DIRECTORATE OF DISTANCE EDUCATION

B.Sc. (INFORMATION TECHNOLOGY)

Second Year – Fourth Semester

12944 – Open Source Lab

Copy Right Reserved For Private Use only
B.Sc. (INFORMATION TECHNOLOGY)

Second Year – Fourth Semester

12944 – Open Source Lab

Author:

Dr. M. Ilayaraja
Assistant Professor
Department of Computer Science and Information Technology
Kalasalingam Academy of Research and Education
Anand Nagar, Krishnankoil. 626126.
LAB PROGRAMMING IN OPEN SOURCE

SYLLABI

BLOCK 1: INTRODUCTION TO LINUX
1. Kernel Configuration, compilation and installation
2. Install various software on Linux, Install and Configure XAMP, Unix Commands and Shell Programming

BLOCK 2: MYSQL
3. Creating simple table with constraints
4. Demonstration of joining tables
5. Database connectivity in PHP with MYSQL

BLOCK 3: PHP
6. PHP Simple Programs
7. PHP Web programs arrays and functions
8. File manipulation using PHP

BLOCK 4: PERL AND PYTHON PROGRAMMING
9. PERL programs
10. Python Programming
11. Python Programming: Arrays
12. Python Programming: String

BLOCK 5: APPLICATION DEVELOPMENT
13. Connect to a MYSQL database with PHP, PERL and Python
14. Developing simple applications using PHP and MYSQL
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 usually 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 will 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 will ensure that all packages required to compile the current running kernel will be installed. We will be using the same tools to compile the newer custom kernel.

Remove traces of old kernel source if they exist. Be very careful 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 usually 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 will be suitable to create a working kernel. We can experiment with modifying the kernel options after compiling the kernel successfully 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 install 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 install 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 successfully and redo the exercise carefully.

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.
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.
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`.

[password for wikihowsteve: ]
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 used 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]

**NOTES**

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

```bash
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.

---

**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**

```bash
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**
```bash
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

Self-Instructional Material
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"
```

NOTES
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 commands

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.

NOT NULL constraint restricts a column from having a NULL value. Once NOT NULL constraint is applied to a column, you cannot pass a null value to that column. It enforces a column to contain a proper value.

CREATE TABLE Student
(
s_id int NOT NULL,
Name varchar(60),
Age int
);

UNIQUE constraint ensures that a field or column will only have unique values. A UNIQUE constraint field will not have duplicate data. This constraint can be applied at column level or table level.

CREATE TABLE Student
(
s_id int NOT NULL UNIQUE,
Name varchar(60),
**Primary key constraint** uniquely identifies each record in a database. A Primary Key must contain unique value and it must not contain null value. Usually Primary Key is used to index the data inside the table.

```sql
CREATE table Student
(
s_id int PRIMARY KEY,
Name varchar(60) NOT NULL,
Age int
);
```
The above command will creates a PRIMARY KEY on the s_id.

**Foreign key constraints:** FOREIGN KEY is used to relate two tables. FOREIGN KEY constraint is also used to restrict actions that would destroy links between tables. To understand FOREIGN KEY, let's see its use, with help of the below tables:

**Customer_Detail Table**

<table>
<thead>
<tr>
<th>c_id</th>
<th>Customer_Name</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Adam</td>
<td>Noida</td>
</tr>
<tr>
<td>102</td>
<td>Alex</td>
<td>Delhi</td>
</tr>
<tr>
<td>103</td>
<td>Stuart</td>
<td>Rohtak</td>
</tr>
</tbody>
</table>

**Order_Detail Table**

<table>
<thead>
<tr>
<th>Order_id</th>
<th>Order_Name</th>
<th>c_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Order1</td>
<td>101</td>
</tr>
<tr>
<td>11</td>
<td>Order2</td>
<td>103</td>
</tr>
<tr>
<td>12</td>
<td>Order3</td>
<td>102</td>
</tr>
</tbody>
</table>

In Customer_Detail table, c_id is the primary key which is set as foreign key in Order_Detail table. The value that is entered in c_id which is set as foreign key in Order_Detail table must be present in Customer_Detail where it is set as primary key. This prevents invalid data to be inserted into c_id column of Order_Detail table.

If you try to insert any incorrect data, DBMS will return error and will not allow you to insert the data.
CREATE table Order_Detail
(
    order_id int PRIMARY KEY,
    order_name varchar(60) NOT NULL,
    c_id int FOREIGN KEY REFERENCES Customer_Detail(c_id)
);

In this query, c_id in table Order_Detail is made as foreign key, which is a reference of c_id column in Customer_Detail table.

**CHECK constraint** is used to restrict the value of a column between a range. It performs check on the values, before storing them into the database. Its like condition checking before saving data into a column.

CREATE table Student
(
    s_id int NOT NULL CHECK(s_id > 0),
    Name varchar(60) NOT NULL,
    Age int
);

**DEFAULT constraint** is used to provide a default value for a column. The default value will be added to all new records IF no other value is specified.

The following SQL sets a DEFAULT value for the "City" column when the "Persons" table is created:

CREATE TABLE Persons
(
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255) DEFAULT 'Sandnes'
);
INSERT, UPDATE AND DELETE ROWS IN TABLE

AIM:
To insert, Update and delete rows in a table

Procedure
Insert Row

Now you are ready to insert some records into the following customer table:

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>CustomerName</th>
<th>Address</th>
<th>City</th>
<th>Country</th>
</tr>
</thead>
</table>

There are two different ways to insert data in all the columns:

Method-1
Syntax:
insert into <table name> (columns name) values (set of value);

INSERT INTO customer (CustomerID, CustomerName, Address, City, Country) VALUES (101, 'John', 'Anna Nagar', 'Chennai', 'India');

Check: select * from customer;

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>CustomerName</th>
<th>Address</th>
<th>City</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>John</td>
<td>Anna Nagar</td>
<td>Chennai</td>
<td>India</td>
</tr>
</tbody>
</table>

Method-2
Syntax:
insert into <table name> values (set of value);

INSERT INTO customer VALUES (102, 'Kumar', 'Ashok Nagar', 'Mumbai', 'India');

Check: select * from customer;

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>CustomerName</th>
<th>Address</th>
<th>City</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
<td>Kumar</td>
<td>Ashok</td>
<td>Mumbai</td>
<td>India</td>
</tr>
</tbody>
</table>

Insert Data Only in Specified Columns:
It is also possible to only insert data in specific columns. The following SQL statement will insert a new record, but only insert data in the “CustomerID”, "CustomerName", "City", and "Country" columns. INSERT INTO Customer (CustomerID, CustomerName, City, Country) VALUES (103, 'Ravi', 'Bangalore', 'India'); (Note the NULL value for the missing middle name.)

Check: select * from Customer;

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>CustomerName</th>
<th>Address</th>
<th>City</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td>Ravi</td>
<td>NULL</td>
<td>Bangalore</td>
<td>India</td>
</tr>
</tbody>
</table>

**Update Row**

The UPDATE statement is used to modify the existing records in a table. Syntax:

UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;

Consider the below table:

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>CustomerName</th>
<th>Address</th>
<th>City</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>John</td>
<td>Anna Nagar</td>
<td>Chennai</td>
<td>India</td>
</tr>
<tr>
<td>102</td>
<td>Kumar</td>
<td>Ashok</td>
<td>Mumbai</td>
<td>India</td>
</tr>
</tbody>
</table>

The following SQL statement updates the customer (CustomerID = 101) with a new city.

UPDATE Customer
SET City = 'Trichy'
WHERE CustomerID = 101;

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>CustomerName</th>
<th>Address</th>
<th>City</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>John</td>
<td>Anna Nagar</td>
<td>Trichy</td>
<td>India</td>
</tr>
<tr>
<td>102</td>
<td>Kumar</td>
<td>Ashok</td>
<td>Mumbai</td>
<td>India</td>
</tr>
</tbody>
</table>

The following SQL statement updates the customer (CustomerID = 102) with a new CustomerName and a new Country.
UPDATE Customer
SET CustomerName='Shankar', Country= 'Srilanka'
WHERE CustomerID = 102;

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>CustomerName</th>
<th>Address</th>
<th>City</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>John</td>
<td>Anna Nagar</td>
<td>Chennai</td>
<td>India</td>
</tr>
<tr>
<td>102</td>
<td>Shankar</td>
<td>Ashok</td>
<td>Trichy</td>
<td>Srilanka</td>
</tr>
</tbody>
</table>

**Table Deletion**

Syntax:
drop table [table name];

For example: drop table Customer;

**Row Deletion**

The DELETE statement is used to delete existing records in a table.

Syntax:
DELETE FROM table_name WHERE condition;

The following SQL statement deletes the customer "Alfreds Futterkiste" from the below "customer" table:

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>CustomerName</th>
<th>Address</th>
<th>City</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>John</td>
<td>Anna Nagar</td>
<td>Chennai</td>
<td>India</td>
</tr>
<tr>
<td>102</td>
<td>Shankar</td>
<td>Ashok</td>
<td>Trichy</td>
<td>Srilanka</td>
</tr>
</tbody>
</table>

DELETE FROM Customer WHERE CustomerName='John';

Check: Select * from customer;

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>CustomerName</th>
<th>Address</th>
<th>City</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
<td>Shankar</td>
<td>Ashok</td>
<td>Trichy</td>
<td>Srilanka</td>
</tr>
</tbody>
</table>

**Delete All Records**

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:
Syntax:

DELETE FROM table_name;

The following SQL statement deletes all rows in the "customer" table, without deleting the table:

DELETE FROM customer;

Check: select * from customer;

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>CustomerName</th>
<th>Address</th>
<th>City</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**SORTING DATA**

**AIM:**
To sort data in a table

**Procedure**
The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

Syntax:

SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;

Consider the following table

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Project</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Henry</td>
<td>Java</td>
<td>50000</td>
</tr>
<tr>
<td>E2</td>
<td>Ford</td>
<td>.net</td>
<td>60000</td>
</tr>
<tr>
<td>E3</td>
<td>Albert</td>
<td>Java</td>
<td>70000</td>
</tr>
<tr>
<td>E6</td>
<td>Silber</td>
<td>Pascal</td>
<td>70000</td>
</tr>
<tr>
<td>E4</td>
<td>Josh</td>
<td>Unix</td>
<td>65000</td>
</tr>
<tr>
<td>E5</td>
<td>Bosch</td>
<td>Pascal</td>
<td>50000</td>
</tr>
</tbody>
</table>

Display the employee details according to their salary (high to low)
Select * from Employee order by salary desc;
Display the employee details according to their salary (high to low) and id descending order.

```
Select * from Employee order by salary desc, id desc;
```

Output:

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Project</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>E6</td>
<td>Silber</td>
<td>Pascal</td>
<td>70000</td>
</tr>
<tr>
<td>E3</td>
<td>Albert</td>
<td>Java</td>
<td>70000</td>
</tr>
<tr>
<td>E4</td>
<td>Josh</td>
<td>Unix</td>
<td>65000</td>
</tr>
<tr>
<td>E2</td>
<td>Ford</td>
<td>.net</td>
<td>60000</td>
</tr>
<tr>
<td>E1</td>
<td>Henry</td>
<td>Java</td>
<td>50000</td>
</tr>
<tr>
<td>E5</td>
<td>Bosch</td>
<td>Pascal</td>
<td>50000</td>
</tr>
</tbody>
</table>

### 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.

**Subqueries with the SELECT Statement**

Subqueries are most frequently used with the SELECT statement.

**Syntax:**

```
SELECT column_name [, column_name ]
FROM table1 [, table2 ]
```
WHERE  column_name OPERATOR  
(SELECT column_name [, column_name ]  
FROM table1 [, table2 ] [WHERE] Condition)

NOTES

Example:
Consider the CUSTOMERS table having the following records:

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Age</th>
<th>Address</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ramesh</td>
<td>35</td>
<td>Ahmedabad</td>
<td>2000</td>
</tr>
<tr>
<td>2</td>
<td>Khilan</td>
<td>25</td>
<td>Delhi</td>
<td>1500</td>
</tr>
<tr>
<td>3</td>
<td>Kaushik</td>
<td>23</td>
<td>Kota</td>
<td>2000</td>
</tr>
<tr>
<td>4</td>
<td>Chaitali</td>
<td>25</td>
<td>Mumbai</td>
<td>6500</td>
</tr>
<tr>
<td>5</td>
<td>Hardik</td>
<td>27</td>
<td>Bhopal</td>
<td>8500</td>
</tr>
<tr>
<td>6</td>
<td>Komal</td>
<td>22</td>
<td>MP</td>
<td>4500</td>
</tr>
<tr>
<td>7</td>
<td>Muffy</td>
<td>24</td>
<td>Indore</td>
<td>10000</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Age</th>
<th>Address</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Chaitali</td>
<td>25</td>
<td>Mumbai</td>
<td>6500</td>
</tr>
<tr>
<td>5</td>
<td>Hardik</td>
<td>27</td>
<td>Bhopal</td>
<td>8500</td>
</tr>
<tr>
<td>7</td>
<td>Muffy</td>
<td>24</td>
<td>Indore</td>
<td>10000</td>
</tr>
</tbody>
</table>

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.

Syntax:

UPDATE table SET column_name = new_value 
[ WHERE OPERATOR [ VALUE ]  
(SELECT COLUMN_NAME FROM TABLE_NAME)  
[ WHERE ] ]

Example:
Assuming, we have 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
WHERE AGE >= 27 );

Output:

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Age</th>
<th>Address</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ramesh</td>
<td>35</td>
<td>Ahmedabad</td>
<td>2500</td>
</tr>
<tr>
<td>2</td>
<td>Khilan</td>
<td>25</td>
<td>Delhi</td>
<td>1500</td>
</tr>
<tr>
<td>3</td>
<td>Kaushik</td>
<td>23</td>
<td>Kota</td>
<td>2000</td>
</tr>
<tr>
<td>4</td>
<td>Chaitali</td>
<td>25</td>
<td>Mumbai</td>
<td>6500</td>
</tr>
<tr>
<td>5</td>
<td>Hardik</td>
<td>27</td>
<td>Bhopal</td>
<td>10625</td>
</tr>
<tr>
<td>6</td>
<td>Komal</td>
<td>22</td>
<td>MP</td>
<td>4500</td>
</tr>
<tr>
<td>7</td>
<td>Muffy</td>
<td>24</td>
<td>Indore</td>
<td>10000</td>
</tr>
</tbody>
</table>

Subqueries with the DELETE Statement

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

Syntax:

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

Example:

Assuming, we have a CUSTOMERS 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.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Age</th>
<th>Address</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Khilan</td>
<td>25</td>
<td>Delhi</td>
<td>1500</td>
</tr>
<tr>
<td>3</td>
<td>Kaushik</td>
<td>23</td>
<td>Kota</td>
<td>2000</td>
</tr>
<tr>
<td>4</td>
<td>Chaitali</td>
<td>25</td>
<td>Mumbai</td>
<td>6500</td>
</tr>
<tr>
<td>6</td>
<td>Komal</td>
<td>22</td>
<td>MP</td>
<td>4500</td>
</tr>
<tr>
<td>7</td>
<td>Muffy</td>
<td>24</td>
<td>Indore</td>
<td>10000</td>
</tr>
</tbody>
</table>
Exercise:

Construct the following queries in SQL.
1. Find the loan details of customers in the branch ‘chennai’;
2. Find the customer details who are depositors.
3. Find the bank balance of customers whose age is above 60;

Consider the following relations:
Customer<id, name, branch, age>
Depositor<id, account_number, balance>
Loan<id, loan_number, amount>

Procedures

Select * from Customer;

<table>
<thead>
<tr>
<th>id</th>
<th>Name</th>
<th>Age</th>
<th>branch</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Alfred</td>
<td>28</td>
<td>Goa</td>
</tr>
<tr>
<td>C2</td>
<td>John</td>
<td>45</td>
<td>Chennai</td>
</tr>
<tr>
<td>C3</td>
<td>Merlin</td>
<td>64</td>
<td>Delhi</td>
</tr>
<tr>
<td>C4</td>
<td>Sam</td>
<td>62</td>
<td>Chennai</td>
</tr>
<tr>
<td>C5</td>
<td>Rose</td>
<td>25</td>
<td>Chennai</td>
</tr>
</tbody>
</table>

Select * from Depositor;

<table>
<thead>
<tr>
<th>Id</th>
<th>Account_number</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>AC11112</td>
<td>100000</td>
</tr>
<tr>
<td>C3</td>
<td>AC11113</td>
<td>200000</td>
</tr>
<tr>
<td>C4</td>
<td>AC11114</td>
<td>500000</td>
</tr>
</tbody>
</table>

Select * from Loan;

<table>
<thead>
<tr>
<th>Id</th>
<th>Loan_number</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>C5</td>
<td>AC11115</td>
<td>50000</td>
</tr>
<tr>
<td>C1</td>
<td>AC11111</td>
<td>80000</td>
</tr>
<tr>
<td>C3</td>
<td>AC11116</td>
<td>70000</td>
</tr>
</tbody>
</table>

Answers:
1. Find the loan details of customers in the branch ‘chennai’;
   
   Select id,loan_number,amount from Loan where id IN(Select id from Customer where branch =‘chennai’);

   Output:

<table>
<thead>
<tr>
<th>Id</th>
<th>Loan_number</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>C5</td>
<td>AC11115</td>
<td>50000</td>
</tr>
</tbody>
</table>

2. Find the customer details who are depositors.
   
   Select id, name, age, branch from Customer where id IN (Select id from Depositor);
Output:

<table>
<thead>
<tr>
<th>Id</th>
<th>name</th>
<th>Age</th>
<th>Branch</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>John</td>
<td>45</td>
<td>Chennai</td>
</tr>
<tr>
<td>C3</td>
<td>Merlin</td>
<td>64</td>
<td>Delhi</td>
</tr>
<tr>
<td>C4</td>
<td>Sam</td>
<td>62</td>
<td>Chennai</td>
</tr>
</tbody>
</table>

3. Find the bank balance of customers whose age is above 60;

Select id, balance from depositor where id IN (select id from Customer where age >60);

Output:

<table>
<thead>
<tr>
<th>Id</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>C3</td>
<td>200000</td>
</tr>
<tr>
<td>C4</td>
<td>500000</td>
</tr>
</tbody>
</table>

---

**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>;

Examples:
SELECT AVG (Salary) FROM Employee;
SELECT DeptNo, AVG (Sal) FROM EMP GROUP BY DeptNo;

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:

i) COUNT (Column_Name)

ii) COUNT (*)

iii) COUNT (DISTINCT Column_Name)

i) Syntax:
SELECT COUNT (Column_Name) FROM <Table_Name>;

Example:
SELECT COUNT (Ename) FROM Employee;

ii) 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”

iii) Syntax:
SELECT COUNT (DISTINCT Column_Name) FROM <Table_Name>;

Example:
SELECT COUNT (DISTINCT Job) FROM EMP;
“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>;

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

Syntax:
SELECT MIN (Column_Name) FROM <Table_Name>;

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;

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>;
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>;

**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>;

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;

**Example:**
SELECT ROUND (Comm, 2) FROM EMP;

**Exercise:**

Construct the following queries in SQL

1. Find the number of employees working in the ‘Sales’ department.
2. Find the employees’ average salary in the department ‘Admin’.
3. Find the employee details who are earning maximum salary in each department.
4. Find the minimum salary of employee.
5. Find the maximum salary of employee.
6. Find the total salary of employees working in department ‘Academic’.

Consider the following relation Employee<id, name, department, salary>
Procedures

Select * from employee;

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>Department</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Ajay</td>
<td>Academic</td>
<td>40000</td>
</tr>
<tr>
<td>E2</td>
<td>Arjun</td>
<td>Academic</td>
<td>50000</td>
</tr>
<tr>
<td>E3</td>
<td>Vijay</td>
<td>Academic</td>
<td>45000</td>
</tr>
<tr>
<td>E4</td>
<td>Manish</td>
<td>Admin</td>
<td>50000</td>
</tr>
<tr>
<td>E5</td>
<td>Shankar</td>
<td>Admin</td>
<td>60000</td>
</tr>
<tr>
<td>E6</td>
<td>Deepak</td>
<td>Admin</td>
<td>40000</td>
</tr>
<tr>
<td>E7</td>
<td>Sri</td>
<td>Sales</td>
<td>60000</td>
</tr>
<tr>
<td>E8</td>
<td>Prasath</td>
<td>Sales</td>
<td>80000</td>
</tr>
<tr>
<td>E9</td>
<td>Vinay</td>
<td>Sales</td>
<td>40000</td>
</tr>
<tr>
<td>E10</td>
<td>jegathesh</td>
<td>Sales</td>
<td>50000</td>
</tr>
</tbody>
</table>

1. Find the number of employees working in the ‘Sales’ department.
   Select count(id) from employee where department = ‘Sales’;

<table>
<thead>
<tr>
<th>Count(id)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

2. Find the employees’ average salary working in the department ‘Admin’.
   Select avg(Salary) from employee where department =’Admin’;

   Output:

<table>
<thead>
<tr>
<th>Avg(Salary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50000</td>
</tr>
</tbody>
</table>

3. Find the employee details who are earning maximum salary in each department.

   Select max(Salary),department from employee group by department;

<table>
<thead>
<tr>
<th>Max(Salary)</th>
<th>department</th>
</tr>
</thead>
<tbody>
<tr>
<td>50000</td>
<td>Academic</td>
</tr>
<tr>
<td>60000</td>
<td>Admin</td>
</tr>
<tr>
<td>80000</td>
<td>Sales</td>
</tr>
</tbody>
</table>
4. Find the minimum salary of employee

   
   \[ \text{Select min(Salary) from employee;} \]
   
   \begin{tabular}{|c|}
   \hline
   \text{Min(Salary)} \\
   \text{40000} \\
   \hline
   \end{tabular}
   
   5. Find the maximum salary of employee

   \[ \text{Select max(Salary) from employee;} \]
   
   \begin{tabular}{|c|}
   \hline
   \text{Max(Salary)} \\
   \text{80000} \\
   \hline
   \end{tabular}
   
   6. Find the total salary of employees working in department ‘Academic’.

   \[ \text{Select sum(Salary), department from employee where department =’Academic’;} \]

   \begin{tabular}{|c|c|}
   \hline
   \text{Sum(Salary)} & \text{department} \\
   \text{135000} & \text{Academic} \\
   \hline
   \end{tabular}

WORKING OF SET OPERATIONS IN MYSQL

AIM:
To learn and implement the set operations in mysql

Consider the following relation
Student<id, name, department>
Marks<id, subject_OS, subject_JAVA>

Construct the following SQL Queries using AND OR NOT OPERATION, UNION, INTERSECTION, PROJECTION.

1. Find the id of students who have passed in at least one subject.
2. Find the id of students who have passed in all the subjects.
3. Find the id of students who passed only in subject_java but not in subject_OS
Procedures

Select * from Marks;

<table>
<thead>
<tr>
<th>Id</th>
<th>Subject_OS</th>
<th>Subject_JAVA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>80</td>
<td>70</td>
</tr>
<tr>
<td>2</td>
<td>42</td>
<td>52</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>41</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>55</td>
</tr>
<tr>
<td>6</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>70</td>
<td>20</td>
</tr>
</tbody>
</table>

1. Find the id of students who have passed in at least one subject.

UNION:

Select id from Marks where subject_OS>40

Union

Select id from Marks where subject_JAVA>40;

Output:

<table>
<thead>
<tr>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

2. Find the id of students who have passed in all the subjects.

INTERSECTION:

Select id from Marks where subject_OS>40

INTERSECT

Select id from Marks where subject_JAVA>40;

Output:

<table>
<thead>
<tr>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>
3. Find the id of students who passed only in subject_java but not in subject_OS

```sql
EXCEPT
Select id from Marks where subject_JAVA>40
EXCEPT
Select id from Marks where subject_OS>40;
```

Output:

<table>
<thead>
<tr>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

---

**WORKING WITH STRINGS IN MYSQL**

**AIM:**

To implement string operations in mysql

**String Functions**

String functions are used to perform an operation on input string and return an output string. Following are the string functions defined in SQL:

1. **ASCII()**: This function is used to find the ASCII value of a character.
   
   **Syntax:** SELECT ascii('t');
   
   **Output:** 116

2. **CHAR_LENGTH()**: This function is used to find the length of a word.
   
   **Syntax:** SELECT char_length('Hello!');
   
   **Output:** 6

3. **CHARACTER_LENGTH()**: This function is used to find the length of a line.
   
   **Syntax:** SELECT CHARACTER_LENGTH('geeks for geeks');
   
   **Output:** 15

4. **CONCAT()**: This function is used to add two words or strings.
   
   **Syntax:** SELECT 'Geeks' || ' ' || 'forGeeks' FROM dual;
   
   **Output:** ‘GeeksforGeeks’
5. **CONCAT_WS()**: This function is used to add two words or strings with a symbol as concatenating symbol.
   **Syntax**: SELECT CONCAT_WS('_', 'geeks', 'for', 'geeks');
   **Output**: geeks_for_geeks

6. **FIND_IN_SET()**: This function is used to find a symbol from a set of symbols.
   **Syntax**: SELECT FIND_IN_SET('b', 'a, b, c, d, e, f');
   **Output**: 2

7. **FORMAT()**: This function is used to display a number in the given format.
   **Syntax**: Format("0.981", "Percent");
   **Output**: ‘98.10%’

8. **INSERT()**: This function is used to insert the data into a database.
    **Syntax**: INSERT INTO database (geek_id, geek_name) VALUES (5000, 'abc');
    **Output**: successfully updated

9. **INSTR()**: This function is used to find the occurrence of an alphabet.
    **Syntax**: INSTR('geeks for geeks', 'e');
    **Output**: 2 (the first occurrence of ‘e’)

    **Syntax**: INSTR('geeks for geeks', 'e', 1, 2 );
    **Output**: 3 (the second occurrence of ‘e’)

10. **LCASE()**: This function is used to convert the given string into lower case.
    **Syntax**: LCASE ("GeeksFor Geeks To Learn");
    **Output**: geeksforgeeks to learn

11. **LEFT()**: This function is used to SELECT a sub string from the left of given size or characters.
    **Syntax**: SELECT LEFT('geeksforgeeks.org', 5);
    **Output**: geeks

12. **LENGTH()**: This function is used to find the length of a word.
    **Syntax**: LENGTH('GeeksForGeeks');
    **Output**: 13

13. **LOCATE()**: This function is used to find the nth position of the given word in a string.
    **Syntax**: SELECT LOCATE('for', 'geeksforgeeks', 1);
    **Output**: 6

14. **LOWER()**: This function is used to convert the upper case string into lower case.
Syntax: SELECT LOWER('GEEKSFOR GEEKS.ORG');
Output: geeksforgeeks.org

15. **LPAD()**: This function is used to make the given string of the given size by adding the given symbol.
   **Syntax**: LPAD('geeks', 8, '0');
   **Output**: 000geeks

16. **LTRIM()**: This function is used to cut the given sub string from the original string.
    **Syntax**: LTRIM('123123geeks', '123');
    **Output**: geeks

17. **MID()**: This function is to find a word from the given position and of the given size.
    **Syntax**: Mid("geeksforgeeks", 6, 2);
    **Output**: for

18. **POSITION()**: This function is used to find position of the first occurrence of the given alphabet.
    **Syntax**: SELECT POSITION('e' IN 'geeksforgeeks');
    **Output**: 2

19. **REPEAT()**: This function is used to write the given string again and again till the number of times mentioned.
    **Syntax**: SELECT REPEAT('geeks', 2);
    **Output**: geeksgeeks

20. **REPLACE()**: This function is used to cut the given string by removing the given sub string.
    **Syntax**: REPLACE('123geeks123', '123');
    **Output**: geeks

21. **REVERSE()**: This function is used to reverse a string.
    **Syntax**: SELECT REVERSE('geeksforgeeks.org');
    **Output**: ‘gro. skeegrofskeeg’

22. **RIGHT()**: This function is used to SELECT a sub string from the right end of the given size.
    **Syntax**: SELECT RIGHT('geeksforgeeks.org', 4);
    **Output**: ‘.org’

23. **RPAD()**: This function is used to make the given string as long as the given size by adding the given symbol on the right.
    **Syntax**: RPAD('geeks', 8, '0');
    **Output**: ‘geeks000’
24. **RTRIM()**: This function is used to cut the given sub string from the original string.
   Syntax: RTRIM('geeksxyzyyyy', 'xyz');
   Output: ‘geeks’

25. **SPACE()**: This function is used to write the given number of spaces.
    Syntax: SELECT SPACE(7);
    Output: ‘       ’

26. **STRCMP()**: This function is used to compare 2 strings.
    a. If string1 and string2 are the same, the STRCMP function will return 0.
    b. If string1 is smaller than string2, the STRCMP function will return -1.
    c. If string1 is larger than string2, the STRCMP function will return 1.
    Syntax: SELECT STRCMP('google.com', 'geeksforgeeks.com');
    Output: -1

27. **SUBSTR()**: This function is used to find a sub string from the a string from the given position.
    Syntax: SUBSTR('geeksforgeeks', 1, 5);
    Output: ‘geeks’

28. **SUBSTRING()**: This function is used to find an alphabet from the mentioned size and the given string.
    Syntax: SELECT SUBSTRING('GeeksForGeeks.org', 9, 1);
    Output: ‘G’

29. **SUBSTRING_INDEX()**: This function is used to find a sub string before the given symbol.
    Syntax: SELECT SUBSTRING_INDEX('www.geeksforgeeks.org', '.', 1);
    Output: ‘www’

30. **TRIM()**: This function is used to cut the given symbol from the string.
    Syntax: TRIM(LEADING '0' FROM '000123');
    Output: 123

31. **UCASE()**: This function is used to make the string in upper case.
    Syntax: UCASE("GeeksForGeeks");
    Output: GEEKSFORGEEKS
AIM:
To implement numeric functions in Mysql

Numeric Functions
Numeric Functions are used to perform operations on numbers and return numbers.
Following are the numeric functions defined in SQL:

1. **ABS()**: It returns the absolute value of a number.
   **Syntax**: `SELECT ABS(-243.5);`
   **Output**: 243.5

2. **ACOS()**: It returns the cosine of a number.
   **Syntax**: `SELECT ACOS(0.25);`
   **Output**: 1.318116071652818

3. **ASIN()**: It returns the arc sine of a number.
   **Syntax**: `SELECT ASIN(0.25);`
   **Output**: 0.25268025514207865

4. **ATAN()**: It returns the arc tangent of a number.
   **Syntax**: `SELECT ATAN(2.5);`
   **Output**: 1.1902899496825317

5. **CEIL()**: It returns the smallest integer value that is greater than or equal to a number.
   **Syntax**: `SELECT CEIL(25.75);`
   **Output**: 26

6. **CEILING()**: It returns the smallest integer value that is greater than or equal to a number.
   **Syntax**: `SELECT CEILING(25.75);`
   **Output**: 26

7. **COS()**: It returns the cosine of a number.
   **Syntax**: `SELECT COS(30);`
   **Output**: 0.15425144988758405

8. **COT()**: It returns the cotangent of a number.
    **Syntax**: `SELECT COT(6);`
    **Output**: -3.4365353004180128

9. **DEGREES()**: It converts a radian value into degrees.
    **Syntax**: `SELECT DEGREES(1.5);`
    **Output**: 85.94366926962348
10. **DIV()**: It is used for integer division.
    Syntax: SELECT 10 DIV 5;
    Output: 2

11. **EXP()**: It returns e raised to the power of number.
    Syntax: SELECT EXP(1);
    Output: 2.718281828459045

12. **FLOOR()**: It returns the largest integer value that is less than or equal to a number.
    Syntax: SELECT FLOOR(25.75);
    Output: 25

13. **GREATEST()**: It returns the greatest value in a list of expressions.
    Syntax: SELECT GREATEST(30, 2, 36, 81, 125);
    Output: 125

14. **LEAST()**: It returns the smallest value in a list of expressions.
    Syntax: SELECT LEAST(30, 2, 36, 81, 125);
    Output: 2

15. **LN()**: It returns the natural logarithm of a number.
    Syntax: SELECT LN(2);
    Output: 0.6931471805599453

16. **LOG10()**: It returns the base-10 logarithm of a number.
    Syntax: SELECT LOG(2);
    Output: 0.6931471805599453

17. **LOG2()**: It returns the base-2 logarithm of a number.
    Syntax: SELECT LOG2(6);
    Output: 2.584962500721156

18. **MOD()**: It returns the remainder of n divided by m.
    Syntax: SELECT MOD(18, 4);
    Output: 2

19. **PI()**: It returns the value of PI displayed with 6 decimal places.
    Syntax: SELECT PI();
    Output: 3.141593

20. **POW()**: It returns m raised to the nth power.
    Syntax: SELECT POW(4, 2);
    Output: 16

21. **RADIANS()**: It converts a value in degrees to radians.
    Syntax: SELECT RADIANS(180);
    Output: 3.141592653589793
22. **RAND()**: It returns a random number.
   Syntax: `SELECT RAND();`
   Output: 0.33623238684258644

23. **ROUND()**: It returns a number rounded to a certain number of decimal places.
   Syntax: `SELECT ROUND(5.553);`
   Output: 6

24. **SIGN()**: It returns a value indicating the sign of a number.
    Syntax: `SELECT SIGN(255.5);`
    Output: 1

25. **SIN()**: It returns the sine of a number.
    Syntax: `SELECT SIN(2);`
    Output: 0.9092974268256817

26. **SQRT()**: It returns the square root of a number.
    Syntax: `SELECT SQRT(25);`
    Output: 5

27. **TAN()**: It returns the tangent of a number.
    Syntax: `SELECT TAN(1.75);`
    Output: -5.52037992250933

28. **ATAN2()**: It returns the arctangent of the x and y coordinates, as an angle and expressed in radians.
    Syntax: `SELECT ATAN2(7);`
    Output: 1.42889927219073

29. **TRUNCATE()**: It returns 7.53635 truncated to 2 places right of the decimal point.
    Syntax: `SELECT TRUNCATE(7.53635, 2);`
    Output: 7.53
**DATE FUNCTIONS**

**AIM:**
To learn date functions in mysql

**Procedure**

SQL | Date functions

In SQL, dates are complicated for newbies, since while working with database, the format of the date in table must be matched with the input date in order to insert. In various scenarios instead of date, datetime (time is also involved with date) is used. In MySql the default date functions are:

1. **NOW()**: Returns the current date and time.
   
   Example:
   
   ```sql
   SELECT NOW();
   ```
   
   Output:
   
   `2017-01-13 08:03:52`

2. **CURDATE()**: Returns the current date.
   
   Example:
   
   ```sql
   SELECT CURDATE();
   ```
   
   Output:
   
   `2017-01-13`

3. **CURTIME()**: Returns the current time.
   
   Example:
   
   ```sql
   SELECT CURTIME();
   ```
   
   Output:
   
   `08:05:15`

4. **DATE()**: Extracts the date part of a date or date/time expression.
   
   Example:
   
   For the below table named ‘Test’
   
<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>BirthTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>4120</td>
<td>Pratik</td>
<td>1996-09-26 16:44:15.581</td>
</tr>
</tbody>
</table>

   ```sql
   SELECT Name, DATE(BirthTime) AS BirthDate FROM Test;
   ```
   
   Output:
   
<table>
<thead>
<tr>
<th>Name</th>
<th>BirthTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pratik</td>
<td>1996-09-26</td>
</tr>
</tbody>
</table>
5. **EXTRACT()**: Returns a single part of a date/time.

Syntax:
```
EXTRACT(unit FORM date);
```

There are several units that can be considered but only some are used such as: MICROSECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, YEAR, etc.

And ‘date’ is a valid date expression.

Example:
For the below table named ‘Test’

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>BirthTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>4120</td>
<td>Pratik</td>
<td>1996-09-26 16:44:15.581</td>
</tr>
</tbody>
</table>

Queries

- **SELECT** Name, Extract(DAY FROM BirthTime) AS BirthDay FROM Test;

  Output:

<table>
<thead>
<tr>
<th>Name</th>
<th>BirthDay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pratik</td>
<td>26</td>
</tr>
</tbody>
</table>

- **SELECT** Name, Extract(YEAR FROM BirthTime) AS BirthYear FROM Test;

  Output:

<table>
<thead>
<tr>
<th>Name</th>
<th>BirthYear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pratik</td>
<td>1996</td>
</tr>
</tbody>
</table>

- **SELECT** Name, Extract(SECOND FROM BirthTime) AS BirthSecond FROM Test;

  Output:

<table>
<thead>
<tr>
<th>Name</th>
<th>BirthSecond</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pratik</td>
<td>581</td>
</tr>
</tbody>
</table>

6. **DATE_ADD()**: Adds a specified time interval to a date

Syntax:
```
DATE_ADD(date, INTERVAL expr type);
```

Where, date – valid date expression and expr is the number of interval we want to add. Type can be one of the following: MICROSECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, YEAR, etc.
Example:
For the below table named ‘Test’

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>BirthTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>4120</td>
<td>Pratik</td>
<td>1996-09-26 16:44:15.581</td>
</tr>
</tbody>
</table>

 Queries

- SELECT Name, DATE_ADD(BirthTime, INTERVAL 1 YEAR) AS BirthTimeModified FROM Test;
  
  Output:

<table>
<thead>
<tr>
<th>Name</th>
<th>BirthTimeModified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pratik</td>
<td>1997-09-26 16:44:15.581</td>
</tr>
</tbody>
</table>

- SELECT Name, DATE_ADD(BirthTime, INTERVAL 30 DAY) AS BirthDayModified FROM Test;
  
  Output:

<table>
<thead>
<tr>
<th>Name</th>
<th>BirthTimeModified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pratik</td>
<td>1996-10-26 16:44:15.581</td>
</tr>
</tbody>
</table>

- SELECT Name, DATE_ADD(BirthTime, INTERVAL 4 HOUR) AS BirthHourModified FROM Test;
  
  Output:

<table>
<thead>
<tr>
<th>Name</th>
<th>BirthTimeModified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pratik</td>
<td>1996-10-26 20:44:15.581</td>
</tr>
</tbody>
</table>

7. **DATE_SUB()**: Subtracts a specified time interval from a date. Syntax for DATE_SUB is same as DATE_ADD just the difference is that DATE_SUB is used to subtract a given interval of date.

8. **DATEDIFF()**: Returns the number of days between two dates.
   
   Syntax:
   
   DATEDIFF(date1, date2);
   
   date1 & date2- date/time expression
   
   Example:
   
   SELECT DATE_DIFF(’2017-01-13’,’2017-01-03’) AS DateDiff;
   
   Output:
   
   DateDiff

9. **DATE_FORMAT()**: Displays date/time data in different formats.
   
   Syntax:
   
   DATE_FORMAT(date,format);
date is a valid date and format specifies the output format for the
date/time. The formats that can be used are:

- %a-Abbreviated weekday name (Sun-Sat)
- %b-Abbreviated month name (Jan-Dec)
- %c-Month, numeric (0-12)
- %D-Day of month with English suffix (0th, 1st, 2nd, 3rd)
- %d-Day of month, numeric (00-31)
- %e-Day of month, numeric (0-31)
- %f-Microseconds (000000-999999)
- %H-Hour (00-23)
- %h-Hour (01-12)
- %i-Hour (01-12)
- %j-Minutes, numeric (00-59)
- %j-Day of year (001-366)
- %k-Hour (0-23)
- %l-Hour (1-12)
- %M-Month name (January-December)
- %m-Month, numeric (00-12)
- %p-AM or PM
- %r-Time, 12-hour (hh:mm:ss followed by AM or PM)
- %S-Seconds (00-59)
- %s-Seconds (00-59)
- %T-Time, 24-hour (hh:mm:ss)
- %U-Week (00-53) where Sunday is the first day of week
- %u-Week (00-53) where Monday is the first day of week
- %V-Week (01-53) where Sunday is the first day of week, used
  with %X
- %v-Week (01-53) where Monday is the first day of week, used
  with %x
- %W-Weekday name (Sunday-Saturday)
- %w-Day of the week (0=Sunday, 6=Saturday)
- %X-Year for the week where Sunday is the first day of week,
  four digits, used with %V
- %x-Year for the week where Monday is the first day of week,
  four digits, used with %x
- %Y-Year, numeric, four digits
- %y-Year, numeric, two digits

Example:
DATE_FORMAT(NOW(),'%d %b %y')

Result:
13 Jan 17
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/>
<h1>Name:</h1> <input type="text" name="name">   <br>
<h1>Password:</h1> <input type="text" name="password">
</div>

<?php
$name = "";
$pw = "";
?>
<html>
<head>
<style>
#main
{
Self-Instructional Material

NOTES
<?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>"; }

$name = "";
$pw = "";

<h2>Enter your username and password</h2>
<br/>

<h1>Name: </h1> <input type="text" name="name">
<br>

<h1>Password: </h1> <input type="text" name="password">
</div>
</body>
</html>
AIM:
To format the output 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);
?>
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>
**Output**

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 TWO NUMBERS USING FUNCTIONS**

**AIM:**
Add two numbers using function in PHP

**Program**

```
<html>
<head>
<title>Writing PHP Function with Parameters</title>
</head>
<body>
<?php
  function addFunction($num1, $num2) {
    $sum = $num1 + $num2;
    echo "Sum of the two numbers is : $sum";
  }
  addFunction(10, 20);
?>
</body>
</html>
```

**Output**
Sum of the two numbers is : 30
**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 
[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>

Output
HELLOWORLD
AIM:
To create a program with conditional statements in PHP

PHP program to find odd or even number from given number

<html>
<head>
<title>Find out odd or even number. !</title>
</head>
<body>
<?php
$a = 10;
if ($a % 2==0)
{
 echo "Given number is" . " <br> " . <b> EVEN</b> ;
}
else
{
 echo "Given number is" . " <br> " . <b> ODD</b> ";
}
?>
</body>
</html>

Output
Given number is
EVEN
**PHP program to find maximum of three numbers**

```php
<?php
$a = 1;
$b = 4;
$c = 3;
if($a > $b)
{
    if($a > $c)
        echo "Maximum num a = $a";
    else
        echo "Maximum num c = $c";
}
else
{
    if($c > $b)
        echo "Maximum num c = $c";
    else
        echo "Maximum num b = $b";
}
?>
```

**Output**
Maximum num b =4
PHP program to swap two numbers

```php
<?php
$a = 10;
$b = 20;
echo "a = $a"."<br>"."b = $b"."<br>";
$a = $a + $b;
$b = $a - $b;
$a = $a - $b;
echo "<b>After Swapping"."<br>"." a = $a"."<br>"."b = $b"."<br>";
?>
</body>
</html>
```

Output

a =10
b =20

After Swapping

a =20
b =10
PHP program to display your favorite color using switch statement

```php
<?php
$favcolor = "red";
switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

Output

Your favorite color is red!
AIM:
To use different loops in PHP

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

PHP program to find the factorial of a given number using for loop
```php
<?php
$num = 3;
$factorial = 1;
for ($x=$num; $x>=1; $x--)
{
    $factorial = $factorial * $x;
}
echo "The factorial of $num is $factorial";
?>
```

Output
The factorial of 3 is 6

PHP program to print values of multiplying 10 by 0 through to 10 using for loop
```php
<?php
for ($i = 0; $i < 10; $i++){
    $product = 10 * $i;
    echo "The product of 10 * $i is $product <br/>
";
}
?>
```
Output

The product of 10 x 0 is 0
The product of 10 x 1 is 10
The product of 10 x 2 is 20
The product of 10 x 3 is 30
The product of 10 x 4 is 40
The product of 10 x 5 is 50
The product of 10 x 6 is 60
The product of 10 x 7 is 70
The product of 10 x 8 is 80
The product of 10 x 9 is 90

**PHP program using a for loop to add all the integers between 0 and 30 and display the sum.**

```php
<?php
$sum = 0;
for($x=1; $x<=30; $x++)
{
    $sum +=$x;
}
echo "The sum of the numbers 0 to 30 is $sum"."
";
?>
```

Output

The sum of the numbers 0 to 30 is 46
PHP program to construct the following pattern, using nested for loop.

```php
<?php
for($x=1;$x<=5;$x++)
{
    for ($y=1;$y<=$x;$y++)
    {
        echo "*";
        if($y< $x)
            {
                echo " ";
            }
    }
echo "\n";
}
?>
```

**Output**

*  
* *  
* * *  
* * * *  
* * * * *  

**While Loop in PHP**

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

**PHP program to reverse a given number**

```php
<?php
$num = 23456;
$revnum = 0;
while ($num > 1)
{
    $revnum = $revnum * 10 + $num % 10;
    $num = intdiv($num, 10);
}
```
$rem = $num % 10;
$revnum = ($revnum * 10) + $rem;
$num = ($num / 10);
}
echo "Reverse number of 23456 is: $revnum";
?>

Output
Reverse number of 23456 is: 65432

**PHP program to compute the sum of the two reversed numbers and display the sum in reversed form.**

```php
<?php
function reverse_sum($n1, $n2)
{
    return reverse_integer($n1) + reverse_integer($n2);
}
function reverse_integer($n)
{
    $reverse = 0;
    while ($n > 0)
    {
        $reverse = $reverse * 10;
        $reverse = $reverse + $n % 10;
        $n = (int)($n/10);
    }
    return $reverse;
}
print_r(reverse_sum(13, 14)."\n");
print_r(reverse_sum(130, 1)."\n");
print_r(reverse_sum(305, 794)."\n");
?>

Output
72
32
1000
Do While loop in PHP

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

**PHP program to print your name five time using do while loop**

```php
<?php
    $x1=1;
    do
    {
        echo "Increment Number : $x1 <br />
        echo "Your Name <br />
        $x1=$x1+1;
    }while ($x1<=5)
?>
```

**Output**

Your Name
Your Name
Your Name
Your Name
Your Name

**PHP program to print even number from 1 to 10 using do while loop**

```php
<?php
    $num = 0;
    do {
        $num += 2;
        echo $num, "\n";
    } while ($num <=10);
?>
```
PHP program to display odd and even number of given number using do-while loop.

```php
<?php
$even="";
$odd="";
$i=1;
do {
  if($i%2==0) {
    $even = $even . " " .$i ;
  }
  else {
    $odd = $odd . " " .$i;
  }
  $i++;
}while($i<=20);
echo "The Even Numbers = " . $even ."<br/>";
echo "The Odd Numbers = " . $odd;
?>
```

**Output**

The Even Numbers = 2 4 6 8 10 12 14 16 18 20
The Odd Numbers = 1 3 5 7 9 11 13 15 17 19
CREATE USER DEFINED FUNCTIONS

Functions are group of statements that can perform a task

### PHP program to perform addition of two numbers using function

```php
<?php
function addition($a,$b)
{
    $sum=$a+$b;
    return $sum;
}

$answer = addition(5,7);
echo “The sum = ” . $answer;
?>
```

**Output**
The sum = 12

### Write a PHP function to check a number is prime or not.

```php
<?php
function IsPrime($n)
{
    for($x=2; $x<$n; $x++)
    {
        if($n %$x ==0)
        { return 0;
        }
    }
}
```
return 1;
}
$a = IsPrime(3);
if ($a==0)
echo 'This is not a Prime Number.....'.'"n"; 
else 
echo 'This is a Prime Number..'.'"n";
?>

Output
This is a Prime Number..

---

**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**

```html
<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 "F{LOWERS: \$FLOWER_SHOP[0].", \
\$FLOWER_SHOP[1].", \$FLOWER_SHOP[2]."};
?>
</body>
</html>

**Output**

Flowers: rose, daisy, orchid

**Associative array in PHP**

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

```php
<?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 
$flower_shop['daisy'].", and orchid
".$flower_shop['orchid'].";
?>
</body>
</html>
```
costs "$flower_shop['orchild']}";

Output

rose costs 5.00, daisy costs 4.00, and orchid costs

Loop through an Associative Array

```php
<?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

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.

```php
<?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**

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.

76
HANDLING COOKIES IN PHP

AIM:
To implement cookies in PHP

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

Updating Cookie
=====================================
```php
<?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
<?php
  unset($_COOKIE["yourcookie"]);
  /*Or*/
  setcookie("yourcookie","yourvalue",time()-1);
  /*it expired so it's deleted*/
?>
```
AIM:
To implement basic file operations such as open close read and write in PHP.

Program
```php
<?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();
}
```

```php
while(!feof($fp))
{
    // Code to read data from the file
}
```
<?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
<?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
<?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'] . ".";
```

NOTES
Self-Instructional Material
<?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
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// remove all session variables
session_unset();
// destroy the session
session_destroy();
?>
</body>
</html>
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>
Address:
<textarea name = "add" type = "textarea"></textarea>
<br><br>
<input type = "Submit" name = "insert" value = "Save">
<input type = "Reset" value = "Cancle">
</form>

NOTES
Self-Instructional Material
<?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
<?php

// Code for inserting data into a table in a MySQL database

mysql_close($con);

?>

O/P

<table>
<thead>
<tr>
<th>Roll No</th>
<th>Name</th>
<th>Gender</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>John</td>
<td>Male</td>
<td>Home</td>
</tr>
<tr>
<td>23</td>
<td>Mary</td>
<td>Female</td>
<td>School</td>
</tr>
</tbody>
</table>

Self-Instructional Material
CREATING TABLES WITH CONSTRAINTS

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"."\n";
$db = mysql_select_db("studinfo",$con);
if(!$db)
{
  die("Database not found".mysql_error());
}
echo "Database is selected"."\n";
$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>";
}
echo "</table>";

</html>

Output

Connection open

Database is selected

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Branch</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Kumar</td>
<td>IT</td>
</tr>
<tr>
<td>11</td>
<td>Ravi</td>
<td>CS</td>
</tr>
<tr>
<td>12</td>
<td>Shankar</td>
<td>CSE</td>
</tr>
</tbody>
</table>
AIM:
To learn how to do some common text processing tasks using Perl.

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.

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

Output
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 greets the user and tells the age of the user.

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 "Enter 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 "Hello, $name!"
print "You are $age years old."
```

On execution:
```
> perl name.pl
Enter you name and press return:Mickey Mouse
Enter your birth year and press return:1928

Output
Hello, Mickey Mouse! You are 83 years old. >
```
3. Analysing text and printing the statistics

This program reads 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.

```perl
> gedit words.pl
#!/usr/bin/env perl
#words.pl word FILE
#if no data filename is given, this program will 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 $words words in this file."
print "There are $char characters in this file."
print "The average number of words per line is $avgw.
print "The average number of characters per word is $avgc."

On execution:
> perl words . pl constitution_preamble.txt

**Output**
Enter the word you want to search for and press return:
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.
The average number of characters per word is 5.52941176470588.

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 command line
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.
The words and their frequency in the text is:

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

Output

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
color: 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
2 15 18 6 7 8 7 9 5 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

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 populated 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
#will be written. If it exists it will be overwritten.

while(<FILE>) {
    chomp;
    tr/A-Z/a-z/;
    tr/.,:;!"(){}//d;
    foreach $wd (split) {
        $count{$wd}++;
    }
}

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

foreach $w (sort by_number keys%count)  #The result will 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

**Output**

<table>
<thead>
<tr>
<th>Word</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>8</td>
</tr>
<tr>
<td>of</td>
<td>7</td>
</tr>
<tr>
<td>the</td>
<td>5</td>
</tr>
<tr>
<td>to</td>
<td>5</td>
</tr>
<tr>
<td>india</td>
<td>2</td>
</tr>
<tr>
<td>this</td>
<td>2</td>
</tr>
<tr>
<td>all</td>
<td>2</td>
</tr>
<tr>
<td>twenty-sixth</td>
<td>1</td>
</tr>
<tr>
<td>constitution</td>
<td>1</td>
</tr>
<tr>
<td>opportunity</td>
<td>1</td>
</tr>
<tr>
<td>constituent</td>
<td>1</td>
</tr>
<tr>
<td>individual</td>
<td>1</td>
</tr>
<tr>
<td>constitute</td>
<td>1</td>
</tr>
<tr>
<td>expression</td>
<td>1</td>
</tr>
<tr>
<td>democratic</td>
<td>1</td>
</tr>
<tr>
<td>fraternity</td>
<td>1</td>
</tr>
<tr>
<td>ourselves</td>
<td>1</td>
</tr>
<tr>
<td>integrity</td>
<td>1</td>
</tr>
<tr>
<td>socialist</td>
<td>1</td>
</tr>
<tr>
<td>political</td>
<td>1</td>
</tr>
<tr>
<td>sovereign</td>
<td>1</td>
</tr>
<tr>
<td>solemnly</td>
<td>1</td>
</tr>
<tr>
<td>assembly</td>
<td>1</td>
</tr>
<tr>
<td>NOTES</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
</tr>
<tr>
<td>citizens</td>
<td>1</td>
</tr>
<tr>
<td>resolved</td>
<td>1</td>
</tr>
<tr>
<td>november</td>
<td>1</td>
</tr>
<tr>
<td>economic</td>
<td>1</td>
</tr>
<tr>
<td>equality</td>
<td>1</td>
</tr>
<tr>
<td>assuring</td>
<td>1</td>
</tr>
<tr>
<td>republic</td>
<td>1</td>
</tr>
<tr>
<td>thought</td>
<td>1</td>
</tr>
<tr>
<td>dignity</td>
<td>1</td>
</tr>
<tr>
<td>worship</td>
<td>1</td>
</tr>
<tr>
<td>liberty</td>
<td>1</td>
</tr>
<tr>
<td>promote</td>
<td>1</td>
</tr>
<tr>
<td>justice</td>
<td>1</td>
</tr>
<tr>
<td>secular</td>
<td>1</td>
</tr>
<tr>
<td>secure</td>
<td>1</td>
</tr>
<tr>
<td>social</td>
<td>1</td>
</tr>
<tr>
<td>people</td>
<td>1</td>
</tr>
<tr>
<td>belief</td>
<td>1</td>
</tr>
<tr>
<td>nation</td>
<td>1</td>
</tr>
<tr>
<td>status</td>
<td>1</td>
</tr>
<tr>
<td>having</td>
<td>1</td>
</tr>
<tr>
<td>hereby</td>
<td>1</td>
</tr>
<tr>
<td>unity</td>
<td>1</td>
</tr>
<tr>
<td>among</td>
<td>1</td>
</tr>
<tr>
<td>faith</td>
<td>1</td>
</tr>
<tr>
<td>adopt</td>
<td>1</td>
</tr>
<tr>
<td>enact</td>
<td>1</td>
</tr>
<tr>
<td>give</td>
<td>1</td>
</tr>
<tr>
<td>Word</td>
<td>Count</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>them</td>
<td>1</td>
</tr>
<tr>
<td>1949</td>
<td>1</td>
</tr>
<tr>
<td>into</td>
<td>1</td>
</tr>
<tr>
<td>our</td>
<td>1</td>
</tr>
<tr>
<td>day</td>
<td>1</td>
</tr>
<tr>
<td>its</td>
<td>1</td>
</tr>
<tr>
<td>in</td>
<td>1</td>
</tr>
<tr>
<td>do</td>
<td>1</td>
</tr>
<tr>
<td>we</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
</tr>
</tbody>
</table>
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,

- if statement (Conditional statement)
- if … else statement (Alternative statement)
- elif statement (Chained condition)
- nested if statement

Program to check if the input year is a leap year or not

```python
year = int(input("Please Enter the Year you wish: "))
if ((year%400 == 0)or ((year%4 == 0 ) and (year%100 != 0))):
    print year, "is a Leap Year"
else:
    print year, "is Not a Leap Year"
```

Output
Please Enter the Year you wish:
2017
2017 is Not a Leap Year
Program to find odd or even of a given number

num = 22
if num % 2 == 0:
    print "Even Number"
else:
    print "Odd Number"

Output
Even Number

Program to find largest among given three numbers

a = int(input("Please Enter the First value: "))
b = int(input("Please Enter the Second value: "))
c = int(input("Please Enter the Third value: "))
if (a > b and a > c):
    print a, "is largest among the three"
elif (b > a and b > c):
    print b, "is largest among the three"
elif (c > a and c > b):
    print c, "is largest among the three"
else:
    print("Either any two values or all the three values are equal")

Output
Please Enter the First value: 25
Please Enter the Second value: 22
Please Enter the Third value: 15
25 is largest among the three
program to check, if the number is positive or negative or zero and display an appropriate message

```python
num = float(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
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,

- while loop
- for loop

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.

Program to get the Fibonacci series between 0 to 50.

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

Output

1
1
2
3
5
8
13
21
34
Program to prints number from 1 to 5.

count = 1
while True:
    print (count)
    count += 1
    if count > 5:
        break

Output
0
1
2
3
4

Program to prints out only odd numbers from 1 to 10.

for x in range(10):
    # Check if x is even
    if x % 2 == 0:
        continue
    print(x)

Output
1
3
5
7
9
Program to print prime numbers for a user provided range

    # change the values of lower and upper for a different result
    lower = int(input("Enter lower range"))
    upper = int(input("Enter upper range"))
    print("Prime numbers between",lower,"and",upper,"are:")
    for num in range(lower,upper + 1):
        # prime numbers are greater than 1
        if num > 1:
            for i in range(2,num):
                if (num % i) == 0:
                    break
            else:
                print(num)

Output

Enter lower range 2
Enter upper range 20
Prime numbers between",lower,"and",upper,"are:
2
3
5
7
11
13
17
19
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"
```

Program to demonstrate various strings functions and operations

```
#str1=raw_input('Enter the 1st String')
#str2=raw_input('Enter the 2nd String')
str1='python program'
str2='string operations'
print 'String1 is: ',str1
print 'String2 is: ',str2

print 'Length of the strings: str1 and str2'
print len(str1)
print len(str2)

print 'First occurrences of o in str and r in str2 is:'
print str1.index("o")
print str2.index("r")

print 'Nummber of Occurrences in str1 and str2 are:'
print str1.count("o")
print str2.count("i")

print 'str1 string slice operations'
print str1[2:9]
print str1[2:9:2]
print str1[2:7]
print str1[2:9:1]
```
print str1[::-1]
print 'str2 string slice operations'
print str2[1:6]
print str2[2:8:2]
print str2[2:8]
print str2[2:8:1]
print str2[::-1]

print 'Print str1 and str2 in upper case'
print str1.upper()
print str2.upper()

print 'Print str1 and str2 in lower case'
print str1.lower()
print str2.lower()

print 'str1 and str2 string functions starts with'
print str1.startswith('python')
print str2.startswith('Hello')

print 'str1 and str2 string functions ends with'
print str1.endswith('kalasalingam')
print str2.endswith('operations')

print 'str1 and str2 split operations'
strsplit=str1.split(" ")
print strsplit
strsplit2=str2.split(" ")
print strsplit2

print 'String concatenation'
print str1+str2
Program to find duplicate characters in a given string

```python
#str='java'
str=raw_input('Enter the String\n')
first_time=[ ]
dup=[ ]
for i in str:
    if i not in first_time:
        first_time.append(i)
    else:
        if i not in dup:
            dup.append(i)
```
dup.append(i)
print ",".join(dup)

**Output**
Enter the String
java

---

**WORKING WITH ARRAYS IN PYTHON**

**AIM:**
To implement arrays using Python

**Program to create an array of 5 integers and display the array items. Access individual element through indexes.**

```python
from array import *
array_num = array('i', [1,3,5,7,9])
for i in array_num:
    print(i)
print("Access first three items individually")
print(array_num[0])
print(array_num[1])
print(array_num[2])
```

**Output**

1
3
5
7
9
Access first three items individually
1
3
5
Program to append a new item to the end of the array.

```python
from array import *
array_num = array('i', [1, 3, 5, 7, 9])
print("Original array: "+str(array_num))
print("Append 11 at the end of the array:")
array_num.append(11)
print("New array: "+str(array_num))
```

**Output**

Original array: array('i', [1, 3, 5, 7, 9])
Append 11 at the end of the array:
New array: array('i', [1, 3, 5, 7, 9, 11])

Program to reverse the order of the items in the array.

```python
from array import *
array_num = array('i', [1, 3, 5, 3, 7, 1, 9, 3])
print("Original array: "+str(array_num))
array_num.reverse()
print("Reverse the order of the items:")
print(str(array_num))
```

**Output**

Original array: array('i', [1, 3, 5, 3, 7, 1, 9, 3])
Reverse the order of the items:
array('i', [3, 9, 1, 7, 3, 5, 3, 1])

Program to insert a new item before the second element in an existing array

```python
from array import *
array_num = array('i', [1, 3, 5, 7, 9])
print("Original array: "+str(array_num))
print("Insert new value 4 before 3:")
array_num.insert(1, 4)
print("New array: "+str(array_num))
```

**Output**

Original array: array('i', [1, 3, 5, 7, 9])
Insert new value 4 before 3:
New array: array('i', [1, 4, 3, 5, 7, 9])
Output

Original array: array('i', [1, 3, 5, 7, 9])
Insert new value 4 before 3:
New array: array('i', [1, 4, 3, 5, 7, 9])

---

**USER DEFINED FUNCTION IN PYTHON**

**AIM:**
To implement user defined functions in python

**Python function to find the Max of three numbers.**

def max_of_two( x, y ):
    if x > y:
        return x
    return y
def max_of_three( x, y, z ):
    return max_of_two( x, max_of_two( y, z ) )
print(max_of_three(3, 6, -5))

**Output**
6

**Python function to sum all the numbers in a list.**

def sum(numbers):
    total = 0
    for x in numbers:
        total += x
    return total
print(sum((8, 2, 3, 0, 7)))

**Output**
20
Python function to check whether a number is in a given range.

def test_range(n):
    if n in range(3,9):
        print(" %s is in the range"%str(n))
    else :
        print("The number is outside the given range.")

test_range(5)

Output
5 is in the range

Python program to print the even numbers from a given list.

def is_even_num(l):
    enum = []
    for n in l:
        if n % 2 == 0:
            enum.append(n)
    return enum

print(is_even_num([1, 2, 3, 4, 5, 6, 7, 8, 9]))

Output
[2, 4, 6, 8]
Python function to calculate the factorial of a number (a non-negative integer). The function accepts the number as an argument.

```python
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

n=int(input("Input a number to compute the factorial : "))
print(factorial(n))
```

**Output**

Input a number to compute the factorial : 4
24
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
<?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){
```
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
<?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)){
            // Process the data, and then...
        }
    }
}

// Validate password
if(empty(trim($_POST["password"]))){
    $password_err = "Please enter a password.";
}
else{
    // Password must have at least six characters, and match the confirmed password
    if(strlen(trim($_POST["password"])) < 6){
        $password_err = "Password must have at least 6 characters";
    }
    elseif(trim($_POST["password"] !== trim($_POST["confirm_password"])){
        $password_err = "Password does not match";
    }
}

// Validate confirm password
if(empty(trim($_POST["confirm_password"]))){
    $confirm_password_err = "Please confirm password";
}
else{
    // Password must have at least six characters, and match the confirmed password
    if(strlen(trim($_POST["confirm_password"])) < 6){
        $confirm_password_err = "Password must have at least 6 characters";
    }
    elseif(trim($_POST["confirm_password"] !== trim($_POST["password"])){
        $confirm_password_err = "Password does not match";
    }
}

if($username_err || $password_err || $confirm_password_err){
    echo "Please correct the error and try again."
}
else{
    echo "Registration successful!"
}
?>
```
/* store result */
mysqli_stmt_store_result($stmt);

if(mysqli_stmt_num_rows($stmt) == 1){
    $username_err = "This username is already taken.";
} else{
    $username = trim($_POST["username"]);
}
}
}

// 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;
      $param_password = password_hash($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($link);
?>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Sign Up</title>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.css">
<style type="text/css">
body { font: 14px sans-serif; }
.wrapper { width: 350px; padding: 20px; }
</style>
</head>
<body>
<div class="wrapper">
<h2>Sign Up</h2>
<p>Please fill this form to create an account.</p>
<form action="<?php echo htmlspecialchars($_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>
</form>
</div>
</body>
</html>
<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>
<input type="submit" class="btn btn-primary" value="Submit">
<input type="reset" class="btn btn-default" value="Reset">
<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
<?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>
</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
<?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">
<body>
</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 intended 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
<?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 on Go.
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.

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:
<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"
<?php
}
// end if
?>
</body>
</html>

NOTES

Self-Instructional Material
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

} else {

// Code for handling form submission

} // End if ($submit)

} // End if ($id)

// Code for handling form submission

// Close MYSQL server connection
mysql_close($db);

// Close HTML tags

</body>
</html>
$result = mysql_query($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
{

133
// 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/");

NOTES
Self-Instructional Material

134
// 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?>">
            First name:<input type="Text" name="first" readonly="readonly" value="<?php echo $myrow["first"] ?>">
            Last name:<input type="Text" name="last" readonly="readonly" value="<?php echo $myrow["last"] ?>">
            Address:<input type="Text" name="address" readonly="readonly" value="<?php echo $myrow["address"] ?>">
            Position:<input type="Text" name="position" value="<?php echo $myrow["position"] ?>">
            <input type="Submit" name="submit" value="Delete information">
        </form>
    }
}

// NOTES
Self-Instructional
Material
<?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 mysql with 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:

```perl
$dbh = DBI->connect($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:

```perl
#!/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:

# without RaiseError 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 disconnect() method of the database handle object as follows:

```perl
# disconnect from the MySQL database
$dbh->disconnect();
```
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 −

```python
#!/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 −

```python
#!/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) \n    ('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 fetch 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 \n    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" \n            (fname, lname, age, sex, income )
except:
    print "Error: unable to fetch 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**

```python
#!/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
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 –

```python
# 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()
```

```python
db.commit()
except:
    # Rollback in case there is any error
db.rollback()
# disconnect from server
db.close()```
**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.