School of Computer Science, UPES, Dehradun.

A

PROJECT REPORT FILE

On

DATABASE MANAGEMENT SYSTEM (DBMS) LAB

B.TECH. -III Semester

AUG. – NOV.- 2024.

Submitted to:

Rakesh Ranjan

Assistant Professor, UPES.

INDEX

Project Title:

"Customized Meal Recommendation System"

AIM:

The aim of this project is to develop a **customized meal recommendation system** that allows users to select meals based on their dietary preferences, health conditions, and personal nutrition goals. The system will store various meals, their nutritional values, and ingredients, and will allow users to place orders, leave reviews, and access meal plans tailored to specific health needs or fitness goals.

Objectives:

1. **User-Friendly Meal Selection**: Allow users to choose meals based on dietary preferences (Vegetarian, Non-Vegetarian, Vegan, etc.) and health conditions (e.g., Diabetes, Heart Disease).
2. **Nutritional Information Management**: Store detailed nutritional information for each meal, including calories, protein, fat, carbohydrates, and special dietary categories like gluten-free, keto, etc.
3. **Meal Plan Creation**: Provide meal plans tailored to specific goals, such as weight loss, muscle gain, or specific health conditions.
4. **Order and Review System**: Implement a system where users can place meal orders, review meals based on their experience, and rate them.
5. **Ingredient Tracking**: Store the ingredients of each meal and their quantities to manage meal compositions.
6. **Custom User Preferences**: Track users' favorite cuisines, preferred meal times, and order history to offer personalized meal recommendations.
7. **Database Management**: Ensure that the system maintains data integrity through the use of foreign keys and relational structures.

Command Used:

CREATE DATABASE meal_db; - Creates a new database named `meal_db`.
USE meal_db; - Selects the `meal_db` database.
CREATE TABLE Users (...); - Creates a table for user data (username, preferences, health).
CREATE TABLE Meals (...); - Creates a table for meal details (name, dietary preference, calories).
INSERT INTO Meals (...); - Adds meal entries to the `Meals` table.
CREATE TABLE Orders (...); - Creates a table for tracking user meal orders.

INSERT INTO Orders (...); - Adds order data (user, meal, date).
CREATE TABLE Reviews (...); - Creates a table for user reviews and ratings of meals.
INSERT INTO Reviews (...); - Adds review data (user, meal, rating).
CREATE TABLE Ingredients (...); - Creates a table for ingredient names.
INSERT INTO Ingredients (...); - Adds ingredient entries.
CREATE TABLE MealIngredients (...); - Links meals with ingredients and quantities.
INSERT INTO MealIngredients (...); - Adds meal-ingredient association data.
ALTER TABLE Users ADD COLUMN preferred_meal_time; - Adds meal time preference column.
ALTER TABLE Users ADD COLUMN favorite_cuisine; - Adds favorite cuisine column.
ALTER TABLE Meals ADD COLUMN protein; - Adds a column for protein content.
ALTER TABLE Meals ADD COLUMN fat; - Adds a column for fat content.
ALTER TABLE Meals ADD COLUMN carbohydrates; - Adds a column for carbohydrate content.
ALTER TABLE Meals ADD COLUMN meal_category; - Adds meal category (e.g., breakfast).
CREATE TABLE MealPlans (...); - Creates a table for different meal plans.
INSERT INTO MealPlans (...); - Adds meal plan data.
CREATE TABLE PlanMeals (...); - Links meals to meal plans.
INSERT INTO PlanMeals (...); - Adds meal-plan associations.

Technologies Used:

MySQL Server
Python

Documentation of "Customized Meal Recommendation System"

This documentation provides a detailed explanation of each function in the "Customized Meal Recommendation System," which allows users to register, get meal recommendations, place orders, add reviews, manage ingredients, create meal plans, and add sample data.

Documentation is divided into 2 parts:
1. Database Documentation
2. Python Program Documentation

DATABASE DOCUMENTATION

Database Name: `meal_db`

This database is designed to manage meal plans, user preferences, meal orders, reviews, and ingredients in a customized meal recommendation system. It stores detailed information about users, meals, their nutritional values, health conditions, ingredients, meal plans, and orders placed by users.

Tables Overview

1. **Users**
2. **Meals**
3. **Orders**
4. **Reviews**
5. **Ingredients**
6. **MealIngredients**
7. **MealPlans**
8. **PlanMeals**

1. Users Table

The Users table stores information about the users of the system, including their dietary preferences, health conditions, and other relevant details.

2. Meals Table

The Meals table stores information about each meal available in the system, including dietary preferences, associated health conditions, and nutritional information.

3. Orders Table

The Orders table stores information about the orders placed by users, associating a specific user with a meal and tracking the date of the order.

4. Reviews Table

The Reviews table stores feedback from users on meals, including a rating and optional review text.

5. Ingredients Table

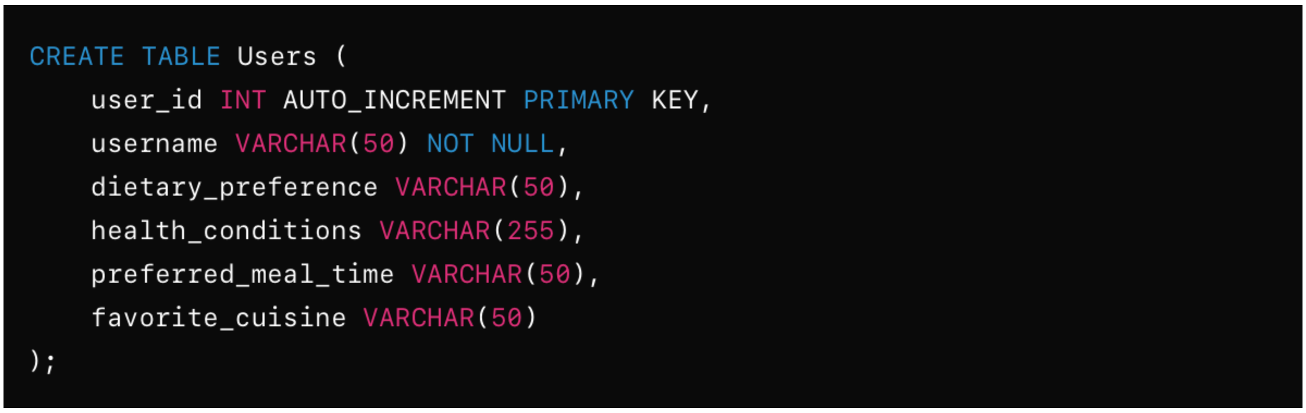
The Ingredients table stores the list of ingredients used in different meals.

6. MealIngredients Table

The MealIngredients table serves as a many-to-many relationship table between Meals and Ingredients. It specifies the quantity of each ingredient used in a particular meal.

7. MealPlans Table

The MealPlans table stores information about different meal plans, including the name and description of the plan.

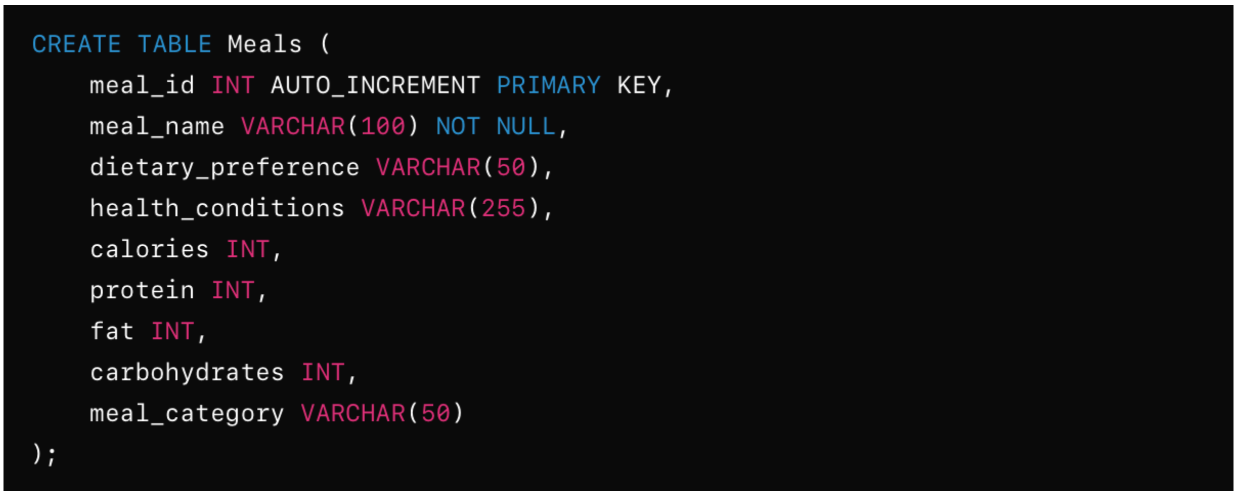


```sql
CREATE TABLE Users (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    dietary_preference VARCHAR(50),
    health_conditions VARCHAR(255),
    preferred_meal_time VARCHAR(50),
    favorite_cuisine VARCHAR(50)
);
```

8. PlanMeals Table

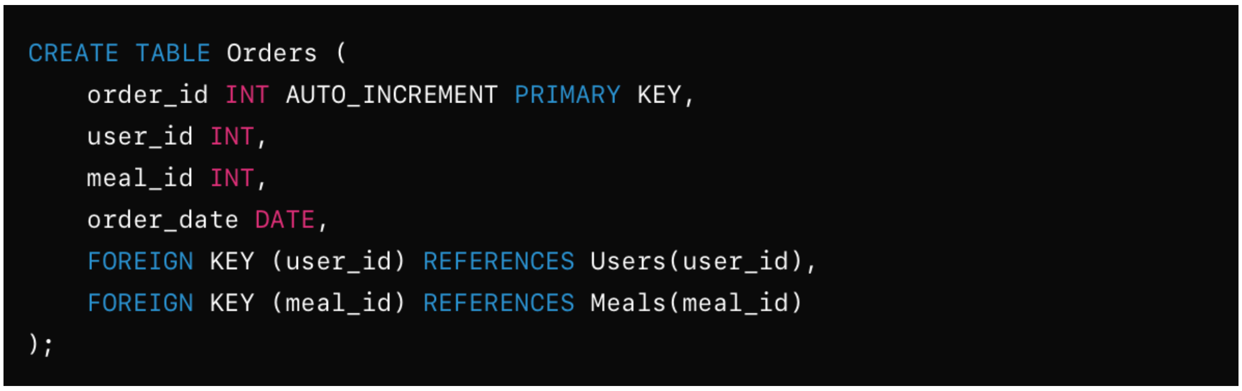
The PlanMeals table serves as a junction table between MealPlans and Meals, linking each meal plan with specific meals.

Data Insertions



```
CREATE TABLE Meals (
    meal_id INT AUTO_INCREMENT PRIMARY KEY,
    meal_name VARCHAR(100) NOT NULL,
    dietary_preference VARCHAR(50),
    health_conditions VARCHAR(255),
    calories INT,
    protein INT,
    fat INT,
    carbohydrates INT,
    meal_category VARCHAR(50)
);
```

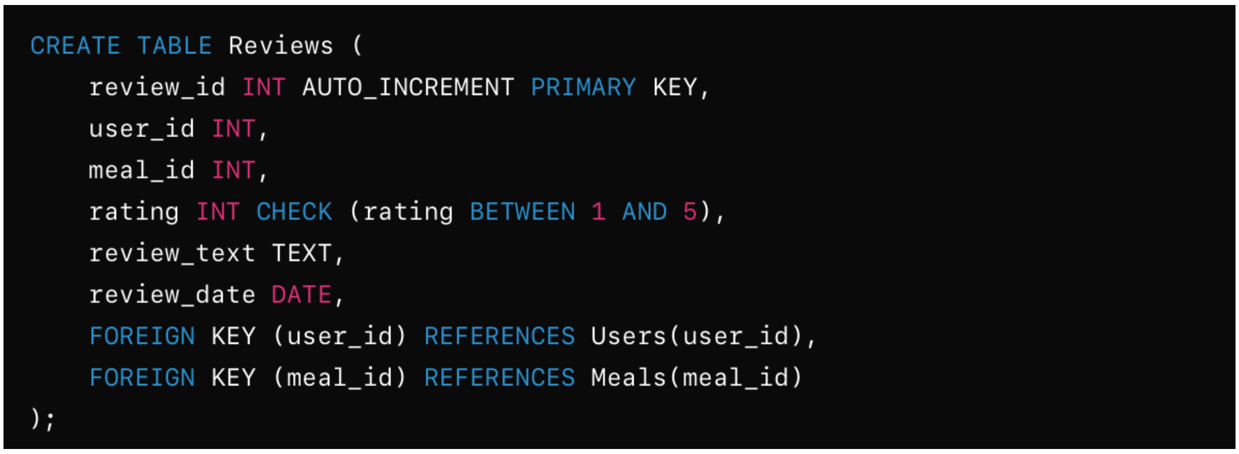
- Sample meals, orders, reviews, ingredients, and meal plans have been inserted as example data into the respective tables.
- **Meals**: The Meals table is populated with various non-vegetarian, vegetarian, vegan, gluten-free, dairy-free, keto, low-calorie, and high-calorie meals, addressing different dietary



```
CREATE TABLE Orders (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    meal_id INT,
    order_date DATE,
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (meal_id) REFERENCES Meals(meal_id)
);
```

preferences and health conditions.

- **Orders**: Example orders have been placed by users.



```
CREATE TABLE Reviews (
    review_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    meal_id INT,
    rating INT CHECK (rating BETWEEN 1 AND 5),
    review_text TEXT,
    review_date DATE,
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (meal_id) REFERENCES Meals(meal_id)
);
```

- **Reviews**: Users have submitted reviews for meals they ordered.

```sql
CREATE TABLE Ingredients (
    ingredient_id INT AUTO_INCREMENT PRIMARY KEY,
    ingredient_name VARCHAR(100) NOT NULL
);
```

Modifications to Schema

- **Users**: Added preferred_meal_time and favorite_cuisine columns to capture user preferences.
- **Meals**: Added columns for protein, fat, carbohydrates, and meal_category to enhance

```sql
CREATE TABLE MealIngredients (
    meal_id INT,
    ingredient_id INT,
    quantity VARCHAR(50),
    PRIMARY KEY (meal_id, ingredient_id),
    FOREIGN KEY (meal_id) REFERENCES Meals(meal_id),
    FOREIGN KEY (ingredient_id) REFERENCES Ingredients(ingredient_id)
);
```

meal nutritional data.

Foreign Key Constraints

```sql
CREATE TABLE MealPlans (
    plan_id INT AUTO_INCREMENT PRIMARY KEY,
    plan_name VARCHAR(100) NOT NULL,
    description TEXT
);
```

- Foreign keys ensure referential integrity across tables (e.g., user_id in Orders, meal_id in MealIngredients, etc.).

ENTITY RELATIONSHIP MODEL (ER-MODEL)

```
CREATE TABLE PlanMeals (
    plan_id INT,
    meal_id INT,
    PRIMARY KEY (plan_id, meal_id),
    FOREIGN KEY (plan_id) REFERENCES MealPlans(plan_id),
    FOREIGN KEY (meal_id) REFERENCES Meals(meal_id)
);
```

PYTHON PROGRAM DOCUMENTATION

1. get_db_connection()

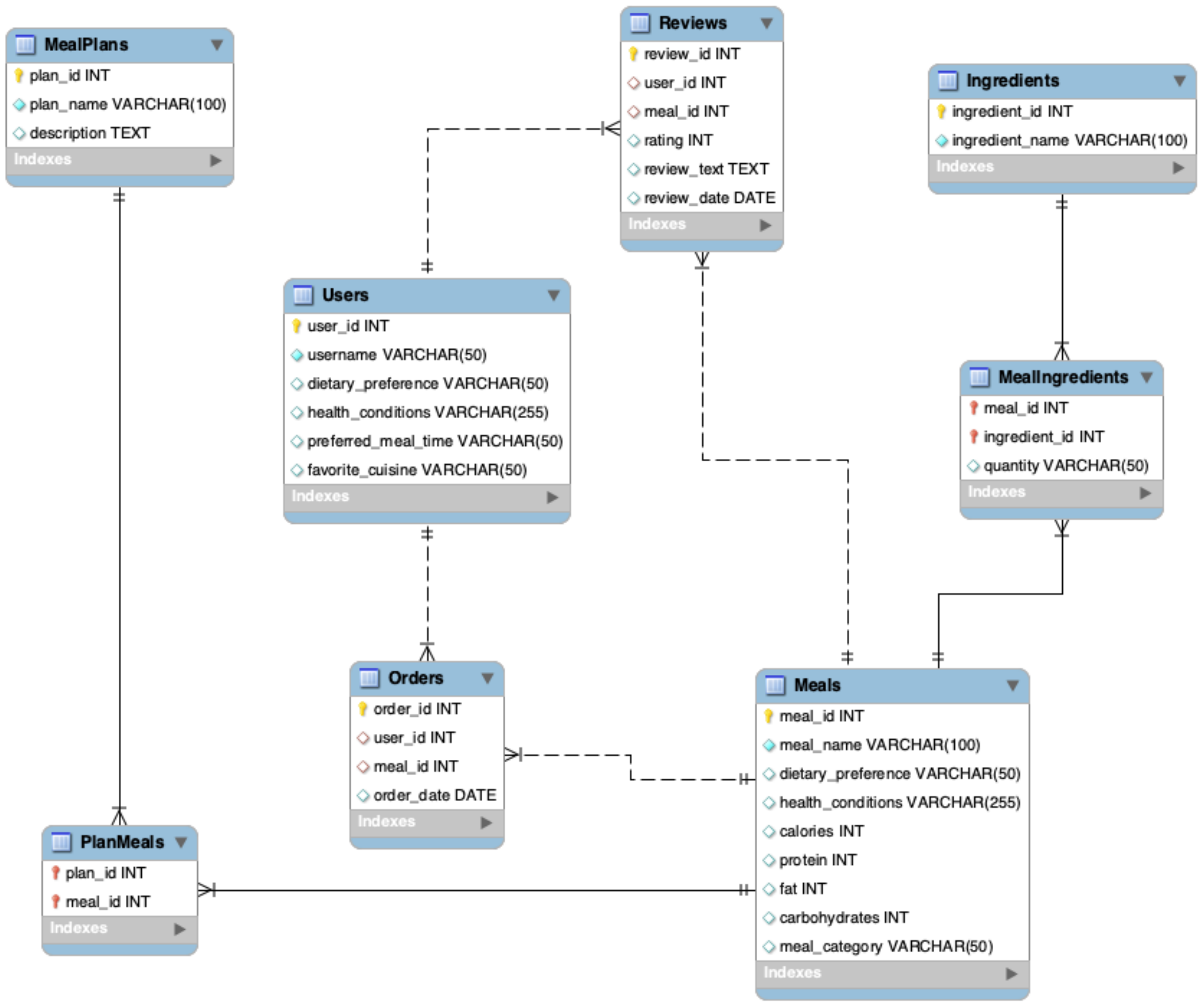
Description:

This function establishes and returns a connection to the MySQL database using the provided credentials (host, user, password, and database). The connection is used to interact with the database throughout the application.

2. register_user()

Description:

This function allows a new user to register by entering details like their username, dietary preference, health conditions, preferred meal time, and favorite cuisine. The details are stored in the Users table



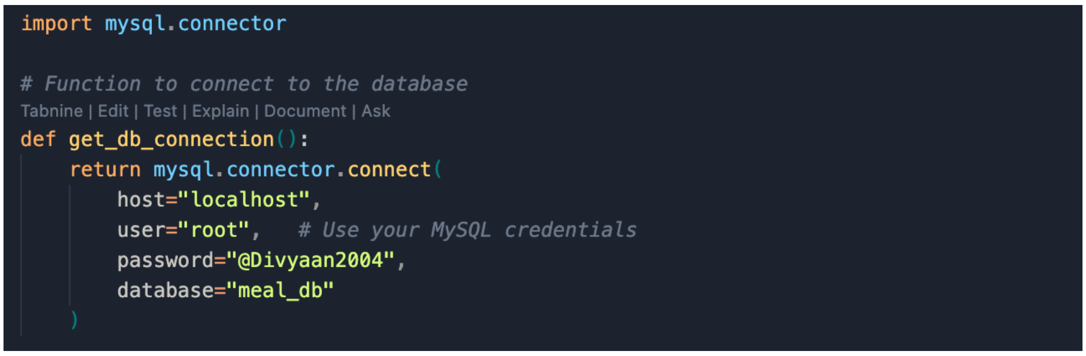
of the database.

3. recommend_meals()

Description:

This function fetches the user's dietary preferences and health conditions from the Users table and recommends meals from the Meals table that match the user's preferences.

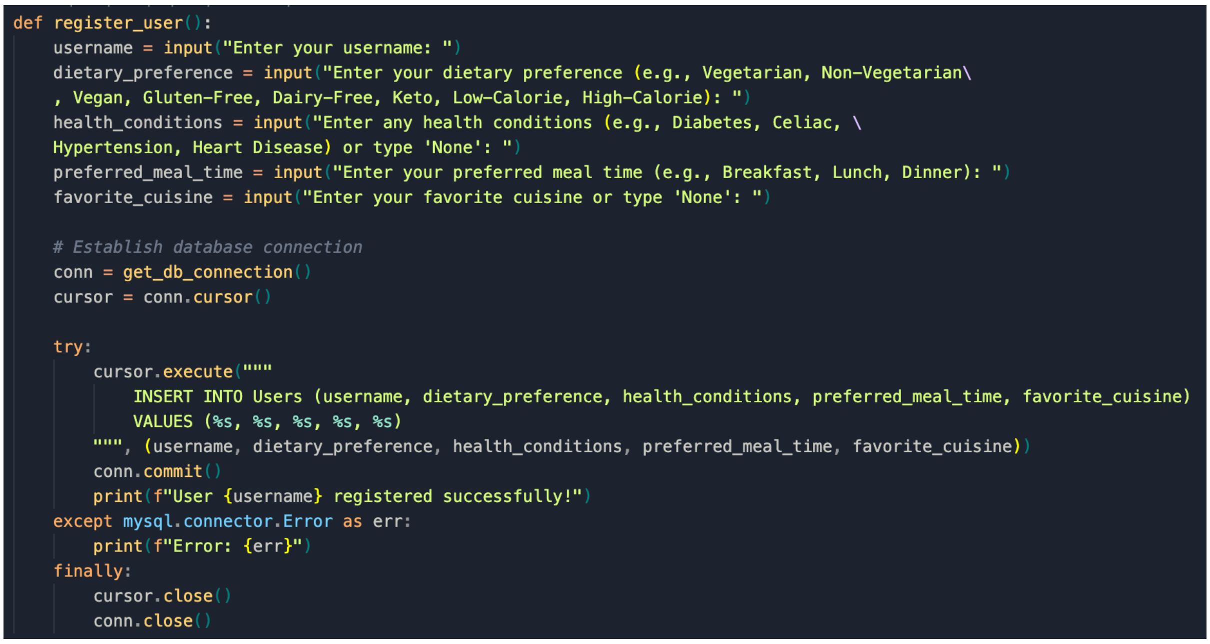
4. place_order()



```
import mysql.connector

# Function to connect to the database
Tabnine | Edit | Test | Explain | Document | Ask
def get_db_connection():
    return mysql.connector.connect(
        host="localhost",
        user="root",    # Use your MySQL credentials
        password="@Divyaan2004",
        database="meal_db"
    )
```

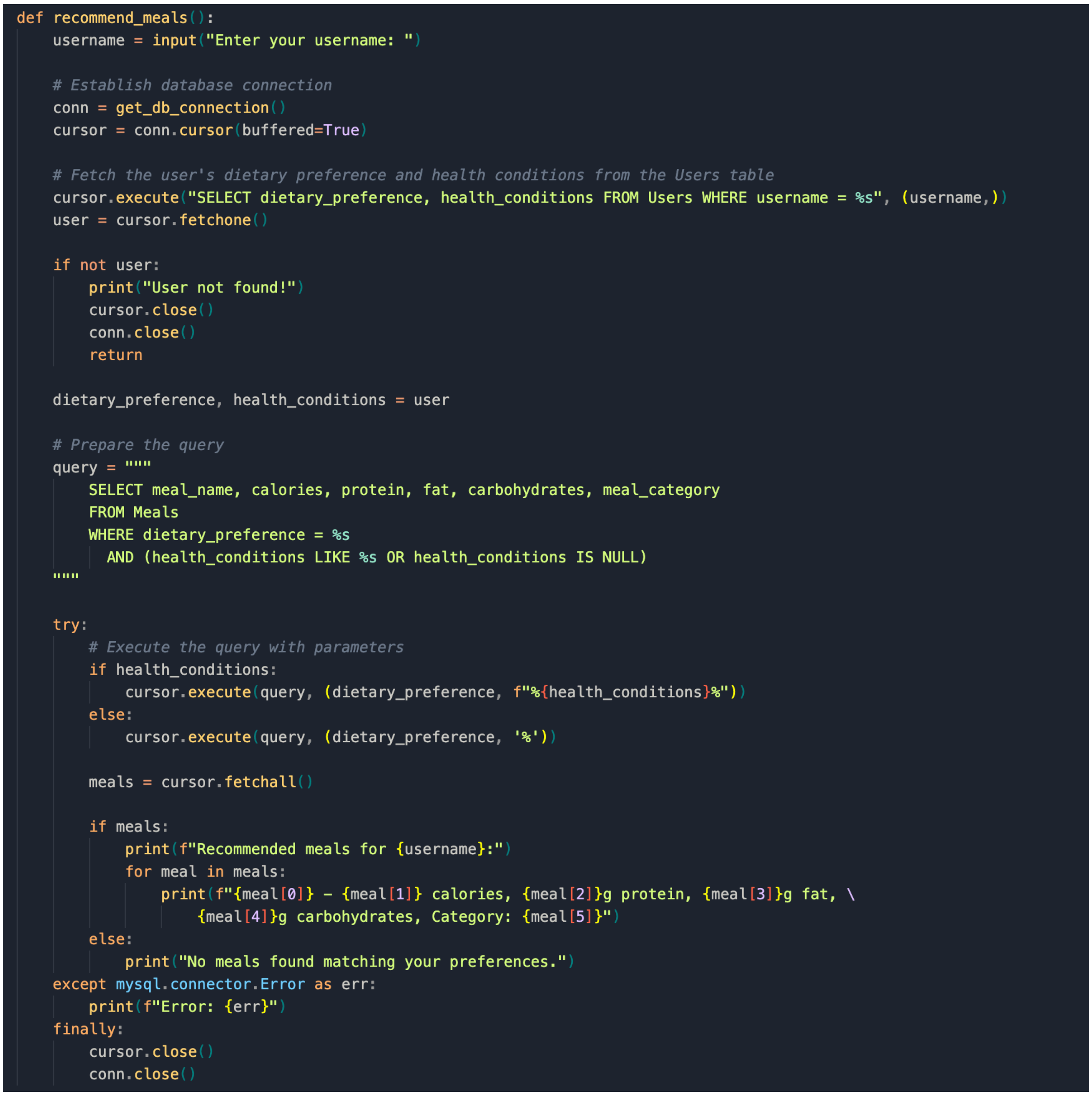
Description:

This function allows a user to place an order by selecting a meal from the Meals table. The meal is associated with the user's ID in the Orders table, and the order date is recorded.



```
def register_user():
    username = input("Enter your username: ")
    dietary_preference = input("Enter your dietary preference (e.g., Vegetarian, Non-Vegetarian\
, Vegan, Gluten-Free, Dairy-Free, Keto, Low-Calorie, High-Calorie): ")
    health_conditions = input("Enter any health conditions (e.g., Diabetes, Celiac, \
Hypertension, Heart Disease) or type 'None': ")
    preferred_meal_time = input("Enter your preferred meal time (e.g., Breakfast, Lunch, Dinner): ")
    favorite_cuisine = input("Enter your favorite cuisine or type 'None': ")

    # Establish database connection
    conn = get_db_connection()
    cursor = conn.cursor()

    try:
        cursor.execute("""
            INSERT INTO Users (username, dietary_preference, health_conditions, preferred_meal_time, favorite_cuisine)
            VALUES (%s, %s, %s, %s, %s)
        """, (username, dietary_preference, health_conditions, preferred_meal_time, favorite_cuisine))
        conn.commit()
        print(f"User {username} registered successfully!")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    finally:
        cursor.close()
        conn.close()
```

5. add_review()

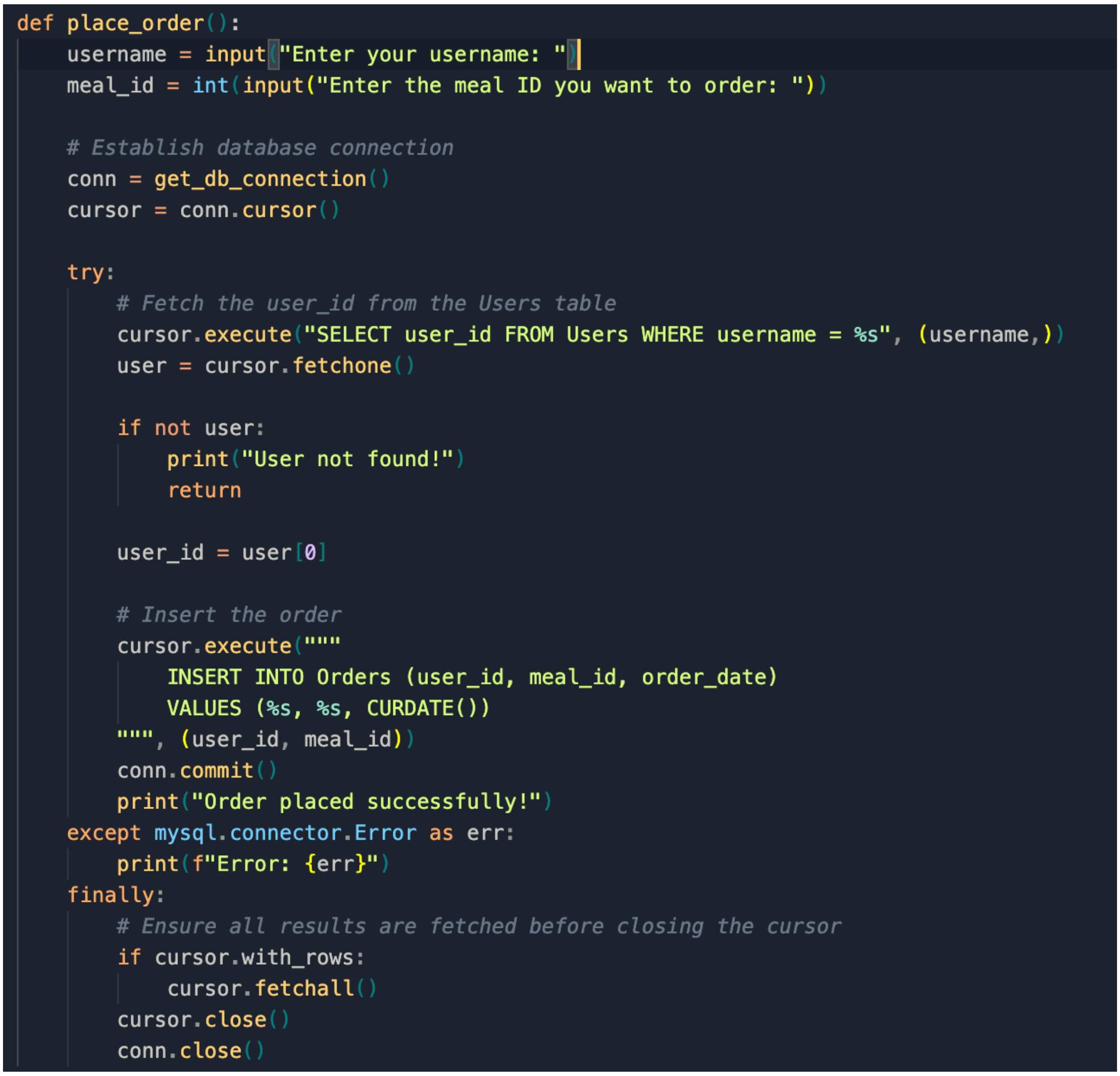


```python
def recommend_meals():
    username = input("Enter your username: ")

    # Establish database connection
    conn = get_db_connection()
    cursor = conn.cursor(buffered=True)

    # Fetch the user's dietary preference and health conditions from the Users table
    cursor.execute("SELECT dietary_preference, health_conditions FROM Users WHERE username = %s", (username,))
    user = cursor.fetchone()

    if not user:
        print("User not found!")
        cursor.close()
        conn.close()
        return

    dietary_preference, health_conditions = user

    # Prepare the query
    query = """
        SELECT meal_name, calories, protein, fat, carbohydrates, meal_category
        FROM Meals
        WHERE dietary_preference = %s
          AND (health_conditions LIKE %s OR health_conditions IS NULL)
    """

    try:
        # Execute the query with parameters
        if health_conditions:
            cursor.execute(query, (dietary_preference, f"%{health_conditions}%"))
        else:
            cursor.execute(query, (dietary_preference, '%'))

        meals = cursor.fetchall()

        if meals:
            print(f"Recommended meals for {username}:")
            for meal in meals:
                print(f"{meal[0]} - {meal[1]} calories, {meal[2]}g protein, {meal[3]}g fat, \
                    {meal[4]}g carbohydrates, Category: {meal[5]}")
        else:
            print("No meals found matching your preferences.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    finally:
        cursor.close()
        conn.close()
```

Description:

This function allows a user to leave a review for a meal by entering a rating (1-5) and a textual review. The review is stored in the Reviews table.

6. add_ingredients()



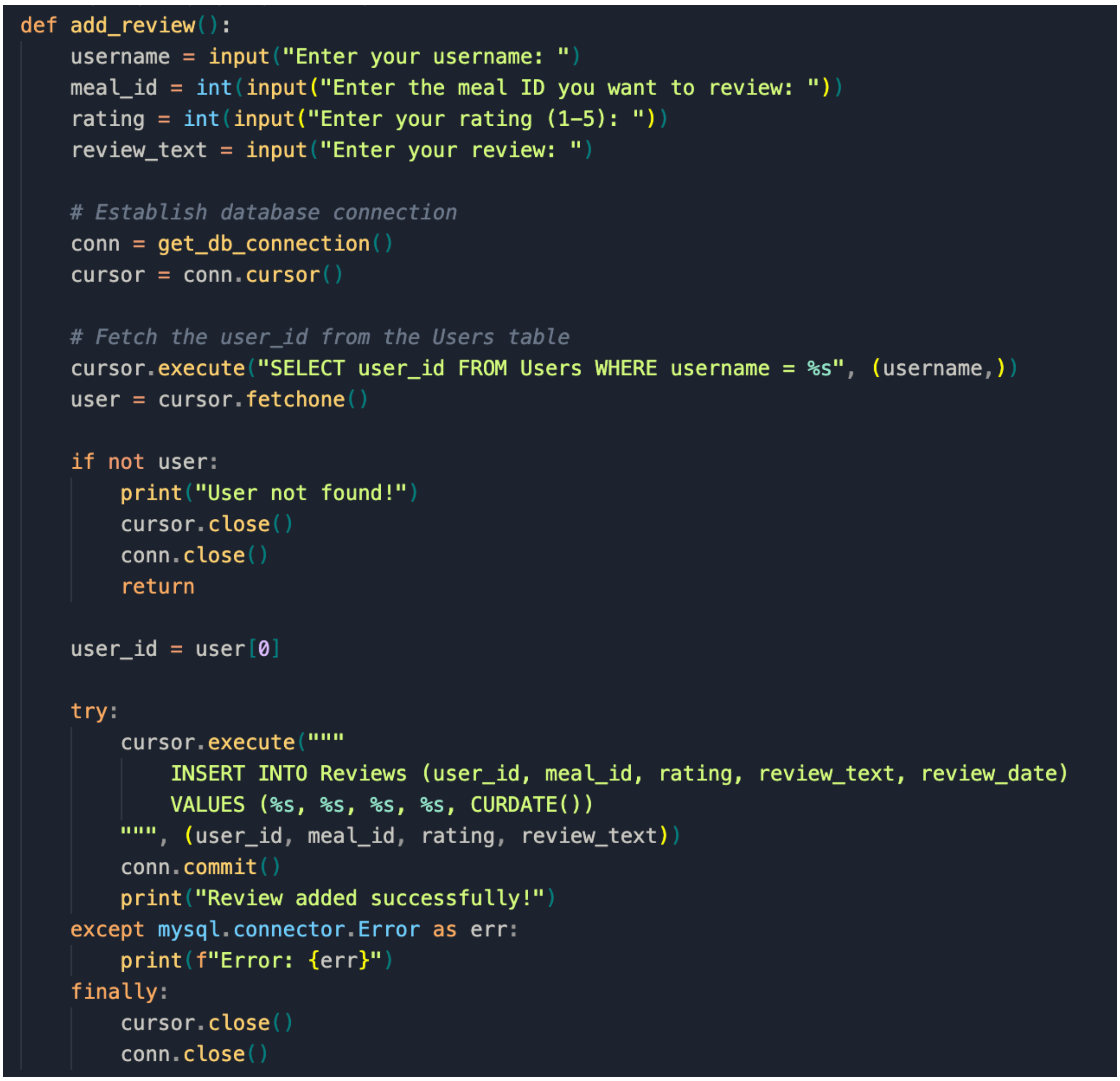
```python
def place_order():
    username = input("Enter your username: ")
    meal_id = int(input("Enter the meal ID you want to order: "))

    # Establish database connection
    conn = get_db_connection()
    cursor = conn.cursor()

    try:
        # Fetch the user_id from the Users table
        cursor.execute("SELECT user_id FROM Users WHERE username = %s", (username,))
        user = cursor.fetchone()

        if not user:
            print("User not found!")
            return

        user_id = user[0]

        # Insert the order
        cursor.execute("""
            INSERT INTO Orders (user_id, meal_id, order_date)
            VALUES (%s, %s, CURDATE())
        """, (user_id, meal_id))
        conn.commit()
        print("Order placed successfully!")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    finally:
        # Ensure all results are fetched before closing the cursor
        if cursor.with_rows:
            cursor.fetchall()
        cursor.close()
        conn.close()
```

Description:

This function allows the addition of ingredients to a meal. If an ingredient is not already in the Ingredients table, it is first inserted into the table. The relationship between meals and ingredients is stored in the MealIngredients table.

7. create_meal_plan()

Description:



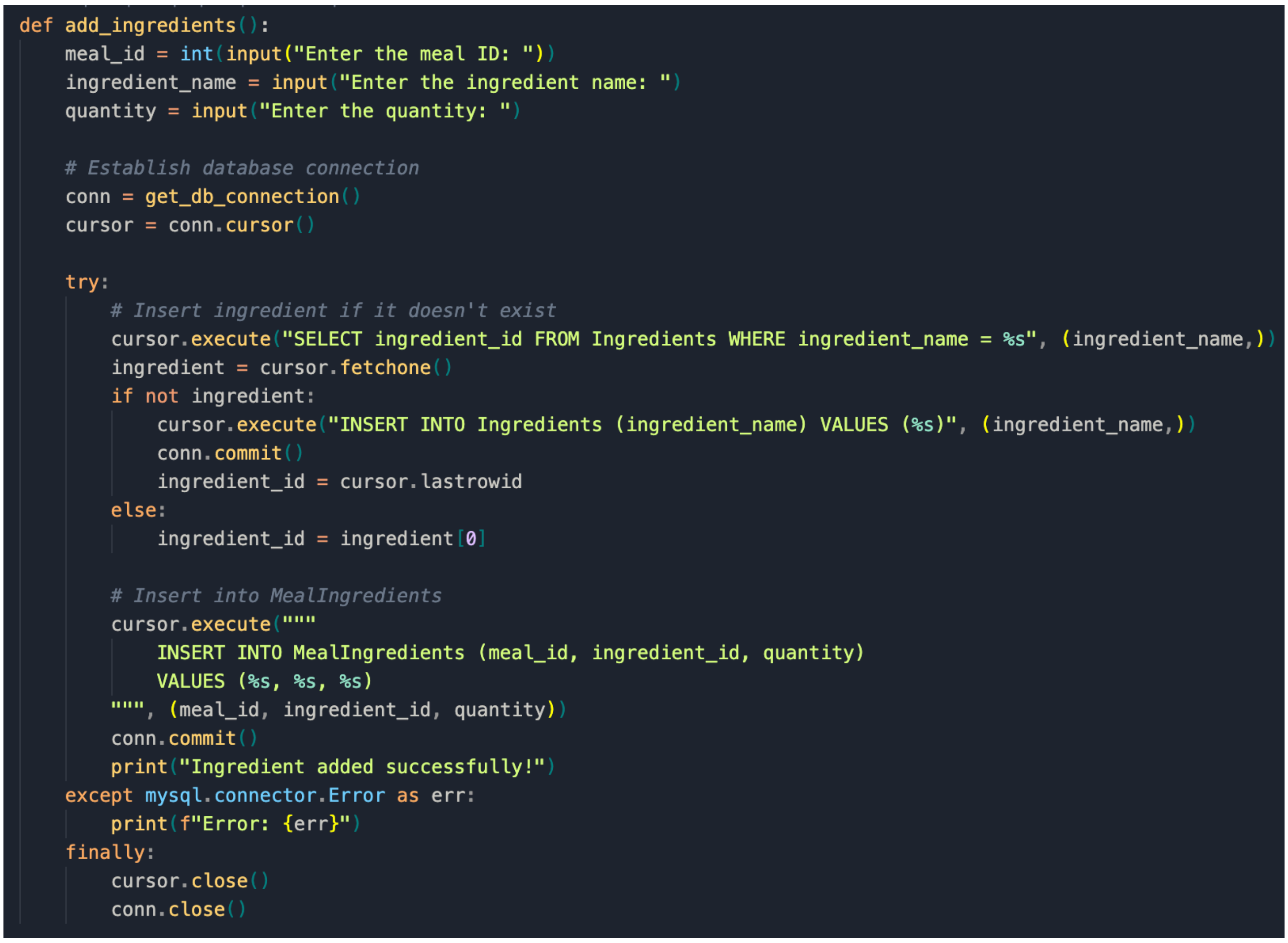
```python
def add_review():
    username = input("Enter your username: ")
    meal_id = int(input("Enter the meal ID you want to review: "))
    rating = int(input("Enter your rating (1-5): "))
    review_text = input("Enter your review: ")

    # Establish database connection
    conn = get_db_connection()
    cursor = conn.cursor()

    # Fetch the user_id from the Users table
    cursor.execute("SELECT user_id FROM Users WHERE username = %s", (username,))
    user = cursor.fetchone()

    if not user:
        print("User not found!")
        cursor.close()
        conn.close()
        return

    user_id = user[0]

    try:
        cursor.execute("""
            INSERT INTO Reviews (user_id, meal_id, rating, review_text, review_date)
            VALUES (%s, %s, %s, %s, CURDATE())
        """, (user_id, meal_id, rating, review_text))
        conn.commit()
        print("Review added successfully!")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    finally:
        cursor.close()
        conn.close()
```

This function allows a user to create a new meal plan by providing the plan's name and description. The meal plan is stored in the MealPlans table.

8. add_meals_to_plan()

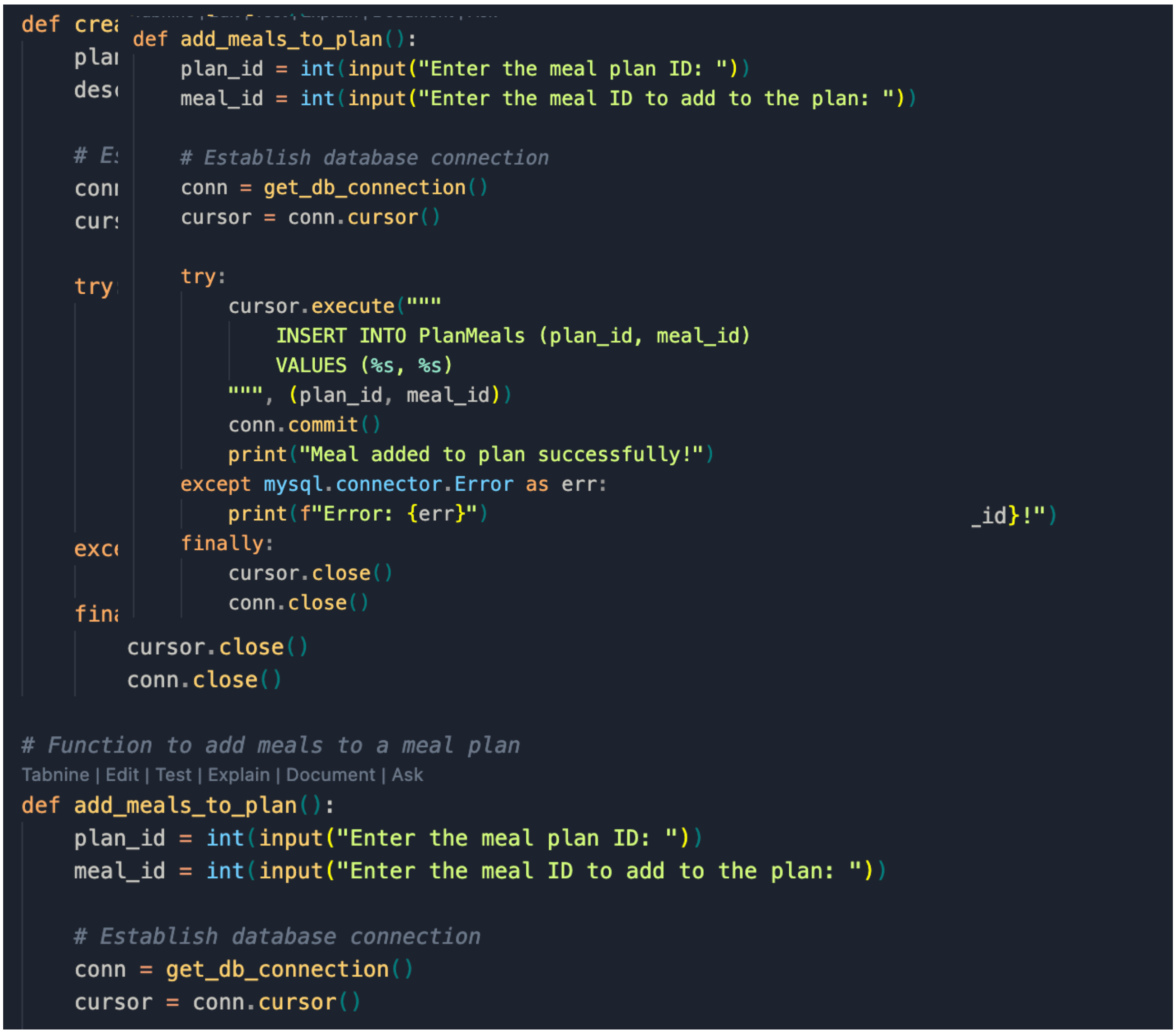
Description:

This function allows a user to add meals to an existing meal plan. The meal-plan association is stored



```python
def add_ingredients():
    meal_id = int(input("Enter the meal ID: "))
    ingredient_name = input("Enter the ingredient name: ")
    quantity = input("Enter the quantity: ")

    # Establish database connection
    conn = get_db_connection()
    cursor = conn.cursor()

    try:
        # Insert ingredient if it doesn't exist
        cursor.execute("SELECT ingredient_id FROM Ingredients WHERE ingredient_name = %s", (ingredient_name,))
        ingredient = cursor.fetchone()
        if not ingredient:
            cursor.execute("INSERT INTO Ingredients (ingredient_name) VALUES (%s)", (ingredient_name,))
            conn.commit()
            ingredient_id = cursor.lastrowid
        else:
            ingredient_id = ingredient[0]

        # Insert into MealIngredients
        cursor.execute("""
            INSERT INTO MealIngredients (meal_id, ingredient_id, quantity)
            VALUES (%s, %s, %s)
        """, (meal_id, ingredient_id, quantity))
        conn.commit()
        print("Ingredient added successfully!")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    finally:
        cursor.close()
        conn.close()
```

in the PlanMealstable.



```
def crea
    plan
    desc

    # E
    conn
    curs

    try

    exce

    fina
        cursor.close()
        conn.close()
```

```
def add_meals_to_plan():
    plan_id = int(input("Enter the meal plan ID: "))
    meal_id = int(input("Enter the meal ID to add to the plan: "))

    # Establish database connection
    conn = get_db_connection()
    cursor = conn.cursor()

    try:
        cursor.execute("""
            INSERT INTO PlanMeals (plan_id, meal_id)
            VALUES (%s, %s)
        """, (plan_id, meal_id))
        conn.commit()
        print("Meal added to plan successfully!")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    finally:
        cursor.close()
        conn.close()
```

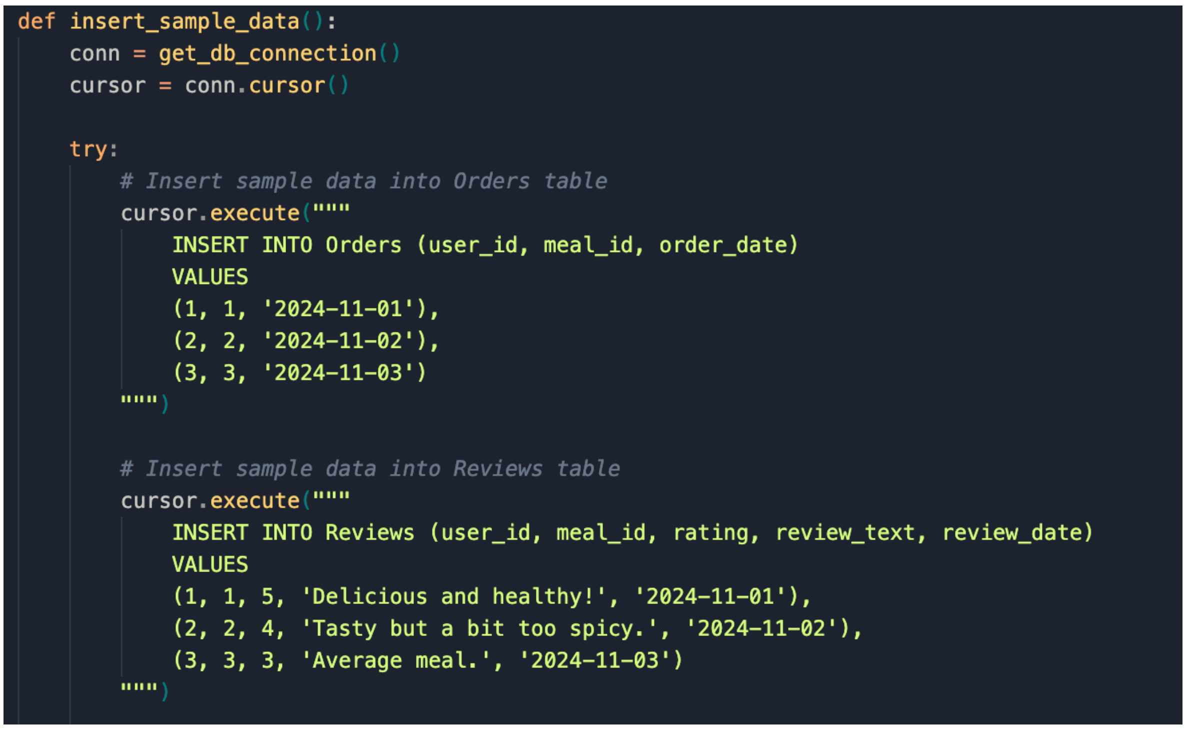
```
_id}!")
```

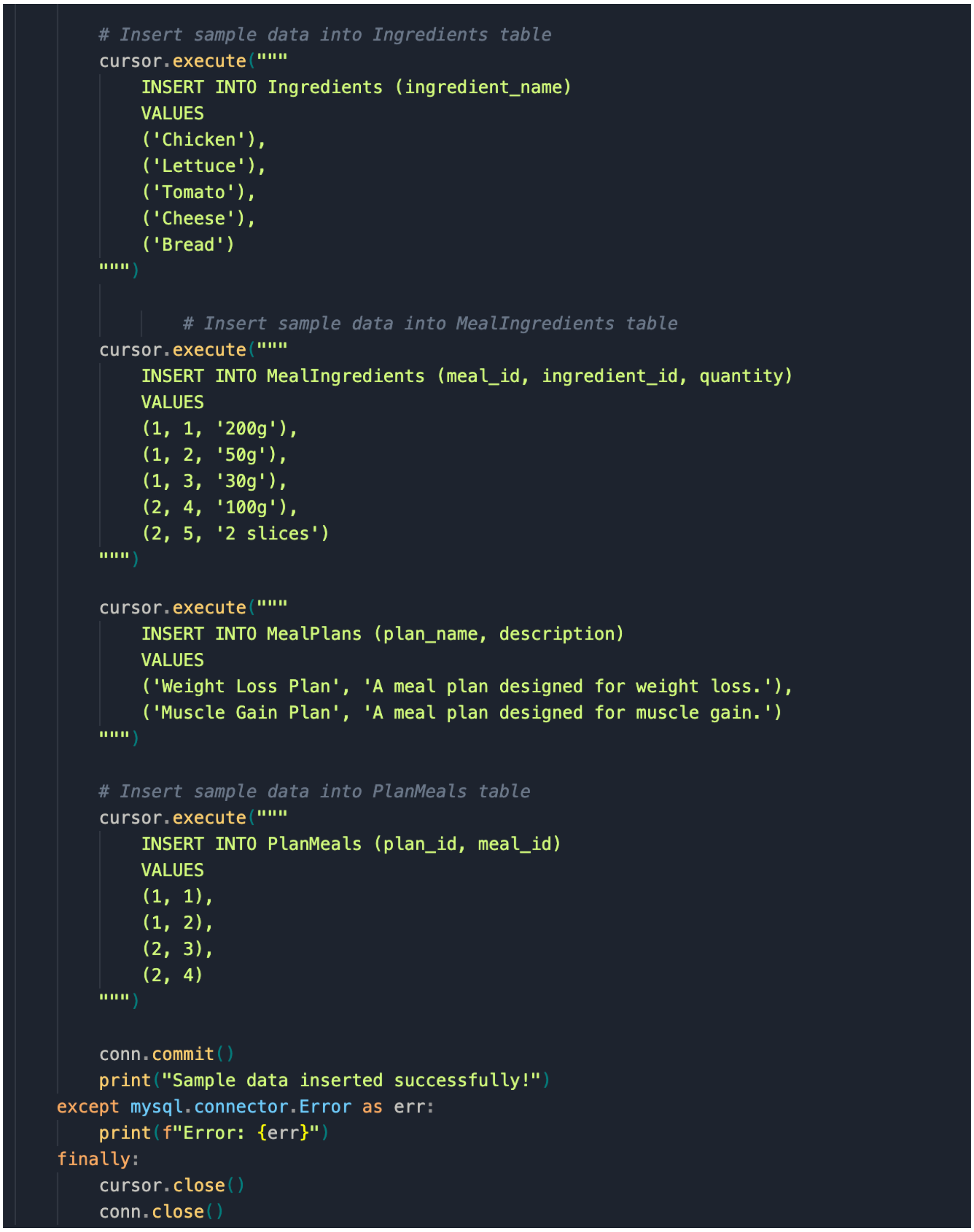
```
# Function to add meals to a meal plan
Tabnine | Edit | Test | Explain | Document | Ask
def add_meals_to_plan():
    plan_id = int(input("Enter the meal plan ID: "))
    meal_id = int(input("Enter the meal ID to add to the plan: "))

    # Establish database connection
    conn = get_db_connection()
    cursor = conn.cursor()
```

9. add_sample_data()

Description:

This function inserts sample data into the Meals, Users, Reviews, Orders, and other related tables for testing purposes.


```python
def insert_sample_data():
    conn = get_db_connection()
    cursor = conn.cursor()

    try:
        # Insert sample data into Orders table
        cursor.execute("""
            INSERT INTO Orders (user_id, meal_id, order_date)
            VALUES
            (1, 1, '2024-11-01'),
            (2, 2, '2024-11-02'),
            (3, 3, '2024-11-03')
        """)

        # Insert sample data into Reviews table
        cursor.execute("""
            INSERT INTO Reviews (user_id, meal_id, rating, review_text, review_date)
            VALUES
            (1, 1, 5, 'Delicious and healthy!', '2024-11-01'),
            (2, 2, 4, 'Tasty but a bit too spicy.', '2024-11-02'),
            (3, 3, 3, 'Average meal.', '2024-11-03')
        """)
```



```
        # Insert sample data into Ingredients table
        cursor.execute("""
            INSERT INTO Ingredients (ingredient_name)
            VALUES
            ('Chicken'),
            ('Lettuce'),
            ('Tomato'),
            ('Cheese'),
            ('Bread')
        """)

                # Insert sample data into MealIngredients table
        cursor.execute("""
            INSERT INTO MealIngredients (meal_id, ingredient_id, quantity)
            VALUES
            (1, 1, '200g'),
            (1, 2, '50g'),
            (1, 3, '30g'),
            (2, 4, '100g'),
            (2, 5, '2 slices')
        """)

        cursor.execute("""
            INSERT INTO MealPlans (plan_name, description)
            VALUES
            ('Weight Loss Plan', 'A meal plan designed for weight loss.'),
            ('Muscle Gain Plan', 'A meal plan designed for muscle gain.')
        """)

        # Insert sample data into PlanMeals table
        cursor.execute("""
            INSERT INTO PlanMeals (plan_id, meal_id)
            VALUES
            (1, 1),
            (1, 2),
            (2, 3),
            (2, 4)
        """)

        conn.commit()
        print("Sample data inserted successfully!")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    finally:
        cursor.close()
        conn.close()
```

Conclusion:

The "Customized Meal Recommendation System" is a comprehensive solution designed to personalize meal recommendations, orders, and meal plans for users based on their dietary preferences and health conditions. The system integrates several key functionalities, such as user registration, meal recommendations, order placement, reviews, and ingredient management, making it versatile and user-friendly.

Key highlights include:

- **Personalization:** The system tailors meal suggestions to individual users, considering factors like dietary preferences, health conditions, and meal timing.
- **Meal Management:** Meals can be added to the system, reviewed by users, and included in personalized meal plans.
- **User Interaction:** The system is interactive, allowing users to register, place orders, and provide feedback on meals easily.
- **Database Integration:** It uses a relational database (MySQL) to store user data, meal information, orders, reviews, and meal plans, ensuring efficient management and retrieval of data.