Food Order Management System

DBMS Lab Project

A Project Report

Submitted by:-

Atin Arora-102103548
Dewang Goyal-102103552
Chirag Mohan Gupta-102103554
Himanshu Bansal-102103568

BE 2nd year, COE

Submitted to

Dr. Geeta Kasana

Thapar Institute of Engineering & Technology, Patiala
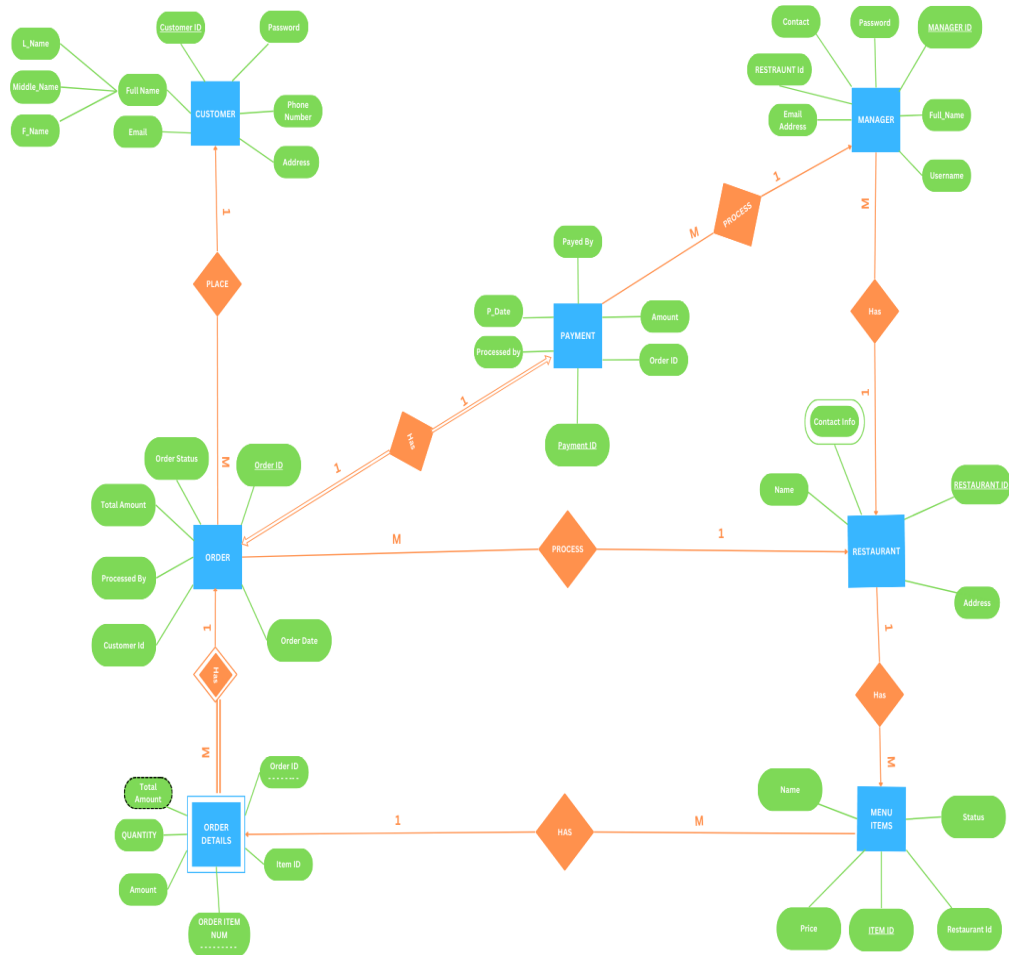
Index

Introduction: -

A food order management system is a software application that helps restaurants and food delivery services manage orders and streamline the order fulfillment process. The system typically includes components for order taking, order fulfillment, menu management, payment processing, and inventory management. The system allows restaurant staff to quickly and accurately enter and manage orders, track inventory levels, process payments, and analyze sales trends and customer behavior. By streamlining the order fulfillment process and providing real-time updates on inventory levels and sales trends, restaurants can improve their efficiency, reduce errors and delays, and provide better service to their customers. Overall, a food order management system is an essential tool for any restaurant or food delivery service looking to optimize their operations and increase their profits.
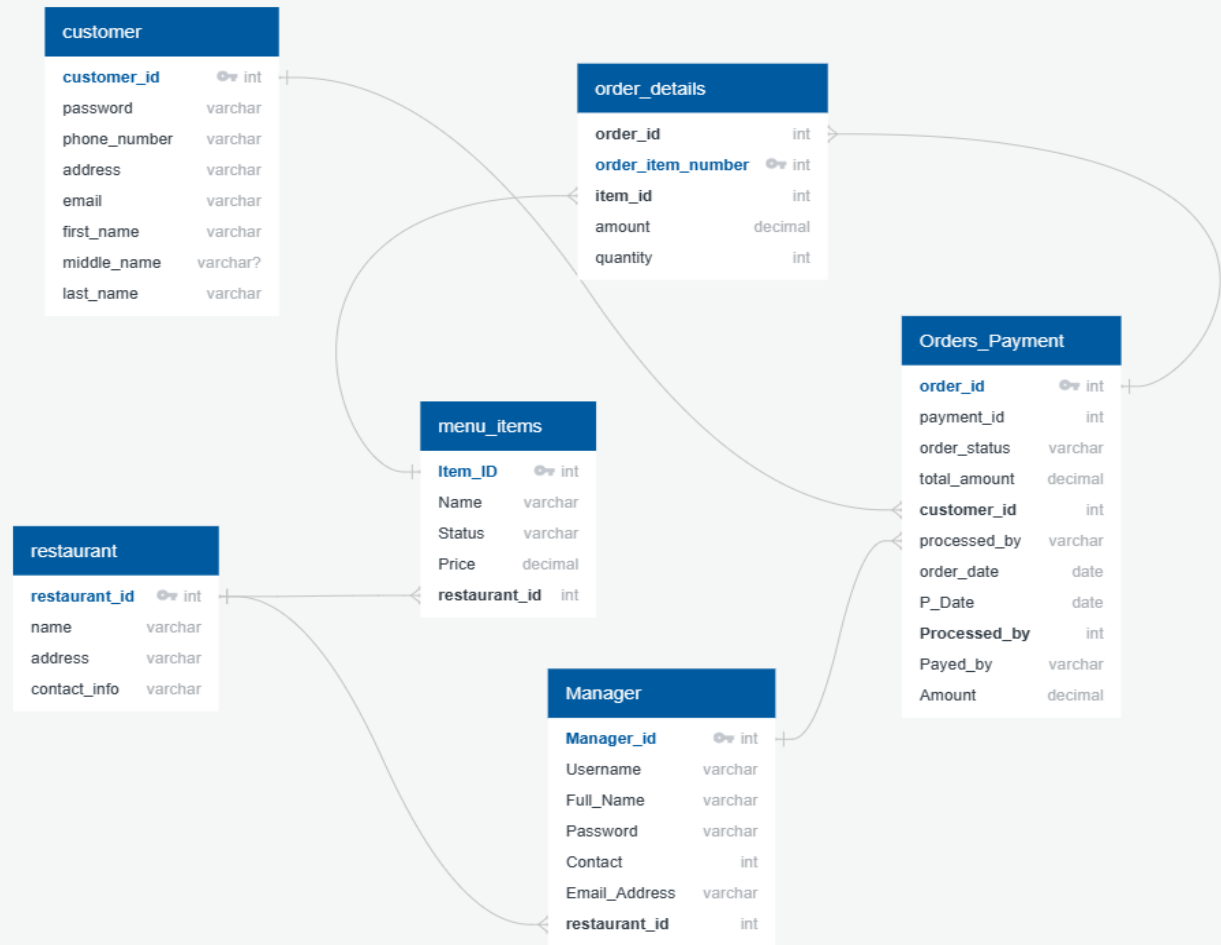
Requirements analysis: -

1. **Order Taking**: The order taking component allows restaurant staff to take orders from customers, either in person or over the phone. The system should be easy to use and should allow staff to quickly and accurately enter orders.
2. **Order Fulfilment**: The order fulfilment component helps restaurant staff manage the preparation and delivery of orders. The system should provide real-time updates on the status of orders and allow staff to easily manage multiple orders at once.
3. **Menu Management**: The menu management component allows restaurant owners or administrators to manage menus, add or remove items, and set prices. The system should be easy to use and should allow owners to make changes quickly and easily.
4. **Payment Processing**: The payment processing component allows staff to process payments for orders, either in person or online. The system should support multiple payment methods, including cash, credit cards, and online payment systems like PayPal.
5. **Inventory Management**: The inventory management component helps restaurant owners and staff manage inventory levels and track usage. The system should provide real-time updates on inventory levels and alert staff when inventory is running low.

ER Diagram:-

ER to Tables:-

customer

customer_id	int
password	varchar
phone_number	varchar
address	varchar
email	varchar
first_name	varchar
middle_name	varchar?
last_name	varchar

order_details

order_id	int
order_item_number	int
item_id	int
amount	decimal
quantity	int

menu_items

Item_ID	int
Name	varchar
Status	varchar
Price	decimal
restaurant_id	int

restaurant

restaurant_id	int
name	varchar
address	varchar
contact_info	varchar

Manager

Manager_id	int
Username	varchar
Full_Name	varchar
Password	varchar
Contact	int
Email_Address	varchar
restaurant_id	int

Orders_Payment

|	
order_id	int
payment_id	int
order_status	varchar
total_amount	decimal
customer_id	int
processed_by	varchar
order_date	date
P_Date	date
Processed_by	int
Payed_by	varchar
Amount	decimal

Normalization:-

Table: Customer

- 1st Normal Form: There is no multi-valued attribute in the table, so it is in1stNormal form.
- 2nd Normal Form: There is no partial dependency in the table as all the fields are dependent only on Customer_id. Hence, the table is in 2nd Normalform.
- 3rd Normal Form: Since there is no transitive dependency in the table (all fields are dependent only on the primary key), the table is in 3rd Normal Form.
- BCNF: Since there is no prime attribute which can be determined from a Prime or Non-prime attribute, the table is in BCNF.

Table: Order_Payment

- 1st Normal Form: There is no multi-valued attribute in the table, so it is in 1st Normal form.
- 2nd Normal Form: There is partial dependency in this table as primary key is (Order_id,Payment_id) all the attributes of payment entity can be determined by Payment_id .
- Hence, the table is in not in 2nd Normal form.
- BCNF: Since there is no prime attribute which can be determined from a Prime or Non-prime attribute, the table is in BCNF.

Before Normalization:

Order_Payment	
OrderID	🔑 int
Payment_ID	🔑 int
CustomerID	int
Order_date	Date
TotalAmount	money
OrderStatus	varchar
Processed_by	varchar
P_date	Date
Payed_by	varchar
Amount	decimal(10,2)
processed_by	varchar

After Normalization:
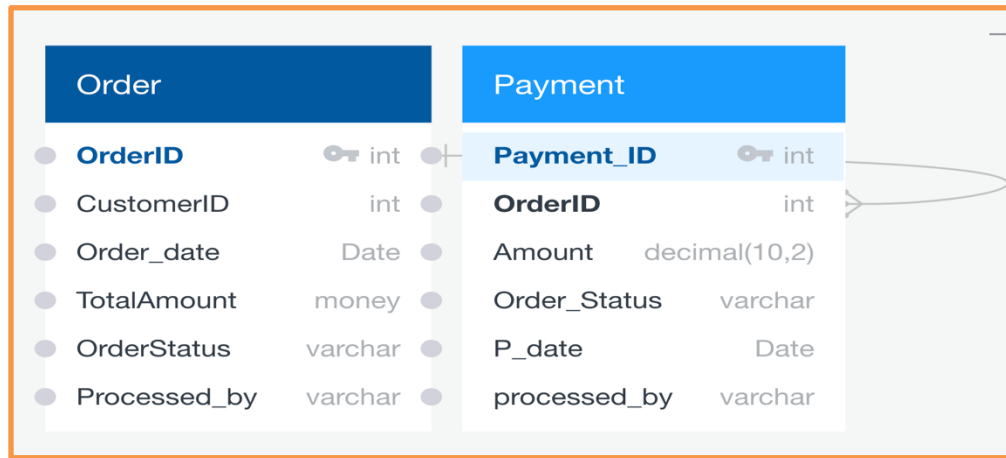
- 3rd Normal Form: Since there is no transitive dependency in the table (all fields are dependent only on the primary key), the table is in 3rd Normal Form.
- BCNF: Since there is no prime attribute which can be determined from a Prime or Non-prime attribute, the table is in BCNF.

Table: Menu_Items

- 1st Normal Form: There is no multi-valued attribute in the table, so it is in 1st Normal form.
- 2nd Normal Form: There is no partial dependency in the table as all the fields are dependent only on Customer id of customer. Hence, the table is in 2ndNormal form.
- 3rd Normal Form: Since there is no transitive dependency in the table (all fields are dependent only on the primary key), so the table is in 3rd Normal Form.
- BCNF: Since there is no prime attribute which can be determined from a Prime or Non-prime attribute, the table is in BCNF.
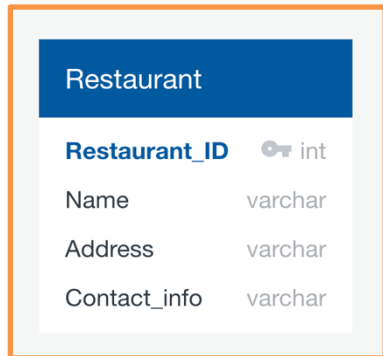
Table: Order_Details

- 1st Normal Form: There is no multi-valued attribute in the table, so it is in 1st Normal form.
- 2nd Normal Form: There is no partial dependency in the table as all the
- fields are dependent on the Contractor id. Hence, the table is in 2nd Normal form.
- 3rd Normal Form: Since there is no transitive dependency in the table (all fields are dependent only on the primary key), the table is in 3rd Normal Form.
- BCNF: Since there is no prime attribute which can be determined from a Prime or Non-prime attribute, the table is in BCNF.

Table: Manager

- 1st Normal Form: There is no multi-valued attribute in the table, so it is in 1st Normal form.
- 2nd Normal Form: There is no partial dependency in the table as all the
- fields are dependent on the Contractor id. Hence, the table is in 2nd Normal form.
- 3rd Normal Form: Since there is no transitive dependency in the table (all fields are dependent only on the primary key), the table is in 3rd Normal Form.
- BCNF: Since there is no prime attribute which can be determined from a Prime or Non-prime attribute, the table is in BCNF.
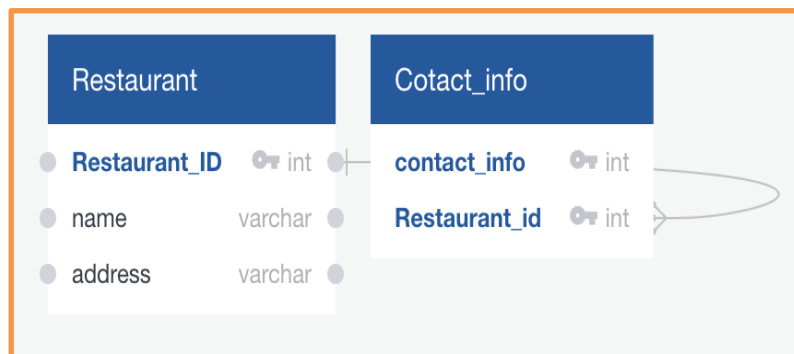
Table: Restaurant

- 1st Normal Form: There is a multi-valued attribute contact info in this table, so it is not in1st Normal form.

Before Normalization:

Restaurant

Restaurant	
Restaurant_ID 🔑	int
Name	varchar
Address	varchar
Contact_info	varchar

After Normalization:

Restaurant			Cotact_info		
Restaurant_ID 🔑	int		contact_info 🔑	int	
name	varchar		**Restaurant_id** 🔑	int	
address	varchar				

- 2nd Normal Form: There is no partial dependency in the table as all the fields are dependent on the Contractor id. Hence, the table is in 2nd Normal form.
- 3rd Normal Form: Since there is no transitive dependency in the table (all fields are dependent only on the primary key), the table is in 3rd Normal Form.
- BCNF: Since there is no prime attribute which can be determined from a Prime or Non-prime attribute, the table is in BCNF.

Code:-

Here is a github repository link to our code:
https://github.com/jimbo-exe/food-ordering

```
Statement 1
CREATE TABLE customer (
    customer_id INT PRIMARY KEY,
    password VARCHAR2(50) NOT NULL,
    phone_number VARCHAR2(20) NOT NULL,
    address VARCHAR2(100),
    email VARCHAR2(100) NOT NULL,
    first_name VARCHAR2(100) NOT NULL,
    Middle_name VARCHAR2(100),
    Last_name VARCHAR2(100)
)


Table created.
```

```
Statement 2
CREATE TABLE restaurant (
    restaurant_id INT PRIMARY KEY,
    name VARCHAR2(100),
    address VARCHAR2(100)
)


Table created.
```

```
Statement 3
CREATE TABLE contact_info (
    contact_info Varchar(14) PRIMARY KEY,
    restaurant_id INT REFERENCES restaurant(restaurant_id)
)


Table created.
```

Statement 4

```sql
CREATE TABLE Manager (
  Manager_id int PRIMARY KEY,
  Username varchar(50),
  Full_Name varchar(100),
  Password varchar(50),
  Contact int,
  Restaurant_id int,
  Email_Address varchar(100),
  FOREIGN KEY (Restaurant_id) REFERENCES Restaurant (Restaurant_id)
)
```

Table created.

Statement 5

```sql
CREATE TABLE menu_items (
  Item_ID int PRIMARY KEY,
  Name varchar(255),
  Status varchar(50) check(Status IN ('Available', 'Out of Stock')),
  Price decimal(10,2),
  Restaurant_ID int,
  FOREIGN KEY (restaurant_id) REFERENCES restaurant(restaurant_id)
)
```

Table created.

Statement 6

```sql
CREATE TABLE Orders(
    order_id INT PRIMARY KEY,
    payment_id INT,
    order_status VARCHAR(50),
    total_amount DECIMAL(10, 2),
    processed_by VARCHAR(50),
    customer_id INT,
    order_date DATE,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
  Check (order_status in ('Accepted')))
```

Table created.

```
CREATE TABLE Payment (

    payment_id int PRIMARY KEY,

    order_id int,

    P_Date date,

    Processed_by int,

    Payed_by varchar(100),

    Amount decimal(10, 2),

    FOREIGN KEY (order_id) REFERENCES Orders(order_id),

    FOREIGN KEY (Processed_by) REFERENCES Manager(Manager_id)

)


Table created.
```

```
INSERT INTO customer (customer_id, password, phone_number, address, email, first_name, middle_name, last_name)
VALUES
  (1, 'password1', '+91 9876543210', '123 Main St, mumbai, India', 'customer1@gmail.com', 'Shubham' , 'Kumar', 'Gupta');
INSERT INTO customer (customer_id, password, phone_number, address, email, first_name, middle_name, last_name)
VALUES
  (2, 'password2', '+91 9876543211', '456 High St, delhi, India', 'customer2@gmail.com', 'Jane', 'B.', 'Smith');
INSERT INTO customer (customer_id, password, phone_number, address, email, first_name, middle_name, last_name)
VALUES
  (3, 'password3', '+91 9876543212', '789 Ocean Blvd, delhi, India', 'customer3@gmail.com', 'Bob', NULL, 'Johnson');
INSERT INTO customer (customer_id, password, phone_number, address, email, first_name, middle_name, last_name)
VALUES
  (4, 'password4', '+91 9876543213', '987 Sakura St, bangalore, India', 'customer4@gmail.com', 'Sakura', NULL, 'Tanaka');
INSERT INTO customer (customer_id, password, phone_number, address, email, first_name, middle_name, last_name)
VALUES
  (5, 'password5', '+91 9876543214', '321 Rua da Praia, Kolkata India', 'customer5@gmail.com', 'Pedro', NULL, 'Souza');
```

```
INSERT INTO restaurant (restaurant_id, name, address)
VALUES
  (1, 'The Spice Room', '123 Main St, Mumbai, India');
INSERT INTO restaurant (restaurant_id, name, address)
VALUES
  (2, 'The Olive Garden', '456 High St, Delhi, India');
INSERT INTO restaurant (restaurant_id, name, address)
VALUES
  (3, 'La Petite France', '789 Ocean Blvd, Delhi, India');
INSERT INTO restaurant (restaurant_id, name, address)
VALUES
  (4, 'Sakura Japanese Restaurant', '987 Sakura St, Bangalore, India');
INSERT INTO restaurant (restaurant_id, name, address)
VALUES
  (5, 'Fogo de Chão', '321 Rua da Praia, Kolkata, India');
```

```sql
INSERT INTO contact_info (contact_info ,restaurant_id)
VALUES
   ( '+91 9876543210',1);
INSERT INTO contact_info (contact_info ,restaurant_id)
VALUES
   ( '+91 9876543211',1);
INSERT INTO contact_info (contact_info ,restaurant_id)
VALUES
   ('+91 9876543212',2);
INSERT INTO contact_info (contact_info ,restaurant_id)
VALUES
   ( '+91 9876543213',3);
INSERT INTO contact_info (contact_info ,restaurant_id)
VALUES
   ('+91 9876543214',3);
```

```sql
INSERT INTO Manager (Manager_id, Username, Full_Name, Password, Contact, Restaurant_id, Email_Address)
VALUES (1, 'john_doe', 'John Doe', 'password1', 1234567890, 1, 'john_doe@example.com');

INSERT INTO Manager (Manager_id, Username, Full_Name, Password, Contact, Restaurant_id, Email_Address)
VALUES (2, 'jane_doe', 'Jane Doe', 'password2', 2345678901, 2, 'jane_doe@example.com');

INSERT INTO Manager (Manager_id, Username, Full_Name, Password, Contact, Restaurant_id, Email_Address)
VALUES (3, 'bob_smith', 'Bob Smith', 'password3', 3456789012, 3, 'bob_smith@example.com');

INSERT INTO Manager (Manager_id, Username, Full_Name, Password, Contact, Restaurant_id, Email_Address)
VALUES (4, 'alice_jones', 'Alice Jones', 'password4', 4567890123, 4, 'alice_jones@example.com');

INSERT INTO Manager (Manager_id, Username, Full_Name, Password, Contact, Restaurant_id, Email_Address)
VALUES (5, 'mark_lee', 'Mark Lee', 'password5', 5678901234, 5, 'mark_lee@example.com');
```

```sql
INSERT INTO menu_items (Item_ID, Name, Status, Price, Restaurant_ID)
VALUES (101, 'Chicken Alfredo', 'Available', 15.99, 1);

INSERT INTO menu_items (Item_ID, Name, Status, Price, Restaurant_ID)
VALUES (102, 'Beef Burger', 'Available', 8.99, 2);

INSERT INTO menu_items (Item_ID, Name, Status, Price, Restaurant_ID)
VALUES (103, 'Veggie Pizza', 'Out of Stock', 12.99, 3);

INSERT INTO menu_items (Item_ID, Name, Status, Price, Restaurant_ID)
VALUES (104, 'Fish and Chips', 'Available', 10.99, 1);

INSERT INTO menu_items (Item_ID, Name, Status, Price, Restaurant_ID)
VALUES (105, 'Caesar Salad', 'Available', 7.99, 2);
```

```
SQL> CREATE TABLE order_details(
  2    order_id INT,
  3    item_id INT,
  4    order_item_number INT,
  5    amount DECIMAL(10,2),
  6    quantity INT,
  7    CONSTRAINT pk_order_details PRIMARY KEY (order_id, order_item_number),
  8    CONSTRAINT fk_order_details_order_id FOREIGN KEY (order_id) REFERENCES orders(order_id) ON DELETE CASCADE,
  9    CONSTRAINT fk_order_details_item_id FOREIGN KEY (item_id) REFERENCES menu_items(item_id) ON DELETE SET NULL);

Table created.

SQL> CREATE OR REPLACE TRIGGER trg_order_details_upd
  2    AFTER UPDATE OF item_id ON order_details
  3    FOR EACH ROW
  4  BEGIN
  5    UPDATE orders
  6    SET total_amount = (SELECT SUM(amount) FROM order_details WHERE order_id = :new.order_id)
  7    WHERE order_id = :new.order_id;
  8  END;
  9  /

Trigger created.

SQL> CREATE OR REPLACE TRIGGER trg_menu_items_upd
  2    AFTER UPDATE OF item_id ON menu_items
  3    FOR EACH ROW
  4  BEGIN
  5    UPDATE order_details
  6    SET amount = :new.price
  7    WHERE item_id = :new.item_id;
  8  END;
  9  /

Trigger created.
```

```
SQL>  insert into order_details values(1102,105,5,15.98,2);

1 row created.

SQL> select * from orders;

  ORDER_ID ORDER_STATUS                                          TOTAL_AMOUNT
---------- --------------------------------------------------- -------------
PROCESSED_BY CUSTOMER_ID ORDER_DAT
------------ ------------ ----------
      1102 Accepted                                                     81.9
           2            2 30-APR-23
```

```
SQL> CREATE OR REPLACE PROCEDURE INSERT_CUSTOMER(
  2      CUSTOMER_ID_P IN customer.customer_id%TYPE,
  3      PASSWORD_P IN customer.password%TYPE,
  4      PHONE_NUMBER_P IN customer.phone_number%TYPE,
  5      ADDRESS_P IN customer.address%TYPE,
  6      EMAIL_P IN customer.email%TYPE,
  7      FIRST_NAME_P IN customer.first_name%TYPE,
  8      MIDDLE_NAME_P IN customer.middle_name%TYPE,
  9      LAST_NAME_P IN customer.last_name%TYPE
 10  )
 11  AS
 12  BEGIN
 13      INSERT INTO CUSTOMER(
 14          customer_id, password, phone_number, address, email, first_name, middle_name, last_name
 15      ) VALUES (
 16          CUSTOMER_ID_P, PASSWORD_P, PHONE_NUMBER_P, ADDRESS_P, EMAIL_P, FIRST_NAME_P, MIDDLE_NAME_P, LAST_NAME_P
 17      );
 18      COMMIT;
 19  END;
 20  /

Procedure created.
```

```
  1   DECLARE
  2       customer_id_var customer.customer_id%TYPE := 1235;
  3       password_var customer.password%TYPE := 'password';
  4       phone_number_var customer.phone_number%TYPE := '5555551234';
  5       address_var customer.address%TYPE := '123 Main St';
  6       email_var customer.email%TYPE := 'test@example.com';
  7       first_name_var customer.first_name%TYPE := 'John';
  8       middle_name_var customer.middle_name%TYPE := 'Q';
  9       last_name_var customer.last_name%TYPE := 'Doe';
 10   BEGIN
 11       INSERT_CUSTOMER(
 12           CUSTOMER_ID_P => customer_id_var,
 13           PASSWORD_P => password_var,
 14           PHONE_NUMBER_P => phone_number_var,
 15           ADDRESS_P => address_var,
 16           EMAIL_P => email_var,
 17           FIRST_NAME_P => first_name_var,
 18           MIDDLE_NAME_P => middle_name_var,
 19           LAST_NAME_P => last_name_var
 20       );
 21*  END;
SQL> /

PL/SQL procedure successfully completed.
```

```
SQL> CREATE OR REPLACE PROCEDURE insert_restaurant(
  2    p_restaurant_id IN INT,
  3    p_name IN VARCHAR2,
  4    p_address IN VARCHAR2
  5  ) AS
  6  BEGIN
  7    INSERT INTO restaurant (restaurant_id, name, address)
  8    VALUES (p_restaurant_id, p_name, p_address);
  9    COMMIT;
 10    DBMS_OUTPUT.PUT_LINE('Restaurant inserted successfully.');
 11  EXCEPTION
 12    WHEN OTHERS THEN
 13      ROLLBACK;
 14      DBMS_OUTPUT.PUT_LINE('Error inserting restaurant: ' || SQLERRM);
 15  END;
 16  /

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE delete_restaurant(
  2    p_restaurant_id IN INT
  3  ) AS
  4  BEGIN
  5    DELETE FROM restaurant
  6    WHERE restaurant_id = p_restaurant_id;
  7    COMMIT;
  8    DBMS_OUTPUT.PUT_LINE('Restaurant deleted successfully.');
  9  EXCEPTION
 10    WHEN OTHERS THEN
 11      ROLLBACK;
 12      DBMS_OUTPUT.PUT_LINE('Error deleting restaurant: ' || SQLERRM);
 13  END;
 14  /

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE update_restaurant(
  2    p_restaurant_id IN INT,
  3    p_name IN VARCHAR2,
  4    p_address IN VARCHAR2
  5  ) AS
  6  BEGIN
  7    UPDATE restaurant
  8    SET name = p_name,
  9        address = p_address
 10    WHERE restaurant_id = p_restaurant_id;
 11    COMMIT;
 12    DBMS_OUTPUT.PUT_LINE('Restaurant updated successfully.');
 13  EXCEPTION
 14    WHEN OTHERS THEN
 15      ROLLBACK;
 16      DBMS_OUTPUT.PUT_LINE('Error updating restaurant: ' || SQLERRM);
 17  END;
 18  /
```

```
  1  DECLARE
  2    -- declare variables to store column values
  3    v_restaurant_id   restaurant.restaurant_id%TYPE;
  4    v_name            restaurant.name%TYPE;
  5    v_address         restaurant.address%TYPE;
  6    -- declare cursor
  7    CURSOR c_restaurant IS
  8      SELECT restaurant_id, name, address
  9      FROM restaurant;
 10  BEGIN
 11    -- loop through the cursor and fetch column values
 12    OPEN c_restaurant;
 13    LOOP
 14      FETCH c_restaurant INTO v_restaurant_id, v_name, v_address;
 15      EXIT WHEN c_restaurant%NOTFOUND;
 16      -- do something with the column values, for example print them out
 17      DBMS_OUTPUT.PUT_LINE('Restaurant ID: ' || v_restaurant_id);
 18      DBMS_OUTPUT.PUT_LINE('Name: ' || v_name);
 19      DBMS_OUTPUT.PUT_LINE('Address: ' || v_address);
 20      DBMS_OUTPUT.PUT_LINE('-------------------------');
 21    END LOOP;
 22    CLOSE c_restaurant;
 23* END;
SQL> /
Restaurant ID: 1
Name: Burger Joint
Address: 456 Elm St
-------------------------
Restaurant ID: 2
Name: The Olive Garden
Address: 456 High St, Delhi, India
-------------------------
Restaurant ID: 3
Name: La Petite France
Address: 789 Ocean Blvd, Delhi, India
-------------------------
Restaurant ID: 4
Name: Sakura Japanese Restaurant
Address: 987 Sakura St, Bangalore, India
-------------------------
Restaurant ID: 5
Name: Fogo de Chao
Address: 321 Rua da Praia, Kolkata, India
-------------------------
Restaurant ID: 6
Name: Pizza Place
Address: 123 Main St
-------------------------
```

```sql
SQL> CREATE OR REPLACE PROCEDURE insert_menu_item (
  2      p_item_id IN menu_items.item_id%TYPE,
  3      p_name IN menu_items.name%TYPE,
  4      p_status IN menu_items.status%TYPE,
  5      p_price IN menu_items.price%TYPE,
  6      p_restaurant_id IN menu_items.restaurant_id%TYPE
  7  ) AS
  8  BEGIN
  9    INSERT INTO menu_items (item_id, name, status, price, restaurant_id)
 10    VALUES (p_item_id, p_name, p_status, p_price, p_restaurant_id);
 11    COMMIT;
 12    DBMS_OUTPUT.PUT_LINE('Menu item inserted successfully');
 13  END;
 14  /

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE update_menu_item (
  2      p_item_id IN menu_items.item_id%TYPE,
  3      p_name IN menu_items.name%TYPE,
  4      p_status IN menu_items.status%TYPE,
  5      p_price IN menu_items.price%TYPE,
  6      p_restaurant_id IN menu_items.restaurant_id%TYPE
  7  ) AS
  8  BEGIN
  9    UPDATE menu_items
 10    SET name = p_name,
 11        status = p_status,
 12        price = p_price,
 13        restaurant_id = p_restaurant_id
 14    WHERE item_id = p_item_id;
 15    COMMIT;
 16    DBMS_OUTPUT.PUT_LINE('Menu item updated successfully');
 17  END;
 18  /

Procedure created.

SQL> -- Procedure to delete a menu item
SQL> CREATE OR REPLACE PROCEDURE delete_menu_item (
  2      p_item_id IN menu_items.item_id%TYPE
  3  ) AS
  4  BEGIN
  5    DELETE FROM menu_items WHERE item_id = p_item_id;
  6    COMMIT;
  7    DBMS_OUTPUT.PUT_LINE('Menu item deleted successfully');
  8  END;
  9  /

Procedure created.
```

```
SQL> BEGIN
  2  update_menu_item(108,'Chicken Spicy Burger' ,'Available', 12.99,1);
  3  END;
  4  /
Menu item updated successfully

PL/SQL procedure successfully completed.

SQL> BEGIN
  2  delete_menu_item(108);
  3  END;
  4  /
Menu item deleted successfully

PL/SQL procedure successfully completed.

SQL>
SQL> CREATE OR REPLACE PROCEDURE place_order(
  2    p_order_id IN orders.order_id%TYPE,
  3    p_customer_id IN orders.customer_id%TYPE,
  4    p_processed_by IN orders.processed_by%TYPE,
  5    p_order_status IN orders.order_status%TYPE,
  6    p_order_date IN orders.order_date%TYPE,
  7    p_menu_item_id IN menu_items.item_id%TYPE,
  8    p_quantity IN order_details.quantity%TYPE,
  9    p_payed_by IN payment.payed_by%TYPE,
 10    p_payment_processed_by IN payment.processed_by%TYPE
 11  )
 12  IS
 13    v_order_item_number order_details.order_item_number%TYPE;
 14    v_total_amount orders.total_amount%TYPE;
 15    v_amount menu_items.price%TYPE;
 16    v_payment_id payment.payment_id%TYPE;
 17  BEGIN
 18    -- Insert into Orders table
 19    INSERT INTO orders (order_id, order_status, total_amount, processed_by, customer_id, order_date)
 20    VALUES (p_order_id, p_order_status, 0, p_processed_by, p_customer_id, p_order_date);
 21    -- Insert into order_details table
 22    SELECT NVL(MAX(order_item_number), 0) + 1 INTO v_order_item_number FROM order_details WHERE order_id = p_order_id;
 23    SELECT price INTO v_amount FROM menu_items WHERE item_id = p_menu_item_id;
 24    INSERT INTO order_details (order_id, item_id, order_item_number, amount, quantity)
 25    VALUES (p_order_id, p_menu_item_id, v_order_item_number, v_amount*p_quantity, p_quantity);
 26    -- Update total amount in Orders table
 27    SELECT SUM(amount ) INTO v_total_amount FROM order_details WHERE order_id = p_order_id;
 28    UPDATE orders SET total_amount = v_total_amount WHERE order_id = p_order_id;
 29    -- Insert into Payment table
 30    SELECT NVL(MAX(payment_id),0) + 1 INTO v_payment_id FROM payment;
 31    INSERT INTO payment (payment_id, order_id, p_date, processed_by, payed_by, amount)
 32    VALUES (v_payment_id, p_order_id, SYSDATE, p_payment_processed_by, p_payed_by, v_total_amount);
 33  END;
 34  /

Procedure created.

SQL> begin
  2  place_order(1102,2,2,'Accepted',SYSDATE,102,2,'CASH',2);
  3  end;
  4  /

PL/SQL procedure successfully completed.
```

```
SQL>    CREATE OR REPLACE PROCEDURE insert_order_detail(
  2         p_order_id IN INT,
  3         p_item_id IN INT,
  4         p_quantity IN INT
  5      )
  6      IS
  7         v_order_item_num INT;
  8         v_amount NUMBER(5,2);
  9      BEGIN
 10        SELECT MAX(order_item_number) + 1
 11        INTO v_order_item_num
 12        FROM order_details
 13        WHERE order_id = p_order_id;
 14        IF v_order_item_num IS NULL THEN
 15            v_order_item_num := 1;
 16        END IF;
 17        SELECT PRICE INTO v_amount FROM menu_items WHERE item_id = p_item_id;
 18        INSERT INTO order_details (
 19            order_id,
 20            order_item_number,
 21            item_id,
 22            quantity,
 23            amount
 24        ) VALUES (
 25            p_order_id,
 26            v_order_item_num,
 27            p_item_id,
 28            p_quantity,
 29            v_amount*p_quantity
 30        );
 31    END;
 32  /

Procedure created.

SQL> Begin
  2  insert_order_detail(1102,102,2);
  3  End;
  4  /

PL/SQL procedure successfully completed.
```

Conclusion :-

In conclusion, a food order management system is an essential tool for any restaurant or food business. It provides an efficient and streamlined process for handling orders, reducing the chances of errors and delays. By automating the order-taking and processing tasks, the system allows staff to focus on providing excellent customer service and preparing delicious food.

Additionally, the system provides valuable insights into customer behaviour, allowing businesses to track order histories and preferences. This information can be used to create personalized promotions and loyalty programs that can help increase customer satisfaction and retention.

Overall, a food order management system is a must-have for any food business that wants to improve its operations, provide better service, and stay ahead of the competition. By investing in a reliable and user-friendly system, businesses can enhance their reputation, increase efficiency, and boost their bottom line.

References:-

1. https://www.inettutor.com/diagrams/online-food-ordering-system-er-diagram/
2. https://www.edrawmax.com/templates/1024779/
3. https://itsourcecode.com/uml/online-food-ordering-system-er-diagram/