**ASSIGNMENT 5**

**TASK 1**

1. **Write a query to retrieve all records from the Customers table**

SYNTAX:

SELECT \* FROM Customers;

1. **Write a query to retrieve the names and email addresses of customers whose names start with 'J'**

SYNTAX:

SELECT \* FROM Customers WHERE NAME LIKE 'J%';

1. **Write a query to retrieve the order details (OrderID, ProductName, Quantity) for all orders.**

SYNTAX:

SELECT OrderID, ProductName, Quantity FROM ORDERS;

1. **Write a query to calculate the total quantity of products ordered**

SYNTAX:

SELECT SUM(QUANTITY) AS TOTAL\_QUANTITY FROM Orders;

1. **Write a query to retrieve the names of customers who have placed an order**

SYNTAX:

SELECT C.NAME,O.PRODUCTNAME FROM Customers C JOIN Orders O ON C.CustomerID=O.CustomerID;

1. **Write a query to retrieve the products with a price greater than $10.00**

SYNTAX:

SELECT \* FROM PRODUCTS WHERE Price>10;

1. **Write a query to retrieve the customer name and order date for all orders placed on or after '2023-07-05'**

SYNTAX:

SELECT C.NAME, O.ORDERDATE FROM Customers C JOIN Orders O ON C.CustomerID=O.CustomerID WHERE O.OrderDate >='2023-07-05';

1. **Write a query to calculate the average price of all products**

SYNTAX:

SELECT AVG(PRICE) AS AVG\_PRICE FROM PRODUCTS;

1. **Write a query to retrieve the customer names along with the total quantity of products they have ordered**

SYNTAX:

SELECT C.NAME, O.QUANTITY FROM Customers C JOIN Orders O ON C.CustomerID=O.CustomerID;

1. **Write a query to retrieve the products that have not been ordered**

SYNTAX:

SELECT P.ProductName FROM PRODUCTS P WHERE NOT EXISTS (SELECT O.ProductName FROM Orders O WHERE O.ProductName=P.ProductName);

**TASK 2**

1. **Write a query to retrieve the top 5 customers who have placed the highest total quantity of orders**

SYNTAX:

SELECT TOP 5 C.NAME,O.Quantity,DENSE\_RANK()OVER(ORDER BY O.Quantity DESC) AS QUANITY\_RANK FROM Customers C JOIN Orders O ON C.CustomerID=O.CustomerID;

1. **Write a query to calculate the average price of products for each product category.**

SYNTAX:

SELECT ROUND(AVG(O.QUANTITY \* P.PRICE),2) AS AVG\_PRICE, O.ProductName FROM Orders AS O JOIN Products AS P ON O.ProductName=P.ProductName GROUP BY O.ProductName;

1. **Write a query to retrieve the customers who have not placed any orders.**

SYNTAX:

SELECT C.NAME, C.CustomerID FROM CUSTOMERS C WHERE NOT EXISTS (SELECT O.CustomerID FROM Orders O WHERE C.CustomerID = O.CustomerID);

1. **Write a query to retrieve the order details (OrderID, ProductName, Quantity) for orders placed by customers whose names start with 'M'.**

SYNTAX:

SELECT C.Name,O.ORDERID, P.PRODUCTNAME,O.QUANTITY FROM Customers C JOIN Orders O ON C.CustomerID=O.CustomerID JOIN Products P ON O.ProductName=P.ProductName WHERE C.Name LIKE 'M%';

1. **Write a query to calculate the total revenue generated from all orders.**

SYNTAX:

SELECT SUM(SUM\_PRICE) AS TOTAL\_REVENUE FROM (SELECT SUM(O.QUANTITY \* P.PRICE) AS SUM\_PRICE, O.ProductName FROM Orders AS O JOIN Products AS P ON O.ProductName=P.ProductName GROUP BY O.ProductName) AS INNER\_QUERY;

1. **Write a query to retrieve the customer names along with the total revenue generated from their orders.**

SYNTAX:

SELECT SUM(O.QUANTITY \* P.PRICE) AS REV\_GEN, C.NAME FROM CUSTOMERS C JOIN ORDERS O ON C.CUSTOMERID=O.CUSTOMERID JOIN PRODUCTS P ON O.PRODUCTNAME=P.PRODUCTNAME GROUP BY C.Name;

1. **Write a query to retrieve the customers who have placed at least one order for each product category.**

SYNTAX:

SELECT O.CustomerID FROM Customers C JOIN Orders O ON C.CustomerID=O.CustomerID GROUP BY O.ProductName,O.CustomerID HAVING COUNT(O.CUSTOMERID)>=1;

1. **Write a query to retrieve the customers who have placed orders on consecutive days.**

SYNTAX:

SELECT O1.CUSTOMERID FROM Orders O1, ORDERS O2 WHERE O1.CustomerID=O2.CustomerID AND ABS(DATEDIFF(DAY,O1.OrderDate,O2.OrderDate))=1;

1. **Write a query to retrieve the top 3 products with the highest average quantity ordered.**

SYNTAX:

SELECT TOP 3 AVG(O1.QUANTITY) AS AVG\_QTY, O1.ProductName FROM Orders O1 JOIN ORDERS O2 ON O1.OrderID=O2.OrderID GROUP BY O1.ProductName;

1. **Write a query to calculate the percentage of orders that have a quantity greater than the average quantity.**

SYNTAX:

SELECT (Quantity\*100)/(SELECT SUM(QUANTITY) FROM Orders) AS PERCENTAGE\_QTY FROM ORDERS O1 WHERE Quantity>(SELECT AVG(Quantity) AS AVG\_QUANTITY FROM Orders O2 WHERE O1.OrderID=O2.OrderID);