Select Single

```
var result = db.Set<Product>().Select(x => x.Code).ToList();

SELECT Code FROM Product
```

Multiple

```
var result = db.Set<Product>()
  .Select(x => new
  {
    x.Code,
    x.Description,
    x.SalesPrice
  }).ToList();

SELECT Code, Description, SalesPrice FROM Product
```

All

```
var result = db.Set<Product>().ToList();

SELECT * FROM Product
```

Alias

```
var result = db.Set<Product>()
  .Select(x => new
  {
    x.Code,
    x.Description,
    TotalCost = x.Stock * x.Cost
  }).ToList();

SELECT  Code,  Description,  Stock * Cost as TotalCostFROM Product
```

Where

Equality

```
var result = db.Set<Product>()
  .Where(x => x.Code == "1001")
  .ToList();

SELECT * FROM Product
WHERE Code = '1001'
```

Contains/Like

```
var result = db.Set<Product>()
  .Where(x => x.Description.Contains("MIE"))
  .ToList();

SELECT * FROM Product WHERE Name LIKE '%MIE%'
```

StartsWith

```
var result = db.Set<Product>()
  .Where(x => x.Description.StartsWith("ABC"))
  .ToList();

SELECT * FROM Product WHERE Code LIKE 'ABC%'
```

EndsWith

```
var result = db.Set<Product>()
  .Where(x => x.Description.EndsWith("ABC"))
  .ToList();

SELECT * FROM Product WHERE Code LIKE '%ABC'
```

Contains/In

```
var arr = new string[] {"A001", "B001", "C001" };var result = db.Set<Product>()
  .Where(x => arr.Contains(x.Code))
  .ToList();

SELECT * FROM Product WHERE Code IN ('A001', 'B001', 'C001')
```

Date Range

```
var start = new DateTime(2022, 1, 1);
var end = new DateTime(2022, 12, 31);var result = db.Set<Product>()
  .Where(x => x.DateCreated.Date >= start && x.DateCreated.Date <= end)
  .ToList();

SELECT * FROM Product
WHERE DateCreated >= '2022-01-01' AND DateCreated <= '2022-12-31'
```

Count

```
var result = db.Set<Product>()
  .Count(x => x.Cost > 1000);

SELECT Count(*) FROM Product WHERE Cost > 1000
```

Sum

```
var result = db.Set<Product>()
  .Sum(x => x.Cost);

SELECT SUM(Cost) FROM Product
```

Min

```
var result = db.Set<Product>()
  .Min(x => x.Cost);

SELECT MIN(Cost) FROM Product
```

Max

```
var result = db.Set<Product>()
  .Max(x => x.Cost);

SELECT MAX(Cost) FROM Product
```

Average

```
var result = db.Set<Product>()
  .Average(x => x.Cost);

SELECT Average(Cost) FROM Product
```

Order

Ascending

```
var result = db.Set<Product>()
  .OrderBy(x => x.Code)
  .ToList();

SELECT * FROM Product ORDER BY Code
```

Descending

```
var result = db.Set<Product>()
  .OrderByDescending(x => x.Code)
  .ToList();

SELECT * FROM Product ORDER BY Code DESC
```

Multiple

```
var result = db.Set<Product>()
  .OrderBy(x => x.Code)
  .ThenBy(x => x.Description)
  .ToList();

SELECT * FROM Product ORDER BY Code, Description
```

Group

Single

```
var result = db.Set<Product>()
  .GroupBy(x => x.Category)
  .Select(x => new
  {
    Category = x.Key,
    TotalCost = x.Sum(a => a.Cost)
  }).ToList();

SELECT Category, SUM(Cost) as TotalCost FROM Product GROUP BY Category
```

Multiple

```
var result = db.Set<Product>()
  .GroupBy(x => new
  {
    x.Category,
    x.Supplier
  })
  .Select(x => new
  {
    Category = x.Key,
    TotalCost = x.Sum(a => a.Cost)
  }).ToList();

SELECT Category, Supplier, SUM(Cost) as TotalCost FROM Product GROUP BY Category, Supplier
```

Paging

```
var page = 3;
var size = 10;
var skip = (page - 1) * size;
var take = size;var result = db.Set<Product>()
  .OrderBy(x => x.Code)
  .Skip(skip)
  .Take(take)
  .ToList();

SELECT * FROM Product ORDER BY Code OFFSET 20 ROWS FETCH NEXT 10 ROWS ONLY
```

Join

```
var result = db.Set<SalesOrder>()
  .Select(x => new
  {
    x.TransactionDate,
    CustomerName = x.Customer.Name,
    ProductCode = x.Product.Code,
    x.Total
  }).ToList();

SELECT
  s.TransactionDate,  c.Name as CustomerName,  p.Code as ProductCode
FROM SalesOrder s
LEFT JOIN Customer c ON s.CustomerId = c.Id
LEFT JOIN Product p ON s.ProductId = p.Id
```

Complex Query

```
var start = new DateTime(2022, 1, 1);
var end = new DateTime(2022, 12, 31);var result = db.Set<SalesOrder>()
  .Where(x.TransactionDate >= start && x.TransactionDate <= end)
  .GroupBy(x => new
  {
    CustomerName = x.Customer.Name,
    ProductCode = x.Product.Code
  })
  .Select(x => new
  {
    CustomerName = x.Key.CustomerName,
    ProductCode = x.Key.ProductCode,
    TotalSales = x.Sum(x.Total)
  })
  .OrderBy(x => x.CustomerName)
  .ThenByDescending(x => x.ProductCode)
  .ToList();

SELECT c.Name as CustomerName, p.Code as ProductCode, SUM(s.Total) AS TotalSales FROM
SalesOrder s
LEFT JOIN Customer c ON s.CustomerId = c.Id
LEFT JOIN Product p ON s.ProductId = p.Id
WHERE s.TransactionDate >= '2022-01-01' AND s.TransactionDate <= '2022-12-31'
GROUP By c.Name, p.Code
ORDER BY c.Name, p.Code DESC
```