SQL-Coding Challenge

Ecommerce – SQL (PDF 6)

Table creation and inserting values (According to given pdf)

```
create table customers (
    customer_id int primary key,
    first_name varchar(50),
    last_name varchar(50),
    email varchar(100),
    address varchar(100)
);


insert into customers values
(1, 'john', 'doe', 'johndoe@example.com', '123 Main St, City'),
(2, 'jane', 'smith', 'janesmith@example.com', '456 Elm St, Town'),
(3, 'robert', 'johnson', 'robert@example.com', '789 Oak St, Village'),
(4, 'sarah', 'brown', 'sarah@example.com', '101 Pine St, Suburb'),
(5, 'david', 'lee', 'david@example.com', '234 Cedar St, District'),
(6, 'laura', 'hall', 'laura@example.com', '567 Birch St, County'),
(7, 'michael', 'davis', 'michael@example.com', '890 Maple St, State'),
(8, 'emma', 'wilson', 'emma@example.com', '321 Redwood St, Country'),
(9, 'william', 'taylor', 'william@example.com', '432 Spruce St, Province'),
(10, 'olivia', 'adams', 'olivia@example.com', '765 Fir St, Territory');
```

```
create table products (
    product_id int primary key,
    name varchar(100),
    description varchar(255),
    price decimal(10,2),
    stock_quantity int
);

insert into products 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);
```

```
create table cart (
    cart_id int primary key,
    customer_id int,
    product_id int,
    quantity int,
    foreign key (customer_id) references customers(customer_id),
    foreign key (product_id) references products(product_id)
);


insert into cart 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(10,2),
    shipping_address varchar(255),
    foreign key (customer_id) references customers(customer_id)
);


insert into orders 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,
    item_amount decimal(10,2),
    foreign key (order_id) references orders(order_id),
    foreign key (product_id) references products(product_id)
);

insert into order_items values
(1, 1, 1, 2, 1600.00),
(2, 1, 3, 1, 300.00),
(3, 2, 2, 3, 1800.00),
(4, 3, 5, 2, 1800.00),
(5, 4, 4, 4, 600.00),
(6, 4, 6, 1, 50.00),
(7, 5, 1, 1, 800.00),
(8, 5, 2, 2, 1200.00),
(9, 6, 10, 2, 240.00),
(10, 6, 9, 3, 210.00);
```

1. Update refrigerator product price to 800

```
update products set price = 800.00 where name = 'refrigerator';
```

2. Remove all cart items for a specific customer (e.g., customer_id = 3)

```
delete from cart where customer_id = 3;
```

3. Retrieve products priced below $100

```
select * from products where price < 100;
```

4. Find products with stock quantity greater than 5

```
select * from products where stock_quantity > 5;
```

5. Retrieve orders with total amount between $500 and $1000

```
select * from orders where total_price between 500 and 1000;
```

6. Find products whose names end with the 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 distinct c.* from customers c
join orders o on c.customer_id = o.customer_id
where year(order_date) = 2023;
```

9. Determine the minimum stock quantity for each product name

```
select name, min(stock_quantity) as min_stock
from products
group by name;
```

10. Calculate the total amount spent by each customer

```
select c.customer_id, c.first_name, c.last_name, sum(o.total_price) as total_spent
from customers c
join orders o on c.customer_id = o.customer_id
group by c.customer_id, c.first_name, c.last_name;
```

11. Find the average order amount for each customer

```
select c.customer_id, c.first_name, c.last_name, avg(o.total_price) as avg_order_amount
from customers c
join orders o on c.customer_id = o.customer_id
group by c.customer_id, c.first_name, c.last_name;
```

12. Count the number of orders placed by each customer

```
select c.customer_id, c.first_name, c.last_name, count(o.order_id) as order_count
from customers c
join orders o on c.customer_id = o.customer_id
group by c.customer_id, c.first_name, c.last_name;
```

13. Find the maximum order amount for each customer

```
select c.customer_id, c.first_name, c.last_name, max(o.total_price) as max_order_amount
from customers c
join orders o on c.customer_id = o.customer_id
group by c.customer_id, c.first_name, c.last_name;
```

14. Get customers who placed orders totaling over $1000

```
select c.customer_id, c.first_name, c.last_name, sum(o.total_price) as total_spent
from customers c
join orders o on c.customer_id = o.customer_id
group by c.customer_id, c.first_name, c.last_name
having sum(o.total_price) > 1000;
```

15. Subquery to find products not in the cart

```
select * from products where product_id not in (select product_id from cart);
```

16. Subquery to find customers who haven't placed orders

```
select * from customers where customer_id not in (select customer_id from orders);
```

17. Subquery to calculate the percentage of total revenue for a product

```
select name, round(
        100.0 * sum(item_amount) / (select sum(item_amount) from order_items), 2
    ) as revenue_percentage
from products p
join order_items o on p.product_id = o.product_id
group by name;
```

18. Subquery to find products with low stock (less than average stock)

```
select * from products
where stock_quantity < (
    select avg(stock_quantity) from products
);
```

19. Subquery to find customers who placed high-value orders (orders over $1000)

```
select c.* from customers c where customer_id in (
    select customer_id
    from orders
    where total_price > 1000
);
```