MY SQL - DATA QUERY LANGUAGE

```
-- 1. List all customers who have made purchases of more than $80.

select distinct o.user_id, u.user_name

from Orders o

join Users u ON o.user_id = u.user_id

        where total_amount>40;


-- the above will verify individual purchases whereas the below take the consolidated values

SELECT u.user_id,u.user_name, SUM(o.total_amount) AS total_spent

FROM Users u

JOIN Orders o ON u.user_id = o.user_id

GROUP BY u.user_id, u.user_name

HAVING total_spent > 80;


-- 2. Retrieve all orders placed in the last 611 days along with the customer name and email

SELECT o.order_id, u.user_name, u.email

FROM Users u

JOIN Orders o ON u.user_id = o.user_id

WHERE o.order_date >= CURRENT_DATE - INTERVAL 611 DAY;


-- 3. Find the average product price for each category.

SELECT

  category,

  ROUND(AVG(price), 2) AS average_price

FROM Products

GROUP BY category;
```

```
-- 4. List all customers who have purchased a product from the category Electronics
SELECT
		DISTINCT u.user_id, u.user_name
   FROM Users u
   JOIN Orders o ON u.user_id = o.user_id
   JOIN OrderDetails od ON o.order_id = od.order_id
   JOIN Products p ON od.product_id = p.product_id
   WHERE p.category = 'Electronics';

-- Faster and professional way to list out customers
SELECT user_id, user_name
FROM Users u
WHERE EXISTS (
   SELECT 1
   FROM Orders o
   JOIN OrderDetails od ON o.order_id = od.order_id
   JOIN Products p ON od.product_id = p.product_id
   WHERE o.user_id = u.user_id
   AND p.category = 'Electronics'
);

-- 5. Find the total number of products sold and the total revenue generated for each product.
select od.product_id, p.product_name, sum(od.quantity), sum(o.total_amount)
FROM OrderDetails od
JOIN products p ON od.product_id = p.product_id
JOIN Orders o ON od.order_id = o.order_id
```

```sql
GROUP BY od.product_id, p.product_name;


-- when a single order multiple products, the above total_amount will be wrong.
SELECT
    p.product_id,
    p.product_name,
    SUM(od.quantity) AS total_units_sold,
    SUM(od.quantity * p.price) AS total_revenue
FROM OrderDetails od
JOIN Products p ON od.product_id = p.product_id
GROUP BY p.product_id, p.product_name;


-- However, here for each order only one product is available and the product price is
not matching, the first one will be perfect.alter


-- 6. Update the price of all products in the Books category, increasing it by 10%.
SET SQL_SAFE_UPDATES = 0;


UPDATE Products
SET price = ROUND(price * 1.10, 2)
WHERE category = 'Books';


SET SQL_SAFE_UPDATES = 1;


select * from orderDetails;


-- 7. Remove all orders that were placed before 2020.
INSERT INTO Orders (user_id, order_date, total_amount) VALUES
```

```
(1, '2019-05-01', 79.98),
(2, '2019-05-03', 129.99);
INSERT INTO OrderDetails (order_id, product_id, quantity) VALUES
(5, 1, 2),
(6, 2, 1);

SELECT * FROM Orders
WHERE order_date < '2021-01-01';

SET SQL_SAFE_UPDATES = 0;

DELETE FROM Orders
WHERE order_date < '2019-05-02';

SET SQL_SAFE_UPDATES = 1;

-- 8. Write a query to fetch the order details, including customer name, product name,
and
-- quantity, for orders placed on 2024-05-01.

SELECT u.user_name, o.order_id, p.product_name, od.quantity
FROM Users u
JOIN Orders o ON u.user_id = o.user_id
JOIN OrderDetails od ON o.order_id = od.order_id
JOIN Products p ON od.product_id = p.product_id
WHERE o.order_date = '2024-05-01';

-- 9. Fetch all customers and the total number of orders they have placed.
```

```
SELECT u.user_id, u.user_name, COUNT(o.order_id) AS 'No.of orders'

FROM Users u

LEFT JOIN Orders o ON u.user_id = o.user_id

GROUP BY u.user_id, u.user_name;


-- 10. Retrieve the average rating for all products in the Electronics category

select * from Products;


-- 11. List all customers who purchased more than 1 units of any product, including the
product

-- name and total quantity purchased.

SELECT u.user_name, p.product_name, sum(od.quantity) AS Total_quantity

FROM Users u

INNER JOIN Orders o ON u.user_id = o.user_id

INNER JOIN OrderDetails od ON o.order_id = od.order_id

INNER JOIN Products p ON od.product_id = p.product_id

GROUP BY u.user_id, u.user_name, p.product_name

HAVING Total_quantity > 1


-- 12. Find the total revenue generated by each category along with the category name


SELECT p.category, sum(o.total_amount) AS total_revenue

FROM Products p

LEFT JOIN OrderDetails od ON p.product_id = od.product_id

LEFT JOIN Orders o ON od.order_id = o.order_id

GROUP BY p.category;
```