Assignment 1 – TechShop, an electronic gadgets shop J724 – Pavithra Balaji

Task 2 – Select, Where, Between, AND, LIKE:

1. Write an SQL query to retrieve the names and emails of all customers.

SELECT concat(first_name,' ',last_name) as Name,email from Customers; [I combined first and last names as a 'name'].

2. Write an SQL query to list all orders with their order dates and corresponding customer names.

```
SELECT o.order_id, o.order_date,
CONCAT(c.first_name, ' ', c.last_name) AS customer_name
FROM Orders o
 JOIN Customers c ON o.customer_id = c.customer_id;
```

3. Write an SQL query to insert a new customer record into the "Customers" table. Include customer information such as name, email, and address.

```
INSERT INTO Customers (first_name, last_name,email, phone, address) VALUES ('Michael', 'Johnson',
'michael.johnson@example.com', '9876543210', '123 Main Street, NY');
```

4. Write an SQL query to update the prices of all electronic gadgets in the "Products" table by increasing them by 10%.

```
UPDATE Products SET price = price * 1.10;
```

5. Write an SQL query to delete a specific order and its associated order details from the "Orders" and "OrderDetails" tables. Allow users to input the order ID as a parameter.

```
SET @order_id = 5;
DELETE FROM OrderDetails WHERE order_id = @order_id;
DELETE FROM Orders WHERE order_id = @order_id;
```

6. Write an SQL query to insert a new order into the "Orders" table. Include the customer ID, order date, and any other necessary information.

```
INSERT INTO Orders (customer_id, order_date, total_amount) VALUES (3, NOW(), 25000);
```

7. Write an SQL query to update the contact information (e.g., email and address) of a specific customer in the "Customers" table. Allow users to input the customer ID and new contact information.

```
UPDATE Customers

SET email = 'newemail@example.com',   address = 'NewZIP City'

WHERE customer_id = 5;
```

8. Write an SQL query to recalculate and update the total cost of each order in the "Orders" table based on the prices and quantities in the "OrderDetails" table.

```
UPDATE Orders o

JOIN (

   SELECT od.order_id, SUM(p.price * od.quantity) AS new_total

   FROM OrderDetails od

   JOIN Products p ON od.product_id = p.product_id

   GROUP BY od.order_id

) AS order_totals

ON o.order_id = order_totals.order_id

SET o.total_amount = order_totals.new_total;
```

9. Write an SQL query to delete all orders and their associated order details for a specific customer from the "Orders" and "OrderDetails" tables. Allow users to input the customer ID as a parameter.

```
SET @CustomerID = 5;

DELETE FROM OrderDetails

WHERE order_id IN (

   SELECT order_id FROM Orders WHERE customer_id = @CustomerID

);

DELETE FROM Orders

WHERE customer_id = @CustomerID;
```

10. Write an SQL query to insert a new electronic gadget product into the "Products" table, including product name, category, price, and any other relevant details.

```
INSERT INTO Products (product_name, description, price) VALUES ('OnePlus 12', '5G smartphone
with 512GB storage and 16GB RAM',89999);
```

11. Write an SQL query to update the status of a specific order in the "Orders" table (e.g., from "Pending" to "Shipped"). Allow users to input the order ID and the new status.

ALTER TABLE Orders ADD COLUMN status VARCHAR(20) DEFAULT 'Pending'; [I have just added the status column]

```
UPDATE Orders

SET status = 'Shipped'

WHERE order_id = 1;
```

12. Write an SQL query to calculate and update the number of orders placed by each customer in the "Customers" table based on the data in the "Orders" table.

ALTER TABLE Customers ADD COLUMN order_count INT DEFAULT 0; [I have added the order count column]

```
UPDATE Customers c

SET order_count = (

   SELECT COUNT(*)

   FROM Orders o

   WHERE o.customer_id = c.customer_id

);
```