1. **.NET Core Basics – 50 Q&A**
2. **C# and Language Features – 50 Q&A**
3. **ASP.NET Core – 70 Q&A**
4. **Entity Framework Core – 40 Q&A**
5. **Dependency Injection, Middleware & Configuration – 40 Q&A**
6. **Advanced Topics (API, Microservices, Testing, Security, Performance) – 50 Q&A**

.NET Core Basics — 50 Interview Questions and Answers

1. What is .NET Core?
.NET Core is an open-source, cross-platform framework developed by Microsoft for building modern, cloud-based, and high-performance applications.

2. What are the main features of .NET Core?
Cross-platform, open-source, modular, high-performance, CLI-based, side-by-side versioning, and cloud-optimized.

3. Which platforms does .NET Core support?
Windows, macOS, and Linux.

4. What are the differences between .NET Framework and .NET Core?
.NET Framework is Windows-only and monolithic; .NET Core is cross-platform, modular, and open source.

5. What is the latest version of .NET Core called?
Starting from .NET 5, Microsoft unified it under the name **.NET** (no "Core" suffix).

6. What is the CLR in .NET Core?
CLR (Common Language Runtime) is the virtual machine component that manages execution of .NET programs — memory, security, and exception handling.

7. What is CoreCLR?
CoreCLR is the cross-platform, modular runtime used in .NET Core applications.

8. What is CoreFX?
CoreFX is the set of foundational class libraries in .NET Core (collections, I/O, networking, etc.).

9. What is the .NET SDK?
The .NET SDK includes the compiler, CLI tools, and libraries required to build and run .NET applications.

10. How do you check the installed .NET Core SDK version?
Run:

```
dotnet --version
```

11. What is the use of the `dotnet` command-line interface?
It is used to create, build, run, and publish .NET applications.

12. How do you create a new .NET Core project?

```
dotnet new console -n MyApp
```

13. How do you run a .NET Core application?

```
dotnet run
```

14. What are Runtime Identifiers (RIDs)?
RIDs define target platforms for deployment (e.g., `win-x64`, `linux-x64`, `osx-arm64`).

15. What is self-contained deployment?
Deployment where the .NET runtime and libraries are included with the app.

16. What is framework-dependent deployment?
Deployment where the target machine must have the .NET runtime installed.

17. What is NuGet in .NET Core?
NuGet is the package manager for .NET, used to install and manage libraries.

18. How do you add a NuGet package to a .NET Core project?

```
dotnet add package <PackageName>
```

19. What are project files in .NET Core?
They are XML-based `.csproj` files that define dependencies and build configuration.

20. What is MSBuild?
MSBuild is the build system used by .NET projects to compile and package code.

21. What is Roslyn?
Roslyn is the open-source C# and VB.NET compiler that provides rich code analysis APIs.

22. What is a Target Framework Moniker (TFM)?
It identifies the version of the .NET platform the project targets (e.g., `net8.0`).

23. How can you target multiple frameworks in a single project?
By using the `<TargetFrameworks>` element in `.csproj`.

24. What are global tools in .NET Core?
They are command-line utilities installed globally using `dotnet tool install`.

25. What is the difference between SDK-style projects and old .NET projects?
SDK-style projects use simplified XML and auto-include common files.

26. What is Kestrel?
Kestrel is the default cross-platform web server used in ASP.NET Core.

27. What is the role of the Host in .NET Core applications?
The Host sets up configuration, dependency injection, and logging before running the app.

28. What are Environment Variables used for in .NET Core?
To configure environment-specific settings (e.g., `ASPNETCORE_ENVIRONMENT`).

29. What are the common environments in .NET Core?
Development, Staging, and Production.

30. How do you set the environment variable in Windows?

```
setx ASPNETCORE_ENVIRONMENT "Development"
```

31. What is the purpose of `appsettings.json`?
It stores configuration data such as connection strings and app settings.

32. How do you read configuration values in .NET Core?
Using `IConfiguration` from the dependency injection container.

33. What is logging in .NET Core?
Logging provides a standardized way to capture and store application events.

34. What logging providers are available by default?
Console, Debug, EventSource, EventLog, and Azure App Services.

35. How do you enable detailed error pages in Development mode?
By using the `DeveloperExceptionPage` middleware.

36. What is the Program.cs file for?
It's the application's entry point that builds and runs the Host.

37. What is the Startup.cs file for?
It defines the request pipeline and registers services.

38. What is Middleware in .NET Core?
Middleware is software that handles requests and responses in the HTTP pipeline.

39. What are the benefits of Middleware architecture?
Modularity, flexibility, and easy customization of request handling.

40. What is the `IHostBuilder` interface used for?
To configure and build a .NET Core host application.

41. What is the difference between WebHost and GenericHost?
WebHost is specific to web apps, while GenericHost supports all types of .NET Core apps.

42. What is dependency injection (DI)?
A design pattern that provides dependencies rather than creating them manually.

43. Is DI built-in in .NET Core?
Yes, it has a built-in IoC container.

44. What is a Service Lifetime?
It defines how long a service instance lives — Transient, Scoped, or Singleton.

45. What is the purpose of the `ConfigureServices` method?
To register services for dependency injection.

46. What is the purpose of the `Configure` method?
To define how the app responds to HTTP requests.

47. What is cross-platform development?
Developing an app that runs on multiple operating systems without modification.

48. How do you publish a .NET Core application?

```
dotnet publish -c Release
```

49. What is the difference between `dotnet build` and `dotnet publish`?
`build` compiles code; `publish` packages it for deployment.

50. What is the .NET Runtime Identifier (RID) used for in publishing?
It specifies which platform the published app will run on.

□ Part 2 — C# and Language Features (50 Interview Questions & Answers)

---

1. What is C#?
C# is a modern, object-oriented, and type-safe programming language developed by Microsoft for the .NET platform.

---

2. What are the main features of C#?
Object-oriented, type-safe, garbage-collected, modern syntax, cross-platform, and supports async programming.

---

3. What is the difference between `ref` and `out` parameters?
Both pass arguments by reference, but `out` parameters must be assigned before returning, while `ref` parameters must be initialized before being passed.

---

4. What is a nullable type in C#?
It allows value types to represent a null value, declared as `int? x = null;`.

---

5. What is the difference between value types and reference types?
Value types store data directly (stack), while reference types store a reference to the data (heap).

---

6. What are some examples of value types?
`int`, `double`, `bool`, `char`, `struct`, and `enum`.

---

7. What are some examples of reference types?
`class`, `interface`, `delegate`, `array`, and `string`.

---

8. What is boxing and unboxing?
Boxing converts a value type to an object; unboxing extracts it back to the original value type.

---

9. What are access modifiers in C#?
They define visibility — `public`, `private`, `protected`, `internal`, `protected internal`, `private protected`.

---

10. What is the difference between `const` and `readonly`?
`const` is compile-time constant, `readonly` can be assigned at runtime (constructor).

---

11. What are properties in C#?
Properties are class members that provide controlled access to private fields using `get` and `set`.

---

12. What is auto-implemented property?
A shorthand property with an implicit backing field:

```
public int Age { get; set; }
```

---

13. What is encapsulation?
The principle of hiding implementation details and exposing only necessary members.

---

14. What is inheritance?
The ability of one class to derive from another and reuse its members.

---

15. What is polymorphism?
The ability to perform a single action in different ways, often via virtual/override or interfaces.

---

16. What is abstraction?
Hiding complex implementation details and exposing only essential functionality.

17. What is an abstract class?
A class that cannot be instantiated and may contain abstract members to be implemented by derived classes.

---

18. What is an interface?
A contract that defines members without implementation.

---

19. What is the difference between abstract class and interface?
Abstract classes can have implementation and fields; interfaces cannot (until default interface methods in C# 8).

---

20. What is a sealed class?
A class that cannot be inherited.

---

21. What is a static class?
A class that cannot be instantiated and can only contain static members.

---

22. What is the `var` keyword used for?
Implicitly types a variable at compile-time based on the assigned value.

---

23. What is the `dynamic` keyword?
Defers type checking until runtime.

---

24. What is the difference between `var` and `dynamic`?
`var` is compile-time, `dynamic` is runtime.

---

25. What is a delegate?
A type that represents references to methods with a specific signature.

---

26. What is an event in C#?
A message broadcast mechanism that uses delegates to notify subscribers of changes.

---

27. What are lambda expressions?
Anonymous functions written using the `=>` syntax.
Example:

```
x => x * x;
```

---

28. What are expression-bodied members?
A concise way to define simple members:

```
public int Square(int x) => x * x;
```

---

29. What is LINQ?
Language Integrated Query — allows querying collections using SQL-like syntax in C#.

---

30. What are anonymous types?
Types created on the fly without defining a class:

```
var person = new { Name = "John", Age = 30 };
```

---

31. What are extension methods?
Static methods that extend existing types without modifying their source code.

---

32. What are generics in C#?
They enable type-safe data structures and methods without specifying the actual data type in advance.

---

33. What is the difference between `IEnumerable`, `IQueryable`, and `ICollection`?

- `IEnumerable`: in-memory, forward-only iteration.
- `IQueryable`: queryable against external data sources (like databases).
- `ICollection`: supports add/remove operations.

---

34. What are async and await keywords used for?
They enable asynchronous programming without blocking threads.

---

35. What is the difference between synchronous and asynchronous programming?
Synchronous blocks execution until completion; asynchronous allows concurrent execution.

---

36. What is a Task in C#?
Represents an asynchronous operation that may return a result.

---

37. What is the difference between `Task` and `ValueTask`?
`ValueTask` is optimized for scenarios where the result is often available synchronously.

---

38. What is the difference between `throw ex` and `throw`?
`throw ex` resets the stack trace; `throw` preserves the original exception details.

---

39. What are tuples in C#?
Data structures that group multiple values into a single object:

```
var tuple = (1, "John");
```

---

40. What is pattern matching?
A feature to test values against patterns, introduced in C# 7+ (e.g., `is`, `switch` expressions).

---

41. What is a record in C#?
An immutable reference type designed for value-based equality (C# 9+).

---

42. What is a struct in C#?
A value type that is stored on the stack and often used for lightweight objects.

---

43. What is the difference between `class` and `struct`?
`class` is a reference type (heap), `struct` is a value type (stack).

---

44. What are indexers in C#?
Allow objects to be accessed like arrays using `this[]` syntax.

---

45. What are partial classes?
Allow a class definition to be split across multiple files.

---

46. What is the `using` statement for?
Automatically disposes of objects implementing `IDisposable`.

---

47. What is garbage collection?
Automatic memory management that frees unused objects from memory.

---

48. How can you force garbage collection?

```
GC.Collect();
```

(Generally not recommended.)

---

49. What is reflection in C#?
The process of inspecting metadata about assemblies, types, and members at runtime.

---

50. What is the difference between early binding and late binding?
Early binding occurs at compile time; late binding happens at runtime (using reflection or `dynamic`).

□ Part 3 — ASP.NET Core (70 Interview Questions & Answers)

---

1. What is ASP.NET Core?
ASP.NET Core is a cross-platform, open-source web framework for building modern, cloud-based web applications and APIs.

---

2. What are the main features of ASP.NET Core?
Cross-platform support, modularity, dependency injection, unified MVC and Web API, built-in configuration and logging, high performance, and cloud integration.

---

3. What are the differences between ASP.NET and ASP.NET Core?
ASP.NET Core is modular, cross-platform, and open-source, while ASP.NET is Windows-only and based on the .NET Framework.

---

4. What web servers can host ASP.NET Core apps?
Kestrel, IIS, Apache, and Nginx.

---

5. What is Kestrel?
Kestrel is the default cross-platform, high-performance web server for ASP.NET Core.

---

6. What is the role of the `Program.cs` file?
It configures and builds the application host and defines the entry point.

---

7. What is the `Startup.cs` file used for?
It defines services (via `ConfigureServices`) and the HTTP request pipeline (via `Configure`).

---

8. What are Middleware components?
Middleware are components that process HTTP requests and responses in a pipeline.

---

9. What is the order of middleware execution?
Middleware runs in the order they're added in `Startup.Configure`.

---

10. What is `app.Use`, `app.Run`, and `app.Map` in middleware configuration?

- `Use`: Adds middleware that can call the next delegate.
- `Run`: Terminates the pipeline.
- `Map`: Branches the pipeline based on request path.

---

11. What is Routing in ASP.NET Core?
Routing maps incoming requests to corresponding endpoints (controllers, Razor pages, etc.).

---

12. What are the types of routing in ASP.NET Core?

- Conventional Routing
- Attribute Routing
- Endpoint Routing (default in newer versions)

---

13. What is Attribute Routing?
Defining routes directly on controller actions using attributes like `[Route("api/products")]`.

---

14. What is Endpoint Routing?
A unified routing system that executes routes and middleware together, introduced in ASP.NET Core 3.0.

---

15. What are Controllers?
Classes that handle HTTP requests and return responses.

---

16. What are Action Methods?
Public methods inside a controller that respond to HTTP requests.

---

17. What is a Model in MVC?
A class representing application data and business logic.

---

18. What is a View in MVC?
A UI template used to display data to users, often written in Razor syntax.

---

19. What is Razor?
A templating syntax that allows embedding C# code within HTML using `@`.

---

20. What is ViewModel?
A specialized model class used to send data from controllers to views.

---

21. What is a Layout View?
A shared view that defines common structure (like headers, menus, footers).

---

22. What are Tag Helpers?
Server-side components that generate HTML elements dynamically in Razor views.

---

23. What are View Components?
Reusable components that encapsulate both logic and UI rendering (like partial views with logic).

---

24. What are Partial Views?
Reusable fragments of HTML and Razor markup used across views.

---

25. What is the difference between ViewBag, ViewData, and TempData?

- `ViewBag`: dynamic object (lives during request).
- `ViewData`: dictionary object (lives during request).
- `TempData`: persists data across redirects.

---

26. What are Filters in ASP.NET Core?
Filters are used to run code before or after controller actions — for logging, validation, or authorization.

---

27. Types of Filters?

- Authorization Filters
- Resource Filters
- Action Filters
- Exception Filters
- Result Filters

---

28. What is Model Binding?
Automatically maps request data (query, form, body, route) to controller parameters or model objects.

---

29. What is Model Validation?
Verifies that model data meets specified rules (via data annotations or custom validation attributes).

---

30. How do you handle exceptions globally in ASP.NET Core?
Use `UseExceptionHandler`, `DeveloperExceptionPage`, or middleware.

---

31. What is the difference between `UseDeveloperExceptionPage` and `UseExceptionHandler`?
`UseDeveloperExceptionPage` shows detailed errors (development only); `UseExceptionHandler` provides a generic error page (production).

---

32. How do you enable HTTPS redirection?
Use middleware:

```
app.UseHttpsRedirection();
```

---

33. What is dependency injection (DI)?
A pattern that provides required dependencies instead of creating them manually.

---

34. What are the default service lifetimes in ASP.NET Core?

- **Singleton** – One instance for app lifetime
- **Scoped** – One per request
- **Transient** – New instance every time

---

35. How to register a service in DI?

```
services.AddSingleton<IMyService, MyService>();
```

---

36. What are configuration providers in ASP.NET Core?
They load configuration data from JSON files, environment variables, command-line args, etc.

---

37. What is `appsettings.json`?
A configuration file containing settings like connection strings, logging, and custom options.

---

38. How to read configuration values?
Inject `IConfiguration` and use:

```
_config["AppSettings:Key"]
```

---

39. How do you use strongly typed configuration?
Bind configuration sections to custom classes using `services.Configure<T>()`.

---

40. What is logging in ASP.NET Core?
A built-in abstraction for recording application events.

---

41. What logging providers are available?
Console, Debug, EventLog, Azure, Application Insights, Serilog, etc.

---

42. What is Entity Framework Core (EF Core)?
The ORM used to interact with databases in ASP.NET Core apps.

---

43. What are Razor Pages?
A simplified programming model for page-based applications introduced in ASP.NET Core 2.0.

---

44. What is SignalR?
A library for adding real-time web functionality (like chat apps) using WebSockets.

---

45. What is a Web API?
A framework for building HTTP-based APIs that return JSON or XML data.

---

46. What is the difference between MVC and Web API in ASP.NET Core?
They are unified — both built on the same controller base class.

---

47. What are HTTP verbs used in ASP.NET Core Web API?
GET, POST, PUT, DELETE, PATCH, OPTIONS.

---

48. What is IActionResult?
A return type for controller actions that can produce different HTTP responses.

---

49. What is FromBody, FromQuery, and FromRoute?
Attributes that specify where to bind parameters from (body, query string, or route).

---

50. What is content negotiation?
ASP.NET Core automatically selects the best response format (JSON/XML) based on request headers.

---

51. What are API Versioning strategies?

- URL versioning
- Query string versioning
- Header versioning

- Media type versioning

---

52. What is Swagger (OpenAPI)?
A tool for generating interactive documentation and testing APIs.

---

53. How do you enable Swagger in ASP.NET Core?
Install `Swashbuckle.AspNetCore` and use:

```
app.UseSwagger();
app.UseSwaggerUI();
```

---

54. What is CORS?
Cross-Origin Resource Sharing — allows web apps from different domains to access resources.

---

55. How do you enable CORS in ASP.NET Core?
Register with:

```
services.AddCors();
app.UseCors("MyPolicy");
```

---

56. What is Authentication?
The process of identifying who the user is.

---

57. What is Authorization?
Determines what actions a user is allowed to perform.

---

58. What are authentication schemes supported?
Cookies, JWT Bearer Tokens, OAuth2, OpenID Connect.

---

59. How do you secure an API using JWT?
Use `Microsoft.AspNetCore.Authentication.JwtBearer` and configure token validation parameters.

---

60. What is Identity in ASP.NET Core?
A framework for handling user registration, login, and role-based access control.

---

61. What are Razor Class Libraries (RCL)?
Reusable libraries that contain Razor UI components.

---

62. What is a Host in ASP.NET Core?
It sets up app startup, configuration, DI, and logging — either a Generic Host or Web Host.

---

63. What is the difference between WebHost and GenericHost?
`WebHost` is web-specific; `GenericHost` supports any type of .NET Core application.

---

64. What is the use of `IWebHostEnvironment`?
Provides information about the environment (Development, Staging, Production).

---

65. What is the purpose of `IApplicationBuilder`?
Used to configure the request pipeline and add middleware components.

---

66. What is static file middleware?
Serves files like images, CSS, and JavaScript from the `wwwroot` folder.

---

67. What is localization and globalization?
Localization adapts content for a specific culture; globalization makes apps culture-aware.

---

68. What is health check middleware?
Provides endpoints to monitor the health status of an application.

---

69. What are background services in ASP.NET Core?
Long-running services implemented using `IHostedService` or `BackgroundService`.

---

70. How do you host ASP.NET Core apps in IIS?
ASP.NET Core runs behind IIS using the ASP.NET Core Module, which forwards requests to Kestrel.

Part 4 — Entity Framework Core (40 Interview Questions & Answers)

---

1. What is Entity Framework Core (EF Core)?
EF Core is Microsoft's lightweight, extensible, cross-platform ORM (Object-Relational Mapper) for .NET.

---

2. What is ORM (Object-Relational Mapping)?
ORM maps database tables to .NET objects, allowing developers to interact with databases using C# instead of SQL.

---

3. What are the advantages of using EF Core?

- Reduces boilerplate SQL code
- Cross-platform
- Supports LINQ queries
- Change tracking
- Migration management
- Strongly-typed data access

---

4. What are the different approaches to using EF Core?

- Code-First
- Database-First
- Model-First (not common in EF Core)

---

5. What is Code-First approach?
The model (classes) is defined first in code, and the database schema is generated from it.

---

6. What is Database-First approach?
An existing database is used to generate models and DbContext via scaffolding.

---

7. How do you install EF Core in a .NET project?
Via NuGet:

```
dotnet add package Microsoft.EntityFrameworkCore.SqlServer
dotnet add package Microsoft.EntityFrameworkCore.Tools
```

---

8. What is `DbContext` in EF Core?
The main class responsible for interacting with the database; it manages entities, queries, and saving changes.

---

9. What is a DbSet?
A collection representing a table in the database.

```
public DbSet<Product> Products { get; set; }
```

---

10. How do you create a migration in EF Core?

```
dotnet ef migrations add InitialCreate
```

---

11. How do you apply migrations to the database?

```
dotnet ef database update
```

---

12. What is a migration?
A way to incrementally evolve the database schema as the model changes.

---

13. What is the difference between `Add-Migration` and `Update-Database`?

- `Add-Migration`: creates migration scripts.
- `Update-Database`: applies migrations to the actual database.

---

14. What is the role of `OnModelCreating` method in `DbContext`?
It is used to configure entity relationships, constraints, and table mappings via the Fluent API.

---

15. What are the types of relationships supported in EF Core?

- One-to-One
- One-to-Many
- Many-to-Many

---

16. How do you configure relationships in EF Core?
Using Fluent API or Data Annotations like `[ForeignKey]`, `[Required]`, `[Key]`.

---

17. What is Fluent API?
A way to configure entity mappings and relationships using method chaining instead of attributes.

---

18. What is Lazy Loading?
When related data is automatically loaded from the database when first accessed.

---

19. What is Eager Loading?
Loading related data immediately with the main entity using `.Include()`.

```
_context.Orders.Include(o => o.Customer).ToList();
```

---

20. What is Explicit Loading?
Manually loading related data using `.Entry()` methods when needed.

---

21. What is Change Tracking?
EF Core keeps track of entity states (Added, Modified, Deleted, Unchanged) to know what to save.

---

22. What are Entity States in EF Core?

- Added
- Modified
- Deleted
- Unchanged
- Detached

---

23. How does EF Core save changes to the database?
By calling:

```
context.SaveChanges();
```

---

24. What is a LINQ query?
A language-integrated syntax for querying data collections or EF entities.

```
var users = context.Users.Where(u => u.IsActive);
```

---

25. What is AsNoTracking()?
A query optimization that prevents EF Core from tracking entities in memory, improving performance for read-only operations.

---

26. What are raw SQL queries in EF Core?
Executing SQL directly using:

```
context.Products.FromSqlRaw("SELECT * FROM Products");
```

---

27. What is concurrency control in EF Core?
Ensures data integrity when multiple users update the same record concurrently, often using a `RowVersion` column.

---

28. What is a shadow property?
A property not defined in the entity class but tracked by EF Core in the model.

---

29. What are value conversions in EF Core?
Allow conversion of property values during read/write operations (e.g., storing enums as strings).

---

30. What are global query filters?
Filters automatically applied to all queries on a DbSet, such as soft deletes.

```
modelBuilder.Entity<User>().HasQueryFilter(u => !u.IsDeleted);
```

---

31. What is the difference between `Find()` and `FirstOrDefault()`?

- `Find()` searches by primary key and uses cache first.
- `FirstOrDefault()` executes a LINQ query against the database.

---

32. What is shadow state tracking?
EF Core tracks properties that exist only in the EF model, not in the CLR class.

---

33. What is the purpose of `DbContextOptions`?
Used to configure the database provider and connection string for the context.

---

34. How do you specify a connection string in EF Core?
In `appsettings.json` and then configure it in `Startup.cs`:

```
services.AddDbContext<AppDbContext>(options =>
    options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));
```

---

35. What are providers supported by EF Core?
SQL Server, PostgreSQL, MySQL, SQLite, Cosmos DB, InMemory, and others.

---

36. What is the In-Memory Database Provider?
A provider for testing purposes without using a real database.

---

37. What is owned entity type?
A type owned by another entity — it doesn't have its own primary key or table.

---

38. What is a composite key?
A key that consists of multiple columns defined using:

```
modelBuilder.Entity<OrderItem>()
    .HasKey(o => new { o.OrderId, o.ProductId });
```

---

39. What is database seeding in EF Core?
Automatically populating the database with initial data:

```
modelBuilder.Entity<Role>().HasData(new Role { Id = 1, Name = "Admin" });
```

---

40. How do you perform CRUD operations in EF Core?

```
// Create
context.Users.Add(user);
context.SaveChanges();

// Read
var user = context.Users.Find(id);

// Update
user.Name = "Updated";
context.SaveChanges();

// Delete
context.Users.Remove(user);
context.SaveChanges();
```

Part 5 — Dependency Injection, Middleware & Configuration (40 Interview Questions & Answers)

Dependency Injection (DI)

1. What is Dependency Injection (DI)?
A design pattern in which a class's dependencies are provided externally rather than being created inside the class.

---

2. Is DI built-in in .NET Core?
Yes — .NET Core includes a lightweight built-in IoC (Inversion of Control) container.

---

3. What are the benefits of Dependency Injection?
Loose coupling, easier testing, better maintainability, and improved scalability.

---

4. What are the different lifetimes for services in DI?

- **Singleton** — One instance for the entire application.
- **Scoped** — One instance per HTTP request.
- **Transient** — A new instance each time it's requested.

---

5. How do you register a service in DI?

```
services.AddSingleton<IMyService, MyService>();
services.AddScoped<IRepository, Repository>();
services.AddTransient<ILogger, ConsoleLogger>();
```

---

6. How do you inject a dependency into a controller or class?
Via **constructor injection**:

```
public class HomeController : Controller
{
    private readonly IMyService _service;
    public HomeController(IMyService service)
    {
        _service = service;
    }
}
```

---

7. What is the difference between AddSingleton, AddScoped, and AddTransient?

- **Singleton:** same instance for the app lifetime.
- **Scoped:** new instance per request.
- **Transient:** always a new instance.

---

8. Can a Singleton service use a Scoped service?
No — it causes a **captured scoped dependency** issue.

---

9. What are the different types of DI in .NET Core?

- Constructor Injection (most common)
- Method Injection
- Property Injection

---

10. What is a ServiceProvider?
The internal .NET Core container that resolves and manages service lifetimes.

---

11. How do you resolve dependencies manually?
Using:

```
var service = serviceProvider.GetService<IMyService>();
```

---

12. Can you use third-party DI containers in .NET Core?
Yes — Autofac, StructureMap, Ninject, and others can replace the default container.

---

13. What is the difference between service registration and resolution?

- **Registration:** adding the service to the container (`ConfigureServices`).
- **Resolution:** retrieving the service when the app runs.

---

14. How do you implement a factory pattern with DI?
Register a factory delegate or use `IServiceProvider` inside the factory class.

---

15. How do you inject configuration values into services?
Use `IOptions<T>` or `IConfiguration` interfaces.

☸Middleware

---

16. What is Middleware in ASP.NET Core?
Software components that process HTTP requests and responses in a pipeline.

---

17. How does Middleware work?
Each middleware component can perform operations before and/or after calling the next one.

---

18. What is the order of Middleware execution?
It executes in the order added in `Startup.Configure` — order matters.

---

19. What are some built-in middlewares in ASP.NET Core?
Static files, routing, authentication, authorization, CORS, exception handling, and HTTPS redirection.

---

20. How do you create custom Middleware?
Example:

```
public class MyMiddleware
{
    private readonly RequestDelegate _next;
    public MyMiddleware(RequestDelegate next) => _next = next;

    public async Task InvokeAsync(HttpContext context)
    {
        // Custom logic
        await _next(context);
    }
}
```

Then register:

```
app.UseMiddleware<MyMiddleware>();
```

---

21. What is the difference between `app.Use`, `app.Run`, and `app.Map`?

- **Use**: Adds middleware that can call next delegate.
- **Run**: Terminates the pipeline (no next call).
- **Map**: Branches the pipeline for specific request paths.

---

22. Can middleware depend on services registered in DI?
Yes — dependencies can be injected into the middleware constructor.

---

23. What is short-circuiting in middleware?
When a middleware stops the request pipeline by not calling the next middleware.

---

24. How do you handle exceptions in middleware?
Wrap the request delegate in a `try-catch` block and log or handle exceptions.

---

25. What is UseDeveloperExceptionPage()?
A built-in middleware that shows detailed error pages during development.

---

26. What is UseExceptionHandler()?
A production-safe middleware for global exception handling.

---

27. How can you conditionally add middleware based on environment?

```
if (env.IsDevelopment())
    app.UseDeveloperExceptionPage();
else
    app.UseExceptionHandler("/Home/Error");
```

---

28. What is the difference between synchronous and asynchronous middleware?
Async middleware returns a `Task` to support non-blocking I/O operations.

---

29. What is terminal middleware?
Middleware that ends the pipeline (e.g., `app.Run()`).

---

30. What is the static file middleware?
Serves files (HTML, CSS, JS, images) from the `wwwroot` folder:

```
app.UseStaticFiles();
```

□ Configuration

---

31. What is configuration in .NET Core?
The system that provides app settings through JSON files, environment variables, command-line arguments, etc.

---

32. What are common configuration sources?

- `appsettings.json`
- `appsettings.{Environment}.json`
- Environment variables
- Command-line arguments
- User secrets
- Azure Key Vault

---

33. How do you access configuration values in a class?
Inject `IConfiguration` and use:

```
var key = _config["AppSettings:Key"];
```

---

34. What is `IOptions<T>` pattern?
Provides strongly-typed access to configuration settings mapped to a class:

```
services.Configure<MySettings>(Configuration.GetSection("MySettings"));
```

---

35. What is the difference between `IOptions`, `IOptionsSnapshot`, and `IOptionsMonitor`?

- **IOptions:** Singleton, reads configuration once.
- **IOptionsSnapshot:** Scoped, reads per request (useful in web apps).
- **IOptionsMonitor:** Supports real-time config change notifications.

---

36. How do you handle different environments in .NET Core?
Use environment-specific config files:
`appsettings.Development.json`, `appsettings.Production.json`.

---

37. How do you set the environment variable?
On Windows:

```
setx ASPNETCORE_ENVIRONMENT "Development"
```

---

38. What is User Secrets in .NET Core?
A secure store for sensitive data during local development (`dotnet user-secrets`).

---

39. What is Azure Key Vault integration?
It securely stores and retrieves secrets and connection strings from Azure.

40. How do you reload configuration at runtime?
By enabling reload-on-change in `appsettings.json`:

```
.AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)
```

□ Part 6 — Advanced Topics (API, Microservices, Testing, Security, Performance)

(50 Interview Questions & Answers)

□ Web APIs & Advanced ASP.NET Core Concepts

---

1. What is REST API?
REST (Representational State Transfer) is an architectural style that uses standard HTTP methods (GET, POST, PUT, DELETE) for CRUD operations over resources.

---

2. What are the main principles of REST?
Statelessness, client-server separation, uniform interface, cacheability, layered system, and resource identification via URIs.

---

3. What is the difference between SOAP and REST?
SOAP is a protocol using XML for message exchange; REST is an architectural style using standard HTTP and multiple formats (JSON, XML).

---

4. What is HATEOAS?
Hypermedia As The Engine Of Application State — REST principle where responses include links to related actions.

---

5. What is content negotiation?
The mechanism that lets clients request specific response formats (e.g., JSON or XML) using `Accept` headers.

---

6. What are API versioning methods supported in ASP.NET Core?

- URL-based (e.g., `/api/v1/products`)

- Query string (e.g., `?v=2`)
- Header-based
- Media type-based

---

7. What is throttling or rate limiting?
Restricting the number of API calls a client can make in a given time to prevent abuse.

---

8. How can you implement rate limiting in .NET Core?
Using middleware or libraries like `AspNetCoreRateLimit`.

---

9. What is gRPC?
A high-performance, contract-based RPC framework using Protocol Buffers for serialization.

---

10. Difference between REST and gRPC?
REST uses JSON over HTTP; gRPC uses Protobuf over HTTP/2 and supports streaming.

---

11. What is GraphQL?
An alternative to REST where clients specify exactly what data they need in a single request.

---

12. What is Swagger (OpenAPI)?
A specification and toolset for documenting and testing APIs in an interactive UI.

---

13. How do you secure a Web API?
Using authentication (JWT, OAuth2) and authorization (policies, roles, claims).

---

14. What is Bearer Authentication?
Token-based authentication where each request includes an `Authorization: Bearer <token>` header.

---

15. What is CORS (Cross-Origin Resource Sharing)?
A security mechanism controlling how resources are shared across different origins (domains).

---

16. How do you enable CORS in ASP.NET Core?

```
services.AddCors(options =>
    options.AddPolicy("AllowAll", builder =>
builder.AllowAnyOrigin().AllowAnyHeader().AllowAnyMethod()));
app.UseCors("AllowAll");
```

---

17. What is ModelState in ASP.NET Core Web API?
Represents the state of model binding and validation — helps check if incoming data is valid.

---

18. How do you return consistent error responses from an API?
Use global exception handling or problem details (`ProblemDetails`) format for standardized error output.

---

19. What is API Gateway?
A single entry point that routes client requests to different backend microservices.

---

20. How can you implement an API Gateway in .NET Core?
Using frameworks like **Ocelot** or **YARP (Yet Another Reverse Proxy)**.

Microservices Architecture

---

21. What are Microservices?
A software architecture style where applications are broken into small, independent services communicating over APIs.

---

22. What are the advantages of microservices?
Scalability, flexibility, independent deployment, fault isolation, and better maintainability.

---

23. What are the challenges of microservices?
Complex communication, data consistency, deployment complexity, and distributed tracing.

---

24. How do microservices communicate?
Via HTTP REST APIs, gRPC, message queues (RabbitMQ, Kafka), or event-driven architectures.

---

25. What is Service Discovery?
The mechanism for automatically locating microservice instances (e.g., using Consul or Eureka).

---

26. What is Docker?
A platform for containerizing applications so they run consistently across environments.

---

27. What is Kubernetes (K8s)?
A container orchestration system for deploying, scaling, and managing containerized applications.

---

28. What is the difference between container and virtual machine?
Containers share the host OS kernel; VMs have their own OS and are heavier.

---

29. What is CQRS pattern?
Command Query Responsibility Segregation — separates read and write operations into different models for scalability.

---

30. What is the Saga pattern?
Manages distributed transactions across multiple microservices using a sequence of compensating actions.

□ Testing

---

31. What types of testing are common in .NET Core apps?
Unit testing, integration testing, functional testing, and performance testing.

---

32. Which testing frameworks are used in .NET Core?
xUnit, NUnit, and MSTest.

---

33. What is the difference between unit testing and integration testing?

- **Unit test:** tests individual components in isolation.
- **Integration test:** verifies how components work together.

---

34. How do you mock dependencies in .NET Core?
Using libraries like **Moq**, **NSubstitute**, or **FakeItEasy**.

---

35. How do you test controllers in ASP.NET Core?
Use `Microsoft.AspNetCore.Mvc.Testing` and mock dependencies with `IServiceProvider`.

---

36. How do you perform in-memory integration testing?
Use `WebApplicationFactory<TEntryPoint>` to spin up a test server.

---

37. What is Test-Driven Development (TDD)?
A software approach where tests are written before writing the actual code implementation.

---

38. What is Continuous Integration (CI)?
The practice of automatically building and testing code whenever changes are pushed to version control.

---

39. What tools are used for CI/CD with .NET Core?
Azure DevOps, GitHub Actions, Jenkins, TeamCity, and GitLab CI.

---

40. What is code coverage?
A measure of how much of the codebase is tested by automated tests.

□ Security & Performance

---

41. What are some common security features in ASP.NET Core?
Data protection, authentication/authorization, HTTPS enforcement, input validation, anti-forgery tokens, and secret management.

---

42. What is Cross-Site Request Forgery (CSRF)?
An attack where unauthorized commands are sent from a trusted user's browser.

---

43. How do you prevent CSRF in ASP.NET Core?
Use anti-forgery tokens:

```
@Html.AntiForgeryToken()
```

and `[ValidateAntiForgeryToken]` attribute on actions.

---

44. What is Cross-Site Scripting (XSS)?
Injecting malicious scripts into web pages — prevented by automatic encoding in Razor views.

---

45. How do you secure sensitive configuration data?
Using User Secrets, environment variables, or Azure Key Vault instead of hardcoding credentials.

---

46. What are Data Protection APIs?
APIs used for encryption, key management, and secure data storage (e.g., cookies and tokens).

---

47. How do you improve performance in ASP.NET Core apps?

- Caching
- Asynchronous programming
- Minimize database queries
- Use `IAsyncEnumerable`
- Compress responses
- Pool database connections

---

48. What is response caching?
Stores HTTP responses to reduce processing on repeated requests.

---

49. How do you enable response caching in ASP.NET Core?

```
services.AddResponseCaching();
app.UseResponseCaching();
[ResponseCache(Duration = 60)]
```

---

50. What are memory and distributed caching options in ASP.NET Core?

- **In-memory caching:** local to the server.
- **Distributed caching:** shared cache using Redis or SQL Server.

SQL Server Interview Questions & Answers — Structure

Part 1 – SQL Basics (30 Q&A)
Part 2 – Joins, Subqueries & Clauses (30 Q&A)
Part 3 – Stored Procedures, Functions & Triggers (25 Q&A)
Part 4 – Indexing, Performance & Query Optimization (25 Q&A)
Part 5 – Transactions, Locks & Isolation Levels (20 Q&A)
Part 6 – Advanced Topics: Views, CTEs, Window Functions, and Interview Scenarios (20 Q&A)

Part 1 — SQL Basics (30 Q&A)

---

1. What is SQL?

Answer: SQL (Structured Query Language) is a standard language used to communicate with and manage relational databases. It allows you to create, read, update, and delete data.

---

2. What are the different types of SQL commands?

Answer:

- **DDL (Data Definition Language):** CREATE, ALTER, DROP
- **DML (Data Manipulation Language):** SELECT, INSERT, UPDATE, DELETE
- **DCL (Data Control Language):** GRANT, REVOKE
- **TCL (Transaction Control Language):** COMMIT, ROLLBACK, SAVEPOINT

---

3. What is a primary key?

Answer: A primary key uniquely identifies each record in a table. It cannot contain NULL values.

---

4. What is a foreign key?

Answer: A foreign key in one table references the primary key in another table, enforcing referential integrity.

---

5. What is the difference between CHAR and VARCHAR?

Answer:

- **CHAR(n):** Fixed-length string, padded with spaces if necessary.
- **VARCHAR(n):** Variable-length string, stores only the actual length.

---

6. What is the difference between DELETE and TRUNCATE?

Answer:

- **DELETE:** Removes rows based on a condition, can be rolled back, fires triggers.
- **TRUNCATE:** Removes all rows, cannot be rolled back in some databases, faster, doesn't fire triggers.

---

7. What is the difference between WHERE and HAVING?

Answer:

- **WHERE:** Filters rows **before grouping**.
- **HAVING:** Filters groups **after aggregation**.

---

8. What is the difference between INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN?

Answer:

- **INNER JOIN:** Returns matching rows from both tables.
- **LEFT JOIN:** Returns all rows from left table + matching rows from right table.
- **RIGHT JOIN:** Returns all rows from right table + matching rows from left table.
- **FULL OUTER JOIN:** Returns all rows from both tables; NULL where no match exists.

---

9. What is a UNION and UNION ALL?

Answer:

- **UNION:** Combines result sets and removes duplicates.
- **UNION ALL:** Combines result sets and **keeps duplicates**.

---

10. What is a subquery?

Answer: A query nested inside another query (can be in SELECT, FROM, WHERE).

Example:

```
SELECT * FROM Employees WHERE DepartmentID = (SELECT DepartmentID FROM Departments WHERE
Name='IT');
```

---

11. What is a view?

Answer: A virtual table created using a SELECT query. It doesn't store data physically but presents data from one or more tables.

---

12. What is an index?

Answer: An index improves query performance by allowing faster data retrieval. Types include **clustered** and **non-clustered**.

---

13. What is the difference between clustered and non-clustered index?

Answer:

- **Clustered index:** Sorts and stores data rows physically (only one per table).
- **Non-clustered index:** Contains pointers to data rows (multiple per table).

---

14. What is a stored procedure?

Answer: A precompiled SQL code block stored in the database that can be executed multiple times.

---

15. What is the difference between a stored procedure and a function?

Answer:

- **Stored procedure:** Performs operations, may return multiple results, can modify data.
- **Function:** Returns a single value or table, cannot perform modifications to database data in some cases.

---

16. What is a trigger?

Answer: A trigger is a special procedure that automatically executes in response to **INSERT, UPDATE, or DELETE** operations on a table.

---

17. What is normalization?

Answer: Process of organizing data to reduce redundancy and improve data integrity. Common forms: 1NF, 2NF, 3NF, BCNF.

---

18. What is denormalization?

Answer: Combining tables or adding redundant data to improve query performance at the cost of redundancy.

---

19. What is a primary difference between DELETE, TRUNCATE, and DROP?

Answer:

- **DELETE:** Removes rows, can use WHERE.
- **TRUNCATE:** Removes all rows, cannot filter.
- **DROP:** Deletes entire table or database structure.

---

20. What is a transaction in SQL?

Answer: A sequence of one or more SQL operations treated as a single unit. Must be **atomic** — either fully succeeds or fails.

---

21. What are ACID properties?

Answer: Properties of a transaction:

- **Atomicity:** All or nothing
- **Consistency:** Database remains consistent
- **Isolation:** Transactions don't interfere
- **Durability:** Changes persist after commit

---

22. What is a foreign key constraint?

Answer: Ensures referential integrity by making sure values in a column match the primary key in another table.

---

23. What are aggregate functions?

Answer: Functions that perform calculations on multiple rows: `SUM()`, `AVG()`, `COUNT()`, `MIN()`, `MAX()`.

---

24. What are scalar functions?

Answer: Functions that return a single value per row, e.g., `LEN()`, `GETDATE()`, `ROUND()`.

---

25. What is a cursor in SQL?

Answer: A database object used to **iterate row by row** over a result set. Often slower than set-based operations.

---

26. What is the difference between local and global temporary tables?

Answer:

- **Local (#TempTable):** Visible only to the session that created it.
- **Global (##TempTable):** Visible to all sessions until the last session disconnects.

---

27. What is the difference between UNION and JOIN?

Answer:

- **UNION:** Combines rows **vertically** (stack results).
- **JOIN:** Combines rows **horizontally** (match columns from tables).

---

28. What is the difference between RANK(), DENSE_RANK(), and ROW_NUMBER()?

Answer:

- **ROW_NUMBER():** Sequential numbering, no ties.
- **RANK():** Same rank for ties, skips numbers.
- **DENSE_RANK():** Same rank for ties, no skipped numbers.

---

29. What is the difference between NVARCHAR and VARCHAR?

Answer:

- **VARCHAR:** Stores ASCII characters.
- **NVARCHAR:** Stores Unicode characters (supports multiple languages), uses more storage.

---

30. What is the difference between CROSS JOIN and INNER JOIN?

Answer:

- **CROSS JOIN:** Cartesian product of two tables.
- **INNER JOIN:** Returns only matching rows between two tables.

Part 2 — Joins, Subqueries & Clauses (30 Q&A)

---

1. What is a JOIN in SQL?

Answer: A JOIN combines rows from two or more tables based on a related column between them.

---

2. What are the types of JOINs?

Answer:

- INNER JOIN
- LEFT JOIN (LEFT OUTER JOIN)
- RIGHT JOIN (RIGHT OUTER JOIN)
- FULL OUTER JOIN
- CROSS JOIN
- SELF JOIN

---

3. What is an INNER JOIN?

Answer: Returns only the rows that have matching values in both tables.

```
SELECT e.Name, d.DepartmentName
FROM Employees e
INNER JOIN Departments d ON e.DepartmentID = d.DepartmentID;
```

---

4. What is a LEFT JOIN?

Answer: Returns all rows from the left table and matching rows from the right table. Non-matching rows in the right table appear as NULL.

---

5. What is a RIGHT JOIN?

Answer: Returns all rows from the right table and matching rows from the left table. Non-matching rows in the left table appear as NULL.

---

6. What is a FULL OUTER JOIN?

Answer: Returns all rows when there is a match in one of the tables. Non-matching rows from either table appear as NULL.

---

7. What is a CROSS JOIN?

Answer: Returns the Cartesian product of two tables (all combinations of rows).

---

8. What is a SELF JOIN?

Answer: A JOIN of a table with itself to compare rows within the same table.

---

9. What is a subquery (nested query)?

Answer: A query inside another query, used to fetch intermediate results.

Example:

```
SELECT Name FROM Employees WHERE DepartmentID = (SELECT DepartmentID FROM Departments
WHERE Name='IT');
```

---

10. What are correlated subqueries?

Answer: Subqueries that refer to a column from the outer query and execute once per row of the outer query.

---

11. What is the difference between correlated and non-correlated subqueries?

Answer:

- **Non-correlated:** Independent, executes once.
- **Correlated:** Depends on outer query, executes repeatedly.

---

12. What is the difference between WHERE and HAVING clauses?

Answer:

- **WHERE:** Filters rows **before aggregation**.
- **HAVING:** Filters groups **after aggregation**.

---

13. What is the difference between DISTINCT and GROUP BY?

Answer:

- **DISTINCT:** Eliminates duplicate rows.
- **GROUP BY:** Aggregates data and can be used with aggregate functions.

---

14. What is the difference between UNION and UNION ALL?

Answer:

- **UNION:** Removes duplicate rows.
- **UNION ALL:** Includes all duplicates.

---

15. What is the difference between EXISTS and IN?

Answer:

- **EXISTS:** Returns TRUE if subquery returns any row; often more efficient for correlated subqueries.
- **IN:** Checks if a value matches any value in a list or subquery.

---

16. What is a CASE statement in SQL?

Answer: Conditional expression used in SELECT, WHERE, or ORDER BY.

Example:

```
SELECT Name,
       CASE WHEN Salary > 50000 THEN 'High' ELSE 'Low' END AS SalaryLevel
FROM Employees;
```

---

17. What are aggregate functions?

Answer: Functions that summarize data: `SUM()`, `AVG()`, `COUNT()`, `MIN()`, `MAX()`.

---

18. What are scalar functions?

Answer: Return a single value per row: `LEN()`, `ROUND()`, `GETDATE()`.

---

19. What is a derived table?

Answer: A subquery in the FROM clause treated as a temporary table.

Example:

```
SELECT d.DepartmentName, e.EmployeeCount
FROM (SELECT DepartmentID, COUNT(*) AS EmployeeCount FROM Employees GROUP BY
DepartmentID) e
JOIN Departments d ON e.DepartmentID = d.DepartmentID;
```

---

20. What is a common table expression (CTE)?

Answer: A temporary named result set that can be referenced within a SELECT, INSERT, UPDATE, or DELETE statement.

Example:

```
WITH EmployeeCTE AS (
    SELECT Name, Salary FROM Employees WHERE Salary > 50000
)
SELECT * FROM EmployeeCTE;
```

---

21. What is the difference between CTE and derived table?

Answer:

- **CTE:** Can be referenced multiple times, supports recursion.
- **Derived table:** Exists only in the query it is defined in.

---

22. What is the difference between RANK(), DENSE_RANK(), and ROW_NUMBER()?

Answer:

- **ROW_NUMBER():** Unique sequential numbers.
- **RANK():** Tied values get the same rank, skips numbers.
- **DENSE_RANK():** Tied values get the same rank, no gaps.

---

23. What is the difference between inner join and subquery?

Answer:

- **INNER JOIN:** Combines tables horizontally based on a condition.
- **Subquery:** Runs a query inside another query; can return one or multiple values.

---

24. What is a window function?

Answer: Functions that perform calculations across a set of rows related to the current row. Examples: `ROW_NUMBER() OVER()`, `SUM() OVER(PARTITION BY ...)`.

---

25. What is the difference between ROW_NUMBER() and IDENTITY()?

Answer:

- **ROW_NUMBER():** Generated at query runtime, resets per query/partition.
- **IDENTITY():** Column property, auto-incremented during inserts.

---

26. What is the difference between TOP and LIMIT in SQL Server?

Answer:

- **TOP(n):** SQL Server syntax to limit rows returned.
- **LIMIT n:** MySQL/PostgreSQL syntax to limit rows.

---

27. What is the difference between inner join and left join?

Answer:

- **INNER JOIN:** Returns only matching rows.
- **LEFT JOIN:** Returns all left table rows + matching rows, NULL if no match.

---

28. What is a pivot table in SQL Server?

Answer: Converts row data into columnar format using `PIVOT`.

Example:

```
SELECT *
FROM (SELECT Year, Product, Sales FROM SalesData) AS SourceTable
PIVOT (SUM(Sales) FOR Year IN ([2023], [2024])) AS PivotTable;
```

---

29. What is the difference between UNPIVOT and PIVOT?

Answer:

- **PIVOT:** Rows → Columns.
- **UNPIVOT:** Columns → Rows.

30. What is the difference between correlated subquery and JOIN?

Answer:

- **Correlated subquery:** Runs once per outer row; may be slower.
- **JOIN:** Combines tables in a single query efficiently; preferred for large datasets.

Part 3 — Stored Procedures, Functions & Triggers (25 Q&A)

---

1. What is a stored procedure?

Answer: A stored procedure is a precompiled collection of SQL statements stored in the database. It can accept parameters, perform operations, and return results.

Example:

```
CREATE PROCEDURE GetEmployeesByDept
    @DeptID INT
AS
BEGIN
    SELECT * FROM Employees WHERE DepartmentID = @DeptID;
END;
```

---

2. What are the advantages of stored procedures?

Answer:

- Reduces network traffic
- Promotes code reusability
- Improves security (can restrict direct table access)
- Enhances performance (precompiled)

---

3. What is a function in SQL Server?

Answer: A function is a database object that returns a value (scalar or table) and can be used in SQL statements.

Types:

- Scalar functions → return single value
- Table-valued functions → return a table

---

4. What is the difference between a stored procedure and a function?

Answer:

Feature	Stored Procedure	Function
Returns value	Optional	Must return a value (scalar or table)
Can modify data	Yes	Usually no (scalar functions cannot modify data)
Can be used in SELECT	No	Yes
Error handling	TRY/CATCH	Limited

---

5. What is a trigger in SQL Server?

Answer: A trigger is a special type of stored procedure that automatically executes in response to **INSERT, UPDATE, or DELETE** operations on a table.

---

6. What are the types of triggers?

Answer:

- **AFTER (FOR) Trigger:** Executes after the operation.
- **INSTEAD OF Trigger:** Executes instead of the operation.
- **DDL Triggers:** Fire on DDL events (CREATE, ALTER, DROP).
- **Logon Triggers:** Fire when a user logs on to the database.

---

7. What is the difference between AFTER and INSTEAD OF triggers?

Answer:

- **AFTER:** Executes **after** the DML operation.
- **INSTEAD OF:** Executes **instead of** the DML operation.

---

8. Can a trigger call a stored procedure?

Answer: Yes, triggers can call stored procedures to perform complex operations.

---

9. Can a stored procedure call another stored procedure?

Answer: Yes, stored procedures can call other procedures.

---

10. What is a recursive stored procedure?

Answer: A stored procedure that calls itself directly or indirectly. Must include a termination condition to avoid infinite recursion.

---

11. What is a table-valued function (TVF)?

Answer: A function that returns a table.

Example:

```
CREATE FUNCTION GetEmployeesByDept(@DeptID INT)
RETURNS TABLE
AS
RETURN
(
    SELECT * FROM Employees WHERE DepartmentID = @DeptID
);
```

---

12. What is a scalar function?

Answer: A function that returns a single value (int, varchar, date, etc.).

Example:

```
CREATE FUNCTION GetFullName(@FirstName VARCHAR(50), @LastName VARCHAR(50))
RETURNS VARCHAR(101)
AS
BEGIN
    RETURN @FirstName + ' ' + @LastName
END;
```

---

13. What is the difference between user-defined functions and system functions?

Answer:

- **User-defined functions:** Created by users for custom logic.
- **System functions:** Predefined by SQL Server (e.g., GETDATE(), LEN()).

---

14. Can a trigger modify the table that fired it?

Answer:

- **AFTER trigger:** Can modify other tables but usually not recommended to update the same table.
- **INSTEAD OF trigger:** Can modify the same table.

---

15. What is the difference between TRIGGER and STORED PROCEDURE?

Answer:

Feature	Trigger	Stored Procedure
Execution	Automatic	Manual/Called
Event-driven	Yes	No
Can modify table	Sometimes	Yes

16. What is the difference between local and global variables in stored procedures?

Answer:

- **Local variables:** Defined within a procedure, accessible only there.
- **Global variables:** Defined at session level (e.g., `@@ROWCOUNT`).

---

17. How do you handle errors in stored procedures?

Answer: Using `TRY...CATCH` blocks:

```
BEGIN TRY
    -- SQL statements
END TRY
BEGIN CATCH
    SELECT ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
```

---

18. Can a trigger return a value?

Answer: No, triggers cannot return values to the calling statement.

---

19. Can a trigger call another trigger?

Answer: Triggers cannot directly call another trigger, but a DML operation inside a trigger may fire another trigger.

---

20. What is the difference between deterministic and non-deterministic functions?

Answer:

- **Deterministic:** Always returns the same result for the same input (e.g., `ABS()`).
- **Non-deterministic:** Result may vary (e.g., `GETDATE()`, `NEWID()`).

---

21. Can you use transactions inside stored procedures?

Answer: Yes, you can use `BEGIN TRANSACTION`, `COMMIT`, and `ROLLBACK` within a stored procedure.

---

22. What is an inline table-valued function?

Answer: A TVF that has a single `SELECT` statement without a BEGIN/END block.

Example:

```
CREATE FUNCTION GetDeptEmployees(@DeptID INT)
RETURNS TABLE
AS
RETURN (SELECT * FROM Employees WHERE DepartmentID = @DeptID);
```

---

23. What is a multi-statement table-valued function?

Answer: A TVF that can have multiple statements, must declare a table variable to return.

Example:

```
CREATE FUNCTION GetHighSalaryEmployees()
RETURNS @EmpTable TABLE (ID INT, Name VARCHAR(50))
AS
BEGIN
    INSERT INTO @EmpTable
    SELECT EmployeeID, Name FROM Employees WHERE Salary > 50000;
    RETURN;
END;
```

---

24. Can a function call a stored procedure?

Answer: No, functions cannot call stored procedures in SQL Server.

---

25. Can a stored procedure return multiple result sets?

Answer: Yes, a stored procedure can return multiple SELECT statements, each producing a separate result set.

Part 4 — Indexing, Performance & Query Optimization (25 Q&A)

---

1. What is an index in SQL Server?

Answer: An index is a database object that improves the speed of data retrieval. It works like a table of contents in a book.

---

2. What are the types of indexes in SQL Server?

Answer:

- **Clustered Index** – Sorts and stores data physically. Only one per table.
- **Non-Clustered Index** – Contains pointers to data rows; multiple per table.
- **Unique Index** – Ensures uniqueness of column values.
- **Composite Index** – Covers multiple columns.
- **Full-Text Index** – Optimized for text search.

---

3. What is the difference between clustered and non-clustered index?

Answer:

Feature	Clustered	Non-Clustered
Physical storage	Data is sorted	Data not sorted, stores pointers
Number per table	1	Many
Retrieval	Faster	Slower than clustered

4. What is a covering index?

Answer: An index that includes all columns needed for a query, eliminating the need to access the base table.

---

5. What is a filtered index?

Answer: A non-clustered index with a WHERE clause that indexes only a subset of rows, improving query performance for selective queries.

---

6. What is index fragmentation?

Answer: Occurs when the logical order of pages does not match the physical order, leading to slower performance.

---

7. How can you fix index fragmentation?

Answer:

- **REBUILD INDEX** – Drops and recreates the index.
- **REORGANIZE INDEX** – Physically reorganizes leaf pages without dropping.

---

8. What is a statistics object in SQL Server?

Answer: Statistics contain information about data distribution in a column or index, helping the query optimizer choose the best execution plan.

---

9. What is the difference between a clustered index scan and a seek?

Answer:

- **Index Seek:** Efficient, searches only matching rows.
- **Index Scan:** Less efficient, reads entire index/table.

---

10. What is a query execution plan?

Answer: A visual or textual representation of how SQL Server executes a query, showing the operations, costs, and order.

---

11. How can you view a query execution plan?

Answer:

- Use `SET SHOWPLAN_ALL ON`
- Use SQL Server Management Studio (SSMS) → Display Estimated Execution Plan or Include Actual Execution Plan.

---

12. What is the difference between a simple query and a complex query optimization?

Answer:

- **Simple query:** Uses one table, minimal joins, usually optimized automatically.
- **Complex query:** Multiple joins, subqueries, aggregation; may require indexing, hints, or query rewriting.

---

13. What is a covering index, and how does it help query performance?

Answer: A covering index contains all columns a query needs, avoiding table lookups, thus improving performance.

---

14. What is parameter sniffing?

Answer: SQL Server caches an execution plan based on initial parameter values, which may be inefficient for subsequent different parameters.

---

15. How can you avoid parameter sniffing issues?

Answer:

- Use `OPTION (RECOMPILE)`
- Use local variables inside stored procedures
- Use query hints like `OPTIMIZE FOR UNKNOWN`

---

16. What is the difference between UNION and UNION ALL regarding performance?

Answer:

- **UNION:** Removes duplicates → may be slower.
- **UNION ALL:** Includes all rows → faster since no sorting/deduplication occurs.

---

17. What is the difference between temp tables and table variables regarding performance?

Answer:

- **Temp Tables (#temp):** Stored in tempdb, support indexes, statistics; better for large datasets.
- **Table Variables (@table):** Limited statistics, stored in memory, better for small datasets.

---

18. What is a CTE and its impact on performance?

Answer:

- **CTE:** Temporary named result set in a query.
- **Performance:** Useful for readability and recursion; may not improve performance alone.

---

19. How do you optimize joins for large tables?

Answer:

- Use proper indexes on join columns
- Avoid functions in join conditions
- Filter rows before joining
- Use appropriate join type (INNER, LEFT)

---

20. What are query hints in SQL Server?

Answer: Directives to the query optimizer to enforce specific behavior, e.g., `FORCESEEK`, `NOLOCK`, `OPTION (RECOMPILE)`.

---

21. What is the difference between NOLOCK and READCOMMITTED?

Answer:

- **NOLOCK:** Allows dirty reads, does not block.
- **READCOMMITTED:** Default isolation, prevents reading uncommitted data.

---

22. What is blocking and deadlock in SQL Server?

Answer:

- **Blocking:** One transaction waits for another to release locks.
- **Deadlock:** Two or more transactions block each other in a cycle; SQL Server kills one to resolve.

---

23. How do you detect deadlocks?

Answer:

- Use SQL Server Profiler / Extended Events
- Use `sys.dm_exec_requests` and `sys.dm_tran_locks`

---

24. How can you improve query performance in SQL Server?

Answer:

- Indexing key columns
- Using joins efficiently
- Avoid SELECT *
- Use WHERE clauses to filter early

- Analyze execution plans
- Consider temp tables, table variables, or CTEs appropriately

---

25. What is the difference between SARGable and non-SARGable queries?

Answer:

- **SARGable (Search ARGument able):** Can use indexes efficiently, e.g., `WHERE Column = 10`.
- **Non-SARGable:** Cannot use indexes efficiently, e.g., `WHERE YEAR(Column) = 2023`.

Part 5 — Advanced SQL Queries, Views & Transactions (25 Q&A)

---

1. What is a view in SQL Server?

Answer: A view is a virtual table representing the result of a stored SELECT query. It does not store data itself but fetches data from underlying tables.

---

2. What are the advantages of using views?

Answer:

- Simplifies complex queries
- Enhances security by restricting column/table access
- Provides abstraction and reusability
- Can be indexed (indexed views) for performance

---

3. What is the difference between a view and a table?

Answer:

Feature	View	Table
Storage	No physical storage (except indexed view)	Stores data physically
Data modification	Limited (depends on view)	Fully modifiable
Definition	SELECT statement	Physical structure

4. What is an indexed view?

Answer: A view with a unique clustered index applied to it. Data is physically stored and can improve performance for frequent complex queries.

---

5. What are the restrictions of indexed views?

Answer:

- Must be schema-bound (`WITH SCHEMABINDING`)
- Cannot include `UNION`, `OUTER JOIN`, `TOP`, `DISTINCT`
- Must be deterministic

---

6. What is a transaction in SQL Server?

Answer: A transaction is a sequence of SQL statements executed as a single unit of work. It ensures ACID properties: Atomicity, Consistency, Isolation, Durability.

---

7. What are the types of transactions?

Answer:

- **Implicit transaction:** Automatically started by SQL Server (rarely used)
- **Explicit transaction:** Defined by `BEGIN TRANSACTION ... COMMIT/ROLLBACK`
- **Autocommit transaction:** Default mode; each statement is a transaction

---

8. What is the difference between COMMIT and ROLLBACK?

Answer:

- **COMMIT:** Saves all changes permanently
- **ROLLBACK:** Undoes all changes within the transaction

---

9. What are transaction isolation levels?

Answer:

- **READ UNCOMMITTED** – Allows dirty reads
- **READ COMMITTED** – Default, no dirty reads
- **REPEATABLE READ** – Prevents non-repeatable reads
- **SERIALIZABLE** – Prevents phantom reads, strictest
- **SNAPSHOT** – Uses row versioning, no blocking

---

10. What is a phantom read?

Answer: Occurs when a transaction reads a set of rows twice and sees new rows inserted by another transaction in between.

---

11. What is the difference between a temporary table and a view?

Answer:

- **Temporary table:** Stores physical data in `tempdb`, can be modified
- **View:** Virtual table, no storage, represents a query result

---

12. How do you create a view?

Answer:

```
CREATE VIEW EmployeeView AS
SELECT Name, DepartmentID, Salary
FROM Employees
WHERE Salary > 50000;
```

---

13. Can a view be updated?

Answer: Yes, if it meets these criteria:

- No joins
- No aggregates
- No DISTINCT or GROUP BY
- Includes all NOT NULL columns without default values

---

14. What is a materialized view?

Answer: SQL Server calls it an **indexed view**. Data is physically stored and refreshed automatically when base tables change.

---

15. What is the difference between a correlated subquery and a derived table?

Answer:

- **Correlated subquery:** Depends on outer query, executes per row
- **Derived table:** Temporary table in FROM clause, executes once

---

16. What is a CTE and its benefits over subqueries?

Answer:

- **CTE (Common Table Expression):** Temporary named result set
- **Benefits:** Improves readability, supports recursion, can be referenced multiple times

---

17. What is a recursive CTE?

Answer: A CTE that references itself to perform hierarchical or iterative queries.

Example:

```
WITH EmployeeCTE AS (
    SELECT EmployeeID, ManagerID, Name FROM Employees WHERE ManagerID IS NULL
    UNION ALL
    SELECT e.EmployeeID, e.ManagerID, e.Name
    FROM Employees e
    INNER JOIN EmployeeCTE cte ON e.ManagerID = cte.EmployeeID
)
SELECT * FROM EmployeeCTE;
```

---

18. What is the difference between DELETE, TRUNCATE, and DROP?

Answer:

Command	Deletes Data	Resets Identity	Logs Transaction	Can be Rolled Back
DELETE	Yes	No	Logged	Yes
TRUNCATE	Yes	Yes	Minimal	Yes
DROP	No (removes table)	N/A	N/A	No

19. What is a transaction savepoint?

Answer: A savepoint allows partial rollback within a transaction.

```
BEGIN TRANSACTION;
INSERT INTO Employees VALUES(...);
SAVE TRANSACTION Save1;
UPDATE Employees SET Salary=60000 WHERE ID=1;
ROLLBACK TRANSACTION Save1; -- only rolls back update
COMMIT TRANSACTION;
```

---

20. What is the difference between implicit and explicit transactions?

Answer:

- **Implicit:** SQL Server auto-begins transaction; each statement commits automatically
- **Explicit:** User defines BEGIN TRANSACTION, COMMIT, and ROLLBACK

---

21. What is the difference between a simple view and a complex view?

Answer:

- **Simple view:** Selects from a single table, no aggregates
- **Complex view:** May include joins, aggregates, subqueries, unions

---

22. What is the difference between a snapshot isolation and read committed?

Answer:

- **Snapshot isolation:** Uses row versioning, no blocking, consistent reads
- **Read committed:** Default, may block reads while writes occur

---

23. How do you improve performance of complex queries?

Answer:

- Use appropriate indexes
- Avoid SELECT *
- Filter rows early (WHERE clause)
- Break queries into smaller parts using temp tables or CTEs
- Review execution plans

---

24. What is a key difference between a temp table and table variable?

Answer:

- **Temp table (#Temp):** Physical storage, statistics available, better for large datasets
- **Table variable (@Table):** Limited statistics, better for small datasets, generally faster in memory

---

25. Can transactions be nested in SQL Server?

Answer: Yes, SQL Server supports nested transactions using `BEGIN TRANSACTION`, but only the outermost `COMMIT` makes changes permanent. Inner `ROLLBACK` rolls back the entire transaction.

Part 6 — Security, Backup & Maintenance (25 Q&A)

---

1. What is authentication in SQL Server?

Answer: Authentication is the process of verifying the identity of a user or login. SQL Server supports:

- **Windows Authentication** – Uses Windows credentials
- **SQL Server Authentication** – Uses username/password stored in SQL Server

---

2. What is authorization in SQL Server?

Answer: Authorization is the process of granting or denying access to database objects after authentication. Controlled via roles and permissions.

---

3. What are the types of SQL Server roles?

Answer:

- **Server Roles:** Sysadmin, Securityadmin, Serveradmin, etc.
- **Database Roles:** db_owner, db_datareader, db_datawriter, db_securityadmin, etc.

---

4. How do you grant permissions in SQL Server?

Answer: Using the `GRANT` statement:

```
GRANT SELECT, INSERT ON Employees TO John;
```

---

5. How do you revoke permissions?

Answer: Using the `REVOKE` statement:

```
REVOKE SELECT, INSERT ON Employees FROM John;
```

---

6. What is a login vs a user in SQL Server?

Answer:

- **Login:** Access to the SQL Server instance
- **User:** Access to a specific database (mapped to a login)

---

7. What is SQL injection?

Answer: SQL injection is a security vulnerability where an attacker can inject malicious SQL statements to manipulate the database.

Prevention: Use parameterized queries, stored procedures, and input validation.

---

8. What is encryption in SQL Server?

Answer: Encryption protects data by converting it into an unreadable format. SQL Server supports:

- **Transparent Data Encryption (TDE)** – Encrypts data at rest
- **Column-level encryption** – Encrypts specific columns
- **Always Encrypted** – Protects sensitive data end-to-end

---

9. What is a backup in SQL Server?

Answer: A backup is a copy of database data used to restore it in case of failure.

Types:

- **Full backup** – Entire database
- **Differential backup** – Changes since last full backup
- **Transaction log backup** – Log of transactions since last log backup

---

10. How do you take a full database backup?

Answer:

```
BACKUP DATABASE MyDB
TO DISK = 'C:\Backup\MyDB.bak';
```

---

11. How do you restore a database from backup?

Answer:

```
RESTORE DATABASE MyDB
FROM DISK = 'C:\Backup\MyDB.bak';
```

---

12. What is the difference between full and differential backup?

Answer:

- **Full backup:** Complete database copy
- **Differential backup:** Only changes since last full backup

---

13. What is the difference between backup and restore?

Answer:

- **Backup:** Creates a copy of the database
- **Restore:** Recovers the database from a backup

---

14. What is a transaction log and why is it important?

Answer: The transaction log records all changes made to the database. It is essential for:

- Rollback transactions
- Point-in-time recovery
- Maintaining database integrity

---

15. What is a maintenance plan in SQL Server?

Answer: A maintenance plan automates routine tasks such as backups, index rebuilding, and database integrity checks.

---

16. How do you check database integrity?

Answer: Using the `DBCC CHECKDB` command:

```
DBCC CHECKDB ('MyDB');
```

---

17. How do you monitor SQL Server performance?

Answer:

- SQL Server Profiler / Extended Events
- Dynamic Management Views (DMVs)
- Activity Monitor in SSMS

---

18. What are DMVs?

Answer: Dynamic Management Views provide system information and performance statistics for SQL Server monitoring and troubleshooting.

---

19. What is SQL Server Agent?

Answer: A Windows service that executes scheduled jobs like backups, maintenance plans, or automated tasks.

---

20. How do you prevent unauthorized access to a database?

Answer:

- Use strong passwords
- Implement Windows Authentication
- Limit permissions
- Encrypt sensitive data
- Audit login activity

---

21. What is a database snapshot?

Answer: A read-only, static view of a database at a specific point in time. Useful for reporting and restoring data.

---

22. What is the difference between DROP, DELETE, and TRUNCATE regarding security?

Answer:

- **DELETE:** Can be rolled back, logs transactions
- **TRUNCATE:** Minimal logging, resets identity, can be rolled back in a transaction
- **DROP:** Removes database or table permanently, cannot be rolled back outside transaction

---

23. What is Transparent Data Encryption (TDE)?

Answer: Encrypts the entire database at rest. Protects physical files from unauthorized access without affecting application code.

---

24. How do you perform database shrink in SQL Server?

Answer:

```
DBCC SHRINKDATABASE (MyDB);
DBCC SHRINKFILE (MyDB_DataFile, target_size);
```

Note: Shrinking should be used sparingly as it can fragment indexes.

---

25. What are best practices for SQL Server maintenance?

Answer:

- Regular backups (full, differential, log)
- Index and statistics maintenance
- Monitoring performance with DMVs
- Apply patches and updates
- Audit security and permissions
- Avoid unnecessary shrinking of databases