|  |  |
| --- | --- |
| **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**

# 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=>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();

}

}

}

# **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**

# **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();

}

}

}

# ***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();

}

}

}

# **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();

}

}

}

# **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();

}

}

}

# **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**

# **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();

}

}

}

# **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();

}

}

}

# **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=>ordermaster.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();

}

}

}

# **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();

}

}

}

# **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**

# **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();

}

}

}

# **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();

}

}

}

# **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();

}

}

}

# **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,product\_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,product\_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();

}

}

}

# **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,orderdetail.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,orderdetail.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();

}

}

}

# **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();

}

}

}

# **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\*product.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\*product.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();

}

}

}

# **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();

}

}

}

# **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();

}

}

}

# **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();

}

}

}

# **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();

}

}

}

# 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();

}

}

}

# **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.totalEmployee);

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();

}

}

}