CS6004NT
Application Development

WEEK - 06

ASP.NET Core Identity

Authentication and Authorization

ASP.NET Core Identity

Custom User Table/Entity

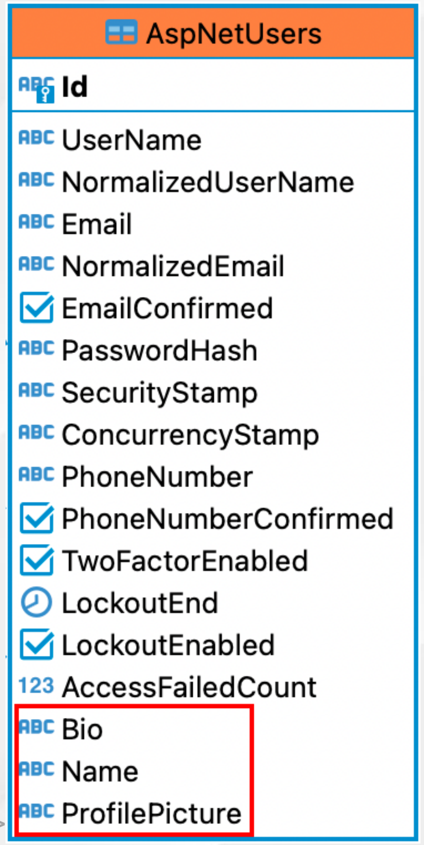
1. **Create a IdentityUser derived class**

```csharp
using Microsoft.AspNetCore.Identity;

namespace BookReview.Api.Infrastructure.Identity;

public class AppUser: IdentityUser
{
    public string Name { get; set; } = string.Empty;
    public string Bio { get; set; } = string.Empty;
    public string ProfilePicture { get; set; } = string.Empty;
    public IEnumerable<IdentityRole> Roles { get; set; } = Enumerable.Empty<IdentityRole>
}
```

11. **Replace all occurrence of *IdentityUser* with *AppUser***

    BookReviewDbContext.cs, Program.cs, SeedIdentityData.cs, etc.

1. **Generate and apply the migration**



AspNetUsers
- Id
- UserName
- NormalizedUserName
- Email
- NormalizedEmail
- EmailConfirmed
- PasswordHash
- SecurityStamp
- ConcurrencyStamp
- PhoneNumber
- PhoneNumberConfirmed
- TwoFactorEnabled
- LockoutEnd
- LockoutEnabled
- AccessFailedCount
- Bio
- Name
- ProfilePicture

ASP.NET Core Identity

Auth in Blazor Wasm

AuthController.cs:

```csharp
[Authorize]
[HttpGet("/api/auth/profile")]
public async Task<ActionResult<UserResponse>> GetProfileAsync()
{
    var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
    var user = await _authService.GetProfileAsync(userId);
    var userResponse = new UserResponse
    {
        Id = user.Id,
        Email = user.Email,
        EmailConfirmed = user.EmailConfirmed,
        Name = user.Name,
        Bio = user.Bio,
        ProfilePicture = user.ProfilePicture,
        Roles = User.FindAll(ClaimTypes.Role).Select(x => x.Value)
    };
    return Ok(userResponse);
}
```

ASP.NET Core Identity

Auth in Blazor Wasm

Program.cs (API):

```
// Turn redirect on auth error (default behaviour of [Authorize]) into error response
builder.Services.ConfigureApplicationCookie(options =>
{
    options.Events.OnRedirectToAccessDenied = context =>
    {
        var errorMessage = new ErrorMessageResponse { Message = "Forbidden" };
        context.Response.StatusCode = 403;
        context.Response.ContentType = "application/json";
        return context.Response.WriteAsync(JsonSerializer.Serialize(errorMessage));
    };
    options.Events.OnRedirectToLogin = context =>
    {
        var errorMessage = new ErrorMessageResponse { Message = "Unauthorized" };
        context.Response.StatusCode = 401;
        context.Response.ContentType = "application/json";
        return context.Response.WriteAsync(JsonSerializer.Serialize(errorMessage));
    };
});
```

```
// Allow cookie credentials to be in request and response
app.UseCors(
    policy =>
        policy
            .WithOrigins("http://localhost:3000", "https://localhost:3001")
            .AllowAnyMethod()
            .AllowAnyHeader()
            .AllowCredentials()
);

// In both Blazor and API to reduce the attack surface of Cross Site Scripting (XSS) attacks
// https://content-security-policy.com/
app.Use(async (ctx, next) =>
{
    ctx.Response.Headers.Add("Content-Security-Policy", "default-src 'self' ws: wss: https://localhost:3001 https://localhost:5001; script-src 'self' 'unsafe-inline' 'unsafe-eval'; img-src 'self' data:; style-src 'self' 'unsafe-inline';");
    await next();
});
```

ASP.NET Core Identity

Auth in Blazor Wasm

HttpClientExtension.cs:

```
namespace BookReview.BlazorWasm.Services;

public static class HttpClientExtension
{
    private static HttpContent Serialize(object data) => new
StringContent(JsonSerializer.Serialize(data, new JsonSerializerOptions(JsonSerializerDefaults.Web)),
Encoding.UTF8, "application/json");

    public static Task<HttpResponseMessage> AuthGetAsync(this HttpClient httpClient, string
requestUri)
    {
        var request = new HttpRequestMessage(HttpMethod.Get, requestUri);
        request.SetBrowserRequestCredentials(BrowserRequestCredentials.Include);
        return httpClient.SendAsync(request);
    }
    public static Task<HttpResponseMessage> AuthPostAsync(this HttpClient httpClient, string requestUri, HttpContent? content)
    {
        var request = new HttpRequestMessage(HttpMethod.Post, requestUri);
        request.Content = content;
        request.SetBrowserRequestCredentials(BrowserRequestCredentials.Include);
        return httpClient.SendAsync(request);
    }

    public static Task<HttpResponseMessage> AuthPostAsJsonAsync<T>(this HttpClient httpClient, string requestUri, T data) where T : class
    {
        var request = new HttpRequestMessage(HttpMethod.Post, requestUri);
        request.Content = Serialize(data);
        request.SetBrowserRequestCredentials(BrowserRequestCredentials.Include);

        return httpClient.SendAsync(request);
    }
    ...
}
```

ASP.NET Core Identity

Auth in Blazor Wasm

AuthService.cs (Blazor):

```csharp
namespace BookReview.BlazorWasm.Services;
public class AuthService : IAuthService
{
    // DI and constructor code ...
    public async Task LoginAsync(LoginRequest loginRequest)
    {
        var response = await _httpClient.AuthPostAsJsonAsync("/api/auth/login", loginRequest);
        await CheckForErrorResponse(response);
    }
   public async Task<UserResponse?> GetProfileAsync()
    {
        var response = await _httpClient.AuthGetAsync("/api/auth/profile");
        await CheckForErrorResponse(response);
        var result = await response.Content.ReadFromJsonAsync<UserResponse>();
        return result;
    }
    ...
}
```

ASP.NET Core Identity

Auth in Blazor Wasm

AppStateService.cs:

```csharp
// Service to share state globally
namespace BookReview.BlazorWasm.Services;

public class AppStateService
{
    public UserResponse? CurrentUser { get; private set; }
    public void SetCurrentUser(UserResponse? value)
    {
        CurrentUser = value;
        OnStateChange?.Invoke();
    }
    public event Action? OnStateChange;
}
```

ASP.NET Core Identity

Auth in Blazor Wasm

Program.cs (Blazor):

```
// Add AppStateService and AuthService to DI
builder.Services.AddSingleton<AppStateService>(); // Singleton to share same instance across the app
builder.Services.AddScoped<IAuthService, AuthService>();
```

Login.razor:

```
@page "/login"
@inject IAuthService AuthService
@inject NavigationManager NavManager
@inject AppStateService StateService

<div class="login-body">
    <div class="form-login">
        <form class="text-center" @onsubmit="LoginHandler">
// Rest of the html for login form...

@code {
    // properties for username, password, error message etc.
    private async Task LoginHandler()
    {
        try
        {
            _errorMessage = "";
            if (Username != null && Password != null)
            {
                var loginRequest = new LoginRequest
                {
                    Username = Username,
                    Password = Password,
                    RememberMe = RememberMe
                };
                await AuthService.LoginAsync(loginRequest);
                var currentUser = await AuthService.GetProfileAsync();
                StateService.SetCurrentUser(currentUser);
                NavManager.NavigateTo("/");
            }
        }
        catch (Exception e)
        {
            _errorMessage = e.Message;
        }
    }
}
```

Index.razor:

```
@page "/"
@inject NavigationManager NavManager

@code {
    protected override void OnInitialized()
    {
        NavManager.NavigateTo("/books");
    }
}
```

ASP.NET Core Identity

Auth in Blazor Wasm

MainLayout.razor:

```
@inherits LayoutComponentBase
@inject AppStateService StateService
@inject IAuthService AuthService

// layout html codes...
@code {
    protected override async Task OnInitializedAsync()
    {
        try
        { // If user is logged-in, fetch profile and set it to currentUser
            var currentUser = await AuthService.GetProfileAsync();
            StateService.SetCurrentUser(currentUser);
            StateHasChanged(); // Notifies the component that its state has changed
        }
        catch
        {
    // ignored
        }
    }
}
```

ASP.NET Core Identity

Auth in Blazor Wasm

Use of OnStateChange Delegate:

```
// In a page where auth is required. Example: create/edit book page

protected override void OnInitialized()
{
    StateService.OnStateChange += StateService_StateChanged; // When StateService.SetCurrentUser is invoked
}

private void StateService_StateChanged()
{
    StateHasChanged(); // Notifies the component that its state has changed.
    if (StateService.CurrentUser == null)
    {
        NavManager.NavigateTo("/books");
    }
}
```

ASP.NET Core Identity

Auth in Blazor Wasm

Role-based Access in Blazor:

```
@if (StateService.CurrentUser?.Roles.Contains("Admin") ?? false)
{
    <a class="edit" title="Edit" role="button" href="/edit-book/@(book.Id)">
        <span class="oi oi-pencil"></span>
    </a>
    <button class="delete" title="Delete" type="button" @onclick="() => OnDeleteBook(book)">
        <span class="oi oi-delete"></span>
    </button>
}
```

ASP.NET Core Identity

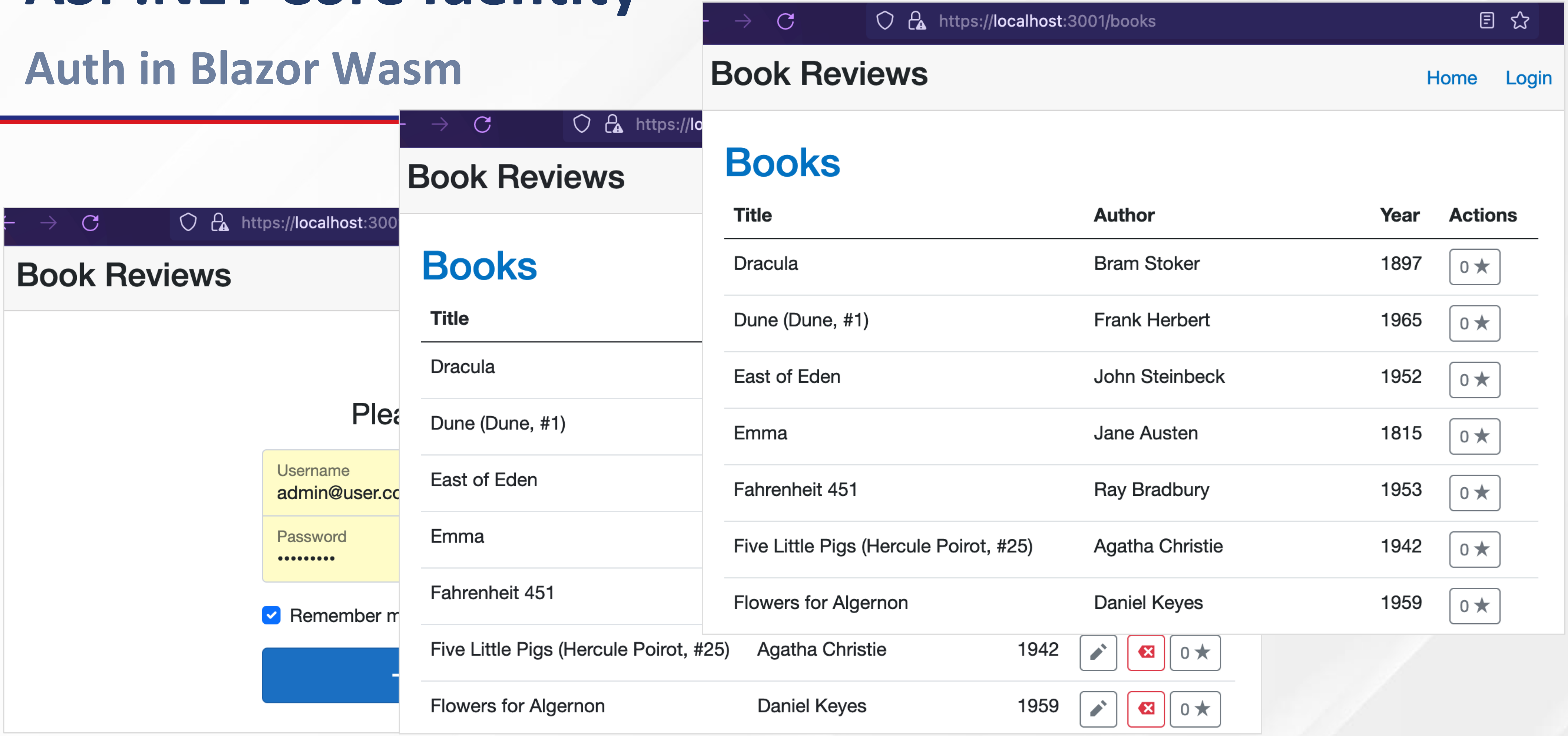
Auth in Blazor Wasm

Role-based Access in API Controller:

```
[Authorize(Roles = "Admin")]
[HttpDelete("/api/books/{id:int}")]
public async Task<IActionResult> DeleteBookAsync(int id)
{
    await _bookService.DeleteBookAsync(id);

    return NoContent();
}
```

ASP.NET Core Identity

Auth in Blazor Wasm



https://localhost:300

Book Reviews

Plea

Username
admin@user.c

Password
••••••••••

Remember m

https://lo

Book Reviews

Books

Title			
Dracula			
Dune (Dune, #1)			
East of Eden			
Emma			
Fahrenheit 451			
Five Little Pigs (Hercule Poirot, #25)	Agatha Christie	1942	0★
Flowers for Algernon	Daniel Keyes	1959	0★

https://localhost:3001/books

Book Reviews

Books

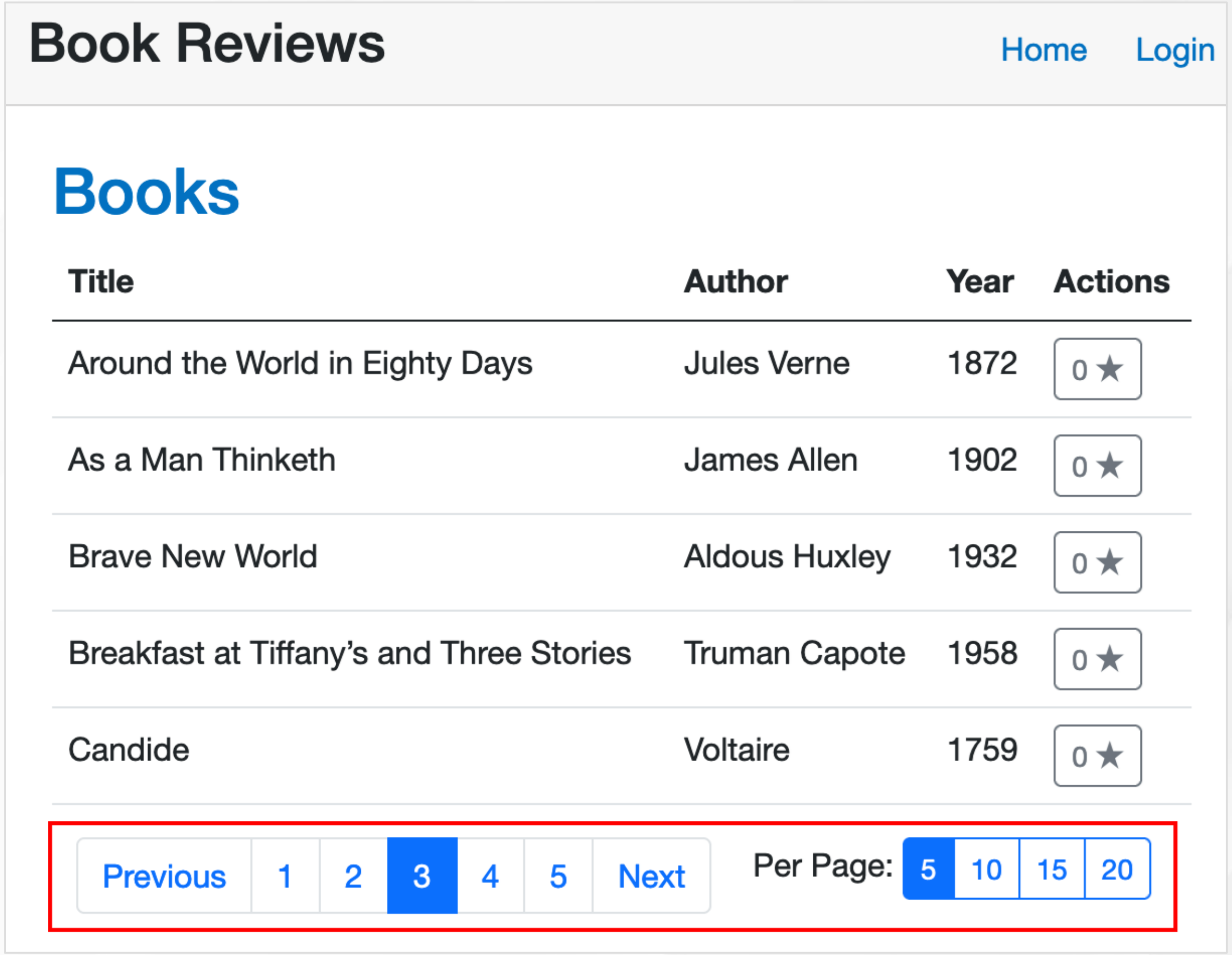
Title	Author	Year	Actions
Dracula	Bram Stoker	1897	0★
Dune (Dune, #1)	Frank Herbert	1965	0★
East of Eden	John Steinbeck	1952	0★
Emma	Jane Austen	1815	0★
Fahrenheit 451	Ray Bradbury	1953	0★
Five Little Pigs (Hercule Poirot, #25)	Agatha Christie	1942	0★
Flowers for Algernon	Daniel Keyes	1959	0★

EF Core

Entity Framework Core

EF Core

Pagination



Book Reviews

Books

Title	Author	Year	Actions
Around the World in Eighty Days	Jules Verne	1872	0 ★
As a Man Thinketh	James Allen	1902	0 ★
Brave New World	Aldous Huxley	1932	0 ★
Breakfast at Tiffany's and Three Stories	Truman Capote	1958	0 ★
Candide	Voltaire	1759	0 ★

EF Core

Pagination

PaginatedResponse.cs:

```
namespace BookReview.Contracts;

public class PaginatedResponse<T>
{
        public int PageNumber { get; set; }
        public int PageSize { get; set; }
        public int TotalPages { get; set; }
        public int TotalRecords { get; set; }
        public IEnumerable<T> Data { get; set; } = Enumerable.Empty<T>();
}
```

PaginationFilter.cs:

```
namespace BookReview.Contracts;

public class PaginationFilter
{
    public int PageNumber { get; set; } = 1;
    public int PageSize { get; set; } = 10;
}
```

EF Core

Pagination

BookController.cs:

```
[HttpGet("/api/books")]
public async Task<ActionResult<PaginatedResponse<BookResponse>>> GetAllBooksAsync([FromQuery] PaginationFilter filter)
{
    var totalRecords = await _bookService.CountAsync();
    var books = await _bookService.GetAllBooksAsync(filter.PageNumber, filter.PageSize);
    var response = new PaginatedResponse<BookResponse>
    {
        PageNumber = filter.PageNumber,
        PageSize = filter.PageSize,
        TotalPages = (int)Math.Ceiling((double)totalRecords / filter.PageSize),
        TotalRecords = totalRecords,
        Data = books.Select(x => new BookResponse
        {
            // Property Mapping Id, Title, Year, AuthorName etc.
        })
    };
    return Ok(response);
}
```

EF Core

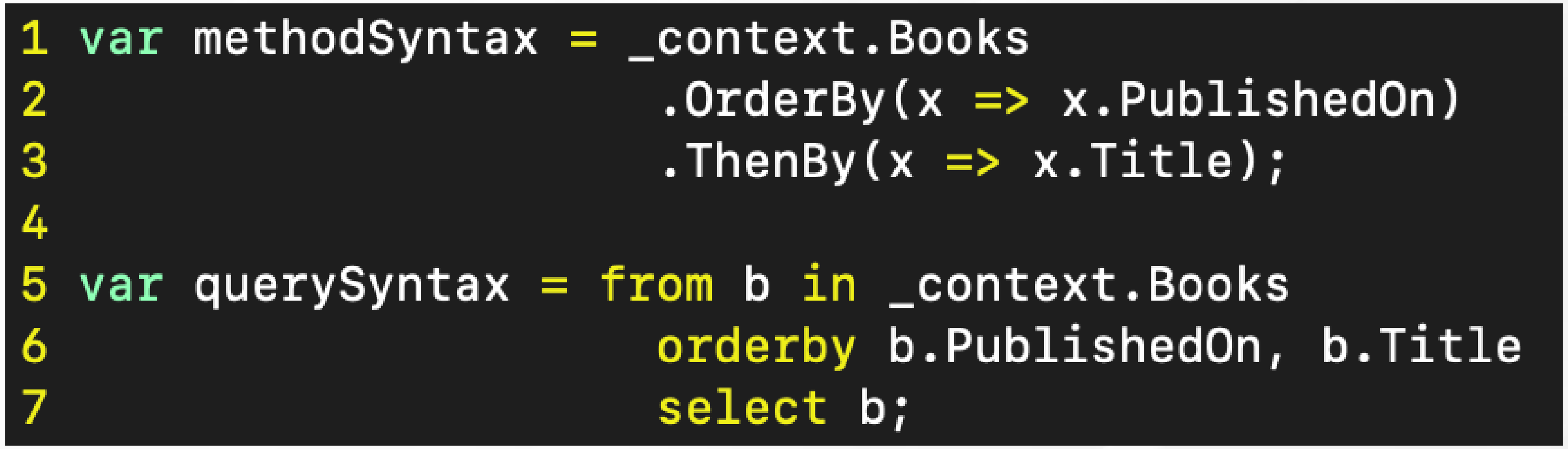
Pagination

BookRepository.cs:

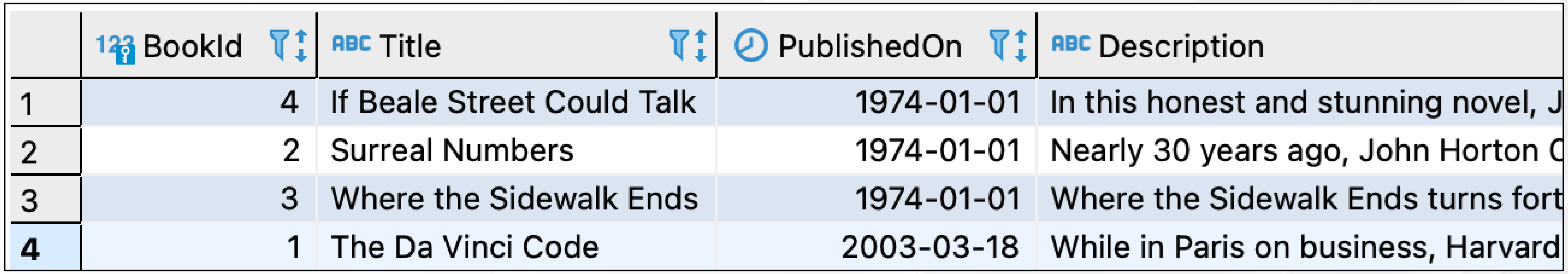
```
public async Task<IEnumerable<Book>> GetAllWithDetailsPaginatedAsync(int pageNumber, int
pageSize)
{
    var result = await _dbContext.Books
        .Include(x => x.Author)
        .Include(x => x.Reviews)
        .Skip((pageNumber - 1) * pageSize)
        .Take(pageSize)
        .ToListAsync();

    return result;
}
```

EF Core

Order By



```
var methodSyntax = _context.Books
                     .OrderBy(x => x.PublishedOn)
                     .ThenBy(x => x.Title);

var querySyntax = from b in _context.Books
                     orderby b.PublishedOn, b.Title
                     select b;
```

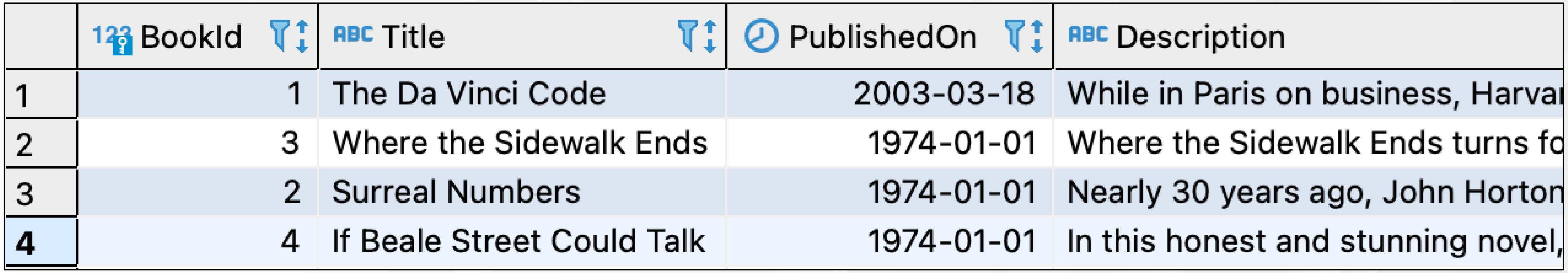


	BookId	Title	PublishedOn	Description
1	4	If Beale Street Could Talk	1974-01-01	In this honest and stunning novel, J
2	2	Surreal Numbers	1974-01-01	Nearly 30 years ago, John Horton C
3	3	Where the Sidewalk Ends	1974-01-01	Where the Sidewalk Ends turns fort
4	1	The Da Vinci Code	2003-03-18	While in Paris on business, Harvard

EF Core

Order By

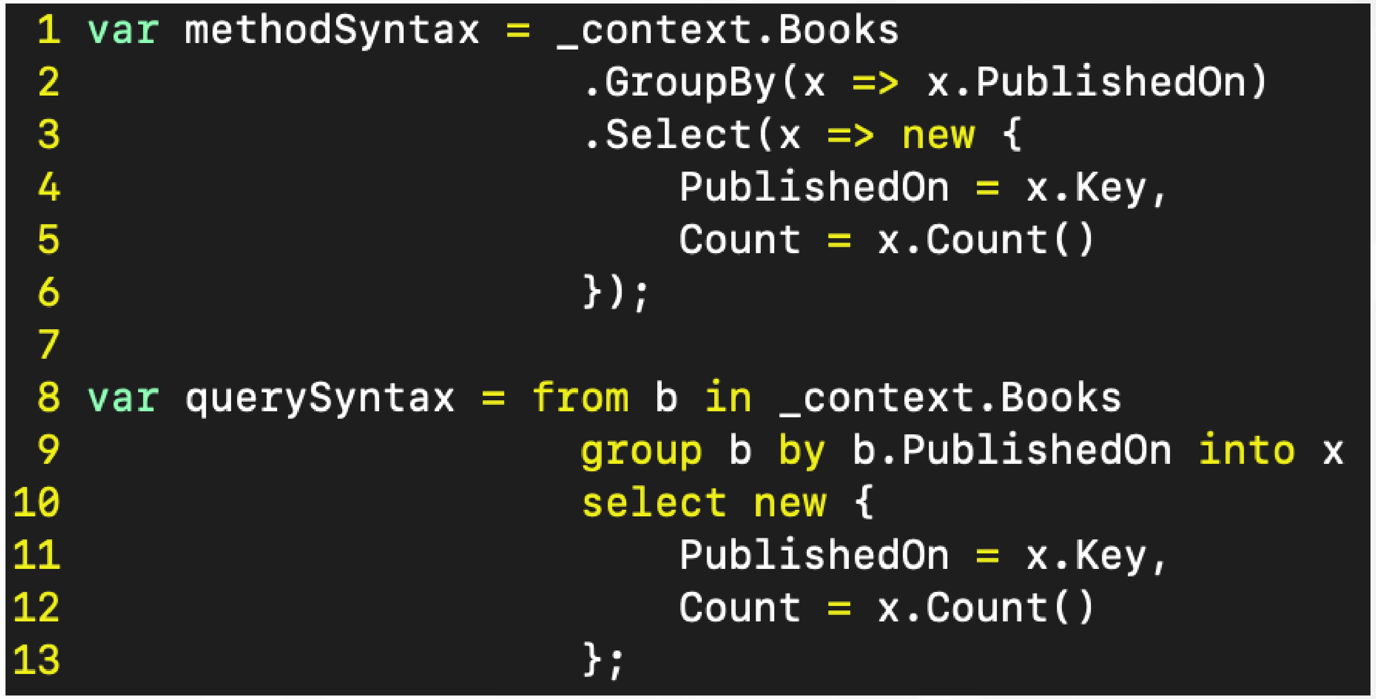
```
var methodSyntax = _context.Books
                    .OrderByDescending(x => x.PublishedOn)
                    .ThenByDescending(x => x.Title);

var querySyntax = from b in _context.Books
                    orderby b.PublishedOn descending, b.Title descending
                    select b;
```



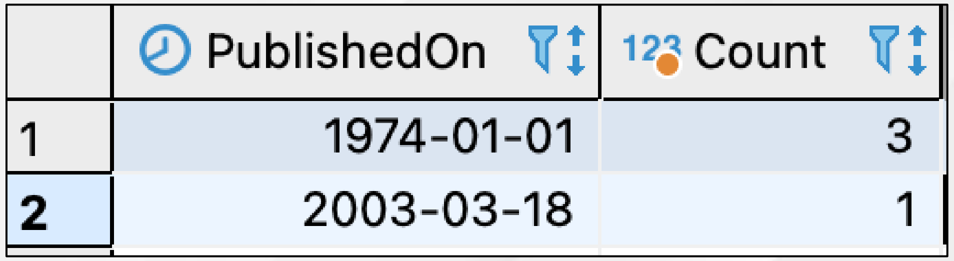
	BookId	Title	PublishedOn	Description
1	1	The Da Vinci Code	2003-03-18	While in Paris on business, Harva
2	3	Where the Sidewalk Ends	1974-01-01	Where the Sidewalk Ends turns fo
3	2	Surreal Numbers	1974-01-01	Nearly 30 years ago, John Horton
4	4	If Beale Street Could Talk	1974-01-01	In this honest and stunning novel,

EF Core

Group By



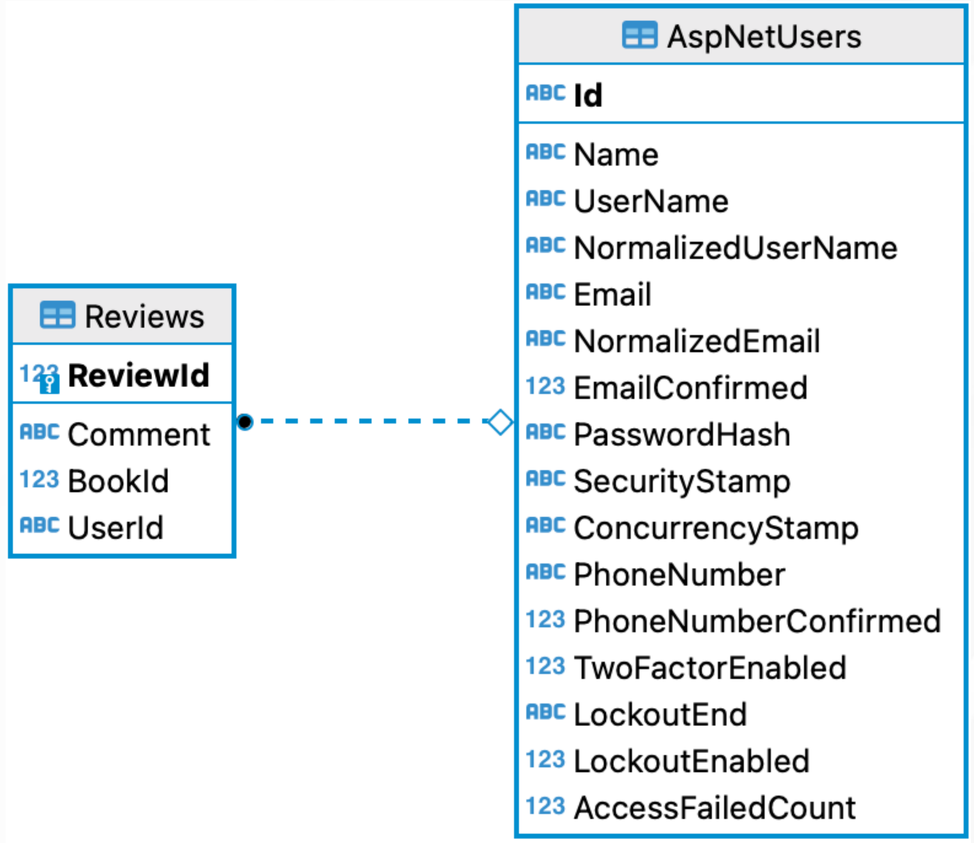
```
var methodSyntax = _context.Books
                          .GroupBy(x => x.PublishedOn)
                          .Select(x => new {
                              PublishedOn = x.Key,
                              Count = x.Count()
                          });

var querySyntax = from b in _context.Books
                          group b by b.PublishedOn into x
                          select new {
                              PublishedOn = x.Key,
                              Count = x.Count()
                          };
```

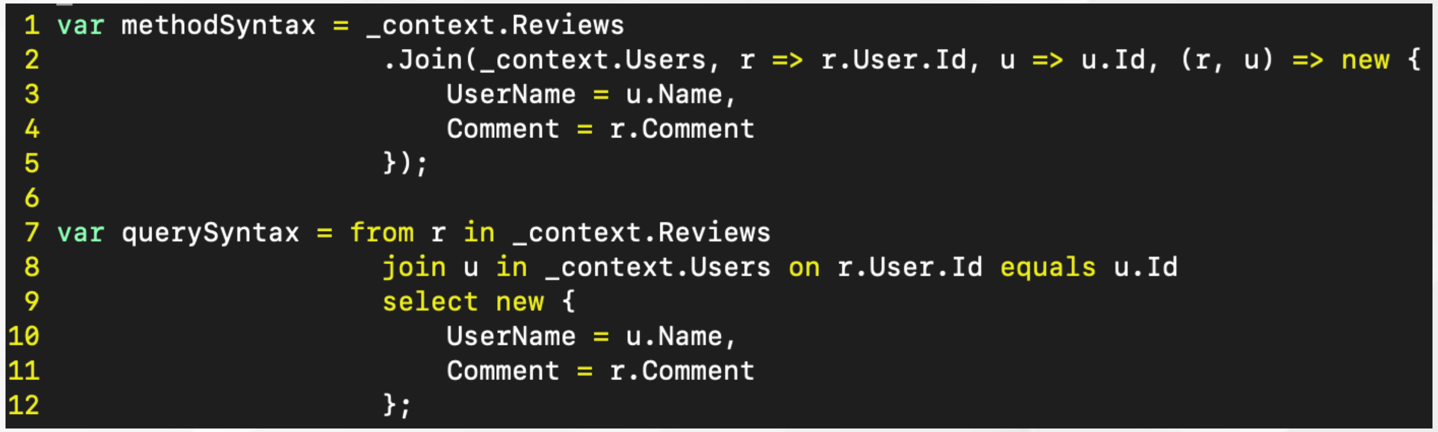


	PublishedOn	Count
1	1974-01-01	3
2	2003-03-18	1

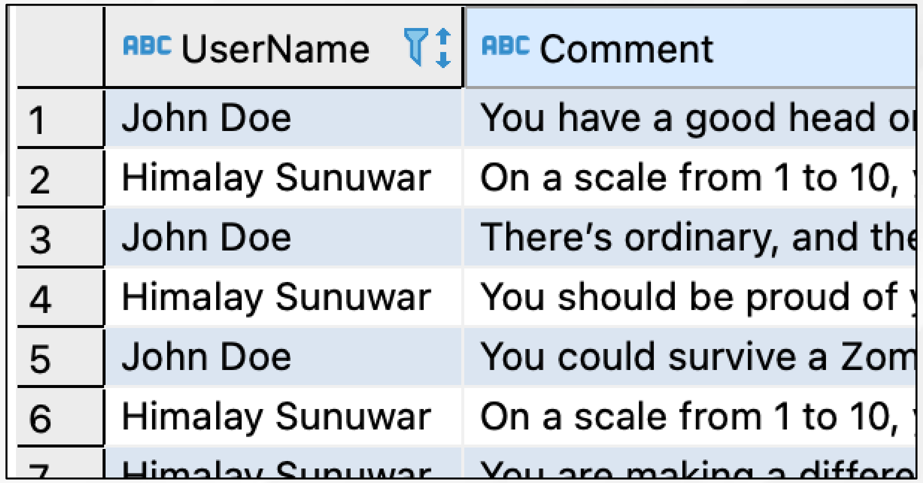
EF Core

Join





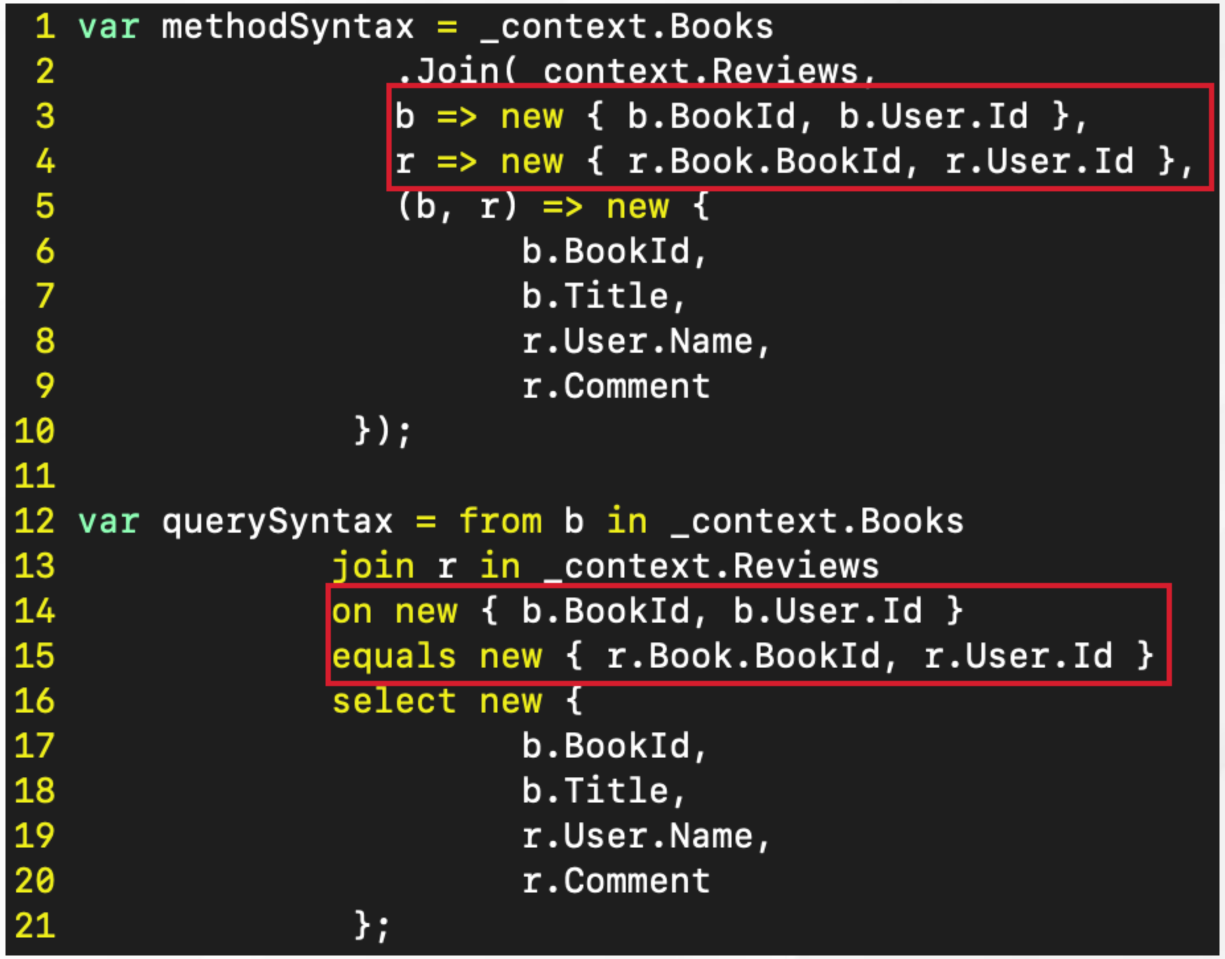
```
var methodSyntax = _context.Reviews
                    .Join(_context.Users, r => r.User.Id, u => u.Id, (r, u) => new {
                        UserName = u.Name,
                        Comment = r.Comment
                    });

var querySyntax = from r in _context.Reviews
                    join u in _context.Users on r.User.Id equals u.Id
                    select new {
                        UserName = u.Name,
                        Comment = r.Comment
                    };
```



	UserName	Comment
1	John Doe	You have a good head o
2	Himalay Sunuwar	On a scale from 1 to 10,
3	John Doe	There’s ordinary, and the
4	Himalay Sunuwar	You should be proud of
5	John Doe	You could survive a Zom
6	Himalay Sunuwar	On a scale from 1 to 10,
7	Himalay Sunuwar	You are making a differe

EF Core

Join with Multiple Columns



```
var methodSyntax = _context.Books
               .Join(_context.Reviews,
               b => new { b.BookId, b.User.Id },
               r => new { r.Book.BookId, r.User.Id },
               (b, r) => new {
                     b.BookId,
                     b.Title,
                     r.User.Name,
                     r.Comment
             });

var querySyntax = from b in _context.Books
            join r in _context.Reviews
            on new { b.BookId, b.User.Id }
            equals new { r.Book.BookId, r.User.Id }
            select new {
                     b.BookId,
                     b.Title,
                     r.User.Name,
                     r.Comment
             };
```

Thank You