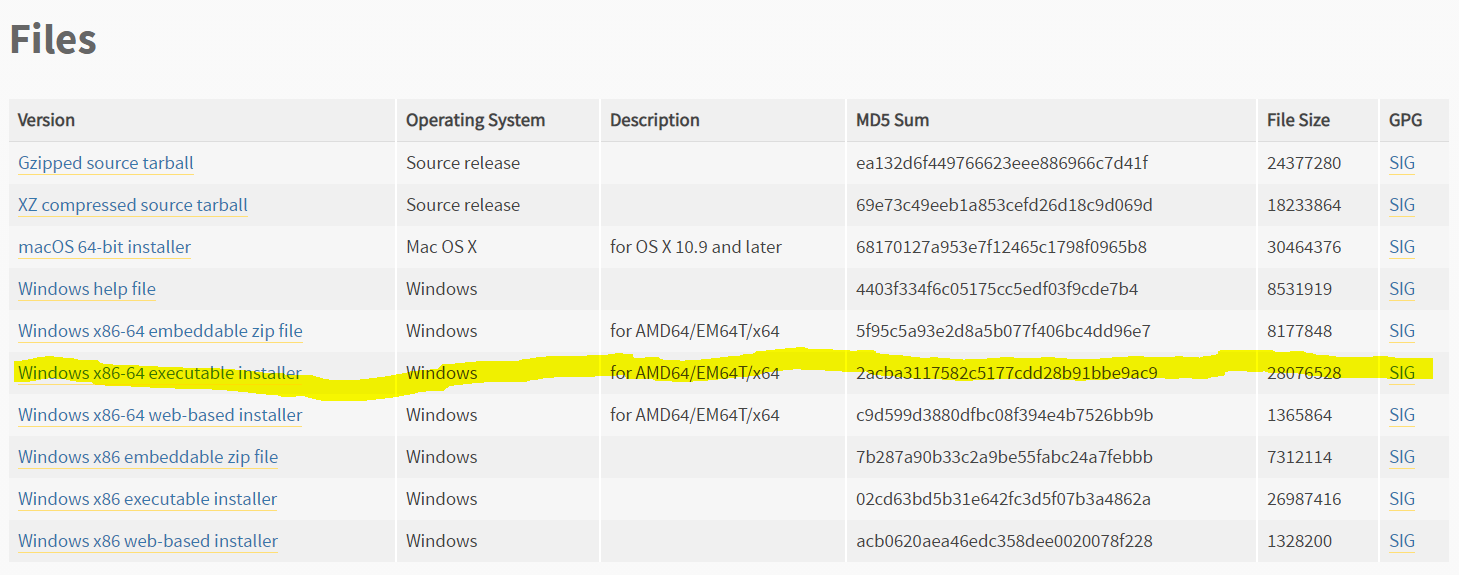
# **PBNJ2 guide to getting started on the Dash web**

1. **Install a 64-bit version of Python 3.** It must be less than version 3.9 to get it to work with Tensorflow. Note: I have version 3.8.6. Link here:

<https://www.python.org/downloads/release/python-386/>



Version 3.8.8 (the most recent 3.8 release) should work as well but I can’t say for certain. Someone should try and let everyone know if they run into problems. 😊

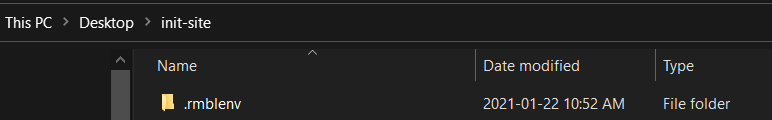
1. **Set-up a project folder and create a virtual environment.** This is so project-specific packages you need only exists in your project folder and you can simply the whole folder when you don’t need them anymore. It also helps you reduce clutter in your system in the long run.
   1. Create a new folder and name it whatever you want.
   2. Open up command prompt or powershell and browse into your project folder



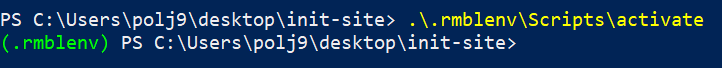
I named mine .rmblenv



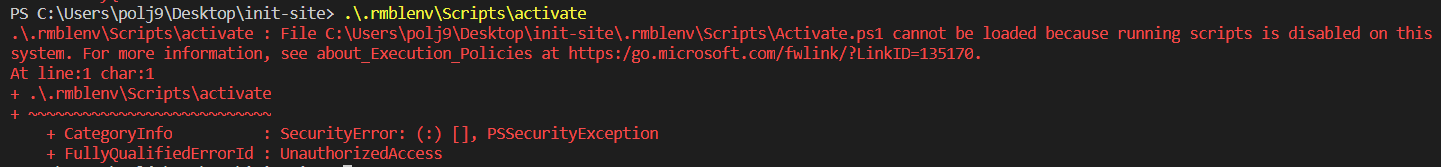
This creates a folder with the name you provided for your virtual environment



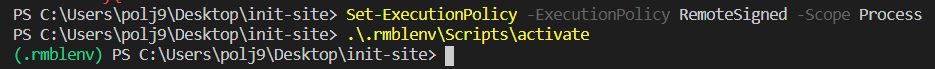
* 1. Activate the virtual environment



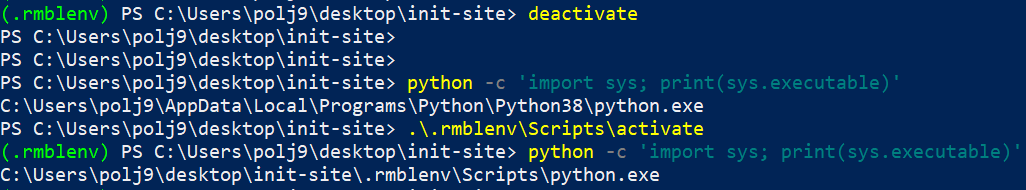
If you run into problems like the following



Try running this command first, and then re-try activation



* 1. Now, ensure that subsequent python commands you execute on command prompt or powershell, the python version from your virtual environment is the one that gets invoked. The following screenshot should be instructive:



* 1. Upgrade pip.



1. **Download the 2 zip files on Whatsapp and unzip it on your project folder.** Version0.zip contains ze python files i.e. ze codezz, and assets.zip contains the images that the web app uses so far.
2. **Download the BERT model from MS Teams.** Create a ‘model’ folder in your project folder and move the bert files there. Rename the files to ‘tf\_model.h5’ (the big file) and ‘tf\_model.preproc’ (the small file). By this point, you should have python files on your project folder, an ‘assets’ folder (in your project folder) containing images, and a ‘model’ folder (in your project folder) containing the bert files.
3. **Install some packages.** Pip install everything.

pip install numpy pandas tensorflow

pip install tweepy unidecode ktrain

pip install dash dash-renderer plotly

pip install dash-core-components dash-html-components dash\_bootstrap\_components

pip install sqlalchemy psycopg2

pip install flask flask\_sqlalchemy

Reminder: the packages listed above is probably incomplete. If you get an error later saying that you’re missing a module/package, please let me know so I can take note of it.

1. **Set-up Postgresql on your system.** The streamer program needs a database to store data to before we can run it.
   1. Download and install Postgres.

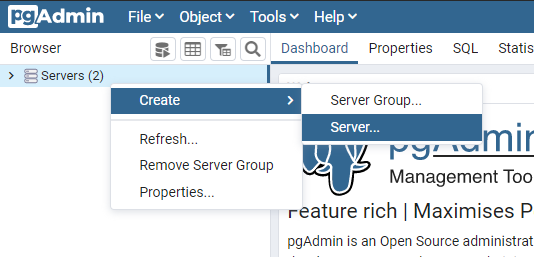
Link here:

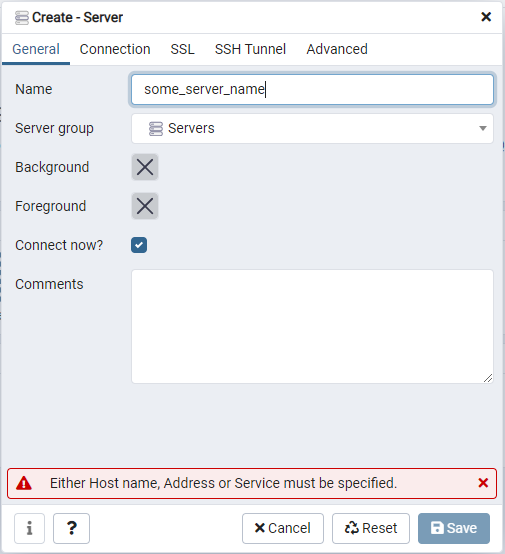
<https://www.enterprisedb.com/downloads/postgres-postgresql-downloads>

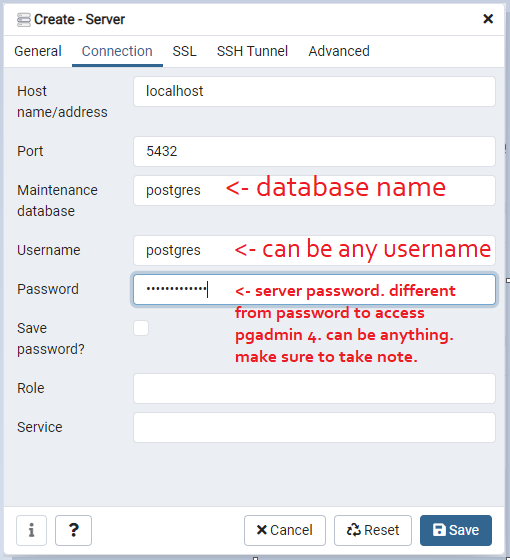
Installation guide here: (take note of the password you create here. It’s for accessing the pgadmin 4 dashboard)

<https://www.postgresqltutorial.com/install-postgresql/>

* 1. Open pgadmin 4. Windows button > pgadmin 4
  2. Create a database server.

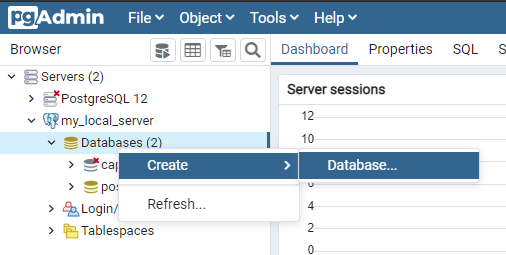




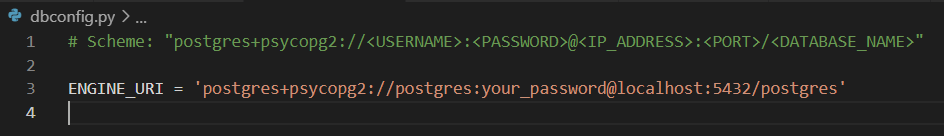


Click save.

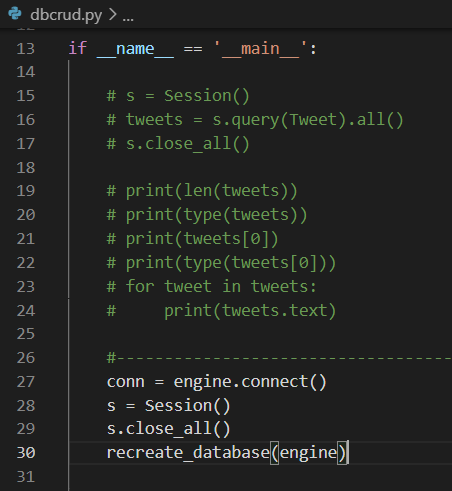
* 1. You can create another database if you wish. It’s probably better practice to do so, but since this is only a student project, let’s just use the “maintenance database” we created above.



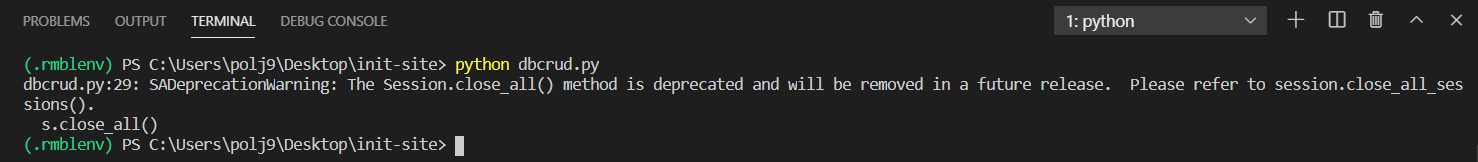
1. **Create tables in your local database.**
   1. Open dbconfig.py and modify the ENGINE\_URI variable to reflect the credentials you made for your local database server.



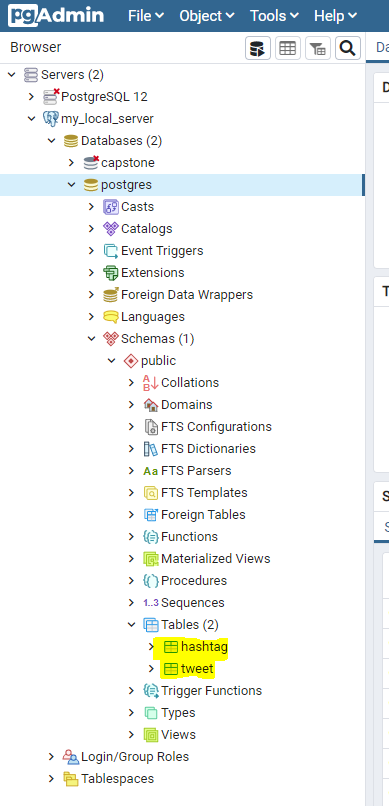
* 1. Open dbcrud.py and uncomment/comment some lines. Yep, I forgot to do this before sending out the files.



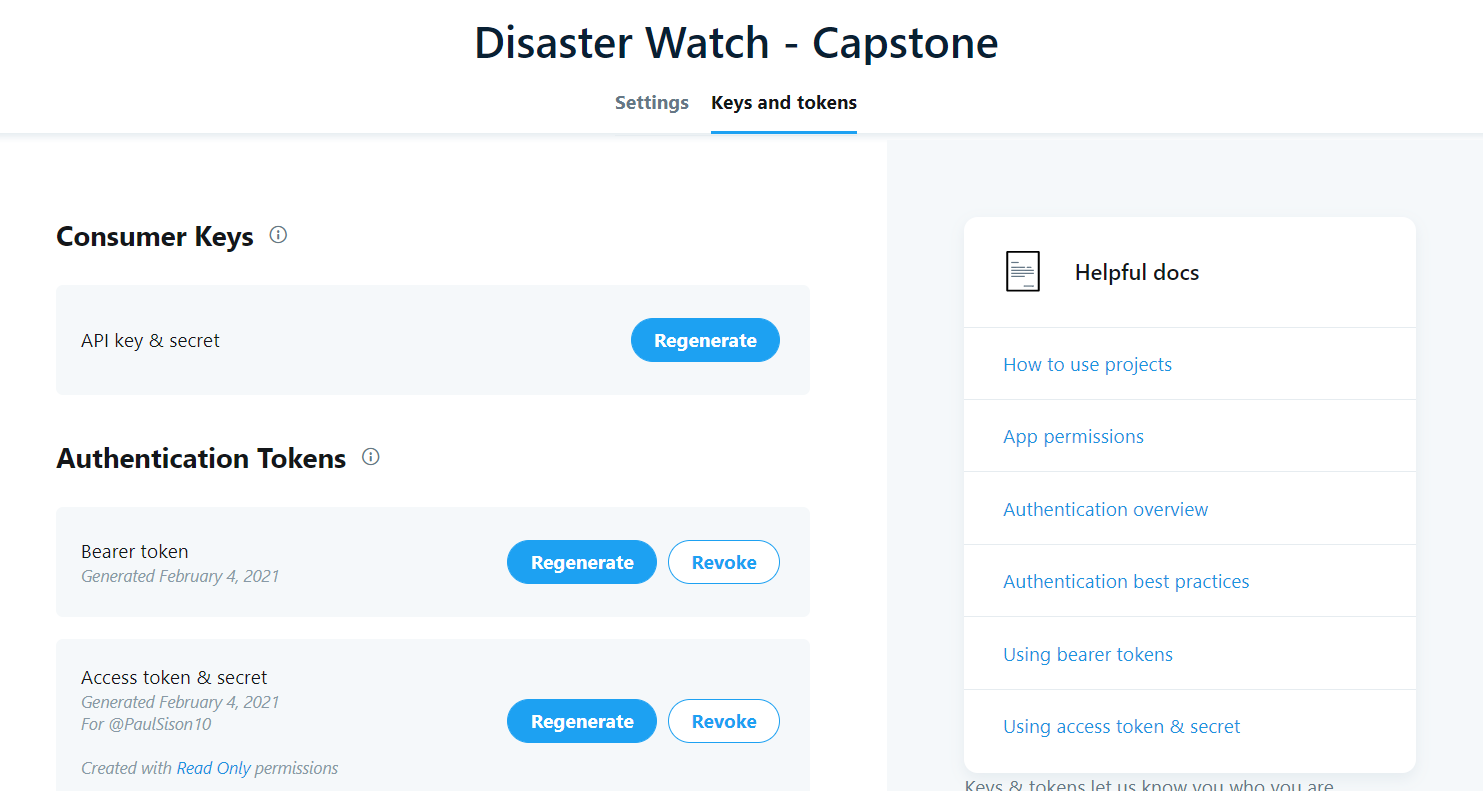
* 1. Run dbcrud.py on your terminal. This should create the tables (declared as classes in dbmodels.py) in the ‘postgres’ database on your local database server.



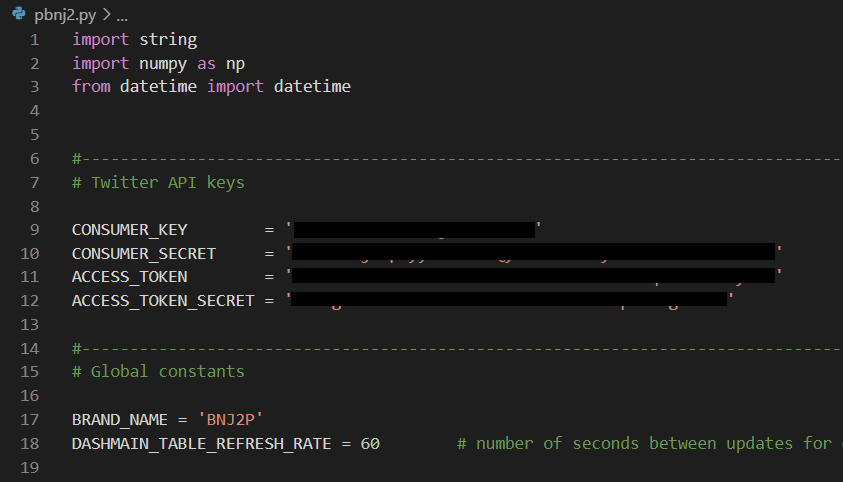
* 1. Check that a ‘tweet’ and a ‘hashtag’ table now exist in your local database.



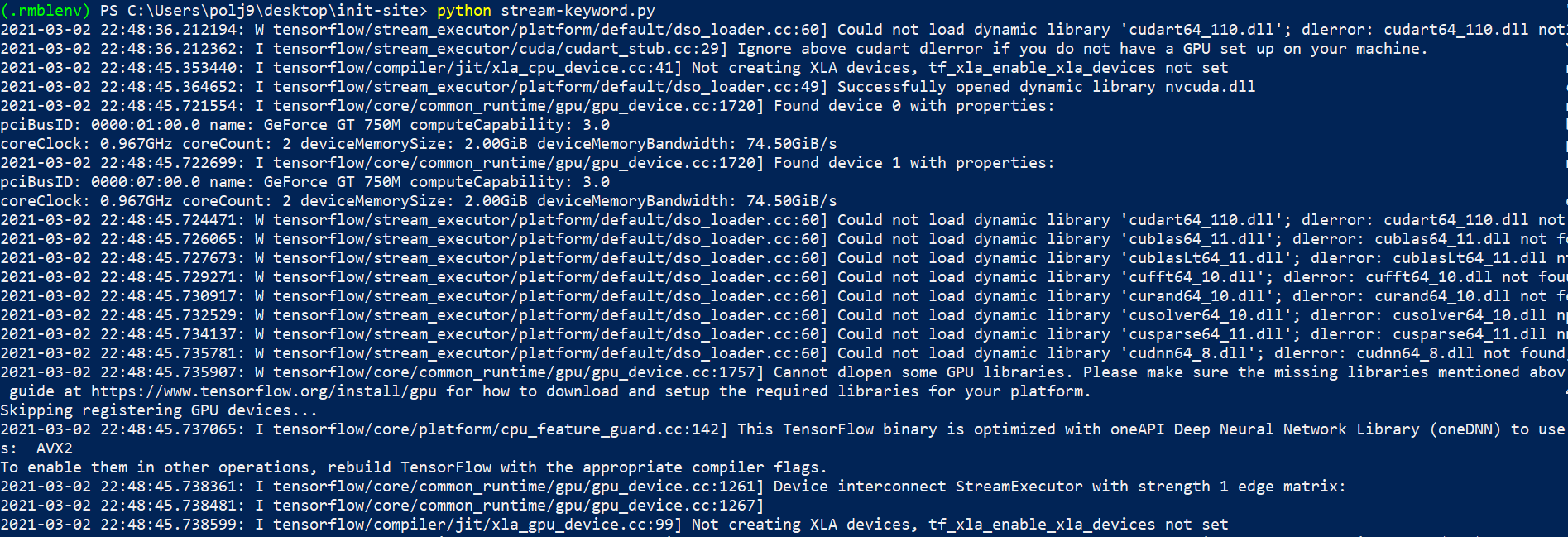
1. **Collect tweet data.** Just so we’ll have something to display on the web app.
   1. Go to the Twitter developer portal. Click on ‘Project & Apps’ > ‘Overview’ and create an App. Click on ‘Keys and Tokens’ tab. Generate your API and access keys. The API key pair will be your consumer keys and the Access key pair will be your access tokens.



* 1. Open pbnj2.py and put in your Twitter API credentials.



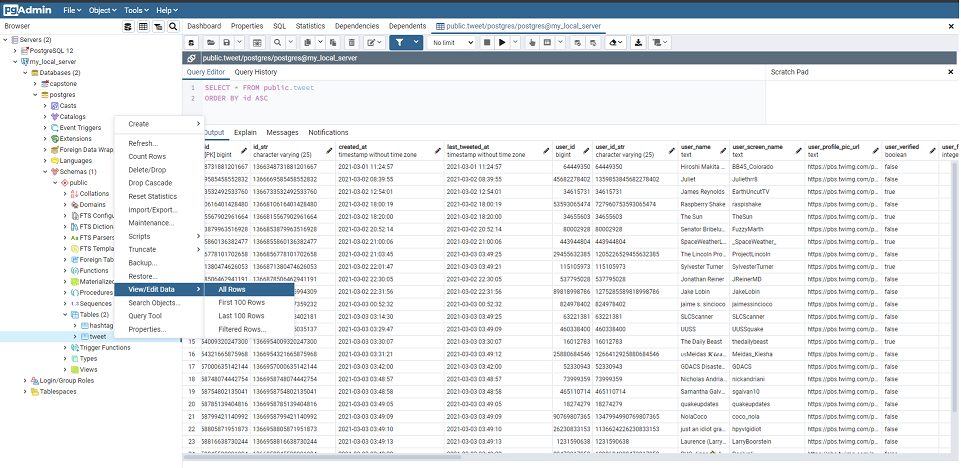
* 1. Run the streamer program to collect data. Don’t worry about the tensorflow setup when the program starts.



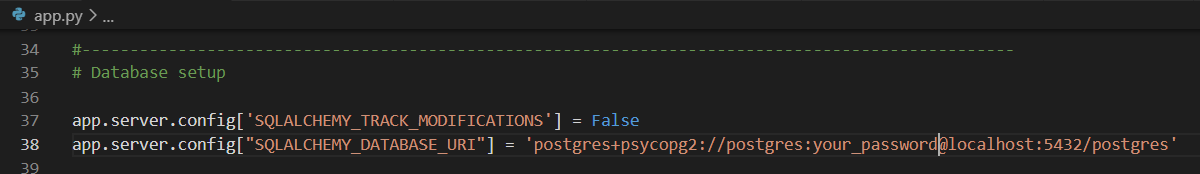
* 1. This is one example of a tweet. It’s a bad tweet by the way.



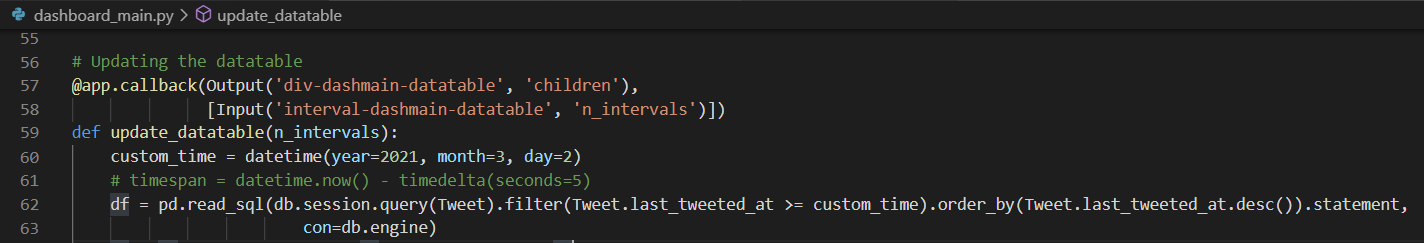
* 1. To stop the streamer program, you need to kill it from the task manager. It should be the Python process consuming the most memory.
  2. Check that data got stored in your ‘tweet’ table. Go back to pgadmin 4. Find the ‘tweet’ table. Right-click the table > view/edit data > all rows. Barring any errors, you should see that you’ve stored some data.



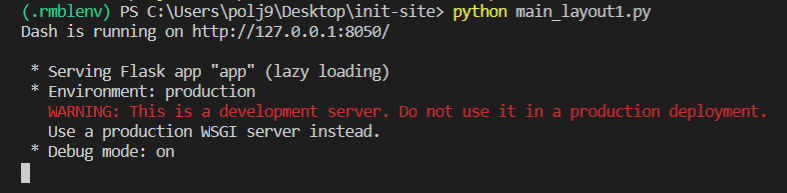
1. **Run the web app.** Yes, finally. But you need to make some changes first.
   1. Open app.py. Find the following lines and enter the same credentials from **7a**.



* 1. Open dashboard\_main.py. Modify the variable custom\_time to reflect today’s date. Yes, I could’ve done this but I forgot. :p I will explain the rest of the code at a later time.



* 1. You are now ready to run the web app. Execute the following command on your terminal.



* 1. Open a browser and visit the site at the URI specified above.