# README – Grazioso Salvare Dashboard

## About

This dashboard was created to provide Grazioso Salvare with an intuitive interface with their Austin Animal Shelter (AAC) database. The dashboard is the view part of the Model View Controller (MVC) design pattern that connects the AAC database with the Grazio Salvare dashboard. The code is designed to be easily updated to add features to the dashboard, such as charts, graphs, or custom search options.

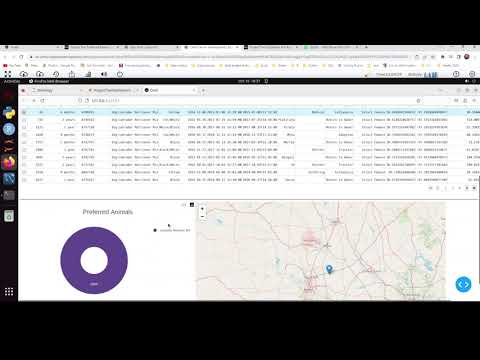
## Usage

Using the dashboard is simple. When the it loads, the data table in the dashboard is populated with the entire dataset. The results are limited to ten per page and each column can be sorted in ascending or descending order. Additionally, radio buttons have been added to sort by the rescue animal types laid out by Grazioso Salvare.

Under the data table, there is a donut chart representing the proportion of each breed in the current data table selection. There is also a geolocation chart that shows the location of the selected animal, along with their breed and name.

Finally, clicking on the Grazioso Salvare logo at the top of the dashboard will take you to their homepage at [www.snhu.edu](http://www.snhu.edu).

Here is a video walkthrough of the dashboard and functionality:

[](https://www.youtube.com/embed/tJ9KHwAtOHk?feature=oembed)

## Installation

This project uses MongoDB for the database, Python for the coding, Dash for the dashboard and datable, Pandas for the data, and Plotly and Dash Leaflet for the charts. Additionally, it uses AnimalShelter, a custom Python class created for interaction between the database and dashboard. The AnimalShelter python class and README can be accessed here:

<https://github.com/dfrance2010/CS340>

MongoDB was chosen because of its capabilities as a NoSQL database and its ability to interact with python using PyMongo. NoSQL allows for data that is less structured than traditional SQL tables. The Dash framework allows for simple interaction between the web-based dashboard and the AnimalShelter Python class. Combined, they form the different components of the MVC design pattern, with MongoDB being the model, Dash the view, and AnimalShelter the controller.

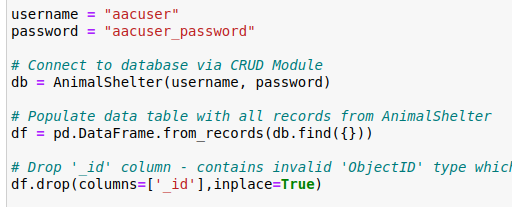
Additional applications that were used include [Jupyter Notebook](https://jupyter.org/) for the coding and [OBS](https://obsproject.com/) for the README screencast.

## Getting Started

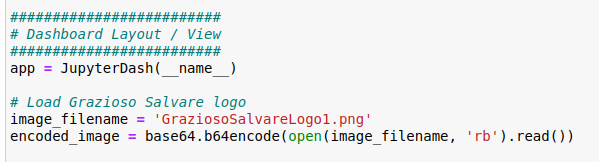
To set up the dashboard code, start by importing the necessary libraries:



Create an AnimalShelter object, then run a general find() query to populate a Pandas data frame with all the documents in the Austin Animal Shelter database. The ‘\_id’ column needs to be dropped because it returns an ObjectID type that is incompatible with the dashboard:



Create a dashboard object using JupyterDash. The optional image is for the logo at the top of the dashboard:



The dashboard layout uses the Dash framework. It can be customized using HTML elements. This is what the code looks like to create the Grazioso Salvare dashboard:



Creating the data table:

A computer screen shot of a computer code

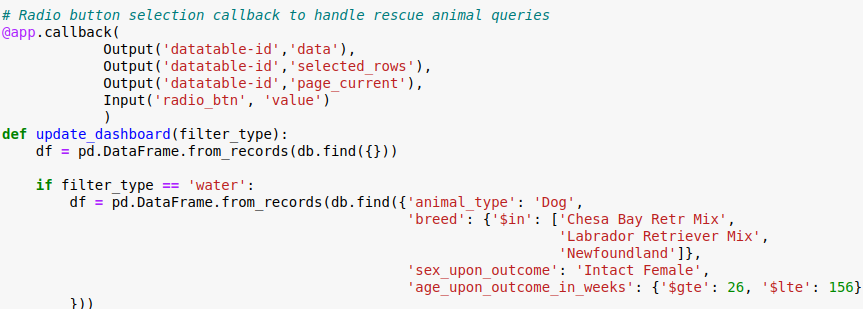
Description automatically generated

Creating space for the pie chart and geolocation map:

A screenshot of a computer code

Description automatically generated

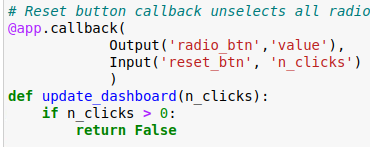
Create a callback for handling the radio buttons that takes in the value from the group and returns a new data frame. The ‘selected\_rows’ and ‘page\_current’ return values are to make sure the first row in the new data frame is selected. If you wanted to change the parameters for any category, this is where you would do it. For example, adding in the various mixes of the selected dogs could be done by adding them to the array for ‘breed’:



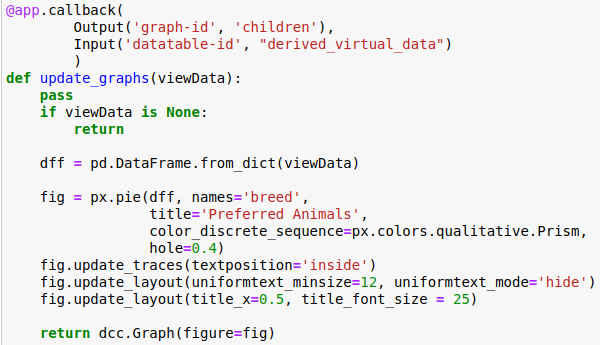




Setting the ‘radio\_btn’ value to ‘False’ means none of the buttons are selected. This method is called each time the ‘Reset’ button is clicked:



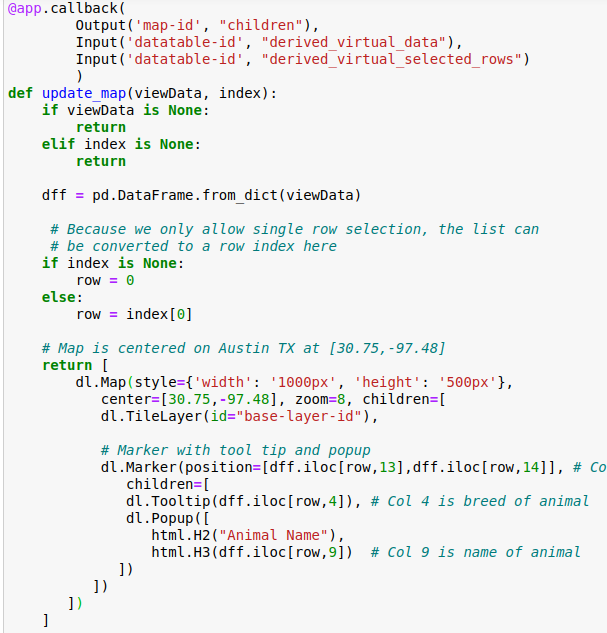
The pie chart uses plotly.express. Creating a different type of chart or customizing the current one would be done in the update\_graphs callback method. For more details visit the [Plotly Express website](https://plotly.com/python/plotly-express/):



To highlight the selected row, this callback is used. Changing the highlight color can be done here:



The update\_map() callback creates the geolocation map. It updates each time the data table changes or a row is selected. The map uses Dash Leaflet, documentation can be found at the [Dash Leaflet website](https://www.dash-leaflet.com/):



## Contact

David France

[david.france@snhu.edu](mailto:david.france@snhu.edu)