Day 3 Assignments:

Schema and Table Structure:

1.  Users Table: Contains information about customers.

CREATE TABLE Users (

    user\_id INT PRIMARY KEY,

    username VARCHAR2(50) NOT NULL,

    password VARCHAR2(50) NOT NULL,

    email VARCHAR2(100),

    first\_name VARCHAR2(50),

    last\_name VARCHAR2(50),

    created\_at DATE DEFAULT SYSDATE

);

2.  Products Table: Contains details about the products that users can add to their cart.

CREATE TABLE Products (

    product\_id INT PRIMARY KEY,

    product\_name VARCHAR2(100) NOT NULL,

    description VARCHAR2(500),

    price NUMBER(10, 2) NOT NULL,

    stock\_quantity INT NOT NULL,

    created\_at DATE DEFAULT SYSDATE

);

3.  Shopping Cart Table: A temporary cart to hold products that a user adds before checking out.

CREATE TABLE ShoppingCart (

    cart\_id INT PRIMARY KEY,

    user\_id INT,

    product\_id INT,

    quantity INT,

    added\_at DATE DEFAULT SYSDATE,

    FOREIGN KEY (user\_id) REFERENCES Users(user\_id),

    FOREIGN KEY (product\_id) REFERENCES Products(product\_id)

);

4.  Orders Table: To store orders once the user checks out.

CREATE TABLE Orders (

    order\_id INT PRIMARY KEY,

    user\_id INT,

    order\_date DATE DEFAULT SYSDATE,

    total\_amount NUMBER(10, 2),

    status VARCHAR2(20),

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

);

5.  Order Details Table: Stores the products that are part of an order.

CREATE TABLE OrderDetails (

    order\_detail\_id INT PRIMARY KEY,

    order\_id INT,

    product\_id INT,

    quantity INT,

    price NUMBER(10, 2),

    FOREIGN KEY (order\_id) REFERENCES Orders(order\_id),

    FOREIGN KEY (product\_id) REFERENCES Products(product\_id)

);

-- Insert into Users

INSERT INTO Users (user\_id, username, password, email, first\_name, last\_name, created\_at) VALUES

(1, 'johndoe', 'password123', 'john.doe@example.com', 'John', 'Doe', TO\_DATE('2025-01-01 10:00:00', 'YYYY-MM-DD HH24:MI:SS')),

(2, 'janedoe', 'password456', 'jane.doe@example.com', 'Jane', 'Doe', TO\_DATE('2025-01-02 11:00:00', 'YYYY-MM-DD HH24:MI:SS')),

(3, 'mike123', 'pass987', 'mike@example.com', 'Mike', 'Johnson', TO\_DATE('2025-01-03 12:00:00', 'YYYY-MM-DD HH24:MI:SS')),

(4, 'sarah\_23', 'hello123', 'sarah23@example.com', 'Sarah', 'Connor', TO\_DATE('2025-01-04 14:00:00', 'YYYY-MM-DD HH24:MI:SS')),

(5, 'daveking', 'kingdave5', 'daveking@example.com', 'Dave', 'King', TO\_DATE('2025-01-05 09:00:00', 'YYYY-MM-DD HH24:MI:SS')),

(6, 'emily\_77', 'emily789', 'emily77@example.com', 'Emily', 'Clark', TO\_DATE('2025-01-06 08:30:00', 'YYYY-MM-DD HH24:MI:SS')),

(7, 'robert12', 'secure123', 'robert12@example.com', 'Robert', 'Lee', TO\_DATE('2025-01-07 10:00:00', 'YYYY-MM-DD HH24:MI:SS'));

-- Insert into Products

INSERT INTO Products (product\_id, product\_name, description, price, stock\_quantity, created\_at) VALUES

(1, 'Laptop', 'High-end gaming laptop', 1200.00, 10, TO\_DATE('2025-01-01 10:00:00', 'YYYY-MM-DD HH24:MI:SS')),

(2, 'Smartphone', 'Latest 5G smartphone', 800.00, 15, TO\_DATE('2025-01-01 11:00:00', 'YYYY-MM-DD HH24:MI:SS')),

(3, 'Wireless Mouse', 'Ergonomic wireless mouse', 50.00, 25, TO\_DATE('2025-01-02 12:00:00', 'YYYY-MM-DD HH24:MI:SS')),

(4, 'Headphones', 'Noise-cancelling headphones', 150.00, 30, TO\_DATE('2025-01-03 10:30:00', 'YYYY-MM-DD HH24:MI:SS')),

(5, 'Tablet', '10-inch Android tablet', 300.00, 20, TO\_DATE('2025-01-04 11:00:00', 'YYYY-MM-DD HH24:MI:SS')),

(6, 'Smart Watch', 'Fitness tracking smartwatch', 200.00, 40, TO\_DATE('2025-01-05 14:00:00', 'YYYY-MM-DD HH24:MI:SS')),

(7, 'Bluetooth Speaker', NULL, 70.00, 50, TO\_DATE('2025-01-06 09:00:00', 'YYYY-MM-DD HH24:MI:SS'));

-- Insert into ShoppingCart

INSERT INTO ShoppingCart (cart\_id, user\_id, product\_id, quantity, added\_at) VALUES

(1, 1, 1, 5, TO\_DATE('2025-01-15 14:00:00', 'YYYY-MM-DD HH24:MI:SS')),

(2, 1, 2, 1, TO\_DATE('2025-01-15 14:05:00', 'YYYY-MM-DD HH24:MI:SS')),

(3, 2, 3, 3, TO\_DATE('2025-01-16 15:00:00', 'YYYY-MM-DD HH24:MI:SS')),

(4, 3, 4, 2, TO\_DATE('2025-01-17 10:30:00', 'YYYY-MM-DD HH24:MI:SS')),

(5, 4, 5, 1, TO\_DATE('2025-01-18 11:15:00', 'YYYY-MM-DD HH24:MI:SS')),

(6, 5, 6, 1, TO\_DATE('2025-01-19 12:00:00', 'YYYY-MM-DD HH24:MI:SS')),

(7, 6, 7, 0, TO\_DATE('2025-01-20 14:30:00', 'YYYY-MM-DD HH24:MI:SS'));

-- Insert into Orders

INSERT INTO Orders (order\_id, user\_id, order\_date, total\_amount, status) VALUES

(1, 1, TO\_DATE('2025-01-16 16:00:00', 'YYYY-MM-DD HH24:MI:SS'), 2400.00, 'Pending'),

(2, 2, TO\_DATE('2025-01-17 10:00:00', 'YYYY-MM-DD HH24:MI:SS'), 150.00, 'Pending'),

(3, 3, TO\_DATE('2025-01-18 12:30:00', 'YYYY-MM-DD HH24:MI:SS'), 300.00, 'Pending'),

(4, 4, TO\_DATE('2025-01-19 14:00:00', 'YYYY-MM-DD HH24:MI:SS'), 450.00, 'Pending'),

(5, 5, TO\_DATE('2025-01-20 09:00:00', 'YYYY-MM-DD HH24:MI:SS'), 200.00, 'Pending'),

(6, 6, TO\_DATE('2025-01-21 15:00:00', 'YYYY-MM-DD HH24:MI:SS'), 140.00, 'Pending'),

(7, 7, TO\_DATE('2025-01-22 17:00:00', 'YYYY-MM-DD HH24:MI:SS'), 140.00, 'Pending');

-- Insert into OrderDetails

INSERT INTO OrderDetails (order\_detail\_id, order\_id, product\_id, quantity, price) VALUES

(1, 1, 1, 5, 1200.00),

(2, 1, 2, 1, 800.00),

(3, 2, 3, 3, 50.00),

(4, 3, 4, 2, 150.00),

(5, 4, 5, 1, 300.00),

(6, 5, 6, 1, 200.00),

(7, 6, 7, 0, 70.00);

1.  If the product description is NULL, replace it with 'No description available':

<SQL Solution>

2.  Find users who have more than 5 items in their cart:

<SQL Solution>

3.  Find the year and month when products were created:

<SQL Solution>

4.  Find users who have more than 3 items in their cart:

<SQL Solution>

5.  Find the number of products in each category:

<SQL Solution>

6.  Find the maximum and minimum prices of products in each category:

<SQL Solution>

7.  Find categories with more than 5 products:

<SQL Solution>

8.  Get the total price for each category and order the results by the total price:

<SQL Solution>

9.  Find the total price of products sold per year (assuming there's a sale\_date column) and orders them chronologically.

<SQL Solution>

10. Retrieve all users and their corresponding orders (if any).

<SQL Solution>

11. Retrieve all orders and their corresponding users (if any).

<SQL Solution>

12. Find pairs of products where the price of one product is greater than the other.

<SQL Solution>

13. Retrieve all users and orders, including unmatched records from both tables.

<SQL Solution>