Final Project README

J. F. Kiddy

Southern New Hampshire University

**Project Title**

Grazioso Salvare International Rescue-Animal Animal Shelter Candidate Locator by Global Rain.

**About the Project**

Grazioso Salvare was seeking an innovative software application that could interface with existing databases from local animal shelters to identify and categorize available canines that are good candidates for search-and-rescue training. First was the importation of the Austin Animal Control database in a base layer in MongoDB to contain all the raw data from the .CSV files on the different animals that populate the shelter. The Second layer uses Python and PYMongo to create the interacting with MongoDB for Creating, Reading, Updating and Deleting (CRUD) the data in the database. And lastly, the use of Python and Plotly\_Dash to create the GUI interface to be utilized by Grazioso Salvare personnel.

**Motivation**

This project is designed make things easier and stream line the process for Grazioso Salvare personnel to identify canines that are potentially good candidates for search-and-rescue training. When trained, these dogs are able to find and help to rescue humans or other animals, often in life-threatening conditions such as earthquakes, tornados, missing children or terror attacks. The goal of this application’s functionality will allow the client and other search-and-rescue training organizations to utilize this open-source designed application to quickly process large volumes of data to search, navigate and locate possible canine candidates for training.

**Installation**

The software needed to complete this project were the use of a Linux system, and access to MongoDB to store the unfiltered database. A Python GUI using Python 3, such as Jupyter Notebook (to create the script used to create and test the CRUD) and Python Applications such as PYMongo (for interfacing with MongoDB), Jupyter Ploty Dash (to create the GUI)

**Getting Started**

To access the database the user must be able to access the AAC database and be able to access the “animals” collection. This must be done by having the correct credentials to access the user authentication that has been set up. Using the imported Austin Animal Center Outcome data set, $ mongoimport –port XXXXX –db AAC –collection animals –type csv –headerline –file aac\_shelter outcomes.csv. Next, using Jupyter Notebook we created code to developed query actions designed to filter the information contained in MongoDB and created a code to test the query action functionality.

Next using Plotly\_Dash and Jupyter Notebook we created a code for a simple web-based application based on the requirements set forth by Grazioso Salvare. Once the dashboard was created, we incorporated requirements in the form of widgets on the web-based application’s dashboard to give the customer the ability to select attributes required by potential search-and-rescue canines such as breed, age, and sex.

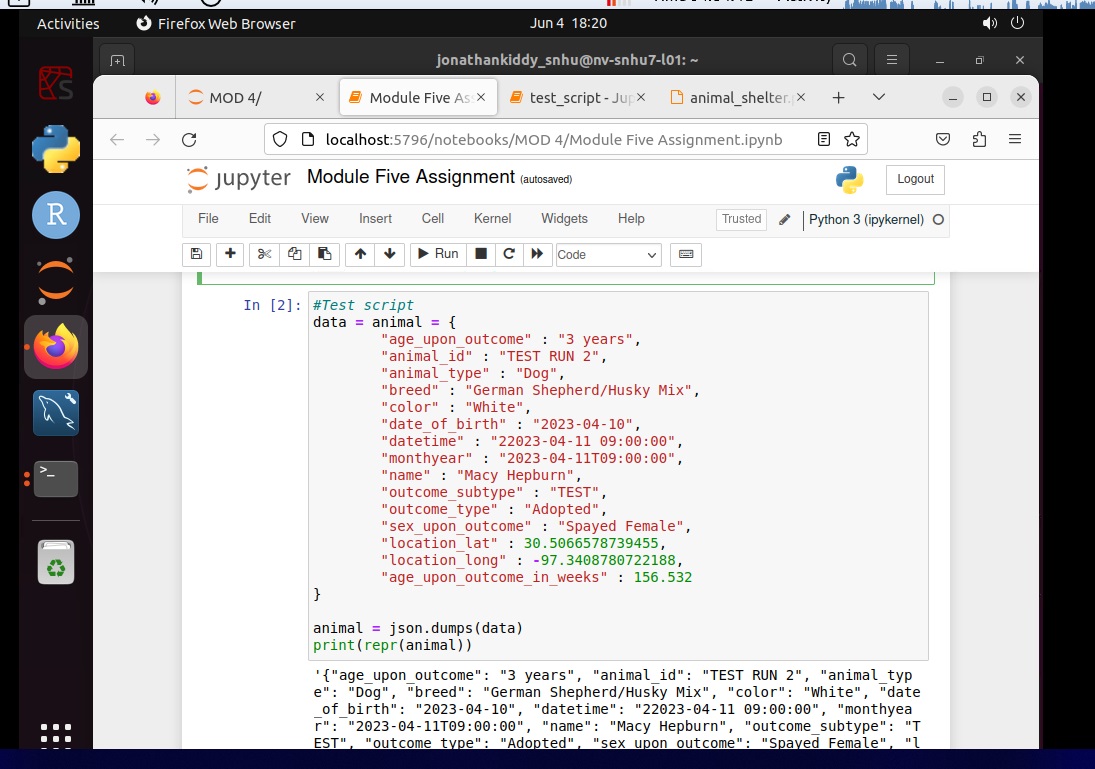
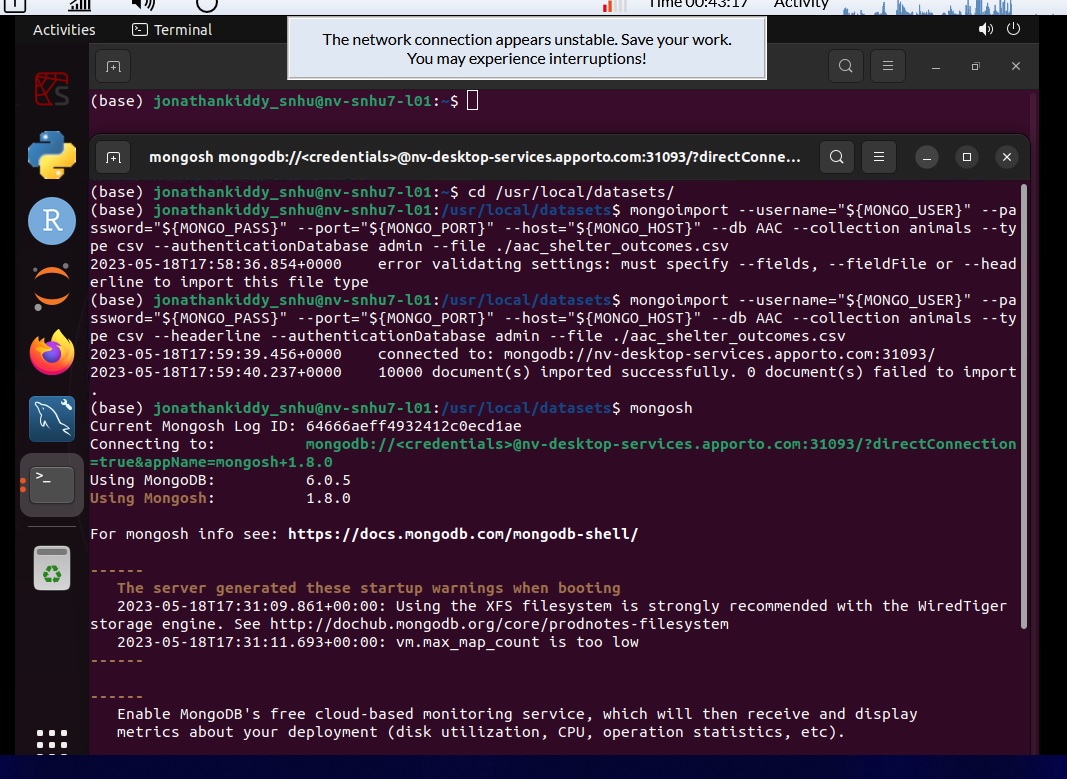
**Usage**

This is to show useful examples of how the project works and how it can be used.

**Code Examples**

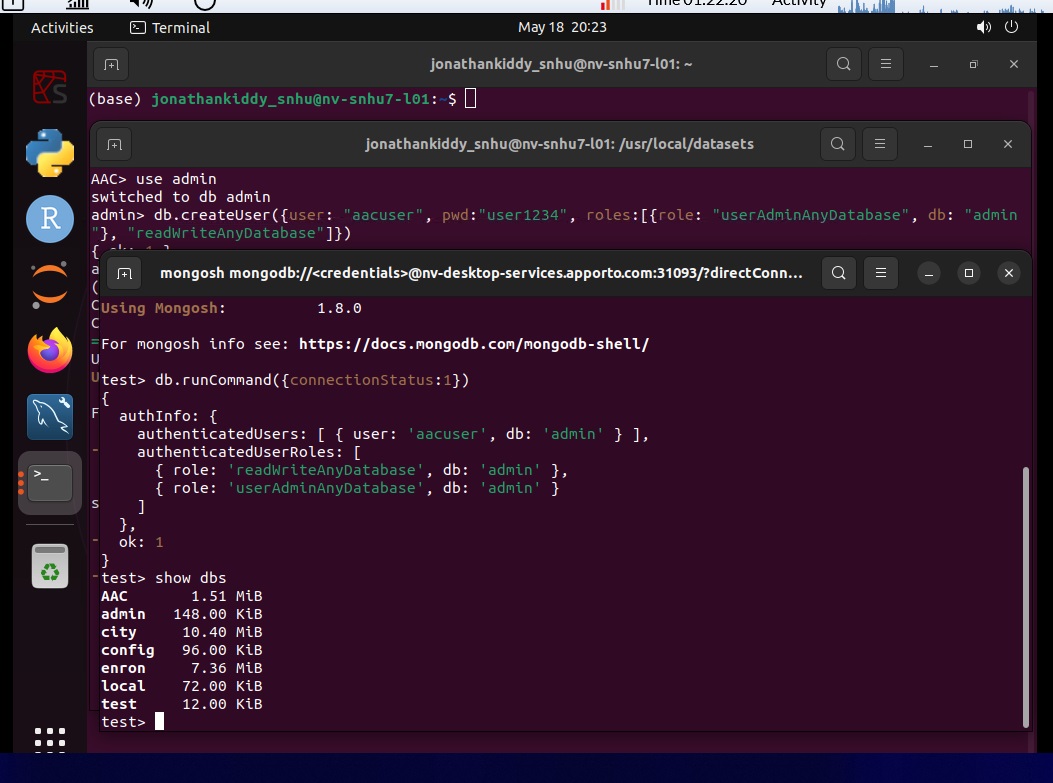
In the figure below, the highlighted section shows the imported Austin Animal Center Outcome data set with a staggering 10,000 records with in. This was imported into MongoDB.

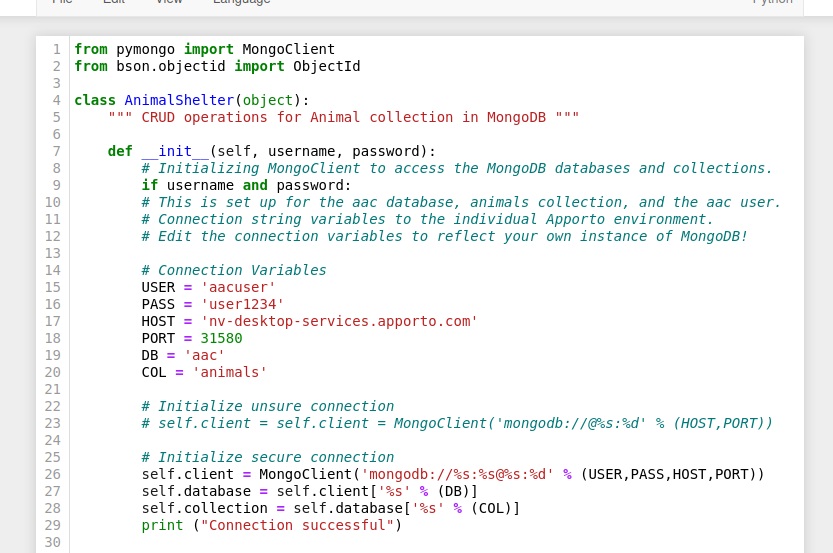


In this figure below, we see the amount of information on each animal.

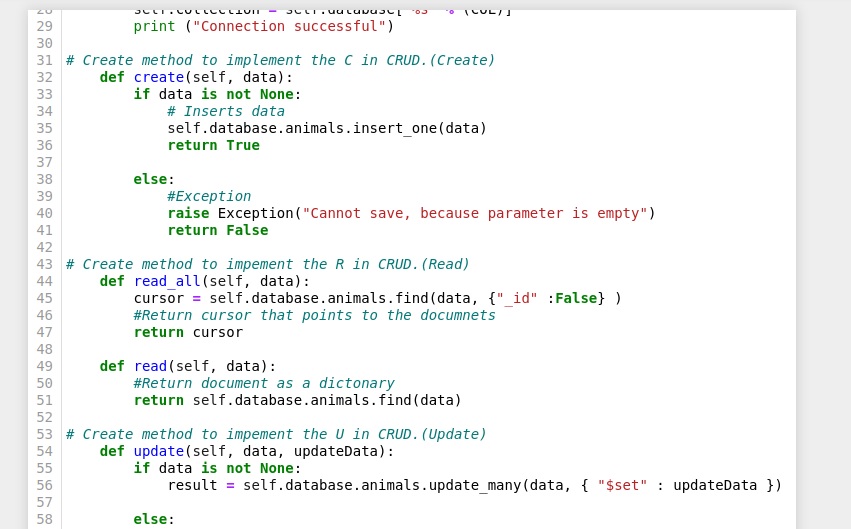


In this figure we see the AAC database secured with an Admin account and the creation of a Username and Password to access the database. In this example the username is “aacuser” and the password is “user1234”.





In this figure, we see the creation of the code to test out the MongoDB database interface.

****

**Tests**

Python script was created to test all the instances of CRUD functionality. Below are examples of the code and the outcomes of each test.

from animal\_shelter import AnimalShelter

shelter = AnimalShelter("aacuser", "user1234")

data = {"1" : 3,

"age\_upon\_outcome" : "2 years",

"animal\_id" : "TEST RUN 1",

"animal\_type" : "Cat",

"breed" : "Domestic Longhair",

"color" : "Grey/White",

"date\_of\_birth" : "2023-04-10",

"datetime" : "22023-04-11 09:00:00",

"monthyear" : "2023-04-11T09:00:00",

"name" : "Charles Emerson Whinchester III",

"outcome\_subtype" : "TEST",

"outcome\_type" : "Adopted",

"sex\_upon\_outcome" : "Neutered Male",

"location\_lat" : 30.5066578739455,

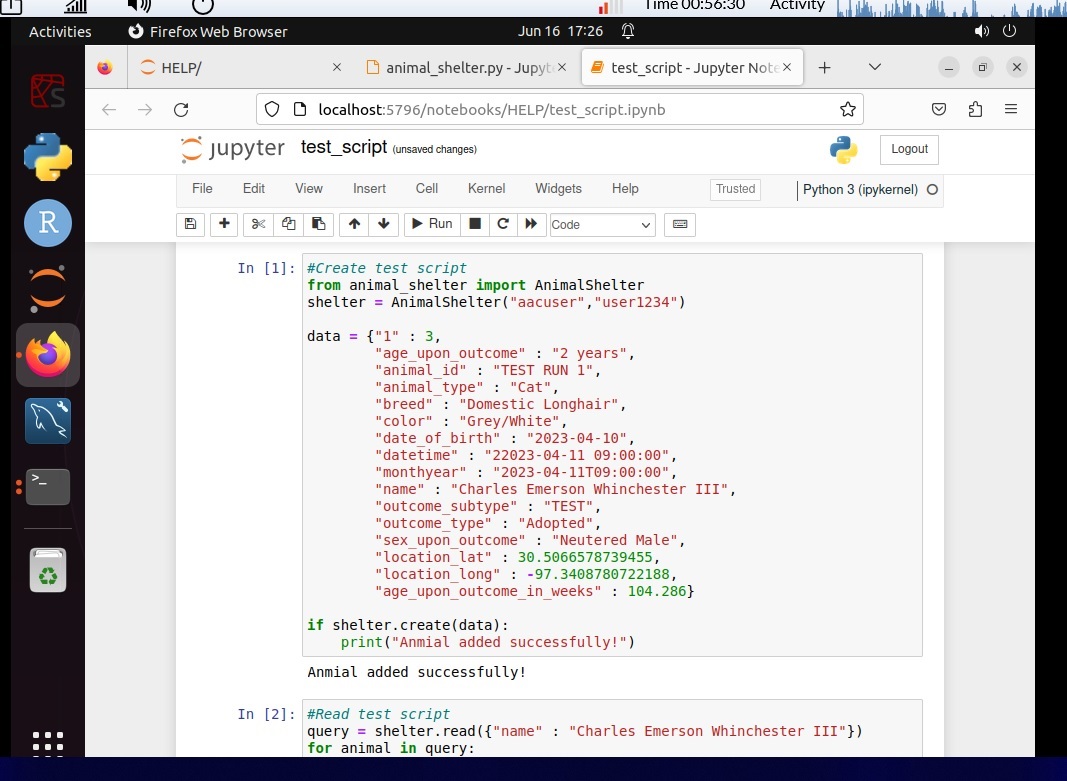
"location\_long" : -97.3408780722188,

"age\_upon\_outcome\_in\_weeks" : 104.286}

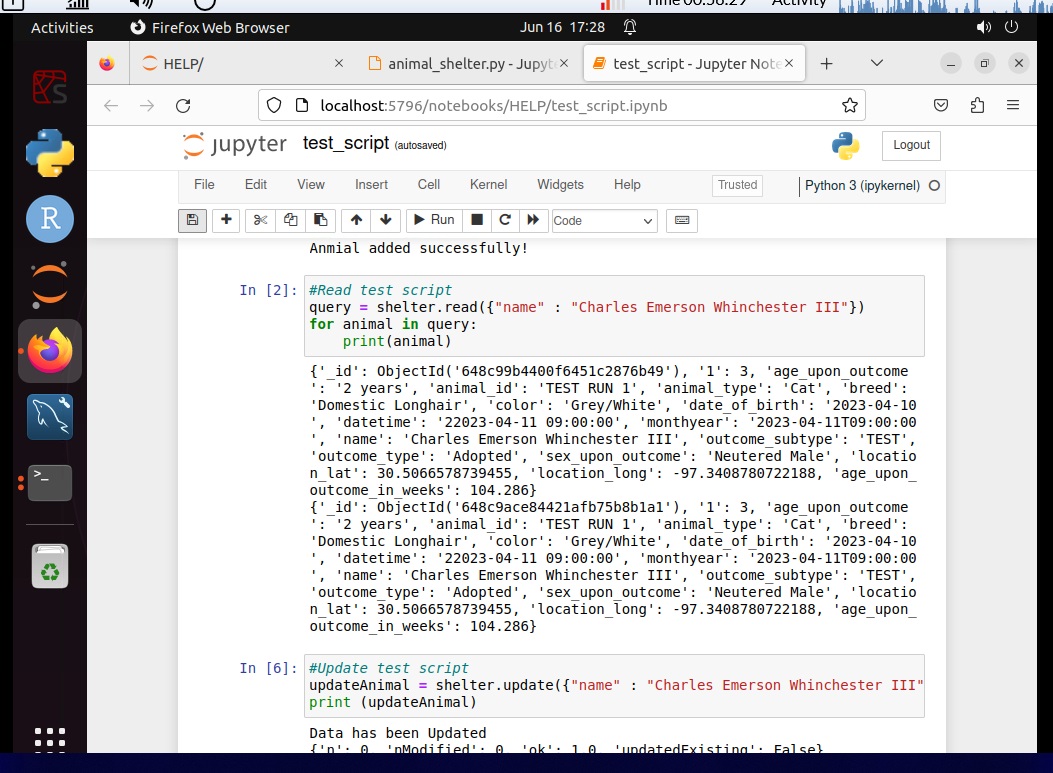
if shelter.create(data):

print("Anmial added successfully!")

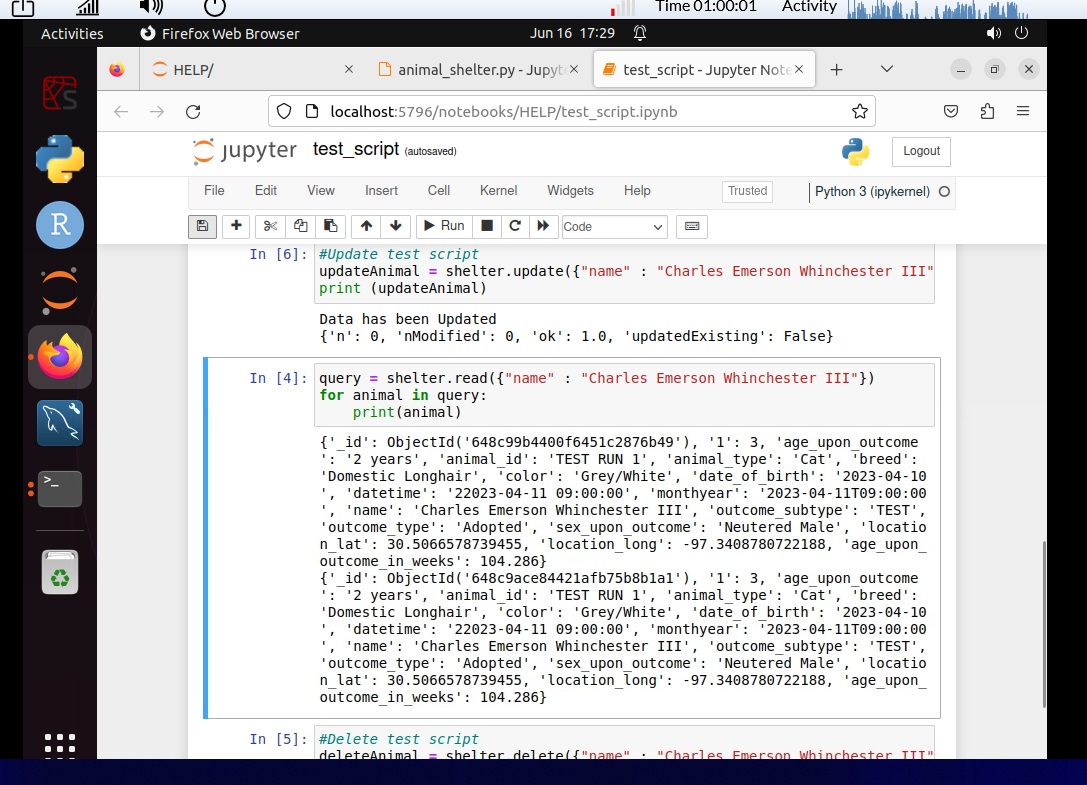
Testing of the **CREATE** function and outcome on the database.



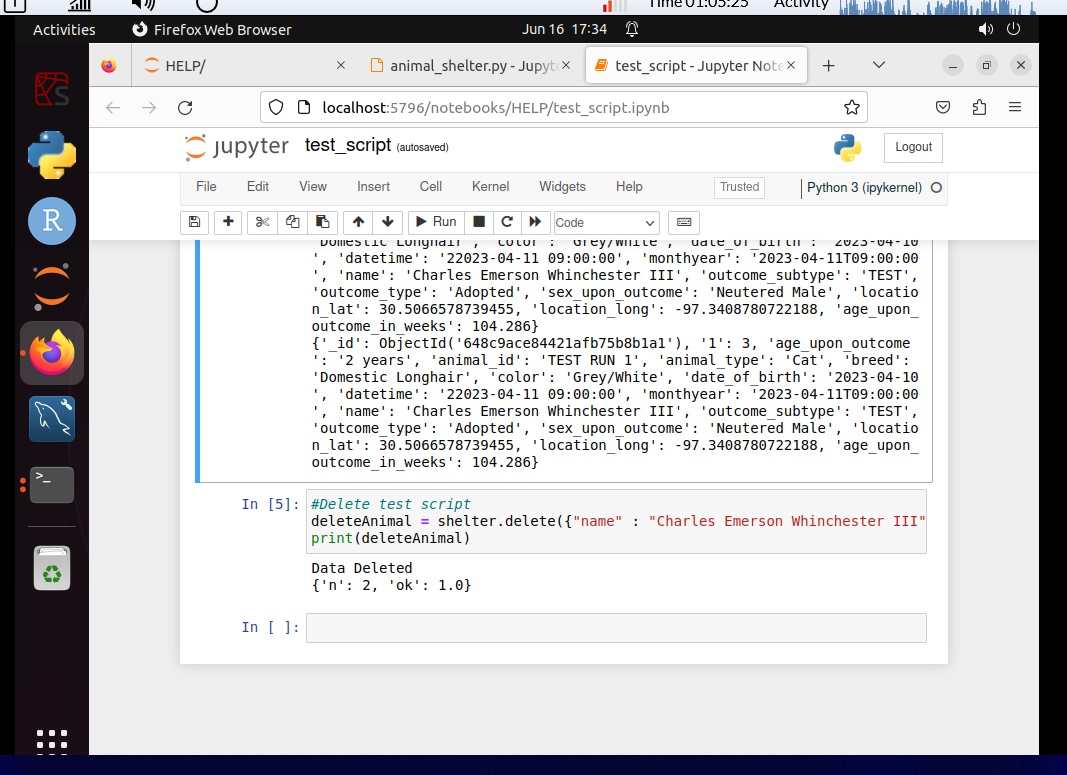
Next is the **READ** functionality.



Next is the **UPDATE** functionality and outcome.

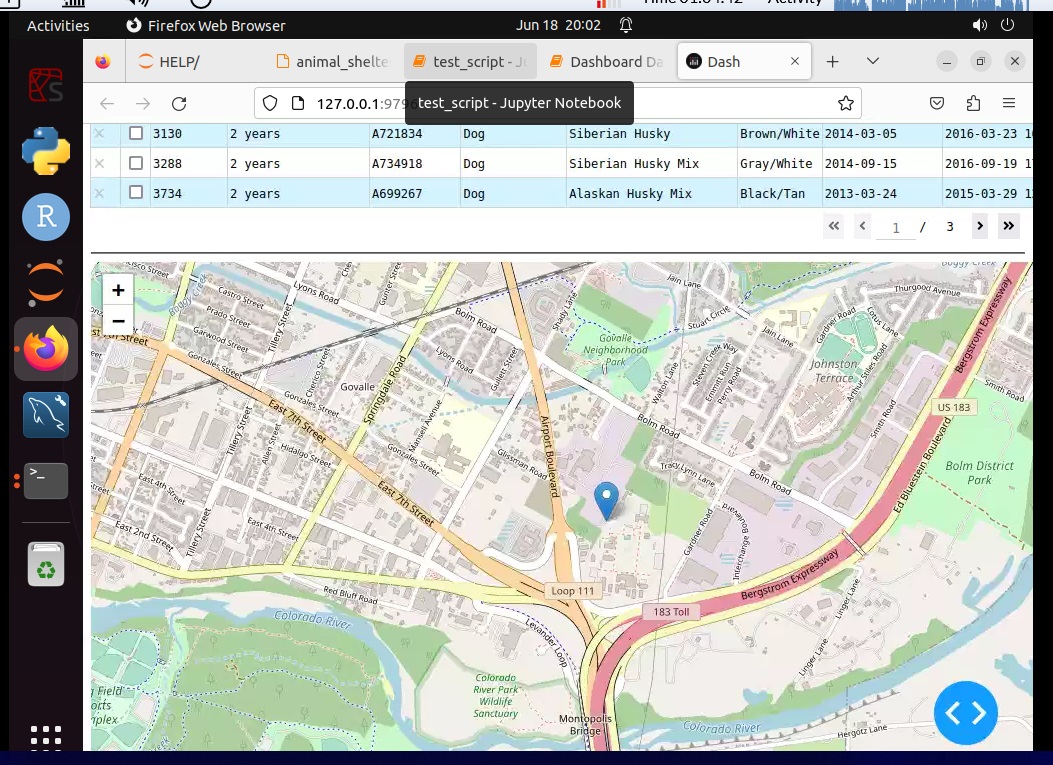
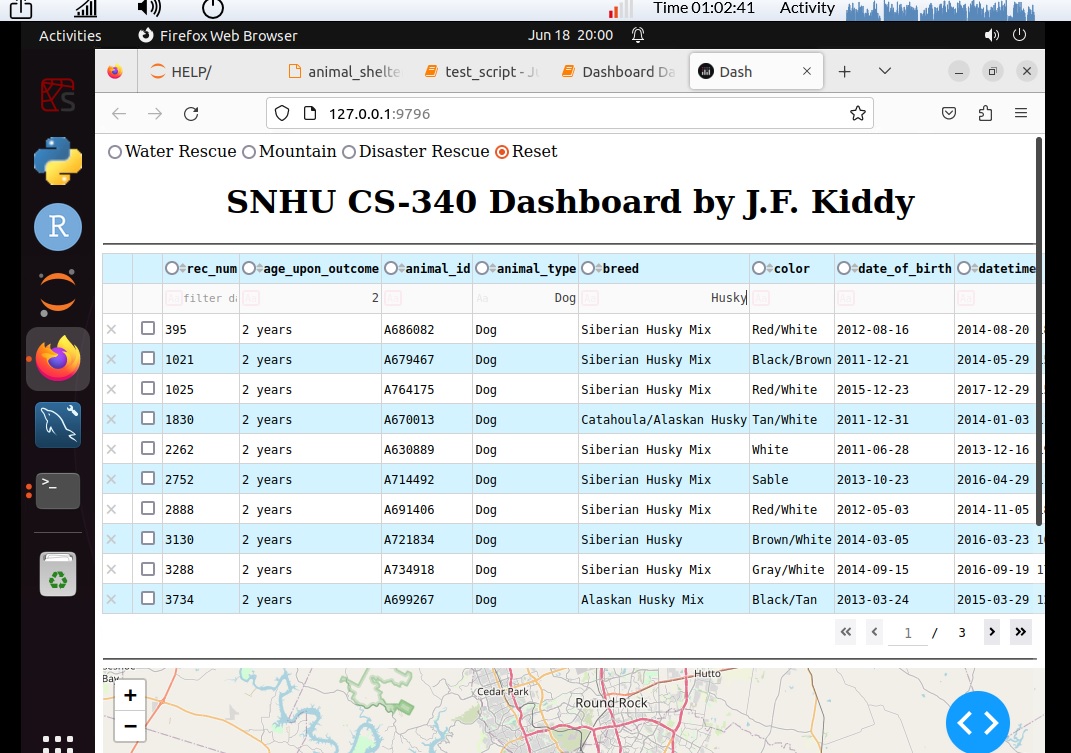


And lastly the **DELETE** function.



**Operation**

Here we see snap shots of the web-based application functionality with the user able to sort from 10,000 (over 150 pages) of files to 3 pages simply by selecting dog, 2 years old, Husky. Of course, this can be refined even further via other parameters, such as gender, and whether the canine has been spayed or neutered as examples.



**Contact**

J.F. Kiddy