USMAN INSTITUTE OF TECHNOLOGY

Department of Computer Science

CS311 Introduction to Database Systems

Lab#5

Objective:

- **Advance SQL Joins.**

Name of Student: Muhammad Waleed

Roll No: 20B-115-SE Sec. B

Date of Experiment:

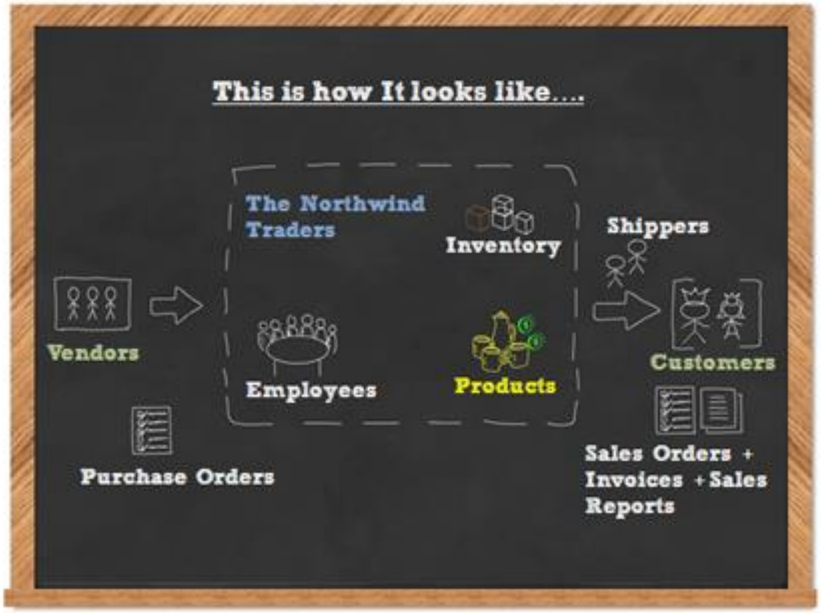
...

Marks Obtained/Remarks: ________________________________

Signature: ________________________________

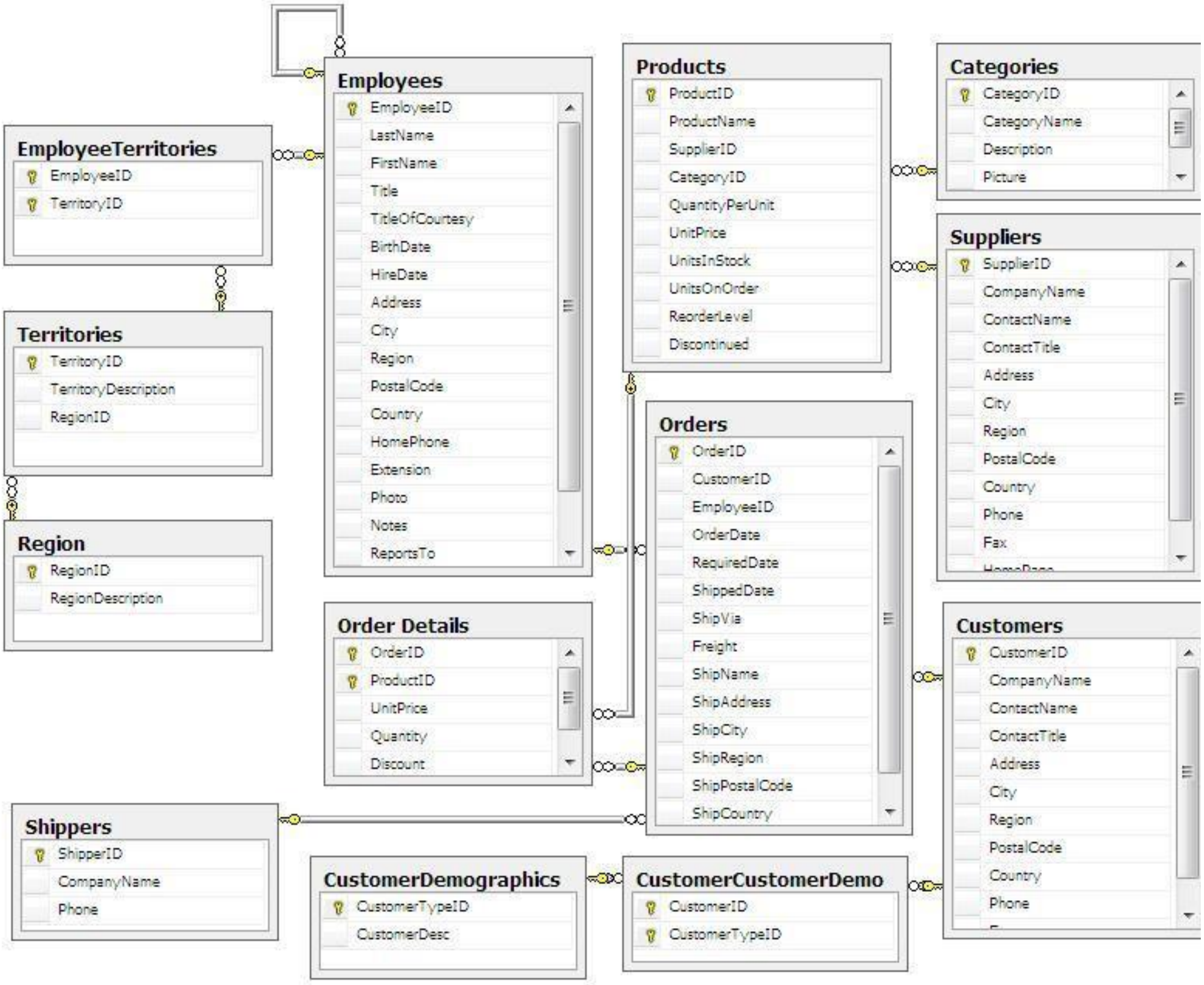
1. Northwind Database

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.



It contains the following detailed information:

1. Suppliers/Vendors of Northwind – who supply to the company.
2. Customers of Northwind – who buy from Northwind
3. Employee details of Northwind traders – who work for Northwind
4. The product information – the products that Northwind trades in
5. The inventory details – the details of the inventory held by Northwind traders.
6. The shippers – details of the shippers who ship the products from the traders to the end-customers
7. PO transactions i.e Purchase Order transactions – details of the transactions taking place between vendors & the company.
8. Sales Order transaction – details of the transactions taking place between the customers & the company.
9. Inventory transactions – details of the transactions taking place in the inventory
10. Invoices – details of the invoice raised against the order.



2. INNER JOINS

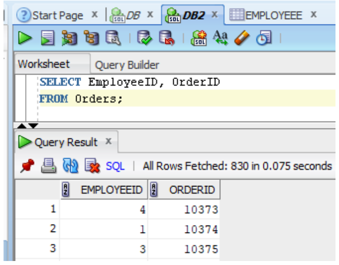
Now, how can we find out

- 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;
```



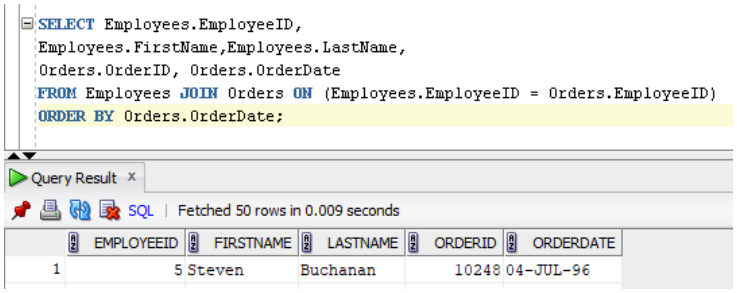
```
SELECT EmployeeID, OrderID
FROM Orders;
```

All Rows Fetched: 830 in 0.075 seconds

```
   EMPLOYEEID  ORDERID
1           4    10373
2           1    10374
3           3    10375
```

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;
```



```
SELECT Employees.EmployeeID,
Employees.FirstName,Employees.LastName,
Orders.OrderID, Orders.OrderDate
FROM Employees JOIN Orders ON (Employees.EmployeeID = Orders.EmployeeID)
ORDER BY Orders.OrderDate;
```

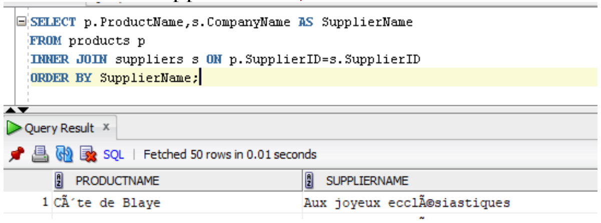
Fetched 50 rows in 0.009 seconds

```
   EMPLOYEEID FIRSTNAME LASTNAME  ORDERID ORDERDATE
1           5 Steven    Buchanan    10248 04-JUL-96
```

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;
```



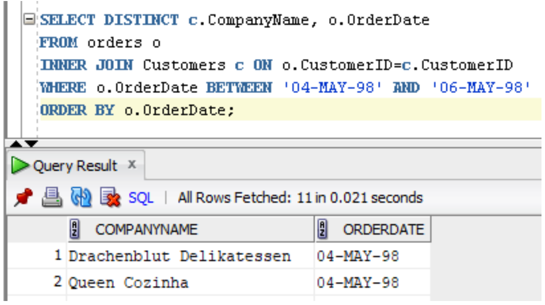
```
SELECT p.ProductName,s.CompanyName AS SupplierName
FROM products p
INNER JOIN suppliers s ON p.SupplierID=s.SupplierID
ORDER BY SupplierName;
```

Fetched 50 rows in 0.01 seconds

```
  PRODUCTNAME      SUPPLIERNAME
1 CÃ´te de Blaye   Aux joyeux ecclÃ©siastiques
```

3. With where clause:

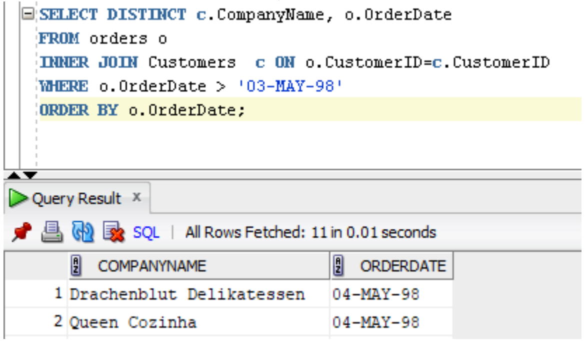
```
/*
The following two queries return the same result set.

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;
```



```
SELECT DISTINCT c.CompanyName, o.OrderDate
FROM orders o
INNER JOIN Customers c ON o.CustomerID=c.CustomerID
WHERE o.OrderDate BETWEEN '04-MAY-98' AND '06-MAY-98'
ORDER BY o.OrderDate;
```

```
  COMPANYNAME              ORDERDATE
1 Drachenblut Delikatessen 04-MAY-98
2 Queen Cozinha            04-MAY-98
```

```
-- 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;
```



```
SELECT DISTINCT c.CompanyName, o.OrderDate
FROM orders o
INNER JOIN Customers  c ON o.CustomerID=c.CustomerID
WHERE o.OrderDate > '03-MAY-98'
ORDER BY o.OrderDate;
```

```
  COMPANYNAME              ORDERDATE
1 Drachenblut Delikatessen 04-MAY-98
2 Queen Cozinha            04-MAY-98
```

4. 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
```

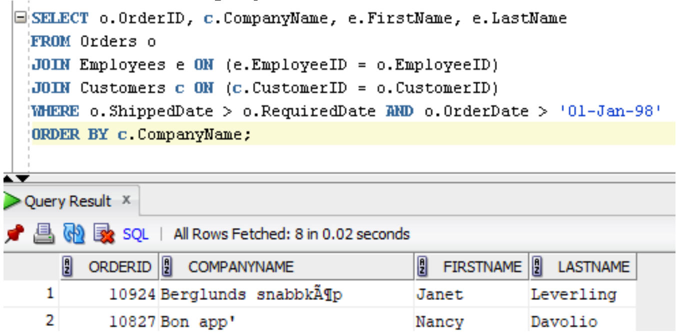
Note that, to join with a table, that table must be in the FROM clause or must already be joined with the table in theFROM clause. Consider the following.

```
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;
```



```
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 > '01-Jan-98'
ORDER BY c.CompanyName;
```

All Rows Fetched: 8 in 0.02 seconds

	ORDERID	COMPANYNAME	FIRSTNAME	LASTNAME
1	10924	Berglunds snabbkÃ¶p	Janet	Leverling
2	10827	Bon app'	Nancy	Davolio

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.

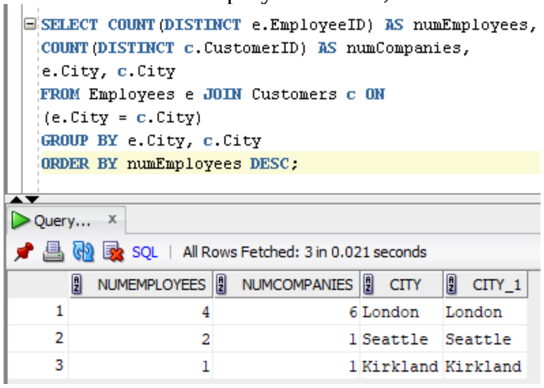
5. OUTTER JOINS

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;
```



```
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;
```

All Rows Fetched: 3 in 0.021 seconds

	NUMEMPLOYEES	NUMCOMPANIES	CITY	CITY_1
1	4	6	London	London
2	2	1	Seattle	Seattle
3	1	1	Kirkland	Kirkland

Left Joins

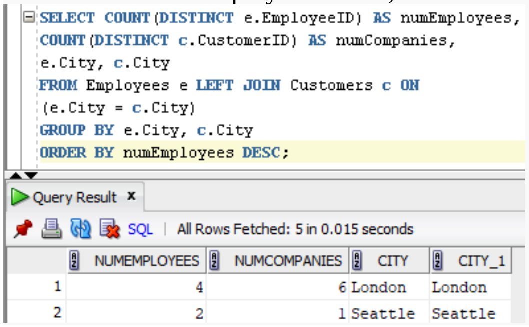
A LEFT JOIN (also called a LEFT OUTER JOIN) returns all the records from the first table even if there are no matches in the second table.

```
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;
```



```
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 Rows Fetched: 5 in 0.015 seconds

	NUMEMPLOYEES	NUMCOMPANIES	CITY	CITY_1
1	4	6	London	London
2	2	1	Seattle	Seattle

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

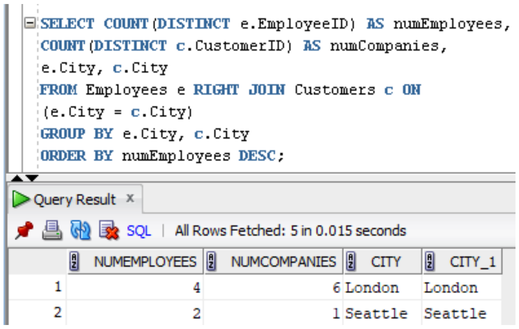
Right Joins

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;
```



```
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 Rows Fetched: 5 in 0.015 seconds

	NUMEMPLOYEES	NUMCOMPANIES	CITY	CITY_1
1	4	6	London	London
2	2	1	Seattle	Seattle

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

Full Outer Joins

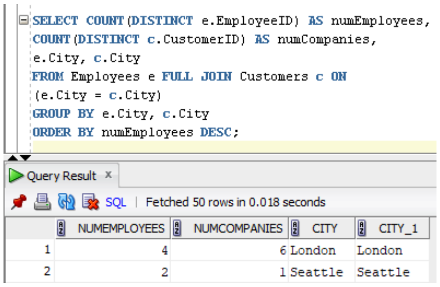
A FULL JOIN (also called a FULL OUTER JOIN) returns all the records from each table even if there are no matches in the joined table.

```
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;
```



```
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;
```

Fetched 50 rows in 0.018 seconds

	NUMEMPLOYEES	NUMCOMPANIES	CITY	CITY_1
1	4	6	London	London
2	2	1	Seattle	Seattle

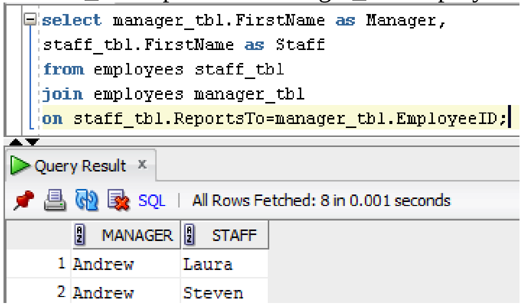
6. Using Self Joins to Combine Data from the Same Table

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;
```



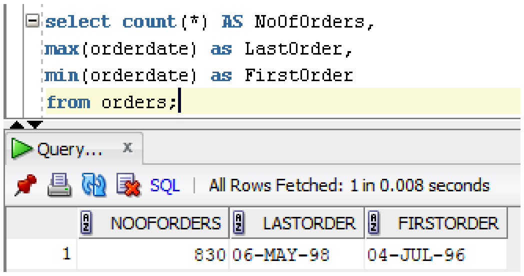
```
select manager_tbl.FirstName as Manager,
staff_tbl.FirstName as Staff
from employees staff_tbl
join employees manager_tbl
on staff_tbl.ReportsTo=manager_tbl.EmployeeID;
```

All Rows Fetched: 8 in 0.001 seconds

```
    MANAGER   STAFF
1   Andrew    Laura
2   Andrew    Steven
```

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

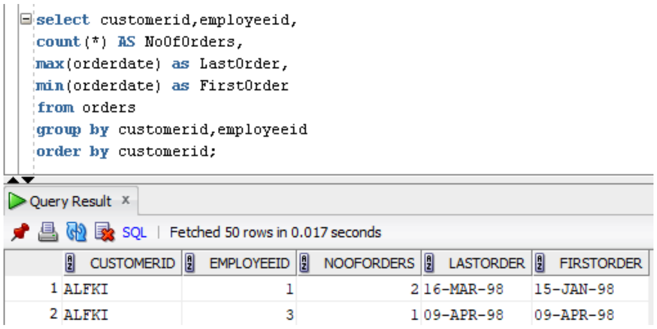


```
select count(*) AS NoOfOrders,
max(orderdate) as LastOrder,
min(orderdate) as FirstOrder
from orders;
```

All Rows Fetched: 1 in 0.008 seconds

	NOOFORDERS	LASTORDER	FIRSTORDER
1	830	06-MAY-98	04-JUL-96

- select customerid,employeeid,count(*) AS NoOfOrders, max(orderdate) as LastOrder, min(orderdate) as FirstOrder from orders group by customerid order by customerid



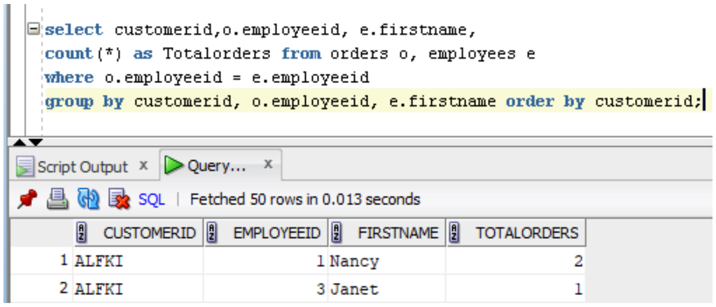
```
select customerid,employeeid,
count(*) AS NoOfOrders,
max(orderdate) as LastOrder,
min(orderdate) as FirstOrder
from orders
group by customerid,employeeid
order by customerid;
```

Fetched 50 rows in 0.017 seconds

	CUSTOMERID	EMPLOYEEID	NOOFORDERS	LASTORDER	FIRSTORDER
1	ALFKI	1	2	16-MAR-98	15-JAN-98
2	ALFKI	3	1	09-APR-98	09-APR-98

- 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

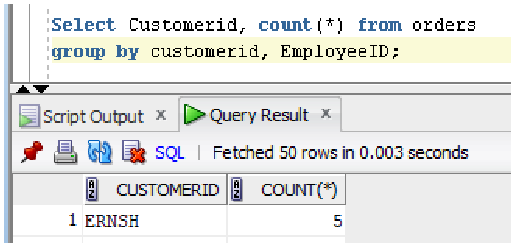


```
select customerid,o.employeeid, e.firstname,
count(*) as Totalorders from orders o, employees e
where o.employeeid = e.employeeid
group by customerid, o.employeeid, e.firstname order by customerid;
```

Fetched 50 rows in 0.013 seconds

	CUSTOMERID	EMPLOYEEID	FIRSTNAME	TOTALORDERS
1	ALFKI	1	Nancy	2
2	ALFKI	3	Janet	1

- Select Customerid,
  count(*) from
  orders
  group by customerid, EmployeeID



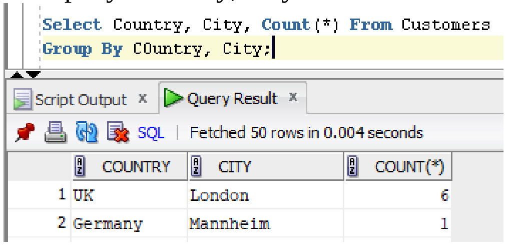
```
Select Customerid, count(*) from orders
group by customerid, EmployeeID;
```

Script Output × | Query Result ×

SQL | Fetched 50 rows in 0.003 seconds

```
     CUSTOMERID   COUNT(*)
   1 ERNSH               5
```

- Select Country,
  City, Count(*) From
  Customers
  Group By COuntry, City\



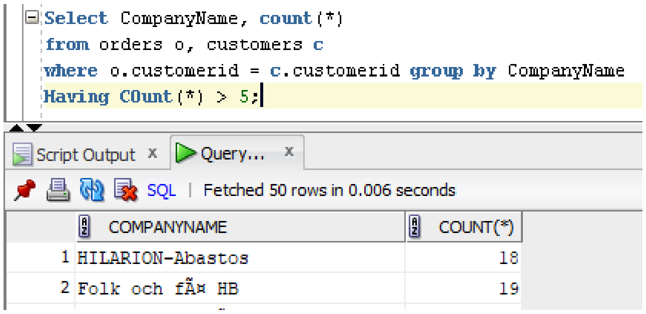
```
Select Country, City, Count(*) From Customers
Group By COuntry, City;
```

Script Output × | Query Result ×

SQL | Fetched 50 rows in 0.004 seconds

```
     COUNTRY   CITY        COUNT(*)
   1 UK        London             6
   2 Germany   Mannheim           1
```

- Select
  CompanyName,
  count(*)
  from orders o,
  customers c
  where o.customerid =
  c.customerid group by
  CompanyName
  Having COunt(*) > 5



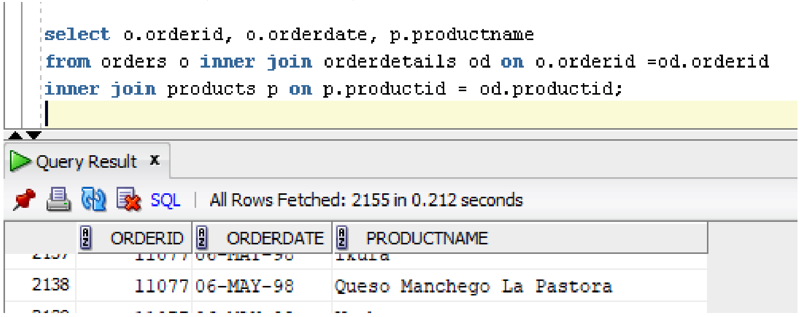
```
Select CompanyName, count(*)
from orders o, customers c
where o.customerid = c.customerid group by CompanyName
Having COunt(*) > 5;
```

Script Output × | Query... ×

SQL | Fetched 50 rows in 0.006 seconds

```
     COMPANYNAME              COUNT(*)
   1 HILARION-Abastos               18
   2 Folk och fÃ¤ HB                 19
```

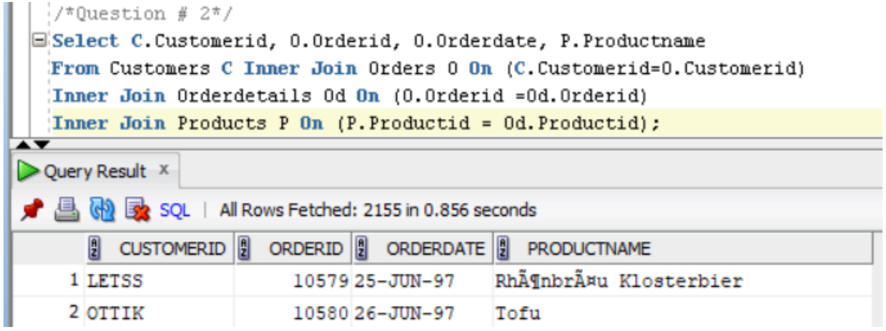
1. Fetch following details
   Result: Order No, Order Date, Product Name



```
select o.orderid, o.orderdate, p.productname
from orders o inner join orderdetails od on o.orderid =od.orderid
inner join products p on p.productid = od.productid;
```

```
All Rows Fetched: 2155 in 0.212 seconds

       ORDERID  ORDERDATE  PRODUCTNAME
2137     11077 06-MAY-98  Ikura
2138     11077 06-MAY-98  Queso Manchego La Pastora
```

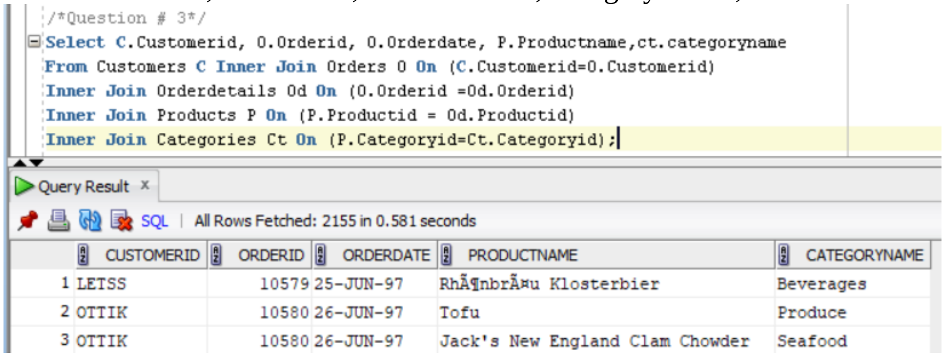
2. Fetch following details
   Result: Order No, Order Date, Product Name, Customer Name



```
/*Question # 2*/
Select C.Customerid, O.Orderid, O.Orderdate, P.Productname
From Customers C Inner Join Orders O On (C.Customerid=O.Customerid)
Inner Join Orderdetails Od On (O.Orderid =Od.Orderid)
Inner Join Products P On (P.Productid = Od.Productid);
```

```
All Rows Fetched: 2155 in 0.856 seconds

    CUSTOMERID  ORDERID ORDERDATE  PRODUCTNAME
  1 LETSS         10579 25-JUN-97  RhÃ¶nbrÃ¤u Klosterbier
  2 OTTIK         10580 26-JUN-97  Tofu
```

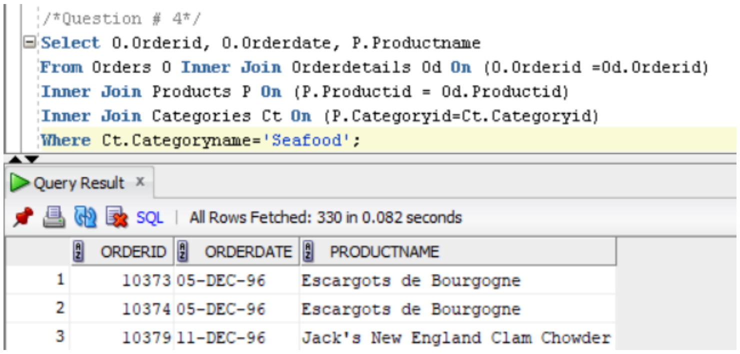
3. Fetch following details
   Result: Order No, Order Date, Product Name, Category Name, Customer Name



```
/*Question # 3*/
Select C.Customerid, O.Orderid, O.Orderdate, P.Productname,ct.categoryname
From Customers C Inner Join Orders O On (C.Customerid=O.Customerid)
Inner Join Orderdetails Od On (O.Orderid =Od.Orderid)
Inner Join Products P On (P.Productid = Od.Productid)
Inner Join Categories Ct On (P.Categoryid=Ct.Categoryid);
```

```
All Rows Fetched: 2155 in 0.581 seconds

    CUSTOMERID  ORDERID ORDERDATE  PRODUCTNAME                    CATEGORYNAME
  1 LETSS         10579 25-JUN-97  RhÃ¶nbrÃ¤u Klosterbier         Beverages
  2 OTTIK         10580 26-JUN-97  Tofu                           Produce
  3 OTTIK         10580 26-JUN-97  Jack's New England Clam Chowder Seafood
```

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

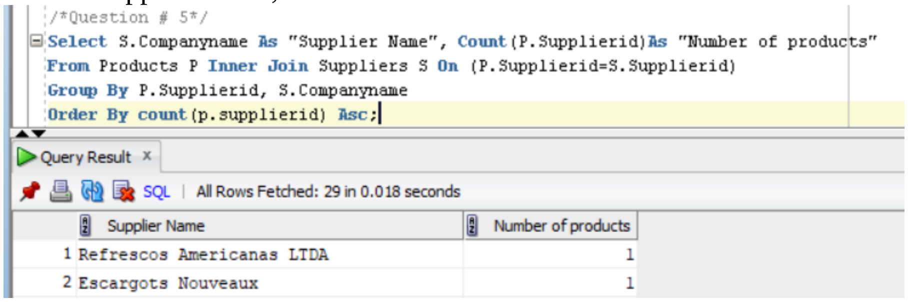


```
/*Question # 4*/
Select O.Orderid, O.Orderdate, P.Productname
From Orders O Inner Join Orderdetails Od On (O.Orderid =Od.Orderid)
Inner Join Products P On (P.Productid = Od.Productid)
Inner Join Categories Ct On (P.Categoryid=Ct.Categoryid)
Where Ct.Categoryname='Seafood';
```

All Rows Fetched: 330 in 0.082 seconds

	ORDERID	ORDERDATE	PRODUCTNAME
1	10373	05-DEC-96	Escargots de Bourgogne
2	10374	05-DEC-96	Escargots de Bourgogne
3	10379	11-DEC-96	Jack's New England Clam Chowder

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

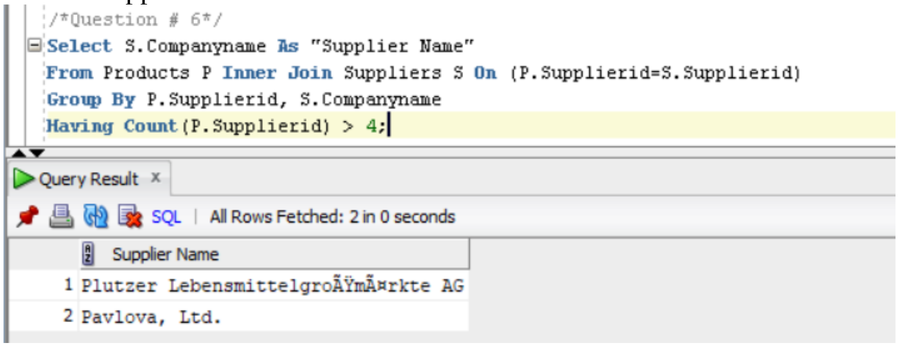


```
/*Question # 5*/
Select S.Companyname As "Supplier Name", Count(P.Supplierid)As "Number of products"
From Products P Inner Join Suppliers S On (P.Supplierid=S.Supplierid)
Group By P.Supplierid, S.Companyname
Order By count(p.supplierid) Asc;
```

All Rows Fetched: 29 in 0.018 seconds

	Supplier Name	Number of products
1	Refrescos Americanas LTDA	1
2	Escargots Nouveaux	1

6. Select Suppliers supplying more than 4 products.
   Result: Supplier Name

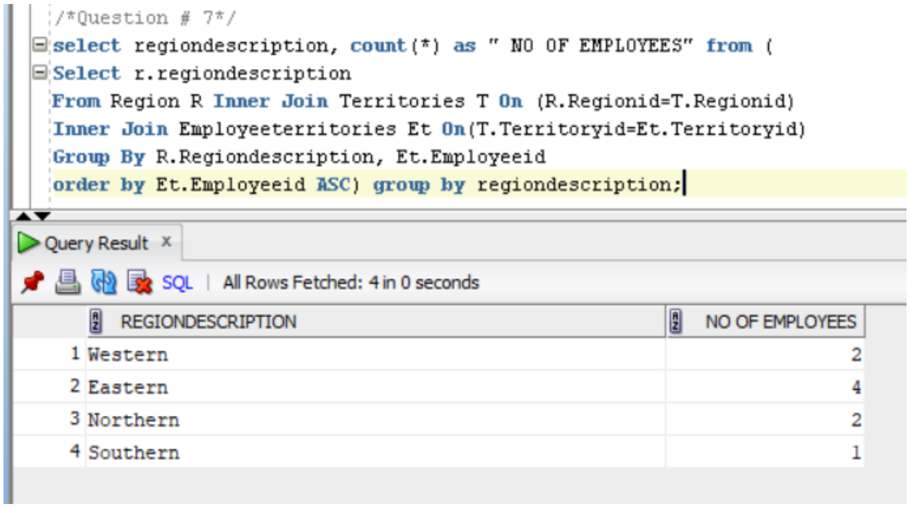


```
/*Question # 6*/
Select S.Companyname As "Supplier Name"
From Products P Inner Join Suppliers S On (P.Supplierid=S.Supplierid)
Group By P.Supplierid, S.Companyname
Having Count(P.Supplierid) > 4;
```

All Rows Fetched: 2 in 0 seconds

	Supplier Name
1	Plutzer LebensmittelgroÃŸmÃ¤rkte AG
2	Pavlova, Ltd.

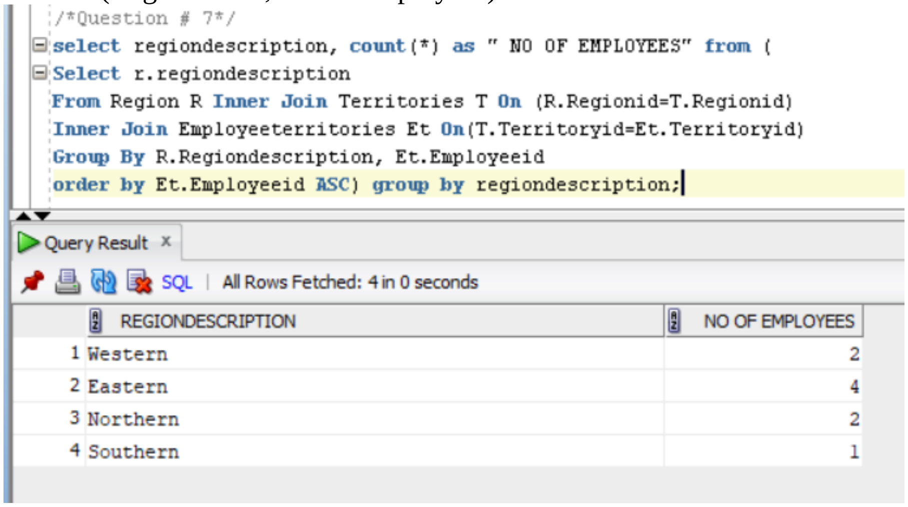
7. Fetch no. of employees working in each region. (RegionName, No. of employees)



```
/*Question # 7*/
select regiondescription, count(*) as " NO OF EMPLOYEES" from (
Select r.regiondescription
From Region R Inner Join Territories T On (R.Regionid=T.Regionid)
Inner Join Employeeterritories Et On(T.Territoryid=Et.Territoryid)
Group By R.Regiondescription, Et.Employeeid
order by Et.Employeeid ASC) group by regiondescription;
```

	REGIONDESCRIPTION	NO OF EMPLOYEES
1	Western	2
2	Eastern	4
3	Northern	2
4	Southern	1

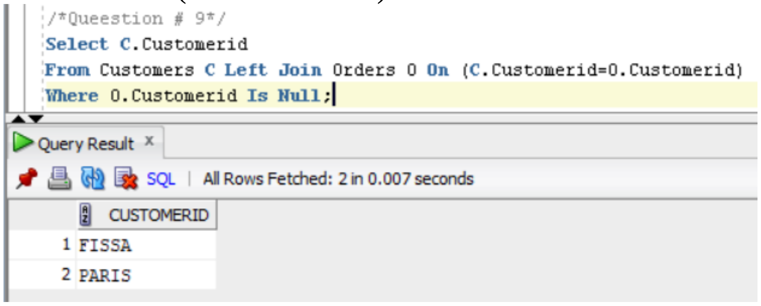
8. Fetch no. of employees in each region. If there is no employee in any region, even then region name should appear in the list with employee count of 0. (RegionName, No. of employees)



```
/*Question # 7*/
select regiondescription, count(*) as " NO OF EMPLOYEES" from (
Select r.regiondescription
From Region R Inner Join Territories T On (R.Regionid=T.Regionid)
Inner Join Employeeterritories Et On(T.Territoryid=Et.Territoryid)
Group By R.Regiondescription, Et.Employeeid
order by Et.Employeeid ASC) group by regiondescription;
```

	REGIONDESCRIPTION	NO OF EMPLOYEES
1	Western	2
2	Eastern	4
3	Northern	2
4	Southern	1

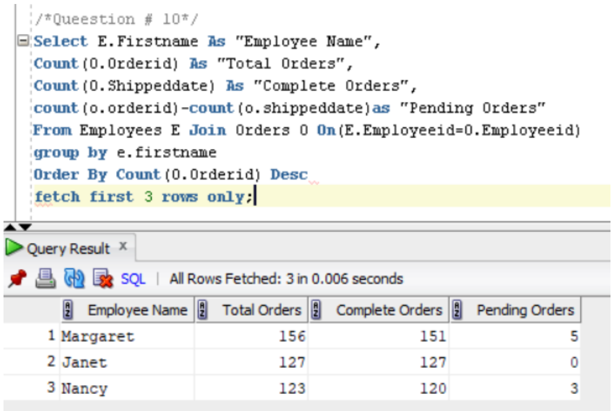
9. Fetch Customers who have not placed any order. (Customer Name)



```
/*Queestion # 9*/
Select C.Customerid
From Customers C Left Join Orders O On (C.Customerid=O.Customerid)
Where O.Customerid Is Null;
```

	CUSTOMERID
1	FISSA
2	PARIS

10. 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)

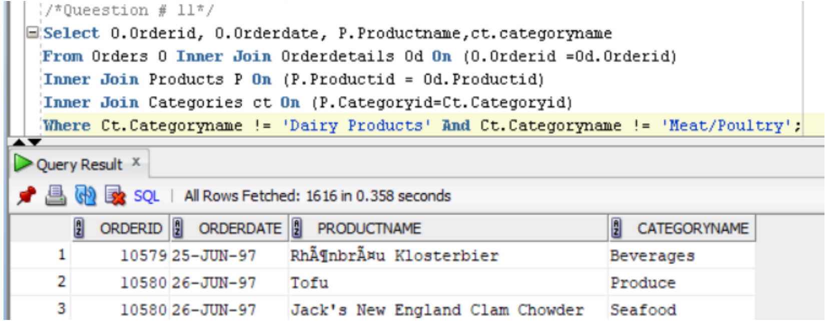


```
/*Queestion # 10*/
Select E.Firstname As "Employee Name",
Count(O.Orderid) As "Total Orders",
Count(O.Shippeddate) As "Complete Orders",
count(o.orderid)-count(o.shippeddate)as "Pending Orders"
From Employees E Join Orders O On(E.Employeeid=O.Employeeid)
group by e.firstname
Order By Count(O.Orderid) Desc
fetch first 3 rows only;
```

All Rows Fetched: 3 in 0.006 seconds

	Employee Name	Total Orders	Complete Orders	Pending Orders
1	Margaret	156	151	5
2	Janet	127	127	0
3	Nancy	123	120	3

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

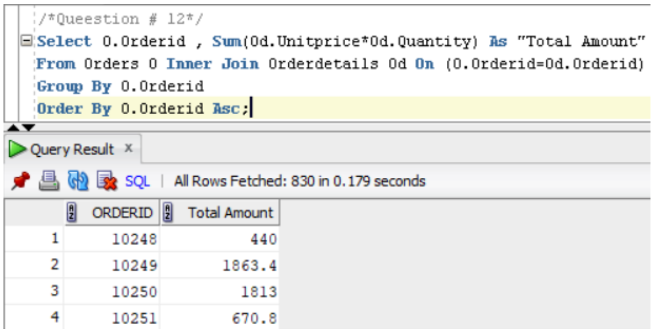


```
/*Queestion # 11*/
Select O.Orderid, O.Orderdate, P.Productname,ct.categoryname
From Orders O Inner Join Orderdetails Od On (O.Orderid =Od.Orderid)
Inner Join Products P On (P.Productid = Od.Productid)
Inner Join Categories ct On (P.Categoryid=Ct.Categoryid)
Where Ct.Categoryname != 'Dairy Products' And Ct.Categoryname != 'Meat/Poultry';
```

All Rows Fetched: 1616 in 0.358 seconds

	ORDERID	ORDERDATE	PRODUCTNAME	CATEGORYNAME
1	10579	25-JUN-97	RhÃ¶nbrÃ¤u Klosterbier	Beverages
2	10580	26-JUN-97	Tofu	Produce
3	10580	26-JUN-97	Jack's New England Clam Chowder	Seafood

12. 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.]*

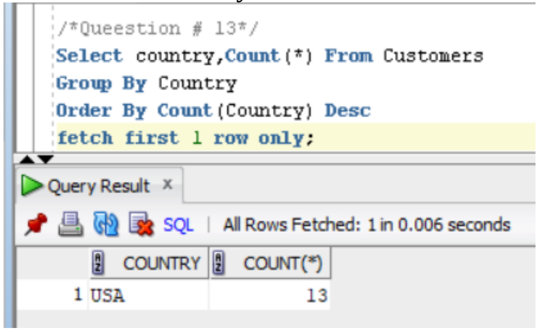


```
/*Queestion # 12*/
Select O.Orderid , Sum(Od.Unitprice*Od.Quantity) As "Total Amount"
From Orders O Inner Join Orderdetails Od On (O.Orderid=Od.Orderid)
Group By O.Orderid
Order By O.Orderid Asc;
```

All Rows Fetched: 830 in 0.179 seconds

	ORDERID	Total Amount
1	10248	440
2	10249	1863.4
3	10250	1813
4	10251	670.8

13. Find country to which maximum of customers belong.



```
/*Queestion # 13*/
Select country,Count(*) From Customers
Group By Country
Order By Count(Country) Desc
fetch first 1 row only;
```

All Rows Fetched: 1 in 0.006 seconds

	COUNTRY	COUNT(*)
1	USA	13