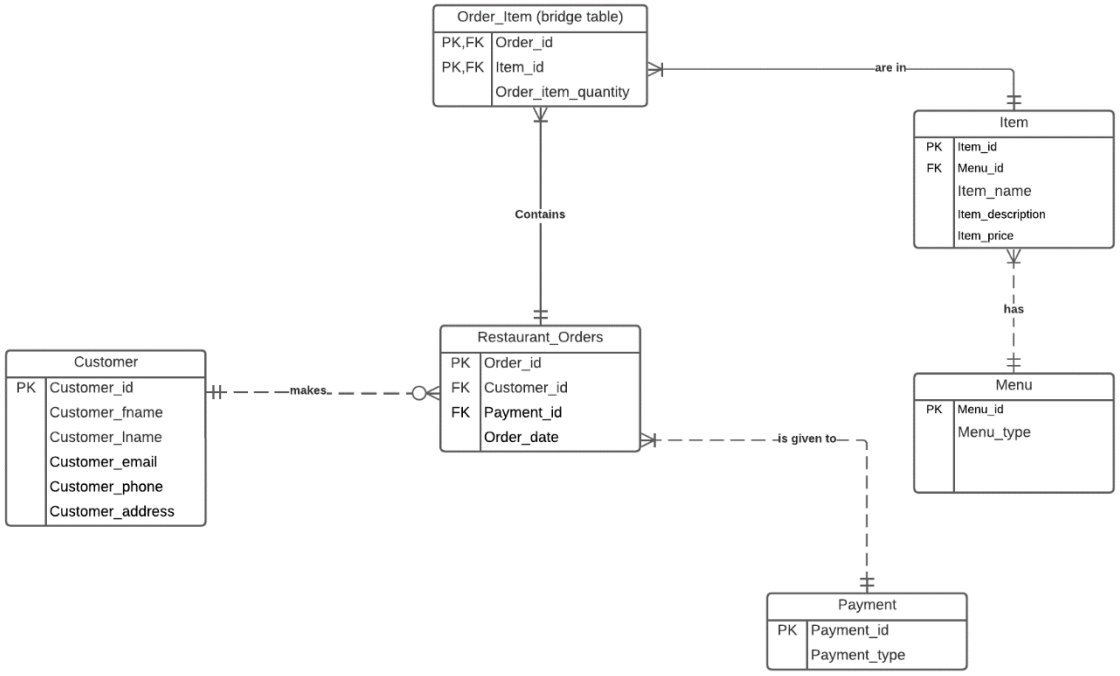
RESTAURANT MANAGEMENT SYSTEM

Table of Contents

Entity Relationship Diagram



Customer: Entity that contains information about customers who purchased at the restaurant

Payment: Entity that contains information about the method of payment

Restaurant_Orders: Entity that contains information about a customer order

Menu: Entity that contains information about the country of origin of the item

Item: Entity that contains information about the items/dishes made by the restaurant

Order_Item: bridge table to break down the many to many relationships between order table and item table

Create Tables

Customer

```
CREATE TABLE CUSTOMER (
        CUSTOMER_ID INTEGER PRIMARY KEY,
        CUSTOMER_FNAME VARCHAR(30) NOT NULL,
        CUSTOMER_LNAME VARCHAR(30) NOT NULL,
        CUSTOMER_EMAIL VARCHAR(30) NOT NULL,
        CUSTOMER_PHONE VARCHAR(20) NOT NULL,
        CUSTOMER_ADDRESS VARCHAR(30) NOT NULL
);
```

Payment

```
CREATE TABLE PAYMENT (
        PAYMENT_ID INTEGER PRIMARY KEY,
        PAYMENT_TYPE VARCHAR(10) NOT NULL
);
```

Restaurant_Orders

```
CREATE TABLE RESTAURANT_ORDERS (
        ORDER_ID SMALLINT PRIMARY KEY,
        CUSTOMER_ID INTEGER NOT NULL,
        PAYMENT_ID INTEGER NOT NULL,
        ORDER_DATE DATE NOT NULL,
        FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER (CUSTOMER_ID),
        FOREIGN KEY (PAYMENT_ID) REFERENCES PAYMENT (PAYMENT_ID)
);
```

Menu

```
CREATE TABLE MENU (
        MENU_ID SMALLINT PRIMARY KEY,
        MENU_TYPE VARCHAR (30) NOT NULL
);
```

Item

```
CREATE TABLE ITEM (
        ITEM_ID INTEGER PRIMARY KEY,
        MENU_ID INTEGER NOT NULL,
        ITEM_NAME VARCHAR(30) NOT NULL,
        ITEM_DESCRIPTION TEXT NOT NULL,
        ITEM_PRICE NUMERIC NOT NULL,
        FOREIGN KEY (MENU_ID) REFERENCES MENU (MENU_ID)
);
```

Order_item

```
CREATE TABLE ORDER_ITEM (
        ORDER_ID SMALLINT NOT NULL,
        ITEM_ID INTEGER NOT NULL,
        ORDER_ITEM_QUANTITY SMALLINT NOT NULL,
        FOREIGN KEY(ORDER_ID) REFERENCES RESTAURANT_ORDERS (ORDER_ID),
        FOREIGN KEY(ITEM_ID) REFERENCES ITEM (ITEM_ID),
        PRIMARY KEY(ORDER_ID,ITEM_ID)
);
```

Queries

CUSTOMERS

Find the number of customers at the restaurant

```
SELECT
	COUNT(DISTINCT(CUSTOMER_ID)) TOTAL_CUSTOMERS
FROM CUSTOMER;
```

	total_customers bigint
1	19

Find our 5 most profitable customers and how much they spent in total

```
SELECT
	CUSTOMER_ID,
	SUM(ORDER_TOTAL) AMOUNT_SPENT
FROM
	(SELECT
		ORDER_ID,
		SUM(ORDER_ITEM_QUANTITY * ITEM.ITEM_PRICE) AS ORDER_TOTAL
	FROM ORDER_ITEM
	LEFT JOIN ITEM ON ITEM.ITEM_ID = ORDER_ITEM.ITEM_ID
	GROUP BY ORDER_ID) AS SQ
LEFT JOIN RESTAURANT_ORDERS RO ON RO.ORDER_ID = SQ.ORDER_ID
GROUP BY CUSTOMER_ID
ORDER BY SUM(ORDER_TOTAL) DESC
LIMIT 5;
```

	customer_id integer	amount_spent numeric
1	2198	276.50
2	3583	252.00
3	1497	184.00
4	1374	169.00
5	1863	146.00

Find the customer(s) name that ordered the most and how many orders they made

```
SELECT
	CONCAT(CUSTOMER_FNAME,' ', CUSTOMER_LNAME) CUSTOMER,
	COUNT(ORDER_ID) TOTAL_ORDER_MADE
FROM RESTAURANT_ORDERS RO
LEFT JOIN CUSTOMER CU ON CU.CUSTOMER_ID = RO.CUSTOMER_ID
GROUP BY CU.CUSTOMER_ID
HAVING COUNT(ORDER_ID) = -- query returns 10
(SELECT
	MAX(ORDER_MADE)
FROM
	(SELECT
		CUSTOMER_ID,
		COUNT(ORDER_ID) ORDER_MADE
	FROM RESTAURANT_ORDERS
	GROUP BY CUSTOMER_ID) SQ);
```

	customer text	total_order_made bigint
1	Margareta Tacy	10
2	Kaleen Bryan	10
3	Meghann Placencia	10

ITEM

Create temporary table (to help answer the following questions)

```
DROP TABLE IF EXISTS TEMP_ITEM;
CREATE TEMPORARY TABLE TEMP_ITEM (
        ITEM_NAME VARCHAR(30) NOT NULL,
        ORDER_ITEM_QUANTITY SMALLINT NOT NULL,
        ITEM_PRICE NUMERIC NOT NULL
);

-- Insert data

INSERT INTO TEMP_ITEM
SELECT
        ITEM_NAME,
        ORDER_ITEM_QUANTITY,
        ITEM_PRICE
FROM ORDER_ITEM
LEFT JOIN ITEM ON ITEM.ITEM_ID = ORDER_ITEM.ITEM_ID;

--Preview temporary table

SELECT *
FROM TEMP_ITEM;
```

	item_name character varying (30)	order_item_quantity smallint	item_price numeric
13	Soba	2	10
14	Bibimbap	3	12.50
15	Sweet and Sour Chicken	1	7.50
16	Wonton Soup	1	6.50
17	Khmer Curry	1	11
18	Pad Thai	2	8.50
19	Bánh mì	1	7
20	Wonton Soup	1	6.50
21	Fish Amok	1	8
22	Onigiri	4	6
23	Bulgogi	3	10.50
24	Fish Amok	1	8
25	Onigiri	2	6
26	Bulgogi	2	10.50

Show the most to least popular item that was sold

```
SELECT
	ITEM_NAME,
	SUM(ORDER_ITEM_QUANTITY) ITEM_SOLD
FROM TEMP_ITEM
GROUP BY ITEM_NAME
ORDER BY ITEM _SOLD DESC;
```

	item_name character varying (30)	item_sold bigint
1	Wonton Soup	28
2	Bánh mì	27
3	Bibimbap	26
4	Fish Amok	24
5	Pad Thai	23
6	Onigiri	22
7	Khmer Curry	21
8	Bulgogi	20
9	Sweet and Sour Chicken	19
10	Soba	17

Find the revenue generated by each item

```
SELECT
	ITEM_NAME,
	SUM(ORDER_ITEM_QUANTITY * ITEM_PRICE) ITEM_TOTAL
FROM TEMP_ITEM
GROUP BY ITEM_NAME
ORDER BY ITEM_TOTAL DESC;
```

	item_name character varying (30)	item_total numeric
1	Bibimbap	325.00
2	Khmer Curry	231
3	Bulgogi	210.00
4	Pad Thai	195.50
5	Fish Amok	192
6	Bánh mì	189
7	Wonton Soup	182.00
8	Soba	170
9	Sweet and Sour Chicken	142.50
10	Onigiri	132

MENU

Create temporary table (to help answer the following questions)

```
DROP TABLE IF EXISTS TEMP_MENU;
CREATE TEMPORARY TABLE TEMP_MENU (
        MENU_TYPE VARCHAR (30) NOT NULL,
        ORDER_ITEM_QUANTITY SMALLINT NOT NULL,
        ITEM_PRICE NUMERIC NOT NULL
);

-- Insert data

INSERT INTO TEMP_MENU
SELECT
        MENU_TYPE,
        ORDER_ITEM_QUANTITY,
        ITEM_PRICE
FROM ORDER_ITEM
LEFT JOIN ITEM ON ITEM.ITEM_ID = ORDER_ITEM.ITEM_ID
LEFT JOIN MENU ON ITEM.MENU_ID = MENU.MENU_ID;

-- Alter temp table columns name

ALTER TABLE TEMP_MENU RENAME COLUMN ORDER_ITEM_QUANTITY TO QUANTITY;
ALTER TABLE TEMP_MENU RENAME COLUMN ITEM_PRICE TO REVENUE;

-- Preview temporary table

SELECT *
FROM TEMP_MENU;
```

	menu_type character varying (30)	quantity smallint	revenue numeric
33	Chinese	3	6.50
34	Japanese	1	10
35	Thai	2	7
36	Japanese	2	6
37	Korean	3	10.50
38	Cambodian	1	8
39	Japanese	1	6
40	Korean	1	12.50
41	Korean	2	12.50
42	Cambodian	1	8
43	Chinese	1	6.50
44	Japanese	1	10
45	Thai	4	7
46	Japanese	3	6
47	Korean	1	10.50

Create a function to find total quantity and revenue a menu generated

```
CREATE FUNCTION MENU_TYPE_DETAILS(MENU VARCHAR(30))
RETURNS SETOF TEMP_MENU AS
$$
SELECT
        MENU_TYPE,
        SUM(QUANTITY),
        SUM(QUANTITY * REVENUE)
FROM TEMP_MENU
WHERE MENU_TYPE ILIKE MENU -- ILIKE to ignore case sensitivity
GROUP BY MENU_TYPE
$$
LANGUAGE SQL;

-- Call the function for Korean Menu

SELECT *
FROM MENU_TYPE_DETAILS('Korean');
```

	menu_type character varying (30)	quantity smallint	revenue numeric
1	Korean	46	535.00

```
-- Call the function for Japanese Menu

SELECT *
FROM MENU_TYPE_DETAILS('Japanese');
```

	menu_type character varying (30)	quantity smallint	revenue numeric
1	Japanese	39	302

Convert total revenue a menu generated in % to total revenue made in the restaurant

```
WITH CTE_MENU AS (
        SELECT
                MENU_TYPE,
                SUM(ORDER_ITEM_QUANTITY * ITEM_PRICE) TOTAL_MENU
        FROM TEMP_MENU
        GROUP BY MENU_TYPE)
SELECT
        MENU_TYPE,
        ROUND(TOTAL_MENU / (SUM(TOTAL_MENU) OVER()) * 100, 2) AS TOTAL_MENU_PERCENT
FROM CTE_MENU
ORDER BY TOTAL_MENU_PERCENT DESC;
```

	menu_type character varying (30)	total_menu_percent numeric
1	Korean	27.17
2	Cambodian	21.48
3	Thai	19.53
4	Chinese	16.48
5	Japanese	15.34

MONTH

Find the total amount made during a month (most to least)

```
WITH CTE_ORDER AS (
	SELECT
		ORDER_ID,
		SUM(ORDER_ITEM_QUANTITY * ITEM_PRICE) TOTAL
	FROM ORDER_ITEM
	LEFT JOIN ITEM ON ITEM.ITEM_ID = ORDER_ITEM.ITEM_ID
	GROUP BY ORDER_ID)
SELECT
	TO_CHAR(ORDER_DATE, 'Month') MONTHS,
	SUM(TOTAL) TOTAL
FROM CTE_ORDER
LEFT JOIN RESTAURANT_ORDERS RO ON CTE_ORDER.ORDER_ID = RO.ORDER_ID
GROUP BY MONTHS
ORDER BY TOTAL DESC;
```

	months text	total numeric
1	October	376.50
2	August	354.50
3	November	328.50
4	July	324.00
5	June	320.50
6	September	265.00