

<b>Arrays</b>	REGULAR EXPRESSION SYNTAX					
array_intersect	^ Start of string					
array_merge	\$ End of string					
array_pop	. Any single character					
array_push	(a b) a or b					
array_reverse	(...) Group section					
array_walk	[abc] Item in range (a or b or c)					
count	[^abc] Not in range (not a or b or c)					
in_array	\s White space					
<b>Date and Time</b>	a? Zero or one of a					
date	a* Zero or more of a					
mktime	a+ One or more of a					
strtotime	a{3} Exactly 3 of a					
time	a{3,} 3 or more of a					
<b>Filesystem</b>	a{3,6} Between 3 and 6 of a					
clearstatcache	\ Escape character					
copy	[:punct:] Any punctuation symbol					
fclose	[:space:] Any space character					
fgets	[:blank:] Space or tab					
file						
filemtime						
filesize						
file_exists						
fopen						
fread						
fwrite						
is_dir						
is_file						
readfile						
<b>Headers</b>						
header						
headers_sent						
setcookie						
<b>Mail</b>						
mail						
<b>Numbers</b>						
ceil						
floor						
number_format						
round						
<b>Miscellaneous</b>						
define						
eval						
exit						
<b>Network</b>						
fsockopen						
fsckopen						
<b>PHP Options</b>						
ini_get						
ini_restore						
ini_set						
phpinfo						
<b>Execution</b>						
exec						
REGULAR EXPRESSION SYNTAX						
DATE FORMATTING						
REGEX						
ereg						
eregi						
ereg_replace						
eregi_replace						
split						
preg_match						
preg_match_all						
preg_replace						
preg_split						
Sessions						
session_destroy						
session_id						
session_start						
Strings						
addslashes						
crypt						
echo						
explode						
implode						
md5						
nl2br						
sprintf						
strip_tags						
stripslashes						
stristr						
strpos						
strrev						
strrchr						
strtolower						
strtoupper						
str_replace						
substr						
trim						
URLs						
rawurldecode						
rawurlencode						
urld decode						
urlencode						
Variables						
empty						
is_array						
is_int						
is_null						
is_numeric						
is_object						
isset						
serialize						
unserialize						
SuperGlobals						
\$_ENV						
\$_GET						
\$_POST						
\$_COOKIE						
\$_SESSION						
\$_SERVER						
\$_FILES						
\$_REQUEST						
\$_GLOBALS						
FUNCTION ARGUMENT ORDERS						
array_push ( array, element )						
in_array ( needle, haystack )						
explode ( separator, string )						
split ( pattern, string )						
preg_split ( pattern, string )						
ereg ( pattern, string )						
strpos ( haystack, needle )						
str_replace ( search, replace, string )						
fopen() MODES						
r Read						
r+ Read and write, prepend						
w Write, truncate						
w+ Read and write, truncate						
a Write, append						
a+ Read and write, append						
1. date("w"): 0 is Sunday, 6 is Saturday.						
2. Week that overlaps two years belongs to year that contains most days of that week. Hence week number for 1st January of a given year can return 53rd week if week belongs to previous year.						
date("W", mktime(0, 0, 0, 12, 28, \$year)) always gives correct number of weeks in \$year.						
3. The Epoch was the 1st January 1970.						

# IQ Question

## **include() and require()**

They both include a specific file but on require the process exits with a fatal error if the file can't be included, while include statement may still pass and jump to the next step in the execution.

## **How can we get the IP address of the client?**

```
$_SERVER["REMOTE_ADDR"];
```

## **What are the main error types in PHP and how do they differ?**

- **Notices** – Simple, non-critical errors that are occurred during the script execution. An example of a Notice would be accessing an undefined variable.
- **Warnings** – more important errors than Notices, however the scripts continue the execution. An example would be `include()` a file that does not exist.
- **Fatal** – this type of error causes a termination of the script execution when it occurs. An example of a Fatal error would be accessing a property of a non-existent object or `require()` a non-existent file.

## **How can you enable error reporting in PHP?**

Check if “`display_errors`” is equal “on” in the **php.ini** or declare “`ini_set('display_errors', 1)`” in your script.

Then, include “`error_reporting(E_ALL)`” in your code to display all types of error messages during the script execution.

## **Can you extend a Final defined class?**

No, you cannot extend a **Final** defined class. A **Final** class or method declaration prevents child class or method overriding.

### **What are the `__construct()` and `__destruct()` methods in a PHP class?**

All objects in PHP have Constructor and Destructor methods built-in. The Constructor method is called immediately after a new instance of the class is being created, and it's used to initialize class properties. The Destructor method takes no parameters.

### **How would you declare a function that receives one parameter name `hello`?**

If `hello` is `true`, then the function must print `hello`, but if the function doesn't receive `hello` or `hello` is `false` the function must print `bye`.

```
<?php
function showMessage($hello=false){
    echo ($hello)?'hello':'bye';
}
?>
```

### **The value of the variable `input` is a string `1,2,3,4,5,6,7`. How would you get the sum of the integers contained inside `input`?**

```
<?php
echo array_sum(explode(',',$input));
?>
```

### **Suppose you receive a form submitted by a post to subscribe to a newsletter. This form has only one field, an input text field named `email`. How would you validate whether the field is empty?**

```
<?php
if(empty($_POST['email'])){
    echo "The email cannot be empty";
}
?>
```

### **How would you implement a class in the following scenario?**

```
<?php
class dragonBall{
    private $ballCount;

    public function __construct(){
        $this->ballCount=0;
    }
}
```

```
public function iFoundABall(){
    $this->ballCount++;
    if($this->ballCount==7){
        echo "You can ask for your wish.";
        $this->ballCount=0;
    }
}
?>
```

## What are the 3 scope levels available in PHP and how would you define them?

**Private** – Visible only in its own class

**Public** – Visible to any other code accessing the class

**Protected** – Visible only to classes parent(s) and classes that extend the current class

## What are getters and setters and why are they important?

Getters and setters are methods used to declare or obtain the values of variables, usually private ones. They are important because it allows for a central location that is able to handle data prior to declaring it or returning it to the developer.

Within a getter or setter one is able to consistently handle data that will eventually be passed into a variable or additional functions. An example of this would be a user's name. If a setter is not being used and the developer is just declaring the `$userName` variable by hand, you could end up with results as such: "kevin", "KEVIN", "KeViN", "", etc.

With a setter, the developer can not only adjust the value, for example, `ucfirst($userName)`, but can also handle situations where the data is not valid such as the example where "" is passed. The same applies to a getter – when the data is being returned, it can be modified the results to include `strtoupper($userName)` for proper formatting further up the chain.

## What are SQL Injections, how do you prevent them and what are the best practices?

SQL injections are a method to alter a query in a SQL statement send to the database server. That modified query then might leak information like username/password combinations and can help the intruder to further compromise the server.

To prevent SQL injections, one should always check & escape all user input. In PHP, this is easily forgotten due to the easy access to `$_GET` & `$_POST`, and is often forgotten by inexperienced developers. But there are also many other ways that users can manipulate variables used in a SQL query through cookies or even uploaded files (filenames). The only real protection is to use prepared statements everywhere consistently.

Do not use any of the `mysql_*` functions which have been deprecated since PHP 5.5 ,but rather use PDO, as it allows you to use other servers than MySQL out of the box. `mysqli_*` are still an option, but there is no real reason nowadays not to use PDO, ODBC or DBA to get real abstraction. Ideally you want to use Doctrine or Propel to get rid of writing SQL queries all together and use object-relational mapping which binds rows from the database to objects in the application.

### **What does the following code output?**

```
$i = 016;  
echo $i / 2;
```

The Output should be 7. The leading zero indicates an octal number in PHP, so the number evaluates to the decimal number 14 instead to decimal 16.

### **Why would you use === instead of ==?**

If you would want to check for a certain type, like an integer or boolean, the `==` will do that exactly like one would expect from a strongly typed language, while `==` would convert the data temporarily and try to match both operand's types.

The identity operator (`==`) also performs better as a result of not having to deal with type conversion. Especially when checking variables for true/false, one should avoid using `==` as this would also take into account 0/1 or other similar representation.

### **What are PSRs? Choose 1 and briefly describe it.**

PSRs are PHP Standards Recommendations that aim at standardizing common aspects of PHP Development.

### **What PSR Standards do you follow? Why would you follow a PSR standard?**

Developers should follow a PSR because coding standards often vary between developers and companies. This can cause issues when reviewing or fixing another developer's code and finding a code structure that is different from yours. A PSR standard can help streamline the expectations of how the code should look, thus cutting down confusion and in some cases, syntax errors.

## **Do you use Composer? If yes, what benefits have you found in it?**

Using Composer is a tool for dependency management. The candidate can declare the libraries your product relies on and Composer will manage the installation and updating of the libraries. The benefit is a consistent way of managing the libraries depended on so less time is spent managing the libraries.

## **What is PHP most used for?**

**PHP** has a plethora of uses for developers and the ones mentioned below are some of the most widely used concepts that PHP offers:

- With PHP, it becomes very easy to provide restricted access to the required content of the website.
- It allows users to access individual cookies and set them as per requirement.
- Database manipulation operations, such as addition, deletion, and modification, can be done easily.
- Form handling, alongside features that involve file handling concepts and email integration, is used widely.
- The system module allows users to perform a variety of system functions such as open, read, write, etc.

## **What is the meaning of PEAR in PHP?**

PEAR stands for PHP Extension and Application Repository. It is one of the frameworks and acting repositories that host all of the reusable PHP components. Alongside containing some of the PHP libraries, it also provides you with a simple interface in PHP to automatically install packages.

## **How is a PHP script executed?**

PHP scripts can be easily executed from the command-line interface (CLI). The syntax is as follows:

```
php filename.php
```

Here, **filename** refers to the file that contains scripts. The extension **.php** is needed alongside the filename.

## **What are the types of variables present in PHP?**

- **Array:** A named and ordered collection of data
- **Boolean:** A logical value (True or False)

- **Double:** Floating point numbers such as 5.1525
- **Integer:** Whole numbers without a floating point
- **Object:** An instance of classes, containing data and functions
- **NULL:** A special data type, supporting only the NULL data
- **Resource:** Special variables that hold references to external resources
- **String:** A sequence of characters such as, "Hello learners!"

### **What are the variable-naming rules you should follow in PHP?**

- Variables can only begin with letters or underscores.
- Special characters such as +, %, -, &, etc. cannot be used.

### **What are the main characteristics of a PHP variable?**

- Variables can be declared before the value assignment.
- A variable value assignment happens using the '=' operator.
- Every variable in PHP is denoted with a \$ (dollar) sign.
- The value of a variable depends on its latest assigned value.
- PHP variables are not intrinsic. There is no explicit declaration.

### **What is NULL in PHP?**

NULL is a special data type in PHP used to denote the presence of only one value, NULL. You cannot assign any other value to it.

NULL is not case sensitive in PHP and can be declared in two ways as shown below:

`$var = NULL;`

Or

`$var = null;`

Both of the above syntaxes work fine in PHP.

### **How are constants defined in PHP?**

Constants can be defined easily in PHP by making use of the **define()** function. This function is used to define and pull out the values of the constants easily.

Constants, as the name suggests, cannot be changed after being definition. They do not require the PHP syntax of starting with the conventional \$ sign.

### **What is the use of the constant() function in PHP?**

The constant() function is used to retrieve the values predefined in a constant variable. It is used especially when you do not know the name of the variable.

### **Differentiate between variables and constants in PHP.**

Variable	Constant
Variables can have changed paths	Constants cannot be changed
The default scope is the current access scope	Constants can be accessed throughout without any scoping rules
The \$ assignment is used for definition	Constants are defined using the define() function
Compulsory usage of the \$ sign at the start	No need for the \$ sign for constants

### **What does the phrase 'PHP escape' mean?**

PHP escape is a mechanism that is used to tell the PHP parser that certain code elements are different from the PHP code. This provides the basic means to differentiate a piece of PHP code from the other aspects of the program.

### **How are two objects compared in PHP?**

PHP provides you with the '==' operator, which is used to compare two objects at a time. This is used to check if there is a common presence of attributes and values between the objects in comparison.

The '===' operator is also used to compare if both objects in consideration are referencing to the same class.

### **What is the meaning of break and continue statements in PHP?**

**Break:** This statement is used in a looping construct to terminate the execution of the iteration and to immediately execute the next code snippet outside the scope of the iterating statement.

**Continue:** This statement is used to skip the current iteration of the loop and continue to execute the next iteration until the looping construct is exited.

### How does JavaScript interact with PHP?

JavaScript is a client-side programming language, while PHP is a server-side scripting language. PHP has the ability to generate JavaScript variables, and this can be executed easily in the browser, thereby making it possible to pass variables to PHP using a simple URL.

### How does the 'foreach' loop work in PHP?

The foreach statement is a looping construct used in PHP to iterate and loop through the array data type. The working of foreach is simple; with every single pass of the value, elements get assigned a value and pointers are incremented. This process is repeated until the end of the array.

The following is the syntax for using the foreach statement in PHP:

```
foreach(array)
{
Code inside the loop;
}
```

### Differentiate between require() and require\_once() functions.

require()	require_once()
The inclusion and evaluation of files	Includes files if they are not included before
Preferred for files with fewer functions	Preferred when there are a lot of functions

### Is it possible to set infinite execution time in PHP?

Yes, it is possible to have an infinite execution time in PHP for a script by adding the **set\_time\_limit(0)** function to the beginning of a script.

This can also be executed in the '**php.ini**' file if not at the beginning of the script.

## **What is the most used method for hashing passwords in PHP?**

The **crypt()** function is widely used for this functionality as it provides a large amount of hashing algorithms that can be used. These algorithms include [md5](#), sha1 or sha256.

## **Differentiate between an indexed array and an associative array.**

Indexed arrays have elements that contain a numerical index value.

Example: `$color=array("red","green","blue");`

Here, red is at index 0, green at 1, and blue at 2.

Associative arrays, on the other hand, hold elements with string indices as shown below:

Example:

```
$salary=array("Jacob"=>"20000","John"=>"44000","Josh"=>"60000");
```

## **What are sessions and cookies in PHP?**

Sessions are global variables that are stored on the server inside the architecture. Every single session is tagged with a unique server ID that is later used to work with the storage and retrieval of values.

Cookies are entities used to identify unique users in the architecture. It is a small file that the server plants into the client system. This is done to get useful information from the client for the development of various aspects of the server.

## **Is typecasting supported in PHP?**

Yes, typecasting is supported by PHP and can be done very easily. Following are the types that can be cast in PHP:

- **(int), (integer):** Cast to integer
- **(bool), (boolean):** Cast to boolean
- **(float), (double), (real):** Cast to float
- **(string):** Cast to string
- **(array):** Cast to array
- **(object):** Cast to object

## **Can a form be submitted in PHP without making use of a submit button?**

Yes, a form can be submitted without the explicit use of a button. This is done by making use of the JavaScript **submit()** function easily.

## **Does PHP support variable length argument functions?**

Yes, **PHP** supports the use of variable-length argument functions. This simply means that you can pass any number of arguments to a function. The syntax simply involves using three dots before the argument name as shown in the following example:

```
<?php
function add(...$num) {
    $sum = 0;
    foreach ($num as $n) {
        $sum += $n;
    }
    return $sum;
}
echo add(5, 6, 7, 8);
?>
```

Output: 26

## **What is the use of session\_start() and session\_destroy() functions?**

In PHP, the **session\_start()** function is used to start a new session. However, it can also resume an existing session if it is stopped. In this case, the return will be the current session if resumed.

Syntax:

```
session_start();
```

The **session\_destroy()** function is mostly used to destroy all of the session variables as shown below:

```
<?php
session_start();
session_destroy();
?>
```

## How can you open a file in PHP?

PHP supports file operations by providing users with a plethora of file-related functions.

In the case of opening a file, the **fopen()** function is used. This function can open a file or a URL as per requirement. It takes two arguments: **\$filename** and **\$mode**.

Syntax:

```
resource fopen ( string $filename , string $mode [, bool  
$use_include_path = false [, resource $context ]] )
```

## What is the use of \$message and \$\$message in PHP?

Both **\$message** and **\$\$message** are variables in PHP and the difference lies in the name. While **\$message** is a variable with a fixed name, **\$\$message** is a variable with a name that is actually stored in **\$message**.

Consider the following example:

If **\$message** consists of ‘var’, then **\$message** is nothing but ‘\$var’. Next up on these core PHP interview questions, you have to know a very important difference in PHP.

## Differentiate between GET and POST methods in PHP.

GET Method	POST Method
The GET method can only send a maximum of 1024 characters simultaneously	There is no restriction on the data size
GET does not support sending binary data	POST supports binary data as well as ASCII
QUERY_STRING env variable is used to access the data that is sent	The HTTP protocol and the header are used to push the data
The <b>\$_GET</b> associative array is used to access the sent information	The <b>\$_POST</b> associative array is used to access the sent information here

## **What is the use of lambda functions in PHP?**

Being an anonymous function, the lambda function is used to first store data into a variable and then to pass it as arguments for the usage in other methods or functions.

Consider the following example:

```
$input = array(2, 5, 10);
$output = array_filter($input, function ($x) { return $x > 2; });
```

The lambda function definition here:

```
function ($x) { return $x > 2; };
```

This is used further to store data into a variable, and then you can use it when required without the requirement of defining it again.

## **Differentiate between compile-time exception and runtime exception in PHP.**

As the name suggests, if there is an occurrence of any sort of exception while the script is being compiled, it is called a compile-time exception. The `FileNotFoundException` is a good example of a compile-time exception.

An exception that interrupts the script while running is called a runtime exception. The `ArrayIndexOutOfBoundsException` is an example of a runtime exception.

## **What is the meaning of type hinting in PHP?**

Type hinting is used in PHP when there is a requirement to explicitly define the data type of an argument when passing it through a function.

When this function is first called, PHP will run a quick check to analyze the presence of all the data types that are specified. If it is different, then the runtime will stop as an exception will be raised.

Next up you have to understand how to connect a URL with PHP.

## **How is a URL connected to PHP?**

Any URL can be connected to PHP easily by making use of the library called cURL. This comes as a default library with the standard installation of PHP.

The term 'cURL' stands for client-side URL, allowing users to connect to a URL and pick up information from that page to display.

## How does string concatenation work in PHP?

String concatenation is carried out easily in PHP by making use of the dot(.) operator. Consider the following example:

```
<?php $string1="Welcome"; $string2="to Intellipaat"; echo $string1 . "  
" . $string2; ?>
```

Output: Welcome to Intellipaat

## How can we encrypt a password using PHP?

The **crypt ()** function is used to create one-way encryption. It takes one input string and one optional parameter. The function is defined as:

```
crypt ($input_string, $salt)
```

Here, \$input\_string consists of the input string that has to be encrypted and \$salt is also a string parameter that is used to generate a strong password and it acts as the base of the hashing process here. PHP uses DES for encryption. The format is as follows:

```
<?php  
$password = crypt('intellipaat');  
print $password . "is the encrypted version of intellipaat";  
?>
```

## Explain how to submit a form without a submit button.

A form can be posted or submitted without the button in the following ways:

- On the OnClick event of a label in the form, a JavaScript function can be called to submit the form.

### Example:

```
document.form_name.submit()
```

- Using a Hyperlink: On clicking the link, a JavaScript function can be called.

```
<a href=" javascript:document.MyForm.submit();">
```

- **A form can be submitted in the following ways as well without using the submit button:**

- Submitting a form by clicking a link
- Submitting a form by selecting an option from the drop-down box with the invocation of an onChange event
- Using JavaScript:

```
document.form.submit();
```

- Using header("location:page.php");

### **How can we increase the execution time of a PHP script?**

- The default time allowed for a PHP script to execute is 30 seconds as mentioned in the **php.ini** file. The function used is **set\_time\_limit(int sec)**. If the value passed is '0', it takes unlimited time. It should be noted that if the default timer is set to 30 seconds and 20 seconds is specified in **set\_time\_limit()**, the script will run for 45 seconds.
- This time can be increased by modifying **max\_execution\_time** in seconds. The time must be changed keeping in mind the environment of the server. This is because modifying the execution time will affect all the sites hosted by the server.
- The script execution time can be increased by:
  - Using the **sleep()** function in the PHP script
  - Using the **set\_time\_limit()** function

The default limit is 30 seconds. The time limit can be set to zero to impose no time limit

### **What is Zend Engine?**

Zend Engine is used internally by PHP as a compiler and runtime engine. PHP Scripts are loaded into memory and compiled into Zend OPCodes.

These OPCodes are executed and the HTML generated is sent to the client.

The Zend Engine provides memory and resource management and other standard services for the PHP language. Its performance, reliability, and extensibility have played a significant role in PHP's increasing popularity.

## **What library is used for PDF in PHP?**

The PDF functions in PHP can create PDF files using PDFlib version 6. PDFlib offers an object-oriented API for PHP5 in addition to the function-oriented API for PHP4.

There is also the » Panda module.

FPDF is a PHP class, which allows generating PDF files with pure PHP (without using PDFlib). F from FPDF stands for Free: we may use it for any requirement and modify it to suit our needs. FPDF requires no extension (except zlib to activate compression and GD for GIF support) and works with PHP4 and PHP5.

## **What are the new features introduced in PHP7?**

- Zend Engine 3 performance improvements and 64-bit integer support on Windows
- Uniform variable syntax
- AST-based compilation process
- Added Closure::call()
- Bitwise shift consistency across platforms
- (Null coalesce) operator
- Unicode codepoint escape syntax
- Return type declarations
- Scalar type (integer, float, string, and Boolean) declarations

## **What is htaccess? Why do we use it and where?**

The **.htaccess** files are configuration files of Apache Server that provide a way to make configuration changes on a per-directory basis. A file, containing one or more configuration directives, is placed in a particular document directory; the directives apply to that directory and all subdirectories thereof.

These .htaccess files are used to change the functionality and features of the Apache web server.

### **For instance:**

- The .htaccess file is used for URL rewrite.
- It is used to make the site password-protected.
- It can restrict some IP addresses so that on these restricted IP addresses, the site will not open.

## What is meant by PEAR in PHP?

**PEAR** is an acronym for **PHP Extension and Application Repository**. The purpose of PEAR is to provide:

- A structured library of open-sourced code for PHP users
- A system for code distribution and package maintenance
- A standard style for writing code in PHP
- PHP Foundation Classes (PFC)
- PHP Extension Community Library (PECL)
- A website, mailing lists, and download mirrors to support the PHP/PEAR community

## Explain soundex() and metaphone().

The **soundex()** function calculates the soundex key of a string. A soundex key is a 4-character long alphanumeric string that represents the English pronunciation of a word. The soundex() function can be used for spelling applications.

```
<?php  
$str= "hello";  
Echo soundex($str);  
?>
```

The **metaphone()** function calculates the metaphone key of a string. A metaphone key represents how a string sounds if it is pronounced by an English (native) person. This function can also be used for spelling applications.

```
<?php  
echo metaphone("world");  
?>
```

## What is Smarty?

Smarty is a template engine written in PHP. Typically, these templates will include variables—like {\$variable}—and a range of logical and loop operators to allow adaptability within the templates.

## What is Memcache?

Memcache is a technology that caches objects in memory such that a web application can get to them really fast. It is used by sites, such as Digg, Facebook, and NowPublic, and is widely recognized as an essential ingredient in scaling any LAMP.

## **How can we execute a PHP script using a command line?**

We just have to run the PHP CLI (Command-line Interface) program and provide the PHP script file name as the command-line argument, for example, **php myScript.php**, assuming **php** as the command to invoke the CLI program.

We have to keep in mind that if our PHP script is written for the Web CGI interface, it may not execute properly in the command-line environment.

## **How can we know the number of days between two given dates using PHP?**

```
$date1 = date('Y-m-d'); $date2 = '2006-07-01'; $days = (strtotime($date1) - strtotime($date2)) / (60 * 60 * 24); echo "Number of days since '2006-07-01': $days";
```

## **What is meant by urlencode and urldecode?**

**urlencode()** returns the URL encoded version of the given string. URL coding converts special characters into % signs followed by two hex digits.

For example: **urlencode("10.00%")** will return **"10%2E00%25"**. URL encoded strings are safe to be used as part of URLs.

**urldecode()** returns the URL decoded version of the given string.

## **How to Get the Uploaded File Information in the Receiving Script?**

Uploaded file information is organized in **\$\_FILES** as a two-dimensional array as:

**\$\_FILES[\$fieldName]['name']** – The Original file name on the browser system.

**\$\_FILES[\$fieldName]['type']** – The file type determined by the browser.

**\$\_FILES[\$fieldName]['size']** – The Number of bytes of the file content.

**\$\_FILES[\$fieldName]['tmp\_name']** – The temporary filename of the file in which the uploaded file was stored on the server.

**\$\_FILES[\$fieldName]['error']** – The error code associated with this file upload.

## **What is the difference between mysql\_fetch\_object and mysql\_fetch\_array?**

MySQL fetch object will collect first single matching record where mysql\_fetch\_array will collect all matching records from the table in an array

## **How do you pass a variable by value?**

Just like in C++, put an ampersand in front of it, like \$a = &\$b.

## **What is the difference between ereg\_replace() and eregi\_replace()?**

eregi\_replace() function is identical to ereg\_replace() except that it ignores case distinction when matching alphabetic characters.

## **How do I find out the number of parameters passed into function?**

func\_num\_args() function returns the number of parameters passed in.

## **How can we extract string ‘abc.com’ from a string http://info@abc.com using regular expression of php?**

We can use the preg\_match() function with “/.\*@(.\*)\$/” as the regular expression pattern. For example:

```
preg_match("/.*@(.*)$/","http://info@abc.com",$data); echo $data[1];
```

## **What is the difference between the functions unlink and unset?**

unlink() is a function for file system handling. It will simply delete the file in context. unset() is a function for variable management. It will make a variable undefined.

## **How can we destroy the session, how can we unset the variable of a session?**

session\_unregister() – Unregister a global variable from the current session

session\_unset() – Free all session variables

## **How can we know the count/number of elements of an array?**

a) sizeof(\$array) – This function is an alias of count()

b) count(\$urarray) – This function returns the number of elements in an array. Interestingly if you just pass a simple var instead of an array, count() will return 1

## **How many values can the SET function of MySQL take?**

MySQL SET function can take zero or more values, but at the maximum it can take 64 values.

**What are the other commands to know the structure of a table using MySQL commands except EXPLAIN command?**

```
DESCRIBE table_name;
```

**How can we find the number of rows in a table using MySQL?**

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

**How can we find the number of rows in a result set using PHP?**

```
$result = mysql_query($any_valid_sql, $database_link);  
  
$num_rows = mysql_num_rows($result);  
  
echo "$num_rows rows found";
```

**What is the difference between CHAR and VARCHAR data types?**

CHAR is a fixed length data type. CHAR(n) will take n characters of storage even if you enter less than n characters to that column. For example, “Hello!” will be stored as “Hello! ” in CHAR(10) column.

VARCHAR is a variable length data type. VARCHAR(n) will take only the required storage for the actual number of characters entered to that column. For example, “Hello!” will be stored as “Hello!” in VARCHAR(10) column.

**mysql\_fetch\_array(), mysql\_fetch\_object(), mysql\_fetch\_row()?**

**mysql\_fetch\_array** – Fetch a result row as an associative array and a numeric array.

**mysql\_fetch\_object** – Returns an object with properties that correspond to the fetched row and moves the internal data pointer ahead. Returns an object with properties that correspond to the fetched row, or FALSE if there are no more rows

**mysql\_fetch\_row()** – Fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

**What is the difference between htmlentities() and htmlspecialchars()?**

**htmlspecialchars()** – Convert some special characters to HTML entities (Only the most widely used)

**htmlentities()** – Convert ALL special characters to HTML entities

**How can we get the properties (size, type, width, height) of an image using php image functions?**

image size use getimagesize() function

image width use imagesx() function

image height use imagesy() function

**Explain the ternary conditional operator in PHP?**

Expression preceding the ? is evaluated, if it's true, then the expression preceding the : is executed, otherwise, the expression following : is executed.

**How many ways can we get the value of current session id?**

session\_id() returns the session id for the current session.

**How can we get the browser properties using php?**

```
<?php  
echo $_SERVER['HTTP_USER_AGENT'] . "\n\n";  
$browser = get_browser(null, true);  
print_r($browser);  
?>
```

**How can we know that a session is started or not?**

A session starts by session\_start()function. this session\_start() is always declared in header portion.it always declares first.then we write session\_register().

**What is the use of obj\_start()?**

Its initializing the object buffer, so that the whole page will be first parsed (instead of parsing in parts and thrown to browser gradually) and stored in output buffer so that after complete page is executed, it is thrown to the browser once at a time.

**39. What is the difference between Split and Explode?**

split()-used for JavaScript for processing the string and the explode()-used to convert

the String to Array, implode()-used for convert the array to String

Here the Example <?php \$x="PHP is a ServerSide Scripting Language"; \$c=explode(" ",\$x); print\_r(\$c); \$d=implode(" ",\$c); echo ".\$d; ?> Javascript Example: list(\$month, \$day, \$year) = split('/.-/', \$date);

### **What is the use of sprintf() function?**

The sprintf() function writes a formatted string to a variable.

### **What is the difference between Notify URL and Return URL?**

Notify URL is used to just notify the status while processing. Return URL is used to return after processing.

### **What is the difference between ucfirst and ucwords?**

ucfirst() to convert the first letter of every string to uppercase, and ucwords(), to convert the first letter of every word in the string to uppercase.

### **What is meant by nl2br()?**

nl2br() inserts a HTML tag <br> before all new line characters \n in a string.

### **How To Read the Entire File into a Single String?**

```
<?php  
  
$file = file_get_contents("/windows/system32/drivers/etc/services");  
  
print("Size of the file: ".strlen($file)."n");  
  
?>
```

### **What are the different functions in sorting an array?**

Sorting functions in PHP: asort() arsort() ksort() krsort() uksort() sort() natsort() rsort()

### **List some of the features of PHP7.**

- Scalar type declarations
- Return type declarations

- Null coalescing operator (??)
- Spaceship operator
- Constant arrays using define()
- Anonymous classes
- Closure::call method
- Group use declaration
- Generator return expressions
- Generator delegation
- Space ship operator

### **What are the ways to define a constant in PHP?**

PHP constants are name or identifier that can't be changed during execution of the script. PHP constants are defined in two ways:

- Using define() function
- Using const() function

### **How many data types are there in PHP?**

PHP data types are used to hold different types of data or values. There are 8 primitive data types which are further categorized in 3 types:

- Scalar types
- Compound types
- Special types

What is the use of header() function in PHP?

The header() function is used to send a raw HTTP header to a client. It must be called before sending the actual output. For example, you can't print any HTML element before using this function.

### **Explain PHP parameterized functions.**

PHP parameterized functions are functions with parameters. You can pass any number of parameters inside a function. These given parameters act as variables inside your function. They are specified inside the parentheses, after the function name. Output depends upon dynamic values passed as parameters into the function.

## **Explain PHP variable length argument function.**

PHP supports variable length argument function. It means you can pass 0, 1 or n

## **Explain setcookie() function in PHP?**

PHP setcookie() function is used to set cookie with HTTP response. Once the cookie is set, you can access it by \$\_COOKIE superglobal variable.

1. bool setcookie ( string \$name [, string \$value [, int \$expire = 0 [, string \$path  
2. [, string \$domain [, bool \$secure = false [, bool \$httponly = false ]]]]] ] )

## **What is \$\_SESSION in PHP?**

A session creates a file in a temporary directory on the server where registered session variables and their session id are stored. This data will be available to all pages on the site amid that visit.

The area of the temporary record is controlled by a setting in the php.ini document called session.save\_path.

At the point when a session is begun following things happen –

1. PHP first makes two duplicates of one of a kind session id for that particular session of the client which is an arbitrary string of 32 hexadecimal numbers, for example, 3c7foj34c3jjhkyepop2fc937e3443.
2. One copy of unique session id automatically sent to the user's computer for the sake of synchronization in future ahead, and one copy is being maintained at server side till the session is running.
3. Whenever you want to access the page of website or web app, then session id of the current user will be associated with the HTTP header, and that will be compared by the session id which is being maintained at the server. After completing the comparison process, you can easily access the page of the website or web app
4. A session ends when the user closes the browser, or after leaving the site, the server will terminate the session after a predetermined period, commonly 30 minutes duration.

## **Write syntax to open a file in PHP?**

PHP fopen() function is used to open file or URL and returns resource. It accepts two arguments: \$filename and \$mode.

```
resource fopen ( string $filename , string $mode [, bool $use_include_path = false [, resource $context ]] )
```

## **How to read a file in PHP?**

PHP provides various functions to read data from the file. Different functions allow you to read all file data, read data line by line, and read data character by character.

PHP file read functions are given below:

- fread()
- fgets()
- fgetc()

## **How to write in a file in PHP?**

PHP fwrite() and fputs() functions are used to write data into file. To write data into a file, you need to use w, r+, w+, x, x+, c or c+ mode.

## **How to upload file in PHP?**

The move\_uploaded\_file() function is used to upload file in PHP.

1. bool move\_uploaded\_file ( string \$filename , string \$destination )

## **How to download file in PHP?**

The readfile() function is used to download the file in PHP.

```
int readfile ( string $filename )
```

## **What is the meaning of a Persistent Cookie?**

A persistent cookie is permanently stored in a cookie file on the browser's computer. By default, cookies are temporary and are erased if we close the browser.

## **What is the use of the function ‘imagetypes()’?**

imagetypes() gives the image format and types supported by the current version of GD-PHP.

## **What is Cookies? How to create cookies in PHP?**

A cookie is used to identify a user. A cookie is a little record that the server installs on the client's Computer. Each time a similar PC asks for a page with a program, it will send the cookie as well. With PHP, you can both make and recover cookie value.

### **Some important points regarding Cookies:**

1. Cookies maintain the session id generated at the back end after verifying the user's identity in encrypted form, and it must reside in the browser of the machine
2. You can store only string values not object because you can't access any object across the website or web apps
3. Scope: – Multiple pages.
4. By default, cookies are temporary and transitory cookie saves in the browser only.
5. By default, cookies are URL particular means Gmail isn't supported in Yahoo and the vice versa.
6. Per site 20 cookies can be created in one website or web app
7. The Initial size of the cookie is 50 bytes.
8. The Maximum size of the cookie is 4096 bytes.

## **What does the PHP error ‘Parse error in PHP – unexpected T\_variable at line x’ means?**

This is a PHP syntax error expressing that a mistake at the line x stops parsing and executing the program.

## **What should we do to be able to export data into an Excel file?**

The most common and used way is to get data into a format supported by Excel. For example, it is possible to write a .csv file, to choose for example comma as a separator between fields and then to open the file with Excel.

## **What is the function file\_get\_contents() useful for?**

file\_get\_contents() lets reading a file and storing it in a string variable.

## **What is the function mysql\_pconnect() useful for?**

mysql\_pconnect() ensure a persistent connection to the database, it means that the connection does not close when the PHP script ends.

This function is not supported in PHP 7.0 and above

## **How is it possible to know the number of rows returned in the result set?**

The function mysqli\_num\_rows() returns the number of rows in a result set.

## **Which function gives us the number of affected entries by a query?**

mysqli\_affected\_rows() return the number of entries affected by an SQL query.

## **How can we check the value of a given variable is a number?**

It is possible to use the dedicated function, is\_numeric() to check whether it is a number or not.

## **How can we check the value of a given variable is alphanumeric?**

It is possible to use the dedicated function, ctype\_alnum to check whether it is an alphanumeric value or not.

## **How do I check if a given variable is empty?**

If we want to check whether a variable has a value or not, it is possible to use the empty() function.

## **How do I escape data before storing it in the database?**

The addslashes function enables us to escape data before storage into the database.

## **How is it possible to remove escape characters from a string?**

The stripslashes function enables us to remove the escape characters before apostrophes in a string.

## **How can we automatically escape incoming data?**

We have to enable the Magic quotes entry in the configuration file of PHP.

## **What does the function get\_magic\_quotes\_gpc() means?**

The function get\_magic\_quotes\_gpc() tells us whether the magic quotes is switched on or no.

## **Is it possible to remove the HTML tags from data?**

The strip\_tags() function enables us to clean a string from the HTML tags.

## **How can we define a variable accessible in functions of a PHP script?**

This feature is possible using the global keyword.

## **How is it possible to return a value from a function?**

A function returns a value using the instruction 'return \$value;'

## **Which cryptographic extension provide generation and verification of digital signatures?**

The PHP-OpenSSL extension provides several cryptographic operations including generation and verification of digital signatures.

## **How is a constant defined in a PHP script?**

The define() directive lets us defining a constant as follows:

```
define ("ACONSTANT", 123);
```

## **How is the ternary conditional operator used in PHP?**

It is composed of three expressions: a condition, and two operands describing what instruction should be performed when the specified condition is true or false as follows:

```
Expression_1?Expression_2 : Expression_3;
```

## **What is the function func\_num\_args() used for?**

The function func\_num\_args() is used to give the number of parameters passed into a function.

## **In PHP, objects are they passed by value or by reference?**

In PHP, objects are passed by reference.

## **Are Parent constructors called implicitly inside a class constructor?**

No, a parent constructor have to be called explicitly as follows:

```
parent::constructor($value)
```

## **What's the difference between \_\_sleep and \_\_wakeup?**

`__sleep` returns the array of all the variables that need to be saved, while `__wakeup` retrieves them.

## **What is faster?**

1- Combining two variables as follows:

```
$variable1 = 'Hello ';  
  
$variable2 = 'World';  
  
$variable3 = $variable1.$variable2;  
Or
```

2- `$variable3 = "$variable1$variable2";`  
`$variable3` will contain "Hello World". The first code is faster than the second code especially for large sets of data.

## **When do sessions end?**

Sessions automatically end when the PHP script finishes executing but can be manually ended using the `session_write_close()`.

## **What is the difference between `session_unregister()` and `session_unset()`?**

The `session_unregister()` function unregisters a global variable from the current session and the `session_unset()` function frees all session variables.

### **What does \$GLOBALS mean?**

\$GLOBALS is associative array including references to all variables which are currently defined in the global scope of the script.

### **What does \$\_SERVER mean?**

\$\_SERVER is an array including information created by the web server such as paths, headers, and script locations.

### **What does \$\_FILES means?**

\$\_FILES is an associative array composed of items sent to the current script via the HTTP POST method..

### **How can we get the error when there is a problem to upload a file?**

\$\_FILES['userfile']['error'] contains the error code associated with the uploaded file.

### **What does \$\_ENV mean?**

\$\_ENV is an associative array of variables sent to the current PHP script via the environment method.

### **What does the scope of variables mean?**

The scope of a variable is the context within which it is defined. For the most part, all PHP variables only have a single scope. This single scope spans included and required files as well.

### **what the difference between the 'BITWISE AND' operator and the 'LOGICAL AND' operator?**

\$a and \$b: TRUE if both \$a and \$b are TRUE.

\$a & \$b: Bits that are set in both \$a and \$b are set.

### **What are the two main string operators?**

The first is the concatenation operator ('.'), which returns the concatenation of its right and left arguments. The second is ('.='), which appends the argument on the right to the argument on the left.

### **What does the array operator '===' means?**

`$a === $b` TRUE if \$a and \$b have the same key/value pairs in the same order and of the same types.

### **What is the differences between `$a != $b` and `$a !== $b`?**

`!=` means inequality (TRUE if \$a is not equal to \$b) and `!==` means non-identity (TRUE if \$a is not identical to \$b).

### **How can we determine whether a PHP variable is an instantiated object of a certain class?**

To be able to verify whether a PHP variable is an instantiated object of a certain class we use `instanceof`.

### **What is the goto statement useful for?**

The goto statement can be placed to enable jumping inside the PHP program. The target is pointed by a label followed by a colon, and the instruction is specified as a `goto` statement followed by the desired target label.

### **what is the difference between `Exception::getMessage` and `Exception::getLine`?**

`Exception::getMessage` lets us getting the Exception message and `Exception::getLine` lets us getting the line in which the exception occurred.

### **What does the expression `Exception::__toString` means?**

`Exception::__toString` gives the String representation of the exception.

### **How is it possible to parse a configuration file?**

The function `parse_ini_file()` enables us to load in the ini file specified in filename and returns the settings in it in an associative array.

### **How can we determine whether a variable is set?**

The boolean function `isset` determines if a variable is set and is not NULL.

## **What is the difference between the functions strstr() and striistr()?**

The string function strstr(string allString, string occ) returns part of allString from the first occurrence of occ to the end of allString. This function is case-sensitive. striistr() is identical to strstr() except that it is case insensitive.

## **Is it possible to protect special characters in a query string?**

Yes, we use the urlencode() function to be able to protect special characters.

## **What are the three classes of errors that can occur in PHP?**

The three basic classes of errors are notices (non-critical), warnings (serious errors) and fatal errors (critical errors).

## **What is the default session time in PHP?**

The default session time in php is until the closing of the browser

## **Is it possible to use COM component in PHP?**

Yes, it's possible to integrate (Distributed) Component Object Model components ((D)COM) in PHP scripts which is provided as a framework.

## **Explain how you can update Memcached when you make changes to PHP?**

When PHP changes you can update Memcached by

- **Clearing the Cache proactively:** Clearing the cache when an insert or update is made
- **Resetting the Cache:** It is similar to the first method but rather than just deleting the keys and waiting for the next request for the data to refresh the cache, reset the values after the insert or update.

## **Differentiate between variables and constants in PHP**

Variables	Constants
The value of a variable can be changed during the execution.	The constant value can't be changed during script execution.

Variables	Constants
Variables require compulsory usage of the \$ sign at the start.	No dollar sign (\$) is required before using a constant.
It is possible to define a variable by simple assignment.	Constants can't be defined by simple assignments. They are defined using the define() function.
The default scope is the current access scope.	Constants can be accessed throughout without any scoping rules.

### What is the difference between “echo” and “print” in PHP?

echo	print
echo can output one or more strings.	print can only output one string and it always returns 1.
echo is faster than print because it does not return any value.	print is slower compared to echo.
If you want to pass more than one parameter to echo, a parenthesis should be used.	Use of parenthesis is not required with the argument list.

### Tell me some of the disadvantages of PHP

- PHP is not suitable for giant content-based web applications.
- Since it is open-source, it is not secure. Because ASCII text files are easily available.
- Change or modification in the core behavior of online applications is not allowed by PHP.
- If we use more features of the PHP framework and tools, it will cause poor performance of online applications.
- PHP features a poor quality of handling errors. PHP lacks debugging tools, which are needed to look for warnings and errors. It has only a few debugging tools in comparison to other programming languages.

## **Explain the importance of Parser in PHP?**

A PHP parser is software that converts source code into the code that computer can understand. This means whatever set of instructions we give in the form of PHP code is converted into a machine-readable format by the parser.

You can parse PHP code with PHP using the token\_get\_all() function.

## **How to connect to a URL in PHP?**

Any URL can be connected to PHP easily by making use of the library called cURL. This comes as a default library with the standard installation of PHP.

The term cURL stands for client-side URL. cURL make use of libcurl(client-side URL Transfer Library) which supports many protocols like FTP, FTPS, HTTP/1, HTTP POST, HTTP PUT, HTTP proxy, HTTPS, IMAP, Kerberos etc. It allows you to connect to a URL and retrieve and display information from that page – like the HTML content of the page, HTTP headers, and their associated data, etc.

### **Steps for connecting with URL using PHP cURL POST are given below:**

- Initialize cURL session.
- Define your URL where you want to post the request. We can directly enter this URL into the URL section inset option parameter or we can assign it to an object.
- Now, define the cURL options that you want to execute with the post option.
- After setting all the functions then it's time to execute our cURL.
- After this, close the cURL and echo your object to check their response.

```
//Step 1 To initialize curl
    $ch = curl_init();
//Step 2 To set url where you want to post
    $url = 'http://www.localhost.com';
//Step 3 Set curl functions which are needs to you
    curl_setopt($ch,CURLOPT_URL,$url);
    curl_setopt($ch,CURLOPT_POST,true);
    curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
    curl_setopt($ch,CURLOPT_POSTFIELDS,'postv1 = value1&postv2 =
value2');
//Step 4 To execute the curl
    $result = curl_exec($ch);
//Step 5 Close curl
    curl_close($ch);
```

## How to create API in PHP?

API stands for Application Programming Interface. It defines the functions and variables. Communication between the database via PHP extensions is handled by API.

Now, REST API is the web architecture that uses HTTP protocol for exchanging data between two functions that means your application or system. Now, let us have a look at how to create REST API in PHP by considering the example of accessing data from a database using PHP script.

**Step 1 – Create a database:** To create a database run the query given below:

```
CREATE DATABASE phptest;
```

**Step 2 – Create a table:** After creating a database, you have to create a table with dummy data. To create a table run the query given below:

```
CREATE TABLE IF NOT EXISTS `transactions`  
(  
    `id` int(20) NOT NULL AUTO_INCREMENT,  
    `order_id` int(50) NOT NULL,  
    `amount` decimal(9,2) NOT NULL,  
    `response_code` int(10) NOT NULL,  
    `response_desc` varchar(50) NOT NULL,  
    PRIMARY KEY (`id`),  
    UNIQUE KEY `order_id` (`order_id`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ;
```

**Step 3 – Create a Database Connection:** Create a db.php file and paste the below-given database connection in it. Make sure to update these credentials with your database credentials.

```
<?php  
// Enter your Host, username, password, database below.  
$con = mysqli_connect("localhost", "root", "", "phptest");  
if (mysqli_connect_errno())  
{  
    echo "Failed to connect to MySQL: " . mysqli_connect_error();  
    die();  
}  
?>
```

**Step 4 – Create a REST API File:** Create an api.php file and copy the following script in it.

```
<?php  
header("Content-Type:application/json");
```

```

if (isset($_GET['order_id']) && $_GET['order_id']!="")
{
include('db.php');
$order_id = $_GET['order_id'];
$result = mysqli_query($con,
    "SELECT * FROM `transactions` WHERE order_id=$order_id");
if(mysqli_num_rows($result)>0)
{
    $row = mysqli_fetch_array($result);
    $amount = $row['amount'];
    $response_code = $row['response_code'];
    $response_desc = $row['response_desc'];
    response($order_id, $amount, $response_code, $response_desc);
    mysqli_close($con);
}
else
{
    response(NULL, NULL, 200,"No Record Found");
}
}
else
{
response(NULL, NULL, 400,"Request is invalid");
}
function response($order_id,$amount,$response_code, $response_desc)
{
$response['order_id'] = $order_id;
$response['amount'] = $amount;
$response['response_code'] = $response_code;
    $response['response_desc'] = $response_desc;
$json_response = json_encode($response);
echo $json_response;
}
?>

```

The above code will accept the GET request and return output in the JSON format.

## Differentiate between GET and POST

GET	POST
GET method is used for requesting data from a specified resource.	POST is used for sending the data to the server as a package in a separate communication with the processing script.
Data is sent in the form of URL parameters which are strings of name-value pairs separated by ampersands(&)	Data sent through the POST method will not be seen in the URL
GET method cannot be used for sending binary data like images or word documents	The POST method can be used to send ASCII as well as binary data like images and word documents
This method must not be used if you have any sensitive information like a password to be sent to the server.	Sensitive information can be sent using this method.
It can be used for submitting the form where the user can bookmark the result.	Submissions by form with POST cannot be bookmarked.
You can use this method only for data that is not secure.	Data sent through this method is secure.
GET method is not safer since parameters may be stored in web server logs or browser history.	POST method is safer than GET because the parameters are not stored in web server logs or browser history.

## Why is object-oriented programming so popular?

There are certain points due to which OOPs become so popular.

- It provides a better programming style, so you don't need to write code again and again which you want to run, just make a class of the object and call it.
- As it supports the inheritance concept, an application created with OOPs can inherit another class property.
- It provides a modularity model that means if you change any part of code that is in a separate module, that won't impact any other module.

- If you are stuck somewhere, this language allows your problems to break down into bite-sized pieces so that you can solve them.
- Due to its polymorphism feature, a single class can be used to create different objects, and that too from the same piece of code.

## **What are the types of Class Variables?**

**We have three different types of Class Variables in OOPs.**

### *1. Local Variables*

These types of variables are declared locally in methods, blocks, and constructors. These variables are created when program control enters the methods, blocks, and constructor and are destroyed once program control leaves them.

### *2. Instance Variables*

These variables are declared outside a block, constructor, or method. These are created once a class object is created and destroyed when the object is destroyed.

### *3. Static Variables*

Static variables are also called class variables and are defined using the static keyword. These are declared within a class but outside a code block and a method. The creation of these variables starts when the program starts and is destroyed when the program ends.

## **What is the difference between Method overriding and Method overloading?**

<b>Method Overloading</b>	<b>Method Overriding</b>
It is the concept where we define two or more methods by the same name but with different signatures.	In method overriding we define two or more identical methods which have the same name and signatures.
The binding of the method is done at compile time.	The binding of the method is done at run time.
There are no class restrictions i.e. it can be achieved in the same or different classes.	There are class restrictions in it i.e. it can only be achieved in different classes.

## **What is virtual and pure virtual function?**

<b>Virtual Function</b>	<b>Pure Virtual Function</b>
Virtual Function has its definition hidden in the Base class	They have no definition in the Base class.
The derived class can override the virtual function if required.	In the case of a pure virtual function, the derived class has to override it.
If the derived class fails to override then it has the option to use the virtual function of the base class	It will throw a compilation error if it fails to override pure virtual function.

## **List out some of the differences between a class and an object?**

<b>CLASS</b>	<b>OBJECT</b>
It is a blueprint from which different instances (objects) are created.	Objects are the instances of the class.
Class is a logical entity.	Objects are physical entities.
Users can declare class only once.	Objects can be declared multiple times depending upon the requirements.
When a class is created there is no memory allocation	Objects allocated memory.
Class is a group of objects.	Objects are real-world entities such as pen, copy, mouse, etc.
Class can be declared using class keyword e.g class Student {}	Objects can be declared using the new keyword e.g. Student s1=new Student();

## **What are the main features of OOPs?**

**You have to list these four features of OOPs while answering this question-**

- Inheritance
- Polymorphism
- Encapsulation
- Data Abstraction

## What is Encapsulation in oops with an example?

The process of binding up data and functions together that manipulates them is known as encapsulation in OOPS. It protects the data from outside interference and misuse.

Let's understand the concept of encapsulation with both real-world examples and with the help of a program.

It is an attribute of an object, and it contains all data which is hidden. That hidden data is restricted to the members of that class.

```
class Account {  
    private int account_number;  
    private int account_balance;  
  
    public void show Data() {  
        //code to show data  
    }  
  
    public void deposit(int a) {  
        if (a < 0) {  
            //show error  
        } else  
            account_balance = account_balance + a;  
    }  
}
```

## What is Polymorphism and its types in OOPS?

Polymorphism is one of the core concepts which is used in object-oriented programming. Polymorphism generally shows the relationships between parent class objects and child class objects.

Types of polymorphism in OOPs

- **Compile Time Polymorphism**– It is also known as Static Binding and allows the programmer to implement various methods. Names can be the same but their parameters should be different.
- **Runtime Polymorphism**– It is also known as Dynamic Binding and allows the programmer to call a single overridden method during the runtime of the program.

## **What is Inheritance?**

It is a technique in which one class acquires the property of another class. With the help of inheritance, we can reuse the fields and methods of the existing class.

**Inheritance** has three type, are given below.

- Single inheritance
- Multiple inheritance
- Multi level inheritance

But PHP supports only **single inheritance**, where only one class can be derived from single parent class. We can also use multiple inheritance by using **interfaces**.

## **What is constructor and destructor?**

Constructor and Destructor both are special functions which are automatically called when an object is created and destroyed.

*Example*

### **Constructor**

```
classAnimal  
{  
    public $name ="No-name animal";  
    public function __construct(){  
        echo "I'm alive!";  
    }  
}
```

### **Destructor**

```
classAnimal{  
    public $name ="No-name animal";
```

```

public function __construct($name){
    echo "I'm alive!";
    $this->name = $name;
}

public function __destruct(){
    echo "I'm dead now :(";
}

$animal = newAnimal("Bob");
echo "Name of the animal: ". $animal->name;

```

## What is different types of Visibility?

PHP have three access modifiers such as public, private and protected.

- **public** scope of this variable or function is available from anywhere, other classes and instances of the object.
- **private** scope of this variable or function is available in its own class only.
- **protected** scope of this variable or function is available in all classes that extend current class including the parent class.

## What are the Final class and Final methods?

**Final Class**– A class that can't be extended and inherited further is known as Final Class. This class is declared with the keyword final and should be declared.

**Final Method**– Methods in the final class are implicitly final and if a user uses the final keyword that means methods can't be overridden by subclasses.

*Example*

```

class childClassname extends parentClassname {
    protected $numPages;
}

```

```
public function __construct($author, $pages) {  
    $this->_author = $author;  
    $this->numPages = $pages;  
}  
  
final public function PageCount() {  
    return $this->numPages;  
}  
}
```

### **What is static keyword ?**

When a variable or function declared as static then it cannot be accessed with an instantiated class object. It is treated as public if no visibility is defined. It can also be used to define static variables and for late static bindings.

### **What are the difference between overloading and overriding in oops?**

**Overloading** : It occurs when two or more methods in one class have the same method name but different parameters.

**Overriding** : It means having two methods with the same method name and parameters. One of the methods is in the parent class and the other is in the child class.

### **What is "this"?**

It refers to the current object of a class.

### **What are the difference between abstract class and interface in OOPS?**

There are many differences between abstract class and interface in PHP.

1. **Abstract methods** can be declared with public, private and protected. But in case of **Interface methods** declared with public.
2. **Abstract classes** can have constants, members, method stubs and defined methods, but **interfaces** can only have constants and method stubs.
3. **Abstract classes** does not support multiple inheritance but **interface** supports this.
4. **Abstract classes** can contain constructors but **interface** does not support constructors.

## **What is namespace in PHP?**

*It allows us to use the same function or class name in different parts of the same program without causing a name collision.*

```
namespace MyAPP;
function output() {
echo 'IOS!';
}
namespace MyNewAPP;
function output(){
echo 'RSS!';
}
```

## **What is traits? How it is used in php?**

**Traits** is a group of methods that reuse in single inheritance. A Trait is intended to reduce some limitations of single inheritance by enabling a developer to reuse sets of methods.

*Example*

```
trait HelloWorld
{
    use Hello, World;

    class MyWorld {
        use HelloWorld;

        $world = new MyWorld();

        echo $world->sayHello() . " " . $world->sayWorld(); //Hello World
    }
}
```

## **What are the advantages of OOPS in PHP?**

- Code Resusability
- Flexibility

- Maintainability
- Security
- Testability

### **What is member function?**

Member function defined inside a class and are used to access object data.

### **What is Encapsulation?**

Encapsulation is an attribute of an object, and it contains all data which is hidden.

That hidden data can be restricted to the members of that class. Levels are Public, Protected, Private, Internal and Protected Internal.

### **What is Polymorphism?**

Polymorphism is nothing but assigning behaviour or value in a subclass to something that was already declared in the main class. Simply, polymorphism takes more than one form.

### **What is Inheritance?**

Inheritance is a concept where one class shares the structure and behaviour defined in another class. If inheritance applied on one class is called Single Inheritance, and if it depends on multiple classes, then it is called multiple Inheritances.

### **What are manipulators?**

Manipulators are the functions which can be used in conjunction with the insertion (<<) and extraction (>>) operators on an object. Examples are endl and setw.

### **Define a constructor?**

Constructor is a method used to initialize the state of an object, and it gets invoked at the time of object creation. Rules for constructor are.

- Constructor Name should be same as class name.
- Constructor must have no return type.

## **Define Destructor?**

Destructor is a method which is automatically called when the object is made of scope or destroyed. Destructor name is also same as class name but with the tilde symbol before the name.

## **What is Inline function?**

Inline function is a technique used by the compilers and instructs to insert complete body of the function wherever that function is used in the program source code.

## **What is friend function?**

Friend function is a friend of a class that is allowed to access to Public, private or protected data in that same class. If the function is defined outside the class cannot access such information.

Friend can be declared anywhere in the class declaration, and it cannot be affected by access control keywords like private, public or protected.

## **What is function overloading?**

Function overloading is defined as a normal function, but it has the ability to perform different tasks. It allows creation of several methods with the same name which differ from each other by type of input and output of the function.

Example

```
void add(int& a, int& b);  
  
void add(double& a, double& b);  
  
void add(struct bob& a, struct bob& b);
```

## **What is operator overloading?**

Operator overloading is a function where different operators are applied and depends on the arguments. Operator,-,\* can be used to pass through the function , and it has their own precedence to execute.

Example:

```
class complex { double real, imag;  
public: complex(double r, double i) : real(r), imag(i) {}  
  
complex operator+(complex a, complex b); complex operator*(complex a, complex  
b); complex& operator=(complex a, complex b); }
```

a=1.2, b=6

## **What is an abstract class?**

An abstract class is a class which cannot be instantiated. Creation of an object is not possible with abstract class, but it can be inherited. An abstract class can contain only Abstract method.

## **What is a ternary operator?**

Ternary operator is said to be an operator which takes three arguments. Arguments and results are of different data types, and it is depends on the function. Ternary operator is also called as conditional operator.

## **What is the use of finalize method?**

Finalize method helps to perform cleanup operations on the resources which are not currently used. Finalize method is protected, and it is accessible only through this class or by a derived class.

## **What are different types of arguments?**

A parameter is a variable used during the declaration of the function or subroutine and arguments are passed to the function, and it should match with the parameter defined. There are two types of Arguments.

- Call by Value – Value passed will get modified only inside the function, and it returns the same value whatever it is passed it into the function.
- Call by Reference – Value passed will get modified in both inside and outside the functions and it returns the same or different value.

### **What is super keyword?**

Super keyword is used to invoke overridden method which overrides one of its superclass methods. This keyword allows to access overridden methods and also to access hidden members of the superclass.

It also forwards a call from a constructor to a constructor in the superclass.

### **What is method overriding?**

Method overriding is a feature that allows sub class to provide implementation of a method that is already defined in the main class. This will overrides the implementation in the superclass by providing the same method name, same parameter and same return type.

### **What is an interface?**

An interface is a collection of abstract method. If the class implements an inheritance, and then thereby inherits all the abstract methods of an interface.

### **What is exception handling?**

Exception is an event that occurs during the execution of a program. Exceptions can be of any type – Run time exception, Error exceptions. Those

exceptions are handled properly through exception handling mechanism like try, catch and throw keywords.

### **What are tokens?**

Token is recognized by a compiler and it cannot be broken down into component elements. Keywords, identifiers, constants, string literals and operators are examples of tokens.

Even punctuation characters are also considered as tokens – Brackets, Commas, Braces and Parentheses.

## **Difference between overloading and overriding?**

Overloading is static binding whereas Overriding is dynamic binding. Overloading is nothing but the same method with different arguments, and it may or may not return the same value in the same class itself.

Overriding is the same method names with same arguments and return types associates with the class and its child class.

## **Difference between class and an object?**

An object is an instance of a class. Objects hold any information, but classes don't have any information. Definition of properties and functions can be done at class and can be used by the object.

Class can have sub-classes, and an object doesn't have sub-objects.

## **What is an abstraction?**

Abstraction is a good feature of OOPS, and it shows only the necessary details to the client of an object. Means, it shows only necessary details for an object, not the inner details of an object. Example – When you want to switch on television, it is not necessary to show all the functions of TV. Whatever is required to switch on TV will be shown by using abstract class.

## **What are access modifiers?**

Access modifiers determine the scope of the method or variables that can be accessed from other various objects or classes. There are 5 types of access modifiers, and they are as follows.

- · Private.
- · Protected.
- · Public.
- · Friend.
- · Protected Friend.

## **What is sealed modifiers?**

Sealed modifiers are the access modifiers where it cannot be inherited by the methods. Sealed modifiers can also be applied to properties, events and methods. This modifier cannot be applied to static members.

## **How can we call the base method without creating an instance?**

Yes, it is possible to call the base method without creating an instance. And that method should be Static method.

Doing inheritance from that class. Use Base Keyword from derived class.

## **What is the difference between new and override?**

The new modifier instructs the compiler to use the new implementation instead of the base class function. Whereas, Override modifier helps to override the base class function.

## **What are the various types of constructors?**

There are three various types of constructors, and they are as follows:-

- Default Constructor – With no parameters.
- Parametric Constructor – With Parameters. Create a new instance of a class and also passing arguments simultaneously.
- Copy Constructor – Which creates a new object as a copy of an existing object.

## **What is early and late binding?**

Early binding refers to assignment of values to variables during design time whereas late binding refers to assignment of values to variables during run time.

## **What is the difference between structure and a class?**

Structure default access type is public, but class access type is private. A structure is used for grouping data whereas class can be used for grouping data and methods. Structures are exclusively used for data and it doesn't require strict validation, but classes are used to encapsulates and inherit data which requires strict validation.

## **What are all the operators that cannot be overloaded?**

Following are the operators that cannot be overloaded –

- 1. Scope Resolution (:: )

- 2. Member Selection (.)
- 3. Member selection through a pointer to function (.\* )

### **What is dynamic or run time polymorphism?**

Dynamic or Run time polymorphism is also known as method overriding in which call to an overridden function is resolved during run time, not at the compile time. It means having two or more methods with the same name, same signature but with different implementation.

### **What are base class, sub class and super class?**

Base class is the most generalized class, and it is said to be a root class.

Sub class is a class that inherits from one or more base classes.

Super class is the parent class from which another class inherits.

### **What is static and dynamic binding?**

Binding is nothing but the association of a name with the class. Static binding is a binding in which name can be associated with the class during compilation time, and it is also called as early Binding. Dynamic binding is a binding in which name can be associated with the class during execution time, and it is also called as Late Binding.

### **How to get current MySQL version?**

SELECT VERSION (); is used to get the current version of MySQL.

### **What storage engines are used in MySQL?**

Storage engines are called table types and data is stored in files using various techniques.

Technique involves:

Storage mechanism Locking levels Indexing Capabilities and functions.

### **What does a TIMESTAMP do on UPDATE CURRENT\_TIMESTAMP data type?**

TIMESTAMP column is updated with Zero when the table is created.

UPDATE CURRENT\_TIMESTAMP modifier updates the timestamp field to current time whenever there is a change in other fields of the table.

### **What does myisamchk do?**

It compresses the MyISAM tables, which reduces their disk or memory usage.

### **How do you control the max size of a HEAP table?**

Maximum size of a HEAP table can be controlled by MySQL config variable called `max_heap_table_size`.

### **What is the difference between MyISAM Static and MyISAM Dynamic?**

In MyISAM static all the fields will have fixed width. The Dynamic MyISAM table will have fields like TEXT, BLOB, etc. to accommodate the data types with various lengths.

MyISAM Static would be easier to restore in case of corruption.

### **What happens when the column is set to AUTO INCREMENT and if you reach maximum value in the table?**

It stops incrementing. Any further inserts are going to produce an error, since the key has been used already.

### **How can we find out which auto increment was assigned on Last insert?**

`LAST_INSERT_ID` will return the last value assigned by `Auto_increment` and it is not required to specify the table name.

### **How can you see all indexes defined for a table?**

Indexes are defined for the table by:

```
SHOW INDEX FROM <tablename>;
```

### **What do you mean by % and \_ in the LIKE statement?**

% corresponds to 0 or more characters, \_ is exactly one character in the LIKE statement.

### **How can we convert between Unix & MySQL timestamps?**

`UNIX_TIMESTAMP` is the command which converts from MySQL timestamp to Unix timestamp

`FROM_UNIXTIME` is the command which converts from Unix timestamp to MySQL timestamp.

## **How can we get the number of rows affected by query?**

Number of rows can be obtained by

```
[sql]SELECT COUNT (user_id) FROM users;[/sql]
```

## **What is the difference between the LIKE and REGEXP operators?**

LKE and REGEXP operators are used to express with ^ and %.

```
[sql]SELECT * FROM employee WHERE emp_name REGEXP '^b'; SELECT * FROM employee WHERE emp_name LIKE "%b";[/sql]
```

## **What is the difference between mysql\_fetch\_array and mysql\_fetch\_object?**

Following are the differences between mysql\_fetch\_array and mysql\_fetch\_object:

mysql\_fetch\_array() -Returns a result row as an associated array or a regular array from database.

mysql\_fetch\_object – Returns a result row as object from database.

## **How can we run batch mode in mysql?**

Following commands are used to run in batch mode:

```
[sql]mysql ; mysql mysql.out[/sql]
```

## **Where MyISAM table will be stored and also give their formats of storage?**

Each MyISAM table is stored on disk in three formats:

## **How MySQL Optimizes DISTINCT?**

DISTINCT is converted to a GROUP BY on all columns and it will be combined with ORDER BY clause.

```
[sql]SELECT DISTINCT t1.a FROM t1,t2 where t1.a=t2.a;[/sql]
```

## **How to enter Characters as HEX Numbers?**

If you want to enter characters as HEX numbers, you can enter HEX numbers with single quotes and a prefix of (X), or just prefix HEX numbers with (Ox).

A HEX number string will be automatically converted into a character string, if the expression context is a string.

### **How to display top 50 rows?**

In MySql, top 50 rows are displayed by using this following query:

```
[sql]SELECT * FROM LIMIT 0,50[/sql]
```

### **How many columns can be used for creating Index?**

Maximum of 16 indexed columns can be created for any standard table.

### **What is the different between NOW() and CURRENT\_DATE()?**

NOW () command is used to show current year,month,date with hours,minutes and seconds.

CURRENT\_DATE() shows current year,month and date only.

### **What are the objects can be created using CREATE statement?**

Following objects are created using CREATE statement:

DATABASE EVENT FUNCTION INDEX PROCEDURE TABLE TRIGGER USER VIEW

### **How many TRIGGERS are allowed in MySql table?**

SIX triggers are allowed in MySql table. They are as follows:

BEFORE INSERT AFTER INSERT BEFORE UPDATE AFTER UPDATE BEFORE DELETE and AFTER DELETE

### **What are the nonstandard string types?**

Following are Non-Standard string types:

TINYTEXT TEXT MEDIUMTEXT LONGTEXT

### **What are all the Common SQL Function?**

CONCAT(A, B) – Concatenates two string values to create a single string output. Often used to combine two or more fields into one single field.

FORMAT(X, D) – Formats the number X to D significant digits.

CURRDATE(), CURRTIME() – Returns the current date or time.

NOW() – Returns the current date and time as one value.

MONTH(), DAY(), YEAR(), WEEK(), WEEKDAY() – Extracts the given data from a date value.

HOUR(), MINUTE(), SECOND() – Extracts the given data from a time value.

DATEDIFF(A, B) – Determines the difference between two dates and it is commonly used to calculate age

SUBTIME(A, B) – Determines the difference between two times.

FROM\_DAYS(INT) – Converts an integer number of days into a date value.

### **What does tee command do in MySQL?**

tee followed by a filename turns on MySQL logging to a specified file. It can be stopped by command note.

### **Can you save your connection settings to a conf file?**

Yes, and name it `~/.my.conf`. You might want to change the permissions on the file to 600, so that it's not readable by others.

### **What are some good ideas regarding user security in MySQL?**

There is no user without a password. There is no user without a user name. There is no user whose Host column contains % (which here indicates that the user can log in from anywhere in the network or the Internet). There are as few users as possible (in the ideal case only root) who have unrestricted access.

### **What is SERIAL data type in MySQL?**

`BIGINT NOT NULL PRIMARY KEY AUTO_INCREMENT`

### **What happens when the column is set to AUTO INCREMENT and you reach the maximum value for that table?**

It stops incrementing. It does not overflow to 0 to prevent data losses, but further inserts are going to produce an error, since the key has been used already.

## **Explain the difference between BOOL, TINYINT and BIT.**

Prior to MySQL 5.0.3: those are all synonyms. After MySQL 5.0.3: BIT data type can store 8 bytes of data and should be used for binary data.

## **Explain the difference between FLOAT, DOUBLE and REAL.**

FLOATs store floating point numbers with 8 place accuracy and take up 4 bytes. DOUBLEs store floating point numbers with 16 place accuracy and take up 8 bytes. REAL is a synonym of FLOAT for now.

## **If you specify the data type as DECIMAL (5, 2), what's the range of values that can go in this table?**

999.99 to -99.99. Note that with the negative number the minus sign is considered one of the digits.

## **What happens if a table has one column defined as TIMESTAMP?**

That field gets the current timestamp whenever the row gets altered.

## **But what if you really want to store the timestamp data, such as the publication date of the article?**

Create two columns of type TIMESTAMP and use the second one for your real data.

## **Explain data type TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP**

The column exhibits the same behavior as a single timestamp column in a table with no other timestamp columns.

## **What does TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP data type do?**

On initialization places a zero in that column, on future updates puts the current value of the timestamp in.

## **Explain TIMESTAMP DEFAULT '2006:09:02 17:38:44' ON UPDATE CURRENT\_TIMESTAMP.**

A default value is used on initialization, a current timestamp is inserted on update of the row.

**If I created a column with data type VARCHAR (3), what would I expect to see in MySQL table?**

CHAR(3), since MySQL automatically adjusted the data type.

**How would you write a query to select all teams that won either 2, 4, 6 or 8 games?**

SELECT team\_name FROM teams WHERE team\_won IN (2, 4, 6, 8). How would you select all the users, whose phone number is null? SELECT user\_name FROM users WHERE ISNULL(user\_phonenumber);

**What are ENUMs used for in MySQL?**

You can limit the possible values that go into the table. CREATE TABLE months (month ENUM 'January', 'February', 'March'); INSERT months VALUES ('April').13. What is the difference between CHAR\_LENGTH and

LENGTH? The first is, naturally, the character count. The second is byte count. For the Latin characters the numbers are the same, but they're not the same for Unicode and other encodings.

**How quoting and escaping work in SELECT QUERY?**

SELECT 'hello', "hello", ""hello""", 'hel"lo', '\hello'.

**How we get Sum of column?**

SELECT SUM(\*) FROM [table name];

**How would you change a table to InnoDB?**

ALTER TABLE name\_file ENGINE innodb;

**How do you concatenate strings in MySQL?**

CONCAT (string1, string2, string3)

**How do you get the month from a timestamp?**

SELECT MONTH(january\_timestamp) from tablename;

## **What do % and \_ mean inside LIKE statement?**

% corresponds to 0 or more characters, \_ is exactly one character.

## **How do you get the current date in Mysql?**

```
SELECT CURRENT_DATE();
```

**You wrote a search engine that should retrieve 10 results at a time, but at the same time you'd like to know how many rows there're total. How do you display that to the user?**

```
SELECT SQL_CALC_FOUND_ROWS page_title FROM web_pages LIMIT 1,10;  
SELECT FOUND_ROWS();
```

**What does this query mean: SELECT user\_name, user\_isp FROM users LEFT JOIN isps USING (user\_id)?**

It's equivalent to saying SELECT user\_name, user\_isp FROM users LEFT JOIN isps WHERE users.user\_id=isps.user\_id

## **How do you display the list of database in mysql?**

```
SHOW DATABASES;
```

## **How do you display the structure of the table?**

```
DESCRIBE table_name;
```

## **How do you find out which auto increment was assigned on the last insert?**

SELECT LAST\_INSERT\_ID() will return the last value assigned by the auto\_increment function. Note that you don't have to specify the table name.

## **What is Like operator for and what are wild cards?**

LIKE operator is used to match patterns. A “%” sign is used to define the pattern. Below SQL statement will return all words with letter “S”

```
SELECT * FROM pcdsEmployee WHERE EmpName LIKE 'S%'
```

Below SQL statement will return all words which end with letter “S”

```
SELECT * FROM pcdsEmployee WHERE EmpName LIKE '%S'
```

Below SQL statement will return all words having letter “S” in between

```
SELECT * FROM pcdsEmployee WHERE EmpName LIKE '%S%'
```

“\_” operator (we can read as “Underscore Operator”). “\_” operator is the character defined at that point. In the below sample fired a query

```
Select name from pcdsEmployee where name like '_s%'
```

So all name where second letter is “s” is returned.

### **What is the SQL ” IN ” clause?**

SQL IN operator is used to see if the value exists in a group of values. For instance the below SQL checks if the Name is either ‘rohit’ or ‘Anuradha’

```
SELECT * FROM pcdsEmployee WHERE name IN ('Rohit','Anuradha')
```

Also you can specify a not clause with the same.

```
SELECT * FROM pcdsEmployee WHERE age NOT IN (17,16)
```

### **How to select the first record in a given set of rows?**

```
Select top 1 * from sales.salesperson
```

### **What is a self-join?**

If we want to join two instances of the same table we can use self-join.

### **What are the technical specifications of MySQL?**

MySQL has the following technical specifications –

- Flexible structure
- High performance
- Manageable and easy to use
- Replication and high availability
- Security and storage management
- Drivers
- Graphical Tools
- MySQL Enterprise Monitor
- MySQL Enterprise Security

- JSON Support
- Replication & High-Availability
- Manageability and Ease of Use
- OLTP and Transactions
- Geo-Spatial Support

## **Why do we use the MySQL database server?**

First of all, the MySQL server is free to use for developers and small enterprises.

MySQL server is open source.

MySQL's community is tremendous and supportive; hence any help regarding MySQL is resolved as soon as possible.

MySQL has very stable versions available, as MySQL has been in the market for a long time. All bugs arising in the previous builds have been continuously removed, and a very stable version is provided after every update.

The MySQL database server is very fast, reliable, and easy to use. You can easily use and modify the software. MySQL software can be downloaded free of cost from the internet.

## **What are the different tables present in MySQL?**

There are many tables that remain present by default. But, MyISAM is the default database engine used in MySQL. There are five types of tables that are present:

- MyISAM
- Heap
- Merge
- INNO DB
- ISAM

## **How to install MySQL?**

Installing MySQL on our system allows us to safely create, drop, and test web applications without affecting our live website's data. There are many ways to use MySQL on our system, but the best way is to install it manually. The manual installation allows us to learn more about the system and provides more control over the database. To see the installation steps of MySQL in Windows goes to the below link:

<https://www.javatpoint.com/how-to-install-mysql>

Manual installation of MySQL has several benefits:

- Backing up, reinstalling, or moving databases from one location to another can be achieved in a second.
- It provides more control to how and when MySQL server starts and closes.
- We can install MySQL anywhere, like in a portable USB drive.

## **How to add foreign keys in MySQL?**

The foreign key is used to link one or more tables together. It matches the primary key field of another table to link the two tables. It allows us to create a parent-child relationship with the tables. We can add a foreign key to a table in two ways:

- Using the CREATE TABLE Statement
- Using the ALTER TABLE Statement

Following is the syntax to define a foreign key using CREATE TABLE OR ALTER TABLE statement:

1. [CONSTRAINT constraint\_name]
2. FOREIGN KEY [foreign\_key\_name] (col\_name, ...)
3. REFERENCES parent\_tbl\_name (col\_name,...)

## **How to connect to the MySQL database?**

MySQL allows us to connect with the database server in mainly two ways:

### **Using Command-line Tool**

We can find the command-line client tool in the bin directory of the [MySQL's installation](#)

folder. To invoke this program, we need to navigate the installation folder's bin directory and type the below command:

```
mysql
```

Next, we need to run the below command to connect to the MySQL Server:

```
shell>mysql -u root -p
```

Finally, type the password for the selected user account root and press Enter:

```
Enter password: *****
```

After successful connection, we can use the below command to use the:

```
USE database_name;
```

## **Using MySQL Workbench**

We can make a connection with database using [MySQL Workbench](#)

, simply clicking the plus (+) icon or navigating to the menu bar -> Database -> Connect to Database, the following screen appears. Now, you need to fill all the details to make a connection:

Once we finished this setup, it will open the MySQL Workbench screen. Now, we can double click on the newly created connection to connect with the database server.

To read more information, [click here](#)

## **How to change the MySQL password?**

We can change the MySQL root password using the below statement in the new notepad file and save it with an appropriate name:

```
ALTER USER 'root'@'localhost' IDENTIFIED BY 'NewPassword';
```

Next, open a Command Prompt and navigate to the MySQL directory. Now, copy the following folder and paste it in our DOS command and press the Enter key.

```
C:\Users\javatpoint> CD C:\Program Files\MySQL\MySQL Server 8.0\bin
```

Next, enter this statement to change the password:

```
mysqld --init-file=C:\\mysql-notepadfile.txt
```

Finally, we can log into the MySQL server as root using this new password. After launching the MySQL server, it is to delete the C:\\mysql-init.txt file to ensure the password change.

### **How to create a database in MySQL Workbench?**

To create a new database in MySQL Workbench, we first need to launch the MySQL Workbench and log in using the username and password. Go to the Navigation tab and click on the Schema menu. Right-click under the Schema menu and select Create

Schema or click the database icon (red rectangle), as shown in the following screen.

A new popup screen appears where we need to fill all the details. After entering the details, click on the Apply button and then the Finish button to complete the database creation.

### **How to create a table in MySQL Workbench?**

Launch the MySQL Workbench and go to the Navigation tab and click on the Schema menu where all the previously created databases are shown. Select any database and double click on it. It will show the sub-menus where we need to select the Tables option.

Select Tables sub-menu, right-click on it and select Create Table option. We can also click on create a new table icon (shown in red rectangle) to create a table. It will open the new popup screen where we need to fill all the details to create a table. Here, we will enter the table name and column details. After entering the details, click on the Apply button and then the Finish button to complete the table creation.

### **How to change the table name in MySQL?**

Sometimes our table name is non-meaningful. In that case, we need to change or rename the table name. MySQL provides the following syntax to rename one or more tables in the current database:

```
mysql> RENAME old_table TO new_table;
```

If we want to change more than one table name, use the below syntax:

```
RENAME TABLE old_tab1 TO new_tab1,
```

```
old_tab2 TO new_tab2, old_tab3 TO new_tab3;
```

### **How to change the database name in MySQL?**

Sometimes we need to change or rename the database name because of its non-meaningful name. To rename the database name, we need first to create a new database into the MySQL server. Next, MySQL provides the mysqldump shell command to create a dumped copy of the selected database and then import all the data into the newly created database. The following is the syntax of using mysqldump command:

```
mysqldump -u username -p "password" -R oldDbName > oldDbName.sql
```

Now, use the below command to import the data into the newly created database:

```
mysql -u username -p"password" newDbName < oldDbName.sql
```

### **How to import a database in MySQL?**

Importing database in MySQL is a process of moving data from one place to another place. It is a very useful method for backing up essential data or transferring our data between different locations. For example, we have a contact book database, which is essential to keep it in a secure place. So we need to export it in a safe place, and whenever it lost from the original location, we can restore it using import options.

In MySQL, we can import a database in mainly two ways:

- Command Line Tool
- MySQL Workbench

### **How to change the column name in MySQL?**

While creating a table, we have kept one of the column names incorrectly. To change or rename an existing column name in MySQL, we need to use the ALTER TABLE and CHANGE commands together. The following are the syntax used to rename a column in MySQL:

```
ALTER TABLE table_name
```

```
CHANGE COLUMN old_column_name new_column_name column_definition [FIRST|AFTER  
existing_column];
```

Suppose the column's current name is S\_ID, but we want to change this with a more appropriate title as Stud\_ID. We will use the below statement to change its name:

```
ALTER TABLE Student CHANGE COLUMN S_ID Stud_ID varchar(10);
```

### **How to delete columns in MySQL?**

We can remove, drop, or delete one or more columns in an existing table using the ALTER TABLE statement as follows:

```
ALTER TABLE table_name DROP COLUMN column_name1, column_name2....;
```

### **How to insert data in MySQL?**

We can insert data in a MySQL table using the INSERT STATEMENT. This statement allows us to insert single or multiple rows into a table. The following is the basic syntax to insert a record into a table:

```
INSERT INTO table_name ( field1, field2,...fieldN )
```

```
VALUES ( value1, value2,...valueN );
```

If we want to insert more than one rows into a table, use the below syntax:

```
INSERT INTO table(field1, field2,...fieldN)
```

1. VALUES
2. (value1, value 2, ...),
3. (value1, value2, ...),
4. ...
5. (value1, value2, ...);

## **How to delete a row in MySQL?**

We can delete a row from the MySQL table using the DELETE STATEMENT within the database. The following is the generic syntax of DELETE statement in MySQL to remove one or more rows from a table:

```
DELETE FROM table_name WHERE Condition_specified;
```

It is noted that if we have not specified the WHERE clause with the syntax, this statement will remove all the records from the given table.

## **How to join two tables in MySQL?**

We can connect two or more tables in MySQL using the JOIN clause. MySQL allows various types of JOIN clauses. These clauses connect multiple tables and return only those records that match the same value and property in all tables. The following are the four easy ways to join two or more tables in MySQL:

- Inner Join
- Left Join
- Right Join
- Cross Join

## **How to join three tables in MySQL?**

Sometimes we need to fetch data from three or more tables. There are two types available to do these types of joins. Suppose we have three tables named Student, Marks, and Details.

Let's say Student has (stud\_id, name) columns, Marks has (school\_id, stud\_id, scores) columns, and Details has (school\_id, address, email) columns.

### **1. Using SQL Join Clause**

This approach is similar to the way we join two tables. The following query returns result from three tables:

1. SELECT name, scores, address, email FROM Student s
2. INNER JOIN Marks m on s.stud\_id = m.stud\_id
3. INNER JOIN Details d on d.school\_id = m.school\_id;

## **2. Using Parent-Child Relationship**

It is another approach to join more than two tables. In the above tables, we have to create a parent-child relationship. First, create column X as a primary key in one table and as a foreign key in another table. Therefore, stud\_id is the primary key in the Student table and will be a foreign key in the Marks table. Next, school\_id is the primary key in the Marks table and will be a foreign key in the Details table. The following query returns result from three tables:

1. SELECT name, scores, address, email
2. FROM Student s, Marks m, Details d
3. WHERE s.stud\_id = m.stud\_id AND m.school\_id = d.school\_id;

### **How to update the table in MySQL?**

We can update existing records in a table using the UPDATE statement that comes with the SET and WHERE clauses. The SET clause changes the values of the specified column. The WHERE clause is optional, which is used to specify the condition. This statement can also use to change values in one or more columns of a single row or multiple rows at a time. Following is a generic syntax of UPDATE command to modify data into the MySQL table:

1. UPDATE table\_name
2. SET field1=new-value1, field2=new-value2, ...
3. [WHERE Clause]

### **What is MySQL Workbench?**

MySQL Workbench is a unified visual database designing or GUI tool used for working on MySQL databases. It is developed and maintained by Oracle that provides SQL development, data migration, and comprehensive administration tools for server configuration, user administration, backup, etc. We can use this Server Administration to create new physical data models, E-R diagrams, and SQL development. It is available for all major operating systems. MySQL provides supports for it from MySQL Server version v5.6 and higher.

It is mainly available in three editions, which are given below:

- Community Edition (Open Source, GPL)
- Standard Edition (Commercial)
- Enterprise Edition (Commercial)

## **How to drop the primary key in MySQL?**

MySQL primary key is a single or combination of the field used to identify each record in a table uniquely. A primary key column cannot be null or empty. We can remove or delete a primary key from the table using the ALTER TABLE statement. The following syntax is used to drop the primary key:

```
ALTER TABLE table_name DROP PRIMARY KEY;
```

## **How to create a Stored Procedure in MySQL?**

A stored procedure is a group of SQL statements that we save in the database. The SQL queries, including INSERT, UPDATE, DELETE, etc. can be a part of the stored procedure. A procedure allows us to use the same code over and over again by executing a single statement. It stores in the database data dictionary.

We can create a stored procedure using the below syntax:

1. CREATE PROCEDURE procedure\_name [ (parameter datatype [, parameter datatype] ) ]
2. BEGIN
3. Body\_section of SQL statements
4. END;

This statement can return one or more value through parameters or may not return any result. The following example explains it more clearly:

1. DELIMITER \$\$
2. CREATE PROCEDURE get\_student\_info()
3. BEGIN
4. SELECT \* FROM Student\_table;
5. END\$\$

## **How to execute a stored procedure in MySQL?**

We can execute a stored procedure in MySQL by simply CALL query. This query takes the name of the stored procedure and any parameters we need to pass to it. The following is the basic syntax to execute a stored procedure:

1. CALL stored\_procedure\_name (argument\_list);

Let's understand it with this example:

1. CALL Product\_Pricing (@pricelow, @pricehigh);

Here, a stored procedure named Product\_Pricing calculates and returns the lowest and highest product prices.

### **How to create a View in MySQL?**

A view is a database object whose values are based on the base table. It is a **virtual table** created by a query by joining one or more tables. It is operated similarly to the base table but does not contain any data of its own. If any changes occur in the underlying table, the same changes reflected in the View also.

Following is the general syntax of creating a VIEW in MySQL:

1. CREATE [OR REPLACE] VIEW view\_name AS
2. SELECT columns
3. FROM tables
4. [WHERE conditions];

### **How to create a Trigger in MySQL?**

A trigger is a procedural code in a database that automatically invokes whenever certain events on a particular table or view in the database occur. It can be executed when records are inserted into a table, or any columns are being updated. We can create a trigger in MySQL using the syntax as follows:

1. CREATE TRIGGER trigger\_name
2. [before | after]
3. {insert | update | delete}
4. ON table\_name [FOR EACH ROW]
5. BEGIN
6.     –variable declarations
7.     –trigger code
8. END;

## **How to clear screen in MySQL?**

If we use MySQL in Windows, it is not possible to clear the screen before version 8. At that time, the Windows operating system provides the only way to clear the screen by exiting the MySQL command-line tool and then again open MySQL.

After the release of MySQL version 8, we can use the below command to clear the command line screen:

```
mysql> SYSTEM CLS;
```

## **How to create a new user in MySQL?**

A USER in MySQL is a record in the USER-TABLE. It contains the login information, account privileges, and the host information for MySQL account to access and manage the databases. We can create a new user account in the database server using the MySQL Create User statement. It provides authentication, SSL/TLS, resource-limit, role, and password management properties for the new accounts.

The following is the basic syntax to create a new user in MySQL:

```
CREATE USER [IF NOT EXISTS] account_name IDENTIFIED BY 'password';
```

## **How to check USERS in MySQL?**

If we want to manage a database in MySQL, it is required to see the list of all user's accounts in a database server. The following command is used to check the list of all users available in the database server:

```
mysql> SELECT USER FROM mysql.user;
```

## **How to import a CSV file in MySQL?**

MySQL allows us to import the CSV (comma separated values) file into a database or table. A CSV is a plain text file that contains the list of data and can be saved in a tabular format. MySQL provides the LOAD DATA INFILE statement to import a CSV file. This statement is used to read a text file and import it into a database table very quickly. The full syntax to import a CSV file is given below:

1. LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/filename.csv'
2. INTO TABLE tablename
3. FIELDS TERMINATED BY ','
4. OPTIONALLY ENCLOSED BY ""
5. LINES TERMINATED BY '\r\n'
6. IGNORE 1 ROWS;

## How to insert Date in MySQL?

MySQL allows us to use the INSERT STATEMENT to add the date in MySQL table. MySQL provides several data types for storing dates such as DATE, TIMESTAMP, DATETIME, and YEAR. The default format of the date in MySQL is YYYY-MM-DD. Following is the basic syntax to insert date in MySQL table:

```
INSERT INTO table_name (column_name, column_date) VALUES ('DATE: Manual Date', '2008-7-04');
```

If we want to insert a date in the mm/dd/yyyy format, it is required to use the below statement:

1. `INSERT INTO table_name VALUES (STR_TO_DATE(date_value, format_specifier));`

## How to check database size in MySQL?

MySQL allows us to query the information\_schema.tables table to get the information about the tables and databases. It will return the information about the data length, index length, collation, creation time, etc. We can check the size of the database on the server using the below syntax:

1. `SELECT table_schema AS 'Database Name',`
2. `SUM(data_length + index_length) 'Size in Bytes',`
3. `ROUND(SUM(data_length + index_length) / 1024 / 1024, 2) 'Size in MB'`
4. `FROM information_schema.tables`
5. `WHERE table_schema = 'testdb'`
6. `GROUP BY table_schema;`

If we want to check the size of the table in a specific database, use the following statement:

1. SELECT table\_name AS 'Table Name',
2. ROUND(((data\_length + index\_length) / 1024 / 1024), 2) AS 'Size in MB'
3. FROM information\_schema.TABLES
4. WHERE table\_schema = 'testdb'
5. ORDER BY (data\_length + index\_length) DESC;

### **How does indexing works in MySQL?**

Indexing is a process to find an unordered list into an ordered list. It helps in maximizing the query's efficiency while searching on tables in MySQL. The working of MySQL indexing is similar to the book index.

Suppose we have a book and want to get information about, say, searching. Without indexing, it is required to go through all pages one by one, until the specific topic was not found. On the other hand, an index contains a list of keywords to find the topic mentioned on pages. Then, we can flip to those pages directly without going through all pages.

### **Who owns MySQL?**

MySQL is the most popular free and open-source database software which comes under the GNU General Public License. In the beginning, it was owned and sponsored by the Swedish company MySQL AB. Now, it is bought by Sun Microsystems (now Oracle Corporation), who is responsible for managing and developing the database.

### **How to view the database in MySQL?**

Working with the MySQL server, it is a common task to view or list the available databases. We can view all the databases on the MySQL server host using the following command:

```
mysql> SHOW DATABASES;
```

### **How to set auto increment in MySQL?**

Auto Increment is a constraint that automatically generates a unique number while inserting a new record into the table. Generally, it is used for the primary key field in a table. In MySQL, we can set the value for an AUTO\_INCREMENT column using the ALTER TABLE statement as follows:

```
ALTER TABLE table_name AUTO_INCREMENT = value;
```

## How to find the second highest salary in MySQL?

MySQL uses the LIMIT keyword, which can be used to limit the result set. It will allow us to get the first few rows, last few rows, or range of rows. It can also be used to find the second, third, or nth highest salary. It ensures that you have use order by clause to sort the result set first and then print the output that provides accurate results. The following query is used to get the second highest salary in MySQL:

1. SELECT salary
2. FROM (SELECT salary FROM employees ORDER BY salary DESC LIMIT 2) AS Emp  
ORDER BY salary LIMIT 1;

There are some other ways to find the second highest salary in MySQL, which are given below:

This statement uses subquery and IN clause to get the second highest salary:

1. SELECT MAX(salary)
2. FROM employees
3. WHERE salary NOT IN ( SELECT Max(salary) FROM employees);

This query uses subquery and < operator to return the second highest salary:

1. SELECT MAX(salary) From employees
2. WHERE salary < ( SELECT Max(salary) FROM employees);

## What is the difference between TRUNCATE and DELETE in MySQL?

- TRUNCATE is a DDL command, and DELETE is a DML command.
- It is not possible to use Where command with TRUNCATE QLbut you can use it with DELETE command.
- TRUNCATE cannot be used with indexed views, whereas DELETE can be used with indexed views.
- The DELETE command is used to delete data from a table. It only deletes the rows of data from the table while truncate is a very dangerous command and should be used carefully because it deletes every row permanently from a table.

## **How many Triggers are possible in MySQL?**

There are only six Triggers allowed to use in the MySQL database.

1. Before Insert
2. After Insert
3. Before Update
4. After Update
5. Before Delete
6. After Delete

## **What is the heap table?**

Tables that are present in memory is known as HEAP tables. When you create a heap table in MySQL, you should need to specify the TYPE as HEAP. These tables are commonly known as memory tables. They are used for high-speed storage on a temporary basis. They do not allow BLOB or TEXT fields.

## **What is BLOB and TEXT in MySQL?**

BLOB is an acronym that stands for a large binary object. It is used to hold a variable amount of data.

There are four types of the BLOB.

1. TINYBLOB
2. BLOB
3. MEDIUMBLOB
4. LONGBLOB

The differences among all these are the maximum length of values they can hold.

TEXT is a case-insensitive BLOB. TEXT values are non-binary strings (character string). They have a character set, and values are stored and compared based on the collation of the character set.

There are four types of TEXT.

1. TINYTEXT
2. TEXT
3. MEDIUMTEXT
4. LONGTEXT

## **What is the difference between the heap table and the temporary table?**

### **Heap tables:**

Heap tables are found in memory that is used for high-speed storage temporarily. They do not allow BLOB or TEXT fields.

Heap tables do not support AUTO\_INCREMENT.

Indexes should be NOT NULL.

### **Temporary tables:**

The temporary tables are used to keep the transient data. Sometimes it is beneficial in cases to hold temporary data. The temporary table is deleted after the current client session terminates.

### **Main differences:**

The heap tables are shared among clients, while temporary tables are not shared.

Heap tables are just another storage engine, while for temporary tables, you need a special privilege (create temporary table).

## **What is the difference between FLOAT and DOUBLE?**

FLOAT stores floating-point numbers with accuracy up to 8 places and allocate 4 bytes. On the other hand, DOUBLE stores floating-point numbers with accuracy up to 18 places and allocates 8 bytes.

## **What are the advantages of MySQL in comparison to Oracle?**

1. MySQL is a free, fast, reliable, open-source relational database while Oracle is expensive, although they have provided Oracle free edition to attract MySQL users.
2. MySQL uses only just under 1 MB of RAM on your laptop, while Oracle 9i installation uses 128 MB.
3. MySQL is great for database enabled websites while Oracle is made for enterprises.
4. MySQL is portable.

## **What are the disadvantages of MySQL?**

1. MySQL is not so efficient for large scale databases.
2. It does not support COMMIT and STORED PROCEDURES functions version less than 5.0.
3. Transactions are not handled very efficiently.
4. The functionality of MySQL is highly dependent on other addons.
5. Development is not community-driven.

## **What is the difference between CHAR and VARCHAR?**

1. CHAR and VARCHAR have differed in storage and retrieval.
2. CHAR column length is fixed, while VARCHAR length is variable.
3. The maximum no. of character CHAR data types can hold is 255 characters, while VARCHAR can hold up to 4000 characters.
4. CHAR is 50% faster than VARCHAR.
5. CHAR uses static memory allocation, while VARCHAR uses dynamic memory allocation.

## **What is the difference between MySQL\_connect and MySQL\_pconnect?**

### **Mysql\_connect:**

1. It opens a new connection to the database.
2. Every time you need to open and close the database connection, depending on the request.
3. Opens page whenever it is loaded.

### **Mysql\_pconnect:**

1. In Mysql\_pconnect, "p" stands for persistent connection, so it opens the persistent connection.
2. The database connection cannot be closed.
3. It is more useful if your site has more traffic because there is no need to open and close connection frequently and whenever the page is loaded.

## **What does "i\_am\_a\_dummy flag" do in MySQL?**

The "i\_am\_a\_dummy flag" enables the MySQL engine to refuse any UPDATE or DELETE statement to execute if the WHERE clause is not present. Hence it can save the programmer from deleting the entire table my mistake if he does not use WHERE clause.

## **How to get the current date in MySQL?**

To get current date, use the following syntax:

```
SELECT CURRENT_DATE();
```

## **What are the security alerts while using MySQL?**

Install antivirus and configure the operating system's firewall.

Never use the MySQL Server as the UNIX root user.

Change the root username and password Restrict or disable remote access.

## **How to change a password for an existing user via mysqladmin?**

Mysqladmin -u root -p password "newpassword".

## **What is the difference between UNIX timestamps and MySQL timestamps?**

Actually, both Unix timestamp and MySQL timestamp are stored as 32-bit integers, but MySQL timestamp is represented in the readable format of YYYY-MM-DD HH:MM:SS format.

## **How to display the nth highest salary from a table in a MySQL query?**

Let us take a table named the employee.

### **To find Nth highest salary is:**

```
select distinct(salary)from employee order by salary desc limit n-1,1
```

### **if you want to find 3rd largest salary:**

```
select distinct(salary)from employee order by salary desc limit 2,1
```

## **What is the MySQL default port number?**

MySQL default port number is 3306.

## **What is REGEXP?**

REGEXP is a pattern match using a regular expression. The regular expression is a powerful way of specifying a pattern for a sophisticated search.

Basically, it is a special text string for describing a search pattern. To understand it better, you can think of a situation of daily life when you search for .txt files to list all text files in the file manager. The regex equivalent for .txt will be `.*\.txt`.

## **How many columns can you create for an index?**

You can create maximum of 16 indexed columns for a standard table.

## **What is the difference between NOW() and CURRENT\_DATE()?**

NOW() command is used to show current year, month, date with hours, minutes, and seconds while CURRENT\_DATE() shows the current year with month and date only.

## **Write a query to display the current date and time?**

If you want to display the current date and time, use –

```
SELECT NOW();
```

If you want to display the current date only, use:

```
SELECT CURRENT_DATE();
```

## **What is the save point in MySQL?**

A defined point in any transaction is known as savepoint.

SAVEPOINT is a statement in MySQL, which is used to set a named transaction savepoint with the name of the identifier.

## **What is SQLyog?**

SQLyog program is the most popular GUI tool for admin. It is the most popular MySQL manager and admin tool. It combines the features of MySQL administrator, phpMyadmin, and others. MySQL front ends and MySQL GUI tools.

## **How do you backup a database in MySQL?**

It is easy to back up data with phpMyAdmin. Select the database you want to backup by clicking the database name in the left-hand navigation bar. Then click the export button and make sure that all tables are highlighted that you want to back up. Then specify the option you want under export and save the output.

## **Write a query to count the number of rows of a table in MySQL.**

```
SELECT COUNT user_id FROM users;
```

## **Write a query to select all teams that won either 1, 3, 5, or 7 games.**

```
SELECT team_name FROM team WHERE team_won IN (1, 3, 5, 7);
```

## **How is the MyISAM table stored?**

MyISAM table is stored on disk in three formats.

- '.frm' file : storing the table definition
- '.MYD' (MYData): data file
- '.MYI' (MYIndex): index file

## **What is the usage of ENUMs in MySQL?**

ENUMs are string objects. By defining ENUMs, we allow the end-user to give correct input as in case the user provides an input that is not part of the ENUM defined data, then the query won't execute, and an error message will be displayed which says "The wrong Query". For instance, suppose we want to take the gender of the user as an input, so we specify ENUM('male', 'female', 'other'), and hence whenever the user tries to input any string any other than these three it results in an error.

ENUMs are used to limit the possible values that go in the table:

### **For example:**

```
CREATE TABLE months (month ENUM 'January', 'February', 'March'); INSERT months  
VALUES ('April');
```

## **What are the advantages of MyISAM over InnoDB?**

MyISAM follows a conservative approach to disk space management and stores each MyISAM table in a separate file, which can be further compressed if required. On the other hand, InnoDB stores the tables in the tablespace. Its further optimization is difficult.

## **What are the differences between MySQL\_fetch\_array(), MySQL\_fetch\_object(), MySQL\_fetch\_row()?**

Mysql\_fetch\_object is used to retrieve the result from the database as objects, while mysql\_fetch\_array returns result as an array. This will allow access to the data by the field names.

Using mysql\_fetch\_object field can be accessed as \$result->name.

Using mysql\_fetch\_array field can be accessed as \$result->[name].

Using mysql\_fetch\_row(\$result) where \$result is the result resource returned from a successful query executed using the mysql\_query() function.

1. \$result = mysql\_query("SELECT \* from students");
2. while(\$row = mysql\_fetch\_row(\$result))
3. {
4.     Some statement;
5. }

## **What is the difference between mysql\_connect and mysql\_pconnect?**

Mysql\_connect() is used to open a new connection to the database, while mysql\_pconnect() is used to open a persistent connection to the database. It specifies that each time the page is loaded, mysql\_pconnect() does not open the database.

## **What is the use of mysql\_close()?**

Mysql\_close() cannot be used to close the persistent connection. However, it can be used to close a connection opened by mysql\_connect().

## **What is MySQL data directory?**

MySQL data directory is a place where MySQL stores its data. Each subdirectory under this data dictionary represents a MySQL database. By default, the information managed by MySQL = server mysqld is stored in the data directory.

## **How do you determine the location of MySQL data directory?**

The default location of MySQL data directory in windows is C:\mysql\data or C:\Program Files\MySQL\MySQL Server 5.0 \data.

## **What is the usage of regular expressions in MySQL?**

In MySQL, regular expressions are used in queries for searching a pattern in a string.

- \* Matches 0 more instances of the string preceding it.
- + matches one more instances of the string preceding it.
- ? Matches 0 or 1 instances of the string preceding it.
- . Matches a single character.
- [abc] matches a or b or z
- | separates strings
- ^ anchors the match from the start.
- ." Can be used to match any single character. "|" can be used to match either of the two strings
- REGEXP can be used to match the input characters with the database.

### **Example:**

The following statement retrieves all rows where column employee\_name contains the text 1000 (example salary):

1. Select employee\_name
2. From employee
3. Where employee\_name REGEXP '1000'
4. Order by employee\_name

## **Explain Access Control Lists.**

An ACL is a list of permissions that are associated with an object. MySQL keeps the Access Control Lists cached in memory, and whenever the user tries to authenticate or

execute a command, MySQL checks the permission required for the object, and if the permissions are available, then execution completes successfully.

## **What is InnoDB ISAM?**

InnoDB is a storage database for SQL. The ACID-transactions are also provided in InnoDB and also includes support for the foreign key. Initially owned by InnobaseOY now belongs to Oracle Corporation after it acquired the latter since 2005.

It is a system for file management developed by IBM, which allows records to access sequentially or even randomly.

## **How can we run batch mode in MySQL?**

To perform batch mode in MySQL, we use the following command:

1. mysql;
2. mysql mysql.out;

## **What are federated tables?**

Federated tables are tables that point to the tables located on other databases on some other server.

## **What is the difference between primary key and candidate key?**

To identify each row of a table, we will use a primary key. For a table, there exists only one primary key.

A candidate key is a column or a set of columns, which can be used to uniquely identify any record in the database without having to reference any other data.

## **What are the drivers in MySQL?**

Following are the drivers available in MySQL:

- PHP Driver
- JDBC Driver
- ODBC Driver
- C WRAPPER
- PYTHON Driver
- PERL Driver

- RUBY Driver
- CAP11PHP Driver
- Ado.net5.mxz

## MySQL Advance Query

### **MYSQL COMMAND LINE COMMANDS**

COMMAND	MEANING	SYNTAX
mysql	Allows user to connect to the MySQL CLI	>MYSQL -U [USERNAME] -P;
exit	Exits the MySQL CLI	>EXIT;
clear	Clears the MySQL shell	>SYSTEM CLEAR;
create user	Creates a new user	>CREATE USER `NEWUSER`@`LOCALHOST` IDENTIFIED BY `NEW_PASSWORD`
show user	Shows all user who have access to the MySQL Client	>SELECT USER, HOST FROM MYSQL.USER;
drop user	To delete an existing user	> DROP USER 'USERNAME'@'LOCALHOST';
grant all privileges	Assigns privileges to a MySQL user	>GRANT ALL PRIVILEGES ON * . * TO 'USERNAME'@'LOCALHOST';
show grants	Displays the privileges that are assigned to a MySQL user	> SHOW GRANTS FOR 'USERNAME'@'LOCALHOST';
revoke all privileges	Revokes all privileges assigned to a MySQL user	>REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'USERNAME'@'LOCALHOST';
mysqldump	Creates a backup of a set of SQL statements that can be used to recreate the original database object definitions and table data.	>mysqldump -U USERNAME -P DATABASENAME> DATABASENAME_BACKUP.SQL

## **MYSQL DATABASE COMMANDS (DATA DEFINITION LANGUAGE;DDL)**

COMMAND	MEANING	SYNTAX
show database	Shows all the databases available in MySQL server.	>SHOW DATABASE;
create database	Creates a new database if it does not exist.	>CREATE DATABASE DATABASENAME;
drop database	To delete an existing database permanently.	>DROP DATABASE DATABASE_NAME
alter database	Changes or modifies the characteristics of an existing database.	>ALTER DATABASE [DATABASENAME] ALTEROPTION ;
use database	Allow you to use a particular database or change from the current database to another database.	>USE DATABASENAME;

## **MySQL Table commands(DDL)**

COMMAND	MEANING	SYNTAX
show tables	Shows all tables within the current database.	>SHOW TABLES;
create table	Creates a new table in the current database.	>CREATE TABLE TABLENAME ( COLUMN1 DATATYPE, COLUMN2 DATATYPE, COLUMN3 DATATYPE, .... CONSTRAINTS .... );
alter table (add column)	Adds a new column to an existing table.	>ALTER TABLE TABLENAME ADD COLUMNNAME DATATYPE;

COMMAND	MEANING	SYNTAX
alter table (drop column)	Deletes a column from an existing table.	>ALTER TABLE TABLENAME DROP COLUMN COLUMNNAME;
alter table (alter column)	Alters an existing column in an already existing table.	>ALTER TABLE TABLENAME ALTER COLUMN COLUMNNAME DATATYPE;
alter table (add primary key)	Alters or adds primary key to an existing table.	>ALTER TABLE TABLENAME ADD PRIMARY KEY (COLUMNNAME, ...);
alter table (drop primary key)	Drops an existing primary key in a table.	>ALTER TABLE TABLENAME DROP PRIMARY KEY;
alter table (add foreign key)	Creates a foreign key on an existing table.	>ALTER TABLE TABLENAME1 ADD FOREIGN KEY (COLUMN1) REFERENCES TABLENAME2(COLUMN2);
alter table (drop foreign key)	Deletes an existing foreign key in an already existing table.	> ALTER TABLE TABLENAME DROP FOREIGN KEY FOREIGNKEY_NAME;
rename table	Changes the name of an existing table.	>RENAME TABLE OLD_TABLENAME TO NEW_TABLENAME;
drop table	Deletes the entire table along with its definition.	>DROP TABLE TABLE_NAME;
truncate table	Remove all records in a MySQL table.	>TRUNCATE TABLE TABLENAME;
describe table	Displays all the columns of an existing table.	>DESCRIBE TABLE_NAME;
describe table column	Displays all the values stored in a particular column.	>DESCRIBE TABLE_NAME COLUMN_NAME;

## **MySQL DML(Data Manipulation Language) Commands**

COMMAND	MEANING	SYNTAX
select *	Displays all rows in a table.	>SELECT * FROM TABLENAME
select * (multiple tables)	Displays all the rows of the cartesian product of the two tables	>SELECT * FROM TABLENAME1, TABLENAME2;
select columns	Select particular columns from table(s)	>SELECT COLUMN1, COLUMN2 FROM TABLENAME;
select with condition	Displays rows based on a particular condition	> SELECT * FROM TABLENAME WHERE CONDITION
select with multiple conditions(AND)	Displays rows only when both the conditions are satisfied.	> SELECT * FROM TABLENAME WHERE CONDITION1 AND CONDITION2.
select with multiple conditions(OR)	Displays rows only when either of the conditions are satisfied.	> SELECT * FROM TABLENAME WHERE CONDITION1 OR CONDITION2.
select with	Displays rows based on	>SELECT * FROM TABLENAME WHERE NOT CONDITION.

COMMAND	MEANING	SYNTAX
condition(NOT)	negation of a particular condition.	
select with group by	Displays rows that have same values into summary rows	> SELECT .. FROM .. WHERE... GROUP BY COLUMN3;
select with having	Used instead of where for aggregate functions.	>SELECT COUNT(COLUMN1) FROM TABLENAME ORDER BY COLUMN2 HAVING COUNT(COLUMN1)>3;
select distinct	Display all unique rows discarding duplicate ones.	>SELECT DISTINCT (COLUMN1) FROM TABLENAME;
order by	Used to sort results in ascending order or descending order	> SELECT ... FROM TABLENAME ORDER BY COLUMN1 ASC DESC;
column alias	Changes the output of the name of the column.	> SELECT COLUMN1 AS NEWNAME FROM TABLENAME;
like	Used to search for a specific pattern.	> SELECT COLUMN1 FROM TABLENAME WHERE COLUMN1 LIKE '%PATTERN%';
insert record	Adds a new row to an	> INSERT INTO TABLENAME (COLUMN1,COLUMN2...) VALUES (VALUE1,VALUE2...);

COMMAND	MEANING	SYNTAX
	existing table.	
insert record(multiple)	Adds multiple records into an existing table.	> INSERT INTO TABLENAME (COLUMN1,COLUMN2...) VALUES (VALUE1A,VALUE2A...),(VALUE1B,VALUE2B,...);
delete	Deletes all records in a table.	> DELETE FROM TABLENAME;
delete with where	Deletes specific records	>DELETE FROM TABLENAME WHERE CONDITION;
between	Selects values in a given range	>SELECT * FROM TABLENAME WHERE AGE BETWEEN 25 AND 30.
in	Used instead of multiple OR operators.	> SELECT * FROM TABLENAME WHERE COLUMN2 IN (V1,V2...);
exists	Tests for existence of a certain record. Returns a boolean value.	> SELECT * FROM TABLE NAME WHERE EXIST (SUB QUERY);
update table	Modifies data in existing tables.	> UPDATE TABLENAME SET COLUMNNAME=VALUE WHERE CONDITION;
inner join	Selects records that have the same values in two same or distinct tables.	> SELECT COLUMN(S) FROM TABLENAME1 INNER JOIN TABLENAME2 ON TABLENAME1.COLUMNNAME=TABLENAME2.COLUMNNAME;
left join	Selects all the records from	>SELECT COLUMN(S) FROM TABLENAME1 LEFT JOIN TABLENAME2 ON

COMMAND	MEANING	SYNTAX
	the left table and matching records from the right table.	TABLENAME1.COLUMNNAME=TABLENAME2.COLUMNNAME;
right join	Selects all the records from the right table and matching records from the left table.	>SELECT COLUMN(S) FROM TABLENAME1 RIGHT JOIN TABLENAME2 ON TABLENAME1.COLUMNNAME=TABLENAME2.COLUMNNAME;
cross join	Selects rows from cartesian product of both the tables.	>SELECT COLUMN(S) FROM TABLE1 CROSS JOIN TABLE2;
full outer join	Selects all records with a match on table1 or table2.	>SELECT COLUMN(S) FROM TABLENAME1 FULL OUTER JOIN TABLENAME2 ON TABLENAME1.COLUMNNAME=TABLENAME2.COLUMNNAME WHERE CONDITION;
union	Combines the result of two select statements.	>SELECT * FROM TABLENAME1 UNION SELECT * FROM TABLENAME2
union all	Similar to Union but allows duplicate values	>SELECT * FROM TABLENAME1 UNION ALL SELECT * FROM TABLENAME2
concat()	Combines two or more columns	>SELECT CONCAT(COLUMN1, " ", POSTALCODE, " ", COLUMN2) AS NEWCOL FROM TABLENAME;

COMMAND	MEANING	SYNTAX
	together.	

## **MySQL DATA TYPES**

In MySQL just like other programming languages, each column, local variable, expression, and parameter has a related data type. A data type is an attribute that specifies the type of data that the object can hold.

### **String Data Types**

DATATYPE	DETAILS
CHAR(size)	Stores Alpha Numeric and special characters. Size varies from 0 to 255 characters.
VARCHAR(size)	Can contain letters, numbers, and characters that are of variable length (size). The size parameter specifies the column length in characters; it can be from 0 to 65535.
BINARY(size)	Similar to CHAR(). But it stores binary strings.
VARBINARY(size)	Similar to Binary() but the length is variable.
TINYBLOB	For Binary Large Objects. Max size=255 bytes.
TINYTEXT	Holds string of max length 255 characters.
TEXT(Size)	Stores a string of max length 65535 bytes.
BLOB	Stores Binary Large Objects up to 65535 bytes of data.
MEDIUMTEXT	Stores $2^8$ times the characters as compared to TINYTEXT.
MEDIUMBLOB	Stores $2^8$ times bytes as compared to TINYBLOB.
LONGTEXT	Stores $2^8$ times the characters as compared to MEDIUMTEXT.
LONGBLOB	Stores $2^8$ times bytes as compared to MEDIUMBLOB.
ENUM(val1, val2, ...)	Stores only one value, which can be chosen from a range of possible values.

DATATYPE	DETAILS
val3, ...)	An ENUM list can contain at most 65535 values. A value that is inserted that is not in the list will be replaced with a blank value. The values are arranged in the order you specify them.
SET(val1, val2, val3, ...)	Stores a string object that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values in a SET list

## Numeric Data Types

DATATYPE	DETAILS
BIT(size)	Stores a bit-value. The size parameter specifies the number of bits per value . The value is represented as a number of bits. The size parameter can hold a value from 1 to 64. The default value for size is 1.
TINYINT(size)	Stores very small int values. Signed ranges from -128 to 127. Unsigned ranges from 0 to 255. Size defines the maximum display width of 255.
BOOL	Zero is considered as false and one is considered as true.
BOOLEAN	Same as BOOL.
SMALLINT(size)	Stores a small integer. Signed ranges from -32768 to 32767. Unsigned ranges from 0 to 65535. Size defines the maximum display width of 255.
MEDIUMINT(size)	Stores a medium valued integer. Signed ranges from -8388608 to 8388607. Unsigned ranges from 0 to 16777215. Size defines the maximum display width of 255.
INT(size)	Stores a medium integer. Signed ranges from -2147483648 to 2147483647. Unsigned ranges from 0 to 4294967295. Size defines the maximum display width of 255.
INTEGER(size)	Same as INT(size)
BIGINT(size)	Stores a large valued integer. Signed ranges from -9223372036854775808 to 9223372036854775807. Unsigned ranges from 0 to 18446744073709551615. Size defines the maximum display width of 255.

DATATYPE	DETAILS
FLOAT(size, d)	Stores a floating point(decimal number). The number of digits is specified in size. The number of digits after the decimal point is specified by the value d.
FLOAT(p)	Stores a floating point(decimal number). If p value is between 0 and 24, the data type becomes FLOAT() else the data type becomes DOUBLE()
DOUBLE(size, d)	Stores a normal-size floating point (decimal)number. The number of digits is specified in size. The number of digits after the decimal point is specified by the value d.
DECIMAL(size, d)	An exact fixed-point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter. The maximum number for size is 65. The maximum number for d is 30. The default value for size is 10. The default value for d is 0.

## Date and Time Data Types

DATATYPE	DETAILS
DATE	Stores a date in the format: YYYY-MM-DD. Supports a range between '1000-01-01' to '9999-12-31'
DATETIME(fsp)	Combination of date and time in the format: YYYY-MM-DD hh:mm:ss. Supports a range between '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.
TIMESTAMP(fsp)	Stores a time stamp in the format YYYY-MM-DD hh:mm:ss UTC. Supports a range between '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC.
TIME(fsp)	Stores time in the format hh:mm:ss. Supports a range between '-838:59:59' to '838:59:59'
YEAR	Stores a year in four-digit format. Supports a range between 1901 to 2155 (includes 0000).

## **MySQL AGGREGATE FUNCTIONS**

A function that performs an arithmetic operation on a set of values and returns a single value is called an aggregate function.

COMMAND	FUNCTION	SYNTAX
count()	Returns the number of rows, (including NULL)	>SELECT COUNT(COLUMN_NAME) FROM TABLE_NAME WHERE CONDITION;
sum()	Returns sum of all non NULL values.	>SELECT SUM(COLUMN_NAME) FROM TABLE_NAME WHERE CONDITION;
avg()	Returns average of all non NULL values.	>SELECT AVG(COLUMN_NAME) FROM TABLE_NAME WHERE CONDITION;
min()	Returns minimum value in the set.	>SELECT MIN(COLUMN_NAME) FROM TABLE_NAME WHERE CONDITION;
max()	Returns maximum value in the set.	>SELECT MAX(COLUMN_NAME) FROM TABLE_NAME WHERE CONDITION;
group_concat()	Concatenates values from multiple rows into one field.	>SELECT COLUMN1, COLUMN2, ... GROUP_CONCAT ( DISTINCTCOLUMN1 ORDER BY .. ) FROM TABLE_NAME GROUP BY COLUMN2;

## **INDEXES AND VIEWS IN MySQL**

An Index retrieves data much faster than otherwise. Indexes speed up the query/search. A user cannot view an Index. Updating a table with an index takes more time because both table and index have to be updated.

The view is a virtual table which takes the result of an SQL query. Users can access a View. They have rows and columns similar to a table.

COMMAND	FUNCTION	SYNTAX
create index	Creates a new index from an existing table. Allows duplicate values.	> CREATE INDEX indexname ON tablename (column1, column2, ...);
create index unique	Similar to creating an index. But only allows unique values.	>CREATE UNIQUE INDEX indexname ON tablename (column1, column2, ...);
drop index	Deletes an existing index.	> DROP INDEX INDEXNAME;
rebuild index	Used to rebuild one or all indexes in a table if corrupted.	>REINDEX INDEX INDEXNAME;
create view	Creates a view if it doesn't exist.	> CREATE VIEW VIEWNAME AS SELECT COLUMN1,COLUMN2 FROM TABLE WHERE CONDITION;
update view	Creates or edits an existing view.	> CREATE OR REPLACE viewname AS SELECT COLUMN1,COLUMN2 FROM TABLE WHERE CONDITION;
rename view	Changes the name of the view.	> RENAME TABLE VIEWNAME TO NEWVIEWNAME;
drop view	Deletes an existing view.	> DROP VIEW VIEWNAME;
drop views	Deletes multiple views.	> DROP VIEW VIEW1,VIEW2...;
show views	Displays all views in a database.	> SHOW FULL TABLES [ {FROM   IN } databasename] WHERE table_type = 'VIEW';

## **TRIGGERS IN MYSQL**

Triggers are DBMS objects which are associated with tables. Triggers are fired when any one of the DML statements (INSERT, DELETE or UPDATE) is activated.

There are two types of triggers,

- Row Level Triggers: A trigger is an instruction that causes a row to trigger to be fired once for each row affected by an insert, update, or delete statement. The row trigger is fired automatically.

- Statement Level Trigger: Trigger is fired once regardless of the number of DML statements.

**There are six types of triggers, namely,**

- Before Insert: Activated before insertion.
- After Insert: Activated after insertion.
- Before Update: Activated before updating.
- After Update: Activated after updating.
- Before Delete: Activated before deletion.
- After Delete: Activated after deletion.

COMMAND	FUNCTION	SYNTAX
create trigger	Creates a new trigger on an existing table.	>CREATE TRIGGER TRIGGERNAME BEFORE   AFTER INSERT   UPDATE  DELETE ON TABLENAME FOR EACH ROW TRIGGERBODY;
drop trigger	Deletes an existing trigger.	> DROP TRIGGER TRIGGERNAME;
show all triggers	Displays all the triggers in the database.	> SHOW TRIGGERS FROM   IN DATABASE_NAME WHERE SEARCH_CONDITION;

## **STORED PROCEDURES AND FUNCTION**

Procedures are reusable SQL codes that we store in a database. We can directly call procedures instead of writing the query again and again.

Functions are reusable code, which runs certain SQL commands and returns an appropriate value.

**Syntax to create a new procedure.**

```
DELIMITER $
CREATE PROCEDURE procedurename(parameterlist)
BEGIN
    body;
END $
DELIMITER ;
```

## Syntax to create a new function

```
DELIMITER $  
CREATE FUNCTION functionname(parameterlist)  
RETURNS datatype  
NOT DETERMINISTIC  
BEGIN  
%statements%  
END $
```

```
DELIMITER ;
```

COMMAND	FUNCTION	SYNTAX
drop procedure	Deletes an existing procedure.	> <code>DROP PROCEDURE PROCEDURENAME;</code>
show all procedures	Displays all the stored procedures in the database.	> <code>SHOW PROCEDURE STATUS LIKE '%PATTERN'   WHERE CONDITION;</code>
drop function	Deletes an existing stored function.	> <code>DROP FUNCTION FUNCTIONNAME;</code>
show stored functions	Displays all the stored functions.	> <code>SHOW FUNCTION STATUS LIKE '%PATTERN'   WHERE CONDITION;</code>

## INBUILT FUNCTIONS IN MySQL

### STRING FUNCTIONS

Function	Description
ASCII	Returns the ASCII value of a character
CHAR_LENGTH	Returns the length of a string.
CHARACTER_LENGTH	Returns the length of a string
CONCAT	Concatenates two or more expressions.
CONCAT_WS	Concatenates with a separator.
FIELD	Returns the index of value in a list.

Function	Description
FIND_IN_SET	Returns the index of a string within a list.
FORMAT	Changes the format/representation.
INSERT	Inserts a string within a string at a given index.
INSTR	Returns the index of the first occurrence of a string in another one.
LCASE	Converts an entire string to lowercase.
LEFT	Extracts a length of characters from the left of a string.
LENGTH	Returns the string length in bytes.
LOCATE	Returns the location of the first occurrence of a substring in a given string
LOWER	Converts an entire string to lowercase.
LPAD	Left-pads a string with a given string.
LTRIM	Removes spaces from the left of a string.
MID	Extracts a substring from a string at a given position.
POSITION	Returns the location of the first occurrence of a substring in a given string
REPEAT	Repeats the string the number of times the user specifies.
REPLACE	Replaces occurrences of a substring in a string with another substring.
REVERSE	Reverses the string.
RIGHT	Extracts a length of characters from the right of a string.
RPAD	Right-pads a string with a given string.
RTRIM	Removes spaces from the right of a string.
STRCMP	Checks whether two strings are equal.
SUBSTR	Extracts a substring from a string at a position mentioned by the user.

Function	Description
SUBSTRING	Same as substr.
TRIM	Trims leading and trailing spaces from a string as specified by the user.
UCASE	Converts an entire string to uppercase.
UPPER	Converts an entire string to uppercase.

## NUMERIC FUNCTIONS

Function	Description
ABS	Returns the absolute value.
ACOS	Returns the cosine inverse.
ASIN	Returns the sine inverse.
ATAN	Returns the tan inverse of one or two numbers.
ATAN2	Returns the tan inverse of two numbers.
AVG	Returns the mean value.
CEIL	Returns the smallest integer that is greater than or equal to the number
CEILING	Returns the smallest integer that is greater than or equal to the number
COS	Returns the cosine.
COT	Returns the cotangent.
COUNT	Returns the number of records returned by a query.
DEGREES	Converts angle in Radians to Degrees.
DIV	Integer division
EXP	Returns e raised to the power of value mentioned.

Function	Description
FLOOR	Returns the largest integer that is less than or equal to a number
GREATEST	Returns the largest value in the list.
LEAST	Returns the smallest value in the list.
<u>LN</u>	Calculates logarithm to the base e.
LOG	Calculates logarithm to the base e.
LOG10	Calculates logarithm to the base 10.
LOG2	Calculates logarithm to the base 2.
MAX	Returns the largest value in a set.
MIN	Returns the least value in a set.
MOD	Returns the remainder after division of two numbers.
PI	Returns value of $\pi$
POW	Used for exponents.
POWER	Used for exponents.
RADIANS	Converts angle in Degree to Radians.
RAND	Generates a random number.
ROUND	Rounds the number to the nearest decimal place.
SIGN	Returns the sign of a number
SIN	Returns the sine.
SQRT	Returns the root of a number.
SUM	Calculates the sum of a set.
TAN	Returns the tangent.

## MYSQL DATE FUNCTION

Function	Description
ADDDATE	Adds a date interval and return the value.
ADDTIME	Adds a time interval and then returns the value.
CURDATE	Returns today's date
CURRENT_DATE	Same as CURDATE
CURRENT_TIME	Returns the time at the moment
CURRENT_TIMESTAMP	Returns date and time at the moment.
CURTIME	Returns time at the moment.
DATE	Picks up the date from an expression of Date/Time.
DATEDIFF	Returns number of days between two given dates.
DATE_ADD	Similar to ADDDATE
DATE_FORMAT	Changes the format in which Date is displayed.
DATE_SUB	Subtracts a time interval and returns the value.
DAY	Returns the weekday for today.
DAYNAME	Returns the weekday name for any date.
DAYOFMONTH	Used to retrieve the index of the day of the month of any date.
DAYOFWEEK	Used to retrieve the index of the weekday of any date.
DAYOFYEAR	Used to retrieve the index of the day of a year of any date.
EXTRACT	Extracts a part of any date.
HOUR	Returns the "hours" in a given time.
LAST_DAY	Return the last day of the given month.

Function	Description
LOCALTIME	Returns the date and time at the moment.
LOCALTIMESTAMP	Similar to LOCALTIME.
MAKEDATE	Returns a date based on the year and the no. of days you specify.
MAKETIME	Returns a time based on the hours , minutes and seconds you specify.
MICROSECOND	Returns the microseconds in a given time.
MINUTE	Returns the minutes in a given time.
MONTH	Returns the month on a given date.
MONTHNAME	Same as MONTH but returns the name of the month.
NOW	Returns date and time at the moment.
PERIOD_ADD	Adds a specific number of months.
PERIOD_DIFF	Return the difference between two time periods.
SECOND	Return the seconds in a given time.
SEC_TO_TIME	Returns time in seconds.
STR_TO_DATE	Formats the date based on a particular string.
SUBDATE	Same as DATE_SUB.
SUBTIME	Subtracts a time interval.
SYSDATE	Returns the date/time reflected by the system.
TIME	Returns the time from a date/time value.
TIME_FORMAT	Time is displayed based on a certain format.
TIME_TO_SEC	Returns time in seconds.
TIMEDIFF	Returns the difference between two date-time values.

Function	Description
TO_DAYS	Returns the number of days between any date and “0000-00-00”

## ADVANCED MYSQL FUNCTION

Function	Description
BIN	Returns binary value of a given number.
BINARY	Converts a given string to a binary string.
CAST	Converts data from one data type to another.
COALESCE	Returns the first non-null value in a set or list.
CONV	Converts a number from one number-base system to another
CONVERT	Similar to CAST in working
CURRENT_USER	Returns the user name and host name for the MySQL account that is currently used.
DATABASE	Returns the name of the database currently in use.
IF	IF condition statement.
SESSION_USER	Returns the current MySQL user name and host name.
SYSTEM_USER	Similar to SESSION_USER.
USER	Similar to SESSION_USER.
VERSION	Returns the current version of the MySQL server installed.

## Working With Triggers

A **trigger** is a database object, associated with a table. It activates whenever a specific event happens for the table.

For example, you can set up triggers for events such as:

- Row or deletes updates
- Row information inserts

This is a more advanced topic, so check the official MySQL trigger FAQ section for more details

### How to Create a Trigger

To create a simple trigger that will pop up before or after a certain operation such as INSERT, UPDATE or DELETE, use this code:

```
CREATE TRIGGER trigger_name
[BEFORE | AFTER] {INSERT | UPDATE| DELETE }
ON table_name FOR EACH ROW
trigger_body;
```

### Review All Triggers in Your Database

Search your database for all the active triggers using LIKE and WHERE clauses.

```
SHOW TRIGGERS
[ {FROM | IN} database_name]
[LIKE 'pattern' | WHERE search_condition];
```

### How to Delete a Trigger

To remove a trigger, use the DROP command:

```
DROP TRIGGER [IF EXISTS] trigger_name;
```

## Stored Procedures for MySQL

**Stored procedures** are reusable SQL code snippets that you can store in your database and use as-needed over and over again. They save you tons of time since you don't need to write a query from scratch. Instead, you just call it to execute it.

### How to Create a Stored Procedure in MySQL

Here's how to create a simple stored procedure with no additional parameters:

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

And here's another stored procedure example featuring WHERE clause:

```
CREATE PROCEDURE SelectAllMovies @Title varchar(30)
AS
SELECT * FROM Movies WHERE Title = @Title
GO;
```

### Review All Stored Procedures

Similarly to triggers, you can review all stored procedures with LIKE and WHERE:

```
SHOW PROCEDURE STATUS
[LIKE 'pattern' | WHERE search_condition];
```

### How to Delete a Stored Procedure

To get rid of a stored procedure you no longer need, use DROP:

```
DROP PROCEDURE [IF EXISTS] procedure_name;
```



## PHP & MySQL Programming Steps :

### 1) Connection

```
$conn = mysqli_connect(Server Name, User Name, Password, Database Name)  
$conn = new mysqli(Server Name, User Name, Password, Database Name)
```

### 2) Run SQL Query

```
mysqli_query(Connection Name, SQL Query)  
$conn->query(SQL Query)
```

### 3) Close Connection

```
mysqli_close(Connection Name)  
$conn->close()
```



## PHP MySQLi Fetch Functions :

### Procedural Methods

- mysqli\_fetch\_assoc()
- mysqli\_fetch\_array()
- mysqli\_fetch\_row()
- mysqli\_fetch\_all()
- mysqli\_fetch\_field()

```
$result = $conn->query(SQL Query);  
$result->fetch_assoc();  
$result->fetch_array();  
$result->fetch_row();  
$result->fetch_all();  
$result->fetch_field();
```



## PHP MySQLi Error Functions :

### Procedural Methods

- mysqli\_connect\_error()
- mysqli\_connect\_errno()
- mysqli\_error()
- mysqli\_error\_list()

### Object Oriented Methods

```
$conn->connect_error();  
$conn->connect_errno();  
$conn->error();  
$conn->error_list();
```



## Guidelines of Index :

- Automatically creates the indexes for PRIMARY KEY and UNIQUE columns.
- Index columns that you frequently use to retrieve the data.
- Index columns that are used for joins to improve join performance.
- Avoid columns that contain too many NULL values.
- Small tables do not require indexes.



## What is PRIMARY KEY Constraint ?

- Primary key always has unique data.
- A primary key cannot have null value.
- A table can contain only one primary key constraint.

**Student Table**

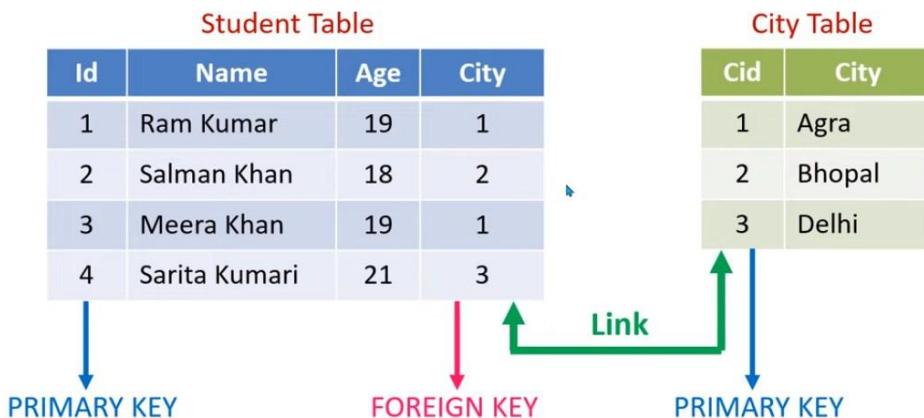
**PRIMARY KEY** →

- Unique Data
- No Null Value

<b>Id</b>	<b>Name</b>	<b>Age</b>	<b>City</b>
1	Ram Kumar	19	Agra
2	Salman Khan	18	Bhopal
3	Meera Khan	19	Agra
4	Sarita Kumari	21	Delhi



## FOREIGN KEY in Table



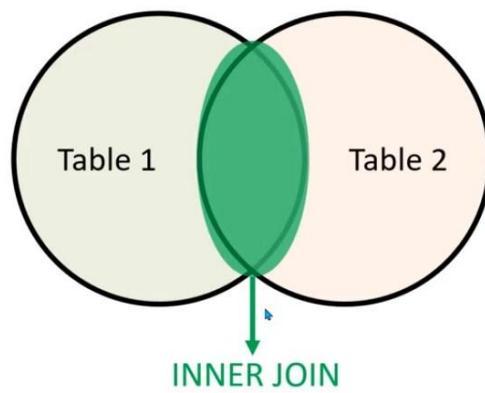


## Create Table with FOREIGN KEY Syntax

```
CREATE TABLE student(  
    id INT NOT NULL AUTO_INCREMENT,  
    name VARCHAR(50) NOT NULL,  
    age INT NOT NULL,  
    city VARCHAR(10) NOT NULL ,  
    PRIMARY KEY (id),  
    FOREIGN KEY (city) REFERENCES City (cid)  
);
```



## What is INNER JOIN ?



The INNER JOIN selects records that have matching values in both tables.



## INNER JOIN Syntax

**SELECT** columns

**FROM** table1

**INNER JOIN** table2

**ON** table1.column\_name = table2.column\_name;

FOREIGN KEY

PRIMARY KEY

**Yahoo Baba** [www.yahooibaba.net](http://www.yahooibaba.net)

personal personal city

```
1 SELECT * FROM personal INNER JOIN city
2 ON personal.city = city.cid;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	id	name	percentage	age	gender	city	cid	cityname
>	1	Ram Kumar	45	19	M	1	1	Agra
	2	Sarita Kumari	55	22	F	2	2	Delhi
	3	Salman Khan	62	20	M	1	1	Agra
	4	Juhি Chawla	47	18	F	3	3	Bhopal
	5	Anil Kapoor	74	22	M	1	1	Agra
	6	John Abraham	64	21	M	2	2	Delhi
	7	Shahid Kapoor	52	20	M	1	1	Agra

The screenshot shows the phpMyAdmin interface. In the top navigation bar, the 'personal' schema is selected. Below it, the 'city' table is selected. The SQL query window contains the following code:

```
1 SELECT * FROM personal p INNER JOIN city c
2 ON p.city = c.cid;
```

The result grid displays the following data:

	id	name	percentage	age	gender	city	cid	cityname
▶	1	Ram Kumar	45	19	M	1	1	Agra
	2	Sarita Kumari	55	22	F	2	2	Delhi
	3	Salman Khan	62	20	M	1	1	Agra
	4	Juhi Chawla	47	18	F	3	3	Bhopal
	5	Anil Kapoor	74	22	M	1	1	Agra
	6	John Abraham	64	21	M	2	2	Delhi
	7	Shahid Kapoor	52	20	M	1	1	Agra

The screenshot shows the MySQL Workbench interface. In the top navigation bar, the 'personal' schema is selected. Below it, the 'city' table is selected. The SQL query window contains the following code:

```
1 SELECT p.id,p.name,p.percentage,p.age,p.gender,c.cityname
2 FROM personal p INNER JOIN city c
3 ON p.city = c.cid;
```

The result grid displays the following data:

	id	name	percentage	age	gender	cityname
▶	1	Ram Kumar	45	19	M	Agra
	2	Sarita Kumari	55	22	F	Delhi
	3	Salman Khan	62	20	M	Agra
	4	Juhi Chawla	47	18	F	Bhopal
	5	Anil Kapoor	74	22	M	Agra
	6	John Abraham	64	21	M	Delhi
	7	Shahid Kapoor	52	20	M	Agra

The screenshot shows the phpMyAdmin interface. In the top navigation bar, the schema is set to 'personal'. Below it, the 'Tables' section shows two tables: 'personal' and 'city'. A query is entered in the SQL tab:

```

1  SELECT p.id,p.name,p.percentage,p.age,p.gender,c.cityname
2  FROM personal p INNER JOIN city c
3  ON p.city = c.cid
4  WHERE c.cityname = "Agra";

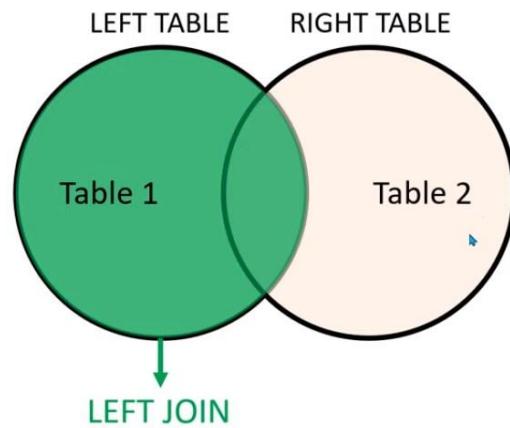
```

The results are displayed in a grid:

	id	name	percentage	age	gender	cityname
▶	1	Ram Kumar	45	19	M	Agra
	3	Salman Khan	62	20	M	Agra
	5	Anil Kapoor	74	22	M	Agra
	7	Shahid Kapoor	52	20	M	Agra



## What is LEFT JOIN ?

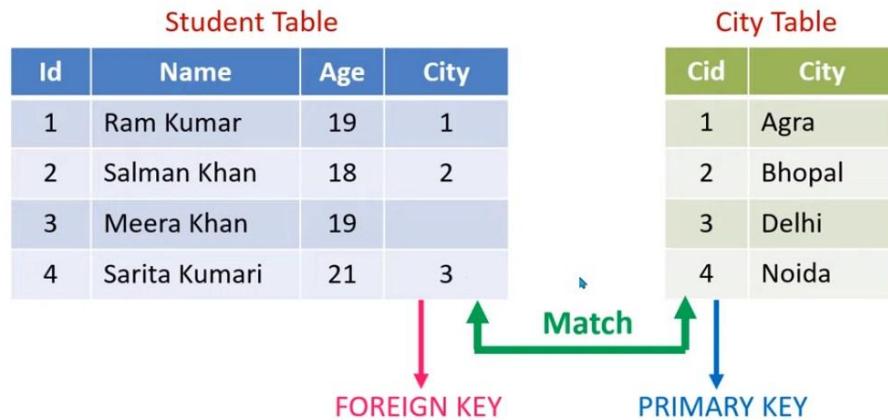


The LEFT JOIN returns all records from the left table (table1),  
and the matched records from the right table (table2)



## JOIN Two Tables

### LEFT JOIN



## LEFT JOIN Syntax

SELECT columns

FROM table1

LEFT JOIN table2

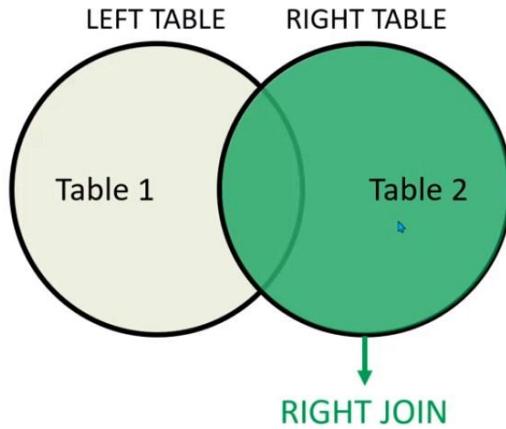
ON table1.column\_name = table2.column\_name;

FOREIGN KEY

PRIMARY KEY



## What is RIGHT JOIN ?



The **RIGHT JOIN** returns all records from the right table (table2),  
and the matched records from the left table (table1).

## Triggers in MySQL

- Trigger is something which automatically generates/executes the code following the condition with which it is written.

Basically ,Triggers are divided into following types

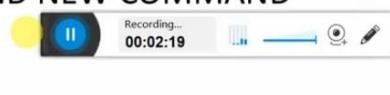
- 1.Before/After Insert
- 2.Before/After Update
- 3.Before/After Delete

All these follows a simple

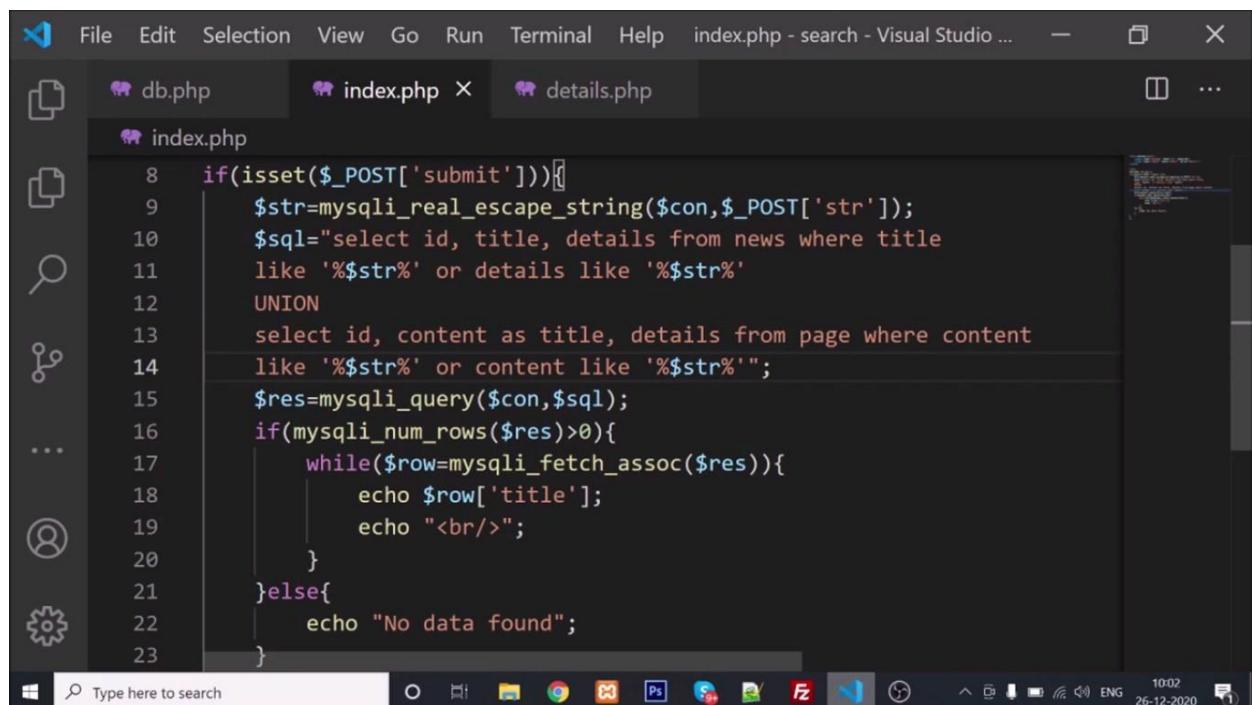


## Syntax of Triggers in MySQL

- CREATE TRIGGER TRIGGER NAME  
BEFORE/AFTER(UPDATE/DELETE/INSERT) ON TABLE NAME FOR EACH  
ROW
  - BEGIN
  - <CODE>
  - END;
- 
- OLD COMMAND AND NEW COMMAND



Grima



```
8 if(isset($_POST['submit'])){  
9     $str=mysqli_real_escape_string($con,$_POST['str']);  
10    $sql="select id, title, details from news where title  
11        like '%$str%' or details like '%$str%'  
12        UNION  
13        select id, content as title, details from page where content  
14        like '%$str%' or content like '%$str%';  
15    $res=mysqli_query($con,$sql);  
16    if(mysqli_num_rows($res)>0){  
17        while($row=mysqli_fetch_assoc($res)){  
18            echo $row['title'];  
19            echo "<br/>";  
20        }  
21    }else{  
22        echo "No data found";  
23    }  
}
```

- Stored Procedure is a sql code that you can save into database so that code can be reuse.
- So if you have an SQL query that you write again and again, save it as Stored Procedure
- Stored Procedure reduce network traffic and increase performance
- Stored Procedure will accept input parameters so that a single procedure used again and again

Syntax:-

```
create procedure procedure_name  
begin  
    select * from table_name;  
end;  
  
call procedure_name();
```

## Add routine

### Details

Routine name

Type

Parameters 

Direction	Name	Type	Length/Values	Options
IN	uid	INT		<input checked="" type="checkbox"/> Drop

[Add parameter](#)

```
1 BEGIN
2   select * from user where id=uid;
3 end;
```

### Definition

Is deterministic

Definer

Security type

SQL data access

Comment

[Go](#)

[Close](#)

## index.php

```
1 <?php
2 $con=mysqli_connect('localhost','root','','youtube');
3 $sql="call getUserInfo(3);";
4 $res=mysqli_query($con,$sql);
5 $row=mysqli_fetch_assoc($res);
6 echo '<pre>';
7 print_r($row);
8 ?>
```

Add routine

Details

Routine name: **getUserInfo1**

Type: PROCEDURE

Parameters	Direction	Name	Type	Length/Values	Options
	IN	uid	INT		Drop
	OUT	uname	VARC	50	Char Drop

Add parameter

```
1 BEGIN
2
3 select name into uname from user where id=uid;
4
5 end;
```

Definition

Go Close

File Edit View Insert Tools Help

<?php

```
$con=mysqli_connect('localhost','root','','youtube');
$sql="call getUserInfo1(3,@uname);";
$res1=mysqli_query($con,$sql);
$res=mysqli_query($con,"select @uname");
$row=mysqli_fetch_assoc($res);
echo '<pre>';
print_r($row);
?>
```

Add routine

**Details**

|              |              |        |         |               |                               |
|--------------|--------------|--------|---------|---------------|-------------------------------|
| Routine name | calculateSal |        |         |               |                               |
| Type         | PROCEDURE    |        |         |               |                               |
| Parameters   | Direction    | Name   | Type    | Length/Values | Options                       |
|              | IN           | uid    | INT     |               | <input type="checkbox"/> Drop |
|              | OUT          | result | VARCHAR | 50            | <input type="checkbox"/> Drop |

**Add parameter**

```

1 BEGIN
2
3 DECLARE usal int;
4
5 SELECT salary into usal from user WHERE id=uid;
6
7 IF (usal > 10000) THEN
8     SET result="Good";
9 ELSE
10    set result="Bad";
11 END if;
12
13 END;
14

```

**Definition**

```

$con=mysqli_connect('localhost','root','','youtube');
/*$sql="call getUserInfo(3,@uname);";
$res1=mysqli_query($con,$sql);
$res=mysqli_query($con,"select @uname");
$row=mysqli_fetch_assoc($res);
echo '<pre>';
print_r($row);*/


/*
$sql="call insertData('Nait',1000);";
mysqli_query($con,$sql);
*/


$sql="call calculateSal(3,@result);";
$res1=mysqli_query($con,$sql);
$res=mysqli_query($con,"select @result");
$row=mysqli_fetch_assoc($res);
echo '<pre>';
print_r($row);

```



## Conditional Ternary Operator Syntax :

(Condition) ? True Statement : False Statement

```
$x = 15;
```

```
($x > 10) ? $z = "True" : $z = "False";
```

```
$x = 15;
```

```
($x > 10) ? $z = "True" : $z = "False";
```

```
echo $z;
```

```
$x = 15;  
  
$z = $x > 20? "Greater" : "Smaller";  
  
echo $z;
```

```
$x = 15;  
  
$z = "Value is : " . ($x > 20 ? "Greater" : "Smaller");  
  
echo $z;
```

## String Operator

```
$a = "Hello ";
$a .= "this is ";
$a .= "my world.";
echo $a;
```

Hello this is my world.

## Goto Statement in PHP :

```
for($a = 1; $a <= 10; $a++){
    if ($a == 3){
        goto abc;
    }
    echo $a;
}
echo "Hello";
abc:
echo "This is new code";
```

```
<?php
echo "Hello Yahoo Baba<br>";
echo "Hello Yahoo Baba<br>";
goto abc;
echo "Hello Yahoo Baba<br>";
echo "Hello Yahoo Baba<br>";
echo "Hello Yahoo Baba<br>";
echo "Hello Yahoo Baba<br>";
abc:
echo "Hey this is label ABC.";
```

Hello Yahoo Baba  
Hello Yahoo Baba  
Hey this is label ABC.

## PHP Constant Variables

```
define(num, 500, true);  
define(_num, 500, true);
```

Can't use \$ sign with constant variable name.

Constant Variables are **Global Variables**.

## Continue Statement in PHP :

```
for($a = 1; $a <= 10; $a++){  
    if ($a == 3){  
        continue;  
    }  
    echo $a;  
}
```

## Break Statement in PHP :

```
for($a = 1; $a <= 10; $a++){  
    if ($a == 3){  
        break;  
    }  
    echo $a;  
}
```

## Date & Time Datatypes in MySQL

1. DATE '1000-01-01' to '9999-12-31'
2. DATETIME(fsp) YYYY-MM-DD hh:mm:ss
3. TIMESTAMP(fsp)
4. TIME(fsp) hh:mm:ss
5. YEAR four-digit format : 1901



## PHP : Three Steps to Set & Get SESSION Value

**Step 1 :** session\_start();

**Step 2 :** \$\_SESSION[name] = value; Set Session name & value

**Step 3 :** echo \$\_SESSION[name]; Get Session value

↓

### Delete Session

**Step 1 :** session\_unset(); Remove all session variables

**Step 2 :** session\_destroy(); Destroy the session

---

```
<?php
//Caching is a temporary storage location, to access data more quickly. Caching is used where the
original data is expensive to fetch.
$start=microtime(true);
$con=new PDO("mysql:host=localhost;dbname=youtube","root","");
$sql="select student.name, city.city, game.game, study.study, teacher.teacher from
student,city,game,study,teacher,fee where student.city=city.id and student.game=game.id and
student.study=study.id and student.teacher=teacher.id and student.id=fee.student_id";
$stmt=$con->prepare($sql);
$stmt->execute();
$arr=$stmt->fetchAll(PDO::FETCH_ASSOC);
$str=<table border='1'>;
    $str.=<tr><td>Name</td><td>City</td><td>Game</td><td>Study</td><td>Teacher</td></tr>;
    foreach($arr as $list){
        $str.=<tr><td>".$list['name']."'</td><td>".$list['city']."'</td><td>".$list['game']."'</td>
        $list['study']."'</td><td>".$list['teacher']."'</td></tr>";
    }
$str.=</table>;
echo $str;
$end=microtime(true);
echo round($end-$start,4);
?>
```

## Caching in PHP SQL

```
<?php
//Caching is a temporary storage location, to access data more quickly. Caching
is used where the original data is expensive to fetch.
$start=microtime(true);
$con=new PDO("mysql:host=localhost;dbname=youtube","root","");
$cache_file="cache/index.cache.php";
if(file_exists($cache_file) && filemtime($cache_file) > time()-20){
    echo "From Cache<br/>";
    include($cache_file);
} else{
    $sql="select student.name, city.city, game.game, study.study, teacher.teacher
from student,city,game,study,teacher,fee where student.city=city.id and
student.game=game.id and student.study=study.id and student.teacher=teacher.id
and student.id=fee.student_id";
    $stmt=$con->prepare($sql);
    $stmt->execute();
    $arr=$stmt->fetchAll(PDO::FETCH_ASSOC);
    $str=<table border='1'>;
    $str.=<tr><td>Name</td><td>City</td><td>Game</td><td>Study</td><td>Teacher</td></tr>;
    foreach($arr as $list){
        $str.=<tr><td>".$list['name']."'</td><td>".$list['city']."'</td><td>".$list['game']."'</td><td>".$list['study']."'</td><td>".$list['teacher']."'</td></tr>;
    }
    $str.=</table>;
    $handle=fopen($cache_file,'w');
    fwrite($handle,$str);
    fclose($handle);
    echo "Cache Created<br/>";
    echo $str;
}
$end=microtime(true);
echo round($end-$start,4);
?>
```

```
1 SELECT * FROM `users` where soundex('Bishal')=soundex(name)|
```

```
SELECT * FROM EmployeeFullName  
WHERE SOUNDEX(Last_Name)=soundex('Reddy')
```

www.thundershare.net

The screenshot shows a SQL query results table titled "EmployeeFullName". The columns are Emp\_Id, First\_Name, Middle\_Name, and Last\_Name. The data includes rows for employees with Last Names like Reddy, Reddi, and Redy.

|   | Emp_Id | First_Name | Middle_Name | Last_Name |
|---|--------|------------|-------------|-----------|
| 1 | 2      | Bhargav    | C           | Reddy     |
| 2 | 3      | Chaitanya  | A           | Reddi     |
| 3 | 4      | Bharath    | B           | Reddy     |
| 4 | 5      | Kunal      | H           | Redy      |
| 5 | 7      | Arav       | L           | Reddy     |
| 6 | 8      | Dhruv      | L           | Redi      |
| 7 | 10     | Lehari     | C           | Reddi     |

```
1 <?php  
2 //define("Msg", "Hello To My Youtube Channel");  
3 //echo Msg;  
4 //echo constant("Msg");  
5 const MSG="Hello friends";  
6  
7 echo MSG;  
8  
9 ?>  
<?php  
//define("Msg", "Hello To My Youtube Channel");  
//echo Msg;  
//echo constant("Msg");  
const MSG="Hello friends";  
  
function testContent(){  
    echo MSG;  
}  
testContent();  
?>
```

## PHP Login logout example with session

```
if(mysqli_num_rows($result) > 0){

    while($row = mysqli_fetch_assoc($result)){
        session_start();
        $_SESSION["username"] = $row['username'];
        $_SESSION["user_id"] = $row['user_id'];
        $_SESSION["user_role"] = $row['role'];

        header("Location: {$hostname}/admin/post.php");
    }
} else{
    echo '<div class="alert alert-danger">Username and Password are incorrect</div>';
}
?>
```

```
<?php

include "config.php";

session_start();

session_unset();

session_destroy();

header("Location: {$hostname}/admin/");

?>
```

```
<?php
    include "config.php";
    session_start();
    if(!isset($_SESSION["username"])){
        header("Location: {$hostname}/admin/");
    }
?>
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
```

## Second method

```
CREATE TABLE `login_user` (
    `id` int(11) NOT NULL,
    `name` varchar(60) NOT NULL,
    `user_name` varchar(20) NOT NULL,
    `password` varchar(20) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

## *login.php*

```
<?php
    session_start();
    $message="";
    if(count($_POST)>0) {
        $con = mysqli_connect('127.0.0.1:3306','root','','admin') or die('Unable
To connect');
        $result = mysqli_query($con,"SELECT * FROM login_user WHERE user_name='"
. $_POST["user_name"] . "' and password = '". $_POST["password"]."')");
        $row = mysqli_fetch_array($result);
        if(is_array($row)) {
            $_SESSION["id"] = $row['id'];
            $_SESSION["name"] = $row['name'];
        } else {
            $message = "Invalid Username or Password!";
        }
    }
    if(isset($_SESSION["id"])) {
```

```

        header("Location:index.php");
    }
?>
<html>
<head>
<title>User Login</title>
</head>
<body>
<form name="frmUser" method="post" action="" align="center">
<div class="message"><?php if($message!="") { echo $message; } ?></div>
<h3 align="center">Enter Login Details</h3>
    Username:<br>
    <input type="text" name="user_name">
    <br>
    Password:<br>
    <input type="password" name="password">
    <br><br>
    <input type="submit" name="submit" value="Submit">
    <input type="reset">
</form>
</body>
</html>

```

## *index.php*

```

<?php
session_start();
?>
<html>
<head>
<title>User Login</title>
</head>
<body>

<?php
if($_SESSION[ "name" ]) {
?>
Welcome <?php echo $_SESSION[ "name" ]; ?>. Click here to <a href="logout.php"
title="Logout">Logout.
<?php
}else echo "<h1>Please login first .</h1>";
?>
</body>
</html>

```

*logout.php*

```
<?php
session_start();
unset($_SESSION["id"]);
unset($_SESSION["name"]);
header("Location:login.php");
?>
```

## How to add captcha in PHP form

*index.php*

```
<?php
session_start();
if ($_POST["vercode"] != $_SESSION["vercode"] OR $_SESSION["vercode"]==" ")
    echo "<script>alert('Incorrect verification code');</script>" ;
}
else{
    echo "<script>alert('Verification code match !');</script>" ;
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Bootstrap Elegant Account Login Form with Avatar Icon</title>
<link rel="stylesheet"
href="https://fonts.googleapis.com/icon?family=Material+Icons">
<link href="https://fonts.googleapis.com/css?family=Roboto|Varela+Round"
rel="stylesheet">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>
```

```

<div class="login-form">
    <form action="" method="post">
        <div class="avatar"><i class="material-icons">鍊</i></div>
        <h4 class="modal-title">Login to Your Account</h4>
        <div class="form-group">
            <input type="text" class="form-control" placeholder="Username"
required="required">
        </div>
        <div class="form-group">
            <input type="password" class="form-control" placeholder="Password"
required="required">
        </div>
        <div class="form-group">
            <input type="text" name="vercode" class="form-control"
placeholder="Verfication Code" required="required">
        </div>
        <div class="form-group small clearfix">
            <label class="checkbox-inline">Verification Code</label>
            
        </div>
        <input type="submit" class="btn btn-primary btn-block btn-lg"
value="Login">
    </form>
</div>
</body>
</html>

```

## captcha.php

```

<?php
    session_start();
    $text = rand(10000,99999);
    $_SESSION["vercode"] = $text;
    $height = 25;
    $width = 65;
    $image_p = imagecreate($width, $height);
    $black = imagecolorallocate($image_p, 0, 0, 0);
    $white = imagecolorallocate($image_p, 255, 255, 255);
    $font_size = 14;
    imagestring($image_p, $font_size, 5, 5, $text, $white);
    imagejpeg($image_p, null, 80);
?>

```

## PHP User Form

```
<?php
/*
* Script Name: PHP Form Login Remember Functionality with Cookies
* Source: www.TutorialsClass.com
*/
?>
<form action="page2.php" method="post" style="border: 2px dotted blue; text-align:center; width: 400px;">
    <p>
        Username: <input name="username" type="text" value="<?php
if(isset($_COOKIE["username"])) { echo $_COOKIE["username"]; } ?>">
        </input>
    </p>
    <p>Password: <input name="password" type="password" value="<?php
if(isset($_COOKIE["password"])) { echo $_COOKIE["password"]; } ?>">
        </input>
    </p>
    <p><input type="checkbox" name="remember" /> Remember me
    </p>
    <p><input type="submit" value="Login"></span></p>
</form>
```

Save username/password in cookies if remember me is selected

```
<?php
/*
* Website: www.TutorialsClass.com
*/
if(!empty($_POST["remember"])) {
    setcookie ("username",$_POST["username"],time()+ 3600);
    setcookie ("password",$_POST["password"],time()+ 3600);
    echo "Cookies Set Successfully";
} else {
    setcookie("username","");
    setcookie("password","");
    echo "Cookies Not Set";
}

?>

<p><a href="page1.php"> Go to Login Page </a> </p>
```

```
1 <div align="center">
2 <form action="" method="post" id="login">
3 <div class="error-message"><?php if(isset($message)) { echo $message; } ?></di
v>
4 <h4> PHP login with remember me function</h4>
5 <div class="field-group">
6 <div><label for="login">Username</label></div>
7 <div>
8 <input name="username" type="text" value=<?php if(isset($_COOKIE["user_logi
n"])) { echo $_COOKIE["user_login"]; } ?>" class="input-field">
9 </div>
10 <div class="field-group">
11 <div><label for="password">Password</label></div>
12 <div><input name="password" type="password" value=<?php if(isset($_CO
OKIE["userpassword"])) { echo $_COOKIE["userpassword"]; } ?>" class="input-fie
ld">
13 </div>
14 <div class="field-group">
15 <div><input type="checkbox" name="remember" id="remember" <?php if(is
et($_COOKIE["user_login"])) { ?> checked <?php } ?> />
16 <label for="remember-me">Remember me</label>
17 </div>
18 <div class="field-group">
19 <div><input type="submit" name="login" value="Login" class="form-submi
t-button"></span></div>
20 </div>
21 </form>
22 </div>
```

## Step 2 : Create a SQL table tbluser

```
1 CREATE TABLE `tbluser` (
2   `id` int(11) NOT NULL,
3   `userName` varchar(200) DEFAULT NULL,
4   `userPassword` varchar(255) DEFAULT NULL
5 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
6 ALTER TABLE `tbluser`
7   ADD PRIMARY KEY (`id`);
```

### Step 3 : Create database configuration file config.php

```
1 <?php
2 define('DB_SERVER','localhost');
3 define('DB_USER','root');
4 define('DB_PASS','');
5 define('DB_NAME', 'test'); // your database name
6 $conn = mysqli_connect(DB_SERVER,DB_USER,DB_PASS,DB_NAME);
7 // Check connection
8 if (mysqli_connect_errno())
9 {
10 echo "Failed to connect to MySQL: " . mysqli_connect_error();
11 }
12 ?>
```

### Step 4 : PHP Cookies to remember username and password. Username and password will be stored \$\_COOKIE array.

```
1 <?php
2 session_start(); // starting a session
3 include('config.php'); // Database configuration file
4 if(isset($_POST["login"]))
5 {
6 // Getting Post Vlaues
7 $username=$_POST['username'];
8 $password=md5($_POST['password']); // MD5 password encryption
9 $sql = "Select * from tbluser where userName ='$username' and userPassword ='$password'";
10 $result = mysqli_query($conn,$sql);
11 $row = mysqli_fetch_array($result);
12 if($row) {
13     $_SESSION["userid"]=$row["id"];
14     // if remember me clicked . Values will be stored in $_COOKIE array
15     if(!empty($_POST["remember"])) {
16 //COOKIES for username
17 setcookie ("user_login",$_POST["username"],time()+ (10 * 365 * 24 * 60 * 60));
18 //COOKIES for password
19 setcookie ("userpassword",$_POST["password"],time()+ (10 * 365 * 24 * 60 * 6
0));
```

```
20 } else {
21 if(isset($_COOKIE["user_login"])) {
22 setcookie ("user_login","");
23 if(isset($_COOKIE["userpassword"])) {
24 setcookie ("userpassword","");
25 }
26 }
27 header('location:welcome.php');
28 } else {
29     $message = "Invalid Login";
30 }
31 }
32 ?>
```

[View Demo](#)-----

**Username :** anuj@gmail.com

**Password :** Test@123

## Logout example

```
< > login.php           x logout.php
1 <?php
2
3 session_start();
4
5 session_destroy();
6
7 setcookie('emailcookie','','time()-86400);
8 setcookie('passwordcookie','','time()-86400);
9
10 header('location:login.php');
11
12 ?>
```

## PHP Visitor Count

```
Index.php
<?php
$servername="localhost";
$username="root";
$password="";
$dbname="traffic";
$conn=mysqli_connect($servername,$username,$password,"$dbname");
if(!$conn){
die('Could not Connect My Sql:' .mysql_error());
}

$find_count=mysqli_query($conn,"SELECT * FROM count_number");
while($row=mysqli_fetch_assoc($find_count))
{
$current_count=$row[ 'count1'];
$new_count=$current_count+1;
try{
$sql= "UPDATE count_number SET count1 = '$new_count' where id=1";
$update_count=mysqli_query ($conn,$sql);
echo $new_count;
}

catch(Exception $e){
echo 'error;'.$e->getMessage();
}
}
?>
```

## Mail with attachment

```
bool mail( $to, $subject, $message, $headers, $parameters );
```

index.php

```
<form enctype="multipart/form-data" method="POST" action="">
    <label>Your Name <input type="text" name="sender_name" /> </label>
    <label>Your Email <input type="email" name="sender_email" /> </label>
    <label>Subject <input type="text" name="subject" /> </label>
    <label>Message <textarea name="message"></textarea> </label>
    <label>Attachment <input type="file" name="attachment" /></label>
    <label><input type="submit" name="button" value="Submit" /></label>
</form>
```

```
<?php
if($_POST['button'] && isset($_FILES['attachment']))
{
    $from_email      = 'sender@abc.com'; //from mail, sender email address
    $recipient_email = 'recipient@xyz.com'; //recipient email address

    //Load POST data from HTML form
    $sender_name = $_POST["sender_name"] //sender name
    $reply_to_email = $_POST["sender_email"] //sender email, it will be used in
    "reply-to" header
    $subject      = $_POST["subject"] //subject for the email
    $message      = $_POST["message"] //body of the email

    /*Always remember to validate the form fields like this
    if(strlen($sender_name)<1)
    {
        die('Name is too short or empty!');
    }
    */

    //Get uploaded file data using $_FILES array
```

```

$tmp_name = $_FILES['my_file']['tmp_name']; // get the temporary file name of
the file on the server
$name     = $_FILES['my_file']['name']; // get the name of the file
$size     = $_FILES['my_file']['size']; // get size of the file for size
validation
$type    = $_FILES['my_file']['type']; // get type of the file
$error   = $_FILES['my_file']['error']; // get the error (if any)

//validate form field for attaching the file
if($file_error > 0)
{
    die('Upload error or No files uploaded');
}

//read from the uploaded file & base64_encode content
$handle = fopen($tmp_name, "r"); // set the file handle only for reading the
file
$content = fread($handle, $size); // reading the file
fclose($handle); // close upon completion

$encoded_content = chunk_split(base64_encode($content));

$boundary = md5("random"); // define boundary with a md5 hashed value

//header
$headers = "MIME-Version: 1.0\r\n"; // Defining the MIME version
$headers .= "From:".$from_email."\r\n"; // Sender Email
$headers .= "Reply-To: ".$reply_to_email."\r\n"; // Email address to reach
back
$headers .= "Content-Type: multipart/mixed;\r\n"; // Defining Content-Type
$headers .= "boundary = $boundary\r\n"; //Defining the Boundary

//plain text
$body = "--$boundary\r\n";
$body .= "Content-Type: text/plain; charset=ISO-8859-1\r\n";
$body .= "Content-Transfer-Encoding: base64\r\n\r\n";
$body .= chunk_split(base64_encode($message));

//attachment
$body .= "--$boundary\r\n";
$body .= "Content-Type: $file_type; name=".$file_name."\r\n";
$body .= "Content-Disposition: attachment; filename=".$file_name."\r\n";
$body .= "Content-Transfer-Encoding: base64\r\n";
$body .= "X-Attachment-Id: ".rand(1000, 99999)."\r\n\r\n";
$body .= $encoded_content; // Attaching the encoded file with email

```

```

$sentMailResult = mail($recipient_email, $subject, $body, $headers);

if($sentMailResult )
{
echo "File Sent Successfully.";
unlink($name); // delete the file after attachment sent.
}
else
{
die("Sorry but the email could not be sent.
Please go back and try again!");
}
}

?>

```

## Pagination

*File: connection.php*

```

<?php
$conn = mysqli_connect('localhost', 'root', '');
if (! $conn) {
die("Connection failed" . mysqli_connect_error());
}
else {
mysqli_select_db($conn, 'pagination');
}
?>

```

*File: index1.php*

```

<html>
<head>
<title>Pagination</title>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<style>
table {

```

```
border-collapse: collapse;
}
.inline{
    display: inline-block;
    float: right;
    margin: 20px 0px;
}

input, button{
    height: 34px;
}

.pagination {
    display: inline-block;
}
.pagination a {
    font-weight:bold;
    font-size:18px;
    color: black;
    float: left;
    padding: 8px 16px;
    text-decoration: none;
    border:1px solid black;
}
.pagination a.active {
    background-color: pink;
}
.pagination a:hover:not(.active) {
    background-color: skyblue;
}
    </style>
</head>
<body>
<center>
<?php

// Import the file where we defined the connection to Database.
require_once "connection.php";

$per_page_record = 4; // Number of entries to show in a page.
// Look for a GET variable page if not found default is 1.
if (isset($_GET["page"])) {
    $page = $_GET["page"];
}
else {


```

```

    $page=1;
}

$start_from = ($page-1) * $per_page_record;

$query = "SELECT * FROM student LIMIT $start_from, $per_page_record";
$rs_result = mysqli_query ($conn, $query);
?>

<div class="container">
<br>
<div>
    <h1>Pagination Simple Example</h1>
    <p>This page demonstrates the basic
        Pagination using PHP and MySQL.
    </p>
    <table class="table table-striped table-condensed
                    table-bordered">
        <thead>
            <tr>
                <th width="10%">ID</th>
                <th>Name</th>
                <th>College</th>
                <th>Score</th>
            </tr>
        </thead>
        <tbody>
<?php
    while ($row = mysqli_fetch_array($rs_result)) {
        // Display each field of the records.
    ?>
        <tr>
            <td><?php echo $row["Rank"]; ?></td>
            <td><?php echo $row["Name"]; ?></td>
            <td><?php echo $row["College"]; ?></td>
            <td><?php echo $row["Score"]; ?>
        <?></td>
        </tr>
        <?php
            ?>;
        ?>
    </tbody>
</table>

<div class="pagination">

```

```

<?php
    $query = "SELECT COUNT(*) FROM student";
    $rs_result = mysqli_query($conn, $query);
    $row = mysqli_fetch_row($rs_result);
    $total_records = $row[0];

echo "</br>";
// Number of pages required.
$total_pages = ceil($total_records / $per_page_record);
$pagLink = "";

if($page>=2){
    echo "<a href='index1.php?page=".($page-1)."'> Prev </a>";
}

for ($i=1; $i<=$total_pages; $i++) {
    if ($i == $page) {
        $pagLink .= "<a class = 'active' href='index1.php?page="
                    . $i ."'">".$i." </a>";
    }
    else {
        $pagLink .= "<a href='index1.php?page=".$i."'"
                    ".$i." </a>";
    }
};
echo $pagLink;

if($page<$total_pages){
    echo "<a href='index1.php?page=".($page+1)."'> Next </a>";
}

?>
</div>

<div class="inline">
<input id="page" type="number" min="1" max="<?php echo $total_pages?>"
placeholder="<?php echo $page."/". $total_pages; ?>" required>
<button onClick="go2Page();>Go</button>
</div>
</div>
</div>
</center>
<script>
function go2Page()

```

```

{
    var page = document.getElementById("page").value;
    page = ((page)<?php echo $total_pages; ?>)?<?php echo $total_pages;
?>:(page<1)?1:page));
    window.location.href = 'index1.php?page=' +page;
}
</script>
</body>
</html>

```

## PHP Pegination 2

```

<?php
$con=mysqli_connect('localhost','root','','youtube');

$per_page=5;
$start=0;
$current_page=1;
if(isset($_GET['start'])){
    $start=$_GET['start'];
    if($start<=0){
        $start=0;
        $current_page=1;
    }else{
        $current_page=$start;
        $start--;
        $start=$start*$per_page;
    }
}
$record=mysqli_num_rows(mysqli_query($con,"select id,title from page"));
$pagi=ceil($record/$per_page);

$sql="select id,title from page limit $start,$per_page";
$res=mysqli_query($con,$sql);
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Pagination Example</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <script src="js/jquery.min.js"></script>

```

```

<script src="js/bootstrap.min.js"></script>
<style>
.mt-100{margin-top:50px;}
.mt-30{margin-top:30px;}
.mb-30{margin-bottom:30px;}
</style>
</head>
<body>

<div class="container mt-100">
<h2 class="mb-30">Pagination Example</h2>
<ul class="list-group">
<?php
if(mysqli_num_rows($res)>0){
while($row=mysqli_fetch_assoc($res)){?>
    <li class="list-group-item"><?php echo $row['title']?></li>
<?php } } else {?>
No records
<?php } ?>
</ul>
<ul class="pagination mt-30">
<?php
for($i=1;$i<=$pagi;$i++){
$class=' ';
if($current_page==$i){
?><li class="page-item active"><a class="page-link"
href="javascript:void(0)"><?php echo $i?></a></li><?php
}else{
?>
    <li class="page-item"><a class="page-link" href=?start=<?php echo
$i?>"><?php echo $i?></a></li>
<?php
}
?>
<?php } ?>
</ul>
</div>

</body>
</html>

```

## Validation PHP function

### PHP Filter Validation :

`filter_var(var, filtername, options/flag)`

- FILTER\_VALIDATE\_INT
- FILTER\_VALIDATE\_FLOAT
- FILTER\_VALIDATE\_BOOLEAN
- FILTER\_VALIDATE\_EMAIL
- FILTER\_VALIDATE\_URL
- FILTER\_VALIDATE\_IP
- FILTER\_VALIDATE\_MAC
- FILTER\_VALIDATE\_REGEXP

```
<?php  
  
$var = 9950;  
  
var_dump(filter_var($var, FILTER_VALIDATE_INT));  
  
if(filter_var($var, FILTER_VALIDATE_INT)){  
    echo "<br>$var is Integer.";  
}else{  
    echo "<br>$var is not an Integer.";  
}  
?>
```

int(9950)  
9950 is Integer.

```
<?php

$var = "hello@yahoobaba.net";

//var_dump(filter_var($var, FILTER_VALIDATE_EMAIL));

if(filter_var($var, FILTER_VALIDATE_EMAIL)){
    echo "<br>$var is valid email.";
} else{
    echo "<br>$var is not an valid email.";
}
?>
```

```
<?php

$var = "https://www.yahoobaba.net/test/page.php";

//var_dump(filter_var($var, FILTER_VALIDATE_URL));

if(filter_var($var, FILTER_VALIDATE_URL)){
    echo "<br>$var is valid URL.";
} else{
    echo "<br>$var is not an valid URL.";
}
?>
```

https://www.yahoobaba.net/test/page.php  
is valid URL.



## PHP Filter Sanitization :

filter\_var(var, filtername, flag)



- FILTER\_SANITIZE\_EMAIL
- FILTER\_SANITIZE\_URL
- FILTER\_SANITIZE\_NUMBER\_INT
- FILTER\_SANITIZE\_NUMBER\_FLOAT
- FILTER\_SANITIZE\_MAGIC\_QUOTES
- FILTER\_SANITIZE\_STRING
- FILTER\_SANITIZE\_ENCODED
- FILTER\_SANITIZE\_SPECIAL\_CHARS

```
<?php

$var = "ram(.kumar@example.com";

$var = filter_var($var, FILTER_SANITIZE_EMAIL);

if(filter_var($var, FILTER_VALIDATE_EMAIL)){
    echo "<br>$var is valid Email.";
} else{
    echo "<br>$var is not an valid Email.";
}
```

```
<?php

$var = "https://www.yahoo baba.net";

$var = filter_var($var, FILTER_SANITIZE_URL);

if(filter_var($var, FILTER_VALIDATE_URL)){
    echo "<br>$var is valid URL.";
} else{
    echo "<br>$var is not an valid URL.";
}
```

## filter\_var(var, filtername, flag)

### Validate Filters

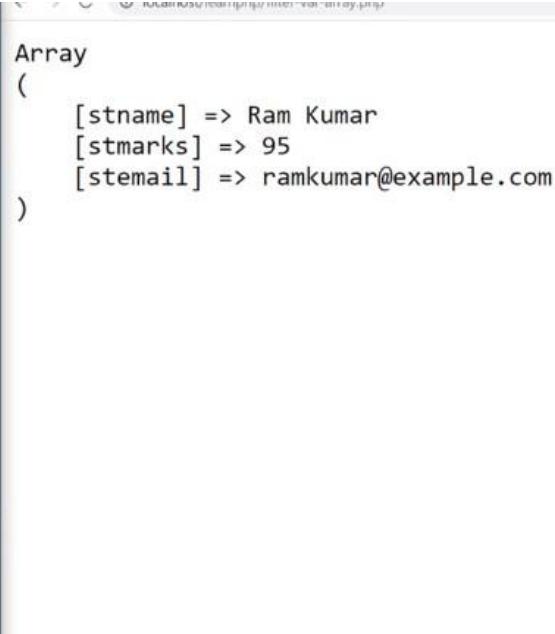
- FILTER\_VALIDATE\_INT
- FILTER\_VALIDATE\_FLOAT
- FILTER\_VALIDATE\_BOOLEAN
- FILTER\_VALIDATE\_EMAIL
- FILTER\_VALIDATE\_URL
- FILTER\_VALIDATE\_IP
- FILTER\_VALIDATE\_MAC
- FILTER\_VALIDATE\_REGEXP

### Sanitize Filters

- FILTER\_SANITIZE\_EMAIL
- FILTER\_SANITIZE\_URL
- FILTER\_SANITIZE\_NUMBER\_INT
- FILTER\_SANITIZE\_NUMBER\_FLOAT
- FILTER\_SANITIZE\_MAGIC\_QUOTES
- FILTER\_SANITIZE\_STRING
- FILTER\_SANITIZE\_ENCODED
- FILTER\_SANITIZE\_SPECIAL\_CHARS

## filter\_var\_array(data array , filter array)

```
<?php  
  
$arr = Array (  
    "stname" => "Ram Kumar",  
    "stmarks" => "95",  
    "stemail" => "ramkumar@example.com",  
);  
$filters = array(  
    "stname" => FILTER_SANITIZE_STRING,  
    "stmarks" => array(  
        "filter" => FILTER_VALIDATE_INT,  
        "options" => array(  
            "min_range" => 1,  
            "max_range" => 100  
        )  
    ),  
    "stemail" => FILTER_SANITIZE_EMAIL  
);  
  
20 echo "<pre>";  
21 print_r(filter_var_array($arr,$filters));  
22 echo "</pre>";
```



```
Array  
(  
    [stname] => Ram Kumar  
    [stmarks] => 95  
    [stemail] => ramkumar@example.com  
)
```

## PHP : Filter Input

`filter_input(type, variable, filter, options)`

- INPUT\_GET
- INPUT\_POST
- INPUT\_COOKIE
- INPUT\_SERVER
- INPUT\_ENV
- Validate Filters
- Sanitize Filters

```
<form method="get" action=<?php echo $_SERVER["PHP_SELF"]; ?>>
E-mail: <input type="text" name="email" autocomplete="off">

<input type="submit" name="submit" value="Submit">
</form>
<br>
<?php
if(isset($_REQUEST['submit'])){
    echo filter_input(INPUT_GET,"email", FILTER_VALIDATE_EMAIL);

}
?>
```

## Validation PHP

```
<?php
// Functions to filter user inputs
function filterName($field){
    // Sanitize user name
    $field = filter_var(trim($field), FILTER_SANITIZE_STRING);

    // Validate user name
    if(filter_var($field, FILTER_VALIDATE_REGEXP,
array("options"=>array("regexp"=>"/^[\a-zA-Z\s]+$/")))){
        return $field;
    } else{
        return FALSE;
    }
}
function filterEmail($field){
    // Sanitize e-mail address
    $field = filter_var(trim($field), FILTER_SANITIZE_EMAIL);

    // Validate e-mail address
    if(filter_var($field, FILTER_VALIDATE_EMAIL)){
        return $field;
    } else{
        return FALSE;
    }
}
function filterString($field){
    // Sanitize string
    $field = filter_var(trim($field), FILTER_SANITIZE_STRING);
    if(!empty($field)){
        return $field;
    } else{
        return FALSE;
    }
}

// Define variables and initialize with empty values
$nameErr = $emailErr = $messageErr = "";
$name = $email = $subject = $message = "";

// Processing form data when form is submitted
if($_SERVER["REQUEST_METHOD"] == "POST"){


```

```

// Validate user name
if(empty($_POST["name"])){
    $nameErr = "Please enter your name.";
} else{
    $name = filterName($_POST["name"]);
    if($name == FALSE){
        $nameErr = "Please enter a valid name.";
    }
}

// Validate email address
if(empty($_POST["email"])){
    $emailErr = "Please enter your email address.";
} else{
    $email = filterEmail($_POST["email"]);
    if($email == FALSE){
        $emailErr = "Please enter a valid email address.";
    }
}

// Validate message subject
if(empty($_POST["subject"])){
    $subject = "";
} else{
    $subject = filterString($_POST["subject"]);
}

// Validate user comment
if(empty($_POST["message"])){
    $messageErr = "Please enter your comment.";
} else{
    $message = filterString($_POST["message"]);
    if($message == FALSE){
        $messageErr = "Please enter a valid comment.";
    }
}

// Check input errors before sending email
if(empty($nameErr) && empty($emailErr) && empty($messageErr)){
    // Recipient email address
    $to = 'webmaster@example.com';

    // Create email headers
    $headers = 'From: ' . $email . "\r\n" .
    'Reply-To: ' . $email . "\r\n" .

```

```

'X-Mailer: PHP/' . phpversion();

    // Sending email
    if(mail($to, $subject, $message, $headers)){
        echo '<p class="success">Your message has been sent
successfully!</p>';
    } else{
        echo '<p class="error">Unable to send email. Please try again!</p>';
    }
}

?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Contact Form</title>
    <style type="text/css">
        .error{ color: red; }
        .success{ color: green; }
    </style>
</head>
<body>
    <h2>Contact Us</h2>
    <p>Please fill in this form and send us.</p>
    <form action="contact.php" method="post">
        <p>
            <label for="inputName">Name:<sup>*</sup></label>
            <input type="text" name="name" id="inputName" value=<?php echo
$name; ?>>
                <span class="error"><?php echo $nameErr; ?></span>
        </p>
        <p>
            <label for="inputEmail">Email:<sup>*</sup></label>
            <input type="text" name="email" id="inputEmail" value=<?php echo
$email; ?>>
                <span class="error"><?php echo $emailErr; ?></span>
        </p>
        <p>
            <label for="inputSubject">Subject:</label>
            <input type="text" name="subject" id="inputSubject" value=<?php echo
$subject; ?>>
        </p>
        <p>
            <label for="inputComment">Message:<sup>*</sup></label>

```

```

        <textarea name="message" id="inputComment" rows="5" cols="30"><?php
echo $message; ?></textarea>
        <span class="error"><?php echo $messageErr; ?></span>
    </p>
    <input type="submit" value="Send">
    <input type="reset" value="Reset">
</form>
</body>
</html>

```

## Another Example

```

<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>
<?php
/* define variables and set to empty values */
$nameErr = $emailError = $mobileError = "";
$name = $email = $mobile = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
        $nameErr = "Name is required";
    }
    else {
        $name = test_input($_POST["name"]);
        /* check if name only contains letters and whitespace */
        if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
            $nameErr = "Only letters and white space allowed";
        }
    }
    if (empty($_POST["name"])) {
        $mobileError = "Name is required";
    } else {
        $mobile = test_input($_POST["mobile"]);
        /* check if name only contains letters and whitespace */
        if (!preg_match('/^([0-9]{10})+$/', $mobile)) {
            $mobileError = "10 digit Number";
        }
    }
}

```

```

        }
        if (empty($_POST["email"])) {
            $emailError = "Email is required";
        } else {
            $email = test_input($_POST["email"]);
            /* check if e-mail address is well-formed */
            if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
                $emailError = "Invalid email format";
            }
        }
    }
function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
?>
<form method="post" action="php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?&gt;"&gt;
    Name: &lt;input type="text" name="name" value="<?php echo
$name;?&gt;"&gt;
    &lt;span class="error"&gt;* &lt;?php echo $nameErr;?&gt;&lt;/span&gt;
    &lt;br&gt;&lt;br&gt;
    Mobile: &lt;input type="text" name="mobile" value="<?php echo
$mobile;?&gt;"&gt;
    &lt;span class="error"&gt;* &lt;?php echo $mobileError;?&gt;&lt;/span&gt;
    &lt;br&gt;&lt;br&gt;
    E-mail: &lt;input type="text" name="email" value="<?php echo
$email;?&gt;"&gt;
    &lt;span class="error"&gt;* &lt;?php echo $emailError;?&gt;&lt;/span&gt;
    &lt;br&gt;&lt;br&gt;
    &lt;input type="submit" name="submit" value="Submit"&gt;
&lt;/form&gt;
&lt;/body&gt;
&lt;/html&gt;
</pre

```

## Login and signup with facebook

```
<!DOCTYPE html>
<html>
<title>HTML Tutorial</title>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<body>
<!-- Display login status --&gt;
&lt;div id="status"&gt;&lt;/div&gt;

<!-- Facebook login or logout button --&gt;
&lt;a href="javascript:void(0); " onclick="fbLogin(); " id="fbLink"&gt;&lt;img
src="images/fb-login-btn.png"/&gt;&lt;/a&gt;

<!-- Display user's profile info --&gt;
&lt;div class="ac-data" id="userData"&gt;&lt;/div&gt;
&lt;script&gt;
window.fbAsyncInit = function() {
    /* FB JavaScript SDK configuration and setup */
    FB.init({
        appId      : '264467978157588', /* FB App ID */
        cookie     : true, /* enable cookies to allow the server to access the
session */
        xfbml      : true, /* parse social plugins on this page */
        version    : 'v3.2' /* use graph api version 2.8 */
    });

    /* Check whether the user already logged in */
    FB.getLoginStatus(function(response) {
        if (response.status === 'connected') {
            /* display user data */
            getFbUserData();
        }
    });
};

/* Load the JavaScript SDK asynchronously */
(function(d, s, id) {
    var js, fjs = d.getElementsByTagName(s)[0];
    if (d.getElementById(id)) return;
    js = d.createElement(s); js.id = id;
    js.src = /* connect.facebook.net/en_US/sdk.js */;
    fjs.parentNode.insertBefore(js, fjs);
})()</pre>
```

```

}(document, 'script', 'facebook-jssdk'));

/* Facebook login with JavaScript SDK */
function fbLogin() {
    FB.login(function (response) {
        if (response.authResponse) {
            /* Get and display the user profile data */
            getFbUserData();
        } else {
            document.getElementById('status').innerHTML = 'User cancelled login or did not fully authorize.';
        }
    }, {scope: 'email'});
}

/* Fetch the user profile data from facebook */
function getFbUserData(){
    FB.api('/me', {locale: 'en_US', fields:
'id,first_name,last_name,email,link,gender,locale,picture'},
    function (response) {
        document.getElementById('fbLink').setAttribute("onclick","fbLogout()");
        document.getElementById('fbLink').innerHTML = 'Logout from Facebook';
        document.getElementById('status').innerHTML = '<p>Thanks for logging in,' +
        response.first_name + '!</p>';
        document.getElementById('userData').innerHTML = '<h2>Facebook Profile Details</h2><p></p><p><b>FB ID:</b> ' +
        response.id+'</p><p><b>Name:</b> ' +response.first_name+
        ' '+response.last_name+'</p><p><b>Email:</b> ' +response.email+'</p>';
    });
}

/* Logout from facebook */
function fbLogout() {
    FB.logout(function() {
        document.getElementById('fbLink').setAttribute("onclick","fbLogin()");
        document.getElementById('fbLink').innerHTML = '';
        document.getElementById('userData').innerHTML = '';
        document.getElementById('status').innerHTML = '<p>You have successfully logout from Facebook.</p>';
    });
}
</script>
</body>
</html>

```

# Regular Expression

Let's begin with a brief overview of the commonly used PHP's built-in pattern-matching functions before delving deep into the world of regular expressions.

Function	What it Does
<code>preg_match()</code>	Perform a regular expression match.
<code>preg_match_all()</code>	Perform a global regular expression match.
<code>preg_replace()</code>	Perform a regular expression search and replace.
<code>preg_grep()</code>	Returns the elements of the input array that matched the pattern.
<code>preg_split()</code>	Splits up a string into substrings using a regular expression.
<code>preg_quote()</code>	Quote regular expression characters found within a string.

## Regular Expression Syntax

Regular expression syntax includes the use of special characters (do not confuse with the [HTML special characters](#)). The characters that are given special meaning within a regular expression, are: `.` `*` `?` `+` `[` `]` `(` `)` `{` `}` `^` `$` `|` `\`. You will need to backslash these characters whenever you want to use them literally. For example, if you want to match `".`, you'd have to write `\.`. All other characters automatically assume their literal meanings.

The following sections describe the various options available for formulating patterns:

## Character Classes

Square brackets surrounding a pattern of characters are called a character class e.g. `[abc]`. A character class always matches a single character out of a list of specified characters that means the expression `[abc]` matches only a, b or c character.

Negated character classes can also be defined that match any character except those contained within the brackets. A negated character class is defined by placing a caret (`^`) symbol immediately after the opening bracket, like this `[^abc]`.

You can also define a range of characters by using the hyphen (`-`) character inside a character class, like `[0-9]`. Let's look at some examples of character classes:

RegExp	What it Does
[abc]	Matches any one of the characters a, b, or c.
[^abc]	Matches any one character other than a, b, or c.
[a-z]	Matches any one character from lowercase a to lowercase z.
[A-Z]	Matches any one character from uppercase a to uppercase z.
[a-Z]	Matches any one character from lowercase a to uppercase Z.
[0-9]	Matches a single digit between 0 and 9.
[a-z0-9]	Matches a single character between a and z or between 0 and 9.

The following example will show you how to find whether a pattern exists in a string or not using the regular expression and PHP `preg_match()` function:

**Example**

[Run this code »](#)

```

1 <?php
2 $pattern = "/ca[kf]e/";
3 $text = "He was eating cake in the cafe.";
4 if(preg_match($pattern, $text)){
5     echo "Match found!";
6 } else{
7     echo "Match not found.";
8 }
9 ?>

```

Similarly, you can use the `preg_match_all()` function to find all matches within a string:

**Example**

[Run this code »](#)

```

1 <?php
2 $pattern = "/ca[kf]e/";
3 $text = "He was eating cake in the cafe.";
4 $matches = preg_match_all($pattern, $text, $array);
5 echo $matches . " matches were found.";
6 ?>

```

## Predefined Character Classes

Some character classes such as digits, letters, and whitespaces are used so frequently that there are shortcut names for them. The following table lists those predefined character classes:

Shortcut	What it Does
.	Matches any single character except newline <code>\n</code> .
<code>\d</code>	matches any digit character. Same as <code>[0-9]</code>
<code>\D</code>	Matches any non-digit character. Same as <code>[^0-9]</code>
<code>\s</code>	Matches any whitespace character (space, tab, newline or carriage return character). Same as <code>[\t\n\r]</code>
<code>\S</code>	Matches any non-whitespace character. Same as <code>[^ \t\n\r]</code>
<code>\w</code>	Matches any word character (definined as a to z, A to Z, 0 to 9, and the underscore). Same as <code>[a-zA-Z_0-9]</code>
<code>\W</code>	Matches any non-word character. Same as <code>[^a-zA-Z_0-9]</code>

The following example will show you how to find and replace space with a hyphen character in a string using regular expression and PHP `preg_replace()` function:

Example	<a href="#">Run this code »</a>
<pre>1 &lt;?php 2 \$pattern = "/\s/"; 3 \$replacement = "-"; 4 \$text = "Earth revolves around\nthe\tSun"; 5 // Replace spaces, newlines and tabs 6 echo preg_replace(\$pattern, \$replacement, \$text); 7 echo "&lt;br&gt;"; 8 // Replace only spaces 9 echo str_replace(" ", "-", \$text); 10 ?&gt;</pre>	

## Repetition Quantifiers

In the previous section we've learnt how to match a single character in a variety of fashions. But what if you want to match on more than one character? For example, let's say you want to find out words containing one or more instances of the letter p, or words containing at least two p's, and so on. This is where quantifiers come into play. With quantifiers you can specify how many times a character in a regular expression should match.

The following table lists the various ways to quantify a particular pattern:

RegExp	What it Does
p+	Matches one or more occurrences of the letter p.
p*	Matches zero or more occurrences of the letter p.
p?	Matches zero or one occurrences of the letter p.
p{2}	Matches exactly two occurrences of the letter p.
p{2,3}	Matches at least two occurrences of the letter p, but not more than three occurrences of the letter p.
p{2,}	Matches two or more occurrences of the letter p.
p{,3}	Matches at most three occurrences of the letter p

The regular expression in the following example will splits the string at comma, sequence of commas, whitespace, or combination thereof using the PHP `preg_split()` function:

Example	Run this code »
<pre>1 &lt;?php 2 \$pattern = "/[\s,]+/"; 3 \$text = "My favourite colors are red, green and blue"; 4 \$parts = preg_split(\$pattern, \$text); 5 6 // Loop through parts array and display substrings 7 foreach(\$parts as \$part){ 8     echo \$part . "&lt;br&gt;"; 9 } 10 ?&gt;</pre>	

## Position Anchors

There are certain situations where you want to match at the beginning or end of a line, word, or string. To do this you can use anchors. Two common anchors are caret ( ^ ) which represent the start of the string, and the dollar ( \$ ) sign which represent the end of the string.

RegExp	What it Does
^p	Matches the letter p at the beginning of a line.
p\$	Matches the letter p at the end of a line.

The regular expression in the following example will display only those names from the names array which start with the letter "J" using the PHP `preg_grep()` function:

### Example

[Run this code »](#)

```
1 <?php
2 $pattern = "/^J/";
3 $names = array("Jhon Carter", "Clark Kent", "John Rambo");
4 $matches = preg_grep($pattern, $names);
5
6 // Loop through matches array and display matched names
7 foreach($matches as $match){
8     echo $match . "<br>";
9 }
10 ?>
```

## Pattern Modifiers

A pattern modifier allows you to control the way a pattern match is handled. Pattern modifiers are placed directly after the regular expression, for example, if you want to search for a pattern in a case-insensitive manner, you can use the `i` modifier, like this: `/pattern/i`. The following table lists some of the most commonly used pattern modifiers.

Modifier	What it Does
<code>i</code>	Makes the match case-insensitive manner.
<code>m</code>	Changes the behavior of <code>^</code> and <code>\$</code> to match against a newline boundary (i.e. start or end of each line within a multiline string), instead of a string boundary.
<code>g</code>	Perform a global match i.e. finds all occurrences.
<code>o</code>	Evaluates the expression only once.
<code>s</code>	Changes the behavior of <code>.</code> (dot) to match all characters, including newlines.
<code>x</code>	Allows you to use whitespace and comments within a regular expression for clarity.

The following example will show you how to perform a global case-insensitive search using the `i` modifier and the PHP `preg_match_all()` function.

```
1 <?php
2 $pattern = "/color/i";
3 $text = "Color red is more visible than color blue in daylight.";
4 $matches = preg_match_all($pattern, $text, $array);
5 echo $matches . " matches were found.";
6 ?>
```

Similarly, the following example shows how to match at the beginning of every line in a multi-line string using `^` anchor and `m` modifier with PHP `preg_match_all()` function.

Example	Run this code »
<pre>1 &lt;?php 2 \$pattern = "/^color/im"; 3 \$text = "Color red is more visible than \ncolor blue in daylight."; 4 \$matches = preg_match_all(\$pattern, \$text, \$array); 5 echo \$matches . " matches were found."; 6 ?&gt;</pre>	

## Word Boundaries

A word boundary character (`\b`) helps you search for the words that begins and/or ends with a pattern. For example, the regexp `/\bcar/` matches the words beginning with the pattern car, and would match cart, carrot, or cartoon, but would not match oscar.

Similarly, the regexp `/car\b/` matches the words ending with the pattern car, and would match scar, oscar, or supercar, but would not match cart. Likewise, the `/\bcar\b/` matches the words beginning and ending with the pattern car, and would match only the word car.

The following example will highlight the words beginning with car in bold:

Example	Run this code »
<pre>1 &lt;?php 2 \$pattern = '/\bcar\b*/'; 3 \$replacement = '&lt;b&gt;\$0&lt;/b&gt;'; 4 \$text = 'Words beginning with car: cart, carrot, cartoon. Words ending with car: 5 scar, oscar, supercar.'; 6 echo preg_replace(\$pattern, \$replacement, \$text); 7 ?&gt;</pre>	

```
preg_match_all("/php|web|the/i", $string, $array);
```

- `preg_match()`
- `preg_match_all()`

```
preg_replace(pattern, replacement, string, limit)
```

```
preg_match(pattern, string, array)      preg_split(pattern, string, limit, flags)
```

↓

*/pattern/modifiers;*

```
preg_quote(string, delimiter)
```

### Expressions : Assertion

<code>?=n</code>	Positive Lookahead
<code>?!n</code>	Negative Lookahead
<code>?&lt;=</code>	Positive Lookbehind
<code>?!=</code>	Negative Lookbehind
<code>?&lt;!</code>	Negative Lookbehind

<code>*n</code>	Greedy Match
<code>+n</code>	Greedy Match
<code>*?n</code>	Lazy Match
<code>+?n</code>	Lazy Match

The image shows two windows of the Notepad++ text editor. The top window displays the file `index.php` with the following code:

```
<?php
if(isset($_POST['submit'])) {
    echo getRevStr($_POST['str']);
}
function getRevStr($str) {
    $revstr="";
    $count=strlen($str)-1;
    for($i=$count;$i>=0;$i--) {
        $revstr.=$str[$i];
    }
    return $revstr;
?
<form method="post">
    <input type="textbox" name="str" required/>
    <input type="submit" name="submit"/>
```

The bottom window displays the file `reverse_string.php` with the following code:

```
<?php
$str="Vishal";
echo strrev($str);
?
```

D:\wamp\www\iphp\hack\index.php - Notepad+

```

1 <?php
2 if(isset($_POST['submit'])){
3     $con=mysqli_connect('localhost','root','','youtube');
4     $name=mysqli_real_escape_string($con,$_POST['name']);
5     $password=mysqli_real_escape_string($con,$_POST['password']);
6     echo $sql="select * from user where name='".$name."' and password='".$password."'";
7     echo "<br>";
8     $res=mysqli_query($con,$sql);
9     if(mysqli_num_rows($res)>0){
10         echo "Working";
11     }else{
12         echo "Please enter correct details";
13     }
14 }
?>
15 <form method="post">
16     <table>
17         <tr>
18             <td>Name</td>
19             <td><input type="text" name="name"></td>
20         </tr>
21         <tr>

```

PHP Hypertext Preprocessor file

D:\xampp\htdocs\jquery\_sound\index.php - Notepad+

```

1 <audio id="audioBox">
2     <source src="notify.mp3" type="audio/mpeg"/>
3     <source src="notify.wav" type="audio/wav"/>
4     <source src="notify.ogg" type="audio/ogg"/>
5 </audio>
6 <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
7 <script>
8 setInterval(function(){
9     load();
10 },3000);
11 function load(){
12     jQuery.ajax({
13         url:'get.php',
14         success:function(result){
15             var data=jQuery.parseJSON(result);
16             if(data.sound=="yes"){
17                 jQuery('#audioBox')[0].play();
18             }
19         }
20     });
21 }

```

PHP Hypertext Preprocessor file

\*D:\xampp\htdocs\php\pdf\index.php - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

index.php

```

1 <?php
2 require('vendor/autoload.php');
3 $con=mysqli_connect('localhost','root','','youtube');
4 $res=mysqli_query($con,"select * from user");
5 if(mysqli_num_rows($res)>0){
6     $html='<style></style><table class="table">';
7     $html.= '<tr><td>ID</td><td>Name</td><td>Email</td></tr>';
8     while($row=mysqli_fetch_assoc($res)){
9         $html.= '<tr><td>' . $row['id'] . '</td><td>' . $row['name'] . '</td><td>' . $row[
10            'email'] . '</td></tr>';
11    }
12    $html.= '</table>';
13 }else{
14     $html="Data not found";
15 }
16 $mpdf=new \Mpdf\Mpdf();
17 $mpdf->WriteHTML($html);
18 $file=time().'.pdf';
19 $mpdf->output($file,'I');
?>

```

PHP Hypertext Preprocessor file

Type here to search

127.0.0.1/php/file\_handling/

Vishal Gupta  
Amit Gupta

D:\xampp\htdocs\php\file\_handling\index.php - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

index.php

```

1 <?php
2 //touch('test.txt');
3 //unlink('test.txt');
4 //mkdir('test');
5 //rmdir('test');
6 //copy("test.txt","abc.txt");
7
8 /*$f=fopen("text.txt","a");
9  fwrite($f,"Vishal Gupta");
10  fwrite($f,"\nAmit Gupta");
11  fclose($f);*/
12
13 /*$f=fopen("text.txt","r");
14  while(!feof($f)){
15      echo fgets($f);
16      echo "<br/>";
17  }*/
18
?>

```

file\_handling

Name Date

index 8/14

text 8/14

PHP Hypertext Preproc length: 314 lines: 19 Ln:18 Col:1 Sel:0|0 Windows (CR LF) UTF-8 INS

Type here to search

8:54 AM 8/14/2020

The screenshot shows two code editors side-by-side in Notepad++.

**File 1: face-face.html**

```
1 <html>
2   <head>
3     <title>@Font-face</title>
4     <style>
5       @font-face{
6         font-family:openSans;
7         src: url(fonts/OpenSans-Regular-webfont.eot),
8              |url(fonts/OpenSans-Regular-webfont.woff);
9       }
10
11      #box{
12        font-family: openSans;
13      }
14    </style>
15  </head>
16
17  <body>
18    <h1>Yahoo Baba : CSS @font-face Rule</h1>
19
20    <div id="box">
21      Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc elit dolor, ornare in tempus vel, cursus
22    </div>
23
24  </body>
25</html>
```

**File 2: create-cookie.php**

```
1 <?php
2 $cookie_name = "user";
3 $cookie_value = "Yahoo Baba";
4
5 setcookie($cookie_name,$cookie_value, time() + (86400 * 30), "/");
6
7 ?>
8 <html>
9   <body>
10      <?php
11        echo $_COOKIE[$cookie_name];
12
13      ?>
14
```

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** On the left, it lists files: db.php, index.php (which is the active file), and details.php.
- Code Editor:** The main area contains the following PHP code:

```
8 if(isset($_POST['submit'])){  
9     $str=mysqli_real_escape_string($con,$_POST['str']);  
10    $sql="select id, title, details from news where title  
11        like '%$str%' or details like '%$str%'  
12        UNION  
13        select id, content as title, details from page where content  
14        like '%$str%' or content like '%$str%'";  
15    $res=mysqli_query($con,$sql);  
16    if(mysqli_num_rows($res)>0){  
17        while($row=mysqli_fetch_assoc($res)){  
18            echo $row['title'];  
19            echo "<br/>";  
20        }  
21    }else{  
22        echo "No data found";  
23    }  
}
```
- Taskbar:** At the bottom, there's a taskbar with various icons for file operations and system status.
- System Tray:** On the far right, it shows the date (26-12-2020), time (10:02), and language (ENG).

- ✓ Speed, Security and SEO optimized
- ✓ Optimized image slider, Popup and CTAs for high-conversion
- ✓ Email Marketing Platforms connection
- ✓ Page & Browser Cache
- ✓ Optimize Core Web Vitals (CLS, LCP and FCP)
- ✓ Fonts Preloading
- ✓ Cdn Setup and Fix Insecure Content
- ✓ Deferring Unused JS/CSS
- ✓ Inline & Combine JS/CSS
- ✓ Font Optimization
- ✓ Cache Preloading
- ✓ Minifying JS/CSS/HTML
- ✓ Lazy Loading Images and iFrame
- ✓ Serving Scaled Images
- ✓ Image Optimization
- ✓ WooCommerce Optimization
- ✓ Database Optimization
- ✓ Gzip File Compression

## Type Casting in PHP

### **PHP Type Casting:**

Converting one type of data value to another type is called as type casting.  
where: data value can be a literal, variable name or constant name

### **There are 2 types of type casting:**

Implicit type casting (Coercion): Done by PHP Engine

Explicit type casting (Conversion): Done by programmers

### **Implicit type casting:**

If required PHP engine automatically converts one type of data value to another type; which is known as implicit type casting.

**Ex:**

```
echo 2 + 4.3;      // 2.0 + 4.3 = 6.3  
echo "<br/>";
```

---

### **Implicit type casting:**

If required PHP engine automatically converts one type of data value to another type; which is known as implicit type casting.

**Ex:**

```
echo 2 + 4.3;      // 2.0 + 4.3 = 6.3|  
echo "<br/>";
```

```
echo "22manjunath"+22;    // 22 + 22 = 44  
echo "<br/>";
```

```
echo "manjunath22"+22;    // 0 + 22 = 22  
echo "<br/>";
```

```
echo 2 * "4a";          // 2 * 4 = 8  
echo "<br/>";  
echo 2 * "a2";          // 2 * 0 = 0  
echo "<br/>";
```

---

## Type Casting in PHP

### **Explicit type casting:**

If required, programmers can also convert one type of data value to another type; which is known as explicit type casting.

I

#### **Syntax:**

**(data type keyword)** data value;

where:

data value can be a literal, variable name, or constant name

#### **Name of the data type keywords for converting data values:**

(integer), (int) – converts a given data value to an integer

(double), (float), (real) - converts a given data value to a double

(boolean), (bool) - converts a given data value to a boolean

(string) - converts a given data value to a string

(array) - converts a given data value to an array

(object) - converts a given data value to an object

---

#### **Name of the data type keywords for converting data values:**

(integer), (int) – converts a given data value to an integer

(double), (float), (real) - converts a given data value to a double

(boolean), (bool) - converts a given data value to a boolean

(string) - converts a given data value to a string

(array) - converts a given data value to an array

(object) - converts a given data value to an object

(unset) - converts a given data value to NULL

#### **Example Code:**

```
echo (int) 3.142 , "<br/>"; // 3
echo (double) "3.142" , "<br/>"; // 3.142
echo (bool) 10 , "<br/>"; // 1
echo (string) 3.142, "<br/>"; // "3.142"
```

---

## Type Casting

Type casting converts the value to a chosen type by writing the type within parentheses before the value to convert.

```
<?php
$foo = 10; // $foo is an integer
$bar = (bool) $foo; // $bar is a boolean
?>
```

The casts allowed are:

- (int) - cast to int
- (bool) - cast to bool
- (float) - cast to float
- (string) - cast to string
- (array) - cast to array
- (object) - cast to object
- (unset) - cast to NULL

**Note:**

(integer) is an alias of the (int) cast. (boolean) is an alias of the (bool) cast. (binary) is an alias of the (string) cast. (double) and (real) are aliases of the (float) cast. These casts do not use the canonical type name and are not recommended.

**Warning** The (real) cast alias has been deprecated as of PHP 8.0.0.



## What is Type Hinting ?

```
function sum(int $value){
    echo $value + 10;
}

sum(10);
sum("Hey");
```

### Type declaration

### Valid DataTypes

- Int
- Float
- String
- Bool
- Array
- Class / Interface Name
- Object

```
6 |<h2>Type Juggling</h2>
7 |<?php
8 |    $count = "3";
9 |?>
10|Type: <?php echo gettype($count); ?><br />
11|<?php $count += 3 ?>
12|Type: <?php echo gettype($count); ?><br />
13|<h2>Type Casting</h2>
14|<?php settype($count,"string"); ?><br />
15|Type: <?php echo gettype($count); ?><br />
16|<?php $test1 = 54; ?>
17|Type of Test 1: <?php echo gettype($test1); ?><br />
18|<?php $test2 = (string)$test1; ?>
19|Type of Test 1: <?php echo gettype($test1); ?><br />
20|Type of Test 2: <?php echo gettype($test2); ?><br />
```

```
File Edit Selection Find View Goto Tools Project Preferences Help
function_multiplereturn.php x
7
8 <?php
9
10 function add_sub($val1, $val2) {
11     $add = $val1 + $val2;
12     $sub = $val1 - $val2;
13     return array($add, $sub);
14 }
15
16 $result_array = add_sub(7,3);
17 echo "Add: " . $result_array[0] . "<br />";
18 echo "Sub: " . $result_array[1] . "<br />";
19
20 list($addition, $subtraction) = add_sub(17,8);
21 echo "Add: " . $addition . "<br />";
22 echo "Sub: " . $subtraction . "<br />";
23
24 ?>
25
26
27
28 RECORDED WITH
29 SCREENCASTOMATIC
30 </html>
```

```
File Edit Selection Find View Goto Tools Project Preferences Help
function_multiplereturn.php *
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Function: Multiple returns</title>
5 </head>
6 <body>
7
8     <?php
9
10    function add_sub($val1, $val2) {
11        $add = $val1 + $val2;
12        $sub = $val1 - $val2;
13        return array($add, $sub);
14    }
15
16    $result_array = add_sub(7,3);
17    echo "Add: " . $result_array[0] . "<br />";
18    echo "Sub: " . $result_array[1] . "<br />";
19
20 ?>
21
22
23 RECORDED WITH
24 SCREENCASTOMATIC
25 </body>
```

## Types of Cookies

S.No	Name	Definition
1.	<b>Session Cookie</b>	This type of cookies are temporary and are expire as soon as the session ends or the browser is closed
2.	<b>Persistent Cookie</b>	Persistent cookies are written on hard disk. Persistent cookies do not expire at the end of the session



## Cookies function in PHP

Name of Function/Variable	Use
<code>setcookie()</code>	To create, modify and delete a cookie
<code>isset()</code>	To check if a cookie is set or not
<code>\$_COOKIE variable</code>	retrieve the value of the cookie

## What is cookies ?

- A cookie is a small text file used to store small piece of data on the user's computer by the web browser while browsing a website. It contains username, password and other information.

# Differentiate between PHP5 and PHP7.

## 1. Performance:

- **PHP 5:** PHP 5 had performance limitations and was relatively slower compared to later versions.
- **PHP 7:** PHP 7 introduced significant performance improvements. It included the Zend Engine 3.0, which made PHP 7 up to twice as fast as PHP 5.6 for many applications. This improvement was achieved through better memory usage and optimizations in the core engine.

## 2. Scalar Type Declarations:

- **PHP 5:** PHP 5 allowed type hinting for class/interface types, but it didn't support scalar type declarations (int, float, string, bool).
- **PHP 7:** PHP 7 introduced scalar type declarations, which allow you to specify the expected data types for function parameters and return values. You can use `int`, `float`, `string`, and `bool` as type hints.

## 3. Return Type Declarations:

- **PHP 5:** PHP 5 did not support return type declarations for functions.
- **PHP 7:** PHP 7 introduced return type declarations, which allow you to specify the expected data type that a function should return. You can use : `returnType` after the function declaration.

## 4. Spaceship Operator (<=>):

- **PHP 5:** The spaceship operator (<=>) was not available in PHP 5.
- **PHP 7:** PHP 7 introduced the spaceship operator for comparing two values, making it easier to implement custom sorting and comparison functions.

## 5. Null Coalescing Operator (??):

- **PHP 5:** The null coalescing operator (??) was not available in PHP 5.
- **PHP 7:** PHP 7 introduced the null coalescing operator, which provides a concise way to handle potentially null values.

## 6. Anonymous Classes:

- **PHP 5:** PHP 5 did not support anonymous classes.
- **PHP 7:** PHP 7 introduced support for anonymous classes, allowing you to define classes on the fly without explicitly naming them.

## 7. Error Handling:

- **PHP 5:** PHP 5 used traditional error handling with `mysql_error()`, `mysql_errno()`, and `die()`.
- **PHP 7:** PHP 7 introduced a more consistent and improved error handling mechanism using exceptions. It also introduced the Throwable interface for better exception handling.

## 8. Improved Type Hinting:

- **PHP 5:** PHP 5 had limited support for type hinting, primarily for class and interface types.
- **PHP 7:** PHP 7 expanded type hinting capabilities, including scalar type hints, return type hints, and improved support for custom types through interfaces and classes.

## 9. Deprecated Features:

- **PHP 5:** PHP 5 included many features and functions that were deprecated and removed in later versions.
- **PHP 7:** PHP 7 removed several deprecated features and improved the consistency of the language.

## 10. Unicode Support:

- **PHP 5:** PHP 5 had limited support for Unicode, making handling multibyte character encodings challenging.
- **PHP 7:** PHP 7 improved Unicode support and introduced functions for working with multibyte strings more effectively.

## 11. New Operators and Functions:

- **PHP 5:** PHP 5 introduced some operators and functions, but the additions were more significant in PHP 7.

## 12. Compatibility:

- **PHP 5:** PHP 5 had a substantial user base and many existing applications running on it.
- **PHP 7:** PHP 7 aimed to maintain backward compatibility with PHP 5 while introducing new features. However, some older code might require modifications to run on PHP 7.

# Types of Error in php

## 1. Parse Errors:

- Parse errors occur during the parsing phase of PHP script execution, often due to syntax errors in the code.
- These errors prevent the script from running and typically include messages like "syntax error, unexpected token."

## 2. Fatal Errors:

- Fatal errors are severe issues that cause the script to terminate abruptly.
- Examples include calling an undefined function or accessing an undefined class.
- Fatal errors cannot be caught or handled within the script.

## 3. Warnings:

- Warnings are non-fatal errors that don't halt script execution but indicate potential issues.

- For example, using an uninitialized variable or including a file that doesn't exist.

#### 4. Notices:

- Notices are less severe than warnings and are often related to variables that are used **without being explicitly initialized**.
- While notices don't halt script execution, they should be addressed to ensure correct behavior.

#### 5. Errors Triggered by User-defined Functions:

- These errors can occur within user-defined functions or methods.
- Developers can use the `trigger_error()` function to generate custom errors and exceptions within their code.

#### 6. Exceptions:

- Exceptions are a more advanced error-handling mechanism introduced in PHP 5.
- They allow for structured error handling using `try`, `catch`, `finally`, and `throw` statements.
- Exceptions can be categorized into two main types:
  - **Built-in Exceptions:** PHP provides several built-in exception classes, such as `Exception`, `InvalidArgumentException`, `RuntimeException`, etc.
  - **Custom Exceptions:** Developers can define their own custom exception classes to handle specific application-level errors.

#### 7. Deprecated Features:

- Deprecated features are functions, classes, or methods that are marked as deprecated and may be removed in future PHP versions.
- Using deprecated features generates warnings to encourage developers to update their code.

#### 8. Compile-time Errors:

- Compile-time errors occur during the compilation phase before script execution.
- These errors include issues like referencing a **non-existent class** or extending a final class.

#### 9. Execution-time Errors:

- Execution-time errors occur during script execution and may be triggered by various runtime conditions.
- Examples include division by zero, accessing an undefined array index, or exceeding memory limits.

#### 10. Assertion Errors:

- Assertions are used to test conditions during development and are typically disabled in production.
- When an assertion fails, an `AssertionError` is thrown, indicating a logical error in the code.

#### 11. Database Errors:

- These errors occur when interacting with a database using database extensions like MySQLi or PDO.
- Examples include connection errors, query errors, and data retrieval errors.

## 12. HTTP Errors:

- HTTP errors are not PHP-specific but may be relevant when dealing with web applications.
- Examples include HTTP status codes like 404 (Not Found) or 500 (Internal Server Error).

# LAMP Setup

Setting up a LAMP (Linux, Apache, MySQL, PHP) stack on a server involves installing and configuring the necessary components to host web applications or websites. Here are the steps to set up a LAMP stack on a Linux server:

### 1. Provision a Linux Server:

1. Choose a Linux distribution such as Ubuntu, CentOS, or Debian for your server.
2. Rent a virtual private server (VPS) from a hosting provider or set up a physical server, if applicable.

### 2. Connect to the Server:

Access your server via SSH using a terminal or an SSH client. Replace `your_server_ip` with the actual IP address of your server.

css

Copy code

```
ssh username@your_server_ip
```

### 3. Update the System:

Before installing any software, update the package list and upgrade installed packages to ensure you have the latest security updates.

sql

Copy code

```
sudo apt update sudo apt upgrade
```

### 4. Install Apache Web Server:

Install the Apache web server using the package manager for your Linux distribution. For example, on Ubuntu:

Copy code  
sudo apt install apache2

## 5. Start and Enable Apache:

Start the Apache service and enable it to start automatically on boot.

Copy code  
bash  
sudo systemctl start apache2 sudo systemctl enable apache2

## 6. Test Apache:

Open a web browser and enter your server's IP address. You should see the default Apache test page, indicating that Apache is running.

## 7. Install MySQL (or MariaDB):

Install the MySQL or MariaDB database server. MariaDB is a drop-in replacement for MySQL and is often used in modern LAMP setups.

Copy code  
sudo apt install mariadb-server

## 8. Secure MySQL (Optional but Highly Recommended):

Run the MySQL secure installation script to set a root password and improve security.

Copy code  
sudo mysql\_secure\_installation

## 9. Install PHP:

Install PHP and necessary PHP modules for your web applications.

Copy code  
sudo apt install php php-mysql

## 10. Test PHP:

Create a PHP info file to test PHP installation. Create a file named `info.php` in the web server's document root (usually `/var/www/html`) with the following content:

Copy code  
php  
<?php phpinfo(); ?>

Access `http://your_server_ip/info.php` in a web browser to see PHP information.

## **11. Configure Apache for PHP:**

To enable Apache to process PHP files, you may need to configure Apache to use the PHP module. On Ubuntu, it's usually enabled by default. On other distributions, you might need to enable it manually.

## **12. Create and Host Your Web Applications:**

Upload your web application files to the appropriate directory (usually `/var/www/html`) and configure Apache virtual hosts if needed.

## **13. Secure Your Server:**

1. Implement firewall rules (e.g., using UFW or iptables) to restrict access to your server.
2. Keep software up to date, apply security patches, and follow best practices for server security.

## **14. Optional: Install phpMyAdmin (Database Management Tool):**

If you need a web-based interface for managing MySQL/MariaDB databases, you can install phpMyAdmin. Be sure to secure it properly.

That's it! You now have a basic LAMP stack set up on your server. You can expand upon this setup by installing additional PHP modules, configuring Apache virtual hosts, and securing your web applications as needed.

C:\xampp\htdocs\test\demo.php - Sublime Text (UNREGISTERED)

```
1 <?php
2 $db=new mysqli("localhost","root","","testdemo");
3 if(isset($_POST['submit'])){
4 $check=implode(' ', $_POST['ch']);
5 $country=$_POST['country'];
6 $qry=$db->query("INSERT INTO `test`(`checkbox`, `country`) VALUES ('$check','$country')");
7 if($qry>0){
8     echo "<script> alert('Data is Submitted');</script>";
9 }else{
10    echo "<script> alert('Data is not Submitted');</script>";
11 }
12 }
13 ?>
14 <!DOCTYPE html>
15 <html>
16 <head>
17     <title>Demo</title>
18 </head>
19 <body>
20 <form action="" method="POST">
21     <input type="checkbox" name="ch[]" value="PHP">PHP<br>
22     <input type="checkbox" name="ch[]" value="JAVA">JAVA<br>
23     <input type="checkbox" name="ch[]" value="CSS">CSS<br>
24     <input type="checkbox" name="ch[]" value="HTML">HTML<br>
25     <input type="checkbox" name="ch[]" value="CORE PHP">CORE PHP<br>
26     <select name="country">
```

Line 4, Column 35

Tab Size: 4

PHP

12:04 PM 2/4/2018

```
<?php
$con=mysqli_connect("localhost","root","","complete_php");
//$sql="insert into student(name,city) values('Vishal','Delhi')";
//$sql="update student set name='Amit' where id=1";
//$sql="delete from student where id=1";
$sql="select * from student";
$res=mysqli_query($con,$sql);
while($row=mysqli_fetch_assoc($res)){
    echo '<pre>';
    print_r($row);
}
?>
```

## Select Insert

Reference Cyber variye youtube chanel

[https://www.youtube.com/watch?v=UNFdrhu\\_5GU&list=PLEjKBGxF74J67G7KSGHA4JPOKW9H3cQq3&index=9](https://www.youtube.com/watch?v=UNFdrhu_5GU&list=PLEjKBGxF74J67G7KSGHA4JPOKW9H3cQq3&index=9)

```
<?php
    if($_POST['register'])
    {
        $fname = $_POST['fname'];
        $lname = $_POST['lname'];
        $pwd = $_POST['password'];
        $cpwd = $_POST['conpassword'];
        $gender = $_POST['gender'];
        $email = $_POST['email'];
        $phone = $_POST['phone'];
        $address = $_POST['address'];

        $query = "INSERT INTO FORM VALUES('$fname','$lname','$pwd','$cpwd','$gender','$email','$phone','$address')";
        $data = mysqli_query($conn,$query);

        if($data)
        {
            echo "Data Inserted into Database";
        }
        else
        {
            echo "Failed";
        }
    }
?>
```

```
<div class="input_field">
    <label>Gender</label>
    <div class="custom_select">
        <select name="gender">
            <option value="Not Selected">Select</option>
            <option value="male">Male</option>
            <option value="female">Female</option>
        </select>
    </div>
</div>
```

```
if($total != 0)
{
    while($result = mysqli_fetch_assoc($data))
    {
        echo $result[fname]." ".$result[lname]." ".$result[gender]." ".$
            result[email]." ".$result[phone]." ".$result[address]."<br>";
    }
}
else
{
    echo "No records found";
}
```

```
<?php
while($result = mysqli_fetch_assoc($data))
{
    echo "<tr>
        <td>".$result[fname]."</td>
        <td>".$result[ lname]."</td>
        <td>".$result[ gender]."</td>
        <td>".$result[ email]."</td>
        <td>".$result[ phone]."</td>
        <td>".$result[ address]."</td>
    </tr>
    ";
}
```

```
<div class="input_field">
    <label>Address</label>
    <textarea class="textarea" name="address" required>
        <?php echo $result['address']; ?>
    </textarea>
</div>
```

Updated

```
<select name="gender" required>
    <option value="">Select</option>

    <option value="male"
        <?php
            if($result['gender'] == 'male')
            {
                echo "selected";
            }
        ?>
    >

        Male</option>
    <option value="female"
        <?php
            if($result['gender'] == 'female')
            {
                echo "selected";
            }
        ?>
    >

        Female</option>
</select>
```

```
$email = $_POST['email'];
$phone = $_POST['phone'];
$address = $_POST['address'];

$query = "UPDATE form2 set fname='$fname',lname='$lname',password
        ='$pwd',cpassword='$cpwd',gender='$gender',email='$email',
        phone='$phone',address='$address' WHERE id='$id'";

$data = mysqli_query($conn,$query);

if($data)
{
    echo "Record Updated";
}
else
{
    echo "Failed to Update";
}
```

```
$query = "UPDATE form2 set fname='$fname',lname='$lname',password='$
        pwd',cpassword='$cpwd',gender='$gender',email='$email',phone='$
        phone',address='$address' WHERE id='$id'";

$data = mysqli_query($conn,$query);

if($data)
{
    echo "<script>alert('Record Updated')</script>";
    ?>
    <meta http-equiv = "refresh" content = "0; url =
    http://localhost:8080/crud/display.php" />
    <?php
}
else
{
    echo "Failed to Update";
}
```

```
        <a href='delete.php?id=$result[id]'><input type='submit'  
            value='Delete' class='delete' onclick = 'return  
            checkdelete()'></a></td>  
    </tr>  
    "  
}
```

```
<script>  
    function checkdelete()  
    {  
        return confirm('Are you sure your want to delete this record ?');  
    }  
</script>
```

```
$id = $_GET['id'];  
  
$query = "DELETE FROM FORM2 WHERE id = '$id' ";  
$data = mysqli_query($conn,$query);  
  
if($data)  
{  
    echo "<script>alert('Record Deleted')</script>";  
    ?>  
    <meta http-equiv = "refresh" content = "0; url = http://localhost:8080  
    /crud/display.php" />  
    <?php  
}  
else  
{  
    echo "<script>alert('Failed To Deleted')</script>";  
}  
?>
```

## Redio

Phone Number

Caste

General  OBC  SC  ST

Language

Hindi  Urdu  English

Address

```
<div class="input_field">
    <label>Phone Number</label>
    <input type="text" class="input" name="phone" required>
</div>

<div class="input_field">
    <label style="margin-right: 100px;">Caste</label>
    <input type="radio" name="r1" value="General" required><label style="margin-left: 5px;">General</label>
    <input type="radio" name="r1" value="OBC" required><label style="margin-left: 5px;">OBC</label>
    <input type="radio" name="r1" value="SC" required><label style="margin-left: 5px;">SC</label>
    <input type="radio" name="r1" value="ST" required><label style="margin-left: 5px;">ST</label>
</div>

<div class="input_field">
    <label>Address</label>
    <textarea class="textarea" name="address" required></textarea>
</div>
```

```
<?php
if($_POST['register'])
{
    $fname = $_POST['fname'];
    $lname = $_POST['lname'];
    $pwd = $_POST['password'];
    $cpwd = $_POST['cpassword'];
    $gender = $_POST['gender'];
    $email = $_POST['email'];
    $phone = $_POST['phone'];
    $caste = $_POST['r1'];
    $address = $_POST['address'];

    $query = "INSERT INTO FORM2 (fname, lname, password, cpassword, gender, email, phone, caste, address)
              VALUES ('$fname', '$lname', '$pwd', '$gender', '$email', '$phone', '$caste', '$address')";
    $data = mysqli_query($conn, $query);

    if($data)
    {
        echo "Data Inserted into Database";
    }
    else
    {
```

```

<div class="input_field">
    <label style="margin-right: 100px;">Caste</label>
    <input type="radio" name="r1" value="General" required

    <?php
        if($result[caste] == "General")
        {
            echo "checked";
        }
    ?>

    <label style="margin-left: 5px;">General</label>
    <input type="radio" name="r1" value="OBC" required

    <?php
        if($result[caste] == "OBC")
        {
            echo "checked";
        }
    ?>

```

Upload Image	<input type="file" value="Choose File"/> No file chosen
First Name	<input type="text"/>
Last Name	<input type="text"/>
Password	<input type="password"/>
Confirm Password	<input type="password"/>
Gender	<input type="button" value="Select"/>
Email Address	<input type="text"/>
Phone Number	<input type="text"/>
Caste	<input type="radio"/> General <input type="radio"/> OBC <input type="radio"/> SC <input type="radio"/> ST
Language	<input type="checkbox"/> Hindi <input type="checkbox"/> Urdu <input type="checkbox"/> English
Address	<input type="text"/>
<input type="checkbox"/> Agree to terms and conditions	
<input type="button" value="Register"/>	

## Checkbox

```
<div class="input_field">
    <label style="margin-right: 100px;">Language</label>
    <input type="checkbox" name="language[]" value="Hindi" required><label style="margin-left: 5px;">Hindi</label>
    <input type="checkbox" name="language[]" value="Urdu" required><label style="margin-left: 5px;">Urdu</label>
    <input type="checkbox" name="language[]" value="English" required><label style="margin-left: 5px;">English</label>
</div>

<div class="input_field">
    <label>Address</label>
    <textarea class="textarea" name="address" required></textarea>
</div>
```

```
{
    $fname = $_POST['fname'];
    $lname = $_POST['lname'];
    $pwd = $_POST['password'];
    $cpwd = $_POST['conpassword'];
    $gender = $_POST['gender'];
    $email = $_POST['email'];
    $phone = $_POST['phone'];
    $caste = $_POST['r1'];

    $lang = $_POST['language'];
    $lang1 = implode(",",$lang);

    $address = $_POST['address'];

    $query = "INSERT INTO FORM2 (fname, lname, password, cpassword, gender, email,
        phone, caste, language, address) VALUES('$fname', '$lname', '$pwd', '$cpwd', '$
        gender', '$email', '$phone', '$caste', '$lang1', '$address')";
    $data = mysqli_query($conn, $query);

    if($data)
    {
        echo "Data Inserted into Database";
    }
}
```

```
$id = $_GET['id'];

$query = "SELECT * FROM FORM4 where id= '$id'";
$data = mysqli_query($conn, $query);
$result = mysqli_fetch_assoc($data);

$language = $result['language'];
$language1 = explode(", ", $language)

?>

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

```

<label style="margin-right: 100px;">Language</label>
    <input type="checkbox" name="language[]" value="Hindi"
        <?php
        if(in_array(Hindi, $language1))
        {
            echo "checked";
        }
        ?>

    >

<label style="margin-left: 5px;">Hindi</label>
<input type="checkbox" name="language[]" value="Urdu"

<?php
if(in_array(Urdu, $language1))
{
    echo "checked";
}
?>

>

```

```

display.php      update_design.php      form.php
0   $phone      = $_POST['phone'];
1   $caste      = $_POST['r1'];
2
3   $lang       = $_POST['language'];
4   $lang1     = implode(",",$lang);
5
6   $address    = $_POST['address'];
7
8   $query = "UPDATE form4 set fname='$fname',lname='$lname',password='$pwd'
              ,cpassword='$cpwd',gender='$gender',email='$email',phone='$phone',
              caste='$caste',language='$lang1',address='$address' WHERE id='$id'";
9
10  $data = mysqli_query($conn,$query);
11
12  if($data)
13  {
14      echo "<script>alert('Record Updated')</script>";
15      ?>
16
17      <meta http-equiv = "refresh" content = "0; url =
18          http://localhost:8080/crud/display.php" />
19
20      </?php

```

### After Insert data Redirection

```
$address = $_POST['address'];

$query = "INSERT INTO FORM4 (std_image,fname, lname, password, cpassword, gender,
    email, phone, caste, language, address) VALUES('$folder', '$fname', '$lname', '$pwd',
    '$cpwd', '$gender', '$email', '$phone', '$caste', '$lang1', '$address')";
$data = mysqli_query($conn,$query);

if($data)
{
    echo "<script> alert('Data Inserted into Database') </script>";
}
else
{
    echo "<script> alert('Failed') </script>";
}

?>
```

```
<body>
<?php
    $connection = mysqli_connect("localhost", "root", "", "aptech");
    $query = "SELECT * FROM tbl_products";
    $result = mysqli_query($connection, $query);
?>
<select>
    <option>Select Any Product</option>
    <?php
        foreach($result as $row)
        {
            echo "<option>$row[product_name]</option>";
        }
    ?>
</select>
</body>
```

## File Upload

```
<body>
    <form action="#" method="POST" enctype="multipart/form-data">
        <input type="file" name="uploadfile"><br><br>
        <input type="submit" name="submit" value="Upload File">
    </form>
</body>

<?php

//print_r($_FILES["uploadfile"]);
$filename = $_FILES["uploadfile"]["name"];
$tempname = $_FILES["uploadfile"]["tmp_name"];
$folder = "images/".$filename;
echo $folder;
move_uploaded_file($tempname, $folder);

?>

<?php

//print_r($_FILES["uploadfile"]);
$filename = $_FILES["uploadfile"]["name"];
$tempname = $_FILES["uploadfile"]["tmp_name"];
$folder = "images/".$filename;
//echo $folder;
move_uploaded_file($tempname, $folder);
    I
echo "<img src='$folder' height='100px' width='100px' ";

?>

```

```
<?php  
if(isset($_POST['submit'])) {  
    echo '<pre>';  
    print_r($_FILES);  
    move_uploaded_file($_FILES['doc']['tmp_name'],  
        'media/'. $_FILES['doc']['name']);  
}  
  
?>  
<form method="post" enctype="multipart/form-data">  
    <input type="file" name="doc"/>  
    <input type="submit" name="submit"/>  
</form>
```

```
1  <?php  
2  if(isset($_POST['submit'])) {  
3      $file=rand(111111111, 999999999);  
4  
5      echo '<pre>';  
6      print_r($_FILES);  
7      move_uploaded_file($_FILES['doc']['tmp_name'],  
8          'media/'.$file.'_'.$_FILES['doc']['name']);  
9  
10 ?>  
11 <form method="post" enctype="multipart/form-data">  
12     <input type="file" name="doc"/>  
13     <input type="submit" name="submit"/>  
14 </form>
```

```
fileupload.php * form.php * display.php
101 </body>
102
103 </html>
104
105
106 <?php
107     if($_POST['register'])
108     {
109
110
111     $filename = $_FILES["uploadfile"]["name"];
112     $tmpname = $_FILES["uploadfile"]["tmp_name"];
113     $folder = "images/".$filename;
114     move_uploaded_file($tmpname, $folder);
115
116
117     $fname    = $_POST['fname'];
118     $lname   = $_POST['lname'];
119     $pwd      = $_POST['password'];
120     $cpwd    = $_POST['conpassword'];
121     $gender   = $_POST['gender'];
122     $email    = $_POST['email'];
123     $phone    = $_POST['phone'];

$lang    = $_POST['language'];
$lang1   = implode(",",$lang);

$address = $_POST['address'];

$query = "INSERT INTO FORM4 (std_image, fname, lname, password, cpassword, gender,
    email, phone, caste, language, address) VALUES( '$folder|', '$fname', '$lname', '$
    pwd', '$cpwd', '$gender', '$email', '$phone', '$caste', '$lang1', '$address')";
$data = mysqli_query($conn,$query);

if($data)
{
    echo "Data Inserted into Database";
}
else
{
```

```

<?php
while($result = mysqli_fetch_assoc($data))
{
    echo "<tr>
        <td>".$result[id]."</td>

        <td><img src= '".$result[std_image]."' height='100px' width='100px'></td>

        <td>".$result[fname]."</td>
        <td>".$result[lname]."</td>
        <td>".$result[gender]."</td>
        <td>".$result[email]."</td>
        <td>".$result[phone]."</td>
        <td>".$result[caste]."</td>
        <td>".$result[language]."</td>
        <td>".$result[address]."</td>

        <td><a href='update_design.php?id=$result[id]'\><input type='submit' value='Update' class='update'\></a>

        <a href='delete.php?id=$result[id]'\><input type='submit' value='Delete' class='delete' onclick = 'return checkdelete()'\></a></td>
    </tr>

```

## PHP file upload

```

<?php
if(isset($_POST['submit'])){
    //move_uploaded_file();
    foreach($_FILES['doc']['name'] as $key=>$val){
        $rand=rand('11111111','99999999');
        $file=$rand.'_'.$val;
        move_uploaded_file($_FILES['doc']['tmp_name'][$key], 'media/'.$file);
        //insert into table(image) values('$file');
    }
}
?>
<form method="post" enctype="multipart/form-data">
    <input type="file" name="doc[]" multiple/>
    <input type="submit" name="submit"/>
</form>

```

## PHP oops file upload

### *upload.php*

```
<?php
class upload{
    public $src = "./upload/";
    public $tmp;
    public $filename;
    public $type;
    public $uploadfile;

    public function startupload(){
        $this -> filename = $_FILES["file"]["name"];
        $this -> tmp = $_FILES["file"]["tmp_name"];
        $this -> uploadfile = $src . basename($this -> name);
    }
    public function uploadfile(){
        if(move_uploaded_file($this -> tmp, $this ->
uploadFile)){
            return true;
        }
    }
}

?>
```

### *index.php*

```
<?php
require_once('./lib/upload.php');
?>
<?php
if(isset($_POST['file'])){
    $fileupload = new upload();
    if($fileupload -> uploadfile()){
        echo 'Success';
    }
}
?>

<html>
```

```

<head></head>
<body>
<form align="center" enctype="multipart/form-data" action="<?php
echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">
Select upload file: <input type="file" name="file"
required="yes" />
<input type="submit" value="Submit" />
<p>
</form>
</body>
</html>

```

## Upload Multiple file

<https://www.studentstutorial.com/php/file-upload-in-php-mysql>

```

<!DOCTYPE html>
<html>
<head>
<title>File Upload</title>
</head>
<body>
<form action="upload.php" method="post" enctype="multipart/form-
data">
<input type="file" name="file" />
<button type="submit" name="upload">upload</button>
</form>
</body>
</html>

```

### upload.php

```

<?php
include_once 'database.php';
if(isset($_POST['upload']))
{
    $file = rand(1000,100000)."-".$_FILES['file']['name'];
    $file_loc = $_FILES['file']['tmp_name'];
    $file_size = $_FILES['file']['size'];
    $file_type = $_FILES['file']['type'];
    $folder="upload/";

    /* new file size in KB */
    $new_size = $file_size/1024;

```

```

/* new file size in KB */

/* make file name in lower case */
$new_file_name = strtolower($file);
/* make file name in lower case */

$final_file=str_replace(' ','-',$new_file_name);

if(move_uploaded_file($file_loc,$folder.$final_file))
{
    $sql="INSERT INTO image(file,type,size)
VALUES('$final_file','$file_type','$new_size')";
mysqli_query($conn,$sql);

echo "File sucessfully upload";

}

else
{

echo "Error.Please try again";

}

?>

```

## For Multiple File

```

<!DOCTYPE html>
<html lang="en" >
<head>
<meta charset="UTF-8">
<title>Multiple image upload</title>
</head>
<body>
<form method="post" action="process.php">
<input type="file" name="image[]" multiple="multiple" >
<p align="center"><button type="submit" class="btn btn-warning" id="butsave">Submit<span class="glyphicon glyphicon-send"></span></button></p>
</form>
</body>
</html>

```

## *process.php*

```
<?php
$output_dir = "upload/"; /* Path for file upload */
$fileCount = count($_FILES["image"]['name']);
for($i=0; $i < $fileCount; $i++)

{
$RandomNum    = time();

$imageName     = str_replace(' ', '-',
',strtolower($_FILES['image']['name'][$i]));
$imageType     = $_FILES['image']['type'][$i];
/*"image/png", image/jpeg etc.*'

$imageExt = substr($imageName, strpos($imageName,
'.'));
$imageExt     = str_replace('.','',$imageExt);
$imageName     = preg_replace("/\.\[^.\s]{3,4}\$/",
'', $imageName);
$newImageName = $imageName.'-
' . $RandomNum . '.' . $imageExt;

$ret[$newImageName] = $output_dir.$newImageName;

/* Try to create the directory if it does not exist
*/
if (!file_exists($output_dir . $last_id))
{
    @mkdir($output_dir . $last_id, 0777);
}

move_uploaded_file($_FILES["image"]["tmp_name"][$i],$output_dir.
$last_id."/". $newImageName);

/*$insert_img = "insert into `category_images` SET
`category_ads_id`='".$category_ads_id_image."',
`image`='".$newImageName."'";
$result = $dbobj->query($insert_img);*/
}

echo "Image Uploaded Successfully";
?>
```

## Pegination

```
<?php

// Sample data array (replace this with your actual data source)
$data = range(1, 100); // Array with 1 to 100 elements

// Number of items per page
$itemsPerPage = 10;

// Get current page number from the URL, default to 1
$page = isset($_GET['page']) ? intval($_GET['page']) : 1;

// Calculate the offset for the data array
$offset = ($page - 1) * $itemsPerPage;

// Get the current page's data
$pageData = array_slice($data, $offset, $itemsPerPage);

// Display the data
echo "<h1>Page $page</h1>";
echo "<ul>";
foreach ($pageData as $item) {
    echo "<li>$item</li>";
}
echo "</ul>";

// Pagination links
$totalPages = ceil(count($data) / $itemsPerPage);

echo "<div>";
for ($i = 1; $i <= $totalPages; $i++) {
    echo "<a href='?page=$i'>$i</a> ";
}
echo "</div>";
?>
```

## Pagination Vishal

```
<?php
$con=mysqli_connect('localhost','root','','youtube');

$per_page=5;
$start=0;

if(isset($_GET['start'])){
    $start=$_GET['start'];
    $start--;
    $start=$start*$per_page;
}
$record=mysqli_num_rows(mysqli_query($con,"select id,title from page"));
$pagi=ceil($record/$per_page);

$sql="select id,title from page limit $start,$per_page";
$res=mysqli_query($con,$sql);
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Pagination Example</title>

<div class="container mt-100">
    <h2 class="mb-30">Pagination Example</h2>
    <ul class="list-group">
        <?php while($row=mysqli_fetch_assoc($res)){?>
            <li class="list-group-item"><?php echo $row['title']?></li>
        <?php } ?>
    </ul>
    <ul class="pagination mt-30">
        <?php for($i=1;$i<=$pagi;$i++){?>
            <li class="page-item"><a class="page-link" href="?start=<?php echo $i?><?php echo $i?></a></li>
        <?php } ?>
    </ul>
</div>
```

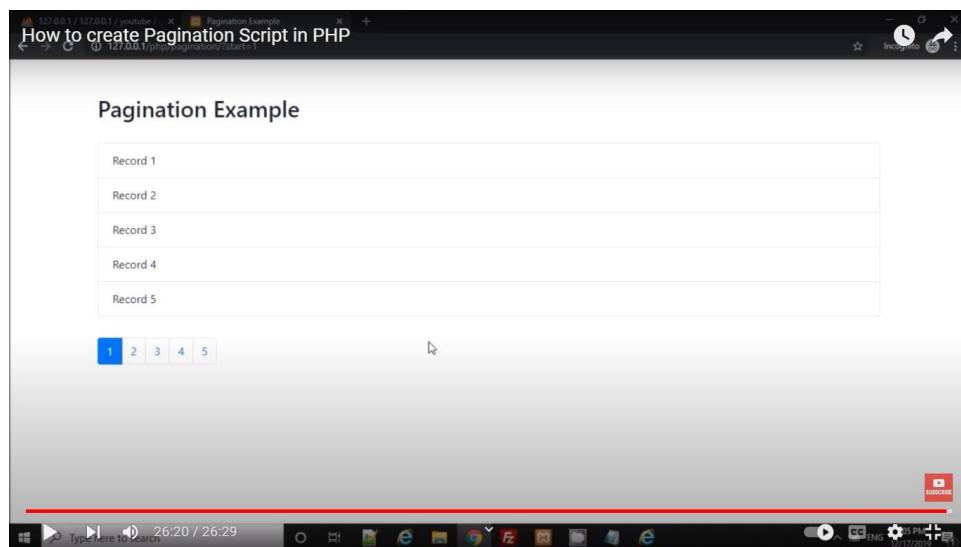
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Pagination Example</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <script src="js/jquery.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
    <style>
        .mt-100{margin-top:50px;} .mt-30{margin-top:30px;} .mb-30{margin-bottom:30px;}
    </style>
<?php
$con=mysqli_connect ('localhost','root','','voutube');
$pri_page=5;
$start=0;
$current_page=1;
if(isset($_GET['start'])){
    $start=$_GET['start'];
    if($start<=0){
        $start=0;
        $current_page=1;
    }else{
        $current_page=$start;
        $start--;
        $start=$start*$pri_page;
    }
}
$record=mysqli_num_rows(mysqli_query($con,"select id,title from page"));
$pagi=ceil($record/$pri_page);

$sql="select id,title from page limit $start,$pri_page";
$res=mysqli_query($con,$sql);
?>
```

```

</head> <body>
<div class="container mt-100">
    <h2 class="mb-30">Pagination Example</h2>
    <ul class="list-group">
        <?php
        if(mysqli_num_rows($res)>0) {
            while($row=mysqli_fetch_assoc($res)) {?>
                <li class="list-group-item"><?php echo $row['title']?></li>
            <?php } } else {?>
                No records
            <?php } ?>
        </ul>
        <ul class="pagination mt-30">
            <?php
            for($i=1;$i<=$pagi;$i++) {
                $class='';
                if($current_page==$i) {
                    ?><li class="page-item active"><a class="page-link" href="javascript:void(0)">
                        <?php echo $i?></a></li>
                <?php
                }else{
                    ?>
                    <li class="page-item"><a class="page-link" href="?start=<?php echo $i?>">
                        <?php echo $i?></a></li>
                <?php
                }
                ?>
            <?php } ?>
        </ul>
    </div> </body> </html>

```



<https://www.youtube.com/watch?v=t0gBYc7QEU8>

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Pagination Example</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <script src="js/jquery.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
    <style>
        .mt-100{margin-top:50px;} .mt-30{margin-top:30px;}.mb-30{margin-bottom:30px;}
    </style>
    <?php
$con=mysqli_connect('localhost','root','','youtube');

$per_page=5;
$start=0;
$current_page=1;
if(isset($_GET['start'])){
    $start=$_GET['start'];
    if($start<=0){
        $start=0;
        $current_page=1;
    }else{
        $current_page=$start;
        $start--;
        $start=$start*$per_page;
    }
}
$record=mysqli_num_rows(mysqli_query($con,"select id,title from page"));
$pagi=ceil($record/$per_page);

$sql="select id,title from page limit $start,$per_page";
$res=mysqli_query($con,$sql);
?>
</head>
<body>
<div class="container mt-100">
    <h2 class="mb-30">Pagination Example</h2>
    <ul class="list-group">
        <?php
        if(mysqli_num_rows($res)>0){
            while($row=mysqli_fetch_assoc($res)){?>
```

```

        <li class="list-group-item"><?php echo $row['title']?></li>
    <?php } } else {?>
    No records
    <?php } ?>
</ul>
<ul class="pagination mt-30">
    <?php
    for($i=1;$i<=$pagi;$i++){
    $class=' ';
    if($current_page==$i){
        ?><li class="page-item active"><a class="page-link"
        href="javascript:void(0)"><?php echo $i?></a></li><?php
    }else{
        ?>
        <li class="page-item"><a class="page-link" href="?start=<?php echo
        $i?>"><?php echo $i?></a></li>
        <?php
    }
    ?>

    <?php } ?>
</ul>
</div>

</body>
</html>

```

```

1 | CREATE TABLE `tbluser` (
2 | `id` int(11) NOT NULL,
3 | `Name` varchar(120) NOT NULL,
4 | `PhoneNumber` int(11) NOT NULL,
5 | `Emailid` varchar(150) NOT NULL,
6 | `PostingDate` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
7 | ) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```

1 <?php
2 // Database Configuration file
3 include('config.php');?>
4 <html>
5 <head>
6   <title>Pagination</title>
7   <!-- Bootstrap CDN -->
8   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
9   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
10  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
11 </head>
12 <body>
13 <table class="table">
14   <tr>
15     <th>#</th>
16     <th>Name</th>
17     <th>Phone Number</th>
18     <th>Email</th>
19     <th>Date</th>
20   </tr>
21 <?php
22 //Getting default page number
23   if (isset($_GET['pageno'])) {
24     $pageno = $_GET['pageno'];
25   } else {
26     $pageno = 1;
27   }
28 // Formula for pagination
29   $no_of_records_per_page = 10;
30   $offset = ($pageno-1) * $no_of_records_per_page;
31 // Getting total number of pages
32   $total_pages_sql = "SELECT COUNT(*) FROM tbluser";
33   $result = mysqli_query($con,$total_pages_sql);
34   $total_rows = mysqli_fetch_array($result)[0];
35   $total_pages = ceil($total_rows / $no_of_records_per_page);
36   $sql = "SELECT * FROM tbluser LIMIT $offset, $no_of_records_per_page";
37   $res_data = mysqli_query($con,$sql);
38   $cnt=1;
39   while($row = mysqli_fetch_array($res_data)){?>
40     <tr>
41       <td><?php echo $cnt;?></td>
42       <td><?php echo $row['Name'];?></td>
43       <td><?php echo $row['PhoneNumber'];?></td>
44       <td><?php echo $row['Emailid'];?></td>
45       <td><?php echo $row['PostingDate'];?></td>
46     </tr>
47   <?php
48   $cnt++;
49   ?>
50 </table>
51 <div align="center">
52   <ul class="pagination" >
53     <li><a href="?pageno=1">First</a></li>
54     <li class="disabled"><?php if($pageno <= 1){ echo 'disabled'; } ?><a href="<?php if($pageno <= 1){ echo '#'; } else { echo "?pageno=".($pageno - 1); } ?>">Prev</a>
55     </li>
56     <li class="disabled"><?php if($pageno >= $total_pages){ echo 'disabled'; } ?><a href="<?php if($pageno >= $total_pages){ echo '#'; } else { echo "?pageno=".($pageno + 1); } ?>">Next</a>
57     </li>
58     <li><a href="?pageno=<?php echo $total_pages; ?>">Last</a></li>
59   </ul>
</div>

```

# Paypal Integration in PHP

## Step 1: Create a PayPal Business Account

If you don't already have a PayPal business account, you need to create one. This account will allow you to receive payments.

## Step 2: Get PayPal API Credentials

PayPal API Username

PayPal API Password

PayPal API Signature

## Step 3: Set Up Your PHP Application

In your PHP application, create a form to collect payment information from users. Here's a simplified example of a payment form:

```
<!DOCTYPE html>
<html>
<head>
    <title>Pay with PayPal</title>
</head>
<body>
    <h2>Pay with PayPal</h2>
    <form action="process_payment.php" method="post">
        <input type="hidden" name="cmd" value="_xclick">
        <input type="hidden" name="business" value="your-paypal-
email@example.com">
        <input type="hidden" name="item_name" value="Product Name">
        <input type="hidden" name="item_number" value="123">
        <input type="hidden" name="amount" value="10.00">
        <input type="hidden" name="currency_code" value="USD">
        <input type="hidden" name="return" value="http://your-
website.com/thank-you.php">
        <input type="submit" name="submit" value="Pay with PayPal">
    </form>
</body>
</html>
```

#### Step 4: Create a PHP Script to Process Payments (process\_payment.php)

```
<?php
$paypal_url = 'https://www.sandbox.paypal.com/cgi-bin/webscr'; // Use the
sandbox URL for testing

// PayPal API credentials
$paypal_email = 'your-paypal-email@example.com'; // Your PayPal business
email
$paypal_currency = 'USD';

// Collect payment data from the form
$item_name = $_POST['item_name'];
$item_number = $_POST['item_number'];
$amount = $_POST['amount'];

// Build PayPal request parameters
$fields = [
    'cmd' => '_xclick',
    'business' => $paypal_email,
    'item_name' => $item_name,
    'item_number' => $item_number,
    'amount' => $amount,
    'currency_code' => $paypal_currency,
    'return' => 'http://your-website.com/thank-you.php', // Redirect URL
after payment
];
// Prepare the query string
$query_string = http_build_query($fields);

// Redirect to PayPal to complete the payment
header("location:$paypal_url?$query_string");
exit();
?>
```

#### Step 5: Handle the PayPal Response

#### Step 6: Test in the PayPal Sandbox

## PHP program with a trait and a class that uses that trait:

```
<?php
// Define a trait called Logger
trait Logger {
    public function log($message) {
        echo "Logging: $message\n";
    }
}

// Create a class that uses the Logger trait
class MyClass {
    use Logger;

    public function someMethod() {
        $this->log("This is a log message from MyClass");
    }
}

// Create another class that uses the Logger trait
class AnotherClass {
    use Logger;

    public function anotherMethod() {
        $this->log("This is a log message from AnotherClass");
    }
}

// Create instances of MyClass and AnotherClass
$obj1 = new MyClass();
$obj2 = new AnotherClass();

// Call methods that use the log method from the trait
$obj1->someMethod();
$obj2->anotherMethod();
?>
```

```

1 <?php
2 class class1{
3     function __construct(){
4         echo "start";
5     }
6
7     function fun1(){
8         echo "Hello";
9     }
10
11    function __destruct(){
12        echo "end";
13    }
14 }
15
16 $obj1=new class1();
17 $obj1->fun1();
18 ?>

```

```

<?php
class class1{
    function fun1(){
        echo "Hello";
    }
}

function __construct(){
    echo "start";
}

function __destruct(){
    echo "end";
}

$obj1=new class1();
$obj1->fun1();
?>

```

 PHP OOP Introduction Tutorial in Hindi / Urdu

## Class & Object in PHP

class calculation{  
 public \$a , \$b , \$c ;  
 function sum(){  
 \$this->c = \$this->a + \$this->b;  
 return \$this->c;  
 }  
 function sub(){  
 \$this->c = \$this->a - \$this->b;  
 return \$this->c;  
 }  
}

\$c1 = new calculation();  
\$c1->a = 10;  
\$c1->b = 20;  
echo \$c1->sum() ;  
↓  
30

Yahoo Baba

13:11 / 21:53

www.yahobaba.net

```
constructor.php
```

```
1 <?php
2
3 class person{
4     public $name = "No Name";
5     public $age = 0;
6
7     function __construct($name , $age){
8         $this->name = $name;
9         $this->age = $age;
10    }
11
12    function show(){
13        echo $this->name . " - " . $this->age;
14    }
15 }
16
17 $p1 = new person("Yahoo Baba",20);
18
19 // $p1->name = "Yahoo Baba";
20 // $p1->age = 20;
21
22 $p1->show();
23
```

Yahoo  
Baba

```
employee.php
```

```
3 class employee{
4     public $name;
5     public $age;
6     public $salary;
7
8     function __construct($n, $a, $s){
9         $this->name = $n;
10        $this->age = $a;
11        $this->salary = $s;
12    }
13
14    function info(){
15        echo "<h3>Employee Profile</h3>";
16        echo "Employee Name : " . $this->name . "<br>";
17        echo "Employee Age : " . $this->age . "<br>";
18        echo "Employee Salary : " . $this->salary . "<br>";
19    }
20
21 }
22
23 $e1 = new employee("Ram", 25, 2000);
24
25 $e1->info();      @
26 ?>
```

Yahoo  
Baba



## Overriding Methods

```
class A{
    public $name;
    public function show(){
        echo "My Name" . $this->name;
    }
}

class B extends class A{
    public function show(){
        echo "Your Name" . $this->name;
    }
}
```



## Abstract Class

```
1 → abstract class A{
    protected $name;
    protected function __construct($n){
        $this->name = $n;
    }
}
2 → abstract protected function show(); ← Declare ✓
class B extends class A{
    3 → public function show(){
        echo $this->name;
    }
}
```

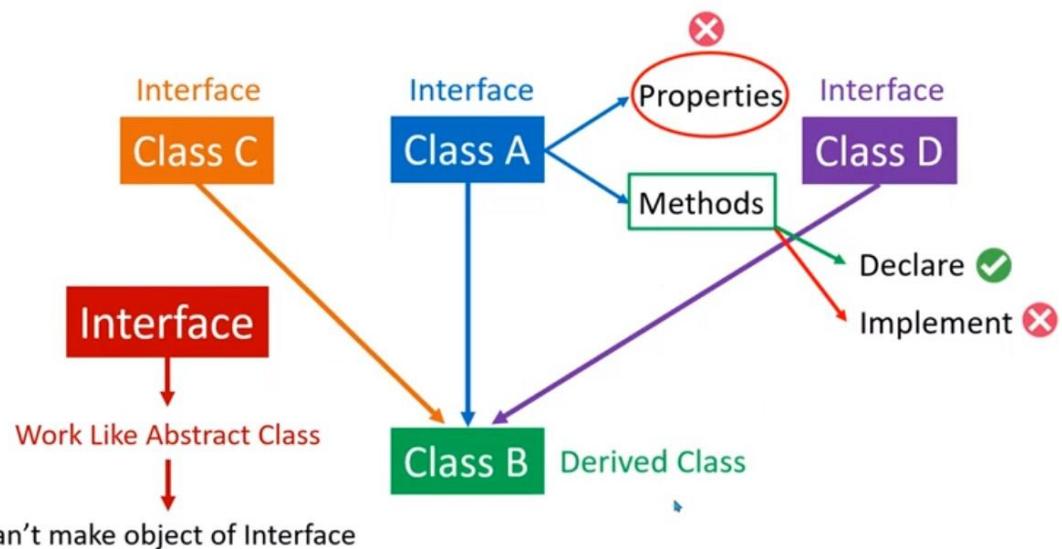
Implement ✗

```
abstract.php
1 <?php
2
3 abstract class parentClass{
4
5     public $name;
6
7     abstract protected function calc($a, $b);
8
9 }
10
11 class childClass extends parentClass{
12
13     public function calc($c, $d){
14         echo $c + $d;
15     }
16
17 }
18
19 $test = new childClass();
20
21 $test->calc(10, 20);
```

Yahoo  
Baba



## What is Interface ?





## Interface

1 → interface A{  
    function hello(\$n);  
}  
  
interface C{  
    2 → function hi(\$n);  
    function bye();  
}

3 → \$test = new A(); 

4 → class B implements A , C{  
    public function hello(\$n){  
        echo "Hello " . \$n;  
    }  
  
    public function hi(\$n){  
        echo "Hi " . \$n;  
    }  
  
    public function bye(){  
        echo "Bye";  
    }  
}



## Static Members in PHP

class personal{ ← Static Class  
    public static \$name = "Yahoo Baba" ;  
    public static function show(){  
        echo self::\$name;  
    }  
}  
  
personal::\$name;  
personal::show();





## Static Members in PHP

```
class personal{  
    public static $name = "Yahoo Baba" ;  
}
```

```
class personal extends accounts{  
    public function show(){  
        echo parent::$name;  
    }  
}
```

```
$test = new accounts();  
$test->show();
```



## Late Static Binding in PHP

```
class personal{  
    protected static $name = "Yahoo" ;  
    public function show(){  
        echo static::name;  
    }  
}  
  
class personal extends accounts{  
    protected static $name = "Baba" ;  
}
```

```
$test = new accounts();  
$test->show();
```

Baba





## Traits in PHP

```
trait test{
    public function hello(){
        echo "Say Hello";
    }
}

class A{
    use test;
}

class B{
    use test;
}
```

```
$obj = new A();
$obj->hello();
```



## What is Type Hinting ?

```
function sum(int $value){
    echo $value + 10;
}

sum(10);
sum("Hey");
```

Type declaration

Valid DataTypes

- Int
- Float
- String
- Bool
- Array
- Class / Interface Name
- Object





## Namespace

First.php

```
namespace first;  
class test{  
}  
}
```

Second.php

```
namespace second;  
class test{  
}
```

Third.php

```
require First.php;  
require second.php;
```

**\$obj = new first\test();**



Yahoo Baba

www.yahooobaba.net

YahOO



## Method Chaining

```
class personal{  
    public function first(){  
        echo "This is first function";  
    }  
  
    public function second(){  
        echo "This is second function";  
    }  
  
    public function third(){  
        echo "This is third function";  
    }  
}
```

```
$test = new personal();  
$test->first();  
$test->second();  
$test->third();
```

**\$test->first()->second()->third();**



Yahoo Baba

www.yahooobaba.net

YahOO



## Magic Methods in PHP

- `__construct()`
- `__destruct()`
- `__get()`
- `__set()`
- `__isset()`
- `__unset()`
- `__autoload()`
- `__clone()`
- `__sleep()`
- `__wakeup()`
- `__call()`
- `__callStatic()`
- `__toString()`
- `__invoke`



## PHP : List of Magic Constants

- `__LINE__`
- `__FILE__`
- `__DIR__`
- `__FUNCTION__`
- `__CLASS__`
- `__METHOD__`
- `__NAMESPACE__`
- `__TRAIT__`



## PHP : List of Conditional Functions

- class\_exists()
- interface\_exists()
- method\_exists()
- trait\_exists()
- property\_exists()
- is\_a()
- is\_subclass\_of()



## PHP : List of Get Functions

- |                        |                           |
|------------------------|---------------------------|
| • get_class            | • get_declared_interfaces |
| • get_parent_class     | • get_declared_traits     |
| • get_class_methods    | • class_alias             |
| • get_class_vars       |                           |
| • get_object_vars      |                           |
| • get_called_class     |                           |
| • get_declared_classes |                           |

```
<?php

// Interface definition
interface Animal {
    public function makeSound();
}

// Class definitions
class Cat implements Animal {
    public function makeSound() {
        echo " Meow ";
    }
}

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

class Mouse implements Animal {
    public function makeSound() {
        echo " Squeak ";
    }
}

// Create a list of animals
$cat = new Cat();
$dog = new Dog();
$mouse = new Mouse();
$animals = array($cat, $dog, $mouse);

// Tell the animals to make a sound
foreach($animals as $animal) {
    $animal->makeSound();
}
```

## Syntax

Get your own PHP Server

```
<?php
interface InterfaceName {
    public function someMethod1();
    public function someMethod2($name, $color);
    public function someMethod3() : string;
}
?>
```

## PHP - Interfaces vs. Abstract Classes

Interface are similar to abstract classes. The difference between interfaces and abstract classes are:

- Interfaces cannot have properties, while abstract classes can
- All interface methods must be public, while abstract class methods is public or protected
- All methods in an interface are abstract, so they cannot be implemented in code and the `abstract` keyword is not necessary
- Classes can implement an interface while inheriting from another class at the same time

## PHP - What are Traits?

PHP only supports single inheritance: a child class can inherit only from one single parent.

So, what if a class needs to inherit multiple behaviors? OOP traits solve this problem.

Traits are used to declare methods that can be used in multiple classes. Traits can have methods and abstract methods that can be used in multiple classes, and the methods can have any access modifier (public, private, or protected).

Traits are declared with the `trait` keyword:

## Syntax

Get your own PHP Server

```
<?php
trait TraitName {
    // some code...
}
?>
```

## Syntax

```
<?php
class MyClass {
    use TraitName;
}
?>
```

# PHP - Static Methods

Static methods can be called directly - without creating an instance of the class first.

Static methods are declared with the `static` keyword:

## Syntax

[Get your own PHP Server](#)

```
<?php
class ClassName {
    public static function staticMethod() {
        echo "Hello World!";
    }
}
?>
```

To access a static method use the class name, double colon (::), and the method name:

## Syntax

```
ClassName::staticMethod();
```

## Example

```
<?php
class greeting {
    public static function welcome() {
        echo "Hello World!";
    }
}

// Call static method
greeting::welcome();
?>
```

[Try it Yourself »](#)

## Example Explained

Here, we declare a static method: `welcome()`. Then, we call the static method by using the class name, double colon (::), and the method name (without creating an instance of the class first).

## Example

```
<?php
class greeting {
    public static function welcome() {
        echo "Hello World!";
    }

    public function __construct() {
        self::welcome();
    }
}

new greeting();
?>
```

```
<?php
class pi {
    public static $value=3.14159;
}

class x extends pi {
    public function xStatic() {
        return parent::$value;
    }
}

// Get value of static property directly via child class
echo x::$value;

// or get value of static property via xStatic() method
$x = new x();
echo $x->xStatic();
?>
```

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate  
FROM Orders  
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```

## JOIN Three Tables

The following SQL statement selects all orders with customer and shipper information:

### Example

```
SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName  
FROM ((Orders  
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID)  
INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID);
```

## MySQL LEFT JOIN Example

The following SQL statement will select all customers, and any orders they might have:

### Example

[Get your own SQL](#)

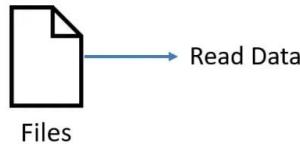
```
SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID  
ORDER BY Customers.CustomerName;
```

[Try it Yourself »](#)

**Note:** The `LEFT JOIN` keyword returns all records from the left table (`Customers`), even if there are no matches in the right table (`Orders`).



## PHP : Read from the File



`fopen(file name, mode)`

`fread(file, length)`

`fclose(file)`

`file_get_contents(filename, include_path, context, start, max_length)`

↓  
FALSE



www.yahooibaba.net

Yahoo  
Baba



## What is UNION & UNION ALL?

Student Table

ID	Name	City
1	Ram Kumar	Agra
2	Salman Khan	Delhi
3	Anil Kapoor	Bhopal

UNION ALL

Name	City
Ram Kumar	Agra
Salman Khan	Delhi
Meera Khan	Bhopal

Lecturer Table

ID	Name	City
1	Salim Khan	Agra
2	Ram Kumar	Delhi
3	Sarita	Agra

UNION

X

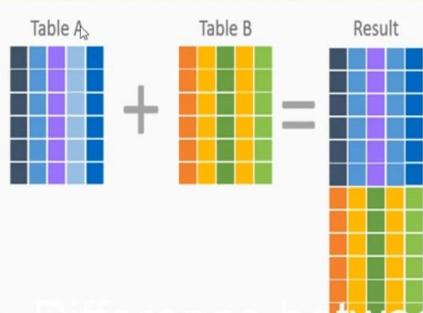


www.yahooibaba.net

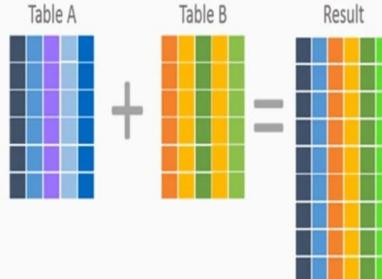


## Difference Between Join and Union

Unions combine data into new rows



Join combine data into columns.



Difference between Union and Join

## JOIN

- JOIN combines attributes of the tuples present in the two different relations that share some common fields or attributes.
- JOIN is applicable when the two involved relations have at least one common attribute.
- Join will fetch records common to the both tables which are joined.
- Join is used to combine the result of two or more tables using single query.



UPSKILL WITH ANAS



## UNION & UNION ALL Syntax

`SELECT column1, column2 FROM table1`

`UNION / UNION ALL`

`SELECT column1, column2 FROM table2;`

## UNION

- UNION combines tuples of the relations that are present in the query.
- UNION is applicable when the number of columns present in query are same and the corresponding attributes has the same domain.
- Union will fetch all the records from both the tables.
- Union is used to combine the result of two or more tables using more than one query.



TECHLEGIONZ

Yahoo Baba [www.yahooibaba.net](http://www.yahooibaba.net)

```
mysql-error-functions.php
```

```
1  <?php
2
3  $conn = mysqli_connect("localhost","root","","test") or die("Connection Failed : ". mysqli_error($conn));
4
5  $sql = "SELECT * FROM students";
6  $result = mysqli_query($conn, $sql) or die("Query Unsuccessful : ". mysqli_error($conn));
7
8  $str = "";
9  if(mysqli_num_rows($result) > 0) {
10
11    while($row = mysqli_fetch_assoc($result)){
12      echo $row['first_name']."' ".$row['last_name'] . "<br>";
13    }
14
15  }
16
17 ?>
18
```

File Line 522 GitHub [Edit on GitHub](#) [Update](#)

```
mysqli-error-functions.php
1 <?php
2
3 $conn = mysqli_connect("localhost","root","","test") or die("Connection Failed :
4
5 $sql = "SELECT * FROM students";
6 $result = mysqli_query($conn, $sql);
7
8 print_r(mysqli_error_list($conn));
9 die();
10
11 $str = "";
12 if(mysqli_num_rows($result) > 0) {
13
14     while($row = mysqli_fetch_assoc($result)){
15         echo $row['first_name']." ".$row['last_name'] . "<br>";
16     }
17
-- myfile.php                               readme.txt
1 <?php
2
3 echo file_get_contents("readme.txt",FALSE,NULL,50);
4
5
6 ?>
7
```

Yahoo  
Baba



## PHP MySQLi Error Functions :

- `mysqli_connect_error()`
- `mysqli_connect_errno()`
- `mysqli_error()`
- `mysqli_error_list()`

```
1 <?php  
2  
3 echo file_put_contents("readme.txt","Another new text.",FILE_APPEND | LOCK_EX);  
4  
5  
6 ?>  
7
```

Yahoo  
Baba



## PHP : Write in the File



`fopen(file name, mode)`

`fwrite(file, string data)`

`fclose(file)`

`file_put_contents(filename, data, mode, context)`

- FILE\_APPEND
- LOCK\_EX

## Magic Method

```
<?php
class UserProfile {
    private $data = [];

    public function __construct($name, $email) {
        $this->data['name'] = $name;
        $this->data['email'] = $email;
    }

    public function __get($property) {
        if (array_key_exists($property, $this->data)) {
            return $this->data[$property];
        }
        return null;
    }

    public function __set($property, $value) {
        $this->data[$property] = $value;
    }

    public function __toString() {
        return "Name: {$this->data['name']}, Email: {$this->data['email']}";
    }
}

// Create an instance of UserProfile
$user = new UserProfile("John Doe", "johndoe@example.com");

// Access and modify properties using magic methods
echo "Name: " . $user->name . "\n"; // __get is called
$user->name = "Jane Doe"; // __set is called
echo "Updated Name: " . $user->name . "\n";

// Convert the object to a string using __toString
echo "User Profile: " . $user . "\n";
?>
```

1. We have a UserProfile class that stores user data in a private \$data array.

2. The \_\_construct method is called when an object of the class is created, initializing the name and email properties.

3.The \_\_get method is used to retrieve property values. If a property doesn't exist, it returns null.

4.The \_\_set method allows us to set or modify property values.

5.The \_\_toString method customizes how the object is converted to a string when we use it in a string context, like echo.

```
<?php
class MagicDemo {
    private $data = [];

    public function __construct() {
        echo "An instance of MagicDemo has been created.\n";
    }

    public function __destruct() {
        echo "An instance of MagicDemo is being destroyed.\n";
    }

    public function __get($property) {
        echo "Getting property '$property': ";
        if (array_key_exists($property, $this->data)) {
            return $this->data[$property];
        } else {
            return "Property does not exist.\n";
        }
    }

    public function __set($property, $value) {
        echo "Setting property '$property' to '$value'.\n";
        $this->data[$property] = $value;
    }

    public function __isset($property) {
        echo "Checking if property '$property' is set: ";
        return isset($this->data[$property]);
    }

    public function __unset($property) {
        echo "Unsetting property '$property'.\n";
        unset($this->data[$property]);
    }

    public function __call($method, $arguments) {
        echo "Calling method '$method' with arguments: ";
        print_r($arguments);
    }
}
```

```
}

public function __toString() {
    return "This is a MagicDemo object.";
}
}

// Create an instance of MagicDemo
$obj = new MagicDemo();

// Access properties using magic __get and __set
$obj->name = "John";
echo "Name: " . $obj->name . "\n";

// Check property existence using __isset
echo isset($obj->age) ? "Age is set.\n" : "Age is not set.\n";

// Unset property using __unset
unset($obj->name);

// Call undefined method using __call
$obj->doSomething(42, "Hello");

// Convert object to string using __toString
echo $obj;
?>
```

## ALL php magic constant uses one program

1. `__LINE__`: Returns the current line number of the file.
2. `__FILE__`: Returns the full path and filename of the file.
3. `__DIR__`: Returns the directory of the file.
4. `__FUNCTION__`: Returns the name of the current function.
5. `__CLASS__`: Returns the name of the current class.
6. `__METHOD__`: Returns the name of the current class method.
7. `__NAMESPACE__`: Returns the current namespace name

```
<?php
class MyClass {
    public function __construct() {
        echo "Inside " . __METHOD__ . " in " . __CLASS__ . " within namespace
" . __NAMESPACE__ . "\n";
    }

    public function showFilePath() {
        echo "Current file: " . __FILE__ . "\n";
        echo "Current directory: " . __DIR__ . "\n";
    }
}

echo "Current line number: " . __LINE__ . "\n";

$obj = new MyClass();
$obj->showFilePath();
?>
```

1. `__LINE__` is used to display the current line number.
2. `__FILE__` and `__DIR__` are used within the `showFilePath` method of the `MyClass` class to display the current file and directory.
3. `__METHOD__`, `__CLASS__`, and `__NAMESPACE__` are used within the constructor of the `MyClass` class to display information about the method, class, and namespace.

```

<?php
namespace MyNamespace;

class MyClass {
    private $property;

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

    public function displayInfo() {
        echo "Inside " . __METHOD__ . " in " . __CLASS__ . " within namespace
" . __NAMESPACE__ . "\n";
    }

    public function showFilePath() {
        echo "Current file: " . __FILE__ . "\n";
        echo "Current directory: " . __DIR__ . "\n";
    }

    public function getProperty() {
        return $this->property;
    }
}

// Use __LINE__ to display the current line number
echo "Current line number: " . __LINE__ . "\n";

// Create an instance of MyClass and use various magic constants
$obj = new MyClass(42);
$obj->displayInfo();
$obj->showFilePath();

// Access class property and display it along with magic constants
echo "Property value: " . $obj->getProperty() . "\n";
?>

```

1. `__LINE__` is used to display the current line number.
2. `__METHOD__`, `__CLASS__`, and `__NAMESPACE__` are used within the `MyClass` class to display information about the method, class, and namespace.
3. `__FILE__` and `__DIR__` are used within the `showFilePath` method of the `MyClass` class to display the current file and directory.
4. Magic constants are used to display information within the main part of the script.

## let var const in js program example

```
// Using var
function varExample() {
    var x = 10;
    if (true) {
        var x = 20; // This reassigns the same variable x
    }
    console.log("Var Example:", x); // Outputs 20
}
varExample();

// Using let
function letExample() {
    let y = 10;
    if (true) {
        let y = 20; // This creates a new variable y within the block
    }
    console.log("Let Example:", y); // Outputs 10
}
letExample();

// Using const
function constExample() {
    const z = 10;
    // z = 20; // This will result in an error (assignment to a constant
    // variable)
    console.log("Const Example:", z);
}
constExample();
```

1.The varExample function uses var to declare a variable x. It reassigns x within an if block, and the modified value is reflected outside the block.

2.The letExample function uses let to declare a variable y. It creates a new variable y within the if block, and the original value remains unchanged outside the block.

3.The constExample function uses const to declare a constant variable z. An attempt to reassign z to a different value would result in an error.

## Calling parent constructors

```
class ParentClass {
    public function __construct() {
        echo "ParentClass constructor called<br>";
    }
}

class ChildClass extends ParentClass {
    public function __construct() {
        parent::__construct(); // Calling the parent class constructor
        echo "ChildClass constructor called<br>";
    }
}

$child = new ChildClass();
```

```
class Name {
    var $_firstName;
    var $_lastName;

    function Name($first_name, $last_name) {
        $this->_firstName = $first_name;
        $this->_lastName = $last_name;
    }

    function toString() {
        return($this->_lastName . ", " . $this->_firstName);
    }
}
class NameSub1 extends Name {
    var $_middleInitial;

    function NameSub1($first_name, $middle_initial, $last_name) {
        Name::Name($first_name, $last_name);
        $this->_middleInitial = $middle_initial;
    }

    function toString() {
        return(Name::toString() . " " . $this->_middleInitial);
    }
}
```

## Calling Parent Constructor

```
class DatabaseConnection {
    protected $host;
    protected $username;
    protected $password;
    protected $database;
    protected $connection;

    public function __construct($host, $username, $password, $database) {
        $this->host = $host;
        $this->username = $username;
        $this->password = $password;
        $this->database = $database;
    }

    public function connect() {
        // Generic database connection logic
        $this->connection = new mysqli($this->host, $this->username, $this-
>password, $this->database);
        if ($this->connection->connect_error) {
            die("Database Connection Error: " . $this->connection-
>connect_error);
        }
        echo "Connected to the database<br>";
    }
}

class MySQLDatabaseConnection extends DatabaseConnection {
    public function __construct($host, $username, $password, $database) {
        parent::__construct($host, $username, $password, $database);
    }

    // Additional methods and MySQL-specific logic can be added here
}

// Example usage
$mysqlDb = new MySQLDatabaseConnection('localhost', 'mysqluser', 'mysqlpass',
'mysqldb');
$mysqlDb->connect();
```

## static Keyword

```
<?php
class Foo {
    public static $my_static = 'foo';

    public function staticValue() {
        return self::$my_static;
    }
}

print Foo::$my_static . "\n";
$foo = new Foo();

print $foo->staticValue() . "\n";
?>
```

## late static binding

```
class ParentClass {
    public static function whoAmI() {
        echo "I am from " . static::class . "<br>";
    }
}

class ChildClass extends ParentClass {}

class AnotherChildClass extends ParentClass {}

ParentClass::whoAmI();      // Output: I am from ParentClass
ChildClass::whoAmI();       // Output: I am from ChildClass
AnotherChildClass::whoAmI(); // Output: I am from AnotherChildClass
```

## Realworld example

```
class Shape {
    protected $area;

    public function getArea() {
        return $this->area;
    }

    public static function create($type, $params) {
        switch ($type) {
            case 'circle':
                return Circle::createCircle($params);
            case 'rectangle':
                return Rectangle::createRectangle($params);
            default:
                throw new InvalidArgumentException("Invalid shape type: $type");
        }
    }
}

class Circle extends Shape {
    protected $radius;

    public function __construct($radius) {
        $this->radius = $radius;
        $this->area = $this->calculateArea();
    }

    public function calculateArea() {
        return pi() * $this->radius * $this->radius;
    }

    public static function createCircle($params) {
        return new static($params['radius']);
    }
}

class Rectangle extends Shape {
    protected $width;
    protected $height;

    public function __construct($width, $height) {
        $this->width = $width;
        $this->height = $height;
    }
}
```

```

        $this->area = $this->calculateArea();
    }

    public function calculateArea() {
        return $this->width * $this->height;
    }

    public static function createRectangle($params) {
        return new static($params['width'], $params['height']);
    }
}

// Create a circle and a rectangle using the factory method
$circle = Shape::create('circle', ['radius' => 5]);
$rectangle = Shape::create('rectangle', ['width' => 4, 'height' => 6]);

// Display the areas of the shapes
echo "Circle Area: " . $circle->getArea() . "<br>";
echo "Rectangle Area: " . $rectangle->getArea() . "<br>";

```

## parent Keyword

```

class ParentClass {
    protected $property = "I am a property in the parent class";

    public function printProperty() {
        echo $this->property . "<br>";
    }
}

class ChildClass extends ParentClass {
    protected $property = "I am a property in the child class";

    public function printParentProperty() {
        // Access the parent class property using the parent keyword
        echo parent::$property . "<br>";

        // Call the parent class method using the parent keyword
        parent::printProperty();
    }
}

$child = new ChildClass();
$child->printParentProperty();

```

## **final**

- **final** keyword is used to create final method or final class.
- A final method cannot be overridden in child class
- A final class cannot be inherited It means we can not create sub class of a final class.

```
Class Father {  
    function display(){  
        echo "You can override me becoz I am not final";  
    }  
    final function show(){  
        echo "I am final so you cannot override me";  
    }  
}  
class Son extends Father {  
    function display(){  
        echo "Yea I overrided";  
    }  
}
```

## **Final Class**

```
final class Father {  
    function display(){  
        echo "Final";  
    }  
}
```

## Final Keword (method)

```
class ParentClass {
    final protected function getProperty() {
        return "This is a final property.";
    }
}

class ChildClass extends ParentClass {
    public function getProperty() {
        return parent::getProperty();
    }
}

$child = new ChildClass();
echo $child->getProperty(); // Output: This is a final property.
```

## Final Keword (class)

```
final class FinalClass {
    public function sayHello() {
        echo "Hello from FinalClass!<br>";
    }
}

// Attempting to extend a final class will result in an error
class ChildClass extends FinalClass {
    // This will cause a fatal error
}

$finalObj = new FinalClass();
$finalObj->sayHello();
```

## Abstract class and method

```
// Abstract class for PaymentGateway
abstract class PaymentGateway {
    protected $apiKey;

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

    // Abstract method for processing payment
    abstract public function processPayment($amount);
}

// Concrete class for PayPal payment gateway
class PayPalGateway extends PaymentGateway {
    public function processPayment($amount) {
        // Implement the PayPal-specific payment processing logic here
        // Use $this->apiKey to authenticate with PayPal API
        // Process the payment and return the result
        echo "Processing payment of $amount USD using PayPal.<br>";
    }
}

// Concrete class for Stripe payment gateway
class StripeGateway extends PaymentGateway {
    public function processPayment($amount) {
        // Implement the Stripe-specific payment processing logic here
        // Use $this->apiKey to authenticate with Stripe API
        // Process the payment and return the result
        echo "Processing payment of $amount USD using Stripe.<br>";
    }
}

// Usage
$paypalGateway = new PayPalGateway('your_paypal_api_key');
$stripeGateway = new StripeGateway('your_stripe_api_key');

$paypalGateway->processPayment(100);
$stripeGateway->processPayment(75);
```

## Other example

```
abstract class PaymentGateway {
    abstract public function processPayment($amount);
}

class PayPalGateway extends PaymentGateway {
    public function processPayment($amount) {
        // Process payment using PayPal API
        echo "Processing payment of $amount via PayPal.<br>";
    }
}

class StripeGateway extends PaymentGateway {
    public function processPayment($amount) {
        // Process payment using Stripe API
        echo "Processing payment of $amount via Stripe.<br>";
    }
}

// Example usage
$paypalGateway = new PayPalGateway();
$paypalGateway->processPayment(100);

$stripeGateway = new StripeGateway();
$stripeGateway->processPayment(150);
```

## Interface Example

```
1 Interface
2
3 => Interface resamble abstract classes
4 => Interface contains abstract function like abstract classes
5 => Interface can't be instantiated like abstract classes
6 => Interface can not contain concrete methods like abstract classes
7 => Interface contains only public methods not private or protected
8 => An abstract can only extends by 1 abstract class
9 => An class can implements more than 1 interface
10 => Multiple inheritance can be achived through interfaces
11 => Interface can not contains constructor
12 => Interface can not contains data members
13
14
```

---

```
<?php

interface demo
{
    I
    public function test1();
    public function test2();
}

interface demo2
{
    public function test3();
}

class childClass implements demo,demo2
{
    public function test1(){}
    public function test2(){}
}
```

```
interface DatabaseInterface {
    public function connect($host, $username, $password, $database);
    public function query($sql);
    public function close();
}

class MySQLDatabase implements DatabaseInterface {
    private $connection;

    public function connect($host, $username, $password, $database) {
        // Implement the connection logic for MySQL
        $this->connection = mysqli_connect($host, $username, $password,
$database);
        if (!$this->connection) {
            die("MySQL Connection Error: " . mysqli_connect_error());
        }
    }

    public function query($sql) {
        // Implement the query execution logic for MySQL
        $result = mysqli_query($this->connection, $sql);
        return $result;
    }

    public function close() {
        // Implement the connection closing logic for MySQL
        mysqli_close($this->connection);
    }
}

class PostgreSQLDatabase implements DatabaseInterface {
    private $connection;

    public function connect($host, $username, $password, $database) {
        // Implement the connection logic for PostgreSQL
        $this->connection = pg_connect("host=$host dbname=$database
user=$username password=$password");
        if (!$this->connection) {
            die("PostgreSQL Connection Error: " . pg_last_error());
        }
    }

    public function query($sql) {
        // Implement the query execution logic for PostgreSQL
        $result = pg_query($this->connection, $sql);
    }
}
```

```

        return $result;
    }

    public function close() {
        // Implement the connection closing logic for PostgreSQL
        pg_close($this->connection);
    }
}

// Example usage
$mysqlDB = new MySQLDatabase();
$mysqlDB->connect('localhost', 'user', 'password', 'mydb');
$result = $mysqlDB->query('SELECT * FROM mytable');
$mysqlDB->close();

$postgresDB = new PostgreSQLDatabase();
$postgresDB->connect('localhost', 'user', 'password', 'mydb');
$result = $postgresDB->query('SELECT * FROM mytable');
$postgresDB->close();

```

## Another Example Interface

```

interface ContentElement {
    public function render();
    public function getMetadata();
}

class Article implements ContentElement {
    private $title;
    private $content;

    public function __construct($title, $content) {
        $this->title = $title;
        $this->content = $content;
    }

    public function render() {
        // Implement rendering logic for an article
        return "<h1>{$this->title}</h1><p>{$this->content}</p>";
    }

    public function getMetadata() {

```

```
// Implement metadata retrieval logic for an article
    return "Title: {$this->title}";
}
}

class Image implements ContentElement {
    private $src;
    private $alt;

    public function __construct($src, $alt) {
        $this->src = $src;
        $this->alt = $alt;
    }

    public function render() {
        // Implement rendering logic for an image
        return "<img src='{$this->src}' alt='{$this->alt}'>";
    }

    public function getMetadata() {
        // Implement metadata retrieval logic for an image
        return "Alt Text: {$this->alt}";
    }
}

// Example usage
$article = new Article("Sample Article", "This is a sample article
content.");
$image = new Image("image.jpg", "Sample Image Alt Text");

echo $article->render();
echo $article->getMetadata();

echo $image->render();
echo $image->getMetadata();
```

## PHP interface example code in details in

In PHP, an interface is a contract that defines a set of methods a class must implement. Interfaces allow you to define a common set of methods that multiple classes can adhere to. Here's an example of a PHP interface along with a class that implements it:

```
<?php

// Define an interface
interface Shape {
    public function getArea();
    public function getPerimeter();
}

// Create a class that implements the interface
class Circle implements Shape {
    private $radius;

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

    public function getArea() {
        return pi() * pow($this->radius, 2);
    }

    public function getPerimeter() {
        return 2 * pi() * $this->radius;
    }
}

// Create another class that implements the interface
class Rectangle implements Shape {
    private $width;
    private $height;

    public function __construct($width, $height) {
        $this->width = $width;
        $this->height = $height;
    }

    public function getArea() {
        return $this->width * $this->height;
    }

    public function getPerimeter() {
```

```

        return 2 * ($this->width + $this->height);
    }
}

// Create instances of the classes and use the interface methods
$circle = new Circle(5);
echo "Circle Area: " . $circle->getArea() . "\n";
echo "Circle Perimeter: " . $circle->getPerimeter() . "\n";

$rectangle = new Rectangle(4, 6);
echo "Rectangle Area: " . $rectangle->getArea() . "\n";
echo "Rectangle Perimeter: " . $rectangle->getPerimeter() . "\n";
?>

```

## Overloading in php

```

class Calculator {
    public function calculate($a, $b = null, $c = null) {
        if ($c !== null) {
            // Three arguments provided, perform a three-operand calculation
            return $a + $b + $c;
        } elseif ($b !== null) {
            // Two arguments provided, perform a two-operand calculation
            return $a + $b;
        } else {
            // Only one argument provided, return the argument itself
            return $a;
        }
    }
}

$calculator = new Calculator();

// Example usages of method "calculate"
$result1 = $calculator->calculate(5);           // Returns 5
$result2 = $calculator->calculate(5, 3);         // Returns 8
$result3 = $calculator->calculate(5, 3, 2);       // Returns 10

echo "Result 1: $result1<br>";
echo "Result 2: $result2<br>";
echo "Result 3: $result3<br>";

```

## Another example

```
class Database {
    private $connection;

    public function __construct($host, $username, $password, $database) {
        // Establish a database connection (you would use your preferred
        database extension)
        $this->connection = new mysqli($host, $username, $password,
$database);
        if ($this->connection->connect_error) {
            die("Database Connection Error: " . $this->connection-
>connect_error);
        }
    }

    public function query($sql, $operation = 'SELECT', $params = []) {
        switch ($operation) {
            case 'SELECT':
                return $this->selectQuery($sql, $params);
            case 'INSERT':
                return $this->insertQuery($sql, $params);
            case 'UPDATE':
                return $this->updateQuery($sql, $params);
            default:
                throw new InvalidArgumentException("Unsupported database
operation: $operation");
        }
    }

    private function selectQuery($sql, $params) {
        // Implement SELECT query logic with prepared statement
        // Execute the query, bind parameters, fetch results, etc.
        return "SELECT result for query: $sql";
    }

    private function insertQuery($sql, $params) {
        // Implement INSERT query logic with prepared statement
        // Execute the query, bind parameters, handle insert IDs, etc.
        return "INSERT result for query: $sql";
    }

    private function updateQuery($sql, $params) {
        // Implement UPDATE query logic with prepared statement
    }
}
```

```
// Execute the query, bind parameters, handle affected rows, etc.
    return "UPDATE result for query: $sql";
}

public function close() {
    // Close the database connection
    $this->connection->close();
}
}

// Example usage
$db = new Database('localhost', 'username', 'password', 'mydb');

$selectResult = $db->query('SELECT * FROM users WHERE id = ?', 'SELECT',
[1]);
$insertResult = $db->query('INSERT INTO orders (product, quantity) VALUES (?, ?)', 'INSERT', ['Widget', 5]);
$updateResult = $db->query('UPDATE products SET price = ? WHERE id = ?', 'UPDATE', [19.99, 123]);

$db->close();

echo $selectResult . "<br>";
echo $insertResult . "<br>";
echo $updateResult . "<br>";
```

## Another example

```
class Calculator {
    public function calculate($operation, ...$operands) {
        switch ($operation) {
            case 'add':
                return $this->add(...$operands);
            case 'subtract':
                return $this->subtract(...$operands);
            case 'multiply':
                return $this->multiply(...$operands);
            default:
                throw new InvalidArgumentException("Unsupported operation: $operation");
        }
    }

    private function add($a, $b) {
        return $a + $b;
    }

    private function subtract($a, $b) {
        return $a - $b;
    }

    private function multiply($a, $b) {
        return $a * $b;
    }
}

$calculator = new Calculator();

// Example usages
$result1 = $calculator->calculate('add', 5, 3);
$result2 = $calculator->calculate('subtract', 10, 2);
$result3 = $calculator->calculate('multiply', 4, 6);

echo "Addition result: $result1<br>";
echo "Subtraction result: $result2<br>";
echo "Multiplication result: $result3<br>";
```

## Overriding example

```
class Content {
    protected $title;
    protected $content;

    public function __construct($title, $content) {
        $this->title = $title;
        $this->content = $content;
    }

    public function display() {
        // Generic content display
        echo "<h2>{$this->title}</h2>";
        echo "<p>{$this->content}</p>";
    }
}

class Article extends Content {
    private $author;

    public function __construct($title, $content, $author) {
        parent::__construct($title, $content);
        $this->author = $author;
    }

    public function display() {
        // Display article-specific content
        echo "<h2>{$this->title}</h2>";
        echo "<p>{$this->content}</p>";
        echo "<p>Author: {$this->author}</p>";
    }
}

class Image extends Content {
    private $imagePath;

    public function __construct($title, $content, $imagePath) {
        parent::__construct($title, $content);
        $this->imagePath = $imagePath;
    }

    public function display() {
        // Display image-specific content
    }
}
```

```

        echo "<h2>{$this->title}</h2>";
        echo "<img src='{$this->imagePath}' alt='{$this->title}'><br>";
        echo "<p>{$this->content}</p>";
    }
}

// Example usage
$article = new Article("Sample Article", "This is a sample article content.", "John Doe");
$image = new Image("Sample Image", "This is a sample image description.", "image.jpg");

$article->display();
$image->display();

```

## Another Example

```

class DatabaseConnection {
    protected $host;
    protected $username;
    protected $password;
    protected $database;

    public function __construct($host, $username, $password, $database) {
        $this->host = $host;
        $this->username = $username;
        $this->password = $password;
        $this->database = $database;
    }

    public function connect() {
        // Generic database connection logic
        $this->connection = new mysqli($this->host, $this->username, $this->password, $this->database);
        if ($this->connection->connect_error) {
            die("Database Connection Error: " . $this->connection->connect_error);
        }
        echo "Connected to the database using generic driver<br>";
    }
}

```

```

class MySQLDatabaseConnection extends DatabaseConnection {
    public function connect() {
        // MySQL-specific database connection logic
        parent::connect(); // Call the parent class's connect method first
        echo "Connected to the MySQL database using MySQLi driver<br>";
    }
}

// Example usage
$mysqlDb = new MySQLDatabaseConnection('localhost', 'mysqluser', 'mysqlpass',
'mysqldb');
$mysqlDb->connect();

```

## Overriding very sample example

```

class Animal {
    public function speak() {
        echo "Animal speaks!";
    }
}

class Dog extends Animal {
    public function speak() {
        echo "Dog barks!";
    }
}

$animal = new Animal();
$dog = new Dog();

$animal->speak(); // Output: Animal speaks!
$dog->speak();   // Output: Dog barks!

```

## Error handling in php

```
// Define a custom error handler function
function customErrorHandler($errno, $errstr, $errfile, $errline) {
    echo "Custom Error Handler: [$errno] $errstr\n";
    echo "Error on line $errline in $errfile\n";
}

// Set the custom error handler
set_error_handler("customErrorHandler");

try {
    // Simulate a division by zero error
    $result = 10 / 0;
} catch (Exception $e) {
    // This block will not be executed for PHP errors, only for exceptions
    echo "Caught Exception: " . $e->getMessage() . "\n";
} finally {
    // This block will always be executed, whether there was an error or not
    echo "Finally block executed\n";
}

// Attempt to open a non-existent file
$file = fopen("nonexistent.txt", "r");
if ($file === false) {
    echo "Error: Failed to open the file.\n";
}

// Trigger a user-defined error
trigger_error("This is a custom error message.", E_USER_WARNING);

// Attempt to call a non-existent function
nonExistentFunction();

// Restore the default error handler
restore_error_handler();
```

## Error handling example

```
class DatabaseConnection {
    protected $host;
    protected $username;
    protected $password;
    protected $database;
    protected $connection;

    public function __construct($host, $username, $password, $database) {
        $this->host = $host;
        $this->username = $username;
        $this->password = $password;
        $this->database = $database;

        // Attempt to establish a database connection
        $this->connect();
    }

    private function connect() {
        // Establish the database connection
        $this->connection = new mysqli($this->host, $this->username, $this-
>password, $this->database);

        // Check for connection errors
        if ($this->connection->connect_error) {
            throw new Exception("Database Connection Error: " . $this-
>connection->connect_error);
        }
    }

    public function query($sql) {
        // Perform a database query
        $result = $this->connection->query($sql);

        // Check for query errors
        if ($result === false) {
            throw new Exception("Database Query Error: " . $this->connection-
>error);
        }

        return $result;
    }
}
```

```

    public function close() {
        // Close the database connection
        $this->connection->close();
    }
}

// Example usage with error handling
try {
    $db = new DatabaseConnection('localhost', 'username', 'password',
'mydb');

    // Example query with an error (table doesn't exist)
    $result = $db->query('SELECT * FROM non_existing_table');

    // Close the database connection
    $db->close();
} catch (Exception $e) {
    // Handle database-related exceptions
    echo "Caught exception: " . $e->getMessage();
}

```

## Another Example

```

class DatabaseConnection {
    protected $host;
    protected $username;
    protected $password;
    protected $database;
    protected $connection;

    public function __construct($host, $username, $password, $database) {
        $this->host = $host;
        $this->username = $username;
        $this->password = $password;
        $this->database = $database;

        // Attempt to establish a database connection
        $this->connect();
    }

    private function connect() {
        // Establish the database connection
        $this->connection = new mysqli($this->host, $this->username, $this-
>password, $this->database);
    }
}

```

```

    // Check for connection errors
    if ($this->connection->connect_error) {
        throw new Exception("Database Connection Error: " . $this-
>connection->connect_error);
    }
}

public function executeQuery($sql) {
    // Execute a database query and return the result
    return $this->connection->query($sql);
}

public function getLastInsertId() {
    // Get the last auto-generated ID from an INSERT query
    return $this->connection->insert_id;
}

public function close() {
    // Close the database connection
    $this->connection->close();
}
}

class User {
    private $db;

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

    public function addUser($username, $email, $password) {
        $username = $this->db->connection->real_escape_string($username);
        $email = $this->db->connection->real_escape_string($email);
        $password = $this->db->connection->real_escape_string($password);

        $sql = "INSERT INTO users (username, email, password) VALUES
('{$username}', '{$email}', '{$password}')";

        $result = $this->db->executeQuery($sql);

        if ($result) {
            echo "User added successfully with ID: " . $this->db-
>getLastInsertId();
        } else {
    }
}

```

```

        throw new Exception("Error adding user: " . $this->db-
>connection->error);
    }
}
}

// Example usage
try {
    $db = new DatabaseConnection('localhost', 'username', 'password',
'mydb');
    $user = new User($db);

    $user->addUser('john_doe', 'john@example.com', 'password123');

    // Close the database connection
    $db->close();
} catch (Exception $e) {
    // Handle exceptions, e.g., display an error message
    echo "Caught exception: " . $e->getMessage();
}

```

## Exception handling in php

```

class DatabaseConnectionException extends Exception {
    public function errorMessage() {
        return "Database Connection Error: " . $this->getMessage();
    }
}

class Database {
    private $connection;

    public function __construct($host, $username, $password, $database) {
        try {
            $this->connection = new mysqli($host, $username, $password,
$database);

            if ($this->connection->connect_error) {
                throw new DatabaseConnectionException($this->connection-
>connect_error);
            }
        } catch (DatabaseConnectionException $e) {
            // Handle the exception
        }
    }
}

```

```

        die($e->errorMessage());
    }
}

public function query($sql) {
    try {
        if (!$this->connection) {
            throw new DatabaseConnectionException("No database
connection.");
        }

        $result = $this->connection->query($sql);

        if ($result === false) {
            throw new Exception("Database query error: " . $this-
>connection->error);
        }

        return $result;
    } catch (Exception $e) {
        // Handle the exception
        die("Error: " . $e->getMessage());
    }
}

public function close() {
    if ($this->connection) {
        $this->connection->close();
    }
}

// Example usage
try {
    $db = new Database("localhost", "username", "password", "mydb");
    $result = $db->query("SELECT * FROM non_existing_table");
    $db->close();
} catch (Exception $e) {
    // Handle any exception thrown during database connection or query
    echo "Caught exception: " . $e->getMessage();
}

```

# Google login API

Google Sign-In with PHP using Google API Client Library:

## 1. Create a Project in Google Cloud Console:

- Go to the [Google Cloud Console](#).
- Create a new project or select an existing one.
- Enable the "Google+ API" for your project.

## 2. Create OAuth 2.0 Credentials:

- In the Cloud Console, navigate to "APIs & Services" > "Credentials."
- Click on "Create Credentials" and choose "OAuth client ID."
- Select "Web application" as the application type.
- Configure the authorized JavaScript origins and redirect URIs. For local development, you can use "<http://localhost>" as the origin.
- Click "Create" to generate your OAuth client ID and client secret.

## 3. Install Google API Client Library for PHP:

- You can install the Google API Client Library for PHP using Composer:

```
composer require google/apiclient:^2.0
```

## 4. Set Up the PHP Code:

- Create a PHP file (e.g., `google_login.php`) and add the following code:

```
<?php
require_once 'vendor/autoload.php';

$client = new Google_Client();
$client->setClientId('YOUR_CLIENT_ID');
$client->setClientSecret('YOUR_CLIENT_SECRET');
$client->setRedirectUri('YOUR_REDIRECT_URI');
$client->addScope('email');
$client->addScope('profile');

if (isset($_GET['code'])) {
    $token = $client->fetchAccessTokenWithAuthCode($_GET['code']);
    $client->setAccessToken($token);

    $oauth2Service = new Google_Service_Oauth2($client);
    $userInfo = $oauth2Service->userinfo->get();

    // Now, you can use $userInfo to access user data (e.g., $userInfo->getEmail(), $userInfo->getGivenName()).
```

```
echo "Welcome, " . $userInfo->getName();
} else {
    $authUrl = $client->createAuthUrl();
    echo "<a href='$authUrl'>Login with Google</a>";
}
?>
```

**For insert data in MySQL first we have to create a table in data base.**

Here we using 3 file for insert data in MySQL:

```
CREATE TABLE `employee` (
    `userid` int(8) NOT NULL AUTO_INCREMENT,
    `first_name` varchar(55) NOT NULL,
    `last_name` varchar(55) NOT NULL,
    `city_name` varchar(55) NOT NULL,
    `email` varchar(50) NOT NULL,
    `datetime` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

**index.php:**for getting the values from the user

```
<!DOCTYPE html>
<html>
    <body>
        <form method="post" action="process.php">
            First name:<br>
            <input type="text" name="first_name">
            <br>
            Last name:<br>
            <input type="text" name="last_name">
            <br>
            City name:<br>
            <input type="text" name="city_name">
            <br>
            Email Id:<br>
            <input type="email" name="email">
            <br><br>
            <input type="submit" name="submit" value="submit">
        </form>
    </body>
</html>
```

**Crud.php:**For connecting data base and function

```
<?php
    class Crud
    {
        private $servername = "localhost";
        private $username   = "root";
        private $password   = "";
        private $dbname     = "oops_db";
        public $con;
        public $employeeTable = "employee";
        public function __construct()
        {
            try {
                $this->con = new mysqli($this->servername, $this->username,
                $this->password, $this->dbname);
            } catch (Exception $e) {
                echo $e->getMessage();
            }
        }
        public function save($first_name, $last_name, $city_name, $email,
        $insertdate)
        {
            $sql = "INSERT INTO $this-
>employeeTable(first_name,last_name,city_name,email,datetime)
VALUES('$first_name', '$last_name','$city_name','$email','$insertdate')";
            $query = $this->con->query($sql);
            if ($query) {
                return true;
            }else{
                return false;
            }
        }
    }
?>
```

**process.php:A PHP file that process the request**

```
<?php
    include_once("Crud.php");
    $insertdata=new Crud();
    if(isset($_POST['submit']))
    {
        $first_name = $_POST['first_name'];
        $last_name = $_POST['last_name'];
        $city_name = $_POST['city_name'];
        $email = $_POST['email'];
        date_default_timezone_set("Asia/Calcutta");
        $insertdate = date("Y-m-d H:i:s");
        $sql=$insertdata->save($first_name,
        $last_name,$city_name,$email,$insertdate);
        if($sql)
        {
            echo "Data inserted successfully !";
        }
        else
        {
            echo "Data inserted error !";
        }
    }
?>
```

**For retrieve data from MySQL first we have to create a table in data base.**

```
CREATE TABLE `employee` (
    `userid` int(8) NOT NULL AUTO_INCREMENT,
    `first_name` varchar(55) NOT NULL,
    `last_name` varchar(55) NOT NULL,
    `city_name` varchar(55) NOT NULL,
    `email` varchar(50) NOT NULL,
    `datetime` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
crud.php
<?php
    class Crud
    {
        private $servername = "localhost";
        private $username = "root";
        private $password = "";
        private $dbname = "oops_db";
        public $con;
        public $employeeTable = "employee";
        public function __construct()
        {
            try {
                $this->con = new mysqli($this->servername, $this->username,
                $this->password, $this->dbname);
            } catch (Exception $e) {
                echo $e->getMessage();
            }
        }
        /* Fetch employee records for show listing */
        public function displayRecord()
        {
            $sql = "SELECT * FROM $this->employeeTable";
            $query = $this->con->query($sql);
            $data = array();
            if ($query->num_rows > 0) {
                while ($row = $query->fetch_assoc()) {
                    $data[] = $row;
                }
                return $data;
            }else{
                return false;
            }
        }
    }
?>
```

```
view.php
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Bootstrap Example</title>
    <meta charset="utf-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</head>
<body>
<?php
include_once("Crud.php");
$viewdata=new Crud();
$customers = $viewdata->displayRecord();
$output ="";
$output .="<table class='table table-striped table-hover'>
<thead>
<tr>
<th>Id</th>
<th>First Name</th>
<th>Last Name</th>
<th>City Name</th>
<th>Email</th>

</tr>
</thead>
<tbody>";
foreach ($customers as $customer) {
$output.= "<tr>
<td>".$customer['userid']."'</td>
<td>".$customer['first_name']."'</td>
<td>".$customer['last_name']."'</td>
<td>".$customer['city_name']."'</td>
<td>".$customer['email']."'</td>

</tr>";
}
$output .= "</tbody>
</table>";
echo $output;
?>
</body></html>
```

## PHP oops update data

*crud.php*

```
<?php

    class Crud
    {
        private $servername = "localhost";
        private $username   = "root";
        private $password   = "";
        private $dbname     = "oops_db";
        public $con;
        public $employeeTable = "employee";
        public function __construct()
        {
            try {
                $this->con = new mysqli($this->servername, $this->username, $this->password,
                $this->dbname);
            } catch (Exception $e) {
                echo $e->getMessage();
            }
        }

        public function update($userid,$first_name,
        $last_name,$city_name,$email,$insertdate)
        {
            echo $sql = "UPDATE $this->employeeTable SET
            first_name='$first_name',last_name='$last_name',city_name='$city_name',email='$em
            ail' where userid=$userid";
            $query = $this->con->query($sql);
            if ($query) {
                return true;
            }else{
                return false;
            }
        }

    }
?>
```

## *update.php*

```
<?php
    include_once("Crud.php");
    $userid=$_GET['userid'];
    $viewdata=new Crud();
    $employee = $viewdata->displayRecordbyid($userid);
?
<!DOCTYPE html>
<html>
<body>
<form method="post" action="update_process.php">
First name:<br>
<input type="text" name="first_name" value="<?php echo
$employee['first_name'];?>">
<br>
Last name:<br>
<input type="text" name="last_name" value="<?php echo
$employee['last_name'];?>">
<br>
City name:<br>
<input type="text" name="city_name" value="<?php echo
$employee['city_name'];?>">
<br>
Email Id:<br>
<input type="email" name="email" value="<?php echo
$employee['email'];?>">
<br><br>
<input type="hidden" name="userid" value="<?php echo
$employee['userid'];?>">
<input type="submit" name="submit" value="submit">
</form>
</body>
</html>
```

## *update\_process.php*

```
<?php
    include_once("Crud.php");
    $insertdata=new Crud();
    if(isset($_POST['submit']))
    {
        $userid = $_POST['userid'];
        $first_name = $_POST['first_name'];
        $last_name = $_POST['last_name'];
    }
}
```

```

$city_name = $_POST['city_name'];
$email = $_POST['email'];
date_default_timezone_set("Asia/Calcutta");
$insertdate = date("Y-m-d H:i:s");
$sql=$insertdata->update($userid,$first_name,
$last_name,$city_name,$email,$insertdate);
if($sql)
{
echo "Data inserted successfully !";
}
else
{
echo "Data inserted error !";
}
}
?>

```

**For delete data from MySQL first we have to create a table in data base.**

*delete\_process.php:For Delete process in backend*

```

CREATE TABLE `employee` (
  `userid` int(8) NOT NULL AUTO_INCREMENT,
  `first_name` varchar(55) NOT NULL,
  `last_name` varchar(55) NOT NULL,
  `city_name` varchar(55) NOT NULL,
  `email` varchar(50) NOT NULL,
  `datetime` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

*crud.php*

```

<?php
class Crud
{
    private $servername = "localhost";
    private $username = "root";
    private $password = "";

```

```

        private $dbname = "oops_db";
    public $con;
    public $employeeTable = "employee";
        public function __construct()
    {
        try {
            $this->con = new mysqli($this-
>servername, $this->username, $this->password, $this->dbname);
        } catch (Exception $e) {
            echo $e->getMessage();
        }
    }
/* Fetch employee records for show listing */
    public function displayRecord()
    {
        $sql = "SELECT * FROM $this->employeeTable";
        $query = $this->con->query($sql);
        $data = array();
        if ($query->num_rows > 0) {
            while ($row = $query->fetch_assoc()) {
                $data[] = $row;
            }
            return $data;
        }else{
            return false;
        }
    }
    public function deleteRecord($id)
    {
        $sql = "DELETE from $this->employeeTable
where userid=$id";
        $query = $this->con->query($sql);
        $data = array();
        if ($query) {
            return true;
        }else{
            return false;
        }
    }
}
?>
\
```

## view.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Bootstrap Example</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <script
    src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script
    src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"></script>
    <script
    src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</head>
<body>
<?php
include_once("Crud.php");
$viewdata=new Crud();
$customers = $viewdata->displayRecord();
$output = "";
$output .="<table class='table table-striped table-hover'>
    <thead>
        <tr>
            <th>Id</th>
            <th>First Name</th>
            <th>Last Name</th>
            <th>City Name</th>
            <th>Email</th>
            <th>Action</th>
        </tr>
    </thead>
    <tbody>";
foreach ($customers as $customer) {
    $output.= "<tr>
        <td>".$customer['userid']."'</td>
        <td>".$customer['first_name']."'</td>
        <td>".$customer['last_name']."'</td>
        <td>".$customer['city_name']."'</td>
        <td>".$customer['email']."'</td>
```

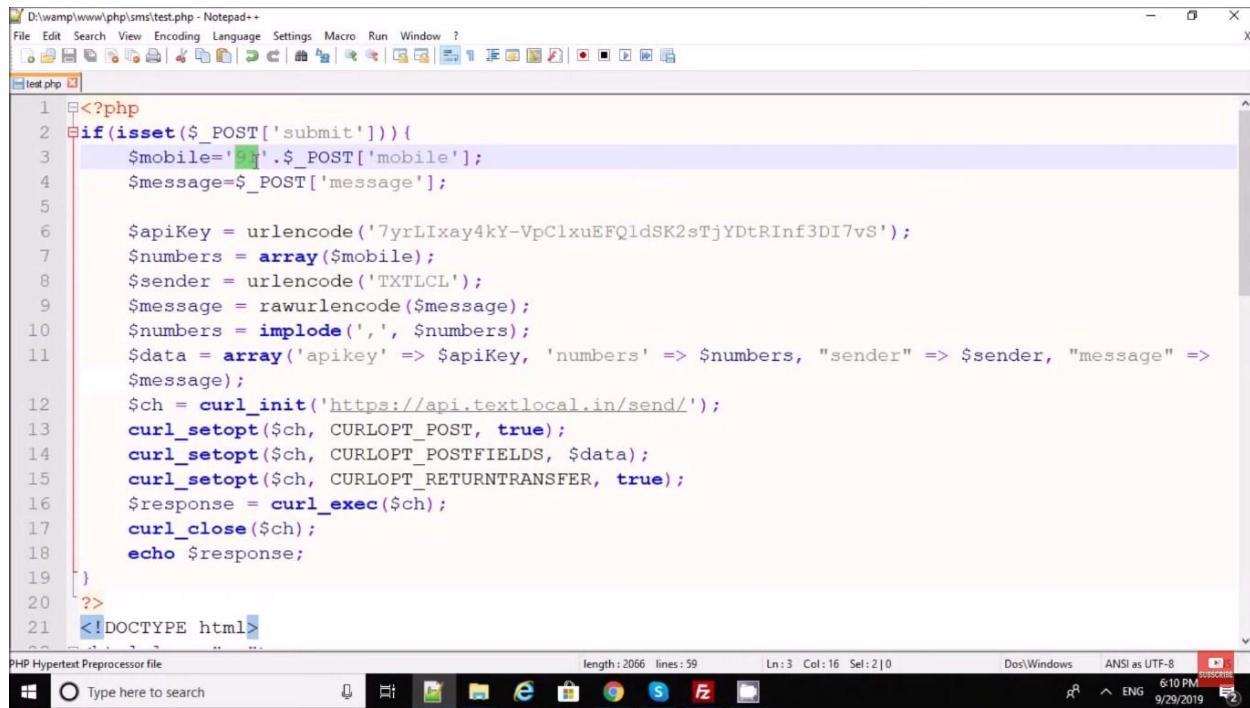
```
        <td><a href='delete_process.php?id='.$customer['userid']."'>Delete</a></td>
    </tr>";
}
$output .= "</tbody>
</table>";
echo $output;
?>
</body>
</html>
```

*crud.php*

```
<?php

include_once("Crud.php");
$insertdata=new Crud();
$id = $_GET['id'];
$sql=$insertdata->deleteRecord($id);
if($sql)
{
    echo "Data deleted successfully !";
}
else
{
    echo "Error ! Please try again";
}
```

# How to send message or sms in PHP



A screenshot of the Notepad+ application window. The title bar reads "D:\wamp\www\php\sms\test.php - Notepad+". The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Macro, Run, Window, and Help. Below the menu is a toolbar with various icons. The main text area contains PHP code for sending an SMS using cURL. The code includes variables for mobile number, message, and API key, and demonstrates the use of curl\_init, curl\_setopt, curl\_exec, and curl\_close functions. The code ends with an HTML DOCTYPE declaration. The status bar at the bottom shows "PHP Hypertext Preprocessor file", "length : 2066 lines : 59", "Ln : 3 Col : 16 Sel : 2 | 0", "Dos\Windows", "ANSI as UTF-8", and a system tray with icons for battery, network, and date/time.

```
1 <?php
2 if(isset($_POST['submit'])){
3     $mobile='91' . $_POST['mobile'];
4     $message=$_POST['message'];
5
6     $apiKey = urlencode('7yrLIXay4kY-VpClxuEFQ1dSK2sTjYDtRInf3DI7vS');
7     $numbers = array($mobile);
8     $sender = urlencode('TXTLCL');
9     $message = rawurlencode($message);
10    $numbers = implode(',', $numbers);
11    $data = array('apikey' => $apiKey, 'numbers' => $numbers, "sender" => $sender, "message" =>
12    $message);
13    $ch = curl_init('https://api.textlocal.in/send/');
14    curl_setopt($ch, CURLOPT_POST, true);
15    curl_setopt($ch, CURLOPT_POSTFIELDS, $data);
16    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
17    $response = curl_exec($ch);
18    curl_close($ch);
19    echo $response;
20 }
21 ?>
22 <!DOCTYPE html>
```

*index.php*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Send Message</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
    <script
    src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.m
in.js"></script>
    <script
    src="https://cdn.jsdelivr.net/npm/popper.js@1.14.3/umd/
popper.min.js"></script>
    <script
    src="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/js/bootstra
p.min.js"></script>
```

```
</head>
<body>

<div class="container">
    <h2>Send Message</h2>
    <p id="message"></p>
        <div class="form-group">
            <label for="email">Sender ID:</label>
            <input type="email" class="form-control" id="senderid"
placeholder="Enter Sender ID" name="senderid">
        </div>
        <div class="form-group">
            <label for="pwd">Mobile No.:</label>
            <input type="text" class="form-control" id="mobile"
placeholder="Enter Mobile No" name="mobile">
        </div>
        <div class="form-group">
            <label for="comment">Message:</label>
            <textarea class="form-control" rows="5" id="message"
name="Message" placeholder="Write Your Message
Here...."></textarea>
        </div>
        <button type="submit" class="btn btn-primary" id="send">Send
Message</button>

</div>
<script>
    $('#send').click(function() {
        $.ajax({
            type: "POST",
            url: "process.php",
            data: {
                senderid:
                    $('#senderid').val(),
                mobile:
                    $('#mobile').val(),
                message:
                    $('#message').val()
            },
            success: function(data) {
                var objJSON =
                    JSON.parse(data);
                $('#message').html(objJSON.message);
            }
        });
    });
</script>
```

```
</script>
</body>
</html>
```

## process.php

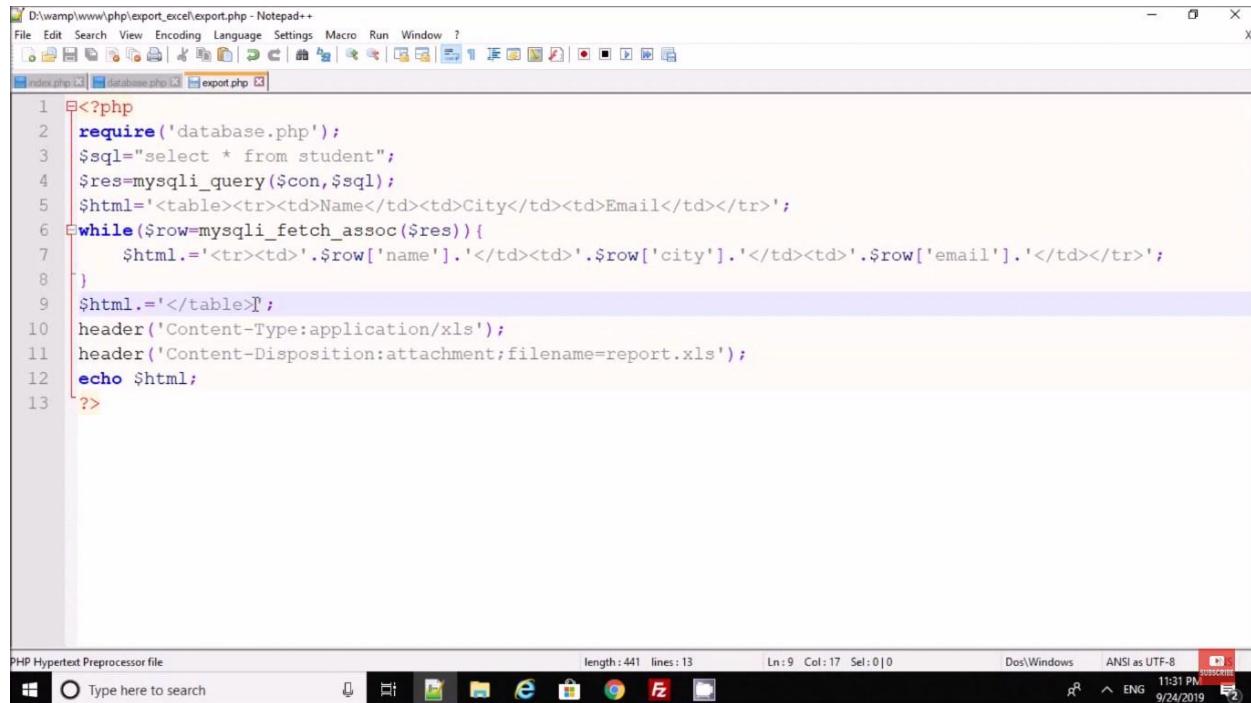
```
<?php
$senderID = $_POST['senderid'];
$mobile = $_POST['mobile'];
$message = $_POST['message'];
/*MESSAGE CODE*/
/* Your authentication key */
$authKey = "234557688009432n";
/* Multiple mobiles numbers separated by comma */
$mobileNumber = $mobile;
/* Sender ID,While using route4 sender id should be 6
characters long. */
$senderId = $senderID;
/* Your message to send, Add URL encoding here. */

$message = urlencode($message);
/* Define route */
$route = "route=4";
/* Prepare you post parameters */
$postData = array(
    'authkey' => $authKey,
    'mobiles' => $mobileNumber,
    'message' => $message,
    'sender' => $senderId,
    'route' => $route
);
/* API URL*/
$url="https://www.fast2sms.com/dev/bulk";
/* init the resource */
$ch = curl_init();
curl_setopt_array($ch, array(
    CURLOPT_URL => $url,
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_POST => true,
    CURLOPT_POSTFIELDS => $postData
    /*,CURLOPT_FOLLOWLOCATION => true */
));
/* Ignore SSL certificate verification */
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
curl_setopt($ch, CURLOPT_SSL_VERIFYPeer, 0);
/* get response */
$output = curl_exec($ch);
```

```

/* Print error if any */
if(curl_errno($ch))
{
    echo 'error:' . curl_error($ch);
}
else{
    $array = array(
        "message"      => $output
    );
}
curl_close($ch);
//MESSAGE CODE END
print json_encode($array);
?>

```



The screenshot shows a Notepad++ window with the following PHP code:

```

1 <?php
2 require('database.php');
3 $sql="select * from student";
4 $res=mysqli_query($con,$sql);
5 $html='<table><tr><td>Name</td><td>City</td><td>Email</td></tr>';
6 while($row=mysqli_fetch_assoc($res)){
7     $html.='|  |  |  |
| --- | --- | --- |
| ' . $row['name'] . ' | ' . $row['city'] . ' | ' . $row['email'] . ' |
';
8 }
$html.='</table>';
header('Content-Type:application/xls');
header('Content-Disposition:attachment;filename=report.xls');
echo $html;
?>

```

The code is intended to generate an Excel file by outputting an HTML table structure. It includes file headers and a closing PHP tag.

## PHP oops login

```
CREATE TABLE `users` (
  `id` int(11) NOT NULL,
  `username` varchar(30) NOT NULL AUTO_INCREMENT,
  `password` varchar(30) NOT NULL,
  `fname` varchar(100) NOT NULL,
  PRIMARY KEY(`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `users` (`id`, `username`, `password`, `fname`)
VALUES
(1, 'neovic', 'devierte', 'Neovic Devierte'),
(2, 'gemalyn', 'cepe', 'Gemalyn Cepe');
```

### *connection.php*

```
<?php
class DbConnection{

    private $host = 'localhost';
    private $username = 'root';
    private $password = '';
    private $database = 'test';

    protected $connection;

    public function __construct() {
        if (!isset($this->connection)) {

            $this->connection = new mysqli($this->host, $this->username, $this->password, $this->database);

            if (!$this->connection) {
                echo 'Cannot connect to database server';
                exit;
            }
        }

        return $this->connection;
    }
}
?>
```

## *user.php*

```
<?php
include_once('DbConnection.php');

class User extends DbConnection{

    public function __construct() {
        parent::__construct();
    }

    public function check_login($username, $password) {
        $sql = "SELECT * FROM users WHERE username = '$username'
AND password = '$password'";
        $query = $this->connection->query($sql);

        if($query->num_rows > 0){
            $row = $query->fetch_array();
            return $row['id'];
        }
        else{
            return false;
        }
    }

    public function details($sql) {
        $query = $this->connection->query($sql);

        $row = $query->fetch_array();

        return $row;
    }

    public function escape_string($value) {
        return $this->connection->real_escape_string($value);
    }
}
```

## *index.php*

```
<?php

    session_start();

    if(isset($_SESSION['user'])){
        header('location:home.php');
    }
?>
<!DOCTYPE html>
<html>
<head>
    <title>PHP oops login</title>
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
<div class="container">
    <h1 class="page-header text-center">PHP oops login</h1>
    <div class="row">
        <div class="col-md-4 col-md-offset-4">
            <div class="login-panel panel panel-primary">
                <div class="panel-heading">
                    <h3 class="panel-title"><span
class="glyphicon glyphicon-lock"></span> Login
                </h3>
                </div>
                <div class="panel-body">
                    <form method="POST" action="login.php">
                        <fieldset>
                            <div class="form-group">
                                <input class="form-
control" placeholder="Username" type="text" name="username"
autofocus required>
                            </div>
                            <div class="form-group">
                                <input class="form-
control" placeholder="Password" type="password" name="password"
required>
                            </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
```

```

                                <button type="submit"
name="login" class="btn btn-lg btn-primary btn-block"><span>


```

## *login.php*

```

<?php
session_start();
include_once('User.php');

$user = new User();

if(isset($_POST['login'])) {
    $username = $user->escape_string($_POST['username']);
    $password = $user->escape_string($_POST['password']);

    $auth = $user->check_login($username, $password);

```

```

        if (!$auth) {
            $_SESSION['message'] = 'Invalid username or
password';
            header('location:index.php');
        }
        else{
            $_SESSION['user'] = $auth;
            header('location:home.php');
        }
    }
else{
    $_SESSION['message'] = 'You need to login first';
    header('location:index.php');
}
?>

```

## *home.php*

```

<?php
session_start();

if (!isset($_SESSION['user']) || (trim($_SESSION['user']) ==
'')) {
    header('location:index.php');
}

include_once('User.php');

$user = new User();

$sql = "SELECT * FROM users WHERE id = '". $_SESSION['user']."' ";
$row = $user->details($sql);

?>
<!DOCTYPE html>
<html>
<head>
    <title>PHP OOPS Login</title>
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>

```

```

<div class="container">
    <h1 class="page-header text-center">PHP OOPS Login</h1>
    <div class="row">
        <div class="col-md-4 col-md-offset-4">
            <h2>Welcome to Homepage </h2>
            <h4>User Info: </h4>
            <p>Name: <?php echo $row['fname']; ?></p>
            <p>Username: <?php echo $row['username']; ?>
        </p>
        <p>Password: <?php echo $row['password']; ?>
    </p>
    <a href="logout.php" class="btn btn-danger"><span class="glyphicon glyphicon-log-out"></span> Logout</a>
</div>
</body>
</html>

```

*logout.php*

```

<?php
    session_start();
    session_destroy();

    header('location:index.php');
?>

```

## PHP oops login

```

CREATE TABLE `tblusers` (
    `id` int(11) NOT NULL,
    `FullName` varchar(120) DEFAULT NULL,
    `Username` varchar(120) DEFAULT NULL,
    `UserEmail` varchar(200) DEFAULT NULL,
    `Password` varchar(250) DEFAULT NULL,
    `RegDate` timestamp NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

## function.php

```
<?php
define('DB_SERVER', 'localhost');
define('DB_USER', 'root');
define('DB_PASS', '');
define('DB_NAME', 'userdb');
class DB_con
{
    function __construct()
    {
        $con = mysqli_connect(DB_SERVER, DB_USER, DB_PASS, DB_NAME);
        $this->dbh=$con;

        if (mysqli_connect_errno())
        {
            echo "Failed to connect to MySQL: " . mysqli_connect_error();
        }
    }

    public function usernameavailbilty($uname) {
        $result =mysqli_query($this->dbh,"SELECT Username FROM tblusers
        WHERE Username='$uname'");
        return $result;
    }

    public function registration($fname,$uname,$uemail,$password)
    {
        $ret=mysqli_query($this->dbh,"insert into
        tblusers(FullName,Username,UserEmail,Password)
        values('$fname','$uname','$uemail','$password')");
        return $ret;
    }

    public function signin($uname,$pasword)
    {
        $result=mysqli_query($this->dbh,"select id,FullName from
        tblusers where Username='$uname' and Password='$pasword'");
        return $result;
    }
}
```

```
?>
```

Like index.php may be

```
<?php

include_once('function.php');

$userdata=new DB_con();
if(isset($_POST['submit']))
{

$fname=$_POST['fullname'];
$uname=$_POST['username'];
$uemail=$_POST['email'];
$password=md5($_POST['password']);

$sql=$userdata->registration($fname,$uname,$uemail,$password);
if($sql)
{

echo "<script>alert('Registration successfull.');//</script>";
echo "<script>window.location.href='signin.php'</script>";
}
else
{

echo "<script>alert('Something went wrong. Please try
again');//</script>";
echo "<script>window.location.href='signin.php'</script>";
}
}

?>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<!-- This file has been downloaded from Bootsnipp.com. Enjoy! -->
<title>User Registration using PHP OOPs Concept</title>
<meta name="viewport" content="width=device-width, initial-
scale=1">
<link href="assests/style.css" rel="stylesheet">
<script src="assests/jquery-1.11.1.min.js"></script>
<script src="assests/bootstrap.min.js"></script>
```

```
<script>
function checkusername(va) {
$.ajax({
type: "POST",
url: "check_availability.php",
data:'username='+va,
success: function(data){
$("#usernameavailblty").html(data);
}
}) ;

}
</script>
</head>
<body>
<form class="form-horizontal" action=' ' method="POST">
<fieldset>
<div id="legend" align="center">
<legend class="">User Registration using PHP OOPS
Concept</legend>
</div>

<div class="control-group">
<!-- Fullname -->
<label class="control-label" for="username">Fullname</label>
<div class="controls">
<input type="text" id="username" name="fullname" placeholder=""
class="input-xlarge" required="true">
</div>
</div>

<div class="control-group">
<!-- Username -->
<label class="control-label" for="username">Username</label>
<div class="controls">
<input type="text" id="username" name="username"
onblur="checkusername(this.value)" class="input-xlarge"
required="true">
<span id="usernameavailblty"></span>
</div>
</div>

<div class="control-group">
<!-- E-mail -->
<label class="control-label" for="email">E-mail</label>
<div class="controls">
```

```

<input type="email" id="email" name="email" placeholder="" class="input-xlarge" required="true">
</div>
</div>

<div class="control-group">
<!-- Password-->
<label class="control-label" for="password">Password</label>
<div class="controls">
<input type="password" id="password" name="password" placeholder="" class="input-xlarge" required="true">
</div>
</div>

<div class="control-group">
<!-- Button -->
<div class="controls">
<button class="btn btn-success" type="submit" id="submit" name="submit">Register</button>
</div>
</div>

<div class="control-group">
<div class="controls">
Already registered <a href="#">Signin</a>
</div>
</div>

</fieldset>
</form>
<script >
</script>
</body>
</html>

```

### *check\_availability.php*

```

<?php

include_once('function.php');

$uusername=new DB_con();

$uname= $_POST["username"];

$sql=$uusername->usernameavailblty($uname);

```

```

$num=mysqli_num_rows($sql);
if($num > 0)
{
echo "<span style='color:red'> Username already associated with
another account .</span>";
echo "<script>$('#submit').prop('disabled',true);</script>";
} else{

echo "<span style='color:green'> Username available for
Registration .</span>";
echo "<script>$('#submit').prop('disabled',false);</script>";
} ?>

```

## *signin.php*

```

<?php

include_once('function.php');

$userdata=new DB_con();
if(isset($_POST['submit']))
{

$fname=$_POST['fullname'];
$uname=$_POST['username'];
$uemail=$_POST['email'];
$password=md5($_POST['password']);

$sql=$userdata->registration($fname,$uname,$uemail,$password);
if($sql)
{

echo "<script>alert('Registration successfull.');//</script>";
echo "<script>window.location.href='signin.php'</script>";
}
else
{

echo "<script>alert('Something went wrong. Please try
again');//</script>";
echo "<script>window.location.href='signin.php'</script>";
}
}
?>
<!DOCTYPE html>

```

```
<html lang="en">
<head>
<meta charset="utf-8">
<title>User Registration using PHP OOPS Concept</title>
<meta name="viewport" content="width=device-width, initial-
scale=1">
<link href="assests/style.css" rel="stylesheet">
<script src="assests/jquery-1.11.1.min.js"></script>
<script src="assests/bootstrap.min.js"></script>
<script>
function checkusername(va) {
$.ajax({
type: "POST",
url: "check_availability.php",
data: 'username='+va,
success: function(data){
$("#usernameavailblty").html(data);
}
});
}

</script>
</head>
<body>
<form class="form-horizontal" action=' ' method="POST">
<fieldset>
<div id="legend" align="center">
<legend class="">User Registration using PHP OOPS
Concept</legend>
</div>

<div class="control-group">
<!-- Fullname -->
<label class="control-label" for="username">Fullname</label>
<div class="controls">
<input type="text" id="username" name="fullname" placeholder="">
</div>
</div>

<div class="control-group">
<!-- Username -->
<label class="control-label" for="username">Username</label>
<div class="controls">
<input type="text" id="username" name="username"
onblur="checkusername(this.value)" class="input-xlarge"
required="true">
```

```
<span id="usernameavailblty"></span>
</div>
</div>

<div class="control-group">
<!-- E-mail -->
<label class="control-label" for="email">E-mail</label>
<div class="controls">
<input type="email" id="email" name="email" placeholder="" class="input-xlarge" required="true">
</div>
</div>

<div class="control-group">
<!-- Password-->
<label class="control-label" for="password">Password</label>
<div class="controls">
<input type="password" id="password" name="password" placeholder="" class="input-xlarge" required="true">
</div>
</div>

<div class="control-group">
<!-- Button -->
<div class="controls">
<button class="btn btn-success" type="submit" id="submit" name="submit">Register</button>
</div>
</div>

<div class="control-group">
<div class="controls">
Already registered <a href="#">Signin</a>
</div>
</div>

</fieldset>
</form>
<script >
</script>
</body>
</html>
```

## *welcome.php*

```
<?php
session_start();
if(strlen($_SESSION['uid'])=="")
{
header('location:logout.php');
} else {
?><!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>User Registration using PHP OOPs Concept</title>
<meta name="viewport" content="width=device-width, initial-
scale=1">
<link href="assests/style.css" rel="stylesheet">
<script src="assests/jquery-1.11.1.min.js"></script>
<script src="assests/bootstrap.min.js"></script>
</head>
<body>
<form class="form-horizontal" action=' ' method="POST">
<fieldset>
<div id="legend">
<legend class="" align="center">Welcome Back : <?php echo
$_SESSION['fname']; ?></legend>
</div>

<div class="control-group" align="center">
<!-- Button -->
<div class="controls">
<a href="logout.php" class="btn btn-success" type="submit"
name="signin">Logout</a>
</div>
</div>

</fieldset>
</form>
<script >
</script>
</body>
</html>
<?php } ?>
```

## *logout.php*

```
<?php
```

```
session_start();
session_destroy();
header('location:signin.php');
?>
```

## PHP Caching

```
<?php
//Caching is a temporary storage location, to access data more quickly. Caching is used where the
original data is expensive to fetch.
$start=microtime(true);
$con=new PDO("mysql:host=localhost;dbname=youtube","root","");
$sql="select student.name, city.city, game.game, study.study, teacher.teacher from
student,city,game,study,teacher,fee where student.city=city.id and student.game=game.id and
student.study=study.id and student.teacher=teacher.id and student.id=fee.student_id";
$stmt=$con->prepare($sql);
$stmt->execute();
$arr=$stmt->fetchAll(PDO::FETCH_ASSOC);
$str=<table border='1'>;
    $str.=<tr><td>Name</td><td>City</td><td>Game</td><td>Study</td><td>Teacher</td></tr>;
    foreach($arr as $list){
        $str.=<tr><td>".$list['name']."'</td><td>".$list['city']."'</td><td>".$list['game']."'</td>
        $list['study']."'</td><td>".$list['teacher']."'</td></tr>";
    }
$str.=</table>;
echo $str;
$end=microtime(true);
echo round($end-$start,4);
?>
```

```

<?php
//Caching is a temporary storage location, to access data more quickly. Caching is used where the original data is
expensive to fetch.
$start=microtime(true);
$con=new PDO("mysql:host=localhost;dbname=youtube","root","");
$cache_file="cache/index.cache.php";
if(file_exists($cache_file) && filemtime($cache_file) > time()-20){
    echo "From Cache<br/>";
    include($cache_file);
}else{
    $sql="select student.name, city.city, game.game, study.study, teacher.teacher from student,city,game,study,
teacher,fee where student.city=city.id and student.game=game.id and student.study=study.id and student.
teacher=teacher.id and student.id=fee.student_id";
    $stmt=$con->prepare($sql);
    $stmt->execute();
    $arr=$stmt->fetchAll(PDO::FETCH_ASSOC);
    $str=<table border='1'>;
    $str.=<tr><td>Name</td><td>City</td><td>Game</td><td>Study</td><td>Teacher</td></tr>;
    foreach($arr as $list){
        $str.=<tr><td>".$list['name']."'</td><td>".$list['city']."'</td><td>".$list['game']."'</td><td>".$list
['study']."'</td><td>".$list['teacher']."'</td></tr>";
    }
    $str.=</table>;
    $handle=fopen($cache_file,'w');
    fwrite($handle,$str);
    fclose($handle);
    echo "Cache Created<br/>";
    echo $str;
}
$end=microtime(true);
echo round($end-$start,4);
?>

```

```

<?php
//Caching is a temporary storage location, to access data more quickly. Caching
is used where the original data is expensive to fetch.
$start=microtime(true);
$con=new PDO("mysql:host=localhost;dbname=youtube","root","");
$cache_file="cache/index.cache.php";
if(file_exists($cache_file) && filemtime($cache_file) > time()-20){
    echo "From Cache<br/>";
    include($cache_file);
}else{
    $sql="select student.name, city.city, game.game, study.study, teacher.teacher
from student,city,game,study,teacher,fee where student.city=city.id and
student.game=game.id and student.study=study.id and student.teacher=teacher.id
and student.id=fee.student_id";
    $stmt=$con->prepare($sql);
    $stmt->execute();
    $arr=$stmt->fetchAll(PDO::FETCH_ASSOC);
    $str=<table border='1'>;

```

```
$str.= "<tr><td>Name</td><td>City</td><td>Game</td><td>Study</td><td>Teacher</td></tr>";
foreach($arr as $list){
    $str.= "<tr><td>".$list['name']."</td><td>".$list['city']."</td><td>".$list['game']."</td><td>".$list['study']."</td><td>".$list['teacher']."</td></tr>";
}
$str.= "</table>";
$handle=fopen($cache_file,'w');
fwrite($handle,$str);
fclose($handle);
echo "Cache Created<br/>";
echo $str;
}
$end=microtime(true);
echo round($end-$start,4);
?>
```

## Array



### Multidimensional Associative Array :

Name	Physics	Chemistry	Math
Krishana	85	89	78
Salman	68	79	73
Mohan	62	92	67

```
$marks = array (
    "Krishana" => array("physics" => 85, "chemistry" => 89, "math" => 78),
    "Salman" => array("physics" => 68, "chemistry" => 79, "math" => 73),
    "Mohan" => array("physics" => 62, "chemistry" => 92, "math" => 67)
);

$marks = [
    "Krishana" => [
        "physics" => 85,
        "maths" => 78,
        "chemistry" => 89
    ],
    "Salman" => [
        "physics" => 68,
        "maths" => 73,
        "chemistry" => 79
    ],
    "Mohan" => [
        "physics" => 62,
        "maths" => 67,
        "chemistry" => 92
    ]
];
```

```
<?php

/* First method to create array. */
$numbers = array( 1, 2, 3, 4, 5);

foreach( $numbers as $value ) {
    echo "Value is $value <br />";
}

/* Second method to create array. */
$numbers[0] = "one";
$numbers[1] = "two";
$numbers[2] = "three";
$numbers[3] = "four";
$numbers[4] = "five";

foreach( $numbers as $value ) {
    echo "Value is $value <br />";
}
?>
```

Loop through and print all the values of an associative array:

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");

foreach($age as $x=>$x_value)
{
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

```
1     "chemistry" => 79
2   ],
3   "Mohan" => [
4     "physics" => 62,
5     "maths" => 67,
6     "chemistry" => 92
7   ]
8 ];
9
10 foreach($marks as $key => $v1){
11   echo $key ;
12   foreach($v1 as $v2){
13     echo $v2 . " ";
14   }
15   echo "<br>";
16 }
17
```

Krishana85 78 89  
Salman68 73 79  
Mohan62 67 92

Array  
(  
 [Krishana] => Array  
 ( [physics] => 85  
 [maths] => 78  
 [chemistry] => 89  
 )  
 [Salman] => Array  
 ( [physics] => 68  
 [maths] => 73  
 [chemistry] => 79  
 )  
 [Mohan] => Array  
 ( [physics] => 62  
 [maths] => 67  
 [chemistry] => 92  
 )  
)

```
1     "chemistry" => 92
2   ];
3
4 echo "<table border='2px' cellpadding='5px' ";
5 foreach($marks as $key => $v1){
6   echo "<tr>
7     <td>$key</td>" ;
8     foreach($v1 as $v2){
9       echo "<td> $v2 </td>";
10      }
11    echo "</tr>";
12  }
13 echo "</table>";
14
```

Krishana	85	78	89
Salman	68	73	79
Mohan	62	67	92

Array  
(  
 [Krishana] => Array  
 ( [physics] => 85  
 [maths] => 78  
 [chemistry] => 89  
 )  
 [Salman] => Array  
 ( [physics] => 68  
 [maths] => 73  
 [chemistry] => 79  
 )  
)

## PHP Array Functions

- **count()** - returns the number of elements in an array.
- **in\_array()** - checks if a value exists in an array.
- **array\_push()** - adds one or more elements to the end of an array.
- **array\_pop()** - removes and returns the last element of an array.
- **array\_shift()** - removes and returns the first element of an array.
- **array\_unshift()** - adds one or more elements to the beginning of an array.
- **array\_reverse()** - reverses the order of the elements in an array.
- **array\_merge()** - merges two or more arrays into one array.
- **array\_slice()** - extracts a slice of an array.
- **array\_splice()** - removes and replaces elements from an array.
- **array\_keys()** - returns an array of the keys of an array.
- **array\_values()** - returns an array of the values of an array.
- **array\_flip()** - flips the keys and values of an array.
- **array\_unique()** - removes duplicate values from an array.
- **array\_search()** - searches for a value in an array and returns its key.
- **array\_rand()** - returns one or more random keys from an array.
- **array\_walk()** - applies a user-defined function to each element of an array.
- **array\_map()** - applies a user-defined function to each element of an array and returns a new array with the results.
- **array\_filter()** - filters an array using a user-defined function and returns a new array with the filtered values.
- **array\_reduce()** - reduces an array to a single value using a user-defined function.
- **array\_column()** - returns an array of values from a single column of a multi-dimensional array.
- **array\_intersect()** - returns an array of the values that exist in two or more arrays.
- **array\_diff()** - returns an array of the values that exist in one array but not in another.



## PHP Array : Extract() Function

```
$color = array('a' => 'red', 'b' => 'green', 'c' => 'blue');
```

↓      ↓      ↓  
  \$ a    \$ b    \$ c

**extract(array, extract\_rules, prefix)**



## PHP Array : Compact Function

```
$first = "red";                          Array(  
$second = "green";      →             "first" => "red",  
$third = "blue";                         "second" => "green",  
                                               "third" => "blue",  
                                               )  
                                               )
```

**compact(var1, var2, var3.....)**



## PHP Array : Range() Function

1      5

array(1, 2, 3, 4 ,5)

**range(start, end, step)**



```
<?php  
  
// $food = array('orange', 'banana', 'apple', '55', 'grapes');  
$food = array('a' => 'orange', 'b' => 'banana', 'c' => 'apple', 'd' => 'grapes');  
  
echo array_search('apple', $food);  
  
?>
```



## PHP Array – Array\_Map() Function :

```
$a = array(  
    "Bill" => 10,  
    "Joe" => 20,  
    "Peter" => 30  
)
```

function myFunction(){  
}

**array\_map(function , array)**

```
$a = array(  
    "Bill" => 10,  
    "Joe" => 20,  
    "Peter" => 30  
)
```

function myFunction(){  
}

Return Array ←

**array\_map(function , array , array2 , array3 ....)**

```
<?php  
function square($n){  
    return $n * $n;  
}  
  
$a = [1, 2, 3, 4, 5];  
  
$newArray = array_map('square', $a);  
  
echo "<pre>";  
print_r($newArray);  
echo "</pre>";
```



## PHP Array – Merge & Combine Functions :

```
$a = ["Sanjay", "Aman", "Rehman"];  
  
$b = ["Karan"];  
  
["Sanjay", "Aman", "Rehman", "Karan"]
```

- `array_merge()` Index or Associative Array
- `array_merge_recursive()` Multidimensional Associative Array
- `array_combine()` Index Array

```
<?php  
$fruit = ["orange", "banana", "grapes"];  
  
$veggie = ['carrot', 'pea'];  
  
$color = ['red', 'blue'];  
  
$newArray = array_merge($fruit,$veggie,$color);  
  
echo "<pre>";  
print_r($newArray);  
echo "</pre>";
```

```
<?php  
$fruit = ['a' => "orange", 'b' => "banana", 'c' => "grapes"];  
  
$veggie = ['carrot', 'pea'];  
  
$newArray = array_merge($fruit,$veggie);  
  
echo "<pre>";  
print_r($newArray);  
echo "</pre>";
```

```
<?php  
$name = array("Ram","Mohan","Salman");  
  
$age = array("35","37","43");  
  
$newArray = array_combine($name,$age);  
  
echo "<pre>";  
print_r($newArray);
```

## PHP Array\_fill\_keys() Function :

```
$a = ["a", "b", "c", "d"];
```

```
Array(  
    "a" => "test",  
    "b" => "test",  
    "c" => "test",  
    "d" => "test"  
)
```

```
array_fill_keys(array , value)
```



## PHP Array – Array\_Reduce() Function :

```
$a = array(
```

```
10,
```

```
20,
```

```
30
```

```
);
```

```
function myFunction(){  
}
```

Return Single Value

```
array_reduce(array , function , initial)
```

```
<?php  
$a1 = array("a"=>"red", "b"=>"green", "c"=>"blue", "d"=>"yellow");  
  
$a2 = array("a"=>"red", "f"=>"green", "d"=>"purple");  
  
$newArray = array_intersect($a1, $a2);  
  
echo "<pre>";  
print_r($newArray);  
echo "</pre>";
```

```
<?php
    $values=['Mohammad','','10,0',false,"2020",null,true];
    $filtered = array_filter($values);
    echo "<pre>";
    print_r($values);
    print_r($filtered);
```

## PHP Array\_fill() Function :

Array(  
 [0] => "test",  
 [1] => "test",  
 [2] => "test",  
 [3] => "test"  
);

Array(  
 [3] => "test",  
 [4] => "test",  
 [5] => "test"  
);

**array\_fill(index , number , value)**

## PHP Array : Difference Functions

- array\_diff
- array\_diff\_key
- array\_diff\_assoc
- array\_diff\_uassoc
- array\_diff\_ukey
- array\_udiff\_assoc
- array\_udiff\_uassoc
- array\_udiff



## PHP Array : Difference Functions

### Different Values

```
$a = ["a" => "Sanjay", "b" => "Aman", "c" => "Mohan"];
```

```
$b = ["a" => "Sanjay", "d" => "Mohan"];
```

▶

array\_diff



## PHP Array : Intersect Functions

### Match Keys

```
$a = ["a" => "Sanjay", "b" => "Aman", "c" => "Mohan"];  
↓  
$b = ["a" => "Sanjay", "d" => "Mohan"];
```

array\_intersect\_key



## PHP Array : Difference Functions

### Different Keys

```
$a = ["a" => "Sanjay", b => "Aman", c => "Mohan"];  
↑  
$b = ["a" => "Sanjay", "d" => "Mohan"];
```

array\_diff\_key



## PHP Array : Intersect Functions

### Match Values

```
$a = ["a" => "Sanjay", "b" => "Aman", "c" => "Mohan"];  
↑  
$b = ["a" => "Sanjay", "d" => "Mohan"];
```

array\_intersect

```
<?php  
$a1 = array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");  
  
$a2 = array("a"=>"red","f"=>"green","d"=>"purple");  
  
$newArray = array_intersect($a1, $a2);  
  
echo "<pre>";  
print_r($newArray);  
echo "</pre>";  
  
<?php  
$values=['Mohammad','','10,0',false,"2020",null,true];  
$filtered = array_filter($values);  
echo "<pre>";  
print_r($values);  
print_r($filtered);
```



## PHP Array – Array\_Reduce() Function :

```
$a = array(  
    10, → function myFunction(){  
    20, →  
    30 }  
);  
                    Return Single Value ←
```

array\_reduce(array , function , initial)

```

<?php
function myFunction($n , $m){
    return $n . "-" . $m;
}

$a = ['orange', 'banana', 'apple'];

$newArray = array_reduce($a, 'myFunction', 'lemon');

echo "<pre>";
print_r($newArray);

```

```

$array = array(
    array(
        'id' => 2201,
        'first_name' => 'Anil',
        'last_name' => 'Kapoor',
    ),
    array(
        'id' => 2202,
        'first_name' => 'Salman',
        'last_name' => 'Khan',
    ),
    array(
        'id' => 2203,
        'first_name' => 'John',
        'last_name' => 'Abraham',
    )
);

```

## array\_column

```

Array
(
    [0] => Anil
    [1] => Salman
    [2] => John
)

```

\$a = ["Red", "Green", "Blue", "Orange", "Brown"];

```

Array
(
    [0] => Array(
        [0] => Red
        [1] => Green
    ),
    [1] => Array(
        [2] => Blue
        [3] => Orange
    ),
    [2] => Array (
        [4] => Brown
    )
)

```

## array\_chunk (array , size)



Don't forget to hit the Subscribe & Like button

```
array.php • Settings
1 <?php
2 $fruits = array(
3     "a" => "Lemon",
4     "b" => "Orange",
5     "c" => "Banana",
6     "d" => "Apple"
7 );
8
9 array_walk($fruits, "myFunction", );
10
11 function myFunction($value, $key){
12     echo "$key : $value <br>";
13 }
14
15
```



## PHP Array – Array\_Walk() Function :

```
$a = array(  
    "Bill" => 10,  
    "Joe" => 20,  
    "Peter" => 30  
)
```

function myFunction(){  
}

array\_walk(array , function , parameter)

## String Function

```
<?php

$data = "12344 $ # % @ ! ^ ' ". 'ram' . " ' & ' ' Have a nice day &
best of luck";

echo "<br/><br/><a href=urlencode0.php?data=".urlencode($data)."
target='_blank'> click here for send data with urlencode()
</a>";

echo "<br/><br/><a href=urlencode0.php?data=". $data."
target='_blank'> click here for send data without
urlencode() </a>";
```

I

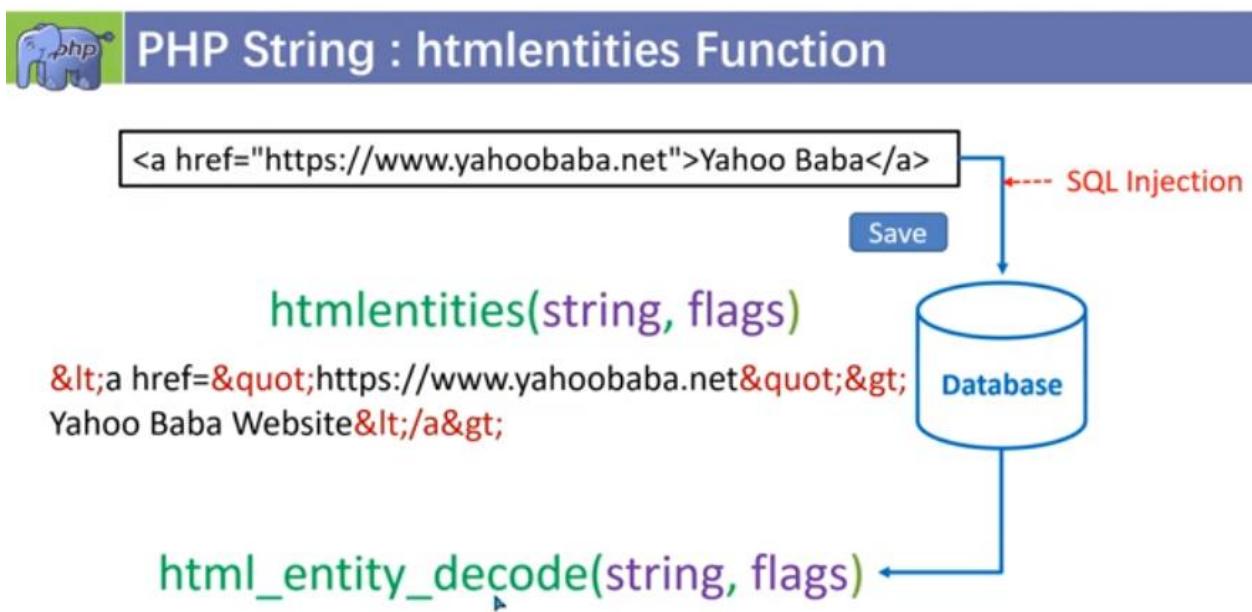
?>

urlencode(\$string);

- Letters, numbers, underscores, dashes are unchanged
- Reserved characters become % + 2 digit hexadecimal value
- Spaces become + sign

```
<html>
  <head>
    <title>First Page</title>
  </head>
  <body>
    <?php $link_name = "Second Page"; ?>
    <?php $id = 2; ?>
    <?php $company = "Johnson & Johnson"; ?>
    <a href="second_page.php?id=<?php echo $id; ?>&company=<?php echo urlencode($company); ?>">
    <?php echo $link_name; ?></a>
    I
  </body>
</html>
```

1. `strlen()`
2. `str_replace()`
3. `strstr()`
4. `substr()`
5. `strtolower()`
6. `strtoupper()`
7. `str_word_count()`
8. `strrev()`
9. `strpos()`
10. `ucwords()`
11. `ucfirst()`
12. `lcfirst()`
13. `wordwrap()`
14. `trim()`
15. `strip_tags()`





## PHP String : htmlentities Function

`htmlentities(string, flags)`



- **ENT\_COMPAT** Default. Encodes only double quotes
- **ENT\_QUOTES** Encodes double and single quotes
- **ENT\_NOQUOTES** Does not encode any quotes

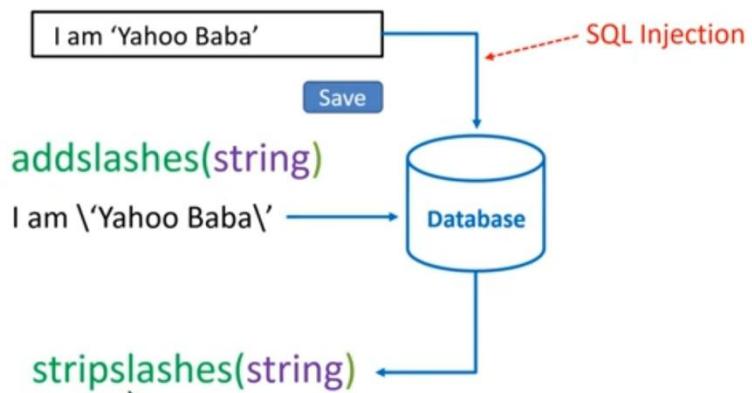
`htmlspecialchars(string, flags)`



`htmlspecialchars_decode(string, flags)`



## PHP String : addslashes & stripslashes Functions





## PHP String Length & Count Functions

\$a = "Hello World"; → 11

strlen(string)

\$a = "Hello World"; → 2

str\_word\_count(string, return)

\$a = "Hello World"; → World

substr\_count(string, substring, start, length)



## PHP String : Find Position Functions

\$a = "I love php, I love php too!";

strpos(string, find, start) → 7

strrpos(string, find, start) → 19

case-sensitive      php    Php



## PHP String : Compare Functions

hello                  Hello

strcmp(string1, string2)                  case-sensitive

strncmp(string1, string2, length)

strcasecmp(string1, string2)                  case-insensitive

strncasecmp(string1, string2 , length)





## PHP String : Search String Functions

`$a = "I love php, I love php too!";`

`strstr(string, search, before_search) → , I love php too!`

`strchr(string, search, before_search)`

`strrchr(string, search) → too!`

case-sensitive

php PHP



Yahoo Baba

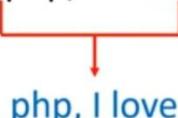
www.yahooomba.net

Yahoo  
Baba



## PHP String : Substr() Function

`$a = "I love php, I love php too!";`



`substr(string, start, length)`



Yahoo Baba

www.yahooomba.net

Yahoo  
Baba



## PHP String : Str\_replace() Function

`$a = "I love php, I love php too!";`

`I love python, I love python too!`

`str_replace(find, replace, string)` case-sensitive  
PHP

`str_ireplace(find, replace, string)`



Yahoo Baba

www.yahooomba.net

Yahoo  
Baba



## PHP String Reverse Function

hello → olleh

`strrev(string)`



www.yahooomba.net

Yahoo  
Baba



## PHP String Shuffle Function

hello → llohe



eohll

lheloo

`str_shuffle(string)`

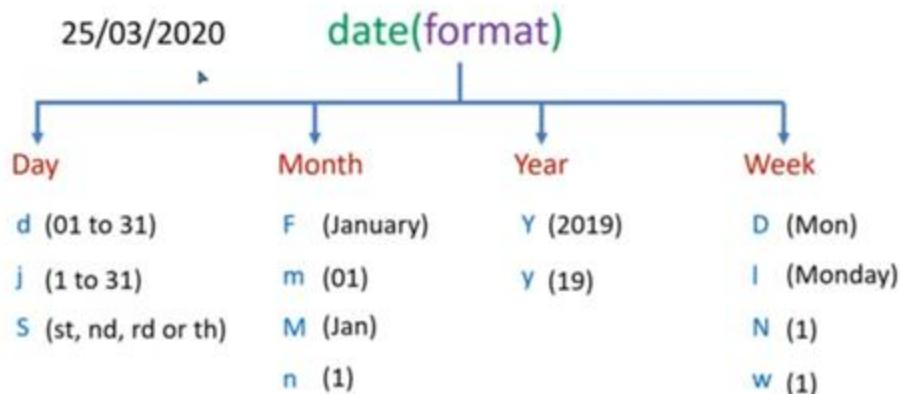


www.yahooomba.net

Yahoo  
Baba



## PHP String : date() Function



www.yahooomba.net



## PHP String : date() Function

