Entity Framework Core

Microsoft

.NET

a lightweight, extensible, and cross-platform version of the popular data access technology

By R&D Team

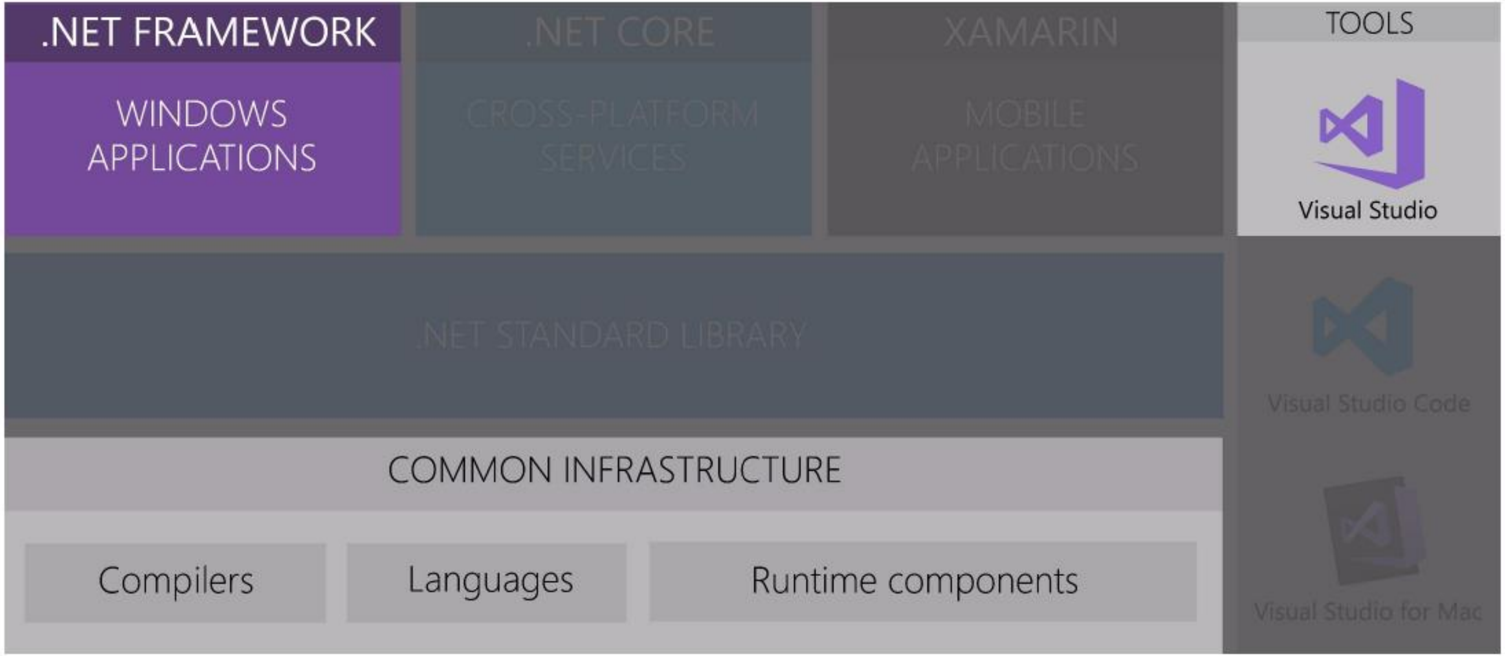
EF Core

Overview

EF Core can serve as an **object-relational mapper (O/RM)** enabling .NET developers to work with a database using .NET objects, and eliminating the need for most of the data-access code they usually need to write.

EF Core

.NET Landscape (EF)



.NET FRAMEWORK

WINDOWS APPLICATIONS

.NET CORE

CROSS-PLATFORM SERVICES

XAMARIN

MOBILE APPLICATIONS

TOOLS

Visual Studio

.NET STANDARD LIBRARY

Visual Studio Code

COMMON INFRASTRUCTURE

Compilers

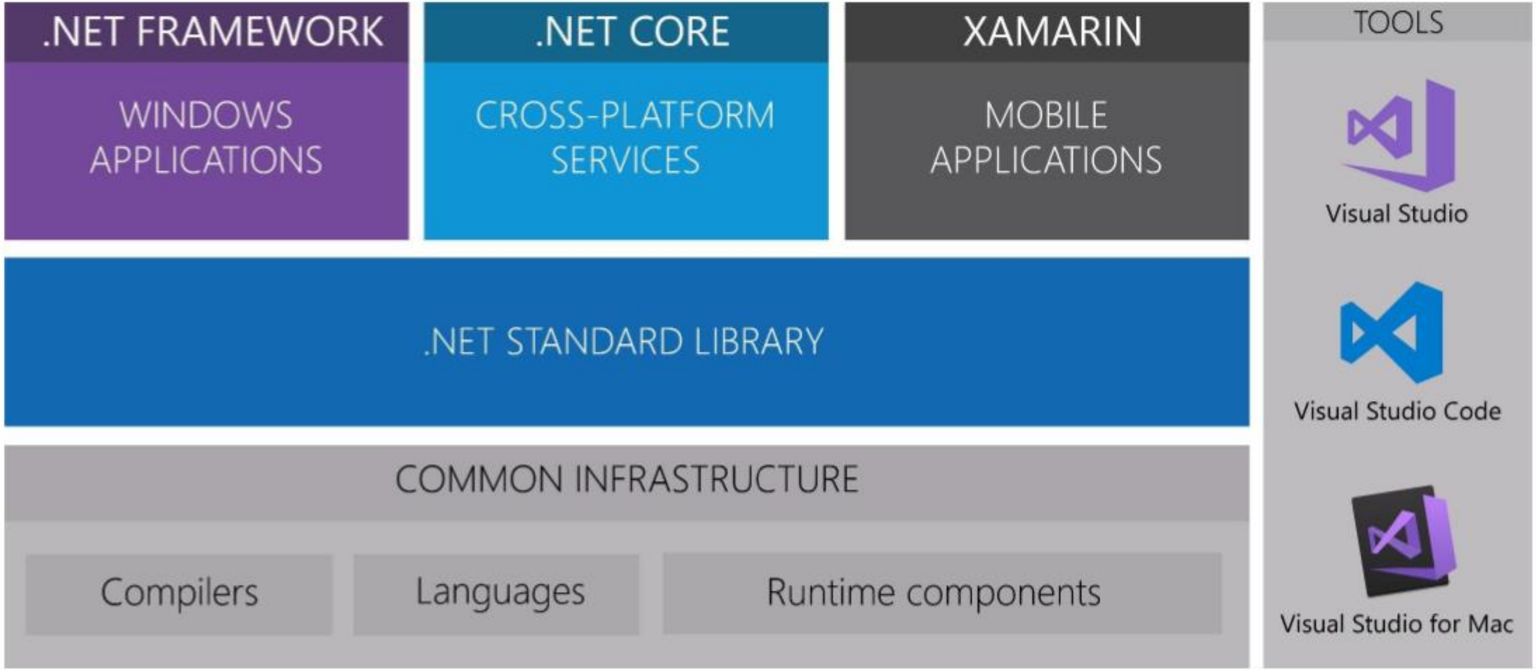
Languages

Runtime components

Visual Studio for Mac

EF Core

.NET Landscape (EF Core)



.NET FRAMEWORK

WINDOWS APPLICATIONS

.NET CORE

CROSS-PLATFORM SERVICES

XAMARIN

MOBILE APPLICATIONS

.NET STANDARD LIBRARY

COMMON INFRASTRUCTURE

Compilers

Languages

Runtime components

TOOLS

Visual Studio

Visual Studio Code

Visual Studio for Mac

EF Core

Roadmap

The stable version of EF Core 2.1 was released on May 30, 2018

Future releases

EF Core 2.2
This release will include many bug fixes, and relatively small number of new features. Details about this release are included in the the EF Core 2.2 roadmap announcement.

EF Core 3.0
Although we have not completed the release planning process for the next release after 2.2, we are currently planning to have a major release, aligned with .NET Core 3.0 and ASP.NET 3.0

EF Core

Database Providers

Entity Framework Core can access many different databases through plug-in libraries called database providers.

- SQL Server
- SQLite
- EF Core in-memory database
- PostgreSQL
- MySQL

Ref. https://docs.microsoft.com/en-us/ef/core/providers/index

EF Core

Get Started (Installation)

Install the NuGet package for the database provider.

```
Install-Package Microsoft.EntityFrameworkCore.SqlServer

Install-Package Microsoft.EntityFrameworkCore.SqlServer.Design

Install-Package Microsoft.EntityFrameworkCore.Tools
```

EF Core

Get Started

Generate database context.

```
Scaffold-DbContext
-Connection "Server={ipaddress};Database={databasename};user id={username};password={password}"
-Provider Microsoft.EntityFrameworkCore.SqlServer
-OutputDir Entities
-Force
```

EF Core

Scaffold-DbContext

- Connection *<String>*
The connection string to the database.

- Provider *<String>*
The provider to use.
(E.g. Microsoft.EntityFrameworkCore.SqlServer)

EF Core

Scaffold-DbContext

- **OutputDir** *<String>*
The directory to put files in. Paths are relative to the project directory.

- **Context** *<String>*
The name of the database context generate.

EF Core

Scaffold-DbContext

- Schemas *<String[]>*

The schemas of tables to generate entity types for.

- Tables *<String[]>*

The tables to generate entity types for.

EF
Core

Scaffold-DbContext

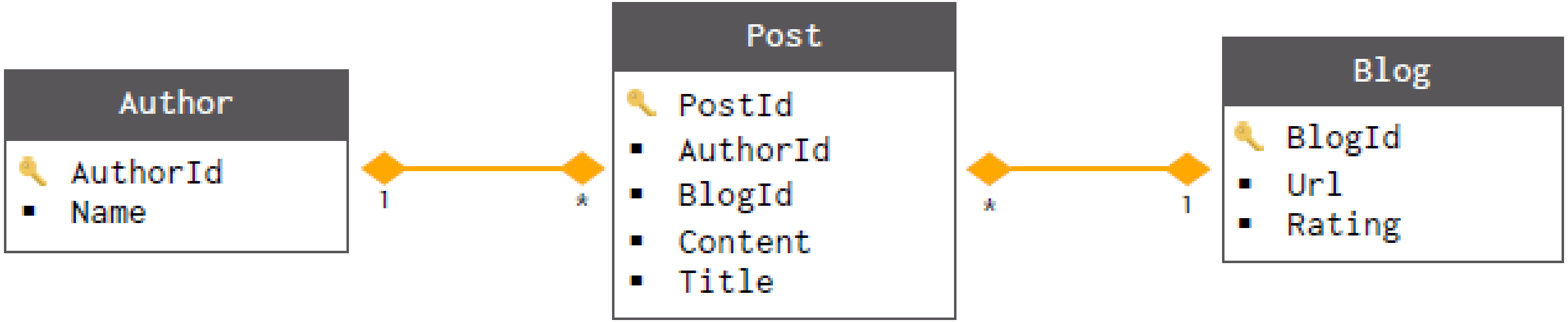
- Force
Overwrite existing files.

EF Core

Workshop 1

Add ef core to project.

IP Address : 192.168.99.19
Username : training
Password : asdf+1234
Database Name : Blogging
DbContext : BloggingContext



Author
- AuthorId
- Name

1 *

Post
- PostId
- AuthorId
- BlogId
- Content
- Title

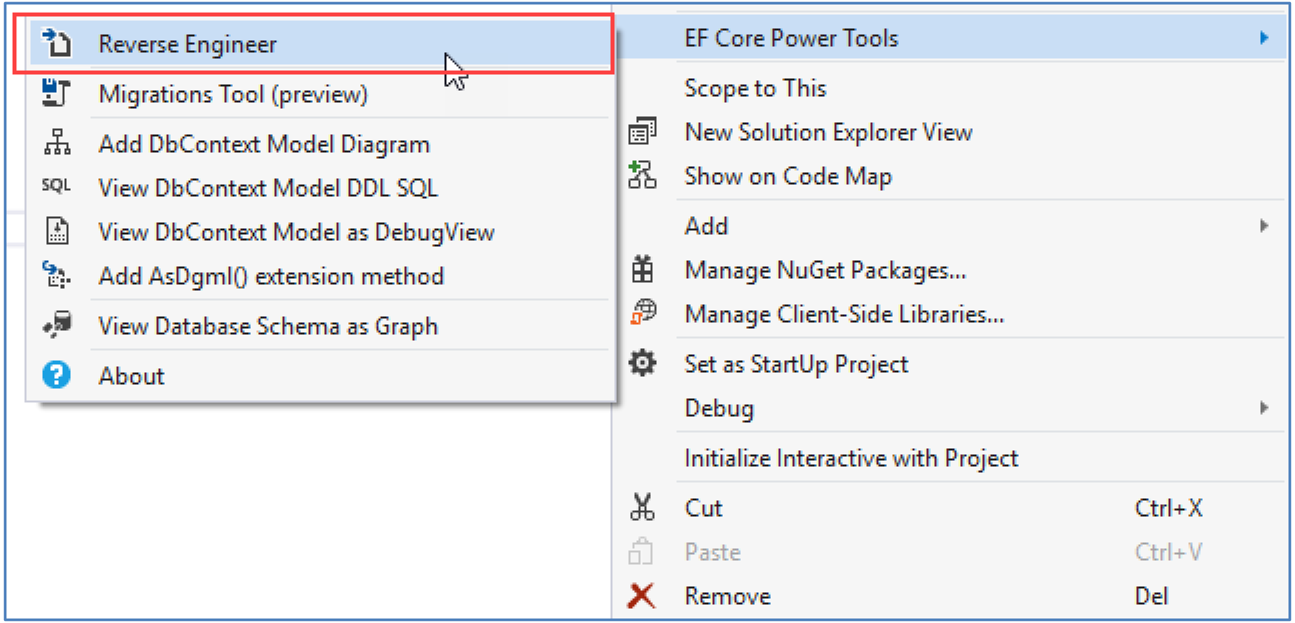
* 1

Blog
- BlogId
- Url
- Rating

EF Core

EF Core Power Tools

Easy to generate database context



Reverse Engineer
Migrations Tool (preview)
Add DbContext Model Diagram
View DbContext Model DDL SQL
View DbContext Model as DebugView
Add AsDgml() extension method
View Database Schema as Graph
About
EF Core Power Tools
Scope to This
New Solution Explorer View
Show on Code Map
Add
Manage NuGet Packages...
Manage Client-Side Libraries...
Set as StartUp Project
Debug
Initialize Interactive with Project
Cut Ctrl+X
Paste Ctrl+V
Remove Del

Ref : https://marketplace.visualstudio.com/items?itemName=ErikEJ.EFCorePowerTools

EF
Core

Mini Profiler EF Core

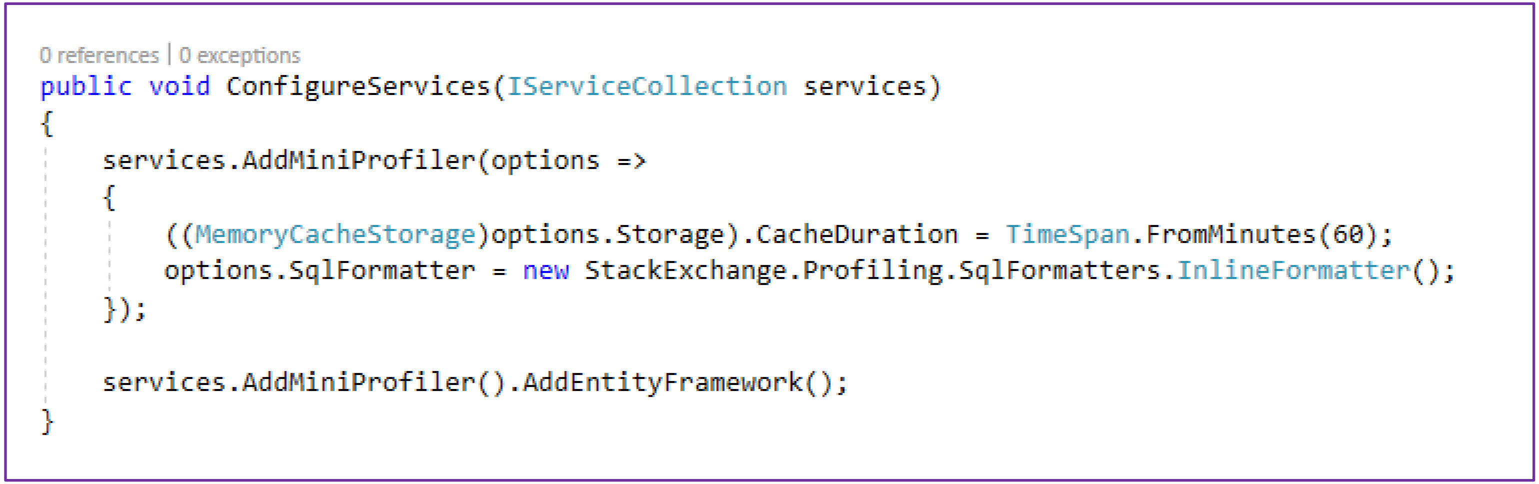
Install nuget package

```
Install-Package MiniProfiler.AspNetCore.Mvc
Install-Package MiniProfiler.EntityFrameworkCore
```

EF Core

Mini Profiler EF Core

Startup.cs register miniprofiler on ConfigureServices function

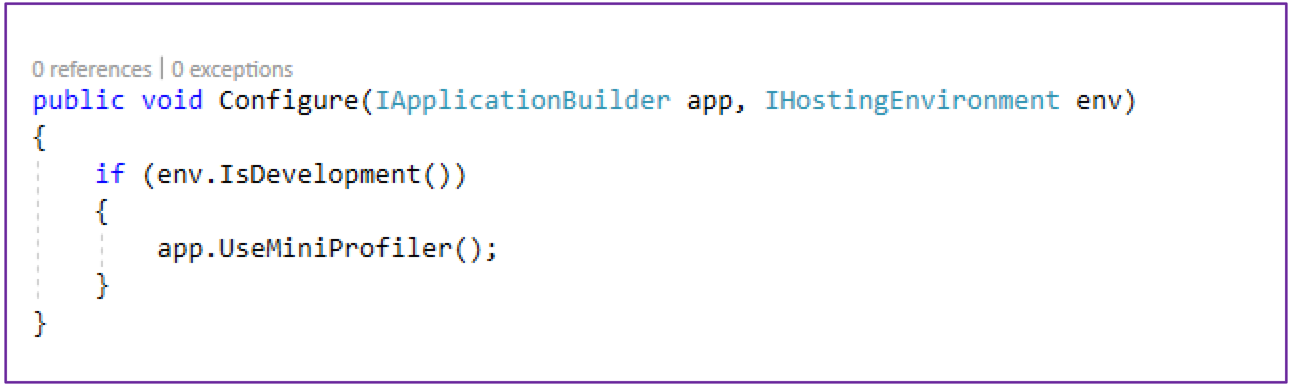


```
0 references | 0 exceptions
public void ConfigureServices(IServiceCollection services)
{
    services.AddMiniProfiler(options =>
    {
        ((MemoryCacheStorage)options.Storage).CacheDuration = TimeSpan.FromMinutes(60);
        options.SqlFormatter = new StackExchange.Profiling.SqlFormatters.InlineFormatter();
    });

    services.AddMiniProfiler().AddEntityFramework();
}
```

EF Core

Mini Profiler EF Core

Startup.cs register miniprofiler on **Configure** function



0 references | 0 exceptions

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseMiniProfiler();
    }
}
```

EF Core

Mini Profiler EF Core

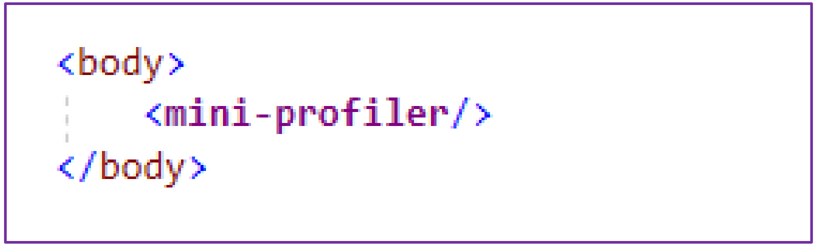
_ViewImports.cshtml add TagHelper MiniProfiler

```
@addTagHelper *, MiniProfiler.AspNetCore.Mvc
```

EF Core

Mini Profiler EF Core

_Layout.cshtml add tag html miniprofiler



```
<body>
    <mini-profiler/>
</body>
```

EF Core

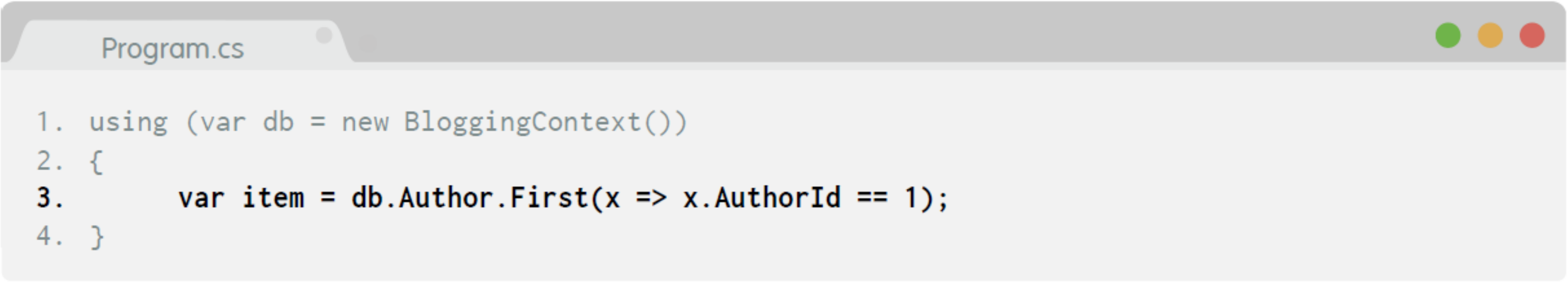
Basic Queries

- LINQ And Function
- Raw SQL Queries
- Global Query Filters
- ThenInclude

EF Core

LINQ And Function

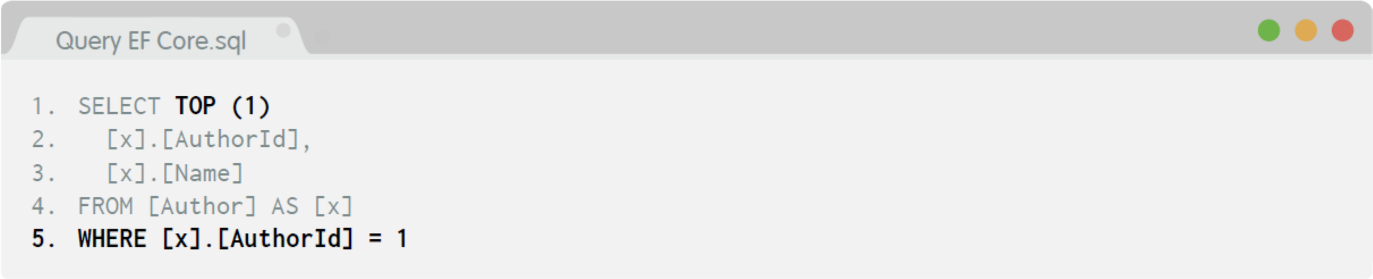
First



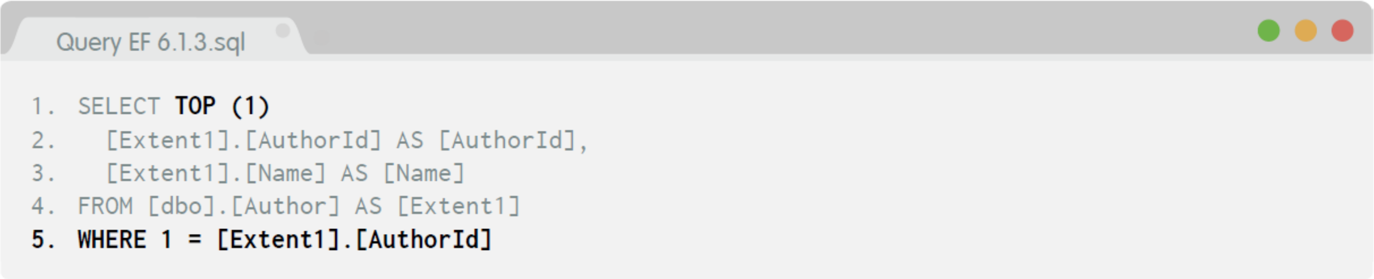
```
using (var db = new BloggingContext())
{
      var item = db.Author.First(x => x.AuthorId == 1);
}
```

EF Core

LINQ And Function



```sql
SELECT TOP (1)
  [x].[AuthorId],
  [x].[Name]
FROM [Author] AS [x]
WHERE [x].[AuthorId] = 1
```

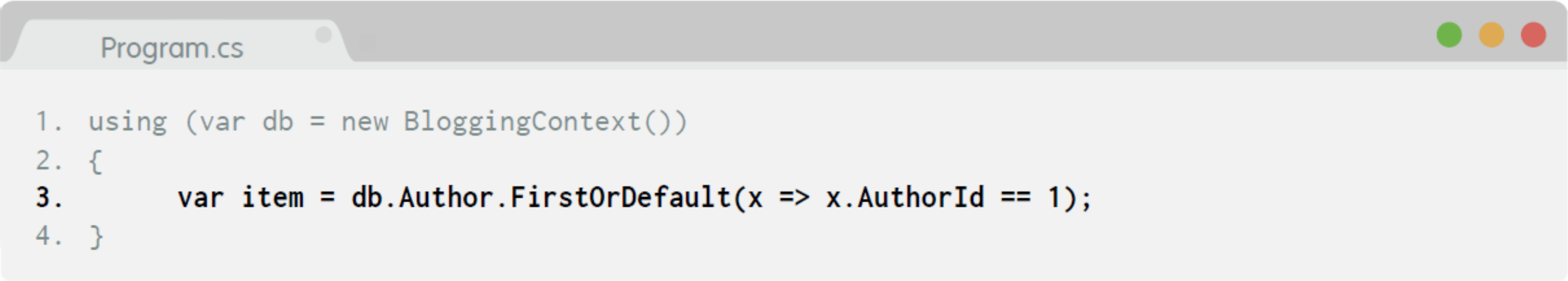


```sql
SELECT TOP (1)
  [Extent1].[AuthorId] AS [AuthorId],
  [Extent1].[Name] AS [Name]
FROM [dbo].[Author] AS [Extent1]
WHERE 1 = [Extent1].[AuthorId]
```

EF Core

LINQ And Function

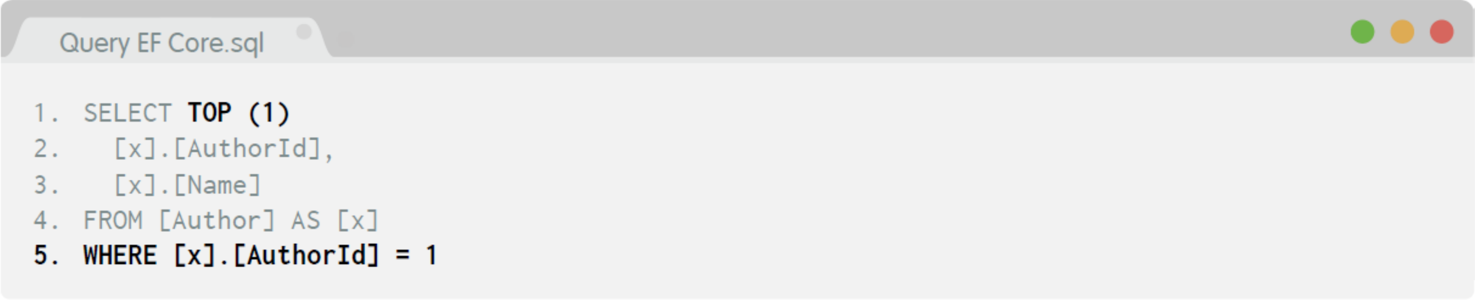
FirstOrDefault



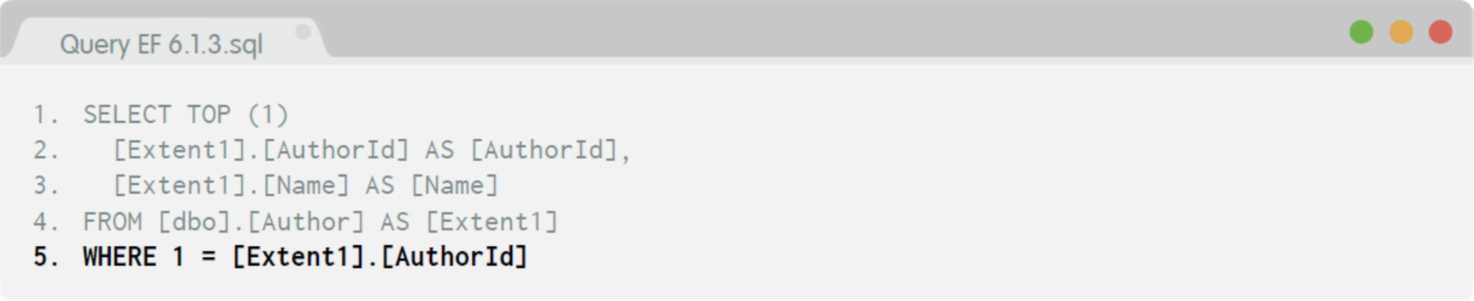
```
using (var db = new BloggingContext())
{
       var item = db.Author.FirstOrDefault(x => x.AuthorId == 1);
}
```

EF
Core

LINQ And Function



```sql
SELECT TOP (1)
  [x].[AuthorId],
  [x].[Name]
FROM [Author] AS [x]
WHERE [x].[AuthorId] = 1
```

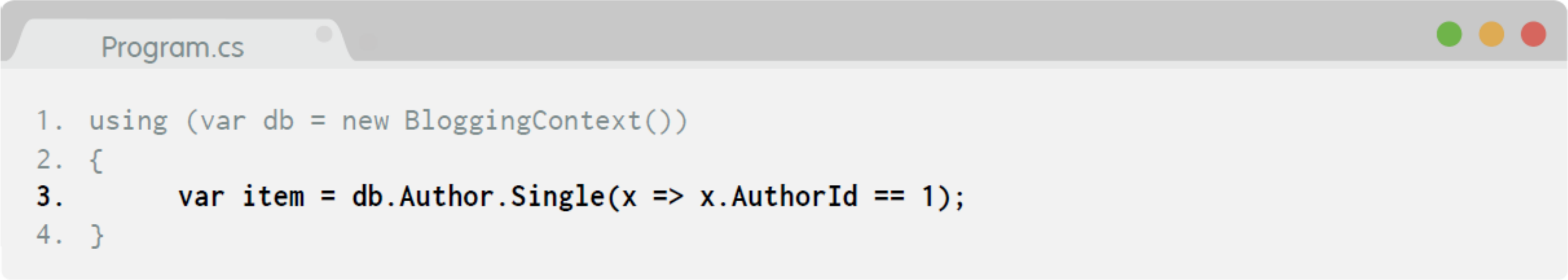


```sql
SELECT TOP (1)
  [Extent1].[AuthorId] AS [AuthorId],
  [Extent1].[Name] AS [Name]
FROM [dbo].[Author] AS [Extent1]
WHERE 1 = [Extent1].[AuthorId]
```

EF Core

LINQ And Function

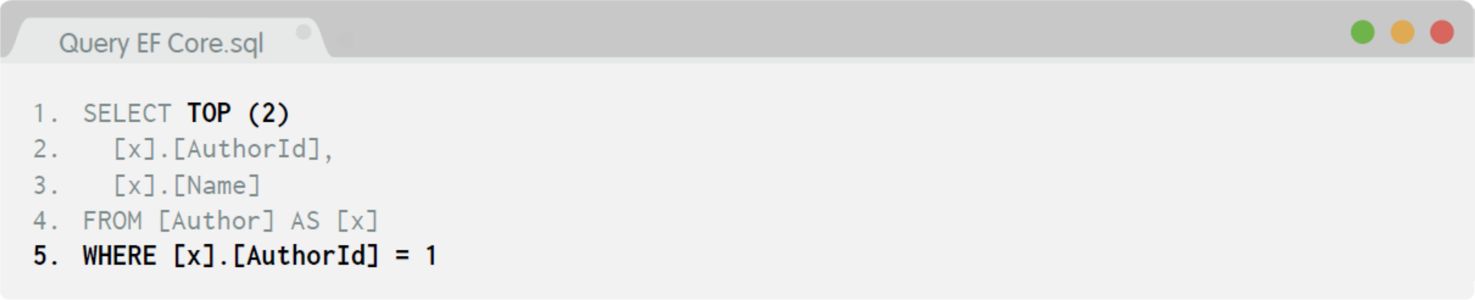
Single



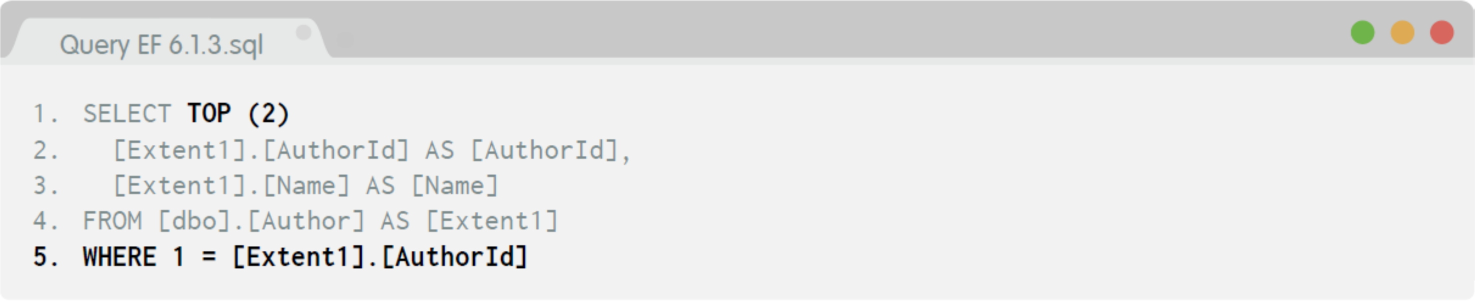
```
using (var db = new BloggingContext())
{
        var item = db.Author.Single(x => x.AuthorId == 1);
}
```

EF Core

LINQ And Function



```sql
SELECT TOP (2)
  [x].[AuthorId],
  [x].[Name]
FROM [Author] AS [x]
WHERE [x].[AuthorId] = 1
```

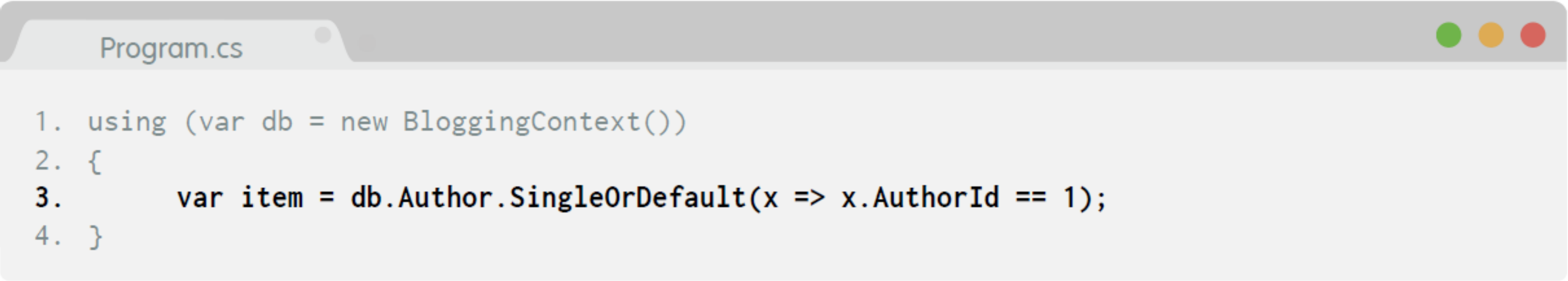


```sql
SELECT TOP (2)
  [Extent1].[AuthorId] AS [AuthorId],
  [Extent1].[Name] AS [Name]
FROM [dbo].[Author] AS [Extent1]
WHERE 1 = [Extent1].[AuthorId]
```

EF Core

LINQ And Function

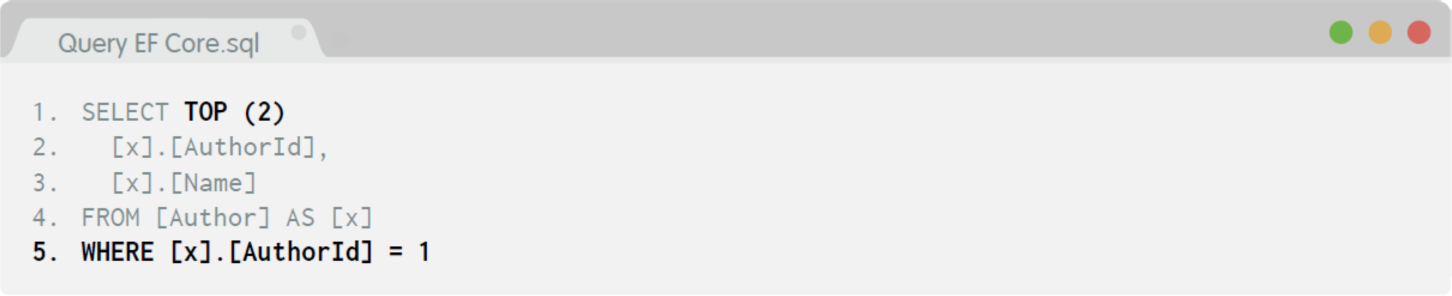
SingleOrDefault



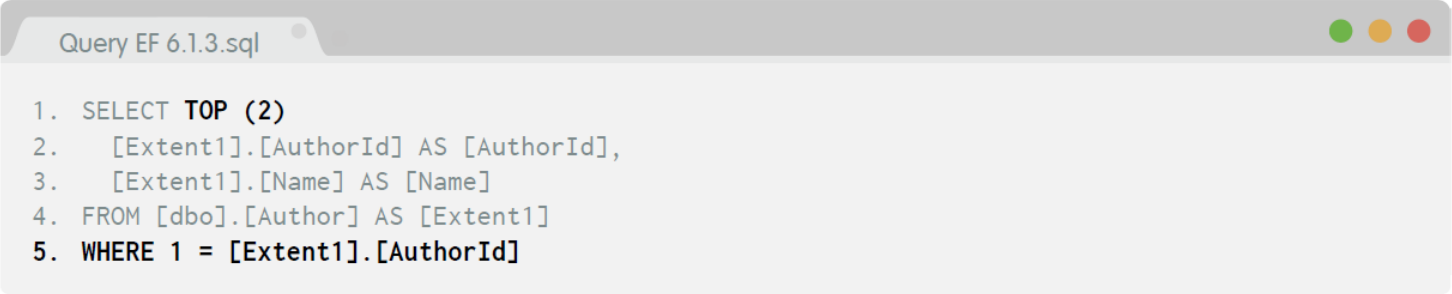
```
using (var db = new BloggingContext())
{
        var item = db.Author.SingleOrDefault(x => x.AuthorId == 1);
}
```

EF Core

LINQ And Function



```sql
SELECT TOP (2)
  [x].[AuthorId],
  [x].[Name]
FROM [Author] AS [x]
WHERE [x].[AuthorId] = 1
```

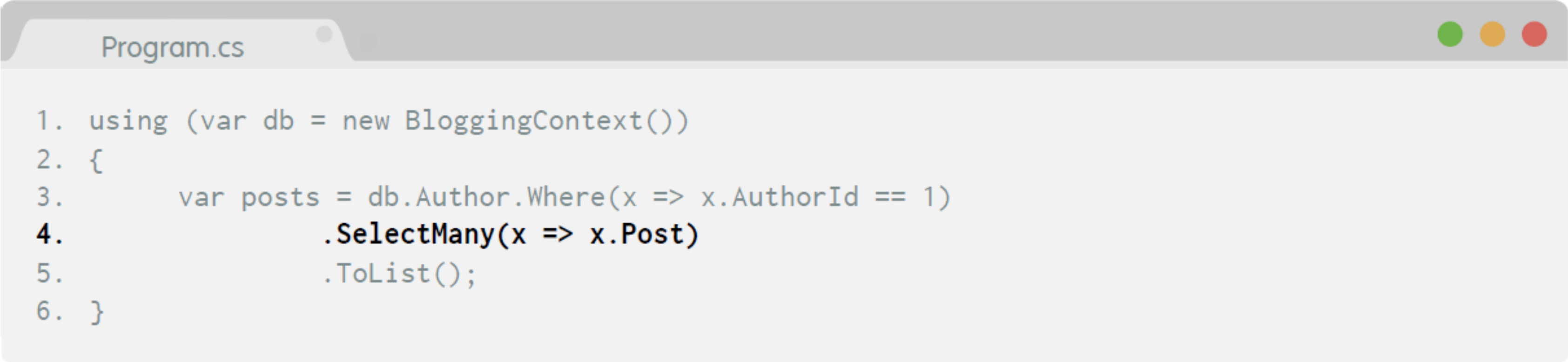


```sql
SELECT TOP (2)
  [Extent1].[AuthorId] AS [AuthorId],
  [Extent1].[Name] AS [Name]
FROM [dbo].[Author] AS [Extent1]
WHERE 1 = [Extent1].[AuthorId]
```

EF Core

LINQ And Function

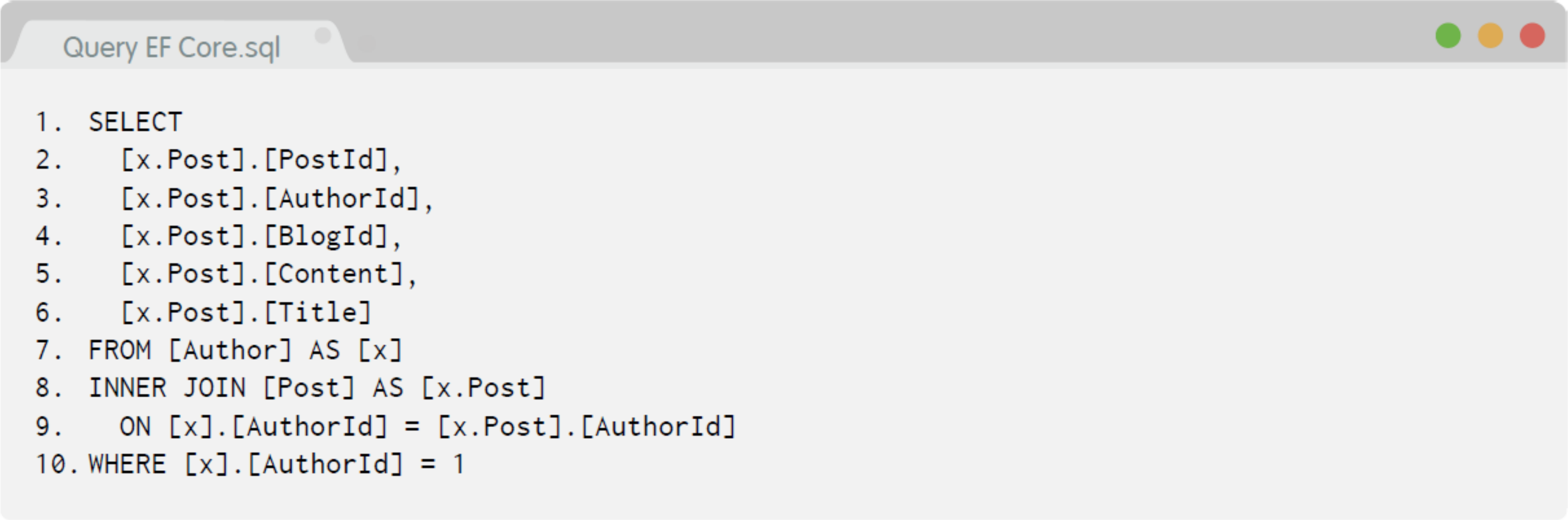
SelectMany



```
using (var db = new BloggingContext())
{
      var posts = db.Author.Where(x => x.AuthorId == 1)
               .SelectMany(x => x.Post)
               .ToList();
}
```

EF Core

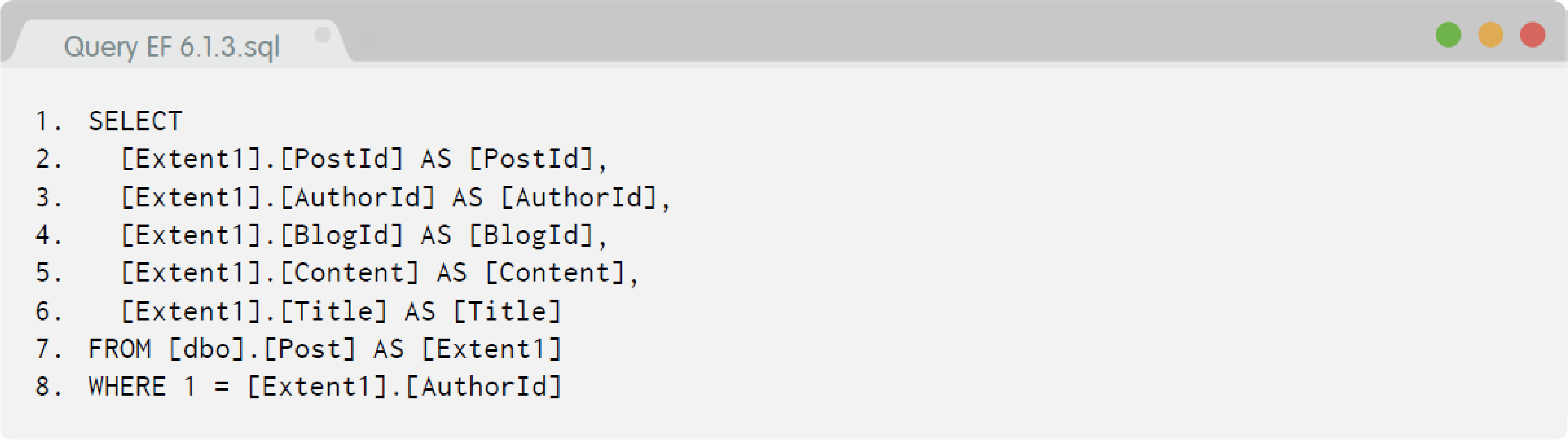
LINQ And Function



```sql
SELECT
  [x.Post].[PostId],
  [x.Post].[AuthorId],
  [x.Post].[BlogId],
  [x.Post].[Content],
  [x.Post].[Title]
FROM [Author] AS [x]
INNER JOIN [Post] AS [x.Post]
  ON [x].[AuthorId] = [x.Post].[AuthorId]
WHERE [x].[AuthorId] = 1
```

EF Core

LINQ And Function

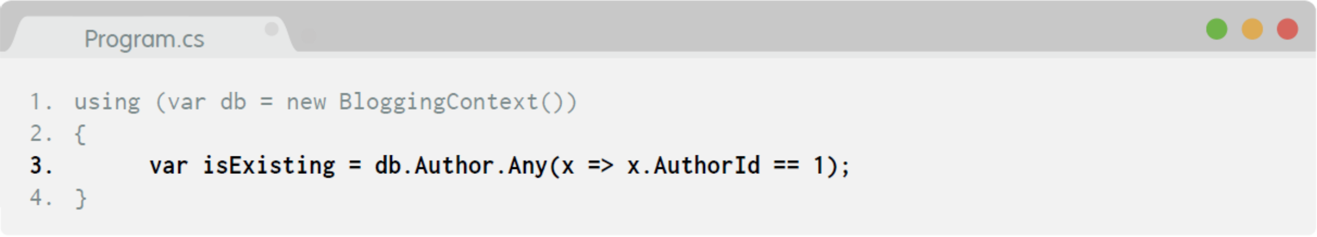


```sql
SELECT
  [Extent1].[PostId] AS [PostId],
  [Extent1].[AuthorId] AS [AuthorId],
  [Extent1].[BlogId] AS [BlogId],
  [Extent1].[Content] AS [Content],
  [Extent1].[Title] AS [Title]
FROM [dbo].[Post] AS [Extent1]
WHERE 1 = [Extent1].[AuthorId]
```

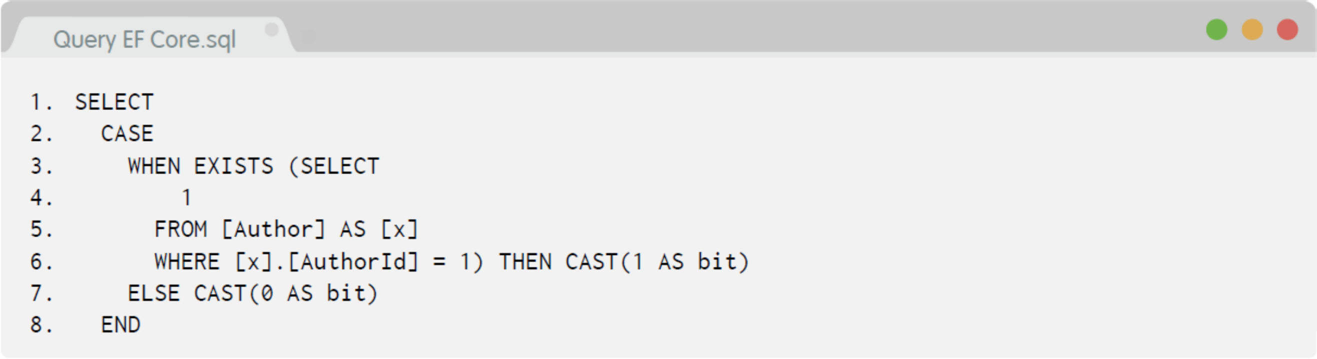
EF Core

LINQ And Function

Any



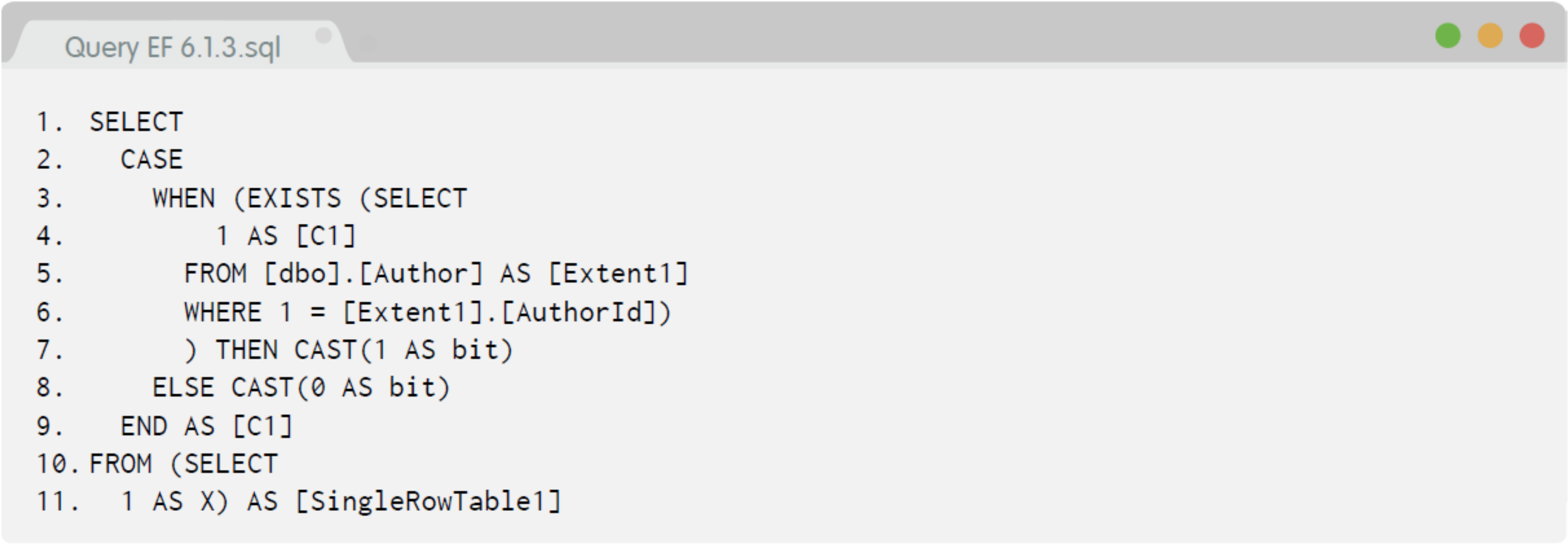
```csharp
using (var db = new BloggingContext())
{
    var isExisting = db.Author.Any(x => x.AuthorId == 1);
}
```



```sql
SELECT
  CASE
    WHEN EXISTS (SELECT
        1
      FROM [Author] AS [x]
      WHERE [x].[AuthorId] = 1) THEN CAST(1 AS bit)
    ELSE CAST(0 AS bit)
  END
```

EF Core

LINQ And Function

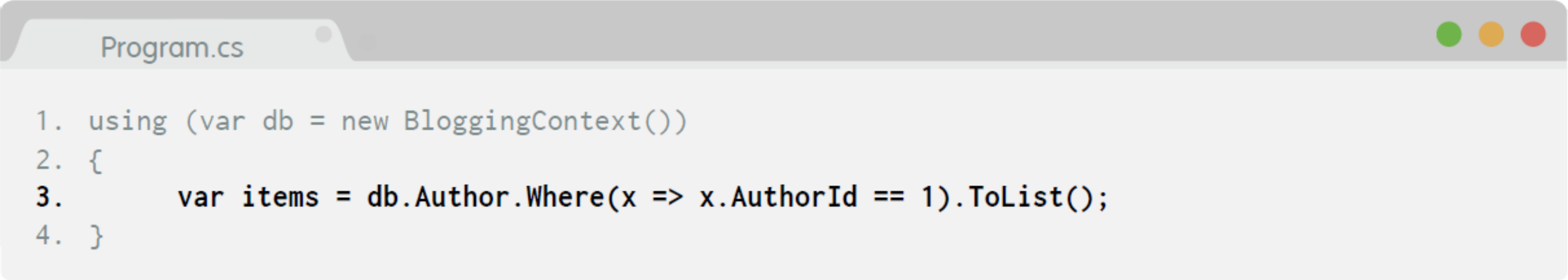


```sql
SELECT
 CASE
  WHEN (EXISTS (SELECT
    1 AS [C1]
   FROM [dbo].[Author] AS [Extent1]
   WHERE 1 = [Extent1].[AuthorId])
   ) THEN CAST(1 AS bit)
  ELSE CAST(0 AS bit)
 END AS [C1]
FROM (SELECT
 1 AS X) AS [SingleRowTable1]
```

EF Core

LINQ And Function

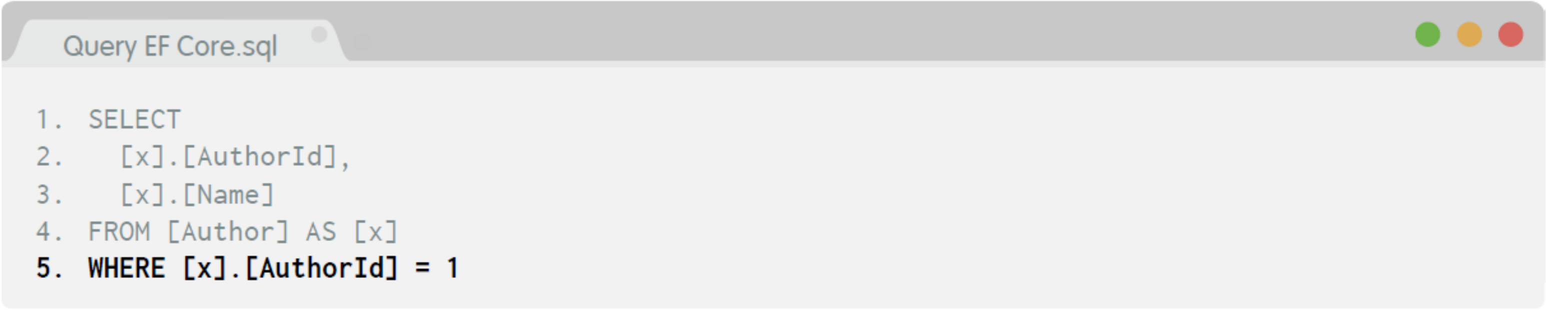
Where



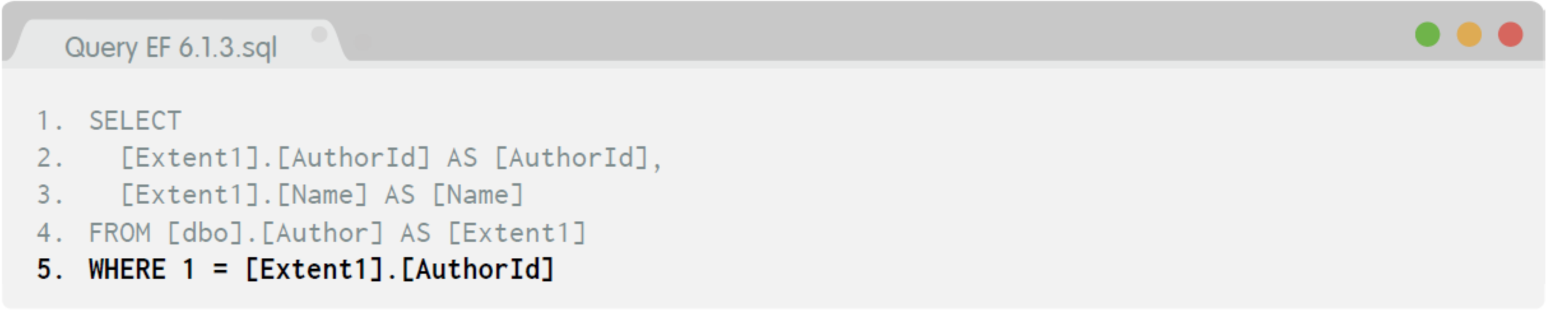
```
using (var db = new BloggingContext())
{
        var items = db.Author.Where(x => x.AuthorId == 1).ToList();
}
```

EF Core

LINQ And Function



```sql
SELECT
  [x].[AuthorId],
  [x].[Name]
FROM [Author] AS [x]
WHERE [x].[AuthorId] = 1
```

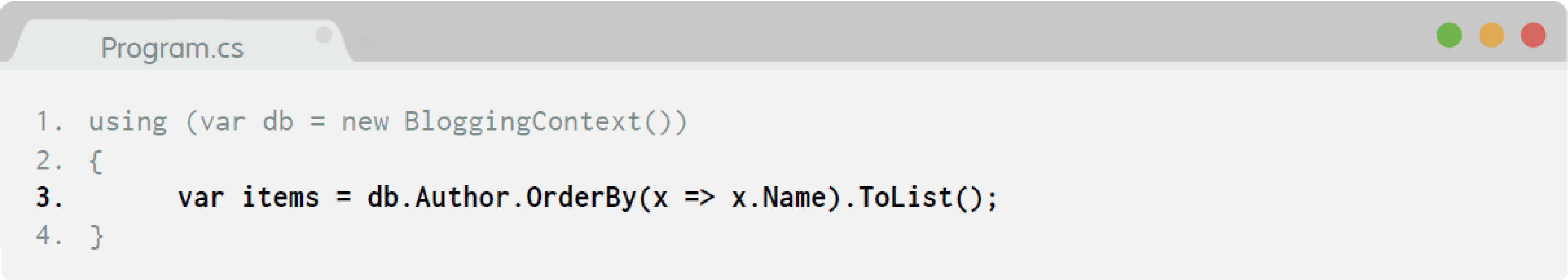


```sql
SELECT
  [Extent1].[AuthorId] AS [AuthorId],
  [Extent1].[Name] AS [Name]
FROM [dbo].[Author] AS [Extent1]
WHERE 1 = [Extent1].[AuthorId]
```

EF Core

LINQ And Function

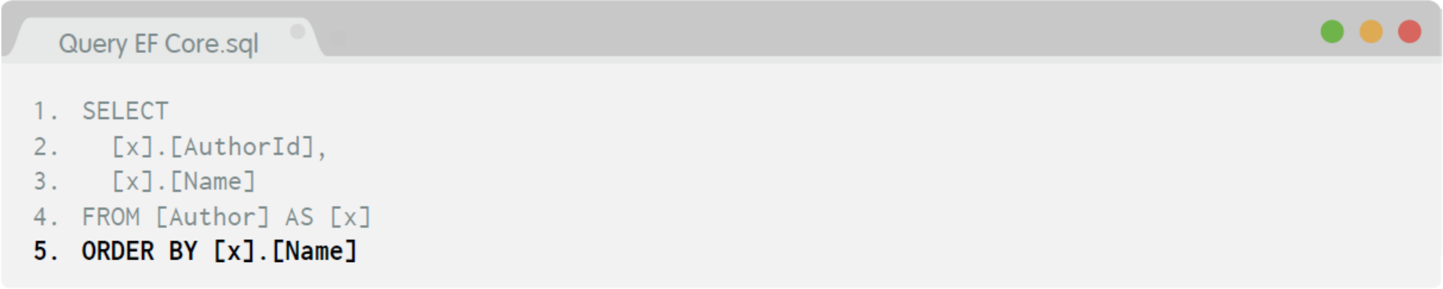
OrderBy



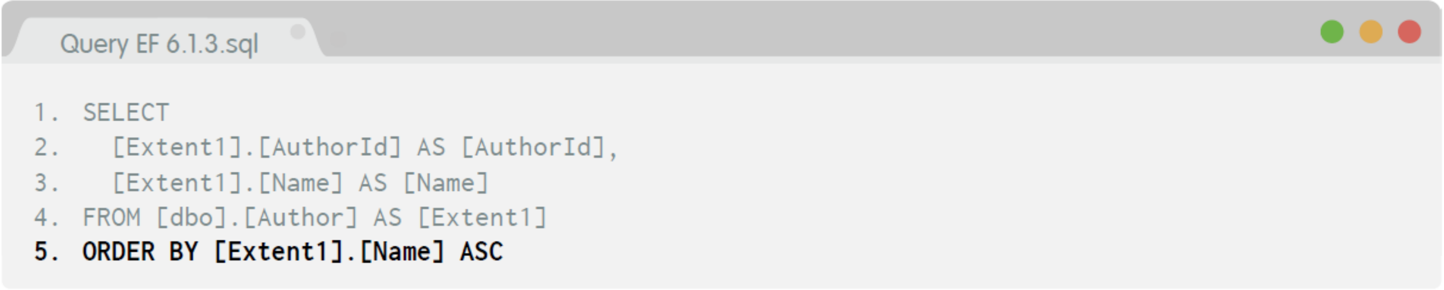
```
using (var db = new BloggingContext())
{
        var items = db.Author.OrderBy(x => x.Name).ToList();
}
```

EF Core

LINQ And Function



```sql
SELECT
  [x].[AuthorId],
  [x].[Name]
FROM [Author] AS [x]
ORDER BY [x].[Name]
```

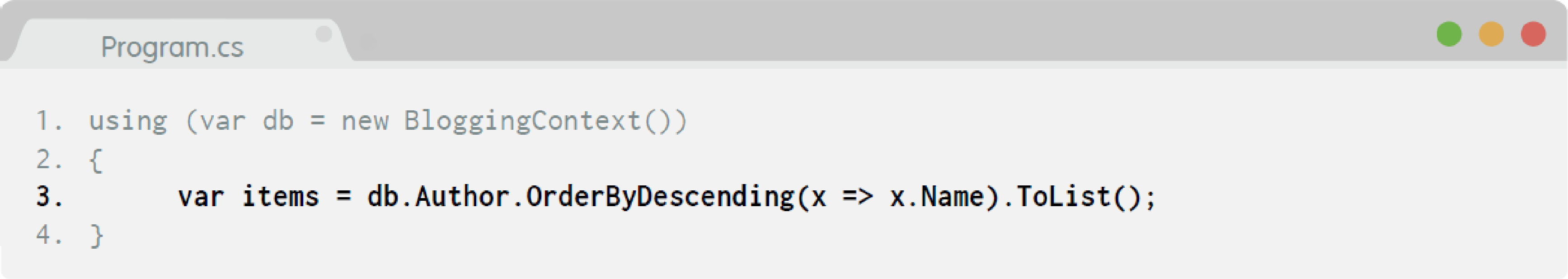


```sql
SELECT
  [Extent1].[AuthorId] AS [AuthorId],
  [Extent1].[Name] AS [Name]
FROM [dbo].[Author] AS [Extent1]
ORDER BY [Extent1].[Name] ASC
```

EF Core

LINQ And Function

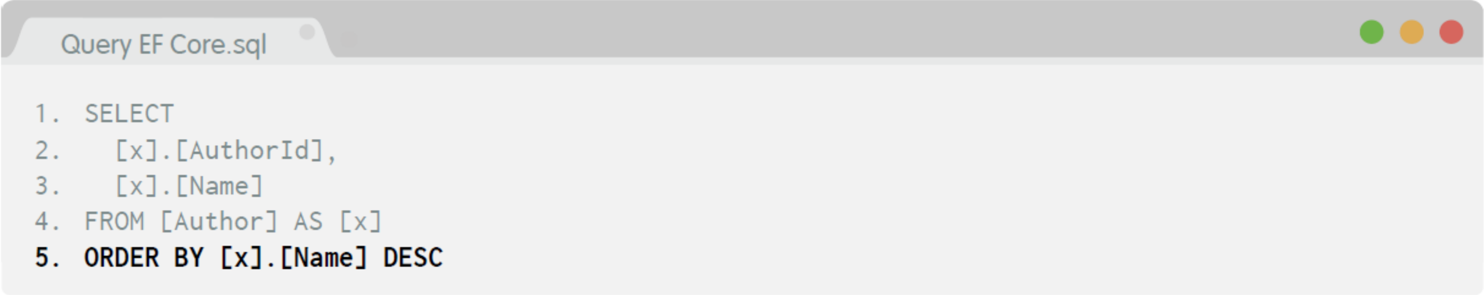
OrderByDescending



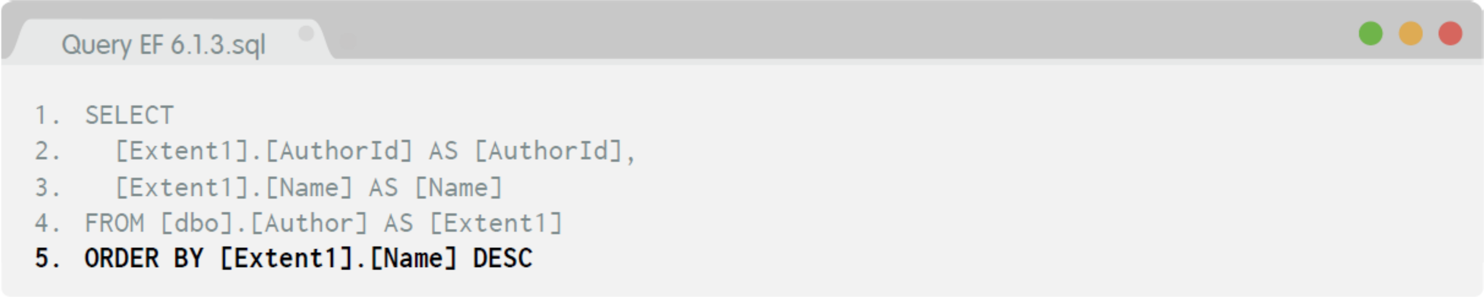
```
using (var db = new BloggingContext())
{
        var items = db.Author.OrderByDescending(x => x.Name).ToList();
}
```

EF Core

LINQ And Function



```sql
SELECT
  [x].[AuthorId],
  [x].[Name]
FROM [Author] AS [x]
ORDER BY [x].[Name] DESC
```

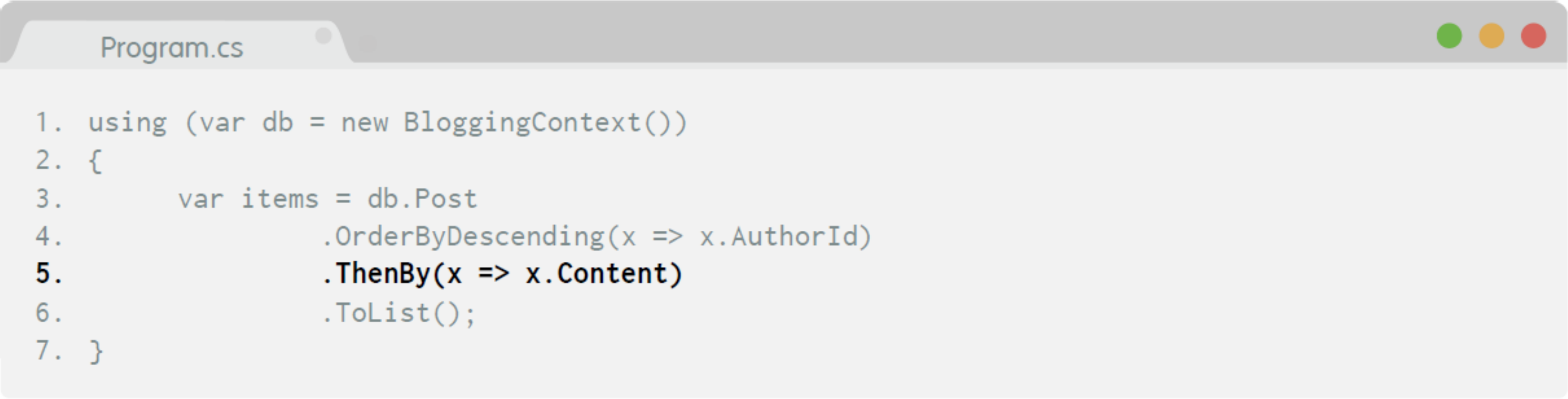


```sql
SELECT
  [Extent1].[AuthorId] AS [AuthorId],
  [Extent1].[Name] AS [Name]
FROM [dbo].[Author] AS [Extent1]
ORDER BY [Extent1].[Name] DESC
```

EF Core

LINQ And Function

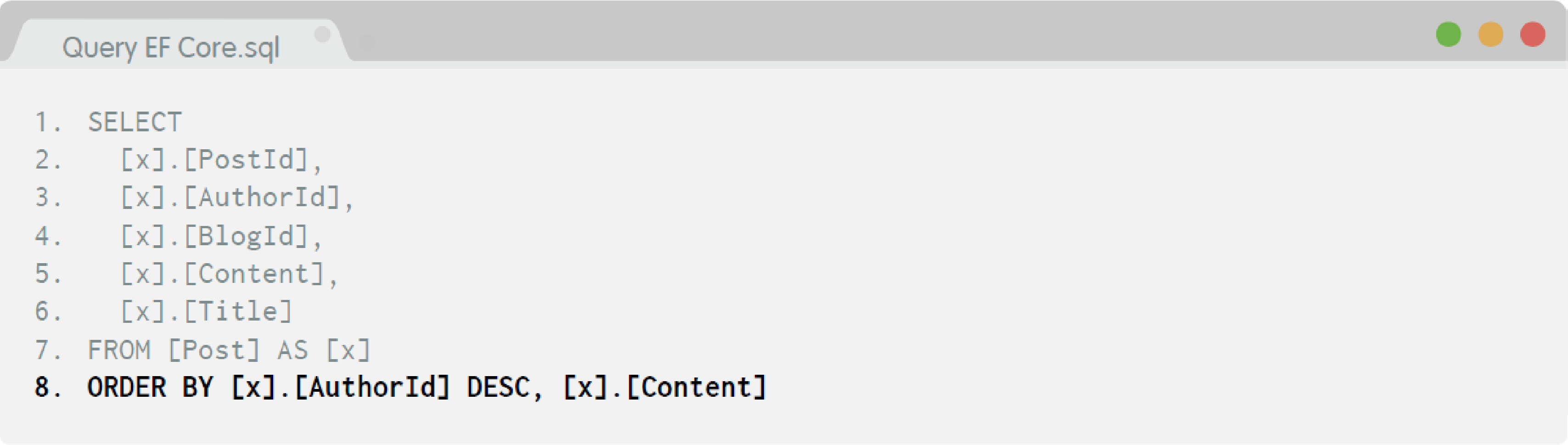
ThenBy



```
using (var db = new BloggingContext())
{
      var items = db.Post
               .OrderByDescending(x => x.AuthorId)
               .ThenBy(x => x.Content)
               .ToList();
}
```

EF
Core

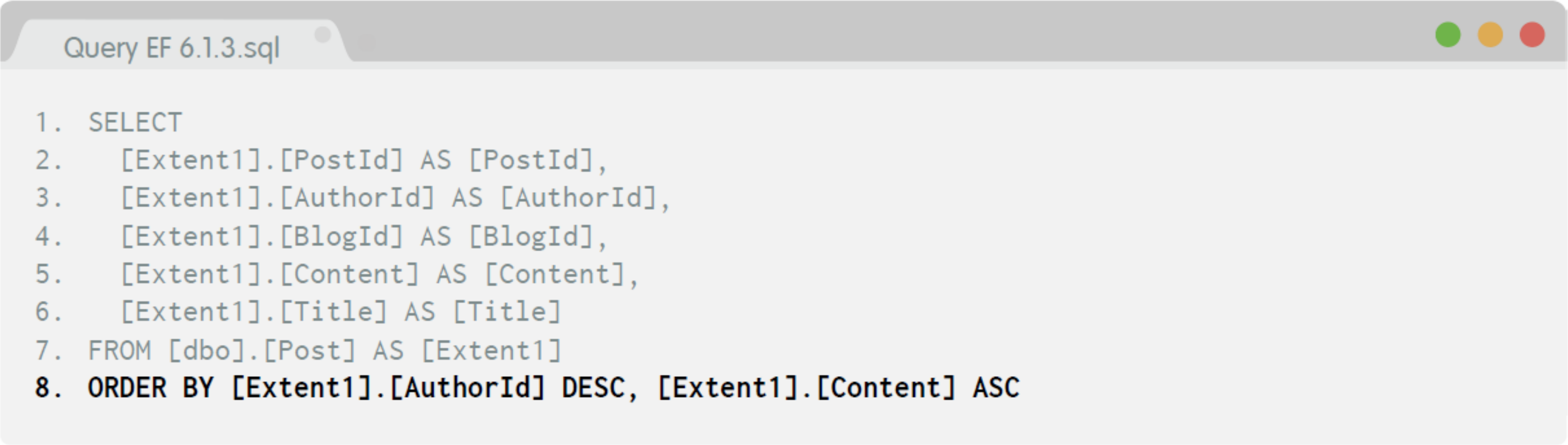
LINQ And Function



```sql
SELECT
  [x].[PostId],
  [x].[AuthorId],
  [x].[BlogId],
  [x].[Content],
  [x].[Title]
FROM [Post] AS [x]
ORDER BY [x].[AuthorId] DESC, [x].[Content]
```

EF Core

LINQ And Function

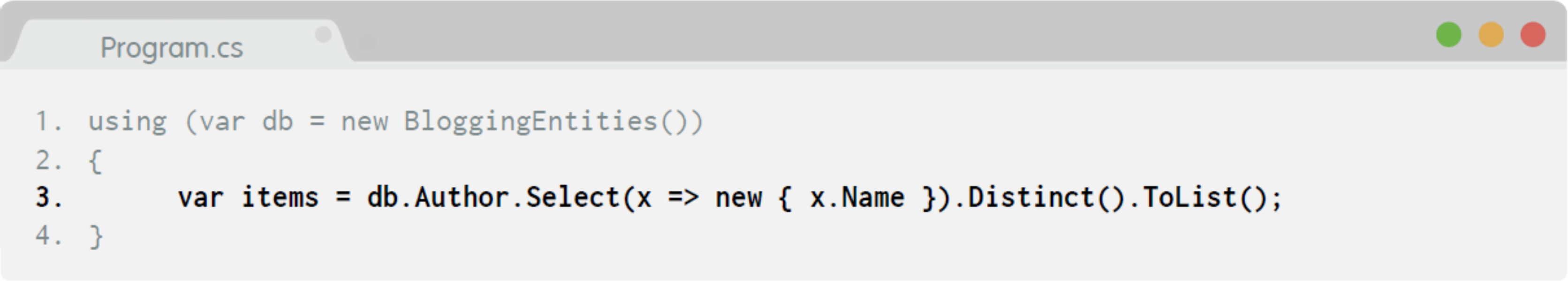


```
SELECT
  [Extent1].[PostId] AS [PostId],
  [Extent1].[AuthorId] AS [AuthorId],
  [Extent1].[BlogId] AS [BlogId],
  [Extent1].[Content] AS [Content],
  [Extent1].[Title] AS [Title]
FROM [dbo].[Post] AS [Extent1]
ORDER BY [Extent1].[AuthorId] DESC, [Extent1].[Content] ASC
```

EF Core

LINQ And Function

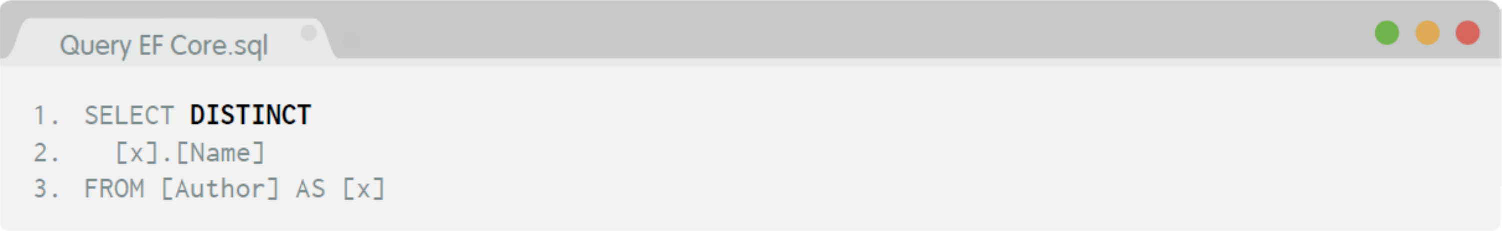
Distinct



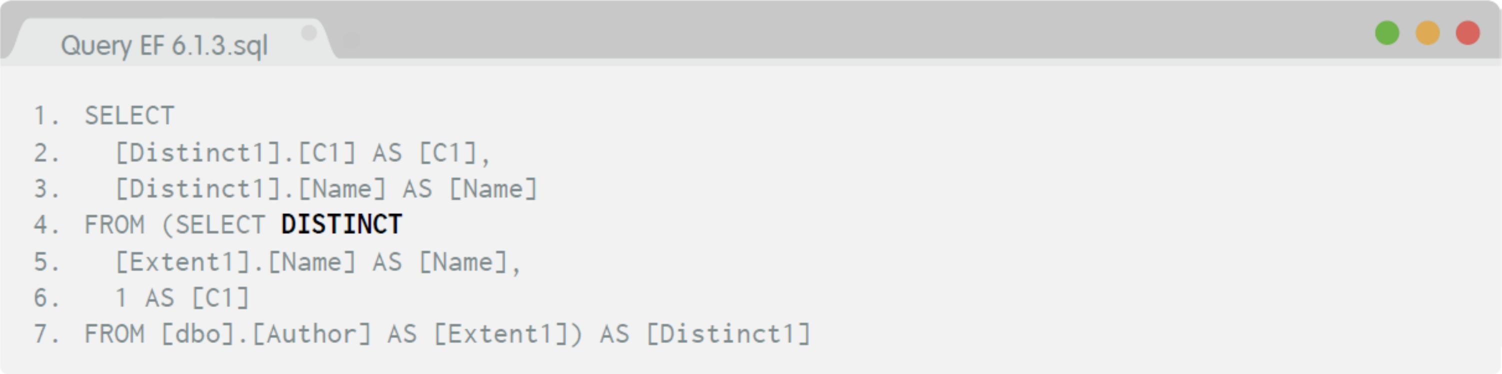
```
using (var db = new BloggingEntities())
{
        var items = db.Author.Select(x => new { x.Name }).Distinct().ToList();
}
```

EF Core

LINQ And Function



```
SELECT DISTINCT
  [x].[Name]
FROM [Author] AS [x]
```

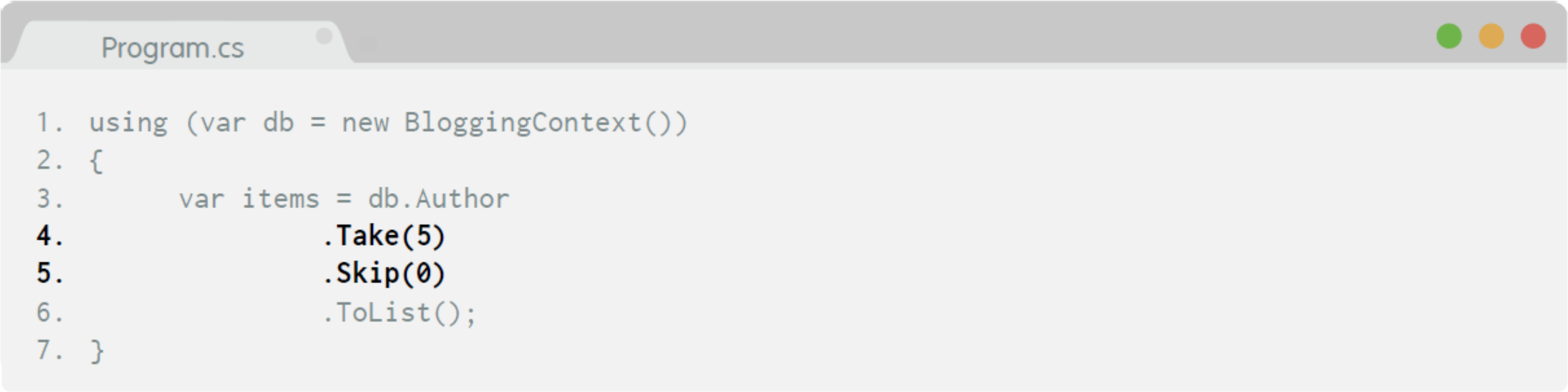


```
SELECT
  [Distinct1].[C1] AS [C1],
  [Distinct1].[Name] AS [Name]
FROM (SELECT DISTINCT
  [Extent1].[Name] AS [Name],
  1 AS [C1]
FROM [dbo].[Author] AS [Extent1]) AS [Distinct1]
```

EF Core

LINQ And Function

Take, Skip

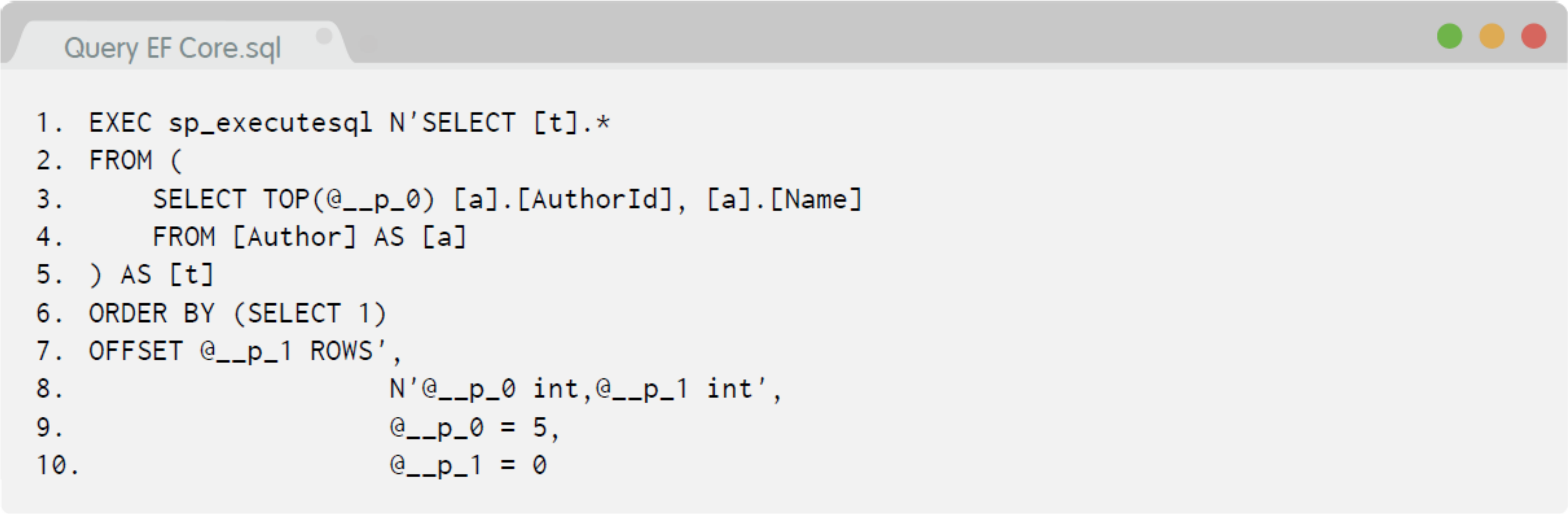


Program.cs

```
using (var db = new BloggingContext())
{
      var items = db.Author
               .Take(5)
               .Skip(0)
               .ToList();
}
```

EF
Core

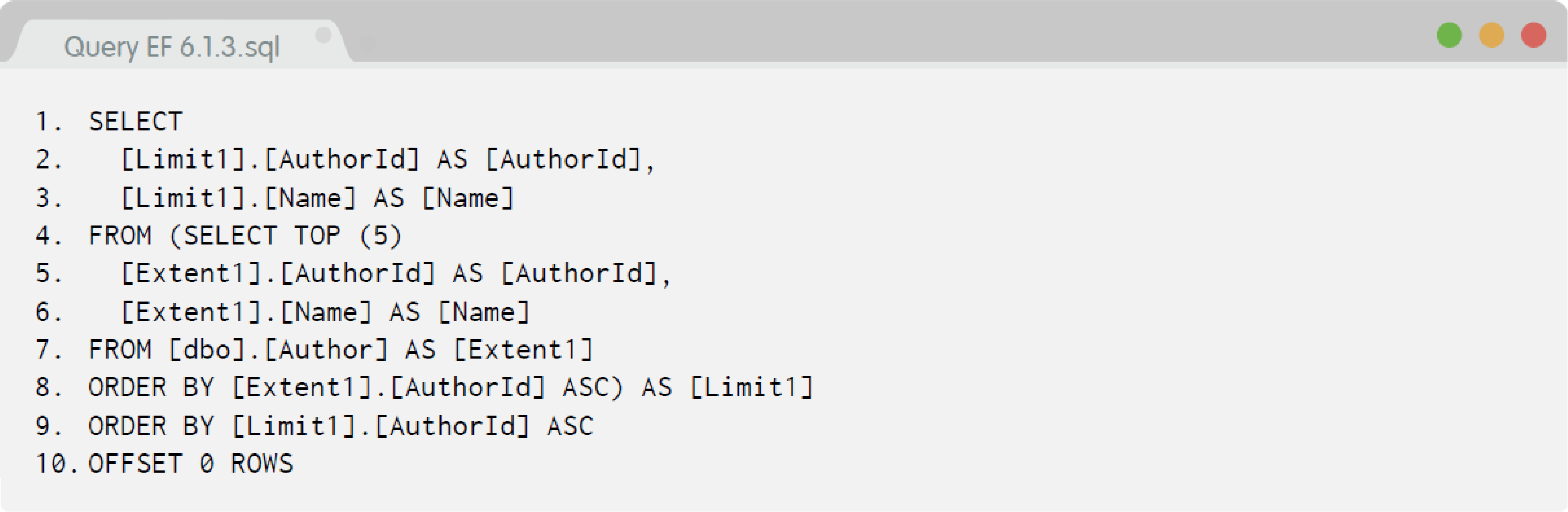
LINQ And Function



```sql
EXEC sp_executesql N'SELECT [t].*
FROM (
    SELECT TOP(@__p_0) [a].[AuthorId], [a].[Name]
    FROM [Author] AS [a]
) AS [t]
ORDER BY (SELECT 1)
OFFSET @__p_1 ROWS',
                   N'@__p_0 int,@__p_1 int',
                   @__p_0 = 5,
                   @__p_1 = 0
```

EF Core

LINQ And Function

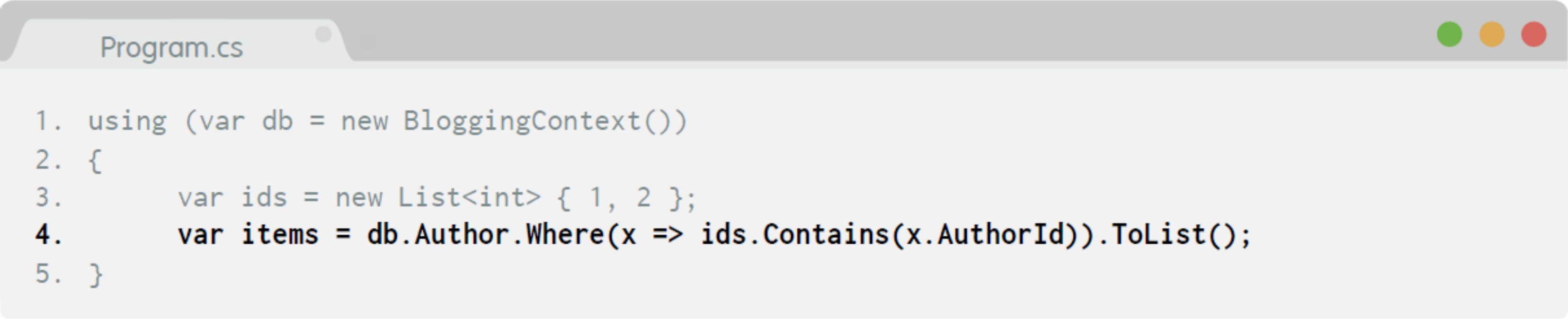


```
SELECT
  [Limit1].[AuthorId] AS [AuthorId],
  [Limit1].[Name] AS [Name]
FROM (SELECT TOP (5)
  [Extent1].[AuthorId] AS [AuthorId],
  [Extent1].[Name] AS [Name]
FROM [dbo].[Author] AS [Extent1]
ORDER BY [Extent1].[AuthorId] ASC) AS [Limit1]
ORDER BY [Limit1].[AuthorId] ASC
OFFSET 0 ROWS
```

EF Core

LINQ And Function

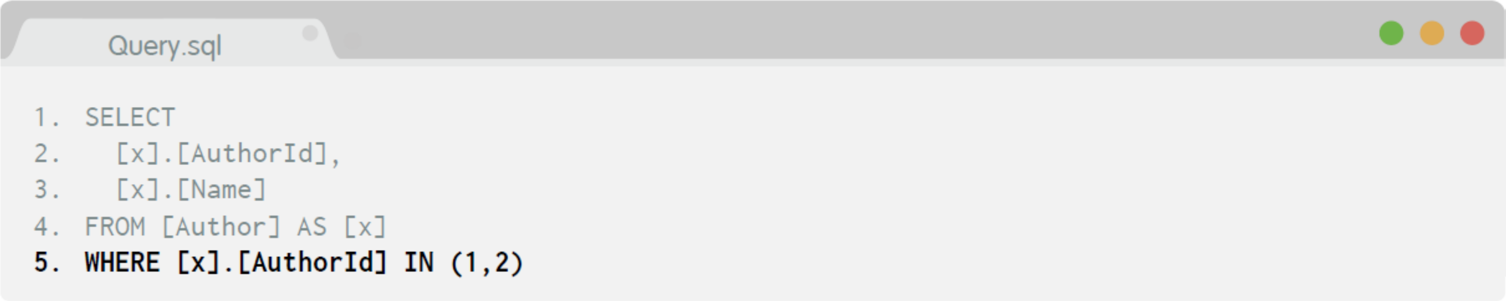
Contains (List)



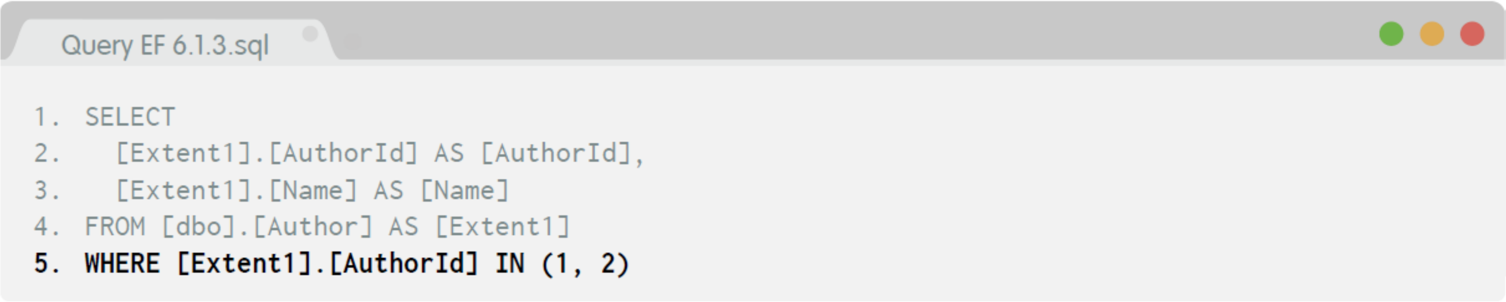
```
using (var db = new BloggingContext())
{
        var ids = new List<int> { 1, 2 };
        var items = db.Author.Where(x => ids.Contains(x.AuthorId)).ToList();
}
```

EF Core

LINQ And Function



```sql
SELECT
  [x].[AuthorId],
  [x].[Name]
FROM [Author] AS [x]
WHERE [x].[AuthorId] IN (1,2)
```

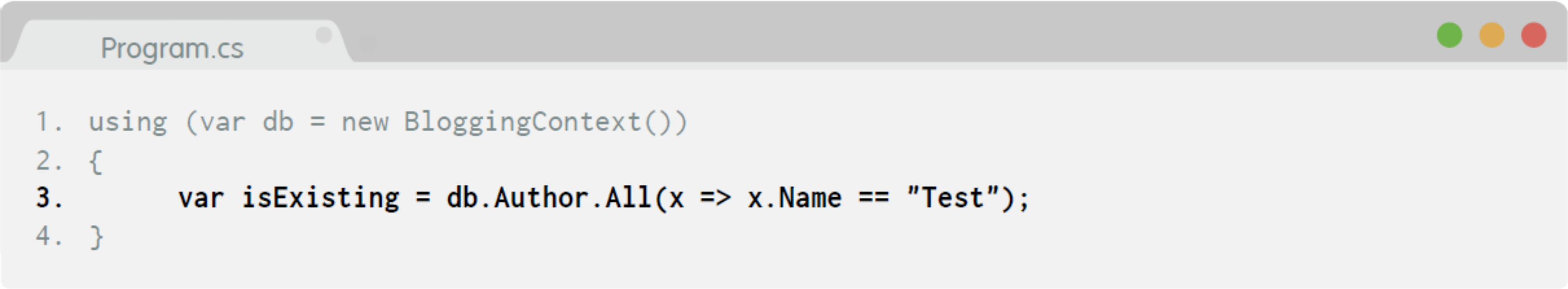


```sql
SELECT
  [Extent1].[AuthorId] AS [AuthorId],
  [Extent1].[Name] AS [Name]
FROM [dbo].[Author] AS [Extent1]
WHERE [Extent1].[AuthorId] IN (1, 2)
```

EF Core

LINQ And Function

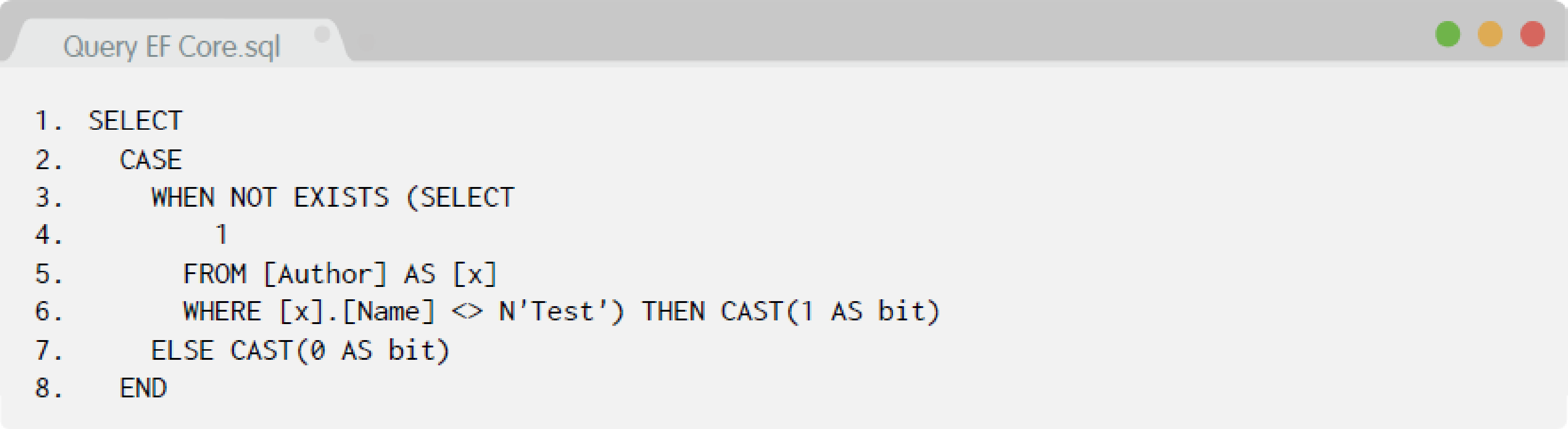
All



```
using (var db = new BloggingContext())
{
        var isExisting = db.Author.All(x => x.Name == "Test");
}
```

EF Core

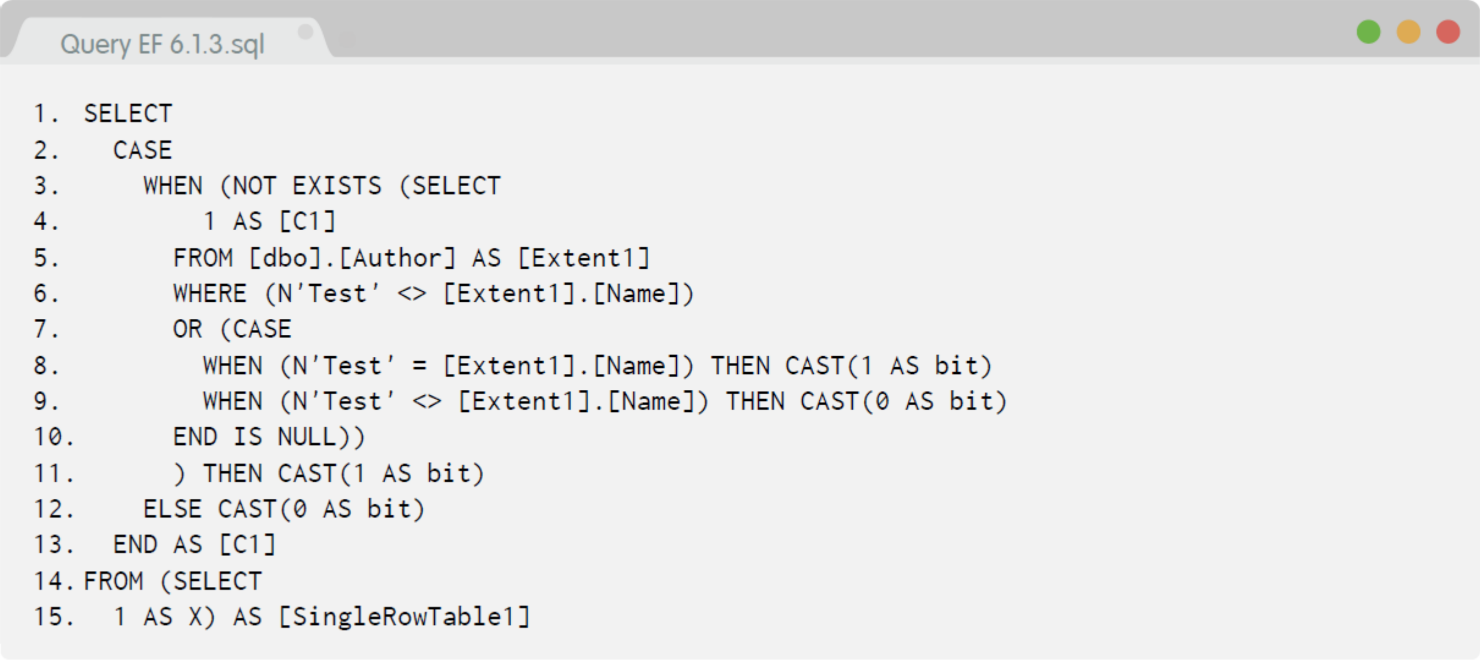
LINQ And Function



```sql
SELECT
  CASE
    WHEN NOT EXISTS (SELECT
        1
      FROM [Author] AS [x]
      WHERE [x].[Name] <> N'Test') THEN CAST(1 AS bit)
    ELSE CAST(0 AS bit)
  END
```

EF
Core

LINQ And Function

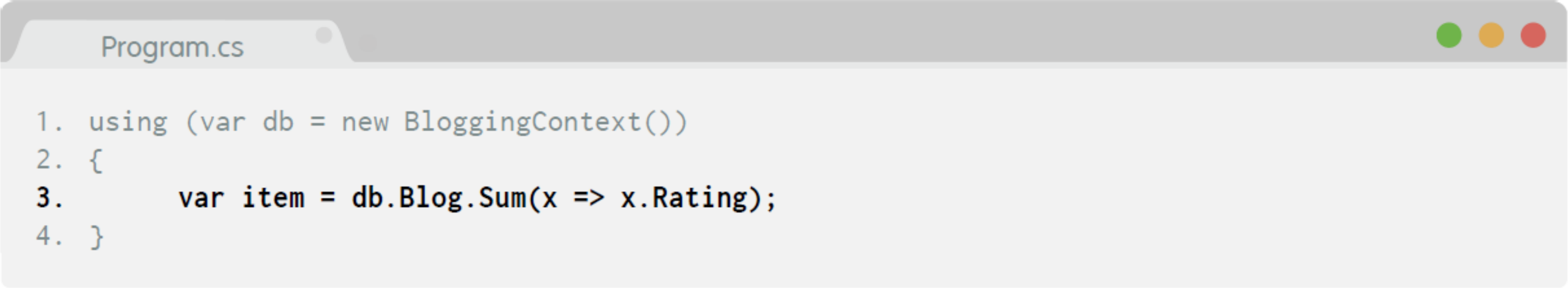


```
SELECT
  CASE
    WHEN (NOT EXISTS (SELECT
        1 AS [C1]
      FROM [dbo].[Author] AS [Extent1]
      WHERE (N’Test’ <> [Extent1].[Name])
      OR (CASE
        WHEN (N’Test’ = [Extent1].[Name]) THEN CAST(1 AS bit)
        WHEN (N’Test’ <> [Extent1].[Name]) THEN CAST(0 AS bit)
      END IS NULL))
      ) THEN CAST(1 AS bit)
    ELSE CAST(0 AS bit)
  END AS [C1]
FROM (SELECT
  1 AS X) AS [SingleRowTable1]
```

EF Core

LINQ And Function

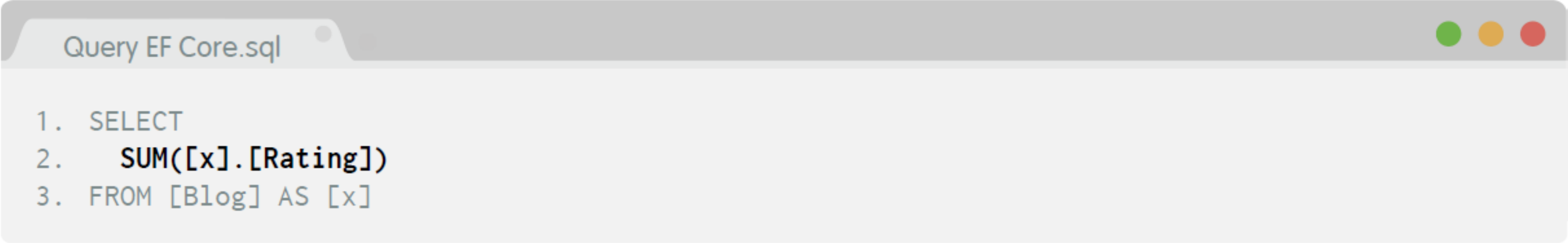
Sum



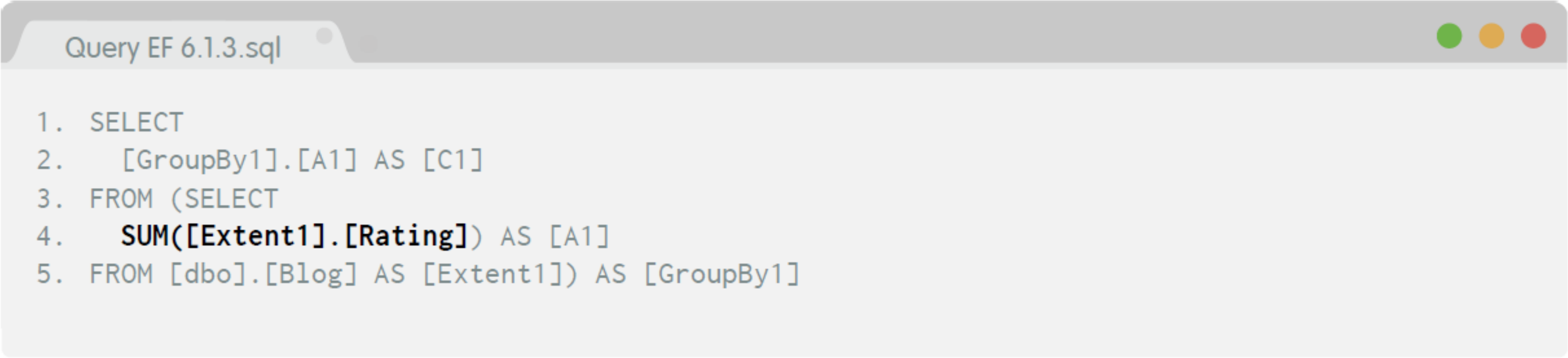
```
using (var db = new BloggingContext())
{
        var item = db.Blog.Sum(x => x.Rating);
}
```

EF Core

LINQ And Function



```sql
SELECT
  SUM([x].[Rating])
FROM [Blog] AS [x]
```

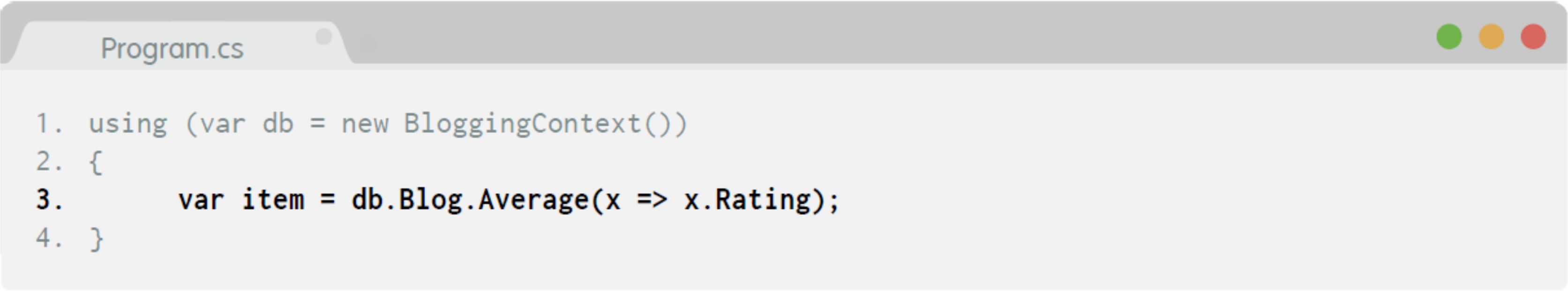


```sql
SELECT
  [GroupBy1].[A1] AS [C1]
FROM (SELECT
  SUM([Extent1].[Rating]) AS [A1]
FROM [dbo].[Blog] AS [Extent1]) AS [GroupBy1]
```

EF Core

LINQ And Function

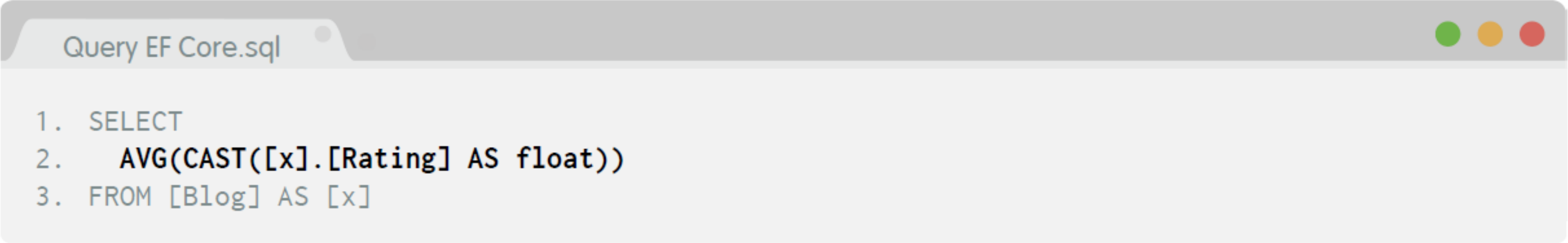
Average


Program.cs

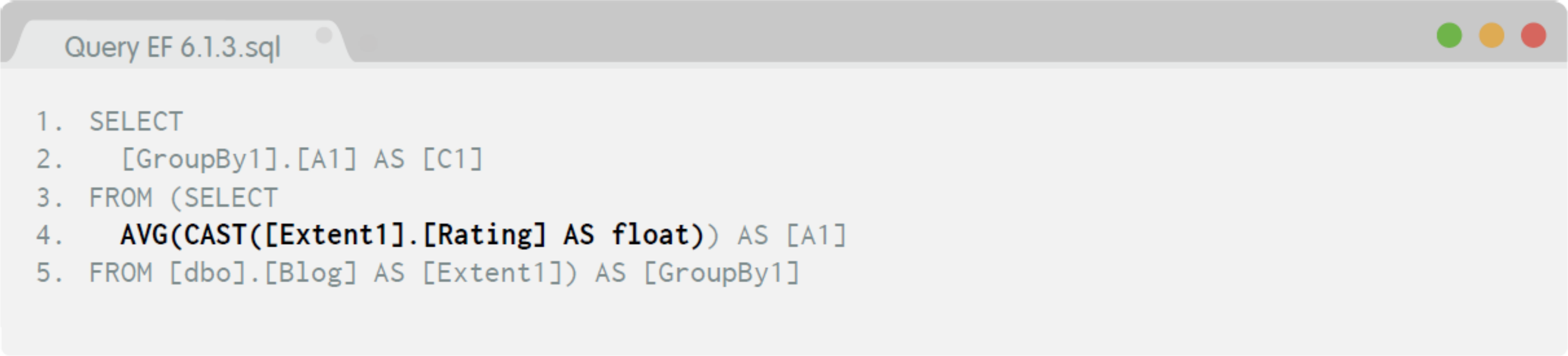
```
using (var db = new BloggingContext())
{
        var item = db.Blog.Average(x => x.Rating);
}
```

EF Core

LINQ And Function



```sql
SELECT
  AVG(CAST([x].[Rating] AS float))
FROM [Blog] AS [x]
```

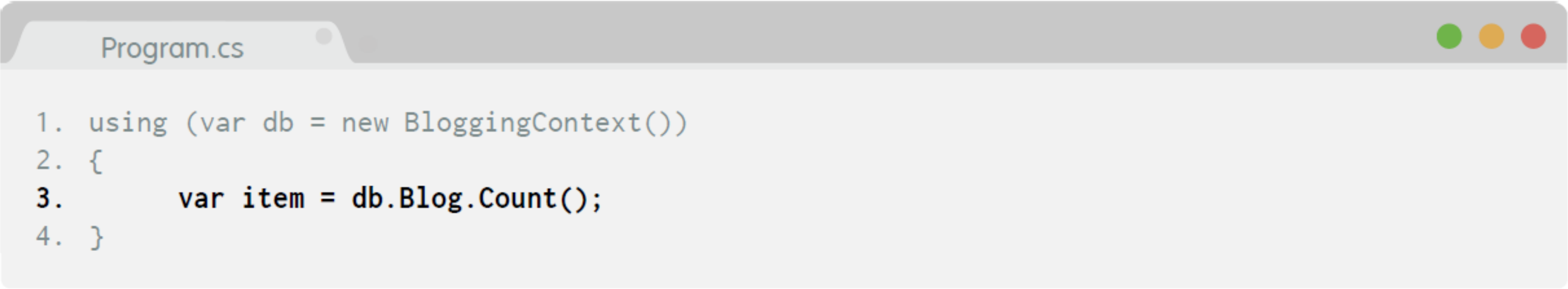


```sql
SELECT
  [GroupBy1].[A1] AS [C1]
FROM (SELECT
  AVG(CAST([Extent1].[Rating] AS float)) AS [A1]
FROM [dbo].[Blog] AS [Extent1]) AS [GroupBy1]
```

EF Core

LINQ And Function

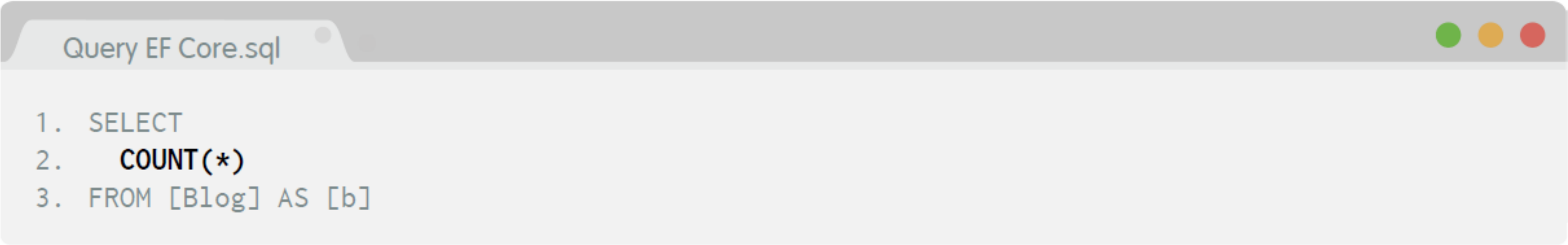
Count



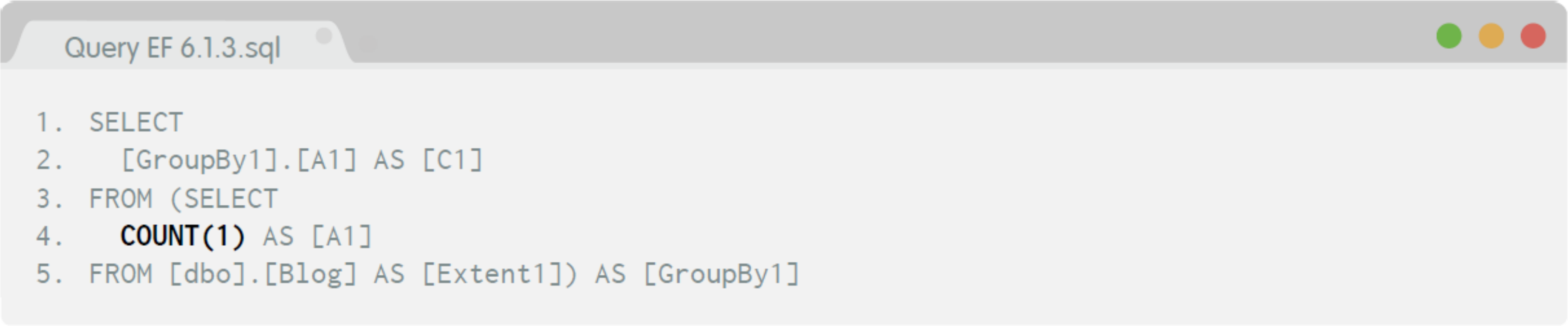
```
using (var db = new BloggingContext())
{
        var item = db.Blog.Count();
}
```

EF Core

LINQ And Function



```sql
SELECT
   COUNT(*)
FROM [Blog] AS [b]
```

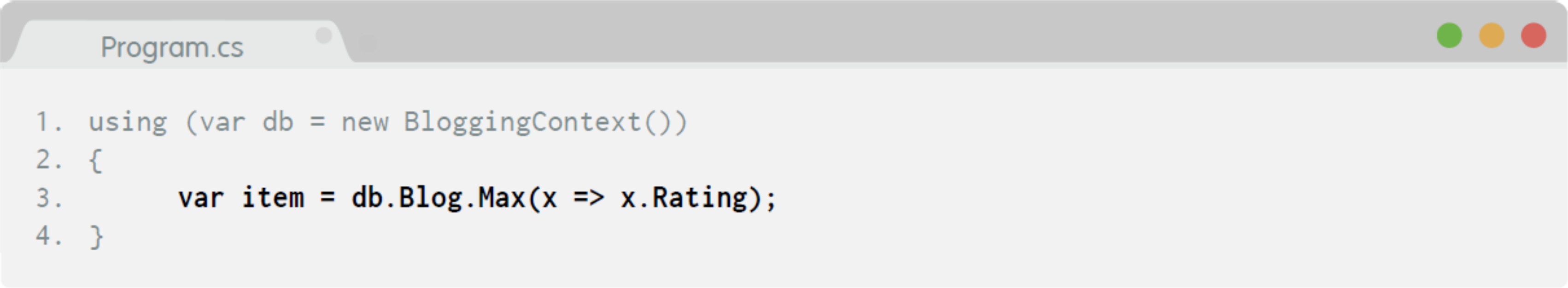


```sql
SELECT
   [GroupBy1].[A1] AS [C1]
FROM (SELECT
   COUNT(1) AS [A1]
FROM [dbo].[Blog] AS [Extent1]) AS [GroupBy1]
```

EF Core

LINQ And Function

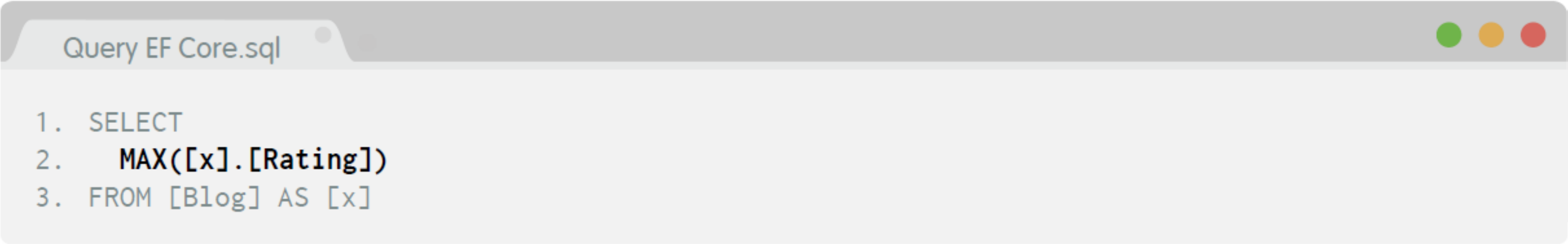
Max



```
using (var db = new BloggingContext())
{
        var item = db.Blog.Max(x => x.Rating);
}
```

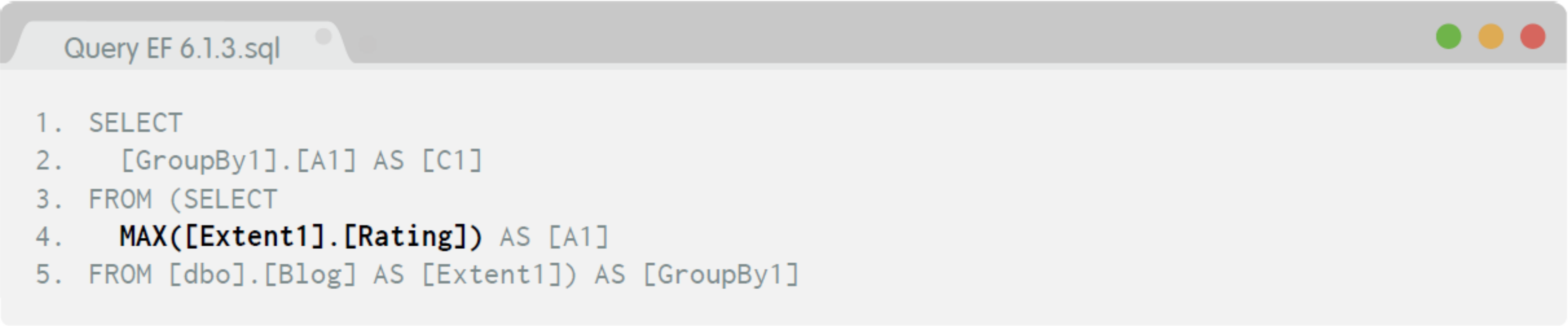
EF Core

LINQ And Function



Query EF Core.sql

```sql
SELECT
  MAX([x].[Rating])
FROM [Blog] AS [x]
```



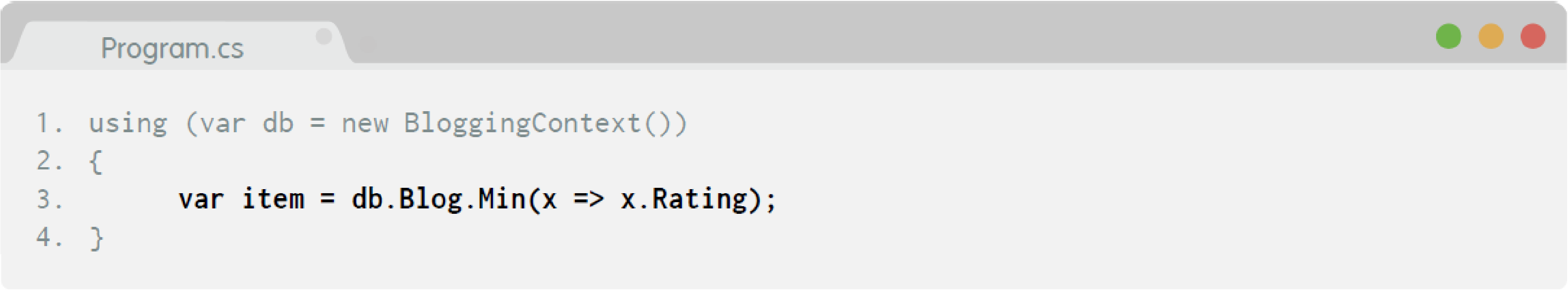
Query EF 6.1.3.sql

```sql
SELECT
  [GroupBy1].[A1] AS [C1]
FROM (SELECT
  MAX([Extent1].[Rating]) AS [A1]
FROM [dbo].[Blog] AS [Extent1]) AS [GroupBy1]
```

EF Core

LINQ And Function

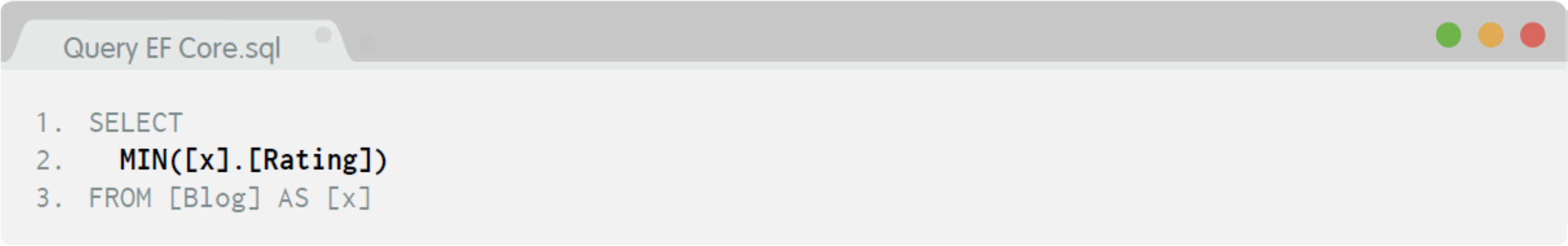
Min



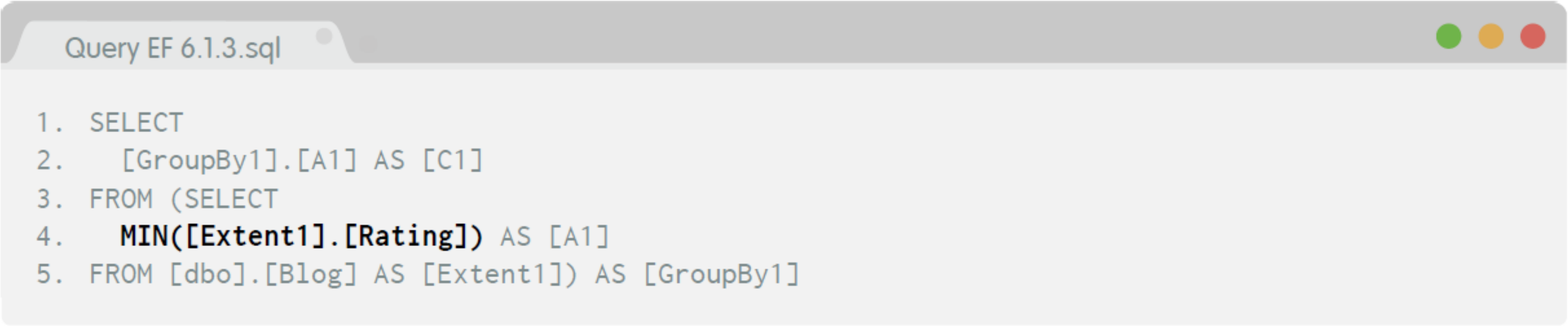
```
using (var db = new BloggingContext())
{
        var item = db.Blog.Min(x => x.Rating);
}
```

EF Core

LINQ And Function



```
SELECT
  MIN([x].[Rating])
FROM [Blog] AS [x]
```

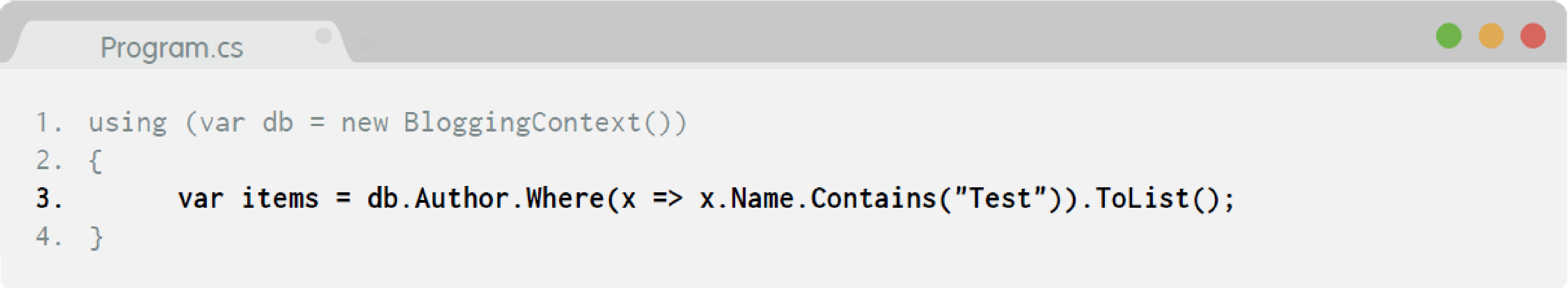


```
SELECT
  [GroupBy1].[A1] AS [C1]
FROM (SELECT
  MIN([Extent1].[Rating]) AS [A1]
FROM [dbo].[Blog] AS [Extent1]) AS [GroupBy1]
```

EF Core

LINQ And Function

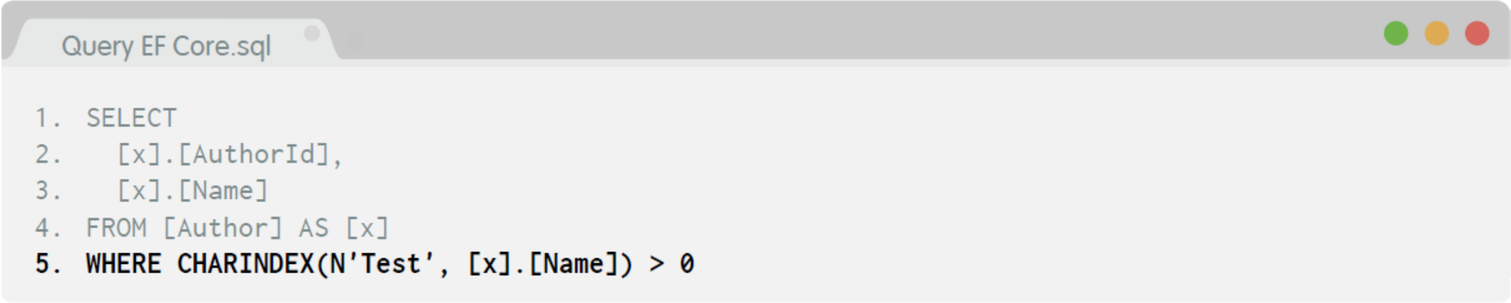
Contains (Strings)



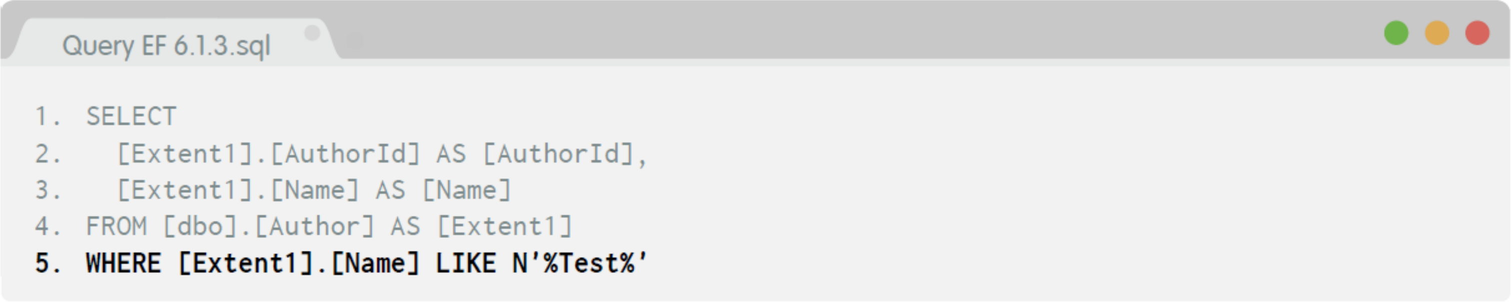
```
using (var db = new BloggingContext())
{
        var items = db.Author.Where(x => x.Name.Contains("Test")).ToList();
}
```

EF Core

LINQ And Function



```sql
SELECT
  [x].[AuthorId],
  [x].[Name]
FROM [Author] AS [x]
WHERE CHARINDEX(N'Test', [x].[Name]) > 0
```

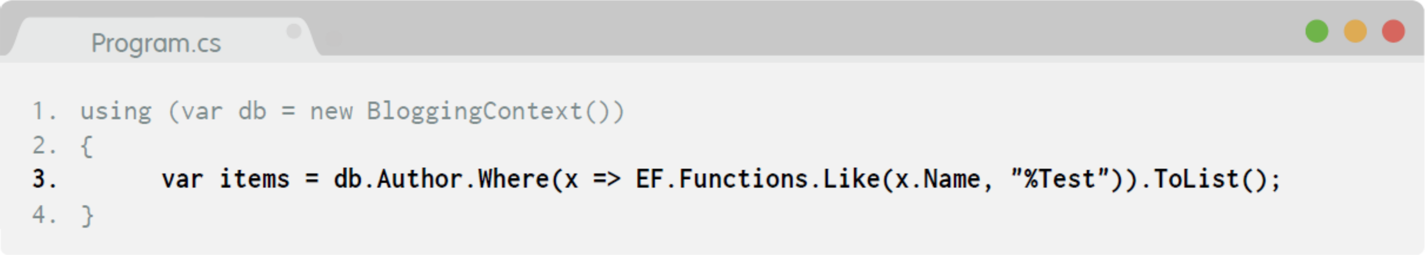


```sql
SELECT
  [Extent1].[AuthorId] AS [AuthorId],
  [Extent1].[Name] AS [Name]
FROM [dbo].[Author] AS [Extent1]
WHERE [Extent1].[Name] LIKE N'%Test%'
```

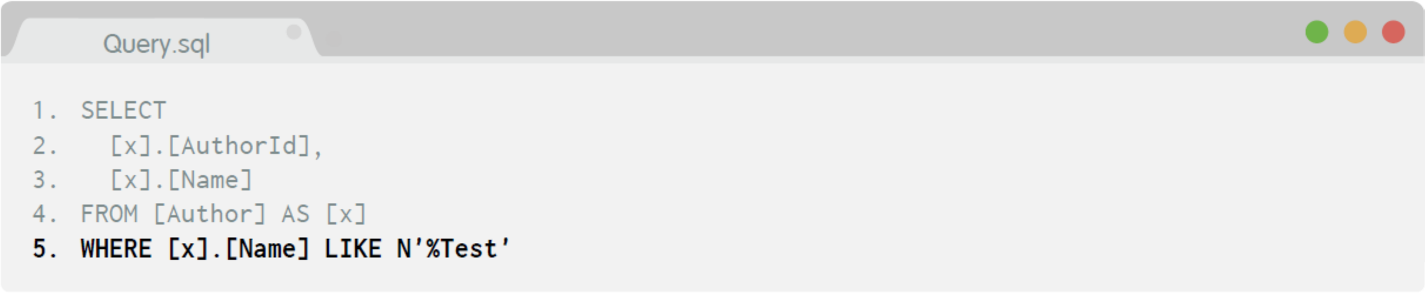
EF Core

LINQ And Function

Like (Db Functions)



```csharp
using (var db = new BloggingContext())
{
        var items = db.Author.Where(x => EF.Functions.Like(x.Name, "%Test")).ToList();
}
```

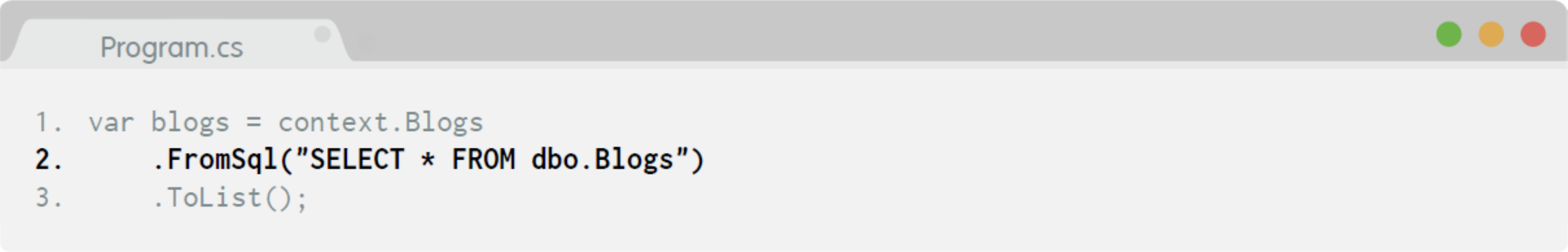


```sql
SELECT
  [x].[AuthorId],
  [x].[Name]
FROM [Author] AS [x]
WHERE [x].[Name] LIKE N'%Test'
```

EF
Core

LINQ And Function

Raw SQL Queries

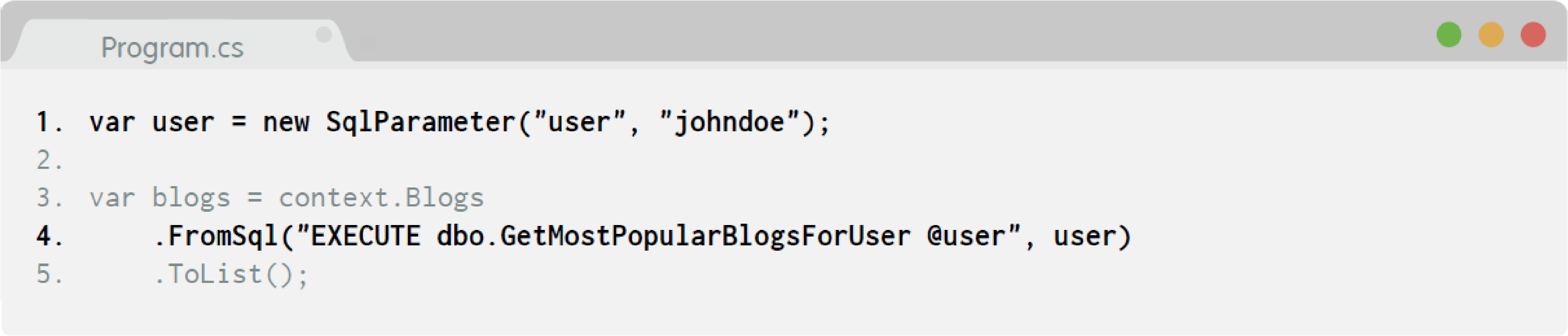


```
var blogs = context.Blogs
    .FromSql(”SELECT * FROM dbo.Blogs”)
    .ToList();
```

EF Core

LINQ And Function

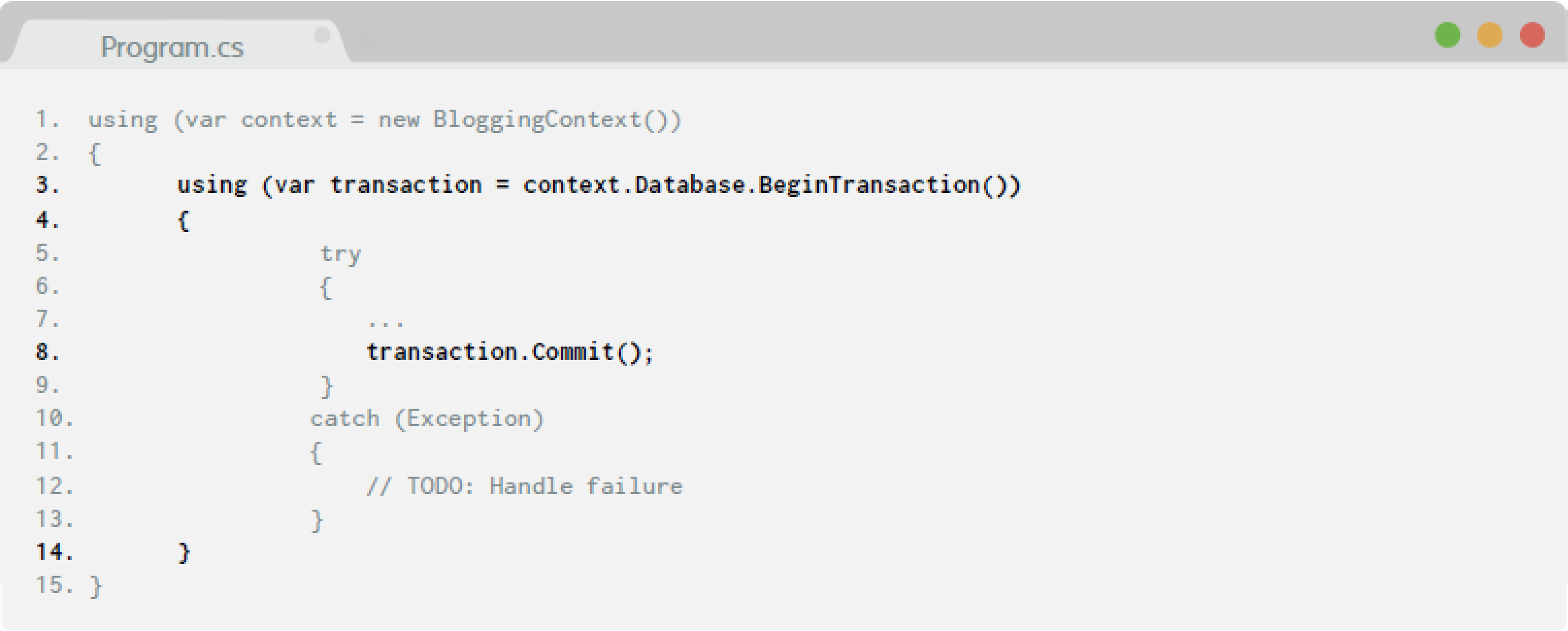
Passing Parameters



```
var user = new SqlParameter("user", "johndoe");

var blogs = context.Blogs
    .FromSql("EXECUTE dbo.GetMostPopularBlogsForUser @user", user)
    .ToList();
```

EF
Core

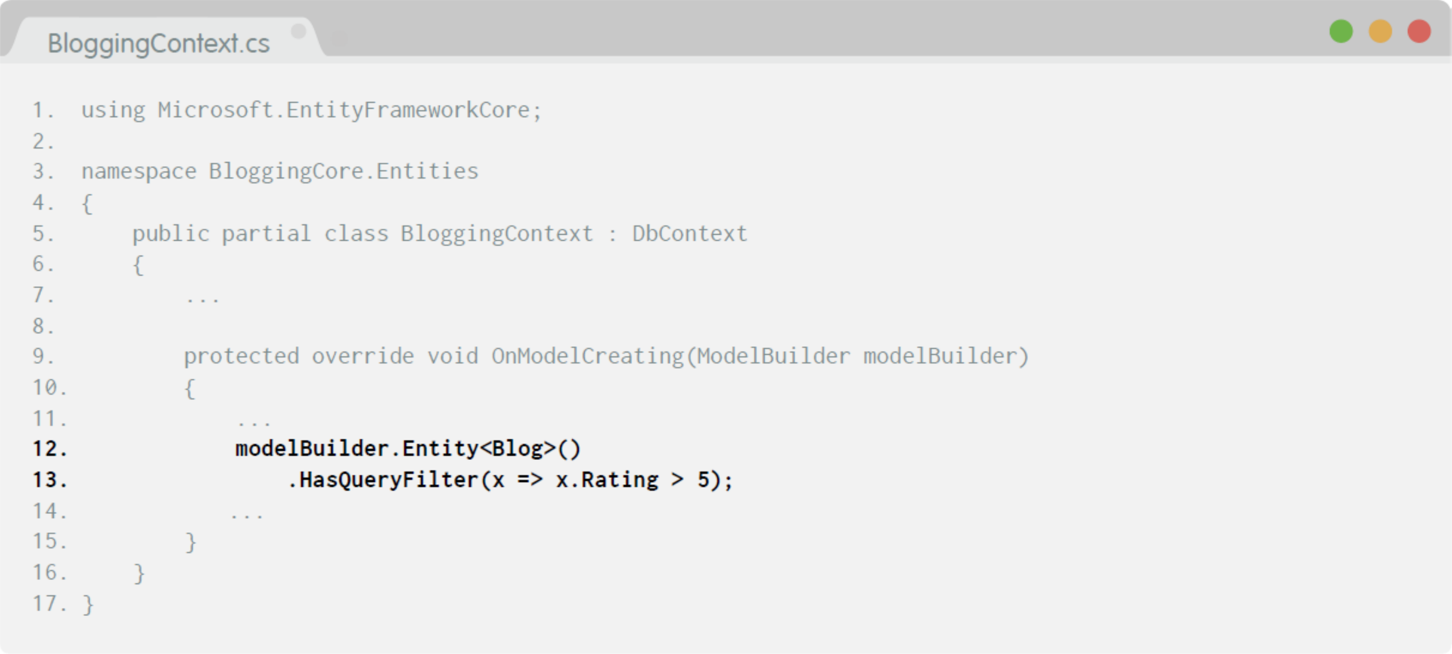
Transactions



```
using (var context = new BloggingContext())
{
    using (var transaction = context.Database.BeginTransaction())
    {
          try
          {
              ...
              transaction.Commit();
          }
         catch (Exception)
         {
              // TODO: Handle failure
         }
    }
}
```

EF Core

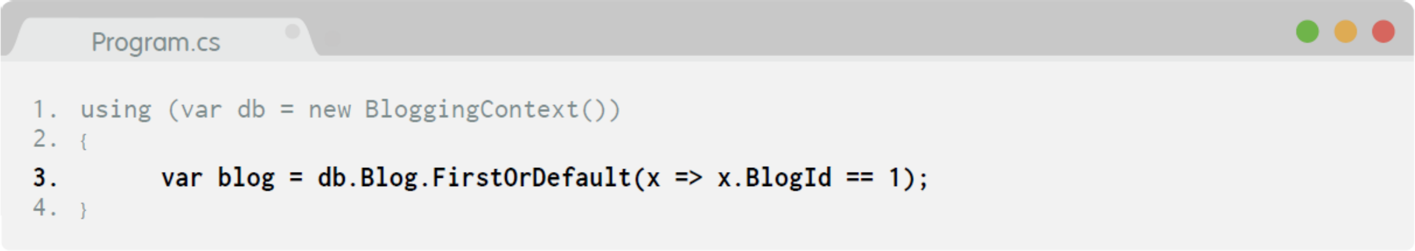
Global Query Filters



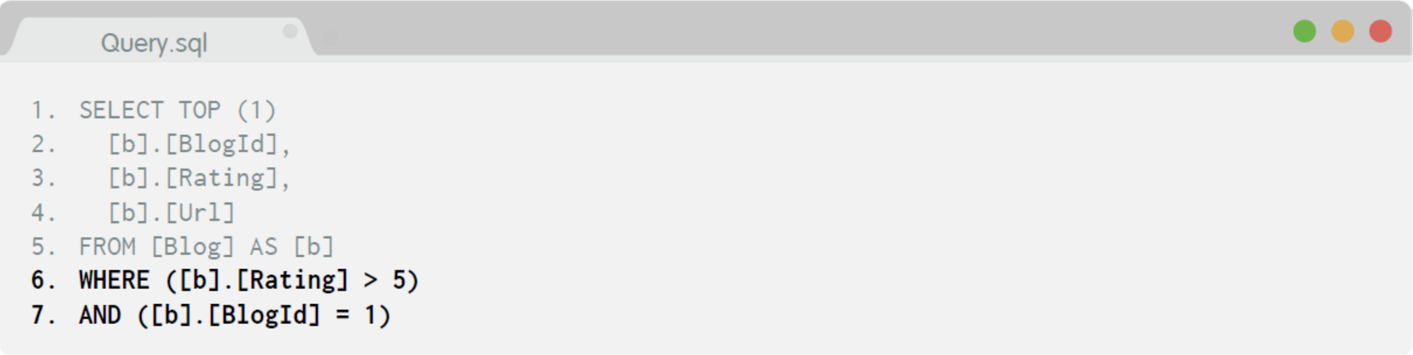
```
using Microsoft.EntityFrameworkCore;

namespace BloggingCore.Entities
{
    public partial class BloggingContext : DbContext
    {
        ...

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            ...
            modelBuilder.Entity<Blog>()
                .HasQueryFilter(x => x.Rating > 5);
            ...
        }
    }
}
```

EF Core

Global Query Filters



```
using (var db = new BloggingContext())
{
        var blog = db.Blog.FirstOrDefault(x => x.BlogId == 1);
}
```



```
SELECT TOP (1)
  [b].[BlogId],
  [b].[Rating],
  [b].[Url]
FROM [Blog] AS [b]
WHERE ([b].[Rating] > 5)
AND ([b].[BlogId] = 1)
```

EF Core

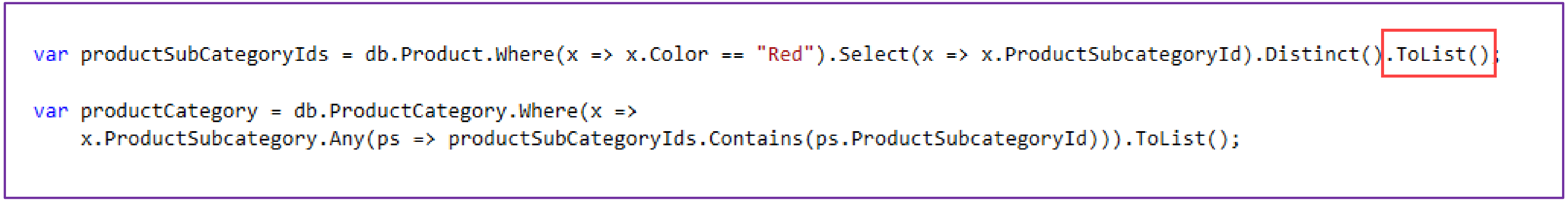
IQueryable vs IEnumerable

IQueryable<T> is the interface that allows LINQ-to-SQL that query will be executed in the database.

IEnumerable<T> is the interface that allows LINQ-to-object, meaning that all objects matching the original query will have to be loaded into memory from the database.

EF Core

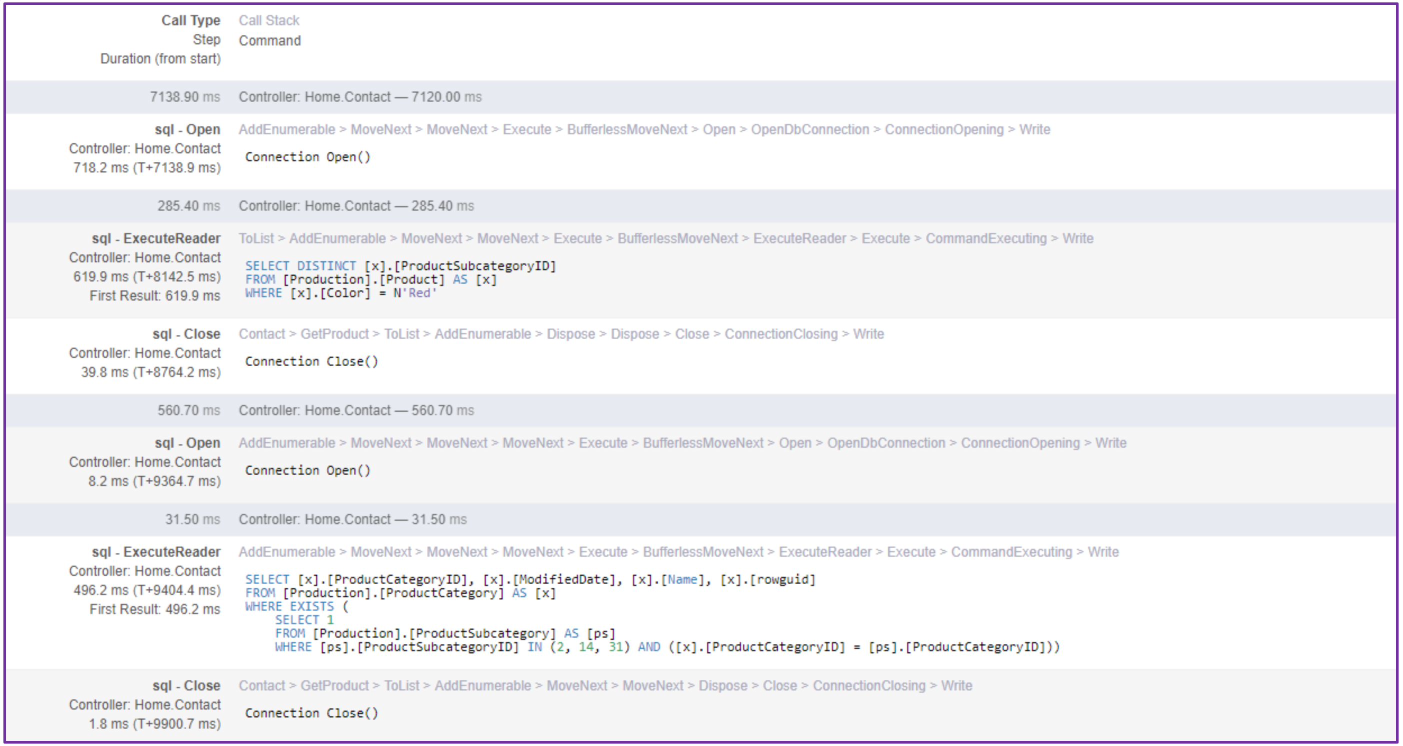
IEnumerable



```
var productSubCategoryIds = db.Product.Where(x => x.Color == "Red").Select(x => x.ProductSubcategoryId).Distinct().ToList();

var productCategory = db.ProductCategory.Where(x =>
    x.ProductSubcategory.Any(ps => productSubCategoryIds.Contains(ps.ProductSubcategoryId))).ToList();
```

EF Core

IEnumerable



Call Type Step Duration (from start)	Call Stack Command
7138.90 ms	Controller: Home.Contact — 7120.00 ms
sql - Open Controller: Home.Contact 718.2 ms (T+7138.9 ms)	AddEnumerable > MoveNext > MoveNext > Execute > BufferlessMoveNext > Open > OpenDbConnection > ConnectionOpening > Write `Connection Open()`
285.40 ms	Controller: Home.Contact — 285.40 ms

sql - ExecuteReader
Controller: Home.Contact
619.9 ms (T+8142.5 ms)
First Result: 619.9 ms
	ToList > AddEnumerable > MoveNext > MoveNext > Execute > BufferlessMoveNext > ExecuteReader > Execute > CommandExecuting > Write
	`SELECT DISTINCT [x].[ProductSubcategoryID]`
	`FROM [Production].[Product] AS [x]`
	`WHERE [x].[Color] = N'Red'`

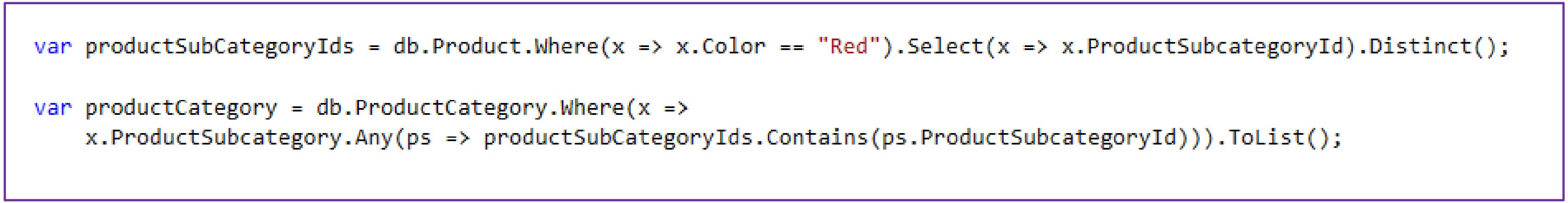
sql - Close Controller: Home.Contact 39.8 ms (T+8764.2 ms)	Contact > GetProduct > ToList > AddEnumerable > Dispose > Dispose > Close > ConnectionClosing > Write `Connection Close()`
560.70 ms	Controller: Home.Contact — 560.70 ms
sql - Open Controller: Home.Contact 8.2 ms (T+9364.7 ms)	AddEnumerable > MoveNext > MoveNext > MoveNext > Execute > BufferlessMoveNext > Open > OpenDbConnection > ConnectionOpening > Write `Connection Open()`
31.50 ms	Controller: Home.Contact — 31.50 ms

sql - ExecuteReader
Controller: Home.Contact
496.2 ms (T+9404.4 ms)
First Result: 496.2 ms
	AddEnumerable > MoveNext > MoveNext > MoveNext > Execute > BufferlessMoveNext > ExecuteReader > Execute > CommandExecuting > Write
	`SELECT [x].[ProductCategoryID], [x].[ModifiedDate], [x].[Name], [x].[rowguid]`
	`FROM [Production].[ProductCategory] AS [x]`
	`WHERE EXISTS (`
	`    SELECT 1`
	`    FROM [Production].[ProductSubcategory] AS [ps]`
	`    WHERE [ps].[ProductSubcategoryID] IN (2, 14, 31) AND ([x].[ProductCategoryID] = [ps].[ProductCategoryID]))`

sql - Close Controller: Home.Contact 1.8 ms (T+9900.7 ms)	Contact > GetProduct > ToList > AddEnumerable > MoveNext > MoveNext > Dispose > Close > ConnectionClosing > Write `Connection Close()`

EF Core

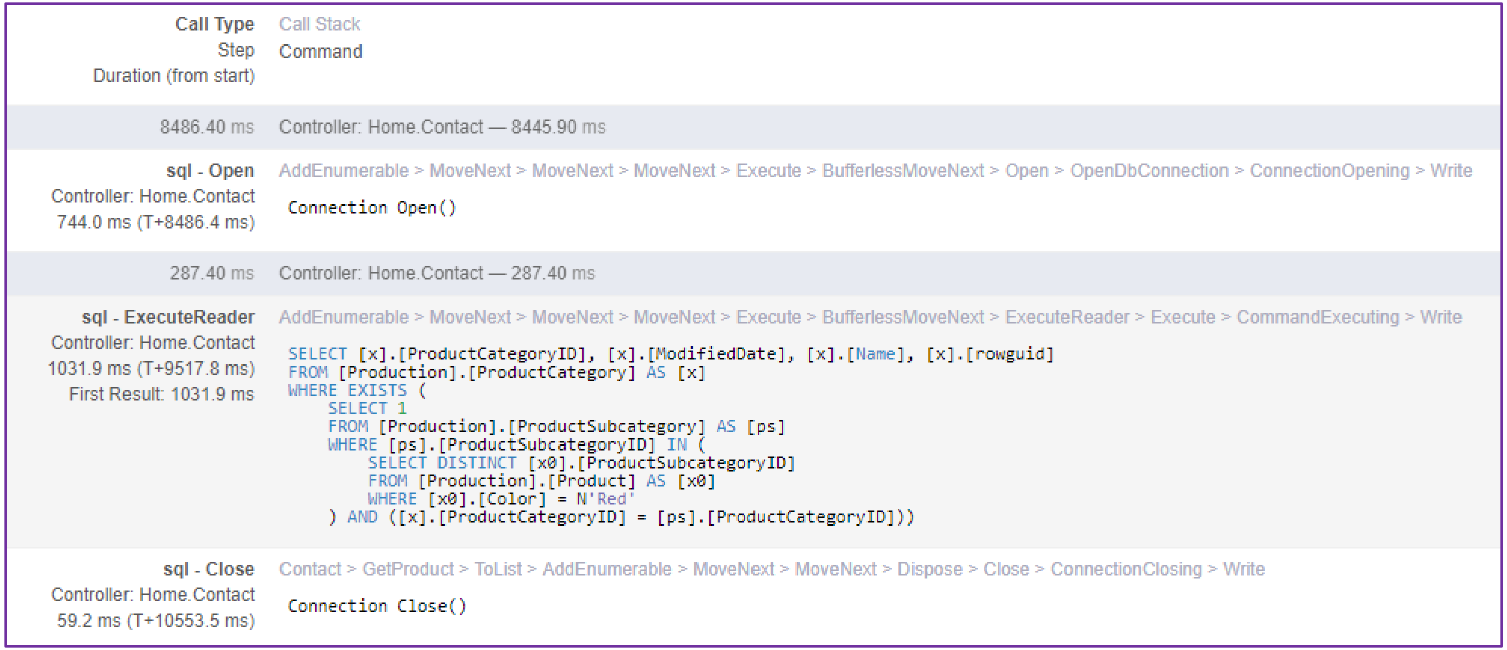
IQueryable



```
var productSubCategoryIds = db.Product.Where(x => x.Color == "Red").Select(x => x.ProductSubcategoryId).Distinct();

var productCategory = db.ProductCategory.Where(x =>
    x.ProductSubcategory.Any(ps => productSubCategoryIds.Contains(ps.ProductSubcategoryId))).ToList();
```

EF
Core

IQueryable



Call Type
Step
Duration (from start)

Call Stack
Command

8486.40 ms — Controller: Home.Contact — 8445.90 ms

sql - Open
Controller: Home.Contact
744.0 ms (T+8486.4 ms)

AddEnumerable > MoveNext > MoveNext > MoveNext > Execute > BufferlessMoveNext > Open > OpenDbConnection > ConnectionOpening > Write

```
Connection Open()
```

287.40 ms — Controller: Home.Contact — 287.40 ms

sql - ExecuteReader
Controller: Home.Contact
1031.9 ms (T+9517.8 ms)
First Result: 1031.9 ms

AddEnumerable > MoveNext > MoveNext > MoveNext > Execute > BufferlessMoveNext > ExecuteReader > Execute > CommandExecuting > Write

```
SELECT [x].[ProductCategoryID], [x].[ModifiedDate], [x].[Name], [x].[rowguid]
FROM [Production].[ProductCategory] AS [x]
WHERE EXISTS (
    SELECT 1
    FROM [Production].[ProductSubcategory] AS [ps]
    WHERE [ps].[ProductSubcategoryID] IN (
        SELECT DISTINCT [x0].[ProductSubcategoryID]
        FROM [Production].[Product] AS [x0]
        WHERE [x0].[Color] = N'Red'
    ) AND ([x].[ProductCategoryID] = [ps].[ProductCategoryID]))
```

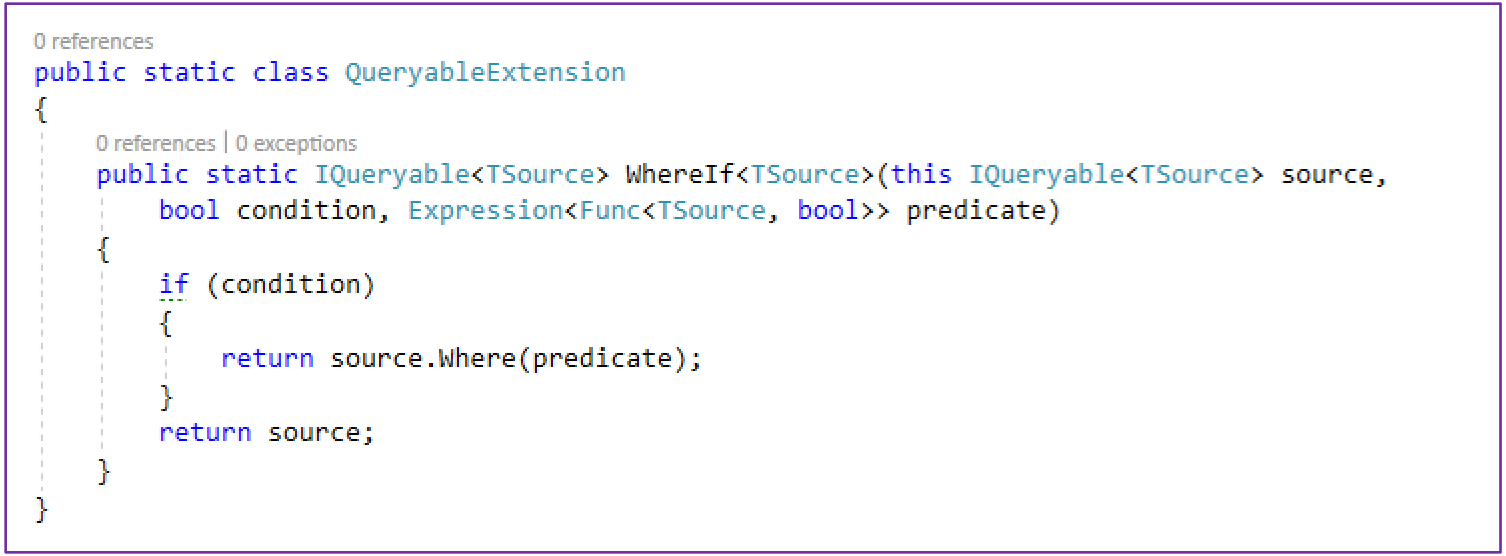
sql - Close
Controller: Home.Contact
59.2 ms (T+10553.5 ms)

Contact > GetProduct > ToList > AddEnumerable > MoveNext > MoveNext > Dispose > Close > ConnectionClosing > Write

```
Connection Close()
```

EF Core

Whereif Extension Method

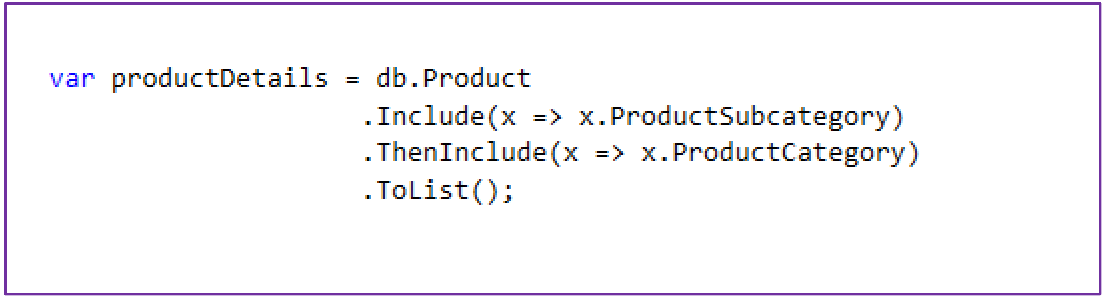


```
0 references
public static class QueryableExtension
{
    0 references | 0 exceptions
    public static IQueryable<TSource> WhereIf<TSource>(this IQueryable<TSource> source,
        bool condition, Expression<Func<TSource, bool>> predicate)
    {
        if (condition)
        {
            return source.Where(predicate);
        }
        return source;
    }
}
```

EF Core

Model Result

This query is result **all column**

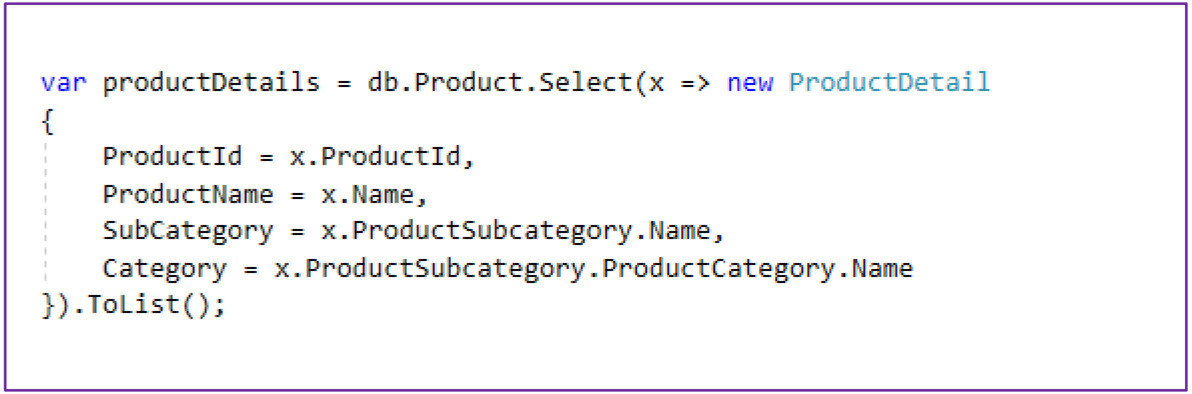


```
var productDetails = db.Product
                        .Include(x => x.ProductSubcategory)
                        .ThenInclude(x => x.ProductCategory)
                        .ToList();
```

EF Core

Model Result

This query is result **by model**



```
var productDetails = db.Product.Select(x => new ProductDetail
{
    ProductId = x.ProductId,
    ProductName = x.Name,
    SubCategory = x.ProductSubcategory.Name,
    Category = x.ProductSubcategory.ProductCategory.Name
}).ToList();
```

EF Core

Loading Related Data

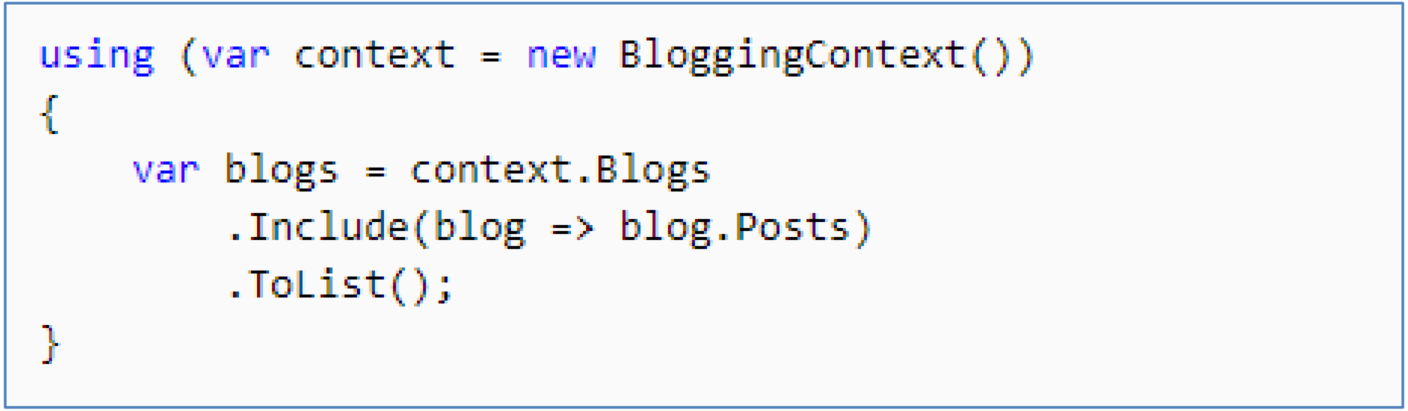
Eager loading
means that the related data is loaded from the database as part of the initial query.

Explicit loading
means that the related data is explicitly loaded from the database at a later time.

Lazy loading (This feature was introduced in EF Core 2.1)
means that the related data is transparently loaded from the database when the navigation property is accessed.

EF Core

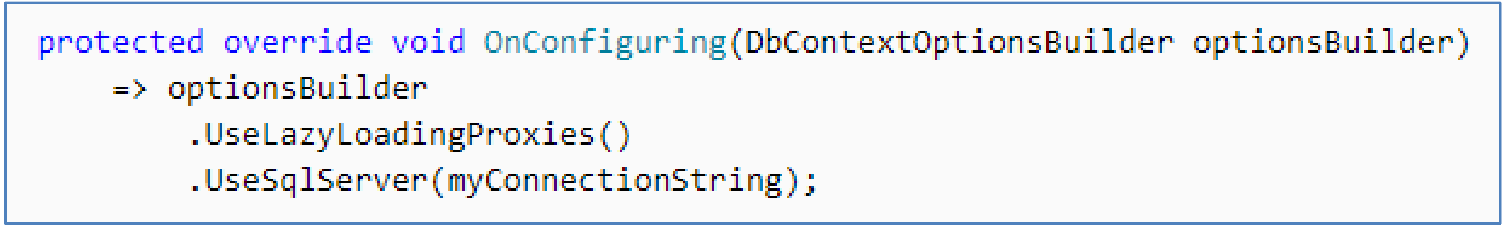
Eager loading



```
using (var context = new BloggingContext())
{
    var blogs = context.Blogs
        .Include(blog => blog.Posts)
        .ToList();
}
```

EF Core

Lazy loading



```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    => optionsBuilder
        .UseLazyLoadingProxies()
        .UseSqlServer(myConnectionString);
```

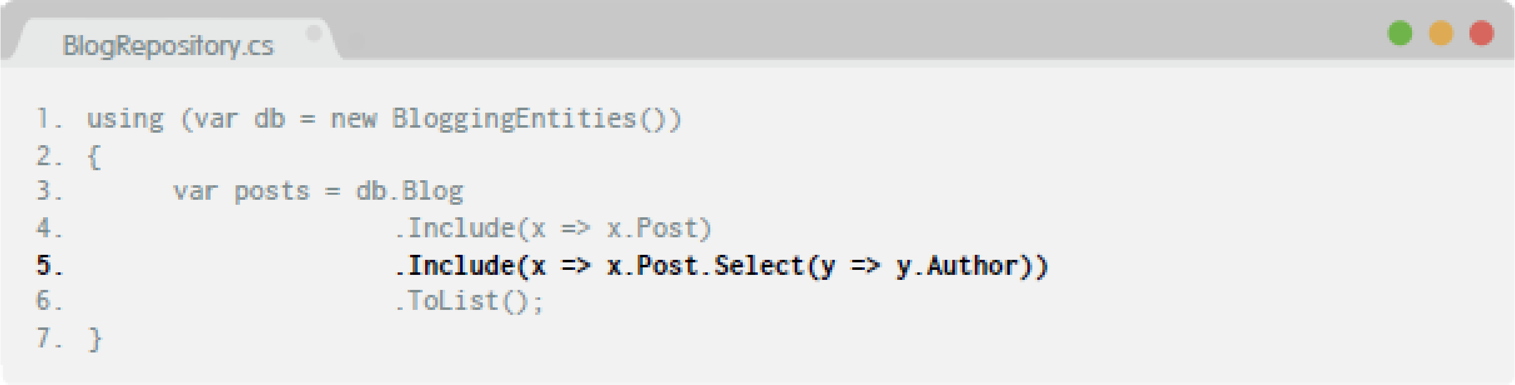
EF Core

Including multiple levels

You can drill down thru relationships to include multiple levels of related data using the **ThenInclude** method

EF Core

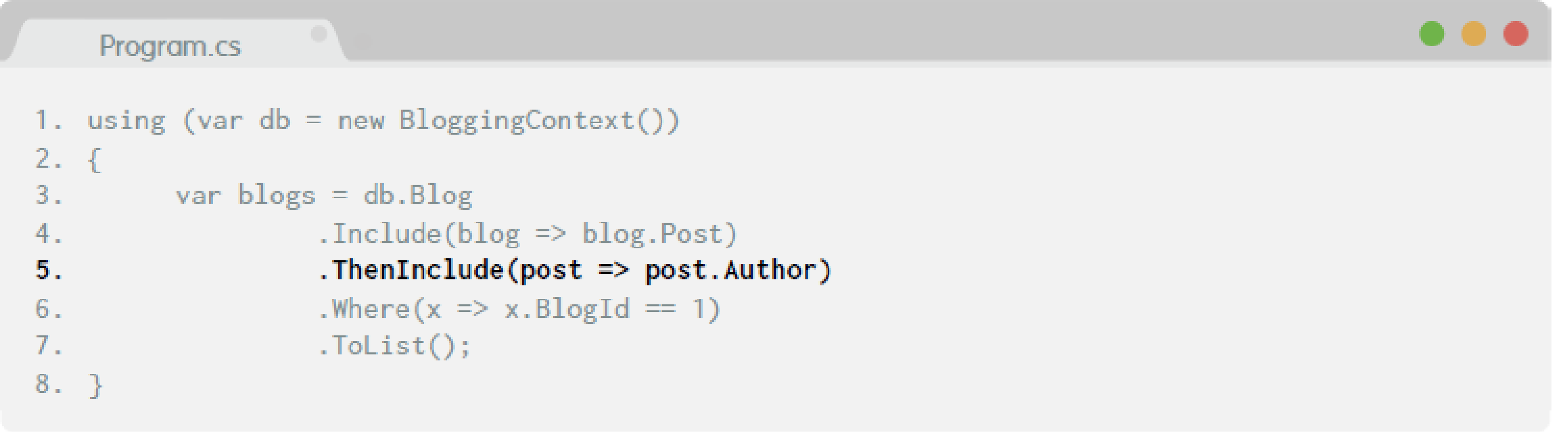
Including (EF 6.X)



```
using (var db = new BloggingEntities())
{
        var posts = db.Blog
                            .Include(x => x.Post)
                            .Include(x => x.Post.Select(y => y.Author))
                            .ToList();
}
```

EF Core

Including (EF Core)

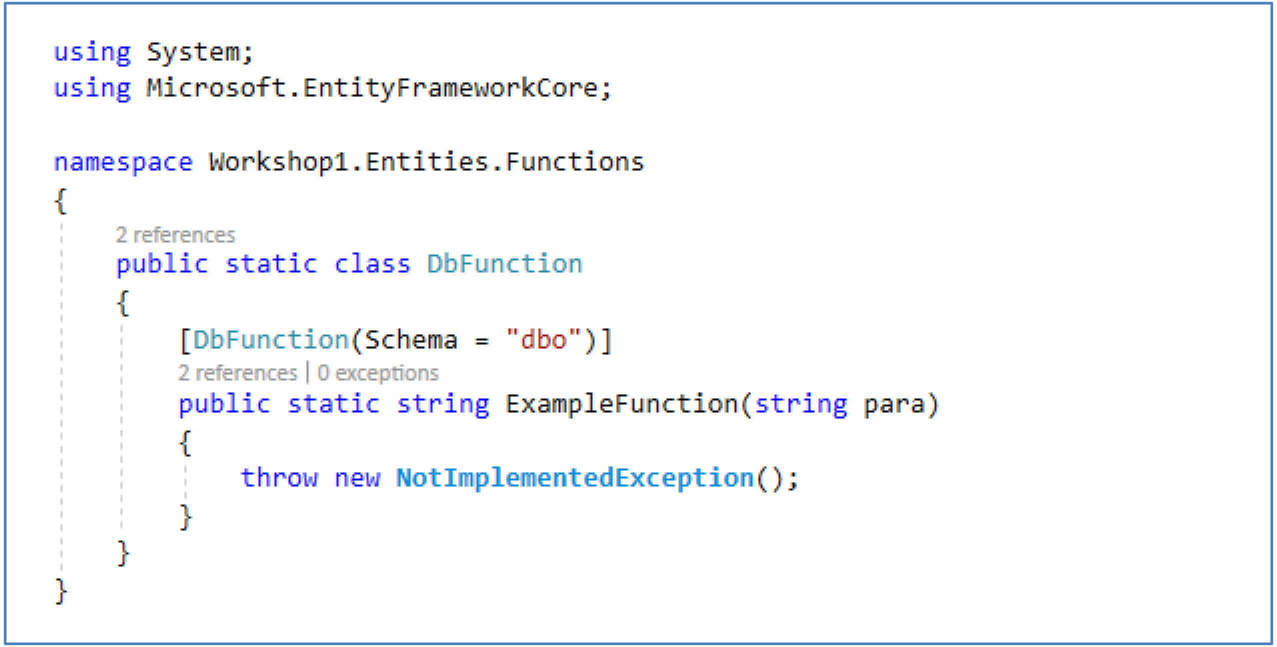


```
using (var db = new BloggingContext())
{
      var blogs = db.Blog
               .Include(blog => blog.Post)
               .ThenInclude(post => post.Author)
               .Where(x => x.BlogId == 1)
               .ToList();
}
```

EF Core

DB Function

Create **DbFunction** class

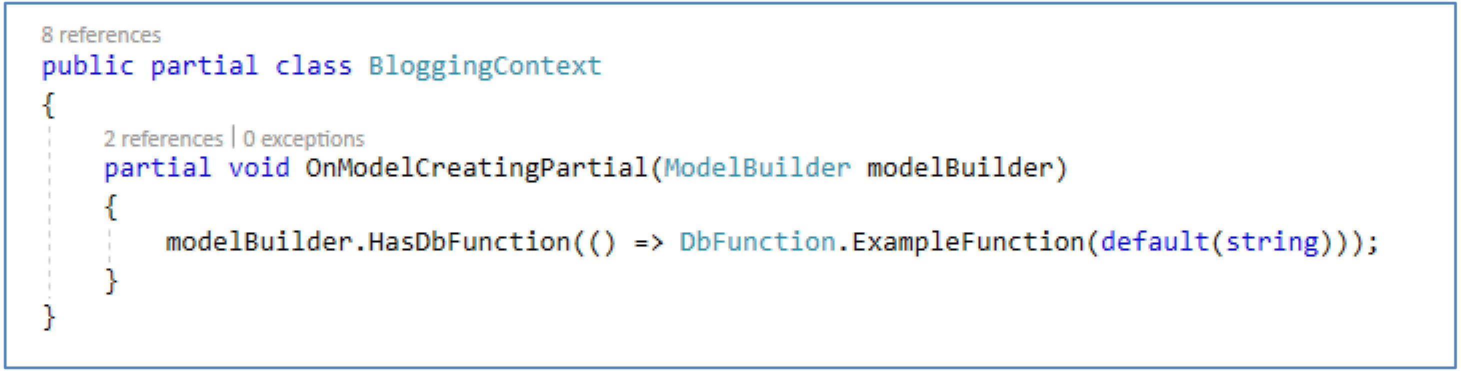


```
using System;
using Microsoft.EntityFrameworkCore;

namespace Workshop1.Entities.Functions
{
    2 references
    public static class DbFunction
    {
        [DbFunction(Schema = "dbo")]
        2 references | 0 exceptions
        public static string ExampleFunction(string para)
        {
            throw new NotImplementedException();
        }
    }
}
```

EF Core

DB Function

Register **Model Builder Dbfunction** to database context class

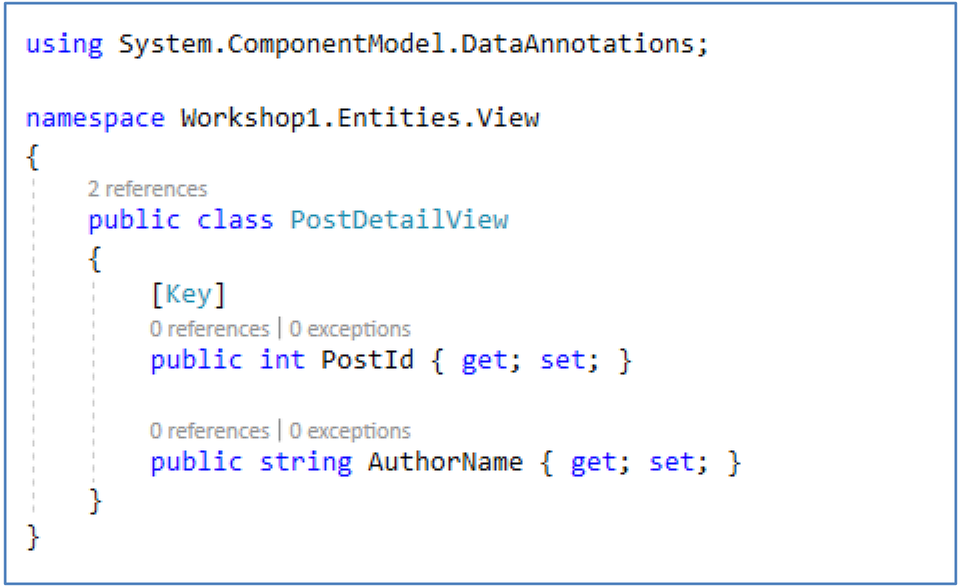


```
8 references
public partial class BloggingContext
{
    2 references | 0 exceptions
    partial void OnModelCreatingPartial(ModelBuilder modelBuilder)
    {
        modelBuilder.HasDbFunction(() => DbFunction.ExampleFunction(default(string)));
    }
}
```

EF Core

View

Create model result

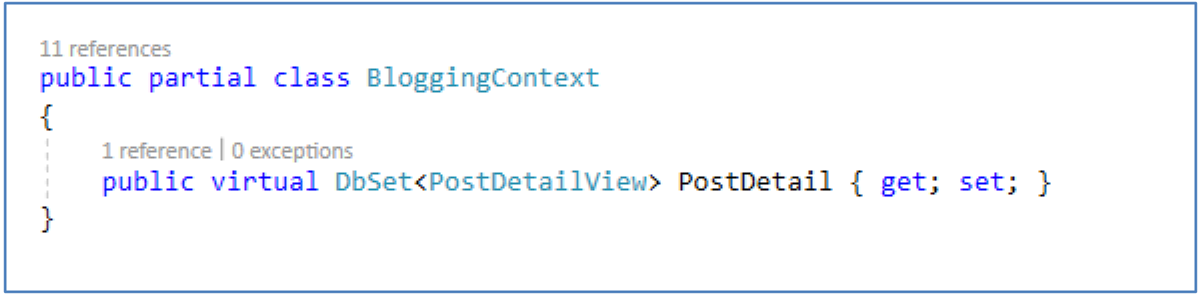


```
using System.ComponentModel.DataAnnotations;

namespace Workshop1.Entities.View
{
    2 references
    public class PostDetailView
    {
        [Key]
        0 references | 0 exceptions
        public int PostId { get; set; }

        0 references | 0 exceptions
        public string AuthorName { get; set; }
    }
}
```

EF Core

View

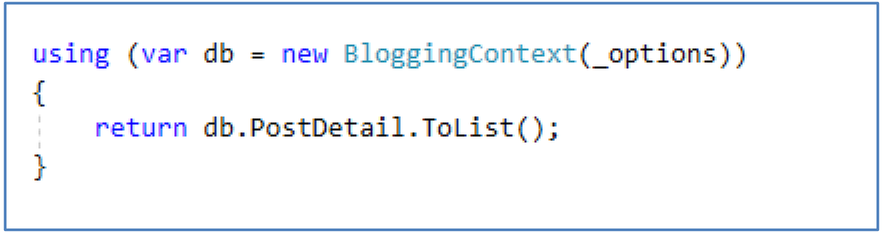
Add property



```
11 references
public partial class BloggingContext
{
    1 reference | 0 exceptions
    public virtual DbSet<PostDetailView> PostDetail { get; set; }
}
```

EF Core

View

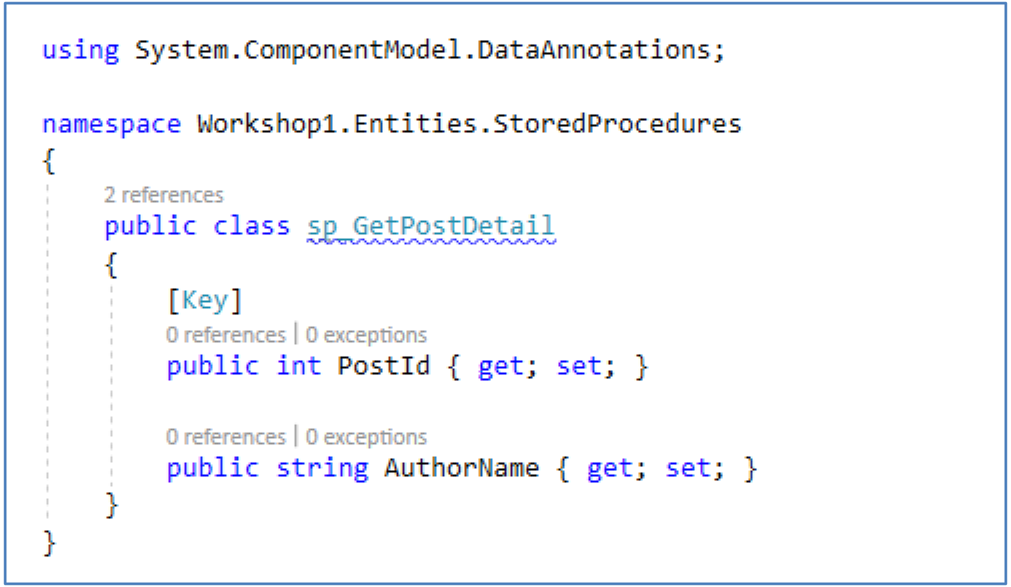


```
using (var db = new BloggingContext(_options))
{
    return db.PostDetail.ToList();
}
```

EF
Core

Stored Procedure

Create model result

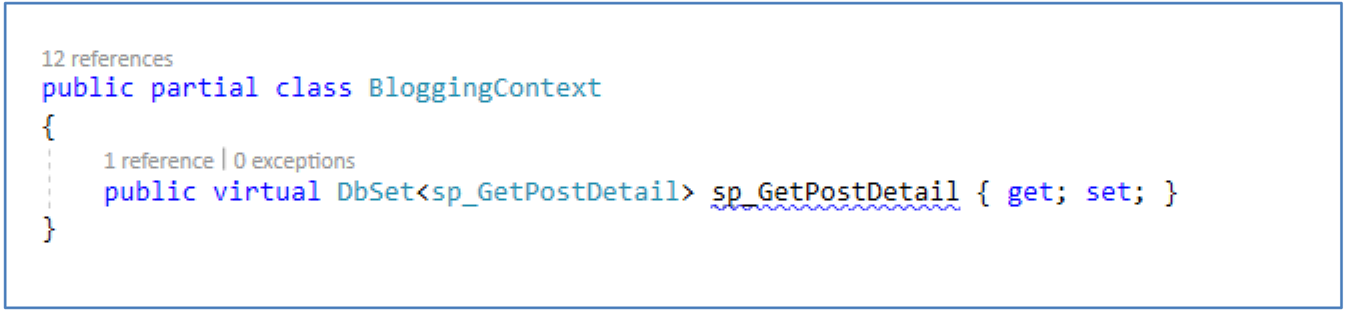


```
using System.ComponentModel.DataAnnotations;

namespace Workshop1.Entities.StoredProcedures
{
    2 references
    public class sp_GetPostDetail
    {
        [Key]
        0 references | 0 exceptions
        public int PostId { get; set; }

        0 references | 0 exceptions
        public string AuthorName { get; set; }
    }
}
```

EF Core

Stored Procedure

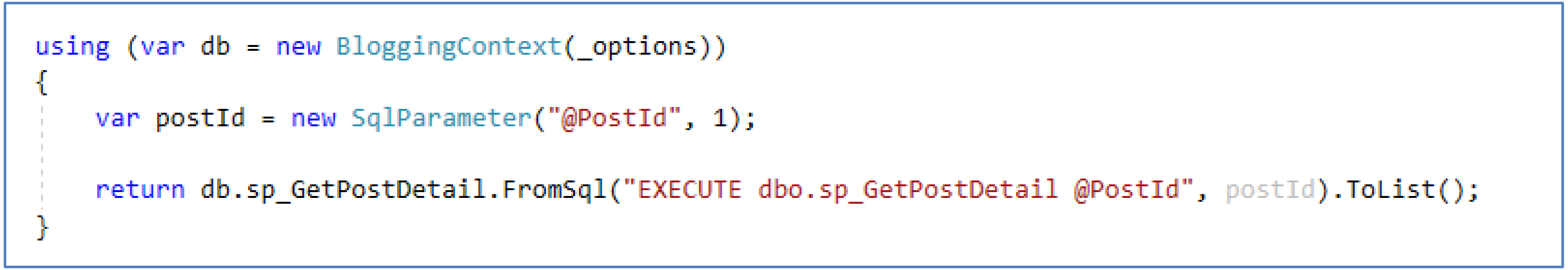
Add property



```
12 references
public partial class BloggingContext
{
    1 reference | 0 exceptions
    public virtual DbSet<sp_GetPostDetail> sp_GetPostDetail { get; set; }
}
```

EF Core

Stored Procedure

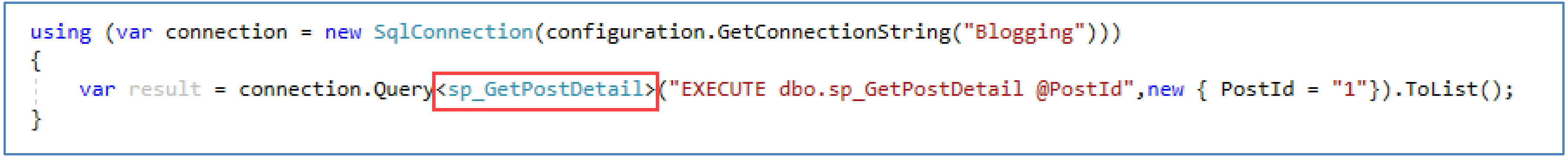


```
using (var db = new BloggingContext(_options))
{
    var postId = new SqlParameter("@PostId", 1);

    return db.sp_GetPostDetail.FromSql("EXECUTE dbo.sp_GetPostDetail @PostId", postId).ToList();
}
```

EF
Core

Dapper ORM

Install-Package dapper



```
using (var connection = new SqlConnection(configuration.GetConnectionString("Blogging")))
{
    var result = connection.Query<sp_GetPostDetail>("EXECUTE dbo.sp_GetPostDetail @PostId",new { PostId = "1"}).ToList();
}
```

EF Core

Workshop 2

Migration EF to EF Core

EF Core

Tools & Extensions

- EF Core Power Tools
- EFSecondLevelCache.Core
- EFCore.BulkExtensions
- More