My SQL Code challenge- Ecommerce Task

**Table creation🡪**

**CREATE TABLE customers (**

**customer\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**name VARCHAR(100),**

**email VARCHAR(100) UNIQUE,**

**password VARCHAR(255)**

**);**

**CREATE TABLE products (**

**product\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**name VARCHAR(100),**

**price DECIMAL(10, 2),**

**description TEXT,**

**stockQuantity INT**

**);**

**CREATE TABLE cart (**

**cart\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**customer\_id INT,**

**product\_id INT,**

**quantity INT,**

**FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id) ON DELETE SET NULL,**

**FOREIGN KEY (product\_id) REFERENCES products(product\_id) ON DELETE SET NULL**

**);**

**CREATE TABLE orders (**

**order\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**customer\_id INT,**

**order\_date DATETIME,**

**total\_price DECIMAL(10, 2),**

**shipping\_address VARCHAR(255),**

**FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id) ON DELETE SET NULL**

**);**

**CREATE TABLE order\_items (**

**order\_item\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**order\_id INT,**

**product\_id INT,**

**quantity INT,**

**FOREIGN KEY (order\_id) REFERENCES orders(order\_id) ON DELETE SET NULL,**

**FOREIGN KEY (product\_id) REFERENCES products(product\_id) ON DELETE SET NULL**

**);**

**INSERT-🡪 CUSTOMERS**

**INSERT INTO customers (name, email, password) VALUES**

**('Alice Johnson', 'alice@example.com', 'password123'),**

**('Bob Smith', 'bob@example.com', 'password456'),**

**('Charlie Brown', 'charlie@example.com', 'password789'),**

**('Diana Prince', 'diana@example.com', 'passwordabc'),**

**('Ethan Hunt', 'ethan@example.com', 'passwordxyz'),**

**('Fiona Gallagher', 'fiona@example.com', 'passwordlmn'),**

**('George Washington', 'george@example.com', 'passwordopq');**

**INSERT-🡪PRODUCTS**

**INSERT INTO products (name, price, description, stockQuantity) VALUES**

**('Wireless Mouse', 25.99, 'A wireless mouse with ergonomic design.', 150),**

**('Bluetooth Keyboard', 45.50, 'A Bluetooth keyboard with backlit keys.', 75),**

**('USB-C Hub', 29.99, 'Multi-port USB-C hub for connecting devices.', 100),**

**('Gaming Headset', 75.00, 'Comfortable gaming headset with surround sound.', 50),**

**('External SSD', 120.00, 'Fast external SSD for storage and backups.', 30),**

**('HDMI Cable', 15.99, 'High-speed HDMI cable for clear video and audio.', 200),**

**('Portable Charger', 35.00, 'Compact portable charger for on-the-go charging.', 80);**

**INSERT-🡪CART**

**INSERT INTO cart (customer\_id, product\_id, quantity) VALUES**

**(1, 1, 2),**

**(1, 3, 1),**

**(2, 5, 1),**

**(3, 2, 2),**

**(4, 4, 1),**

**(5, 6, 5),**

**(6, 7, 3);**

**INSERT🡪ORDERS**

**INSERT INTO orders (customer\_id, order\_date, total\_price, shipping\_address) VALUES**

**(1, '2024-10-01 10:30:00', 77.98, '123 Main St, Cityville, ST 12345'),**

**(2, '2024-10-02 12:15:00', 120.00, '456 Elm St, Townsville, ST 67890'),**

**(3, '2024-10-03 14:45:00', 91.00, '789 Oak St, Villageburg, ST 11223'),**

**(4, '2024-10-03 15:00:00', 75.00, '321 Pine St, Hamletton, ST 33445'),**

**(5, '2024-10-03 16:30:00', 79.99, '654 Maple St, Countryside, ST 55667');**

**INSERT 🡪order\_items**

**INSERT INTO order\_items (order\_id, product\_id, quantity) VALUES**

**(1, 1, 2),**

**(1, 3, 1),**

**(2, 5, 1),**

**(3, 2, 2),**

**(4, 4, 1),**

**(5, 6, 5);**

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=(select customer\_id from customers where name='Alice Johnson');**

3. Retrieve Products Priced Below $100.

**SELECT \* from Products WHERE price<100;**

4. Find Products with Stock Quantity Greater Than 5.

**SELECT name from products where product\_id IN(select product\_id from cart WHERE 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 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 \* from customer WHERE customer\_id IN(select customer\_id from orders where order\_date BETWEEN ‘2023-01-01’ and ‘2023-12-31’);**

9. Determine the Minimum Stock Quantity for Each Product Category.

**ALTER TABLE products**

**ADD category VARCHAR(100);**

**UPDATE products SET category = 'Accessories' WHERE product\_id = 1;**

**UPDATE products SET category = 'Accessories' WHERE product\_id = 2;**

**UPDATE products SET category = 'Accessories' WHERE product\_id = 3;**

**UPDATE products SET category = 'Gaming' WHERE product\_id = 4;**

**UPDATE products SET category = 'Storage' WHERE product\_id = 5;**

**UPDATE products SET category = 'Cables' WHERE product\_id = 6;**

**UPDATE products SET category = 'Accessories' WHERE product\_id = 7;**

**SELECT p.category,MIN(p.stockQuantity) AS minimum\_stock\_quantity FROM products p GROUP BY p.category ORDER BY p.category;**

10. Calculate the Total Amount Spent by Each Customer.

**SELECT c.customer\_id,c.name,COALESCE(SUM(o.total\_price), 0) AS total\_amount\_spent FROM customers c LEFT JOIN orders o ON c.customer\_id = o.customer\_id GROUP BY c.customer\_id;**

11. Find the Average Order Amount for Each Customer.

**SELECT c.customer\_id,c.name,IFNULL(AVG(o.total\_price),0) AS average\_order\_amount FROM customers c LEFT JOIN orders o ON c.customer\_id = o.customer\_id GROUP BY c.customer\_id, c.name;**

12. Count the Number of Orders Placed by Each Customer.

**SELECT c.customer\_id,c.name,COUNT(o.order\_id) AS order\_count FROM customers c LEFT JOIN orders o ON c.customer\_id = o.customer\_id GROUP BY c.customer\_id, c.name;**

13. Find the Maximum Order Amount for Each Customer.

**SELECT c.name,COALESCE(a.Maximum\_order, 0) AS Maximum\_order FROM customers c LEFT JOIN (SELECT customer\_id, MAX(total\_price) AS Maximum\_order FROM orders GROUP BY customer\_id) a ON a.customer\_id = c.customer\_id;**

14. Get Customers Who Placed Orders Totaling Over $1000.

**select name from customers where customer\_id in (select customer\_id from orders group by customer\_id having sum(total\_price)>1000);**

15. Subquery to Find Products Not in the Cart.

**select name from products where product\_id not in (select product\_id from cart);**

16. Subquery to Find Customers Who Haven't Placed Orders.

**select name 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 p.product\_id,p.name,(SUM(o.total\_price) / (SELECT SUM(total\_price) FROM orders)) \* 100 AS revenue\_percentage FROM orders o JOIN order\_items oi ON o.order\_id = oi.order\_id JOIN products p ON oi.product\_id = p.product\_id WHERE p.name='Bluetooth Keyboard' GROUP BY p.product\_id;**

18. Subquery to Find Products with Low Stock.

**select name,stockQuantity from products where stockQuantity in (select min(stockQuantity) from products**);

19. Subquery to Find Customers Who Placed High-Value Orders.

**select name from customers where customer\_id=(select customer\_id from orders where total\_price=(select max(total\_price) from orders));**