| Document Name | Converting Linq Assignment |
|---|---|
| Version | 1.0 |

| Ver. No | Created by | Author Designation |
|---|---|---|
| 1.0 | Mr. Sudhakar Verma | Senior Software Developer |

# Assignment Queries through Linq in C#

## Products

### 1) Display name of products which are not sold by employee Peter.

Solution:-

```
using System;

using System.Linq;


namespace Queries
{
    class Query1
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 1.1 \nDisplay name of products which are not sold by
employee Peter ");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            var empid = from emp in dbe.employees where emp.emp_name=="Peter" select
emp.emp_id;
            var orderid = from ordermas in dbe.ordermasters where
empid.Contains(ordermas.emp_id) select ordermas.order_id;
            var productid = from orderdtl in dbe.orderdetails where
orderid.Contains(orderdtl.order_id) select orderdtl.product_id;
            var query = from pro in dbe.products where
!productid.Contains(pro.product_id) select pro;
```

```
            foreach (var product in query)
            {
                Console.WriteLine("Product id="+product.product_id+"\t Product Name=" +
product.product_name);
            }

            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("-------------------------------------------------------------
--------------");

            empid = dbe.employees.Where(emp=>emp.emp_name=="Peter").Select(emp =>
emp.emp_id);
            orderid = dbe.ordermasters.Where(ordermas =>
empid.Contains(ordermas.emp_id)).Select(ordermas => ordermas.order_id);
            productid = dbe.orderdetails.Where(orderdtls =>
orderid.Contains(orderdtls.order_id)).Select(orderdtls => orderdtls.product_id);
            query = dbe.products.Where(pro =>
!productid.Contains(pro.product_id)).Select(pro => pro);
            foreach (var product in query)
            {
                //Console.WriteLine("emp id=" + product);
                Console.WriteLine("Product id=" + product.product_id + "\t Product Name="
+ product.product_name);
            }


            Console.WriteLine("Solution using Query Expression  using Combination into
one Query");
            Console.WriteLine("-------------------------------------------------------------
--------------");


            query = dbe.products.Where(pro => !(dbe.orderdetails.Where(orderdtls =>
            dbe.ordermasters.Where(ordermas =>
            dbe.employees.Where(emp => emp.emp_name == "Peter").Select(emp => emp.emp_id)
            .Contains(ordermas.emp_id)).Select(ordermas => ordermas.order_id)
            .Contains(orderdtls.order_id)).Select(orderdtls => orderdtls.product_id))
            .Contains(pro.product_id)).Select(pro => pro);
            foreach (var product in query)
            {
                //Console.WriteLine("emp id=" + product);
                Console.WriteLine("Product id=" + product.product_id + "\t Product Name="
+ product.product_name);
            }
            Console.ReadKey();
        }
    }
}
```

## 2) Display name of products which are not purchased by customer Smith.

```
/*
2) Display name of products which are not purchased by customer Smith.
    */
using Queries;
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query2
    {

            public static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();


            /*

             Select product_name from product where product_id not in
                 (Select product_id from orderdetail where order_id in
                 (Select order_id from ordermaster where customer_id in
                 (Select customer_id from customer where customer_name like 'Smith')
                 )
             )
         */

            Console.WriteLine("Q 1.2 \nDisplay name of products which are not purchased
by customer Smith. ");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            var custid = from cust in dbe.customers where cust.customer_name == "Smith"
select cust.customer_id;
            var orderid = from ordermas in dbe.ordermasters where
custid.Contains(ordermas.customer_id) select ordermas.order_id;
            var productid = from orderdtl in dbe.orderdetails where
orderid.Contains(orderdtl.order_id) select orderdtl.product_id;
            var query = from pro in dbe.products where
!productid.Contains(pro.product_id) select pro;
            foreach (var product in query)
            {
                Console.WriteLine("Product id=" + product.product_id + "\t Product Name="
+ product.product_name);
            }

            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("-----------------------------------------------------------
--------------");

            custid = dbe.customers.Where(cust => cust.customer_name ==
"Smith").Select(cust => cust.customer_id);
            orderid = dbe.ordermasters.Where(ordermas =>
custid.Contains(ordermas.customer_id)).Select(ordermas => ordermas.order_id);
            productid = dbe.orderdetails.Where(orderdtls =>
orderid.Contains(orderdtls.order_id)).Select(orderdtls => orderdtls.product_id);
```

```
            query = dbe.products.Where(pro =>
!productid.Contains(pro.product_id)).Select(pro => pro);
            foreach (var product in query)
            {
                //Console.WriteLine("emp id=" + product);
                Console.WriteLine("Product id=" + product.product_id + "\t Product Name="
+ product.product_name);
            }
            Console.ReadKey();


        }
    }
}
```

## 3) Display name of products which are purchased individually.

```
/*
Q 1
3) Display name of products which are purchased individually.
*/
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query3
    {
        public static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();


            /*

            Select Product_name from product where Product_id in
            (Select product_id from orderdetail where order_id in
            (Select order_id  from orderdetail group by order_id having count(*)=1));


         */

            Console.WriteLine("Q 1.3 \nDisplay name of products which are purchased
individually. ");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            var orderid = from ordrtl in dbe.orderdetails group ordrtl by ordrtl.order_id
into g where g.Count()==1 select  g.Key;
            var productid = from ordrtl in dbe.orderdetails where
orderid.Contains(ordrtl.order_id) select ordrtl.product_id;
```

```
            var productname = from pro in dbe.products where
productid.Contains(pro.product_id) select pro.product_name;

            int count = 1;
            foreach (var product in productname)
            {

                Console.WriteLine("Row No="+count+"\tProduct Name=" + product);
                count++;

            }
            // Console.WriteLine("Count==" + count);

            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("------------------------------------------------------------
--------------");

             orderid =dbe.orderdetails.GroupBy(ordrtl=>
ordrtl.order_id).Where(g=>g.Count()==1).Select(g=>g.Key);
             productid = dbe.orderdetails.Where(ordrtl =>
orderid.Contains(ordrtl.order_id)).Select(ordrtl => ordrtl.product_id);
             productname = dbe.products.Where(pro =>
productid.Contains(pro.product_id)).Select(pro => pro.product_name);
             count = 1;
            foreach (var product in productname)
            {

                Console.WriteLine("Row No="+count+"\tProduct Name=" + product);
                count++;

            }

            Console.ReadKey();


        }
    }
}
```

## 4) Display name of products which are purchased by maximum number of customers.

Solution:

```
/*
4) Display name of products which are purchased by maximum number of customers.
*/
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query4
```

```
    {
        public static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();


            /*

            Select product_name from product where product_id in (Select product_id from
(Select product_id,Count(*)over(partition by orderdetail.product_id order by product_id)
as NoOFProductID from Ordermaster inner join orderdetail on
ordermaster.order_id=orderdetail.order_id
            )as t1 where t1.NoOFProductID in(Select  max(t.NoOFProductID) as maximum from
(Select product_id,Count(*)over(partition by orderdetail.product_id order by product_id)
as NoOFProductID from Ordermaster inner join orderdetail on
ordermaster.order_id=orderdetail.order_id
            )as t));



         */

            Console.WriteLine("Q 1.4 \nDisplay name of products which are purchased by
maximum number of customers. ");
            Console.WriteLine("------------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("------------------------------------------------------------
--------------");
            ////////////////////////////
            var grpquery = from or in dbe.orderdetails
                           group or by or.product_id into g
                           select new
                           {

                               productid = g.Key,
                               noofproduct = g.Count()
                           };
            var maximumquery = (from order in (grpquery)
                                select order).Max(maximum => maximum.noofproduct);
            var allmaximumquery = from allqry in (grpquery) where allqry.noofproduct ==
maximumquery select allqry.productid;
            //Console.WriteLine("Maximum is=" + maximumquery);
            var resultquery = from pro in dbe.products where
allmaximumquery.Contains(pro.product_id) select pro.product_name;
            int count = 1;
            foreach (var product_name in resultquery)
            {
                Console.WriteLine("Row No=" + count + "\tProduct Name=" + product_name);
                count++;
            }
            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("------------------------------------------------------------
--------------");
            grpquery = /*from or in dbe.orderdetails
                          group or by or.product_id into g
                          select new
                          {
```

```
                                productid = g.Key,
                                noofproduct = g.Count()
                            };*/
                             dbe.orderdetails.GroupBy(or => or.product_id).Select(g => new
{ productid = g.Key, noofproduct = g.Count() });



            maximumquery = /*(from order in (grpquery)
                                 select order).Max(maximum => maximum.noofproduct);
                                 */
                                 (grpquery.Select(order => order).Max(maximum =>
maximum.noofproduct));

            allmaximumquery = /*
                from allqry in (grpquery) where allqry.noofproduct == maximumquery select
allqry.productid;
                                 */
                                grpquery.Where(allqry => allqry.noofproduct ==
maximumquery).Select(allqry => allqry.productid);
            //Console.WriteLine("Maximum is=" + maximumquery);
            resultquery = /*
                from pro in dbe.products where allmaximumquery.Contains(pro.product_id)
select pro.product_name;
                */
               dbe.products.Where(pro =>
allmaximumquery.Contains(pro.product_id)).Select(pro => pro.product_name);
             count = 1;
            foreach (var product_name in resultquery)
            {
                Console.WriteLine("Row No=" + count + "\tProduct Name=" + product_name);
                count++;
            }

            Console.ReadKey();


        }
    }
}
```

## 5) Display name of products which are sold by employees whose manager is Michael.

Solution:

```
/*
5) Display name of products which are sold by employees whose manager is Michael
*/
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Query1
{
    class Query5
    {
        public static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();


            /*

           Select product_name from product where product_id in (

Select product_id from orderdetail where order_id in ( Select order_id from ordermaster
where emp_id in
( Select e1.emp_id from employee e1  inner join employee e2 on e1.emp_manager_id in
(Select emp_id from employee where emp_name like 'Michael')
)
))




         */

            Console.WriteLine("Q 1.4 \n Display name of products which are sold by
employees whose manager is Michael ");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            ////////////////////////////
            var managerid = from emp in dbe.employees
                            where emp.emp_name.Contains("Michael")
                            select  emp.emp_id;


            var empid = from e1 in dbe.employees
                        where managerid.Contains(e1.emp_manager_id)
                        select e1.emp_id;
            var orderid = from ordermas in dbe.ordermasters where
empid.Contains(ordermas.emp_id) select ordermas.order_id;
            var productid = from order in dbe.orderdetails where
orderid.Contains(order.order_id) select order.product_id;
            var productname = from product in dbe.products where
productid.Contains(product.product_id) select product.product_name;
            int count = 1;
            foreach (var product_name in productname)
            {
                Console.WriteLine("Row No=" + count + "\tProduct Name=" + product_name);
                count++;
            }
            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            managerid = dbe.employees.Where(emp =>
emp.emp_name.Contains("Michael")).Select(emp => emp.emp_id);
```

```
            empid = dbe.employees.Where(emp =>
managerid.Contains(emp.emp_manager_id)).Select(emp => emp.emp_id);
            orderid = dbe.ordermasters.Where(ordermaster =>
empid.Contains(ordermaster.emp_id)).Select(ordermaster => ordermaster.order_id);
            productid =  dbe.orderdetails.Where(orderdetail =>
orderid.Contains(orderdetail.order_id)).Select(orderdetail => orderdetail.product_id);
            productname = dbe.products.Where(product =>
productid.Contains(product.product_id)).Select(product => product.product_name);
            count = 1;
            foreach (var product_name in productname)
            {
                Console.WriteLine("Row No=" + count + "\tProduct Name=" + product_name);
                count++;
            }

            Console.ReadKey();


        }
    }
}
```

## 6) Display name of products which are not purchased by any customer from last 4 months.

Solution:-

```
/*
6) Display name of products which are not purchased by any customer from last 4 months.
*/
using Queries;
using System;
using System.Collections.Generic;
using System.Data.Entity.Core.Objects;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query6
    {

        public static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();


            /*

            Select product_name from product where product_id in (Select product_id from
(Select product_id,Count(*)over(partition by orderdetail.product_id order by product_id)
as NoOFProductID from Ordermaster inner join orderdetail on
ordermaster.order_id=orderdetail.order_id
```

```
            )as t1 where t1.NoOFProductID in(Select  max(t.NoOFProductID) as maximum from
(Select product_id,Count(*)over(partition by orderdetail.product_id order by product_id)
as NoOFProductID from Ordermaster inner join orderdetail on
ordermaster.order_id=orderdetail.order_id
            )as t));



         */

            Console.WriteLine("Q 1.6 \n Display name of products which are not purchased
by any customer from last 4 months.   ");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            ////////////////////////////

            DateTime now = DateTime.Today;
            Console.WriteLine("date time is =" + now);
            Console.WriteLine("date time is =" + now.AddMonths(-4));
            DateTime date = now.AddMonths(-4);

            /*
            Select* from ordermaster where Order_date > (Select DATEADD(MONTH, -4,
GETDATE()) as dateAdd)
            */

             //var orderid = from ordermaster in dbe.ordermasters where
ordermaster.Order_date > now.AddMonths(-4) select ordermaster.order_id;

            var order_id =dbe.ordermasters.ToList().Where(ordermaster =>
ordermaster.Order_date > now.AddMonths(-4)).Select(ordermaster => ordermaster.order_id);
            var product_id = from orderdetail in dbe.orderdetails where
order_id.Contains(orderdetail.order_id) select orderdetail.product_id;
            var product_name = from product in dbe.products where
!product_id.Contains(product.product_id) select product.product_name;
            int    count = 1;
            foreach (var prod_name in product_name)
            {
                Console.WriteLine("Row No=" + count + "\t Product Name=" + prod_name);
                count++;
            }
            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("-----------------------------------------------------------
--------------");
             order_id = dbe.ordermasters.ToList().Where(ordermaster =>
ordermaster.Order_date > now.AddMonths(-4)).Select(ordermaster => ordermaster.order_id);
            product_id = dbe.orderdetails.Where(orderdetail =>
order_id.Contains(orderdetail.order_id)).Select(orderdetail=> orderdetail.product_id);
            //from orderdetail in dbe.orderdetails where
order_id.Contains(orderdetail.order_id) select orderdetail.product_id;
            product_name = dbe.products.Where(product =>
!product_id.Contains(product.product_id)).Select(product => product.product_name);
                //from product in dbe.products where
!product_id.Contains(product.product_id) select product.product_name;
             count = 1;
```

```
            foreach (var prod_name in product_name)
            {
                Console.WriteLine("Row No=" + count + "\t Product Name=" + prod_name);
                count++;
            }
            Console.ReadKey();


        }
    }
}
```

## 7) Display name of products which are sold maximum in months June and July.

Solution:

```
/*
Q 1.7 \n Display name of products which are sold maximum in months June and July.
*/
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query7
    {
        public static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();


            /*

            Select product_name from product where product_id in (Select product_id from
(Select product_id,Count(*)over(partition by orderdetail.product_id order by product_id)
as NoOFProductID from Ordermaster inner join orderdetail on
ordermaster.order_id=orderdetail.order_id
            )as t1 where t1.NoOFProductID in(Select  max(t.NoOFProductID) as maximum from
(Select product_id,Count(*)over(partition by orderdetail.product_id order by product_id)
as NoOFProductID from Ordermaster inner join orderdetail on
ordermaster.order_id=orderdetail.order_id
            )as t));



         */

            Console.WriteLine("Q 1.7 \n Display name of products which are sold maximum
in months June and July.   ");
```

```
            Console.WriteLine("------------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("------------------------------------------------------------
--------------");
            ////////////////////////////



            /* Select Order_id from ordermaster where MONTH(Order_date) in ('6','7')*/
            var list = (from ordermaster in dbe.ordermasters select new { Order_id =
ordermaster.order_id, OrderDate = ordermaster.Order_date }).ToList();
            var order_id = from lst in list where
Convert.ToDateTime(lst.OrderDate.ToString()).Month == 6 ||
Convert.ToDateTime(lst.OrderDate.ToString()).Month == 7 select lst.Order_id;
            /*  Select product_id,quantity from orderdetail where order_id  in  (
                Select Order_id from ordermaster where MONTH(Order_date) in ('6','7'))
        */
            var innerquery = from orderdetail in dbe.orderdetails where
order_id.Contains(orderdetail.order_id) select new { product_id = orderdetail.product_id,
quantity = orderdetail.quantity };




            /*  Select product_id,sum(quantity)as quantity from (Select
product_id,quantity from orderdetail where order_id  in  (
    Select Order_id from ordermaster where MONTH(Order_date) in ('6','7'))) as t1 group
by product_id
        */
            var productid_sumquantity = from t1 in innerquery
                                        group t1 by t1.product_id into g
                                        select new
                                        {
                                            product_id = g.Key,
                                            quantity = g.Sum(C=>C.quantity)

                                        };






            /*Select Max(quantity) from (
  Select product_id,sum(quantity)as quantity from (Select product_id,quantity from
orderdetail where order_id  in  (
  Select Order_id from ordermaster where MONTH(Order_date) in ('6','7'))) as t1 group by
product_id)as t2
*/
            var maximumvalue = (from t2 in productid_sumquantity select
t2.quantity).Max();
            var product_id = from t3 in productid_sumquantity where t3.quantity ==
maximumvalue select t3.product_id;
            var product_name = from product in dbe.products where
product_id.Contains(product.product_id) select product.product_name;
            int count = 1;
            foreach (var prod_name in product_name)
```

```
            {
                Console.WriteLine("Row No=" + count + "\t Product Name=" + prod_name);
                count++;
            }

            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("-------------------------------------------------------------
--------------");


            list = dbe.ordermasters.Select(ordermaster => new { Order_id =
ordermaster.order_id, OrderDate = ordermaster.Order_date }).ToList();
            //(from ordermaster in dbe.ordermasters select new { Order_id =
ordermaster.order_id, OrderDate = ordermaster.Order_date }).ToList();
            order_id = list.Where(lst => Convert.ToDateTime(lst.OrderDate).Month == 6 ||
Convert.ToDateTime(lst.OrderDate).Month == 7).Select(lst => lst.Order_id);
            //from lst in list where Convert.ToDateTime(lst.OrderDate.ToString()).Month
== 6 || Convert.ToDateTime(lst.OrderDate.ToString()).Month == 7 select lst.Order_id;

            innerquery = dbe.orderdetails.Where(orderdetail =>
order_id.Contains(orderdetail.order_id)).Select(orderdetail => new { product_id =
orderdetail.product_id, quantity = orderdetail.quantity });
            //from orderdetail in dbe.orderdetails where
order_id.Contains(orderdetail.order_id) select new { product_id = orderdetail.product_id,
quantity = orderdetail.quantity };




            productid_sumquantity = innerquery.GroupBy(t1 => t1.product_id).Select(g =>
new
            {
                product_id = g.Key,
                quantity = g.Sum(C => C.quantity)

            });
            /*from t1 in innerquery
                                group t1 by t1.product_id into g
                                select new
                                {
                                    product_id = g.Key,
                                    quantity = g.Sum(C => C.quantity)

                                };*/
            maximumvalue = productid_sumquantity.Select(pro => pro.quantity).Max();

            //(from t2 in productid_sumquantity select t2.quantity).Max();
            product_id = productid_sumquantity.Where(t3 => t3.quantity ==
maximumvalue).Select(t3 => t3.product_id);
            //from t3 in productid_sumquantity where t3.quantity == maximumvalue select
t3.product_id;
            product_name =
dbe.products.Where(product=>product_id.Contains(product.product_id)).Select(product=>prod
uct.product_name);
                //from product in dbe.products where
product_id.Contains(product.product_id) select product.product_name;
             count = 1;
```

```
            foreach (var prod_name in product_name)
            {
                Console.WriteLine("Row No=" + count + "\t Product Name=" + prod_name);
                count++;
            }
            Console.ReadKey();


        }
    }
}
```

## 8) Display name of top five products which are in high demand in all 12 months.

```
/*
1.8) Display name of top five products which are in high demand in all 12 months.
*/
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query8
    {
        public static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();


            /*

            Select product_name from product where product_id in (Select product_id from
(Select product_id,Count(*)over(partition by orderdetail.product_id order by product_id)
as NoOFProductID from Ordermaster inner join orderdetail on
ordermaster.order_id=orderdetail.order_id
            )as t1 where t1.NoOFProductID in(Select  max(t.NoOFProductID) as maximum from
(Select product_id,Count(*)over(partition by orderdetail.product_id order by product_id)
as NoOFProductID from Ordermaster inner join orderdetail on
ordermaster.order_id=orderdetail.order_id
            )as t));



         */

            Console.WriteLine("Q 1.8 \n  Display name of top five products which are in
high demand in all 12 months.   ");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-------------------------------------------------------------
--------------");
```

```
            ////////////////////////////////


            /* (Select Product_Id,quantity from OrderDetail where Order_Id in
(Select Order_Id from OrderMaster where
DatePart(year,Order_Date)=DATEPART(year,getdate())-1))*/
            var list = from lst in (from ordermaster in dbe.ordermasters select
ordermaster).ToList() where
Convert.ToDateTime(lst.Order_date.ToString()).Year==DateTime.Today.Year-1 select
lst.order_id;




            //(from ordermaster in dbe.ordermasters select new { Order_id =
ordermaster.order_id, OrderDate = ordermaster.Order_date }).ToList();
            // var order_id = from lst in list where
Convert.ToDateTime(lst.OrderDate.ToString()).Month == 6 ||
Convert.ToDateTime(lst.OrderDate.ToString()).Month == 7 select lst.Order_id;
            /*  Select product_id,quantity from orderdetail where order_id  in  (
                Select Order_id from ordermaster where MONTH(Order_date) in ('6','7'))
        */
            var innerquery = from orderdetail in dbe.orderdetails where
(list).Contains(orderdetail.order_id) select new { Product_Id = orderdetail.product_id,
quantity = orderdetail.quantity };






            /* select product.product_name,Quantity=sum(pro.quantity) from pro inner join
product
on pro.product_id=product.product_id
group by product.product_name)n
        */
           var productid_sumquantity = from pro in innerquery join product in
dbe.products on pro.Product_Id equals product.product_id
                                        group pro by product.product_name into g
                                        select new
                                        {
                                            product_name = g.Key,
                                            quantity = g.Sum(C => C.quantity)

                                        };
            /* select top(5)* from
(select product.product_name,Quantity=sum(pro.quantity) from pro inner join product
on pro.product_id=product.product_id
group by product.product_name)n order by n.Quantity desc
*/
            var resultquery = (from n in productid_sumquantity orderby n.quantity
descending select n).Take(5);

            int count = 1;
            foreach (var prod_name in resultquery)
            {
```

```
                Console.WriteLine("Row No=" + count + "\t Product Name=" +
prod_name.product_name+"\t\tQuantity="+prod_name.quantity);
                count++;
            }

            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("-----------------------------------------------------------
--------------");

            list = dbe.ordermasters.Select(ordermaster => ordermaster).ToList().Where(lst
=> Convert.ToDateTime(lst.Order_date.ToString()).Year == DateTime.Today.Year -
1).Select(lst => lst.order_id);
            // from lst in (from ordermaster in dbe.ordermasters select
ordermaster).ToList() where Convert.ToDateTime(lst.Order_date.ToString()).Year ==
DateTime.Today.Year - 1 select lst.order_id;




            innerquery = dbe.orderdetails.Where(orderdetail =>
(list).Contains(orderdetail.order_id)).Select(orderdetail => new { Product_Id =
orderdetail.product_id, quantity = orderdetail.quantity });

            //from orderdetail in dbe.orderdetails where
(list).Contains(orderdetail.order_id) select ;




            productid_sumquantity = from pro in innerquery
                                    join product in dbe.products on pro.Product_Id
equals product.product_id
                                    group pro by product.product_name into g
                                    select new
                                    {
                                        product_name = g.Key,
                                        quantity = g.Sum(C => C.quantity)

                                    };

            resultquery = productid_sumquantity.OrderByDescending(x =>
x.quantity).Select(x => x).Take(5);
                //(from n in productid_sumquantity orderby n.quantity descending select
n).Take(5);
                count = 1;
            foreach (var prod_name in resultquery)
            {
                Console.WriteLine("Row No=" + count + "\t Product Name=" +
prod_name.product_name + "\t\tQuantity=" + prod_name.quantity);
                count++;
            }

            Console.ReadKey();
```

```
        }
    }
}
```

## 9) Display name of products which are purchased in all 12 months.

```
/*
1.9) Display name of products which are purchased in all 12 months.
*/
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query9
    {
        public static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();


            /*

         */

            Console.WriteLine("Q 1.9 \n  Display name of products which are purchased in
all 12 months.   ");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            ////////////////////////////



            /* select Orderdetail.product_id,ordermaster.Order_Date
 from ordermaster inner join orderdetail on ordermaster.order_Id=orderdetail.order_id
 where DATEDIFF(Year,ordermaster.Order_date,GETDATE())=1*/
            var joinquery = from lst in (from ordermaster in dbe.ordermasters
                                         join orderdetail in dbe.orderdetails on
ordermaster.order_id equals orderdetail.order_id
                                         select new { product_id =
orderdetail.product_id, Order_date = ordermaster.Order_date }).ToList()
                            where Convert.ToDateTime(lst.Order_date.ToString()).Year -
DateTime.Today.Year == 1|| Convert.ToDateTime(lst.Order_date.ToString()).Year -
DateTime.Today.Year == -1
                            select new { product_id = lst.product_id, Order_date =
lst.Order_date };

            var groupquery = from joins in joinquery
                                        group joins by joins.product_id into g
                                        select new
```

```
                                        {
                                            product_id = g.Key,
                                            Count1 = g.Select(x=>x.Order_date).Count()

                                        };
            var finalquery = from pro in groupquery
                             join product in dbe.products on pro.product_id equals
product.product_id
                             where pro.Count1==12//1
                             select new
                             {
                                 pro.product_id,
                                 pro.Count1,
                                 product.product_name,
                                 product.product_rate
                             };




            int count = 1;
            foreach (var prod_name in finalquery)
            {
                Console.WriteLine("Row No=" + count + "\t Product Name=" + prod_name);
                count++;
            }
            if (count == 1)
                Console.WriteLine("----------------Record Not Found----------");
          //  Console.WriteLine("Group query ------------------------------");

            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("------------------------------------------------------------
--------------");
            joinquery = dbe.ordermasters.Join(dbe.orderdetails, ordermaster =>
ordermaster.order_id, orderdetail => orderdetail.order_id, (ordermaster, orderdetail) =>
new { ordermaster, orderdetail })
               .Select(x =>
               new
               {
                   product_id = x.orderdetail.product_id,
                   Order_date = x.ordermaster.Order_date
               }
               ).ToList().Where(lst => Convert.ToDateTime(lst.Order_date.ToString()).Year
- DateTime.Today.Year == 1 || Convert.ToDateTime(lst.Order_date.ToString()).Year -
DateTime.Today.Year == -1).
               Select(lst => new
               {
                   product_id = lst.product_id,
                   Order_date = lst.Order_date
               });
            /* from lst in (from ordermaster in dbe.ordermasters
                                    join orderdetail in dbe.orderdetails on
ordermaster.order_id equals orderdetail.order_id
                                    select new { product_id = orderdetail.product_id,
Order_date = ordermaster.Order_date }).ToList()
```

```
                        where Convert.ToDateTime(lst.Order_date.ToString()).Year -
DateTime.Today.Year == 1 || Convert.ToDateTime(lst.Order_date.ToString()).Year -
DateTime.Today.Year == -1
                        select new { product_id = lst.product_id, Order_date =
lst.Order_date };
                        */
            groupquery = joinquery.GroupBy(joins => joins.product_id).Select(g => new
            {
                product_id = g.Key,
                Count1 = g.Select(x => x.Order_date).Count()
            });
            /*from joins in joinquery
                        group joins by joins.product_id into g
                        select new
                        {
                            product_id = g.Key,
                            Count1 = g.Select(x => x.Order_date).Count()

                        };*/
            finalquery = groupquery.Join(dbe.products, pro => pro.product_id, product =>
product.product_id, (pro, product) => new { pro, product })
                .Where(x => x.pro.Count1 == 12)//1)
               .Select(x => new
               {
                   x.pro.product_id,
                   x.pro.Count1,
                   x.product.product_name,
                   x.product.product_rate
               });
                /*from pro in groupquery
                            join product in dbe.products on pro.product_id equals
product.product_id
                            where pro.Count1 == 12//1
                            select new
                            {
                                pro.product_id,
                                pro.Count1,
                                product.product_name,
                                product.product_rate
                            };
                            */




            count = 1;
            foreach (var prod_name in finalquery)
            {
                Console.WriteLine("Row No=" + count + "\t Product Name=" + prod_name);
                count++;
            }
            if (count == 1)
            {
                Console.WriteLine("------------------------------Record Not Found-------
------------------------------");
            }
```

```
            Console.ReadKey();


        }

    }
}
```

## *10) Display name of products which are purchased only once but in all 12 months*

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query10
    {
        public static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();


            /*

         */

            Console.WriteLine("Q 1.9 \n Display name of products which are purchased only
once but in all 12 months  ");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            ////////////////////////////



            /* with pro as
                (
                select Orderdetail.product_id from ordermaster inner join
                 orderdetail on ordermaster.order_Id=orderdetail.order_id
                group by Orderdetail.product_id having
                COUNT(DATEPART(mm,ordermaster.Order_Date))=12
                and COUNT(Distinct(DATEPART(mm,ordermaster.Order_Date)))=12
                )select Product.product_name from pro inner join Product on
                 pro.product_id=Product.product_id

            */
            var joinquery1 = (from ordermaster in dbe.ordermasters
```

```
                                    join orderdetail in dbe.orderdetails on
ordermaster.order_id equals orderdetail.order_id
                                    select new { product_id =
orderdetail.product_id, Order_date = ordermaster.Order_date });

            var groupquery = from joins in joinquery1
                             group joins by joins.product_id into g
                             select new
                             {
                                 product_id = g.Key,
                                 Count1 = g.Select(x => x.Order_date).Count(),
                                 count2=g.Select(x=>x.Order_date).Distinct().Count()
                             };
            var finalquery = from pro in groupquery
                             join product in dbe.products on pro.product_id equals
product.product_id
                             where pro.Count1 == 12//1
                             && pro.count2==12
                             select new
                             {
                                 pro.product_id,
                                 pro.Count1,
                                 product.product_name,
                                 product.product_rate
                             };




            int count = 1;
            foreach (var prod_name in finalquery)
            {
                Console.WriteLine("Row No=" + count + "\t Product Name=" + prod_name);
                count++;
            }
            if (count == 1)
                Console.WriteLine("----------------Record Not Found----------");
            //  Console.WriteLine("Group query ------------------------------");

            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("------------------------------------------------------------
--------------");
            var joinquery = dbe.ordermasters.Join(dbe.orderdetails, ordermaster =>
ordermaster.order_id, orderdetail => orderdetail.order_id, (ordermaster, orderdetail) =>
new { ordermaster, orderdetail })
               .Select(x =>
               new
               {
                   product_id = x.orderdetail.product_id,
                   Order_date = x.ordermaster.Order_date
               }
               );
                groupquery = joinquery.GroupBy(joins => joins.product_id).Select(g => new
            {
                product_id = g.Key,
                Count1 = g.Select(x => x.Order_date).Count(),
```

```
                count2=g.Select(x=>x.Order_date).Distinct().Count()
               });
             finalquery = groupquery.Join(dbe.products, pro => pro.product_id, product
=> product.product_id, (pro, product) => new { pro, product })
              .Where(x => x.pro.Count1 == 12 && x.pro.count2 == 12)//1)
             .Select(x => new
             {
                 x.pro.product_id,
                 x.pro.Count1,
                 x.product.product_name,
                 x.product.product_rate
             });



            count = 1;
            foreach (var prod_name in finalquery)
            {
                Console.WriteLine("Row No=" + count + "\t Product Name=" + prod_name);
                count++;
            }
            if (count == 1)
            {
                Console.WriteLine("-------------------------------Record Not Found-------
-----------------------------");
            }

            Console.ReadKey();


        }

    }
}
```

# Employees Related Queries

## 1) Display name of employees which have only two a's.

Solution:

```
/*2.1) Display name of employees which have only two a's.*/
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query2_1
```

```
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 2.1 \n1) Display name of employees which have only two
a's.");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            /* Select emp_name from (Select (len(emp_name)-len(REPLACE(emp_name,'a','')))
as totalA,emp_name from employee )as t where totalA='2';*/
            var emp_name = from emplist in (from emp in dbe.employees select new { totalA
= emp.emp_name.Length - emp.emp_name.Replace("a", "").Length, emp_name = emp.emp_name })
where emplist.totalA == 2 select emplist.emp_name;

            //Console.WriteLine("Emp Name=" + emp_name);
            foreach (var emp_name1 in emp_name)
            {
                Console.WriteLine("Emp Name=" + emp_name1);
            }

            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("-----------------------------------------------------------
--------------");

            var empname = (dbe.employees.Select(emp => new
            {
                totalA = emp.emp_name.Length - emp.emp_name.Replace("a", "").Length,
                emp_name = emp.emp_name
            })).Where(x=>x.totalA==2).Select(x=>x.emp_name);
            //from emplist in (from emp in dbe.employees select new { totalA =
emp.emp_name.Length - emp.emp_name.Replace("a", "").Length, emp_name = emp.emp_name })
where emplist.totalA == 2 select emplist.emp_name;

            foreach (var emp_name1 in empname)
            {
                Console.WriteLine("Emp Name=" + emp_name1);
            }



            Console.ReadKey();
        }
    }
}
```

## 2) Display name of employees in ascending order according to last two characters of each name

Solution:

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
/*
2.2) Display name of employees in ascending order according to last two characters of
each name.
*/
namespace Query1
{
    class Query2_2
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 2.2 \n1)  Display name of employees in ascending order
according to last two characters of each name.");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            /*
            Select * from (select emp_name,Right(emp_name,2)as twochacter from employee)
as t  order by twochacter;
            */
            var emp_name = from emplist in (from emp in dbe.employees select new {
twochacter = emp.emp_name.Substring(emp.emp_name.Length-2), emp_name = emp.emp_name })
orderby emplist.twochacter  select emplist.emp_name;

            //Console.WriteLine("Emp Name=" + emp_name);
            int count = 1;
            foreach (var emp_name1 in emp_name)
            {
                Console.WriteLine("Row="+count+"\tEmp Name=" + emp_name1);
                count++;
            }

            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("-------------------------------------------------------------
--------------");

            var empname = (dbe.employees.Select(emp => new
            {
                twochacter = emp.emp_name.Substring(emp.emp_name.Length - 2),
                emp_name = emp.emp_name
            })).OrderBy(x=>x.twochacter).Select(x => x.emp_name);
            //from emplist in (from emp in dbe.employees select new { totalA =
emp.emp_name.Length - emp.emp_name.Replace("a", "").Length, emp_name = emp.emp_name })
where emplist.totalA == 2 select emplist.emp_name;
            count = 1;
            foreach (var emp_name1 in emp_name)
```

```
            {
                Console.WriteLine("Row=" + count + "\tEmp Name=" + emp_name1);
                count++;
            }



            Console.ReadKey();
        }
    }
}
```

## 3) Display name of employees who attended maximum number of customers in last month.

Solution:

```
using Queries;
using System;
using System.Collections.Generic;
using System.Data.Entity.Core.Objects;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
/*3) Display name of employees who attended maximum number of customers in last month.*/
namespace Query1
{
    class Query2_3
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 2.3 \n1)  Display name of employees who attended maximum
number of customers in last month.");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            /*
          ---Right Query
        with emp as
        (
        select Employee.Emp_name,Ordermaster.emp_id from
         Employee inner join ordermaster on employee.emp_id=ordermaster.emp_id
        where DATEDIFF(mm,ordermaster.Order_date,GETDATE())=3
        )select emp_name from emp group by
         emp_id,emp_name having count(emp_id)=
         (select top(1)COUNT(emp_id) from emp group by emp_id order by COUNT(emp_id)
desc)

            */
            DateTime dt = DateTime.Today;
```

```
            var joinquery = from lst in (from employee in dbe.employees
                                         join ordermaster in dbe.ordermasters on
employee.emp_id equals ordermaster.emp_id
                                         select new { emp_name = employee.emp_name,
emp_id = ordermaster.emp_id, order_date = ordermaster.Order_date })
                            where EntityFunctions.DiffMonths(lst.order_date, dt) == 3


                            select lst;

            var maximun = (from joinqry in joinquery group joinqry by joinqry.emp_id into
g select  g.Count()).Max();
            var emp_id =(from lst in (from joinqry in joinquery group joinqry by
joinqry.emp_id into g select new {emp_id= g.Key, count=g.Count() }).ToList() where
lst.count==maximun select lst.emp_id);
            var emp_name = from emp in dbe.employees where emp_id.Contains(emp.emp_id)
select emp.emp_name;

            // from joinqry in joinquery group joinqry by new { joinqry.emp_id,
joinqry.emp_name } into g  select new { count=g.Key.emp_id.Count() };

            //Console.WriteLine("Emp Name=" + maximun);
            int count = 1;
            foreach (var emp_name1 in emp_name)
            {
                Console.WriteLine("Row=" + count + "\tEmp Name=" + emp_name1);
                count++;
            }

            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("-----------------------------------------------------------
--------------");

             joinquery = from lst in (from employee in dbe.employees
                                         join ordermaster in dbe.ordermasters on
employee.emp_id equals ordermaster.emp_id
                                         select new { emp_name = employee.emp_name,
emp_id = ordermaster.emp_id, order_date = ordermaster.Order_date })
                            where EntityFunctions.DiffMonths(lst.order_date, dt) == 3


                            select lst;

             maximun = (from joinqry in joinquery group joinqry by joinqry.emp_id into g
select g.Count()).Max();
             emp_id = (from lst in (from joinqry in joinquery group joinqry by
joinqry.emp_id into g select new { emp_id = g.Key, count = g.Count() }).ToList() where
lst.count == maximun select lst.emp_id);
             emp_name = from emp in dbe.employees where emp_id.Contains(emp.emp_id)
select emp.emp_name;
            count = 1;
            foreach (var emp_name1 in emp_name)
            {
                Console.WriteLine("Row=" + count + "\tEmp Name=" + emp_name1);
                count++;
            }
```

```
            Console.ReadKey();
        }
    }
}
```

## 4) Display name of employees who sold maximum number of products in last month.

Solution:

```
using Queries;
using System;
using System.Collections.Generic;
using System.Data.Entity.Core.Objects;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
/*4) Display name of employees who sold maximum number of products in last month.*/
namespace Query1
{
    class Query2_4
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 2.4 \n1) Display name of employees who sold maximum
number of products in last month.");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            /*
          ---Right Query
with pro as
(
Select Employee.emp_id,OrderDetail.quantity from OrderDetail inner join ordermaster
on orderdetail.order_id=ordermaster.order_id inner join Employee on
Employee.emp_id=ordermaster.emp_id where orderdetail.order_id in
(Select Order_Id from OrderMaster where DATEDIFF(month,order_date,getdate())=2)
)
select * from
(select Employee.emp_name,Quantity=SUM(pro.quantity) from pro inner join
Employee on pro.Emp_id=Employee.emp_id
group by Employee.emp_name)n where n.Quantity=
(
select top(1)Quantity=SUM(pro.quantity) from pro inner join Employee on
pro.Emp_id=Employee.emp_id
group by Employee.emp_name order by Quantity desc
)
            */
            DateTime dt = DateTime.Today;
```

```
            var order_id = from ordermaster in dbe.ordermasters where
EntityFunctions.DiffMonths(ordermaster.Order_date, dt) == 3
                           select ordermaster.order_id;
            var joinquery =from orderdetail in dbe.orderdetails
                                       join ordermaster in dbe.ordermasters on
orderdetail.order_id equals ordermaster.order_id
                                       join employee in dbe.employees on
ordermaster.emp_id equals employee.emp_id
                           where  order_id.Contains(orderdetail.order_id)
                                       select new {  emp_id = employee.emp_id,
order_date = ordermaster.Order_date,quantity=orderdetail.quantity };
            var groupquery = from pro in joinquery
                           join employee in dbe.employees on pro.emp_id equals
employee.emp_id
                           group new { pro, employee } by employee.emp_name into g
                           select new { name = g.Key, quantity = g.Sum(x =>
x.pro.quantity) };

           var maximun = (from groupqry in groupquery  select groupqry.quantity).Max();
           var emp_name = from  groupqry in groupquery where groupqry.quantity == maximun
select new { groupqry.name, groupqry.quantity };

            int count = 1;
            foreach (var emp_name1 in groupquery)
            {
                Console.WriteLine("Row=" + count + "\tEmp Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }

            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("-----------------------------------------------------------
--------------");


            Console.ReadKey();
        }
    }
}
```

## 5) Display name of employees who have grade B and having manager belongs to Admin department.

Solution:

```
using System.Linq;
using Queries;
using System;
using System.Collections.Generic;
using System.Data.Entity.Core.Objects;

using System.Text;
using System.Threading.Tasks;
/*5) Display name of employees who have grade B and having manager belongs to Admin
department.*/
```

```
namespace Query1
{
    class Query2_5
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 2.5 \n1) Display name of employees who have grade B and
having manager belongs to Admin department.");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");

            /*
             ---Right Query
             Select * from employee e1  inner join employee e2 on e1.emp_manager_id=e2.emp_id
inner join department d on d.dept_id=e2.dept_id where d.dept_name='admin'
 and e1.emp_salary>=(
 select min_salary from salarygrades where grade='B') and  e1.emp_salary<=(
 select max_Salary from salarygrades where grade='B')

            */


                    var min_salary = (from salarygrade in dbe.salarygrades where
salarygrade.grade == "B" select salarygrade.min_Salary).First();
                    var max_salary = (from salarygrade in dbe.salarygrades where
salarygrade.grade == "B" select salarygrade.max_Salary).First();
            var joinquery = from e1 in dbe.employees join e2 in dbe.employees on
e1.emp_manager_id equals e2.emp_id
                            join department in dbe.departments on e2.dept_id equals
department.dept_id
                            where department.dept_name == "Admin"
                            select new { e1.emp_name, e1.emp_salary };

            var result = joinquery.ToList().Where(x => x.emp_salary >= min_salary &&
x.emp_salary <= max_salary);
                    //Console.WriteLine("Row=" + max_salary.ToList());
                    int count = 1;
                    foreach (var emp_name1 in result)
                    {
                        Console.WriteLine("Row=" + count + "\tEmp Name=" + emp_name1);
                        //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                        count++;
                    }


                    Console.WriteLine("Solution using Lamda Expression and Methods");
                    Console.WriteLine("-----------------------------------------------
----------------------");
```

```
                    min_salary = dbe.salarygrades.Where(salarygrade => salarygrade.grade
== "B").Select(salarygrade => salarygrade.min_Salary).First();
                   max_salary = dbe.salarygrades.Where(salarygrade => salarygrade.grade
== "B").Select(salarygrade => salarygrade.max_Salary).First();

           /* joinquery = dbe.employees.Join(dbe.employees, e1 => e1.emp_id, e2 =>
e2.emp_manager_id, (e1, e2) => new { e1, e2 })
                .Join(dbe.departments, inner => inner.e1.dept_id, department =>
department.dept_id, (inner, department) => new { inner, department })

                .Select(x => new { x.department.dept_id});

                /* from e1 in dbe.employees
                        join e2 in dbe.employees on e1.emp_manager_id equals e2.emp_id
                        join department in dbe.departments on e2.dept_id equals
department.dept_id
                        where department.dept_name == "Admin"
                        select new { e1.emp_name, e1.emp_salary };*/
            count = 1;
                     foreach (var emp_name1 in result)
                     {
                         Console.WriteLine("Row=" + count + "\tEmp Name=" + emp_name1);
                         //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                         count++;
                     }

            Console.ReadKey();
        }
    }
}
```

# Customers Related Queries

## 1) Display name of customers giving maximum number of orders.

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
/*3.1) Display name of customers giving maximum number of orders.*/
namespace Query1
{
    class Query3_1
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 3.1 \n1) Display name of customers giving maximum number
of orders.");
```

```
            Console.WriteLine("------------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("------------------------------------------------------------
--------------");

            /*
           ---Right Query
            with cust as
                (
                select customer.customer_name,count(customer.customer_id) as 'Total'
                from customer inner join ordermaster
                on customer.customer_id=ordermaster.customer_id group by
customer.customer_name,
                customer.customer_id
                )select Customer_Name from cust where Total=(select max(total) from cust)
            */

            var joinqry = from customer in dbe.customers
                          join ordermaster in dbe.ordermasters on customer.customer_id
equals ordermaster.customer_id
                          group  customer by new { customer.customer_name,
customer.customer_id } into g
                          select new {CustomerName=g.Key.customer_name,Total=g.Count() };

            var customername = from jnyqry in joinqry where jnyqry.Total == (from qry in
joinqry select qry.Total).Max() select jnyqry.CustomerName;
              //Console.WriteLine("Row=" + max_salary.ToList());
              int count = 1;
            foreach (var emp_name1 in customername)
            {
                Console.WriteLine("Row=" + count + "\tEmp Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }


            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("------------------------------------------------------------
--------------");

            joinqry = dbe.ordermasters.Join(dbe.customers, ordermaster =>
ordermaster.customer_id, customer => customer.customer_id, (ordermaster, customer) => new
{ ordermaster, customer }).GroupBy(x => new { x.customer.customer_name,
x.customer.customer_id })
                .Select(g => new
                {
                    CustomerName = g.Key.customer_name,
                    Total = g.Count()
                });
            /*from customer in dbe.customers
                      join ordermaster in dbe.ordermasters on customer.customer_id equals
ordermaster.customer_id
                      group customer by new { customer.customer_name,
customer.customer_id } into g
                      select new { CustomerName = g.Key.customer_name, Total = g.Count()
};*/
```

```
            customername = joinqry.Where(x => x.Total == (joinqry.Select(p =>
p.Total)).Max())
                 .Select(x => x.CustomerName);
                //from jnyqry in joinqry where jnyqry.Total == (from qry in joinqry
select qry.Total).Max() select jnyqry.CustomerName;
            //Console.WriteLine("Row=" + max_salary.ToList());
             count = 1;
            foreach (var emp_name1 in customername)
            {
                Console.WriteLine("Row=" + count + "\tEmp Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }


            Console.ReadKey();
        }
    }
}
```

## 2) Display name of customers who purchased maximum number of products

Solution:=

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query3_2
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();




            Console.WriteLine("Q 3.2 \n1) Display name of customers who purchased maximum
number of products ");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");

            /*
        --Right Query for Submit to Sir
```

```
                Select * from customer where customer_id in (Select customer_id From
(Select customer_id,count(*)as totalProduct from ordermaster inner join orderdetail on
ordermaster.order_id=orderdetail.order_id group by customer_id
                ) t1 where totalProduct in (Select Max(totalProduct) as MaximumProduct
from (Select customer_id,count(*)as totalProduct from ordermaster inner join orderdetail
on ordermaster.order_id=orderdetail.order_id group by customer_id
                )as t));
            */

            var joinqry = from ordermaster in dbe.ordermasters join orderdetail in
dbe.orderdetails on ordermaster.order_id equals orderdetail.order_id
                          group ordermaster by ordermaster.customer_id into g
                          select new { customer_id=g.Key,total=g.Count() };
            var maximum = (from jnqry in joinqry select jnqry.total).Max();
            var actual_customer_id = from jnqry in joinqry where jnqry.total == maximum
select jnqry.customer_id;
            var customername = from customer in dbe.customers where
actual_customer_id.Contains(customer.customer_id) select customer.customer_name;

            int count = 1;
            foreach (var emp_name1 in customername)
            {
                Console.WriteLine("Row=" + count + "\tCustomer Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }


            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("-----------------------------------------------------------
--------------");

            joinqry = dbe.orderdetails.Join(dbe.ordermasters, orderdetail =>
orderdetail.order_id, ordermaster => ordermaster.order_id, (orderdetail, ordermaster) =>
new { orderdetail, ordermaster }
            ).GroupBy(x => x.ordermaster.customer_id).Select(g => new { customer_id =
g.Key, total = g.Count() });
            /*from ordermaster in dbe.ordermasters
                      join orderdetail in dbe.orderdetails on ordermaster.order_id equals
orderdetail.order_id
                      group ordermaster by ordermaster.customer_id into g
                      select new { customer_id = g.Key, total = g.Count() };*/
            maximum = joinqry.Select(x => x.total).Max();
            //(from jnqry in joinqry select jnqry.total).Max();
            actual_customer_id = joinqry.Where(x => x.total == maximum).Select(x =>
x.customer_id);

            //from jnqry in joinqry where jnqry.total == maximum select
jnqry.customer_id;
            customername = dbe.customers.Where(x =>
actual_customer_id.Contains(x.customer_id)).Select(x => x.customer_name);
                //from customer in dbe.customers where
actual_customer_id.Contains(customer.customer_id) select customer.customer_name;

             count = 1;
            foreach (var emp_name1 in customername)
            {
```

```
                Console.WriteLine("Row=" + count + "\tCustomer Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            Console.ReadKey();
        }
    }
}
```

## 3) Display name of customers who purchased maximum number of different products

Solution:=

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query3_3
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 3.2 \n1) Display name of customers who purchased maximum
number of products ");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");

            /*
        --Right Query for Sir to submit
        Select customer_id,customer_name from customer where customer_id in (Select
customer_id from (Select customer_id,Count(DISTINCT  product_id ) as totalDiffProduct
from ordermaster inner join orderdetail on ordermaster.order_id=orderdetail.order_id
group by customer_id
        ) as t1,(
        Select max(totalDiffProduct) as MaximumProduct from (Select
customer_id,Count(DISTINCT  product_id ) as totalDiffProduct from ordermaster inner join
orderdetail on ordermaster.order_id=orderdetail.order_id group by customer_id
        )as t) as t2 where t1.totalDiffProduct=t2.MaximumProduct);
            */
            //Select customer_id, Count(DISTINCT  product_id ) as totalDiffProduct from
ordermaster inner join orderdetail on ordermaster.order_id = orderdetail.order_id group
by customer_id
```

```
            var joinqry = from ordermaster in dbe.ordermasters
                          join orderdetail in dbe.orderdetails on ordermaster.order_id
equals orderdetail.order_id

                          select new { ordermaster.order_id, ordermaster.customer_id,
orderdetail.product_id };
            var newjoinqry = (from jnqry in joinqry group jnqry by jnqry.customer_id into
g select new { customer_id = g.Key, product_id =
g.Select(x=>x.product_id).Distinct().Count() }).ToList();
            var maximum = (from jnqry in newjoinqry select jnqry.product_id).Max();
            var actual_customer_id = from jnqry in newjoinqry where jnqry.product_id ==
maximum select jnqry.customer_id;
            var customername = from customer in dbe.customers where
actual_customer_id.Contains(customer.customer_id) select customer.customer_name;

            int count = 1;
            foreach (var emp_name1 in customername)
            {
                Console.WriteLine("Row=" + count + "\tCustomer Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }


            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("----------------------------------------------------------
--------------");
             joinqry
=dbe.orderdetails.Join(dbe.ordermasters,orderdetail=>orderdetail.order_id,ordermaster=>or
dermaster.order_id,(orderdetail,ordermaster)=> new{orderdetail,ordermaster }).Select(x
=>new { x.ordermaster.order_id, x.ordermaster.customer_id, x.orderdetail.product_id });
            /*from ordermaster in dbe.ordermasters
                      join orderdetail in dbe.orderdetails on ordermaster.order_id equals
orderdetail.order_id

                      select new { ordermaster.order_id, ordermaster.customer_id,
orderdetail.product_id };*/
            newjoinqry = joinqry.GroupBy(x => x.customer_id).Select(x => new {
customer_id = x.Key, product_id = x.Select(x1 => x1.product_id).Distinct().Count()
}).ToList();
                // (from jnqry in joinqry group jnqry by jnqry.customer_id into g select
new { customer_id = g.Key, product_id = g.Select(x => x.product_id).Distinct().Count()
}).ToList();


             maximum = newjoinqry.Select(x => x.product_id).Max();

           actual_customer_id = newjoinqry.Where(x => x.product_id == maximum).Select(x
=> x.customer_id);

           //from jnqry in joinqry where jnqry.total == maximum select jnqry.customer_id;
           customername = dbe.customers.Where(x =>
actual_customer_id.Contains(x.customer_id)).Select(x => x.customer_name);

            count = 1;
            foreach (var emp_name1 in customername)
            {
```

```
                Console.WriteLine("Row=" + count + "\tCustomer Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }

            Console.ReadKey();
        }
    }
}
```

## 4) Display name of customers who are not purchased any product from last three months.

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Data.Entity.Core.Objects;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
/*   4) Display name of customers who are not purchased any product from last three
months.*/
namespace Query1
{
    class Query3_4
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 3.4 \n1) Display name of customers who are not purchased
any product from last three months. ");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-------------------------------------------------------------
--------------");

            /*
        --Right Query for Sir to submit

select Customer_Name from customer where customer_id not in
(select customer_id from ordermaster where DATEDIFF(Month,Order_date,GETDATE())<=3)
             */
           DateTime dt = DateTime.Today;
            var joinqry = from customer in dbe.customers
                          where !(from ordermaster in dbe.ordermasters
                                  where EntityFunctions.DiffMonths(ordermaster.Order_date,
dt) <= 3
                                  select
ordermaster.customer_id).Contains(customer.customer_id)
```

```
                        select customer.customer_name;
            int count = 1;
            foreach (var emp_name1 in joinqry)
            {
                Console.WriteLine("Row=" + count + "\tCustomer Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }


            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            joinqry =
                dbe.customers.Where(customer => !(dbe.ordermasters.Where(ordermaster =>
EntityFunctions.DiffMonths(ordermaster.Order_date, dt) <= 3).Select(ordermaster =>
ordermaster.customer_id)
              ).Contains(customer.customer_id)).Select(customer =>
customer.customer_name);

//dbe.ordermasters.Where(ordermaster=>EntityFunctions.DiffMonths(ordermaster.Order_date,
dt)<=3).Select(ordermaster=>ordermaster.customer_id)
                /*from customer in dbe.customers
                      where !(from ordermaster in dbe.ordermasters
                              where EntityFunctions.DiffMonths(ordermaster.Order_date,
dt) <= 3
                              select
ordermaster.customer_id).Contains(customer.customer_id)
                      select customer.customer_name;*/
            count = 1;
            foreach (var emp_name1 in joinqry)
            {
                Console.WriteLine("Row=" + count + "\tCustomer Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }

            Console.ReadKey();
        }
    }
}
```

## 5) Display name of customers who purchased every month.

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Data.Entity.Core.Objects;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
/*5) Display name of customers who purchased every month.
*/
namespace Query1
```

```
{
    class Query3_5

    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 3.5 \n1) Display name of customers who purchased every
month. ");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-------------------------------------------------------------
--------------");

            /*
        --Right Query for Sir to submit

               with Cust as
            (
            select * from ordermaster where Datediff(year,Order_date,GETDATE())=1
            )select Cust.customer_id,customer.customer_name
            from Cust inner join customer on Cust.customer_id=customer.customer_id
            group by Cust.customer_id,customer.customer_name
            having COUNT(Distinct(DATEPART(mm,Cust.Order_Date)))=12
             */
            DateTime dt = DateTime.Today;
            var joinqry = from ordermaster in dbe.ordermasters
                          where EntityFunctions.DiffYears(ordermaster.Order_date, dt) ==
1
                          select new { ordermaster.order_id,
ordermaster.customer_id,ordermaster.Order_date };
            var mainquery = from cust in joinqry
                            join customer in dbe.customers on cust.customer_id equals
customer.customer_id
                            group new { cust, customer } by new { cust.customer_id,
customer.customer_name } into g
                            where g.Select(x => x.cust.Order_date).Distinct().Count()==12
                            select new
                            {
                                g.Key.customer_id,
                                g.Key.customer_name

                            };
            //var resultqry= from mainqry in mainquery

            int count = 1;
            foreach (var emp_name1 in mainquery)
            {
                Console.WriteLine("Row=" + count + "\tCustomer Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
```

```
            if (count == 1)
                Console.WriteLine("\tNO Record Found ");


            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("------------------------------------------------------------
--------------");
            joinqry = dbe.ordermasters.Where(ordermaster =>
EntityFunctions.DiffYears(ordermaster.Order_date, dt) == 1)
                        .Select(ordermaster => new { ordermaster.order_id,
ordermaster.customer_id, ordermaster.Order_date });
            /*from ordermaster in dbe.ordermasters
                      where EntityFunctions.DiffYears(ordermaster.Order_date, dt) == 1
                      select new { ordermaster.order_id, ordermaster.customer_id,
ordermaster.Order_date };
                      */
            mainquery = dbe.customers.Join(joinqry, customer => customer.customer_id,
cust => cust.customer_id, (customer, cust) => new
            {
                customer,
                cust
            }).GroupBy(x => new
            {
                x.cust.customer_id,
                x.customer.customer_name
            }).Where(g => g.Select(x => x.cust.Order_date).Distinct().Count() ==
12).Select(x => new { x.Key.customer_id, x.Key.customer_name });

                /*from cust in joinqry
                            join customer in dbe.customers on cust.customer_id equals
customer.customer_id
                            group new { cust, customer } by new { cust.customer_id,
customer.customer_name } into g
                            where g.Select(x => x.cust.Order_date).Distinct().Count() ==
12
                            select new
                            {
                                g.Key.customer_id,
                                g.Key.customer_name

                            };*/
            count = 1;
            foreach (var emp_name1 in mainquery)
            {
                Console.WriteLine("Row=" + count + "\tCustomer Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            if (count == 1)
                Console.WriteLine("\t NO Record Found");

            Console.ReadKey();
        }
    }
}
```

## ***Miscellaneous Related Queries***

## 1) Display the name of the product which is costliest.

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query4_1
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 4.1 \n Display the name of the product which is
costliest. ");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-------------------------------------------------------------
--------------");

            /*
        --Right Query for Sir to submit


        Select product_name from product where product_rate in(Select max(product_rate)
from product);
             */
            var query = from product in dbe.products
                        where product.product_rate == (from product1 in dbe.products
select product1.product_rate).Max()
                        select product.product_name;
            int count = 1;
            foreach (var emp_name1 in query)
            {
                Console.WriteLine("Row=" + count + "\tProduct Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            if (count == 1)
                Console.WriteLine("\tNO Record Found ");


            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            query = dbe.products.Where(product => product.product_rate ==
(dbe.products.Select(x => x.product_rate)).Max()).Select(x => x.product_name);
```

```
                /* from product in dbe.products
                   where product.product_rate == (from product1 in dbe.products select
product1.product_rate).Max()
                   select product.product_name;*/
            count = 1;
            foreach (var emp_name1 in query)
            {
                Console.WriteLine("Row=" + count + "\tProduct Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            if (count == 1)
                Console.WriteLine("\t NO Record Found");

            Console.ReadKey();
        }
    }
}
```

## 2) Display the name of customers who are never attended by employee “Peter”.

Solution:=

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query4_2
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 4.2 \n ) Display the name of customers who are never
attended by employee “Peter”.");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-------------------------------------------------------------
--------------");

            /*
        --Right Query for Sir to submit

    Select customer_name from customer where customer_id not in(Select customer_id from
ordermaster where emp_id in (Select emp_id from employee where emp_name like 'Peter'));
```

```
             */
            var emp_id = from employee in dbe.employees where
employee.emp_name.Contains("peter") select employee.emp_id;

            var customer_ids = from ordermaster in dbe.ordermasters
                               where emp_id.Contains(ordermaster.emp_id)
                               select ordermaster.customer_id;
                         /* new
                          {
                          ordermaster.Order_date,
                          ordermaster.order_id,
                          ordermaster.customer_id,
                          ordermaster.emp_id
                          };*/
            var customer_name = from customer in dbe.customers where
!customer_ids.Contains(customer.customer_id) select customer.customer_name;
            int count = 1;
            foreach (var emp_name1 in customer_name)
            {
                Console.WriteLine("Row=" + count + "\tCustomer Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            if (count == 1)
                Console.WriteLine("\tNO Record Found ");


            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            emp_id = dbe.employees.Where(employee =>
employee.emp_name.Contains("peter")).Select(employee => employee.emp_id);
            //from employee in dbe.employees where employee.emp_name.Contains("peter")
select employee.emp_id;
            customer_ids = dbe.ordermasters.Where(ordermaster =>
emp_id.Contains(ordermaster.emp_id)).Select(ordermaster => ordermaster.customer_id);

            /*from ordermaster in dbe.ordermasters
                           where emp_id.Contains(ordermaster.emp_id)
                           select ordermaster.customer_id;*/
            /* new
             {
             ordermaster.Order_date,
             ordermaster.order_id,
             ordermaster.customer_id,
             ordermaster.emp_id
             };*/
            customer_name = dbe.customers.Where(customer =>
!customer_ids.Contains(customer.customer_id)).Select(customer => customer.customer_name);
                //from customer in dbe.customers where
!customer_ids.Contains(customer.customer_id) select customer.customer_name;
             count = 1;
            foreach (var emp_name1 in customer_name)
            {
                Console.WriteLine("Row=" + count + "\tCustomer Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
```

```
                count++;
            }
            if (count == 1)
                Console.WriteLine("\tNO Record Found ");

            Console.ReadKey();
        }
    }
}
```

## 3) Display the total billing done by employee “Peter”.

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query4_3
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 4.3 \n ) ) Display the total billing done by employee
“Peter”.");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");

            /*
        --Right Query for Sir to submit

   select Total_Billing=sum(Orderdetail.quantity*Product.product_rate)fromOrderdetail
innerjoin Product
   on Orderdetail.product_id=Product.product_idwhere Orderdetail.order_id
   in(select order_id from Ordermaster where emp_id=(select emp_id from employee where
emp_name='Peter'))
             */
            var emp_id = from employee in dbe.employees where
employee.emp_name.Contains("peter") select employee.emp_id;

            var order_ids = from ordermaster in dbe.ordermasters
                            where emp_id.Contains(ordermaster.emp_id)
                            select ordermaster.order_id;
            var order_details = from orderdetail in dbe.orderdetails
                             where order_ids.Contains(orderdetail.order_id)
                             select  orderdetail.order_id;
```

```
            var joinqry = from orderdetail in dbe.orderdetails join product in
dbe.products on orderdetail.product_id equals product.product_id
                          where order_details.Contains(orderdetail.order_id)
                          select new {
orderdetail.order_id,totalRate=product.product_rate*orderdetail.quantity };
            var groupquery = from jnqry in joinqry
                             group jnqry by jnqry.order_id into g
                             select new
                             {
                                 order_id = g.Key,
                                 sum = g.Select(x => x.totalRate).Sum()
                             };
            var TotalBilling = (from grp in groupquery select grp.sum).Sum();
            Console.WriteLine("Total Billing done by Peter=" + TotalBilling);



            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("------------------------------------------------------------
--------------");
            emp_id = dbe.employees.Where(employee =>
employee.emp_name.Contains("peter")).Select(employee => employee.emp_id);

            //from employee in dbe.employees where employee.emp_name.Contains("peter")
select employee.emp_id;
           var inner_order_id = dbe.ordermasters.Where(ordermaster =>
emp_id.Contains(ordermaster.emp_id)).Select(ordermaster => ordermaster.order_id);
            order_details = dbe.orderdetails.Where(orderdetail =>
inner_order_id.Contains(orderdetail.order_id)).Select(orderdetail =>
orderdetail.order_id);
            /* order_details = from orderdetail in dbe.orderdetails
                                where order_ids.Contains(orderdetail.order_id)
                                select orderdetail.order_id;
                */
             joinqry = dbe.orderdetails.Join(dbe.products, orderdetail =>
orderdetail.product_id,product=>product.product_id,(orderdetail,product)=> new {
orderdetail, product }).Where(x=> order_details.Contains(x.orderdetail.order_id))
                .Select(

                x=>new
                {
                   order_id= x.orderdetail.order_id,
                    totalRate=x.product.product_rate*x.orderdetail.quantity
                }
                );

            /*from orderdetail in dbe.orderdetails
                        join product in dbe.products on orderdetail.product_id equals
product.product_id
                        where order_details.Contains(orderdetail.order_id)
                        select new { orderdetail.order_id, totalRate = product.product_rate
* orderdetail.quantity };
       */
            groupquery = joinqry.GroupBy(x => x.order_id).Select(g => new
            {
                order_id = g.Key,
                sum = g.Select(x => x.totalRate).Sum()
```

```
            });
            /*
            var groupquery = from jnqry in joinqry
                             group jnqry by jnqry.order_id into g
                             select new
                             {
                                 order_id = g.Key,
                                 sum = g.Select(x => x.totalRate).Sum()
                             };*/
            TotalBilling = groupquery.Select(x => x.sum).Sum();
                //(from grp in groupquery select grp.sum).Sum();
            Console.WriteLine("Total Billing done by Peter=" + TotalBilling);
            /*int count = 1;
            foreach (var emp_name1 in groupquery)
            {
                Console.WriteLine("Row=" + count + "\tCustomer Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            if (count == 1)
                Console.WriteLine("\tNO Record Found ");
                */
            Console.ReadKey();
        }
    }
}
```

## 4) Display the name of customer who has purchased “Pepsi” but not “Lime Water”.

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
/*4_4) Display the name of customer who has purchased “Pepsi” but not “Lim e  Wat er”.*/
namespace Query1
{
    class Query4_4
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 4.4 \n ) )4) Display the name of customer who has
purchased “Pepsi” but not “Lime  Wat er”.");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
```

```
            Console.WriteLine("------------------------------------------------------------
--------------");

            /*
     --Right Query to Submit to The Sir
        Select customer_id,customer_name from customer where customer_id in (Select
customer_id from (Select * from (Select
ordermaster.order_id,Order_date,customer_id,emp_id,quantity,orderdetail.product_id,produc
t_name,product_rate from ordermaster,orderdetail,product where
ordermaster.order_id=orderdetail.order_id and orderdetail.product_id=product.product_id
        ) as t1 where product_name like 'Pepsi'
        )as t where customer_id not in (Select customer_id from (Select
ordermaster.order_id,Order_date,customer_id,emp_id,quantity,orderdetail.product_id,produc
t_name,product_rate from ordermaster,orderdetail,product where
ordermaster.order_id=orderdetail.order_id and orderdetail.product_id=product.product_id
        ) as t2 where product_name like 'Lime Water'))
          */
            var joinquery = from ordermaster in dbe.ordermasters
                            join orderdetail in dbe.orderdetails on ordermaster.order_id
equals orderdetail.order_id join
                            product in dbe.products on orderdetail.product_id equals
product.product_id
                            select new
                            {
                                ordermaster.order_id,
                                ordermaster.customer_id,
                                orderdetail.product_id,
                                product.product_name,
                                product.product_rate

                            };
            var onlyPepsis = from joinqry in joinquery where
joinqry.product_name.Contains("pepsi") select joinqry;
            var onlyLimeWater = from joinqry in joinquery where
joinqry.product_name.Contains("Lime Water") select joinqry.customer_id;
            var customer_id = from onlyPepsi in onlyPepsis where
!onlyLimeWater.Contains(onlyPepsi.customer_id) select onlyPepsi.customer_id;
            var customer_name = from cust in dbe.customers where
customer_id.Contains(cust.customer_id) select cust.customer_name;
            int count = 1;
            foreach (var emp_name1 in customer_name)
            {
                Console.WriteLine("Row=" + count + "\tCustomer Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            if (count == 1)
                Console.WriteLine("\tNO Record Found ");
            Console.WriteLine("Solution using Lamda Expression and Methods");
            Console.WriteLine("------------------------------------------------------------
--------------");
            joinquery = dbe.ordermasters.Join(dbe.orderdetails, ordermaster =>
ordermaster.order_id, orderdetail => orderdetail.order_id, (ordermaster, orderdetail) =>
new { ordermaster, orderdetail })
                .Join(dbe.products, oldjoin => oldjoin.orderdetail.product_id, product =>
product.product_id, (oldjoin, product) => new { oldjoin, product })
                .Select(x =>
```

```
                new
                {x.oldjoin.ordermaster.order_id,
                x.oldjoin.ordermaster.customer_id,
                x.oldjoin.orderdetail.product_id,
                x.product.product_name,
                x.product.product_rate

                });



            /* from ordermaster in dbe.ordermasters
                        join orderdetail in dbe.orderdetails on ordermaster.order_id
equals orderdetail.order_id
                        join
            product in dbe.products on orderdetail.product_id equals product.product_id
                        select new
                        {
                            ordermaster.order_id,
                            ordermaster.customer_id,
                            orderdetail.product_id,
                            product.product_name,
                            product.product_rate

                        };*/
            onlyPepsis = joinquery.Where(joinqry =>
joinqry.product_name.Contains("pepsi")).Select(joinqry => joinqry);
                //from joinqry in joinquery where joinqry.product_name.Contains("pepsi")
select joinqry;
             onlyLimeWater = joinquery.Where(joinqry =>
joinqry.product_name.Contains("Lime Water")).Select(joinqry => joinqry.customer_id);
            //from joinqry in joinquery where joinqry.product_name.Contains("Lime Water")
select joinqry.customer_id;
            customer_id = onlyPepsis.Where(x =>
!onlyLimeWater.Contains(x.customer_id)).Select(x => x.customer_id);
            //from onlyPepsi in onlyPepsis where
!onlyLimeWater.Contains(onlyPepsi.customer_id) select onlyPepsi.customer_id;
            customer_name = dbe.customers.Where(x =>
customer_id.Contains(x.customer_id)).Select(x => x.customer_name);
                //from cust in dbe.customers where customer_id.Contains(cust.customer_id)
select cust.customer_name;
             count = 1;
            foreach (var emp_name1 in customer_name)
            {
                Console.WriteLine("Row=" + count + "\tCustomer Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            if (count == 1)
                Console.WriteLine("\tNO Record Found ");



            Console.ReadKey();
        }
    }
}
```

## 5) Display the name of employee who generated maximum revenue for month of January.

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Data.Entity.Core.Objects;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query4_5
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 4.5) \n  Display the name of employee who generated
maximum revenue for month of January");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");

            /*
     --Right Query to Submit to The Sir

Select * from (
Select
employee.emp_name,ordermaster.order_id,ordermaster.emp_id,orderdetail.product_id,orderdet
ail.quantity,product.product_rate,(orderdetail.quantity * product.product_rate) as Total
from ordermaster inner join orderdetail on ordermaster.order_id=orderdetail.order_id
inner join product on orderdetail.product_id=product.product_id
inner join employee on employee.emp_id=ordermaster.emp_id
where Month(ordermaster.Order_date)=1)as t1 where   t1.Total=(
Select max(t1.Total) from (
Select
employee.emp_name,ordermaster.order_id,ordermaster.emp_id,orderdetail.product_id,orderdet
ail.quantity,product.product_rate,(orderdetail.quantity * product.product_rate) as Total
from ordermaster inner join orderdetail on ordermaster.order_id=orderdetail.order_id
inner join product on orderdetail.product_id=product.product_id
inner join employee on employee.emp_id=ordermaster.emp_id
where Month(ordermaster.Order_date)=1)as t1)  */

            var joinquery = from lst in (from ordermaster in dbe.ordermasters
                                         join orderdetail in dbe.orderdetails on
ordermaster.order_id equals orderdetail.order_id
                                         join
```

```
                                        product in dbe.products on
orderdetail.product_id equals product.product_id
                                        join
                                        employee in dbe.employees on ordermaster.emp_id
equals employee.emp_id

                                        select new
                                        {
                                            employee.emp_name,
                                            ordermaster.order_id,
                                            ordermaster.Order_date,
                                            ordermaster.emp_id,
                                            orderdetail.product_id,
                                            orderdetail.quantity,
                                            product.product_rate,
                                            Total = (orderdetail.quantity *
product.product_rate)

                                        }).ToList() where
Convert.ToDateTime(lst.Order_date).Month == 1 select lst;
            var maximum = (from lst in joinquery select lst.Total).Max();
            // Console.WriteLine("total=" + maximum);
            var resultqry = from lst in joinquery where lst.Total == maximum select
lst.emp_name;

            int count = 1;
            foreach (var emp_name1 in resultqry)
            {
                Console.WriteLine("Row=" + count + "\tEmployee Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            if (count == 1)
                Console.WriteLine("\tNO Record Found ");
            Console.WriteLine("Solution using Lamda Expression and Methods");

            joinquery = dbe.ordermasters.Join(dbe.orderdetails, ordermaster =>
ordermaster.order_id, orderdetail => orderdetail.order_id, (ordermaster, orderdetail) =>
new { ordermaster, orderdetail })
                .Join(dbe.products, outer1query => outer1query.orderdetail.product_id,
product => product.product_id, (outer1query, product) => new { outer1query, product })
                .Join(dbe.employees, outer2query =>
outer2query.outer1query.ordermaster.emp_id, employee => employee.emp_id, (outer2query,
employee) => new { outer2query, employee })
                .Select(x => new
                {
                    x.employee.emp_name,
                    x.outer2query.outer1query.ordermaster.order_id,
                    x.outer2query.outer1query.ordermaster.Order_date,
                    x.outer2query.outer1query.ordermaster.emp_id,
                    x.outer2query.outer1query.orderdetail.product_id,
                    x.outer2query.outer1query.orderdetail.quantity,
                    x.outer2query.product.product_rate,
                    Total = (x.outer2query.outer1query.orderdetail.quantity *
x.outer2query.product.product_rate)
                }).ToList().Where(x => Convert.ToDateTime(x.Order_date).Month ==
1).Select(lst => lst);
```

```
            /*from lst in (from ordermaster in dbe.ordermasters
                                    join orderdetail in dbe.orderdetails on
ordermaster.order_id equals orderdetail.order_id
                                    join
                                    product in dbe.products on orderdetail.product_id
equals product.product_id
                                    join
                                    employee in dbe.employees on ordermaster.emp_id
equals employee.emp_id

                                    select new
                                    {
                                        employee.emp_name,
                                        ordermaster.order_id,
                                        ordermaster.Order_date,
                                        ordermaster.emp_id,
                                        orderdetail.product_id,
                                        orderdetail.quantity,
                                        product.product_rate,
                                        Total = (orderdetail.quantity *
product.product_rate)

                                    }).ToList()
                          where Convert.ToDateTime(lst.Order_date).Month == 1
                          select lst;
                          */
            maximum = joinquery.Select(lst => lst.Total).Max();
            //(from lst in joinquery select lst.Total).Max();
            // Console.WriteLine("total=" + maximum);
            resultqry = joinquery.Where(lst => lst.Total == maximum).Select(lst =>
lst.emp_name);
                //from lst in joinquery where lst.Total == maximum select lst.emp_name;


            count = 1;
            foreach (var emp_name1 in resultqry)
            {
                Console.WriteLine("Row=" + count + "\tEmployee Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            if (count == 1)
                Console.WriteLine("\tNO Record Found ");



            Console.ReadKey();
        }
    }
}
```

## 6) Display the name of customer who is attended by “Peter” & “Bob”.

Solution:-

```
using Queries;
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
/*Display the name of customer who is attended by “Peter” & “Bob”*/
namespace Query1
{
    class Query4_6
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 4.6) \n Display the name of customer who is attended by
“Peter” & “Bob”.");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");

            /*
     --Right Query to Submit to The Sir

select Customer_Name from customer where customer_id in
(select ordermaster.customer_id from ordermaster where customer_id in
(select customer_id from ordermaster where emp_id in
(select emp_id from employee where emp_name='Peter'))
and emp_id=(select emp_id from employee where emp_name='Bob'))
 */

            /* Select customer_id from ordermaster where emp_id in (Select emp_id from
employee where emp_name like'Peter');*/

            var Peter_id = from employee in dbe.employees where
employee.emp_name.Contains("Peter") select employee.emp_id;
            var Bob_id = from employee in dbe.employees where
employee.emp_name.Contains("Bob") select employee.emp_id;

            var customer_id1 = from ordermaster in dbe.ordermasters
                               where Peter_id.Contains(ordermaster.emp_id)
                               select ordermaster.customer_id;
            var customer_id2 = from ordermaster in dbe.ordermasters
                              where Bob_id.Contains(ordermaster.emp_id)
                              select ordermaster.customer_id;
            var customer_id = from cust1 in customer_id1 join cust2 in customer_id2 on
cust1 equals cust2 select cust1;
            var customer_name = from customer in dbe.customers where
customer_id.Contains(customer.customer_id) select customer.customer_name;

            int count = 1;
            foreach (var emp_name1 in customer_name)
            {
                Console.WriteLine("Row=" + count + "\tCustomer Name=" + emp_name1);
```

```
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            if (count == 1)
                Console.WriteLine("\tNO Record Found ");
            Console.WriteLine("Solution using Lamda Expression and Methods");
            Peter_id = dbe.employees.Where(employee =>
employee.emp_name.Contains("Peter")).Select(employee => employee.emp_id);
                //from employee in dbe.employees where
employee.emp_name.Contains("Peter") select employee.emp_id;
             Bob_id = dbe.employees.Where(employee =>
employee.emp_name.Contains("Bob")).Select(employee => employee.emp_id);
            //from employee in dbe.employees where employee.emp_name.Contains("Bob")
select employee.emp_id;

            customer_id1 = dbe.ordermasters.Where(ordermaster =>
Peter_id.Contains(ordermaster.emp_id)).Select(ordermaster => ordermaster.customer_id);
                /*from ordermaster in dbe.ordermasters
                                where Peter_id.Contains(ordermaster.emp_id)
                                select ordermaster.customer_id;*/
             customer_id2 = dbe.ordermasters.Where(ordermaster =>
Bob_id.Contains(ordermaster.emp_id)).Select(ordermaster => ordermaster.customer_id);
            /*from ordermaster in dbe.ordermasters
                           where Bob_id.Contains(ordermaster.emp_id)
                           select ordermaster.customer_id;*/
            customer_id = customer_id1.Join(customer_id2, cust1 => cust1, cust2 => cust2,
(cust1, cust2) => new { cust1, cust2 }).Select(x => x.cust1);
            //from cust1 in customer_id1 join cust2 in customer_id2 on cust1 equals cust2
select cust1;
            customer_name = dbe.customers.Where(customer =>
customer_id.Contains(customer.customer_id)).Select(customer => customer.customer_name);
                //from customer in dbe.customers where
customer_id.Contains(customer.customer_id) select customer.customer_name;



            count = 1;
            foreach (var emp_name1 in customer_name)
            {
                Console.WriteLine("Row=" + count + "\tCustomer Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            if (count == 1)
                Console.WriteLine("\tNO Record Found ");



            Console.ReadKey();
        }
    }
}
```

## 7) Display the name of employee who has generated maximum revenue for today.

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{


    class Query4_7
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 4.7) \n Display the name of employee who has generated
maximum revenue for today.");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-------------------------------------------------------------
--------------");

            /*
     --Right Query to Submit to The Sir
 Select * from employee where emp_id in (
Select top(1) emp_id from (
Select sum(total) as total,emp_id from (Select
ordermaster.order_id,ordermaster.Order_date,ordermaster.emp_id,orderdetail.quantity*produ
ct.product_rate as total from ordermaster inner join orderdetail on
orderdetail.order_id=ordermaster.order_id inner join product  on
orderdetail.product_id=product.product_id where
CONVERT(date,Order_date)=CONVERT(date,GETDATE())
) as t group by emp_id ) as t
order by total desc
)
 */

            /*
Select
ordermaster.order_id,ordermaster.Order_date,ordermaster.emp_id,orderdetail.quantity*produ
ct.product_rate as total from
ordermaster inner join orderdetail on orderdetail.order_id=ordermaster.order_id inner
join product  on orderdetail.product_id=product.product_id
*/
            var joinquery = from lst in(from ordermaster in dbe.ordermasters
```

```
                            join orderdetail in dbe.orderdetails on ordermaster.order_id
equals orderdetail.order_id
                            join product in dbe.products on orderdetail.product_id equals
product.product_id

                            select new
                            {
                                ordermaster.order_id,
                                ordermaster.Order_date,
                                ordermaster.emp_id,
                                Total = orderdetail.quantity * product.product_rate

                            }).ToList()
                            where   Convert.ToDateTime(lst.Order_date) ==DateTime.Today
                            select lst;
            var grpqry = from joinqry in joinquery
                         group joinqry by joinqry.emp_id into g
                         select new
                         {
                             emp_id = g.Key,
                             total = g.Select(x => x.Total).Sum()
                         };
            var maximumTotal = (from tt in grpqry select tt.total).Max();

            Console.WriteLine("Maximumtotal=" + maximumTotal);
            var emp_id = from tt in grpqry where tt.total == maximumTotal select
tt.emp_id;
            var emp_name = from emp in dbe.employees where emp_id.Contains(emp.emp_id)
select emp.emp_name;

            int count = 1;
            foreach (var emp_name1 in emp_name)
            {
                Console.WriteLine("Row=" + count + "\tEmployee Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            if (count == 1)
                Console.WriteLine("\tNO Record Found ");
            Console.WriteLine("-----------Solution using Lamda Expression and Methods----
--------------");

            joinquery = dbe.ordermasters.Join(dbe.orderdetails, ordermaster =>
ordermaster.order_id, orderdetail => orderdetail.order_id, (ordermaster, orderdetail) =>
new { ordermaster, orderdetail })
                .Join(dbe.products, innerquery => innerquery.orderdetail.product_id,
product => product.product_id, (innerquery, product) => new { innerquery, product })
                .Select(x => new
                {
                    x.innerquery.ordermaster.order_id,

                    x.innerquery.ordermaster.Order_date,
                    x.innerquery.ordermaster.emp_id,
                    Total = x.innerquery.orderdetail.quantity * x.product.product_rate

                }).ToList().Where(lst=>Convert.ToDateTime(lst.Order_date) ==
DateTime.Today);
```

```
            joinquery = dbe.ordermasters.Join(dbe.orderdetails, ordermaster =>
ordermaster.order_id, orderdetail => orderdetail.order_id, (ordermaster, orderdetail) =>
new { ordermaster, orderdetail })
              .Join(dbe.products, innerquery => innerquery.orderdetail.product_id,
product => product.product_id, (innerquery, product) => new { innerquery, product })
              .Select(x => new
              {
                  x.innerquery.ordermaster.order_id,

                  x.innerquery.ordermaster.Order_date,
                  x.innerquery.ordermaster.emp_id,
                  Total = x.innerquery.orderdetail.quantity * x.product.product_rate

              }).ToList().Where(lst => Convert.ToDateTime(lst.Order_date) ==
DateTime.Today);
            /*from lst in (from ordermaster in dbe.ordermasters
                                    join orderdetail in dbe.orderdetails on
ordermaster.order_id equals orderdetail.order_id
                                    join product in dbe.products on
orderdetail.product_id equals product.product_id

                                    select new
                                    {
                                        ordermaster.order_id,
                                        ordermaster.Order_date,
                                        ordermaster.emp_id,
                                        Total = orderdetail.quantity *
product.product_rate

                                    }).ToList()
                        where Convert.ToDateTime(lst.Order_date) == DateTime.Today
                        select lst;*/
            grpqry = joinquery.GroupBy(joinqry => joinqry.emp_id).Select(g => new
            {
                emp_id = g.Key,
                total = g.Select(x => x.Total).Sum()
            });
            /*from joinqry in joinquery
                     group joinqry by joinqry.emp_id into g
                     select new
                     {
                         emp_id = g.Key,
                         total = g.Select(x => x.Total).Sum()
                     };*/
            maximumTotal = grpqry.Select(x => x.total).Max();
                //(from tt in grpqry select tt.total).Max();

            Console.WriteLine("Maximumtotal=" + maximumTotal);
             emp_id = grpqry.Where(tt => tt.total == maximumTotal).Select(tt =>
tt.emp_id);
            //from tt in grpqry where tt.total == maximumTotal select tt.emp_id;
            emp_name = dbe.employees.Where(emp => emp_id.Contains(emp.emp_id)).Select(emp
=> emp.emp_name);
                //from emp in dbe.employees where emp_id.Contains(emp.emp_id) select
emp.emp_name;
```

```
            count = 1;
            foreach (var emp_name1 in emp_name)
            {
                Console.WriteLine("Row=" + count + "\tEmployee Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            if (count == 1)
                Console.WriteLine("\tNO Record Found ");



            Console.ReadKey();
        }
    }
}
```

## 8) Display name of employees who has attended customer “Thompson”.

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
/*8) Display name of employees who has attended customer “Thompson”.
*/
namespace Query1
{
    class Query4_8
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();




            Console.WriteLine("Q 4.8) \n Display name of employees who has attended
customer “Thompson”.");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");

            /*
     --Right Query to Submit to The Sir

            select emp_name from employee where emp_id in
            (select emp_id from ordermaster where customer_id=
```

```
            (select customer_id from customer where customer_name='Thompson'))

 */
            var emp_name = from employee in dbe.employees
                                where (
(from ordermaster in dbe.ordermasters
where
(from customer in dbe.customers where customer.customer_name == "Thompson" select
customer.customer_id)
.Contains(ordermaster.customer_id)

select ordermaster.emp_id)//end of secound qry
).Contains(employee.emp_id)
                                select employee.emp_name;

            int count = 1;
            foreach (var emp_name1 in emp_name)
            {
                Console.WriteLine("Row=" + count + "\tEmployee Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            if (count == 1)
                Console.WriteLine("\tNO Record Found ");
            Console.WriteLine("-----------Solution using Lamda Expression and Methods----
--------------");

            emp_name = dbe.employees
                .Where(emp =>
                (dbe.ordermasters.
               Where(ordermaster =>
               dbe.customers.Where(customer => customer.customer_name ==
"Thompson").Select(customer => customer.customer_id)
                .Contains(ordermaster.customer_id))
                .Select(ordermaster => ordermaster.emp_id))//end of secound qry;
                .Contains(emp.emp_id)).Select(emp => emp.emp_name);

                /*from employee in dbe.employees
                           where (
(from ordermaster in dbe.ordermasters
where
 (from customer in dbe.customers where customer.customer_name == "Thompson" select
customer.customer_id)
 .Contains(ordermaster.customer_id)

select ordermaster.emp_id)//end of secound qry
).Contains(employee.emp_id)
                           select employee.emp_name;


                */

            count = 1;
            foreach (var emp_name1 in emp_name)
            {
                Console.WriteLine("Row=" + count + "\tEmployee Name=" + emp_name1);
```

```
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            if (count == 1)
                Console.WriteLine("\tNO Record Found ");



            Console.ReadKey();
        }
    }
}
```

## 9) Display the order ids of the order which are placed by “Thompson” but not attended by “Kevin”.

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
/*9) Display the order ids of the order which are placed by “Thompson” but not attended
 by “Kevin”.*/
namespace Query1
{
    class Query4_9
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();




            Console.WriteLine("Q 4.9) \n Display the order ids of the order which are
placed by “Thompson” but not attended by “Kevin”.");
            Console.WriteLine("------------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("------------------------------------------------------------
--------------");

            /*
     --Right Query to Submit to The Sir
         select order_id from ordermaster where customer_id=
        (select customer_id from customer where customer_name='Thompson') and
        emp_id!=(select emp_id from employee where emp_name='Kevin')

 */
            var order_id = from ordermaster in dbe.ordermasters
                           where
(from customer in dbe.customers where customer.customer_name.Contains("Thompson") select
customer.customer_id)
```

```
.Contains(ordermaster.customer_id) && !(from employee in dbe.employees where
employee.emp_name.Contains("Kevin") select employee.emp_id)
.Contains(ordermaster.emp_id)
                          select ordermaster.order_id;



            int count = 1;
            foreach (var emp_name1 in order_id)
            {
                Console.WriteLine("Row=" + count + "\tOrder_id =" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            if (count == 1)
                Console.WriteLine("\tNO Record Found ");
            Console.WriteLine("-----------Solution using Lamda Expression and Methods----
--------------");
            order_id = dbe.ordermasters.Where(ordermaster =>

             (dbe.customers.Where(customer =>
customer.customer_name.Contains("Thompson")).Select(customer => customer.customer_id)).
             Contains(ordermaster.customer_id)
             && !(dbe.employees.Where(employee =>
employee.emp_name.Contains("Kevin")).Select(employee => employee.emp_id)).
             Contains(ordermaster.emp_id)
            ).Select(ordermaster => ordermaster.order_id);

                /*from ordermaster in dbe.ordermasters
                      where
(from customer in dbe.customers where customer.customer_name.Contains("Thompson") select
customer.customer_id)
.Contains(ordermaster.customer_id) && !(from employee in dbe.employees where
employee.emp_name.Contains("Kevin") select employee.emp_id)
.Contains(ordermaster.emp_id)
                      select ordermaster.order_id;*/


            count = 1;
            foreach (var emp_name1 in order_id)
            {
                Console.WriteLine("Row=" + count + "\tOrder id =" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            if (count == 1)
                Console.WriteLine("\tNO Record Found ");



            Console.ReadKey();
        }
    }
}
```

## 10) Display the name of manager whose team has generated maximum revenue for current financial year.

Solution;-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{/*10) Display the name of manager whose team has generated maximum revenue for current
financial year.
*/
    class Query4_10
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 4.10) \n Display the name of manager whose team has
generated maximum revenue for current financial year.");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-------------------------------------------------------------
--------------");

            /*
     --Right Query to Submit to The Sir
         select order_id from ordermaster where customer_id=
        (select customer_id from customer where customer_name='Thompson') and
        emp_id!=(select emp_id from employee where emp_name='Kevin')

 */



            var joinquery = from lst in (from ordermaster in dbe.ordermasters
                                         join orderdetail in dbe.orderdetails on
ordermaster.order_id equals orderdetail.order_id
                                         join product in dbe.products on
orderdetail.product_id equals product.product_id
                                         join employee in dbe.employees on
ordermaster.emp_id equals employee.emp_id
                                         select new
                                         {
                                             ordermaster.order_id,
                                             ordermaster.Order_date,
                                             ordermaster.emp_id,
                                             employee.emp_manager_id,
                                             employee.emp_name,
```

```
                                        Total = orderdetail.quantity *
product.product_rate

                                    }).ToList()
                           where Convert.ToDateTime(lst.Order_date).Year ==
DateTime.Today.Year
                           select lst;
            var grpqry = from joinqry in joinquery
                         group joinqry by new { joinqry.emp_manager_id, joinqry.emp_name
} into g
                         select new
                         {
                             emp_id = g.Key.emp_manager_id,
                             total = g.Select(x => x.Total).Sum()
                         };
            var maximumTotal = (from tt in grpqry select tt.total).Max();
            Console.WriteLine("Maximumtotal=" + maximumTotal);
            var emp_id = from tt in grpqry where tt.total == maximumTotal select
tt.emp_id;
            var emp_name = from emp in dbe.employees where emp_id.Contains(emp.emp_id)
select emp.emp_name;

            var order_id = from ordermaster in dbe.ordermasters
                           where
(from customer in dbe.customers where customer.customer_name.Contains("Thompson") select
customer.customer_id)
.Contains(ordermaster.customer_id) && !(from employee in dbe.employees where
employee.emp_name.Contains("Kevin") select employee.emp_id)
.Contains(ordermaster.emp_id)
                           select ordermaster.order_id;



            int count = 1;
            foreach (var emp_name1 in emp_name)
            {
                Console.WriteLine("Row=" + count + "\tManager Name =" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            if (count == 1)
                Console.WriteLine("\tNO Record Found ");
            Console.WriteLine("-----------Solution using Lamda Expression and Methods----
--------------");


            var joinquery1 = dbe.ordermasters.Join(dbe.orderdetails, ordermaster =>
ordermaster.order_id, orderdetail => orderdetail.order_id, (ordermaster, orderdetail) =>
new { ordermaster, orderdetail })
            .Join(dbe.products, innerquery => innerquery.orderdetail.product_id, product
=> product.product_id, (innerquery, product) => new { innerquery, product })
            .Join(dbe.employees, outerquery => outerquery.innerquery.ordermaster.emp_id,
employee => employee.emp_id, (outerquery, employee) => new { outerquery, employee })
           .Select(x => new
            {
                x.outerquery.innerquery.ordermaster.order_id,
```

```
                x.outerquery.innerquery.ordermaster.Order_date,
                x.outerquery.innerquery.ordermaster.emp_id,
               x.outerquery.innerquery.ordermaster.employee.emp_name,
               x.outerquery.innerquery.ordermaster.employee.emp_manager_id,
               Total = x.outerquery.innerquery.orderdetail.quantity *
x.outerquery.product.product_rate

            }).ToList().Where(lst => Convert.ToDateTime(lst.Order_date).Year ==
DateTime.Today.Year);


            /*from lst in (from ordermaster in dbe.ordermasters
                                    join orderdetail in dbe.orderdetails on
ordermaster.order_id equals orderdetail.order_id
                                    join product in dbe.products on
orderdetail.product_id equals product.product_id
                                    join employee in dbe.employees on ordermaster.emp_id
equals employee.emp_id
                                    select new
                                    {
                                        ordermaster.order_id,
                                        ordermaster.Order_date,
                                        ordermaster.emp_id,
                                        employee.emp_manager_id,
                                        employee.emp_name,
                                        Total = orderdetail.quantity *
product.product_rate

                                    }).ToList()
                         where Convert.ToDateTime(lst.Order_date).Year ==
DateTime.Today.Year
                         select lst;
            */
            grpqry = joinquery1.GroupBy(joinqry => new { joinqry.emp_manager_id,
joinqry.emp_name }).Select(g => new
            {
                emp_id = g.Key.emp_manager_id,
                total = g.Select(x => x.Total).Sum()
            });
            /*from joinqry in joinquery
                         group joinqry by new { joinqry.emp_manager_id, joinqry.emp_name
} into g
                         select new
                         {
                             emp_id = g.Key.emp_manager_id,
                             total = g.Select(x => x.Total).Sum()
                         };*/
            maximumTotal = grpqry.Select(x => x.total).Max();
            //(from tt in grpqry select tt.total).Max();

            Console.WriteLine("Maximumtotal=" + maximumTotal);
            emp_id = grpqry.Where(tt => tt.total == maximumTotal).Select(tt =>
tt.emp_id);
            //from tt in grpqry where tt.total == maximumTotal select tt.emp_id;
            emp_name = dbe.employees.Where(emp => emp_id.Contains(emp.emp_id)).Select(emp
=> emp.emp_name);
            //from emp in dbe.employees where emp_id.Contains(emp.emp_id) select
emp.emp_name;
```

```
            count = 1;
            foreach (var emp_name1 in emp_name)
            {
                Console.WriteLine("Row=" + count + "\tEmployee Name=" + emp_name1);
                //Console.WriteLine("Row=" + count + "\tEmp Name=" +
emp_name1.name+"\tQuantity="+emp_name1.quantity);
                count++;
            }
            if (count == 1)
                Console.WriteLine("\tNO Record Found ");




            Console.ReadKey();
        }
    }
}
```

## General Perpose Releated Queries

Display dept id along with name of all employees in that department .

The output will be as such:

| Dept ID | Employee |
|---|---|
| 10 | Michael, Arnold |
| 20 | Bob,Maria,Peter |

.....

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
/*
1) Display dept id along with name of all employees in that department
The output will be as such:
Dept ID Employee
10 Michael, Arnold
20 Bob,Maria,Peter
```

```
*/
namespace Query1
{
    class Query5_1
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 5.1) \n Display dept id along with name of all employees
in that department\nThe output will be as such:\n Dept ID Employee"+
            "\n  10 \tMichael, Arnold"+
            "\n  20 \tBob, Maria, Peter");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            /*
            1) Display dept id along with name of all employees in that department
                The output will be as such:
                Dept ID Employee
                10 Michael, Arnold
                20 Bob,Maria,Peter
                */
            /*

                Begin
                 DECLARE @listStr VARCHAR(MAX)

                 DECLARE @dept int
                 DECLARE @count int
                 set @count=1;
                 Create table #Temp
                 (
                DeptId int,Employee varchar(max)
                 )
                 while(@count<= (select count(*) from department))
                 Begin
                SELECT @listStr = COALESCE(@listStr+',' ,'') +
                emp_name,@dept=dept_id
                FROM Employee where dept_id=
                (select n.dept_id from (select Dept_id,RowNumber=ROW_NUMBER()
                over(order by Dept_Id) from department)n
                where n.RowNumber=@count)
                insert into #Temp SELECT @dept as DeptId,@listStr as Employee
                set @count=@count+1;
                set @listStr=null
                 End
                 select * from #Temp
                 drop table #Temp
                End
        */
```

```
            var temptbl = from employee in dbe.employees orderby employee.dept_id select
new { employee.dept_id, employee.emp_name };
            var tempdeptid = from department in dbe.departments select
department.dept_id;
            int count = 1;
            Console.WriteLine("Row no\t Department Name\t\t Employees Name");
            foreach (var dep in tempdeptid)
            {
                int d = Convert.ToInt32(dep);
                var emp_name1 = string.Join(",", temptbl.Where(x=> d ==
x.dept_id).Select(x => x.emp_name));
                Console.WriteLine(count+"\t\t"+dep+"\t\t\t"+emp_name1);
                count++;

            }

            if (count == 1)
                Console.WriteLine("\tNO Record Found ");
            Console.WriteLine("-----------Solution using Lamda Expression and Methods----
--------------");



            temptbl = dbe.employees.OrderBy(x => x.dept_id).Select(x => new { x.dept_id,
x.emp_name });
            //from employee in dbe.employees orderby employee.dept_id select new {
employee.dept_id, employee.emp_name };
            tempdeptid = dbe.departments.Select(x => x.dept_id);
                //from department in dbe.departments select department.dept_id;
            //Console.WriteLine(emp_name);
             count = 1;
            Console.WriteLine("Row no\t Department Name\t\t Employees Name");
            foreach (var dep in tempdeptid)
            {
                int d = Convert.ToInt32(dep);
                var emp_name1 = string.Join(",", temptbl.Where(x => d ==
x.dept_id).Select(x => x.emp_name));
                Console.WriteLine(count + "\t\t" + dep + "\t\t\t" + emp_name1);
                count++;

            }

            if (count == 1)
                Console.WriteLine("\tNO Record Found ");


            Console.ReadKey();
        }
    }
}
```

## 2) Display name, salary and running total salary. The output will be as such:

| Name | Salary | Running Total Salary |
|---|---|---|
| Bob | 8000 | 8000 |
| Maria | 12000 | 20000 |
| Peter | 16000 | 36000 |

.....

Solution:

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query5_2
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 5) \n  Display name, salary and running total
salary.\n\t\tThe output will be as such:\n Name Salary Running Total Salary" +
            "\n  Bob\t8000\t8000" +
            "\n  Maria\t12000\t20000 \n  Peter\t16000\t36000");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            /*
            1) Display dept id along with name of all employees in that department
                The output will be as such:
                Dept ID Employee
                10 Michael, Arnold
                20 Bob,Maria,Peter
                */
            /*
```

```
Select emp_name,emp_salary,sum(emp_salary)Over(order by emp_id) as TotalRunningSalary
from employee

            var temptbl = (from emp in dbe.employees

                           orderby emp.emp_id

                           select new { emp.emp_name }).ToList().Select(x => new { rank =
index++, x.emp_name });
            var result = from tmp in temptbl where tmp.rank % 2 != 0 select new {
tmp.rank, tmp.emp_name };
             */
            int salary = 0;
            var temptbl = (from employee in dbe.employees orderby employee.emp_id select
new { employee.emp_salary, employee.emp_name }).ToList().Select(x => new {
x.emp_name,x.emp_salary ,Total_Running =(salary +=Convert.ToInt32(x.emp_salary)) }); ;
             int count = 1;
            Console.WriteLine("\nRow No \t Name \t\t\tSalary \t\tRunning Total Salary");
            foreach (var dep in temptbl)
            {
                if (count == 1 || count == 4)
                {
                    Console.WriteLine(count + "\t" + dep.emp_name + "\t\t\t" +
dep.emp_salary + "\t\t\t" + dep.Total_Running);

                }
                else if(count == 9)
                {
                    Console.WriteLine(count + "\t" + dep.emp_name + "\t\t" +
dep.emp_salary + "\t\t" + dep.Total_Running);
                }
                else
                {
                    Console.WriteLine(count + "\t" + dep.emp_name + "\t\t\t" +
dep.emp_salary + "\t\t" + dep.Total_Running);
                }

                count++;

            }



            if (count == 1)
                Console.WriteLine("\tNO Record Found ");
            Console.WriteLine("-----------Solution using Lamda Expression and Methods----
--------------");
             salary = 0;
            var temptbl1 = dbe.employees.OrderBy(x => x.emp_id).Select(x => new {
x.emp_salary, x.emp_name }).ToList().Select(x => new { x.emp_name, x.emp_salary,
Total_Running = (salary += Convert.ToInt32(x.emp_salary)) });
                //(from employee in dbe.employees orderby employee.emp_id select new {
employee.emp_salary, employee.emp_name }).ToList().Select(x => new { x.emp_name,
x.emp_salary, Total_Running = (salary += Convert.ToInt32(x.emp_salary)) }); ;
              count = 1;
            Console.WriteLine("\nRow No \t Name \t\t\tSalary \t\tRunning Total Salary");
```

```
            foreach (var dep in temptbl1)
            {
                if (count == 1 || count == 4)
                {
                    Console.WriteLine(count + "\t" + dep.emp_name + "\t\t\t" +
dep.emp_salary + "\t\t\t" + dep.Total_Running);

                }
                else if (count == 9)
                {
                    Console.WriteLine(count + "\t" + dep.emp_name + "\t\t" +
dep.emp_salary + "\t\t" + dep.Total_Running);
                }
                else
                {
                    Console.WriteLine(count + "\t" + dep.emp_name + "\t\t\t" +
dep.emp_salary + "\t\t" + dep.Total_Running);
                }
                count++;

            }


            Console.ReadKey();
        }
    }
}
```

## 3) Display name of first employee, third employee, and so forth.

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query5_3
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();




            Console.WriteLine("Q 5.3) Display name of first employee, third employee, and
so forth.");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            /*
```

```
            1) Display dept id along with name of all employees in that department
              Select emp_name, count(*) over(order by emp_id) from employee

        */



            int index = 1;
            var temptbl = (from emp in dbe.employees

                           orderby emp.emp_id

                           select new {emp.emp_name }).ToList().Select(x=> new
{rank=index++, x.emp_name });
            var result = from tmp in temptbl where tmp.rank % 2 != 0 select new {
tmp.rank, tmp.emp_name };
                           int count = 1;

            foreach (var dep in result)
            {
                  Console.WriteLine("Row No"+count+"\t"+dep);
                count++;

            }

            if (count == 1)
                Console.WriteLine("\tNO Record Found ");
            Console.WriteLine("-----------Solution using Lamda Expression and Methods----
--------------");

             index = 1;
             temptbl = dbe.employees.OrderBy(x=>x.emp_id).Select(x=>new { x.emp_name
}).ToList().Select(x => new { rank = index++, x.emp_name});



             result =  temptbl.Where(x => x.rank % 2 != 0).Select(x => new { x.rank,
x.emp_name });
             count = 1;

            foreach (var dep in result)
            {
                Console.WriteLine("Row No" + count + "\t" + dep);
                count++;

            }

            if (count == 1)
                Console.WriteLine("\tNO Record Found ");



            Console.ReadKey();
        }
    }
```

}

## 4) Display name and department information for all employees in departments 10 and 20 along with department information for departments 30 and 40.

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Qeury5_4
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 5.4) \n 4) Display name and department information for
all employees in departments 10 and 20"
                +"along with department information for departments 30 and 40.");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            /*
            4) Display name and department information for all employees in departments
10 and 20
along with department information for departments 30 and 40.

                */
            /*

               with emp as
                (
                select Employee.emp_name,Department.dept_id,department.dept_name
                from Employee inner join department on
Employee.dept_id=department.dept_id
                where Employee.dept_id in(10,20)
                )select department.dept_id,department.dept_name,emp.emp_name
                from department left outer join emp on department.dept_id=emp.dept_id
                where department.dept_id in(10,20,30,40)
        */


            var innerjoinqry = (from employee in dbe.employees
```

```
                                join department in dbe.departments on employee.dept_id
equals department.dept_id
                                select new { employee.emp_name, department.dept_id,
department.dept_name }).ToList().Where(x=>x.dept_id==10||x.dept_id==20).Select(x=>x);

            var outerjoinqry = (from employee in dbe.employees
                                 join department in dbe.departments on employee.dept_id
equals department.dept_id
                                 select new { employee.emp_name, department.dept_id,
department.dept_name }).ToList().Where(x => x.dept_id == 30 || x.dept_id == 40).Select(x
=> x);
            var uniounqry = innerjoinqry.Union(outerjoinqry);

             int count = 1;
            Console.WriteLine("Row no\t Employee Name\t\t Department id\t\t Department
Name");
            foreach (var dep in uniounqry)
            {

                Console.WriteLine(count + "\t\t" +
dep.emp_name+"\t\t\t"+dep.dept_id+"\t\t\t"+dep.dept_name);
                count++;

            }

            if (count == 1)
                Console.WriteLine("\tNO Record Found ");
            Console.WriteLine("-----------Solution using Lamda Expression and Methods----
--------------");



             innerjoinqry = (dbe.employees.Join(dbe.departments, x => x.dept_id, y =>
y.dept_id, (x, y) =>new { x,y}).Select(x=>
             new { x.x.emp_name,x.y.dept_id,x.y.dept_name}).ToList().Where(x => x.dept_id
== 10 || x.dept_id == 20)).Select(x => x);

            /*(from employee in dbe.employees
                             join department in dbe.departments on employee.dept_id equals
department.dept_id
                             select new { employee.emp_name, department.dept_id,
department.dept_name }).ToList().Where(x => x.dept_id == 10 || x.dept_id == 20).Select(x
=> x);
                             */
            outerjoinqry = (dbe.employees.Join(dbe.departments, x => x.dept_id, y =>
y.dept_id, (x, y) => new { x, y }).Select(x =>
                new { x.x.emp_name, x.y.dept_id, x.y.dept_name }).ToList().Where(x =>
x.dept_id == 30 || x.dept_id == 40)).Select(x => x);
             uniounqry = innerjoinqry.Union(outerjoinqry);

              count = 1;
            Console.WriteLine("Row no\t Employee Name\t\t Department id\t\t Department
Name");
            foreach (var dep in uniounqry)
            {

                Console.WriteLine(count + "\t\t" + dep.emp_name + "\t\t\t" + dep.dept_id
+ "\t\t\t" + dep.dept_name);
```

```
                count++;

            }

            if (count == 1)
                Console.WriteLine("\tNO Record Found ");

            Console.ReadKey();
        }
    }
}
```

## 5)Display names and salaries of the employees with the top five salaries

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query5_5
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 5.5)\n Display names and salaries of the employees with
the top five salaries");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            /*
           select * from employee where emp_salary in(select top(5)* from (
        select distinct(emp_salary) from employee)n order by n.emp_salary desc)

        */
            var topfivesalary = (from employee in dbe.employees orderby
employee.emp_salary descending select employee.emp_salary).Take(5);
            var emp_salary = from employee in dbe.employees where
topfivesalary.Contains(employee.emp_salary) select new { employee.emp_name,
employee.emp_salary };



            int count = 1;
```

```
            foreach (var dep in emp_salary)
            {
                Console.WriteLine("Row No=" + count + "\t" + dep);
                count++;

            }

            if (count == 1)
                Console.WriteLine("\tNO Record Found ");
            Console.WriteLine("-----------Solution using Lamda Expression and Methods----
--------------");

            topfivesalary = dbe.employees.OrderByDescending(x => x.emp_salary).Select(x
=> x.emp_salary).Take(5);
            //(from employee in dbe.employees orderby employee.emp_salary descending
select employee.emp_salary).Take(5);
            emp_salary = dbe.employees.Where(x =>
topfivesalary.Contains(x.emp_salary)).Select(x => new { x.emp_name, x.emp_salary });
                //from employee in dbe.employees where
topfivesalary.Contains(employee.emp_salary) select new { employee.emp_name,
employee.emp_salary };

            count = 1;

            foreach (var dep in emp_salary)
            {
                Console.WriteLine("Row No=" + count + "\t" + dep);
                count++;

            }

            if (count == 1)
                Console.WriteLine("\tNO Record Found ");



            Console.ReadKey();
        }
    }
}
```

6) Display rank the salaries in table employee while allowing for ties
The output will be as such:

| Rank | Salary |
|---|---|
| 1 | 8000 |
| 2 | 9000 |
| 3 | 12000 |
| 3 | 12000 |

.....

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query5_6
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 5.6) \n  Display rank the salaries in table employee
while allowing for ties.\n\t\tThe output will be as such:\n Rank Salary" +
            "\n  1\t8000" +
            "\n  2\t9000 \n  3\t12000  \n  3\t12000");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            /*
           6) Display rank the salaries in table employee while allowing for ties
            The output will be as such:
            Rank Salary
            1 8000
            2 9000
            3 12000
```

```
            3 12000
                */
            /*


                    --Without Using Dense_Rank Function
Select e2.emp_id,e2.emp_name,e2.emp_salary,(Select count(*)+1 from (Select distinct
emp_salary from employee)
e1 where  e1.emp_salary<e2.emp_salary and e1.emp_salary=e1.emp_salary) as [Rank] from
employee e2 order by emp_salary

                */
             var result = from employee in dbe.employees
                          orderby employee.emp_salary
                          select new
                          {
                              employee.emp_id,
                              employee.emp_name,
                              employee.emp_salary
                                   };
            int salary = 0;

              int row = 1;
              int count = 1;
            Console.WriteLine("\nRow No \tRank \t\tSalary ");
            foreach (var dep in result)
            {
                if (row == 1)
                {
                    salary =Convert.ToInt32( dep.emp_salary);
                }

                 if (salary == dep.emp_salary)
                {

                }
                else
                {
                    count++;
                }
                salary = Convert.ToInt32(dep.emp_salary);
                Console.WriteLine(row + " \t" + count+" \t\t"+dep.emp_salary);
                row++;
            }



            if (row == 1)
                Console.WriteLine("\tNO Record Found ");
            Console.WriteLine("-----------Solution using Lamda Expression and Methods----
--------------");
            result = dbe.employees.OrderBy(x => x.emp_salary).Select(x => new { x.emp_id,
x.emp_name, x.emp_salary });


                /*from employee in dbe.employees
                     orderby employee.emp_salary
```

```
                    select new
                    {
                        employee.emp_id,
                        employee.emp_name,
                        employee.emp_salary
                    };*/
                salary = 0;

                row = 1;
                count = 1;
               Console.WriteLine("\nRow No \tRank \t\tSalary ");
               foreach (var dep in result)
               {
                   if (row == 1)
                   {
                       salary = Convert.ToInt32(dep.emp_salary);
                   }

                   if (salary == dep.emp_salary)
                   {

                   }
                   else
                   {
                       count++;
                   }
                   salary = Convert.ToInt32(dep.emp_salary);
                   Console.WriteLine(row + " \t" + count + " \t\t" + dep.emp_salary);
                   row++;
               }



               if (row == 1)
                   Console.WriteLine("\tNO Record Found ");


               Console.ReadKey();
           }
       }
}
```

7) Display the number of employees in each department as a horizontal histogram with each employee represented by an instance of "*". The output will be as such:

| DEPTID | CNT |
|---|---|
| 10 | ** |
| 20 | *** |

.....

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query5_7
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 5.7) \n 7) Display the number of employees in each
department as a horizontal histogram with"+
"each employee represented by an instance of \"*\"\n\t\tThe output will be as such:\n
\t\tDEPTID CNT" +
            "\n \t\t 10\t**" +
            "\n \t\t 20\t***");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            /*
       7) Display the number of employees in each department as a horizontal histogram
with
        each employee represented by an instance of "*".
        The output will be as such:
        DEPTID CNT
        10 **
        20 ***
                */
```

```
            /*


Select count(*)as CNT,dept_id  from employee group by dept_id;

                */

            var query = from employee in dbe.employees
                        group employee by employee.dept_id into g
                        select new
                        {
                            CNT = g.Count(),
                            dept_id = g.Key
                        };
            int row = 1;

            Console.WriteLine("\nRow No \tRank \t\tCNT ");
            foreach (var dep in query)
            {
                string astrick = "";
                for(int i = 1; i <=dep.CNT; i++)
                {
                    astrick += "*";

                }
                Console.WriteLine(row + " \t" + dep.dept_id + " \t\t" + astrick);
                row++;
            }



            if (row == 1)
                Console.WriteLine("\tNO Record Found ");
            Console.WriteLine("-----------Solution using Lamda Expression and Methods----
--------------");
            query = dbe.employees.GroupBy(x => x.dept_id).Select(g => new
            {
                CNT = g.Count(),
                dept_id = g.Key
            });
                /* from employee in dbe.employees
                    group employee by employee.dept_id into g
                    select new
                    {
                        CNT = g.Count(),
                        dept_id = g.Key
                    };*/
             row = 1;

            Console.WriteLine("\nRow No \tRank \t\tCNT ");
            foreach (var dep in query)
            {
                string astrick = "";
                for (int i = 1; i <= dep.CNT; i++)
                {
                    astrick += "*";
```

```
                }
                Console.WriteLine(row + " \t" + dep.dept_id + " \t\t" + astrick);
                row++;
            }



            if (row == 1)
                Console.WriteLine("\tNO Record Found ");









            Console.ReadKey();
        }
    }
}
```

8) Display the number of employees in each department as a vertical histogram with each employee represented by an instance of "*".

The output will be as such:

| D10 | D20 | D30 |
|---|---|---|
| | * | |
| * | * | |
| * | * | * |

.....

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query5_8
```

```
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 5.7) \n 7) Display the number of employees in each
department as a vertical histogram with" +
"each employee represented by an instance of \"*\"\n\t\tThe output will be as such:\n
\t\tD10\tD20\tD30" +
            "\n \t\t \t* \t" +
            "\n \t\t *\t* \t"+
              "\n \t\t *\t* \t*");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-----------------------------------------------------------
--------------");
            /*
           8) Display the number of employees in each department as a vertical histogram
with each
employee represented by an instance of "*".
            The output will be as such:
            D10 D20 D30
            *
            *   *
            *   *   *


            */
            /*

               select D10=MAX(D10),D20=Max(D20),D30=MAX(D30),D40=Max(D40),D50=MAX(D50)
        from
        (select RowNumber=Row_Number() over(Partition by Dept_Id Order by emp_id),
        D10=case when employee.dept_id=10 then '*' else '' end,
        D20=case when employee.dept_id=20 then '*' else '' end,
        D30=case when employee.dept_id=30 then '*' else '' end,
        D40=case when employee.dept_id=40 then '*' else '' end,
        D50=case when employee.dept_id=50 then '*' else '' end
        from employee)temp group by RowNumber order by RowNumber desc
 */
            var query = from employee in dbe.employees orderby employee.emp_id
                        select new
                            {   emp_id=employee.emp_id,
                                dept_id=employee.dept_id,
                                D10 = (employee.dept_id == 10) ? "*" : "B",
                                D20 = (employee.dept_id == 20) ? "*" : "B",
                                D30 = (employee.dept_id == 30) ? "*" : "B",
                                D40 = (employee.dept_id == 40) ? "*" : "B",
                                D50 = (employee.dept_id == 50) ? "*" : "B"
                        };

            Console.WriteLine("D10\tD20\tD30\tD40\tD50");

            foreach (var x in query)
```

```
            {

                Console.WriteLine(x.D10 + "\t" + x.D20 + "\t" + x.D30 + "\t" + x.D40 +
"\t" + x.D50);
            }


            Console.ReadKey();
        }
    }
}
```

## 9)Display employee's name, his department, the number of employees in his department (himself included), and the total number of employees.

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query5_9
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 5.9) \n  Display employee's name, his department, the
number of employees in his department"
+"(himself included), and the total number of employees.");

            Console.WriteLine("-------------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("-------------------------------------------------------------
--------------");
            /*
          Display employee's name, his department, the number of employees in his
department
        (himself included), and the total number of employees.
                */
            /*

              with emp as
            (
            select dept_id,No_of_Emp_In_Dept=count(dept_id) from employee
```

```
            group by dept_id
            )select Employee.emp_name,department.dept_name,emp.No_of_Emp_In_Dept,
            Total_Employees=(Select COUNT(*) from Employee) from emp inner join
            department on emp.dept_id=department.dept_id
            inner join Employee on department.dept_id=Employee.dept_id

        */


            var innerjoinqry = from employee in dbe.employees
                               group employee by employee.dept_id into g
                               select new
                               {
                                   dept_id = g.Key,
                                   No_of_Emp_In_Dept = g.Count()
                               };
            var totalEmployee = (from employee in dbe.employees select
employee.emp_id).Count();
            Console.WriteLine("total emp=" + totalEmployee);

            var outerqury = from emp in innerjoinqry
                            join department in dbe.departments on emp.dept_id equals
department.dept_id
                            join employee in dbe.employees on department.dept_id equals
employee.dept_id
                            select new
                            {
                                employee.emp_name,
                                department.dept_name,
                                emp.No_of_Emp_In_Dept,
                                totalEmployee
                            };
            int count = 1;
            foreach (var dep in outerqury)
            {

                Console.WriteLine(count + "\tEmp Name=" + dep.emp_name+"\tDept
Name="+dep.dept_name+"\tNoOfEmpInDept="+dep.No_of_Emp_In_Dept+"\tTotalEmployee="+dep.tota
lEmployee);
                count++;

            }

            if (count == 1)
                Console.WriteLine("\tNO Record Found ");
            Console.WriteLine("-----------Solution using Lamda Expression and Methods----
--------------");



            innerjoinqry = dbe.employees.GroupBy(x => x.dept_id).Select(g => new
            {
                dept_id = g.Key,
                No_of_Emp_In_Dept = g.Count()
            });

            /*from employee in dbe.employees
                           group employee by employee.dept_id into g
```

```
                            select new
                            {
                                dept_id = g.Key,
                                No_of_Emp_In_Dept = g.Count()
                            };*/
            totalEmployee = dbe.employees.Select(x => x.emp_id).Count();
                /*(from employee in dbe.employees select employee.emp_id).Count();*/
            Console.WriteLine("total emp=" + totalEmployee);

            outerqury = innerjoinqry.Join(dbe.departments, x => x.dept_id, y =>
y.dept_id, (x, y) => new { x, y })
                        .Join(dbe.employees, dx => dx.y.dept_id, emp => emp.dept_id, (dx,
emp) => new { dx, emp }).Select(s =>
                                    new
                        {
                            s.emp.emp_name,
                            s.dx.y.dept_name,
                            s.dx.x.No_of_Emp_In_Dept,
                            totalEmployee
                        });
                /*from emp in innerjoinqry
                             join department in dbe.departments on emp.dept_id equals
department.dept_id
                             join employee in dbe.employees on department.dept_id equals
employee.dept_id
                             select new
                             {
                                 employee.emp_name,
                                 department.dept_name,
                                 emp.No_of_Emp_In_Dept,
                                 totalEmployee
                             };*/
             count = 1;
            foreach (var dep in outerqury)
            {

                Console.WriteLine(count + "\tEmp Name=" + dep.emp_name + "\tDept Name=" +
dep.dept_name + "\tNoOfEmpInDept=" + dep.No_of_Emp_In_Dept + "\tTotalEmployee=" +
dep.totalEmployee);
                count++;

            }



            if (count == 1)
                Console.WriteLine("\tNO Record Found ");

            Console.ReadKey();
        }
    }
}
```

## 10) Display a list of all Fridays for the current year.

Solution:-

```
using Queries;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Query1
{
    class Query5_10
    {
        static void Main(string[] args)
        {
            AIMSEntities dbe = new AIMSEntities();



            Console.WriteLine("Q 5.10) \n) Display a list of all Fridays for the current
year.  ");

            Console.WriteLine("------------------------------------------------------------
--------------");
            Console.WriteLine("Solution using Query Expression");
            Console.WriteLine("------------------------------------------------------------
--------------");
            /*
         Display a list of all Fridays for the current year.
         */
            /*

           select Fridays=dateadd(yy,datediff(YY,0,getdate()),n.num) from
(select top 366 num=ROW_NUMBER() over(order by a.name)-1 from syscolumns a,
syscolumns b)n
where DATENAME(weekday,DATEADD(yy,datediff(yy,0,getdate()),n.num))='Friday'
        */

            DateTime startdate = new DateTime(DateTime.Today.Year,1,1);
            DateTime lastdate = new DateTime(DateTime.Today.Year, 12, 31);
            int row = 1;
          for(DateTime i = startdate; i.Year != DateTime.Today.Year + 1; i =
i.AddDays(1))
            {
                string day = (i).ToString("dddd");
                if (day == "Friday")
                {
                    Console.WriteLine("Row No =" + row + "\tDate=" + i + "\tDay=" + day);
                    row++;
                }
            }


            Console.ReadKey();
        }
    }
}
```