Neelima Patnaik

Projec-2 Readme

Sothern New Hampshire university

2/21/2022

**About Project:**

An online application sifting through data stored in a database powered by MongoDB is designed to return combinations of dog breed, age, and sex desirable to training rescue dogs. The project consists of a database, an API, and a dynamic dashboard.

The purpose of project two is to create a dashboard that allows a user (specifically Grazio Salvare) to identify and navigate the pets available for adoption in the Austin, TX area.

The dashboard has three main features:

**1. A table containing all the pet's information**

**2. A pie chart identifying the breeds of the pets in the table**

**3. A map of the pet's locationMotivation for Using Mongo DB**

It was specifically chosen for this program because it is easy to set up a database from a CSV document and has a Python-friendly interface. Python supports database tools such as SQL, but the syntax is so different that switching between them is difficult. SQL databases employ significantly more complex selection tools compared to MongoDB's, which use Python to run the essential CRUD functions of a database

**Dash Motivation**

Dash's dynamic nature made it a desirable tool for building dashboards. Dash is a JavaScript-based tool that offers incredibly responsive functionality. Dash tags are HTML tags that control output to segments. After that, updates will be made to any of the target inputs specified in the sapp callbacks process according to instructions programmed in the Python module.

Motivation

The goal of this project is to streamline the effort of querying databases and providing the customer with relevant information.

**Getting Started**

**Section A: setting up mongoDB**

The first step of this process is to import the AAC data needed to MongoDB. To do so follow the guide provided:

To setup your mongoDB session complete the following:

1. Open a terminal window
2. #start mongo with authentication

/usr/local/bin/mongod\_ctl start

1. #To load the data sets into this new MongoDB server environment, execute the following commands:

cd /usr/local/datasets

/usr/local/bin/mongod\_ctl start-noauth

mongoimport --port ##### --db city --collection inspections ./city\_inspections.json

1. mongoimport --port ##### --db AAC --collection animals --type=csv --headerline ./aac\_shelter\_outcomes.csv
2. Enable authentication
3. # now stop mongo

/usr/local/bin/mongod\_ctl stop

# and start mongo with authentication, again make note of the

/usr/local/bin/mongod\_ctl start

Section -B:

1.Set up a MongoDB database and name it AAC and Admin.

2.Set up a user with read/write privileges to that AAC database.

3.Data from aac\_shelter\_outcomes.csv should be imported

Text

Description automatically generated

Admin and accuser account

Text

Description automatically generated

Text

Description automatically generated

Unfortunately, not able to see the dashboard;

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Code used for Map Example**:

elif len(selected\_rows) == 4:

return [

dl.Map(style={'width':'1000px', 'height': '500px'}, center=[30.75,-97.48], zoom=10, children=[

dl.TileLayer(id="base-layer-id"),

#marker with tool tip and popup

dl.Marker(position=[(dff.iloc[selected\_rows[0],13]), (dff.iloc[selected\_rows[0],14])], children=[

dl.Tooltip(dff.iloc[selected\_rows[0],4]),

dl.Popup([

html.H4("Animal Name"),

html.P(dff.iloc[selected\_rows[0],9]),

html.H4("Sex"),

html.P(dff.iloc[selected\_rows[0],12]),

html.H4("Breed"),

html.P(dff.iloc[selected\_rows[0],4]),

html.H4("Age"),

html.P(dff.iloc[selected\_rows[0],15])

])

]),

dl.Marker(position=[(dff.iloc[selected\_rows[1],13]), (dff.iloc[selected\_rows[1],14])], children=[

dl.Tooltip(dff.iloc[selected\_rows[1],4]),

dl.Popup([

html.H4("Animal Name"),

html.P(dff.iloc[selected\_rows[1],9]),

html.H4("Sex"),

html.P(dff.iloc[selected\_rows[1],12]),

html.H4("Breed"),

html.P(dff.iloc[selected\_rows[1],4]),

html.H4("Age"),

html.P(dff.iloc[selected\_rows[1],15])

])

]),

dl.Marker(position=[(dff.iloc[selected\_rows[2],13]), (dff.iloc[selected\_rows[2],14])], children=[

dl.Tooltip(dff.iloc[selected\_rows[2],4]),

dl.Popup([

html.H4("Animal Name"),

html.P(dff.iloc[selected\_rows[2],9]),

html.H4("Sex"),

html.P(dff.iloc[selected\_rows[2],12]),

html.H4("Breed"),

html.P(dff.iloc[selected\_rows[2],4]),

html.H4("Age"),

html.P(dff.iloc[selected\_rows[2],15])

])

]),

dl.Marker(position=[(dff.iloc[selected\_rows[3],13]), (dff.iloc[selected\_rows[3],14])], children=[

dl.Tooltip(dff.iloc[selected\_rows[3],4]),

dl.Popup([

html.H4("Animal Name"),

html.P(dff.iloc[selected\_rows[3],9]),

html.H4("Sex"),

html.P(dff.iloc[selected\_rows[3],12]),

html.H4("Breed"),

html.P(dff.iloc[selected\_rows[3],4]),

html.H4("Age"),

html.P(dff.iloc[selected\_rows[3],15])

])

])

])

]

#creats a map with 5 markers

elif len(selected\_rows) == 5:

return [

dl.Map(style={'width':'1000px', 'height': '500px'}, center=[30.75,-97.48], zoom=10, children=[

dl.TileLayer(id="base-layer-id"),

#marker with tool tip and popup

dl.Marker(position=[(dff.iloc[selected\_rows[0],13]), (dff.iloc[selected\_rows[0],14])], children=[

dl.Tooltip(dff.iloc[selected\_rows[0],4]),

dl.Popup([

html.H4("Animal Name"),

html.P(dff.iloc[selected\_rows[0],9]),

html.H4("Sex"),

html.P(dff.iloc[selected\_rows[0],12]),

html.H4("Breed"),

html.P(dff.iloc[selected\_rows[0],4]),

html.H4("Age"),

html.P(dff.iloc[selected\_rows[0],15])

])

]),

dl.Marker(position=[(dff.iloc[selected\_rows[1],13]), (dff.iloc[selected\_rows[1],14])], children=[

dl.Tooltip(dff.iloc[selected\_rows[1],4]),

dl.Popup([

html.H4("Animal Name"),

html.P(dff.iloc[selected\_rows[1],9]),

html.H4("Sex"),

html.P(dff.iloc[selected\_rows[1],12]),

html.H4("Breed"),

html.P(dff.iloc[selected\_rows[1],4]),

html.H4("Age"),

html.P(dff.iloc[selected\_rows[1],15])

])

]),

dl.Marker(position=[(dff.iloc[selected\_rows[2],13]), (dff.iloc[selected\_rows[2],14])], children=[

dl.Tooltip(dff.iloc[selected\_rows[2],4]),

dl.Popup([

html.H4("Animal Name"),

html.P(dff.iloc[selected\_rows[2],9]),

html.H4("Sex"),

html.P(dff.iloc[selected\_rows[2],12]),

html.H4("Breed"),

html.P(dff.iloc[selected\_rows[2],4]),

html.H4("Age"),

html.P(dff.iloc[selected\_rows[2],15])

])

]),

dl.Marker(position=[(dff.iloc[selected\_rows[3],13]), (dff.iloc[selected\_rows[3],14])], children=[

dl.Tooltip(dff.iloc[selected\_rows[3],4]),

dl.Popup([

html.H4("Animal Name"),

html.P(dff.iloc[selected\_rows[3],9]),

html.H4("Sex"),

html.P(dff.iloc[selected\_rows[3],12]),

html.H4("Breed"),

html.P(dff.iloc[selected\_rows[3],4]),

html.H4("Age"),

html.P(dff.iloc[selected\_rows[3],15])

])

]),

dl.Marker(position=[(dff.iloc[selected\_rows[4],13]), (dff.iloc[selected\_rows[4],14])], children=[

dl.Tooltip(dff.iloc[selected\_rows[4],4]),

dl.Popup([

html.H4("Animal Name"),

html.P(dff.iloc[selected\_rows[4],9]),

html.H4("Sex"),

html.P(dff.iloc[selected\_rows[4],12]),

html.H4("Breed"),

html.P(dff.iloc[selected\_rows[4],4]),

html.H4("Age"),

html.P(dff.iloc[selected\_rows[4],15])

])

])

])

]

**#function to update the pie chart**

@app.callback(

Output('graph-id', "children"),

[Input('datatable-id', "derived\_viewport\_data")])

def update\_graphs(viewData):

###FIX ME ####

# imports the currently displayed data

dff = pd.DataFrame.from\_dict(viewData)

#creates the values needed for the names (breed) and values (recurring counts)

names = dff['breed'].value\_counts().keys().tolist()

values = dff['breed'].value\_counts().tolist()

#creates a pie chart based on the data above

return [

dcc.Graph(

figure = px.pie(

data\_frame=dff,

values = values,

names = names,

color\_discrete\_sequence=px.colors.sequential.RdBu,

width=800,

height=500

)

)

]

**Other Resources used:**

Installation

This will require Jupyter Notebooks, Python for command-line, and MongoDB. The installation of each is explained in labeled sections under this line.

Notebooks: Jupyter can be installed from the command line using the following instructions: https://jupyter.org/install. For detailed instructions like Proxy servers for Windows, Mac, and Linux, follow the instructions here: https://jupyterlab.readthedocs.io/en/stable/getting\_started/installation.html

Python: Detailed installation instructions for Python are available here: https://realpython.com/installing-python/. You can run this program from the Command Prompt on Windows or the Terminal on Mac or Linux once you have Python installed.

MongoDB: MongoDB comes in Community or Enterprise editions. Detailed instructions for the installation and downloading of MongoDB are available here: https://docs.mongodb.com/manual/installation/.

Plotly

Plotly must be imported in order to generate the proper charts. Plotly is a charting tool for Python applications and can be imported directly into your Python module from your Jupyter notebookThe tools you will need to run this include Jupyter Notebooks, Python for command line, and MongoDB. The installation of each is detailed in labeled sections right below this line.

Jupyter Notebooks: Jupyter can be installed from the command line in any major operating system using the simple instructions here:

<https://jupyter.org/install>

<https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html>

**Python**: Detailed installation instructions for Python are available here: <https://realpython.com/installing-python/>

Once you have Python installed, you should be able to use this program from the Terminal on Mac or Linux or from the Command Prompt for Windows.

**MongoDB**: MongoDB comes in Community or Enterprise editions. Detailed instructions for the installation and downloading of MongoDB are available here: <https://docs.mongodb.com/manual/installation/>

**Dash**

Dash is a framework used to build web applications. You can import the Dash Core Components into your Jupyter notebook, and you can install Dash using the following information: <https://pypi.org/project/dash/>

**Pandas**

Pandas is used in this web application as well. Pandas is a tool for Python that creates the data frames. Pandas has other dependencies and information that should be reviewed before use here: <https://pandas.pydata.org/pandas-docs/stable/getting_started/install.html>