Mahendra Leonard

CIS 344

Prof. Yanilda Peralta Ramos

Restaurant Reservations Database Report

This report outlines the structure and functionality of a restaurant reservation system, detailing the SQL code used to create and manage the system's database. We will discuss the creation of tables, insertion of data, and stored procedures to illustrate how the system operates, and how the python files are connected to SQL.

Database Creation and Tables

To begin, a new database named restaurant_reservations is created and selected for use. Then a Customers table was created and populated with three tuples. The Customers table stores information about the customers. It has three columns: customerId, customerName, and contactInfo.

- customerId: A unique identifier for each customer, automatically incremented.
- customerName: The name of the customer, a mandatory field.
- contactInfo: Contact information, such as an email address.

Next, The Reservations table stores reservation details and is linked to the Customers table via customerId.Data is then inserted into the Reservations table. Similarly, The DiningPreferences table records customers' dining preferences, also linked to the Customers table. This table was also populated by three tuples.

Stored Procedures

Stored procedures are a set of SQL statements that are saved to be reused multiple times. They encapsulate complex queries and transactions, enabling more efficient and cleaner database management. In the restaurant_reservations database, three stored procedures are defined: findReservations, addSpecialRequest, and addReservation.

-The findReservations procedure retrieves all reservations for a specific customer based on their customerId.

-The addSpecialRequest procedure updates the specialRequests field for a specific reservation based on its reservationId.

-The addReservation procedure allows the addition of a reservation for a customer. It checks if the customer exists; if not, it creates a new customer and then adds the reservation.

The stored procedures in the restaurant_reservations database enable efficient management of customer reservations, special requests, and new customer entries. By encapsulating complex logic and operations, these procedures help maintain the integrity and functionality of the database system.

Connecting Python and SQL

To connect Python and SQL we need to enter “pip install mysql-connector-python” via the command prompt. Next, we enter the resturantDatabase.py file and change the parameter values to match the MySQL server information.

We add code to resturantDatabase.py to achieve the results below:

- getAllReservations: Retrieves all reservations from the reservations table.
- addReservation: Inserts a new reservation into the reservations table.
- addCustomer: Inserts a new customer into the customers table.

- getCustomerPreferences: Retrieves dining preferences for a specific customer from the diningPreferences table.

- addSpecialRequest: Updates the special_requests field for an existing reservation.

- findReservations: Retrieves all reservations for a specific customer.

- deleteReservation: Deletes a reservation from the reservations table.

- searchPreferences: Searches for dining preferences based on dietary restrictions.

To run Application:

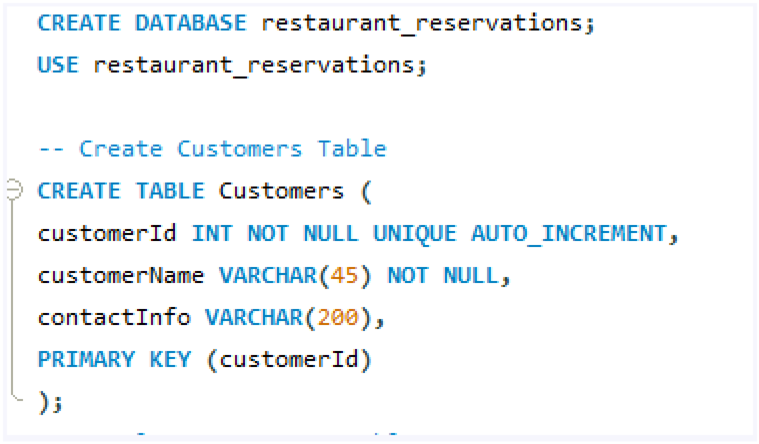
1. Set up the Database: Run the SQL script to create the necessary tables and initial data.
2. Run the Web Server: Execute the 'restaurantDatabase.py' and 'RestaurantServer.py' script to start the web server.
3. Access the Web Interface: Open a browser and go to localhost:8000

To view code and SQL script visit the GitHub repository link below:

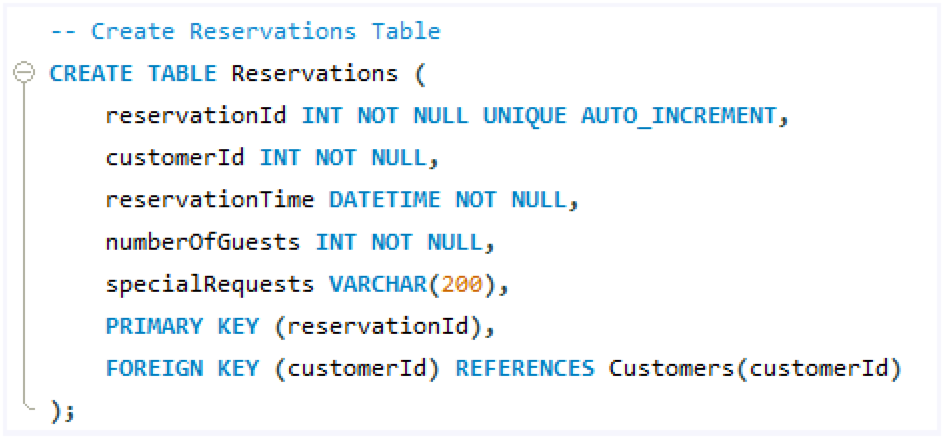
https://github.com/MLeonard0/CIS-344.git

APPENDIX

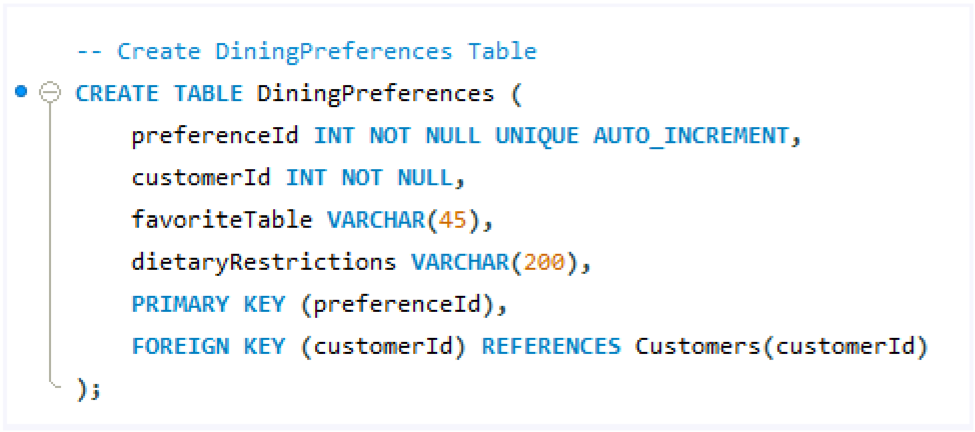
Screenshot 1: Database and Table Creation



```sql
CREATE DATABASE restaurant_reservations;
USE restaurant_reservations;

-- Create Customers Table
CREATE TABLE Customers (
customerId INT NOT NULL UNIQUE AUTO_INCREMENT,
customerName VARCHAR(45) NOT NULL,
contactInfo VARCHAR(200),
PRIMARY KEY (customerId)
);
```

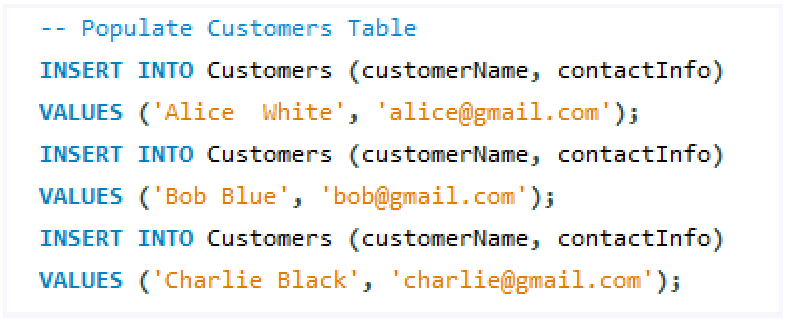


```sql
-- Create Reservations Table
CREATE TABLE Reservations (
    reservationId INT NOT NULL UNIQUE AUTO_INCREMENT,
    customerId INT NOT NULL,
    reservationTime DATETIME NOT NULL,
    numberOfGuests INT NOT NULL,
    specialRequests VARCHAR(200),
    PRIMARY KEY (reservationId),
    FOREIGN KEY (customerId) REFERENCES Customers(customerId)
);
```

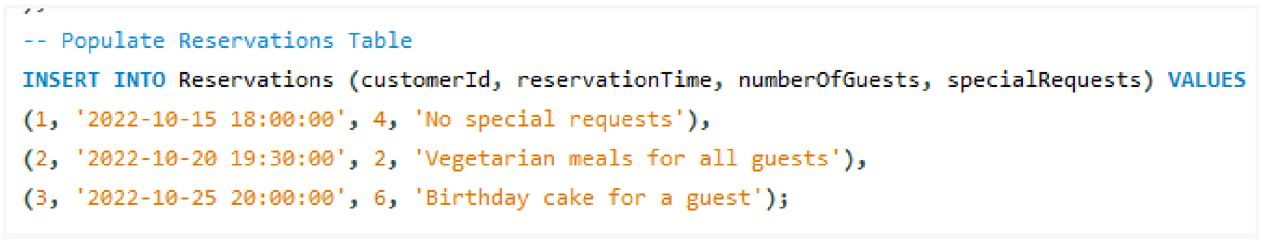


```sql
-- Create DiningPreferences Table
CREATE TABLE DiningPreferences (
    preferenceId INT NOT NULL UNIQUE AUTO_INCREMENT,
    customerId INT NOT NULL,
    favoriteTable VARCHAR(45),
    dietaryRestrictions VARCHAR(200),
    PRIMARY KEY (preferenceId),
    FOREIGN KEY (customerId) REFERENCES Customers(customerId)
);
```

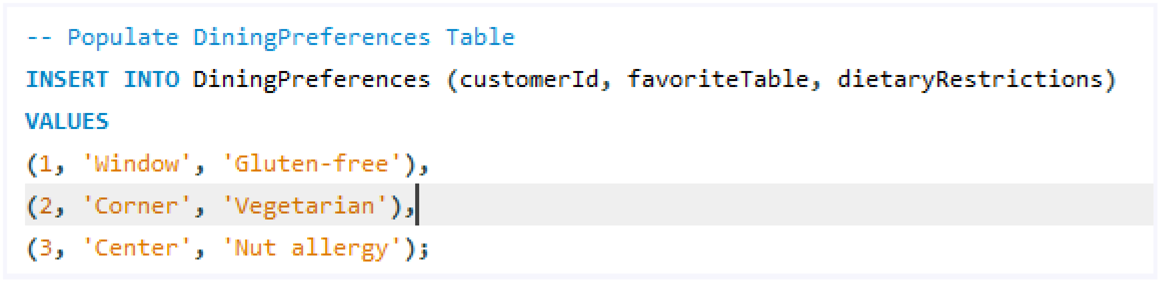
Screenshot 2: Data Insertion



```sql
-- Populate Customers Table
INSERT INTO Customers (customerName, contactInfo)
VALUES ('Alice  White', 'alice@gmail.com');
INSERT INTO Customers (customerName, contactInfo)
VALUES ('Bob Blue', 'bob@gmail.com');
INSERT INTO Customers (customerName, contactInfo)
VALUES ('Charlie Black', 'charlie@gmail.com');
```

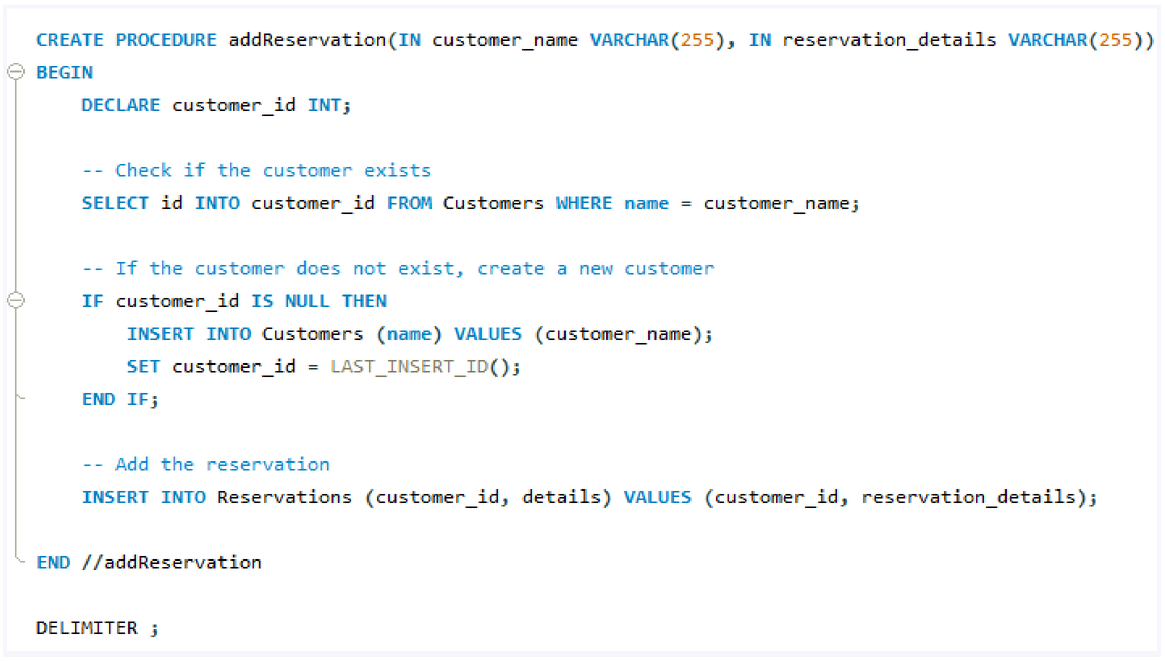


```sql
-- Populate Reservations Table
INSERT INTO Reservations (customerId, reservationTime, numberOfGuests, specialRequests) VALUES
(1, '2022-10-15 18:00:00', 4, 'No special requests'),
(2, '2022-10-20 19:30:00', 2, 'Vegetarian meals for all guests'),
(3, '2022-10-25 20:00:00', 6, 'Birthday cake for a guest');
```



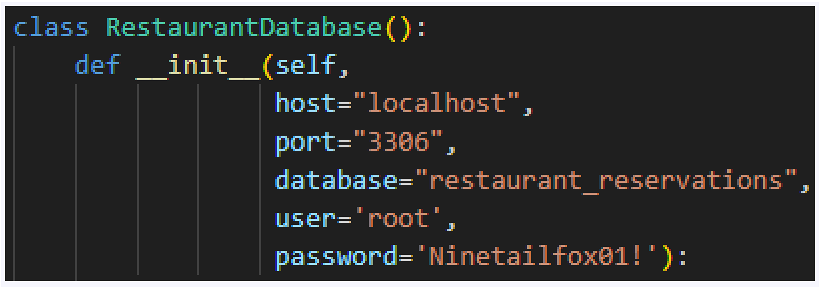
```sql
-- Populate DiningPreferences Table
INSERT INTO DiningPreferences (customerId, favoriteTable, dietaryRestrictions)
VALUES
(1, 'Window', 'Gluten-free'),
(2, 'Corner', 'Vegetarian'),
(3, 'Center', 'Nut allergy');
```

Screenshot 3: Stored Procedures



```sql
CREATE PROCEDURE addReservation(IN customer_name VARCHAR(255), IN reservation_details VARCHAR(255))
BEGIN
    DECLARE customer_id INT;

    -- Check if the customer exists
    SELECT id INTO customer_id FROM Customers WHERE name = customer_name;

    -- If the customer does not exist, create a new customer
    IF customer_id IS NULL THEN
        INSERT INTO Customers (name) VALUES (customer_name);
        SET customer_id = LAST_INSERT_ID();
    END IF;

    -- Add the reservation
    INSERT INTO Reservations (customer_id, details) VALUES (customer_id, reservation_details);

END //addReservation

DELIMITER ;
```

ScreenShot 4: Connecting Python and SQL



```python
class RestaurantDatabase():
    def __init__(self,
                host="localhost",
                port="3306",
                database="restaurant_reservations",
                user='root',
                password='Ninetailfox01!'):
```