1. Write a query that displays for each employee's first name, Last name & his office phone.

```
SELECT employees.firstName,employees.lastName ,offices.phone
FROM employees join offices
on employees.officeCode = offices.officeCode;
```

2. Write a query that returns for each customer his name, postal code & his order date.

```
SELECT customers.customerName, customers.postalCode, orders.orderdate
FROM customers join orders
on customers.customerNumber = orders.customerNumber;
```

3. Retrieve and display the customer names and the shipping dates of their orders. This query utilizes a LEFT JOIN to ensure that even customers with no orders are included in the results.

```
SELECT customers.customerName, orders.shippeddate
FROM customers left join orders
on customers.customerNumber = orders.customerNumber;
```

4. retrieve and display the first names and email addresses of employees along with the second address lines of their corresponding offices. The query employs a RIGHT JOIN to ensure that all office addresses are included in the results, and it orders the results by country in ascending order.

```
SELECT employees.firstname, employees.email, offices.addressline2
FROM employees RIGHT JOIN offices
ON employees.officecode = offices.officecode
ORDER BY offices.country ASC;
```

5. Retrieve the first and last names of customers along with the first and last names of the employees who are responsible for the customers. Display the results in a single table

```
SELECT c.contactFirstName, c.contactLastName, e.firstName, e.lastName
FROM customers AS c JOIN employees AS e
ON c.salesRepEmployeeNumber = e.employeeNumber;
```

6. Retrieve the customer names, sales employee names, and office cities for all customers.

Show results in a single table.

```
SELECT c.customerName, e.firstName, e.lastName, o.city
FROM customers AS c JOIN employees AS e
ON c.salesRepEmployeeNumber = e.employeeNumber
JOIN offices AS o
ON e.officecode = o.officecode
```

7. Retrieve a list of customers along with the product line and text description.

```
SELECT c.customerName, pl.productLine, pl.textDescription
FROM customers AS c JOIN orders AS o
ON c.customerNumber = o.customerNumber
JOIN orderdetails AS od
ON o.ordernumber = od.ordernumber
JOIN products AS p
ON od.productcode = p.productcode
JOIN productlines AS pl
ON p.productline = pl.productline
```

8. Retrieve a list of customers with their names and phone numbers, along with the status

of their orders.

```
SELECT c.customerName, c.phone, o.status
FROM customers AS c JOIN orders AS o
ON c.customerNumber = o.customerNumber
```

9. Write a query that returns for each customer his first name , country and the quantity of the orders.

```
SELECT c.contactFirstName, c.country, od.quantityOrdered
FROM customers AS c JOIN orders AS o
ON c.customerNumber = o.customerNumber
JOIN orderdetails AS od
ON o.ordernumber = od.ordernumber
```

10. Write a query that returns for each customer his first name , country and the quantity of the orders.

```
SELECT c.contactFirstName, c.country, od.quantityOrdered
FROM customers AS c JOIN orders AS o
ON c.customerNumber = o.customerNumber
JOIN orderdetails AS od
ON o.ordernumber = od.ordernumber
```

11. Retrieve the customer names, order numbers, and product names for all products ordered by each customer. Display the results in a single table.

```
SELECT c.customerName, o.orderNumber, p.productName
FROM customers AS c JOIN orders AS o
ON c.customerNumber = o.customerNumber
JOIN orderdetails AS od
ON o.ordernumber = od.ordernumber
JOIN products AS p
ON od.productCode = p.productCode
```

12. Retrieve the customer names, order numbers, and product names for all products ordered by each customer, and show the results in a single table. Additionally, include the quantity ordered for each product.

```
SELECT c.customerName, o.orderNumber, p.productName, od.quantityOrdered
FROM customers AS c JOIN orders AS o
ON c.customerNumber = o.customerNumber
JOIN orderdetails AS od
ON o.ordernumber = od.ordernumber
JOIN products AS p
ON od.productCode = p.productCode
```

13. Retrieve the customer names, order numbers, product names, and product line descriptions for all products ordered by each customer, and show the results in a single table.

```
SELECT c.customerName, o.orderNumber, p.productName, pl.textDescription
FROM customers AS c JOIN orders AS o
ON c.customerNumber = o.customerNumber
JOIN orderdetails AS od
ON o.ordernumber = od.ordernumber
JOIN products AS p
ON od.productCode = p.productCode
JOIN productlines AS pl
ON p.productline = pl.productline
```

14. Retrieve the customer names, order numbers, and product codes for products ordered by customers, but only for orders where the total order price (quantity ordered * price each) is greater than $2,000. Show the results in a single table.

```
SELECT c.customerName, o.orderNumber, p.productCode
FROM customers AS c JOIN orders AS o
ON c.customerNumber = o.customerNumber
JOIN orderdetails AS od
ON o.ordernumber = od.ordernumber
```

```
JOIN products AS p
ON od.productCode = p.productCode
HAVING SUM(od.quantityOrdered * od.priceEach)> 2000;
```

15. Write a query that returns the customer's first name, phone, the date and the status of his order for the first quarter of 2004's year in descending order by the year.

```
SELECT c.contactFirstName, c.phone, o.orderDate, o.status
FROM customers AS c JOIN orders AS o
ON c.customerNumber = o.customerNumber
WHERE YEAR(o.orderDate) = 2004 AND QUARTER(o.orderDate) = 1
ORDER BY YEAR(o.orderDate) DESC;
```

16. Display customers name and their required date in ascending order.

```
SELECT c.contactFirstName, o.requiredDate
FROM customers AS c JOIN orders AS o
ON c.customerNumber = o.customerNumber
ORDER BY o.requiredDate ASC;
```

17. Write a query that returns the customer's last name, postal code, the status of his order and the quantity of the order in descending alphabetical order.

```
SELECT c.contactLastName, c.postalCode, o.status, od.quantityOrdered
FROM customers AS c JOIN orders AS o
ON c.customerNumber = o.customerNumber
JOIN orderdetails AS od
ON o.ordernumber = od.ordernumber
ORDER BY c.contactLastName DESC;
```

18. Retrieve a list of employees and their first names, last names, along with the city and country where their offices are located. Display the employee information in alphabetical

ascending order based on the employee's first name.

```
SELECT e.firstName, e.lastName, o.city, o.country
FROM employees AS e JOIN offices AS o
ON e.officeCode = o.officeCode
ORDER BY e.firstName ASC;
```

19. Retrieve for each customer, his name, phone, and the status of his order, in alphabetical descending order based on the employee's name

```
SELECT c.customerName, c.phone, o.status
FROM employees AS e JOIN customers AS c
ON e.employeeNumber = c.salesRepEmployeeNumber
JOIN orders AS o
ON c.customerNumber = o.customerNumber
ORDER BY e.firstName DESC;
```

20. Retrieve for each customer, his name, phone, and the status of his order where his quantity order is more than 40 in ascending order based on the quantity of the order.

```
SELECT c.customerName, c.phone, o.status
FROM customers AS c JOIN orders AS o
ON c.customerNumber = o.customerNumber
JOIN orderdetails AS od
ON o.ordernumber = od.ordernumber
WHERE od.quantityOrdered > 40
ORDER BY od.quantityOrdered ASC;
```

21. Retrieve the customer names and the quantity ordered for each order

```
SELECT c.customerName, od.quantityOrdered
FROM customers AS c JOIN orders AS o
ON c.customerNumber = o.customerNumber
```

```
JOIN orderdetails AS od
ON o.ordernumber = od.ordernumber
```

22. Retrieve and display information about customers, the products they've ordered, and the buying price of those products. Ensure that the results are ordered by the buying price in descending order.

```
SELECT c.customerName, p.productName, od.priceEach
FROM customers AS c JOIN orders AS o
ON c.customerNumber = o.customerNumber
JOIN orderdetails AS od
ON o.ordernumber = od.ordernumber
JOIN products AS p
ON od.productCode = p.productCode
ORDER BY od.priceEach DESC;
```

23. Retrieve the customer names, order numbers, and order dates for orders placed by customers whose contact last name contains 'son'. Calculate the number of days between the order date and the current date. Sort the results by the number of days in ascending order.

```
SELECT c.customerName, o.orderNumber, o.orderDate,
DATEDIFF(CURDATE(), o.orderDate) AS days
FROM customers AS c JOIN orders AS o
ON c.customerNumber = o.customerNumber
WHERE c.contactLastName LIKE '%son%'
ORDER BY days ASC;
```

24. Retrieve the employee numbers, first names, and last names of employees working in offices located in cities starting with the letter 'S'. Sort the results by office city in ascending order

```
SELECT e.firstName, e.lastName
FROM employees AS e JOIN offices AS o
ON e.officeCode = o.officeCode
Where o.city like 's%'
ORDER BY o.city ASC;
```

25. Retrieve the customer names, sales employee names, payment dates, and the difference in days between the payment date and the order date for each payment. Include only payments made more than 10 days after the order date. Sort the results by the difference in days in descending order.

```
SELECT c.customerName,e.firstName, e.lastName, p.paymentDate,
DATEDIFF(p.paymentDate, o.orderDate) AS diff
FROM employees AS e JOIN customers AS c
ON e.employeeNumber = c.salesRepEmployeeNumber
JOIN orders AS o
ON c.customerNumber = o.customerNumber
JOIN payments AS p
ON c.customerNumber = p.customerNumber
WHERE DATEDIFF(p.paymentDate, o.orderDate) > 10
ORDER BY diff DESC;
```

26. Retrieve the customer names, sales employee names, and the total payments made by customers whose contact last name contains 'son'. Calculate the total payments as the sum of all payment amounts. Sort the results by total payments in descending order.

```
SELECT c.customerName,e.firstName, e.lastName, SUM(p.amount) AS total
FROM employees AS e JOIN customers AS c
ON e.employeeNumber = c.salesRepEmployeeNumber
JOIN orders AS o
ON c.customerNumber = o.customerNumber
```

```
JOIN payments AS p
ON c.customerNumber = p.customerNumber
WHERE c.contactLastName LIKE '%son%'
ORDER BY total DESC;
```

27. Retrieve the order numbers, product names, and the total order amounts for each order. Calculate the total order amount by multiplying the quantity ordered by the price each. Include only orders with a total amount greater than $2,000. Sort the results by order amount in descending order.

```
SELECT o.orderNumber, p.productName, (od.quantityOrdered * od.priceEach) AS total
FROM customers AS c JOIN orders AS o
ON c.customerNumber = o.customerNumber
JOIN orderdetails AS od
ON o.ordernumber = od.ordernumber
JOIN products AS p
ON od.productcode = p.productcode
JOIN productlines AS pl
ON p.productline = pl.productline
WHERE (od.quantityOrdered * od.priceEach) > 2000
ORDER BY total DESC;
```

28. Retrieve the customer names, order numbers, employee names, and payment dates for orders placed by customers. Calculate the number of days between the order date and the payment date for each payment. Include only payments made more than 30 days after the order date. Sort the results by the number of days in ascending order.

```
SELECT c.customerName, o.orderNumber, e.firstName , e.lastName, p.paymentDate,
DATEDIFF(p.paymentDate, o.orderDate) AS diff
FROM employees AS e JOIN customers AS c
```

```
ON e.employeeNumber = c.salesRepEmployeeNumber
JOIN orders AS o
ON c.customerNumber = o.customerNumber
JOIN payments AS p
ON c.customerNumber = p.customerNumber
WHERE DATEDIFF(p.paymentDate, o.orderDate) > 30
ORDER BY diff ASC;
```

29. Retrieve the customer names, product names, product line descriptions, and the total quantities ordered for each product. Include only products ordered by customers. Sort the results by total quantity ordered in descending order.

```
SELECT c.customerName, p.productName, pl.textDescription, SUM(od.quantityOrdered) AS total
FROM customers AS c JOIN orders AS o
ON c.customerNumber = o.customerNumber
JOIN orderdetails AS od
ON o.ordernumber = od.ordernumber
JOIN products AS p
ON od.productcode = p.productcode
JOIN productlines AS pl
ON p.productline = pl.productline
ORDER BY total DESC;
```

30. retrieve and display information about products, including their names, Manufacturer's Suggested Retail Price (MSRP), discounted prices (10% off MSRP), and the text descriptions of their product lines. Ensure that the results are ordered by the discounted price in ascending order.

```
SELECT p.productName, p.MSRP, (p.msrp * 0.9) AS discounte, pl.textDescription
FROM customers AS c JOIN orders AS o
ON c.customerNumber = o.customerNumber
```

```
JOIN orderdetails AS od
ON o.ordernumber = od.ordernumber
JOIN products AS p
ON od.productcode = p.productcode
JOIN productlines AS pl
ON p.productline = pl.productline
ORDER BY discounte ASC;
```