Basic Select Queries

// 1. Get all customers

var customers = db.Customers.ToList();

// 2. Get all product names

var productNames = db.Products.Select(p => p.ProductName).ToList();

// 3. Get all orders with order date and shipped date

var orders = db.Orders.Select(o => new { o.OrderID, o.OrderDate, o.ShippedDate }).ToList();

// 4. Get top 5 most expensive products

var topProducts = db.Products.OrderByDescending(p => p.UnitPrice).Take(5).ToList();

// 5. Get distinct countries from customers

var distinctCountries = db.Customers.Select(c => c.Country).Distinct().ToList();

Filtering Queries (Where Clause)

// 6. Get customers from USA

var usCustomers = db.Customers.Where(c => c.Country == "USA").ToList();

// 7. Get orders placed in 1997

var orders1997 = db.Orders.Where(o => o.OrderDate.Value.Year == 1997).ToList();

// 8. Get products with price greater than $50

var expensiveProducts = db.Products.Where(p => p.UnitPrice > 50).ToList();

// 9. Get employees hired after 1995

var employeesAfter1995 = db.Employees.Where(e => e.HireDate.Value.Year > 1995).ToList();

// 10. Get customers whose name starts with 'A'

var customersWithA = db.Customers.Where(c => c.ContactName.StartsWith("A")).ToList();

Sorting Queries (OrderBy & OrderByDescending)

// 11. Get products sorted by price (ascending)

var sortedProducts = db.Products.OrderBy(p => p.UnitPrice).ToList();

// 12. Get employees sorted by hire date (descending)

var sortedEmployees = db.Employees.OrderByDescending(e => e.HireDate).ToList();

// 13. Get customers sorted by contact name

var sortedCustomers = db.Customers.OrderBy(c => c.ContactName).ToList();

// 14. Get top 3 most ordered products

var topOrderedProducts = db.OrderDetails

.GroupBy(od => od.ProductID)

.Select(g => new { ProductID = g.Key, TotalQuantity = g.Sum(od => od.Quantity) })

.OrderByDescending(g => g.TotalQuantity)

.Take(3)

.ToList();

Aggregation Queries (Count, Sum, Min, Max, Avg)

// 15. Count total customers

var totalCustomers = db.Customers.Count();

// 16. Get total orders in Northwind

var totalOrders = db.Orders.Count();

// 17. Get total revenue

var totalRevenue = db.OrderDetails.Sum(od => od.Quantity \* od.UnitPrice);

// 18. Get highest product price

var maxPrice = db.Products.Max(p => p.UnitPrice);

// 19. Get lowest product price

var minPrice = db.Products.Min(p => p.UnitPrice);

// 20. Get average product price

var avgPrice = db.Products.Average(p => p.UnitPrice);

Grouping Queries (GroupBy)

// 21. Get number of customers per country

var customersByCountry = db.Customers

.GroupBy(c => c.Country)

.Select(g => new { Country = g.Key, Count = g.Count() })

.ToList();

// 22. Get total orders per customer

var ordersPerCustomer = db.Orders

.GroupBy(o => o.CustomerID)

.Select(g => new { CustomerID = g.Key, TotalOrders = g.Count() })

.ToList();

// 23. Get total sales per product

var salesPerProduct = db.OrderDetails

.GroupBy(od => od.ProductID)

.Select(g => new { ProductID = g.Key, TotalSales = g.Sum(od => od.Quantity \* od.UnitPrice) })

.ToList();

// 24. Get orders per year

var ordersPerYear = db.Orders

.GroupBy(o => o.OrderDate.Value.Year)

.Select(g => new { Year = g.Key, OrderCount = g.Count() })

.ToList();

Joins (Inner Join, Left Join)

// 25. Get all orders with customer details (Inner Join)

var ordersWithCustomers = db.Orders

.Join(db.Customers,

order => order.CustomerID,

customer => customer.CustomerID,

(order, customer) => new { order.OrderID, customer.ContactName, order.OrderDate })

.ToList();

// 26. Get all products with their category name

var productsWithCategory = db.Products

.Join(db.Categories,

product => product.CategoryID,

category => category.CategoryID,

(product, category) => new { product.ProductName, category.CategoryName })

.ToList();

// 27. Get employees and their managers (Self Join)

var employeesWithManagers = db.Employees

.Join(db.Employees,

emp => emp.ReportsTo,

manager => manager.EmployeeID,

(emp, manager) => new { emp.FirstName, emp.LastName, ManagerName = manager.FirstName })

.ToList();

Set Operators (Union, Intersect, Except)

// 28. Get customers from USA or UK

var usUkCustomers = db.Customers.Where(c => c.Country == "USA")

.Union(db.Customers.Where(c => c.Country == "UK"))

.ToList();

// 29. Get common customers who placed orders

var customersWithOrders = db.Customers.Select(c => c.CustomerID)

.Intersect(db.Orders.Select(o => o.CustomerID))

.ToList();

// 30. Get customers who never placed an order

var customersWithoutOrders = db.Customers.Select(c => c.CustomerID)

.Except(db.Orders.Select(o => o.CustomerID))

.ToList();

Advanced Queries (Subqueries, Any, All)

// 31. Get customers who have placed at least one order

var customersWithOrders = db.Customers

.Where(c => db.Orders.Any(o => o.CustomerID == c.CustomerID))

.ToList();

// 32. Get products that were never ordered

var neverOrderedProducts = db.Products

.Where(p => !db.OrderDetails.Any(od => od.ProductID == p.ProductID))

.ToList();

// 33. Get employees who manage others

var employeesWithSubordinates = db.Employees

.Where(e => db.Employees.Any(emp => emp.ReportsTo == e.EmployeeID))

.ToList();