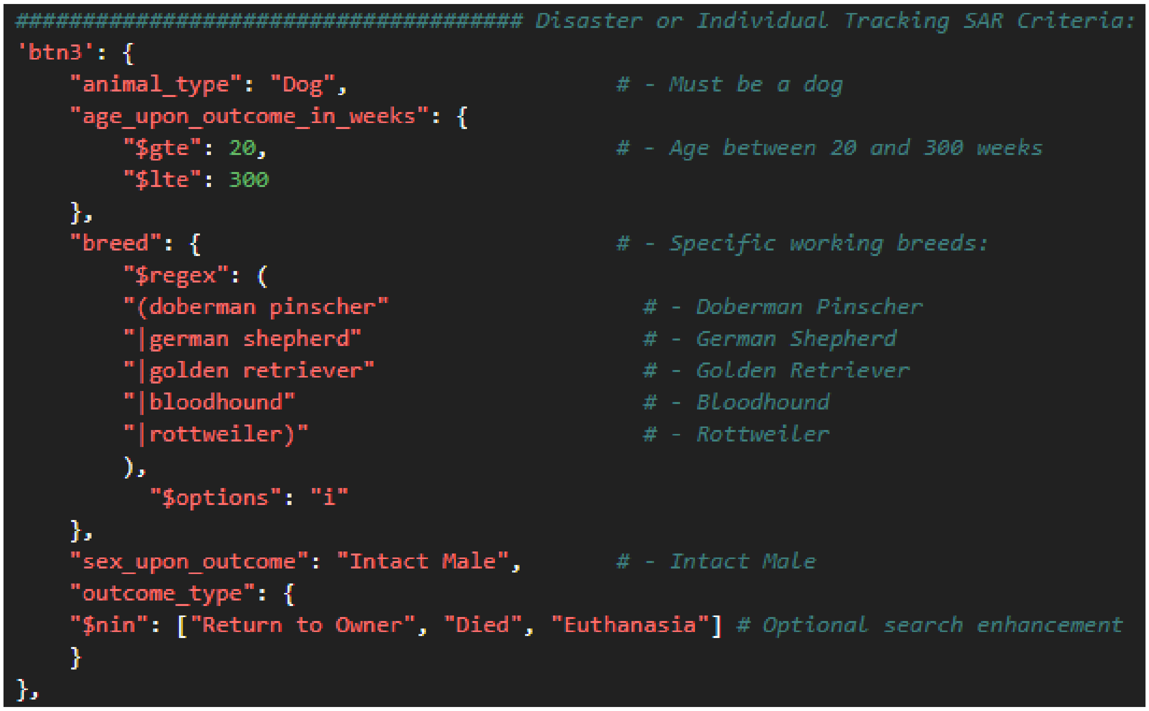


```
############################################ Mountain or Wilderness SAR Criteria:
'btn2': {
    "animal_type": "Dog",                   # - Must be a dog
    "age_upon_outcome_in_weeks": {
        "$gte": 26,                         # - Age between 26 and 156 weeks
        "$lte": 156
    },
    "breed": {                              # - Specific working breeds:
        "$regex": (
        "(german shepherd"                    # - German Shepherd
        "|alaskan malamute"                   # - Alaskan Malamute
        "|old english sheepdog"               # - Old English Sheepdog
        "|siberian husky"                     # - Siberian Husky
        "|rottweiler)"                        # - Rottweiler
        ),
        "$options": "i"
    },
    "sex_upon_outcome": "Intact Male",      # - Intact Male
    "outcome_type": {
    "$nin": ["Return to Owner", "Died", "Euthanasia"] # Optional search enhancement
    }
},
```

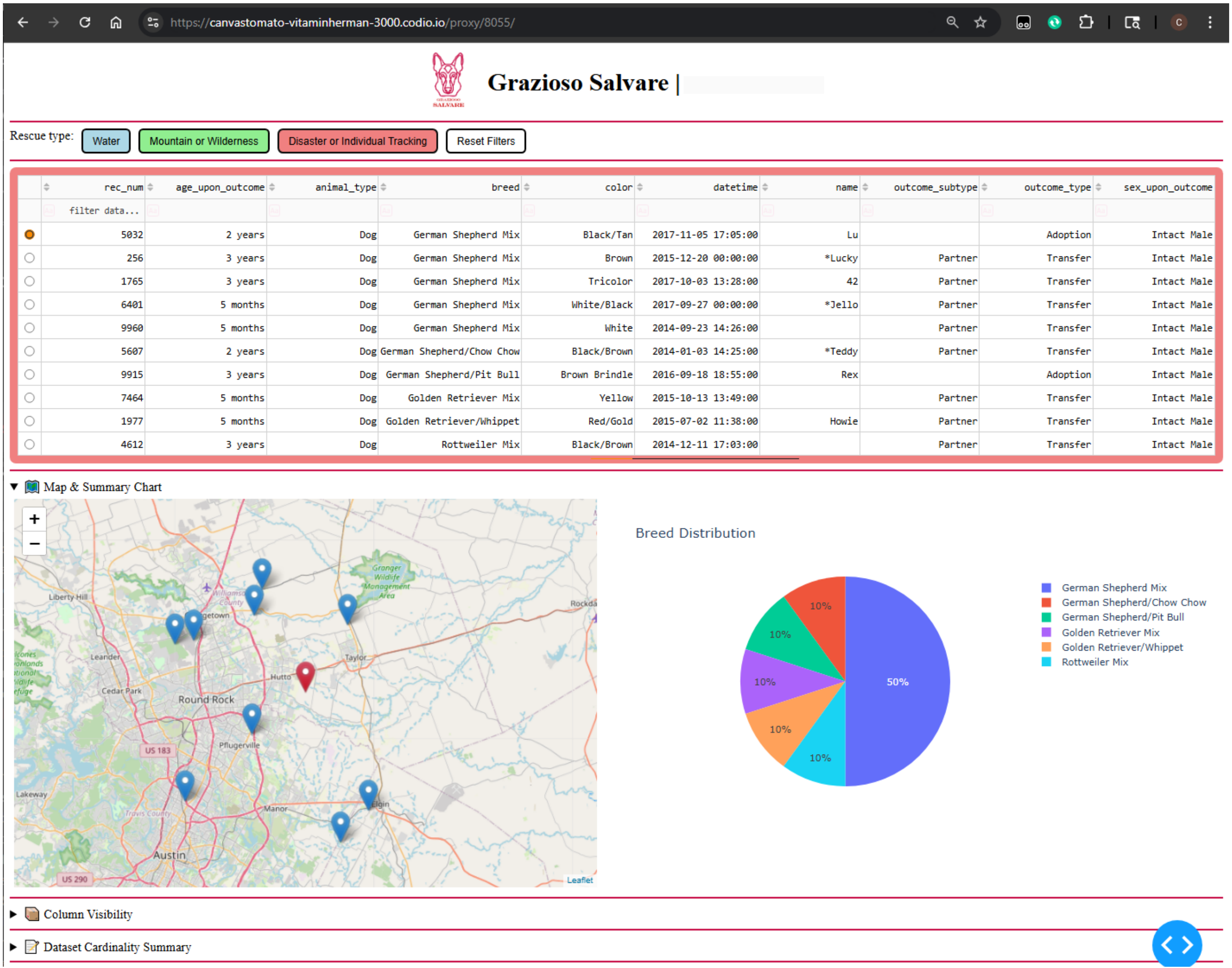


https://canvastomato-vitaminherman-3000.codio.io/proxy/8055/
Grazioso Salvare |
Rescue type:
Water
Mountain or Wilderness
Disaster or Individual Tracking
Reset Filters

	rec_num	age_upon_outcome	animal_type	breed	color	datetime	name	outcome_subtype	outcome_type	sex_upon_outcome
	filter data...									
●	5032	2 years	Dog	German Shepherd Mix	Black/Tan	2017-11-05 17:05:00	Lu		Adoption	Intact Male
○	5607	2 years	Dog	German Shepherd/Chow Chow	Black/Brown	2014-01-03 14:25:00	*Teddy	Partner	Transfer	Intact Male



```
######################################## Disaster or Individual Tracking SAR Criteria:
'btn3': {
    "animal_type": "Dog",                    # - Must be a dog
    "age_upon_outcome_in_weeks": {
        "$gte": 20,                          # - Age between 20 and 300 weeks
        "$lte": 300
    },
    "breed": {                               # - Specific working breeds:
        "$regex": (
        "(doberman pinscher"                   # - Doberman Pinscher
        "|german shepherd"                     # - German Shepherd
        "|golden retriever"                    # - Golden Retriever
        "|bloodhound"                          # - Bloodhound
        "|rottweiler)"                         # - Rottweiler
        ),
          "$options": "i"
    },
    "sex_upon_outcome": "Intact Male",       # - Intact Male
    "outcome_type": {
    "$nin": ["Return to Owner", "Died", "Euthanasia"] # Optional search enhancement
    }
},
```


https://canvastomato-vitaminherman-3000.codio.io/proxy/8055/
GRAZIOSO SALVARE

Grazioso Salvare |

Rescue type: Water | Mountain or Wilderness | Disaster or Individual Tracking | Reset Filters

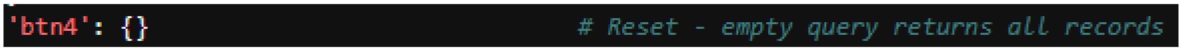
	rec_num	age_upon_outcome	animal_type	breed	color	datetime	name	outcome_subtype	outcome_type	sex_upon_outcome
	filter data...									
●	5032	2 years	Dog	German Shepherd Mix	Black/Tan	2017-11-05 17:05:00	Lu		Adoption	Intact Male
○	256	3 years	Dog	German Shepherd Mix	Brown	2015-12-20 00:00:00	*Lucky	Partner	Transfer	Intact Male
○	1765	3 years	Dog	German Shepherd Mix	Tricolor	2017-10-03 13:28:00	42	Partner	Transfer	Intact Male
○	6401	5 months	Dog	German Shepherd Mix	White/Black	2017-09-27 00:00:00	*Jello	Partner	Transfer	Intact Male
○	9960	5 months	Dog	German Shepherd Mix	White	2014-09-23 14:26:00		Partner	Transfer	Intact Male
○	5607	2 years	Dog	German Shepherd/Chow Chow	Black/Brown	2014-01-03 14:25:00	*Teddy	Partner	Transfer	Intact Male
○	9915	3 years	Dog	German Shepherd/Pit Bull	Brown Brindle	2016-09-18 18:55:00	Rex		Adoption	Intact Male
○	7464	5 months	Dog	Golden Retriever Mix	Yellow	2015-10-13 13:49:00		Partner	Transfer	Intact Male
○	1977	5 months	Dog	Golden Retriever/Whippet	Red/Gold	2015-07-02 11:38:00	Howie	Partner	Transfer	Intact Male
○	4612	3 years	Dog	Rottweiler Mix	Black/Brown	2014-12-11 17:03:00		Partner	Transfer	Intact Male

▼ Map & Summary Chart

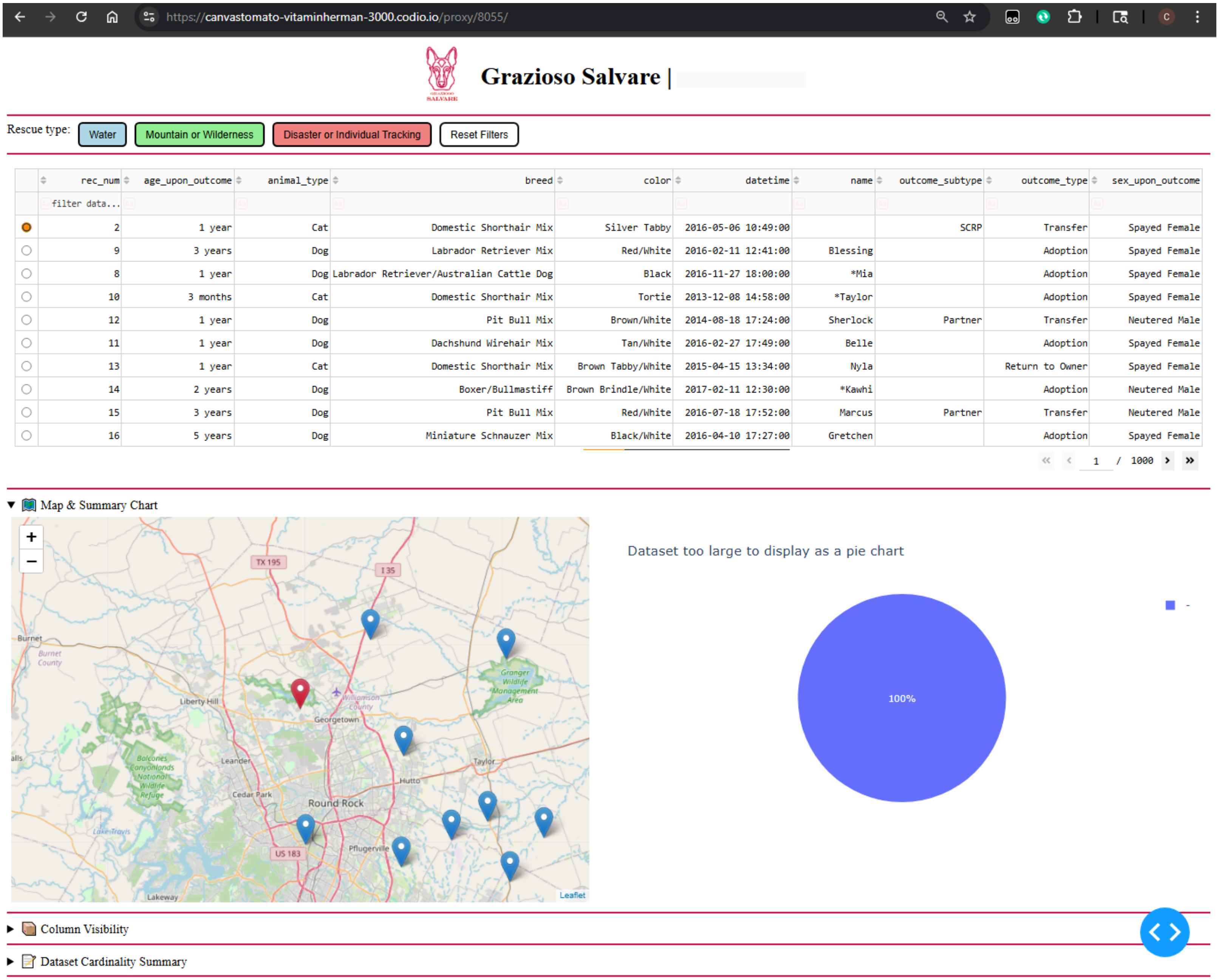
▶ Column Visibility

▶ Dataset Cardinality Summary

Pressing the button “Reset Filters” removes all the previous filters, and returns all the records of the database. Note how the background of the table is now white, like the color of the button.



```
'btn4': {}                    # Reset - empty query returns all records
```



https://canvastomato-vitaminherman-3000.codio.io/proxy/8055/

Grazioso Salvare |

Rescue type: Water | Mountain or Wilderness | Disaster or Individual Tracking | Reset Filters

	rec_num	age_upon_outcome	animal_type	breed	color	datetime	name	outcome_subtype	outcome_type	sex_upon_outcome
	filter data...									
●	2	1 year	Cat	Domestic Shorthair Mix	Silver Tabby	2016-05-06 10:49:00		SCRP	Transfer	Spayed Female
○	9	3 years	Dog	Labrador Retriever Mix	Red/White	2016-02-11 12:41:00	Blessing		Adoption	Spayed Female
○	8	1 year	Dog	Labrador Retriever/Australian Cattle Dog	Black	2016-11-27 18:00:00	*Mia		Adoption	Spayed Female
○	10	3 months	Cat	Domestic Shorthair Mix	Tortie	2013-12-08 14:58:00	*Taylor		Adoption	Spayed Female
○	12	1 year	Dog	Pit Bull Mix	Brown/White	2014-08-18 17:24:00	Sherlock	Partner	Transfer	Neutered Male
○	11	1 year	Dog	Dachshund Wirehair Mix	Tan/White	2016-02-27 17:49:00	Belle		Adoption	Spayed Female
○	13	1 year	Cat	Domestic Shorthair Mix	Brown Tabby/White	2015-04-15 13:34:00	Nyla		Return to Owner	Spayed Female
○	14	2 years	Dog	Boxer/Bullmastiff	Brown Brindle/White	2017-02-11 12:30:00	*Kawhi		Adoption	Neutered Male
○	15	3 years	Dog	Pit Bull Mix	Red/White	2016-07-18 17:52:00	Marcus	Partner	Transfer	Neutered Male
○	16	5 years	Dog	Miniature Schnauzer Mix	Black/White	2016-04-10 17:27:00	Gretchen		Adoption	Spayed Female

« ‹ 1 / 1000 › »

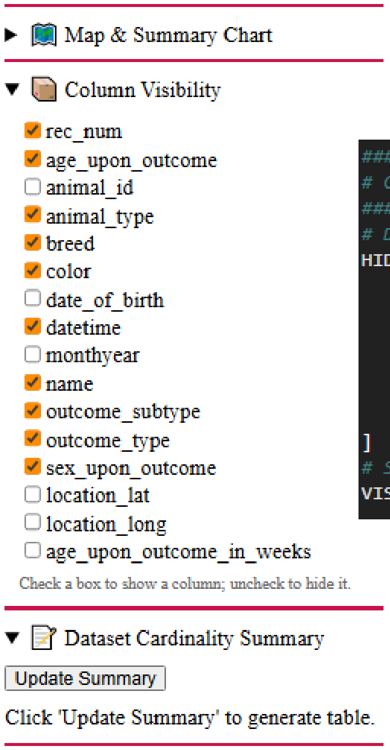
▼ Map & Summary Chart

Dataset too large to display as a pie chart

▶ Column Visibility

▶ Dataset Cardinality Summary

From the “Column Visibility” row, once expanded, it is possible to select the visibility of the various columns of the database. Some of them are currently hidden to improve the UX and use the space on screen to show only relevant data.



▶ Map & Summary Chart

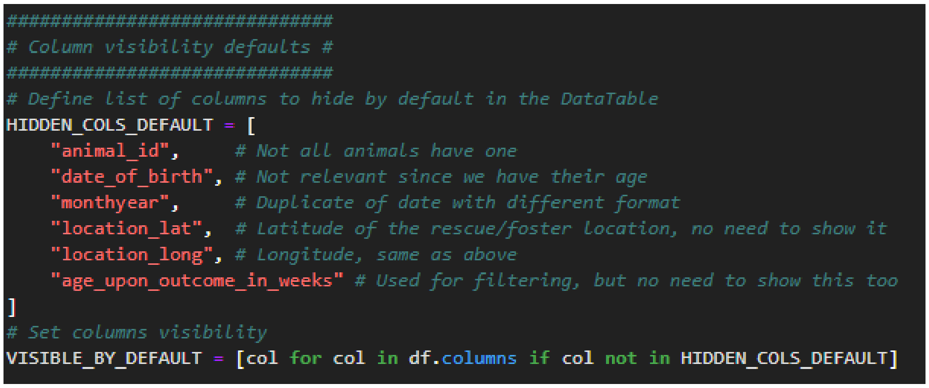
▼ Column Visibility

☑ rec_num
☑ age_upon_outcome
☐ animal_id
☑ animal_type
☑ breed
☑ color
☐ date_of_birth
☑ datetime
☐ monthyear
☑ name
☑ outcome_subtype
☑ outcome_type
☑ sex_upon_outcome
☐ location_lat
☐ location_long
☐ age_upon_outcome_in_weeks

Check a box to show a column; uncheck to hide it.

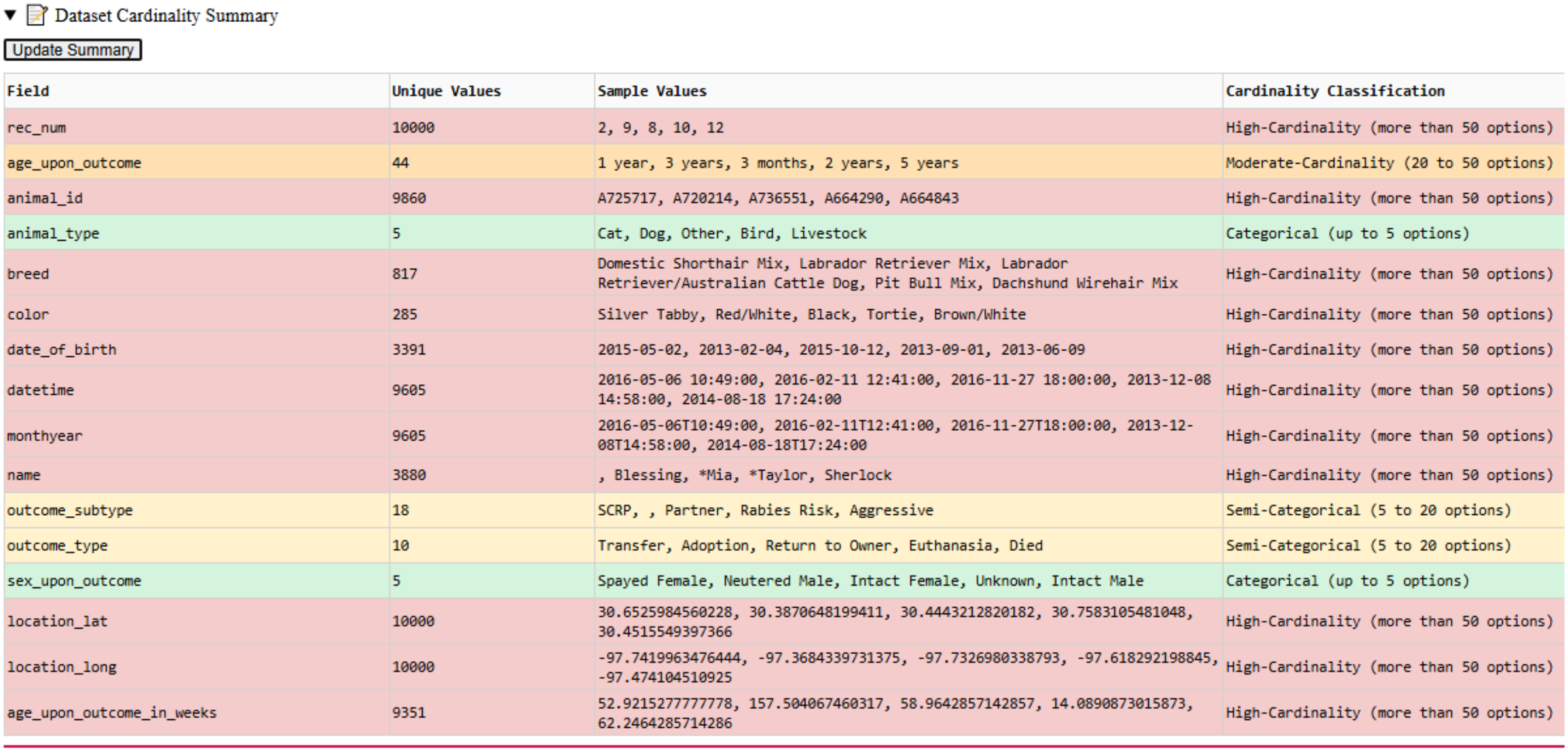
▼ Dataset Cardinality Summary

Click 'Update Summary' to generate table.



```
###############################
# Column visibility defaults #
###############################
# Define list of columns to hide by default in the DataTable
HIDDEN_COLS_DEFAULT = [
    "animal_id",     # Not all animals have one
    "date_of_birth", # Not relevant since we have their age
    "monthyear",     # Duplicate of date with different format
    "location_lat",  # Latitude of the rescue/foster location, no need to show it
    "location_long", # Longitude, same as above
    "age_upon_outcome_in_weeks" # Used for filtering, but no need to show this too
]
# Set columns visibility
VISIBLE_BY_DEFAULT = [col for col in df.columns if col not in HIDDEN_COLS_DEFAULT]
```

On the last row, the Dataset Cardinality Summary provides a high-level overview of the dataset. The summary table is generated only when the user clicks the Update Summary button. After the first generation, the results are stored in cache memory to reduce computational overhead and improve responsiveness during subsequent updates. This table helps users better understand the structure and distribution of the data, identify patterns, and assess dataset composition at a glance.



▼ Dataset Cardinality Summary

Field	Unique Values	Sample Values	Cardinality Classification
rec_num	10000	2, 9, 8, 10, 12	High-Cardinality (more than 50 options)
age_upon_outcome	44	1 year, 3 years, 3 months, 2 years, 5 years	Moderate-Cardinality (20 to 50 options)
animal_id	9860	A725717, A720214, A736551, A664290, A664843	High-Cardinality (more than 50 options)
animal_type	5	Cat, Dog, Other, Bird, Livestock	Categorical (up to 5 options)
breed	817	Domestic Shorthair Mix, Labrador Retriever Mix, Labrador Retriever/Australian Cattle Dog, Pit Bull Mix, Dachshund Wirehair Mix	High-Cardinality (more than 50 options)
color	285	Silver Tabby, Red/White, Black, Tortie, Brown/White	High-Cardinality (more than 50 options)
date_of_birth	3391	2015-05-02, 2013-02-04, 2015-10-12, 2013-09-01, 2013-06-09	High-Cardinality (more than 50 options)
datetime	9605	2016-05-06 10:49:00, 2016-02-11 12:41:00, 2016-11-27 18:00:00, 2013-12-08 14:58:00, 2014-08-18 17:24:00	High-Cardinality (more than 50 options)
monthyear	9605	2016-05-06T10:49:00, 2016-02-11T12:41:00, 2016-11-27T18:00:00, 2013-12-08T14:58:00, 2014-08-18T17:24:00	High-Cardinality (more than 50 options)
name	3880	, Blessing, *Mia, *Taylor, Sherlock	High-Cardinality (more than 50 options)
outcome_subtype	18	SCRP, , Partner, Rabies Risk, Aggressive	Semi-Categorical (5 to 20 options)
outcome_type	10	Transfer, Adoption, Return to Owner, Euthanasia, Died	Semi-Categorical (5 to 20 options)
sex_upon_outcome	5	Spayed Female, Neutered Male, Intact Female, Unknown, Intact Male	Categorical (up to 5 options)
location_lat	10000	30.6525984560228, 30.3870648199411, 30.4443212820182, 30.7583105481048, 30.4515549397366	High-Cardinality (more than 50 options)
location_long	10000	-97.7419963476444, -97.3684339731375, -97.7326980338793, -97.618292198845, -97.474104510925	High-Cardinality (more than 50 options)
age_upon_outcome_in_weeks	9351	52.9215277777778, 157.504067460317, 58.9642857142857, 14.0890873015873, 62.2464285714286	High-Cardinality (more than 50 options)

Challenges & Solutions

After having documented the steps taken during Project One and Project Two to provide Grazioso Salvare with a full-stack development application, it is enjoyable to look back and reflect on the various obstacles I had to overcome.

Writing the CRUD module itself was straightforward, as it mostly involved implementing standard create, read, update, and delete operations against MongoDB. The more challenging part was designing the testing file, which had to verify that each CRUD operation behaved correctly under different scenarios. I missed having the guidance of JUnit to achieve full test coverage of my code, but the benefit has been encouraging me to think independently about possible ways to verify my code. I am particularly proud of my idea of verifying that my code did not leave any footprints of testing records by verifying the number of documents before and after the tests. By no means rocket science, and it is also entirely possible that I read about it somewhere and then forgot this method, but I enjoy implementing simple yet effective solutions whenever and wherever possible.

Creating the dashboard has been even more fun because HTML is the very first language I learned twenty or more years ago, and I did not have many opportunities to use it during my studies here. However, even this pleasant activity did not come without puzzling moments, like when I noticed the "Toggle Columns" button for the first time, and I was not sure where it came from. It turned out that Dash creates that button automatically when we hide some columns, and it has been fun finding a solution and finally hiding it with the injection of some custom CSS code.

Another small challenge has been deciding what to do with the large dataset and the pie chart. Visualizing hundreds of slices at the 0.01% was not useful, so I decided to exclude it altogether if not to display results from the very specific queries.

My initial idea was to add the ability to filter results by radius, but since the SAR queries returned only a small number of records (one or two pages at most), I decided to implement two different markers to display all results on the map: red for the active row, and blue for the others. At the start, I initially displayed all 10,000 records at once, which nearly caused my browser to freeze. This was later fixed by using `derived_viewport_data` instead of the initial `derived_virtual_data` as the input for populating the table.