Southern New Hampshire University

CS 340 Dashboard Sample Walkthrough

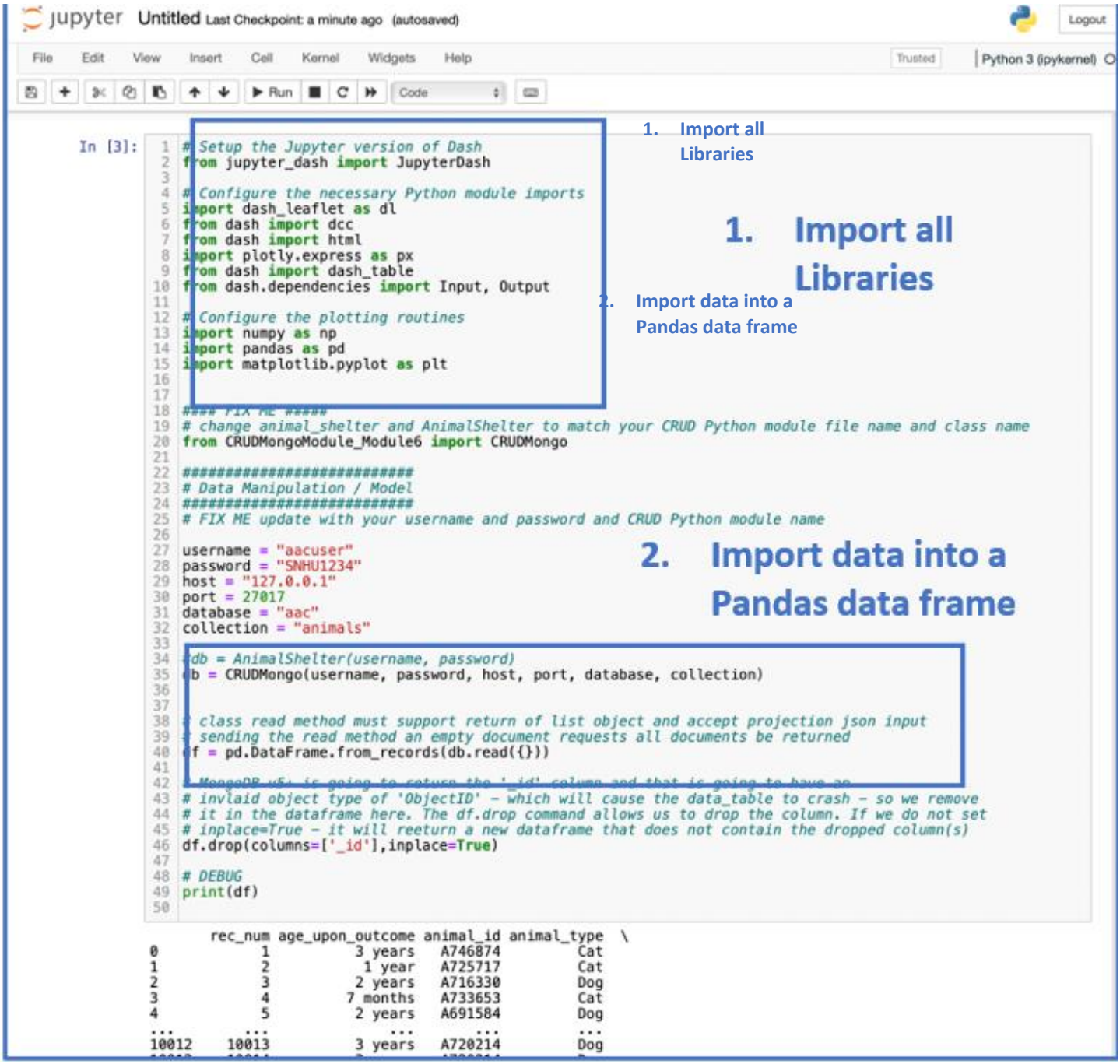
You can use the Dash framework in many different ways. This walkthrough will take you step by step through the process of creating a dashboard with data in MongoDB and a CRUD Python module. This dashboard will be similar to the one you will create in your assignments and projects.

Important note: The line numbers in the screenshots may vary slightly from the sample code.

Importing Data From MongoDB Using CRUD Python Module
As always, before writing your code, make sure to import all the libraries you'll need for your project. These import statements are included in the starter notebooks for your assignments, but normally, you would find them in the documentation for any framework or libraries you choose to use.

We first use the CRUD Python module to import data from MongoDB into a Pandas data frame. Pandas has good "glue" functionality to translate data between MongoDB and other libraries. We then run the notebook to make sure that the data imported correctly. Notice that the output underneath the cell shows the records from the data set. The formatting is not easy to read, and it is plain text. However, this test ensures that our CRUD Python module is working properly. It is important to build out large projects in small pieces, testing as we go to ensure that each component is working before incorporating the next element.

Note: In this walkthrough, the CRUD Python module is titled **CRUDMongoModule_Module6** and imported as **CRUDMongo.** When writing your own code, you should use the title of *your* CRUD Python module. The code for this example is available here: Tutorial_SampleCode.ipynb



Simple Dashboard Layout

Now that we know our data is importing correctly, we can begin using the Dash framework to create the web application, or client-side code. We start by building the layout of the dashboard in the **app.layout** section of the sample code. The layout allows us to organize the different components that make up our dashboard, such as tables, graphs, headers, text, and so on. You will learn more about how to build Dash layouts and the DataTable component in the module readings.

The sample code pictured below shows code for a basic layout that includes a title and a data table. The Dash framework has a built-in data table component, titled “DataTable,” which will format the data in a much more user-friendly way. Note that the DataTable component uses our Pandas data frame to populate the table:

data=df.to_dict("records"). For now, the table is populating the whole data set because that is what we imported in the previous step.

```
#########################
# Dashboard Layout / View
#########################
app = JupyterDash(__name__)

app.layout = html.Div([
    html.Div(id='hidden-div', style={'display':'none'}),
    html.Center(html.B(html.H1('SNHU CS-340 Dashboard'))),
    html.Hr(),
    dash_table.DataTable(id='datatable-id',
                         columns=[{"name": i, "id": i, "deletable": False, "selectable": True}
                                  for i in df.columns],
                         data=df.to_dict('records'),
                         editable=False,
                         filter_action="native",
                         sort_action="native",
                         sort_mode="multi",
                         column_selectable=False,
                         row_selectable=False,
                         row_deletable=False,
                         selected_columns=[],
                         selected_rows=[],
                         page_action="native",
                         page_current=0,
                         page_size=10
                        ),
    html.Br(),
    html.Hr()
])
```

Title

Imports Data

Data Table

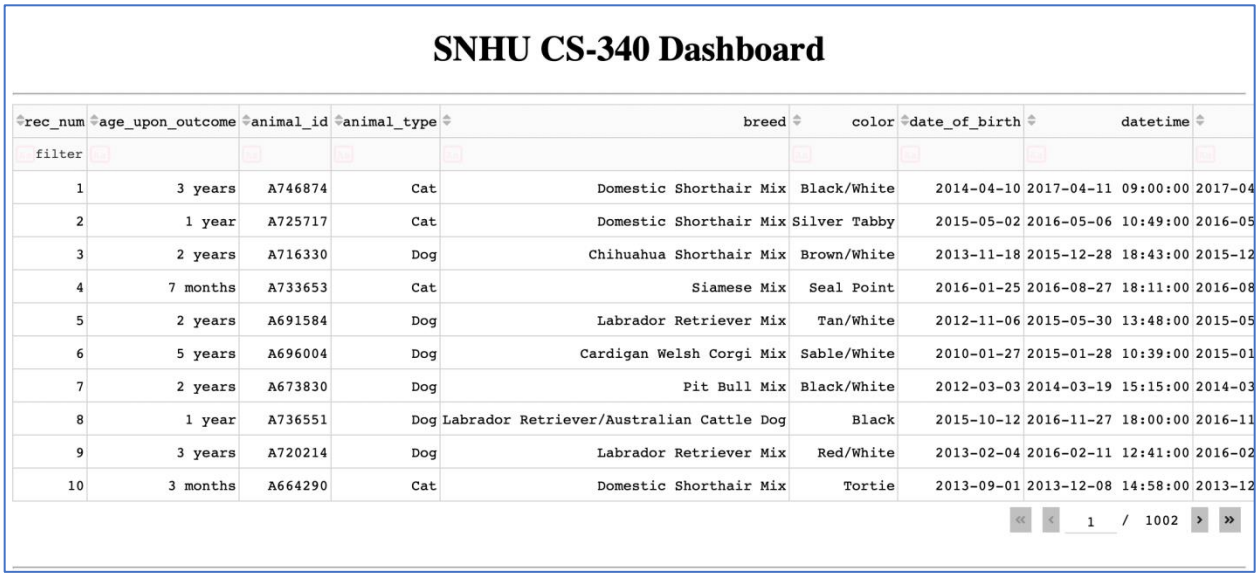
Data Table

Note: This code is a very simple example based on samples from the Dash DataTable documentation. Notice that different sorting and filtering options have been enabled. Learn more about these features in the Dash documentation, and experiment with different ways of setting up your data table or layout.

Now, we will test our code for the data table by running the application. In the next image, we have added the line "app.run_server(debug=True)" to the end of the code and have run the cell in Jupyter. Notice how the header and title for our dashboard are at the top with the data table underneath, just like in the app.layout code. Already, this data table is neater and much easier to read!

```
    html.Br(),
    html.Hr()
])

app.run_server(debug=True)
```



SNHU CS-340 Dashboard

rec_num	age_upon_outcome	animal_id	animal_type	breed	color	date_of_birth	datetime
filter							
1	3 years	A746874	Cat	Domestic Shorthair Mix	Black/White	2014-04-10	2017-04-11 09:00:00 2017-04
2	1 year	A725717	Cat	Domestic Shorthair Mix	Silver Tabby	2015-05-02	2016-05-06 10:49:00 2016-05
3	2 years	A716330	Dog	Chihuahua Shorthair Mix	Brown/White	2013-11-18	2015-12-28 18:43:00 2015-12
4	7 months	A733653	Cat	Siamese Mix	Seal Point	2016-01-25	2016-08-27 18:11:00 2016-08
5	2 years	A691584	Dog	Labrador Retriever Mix	Tan/White	2012-11-06	2015-05-30 13:48:00 2015-05
6	5 years	A696004	Dog	Cardigan Welsh Corgi Mix	Sable/White	2010-01-27	2015-01-28 10:39:00 2015-01
7	2 years	A673830	Dog	Pit Bull Mix	Black/White	2012-03-03	2014-03-19 15:15:00 2014-03
8	1 year	A736551	Dog	Labrador Retriever/Australian Cattle Dog	Black	2015-10-12	2016-11-27 18:00:00 2016-11
9	3 years	A720214	Dog	Labrador Retriever Mix	Red/White	2013-02-04	2016-02-11 12:41:00 2016-02
10	3 months	A664290	Cat	Domestic Shorthair Mix	Tortie	2013-09-01	2013-12-08 14:58:00 2013-12

« ‹ 1 / 1002 › »

Adding Interactive Filter Options: Layout

Our next task will be to add some filtering options so the client can easily filter the data set to find cats or dogs. First, let's add the code for these filtering options to the layout. The Dash Core Components documentation describes several built-in elements we could use: buttons, radio items, a drop-down menu, and so on. In this example, we have chosen to use the **button** element.

In the app.layout code pictured below, you can see new lines added for the button, based on the documentation. In this example, lines 63 and 64 show the code for the two buttons we are adding, an initialized value of 0 for the number of clicks, and labels for the buttons. The new lines of code place the buttons in one row so they are side by side instead of stacked vertically. The code lines also add optional styling to the buttons.

Notice that we need to specify an "id" for each button. There is an "id" for the DataTable element as well. Pay special attention to these "id" elements because they will be important for the callbacks in the next step.

```
51 ########################
52 # Dashboard Layout / View
53 ########################
54 app = JupyterDash(__name__)
55
56 app.layout = html.Div([
57     html.Div(id='hidden-div', style={'display':'none'}),
58     html.Center(html.B(html.H1('SNHU CS-340 Dashboard'))),
59     html.Hr(),
60     html.Div(className='buttonRow',
61             style={'display' : 'flex'},
62                 children=[
63                     html.Button(id='submit-button-one'  n_clicks=0, children='Cats'),
64                     html.Button(id='submit-button-two'  n_clicks=0, children='Dogs')
65                 ]),
66     dash_table.DataTable(id='datatable-id',
67                          columns=[{"name": i, "id": i, "deletable": False, "selectable": True}
68                                   for i in df.columns],
```

Button Shapes

Component IDs

Running this code will produce the result shown below. Notice that the buttons are included in the dashboard layout just as we have structured them in the code: under the table's title but before the table. The buttons are also in a single, horizontal row instead of stacked one on top of the other. We could move these buttons to a different part of the dashboard by moving those lines of code in the app.layout section.

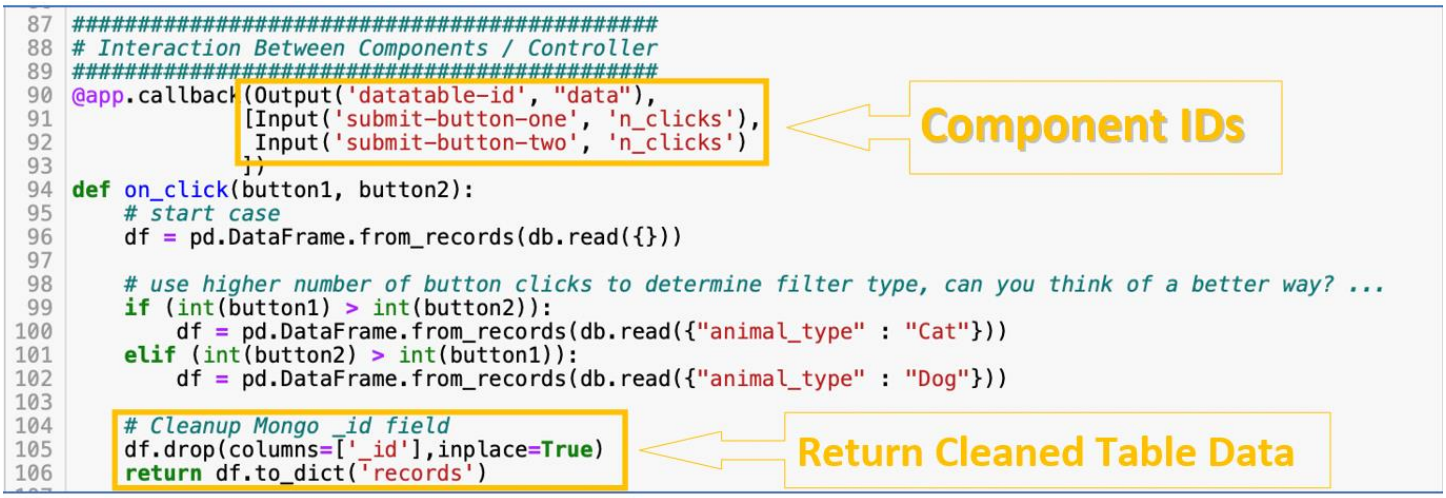
At this point, if we were to click on these buttons, they would **not** have any functionality. To make the buttons interact with the table, we need to add in the appropriate callbacks.

rec_num	age_upon_outcome	animal_id	animal_type	breed	color	date_of_birth	datetime	
filter								
1	3 years	A746874	Cat	Domestic Shorthair Mix	Black/White	2014-04-10	2017-04-11 09:00:00	2017-04
2	1 year	A725717	Cat	Domestic Shorthair Mix	Silver Tabby	2015-05-02	2016-05-06 10:49:00	2016-05
3	2 years	A716330	Dog	Chihuahua Shorthair Mix	Brown/White	2013-11-18	2015-12-28 18:43:00	2015-12
4	7 months	A733653	Cat	Siamese Mix	Seal Point	2016-01-25	2016-08-27 18:11:00	2016-08
5	2 years	A691584	Dog	Labrador Retriever Mix	Tan/White	2012-11-06	2015-05-30 13:48:00	2015-05
6	5 years	A696004	Dog	Cardigan Welsh Corgi Mix	Sable/White	2010-01-27	2015-01-28 10:39:00	2015-01
7	2 years	A673830	Dog	Pit Bull Mix	Black/White	2012-03-03	2014-03-19 15:15:00	2014-03
8	1 year	A736551	Dog	Labrador Retriever/Australian Cattle Dog	Black	2015-10-12	2016-11-27 18:00:00	2016-11
9	3 years	A720214	Dog	Labrador Retriever Mix	Red/White	2013-02-04	2016-02-11 12:41:00	2016-02
10	3 months	A664290	Cat	Domestic Shorthair Mix	Tortie	2013-09-01	2013-12-08 14:58:00	2013-12

« ‹ 1 / 1002 › »

Adding Interactive Filter Options: Callbacks

The documentation for the Dash Core Components includes samples of code for the callbacks. In the callback structure, an "output" id matches the id of the data table element from the layout. Then two "input" IDs match the IDs of the two button elements from the layout. Matching indicates that this callback will take input from the button elements and output it to the data table. The "return" for the "on_click" function in the code pictured below is "df.to_dict('records')", which corresponds to the "data=df.to_dict('records')" line for the DataTable component. This setting ensures that the results of the callback will return the correct data type for the data table.



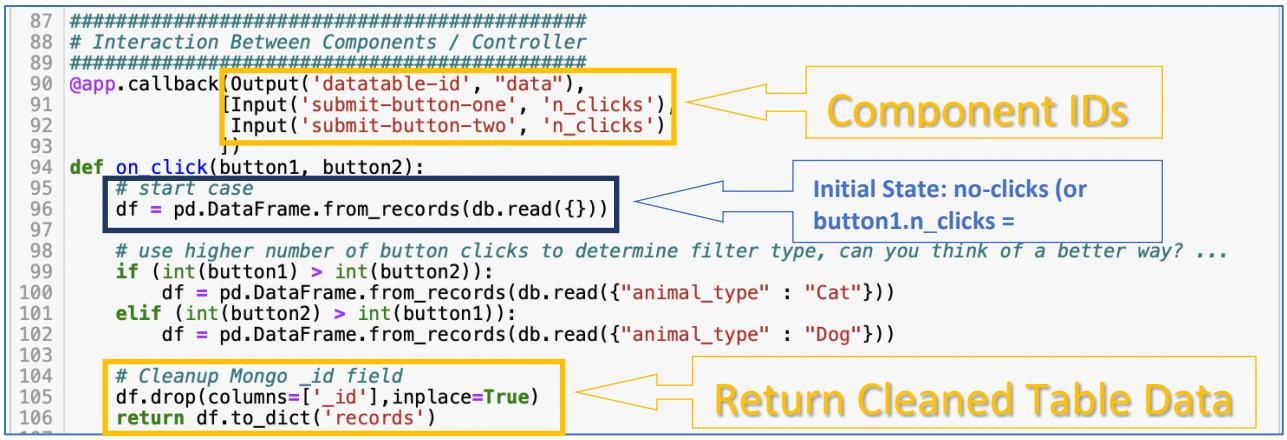

```
#############################################
# Interaction Between Components / Controller
#############################################
@app.callback(Output('datatable-id', "data"),
              [Input('submit-button-one', 'n_clicks'),
               Input('submit-button-two', 'n_clicks')
              ])
def on_click(button1, button2):
    # start case
    df = pd.DataFrame.from_records(db.read({}))

    # use higher number of button clicks to determine filter type, can you think of a better way? ...
    if (int(button1) > int(button2)):
        df = pd.DataFrame.from_records(db.read({"animal_type" : "Cat"}))
    elif (int(button2) > int(button1)):
        df = pd.DataFrame.from_records(db.read({"animal_type" : "Dog"}))

    # Cleanup Mongo _id field
    df.drop(columns=['_id'],inplace=True)
    return df.to_dict('records')
```

You will have to determine the appropriate logic based on the element that you choose. This button component uses clicks. This logic is structured with an initial state for the callback: *What happens if nothing is pressed*?
In this case, the table will just display all records, based on the command:
"df = pd.DataFrame.from_records(shelter.read({}))".

Note: "read" was the name of the read method for the "CRUDMongoModule_Module6" CRUD Python module. You will need to use the name of the method from your CRUD Python module.




```
#############################################
# Interaction Between Components / Controller
#############################################
@app.callback(Output('datatable-id', "data"),
              [Input('submit-button-one', 'n_clicks'),
               Input('submit-button-two', 'n_clicks')
              ])
def on_click(button1, button2):
    # start case
    df = pd.DataFrame.from_records(db.read({}))

    # use higher number of button clicks to determine filter type, can you think of a better way? ...
    if (int(button1) > int(button2)):
        df = pd.DataFrame.from_records(db.read({"animal_type" : "Cat"}))
    elif (int(button2) > int(button1)):
        df = pd.DataFrame.from_records(db.read({"animal_type" : "Dog"}))

    # Cleanup Mongo _id field
    df.drop(columns=['_id'],inplace=True)
    return df.to_dict('records')
```

Different filtering queries are written for each of the two buttons, **Cat** and **Dog**.

1. df = pd.DataFrame(db.read({"animal_type":"Cat"}))
2. df = pd.DataFrame(db.read({"animal_type":"Dog"}))

```
87 ##########################################
88 # Interaction Between Components / Controller
89 ##########################################
90 @app.callback(Output('datatable-id', "data"),
91               [Input('submit-button-one', 'n_clicks'),
92                Input('submit-button-two', 'n_clicks')
93               ])
94 def on_click(button1, button2):
95     # start case
96     df = pd.DataFrame.from_records(db.read({}))
97
98     # use higher number of button clicks to determine filter type, can you think of a better way? ...
99     if (int(button1) > int(button2)):
100        df = pd.DataFrame.from_records(db.read({"animal_type" : "Cat"}))
101    elif (int(button2) > int(button1)):
102        df = pd.DataFrame.from_records(db.read({"animal_type" : "Dog"}))
103
104    # Cleanup Mongo _id field
105    df.drop(columns=['_id'],inplace=True)
106    return df.to_dict('records')
```

Component IDs

State Changes

Notice that the logic is set up so that these actions “trigger” when one button is clicked more than another. This approach is not the only approach, and it may include some risks. If the **Cat** button is clicked once, how many times must the **Dog** button be clicked to filter the data table to show dogs? Can you think of another way to set up the logic for these buttons? How would you set up buttons for more than two filtering options? What happens when the buttons are both clicked the same number of times?

When the app is run again, the buttons are now functional. Clicking the buttons should filter the data set. Based on the current button logic, from the initial state, clicking the **Cats** button once will filter the table to show only cats. To filter the table to show dogs, the **Dogs** button would need to be clicked twice. See the output examples below.

SNHU CS-340 Dashboard

Click “Cats” Button Once

Cats Dogs

rec_num	age_upon_outcome	animal_id	animal_type	breed	color	date_of_birth	datetime	monthyea
filter								
1	3 years	A746874	Cat	Domestic Shorthair Mix	Black/White	2014-04-10	2017-04-11 09:00:00	2017-04-11T09:00:0
2	1 year	A725717	Cat	Domestic Shorthair Mix	Silver Tabby	2015-05-02	2016-05-06 10:49:00	2016-05-06T10:49:0
4	7 months	A733653	Cat	Siamese Mix	Seal Point	2016-01-25	2016-08-27 18:11:00	2016-08-27T18:11:0
10	3 months	A664290	Cat	Domestic Shorthair Mix	Tortie	2013-09-01	2013-12-08 14:58:00	2013-12-08T14:58:0
13	1 year	A700408	Cat	Domestic Shorthair Mix	Brown Tabby/White	2014-04-13	2015-04-15 13:34:00	2015-04-15T13:34:0
18	2 months	A693288	Cat	Domestic Shorthair Mix	Brown Tabby/White	2014-09-28	2014-12-09 18:36:00	2014-12-09T18:36:0
19	4 months	A709511	Cat	Domestic Medium Hair Mix	White/Black	2015-07-10	2015-11-13 14:00:00	2015-11-13T14:00:0
24	3 weeks	A704707	Cat	Siamese Mix	Seal Point	2015-05-17	2015-06-08 16:25:00	2015-06-08T16:25:0
25	2 months	A725966	Cat	Domestic Shorthair Mix	Tortie	2016-03-20	2016-05-24 11:01:00	2016-05-24T11:01:0
26	1 year	A743231	Cat	Domestic Shorthair Mix	Black	2016-02-06	2017-02-07 09:00:00	2017-02-07T09:00:0

« ‹ 1 / 379 › »

Click “Dogs” Button Twice

SNHU CS-340 Dashboard

Cats Dogs

rec_num	age_upon_outcome	animal_id	animal_type	breed	color	date_of_birth	datetime
filter							
3	2 years	A716330	Dog	Chihuahua Shorthair Mix	Brown/White	2013-11-18	2015-12-28 18:43:00
5	2 years	A691584	Dog	Labrador Retriever Mix	Tan/White	2012-11-06	2015-05-30 13:48:00
6	5 years	A696004	Dog	Cardigan Welsh Corgi Mix	Sable/White	2010-01-27	2015-01-28 10:39:00
7	2 years	A673830	Dog	Pit Bull Mix	Black/White	2012-03-03	2014-03-19 15:15:00
8	1 year	A736551	Dog	Labrador Retriever/Australian Cattle Dog	Black	2015-10-12	2016-11-27 18:00:00
9	3 years	A720214	Dog	Labrador Retriever Mix	Red/White	2013-02-04	2016-02-11 12:41:00
11	1 year	A721199	Dog	Dachshund Wirehair Mix	Tan/White	2015-02-23	2016-02-27 17:49:00
12	1 year	A664843	Dog	Pit Bull Mix	Brown/White	2013-06-09	2014-08-18 17:24:00
14	2 years	A742287	Dog	Boxer/Bullmastiff	Brown Brindle/White	2015-01-18	2017-02-11 12:30:00
15	3 years	A712638	Dog	Pit Bull Mix	Red/White	2012-09-26	2016-07-18 17:52:00

« ‹ 1 / 561 › »