1. Unique elements from two collections (LINQ)

Answer: Use Union() or Concat().Distinct().
Example: var unique = coll1.Union(coll2);
Note: Union removes duplicates; Concat().Distinct() is equivalent.

---

2. Difference between two collections

Answer: Use Except() to get elements in first not in second.
Example: var diff = coll1.Except(coll2);

---

3. When will finally block execute?

Answer: finally executes after try (and any catch) runs — even if the method returns — except when the process is terminated (e.g., Environment.FailFast, killing the process, or certain CLR fatal errors).
Note: break, return, or exceptions do not prevent finally from running.

---

4. What depends on access specifiers?

Answer: Visibility of types and members (which code can access a class/method/field). Access affects inheritance, accessibility from other assemblies, and encapsulation.
Note: Access modifiers: public, protected, internal, private, protected internal, private protected.

---

5. What are access modifiers?

Answer: Language keywords that control accessibility: public, private, protected, internal, protected internal, private protected.
Note: Default accessibility for a top-level class is internal if no modifier is specified.

---

6. Which of the below are creational design patterns?

Answer: Examples: Factory Method, Abstract Factory, Builder, Prototype, Singleton.
Note: Creational patterns focus on object creation.

---

7. Builder is what design pattern?

Answer: Builder is a **creational** design pattern used to build complex objects step-by-step.
Example: A CarBuilder that constructs Car in steps.

---

8. (Illegible/ambiguous) — looks like an unclear note about errors and try/catch

Answer (likely intent): try handles exceptions thrown in try; catch receives thrown exception types; finally always runs.
Note: Put only the code likely to throw in try and keep catch blocks specific.

---

9. What is true about finally block — "no break/exit can stop finally" (handwritten)

Answer: Correct: finally runs even if break or return occurs inside try/catch. Only abnormal process termination prevents it.
Example:

```
int Foo() {
  try { return 1; }
  finally { Console.WriteLine("runs"); }
}
```

---

10. Why do we use singleton class / what is true about it?

Answer: Singleton ensures a single instance for a class and provides a global access point.
Note: Use careful thread-safe initialization (e.g., Lazy<T> or double-check locking).

---

11. General purpose of singleton class

Answer: Control access to a single shared resource (logging, configuration, caches). Avoid overuse—it's effectively global state.

---

12. What object do we use to read from / write to streams?

Answer: Use FileStream, StreamReader (read), StreamWriter (write); BinaryReader/BinaryWriter for binary.
Example: using(var sr = new StreamReader(new FileStream(path, FileMode.Open))) { ... }

---

13. What is true about abstract class?

Answer: You **cannot instantiate** an abstract class directly; it can have abstract and concrete members; a derived class must implement abstract members; abstract classes support inheritance but cannot be sealed; they can have static members.
Note: Use abstract class when you want some shared implementation.

---

14. FileStreams — how to open

Answer: var fs = new FileStream(path, FileMode.Open, FileAccess.Read); or use File.OpenRead(path) plus using pattern to ensure disposal.

---

15. Query expression vs lambda vs anonymous

Answer: LINQ supports both query syntax (from x in coll where ... select) and method syntax with lambda expressions (coll.Where(x => ...)). Anonymous types are new { Name = ..., Age = ... } used in queries.
Example: var q = from p in people where p.Age>20 select new { p.Name };

---

16. Extension method

Answer: A static method in a static class with this on the first parameter, e.g. public static void Foo(this string s).
Notes:

- Can appear to "add" methods to existing types.
- Cannot access private/internal members of the target type.
- Resolution: instance methods take precedence over extension methods.

---

17. When prefer Dictionary over Array/List?

Answer: Use Dictionary<TKey, TValue> for fast key-based lookup (average O(1)). Use List<T> or arrays when you need ordered access or index-based iteration.
Note: When keys are unique and lookups frequent, use Dictionary.

---

18. Default access specifier of an interface

Answer: Top-level interfaces are internal by default in C# (if no access modifier and in a namespace). Inside a class an interface member's rules differ.
Note: If you declare public interface IName {} then it's public.

---

19. Can an interface have multiple methods with same name and same signature?

Answer: You cannot have two members in the same interface with identical signatures. However, an interface can inherit from multiple interfaces that happen to have the same method signature — that's fine and they merge.
Note: Explicit interface implementation can resolve name conflicts between different interfaces if they have same method name but different semantics.

---

20. What is a synchronous process in threads?

Answer: A synchronous call blocks the calling thread until the operation completes. Asynchronous operations return control immediately (e.g., Task/async/await) and do not block.
Example: Thread.Sleep(1000) is synchronous; await Task.Delay(1000) is asynchronous.

---

21. (Handwritten "universalization/" unclear) — likely: virtualization/applicability to public/private/protected/internal

Answer (likely interpreted): Modifiers like virtual/override apply to instance members and are subject to accessibility. virtual can be public, protected, etc. private members cannot be virtual (because they can't be overridden by derived types).
Note: private methods are not visible to derived types, so virtual/override isn't meaningful there.

---

22. Which of the following is true about garbage collector

Answer (core facts): GC reclaims managed heap memory non-deterministically, uses generations (0/1/2), can compact the heap, and finalizers (~Class()) run before collection but timing is non-deterministic. Call GC.Collect() rarely — prefer using/IDisposable for unmanaged resources.

---

23. Which is correct about memory management

Answer: Managed memory is handled by CLR GC; unmanaged resources must be released explicitly (implement IDisposable and use using or call Dispose()).

---

24. Which design pattern is associated with object creation

Answer: Creational patterns: Factory Method, Abstract Factory, Builder, Prototype, Singleton.

---

25. Which of the following is not true about extension methods

Answer (common false statements): *"Extension methods can override instance methods"* — That's false. Extension methods **do not** override instance methods; instance methods always win. Also extension methods cannot access private members.

---

26. Which keyword will you use to change behaviour of base class?

Answer: To allow changing behavior you mark base method virtual and the derived override it with override. You can also use new to hide a base member (but override is the standard polymorphic way).
Example:

public virtual void Foo() { }

public override void Foo() { } // in derived

---

27. override (27) Virtual - keyword in base class allows override

Answer: Correct: virtual on base + override in derived. You can also seal an override: public sealed override void Foo() { }.

---

28. Boxing

Answer: Boxing converts a value type to object (or to an interface). Unboxing converts it back to the value type. Boxing allocates on heap and can be expensive in hot code paths.
Example:

```
int i = 5;

object o = i; // boxing

int j = (int)o; // unboxing
```

---

29. Array length question (handwriting: int[] a = {1,2,3,4,3} a.length = ?)

Answer: a.Length == 5 because there are five elements (1,2,3,4,3).
Note: Length is the property on arrays (capital L).

---

30. "2 methods lock{} -> deadlock" (from the other photo)

Answer: If two threads lock two resources in opposite order, a deadlock can occur. Avoid by consistent lock ordering or using Monitor.TryEnter with timeout.
Example problem: Thread A locks R1 then R2; Thread B locks R2 then R1 -> deadlock possible.

---

31. Finally -> cannot have exit/break/return ? (handwritten)

Clarification & Answer: You *can* return or break inside try or catch; finally still runs. You **should not** rely on finally for resource disposal when process may be terminated; use using for deterministic disposal.

---

32. "Method name same but different params & types -> Overloading"

Answer: Correct — same method name with different parameter list (type/number/order) is **compile-time overloading**. Overload resolution happens at compile time.

---

33. "Exception raised at -> Runtime"

Answer: Many exceptions in .NET are runtime exceptions (e.g., NullReferenceException, IndexOutOfRangeException). Some errors are compile-time (syntax, missing references). Runtime exceptions happen while program runs.

Quick small code examples you can try

Overloading

```
void Foo(int x) { }

void Foo(string s) { }
```

Finally always runs

```
int Test() {
  try { return 1; }
  finally { Console.WriteLine("Finally runs"); }
}
```

Singleton (thread-safe, simple)

```
public sealed class Singleton {
  private static readonly Lazy<Singleton> _instance = new(() => new Singleton());
  public static Singleton Instance => _instance.Value;
  private Singleton() { }
}
```

Using file reading

```
using(var sr = new StreamReader("file.txt")) {
  var text = sr.ReadToEnd();
}
```

Important exam/assignment tips

- For resource cleanup use using / IDisposable — don't rely on finalizers or GC timing.
- Remember virtual + override for polymorphism; new hides members (not polymorphic).

- LINQ: know Where, Select, Union, Except, Distinct, and query vs method syntax.
- Understand difference between compile-time (overload) and run-time (override) polymorphism.
- Avoid boxing in tight loops — use generics (List<T>, Dictionary<TKey,TValue>) to prevent boxing.

🧪 **Page 1 (NUnit / Unit Testing Concepts)**

30. How do you mark a method for testing in NUnit?

✅ **Answer:** Use the [Test] attribute.
Example:

```
[Test]
public void Add_TwoNumbers_ReturnsSum() {
    Assert.AreEqual(4, Calculator.Add(2, 2));
}
```

Explanation:

- [Test] tells NUnit that the method is a test case.
- Must be a public void method inside a [TestFixture] class.

---

31. How to group related tests with setup and teardown?

✅ **Answer:** Use [SetUp] and [TearDown] (or [OneTimeSetUp], [OneTimeTearDown]).
Example:

```
[TestFixture]
public class MathTests {
    [SetUp]
    public void Init() => Console.WriteLine("Before each test");
```

```
    [TearDown]
    public void Cleanup() => Console.WriteLine("After each test");

    [Test]
    public void Add_Test() => Assert.AreEqual(5, 2 + 3);
}
```

Explanation:

- [SetUp]: Runs before each [Test].
- [TearDown]: Runs after each [Test].
- [OneTimeSetUp] and [OneTimeTearDown] run once per test class (useful for DB connections, expensive setup, etc.).

Important: NUnit 3 deprecated [TestFixtureSetUp] → replaced with [OneTimeSetUp].

🗄️ **Page 2 (Entity Framework, ADO.NET, .NET Core, and DB basics)**

Let's go line by line 👇

---

1. To know the status of a database (Property)

✅ **Answer:** State property of the SqlConnection object.
Example:

```
SqlConnection con = new SqlConnection(connStr);
Console.WriteLine(con.State); // Closed, Open, Connecting, etc.
```

---

2. Namespace for data views

✅ **Answer:** System.Data.
Explanation: ADO.NET core classes like DataSet, DataTable, DataView are in System.Data.

---

3. Entity Data Model in memory with CSDL, SSDL, and MSL

✅ **Answer: Entity Framework (EF)** uses these XML schema definitions.

- **CSDL** – Conceptual Schema Definition Language (Entity model)
- **SSDL** – Store Schema Definition Language (Database schema)
- **MSL** – Mapping Specification Language (Mapping between conceptual & storage models)

---

4. Line while verifying if test passes or not

✅ **Answer:** Assert
Example:

```
Assert.AreEqual(expected, actual);
```

---

5. Check behavior of a method without modifying original code

✅ **Answer: Mocking**
Explanation: Create a *mock object* to simulate dependencies (often using libraries like Moq).

---

6. 2 developers work on Product and Cost... (Product not available, cost should check)

✅ **Answer: Mocking** again — allows testing code independently of unavailable modules.

---

7. Create objects → initialize data → work in AAA

✅ **Answer: Arrange** (AAA = Arrange, Act, Assert)

---

8. AAA contains

✅ **Answer: Arrange, Act, Assert**

---

9. .NET Core framework requires licensing to use?

✅ **Answer: False** — .NET Core is open-source and free.

---

10. .NET Core is open source

✅ **Answer: True**

---

11. Which helps with adding unit testing and dependencies easily?

✅ **Answer: Dependency Injection (DI)**
Explanation: Improves **testability**, **modularity**, and **loose coupling**.

---

12. True/False: Dataset is faster than DataReader

✅ **Answer: False**
Explanation:

- **DataReader** is faster (forward-only, read-only).
- **DataSet** is slower (disconnected, in-memory, more overhead).

---

13. Instead of creating your own dependencies...

✅ **Answer:** Use **Dependency Injection (DI)** framework.
Advantages: Maintainability, Testability, Flexibility.

---

14. Advantages of Dependency Injection

✅ **Answer:**

- Maintainability
- Testability
- Flexibility (Loose coupling)

---

15. Not ADO.NET properties: Fill, FillSchema, Update, ReadData/WriteData

✅ **Answer:** ReadData/WriteData are **not** ADO.NET methods.
ADO.NET uses Fill(), FillSchema(), Update() in DataAdapter.

---

16. Work cannot be applied on classes?

✅ **Answer: Static classes** cannot be instantiated or inherited.

---

17. Can't access a row or record with ID = X if saved or updated?

✅ **Answer:** Likely referring to **Persistence** — data not yet committed or detached from EF context.
Explanation: Detached entities lose EF tracking; need to re-attach to update.

---

18. Connection string: the part specifying the DB name

✅ **Answer:** Initial Catalog or Database keyword.
Example:
"Server=.;Initial Catalog=StudentDB;Integrated Security=True;"

---

19. Where is key typically used to create a primary key

✅ **Answer: Column** (in a table).
Explanation: Primary key uniquely identifies each row.

---

20. Namespace which enables accessing & managing web.config file

✅ **Answer:** System.Web.Configuration

---

21. Binary in SQL, what is it in C#?

✅ **Answer:** byte[]

---

22. True about .NET Core (2 options)

✅ **Answer:** Cross-platform and open-source.

---

23. Reference external DLLs in .NET Core

✅ **Answer:** Add references using <PackageReference> in .csproj or dotnet add reference.
Example:
dotnet add package Newtonsoft.Json

---

24. Default isolation level

✅ **Answer: ReadCommitted**
Explanation: Prevents dirty reads but allows non-repeatable reads and phantom reads.

---

25. Valid EF frameworks

✅ **Answer:**

- Model First
- Database First
- Code First

---

26. If you already have a database, what approach?

✅ **Answer: Database First**

---

27. Which allows reuse of database connection?

✅ **Answer: Connection Pooling**
Explanation: ADO.NET reuses open connections for performance.

---

28. Insert during database initialization / custom DB using which class

✅ **Answer:** CreateDatabaseIfNotExists<TContext> (EF initializer).
Explanation: EF Database Initializers —

- CreateDatabaseIfNotExists
- DropCreateDatabaseIfModelChanges
- DropCreateDatabaseAlways

Example:

Database.SetInitializer(new CreateDatabaseIfNotExists<MyContext>());

---

29. Create Database If Not Exists → Class?

✅ **Answer:** CreateDatabaseIfNotExists<TContext>
(same as 28)

⚙️ **Quick Concepts Recap**

Concept	Description
Mocking	Faking dependencies to test logic in isolation
AAA	Arrange → Act → Assert testing pattern
Dependency Injection	Provides dependencies instead of hardcoding
Connection Pooling	Reuses DB connections for better performance
Isolation Level	Controls concurrency and transaction consistency
EF Approaches	Model First, Database First, Code First
NUnit Attributes	[Test], [SetUp], [TearDown], [TestFixture]
Assert Methods	Assert.AreEqual, Assert.IsTrue, Assert.Throws, etc.

📘 **ASP.NET MVC / Web API — Questions & Answers**

1️⃣ **if(foo) : ... => ? if(foo) { ... }**

✅ **Answer:** if(foo) ...; else ...;
This shorthand means: **conditional execution** — same as normal if statement.

2️⃣ **What are strongly-typed views?**

✅ **Answer:** Views bound to a specific model class using @model.
Example:

@model Student

@Model.Name

Note: Helps with IntelliSense and compile-time type checking.

3️⃣ **Global.asax is not mandatory**

✅ **Answer: True**
Explanation: Global.asax handles application-level events (e.g. Application_Start), but it's optional in .NET Core.

4️⃣ **Web form derived from which class?**

✅ **Answer:** System.Web.UI.Page

5️⃣ **MVC stands for?**

✅ **Answer: Model-View-Controller**

6️⃣ **What does HTML helper generate?**

✅ **Answer:** HTML elements.
Example: @Html.TextBoxFor(m => m.Name)

7️⃣ **All static data, CSS, and JS files are under which folder?**

✅ **Answer:** /wwwroot (in ASP.NET Core) or /Content & /Scripts (in classic MVC).

8️⃣ To create a button with no data type — use?

✅ **Answer:** @Html.TextBox() or @Html.Button() (depending on type)
More likely: @Html.ActionLink() to create clickable links.

9️⃣ Email validation

✅ **Answer:** Use RegularExpressionValidator or data annotation [EmailAddress].

🔟 Currency validation

✅ **Answer:** CompareValidator or [Range] with currency formatting.

1️⃣1️⃣ All file names of an application are stored in?

✅ **Answer:** RouteTable (for MVC route configuration).

1️⃣2️⃣ Filters in MVC?

✅ **Answer:**

- **Action Filters** — run before/after controller actions
- **Result Filters** — before/after view rendering
- **Exception Filters** — handle unhandled exceptions
- **Authorization Filters** — before executing actions

1️⃣3️⃣ For unhandled errors → ?

✅ **Answer:** [HandleError] attribute or middleware (UseExceptionHandler in Core).

1️⃣4️⃣ Shared elements in MVC?

✅ **Answer: Layout Pages, Partial Views, Sections, AJAX views.**

1️⃣5️⃣ Valid Session types

✅ **Answer:**

- InProc (default)
- StateServer
- SQLServer
- Custom
- Off

1️⃣6️⃣ Display complete model or data?

✅ **Answer: View**

1️⃣7️⃣ What handles business logic?

✅ **Answer: Controller**

1️⃣8️⃣ ViewState disadvantage

✅ **Answer:** Cannot be used across pages; increases page size; only works in ASP.NET Web Forms (not MVC).

1️⃣9️⃣ Which is NOT an action result?

✅ **Answer:** NonActionResult (trick question — not real).
Action Results examples: ViewResult, JsonResult, RedirectResult, ContentResult.

2️⃣0️⃣ An application can have multiple web.config files

✅ **Answer: True** (one per folder for configuration inheritance).

2️⃣1️⃣ ViewBag type

✅ **Answer: Dynamic** type (no need to declare properties).
Example: ViewBag.Title = "Home Page";

2️⃣2️⃣ ViewBag is dynamic, so no need to type cast

✅ **Answer: True**

2️⃣3️⃣ Where is connection string present?

✅ **Answer:** web.config (or appsettings.json in .NET Core).

2️⃣4️⃣ What do you use to update (data)?

✅ **Answer: PUT** (HTTP verb).

2️⃣5️⃣ Which made API creation faster?

✅ **Answer: Minimal APIs** (introduced in .NET 6+).

🌐 **Web API / ASP.NET Advanced**

2️⃣7️⃣ For Web API, what must be used with client?

✅ **Answer: HTTP Client** (HttpClient class).
Example:

```
HttpClient client = new HttpClient();

var response = await client.GetAsync("https://api.example.com/data");
```

---

2 8 Reusable component of web?

✅ **Answer: Partial View** (like shared header/footer).

---

2 9 What executes before and after an action method?

✅ **Answer: Action Filters** (OnActionExecuting, OnActionExecuted).

---

3 0 What handles requests/responses in pipeline?

✅ **Answer: Middleware**

---

3 1 MVC controller inherits from which class?

✅ **Answer:** ControllerBase or Controller.

---

3 2 Which of the following is NOT true about action methods?

✅ **Answer:** They cannot be private; must be public.
Note: Can be overloaded, can return ActionResult, cannot be static.

---

3 3 Which of the following is used to define user settings/configuration?

✅ **Answer:** appsettings.json (in .NET Core) or web.config (in classic).

---

3 4 Storage of state of server objects across requests?

✅ **Answer:** ViewState (in Web Forms only).

---

3 5 Which is NOT a type of validation?

✅ **Answer:** Example of non-validation: *State validator* (made-up).
Actual types: RequiredField, Range, Compare, RegularExpression, Custom.

3️⃣6️⃣ Where should be a variable declared to be used across app?

✅ **Answer:** Application_Start in Global.asax.
Explanation: Declares global variables accessible to all sessions/pages.

3️⃣7️⃣ Select routing statements?

✅ **Answer:** Defined in RouteConfig.cs (MVC 5) or via MapControllerRoute() in .NET Core.
Example:

```
app.MapControllerRoute(
  name: "default",
  pattern: "{controller=Home}/{action=Index}/{id?}");
```

3️⃣8️⃣ Why is web.config used?

✅ **Answer:** To configure and declare variables globally (connection strings, custom errors, etc.)

3️⃣9️⃣ Which file defines settings of web application?

✅ **Answer:**

- .NET Framework: web.config
- .NET Core: appsettings.json
- Machine-wide: machine.config
- Global events: global.asax

📝 **Extra Note at Bottom**

Session State Modes:
InProc, OutProc (StateServer), SQLServer, Custom, Off

💡 Important Summary Table

Concept	Description / Example
HTTP Client	Used to call Web APIs
Partial View	Reusable part of UI
Action Filter	Runs before/after action
Middleware	Handles requests/responses in pipeline
ControllerBase	Base class for controllers
AppSettings / Web.Config	Stores user & app settings
ViewState	Stores state on client side (Web Forms only)
Application_Start	Global variable setup in Global.asax
Routing	Defines URL patterns
Minimal API	Simplified API creation (no controllers)

✅ Recommended short revision order:

1. MVC lifecycle (Request → Routing → Controller → View)
2. Filters (Action, Authorization, Exception)
3. State management (Session, ViewState, Cookies)
4. Validation types (Required, Range, Compare, Regex, Custom)
5. Configuration (web.config, appsettings.json, Global.asax)
6. HTTP methods (GET, POST, PUT, DELETE)
7. Middleware & Routing setup.

1️⃣ Let is block scope

✅ **Answer:** True
Explanation: let and const are **block-scoped** (inside {}), unlike var (function-scoped).

2️⃣ **Const can't be re-declared or updated**

✅ **Answer:** True
Explanation:
const variables are immutable references — can't be re-assigned or re-declared.

3️⃣ **Determine type of a variable**

✅ **Answer:** typeof operator.
Example: typeof "Hello" // "string"

4️⃣ **Role: font-size – div3**

✅ **Answer:** CSS example → div3 { font-size: … } sets the font size for div with class/id.

5️⃣ **Which contains related functions, classes, variables, interfaces**

✅ **Answer: Module** (in TypeScript).

6️⃣ **JavaScript automatic type conversion**

✅ **Answer:** True
Explanation: JS converts types implicitly ('5' + 2 → '52').

7️⃣ **Role (date): 3/12/25 to Min (for short)**

✅ **Answer:** Example of <input type="date" min="2025-03-12">.

8️⃣ **Const is block scope**

✅ **Answer:** True

9️⃣ **Let can be updated but not re-declared**

✅ **Answer:** True
Example:

let x = 1;

x = 2; // ok

let x = 3; // ❌ error

🔟 **Let can be re-declared but not updated**

✅ **Answer:** False

1️⃣1️⃣ Section

✅ **Answer:** <section> tag in HTML5 — used to group related content.

1️⃣2️⃣ Color: gray → section is gray & link is gray

✅ **Answer:** Example CSS inheritance — parent section { color: gray; } affects links unless overridden.

1️⃣3️⃣ Button class for large

✅ **Answer:** btn-lg (Bootstrap class).

1️⃣4️⃣ How to mention HTML5 doctype

✅ **Answer:** <!DOCTYPE html>

---

1 5 Body color

✅ **Answer:** CSS — body { color: black; }

---

1 6 Tags introduced in HTML5

✅ **Answer:** <nav>, <aside>, <section>, <article>, <header>, <footer>

---

1 7 What is not true about pipes (Angular)

✅ **Answer:** *They modify the original data.* → ❌ False
Explanation: Pipes transform data for display but do not change source values.

---

1 8 Timeout / SetTimeout

✅ **Answer:** setTimeout(function, delay)
Explanation: Schedules execution of a function after delay (in milliseconds).

---

1 9 Increment by 1 examples

✅ **Answer:** x++, ++x, x += 1, x = x + 1, ++x (prefix gives incremented value).

---

2 0 Input type number

✅ **Answer:**

<input type="number" id="age" min="0" max="100">

---

2 1 Defines scope and behavior of variable

✅ **Answer:** var, let, const
Explanation:

- var: function-scoped
- let, const: block-scoped

2️⃣2️⃣ Function add(a=10, b=5) { return a+b } → output?

✅ **Answer:** 15
Explanation: Default parameters → 10+5 = 15.

⚙️ **TypeScript + Angular + DOM + CSS**

2️⃣3️⃣ let var = (x) => return 2*x+4

✅ **Answer:** Arrow function syntax:

let calc = (x) => 2 * x + 4;

Explanation: Implicit return of expression in arrow functions.

2️⃣4️⃣ document.getElementById

✅ **Answer:** Accesses HTML element by ID.
Example:

document.getElementById("demo").innerHTML = "Hi";

2️⃣5️⃣ Unique value → ID

✅ **Answer:** Each element must have unique id in HTML.

2️⃣6️⃣ Center justify

✅ **Answer:** CSS — text-align: center;

2️⃣7️⃣ TypeScript compilation cmd

✅ **Answer:** tsc filename.ts
Explanation: Compiles .ts → .js.

---

2 8 npm install -g typescript

✅ **Answer:** Installs TypeScript globally using Node Package Manager (NPM).

---

2 9 TypeScript configuration file

✅ **Answer:** tsconfig.json
Explanation: Defines compiler options, file paths, target version, etc.

---

3 0 Add style to border-left

✅ **Answer:** CSS — border-left-style: solid;

---

3 1 Space inside div (content spacing)

✅ **Answer:** padding (inside) and margin (outside).

---

3 2 Data binding can be done (Angular)

✅ **Answer: Input** and **Output** bindings — @Input(), @Output()
Example: <child [data]="parentData" (notify)="onNotify()">

---

3 3 To bind input text box value

✅ **Answer:** Use ngModel (two-way binding).
Example: <input [(ngModel)]="username">

---

3 4 CSS styling to a conditional element

✅ **Answer:** Use [ngStyle] or [ngClass] directives.
Example:

<div [ngStyle]="{'color': isRed ? 'red' : 'blue'}">

---

3 5 Correct way to initialize array in JS

✅ **Answer:**

let arr = [1, 2, 3];

or

let arr = new Array(1, 2, 3);

---

3 6 Next Angular lifecycle hook after initializing data-bound elements

✅ **Answer:** ngAfterViewInit()
Explanation: Called after component's view (and child views) initialized.

---

3 7 How to include style?

✅ **Answer:** <style> tag inside HTML <head> or <link rel="stylesheet" href="style.css">
In Angular: via styleUrls or styles in @Component.

🌟 **Important JS / TS / Angular Revision Summary**

Concept	Example	Explanation
let / const	let a=1; const b=2;	Block scope
typeof	typeof 42 // "number"	Finds data type
Arrow function	(x) => x*2	Short syntax
DOM Access	document.getElementById()	Manipulate elements

Concept	Example	Explanation
Data Binding	[(ngModel)]="name"	Two-way binding
Lifecycle Hooks	ngOnInit(), ngAfterViewInit()	Component events
CSS in Angular	[ngStyle], [ngClass]	Conditional styling
TypeScript Compile	tsc app.ts	Transpile to JS
Config file	tsconfig.json	TypeScript compiler settings
State Mgmt	@Input, @Output, services	Component communication

1️⃣ **Symbol for creating temporary tables in SQL Server**

✅ **Answer:** #
Explanation:

- #temp → Local temporary table (exists for current session)
- ##temp → Global temporary table (available for all sessions)

Example:

CREATE TABLE #TempStudents (ID INT, Name NVARCHAR(50));

2️⃣ **GO keyword**

✅ **Answer: Batch separator** in SQL Server Management Studio (SSMS).
Explanation: Tells SQL Server to execute the current batch of T-SQL statements.

3️⃣ **Product ID ends with 10_3**

✅ **Answer:** Use LIKE with wildcard _ (represents a single character).

SELECT * FROM Product WHERE ProductID LIKE '%10_3';

4️⃣ **Properties of SSMS (SQL Server Management Studio)**

✅ **Answer:**

- GUI tool to manage SQL Server
- Query execution & debugging
- Database design (tables, views, SPs)
- Supports Import/Export, Backup/Restore
- Object Explorer & Template Explorer

5️⃣ **SELECT 1/2**

✅ **Answer:** Output → 0
Explanation: Integer division (1 and 2 are integers).
Use float division:

SELECT 1.0/2; -- Output: 0.5

6️⃣ **SELECT 15 + 25**

✅ **Answer:** 40

7️⃣ **Find index of 'e' in 'jenna'**

✅ **Answer:** In SQL:

SELECT CHARINDEX('e', 'jenna'); -- Output: 2

Explanation: CHARINDEX() returns the position of substring in string.

8️⃣ **To use external library in C++**

✅ **Answer:** #include
Example:

#include <iostream>

#include <math.h>

9️⃣ **Cardinality of University to College**

✅ **Answer: One-to-Many (1:N)**
Explanation: One University → Many Colleges.

🔟 **SQL: SELECT 123 + 'abc'**

✅ **Answer:** Implicit conversion error or output = 123 (depending on SQL mode).
In SQL Server, 'abc' converts to 0, so **result = 123**.
Explanation: String to numeric implicit conversion → 'abc' → 0.

1️⃣1️⃣ **TRY–CATCH Example**

✅ **Answer:**

```
BEGIN TRY
  SELECT 'foo' AS Result;
END TRY
BEGIN CATCH
  SELECT 'bar' AS Result;
END CATCH;
```

Output: 'foo' (no error, so CATCH not executed).

1️⃣2️⃣ **At the end '10_3'**

✅ **Answer:** Similar to earlier — use LIKE '%10_3'.

🧰 GIT COMMANDS

1️⃣ Create empty directory (repo)

✅ **Answer:**

git init

2️⃣ Undo staged changes but keep file modifications

✅ **Answer:**

git reset

Explanation: Removes from staging area but preserves working directory changes.

3️⃣ Commit with a message

✅ **Answer:**

git commit -m "Your message"

4️⃣ Git command to copy (download) repo to client

✅ **Answer:**

git clone <repository_url>

5️⃣ Git command to show difference between working files and staged files

✅ **Answer:**

git diff

6️⃣ Git command to push local commits to remote

✅ **Answer:**

git push

7️⃣ **git status**

✅ **Answer:** Displays current branch, staged/unstaged files, and untracked files.

💻 **PROGRAMMING & GENERAL SOFTWARE ENGINEERING**

1️⃣ **Programming type for general purpose**

✅ **Answer: Procedural programming** or **Object-Oriented programming (OOP)**
Explanation: Used for general software development — examples: C, C++, Java, C#.

2️⃣ **Datatype of image in SQL Server**

✅ **Answer:**

- **Deprecated:** image
- **Use instead:** VARBINARY(MAX)

3️⃣ **Not an aggregate function**

✅ **Answer: COMPUTE** (not a function; it's an old T-SQL clause).
Examples of aggregate functions: SUM, AVG, MIN, MAX, COUNT.

4️⃣ **Keyword to finalize a transaction**

✅ **Answer:** COMMIT

5️⃣ **Software changes to correct errors (debugging)**

✅ **Answer:** a) **Corrective Maintenance**

6️⃣ **Features of SQL Server**

✅ **Answer:**

- High performance RDBMS
- Transaction management
- Data security & backup
- Stored procedures, triggers
- Integration with .NET, Reporting Services

7️⃣ **Step done in requirement gathering**

✅ **Answer:** a) **Analysis**

8️⃣ **Taking input values & displaying output**

✅ **Answer:** b) **Terminal symbol** (in flowchart)

9️⃣ **SQL syntax to check for NULL**

✅ **Answer:**

SELECT * FROM Students WHERE Marks IS NULL;

🔟 **Application layer functions in N-tier architecture**

✅ **Answer:**

- Business logic execution
- Validation
- Security and authentication

- Communication with Data Layer

1️⃣ Flowchart feature

✅ **Answer:** Visual representation of algorithm using symbols (process, decision, input/output).

1️⃣2️⃣ Order of Normal Forms

✅ **Answer:**
1NF → 2NF → 3NF → BCNF → 4NF → 5NF

1️⃣3️⃣ Pre-requisite for 4NF

✅ **Answer:** Must already satisfy **BCNF** and have **no multi-valued dependencies**.

1️⃣4️⃣ Symbol for Entity Set

✅ **Answer: Rectangle**

1️⃣5️⃣ Computer program structure

✅ **Answer:** Sequence → Decision → Iteration/Recursion
Explanation: Fundamental control structures.

1️⃣6️⃣ Stage where developer and tester work together

✅ **Answer: Integration testing** or **Agile / DevOps collaboration phase** (Verification & Validation).

1️⃣7️⃣ Why use normalization

✅ **Answer:** To reduce data redundancy and improve data integrity.

1️⃣8️⃣ Informal or artificial language to represent a program

✅ **Answer: Pseudocode**

1️⃣9️⃣ Join to get common elements

✅ **Answer: INNER JOIN**

2️⃣0️⃣ Join giving Cartesian product

✅ **Answer: CROSS JOIN**

🧠 **BONUS QUICK RECALL TABLE**

Topic	Keyword / Syntax	Meaning
Temp Table	#	Local temporary table
Go	Batch separator	Executes batch
CHARINDEX	CHARINDEX('e','jenna')	Finds index
External library in C++	#include	Header import
SQL Image datatype	VARBINARY(MAX)	Store images
Transaction End	COMMIT	Save changes
Undo staged Git changes	git reset	Unstage
Push commits	git push	Send to remote
Null check	IS NULL	Check for nulls
Normalization Order	1NF → 2NF → 3NF → BCNF → 4NF → 5NF	Table normalization
Entity Symbol	Rectangle	ER Diagram

Topic	Keyword / Syntax	Meaning
Join (Common Elements)	INNER JOIN	Matching rows
Join (Cartesian)	CROSS JOIN	All combinations
Maintenance for Bugs	Corrective	Error fixing