# ALAGAPPA UNIVERSITY

**(Accredited with 'A+' Grade by NAAC (with CGPA: 3.64) in the Third Cycle and Graded as category - I University by MHRD-UGC)**
**(A State University Established by the Government of Tamilnadu)**

**KARAIKUDI – 630 003**

## DIRECTORATE OF DISTANCE EDUCATION

## M.Sc. (INFORMATION TECHNOLOGY)

**Second Year – Third Semester**

# 31334 – Open Source Lab

**Copy Right Reserved** **For Private Use only**

**Author:**
**Dr. M. Illayaraja**
Assistant Professor
Department of computer Application Technology,
Kalasailingam Academy of Research and Education.
Krishnankoil-626126

**The Copyright shall be vested with Alagappa University"**

**All rights reserved. No part of this publication which is material protected by this copyright notice may be reproduced or transmitted or utilized or stored in any form or by any means now known or hereinafter invented, electronic, digital or mechanical, including photocopying, scanning, recording or by any information storage or retrieval system, without prior written permission from the Alagappa University, Karaikudi, Tamil Nadu.**

# LAB PROGRAMMING IN OPEN SOURCE SOFTWARE

**Syllabi** **Page No.**


**BLOCK 1**

1. Kernel Configuration, compilation and installation **1-4**
2. Install various software on Linux, Install and Configure XAMP,
3. Unix Commands and Shell Programming **5-25**

**BLOCK 2**

4. Creating simple table with constraints **26-27**
5. Demonstration of joining tables **28-40**
6. Database connectivity in PHP with MYSQL **41-77**

**BLOCK 3**

7. PHP Simple Programs **77-79**
8. PHP Web programs arrays and functions **80-95**
9. File manipulation using PHP **95-104**

**BLOCK 4**

10. PERL programs **105-117**
11. Python Programming **118-131**
12. Python Programming: String **132-139**
13. Python Programming: Arrays **140-157**

**BLOCK 5**

14. Connect to a MYSQL database with PHP, PERL and Python **157-183**
    Developing simple applications using PHP and MYSQL

*Notes*

# 1. KERNEL CONFIGURATION, COMPILATION AND INSTALLATION

## AIM:

To learn how to configure, compile and install Linux kernel from source.

**Configure, compile and install the latest kernel from source.**

## INTRODUCTION:

The Linux kernel in the distributions are configured to work correctly in a wide variety of hardware and there is usual y no need to use any other kernel.

A user may want to rebuild the kernel for various reasons. The main reason was once to optimize the kernel to the environment (hardware and usage patterns). With modern hardware there is rarely a need to recompile unless there is a particular feature of a new kernel that is required. The performance gains are probably not noticeable unless specific benchmarks are being run.

## DESCRIPTION:

Students willl compile a custom kernel using the new kernel source available in the FOSS Lab server. The students should be able to boot the system using the newly compiled kernel.

## Pre-requisites:

The latest kernel source from the FOSS Lab server. It is located in http://<fosslab-server ip>/content/packages/Linux_Kernel/v2.6/linux-2.6.39.2.tar.bz2

## The exercise:

All actions are performed as root.
> su
#

We need to ensure that all tools required for compiling the kernel are installed.

# yum install kernel-devel

This command willl ensure that all packages required to compile the current running kernel willl be installed. We willl be using the same tools to compile the newer custom kernel.

*Notes*

Remove traces of old kernel source if they exist. Be very carefull with the rm command as you can completely trash the system if you are careless.

# rm -rf /usr/src/linux/
# rm -rf /usr/src/linux-2.6/
The kernel source is usual y kept under /usr/src. Copy the downloaded kernel source to /usr/src.

# cp linux-2.6.39.2.tar.bz2 /usr/src
# cd /usr/src/
# tar -xjvf linux-2.6.39.2.tar.bz2
# cd linux-2.6.39.2

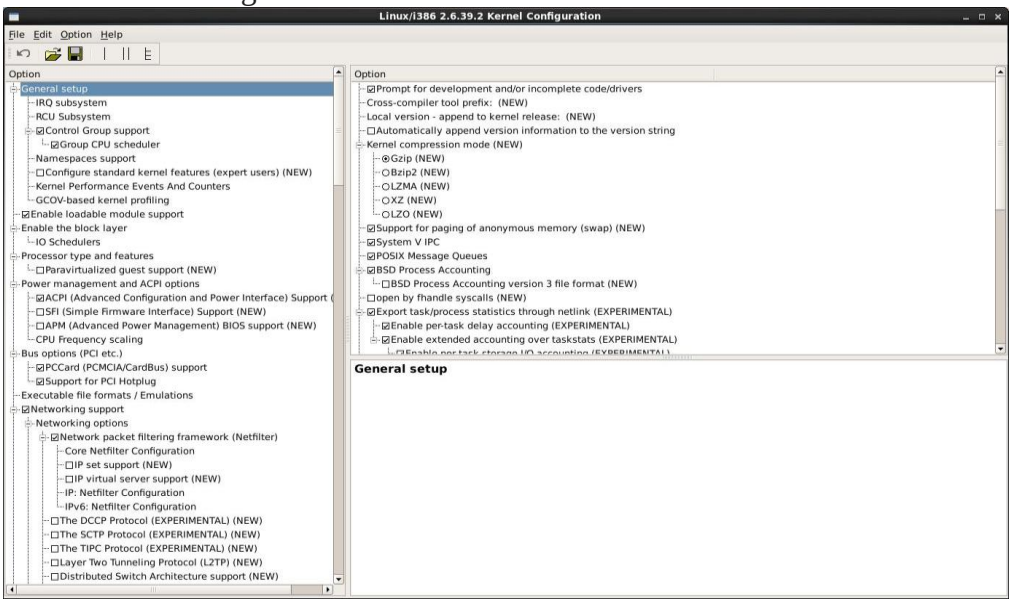
We now create two symlinks to the kernel tree.
# ln -s /usr/src/linux-2.6.39.2 /usr/src/linux
# ln -s /usr/src/linux-2.6.39.2 /usr/src/linux-2.6

Now we clean out all previous configurations and reset the source directory to a pristine state. The main reason for doing this is that some files do not automatically get rebuilt, which can lead to failed builds, or at worst, a buggy kernel

# make mrproper

Now we configure the kernel. The build system is intelligent enough to take most of the current configuration from the currently running kernel. There are thousands of options and usually the current options willl be suitable to create a working kernel. We can experiment with modifying the kernel options after compiling the kernel successful y with the default configuration.

# make xconfig
(or)
# make menuconfig



Save and exit the tool.
Now we are ready to build the kernel.

```
# make clean
# make all
```

This can take up anywhere from 10 minutes to upto 2 hours depending on the hardware. Once the compilation is completed we can instal the kernel and its modules.

```
# make modules_install
# make install
```

The newly created kernel will be in /boot
Now we need to check that the instal process has configured the boot loader to point to the new kernel.

# vi /boot/grub/menu.lst

The new kernel will have an entry at the top of the kernel list. It can be identified by the kernel version number.Change the lines containing default, timeout to the values shown and comment out the hidden menu entry.

```
default=0
timeout=5
#hiddenmenu
```

Now reboot the computer and the computer will boot into the new kernel. If it fails, reboot the machine and select the previously running kernel to boot successful y and redo the exercise careful y.

To check the version of the running kernel, use the uname command.

```
# uname -r
2.6.39.2
```

Now the process can be repeated with different kernel configuration options.

## Result:

Thus to download / access the latest kernel source code from kernel.org, compiling the kernel and install it in the local system and trying to view the source code of the kernel is done successfully.

# 2 . INSTALL VARIOUS SOFTWARE ON LINUX

## AIM:

To learn and install various software's in Linux platform

## Procedure

A package manager is a sub-system on Linux that, as the title says, manages the packages (software) on your computer. It's a crucial component of Linux, in that it keeps track of everything installed; downloads packages; ensures all packages are installed in a common location; helps to upgrade packages; resolves dependencies; and keeps users from having to install from source code.

The biggest point of confusion is that there are numerous package managers available, but only one can be used on a distribution. In fact, distributions are differentiated, primarily, on which package manager they choose. For example: Debian and Ubuntu (and its derivatives) use apt; Red Hat Enterprise Linux, CentOS, and Fedora use yum; SUSE and openSUSE use zypper; and Arch Linux uses pacman. There are more package managers out there, but this is a good place to start.

Each package manager works with a different file type. For example, apt works with .deb files and yum and zypper work with .rpm files. The apt package manager cannot install .rpm files and neither yum or zypper can install .deb files. To make matters even more confusing, Ubuntu (and its derivatives) uses the dpkg command for installing local .deb files, and Red Hat (and its derivatives) use the rpm command to install local .rpm files.

Most package managers have GUI front ends. These front ends are similar to the Apple App Store. It should be no surprise that there are numerous such GUI front ends available. The good news here is that most of them are similarly titled (such as GNOME Software, Ubuntu Software, Elementary AppCenter). These app stores allow you to easily search for a software title and install it with the click of a button (more on this in a bit).

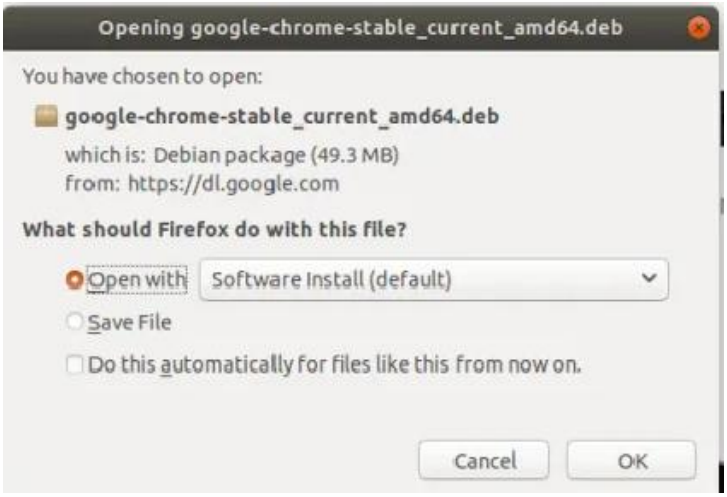
There is one other issue with package managers: repositories. Repositories are a key aspect of package managers, but for new users the concept can add yet another level of confusion we don't want. For a quick overview, however, consider this: Out of the box you only have a certain selection of software titles available. That selection is dictated by the repositories that are configured. There are numerous third-party repositories you can add to the system. Once added, you can then install any software titles associated with those third-party repositories. Software repositories can be added either from a GUI tool or the command line.

In any case, repositories are an issue for a different day, and not necessary to understand for the type of software downloads discussed in this article.

## Installing a downloaded file

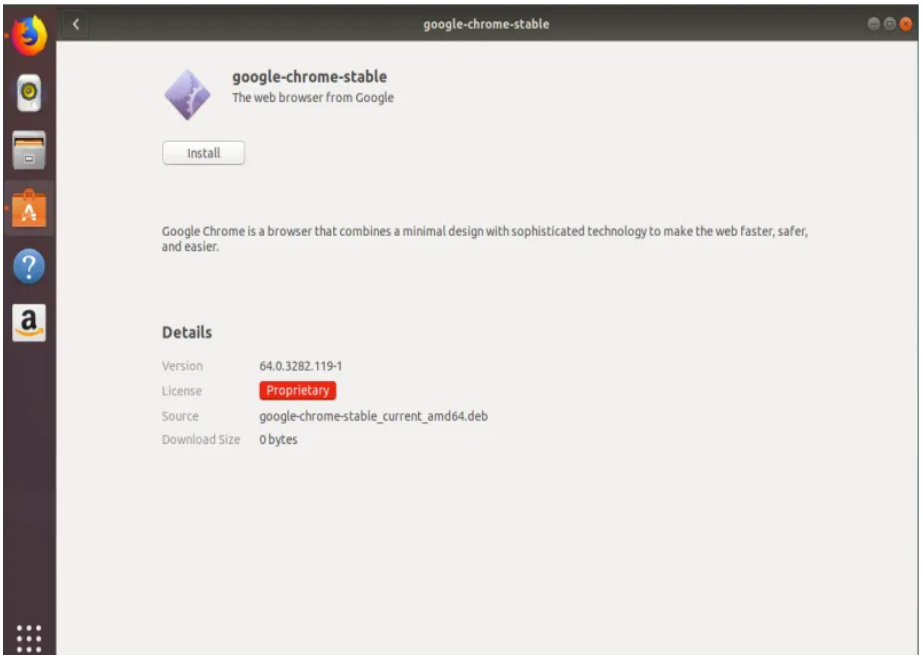
I know, I know… I said one of the benefits of modern Linux operating systems is that you don't need to install from a downloaded file. That being said, I want to start here. Why? There may be times when you find a piece of software not available in your distribution's "app store." When that occurs, you'll need to know how to install that application manually. I will say that, for every day, average use, it's a rare occasion that you'll need to do this. And even if you never do install using this method, at least you'll have a very basic understanding of how it works.

Here, we'll demonstrate using the latest release of Ubuntu Linux (as of this writing, 17.10). Most package managers install in similar fashion (with slight variations on the commands used). Let's say you want to install the Google Chrome browser on Ubuntu. You won't find this particular browser in the Ubuntu Software tool. To install it from the command line, you must download the correct file. As stated earlier, the correct file for Ubuntu will be a .deb file. So point your browser to the Chrome download page and click the Download Chrome button. The good news here is that your browser will be detected and the Chrome download page will know which file you need. Click the ACCEPT AND INSTALL button and a new window will appear, giving you two options




You can either save the file to your hard drive (and then install via the command line), or open the file with the Software Installer. It is important to understand that not every distribution includes the latter. If you do not get the Open with option, then you'll have to install from the command line.

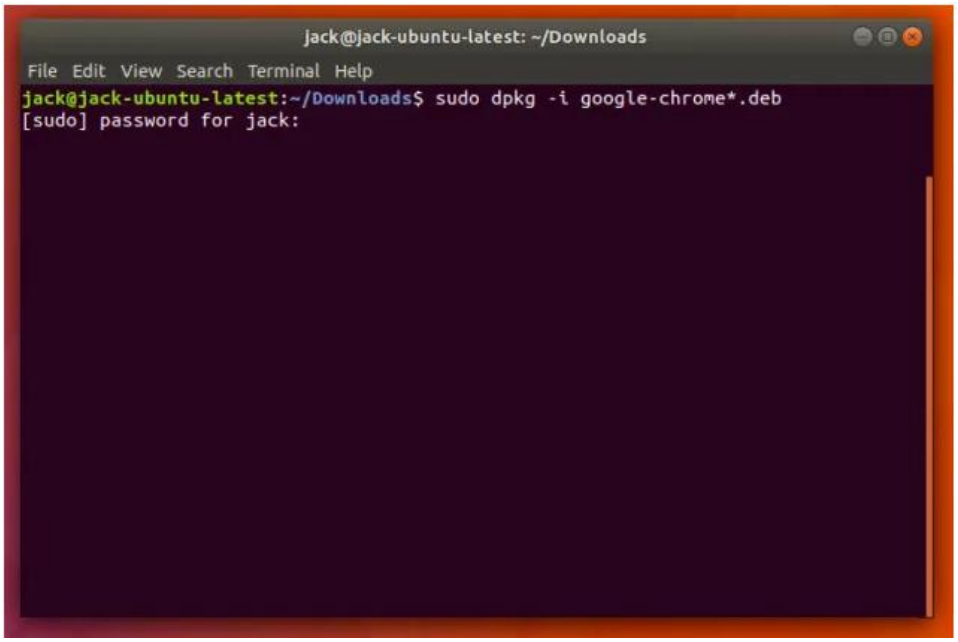
Let's first use the Open with option. Make sure Software Install (default) is selected and click OK. The file will download and then Ubuntu Software will open, giving you the option to install




Click Install and you will be prompted for your user password. The installation will complete and Chrome is ready to use. You can close the Ubuntu Software tool and open Chrome from the Dash.

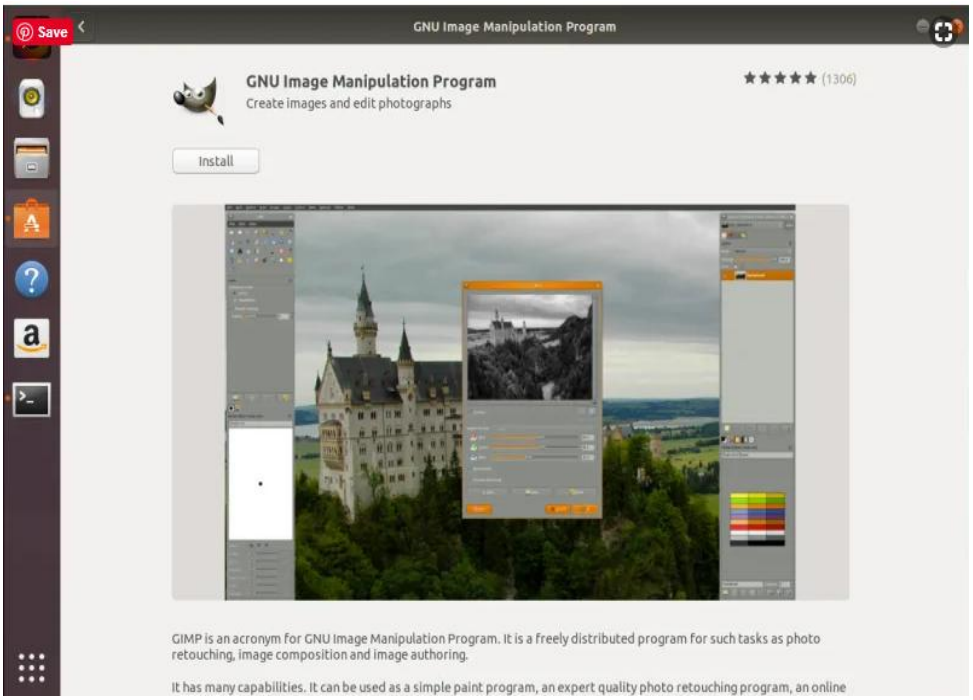
But what if you don't get the option to install with the GUI tool? Then you have to select the Save File and run the installation from the command line. Don't worry, it's not that hard. Here are the steps to install the latest release of Chrome, on Ubuntu Linux, from the command line:

1. Click on the square of dots at the bottom of the desktop
2. When the Dash opens, type terminal
3. Change into the Downloads directory with the command cd ~/Downloads
4. Install Chrome with the command sudo dpkg -i google-chrome*.deb
5. When prompted (see below), type your user password and hit Enter on your keyboard
6. Allow the installation to complete




**Installing from the GUI**

This is where things get very easy. To install from your distribution's GUI, you only need open up the tool, search for the software you want, and click Install. Say, for instance, you want to install the GIMP Image editor. To do that, open Ubuntu Software and type gimp in the search bar. When the results appear, click on the GIMP entry, click the Install button (see below), and (when prompted) type your user password. Wait for the installation to complete and your new software is ready to be opened and used.



## Bottom line: it's all easier than it seems

Installing software on Linux isn't nearly as hard as you might have thought. Yes, there may be the rare occasion when you need to install something from the command line, but even that isn't much of a challenge. Besides, chances are, you'll never have to install software outside of the GUI front end.

Do remember, if you use a distribution other than Ubuntu (or its derivatives), you'll want to do a quick bit of googling to make sure you understand the differences between the apt package manager and the one used on your desktop.

## Result:

Thus the software's on Linux are installed

---

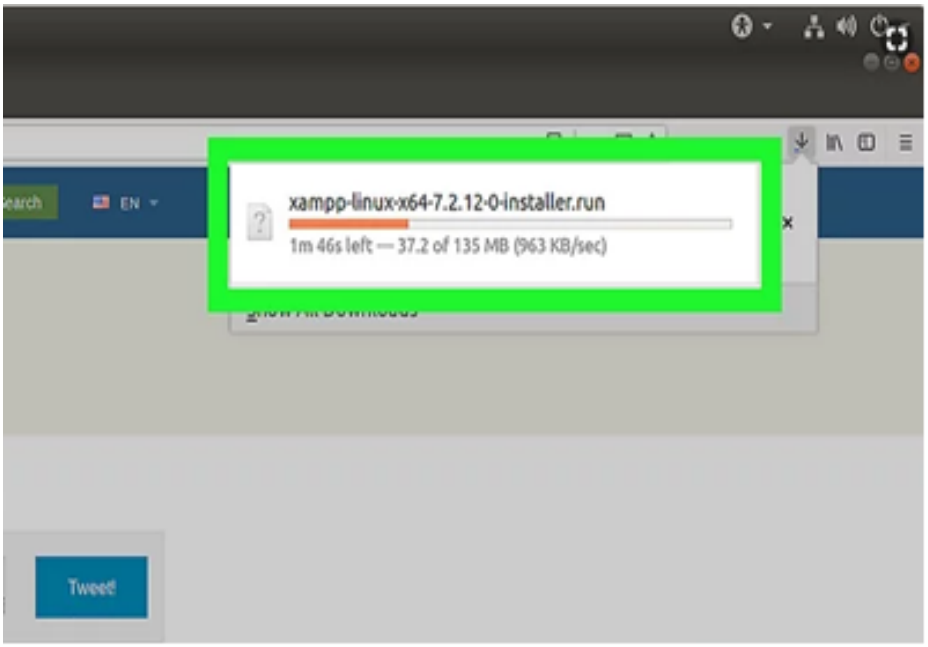
# INSTALL AND CONFIGURE XAMP

## AIM:

To install and configure Xamp in Linux

## Procedure

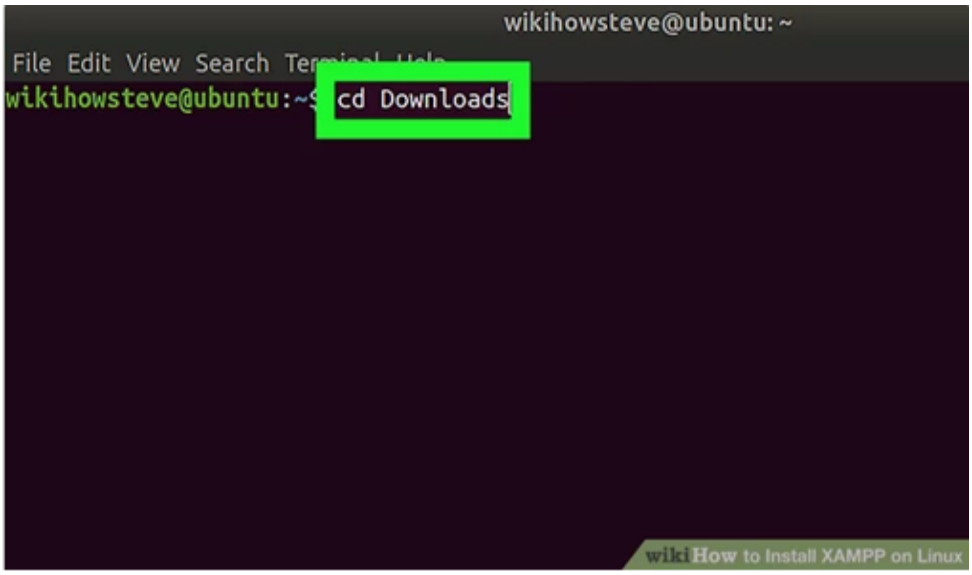
1. Open the XAMPP download page. Go to https://www.apachefriends.org/index.html in your computer's web browser. This is the official download site for XAMPP.



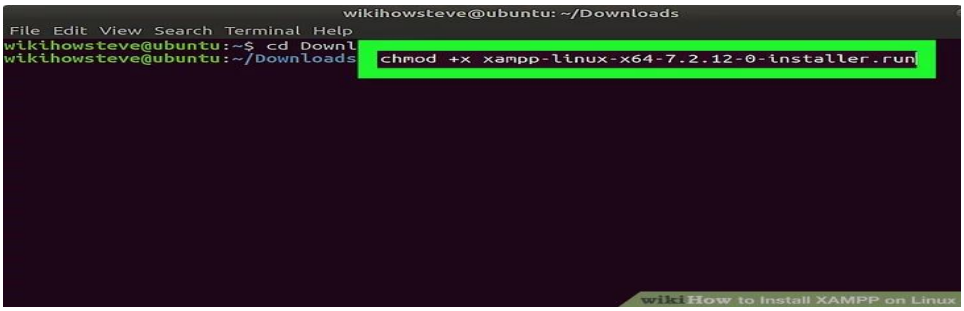
2. **Click XAMPP for Linux.** It's in the middle of the page. This will prompt the XAMPP setup file to begin downloading onto your computer. You may have to click **Save File** or select the "Downloads" folder as your


save location before proceeding.

3.Allow the download to complete. Once XAMPP's installation file finishes downloading onto your computer, you can proceed.

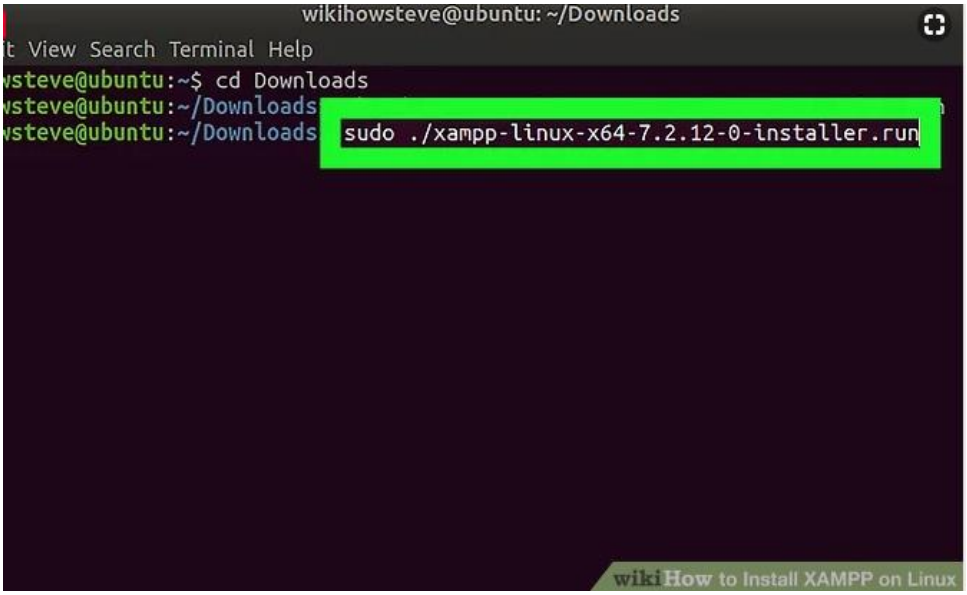



4.Open Terminal. Click the Terminal app icon, which resembles a black box with a white ">_" inside of it.You can also just press Alt+Ctrl+T to open a new Terminal window.

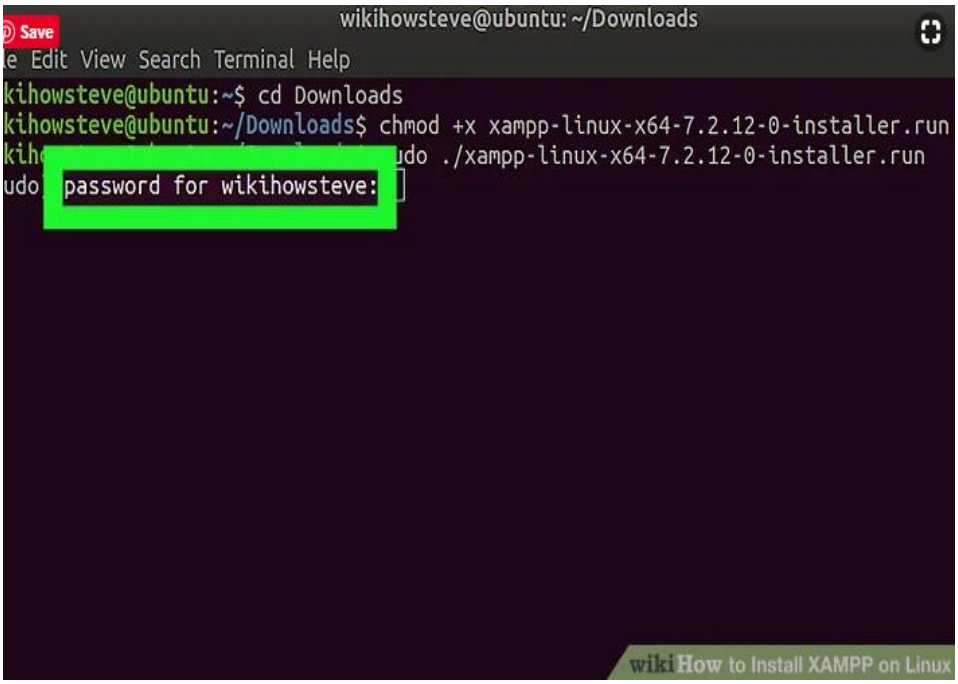



5. Change over to the "Downloads" directory. Type in cd Downloads and press ↲ Enter.Make sure you capitalize "Downloads". If your default downloads location is in a different folder, you'll have to change the directory to that folder.

6. Make the downloaded file executable. Type in chmod +x xampp-linux-x64-7.2.9-0-installer.run and press ↲ Enter.If you download a different version of XAMPP (e.g., version 5.9.3), you'll replace "7.2.9" with your XAMPP version's number.




7. **Enter the installation command.** Type in `sudo ./xampp-linux-x64-7.2.9-0-installer.run` and press ↲ Enter.




8. Enter your password when prompted. Type in the password you use to log into your computer, then press ↵ Enter. The installation window will pop up.You won't see the characters appear in Terminal when you type.


9. Follow the installation prompts. Once the installation window appears, do the following:

- Click Next three times.
- Uncheck the "Learn more about Bitnami for XAMPP" box.

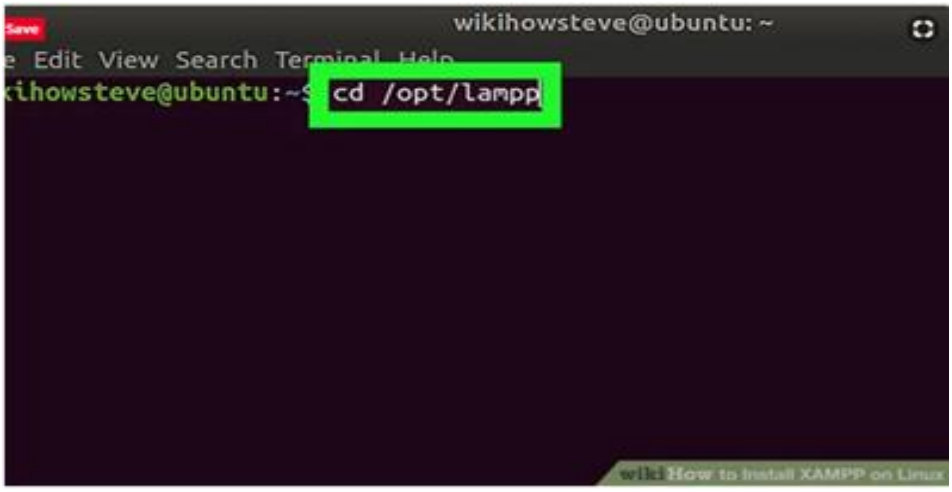
- Click Next, then click Next again to begin installing XAMPP.


10. Uncheck the "Launch XAMPP" box. It's in the middle of the final installation window. Since XAMPP needs a few extra steps to actually run on Linux, you'll need to finish the installation without automatically running XAMPP.

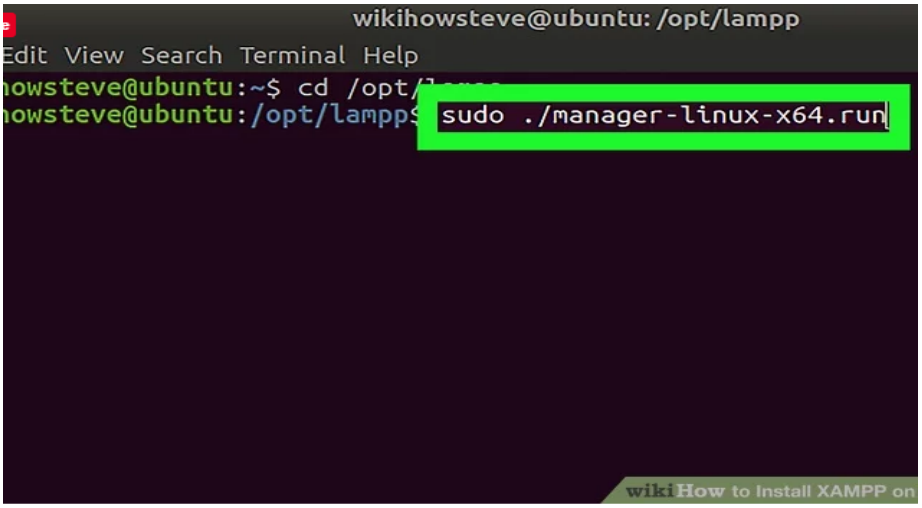

11. Click Finish. This option is at the bottom of the window. Doing so will close the installation window. At this point, you're ready to run XAMPP.

## Running XAMPP

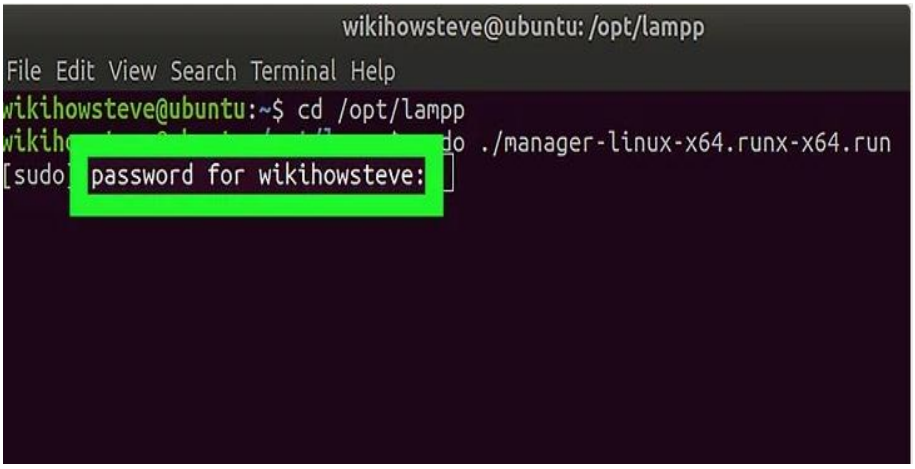
1.Re-open Terminal if necessary. If you closed the Terminal window that you used to install XAMPP, re-open Terminal.XAMPP doesn't have any desktop files, so you'll need to launch it from within its installation directory via Terminal each time you want to run it.



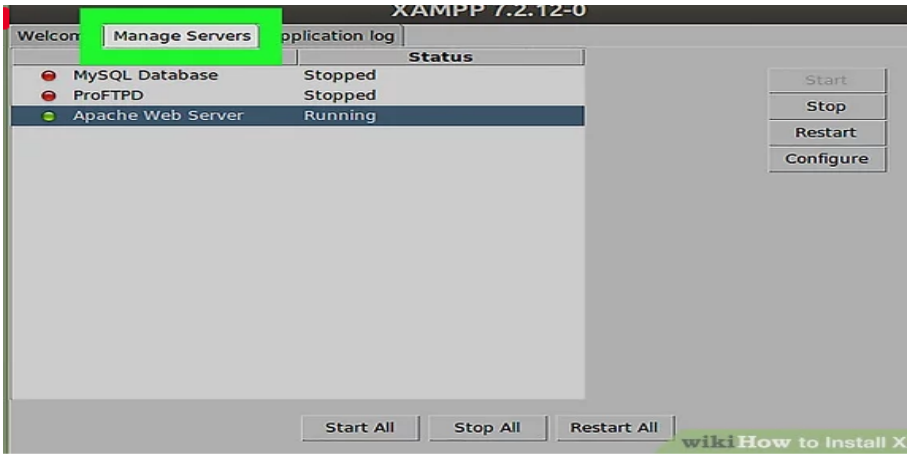

2. Switch to the XAMPP installation directory. Type in `cd /opt/lampp` and press `↵ Enter`



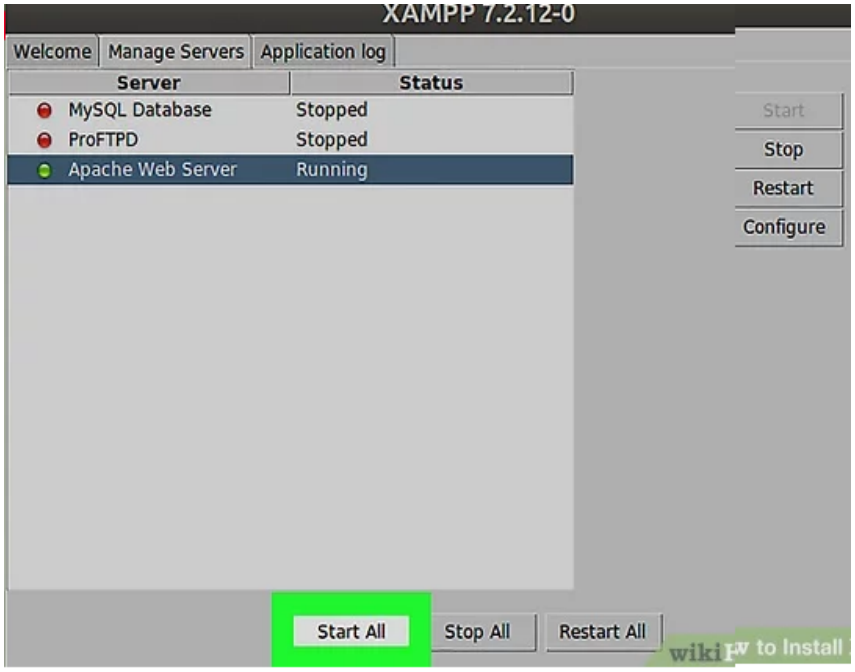

3, Enter the "Open" command. Type in `sudo ./manager-linux-x64.run` and press `↵ Enter`.



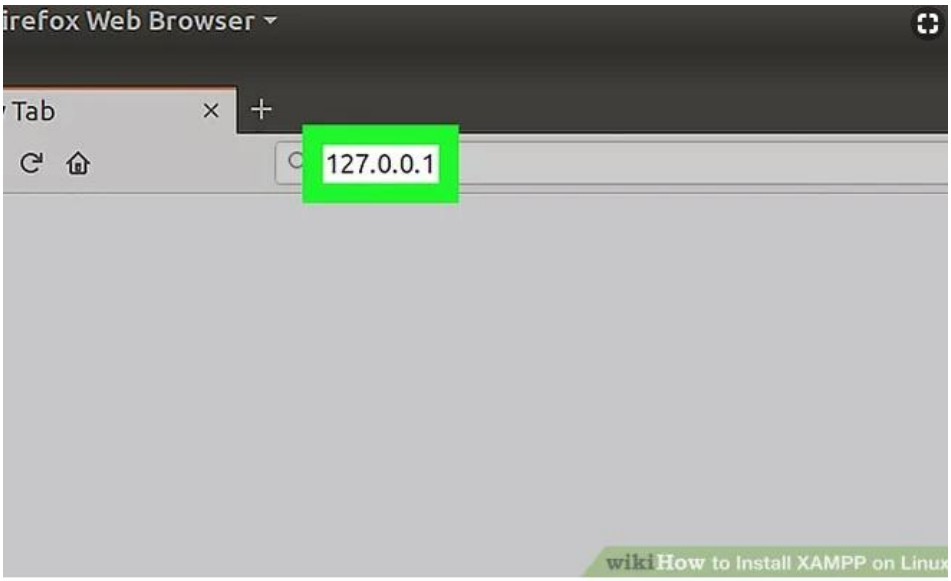
4. Enter your password when prompted. Type in the password you usually use to log into your computer, then press ↵ Enter.




5. Click the Manage Servers tab. This option is at the top of the window.




6.Click Start All. It's at the bottom of the window. Doing so prompts any active components of XAMPP to begin running.




7. Open your computer's localhost page. Go to 127.0.0.1 in your computer's web browser. You should see the XAMPP dashboard here; at this point, you're able to begin using XAMPP as you please.

## UNIX COMMANDS

### AIM:

To study basic Unix commands.

### File and Directory Related commands

1. pwd

This command prints the current working directory

2. ls

This command displays the list of files in the current working directory.

$ls –l Lists the files in the long format

$ls –t Lists in the order of last modification time

$ls –d Lists directory instead of contents

$ls -u Lists in order of last access time

3. cd

This command is used to change from the working directory to any other directory specified.

$cd directoryname

4. cd ..

This command is used to come out of the current working directory.

$cd ..

5. mkdir

This command helps us to make a directory.

$mkdir directoryname

6. rmdir

This command is used to remove a directory specified in the command line. It requires the specified directory to be empty before removing it.

$rmdir directoryname

7. cat

This command helps us to list the contents of a file we specify.

$cat [option][file]

cat > filename – This is used to create a new file.

cat >>filename – This is used to append the contents of the file

8. cp

This command helps us to create duplicate copies of ordinary files.

$cp source destination

9. mv

This command is used to move files.

$mv source destination

10. ln

This command is to establish an additional filename for the same ordinary file.

$ln firstname secondname

11. rm

This command is used to delete one or more files from the directory.

$rm [option] filename

$rm –i Asks the user if he wants to delete the file mentioned.

$rm –r Recursively delete the entire contents of the directory as well as the directory itself.

## Process and status information commands

1) who

This command gives the details of who all have logged in to the UNIX system currently.

$ who

2) who am i

This command tells us as to when we had logged in and the system's name for the connection being used.

$who am i

3) date

This command displays the current date in different formats.

| | | | |
|---|---|---|---|
| +%D | mm/dd/yy | +%w | Day of the week |
| +%H | Hr-00 to 23 | +%a | Abbr.Weekday |
| +%M | Min-00 to 59 | +%h | Abbr.Month |
| +%S | Sec-00 to 59 | +%r | Time in AM/PM |
| +%T | HH:MM:SS | +%y | Last two digits of the year |

4) echo

This command will display the text typed from the keyboard.

$echo

Eg : $echo Have a nice day

O/p : Have a nice day

## Text related commands

1. head

This command displays the initial part of the file. By default it displays first ten lines of the file.

$head [-count] [filename]

2. tail

This command displays the later part of the file. By default it displays last ten lines of the file.

$tail [-count] [filename]

3. wc

This command is used to count the number of lines, words or characters in a file.

$wc [-lwc] filename

4. find

The find command is used to locate files in a directory and in a subdirectory.

The –name option

This lists out the specific files in all directories beginning from the named directory. Wild cards can be used.

The –type option

This option is used to identify whether the name of files specified are ordinary files or directory files. If the name is a directory then use "-type d " and if it is a file then use "-type f".

The –mtime option

This option will allow us to find that file which has been modified before or after a specified time. The various options available are –mtime n(on a particular day),-mtime +n(before a particular day),-mtime –n(after a particular day)

The –exec option

This option is used to execute some commands on the files that are found by the find command.

## File Permission commands

1. chmod

Changes the file/directory permission mode: $ chmod 777 file1

Gives full permission to owner, group and others

$ chmod o-w file1

Removes write permission for others.

## Useful Commands:

1. Exit - Ends your work on the UNIX system.

# SHELL PROGRAMS



## Sum of n numbers

## AIM:

**To write a shell script to find the sum of n numbers.**

## ALGORITHM

Step 1: Start the program

Step 2: Enter the value of n

Step 3: Declare the variables of i and sum and both are initialized to zero.

Step 4: Perform the addition of n numbers using while loop

Step 5: Print the sum.

Step 6 : Stop the program

## Program

```
echo "Sum of n natural numbers \n”
echo "Enter the value of n "
read n
i = 0
sum = 0
while[$i –lt $n]
do
sum=’expr $sum + expr $i’
i = ‘expr $i + 1’
done
echo "Sum of n natural numbers are : $sum”
```

## OUTPUT

```
[csestaff@localhost csestaff]$ sh sum.sh
Sum of n natural numbers
Enter the value of n : 3
Sum of n natural numbers are : 6
```

## Result:

Thus, the shell script to find the sum of n natural numbers is entered and its output was verified.

---

## SWAPPING

---

## AIM:

To write a shell program for swapping of two numbers.

## ALGORITHM

1. Start the program.
2. Read the variables a, b.
3. Interchange the values of a and b using another temporary variable c as follows:

   c=a

   a=b

   b=c

4. Print the a and b.
5. Stop the program.

## Program

```
echo "swapping using temporary variable"
echo "enter a"
read a
echo "enter b"
read b
c=$a
a=$b
b=$c
echo "after swapping"
echo "$a"
echo "$b"
```

## OUTPUT

```
Enter a
10
Enter b
```

20

After swapping

20

10

## Result:

Thus, the shell script to swap two numbers is entered and its output was verified.

---

## LEAP YEAR

---

**AIM:**

To write a shell program to check whether the given year is leap year or not.

**ALGORITHM**

1. Start the program.
2. Read the year.
3. Check whether year%4,year%100,year%400 is zero.
4. If zero then print year is leap year.
5. Else print year is not leap year.
6. Stop the program.

**Program**

```
echo "Finding Leap Year"
echo "enter any year"
read y
a=`expr $y % 4`
b=`expr $y % 100`
c=`expr $y % 400`
if [ $a -eq 0 -a $b -ne 0 -o $c -eq 0 ]
then
echo "$y is a leap year "
else
echo "$y is not a leap year "
fi
```

**OUTPUT**

Enter a year:

2000

Year is leap year

## Result:

Thus, the shell script to check whether the given year is leap year or not is entered and its output was verified.

---

## FACTORIAL OF A NUMBER

---

## AIM:

To write a shell program for finding the factorial of a number.

## ALGORITHM

1. Start the program.
2. Read the number as n.
3. For every iteration until n<1 compute f=f*n.
4. Print the factorial of the given number as f.
5. Stop the program.

## Program

```
echo  “enter a positive number”
read  n
f=1
until  [  $n  -lt  1  ]
do
f=`expr  $f  \*  $n`
n=`expr  $n  -  1`
done
echo  “factorial is $f”
```

## OUTPUT

Enter positive number

4

Factorial is 24

## Result:

Thus, the shell script to find the factorial of a given number is entered and its output was verified

## SUM OF DIGITS OF A GIVEN NUMBER

## AIM:

To write a shell program for finding the sum of digits of a given number.

## ALGORITHM

1. Start the program.
2. Read the number as n.
3. Initialise sum=0
4. For every iteration n> 0 compute
   rem=n%10
   n=n/10
   sum=sum + rem
5. Print the sum of digits of the given number as sum.
6. Stop the program.

**Program**

```
echo "enter the number"
read n
sum=0
while [ $n -gt 0 ]
do
rem=`expr $n % 10`
n=`expr $n / 10`
sum=`expr $sum + $rem`
done
echo "sum of digits is:$sum"
```

**OUTPUT**

```
Enter a number:1234
Sum of digits is:10
```

## Result:

Thus, the shell script to find the sum the digits of a given number is entered and its output was verified.

---

## GREATEST AMONG THREE NUMBERS

## AIM:

To write a shell program for finding the greatest among three numbers.

## ALGORITHM

1. Start the program.
2. Read the three numbers a, b, c.
3. Check whether a is greater than b and c.
4. If yes then print a is big.
5. Else check whether b is greater than c.
6. If yes then print b is big.
7. Else print c is big.
8. Stop the program.

## Program

```
echo "enter a b c"
read a
read b
read c
if [ $a -gt $b ] && [ $a -gt $c ]
then
echo "$a big"
elif [ $b -gt $c ]
then
echo "$b big"
else
echo "$c big"
```

fi

## OUTPUT



Enter a b c

10

20

30

C big

## Result:

Thus, the shell script to find the sum the digits of a given number is entered and its output was verified.

# CREATING SIMPLE TABLES WITH CONSTRAINTS

## AIM:

To create tables using Sql Commads

## Procedure

After logging into the MySQL server, you are ready to do some work on your database creation.

You have a database space allocated to your use on the MySQL server.

To get to your database space, type:

use MIS3500_YourUserID;

(replace 'YourUserID' with your user ID as before)

The system response should be: Database changed .
To check if you are in the right place, type: \s .
To see if there is any table already in the database, type:

show tables;
The system response should be "Empty set". Remember this command for the latter use, once you will have some tables created and want to make sure they exist.

Now you can create your first table using the SQL statements below. You will be using a CREATE query. Note that the system will assume a continuous input as long as it does not encounter the semi-colon character (;). You can copy and paste the statements, although entering them manually for a bit will help you learn about the query structure. Note that the format of statement is given for the clarity purposes and not because the system requires it as such.

```
create table Customer
(
CustomerID int NOT NULL AUTO_INCREMENT PRIMARY KEY,
CustFirstName  varchar(25),
CustMidName varchar(10),
CustLastName varchar(25)
)
;
================
```

NOTE: Commands can be written either in the lower or upper case. Although the UNIX standard is lower case and MySQL runs on the UNIX operating system, the MySQL database engine is apparently not that restrictive. But, names of database objects are case sensitive! To do the work in an orderly fashion, you may decide to always enter system commands in the lower case, while the database objects must be addressed with the original case(s).

==========

To see how the table looks like type:
desc Customer;



**Output:**

```
mysql> desc Customer;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| CustomerID   | int(11)     | NO   | PRI | NULL    | auto_increment |
| CustFirstName| varchar(25) | YES  |     | NULL    |                |
| CustMidName  | varchar(10) | YES  |     | NULL    |                |
| CustLastName | varchar(25) | YES  |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

But in accord with the Note above, the command addressing the table name with a lower ‘c’ (desc customer) would cause an error message.

Move to creating the next table by entering the statements below.

```
create table SalesOrder
(
  OrderID int NOT NULL AUTO_INCREMENT PRIMARY KEY,
  OrderDate date,
  CustomerID int,
  FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
);
```

The next table is Product:

```
create table Product
(
ProductID int NOT NULL AUTO_INCREMENT PRIMARY KEY,
ProductName varchar(25),
UnitPrice decimal,
NumberInStock int
);
```

Then create the table OrderLine:

```
create table OrderLine
(OrderLineID int NOT NULL AUTO_INCREMENT PRIMARY KEY,
OrderID int,
ProductID int,
QuantityOrdered varchar(15),
FOREIGN KEY (OrderID) REFERENCES SalesOrder(OrderID),
FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
);
```

To see all the tables type:

show tables;

## INSERT UPDATE AND DELETE ROWS IN TABLE

## AIM:

To insert, Update and delete rows in a table

## Procedure

Now you are ready to insert some records into these tables. Insert records into the tables without foreign key first, then insert records into tables with foreign keys.

INSERT INTO Customer (CustFirstName, CustLastName)

VALUES ('John', 'Doe');

Note that the first line names columns, while the second inputs the data. The system checks (a) that the number of items is the same across these lines, and (b) the data types.

Check: select * from Customer;

Note the NULL value for the missing middle name.

Insert another record into table Customer:

INSERT INTO Customer (CustFirstName, CustMidName, CustLastName)

VALUES ('Bob', 'R.', 'Travis');

Check: select * from Customer;

Work then with other tables as follows.

INSERT INTO Product (ProductID, ProductName, UnitPrice, NumberInStock)

VALUES (1, 'Strawberry', 4.99, 16);

Check:

select * from Product;

## OUTPUT:

mysql> select * from Product;

```
+-----------+-------------+-----------+---------------+
| ProductID | ProductName | UnitPrice | NumberInStock |
+-----------+-------------+-----------+---------------+
|         1 | Strawberry  |         5 |            16 |
```

+----------+------------+----------+--------------+

1 row in set (0.00 sec)

You can see that one record has been created, however, the price is 5 instead of 4.99. It is because the data type was set with no decimal digits. You need to change it so that 2 decimal places are allocated. Run this alter table procedure:

## alter table Product

modify UnitPrice decimal (5, 2);

Check that result is as shown below.

Check the content of table Product. The incorrect price is still in it (5). So, you need to update the price:

## UPDATE Product

SET UnitPrice =4.99 WHERE ProductID =1;

Check that the right price is now inserted.

Keep entering data into SalesOrder:

mysql> INSERT INTO SalesOrder (OrderDate, CustomerID)

-> VALUES (DATE '2006-10-11', 1);

Query OK, 1 row affected (0.06 sec)

Check:

mysql> select * from SalesOrder;

## Output:

+---------+-----------+-----------+

| OrderID | OrderDate | CustomerID |

+---------+-----------+-----------+

| 1 | 2006-10-11 | 1 |

+---------+-----------+-----------+

1 row in set (0.00 sec)

Check:

mysql> select * from SalesOrder;

## Output:

```
+---------+------------+------------+

| OrderID | OrderDate  | CustomerID |

+---------+------------+------------+

|       1 | 2006-10-11 |          1 |

+---------+------------+------------+
```

1 row in set (0.00 sec)

Finally, enter a row into table OrderLine:

INSERT INTO OrderLine (OrderLineID, OrderID, ProductID, QuantityOrdered)

VALUES (1,1,1,'3kg');

Check:

Useful Row and Table Commands

These commands are compliant with the SQL standard that works across DBMSes.

## Table Copying

To copy a table, there are several options, with regard to what is copied (just the structure (metadata) o the structure and indexes – all without data, or all these with the data). Below is that last total copy options shown via screen copies.

mysql> create table Customer2 as select * from Customer;

Query OK, 2 rows affected (0.04 sec)

Records: 2  Duplicates: 0  Warnings: 0

Check:

mysql> select * from Customer2;

```
+------------+--------------+-------------+--------------+

| CustomerID | CustFirstName | CustMidName | CustLastName |

+------------+--------------+-------------+--------------+

|          1 | John         | NULL        | Doe          |

|          2 | Bob          | R.          | Travis       |

+------------+--------------+-------------+--------------+
```

2 rows in set (0.00 sec)



## Table Deletion

The syntax is: drop table [table name];

For example: drop table Customer2;

## Row Deletion

A delete query works as in any relational DBMS supporting SQL. So, the syntax is:

delete from [table name] where [condition] .

Let's assume a table Customer5 is created via the copy statement above, and then we want to delete the row for the customer Bob. The screenshot below demonstrates all these steps.

Note: Error is reported because of disrespecting the upper case in the table name.

## Clear Screen

Type \c .

For example, you use this command when you want to exit a sequence of command lines before typing the semi-colon symbol because you made some error that cannot be corrected.

## Exit MySQL

Type exit or quit or \q .

---

# SEARCHING OF DATA

---

## AIM:

Searching More than One Table

## Procedure

The SELECT query of multiple tables works in a MySQL system as in any other database system supporting the SQL standard. The INNER JOIN and OUTER JOINS work as with, say, MS Access.Try the simplest query first on tables Customer and SalersOrder, as shown below.

```
mysql> select * from Customer
    -> inner join SalesOrder on
    -> Customer.CustomerID=SalesOrder.CustomerID
    -> ;
```

The LEFT JOIN works too:

The old SQL syntax should also works:

How about joining more than 2 tables (something students really like!) Let us see who likes strawberrie. A join of 4 tables is needed as follows.

```
SELECT Customer.CustFirstName, CustLastName, Product.ProductName
As 'Bought product'

FROM Product INNER JOIN

(

  ( OrderLine INNER JOIN

     (Customer INNER JOIN SalesOrder ON Customer.CustomerID
        =SalesOrder.CustomerID)

   ON OrderLine.OrderID=SalesOrder.OrderID

  )

 )

   ON Product.ProductID=OrderLine.ProductID

WHERE Product.ProductName LIKE 'Strawb%' ;
```

Screen shot of the statement and output:

## Analysis:

1. The schema is: Customer---SalesOrder---OrderLine---Product. The INNER JOIN works the same way as in MS Access. Starting from the innermost part of the query, the first join references tables Customer to SalesOrder, the result of the next join references OrderLine, and the result of this join references Product.

2. The LIKE operator for strings is supported as in MS Access. Quotes must be used.

3. The wild card symbol replacing any character and any number of characters is a '%'.

The old SQL syntax also works:

```
SELECT Customer.CustFirstName, CustLastName, Product.ProductName
As 'Bought product'

FROM Product, OrderLine, SalesOrder, Customer

WHERE

  Product.ProductID=OrderLine.ProductID AND
```

OrderLine.OrderID=SalesOrder.OrderID AND

SalesOrder.CustomerID=Customer.CustomerID

AND

Product.ProductName LIKE 'Strawb%' ;

## SORTING DATA

**AIM:**

To sort data in a table

### Procedure

We have seen the SQL SELECT command to fetch data from a MySQL table. When you select rows, the MySQL server is free to return them in any order, unless you instruct it otherwise by saying how to sort the result. But, you sort a result set by adding an ORDER BY clause that names the column or columns which you want to sort.

### Syntax

The following code block is a generic SQL syntax of the SELECT command along with the ORDER BY clause to sort the data from a MySQL table.

SELECT field1, field2,...fieldN table_name1, table_name2...

ORDER BY field1, [field2...] [ASC [DESC]]

You can sort the returned result on any field, if that field is being listed out.You can sort the result on more than one field.You can use the keyword ASC or DESC to get result in ascending or descending order. By default, it's the ascending order.You can use the WHERE...LIKE clause in the usual way to put a condition.

Using ORDER BY clause at the Command Prompt

This will use the SQL SELECT command with the ORDER BY clause to fetch data from the MySQL table – tutorials_tbl.

Example

Try out the following example, which returns the result in an ascending order.

root@host# mysql -u root -p password;

Enter password:*******

mysql> use TUTORIALS;

```
Database changed
mysql> SELECT * from tutorials_tbl ORDER BY tutorial_author ASC
+-------------+---------------+----------------+-----------------+
| tutorial_id | tutorial_title | tutorial_author | submission_date |
+-------------+---------------+----------------+-----------------+
|      2      | Learn MySQL   |     Abdul S    |    2007-05-24   |
|      1      |  Learn PHP    |    John Poul   |    2007-05-24   |
|      3      | JAVA Tutorial |     Sanjay     |    2007-05-06   |
+-------------+---------------+----------------+-----------------+
3 rows in set (0.42 sec)
mysql>
```

Verify all the author names that are listed out in the ascending order.

Using ORDER BY clause inside a PHP Script

You can use a similar syntax of the ORDER BY clause into the PHP function – mysql_query(). This function is used to execute the SQL command and later another PHP function mysql_fetch_array() can be used to fetch all the selected data.

Example

Try out the following example, which returns the result in a descending order of the tutorial authors.

```
<?php
  $dbhost = 'localhost:3036';
  $dbuser = 'root';
  $dbpass = 'rootpassword';
  $conn = mysql_connect($dbhost, $dbuser, $dbpass);
    if(! $conn ) {
    die('Could not connect: ' . mysql_error());
  }
  $sql = 'SELECT tutorial_id, tutorial_title,
    tutorial_author, submission_date
```

```
   FROM tutorials_tbl
   ORDER BY  tutorial_author DESC';

  mysql_select_db('TUTORIALS');
  $retval = mysql_query( $sql, $conn );

  if(! $retval ) {
    die('Could not get data: ' . mysql_error());
  }

  while($row = mysql_fetch_array($retval, MYSQL_ASSOC)) {
    echo "Tutorial ID :{$row['tutorial_id']}  <br> ".
      "Title: {$row['tutorial_title']} <br> ".
      "Author: {$row['tutorial_author']} <br> ".
      "Submission Date : {$row['submission_date']} <br> ".
      "--------------------------------<br>";
  }
  echo "Fetched data successfully\n";
  mysql_close($conn);
?>
```

## USAGE OF SUBQUERIES IN SQL

### AIM:

To implement subqueries in mysql commands

### Procedure

A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

There are a few rules that subqueries must follow

Subqueries must be enclosed within parentheses.

A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.

An ORDER BY command cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY command can be used to perform the same function as the ORDER BY in a subquery.

Subqueries that return more than one row can only be used with multiple value operators such as the IN operator.

The SELECT list cannot include any references to values that evaluate to a BLOB, ARRAY, CLOB, or NCLOB.

A subquery cannot be immediately enclosed in a set function.

The BETWEEN operator cannot be used with a subquery. However, the BETWEEN operator can be used within the subquery.Subqueries with the SELECT Statement

Subqueries are most frequently used with the SELECT statement. **The basic syntax is as follows**

```
SELECT column_name [, column_name ]

FROM   table1 [, table2 ]

WHERE  column_name OPERATOR

  (SELECT column_name [, column_name ]

  FROM table1 [, table2 ]

  [WHERE])
```

## Example

Consider the CUSTOMERS table having the following records −

```
+----+----------+-----+-----------+----------+

| ID | NAME     | AGE | ADDRESS   | SALARY   |

+----+----------+-----+-----------+----------+

|  1 | Ramesh   |  35 | Ahmedabad |  2000.00 |
```

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

+----+----------+-----+-----------+----------+

Now, let us check the following subquery with a SELECT statement.

```
SQL> SELECT *
 FROM CUSTOMERS
 WHERE ID IN (SELECT ID
 FROM CUSTOMERS
  WHERE SALARY > 4500) ;
```

This would produce the following result.

+----+----------+-----+---------+----------+

| ID | NAME | AGE | ADDRESS | SALARY |

+----+----------+-----+---------+----------+

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

+----+----------+-----+---------+----------+

## Subqueries with the INSERT Statement

Subqueries also can be used with INSERT statements. The INSERT statement uses the data returned from the subquery to insert into another table. The selected data in the subquery can be modified with any of the character, date or number functions.

## The basic syntax is as follows.

```
INSERT INTO table_name [ (column1 [, column2 ]) ]
   SELECT [ *|column1 [, column2 ]
```

FROM table1 [, table2 ]

[ WHERE VALUE OPERATOR ]

## Example

Consider a table CUSTOMERS_BKP with similar structure as CUSTOMERS table. Now to copy the complete CUSTOMERS table into the CUSTOMERS_BKP table, you can use the following syntax.

SQL> INSERT INTO CUSTOMERS_BKP

SELECT * FROM CUSTOMERS

WHERE ID IN (SELECT ID

FROM CUSTOMERS) ;

## Subqueries with the UPDATE Statement

The subquery can be used in conjunction with the UPDATE statement. Either single or multiple columns in a table can be updated when using a subquery with the UPDATE statement.

## The basic syntax is as follows.

UPDATE table

SET column_name = new_value

[ WHERE OPERATOR [ VALUE ]

(SELECT COLUMN_NAME

FROM TABLE_NAME)

[ WHERE) ]

## Example

Assuming, we have CUSTOMERS_BKP table available which is backup of CUSTOMERS table. The following example updates SALARY by 0.25 times in the CUSTOMERS table for all the customers whose AGE is greater than or equal to 27.

SQL> UPDATE CUSTOMERS

SET SALARY = SALARY * 0.25

WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP

WHERE AGE >= 27 );

This would impact two rows and finally CUSTOMERS table would have the following records.

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  35 | Ahmedabad |   125.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  2125.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

## Subqueries with the DELETE Statement

The subquery can be used in conjunction with the DELETE statement like with any other statements mentioned above.

## The basic syntax is as follows.

```
DELETE FROM TABLE_NAME
[ WHERE OPERATOR [ VALUE ]
  (SELECT COLUMN_NAME
  FROM TABLE_NAME)
  [ WHERE) ]
```

## Example

Assuming, we have a CUSTOMERS_BKP table available which is a backup of the CUSTOMERS table. The following example deletes the records from the CUSTOMERS table for all the customers whose AGE is greater than or equal to 27.

```
SQL> DELETE FROM CUSTOMERS
  WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP
    WHERE AGE >= 27 );
```

This would impact two rows and finally the CUSTOMERS table would have the following records.

```
+----+----------+-----+---------+----------+
| ID | NAME     | AGE | ADDRESS | SALARY   |
+----+----------+-----+---------+----------+
| 2 | Khilan   | 25 | Delhi   | 1500.00 |
| 3 | kaushik  | 23 | Kota    | 2000.00 |
| 4 | Chaitali | 25 | Mumbai  | 6500.00 |
| 6 | Komal    | 22 | MP      | 4500.00 |
| 7 | Muffy    | 24 | Indore  | 10000.00 |
+----+----------+-----+---------+----------+
```

AGGREGATE FUNCTIONS IN SQL

## AIM:

To implement and learn aggregate functions and its usage in queries

## Procedure

Aggregate Functions are the Functions where the value of input is grouped together and fetches the output as a single value. Following is single line Explanation of the Aggregate functions:

SQL Aggregate functions

AVG() – Returns the average value

COUNT() – Returns the number of rows

MAX() – Returns the largest value

MIN() – Returns the smallest value

SUM() – Returns the sum

UCASE() – Converts a field to upper case

LCASE() – Converts a field to lower case

SUBSTR() – Extract characters from a text field

LEN()/LENGTH() – Returns the length of a text field

ROUND() – Rounds a numeric field to the number of decimals specified

**1. The AVG () Function:**The AVG () function returns the average value of a numeric column.

## Syntax:

SELECT AVG (Column_Name)

FROM <Table_Name>;

Scenario: How to Find Average salary from Employee Table departmentwise?

Example:

SELECT AVG (Salary)

FROM Employee;

SELECT DeptNo, AVG (Sal) FROM EMP

GROUP BY DeptNo;

SELECT DeptNo, AVG (Sal) AS AvgSal FROM EMP

GROUP BY DeptNo

ORDER BY DeptNo;

SELECT EMPNo, Ename, Job, Sal, DeptNo FROM EMP

WHERE Sal > (SELECT AVG (Sal) FROM EMP);

**2) The COUNT () Function**:The COUNT () function returns the number of rows that matches a specified criteria.We Can count the Number of rows using following 3 types:

1) COUNT (Column_Name)

2) COUNT (*)

3) COUNT (DISTINCT Column_Name)

1**) Syntax:**

SELECT COUNT (Column_Name)

FROM <Table_Name>;

Scenario : How to find the count of names of Employees who are doing job as CLERK?

## Example:

SELECT COUNT (Ename) FROM Employee;

SELECT Job, COUNT (Job) FROM Employee

WHERE Job='CLERK'

GROUP BY Job;

"The COUNT (Column_Name) function returns the number of values (NULL values will not be counted) of the specified column."

**2) Syntax:**

SELECT COUNT (*)

FROM <Table_Name>;

Example:

SELECT COUNT (*) FROM Employee;

"The COUNT (*) function returns the total number of records in a table, counts NULL values also"

**3) Syntax:**

SELECT COUNT (DISTINCT Column_Name)

FROM <Table_Name>;

**Example:**

SELECT COUNT (DISTINCT Job) FROM EMP;

SELECT COUNT (DISTINCT (Ename)) FROM EMP;

SELECT Ename, COUNT (DISTINCT (Ename)) FROM EMP GROUP BY Ename;

SELECT Job, COUNT (DISTINCT (Job)) FROM EMP GROUP BY Job;

"The COUNT (DISTINCT column_name) function returns the number of distinct values of the specified column."

**3) The MAX () Function:**The MAX () function returns the largest value of the selected column.

**Syntax:**

SELECT MAX (Column_Name)

FROM <Table_Name>;

SELECT MAX (Sal) FROM EMP;

SELECT DeptNo, Max (Sal) FROM EMP

GROUP BY DeptNo

ORDER BY DeptNo;

**4) The MIN () Function:**The MIN () function returns the smallest value of the selected column.

## Syntax:

SELECT MIN (Column_Name) FROM <Table_Name>;

SELECT MIN (Sal) FROM EMP;

SELECT DeptNo, Min (Sal) FROM EMP

GROUP BY DeptNo

ORDER BY DeptNo;

**5) The SUM () Function:**The SUM () function returns the total sum of a numeric column.

## Syntax:

SELECT SUM (Column_Name) FROM <Table_Name>;

## Example:

SELECT SUM (Sal) FROM EMP;

SELECT DeptNo, Sum (Sal) FROM EMP

GROUP BY DeptNo

ORDER BY DeptNo;

Types of SQL Functions

**6) The UPPER () Function:**The UPPER () function converts the value of a field to Upper-Case.

## Syntax:

SELECT UPPER (Column_Name) FROM <Table_Name>;

## Example :

SELECT UPPER (Ename) FROM EMP;

**7) The LOWER () Function:**The LOWER () function converts the value of a field to Lower-Case.

## Syntax:

SELECT LOWER (Column_Name) FROM <Table_Name>;

## Example:

SELECT LOWER (Ename) FROM EMP;

**8) The INITCAP () Function:** The INITCAP () function converts the value of a field to Initial-Case.

## Syntax:

SELECT INITCAP (Column_Name) FROM <Table_Name>;

SELECT INITCAP (Ename) FROM EMP;

**9) The SUBSTR () Function:**The SUBSTR () function is used to extract characters from a text field.

## Syntax:

SELECT SUBSTR (Column_Name, Start Position, Length)

FROM <Table_Name>;

Here is the description of parameters of Substr function.Substring function is widely used functions in SQL and PLSQL development.

Parameter Description

column_name Required. The field to extract characters from

Start Required. Specifies the starting position (starts at 1)

Length Optional. The number of characters to return. If omitted, the MID() function returns the rest of the text

## Example:

SELECT Substr (Ename, 1, 3) FROM EMP;

## 10) The LENGTH () Function:

The LENGTH () function returns the length of the value in a text field.

**Syntax:**

SELECT LENGTH (Column_Name)

FROM <Table_Name>;

SELECT LENGTH (Ename) FROM EMP;

## 11) The ROUND () Function:

The ROUND () function is used to round a numeric field to the number of decimals specified.

## Syntax:

SELECT ROUND (Column_Name, Decimals)

FROM EMP;

Following are Parameter and its description for Round Function:

Parameter Description

column_name Required. The field to round.

Decimals Required. Specifies the number of decimals to be returned.

Example:

SELECT ROUND (Comm, 2) FROM EMP;

## WORKING OF SET OPERATIONS IN MYSQL

### AIM:

To learn and implement the set operations in mysql

### Procedure

Set operators allow you to combine the results of multiple separate queries into a single result set.The following two queries will be used for most of the examples in this article. The first returns the departments 10, 20 and 30. The second returns the departments 20, 30 and 40. As you can see, departments 20 and 30 are common to both result sets.

-- Department 10, 20 and 30.

SELECT department_id, department_name

FROM departments

WHERE department_id <= 30;

DEPARTMENT_ID DEPARTMENT_NAM

---------- --------------

10 ACCOUNTING

20 RESEARCH

30 SALES

3 rows selected.

SQL>

--Department 20, 30 and 40.

SELECT department_id, department_name

FROM departments

```
WHERE  department_id >= 20;

DEPARTMENT_ID DEPARTMENT_NAM
------------ --------------
          20 RESEARCH
          30 SALES
          40 OPERATIONS

3 rows selected.

SQL>
```

You will see, these are not real-world examples, but they serve to demonstrate how each of the set operators work

## UNION

The UNION set operator returns all distinct rows selected by either query. That means any duplicate rows will be removed.In the example below, notice there is only a single row each for departments 20 and 30, rather than two each.

```
SELECT department_id, department_name
FROM   departments
WHERE  department_id <= 30
UNION
SELECT department_id, department_name
FROM   departments
WHERE  department_id >= 20
ORDER BY 1;

DEPARTMENT_ID DEPARTMENT_NAM
------------ --------------
          10 ACCOUNTING
          20 RESEARCH
          30 SALES
          40 OPERATIONS

4 rows selected.
```

SQL>

The removal of duplicates requires extra processing, so you should consider using UNION ALL if possible

## UNION ALL

The UNION ALL set operator returns all rows selected by either query. That means any duplicates will remain in the final result set.In the example below, notice there are two rows each for departments 20 and 30.

```
SELECT department_id, department_name

FROM   departments

WHERE  department_id <= 30

UNION ALL

SELECT department_id, department_name

FROM   departments

WHERE  department_id >= 20

ORDER BY 1;

DEPARTMENT_ID DEPARTMENT_NAM

------------- --------------

           10 ACCOUNTING

           20 RESEARCH

           20 RESEARCH

           30 SALES

           30 SALES

           40 OPERATIONS

6 rows selected.

SQL>
```

## INTERSECT

The INTERSECT set operator returns all distinct rows selected by both queries. That means only those rows common to both queries will be present in the final result set. In the example below, notice there is one row each for departments 20 and 30, as both these appear in the result sets for their respective queries.

```
SELECT department_id, department_name
FROM   departments
WHERE  department_id <= 30
INTERSECT
SELECT department_id, department_name
FROM   departments
WHERE  department_id >= 20
ORDER BY 1;

DEPARTMENT_ID DEPARTMENT_NAM
------------- --------------
           20 RESEARCH
           30 SALES

2 rows selected.

SQL>
```

## MINUS

The MINUS set operator returns all distinct rows selected by the first query but not the second. This is functionally equivalent to the ANSI set operator EXCEPT DISTINCT. In the example below, the first query would return departments 10, 20, 30, but departments 20 and 30 are removed because they are returned by the second query. This leaves a single rows for department 10.

```
SELECT department_id, department_name
FROM   departments
WHERE  department_id <= 30
MINUS
SELECT department_id, department_name
FROM   departments
WHERE  department_id >= 20
ORDER BY 1;

DEPARTMENT_ID DEPARTMENT_NAM
------------- --------------
```

10 ACCOUNTING

1 row selected.

SQL>

## ORDER BY

The ORDER BY clause is applied to all rows returned in the final result set. Columns in the ORDER BY clause can be referenced by column names or column aliases present in the first query of the statement, as these carry through to the final result set. Typically, you will see people use the column position as it is less confusing when the data is sourced from different locations for each query block.

-- Column name.

SELECT employee_id, employee_name

FROM employees

WHERE department_id = 10

UNION ALL

SELECT department_id, department_name

FROM departments

WHERE department_id >= 20

ORDER BY employee_id;

EMPLOYEE_ID EMPLOYEE_NAME

---------- --------------

20 RESEARCH

30 SALES

40 OPERATIONS

7782 CLARK

7839 KING

7934 MILLER

6 rows selected.

SQL>

-- Column Alias

SELECT employee_id AS emp_id, employee_name

```
FROM   employees
WHERE  department_id = 10
UNION ALL
SELECT department_id, department_name
FROM   departments
WHERE  department_id >= 20
ORDER BY emp_id;
EMP_ID EMPLOYEE_NAME
---------- --------------
        20 RESEARCH
        30 SALES
        40 OPERATIONS
      7782 CLARK
      7839 KING
      7934 MILLER
6 rows selected.
SQL>
-- Column position
SELECT employee_id, employee_name
FROM   employees
WHERE  department_id = 10
UNION ALL
SELECT department_id, department_name
FROM   departments
WHERE  department_id >= 20
ORDER BY 1;
EMPLOYEE_ID EMPLOYEE_NAME
----------- --------------
        20 RESEARCH
        30 SALES
```

40 OPERATIONS

7782 CLARK

7839 KING

7934 MILLER

6 rows selected.

SQL>

## WORKING WITH STRINGS IN MYSQL

### AIM:

To implement string operations in mysql

### String Functions

**ASCII()-**Ascii code value will come as output for a character expression.

### Example

The following query will give the Ascii code value of a given character.

Select ASCII ('word')

**CHAR()-**Character will come as output for given Ascii code or integer.

### Example

The following query will give the character for a given integer.

Select CHAR(97)

**NCHAR()**-Unicode character will come as output for a given integer.

### Example

The following query will give the Unicode character for a given integer.

Select NCHAR(300)

**CHARINDEX()-**Starting position for given search expression will come as output in a given string expression.

### Example

The following query will give the starting position of 'G' character for given string expression 'KING'.

Select CHARINDEX('G', 'KING')

**LEFT()-**Left part of the given string till the specified number of characters will come as output for a given string.

## Example

The following query will give the 'WORL' string as mentioned 4 number of characters for given string 'WORLD'.

Select LEFT('WORLD', 4)

**RIGHT()-**Right part of the given string till the specified number of characters will come as output for a given string.

## Example

The following query will give the 'DIA' string as mentioned 3 number of characters for given string 'INDIA'.

Select RIGHT('INDIA', 3)

**SUBSTRING()-**Part of a string based on the start position value and length value will come as output for a given string.

Example

The following queries will give the 'WOR', 'DIA', 'ING' strings as we mentioned (1,3), (3,3) and (2,3) as start and length values respectively for given strings 'WORLD', 'INDIA' and 'KING'.

Select SUBSTRING ('WORLD', 1,3)

Select SUBSTRING ('INDIA', 3,3)

Select SUBSTRING ('KING', 2,3)

**LEN()-**Number of characters will come as output for a given string expression.

## Example

The following query will give the 5 for the 'HELLO' string expression.

Select LEN('HELLO')

**LOWER()-**Lowercase string will come as output for a given string data.

## Example

The following query will give the 'sqlserver' for the 'SQLServer' character data.

Select LOWER('SQLServer')

**UPPER()-**Uppercase string will come as output for a given string data.

**Example**

The following query will give the 'SQLSERVER' for the 'SqlServer' character data.

Select UPPER('SqlServer')

LTRIM()

String expression will come as output for a given string data after removing leading blanks.

## Example

The following query will give the 'WORLD' for the '  WORLD' character data.

Select LTRIM('  WORLD')

**RTRIM()-**String expression will come as output for a given string data after removing trailing blanks.

## Example

The following query will give the 'INDIA' for the 'INDIA  ' character data.

Select RTRIM('INDIA  ')

**REPLACE()-**String expression will come as output for a given string data after replacing all occurrences of specified character with specified character.

## Example

The following query will give the 'KNDKA' string for the 'INDIA' string data.

Select REPLACE('INDIA', 'I', 'K')

**REPLICATE()**-Repeat string expression will come as output for a given string data with specified number of times.

## Example

The following query will give the 'WORLDWORLD' string for the 'WORLD' string data.

Select REPLICATE('WORLD', 2)

**REVERSE()-**Reverse string expression will come as output for a given string data.

## Example

The following query will give the 'DLROW' string for the 'WORLD' string data.

Select REVERSE('WORLD')

**SOUNDEX()-**Returns four-character (SOUNDEX) code to evaluate the similarity of two given strings.

## Example

The following query will give the 'S530' for the 'Smith', 'Smyth' strings.

Select SOUNDEX('Smith'), SOUNDEX('Smyth')

**DIFFERENCE()-**Integer value will come as output of given two expressions.

## Example

The following query will give the 4 for the 'Smith', 'Smyth' expressions.

Select Difference('Smith','Smyth')

**SPACE()-**String will come as output with the specified number of spaces.

## Example

The following query will give the 'I LOVE INDIA'.

Select 'I'+space(1)+'LOVE'+space(1)+'INDIA'

**STUFF()-**String expression will come as output for a given string data after replacing from starting character till the specified length with specified character.

## Example

The following query will give the 'AIJKFGH' string for the 'ABCDEFGH' string data as per given starting character and length as 2 and 4 respectively and 'IJK' as specified target string.

Select STUFF('ABCDEFGH', 2,4,'IJK')

**STR()-**Character data will come as output for the given numeric data.

## Example

The following query will give the 187.37 for the given 187.369 based on specified length as 6 and decimal as 2.

Select STR(187.369,6,2)

**UNICODE()-**Integer value will come as output for the first character of given expression.

## Example

The following query will give the 82 for the 'RAMA' expression.

Select UNICODE('RAMA')

**QUOTENAME()-**Given string will come as output with the specified delimiter.

## Example

The following query will give the "RAMA" for the given 'RAMA' string as we specified double quote as delimiter.

Select QUOTENAME('RAMA','"')

**PATINDEX()-**Starting position of the first occurrence from the given expression as we specified 'I' position is required.

## Example

The following query will give the 1 for the 'INDIA'.

Select PATINDEX('I%','INDIA')

**FORMAT()-**Given expression will come as output with the specified format.

## Example

The following query will give the ' Monday, November 16, 2015' for the getdate function as per specified format with 'D' refers weekday name.

SELECT FORMAT ( getdate(), 'D')

**CONCAT()-**Single string will come as output after concatenating the given parameter values.

## Example

The following query will give the 'A,B,C' for the given parameters.

Select CONCAT('A',',','B',',','C')

---

# NUMERIC FUNCTIONS IN MYSQL

---

## AIM:

To implement numeric functions in Mysql

## Numeric Functions

**ABS(X)-**The ABS() function returns the absolute value of X. Consider the following example

SQL> SELECT ABS(2);

+----------+

| ABS(2) |

+----------+

| 2 |

+----------+

1 row in set (0.00 sec)

SQL> SELECT ABS(-2);

+----------+

| ABS(2) |

+----------+

| 2 |

+----------+

1 row in set (0.00 sec)

**ACOS(X)-**This function returns the arccosine of X. The value of X must range between -1 and 1 or NULL will be returned. Consider the following example −

SQL> SELECT ACOS(1);

+----------+

| ACOS(1) |

+----------+

| 0.000000 |

+----------+

1 row in set (0.00 sec)

**ASIN(X)-**The ASIN() function returns the arcsine of X. The value of X must be in the range of -1 to 1 or NULL is returned.

SQL> SELECT ASIN(1);

+----------+

| ASIN(1) |

+----------+

| 1.5707963267949 |

+----------+

1 row in set (0.00 sec)

**ATAN(X)-**This function returns the arctangent of X.

SQL> SELECT ATAN(1);

+----------------------------------------------------+

| ATAN(1) |

+----------------------------------------------------+

| 0.78539816339745 |

+----------------------------------------------------+

1 row in set (0.00 sec)

**ATAN2(Y,X)-**This function returns the arctangent of the two arguments: X and Y. It is similar to the arctangent of Y/X, except that the signs of both are used to find the quadrant of the result.

SQL> SELECT ATAN2(3,6);

+----------------------------------------------------+

| ATAN2(3,6) |

+----------------------------------------------------+

| 0.46364760900081 |

+----------------------------------------------------+

1 row in set (0.00 sec)

**BIT_AND(expression)-**The BIT_AND function returns the bitwise AND of all bits in expression. The basic premise is that if two corresponding bits are the same, then a bitwise AND operation will return 1, while if they are different, a bitwise AND operation will return 0. The function itself returns a 64-bit integer value. If there are no matches, then it will return 18446744073709551615. The following example performs the BIT_AND function on the PRICE column grouped by the MAKER of the car −

SQL> SELECT

MAKER, BIT_AND(PRICE) BITS

FROM CARS GROUP BY MAKER

+----------------------------------------------------+

|MAKER BITS |

```
+-----------------------------------------------------+
|CHRYSLER       512                                   |
|FORD           12488                                 |
|HONDA          2144                                  |
+-----------------------------------------------------+
```

1 row in set (0.00 sec)

**BIT_COUNT(numeric_value)-**The BIT_COUNT() function returns the number of bits that are active in numeric_value. The following example demonstrates using the BIT_COUNT() function to return the number of active bits for a range of numbers −

```
SQL> SELECT
        BIT_COUNT(2) AS TWO,
        BIT_COUNT(4) AS FOUR,
        BIT_COUNT(7) AS SEVEN
+-----+------+-------+
| TWO | FOUR | SEVEN |
+-----+------+-------+
|   1 |    1 |     3 |
+-----+------+-------+
```

1 row in set (0.00 sec)

**BIT_OR(expression)-**The BIT_OR() function returns the bitwise OR of all the bits in expression. The basic premise of the bitwise OR function is that it returns 0 if the corresponding bits match and 1 if they do not. The function returns a 64-bit integer, and if there are no matching rows, then it returns 0. The following example performs the BIT_OR() function on the PRICE column of the CARS table, grouped by the MAKER −

```
SQL> SELECT
        MAKER, BIT_OR(PRICE) BITS
        FROM CARS GROUP BY MAKER
+-----------------------------------------------------+
|MAKER          BITS                                  |
+-----------------------------------------------------+
```

|CHRYSLER 62293 |

|FORD 16127 |

|HONDA 32766 |

+----------+

1 row in set (0.00 sec)

**CEIL(X)**

**CEILING(X)-**These functions return the smallest integer value that is not smaller than X. Consider the following example −

SQL> SELECT CEILING(3.46);

+----------+

| CEILING(3.46) |

+----------+

| 4 |

+----------+

1 row in set (0.00 sec)

SQL> SELECT CEIL(-6.43);

+----------+

| CEIL(-6.43) |

+----------+

| -6 |

+----------+

1 row in set (0.00 sec)

**CONV(N,from_base,to_base)-**The purpose of the CONV() function is to convert numbers between different number bases. The function returns a string of the value N converted from from_base to to_base. The minimum base value is 2 and the maximum is 36. If any of the arguments are NULL, then the function returns NULL. Consider the following example, which converts the number 5 from base 16 to base 2 −

SQL> SELECT CONV(5,16,2);

+----------+

| CONV(5,16,2) |

+----------+

| 101 |

+----------+

1 row in set (0.00 sec)

**COS(X)-**This function returns the cosine of X. The value of X is given in radians.

SQL>SELECT COS(90);

+----------+

| COS(90) |

+----------+

| -0.44807361612917 |

+----------+

1 row in set (0.00 sec)

**COT(X)-**This function returns the cotangent of X. Consider the following example −

SQL>SELECT COT(1);

+----------+

| COT(1) |

+----------+

| 0.64209261593433 |

+----------+

1 row in set (0.00 sec)

**DEGREES(X)-**This function returns the value of X converted from radians to degrees.

SQL>SELECT DEGREES(PI());

+----------+

| DEGREES(PI()) |

+----------+

| 180.000000 |

+----------------------------------------------------+

1 row in set (0.00 sec)

**EXP(X**)-This function returns the value of e (the base of the natural logarithm) raised to the power of X.

SQL>SELECT EXP(3);

+----------------------------------------------------+

| EXP(3) |

+----------------------------------------------------+

| 20.085537 |

+----------------------------------------------------+

1 row in **set (0.00 sec)**

**FLOOR**(X)-This function returns the largest integer value that is not greater than X.

SQL>SELECT FLOOR(7.55);

+----------------------------------------------------+

| FLOOR(7.55) |

+----------------------------------------------------+

| 7 |

+----------------------------------------------------+

1 row in set (0.00 sec)

**FORMAT(X,D**)-The FORMAT() function is used to format the number X in the following format: ###,###,###.## truncated to D decimal places. The following example demonstrates the use and output of the FORMAT() function −

SQL>SELECT FORMAT(423423234.65434453,2);

+----------------------------------------------------+

| FORMAT(423423234.65434453,2) |

+----------------------------------------------------+

| 423,423,234.65 |

+----------------------------------------------------+

1 row in set (0.00 sec)

**GREATEST(n1,n2,n3,.....)**-The GREATEST() function returns the greatest value in the set of input parameters (n1, n2, n3, a nd so on). The following example uses the GREATEST() function to return the largest number from a set of numeric values −

```
SQL>SELECT GREATEST(3,5,1,8,33,99,34,55,67,43);

+-----------------------------------------------------+

| GREATEST(3,5,1,8,33,99,34,55,67,43)                 |

+-----------------------------------------------------+

| 99                                                  |

+-----------------------------------------------------+
```

1 row in set (0.00 sec)

**INTERVAL(N,N1,N2,N3,.....)**-The INTERVAL() function compares the value of N to the value list (N1, N2, N3, and so on ). The function returns 0 if N < N1, 1 if N < N2, 2 if N <N3, and so on. It will return .1 if N is NULL. The value list must be in the form N1 < N2 < N3 in order to work properly. The following code is a simple example of how the INTERVAL() function works −

```
SQL>SELECT INTERVAL(6,1,2,3,4,5,6,7,8,9,10);

+-----------------------------------------------------+

| INTERVAL(6,1,2,3,4,5,6,7,8,9,10)                    |

+-----------------------------------------------------+

| 6                                                   |

+-----------------------------------------------------+
```

1 row in set (0.00 sec)

**INTERVAL(N,N1,N2,N3,.....)**

The INTERVAL() function compares the value of N to the value list (N1, N2, N3, and so on ). The function returns 0 if N < N1, 1 if N < N2, 2 if N <N3, and so on. It will return .1 if N is NULL. The value list must be in the form N1 < N2 < N3 in order to work properly. The following code is a simple example of how the INTERVAL() function works −

```
SQL>SELECT INTERVAL(6,1,2,3,4,5,6,7,8,9,10);
```

```
+-----------------------------------------------------+
| INTERVAL(6,1,2,3,4,5,6,7,8,9,10)                    |
+-----------------------------------------------------+
| 6                                                   |
+-----------------------------------------------------+
```

1 row in set (0.00 sec)

Remember that 6 is the zero-based index in the value list of the first value that was greater than N. In our case, 7 was the offending value and is located in the sixth index slot.

**LEAST(N1,N2,N3,N4,.....)**The LEAST() function is the opposite of the GREATEST() function. Its purpose is to return the least-valued item from the value list (N1, N2, N3, and so on). The following example shows the proper usage and output for the LEAST() function −

```
SQL>SELECT LEAST(3,5,1,8,33,99,34,55,67,43);
+-----------------------------------------------------+
| LEAST(3,5,1,8,33,99,34,55,67,43)                    |
+-----------------------------------------------------+
| 1                                                   |
+-----------------------------------------------------+
```

1 row in set (0.00 sec)

**LOG(X)**

**LOG(B,X**)-The single argument version of the function will return the natural logarithm of X. If it is called with two arguments, it returns the logarithm of X for an arbitrary base B. Consider the following example −

```
SQL>SELECT LOG(45);
+-----------------------------------------------------+
| LOG(45)                                             |
+-----------------------------------------------------+
| 3.806662                                            |
+-----------------------------------------------------+
```

1 row in set (0.00 sec)

SQL>SELECT LOG(2,65536);

```
+-----------------------------------------------------+
| LOG(2,65536)                                        |
+-----------------------------------------------------+
| 16.000000                                           |
+-----------------------------------------------------+
```

1 row in set (0.00 sec)

**LOG10(X)-**This function returns the base-10 logarithm of X.

SQL>SELECT LOG10(100);

```
+-----------------------------------------------------+
| LOG10(100)                                          |
+-----------------------------------------------------+
| 2.000000                                            |
+-----------------------------------------------------+
```

1 row in set (0.00 sec)

**MOD(N,M**)-This function returns the remainder of N divided by M. Consider the following example −

SQL>SELECT MOD(29,3);

```
+-----------------------------------------------------+
| MOD(29,3)                                           |
+-----------------------------------------------------+
| 2                                                   |
+-----------------------------------------------------+
```

1 row in set (0.00 sec)

**OCT(N)-**The OCT() function returns the string representation of the octal number N. This is equivalent to using CONV(N,10,8).

SQL>SELECT OCT(12);

```
+-----------------------------------------------------+
| OCT(12)                                             |
```

+----------+

| 14 |

+----------+

1 row in set (0.00 sec)

**PI()-**This function simply returns the value of pi. SQL internally stores the full double-precision value of pi.

SQL>SELECT PI();

+----------+

| PI() |

+----------+

| 3.141593 |

+----------+

1 row in set (0.00 sec)

**POW(X,Y)**

**POWER(X,Y**)-These two functions return the value of X raised to the power of Y.

SQL> SELECT POWER(3,3);

+----------+

| POWER(3,3) |

+----------+

| 27 |

+----------+

1 row in set (0.00 sec)

**RADIANS(X**)-This function returns the value of X, converted from degrees to radians.

SQL>SELECT RADIANS(90);

+----------+

| RADIANS(90) |

+----------+

|1.570796 |

+----------+

1 row in set (0.00 sec)

**ROUND(X)**

**ROUND(X,D**)-This function returns X rounded to the nearest integer. If a second argument, D, is supplied, then the function returns X rounded to D decimal places. D must be positive or all digits to the right of the decimal point will be removed. Consider the following example −

SQL>SELECT ROUND(5.693893);

+----------+

| ROUND(5.693893) |

+----------+

| 6 |

+----------+

1 row in set (0.00 sec)

SQL>SELECT ROUND(5.693893,2);

+----------+

| ROUND(5.693893,2) |

+----------+

| 5.69 |

+----------+

1 row in set (0.00 sec)

**SIGN(X)-**This function returns the sign of X (negative, zero, or positive) as -1, 0, or 1.

SQL>SELECT SIGN(-4.65);

+----------+

| SIGN(-4.65) |

+----------+

| -1 |

+----------+

1 row in set (0.00 sec)

SQL>SELECT SIGN(0);

+----------+

| SIGN(0) |

+----------+

| 0 |

+----------+

1 row in set (0.00 sec)

SQL>SELECT SIGN(4.65);

+------------+

| SIGN(4.65) |

+------------+

| 1 |

+------------+

1 row in set (0.00 sec)

**SIN(X)-**This function returns the sine of X. Consider the following example −

SQL>SELECT SIN(90);

+----------+

| SIN(90) |

+----------+

| 0.893997 |

+----------+

1 row in set (0.00 sec)

**SQRT(X)-**This function returns the non-negative square root of X. Consider the following example −

SQL>SELECT SQRT(49);

+-----------+

| SQRT(49) |

+----------------------------------------------------+

| 7 |

+----------------------------------------------------+

1 row in set (0.00 sec)

STD(expression)

STDDEV(expression)

The STD() function is used to return the standard deviation of expression. This is equivalent to taking the square root of the VARIANCE() of expression. The following example computes the standard deviation of the PRICE column in our CARS table −

SQL>SELECT STD(PRICE) STD_DEVIATION FROM CARS;

+----------------------------------------------------+

| STD_DEVIATION |

+----------------------------------------------------+

| 7650.2146 |

+----------------------------------------------------+

1 row in set (0.00 sec)

**TAN(X)-**This function returns the tangent of the argument X, which is expressed in radians.

SQL>SELECT TAN(45);

+----------------------------------------------------+

| TAN(45) |

+----------------------------------------------------+

| 1.619775 |

+----------------------------------------------------+

1 row in set (0.00 sec)

**TRUNCATE(X,D)-**This function is used to return the value of X truncated to D number of decimal places. If D is 0, then the decimal point is removed. If D is negative, then D number of values in the integer part of the value is truncated. Consider the following example −

SQL>SELECT TRUNCATE(7.536432,2);

+----------------------------------------------------+

```
| TRUNCATE(7.536432,2)                                |
+-----------------------------------------------------+
| 7.53                                                |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

## DATE FUNCTIONS

### AIM:

To learn date functions in mysql

### Procedure

ADDDATE(date,INTERVAL expr unit), ADDDATE(expr,days). When invoked with the INTERVAL form of the second argument, ADDDATE() is a synonym for DATE_ADD(). The related function SUBDATE() is a synonym for DATE_SUB(). For information on the INTERVAL unit argument, see the discussion for DATE_ADD().

```
mysql> SELECT DATE_ADD('1998-01-02', INTERVAL 31 DAY);
+-----------------------------------------------------+
| DATE_ADD('1998-01-02', INTERVAL 31 DAY)             |
+-----------------------------------------------------+
| 1998-02-02                                          |
+-----------------------------------------------------+
1 row in set (0.00 sec)
mysql> SELECT ADDDATE('1998-01-02', INTERVAL 31 DAY);
+-----------------------------------------------------+
| ADDDATE('1998-01-02', INTERVAL 31 DAY)              |
+-----------------------------------------------------+
| 1998-02-02                                          |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

When invoked with the days form of the second argument, MySQL treats it as an integer number of days to be added to expr.

```
mysql> SELECT ADDDATE('1998-01-02', 31);
+-------------------------------------------------------+
| DATE_ADD('1998-01-02', INTERVAL 31 DAY)               |
+-------------------------------------------------------+
| 1998-02-02                                            |
+-------------------------------------------------------+
1 row in set (0.00 sec)
```

**ADDTIME(expr1,expr2)**-ADDTIME() adds expr2 to expr1 and returns the result. The expr1 is a time or datetime expression, while the expr2 is a time expression.

```
mysql> SELECT ADDTIME('1997-12-31 23:59:59.999999','1 1:1:1.000002');
+-------------------------------------------------------+
| DATE_ADD('1997-12-31 23:59:59.999999','1 1:1:1.000002') |
+-------------------------------------------------------+
| 1998-01-02 01:01:01.000001                            |
+-------------------------------------------------------+
1 row in set (0.00 sec)
```

**CONVERT_TZ(dt,from_tz,to_tz)**

This converts a datetime value dt from the time zone given by from_tz to the time zone given by to_tz and returns the resulting value. This function returns NULL if the arguments are invalid.

```
mysql> SELECT CONVERT_TZ('2004-01-01 12:00:00','GMT','MET');
+-------------------------------------------------------+
| CONVERT_TZ('2004-01-01 12:00:00','GMT','MET')         |
+-------------------------------------------------------+
| 2004-01-01 13:00:00                                   |
+-------------------------------------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT CONVERT_TZ('2004-01-01 12:00:00','+00:00','+10:00');
+-----------------------------------------------------+
| CONVERT_TZ('2004-01-01 12:00:00','+00:00','+10:00') |
+-----------------------------------------------------+
| 2004-01-01 22:00:00                                 |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

**CURDATE()**

Returns the current date as a value in 'YYYY-MM-DD' or YYYYMMDD format, depending on whether the function is used in a string or in a numeric context.

```
mysql> SELECT CURDATE();
+------------------------------------------------+
| CURDATE()                                      |
+------------------------------------------------+
| 1997-12-15                                     |
+------------------------------------------------+
1 row in set (0.00 sec)
mysql> SELECT CURDATE() + 0;
+------------------------------------------------+
| CURDATE() + 0                                  |
+------------------------------------------------+
| 19971215                                       |
+------------------------------------------------+
1 row in set (0.00 sec)
```

**CURRENT_DATE and CURRENT_DATE()**

**CURRENT_DATE and CURRENT_DATE() are synonyms for CURDATE()**

**CURTIME()**

Returns the current time as a value in 'HH:MM:SS' or HHMMSS format, depending on whether the function is used in a string or in a numeric context. The value is expressed in the current time zone.

```
mysql> SELECT CURTIME();
+-----------------------------------------------------+
| CURTIME()                                           |
+-----------------------------------------------------+
| 23:50:26                                            |
+-----------------------------------------------------+
1 row in set (0.00 sec)
mysql> SELECT CURTIME() + 0;
+-----------------------------------------------------+
| CURTIME() + 0                                       |
+-----------------------------------------------------+
| 235026                                              |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

CURRENT_TIME and CURRENT_TIME()

CURRENT_TIME and CURRENT_TIME() are synonyms for CURTIME().

CURRENT_TIMESTAMP and CURRENT_TIMESTAMP()

CURRENT_TIMESTAMP and CURRENT_TIMESTAMP() are synonyms for NOW().

**DATE(expr)**

Extracts the date part of the date or datetime expression expr.

```
mysql> SELECT DATE('2003-12-31 01:02:03');
+-----------------------------------------------------+
| DATE('2003-12-31 01:02:03')                         |
+-----------------------------------------------------+
|  2003-12-31                                         |
```

+----------------------------------------------------+

1 row in set (0.00 sec)

**DATEDIFF(expr1,expr2)**

DATEDIFF() returns expr1 . expr2 expressed as a value in days from one date to the other. Both expr1 and expr2 are date or date-and-time expressions. Only the date parts of the values are used in the calculation.

mysql> SELECT DATEDIFF('1997-12-31 23:59:59','1997-12-30');

+----------------------------------------------------+

| DATEDIFF('1997-12-31 23:59:59','1997-12-30') |

+----------------------------------------------------+

| 1 |

+----------------------------------------------------+

1 row in set (0.00 sec)

DATE_ADD(date,INTERVAL expr unit), DATE_SUB(date,INTERVAL expr unit)

These functions perform date arithmetic. The date is a DATETIME or DATE value specifying the starting date. The expr is an expression specifying the interval value to be added or subtracted from the starting date. The expr is a string; it may start with a '-' for negative intervals.A unit is a keyword indicating the units in which the expression should be interpreted.

The INTERVAL keyword and the unit specifier are not case sensitive.

mysql> SELECT DATE_ADD('1997-12-31 23:59:59',

-> INTERVAL '1:1' MINUTE_SECOND);

+----------------------------------------------------+

| DATE_ADD('1997-12-31 23:59:59', INTERVAL... |

+----------------------------------------------------+

| 1998-01-01 00:01:00 |

```
+-----------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_ADD('1999-01-01', INTERVAL 1 HOUR);
+-----------------------------------------------------+
| DATE_ADD('1999-01-01', INTERVAL 1 HOUR)             |
+-----------------------------------------------------+
| 1999-01-01 01:00:00                                 |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

## DATABASE CONNECTIVITY IN PHP WITH MYSQL

### AIM:

To validate a form using PHP with mysql

### Program for validation

```
<?php
// Create connection
$con=mysqli_connect("localhost","userdb","","");
// Check connection
if (mysqli_connect_errno())
  {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
  }
   else { echo "<h1>You're connected to userdb</h1>"; }
?>
<h2>Enter your username and password</h2>
<br/><br/>
<h1>Name:</h1> <input type="text" name="name">   <br>
<h1>Password:</h1> <input type="text" name="password">
```

Notes

```
</div>
<?php
$name = "";
$pw = "";
?>
<html>
<head>
<style>
#main
{
width: 700px;
margin-left: auto;
margin-right: auto;
}
</style>
</head>
<body>
<div id="main">
<?php
// Create connection
$con=mysqli_connect("localhost","userdb","","");
// Check connection
if (mysqli_connect_errno())
  {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
  }
   else { echo "<h1>You're connected to userdb</h1>"; }
?>
<?php
```

```
$name = "";
$pw = "";
?>
<h2>Enter your username and password</h2>
<br/><br/>
<h1>Name:</h1> <input type="text" name="name">   <br>
<h1>Password:</h1> <input type="text" name="password">
</div>
</body>
</html>
```

## FORMATTING INPUT

### AIM:

To format the input in mysql

### Program

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = mysqli_query($conn, $sql);
```

```
if (mysqli_num_rows($result) > 0) {
   // output data of each row
   while($row = mysqli_fetch_assoc($result)) {
      echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
   }
} else {
   echo "0 results";
}
mysqli_close($conn);
?>
```

## PHP SIMPLE PROGRAM –TO PERFORM ARITHMETIC OPERATION

### AIM:

To perform the Arithmetic operations using PHP

### Program

```
<html>
    <head>
    <title>Arithmetical Operators</title>
  </head>
    <body>
      <?php
      $a = 42;
      $b = 20;
      $c = $a + $b;
      echo "Addtion Operation Result: $c <br/>";
      $c = $a - $b;
      echo "Substraction Operation Result: $c <br/>";
      $c = $a * $b;
```

```
      echo "Multiplication Operation Result: $c <br/>";
      $c = $a / $b;
      echo "Division Operation Result: $c <br/>";
      $c = $a % $b;
      echo "Modulus Operation Result: $c <br/>";
       $c = $a++;
      echo "Increment Operation Result: $c <br/>";
      $c = $a--;
      echo "Decrement Operation Result: $c <br/>";
   ?>
  </body>
</html>
```

**This will produce the following result −**

Addtion Operation Result: 62

Substraction Operation Result: 22

Multiplication Operation Result: 840

Division Operation Result: 2.1

Modulus Operation Result: 2

Increment Operation Result: 42

Decrement Operation Result: 43

## ADDING NUMBERS USING FUNCTIONS

### AIM:

Add two numbers using function in PHP

### Program

```
<html>
  <head>
    <title>Arithmetical Operators</title>
  </head>
  <body>
```

Notes

```
<?php
    $a = 42;
    $b = 20;
    $c = $a + $b;
    echo "Addtion Operation Result: $c <br/>";
    $c = $a - $b;
    echo "Substraction Operation Result: $c <br/>";
    $c = $a * $b;
    echo "Multiplication Operation Result: $c <br/>";
    $c = $a / $b;
    echo "Division Operation Result: $c <br/>";
    $c = $a % $b;
    echo "Modulus Operation Result: $c <br/>";
    $c = $a++;
    echo "Increment Operation Result: $c <br/>";

    $c = $a--;
    echo "Decrement Operation Result: $c <br/>";
  ?>
 </body>
</html>
```

## This will produce the following result −

Addtion Operation Result: 62

Substraction Operation Result: 22

Multiplication Operation Result: 840

Division Operation Result: 2.1

Modulus Operation Result: 2

Increment Operation Result: 42

Decrement Operation Result: 43

## PHP WEB PROGRAMS ARRAY AND FUNCTIONS

### AIM:

To create array functions in PHP

### Program

```
$hatchbacks = array(

        "Suzuki" => "Baleno",

        "Skoda" => "Fabia",

        "Hyundai" => "i20",

        "Tata" => "Tigor"

    );// friends who own the above cars

$friends = array("Vinod", "Javed", "Navjot", "Samuel");

// let's merge the two arrays into one

$merged = array_merge($hatchbacks, $friends);

//getting only the values

$merged = array_values($merged);

print_r($merged);

?>
```

### OUTPUT

```
Array (
[0] => Baleno
[1] => Fabia
[2] => i20
[3] => Tigor
[4] => Vinod
[5] => Javed
[6] => Navjot
[7] => Samuel
)
```

## CREATE A SIMPLE WEB PAGE IN PHP

### AIM:

To create a simple web page using PHP

### Program

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

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
<!DOCTYPE html>

<html>

<body>

<?php

ECHO "Hello World!<br>";

echo "Hello World!<br>";

EcHo "Hello World!<br>";

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

$color = "red";

echo "My car is " . $color . "<br>";
```

```
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
</body>
</html>
```

**OUTPUT**

HELLOWORLD

## USE OF CONDITIONAL STATEMENTS IN PHP

### AIM:

To create a program with conditional statements in PHP

### Program

The if Statement in PHP

If statement executes some code only if a specified condition is true

### Syntax:

```
if (condition) {
code to be executed if condition is true;
}
```

### The if Statement Example

```
<html>
<body>
<?php
$i=0;
/* If condition is true, statement is
executed*/
if($i==0)
echo "i is 0";
?>
<body>
```

</html>

**OUTPUT** of the above given Example is as follows:

i is 0

## The if…else Statement in PHP

If…else statement executes some code if a condition is true and some another code if the condition is false

## Syntax:

if (condition) {

code to be executed if condition is true;

}

else {

code to be executed if condition is false;

}

The if…else Statement Example

<html>

<body>

<?php

$i=1;

/* If condition is true, statement1 is

executed, else statement2 is executed*/

if($i==0)

echo "i is 0"; //statement1

else

echo "i is not 0"; //statement2

?>

<body>

</html>

**OUTPUT** of the above given Example is as follows:

i is not 0

## The if…elseif…else Statement in PHP

If…elseif…else statement selects one of several blocks of code to be executed

## Syntax:

```
if (condition) {
code to be executed if condition is true;
}
elseif (condition) {
code to be executed if condition is true;
}
else {
code to be executed if condition is false;
}
```

The if…elseif…else Statement Example (Comparing two numbers)

```
<html>
<body>
<?php
$i=22;
$j=22;
/* If condition1 is true, statement1 is executed,
if condition1 is false and condition2 is true,
statement2 is executed, if both the conditions
are false statement3 is executed */
if($i>$j)
echo "i is greater"; //statement1
elseif($i<$j)
echo "j is greater"; //statement2
else
```

```
echo "numbers are equal"; //Statement3
?>
<body>
</html>
```

**OUTPUT** of the above given Example is as follows:

numbers are equal

## Switch Statement in PHP

Switch statement selects one from multiple blocks of code to be executed

## Syntax:

```
switch (n) {
case label1:
code to be executed if n=label1;
break;
case label2:
code to be executed if n=label2;
break;
 ...
default:
code to be executed if n is different from all labels;
}
```

**Switch Statement Example**

```
<html>
<body>
<?php
$x=3;
/* Expression value is compared with each case
value. If it matches, statements following
case would be executed. Break statement is
```

used to terminate the execution of

statement.*/

switch ($x)

{

case 1:

echo "Number 1";

break;

case 2:

echo "Number 2";

break;

case 3:

echo "Number 3";

break;

default:

echo "No number between 1 and 3";

}

?>

</body>

</html>

Integrating PHP with Embedded System

www.researchdesignlab.com Page 28

**OUTPUT** of the above given Example is as follows:

Number 3

## LOOPING STATEMENTS IN PHP

### AIM:

To use different loops in PHP

### Program

### For loop in PHP

PHP for loop executes a block of code, a specified number of times

**Syntax:**

```
for (initialization; test condition; increment/decrement) {
code to be executed;
}
```

For loop Example

```
<html>
<body>
<?php
echo "Numbers from 1 to 20 are: <br>";
/*in for loop, initialization usually declares
a loop variable, condition is a Boolean
expression such that if the condition is true,
loop body will be executed and after each
iteration of loop body, expression is executed
which usually increase or decrease loop
variable*/
for ($x=0; $x<=20; $x++) {
echo "$x ";
}
?>
</body>
</html>
```

OUTPUT of the above given Example is as follows:

Numbers from 1 to 20 are:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

**Declaring multiple variables in for loop Example**

```
<html>
```

<body>

<?php

/* Multiple variables can be declared in

declaration block of for loop */

for ($x=0,$y=1,$z=2;$x<=3;$x++) {

echo "x = $x, y = $y, z = $z <br>";

}

?>

</body>

</html>

OUTPUT of the above given Example is as follows:

x = 0, y = 1, z = 2

x = 1, y = 1, z = 2

x = 2, y = 1, z = 2

x = 3, y = 1, z = 2

## While Loop in PHP

While loop, loops through a block of code as long as the specified condition

is true

## Syntax:

while (condition) {

code to be executed;

}

While Loop Example

<html>

<body>

<?php

$i=1;

/* here <condition> is a Boolean expression.

Loop body is executed as long as condition is

true*/

while($i<5){

echo "i is = $i <br>";

$i++;

}

?>

</body>

</html>

**OUTPUT** of the above given Example is as follows:

i is = 1

i is = 2

i is = 3

i is = 4

## Do While loop in PHP

Do while loop will always execute the block of code once, it will then check

the condition, and if the condition is true then it repeats the loop

## Syntax:

do {

code to be executed;

} while (condition );

Do While loop Example

<html>

<body>

<?php

$i=1;

/* here <condition> is a Boolean expression. Please

note that the condition is evaluated after executing

```
the loop body. So loop will be executed at least
once even if the condition is false*/
do
{
echo "i is = $i <br>";
$i++;
}while($i<5);
?>
</body>
</html>
```

**OUTPUT** of the above given Example is as follows:

```
i is = 1
i is = 2
i is = 3
i is = 4
```

## CREATE USER DEFINED FUNCTIONS

## AIM:

To implement user defined functions in PHP

## Program

**User Defined Function in PHP**

Functions are group of statements that can perform a task

## Syntax:

```
function functionName() {
code to be executed;
}
```

User Defined Function Example

```
<html>
<body>
<?php
```

```
// Function definition
function myFunction()
{
echo "Hello world";
}
// Function call
myFunction();
?>
</body>
</html>
```

**OUTPUT** of the above given Example is as follows:

Hello world

## TYPES OF ARRAYS IN PHP

### AIM:

To implement different types of Arrays in PHP

### Array in PHP

- An array stores multiple values in one single variable
- In PHP, there are three kinds of arrays:
- Numeric array
- Associative array
- Multidimensional array
- Numeric Array in PHP

### Numeric array is an array with a numeric index

Numeric Array Example

```
<html>
<body>
<?php
```

```
/* An array $flower_shop is created with three
Values - rose, daisy,orchid */
$flower_shop = array (
"rose",
"daisy",
"orchid"
);
/* Values of array $flower_shop is displayed based
on index. The starting index of an array is Zero */
echo "Flowers: ".$flower_shop[0].",
".$flower_shop[1].", ".$flower_shop[2]."";
?>
</body>
</html>
```

**OUTPU**T of the above given Example is as follows:

Flowers: rose, daisy, orchid

## Associative array in PHP

Associative array is an array where each ID key is associated with a value

Associative array Example

```
<html>
<body>
<?php
/* Here rose, daisy and orchid indicates ID key and
5.00, 4.00, 2.00 indicates their values respectively
*/
$flower_shop = array (
"rose" => "5.00",
"daisy" => "4.00",
```

```
"orchid" => "2.00"
);
// Display the array values
echo "rose costs
.$flower_shop['rose'].",daisy costs
".$flower_shop['daisy'].",and orchild
costs ".$flower_shop['orchild']."";
?>
</body>
</html>
```

**OUTPUT** of the above given Example is as follows:

rose costs 5.00,daisy costs 4.00,and orchild costs

## Loop through an Associative Array

```
<html>
<body>
<?php
$flower_shop=array("rose"=>"5.00",
"daisy"=>"4.00","orchid"=>"2.00");
/* for each loop works only on arrays, and is used
 to loop through each key/value pair in an array */
foreach($flower_shop as $x=>$x_value) {
echo "Flower=" . $x .
", Value=" . $x_value;
echo "<br>";
}
?>
</body>
```

```
</html>
```

**OUTPUT** of the above given Example is as follows:

Flower=rose, Value=5.00

Flower=daisy, Value=4.00

Flower=orchid, Value=2.00

## Multidimensional array in PHP

Multidimensional array is an array containing one or more arrays

Multidimensional array Example

```
<html>
<body>
 <?php
 /* Here $flower_shop is an array, where rose, daisy and orchid
are the ID key which indicates rows and points to array which
have column values. */
 $flower_shop = array(
 "rose" => array( "5.00", "7 items", "red" ),
 "daisy" => array( "4.00", "3 items", "blue" ),
 "orchid" => array( "2.00", "1 item", "white" ),
 );
/* in the array $flower_shop['rose'][0], ‘rose’ indicates row
and ‘0’ indicates column */
 echo "rose costs ".$flower_shop['rose'][0].
 ", and you get ".$flower_shop['rose'][1].".<br>";
 echo "daisy costs ".$flower_shop['daisy'][0].
 ", and you get ".$flower_shop['daisy'][1].".<br>";
 echo "orchid costs ".$flower_shop['orchid'][0].
 ", and you get ".$flower_shop['orchid'][1].".<br>";
 ?>
```

```
</body>
```

```
</html>
```

**OUTPUT** of the above given Example is as follows:

rose costs 5.00, and you get 7 items.

daisy costs 4.00, and you get 3 items.

orchid costs 2.00, and you get 1 item.

## HANDLING COOKIES IN PHP

## AIM:

To implement cookies in PHP

## Program

Setting new cookie

```
============================
<?php
setcookie("name","value",time()+$int);
/*name is your cookie's name
value is cookie's value
$int is time of cookie expires*/
?>
```

Getting Cookie

```
============================
<?php
echo $_COOKIE["your cookie name"];
?>
```

Updating Cookie

```
============================
<?php
setcookie("color","red");
echo $_COOKIE["color"];
/*color is red*/
/* your codes and functions*/
setcookie("color","blue");
echo $_COOKIE["color"];
/*new color is blue*/
?>
```

Deleting Cookie

```
============================
```

```
<?php
unset($_COOKIE["yourcookie"]);
/*Or*/
setcookie("yourcookie","yourvalue",time()-1);
/*it expired so it's deleted*/
?>
```

**\HANDLING FILES IN PHP**

## AIM:

To implement basic file operations such as open close read and write in PHP

## Program

```
<?php
$fileName = "/doc/myFile.txt";
$fp = fopen($fileName,"r");
if( $fp == false )
{
  echo ( "Error in opening file" );
  exit();
}
?>
<?php
$fileName = "/doc/myFile.txt";
$fp = fopen($fileName,"r");
if( $fp == false )
{
  echo ( "Error in opening file" );
  exit();
}

$fileSize = filesize( $fileName );
$fileData = fread( $fp, $fileSize );
?>
<?php

$fileName = "/doc/myFile.txt";

$fp = fopen($fileName,"r");

if( $fp == false )

{

  echo ( "Error in opening file" );

  exit();

}
```

```
while(!feof($fp))
{
  echo fgets($fp). "<br>";
}
?>
```

```
<?php
$fileName = "/doc/myFile.txt";
$fp = fopen($fileName,"w");
if( $fp == false )
{
  echo ( "Error in opening file" );
  exit();
}
fwrite( $fp, "This is a sample text to write\n" );
?>
```

```
<?php
$fileName = "/doc/myFile.txt";
$fp = fopen($fileName,"w");
if( $fp == false )
{
  echo ( "Error in opening file" );
  exit();
}
 //some code to be executed
  fclose( $fp );
?>
```

## SESSIONS IN PHP

## AIM:

To implement sessions in PHP

## Program

**Start a session**

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>
</body>
</html>
```

Session Variables

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>


<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>
</body>
```

```
</html>
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>
</body>
</html>
```

## Destroy a PHP Session

## Example:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// remove all session variables
session_unset();
// destroy the session
session_destroy();
?>
```

```
</body>
</html>
```

## SIMPLE APPLICATION USING PHP

### AIM:

To Implement a student Registration in PHP and Save and Display the student Records.

### Program

```
<html>
<head>
<title>general form</title>
</head>
<body bgcolor="aakk">
<form action = "<?php $_PHP_SELF ?>" method = "POST">
Name:
<input type = "text" name = "txtname">
<br><br>
Roll no.:
<input type = "text" name = "txtr_no">
<br><br>
Gender:
<input type = "text" name = "txtgen">
<br><br>
PHP With MySQL (2360701) 6
th CE
N. B. Patel Polytechnic, Piludara Prepared By: Bipin Prajapati
36
Address:
<textarea name = "add" type = "textarea"></textarea>
<br><br>
<input type = "Submit" name = "insert" value = "Save">
<input type = "Reset" value = "Cancle">
```

```
</form>
</body>
</html>
<?php
if(isset($_POST['insert']))
 {
 $con = mysql_connect("localhost","root","");
 if($con)
 {
echo "Mysql connection ok<br>";
mysql_select_db("studinfo",$con);
$name = strval($_POST['txtname']);
$rollno = intval($_POST['txtr_no']);
$gender = strval($_POST['txtgen']);
$address = strval($_POST['add']);
$insert = "insert into info values('$name',$rollno,'$gender','$address')";
if(mysql_query($insert,$con))
{
echo "Data inserted successfully<br>";
}
$query = "select * from info";
$sldt = mysql_query($query,$con);
echo "<table border='1'>
<tr>
<th>Name</th>
<th>Roll No</th>
<th>Gender</th>
<th>Address</th>
</tr>";
while($row = mysql_fetch_array($sldt))
```

PHP With MySQL (2360701) 6

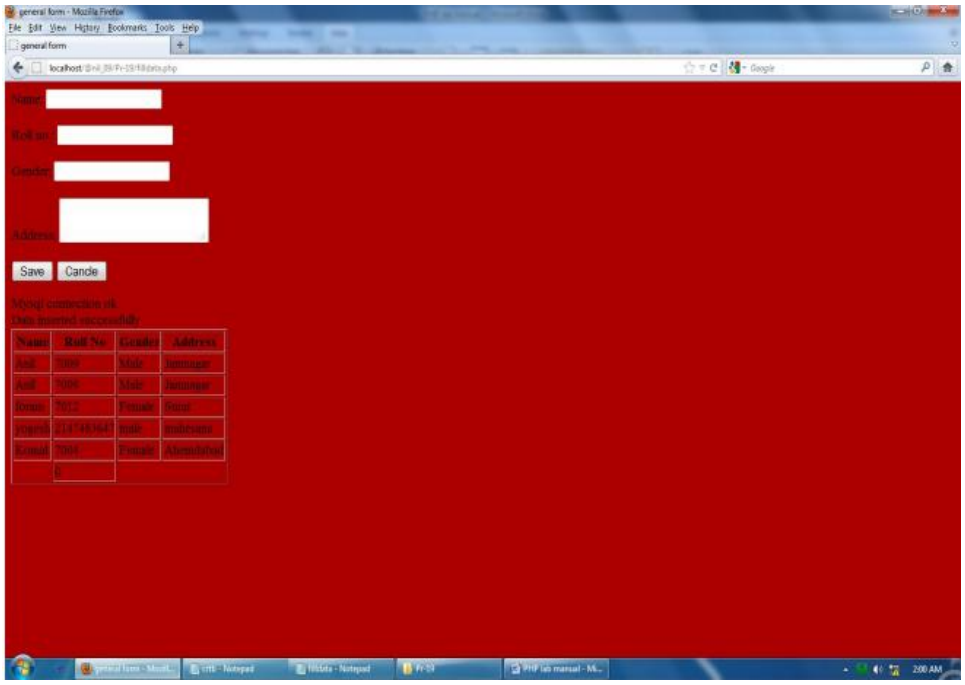
th CE

N. B. Patel Polytechnic, Piludara Prepared By: Bipin Prajapati

37

```
{
echo "<tr>";
echo "<td>".$row['name']."</td>";
echo "<td>".$row['rollno']."</td>";
echo "<td>".$row['gen']."</td>";
echo "<td>".$row['address']."</td>";
echo "</tr>";
}
echo "</table>";
mysql_close($con);
 }
 }
?>
```

O/P



## CREATING TABLES WITH CONSTRAINTS

*Notes*

### AIM:

To create tables in PHP

### Program

```
<html>
<head>
<title>Create Database. </title>
</head>
<body>
 <?php
$con = mysql_connect("localhost","root","");
if(!$con)
{
 die("not opened");
}
echo "Connection open"."</br>";
$db = mysql_select_db("studinfo",$con);
if(!$db)
{
 die("Database not found".mysql_error());
}
echo "Database is selected"."</br>";
 $query = "select * from computer";
$sldt = mysql_query($query,$con);
if(!$sldt)
{
 die("data not selected".mysql_error());
}
echo "<table border='1'>
```

```
<tr>
<th>ID</th>
<th>Name</th>
<th>Branch</th>
</tr>";
while($row = mysql_fetch_array($sldt))
{
echo "<tr>";
echo "<td>".$row['id']."</td>";
echo "<td>".$row['name']."</td>";
echo "<td>".$row['branch']."</td>";
echo "</tr>";
}
```

PHP With MySQL (2360701) 6

th CE

N. B. Patel Polytechnic, Piludara Prepared By: Bipin Prajapati

35

```
echo "</table>";
 ?>
</body>
</html>
```

O/P:

Connection open

Database is selected

ID Name Branch

9 Anil J Basantani CE

9 Anil J Basantani CE

9 Anil J Basantani CE

# TEXT PROCESSING WITH PERL

## AIM:

To learn how to do some common text processing tasks using Perl.

## Introduction:

Perl is a high-level, general-purpose, interpreted, dynamic programming language. Perl was originally developed by Larry Wall in 1987 as a general-purpose Unix scripting language to make report processing easier.

Perl borrows features from other programming languages including C, shell scripting (sh), AWK, and sed. The language provides powerful text processing facilities without the arbitrary data length limits of many contemporary Unix tools, facilitating easy manipulation of text files.

Though originally developed for text manipulation, Perl is used for a wide range of tasks including system administration, web development, network programming, games, bioinformatics, and GUI development.

The language is intended to be practical (easy to use, efficient, complete) rather than beautiful (tiny, elegant, minimal). Its major features include support for multiple programming paradigms (procedural, object-oriented, and functional styles), reference counting memory management , built-in support for text processing, and a large collection of third-party modules.

CPAN, the Comprehensive Perl Archive Network, is an archive of over 90,000 modules of software written in Perl, as well as documentation for it. It has a presence on the World Wide Web at w ww .cpan .o rg and is mirrored worldwide at more than 200 locations. CPAN can denote either the archive network itself, or the Perl program that acts as an interface to the network and as an automated software installer (somewhat like a package manager). Most software on CPAN is free and open source software.

This exercise consists of 7 programs of increasing complexity in Perl.

Description:

Students willl write seven programs in Perl and test their results. The programs willl also use 3rd party modules to the language. The third party modules willl be installed from the distribution packages rather than through CPAN.

## Pre-requisites:

Perl is installed by default in all Linux distributions. So the students can start programming with any text editor of their choice. When a program

requires a third-party module and support files it willl be mentioned with instructions on how to install them.

The Programs:

The seven programs to be done in this exercise are:

1. Hello World

2. Greeting the user

3. Analysing text from a file and printing some statistics

4. Proper command line processing and analysing a text file to get word frequency, word size frequency and the type-token ratio.

5. Text analysis and outputting the result to another text file with proper formatting.

6. Read data from a flat file using Perl's database interface and performing SQL queries on the data.

7. Read rainfall data from a csv file, do some computations and produce a graph based on the results.

Create a new directory for the programs.

> mkdir perl_exercises

> cd perl_exercises

Download the supporting materials zip file to the newly created directory and unzip the contents to the directory.

## 1. Hello World

Create a new file using the gedit text editor.

> gedit hello.pl Use the following code:

#!/usr/bin/env perl

#The above statement tells the system that this is a perl program.

print "Hello World!\n"; #print the text Hello World and a newline. Save the file.

Now run the program as follows:

> perl hello.pl Hello World! >

The above command asks the perl interpreter to load the file called hello.pl and execute it. On execution the text "Hello World" is printed on the screen.

## 2. Greeting the user

This program asks the user's name and the year of birth. It then greats the user and tells the age of the user.

*Notes*

> gedit name.pl

The Code:

```
#!/usr/bin/env perl

#

# name.pl

print "Enter you name and press return:";

$name=<STDIN>;   #read the data

chomp($name);    #remove the newline

print "\nEnter your birth year and press return:";

$byear=<STDIN>;

chomp($byear);

#localtime gives the data with 9 distinct values. Collect them. my ($sec, $min, $hour, $mday, $mon, $year, $wday, $yday, $dst) = localtime time;

$age=($year + 1900) - $byear; #the year starts from 1900 according to localtime


print

"\nHello, $name!\n";

print "You are $age years old.\n";
```

On execution:

> perl name.pl

Enter you name and press return:Mickey Mouse

Enter your birth year and press return:1928

Hello, Mickey Mouse! You are 83 years old. >

## 3. Analysing text and printing the statistics

This programs read the text file given in the command line, asks the user for the word to search in the text and prints some statistics about the text. Note that the program will hang if the user fails to give the name of the file when the program is run. Proper handling of commandline arguments is explored in the next exercise.

> gedit words . pl

The Code:

```
#!/usr/bin/env perl

#

#words.pl word FILE

#

#if no data filename is given, this program willl hang

print "Enter the word you want to search for an press return:"; $sword=<STDIN>;

chomp($sword);

$scount = 0;     #search counter $bcount = 0;     #blank line counter

while(<>)  #continue reading as long as there is input {

chomp;     #remove newline from each line

foreach $w (split)     #split each line into words

{

if ($w eq $sword) {

$scount++; #search hit counter

}

$words++;

$char += length($w); }

$bcount++;

}

#if the length of the current line is 0, we have a blank line if (length($_) == 0)

$avgw = $words/$.;     #average words per line including blank lines $avgc = $char/$words;  #average characters per word
```

print "There are $. lines in this file including $bcount blank

lines.\n";

print "There are $words words in this file.\n";

print "There are $char characters in this file.\n";

print "The average number of words per line is $avgw.\n";

print "The average number of characters per word is $avgc.\n";

On execution:

> perl words . pl constitution_preamble.txt

Enter the word you want to search for an press return:the

There are 13 lines in this file including 6 blank lines.

There are 85 words in this file.

There are 470 characters in this file.

The average number of words per line is 6.53846153846154.

print "the word $sword occurs in the text $scount times.\n";

The average number of characters per word is 5.52941176470588. the word the occurs in the text 4 times.

The file constitution_preamble.txt is part of the support file archive which was unzipped at the beginning.

4. Command line processing and more text analysis

This program also reads from a text file and analyses the text. Proper command line handling is now performed. The program converts all input text into lower case and strips off all the punctuation marks in the text. The use of hashes is introduced.

> gedit wordcount.pl The Code:

#!/usr/bin/env perl

#

#wordcount.pl FILE

#if no filename is given, print help and exit

if (length($ARGV[0]) < 1)

```
{
print "Usage is : words.pl word filename\n";
}
exit;
my $file = $ARGV[0];   #filename given in commandline
open(FILE, $file);     #open the mentioned filename
while(<FILE>)          #continue reading until the file ends
{
chomp;
tr/A-Z/a-z/;     #convert all upper case words to lower case
tr/.,:;!?"(){}//d;     #remove some common punctuation symbols

#We are creating a hash with the word as the key.
#Each time a word is encountered, its hash is incremented by 1.
#If the count for a word is 1, it is a new distinct word.
#We keep track of the number of words parsed so far.
#We also keep track of the no. of words of a particular length.
foreach $wd (split) {
$count{$wd}++;
if ($count{$wd} == 1)
{
$dcount++;
}
$wcount++;
$lcount{length($wd)}++; } }
#To print the distinct words and their frequency,
#we iterate over the hash containing the words and their count.
print "\nThe words and their frequency in the text is:\n";
foreach $w (sort keys%count)
```

```
{
print "$w : $count{$w}\n"; }
#For the word length and frequency we use the word length hash print "The word length and frequency in the given text is:\n"; foreach $w (sort keys%lcount) {
print "$w : $lcount{$w}\n"; }
print "There are $wcount words in the file.\n";
print "There are $dcount distinct words in the file.\n";
$ttratio = ($dcount/$wcount)*100;  #Calculating the type-token ratio.
```

On execution:

```
> perl wordcount.pl constitution_preamble.txt
```

The words and their frequency in the text is:

```
1949 : 1
a : 1
adopt : 1
all : 2
among : 1
print "The type-token ratio of the file is $ttratio.\n";
and : 8
assembly : 1 a
ssuring : 1
 belief : 1
 citizens : 1
constituent : 1
constitute : 1
constitution : 1
day : 1
democratic : 1
dignity : 1
```

do : 1

economic : 1

enact : 1

equality : 1

expression : 1

faith : 1

fraternity : 1

give : 1

having : 1

hereby : 1

in : 1

india : 2

individual : 1

integrity : 1

into : 1

its : 1

justice : 1

liberty : 1

nation : 1

november : 1 of : 7

opportunity : 1

our : 1

ourselves : 1

people : 1

political : 1

promote : 1

republic : 1

resolved : 1

secular : 1

secure : 1

social : 1

socialist : 1

solemnly : 1

sovereign : 1

status : 1

the : 5

them : 1

this : 2

thought : 1

to : 5

twenty-sixth : 1

unity : 1

we : 1

worship : 1

The word length and frequency in the given text is:

1 : 1

10

11

12

2

3

4

5

6

7

5

2

2 15 18 6 7 8 7 9 5 There are There are 85 words in the file.

61 distinct words in the file.

The type-token ratio of the file is 71.7647058823

5. Text analysis with results output to another file

This program analyses the text of a file and outputs the results to another file after formatting the output.

> gedit freqcount.pl

The Code:

```
#!/usr/bin/env perl
#


#freqcount.pl FILE
#
use strict; #using strict mode to help us find errors easily
#all variables being used are declared my $file; my $wd; my %count; my $w;
if (@ARGV) #Check if the ARGV array exists. This array is poulated with #the arguments passed to the program.
{
$file = $ARGV[0]; #First argument is the data file name. }
else {
die "Usage : freqcount.pl FILE\n"; #Bail out if no data filename #is given
}
open(FILE, $file);
open(RESULTS, ">freqcount.txt");     #Open the file where the results #willl be written. If it exists it willl be overwritten.
while(<FILE>) {
```

```
chomp;

tr/A-Z/a-z/;

tr/.,:;!?"(){}//d;

foreach $wd (split) {

$count{$wd}++; } }

print RESULTS "Word\t\tFrequency\n";     #Writing to newly opened file

foreach $w (sort by_number keys%count)    #The result willl be sorted
#using our by_number function

{

write(RESULTS); }

close(RESULTS);

#Our sorting function.

#The <=> is used to sort the result in a descending order of frequency.
#The second <=> is used to sort the result based on the length of the #word
if the frequency is same.

sub by_number {

$count{$b} <=> $count{$a} || length($b) <=> length($a); }

#Formatting the results.

#A @ denotes the values to be printed.

#A < stands for left justify text in that position, > stand for right

#justify.

#The formatting ends with a final .

format RESULTS=

@<<<<<<<<<<<<<<< @>>
```

On Execution:

```
$w,  $count{$w}

>        perl freqcount.pl constitution_preamble.txt

>        cat freqcount.txt
```

| Word | Frequency |
|---|---|
| and | 8 |
| of | 7 |
| the | 5 |
| to | 5 |
| india | 2 |
| this | 2 |
| all | 2 |
| twenty-sixth | 1 |
| constitution | 1 |
| opportunity | 1 |
| constituent | 1 |
| individual | 1 |
| constitute | 1 |
| expression | 1 |
| democratic | 1 |
| fraternity | 1 |
| ourselves | 1 |
| integrity | 1 |
| socialist | 1 |
| political | 1 |
| sovereign | 1 |
| solemnly | 1 |
| assembly | 1 |
| citizens | 1 |
| resolved | 1 |
| november | 1 |
| economic | 1 |
| equality | 1 |

| | |
|---|---|
| assuring | 1 |
| republic | 1 |
| thought | 1 |
| dignity | 1 |
| worship | 1 |
| liberty | 1 |
| promote | 1 |
| justice | 1 |
| secular | 1 |
| secure | 1 |
| social | 1 |
| people | 1 |
| belief | 1 |
| nation | 1 |
| status | 1 |
| having | 1 |
| hereby | 1 |
| unity | 1 |
| among | 1 |
| faith | 1 |
| adopt | 1 |
| enact | 1 |
| give | 1 |
| them | 1 |
| 1949 | 1 |
| into | 1 |
| our | 1 |
| day | 1 |

its 1

in 1

do 1

we 1

a 1

**PYTHON PROGRAMMING USING CONDITIONAL SATATEMENTS**

## AIM:

To implement conditional statements in python programming

Decision making statements (Conditional)

Decision making constructs with Boolean expression, an expression returns either TRUE or FALSE

(i.e., 0-false and 1-true). Decision making structure is to perform an action or a calculation only when a certain condition is met. There are four types of decision making structure. They are,

1. if statement (Conditional statement)
2. if … else statement (Alternative statement)
3. elif statement (Chained condition)
4. nested if statement

## if statement:

The program evaluates the condition and will execute statement(s) or process only if the test expression is True.

## Syntax:

```
if (test expression/condition):
        True statement
```

**Program:**

```
num = 3
if num > 0:
    print(num, "is a positive number.")
print("This is always printed.")
```

**Output:**

3 is a positive number

This is always printed

## if…else statement:

The if else statement evaluates condition and will execute body of if only when test condition is True. If the condition is False, body of else is executed.

## Syntax:

```
if (test expression/condition):
        True statement
else:
        False statement
```

## Program:

```
num=3
if (num >= 0):
    print("Positive or Zero")
else:
    print("Negative number")
```

**Output:**

Positive or Zero

## if…elif…else statement:

The elif is short for else if. It allows us to check for multiple expressions. If the condition for if is False, it checks the condition of the next elif block and so on. If all the conditions are False, body of else is executed.

**Syntax:**

```
if (test expression/condition):
        True statement 1
elif (test expression/condition):
True statement 2
else:
```

False statement

Program:

```
num=3
if num > 0:
        print("Positive number")
elif num = = 0:
        print("Zero")
else:
        print("Negative number")
```

**Output:**

Positive number

## Nested if statement:

Nested conditional statements are used whenever there is a need to check for another condition after the first condition has been evaluated as True. if...else statement inside another if...else statement.

## Syntax:

```
if (test expression/condition 1):
if (test expression/condition 2):
                        True statement 1
else:
False statement 1
else:
False statement 2
```

Program:

```
num = int(input("Enter a number: "))
if num >= 0:
if num = = 0:
                print("Zero")
        else:
                print("Positive number")
else:
```

```
    print("Negative number")
```

**Output:**

Enter a number: 5

Positive number

## PHYTHON LOOPING STATEMENTS

## AIM:

To implement loops in Python

## (Looping/ Repetition statement)

Loop statement is to execute a specific block of code in multiple numbers of times. A loop is a programming control structure that facilitates the repetition execution of a statement or group of statement. There are two types of loop statement. They are,

1. while loop
2. for loop

2.3.1 while statement:

A while loop statement in Python programming language repeatedly executes a block of statement until the condition is True. It tests the condition before executing the loop body.

## Syntax:

while (test expression/ condition):

```
    body of loop
```

Program:

```
count = 0
while (count < 5):
    print ('The count is:', count )
    count = count + 1
print ("Good bye!")
```

Output:

The count is:0

The count is:1

The count is:2

The count is:3

The count is:4

Good bye!

## Nested while loop:

Nesting defined as the placing of one while loop inside the body of another while loop.

**Syntax:**

while (test expression/ condition):

while (test expression/ condition):

body of loop

Program:

```
count = 1
while (count!=0):
    while(count<5):
        print ('The count is:', count )
        count = count + 1
print ("Good bye!")
```

**Output:**

The count is:1

The count is:2

The count is:3

The count is:4

Good bye!

## Using else statement with while loops:

If the else statement is used with a while loop, the else statement is executed when the condition become false.

## Program:

```
i=0
while i < 5:
    print(i,"is less than 5")
```

```
    i=i+1
else:
    print(i,"is not less than 5)
```

**Output**

```
0 is less than 5
1 is less than 5
2 is less than 5
3 is less than 5
4 is less than 5
5 is not less than 5
```

## The Infinite while Loop:

While the loop runs continuously without termination

## Program:

```
while (1):
    num = input("Enter a number :")
    print ("You entered: ", num )
print "Good bye!"
```

**Output**

```
Enter a number: 20
You entered:  20
Enter a number: 25
You entered:  25
```

For loop:

for loop executes a sequence of statements that allows a code to be repeated a certain number of times using "range" function.

## Syntax:

```
for  variableName  in  <Group of numbers>:
        body of loop
```

**Program:**

```
num = [6, 5, 3, 8, 4, 2, 5, 4, 11]

sum = 0

for v in num:

        sum = sum+v

print("The sum is", sum)
```

**Output:**

The sum is 48

**Nested for loop:**

Nesting defined as the placing of one for loop inside the body of another for loop.

**Syntax:**

```
for  variableName  in  <Group of numbers>:

        for  variableName  in  <Group of numbers>:

body of loop
```

**Program:**

```
        S=0

        for i in range(3):

                for j in range(3):

                        S=i+j

                        Print(s)

        print(“end”)
```

**Output:**

```
        0

        2

4

End
```

## Using else statement with for loops:

If the else statement is used with a for loop, the else statement is executed when the loop has exhausted iterating the list.

## Program:

```
for i in range(0,5):
        print(i,"is less than 5")
else:
        print(i,"is not less than 5)
```

**Output:**

0 is less than 5

1 is less than 5

2 is less than 5

3 is less than 5

4 is less than 5

5 is not less than 5

## For loop using Range:

The range function specifies a range of integers:

range (start, stop)

The integers between start (inclusive) and stop (exclusive)

## Syntax:

```
for  variableName  in range (start, stop):
        statements
```

## Example:

```
for x in range(1, 6):
        print (x, "squared is", x * x)
```

**Output:**

1 squared is 1

2 squared is 4

3 squared is 9

4 squared is 16

5 squared is 25

How to use range:

range(10) #produces the list: [0,1,2,3,4,5,6,7,8,9]

range(1, 7) #produces the list: [1,2,3,4,5,6]

range(0, 30, 5) #produces the list: [0,5,10,15,20,25]

range(5, -1, -1) #produces the list: [5,4,3,2,1,0]

## For loop using Variable name:

## Syntax:

for variableName in listname:

statements

## Program:

```
list=[“apple”,”orange”,”banana”]
for fruits in list:
        print(“current fruit:”,fruits)
print(“End”)
```

**Output:**

```
Current fruit : apple
Current fruit : orange
Current fruit : banana
End
```

## for loop using Length of the list:

## Program:

```
lst=[“sam”,”abc”,”zara”]
for i in range(len(lst)):
        print(lst[i])
print(“End”)
```

**Output:**

Sam

abc

zara

End

## Jump statement

Control statements change the execution from normal sequence. It controls the flow of program execution to get desire behavior or result. There are three types of control statements. They are,

1. break
2. continue
3. pass

## Break Statement:

The break statement terminates the current loop and resumes execution at the next statement. The most common use for break is when some external condition is triggered requiring a immediate exit from a loop. The break statement can be used in both while and for loops.

## Syntax:

break;

Program:

```
for val in range (1,5):
    if (val = = 3):
        break
    print(val)
print("The end")
```

**Output:**

1

2

The end

## Continue Statement:

The continue statement returns the control to the beginning of the while loop. The continue statement rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop. The continue statement can be used in both while and for loops.

## Syntax:

continue

## Program:

```
for val in range (1,5):
        if (val = = 3):
            continue
        print(val)
print("The end")
```

**Output:**

1

2

4

The end

2.4.3

## Pass Statement:

The pass statement is used when a statement is required syntactically but you do not want any command or code to execute. The pass statement is a null operation; nothing happens when it executes.

## Syntax:

pass

Program:

```
for val in range (1,5):
        if (val = = 3):
            pass
        print(val)
print("The end")
```

**Output:**

1

2

3

4

The end

## SAMPLE EXERCISES ON CONDITIONAL AND LOOP BLOCK

### Program 1:

Write a Python program to convert temperatures to and from celsius, fahrenheit.

```
temp = input("Input the  temperature you like to convert? (e.g., 45F, 102C etc.) : ")

degree = int(temp[:-1])

i_convention = temp[-1]

if i_convention.upper() == "C":

  result = int(round((9 * degree) / 5 + 32))

  o_convention = "Fahrenheit"

elif i_convention.upper() == "F":

  result = int(round((degree - 32) * 5 / 9))

  o_convention = "Celsius"

else:

  print("Input proper convention.")

  quit()

print("The temperature in", o_convention, "is", result, "degrees.")
```

Sample Output:

Input the  temperature you like to convert? (e.g., 45F, 102C etc.) : 104f

The temperature in Celsius is 40 degrees.

## Program 2:

Write a Python program to get the Fibonacci series between 0 to 50.

```
x,y=0,1
while y<50:
    print(y)
    x,y = y,x+y
```

Sample Output:

```
1
1
2
3
5
8
13
21
34
```

## Program 3:

Python code to prints out 0,1,2,3,4

```
count = 0
while True:
    print(count)
    count += 1
    if count >= 5:
        break
# Prints out only odd numbers - 1,3,5,7,9
for x in range(10):
    # Check if x is even
    if x % 2 == 0:
```

```
        continue
    print(x)
```

**Output:**

0

1

2

3

4

1

3

5

7

9

## Program 4:

Python code to prints out 0,1,2,3,4 and then it prints "count value reached 5"

```
count=0
while(count<5):
    print(count)
    count +=1
else:
    print("count value reached %d" %(count))
# Prints out 1,2,3,4
for i in range(1, 10):
    if(i%5==0):
        break
    print(i)
else:
```

print("this is not printed because for loop is terminated because of break but not due to fail in condition")

**Output:**

0

1

2

3

4

count value reached 5

1

2

3

4

## STRING HANDLING IN PYTHON

## AIM:

To perform string manipulation operations

A string is a sequence of characters.

Strings are the data types in Python.

Python treats single quotes the same as double quotes.

```
var1 = 'Hello World!'
var2 = "Python Programming"
```

## Simple program

```
my_string = "Hello“
print(my_string)
```

**Output:**

Hello

## String Slicing method

Given a string s, the syntax for a slice is:

```
s[startIndex:pastIndex]
```

The startIndex is the start index of the string. pastIndex is one past the end of the slice.

If you omit the first index, the slice will start from the beginning. If you omit the last index, the slice will go to the end of the string.

**Program:**

```
var1 = 'Hello World!'
var2 = "Python Programming"
print ("var1[0]: ", var1[0] )
print( "var2[1:5]: ", var2[1:5])
```

**Output:**

```
var1[0]: H
var2[1:5]: ytho
```

**Program:**

```
str = 'programiz‘
print('str = ', str)
print('str[0] = ', str[0])
print('str[-1] = ', str[-1])
print('str[1:5] = ', str[1:5])
print('str[5:-2] = ', str[5:-2])
```

**Output:**

```
str =  programiz
str[0] =  p
str[-1] =  z
str[1:5] =  rogr
str[5:-2] =  am
```

## String Built in Function

### 1-lower()

Converts all uppercase letters in string to lowercase.

**Syntax:**

```
s.lower()
```

**Program:**

```
string = "Hello World"
print (string.lower() )
```

**Output:**

```
Hello world
```

### 2-upper()

returns uppercase version of the string

### Syntax:

```
s.upper()
```

### Program:

```
string = "Hello World"
print (string.upper() )
```

**Output:**

HELLO WORLD

## 3-strip()

returns a string with whitespace removed from the start andend

## Syntax:

s.strip()

## Program:

```
string = "Hello World"
print (string.strip() )
```

**Output:**

HelloWorld

## 4-replace('old', 'new')

returns a string where all occurrences of 'old' have been replaced by 'new‘

## Syntax:

s.replace()

## Program:

```
string = "Hello World"
print (string.replace(“Hello”,”Welcome”) )
```

**Output:**

Welcome World

## 5-split('delim') –

returns a list of substrings separated by the given delimiter.

## Syntax:

s.split()

## Program:

```
string = "Hello World"
print (string.split() )
```

**Output:**

['Hello', 'World']

## 6-join(list) --

opposite of split(), joins the elements in the given list together using the string as the delimiter.

**Syntax:**

s.join()

**Program:**

```
string = ['Hello', 'World']
print (string.join() )
```

**Output:**

"Hello World"

## 7-capitalize()

Capitalizes first letter of string

**Syntax:**

s.capitalize()

**Program:**

```
string = "hello world"
print (string.capitalize() )
```

**Output:**

Hello world

## 8-len(string)-

Returns the length of the string

**Syntax:**

len(string)

**Program:**

```
string = "Hello World"
print (len(string) )
```

**Output:**

10

## 9-isalnum() –

Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise.

**Syntax:**

s.isalnum()

**Program:**

```
string = "Hello World"
print (string.isalnum() )
```

**Output:**

False

## 10-isalpha() -

Returns true if string has at least 1 character and all characters are alphabetic and false otherwise.

## Syntax:

s.alpha()

## Program:

```
string = "Hello World"
print (string.isalpha() )
```

**Output:**

True

## 11-isdigit() -

Returns true if string contains only digits and false otherwise.

## Syntax:

s.isdigit()

## Program:

```
string = "Hello World"
print (string.isdigit() )
```

**Output:**

False

## 12-islower() -

Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise.

## Syntax:

s.islower()

## Program:

```
string = "Hello World"
print (string.islower() )
```

**Output:**

False

## 13-isnumeric() -

Returns true if a unicode string contains only numeric characters and false otherwise.

**Syntax:**

s.isnumeric()

**Program:**

```
string = "Hello World"
print (string.isnumeric() )
```

**Output:**

False

## 14-isspace() -

Returns true if string contains only whitespace characters and false otherwise.

**Syntax:**

s.isspace()

**Program:**

```
string = "Hello World"
print (string.isspace() )
```

**Output:**

----------True

## 15-istitle() -

Returns true if string is properly "titlecased" and false otherwise.

**Syntax:**

s.istitle()

**Program:**

```
string = "Hello World"
print (string.istitle() )
```

**Output:**

----------True

## 16-isupper()

Returns true if string has at least one cased character and all cased characters are in uppercase and false otherwise.

**Syntax:**

s.isupper()

**Program:**

```
string = "Hello World"
print (string.isupper() )
```

**Output:**

False

## String Module

String module is a python script file, which contains several number of related functions to strings that script is used as module without its extension(.py) in other python program. This is called string module

## Program:

```
import string   #importing srting module
text = "Monty Python's Flying Circus"
print ("upper", "=>", string.upper(text))
print ("lower", "=>", string.lower(text))
print ("split", "=>", string.split(text))
print ("join", "=>", string.join(string.split(text), "+"))
print ("replace", "=>", string.replace(text, "Python", "Java"))
print ("find", "=>", string.find(text, "Python"), string.find(text, "Java"))
Print("count", "=>", string.count(text, "n"))
```

**Output:**

```
upper => MONTY PYTHON'S FLYING CIRCUS
lower => monty python's flying circus
split => ['Monty', "Python's", 'Flying', 'Circus']
join => Monty+Python's+Flying+Circus
replace => Monty Java's Flying Circus
find => 6 -1
count => 3
```

## #1: GCD of Two Numbers:

```
defgcd(a,b):

if(b==0):

return a

else:

returngcd(b,a%b)

a=int(input("Enter first number:"))

b=int(input("Enter second number:"))

GCD=gcd(a,b)

print("GCD is: ")

print(GCD)
```

**Output:**

Case 1:

Enter first number:5

Enter second number:15

GCD is:

5

Case 2:

Enter first number:30

Enter second number:12

GCD is:

6

## #2. Exponentiation of a number:

```
def power(base,exp):
if(exp==1):
return(base)
if(exp!=1):
return(base*power(base,exp-1))
base=int(input("Enter base: "))
exp=int(input("Enter exponential value: "))
print("Result:",power(base,exp))
```

**Output:**

Enter base: 2

Enter exponential value: 5

Result: 32

Enter base: 5

Enter exponential value: 3

Result: 125

# WORKING WITH ARRAYS IN PYTHON

## AIM:

To implement arrays using Python

## Program

```
# Python code to demonstrate the working of

# array(), append(), insert()

# importing "array" for array operations

Import array

# initializing array with array values

# initializes array with signed integers

arr = array.array('i', [1, 2, 3])

# printing original array

print ("The new created array is : ",end=" ")

for i in range (0, 3):

    print (arr[i], end=" ")

  print("\r")

  # using append() to insert new value at end

arr.append(4);

# printing appended array

print("The appended array is : ", end="")

for i in range (0, 4):

    print (arr[i], end=" ")

 # using insert() to insert value at specific position

# inserts 5 at 2nd position

arr.insert(2, 5)


print("\r")

 # printing array after insertion

print ("The array after insertion is : ", end="")
```

```
for i in range (0, 5):
    print (arr[i], end=" ")
```

**Output:**

The new created array is : 1 2 3

The appended array is : 1 2 3 4

The array after insertion is : 1 2 5 3 4

## Pop and Remove in arrays

```
# Python code to demonstrate the working of
# pop() and remove()
# importing "array" for array operations
import array
 # initializing array with array values
# initializes array with signed integers
arr= array.array('i',[1, 2, 3, 1, 5])
# printing original array
print ("The new created array is : ",end="")
for i in range (0,5):
    print (arr[i],end=" ")
  print ("\r")
# using pop() to remove element at 2nd position
print ("The popped element is : ",end="")
print (arr.pop(2));
# printing array after popping
print ("The array after popping is : ",end="")
for i in range (0,4):
    print (arr[i],end=" ")
    print("\r")
```

```
# using remove() to remove 1st occurrence of 1
arr.remove(1)
# printing array after removing
print ("The array after removing is : ",end="")
for i in range (0,3):
    print (arr[i],end=" ")
```

**Output:**

```
The new created array is : 1 2 3 1 5
The popped element is : 3
The array after popping is : 1 2 1 5
The array after removing is : 2 1 5
```

## Index and reverse

```
# Python code to demonstrate the working of
# index() and reverse()
# importing "array" for array operations
import array
 # initializing array with array values
# initializes array with signed integers
arr= array.array('i',[1, 2, 3, 1, 2, 5])
# printing original array
print ("The new created array is : ",end="")
for i in range (0,6):
    print (arr[i],end=" ")
  print ("\r")
# using index() to print index of 1st occurrenece of 2
print ("The index of 1st occurrence of 2 is : ",end="")
print (arr.index(2))
#using reverse() to reverse the array
```

```
arr.reverse()
# printing array after reversing
print ("The array after reversing is : ",end="")
for i in range (0,6):
    print (arr[i],end=" ")
```

**Output:**

The new created array is : 1 2 3 1 2 5

The index of 1st occurrence of 2 is : 1

The array after reversing is : 5 2 1 3 2 1

## USER DEFINED FUNCTION IN PYTHON

### AIM:

To implement user defined functions in python

### Program

Program to illustrate

```
# the use of user-defined functions
def add_numbers(x,y):
   sum = x + y
   return sum
num1 = 5
num2 = 6
print("The sum is", add_numbers(num1, num2))
```

**Output**

Enter a number: 2.4

Enter another number: 6.5

The sum is 8.9

# BUILDING REGISTRATION SYSTEM USING MYSQL AND PHP

## AIM:

To implement a simple application using PHP and Mysql

## Procedure

## Step 1: Creating the Database Table

Execute the following SQL query to create the users table inside your MySQL database.

```
CREATE TABLE users (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

## Step 2: Creating the Config File

After creating the table, we need create a PHP script in order to connect to the MySQL database server. Let's create a file named "config.php" and put the following code inside it.

```
?php
/* Database credentials. Assuming you are running MySQL
server with default setting (user 'root' with no password) */
define('DB_SERVER', 'localhost');
define('DB_USERNAME', 'root');
define('DB_PASSWORD', '');
define('DB_NAME', 'demo');

/* Attempt to connect to MySQL database */
$link = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD, DB_NAME);
```

```
// Check connection
if($link === false){
    die("ERROR: Could not connect. " . mysqli_connect_error());
}
?>
```

## Step 3: Creating the Registration Form

Let's create another PHP file "register.php" and put the following example code in it. This example code will create a web form that allows user to register them. This script will also generate errors if a user tries to submit the form without entering any value, or if username entered by the user is already taken by another user.

```
<?php
// Include config file
require_once "config.php";

// Define variables and initialize with empty values
$username = $password = $confirm_password = "";
$username_err = $password_err = $confirm_password_err = "";

// Processing form data when form is submitted
if($_SERVER["REQUEST_METHOD"] == "POST"){

    // Validate username
    if(empty(trim($_POST["username"]))){
        $username_err = "Please enter a username.";
    } else{
        // Prepare a select statement
        $sql = "SELECT id FROM users WHERE username = ?";

        if($stmt = mysqli_prepare($link, $sql)){
```

```
        // Bind variables to the prepared statement as parameters
        mysqli_stmt_bind_param($stmt, "s", $param_username);

        // Set parameters
        $param_username = trim($_POST["username"]);

        // Attempt to execute the prepared statement
        if(mysqli_stmt_execute($stmt)){
            /* store result */
            mysqli_stmt_store_result($stmt);

            if(mysqli_stmt_num_rows($stmt) == 1){
                $username_err = "This username is already taken.";
            } else{
                $username = trim($_POST["username"]);
            }
        } else{
            echo "Oops! Something went wrong. Please try again later.";
        }
    }

    // Close statement
    mysqli_stmt_close($stmt);
}

// Validate password
if(empty(trim($_POST["password"]))){
    $password_err = "Please enter a password.";
} elseif(strlen(trim($_POST["password"])) < 6){
```

```
    $password_err = "Password must have atleast 6 characters.";
  } else{
    $password = trim($_POST["password"]);
  }

  // Validate confirm password
  if(empty(trim($_POST["confirm_password"]))){
    $confirm_password_err = "Please confirm password.";
  } else{
    $confirm_password = trim($_POST["confirm_password"]);
    if(empty($password_err) && ($password != $confirm_password)){
      $confirm_password_err = "Password did not match.";
    }
  }

  // Check input errors before inserting in database
  if(empty($username_err) && empty($password_err) &&
empty($confirm_password_err)){

    // Prepare an insert statement
    $sql = "INSERT INTO users (username, password) VALUES (?, ?)";

    if($stmt = mysqli_prepare($link, $sql)){
      // Bind variables to the prepared statement as parameters
      mysqli_stmt_bind_param($stmt, "ss", $param_username,
$param_password);

      // Set parameters
      $param_username = $username;
```

*Notes*

```
            $param_password = password_hash($password, PASSWORD_DEFAULT); // Creates a password hash

            // Attempt to execute the prepared statement
            if(mysqli_stmt_execute($stmt)){
                // Redirect to login page
                header("location: login.php");
            } else{
                echo "Something went wrong. Please try again later.";
            }
        }

        // Close statement
        mysqli_stmt_close($stmt);
    }

    // Close connection
    mysqli_close($link);
}
?>


<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Sign Up</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.css">
    <style type="text/css">
```

Notes

```
        body{ font: 14px sans-serif; }
        .wrapper{ width: 350px; padding: 20px; }
    </style>
</head>
<body>
    <div class="wrapper">
        <h2>Sign Up</h2>
        <p>Please fill this form to create an account.</p>
<form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">
            <div class="form-group <?php echo (!empty($username_err)) ? 'has-error' : ''; ?>">
                <label>Username</label>
                <input type="text" name="username" class="form-control" value="<?php echo $username; ?>">
                <span class="help-block"><?php echo $username_err; ?></span>
            </div>
            <div class="form-group <?php echo (!empty($password_err)) ? 'has-error' : ''; ?>">
                <label>Password</label>
                <input type="password" name="password" class="form-control" value="<?php echo $password; ?>">
                <span class="help-block"><?php echo $password_err; ?></span>
            </div>
            <div class="form-group <?php echo (!empty($confirm_password_err)) ? 'has-error' : ''; ?>">
                <label>Confirm Password</label>
                <input type="password" name="confirm_password" class="form-control" value="<?php echo $confirm_password; ?>">
```

```
            <span class="help-block"><?php echo $confirm_password_err; ?></span>
        </div>
        <div class="form-group">
            <input type="submit" class="btn btn-primary" value="Submit">
            <input type="reset" class="btn btn-default" value="Reset">
        </div>
        <p>Already have an account? <a href="login.php">Login here</a>.</p>
    </form>
</div>
</body>
</html>
```

## BUILDING THE LOGIN SYSTEM

In this section we'll create a login form where user can enter their username and password. When user submit the form these inputs will be verified against the credentials stored in the database, if the username and password match, the user is authorized and granted access to the site, otherwise the login attempt will be rejected.

### Step 1: Creating the Login Form

Let's create a file named "login.php" and place the following code inside it.

```
<?php
// Initialize the session
session_start();


// Check if the user is already logged in, if yes then redirect him to welcome page
if(isset($_SESSION["loggedin"]) && $_SESSION["loggedin"] === true){
    header("location: welcome.php");
    exit;
}
```

```
// Include config file
require_once "config.php";

// Define variables and initialize with empty values
$username = $password = "";
$username_err = $password_err = "";

// Processing form data when form is submitted
if($_SERVER["REQUEST_METHOD"] == "POST"){

    // Check if username is empty
    if(empty(trim($_POST["username"]))){
        $username_err = "Please enter username.";
    } else{
        $username = trim($_POST["username"]);
    }

    // Check if password is empty
    if(empty(trim($_POST["password"]))){
        $password_err = "Please enter your password.";
    } else{
        $password = trim($_POST["password"]);
    }

    // Validate credentials
    if(empty($username_err) && empty($password_err)){
        // Prepare a select statement
```

```
$sql = "SELECT id, username, password FROM users WHERE username = ?";

if($stmt = mysqli_prepare($link, $sql)){
    // Bind variables to the prepared statement as parameters
    mysqli_stmt_bind_param($stmt, "s", $param_username);

    // Set parameters
    $param_username = $username;

    // Attempt to execute the prepared statement
    if(mysqli_stmt_execute($stmt)){
        // Store result
        mysqli_stmt_store_result($stmt);

        // Check if username exists, if yes then verify password
        if(mysqli_stmt_num_rows($stmt) == 1){
            // Bind result variables
            mysqli_stmt_bind_result($stmt, $id, $username, $hashed_password);
            if(mysqli_stmt_fetch($stmt)){
                if(password_verify($password, $hashed_password)){
                    // Password is correct, so start a new session
                    session_start();

                    // Store data in session variables
                    $_SESSION["loggedin"] = true;
                    $_SESSION["id"] = $id;
                    $_SESSION["username"] = $username;
```

```
                    // Redirect user to welcome page
                    header("location: welcome.php");
                } else{
                    // Display an error message if password is not valid
                    $password_err = "The password you entered was not valid.";
                }
            }
        } else{
            // Display an error message if username doesn't exist
            $username_err = "No account found with that username.";
        }
    } else{
        echo "Oops! Something went wrong. Please try again later.";
    }
}

    // Close statement
    mysqli_stmt_close($stmt);
}

// Close connection
mysqli_close($link);
}
?>

<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.css">
    <style type="text/css">
        body{ font: 14px sans-serif; }
        .wrapper{ width: 350px; padding: 20px; }
    </style>
</head>
<body>
    <div class="wrapper">
        <h2>Login</h2>
        <p>Please fill in your credentials to login.</p>
        <form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">
            <div class="form-group <?php echo (!empty($username_err)) ? 'has-error' : ''; ?>">
                <label>Username</label>
                <input type="text" name="username" class="form-control" value="<?php echo $username; ?>">
                <span class="help-block"><?php echo $username_err; ?></span>
            </div>
            <div class="form-group <?php echo (!empty($password_err)) ? 'has-error' : ''; ?>">
                <label>Password</label>
                <input type="password" name="password" class="form-control">
                <span class="help-block"><?php echo $password_err; ?></span>
```

```
        </div>
        <div class="form-group">
            <input type="submit" class="btn btn-primary" value="Login">
        </div>
        <p>Don't have an account? <a href="register.php">Sign up now</a>.</p>
    </form>
</div>
</body>
</html>
```

## Step 2: Creating the Welcome Page

Here's the code of our "welcome.php" file, where user is redirected after successful login.

```
<?php
// Initialize the session
session_start();

// Check if the user is logged in, if not then redirect him to login page
if(!isset($_SESSION["loggedin"]) || $_SESSION["loggedin"] !== true){
    header("location: login.php");
    exit;
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Welcome</title>
```

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.css">
<style type="text/css">
body{ font: 14px sans-serif; text-align: center; }
</style>
</head>
<body>
<div class="page-header">
<h1>Hi, <b><?php echo htmlspecialchars($_SESSION["username"]); ?></b>. Welcome to our site.</h1>
</div>
<p>
<a href="reset-password.php" class="btn btn-warning">Reset Your Password</a>
<a href="logout.php" class="btn btn-danger">Sign Out of Your Account</a>
</p>
</body>
</html>
```

If data comes from external sources like form filled in by anonymous users, there is a risk that it may contain malicious script indented to launch cross-site scripting (XSS) attacks. Therefore, you must escape this data using the PHP htmlspecialchars() function before displaying it in the browser, so that any HTML tag it contains becomes harmless.

For example, after escaping special characters the string <script>alert("XSS")</script> becomes <script>alert("XSS")</script> which is not executed by the browser.

## Step 3: Creating the Logout Script

Now, let's create a "logout.php" file. When the user clicks on the log out or sign out link, the script inside this file destroys the session and redirect the user back to the login page.

```
<?php
// Initialize the session
session_start();

// Unset all of the session variables
$_SESSION = array();

// Destroy the session.
session_destroy();

// Redirect to login page
header("location: login.php");
exit;
?>
```

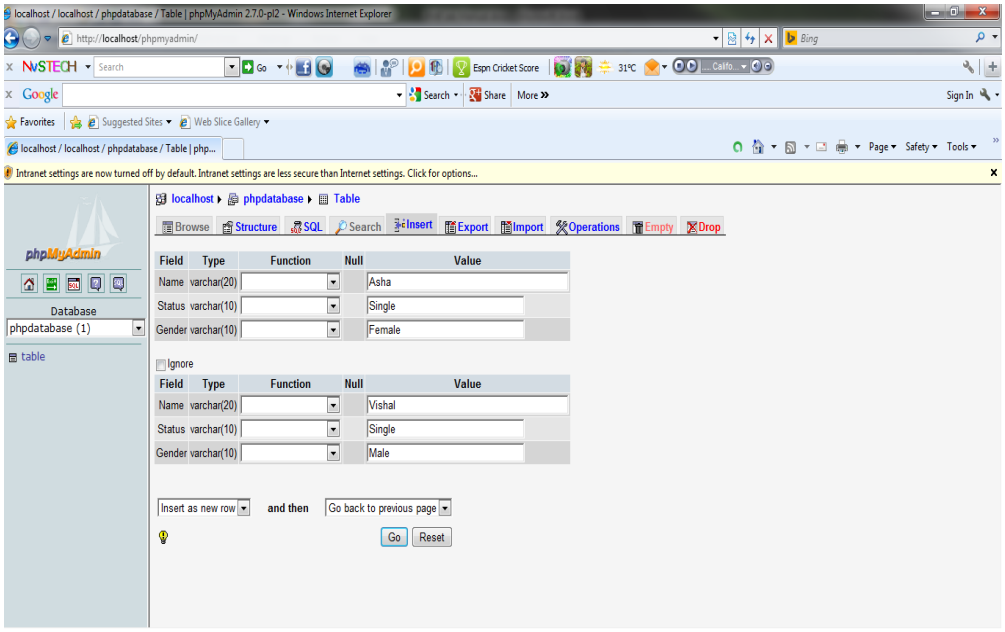
## CONNECTING MYSQL DATABASE WITH PHP

### AIM:

To implement and learn by connecting mysql database with PHP

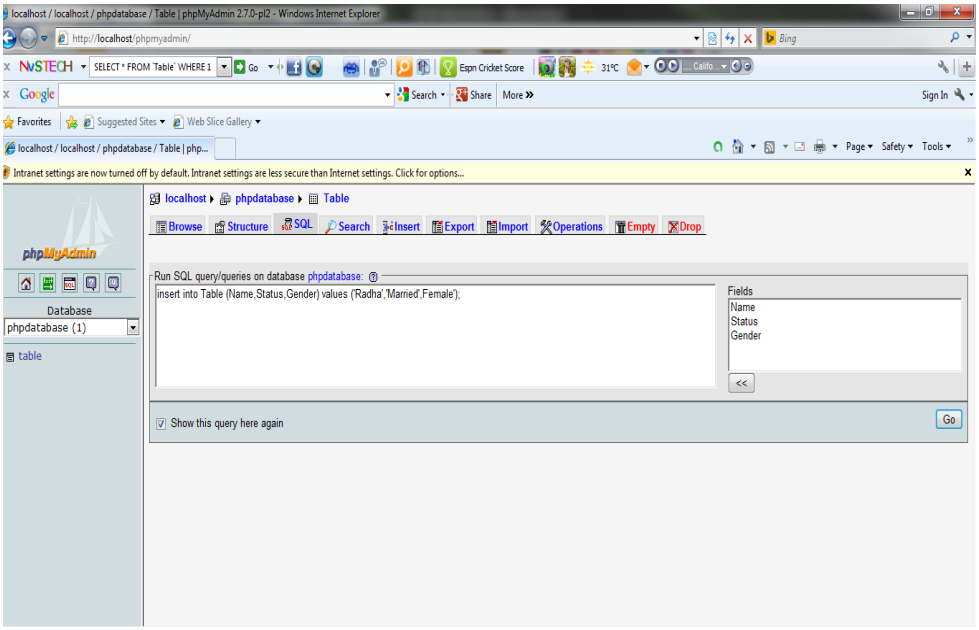
### Procedure

- To connect MYSQL using PHP go to: http://localhost//phpmyadmin
- Enter the username and password
- Give the database name in the field 'create new database'
- Click on create button
- Create a new table in the database by giving a table name and number of fields then click on Go
- To give field name to the created table, write the field name in the 'field' column,
- select the data types for each fields, specify the length of each field then click on
- save to save the fields and click on Go
- To insert values in the field, go to insert and enter the values. Then click on Go




To view the created table, go to browse

To insert the values, go to SQL and write the query to insert the values and click on Go




SQL query for insert:

## Syntax:

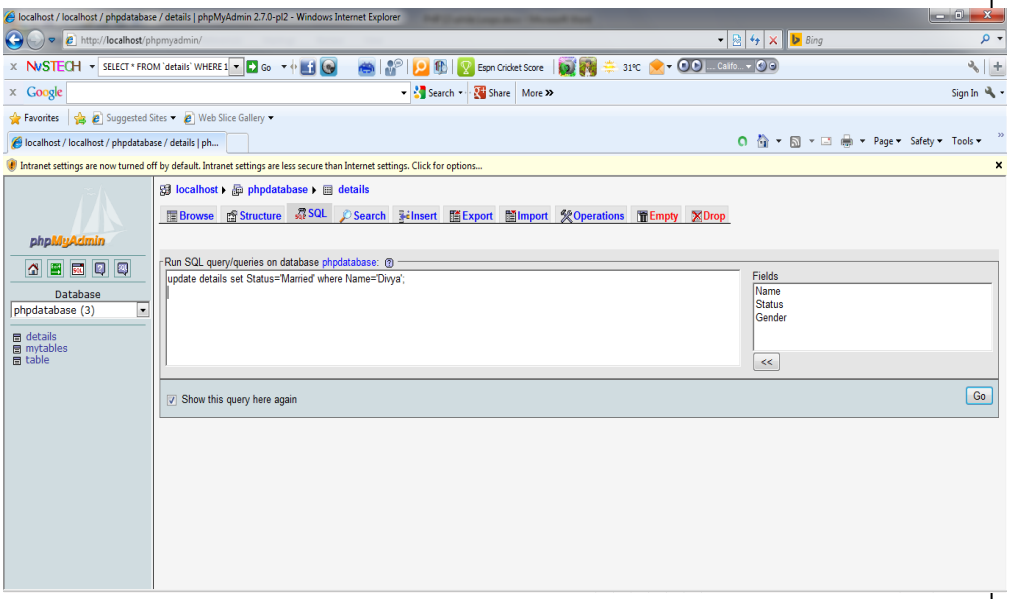
Insert into table_name values('value1','value2',…);

## Example:

Insert into Login values('Radha','hello');

To update the values, go to SQL and write the query to update the values and

click on Go




SQL query for update:

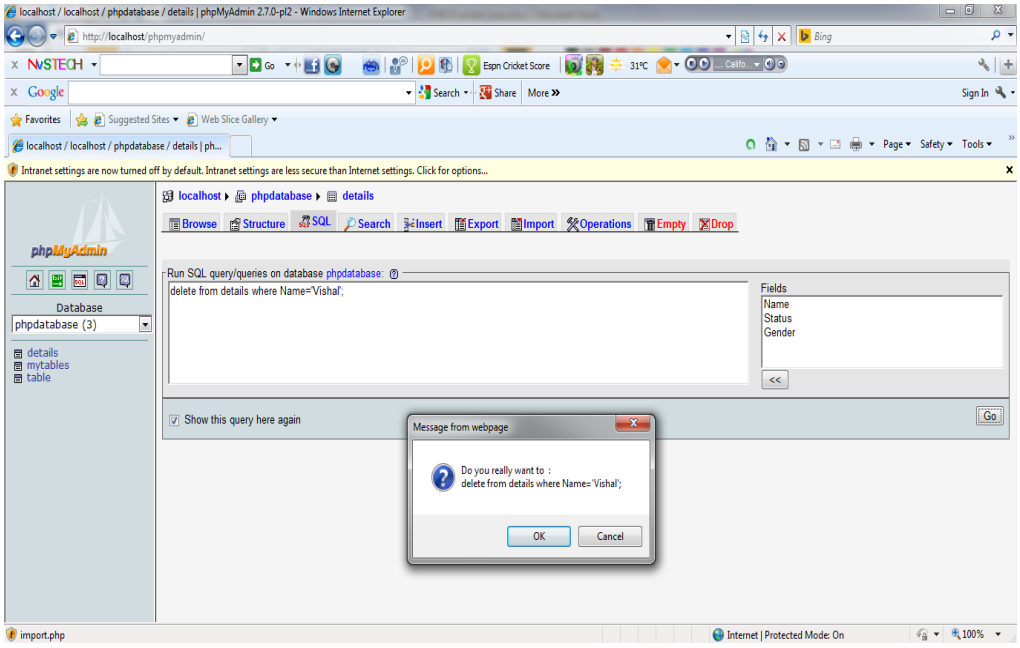
Syntax:

Update table_name set field_name='value' where field_name='value';

Example:

Update Login set password='abcde' where name='Radha'

To delete the values, go to SQL and write the query to delete the values and click




**SQL query for delete:**

## Syntax:

Delete from table_name where field_name=’value’;

## Example:

Delete from Login where name=’Radha’;

## THE FUNCTIONS USED TO CONNECT WEB FORM TO THE MYSQL DATABASE:

### mysql_connect():

This function opens a link to a MySQL server on the specified host (in this case it's localhost) along with a username (root) and password (q1w2e3r4/). The result of the connection is stored in the variable $db.

### mysql_select_db():

This tells PHP that any queries we make are against the mydb database.

### mysql_query():

Using the database connection identifier, it sends a line of SQL to the MySQL server to be processed. The results that are returned are stored in the variable $result.

Notes

## mysql_result():

This is used to display the values of fields from our query. Using $result, we go to the first row, which is numbered 0, and display the value of the specified fields.

mysql_result($result,0,"position")):

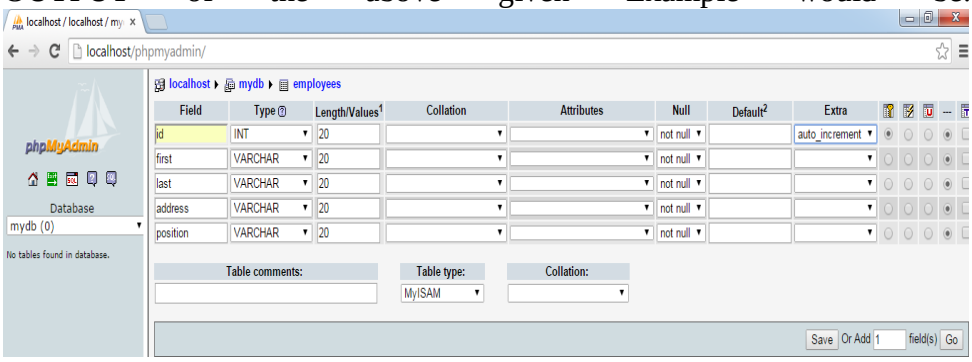
This should be treated as a string and printed.

## Display the data from MYSQL database in web form

```
<html>
<body>
<?php
// Open MYSQL server connection
$db = mysql_connect("localhost", "root","q1w2e3r4/");
// Select the database using MYSQL server connection
mysql_select_db("mydb",$db);
/* Using the database connection identifier, it sends
a line of SQL to the MySQL server to be processed
and the results are stored in the variable
$result. */
$result = mysql_query("SELECT * FROM employees",$db);
// Displaying the details in a table
echo "<table border=1>";
echo "<tr><th>Name</th><th>Position</th></tr>";
while ($myrow = mysql_fetch_row($result)) {
printf("<tr><td>%s %s</td><td>%s</td></tr>",
$myrow[1], $myrow[2],$myrow[4]);
}
echo "</table>";
?>
</body>
```

</html>

OUTPUT of the above given Example would be:




Insert the data into MYSQL database using web form

<html>

<body>

<?php

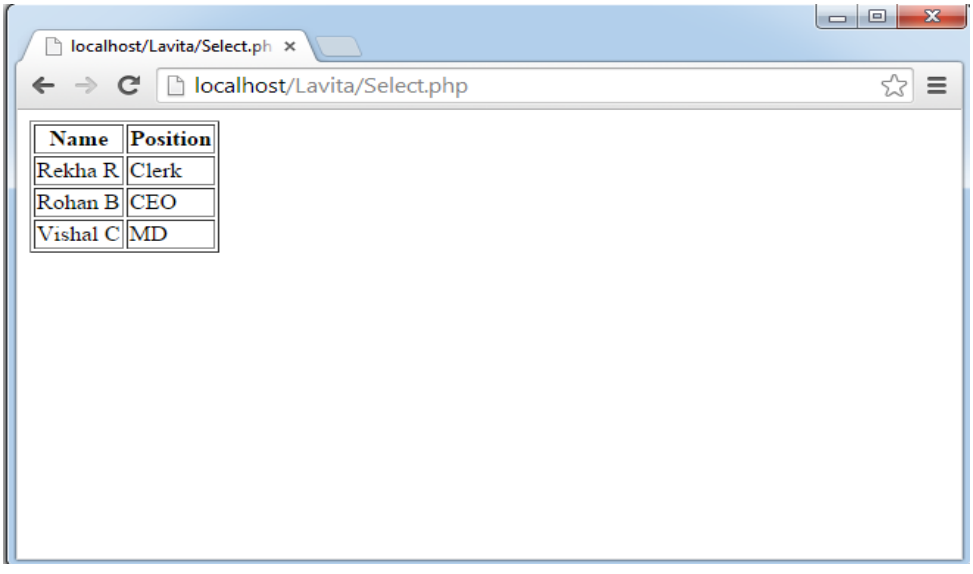
if ($submit) {

// Open MYSQL server connection

$db = mysql_connect("localhost", "root","q1w2e3r4/");

// Select the database using MYSQL server connection

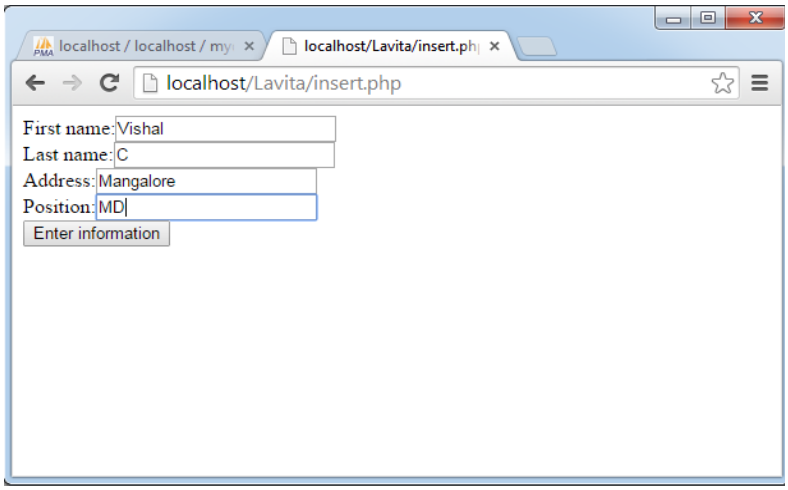
mysql_select_db("mydb",$db);

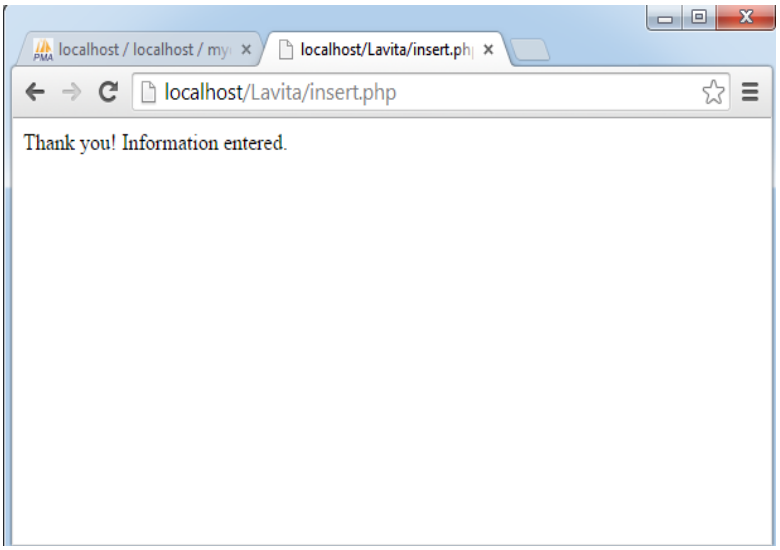


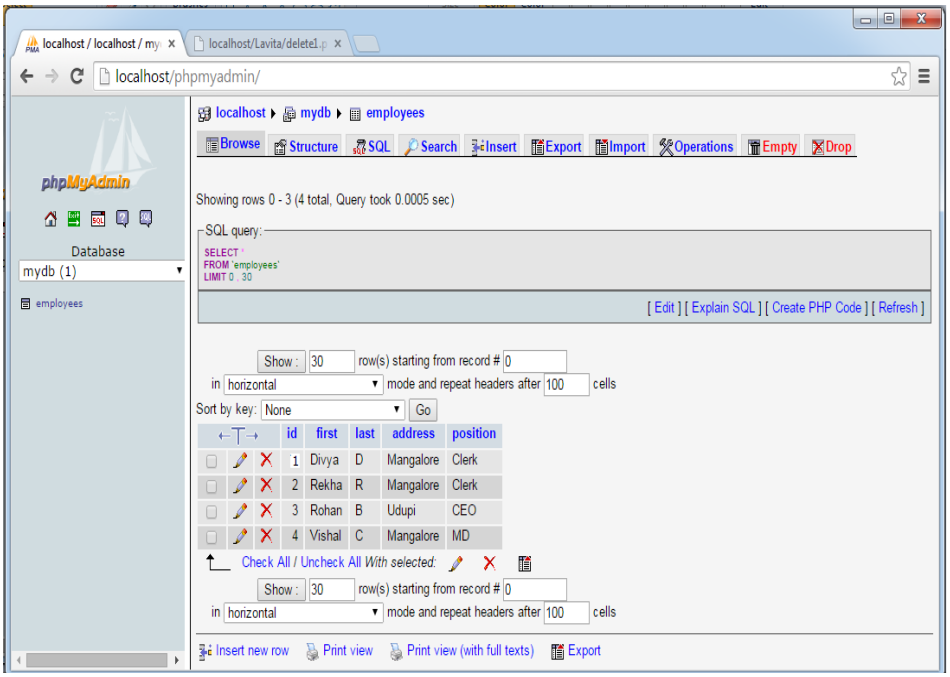

/* Write insert query and assign the query in $sql

Variable */

```
$sql = "INSERT INTO employees (first,last,address,position)
VALUES('$first','$last','$address','$position')";
// Execute the query
$result = mysql_query($sql);
echo "Thank you! Information entered.";
}
else
{
// display form
?>
<form method="post" action="<?php echo $PHP_SELF?>">
First name:<input type="Text" name="first"><br>
Last name:<input type="Text" name="last"><br>
Address:<input type="Text" name="address"><br>
Position:<input type="Text" name="position"><br>
<input type="Submit" name="submit" value="Enter
information">
</form>
<?php
} // end if
?>
</body>
</html>
```


localhost / localhost / my
localhost/Lavita/insert.ph
localhost/Lavita/insert.php
First name: Vishal
Last name: C
Address: Mangalore
Position: MD
Enter information


localhost / localhost / my
localhost/Lavita/insert.ph
localhost/Lavita/insert.php
Thank you! Information entered.




## Update the data present in MYSQL

database using web form

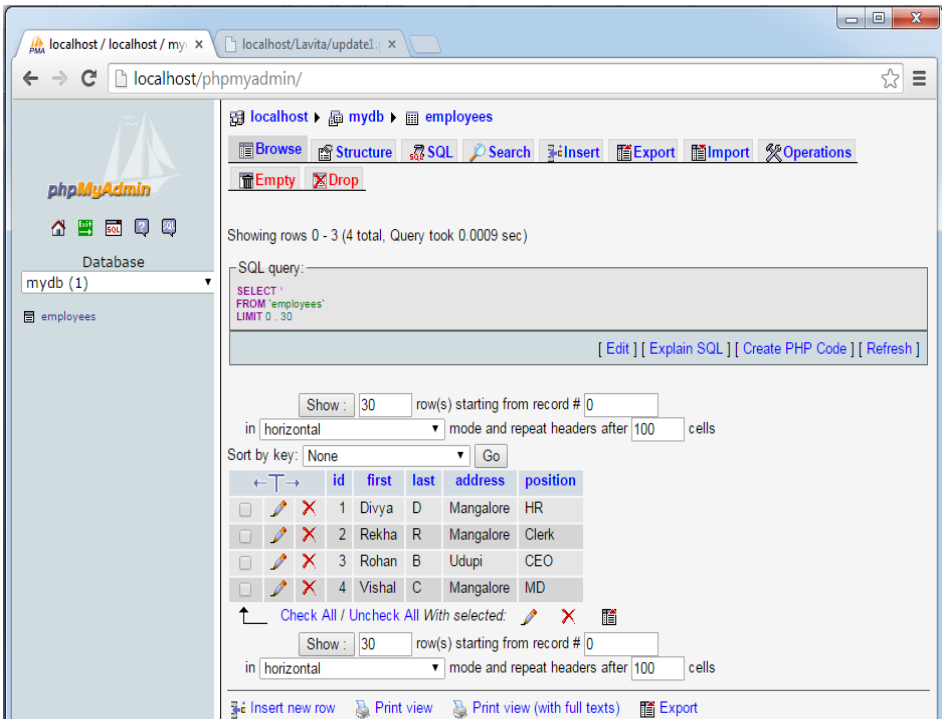
```
<html>
<body>
<?php
// Open MYSQL server connection
$db = mysql_connect("localhost", "root","q1w2e3r4/");
// Select the database using MYSQL server connection
mysql_select_db("mydb",$db);
if ($id) {
if ($submit) {
// Write UPDATE query and assign to $sql Variable
$sql = "UPDATE employees SET
first='$first', last='$last',
address='$address',
position='$position'
WHERE id=$id";
```

```
// Execute the query
$result = mysql_query($sql);
echo "Thank you! Information updated.";
}
else
{
// Write query to SELECT data from table
$sql = "SELECT * FROM employees WHERE id=$id";
// Execute the query
$result = mysql_query($sql);
// Fetch the values
$myrow = mysql_fetch_array($result);
?>
<form method="post" action="<?php echo $PHP_SELF?>">
<input type=hidden name="id" value="<?php echo
$myrow["id"] ?>">
First name:<input type="Text" name="first"
value="<?php echo $myrow["first"] ?>"><br>
Last name:<input type="Text" name="last"
value="<?php echo $myrow["last"] ?>"><br>
Address:<input type="Text" name="address"
value="<?php echo $myrow["address"]?>"><br>
Position:<input type="Text" name="position"
value="<?php echo $myrow["position"]?>"><br>
<input type="Submit" name="submit" value="Enter
information">
</form>
```

Integrating PHP with Embedded System

www.researchdesignlab.com Page 67

```
<?php
}
}
else
{
// display list of employees
$result = mysql_query("SELECT * FROM
employees",$db);
while ($myrow = mysql_fetch_array($result)) {
printf("<a href=\"%s?id=%s\">%s %s</a><br>",
$PHP_SELF, $myrow["id"],$myrow["first"],
$myrow["last"]);
}
}
?>
</body>
</html>
```

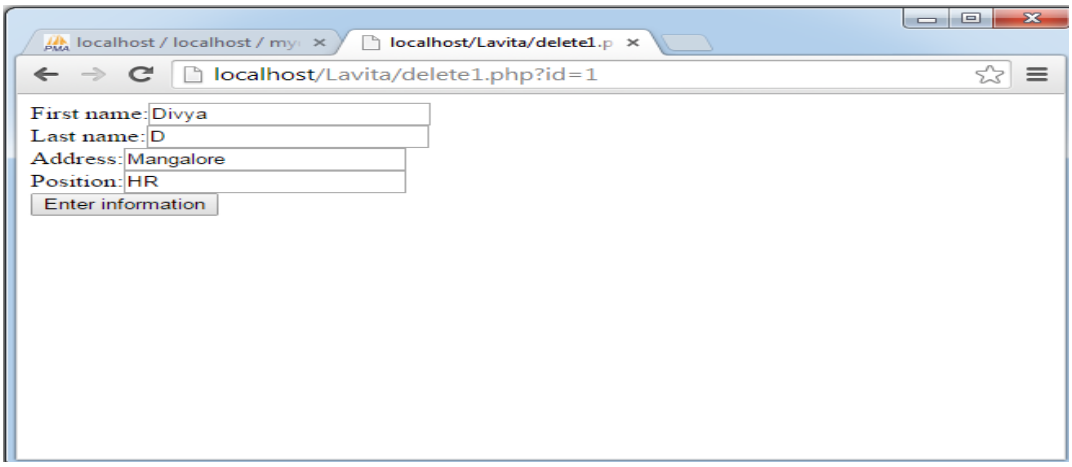



**Delete the data from MYSQL database using web form**

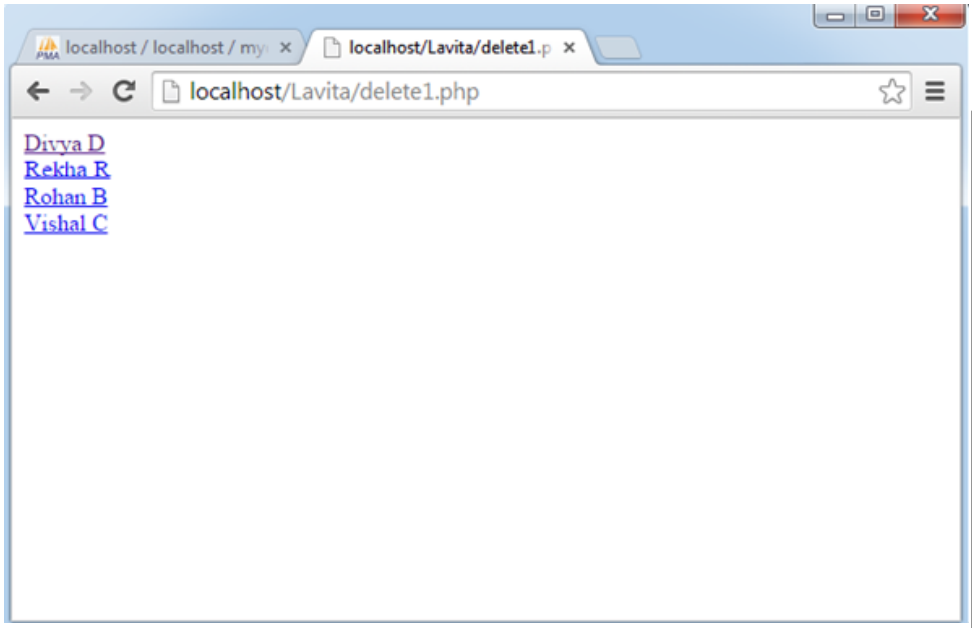
```
<html>
<body>
<?php
// Open MYSQL server connection
$db = mysql_connect("localhost", "root","q1w2e3r4/");
// Select the database using MYSQL server connection
mysql_select_db("mydb",$db);
if ($id) {
if ($submit) {
// Write DELETE query to delete data from table based on ID
$sql = "DELETE FROM employees WHERE id=$id";
// Execute the query
$result = mysql_query($sql);
echo "Thank you! Information deleted.";
}
else
```

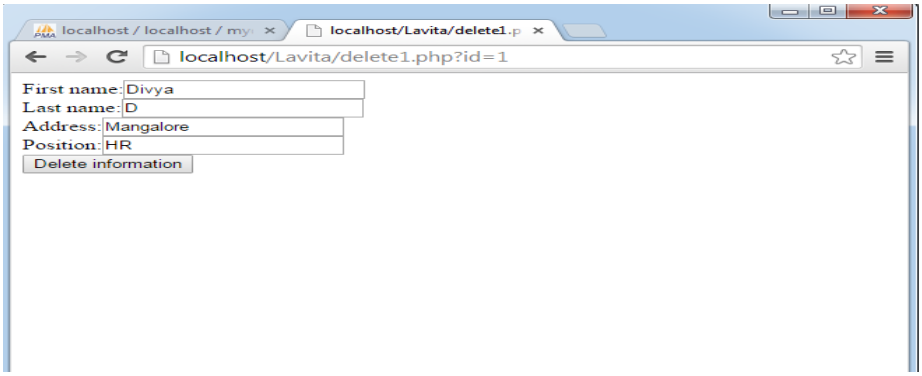
```
{
// Write SELECT query to select data from table based on ID
$sql = "SELECT * FROM employees WHERE id=$id";
$result = mysql_query($sql);
$myrow = mysql_fetch_array($result);
?>
<form method="post" action="<?php echo $PHP_SELF?>">
<input type=hidden name="id"
value="<?php echo $myrow["id"] ?>">
First name:<input type="Text" name="first"
readonly="readonly"
value="<?php echo $myrow["first"] ?>"><br>
Last name:<input type="Text" name="last"
readonly="readonly"
value="<?php echo $myrow["last"] ?>"><br>
Address:<input type="Text" name="address"
readonly="readonly"
value="<?php echo $myrow["address"]?>"><br>
Position:<input type="Text" name="position"
value="<?php echo $myrow["position"]?>"><br>
<input type="Submit" name="submit"
value="Delete information">
</form>
<?php
}
}
else
{
```

```
// display list of employees
$result = mysql_query("SELECT * FROM
employees",$db);
while ($myrow = mysql_fetch_array($result)) {
printf("<a href=\"%s?id=%s\">%s %s</a><br>",
$PHP_SELF, $myrow["id"],$myrow["first"],
$myrow["last"]);
}
}
?>
</body>
</html>
```




OUTPUT of the above given Example would be:




## CONNECTING MYSQL WITH PERL

### AIM:

To implement and learn in connecting to mysqlwith Perl

### Procedure

### Connecting to MySQL database

Perl MySQL ConnectWhen you connect to a MySQL database, you need to specify the following information:

First, you need to tell DBI where to find the MySQL database server. This information is called data source name or DSN. The data source name specifies which driver to use, what database that you want to connect to. Perl requires the data source name to begin with dbi: and the name of the driver, in this case, it is mysql , followed by another colon : e.g., dbi:mysql: , and then the database name e.g., dbi:mysql:perlmysqldb .

Second, you need to provide the username and password of the MySQL account that you connect to the database.

Third, the optional connection attributes specify the way DBI handles exceptions that may occur when it connects to the MySQL database.

The syntax for creating a connection to the MySQL database is as follows:

```
$dbh = DBI->connect($dsn,$username,$password,\%attr);
```

The connect() method returns a database handle if the connection to the database established successfully. For example to connect to the perlmysqldb , you use the following script:

```
#!/usr/bin/perl

use strict;

use warnings;

use v5.10; # for say() function

 use DBI;

say "Perl MySQL Connect Demo";

# MySQL database configuration

my $dsn = "DBI:mysql:perlmysqldb";

my $username = "root";

my $password = '';

 # connect to MySQL database

my %attr = ( PrintError=>0,  # turn off error reporting via warn()

            RaiseError=>1};   # turn on error reporting via die()

 my $dbh  = DBI->connect($dsn,$username,$password, \%attr);

 say "Connected to the MySQL database.";
```

## Working of commands

First, to use DBI module, we put the use DBI; statement at the top of the script.

Next, we defined some variables that hold the data source name, username and password.

Then, we defined a hash that contains connection's attributes. Those connection attributes will be discussed in the error handling section later.

After that, we passed the corresponding arguments to the connect() method to create a connection to the perlmysqdb database.

Finally, we displayed a message to indicate that the script has been connected to the MySQL database successfully. The following is the output of the script:

Perl MySQL Connect Demo

Connected to the MySQL database.

## Handling errors

Perl DBI allows you to handle error manually and/or automatically. Perl DBI detects the error when it occurs and calls either warn() or die() function with an appropriate error message. The PrintError attribute instructs DBI to call the warn() function that prints the errors to the screen. The RaiseError attribute tells DBI to call the die() function upon error and to abort the script immediately.

Perl DBI enables the PrintError by default. However, we strongly recommend that you turn the PrintError attribute off and RaiseError attribute on to instruct DBI to handle the error automatically.If you don't turn the RaiseError on, you have to handle the error manually as follows:

```
# withou RasieError off:

 my $dbh  = DBI->connect($dsn,$username,$password) or

  die("Error connecting to the database: $DBI::errstr\n");
```

When an error occurred, DBI stored the error message in the $DBI::errstr variable. Basically, the above statement means if the connection to the database failed, it displays the error message and aborts the script immediately.Another benefit of turning on the RaiseError attribute is that the code will look more readable because you don't have to include the or die() statement everywhere you call a DBI method.

## Disconnecting from MySQL Database

If you are no longer interacting with the database, you should explicitly disconnect from it. This is a good programming practice.To disconnect from a database, you use the disconect() method of the database handle object as follows:

```
# disconnect from the MySQL database

$dbh->disconnect();
```

# CONNECTING MYSQL WITH PYTHON

## AIM:

To implement and learn Mysql with Python

## Procedure

Python Database Interfaces and APIs. You must download a separate DB API module for each database you need to access. For example, if you need to access an Oracle database as well as a MySQL database, you must download both the Oracle and the MySQL database modules.

The DB API provides a minimal standard for working with databases using Python structures and syntax wherever possible. This API includes the following −

- Importing the API module.
- Acquiring a connection with the database.
- Issuing SQL statements and stored procedures.
- Closing the connection

## MySQLdb

MySQLdb is an interface for connecting to a MySQL database server from Python. It implements the Python Database API v2.0 and is built on top of the MySQL C API.

## Install MySQLdb

Before proceeding, you make sure you have MySQLdb installed on your machine. Just type the following in your Python script and execute it

```
#!/usr/bin/python
import MySQLdb
```

If it produces the following result, then it means MySQLdb module is not installed Traceback (most recent call last):

```
 File "test.py", line 3, in <module>

  import MySQLdb

ImportError: No module named MySQLdb
```

To install MySQLdb module, use the following command −

For Python command prompt, use the following command -

pip install MySQL-python

## Database Connection



Before connecting to a MySQL database, make sure of the followings −

You have created a database TESTDB.

You have created a table EMPLOYEE in TESTDB.

This table has fields FIRST_NAME, LAST_NAME, AGE, SEX and INCOME.

User ID "testuser" and password "test123" are set to access TESTDB.

Python module MySQLdb is installed properly on your machine.

You have gone through MySQL tutorial to understand MySQL Basics.

## Example:

Following is the example of connecting with MySQL database "TESTDB"

```
#!/usr/bin/python

import MySQLdb

# Open database connection

db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method

cursor = db.cursor()

# execute SQL query using execute() method.

cursor.execute("SELECT VERSION()")

# Fetch a single row using fetchone() method.

data = cursor.fetchone()

print "Database version : %s " % data

# disconnect from server

db.close()
```

While running this script, it is producing the following result in my Linux machine.

## Database version : 5.0.45

If a connection is established with the datasource, then a Connection Object is returned and saved into db for further use, otherwise db is set to None. Next, db object is used to create a cursor object, which in turn is used to execute SQL queries. Finally, before coming out, it ensures that database connection is closed and resources are released.

## Creating Database Table

Once a database connection is established, we are ready to create tables or records into the database tables using execute method of the created cursor.

**Example:**

Let us create Database table EMPLOYEE −

```
#!/usr/bin/python

import MySQLdb

# Open database connection

db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method

cursor = db.cursor()

# Drop table if it already exist using execute() method.

cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")

# Create table as per requirement

sql = """CREATE TABLE EMPLOYEE (

        FIRST_NAME  CHAR(20) NOT NULL,

        LAST_NAME  CHAR(20),

        AGE INT,

        SEX CHAR(1),

        INCOME FLOAT )"""

cursor.execute(sql)

# disconnect from server

db.close()
```

## INSERT Operation

It is required when you want to create your records into a database table.

## Example

**The following** example, executes SQL INSERT statement to create a record into EMPLOYEE table −

```
#!/usr/bin/python

import MySQLdb

# Open database connection

db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = """INSERT INTO EMPLOYEE(FIRST_NAME,

      LAST_NAME, AGE, SEX, INCOME)

      VALUES ('Mac', 'Mohan', 20, 'M', 2000)"""

try:

  # Execute the SQL command

  cursor.execute(sql)

  # Commit your changes in the database

  db.commit()

except:

  # Rollback in case there is any error

  db.rollback()

# disconnect from server

db.close()
```

Above example can be written as follows to create SQL queries dynamically −

```
#!/usr/bin/python

import MySQLdb

# Open database connection

db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )
```

```
# prepare a cursor object using cursor() method
cursor = db.cursor()
# Prepare SQL query to INSERT a record into the database.
sql = "INSERT INTO EMPLOYEE(FIRST_NAME, \
      LAST_NAME, AGE, SEX, INCOME) \
      VALUES ('%s', '%s', '%d', '%c', '%d' )" % \
      ('Mac', 'Mohan', 20, 'M', 2000)
try:
   # Execute the SQL command
   cursor.execute(sql)
   # Commit your changes in the database
   db.commit()
except:
   # Rollback in case there is any error
   db.rollback()
# disconnect from server
db.close()
```

Example

Following code segment is another form of execution where you can pass parameters directly −

```
.....
user_id = "test123"
password = "password"
con.execute('insert into Login values("%s", "%s")' % \
           (user_id, password))
.....
```

## READ Operation

READ Operation on any database means to fetch some useful information from the database.Once our database connection is established, you are ready to make a query into this database. You can use either fetchone()

method to fetch single record or fetchall() method to fetech multiple values from a database table.

**fetchone()** − It fetches the next row of a query result set. A result set is an object that is returned when a cursor object is used to query a table.

**fetchall()** − It fetches all the rows in a result set. If some rows have already been extracted from the result set, then it retrieves the remaining rows from the result set.

**rowcount** − This is a read-only attribute and returns the number of rows that were affected by an execute() method.

## Example:

The following procedure queries all the records from EMPLOYEE table having salary more than 1000 −

```
#!/usr/bin/python

import MySQLdb

# Open database connection

db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method

cursor = db.cursor()

sql = "SELECT * FROM EMPLOYEE \

     WHERE INCOME > '%d'" % (1000)

try:

  # Execute the SQL command

  cursor.execute(sql)

  # Fetch all the rows in a list of lists.

  results = cursor.fetchall()

  for row in results:

    fname = row[0]

    lname = row[1]

    age = row[2]

    sex = row[3]

    income = row[4]
```

```
   # Now print fetched result
   print "fname=%s,lname=%s,age=%d,sex=%s,income=%d" % \
         (fname, lname, age, sex, income )
except:
  print "Error: unable to fecth data"
# disconnect from server
db.close()
```

This will produce the following result −

```
fname=Mac, lname=Mohan, age=20, sex=M, income=2000
```

## Update Operation

UPDATE Operation on any database means to update one or more records, which are already available in the database.The following procedure updates all the records having SEX as 'M'. Here, we increase AGE of all the males by one year.

## Example

```
#!/usr/bin/python
import MySQLdb
# Open database connection
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )
# prepare a cursor object using cursor() method
cursor = db.cursor()
# Prepare SQL query to UPDATE required records
sql = "UPDATE EMPLOYEE SET AGE = AGE + 1
                    WHERE SEX = '%c'" % ('M')
try:
  # Execute the SQL command
  cursor.execute(sql)
  # Commit your changes in the database
  db.commit()
except:
```

```
   # Rollback in case there is any error
   db.rollback()
# disconnect from server
db.close()
```

## DELETE Operation

DELETE operation is required when you want to delete some records from your database. Following is the procedure to delete all the records from EMPLOYEE where AGE is more than 20

## Example

```
#!/usr/bin/python
import MySQLdb
# Open database connection
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )
# prepare a cursor object using cursor() method
cursor = db.cursor()
# Prepare SQL query to DELETE required records
sql = "DELETE FROM EMPLOYEE WHERE AGE > '%d'" % (20)
try:
   # Execute the SQL command
   cursor.execute(sql)
   # Commit your changes in the database
   db.commit()
except:
   # Rollback in case there is any error
   db.rollback()
# disconnect from server
db.close()
```

## Performing Transactions

Transactions are a mechanism that ensures data consistency. Transactions have the following four properties −

**Atomicity** − Either a transaction completes or nothing happens at all.

**Consistency** − A transaction must start in a consistent state and leave the system in a consistent state.

**Isolation** − Intermediate results of a transaction are not visible outside the current transaction.

**Durability** − Once a transaction was committed, the effects are persistent, even after a system failure.

The Python DB API 2.0 provides two methods to either commit or rollback a transaction.

## Example

You already know how to implement transactions. Here is again similar example −

```
# Prepare SQL query to DELETE required records

sql = "DELETE FROM EMPLOYEE WHERE AGE > '%d'" % (20)

try:

   # Execute the SQL command

   cursor.execute(sql)

   # Commit your changes in the database

   db.commit()

except:

   # Rollback in case there is any error

   db.rollback()
```

## COMMIT Operation

Commit is the operation, which gives a green signal to database to finalize the changes, and after this operation, no change can be reverted back.Here is a simple example to call commit method.

```
db.commit()
```

## ROLLBACK Operation

If you are not satisfied with one or more of the changes and you want to revert back those changes completely, then use rollback() method. Here is a simple example to call rollback() method.

```
db.rollback()
```

## Disconnecting Database

To disconnect Database connection, use close() method.

```
db.close()
```

If the connection to a database is closed by the user with the close() method, any outstanding transactions are rolled back by the DB. However, instead of depending on any of DB lower level implementation details, your application would be better off calling commit or rollback explicitly.