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

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

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

```
SELECT productName, orderDate , DAYNAME(orderDate)
FROM products
JOIN orderdetails ON products.productCode = orderdetails.productCode
JOIN orders ON orderdetails.orderNumber = orders.orderNumber
WHERE DAYNAME(orderDate) = 'Saturday';
```

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

```
SELECT DAYNAME(orderDate) AS dayOfWeek, COUNT(*) AS orderCount
FROM orders
GROUP BY dayOfWeek;
```

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

```
SELECT customerName, MIN(orderDate) AS firstOrderDate
FROM customers
JOIN orders ON customers.customerNumber = orders.customerNumber
GROUP BY customerName;
```

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

```
SELECT
    c.customerName,
    SUM(p.amount) AS total_payments
FROM
    customers c
JOIN
    payments p ON c.customerNumber = p.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 count(*), year(orderDate)
 FROM `orders`
group BY year(orderDate)
WITH ROLLUP

=====================================================

SELECT
    IFNULL(YEAR(orderDate), 'Grand Total') AS OrderYear,
    COUNT(*) AS OrderCount
FROM
    orders
GROUP BY
    YEAR(orderDate) WITH ROLLUP
;
```

7. 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 count(*), year(orderDate) , month(orderDate)
 FROM `orders`
group BY year(orderDate) , month(orderDate)
WITH ROLLUP;
```

```

```

8. 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
    p.productName,
    SUM(p.quantityInStock * od.priceEach) AS totalValue
FROM
    products p
JOIN
    orderdetails od ON p.productCode = od.productCode
GROUP BY
    p.productName WITH ROLLUP;
```

9. 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 p.productName,
  SUM(p.quantityInStock * od.priceEach) AS totalValue
FROM products p
JOIN orderdetails od ON p.productCode = od.productCode
GROUP BY p.productName WITH ROLLUP
HAVING totalValue > 15000000;
```

10. 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
    C.country,
```

```
    SUM(od.quantityOrdered) AS totalQuantitySold,
    SUM(od.quantityOrdered * od.priceEach) AS totalSalesAmount
FROM customers c
JOIN orders o ON c.customerNumber = o.customerNumber
JOIN orderdetails od ON o.orderNumber = od.orderNumber
GROUP BY c.country
HAVING totalquantitysold > 2500
ORDER BY totalsalesamount
```

11. 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
    p.productLine,
    COUNT(*) AS productCount,
    pl.textDescription
FROM
    products p
JOIN
    productlines pl USING(productLine)
GROUP BY
    p.productLine, pl.textDescription
HAVING
    productCount > 10;
```

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

```
SELECT
    e.lastName, e.firstName
FROM
    employees e
JOIN
    offices o ON e.officeCode = o.officeCode
```

```
WHERE
   o.country = 'USA';
```

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

```
SELECT
   lastName, firstName
FROM
   employees
WHERE
   officeCode IN (SELECT
         officeCode
      FROM
         offices
      WHERE
         country = 'USA');
```

14. 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)
GROUP BY customerNumber
ORDER BY amount;
```

15. 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 subquerry)

```
SELECT
    COUNT(customerName) AS CustomerCount,
    customerName,
    customerNumber
FROM
    customers
WHERE
    customerNumber NOT IN (SELECT DISTINCT
            customerNumber
        FROM
            orders)
GROUP BY
    customerName WITH ROLLUP;
```

16. 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,
    SUM(amount) AS totalSales,
    IF(SUM(amount) > 100000, 'High Value', 'Regular Value') AS
purchaseCategory
FROM
    customers
JOIN
    payments USING (customerNumber)
GROUP BY
    customerNumber, customerName;
```

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

```
SELECT emp.firstName as "EmployeeName" , mng.firstName as
"Manager_name"
FROM employees emp
JOIN employees mng on emp.employeeNumber = mng.reportsTo
```

18. 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
   emp.firstName AS "EmployeeName",
   mng.firstName AS "ManagerName"
FROM
   employees emp
JOIN
   employees mng ON emp.employeeNumber = mng.reportsTo
WHERE
   emp.jobTitle = mng.jobTitle;
```

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.

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 productName
FROM products p
WHERE EXISTS (
    SELECT 1
    FROM orderdetails od
    JOIN orders o ON od.orderNumber = o.orderNumber
    WHERE YEAR(o.orderDate) = YEAR(CURRENT_DATE())
    AND od.productCode = p.productCode
);
```