Web Performance Optimization Interview Questions & Answers

1. What is web performance optimization?

Answer:
Web performance optimization (WPO) refers to **techniques and best practices** used to improve website speed, reduce load time, enhance user experience, and improve SEO rankings.

---

2. Why is website speed important?

Answer:

- Improves **user experience**
- Reduces **bounce rate**
- Enhances **SEO rankings** (Google prefers fast-loading pages)
- Increases **conversion rates**

---

3. What are the core web vitals in performance optimization?

Answer:
Google's **Core Web Vitals** focus on user experience metrics:

- **LCP (Largest Contentful Paint)** → Measures **loading performance**
- **FID (First Input Delay)** → Measures **interactivity**
- **CLS (Cumulative Layout Shift)** → Measures **visual stability**

---

4. How can you measure website performance?

Answer:
Using tools like:

- **Google PageSpeed Insights**
- **Lighthouse**
- **WebPageTest**
- **GTmetrix**
- **Chrome DevTools (Performance Tab)**

---

5. What are the key factors that affect website performance?

Answer:

- Server response time

- Large images
- Unoptimized CSS & JavaScript
- Excessive HTTP requests
- Poor caching strategies

---

6. What is Time to First Byte (TTFB)?

Answer:
TTFB is the time taken from a user's request to the first byte received from the server. Lower TTFB means **faster server response**.

---

7. How do you reduce server response time?

Answer:

- Use **fast hosting** and **CDN**
- Enable **caching**
- Optimize **database queries**
- Use **Gzip or Brotli compression**

---

8. What is a Content Delivery Network (CDN) and how does it improve performance?

Answer:
A CDN stores website content on **multiple servers worldwide**, reducing **latency** and improving **load times** for users based on location.

---

9. What is lazy loading and how does it work?

Answer:
Lazy loading **delays** the loading of images/videos until they are needed.

```
<img src="placeholder.jpg" data-src="real-image.jpg" loading="lazy">
```

---

10. What are the benefits of caching?

Answer:

- **Reduces load time**
- **Decreases server load**
- **Improves user experience**

---

11. What is browser caching?

Answer:
Browser caching **stores static assets (CSS, JS, images)** in the user's browser, reducing the need for repeated downloads.

```
# Enable caching for images

ExpiresByType image/jpeg "access plus 1 year"
```

---

12. How does Gzip compression improve performance?

Answer:
Gzip **compresses files** before sending them, reducing size and improving speed.

```
# Enable Gzip compression

AddOutputFilterByType DEFLATE text/html text/css application/javascript
```

---

13. What is Brotli compression?

Answer:
A more **efficient** compression algorithm than Gzip, supported by modern browsers.

---

14. What is minification?

Answer:
Minification removes **unnecessary characters** (spaces, comments) from CSS, JS, and HTML to reduce file size.

---

15. How do you reduce render-blocking resources?

Answer:

- **Defer JavaScript** (defer or async attributes)
- Use **critical CSS**
- **Load CSS asynchronously**

```
<script src="script.js" defer></script>
```

---

16. What is asynchronous JavaScript loading?

Answer:
It allows JavaScript to load **without blocking** page rendering.

---

17. What is critical CSS?

Answer:
Extracts **above-the-fold** styles and loads them **inline**, improving **render speed**.

---

18. How do you optimize images for web performance?

Answer:

- Use **next-gen formats** (WebP, AVIF)
- Compress images (TinyPNG, ImageOptim)
- Implement **lazy loading**

---

19. What is responsive image loading?

Answer:
Loads different **image sizes** based on screen size.

```
<img src="small.jpg" srcset="large.jpg 1024w, medium.jpg 768w" alt="Example">
```

---

20. What is the impact of excessive HTTP requests?

Answer:
More HTTP requests = **slower load times**. Reduce them by:

- Using **CSS sprites**
- Combining **CSS & JS files**
- Enabling **HTTP/2**

---

21. What is HTTP/2 and why is it better than HTTP/1.1?

Answer:
HTTP/2 supports **multiplexing**, reducing **multiple requests overhead**.

---

22. What is lazy script loading?

Answer:
Loads scripts **only when needed**, improving performance.

---

23. How does WebP improve web performance?

Answer:
WebP provides **better compression** than PNG/JPEG, reducing image size.

```
<picture>
  <source srcset="image.webp" type="image/webp">
  <img src="image.jpg" alt="Image">
</picture>
```

---

24. What is First Contentful Paint (FCP)?

Answer:
Measures how **quickly content appears** after loading.

---

25. What is First Input Delay (FID)?

Answer:
Measures how **responsive a website** is to user input.

---

26. What is Cumulative Layout Shift (CLS)?

Answer:
Measures **unexpected layout shifts**, affecting user experience.

---

27. What is async vs defer in JavaScript?

Answer:

Attribute	Loads	Executes
async	Parallel	Immediately
defer	Parallel	After parsing

---

28. What is preloading in web performance?

Answer:
Preloads important assets **before they are needed**.

```
<link rel="preload" href="style.css" as="style">
```

---

29. What is DNS prefetching?

Answer:
Resolves domain names **before user clicks a link**, speeding up navigation.

```
<link rel="dns-prefetch" href="//example.com">
```

---

30. What is a service worker?

Answer:
Runs in the background and enables **offline caching**.

31. What is Page Speed Index?

Answer:
Page Speed Index (PSI) measures **how quickly the visible parts of a webpage load**. A lower score means a faster website.

---

32. What is Lazy Evaluation in JavaScript?

Answer:
Lazy evaluation means **delaying computations** until the result is actually needed.

```
const lazyValue = () => {

  console.log("Evaluated only when called");

  return 42;

};

console.log(lazyValue()); // Output: 42
```

---

33. What is a fast hosting provider for performance?

Answer:
Some of the best hosting providers for **fast performance** include:

- **Cloudflare Pages**
- **Vercel**
- **Netlify**
- **AWS (Amazon Web Services)**
- **Google Cloud**
- **DigitalOcean**

---

34. How do you reduce DOM size?

Answer:

- **Remove unnecessary elements**
- **Use efficient selectors**

- **Minimize nesting of elements**
- **Use Virtual DOM (React, Vue.js)**
- **Lazy load components**

---

35. How do web fonts affect performance?

Answer:

- **Too many font files** slow down rendering
- **Font-display: swap** ensures **faster text rendering**

```
@font-face {
  font-family: "MyFont";
  src: url("myfont.woff2") format("woff2");
  font-display: swap;
}
```

---

36. What is the IntersectionObserver API?

Answer:
The **IntersectionObserver API** detects when elements enter the viewport, improving lazy loading.

```
const observer = new IntersectionObserver(entries => {
  entries.forEach(entry => {
    if (entry.isIntersecting) {
      console.log("Element is visible");
    }
  });
});
observer.observe(document.querySelector("#targetElement"));
```

---

37. How does AMP (Accelerated Mobile Pages) improve performance?

Answer:
AMP **removes heavy JavaScript**, optimizes images, and **caches pages on Google's CDN**, making pages load **faster**.

---

38. What is a Progressive Web App (PWA)?

Answer:
A **PWA** is a web app that works **offline**, loads quickly, and provides an **app-like experience**.

---

39. What is a Skeleton Screen in performance optimization?

Answer:
A **skeleton screen** is a placeholder UI that appears before content loads, improving perceived performance.

```
<div class="skeleton-loader"></div>
```

---

40. How do you optimize CSS animations for performance?

Answer:

- Use **GPU-accelerated properties** (transform, opacity)
- Avoid **layout-triggering properties** (width, height, top, left)

```
/* Good */
.element {
  transform: translateX(100px);
}


/* Bad */
.element {
  left: 100px;
}
```

---

41. What is HTTP Keep-Alive?

Answer:
Keep-Alive keeps connections open between the browser and server, reducing the overhead of re-establishing new connections.

```
KeepAlive On
```

---

42. What is Connection Pooling?

Answer:
Connection pooling **reuses open connections** instead of creating new ones for each request, improving speed.

---

43. What is Render Throttling?

Answer:
Browsers **limit rendering frequency** to save resources. Developers can optimize by **reducing JavaScript execution** and **avoiding unnecessary reflows**.

---

44. How do third-party scripts affect performance?

Answer:

- **Increase load time**
- **Block rendering**
- **Introduce security risks**

Solution: Load **third-party scripts asynchronously**.

```
<script async src="analytics.js"></script>
```

---

45. How does Server Push work in HTTP/2?

Answer:
Server Push **preloads resources** before the client requests them, **reducing latency**.

```
http2_push /styles.css;
```

---

46. How does Memory Leak affect performance?

Answer:
A **memory leak** occurs when JavaScript objects remain in memory **after they are no longer needed**, causing performance issues.

```
let data = [];

setInterval(() => {

  data.push(new Array(1000000)); // Memory leak

}, 1000);
```

Solution:

- **Use garbage collection-friendly patterns**
- **Remove event listeners**

---

47. What is GPU Acceleration in Web Performance?

Answer:
Certain CSS properties (transform, opacity) **offload rendering** to the GPU, making animations smoother.

```
.element {
  will-change: transform;
}
```

---

48. What is a Web Worker?

Answer:
Web Workers allow running **JavaScript in the background**, preventing UI freezing.

```
let worker = new Worker("worker.js");
worker.postMessage("Hello");
```

---

49. How do you reduce JavaScript execution time?

Answer:

- **Minify JavaScript**
- **Use tree shaking**
- **Load scripts asynchronously**
- **Optimize loops and event listeners**

```
// Instead of:
for (let i = 0; i < arr.length; i++) { ... }


// Use:
for (const item of arr) { ... }
```

---

50. What is an Image CDN?

Answer:
An **Image CDN** optimizes and serves images **based on user location and device**, reducing load time.

Examples:

- Cloudinary
- Imgix
- Cloudflare Images

50 very important PHP questions with answers

1. What is PHP?

Answer:
PHP (Hypertext Preprocessor) is a **server-side scripting language** used for web development. It is embedded in HTML and executes on the server.

---

2. What are the key features of PHP?

Answer:

- Open-source
- Cross-platform compatibility
- Supports databases like MySQL, PostgreSQL
- Embeds directly into HTML
- Supports object-oriented programming
- Secure and scalable

---

3. What is the difference between echo and print in PHP?

Answer:

- echo can output **multiple values** and is **faster**.
- print returns **1** and can be used in expressions.

```
echo "Hello ", "World"; // Works

print "Hello World"; // Works but cannot take multiple arguments
```

---

4. How do you declare a variable in PHP?

Answer:
PHP variables start with $ and do not require type declarations.

```
$name = "John";

$age = 25;
```

---

5. What are the different types of PHP variables?

Answer:

- **String** ($name = "John";)

- **Integer** ($age = 25;)
- **Float** ($price = 12.99;)
- **Boolean** ($isTrue = true;)
- **Array** ($colors = ["red", "blue"];)
- **Object** ($car = new Car(););

---

6. What is the difference between == and === in PHP?

Answer:

- == checks only **value equality**.
- === checks **both value and data type**.

```
var_dump(5 == "5");  // true
var_dump(5 === "5"); // false
```

---

7. What are superglobals in PHP?

Answer:
Superglobals are **built-in global arrays** used to access data:

- $_GET → Data from URL parameters
- $_POST → Data from forms
- $_REQUEST → Combines $_GET and $_POST
- $_SERVER → Server info
- $_SESSION → User session data
- $_COOKIE → Cookies

---

8. What is the difference between GET and POST?

Answer:

GET	POST
Data sent in URL	Data sent in request body
Less secure	More secure
Limited data (max 2048 chars)	Large data allowed
Used for fetching data	Used for submitting data

```
$_GET["username"];
```

```
$_POST["password"];
```

---

9. How to connect PHP with MySQL?

Answer:

```
$conn = new mysqli("localhost", "root", "", "database_name");


if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
```

---

10. What is the use of isset() in PHP?

Answer:
Checks if a variable is **set and not null**.

```
$name = "John";

echo isset($name); // true
```

---

11. What is the difference between isset() and empty()?

Answer:

- isset($var) → **Returns true** if variable is **defined** and **not null**.
- empty($var) → **Returns true** if variable is **undefined, null, or false**.

---

12. What is the difference between include and require?

Answer:

- include → **Shows a warning but continues execution** if the file is missing.
- require → **Throws a fatal error** and stops execution if the file is missing.

```
include "file.php";

require "file.php";
```

---

13. What is a session in PHP?

Answer:
A **session** stores **user data across multiple pages**.

```
session_start();
```

```
$_SESSION["username"] = "John";
```

---

14. What is a cookie in PHP?

Answer:
A **cookie** stores **small amounts of user data on the client's browser**.

```
setcookie("username", "John", time() + 3600);
```

---

15. What is the difference between session and cookie?

Answer:

Session	**Cookie**
Stored on server	Stored on user's browser
More secure	Less secure
Data is lost when browser is closed	Data persists until expiration

---

16. How do you prevent SQL injection in PHP?

Answer:
Use **prepared statements**:

```
$stmt = $conn->prepare("SELECT * FROM users WHERE username = ?");

$stmt->bind_param("s", $username);

$stmt->execute();
```

---

17. What is PDO in PHP?

Answer:
PHP Data Objects (PDO) is a database abstraction layer for secure and flexible database access.

---

18. How to upload a file in PHP?

Answer:

```
move_uploaded_file($_FILES["file"]["tmp_name"], "uploads/" . $_FILES["file"]["name"]);
```

---

19. How to send an email in PHP?

Answer:

```
mail("test@example.com", "Subject", "Message", "From: sender@example.com");
```

---

20. What is Composer in PHP?

Answer:
Composer is a **dependency manager** for PHP.

---

21. How to debug PHP code?

Answer:
Use error_reporting(E_ALL); and var_dump().

---

22. What is an associative array in PHP?

Answer:
An array with **key-value pairs**:

```
$person = ["name" => "John", "age" => 25];
```

---

23. What is a lambda function in PHP?

Answer:
An **anonymous function**:

```
$greet = function($name) {
   return "Hello, $name!";
};
echo $greet("John");
```

---

24. What is JSON in PHP?

Answer:

- **Encode:** json_encode($array);
- **Decode:** json_decode($json);

---

25. What are explode() and implode() functions?

Answer:

- explode() → Splits a string into an array.
- implode() → Joins an array into a string.

```
$arr = explode(",", "apple,banana,mango");

$str = implode("-", ["apple", "banana", "mango"]);
```

26. How do you use Object-Oriented Programming (OOP) in PHP?

Answer:
PHP supports OOP concepts like **classes, objects, inheritance, polymorphism, and encapsulation**.

```
class Car {
    public $brand;


    public function setBrand($brand) {
        $this->brand = $brand;
    }


    public function getBrand() {
        return $this->brand;
    }
}


$myCar = new Car();
$myCar->setBrand("Toyota");
echo $myCar->getBrand(); // Output: Toyota
```

---

27. What is a constructor in PHP?

Answer:
A **constructor** is a special function that is automatically called when an object is created.

```
class Car {
    public $brand;


    public function __construct($brand) {
        $this->brand = $brand;
    }


    public function getBrand() {
        return $this->brand;
    }
}

$car = new Car("Honda");
echo $car->getBrand(); // Output: Honda
```

---

28. What is a destructor in PHP?

Answer:
A **destructor** is a special method (__destruct()) that is automatically called when an object is destroyed.

```
class Car {
    public function __destruct() {
        echo "Car object is being destroyed!";
    }
}

$car = new Car();
unset($car); // Output: Car object is being destroyed!
```

---

29. What are PHP magic methods?

Answer:
Magic methods **start with two underscores (__)** and perform special tasks.

Magic Method	Description
__construct()	Called when an object is created
__destruct()	Called when an object is destroyed
__get()	Handles inaccessible properties
__set()	Handles setting inaccessible properties
__call()	Handles calls to undefined methods
__toString()	Converts an object to a string

Example:

```
class Test {
    public function __toString() {
        return "This is a Test class object!";
    }
}

$test = new Test();
echo $test; // Output: This is a Test class object!
```

---

30. Explain inheritance in PHP.

Answer:
Inheritance allows a class to **reuse** properties and methods from another class.

```
class Animal {
    public function makeSound() {
        echo "Some sound";
    }
}

class Dog extends Animal {
    public function bark() {
        echo "Woof!";
    }
}

$dog = new Dog();
$dog->makeSound(); // Output: Some sound
$dog->bark(); // Output: Woof!
```

---

31. What is the final keyword in PHP?

Answer:
The final keyword **prevents a class from being extended** or a method from being overridden.

```
final class Animal {
    public function sound() {
        echo "Some sound";
    }
}

// This will cause an error
// class Dog extends Animal {}

class Dog {
    final public function bark() {
        echo "Woof!";
    }
}

// This will cause an error
// class Puppy extends Dog {
//     public function bark() {}
// }
```

---

32. What is the static keyword in PHP?

Answer:
The static keyword allows methods and properties to be accessed without creating an object.

```
class Math {
    public static function add($a, $b) {
        return $a + $b;
    }
}

echo Math::add(5, 10); // Output: 15
```

---

33. How do you create an interface in PHP?

Answer:
An interface defines methods that **must be implemented** in a class.

```
interface Animal {
  public function makeSound();
}

class Dog implements Animal {
  public function makeSound() {
    echo "Woof!";
  }
}

$dog = new Dog();
$dog->makeSound(); // Output: Woof!
```

---

34. What are traits in PHP?

Answer:
Traits allow **multiple inheritance** by including methods in multiple classes.

```
trait Logger {
  public function log($message) {
    echo "Log: $message";
  }
}

class User {
  use Logger;
}

$user = new User();
$user->log("User created!"); // Output: Log: User created!
```

---

35. How do you hash passwords in PHP?

Answer:
Use password_hash() for secure password storage.

```
$hashedPassword = password_hash("mypassword", PASSWORD_BCRYPT);
```

To verify:

```
if (password_verify("mypassword", $hashedPassword)) {
    echo "Password matches!";
}
```

---

36. What is unlink() in PHP?

Answer:
unlink() deletes a file from the server.

```
unlink("file.txt"); // Deletes file.txt
```

---

37. How to handle exceptions in PHP?

Answer:
Use try-catch blocks.

```
try {
    throw new Exception("Something went wrong!");
} catch (Exception $e) {
    echo $e->getMessage();
}
```

---

38. What is try-catch in PHP?

Answer:
It handles exceptions **gracefully**.

```
try {
    $num = 5 / 0; // Error
} catch (Exception $e) {
    echo "Error: " . $e->getMessage();
}
```

---

39. What is the header() function in PHP?

Answer:
Used to modify HTTP headers.

```
header("Location: home.php");

exit();
```

---

40. How do you use AJAX with PHP?

Answer:
AJAX **sends requests to PHP without reloading the page**.

```
fetch("server.php")

    .then(response => response.text())

    .then(data => console.log(data));
```

---

41. What is cURL in PHP?

Answer:
cURL handles **API requests**.

```
$ch = curl_init("https://api.example.com");

curl_exec($ch);

curl_close($ch);
```

---

42. What is XML parsing in PHP?

Answer:
PHP uses simplexml_load_string() for XML parsing.

---

43. What is file_get_contents()?

Answer:
Reads the contents of a file into a string.

```
$data = file_get_contents("file.txt");
```

---

44. What is fopen() in PHP?

Answer:
Opens a file.

```
$file = fopen("file.txt", "r");
```

---

45. What are PHP filters?

Answer:
Filters validate and sanitize data.

```
$input = filter_var("user@example.com", FILTER_VALIDATE_EMAIL);
```

---

46. How do you prevent XSS in PHP?

Answer:
Use htmlspecialchars().

```
echo htmlspecialchars("<script>alert('XSS')</script>");
```

---

47. What is ob_start() in PHP?

Answer:
Turns on output buffering.

---

48. How do you set a timezone in PHP?

Answer:

```
php
date_default_timezone_set("Asia/Kolkata");
```

---

49. How to generate a unique ID in PHP?

Answer:
Use uniqid().

```
echo uniqid();
```

---

50. What is .htaccess in PHP?

Answer:
A configuration file for **Apache** servers.

```
RewriteEngine On
RewriteRule ^home$ home.php
```

50 very important MySQL questions and answers

1. What is MySQL?

Answer:
MySQL is an **open-source relational database management system (RDBMS)** that stores and manages data using SQL (Structured Query Language). It is widely used in web applications.

---

2. What are the different types of MySQL databases?

Answer:
MySQL supports different storage engines:

- **InnoDB** (default, supports transactions, foreign keys)
- **MyISAM** (faster for reads, no transactions)
- **Memory** (stores data in RAM, fast but volatile)
- **CSV** (stores data in CSV format)
- **Archive** (optimized for log storage)

---

3. What is the difference between MySQL and SQL?

Answer:

- **SQL (Structured Query Language)** is a language used to manage databases.
- **MySQL** is an RDBMS that uses SQL to manage data.

---

4. What is a Primary Key?

Answer:
A **Primary Key** is a unique identifier for a record in a table. It must be unique and **cannot contain NULL values**.

```
CREATE TABLE students (
   id INT PRIMARY KEY,
   name VARCHAR(100)
);
```

---

5. What is a Foreign Key?

Answer:
A **Foreign Key** links two tables and enforces referential integrity.

```
CREATE TABLE orders (
   order_id INT PRIMARY KEY,
   user_id INT,
   FOREIGN KEY (user_id) REFERENCES users(id)
);
```

---

6. What are the different types of JOINs in MySQL?

Answer:

- **INNER JOIN** (matches records in both tables)
- **LEFT JOIN** (all records from left table + matching records from right)
- **RIGHT JOIN** (all records from right table + matching records from left)
- **FULL JOIN** (not supported in MySQL, but can be simulated)

Example of **INNER JOIN**:

```
SELECT users.name, orders.order_id
FROM users
INNER JOIN orders ON users.id = orders.user_id;
```

---

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

Answer:

- **DELETE**: Removes specific records (WHERE clause allowed).
- **TRUNCATE**: Deletes all records but keeps the table structure.
- **DROP**: Deletes the entire table (including structure).

```
DELETE FROM students WHERE id = 1;  -- Deletes a specific row
TRUNCATE TABLE students;  -- Deletes all rows
DROP TABLE students;  -- Deletes table completely
```

---

8. What is an Index in MySQL?

Answer:
An **index** speeds up queries by allowing fast lookups.

```
CREATE INDEX idx_name ON students(name);
```

---

9. What is the default port of MySQL?

Answer:
Port 3306 is the default for MySQL.

---

10. How to change a column name in MySQL?

Answer:
Use the ALTER TABLE command.

```
ALTER TABLE students CHANGE old_name new_name VARCHAR(100);
```

---

11. What is ACID in MySQL?

Answer:
ACID properties ensure database reliability:

- **Atomicity** (all or nothing)
- **Consistency** (valid state before and after transactions)
- **Isolation** (transactions don't interfere)
- **Durability** (data remains after a crash)

---

12. How to retrieve the current date and time in MySQL?

Answer:

```
SELECT NOW(); -- Returns current date and time

SELECT CURDATE(); -- Returns current date

SELECT CURTIME(); -- Returns current time
```

---

13. What is a View in MySQL?

Answer:
A **view** is a virtual table.

```
CREATE VIEW user_orders AS

SELECT users.name, orders.order_id FROM users

JOIN orders ON users.id = orders.user_id;
```

---

14. What is a Stored Procedure?

Answer:
A **stored procedure** is a set of SQL statements.

```
DELIMITER //
CREATE PROCEDURE GetUsers()
BEGIN
   SELECT * FROM users;
END //
DELIMITER ;
```

Call it using:

```
CALL GetUsers();
```

---

15. How to check MySQL version?

Answer:

```
SELECT VERSION();
```

---

16. How to get the total number of rows in a table?

Answer:

```
SELECT COUNT(*) FROM students;
```

---

17. What is a Trigger in MySQL?

Answer:
A **trigger** is an automatic action executed before/after INSERT, UPDATE, or DELETE.

```
CREATE TRIGGER before_insert_student
BEFORE INSERT ON students
FOR EACH ROW
SET NEW.name = UPPER(NEW.name);
```

---

18. How do you import and export databases in MySQL?

Answer:

- **Export:**

```
mysqldump -u root -p database_name > backup.sql
```

- **Import:**

```
mysql -u root -p database_name < backup.sql
```

---

19. How to find duplicate records?

Answer:

SELECT name, COUNT(*)

FROM students

GROUP BY name

HAVING COUNT(*) > 1;

---

20. How do you limit query results in MySQL?

Answer:

SELECT * FROM students LIMIT 5;

---

21. What is Normalization?

Answer:
Normalization organizes data to reduce redundancy.

- **1NF**: Atomic values
- **2NF**: No partial dependencies
- **3NF**: No transitive dependencies

---

22. What is the difference between CHAR and VARCHAR?

Answer:

- **CHAR** (fixed-length, faster for short data).
- **VARCHAR** (variable-length, saves space).

CREATE TABLE example (name CHAR(10), email VARCHAR(50));

---

23. What is MySQL Workbench?

Answer:
A GUI tool for managing MySQL databases.

---

24. What is the use of GROUP BY?

Answer:
Used to **group results based on a column**.

SELECT department, COUNT(*) FROM employees GROUP BY department;

---

25. How to update a record in MySQL?

Answer:

UPDATE students SET name = 'John' WHERE id = 1;

---

26. What is the difference between NOW() and CURRENT_TIMESTAMP()?

Answer:
Both return the current date and time, but **NOW()** allows mathematical operations.

---

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

Answer:

- **UNION** removes duplicates.
- **UNION ALL** keeps all rows.

---

28. How to find the highest salary?

Answer:

SELECT MAX(salary) FROM employees;

---

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

Answer:

- **INNER JOIN** returns matched rows.
- **CROSS JOIN** returns all combinations.

---

30. How do you delete duplicate rows?

Answer:

DELETE FROM students

WHERE id NOT IN (SELECT MIN(id) FROM students GROUP BY name);

31. What is the difference between WHERE and HAVING?

Answer:

- **WHERE** filters rows before aggregation.
- **HAVING** filters rows after aggregation.

```
SELECT department, COUNT(*)

FROM employees

GROUP BY department

HAVING COUNT(*) > 5;
```

---

32. What is a Subquery?

Answer:
A **subquery** is a query inside another query.

```
SELECT name FROM employees WHERE salary > (SELECT AVG(salary) FROM employees);
```

---

33. What is the difference between EXISTS and IN?

Answer:

- **IN** checks if a value exists in a list.
- **EXISTS** checks if a subquery returns rows.

```
SELECT * FROM students WHERE id IN (SELECT student_id FROM exams);
```

---

34. How do you find the second highest salary?

Answer:

```
SELECT DISTINCT salary FROM employees ORDER BY salary DESC LIMIT 1 OFFSET 1;
```

---

35. How do you get the last inserted ID?

Answer:

```
SELECT LAST_INSERT_ID();
```

---

36. What is the difference between DDL, DML, and DCL?

Answer:

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

---

37. How do you find null values in MySQL?

Answer:

```
SELECT * FROM students WHERE name IS NULL;
```

---

38. How to remove null values from a table?

Answer:

```
DELETE FROM students WHERE name IS NULL;
```

---

39. What is AUTO_INCREMENT in MySQL?

Answer:
It automatically increases the value of an integer column.

```
CREATE TABLE users (

   id INT AUTO_INCREMENT PRIMARY KEY,

   name VARCHAR(50)

);
```

---

40. What is the use of COALESCE() function?

Answer:
Returns the first non-null value.

```
SELECT COALESCE(NULL, 'Default', 'Another') AS result;
```

---

41. What is the difference between CHAR_LENGTH() and LENGTH()?

Answer:

- **CHAR_LENGTH()** counts characters.
- **LENGTH()** counts bytes.

```
SELECT CHAR_LENGTH('Hello'), LENGTH('Hello');
```

---

42. How do you fetch even and odd rows from a table?

Answer:
Even rows:

```
SELECT * FROM students WHERE id % 2 = 0;
```

Odd rows:

```
SELECT * FROM students WHERE id % 2 <> 0;
```

---

43. What is the IFNULL() function in MySQL?

Answer:
Returns a default value if the column is NULL.

```
SELECT IFNULL(NULL, 'Default');
```

---

44. How do you update multiple rows in MySQL?

Answer:

```
UPDATE employees

SET salary = CASE

   WHEN department = 'IT' THEN salary + 5000

   WHEN department = 'HR' THEN salary + 3000

END;
```

---

45. What is the REPLACE() function in MySQL?

Answer:
It replaces substrings in a column.

```
SELECT REPLACE('Hello World', 'World', 'MySQL');
```

---

46. What is the difference between NOW(), SYSDATE(), and CURDATE()?

Answer:

- **NOW()** → Returns current date & time.
- **SYSDATE()** → Returns system time at execution.
- **CURDATE()** → Returns only the date.

```
SELECT NOW(), SYSDATE(), CURDATE();
```

---

47. How do you concatenate strings in MySQL?

Answer:

```
SELECT CONCAT('Hello', ' ', 'World');
```

---

48. How to add a new column to an existing table?

Answer:

```
ALTER TABLE students ADD email VARCHAR(100);
```

---

49. What is the difference between a Unique Key and a Primary Key?

Answer:

- **Primary Key**: Unique + NOT NULL
- **Unique Key**: Only unique, allows NULL

```
CREATE TABLE users (
   id INT PRIMARY KEY,
   email VARCHAR(100) UNIQUE
);
```

---

50. How do you rename a table in MySQL?

Answer:

```
RENAME TABLE old_table TO new_table;
```