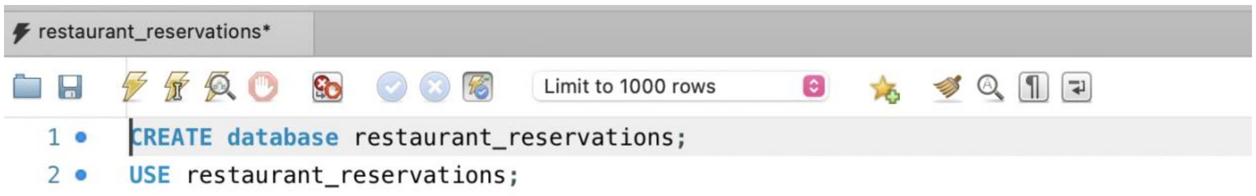
Khadija Abdallah

CIS 344

Prof. Yanilda Peralta Ramos

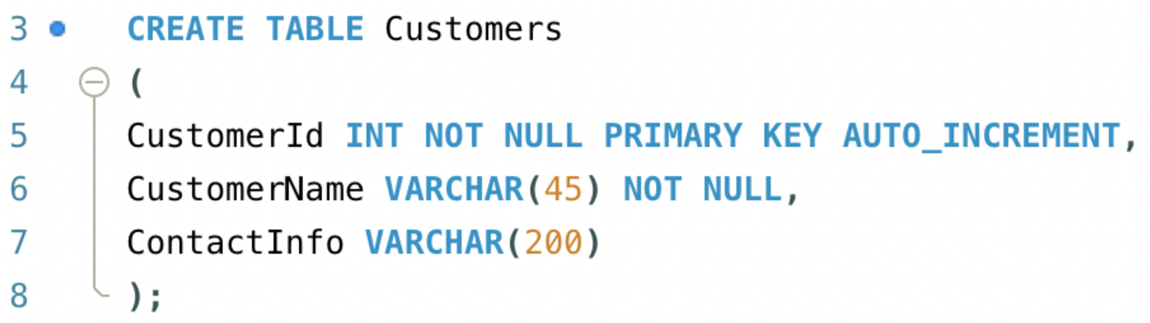
Final Project

The construction of a restaurant reservation system with Python interface connection and MySQL server database is described in detail in this paper. The project includes setting up the database, making tables, drafting stored procedures, adding initial data to the tables, and putting Python methods to use in database interaction into practice. First, I created the **restaurant_reservations** database using the following SQL commands:



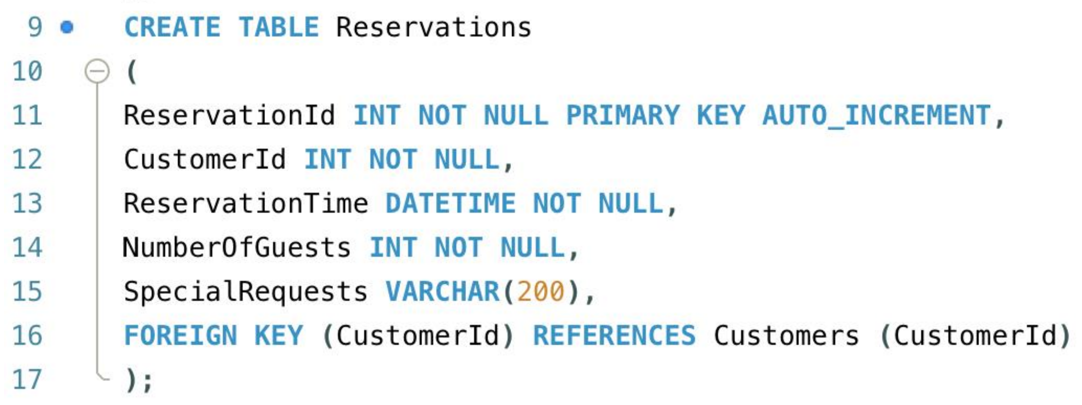
```sql
CREATE database restaurant_reservations;
USE restaurant_reservations;
```

Next, I created the **Customers** table to store customer information:



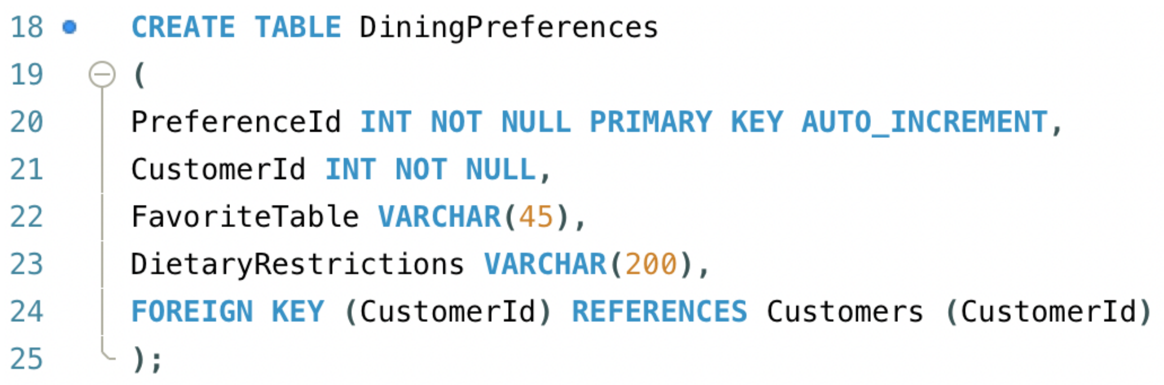
```sql
CREATE TABLE Customers
(
CustomerId INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
CustomerName VARCHAR(45) NOT NULL,
ContactInfo VARCHAR(200)
);
```

Then **Reservations**table was then created to store reservation details:



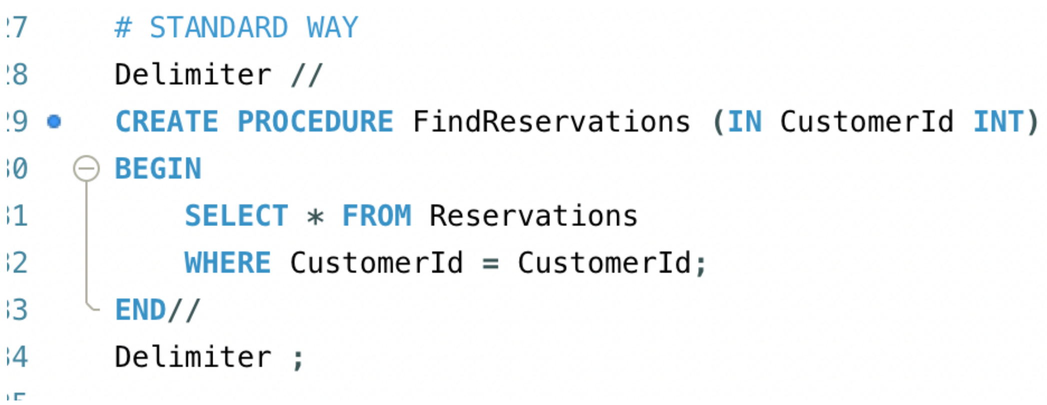
```sql
CREATE TABLE Reservations
(
ReservationId INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
CustomerId INT NOT NULL,
ReservationTime DATETIME NOT NULL,
NumberOfGuests INT NOT NULL,
SpecialRequests VARCHAR(200),
FOREIGN KEY (CustomerId) REFERENCES Customers (CustomerId)
);
```

Finally, I created the **DiningPreferences** table to store customers' dining preferences:



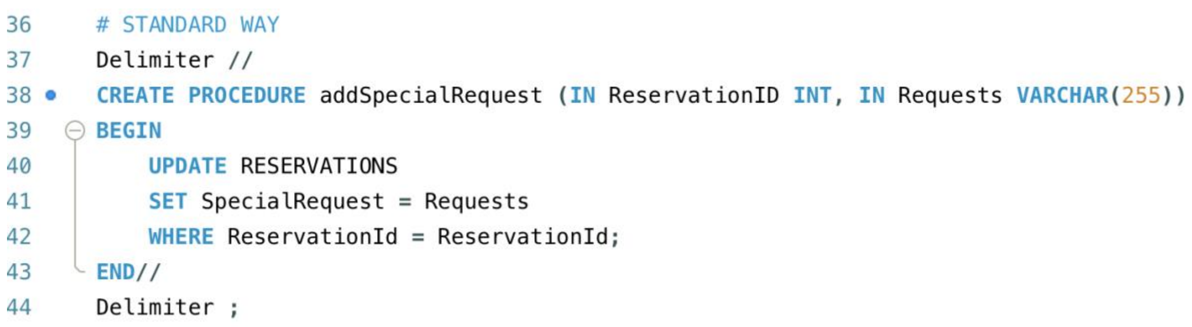
```
CREATE TABLE DiningPreferences
(
PreferenceId INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
CustomerId INT NOT NULL,
FavoriteTable VARCHAR(45),
DietaryRestrictions VARCHAR(200),
FOREIGN KEY (CustomerId) REFERENCES Customers (CustomerId)
);
```

The next step on the project was to create the stored procedure and relationships retrieving the necessary data required. I went ahead to create a stored procedure **findReservations** to retrieve all reservations for a specific customer:



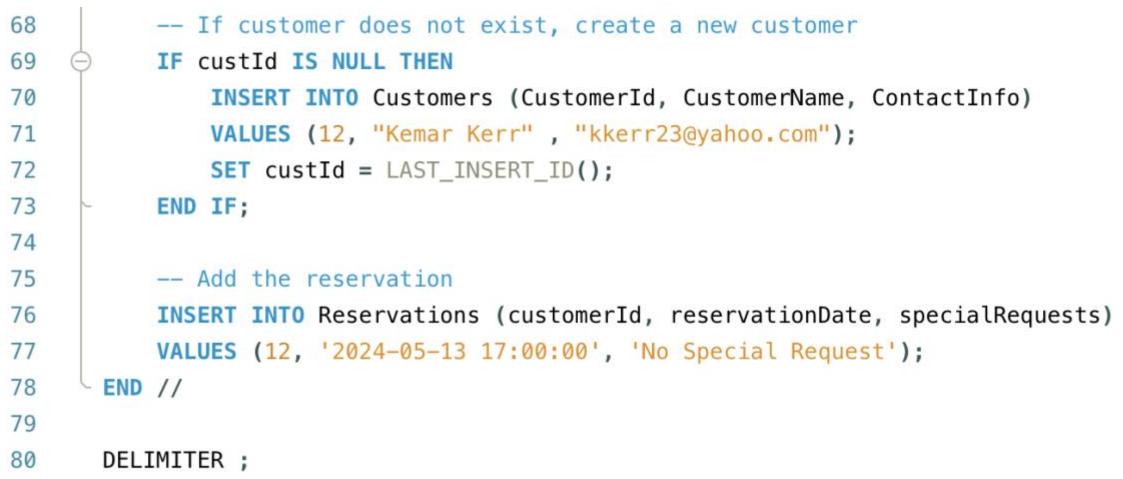
```
# STANDARD WAY
Delimiter //
CREATE PROCEDURE FindReservations (IN CustomerId INT)
BEGIN
    SELECT * FROM Reservations
    WHERE CustomerId = CustomerId;
END//
Delimiter ;
```

Next, I created a stored procedure **addSpecialRequest** to update the **specialRequests** field in the Reservations table:



```
# STANDARD WAY
Delimiter //
CREATE PROCEDURE addSpecialRequest (IN ReservationID INT, IN Requests VARCHAR(255))
BEGIN
    UPDATE RESERVATIONS
    SET SpecialRequest = Requests
    WHERE ReservationId = ReservationId;
END//
Delimiter ;
```

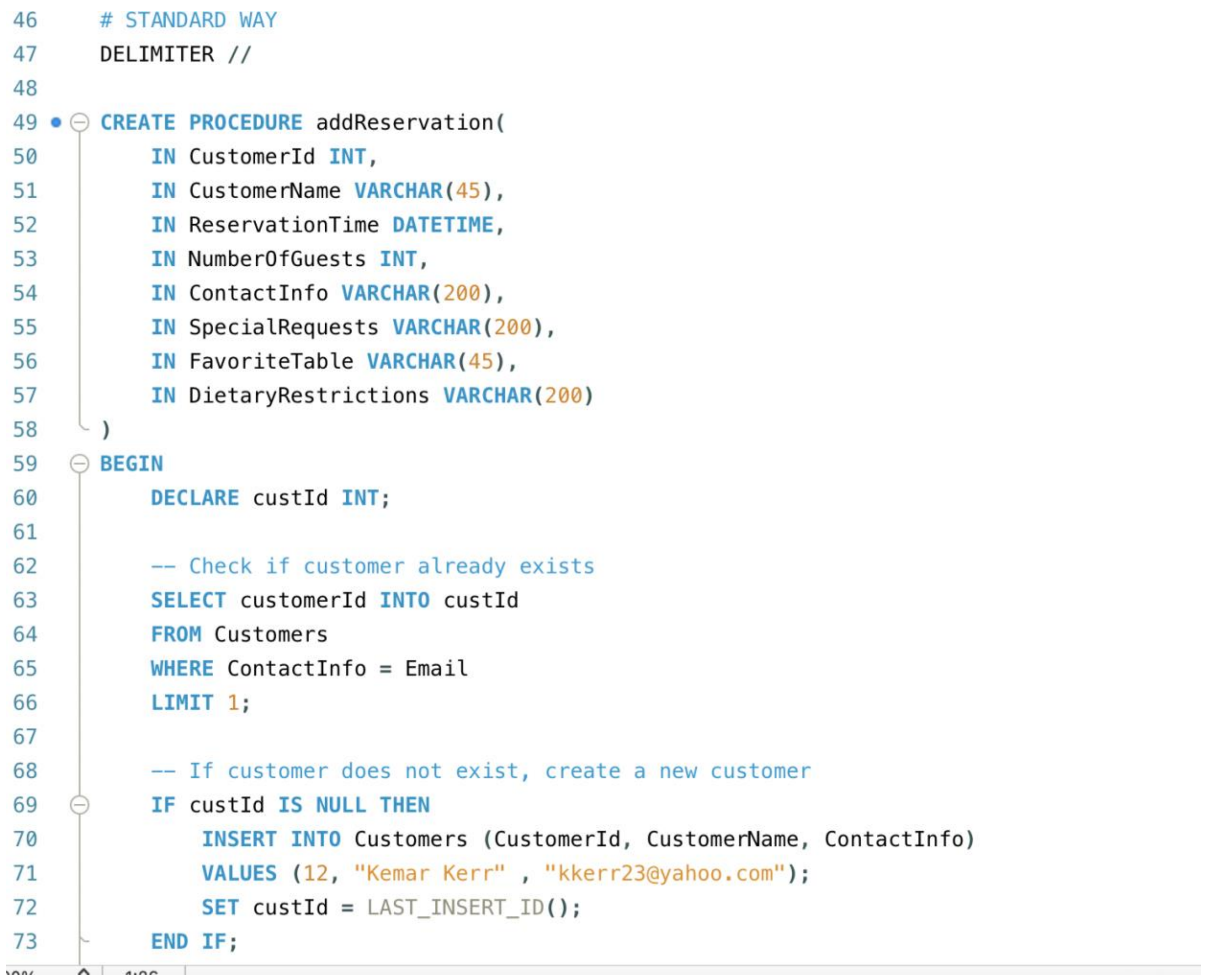
Finally, I created a stored procedure **addReservation** to check or create a customer before adding a reservation:



```sql
    -- If customer does not exist, create a new customer
    IF custId IS NULL THEN
        INSERT INTO Customers (CustomerId, CustomerName, ContactInfo)
        VALUES (12, "Kemar Kerr" , "kkerr23@yahoo.com");
        SET custId = LAST_INSERT_ID();
    END IF;

    -- Add the reservation
    INSERT INTO Reservations (customerId, reservationDate, specialRequests)
    VALUES (12, '2024-05-13 17:00:00', 'No Special Request');
END //

DELIMITER ;
```



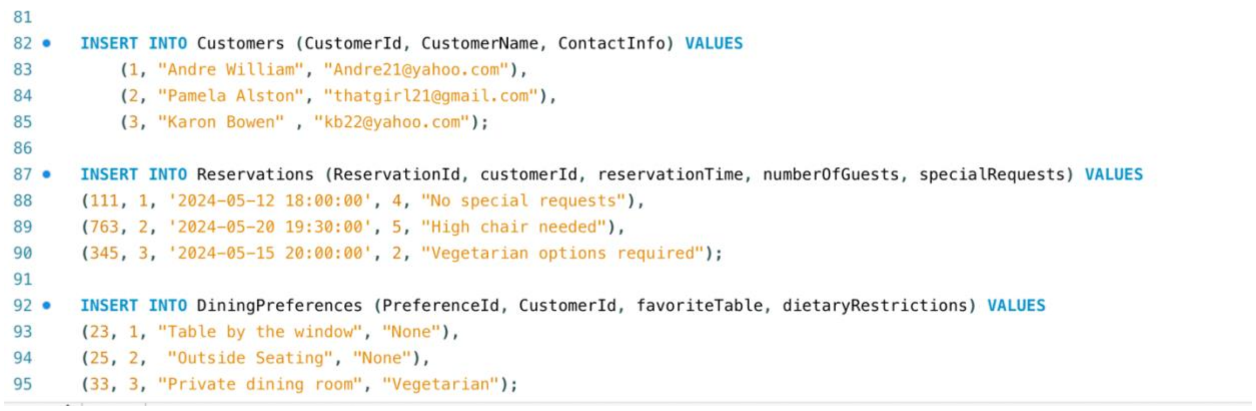
```sql
# STANDARD WAY
DELIMITER //

CREATE PROCEDURE addReservation(
    IN CustomerId INT,
    IN CustomerName VARCHAR(45),
    IN ReservationTime DATETIME,
    IN NumberOfGuests INT,
    IN ContactInfo VARCHAR(200),
    IN SpecialRequests VARCHAR(200),
    IN FavoriteTable VARCHAR(45),
    IN DietaryRestrictions VARCHAR(200)
)
BEGIN
    DECLARE custId INT;

    -- Check if customer already exists
    SELECT customerId INTO custId
    FROM Customers
    WHERE ContactInfo = Email
    LIMIT 1;

    -- If customer does not exist, create a new customer
    IF custId IS NULL THEN
        INSERT INTO Customers (CustomerId, CustomerName, ContactInfo)
        VALUES (12, "Kemar Kerr" , "kkerr23@yahoo.com");
        SET custId = LAST_INSERT_ID();
    END IF;
```

To test the system, I entered some basic data into the tables

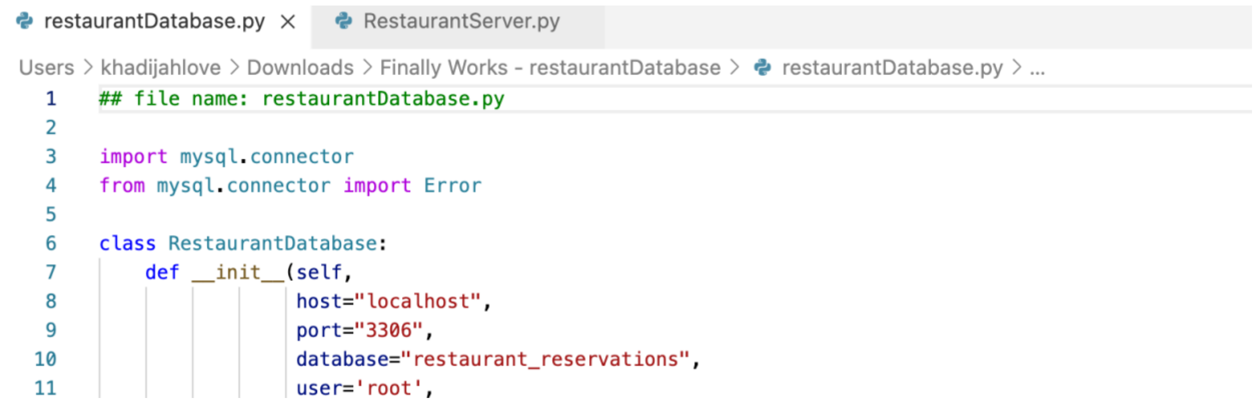


```sql
INSERT INTO Customers (CustomerId, CustomerName, ContactInfo) VALUES
    (1, "Andre William", "Andre21@yahoo.com"),
    (2, "Pamela Alston", "thatgirl21@gmail.com"),
    (3, "Karon Bowen" , "kb22@yahoo.com");

INSERT INTO Reservations (ReservationId, customerId, reservationTime, numberOfGuests, specialRequests) VALUES
(111, 1, '2024-05-12 18:00:00', 4, "No special requests"),
(763, 2, '2024-05-20 19:30:00', 5, "High chair needed"),
(345, 3, '2024-05-15 20:00:00', 2, "Vegetarian options required");

INSERT INTO DiningPreferences (PreferenceId, CustomerId, favoriteTable, dietaryRestrictions) VALUES
(23, 1, "Table by the window", "None"),
(25, 2,  "Outside Seating", "None"),
(33, 3, "Private dining room", "Vegetarian");
```

After completing the MYSQL queries I downloaded the two files provided which where the RestuarantServer.py and the restaurantDatabase.py to connect to mysql inorder to run the portal. In my visual studios I went ahead to install pip3 install mysql-connector and then pip3 install mysql-connector-python which allowed me to connect to the Restaurant portal.



restaurantDatabase.py × RestaurantServer.py

Users > khadijahlove > Downloads > Finally Works - restaurantDatabase > restaurantDatabase.py > ...

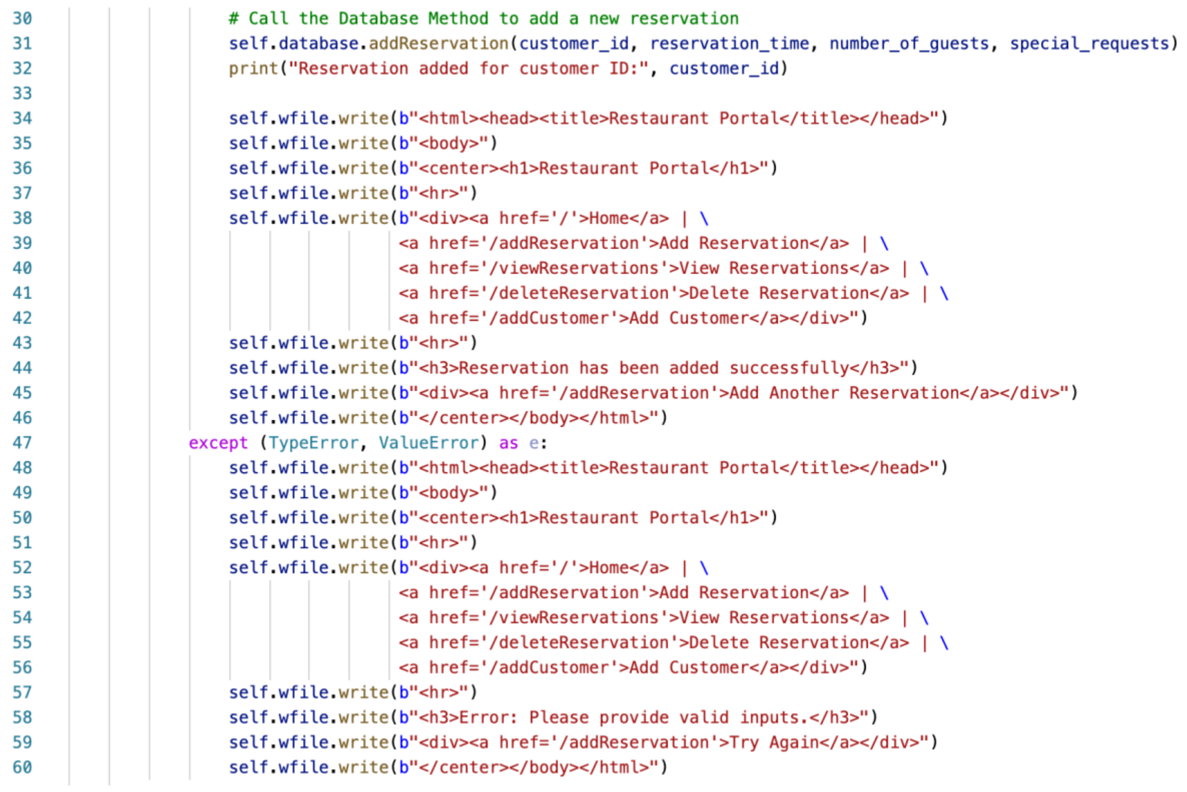
```python
## file name: restaurantDatabase.py

import mysql.connector
from mysql.connector import Error

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

The Password is ‘1105Friends’.

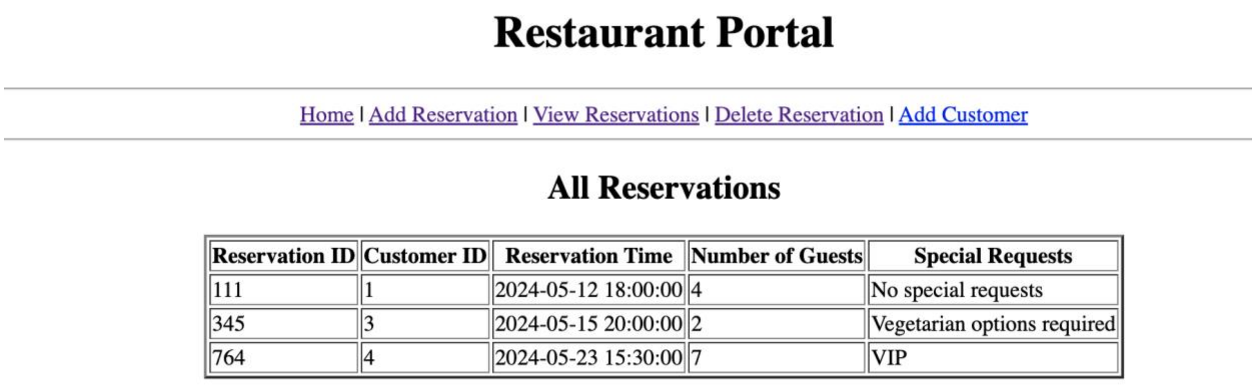
I completed the **addReservation** method to insert a new reservation into the database:



```
        # Call the Database Method to add a new reservation
        self.database.addReservation(customer_id, reservation_time, number_of_guests, special_requests)
        print("Reservation added for customer ID:", customer_id)

        self.wfile.write(b"<html><head><title>Restaurant Portal</title></head>")
        self.wfile.write(b"<body>")
        self.wfile.write(b"<center><h1>Restaurant Portal</h1>")
        self.wfile.write(b"<hr>")
        self.wfile.write(b"<div><a href='/'>Home</a> | \
                         <a href='/addReservation'>Add Reservation</a> | \
                         <a href='/viewReservations'>View Reservations</a> | \
                         <a href='/deleteReservation'>Delete Reservation</a> | \
                         <a href='/addCustomer'>Add Customer</a></div>")
        self.wfile.write(b"<hr>")
        self.wfile.write(b"<h3>Reservation has been added successfully</h3>")
        self.wfile.write(b"<div><a href='/addReservation'>Add Another Reservation</a></div>")
        self.wfile.write(b"</center></body></html>")
    except (TypeError, ValueError) as e:
        self.wfile.write(b"<html><head><title>Restaurant Portal</title></head>")
        self.wfile.write(b"<body>")
        self.wfile.write(b"<center><h1>Restaurant Portal</h1>")
        self.wfile.write(b"<hr>")
        self.wfile.write(b"<div><a href='/'>Home</a> | \
                         <a href='/addReservation'>Add Reservation</a> | \
                         <a href='/viewReservations'>View Reservations</a> | \
                         <a href='/deleteReservation'>Delete Reservation</a> | \
                         <a href='/addCustomer'>Add Customer</a></div>")
        self.wfile.write(b"<hr>")
        self.wfile.write(b"<h3>Error: Please provide valid inputs.</h3>")
        self.wfile.write(b"<div><a href='/addReservation'>Try Again</a></div>")
        self.wfile.write(b"</center></body></html>")
```

I ran restaurantServer.py and checked the web interface at localhost:8002/ to verify that the reservation system worked correctly.



Restaurant Portal

Home | Add Reservation | View Reservations | Delete Reservation | Add Customer

All Reservations

Reservation ID	Customer ID	Reservation Time	Number of Guests	Special Requests
111	1	2024-05-12 18:00:00	4	No special requests
345	3	2024-05-15 20:00:00	2	Vegetarian options required
764	4	2024-05-23 15:30:00	7	VIP

The creation of an extensive restaurant reservation system was the focus of this project. I effectively finished all necessary functionality and numerous extra features by configuring a MySQL database, making pertinent tables and stored procedures, adding initial data to the tables, and connecting the database with a Python frontend. The project offers a strong basis for future growth and improvements by demonstrating efficient database administration and Python-MySQL interface.

This is the link to my github repository.

https://github.com/khadijah6/MYSQL-SOURCECODE

https://github.com/khadijah6/MYSQL-SOURCECODE.git