In [31]:

```
# Importing the MySQL Connector/Python
import mysql.connector as connector

# Establishing connection between Python and MySQL database via connector API
connection=connector.connect(
                             user="root",
                             password="Atdimpley$01",
                            )

# Creating a cursor object to communicate with entire MySQL database
cursor = connection.cursor()

# If exist, drop the database first, and create again
try:
    cursor.execute("CREATE DATABASE little_lemon")
except:
    cursor.execute("drop database little_lemon")
    cursor.execute("CREATE DATABASE little_lemon")
print("The database little_lemon is created.\n")

# Set little_lemon database for use
cursor.execute("USE little_lemon")
print("The database little_lemon is set for use.\n")

# The SQL query for MenuItems table is:
create_menuitem_table="""
CREATE TABLE MenuItems (
ItemID INT AUTO_INCREMENT,
Name VARCHAR(200),
Type VARCHAR(100),
Price INT,
PRIMARY KEY (ItemID)
);"""

# Create MenuItems table
cursor.execute(create_menuitem_table)
print("MenuItems table is created.\n")

# The SQL query for Menu table is:
create_menu_table="""
CREATE TABLE Menus (
MenuID INT,
ItemID INT,
Cuisine VARCHAR(100),
PRIMARY KEY (MenuID,ItemID)
);"""

# Create Menu table
cursor.execute(create_menu_table)
print("Menu table is created.\n")

# The SQL query for Bookings table is:
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)
);"""

# Create Bookings table
cursor.execute(create_booking_table)
print("Bookings table is created.\n")

# The SQL query for Orders table is:
create_orders_table="""
CREATE TABLE Orders (
OrderID INT,
TableNo INT,
MenuID INT,
BookingID INT,
BillAmount INT,
Quantity INT,
PRIMARY KEY (OrderID,TableNo)
);"""

# Create Orders table
cursor.execute(create_orders_table)
print("Orders table is created.\n")

# Create Employee table
create_employees_table="""
CREATE TABLE Employees (
EmployeeID INT AUTO_INCREMENT PRIMARY KEY,
```

```
Name VARCHAR(200),
Role VARCHAR(200),
Address VARCHAR(200),
Contact_Number INT,
Email VARCHAR(200),
Annual_Salary VARCHAR(200)
);"""

# Create Employee table
cursor.execute(create_employees_table)
print("Employees table is created.\n")

# Confirm if the tables are created
print("Following tables are created in the little_lemon database.\n")
cursor.execute("SHOW TABLES")
for table in cursor:
    print(table)
```

```
The database little_lemon is created.

The database little_lemon is set for use.

MenuItems table is created.

Menu table is created.

Bookings table is created.

Orders table is created.

Employees table is created.

Following tables are created in the little_lemon database.

('bookings',)
('employees',)
('menuitems',)
('menus',)
('orders',)
```

In [32]:

```
# Inserting query to populate "MenuItems" table

insert_menuitmes="""
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);"""


# Inserting 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');"""


# Inserting 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);"""

# 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);"""

# 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
(02,'Adrian Gollini','Assistant Manager','334, Dill Square, Lincoln Park, Chicago, IL',351474048,'Adrian.g@little
(03,'Giorgos Dioudis','Head Chef','879 Sage Street, West Loop, Chicago, IL',351970582,'Giorgos.d@littlelemon.com'
(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','
(06,'John Millar','Receptionist','245 Dill Square, Lincoln Park, Chicago, IL',351584508,'John.m@littlelemon.com',


print("Inserting data in MenuItems table.")
# Populating MenuItems table
cursor.execute(insert_menuitmes)
print("Total number of rows in MenuItem table: ", cursor.rowcount)
connection.commit()

print("Inserting data in Menus table.")
# Populating MenuItems table
cursor.execute(insert_menu)
print("Total number of rows in Menu table: ", cursor.rowcount)
connection.commit()

print("Inserting data in Bookings table.")
# Populating Bookings table
cursor.execute(insert_bookings)
print("Total number of rows in Bookings table: ", cursor.rowcount)
connection.commit()

print("Inserting data in Orders table.")
# Populating Orders table
cursor.execute(insert_orders)
print("Total number of rows in Orders table: ", cursor.rowcount)
connection.commit()

print("Inserting data in Employees table.")
# Populating Employees table
cursor.execute(insert_employees)
print("Total number of rows in Employees table: ", cursor.rowcount)
connection.commit()
```

```
Inserting data in MenuItems table.
Total number of rows in MenuItem table:  17
Inserting data in Menus table.
Total number of rows in Menu table:  12
Inserting data in Bookings table.
Total number of rows in Bookings table:  6
Inserting data in Orders table.
Total number of rows in Orders table:  5
Inserting data in Employees table.
Total number of rows in Employees table:  6
```

In [33]:
```
from mysql.connector.pooling import MySQLConnectionPool
from mysql.connector import Error
```

In [34]:
```
dbconfig = {

    "database":"little_lemon",
    "user":"root",
    "password":"Atdimpley$01"
}
```

In [35]:
```
try:
    pool = MySQLConnectionPool(pool_name = "pool_b",
                               pool_size = 2,
                               **dbconfig)
    print("The connection pool is created with a name:", pool.pool_name)
    print("The pool size is:", pool.pool_size)

except Error as er:
    print("Error code:", er.errno)
    print("Error message:", er.msg)
```

```
The connection pool is created with a name: pool_b
The pool size is: 2
```

In [36]:
```
try:
    # Get the first connection from the pool
    conn1 = pool.get_connection()
    if conn1.is_connected():
        print("Connection 1 established")

        # Perform the insertion for Guest 1
        cursor1 = conn1.cursor()
        query1 = "INSERT INTO Bookings (TableNo, GuestFirstName, GuestLastName, BookingSlot, EmployeeID) VALUES (
        data1 = (8, 'Anees', 'Java', '18:00', 6)
        cursor1.execute(query1, data1)
        conn1.commit()
        cursor1.close()

    # Get the second connection from the pool
    conn2 = pool.get_connection()
    if conn2.is_connected():
        print("Connection 2 established")

        # Perform the insertion for Guest 2
        cursor2 = conn2.cursor()
        query2 = "INSERT INTO Bookings (TableNo, GuestFirstName, GuestLastName, BookingSlot, EmployeeID) VALUES (
        data2 = (5, 'Bald', 'Vin', '19:00', 6)
        cursor2.execute(query2, data2)
        conn2.commit()
        cursor2.close()

except Error as e:
    print("Error occurred:", e)
```

```
Connection 1 established
Connection 2 established
```

In [37]:
```
# Create a connection
connection=connector.connect(user="root",password="Atdimpley$01")
# Add the connection into the pool
pool.add_connection(cnx=connection)
print("A new connection is added in the pool.\n")
```

```
A new connection is added in the pool.
```

In [38]:
```
try:
    # Get the third connection from the pool
    conn3 = pool.get_connection()
    if conn3.is_connected():
        print("Connection 3 established")

        # Perform the insertion for Guest 3
        cursor3 = conn3.cursor()
        query3 = "INSERT INTO Bookings (TableNo, GuestFirstName, GuestLastName, BookingSlot, EmployeeID) VALUES (
        data3 = (12, 'Jay', 'Kon', '19:30', 6)
        cursor3.execute(query3, data3)
        conn3.commit()
        cursor3.close()
except Error as e:
    print("Error occurred:", e)
```

```
Connection 3 established
```

What are the name and EmployeeID of the Little Lemon manager?

In [39]:
```
# Create a connection
```

```
connection=connector.connect(user="root",password="Atdimpley$01")
# Add the connection into the pool
pool.add_connection(cnx=connection)
print("A new connection is added in the pool.\n")


try:
    cursor = pool.get_connection().cursor()

    query = """SELECT Name, EmployeeID FROM employees WHERE Role = 'Manager'"""

    cursor.execute(query)
    result = cursor.fetchone()

    if result:
        manager_name, manager_employee_id = result
        print("Little Lemon Manager:")
        print("Name:", manager_name)
        print("EmployeeID:", manager_employee_id)
    else:
        print("No Little Lemon manager found.")

    cursor.close()
except Error as e:
    print("Error while retrieving Little Lemon manager information:", e)
```

```
A new connection is added in the pool.

Little Lemon Manager:
Name: Mario Gollini
EmployeeID: 1
```

What are the name and role of the employee who receives the highest salary?

In [40]:

```
# Create a connection
connection=connector.connect(user="root",password="Atdimpley$01")
# Add the connection into the pool
pool.add_connection(cnx=connection)
print("A new connection is added in the pool.\n")


try:
    cursor = pool.get_connection().cursor()

    query = """SELECT Name, Role FROM Employees ORDER BY Annual_Salary DESC LIMIT 1"""

    cursor.execute(query)
    result = cursor.fetchone()

    if result:
        highest_salary_employee_name, highest_salary_employee_role = result
        print("Employee with Highest Salary:")
        print("Name:", highest_salary_employee_name)
        print("Role:", highest_salary_employee_role)
    else:
        print("No employees found.")

    cursor.close()
except Error as e:
    print("Error while retrieving employee with highest Annual_salary information:", e)
```

```
A new connection is added in the pool.

Employee with Highest Salary:
Name: Mario Gollini
Role: Manager
```

What is the number of guests booked between 18:00 and 20:00?

In [41]:

```
# Create a connection
connection=connector.connect(user="root",password="Atdimpley$01")
# Add the connection into the pool
pool.add_connection(cnx=connection)
print("A new connection is added in the pool.\n")

try:
    cursor = pool.get_connection().cursor()

    query = "SELECT COUNT(*) FROM Bookings WHERE BookingSlot >= '18:00' AND BookingSlot <= '20:00'"
```

```
    cursor.execute(query)
    result = cursor.fetchone()

    if result:
        guests_booked_count = result[0]
        print("Number of Guests Booked between 18:00 and 20:00:", guests_booked_count)
    else:
        print("No guests found.")

    cursor.close()
except Error as e:
    print("Error while retrieving the number of guests booked between 18:00 and 20:00:", e)
```

```
A new connection is added in the pool.

Number of Guests Booked between 18:00 and 20:00: 7
```

The full names and BookingID of all guests waiting to be seated with the receptionist

In [42]:

```
# Create a connection
connection=connector.connect(user="root",password="Atdimpley$01")
# Add the connection into the pool
pool.add_connection(cnx=connection)
print("A new connection is added in the pool.\n")

try:
    cursor = pool.get_connection().cursor()

    query = "SELECT CONCAT(GuestFirstName, ' ', GuestLastName) AS FullName, BookingID FROM Bookings ORDER BY Book

    cursor.execute(query)
    results = cursor.fetchall()

    if results:
        print("Guests Waiting to be Seated:")
        for row in results:
            full_name, booking_id = row
            print("Full Name:", full_name)
            print("BookingID:", booking_id)
    else:
        print("No guests waiting to be seated.")

    cursor.close()
except Error as e:
    print("Error while retrieving guests waiting to be seated information:", e)
```

```
A new connection is added in the pool.

Guests Waiting to be Seated:
Full Name: Vanessa McCarthy
BookingID: 3
Full Name: Marcos Romero
BookingID: 4
Full Name: Anees Java
BookingID: 7
Full Name: Hiroki Yamane
BookingID: 5
Full Name: Anna Iversen
BookingID: 1
Full Name: Joakim Iversen
BookingID: 2
Full Name: Bald Vin
BookingID: 8
Full Name: Jay Kon
BookingID: 9
Full Name: Diana Pinto
BookingID: 6
```

Stored procedure for the Total and Average sales; Minimum and Maximum bills paid

In [43]:

```
# Create a connection
connection=connector.connect(user="root",password="Atdimpley$01")
# Add the connection into the pool
pool.add_connection(cnx=connection)
print("A new connection is added in the pool.\n")
```

```
cursor = pool.get_connection().cursor()
cursor.execute("DROP PROCEDURE IF EXISTS BasicSalesReport;")
stored_procedure_query="""

CREATE PROCEDURE BasicSalesReport()
BEGIN
    DECLARE total_sales DECIMAL(10, 2);
    DECLARE avg_sale DECIMAL(10, 2);
    DECLARE min_bill DECIMAL(10, 2);
    DECLARE max_bill DECIMAL(10, 2);

    SELECT SUM(BillAmount) INTO total_sales FROM Orders;
    SELECT AVG(BillAmount) INTO avg_sale FROM Orders;
    SELECT MIN(BillAmount) INTO min_bill FROM Orders;
    SELECT MAX(BillAmount) INTO max_bill FROM Orders;

    SELECT total_sales AS TotalSales, avg_sale AS AverageSale, min_bill AS MinimumBillPaid, max_bill AS MaximumBi
END;

"""

# Execute the query
cursor.execute(stored_procedure_query)

#*****************************************#

# Call the stored procedure with its name
cursor.callproc("BasicSalesReport")

# Retrieve recrods in "dataset"
results = next(cursor.stored_results())
dataset = results.fetchall()

# Retrieve column names using list comprehension in a 'for' loop
for column_id in cursor.stored_results():
    columns = [column[0] for column in column_id.description]

# Print column names
print(columns)

# Print data
for data in dataset:
    print(data)
```

```
A new connection is added in the pool.

['TotalSales', 'AverageSale', 'MinimumBillPaid', 'MaximumBillPaid']
(Decimal('243.00'), Decimal('48.60'), Decimal('37.00'), Decimal('86.00'))
```

What are the next three upcoming bookings to be displayed on the kitchen screen?

In [44]:

```
# Create a connection
connection=connector.connect(user="root",password="Atdimpley$01")
# Add the connection into the pool
pool.add_connection(cnx=connection)
print("A new connection is added in the pool.\n")

try:
    conn = pool.get_connection()
    cursor = conn.cursor(buffered=True)

    query = """
    SELECT b.BookingID, b.GuestFirstName, b.GuestLastName, b.BookingSlot, e.Name AS EmployeeName
    FROM Bookings b
    JOIN Employees e ON b.EmployeeID = e.EmployeeID

    ORDER BY b.BookingSlot ASC
    LIMIT 3
    """

    cursor.execute(query)
    results = cursor.fetchall()

    if results:
        print("Upcoming Bookings:")
        for row in results:
            booking_id, first_name, last_name, booking_time, employee_name = row
            print("Booking ID:", booking_id)
            print("Guest Name:", first_name, last_name)
            print("Booking Time:", booking_time)
            print("Employee Name:", employee_name)
            print("-----------")
```

```
    else:
        print("No upcoming bookings found.")

    cursor.close()
    conn.close()
except Error as e:
    print("Error while retrieving upcoming bookings:", e)
```

```
A new connection is added in the pool.

Upcoming Bookings:
Booking ID: 3
Guest Name: Vanessa McCarthy
Booking Time: 15:00:00
Employee Name: Giorgos Dioudis
-----------
Booking ID: 4
Guest Name: Marcos Romero
Booking Time: 17:30:00
Employee Name: Fatma Kaya
-----------
Booking ID: 7
Guest Name: Anees Java
Booking Time: 18:00:00
Employee Name: John Millar
-----------
```