## Topic: Custom Management Command 8: Importing Large Data Using Celery & Reddis

***Speaker: Udemy Instructor Rathan Kumar | Notebook: Django: Automating Common Tasks***

---

1. After we installed REDIS, we installed REDIS as a package in our VIRTUAL ENVIRONMENT.

```
(env)
Rosilie@DELL MINGW64 C:/Users/Rosilie/AppData/Local/Programs/Microsoft VS Code (main)
$ pip install redis
Collecting redis
  Downloading redis-5.0.8-py3-none-any.whl (255 kB)
     |████████████████████████████████| 255 kB 1.7 MB/s
Collecting async-timeout>=4.0.3
  Downloading async_timeout-4.0.3-py3-none-any.whl (5.7 kB)
Installing collected packages: async-timeout, redis
Successfully installed async-timeout-4.0.3 redis-5.0.8
WARNING: You are using pip version 21.2.4; however, version 24.2 is available.
You should consider upgrading via the 'C:\Users\Rosilie\OneDrive\Desktop\LEARNING DJANGO PROJECTS\AutomatingComm
onTasks\env\Scripts\python.exe -m pip install --upgrade pip' command.
(env)
Rosilie@DELL MINGW64 C:/Users/Rosilie/AppData/Local/Programs/Microsoft VS Code (main)
$
* History restored
```

2. You installed CELERY.

```
$ pip install celery
```

3. Run this to create a Celery WOrker:

windows:

```
$ celery -A autocommontasks_main worker --loglevel=info --pool=solo
```

macos:

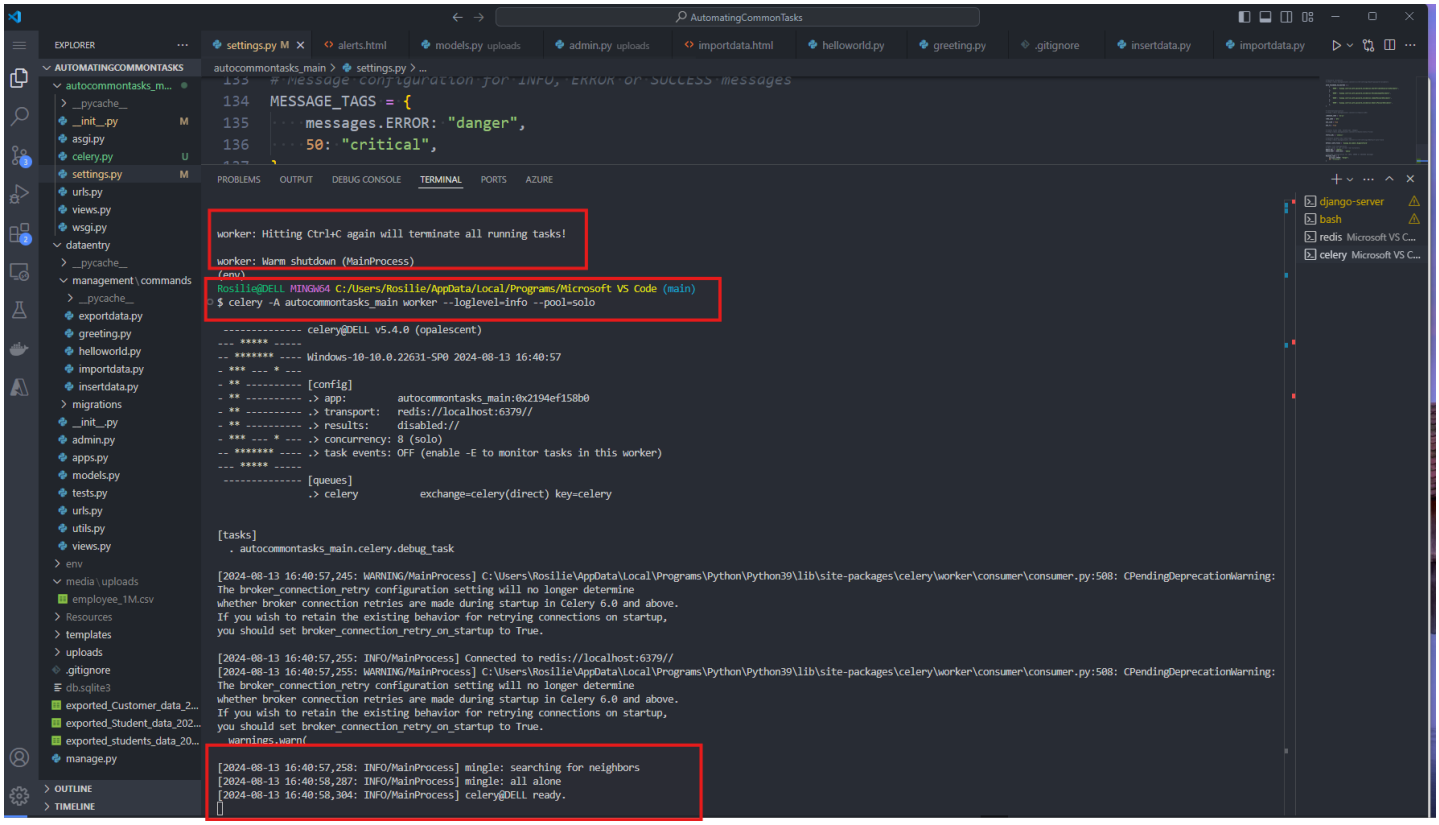
```
$ celery -A autocommontasks_main worker --loglevel=info
```

This results to this with an error:

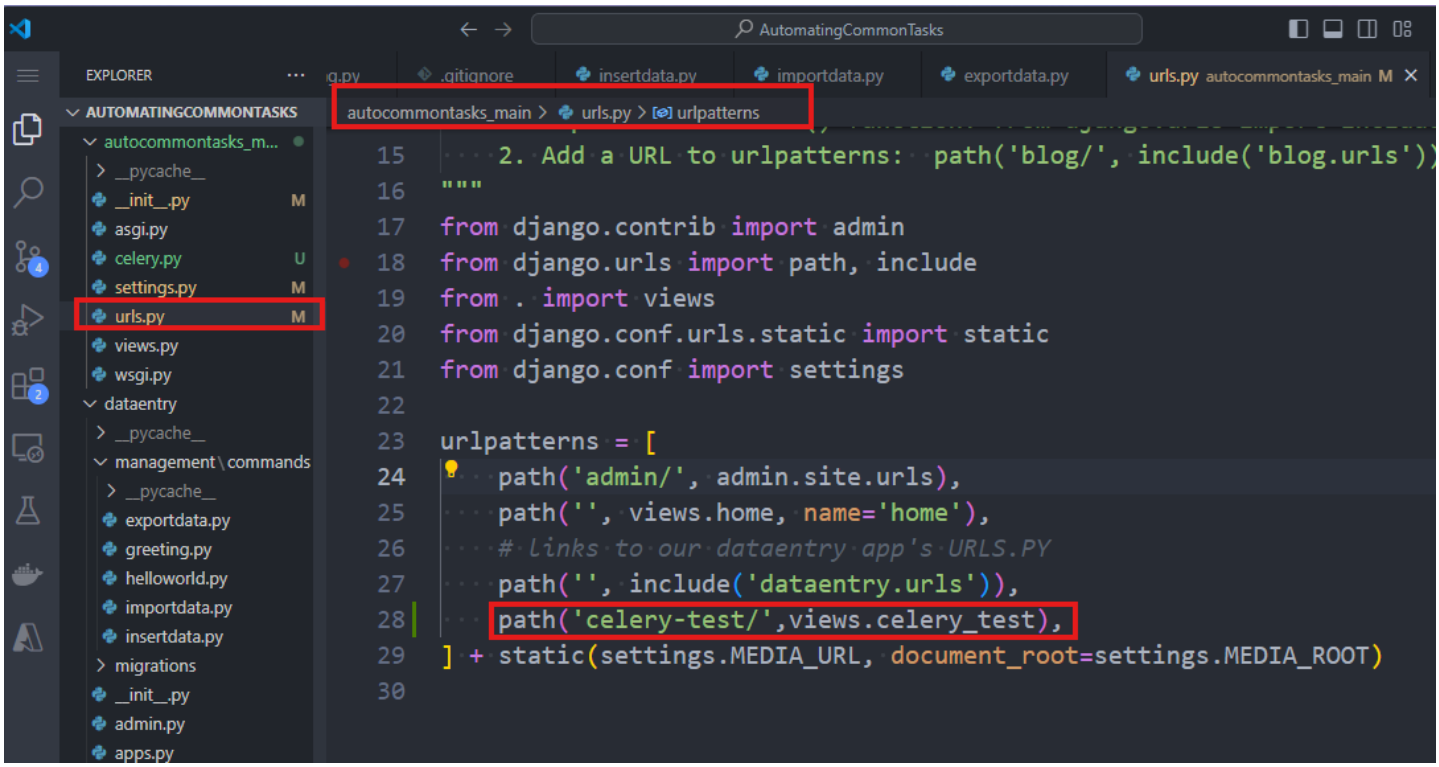
```
(env)
Rosilie@DELL MINGW64 C:/Users/Rosilie/AppData/Local/Programs/Microsoft VS Code (main)
$ celery -A autocommontasks_main worker --loglevel=info --pool=solo
[2024-08-13 15:52:03,052: WARNING/MainProcess] No hostname was supplied. Reverting to default 'localhost'

 -------------- celery@DELL v5.4.0 (opalescent)
--- ***** -----
-- ******* ---- Windows-10-10.0.22631-SP0 2024-08-13 15:52:03
- *** --- * ---
- ** ---------- [config]
- ** ---------- .> app:         autocommontasks_main:0x278ef7c2a60
- ** ---------- .> transport:   amqp://guest:**@localhost:5672//
- ** ---------- .> results:     disabled://
- *** --- * --- .> concurrency: 8 (solo)
-- ******* ---- .> task events: OFF (enable -E to monitor tasks in this worker)
--- ***** -----
 -------------- [queues]
                .> celery           exchange=celery(direct) key=celery


[tasks]
  . autocommontasks_main.celery.debug_task

[2024-08-13 15:52:03,097: WARNING/MainProcess] C:\Users\Rosilie\AppData\Local\Programs\Python\Python39\lib\site-packages\celery\worker\consumer\consumer.py:508: CPendingDeprecationWarning: The broker_con
_retry configuration setting will no longer determine
whether broker connection retries are made during startup in Celery 6.0 and above.
If you wish to retain the existing behavior for retrying connections on startup,
you should set broker_connection_retry_on_startup to True.
  warnings.warn(

[2024-08-13 15:52:05,136: ERROR/MainProcess] consumer: Cannot connect to amqp://guest:**@127.0.0.1:5672//: [WinError 10061] No connection could be made because the target machine actively refused it.
Trying again in 2.00 seconds... (1/100)

[2024-08-13 15:52:09,168: ERROR/MainProcess] consumer: Cannot connect to amqp://guest:**@127.0.0.1:5672//: [WinError 10061] No connection could be made because the target machine actively refused it.
Trying again in 4.00 seconds... (2/100)

[2024-08-13 15:52:15,252: ERROR/MainProcess] consumer: Cannot connect to amqp://guest:**@127.0.0.1:5672//: [WinError 10061] No connection could be made because the target machine actively refused it.
Trying again in 6.00 seconds... (3/100)

[2024-08-13 15:52:23,339: ERROR/MainProcess] consumer: Cannot connect to amqp://guest:**@127.0.0.1:5672//: [WinError 10061] No connection could be made because the target machine actively refused it.
Trying again in 8.00 seconds... (4/100)

[2024-08-13 15:52:33,452: ERROR/MainProcess] consumer: Cannot connect to amqp://guest:**@127.0.0.1:5672//: [WinError 10061] No connection could be made because the target machine actively refused it.
Trying again in 10.00 seconds... (5/100)
```

4. To correct this, we have to update our SETTINGS.PY

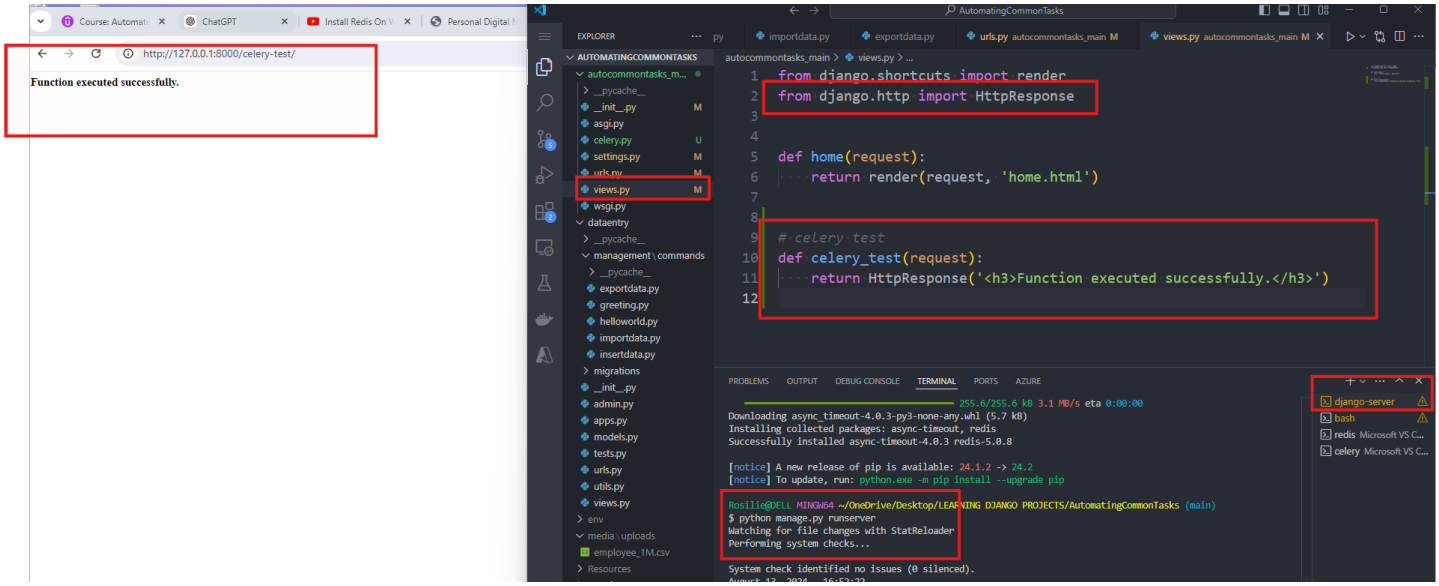

In the bash terminal, press CTRL + C to terminate the process and try again. Your celery should be ready.



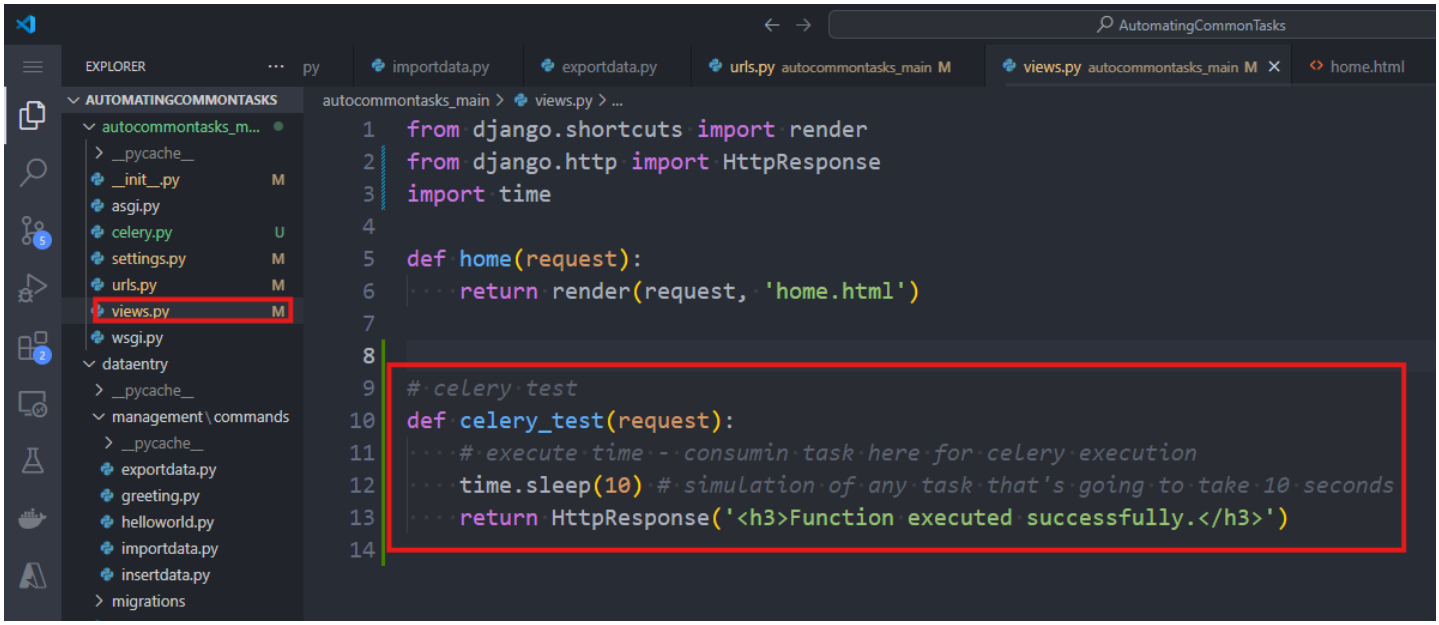

5. To use CELERY for testing in our project, create a new URL in our root project's URLS.PY



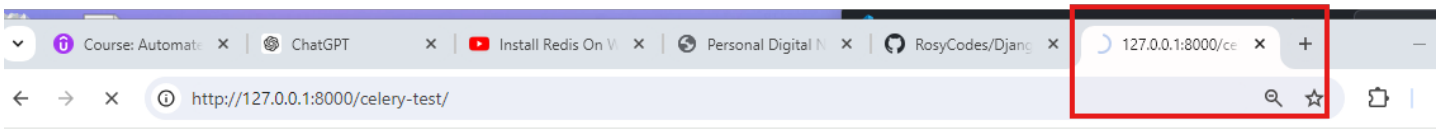

We run our django server and att the url in our browser: `http://127.0.0.1:8000/celery-test/`



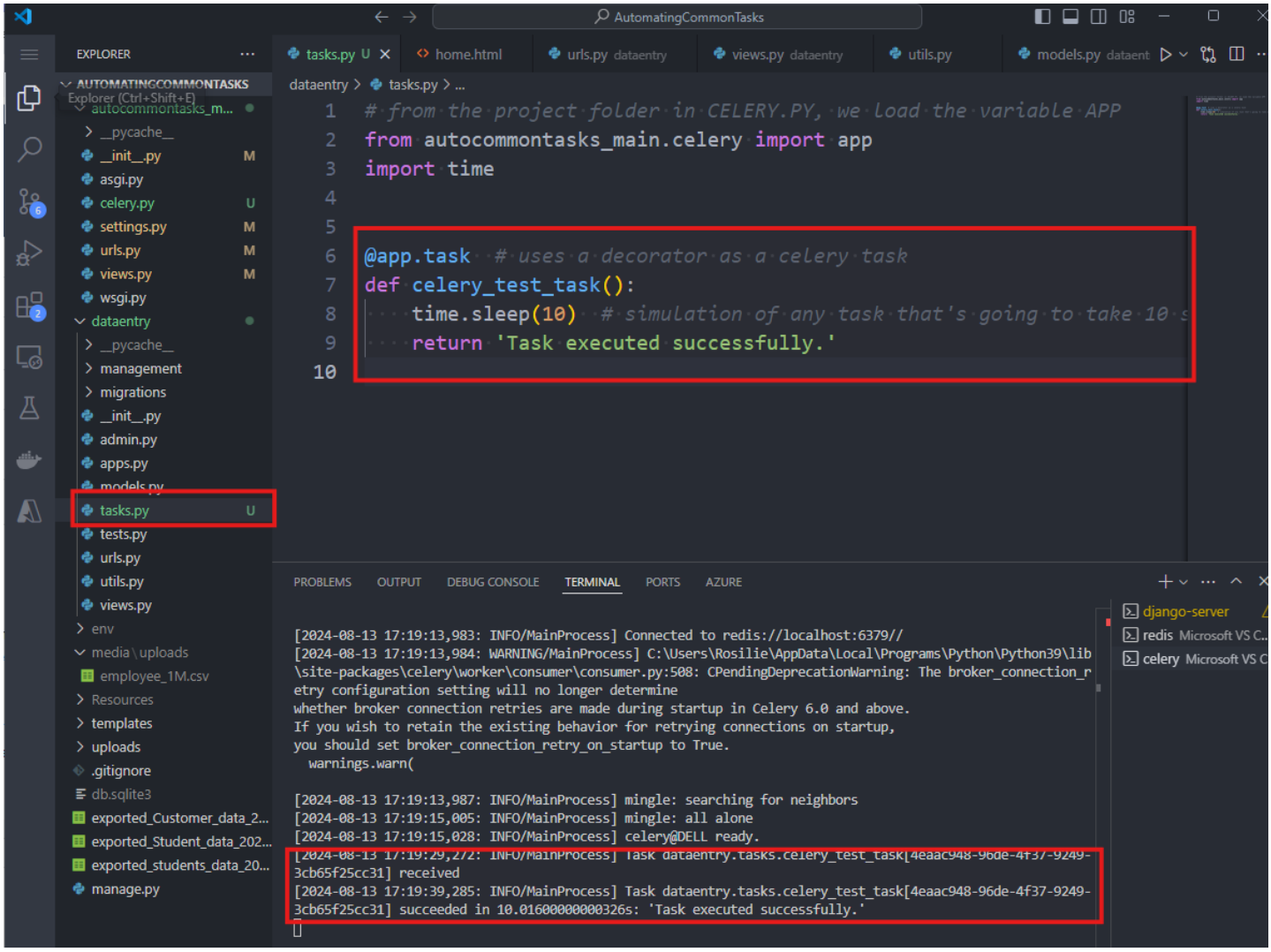

6. Now update our CELERY_TEST FUNCTION to perform time-consuming operations.



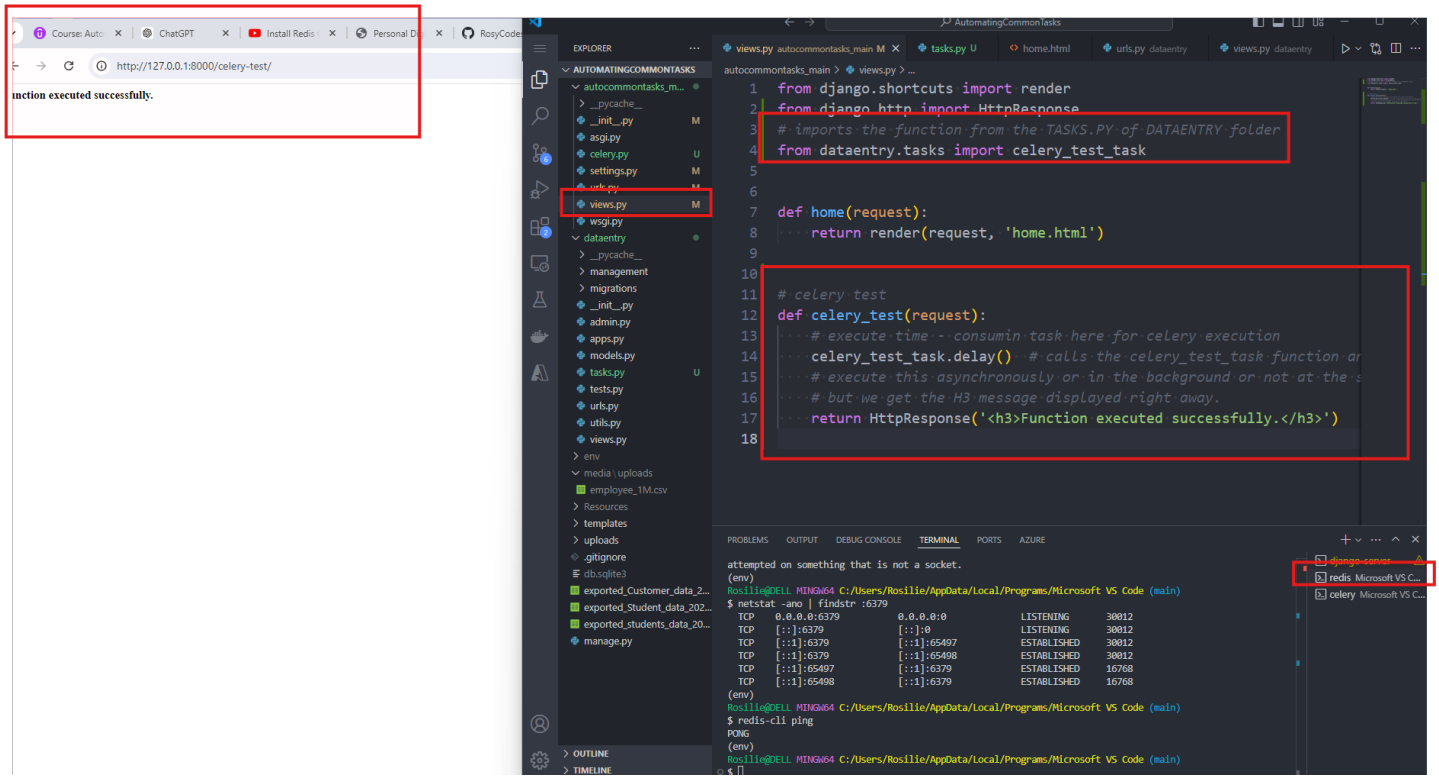

So, when you reload your page, in the background, it will load 10 seconds (Sleep)




7. We want to pass then the SLEEP(10) TASK to celery, so that our project or user can do or see other things while Celery is working on something else. To do this, in the DATAENTRY folder, create a new file, TASKS.PY and update as




8. To test all this, LOAD your URL `http://127.0.0.1:8000/celery-test/` again, click on your CELERY TERMINAL, and it should RECEIVE and DISPLAY the message 'TASK EXECUTED SUCCESSFULLY' while we wee our webpage with H3 tag showing the message right away.

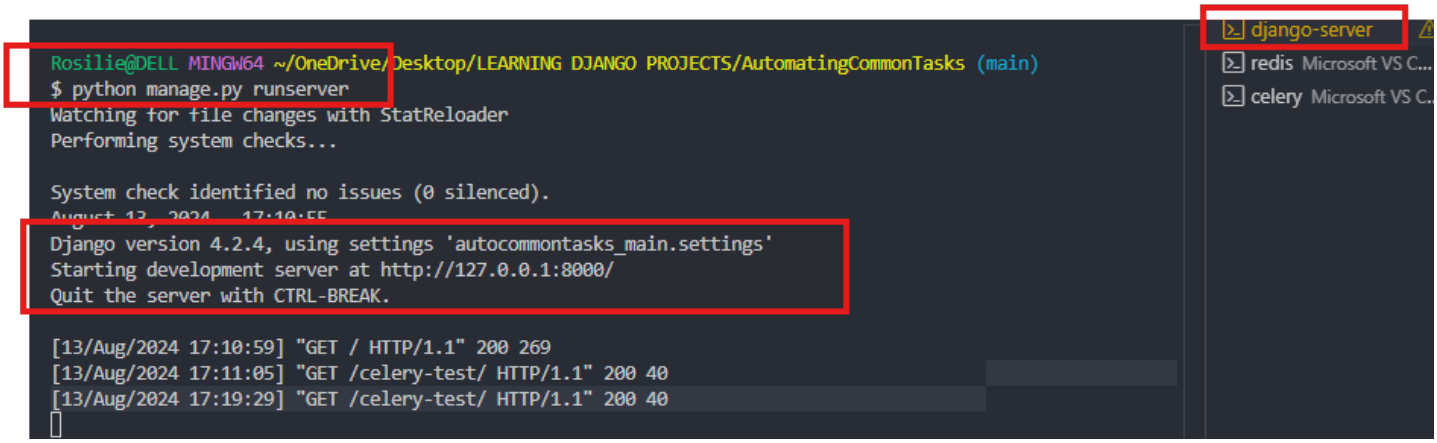



***IMPORTANT REMINDER:***

So, when you run your django project with celery and redis, THERE SHOULD BE 3 BASH TERMINALS AND YOU NEED TO NAME THEM APPROPRIATELY:

1. DJANGO-SERVER- this is where you will run your python server to run your Django project:

```
$ python manage.py runserver
```

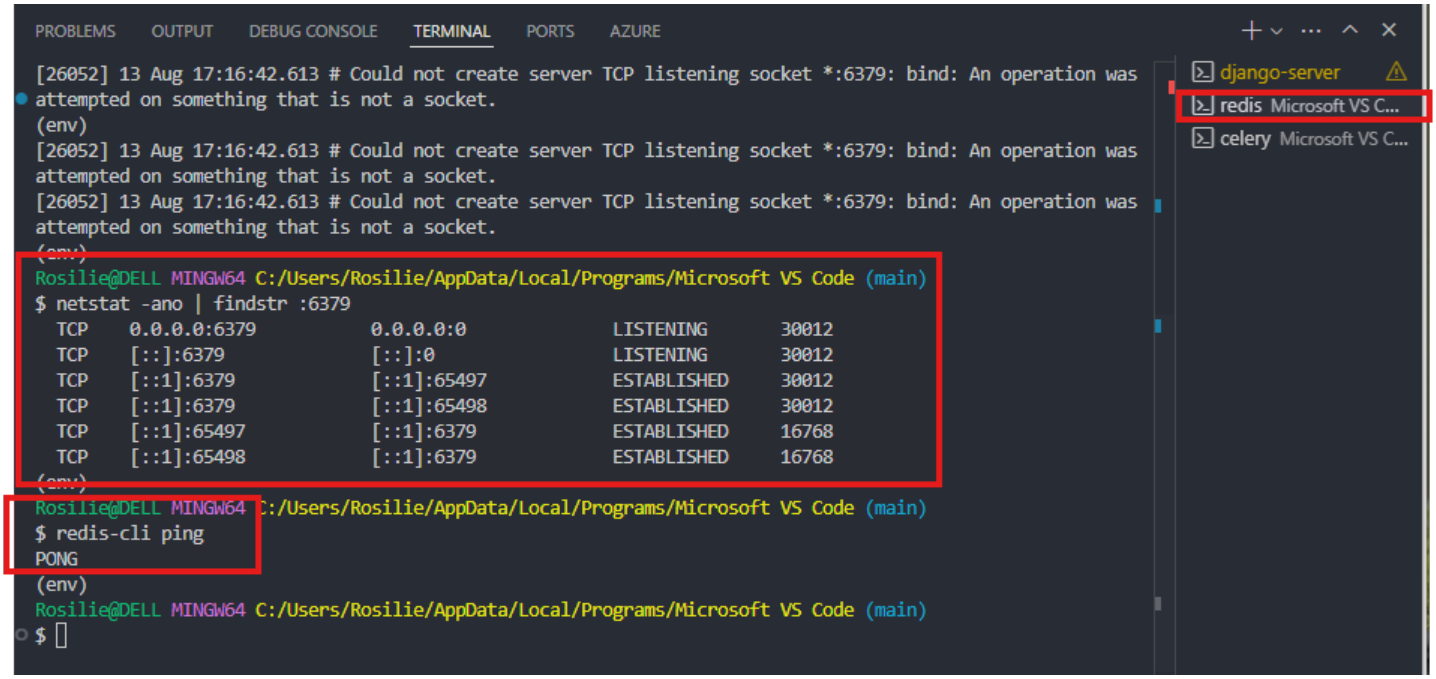



2. REDIS - this is where you run your REDIS-SERVER. Every time you start this, it will create a new process of REDIS, so you can simply kill it or ignore it. As long as when you PING your REDIS SERVER, it returns a PONG message, then you are fine. You have multiple processes (TCP) here because you have started your REDIS-SERVER multiple times.

```
$ redis-server
```

```
$ netstat -ano | findstr :6379
```
(this is to see what process is listening to the port 6379 (Redis)

```
$ redis-cli ping
```
(Run this before you run the REDIS-SERVER so you wont have several TCPs)




3. CELERY- this is where you will run this code whenever you need to launch a new task or when you reload your Django page. Meaning, you have to launch this command again if you have made new changes to your CELERY TASKS like changing 10 seconds to 5 seconds to see the effect.

*autocommontasks_main - this is your Django Project Name

```
$ celery -A autocommontasks_main  worker --loglevel=info --pool=solo
```