Shorthand Representation

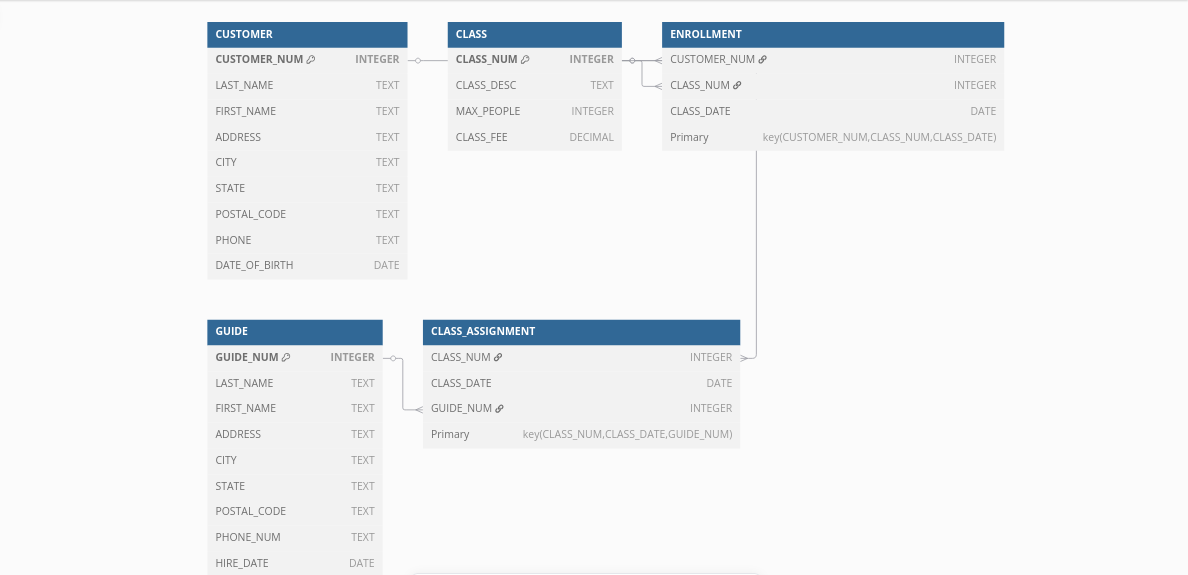
a) CUSTOMER (Customer\_Num, Last\_Name, First\_Name, Address, City, State, Postal\_Code, Phone, Date\_of\_Birth)

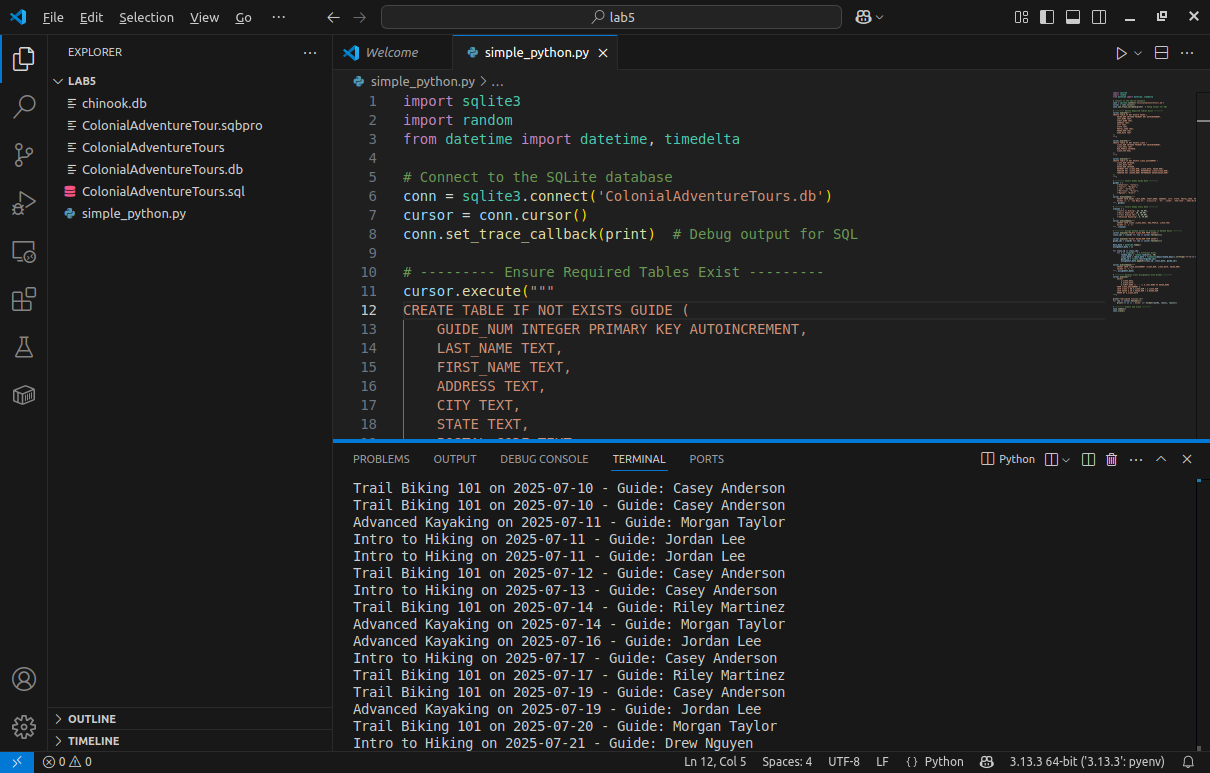
b) CLASS (Class\_Num, Class\_Desc, Max\_People, Class\_Fee)

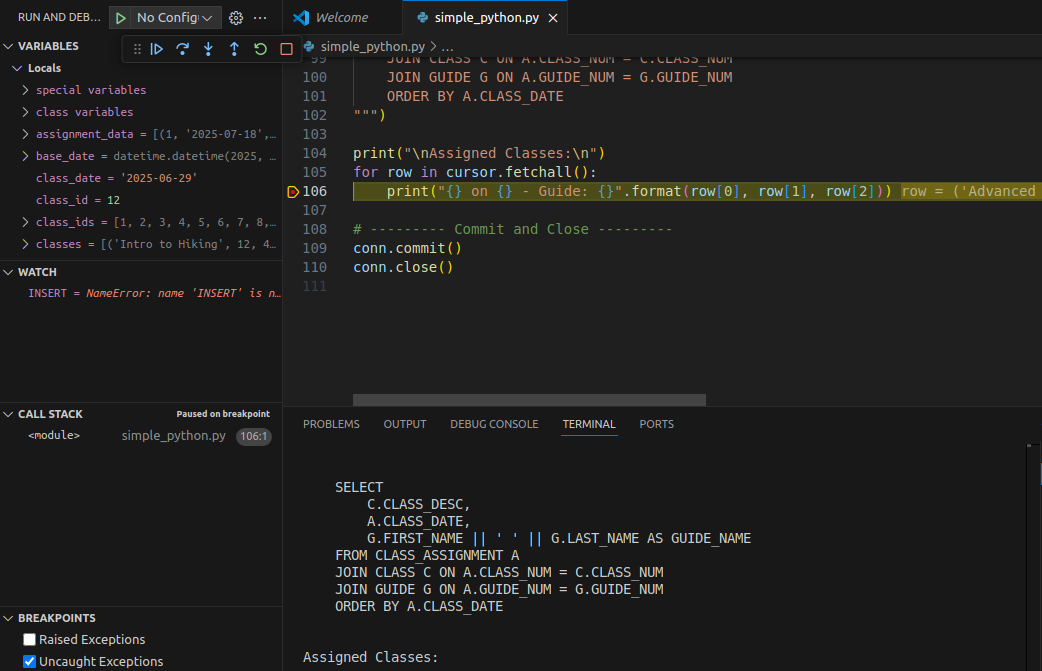
c) ENROLLMENT (Customer\_Num, Class\_Num, Class\_Date), CUSTOMER (Customer\_Num, Last\_Name, First\_Name), CLASS (Class\_Num, Class\_Desc)

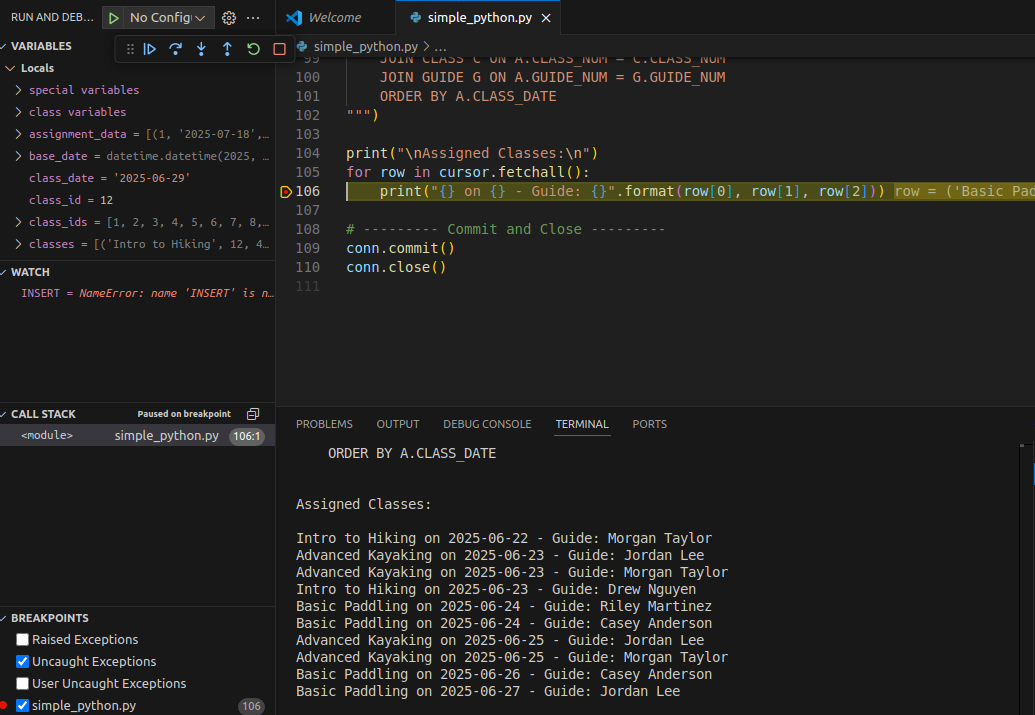
d) ENROLLMENT (Class\_Date, Class\_Num, Customer\_Num), CLASS (Class\_Desc), CUSTOMER (Last\_Name, First\_Name)

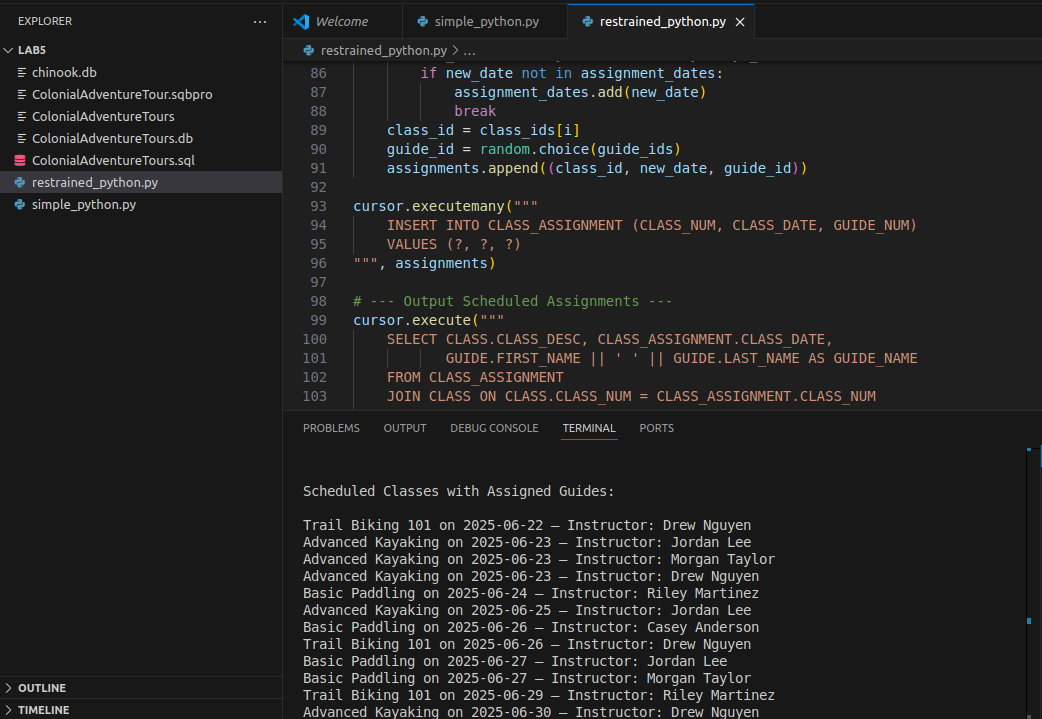
Diagram

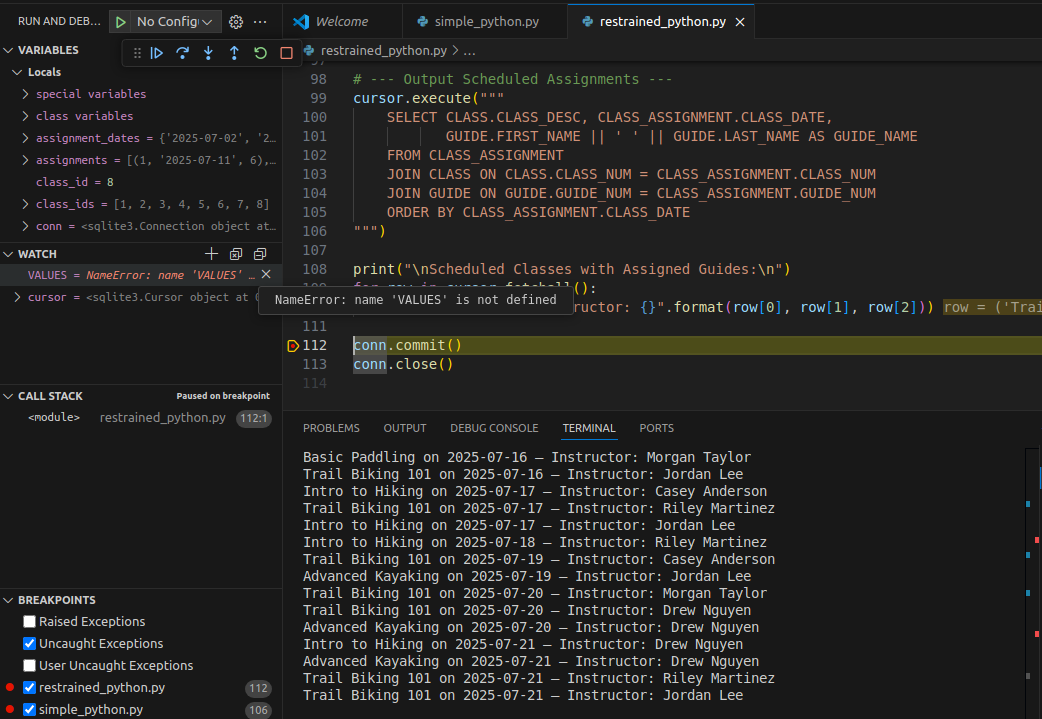
Debugging and Sample Run

Simple Python Program that Builds the Table, Seeds them with Data and a Random Guide

Debugging

Debugging After the generation of randomized names and schedules

Python Program that restraints to one class per day

Debugging of Restraints program

Discussion:

The Python program helps Colonial Adventure Tours set up outdoor adventure classes in a simple and automatic way. It creates all the needed tables in the database, including those for guides, classes, and class assignments. Then it checks if these tables are empty and fills them with sample data like fake guides and classes if needed.

It also makes sure that only one class is scheduled per day. It picks random dates and guides, so each class has one instructor and happens on its own day. This follows the rule that participants don’t know the guide until the day of the class. After setting the schedule, the program shows a list of upcoming classes and which guide is assigned to each one.

While testing, I saw all the steps clearly printed out. I used a debug feature that showed every SQL command being used. If something went wrong, like a missing table, the program would stop and help me find the mistake. The program ran smoothly and did what the class system needs.