jupyter little-lemon-project Last Checkpoint: 2 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[46]:
```
import mysql.connector as connector
```

[2]:
```
connection = connector.connect(user="root", password="OJTJYUyEd5@V0Nq")
```

[3]:
```
cursor = connection.cursor()
```

[4]:
```
cursor.execute("CREATE DATABASE little_lemon_db")
```

[5]:
```
cursor.execute("USE little_lemon_db")
```

[8]:
```
#MenuItems table
create_menuitem_table = """CREATE TABLE MenuItems (
ItemID INT AUTO_INCREMENT,
Name VARCHAR(200),
Type VARCHAR(100),
Price INT,
PRIMARY KEY (ItemID)
);"""
```

[9]:
```
create_menu_table = """CREATE TABLE Menus (
MenuID INT,
ItemID INT,
Cuisine VARCHAR(100),
PRIMARY KEY (MenuID,ItemID)
);"""
```

[21]:
```
Create_booking_table = """CREATE TABLE Bookings (
BookingID INT AUTO_INCREMENT,
TableNo INT,
GuestFirstName VARCHAR(100) NOT NULL,
GuestLastName VARCHAR(100) NOT NULL,
BookingSlot TIME NOT NULL,
EmployeeID INT,
PRIMARY KEY (BookingID)
);"""
```

[18]:
```
create_orders_table = """CREATE TABLE Orders (
OrderID INT,
TableNo INT,
MenuID INT,
BookingID INT,
BillAmount INT,
Quantity INT,
PRIMARY KEY (OrderID,TableNo)
);"""
```

[19]:
```
create_employees_table = """CREATE TABLE Employees (
EmployeeID INT AUTO_INCREMENT PRIMARY KEY,
Name VARCHAR (255),
Role VARCHAR (100),
Address VARCHAR (255),
Contact_Number INT,
Email VARCHAR (255),
Annual_Salary VARCHAR (100)
);"""
```

[13]:
```
# Create MenuItems table
cursor.execute(create_menuitem_table)
```

[14]:
```
# Create Menu table
cursor.execute(create_menu_table)
```

[23]:
```
# Create Bookings table
cursor.execute(Create_booking_table)
```

[24]:
```
# Create Orders table
cursor.execute(create_orders_table)
```

[25]:
```
# Create Employees table
cursor.execute(create_employees_table)
```

[29]:
```
#########################################################
# Insert query to populate "MenuItems" table:
#########################################################
insert_menuitems="""
INSERT INTO MenuItems (ItemID, Name, Type, Price)
VALUES
(1, 'Olives','Starters',5),
(2, 'Flatbread','Starters', 5),
(3, 'Minestrone', 'Starters', 8),
(4, 'Tomato bread','Starters', 8),
(5, 'Falafel', 'Starters', 7),
(6, 'Hummus', 'Starters', 5),
(7, 'Greek salad', 'Main Courses', 15),
(8, 'Bean soup', 'Main Courses', 12),
(9, 'Pizza', 'Main Courses', 15),
(10, 'Greek yoghurt','Desserts', 7),
(11, 'Ice cream', 'Desserts', 6),
(12, 'Cheesecake', 'Desserts', 4),
(13, 'Athens White wine', 'Drinks', 25),
(14, 'Corfu Red Wine', 'Drinks', 30),
(15, 'Turkish Coffee', 'Drinks', 10),
(16, 'Turkish Coffee', 'Drinks', 10),
(17, 'Kabasa', 'Main Courses', 17);"""
```

[30]:
```
#########################################################
# Insert query to populate "Menu" table:
#########################################################
insert_menu="""
INSERT INTO Menus (MenuID,ItemID,Cuisine)
VALUES
(1, 1, 'Greek'),
(1, 7, 'Greek'),
(1, 10, 'Greek'),
(1, 13, 'Greek'),
(2, 3, 'Italian'),
(2, 9, 'Italian'),
(2, 12, 'Italian'),
(2, 15, 'Italian'),
(3, 5, 'Turkish'),
(3, 17, 'Turkish'),
(3, 11, 'Turkish'),
(3, 16, 'Turkish');"""
```

[31]:
```
#########################################################
```

[31]:
```
#*********************************************************#
# Insert query to populate "Bookings" table:
#*********************************************************#
insert_bookings="""
INSERT INTO Bookings (BookingID, TableNo, GuestFirstName,
GuestLastName, BookingSlot, EmployeeID)
VALUES
(1, 12, 'Anna','Iversen','19:00:00',1),
(2, 12, 'Joakim', 'Iversen', '19:00:00', 1),
(3, 19, 'Vanessa', 'McCarthy', '15:00:00', 3),
(4, 15, 'Marcos', 'Romero', '17:30:00', 4),
(5, 5, 'Hiroki', 'Yamane', '18:30:00', 2),
(6, 8, 'Diana', 'Pinto', '20:00:00', 5);"""
```

[32]:
```
#*********************************************************#
# Insert query to populate "Orders" table:
#*********************************************************#
insert_orders="""
INSERT INTO Orders (OrderID, TableNo, MenuID, BookingID, Quantity, BillAmount)
VALUES
(1, 12, 1, 1, 2, 86),
(2, 19, 2, 2, 1, 37),
(3, 15, 2, 3, 1, 37),
(4, 5, 3, 4, 1, 40),
(5, 8, 1, 5, 1, 43);"""
```

[41]:
```
#*********************************************************#
# Insert query to populate "Employees" table:
#*********************************************************#
insert_employees = """
INSERT INTO employees (EmployeeID, Name, Role, Address, Contact_Number, Email, Annual_Salary) VALUES
(01,'Mario Gollini','Manager','724, Parsley Lane, Old Town, Chicago, IL',351258074,'Mario.g@littlelemon.com','$70,000'),
(02,'Adrian Gollini','Assistant Manager','334, Dill Square, Lincoln Park, Chicago, IL',351474048,'Adrian.g@littlelemon.com','$65,000'),
(03,'Giorgos Dioudis','Head Chef','879 Sage Street, West Loop, Chicago, IL',351970582,'Giorgos.d@littlelemon.com','$50,000'),
(04,'Fatma Kaya','Assistant Chef','132  Bay Lane, Chicago, IL',351963569,'Fatma.k@littlelemon.com','$45,000'),
(05,'Elena Salvai','Head Waiter','989 Thyme Square, EdgeWater, Chicago, IL',351074198,'Elena.s@littlelemon.com','$40,000'),
(06,'John Millar','Receptionist','245 Dill Square, Lincoln Park, Chicago, IL',351584508,'John.m@littlelemon.com','$35,000');"""
```

[ ]:
```
# Populate MenuItems table
cursor.execute(insert_menuitems)
connection.commit()
```

[37]:
```
# Populate MenuItems table
cursor.execute(insert_menu)
connection.commit()
```

[38]:
```
# Populate Bookings table
cursor.execute(insert_bookings)
connection.commit()
```

[39]:
```
# Populate Orders table
cursor.execute(insert_orders)
connection.commit()
```

[42]:
```
# Populate Employees table
cursor.execute(insert_employees)
connection.commit()
```

[48]:
```
from mysql.connector.pooling import MySQLConnectionPool
```

[88]:
```
from mysql.connector import Error, PoolError
```

[50]:
```
dbconfig = {
    "database": "little_lemon_db",
    "user": "root",
    "password": "OJTJYUyEd5@V0Nq"
}
```

[82]:
```
try:
    # Create a connection pool
    pool_a = MySQLConnectionPool(pool_name="pool_a",
                                         pool_size=2,
                                         **dbconfig)
    print("Connection pool created successfully.")

    connection = pool_a.get_connection()

    cursor = connection.cursor()

except Error as e:
    print(f"Error creating connection pool: {e}")
```

```
Connection pool created successfully.
```

[83]:
```
peak_hours_sp = """
    CREATE PROCEDURE PeakHours()
    BEGIN
        SELECT HOUR(BookingSlot) AS booking_hour, COUNT(*) AS number_of_bookings
        FROM Bookings
        GROUP BY booking_hour
        ORDER BY number_of_bookings DESC;
    END;
    """
```

[84]:
```
try:
    # Step two: Run the stored procedure query
    cursor.execute(peak_hours_sp)
    # Step three: Invoke callproc to call the stored procedure
    cursor.callproc('PeakHours')
    # Step four: Fetch the results into a variable called dataset
    dataset = []
    for result in cursor.stored_results():
        dataset = result.fetchall()

    # Step five: Extract the names of the columns
    column_names = [column[0] for column in result.description]

    # Step six: Print the names of the columns
    print("Column Names:", column_names)

    # Step seven: Print the sorted data using a for loop
    for row in dataset:
        print(row)

except Error as e:
    print(f"Error: {e}")
```

```
Column Names: ['booking_hour', 'number_of_bookings']
(19, 2)
(15, 1)
(17, 1)
(18, 1)
(20, 1)
```

[85]:
```
guest_status_sp = """
    CREATE PROCEDURE GuestStatus()
```

```
    BEGIN
        SELECT
            CONCAT(b.GuestFirstName, ' ', b.GuestLastName) AS GuestName,
            CASE
                WHEN e.Role IN ('Manager', 'Assistant Manager') THEN 'Ready to pay'
                WHEN e.Role = 'Head Chef' THEN 'Ready to serve'
                WHEN e.Role = 'Assistant Chef' THEN 'Preparing Order'
                WHEN e.Role = 'Head Waiter' THEN 'Order served'
                ELSE 'Unknown Status'
            END AS OrderStatus
        FROM
            Bookings b
        LEFT JOIN
            Employees e ON b.EmployeeID = e.EmployeeID;
    END;
    """
```

[86]:
```
try:
    # Step five: Run the stored procedure query by invoking execute module on the cursor.
    cursor.execute(guest_status_sp)

    # Step six: Invoke callproc to call the stored procedure
    cursor.callproc('GuestStatus')

    # Step seven: Fetch the results into a variable called dataset
    dataset = []
    for result in cursor.stored_results():
        dataset = result.fetchall()

    # Step eight: Extract the names of the columns
    column_names = [column[0] for column in result.description]

    # Step nine: Print the names of the columns
    print("Column Names:", column_names)

    # Step ten: Print the sorted data using a for loop
    for row in dataset:
        print(row)

except Error as e:
    print(f"Error: {e}")

finally:
    # Step eleven: Close the cursor and connection when done
    if cursor:
        cursor.close()
    if connection:
        connection.close()  # This returns the connection to the pool
```

```
Column Names: ['GuestName', 'OrderStatus']
('Anna Iversen', 'Ready to pay')
('Joakim Iversen', 'Ready to pay')
('Vanessa McCarthy', 'Ready to serve')
('Marcos Romero', 'Preparing Order')
('Hiroki Yamane', 'Ready to pay')
('Diana Pinto', 'Order served')
```

[87]:
```
try:
    # Establish a connection pool named pool_b with two connections
    pool_b = pooling.MySQLConnectionPool(
        pool_name="pool_b",
        pool_size=2,  # Define the number of connections in the pool
        **dbconfig  # Unpack dbconfig to pass the configurations
    )
    print("Connection pool 'pool_b' created successfully.")

except Error as e:
    print(f"Error: Could not create connection pool 'pool_b'. Details: {e}")
```

```
Connection pool 'pool_b' created successfully.
```

[89]:
```
# Data for each guest booking
guest_bookings = [
    (8, "Anees", "Java", "18:00", 6),  # Guest 1
    (5, "Bald", "Vin", "19:00", 6),    # Guest 2
    (12, "Jay", "Kon", "19:30", 6)     # Guest 3
]
```

[90]:
```
# Store connections in a list to manage them dynamically
connections = []
```

[91]:
```
try:
    for i, booking in enumerate(guest_bookings):
        try:
            # Attempt to get a connection from the pool
            connection = pool_b.get_connection()
            connections.append(connection)  # Store the connection for later use

            # Create a cursor and insert the booking
            cursor = connection.cursor()
            insert_booking_query = """
            INSERT INTO Bookings (TableNo, GuestFirstName, GuestLastName, BookingSlot, EmployeeID)
            VALUES (%s, %s, %s, %s, %s)
            """
            cursor.execute(insert_booking_query, booking)
            connection.commit()  # Commit the transaction

            print(f"Booking for Guest {i + 1} added successfully.")
            cursor.close()  # Close the cursor after the transaction

        except PoolError as e:
            print(f"PoolError: Could not obtain a connection for Guest {i + 1}. Details: {e}")
            break  # Exit loop if no more connections are available

except Error as e:
    print("Error: Could not complete booking. Details:", e)

finally:
    # Return each connection to the pool safely
    for i, connection in enumerate(connections):
        try:
            connection.close()
            print(f"Connection {i + 1} returned to the pool.")
        except PoolError as e:
            print("Error: Attempted to return too many connections to the pool. Details:", e)
```

```
Booking for Guest 1 added successfully.
Booking for Guest 2 added successfully.
PoolError: Could not obtain a connection for Guest 3. Details: Failed getting connection; pool exhausted
Connection 1 returned to the pool.
Connection 2 returned to the pool.
```

[92]:
```
try:
    # Obtain a connection from the pool
    connection = pool_b.get_connection()
    cursor = connection.cursor()
```

```
except Error as e:
    print("Error while fetching data:", e)
```

[96]:
```
# Query 1: The name and EmployeeID of the Little Lemon manager
manager_query = """
SELECT name AS ManagerName, EmployeeID
FROM Employees
WHERE Role = 'Manager';
"""
cursor.execute(manager_query)
manager_data = cursor.fetchone()
print("Manager Name and EmployeeID:", manager_data)
```

```
Manager Name and EmployeeID: ('Mario Gollini', 1)
```

[98]:
```
# Query 2: The name and role of the employee who receives the highest salary
highest_salary_query = """
SELECT name AS EmployeeName, Role
FROM Employees
ORDER BY Annual_Salary DESC
LIMIT 1;
"""
cursor.execute(highest_salary_query)
highest_salary_data = cursor.fetchone()
print("Employee with Highest Salary:", highest_salary_data)
```

```
Employee with Highest Salary: ('Mario Gollini', 'Manager')
```

[99]:
```
# Query 3: Number of guests booked between 18:00 and 20:00
guest_count_query = """
SELECT COUNT(*) AS GuestCount
FROM Bookings
WHERE BookingSlot BETWEEN '18:00:00' AND '20:00:00';
"""
cursor.execute(guest_count_query)
guest_count_data = cursor.fetchone()
print("Number of guests booked between 18:00 and 20:00:", guest_count_data[0])
```

```
Number of guests booked between 18:00 and 20:00: 6
```

[101]:
```
# Query 4: Full name and BookingID of all guests waiting to be seated with the receptionist
# Sorted by BookingSlot
waiting_guests_query = """
SELECT CONCAT(GuestFirstName, ' ', GuestLastName) AS GuestName, BookingID
FROM Bookings
ORDER BY BookingSlot;
"""
cursor.execute(waiting_guests_query)
waiting_guests_data = cursor.fetchall()
print("Guests waiting to be seated (sorted by BookingSlot):")
for guest in waiting_guests_data:
    print(guest)
```

```
Guests waiting to be seated (sorted by BookingSlot):
('Vanessa McCarthy', 3)
('Marcos Romero', 4)
('Anees Java', 7)
('Hiroki Yamane', 5)
('Anna Iversen', 1)
('Joakim Iversen', 2)
('Bald Vin', 8)
('Diana Pinto', 6)
```

[104]:
```
basic_sales_report_sp = """
    CREATE PROCEDURE BasicSalesReport()
    BEGIN
        SELECT
            SUM(Quantity) AS TotalSales,
            AVG(Quantity) AS AverageSale,
            MIN(BillAmount) AS MinimumBill,
            MAX(BillAmount) AS MaximumBill
        FROM Orders;
    END;
"""
```

[105]:
```
try:
    # Step 1: Obtain a connection from the pool
    connection = pool_b.get_connection()

    # Step 2: Create a buffered cursor
    cursor = connection.cursor(buffered=True)

    # Step 3: Combine data from Bookings and Employees tables
    upcoming_bookings_query = """
    SELECT
        b.BookingSlot,
        CONCAT(b.GuestFirstName, ' ', b.GuestLastName) AS GuestName,
        e.name AS EmployeeName,
        e.Role AS EmployeeRole
    FROM
        Bookings b
    JOIN
        Employees e ON b.EmployeeID = e.EmployeeID
    ORDER BY
        b.BookingSlot ASC
    LIMIT 3;
    """

    cursor.execute(upcoming_bookings_query)
    upcoming_bookings = cursor.fetchall()

    # Step 4: Display the first three upcoming bookings
    print("Upcoming Bookings:")
    for booking in upcoming_bookings:
        booking_slot, guest_name, employee_name, employee_role = booking
        print(f"[{booking_slot}]\n[Guest_name: {guest_name}]\n[Assigned to: {employee_name} [{employee_role}]]\n")

except Error as e:
    print("Error while fetching upcoming bookings:", e)

finally:
    # Step 5: Close the cursor and return the connection to the pool
    if cursor:
        cursor.close()
    if connection:
        connection.close()
```

```
Upcoming Bookings:
[15:00:00]
[Guest_name: Vanessa McCarthy]
[Assigned to: Giorgos Dioudis [Head Chef]]

[17:30:00]
[Guest_name: Marcos Romero]
[Assigned to: Fatma Kaya [Assistant Chef]]

[18:00:00]
[Guest_name: Anees Java]
[Assigned to: John Millar [Receptionist]]
```

[Assigned to: John Millar (Receptionist)]

[ ]: