# **CSYE 7245 - Big Data Sys and Int Analytics**

## **Lab - SQLAlchemy**

### **Team Information:**

|  |  |
| --- | --- |
| **Name** | **NUID** |
| Tanvi Gurav | 001306848 |
| Keerti Ojha | 001050173 |
| Priyanka Malpekar | 001302741 |

**Lab Completion Date:** **12th February’21**

# **About**

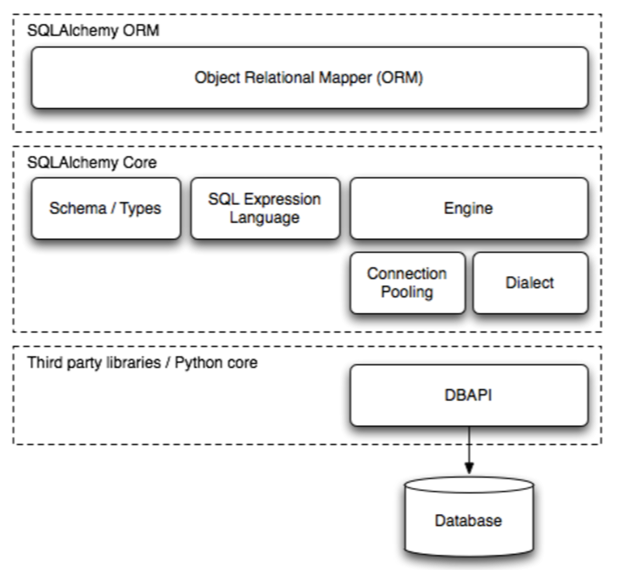
This lab demonstrates how to leverage SQL concepts through a traditional Object-Oriented Programming approach with the help of SQLAlchemy.



**What is SQLAlchemy?**

SQLAlchemy is the Python-SQL toolkit and Object Relational Mapper that gives application developers the flexibility of SQL. It provides a full suite of well known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

The general structure can be illustrated as follows:



SQLAlchemy uses object-relational mapper (ORM), an Object-relational mapping is a method of coding that translates data between different system types using an object-oriented programming language and converting function calls to SQL statements.

# **Dataset**

The lab was implemented using a sample data for loading basic customer information:

|  |
| --- |
| **Table: Customer** |
| + id: number |
| + name: string |
| + age: number |
| + address: string |
| + email: string |

# **Experiment Setup**

The following are the prerequisite setup we made for the implementation of lab:

**1. Installing Python on our machine using the following commands:**

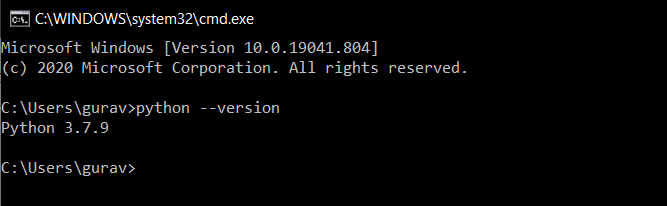
**For Ubuntu/Linux:**

sudo apt install python3

sudo apt install python3-pip

**For Windows:**

Link for downloading Python binaries:<https://www.python.org/downloads/>

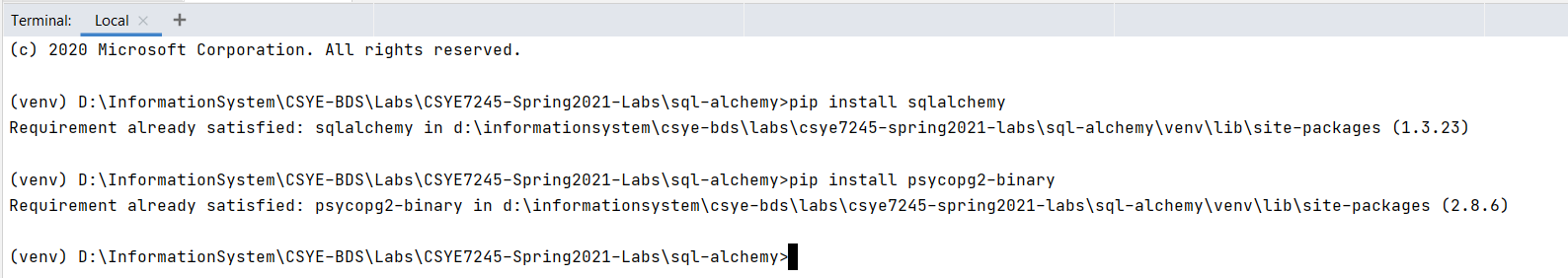


**2. After Python is configured, we installed SQLAlchemy:**

Create a new project in Pycharm and run the following commands for installation the libraries of SQLAlchemy in Python environment:

pip install sqlalchemy

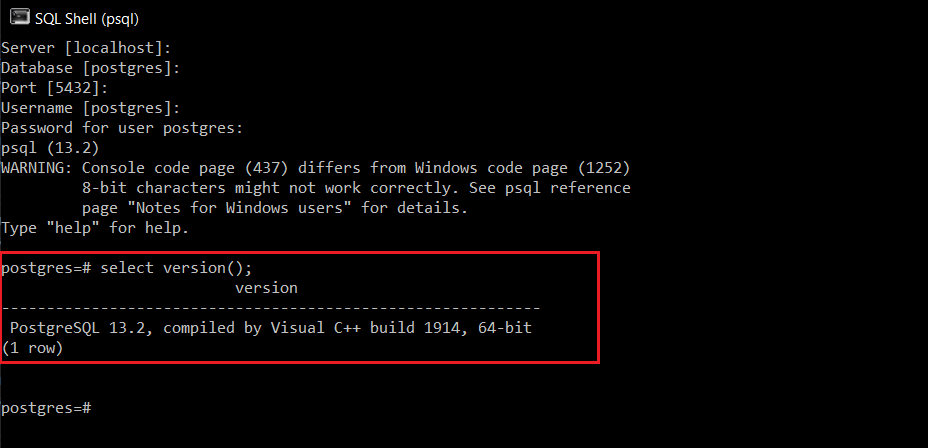
pip install psycopg2-binary



**3. Setting up the relational PostgreSQL database:**

We downloaded the [PostgreSQL binaries](https://www.enterprisedb.com/downloads/postgres-postgresql-downloads) and ran the executable file

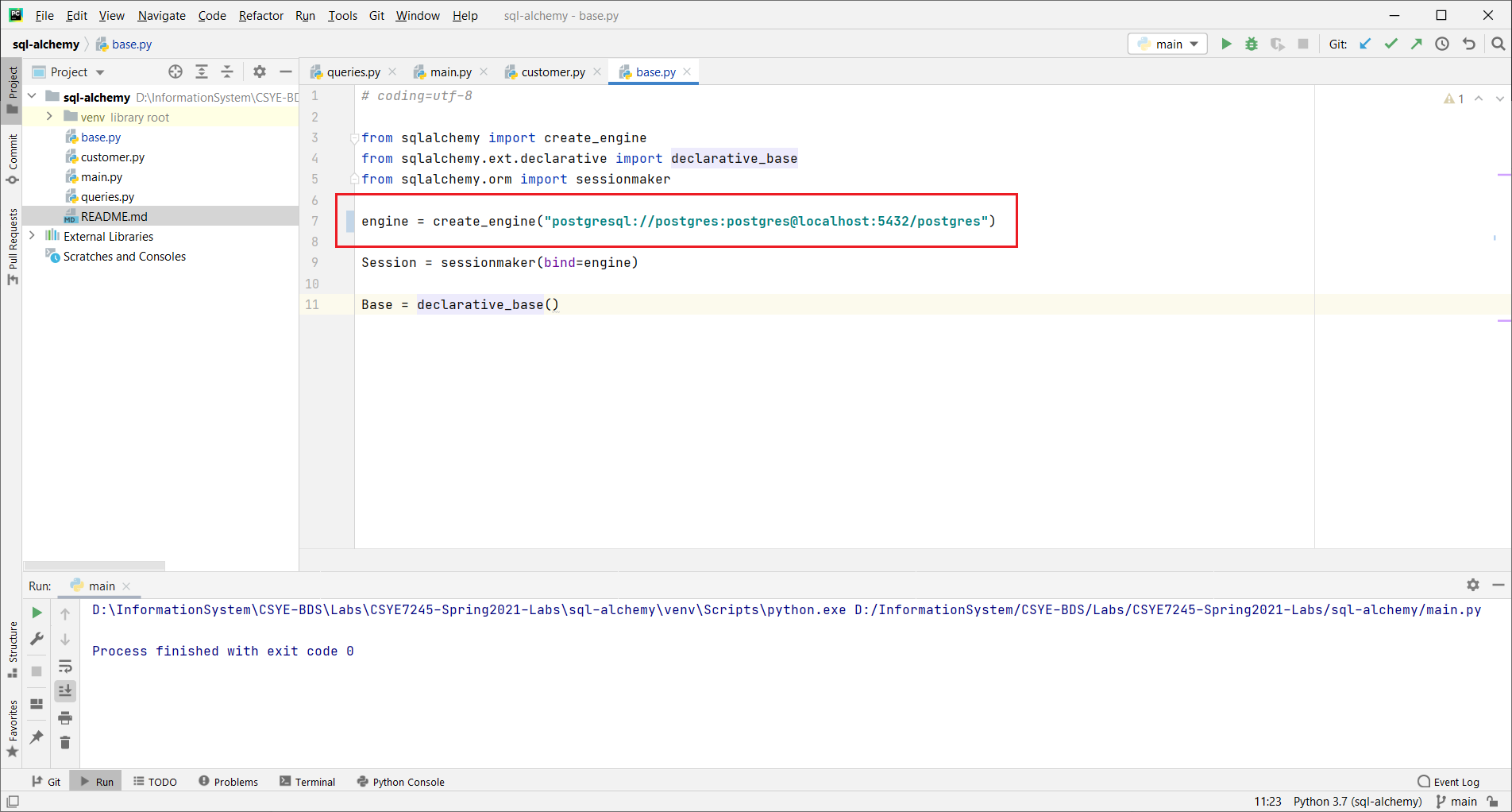




# **Tests Cases**

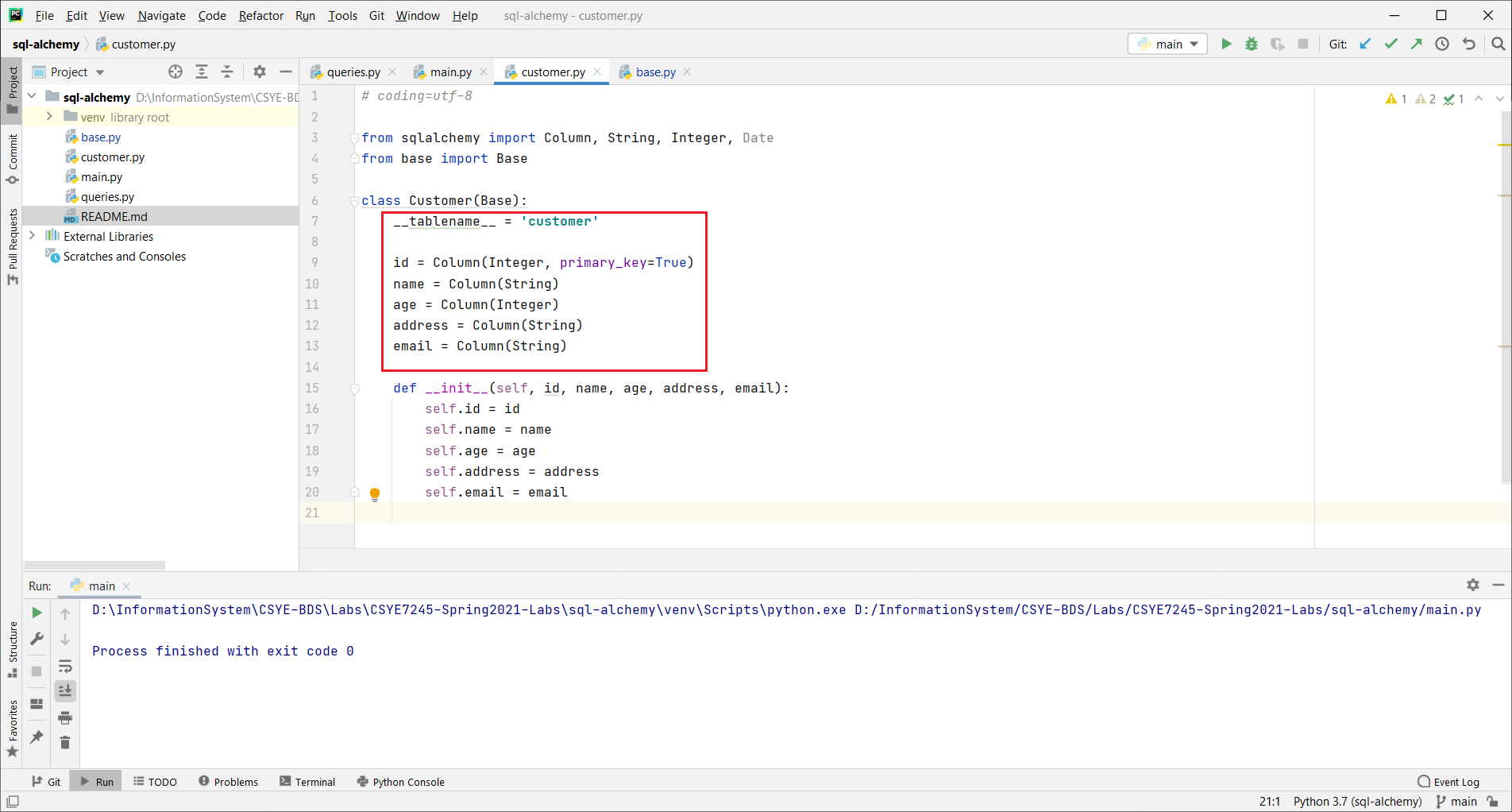
1. We entered the connection parameters to connect to the PostgreSQL server instance in program base.py

**Syntax:** postgresql://username:password@host:port/database

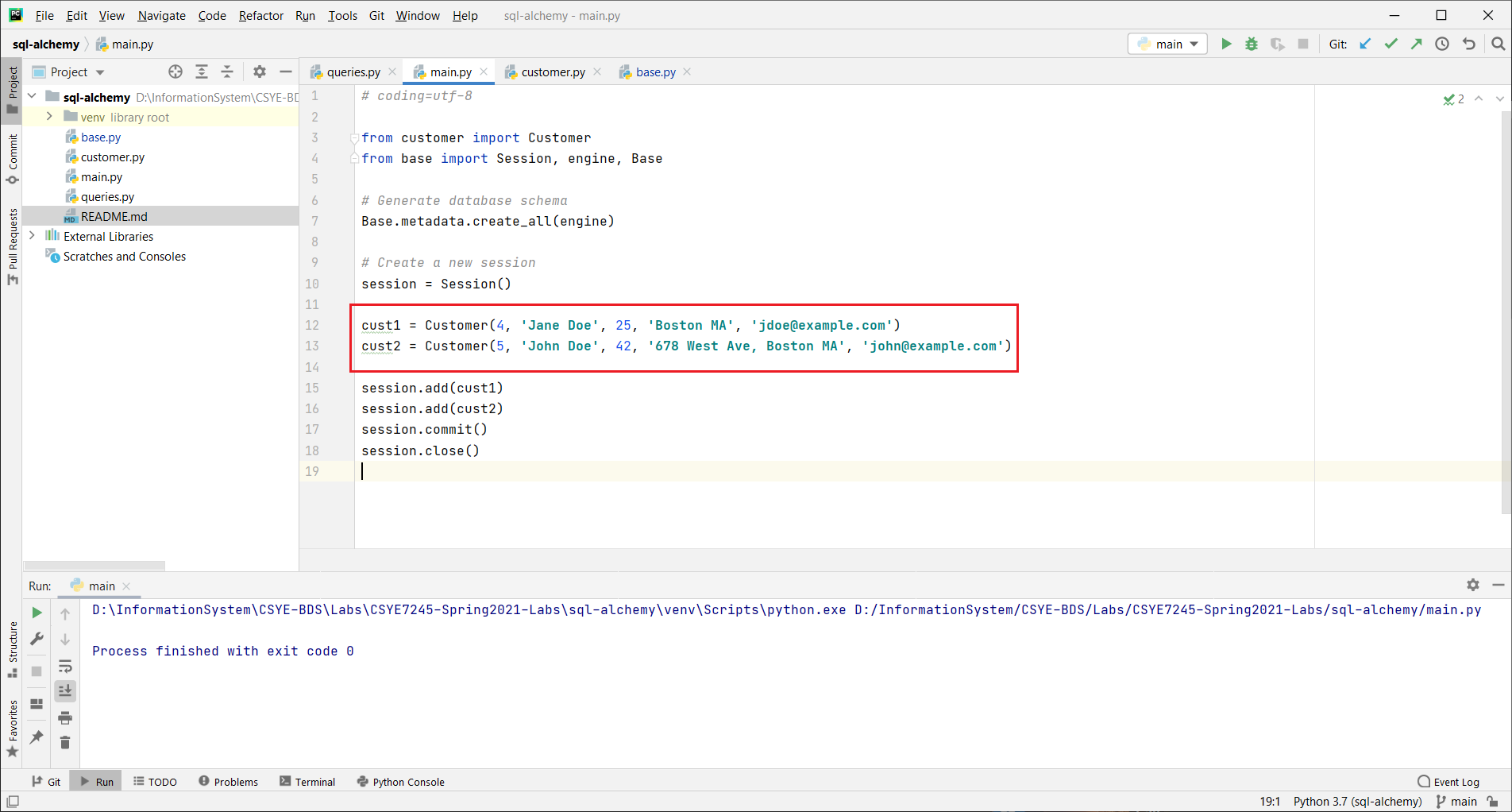


After entering the connection details to the database the engine does not connect to the database instantly. This process is postponed to when it's needed (like when we submit a query, or when create/update a row in a table). The create\_engine() returns an instance of Engine, and it represents the core interface to the database, adapted through a dialect that handles the details of the database and DBAPI in use.

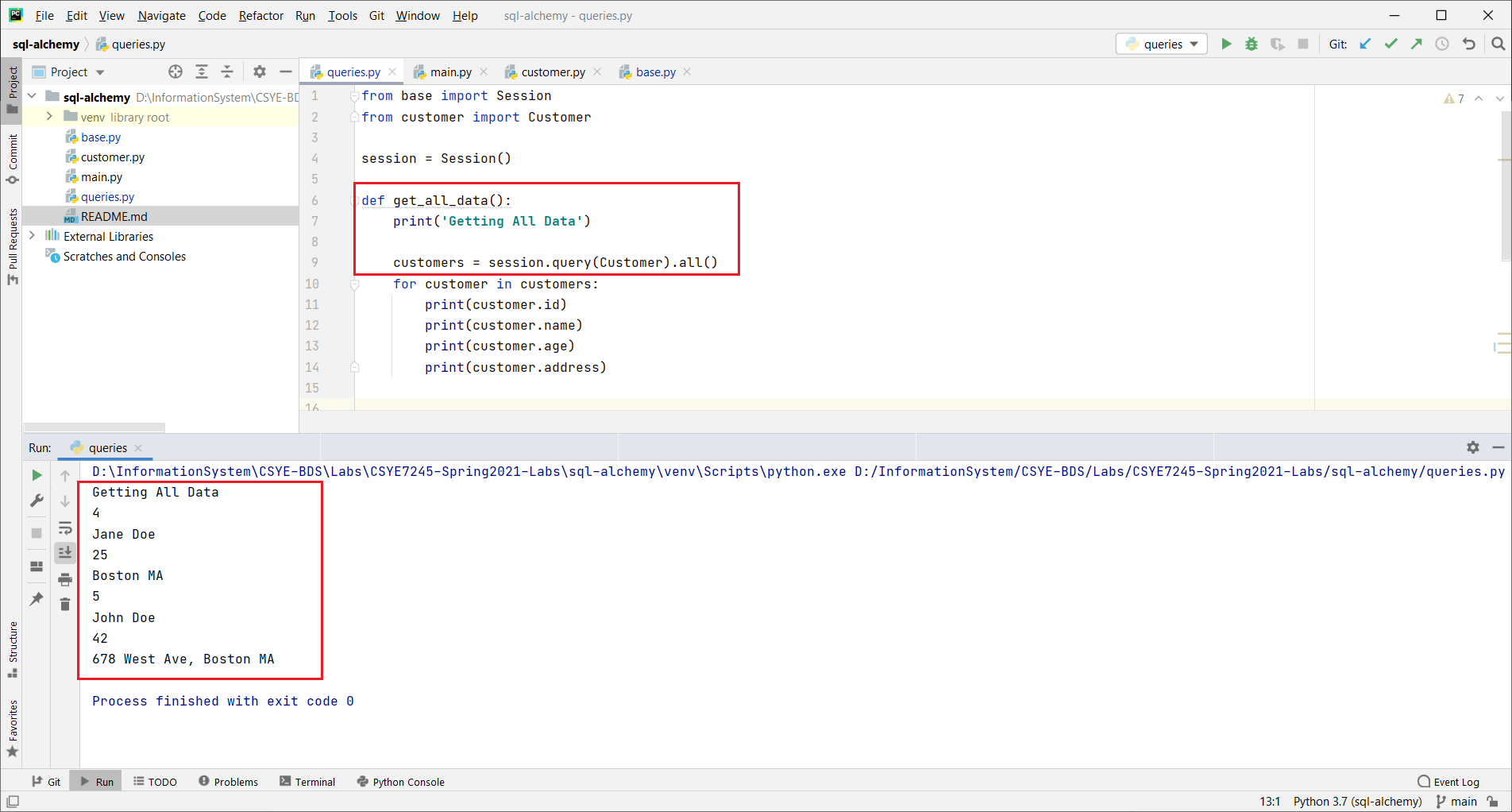
2. The customer.py is a sample Python class containing 5 attributes that we'd like to be converted to a table and attributes as its columns



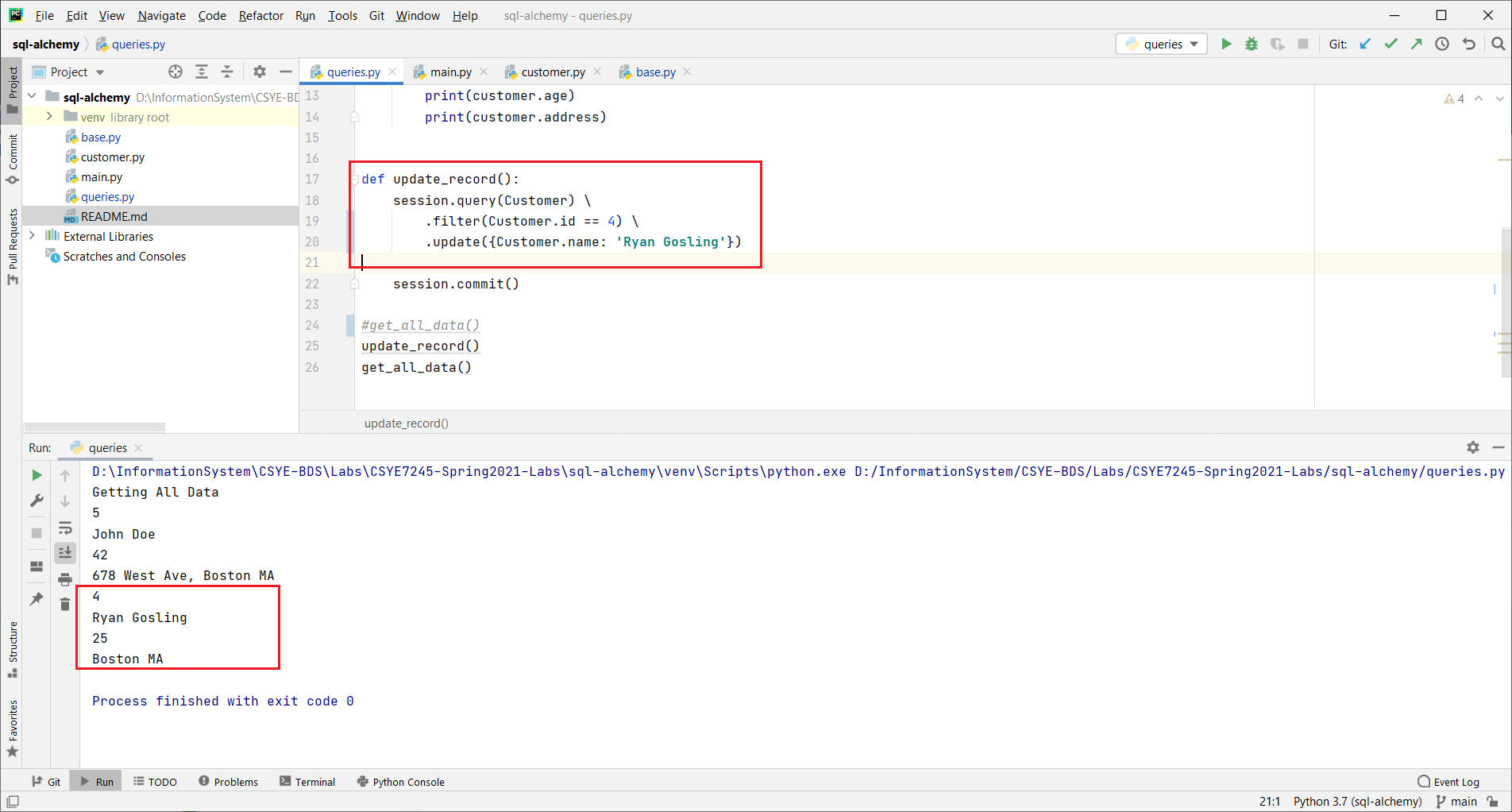
3. The main.py program inserts data using the newly created class



4. The queries.py contains a get\_all\_data() function which is used to query the table and retrieve the inserted records in the output console



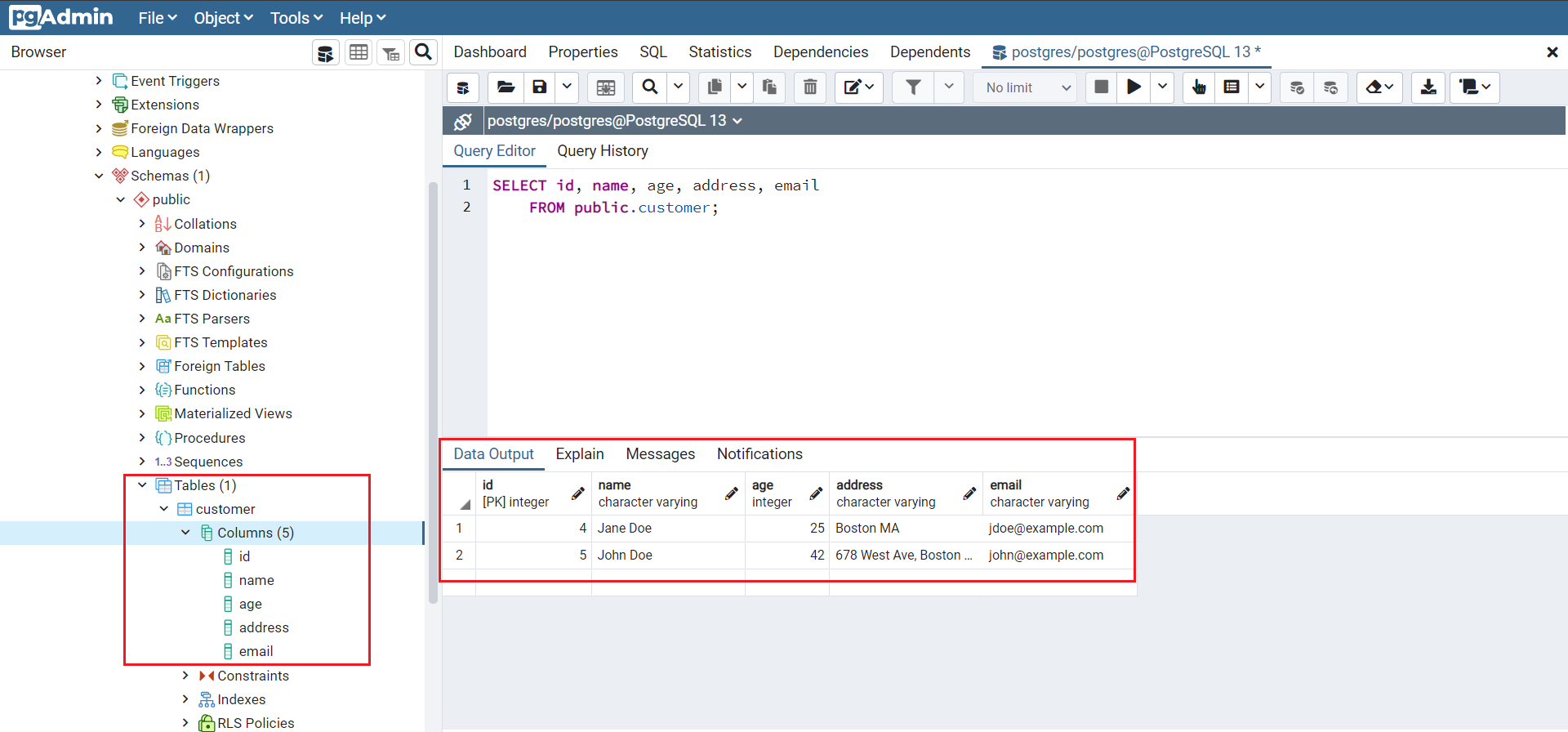
Apart from the get\_all\_data() function, the update\_record() function is used to update any record based on the id using filter()



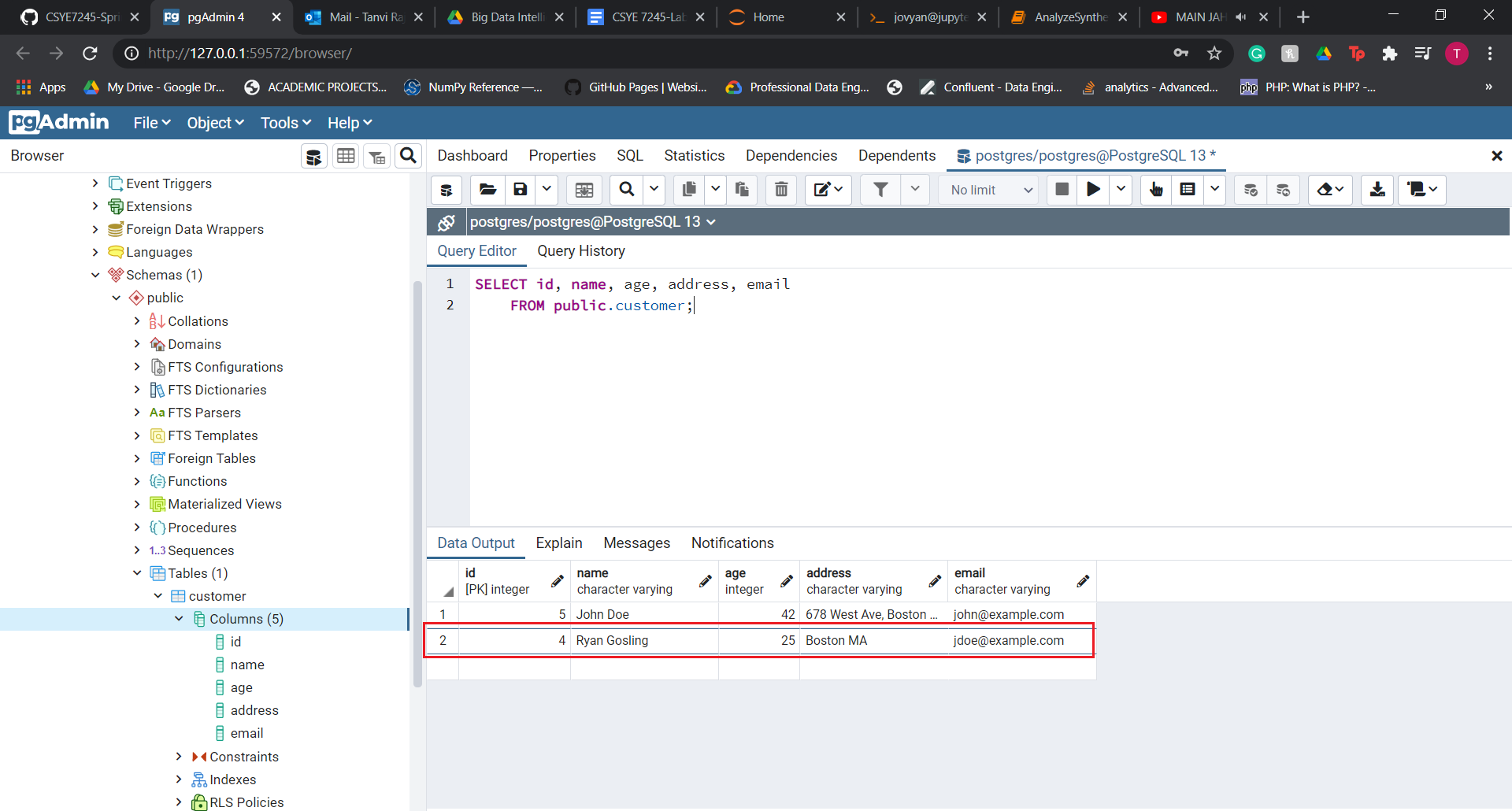
# **Results**

Once the engine is created and data is inserted using main.py program, we login to pgadmin console to check if the data in inserted correctly into the intended database in our case, the data is inserted in:

* Database: postgres
* Table: Customer
* Columns: id, name, age, address, email



Further, after executing the update\_record() function for changing the name with id 4, we see the changes reflected into the database with updated name from **‘Jane Doe’** to **‘Ryan Gosling’**



# 

# Lessons Learned

1. Learned how to write an object-oriented Python program which encapsulates SQL concepts, connects to a relational database and translates function calls to SQL queries.
2. Learned how to insert data into the relational database using a Python function with the help of SQLAlchemy libraries
3. Learned to update data from a function and retrieve the data in the Python output console without querying the database which cut downs switching back and forth between writing Python and SQL

# References

[auth0 blog](https://auth0.com/blog/sqlalchemy-orm-tutorial-for-python-developers/?utm_source=medium&utm_medium=sc&utm_campaign=sqlalchemy_python)

<https://www.sqlalchemy.org/>

<https://docs.sqlalchemy.org/en/13/orm/tutorial.html>