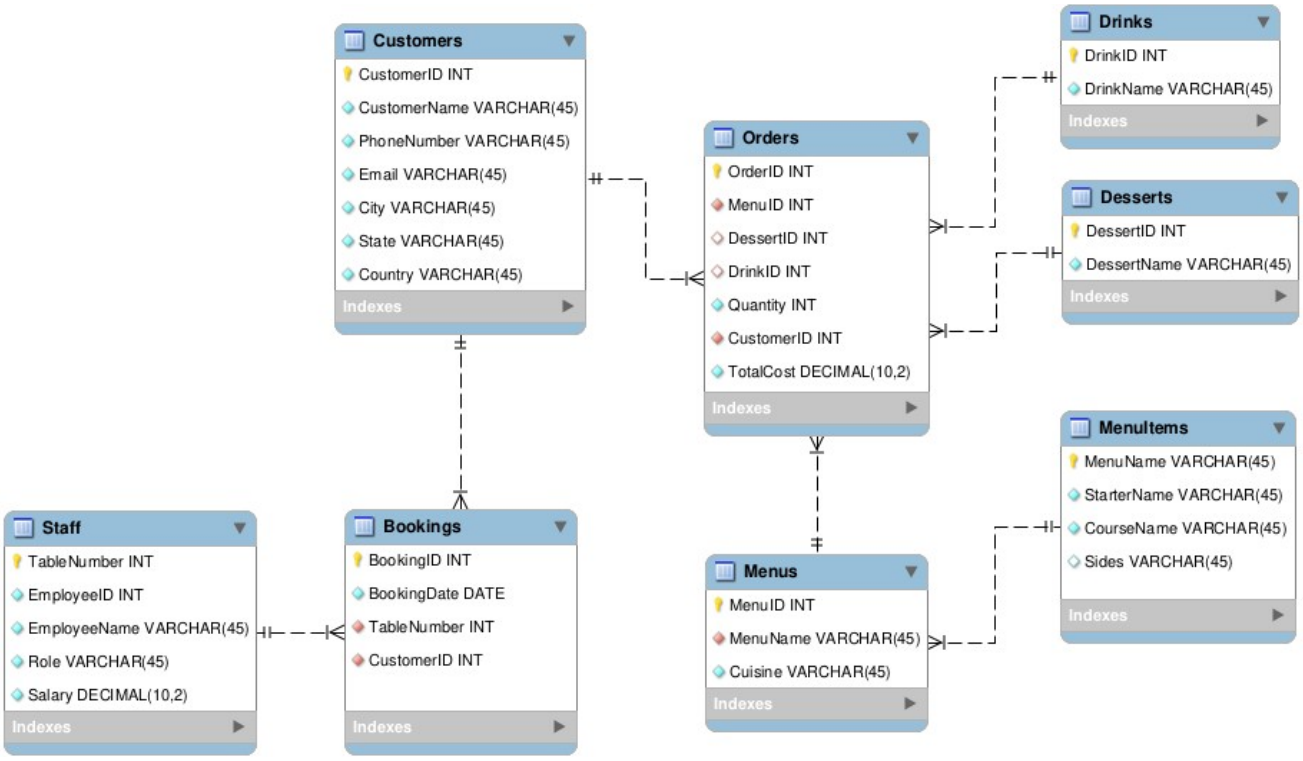
Little Lemon step by step database construction, procedures, TABLEAU output and Jupyter tasks

This is a reference document of all the tasks related to the project, in a visual format as a backup or reference from the working files

ER Diagram



SCHEMA in MySQL, forward engineer output (data is filled later with a Python program)

```
-- MySQL Script generated by MySQL Workbench
-- sáb 16 sep 2023 23:25:17
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_Z
ERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema LittleLemonDB
-- -----------------------------------------------------

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

-- -----------------------------------------------------
-- Table `LittleLemonDB`.`Staff`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `LittleLemonDB`.`Staff` (
  `TableNumber` INT NOT NULL,
  `EmployeeID` INT NOT NULL,
  `EmployeeName` VARCHAR(45) NOT NULL,
  `Role` VARCHAR(45) NOT NULL,
  `Salary` DECIMAL(10,2) NOT NULL,
  PRIMARY KEY (`TableNumber`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `LittleLemonDB`.`MenuItems`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `LittleLemonDB`.`MenuItems` (
  `MenuName` VARCHAR(45) NOT NULL,
  `StarterName` VARCHAR(45) NOT NULL,
  `CourseName` VARCHAR(45) NOT NULL,
  `Sides` VARCHAR(45) NULL,
  PRIMARY KEY (`MenuName`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `LittleLemonDB`.`Menus`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `LittleLemonDB`.`Menus` (
  `MenuID` INT NOT NULL,
  `MenuName` VARCHAR(45) NOT NULL,
  `Cuisine` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`MenuID`),
  INDEX `menuname_fk_idx` (`MenuName` ASC) VISIBLE,
  CONSTRAINT `menuname_fk`
    FOREIGN KEY (`MenuName`)
    REFERENCES `LittleLemonDB`.`MenuItems` (`MenuName`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `LittleLemonDB`.`Customers`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `LittleLemonDB`.`Customers` (
  `CustomerID` INT NOT NULL,
  `CustomerName` VARCHAR(45) NOT NULL,
  `PhoneNumber` VARCHAR(45) NOT NULL,
  `Email` VARCHAR(45) NOT NULL,
  `City` VARCHAR(45) NOT NULL,
  `State` VARCHAR(45) NOT NULL,
  `Country` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`CustomerID`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `LittleLemonDB`.`Drinks`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `LittleLemonDB`.`Drinks` (
  `DrinkID` INT NOT NULL,
  `DrinkName` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`DrinkID`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `LittleLemonDB`.`Desserts`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `LittleLemonDB`.`Desserts` (
  `DessertID` INT NOT NULL,
```

```
  `DessertName` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`DessertID`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `LittleLemonDB`.`Orders`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `LittleLemonDB`.`Orders` (
  `OrderID` INT NOT NULL,
  `MenuID` INT NOT NULL,
  `DessertID` INT NULL,
  `DrinkID` INT NULL,
  `Quantity` INT NOT NULL,
  `CustomerID` INT NOT NULL,
  `TotalCost` DECIMAL(10,2) NOT NULL,
  PRIMARY KEY (`OrderID`),
  INDEX `menuid_fk_idx` (`MenuID` ASC) VISIBLE,
  INDEX `orders_fk_idx` (`CustomerID` ASC) VISIBLE,
  INDEX `drinkid_fk_idx` (`DrinkID` ASC) VISIBLE,
  INDEX `dessert_id_idx` (`DessertID` ASC) VISIBLE,
  CONSTRAINT `menuid_fk`
    FOREIGN KEY (`MenuID`)
    REFERENCES `LittleLemonDB`.`Menus` (`MenuID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `orders_fk`
    FOREIGN KEY (`CustomerID`)
    REFERENCES `LittleLemonDB`.`Customers` (`CustomerID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `drinkid_fk`
    FOREIGN KEY (`DrinkID`)
    REFERENCES `LittleLemonDB`.`Drinks` (`DrinkID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `dessert_id`
    FOREIGN KEY (`DessertID`)
    REFERENCES `LittleLemonDB`.`Desserts` (`DessertID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `LittleLemonDB`.`Bookings`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `LittleLemonDB`.`Bookings` (
  `BookingID` INT NOT NULL AUTO_INCREMENT,
```

```
  `BookingDate` DATE NOT NULL,
  `TableNumber` INT NOT NULL,
  `CustomerID` INT NOT NULL,
  PRIMARY KEY (`BookingID`),
  INDEX `customerid_fk_idx` (`CustomerID` ASC) VISIBLE,
  INDEX `tablenumber_fk_idx` (`TableNumber` ASC) VISIBLE,
  CONSTRAINT `customerid_fk`
    FOREIGN KEY (`CustomerID`)
    REFERENCES `LittleLemonDB`.`Customers` (`CustomerID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `tablenumber_fk`
    FOREIGN KEY (`TableNumber`)
    REFERENCES `LittleLemonDB`.`Staff` (`TableNumber`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Python connection to populate the database

```
# 1.Establish a connection
# Import MySQL Connector/Python
import mysql.connector as connector
connection = connector.connect(database="LittleLemonDB",host="localhost",
user="admin1", password="R0drig0#")

# 2. Create a cursor
cursor = connection.cursor()

# 3. Create the database and set it for use
cursor.execute("USE LittleLemonDB")

#*********************************************************#
# Insert query to populate "Entries" table:
#*********************************************************#

insert_Customers="""
INSERT INTO Customers (CustomerID, CustomerName, PhoneNumber, Email, City, State,
Country) VALUES
(1,'Andres Chang','555-555-0001','andres@metal23.com','New York','New York','USA'),
(2,'Maria Rodriguez','555-555-0002','maria@email.com','Los Angeles','California','USA'),
(3,'John Doe','555-555-0003','john@example.com','Chicago','Illinois','USA'),
(4,'Michael Smith','555-555-0004','michael@example.com','Miami','Florida','USA'),
(5,'Sarah Johnson','555-555-0005','sarah@email.com','Houston','Texas','USA'),
(6,'Emily Davis','555-555-0006','emily@example.com','San Francisco','California','USA'),
(7,'Robert Wilson','555-555-0007','robert@email.com','Phoenix','Arizona','USA'),
(8,'Laura Martinez','555-555-0008','laura@email.com','Dallas','Texas','USA'),
(9,'William Johnson','555-555-0009','william@email.com','Denver','Colorado','USA'),
(10,'Christopher Lee','555-555-0010','chris@email.com','Seattle','Washington','USA'),
(11,'Daniel Thompson','555-555-1001','daniel@g.com','Atlanta','Georgia','USA'),
(12,'Jennifer Adams','555-555-1002','jennifer@email.com','Boston','Massachusetts','USA'),
(13,'Amanda Harris','555-555-1003','amanda@email.com','Philadelphia','Pennsylvania','USA'),
(14,'Jonathan Davis','555-555-1004','jonathan@email.com','Detroit','Michigan','USA'),
(15,'Olivia Clark','555-555-1005','oliviac@email.com','San Diego','California','USA'),
(16,'Matthew Turner','555-555-1006','matthew@email.com','Seattle','Washington','USA'),
(17,'Sophia White','555-555-1007','sophia@email.com','Miami','Florida','USA'),
(18,'Ethan King','555-555-1008','ethan@email.com','Dallas','Texas','USA'),
(19,'Sophie Baker','555-555-1011','sophie@email.com','New Orleans','Louisiana','USA'),
(20,'James Miller','555-555-1012','james@vvvemail.com','San Francisco','California','USA'),
(21,'Olivia Garcia','555-555-1013','olivia@email.com','Phoenix','Arizona','USA'),
(22,'Daniel Harris','555-555-1014','daniel@email.com','Portland','Oregon','USA'),
(23,'Ava Turner','555-555-1015','ava@email.com','Dallas','Texas','USA'),
(24,'Liam Jackson','555-555-1016','liam@email.com','Seattle','Washington','USA'),
(25,'Sophia Wilson','555-555-1017','sophia@email.com','Denver','Colorado','USA'),
(26,'Ella Smith','555-555-1018','ella@email.com','Chicago','Illinois','USA'),
```

(27,'William Brown','555-555-1019','williambrown@email.com','Houston','Texas','USA'),
(28,'Mia Rodriguez','555-555-1020','mia@email.com','Los Angeles','California','USA'),
(30,'Natalie Turner','555-555-1023','natalie@email.com','New Orleans','Louisiana','USA'),
(31,'Ethan Martinez','555-555-1024','ethanm@email.com','New Orleans','Louisiana','USA'),
(32,'Marta Lopez','555-555-2001','marta@email.com','New Orleans','Louisiana','USA'),
(33,'Alejandro Rodriguez','555-555-2002','alejandro@email.com','New Orleans','Louisiana','USA'),
(34,'Lucia Fernandez','555-555-2003','lucia@email.com','New York','New York','USA'),
(35,'Diego Garcia','555-555-2004','diego@email.com','Houston','Texas','USA'),
(36,'Carmen Martinez','555-555-2005','carmen@email.com','Chicago','Illinois','USA'),
(37,'Manuel Sanchez','555-555-2006','manuel@email.com','San Francisco','California','USA'),
(38,'Elena Perez','555-555-2007','elena@email.com','Dallas','Texas','USA'),
(39,'Javier Gonzalez','555-555-2008','javier@email.com','Phoenix','Arizona','USA'),
(40,'Isabel Ramirez','555-555-2009','isabel@email.com','Miami','Florida','USA'),
(41,'Sergio Torres','555-555-2010','sergio@email.com','Los Angeles','California','USA'),
(42,'Ji-hyun Kim','555-555-3001','jihyun@email.com','Los Angeles','California','USA'),
(43,'Min-jun Park','555-555-3002','minjun@email.com','New York','New York','USA'),
(44,'Eun-ji Lee','555-555-3003','eunji@email.com','Chicago','Illinois','USA'),
(45,'Sang-hoon Choi','555-555-3004','sanghoon@email.com','San Francisco','California','USA'),
(46,'Hae-won Kang','555-555-3005','haewon@email.com','Houston','Texas','USA'),
(47,'Yoon-hee Park','555-555-3006','yoonhee@email.com','Miami','Florida','USA'),
(48,'Joon-ho Kim','555-555-3007','joonho@email.com','Phoenix','Arizona','USA'),
(49,'Seo-yeon Lee','555-555-3008','seoyeon@email.com','Dallas','Texas','USA'),
(50,'Min-seok Cho','555-555-3009','minseok@email.com','Seattle','Washington','USA'),
(51,'Ji-eun Kim','555-555-3010','jieun@email.com','Denver','Colorado','USA'),
(52,'Claire Dubois','555-555-4001','claire@email.com','New Orleans','Louisiana','USA'),
(53,'Pierre Leclerc','555-555-4002','pierre@email.com','New Orleans','Louisiana','USA'),
(54,'Isabelle Dupont','555-555-4003','isabelle@email.com','New Orleans','Louisiana','USA'),
(55,'François Martin','555-555-4004','francois@email.com','New Orleans','Louisiana','USA'),
(56,'Amélie Renaud','555-555-4005','amelie@email.com','New Orleans','Louisiana','USA'),
(57,'Louis Dupuis','555-555-4006','louis@email.com','New Orleans','Louisiana','USA'),
(58,'Sophie Lambert','555-555-4007','sophie@v.com','New Orleans','Louisiana','USA'),
(59,'Luc Dubois','555-555-4008','luc@email.com','New Orleans','Louisiana','USA'),
(60,'Marie Leclerc','555-555-4009','marie@email.com','New Orleans','Louisiana','USA'),
(61,'Antoine Blanc','555-555-4010','antoine@email.com','New Orleans','Louisiana','USA'),
(62,'Renata Patel','555-555-5001','renata@email.com','Boulder','Colorado','USA'),
(63,'Carlos Kim','555-555-5002','carlos@email.com','Houston','Texas','USA'),
(64,'Sofia Nakamura','555-555-5003','sofia@email.com','San Francisco','California','USA'),
(65,'Fabio Nguyen','555-555-5004','fabio@email.com','Los Angeles','California','USA'),
(66,'Michael Johnson','555-555-5005','michael@email.com','New Orleans','Louisiana','USA'),
(67,'Emily Thomas','555-555-5006','emily@email.com','New Orleans','Louisiana','USA'),
(68,'James Brown','555-555-5007','james@email.com','New Orleans','Louisiana','USA'),
(69,'Susan White','555-555-5008','susan@email.com','New Orleans','Louisiana','USA'),
(70,'Robert Johnson','555-555-5009','robertjohnson@email.com','New Orleans','Louisiana','USA'),
(71,'Lisa Anderson','555-555-5010','lisa@email.com','New Orleans','Louisiana','USA'),
(72,'Andrew Brown','555-555-5011','andrew@email.com','New Orleans','Louisiana','USA'),

```
(73,'Karen Wilson','555-555-5012','karen@email.com','New Orleans','Louisiana','USA'),
(74,'Mark Davis','555-555-5013','mark@email.com','New Orleans','Louisiana','USA');
"""

# Populate table
cursor.execute(insert_Customers)
connection.commit()



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

# Insert query to populate "Staff" table:
#*********************************************************#

insert_Staff="""
INSERT INTO Staff (TableNumber, EmployeeID, EmployeeName, Role, Salary)
VALUES
(1,1, 'John Smith', 'Chief Waiter', 3500),
(2,2, 'Ava Wilson', 'Waiter', 1800),
(3,3, 'Olivia Smith', 'Waiter', 1400),
(4,4, 'Emma Brown', 'Waiter', 1100),
(5,5, 'Rodrigo Guedes', 'Waiter', 1500),
(6,6, 'Slavik Kinski', 'Waiter', 2000);
"""

# Populate table
cursor.execute(insert_Staff)
connection.commit()



#*********************************************************#
# Insert query to populate "Bookings" table:
#*********************************************************#

insert_Bookings="""
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);
"""

# Populate table
cursor.execute(insert_Bookings)
connection.commit()
```

```
#*********************************************************#
# Insert query to populate "MenuItems" table:
#*********************************************************#

insert_MenuItems="""
INSERT INTO MenuItems (MenuName, StarterName, CourseName, Sides) VALUES
('GreekDay', 'Olives', 'Greek Salad', 'Tapas'),
('Harakara', 'Hummus', 'Sushi', 'Fries'),
('Mafia33', 'Bruschetta', 'Pasta', 'Salad'),
('AztecNoon', 'Calamari', 'Tacos', 'Onion Rings'),
('Yosemite', 'Mozzarella', 'Steak', 'Coleslaw'),
('Alahamaha', 'Falafel', 'Kebab', 'Tabouleh'),
('Athenea', 'Tzatziki', 'Gyros', 'Hummus'),
('Fuji', 'Spring Rolls', 'Sushi', 'Edamame'),
('Cappone', 'Caprese Salad', 'Pizza', 'Garlic Knots'),
('Maya3', 'Guacamole', 'Tacos', 'Rice and Beans'),
('SeattleWink', 'Shrimp Cocktail', 'Steak', 'Smashed Potatoes');
"""

# Populate table
cursor.execute(insert_MenuItems)
connection.commit()


#*********************************************************#
# Insert query to populate "Menus" table:
#*********************************************************#

insert_Menus="""
INSERT INTO Menus (MenuID, MenuName, Cuisine) VALUES
(1, 'GreekDay', 'Greek'),
(2, 'Harakara', 'Japanese'),
(3, 'Mafia33', 'Italian'),
(4, 'AztecNoon', 'Mexican'),
(5, 'Yosemite', 'American'),
(6, 'Alahamaha', 'Middle Eastern'),
(7, 'Athenea', 'Greek'),
(8, 'Fuji', 'Japanese'),
(9, 'Cappone', 'Italian'),
(10,'Maya3', 'Mexican'),
(11,'SeattleWink', 'American');

"""

# Populate table
cursor.execute(insert_Menus)
connection.commit()
```

```
#*********************************************************#
# Insert query to populate "Desserts" table:
#*********************************************************#

insert_Desserts="""
INSERT INTO Desserts (DessertID, DessertName) VALUES
(1, 'IceCream'),
(2, 'Tiramisu'),
(3, 'Cheesecake'),
(4, 'Apple Pie');
"""

# Populate table
cursor.execute(insert_Desserts)
connection.commit()


#*********************************************************#
# Insert query to populate "Drinks" table:
#*********************************************************#

insert_Drinks="""
INSERT INTO Drinks (DrinkID, DrinkName) VALUES
(1, 'Water'),
(2, 'Barolo Red Wine'),
(3, 'Merlot'),
(4, 'Lemonade');
"""

# Populate table
cursor.execute(insert_Drinks)
connection.commit()



#*********************************************************#
# Insert query to populate "Orders" table:
#*********************************************************#

insert_Orders="""
INSERT INTO Orders (OrderID, MenuID, DessertID, DrinkID, Quantity, CustomerID, TotalCost)
VALUES
(1, 1, 1, 3, 1, 4, 100.8),
(2, 2, 4, 4, 2, 5, 341.6),
(3, 3, 3, 3, 1, 6, 72.8),
(4, 4, 3, 3, 1, 8, 93.8),
(5, 5, 4, 3, 4, 2, 291.2),
```

```
(6, 6, 1, 1, 1, 19, 151.2),
(7, 7, 2, 2, 1, 18, 65.8),
(8, 8, 4, 3, 1, 20, 109.2),
(9, 9, 3, 2, 3, 21, 235.2),
(10, 10, 2, 2, 1, 39, 140),
(11, 11, 1, 3, 1, 11, 169.4);
"""

# Populate table
cursor.execute(insert_Orders)
connection.commit()
```

Sales Report procedures and statements

DELIMITERS here are for Command Line Test. On MySQL Workbench are not required.

Adding Sales Report

```
CREATE VIEW `OrdersView` AS SELECT OrderID, Quantity, TotalCost AS Cost
FROM Orders WHERE Quantity > 2 ORDER BY Cost ASC;
```

```
1 •  SELECT * FROM OrdersView;
```

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

#	OrderID	Quantity	Cost
1	5	4	291.20
2	9	3	235.20

Information on customers with cost more than $150.

```
SELECT Customers.CustomerID, Customers.CustomerName, Orders.OrderID,
Orders.TotalCost AS Cost, Menus.MenuName, MenuItems.CourseName FROM
Customers INNER JOIN Orders USING (CustomerID) INNER JOIN Menus USING
(MenuID) INNER JOIN MenuItems USING (MenuName) WHERE TotalCost > 150;
```

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

#	CustomerID	CustomerName	OrderID	Cost	MenuName	CourseName
1	5	Sarah Johnson	2	341.60	Harakara	Sushi
2	2	Maria Rodriguez	5	291.20	Yosemite	Steak
3	19	Sophie Baker	6	151.20	Alahamaha	Kebab
4	21	Olivia Garcia	9	235.20	Cappone	Pizza

Find all menu tiems for which more than 2 orders have been placed

```
SELECT MenuName FROM Menus WHERE MenuID = ANY (SELECT MenuID FROM Orders
WHERE Quantity > 2);
```

#	MenuName
1	Yosemite
2	Cappone

Stored Procedure that displays the maximum ordered quantity in the Orders Table

```
DELIMITER //
CREATE PROCEDURE GetMaxQuantity()
BEGIN
SELECT MAX(Quantity) AS 'Max Quantity in Order' FROM Orders;
END //
DELIMITER ;

CALL GetMaxQuantity();
```

#	Max Quantity in Order
1	4

Prepared Statement GetOrderDetail, for Customer Number 1

```
PREPARE GetOrderDetail FROM 'SELECT OrderID, Quantity, TotalCost AS Cost
FROM Orders WHERE OrderID = ?';
SET @id = 1;
EXECUTE GetOrderDetail USING @id;
```

#	OrderID	Quantity	Cost
1	1	1	100.80

CancelOrder Procedure

```
DELIMITER //
CREATE PROCEDURE CancelOrder(IN o_id INT)
BEGIN
DECLARE msg VARCHAR(100);
DELETE FROM Orders WHERE OrderID=o_id;
SET msg = CONCAT('Order ',o_id, ' is cancelled');
SELECT msg AS Confirmation;
END //
DELIMITER;

CALL CancelOrder(10);
```

#	Confirmation
1	Order 10 is cancelled

Check Order 10 cancelled

```
1 • select*from Orders;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

#	OrderID	MenuID	DessertID	DrinkID	Quantity	CustomerID	TotalCost
1	1	1	1	3	1	4	100.80
2	2	2	4	4	2	5	341.60
3	3	3	3	3	1	6	72.80
4	4	4	3	3	1	8	93.80
5	5	5	4	3	4	2	291.20
6	6	6	1	1	1	19	151.20
7	7	7	2	2	1	18	65.80
8	8	8	4	3	1	20	109.20
9	9	9	3	2	3	21	235.20
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Populate Bookings Table with given data

This table filling is done in the initial Python routine.

```
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);
```

Check Booking Procedure

```
DELIMITER //
CREATE PROCEDURE CheckBooking(IN book_date DATE, IN table_num INT)
BEGIN
DECLARE msg VARCHAR(100);
IF (SELECT COUNT(*) FROM Bookings WHERE BookingDate = book_date AND
TableNumber = table_num) > 0 THEN
SET msg = CONCAT('Table ', table_num, ' is already booked');
ELSE
SET msg = CONCAT('Table ', table_num, ' is free! Booking set.');
END IF;
SELECT msg AS "Booking status"; -- Return the message as "Booking status"
END;
//
DELIMITER ;
```

```
CALL CheckBooking('2022-11-12', 3);
```


```
CALL CheckBooking('2022-11-12', 5);
```

Add a Valid Booking prior verification

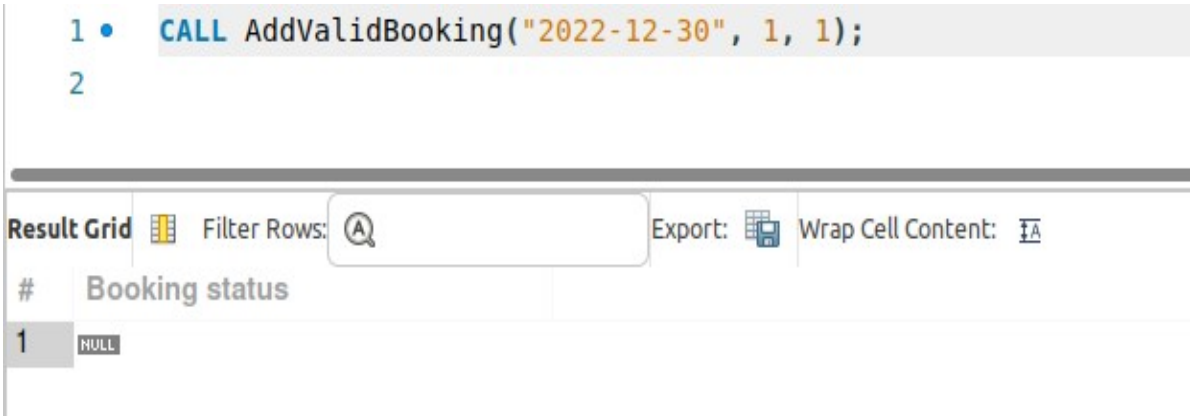
```
DELIMITER //
CREATE PROCEDURE AddValidBooking(IN booking_date DATE, IN table_number INT,
IN customer_id INT)
BEGIN
DECLARE msg VARCHAR(100);
START TRANSACTION;
INSERT INTO Bookings (BookingDate, TableNumber, CustomerID) VALUES
(booking_date, table_number, customer_id);

# comparator is > 1 because the new data is already inserted and if the
data matches, we have 2 cases, not 1
IF (SELECT COUNT(*) FROM Bookings WHERE BookingDate = booking_date AND
TableNumber = table_number) > 1 THEN
   SET msg = CONCAT('Table ',table_number,' is already booked - booking
cancelled');
   ROLLBACK;
ELSE
   COMMIT;
END IF;

SELECT * FROM Bookings; # visual check of the table and new data or same
data
SELECT msg AS "Booking status";
END; //

DELIMITER ;
```

```
# this should add an entry → commit ok (nothing is shown)
CALL AddValidBooking("2022-12-30", 1, 1);
```




```
# this should fail, same entry as before but is booked!
CALL AddValidBooking("2022-12-30", 1, 1);
```

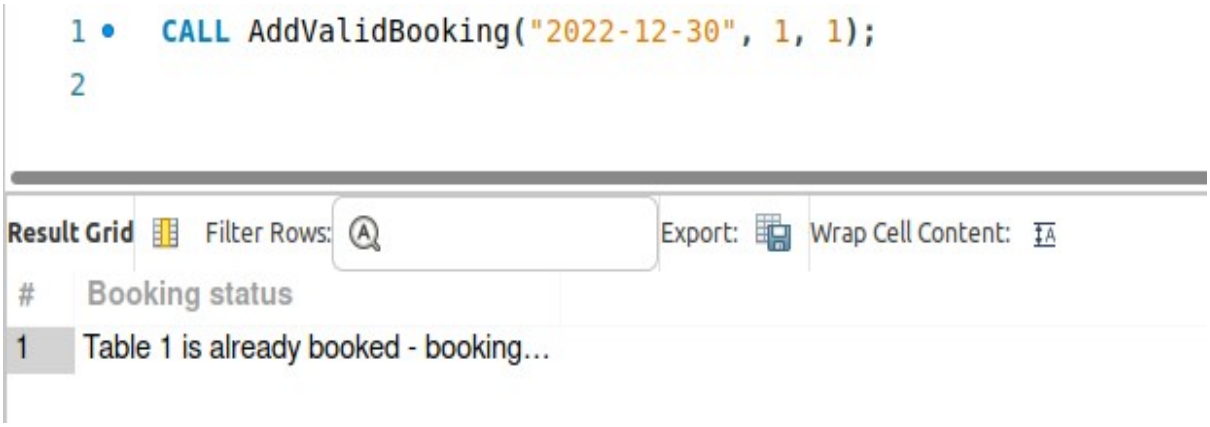
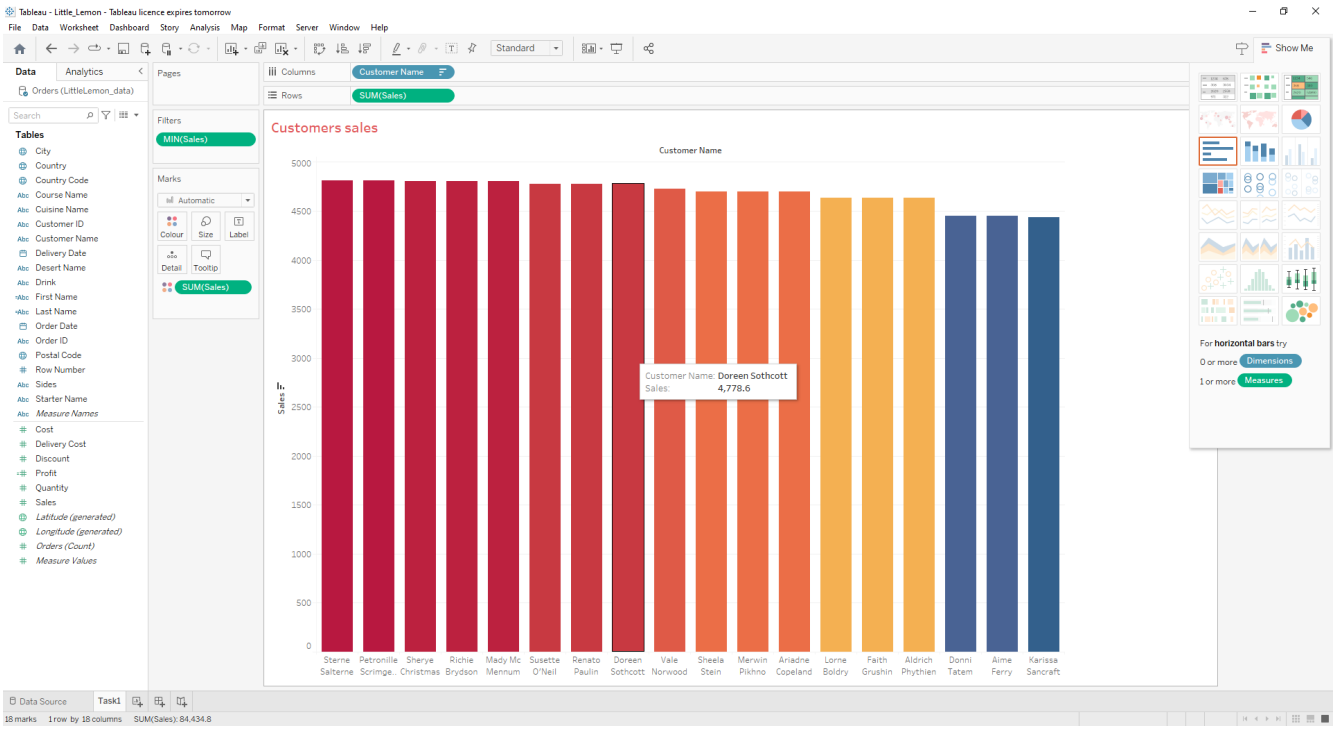
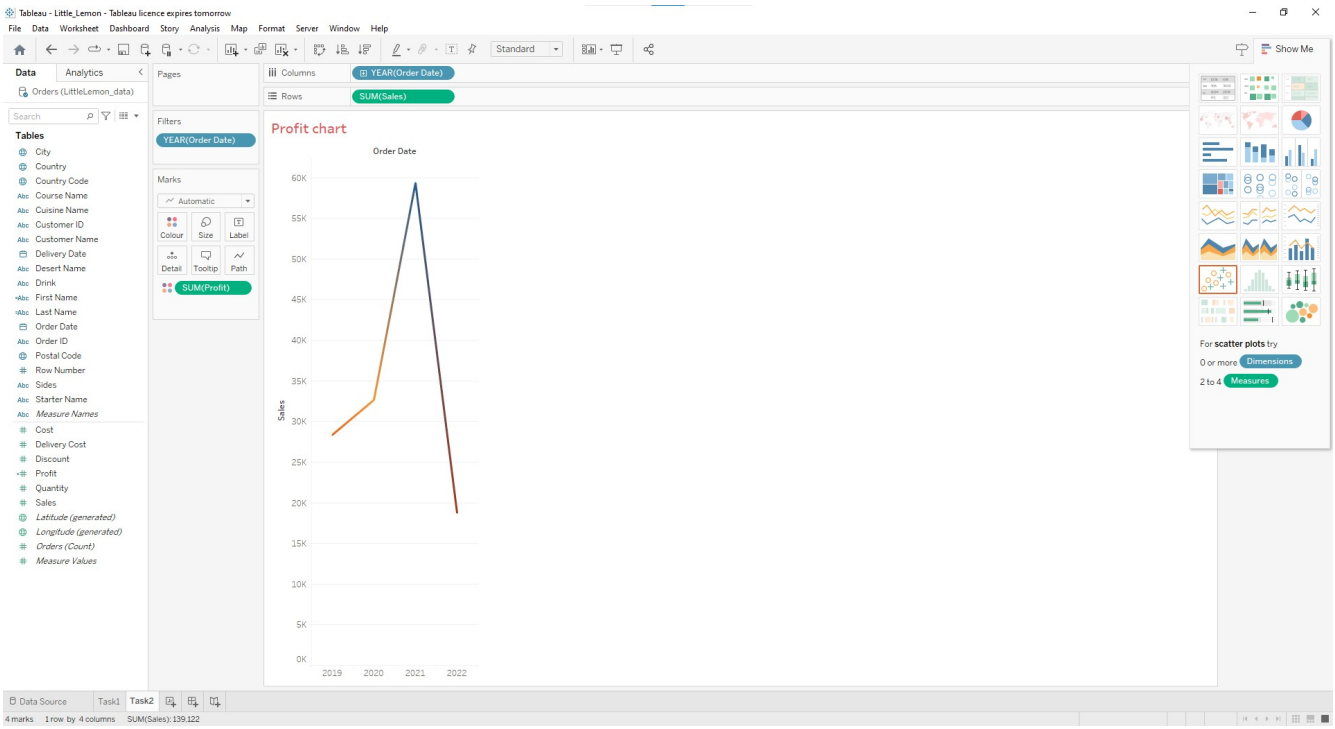


TABLEAU visualization for given data in excel file

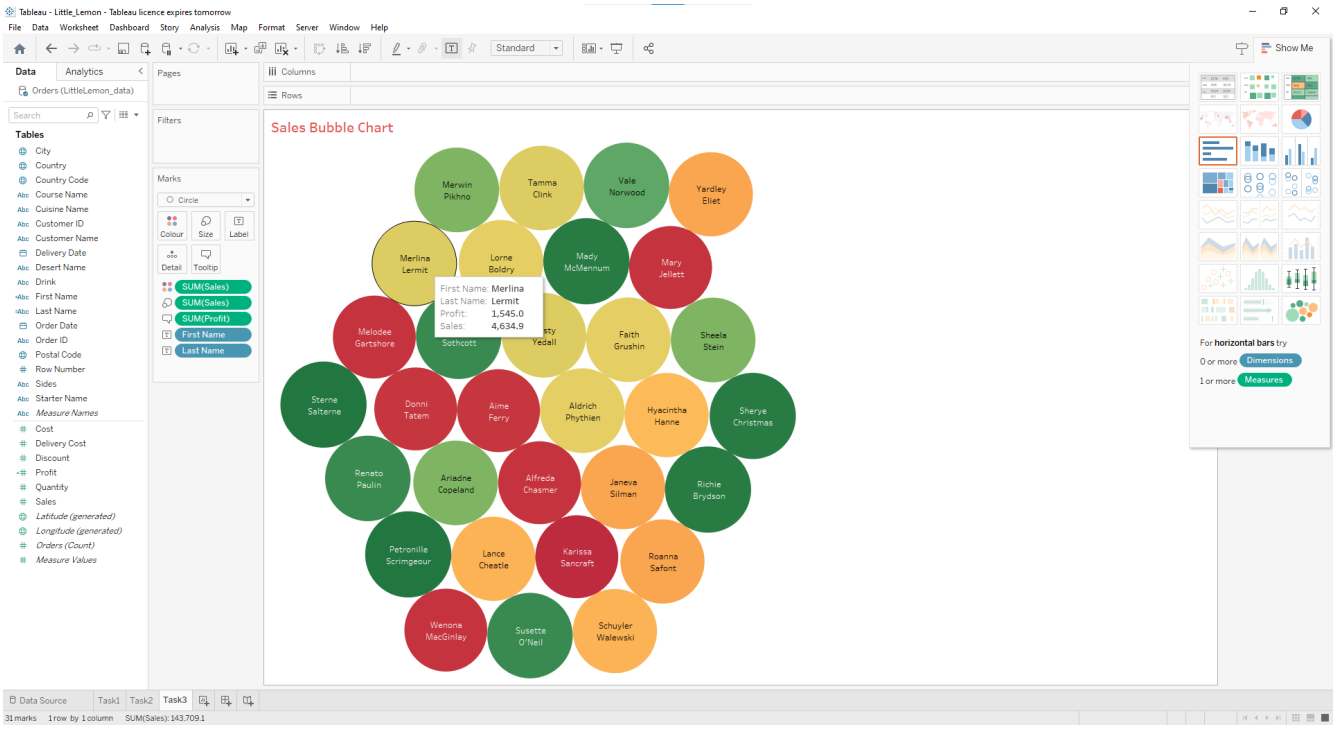
Task 1 Customers sales and filter data based on sales with at least $70.



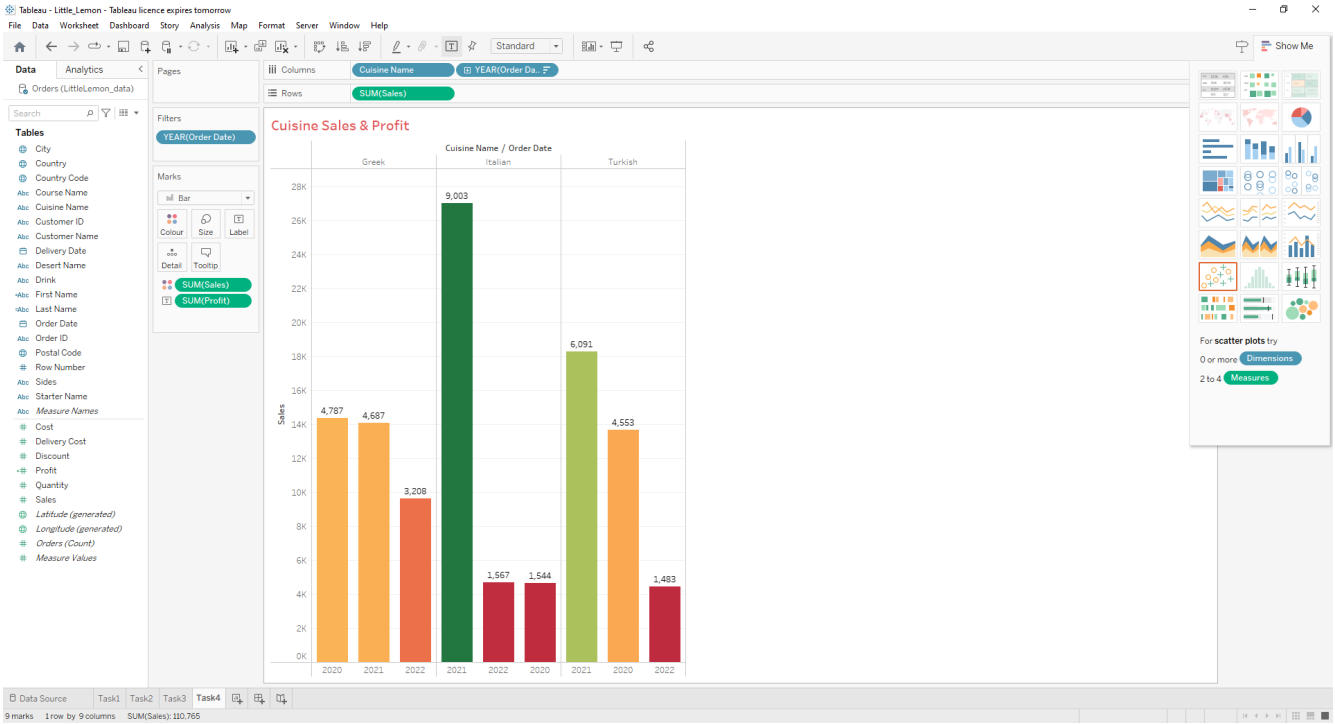
Task 2 Sales trend from 2019 to 2022



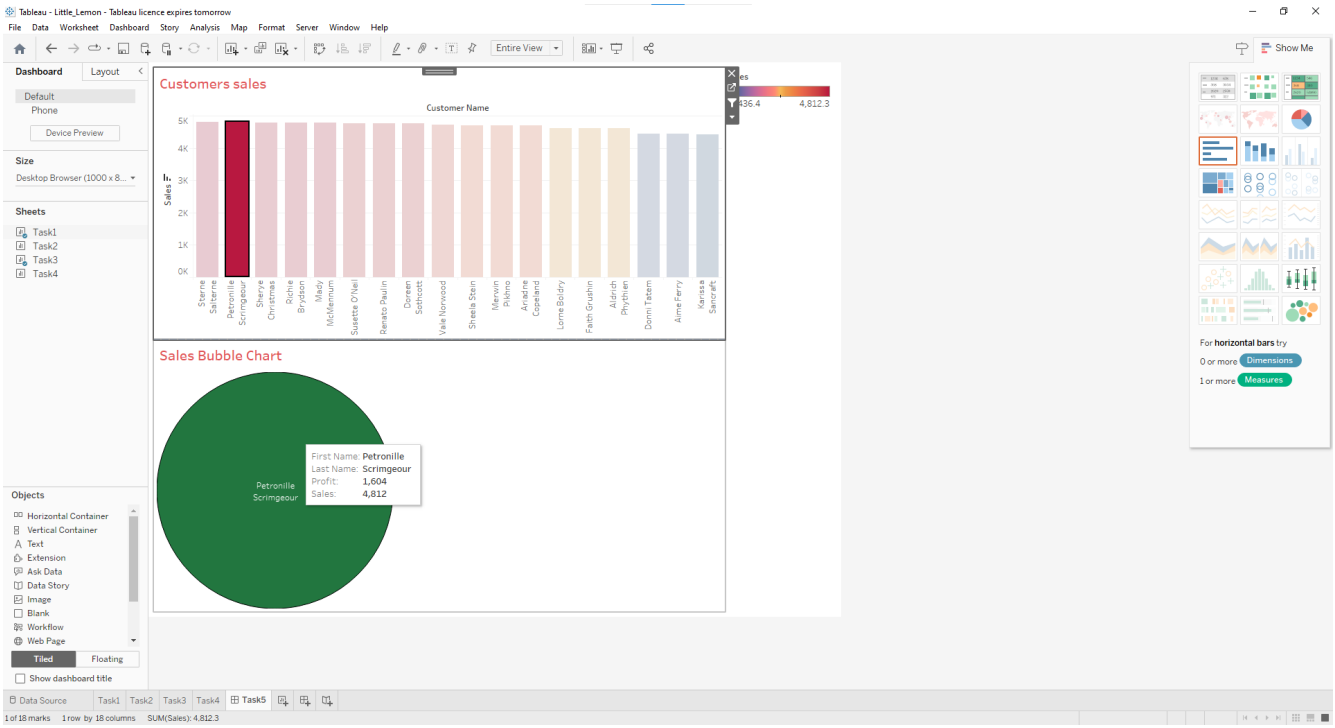
Task 3 Sales for all customers



Task 4 Sales of the Turkish, Italian and Greek cuisines



Task 5 Interactive dashboard that combines the Bar chart *Customers sales* and the *Sales* Bubble Chart



Jupyter Notebooks code

```
#***********************************
# TASK 1: Connection to the dabatase
#***********************************

# 1.Establish a connection
# Import MySQL Connector/Python
import mysql.connector as connector
connection = connector.connect(user="admin1", password="R0drig0#", db="LittleLemonDB")
print("Connection SET")

# 2. Create a cursor
cursor = connection.cursor()
print("Cursor SET")

# 3. Set database for use
cursor.execute("USE LittleLemonDB")
print("Using LittleLemonDB SET")
```

```
Connection SET
Cursor SET
Using LittleLemonDB SET
```

```
#********************************
# TASK 2: Show tables in database
#********************************
print("Tables in database, and OrdersView")
print("---------------------------------")
show_tables_query="""SHOW TABLES;"""
cursor.execute(show_tables_query)
results = cursor.fetchall()
for x in results:
    print (x)
```

```
Tables in database, and OrdersView
---------------------------------
('Bookings',)
('Customers',)
('Desserts',)
('Drinks',)
('MenuItems',)
('Menus',)
('Orders',)
('OrdersView',)
('Staff',)
```

```
#*********************************************************************
# TASK 3: Full name and contact details for clients with orders > 60
#*********************************************************************


print("Promotion Data: Name and contact details for clients with orders over $60")
print("-----------------------------------------------------------------------")
orders_60plus_query = """
SELECT Customers.CustomerName, Customers.PhoneNumber, Customers.Email, Customers.City,
Customers.State,
Orders.TotalCost AS Bill
FROM Customers INNER JOIN Orders ON Orders.CustomerID = Customers.CustomerID
WHERE TotalCost > 60;
"""


cursor.execute(orders_60plus_query)
for results2 in cursor:
    print(results2)
```

```
Promotion Data: Name and contact details for clients with orders over $60
-----------------------------------------------------------------------
('Michael Smith', '555-555-0004', 'michael@example.com', 'Miami', 'Florida', Decimal('100.80'))
('Sarah Johnson', '555-555-0005', 'sarah@email.com', 'Houston', 'Texas', Decimal('341.60'))
('Emily Davis', '555-555-0006', 'emily@example.com', 'San Francisco', 'California', Decimal('72.80'))
('Laura Martinez', '555-555-0008', 'laura@email.com', 'Dallas', 'Texas', Decimal('93.80'))
('Maria Rodriguez', '555-555-0002', 'maria@email.com', 'Los Angeles', 'California', Decimal('291.20'))
('Sophie Baker', '555-555-1011', 'sophie@email.com', 'New Orleans', 'Louisiana', Decimal('151.20'))
('Ethan King', '555-555-1008', 'ethan@email.com', 'Dallas', 'Texas', Decimal('65.80'))
('James Miller', '555-555-1012', 'james@vvvemail.com', 'San Francisco', 'California',
Decimal('109.20'))
('Olivia Garcia', '555-555-1013', 'olivia@email.com', 'Phoenix', 'Arizona', Decimal('235.20'))
```