ECOMMERCE – CODING CHALLENGE

- VIKAS REDDY GORANTLA

1) Define the schema for the customers, products,cart, orders, order_items and Inventory tables based on the provided schema.

```
create table customers(
customer_id int IDENTITY PRIMARY KEY,
name varchar(20),
email varchar(300),
password varchar(300)
);

create table products(
product_id int PRIMARY KEY,
name varchar(30),
price decimal(10,2),
description varchar(200),
stockQuantity int
);

create table cart(
cart_id int PRIMARY KEY,
customer_id INT,
product_id INT,
Quantity INT,
FOREIGN KEY(customer_id) REFERENCES customers(customer_id) ON DELETE CASCADE,
FOREIGN KEY(product_id) REFERENCES products(product_id) ON DELETE CASCADE);
```

```
create table orders(
order_id int PRIMARY KEY,
customer_id INT,
order_date date,
total_price decimal(10,2),
shipping_address varchar(50),
FOREIGN KEY(customer_id) REFERENCES customers(customer_id) ON DELETE CASCADE
);

create table order_items(
order_item_id int PRIMARY KEY,
order_id INT,
product_id INT,
Quantity INT,
FOREIGN KEY(order_id) REFERENCES orders(order_id) ON DELETE CASCADE,
FOREIGN KEY(product_id) REFERENCES products(product_id) ON DELETE CASCADE);
```

2)Insert at least 10 sample records into each of the following tables.

a. Customers b. Products c. cart d. orders e. order_items

a) customers :

```
insert into customers(name,email) values('John','john21@gmail.com ');
insert into customers(name,email) values('Jane','jane34@gmail.com ');
insert into customers(name,email) values('Robert','robert67@gmail.com
');
insert into customers(name,email) values('Sarah','sarah98@gmail.com ');
insert into customers(name,email) values('David','david14@gmail.com ');
insert into customers(name,email) values('Laura','laura54@gmail.com ');
insert into customers(name,email)
values('Michael','michael78@gmail.com ');
insert into customers(name,email) values('Emma','emma40@gmail.com’);
insert into customers(name,email) values('William','william87@gmail.com
');
insert into customers(name,email) values('Olivia','olivia10@gmail.com ');
```

b) products :

```
INSERT INTO products VALUES(101, 'Laptop', 999.99, 'High-performance
laptop', 10);
INSERT INTO products VALUES(102, 'Smartphone', 499.99, 'Latest
smartphone model', 20);
INSERT INTO products VALUES(103, 'Headphones', 199.99, 'Noise-
cancelling headphones', 15);
INSERT INTO products VALUES(104, 'Smartwatch', 299.99, 'Stylish
smartwatch', 25);
```

INSERT INTO products VALUES(105, 'Tablet', 399.99, 'Lightweight tablet', 30);

INSERT INTO products VALUES(106, 'Camera', 799.99, 'Digital camera with 4K recording', 8);

INSERT INTO products VALUES(107, 'Keyboard', 89.99, 'Mechanical keyboard', 50);

INSERT INTO products VALUES(108, 'Mouse', 39.99, 'Wireless ergonomic mouse', 45);

INSERT INTO products VALUES(109, 'Refrigerator', 249.99, '5 Star Rated', 12);

INSERT INTO products VALUES(110, 'Printer', 159.99, 'All-in-one printer', 18);

INSERT INTO products VALUES(111, 'SmartTv', 25000.99, 'Latest smart Tv', 12);

INSERT INTO products VALUES(112, 'Samsung EarBuds', 2599.99, 'Wireless Earbuds', 18);

c) cart :

INSERT INTO cart VALUES(201, 1, 101, 1);

INSERT INTO cart VALUES(202, 2, 102, 2);

INSERT INTO cart VALUES(203, 3, 103, 1);

INSERT INTO cart VALUES(204, 4, 104, 1);

INSERT INTO cart VALUES(205, 5, 105, 1);

INSERT INTO cart VALUES(206, 6, 106, 1);

INSERT INTO cart VALUES(207, 7, 107, 2);

INSERT INTO cart VALUES(208, 8, 108, 1);

INSERT INTO cart VALUES(209, 9, 109, 1);

INSERT INTO cart VALUES(210, 10, 110, 1);

d) orders :

INSERT INTO orders VALUES(301, 1, '2024-09-01', 999.99, '5th Avenue, NewYork');

INSERT INTO orders VALUES(302, 2, '2024-09-02', 999.98, 'Sunset Boulevard, Chicago');

INSERT INTO orders VALUES(303, 3, '2024-09-03', 199.99, 'State Street, LasVegas');

INSERT INTO orders VALUES(304, 4, '2024-09-04', 299.99, 'Main Street, Orlando');

INSERT INTO orders VALUES(305, 5, '2024-09-05', 399.99, 'Camelback Road, Aspen');

INSERT INTO orders VALUES(306, 6, '2024-09-06', 799.99, 'Alamo Plaza, Columbus');

INSERT INTO orders VALUES(307, 7, '2024-09-07', 179.98, 'Harbor Drive, St.Louis');

INSERT INTO orders VALUES(308, 8, '2024-09-08', 39.99, 'Almaden Boulevard, Cincinnati');

INSERT INTO orders VALUES(309, 9, '2024-09-09', 249.99, 'Market Street, Dallas');

INSERT INTO orders VALUES(310, 10, '2024-09-10', 159.99, 'Mockingbird Lane, Washington, D.C.');

INSERT INTO orders VALUES(311, 1, '2024-09-15', 159.99, 'School Lane, Texas');

INSERT INTO orders VALUES(312, 3, '2024-09-20', 799.99, 'Rock Street, California');

INSERT INTO orders VALUES(314, 5, '2024-09-18', 999.99, 'Garden House, Orlando');

INSERT INTO orders VALUES(315, 8, '2024-09-30', 199.99, 'Mockingbird Lane, Aspen');

INSERT INTO orders VALUES(316, 2, '2024-09-14', 25000.99, 'Hospital Lane, Texas');

INSERT INTO orders VALUES(317, 4, '2024-09-29', 799.99, 'Hirecy Street, California');

INSERT INTO orders VALUES(318, 6, '2024-09-19', 999.99, 'Jake House, Orlando');

INSERT INTO orders VALUES(319, 9, '2024-09-28', 2599.99, 'Mockingbird Lane, Aspen');

e) order_items :

INSERT INTO order_items VALUES(401, 301, 101, 1);

INSERT INTO order_items VALUES(402, 302, 102, 2);

INSERT INTO order_items VALUES(403, 303, 103, 1);

INSERT INTO order_items VALUES(404, 304, 104, 1);

INSERT INTO order_items VALUES(405, 305, 105, 1);

INSERT INTO order_items VALUES(406, 306, 106, 1);

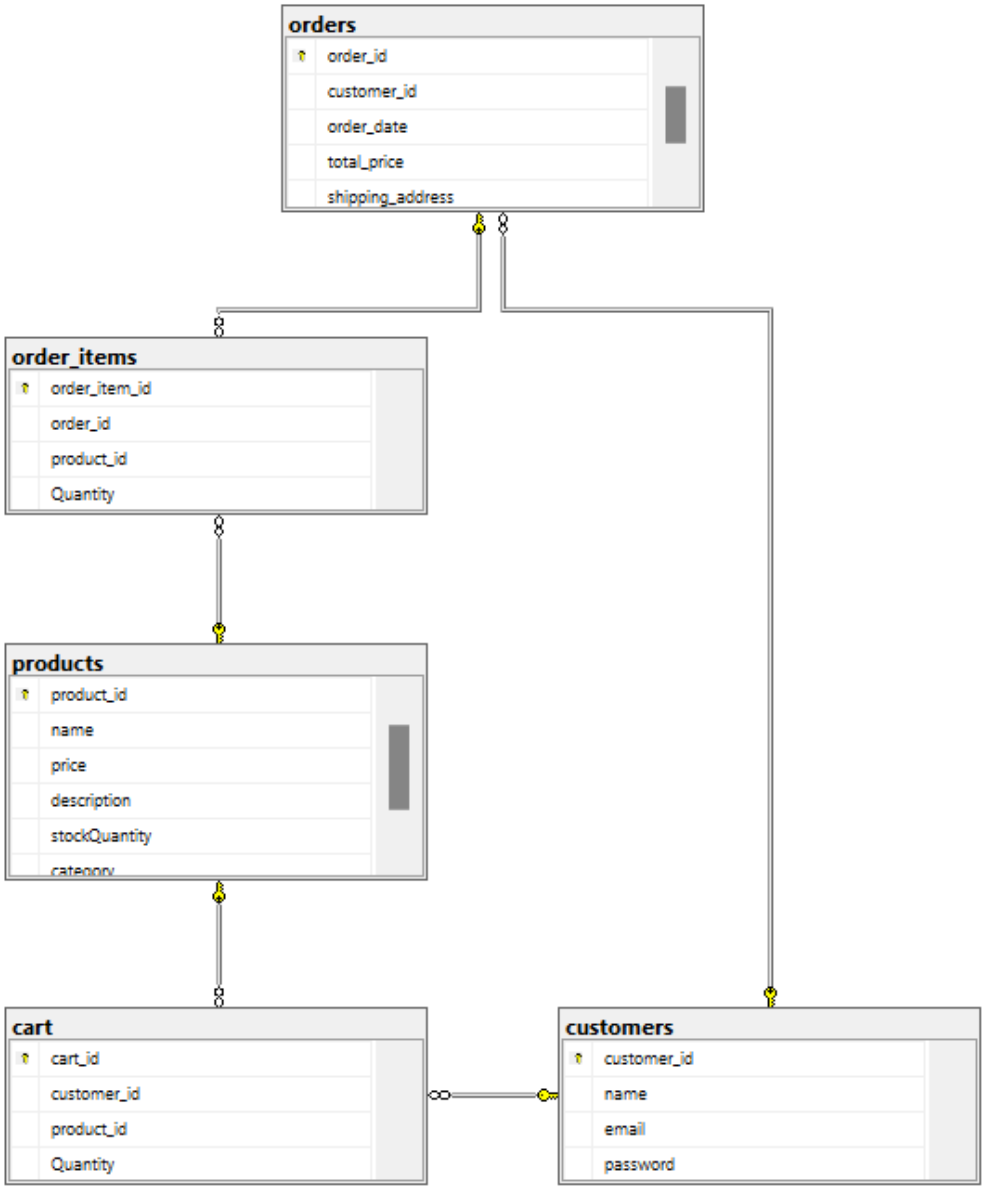
INSERT INTO order_items VALUES(407, 307, 107, 2);

INSERT INTO order_items VALUES(408, 308, 108, 1);

INSERT INTO order_items VALUES(409, 309, 109, 1);

INSERT INTO order_items VALUES(410, 310, 110, 1);

3) Create an ERD (Entity Relationship Diagram) for the database.



4) Queries :

1. Update refrigerator product price to 800.

```
update products
set price=800
where name = 'Refrigerator';
```

100 %

Results | Messages

	product_id	name	price	description	stockQuantity
1	101	Laptop	999.99	High-performance laptop	10
2	102	Smartphone	499.99	Latest smartphone model	20
3	103	Headphones	199.99	Noise-cancelling headphones	15
4	104	Smartwatch	299.99	Stylish smartwatch	25
5	105	Tablet	399.99	Lightweight tablet	30
6	106	Camera	799.99	Digital camera with 4K recording	8
7	107	Keyboard	89.99	Mechanical keyboard	50
8	108	Mouse	39.99	Wireless ergonomic mouse	45
9	109	Refrigerator	800.00	5 Star Rated	12
10	110	Printer	159.99	All-in-one printer	18
11	111	SmartTv	25000.99	Latest smart Tv	12
12	112	Samsung EarBuds	2599.99	Wireless Earbuds	18

2. Remove all cart items for a specific customer.

```
delete from cart
where customer_id=10;
```

100 %

Results | Messages

	cart_id	customer_id	product_id	Quantity
1	201	1	101	1
2	202	2	102	2
3	203	3	103	1
4	204	4	104	1
5	205	5	105	1
6	206	6	106	1
7	207	7	107	2
8	208	8	108	1
9	209	9	109	1

3. Retrieve Products Priced Below $100.

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

100 %

Results | Messages

	name
1	Keyboard
2	Mouse

4. Find Products with Stock Quantity Greater Than 5.

```
select name from products
where stockQuantity>5;
```

100 %

Results | Messages

	name
1	Laptop
2	Smartphone
3	Headphones
4	Smartwatch
5	Tablet
6	Camera
7	Keyboard
8	Mouse
9	Refrigerator
10	Printer
11	SmartTv
12	Samsung EarBuds

5. Retrieve Orders with Total Amount Between $500 and $1000.

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

100 %

Results | Messages

	order_id	customer_id	order_date	total_price	shipping_address
1	301	1	2024-09-01	999.99	5th Avenue, NewYork
2	302	2	2024-09-02	999.98	Sunset Boulevard, Chicago
3	306	6	2024-09-06	799.99	Alamo Plaza, Columbus
4	312	3	2024-09-20	799.99	Rock Street, California
5	314	5	2024-09-18	999.99	Garden House, Orlando
6	317	4	2024-09-29	799.99	Hirecy Street, California
7	318	6	2024-09-19	999.99	Jake House, Orlando

6. Find Products which name end with letter ‘r’.

```
select name from products
where name like '%r';
```

100 %

Results | Messages

	name
1	Refrigerator
2	Printer

7. Retrieve Cart Items for Customer 5.

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

100 %

Results | Messages

	cart_id	customer_id	product_id	Quantity
1	205	5	105	1

8. Find Customers Who Placed Orders in 2023.

```sql
select customers.name
from customers
join orders
on customers.customer_id = orders.customer_id
where order_date like '2023-__-__';
```

100 %

Results | Messages

name

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

```sql
ALTER TABLE products
ADD category VARCHAR(50);
UPDATE products SET category = 'Electronics'
WHERE product_id IN (101, 102, 103, 104, 105, 106, 107, 108, 110, 112);
UPDATE products SET category = 'Accessories'
WHERE product_id IN (109,111);
select * from products;
SELECT category, MIN(stockQuantity) AS min_stock_quantity
FROM products
```

100 %

Results | Messages

	category	min_stock_quantity
1	Accessories	12
2	Electronics	8

10. Calculate the Total Amount Spent by Each Customer.

```
SELECT customers.name, SUM(orders.total_price) TotalAmount
FROM customers
JOIN orders ON customers.customer_id = orders.customer_id
GROUP BY customers.name;
```

100 %

Results | Messages

	name	TotalAmount
1	David	1399.98
2	Emma	239.98
3	Jane	26000.97
4	John	1159.98
5	Laura	1799.98
6	Michael	179.98
7	Olivia	159.99
8	Robert	999.98
9	Sarah	1099.98
10	William	2849.98

11. Find the Average Order Amount for Each Customer.

```
SELECT customers.name, avg(orders.total_price) Average_Order_Amount
FROM customers
JOIN orders
ON customers.customer_id = orders.customer_id
GROUP BY customers.name;
```

100 %

Results | Messages

	name	Average_Order_Amount
1	David	699.990000
2	Emma	119.990000
3	Jane	13000.485000
4	John	579.990000
5	Laura	899.990000
6	Michael	179.980000
7	Olivia	159.990000
8	Robert	499.990000
9	Sarah	549.990000
10	William	1424.990000

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

```
SELECT customers.name, count(orders.order_id) Order_Count
FROM customers
JOIN orders
ON customers.customer_id = orders.customer_id
GROUP BY customers.name;
```

100 %

Results | Messages

	name	Order_Count
1	David	2
2	Emma	2
3	Jane	2
4	John	2
5	Laura	2
6	Michael	1
7	Olivia	1
8	Robert	2
9	Sarah	2
10	William	2

13. Find the Maximum Order Amount for Each Customer.

```
SELECT customers.name, max(orders.total_price) Max_Order_Price
FROM customers
JOIN orders
ON customers.customer_id = orders.customer_id
GROUP BY customers.name;
```

100 %

Results | Messages

	name	Max_Order_Price
1	David	999.99
2	Emma	199.99
3	Jane	25000.99
4	John	999.99
5	Laura	999.99
6	Michael	179.98
7	Olivia	159.99
8	Robert	799.99
9	Sarah	799.99
10	William	2599.99

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

```
SELECT c.customer_id, c.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.name
HAVING SUM(o.total_price) > 1000;
```

110 %

Results | Messages

	customer_id	name	total_spent
1	1	John	1159.98
2	2	Jane	26000.97
3	4	Sarah	1099.98
4	5	David	1399.98
5	6	Laura	1799.98
6	9	William	2849.98

15. Subquery to Find Products Not in the Cart.

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

110 %

Results | Messages

	product_id	name	price	description	stockQuantity	category
1	110	Printer	159.99	All-in-one printer	18	Electronics
2	111	SmartTv	25000.99	Latest smart Tv	12	Accessories
3	112	Samsung EarBuds	2599.99	Wireless Earbuds	18	Electronics

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

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

110 %

Results | Messages

customer_id	name	email	password

17. Subquery to Calculate the Percentage of Total Revenue for a Product.

```
SELECT p.product_id, p.name,
       (SUM(oi.Quantity * p.price) / (SELECT SUM(oi.Quantity * p2.price)
                                      FROM order_items oi
                                      JOIN products p2 ON oi.product_id = p2.product_id)) * 100 AS revenue_percentage
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
GROUP BY p.product_id, p.name;
```

110 %

Results | Messages

	product_id	name	revenue_percentage
1	101	Laptop	20.492000
2	102	Smartphone	20.491800
3	103	Headphones	4.098200
4	104	Smartwatch	6.147400
5	105	Tablet	8.196700
6	106	Camera	16.393600
7	107	Keyboard	3.688100
8	108	Mouse	0.819400
9	109	Refrigerator	16.393800
10	110	Printer	3.278500

18. Subquery to Find Products with Low Stock.

```
select name from products
where stockQuantity < (SELECT AVG(stockQuantity) FROM products);
```

110 %

Results | Messages

	name
1	Laptop
2	Smartphone
3	Headphones
4	Camera
5	Refrigerator
6	Printer
7	SmartTv
8	Samsung EarBuds

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

```
SELECT name FROM customers
WHERE customer_id IN (SELECT customer_id FROM orders
                      WHERE total_price > (SELECT AVG(total_price) FROM orders)
);
```

110 %

Results | Messages

	name
1	Jane
2	William