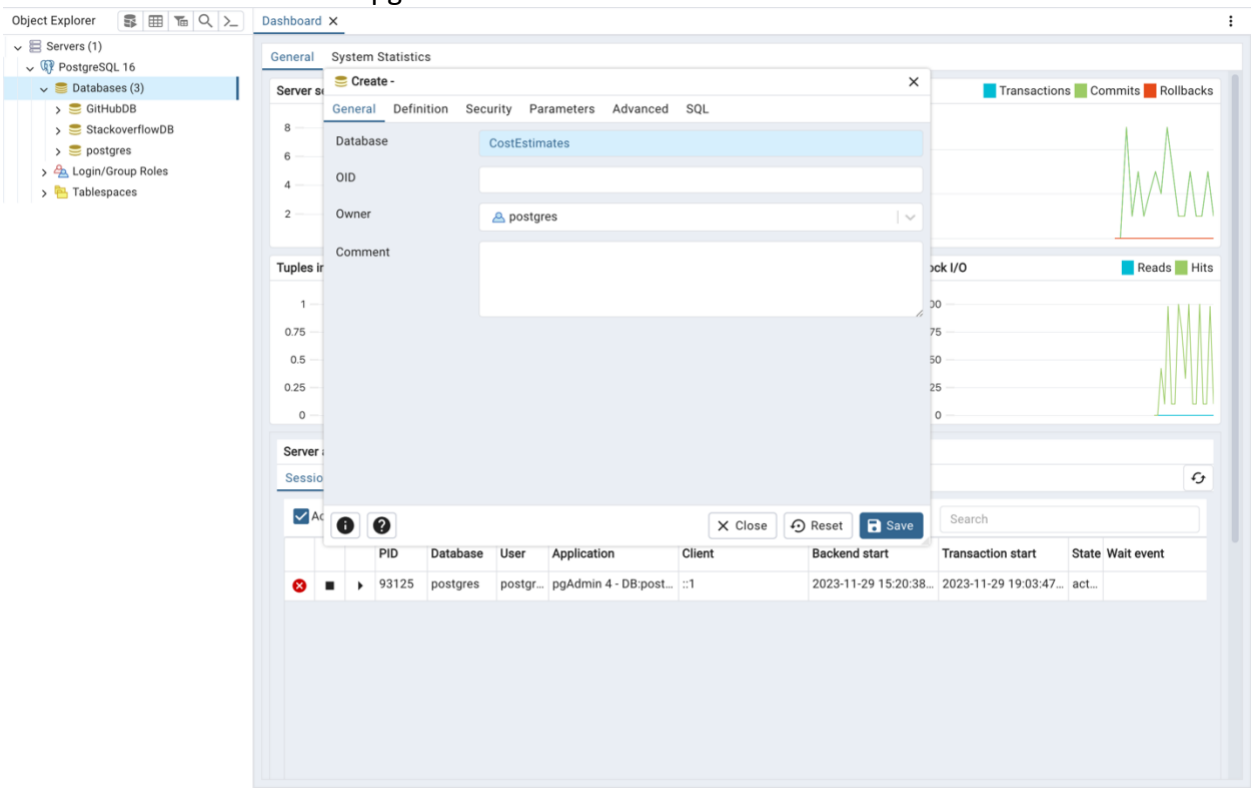
Group 17

The use of generative AI to generate estimates for the project plan, design document, source code, test plan and test cases, and user API documentation.

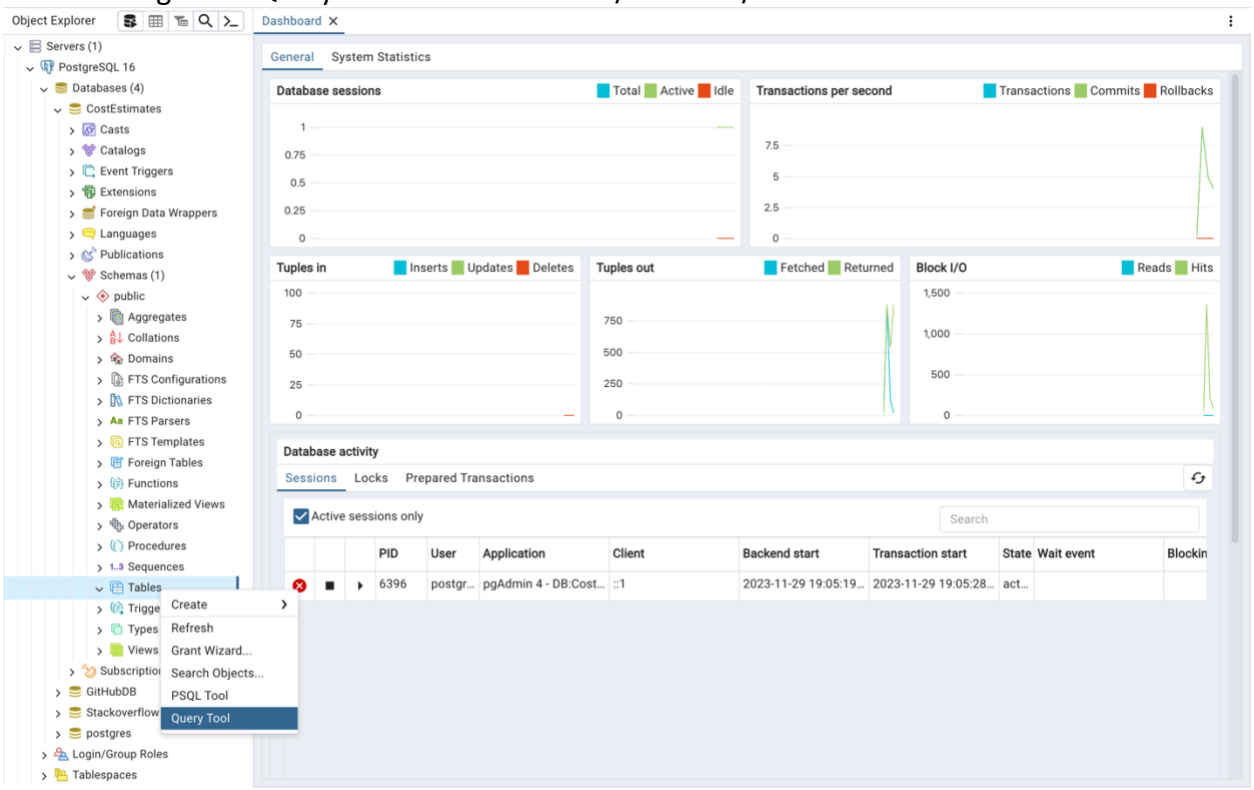
Kamble Soham skamble2@hawk.iit.edu <A20517098>
Chawla Bhavya bchawla@hawk.iit.edu <A20516957>

Total Lines of Code - 300

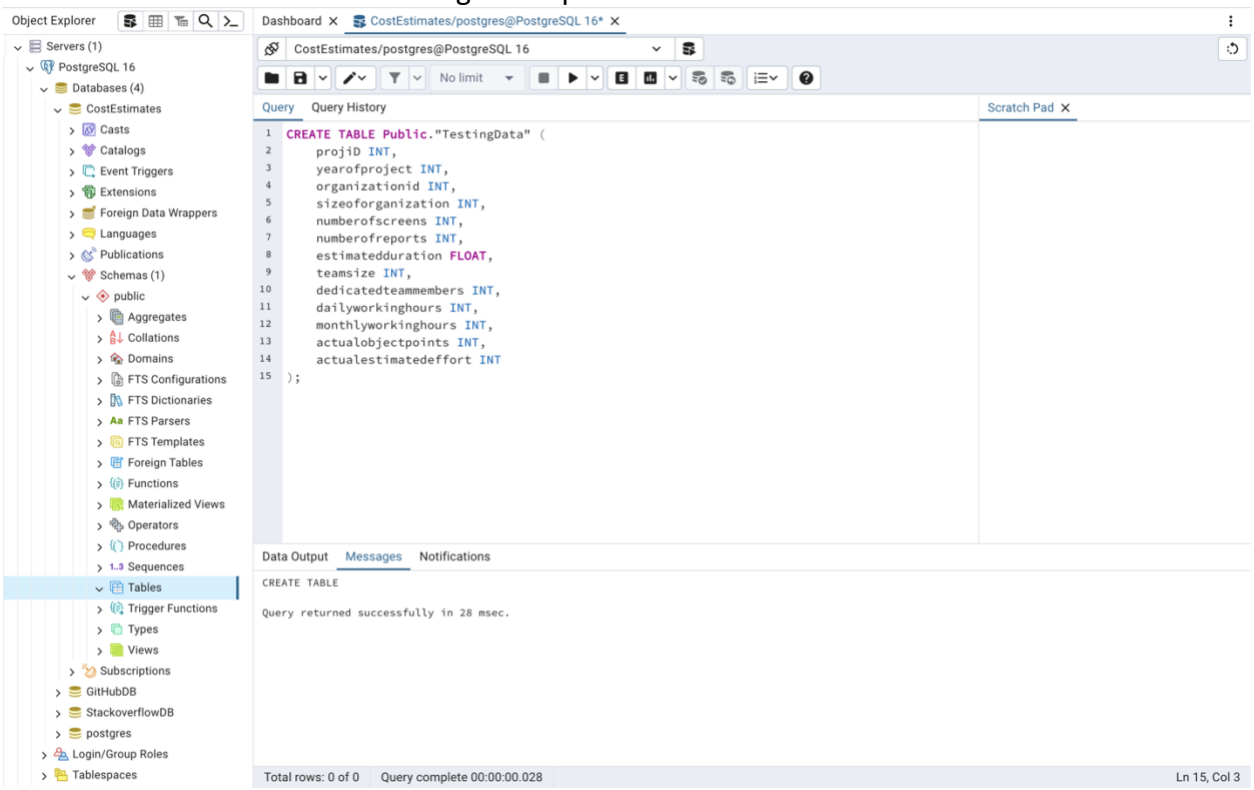
1. Create a database in pgAdmin and name it CostEstimates.



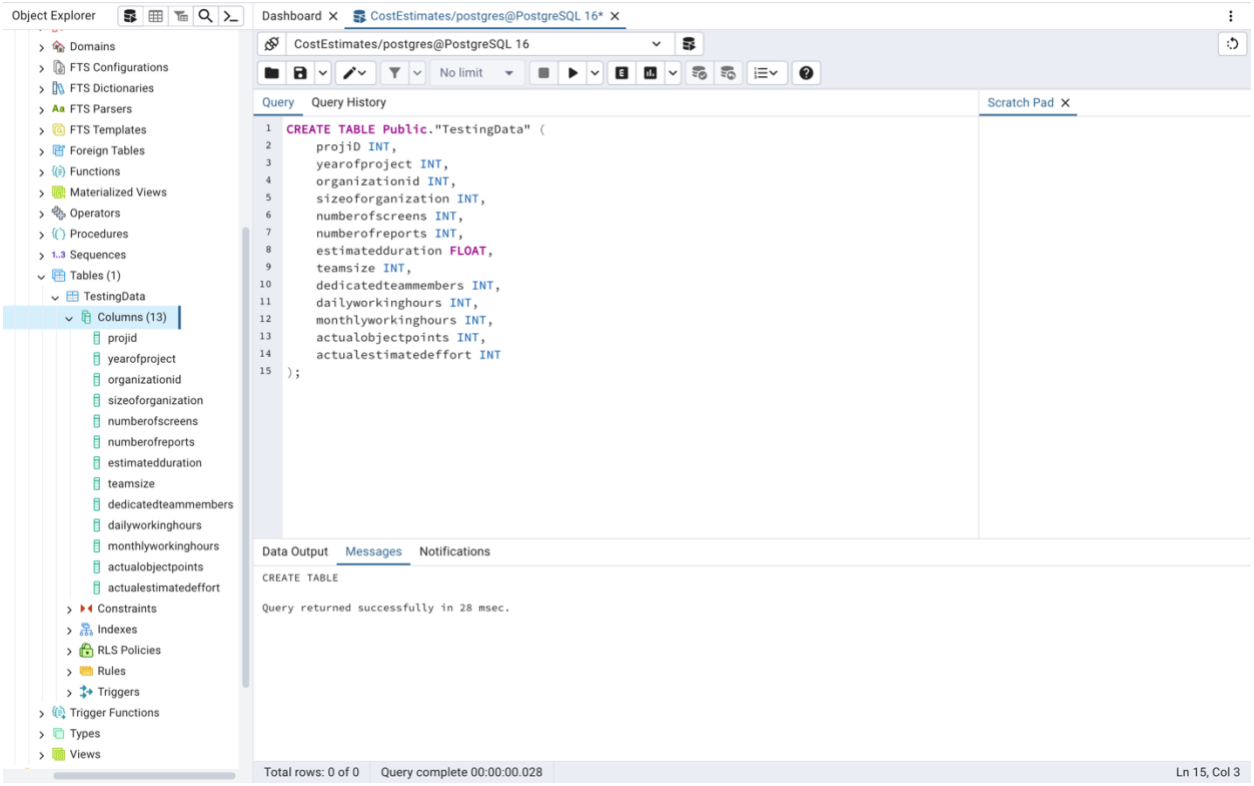
2. Navigate to Query Tool in CostEstimates/Schemas/Tables.



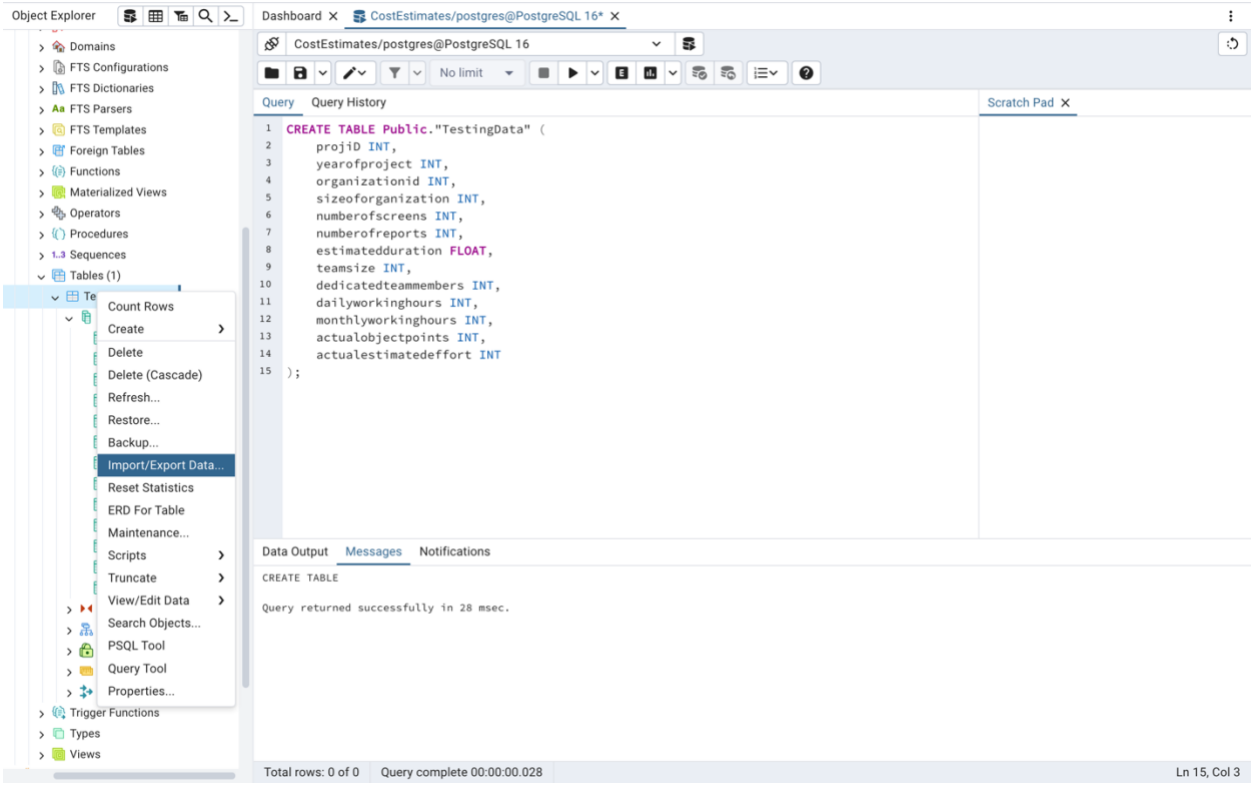
3. Paste the contents of TestingData.sql file and hit run.



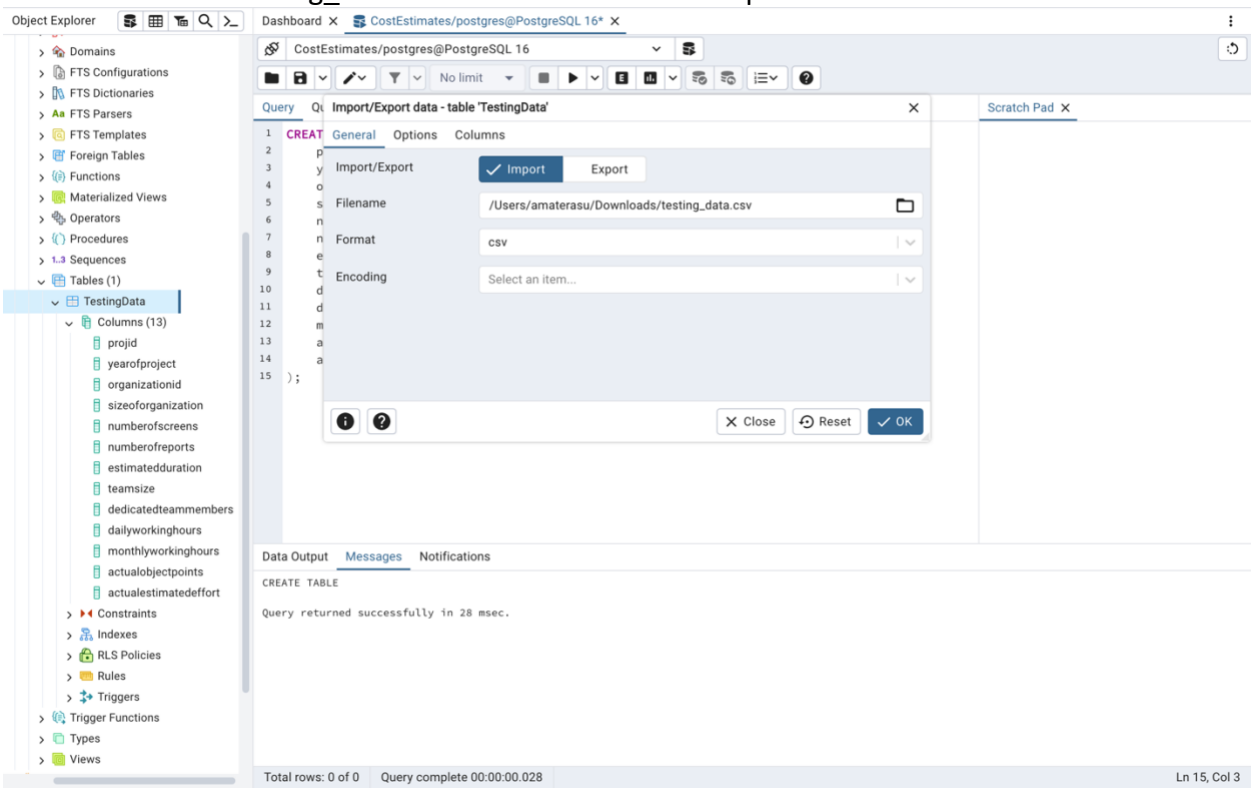
4. Refresh the TestingData table to ensure all the columns are created.



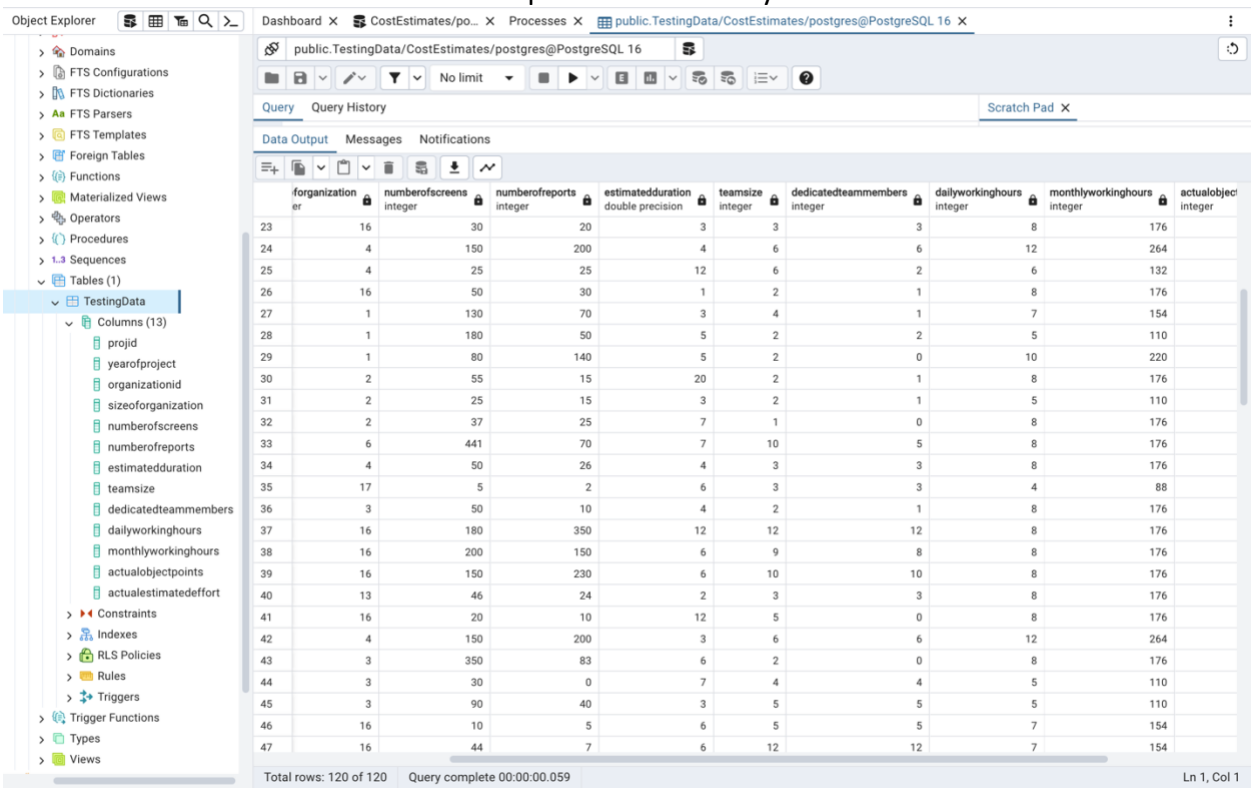
5. Select Import/Export Data from TestingData



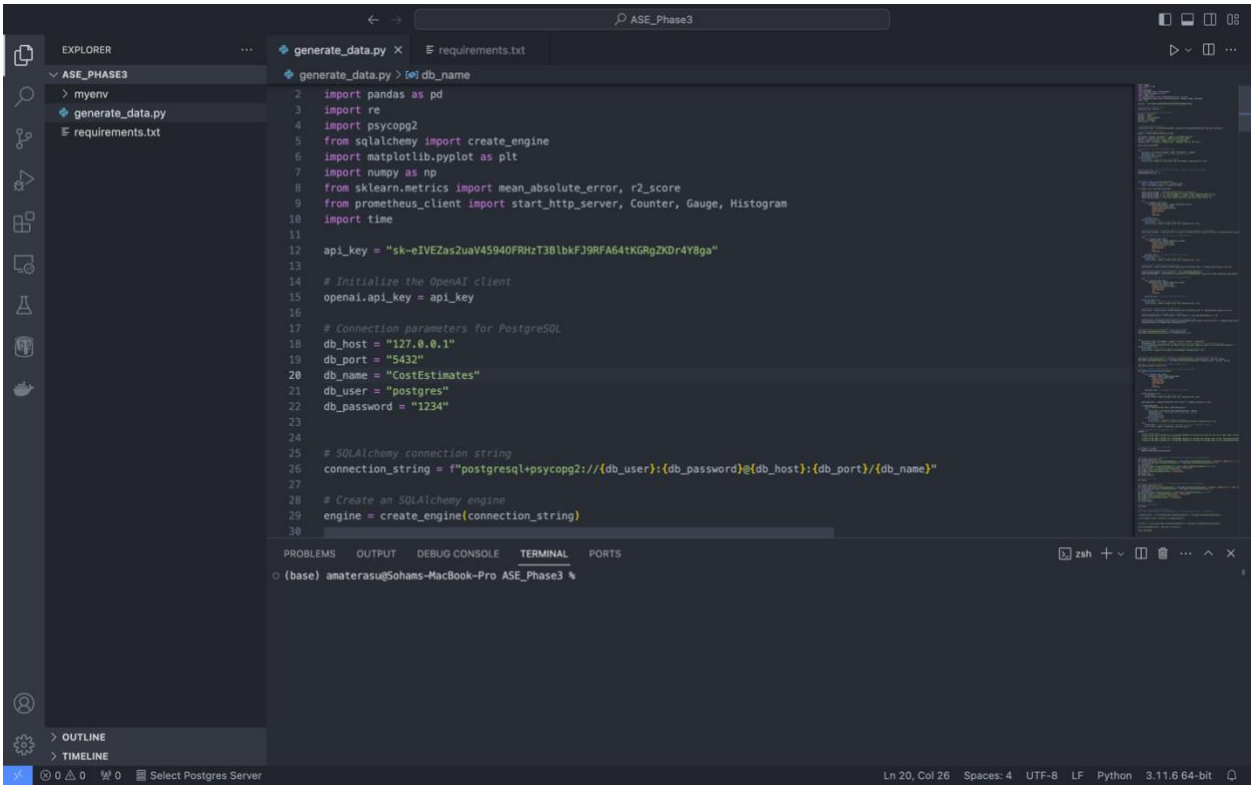
6. Select the testing_data.csv file and ensure that Import and csv Format is selected.



7. Ensure that the data has been imported successfully.



8. Open the folder in your favorite code editor and replace the api_key variable with your OpenAI API Key, db_host to your DB Host, db_port to your DB Port, db_user to your DB user, and db_password to your DB Password.



9. In the terminal execute “python3 -m venv myenv” for creating a virtual environment and then “source myenv/bin/activate” to select the virtual environment and then “pip install -r requirements.txt” to install the necessary requirements.

10. Make the following changes to the prometheus.yml file to listen to the port 8000 “scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "openai-application"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:8000"]”

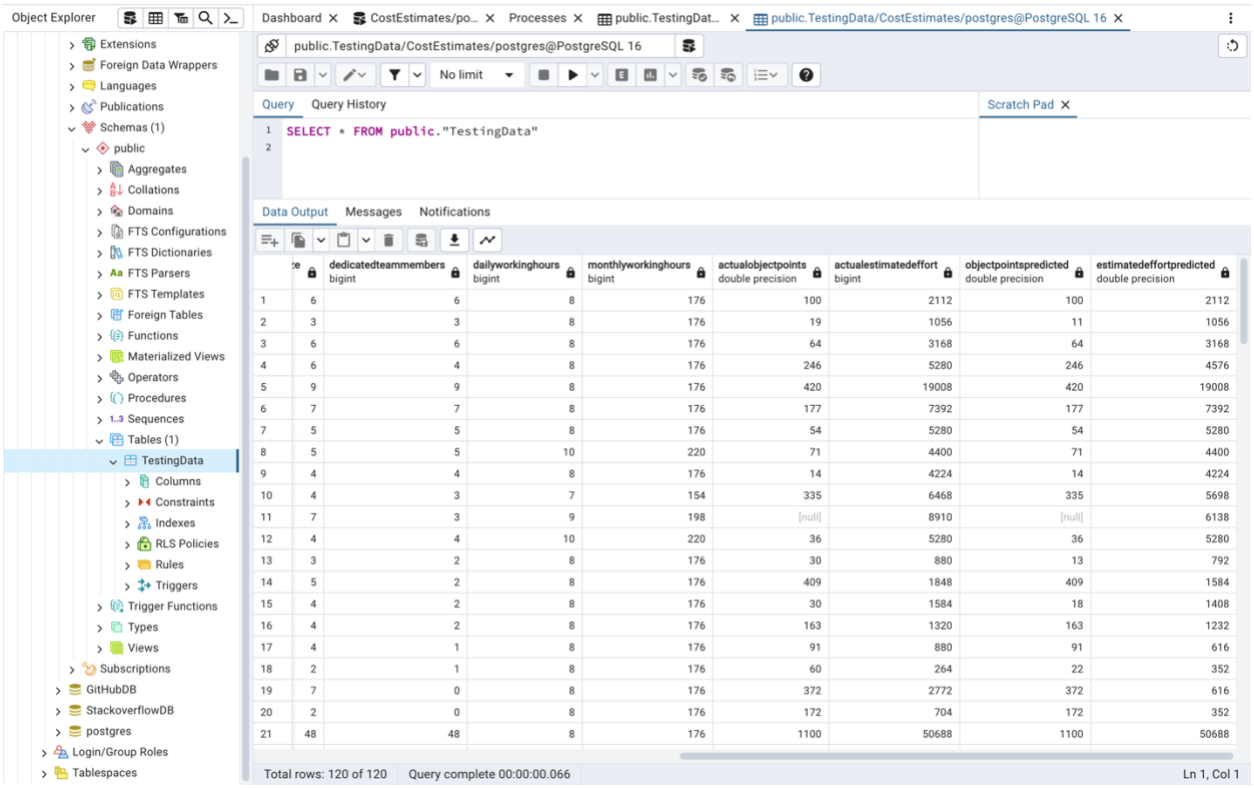
11. Start your Prometheus Client by typing “./prometheus”.

12. Run the application by typing “python3 generate_data.py”.

13. Navigate to “localhost:8000/metrics”.

14. Refresh your Tables in pgAdmin to view the updated predicted columns in the TestingData table.

15. Ensure that the “objectpointspredicted” and “estimatedeffortpredicted” columns are added to the database.



16. Now you should be able to view the output of the program.

Functions and Features

1. **OpenAI API Initialization**: Initializes the OpenAI client with a given API key, enabling access to OpenAI's GPT models.
2. **PostgreSQL Database Connection**: Establishes a connection to a PostgreSQL database using SQLAlchemy, facilitating data retrieval and storage.
3. **Prometheus Metrics Tracking**: Implements Prometheus metrics like counters and histograms to track database queries, API calls, and response times, useful for monitoring and performance analysis.
4. **Data Retrieval from PostgreSQL**: Fetches test data from a PostgreSQL database, handling potential errors and incrementing a query counter for monitoring.
5. **Numeric Value Extraction from Text**: Utilizes a regular expression to extract numeric values from text, aiding in data processing and analysis.
6. **API Call for Estimating Object Points**: Makes an API call to OpenAI's model to calculate object points based on specific criteria, contributing to the project's estimation capabilities.
7. **Effort Estimation Process**: Calculates estimated effort through a series of API calls and internal computations, providing insights into project management and planning.
8. **Data Storage in PostgreSQL**: Saves the calculated estimates back into the PostgreSQL database, ensuring data persistence and tracking.
9. **Handling NaN and Infinite Values**: Addresses data quality issues by replacing NaN and infinite values in specific columns, crucial for maintaining data integrity.
10. **Dynamic SQL Query Generation and Execution**: Generates and executes SQL queries based on prompts, showcasing the integration of natural language processing with database management.
11. **Data Visualization**: Plots relationships between actual and predicted values of estimated effort and object points, offering visual insights into the model's performance.
12. **Performance Metrics Calculation**: Computes R^2 scores for object points and estimated effort, providing a quantitative measure of the model's accuracy.
13. **Program Sleep**: Puts the execution on hold, used here for testing or demonstration purposes.
14. **Error Handling**: Implements robust error handling throughout, ensuring the program's stability and providing informative error messages.