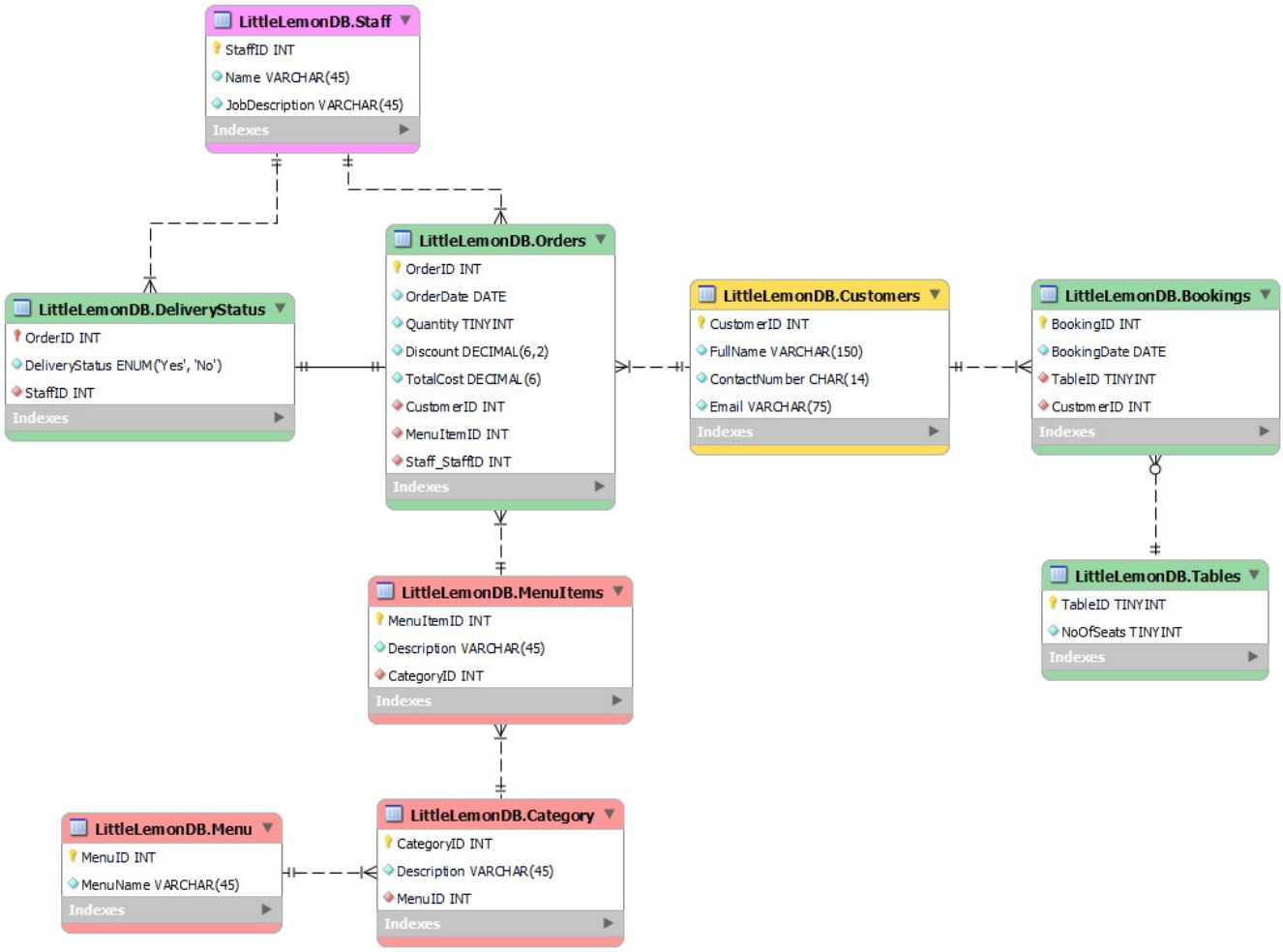
Meta Database Engineer – Capstone

Report



The following ERD show en00es that include all a6ributes of the spreadsheet. Also, all en0tles are normalized to the third normal form (NF-3) as detailed in the following:

1) Entities don't have any attributes with multi-valued domains. All domains have atomic domains. Also, attributes don't have any repea0ng groups. Accordingly, entities are in conformance with NF-1 requirements.
2) All non-key a6ributes in each table are not functionally dependent on part of the Primary Key. And because all tables are already in conformance with NF-1, they are accordingly in conformance with NF-2 (partial functional dependency is eliminated)
3) All non-key a6ributes in each table are not functionally dependent on another non-key attributes and dependent on the Primary Key. And because all tables are already in conformance with NF-2, they are accordingly in conformance with NF-3 (transitive functional dependency is eliminated)

Conclusion: the database is in conformance with the requirements of NF-3

Limit to 1000 rows

```sql
-- -----------------------------------------------------
-- Schema LittleLemonDB
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS LittleLemonDB ;
USE LittleLemonDB ;

-- -----------------------------------------------------
-- Table `LittleLemonDB`.`Tables`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS LittleLemonDB.Table (
  TableID TINYINT NOT NULL,
  NoOfSeats TINYINT NOT NULL,
  PRIMARY KEY (TableID));

-- -----------------------------------------------------
-- Table LittleLemonDB.Customers
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS LittleLemonDB.Customers (
  CustomerID INT NOT NULL AUTO_INCREMENT,
  FullName VARCHAR(150) NOT NULL,
  ContactNumber CHAR(14) NOT NULL,
  Email VARCHAR(75) NOT NULL,
  PRIMARY KEY (CustomerID));

-- -----------------------------------------------------
-- Table LittleLemonDB.Bookings
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS LittleLemonDB.Bookings (
  BookingID INT NOT NULL AUTO_INCREMENT,
  BookingDate DATE NOT NULL,
  TableID TINYINT NOT NULL,
  CustomerID INT NOT NULL,
  PRIMARY KEY (BookingID),
  CONSTRAINT fk_Bookings_Tables
    FOREIGN KEY (TableID)
    REFERENCES LittleLemonDB.Tables (TableID)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT fk_Bookings_Customers1
    FOREIGN KEY (CustomerID)
    REFERENCES LittleLemonDB.Customers (CustomerID)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

```sql
-- -----------------------------------------------------
-- Table LittleLemonDB.Menu
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS LittleLemonDB.Menu (
  MenuID INT NOT NULL AUTO_INCREMENT,
  MenuName VARCHAR(45) NOT NULL,
  PRIMARY KEY (MenuID));

-- -----------------------------------------------------
-- Table LittleLemonDB.Category
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS LittleLemonDB.Category (
  CategoryID INT NOT NULL AUTO_INCREMENT,
  Description VARCHAR(45) NOT NULL,
  MenuID INT NOT NULL,
  PRIMARY KEY (CategoryID),
  CONSTRAINT fk_Category_Menu1
    FOREIGN KEY (MenuID)
    REFERENCES LittleLemonDB.Menu (MenuID)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);

-- -----------------------------------------------------
-- Table LittleLemonDB.MenuItems
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS LittleLemonDB.MenuItems (
  MenuItemID INT NOT NULL AUTO_INCREMENT,
  CategoryID INT NOT NULL,
  PRIMARY KEY (MenuItemID),
  CONSTRAINT fk_MenuItems_Category
    FOREIGN KEY (CategoryID)
    REFERENCES LittleLemonDB.Category (CategoryID)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

```sql

-- -----------------------------------------------------
-- Table LittleLemonDB.Staff
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS LittleLemonDB.Staff (
  StaffID INT NOT NULL AUTO_INCREMENT,
  Name VARCHAR(45) NOT NULL,
  JobDescription VARCHAR(45) NOT NULL,
  PRIMARY KEY (StaffID));

-- -----------------------------------------------------
-- Table LittleLemonDB.Orders
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS LittleLemonDB.Orders (
  OrderID INT NOT NULL AUTO_INCREMENT,
  OrderDate DATE NOT NULL,
  Quantity TINYINT NOT NULL,
  Discount DECIMAL(6,2) NOT NULL,
  TotalCost DECIMAL(6) NOT NULL,
  CustomerID INT NOT NULL,
  MenuItemID INT NOT NULL,
  Staff_StaffID INT NOT NULL,
  PRIMARY KEY (OrderID),
  CONSTRAINT fk_Orders_Customers
    FOREIGN KEY (CustomerID)
    REFERENCES LittleLemonDB.Customers (CustomerID)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT fk_Orders_MenuItems
    FOREIGN KEY (MenuItemID)
    REFERENCES LittleLemonDB.MenuItems (MenuItemID)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT fk_Orders_Staff
    FOREIGN KEY (Staff_StaffID)
    REFERENCES LittleLemonDB.Staff (StaffID)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);

```

```sql
-- -----------------------------------------------------
-- Table LittleLemonDB.DeliveryStatus
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS LittleLemonDB.DeliveryStatus (
  OrderID INT NOT NULL,
  DeliveryStatus ENUM('Yes', 'No') NOT NULL,
  StaffID INT NOT NULL,
  PRIMARY KEY (OrderID),
  CONSTRAINT fk_DeliveryStatus_Orders
    FOREIGN KEY (OrderID)
    REFERENCES LittleLemonDB.Orders (OrderID)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT fk_DeliveryStatus_Staff
    FOREIGN KEY (StaffID)
    REFERENCES LittleLemonDB.Staff (StaffID)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

Data Insertion

```sql
-- Insert into Tables
INSERT INTO Tables (TableID, NoOfSeats) VALUES
  (1, 4),
  (2, 6),
  (3, 6),
  (4, 3),
  (5, 2),
  (6, 6),
  (20, 4),
  (21, 4);

-- Insert into Customers
INSERT INTO Customers (CustomerID, FullName, ContactNumber, Email) VALUES
  (1, 'Alice Johnson', '+1 555-1234', 'alice.johnson@email.com'),
  (2, 'Michael Davis', '+1 555-5678', 'michael.davis@email.com'),
  (3, 'Emily Smith', '+1 555-9876', 'emily.smith@email.com'),
  (4, 'Joshua Miller', '+1 555-4321', 'joshua.miller@email.com'),
  (5, 'Sophia White', '+1 555-8765', 'sophia.white@email.com'),
  (15, 'Scott Roseworn', '1 905 664 9876', 'scott.roseworn@cmail.com'),
  (16, 'Jon Stewart', '669 654 5841', 'jon.stewart@cmail.com');

-- Insert into Bookings
INSERT INTO Bookings (BookingDate, TableID, CustomerID) VALUES
  ('2022-10-10', 1, 1),  -- BookingID: 1, TableNumber: 5, CustomerID: 1
  ('2022-11-12', 2, 3),  -- BookingID: 2, TableNumber: 3, CustomerID: 3
  ('2022-10-11', 3, 2),  -- BookingID: 3, TableNumber: 2, CustomerID: 2
  ('2022-10-13', 4, 1),  -- BookingID: 4, TableNumber: 2, CustomerID: 1
  ('2023-06-17', 5, 1),  -- BookingID: 5, TableNumber: 6, CustomerID: 1
  ('2023-12-18', 6, 15), -- BookingID: 7, TableNumber: 20, CustomerID: 15
  ('2023-12-29', 7, 16); -- BookingID: 8, TableNumber: 21, CustomerID: 16

-- Insert into Menus
INSERT INTO menus (MenuID, MenuName) VALUES
  (1, 'Moussaka'),
  (2, 'Manti');
```

```sql

-- Insert into Categories
INSERT INTO Categories (Description, MenuID) VALUES
  ('Appetizers_1', 1),    -- CategoryID: 1, MenuID: 1
  ('Main Course_1', 2),    -- CategoryID: 2, MenuID: 2
  ('Desserts_1', 1),       -- CategoryID: 3, MenuID: 1
  ('Drinks_1', 1),         -- CategoryID: 4, MenuID: 1
  ('Appetizers_2', 2),     -- CategoryID: 5, MenuID: 2
  ('Main Course_2', 2),    -- CategoryID: 6, MenuID: 2
  ('Desserts_2', 2),       -- CategoryID: 7, MenuID: 2
  ('Drinks_2', 2);         -- CategoryID: 8, MenuID: 2

-- Insert into MenuItems
INSERT INTO MenuItems (MenuItemID, Description, CategoryID) VALUES
  (1, 'Green Salad Skewers', 1),
  (2, 'Hummus with Pita Bread', 1),
  (3, 'Spanakopita (Spinach Pie)', 1),
  (4, 'Grilled Lemon Herb Chicken', 2),
  (5, 'Seafood Paella', 2),
  (6, 'Eggplant Moussaka', 2),
  (7, 'Baklava', 3),
  (8, 'Tiramisu', 3),
  (9, 'Orange Blossom Panna Cotta', 3),
  (10, 'Mint Lemonade', 4),
  (11, 'Pomegranate Mojito', 4),
  (12, 'Tzatziki Yogurt Smoothie', 4);

-- Insert into Orders
INSERT INTO Orders (OrderID, OrderDate, Quantity, Discount, TotalCost, CustomerID, MenuItemID, Staff_StaffID) VALUES
  (1, '2023-08-19', 4, 0, 150, 5, 3, 3),
  (2, '2023-10-14', 2, 0, 255, 3, 1, 1),
  (3, '2023-06-18', 5, 0, 250, 2, 3, 4),
  (4, '2023-06-19', 3, 0, 200, 4, 6, 4),
  (5, '2023-11-20', 2, 0, 260, 1, 2, 2);

-- Insert into Staff
INSERT INTO Staff (StaffID, Name, JobDescription) VALUES
  (1, 'John Smith', 'Manager'),
  (2, 'Joan', 'Staff'),
  (3, 'Dan', 'Staff'),
  (4, 'Ron', 'Shipper');
```

```sql
-- Insert into DeliveryStatus
INSERT INTO DeliveryStatus (OrderID, DeliveryStatus, StaffID) VALUES
  (3, 'Yes', 4),
  (4, 'Yes', 4);
```

Sales Report (Week 2)

Task #1

```
-- Task #1

CREATE OR REPLACE VIEW OrdersView AS
SELECT OrderID, Quantity, TotalCost
From Orders
WHERE Quantity > 2;
```

Output:

```
SELECT * FROM littlelemondb.ordersview;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Co

OrderID	Quantity	TotalCost
1	4	150
3	5	250
4	3	200

Task #2

```
-- Task #2

CREATE OR REPLACE VIEW orders_customers AS
SELECT c.CustomerID, c.FullName, o.Quantity, o.OrderID, o.TotalCost As Cost, o.MenuItemID
FROM Customers c JOIN Orders o ON c.CustomerID = o.CustomerID;

CREATE OR REPLACE VIEW menu_menuitems AS
SELECT i.MenuItemID, m.MenuName, i.Description AS CourseName
FROM  Category c
                JOIN MenuItems i ON c.CategoryID = i.CategoryID
                JOIN Menu m ON c.MenuID = m.MenuID;

SELECT o.CustomerID, o.FullName, o.OrderID, o.Cost, m.MenuName, m.CourseName
FROM orders_customers o JOIN menu_menuitems m ON o.MenuItemID = m.MenuItemID
WHERE o.Cost > 150 ;
```

Output:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CustomerID	FullName	OrderID	Cost	MenuName	CourseName
3	Emily Smith	2	255	Moussaka	Green Salad Skewers
2	Michael Davis	3	250	Moussaka	Spanakopita (Spinach Pie)
4	Joshua Miller	4	200	Manti	Eggplant Moussaka
1	Alice Johnson	5	260	Moussaka	Hummus with Pita Bread

Task #3

```
-- Task #3
-- Using subquery (as required)
SELECT MenuName
FROM  menu_menuitems
WHERE MenuItemID IN (
                     SELECT MenuItemID
                     FROM orders_customers
                     WHERE Quantity > 2);

-- Using JOIN (verification)
SELECT MenuName
FROM orders_customers o JOIN menu_menuitems i ON o.MenuItemID = i.MenuItemID
WHERE o.Quantity > 2;
```

Output:

MenuName
Moussaka
Manti

Week-2 Query Optimization

Task #1

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `GetMaxQuantity`()
BEGIN
    SELECT MAX(Quantity) AS 'Max Quantity in Order'
    FROM Orders;
END
```

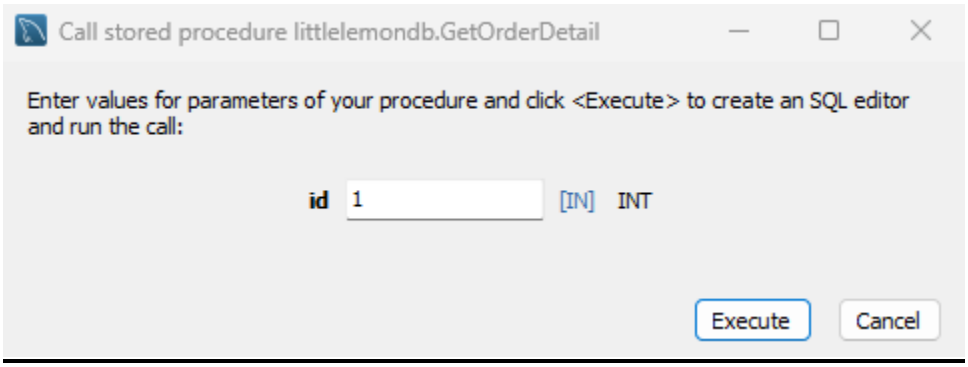
```
call littlelemondb.GetMaxQuantity();
```

Output:

	Max Quantity in Order
▶	5

Task #2

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `GetOrderDetail`(IN id INT)
BEGIN
    SELECT OrderID, Quantity, TotalCost AS Cost
    FROM Orders
    WHERE OrderID = id;
END
```



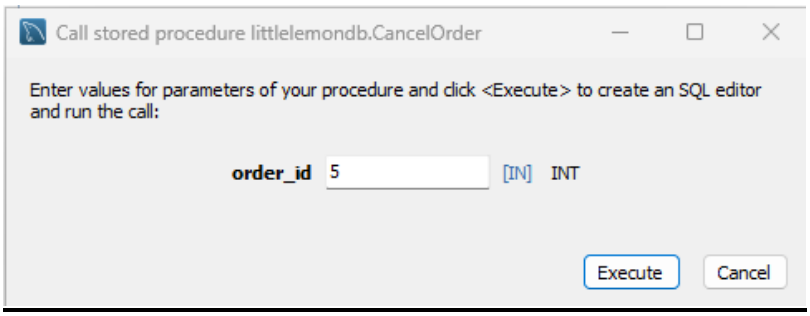

```
call littlelemondb.GetOrderDetail(1);
```

Output:

	OrderID	Quantity	Cost
▶	1	4	150

Task #3

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `CancelOrder`(IN order_id INT)
BEGIN
    DELETE FROM Orders
    WHERE OrderID = order_id;
    SELECT CONCAT("Order " , order_id , " is cancelled") AS "Confirmation";
END
```




```
call littlelemondb.CancelOrder(5);
```

	Confirmation
▶	Order 5 is cancelled

Table Booking System (WK-2):

Booking based on user input

Task #1

```sql
-- Task #1

INSERT INTO Bookings (BookingID, BookingDate, TableNumber, CustomerID)
VALUES
    (1, '2022-10-10', 5, 1),
    (2, '2022-11-12', 3, 3),
    (3, '2022-10-11', 2, 2),
    (4, '2022-10-13', 2, 1);
```

Limit to 1000 rows

```sql
SELECT * FROM littlelemondb.bookings;
```

Result Grid | Filter Rows: | Edit:

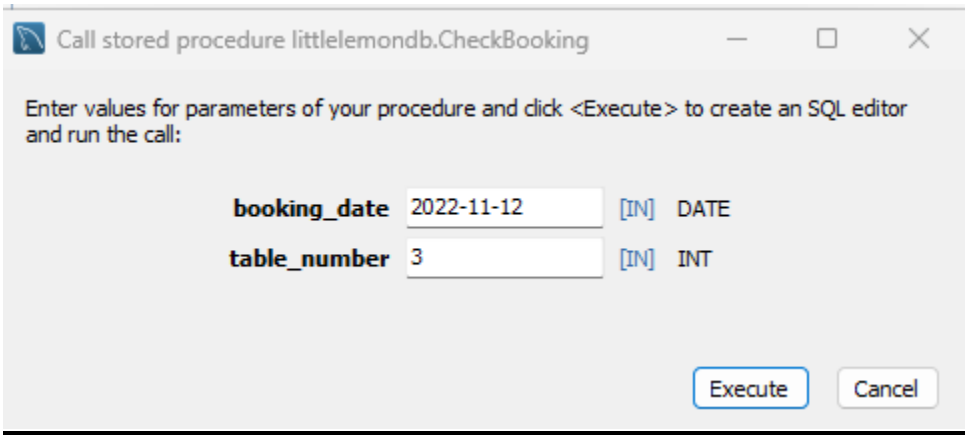
BookingID	BookingDate	TableNumber	CustomerID
1	2022-10-10	5	1
2	2022-11-12	3	3
3	2022-10-11	2	2
4	2022-10-13	2	1

Task #2

```sql
CREATE DEFINER=`root`@`localhost` PROCEDURE `CheckBooking`(
    IN booking_date DATE,
    IN table_number INT
)
BEGIN
    DECLARE bookedTable INT DEFAULT 0;
    SELECT COUNT(bookedTable)
        INTO bookedTable
        FROM Bookings WHERE BookingDate = booking_date and TableNumber = table_number;
        IF bookedTable > 0 THEN
            SELECT concat("Table ", table_number, " is already booked") AS "Booking status";
            ELSE
            SELECT concat("Table ", table_number, " has not been booked yet") AS "Booking status";
        END IF;
END
```

Limit to 1000 rows

```sql
call littlelemondb.CheckBooking('2022-11-12', 3);
```



	Booking status
▶	Table 3 is already booked

Task #3

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `AddValidBooking`(IN bookingDate DATE, IN tableNumber INT)
BEGIN
    DECLARE tableCount INT;

    -- Start a transaction
    START TRANSACTION;

    -- Check if the table is already booked on the given date
    SELECT COUNT(*) INTO tableCount
    FROM Bookings
    WHERE TableNumber = tableNumber AND BookingDate = bookingDate;

    IF tableCount > 0 THEN
        -- Table is already booked, rollback the transaction
        ROLLBACK;
        SELECT CONCAT('Table ', tableNumber, ' is already booked - booking cancelled') AS 'Booking status';
    ELSE
        -- Table is available, proceed with the booking
        INSERT INTO Bookings (BookingDate, TableNumber)
        VALUES (bookingDate, tableNumber);

        -- Commit the transaction
        COMMIT;
        SELECT 'Booking successful.' AS Message;
    END IF;
END
```

Call stored procedure littlelemondb.AddValidBooking

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

bookingDate	2022-12-17	[IN]	DATE
tableNumber	6	[IN]	INT

Execute Cancel

Limit to 1000 rows

```
1 •  call littlelemondb.AddValidBooking('2022-12-17', 6);
```

	Booking status
▶	Table 6 is already booked - booking cancelled

Create SQL queries to add and update bookings

Task #1

```sql
CREATE DEFINER=`root`@`localhost` PROCEDURE `AddBooking`(
    IN booking_date DATE,
    IN no_of_seats TINYINT,
    IN full_name VARCHAR(150),
    IN contact_number CHAR(14),
    IN email_ VARCHAR(75)
)
BEGIN
    DECLARE new_table_id INT;
    DECLARE new_customer_id INT;
    DECLARE new_booking_id INT;

    -- Insert into tables and retrieve the auto-generated ID
    INSERT INTO tables (NoOfSeats) VALUES (no_of_seats);
    SET new_table_id = last_insert_id();

    -- Insert into customers and retrieve the auto-generated ID
    INSERT INTO customers (FullName, ContactNumber, Email)
    VALUES (full_name, contact_number, email_);
    SET new_customer_id = last_insert_id();

    -- Insert into bookings and retrieve the auto-generated ID
    INSERT INTO bookings (BookingDate, TableID, CustomerID)
    VALUES (booking_date, new_table_id, new_customer_id);

    SET new_booking_id = LAST_INSERT_ID();

    SELECT 'New booking added with BookingID: ' AS 'Confirmation', new_booking_id;
END
```

Note that BookingID, and CustomerID are auto-generated

Call stored procedure littlelemondb.AddBooking

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

booking_date	2022-12-30	[IN]	DATE
no_of_seats	4	[IN]	TINYINT
full_name	Jon Stewart	[IN]	VARCHAR(150)
contact_number	905-667-2594	[IN]	CHAR(14)
email_	stewart@cmail.com	[IN]	VARCHAR(75)

Execute Cancel

Limit to 1000 rows

```
call littlelemondb.AddBooking('2022-12-30', 4, 'Jon Stewart', '905-667-2594', 'jon.stewart@cmail.com');
```

	Confirmation	new_booking_id
▶	New booking added with BookingID:	9

Task #2

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `UpdateBooking`(IN booking_id INT, IN booking_date DATE)
BEGIN
    UPDATE Bookings
    SET BookingDate = booking_date
    WHERE BookingID = booking_id;
    SELECT CONCAT("Booking ", booking_id, " updated") AS "Confirmation";
END
```

Call stored procedure littlelemondb.UpdateBooking

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

booking_id	9	[IN]	INT
booking_date	2022-12-17	[IN]	DATE

Execute Cancel

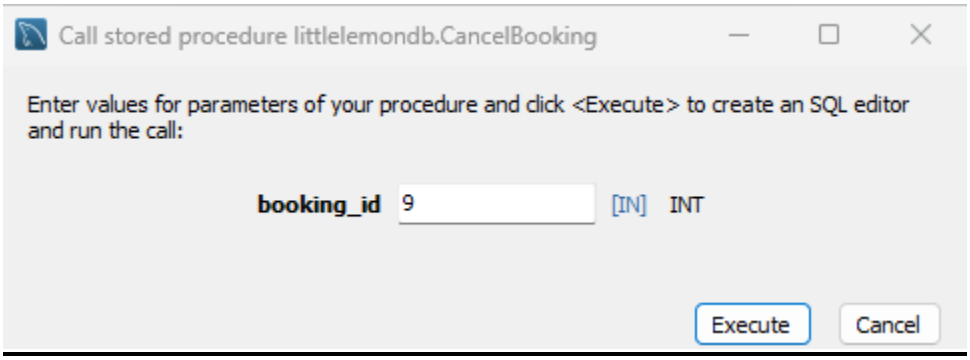
Limit to 1000 rows

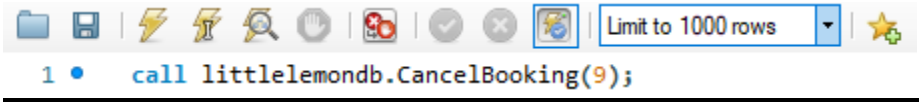
```
call littlelemondb.UpdateBooking(9, '2022-12-17');
```

	Confirmation
▶	Booking 9 updated

Task 3

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `CancelBooking`(IN booking_id INT)
BEGIN
    -- DELETE FROM Bookings
    -- WHERE BookingID = booking_id;
    SELECT CONCAT("Booking " , booking_id , " cancelled") AS "Confirmation";
END
```



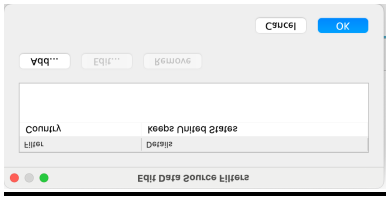



```
call littlelemondb.CancelBooking(9);
```

	Confirmation
▶	Booking 9 cancelled

Data Visualization

Task #1



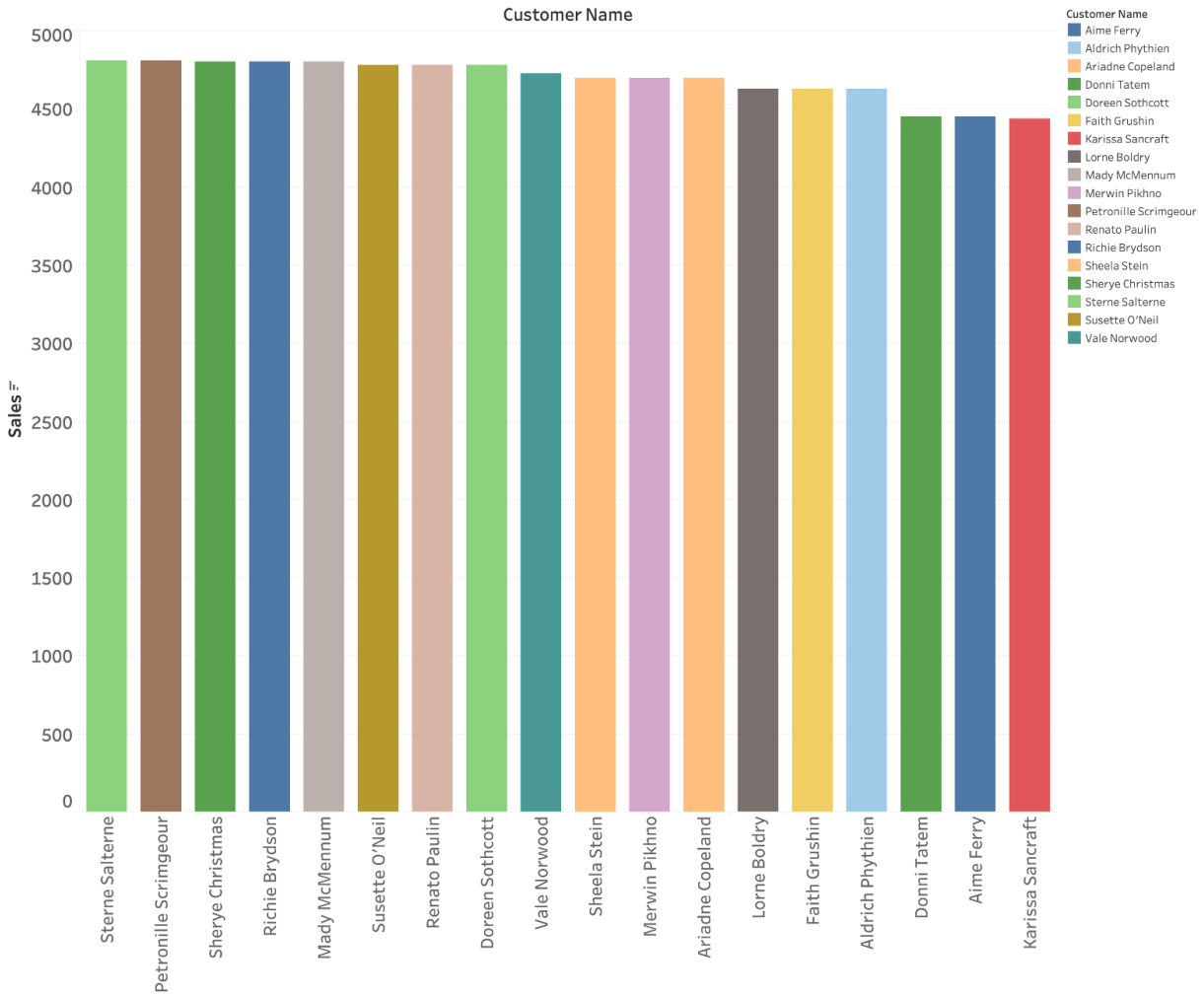
Task #2

Abc Calculation First Name	Abc Calculation Last Name
Sterne	Salterne
Mady	McMennum
Merwin	Pikhno
Aime	Ferry
Merlina	Lermit
Schuyler	Walewski
Ariadne	Copeland
Tamma	Clink
Susette	O'Neil
Aldrich	Phythien
Dusty	Yedall
Roanna	Safont
Richie	Brydson
A Richie	Chasmer
Renato	Paulin

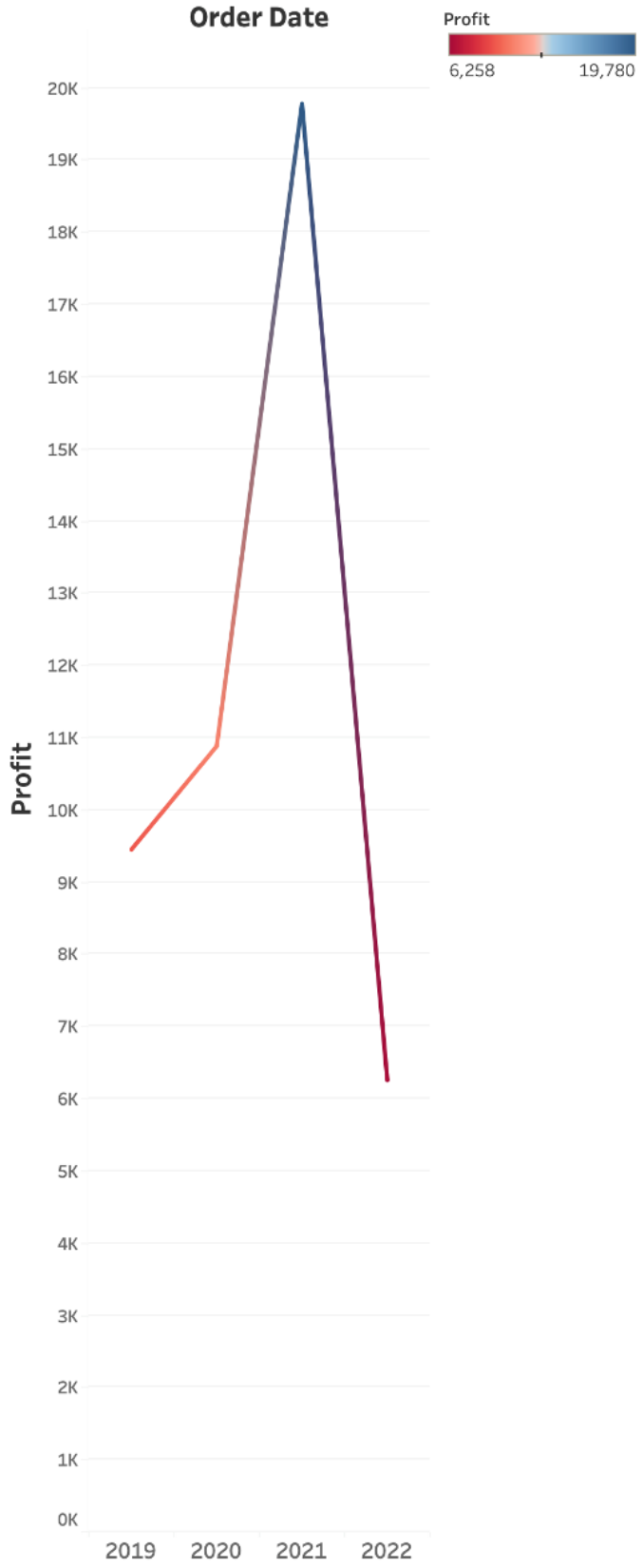
Task #3

# Orders Sales	# Calculation Profit
285.000	95.000
216.135	72.045
187.500	62.500
137.760	45.920
352.500	117.500
179.700	59.900
187.500	62.500
352.500	117.500
480.000	160.000
78.120	26.040
352.500	117.500
315.000	105.000
216.135	72.045
199.755	66.585
480.000	160.000

Customers sales

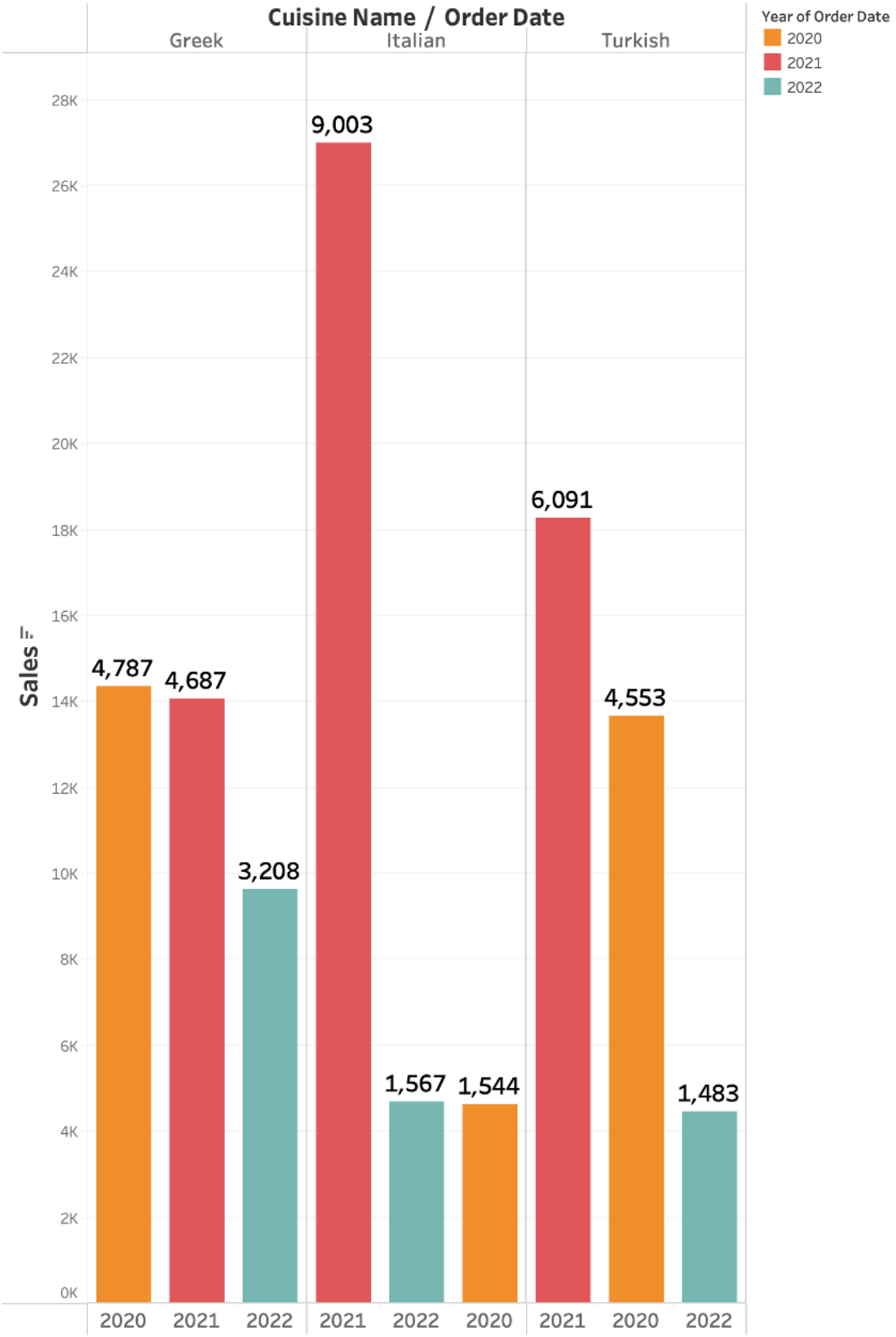


Profit chart

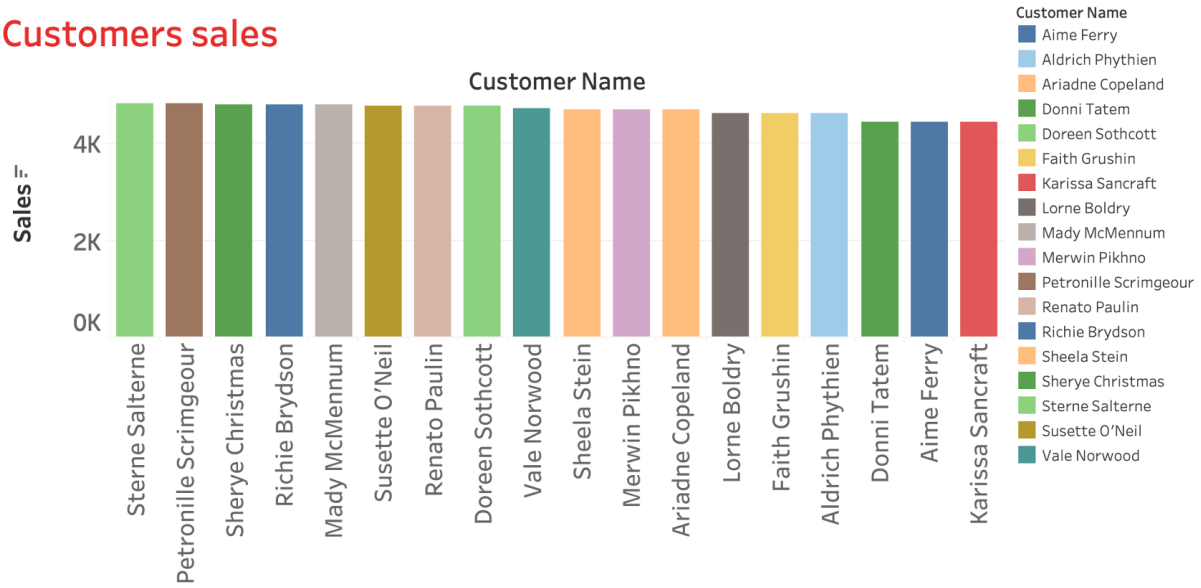


Sale Bubble Chart

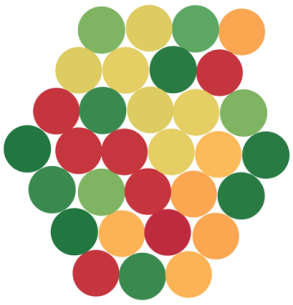
Cuisine Sales & Profit

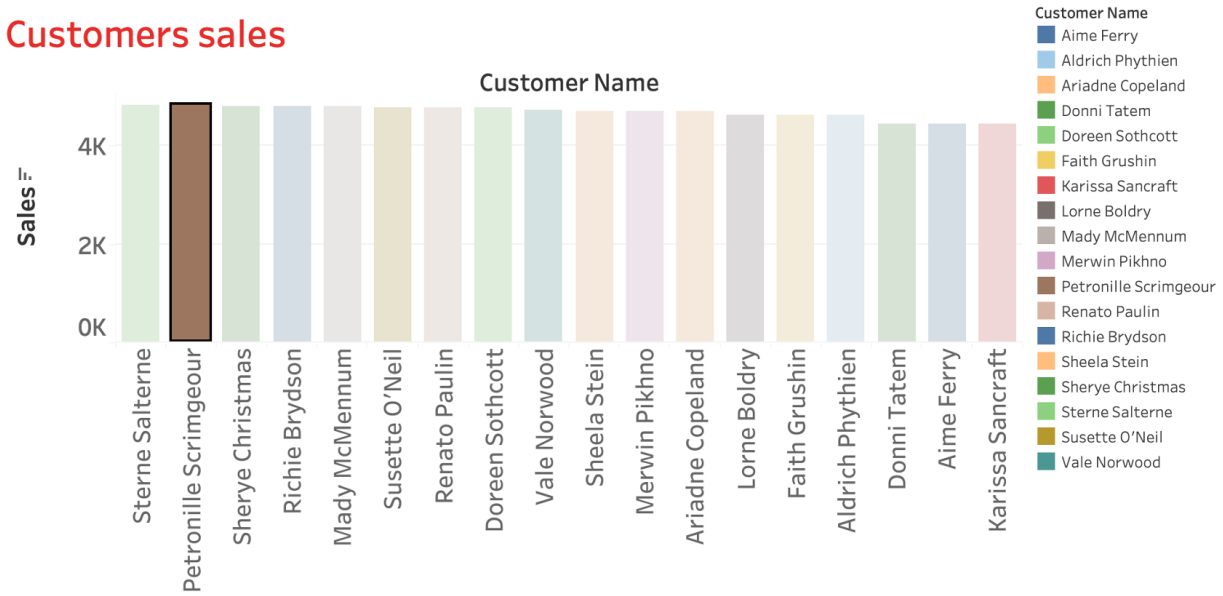


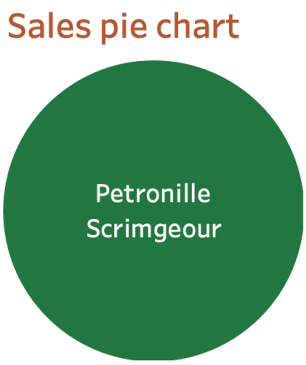
Customers sales



Sale Bubble Chart







Database Client

```
%pip install mysql-connector-python
```
✓ 1.1s Python

```
!pip show mysql-connector-python
```
✓ 0.8s Python

```
import mysql.connector as connector
```
✓ 0.1s Python

```
connection = connector.connect(user='root', password='root@123', port=3306, host='localhost',database= 'littlelemondb')
```
✓ 0.0s Python

```
# Create a cursor object to execute SQL statements
cursor = connection.cursor()

```
✓ 0.0s Python

```python
# Define SQL statements (embedded directly in Python code)
show_table_query = "SHOW tables"
cursor.execute(show_table_query)
```

✓ 0.0s Python

```python
results = cursor.fetchall()
print(results)
```

✓ 0.0s Python

```
[('Bookings',), ('Category',), ('Customers',), ('DeliveryStatus',), ('Menu',), ('menu_menuitems',), ('MenuItems',), ('Orders',), ('orde
```

```python
sql_query = """

SELECT b.BookingID, b.TableID, c.FullName, c.ContactNumber, c.Email, o.TotalCost
FROM Customers c
        JOIN Orders o ON c.CustomerID = o.CustomerID
        JOIN Bookings b ON c.CustomerID = b.CustomerID
WHERE o.TotalCost > 60

"""
cursor.execute(sql_query)
```

✓ 0.0s Python

```python
results = cursor.fetchall()

```

✓ 0.0s Python

```python
print(cursor.column_names)
print(results)
```

✓ 0.0s Python

```
('BookingID', 'TableID', 'FullName', 'ContactNumber', 'Email', 'TotalCost')
[(1, 5, 'Alice Johnson', '+1 555-1234', ' alice.johnson@email.com', Decimal('260')), (2, 3, 'Emily Smith', '+1 555-9876', 'emily.smith@
```