PHP {HYPERTEXT PREPROCESSOR}

Introduction :

i. PHP is a general purpose scripting language towards web development.
ii. It was originally created by Danish and Canadian programmer Rasmus Lerdorf in 1993 and released in 1995.
iii. PHP was originally an abbreviation of Personal Home Page, but it now stands for Hypertext Preprocessor.
iv. It creates dynamic website with database.
v. It is a web based software applications.

NOTE:-

a) PHP is a case sensitive language.
b) PHP is a interpreted language.

Why we learn PHP ?

- Easy to use and learn.
- Cost efficient.
- Object Oriented Programming.
- Cross-platform compatibility.
- Work with all servers like Apache, IIS, etc.
- Security.
- Huge communication.

Difference between Client Side Script and Server Side Script

Basic of comparison	Client Side Scripting	Server Side Scripting
Basic	Works at the front end and script are visible among the users.	Works in the back end which could not be visible at the client end.
Processing	Does not need interaction with the server.	Requires server interaction.
Language involved	HTML, CSS, JAVASCRIPT, etc.	PHP, ASP.net, ruby on rails, coldfusion, python, etc.

PHP {HYPERTEXT PREPROCESSOR}

Affect	Can reduce the load to the server.	Could affectively customize the web pages and provide dynamic websites.
Security	Insure.	Relatively secure.

Uses of PHP in web development :

- CMS (Content Management System).
- E-commerce website.
- Mail server.
- Chatting system.
- File storage system.
- ERP(Enterprise Resource Planning).
- CRM(Customer Relationship Management).

Popular websites in PHP :

- Facebook
- Yahoo
- Wikipedia
- Flipkart
- Wordpress
- MailChimp
- Flickr

Prior knowledge before learning PHP :

- HTML
- CSS(Basic)
- JavaScript(Optional)

Software required for PHP :

a) HTML Editor
b) Web Browser
c) Wamp or Xampp Server

A) Wamp Server :

PHP {HYPERTEXT PREPROCESSOR}

- WAMP stands for Window, Apache, MySQL and PHP.
- Wamp server is a Windows web development environment.
- It allows you to create web application with Apache2, PHP and a MySQL database.
- It is available for free in two distinct versions:-

i. WampServer 64 bits (X64) 3.2.6
ii. WampServer 32 bits (X86) 3.2.6

Steps to install Wamp server -

1. Visit the Wamp server website in your web browser www.wampserver.com.
2. Click on “WAMP SERVER 64 BITS (X64)”.
3. Now, click on “download directly” link to start downloading.
4. Double click the downloaded file to launch the WAMP installer.
5. Now, select the setup language.
6. Under “License Agreement”, click on “I accept the agreement” and click on “Next” button.
7. Click on “Next” button.
8. Choose the location(folder) to install Wamp server and then click on “Next” button.
9. Select the components that you want to install and click on “Next” button.
10. Select the start menu folder.
11. “Ready to install” window will appear on the screen and then click on “Next” button.
12. Now, select the default browser and default text editor for the Wamp server.
13. Click on “Finish” button.

B) Xampp Server :

- XAMPP stands for X – operating system, Apache, MySQL, PHP, Perl.
- Xampp is a free and open-source cross-platform web server solution.
- It is developed by Apache Friends.

PHP {HYPERTEXT PREPROCESSOR}

- It consist mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in PHP and Perl programming language.

Steps to install Xampp server -

1. Visit the Xampp server website in your web browser www.apachefriends.org and download Xampp installer.
2. During the installing process, select required components like MySQL, FileZilla ftp server, PHP, phpMyAdmin or leave the default option and click on "Next" button.
3. Uncheck the **Learn more about bitnami** option and click on "Next" button.
4. Choose the root directory path too setup the *htdocs* folder for our application.
5. Click on "Allow access" button to aloe the Xampp modules from the windows firewall.
6. After installation process, click on "Finish" button of the Xampp setup wizard.
7. Now, the Xampp icon is clearly visible on the right side of start menu.
8. To start Apache and MySQL, just click on start button on the control panel.

NOTE:-

C drive → htdocs → create new folder for code in PHP.

PHP syntax –

```
<?php
PHP code
?>
```

PHP "echo" statement : Print message or variable value.

- echo "Hello world";
- echo 'Hello world';
- echo "Hello"," world";
- echo "Hello"."world";

PHP {HYPERTEXT PREPROCESSOR}

- echo “Hello world”;
- echo 538;

PHP “print” statement :

- print “Hello world”;
- print “Hello”.“ world”;
- print “Hello world”;
- print 684;

NOTE :-

Echo is faster than print.

PHP Variables :

- $firstname
- $_firstname
- $first_name
- $first-name
- $firstName

$firstname99

DATA TYPES :

i. String
ii. Integer
iii. Float
iv. Boolean
v. Array
vi. Object
vii. Null

Comment-

1. Single line comment :-
   //This is a comment
   Or
   # This is a comment
2. Multiple line comment :-

/* This is a comment */

Constant variable syntax :

define(name, value, case-insensitive)

Example – define(num, 564)

NOTE :-

a) Constant variable is a **global variable**.
b) Case-insensitive default value is **false**.

OPERATORS

Arithmetic Operators

OPERATORS	DESCRIPTION	EXAMPLE
+	Addition	$x + $y
-	Subtraction	$x - $y
*	Multiplication	$x * $y
**	Exponentiation	$x ** $y
/	Division	$x / $y
%	Modulus(Remainder)	$x % $y
++	Increment Operator	$x++ or ++$x
--	Decrement Operator	$x-- or --$x

Assignment Operators

OPERATORS	EXAMPLE	SAME AS
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

PHP {HYPERTEXT PREPROCESSOR}

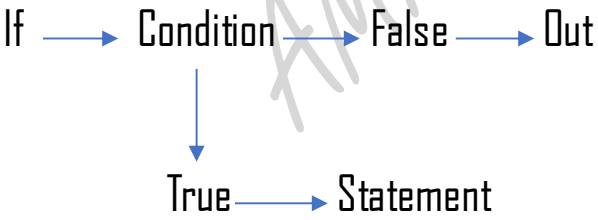
Comparison Operators

OPERATORS	DESCRIPTION	EXAMPLE
==	Equal to	$x == $y
===	Equal value & equal data type	$x === $y
!=	Not equal	$x != $y
<>	Not equal	$x <> $y
!==	Not equal value or not equal data type	$x !== $y
>	Greater than	$x > $y
<	Less than	$x < $y
>=	Greater than or equal to	$x >= $y
<=	Less than or equal to	$x <= $y

NOTE :-

a) True → "1"
b) False → " " → empty space.
c) New comparison operator in PHP 7 version –
<=> → Spaceship → $x <=> $y
It return -1, 0 , or 1 respectively less than, equal to, or greater than.

If statement :-



Syntax –

```
if(condition) {
  Statement;
}
```

OR

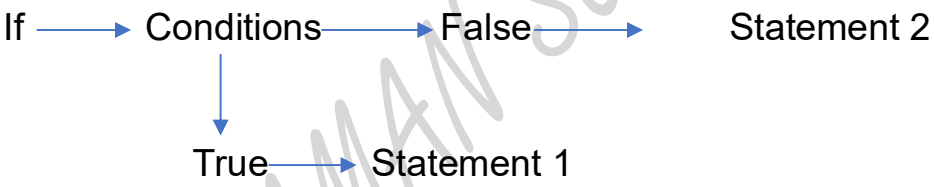
```
if(condition):
Statement;
endif;
```

Logical Operators

OPERATORS	NAME	EXAMPLE
&&	Logical AND	Condition 1 && Condition 2
||	Logical OR	Condition 1 || Condition 2
!	Logical NOT	!(Condition)
and	Logical AND	Condition 1 and Condition 2
or	Logical OR	Condition 1 or Condition 2
xor	Exclusive OR	Condition 1 xor Condition 2

LOOPS

If Else statement :-



Syntax –

```
if(condition) {
Statement for true;
} else {
Statement for false;
}
```

If Elseif statement :-

Syntax –

```
if(Condition 1) {
Statement 1;
} elseif(Condition 2) {
Statement 2;
} else {
Default statement;
}
```

OR

```
if(Condition 1):
Statement 1;
elseif(Condition 2):
Statement 2;
else:
Default statement;
endif;
```

Switch statement :-

Syntax –

```
switch(expression) {
case condition 1: statement 1;
break;
case condition 2: statement 2;
break;
case condition 3: statement 3;
break;
default: statement;
```

}

OR

switch(expression) :

case condition 1: statement 1;

break;

case condition 2: statement 2;

break;

case condition 3: statement 3;

break;

default: statement;

endswitch;

Ternary Operator :-

Syntax –

(Condition)? True statement : False statement;

String Operator

```
$a = “Hello                                    ”;
$a .= “world ”;
$a .= “! 😊”;
echo $a;
```

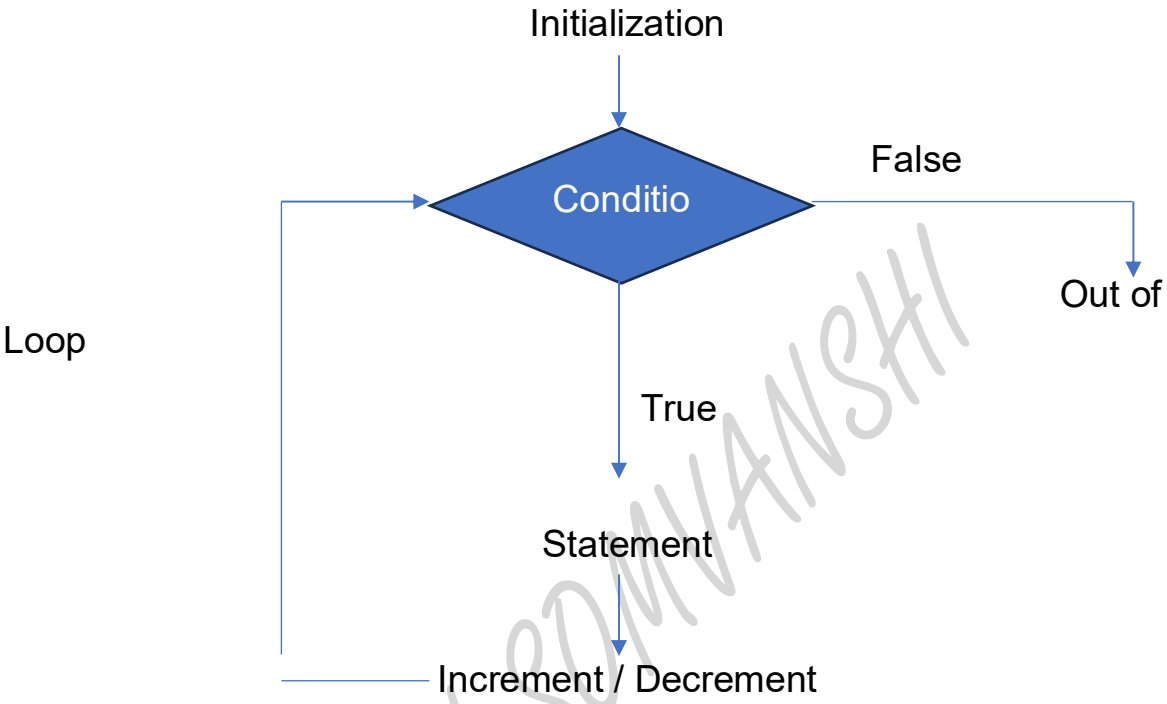
OUTPUT :-

Hello world ! 😊

NOTE :-

There will be no space between .=

Loop :



Types of loops :

- While loop
- do while loop
- for loop
- foreach

While loop –

Syntax

```
Initialization;
while(condition){
statement;
```

```
increment / decrement;
}
```

Do while loop –

Syntax

```
Initialization;
do{
statement;
increment / decrement;
} while (condition)
```

For loop –

Syntax

```
for(initialization; condition; increment / decrement){
statement;
}
```

Nested loop :-

Syntax

```
for(initialization; condition 1; increment / decrement){
for(initialization; condition 2; increment / decrement){
statement;
}
}
```

Continue and Break statement :

1. Continue statement syntax -

```
for(initialization; condition; increment / decrement){
```

```
if(condition){
 continue;
}
statement;
}
```

2. Break statement syntax –

```
for(initialization; condition; increment / decrement){
 if(condition){
  break;
 }
statement;
}
```

GOTO Statement :

Syntax –

```
for(initialization; condition; increment / decrement){
 if(condition){
  goto abc;  ──────►  Lable
 }
statement;
}
 abc:
 echo $a;  ──┐
             └──  defining lable
```

Functions:

Syntax

function functionName(){ → Function definition / function decleration

Statement;

}

functionName(); → Calling a function

Note:-

- var_dump() → gives data type, value, and length.

Function with parameter –

Syntax

function functionName (parameter 1, parameter 2){

Statement;

}

functionName(argument 1, argument 2)

Function with Return value :

function functionName (parameter 1, parameter 2){

Statement;

return value;

}

$a = function(argument 1, argument 2);

Function arguments by reference :

Variable functions :

function function_name(){

statement;

}

```
$func = “function_name”;
$func();
```

Recursive functions :

```
function function_name(){
    statement;
function_name();
    }
function_name();
```

Global and Local Variable :

Local variable –

```
function function_name(){
    $a = value;
    echo $a;
    }
```

→ **Local Variable**

OR

```
$a = value;
function function_name(){
    global $a;
    echo $a;
    }
```

→ **Using “*global*” keyword to use the value of global variable.**

Global variable –

```
$a = value;
function function_name(){
```

→ **Global Variable**

```
}
function_name();
echo $a;
```

ARRAY

Array :

Syntax

```
$a = array(value1, value2, value3);
```

OR

```
$a = [value1, value2, value3];
```

print_r();

Associative array :

```
<?php
$age = [
1 => "21"
2 => "24"
3 => "31"
];
$age["Aman"] = 45;
echo $age[1] . "<br>";
echo $age[2] . "<br>";
echo $age[3] . "<br>";
?>
```

OUTPUT :-

21

24

31

Foreach loop :

Syntax –

```
foreach($array as $value){
echo $value;
}
```

Multidimensional array :

- A multidimensional array is an array containing one or more arrays.
- As the name suggests, every element in this array can be an array and they can also hold other sub-array within.
- Array or sub-array in multidimensional arrays can be accessed using multiple dimensions.

Multidimensional Associative array :

```
<?php
$marks = [
“Aman” =>
[
“Physics” => 85
“Maths” => 78
“Chemistry” => 89
]
“Awan” =>
[
“Physics” => 86
“Maths” => 98
“Chemistry” => 99
]
“Piyush” =>
[
```

```
"Physics" => 75
"Maths" => 88
"Chemistry" => 69
]
];
echo"<table border = '2px' cellpadding = '5px' cellspacing = '3px'>";
<tr>
<th>Student Name</th>
<th>Physics</th>
<th>Maths</th>
<th>Chemistry</th>
</tr>
foreach($marks as $key => $v1){
echo "<tr>"
<td>$key</td>;
foreach($v1 as $v2){
echo "<td>$v2</td>";
}
echo "</tr>";
echo "<pre>"
print_r($marks);
echo "</pre>"
?>
```

OUTPUT :-

Student Name	Physics	Maths	Chemistry
Aman	85	78	89
Awan	86	98	99
Piyush	75	88	69

Foreach loop syntax with list() –

```
foreach($array as list()){
```

statement;

}

count() & sizeof() :

Syntax

- count($array);
- sizeof($array);

ARRAY FUNCTIONS

1. In_array() & Array_search() :

- in_array() return two value **1 for true** and **0 or empty space for false**.
- array_search() return **index/key**.

Syntax

- in_array(value, $array);
- array_search(value, $array);

Here, **value** is the element of array.

2. Array_replace() & Array_replace_recursive() :

- array_replace() is used for index or associative array.
- array_replace_recursive() is used only for multidimensional array.

Syntax

- array_replace($array1, $array2);
- array_replace_recursive($array1, $array2);

3. Array_pop() & Array_push() :

- array_pop() is used to delete an element of an array from last.
- array_push() is used to add an element in the last of an array.

Syntax

- array_pop($array);
- array_push($array);

4. Array_shift() & Array_unshift() :

- array_shift() is used to remove an element of an array from first.
- array_unshift() is used to add an element at first of an array.

Syntax

- array_shift($array);
- array_unshift($array);

5. Array_merge() & Array_combine() :

- array_merge() is used in index or associative array.
- array_merge_recursive() is used only for multidimensional array.
- array_combine() is used only for index array.

Syntax

- array_merge($array1, $array2);
- array_merge_recursive($array1, $array);
- array_combine($array1, $array2);

array_merge_recursive() is use to merge common array.

6. Array_slice() :

- Used to make a new array from the elements of other array.

Syntax

array_slice($array, start, length);

- Start = from which element is has to be start.
- length = how much element it has to take.

7. Array_splice() :

Used to make a new array by replacing the elements of an array.

Syntax

array_splice($array1, start, length, $array2);

8. Array Key Functions :

array_keys() → Used to return all the keys of an array.

array_key_first() → Used to return first keys of an array. In PHP

array_key_last() → Used to return last keys of an array. version 7.3.0

array_key_exist() → Used to check if key exist or not in an array.

key_exists() → Used to check if key exist or not in an array. It is the shorthand of

array_key_exist()

Syntax

- array_keys($array);
- array_key_first($array);
- array_key_last($array);
- array_key_exist(value, $array);
- key_exist(value, $array);

Here, value is the element of an array.

9. Array Intersect Functions :

array_intersect() → Used to return same value of arrays.

array_intersect_key() → Used to return same keys of arrays.

array_intersect_assoc() → Used to return same value with same key of arrays.

array_intersect_uassoc() → Used to return same value with same key of arrays by

user-define function.

array_intersect_ukey() → Used to return same key of arrays by user-define function.

array_uintersect() → Used to return same value of arrays by user-define function.

array_uintersect_assoc() → Used to return same value of arrays by user-define

function.

array_uintersect_uassoc() → Used to return both same value with same key of array

by user-define function.

Syntax

- array_intersect($array1, $array2);
- array_intersect_key($array1, $array2);
- array_intersect_assoc($array1, $array2);
- array_intersect_uassoc($array1, $array2, “compareFunction”);
- array_intersect_ukey($array1, $array2, “compareFunction”);
- array_uintersect($array1, $array2, “compareFunction”);
- array_uintersect_assoc($array1, $array2, “compareFunction”);
- array_uintersect_uassoc($array1, $array2, “compareFunction1”, “compareFunction2”);

10. Array Difference Functions :

array_diff() → Used to return diff value of arrays.

array_diff_key() → Used to return different keys of arrays.

array_diff_assoc() → Used to return different value with different key of arrays.

array_diff_uassoc() → Used to return different value with different key of arrays by

user-define function.

array_diff_ukey() → Used to return different key of arrays by user-define function.

array_udiff() → Used to return different value of arrays by user-define function.

array_udiff_assoc() → Used to return different value of arrays by user-define

function.

array_udiff_uassoc() → Used to return both different value with different key of array

by user-define function.

Syntax

- array_diff($array1, $array2);
- array_diff_key($array1, $array2);
- array_diff_assoc($array1, $array2);
- array_diff_uassoc($array1, $array2, “compareFunction”);
- array_diff_ukey($array1, $array2, “compareFunction”);
- array_udiff($array1, $array2, “compareFunction”);
- array_udiff_assoc($array1, $array2, “compareFunction”);
- array_udiff_uassoc($array1, $array2, “compareFunction1”, “compareFunction2”);

11. Array_values() & Array_unique() :

- array_values() used to make an index array by using the values of an associative array.
- array_unique() used to make array by using the unique value of an array not by the duplicate value.

Syntax

- array_values($array);
- array_unique($array);

12. Array_column() & Array_chunk() :

- array_column() is used to make a new index array by using a specific field of an associative array.
- array_chunk() used to make an index array by pairing up the value of an associative array.

Syntax

- array_column($array, perimeter);
- array_chunk($array);

13. Array_flip() & Array_change_key_case() :

- array_flip() is used to swap/flip the value with key in an array.

- array_change_key_case() is used to change the case of array i.e, lower case to upper case and vice-verse.

Syntax

- array_flip($array);
- array_change_key_case($array, case);

14. Array_sum() & Array_product() :

- array_sum() is used to add the numeric value of an array.
- array_product() is used to multiply the numeric value of an array.

Syntax

- array_sum($array);
- array_product($array);

15. Array_rand() & Shuffle() :

- array_rand() is used to give a random value from array.
- shuffle() is used to interchange the position of the value in an array.

Syntax

- array_rand($array, number);
- shuffle($array);

Here, **number** means how many value you want to take randomly.

16. Array_fill() & Array_fill_key() :

- array_fill_key() is used to make key of an associative array with the value of an other array.
- array_fill() is used to an array of your own decided index.

Syntax

- array_fill_key($array, value);
- array_fill(index, number, value);

17. Array_walk() & Array_walk_recursive() :

- array_walk() is used to run a function for every value of an array. It work only with index array and associative array.

- array_walk_recursive() works same as array_walk() but for multidimensional associative array.

Syntax

- array_walk($array, function, parameter);
- array_walk_recursive($array, function, parameter);

18. Array_map() :

- It works similar to array_walk() but it return a new array by manipulating the value of given array.
- We can take multiple array in this.

Syntax

```
array_map(function, $array);
```

19. Array_reduce() :

- It works similar to array_reduce() but it return a single value by manipulating the value of given array.

Syntax

```
array_reduce($array, function, initial);
```

Here, **initial** is optional. It is used to provide a starting value other then the value present in that array.

20. Array sorting function :

- It is used to arrange the value of an array in either ascending order or descending order or reverse order.

Syntax

- sort($array);
- rsort($array);

} Used only for index array

- arsort($array);

- asort($array): Used to sort associative array
- krsort($array);
- ksort($array); Used in sort keys of associative array
- natcasesort($array);
- natsort($array); Used to sort in natural order
- array_multisort($array); → Used to sort multiple array
- array_reverse($array); → Used to sort array in reverse order

21. Array traversing function :

- This function is used to pick different value of an array by moving the pointer.
- There are 8 different type of traversing function:
  1. next($array); →Use to jump pointer to the next value.
  2. prev($array); → Use to jump pointer to the previous value.
  3. end($array); → Use to jump pointer to the end value.
  4. each($array); →Gives exact value and key of current pointer position.
  5. pos($array); → Gives current position of pointer.
  6. current($array); →Gives the value of current position of pointer.
  7. key($array); → Gives index value of current position of pointer.
  8. reset($array); →reset pointer position to 0 index.

22. Array list() :

- List() function is only works with index array or associative array whose keys are numeric.

Syntax

list($a, $b, $c) = $array;

Here, $a, $b, $c are the variable taken according to the values present in $array.

23. Extract() & Compact() :

- Extract() is used to extract/change the key of an array with variables.
- Compact() is use to make an array using different variables.

Syntax

- extract($array);
- compact(var1, var2, var3, …….);

24. Range Function :

- It is use to print a range of value between two values.

Syntax

range(start, end, step);

Here, **step** is use for incrementation from start value. Its default value is 1.

25. Explode & Implode Function :

- Explode() function is use to convert string value into an array.
- Implode() function is use to convert an array values into a string.

Syntax

- explode(separator, string, limit);
- implode(separator, $array);

STRING FUNCTIONS

1. Str_split() & Chunk_split() :

➢ Str_split() is used to split every character of a string.

- Chunk_split() is used to split the character by adding some new character in that string.

Syntax

- str_split($string, length);
- Chunk_split($string, length, end);

2. Upper case & Lower case:

- There are basically 5 type of functions :-
  - ✓ strtolower($string); →Convert string to lower case.
  - ✓ strtoupper($string); → Convert string to upper case.
  - ✓ ucfirst($string); → Convert first character of string to upper case.
  - ✓ ucwords($string); → Convert first character of every word in string to upper case.
  - ✓ lcfirst($string); → Convert first character of string to lower case.

3. String length & count function :

- Strlen() is use to count length of the string.
- Str_word_count() is use to count words of the string.

Syntax

- strlen($string);
- str_word_count($string,return);

4. String find position function :

- There are 4 type of function to find string :-
  - ✓ strpos($string, find, start);
  - ✓ strrpos($string, find, start); **case-sensitive**
  - ✓ stripos($string, find, start);
  - ✓ strripos($string, find, start); **case-insensitive**
- They all returns a position of the string.

5. String search function :

- There are 5 type of function to search string inside a string :-
  - ✓ strstr($string, search, before_search);
  - ✓ strchr($string, search, before_search); **case-sensitive**
  - ✓ strrchr($string, search);

- ✓ stristr($string, search, before_search); **case-insensitive**
- ✓ strpbrk($string, charlist); → Use to find character form string.

➢ They all returns a string from a string.

6. String substr() :

➢ It is use to cut out string from a string.

Syntax

- substr(string, start, length);

7. String replace function :

➢ There are 4 type of functions to replace string.

- ✓ str_replace(find, replace, $string); →**case-sensitive**
- ✓ str_ireplace(find, replace, $string); →**case-insensitive**
- ✓ substr_replace($string, replacement, start, length); → Use to replace string by sub string
- ✓ strtr($string, from, to); → Use to replace characters of a string.

8. String compare function :

➢ There are 8 different functions to compare string.

- ✓ strcmp($string1, $string2); → **case-sensitive**
- ✓ strncmp($string1, $string2, length);
- ✓ strcasecmp($string1, $string2); → **case-insensitive**
- ✓ strncasecmp($string1, $string2, length);
- ✓ strnatcmp($string1, $string2);
- ✓ strnatcasecmp($string1, $string2, length);
- ✓ substr_compare($string1, $string2, start, length, case);
- ✓ similar_text($string1, $string2, percent);

9. Strrev() & Str_shuffle() :

➢ Strrev() is used to reverse the string.

➢ Str_shuffle() is used to shuffle the character of string.

Syntax

- strrev($string);
- str_shuffle($string);

10. Str_pad() & Str_repeat() :

➢ Str_pad() is used to increase the length and fill the space of string.

- Str_repeat() is used to repeat the word.

Syntax

- str_pad($string, length, pad_string,pad_type);
- str_repeat($string, repeat);

Here, **pad_string** is for symbol/character and **pad_type** is for deciding

the side from where padding has to start. By default its value is right.

11. String trim function :

- This function is used to trim word from either side.
- There are 4 type of function for trim.
  - ✓ trim($string, charlist); →Use to trim word in a string.
  - ✓ ltrim($string, charlist); → Use to trim word from left side of the string.
  - ✓ rtrim($string, charlist); → Use to trim word right side of the string.
  - ✓ chop($string, charlist); → Similar to rtrim().

12. Addslashes & Stripslashes :

- Addslashes is used to add slashes in string to protect data from **SQL injection** while saving in database.
- Stripslashes is used to remove slashes while printing data from database.

Syntax

- addslashes($string);
- stripslashes($string);

Note :-

a) addcslashes($string, characters); → Used to add slashes before character.
b) stripcslashes($string); → Use to remove slashes before characters.

13. Htmlentities() & Htmlspecialchar() :

- Htmlentities used to convert string symbol by entity codes like < →&lt, > → &gt.

- Htmlspecialchar works similar to htmlentities but convert only limited character.

Syntax

- htmlentities($string, flags);
- htmlspecialchar($string, flags);

Note :

- ✓ html_entity_decode($string, flags); → Used to convert entity code to symbol.
- ✓ htmlspecialchar_decode($string, flags); → Used to convert special character code to characters.

Flags –

- ENT_COMPAT → Default, encodes only double quotes.
- ENT_QUOTES → Encodes double & single string.
- ENT_NOQUOTES → Does not encode any quotes.

HTML Entities –

- & → &
- " → "
- ' → '
- < → <
- → >

14. Md5 & Sha1 :

a) Md5 -

- md5 works on Message-digest algorithm.
- Used to secure password which are saved in database.

Syntax

- md5($string, raw);

Here, ***raw*** has two values :

i. true → Binary format of 16 characters.

ii. False → Default, hex number of 32 characters.

b) Sha1 –

- sha1 works on US Hash Algorithm 1.
- It is more secure than md5.

Syntax

- sha1($string, raw);

Here, ***raw*** has two values :

i. true → Binary format of 20 characters.
ii. False → Default, hex number of 40 characters.

15. Convert_uuencode() & Convert_uudecode() :

- convert_uuencode() is use to encrypt message. It works on **uuencode algorithm.**
- convert_uudecode() is use to decrypt message.

Syntax

- convert_uuencode($string);
- convert_uudecode($string);

16. bin2hex() & hex2bin() :

- bin2hex() converts ASCII characters to hexadecimal values.
- hex2bin() convert hexadecimal values to ASCII characters.

Syntax

- bin2hex($string);
- hex2bin($string);

17. Chr() & Ord() :

- chr() converts ASCII characters to character.
- ord() gives ASCII value of first character.

Syntax

- chr(ascii);
- ord($string);

18. Strip_tag() & Wordwrap() :

- strip_tag() used to remove all styles from string.
- wordwrap() used to break string.

Syntax

- strip_tag($string, allow);
- wordwrap($string, width, break, cut);
  - true → Wrap
  - false → No wrap

19. Maths :

Syntax

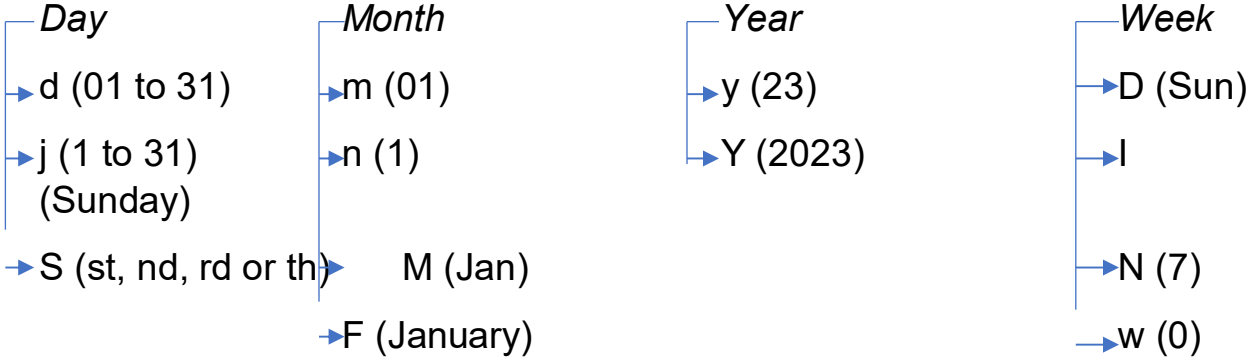
- min(value1, value2, value3, ……) **OR** min(array_value) → Minimum value
- max(value1, value2, value3, ……) **OR** max(array_value) → Maximum value
- abs(value) → Absolute value
- ceil(value) →Greater than or equal to
- floor(value) →Less than or equal to
- round(value) → Round-off
- intdiv(dividend, divisor) → Division
- pow(base, exp) → Power
- sqrt(number) → Square
- rand(min, max) → Random number between min and max
- mt_rand(min, max) → Random number between min and max
  - Works on ***Mersenne Twister Algorithm.***
  - 4 time faster than rand() function.
  - lcg_value() → Gives decimal value between 0 and 1.

20.Date function :

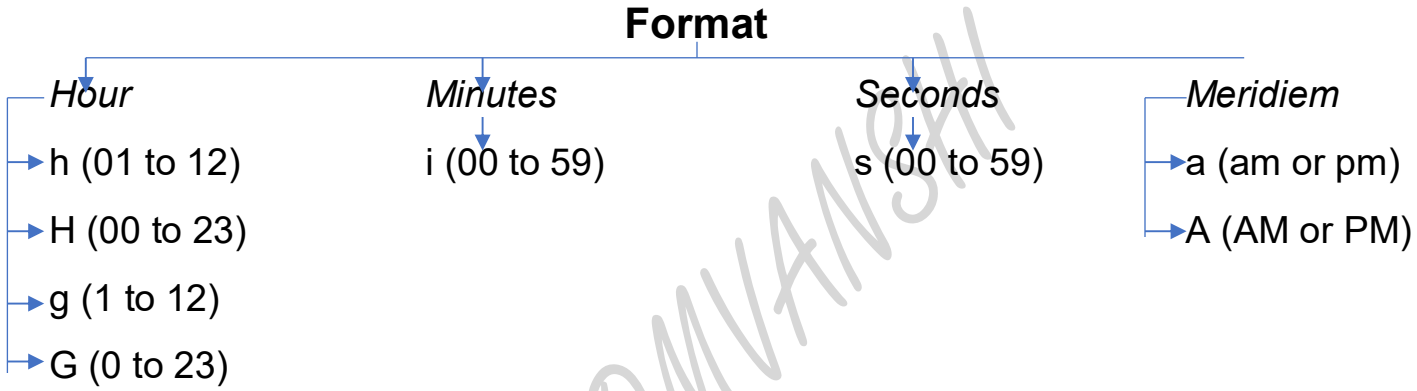
Syntax

- date(format);

Format



21. Time function :



Syntax

- date(format, timestamp);

Timestamp :-

- ✓ mktime(hour, minute, second, month, day, year);
- ✓ gmmktime(hour, minute, second, month, day, year); → Works on GMT (Greenwich Mean Time);

22.date_create() & date_format() :

Syntax

- date_create(time, timezone);
  - timezone_open() → optional
- date_format(date_create(), format);

23.Checkdate() & Date_diff() :

Syntax

- checkdate(month, day, year);
- date_diff(datetime1, datetime2);

Include & Require Statement :

PHP {HYPERTEXT PREPROCESSOR}

- Include is use to merge a php file to another php file. If a file has error then except that file all file will execute.
- Require works similar to include but if a file has error then error will shown up and no next file will execute.

Syntax

- include'filename'
- require'filename'

Include_once & Require_once :

- include_once & require_once are use to reduce the chance of duplicacy.

Syntax

- include_once'filename'
- require_once'filename'

SUPER GLOBAL VARIABLE

Super Global Variable are those whose value we can use in other file.

There are 7 different types of Super Global Variable :-

1. $_GET → Used to get user data.
2. $_POST → Used to keep user input in other file to save in database or to print.
3. $_REQUEST → It works with both $_GET and $_ POST
4. $_SERVER → Used to retrieve extra data from a form. Save data in same page database.
5. $_COOKIE → Use to save temporary information in user device.
   - Create Cookie -
     - setcookie(name, value, expire, path, domain, secure, httponly)
   - View Cookie Value -
     - $_COOKIE[name]
6. $_SESSION → Use to save temporary information in server. Use in login pages.
   - Set & Get SESSION Value –

- session_start();
- $_SESSION[name] = value; → Set Session name & value
- echo $_SESSION[name]; → Get Session value
- Delete Session –
  - session_unset(); → Remove all session variables
  - session_destroy(); → Destroy the session

7. $_FILES → Use to uplode user file (image, document, pdf, etc) to server.
   - $_FILES['value']
     - Name
     - Size
     - tmp_name
     - Type
   - move_uploded_file(file, dest)

die() & exit() :

- This are used to stop the flow of php code.
- Used to rectify the error.

Syntax

- die();
- exit();

PHP {HYPERTEXT PREPROCESSOR}

- MySQL server is a open-source relational database management system which is a major support for web application.
- Databases and related tables are the main component of many websites and applications as the data is stored and exchanged over the web.

Types of MySQL:

1. Relational –
   - All the data is stored in the form of table.
   - All the tables are linked to each other.
   - This is called ***RDBMS*** (Relational Database Management System)***.***
   - RDBMS use **SQL** (Structured Query Language).
   - Example :- Oracle, MS SQL Server, MySQL, PostgreSQL, etc.
2. NoSQL -
   - All the data is stored in the form of document.
   - Example :- MongoDB, Redis, Cassandra, etc.

Advantages of MySQL :

- Cross platform.
- Used with multiple languages (PHP, NodeJS, Python, C#).
- MySQL software is open source.
- MySQL is RDBMS.
- MySQL database server is fast, reliable, scalable and easy to use.
- MySQL server works in client/server or embedded systems.

Popular Websites using MySQL:

Facebook	Twitter	GOOGLE
Wikipedia	YouTube	Pinterest
	Flickr	

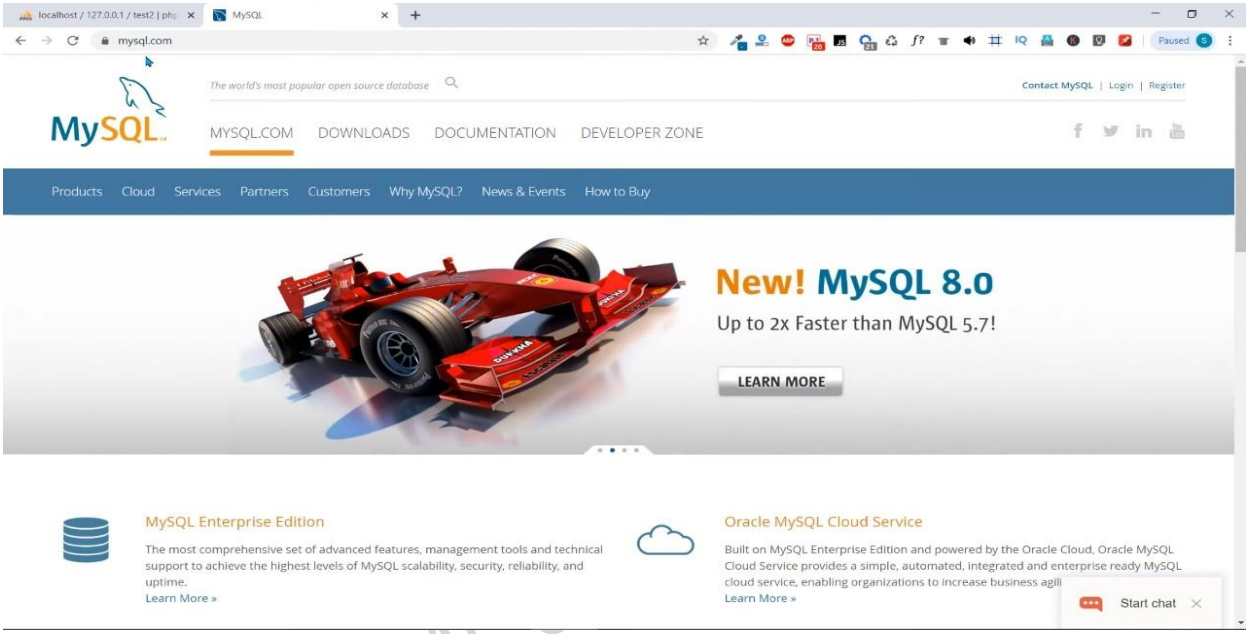
Content Management System using MySQL :

PHP {HYPERTEXT PREPROCESSOR}

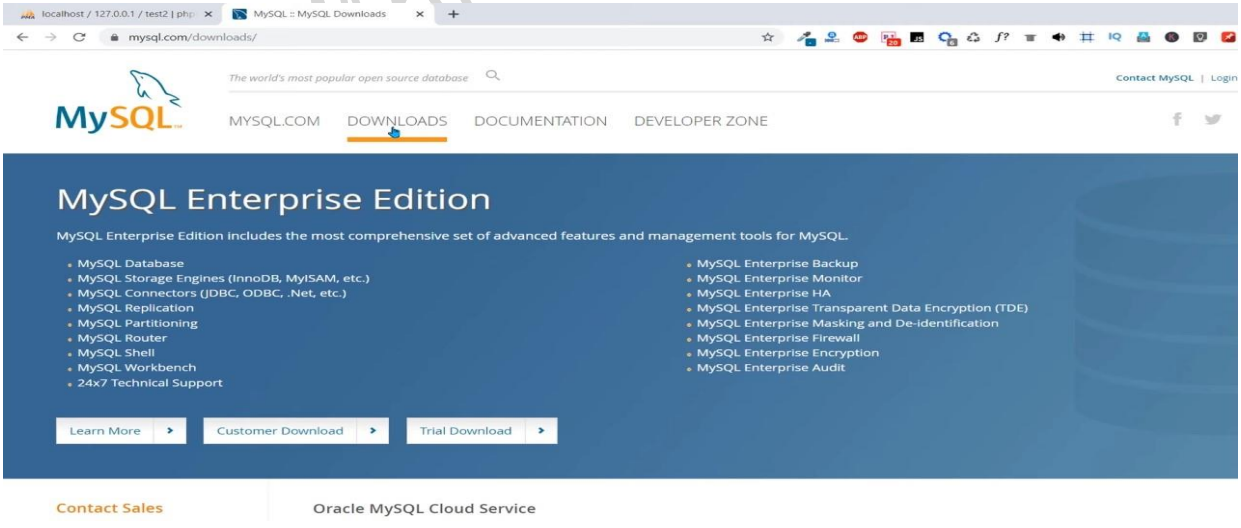
Wordpress	Joomla	Drupal
Magento	phpBB	TYPO3
	MODx	

Installation of MySQL Workbench :

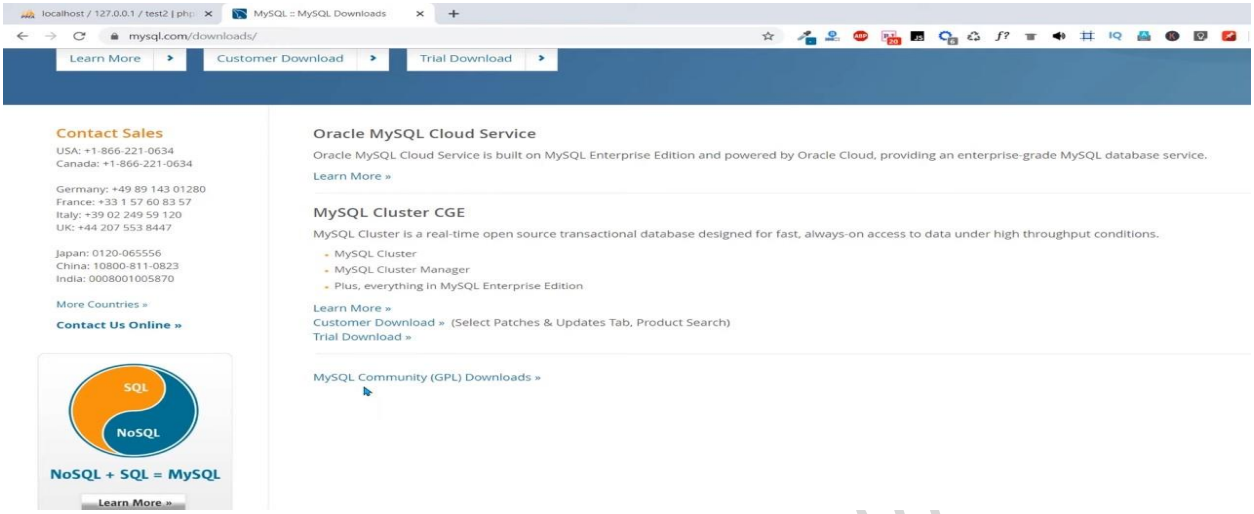
i. Search **mysql.com** on browser.



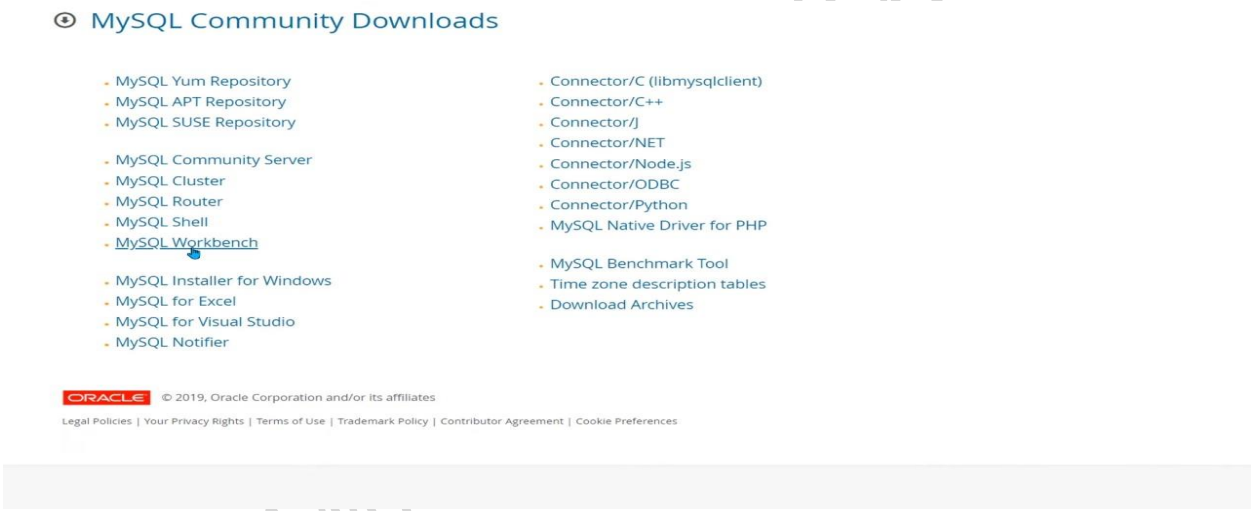
ii. Click on **Download**.



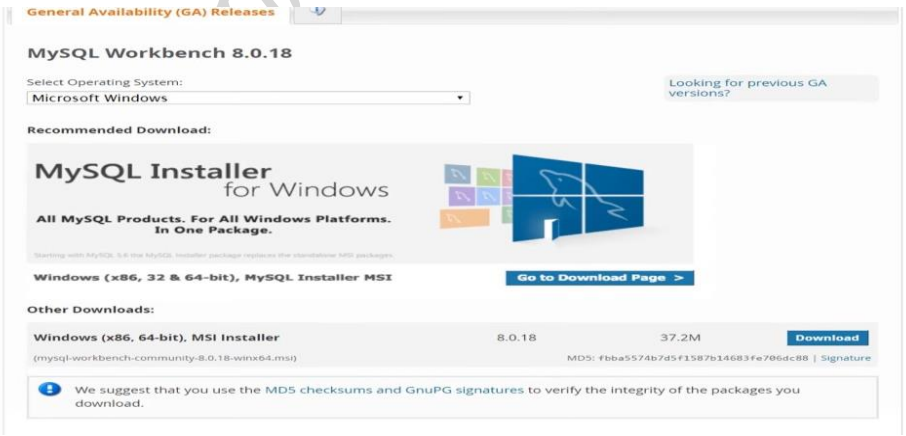
iii. Click on MySQL Community (GPL) Downloads.



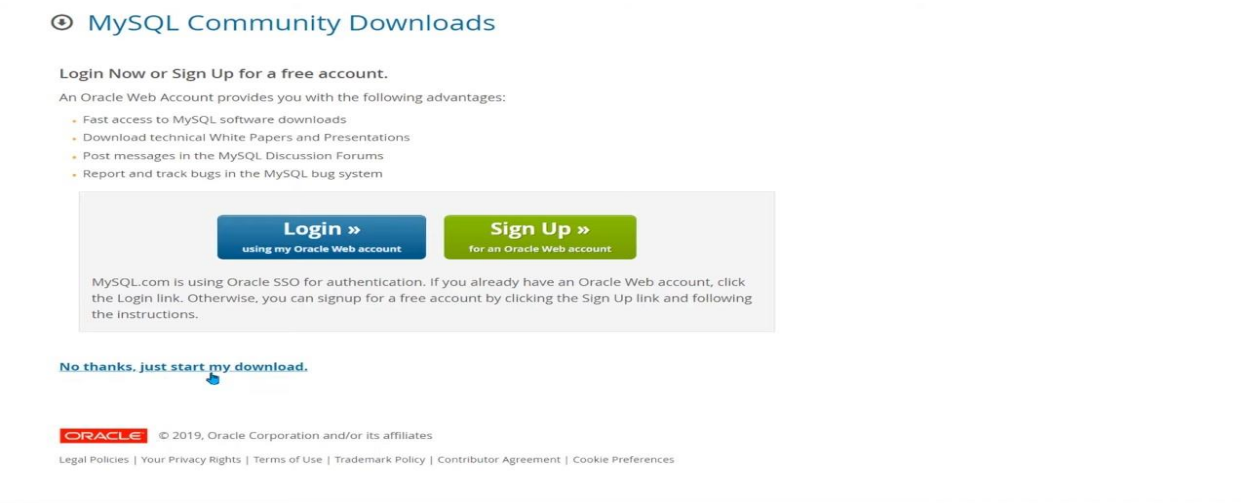
iv. Click on MySQL Workbench.



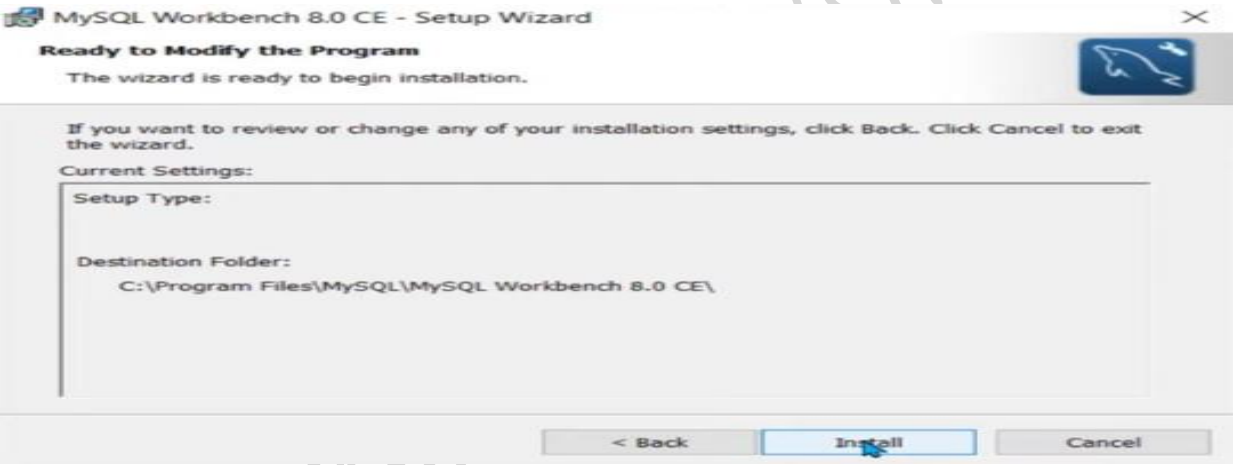
v. Select Operating System and click on download.



vi. Now, click on **No thanks, just start my download**.



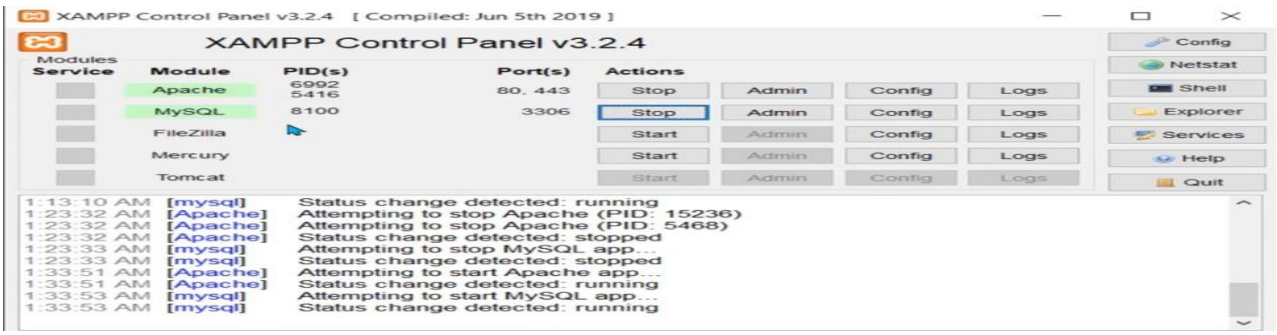
vii. Save the file and install it.



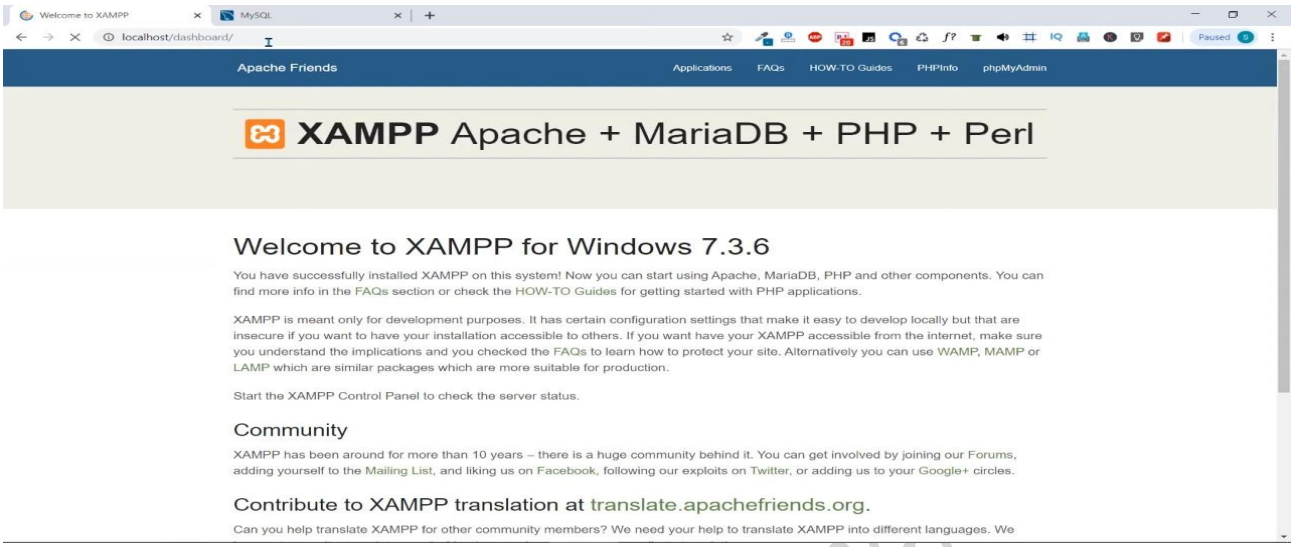
Creating database in MySQL:

1) Method 1 –

a) Turn ON Apache and MySQL from XAMPP Control Panel.

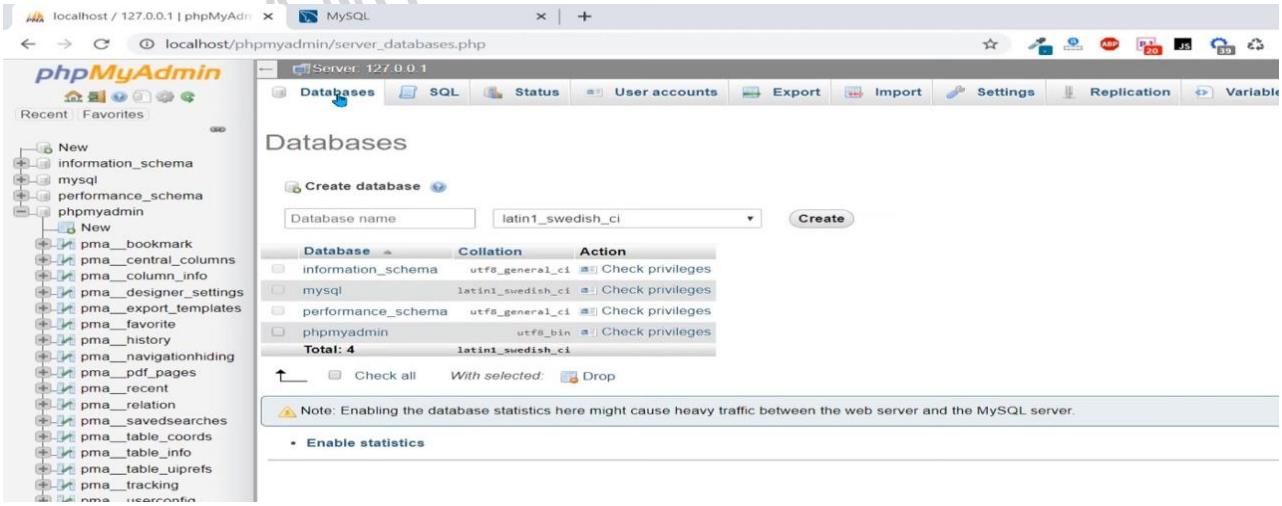


b) Search localhost on browser.

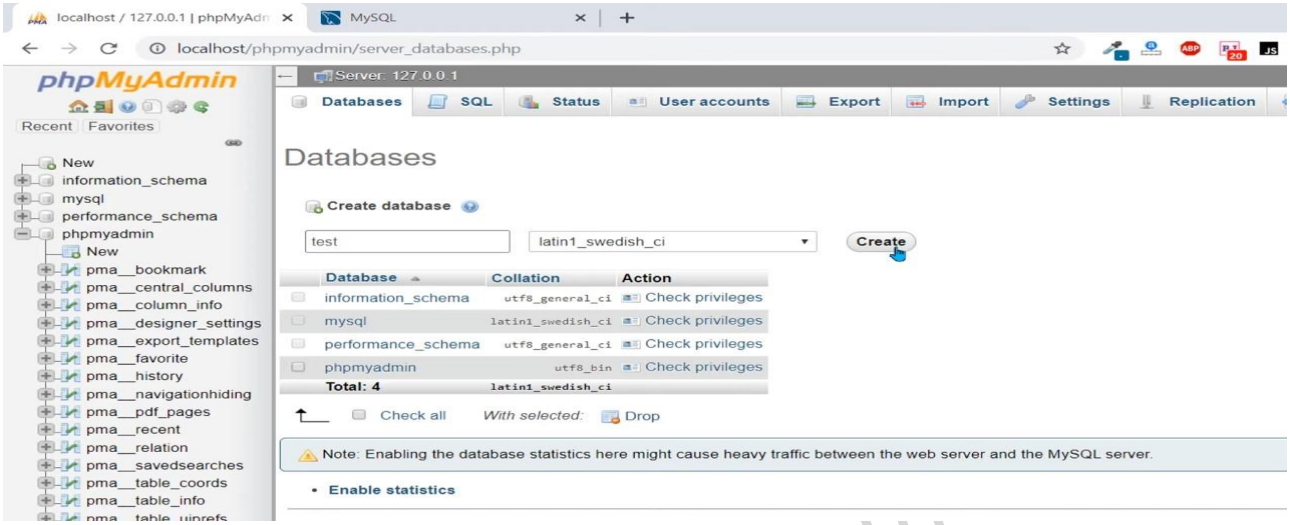


c) Click on **phpMyAdmin** in navigation bar.

d) Click on **Database**.

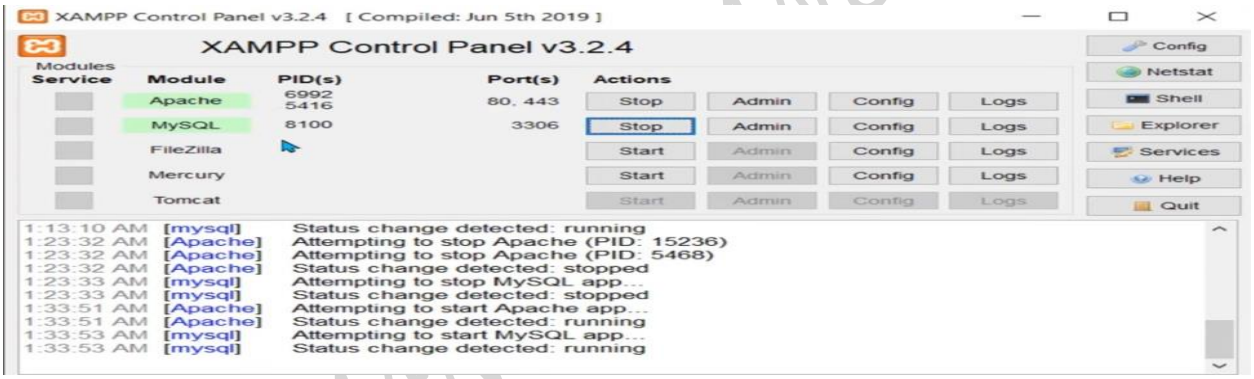


e) Give the name of database in create database then click on **create**.

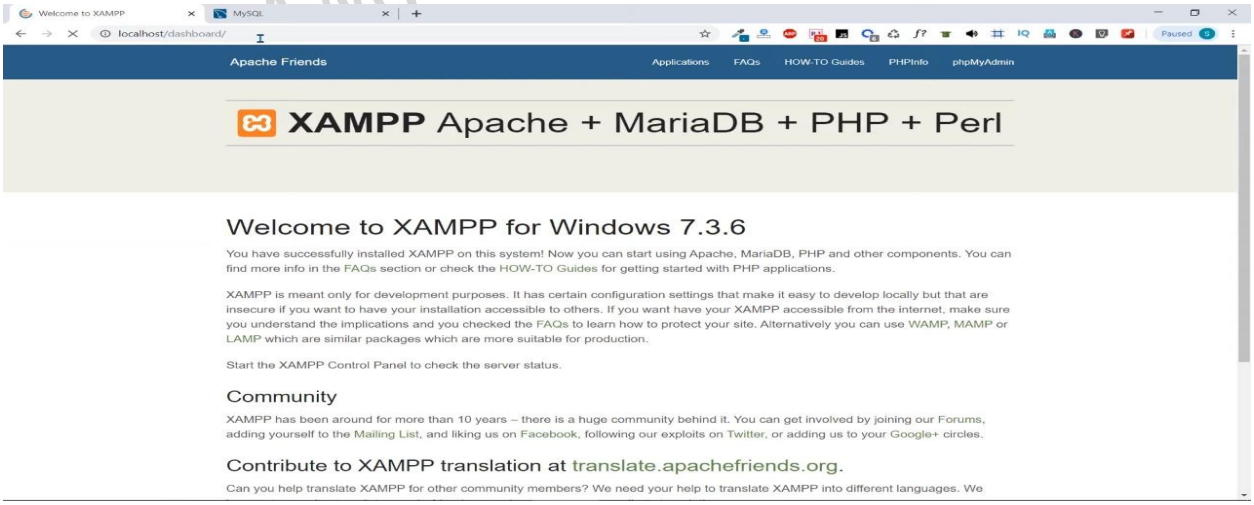


2) Method 2 -

a) Turn ON Apache and MySQL from XAMPP Control Panel.



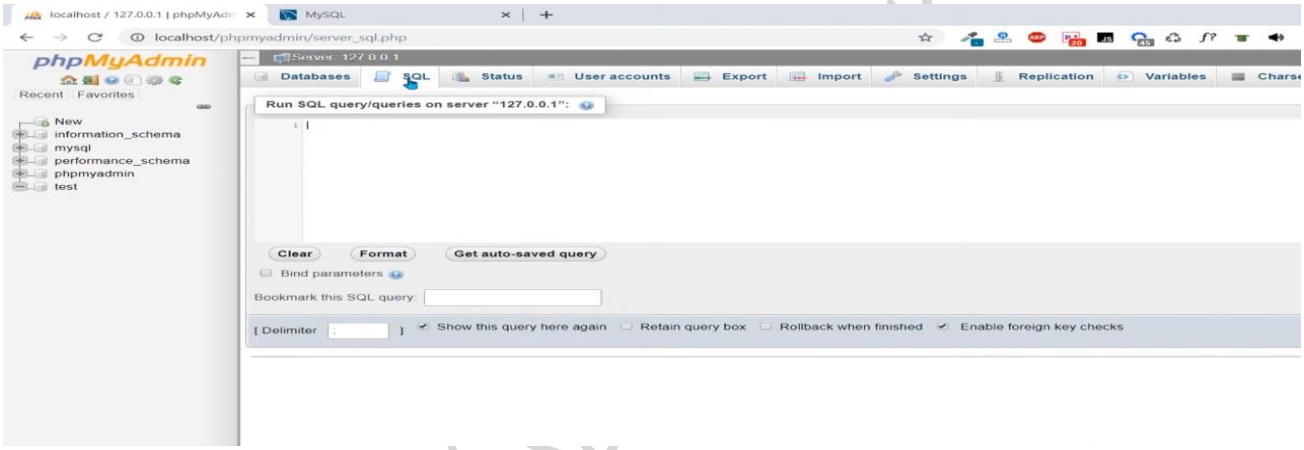
b) Search localhost on browser.



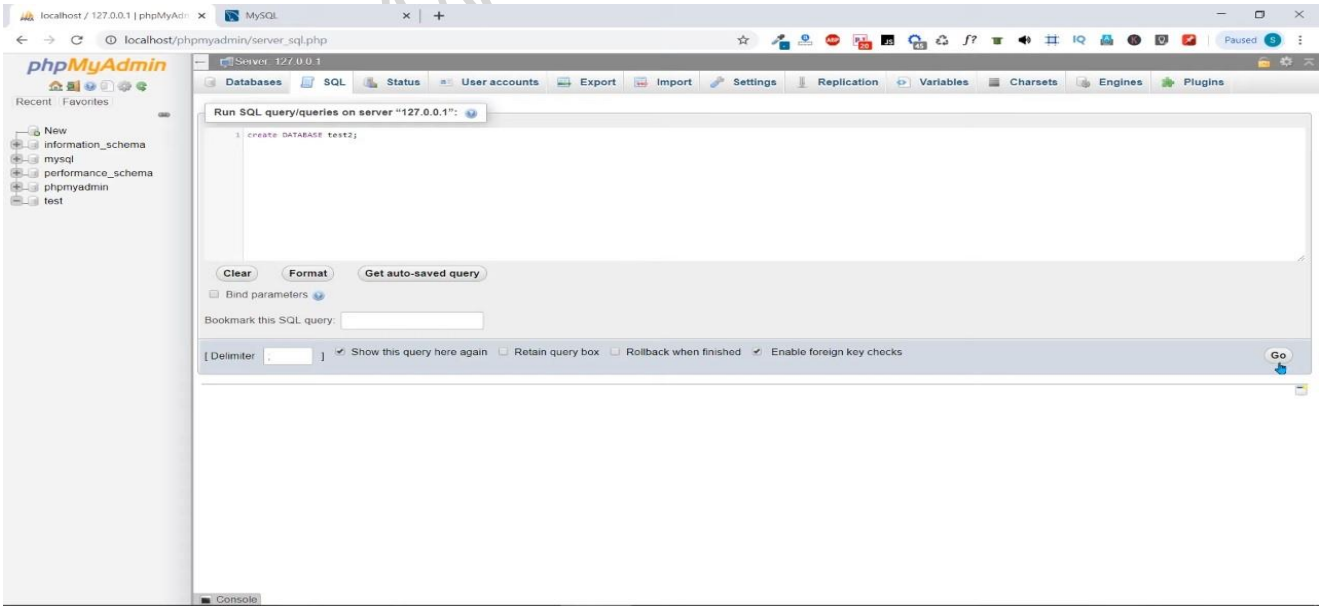
c) Click on **phpMyAdmin** in navigation bar.

PHP {HYPERTEXT PREPROCESSOR}

d) Click on **SQL**.



e) Write a command **create DATABASE** name**;** then click on **GO**.



Create SQL table :

```
CREATE TABLE table_name(
column1 datatype,
column2 datatype,
column3 datatype,
.....
);
```

Datatype in MySQL :

1. String
   - CHAR(size) → 0 to 255
   - VARCHAR(size) → 0 to 65535
   - BINARY(size)
   - VARBINARY(size)
   - TINYTEXT → 255 characters
   - TEXT(size) →65535 bytes
   - MEDIUMTEXT → 16777215 characters
   - LONGTEXT → 4294967295 characters
   - TINYBLOB → 255 bytes
   - BLOB(size) → 65535 bytes
   - MEDIUMBLOB → 16777215 bytes
   - LONGBLOB → 4294967295 bytes
   - ENUM(val1, val2, val3, ….) → list up to 65535 values
   - SET(val1, val2, val3, ….) →list up to 64 values
2. Numeric
   - BIT(size) → 1 to 64
   - TINYINT(size) → -128 to127
   - INT(size) → - 2147483648 to 2147483647
   - INTEGER(size)
   - SMALLINT(size) → - 32768 to 32767
   - MEDIUMINT(size) → - 8388608 to 8388607
   - BIGINT(size) → - 9223372036854775808 to 9223372036854775807
   - BOOL
   - BOOLEAN → 0/1

- FLOAT(p)
- DOUBLE(size, d) → 255.568
- DECIMAL(size, d) → size = 60, d = 30
- DEC(size, d)

3. Date & Time
   - DATE → 1000 - 01 – 01 to 9999 – 12 - 31
   - DATETIME(fsp) → YY – MM – DD hh:mm:ss
   - TIMESTAMP(fsp)
   - TIME(fsp) → hh:mm:ss
   - YEAR → four-digit format : 1901