**Project Title and Description:**  **Order Management System**

**Web browser Link:** <http://127.0.0.1:5000/>

**Github**: https://github.com/MeenaMohan346/Order-Management-System

**Login:**

Username: admin

Password: admin123

* + This project is a web-based application using flask framework interacting with the SQLAlchemy PostgreSQL server.
  + The goal of the project:

Small business order managing system with the groceries imported from India for the users to order

* + The Technology used:
    1. Added the **API call** to get response for city, state, country responses when a zipcode is entered while adding/editing address field in customers tab.

<http://ZiptasticAPI.com/65802> which outputs

{"country":"US","state":"MO","city":"SPRINGFIELD"}

* + 1. Implemented the **modal** **dialogue** popup for editing the rows populated from the tables using jQuery and bootstrap
    2. Implemented **Select2** library which is a jQuery based replacement for drop down boxes in html to add additional features like attaching images to items and adding themes. Included the compiled JavaScript and CSS files as the reference links.
    3. Web pages are html pages with JS, JQuery, and Select2 from a CDN for the GUI
    4. Python Flask framework using PostgreSQL server (pgAdmin4) for the backend
    5. Docker image, docker compose
    6. Screen data export to excel available
  + This project folder contains the following sub folders
    1. **Db** folder which contains
       - 1. insert\_data.sql file to insert the data into the tables customers, products, orders, and orders\_details
         2. SQL function search\_customers.sql code is attached
    2. **Static** folder has
       - 1. sub folder **js** for JavaScript related files namely update.js and validate.js

**Update.js** main purpose is to work with Edit rows functionalities using modal dialogue. It will open a modal dialogue popup when Edit button is clicked and populating with the existing data from the row and allow for the user to enter updated values and once save changes button is clicked the updated data will be saved to the database.

* + - * 1. Sub folder **images** contains the images
        2. Styles.css file
        3. **templates** folder has html files for login, search, search results, base.html, ordermanagement.html, add\_new\_order, update\_order\_form.
      * ordermanagement.html file has the entire code for this project. This file has html code for the layout of the top row buttons, middle row data population and Delete, Edit and Add row buttons for all the four tables. In addition, it has modal dialogue popup design as well for edit
      * base.html is basic file
    1. **app folder:** It contains app.py file which is the main file, .dockerignore, Docker file and requirements.txt
       - **app.py** is the main file which contains the code for the database server connection, initialize connection, login, search, search results, index, delete, update and add (create) functions with appropriate route decorations for all the four tables data using SQLAlchemy
    2. **docker-compose.yml** file serves as a configuration file for defining and running multi-container Docker applications**.**
    3. **models.py** contains the four class files Customer, Product, Order and OrdersDetails for the four tables customers, products, orders and orders\_details
    4. **routes.py** file is for querying data from all the four tables
    5. **order\_management\_postgres.pgred** has the ERD diagram of the four tables and their relationships

**Installation Instructions:** Steps required to set up the project locally, including installing Python, Flask, Flask-SQLAlchemy, and any other dependencies. This may also cover database setup and configuration.

Following is the list of steps I ran to install

python -m venv venv

venv/Scripts/activate

deactivate

venv/Scripts/activate

pip3 install Flask Flask-Scss Flask-SQLAlchemy

pip list

go to extension and search for sass and install it

pip install psycopg2

pip install python-dotenv

pip install requests

pip install xlsxwriter

Installed PostgreSQL server version 17. Currently using **postgres** database for this project. First time when running app.py, all the four tables will be automatically created using the following script in app.py and will auto populate using insert scripts first time. On the subsequent runs, if any of the tables are missing, the app will check for the missing tables and will recreate them and insert data.

with app.app\_context():

    models.db.create\_all()

* **Usage Instructions:** How to run the Flask application, access its features, and interact with the database through the application's interface.

After login, this website GUI has four buttons (**Customers, Products, Orders and Orders Details**) on the top row to display the data in the bottom table rows from the database.

On clicking each of the buttons, the corresponding table data will display in the bottom table rows.

Each row has Delete and Edit buttons at the right side.

**Delete** button once clicked will display confirm popup to confirm the user before deleting the row. Delete function in app.py is used for this purpose

**Edit** button will open a modal dialogue populating with the row data in the input html boxes and allow the user to enter values to update. Html tags are configured with the proper validations for text, email, phone number, integer values and float input values. If any of the user input values does not satisfy the formats, the screen will display the error message for the user to reenter the correct format. Once click the save button, the data will be sent to the server using update function in app.py

**Adding** new entries to the tables are at the very bottom of the screen. Like edit operations, add operations also undergo validations for the user input values and finally sent to the server using create function in app.py

**Order\_details button** ispopulating the data from the orders that are created.

So there is no edit, delete, add buttons available for this table as this table data is read only. Order can have multiple products so each order will display all the products in a dropdown list with images for the admin to select and save

**Other functionalities:**

**Flask**: Flask's flashing system provides a mechanism to record messages at the end of a request and display them on the subsequent request, typically after a redirect. This was added to display the error/success message at the top of the screen after each request.

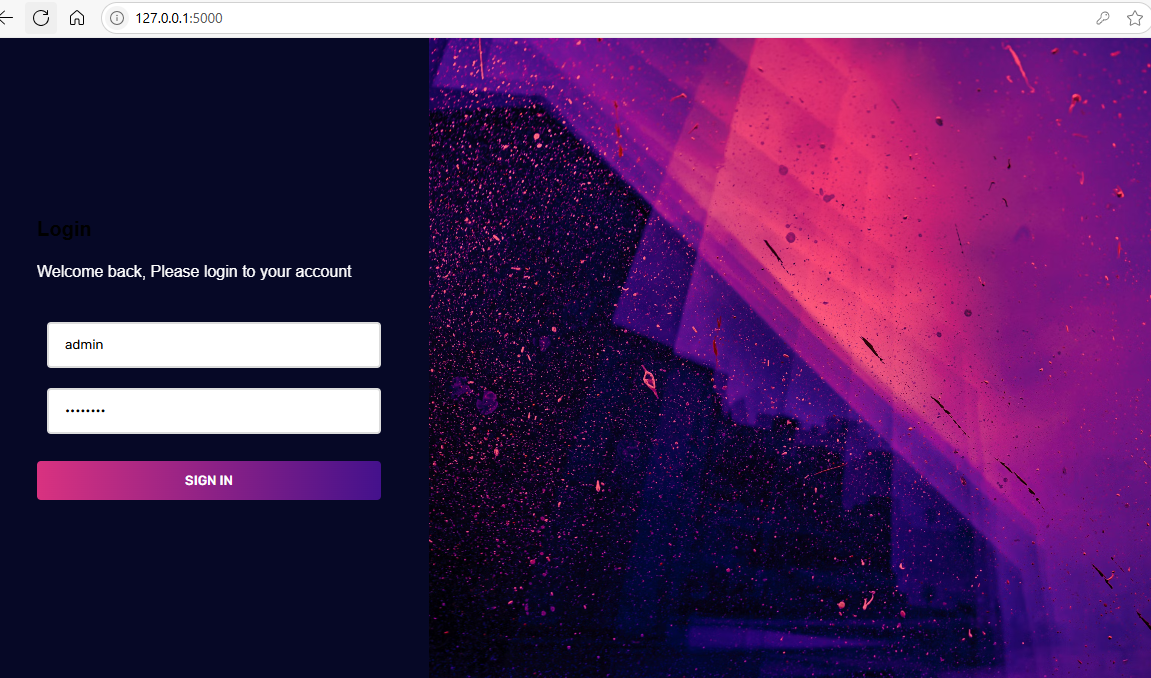
**Foreign key** handledcarefullyupon delete, edit, add request

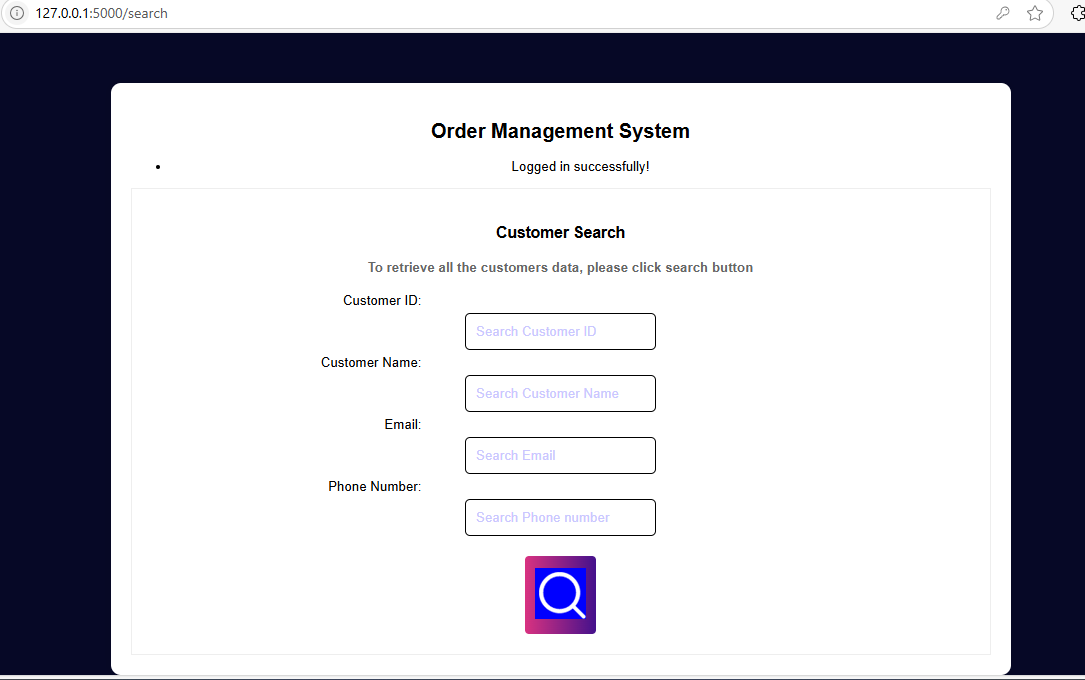
**Delete operation** follows cascade delete set null

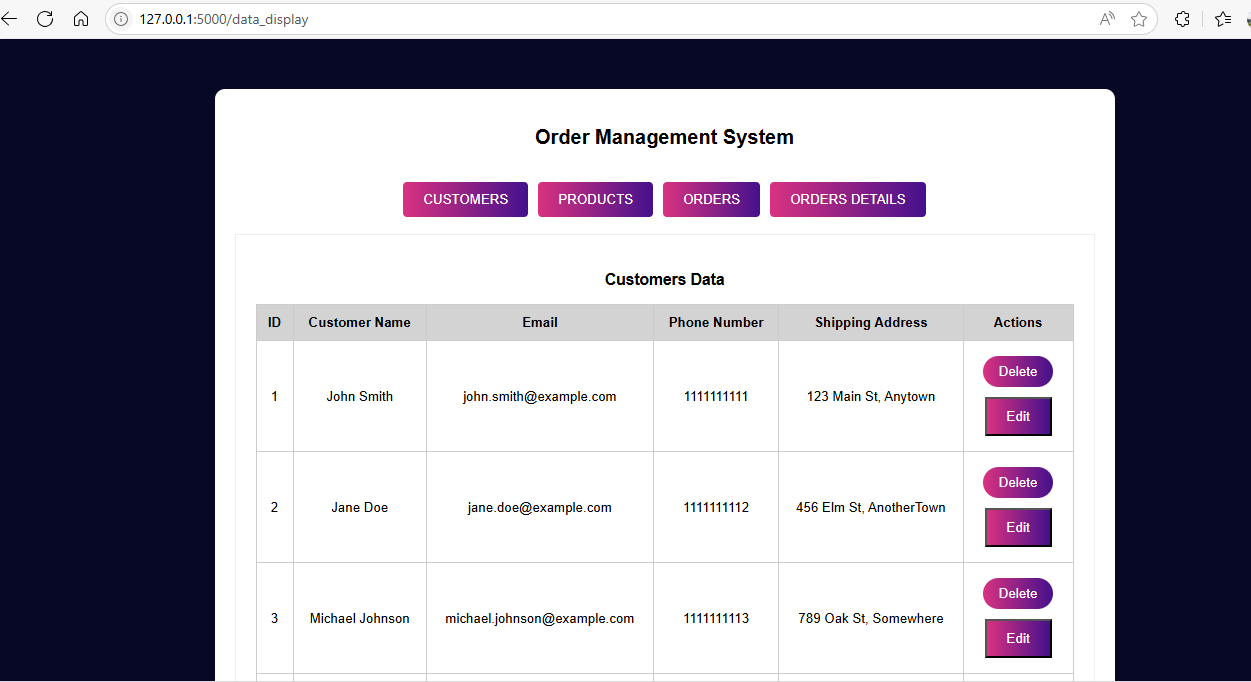
**Edit operation**: It checks for the primary key columns if they really exist in the primary table when the user enters values. If not, it throws error message to display in **flash**.

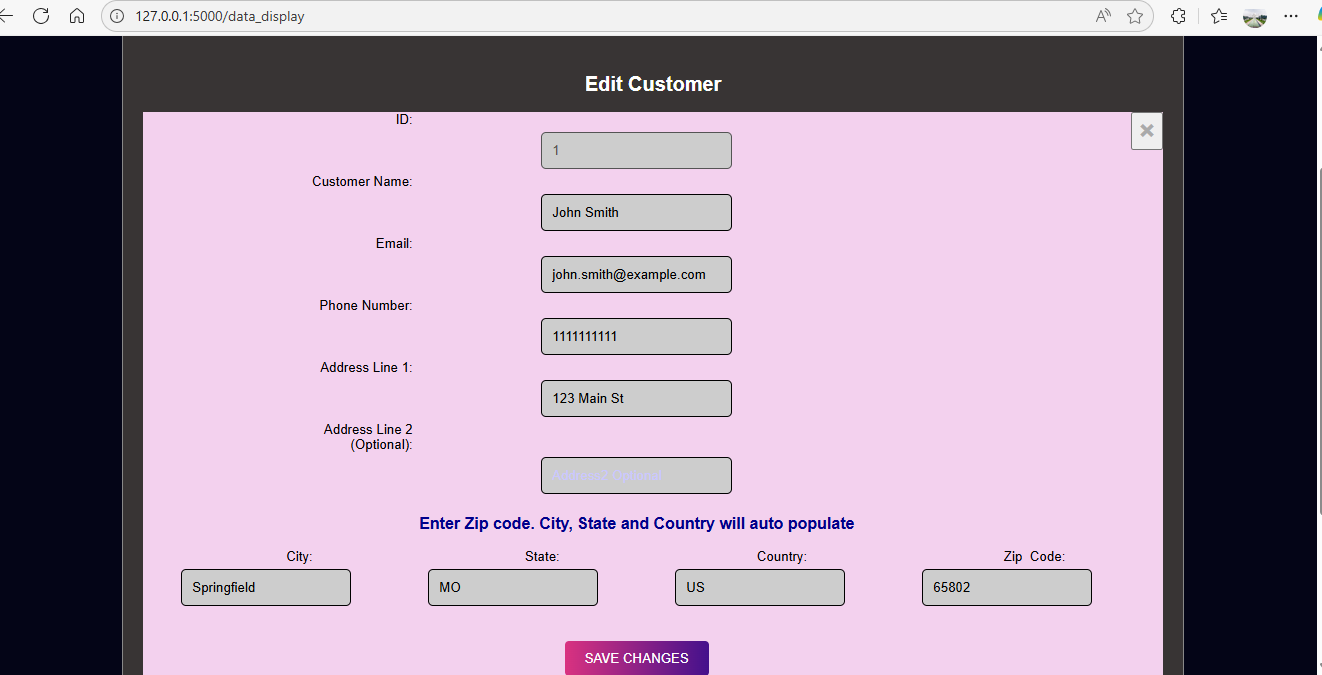
Similarly **add operation** follows the same.

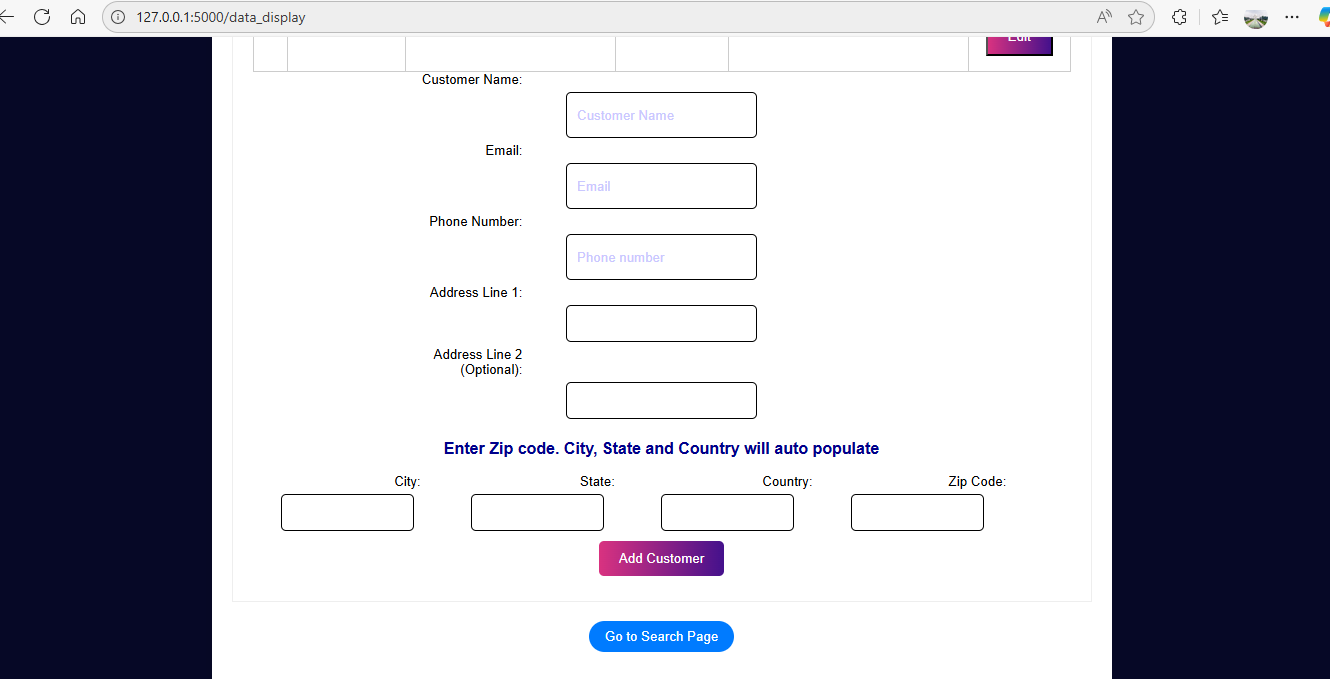
Screen shots shared below

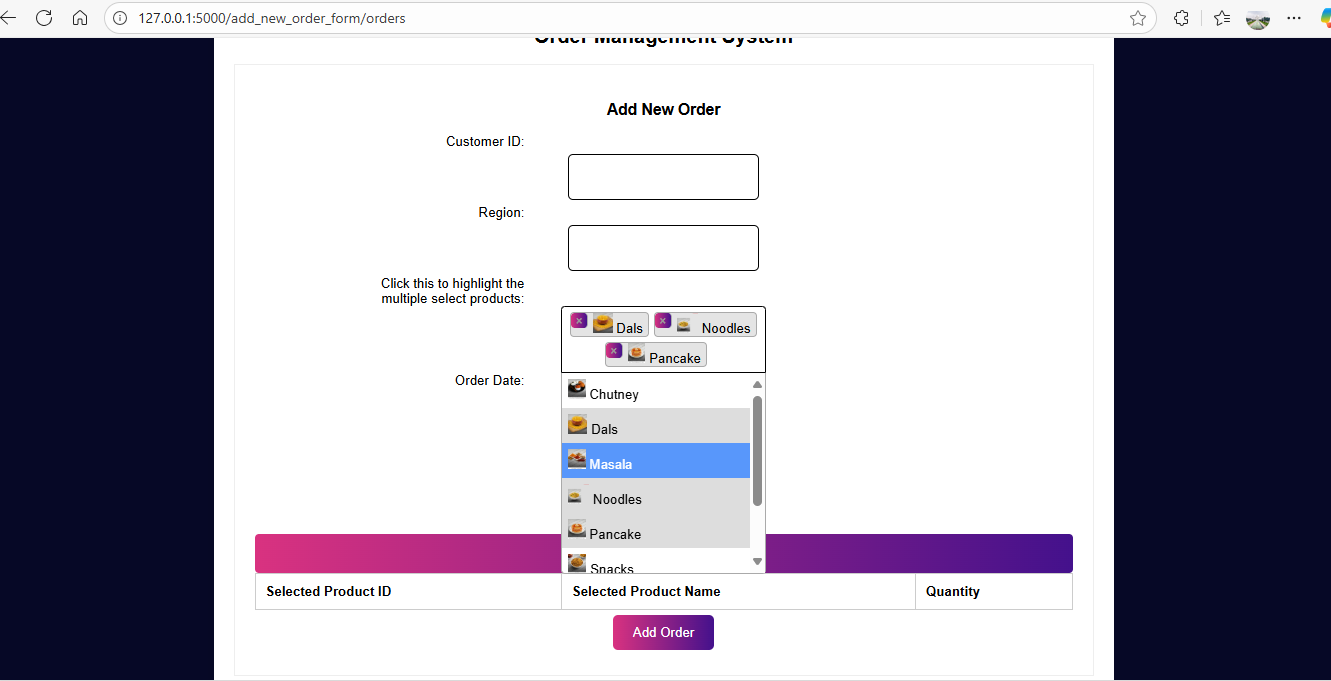


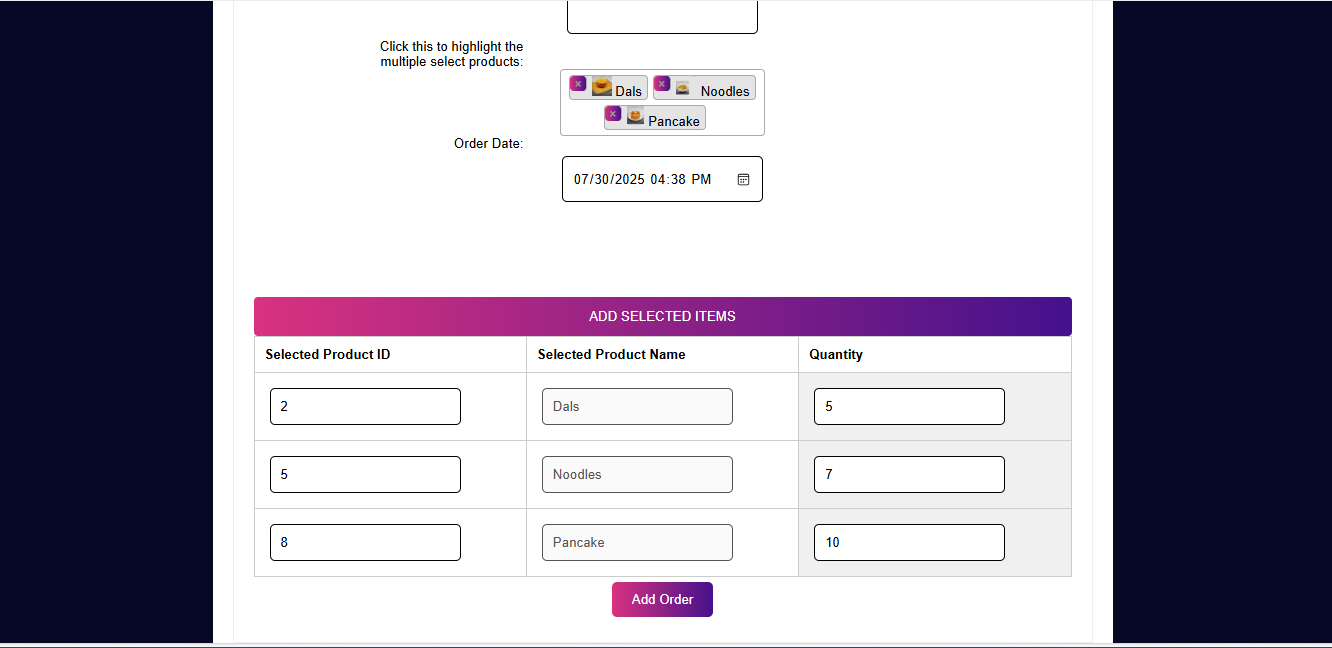




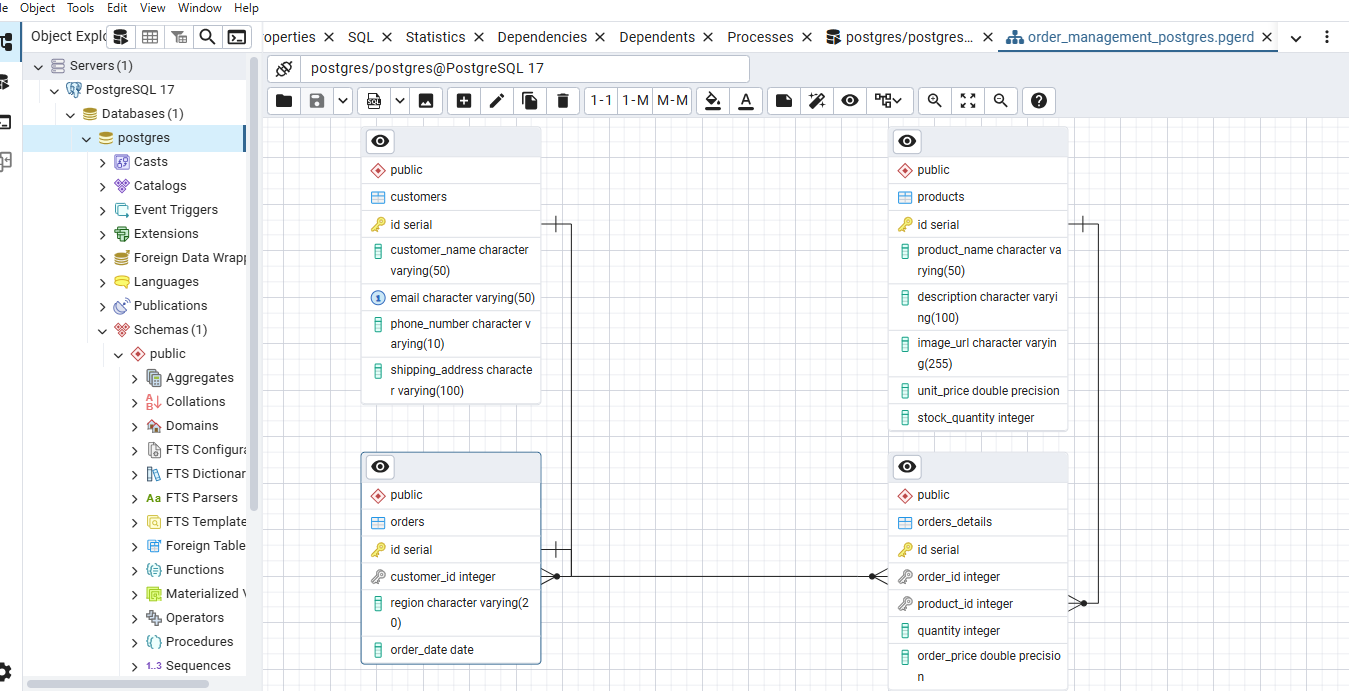








* **Database Schema (Optional but helpful):** A high-level description or diagram of the database models defined using SQLAlchemy.



Primary key – Foreign key relationships:

**Foreign Key**

* + 1. Orders.customer\_id column has foreign key relationship with customers.id column
    2. Orders.product\_id column has foreign key relationship with products.id column
    3. Orders\_details.order\_id column has foreign key relationship with orders.id column
    4. Orders\_details.product\_id column has foreign key relationship with products.id column

**Primary key**

Customers, products, orders, orders\_details tables have their id column as the primary keys.

* Retrospective answering of the following questions:
  1. How did the project's design evolve over time?

It started with just one table to display the data in the GUI with delete, edit, add operations, and finally extended to 3 more tables following the same logic. Added the html validations in the user input controls, Select2 feature to the select options with image in the orders tab, adding the selected products to the below table for the user to enter the quantities for each product item.

Did you choose to use an ORM or raw SQL? Why? ORM SQLAlchemy

* 1. What future improvements are in store, if any?

Password reset can be added.

Adding payments, creating the invoices and shipping order status display and so many can be enhanced.

* 1. Challenges faced and solutions found:

Spent time to find out the best option to add themes and attaching the images to the product items dropdown list in html and figured out and read the full manuals in Select2. (https://select2.org/)