SQL case-based assignment with a scenario and 10 queries for a food booking application like Zomato.

8. FOOD BOOKING APPLICATION

```
CREATE DATABASE FOOD;

USE FOOD;
```

RESTAURANTS TABLE

```
CREATE TABLE RESTAURANTS (
RESTAURANT_ID INT PRIMARY KEY,
NAME VARCHAR(100),
CITY VARCHAR(100),
CUISINE_TYPE ENUM ('INDIAN','CHINESE','ITALIAN','MEXICAN'),
RATING VARCHAR(20),
AVERAGE_COST_FOR_TWO DECIMAL(10,2) );
INSERT INTO RESTAURANTS (RESTAURANT_ID, NAME, CITY, CUISINE_TYPE, RATING,
AVERAGE_COST_FOR_TWO)
VALUES
  (1, 'Spice Garden', 'Delhi', 'INDIAN', '4.5', 800.00),
  (2, 'Dragon House', 'Mumbai', 'CHINESE', '4.2', 950.00),
  (3, 'Pasta Palace', 'Bangalore', 'ITALIAN', '4.7', 1200.00),
  (4, 'Taco Town', 'Hyderabad', 'MEXICAN', '4.1', 700.00),
  (5, 'Masala Magic', 'Chennai', 'INDIAN', '4.6', 850.00),
  (6, 'Beijing Express', 'Pune', 'CHINESE', '4.0', 780.00);
```

CREATE TABLE CUSTOMERS

```
CREATE TABLE CUSTOMERS (

CUSTOMER_ID VARCHAR (20) PRIMARY KEY,

FIRST_NAME VARCHAR(100),

LAST_NAME VARCHAR(100),

PHONE VARCHAR(20),

CITY VARCHAR(100),

JOIN_DATE DATE );


INSERT INTO CUSTOMERS (CUSTOMER_ID, FIRST_NAME, LAST_NAME, PHONE, CITY, JOIN_DATE)

VALUES

   ('CUST001', 'Amit', 'Sharma', '9876543210', 'Delhi', '2023-02-15'),

   ('CUST002', 'Neha', 'Verma', '9876543211', 'Mumbai', '2023-06-10'),

   ('CUST003', 'Rahul', 'Kapoor', '9876543212', 'Bangalore', '2024-01-20'),

   ('CUST004', 'Priya', 'Singh', '9876543213', 'Hyderabad', '2024-11-05'),

   ('CUST005', 'Karan', 'Mehta', '9876543214', 'Chennai', '2025-03-18');
```

ORDERS TABLE

```
CREATE TABLE ORDERS (

ORDER_ID VARCHAR(20) PRIMARY KEY,

CUSTOMER_ID VARCHAR(20),

RESTAURANT_ID INT,

ORDER_DATE DATETIME,

ORDER_AMOUNT DECIMAL(10,2),
ORDER_STATUS ENUM ('DELIVERED','PENDING','CANCELLED'),

FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMERS(CUSTOMER_ID),

FOREIGN KEY (RESTAURANT_ID) REFERENCES RESTAURANTS(RESTAURANT_ID) );


INSERT INTO ORDERS (ORDER_ID, CUSTOMER_ID, RESTAURANT_ID, ORDER_DATE, ORDER_AMOUNT,
ORDER_STATUS)

VALUES

  ('ORD001', 'CUST001', 1, '2025-05-01 12:30:00', 1500.00, 'DELIVERED'),

  ('ORD002', 'CUST002', 2, '2025-05-02 13:15:00', 950.00, 'PENDING'),

  ('ORD003', 'CUST003', 3, '2025-05-03 18:45:00', 1200.00, 'DELIVERED'),

  ('ORD004', 'CUST004', 4, '2025-05-04 20:00:00', 700.00, 'CANCELLED'),

  ('ORD005', 'CUST005', 5, '2025-05-05 11:00:00', 850.00, 'DELIVERED'),

  ('ORD006', 'CUST001', 6, '2025-05-06 14:30:00', 780.00, 'PENDING');
```

REVIEWS TABLE

```
CREATE TABLE REVIEWS (

REVIEW_ID INT PRIMARY KEY,

CUSTOMER_ID VARCHAR(20),

RESTAURANT_ID INT,

RATING VARCHAR(20),

COMMENT TEXT,

REVIEW_DATE DATETIME,

FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMERS(CUSTOMER_ID),

FOREIGN KEY (RESTAURANT_ID) REFERENCES RESTAURANTS(RESTAURANT_ID) );
```

```
INSERT INTO REVIEWS (REVIEW_ID, CUSTOMER_ID, RESTAURANT_ID, RATING, COMMENT, 
REVIEW_DATE)

VALUES

    (1, 'CUST001', 1, '4.5', 'Great Indian food, loved the ambiance!', '2025-05-10 13:00:00'),

    (2, 'CUST002', 2, '4.0', 'Good Chinese dishes but a bit pricey.', '2025-05-11 14:30:00'),

    (3, 'CUST003', 3, '4.7', 'Amazing Italian pasta and excellent service.', '2025-05-12 19:15:00'),

    (4, 'CUST004', 4, '3.8', 'Tacos were okay, but the place was noisy.', '2025-05-13 20:45:00'),

    (5, 'CUST005', 5, '4.6', 'Authentic Indian flavors and friendly staff.', '2025-05-14 12:00:00');
```

PAYMENTS TABLE

```
CREATE TABLE PAYMENTS (

PAYMENT_ID INT PRIMARY KEY,

ORDER_ID VARCHAR(20),

PAYMENT_METHOD ENUM('CARD','CASH','WALLET'),

AMOUNT DECIMAL(10,2),

PAYMENT_DATE DATETIME,

FOREIGN KEY (ORDER_ID) REFERENCES ORDERS(ORDER_ID) );
```

```
INSERT INTO PAYMENTS (PAYMENT_ID, ORDER_ID, PAYMENT_METHOD, AMOUNT, PAYMENT_DATE)

VALUES

    (1, 'ORD001', 'CARD', 1500.00, '2025-05-01 13:00:00'),

    (2, 'ORD002', 'CASH', 950.00, '2025-05-02 14:00:00'),

    (3, 'ORD003', 'WALLET', 1200.00, '2025-05-03 19:00:00'),

    (4, 'ORD004', 'CARD', 700.00, '2025-05-04 20:30:00'),

    (5, 'ORD005', 'CASH', 850.00, '2025-05-05 12:00:00'),
(6, 'ORD006', 'WALLET', 780.00, '2025-05-06 15:00:00');
```

Assignment Queries

1. Retrieve the names and locations of restaurants with a rating of 4.5 or higher.

```
SELECT NAME, CITY

FROM RESTAURANTS

WHERE CAST(RATING AS DECIMAL(3,1)) >= 4.5;
```

---

2. Find the total number of orders placed by each customer.

```
SELECT CUSTOMER_ID, COUNT(*) AS TOTAL_ORDERS

FROM ORDERS

GROUP BY CUSTOMER_ID;
```

3. List all restaurants offering "Italian" cuisine in "Mumbai".

```
SELECT NAME, CITY

FROM RESTAURANTS

WHERE CUISINE_TYPE = 'ITALIAN' AND CITY = 'Mumbai';
```

4. Calculate the total revenue generated by each restaurant from completed orders.

```
SELECT R.RESTAURANT_ID, R.NAME, SUM(O.ORDER_AMOUNT) AS TOTAL_REVENUE

FROM RESTAURANTS R

JOIN ORDERS O ON R.RESTAURANT_ID = O.RESTAURANT_ID

WHERE O.ORDER_STATUS = 'DELIVERED'

GROUP BY R.RESTAURANT_ID, R.NAME;
```

5. Retrieve the most recent order placed by each customer.

```
SELECT O1.*

FROM ORDERS O1

JOIN (
    SELECT CUSTOMER_ID, MAX(ORDER_DATE) AS MAX_DATE

    FROM ORDERS

    GROUP BY CUSTOMER_ID

) O2 ON O1.CUSTOMER_ID = O2.CUSTOMER_ID AND O1.ORDER_DATE = O2.MAX_DATE;
```

6. List customers who have not placed any orders yet.

```
SELECT C.CUSTOMER_ID, C.FIRST_NAME, C.LAST_NAME

FROM CUSTOMERS C

LEFT JOIN ORDERS O ON C.CUSTOMER_ID = O.CUSTOMER_ID

WHERE O.ORDER_ID IS NULL;
```

7. Identify the most reviewed restaurants (by number of reviews).

```
SELECT R.RESTAURANT_ID, R.NAME, COUNT(REV.REVIEW_ID) AS REVIEW_COUNT

FROM RESTAURANTS R

JOIN REVIEWS REV ON R.RESTAURANT_ID = REV.RESTAURANT_ID

GROUP BY R.RESTAURANT_ID, R.NAME

ORDER BY REVIEW_COUNT DESC

LIMIT 5;
```

8. Find the most preferred payment method.

```
SELECT PAYMENT_METHOD, COUNT(*) AS METHOD_COUNT

FROM PAYMENTS

GROUP BY PAYMENT_METHOD

ORDER BY METHOD_COUNT DESC

LIMIT 1;
```

9. List the top 5 restaurants by total revenue.

```
SELECT R.RESTAURANT_ID, R.NAME, SUM(O.ORDER_AMOUNT) AS TOTAL_REVENUE

FROM RESTAURANTS R

JOIN ORDERS O ON R.RESTAURANT_ID = O.RESTAURANT_ID
```

WHERE O.ORDER_STATUS = 'DELIVERED'

GROUP BY R.RESTAURANT_ID, R.NAME

ORDER BY TOTAL_REVENUE DESC

LIMIT 5;

10. Show the details of all cancelled orders along with the customer's and restaurant's names.

SELECT O.ORDER_ID, C.FIRST_NAME, C.LAST_NAME, R.NAME AS RESTAURANT_NAME, O.ORDER_DATE, O.ORDER_AMOUNT, O.ORDER_STATUS

FROM ORDERS O

JOIN CUSTOMERS C ON O.CUSTOMER_ID = C.CUSTOMER_ID

JOIN RESTAURANTS R ON O.RESTAURANT_ID = R.RESTAURANT_ID

WHERE O.ORDER_STATUS = 'CANCELLED';