CODING CHALLENGE 1 – MYSQL [ECOMMERCE]

SWATHI BASKARAN

SQL Tables:

1. Customers table:

- •customer id (Primary Key)
- name
- email
- address

2. Products table:

- product id (Primary Key)
- name
- description
- price
- stockQuantity

3. Cart table:

- cart_id (Primary Key)
- customer id (Foreign Key)
- product_id (Foreign Key)
- quantity

4. Orders table:

- order id (Primary Key)
- customer_id (Foreign Key)
- order date
- totalAmount

5. OrderItems table:

- order item id (Primary Key)
- order id (Foreign Key)
- product_id (Foreign Key)
- quantity
- itemAmount

1. Update refrigerator product price to 800.

Query:

UPDATE Products SET Price = 800 WHERE Name = 'Refrigerator';

Output:

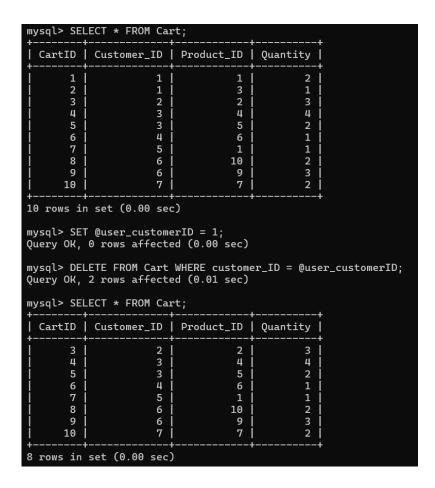
Product_id	Name	Description	Price	StockQuantity
1	Laptop	 High-performance laptop	+ 800	 10
2	Smartphone	Latest smartphone	600	15
3	Tablet	Portable tablet	300	20
4	Headphones	Noise-canceling	150	30
5	TV	4K Smart TV	900	5
6	Coffee Maker	Automatic coffee maker	50	25
7	Refrigerator	Energy-efficient	700	10
8		Countertop microwave	80	15
9	Blender	High-speed blender	70	20
10	Vacuum Cleaner	Bagless vacuum cleaner	120	10
sql> UPDATE ery OK, 1 ro ws matched:	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,		igerator	' ;
ery OK, 1 rows matched:	Products SET Prious affected (0.01 1 Changed: 1 W.	sec)	+	'; StockQuantity
sql> UPDATE ery OK, 1 ro ws matched: sql> SELECT Product_id	Products SET Prious affected (0.01 1 Changed: 1 W.	sec) arnings: 0 +	+	· •
sql> UPDATE ery OK, 1 ro ws matched: sql> SELECT Product_id	Products SET Prious affected (0.01 1 Changed: 1 Ware * FROM Products;	sec) arnings: 0 Description	+ Price +	
sql> UPDATE ery OK, 1 r ws matched: sql> SELECT	Products SET Pri w affected (0.01 1 Changed: 1 W * FROM Products; Name	sec) arnings: 0	+ Price +	 StockQuantity 10
sql> UPDATE ery OK, 1 r ws matched: sql> SELECT	Products SET Prious affected (0.01 1 Changed: 1 Wards (0.01) * FROM Products; Name Laptop Smartphone	sec) arnings: 0	+ Price + 800 600	StockQuantity
sql> UPDATE ery OK, 1 r ws matched: sql> SELECT	Products SET Privalent Products SET Privalent Products; * FROM Products; Name Laptop Smartphone Tablet Headphones TV	sec) arnings: 0	+ Price + 800 600 300	StockQuantity
sql> UPDATE ery OK, 1 r ws matched: sql> SELECT	Products SET Private Products SET Private Products; * FROM Products; Name Laptop Smartphone Tablet Headphones TV Coffee Maker	sec) arnings: 0	+ Price + 800 600 300 150 900 50	StockQuantity 10 15 20 30 5
sql> UPDATE ery OK, 1 r ws matched: sql> SELECT	Products SET Prious affected (0.01 1 Changed: 1 Winter FROM Products; Name Laptop Smartphone Tablet Headphones TV Coffee Maker Refrigerator	sec) arnings: 0	+ Price + 800 600 300 150 900 50 800	StockQuantity 10 15 20 30 5 25
sql> UPDATE ery OK, 1 r ws matched: sql> SELECT	Products SET Priow affected (0.01 1 Changed: 1 Ward Products; Name Laptop Smartphone Tablet Headphones TV Coffee Maker Refrigerator Microwave Oven	sec) arnings: 0	+ Price + 800 600 300 150 900 50 800	StockQuantity 10 15 20 30 5 25 10
rsql> UPDATE lery OK, 1 r lws matched: rsql> SELECT Product_id 1 2 3 4 5 6 7	Products SET Prious affected (0.01 1 Changed: 1 Winter FROM Products; Name Laptop Smartphone Tablet Headphones TV Coffee Maker Refrigerator	sec) arnings: 0	+ Price + 800 600 300 150 900 50 800	StockQuantity 10 15 20 30 5 25

2. Remove all cart items for a specific customer.

Query:

SET @user_customerID = 1;

DELETE FROM Cart WHERE customer_ID = @user_customerID;

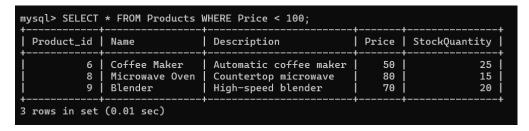


3. Retrieve products priced below \$100

Query:

SELECT * FROM Products WHERE Price < 100;

Output:



4. Find Products with Stock Quantity Greater Than 5.

Query:

SELECT * FROM Products WHERE StockQuantity > 5;

Product_id	Name	Description	Price	StockQuantity
1	Laptop	 High-performance laptop	+ 800	10
2	Smartphone	Latest smartphone	600	15
3	Tablet	Portable tablet	300	20
4	Headphones	Noise-canceling	150	30
6	Coffee Maker	Automatic coffee maker	50	25
7	Refrigerator	Energy-efficient	800	10
8	Microwave Oven	Countertop microwave	80	15
9	Blender	High-speed blender	70	20
10	Vacuum Cleaner	Bagless vacuum cleaner	120	10

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

Query:

SELECT * FROM Orders WHERE TotalAmount BETWEEN 500 AND 1000;

Output:

6. Find Products which name end with letter 'r'.

Query:

SELECT * FROM Products WHERE Name LIKE '%r';

Output:

mysql> SELECT * FROM Products WHERE Name LIKE '%r';						
Product_id	Name	Description	Price	StockQuantity		
j 7 j j 9 j	Refrigerator Blender	Automatic coffee maker Energy-efficient High-speed blender Bagless vacuum cleaner	800 70	25 10 20 10		
4 rows in set (0.00 sec)						

7. Retrieve Cart Items for Customer 5.

Query:

SELECT * FROM Cart WHERE Customer ID = 5;

```
mysql> SELECT * FROM Cart WHERE Customer_ID = 5;
+-----+
| CartID | Customer_ID | Product_ID | Quantity |
+-----+
| 7 | 5 | 1 | 1 |
+-----+
1 row in set (0.01 sec)
```

8. Find Customers Who Placed Orders in 2023.

Query:

SELECT

Cus.*

FROM Customers Cus

RIGHT JOIN Orders Ord

ON Cus.Customer ID = Ord.Customer ID

WHERE YEAR(Ord.Order Date) = 2023;

Output:

```
mysql> -- Finding customers who placed orders in 2023
mysql> SELECT
     -> Cus.*
    -> FROM Customers Cus
    -> RIGHT JOIN Orders Ord
    -> ON Cus.Customer_ID = Ord.Customer_ID
-> WHERE YEAR(Ord.Order_Date) = 2023;
  customer_id
                                        email
                                                                     Address
                   name
                   John Doe
                                        johndoe@example.com
                                                                     123 Main St, City
                                                                     456 Elm St,
              2
3
                   Jane Smith
                                        janesmith@example.com
                                                                                   Town
                                                                     789 Oak St, Village
101 Pine St, Suburb
                   Robert Johnson
                                        robert@example.com
                   Sarah Brown
                                        sarah@example.com
              5
                                        david@example.com
                   David Lee
                                                                     234 Cedar St, District
                   Laura Hall
                                        laura@example.com
michael@example.com
                                                                     567 Birch St,
                                                                                      County
                                                                     890 Maple St, State
321 Redwood St, Country
432 Spruce St, Province
                   Michael Davis
              8
                   Emma Wilson
                                        emma@example.com
                   William Taylor
                                        william@example.com
                   Olivia Adams
                                                                     765 Fir St, Territory
             10
                                        olivia@example.com
10 rows in set (0.01 sec)
```

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

Query:

-- Initially, adding a column in Products Table called category and categorizing the products as "Electronics" and "Appliances" accordingly.

ALTER TABLE Products ADD COLUMN Category VARCHAR(50);

UPDATE products SET category = 'Electronics' WHERE product_ID IN (1, 2, 3, 4, 5);

UPDATE products SET category = 'Appliances' WHERE product_ID IN (6, 7, 8, 9, 10);

--Now finding the minimum stock quantity for each product category

SELECT Category, MIN(StockQuantity) AS "Mininum Stock Quantity" FROM Products GROUP BY Category;

Output:

```
mysql> ALTER TABLE Products ADD COLUMN Category VARCHAR(50);
Query OK, 0 rows affected (0.03 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> UPDATE products SET category = 'Electronics' WHERE product_ID IN (1, 2, 3, 4, 5); Query OK, 5 rows affected (0.01 sec)
Rows matched: 5 Changed: 5 Warnings: 0

mysql> UPDATE products SET category = 'Appliances' WHERE product_ID IN (6, 7, 8, 9, 10); Query OK, 5 rows affected (0.01 sec)
Rows matched: 5 Changed: 5 Warnings: 0
```

Product_id	Name	Description	Price	StockQuantity	Category
 1	 Laptop	High-performance laptop	800	10	Electronics
2	Smartphone	Latest smartphone	600	15	Electronics
3	Tablet	Portable tablet	300	20	Electronics
4	Headphones	Noise-canceling	150	30	Electronics
5	TV	4K Smart TV	900	5	Electronics
6	Coffee Maker	Automatic coffee maker	50	25	Appliances
7	Refrigerator	Energy-efficient	800	10	Appliances
8	Microwave Oven	Countertop microwave	80	15	Appliances
9	Blender	High-speed blender	70	20	Appliances
10	Vacuum Cleaner	Bagless vacuum cleaner	120	10	Appliances
.0 rows in set (0.00 sec) nysql> SELECT Category, MIN(StockQuantity) AS "Mininum Stock Quantity" FROM Products GROUP BY Cate 					
Electronics 5 Appliances 10					

10. Calculate the Total Amount Spent by Each Customer.

Query:

SELECT

Cus.Name,

Cus.Customer ID,

IFNULL(ROUND(SUM(Ord.TotalAmount),2),0) AS "Total Amount Spent"

FROM Customers Cus

LEFT JOIN Orders Ord

ON Cus.Customer_ID = Ord.Customer_ID

GROUP BY Customer ID;

Output:

```
mysql> -- Calculating the total amount spent by each customer
mysql> SELECT
    -> Cus.Name,
    -> Cus.Customer_ID,
-> IFNULL(ROUND(SUM(Ord.TotalAmount),2),0) AS "Total Amount Spent"
    -> FROM Customers Cus
    -> LEFT JOIN Orders Ord
    -> ON Cus.Customer_ID = Ord.Customer_ID
    -> GROUP BY Customer_ID;
  Name
                     Customer_ID
                                   | Total Amount Spent
  John Doe
                                                     1200
                                2
3
4
  Jane Smith
                                                      900
  Robert Johnson
                                                      300
                                                      150
  Sarah Brown
  David Lee
                                5
                                                     1800
  Laura Hall
                                6
                                                      400
                                7
  Michael Davis
                                                      700
  Emma Wilson
William Taylor
                                8
                                                     160
                                9
                                                      140
  Olivia Adams
                               10
                                                    1400
10 rows in set (0.00 sec)
```

11. Find the Average Order Amount for Each Customer.

Query:

SELECT

Cus.Name,

Cus.Customer ID,

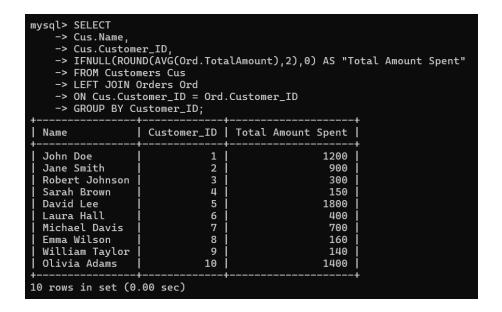
IFNULL(ROUND(AVG(Ord.TotalAmount),2),0) AS "Total Amount Spent"

FROM Customers Cus

LEFT JOIN Orders Ord

ON Cus.Customer ID = Ord.Customer ID

GROUP BY Customer ID;



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

Query:

SELECT

Cus.Name,

Cus.Customer ID,

COUNT(Order ID) AS "Number of Orders Placed"

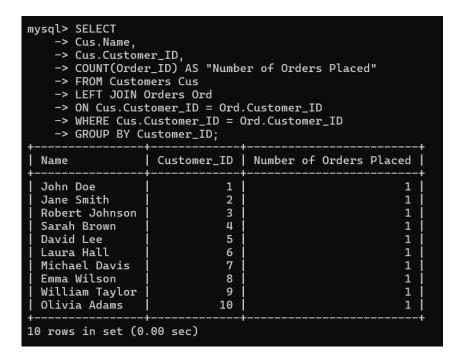
FROM Customers Cus

LEFT JOIN Orders Ord

ON Cus.Customer ID = Ord.Customer ID

WHERE Cus.Customer_ID = Ord.Customer_ID

GROUP BY Customer ID;



13. Find the Maximum Order Amount for Each Customer.

Query:

SELECT

Cus.Name,

Cus.Customer_ID,

IFNULL(MAX(Ord.TotalAmount),0) AS "Maximum Order Amount"

FROM Customers Cus

LEFT JOIN Orders Ord

ON Cus.Customer ID = Ord.Customer ID

GROUP BY Customer ID;

```
mysql> -- Maximum order amount for each customer
mysql> SELECT
    -> Cus.Name,
    -> Cus.Customer_ID,
-> IFNULL(MAX(Ord.TotalAmount),0) AS "Maximum Order Amount"
    -> FROM Customers Cus
    -> LEFT JOIN Orders Ord
    -> ON Cus.Customer_ID = Ord.Customer_ID
-> GROUP BY Customer_ID;
                      Customer_ID
                                       Maximum Order Amount
  Name
  John Doe
                                  1
2
4
5
  Jane Smith
                                                            900
                                                            300
  Robert Johnson
  Sarah Brown
                                                           150
  David Lee
                                                           1800
  Laura Hall
                                  6
7
8
                                                            400
  Michael Davis
                                                            700
  Emma Wilson
William Taylor
                                                            160
                                  9
                                                            140
  Olivia Adams
                                 10
                                                          1400
10 rows in set (0.00 sec)
```

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

Query:

SELECT

Cus.Name,

Cus.Customer ID,

IFNULL(ROUND(SUM(Ord.TotalAmount),2),0) AS "Total Order Amount"

FROM Customers Cus

LEFT JOIN Orders Ord

ON Cus.Customer_ID = Ord.Customer_ID

GROUP BY Cus.Customer ID

HAVING IFNULL(ROUND(SUM(Ord.TotalAmount),2),0) > 1000;

```
mysql> SELECT
    -> Cus.Name,
    -> Cus.Customer_ID,
    -> IFNULL(ROUND(SUM(Ord.TotalAmount),2),0) AS "Total Order Amount"
    -> FROM Customers Cus
    -> LEFT JOIN Orders Ord
    -> ON Cus.Customer_ID = Ord.Customer_ID
-> GROUP BY Cus.Customer_ID
    -> HAVING IFNULL(ROUND(SUM(Ord.TotalAmount),2),0) > 1000
 Name
                  Customer_ID
                                 Total Order Amount
  John Doe
                                                 1200
 David Lee
                             5
                                                 1800
 Olivia Adams
                            10
                                                 1400
 rows in set (0.00 sec)
```

15. Subquery to Find Products Not in the Cart.

```
Query:
```

```
SELECT *
FROM Products
WHERE Product_ID NOT IN
(
SELECT Product_ID
FROM CART
);
```

Output:

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

Query:

SELECT *

FROM Customers

WHERE Customer ID NOT IN

```
SELECT Customer_ID
FROM Orders
);
```

Output:

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

Query:

SELECT

P.Product_ID,

P.Name,

IFNULL(ROUND(SUM(Oi.ItemAmount),2),0) AS "Product Revenue",

ROUND(IFNULL(SUM(Oi.ItemAmount),0) /

(SELECT SUM(ItemAmount) FROM OrderItems) * 100,2)

AS "Revenue Percentage"

FROM Products P

LEFT JOIN OrderItems Oi

ON P.Product ID = Oi.Product ID

GROUP BY P.Product_ID, P.Name;

```
mysql> SELECT
     -> P.Product_ID,
    -> P.Name,
-> IFNULL(ROUND(SUM(Oi.ItemAmount),2),0) AS "Product Revenue",
-> ROUND(IFNULL(SUM(Oi.ItemAmount),0) /
     -> (SELECT SUM(ItemAmount) FROM OrderItems) * 100,2)
-> AS "Revenue Percentage"
     -> FROM Products P
    -> LEFT JOIN OrderItems Oi
-> ON P.Product_ID = Oi.Product_ID
-> GROUP BY P.Product_ID, P.Name;
  Product_ID | Name
                                           Product Revenue
                                                                   Revenue Percentage
                                                                                     27.91
34.88
                    Laptop
                                                          2400
                    Smartphone
                                                          3000
                    .
Tablet
                                                                                      3.49
                                                           300
                   Headphones
                                                           600
                                                                                      6.98
                                                                                     20.93
              5
                                                          1800
                    TV
                    Coffee Maker
                                                                                      0.58
              6
                                                             50
                    Refrigerator
                                                             0
                                                                                      0.00
                                                              0
              8
                    Microwave Oven
                                                                                      0.00
                    Blender
                                                           210
                                                                                      2.44
                    Vacuum Cleaner
             10
                                                           240
                                                                                      2.79
10 rows in set (0.00 sec)
```

18. Subquery to Find Products with Low Stock.

Query:

```
SELECT * FROM Products

WHERE StockQuantity <
(
SELECT AVG(StockQuantity)

FROM Products
);
```

mysql> SELECT * FROM Products WHERE StockQuantity < (SELECT AVG(StockQuantity) FROM Products);						
Product_id	Name	Description	Price	StockQuantity	Category	
1 1 2 5 7 8 8 10	Microwave Oven	High-performance laptop Latest smartphone 4K Smart TV Energy-efficient Countertop microwave Bagless vacuum cleaner	800 600 900 800 80	15 5 10	Electronics Electronics Electronics Appliances Appliances Appliances	
6 rows in set	++ 6 rows in set (0.00 sec)					

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

Query:

```
SELECT * FROM Customers

WHERE Customer_ID IN

(
    SELECT Customer_ID
    FROM Orders
    WHERE TotalAmount =
    (
        SELECT MAX(TotalAmount)
        FROM Orders
    )
);
```