Here are **50 different LINQ queries** that cover a variety of LINQ operations on the **Northwind database**. These queries include LINQ operations like filtering, projection, grouping, joining, sorting, and aggregation. I'll assume the Northwind database has common tables such as Customers, Orders, OrderDetails, Products, Categories, Employees, and Suppliers.

**Filtering**

1. **Customers from a specific city**
2. var customersInLondon = db.Customers.Where(c => c.City == "London");
3. **Orders placed after a specific date**
4. var recentOrders = db.Orders.Where(o => o.OrderDate > new DateTime(2023, 1, 1));
5. **Products with stock greater than 50**
6. var highStockProducts = db.Products.Where(p => p.UnitsInStock > 50);
7. **Employees hired in the last year**
8. var recentHires = db.Employees.Where(e => e.HireDate > DateTime.Now.AddYears(-1));
9. **Suppliers from the USA**
10. var usSuppliers = db.Suppliers.Where(s => s.Country == "USA");

**Projection**

1. **Customer names and cities**
2. var customerInfo = db.Customers.Select(c => new { c.ContactName, c.City });
3. **Product names and prices**
4. var productInfo = db.Products.Select(p => new { p.ProductName, p.UnitPrice });
5. **Order IDs and order dates**
6. var orderDetails = db.Orders.Select(o => new { o.OrderID, o.OrderDate });
7. **Employee full names**
8. var employeeNames = db.Employees.Select(e => new { FullName = e.FirstName + " " + e.LastName });
9. **Category names only**
10. var categoryNames = db.Categories.Select(c => c.CategoryName);

**Sorting**

1. **Products sorted by price**
2. var sortedProducts = db.Products.OrderBy(p => p.UnitPrice);
3. **Customers sorted by name in descending order**
4. var customersByNameDesc = db.Customers.OrderByDescending(c => c.ContactName);
5. **Orders sorted by date, then by customer ID**
6. var sortedOrders = db.Orders.OrderBy(o => o.OrderDate).ThenBy(o => o.CustomerID);
7. **Employees sorted by hire date**
8. var employeesByHireDate = db.Employees.OrderBy(e => e.HireDate);
9. **Suppliers sorted by country and company name**
10. var sortedSuppliers = db.Suppliers.OrderBy(s => s.Country).ThenBy(s => s.CompanyName);

**Grouping**

1. **Products grouped by category**
2. var productsByCategory = db.Products.GroupBy(p => p.CategoryID);
3. **Orders grouped by customer**
4. var ordersByCustomer = db.Orders.GroupBy(o => o.CustomerID);
5. **Employees grouped by title**
6. var employeesByTitle = db.Employees.GroupBy(e => e.Title);
7. **Suppliers grouped by country**
8. var suppliersByCountry = db.Suppliers.GroupBy(s => s.Country);
9. **Order details grouped by product**
10. var orderDetailsByProduct = db.OrderDetails.GroupBy(od => od.ProductID);

**Joining**

1. **Customers with their orders**
2. var customerOrders = db.Customers.Join(db.Orders,
3. c => c.CustomerID,
4. o => o.CustomerID,
5. (c, o) => new { c.ContactName, o.OrderID, o.OrderDate });
6. **Products with their categories**
7. var productsWithCategories = db.Products.Join(db.Categories,
8. p => p.CategoryID,
9. c => c.CategoryID,
10. (p, c) => new { p.ProductName, c.CategoryName });
11. **Orders with their order details**
12. var ordersWithDetails = db.Orders.Join(db.OrderDetails,
13. o => o.OrderID,
14. od => od.OrderID,
15. (o, od) => new { o.OrderID, od.ProductID, od.Quantity });
16. **Employees with their territories**
17. var employeeTerritories = db.Employees.Join(db.EmployeeTerritories,
18. e => e.EmployeeID,
19. et => et.EmployeeID,
20. (e, et) => new { e.FirstName, et.TerritoryID });
21. **Suppliers with their products**
22. var suppliersWithProducts = db.Suppliers.Join(db.Products,
23. s => s.SupplierID,
24. p => p.SupplierID,
25. (s, p) => new { s.CompanyName, p.ProductName });

**Set Operations**

1. **Distinct product categories**
2. var distinctCategories = db.Products.Select(p => p.CategoryID).Distinct();
3. **Union of two customer lists**
4. var allCustomers = customersList1.Union(customersList2);
5. **Intersection of two product lists**
6. var commonProducts = productsList1.Intersect(productsList2);
7. **Difference of two employee lists**
8. var exclusiveEmployees = employeesList1.Except(employeesList2);
9. **Concat two supplier lists**
10. var combinedSuppliers = suppliersList1.Concat(suppliersList2);

**Aggregation**

1. **Count of products**
2. var productCount = db.Products.Count();
3. **Sum of quantities in order details**
4. var totalQuantity = db.OrderDetails.Sum(od => od.Quantity);
5. **Average product price**
6. var averagePrice = db.Products.Average(p => p.UnitPrice);
7. **Maximum product price**
8. var maxPrice = db.Products.Max(p => p.UnitPrice);
9. **Minimum product price**
10. var minPrice = db.Products.Min(p => p.UnitPrice);

**Element Operations**

1. **First product with stock**
2. var firstInStock = db.Products.First(p => p.UnitsInStock > 0);
3. **Last order of a customer**
4. var lastOrder = db.Orders.Where(o => o.CustomerID == "ALFKI").OrderByDescending(o => o.OrderDate).First();
5. **Single product by ID**
6. var specificProduct = db.Products.Single(p => p.ProductID == 1);
7. **FirstOrDefault for a non-existent product**
8. var nonExistentProduct = db.Products.FirstOrDefault(p => p.ProductName == "NonExistent");
9. **ElementAt for the 5th customer**
10. var fifthCustomer = db.Customers.ElementAt(4);

**Quantifiers**

1. **Any products out of stock**
2. var anyOutOfStock = db.Products.Any(p => p.UnitsInStock == 0);
3. **All orders shipped**
4. var allShipped = db.Orders.All(o => o.ShippedDate != null);

**Partitioning**

1. **Take top 10 orders**
2. var topOrders = db.Orders.Take(10);
3. **Skip first 5 customers**
4. var skipFirstFive = db.Customers.Skip(5);
5. **Orders in second page (10 per page)**
6. var page2Orders = db.Orders.Skip(10).Take(10);

**Miscellaneous**

1. **Join with a condition**
2. var conditionalJoin = db.Products.Join(db.Categories,
3. p => p.CategoryID,
4. c => c.CategoryID,
5. (p, c) => new { p.ProductName, c.CategoryName })
6. .Where(pc => pc.CategoryName == "Beverages");
7. **GroupJoin example**
8. var customersWithOrders = db.Customers.GroupJoin(db.Orders,
9. c => c.CustomerID,
10. o => o.CustomerID,
11. (c, orders) => new { c.ContactName, Orders = orders });
12. **SelectMany for customer orders**
13. var customerOrders = db.Customers.SelectMany(c => db.Orders.Where(o => o.CustomerID == c.CustomerID));
14. **Left join**
15. var leftJoin = db.Customers.GroupJoin(db.Orders,
16. c => c.CustomerID,
17. o => o.CustomerID,
18. (c, orders) => new { c.ContactName, Orders = orders.DefaultIfEmpty() });
19. **Cross join**
20. var crossJoin = db.Customers.SelectMany(c => db.Products, (c, p) => new { c.ContactName, p.ProductName });

Let me know if you'd like further explanation or details on any of these!