We chose to use Postgres for our data storage because of its ease of use and our familiarity with PgAdmin.

One CATCH!

Team Access to THE database would require WS. Which requires subscription like AWS and has potential costs associated with it.

We Decided “not” to go with WS so there were no worries about incurring costs

This choice created a major hurdle: ***PAUSE*** How to allow for team access to the same database?

We overcame this hurdle by using the data loading lesson in module 8.5.1 and jupyter notebook to recreate the database on each team members workstation, allowing us to run and test the ML module with same results.

How: ***PAUSE***

* First, we made a connection to our local postgres database instance on our machine using sqlAlchemy.
* Then We imported targeted CSV files into dataframes using the read\_csv method in pandas.

This step gives the ability to quickly review the data before importing and drop any irrelevant data columns.

* After review we loaded the cleaned dataframe to our postgres database using the to\_sql method in pandas which creates a table from the dataframe.
* Continuing this step for every csv we wanted in the database

By using juptyer notebook it made it easy for each team member to repeat the steps of the process thus creating a replica of the database on their machine. More complete directions are included in our ReadMe file on the repository. With the same data files and our Jupyter notebook anyone could re-create our database and ML model on their workstation.

Once the database is created and data loaded the next step is to extract the appropriate data for our ML model from database.

* To do this we used the same SQLALchemy connection from the first step but this time we used the from\_sql method in pandas to extract the data from our postGres database for preprocessing.

On this slide you can see we successfully tested by extracting the Fielding table into a dataframe. We are displaying the header.