**Database Installation and Setup: Little Lemon DB**

Getting your Little Lemon database up and running is straightforward. Just follow these steps:

**1. Install MySQL**

First things first, you'll need **MySQL** on your machine. If you haven't already, download and install it. This will provide the database server where your Little Lemon data will live.

**2. Download the SQL File**

Next, grab the **LittleLemonDB.sql** file. You can find this file in the project's repository. This single file contains all the necessary commands to create your database, its tables, and any stored procedures.

**3. Import and Execute in MySQL Workbench**

Once MySQL is installed and you have the SQL file, you'll use **MySQL Workbench** to set up the database:

* **Open MySQL Workbench**.
* Go to the **Server** menu at the top.
* Select **Data Import** from the dropdown.
* In the Data Import window, choose the option **Import from Self-Contained File**.
* Browse for and **load the LittleLemonDB.sql file** you downloaded earlier.
* Finally, click the **Start Import** button.

MySQL Workbench will now import and execute all the SQL commands contained within the file. After it finishes, your Little Lemon database will be fully set up, complete with all its tables and pre-configured stored procedures, ready for you to use.

* **GetMaxQuantity()**

This stored procedure is designed to extract the highest recorded quantity of a particular item ordered, serving as a valuable tool for optimizing inventory control and decision-making processes.

CREATE PROCEDURE GetMaxQuantity()

BEGIN

DECLARE maxQty INT;

SELECT MAX(Quantity) INTO maxQty FROM `LittleLemonDB`.`Orders`;

SELECT maxQty AS 'Maximum Ordered Quantity';

END;

CALL GetMaxQuantity()

* **CheckBooking()**

The CheckBooking stored procedure performs a validation to determine the booking status of a specific table for a given date. It returns a status message indicating whether the table is currently available or has already been reserved.

CREATE PROCEDURE `LittleLemonDB`.`CheckBooking`(IN booking\_date DATE, IN table\_number INT)

BEGIN

DECLARE table\_status VARCHAR(50);

SELECT COUNT(\*) INTO @table\_count

FROM `LittleLemonDB`.`Bookings`

WHERE `Date` = booking\_date AND `TableNumber` = table\_number;

IF (@table\_count > 0) THEN

SET table\_status = 'Table is already booked.';

ELSE

SET table\_status = 'Table is available.';

END IF;

SELECT table\_status AS 'Table Status';

END;

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

* **UpdateBooking()**

This stored procedure is responsible for updating booking records within the database by accepting the booking ID and a revised booking date as input parameters, thereby ensuring that all modifications are accurately reflected in the system.

CREATE PROCEDURE `LittleLemonDB`.`UpdateBooking`(

IN booking\_id\_to\_update INT,

IN new\_booking\_date DATE)

BEGIN

UPDATE `LittleLemonDB`.`Bookings`

SET `Date` = new\_booking\_date

WHERE `BookingID` = booking\_id\_to\_update;

SELECT CONCAT('Booking ', booking\_id\_to\_update, ' updated') AS 'Confirmation';

END;

CALL `LittleLemonDB`.`UpdateBooking`(9, '2022-11-15');

* **AddBooking()**

This procedure facilitates the insertion of a new booking record into the system by accepting multiple input parameters, including the booking ID, customer ID, booking date, and table number, to ensure the comprehensive registration of the reservation.

CREATE PROCEDURE `LittleLemonDB`.`AddBooking`(

IN new\_booking\_id INT,

IN new\_customer\_id INT,

IN new\_booking\_date DATE,

IN new\_table\_number INT,

IN new\_staff\_id INT)

BEGIN

INSERT INTO `LittleLemonDB`.`Bookings`(

`BookingID`,

`CustomerID`,

`Date`,

`TableNumber`,

`StaffID`)

VALUES(

new\_booking\_id,

new\_customer\_id,

new\_booking\_date,

new\_table\_number,

new\_staff\_id

);

SELECT 'New booking added' AS 'Confirmation';

END;

CALL `LittleLemonDB`.`AddBooking`(17, 1, '2022-10-10', 5, 2);

* **CancelBooking()**

This stored procedure removes a specified booking record from the database, thereby enhancing system manageability and optimizing resource allocation.

CREATE PROCEDURE `LittleLemonDB`.`CancelBooking`(IN booking\_id\_to\_cancel INT)

BEGIN

DELETE FROM `LittleLemonDB`.`Bookings`

WHERE `BookingID` = booking\_id\_to\_cancel;

SELECT CONCAT('Booking ', booking\_id\_to\_cancel, ' cancelled') AS 'Confirmation';

END;

CALL `LittleLemonDB`.`CancelBooking`(9);

* **AddValidBooking()**

The AddValidBooking stored procedure is designed to securely insert a new table booking record into the database. It initiates a transaction to ensure data integrity and consistency throughout the operation.

CREATE PROCEDURE `LittleLemonDB`.`AddValidBooking`(IN new\_booking\_date DATE, IN new\_table\_number INT, IN new\_customer\_id INT, IN new\_staff\_id INT)

BEGIN

DECLARE table\_status INT;

START TRANSACTION;

SELECT COUNT(\*) INTO table\_status

FROM `LittleLemonDB`.`Bookings`

WHERE `Date` = new\_booking\_date AND `TableNumber` = new\_table\_number;

IF (table\_status > 0) THEN

ROLLBACK;

SELECT 'Booking could not be completed. Table is already booked on the specified date.' AS 'Status';

ELSE

INSERT INTO `LittleLemonDB`.`Bookings`(`Date`, `TableNumber`, `CustomerID`, `StaffID`)

VALUES(new\_booking\_date, new\_table\_number, new\_customer\_id, new\_staff\_id);

COMMIT;

SELECT 'Booking completed successfully.' AS 'Status';

END IF;

END;

CALL AddValidBooking('2022-10-10', 5, 1, 1);

* **CancelOrder()**

The CancelOrder stored procedure is intended to revoke or delete a specific order based on its Order ID. It performs a DELETE operation to remove the corresponding record from the Orders table, ensuring the order is effectively canceled within the system.

CREATE PROCEDURE CancelOrder(IN orderIDToDelete INT)

BEGIN

DECLARE orderExistence INT;

SELECT COUNT(\*) INTO orderExistence FROM `LittleLemonDB`.`Orders` WHERE OrderID = orderIDToDelete;

IF orderExistence > 0 THEN

DELETE FROM `LittleLemonDB`.`OrderDeliveryStatuses` WHERE OrderID = orderIDToDelete;

DELETE FROM `LittleLemonDB`.`Orders` WHERE OrderID = orderIDToDelete;

SELECT CONCAT('Order ', orderIDToDelete, ' is cancelled') AS 'Confirmation';

ELSE

SELECT CONCAT('Order ', orderIDToDelete, ' does not exist') AS 'Confirmation';

END IF;

END;

CALL CancelOrder(5);