DBMS PROJECT

PROJECT NAME: Enquiry Transfer System

TEAM MEMBERS: NEELA SAI(H1)

REDDY SWARTIH REDDY(J5)

KUMMARI SHIVA CHARAN (LE16)

**Introduction**

In the rapidly evolving digital landscape, ecommerce platforms have revolutionized the way we shop, offering unmatched convenience and an extensive range of products. To ensure these platforms operate seamlessly, robust data management systems are essential. This project focuses on developing a comprehensive Database Management System (DBMS) for an e-commerce site, aimed at efficiently managing and integrating various aspects of the platform.

The core components of this DBMS include:

1. **Users**: Storing detailed personal data such as user IDs, names, phone numbers, email addresses, and physical addresses.
2. **Items**: Tracking all items available on the site, including item IDs, names, prices, stock quantities, and ratings.
3. **Orders**: Recording each order placed by users, with details like order IDs, user IDs, item IDs, order dates, quantities sold, and total amounts.
4. **Memberships**: Monitoring the memberships applied for by users, including membership IDs, user IDs, and membership expiry dates.
5. **Ratings**: Logging ratings given by users to staff members, including user IDs, staff IDs, rating scores, and timestamps.
6. **Staff**: Storing information about staff members, including staff IDs, names, departments, and salaries.

**Working of the DBMS**

1. **User Management**:
   * **Registration**: New users register on the platform by providing personal details. This information is stored in the user’s table.
   * **Authentication**: The system verifies user credentials during login to ensure secure access.
2. **Item Management**:
   * **Inventory Tracking**: The item table maintains records of all items, including their stock levels. Updates are made whenever new items are added or stock levels change.
   * **Price and Ratings**: Each item's price and user ratings are stored in the item table, providing essential details for users before making a purchase.
3. **Order Processing**:
   * **Order Placement**: Users place orders, which are recorded in the orders table with details such as order date, items purchased, quantity, and total amount.
   * **Order Fulfillment**: Inventory levels in the item table are updated based on the quantity sold.
4. **Membership Management**:
   * **Membership Application**: Users apply for memberships recorded in the membership table, tracking the validity of each membership.
   * **Expiry Tracking**: The system monitors membership expiry dates and notifies users when renewal is due.
5. **Rating and Feedback**:
   * **Rating Submission**: Users rate staff members based on their interactions. These ratings are stored in the rating table with timestamps.
   * **Quality Assurance**: Staff ratings help assess the performance and service quality provided by staff members.
6. **Staff Management**:
   * **Staff Information**: Details of staff members, including their department and salary, are stored in the staff table.
   * **Performance Tracking**: Staff ratings from users are linked to their records, enabling the system to evaluate their performance.

**DATABASE SCHEMA**

# User**:**

* **user\_id**: An integer that serves as the primary key and automatically increments with each new entry.
* **user\_name**: A variable character string to store the user's name.
* **phone\_no**: A variable character string for the phone number, which can include special characters (e.g., +, -, spaces).
* **g mail**: A variable character string for the email address (you might want to consider a more generic name like email).
* **address**: A variable character string for the user's address

# Item:

* **item\_id**: An integer that serves as the primary key and automatically increments with each new item.
* **item\_name**: A variable character string to store the item's name.
* **price**: A decimal type to represent the item's price, allowing for two decimal places.
* **stock**: An integer to track the available quantity of the item.
* **rating**: An integer for the item's rating, which can represent a score (e.g., 1 to 5).

# Order:

* **order\_id**: An integer that serves as the primary key and automatically increments with each new order.
* **user\_id**: An integer that acts as a foreign key referencing the user\_id in the users table.
* **bill\_date**: A date time field to record when the order was placed.
* **item\_id**: An integer that acts as a foreign key referencing the item\_id in the items table.
* **quantity\_sold**: An integer to track the number of items sold in the order.
* **total\_amount**: An integer representing the total amount for the order

# Staff:

* **staff\_id**: An integer that serves as the primary key and automatically increments with each new staff member.
* **staff\_name**: A variable character string to store the staff member's name.
* **department**: A variable character string to specify the department where the staff member works.
* **salary**: An integer to represent the staff member's salary.
* **rating**: An integer to indicate the staff member's performance rating.

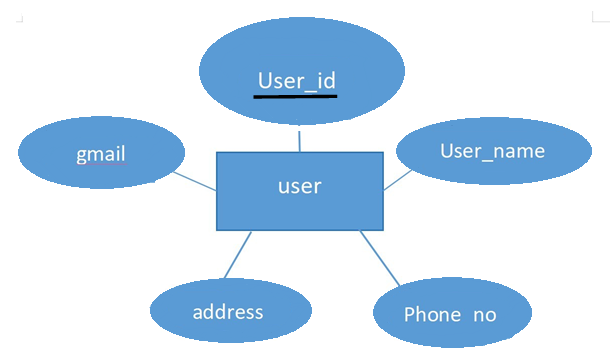
# Membership:

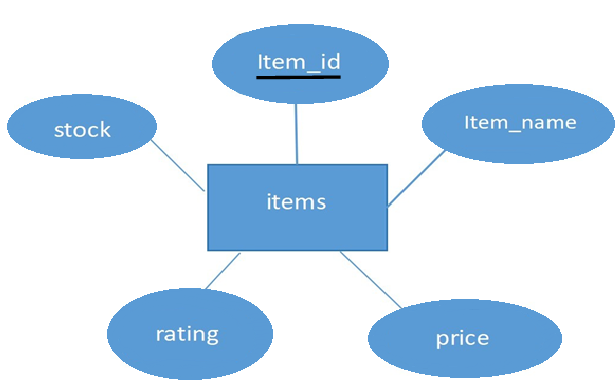
* **mem\_id**: An integer that serves as the primary key and automatically increments for each new membership.
* **user\_id**: An integer that acts as a foreign key referencing the user\_id in the users table.
* **valid\_till**: A date time field that defaults to one year from the current date and time, representing the membership's expiration date.

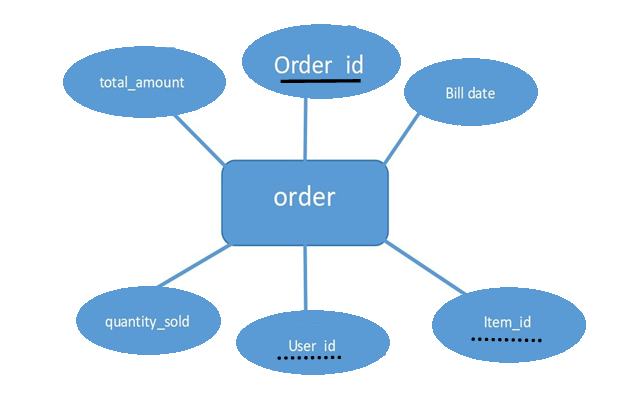
# Rating:

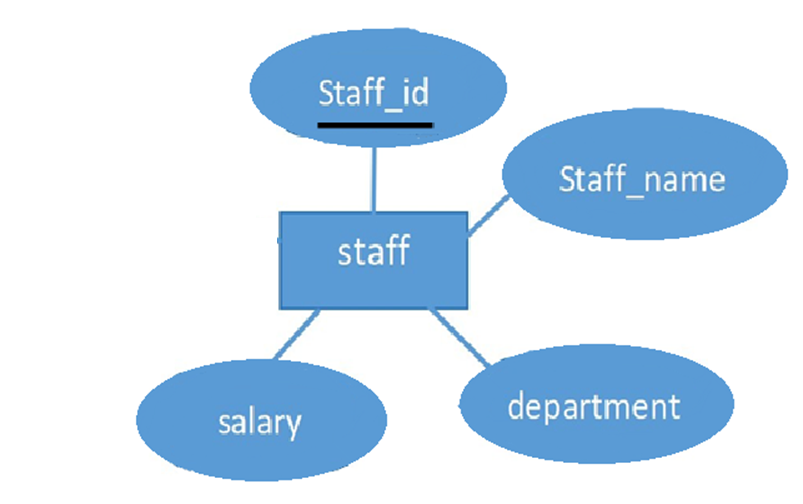
* **cus\_id**: An integer that acts as a foreign key referencing the user\_id in the users table.
* **to\_staff**: An integer that acts as a foreign key referencing the staff\_id in the staff table.
* **rating**: An integer representing the rating given by the customer.
* **timestamp**: A timestamp field that defaults to the current date and time, indicating when the feedback was given.

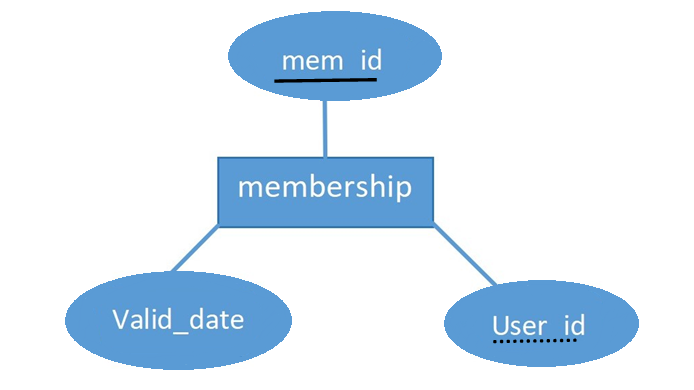
**ER-DIAGRAMS:**

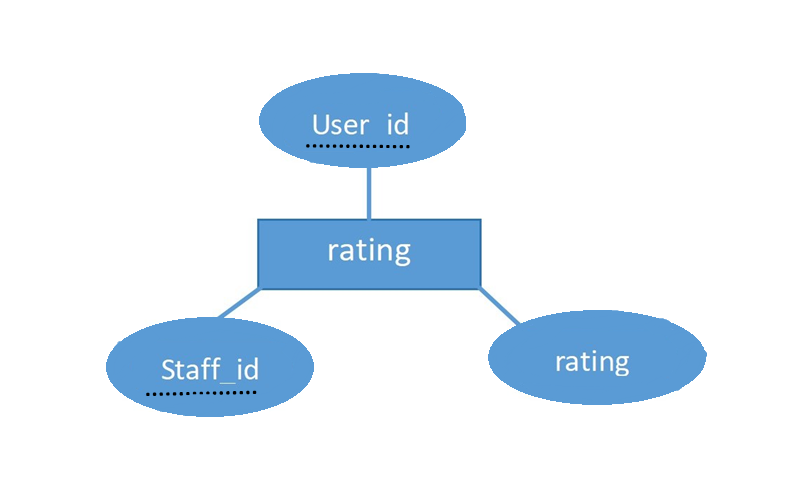
****

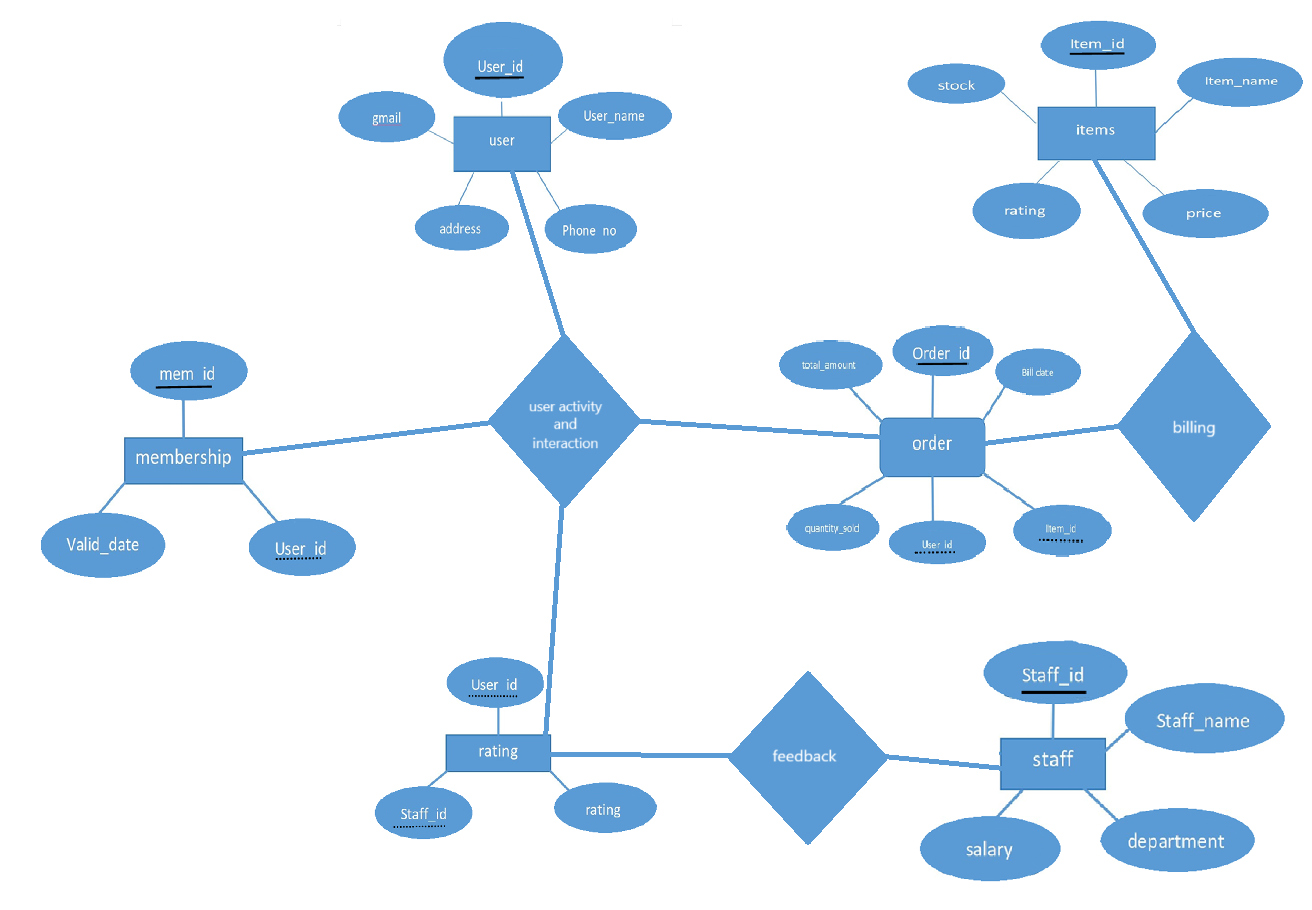
****



****



****



**E-COMMERCE ENQUIRY DATABASE**

**TABLES:**

1. USERS [user\_id, user\_name, phone\_no, g mail, address]
2. ITEMS [item\_id, item\_name, price, stock, rating]
3. ORDERS [order\_id, user\_id, bill\_date, item\_id, quantity\_sold, total\_amount]
4. STAFF [staff\_id, staff\_name,department, salary, rating]
5. MEMBERSHIP [mem\_id,user\_id, valid\_till]
6. RATING [cus\_id, to\_staff, rating, timestamp]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | user\_id | | user\_name | phone\_no | g mail | address |  |
|  | Int primary key, auto  increment | | var-char | var char | var char | var char |
| item\_id | | item\_name | | price | stock | Rating |
| Int Primary key,  auto-increment | | var char | | int | int | Int |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| order\_id, | user\_id, | bill\_date, | item\_id, | quantity\_sold, | Total\_amount |
| Int primary  key auto- increment | Int foreign key | Date-time | Int foreign key | int | Int |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [staff\_id, | staff\_name, | department, | salary, | Rating |
| Int primary key  auto increment | var char | var char | int | int |

|  |  |  |
| --- | --- | --- |
| mem\_id, | user\_id, | valid\_till |
| Int auto-increment | Int foreign key | Date-time(to be set using  function) |

|  |  |  |  |
| --- | --- | --- | --- |
| cus\_id, | to\_staff, | rating, | timestamp |
| Int foreign key | Int foriegn key | int | timestamp |

**SQL QUERIES**

1. Display the items ordered by the user along with total amount and address to deliver where user id=5;
2. Display the ratings given by the user to the staff;
3. Display the staff details whose rating is greater than 4;
4. Display the items which are available to the users along with cost;
5. Display the items which are deliver on date 20-05-2024;
6. Display the items whose rating is greater than 4;
7. Display the staff names along with their ids whose salary is greater than 50000;
8. Display the items id and item names whose letter starts with letter ‘s’;
9. Create a view which contains the all staff details along with rating ;
10. Create a view of the items which are available in the stock along with price and quantity;

**DATABASE:**

create database dbms;

use dbms;

CREATE TABLE users (

user\_id INT PRIMARY KEY, user\_name VARCHAR(100), phno INT UNIQUE,

gmail VARCHAR(100),

address VARCHAR(255)

);

CREATE TABLE item (

item\_id INT PRIMARY KEY,

item\_name VARCHAR(100) UNIQUE, price INT,

stockINT, ratingINT

);

CREATE TABLE orders ( order\_id INT PRIMARY KEY, user\_id INT,

item\_id INT, bill\_date DATE, qty\_sold INT, tot\_amt INT,

FOREIGN KEY (user\_id) REFERENCES users(user\_id), FOREIGN KEY (item\_id) REFERENCES item(item\_id)

);

CREATE TABLE membership ( m\_id INT PRIMARY KEY,

user\_id INT, valid\_till DATE,

FOREIGN KEY (user\_id) REFERENCES users(user\_id)

);

CREATE TABLE rating( user\_id INT,

staff\_id INT,

rating INT, timeDATE,

PRIMARY KEY (user\_id, staff\_id),

FOREIGN KEY (user\_id) REFERENCES users(user\_id), FOREIGN KEY (staff\_id) REFERENCES staff(staff\_id)

);

CREATE TABLE staff (

staff\_id INT PRIMARY KEY, staff\_name VARCHAR(100), dept VARCHAR(100),

sal INT

);

show tables;

INSERT INTO users (user\_id, user\_name, phno, gmail, address) VALUES (1, 'John Doe', 1234567890, 'john@example.com', '123 Main St'),

(2, 'Jane Smith', 1234567891, 'jane@example.com', '456 Maple St'),

(3, 'Alice Johnson', 1234567892, 'alice@example.com', '789 Oak St'),

(4, 'Bob Brown', 1234567893, 'bob@example.com', '101 Pine St'),

(5, 'Carol White', 1234567894, 'carol@example.com', '202 Birch St'),

(6, 'David Green', 1234567895, 'david@example.com', '303 Cedar St'),

(7, 'Eve Black', 1234567896, 'eve@example.com', '404 Elm St'),

(8, 'Frank Wilson', 1234567897, 'frank@example.com', '505 Spruce St'),

(9, 'Grace Taylor', 1234567898, 'grace@example.com', '606 Willow St'),

(10, 'Hank Martinez', 1234567899, 'hank@example.com', '707 Fir St'),

(11, 'Ivy Lewis', 1234567880, 'ivy@example.com', '808 Cypress St'),

(12, 'Jack Lee', 1234567881, 'jack@example.com', '909 Redwood St'),

(13, 'Kara Clark', 1234567882, 'kara@example.com', '1010 Sequoia St'),

(14, 'Liam Walker', 1234567883, 'liam@example.com', '1111 Magnolia St'),

(15, 'Mia Hill', 1234567884, 'mia@example.com', '1212 Palm St'),

(16, 'Noah Scott', 1234567885, 'noah@example.com', '1313 Bay St'),

(17, 'Olivia Adams', 1234567886, 'olivia@example.com', '1414 Hickory St'),

(18, 'Paul Evans', 1234567887, 'paul@example.com', '1515 Poplar St'),

(19, 'Quincy Harris', 1234567888, 'quincy@example.com', '1616 Cottonwood St'),

(20, 'Rachel Carter', 1234567889, 'rachel@example.com', '1717 Sycamore St');

INSERT INTO item (item\_id, item\_name, price, stock, rating) VALUES (1, 'Laptop', 1000, 50, 4),

(2, 'Mouse', 20, 200, 5),

(3, 'Keyboard', 50, 150, 4),

(4, 'Monitor', 200, 75, 5),

(5, 'Printer', 150, 40, 3),

(6, 'Tablet', 300, 60, 4),

(7, 'Smartphone', 600, 80, 5),

(8, 'Headphones', 100, 120, 4),

(9, 'Speaker', 80, 90, 5),

(10, 'Camera', 500, 30, 4),

(11, 'Webcam', 70, 100, 4),

(12, 'Microphone', 60, 85, 3),

(13, 'Charger', 30, 150, 4),

(14, 'Power Bank', 40, 120, 4),

(15, 'USB Drive', 10, 300, 5),

(16, 'External HDD', 80, 70, 4),

(17, 'Router', 90, 45, 4),

(18, 'Projector', 400, 25, 3),

(19, 'Smartwatch', 200, 65, 5),

(20, 'Fitness Tracker', 100, 110, 4);

INSERT INTO orders (order\_id, user\_id, item\_id, bill\_date, qty\_sold, tot\_amt) VALUES

(1, 1, 1, '2024-10-01', 1, 1000),

(2, 2, 2, '2024-10-02', 2, 40),

(3, 3, 3, '2024-10-03', 3, 150),

(4, 4, 4, '2024-10-04', 1, 200),

(5, 5, 5, '2024-10-05', 1, 150),

(6, 6, 6, '2024-10-06', 2, 600),

(7, 7, 7, '2024-10-07', 1, 600),

(8, 8, 8, '2024-10-08', 3, 300),

(9, 9, 9, '2024-10-09', 1, 80),

(10, 10, 10, '2024-10-10', 1, 500),

(11, 11, 11, '2024-10-11', 2, 140),

(12, 12, 12, '2024-10-12', 1, 60),

(13, 13, 13, '2024-10-13', 3, 90),

(14, 14, 14, '2024-10-14', 1, 40),

(15, 15, 15, '2024-10-15', 4, 40),

(16, 16, 16, '2024-10-16', 2, 160),

(17, 17, 17, '2024-10-17', 1, 90),

(18, 18, 18, '2024-10-18', 1, 400),

(19, 19, 19, '2024-10-19', 3, 600),

(20, 20, 20, '2024-10-20', 2, 200);

INSERT INTO membership (m\_id, user\_id, valid\_till) VALUES (1, 1, '2025-10-01'),

(2, 2, '2025-10-02'),

(3, 3, '2025-10-03'),

(4, 4, '2025-10-04'),

(5, 5, '2025-10-05'),

(6, 6, '2025-10-06'),

(7, 7, '2025-10-07'),

(8, 8, '2025-10-08'),

(9, 9, '2025-10-09'),

(10, 10, '2025-10-10'),

(11, 11, '2025-10-11'),

(12, 12, '2025-10-12'),

(13, 13, '2025-10-13'),

(14, 14, '2025-10-14'),

(15, 15, '2025-10-15'),

(16, 16, '2025-10-16'),

(17, 17, '2025-10-17'),

(18, 18, '2025-10-18'),

(19, 19, '2025-10-19'),

(20, 20, '2025-10-20');

INSERT INTO rating (user\_id, staff\_id, rating, time) VALUES (1, 1, 5, '2024-10-01'),

(2, 2, 4, '2024-10-02'),

(3, 3, 5, '2024-10-03'),

(4, 4, 4, '2024-10-04'),

(5, 5, 3, '2024-10-05'),

(6, 6, 5, '2024-10-06'),

(7, 7, 4, '2024-10-07'),

(8, 8, 5, '2024-10-08'),

(9, 9, 4, '2024-10-09'),

(10, 10, 3, '2024-10-10'),

(11, 11, 5, '2024-10-11'),

(12, 12, 4, '2024-10-12'),

(13, 13, 5, '2024-10-13'),

(14, 14, 3, '2024-10-14'),

(15, 15, 4, '2024-10-15'),

(16, 16, 5, '2024-10-16'),

(17, 17, 4, '2024-10-17'),

(18, 18, 5, '2024-10-18'),

(19, 19, 3, '2024-10-19'),

(20, 20, 4, '2024-10-20');

INSERT INTO staff (staff\_id, staff\_name, dept, sal) VALUES (1, 'Alice', 'Sales', 50000),

(2, 'Bob', 'Support', 45000),

(3, 'Charlie', 'Development', 60000),

(4, 'Dana', 'Marketing', 55000),

(5, 'Eli', 'HR', 48000),

(6, 'Faith', 'Finance', 53000),

(7, 'Gabe', 'Support', 47000),

(8, 'Hank', 'Sales', 52000),

(9, 'Ivy', 'Development', 61000),

(10, 'Jack', 'Marketing', 54000),

(11, 'Kara', 'HR', 49000),

(12, 'Liam', 'Finance', 52000),

(13, 'Mia', 'Support', 47000),

(14, 'Nina', 'Sales', 53000),

(15, 'Owen', 'Development', 60000),

(16, 'Paul', 'Marketing', 56000),

(17, 'Quinn', 'HR', 50000),

(18, 'Ruth', 'Finance', 54000),

(19, 'Steve', 'Support', 46000),

(20, 'Tina', 'Sales', 52000);

-- Display data from users table

SELECT \* FROM users;

-- Display data from item table

SELECT \* FROM item;

-- Display data from orders table

SELECT \* FROM orders;

-- Display data from membership table

SELECT \* FROM membership;

-- Display data from rating table

SELECT \* FROM rating;

-- Display data from staff table

SELECT \* FROM staff;

-- 1

SELECT o.item\_id, i.item\_name, o.tot\_amt, u.address FROM orders o

JOIN item i ON o.item\_id = i.item\_id JOIN users u ON o.user\_id = u.user\_id WHERE o.user\_id =5;

-- 2

SELECT r.staff\_id, s.staff\_name, r.rating, r.time

FROM rating r

JOIN staff s ON r.staff\_id = s.staff\_id

WHERE r.user\_id = 5;

-- 3

SELECT s.staff\_id, s.staff\_name, s.dept, s.sal

FROM staff s

JOIN rating r ON s.staff\_id = r.staff\_id

WHERE r.rating > 4;

-- 4

SELECT item\_id, item\_name, price

FROM item

WHERE stock > 0;

-- 5

SELECT o.item\_id, i.item\_name

FROM orders o

JOIN item i ON o.item\_id = i.item\_id

WHERE o.bill\_date >= '2024-10-10';

-- 6

SELECT item\_id, item\_name

FROM item

WHERE rating > 4;

-- 7

SELECT staff\_id, staff\_name

FROM staff

WHERE sal > 50000;

-- 8

SELECT item\_id, item\_name

FROM item

WHERE item\_name LIKE 'S%';

-- 9

CREATE VIEW staff\_details\_with\_rating AS

SELECT s.staff\_id, s.staff\_name, s.dept, s.sal, r.rating

FROM staff s

JOIN rating r ON s.staff\_id = r.staff\_id;

-- 10

CREATE VIEW available\_items AS

SELECT item\_id, item\_name, price, stock

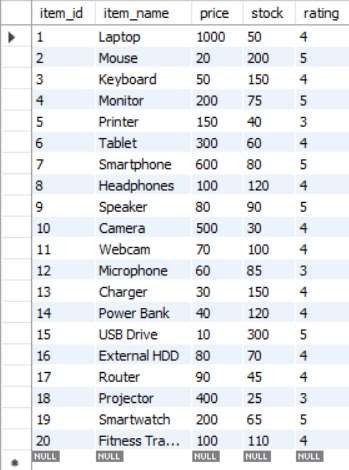
FROM item

WHERE stock > 0;

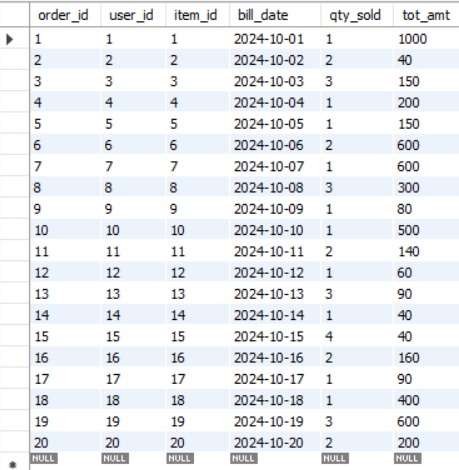
**OUTPUTS:**

**USER TABLE:**

**ITEM TABLE:**



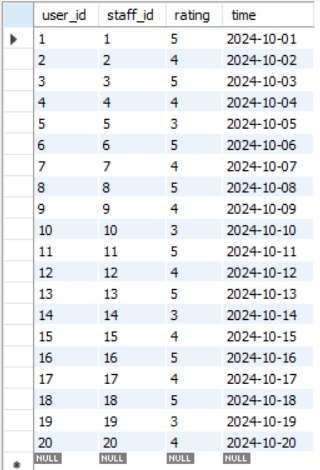
**ORDER TABLE:**



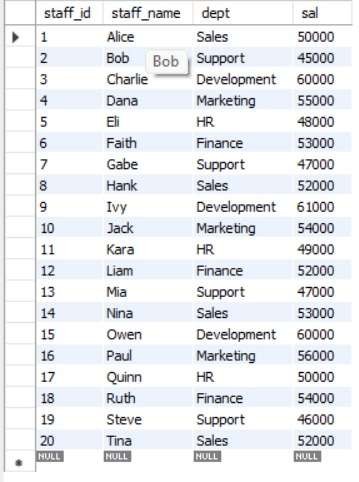
**MEMBERSHIP TABLE:**



**RATING TABLE:**



**STAFF TABLE:**



Display the items ordered by the user along with total amount and

address to deliver where item id=5;



Display the ratings given by the user to the staff whsoe staff\_id=5;

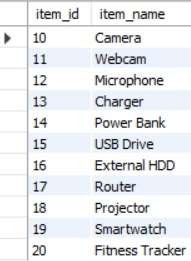
Display the staff details whose rating is greater than 4;

Display the items which are available to the users along with cost;

. Display the items which are deliver on date 20-05- 2024;

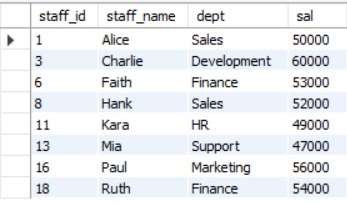


Display the items whose rating is greater than 4;



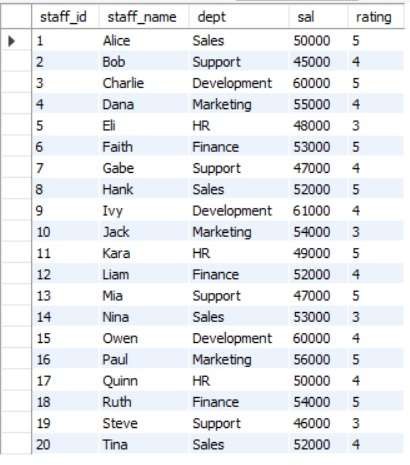
. Display the staff names along with their ids whose salary is greater than

50000;

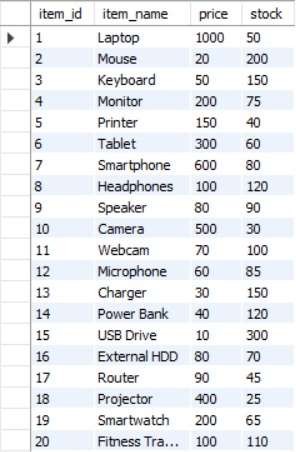


Display the items id and item names whose letter starts with letter ‘s’;

Create a view which contains the all staff details along with rating ;



Create a view of the items which are available in the stock along with price and quantity;

**Conclusion**

The implementation of this DBMS project for an ecommerce platform demonstrates the critical role of structured data management in modern online retail. By effectively managing user information, order details, item inventories, memberships, and ratings, the system ensures operational efficiency and enhances customer satisfaction. The relationships established between various entities such as users, orders, items, memberships, and staff provide a holistic view of the platform's ecosystem, enabling better decisionmaking and personalized user interactions. This project underscores the importance of a welldesigned database in supporting the dynamic needs of an e-commerce business, paving the way for future enhancements and innovation.