**Database Engineer Capstone Project**

**Content**

This document contains the solutions to the following exercises and tasks:

Exercise-1: Committing the project (Task-1,2,3)

Exercise-2: Create a virtual table to summarize data (Task-1,2,3)

Exercise-3: Create optimized queries to manage and analyze data

**Evidences overview**

The evidence files are uploaded in Git. The below table provides a mapping between the tasks and the related evidence.

Github link: <https://github.com/annareuss/db-capstone-project/tree/main>

Exercise Task Description Evidence filename in Git

-------- ------- ------------------------------------- -----------------------------------------------------

Exercise-1 Task-1 Create a normalized ER diagram LittleLemonDM.png

Task-2 Implement the Little Lemon data model LittleLemonDB.sql

Task-3 Write ‘show databases’ SQL code show\_databases\_result.png

Exercise-2Task-1 Create a virtual table OrdersView.png

Task-2 Create a JOIN join.png

Task-3 Use the ANY operator in a subquery any.png

Exercise-3Task-1 Create stored procedure GetMaxQuantity stored\_procedure\_GetMaxQuantity.png

Task-2 Create prepared statement GetOrderDetail prepared\_statement\_GetOrderDetail.png

Task-3 Create stored procedure CancelOrder stored\_procedure\_CancelOrder.png

**Exercise-1: Committing the project**

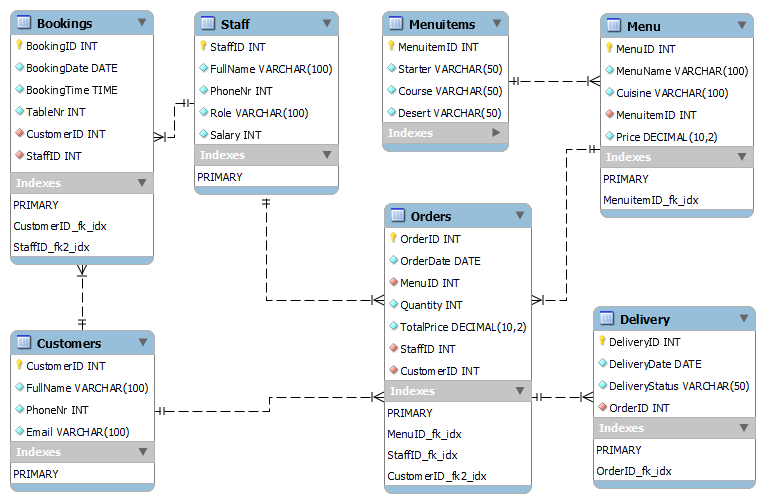
**Task-1 Create a normalized ER diagram**

First the conceptual model, then the logical model, finally the physical model was created.

The data type and length was considered for all columns. Primary and foreign keys were designed.

The normalization process steps were performed. The final version is in 3rd normal form.

**Evidence(s):**

****

**Task-2 Implement the Little Lemon data model**

The data model was implemented using the forward engineer feature in MySQL Workbench.

**Evidence(s):**



**Task-3 Write the ‘show databases’ SQL code**

Create SQL statement to show the databases.

The SQL statement and the result was also captured.

The created database **littlelemon** is highlighted in the picture.

**Evidence(s):**

A screenshot of a computer

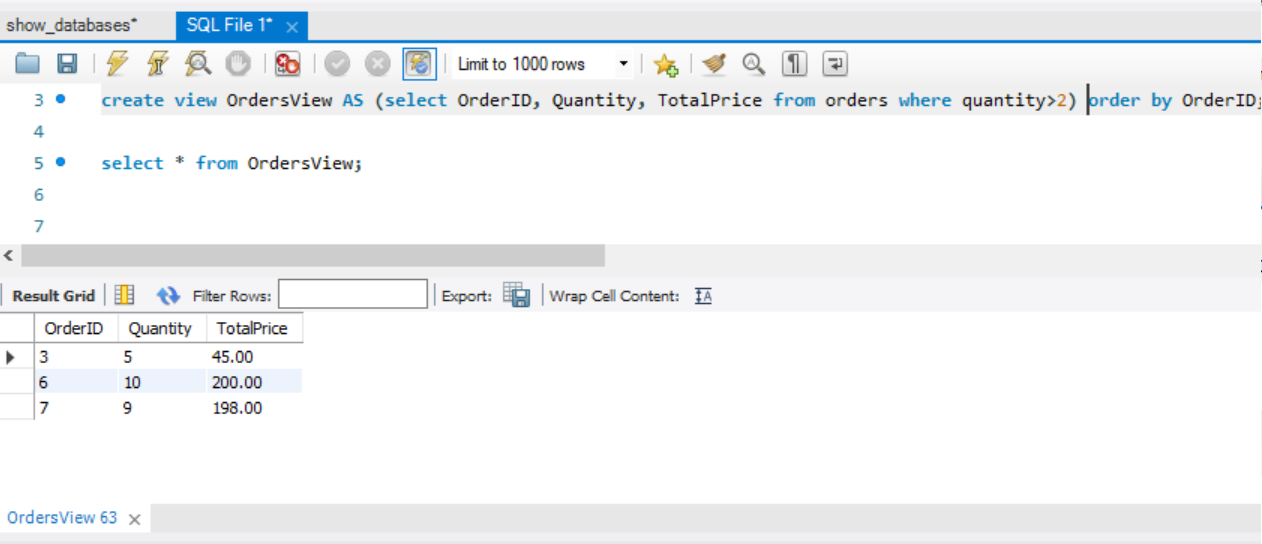
Description automatically generated

**Exercise-2: Create a virtual table to summarize data**

**Task-1 Create a virtual table**

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

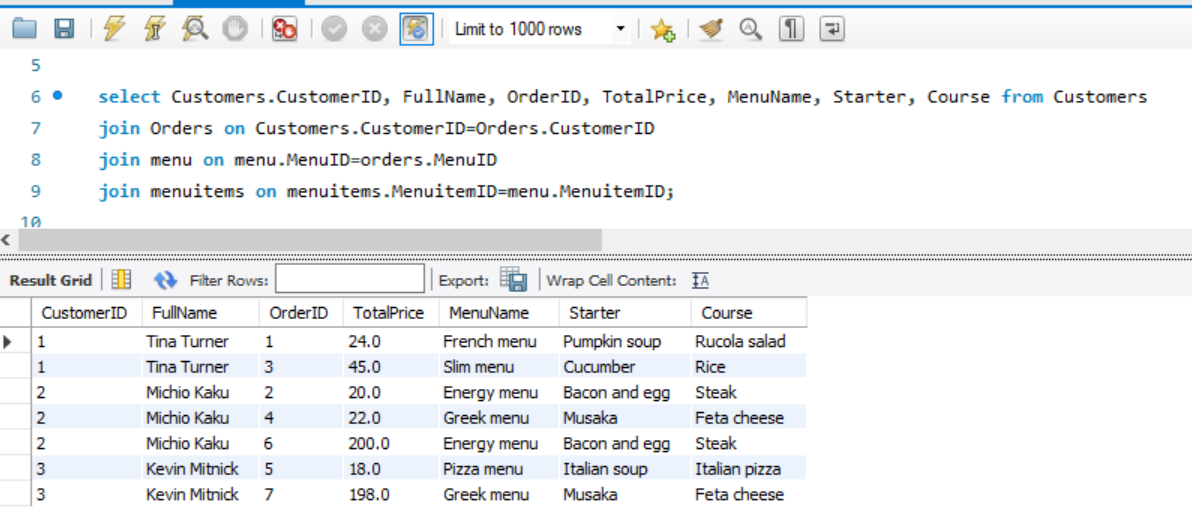
**Evidence(s):**



**Task-2 Create a JOIN**

Query the information from four tables on all customers with orders that cost more than $150. Extract the information by using the relevant JOIN clause.

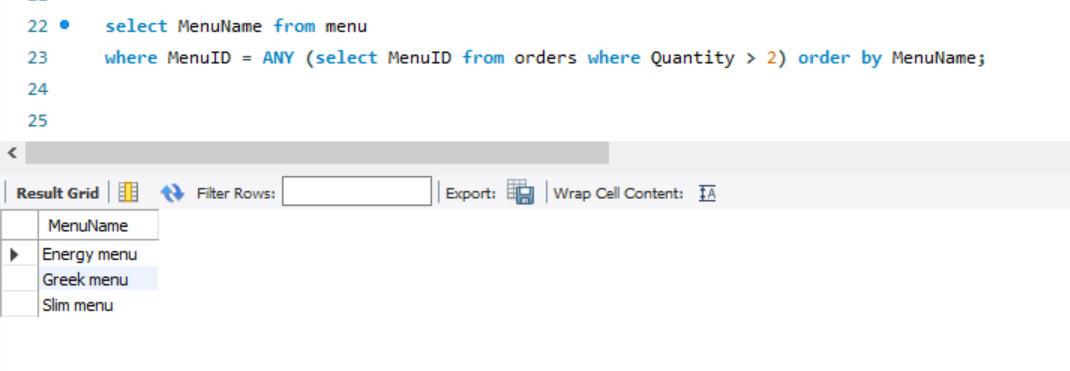
**Evidence(s):**



**Task-3 Use the ANY operator in a subquery**

Prepare a query in order to find all menu items for which more than 2 orders have been placed.

**Evidence(s):**

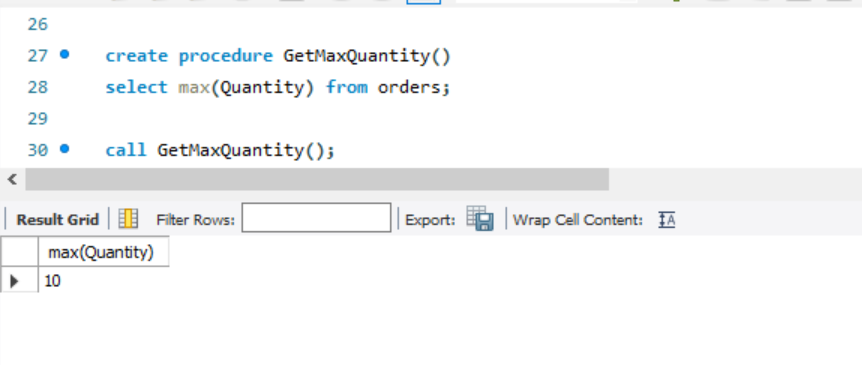


# Exercise-3: Create optimized queries to manage and analyze data

**Task-1 Create a procedure to show the max order quantity**

Create a procedure that displays the maximum ordered quantity in the Orders table. The name of the procedure should be **GetMaxQuantity**.

**Evidence(s):**



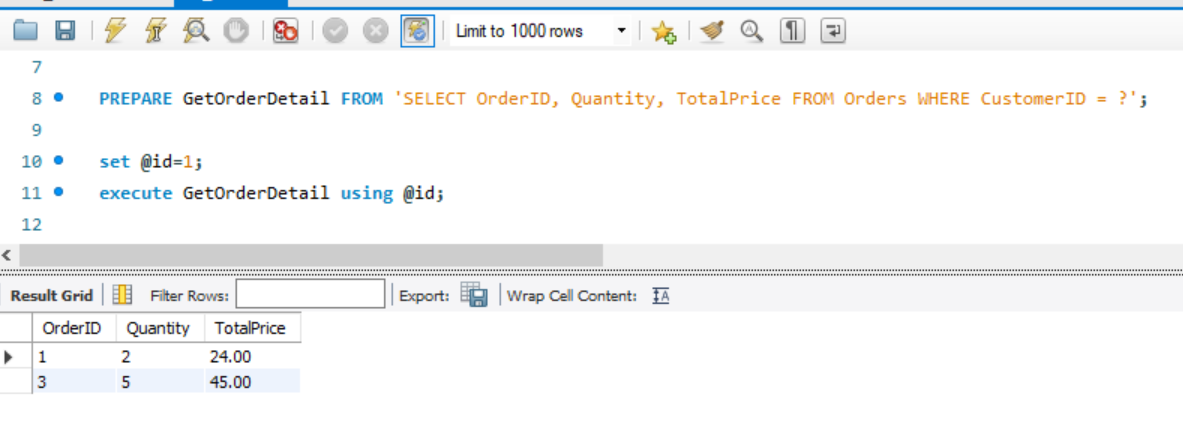
**Task-2 Create prepared statement GetOrderDetail**

Create a procedure that displays order information for a specified CustomerID. The name of the prepared statement should be **GetOrderDetail**.

The prepared statement should accept one input argument, the **CustomerID** value, from a variable.

The statement should return the order id, the quantity and the order cost from the Orders table.

**Evidence(s):**



**Task-3 Create a stored procedure to cancel order**

Create a stored procedure to delete an order record based on the user input of the order id.

The name of the stored procedure should be **CancelOrder**.

I created a delete trigger on the orders table, and an audit table which logs the delete trigger results.

The procedure deletes the given row, and the trigger writes the event into the audit log. Then the audit log is shown in the second step.

--------------------------------------------------------------------

CREATE TABLE Audit (

Confirmation VARCHAR(100) NOT NULL,

Date DATE NOT NULL);

CREATE TRIGGER AfterDeleteOrder AFTER DELETE ON Orders FOR EACH ROW

INSERT INTO Audit VALUES (CONCAT('Order ',OLD.OrderID,' is cancelled'),CURRENT\_DATE());

delimiter //

CREATE PROCEDURE CancelOrder(OrderIDinput INT)

begin

delete from orders where OrderID=OrderIDinput;

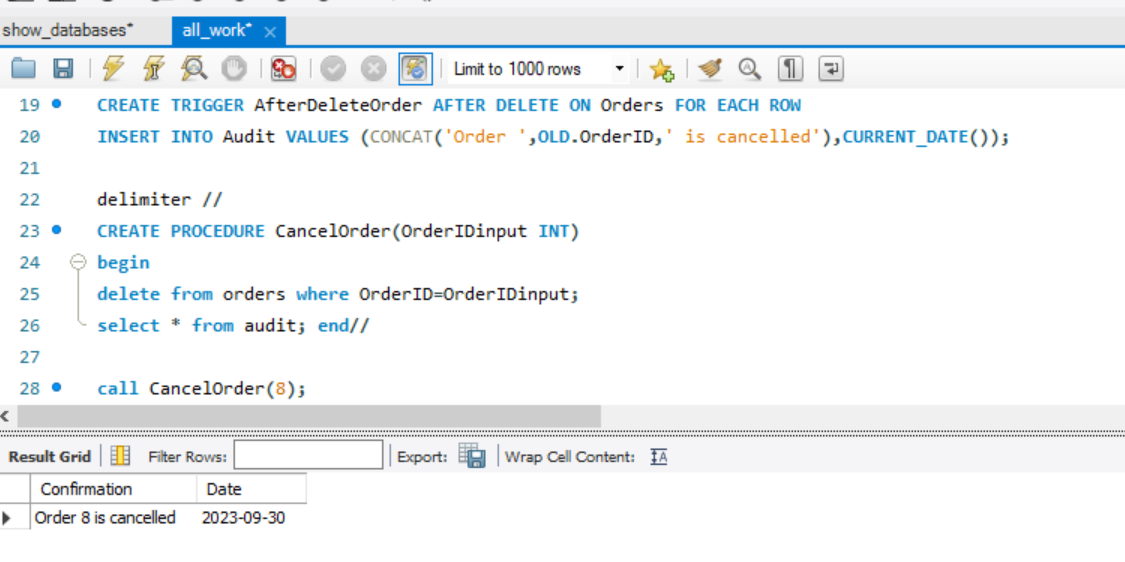
select \* from audit;

end//

call CancelOrder(8);

--------------------------------------------------------------------

**Evidence(s):**



# Appendix

**insert data**

(run after the tables are created, and then in case the tables are dropped and need to be populated again)

--------------------------------------------------------------------

use littlelemon;

insert into staff (StaffID, FullName, PhoneNr, Role, Salary) values

(0,'Bill Gates','01234567','waiter',1200),

(0,'Diana Ross','15644409','singer',5000),

(0,'George Hamilton','91562368','accountant',2500);

select \* from staff;

insert into menuitems (MenuitemID, Starter, Course, Desert) values

(0,'Pumpkin soup','Rucola salad','Fruit yogurt'),

(0,'Bacon and egg','Steak','Apple pie'),

(0,'Cucumber','Rice','Icecream'),

(0,'Musaka','Feta cheese','Vanile pudding'),

(0,'Italian soup','Italian pizza','Italian Cake');

select \* from menuitems;

INSERT INTO menu (MenuName, Cuisine, MenuitemID, Price) VALUES

('French menu', 'French', '1', '12.0'),

('Energy menu', 'American', '2', '20.0'),

('Slim menu', 'International', '3', '15.0'),

('Greek menu', 'Greek', '4', '11.0'),

('Pizza menu', 'Italian', '5', '18.0');

insert into customers values (0,'Tina Turner','221245671','tinaturner@gmail.com');

insert into customers values (0,'Michio Kaku','88644409','mkaku@gmail.com');

insert into customers values (0,'Kevin Mitnick','99562368','kmitnick@gmail.com');

select \* from customers;

insert into customers (CustomerID, FullName, PhoneNr, Email) values

(0,'Tina Turner','221245671','tinaturner@gmail.com'),

(0,'Michio Kaku','88644409','mkaku@gmail.com'),

(0,'Kevin Mitnick','99562368','kmitnick@gmail.com');

select \* from customers;

insert into orders (OrderID, OrderDate, MenuID, Quantity, TotalPrice, StaffID, CustomerID) values

(0,'2023-09-01',1,2,24.0,1,1),

(0,'2023-09-02',2,1,20.0,2,2),

(0,'2023-09-03',3,5,45.0,3,1),

(0,'2023-09-05',4,2,22.0,1,2),

(0,'2023-09-08',5,1,18.0,2,3),

(0,'2023-09-19',2,10,200.0,1,2),

(0,'2023-09-19',4,9,198.0,2,3);

select \* from orders;

insert into delivery(DeliveryID, DeliveryDate, DeliveryStatus, OrderID) values

(0,'2023-09-02','delivered',1),

(0,'2023-09-02','in\_progress',2),

(0,'2023-09-03','delivered',3),

(0,'2023-09-05','in\_progress',4),

(0,'2023-09-08','delivered',5);

select \* from delivery;

insert into bookings (BookingID, BookingDate, BookingTime, TableNr, CustomerID, StaffID) values

(0,'2023-09-05','18:00',3,1,2),

(0,'2023-09-05','19:00',5,2,2),

(0,'2023-09-06','12:00',3,3,1),

(0,'2023-09-08','17:00',9,1,3),

(0,'2023-09-15','16:00',3,2,2);

select \* from bookings;

**verify the insert**

--------------------------------------------------------------------

select \* from staff;

select \* from menuitems;

select \* from menu;

select \* from customers;

select \* from orders;

select \* from delivery;

select \* from bookings;

--------------------------------------------------------------------

--------------------------------------------------------------------

**drop tables**

(run in case the ER diagram or the database has to be re-created)

--------------------------------------------------------------------

use littlelemon;

drop table bookings;

drop table delivery;

drop table orders;

drop table menu;

drop table menuitems;

drop table staff;

drop table customers;

--------------------------------------------------------------------