SQL CODING CHALLENGE
E-COMMERCE

P218-NALLAPANENI PENCHALA BALA TEJA

```
create database ecom;
use ecom
CREATE TABLE customers (
customer_id INT PRIMARY KEY,
name VARCHAR(50) NOT NULL,
email VARCHAR(50) NOT NULL,
password VARCHAR(100) NOT NULL
);

INSERT INTO customers (customer_id, name, email, password) VALUES
(1, 'John Doe', 'johndoe@example.com', 'password123'),
(2, 'Jane Smith', 'janesmith@example.com', 'password123'),
(3, 'Robert Johnson', 'robert@example.com', 'password123'),
(4, 'Sarah Brown', 'sarah@example.com', 'password123'),
(5, 'David Lee', 'david@example.com', 'password123'),
(6, 'Laura Hall', 'laura@example.com', 'password123'),
(7, 'Michael Davis', 'michael@example.com', 'password123'),
(8, 'Emma Wilson', 'emma@example.com', 'password123'),
(9, 'William Taylor', 'william@example.com', 'password123'),
(10, 'Olivia Adams', 'olivia@example.com', 'password123');

CREATE TABLE products (
product_id INT PRIMARY KEY,
name VARCHAR(50) NOT NULL,
price DECIMAL(8, 2) NOT NULL,
description TEXT,
stockQuantity INT
);

INSERT INTO products (product_id, name, description, price,
stockQuantity)VALUES
(1, 'Laptop', 'High-performance laptop', 800.00, 10),
(2, 'Smartphone', 'Latest smartphone', 600.00, 15),
(3, 'Tablet', 'Portable tablet', 300.00, 20),
(4, 'Headphones', 'Noise-canceling', 150.00, 30),
(5, 'TV', '4K Smart TV', 900.00, 5),
(6, 'Coffee Maker', 'Automatic coffee maker', 50.00, 25),
(7, 'Refrigerator', 'Energy-efficient', 700.00, 10),
(8, 'Microwave Oven', 'Countertop microwave', 80.00, 15),
(9, 'Blender', 'High-speed blender', 70.00, 20),
(10, 'Vacuum Cleaner', 'Bagless vacuum cleaner', 120.00, 10);
```

```sql
CREATE TABLE cart (
cart_id INT PRIMARY KEY,
customer_id INT,
product_id INT,
quantity INT NOT NULL,
FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
FOREIGN KEY (product_id) REFERENCES products(product_id));
INSERT INTO cart (cart_id, customer_id, product_id, quantity) VALUES
(1, 1, 1, 2),
(2, 1, 3, 1),
(3, 2, 2, 3),
(4, 3, 4, 4),
(5, 3, 5, 2),
(6, 4, 6, 1),
(7, 5, 1, 1),
(8, 6, 10, 2),
(9, 6, 9, 3),
(10, 7, 7, 2);


CREATE TABLE orders (
order_id INT PRIMARY KEY,
customer_id INT,
order_date DATE,
total_price DECIMAL(8, 2) ,
shipping_address VARCHAR(50),
FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);

INSERT INTO orders (order_id, customer_id, order_date,
total_price,shipping_address) VALUES
(1, 1, '2023-01-05', 1200.00, '123 Main St, City'),
(2, 2, '2023-02-10', 900.00, '456 Elm St, Town'),
(3, 3, '2023-03-15', 300.00, '789 Oak St, Village'),
(4, 4, '2023-04-20', 150.00, '101 Pine St, Suburb'),
(5, 5, '2023-05-25', 1800.00, '234 Cedar St, District'),
(6, 6, '2023-06-30', 400.00, '567 Birch St, County'),
(7, 7, '2023-07-05', 700.00, '890 Maple St, State'),
(8, 8, '2023-08-10', 160.00, '321 Redwood St, Country'),
(9, 9, '2023-09-15', 140.00, '432 Spruce St, Province'),
(10, 10, '2023-10-20', 1400.00, '765 Fir St, Territory');


CREATE TABLE order_items (
order_item_id INT PRIMARY KEY,
order_id INT,
product_id INT,
quantity INT NOT NULL,
FOREIGN KEY (order_id) REFERENCES orders(order_id),
FOREIGN KEY (product_id) REFERENCES products(product_id)
);

INSERT INTO order_items (order_item_id, order_id, product_id,
quantity)VALUES
(1, 1, 1, 2),
(2, 1, 3, 1),
(3, 2, 2, 3),
(4, 3, 5, 2),
(5, 4, 4, 4),
(6, 4, 6, 1),
(7, 5, 1, 1),
(8, 5, 2, 2),
(9, 6, 10, 2),
(10, 6, 9, 3);


--1.Update refrigerator product price to 800.
UPDATE products SET price = 800 WHERE name='refrigerator';



--2. Remove all cart items for a specific customer.
delete from cart where customer_id=1;



--3. Retrieve Products Priced Below $100.
select * from products where price < 100.00;



--4. Find Products with Stock Quantity Greater Than5.
select product_id,name,stockQuantity from products where
stockQuantity > 5;



--5. Retrieve Orders with Total Amount Between $500and$1000.
select * from orders where total_price between 500 and 1000;



--6. Find Products which name end with letter ‘r’.
select * from products where name like '%r';



--7. Retrieve Cart Items for Customer 5.
```

```
select * from cart where customer_id=5;


-- 8. Find Customers Who Placed Orders in 2023.
select c.first_name,c.last_name,o.order_date from
customers c join orders o on c.customer_id=o.customer_id where
o.order_date like '2023%';

--9. Determine the Minimum Stock Quantity for EachProductCategory.
select min(stockquantity) as min_stockquantity from products



--10. Calculate the Total Amount Spent by Each Customer.
select customer_id,total_price from orders



--11. Find the Average Order Amount for Each Customer.
SELECT c.customer_id, c.name, AVG(o.total_price)
ASavgOrderAmountFROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.name;



--12. Count the Number of Orders Placed.
select customer_id,sum(quantity) as orders_placed from cart group by
customer_id



--13. Find the Maximum Order Amount for Each Customer.
select customer_id,max(total_price) as max_order_amount from orders
group by customer_id;



--14. Get Customers Who Placed Orders Totaling Over$1000.
select * from orders where total_price > 1000



--15. Subquery to Find Products Not in the Cart.
select p.product_id,c.quantity from products p
left join cart c on p.product_id=c.product_id
where quantity is null
```

```sql
--16. Subquery to Find Customers Who Haven't PlacedOrders.
select c.* from customers c
left join orders o on c.customer_id=o.customer_id where o.order_id
is null


--17. Subquery to Calculate the Percentage of
TotalRevenueforaProduct.
select * ,(price*stockquantity) as total_revenue ,((price*
stockquantity)/100)
as revenue_percentage from products


--18. Subquery to Find Products with Low Stock.
select * from products where stockquantity < (select avg
(stockquantity) from products):


--19. Subquery to Find Customers Who Placed High-ValueOrders.SELECT
DISTINCT c.customer_id, o.total_price
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
WHERE o.total_price > 1000;
```