CODE FIRST GIRLS

SQL Project-

Restaurant Ordering Database

Prepared by:

Hai Ling Tan

29 August 2023

PROJECT OVERVIEW

I've chosen the restaurant ordering system as my theme for this project as this is one of the areas I'm most interested in as I really enjoy eating good food at restaurants.

As the food delivery industry is growing especially after the pandemic, having a structured database to store necessary data is essential for restaurants when it comes to handling delivery orders.

This is to ensure the accuracy and completeness of the order information for the delivery orders, as well to track the information of the payment and items in the menu.

Since the actual operations of a restaurant ordering system can be complicated, I've created a simplified database using the typical information that could be found in a restaurant's delivery database. The 6 tables created in this project covers the order, payment, customer, menu, user and chef details.

Please refer to the following page for the ER diagram for diagram showing the relationship between different tables within the database. Kindly refer to the remaining pages for the different instructions used to create and run the queries within the database.

Despite facing some challenges while completing this project, I still enjoyed every bit of this project as it allows me to incorporate my creativity to come up with different ideas for creating logical queries.

Restaurant Ordering Database

The Restaurant Ordering Database is designed to stores details about the orders placed, customer information, payment information, chef, menus offered and user information of the employees that created the order.

There are 6 tables in total, namely **ORDERS, CUSTOMER, PAYMENT, USER, MENU** and **CHEF**.

Kindly refer to the **ER Diagram** below, which shows the relationship between all the tables within the Database.

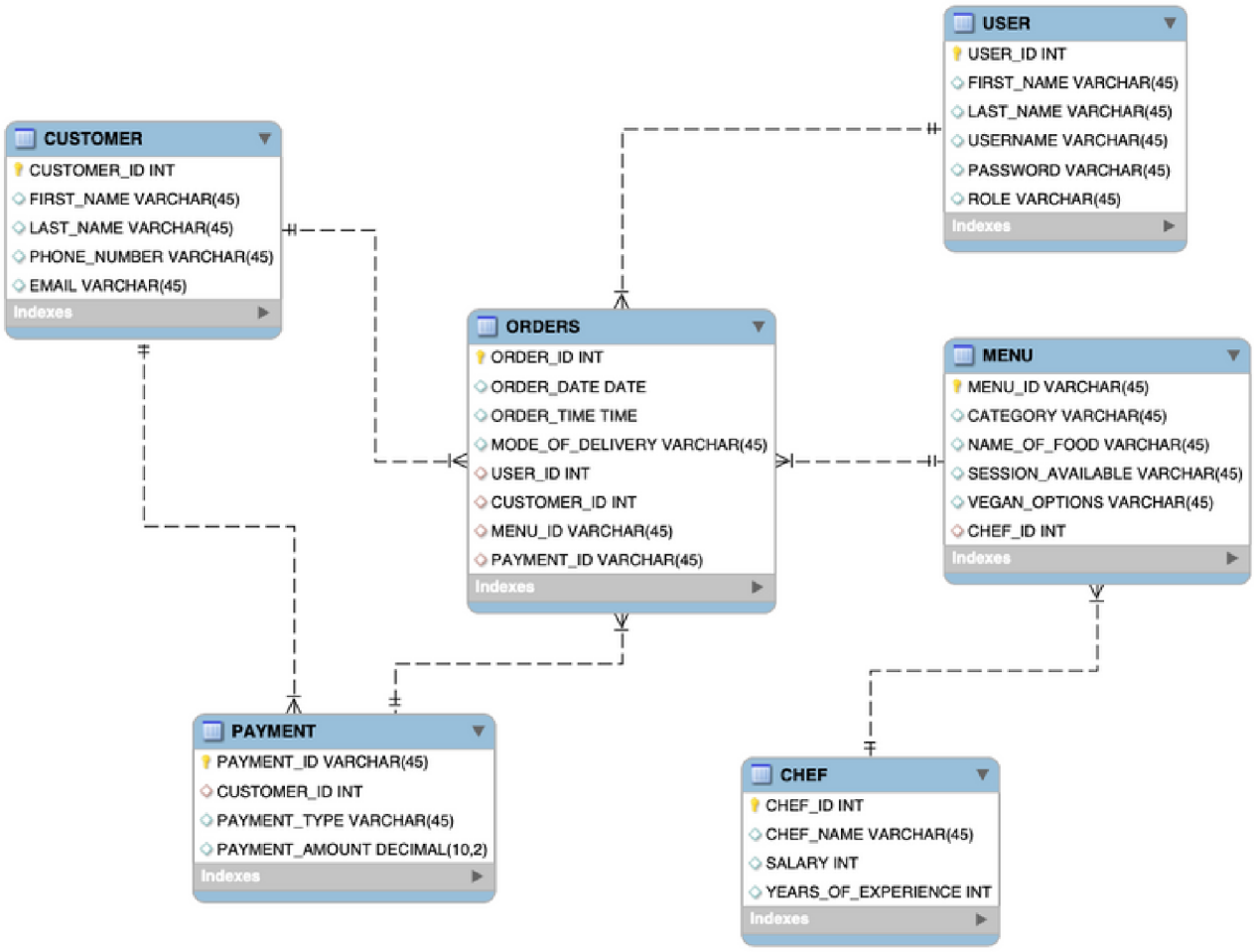

Diagram 1: ER Diagram of the Restaurant Ordering Database

Tables

ORDERS

The **ORDERS** table consists of the following information : Order ID, Order Date, Order Time, Mode of Deliver, User ID, Customer ID, Menu ID and Payment ID.

Input in MySQL :

```
CREATE TABLE ORDERS (
    ORDER_ID INT UNIQUE PRIMARY KEY,
    ORDER_DATE DATE,
    ORDER_TIME TIME,
    MODE_OF_DELIVERY VARCHAR(45),
    USER_ID INT,
    FOREIGN KEY (USER_ID) REFERENCES USER(USER_ID),
    CUSTOMER_ID INT,
    FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER(CUSTOMER_ID),
    MENU_ID VARCHAR(45),
    FOREIGN KEY (MENU_ID) REFERENCES MENU(MENU_ID),
    PAYMENT_ID VARCHAR(45),
    FOREIGN KEY (PAYMENT_ID) REFERENCES PAYMENT(PAYMENT_ID));
```

Output:

ORDER_ID	ORDER_DATE	ORDER_TIME	MODE_OF_DELIVERY	USER_ID	CUSTOMER_ID	MENU_ID	PAYMENT_ID
1001	2023-08-25	19:18:29	Self-pickup	9006	5	M01	1166
1002	2023-08-22	20:50:04	Deliveroo	9009	5	M02	1121
1003	2023-08-16	17:25:46	Uber Eats	9007	5	M03	1213
1004	2023-08-23	13:10:48	Uber Eats	9010	6	M04	1122
1005	2023-08-21	20:52:36	Just Eat	9006	6	M05	1211
1006	2023-08-19	14:07:20	Just Eat	9008	6	M04	1188
1007	2023-08-17	17:35:13	Self-pickup	9010	6	M02	1155
1008	2023-08-23	17:44:06	Deliveroo	9009	7	M06	1144
1009	2023-08-21	14:36:55	Deliveroo	9007	7	M05	1210
1010	2023-08-24	13:15:13	Uber Eats	9010	7	M03	1215
1011	2023-08-22	20:19:57	Deliveroo	9008	7	M02	1212
1012	2023-08-16	20:34:53	Just Eat	9009	7	M06	1177
1013	2023-08-15	19:46:53	Just Eat	9006	8	M05	1214
1014	2023-08-23	15:24:52	Self-pickup	9010	8	M03	1133
1015	2023-08-22	16:35:34	Uber Eats	9007	8	M01	1199

USER

The **USER** table consists of the following information : User ID, First Name, Last Name, Username, Password and Role,

Input in MySQL :

```
CREATE TABLE USER(
USER_ID INT PRIMARY KEY,
FIRST_NAME VARCHAR(45),
LAST_NAME VARCHAR(45),
USERNAME VARCHAR(45),
PASSWORD VARCHAR(45),
ROLE VARCHAR(45));
```

Output:

USER_ID	FIRST_NAME	LAST_NAME	USERNAME	PASSWORD	ROLE
9006	Horyew	Yim	yewyew	HYY990099!	CASHIER
9007	Lynn	Theng	sltheng	SLTH981221#	ADMIN
9008	Xinyi	Teoh	xy.teoh	XYT970924?	WAITRESS
9009	Zahim	Ismail	zahim.ismail	ZIZ989796%	WAITER
9010	Izzati	Mustaffa	izz.mustaffa	NIM8796^	CASHIER
NULL	NULL	NULL	NULL	NULL	NULL

CHEF

The **CHEF** table consists of the following information : Chef ID, Chef Name, Salary and Years of Experience.

Input in MySQL :

```
CREATE TABLE CHEF(
CHEF_ID INT PRIMARY KEY,
CHEF_NAME VARCHAR(45),
SALARY INT,
YEARS_OF_EXPERIENCE INT);
```

Output:

CHEF_ID	CHEF_NAME	SALARY	YEARS_OF_EXPERIENCE
201	TAMMY	2500	2
202	AMIRAH	3300	3
203	ADAM	6000	5
204	RACHEL	4500	4
205	RIZAL	2000	1
NULL	NULL	NULL	NULL

PAYMENT

The **PAYMENT** table consists of the following information : Payment ID, Customer ID, Payment Type and Payment Amount.

Input in MySQL :

```
CREATE TABLE PAYMENT
(PAYMENT_ID VARCHAR(45) PRIMARY KEY,
CUSTOMER_ID INT,
FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER(CUSTOMER_ID),
PAYMENT_TYPE VARCHAR(45),
PAYMENT_AMOUNT DECIMAL(10, 2));
```

Output:

PAYMENT_ID	CUSTOMER_ID	PAYMENT_TYPE	PAYMENT_AMOUNT
1121	5	Credit Card	30.13
1122	6	Online Banking	45.45
1133	8	E-Wallet	23.20
1144	7	Debit Card	15.16
1155	6	Debit Card	48.38
1166	5	Debit Card	27.01
1177	7	Debit Card	35.14
1188	6	E-Wallet	87.39
1199	8	Credit Card	15.75
1210	7	Online Banking	27.78
1211	6	E-Wallet	92.70
1212	7	Online Banking	24.25
1213	5	E-Wallet	140.70
1214	8	Debit Card	113.92
1215	7	Credit Card	23.24
NULL	NULL	NULL	NULL

MENU

The **MENU** table consists of the following information : Menu ID, Category, Name of Food, Session Available, Vegan Options and Chef ID.

Input in MySQL :

```
CREATE TABLE MENU(
MENU_ID VARCHAR(45) PRIMARY KEY,
CATEGORY VARCHAR(45),
NAME_OF_FOOD VARCHAR(45),
SESSION_AVAILABLE VARCHAR(45),
VEGAN_OPTIONS VARCHAR(45),
CHEF_ID INT,
FOREIGN KEY (CHEF_ID) REFERENCES CHEF(CHEF_ID));
```

Output:

MENU_ID	CATEGORY	NAME_OF_FOOD	SESSION_AVAILABLE	VEGAN_OPTIONS	CHEF_ID
M01	Main	Chicken Parmesan	BOTH	NO	201
M02	Dessert	Pana Cotta	DINNER	YES	204
M03	Main	Seafood Pasta	BOTH	NO	203
M04	Snacks	Arancini	LUNCH	YES	204
M05	Main	Pizza	BOTH	YES	202
M06	Dessert	Tiramisu	DINNER	YES	205
NULL	NULL	NULL	NULL	NULL	NULL

CUSTOMER

The **CUSTOMER** table consists of the following information : Chef ID, Chef Name, Salary and Years of Experience.

Input in MySQL :

```
CREATE TABLE CUSTOMER(
CUSTOMER_ID INT PRIMARY KEY,
FIRST_NAME VARCHAR(45),
LAST_NAME VARCHAR(45),
PHONE_NUMBER VARCHAR(45),
EMAIL VARCHAR(45));
```

Output:

CUSTOMER_ID	FIRST_NAME	LAST_NAME	PHONE_NUMBER	EMAIL
5	Yixin	Tan	07532219754	yixinnn@gmail.com
6	Phylicia	Ng	07327045458	psyn97@gmail.com
7	Meisin	Ong	07721683720	msong98@gmail.com
8	Yiwen	Phang	07613683261	yiwen00@gmail.com
NULL	NULL	NULL	NULL	NULL

View

Order details for main dish

The view below shows the detailed information for food which are categorised as 'Main Dish', which includes details of the Chef cooking the food and if Vegan options are available.

Input in MySQL :

```
CREATE VIEW Main_dish_order_details AS
SELECT
DISTINCT(O.MENU_ID), C.CHEF_ID, C.CHEF_NAME, C.YEARS_OF_EXPERIENCE,
M.NAME_OF_FOOD, M.CATEGORY, M.VEGAN_OPTIONS
FROM ORDERS O
JOIN MENU M ON O.MENU_ID = M.MENU_ID
JOIN CHEF C ON M.CHEF_ID = C.CHEF_ID
WHERE CATEGORY LIKE 'Main'
ORDER BY C.CHEF_ID;

SELECT * FROM Main_dish_order_details;
```

Output:

MENU_ID	CHEF_ID	CHEF_NAME	YEARS_OF_EXPERIENCE	NAME_OF_FOOD	CATEGORY	VEGAN_OPTIONS
M01	201	TAMMY	2	Chicken Parmesan	Main	NO
M05	202	AMIRAH	3	Pizza	Main	YES
M03	203	ADAM	5	Seafood Pasta	Main	NO

Stored Function

Membership Status

The Store Function was created to determine the membership status for the customers based on the total amount spent in aggregate (SUM of Payment Amount in the PAYMENT Table).

Input in MySQL :

```
USE RESTAURANT;

DELIMITER //

CREATE FUNCTION membership_status
(total_payment_amount DECIMAL(10, 2)) -- equals to((SUM(P.PAYMENT_AMOUNT) --
RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
    DECLARE customer_status VARCHAR(20);
    IF total_payment_amount > 200.00 THEN
        SET customer_status = 'Gold';
    ELSEIF total_payment_amount BETWEEN 150.00 AND 200.00 THEN
        SET customer_status = 'Silver';
    ELSE
        SET customer_status = 'Normal';
    END IF;

    RETURN customer_status;
END //
DELIMITER ;
```

```
SELECT C.CUSTOMER_ID, C.FIRST_NAME, C.LAST_NAME,
membership_status(SUM(P.PAYMENT_AMOUNT)) AS MEMBERSHIP_STATUS
FROM CUSTOMER C
INNER JOIN PAYMENT P ON P.CUSTOMER_ID = C.CUSTOMER_ID
GROUP BY C.CUSTOMER_ID;
```

Output:

CUSTOMER_ID	FIRST_NAME	LAST_NAME	MEMBERSHIP_STATUS
5	Yixin	Tan	Silver
6	Phylicia	Ng	Gold
7	Meisin	Ong	Normal
8	Yiwen	Phang	Silver

Query with subquery

Orders entered by 'Cashier'

The query below returns the details for the orders entered by users under the role 'Cashier'. The Order ID has been sorted in ascending order for easy reference.

Input in MySQL :

```
SELECT ORDER_ID, ORDER_DATE, ORDER_TIME
FROM ORDERS
WHERE USER_ID IN (SELECT USER_ID
                  FROM USER
                  WHERE ROLE = 'CASHIER')
ORDER BY ORDER_ID;
```

Output:

ORDER_ID	ORDER_DATE	ORDER_TIME
1001	2023-08-25	19:18:29
1004	2023-08-23	13:10:48
1005	2023-08-21	20:52:36
1007	2023-08-17	17:35:13
1010	2023-08-24	13:15:13
1013	2023-08-15	19:46:53
1014	2023-08-23	15:24:52
NULL	NULL	NULL

Stored Procedure

Rating & feedback for chefs

The stored procedure concats the chef's name, rating received and the comments received in a row, with commas separating each of them.

Input in MySQL :

```
DELIMITER //
CREATE PROCEDURE Rating (CHEF_NAME VARCHAR(50), CHEF_RATING INT,
COMMENTS VARCHAR(50))
BEGIN
   DECLARE FullRating VARCHAR(201);
   SET FullRating = CONCAT(CHEF_NAME, ',', CHEF_RATING,',', COMMENTS);
   SELECT FullRating;
END //
DELIMITER ;

CALL Rating('RIZAL',3,'Improvement neeeded.');
```

Output:

FullRating
RIZAL,3,Improvement neeeded.

Event

Departure time for order deliveries

As the delivery partners will pick-up and deliver the orders from the restaurant every 30 mins, the event function is used to record the time which the orders have been departed from the restaurant. The timestamp will be updated every 30 mins when the orders have departed from the restaurant.

Input in MySQL :

```
SET GLOBAL event_scheduler = ON;

CREATE TABLE DELIVERY_EVENT (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    DEPARTURE_TIME TIMESTAMP
);

DELIMITER //

CREATE EVENT RECURRING_EVENT
ON SCHEDULE EVERY 30 minute
STARTS NOW()
DO BEGIN
  INSERT INTO DELIVERY_EVENT (DEPARTURE_TIME)
  VALUES (NOW());
END //

DELIMITER ;

SELECT * FROM DELIVERY_EVENT;
```

Output:

ID	DEPARTURE_TIME
1	2023-08-29 01:08:56
2	2023-08-29 01:38:56

GROUP BY and HAVING

Quantity of non self-pickup orders

The query below shows the quantity of orders based on mode of delivery, which excludes the 'Self-pickup' option and has less than 5 orders in total. The output is being grouped by the mode of delivery.

Input in MySQL :

```
SELECT MODE_OF_DELIVERY, COUNT(ORDER_ID) AS NO_OF_ORDERS
FROM ORDERS
WHERE MODE_OF_DELIVERY NOT LIKE 'Self-pickup'
GROUP BY MODE_OF_DELIVERY
HAVING COUNT(ORDER_ID) <5;
```

Output:

MODE_OF_DELIVERY	NO_OF_ORDERS
Uber Eats	4
Just Eat	4

CODE FIRST GIRLS

Thank You !

Contact details:

haiilingg99@gmail.com

https://github.com/haiilingg

https://www.linkedin.com/in/tanhailing/