**USMAN INSTITUTE OF TECHNOLOGY**

**Department of Computer Science**

**CS311 Introduction to Database Systems**

Lab#5

**Objective:**

**-Advance SQL Joins.**

**Name of Student MAIRA USMAN**

**Roll No: 21b-011-SE Sec. B**

**Date of Experiment:**

**Marks Obtained/Remarks: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

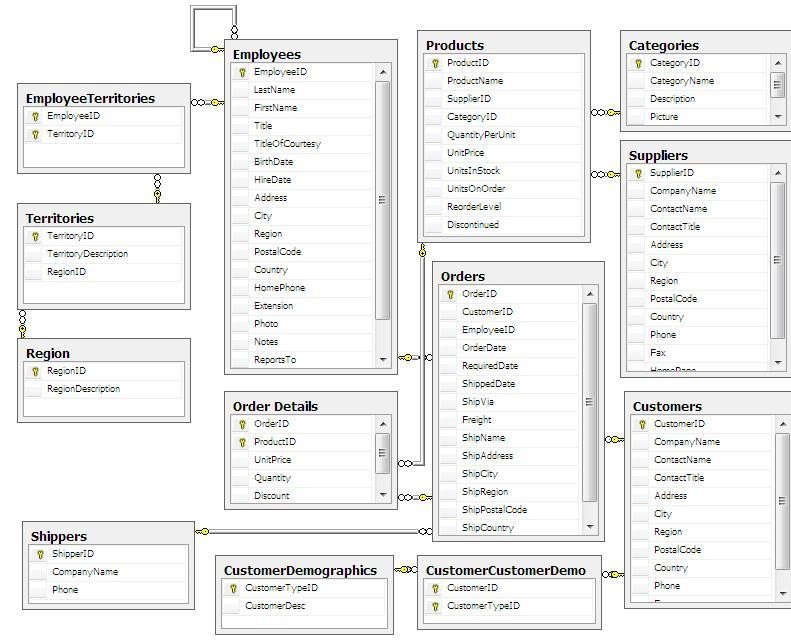
1. **Northwind Database**

The database is about a company named "**Northwind** **Traders**". The database captures all the sales transactions that occurs between the company i.e. Northwind traders and its customers as well as the purchase transactions between Northwind and its suppliers.



The Northwind dataset includes sample data for the following.

* **Suppliers**: Suppliers and vendors of Northwind
* **Customers**: Customers who buy products from Northwind
* **Employees**: Employee details of Northwind traders
* **Products**: Product information
* **Shippers**: The details of the shippers who ship the products from the traders to the end-customers
* **Orders and Order\_Details**: Sales Order transactions taking place between the customers & the company

The Northwind sample database includes 14 tables and the table relationships are showcased in the following entity relationship diagram.

1. **INNER JOINS**

Now, how can we find out

* Which products are provided by which suppliers?
* Which customers placed which orders?
* Which customers are buying which products?

Such reports require data from multiple tables.

Creating a report that returns the employee id and order id from the Orders table is not difficult.

SELECT EmployeeID, OrderID

FROM Orders;

But this is not very useful as we cannot tell who the employee is that got this order. The next sample shows how we can use a join to make the report more useful.

SELECT Employees.EmployeeID, Employees.FirstName,

    Employees.LastName, Orders.OrderID, Orders.OrderDate

FROM Employees JOIN Orders ON

    (Employees.EmployeeID = Orders.EmployeeID)

ORDER BY Orders.OrderDate;

Do this using table aliases.

Products table in Northwind database only stores SupplierID which is a foreign key pointing back to SupplierID column in suppliers table. If we want to know the supplier's name for a product, we need to write a query to join with suppliers table to get this information. In this practice, a single result set is returned which displays product name and the supplier's name for each product.

/\*  
This query returns supplier's name for each product.  
Note that the result is ordered by column alias SupplierName.  
\*/  
SELECT p.ProductName,   
 s.CompanyName AS SupplierName  
FROM products p  
INNER JOIN suppliers s ON p.SupplierID=s.SupplierID  
ORDER BY SupplierName;

1. **With where clause:**

/\*  
The following two queries return the same result set.  
   
The first query displays which companies placed orders   
in between 1998-05-04 and 1998-05-06.  
   
The second query displays which companies placed orders   
after 1998-05-03.  
\*/  
   
-- Query 1  
SELECT DISTINCT c.CompanyName, o.OrderDate  
FROM orders AS o  
INNER JOIN Customers AS c ON o.CustomerID=c.CustomerID  
WHERE o.OrderDate BETWEEN '1998-05-04' AND '1998-05-06'  
ORDER BY o.OrderDate;  
   
-- Query 2  
SELECT DISTINCT c.CompanyName, o.OrderDate  
FROM orders AS o  
INNER JOIN Customers AS c ON o.CustomerID=c.CustomerID  
WHERE o.OrderDate > '1998-05-03'  
ORDER BY o.OrderDate;

1. **Multi-table Joins**

**Syntax**

|  |  |
| --- | --- |
|  | SELECT table1.column, table2.column, table3.column  FROM table1      JOIN table2 ON (table1.column=table2.column)      JOIN table3 ON (table2.column=table3.column)  WHERE conditions |

Note that, to join with a table, that table must be in the FROM clause or must already be joined with the table in theFROM clause. Consider the following.

|  |  |
| --- | --- |
|  | SELECT table1.column, table2.column, table3.column  FROM table1      JOIN table3 ON (table2.column=table3.column)      JOIN table2 ON (table1.column=table2.column)  WHERE conditions |

The above code would break because it attempts to join table3 with table2 before table2 has been joined withtable1.

* Create a report showing the Order ID, the name of the company that placed the order, and the first and last name of the associated employee. Only show orders placed after January 1, 1998 that shipped after they were required. Sort by Company Name.

SELECT o.OrderID, c.CompanyName, e.FirstName, e.LastName

FROM Orders o

    JOIN Employees e ON (e.EmployeeID = o.EmployeeID)

    JOIN Customers c ON (c.CustomerID = o.CustomerID)

WHERE o.ShippedDate > o.RequiredDate AND o.OrderDate > '1-Jan-1998'

ORDER BY c.CompanyName;

**In this exercise, you will practice using joins.**

1. Create a report that shows the order ids and the associated employee names for orders that shipped after the required date. It should return the following. (37)
2. Create a report that shows the total quantity of products (from the Order\_Details table) ordered. Only show records for products for which the quantity ordered is fewer than 200. (5)
3. Create a report that shows the total number of orders by Customer since December 31, 1996. The report should only return rows for which the NumOrders is greater than 15.(5)
4. Create a report that shows the company name, order id, and total price of all products of which Northwind has sold more than $10,000 worth. There is no need for a GROUP BY clause in this report.
5. **OUTTER JOINS**

So far, all the joins we have worked with are inner joins, meaning that rows are only returned that have matches in both tables. For example, when doing an inner join between the Employees table and the Orders table, only employees that have matching orders and orders that have matching employees will be returned.

As a point of comparison, let's first look at another inner join.

Create a report that shows the number of employees and customers from each city that has employees in it.

SELECT COUNT(DISTINCT e.EmployeeID) AS numEmployees,

    COUNT(DISTINCT c.CustomerID) AS numCompanies,

    e.City, c.City

FROM Employees e JOIN Customers c ON

    (e.City = c.City)

GROUP BY e.City, c.City

ORDER BY numEmployees DESC;

**Left Joins**

A LEFT JOIN (also called a LEFT OUTER JOIN) returns all the records from the first table even if there are no matches in the second table.

|  |
| --- |
| SELECT table1.column, table2.column  FROM table1      LEFT [OUTER] JOIN table2 ON (table1.column=table2.column)  WHERE conditions |

All rows in table1 will be returned even if they do not have matches in table2.

|  |
| --- |
| Create a report that shows the number of      employees and customers from each city that has employees in it.  \*/    SELECT COUNT(DISTINCT e.EmployeeID) AS numEmployees,      COUNT(DISTINCT c.CustomerID) AS numCompanies,      e.City, c.City  FROM Employees e LEFT JOIN Customers c ON      (e.City = c.City)  GROUP BY e.City, c.City  ORDER BY numEmployees DESC; |

All records in the  Employees table will be counted whether or not there are matching cities in the Customers table.

**Right Joins**

A RIGHT JOIN (also called a RIGHT OUTER JOIN) returns all the records from the second table even if there are no matches in the first table.

SELECT table1.column, table2.column

FROM table1

 RIGHT [OUTER] JOIN table2 ON (table1.column=table2.column)

WHERE conditions

All rows in table2 will be returned even if they do not have matches in table1.

/\*

    Create a report that shows the number of

    employees and customers from each city that has customers in it.

\*/

SELECT COUNT(DISTINCT e.EmployeeID) AS numEmployees,

    COUNT(DISTINCT c.CustomerID) AS numCompanies,

    e.City, c.City

FROM Employees e RIGHT JOIN Customers c ON

    (e.City = c.City)

GROUP BY e.City, c.City

ORDER BY numEmployees DESC;

All records in the Customers table will be counted whether or not there are matching cities in the Employees table.

**Full Outer Joins**

A FULL JOIN (also called a FULL OUTER JOIN) returns all the records from each table even if there are no matches in the joined table.

|  |
| --- |
| SELECT table1.column, table2.column  FROM table1      FULL [OUTER] JOIN table2 ON (table1.column=table2.column)  WHERE conditions |

All rows in table1 and table2 will be returned.

* Create a report that shows the number of employees and customers from each city.

SELECT COUNT(DISTINCT e.EmployeeID) AS numEmployees,

    COUNT(DISTINCT c.CustomerID) AS numCompanies,

    e.City, c.City

FROM Employees e FULL JOIN Customers c ON

    (e.City = c.City)

GROUP BY e.City, c.City

ORDER BY numEmployees DESC;

1. **Using Self Joins to Combine Data from the Same Table**

﻿When you join a table to itself on columns with common values, you can picture how each record is related to one another. This is known as self-join.

Self-join is normally used to represent hierarchical relationship or tree structure in a table. In Northwind employees table, an employee has a manager who is also an employee. Every employee has a ReportsTo value which stores the EmployeeID of employee's manager.

In employees table, EmployeeID is primary key and ReportsTo is foreign key which relates back to EmployeeID in the same table. So we can use ReportsTo and EmployeeID to join the employees table to itself and find out the manager for each employee.

/\*  
This query displays manager and staff relationship.  
   
The query uses self-join where employees table   
joined with itself.   
   
Joined columns:  
   
Foreign key column ReportsTo in the employees table   
which is aliased as staff table.  
   
Primary key column EmployeeID in the employees table  
which is aliased as manager table.  
\*/  
select manager\_tbl.FirstName as Manager,  
 staff\_tbl.FirstName as Staff   
from employees staff\_tbl  
self join employees manager\_tbl   
on staff\_tbl.ReportsTo=manager\_tbl.EmployeeID;

**Understand and Run sample queries and then write SQL for queries given in exercise.**

* select count(\*) AS NoOfOrders, max(orderdate) as LastOrder, min(orderdate) as FirstOrder from orders
* select customerid,employeeid,count(\*) AS NoOfOrders, max(orderdate) as LastOrder, min(orderdate) as FirstOrder from orders group by customerid order by customerid
* select customerid,o.employeeid, e.firstname,count(\*) as Totalorders from orders o, employees e

where o.employeeid = e.employeeid

group by customerid, o.employeeid, e.firstname

order by customerid

* Select Customerid, count(\*) from orders

group by customerid, EmployeeID

* Select Country, City, Count(\*) From Customers

Group By COuntry, City

* Select CompanyName, count(\*)

from orders o, customers c

where o.customerid = c.customerid group by CompanyName

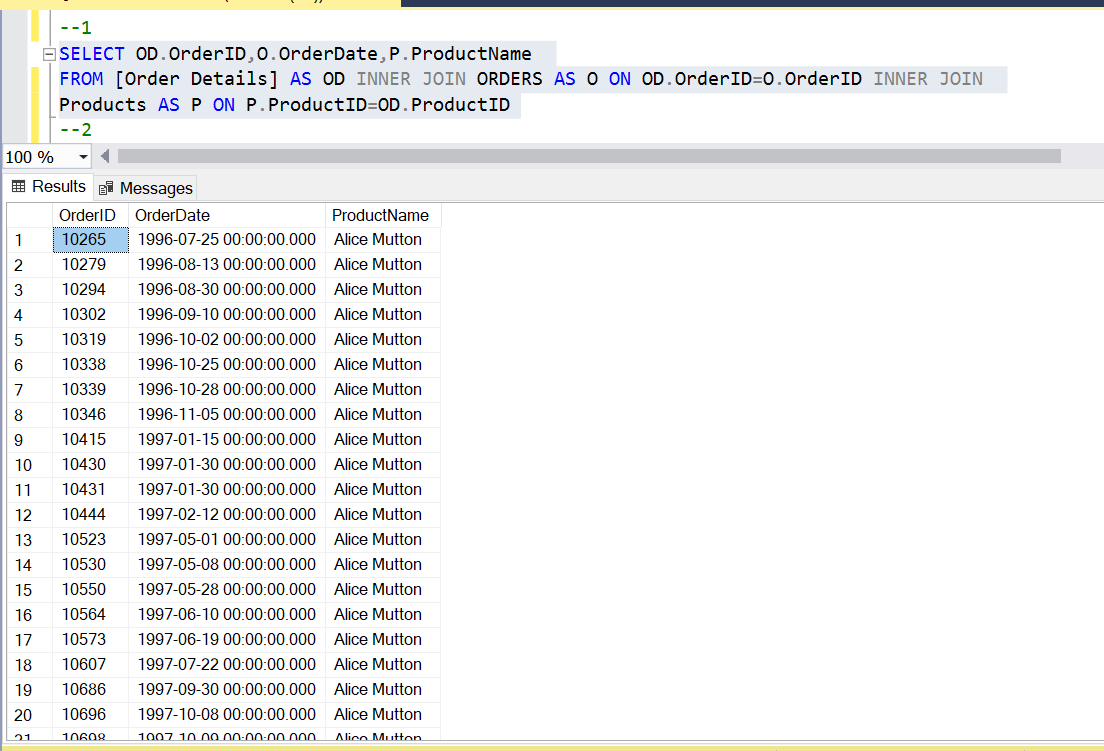
Having COunt(\*) > 5

1. Fetch following details

Result: Order No, Order Date, Product Name

SELECT OD.OrderID,O.OrderDate,P.ProductName

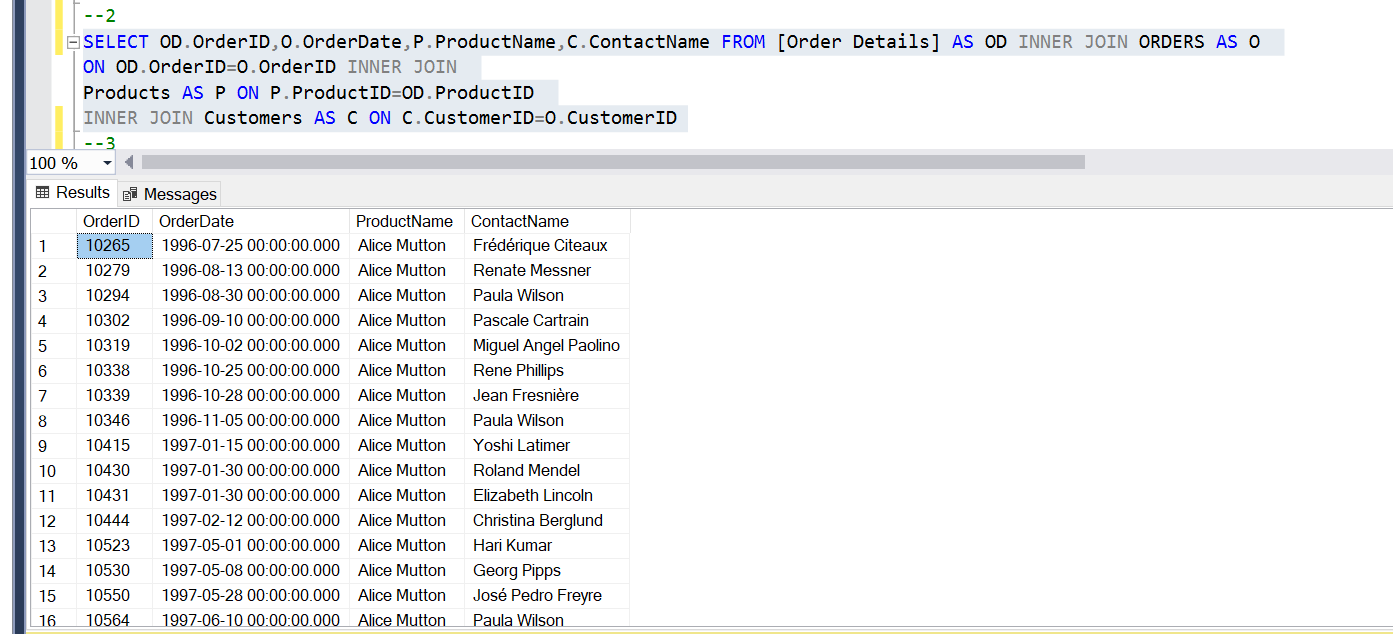
FROM [Order Details] AS OD INNER JOIN ORDERS AS O ON OD.OrderID=O.OrderID INNER JOIN Products AS P ON P.ProductID=OD.ProductID



1. Fetch following details

Result: Order No, Order Date, Product Name, Customer Name

SELECT OD.OrderID,O.OrderDate,P.ProductName,C.ContactName FROM [Order Details] AS OD INNER JOIN ORDERS AS O ON OD.OrderID=O.OrderID INNER JOIN Products AS P ON P.ProductID=OD.ProductID INNER JOIN Customers AS C ON C.CustomerID=O.CustomerID

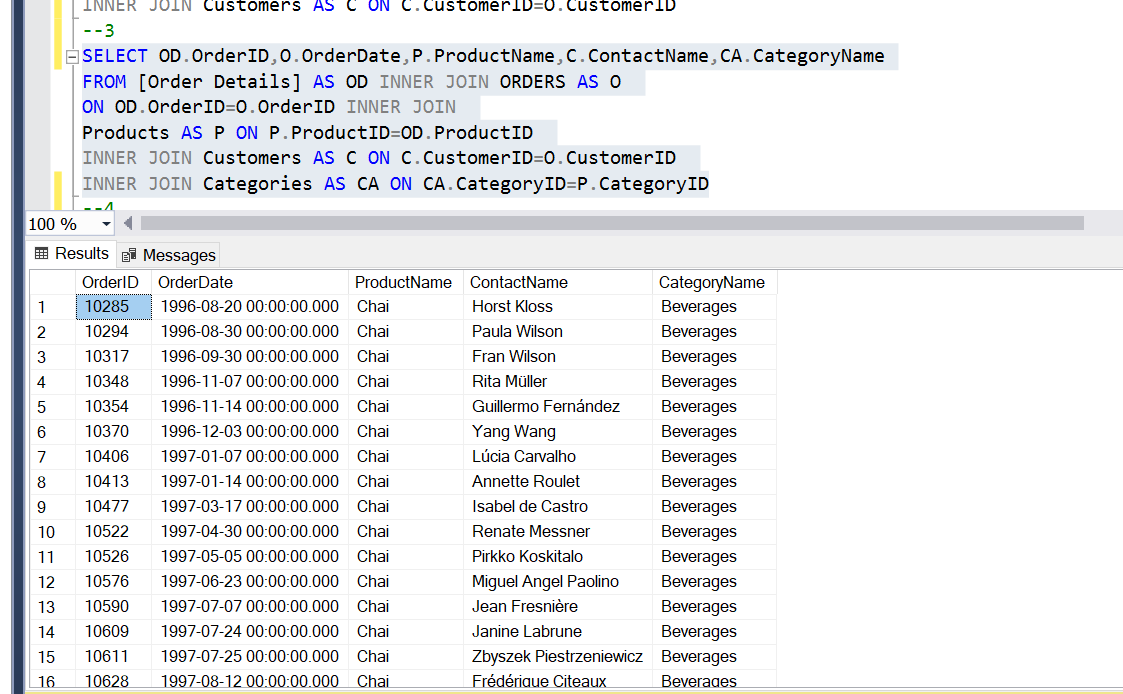


1. Fetch following details

Result: Order No, Order Date, Product Name, Category Name, Customer Name

SELECT OD.OrderID,O.OrderDate,P.ProductName,C.ContactName,CA.CategoryName

FROM [Order Details] AS OD INNER JOIN ORDERS AS O ON OD.OrderID=O.OrderID INNER JOIN Products AS P ON P.ProductID=OD.ProductID INNER JOIN Customers AS C ON C.CustomerID=O.CustomerID INNER JOIN Categories AS CA ON CA.CategoryID=P.CategoryID



1. Select all orders having products belonging to ‘Sea Food’ category Result: OrderNo, OrderDate, Product Name

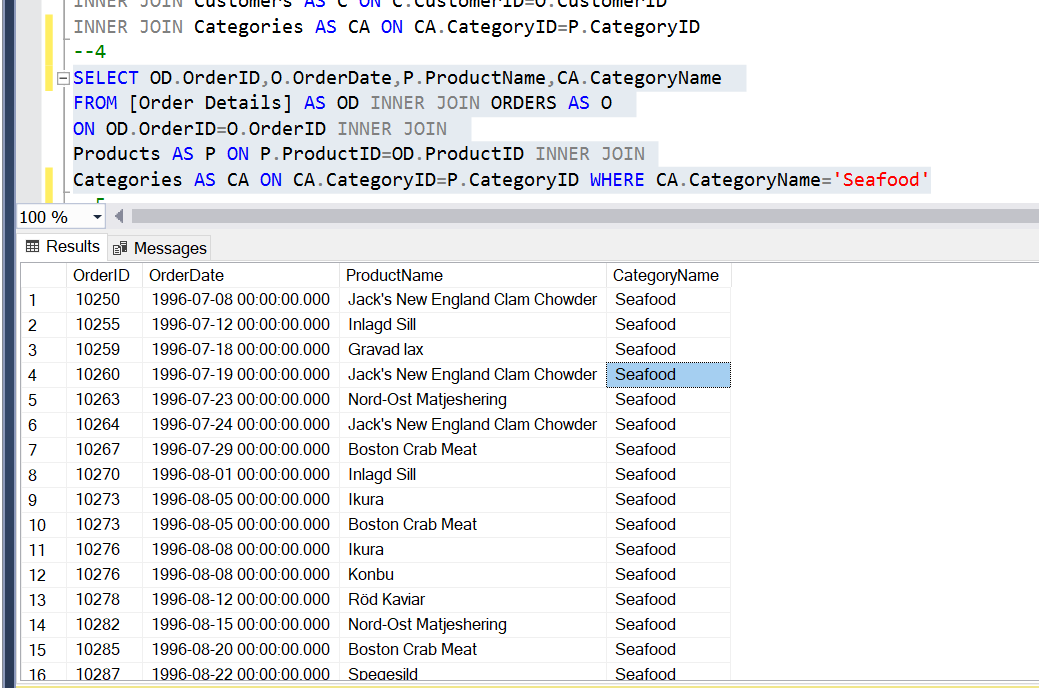
SELECT OD.OrderID,O.OrderDate,P.ProductName,CA.CategoryName

FROM [Order Details] AS OD INNER JOIN ORDERS AS O

ON OD.OrderID=O.OrderID INNER JOIN

Products AS P ON P.ProductID=OD.ProductID INNER JOIN

Categories AS CA ON CA.CategoryID=P.CategoryID WHERECA.CategoryName='Seafood'



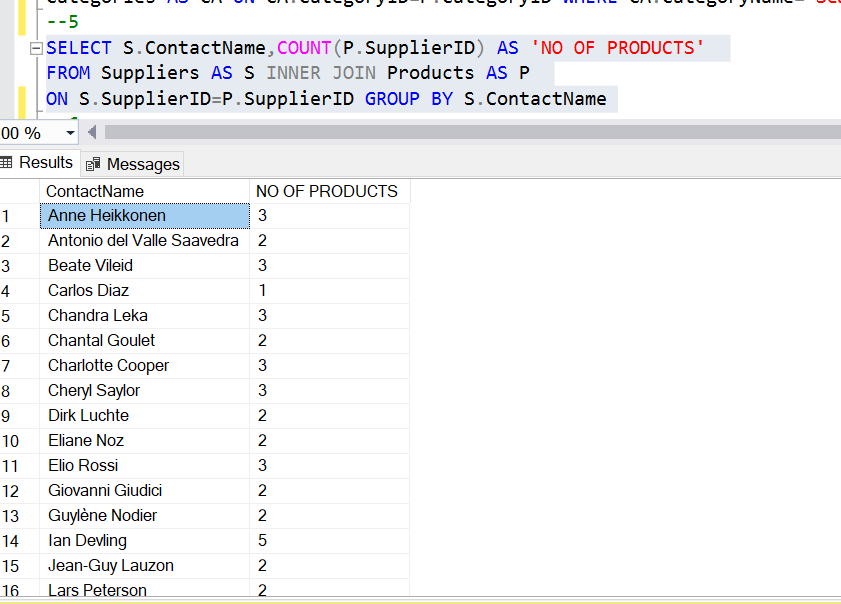
1. List suppliers in the order of no. of products supplied (Supplier Name, No Of Products).

Result: Supplier Name, No. of Products

SELECT S.ContactName,COUNT(P.SupplierID) AS 'NO OF PRODUCTS'

FROM Suppliers AS S INNER JOIN Products AS P

ON S.SupplierID=P.SupplierID GROUP BY S.ContactName



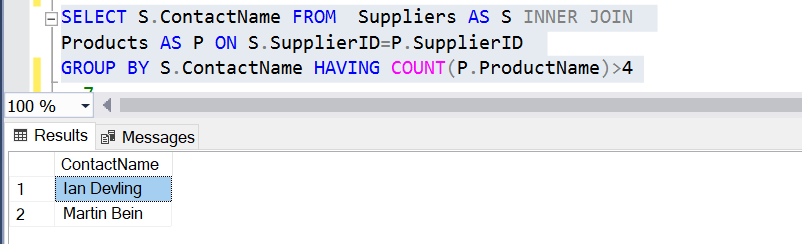
1. Select Suppliers supplying more than 4 products.

Result: Supplier Name

SELECT S.ContactName FROM Suppliers AS S INNER JOIN

Products AS P ON S.SupplierID=P.SupplierID

GROUP BY S.ContactName HAVING COUNT(P.ProductName)>4



1. Fetch no. of employees working in each region. (RegionName, No. of employees)

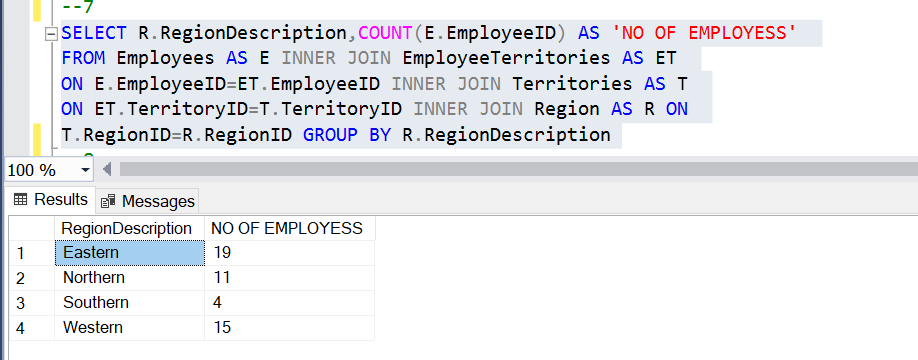
SELECT R.RegionDescription,COUNT(E.EmployeeID) AS 'NO OF EMPLOYESS'

FROM Employees AS E INNER JOIN EmployeeTerritories AS ET

ON E.EmployeeID=ET.EmployeeID INNER JOIN Territories AS T

ON ET.TerritoryID=T.TerritoryID INNER JOIN Region AS R ON

T.RegionID=R.RegionID GROUP BY R.RegionDescription



1. Fetch no. of employees in each region. If there is no employee in any region, even then region name should appear in the list with employee count of 0.

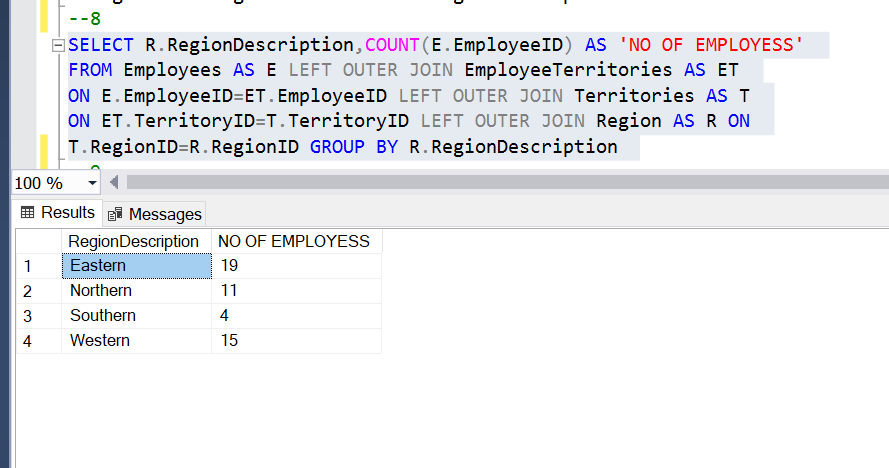
(RegionName, No. of employees)

SELECT R.RegionDescription,COUNT(E.EmployeeID) AS 'NO OF EMPLOYESS'

FROM Employees AS E LEFT OUTER JOIN EmployeeTerritories AS ET

ON E.EmployeeID=ET.EmployeeID LEFT OUTER JOIN Territories AS T

ON ET.TerritoryID=T.TerritoryID LEFT OUTER JOIN Region AS R ON T.RegionID=R.RegionID GROUP BY R.RegionDescription

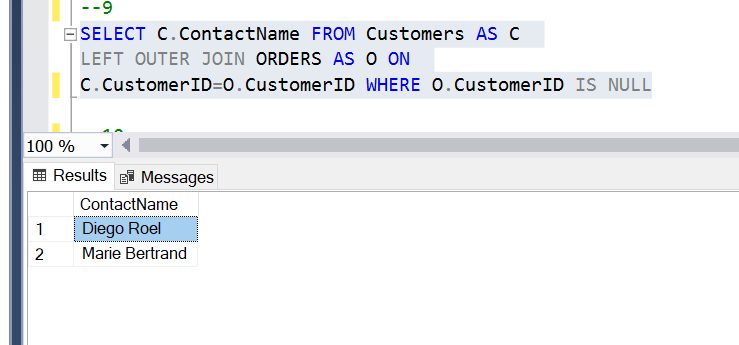


1. Fetch Customers who have not placed any order. (Customer Name)

SELECT C.ContactName FROM Customers AS C

LEFT OUTER JOIN ORDERS AS O ON

C.CustomerID=O.CustomerID WHERE O.CustomerID IS NULL



1. Select Top 3 employees of company. Employees are ranked on the basis of no. of orders they have processed.

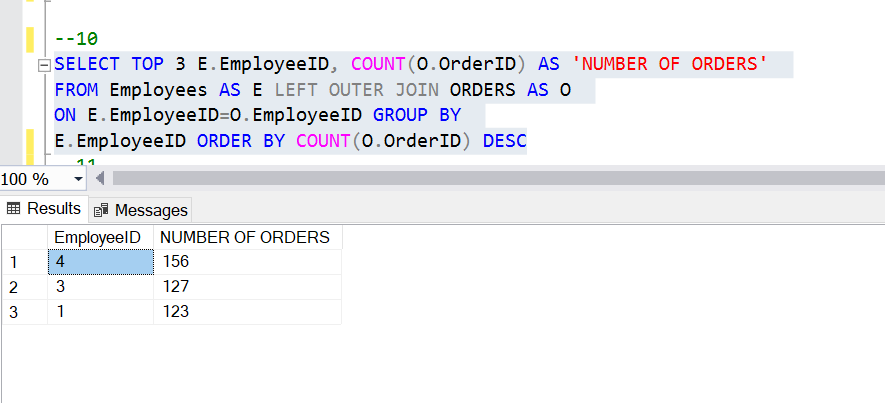
*(ROWNUM <= 3, ORDER BY number of orders processed)*

SELECT TOP 3 E.EmployeeID, COUNT(O.OrderID) AS 'NUMBER OF ORDERS'

FROM Employees AS E LEFT OUTER JOIN ORDERS AS O

ON E.EmployeeID=O.EmployeeID GROUP BY

E.EmployeeID ORDER BY COUNT(O.OrderID) DESC



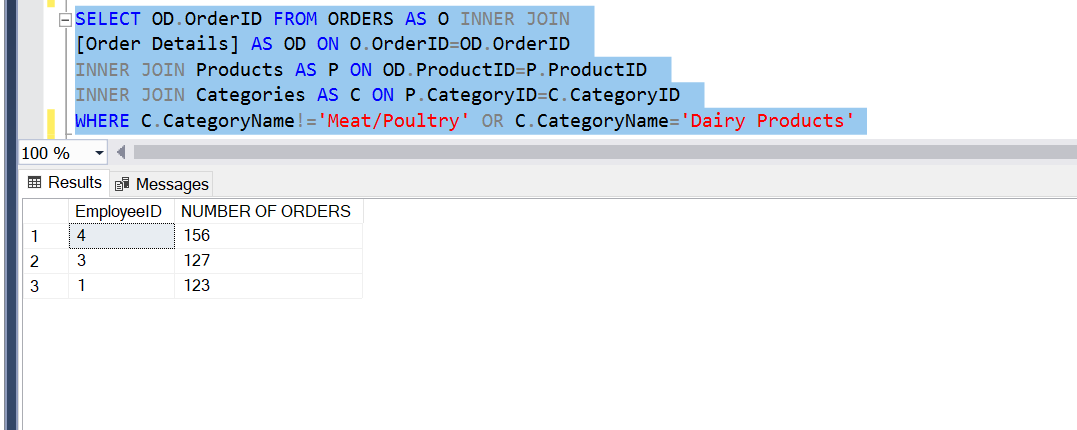
1. Select orders in which products of neither ‘Meat/Poultry’ nor ‘Dairy Products’ categories exist. (Order ID)

SELECT OD.OrderID FROM ORDERS AS O INNER JOIN

[Order Details] AS OD ON O.OrderID=OD.OrderID

INNER JOIN Products AS P ON OD.ProductID=P.ProductID

INNER JOIN Categories AS C ON P.CategoryID=C.CategoryID

WHERE C.CategoryName!='Meat/Poultry' OR C.CategoryName='Dairy Products'

1. Select total amount of each order.

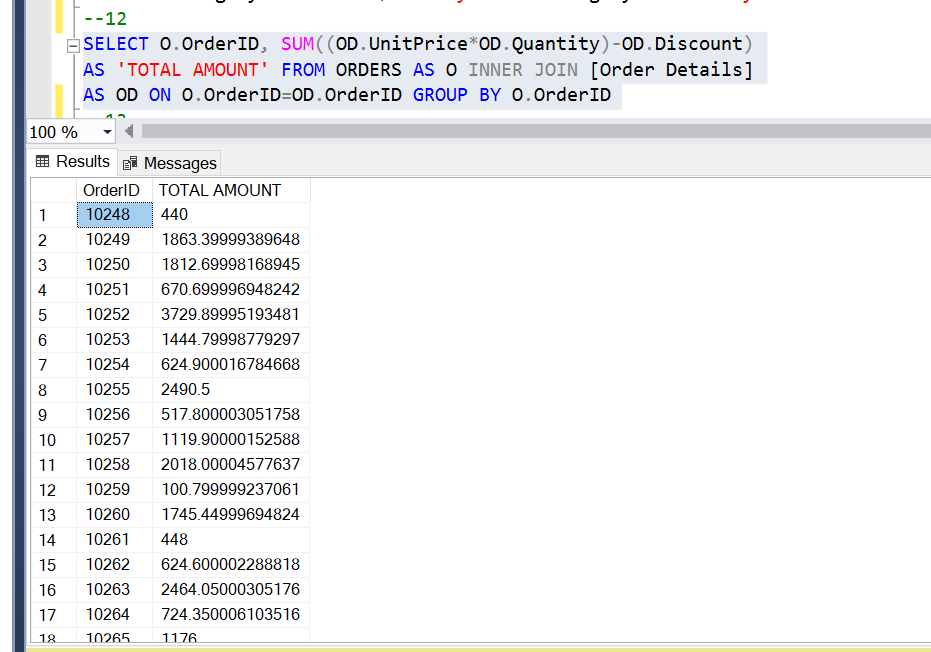
Result: Order ID, Total Amount

*[Total amount is calculated by summing up (Unit Price \* Qty)-Discount in order details.]*

SELECT O.OrderID, SUM((OD.UnitPrice\*OD.Quantity)-OD.Discount)

AS 'TOTAL AMOUNT' FROM ORDERS AS O INNER JOIN [Order Details]

AS OD ON O.OrderID=OD.OrderID GROUP BY O.OrderID

**

1. Find country to which maximum of customers belong.

SELECT top 1 COUNT(CUSTOMERID), COUNTRY FROM

Customers GROUP BY COUNTRY ORDER BY COUNT(CUSTOMERID) DESC

