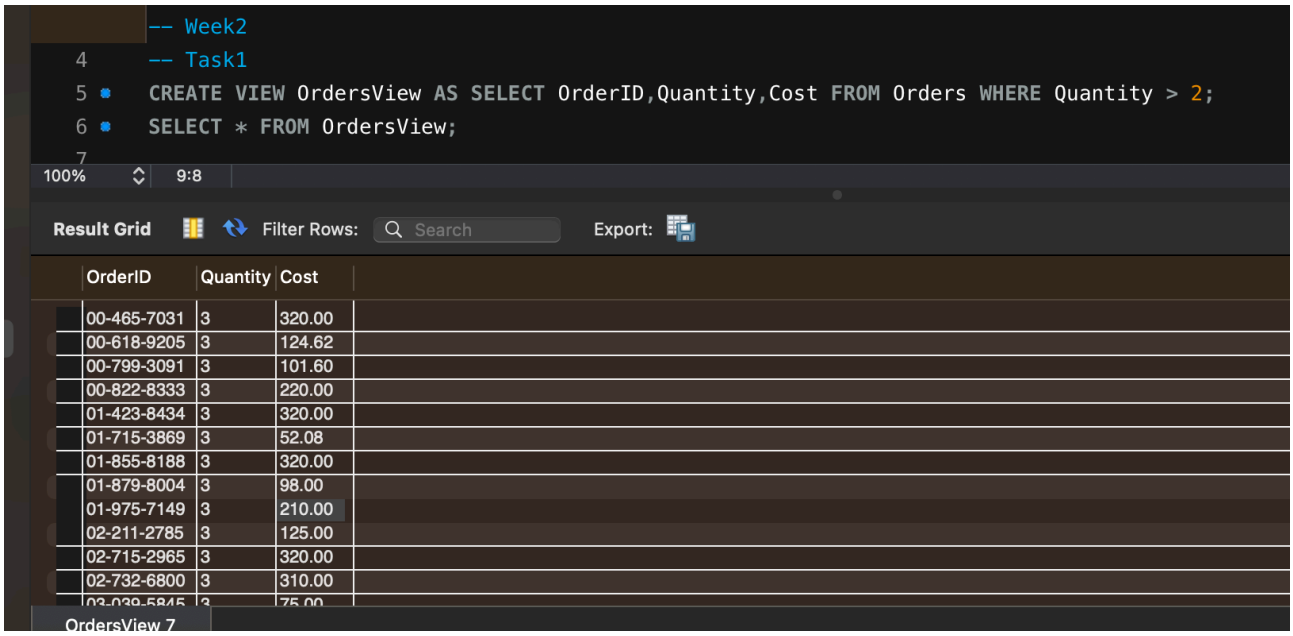
Please check my implementation of all exercises from the following captured screens.

Exercise: Create a virtual table to summarize data



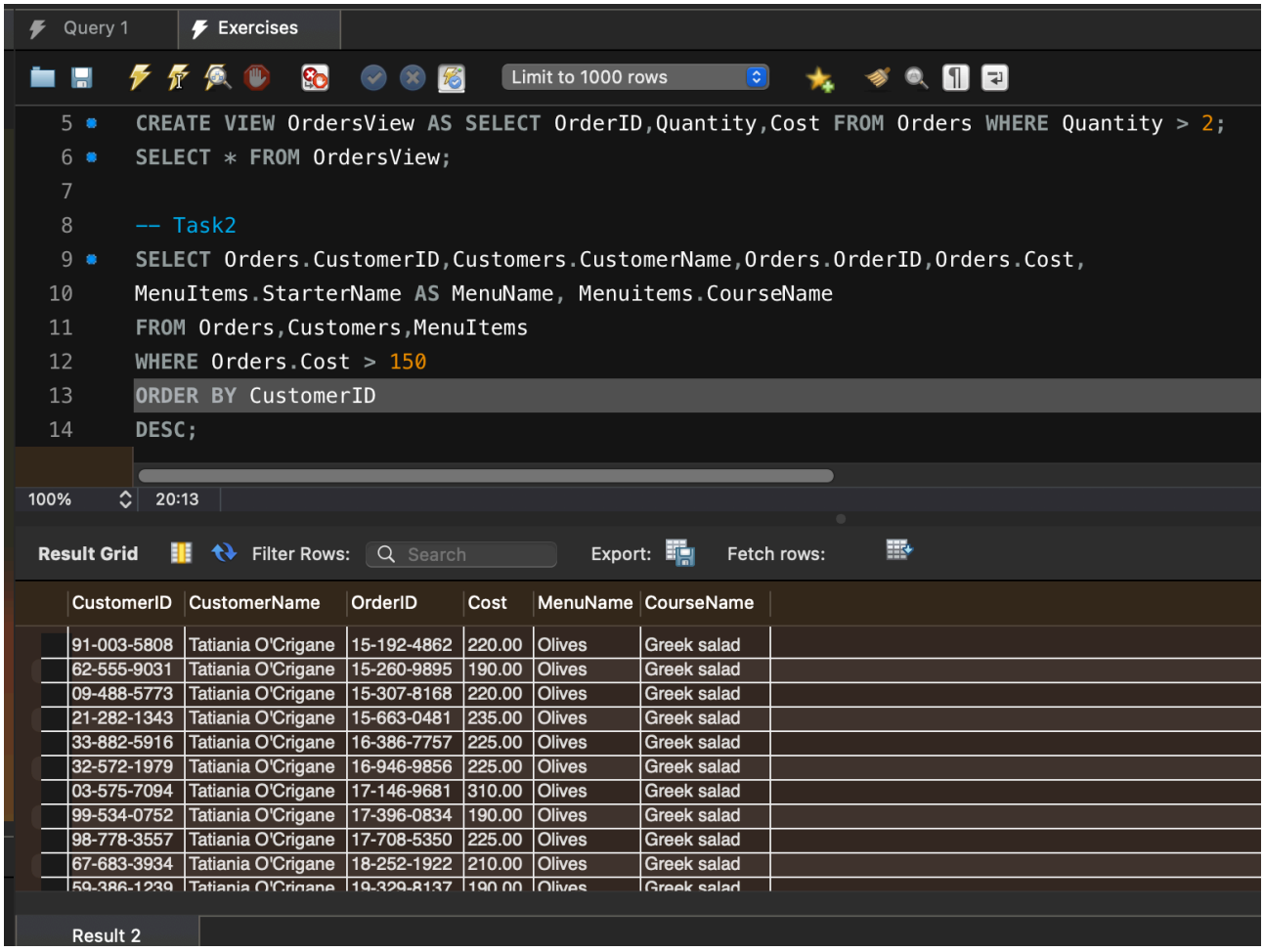
```
-- Week2
-- Task1
CREATE VIEW OrdersView AS SELECT OrderID,Quantity,Cost FROM Orders WHERE Quantity > 2;
SELECT * FROM OrdersView;
```

100% 9:8

Result Grid Filter Rows: Search Export:

OrderID	Quantity	Cost
00-465-7031	3	320.00
00-618-9205	3	124.62
00-799-3091	3	101.60
00-822-8333	3	220.00
01-423-8434	3	320.00
01-715-3869	3	52.08
01-855-8188	3	320.00
01-879-8004	3	98.00
01-975-7149	3	210.00
02-211-2785	3	125.00
02-715-2965	3	320.00
02-732-6800	3	310.00
03-039-5845	3	75.00

OrdersView 7



Query 1 Exercises

Limit to 1000 rows

```
CREATE VIEW OrdersView AS SELECT OrderID,Quantity,Cost FROM Orders WHERE Quantity > 2;
SELECT * FROM OrdersView;

-- Task2
SELECT Orders.CustomerID,Customers.CustomerName,Orders.OrderID,Orders.Cost,
MenuItems.StarterName AS MenuName, Menuitems.CourseName
FROM Orders,Customers,MenuItems
WHERE Orders.Cost > 150
ORDER BY CustomerID
DESC;
```

100% 20:13

Result Grid Filter Rows: Search Export: Fetch rows:

CustomerID	CustomerName	OrderID	Cost	MenuName	CourseName
91-003-5808	Tatiania O'Crigane	15-192-4862	220.00	Olives	Greek salad
62-555-9031	Tatiania O'Crigane	15-260-9895	190.00	Olives	Greek salad
09-488-5773	Tatiania O'Crigane	15-307-8168	220.00	Olives	Greek salad
21-282-1343	Tatiania O'Crigane	15-663-0481	235.00	Olives	Greek salad
33-882-5916	Tatiania O'Crigane	16-386-7757	225.00	Olives	Greek salad
32-572-1979	Tatiania O'Crigane	16-946-9856	225.00	Olives	Greek salad
03-575-7094	Tatiania O'Crigane	17-146-9681	310.00	Olives	Greek salad
99-534-0752	Tatiania O'Crigane	17-396-0834	190.00	Olives	Greek salad
98-778-3557	Tatiania O'Crigane	17-708-5350	225.00	Olives	Greek salad
67-683-3934	Tatiania O'Crigane	18-252-1922	210.00	Olives	Greek salad
59-386-1239	Tatiania O'Crigane	19-329-8137	190.00	Olives	Greek salad

Result 2

```sql
-- Task3
SELECT MenuItems.StarterName AS MenuName FROM MenuItems
WHERE EXISTS (SELECT Orders.Quantity FROM Orders WHERE Orders.Quantity > 2);
```

100% 77:18

Result Grid Filter Rows: Search Export: Fetch rows:

MenuName
Olives
Flatbread
Minestrone
Tomato bread
Falafel
Hummus
Olives
Flatbread
Minestrone
Tomato bread

Exercise: Create optimized queries to manage and analyze data

```sql
-- Creating stored procedure
-- Task1
DELIMITER //

CREATE PROCEDURE GetMaxQuantity()
BEGIN
SELECT MAX(Quantity) as MaxQuantity FROM Orders;
END //

DELIMITER ;
CALL GetMaxQuantity();
```

100% 23:30

Result Grid Filter Rows: Search Export:

MaxQuantity
3

```
31
32      -- Task2 Creating prepared statement
33      PREPARE GetOrderDetail FROM "SELECT OrderID,Quantity,Cost FROM Orders WHERE OrderID = ?";
34      SET @id = '03-132-2890';
35      Execute GetOrderDetail using @id;
36
```

100% 34:35

Result Grid | Filter Rows: Search | Export:

OrderID	Quantity	Cost
03-132-2890	2	75.00

Limit to 1000 rows

```
38      DELIMITER //
39
40      CREATE PROCEDURE CancelOrder(IN order_id VARCHAR(255))
41      BEGIN
42      SET @order_id_out = order_id;
43      DELETE FROM Orders WHERE OrderID = order_id;
44      INSERT INTO OrderStatus (Confirmation) VALUES (CONCAT("Order ",@order_id_out," is cancelled"));
45      END //
46
47      DELIMITER ;
48      CALL CancelOrder("00-669-6040");
49
50      SELECT * FROM OrderStatus;
51
```

100% 7:45

Result Grid | Filter Rows: Search | Edit: | Export/Import:

ID	Confirmation
1	Order 00-669-6040 is cancelled
NULL	NULL

Exercise: Create SQL queries to check available bookings based on user input

```sql
CREATE TABLE IF NOT EXISTS Bookings (
BookingID INT PRIMARY KEY,
BookingDate DATE,
TableNumber INT,
CustomerID INT);

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);

SELECT * FROM Bookings;
```

100% 24:13

Result Grid Filter Rows: Search Edit: Export/Import:

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

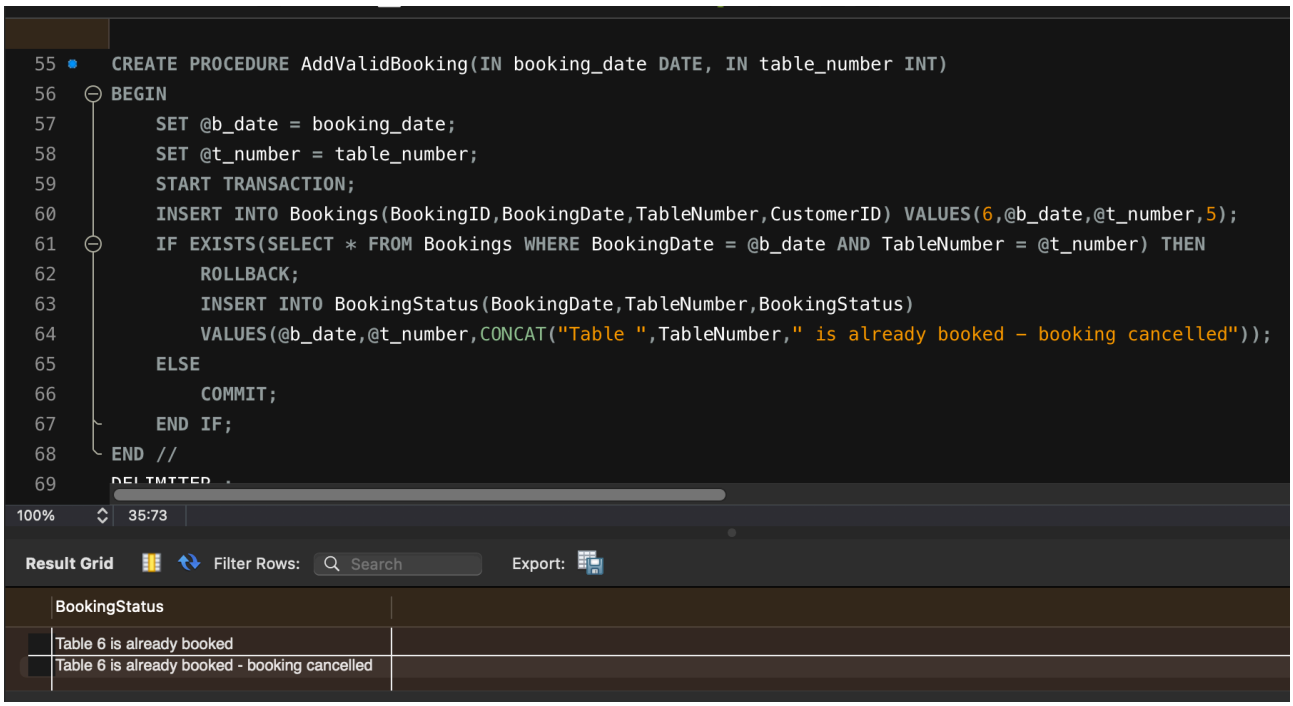
Limit to 1000 rows

```sql
DELIMITER //

CREATE PROCEDURE CheckBooking(IN booking_date DATE, IN table_number INT)
BEGIN
SELECT BookingStatus FROM BookingStatus WHERE BookingDate = booking_date AND TableNumber = table_number;
END //

DELIMITER ;
CALL CheckBooking("2022-11-12",3);

-- DROP PROCEDURE CheckBooking;
```

100% 35:48

Result Grid Filter Rows: Search Export:

BookingStatus
Table 3 is already booked



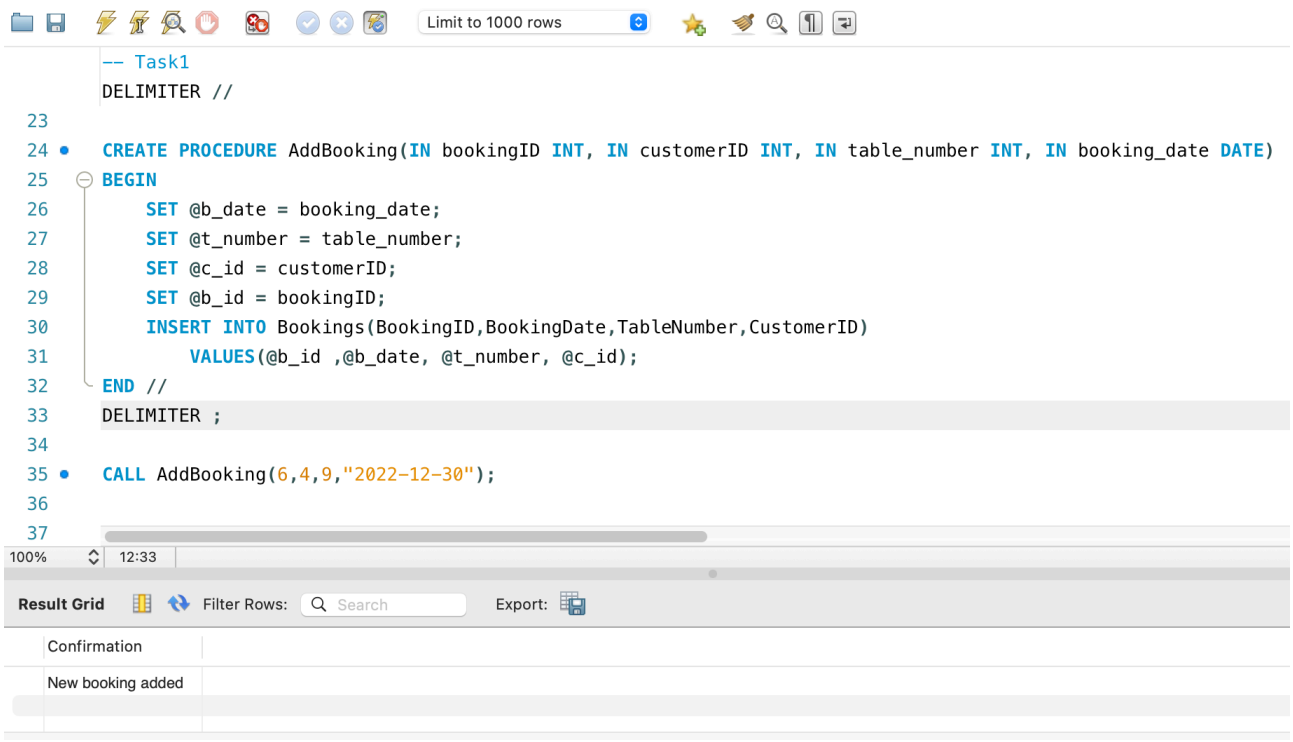
```
CREATE PROCEDURE AddValidBooking(IN booking_date DATE, IN table_number INT)
BEGIN
    SET @b_date = booking_date;
    SET @t_number = table_number;
    START TRANSACTION;
    INSERT INTO Bookings(BookingID,BookingDate,TableNumber,CustomerID) VALUES(6,@b_date,@t_number,5);
    IF EXISTS(SELECT * FROM Bookings WHERE BookingDate = @b_date AND TableNumber = @t_number) THEN
        ROLLBACK;
        INSERT INTO BookingStatus(BookingDate,TableNumber,BookingStatus)
        VALUES(@b_date,@t_number,CONCAT("Table ",TableNumber," is already booked - booking cancelled"));
    ELSE
        COMMIT;
    END IF;
END //
DELIMITER ;
```

100% 35:73

Result Grid Filter Rows: Search Export:

BookingStatus
Table 6 is already booked
Table 6 is already booked - booking cancelled

Exercise: Create SQL queries to add and update bookings



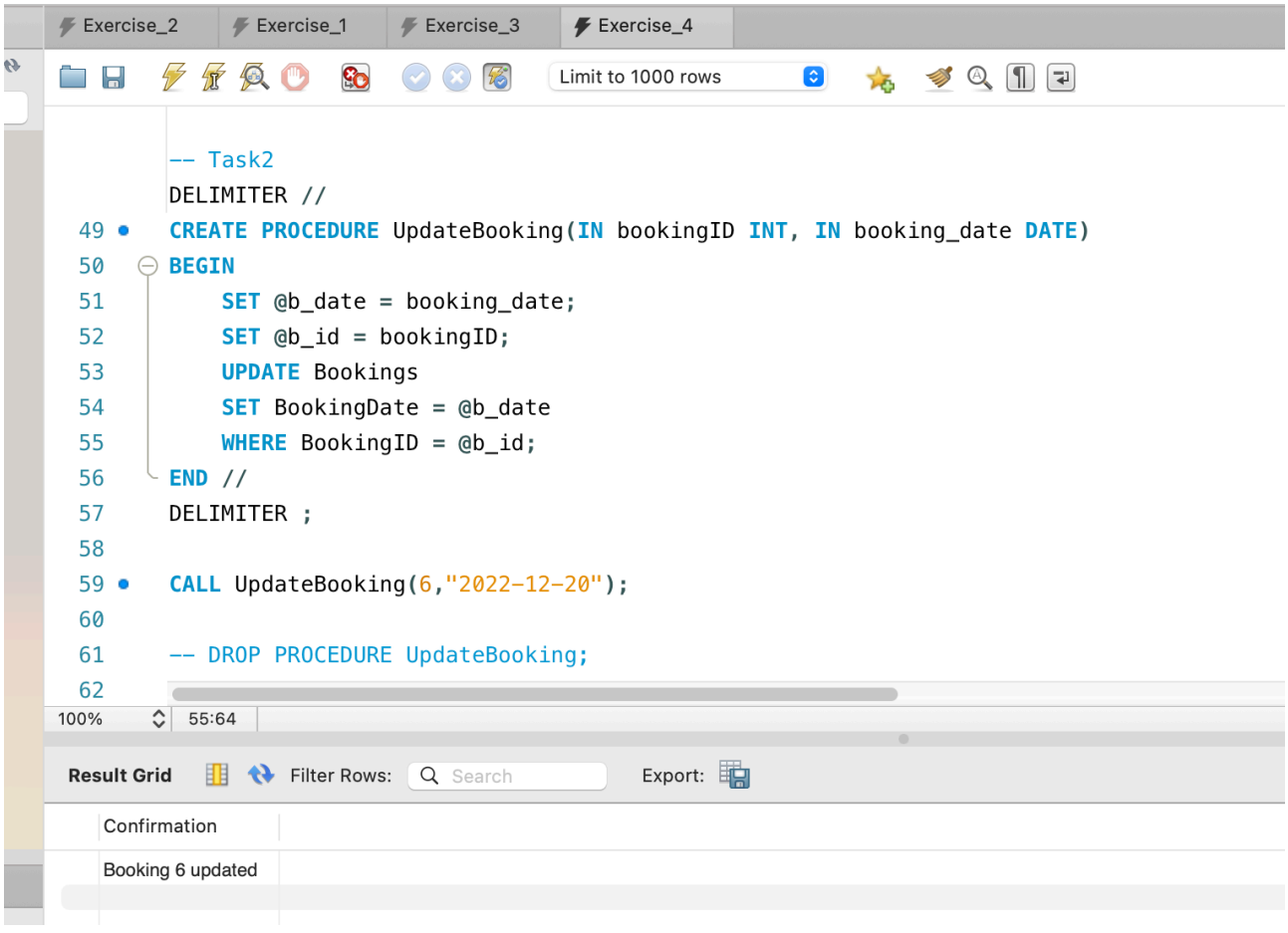
```
-- Task1
DELIMITER //

CREATE PROCEDURE AddBooking(IN bookingID INT, IN customerID INT, IN table_number INT, IN booking_date DATE)
BEGIN
    SET @b_date = booking_date;
    SET @t_number = table_number;
    SET @c_id = customerID;
    SET @b_id = bookingID;
    INSERT INTO Bookings(BookingID,BookingDate,TableNumber,CustomerID)
        VALUES(@b_id ,@b_date, @t_number, @c_id);
END //
DELIMITER ;

CALL AddBooking(6,4,9,"2022-12-30");
```

Limit to 1000 rows

100% 12:33

Result Grid Filter Rows: Search Export:

Confirmation
New booking added



Exercise_2 Exercise_1 Exercise_3 Exercise_4

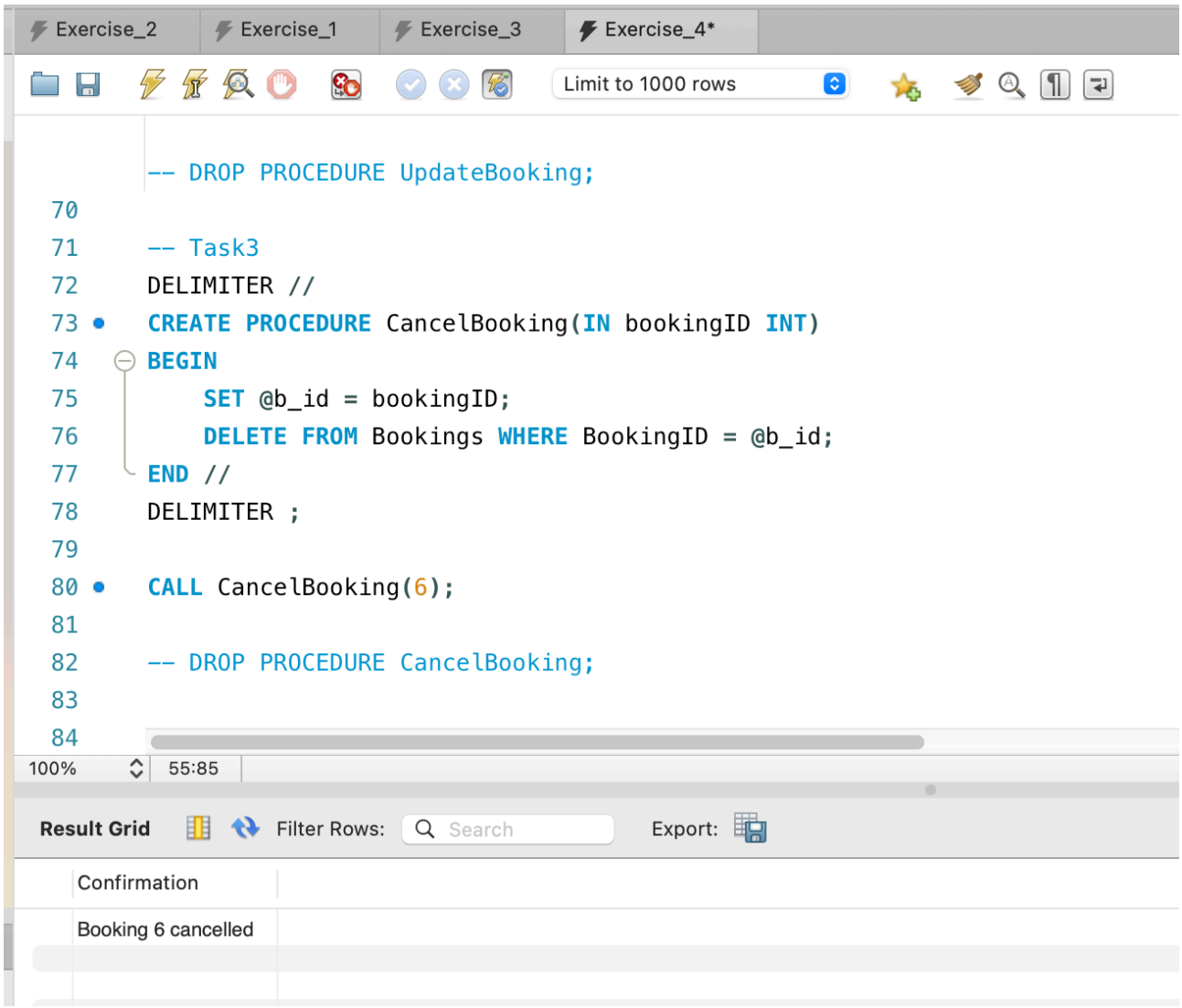
Limit to 1000 rows

```
-- Task2
DELIMITER //
CREATE PROCEDURE UpdateBooking(IN bookingID INT, IN booking_date DATE)
BEGIN
    SET @b_date = booking_date;
    SET @b_id = bookingID;
    UPDATE Bookings
    SET BookingDate = @b_date
    WHERE BookingID = @b_id;
END //
DELIMITER ;

CALL UpdateBooking(6,"2022-12-20");

-- DROP PROCEDURE UpdateBooking;
```

100% 55:64

Result Grid Filter Rows: Search Export:

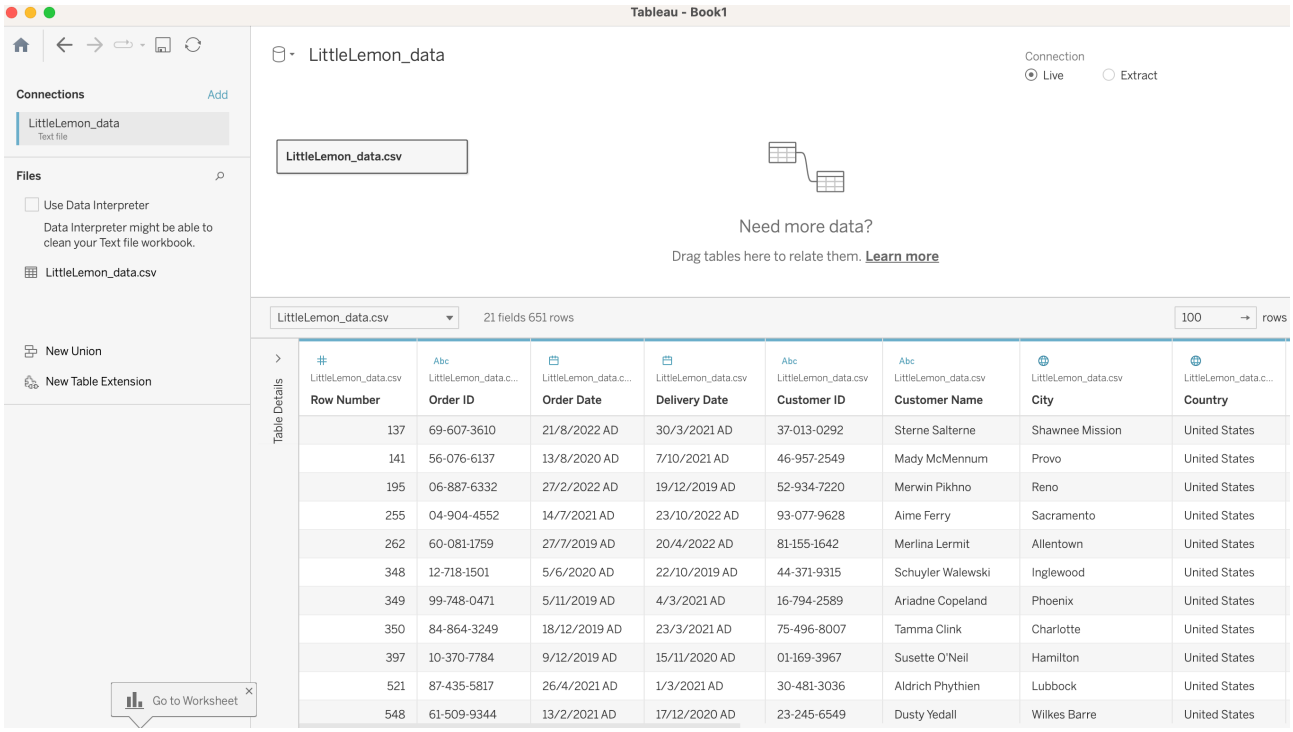
Confirmation
Booking 6 updated


Exercise_2
Exercise_1
Exercise_3
Exercise_4*
Limit to 1000 rows
-- DROP PROCEDURE UpdateBooking;
-- Task3
DELIMITER //
CREATE PROCEDURE CancelBooking(IN bookingID INT)
BEGIN
SET @b_id = bookingID;
DELETE FROM Bookings WHERE BookingID = @b_id;
END //
DELIMITER ;
CALL CancelBooking(6);
-- DROP PROCEDURE CancelBooking;
100%
55:85
Result Grid
Filter Rows:
Search
Export:
Confirmation
Booking 6 cancelled

LittleLemon Data Analysis with Tableau

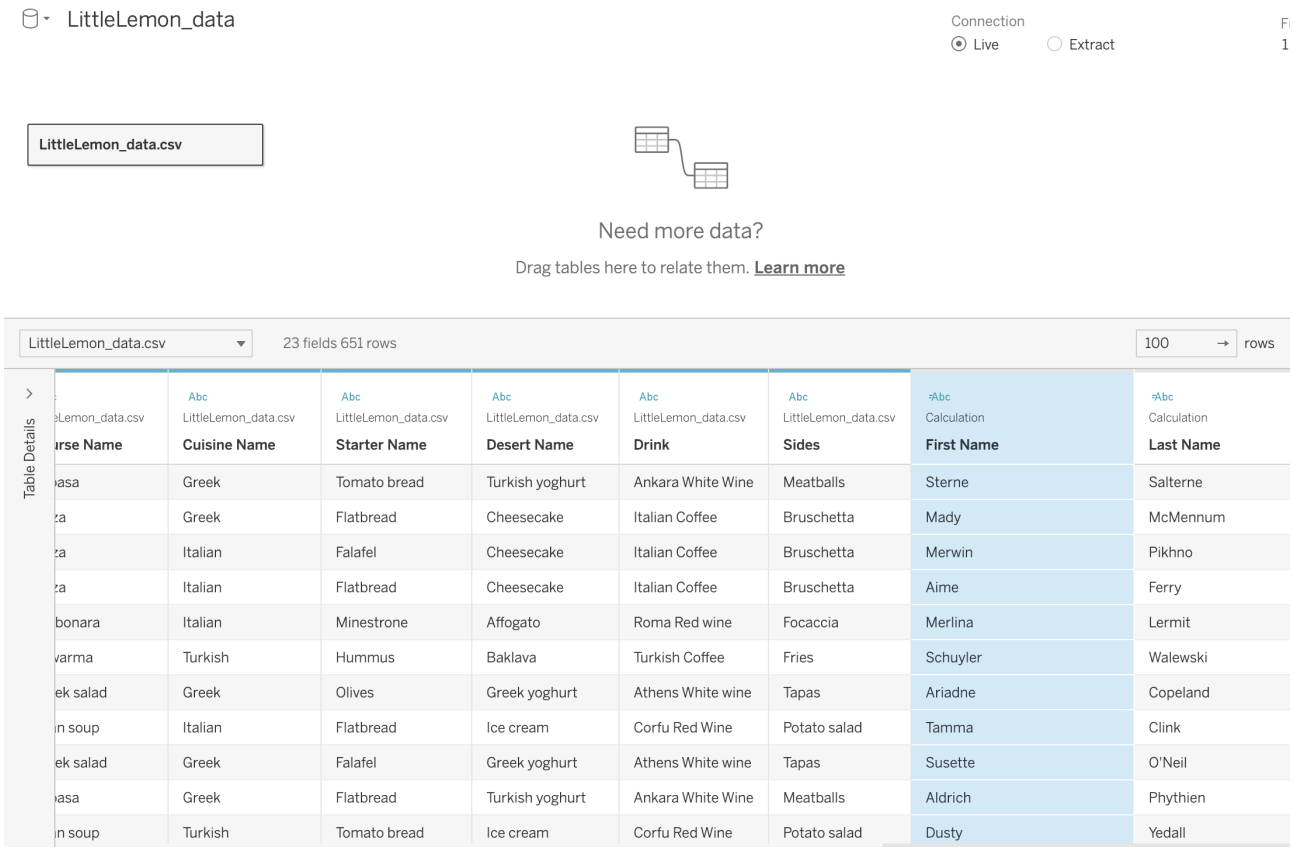
Exercise: Set up the Tableau Workspace for data analysis

Task 1



Row Number	Order ID	Order Date	Delivery Date	Customer ID	Customer Name	City	Country
137	69-607-3610	21/8/2022 AD	30/3/2021 AD	37-013-0292	Sterne Salterne	Shawnee Mission	United States
141	56-076-6137	13/8/2020 AD	7/10/2021 AD	46-957-2549	Mady McMennum	Provo	United States
195	06-887-6332	27/2/2022 AD	19/12/2019 AD	52-934-7220	Merwin Pikhno	Reno	United States
255	04-904-4552	14/7/2021 AD	23/10/2022 AD	93-077-9628	Aime Ferry	Sacramento	United States
262	60-081-1759	27/7/2019 AD	20/4/2022 AD	81-155-1642	Merlina Lermit	Allentown	United States
348	12-718-1501	5/6/2020 AD	22/10/2019 AD	44-371-9315	Schuyler Walewski	Inglewood	United States
349	99-748-0471	5/11/2019 AD	4/3/2021 AD	16-794-2589	Ariadne Copeland	Phoenix	United States
350	84-864-3249	18/12/2019 AD	23/3/2021 AD	75-496-8007	Tamma Clink	Charlotte	United States
397	10-370-7784	9/12/2019 AD	15/11/2020 AD	01-169-3967	Susette O'Neil	Hamilton	United States
521	87-435-5817	26/4/2021 AD	1/3/2021 AD	30-481-3036	Aldrich Phythien	Lubbock	United States
548	61-509-9344	13/2/2021 AD	17/12/2020 AD	23-245-6549	Dusty Yedall	Wilkes Barre	United States

Task 2



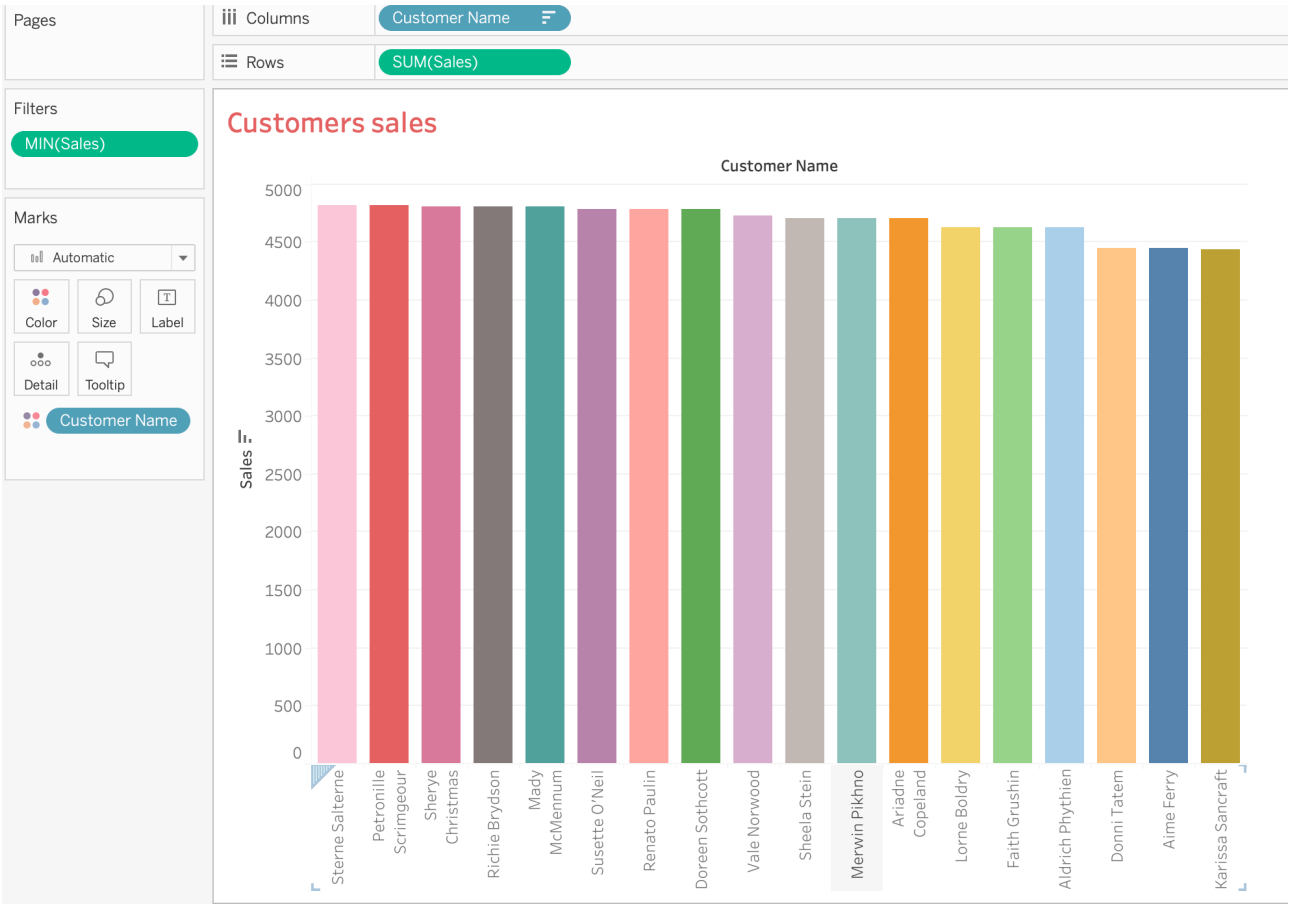
Abc LittleLemon_data.csv urse Name	Abc LittleLemon_data.csv Cuisine Name	Abc LittleLemon_data.csv Starter Name	Abc LittleLemon_data.csv Desert Name	Abc LittleLemon_data.csv Drink	Abc LittleLemon_data.csv Sides	=Abc Calculation First Name	=Abc Calculation Last Name
asa	Greek	Tomato bread	Turkish yoghurt	Ankara White Wine	Meatballs	Sterne	Salterne
za	Greek	Flatbread	Cheesecake	Italian Coffee	Bruschetta	Mady	McMennum
za	Italian	Falafel	Cheesecake	Italian Coffee	Bruschetta	Merwin	Pikhno
za	Italian	Flatbread	Cheesecake	Italian Coffee	Bruschetta	Aime	Ferry
bonara	Italian	Minestrone	Affogato	Roma Red wine	Focaccia	Merlina	Lermit
varma	Turkish	Hummus	Baklava	Turkish Coffee	Fries	Schuyler	Walewski
ek salad	Greek	Olives	Greek yoghurt	Athens White wine	Tapas	Ariadne	Copeland
n soup	Italian	Flatbread	Ice cream	Corfu Red Wine	Potato salad	Tamma	Clink
ek salad	Greek	Falafel	Greek yoghurt	Athens White wine	Tapas	Susette	O'Neil
asa	Greek	Flatbread	Turkish yoghurt	Ankara White Wine	Meatballs	Aldrich	Phythien
n soup	Turkish	Tomato bread	Ice cream	Corfu Red Wine	Potato salad	Dusty	Yedall

Task 3

a.csv e	# LittleLemon_data.c... Cost	# LittleLemon_data.c... Sales	=# Calculation Profit	
	190.000	285.000	95.000	
	144.090	216.135	72.045	
	125.000	187.500	62.500	
	91.840	137.760	45.920	
	235.000	352.500	117.500	
	119.800	179.700	59.900	
	125.000	187.500	62.500	
	235.000	352.500	117.500	
	320.000	480.000	160.000	
	52.080	78.120	26.040	
	235.000	352.500	117.500	

Exercise: Create interactive dashboard for sales and profits

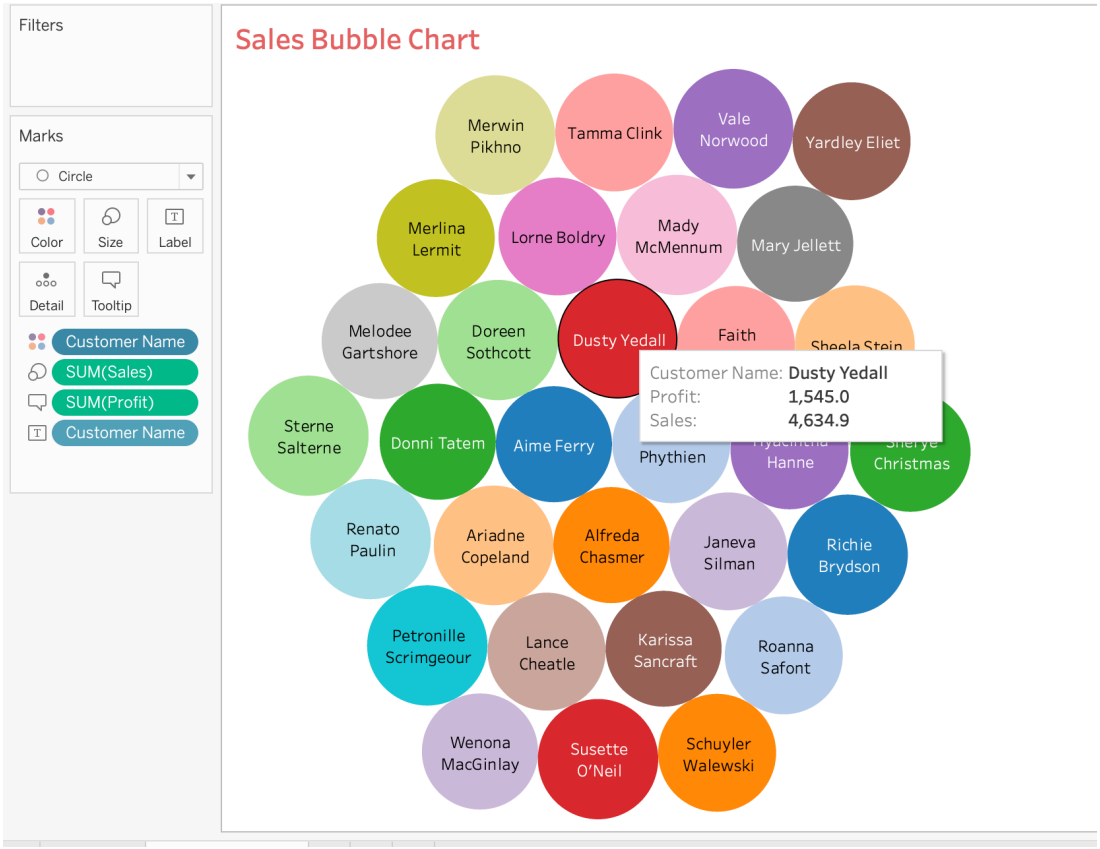
Task 1



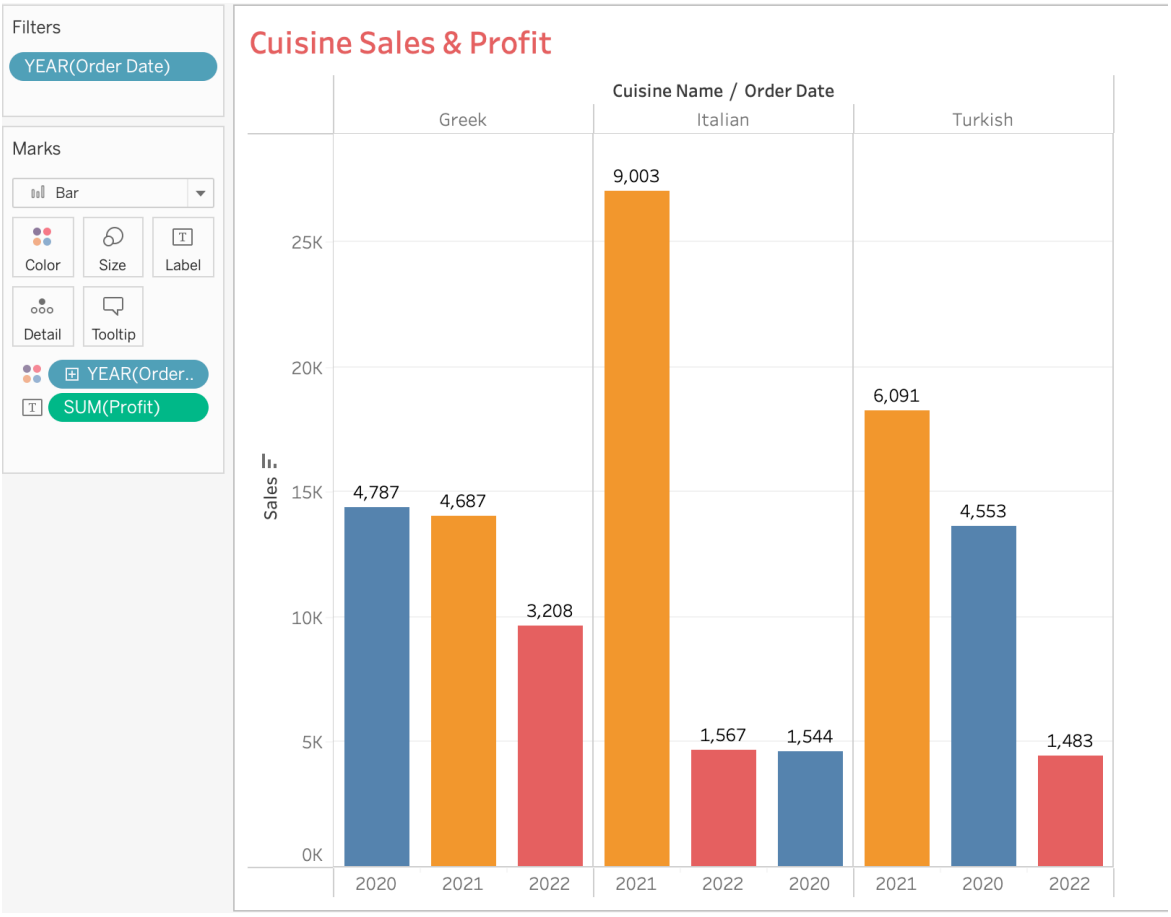
Task 2


Profit Chart
Order Date
Profit
20K
18K
16K
14K
12K
10K
8K
6K
4K
2K
0K
2019
2020
2021
2022

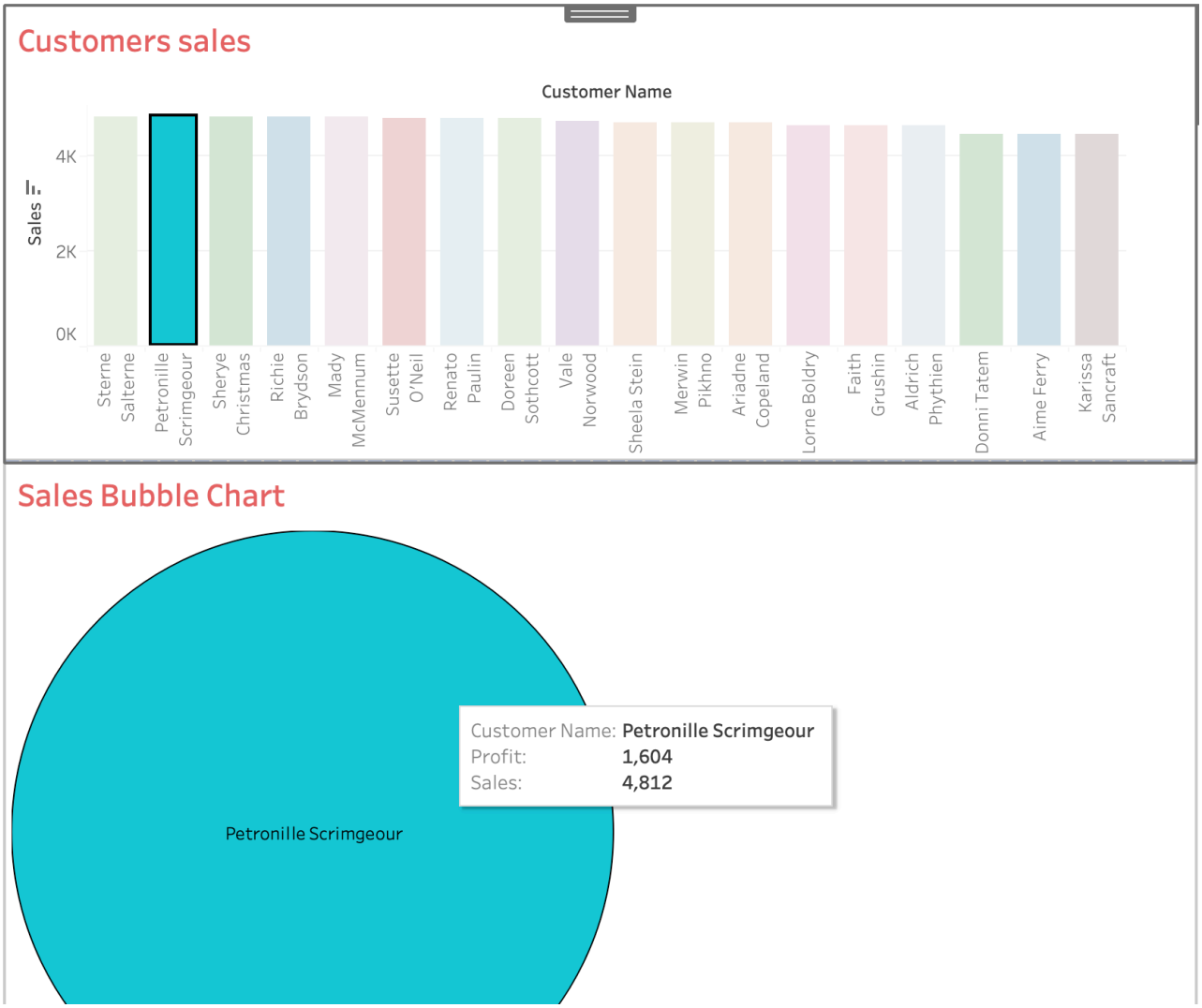
Task 3


Filters
Marks
Circle
Color
Size
Label
Detail
Tooltip
Customer Name
SUM(Sales)
SUM(Profit)
Customer Name
Sales Bubble Chart
Merwin Pikhno
Tamma Clink
Vale Norwood
Yardley Eliet
Merlina Lermit
Lorne Boldry
Mady McMennum
Mary Jellett
Melodee Gartshore
Doreen Sothcott
Dusty Yedall
Faith
Sheela Stein
Customer Name: Dusty Yedall
Profit: 1,545.0
Sales: 4,634.9
Sterne Salterne
Donni Tatem
Aime Ferry
Phythien
Hanne
Christmas
Renato Paulin
Ariadne Copeland
Alfreda Chasmer
Janeva Silman
Richie Brydson
Petronille Scrimgeour
Lance Cheatle
Karissa Sancraft
Roanna Safont
Wenona MacGinlay
Susette O'Neil
Schuyler Walewski

Task 4


Filters
YEAR(Order Date)
Marks
Bar
Color
Size
Label
Detail
Tooltip
YEAR(Order..
SUM(Profit)
Cuisine Sales & Profit
Cuisine Name / Order Date
Greek
Italian
Turkish
Sales
25K
20K
15K
10K
5K
0K
4,787
4,687
3,208
9,003
1,567
1,544
6,091
4,553
1,483
2020
2021
2022
2021
2022
2020
2021
2020
2022

Task 5


Customers sales
Customer Name
Sales
4K
2K
0K
Sterne Salterne
Petronille Scrimgeour
Sherye Christmas
Richie Brydson
Mady McMennum
Susette O'Neil
Renato Paulin
Doreen Sothcott
Vale Norwood
Sheela Stein
Merwin Pikhno
Ariadne Copeland
Lorne Boldry
Faith Grushin
Aldrich Phythien
Donni Tatem
Aime Ferry
Karissa Sancraft
Sales Bubble Chart
Petronille Scrimgeour
Customer Name: Petronille Scrimgeour
Profit: 1,604
Sales: 4,812

Exercise: Set up the client project

Task 1

Task 2

Jupyter notebook installed.

Task 3

`mysql-connector-python installed.`

Exercise: Add query functions

Task 1 & Task 2

```
In [7]: # Import mysql connector and connect to the data base
        import mysql.connector as connector

        connection=connector.connect(user="root",password="",db = "LittleLemonDB")

        cursor=connection.cursor()

In [8]: show_tables_query = "SHOW tables"
        cursor.execute(show_tables_query)
        results = cursor.fetchall()

In [9]: print(results)

        [('Bookings',), ('BookingStatus',), ('Customers',), ('MenuItems',), ('Menus',), ('Orders',), ('OrderStatus',), ('or
        dersview',)]

In [ ]:
```

Task 3

In [10]:
```
query = '''SELECT Customers.CustomerName, Orders.Sales FROM Orders
        INNER JOIN Customers
        ON Customers.CustomerID = Orders.CustomerID
        WHERE Orders.Sales > 60'''
```

In [11]:
```
cursor.execute(query)
results = cursor.fetchall()
```

In [14]:
```
print(cursor.column_names)
for i in range(10): #printout only first 10 results
    print(results[i])
```

```
('CustomerName', 'Sales')
('Pansie Alldis', Decimal('252.41'))
('Xymenes Hands', Decimal('139.70'))
('Lulu Hastin', Decimal('352.50'))
('Salvidor Francesch', Decimal('315.00'))
('Noni Quickenden', Decimal('480.00'))
('Jilleen Foux', Decimal('186.93'))
('Grenville Hyne', Decimal('216.14'))
('Ram Sinclaire', Decimal('152.40'))
('Jilly Bursnoll', Decimal('330.00'))
('Gabe Boyan', Decimal('139.70'))
```

In [ ]: