**Entity Framework (EF Core)**

1. **Q:** How would you optimize EF Core performance for a high-traffic application?  
   **A:** Use eager loading, disable change tracking for read-only queries, and optimize database indexes.
2. **Q:** What is the difference between eager loading and lazy loading in EF Core?  
   **A:** Eager loading loads related entities upfront, while lazy loading loads them on demand.
3. **Q:** How would you handle database migrations in EF Core?  
   **A:** Use Add-Migration and Update-Database commands to manage schema changes.
4. **Q:** What is the difference between DbContext and DbSet?  
   **A:** DbContext represents a session with the database, while DbSet represents a collection of entities.
5. **Q:** How would you implement a many-to-many relationship in EF Core?  
   **A:** Use a join entity to represent the relationship and configure it in the OnModelCreating method.
6. **Q:** What is the difference between AsNoTracking and AsTracking in EF Core?  
   **A:** AsNoTracking disables change tracking for read-only queries, while AsTracking enables it.
7. **Q:** How would you implement a custom query in EF Core?  
   **A:** Use FromSqlRaw or FromSqlInterpolated to execute raw SQL queries.
8. **Q:** What is the difference between IQueryable and IEnumerable in EF Core?  
   **A:** IQueryable builds queries that are executed on the database, while IEnumerable executes queries in memory.
9. **Q:** How would you handle concurrency conflicts in EF Core?  
   **A:** Use optimistic concurrency control with a RowVersion column.
10. **Q:** What is the difference between Add and Attach in EF Core?  
    **A:** Add marks an entity as new, while Attach marks it as existing but unchanged.
11. **Q:** How would you implement a transaction in EF Core?  
    **A:** Use DbContext.Database.BeginTransaction to start a transaction and commit or rollback as needed.
12. **Q:** What is the difference between Find and FirstOrDefault in EF Core?  
    **A:** Find uses the primary key to locate an entity, while FirstOrDefault uses a query.
13. **Q:** How would you implement a custom migration in EF Core?  
    **A:** Use MigrationBuilder to define custom SQL commands in the Up and Down methods.
14. **Q:** What is the difference between Include and ThenInclude in EF Core?  
    **A:** Include loads related entities, while ThenInclude loads nested related entities.
15. **Q:** How would you handle soft deletes in EF Core?  
    **A:** Use a IsDeleted flag and override the SaveChanges method to filter deleted entities.
16. **Q:** What is the difference between HasOne and WithOne in EF Core?  
    **A:** HasOne defines the principal entity, while WithOne defines the dependent entity.
17. **Q:** How would you implement a custom value converter in EF Core?  
    **A:** Use HasConversion to define a custom conversion between database and application types.
18. **Q:** What is the difference between OnConfiguring and OnModelCreating in EF Core?  
    **A:** OnConfiguring configures the database provider, while OnModelCreating configures the model.
19. **Q:** How would you implement a global query filter in EF Core?  
    **A:** Use HasQueryFilter in the OnModelCreating method to apply a filter to all queries.
20. **Q:** What is the difference between ExecuteSqlRaw and ExecuteSqlInterpolated in EF Core?  
    **A:** ExecuteSqlRaw executes raw SQL, while ExecuteSqlInterpolated uses string interpolation for safer queries.

**Object-Oriented Programming (OOP)**

1. **Q:** How would you design a system where multiple inheritance is required, but the language (e.g., C#) doesn’t support it?  
   **A:** Use interfaces and composition to simulate multiple inheritance.
2. **Q:** Explain the difference between composition and inheritance. When would you prefer composition over inheritance?  
   **A:** Composition builds objects by combining others, while inheritance derives from a base class. Prefer composition for flexibility and to avoid tight coupling.
3. **Q:** How would you implement a deep copy of an object in C# without using serialization?  
   **A:** Implement the ICloneable interface and manually copy all fields and nested objects.
4. **Q:** What is the diamond problem in OOP, and how would you resolve it in a language like C++?  
   **A:** The diamond problem occurs in multiple inheritance when a class inherits from two classes that have a common base. Resolve it using virtual inheritance.
5. **Q:** How do you ensure thread safety in a singleton class without using locks?  
   **A:** Use the Lazy<T> class or static initialization for thread-safe singleton creation.
6. **Q:** What is the difference between early binding and late binding? Provide a real-world example of each.  
   **A:** Early binding occurs at compile time (e.g., method overloading), while late binding occurs at runtime (e.g., dynamic method invocation).
7. **Q:** How would you design a system to handle multiple types of notifications (email, SMS, push) using OOP principles?  
   **A:** Use the Strategy pattern to encapsulate each notification type and switch between them dynamically.
8. **Q:** What is the Liskov Substitution Principle, and how would you handle a scenario where it’s violated?  
   **A:** LSP states that derived classes should be substitutable for their base classes. Refactor the design to ensure derived classes don’t alter base behavior.
9. **Q:** How would you refactor a tightly coupled system into a loosely coupled one using OOP?  
   **A:** Use dependency injection, interfaces, and design patterns like Observer or Mediator.
10. **Q:** What is the difference between an abstract class and an interface? When would you use one over the other?  
    **A:** Abstract classes can have implementations, while interfaces cannot. Use interfaces for multiple inheritance-like behavior and abstract classes for shared logic.
11. **Q:** How would you implement a state machine using OOP principles?  
    **A:** Use the State pattern, where each state is a separate class implementing a common interface.
12. **Q:** What is the difference between polymorphism and method overloading?  
    **A:** Polymorphism allows a method to behave differently based on the object type, while overloading allows multiple methods with the same name but different parameters.
13. **Q:** How would you design a system to handle undo/redo functionality using OOP?  
    **A:** Use the Command pattern to encapsulate actions as objects and maintain a history stack.
14. **Q:** What is the difference between aggregation and association in OOP?  
    **A:** Aggregation implies a "has-a" relationship with independent lifetimes, while association implies a weaker relationship.
15. **Q:** How would you implement a custom iterator for a complex data structure?  
    **A:** Implement the IEnumerable and IEnumerator interfaces to define custom iteration logic.
16. **Q:** What is the role of the sealed keyword in C#, and when would you use it?  
    **A:** sealed prevents a class from being inherited or a method from being overridden. Use it to enforce immutability or security.
17. **Q:** How would you design a system to handle multiple payment gateways using OOP?  
    **A:** Use the Strategy pattern to encapsulate each gateway’s logic and switch between them dynamically.
18. **Q:** What is the difference between a static class and a singleton class?  
    **A:** A static class cannot be instantiated, while a singleton class allows a single instance with lazy initialization.
19. **Q:** How would you implement a generic repository pattern in C#?  
    **A:** Define a base repository interface with CRUD methods and implement it for specific entities.
20. **Q:** What is the role of the yield keyword in C#, and how does it work under the hood?  
    **A:** yield enables deferred execution in iterators by generating a state machine behind the scenes.

**Entity Framework (EF Core)**

1. **Q:** How would you implement a custom migration in EF Core to handle complex schema changes?  
   **A:** Use MigrationBuilder.Sql() to execute raw SQL commands during migrations.
2. **Q:** What is the difference between IQueryable and IEnumerable in EF Core, and how does it impact performance?  
   **A:** IQueryable builds queries that are executed on the database, while IEnumerable executes queries in memory. Using IQueryable is more efficient for database operations.
3. **Q:** How would you implement a soft delete mechanism in EF Core?  
   **A:** Override the SaveChanges method to filter out entities marked as deleted (e.g., using an IsDeleted flag).
4. **Q:** What is the difference between Add, Attach, and Update in EF Core?  
   **A:** Add marks an entity as new, Attach marks it as existing but unchanged, and Update marks it as modified.
5. **Q:** How would you handle concurrency conflicts in EF Core?  
   **A:** Use a RowVersion column and configure it with the IsConcurrencyToken method.
6. **Q:** What is the difference between Eager Loading, Lazy Loading, and Explicit Loading in EF Core?  
   **A:** Eager loading loads related entities upfront, lazy loading loads them on demand, and explicit loading loads them manually using Load().
7. **Q:** How would you implement a custom value converter in EF Core?  
   **A:** Use HasConversion to define a custom conversion between database and application types.
8. **Q:** What is the difference between DbContext and DbSet in EF Core?  
   **A:** DbContext represents a session with the database, while DbSet represents a collection of entities.
9. **Q:** How would you implement a global query filter in EF Core?  
   **A:** Use HasQueryFilter in the OnModelCreating method to apply a filter to all queries.
10. **Q:** What is the difference between FromSqlRaw and FromSqlInterpolated in EF Core?  
    **A:** FromSqlRaw executes raw SQL, while FromSqlInterpolated uses string interpolation for safer queries.
11. **Q:** How would you implement a transaction in EF Core?  
    **A:** Use DbContext.Database.BeginTransaction to start a transaction and commit or rollback as needed.
12. **Q:** What is the difference between Find and FirstOrDefault in EF Core?  
    **A:** Find uses the primary key to locate an entity, while FirstOrDefault uses a query.
13. **Q:** How would you implement a custom database provider in EF Core?  
    **A:** Implement IDatabaseProvider and register it in the DbContextOptions.
14. **Q:** What is the difference between HasOne and WithOne in EF Core?  
    **A:** HasOne defines the principal entity, while WithOne defines the dependent entity.
15. **Q:** How would you implement a custom function in EF Core for use in LINQ queries?  
    **A:** Use DbFunction to map a database function to a C# method.
16. **Q:** What is the difference between ExecuteSqlRaw and ExecuteSqlInterpolated in EF Core?  
    **A:** ExecuteSqlRaw executes raw SQL, while ExecuteSqlInterpolated uses string interpolation for safer queries.
17. **Q:** How would you implement a custom migration strategy in EF Core?  
    **A:** Override the Migrate method in IMigrationsAssembly to customize migration behavior.
18. **Q:** What is the difference between OnConfiguring and OnModelCreating in EF Core?  
    **A:** OnConfiguring configures the database provider, while OnModelCreating configures the model.
19. **Q:** How would you implement a custom database initializer in EF Core?  
    **A:** Override the Initialize method in IDatabaseInitializer.
20. **Q:** What is the difference between AddDbContext and AddDbContextPool in EF Core?  
    **A:** AddDbContext creates a new instance each time, while AddDbContextPool reuses instances from a pool.

**C#**

1. **Q:** How would you implement a thread-safe singleton in C# without using locks?  
   **A:** Use Lazy<T> or static initialization for thread-safe singleton creation.
2. **Q:** What is the difference between Task and ValueTask in C#?  
   **A:** Task is a reference type, while ValueTask is a value type optimized for performance.
3. **Q:** How would you implement a custom exception in C#?  
   **A:** Create a class that inherits from Exception and add custom properties or methods.
4. **Q:** What is the difference between ref and out parameters in C#?  
   **A:** ref passes a reference to a variable, while out is used for output parameters.
5. **Q:** How would you implement a custom attribute in C#?  
   **A:** Create a class that inherits from Attribute and define its properties or methods.
6. **Q:** What is the difference between IEnumerable and IQueryable in C#?  
   **A:** IEnumerable executes queries in memory, while IQueryable builds queries that are executed on the database.
7. **Q:** How would you implement a custom collection in C#?  
   **A:** Create a class that implements IEnumerable<T> and define its methods.
8. **Q:** What is the difference between StringBuilder and string in C#?  
   **A:** StringBuilder is mutable and optimized for concatenation, while string is immutable.
9. **Q:** How would you implement a custom delegate in C#?  
   **A:** Use the delegate keyword to define a custom delegate type.
10. **Q:** What is the difference between IComparable and IComparer in C#?  
    **A:** IComparable defines natural ordering, while IComparer defines custom ordering.
11. **Q:** How would you implement a custom LINQ operator in C#?  
    **A:** Create an extension method that operates on IEnumerable<T> or IQueryable<T>.
12. **Q:** What is the difference between yield return and return in C#?  
    **A:** yield return enables deferred execution, while return immediately exits the method.
13. **Q:** How would you implement a custom event in C#?  
    **A:** Use the event keyword to define an event and EventHandler to handle it.
14. **Q:** What is the difference between struct and class in C#?  
    **A:** struct is a value type, while class is a reference type.
15. **Q:** How would you implement a custom indexer in C#?  
    **A:** Use the this keyword to define an indexer property.
16. **Q:** What is the difference between Task.Run and Task.Factory.StartNew in C#?  
    **A:** Task.Run is a simplified API for starting tasks, while Task.Factory.StartNew provides more options.
17. **Q:** How would you implement a custom serializer in C#?  
    **A:** Create a class that implements ISerializer and define its methods.
18. **Q:** What is the difference between async void and async Task in C#?  
    **A:** async void is for event handlers and cannot be awaited, while async Task can be awaited.
19. **Q:** How would you implement a custom comparer in C#?  
    **A:** Create a class that implements IComparer<T> and define its Compare method.
20. **Q:** What is the difference between Expression and Func in C#?  
    **A:** Expression represents code as a tree, while Func is a delegate that represents a method.

**ASP.NET Core**

1. **Q:** How would you implement custom middleware in ASP.NET Core?  
   **A:** Create a class with Invoke or InvokeAsync method and register it in Startup.cs.
2. **Q:** What is the difference between AddTransient, AddScoped, and AddSingleton in dependency injection?  
   **A:** AddTransient creates a new instance each time, AddScoped creates one per request, and AddSingleton creates one for the application lifetime.
3. **Q:** How would you implement custom model validation in ASP.NET Core?  
   **A:** Create a custom validation attribute by inheriting from ValidationAttribute.
4. **Q:** What is the difference between IApplicationBuilder.Use and IApplicationBuilder.Run?  
   **A:** Use allows the next middleware to execute, while Run terminates the pipeline.
5. **Q:** How would you implement JWT authentication in ASP.NET Core?  
   **A:** Use AddAuthentication with JwtBearerDefaults.AuthenticationScheme and configure the token validation parameters.
6. **Q:** What is the difference between ActionResult and IActionResult?  
   **A:** ActionResult is a concrete type, while IActionResult is an interface representing various HTTP responses.
7. **Q:** How would you implement rate limiting in ASP.NET Core?  
   **A:** Use middleware like AspNetCoreRateLimit to limit requests based on IP or other criteria.
8. **Q:** What is the difference between appsettings.json and environment variables in ASP.NET Core?  
   **A:** appsettings.json is a configuration file, while environment variables are used for environment-specific settings.
9. **Q:** How would you implement a custom route constraint in ASP.NET Core?  
   **A:** Create a class that implements IRouteConstraint and register it in Startup.cs.
10. **Q:** What is the difference between FromServices and FromBody in ASP.NET Core?  
    **A:** FromServices injects a service, while FromBody binds data from the request body.
11. **Q:** How would you implement a custom exception filter in ASP.NET Core?  
    **A:** Create a class that implements IExceptionFilter and register it in Startup.cs.
12. **Q:** What is the difference between Map and MapWhen in middleware?  
    **A:** Map branches the pipeline based on the path, while MapWhen branches based on a condition.
13. **Q:** How would you implement a custom output formatter in ASP.NET Core?  
    **A:** Create a class that implements OutputFormatter and register it in Startup.cs.
14. **Q:** What is the difference between AddMvc and AddControllersWithViews?  
    **A:** AddMvc registers all MVC services, while AddControllersWithViews registers only controllers and views.
15. **Q:** How would you implement a custom authorization policy in ASP.NET Core?  
    **A:** Use AuthorizationPolicyBuilder to define the policy and register it in Startup.cs.
16. **Q:** What is the difference between IHostedService and BackgroundService?  
    **A:** IHostedService is the base interface, while BackgroundService is an abstract class for background tasks.
17. **Q:** How would you implement a custom view component in ASP.NET Core?  
    **A:** Create a class that inherits from ViewComponent and define its Invoke method.
18. **Q:** What is the difference between UseRouting and UseEndpoints?  
    **A:** UseRouting sets up routing, while UseEndpoints defines the endpoints.
19. **Q:** How would you implement a custom tag helper in ASP.NET Core?  
    **A:** Create a class that inherits from TagHelper and define its Process method.
20. **Q:** What is the difference between AddDbContext and AddDbContextPool?  
    **A:** AddDbContext creates a new instance each time, while AddDbContextPool reuses instances from a pool.