EX.NO:1

DDL COMMANDS

Aim:

To work with DDL commands.

Procedure:

The Data Definition Language (DDL) Commands are-

CREATE:

-It is used to create a new table or new database.

i)CREATE DATABASE:

Syntax :

CREATE DATABASE database_name;

Command :

CREATE DATABASE sfms;

Output:

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0545 seconds.)

```
CREATE DATABASE sfms
```

ii) CREATE TABLE:

Syntax:

CREATE TABLE tablename

(column_name1 datatype,

… …….,

column_name_n datatype);

Command:

```
CREATE TABLE `student` (
  `register` varchar(255) NOT NULL,
  `name` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  `gender` varchar(100) NOT NULL,
  `updationDate` date NOT NULL,
  `password` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Output:

register	name	email	gender	updationDate	password

Alter:

-The ALTER statement is used to add, delete or modify columns in an existing table.

i) ALTER TABLE-ADD COLUMN:

Syntax:

ALTER TABLE table_name

ADD column_name datatype;

Example:

ALTER TABLE student

ADD mobile int(11);

Output:

register	name	email	gender	updationDate	password	mobile

ii) ALTER TABLE-DROP COLUMN:

Syntax:

ALTER TABLE table_name

DROP COLUMN column_name;

Example:

ALTER TABLE student

DROP COLUMN mobile;

Output:

register	name	email	gender	updationDate	password

iii) ALTER TABLE - RENAME COLUMN:

Syntax:

```
ALTER TABLE table_name
CHANGE old_name   new_name DATA TYPE;
```

Example:

```
ALTER TABLE  student
CHANGE updationDate  regDate  DATE NOT NULL;
```

Output:

register	name	email	gender	regDate	password

iv) ALTER TABLE-MODIFY DATATYPE:

Syntax:

```
ALTER TABLE table_name
MODIFY column_name DATA TYPE;
```

Example:

```
ALTER TABLE  student
MODIFY regDate  varchar(255);
```

Output:

	#	Name	Type	Collation
☐	1	**register**	varchar(255)	latin1_swedish_ci
☐	2	**name**	varchar(255)	latin1_swedish_ci
☐	3	**email**	varchar(255)	latin1_swedish_ci
☐	4	**gender**	varchar(255)	latin1_swedish_ci
☐	5	**regDate**	date	
☐	6	**password**	varchar(255)	latin1_swedish_ci

	#	Name	Type	Collation
☐	1	**register**	varchar(255)	latin1_swedish_ci
☐	2	**name**	varchar(255)	latin1_swedish_ci
☐	3	**email**	varchar(255)	latin1_swedish_ci
☐	4	**gender**	varchar(255)	latin1_swedish_ci
☐	5	**regDate**	varchar(255)	latin1_swedish_ci
☐	6	**password**	varchar(255)	latin1_swedish_ci

Truncate:

-This command is used to remove all the rows from the table.

Syntax:

```
TRUNCATE TABLE table_name;
```

Example:

TRUNCATE TABLE student;

RENAME:

-It is used to rename the table from the database.

Syntax:

RENAME TABLE old_table_name TO new_table_name;

Example:

RENAME TABLE student TO details;

Output:

MySQL returned an empty result set (i.e. zero rows). (

```
SELECT * FROM `details`
```

register	name	email	gender	regDate	password

DROP:

-It is used to delete a table,index or view from database.

Syntax:

DROP TABLE table_name;

Example:

DROP TABLE details;

Output:

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0174 seconds.)

```
DROP TABLE details
```

Result:

The DDL COMMANDS are successfully executed and the output is verified.

Ex No:02

DML COMMANDS

Aim:

To work with DML commands

Procedure:

The Data Manipulation Language(DML) commands are-

Select - Retrieve data from the database.

Insert - Insert data into a table.

Update - Update existing data within a table.

Delete – Delete records from a database table.

Insert:

-Insert command is used to insert a data into a table.

Syntax:

INSERT INTO table_name(column_list)

VALUES (column_values);

Example:

INSERT INTO `student` (`register`, `name`, `email`, `gender`, `updationDate`, `password`)

VALUES('19cs0401', 'Raju', 'raju@gmail.com', 'male', curdate(), '121212');

Output:

register	name	email	gender	updationDate	password
19cs0401	Raju	raju@gmail.com	male	2023-03-10	121212

Update:

-Update command is used to update existing data with in a table.

Syntax:

UPDATE table_name SET column_name = column_value WHERE CONDITION;

Example:

UPDATE student SET password = '12345678' WHERE register='19cs0401';

Output:

register	name	email	gender	updationDate	password
19cs0401	Raju	raju@gmail.com	male	2023-03-10	12345678

SELECT:

-Select command is used to retrieve data from the database.

Syntax:

- SELECT * FROM table_name;
- SELECT column_name FROM table_name WHERE CONDITION;

EXAPMLE:

- SELECT * FROM student;

Output:

register	name	email	gender	updationDate	password
19cs0401	Raju	raju@gmail.com	male	2023-03-10	12345678

- SELECT name FROM student WHERE register=’19cs0401’;

Output:

name
Raju

DELETE:

-Delete command is used to delete a records from a database table.

Syntax:

DELETE FROM table_name WHERE CONDITION;

Example:

DELETE FROM student WHERE register='19cs0401';

Output:

1 row affected. (Query took 0.0119 seconds.)

```
DELETE FROM student WHERE register='19cs0401'
```

Result:

The DML COMMANDS are successfully executed and the output is verified.

Ex No:03

DCL COMMANDS

Aim:

To work with DCL commands.

Procedure:

The Data Control Language (DCL) are-

GRAND :

- It is used to grant a privilege to a user & allow to perform specific task.

Syntax:

GRANT [privilege_name or ALL] ON table_name TO ‘user’@’localhost’;

Example:

GRANT SELECT,UPDATE,DELETE,INSERT ON student TO ‘Raju’@’localhost’;

Output:

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0078 seconds.)

```
GRANT SELECT,UPDATE,DELETE,INSERT ON student TO 'Raju'@'%'
```

REVOKE:

- It is used to remove granted privilege from a user.

Syntax:

REVOKE [privilege_name or ALL] ON table_name FROM user;

Example:

REVOKE SELECT,UPDATE,DELETE,INSERT ON student FROM 'Raju'@'%';

Output:

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0043 seconds.)

```
REVOKE SELECT,UPDATE,DELETE,INSERT ON student FROM 'Raju'@'%'
```

Result:

The DCL COMMANDS are successfully executed and the output is verified.

EX.NO:4

SUB QUERY

Aim:

To work with My SQL Sub Query.

Procedure:

Sub Query can be simply defined as a query within another query. In other words we can say that a subquery is a query that is embedded in where clause of another query.

SELECT:

-Sub Query in select statement(command) .

Syntax :

SELECT column_name FROM table_name

WHERE (SELECT column_name FROM table_name WHERE CONDITION);

Example :

SELECT * FROM student WHERE (SELECT reg_no FROM profile_log WHERE id=3);

Output:

register	name	email	gender	password
19cs0401	Raju	raju@gmail.com	male	121212

INSERT:

-Sub Query in insert command.

Syntax:

- Insert into table1 select*From table2;
- Insert into table1 select*From table2 where condition;

Example:

```
INSERT INTO `student` SELECT * FROM request where register='qwe123';
```

Output:

1 row inserted. (Query took 0.0113 seconds.)

```
INSERT INTO `student` SELECT * FROM request where register='qwe123'
```

UPDATE:

-Sub Query in update command.

Syntax:

UPDATE table_name SET column_Name = Value

WHERE column_name IN (SELECT column_name FROM table2 WHERE CONDITION);

Example:

UPDATE student SET `password` ='abcd'

WHERE register IN (SELECT reg_no FROM profile_log WHERE id=3);

Output:

register	name	email	gender	password
19cs0439	Raju	raju@gmail.com	male	abcd

DELETE:

-Sub Query in delete command.

Syntax:

Delete from table where column_name in (select column_name from table2 where CONDITION)

Example:

DELETE FROM `student` WHERE register IN(SELECT reg_no FROM profile_log WHERE id=3);

Output:

register	name	email	gender	password

Check all *With selected:* Edit Copy Delete Export

Result:

The SUB QUERY COMMANDS are successfully executed and the output is verified.

EX.NO:4

SQL JOINS

Aim:

To work with MySQL JOINS.

Procedure:

A join clause is used to combine rows from two or more tables, based on a related column between them.

INNER JOIN:

-this keyword selects records that have matching values in both tables.

Syntax :

Select column name(s) from table;

Inner Join table2

ON table1.column_name= table2.column_name;

Example :

SELECT student.register,student.name,student.email,profile_log.type FROM student

INNER JOIN profile_log on

student.register=profile_log.reg_no;

Output:

register	name	email	type
19cs0439	Raju	raju@gmail.com	profile created
19cs0439	Raju	raju@gmail.com	profile updated
19cs0439	Raju	raju@gmail.com	profile deleted

LEFT JOIN:

-returns all records from the left table and matching records(if any) from the right table.

Syntax:

SELECT column_name(s) From table1;

LEFT JOIN table2 ON table1. Column_name = table2.column_name;

Example:

```
SELECT student.register,student.name,student.email,profile_log.type
FROM student LEFT JOIN profile_log on
student.register=profile_log.reg_no;
```

Output:

register	name	email	type
19cs0439	Raju	raju@gmail.com	profile created
19cs0439	Raju	raju@gmail.com	profile updated
19cs0439	Raju	raju@gmail.com	profile deleted
19cs0401	sam	sam@gmail.com	NULL

RIGHT JOIN:

-returns all records from the right table and matching records (if any) from the left table.

Syntax:

SELECT column_name(s) From table1;

RIGHT JOIN table2 ON table1. Column_name = table2.column_name;

Example:

SELECT student.register,student.name,student.email,profile_log.type FROM student

RIGHT JOIN profile_log on student.register=profile_log.reg_no;

Output:

register	name	email	type
19cs0439	Raju	raju@gmail.com	profile created
19cs0439	Raju	raju@gmail.com	profile updated
19cs0439	Raju	raju@gmail.com	profile deleted
NULL	NULL	NULL	profile created
NULL	NULL	NULL	profile deleted

CROSS JOIN:

-returns all records from both tables.

Syntax:

Select column_name(s)

From table1.cross join table2;

Example:

SELECT student.register,student.name,student.email,profile_log.type FROM student

CROSS JOIN profile_log ;

Output:

register	name	email	type
19cs0401	sam	sam@gmail.com	profile created
19cs0439	Raju	raju@gmail.com	profile created
19cs0401	sam	sam@gmail.com	profile updated
19cs0439	Raju	raju@gmail.com	profile updated
19cs0401	sam	sam@gmail.com	profile deleted
19cs0439	Raju	raju@gmail.com	profile deleted
19cs0401	sam	sam@gmail.com	profile created
19cs0439	Raju	raju@gmail.com	profile created
19cs0401	sam	sam@gmail.com	profile deleted
19cs0439	Raju	raju@gmail.com	profile deleted

Syntax 2:

Select column_name(s)

From table1 cross join table2

Where condition;[if using where act as a inner join];

Example:

SELECT student.register,student.name,student.email,profile_log.type FROM student

CROSS JOIN profile_log on student.register=profile_log.reg_no;

Output:

register	name	email	type
19cs0439	Raju	raju@gmail.com	profile created
19cs0439	Raju	raju@gmail.com	profile updated
19cs0439	Raju	raju@gmail.com	profile deleted

Result:

The SQL JOINS are successfully executed and the output is verified.

Ex No:05

PL/SQL

Aim:

To work with PL/SQL

Procedure:

- PL/SQL is a combination of SQL along with the procedural features of programming languages.
- PL/SQL is one of three key programming languages embedded in the Oracle Database, along with SQL itself and Java.
- The PL/SQL programs are divided and written in logical blocks of code. Each block consists of three sub-parts,
    - Declare
    - Begin
    - Exception
    - End

SYNTAX:

```
DECLARE

<declarations section>

BEGIN

<executable  section commands>

EXCEPTION

<exception handling>

END;
```

EXAMPLE:

1.Addition of Two Numbers:

Program Code:

```
declare
x number(5);
y number(5);
z number(5);
begin
x:=50;
y:=20;
z:=x+y;
dbms_output.put_line('sum is'||z);
end;
/
```

Output for example-1:

```
1   declare
2   x number(5);
3   y number(5);
4   z number(7);
5   begin
6   -- Here we Assigning 10 into x
7   x:=10;
8   -- Assigning 20 into x
9   y:=20;
10  -- Assigning sum of x and y into z
11  z:=x+y;
12  -- Print the Result
13  dbms_output.put_line('Sum is '||z);
14  end;
15  /
16
17
18
```

Results | Explain | Describe | Saved SQL | History

```
Sum is 30

Statement processed.
```

2.Generating Series:

Program Code:

```
declare
n number(5);
begin
n:=1;
for i in 1..10 loop
case n
when 1 then
dbms_output.put_line(i);
when 2 then
if mod(i,2)=0 then
dbms_output.put_line(i);
end if;
when 3 then
if mod(i,2)!=0 then
dbms_output.put_line(i);
end if;
end case;
end loop;
end;
/
```

Output for example-2:

```
1   declare
2   n number(5);
3   begin
4   n:=1;
5   for i in 1..10 loop
6   case n
7   when 1 then
8   dbms_output.put_line(i);
9   when 2 then
10  if mod(i,2)=0 then
11  dbms_output.put_line(i);
12  end if;
13  when 3 then
14  if mod(i,2)!=0 then
15  dbms_output.put_line(i);
16  end if;
17  end case;
```

Results | Explain | Describe | Saved SQL | History

```
1
2
3
4
5
6
7
8
9
10
```

Result: The PL/SQL program is successfully executed and the output is verified.

EX No:06

Cursors, Procedures and Functions

Aim:

To work with the Cursor, Procedure and Function in PL/SQL

Procedure:

Step 1: Create a necessary table for perform action .

Step 2: Then work with database using PL/SQL Block commands

1.CURSOR:

Syntax:

```
DECLARE cursor_name CURSOR
FOR select_statement;
OPEN cursor_name;
FETCH NEXT FROM cursor INTO variable_list;
CLOSE cursor_name;
```

2.PROCEDURE:

Syntax:

```
DECLARE

CREATE PROCEDURE procedure_name[[[IN]|OUT|INOUT]parameter_name
datatype[.parameter_datatype]

AS

Declaration_section

BEGIN

Executable_section
END;
BEGIN
[procedure_name]();
END;
```

3.FUNCTION:

Syntax:

```
CREATE [OR REPLACE] FUNCTION function_name [parameters]
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
RETURN return_datatype
{IS | AS}
```

```
        BEGIN
      < function_body >
        END [function_name];
```

Example:

```
DECLARE
  n number ;
  t number;
  s_id student.std_id%type;
  s_name student.name%type;
  s_cgpa student.cgpa%type;
  CURSOR s_student is                    --cursor
    SELECT std_id, name, cgpa FROM student;
PROCEDURE pro AS                         --procedure
BEGIN
  OPEN s_student;
  LOOP
  FETCH s_student into s_id, s_name, s_cgpa;
    EXIT WHEN s_student%notfound;
    s_cgpa:=s_cgpa*10;
    dbms_output.put_line('Id:' || s_id || '    Name:' || s_name || '    Percentage:' || s_cgpa);
  END LOOP;
  CLOSE s_student;
END;
FUNCTION func                      -- function
RETURN number IS
  total number(2) := 0;
BEGIN
  SELECT count(*) into total
  FROM student;
   RETURN total;
```

```
END;
BEGIN          --main program
n:=1;
case n
   when 1 then
     PRO();
   when 2 then
       t:=func();
       dbms_output.put_line(t);
END case;
END;
/
```

Output:

when n=1:

```
 9
10  DECLARE
11     n number ;
12     t number;
13     s_id student.std_id%type;
14     s_name student.name%type;
15     s_cgpa student.cgpa%type;
16     CURSOR s_student is
17        SELECT std_id, name, cgpa FROM student;
18  PROCEDURE pro AS
```

Results | Explain | Describe | Saved SQL | History

```
Id:2    Name:abc    Percentage:79
Id:3    Name:dca    Percentage:77
Id:1    Name:raju    Percentage:68
```

Output:

when n=2

```
BEGIN
n:=2;
case n
    when 1 then
      PRO();
    when 2 then
        t:=func();
        dbms_output.put_line(t);
END case;
END;
```

Results | Explain | Describe | Saved SQL | History

```
3
```

Result:

The cursor, procedure and function in PL/SQL is successfully executed and the output is verified.

Ex No:07

TRIGGERS

Aim:

To work with TRIGGER in mySQL

Procedure:

Syntax:

CREATE [DEFINER = { user | CURRENT_USER }]

TRIGGER trigger_name

trigger_time trigger_event ON tbl_name

FOR EACH ROW

trigger_body

- trigger_time: { BEFORE | AFTER }
- trigger_event: { INSERT | UPDATE | DELETE }

Example:

1) Trigger for insert command.

Trigger Command:

```
DROP TRIGGER IF EXISTS `insert`;

CREATE DEFINER=`root`@`localhost` TRIGGER `insert`

AFTER INSERT ON `student`

FOR EACH ROW

INSERT into profile_log VALUES(null,new.register,'profile created',curdate());
```

SQL command:

```
INSERT INTO `student` (`register`, `name`, `email`, `gender`, `updationDate`, `password`) VALUES
('19cs0439', 'Raju', 'rajuraj@gmail.com', 'male', CURRENT_DATE(), 'Raju@1234');
```

Table:student

register	name	email	gender	password
19cs0439	Raju	Raju@gmail.com	male	12345678

Edit Copy Delete Check all *With selected:* Edit Copy Delete Export

Table:profile_log

+ Options

id	reg_no	type	time
1	19cs0439	profile created	2023-03-04

Edit Copy Delete Check all *With selected:* Edit Copy Delete Export

2) Trigger for update command.

Trigger Command:

```
DROP TRIGGER IF EXISTS `update`;

CREATE DEFINER=`root`@`localhost` TRIGGER `update`

AFTER UPDATE ON `student`

FOR EACH ROW

INSERT into profile_log VALUES(null,new.register,'profile updated',curdate());
```

SQL command:

```
UPDATE `student` SET `email`='Raju@gmail.com',`password`=12345678 WHERE register='19cs0439';
```

Table:student

register	name	email	gender	password
19cs0439	Raju	Raju@gmail.com	male	12345678

Edit Copy Delete Check all *With selected:* Edit Copy Delete Export

Table:profile_log

			id	reg_no	type	time
Edit	Copy	Delete	1	19cs0439	profile created	2023-03-04
Edit	Copy	Delete	2	19cs0439	profile updated	2023-03-04

3) Trigger for delete command

Trigger Command:

```
DROP TRIGGER IF EXISTS `delete`;

CREATE DEFINER=`root`@`localhost` TRIGGER `delete`

BEFORE DELETE ON `student`

FOR EACH ROW

 INSERT into profile_log VALUES(null,old.register,'profile deleted',curdate());
```

SQL command:

Delete from `student` where register='19cs0439';

Table:student

register	name	email	gender	password

Table:profile_log

			id	reg_no	type	time
Edit	Copy	Delete	1	19cs0439	profile created	2023-03-04
Edit	Copy	Delete	2	19cs0439	profile updated	2023-03-04
Edit	Copy	Delete	3	19cs0439	profile deleted	2023-03-05

Result:

The Triggers in SQL is successfully executed and the output is noted.

Ex.No:8

STUDENT'S FILE MANAGEMENT SYSTEM

Aim:

To create a Web Application for STUDENT'S FILE MANAGEMENT SYSTEM(SFMS)

ABSTRACT:

The Student's File Management System (SFMS) is a web application designed to help students manage their academic files and assignments online. It simplifies file management and enhances productivity by eliminating the need for physical storage devices.

The SFMS provides a secure centralized location for students to store their academic files and assignments, reducing the risk of data loss or corruption. It is also highly customizable and scalable to meet the specific needs of educational institutions, departments, and individual students.

This paper presents the development and implementation of the SFMS as a web application, highlighting its key features and benefits to students. We also discuss the challenges and opportunities associated with deploying and managing a web-based file management system in an educational environment.

MODULES:

The entire project mainly consists of **2** modules, which are

1. Admin module.
2. Student module.

Software Requirements:

- Operating system – Windows 10
- Web Server : XAMPP [5.6.4]
- Database : MYSQL [5.0.21]
- Coding Language : Web Tech (HTML, CSS, Java Script, PHP)

LIMITATION OF EXISTING SYSTEM

- Manual work.
- Time consuming.
- Lack of reliability.

Database Structure:

Database **:** sfms

Table1 : admin

#	Name	Type	Collation
1	**id**	int(11)	
2	**username**	varchar(255)	latin1_swedish_ci
3	**password**	varchar(255)	latin1_swedish_ci
4	**updationDate**	varchar(255)	latin1_swedish_ci

Table2 : request

#	Name	Type	Collation
1	**register**	varchar(255)	latin1_swedish_ci
2	**name**	varchar(255)	latin1_swedish_ci
3	**email**	varchar(255)	latin1_swedish_ci
4	**gender**	varchar(255)	latin1_swedish_ci
5	**requestDate**	date	
6	**password**	varchar(255)	latin1_swedish_ci

Table3 : student

#	Name	Type	Collation
1	**register**	varchar(255)	latin1_swedish_ci
2	**name**	varchar(255)	latin1_swedish_ci
3	**email**	varchar(255)	latin1_swedish_ci
4	**gender**	varchar(100)	latin1_swedish_ci
5	**updationDate**	date	
6	**password**	varchar(255)	latin1_swedish_ci

Code:

1)Config.php

```
<?php

define('DB_SERVER','localhost');

define('DB_USER','root');

define('DB_PASS' ,'');

define('DB_NAME', 'sfms');

$con = mysqli_connect(DB_SERVER,DB_USER,DB_PASS,DB_NAME);

// Check connection

if (mysqli_connect_errno())

{

 echo "Failed to connect to MySQL: " . mysqli_connect_error();

}

?>
```

2)registration.php

```
<?php

include_once('include/config.php');

if(isset($_POST['submit']))

{

$name=$_POST['name'];

$register=$_POST['register'];

$gender=$_POST['gender'];

$email=$_POST['email'];

$password=$_POST['password'];

$query=mysqli_query($con,"insert into request(name,register,email,gender,password,requestDate)
values('$name','$register','$email','$gender','$password',curdate())");


if($query)
```

```
{           echo "<script>alert('Successfully Requested. You can login after few minutes ');</script>";

            //header('location:index.php');

}

else {

echo "<script>alert('request unsuccessfully');</script>";

}

}

?>
```

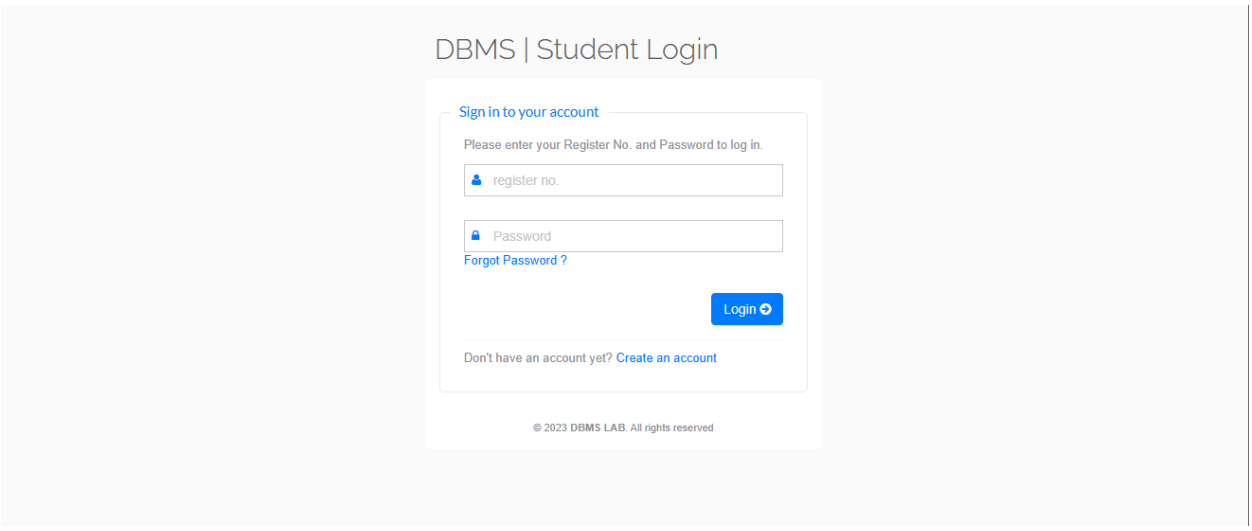
3)user-login.php

```
<?php

session_start();

error_reporting(0);

include("include/config.php");

if(isset($_POST['submit']))

{

$ret=mysqli_query($con,"SELECT * FROM student WHERE register='".$_POST['register']."' and
password='".$_POST['password']."'");

$num=mysqli_fetch_array($ret);

if($num>0)

{

$extra="dashboard.php";//

$_SESSION['login']=$_POST['register'];

$_SESSION['id']=$num['id'];

$host=$_SERVER['HTTP_HOST'];

$uip=$_SERVER['REMOTE_ADDR'];

$status=1;

// For stroing log if user login successfull
```

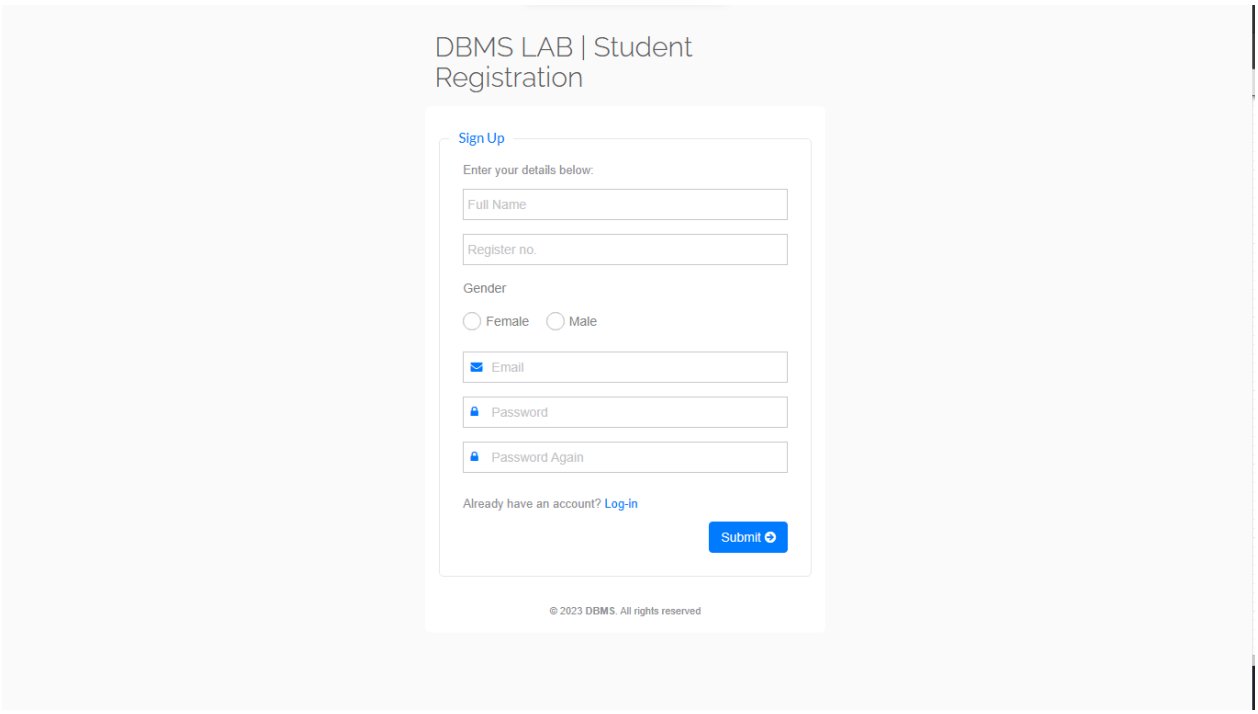
```
$log=mysqli_query($con,"insert into userlog(uid,username,userip,status)
values('".$_SESSION['id']."','".$_SESSION['login']."','$uip','$status')");

$uri=rtrim(dirname($_SERVER['PHP_SELF']),'/\\');

header("location:http://$host$uri/$extra");

exit();

}

else

{           // For stroing log if user login unsuccessfull

$_SESSION['login']=$_POST['register'];

$uip=$_SERVER['REMOTE_ADDR'];

$status=0;

mysqli_query($con,"insert into userlog(username,userip,status)
values('".$_SESSION['login']."','$uip','$status')");

$_SESSION['errmsg']="Invalid username or password";

$extra="index.php";

$host  = $_SERVER['HTTP_HOST'];

$uri  = rtrim(dirname($_SERVER['PHP_SELF']),'/\\');

header("location:http://$host$uri/$extra");

exit();

}

}

?>
```

Output:

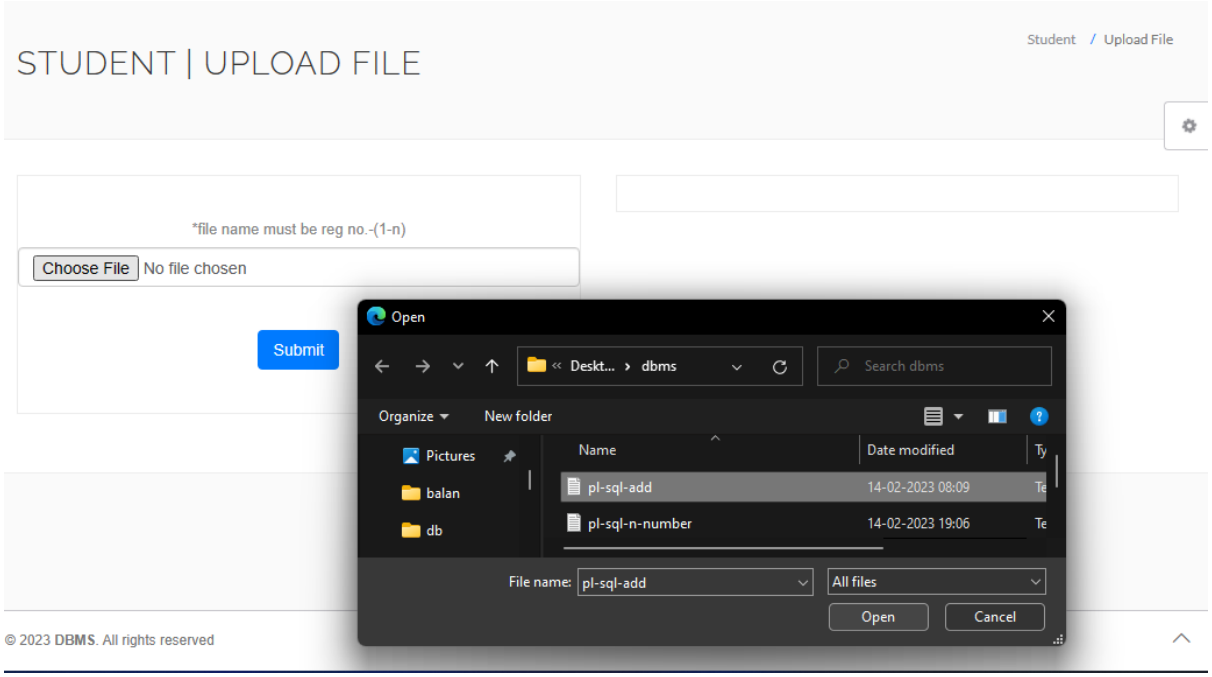
1)Student Login:



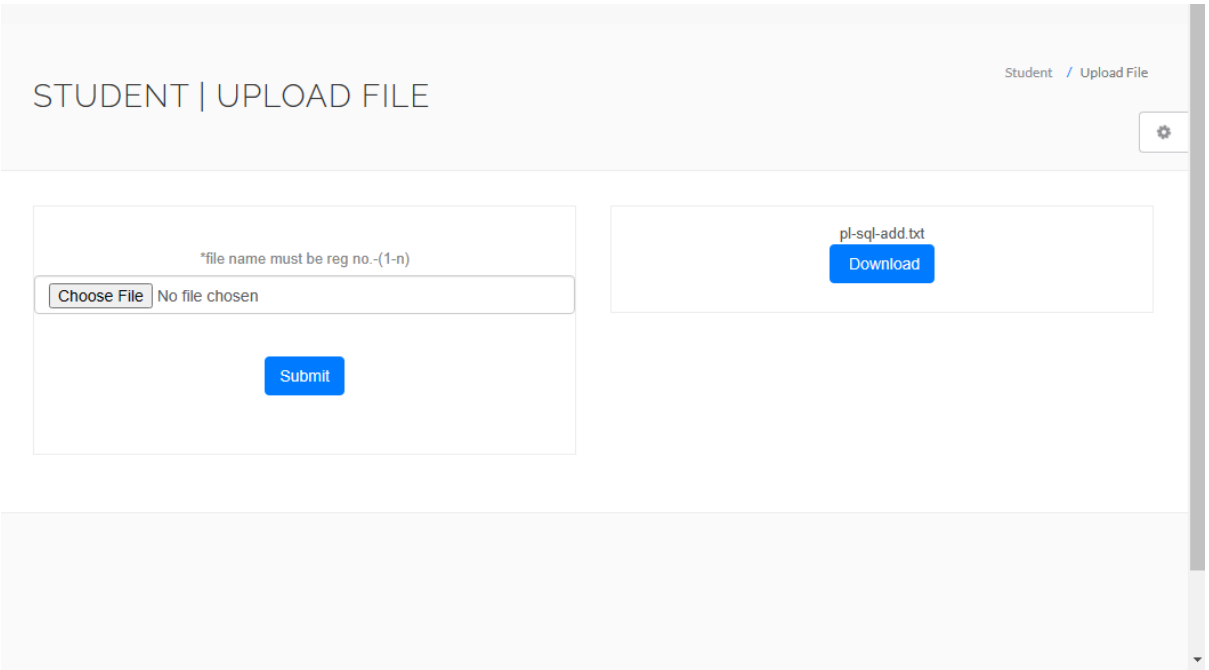
2)Student Request:



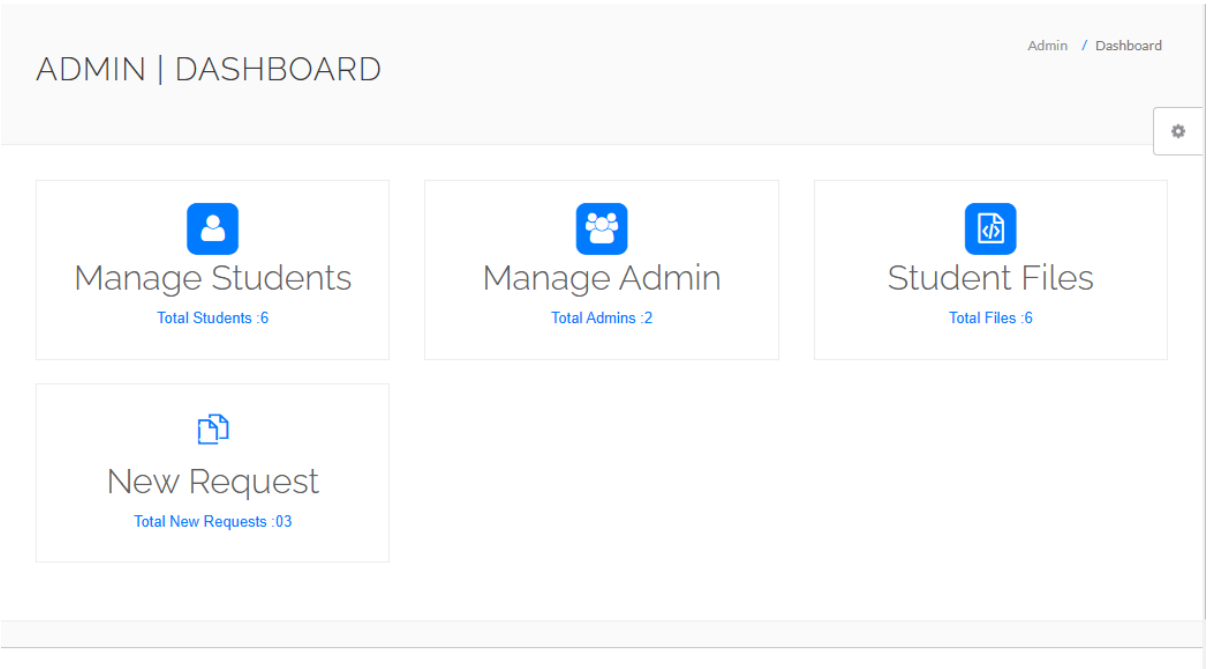
3)Student file upload:



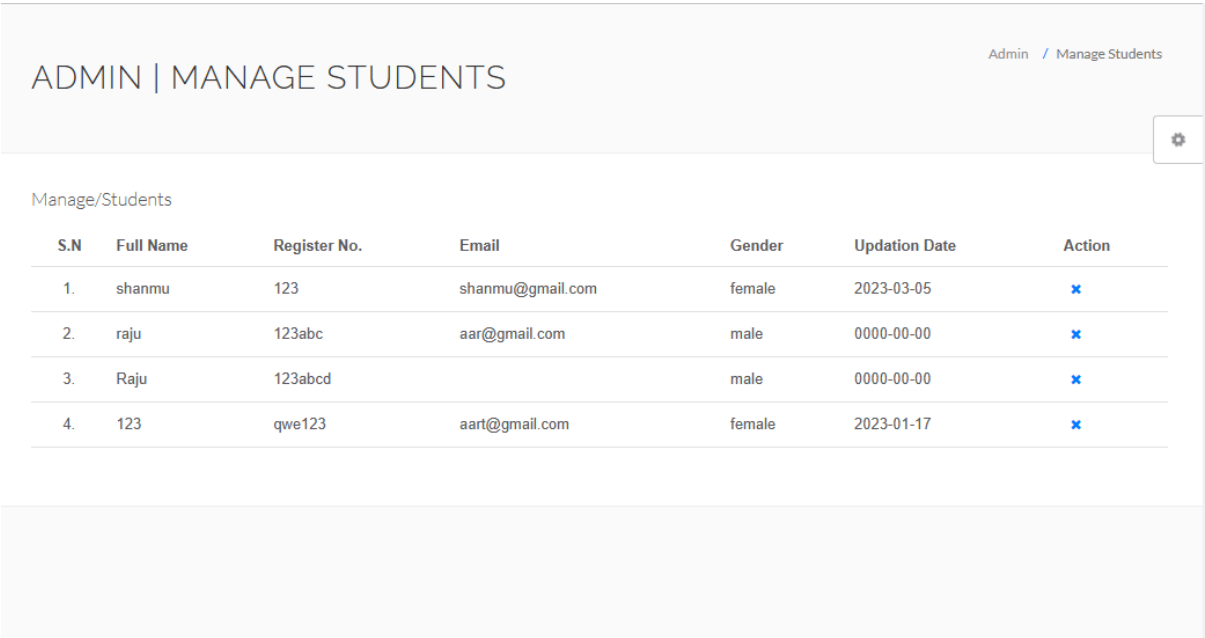
4)Student File view:



5)Admin Dashboard:

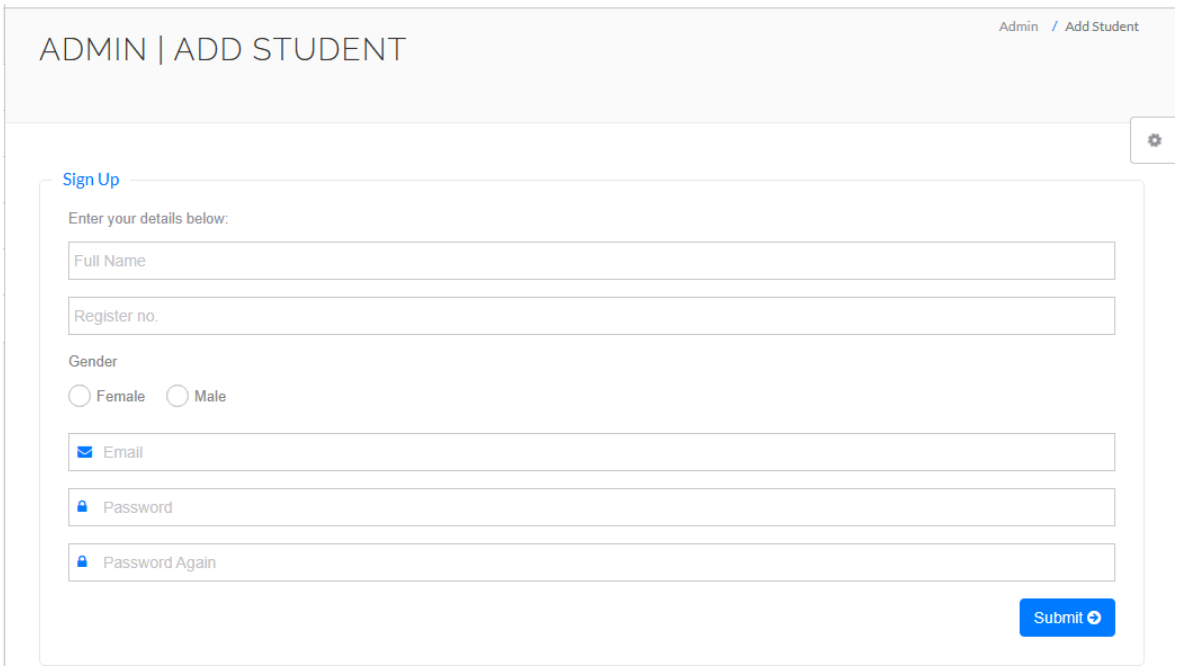


6)Admin Manage(delete) Students:



S.N	Full Name	Register No.	Email	Gender	Updation Date	Action
1.	shanmu	123	shanmu@gmail.com	female	2023-03-05	×
2.	raju	123abc	aar@gmail.com	male	0000-00-00	×
3.	Raju	123abcd		male	0000-00-00	×
4.	123	qwe123	aart@gmail.com	female	2023-01-17	×

7)Admin Add Student:



8)Admin View Students Files: