```
-- Northwind db

USE northwinddb;

-- 1. How many employees are there in the territory Cambridge?

SELECT COUNT(DISTINCT e.EmployeeID) AS NumberOfEmployees

FROM Employees e

JOIN EmployeeTerritories et ON e.EmployeeID = et.EmployeeID

JOIN Territories t ON et.TerritoryID = t.TerritoryID

WHERE t.TerritoryDescription = 'Cambridge';

-- 2. Write a query to display the details of the employees and whom they are reporting to in the
following format:

-- EmployeeId      EmployeeName      ManagerName

SELECT

   e.EmployeeID AS EmployeeId,

   CONCAT(e.FirstName, ' ', e.LastName) AS EmployeeName,

   CONCAT(m.FirstName, ' ', m.LastName) AS ManagerName

FROM

   Employees e

LEFT JOIN

   Employees m ON e.ReportsTo = m.EmployeeID;

-- Q3: Shipper used mostly in delivering orders in Jan 2010


SELECT company_name AS shipper_name, total_orders

FROM (

  SELECT s.company_name, COUNT(o.order_id) AS total_orders,

       RANK() OVER (ORDER BY COUNT(o.order_id) DESC) AS rnk

  FROM orders o

  JOIN shippers s ON o.ship_via = s.shipper_id

  WHERE o.shipped_date BETWEEN '2010-01-01' AND '2010-01-31'

  GROUP BY s.company_name

) t

WHERE rnk = 1;
```

```
-- 4.List Customer wise, Product wise, total Number of products they ordered in following format:

-- customer_name|product_name|no_of_products

SELECT c.company_name AS customer_name,

p.product_name AS product_name,

SUM(od.quantity) AS no_of_products

FROM orders o

JOIN customers c ON o.customer_id=c.customer_id

JOIN order_details od ON o.order_id=od.order_id

JOIN products p ON od.product_id=p.product_id

GROUP BY c.company_name,p.product_name

ORDER BY c.company_name,no_of_products DESC;

-- 5.List the details of the customers and how many times their orders were

-- delayed?

SELECT c.contact_name AS CustomerName,

       c.customer_id,

       COUNT(o.order_id) AS DelayedOrders

FROM Orders o

JOIN Customers c ON o.customer_id = c.customer_id

WHERE o.shipped_date > o.required_date

GROUP BY c.customer_id, c.contact_name

ORDER BY DelayedOrders DESC;

-- 6.Display the top-10 customers of Southern region based on the number of orders they made in
the 1st and 2nd quarter of year 2010

SELECT customer_name, customer_id, total_orders

FROM (

   SELECT

      c.contact_name AS customer_name,

      c.customer_id,

      COUNT(o.order_id) AS total_orders,

      DENSE_RANK() OVER (ORDER BY COUNT(o.order_id) DESC) AS customer_rank

   FROM
```

```
        orders o
    JOIN
        customers c ON o.customer_id = c.customer_id
    WHERE
        c.region = 'Southern'
        AND o.order_date BETWEEN '2010-01-01' AND '2010-06-30'
    GROUP BY
        c.contact_name, c.customer_id
) ranked_customers
WHERE
    customer_rank <= 10
ORDER BY
    total_orders DESC;
-- 7.  Assume Remarks column is stored as a document in the Customer table. Write
-- a query to display the details of the customers which consists of the word ‘good
-- or better or best’.
SELECT * FROM Customers
WHERE Remarks LIKE '%good%'
      OR Remarks LIKE '%better%'
      OR Remarks LIKE '%best%';
-- 8.Write a query to display the 2nd costliest product in each category in the following
-- format
SELECT categoryName, product_name, unit_price
FROM (
    SELECT c.categoryName,
           p.product_name,
           p.unit_price,
           DENSE_RANK() OVER (PARTITION BY p.category_id ORDER BY p.unit_price DESC) AS ranks
    FROM Products p
    JOIN Categories c ON p.category_id = c.categories_id
) ranked_products
```

```
WHERE ranks = 2;
-- Q9: Display Customer ID, Company Name, Total Purchased Amount, and Classification
SELECT
  c.customer_id,
  c.company_name,
  ROUND(SUM(od.unit_price * od.quantity * (1 - od.discount)), 2) AS total_amount,
  CASE
    WHEN SUM(od.unit_price * od.quantity * (1 - od.discount)) BETWEEN 0 AND 5000 THEN 'Micro'
    WHEN SUM(od.unit_price * od.quantity * (1 - od.discount)) BETWEEN 5001 AND 10000 THEN
'Small'
    WHEN SUM(od.unit_price * od.quantity * (1 - od.discount)) BETWEEN 10001 AND 15000 THEN
'Medium'
    WHEN SUM(od.unit_price * od.quantity * (1 - od.discount)) BETWEEN 15001 AND 20000 THEN
'Large'
    ELSE 'Very Large'
  END AS customer_type
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
JOIN order_details od ON o.order_id = od.order_id
GROUP BY c.customer_id, c.company_name;
-- Q10: Dynamic SQL to pivot Customer vs Product total purchase amount

DECLARE @columns NVARCHAR(MAX), @sql NVARCHAR(MAX);

-- Step 1: Get distinct product names as pivot columns
SELECT @columns = STRING_AGG(QUOTENAME(product_name), ',')
FROM (
    SELECT DISTINCT p.product_name
    FROM products p
    JOIN order_details od ON p.product_id = od.product_id
) AS prod;
```

```
-- Step 2: Build dynamic SQL
SET @sql = '
SELECT customer_name, ' + @columns + '
FROM (
   SELECT
      c.company_name AS customer_name,
      p.product_name,
      ROUND(od.unit_price * od.quantity * (1 - od.discount), 2) AS total_amount
   FROM customers c
   JOIN orders o ON c.customer_id = o.customer_id
   JOIN order_details od ON o.order_id = od.order_id
   JOIN products p ON od.product_id = p.product_id
) AS source_table
PIVOT (
   SUM(total_amount) FOR product_name IN (' + @columns + ')
) AS pivot_table
ORDER BY customer_name;
';

-- Step 3: Execute the dynamic SQL
EXEC sp_executesql @sql;
-- 11. Generate a report as Monthwise, No. of orders placed by customer
SELECT c.contact_name AS CustomerName ,
     SUM(CASE WHEN MONTH(o.order_date) = 1 THEN 1 ELSE 0 END) AS JAN,
     SUM(CASE WHEN MONTH(o.order_date) = 2 THEN 2 ELSE 0 END) AS FEB,
     SUM(CASE WHEN MONTH(o.order_date) = 3 THEN 3 ELSE 0 END) AS MAR,
     SUM(CASE WHEN MONTH(o.order_date) = 4 THEN 4 ELSE 0 END) AS APR,
     SUM(CASE WHEN MONTH(o.order_date) = 5 THEN 5 ELSE 0 END) AS MAY,
     SUM(CASE WHEN MONTH(o.order_date) = 6 THEN 6 ELSE 0 END) AS JUNE
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id
```

```
GROUP BY c.contact_name

ORDER BY c.contact_name;

-- 12. Generate a report as Monthwise, No. of products ordered in each category

SELECT c.categoryName ,

    SUM(CASE WHEN MONTH(o.order_date) = 1 THEN 1 ELSE 0 END) AS JAN ,

    SUM(CASE WHEN MONTH(o.order_date) = 2 THEN 2 ELSE 0 END) AS FEB ,

    SUM(CASE WHEN MONTH(o.order_date) = 3 THEN 3 ELSE 0 END) AS MARCH ,

    SUM(CASE WHEN MONTH(o.order_date) = 4 THEN 4 ELSE 0 END) AS APRIL,

    SUM(CASE WHEN MONTH(o.order_date) = 5 THEN 5 ELSE 0 END) AS MAY ,

    SUM(CASE WHEN MONTH(o.order_date) = 6 THEN 6 ELSE 0 END) AS JUNE ,

    SUM(CASE WHEN MONTH(o.order_date) = 7 THEN 7 ELSE 0 END) AS JUL ,

    SUM(CASE WHEN MONTH(o.order_date) = 8 THEN 8 ELSE 0 END) AS AUG ,

    SUM(CASE WHEN MONTH(o.order_date) = 9 THEN 9 ELSE 0 END) AS SEP ,

    SUM(CASE WHEN MONTH(o.order_date) = 10 THEN 10 ELSE 0 END) AS OCT ,

    SUM(CASE WHEN MONTH(o.order_date) = 11 THEN 11 ELSE 0 END) AS NOV ,

    SUM(CASE WHEN MONTH(o.order_date) = 12 THEN 12 ELSE 0 END) AS DECEMBER

FROM order_details od

JOIN Orders o ON od.order_id = o.order_id

JOIN Products p ON od.product_id = p.product_id

JOIN Categories c ON p.category_id = c.categories_id

GROUP BY c.CategoryName

ORDER BY c.CategoryName;

-- 13. Customerwise, Employeewise, No. of orders, prepare a report using Rollup/Cube.

SELECT c.contact_name AS customer_name, e.first_name AS Employee_name , COUNT(o.order_id)
AS Total_no_orders

FROM orders o

JOIN customers c ON o.customer_id = c.customer_id

JOIN employees e ON o.employee_id = e.employee_id

GROUP BY c.contact_name , e.first_name WITH ROLLUP;

-- 14. Categorywise, Productwise, No. of quantity ordered, prepare a report using Rollup/Cube.

SELECT c.categoryName AS categories_name , p.product_name AS Product_name ,
COUNT(od.quantity) AS TotalQuantityOrdered
```

```
FROM order_details od

JOIN Products p ON od.product_id = p.product_id

JOIN categories c ON p.category_id = c.categories_id

GROUP BY c.categoryName , p.product_name WITH ROLLUP;


-- 15 . Create a report based on purchase for the following conditions for employees.

-- a) Orders greater than 30 – Good Performer.

-- b) Orders greater than 10 – Average Performer.

-- c) Orders less than 10 – Poor Performer.

SELECT e.first_name  AS Employee_Name , COUNT(o.order_id) AS No_of_orders ,

    CASE

    WHEN COUNT(o.order_id) > 30 THEN 'Good Performer'

    WHEN COUNT(o.order_id) > 10 THEN 'Average Performer'

    ELSE 'Poor performer'

    END AS Performance

FROM orders o

JOIN  employees e ON o.employee_id = e.employee_id

GROUP BY e.first_name , e.employee_id

ORDER BY No_of_orders DESC;
```