ASSIGNMENT - I

INTRODUCTION TO APPLICATION DEVELOPMENT WITH C#

21071A05A9 P. Sai Kumar

UNIT-1:

Short Questions:

1) **Explain switch, cables and connectors, hub, Router, Modem?**

Answer:

Switch:

Switch is a networking device that groups all the devices over the network to transfer the data to another device. A switch is better than Hub as it does not broadcast the message over the network.

Cables and Connectors:

Cable is a transmission media that transmits the communication signals. There are three types:

- Twisted Pair Cable
- Coaxial Cable
- Fibre Optic Cable

Hub:

Hub is a central device that splits the network connection into multiple devices. When computer requests for information from a computer, it sends the request to the Hub. Hub distributes this request to all the interconnected computers.

Router:

Router is a device that connects the LAN to the internet. The router is mainly used to connect the distinct networks or connect the internet to multiple computers.

Modem:

Modem connects the computer to the internet over the existing telephone line. A modem is not integrated with the computer motherboard. A modem is a separate part on the PC slot found on the motherboard.

2) **What are Types of Networks?**
Answer:

The types of Networks are:

1. Local Area Network (LAN)
2. Metropolitan Area Network (MAN)
3. Wide Area Network (WAN)

3) **What is Network Addressing?**
Answer:

Network Addressing is a fundamental concept in computer networking that involves assigning unique identifiers to devices and systems on a network to enable communication. There are two types:

- IP Addressing
- MAC Addressing

4) **What is Protocol Stack?**
Answer:

A protocol stack is a group of protocols that all work together to allow software or hardware to perform a function. The TCP/IP protocol stack is a good example. It uses four layers that map to the OSI model.

5) **What is Software Engineering?**
Answer:

The term software engineering is the product of two words, software, and engineering. The software is a collection of integrated programs. Engineering is the application of scientific and practical knowledge to invent, design, build, maintain, and improve frameworks, processes, etc.

6) **Why is Software Engineering required?**
Answer:

Software Engineering is required due to the following reasons:

- To manage large software

•For more Scalability (growing amount of work by adding resources to the system.)

•Cost Management

•To manage the dynamic nature of software

•For better quality Management

7) Need of Software Engineering?

Answer:

Need for Software Engineering:

- Software engineering is essential due to rapidly evolving user requirements and the operating environment.
- Just as building a house is more complex than a wall, extensive programming demands a scientific approach. Without this foundation, creating new software is easier than scaling existing solutions.
- While hardware costs decrease, software expenses remain high without proper processes.
- Software's dynamic nature hinges on the evolving client environment, necessitating constant updates for quality management and a superior end product.

8) Define Objects and Classes in C#.

Answer:

Objects:

Object is a basic unit of Object-Oriented Programming and represents the real-life entities. A typical C# program creates many objects, which as you know, interact by invoking methods.

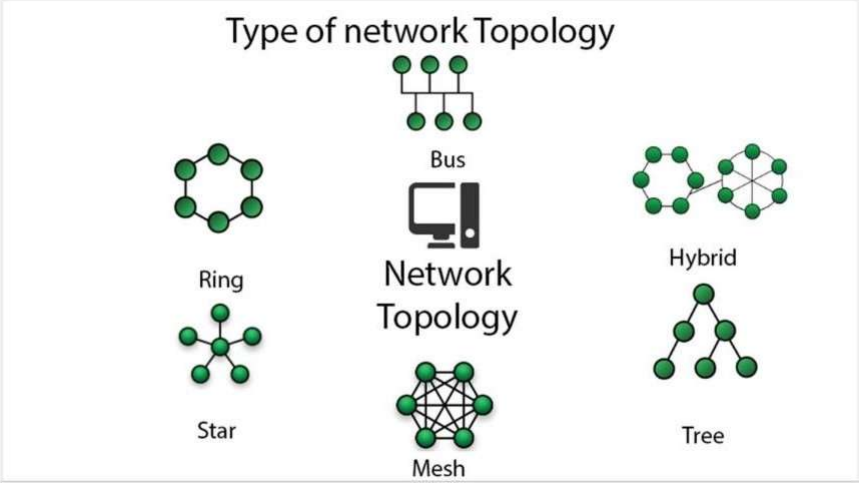
Classes:

A class is a user-defined blueprint or prototype from which objects are created. Basically, a class combines the fields and methods (member function which defines actions) into a single unit.

Long Questions:

1) **What is Topology? Types Of Topologies?**

Answer:



1. Bus Topology:

The simplest type of topology is called a bus topology, in which network communication takes place over a single bus or channel. There are numerous taps and drop lines connected to the bus. Drop Lines are cables that connect the bus to the computer, and taps are connectors. In other words, each node is connected to a single transmission line.

Advantages of Bus Topology:

- Low-cost cable: In bus topology, nodes are directly connected to the cable without passing through a hub. Therefore, the initial cost of installation is low.
- Moderate data speeds: Coaxial or twisted pair cables are mainly used in bus-based networks that support up to 10 Mbps.
- Familiar technology: Bus topology is a familiar technology as the installation and troubleshooting techniques are well known, and hardware components are easily available.
- Limited failure: If Cable Fails then complete network fails.

Disadvantages of Bus Topology:

- Extensive cabling: A bus topology is quite simpler, but still, it requires a lot of cabling.
- Difficult troubleshooting: It requires specialized test equipment to determine the cable faults. If any fault occurs in the cable, then it would disrupt the communication for all the nodes.
- Signal interference: If two nodes send the messages simultaneously, then the signals of both the nodes collide with each other.

- Reconfiguration difficult: Adding new devices to the network would slow down the network.

2. Ring Topology:

When two computers are connected to form a ring, the topology is known as a ring topology. The message passing is circular and unidirectional. A fixed amount of time is allotted for each computer to access the network for transmission in this deterministic network topology. Each node is a part of a closed loop.

Advantages of Ring Topology:

- Network Management: Faulty devices can be removed from the network without bringing the network down.
- Product availability: Many hardware and software tools for network operation and monitoring are available.
- Cost: Twisted pair cabling is inexpensive and easily available. Therefore, the installation cost is very low.
- Reliable: It is a more reliable network because the communication system is not dependent on the single host computer.
- No collisions (Because of unidirectional)
- Easy to fault Tolerance (If one System is disconnected, we can easily identify).

Disadvantages of Ring Topology:

- Difficult troubleshooting: It requires specialized test equipment to determine the cable faults. If any fault occurs in the cable, then it would disrupt the communication for all the nodes.
- Failure: The breakdown in one station leads to the failure of the overall network.
- Reconfiguration difficult: Adding new devices to the network would slow down the network.
- Delay: Communication delay is directly proportional to the number of nodes. Adding new devices increases the communication delay.
- If any system Fails Transmission fails
- Less security
- Difficult to reconfigure

3. Star Topology:

A computer network topology known as a star topology connects each node to a central hub. The hub or switch acts as a bridge between the nodes. Any node making a service request or offering a service must first get in touch with the hub. The other connected devices function as clients in a star topology, w hile the hub and switch serve as a server.

Advantages of Star Topology:

•Limited failure: As each station is connected to the central hub with its own cable, therefore failure in one cable will not affect the entire network.
•Familiar technology: Star topology is a familiar technology as its tools are cost-effective.
•Easily expandable: It is easily expandable as new stations can be added to the open ports on the hub.
•Cost effective: Star topology networks are cost-effective as it uses inexpensive coaxial cable.
•High data speeds: It supports a bandwidth of approx. 100Mbps. Ethernet 100BaseT is one of the most popular Star topology networks.
•If the centralized device is switch then there will be security.
•Less expensive because we are using less no of cables.

Disadvantages of Star Topology:

•A Central point of failure: If the central hub or switch goes down, then all the connected nodes will not be able to communicate with each other.
•Cable: Sometimes cable routing becomes difficult when a significant amount of routing is required.

4. Mesh Topology:

Mesh technology is a network configuration in which computers are linked to one another by numerous redundant connections. There are numerous methods for transferring from one computer to another. It lacks a switch, hub, or any other central computer that acts as a communication hub.

Mesh is divided into two types:

- Fully connected mesh topology
- Partially connected mesh topology

Advantages of Mesh Topology:

- Reliable: The mesh topology networks are very reliable as if any link breakdown will not affect the communication between connected computers.
- Fast Communication: Communication is very fast between the nodes.
- Easier Reconfiguration: Adding new devices would not disrupt the communication between other devices.
- If any system fails transmission will not fail.

Disadvantages of Mesh Topology:

- Cost: A mesh topology contains a large number of connected devices such as a router and more transmission media than other topologies.
- Management: Mesh topology networks are very large and very difficult to maintain and manage. If the network is not monitored carefully, then the communication link failure goes undetected.
- Efficiency: In this topology, redundant connections are high that reduces the efficiency of the network.

5. Tree Topology:

   A computer network topology known as a "tree topology" is one in which all nodes are either directly or indirectly connected to the main bus cable. Bus and Star topologies are combined to create tree topology. With a tree topology, the network is split up into manageable segments that can be easily maintained.

   Advantages of Tree Topology:

   - Support for broadband transmission: Tree topology is mainly used to provide broadband transmission, i.e., signals are sent over long distances without being attenuated.
   - Easily expandable: We can add the new device to the existing network. Therefore, we can say that tree topology is easily expandable.
   - Easily manageable: In tree topology, the whole network is divided into segments known as star networks which can be easily managed and maintained.
   - Error detection: Error detection and error correction are very easy in a tree topology.
   - Limited failure: The breakdown in one station does not affect the entire network.
   - Point-to-point wiring: It has point-to-point wiring for individual segments.

   Disadvantages of Tree Topology:

   - Difficult troubleshooting: If any fault occurs in the node, then it becomes difficult to troubleshoot the problem.
   - High cost: Devices required for broadband transmission are very costly.
   - Failure: A tree topology mainly relies on main bus cable and failure in main bus cable will damage the overall network.
   - Reconfiguration difficult: If new devices are added, then it becomes difficult to reconfigure.

6. Hybrid Topology:
   A hybrid topology is a computer topology made up of two or more topologies. All topologies in this topology are interconnected based on their needs to form a hybrid.
   Advantages of Hybrid Topology:
   - Reliable: If a fault occurs in any part of the network will not affect the functioning of the rest of the network.
   - Scalable: Size of the network can be easily expanded by adding new devices without affecting the functionality of the existing network.
   - Flexible: This topology is very flexible as it can be designed according to the requirements of the organization.
   - Effective: Hybrid topology is very effective as it can be designed in such a way that the strength of the network is maximized and weakness of the network is minimized.

   Disadvantages of Hybrid Topology:

   - Complex design: The major drawback of the Hybrid topology is the design of the Hybrid network. It is very difficult to design the architecture of the Hybrid network.
   - Costly Hub: The Hubs used in the Hybrid topology are very expensive as these hubs are different from usual Hubs used in other topologies.
   - Costly infrastructure: The infrastructure cost is very high as a hybrid network requires a lot of cabling, network devices, etc.

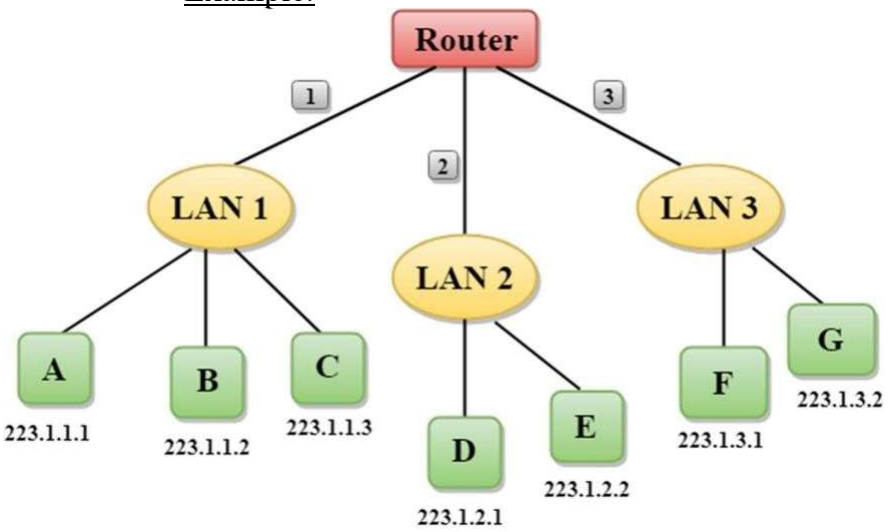
2) Network Addressing

Answer:

- Network Addressing is one of the major responsibilities of the network layer.
- Network addresses are always logical, i.e., software-based addresses.
- A host is also known as end system that has one link to the network. The boundary between the host and link is known as an interface. Therefore, the host can have only one interface.
- A router is different from the host in that it has two or more links that connect to it. When a router forwards the datagram, then it

forwards the packet to one of the links. The boundary between the router and link is known as an interface, and the router can have multiple interfaces, one for each of its links. Each interface can send and receive the IP packets, so IP requires each interface to have an address.

- Each IP address is 32 bits long, and they are represented in the form of "dot-decimal notation" where each byte is written in the decimal form, and they are separated by the period. An IP address would look like 193.32.216.9 where 193 represents the decimal notation of first 8 bits of an address, 32 represents the decimal notation of second 8 bits of an address.
- Example:



• In the above figure, a router has three interfaces labelled as 1, 2 & 3 and each router interface contains its own IP address.

- Each host contains its own interface and IP address.
- All the interfaces attached to the LAN 1 is having an IP address in the form of 223.1.1.xxx, and the interfaces attached to the LAN 2 and LAN 3 have an IP address in the form of 223.1.2.xxx and 223.1.3.xxx respectively.
- Each IP address consists of two parts. The first part (first three bytes in IP address) specifies the network, and second part (last byte of an IP address) specifies the host in the network.

3) SDLC – Software Development Life Cycle

Answer:

Software Development Life Cycle:

- SDLC stands for Software Development Life Cycle.
- It is a process that gives a complete idea about developing, designing, and maintaining a software project by ensuring that all the functionalities along with user requirements, objectives, and end goals are addressed.
- With SDLC, the software project's quality and the overall software development process get enhanced.

For any software project, SDLC offers the following benefits

•With SDLC, one can address the goals and problems so that the project is implemented with the highest precision and accuracy

•In SDLC, the project members cannot proceed ahead before completion & approval of the prior stages

•Any installation in the project that has been executed using the SDLC has necessary checks so that it is tested with precision before entering the installation stage

•With a well-defined SDLC in place, project members can continue the software development process without incurring any complications

•SDLC offers optimal control with minimum problems, allowing the project members to run the project smoothly.

Stage 1: Requirement Gathering & Analysis Phase

In an SDLC, this is the first and most crucial phase for a software project's success. In this phase, communication takes place

between stakeholders, end-users, and project teams, as both functional and non-functional requirements are gathered from customers. The Requirement Gathering & Analysis Phase of SDLC involves the following:

1. Analysis of functionality and financial feasibility
2. Identifying and capturing requirements of stakeholders through customer interactions like interviews, surveys, etc.
3. Clearly defining and documenting customer requirements in an SRS (Software Resource Specification Document) comprising of all product requirements that need to be developed
4. Creating project prototypes to show the end-user how the project will look

Stage 2: Design Phase

In the design phase of an SDLC, the architectural design is proposed for the project based on the SRS Document requirements. The Designing Phase of SDLC involves the following:

1. Separation of hardware and software system requirements
2. Designing the system architecture based on gathered requirements
3. Creating Unified Modelling Language (UML) diagrams like- use cases, class diagrams, sequence diagrams, and activity diagrams.

Stage 3: Development Phase

In the entire SDLC, the development phase is the longest one. In this phase, the actual project is developed and built. The Development Phase of SDLC involves the following

1. Actual code is written.
2. Demonstration of accomplished work presented before a Business Analyst for further modification of work.
3. Unit testing is performed, i.e., verifying the code based on requirements.

Stage 4: Testing Phase

Almost all stages of SDLC involves the testing strategy. However, SDLC's testing phase refers to checking, reporting, and fixing the system for any bug/defect.

In this phase, the on-going system or project is migrated to a test environment where different testing forms are performed. This testing continues until the project has achieved the quality standards, as mentioned in the SRS document during the requirement gathering phase.

The Testing Phase involves the following-

1. Testing the system as a whole
2. Performing different types of tests in the system.
3. Reporting and fixing all forms of bugs & defects.

Stage 5: Deployment & Maintenance Phase

In this SDLC phase, once the system testing has been done, it is ready to be launched.

The system may be initially released for limited users by testing it in a real business environment for UAT (User Acceptance Testing).

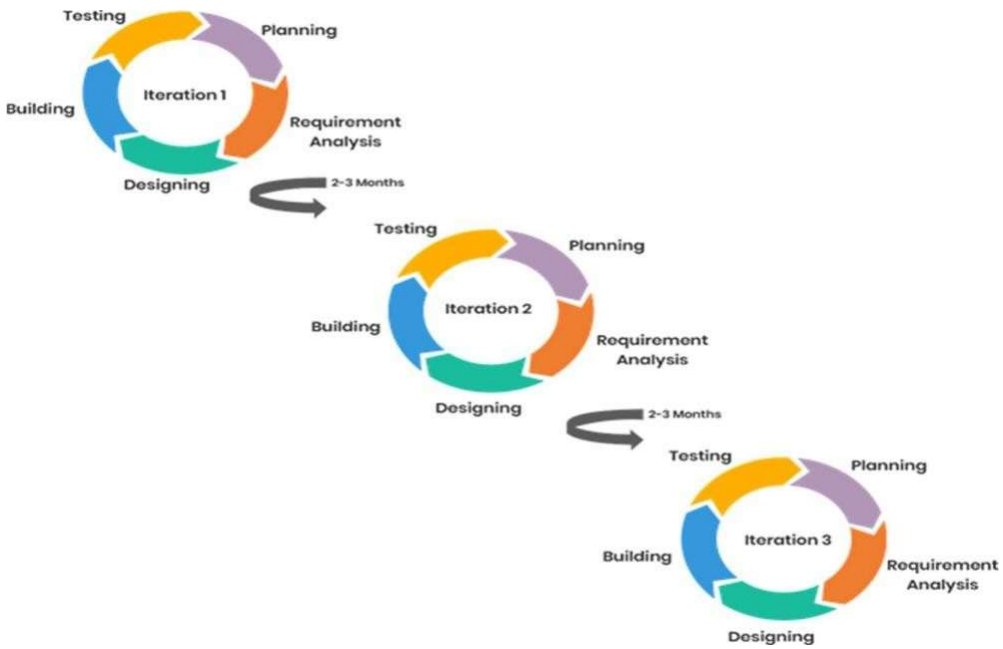
The Deployment & Maintenance Phase involves the following-

1. The system is ready for delivery
2. The system is installed and used
3. Errors are rectified that might have been previously missed
4. Enhancing the system inside a data center.

4) Agile Model

Answer:

1. The agile model is the combination of the iterative-incremental model that depends on process adaptability along with customer satisfaction through the delivery of software products.
2. In this model, the project is broken down into smaller time frames for delivering certain features during a release.



3. Strengths of the Agile Model:
   •Easy to accommodate changing requirements.

   •Regular communication takes place between customers and developers.

   •Functionalities can be developed quickly and demonstrated to customers.

4. Weakness of the Agile Model:
   •Not ideal for handling complex dependencies.

   •Teams need to have the desired experience levels for adhering method rules.

5) OOP - Object Oriented Concepts in C#

Answer:

Object Oriented Concepts:

- Object
- Class
- Encapsulation
- Abstraction
- Polymorphism
- Inheritance

Object:

- It is a basic unit of Object-Oriented Programming and represents the real-life entities. A typical C# program creates many objects, which as you know, interact by invoking methods. An object consists of:
  - State: It is represented by attributes of an object. It also reflects the properties of an object.
  - Behaviour: It is represented by methods of an object. It also reflects the response of an object with other objects.
  - Identity: It gives a unique name to an object and enables one object to interact with other objects.

Class:

- A class is a user-defined blueprint or prototype from which objects are created. Basically, a class combines the fields and methods (member function which defines actions) into a single unit.
- Generally, a class declaration contains only keyword class, followed by an identifier(name) of the class.
- But there are some optional attributes that can be used with class declaration according to the application requirement.
- In general, class declarations can include these components, in order:
  - Modifiers
  - Keyword Class
  - Class Identifier
  - Base Class or Super Class
  - Interfaces
  - Body
  - Constructors
- When an object of a class is created, the class is said to be instantiated.
- All the instances share the attributes and the behaviour of the class.
- But the values of those attributes, i.e., the state are unique for each object.
- A single class may have any number of instances.

Encapsulation:

- Encapsulation in object-oriented programming (OOP) is a fundamental concept that allows you to bundle data (attributes) and methods (functions) that operate on that data into a single unit, known as a class. This unit is like a black box where the internal details of the class are hidden from the outside and can only be accessed through well-defined interfaces (methods).

- Example:

```
public class Student {
    private String name;
    private int age;
    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        if (age >= 0 && age <= 150) {
            this.age = age;
        }
    }
    public static void main(String[] args) {
        Student student1 = new Student("Alice", 20);
        System.out.println("Name: " + student1.getName());
        System.out.println("Age: " + student1.getAge());
        student1.setAge(25);
        System.out.println("New Age: " + student1.getAge());
        student1.setName("Bob");
        System.out.println("New Name: " + student1.getName());
    }
}
```

Abstraction:

- We can create interfaces or abstract classes and implement them in classes when we need similar functionality. This approach is recommended. Abstraction, on the other hand, is about designing the application on paper or UML diagrams, identifying common object behaviours, and implementing them with abstract classes or interfaces. Abstraction doesn't primarily involve hiding data; instead, it focuses on generalizing object behaviours, ensuring a high-level overview of their functionality.

- Example:

```
abstract class Shape {
    public abstract double calculateArea();
    public void printShape() {
        System.out.println("This is a shape.");
    }
}

class Circle extends Shape {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    @Override
    public double calculateArea() {
        return Math.PI * radius * radius;
    }
}

class Rectangle extends Shape {
    private double width;
    private double height;

    public Rectangle(double width, double height) {
        this.width = width;
        this.height = height;
    }

    @Override
    public double calculateArea() {
        return width * height;
    }
}

public class AbstractionExample {
    public static void main(String[] args) {
        Shape circle = new Circle(5.0);
        Shape rectangle = new Rectangle(4.0, 6.0);

        circle.printShape();
        System.out.println("Circle Area: " + circle.calculateArea());
```

```
        rectangle.printShape();
        System.out.println("Rectangle Area: " +
rectangle.calculateArea());
    }
}
```

Polymorphism:

- Polymorphism is the ability of objects to take on multiple forms. In Java, this primarily refers to the ability of a reference variable to refer to different types of objects. There are two types of polymorphism: compile-time (method overloading) and runtime (method overriding). Method overriding is particularly important for achieving runtime polymorphism, which is based on inheritance.
- Here's an example of runtime polymorphism:

```
Animal myAnimal = new Dog();
myAnimal.makeSound();
```

Inheritance:

- Inheritance is a mechanism that allows one class (subclass or derived class) to inherit the properties and behaviors (fields and methods) of another class (superclass or base class). This promotes code reuse and the creation of a hierarchical relationship between classes. In Java, the extends keyword is used to establish inheritance.
- For example:

```
class Animal {
    void makeSound() {
        System.out.println("Some generic sound");
    }
}

class Dog extends Animal {
    void makeSound() {
        System.out.println("Bark");
    }
}
```

6) Networking Levels and Layers and Protocols

Answer:

In networking, there are various levels, layers, and protocols that work together to facilitate communication between devices in a structured and organized manner.

1. Networking Levels:
   - Physical Level: This level deals with the actual physical hardware components of a network. It includes cables, switches, routers, and network interfaces. The physical level is responsible for transmitting raw binary data over the physical medium.
   - Data Link Level: The data link level manages communication between directly connected devices over a physical link or network segment. It handles error detection and correction, as well as the framing and addressing of data.
   - Network Level: The network level is responsible for routing data packets between different networks. It determines the best path for data to travel from the source to the destination. The Internet Protocol (IP) operates at this level.
   - Transport Level: This level ensures end-to-end communication between devices. It is responsible for data integrity, reliability, and flow control. Protocols like TCP and UDP operate at the transport level.
   -Session Level: The session level establishes, maintains, and terminates communication sessions between devices. It can manage multiple conversations simultaneously and ensures data integrity during these sessions.
   - Presentation Level: The presentation level handles data translation, encryption, and compression. It ensures that data is presented in a format that both the sender and receiver can understand.
   - Application Level: The application level directly interacts with end-user applications and provides a platform-independent interface for software services. It includes protocols for services like email, web browsing, and file transfer.

2. Networking Layers:
   - Networking layers refer to the hierarchical organization of networking functions based on the OSI (Open Systems Interconnection) model. The OSI model defines seven layers, as mentioned in the previous response.

3. Protocols:
   - Protocols are sets of rules and conventions that govern how data is transmitted, received, and processed in a network. Different layers in the OSI model have their own associated protocols.
   - For example, at the network layer, the Internet Protocol (IP) is a key protocol responsible for addressing and routing packets. At the transport

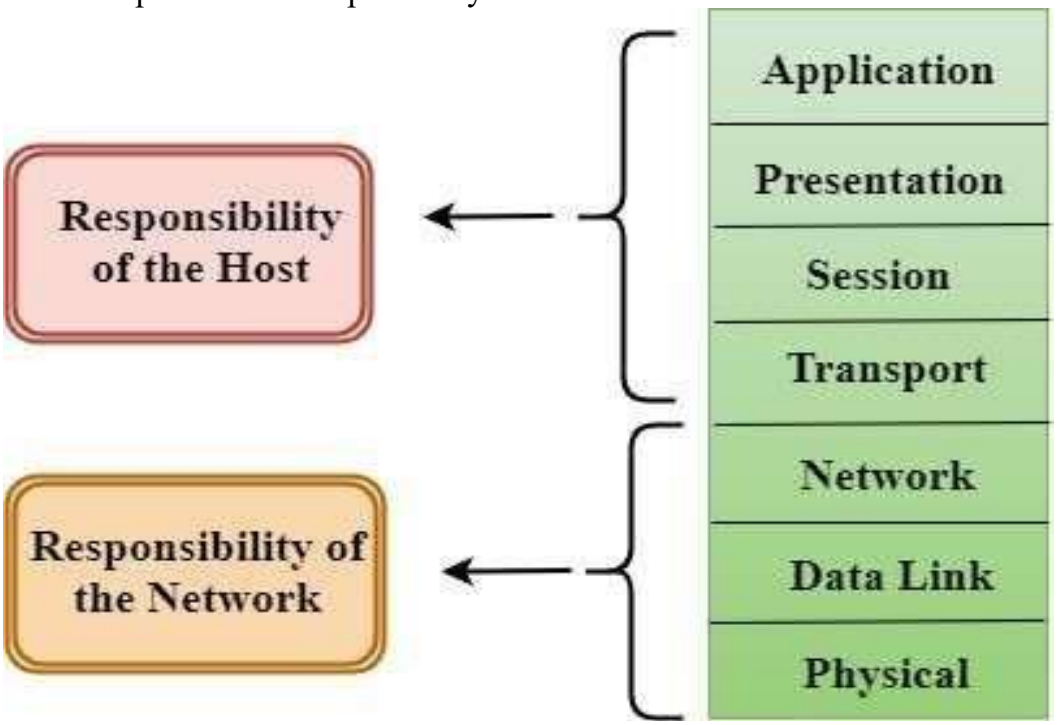
layer, protocols like TCP and UDP provide reliable or connectionless data transmission, respectively.

- In addition to these, there are numerous other protocols at various layers, including HTTP for web communication, SMTP for email, and Ethernet for data link layer communication.

7) OSI Model

Answer:

- OSI stands for Open System Interconnection is a reference model that describes how information from a software application in one computer moves through a physical medium to the software application in another computer.
- OSI consists of seven layers, and each layer performs a particular network function.
- OSI model was developed by the International Organization for Standardization (ISO) in 1984, and it is now considered as an architectural model for the inter-computer communications.
- OSI model divides the whole task into seven smaller and manageable tasks. Each layer is assigned a particular task.
- Each layer is self-contained, so that task assigned to each layer can be performed independently.



- The OSI model comprises two layers: upper and lower.

- The upper layer, closest to end-users, handles application-related concerns solely in software. It interacts with software applications and users. A layer above another is referred to as an upper layer.
- In contrast, the lower layer manages data transport issues, combining hardware and software. The physical layer, the lowest OSI layer and closest to the physical medium, is primarily responsible for placing data on it.
- The seven layers of OSI Model:

  - Application Layer (Layer 7):

    The top layer is the Application Layer, which directly interacts with end-user applications and provides a platform-independent interface for software services.
    It includes protocols for services like email (SMTP), web browsing (HTTP), and file transfer (FTP).

  - Presentation Layer (Layer 6):

    The Presentation Layer is responsible for data translation, encryption, and compression. It ensures that data is presented in a format that both the sender and receiver can understand.
    Examples include data encryption and character encoding conversion.

  - Session Layer (Layer 5):

    The Session Layer manages, establishes, maintains, and terminates communication sessions between two devices.
    It can handle multiple conversations simultaneously and ensures data integrity during these sessions.

  - Transport Layer (Layer 4):

    The Transport Layer is responsible for end-to-end communication between devices. It ensures data integrity, reliability, and flow control.
    Protocols like TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) operate at this layer.

  - Network Layer (Layer 3):

The Network Layer handles routing and forwarding of data packets between different networks. It determines the best path for data to travel from the source to the destination.
The Internet Protocol (IP) operates at this layer.

- Data Link Layer (Layer 2):

The Data Link Layer manages the flow of data between directly connected nodes over a physical link or network segment.
It is responsible for error detection and correction, as well as the framing and addressing of data.

- Physical Layer (Layer 1):

The Physical Layer deals with the physical medium and transmission of raw binary data over it. It specifies the electrical, mechanical, and functional characteristics of the hardware. This layer includes details about cables, switches, and physical topologies.

Unit 2:

Short Questions:

1) **Difference between World Wide Web and Internet?**
   Answer:
   - Internet is entirely different from WWW. It is a worldwide network of devices like computers, laptops, tablets, etc.
   - It enables users to send emails to other users and chat with them online.
   - For example, when you send an email or chatting with someone online, you are using the internet
   - But, when you have opened a website like google.com for information, you are using the World Wide Web; a network of servers over the internet.
   - You request a webpage from your computer using a browser, and the server renders that page to your browser.

2) **What is IOT (Internet of Things)?**
   Answer:
- Connecting everyday things embedded with electronics, software, and sensors to internet enabling to collect and exchange data without human interaction called as the Internet of Things (IoT).

3) **What are Semantic Elements?**

Answer:

A semantic element clearly describes its meaning to both the browser and the developer.

Examples of **non-semantic** elements:

 and - Tells nothing about its content.

Examples of **semantic** elements:

<form>, <table>, and <article> - Clearly defines its content.

4) **Syntax of CSS?**

Answer:

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in the document.

CSS syntax as follows:

selector {property: value}

5) **Write any 5 properties of CSS?**

Answer:

Five properties of CSS:

- Color (color): It defines the text and background color of elements.
- Font (font-family, font-size): Specifies the typeface and size of text.
- Margin and Padding (margin, padding): Controls the spacing around and within elements.
- Border (border): Sets the border around an element, including width, style, and color.
- Position (position): Determines the layout and positioning of elements on the page, such as absolute, relative, or fixed positioning

6) **HTML 5 Semantic elements?**

Answer:

HTML 5 Semantic elements:

- <article>
- <footer>
- <section>
- <aside>

- <time>
- <header>
- <details>
- <main>
- <summary>
- <mark>
- <nav>
- <figcaption>
- <figure>

7) **What is element and attribute?**

Answer:

Element: An HTML element is a component of an HTML document that tells a web browser how to structure and interpret a part of the HTML document.

Attribute: An HTML attribute is a piece of markup language used to adjust the behavior or display of an HTML element. For example, attributes can be used to change the color, size, or functionality of HTML elements.

8) **What is Cell Padding?**

Answer:

Cellpadding represents the distance between cell borders and the content within a cell. This attribute is used to adjust the white space in the table cells.

Long Questions:

1) **HTML5 Semantic Elements and explain each element briefly?**

Answer:

Semantic HTML elements are those that clearly describe their meaning in a human- and machine-readable way.

- <article>: Defines an article
- <footer>: Defines a footer for a document or section
- <section>: Defines a section in a document
- <aside>: Defines content aside from the page content
- <time>: Defines a date/time
- <header>: Specifies a header for a document or section
- <details>: Defines additional details that the user can view or hide
- <main>: Specifies the main content of a document

- <summary>: Defines a visible heading for a <details> element
- <mark>: Defines marked/highlighted text
- <nav>: Defines navigation links
- <figcaption>: Defines a caption for a <figure> element
- <figure>: Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.

2) List, Tables, images, forms

Answer:

Lists:

Lists must contain one or more list elements. Lists may contain −

- − An unordered list. This will list items using plain bullets.
- − An ordered list. This will use different schemes of numbers to list items.
- <dl> − A definition list. This arranges your items in the same way as they are arranged in a dictionary.

Tables:

- The HTML tables allow web developers to arrange data like text, images, links, other tables, etc. into rows and columns of cells.
- The <table> tag creates HTML table .
- The <tr> tag is used to create table rows and <td> tag is used to create data cells.
- The three elements for separating the head, body, and foot of a table are −
  - ❖ <thead> − to create a separate table header.
  - ❖ <tbody> − to indicate the main body of the table.
  - ❖ <tfoot> − to create a separate table footer

Images:

- You can insert any image in your web page by using <img> tag.
- Set Image Width/Height
  - ❖ Use width and height attributes.
  - ❖ specify width and height of the image in terms of either pixels or percentage of its actual size.
  - ❖ By default, image will have a border around it, A thickness of 0 means, no border around the picture.
- Image Alignment
  - ❖ By default, image will align at the left side of the page, use align attribute to set it in the center or right.

Forms:

HTML Forms are required to collect data from the user.

- Single-line text input controls − This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML <input> tag.
- Password input controls − This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTMl <input> tag.
- Multi-line text input controls − This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML <textarea> tag.

3) What is CSS? Types of CSS?

Answer:

A CSS(Cascading Style Sheets) comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in the document.

There are the three types of CSS:

- Internal or embedded
- External
- Inline

Internal CSS:

- This CSS style is an effective method of styling a single page. However, using this style for multiple pages is time-consuming as you need to put CSS rules on every page of your website.
- Add <style> tag in the <head> section of HTML document

External CSS:

- With external CSS, you'll link your web pages to an external .css file, which can be created by any text editor in your device.
- This CSS type is a more efficient method, especially for styling a large website. By editing one .css file, you can change your entire site at once.
- Link the HTML sheet to a separate .css file.

Inline CSS:

- Inline CSS is used to style a specific HTML element. For this CSS style, you'll only need to add the style attribute to each HTML tag, without using selectors.

- This CSS type is not really recommended, as each HTML tag needs to be styled individually. Managing your website may become too hard if you only use inline CSS.
- Apply CSS rules for specific elements.

4) Programs for List, Tables, images, forms, CSS?

Answer:

Example program on-

Lists:

```
<!DOCTYPE html>
<html>
<head>
   <style>
      ul {
         list-style-type: disc;
      }
   </style>
</head>
<body>
   <h1>Sample List in HTML</h1>
   <ul>
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
      <li>Item 4</li>
   </ul>
</body>
</html>
```

Tables:

```
<!DOCTYPE html>
<html>
<head>
   <style>
      table {
         width: 50%;
         border-collapse: collapse;
         margin: 20px;
      }

      th, td {
         border: 1px solid black;
         padding: 8px;
```

```
            text-align: center;
        }

        th {
            background-color: #f2f2f2;
        }
    </style>
</head>
<body>
    <h1>Sample Table in HTML</h1>
    <table>
        <tr>
            <th>Header 1</th>
            <th>Header 2</th>
        </tr>
        <tr>
            <td>Data 1</td>
            <td>Data 2</td>
        </tr>
        <tr>
            <td>Data 3</td>
            <td>Data 4</td>
        </tr>
    </table>
</body>
</html>
```

Images:

```
<!DOCTYPE html>
<html>
<head>
    <title>Image Example</title>
</head>
<body>
    <h1>Displaying an Image in HTML</h1>
    <p>Below is an example of an image:</p>
    <img src="https://www.example.com/image.jpg" alt="An
example image">
</body>
</html>
```

Forms:

```
<!DOCTYPE html>
```

```
<html>
<head>
   <title>Simple HTML Form</title>
</head>
<body>
   <h1>Sample Form</h1>
   <form action="submit.php" method="post">
      <label for="name">Name:</label>
      <input type="text" id="name" name="name"
required><br><br>

      <label for="email">Email:</label>
      <input type="email" id="email" name="email"
required><br><br>

      <label for="message">Message:</label><br>
      <textarea id="message" name="message" rows="4" cols="50"
required></textarea><br><br>

      <input type="submit" value="Submit">
   </form>
</body>
</html>
```

CSS:

```
<!DOCTYPE html>
<html>
<head>
   <style>
      body {
         font-family: Arial, sans-serif;
         background-color: #f2f2f2;
      }

      h1 {
         color: #333;
      }

      p {
         color: #666;
      }

      .highlight {
         background-color: yellow;
```

```
        }
    </style>
</head>
<body>
    <h1>Styling with CSS</h1>
    <p>This is a simple example of using CSS to style HTML
elements.</p>

    <p>Here is a <span class="highlight">highlighted</span>
word.</p>
</body>
</html>
```

5) Create a timetable using table tag in html.

Answer:

```
<!DOCTYPE html>
<html>
<head>
    <style>
        table {
            width: 80%;
            border-collapse: collapse;
            margin: 20px auto;
        }
        th, td {
            border: 1px solid #ccc;
            padding: 8px;
            text-align: center;
        }
        th {
            background-color: #f2f2f2;
        }
    </style>
</head>
<body>
    <table>
        <tr>
            <th>Day</th>
            <th>10:00-11:00</th>
            <th>11:00-12:00</th>
            <th>12:00-1:00</th>
            <th>1:40-2:40</th>
            <th>2:40-3:40</th>
        </tr>
        <tr>
            <th>Monday</th>
```

```
            <td>CN</td>
            <td>PE</td>
            <td>PE</td>
            <td>OE</td>
            <td>OE</td>
        </tr>
        <tr>
            <th>Tuesday</th>
            <td>CN/CD Lab</td>
            <td>CN/CD Lab</td>
            <td>CN/CD Lab</td>
            <td>PE</td>
            <td>CD</td>
        </tr>
        <tr>
            <th>Wednesday</th>
            <td>OS Lab</td>
            <td>OS Lab</td>
            <td>OS Lab</td>
            <td>OS</td>
            <td>DT</td>
        </tr>
        <tr>
            <th>Thursday</th>
            <td>Training</td>
            <td>Training</td>
            <td>Training</td>
            <td>Training</td>
            <td>Training</td>
        </tr>
        <tr>
            <th>Friday</th>
            <td>CD</td>
            <td>OS</td>
            <td>OS</td>
            <td>AECS Lab</td>
            <td>AECS Lab</td>
        </tr>
    </table>
</body>
</html>
```

UNIT 3:

Short Questions:

1) **What is JavaScript?**

Answer:

- JavaScript is a high-level, versatile, and widely-used programming language primarily used for adding interactivity and behaviour to websites.
- It's supported by web browsers and allows developers to create dynamic web content, handle user interactions, and perform client-side scripting.
- JavaScript is a core technology for web development, enabling the creation of responsive and engaging web applications.

2) **What is Document Object Model?**

Answer:

- JavaScript can access all the elements in a webpage making use of Document Object Model (DOM).
- In fact, the web browser creates a DOM of the webpage when the page is loaded.

3) **What is Data Base?**

Answer:

A database is a structured and organized collection of data, typically stored electronically. It allows for efficient storage, retrieval, and management of information. Databases can be as simple as a single file or as complex as large, distributed systems. They serve as a crucial tool for storing, managing, and retrieving data in various applications, from business operations to web applications, enabling data organization and analysis.

Long Questions:

1) **Functions in JavaScript?**

Answer:

- JavaScript functions are used to perform operations. We can call JavaScript function many times to reuse the code.
- Advantages of JavaScript function. There are mainly two advantages of JavaScript functions:
    - Code reusability: We can call a function several times so it saves coding.

- Less coding: It makes our program compact. We don't need to write many lines of code each time to perform a common task.

- The syntax of declaring function is given below.

```
function functionName([arg1, arg2, ...argN])
{
   //code to be executed
}
```

- JavaScript Functions can have 0 or more arguments.
- Let's see the simple example of function in JavaScript that does not has arguments.

```
<script>
function msg(){
alert("hello! this is message");
}
</script>
<input type="button" onclick="msg()" value="call function"/>
```

- We can call function by passing arguments. Let's see the example of function that has one argument.

```
<script>
function getcube(number){
alert(number*number*number);
</script>
<form>
<input type="button" value="click" onclick="getcube(4)"/>
</form>
```

- We can call function that returns a value and use it in our program. Let's see the example of function that returns value.

```
<script>
function getlnfo({
return "hello javatpoint! How r u?";
</script>
<script>
document.write(getlnfo0));
</script>
```

- Let's see function methods with description:
  - apply0 : It is used to call a function contains this value and a single array of arguments.
  - bind0 : It is used to create a new function.

- call() : It is used to call a function contains this value and an argument list.
- toString0 : It returns the result in a form of a string.

2) JavaScript validation, Events, JavaScript event handling, JavaScript Strings, JavaScript Dates, Array in JavaScript?

Answer:

JavaScript Validation:

JavaScript can be used for form validation on web pages. It allows you to check user input and ensure it meets specific criteria before processing the form data.

Events in JavaScript:

JavaScript can respond to various events triggered by user interactions or other sources, such as clicks, mouse movements, keyboard input, and more. Event handling allows you to define how your code reacts to these events.

JavaScript Event Handling:

Event handling in JavaScript involves attaching event listeners to HTML elements. These listeners wait for specific events to occur and then execute predefined JavaScript functions in response.

JavaScript Strings:

In JavaScript, strings are sequences of characters, such as text or data. You can manipulate and work with strings using various built-in methods for tasks like concatenation, substring extraction, and searching.

JavaScript Dates:

JavaScript provides built-in date and time handling capabilities. You can create, manipulate, and format dates, making it useful for tasks involving timestamps, scheduling, and time-based calculations.

Arrays in JavaScript:

Arrays are ordered lists of values in JavaScript. They are versatile and can hold various data types. You can perform operations on arrays, such as adding, removing, and iterating over elements.

3) Programs Of Java Script?

Answer:

Examples of Programs on JavaScript:

1. Factorial of a number:

```
function factorial(n) {
  if (n === 0 || n === 1) return 1;
  return n * factorial(n - 1);
}
let number = parseInt(prompt("Enter a number:"));
console.log(`Factorial of ${number} is: ${factorial(number)}`);
```

2. Example Program for Functions:

```
function addNumbers(a, b) {
      return a + b;
}
const result = addNumbers(5, 7);
console.log(result);
```

3. Example Program for Arrays:

```
const fruits = ["apple", "banana", "orange"];
fruits.push("grape");
fruits[1] = "kiwi";
console.log(fruits);
```

4. Example Program for Validation:

```
function validateEmail(email) {
  const emailPattern = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-
]+\.[a-zA-Z]{2,4}$/;
  return emailPattern.test(email);
}
const email = "example@email.com";
if (validateEmail(email)) {
console.log("Valid email address");
} else {
  console.log("Invalid email address");
}
```

5. Example Program for Events and Event Handling:

```
<!DOCTYPE html>
<html>
```

```
<head>
  <script>
    // Event handling function
    function showMessage() {
    alert("Button Clicked!");
    }
  </script>
</head>
<body>
  <button onclick="showMessage()">Click Me</button>
</body>
</html>
```

6. Example Program for Strings:

```
const text = "Hello, World!";
const upperCaseText = text.toUpperCase();
const subString = text.substring(0, 5);
console.log(upperCaseText);
console.log(subString);
```

7. Example Program for Dates:

```
const currentDate = new Date();
const year = currentDate.getFullYear();
const month = currentDate.getMonth();
const day = currentDate.getDate();
console.log(`Today is ${day}/${month + 1}/${year}`);
```

4) Explain Briefly System Defined Function.

Answer:

In JavaScript, system-defined functions, also known as built-in functions or standard functions, are functions that are pre-defined in the JavaScript language and are readily available for use without the need for explicit declaration or definition. These functions are part of the JavaScript core and provide a wide range of functionality to perform common tasks and operations. They are crucial for building web applications and working with data.

Here are some examples of system-defined functions in JavaScript:

1. Math Functions: JavaScript provides a Math object with built-in functions for mathematical operations, such as `Math.Random()`, `Math.floor()`, and `Math.max()`.

2. String Functions: String functions like `toUpperCase()`, `toLowerCase()`, `charAt()`, and `substring()` are used to manipulate and process text data.

3. Array Functions: Arrays have several built-in methods like `push()`, `pop()`, `map()`, and `filter()` that simplify array manipulation.

4. Date Functions: JavaScript has a Date object with functions to work with dates and times, including `getHours()`, `getMinutes()`, and `toLocaleDateString()`.

5. DOM Manipulation Functions: Functions like `getElementById()`, `addEventListener()`, and `querySelector()` are used to interact with the Document Object Model (DOM) and handle web page elements.

6. Type Conversion Functions: Functions like `parseInt()`, `parseFloat()`, and `String()` are used for converting data types.

7. Regular Expression Functions: JavaScript supports regular expressions with functions like `match ()`, `replace()`, and `test()` for pattern matching and manipulation of strings.

8. JSON Functions: Functions like `JSON.parse()` and `JSON.stringify()` are used for working with JSON data.

5) Explain Briefly Database Languages.

Answer:

Database languages are specialized languages used to interact with and manage databases. There are several types of database languages, each with a specific purpose. Here's a brief explanation of some common database languages:

1. DDL (Data Definition Language):
   DDL is used to define and manage the structure of a database. It includes commands like `CREATE`, `ALTER`, and `DROP` for creating and modifying database objects such as tables, indexes, and constraints. DDL statements are used to design and configure the database schema.

2. DML (Data Manipulation Language):

DML is used to manipulate and query the data stored in the database. Common DML statements include `SELECT` (for querying data), `INSERT` (for adding new data), `UPDATE` (for modifying existing data), and `DELETE` (for removing data). DML is focused on data retrieval and modification.

3. DCL (Data Control Language):
   DCL is concerned with access control and permissions. It includes commands like `GRANT` (to grant privileges or permissions to users) and `REVOKE` (to remove previously granted permissions). DCL statements are used to manage the security and authorization of database objects.

4. TCL (Transaction Control Language):
   TCL is used to manage transactions within a database. Transactions are sequences of one or more SQL statements that are executed as a single unit of work. TCL commands include `COMMIT` (to save changes made during a transaction), `ROLLBACK` (to undo changes and revert to a previous state), and `SAVEPOINT` (to set points within a transaction for future rollback).

5. PL/SQL (Procedural Language/SQL):
   PL/SQL is a procedural extension to SQL, primarily used in Oracle databases. It allows you to create stored procedures, functions, and triggers within the database. PL/SQL enables the development of complex, programmatic logic for data manipulation and business logic directly within the database.

These database languages provide the necessary tools to design, manage, and interact with databases effectively.

6) Explain Integrity Constraints.

Answer:

Integrity constraints serve as a critical mechanism for maintaining data integrity within a database. Common types of constraints include:

1. UNIQUE constraints: These ensure that a particular column's values are unique across the table, preventing duplicate entries.

2. NOT NULL constraints: They mandate that a column must contain valid (non-null) values, prohibiting empty or undefined data.

3. FOREIGN KEY constraints: These enforce relationships between tables by ensuring that a foreign key references a valid primary key in another table.

In a data warehouse, constraints are valuable for two primary purposes:

1. Data Cleanliness: Constraints validate data within the data warehouse, promoting consistency and correctness. They help prevent the introduction of erroneous or inconsistent data, maintaining data quality.

2. Query Optimization: Database management systems, like Oracle, use constraints to optimize SQL queries. Constraints are especially crucial for query optimization in scenarios such as query rewriting for materialized views, improving query performance by utilizing predefined rules.

Overall, integrity constraints are a fundamental component of database management, supporting data quality and query performance in various database environments.

7) Explain Aggregate Function.

Answer:

In database management an aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning.

Various Aggregate Functions:

1) Count()
2) Sum()
3) Avg()
4) Min()
5) Max()

SUM():

- The SUM() function calculates the sum of all values in a specified column. It is often used for adding numeric values.
- Example: SELECT SUM(salary) FROM employees; returns the total sum of salaries in the "employees" table.

AVG():

- The AVG() function calculates the average (mean) of all values in a specified column. It is useful for finding the average value of a set of numbers.
- Example: SELECT AVG(age) FROM customers; returns the average age of customers in the "customers" table.

COUNT():

- The COUNT() function counts the number of rows or non-null values in a specified column. It is often used to determine the size of a result set.
- Example: SELECT COUNT(*) FROM orders; returns the total number of rows in the "orders" table.

MAX():

- The MAX() function retrieves the maximum (largest) value from a specified column. It is useful for finding the highest value within a dataset.
- Example: SELECT MAX(price) FROM products; returns the highest price of a product in the "products" table.

MIN():

- The MIN() function retrieves the minimum (smallest) value from a specified column. It is used to find the lowest value in a dataset.
- Example: SELECT MIN(score) FROM exam_results; returns the lowest exam score from the "exam_results" table.