Food Order Management System

DBMS Lab Project

A Project Report

Submitted by:-

Devansh Pratap-102203437
Parul Jain-102203438
Devansh Dhir-102203449
Amit Katoch-102203451

BE 2nd year, COE

Submitted to

Dr. Archana Singh

Thapar Institute of Engineering & Technology, Patiala

Index

1. Title Page
2. Introduction
    - Requirements analysis
3. ER- Diagram
4. ER to Table
5. Normalization
6. SQL & PL/SQL Code with output
    - Cursor
    - Stored Procedures
    - Functions & Exceptions
    - Triggers
7. Conclusion
8. References

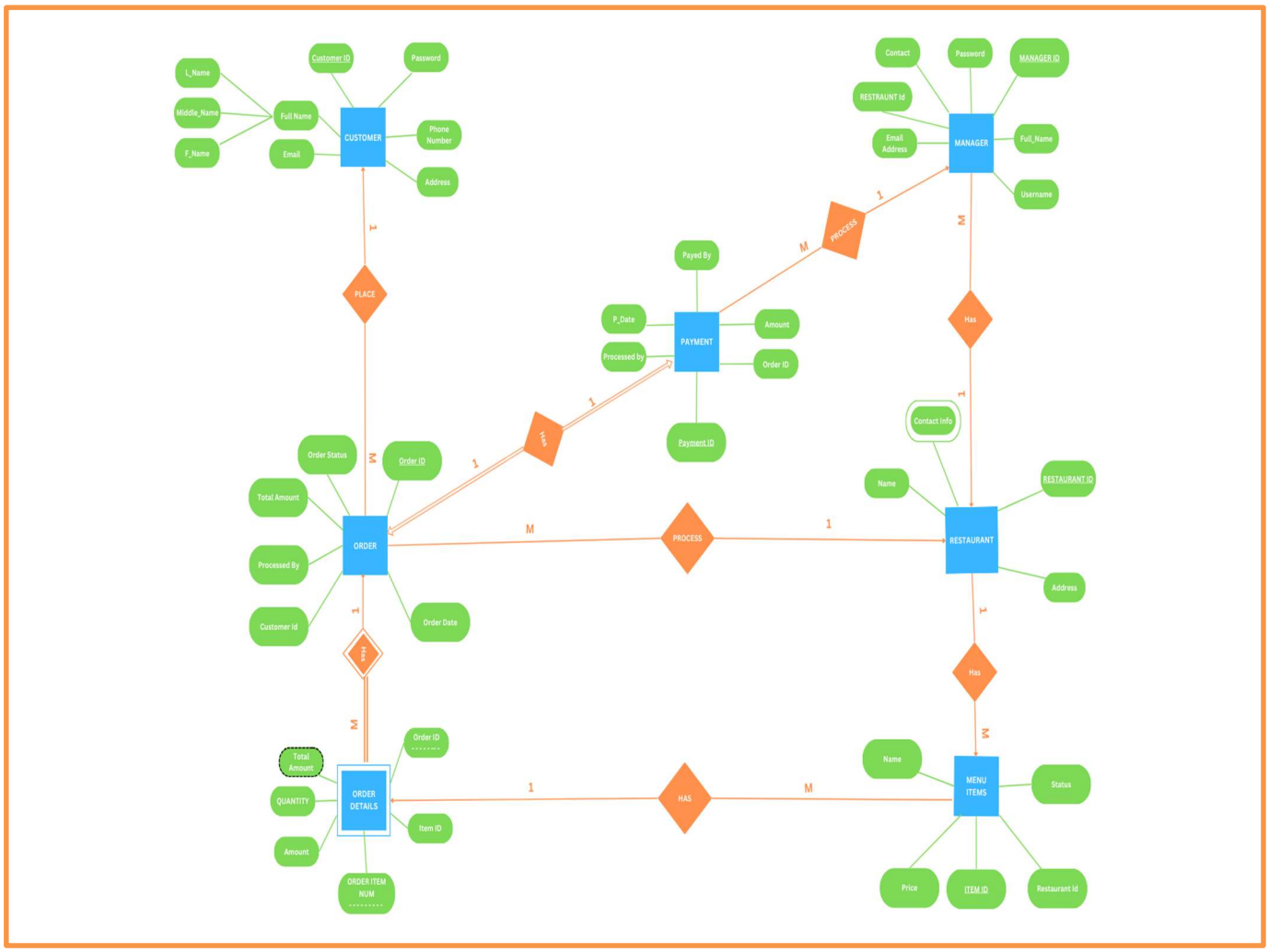
Introduction

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.

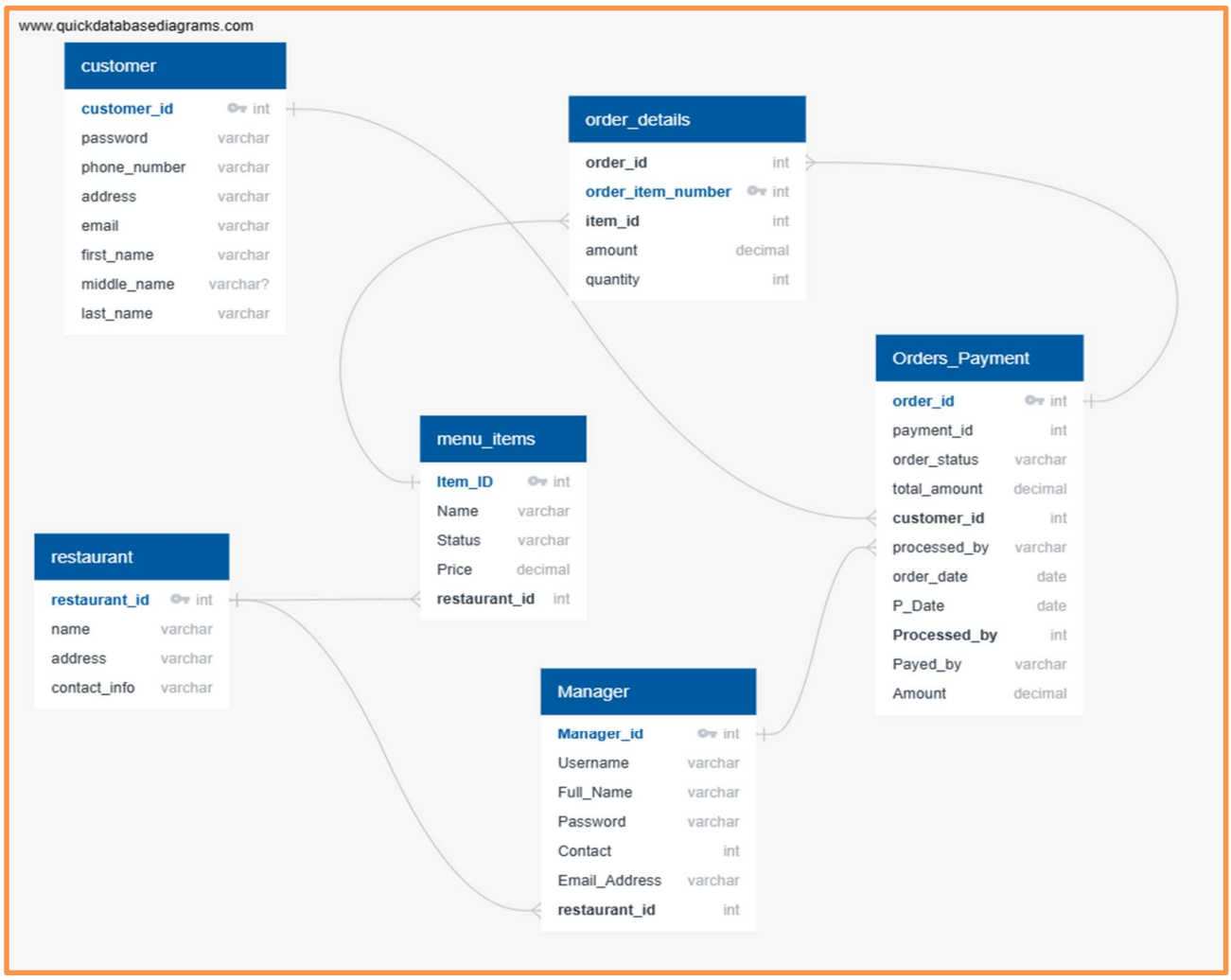
Requirements analysis

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


L_Name
Middle_Name
F_Name
Full Name
Email
Customer ID
Password
Phone Number
Address
CUSTOMER
PLACE
1
M
Contact
Password
MANAGER ID
RESTRAUNT Id
Email Address
MANAGER
Full_Name
Username
PROCESS
Payed By
P_Date
Processed by
PAYMENT
Amount
Order ID
Payment ID
Has
Contact Info
Order Status
Order ID
Total Amount
Processed By
Customer Id
Order Date
ORDER
Name
RESTAURANT ID
Address
RESTAURANT
PROCESS
HAS
Total Amount
QUANTITY
Amount
Order ID
Item ID
ORDER ITEM NUM
ORDER DETAILS
Name
Status
Price
ITEM ID
Restaurant Id
MENU ITEMS

ER to Tables


www.quickdatabasediagrams.com
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
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
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

Normalization

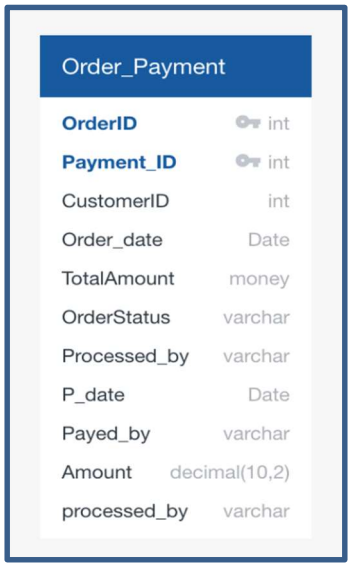
Table: Customer

- 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.

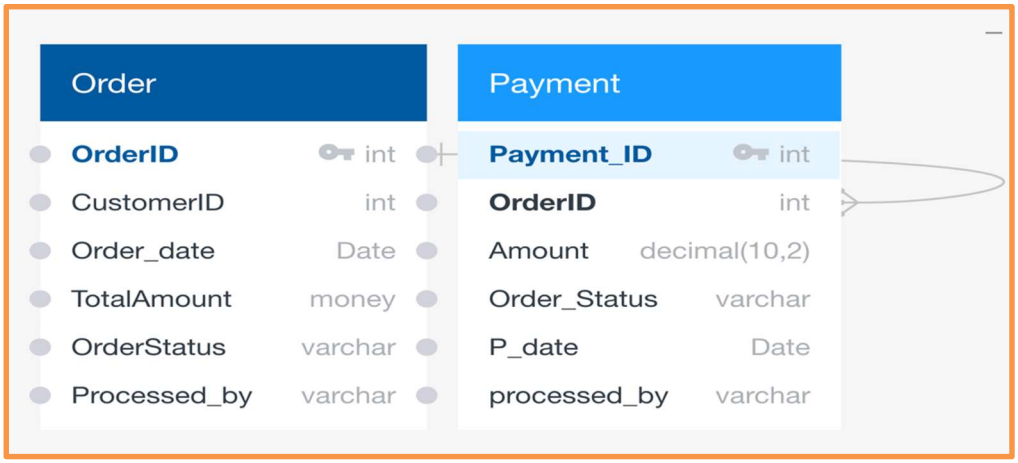
Table: Order_Payment

- 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.

Before Normalization:



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.

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.

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.

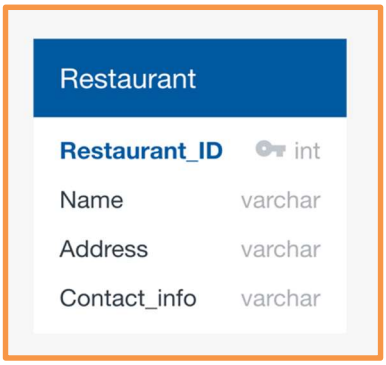
Table: Manager

- 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.

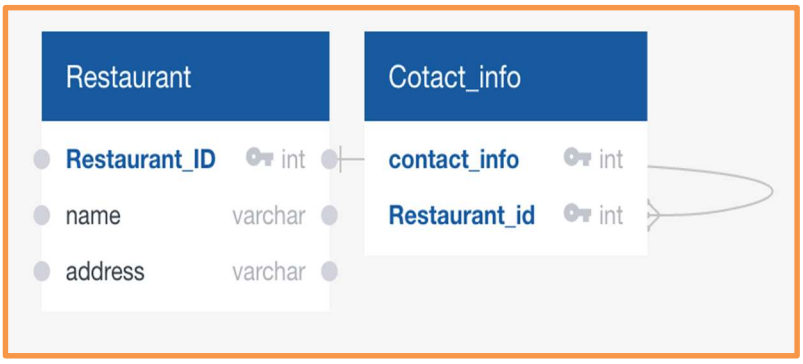
Table: Restaurant

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

Before Normalization:



After Normalization:



- 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.

Implementation of PL/SQL

Creation of Tables:

Statement 1

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

```
Table created.
```

Statement 2

```
CREATE TABLE restaurant (
  restaurant_id INT PRIMARY KEY,
  name VARCHAR (100),
    address VARCHAR(100)
)
```

```
Table created.
```

Statement 58

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

```
Table created.
```

Statement 74

```
CREATE TABLE order_details(
  order_id INT,
  item_id INT,
  order_item_number INT,
  amount DECIMAL(10,2),
  quantity INT,
  CONSTRAINT pk_order_details PRIMARY KEY (order_id, order_item_number),
  CONSTRAINT fk_order_details_order_id FOREIGN KEY (order_id) REFERENCES orders(order_id) ON DELETE CASCADE,
  CONSTRAINT fk_order_details_item_id FOREIGN KEY (item_id) REFERENCES menu_items(item_id) ON DELETE SET NULL)
```

```
Table created.
```

Statement 4

```
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

```
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

```
CREATE TABLE Orders(
     order_id INT PRIMARY KEY,
     order_status VARCHAR(50),
     total_amount DECIMAL(10, 2),
     processed_by int,
     customer_id INT,
     order_date DATE,
     CONSTRAINT fk_order_customer FOREIGN KEY (customer_id) REFERENCES Customer(customer_id) ON DELETE CASCADE,
     constraint fk_orders_restraunt foreign key (processed_by) references restaurant(restaurant_id) on delete set null,
     CHECK (order_status in ('Accepted'))
)
```

Table created.

Statement 7

```
CREATE TABLE Payment (
    payment_id int,
    order_id int,
    P_Date date,
    Processed_by int,
    Payed_by varchar(100),
    Amount decimal(10, 2),
    CONSTRAINT pk_payment PRIMARY KEY (payment_id),
    CONSTRAINT fk_payment_order FOREIGN KEY (order_id) REFERENCES Orders(order_id) ,
    CONSTRAINT fk_payment_manager FOREIGN KEY (Processed_by) REFERENCES Manager(Manager_id) ON DELETE SET NULL
)
```

Table created.

Insertion of Values in Tables:

```
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')
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

```

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');

```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

```sql
INSERT INTO contact_info (contact_info, restaurant_id) VALUES ('+91 9876543210', 1);
INSERT INTO contact_info (contact_info, restaurant_id) VALUES ('+91 9876543211', 2);
INSERT INTO contact_info (contact_info, restaurant_id) VALUES ('+91 9876543212', 3);
INSERT INTO contact_info (contact_info, restaurant_id) VALUES ('+91 9876543213', 4);
INSERT INTO contact_info (contact_info, restaurant_id) VALUES ('+91 9876543214', 5);
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

```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');
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

SQL Worksheet

Clear | Find | Actions | Save | Run

```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);
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

Cursors :

```
DECLARE
  p_restaurant_id INT := 1; -- Replace 123 with the desired restaurant ID
  CURSOR c_menu_items(p_restaurant_id IN INT) IS
  SELECT * FROM menu_items WHERE restaurant_id = p_restaurant_id;
  v_item_id menu_items.item_id%TYPE;
  v_name menu_items.name%TYPE;
  v_status menu_items.status%TYPE;
  v_price menu_items.price%TYPE;
  v_restaurant_id menu_items.restaurant_id%TYPE;
BEGIN
  OPEN c_menu_items(p_restaurant_id);
  LOOP
    FETCH c_menu_items INTO v_item_id, v_name, v_status, v_price, v_restaurant_id;
    EXIT WHEN c_menu_items%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE (v_item_id || ' ' || v_name || ' ' || v_status || ' ' || v_price || ' '||v_restaurant_id || CHR(10)); -- Use CHR(10) for line break
  END LOOP;
  CLOSE c_menu_items;
END;
/
```

```
Statement processed.
101 Chicken Alfredo Available 15.99 1

104 Fish and Chips Available 10.99 1
```

```
DECLARE
  -- declare variables to store column values
  v_restaurant_id   restaurant.restaurant_id%TYPE;
  v_name            restaurant.name%TYPE;
  v_address         restaurant.address%TYPE;
  -- declare cursor
  CURSOR c_restaurant IS
    SELECT restaurant_id, name, address
    FROM restaurant;
BEGIN
  -- loop through the cursor and fetch column values
  OPEN c_restaurant;
  LOOP
    FETCH c_restaurant INTO v_restaurant_id, v_name, v_address;
    EXIT WHEN c_restaurant%NOTFOUND;
    -- do something with the column values, for example print them out
    DBMS_OUTPUT.PUT_LINE('Restaurant ID: ' || v_restaurant_id);
    DBMS_OUTPUT.PUT_LINE('Name: ' || v_name);
    DBMS_OUTPUT.PUT_LINE('Address: ' || v_address);
    DBMS_OUTPUT.PUT_LINE('-------------------------');
  END LOOP;
  CLOSE c_restaurant;
END;
```

```
-------------------------
Restaurant ID: 1
Name: The Spice Room
Address: 123 Main St, Mumbai, India
-------------------------
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
```

```
DECLARE
p_manager_id INT:=1;
CURSOR manager_info(p_manager_id IN INT) IS
SELECT * FROM manager WHERE manager_id=p_manager_id;
v_manager_id manager.Manager_id%TYPE;
v_username manager.Username%TYPE;
v_full_name manager.Full_Name%TYPE;
v_password manager.Password%TYPE;
v_contact manager.Contact%TYPE;
v_restaurant_id manager.Restaurant_id%TYPE;
v_email manager.Email_Address%TYPE;

BEGIN
OPEN manager_info(p_manager_id);
LOOP
FETCH manager_info INTO v_manager_id,v_username,v_full_name,v_password,v_contact,v_restaurant_id,v_email;
EXIT WHEN manager_info%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(v_manager_id||' '||v_username||' '||v_full_name||' '||v_password||' '||v_contact||' '||v_restaurant_id||' '||v_email);
END LOOP;
CLOSE manager_info;
END;
```

```
Statement processed.
1 john_doe John Doe password1 1234567890 1 john_doe@example.com
```

Procedures :

```
CREATE OR REPLACE PROCEDURE INSERT_CUSTOMER(
    CUSTOMER_ID_P IN customer.customer_id%TYPE,
    PASSWORD_P IN customer.password%TYPE,
    PHONE_NUMBER_P IN customer.phone_number%TYPE,
    ADDRESS_P IN customer.address%TYPE,
    EMAIL_P IN customer.email%TYPE,
    FIRST_NAME_P IN customer.first_name%TYPE,
    MIDDLE_NAME_P IN customer.middle_name%TYPE,
    LAST_NAME_P IN customer.last_name%TYPE
)
AS
BEGIN
    INSERT INTO CUSTOMER(
        customer_id, password, phone_number, address, email, first_name, middle_name, last_name
    ) VALUES (
        CUSTOMER_ID_P, PASSWORD_P, PHONE_NUMBER_P, ADDRESS_P, EMAIL_P, FIRST_NAME_P, MIDDLE_NAME_P, LAST_NAME_P
    );
    COMMIT;
END;
```

```
Procedure created.
```

```
DECLARE
    customer_id_var customer.customer_id%TYPE := 1234;
    password_var customer.password%TYPE := 'password';
    phone_number_var customer.phone_number%TYPE := '5555551234';
    address_var customer.address%TYPE := '123 Main St';
    email_var customer.email%TYPE := 'test@example.com';
    first_name_var customer.first_name%TYPE := 'John';
    middle_name_var customer.middle_name%TYPE := 'Q';
    last_name_var customer.last_name%TYPE := 'Doe';
BEGIN
    INSERT_CUSTOMER(
        CUSTOMER_ID_P => customer_id_var,
        PASSWORD_P => password_var,
        PHONE_NUMBER_P => phone_number_var,
        ADDRESS_P => address_var,
        EMAIL_P => email_var,
        FIRST_NAME_P => first_name_var,
        MIDDLE_NAME_P => middle_name_var,
        LAST_NAME_P => last_name_var
    );
END;
```

```
Statement processed.
```

```
CREATE OR REPLACE PROCEDURE insert_restaurant(
  p_restaurant_id IN INT,
  p_name IN VARCHAR,
  p_address IN VARCHAR
) AS
BEGIN
  INSERT INTO restaurant (restaurant_id, name, address)
  VALUES (p_restaurant_id, p_name, p_address);
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Restaurant inserted successfully.');
EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
    DBMS_OUTPUT.PUT_LINE('Error inserting restaurant: ' || SQLERRM);
END;
```

```
Procedure created.
```

```
BEGIN
insert_restaurant(7,'BABA KA DHABA','DELHI');
END;
```

```
Statement processed.
Restaurant inserted successfully.
```

```
CREATE OR REPLACE PROCEDURE insert_menu_item (
  p_item_id IN menu_items.item_id%TYPE,
  p_name IN menu_items.name%TYPE,
  p_status IN menu_items.status%TYPE,
  p_price IN menu_items.price%TYPE,
  p_restaurant_id IN menu_items.restaurant_id%TYPE
) AS
BEGIN
  INSERT INTO menu_items (item_id, name, status, price, restaurant_id)
  VALUES (p_item_id, p_name, p_status, p_price, p_restaurant_id);
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Menu item inserted successfully');
END;
```

```
Procedure created.
```

```sql
CREATE OR REPLACE PROCEDURE update_menu_item (
  p_item_id IN menu_items.item_id%TYPE,
  p_name IN menu_items.name%TYPE,
  p_status IN menu_items.status%TYPE,
  p_price IN menu_items.price%TYPE,
  p_restaurant_id IN menu_items.restaurant_id%TYPE
) AS
BEGIN
  UPDATE menu_items
  SET name = p_name,
      status = p_status,
      price = p_price,
      restaurant_id = p_restaurant_id
  WHERE item_id = p_item_id;
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Menu item updated successfully');
END;
```

```
Procedure created.
```

```sql
CREATE OR REPLACE PROCEDURE delete_menu_item (
  p_item_id IN menu_items.item_id%TYPE
) AS
BEGIN
  DELETE FROM menu_items WHERE item_id = p_item_id;
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Menu item deleted successfully');
END;

```

```
Procedure created.
```

```sql
BEGIN
insert_menu_item(108, 'Burger', 'Available', 10.99,1);
END;
/
BEGIN
update_menu_item(108,'Chicken Spicy Burger' ,'Available', 12.99,1);
END;
/
BEGIN
delete_menu_item(108);
END;
/
```

```
Statement processed.
Menu item inserted successfully

Statement processed.
Menu item updated successfully

Statement processed.
Menu item deleted successfully
```

```
CREATE OR REPLACE PROCEDURE place_order(
   p_order_id IN orders.order_id%TYPE,
   p_customer_id IN orders.customer_id%TYPE,
   p_processed_by IN orders.processed_by%TYPE,
   p_order_status IN orders.order_status%TYPE,
   p_order_date IN orders.order_date%TYPE,
   p_menu_item_id IN menu_items.item_id%TYPE,
   p_quantity IN order_details.quantity%TYPE,
   p_payed_by IN payment.payed_by%TYPE,
   p_payment_processed_by IN payment.processed_by%TYPE
)
IS
   v_order_item_number order_details.order_item_number%TYPE;
   v_total_amount orders.total_amount%TYPE;
   v_amount menu_items.price%TYPE;
   v_payment_id payment.payment_id%TYPE;
BEGIN
   -- Insert into Orders table
   INSERT INTO orders (order_id, order_status, total_amount, processed_by, customer_id, order_date)
   VALUES (p_order_id, p_order_status, 0, p_processed_by, p_customer_id, p_order_date);
   -- Insert into order_details table
   SELECT NVL(MAX(order_item_number), 0) + 1 INTO v_order_item_number FROM order_details WHERE order_id = p_order_id;
   SELECT price INTO v_amount FROM menu_items WHERE item_id = p_menu_item_id;
   INSERT INTO order_details (order_id, item_id, order_item_number, amount, quantity)
   VALUES (p_order_id, p_menu_item_id, v_order_item_number, v_amount*p_quantity, p_quantity);
   -- Update total amount in Orders table
   SELECT SUM(amount ) INTO v_total_amount FROM order_details WHERE order_id = p_order_id;
   UPDATE orders SET total_amount = v_total_amount WHERE order_id = p_order_id;
   -- Insert into Payment table
   SELECT NVL(MAX(payment_id),0) + 1 INTO v_payment_id FROM payment;
   INSERT INTO payment (payment_id, order_id, p_date, processed_by, payed_by, amount)
   VALUES (v_payment_id, p_order_id, SYSDATE, p_payment_processed_by, p_payed_by, v_total_amount);
```

```
Procedure created.
```

```
begin
place_order(1102,2,2,'Accepted',SYSDATE,102,2,'CASH',2);
end;
```

```
Statement processed.
```

```sql
CREATE OR REPLACE PROCEDURE insert_order_detail(
        p_order_id IN INT,
        p_item_id IN INT,
        p_quantity IN INT
    )
    IS
        v_order_item_num INT;
        v_amount NUMBER(5,2);
    BEGIN
       SELECT MAX(order_item_number) + 1
       INTO v_order_item_num
       FROM order_details
       WHERE order_id = p_order_id;
       IF v_order_item_num IS NULL THEN
           v_order_item_num := 1;
       END IF;
       SELECT PRICE INTO v_amount FROM menu_items WHERE item_id = p_item_id;
       INSERT INTO order_details (
           order_id,
           order_item_number,
           item_id,
           quantity,
           amount
       ) VALUES (
           p_order_id,
           v_order_item_num,
           p_item_id,
           p_quantity,
           v_amount*p_quantity
       );
  END;

```

```
Procedure created.
```

```sql
Begin
insert_order_detail(1102,102,2);
End;
```

```
Statement processed.
```

Function with Exceptions :

```
CREATE OR REPLACE FUNCTION total_managers_of_restaurant(v_restaurant_id IN INT)
RETURN NUMBER IS
  total_managers NUMBER := 0;
BEGIN
  SELECT COUNT(*) INTO total_managers FROM Manager WHERE Restaurant_id = v_restaurant_id;

  RETURN total_managers;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN -1; -- Return a negative value to signify "NO DATA FOUND" exception
  WHEN TOO_MANY_ROWS THEN
    RETURN -2; -- Return another negative value to signify "TOO MANY ROWS" exception
END;
/
```

```
Function created.
```

```
DECLARE
  m_id INT;
  c NUMBER;
BEGIN
  m_id := 1;
  c := total_managers_of_restaurant(m_id);

  IF c = -1 THEN
    DBMS_OUTPUT.PUT_LINE('No managers found for the restaurant with ID ' || m_id);
  ELSIF c = -2 THEN
    DBMS_OUTPUT.PUT_LINE('Error: Too many managers found for the restaurant with ID ' || m_id);
  ELSE
    DBMS_OUTPUT.PUT_LINE('Total number of managers in restaurant with ID ' || m_id || ' is ' || c);
  END IF;
END;
/
```

```
Statement processed.
Total number of managers in restaurant with ID 1 is 1
```

Triggers:

```
CREATE OR REPLACE TRIGGER trg_order_details_upd
BEFORE INSERT OR UPDATE ON order_details
FOR EACH ROW
BEGIN
UPDATE orders
SET total_amount = (SELECT SUM(amount) FROM order_details WHERE order_id = :new.order_id)
WHERE order_id = :new.order_id;
END;
/
```

```
Trigger created.
```

```
CREATE OR REPLACE TRIGGER trg_menu_items_upd
  AFTER UPDATE OF item_id ON menu_items
  FOR EACH ROW
BEGIN
  UPDATE order_details
  SET amount = :new.price
  WHERE item_id = :new.item_id;
END;
```

```
Trigger created.
```

```
CREATE OR REPLACE TRIGGER update_payment_amount
AFTER UPDATE OF total_amount ON Orders
FOR EACH ROW
BEGIN
    UPDATE Payment SET Amount = :new.total_amount WHERE order_id = :new.order_id;
END;

```

```
Trigger created.
```

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. draw.io (diagrams.net)
2. Triggers in SQL (Hindi) l SQL Tutorial l Javatpoint (youtube.com)
3. Github.com