

CREATE VIRTUAL TABLE TO SUMMARIZE DATA

TASK 1:

Little Lemon needs you to create a virtual table called OrdersView that focuses on OrderID, Quantity and Cost columns within the Orders table for all orders with a quantity greater than 2.

The screenshot shows the MySQL Workbench interface with the 'capstone-project' database selected. The 'Navigator' pane on the left shows the 'Littlelemondb' database structure, including tables like 'bookings', 'customers', 'menu', 'menu items', 'order delivery status', 'orders', 'staff', and views like 'ordersview'. The 'SQL File 3' editor contains the following SQL script:

```
1 • USE Littlelemondb;
2
3 • CREATE VIEW OrdersView AS
4   SELECT Order_id, Quantity, Total_cost
5   FROM Orders WHERE Quantity > 2;
```

The 'Output' pane at the bottom shows the execution results:

#	Time	Action	Message	Duration / Fetch
13	13:17:23	SELECT * FROM Littlelemondb.`order delivery status` LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
14	13:21:47	CREATE VIEW OrdersView AS SELECT Order_id, Quantity, Total_cost FROM Orders WHERE Quantity > 2	Error Code: 1046. No database selected Select the default DB to be used by double-clicking its name in the SC...	0.000 sec
15	13:22:07	USE Littlelemondb	0 row(s) affected	0.000 sec
16	13:22:07	CREATE VIEW OrdersView AS SELECT Order_id, Quantity, Total_cost FROM Orders WHERE Quantity > 2	0 row(s) affected	0.031 sec

The screenshot shows the MySQL Workbench interface with the 'capstone-project' database selected. The 'Navigator' pane on the left shows the 'Littlelemondb' database structure, including tables like 'bookings', 'customers', 'menu', 'menu items', 'order delivery status', 'orders', 'staff', and views like 'ordersview'. The 'SQL File 3' editor contains the following SQL script:

```
1 • SELECT * FROM ordersview;
```

The 'Result Grid' pane shows the data returned by the query:

Order_id	Quantity	Total_cost
2	4	98.50
3	5	25.00
4	3	72.30

The 'Output' pane at the bottom shows the execution results:

#	Time	Action	Message	Duration / Fetch
15	13:22:07	USE Littlelemondb	0 row(s) affected	0.000 sec
16	13:22:07	CREATE VIEW OrdersView AS SELECT Order_id, Quantity, Total_cost FROM Orders WHERE Quantity > 2	0 row(s) affected	0.031 sec
17	13:22:23	SELECT * FROM Littlelemondb.ordersview LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
18	13:22:31	SELECT * FROM ordersview LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

TASK 2:

Little Lemon needs information from four tables on all customers with orders that cost more than \$150.

The screenshot shows MySQL Workbench with a query window containing a complex JOIN query. The query selects customer information, order details, and menu items for orders with a total cost greater than \$150. The result grid shows two rows of data.

```
1 • SELECT c.customer_id AS "CustomerID", c.customer_name AS "FullName",
2   o.order_id AS "OrderID", o.total_cost AS "Cost", m.name AS "MenuName",
3   mi.course_name AS "CourseName", mi.starter_name AS "StarterName"
4 FROM customers c
5 JOIN bookings b ON c.customer_id = b.customer_id
6 JOIN orders o ON b.booking_id = o.booking_id
7 JOIN menu m ON o.menu_id = m.menu_id
8 JOIN 'menu items' mi ON m.menu_item_id = mi.menu_item_id
9 WHERE o.total_cost > 150
10 ORDER BY o.total_cost ASC;
```

CustomerID	FullName	OrderID	Cost	MenuName	CourseName	StarterName
4	Maddy	5	151.80	Vegan Delights	Main Course	Caesar Salad
1	John	1	245.75	Summer Delights	Appetizers	Garlic Shrimp

Table: orders
Columns: Order_id (int PK), Order_date (datetime), Quantity (int), Total_cost (decimal(10,2)), Booking_id (int), Staff_id (int), Menu_id (int).

TASK 3:

Little Lemon needs you to find all menu items for which more than 2 orders have been placed. You can carry out this task by creating a subquery that lists the menu names from the menus table for any order quantity with more than 2.

The screenshot shows MySQL Workbench with a query window containing a subquery. The query selects menu names from the menu table where the quantity is greater than 2. The result grid shows four rows of data.

```
1 • SELECT name AS "MenuName" FROM menu
2 WHERE menu_id = ANY (
3   SELECT menu_id
4   FROM orders
5   WHERE quantity > 2
6 )
```

MenuName
Chef's Specials
Sweet Temptations
Mediterranean Feast

Table: orders
Columns: Order_id (int PK), Order_date (datetime), Quantity (int), Total_cost (decimal(10,2)), Booking_id (int), Staff_id (int), Menu_id (int).

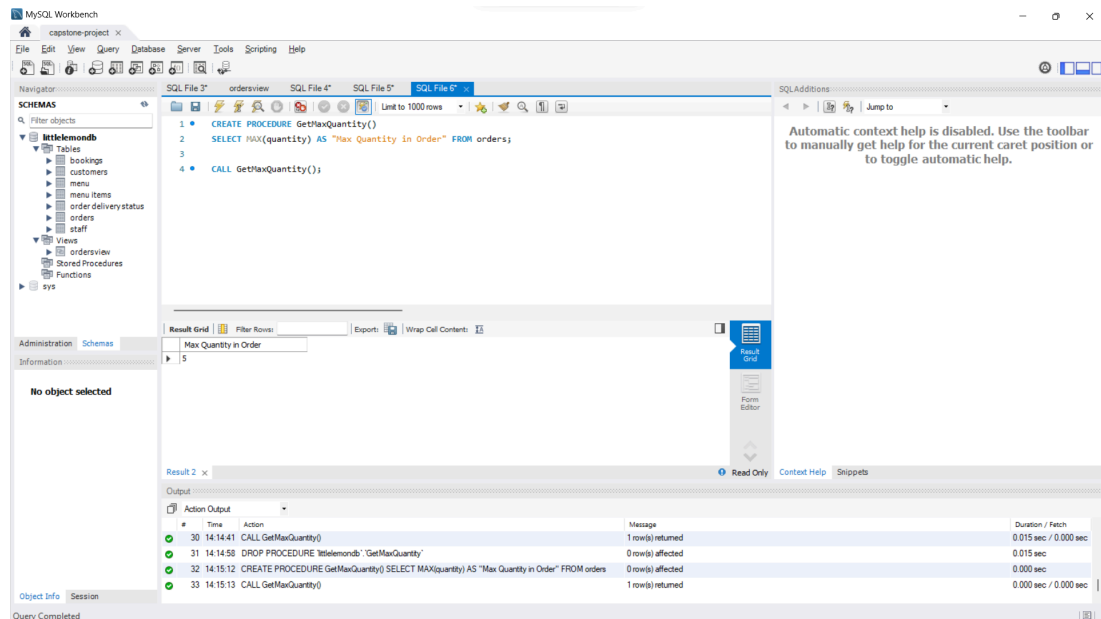
Output:

#	Time	Action	Message	Duration / Fetch
25	13:38:05	SELECT c.customer_id AS "CustomerID", c.customer_name AS "FullName", o.order_id AS "OrderID", o.total...	Error Code: 1054. Unknown column 'o.cost' in 'order clause'	0.000 sec
26	13:38:16	SELECT c.customer_id AS "CustomerID", c.customer_name AS "FullName", o.order_id AS "OrderID", o.total...	2 row(s) returned	0.000 sec / 0.000 sec
27	13:45:57	SELECT name FROM menu WHERE menu_id = ANY (SELECT menu_id FROM orders WHERE quantity...	3 row(s) returned	0.015 sec / 0.000 sec
28	13:46:13	SELECT name AS "MenuName" FROM menu WHERE menu_id = ANY (SELECT menu_id FROM orders ...	3 row(s) returned	0.000 sec / 0.000 sec

CREATE OPTIMIZED QUERIES TO MANAGE AND ANALYZE DATA

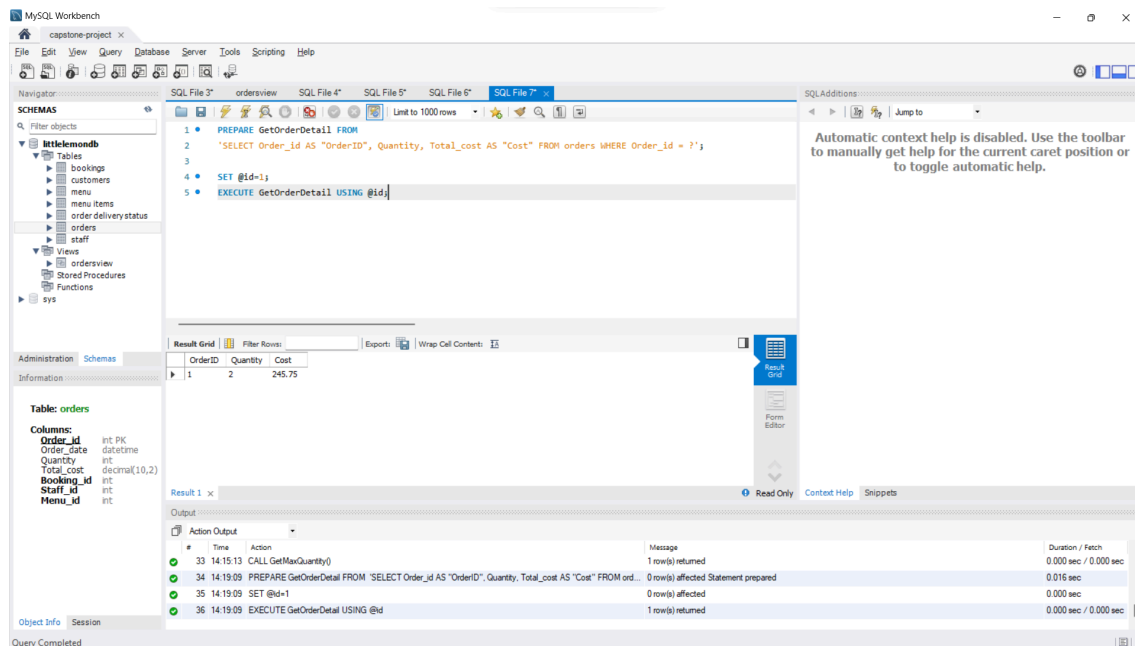
Task 1:

Little Lemon needs you to create a procedure that displays the maximum ordered quantity in the Orders table.



Task 2:

Little Lemon needs you to help them to create a prepared statement called GetOrderDetail. This prepared statement will help to reduce the parsing time of queries.



Task 3:

Create a stored procedure called CancelOrder. Little Lemon wants to use this stored procedure to delete an order record based on the user input of the order id.

The screenshot shows the MySQL Workbench interface with the 'capstone-project' database selected. The 'Schemas' pane on the left shows the database structure, including tables like 'bookings', 'customers', 'menu', 'order delivery status', 'orders', and 'staff'. The main editor window displays the SQL code for creating the 'CancelOrder' stored procedure. The code is as follows:

```
1 DELIMITER //
2
3 CREATE PROCEDURE CancelOrder(IN id INT)
4 BEGIN
5     DECLARE confirmationmsg VARCHAR(255);
6     IF EXISTS (SELECT * FROM Orders WHERE order_id = id) THEN
7         DELETE FROM orders WHERE order_id = id;
8         SET confirmationmsg = CONCAT('Order ', id, ' is cancelled');
9     ELSE
10        SET confirmationmsg = CONCAT('Order ', id, ' does not exist');
11    END IF;
12    SELECT 'Confirmation';
13    SELECT confirmationmsg;
14 END //
15
16 DELIMITER ;
17
18 call CancelOrder(3);
```

The 'Result Grid' shows the output of the stored procedure call, displaying the confirmation message: 'confirmationmsg' and 'Order 3 is cancelled'.

The 'Action Output' pane at the bottom shows the execution details:

#	Time	Action	Message	Duration / Fetch
46	14:33:16	call CancelOrder(4)	1 row(s) returned	- / 0.000 sec

The status bar at the bottom indicates 'Query Completed'.