1. Retrieve the order date and day of the week for all orders.

Retrieve the order date and day of the week for all orders

```
SELECT orderDate , DAYOFWEEK(orderDate)
FROM orders;
```

2.List the product names and order dates for products ordered on a Saturday

```
select p.productName
from orders as o left join orderdetails as od using(orderNumber) left join
products as p using(productCode)
where DayName(o.orderDate) like "Saturday"
```

3. Find the number of orders placed on each day of the week

```
select count(*) , DayName(orderDate)
from orders
group by DayName(orderDate)
```

4. Retrieve the customer names and their first order date.

```
select c.customerName, Min(o.orderDate)
from customers as c join orders as o using(customerNumber)
group by c.customerName
```

5. Calculate the total payments received for each customer. Include the customer name and the total payments.

```
select c.customerName , sum(amount) as total_payments
```

```
from customers as c join payments as pa using(customerNumber)
group by c.customerName
```

6. Retrieve the count of orders for each year, and include a grand total count. Display the year and the corresponding order count.

```
select year(orderDate) , count(*)
from orders
group by year(orderDate)
with rollup
```

1. For each year and month, find the total number of orders placed. Additionally, provide a grand total for all orders. Display the results with the count of orders, year, and month.

```
SELECT
   YEAR(orderDate) AS OrderYear,
   MONTH(orderDate) AS OrderMonth,
   COUNT(*) AS OrderCount
FROM orders
GROUP BY OrderYear, OrderMonth WITH ROLLUP;
```

7. Retrieve the total value of products in stock, considering the quantity in stock and the price each. Display the product name and the corresponding total value. Additionally, include a grand total row that represents the overall total value of all products.

```
SELECT productName, sum(quantityInStock * buyPrice) as total
FROM products
GROUP BY productName
WITH ROLLUP;
```

8. Retrieve the products with a total value exceeding $15M. Display the product name and the corresponding total value. Additionally, include a grand total row that represents the overall total value of all products.

```
SELECT productName, sum(quantityInStock * buyPrice) as total
FROM products
GROUP BY productName
WITH ROLLUP
HAVING total > 15000000;
```

9. Retrieve the total quantity of products sold and the total sales amount for each country. Display the country, the total quantity of products sold, and the total sales amount ((quantityOrdered * priceEach)) . Include only countries where the total quantity sold is greater than 2500. Sort the results by the total sales amount in ascending order.

```
SELECT
   country,
   SUM(quantityOrdered) AS total_sold,
   SUM(quantityOrdered * priceEach) AS total_sales
FROM orders
JOIN customers ON orders.customerNumber = customers.customerNumber
JOIN orderdetails ON orders.orderNumber = orderdetails.orderNumber
GROUP BY country
HAVING total_sold > 2500
ORDER BY total_sales ASC;
```

10.Retrieve the number of products in each product lines their text descriptions. Display the product line, the number of products in each line, and the text description. Include only those product lines where the count of products is greater than 10.

```
SELECT
   productlines.productLine,
   COUNT(products.productCode) AS product_count,
   productlines.textDescription
FROM productlines
JOIN products ON productlines.productLine = products.productLine
GROUP BY productlines.productLine
HAVING product_count > 10;
```

11. Retrieve using JOIN the last name and first name of employees working in offices located in the USA.

```
SELECT e.firstname,e.lastname
FROM employees e
JOIN offices o ON e.officecode = e.officecode
HAVING o.country = 'usa';
```

12. Retrieve using Subquerry the last name and first name of employees working in offices located in the USA.

```
SELECT e.firstname,e.lastname
FROM employees e
WHERE officeCode IN (SELECT officeCode FROM offices WHERE country = 'USA');
```

13. Retrieve the customer numbers and payment amounts for customers whose payment amount is below the average payment amount, using a subquery.

```
SELECT customerNumber, amount
FROM payments
WHERE amount < (SELECT AVG(amount) FROM payments);
```

14.Retrieve the count, customer name, and customer number for customers who have not placed any orders. Include a grand total row that represents the overall count. (use subquery)

```
SELECT c.customerNumber, c.customerName, COUNT(orderNumber) AS orderCount
FROM customers c
LEFT JOIN orders ON c.customerNumber = orders.customerNumber
GROUP BY c.customerNumber, c.customerName
WITH ROLLUP;
```

15.Write a SQL query to retrieve customer numbers, names, total sales, and purchase categories from a retail database. The purchase category should be labeled as 'High Value' if the total sales for a customer exceed $100,000, and 'Regular Value' otherwise. Use the tables customers and payments, and include necessary aliases.

```
SELECT customerNumber, customerName, totalSales,
   CASE
      WHEN totalSales > 100000 THEN 'High Value'
      ELSE 'Regular Value'
   END AS purchaseCategory
FROM (SELECT c.customerNumber, c.customerName,
      COALESCE(SUM(p.amount), 0) AS totalSales
```

```
    FROM customers c
    LEFT JOIN payments p ON c.customerNumber = p.customerNumber
    GROUP BY c.customerNumber, c.customerName
) AS customer_sales;
```

16.List the employees and their respective managers employee name as "EmployeeName" and the manager name as "ManagerName".

```
SELECT CONCAT(e1.firstName, ' ', e1.lastName) AS EmployeeName,
      CONCAT(e2.firstName, ' ', e2.lastName) AS ManagerName
FROM employees e1
JOIN employees e2 ON e1.reportsTo = e2.employeeNumber;
```

17.List the employees and their respective managers who have the same job title. Display the employee name as "EmployeeName" and the manager name as "ManagerName".

```
SELECT CONCAT(e1.firstName, ' ', e1.lastName) AS EmployeeName,
      CONCAT(e2.firstName, ' ', e2.lastName) AS ManagerName
FROM employees e1
JOIN employees e2 ON e1.reportsTo = e2.employeeNumber
where e1.jobtitle=e2.jobtitle
```

18.List the employees and their respective managers employee name as "EmployeeName" and the manager name as "ManagerName". Show all the employees even if they don’t have a manager.

```
SELECT CONCAT(e1.firstName, ' ', e1.lastName) AS EmployeeName,
            COALESCE(e2.firstName, 'no manager') AS ManagerName
```

```
FROM employees e1
JOIN employees e2 ON e1.reportsTo = e2.employeeNumber
```

19.List the employees and their respective managers employee name as "EmployeeName" and the manager name as "ManagerName". Show all the employees even if they don’t have a manager.

```
SELECT CONCAT(e1.firstName, ' ', e1.lastName) AS EmployeeName,
        COALESCE(e2.firstName, 'no manager') AS ManagerName
FROM employees e1
JOIN employees e2 ON e1.reportsTo = e2.employeeNumber
```

20.Find the names of all customers who have placed at least one order. Use EXISTS

```
SELECT customerName
FROM customers c
WHERE EXISTS (
   SELECT 1
   FROM orders o
   WHERE o.customerNumber = c.customerNumber
 );
```

21.Retrieve the product names that have been ordered in the 2004 year. Use EXISTS

```
SELECT DISTINCT productName
FROM products p
WHERE EXISTS (
   SELECT 1
```

```
  FROM orderDetails od
    JOIN orders o ON od.orderNumber = o.orderNumber
  WHERE
    od.productCode = p.productCode
    AND YEAR(o.orderDate) = 2004
);
```