|  |  |
| --- | --- |
| **Day 1: .NET Basics and C# Fundamentals**  **Morning (9:00 AM - 12:00 PM)**  1. \*Introduction to .NET Framework and .NET Core\*  - Overview of .NET Framework, .NET Core, and .NET 5/6  - CLR, CTS, and CLS  - Managed vs. unmanaged code  - Key differences between .NET Framework and .NET Core  2. \*C# Basics\*  - Data types and variables  - Control structures (if, switch, loops)  - Methods and parameters  **Afternoon (1:00 PM - 5:00 PM)**  1. \*Object-Oriented Programming (OOP) in C#\*  - Classes and objects  - Inheritance  - Polymorphism  - Encapsulation and abstraction  2. \*Error Handling\*  - Exceptions and try-catch blocks  - Custom exceptions  - Using finally for cleanup  **\*Practice Questions:\***  - Explain the differences between .NET Framework and .NET Core.  - What is the Common Language Runtime (CLR)?  - Write a C# program that demonstrates inheritance and polymorphism.  - How do you handle exceptions in C#? | **Day 1: .NET Basics and C# Fundamentals**  **\*Morning (9:00 AM - 12:00 PM)\***  1. \*Introduction to .NET Framework and .NET Core\*  - \*Topics to Cover:\*  - Overview of .NET Framework, .NET Core, and .NET 5/6  - CLR, CTS, and CLS  - Managed vs. unmanaged code  - Key differences between .NET Framework and .NET Core  - \*Practical Questions:\*  - Write a simple console application in .NET Core.  - Create a basic class library in .NET Core and reference it from a console application.  2. \*C# Basics\*  - \*Topics to Cover:\*  - Data types and variables  - Control structures (if, switch, loops)  - Methods and parameters  - \*Practical Questions:\*  - Write a program that calculates the factorial of a number using loops.  - Implement a method that takes an array of integers and returns the maximum value.  **\*Afternoon (1:00 PM - 5:00 PM)\***  1. \*Object-Oriented Programming (OOP) in C#\*  - \*Topics to Cover:\*  - Classes and objects  - Inheritance  - Polymorphism  - Encapsulation and abstraction  - \*Practical Questions:\*  - Create a base class Animal with derived classes Dog and Cat. Implement polymorphism to make the animals speak.  G6grtf - Write a program that demonstrates the use of properties and encapsulation.  2. \*Error Handling\*  - \*Topics to Cover:\*  - Exceptions and try-catch blocks  - Custom exceptions  - Using finally for cleanup  - \*Practical Questions:\*  - Write a method that reads an integer from the user and handles invalid input using try-catch.  - Create a custom exception and use it in a practical scenario.  **\*Practice Questions:\***  - Explain the differences between .NET Framework and .NET Core.  - What is the Common Language Runtime (CLR)?  - Write a C# program that demonstrates inheritance and polymorphism.  - How do you handle exceptions in C#? |
| **Day 2: Advanced C# Concepts**  **\*Morning (9:00 AM - 12:00 PM)\***  1. \*Advanced C#\*  - Delegates and events  - Lambda expressions  - LINQ (Language Integrated Query)  2. \*Collections and Generics\*  - List, Dictionary, Queue, Stack  - Generic collections  - Custom generic types  **\*Afternoon (1:00 PM - 5:00 PM)\***  1. \*Asynchronous Programming\*  - Async and await  - Task Parallel Library (TPL)  - Handling async exceptions  2. \*Reflection and Attributes\*  - Using reflection  - Custom attributes  **\*Practice Questions:\***  - Explain delegates and events in C#.  - How do you use LINQ to filter a list of objects?  - Write a program that demonstrates async and await in C#.  - What is reflection and how is it used in .NET? | **Day 2: Advanced C# Concepts and SQL Basics**  **\*Morning (9:00 AM - 12:00 PM)\***  1. \*Advanced C#\*  - \*Topics to Cover:\*  - Delegates and events  - Lambda expressions  - LINQ (Language Integrated Query)  - \*Practical Questions:\*  - Write a program that uses a delegate to perform a calculation (e.g., addition and subtraction).  - Implement a LINQ query to filter and sort a list of objects.  2. \*Collections and Generics\*  - \*Topics to Cover:\*  - List, Dictionary, Queue, Stack  - Generic collections  - Custom generic types  - \*Practical Questions:\*  - Write a program that demonstrates the use of a Dictionary to store and retrieve student grades.  - Create a custom generic class that implements a simple data structure (e.g., a generic stack).  **\*Afternoon (1:00 PM - 5:00 PM)\***  1. \*Asynchronous Programming\*  - \*Topics to Cover:\*  - Async and await  - Task Parallel Library (TPL)  - Handling async exceptions  - \*Practical Questions:\*  - Write an asynchronous method that downloads data from a URL and processes it.  - Implement a simple console application that demonstrates the use of async and await.  2. \*SQL Basics\*  - \*Topics to Cover:\*  - Basic SQL queries (SELECT, INSERT, UPDATE, DELETE)  - Joins (INNER, LEFT, RIGHT, FULL)  - Aggregate functions (SUM, AVG, COUNT, MAX, MIN)  - \*Practical Questions:\*  - Write SQL queries to perform CRUD operations on a sample database.  - Write a query to find the top 5 highest-paid employees from the Employee table.  **\*Practice Questions:\***  - Explain delegates and events in C#.  - How do you use LINQ to filter a list of objects?  - Write a program that demonstrates async and await in C#.  - Write a SQL query to join two tables and retrieve specific columns. |
| **Day 3: .NET Core and ASP.NET Core**  **\*Morning (9:00 AM - 12:00 PM)\***  1. \*.NET Core Fundamentals\*  - Project structure  - Dependency Injection (DI)  - Configuration in .NET Core  2. \*ASP.NET Core Basics\*  - MVC pattern  - Creating a basic MVC application  - Controllers, Views, and Models  **\*Afternoon (1:00 PM - 5:00 PM)\***  1. \*Web APIs in ASP.NET Core\*  - Creating Web APIs  - Routing and Action methods  - Model Binding and Validation  2. \*Middlewares and Filters\*  - Creating custom middleware  - Action filters, result filters, and exception filters  **\*Practice Questions:\***  - Describe the .NET Core project structure.  - How does dependency injection work in .NET Core?  - Write a simple ASP.NET Core MVC application.  - Explain the lifecycle of an ASP.NET Core request. | **Day 3: .NET Core, ASP.NET Core, and AJAX**  **\*Morning (9:00 AM - 12:00 PM)\***  1. \*.NET Core Fundamentals\*  - \*Topics to Cover:\*  - Project structure  - Dependency Injection (DI)  - Configuration in .NET Core  - \*Practical Questions:\*  - Create a .NET Core console application that reads configuration from appsettings.json.  - Implement dependency injection in a .NET Core console application.  2. \*ASP.NET Core Basics\*  - \*Topics to Cover:\*  - MVC pattern  - Creating a basic MVC application  - Controllers, Views, and Models  - \*Practical Questions:\*  - Create a simple ASP.NET Core MVC application with a controller, view, and model.  - Implement form handling and model binding in an ASP.NET Core MVC application.  **\*Afternoon (1:00 PM - 5:00 PM)\***  1. \*Web APIs in ASP.NET Core\*  - \*Topics to Cover:\*  - Creating Web APIs  - Routing and Action methods  - Model Binding and Validation  - \*Practical Questions:\*  - Create a simple ASP.NET Core Web API with CRUD operations.  - Implement validation in your ASP.NET Core Web API using data annotations.  2. \*AJAX Basics\*  - \*Topics to Cover:\*  - What is AJAX and how it works  - Making AJAX requests using JavaScript/jQuery  - Handling AJAX responses  - \*Practical Questions:\*  - Write an AJAX call to fetch data from an ASP.NET Core Web API.  - Implement an AJAX-based form submission in an ASP.NET Core MVC application.  **\*Practice Questions:\***  - Describe the .NET Core project structure.  - How does dependency injection work in .NET Core?  - Write a simple ASP.NET Core MVC application.  - Explain the lifecycle of an ASP.NET Core request.  - What is AJAX and how do you use it in a web application? |
| **Day 4: Entity Framework Core and Advanced ASP.NET Core**  **\*Morning (9:00 AM - 12:00 PM)\***  1. \*Entity Framework Core\*  - Code-first vs. database-first  - Creating models and DbContext  - CRUD operations  2. \*EF Core Migrations\*  - Adding and applying migrations  - Handling model changes  **\*Afternoon (1:00 PM - 5:00 PM)\***  1. \*Authentication and Authorization\*  - ASP.NET Core Identity  - JWT Authentication  - Policy-based authorization  2. \*Advanced Topics in ASP.NET Core\*  - SignalR for real-time communication  - gRPC services  **\*Practice Questions:\***  - Explain the differences between code-first and database-first approaches in EF Core.  - How do you handle migrations in EF Core?  - Describe how JWT authentication works in ASP.NET Core.  - What is SignalR and how is it used? | **Day 4: Entity Framework Core, Advanced ASP.NET Core, and SQL Advanced**  **\*Morning (9:00 AM - 12:00 PM)\***  1. \*Entity Framework Core\*  - \*Topics to Cover:\*  - Code-first vs. database-first  - Creating models and DbContext  - CRUD operations  - \*Practical Questions:\*  - Create an EF Core model and DbContext for a simple application.  - Implement CRUD operations in an ASP.NET Core application using EF Core.  2. \*EF Core Migrations\*  - \*Topics to Cover:\*  - Adding and applying migrations  - Handling model changes  - \*Practical Questions:\*  - Add a new property to an existing EF Core model and create a migration to update the database.  - Implement a method to seed initial data using EF Core migrations.  **\*Afternoon (1:00 PM - 5:00 PM)\***  1. \*Authentication and Authorization\*  - \*Topics to Cover:\*  - ASP.NET Core Identity  - JWT Authentication  - Policy-based authorization  - \*Practical Questions:\*  - Implement ASP.NET Core Identity for user registration and login.  - Secure an ASP.NET Core Web API using JWT authentication.  2. \*Advanced SQL\*  - \*Topics to Cover:\*  - Subqueries and CTEs (Common Table Expressions)  - Transactions  - Indexes and performance tuning  - \*Practical Questions:\*  - Write a SQL query using a CTE to find hierarchical data.  - Implement a transaction in SQL to update multiple tables.  **\*Practice Questions:\***  - Explain the differences between code-first and database-first approaches in EF Core.  - How do you handle migrations in EF Core?  - Describe how JWT authentication works in ASP.NET Core.  - Write a SQL query to create an index on a table and explain its benefits. |
| **Day 5: Design Patterns, Best Practices, and Mock Interviews**  **\*Morning (9:00 AM - 12:00 PM)\***  1. \*Design Patterns\*  - Singleton, Factory, Repository  - Dependency Injection  - CQRS (Command Query Responsibility Segregation)  2. \*Best Practices\*  - SOLID principles  - Clean Code practices  - Testing (Unit Tests and Integration Tests)  **\*Afternoon (1:00 PM - 5:00 PM)\***  1. \*Mock Interviews\*  - Conduct mock interviews with a peer or mentor  - Review and refine answers based on feedback  2. \*Review and Practice\*  - Go through all previous topics  - Practice coding problems on platforms like LeetCode or HackerRank  **\*Practice Questions:\***  - Explain the Singleton pattern and provide an example.  - What are SOLID principles?  - How do you write unit tests for a service in ASP.NET Core?  - Conduct a mock interview and review feedback. | **Day 5: Design Patterns, Best Practices, Mock Interviews, and Review**  **\*Morning (9:00 AM - 12:00 PM)\***  1. \*Design Patterns\*  - \*Topics to Cover:\*  - Singleton, Factory, Repository  - Dependency Injection  - CQRS (Command Query Responsibility Segregation)  - \*Practical Questions:\*  - Implement the Singleton pattern in C#.  - Create a simple repository pattern implementation for data access.  2. \*Best Practices\*  - \*Topics to Cover:\*  - SOLID principles  - Clean Code practices  - Testing (Unit Tests and Integration Tests)  - \*Practical Questions:\*  - Refactor a piece of code to adhere to SOLID principles.  - Write unit tests for a service using a mocking framework like Moq.  **\*Afternoon (1:00 PM - 5:00 PM)\***  1. \*Mock Interviews\*  - \*Activity:\*  - Conduct mock interviews with a peer or mentor.  - Review and refine answers based on feedback.  2. \*Review and Practice\*  - \*Activity:\*  - Go through all previous topics.  - Practice coding problems on platforms like LeetCode or HackerRank.  **\*Practice Questions:\***  - Explain the Singleton pattern and provide an example.  - What are SOLID principles?  - How do you write unit tests for a service in ASP.NET Core?  - Conduct |
| **General Interview Questions**  - Describe the difference between an interface and an abstract class in C#.  - What is the difference between IEnumerable and IQueryable?  - How does garbage collection work in .NET?  - Explain the concept of dependency injection and how it is implemented in .NET Core.  - What are the benefits of using Entity Framework Core?  - How do you handle concurrency in a multi-threaded application? | **Additional Practice Questions**  \*General Interview Questions:\*  - Describe the difference between an interface and an abstract class in C#.  - What is the difference between IEnumerable and IQueryable?  - How does garbage collection work in .NET?  - Explain the concept of dependency injection and how it is implemented in .NET Core.  - What are the benefits of using Entity Framework Core?  - How do you handle concurrency in a multi-threaded application?  - Write an AJAX call to fetch data from an ASP.NET Core Web API and display it in a web page.  - Write a SQL query to find employees who have not submitted any timesheets in the last month. |

# SQL and Database:

Practical

1. Walk me through the process of creating a stored procedure in SQL Server and explain its advantages.

2. How would you handle concurrency issues in a multi-user SQL Server environment?

3. Give an example of using indexes to improve query performance.

4. Explain the process of backing up and restoring a SQL Server database.

5. Demonstrate how you would write a SQL query to retrieve the second highest salary from an Employee table.

|  |  |
| --- | --- |
| SELECT salary FROM Employee  ORDER BY salary  OFFSET 1 ROW  FETCH NEXT 1 ROW ONLY; | Select max(salary) from employee  Where  salary < (  SELECT MAX(salary)  FROM Employee |
| WITH RankedEmployees AS (  SELECT \*,  ROW\_NUMBER() OVER(ORDER BY salary) AS Row\_Number  FROM Employee  )  SELECT \*  FROM RankedEmployees  WHERE Row\_Number = 2; | with cte as  (select row\_number() over(order by salary) as rownumber desc from employee)  select salary from employee where rownumber=2 |

6. Discuss techniques for optimizing database performance, including index maintenance and query tuning.

7. How would you implement paging in an SQL query to retrieve a subset of records?

8. Describe the steps you would take to troubleshoot a database connectivity issue.

9. What are the different types of triggers in SQL Server, and when would you use them?

10. How would you handle error handling and logging in a SQL Server stored procedure?

Theory

1. Describe the difference between clustered and non-clustered indexes. When would you use each?

2. What are the different types of triggers in SQL Server, and when would you use them?

3. Discuss the concept of transactions and their role in ensuring data integrity.

4. Explain the purpose of the SQL Server Agent and its common use cases.

5. Describe the process of database normalization and its benefits.

6. How would you approach database design for a new application, considering scalability and performance?

7. Explain the concept of referential integrity and how it is enforced in SQL Server.

8. What are SQL injection attacks, and how can they be prevented?

9. Discuss the role of execution plans in SQL Server query optimization.

10. How would you monitor and manage SQL Server performance?

LOGICAL

‘1. Explain the concept of normalization and its importance in database design.

2. How would you optimize a slow-performing SQL query?

3. Can you describe the differences between INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL JOIN?

4. Discuss the ACID properties of transactions and their significance in database management.

5. What are the differences between a primary key, a unique key, and a foreign key constraint?

6. How would you handle database security in SQL Server? Discuss roles, permissions, and encryption.

7. Explain the concept of a deadlock in SQL Server and how you would prevent or resolve it.

8. Describe the various types of joins in SQL Server and provide examples of when each would be used.

9. Explain the concept of normalization forms (1NF, 2NF, 3NF) and give examples of each.

10. What are the advantages and disadvantages of using stored procedures versus ad hoc SQL statements?