# Northwind Database
# Querying with Linq

1. Give the name, address, city, and region of employees.
2. Give the name, address, city, and region of employees living in USA
3. Give the name, address, city, and region of employees older than 50 years old
4. Give the name, address, city, and region of employees that have placed orders to be delivered in Belgium. Write two versions of the query, with and without `join`.
5. Give the employee name and the customer name for orders that are sent by the company 'Speedy Express' to customers who live in Brussels.
6. Give the title and name of employees who have sold at least one of the products 'Gravad Lax' or 'Mishi Kobe Niku'.
7. Give the name and title of employees and the name and title of the person to which they refer (or null for the latter values if they don't refer to another employee).
8. Give the customer name, the product name and the supplier name for customers who live in London and suppliers whose name is 'Pavlova, Ltd.' or 'Karkki Oy'.
9. Give the name of products that were bought or sold by people who live in London. Write two versions of the query, with and without `union`.
10. Give the names of employees who are strictly older than:
    (a) an employee who lives in London.
    (b) any employee who lives in London.
11. Give the name of employees who work longer than any employee of London.
12. Give the name of employees and the city where they live for employees who have sold to customers in the same city.
13. Give the name of customers who have not purchased any product.
14. Give the name of customers who bought all products with price less than 5.
15. Give the name of the products sold by all employees.
16. Give the name of customers who bought all products purchased by the customer whose identifier is 'LAZYK'
17. Give the name of customers who bought exactly the same products as the customer whose identifier is 'LAZYK'
18. Give the average price of products by category.
19. Given the name of the categories and the average price of products in each category.
20. Give the identifier and the name of the companies that provide more than 3 products.
21. Give the identifier, name, and number of orders of employees, ordered by the employee identifier.
22. For each employee give the identifier, name, and the number of distinct products sold, ordered by the employee identifier.
23. Give the identifier, name, and total sales of employees, ordered by the employee identifier.
24. Give the identifier, name, and total sales of employees, ordered by the employee identifier for employees who have sold more than 70 different products.
25. Give the names of employees who sell the products of more than 7 suppliers.
26. Give the customer name and the product name such that the quantity of this product bought by the customer in a single order is more than 5 times the average quantity of this product bought in a single order among all clients.

# Northwind Database
# Querying with Linq
# Answers

1. Give the name, address, city, and region of employees.

```
from E in Employees
select new { E.FirstName, E.LastName, E.Address, E.City, E.Region }
```

2. Give the name, address, city, and region of employees living in USA

```
from E in Employees
where E.Country == "USA"
select new { E.FirstName, E.LastName, E.Address, E.City, E.Region }
```

3. Give the name, address, city, and region of employees older than 50 years old

```
from E in Employees
where E.BirthDate < DateTime.Today.AddYears(-50)
select new { E.FirstName, E.LastName, E.Address, E.City, E.Region }
```

4. Give the name, address, city, and region of employees that have placed orders to be delivered in Belgium. Write two versions of the query, with and without `join`.

```
from E in Employees
from O in E.Orders
where O.ShipCountry == "Belgium"
select new { E.FirstName, E.LastName, E.Address, E.City, E.Region }
```

   Another version

```
from E in Employees
join O in Orders on E.EmployeeID equals O.EmployeeID
where O.ShipCountry == "Belgium"
select new { E.FirstName, E.LastName, E.Address, E.City, E.Region }
```

5. Give the employee name and the customer name for orders that are sent by the company 'Speedy Express' to customers who live in Brussels.

```
from E in Employees
join O in Orders on E.EmployeeID equals O.EmployeeID
join C in Customers on O.CustomerID equals C.CustomerID
join S in Shippers on O.ShipVia equals S.ShipperID
where C.City == "Bruxelles"
where S.CompanyName == "Speedy Express"
select new {E.FirstName, E.LastName, C.CompanyName}
```

6. Give the title and name of employees who have sold at least one of the products 'Gravad Lax' or 'Mishi Kobe Niku'.

```
(from E in Employees
from O in E.Orders
from D in O.OrderDetails
join P in Products on D.ProductID equals P.ProductID
where P.ProductName == "Gravad Lax" || P.ProductName == "Mishi Kobe Niku"
select new {E.Title, E.FirstName, E.LastName}).Distinct()
```

7. Give the name and title of employees and the name and title of the person to which they refer (or null for the latter values if they don't refer to another employee).

```
from E in Employees
join M1 in Employees on E.ReportsTo equals M1.EmployeeID into M2
from M in M2.DefaultIfEmpty()
select new { E.Title, E.FirstName, E.LastName,
    MgrTitle = (M == null ? String.Empty : M.Title),
    MgrFistName = (M == null ? String.Empty : M.FirstName),
    MgrLastName = (M == null ? String.Empty : M.LastName) }
```

8. Give the customer name, the product name and the supplier name for customers who live in London and suppliers whose name is 'Pavlova, Ltd.' or 'Karkki Oy'.

```
(from S in Suppliers
where S.CompanyName == "Pavlova, Ltd." || S.CompanyName == "Karkki Oy"
from P in S.Products
from D in P.OrderDetails
join O in Orders on D.OrderID equals O.OrderID
join C in Customers on O.CustomerID equals C.CustomerID
where C.City == "London"
select new { C.CompanyName, P.ProductName,
    SupplierName = S.CompanyName }).Distinct()
```

Another syntax for the same query

```
(from S in Suppliers.Where( S => S.CompanyName == "Pavlova, Ltd." ||
    S.CompanyName == "Karkki Oy" )
from P in S.Products
from D in P.OrderDetails
join O in Orders on D.OrderID equals O.OrderID
join C in Customers.Where ( C => C.City == "London")
    on O.CustomerID equals C.CustomerID
select new { C.CompanyName, P.ProductName,
    SupplierName = S.CompanyName }).Distinct()
```

9. Give the name of products that were bought or sold by people who live in London. Write two versions of the query, with and without `union`.

```
(from E in Employees.Where( E => E.City == "London" )
from O in E.Orders
from D in O.OrderDetails
```

```
join P in Products on D.ProductID equals P.ProductID
select new { P.ProductName }).
Union(
from C in Customers.Where( C => C.City == "London" )
from O in C.Orders
from D in O.OrderDetails
join P in Products on D.ProductID equals P.ProductID
select new { P.ProductName }).Distinct()
```

Another version

```
(from P in Products
from D in P.OrderDetails
join O in Orders on D.OrderID equals O.OrderID
join E in Employees on O.EmployeeID equals E.EmployeeID
join C in Customers on O.CustomerID equals C.CustomerID
where (E.City == "London") || (C.City == "London" )
select new { P.ProductName }).Distinct()
```

10. Give the names of employees who are strictly older than:

    (a) an employee who lives in London.

    ```
    from E1 in Employees
    where E1.BirthDate <
    (from E2 in Employees.Where ( E2 => E2.City == "London" )
    select E2.BirthDate).Max()
    select new { E1.FirstName, E1.LastName }
    ```

    (b) any employee who lives in London.

    ```
    from E1 in Employees
    where E1.BirthDate <
    (from E2 in Employees.Where ( E2 => E2.City == "London" )
    select E2.BirthDate).Min()
    select new { E1.FirstName, E1.LastName }
    ```

11. Give the name of employees who work longer than any employee of London.

```
from E1 in Employees
where E1.HireDate <
(from E2 in Employees.Where ( E2 => E2.City == "London" )
select E2.HireDate).Min()
select new { E1.FirstName, E1.LastName }
```

12. Give the name of employees and the city where they live for employees who have sold to customers in the same city.

```
(from E in Employees
from O in E.Orders
join C in Customers on O.CustomerID equals C.CustomerID
where E.City == C.City
select new { E.FirstName, E.LastName, E.City }).Distinct()
```

13. Give the name of customers who have not purchased any product.

```
from C in Customers
where C.Orders.Count() == 0
select new { C.CompanyName }
```

Another version

```
from C in Customers
where !C.Orders.Any()
select new { C.CompanyName }
```

14. Give the name of customers who bought all products with price less than 5.

```
from C in Customers
let allProducts = from P in Products.Where
  ( P => P.UnitPrice < 5) select P.ProductID
where !allProducts.Except(
  from O in C.Orders
  from D in O.OrderDetails
  select D.ProductID).Any()
select C.CompanyName
```

15. Give the name of the products sold by all employees.

```
from P in Products
let allEmployees = from E in Employees select E.EmployeeID
where !allEmployees.Except(
  from D in P.OrderDetails
  join O in Orders on D.OrderID equals O.OrderID
  join E in Employees on O.EmployeeID equals E.EmployeeID
  select E.EmployeeID ).Any()
select P.ProductName
```

16. Give the name of customers who bought all products purchased by the company whose identifier is 'LAZYK'

```
from C in Customers
where C.CustomerID != "LAZYK"
let allProdsCustomer =
  from O in C.Orders
  from D in O.OrderDetails
  select D.ProductID
let allProdsLazyk =
  from C1 in Customers
  where C1.CustomerID == "LAZYK"
  from O1 in C1.Orders
  from D1 in O1.OrderDetails
  select D1.ProductID
where !allProdsLazyk.Except(allProdsCustomer).Any()
select C.CompanyName
```

17. Give the name of customers who bought exactly the same products as the company whose identifier is 'LAZYK'

```
from C in Customers
where C.CustomerID != "LAZYK"
let allProdsCustomer =
  from O in C.Orders
  from D in O.OrderDetails
  select D.ProductID
let allProdsLazyk =
  from C1 in Customers
  where C1.CustomerID == "LAZYK"
  from O1 in C1.Orders
  from D1 in O1.OrderDetails
  select D1.ProductID
where !allProdsLazyk.Except(allProdsCustomer).Any()
where !allProdsCustomer.Except(allProdsLazyk).Any()
select C.CompanyName
```

18. Give the average price of products by category.

```
from P in Products
group P by P.CategoryID into categProds
select new { categProds.Key,
    AvgPrice = categProds.Average(C => C.UnitPrice) }
```

19. Given the name of the categories and the average price of products in each category.

```
from P in Products
join C in Categories on P.CategoryID equals C.CategoryID
group P by P.Category.CategoryName into categProds
select new { categProds.Key,
    AvgPrice = categProds.Average(C => C.UnitPrice) }
```

20. Give the identifier and the name of the companies that provide more than 3 products.

```
from S in Suppliers
where S.Products.Count() > 3
select new { S.SupplierID, S.CompanyName }
```

21. Give the identifier, name, and number of orders of employees, ordered by the employee identifier.

```
from E in Employees
orderby E.EmployeeID
select new { E.EmployeeID, E.FirstName, E.LastName,
  NbOrders = E.Orders.Count() }
```

22. For each employee give the identifier, name, and the number of distinct products sold, ordered by the employee identifier.

```
from E in Employees
orderby E.EmployeeID
select new { E.EmployeeID, E.FirstName, E.LastName, NbProds =
  (from O in E.Orders
  from D in O.OrderDetails
  select D.ProductID).Distinct().Count() }
```

23. Give the identifier, name, and total sales of employees, ordered by the employee identifier.

```
from E in Employees
orderby E.EmployeeID
select new { E.EmployeeID, E.FirstName, E.LastName, TotalSales =
  (from O in E.Orders
  from D in O.OrderDetails
  let LineTotal = System.Convert.ToDouble(D.UnitPrice) *
      D.Quantity * (1 - D.Discount)
  select LineTotal).Sum() }
```

24. Give the identifier, name, and total sales of employees, ordered by the employee identifier for employees who have sold more than 70 different products.

```
from E in Employees
let nbProds =
  (from O in E.Orders
  from D in O.OrderDetails
  select D.ProductID).Distinct().Count()
where nbProds > 70
orderby E.EmployeeID
select new { E.EmployeeID, E.FirstName, E.LastName, TotalSales =
  (from O in E.Orders
  from D in O.OrderDetails
  let LineTotal = System.Convert.ToDouble(D.UnitPrice) *
      D.Quantity * (1 - D.Discount)
  select LineTotal).Sum() }
```

25. Give the names of employees who sell the products of more than 7 suppliers.

```
from E in Employees
let nbSuppliers =
  (from O in E.Orders
  from D in O.OrderDetails
  join P in Products on D.ProductID equals P.ProductID
  select P.SupplierID).Distinct().Count()
where nbSuppliers > 7
select new { E.FirstName, E.LastName }
```

26. Give the customer name and the product name such that the quantity of this product bought by the customer in a single order is more than 5 times the average quantity of this product bought in a single order among all clients.

```
from C in Customers
from O in C.Orders
from D in O.OrderDetails
join P in Products on D.ProductID equals P.ProductID
orderby C.CompanyName, P.ProductName
let avgSales =
  (from D1 in OrderDetails
```

```
  where D1.ProductID == P.ProductID
  select D1.Quantity).Cast<int>().Average()
where D.Quantity > 5 * avgSales
select new { C.CompanyName, P.ProductName }
```

| Query | Nb of Answers |
|---|---|
| 1 | 9 |
| 2 | 5 |
| 3 | 6 |
| 4 | 19 |
| 5 | 2 |
| 6 | 6 |
| 7 | 9 |
| 8 | 9 |
| 9 | 76 |
| 10 | a) 8 |
| | b) 3 |
| 11 | 4 |
| 12 | 6 |
| 13 | 2 |
| 14 | 9 |
| 15 | 27 |
| 16 | 10 |
| 17 | 0 |
| 18 | 8 |
| 19 | 8 |
| 20 | 4 |
| 21 | 9 |
| | Davolio 123 |
| 22 | 9 |
| | Davolio 72 |
| 23 | 9 |
| | Davolio 192 107 |
| 24 | 3 |
| 25 | 9 |
| 26 | 3 |