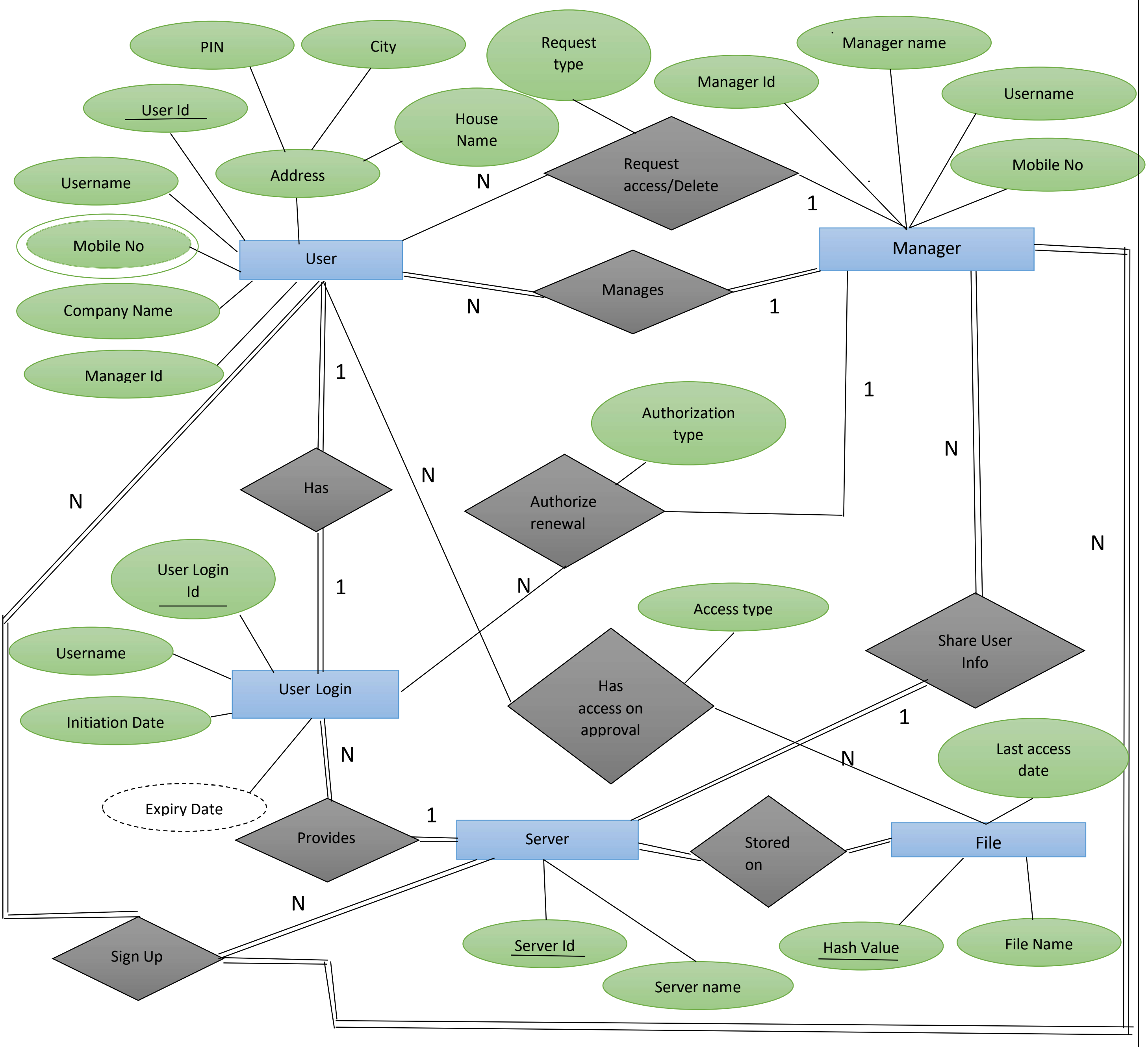
ROSE S JOSE

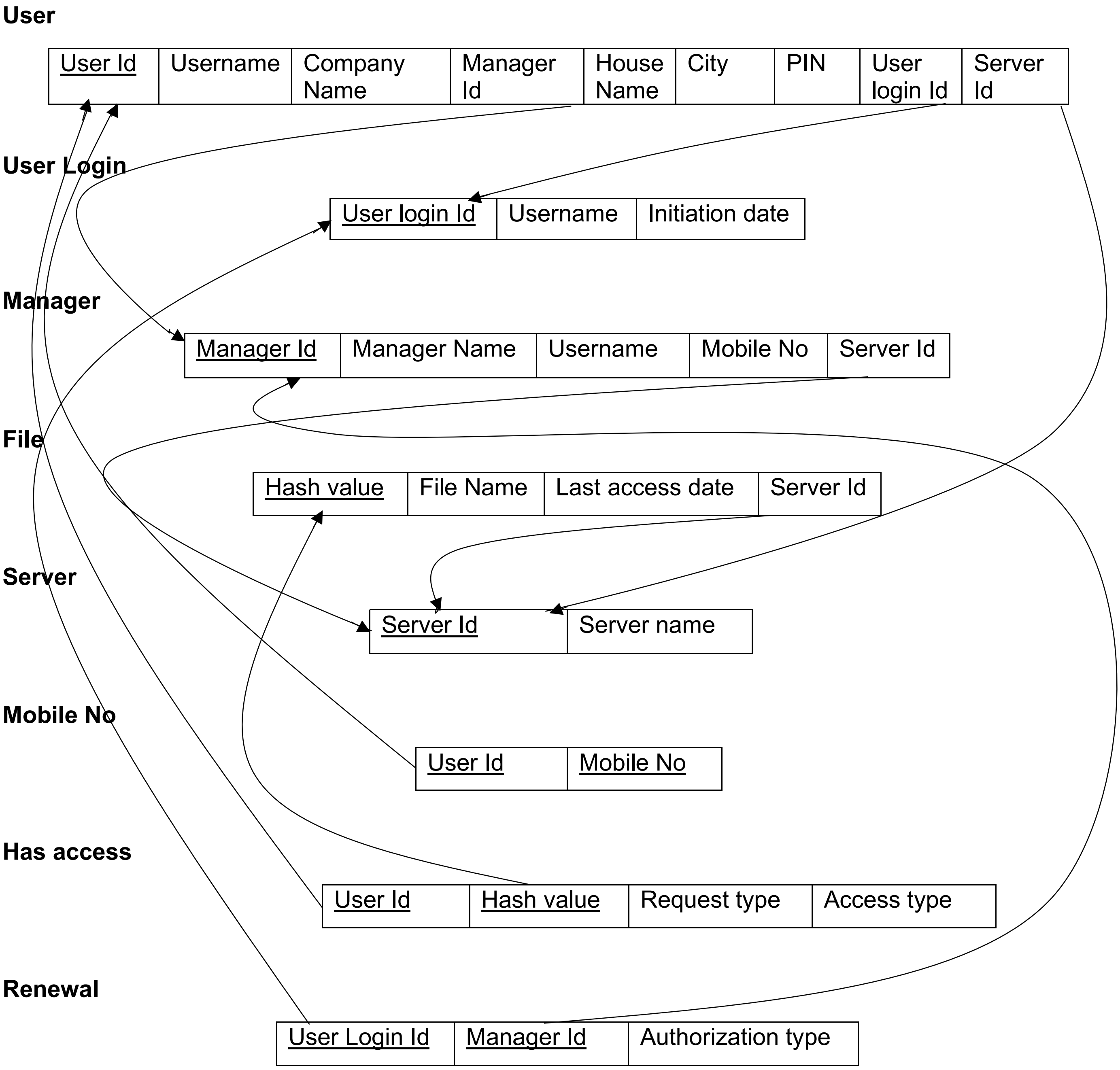
B190839CS

DBMS ASSIGNMENT 3 & 4

ER DIAGRAM:



RELATIONAL MODEL



User

User Id	Username	Company Name	Manager Id	House Name	City	PIN	User login Id	Server Id

User Login

User login Id	Username	Initiation date

Manager

Manager Id	Manager Name	Username	Mobile No	Server Id

File

Hash value	File Name	Last access date	Server Id

Server

Server Id	Server name

Mobile No

User Id	Mobile No

Has access

User Id	Hash value	Request type	Access type

Renewal

User Login Id	Manager Id	Authorization type

CREATING TABLES

- Servers

```
CREATE TABLE servers ( serverid INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
 servername VARCHAR(30));
```

- Manager

```
CREATE TABLE Manager (
    ManagerId INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ManagerName VARCHAR(50) NOT NULL,
    Username VARCHAR(20) NOT NULL,
    MobileNo VARCHAR(10),
    ServerId INT,
    FOREIGN KEY(ServerId) REFERENCES servers(serverid)
    );
```

- UserLogin

```
CREATE TABLE UserLogin(
    UserLoginId INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    Username VARCHAR(50) NOT NULL,
    InitiationDate DATE
    );
```

- File

```
CREATE TABLE file (
    HashValue INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    FileName VARCHAR(50),
    LastAccessDate Date,
    ServerId INT,
    FOREIGN KEY (ServerId) REFERENCES servers(serverid)
    );
```

- Users

```
CREATE TABLE users (
    UserId INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    Username VARCHAR(30) NOT NULL,
    CompanyName VARCHAR(50),
    ManagerId INT,
    HouseName VARCHAR(50),
    City VARCHAR(50),
    PIN VARCHAR(6),
    UserLoginId INT,
    ServerId INT,
    FOREIGN KEY (ServerId) REFERENCES servers(serverid),
    FOREIGN KEY (UserLoginId) REFERENCES userlogin(UserLoginId)
    );
```

- Hasaccess

```
CREATE TABLE hasaccess(
    UserId INT NOT NULL,
    HashValue INT NOT NULL,
    RequestType VARCHAR(30),
    AccessType VARCHAR(30),
    FOREIGN KEY (UserId) REFERENCES users(UserId),
    FOREIGN KEY (HashValue) REFERENCES file(HashValue),
    PRIMARY KEY (UserId, HashValue)
    );
```

- Renewal

```
CREATE TABLE renewal(
    UserLoginId INT NOT NULL,
    ManagerId INT NOT NULL,
    AuthorizationType VARCHAR(30),
    FOREIGN KEY (UserLoginId) REFERENCES userlogin(UserLoginId),
    FOREIGN KEY (ManagerId) REFERENCES manager(ManagerId),
    PRIMARY KEY (UserLoginId, ManagerId)
    );
```

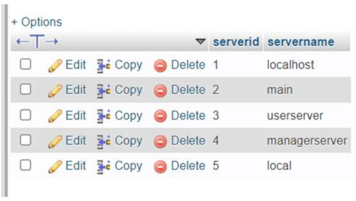
- Mobileno

```
CREATE TABLE mobileno (
    UserId INT NOT NULL,
    MobileNo VARCHAR(10),
    PRIMARY KEY (UserId, MobileNo),
    FOREIGN KEY (UserId) REFERENCES users (UserId)
    );
```

INSERTING TUPLES

- Servers

```
INSERT INTO servers(servername)

VALUES ('localhost');

INSERT INTO servers(servername)

VALUES ('main');

INSERT INTO servers(servername)

VALUES ('userserver');

INSERT INTO servers(servername)

VALUES ('managerserver');

INSERT INTO servers(servername)

VALUES ('local');
```



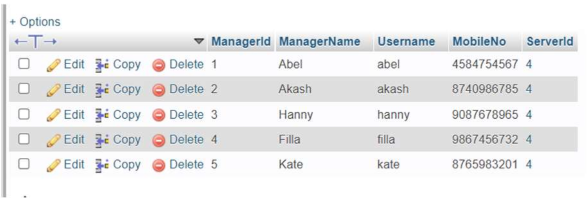
+ Options

←T→	serverid	servername
□ Edit Copy Delete	1	localhost
□ Edit Copy Delete	2	main
□ Edit Copy Delete	3	userserver
□ Edit Copy Delete	4	managerserver
□ Edit Copy Delete	5	local

- Manager

```
INSERT INTO manager( ManagerName, UserName, MobileNo, ServerId)
VALUES ('Abel', 'abel', '4584754567', 4);
INSERT INTO manager( ManagerName, UserName, MobileNo, ServerId)
VALUES ('Akash', 'akash', '8740986785', 4);
```

```
INSERT INTO manager( ManagerName, UserName, MobileNo, ServerId)
VALUES ('Hanny', 'hanny', '9087678965', 4);
INSERT INTO manager( ManagerName, UserName, MobileNo, ServerId)
VALUES ('Filla', 'filla', '9867456732', 4);
INSERT INTO manager( ManagerName, UserName, MobileNo, ServerId)
VALUES ('Kate', 'kate', '8765983201', 4);
```

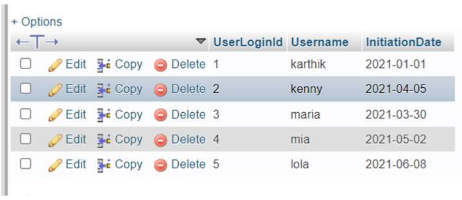


+ Options

←T→			▼	ManagerId	ManagerName	Username	MobileNo	ServerId
□	Edit	Copy	Delete	1	Abel	abel	4584754567	4
□	Edit	Copy	Delete	2	Akash	akash	8740986785	4
□	Edit	Copy	Delete	3	Hanny	hanny	9087678965	4
□	Edit	Copy	Delete	4	Filla	filla	9867456732	4
□	Edit	Copy	Delete	5	Kate	kate	8765983201	4

- UserLogin

```
INSERT INTO userlogin(UserName, InitiationDate)

VALUES ('karthik', '2021-01-01');

INSERT INTO userlogin(UserName, InitiationDate)

VALUES ('kenny', '2021-04-05');

INSERT INTO userlogin(UserName, InitiationDate)

VALUES ('maria', '2021-03-30');

INSERT INTO userlogin(UserName, InitiationDate)

VALUES ('mia', '2021-05-02');

INSERT INTO userlogin(UserName, InitiationDate)

VALUES ('lola', '2021-06-08');
```

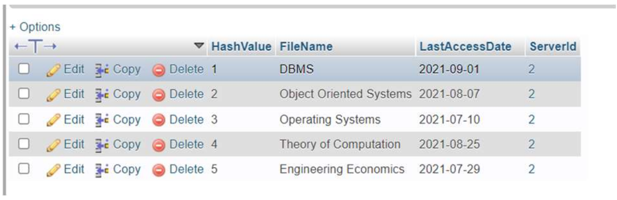


+ Options

←T→			▼	UserLoginId	Username	InitiationDate
□	Edit	Copy	Delete	1	karthik	2021-01-01
□	Edit	Copy	Delete	2	kenny	2021-04-05
□	Edit	Copy	Delete	3	maria	2021-03-30
□	Edit	Copy	Delete	4	mia	2021-05-02
□	Edit	Copy	Delete	5	lola	2021-06-08

- File

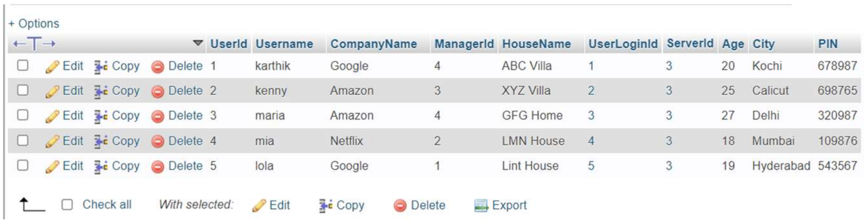
```
INSERT INTO file(FileName, LastAccessDate, ServerId)
VALUES ('DBMS', '2021-09-01',2);
INSERT INTO file(FileName, LastAccessDate, ServerId)
VALUES ('Object Oriented Systems', '2021-08-07',2);
INSERT INTO file(FileName, LastAccessDate, ServerId)
VALUES ('Operating Systems', '2021-07-10',2);
INSERT INTO file(FileName, LastAccessDate, ServerId)
VALUES ('Theory of Computation', '2021-08-25',2);
INSERT INTO file(FileName, LastAccessDate, ServerId)
VALUES ('Engineering Economics', '2021-07-29',2);
```



	HashValue	FileName	LastAccessDate	ServerId
□ Edit Copy Delete	1	DBMS	2021-09-01	2
□ Edit Copy Delete	2	Object Oriented Systems	2021-08-07	2
□ Edit Copy Delete	3	Operating Systems	2021-07-10	2
□ Edit Copy Delete	4	Theory of Computation	2021-08-25	2
□ Edit Copy Delete	5	Engineering Economics	2021-07-29	2

- Users

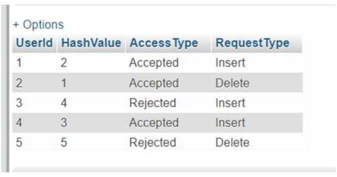
```
INSERT INTO users(Username, CompanyName, ManagerId, HouseName, UserLoginId, ServerId, City, PIN )

VALUES ('karthik', 'Google', 4, 'ABC Villa', 1, 3, ‘Kochi’, 678987);

INSERT INTO users(Username, CompanyName, ManagerId, HouseName, UserLoginId, ServerId, City, PIN)

VALUES ('kenny', 'Amazon', 3, 'XYZ Villa', 2, 3, “Calicut”, 698765);

INSERT INTO users(Username, CompanyName, ManagerId, HouseName, UserLoginId, ServerId, City, PIN)

VALUES ('maria', '8765498765', 'Amazon', 4, 'GFG Home', 3, 3, ‘Delhi’, 320987);

INSERT INTO users(Username, CompanyName, ManagerId, HouseName, UserLoginId, ServerId, City, PIN)

VALUES ('mia', '8768907654', 'Netflix', 2, 'LMN House', 4, 3, ‘Mumbai’, 109876);

INSERT INTO users(Username, CompanyName, ManagerId, HouseName, UserLoginId, ServerId, City, PIN)

VALUES ('lola', '7869765476', 'Google', 1, 'Lint House', 5, 3, ‘Hyderabad’, 543567);
```



	UserId	Username	CompanyName	ManagerId	HouseName	UserLoginId	ServerId	Age	City	PIN
□ Edit Copy Delete	1	karthik	Google	4	ABC Villa	1	3	20	Kochi	678987
□ Edit Copy Delete	2	kenny	Amazon	3	XYZ Villa	2	3	25	Calicut	698765
□ Edit Copy Delete	3	maria	Amazon	4	GFG Home	3	3	27	Delhi	320987
□ Edit Copy Delete	4	mia	Netflix	2	LMN House	4	3	18	Mumbai	109876
□ Edit Copy Delete	5	lola	Google	1	Lint House	5	3	19	Hyderabad	543567

- Hasaccess

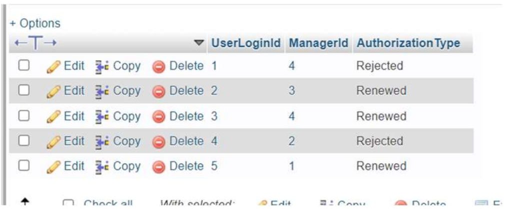
```
INSERT INTO hasaccess(UserId, HashValue, RequestType, AccessType)

VALUES (1, 2, ‘Insert’, 'Accepted');

INSERT INTO hasaccess(UserId, HashValue, RequestType, AccessType)

VALUES (2, 1, ‘Delete’, 'Accepted');

INSERT INTO hasaccess(UserId, HashValue, RequestType, AccessType)

VALUES (3, 4, ‘Insert’, 'Rejected');

INSERT INTO hasaccess(UserId, HashValue, RequestType, AccessType)

VALUES (4, 3, ‘Insert’, 'Accepted');

INSERT INTO hasaccess(UserId, HashValue, RequestType, AccessType)

VALUES (5, 5, ‘Delete’, 'Rejected');
```



UserId	HashValue	AccessType	RequestType
1	2	Accepted	Insert
2	1	Accepted	Delete
3	4	Rejected	Insert
4	3	Accepted	Insert
5	5	Rejected	Delete

- Renewal

```
INSERT INTO renewal(UserLoginId, ManagerId, AuthorizationType)

VALUES (3, 4, 'Renewed');

INSERT INTO renewal(UserLoginId, ManagerId, AuthorizationType)

VALUES (1, 4, 'Rejected');

INSERT INTO renewal(UserLoginId, ManagerId, AuthorizationType)

VALUES (2, 3, 'Renewed');

INSERT INTO renewal(UserLoginId, ManagerId, AuthorizationType)

VALUES (4, 2, 'Rejected');

INSERT INTO renewal(UserLoginId, ManagerId, AuthorizationType)

VALUES (5, 1, 'Renewed');
```

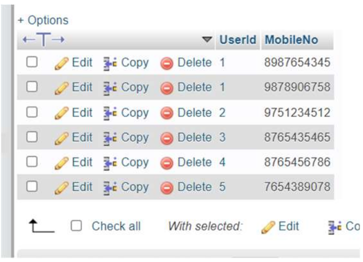


UserLoginId	ManagerId	AuthorizationType
1	4	Rejected
2	3	Renewed
3	4	Renewed
4	2	Rejected
5	1	Renewed

- Mobileno

```
INSERT INTO mobileno
VALUES (1, 9878906758);
INSERT INTO mobileno
VALUES (1, 8987654345);
INSERT INTO mobileno
VALUES (2, 9751234512);
INSERT INTO mobileno
VALUES (3, 8765435465);
```

```
INSERT INTO mobileno
VALUES (4, 8765456786);
INSERT INTO mobileno
VALUES (5, 7654389078);
```

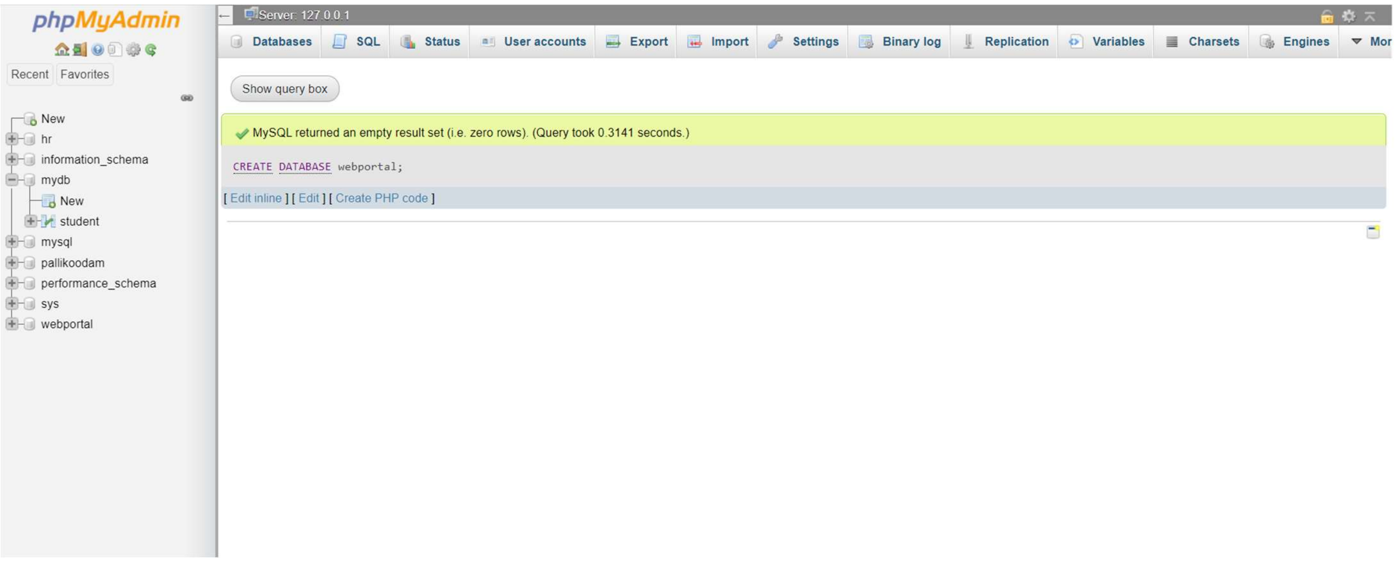


				UserId	MobileNo
□	Edit	Copy	Delete	1	8987654345
□	Edit	Copy	Delete	1	9878906758
□	Edit	Copy	Delete	2	9751234512
□	Edit	Copy	Delete	3	8765435465
□	Edit	Copy	Delete	4	8765456786
□	Edit	Copy	Delete	5	7654389078

SQL Commands

1. CREATE DATABASE

Creating database named ‘webportal’

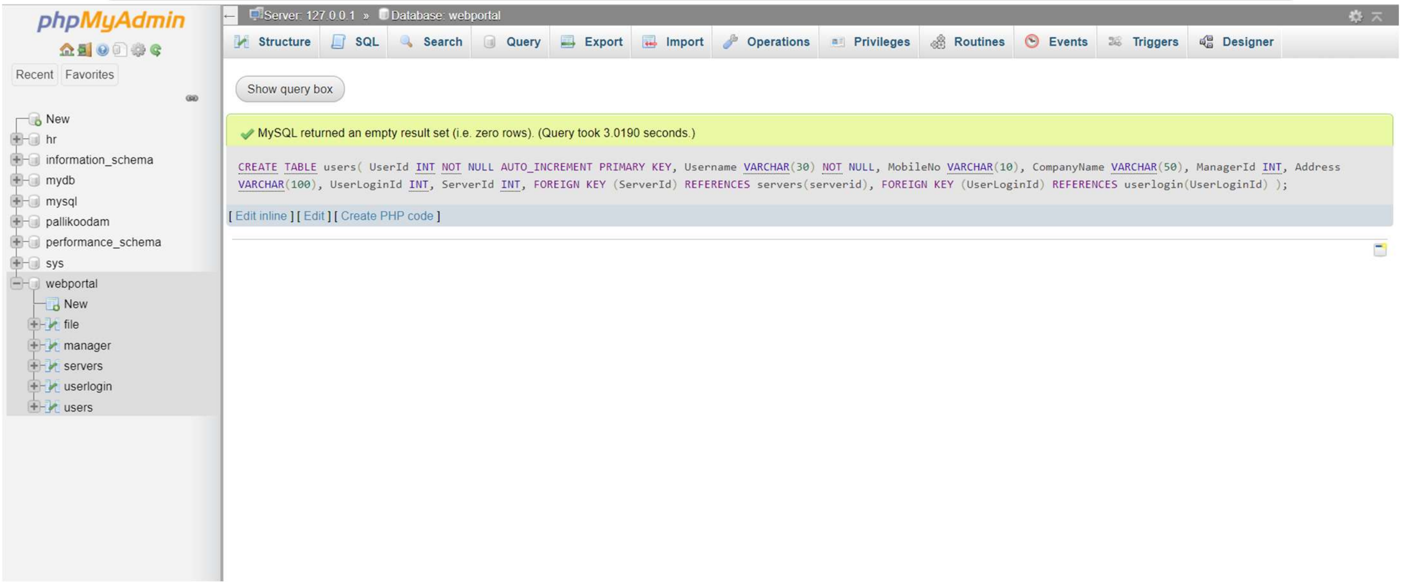


✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.3141 seconds.)

```
CREATE DATABASE webportal;
```

2. CREATE TABLE

Creating table named ‘users’



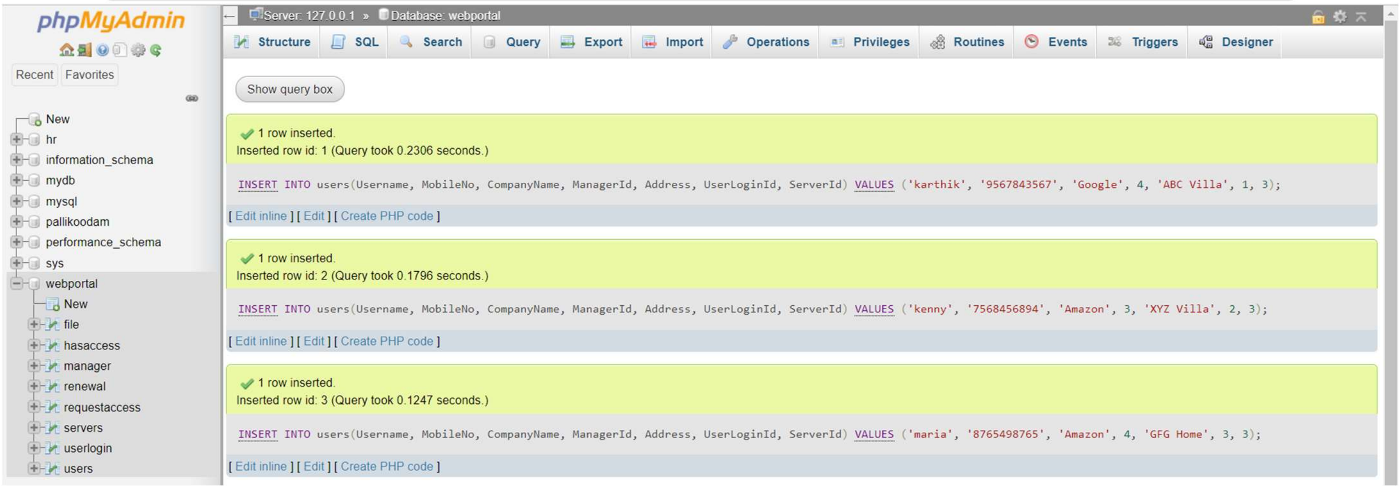
✓ MySQL returned an empty result set (i.e. zero rows). (Query took 3.0190 seconds.)

```
CREATE TABLE users( UserId INT NOT NULL AUTO_INCREMENT PRIMARY KEY, Username VARCHAR(30) NOT NULL, MobileNo VARCHAR(10), CompanyName VARCHAR(50), ManagerId INT, Address VARCHAR(100), UserLoginId INT, ServerId INT, FOREIGN KEY (ServerId) REFERENCES servers(serverid), FOREIGN KEY (UserLoginId) REFERENCES userlogin(UserLoginId) );
```

[Edit inline] [Edit] [Create PHP code]

3. INSERT

Inserting into table ‘users’



✓ 1 row inserted.
Inserted row id: 1 (Query took 0.2306 seconds.)

```
INSERT INTO users(Username, MobileNo, CompanyName, ManagerId, Address, UserLoginId, ServerId) VALUES ('karthik', '9567843567', 'Google', 4, 'ABC Villa', 1, 3);
```

[Edit inline] [Edit] [Create PHP code]

✓ 1 row inserted.
Inserted row id: 2 (Query took 0.1796 seconds.)

```
INSERT INTO users(Username, MobileNo, CompanyName, ManagerId, Address, UserLoginId, ServerId) VALUES ('kenny', '7568456894', 'Amazon', 3, 'XYZ Villa', 2, 3);
```

[Edit inline] [Edit] [Create PHP code]

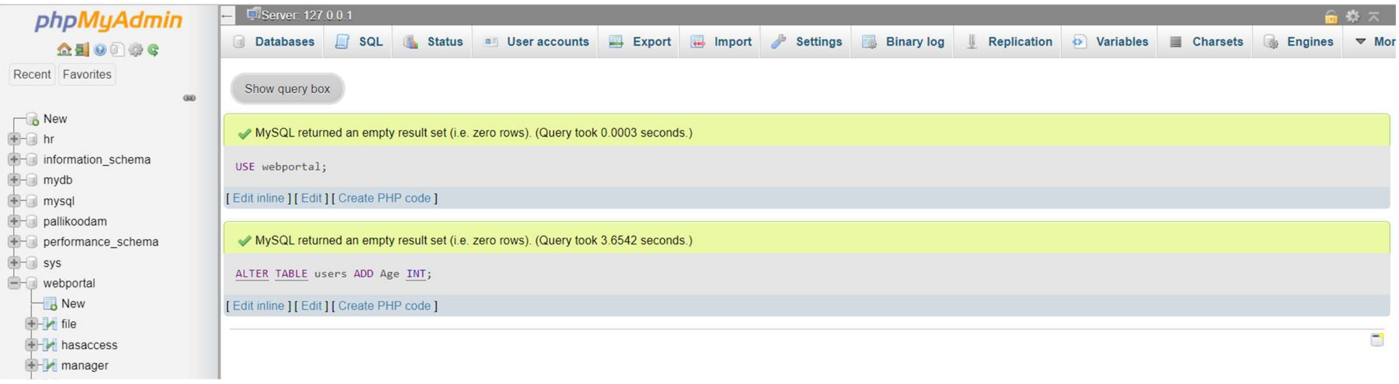
✓ 1 row inserted.
Inserted row id: 3 (Query took 0.1247 seconds.)

```
INSERT INTO users(Username, MobileNo, CompanyName, ManagerId, Address, UserLoginId, ServerId) VALUES ('maria', '8765498765', 'Amazon', 4, 'GFG Home', 3, 3);
```

[Edit inline] [Edit] [Create PHP code]

4. ALTER TABLE

Adding a column age to users table



✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)

```
USE webportal;
```

[Edit inline] [Edit] [Create PHP code]

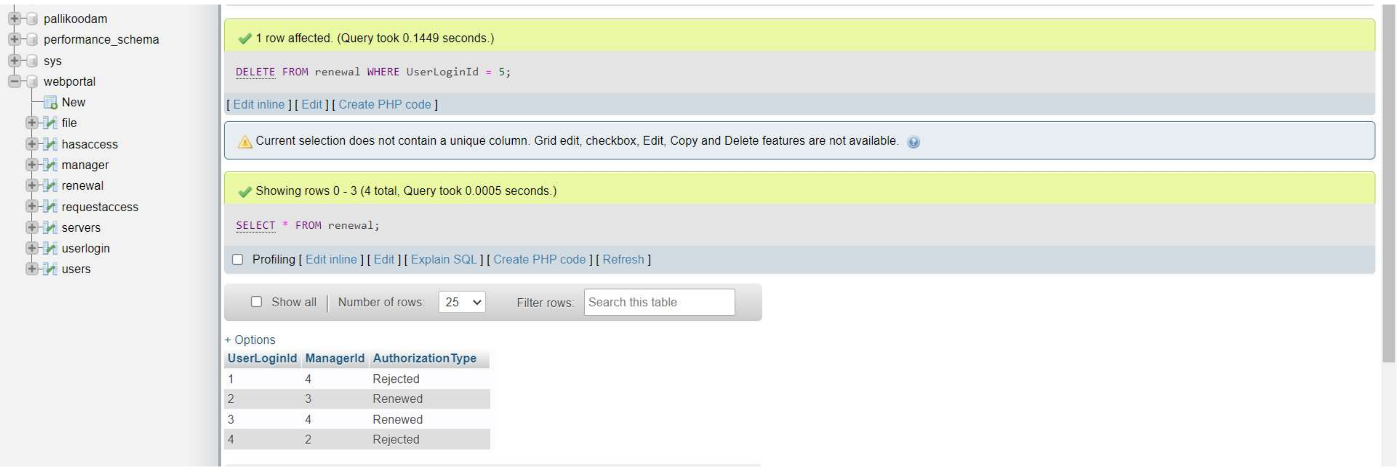
✓ MySQL returned an empty result set (i.e. zero rows). (Query took 3.6542 seconds.)

```
ALTER TABLE users ADD Age INT;
```

[Edit inline] [Edit] [Create PHP code]

5. DELETE

Delete a tuple from the renewal table



✓ 1 row affected. (Query took 0.1449 seconds.)

```
DELETE FROM renewal WHERE UserLoginId = 5;
```

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

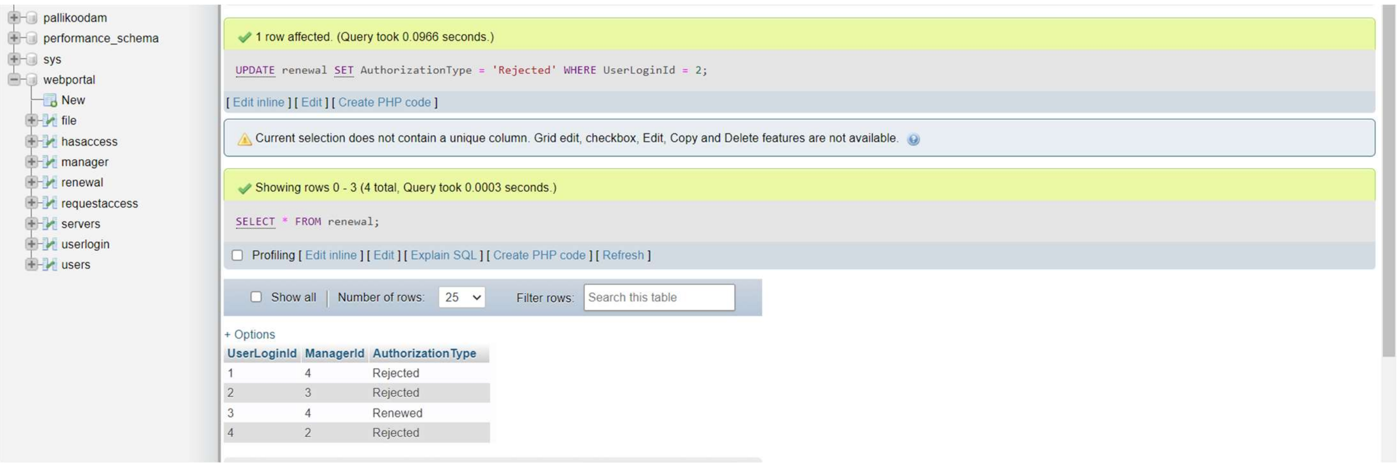
✓ Showing rows 0 - 3 (4 total, Query took 0.0005 seconds.)

```
SELECT * FROM renewal;
```

UserLoginId	ManagerId	AuthorizationType
1	4	Rejected
2	3	Renewed
3	4	Renewed
4	2	Rejected

6. UPDATE

Updating a tuple in the renewal table



✓ 1 row affected. (Query took 0.0966 seconds.)

```
UPDATE renewal SET AuthorizationType = 'Rejected' WHERE UserLoginId = 2;
```

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

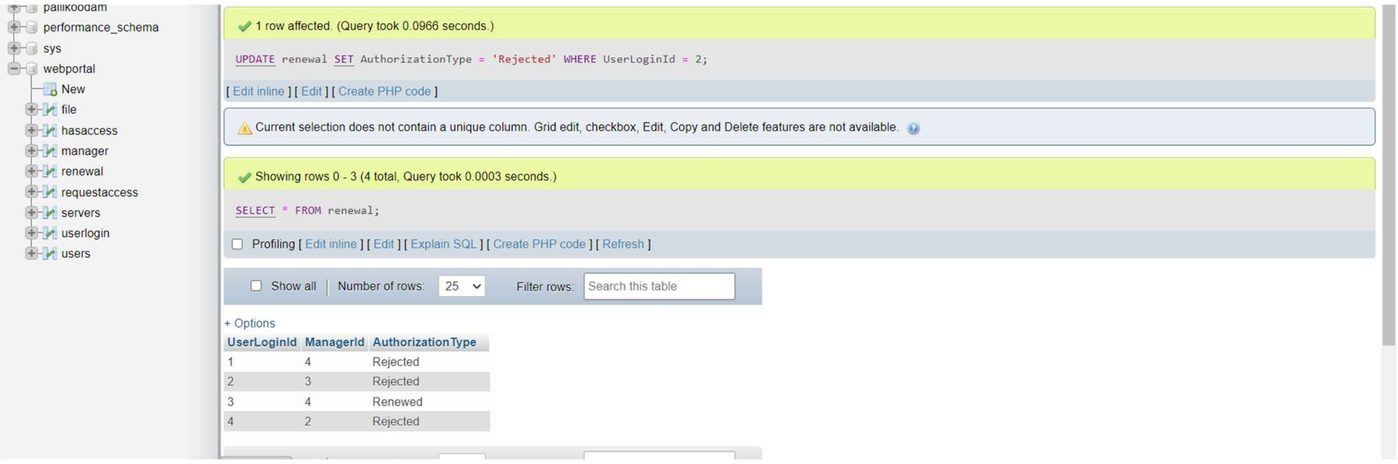
✓ Showing rows 0 - 3 (4 total, Query took 0.0003 seconds.)

```
SELECT * FROM renewal;
```

UserLoginId	ManagerId	AuthorizationType
1	4	Rejected
2	3	Rejected
3	4	Renewed
4	2	Rejected

7. WHERE

The tuple where UserLoginId = 2 is updated to ‘Rejected’.



✓ 1 row affected. (Query took 0.0966 seconds.)

```
UPDATE renewal SET AuthorizationType = 'Rejected' WHERE UserLoginId = 2;
```

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

✓ Showing rows 0 - 3 (4 total, Query took 0.0003 seconds.)

```
SELECT * FROM renewal;
```

UserLoginId	ManagerId	AuthorizationType
1	4	Rejected
2	3	Rejected
3	4	Renewed
4	2	Rejected

8. NOT NULL

Cannot have a tuple with NULL value in column with NOT NULL constraint.

9. PRIMARY KEY

The PRIMARY KEY constraint uniquely identifies each record in a table.

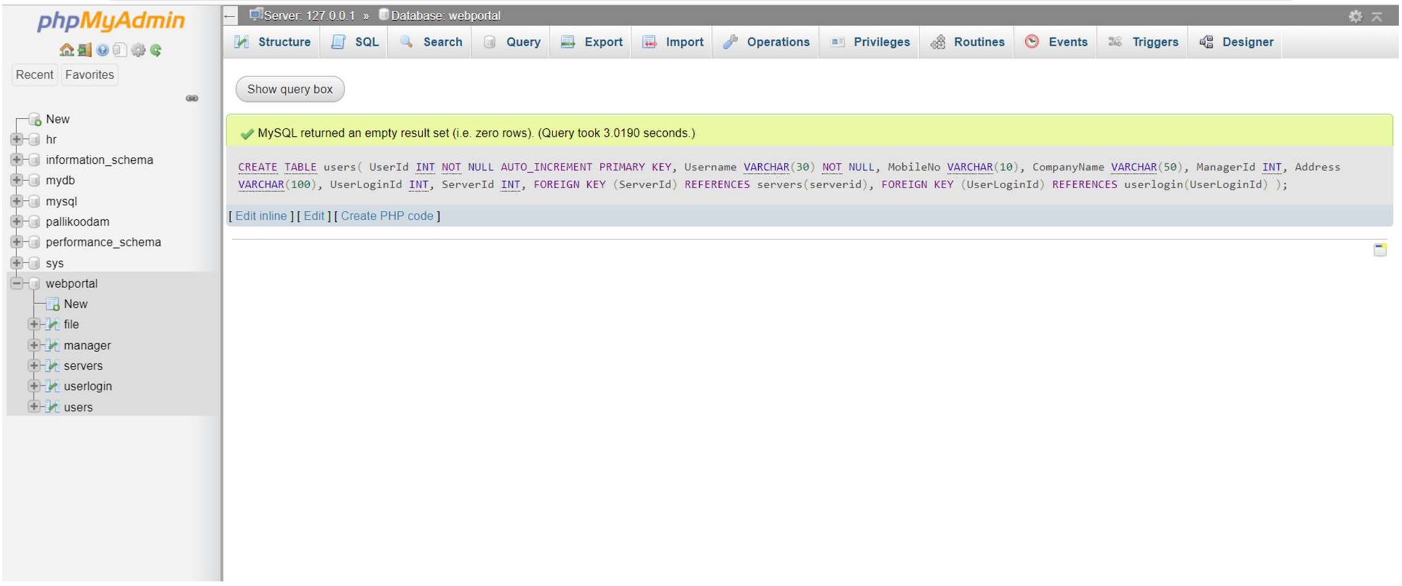
Here UserId is the PRIMARY KEY.

10. FOREIGN KEY

A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table. Here SeverId and UserLoginId are FOREIGN KEYS.

11. AUTO INCREMENT

Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.



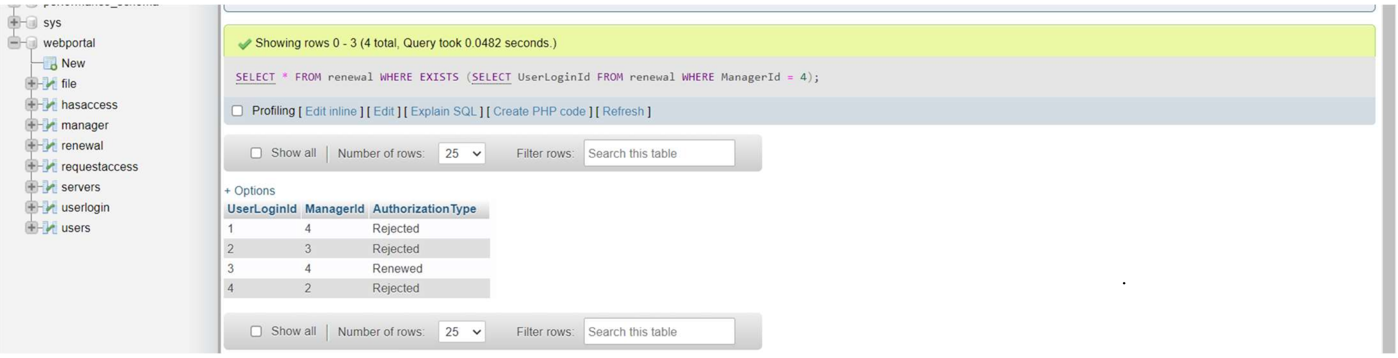
✓ MySQL returned an empty result set (i.e. zero rows). (Query took 3.0190 seconds.)

```
CREATE TABLE users( UserId INT NOT NULL AUTO_INCREMENT PRIMARY KEY, Username VARCHAR(30) NOT NULL, MobileNo VARCHAR(10), CompanyName VARCHAR(50), ManagerId INT, Address
VARCHAR(100), UserLoginId INT, ServerId INT, FOREIGN KEY (ServerId) REFERENCES servers(serverid), FOREIGN KEY (UserLoginId) REFERENCES userlogin(UserLoginId) );
```

[Edit inline] [Edit] [Create PHP code]

12. EXISTS

The EXISTS operator is used to test for the existence of any record in a subquery.
The EXISTS operator returns TRUE if the subquery returns one or more records.



✓ Showing rows 0 - 3 (4 total, Query took 0.0482 seconds.)

```
SELECT * FROM renewal WHERE EXISTS (SELECT UserLoginId FROM renewal WHERE ManagerId = 4);
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

□ Show all | Number of rows: 25 | Filter rows: Search this table

+ Options

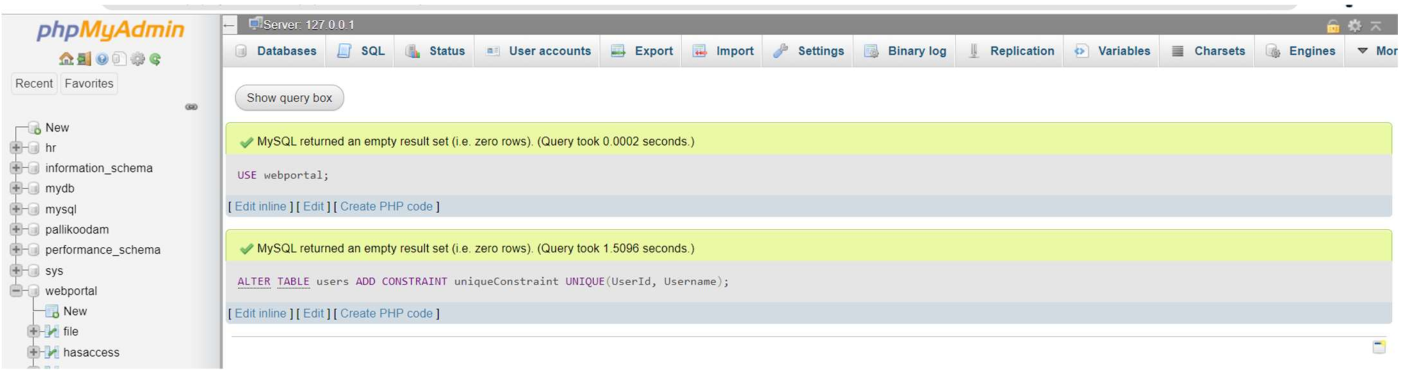
UserLoginId	ManagerId	AuthorizationType
1	4	Rejected
2	3	Rejected
3	4	Renewed
4	2	Rejected

□ Show all | Number of rows: 25 | Filter rows: Search this table

13. UNIQUE

The UNIQUE constraint ensures that all values in a column are different.

Here a unique constraint is added to columns UserId and Username.



MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)

```
USE webportal;
```

[Edit inline] [Edit] [Create PHP code]

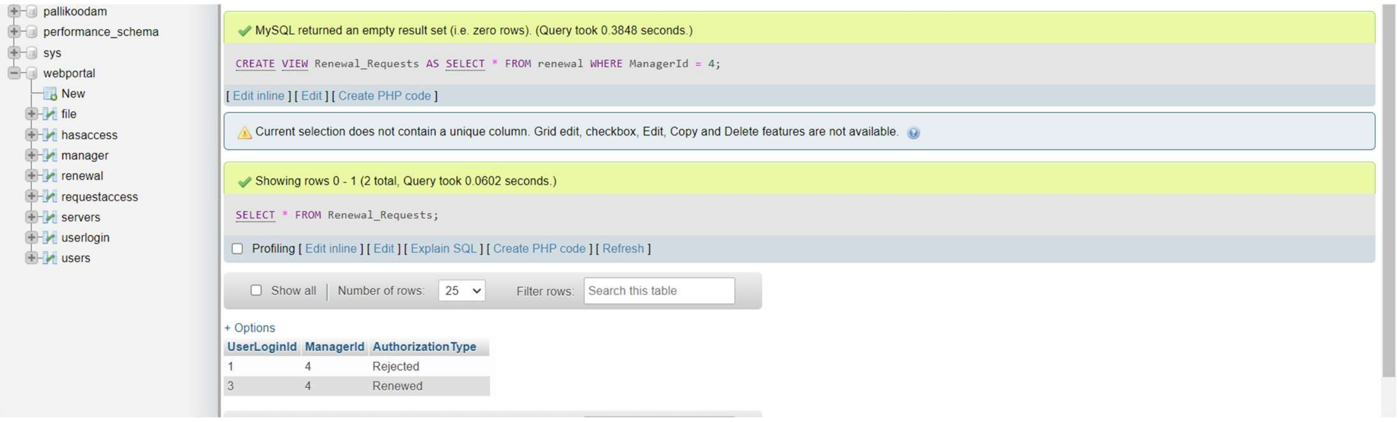
MySQL returned an empty result set (i.e. zero rows). (Query took 1.5096 seconds.)

```
ALTER TABLE users ADD CONSTRAINT uniqueConstraint UNIQUE(UserId, Username);
```

[Edit inline] [Edit] [Create PHP code]

14. VIEWS

A view is a virtual table based on the result-set of an SQL statement.



MySQL returned an empty result set (i.e. zero rows). (Query took 0.3848 seconds.)

```
CREATE VIEW Renewal_Requests AS SELECT * FROM renewal WHERE ManagerId = 4;
```

[Edit inline] [Edit] [Create PHP code]

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 1 (2 total, Query took 0.0602 seconds.)

```
SELECT * FROM Renewal_Requests;
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

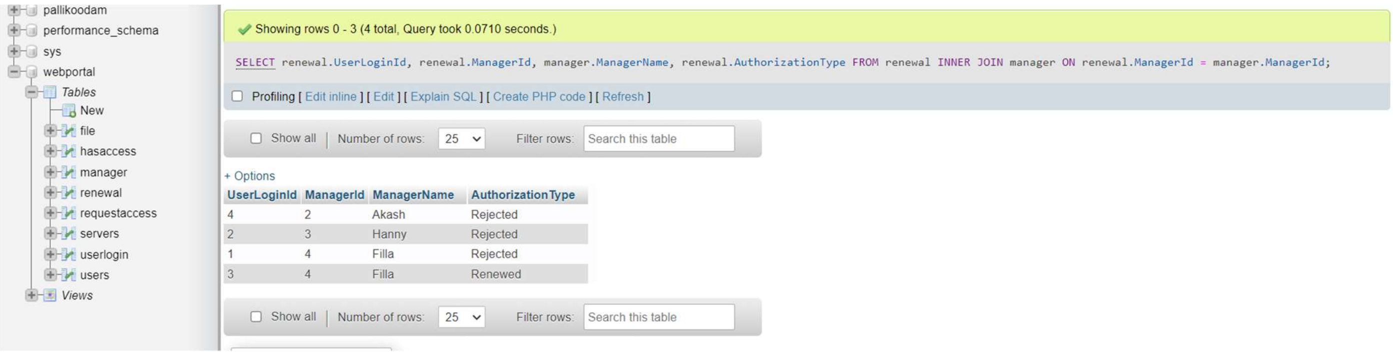
□ Show all | Number of rows: 25 Filter rows: Search this table

+ Options

UserLoginId	ManagerId	AuthorizationType
1	4	Rejected
3	4	Renewed

15. INNER JOIN

The INNER JOIN keyword selects records that have matching values in both tables.



Showing rows 0 - 3 (4 total, Query took 0.0710 seconds.)

```
SELECT renewal.UserLoginId, renewal.ManagerId, manager.ManagerName, renewal.AuthorizationType FROM renewal INNER JOIN manager ON renewal.ManagerId = manager.ManagerId;
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

□ Show all | Number of rows: 25 Filter rows: Search this table

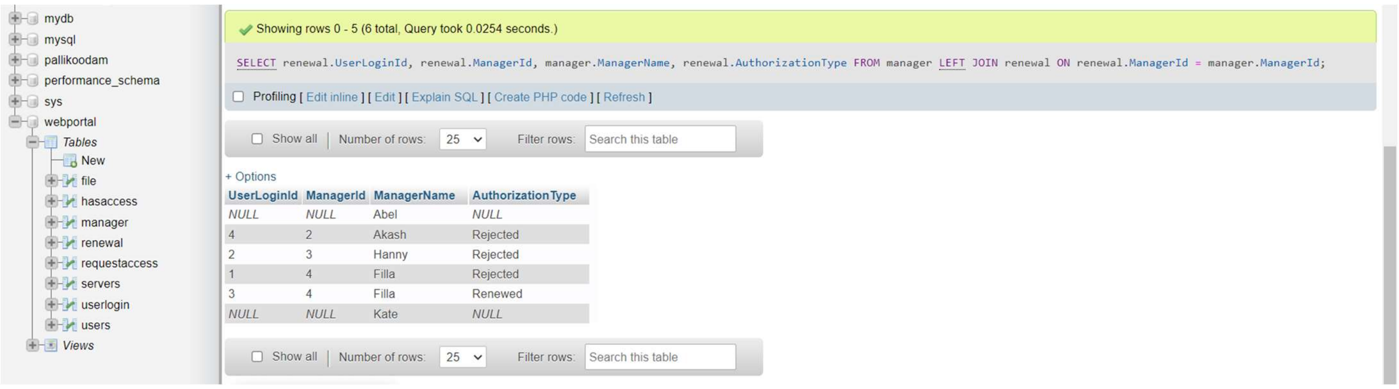
+ Options

UserLoginId	ManagerId	ManagerName	AuthorizationType
4	2	Akash	Rejected
2	3	Hanny	Rejected
1	4	Filla	Rejected
3	4	Filla	Renewed

□ Show all | Number of rows: 25 Filter rows: Search this table

16.LEFT JOIN

The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2).



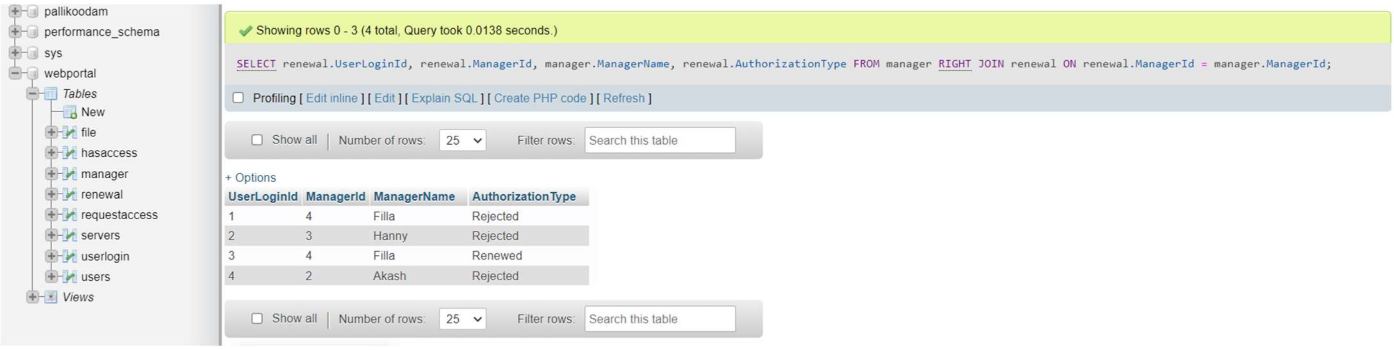
Showing rows 0 - 5 (6 total, Query took 0.0254 seconds.)

```
SELECT renewal.UserLoginId, renewal.ManagerId, manager.ManagerName, renewal.AuthorizationType FROM manager LEFT JOIN renewal ON renewal.ManagerId = manager.ManagerId;
```

UserLoginId	ManagerId	ManagerName	AuthorizationType
NULL	NULL	Abel	NULL
4	2	Akash	Rejected
2	3	Hanny	Rejected
1	4	Filla	Rejected
3	4	Filla	Renewed
NULL	NULL	Kate	NULL

17. RIGHT JOIN

The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1).



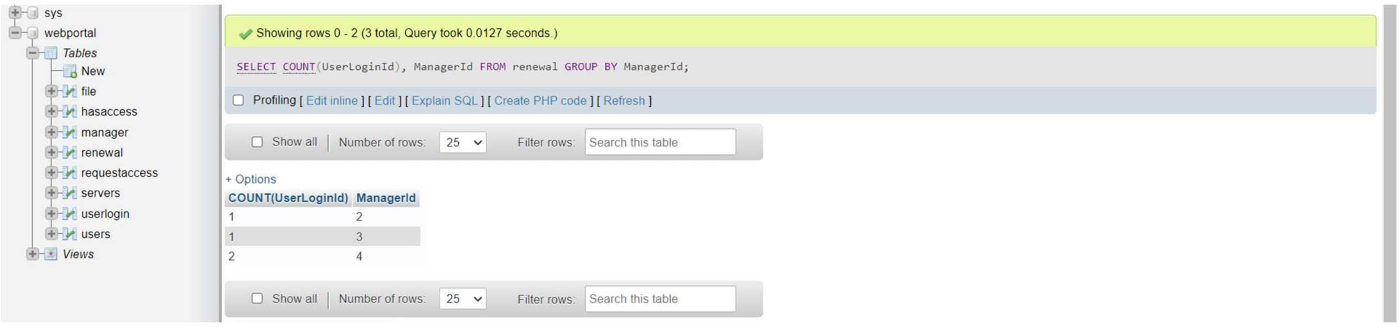
Showing rows 0 - 3 (4 total, Query took 0.0138 seconds.)

```
SELECT renewal.UserLoginId, renewal.ManagerId, manager.ManagerName, renewal.AuthorizationType FROM manager RIGHT JOIN renewal ON renewal.ManagerId = manager.ManagerId;
```

UserLoginId	ManagerId	ManagerName	AuthorizationType
1	4	Filla	Rejected
2	3	Hanny	Rejected
3	4	Filla	Renewed
4	2	Akash	Rejected

18. GROUP BY

The GROUP BY statement groups rows that have the same values into summary rows.



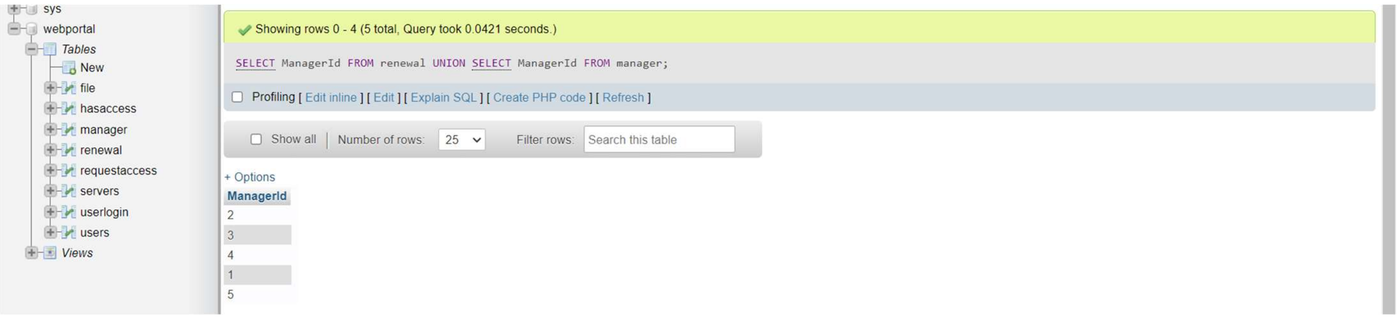
Showing rows 0 - 2 (3 total, Query took 0.0127 seconds.)

```
SELECT COUNT(UserLoginId), ManagerId FROM renewal GROUP BY ManagerId;
```

COUNT(UserLoginId)	ManagerId
1	2
1	3
2	4

19. UNION

The UNION operator is used to combine the result-set of two or more SELECT statements.


sys
webportal
Tables
New
file
hasaccess
manager
renewal
requestaccess
servers
userlogin
users
Views
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
+ Options

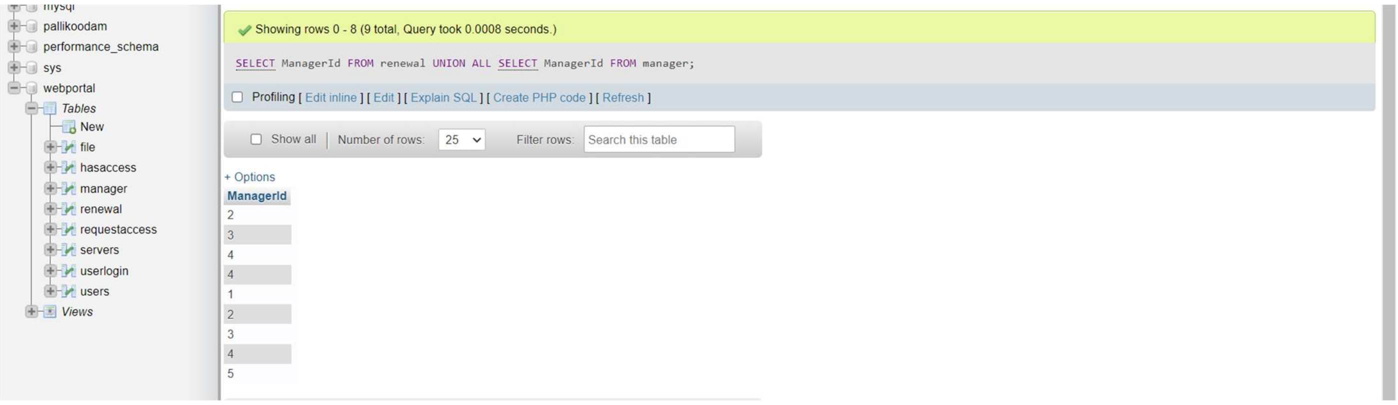
✓ Showing rows 0 - 4 (5 total, Query took 0.0421 seconds.)

```sql
SELECT ManagerId FROM renewal UNION SELECT ManagerId FROM manager;
```

ManagerId
2
3
4
1
5

20. UNION ALL

Allow duplicate values in the UNION operation.


mysql
pallikoodam
performance_schema
sys
webportal
Tables
New
file
hasaccess
manager
renewal
requestaccess
servers
userlogin
users
Views
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
+ Options

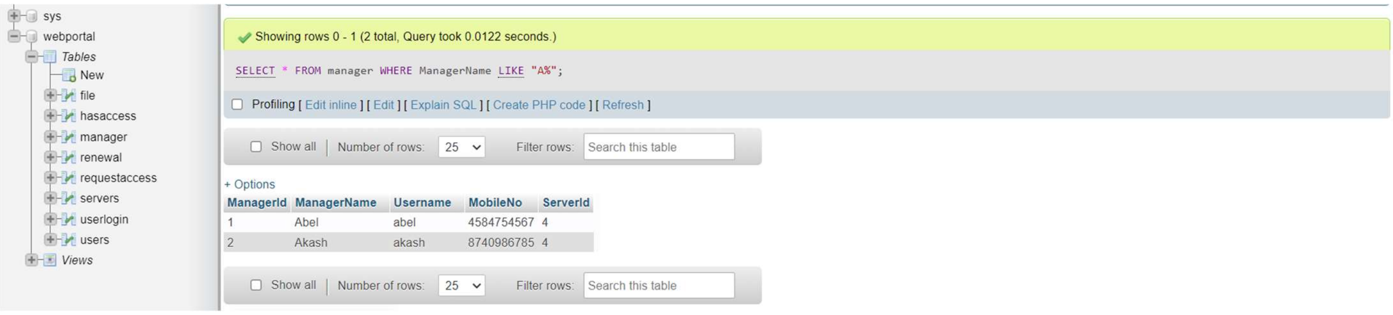
✓ Showing rows 0 - 8 (9 total, Query took 0.0008 seconds.)

```sql
SELECT ManagerId FROM renewal UNION ALL SELECT ManagerId FROM manager;
```

ManagerId
2
3
4
4
1
2
3
4
5

21. LIKE

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.


sys
webportal
Tables
New
file
hasaccess
manager
renewal
requestaccess
servers
userlogin
users
Views
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
+ Options
Show all | Number of rows: 25 Filter rows: Search this table

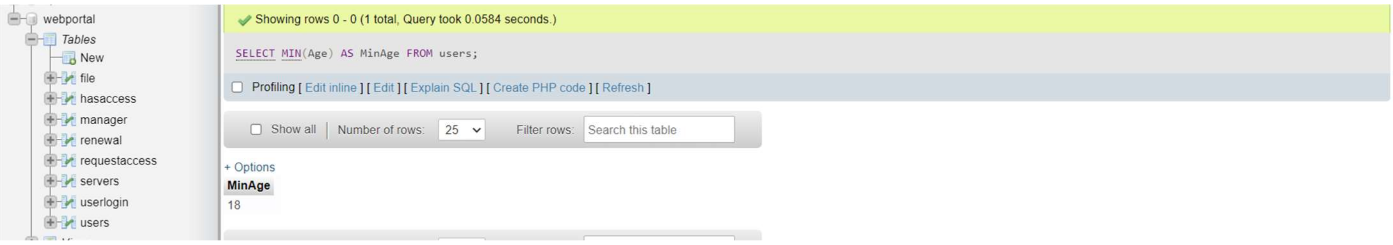
✓ Showing rows 0 - 1 (2 total, Query took 0.0122 seconds.)

```sql
SELECT * FROM manager WHERE ManagerName LIKE "A%";
```

ManagerId	ManagerName	Username	MobileNo	ServerId
1	Abel	abel	4584754567	4
2	Akash	akash	8740986785	4

22. MIN

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


webportal
Tables
New
file
hasaccess
manager
renewal
requestaccess
servers
userlogin
users
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
+ Options

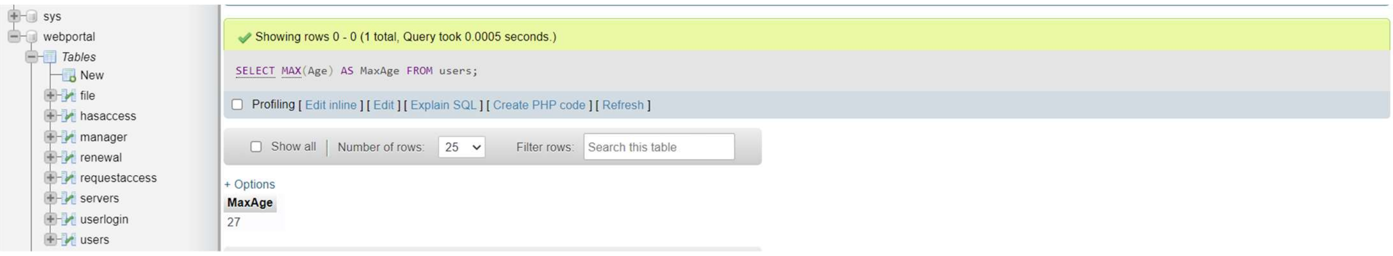
✓ Showing rows 0 - 0 (1 total, Query took 0.0584 seconds.)

```sql
SELECT MIN(Age) AS MinAge FROM users;
```

MinAge
18

23. MAX

The MAX () function returns the largest value of the selected column.



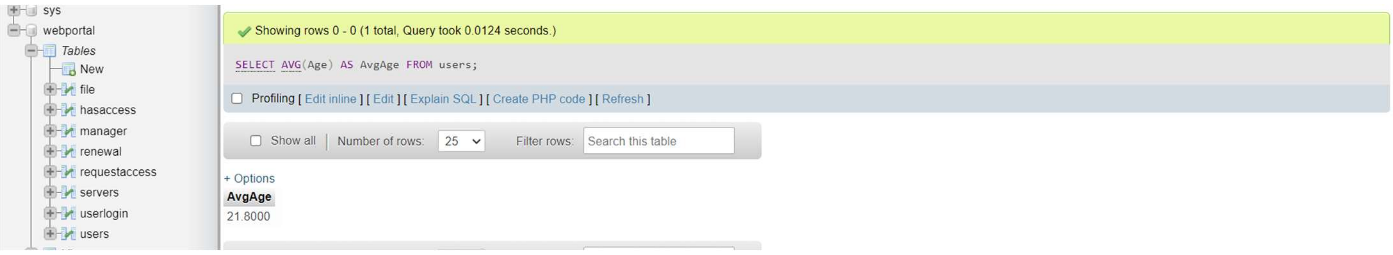
Showing rows 0 - 0 (1 total, Query took 0.0005 seconds.)

```
SELECT MAX(Age) AS MaxAge FROM users;
```

MaxAge
27

24. AVG

The AVG () function returns the average value of a numeric column.



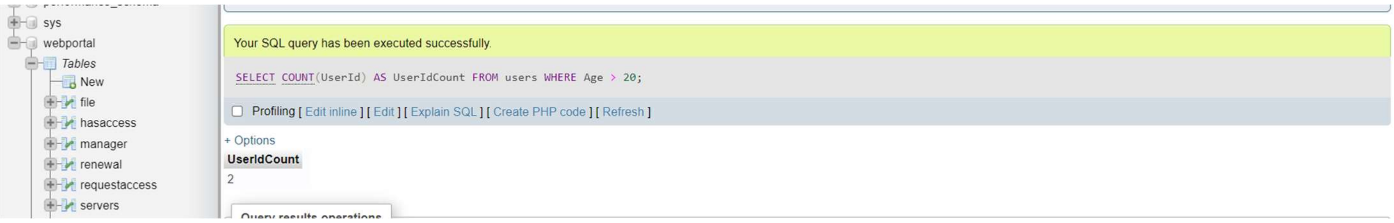
Showing rows 0 - 0 (1 total, Query took 0.0124 seconds.)

```
SELECT AVG(Age) AS AvgAge FROM users;
```

AvgAge
21.8000

25. COUNT

The COUNT () function returns the number of rows that matches a specified criterion.



Your SQL query has been executed successfully.

```
SELECT COUNT(UserId) AS UserIdCount FROM users WHERE Age > 20;
```

UserIdCount
2

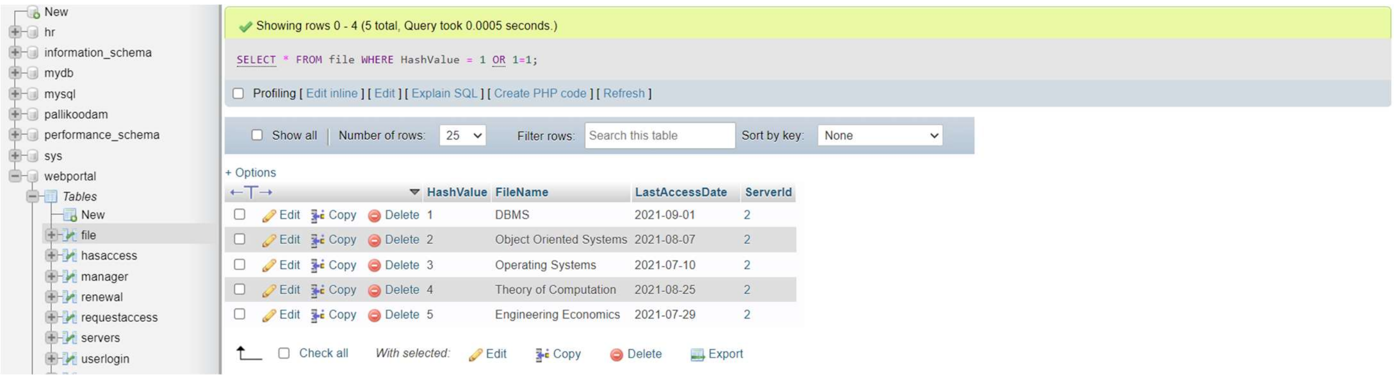
26. SQL INJECTION

SQL injection is the placement of malicious code in SQL statements.

```
txtHashId = getRequestString("HashValue");
txtSQL = "SELECT * FROM file WHERE HashValue = " + txtHashId;
```

```
SELECT * FROM file
WHERE HashValue = 1 OR 1=1;
```

If the input given for searching is ‘1 OR 1 = 1’ instead of 1 you will get all the records as 1=1 is always true instead of the one needed.



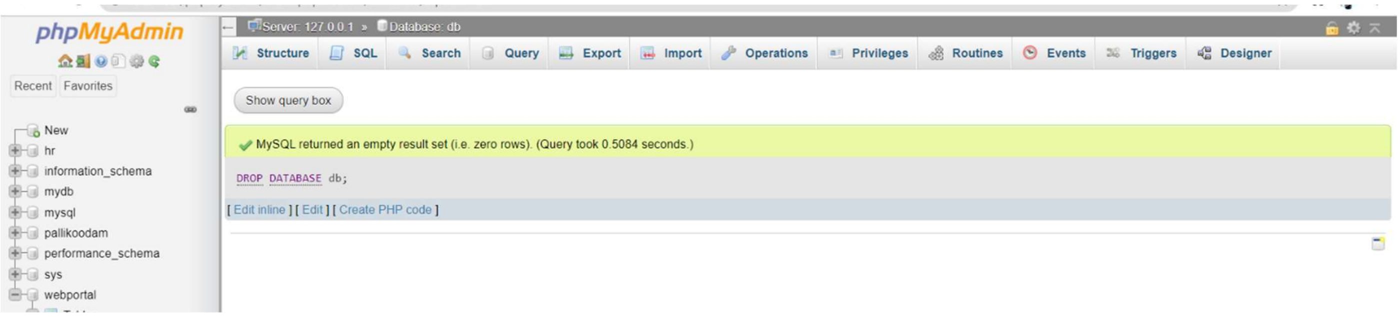
✓ Showing rows 0 - 4 (5 total, Query took 0.0005 seconds.)

```
SELECT * FROM file WHERE HashValue = 1 OR 1=1;
```

HashValue	FileName	LastAccessDate	ServerId
1	DBMS	2021-09-01	2
2	Object Oriented Systems	2021-08-07	2
3	Operating Systems	2021-07-10	2
4	Theory of Computation	2021-08-25	2
5	Engineering Economics	2021-07-29	2

27. DROP DB

The DROP DATABASE statement is used to drop an existing SQL database.

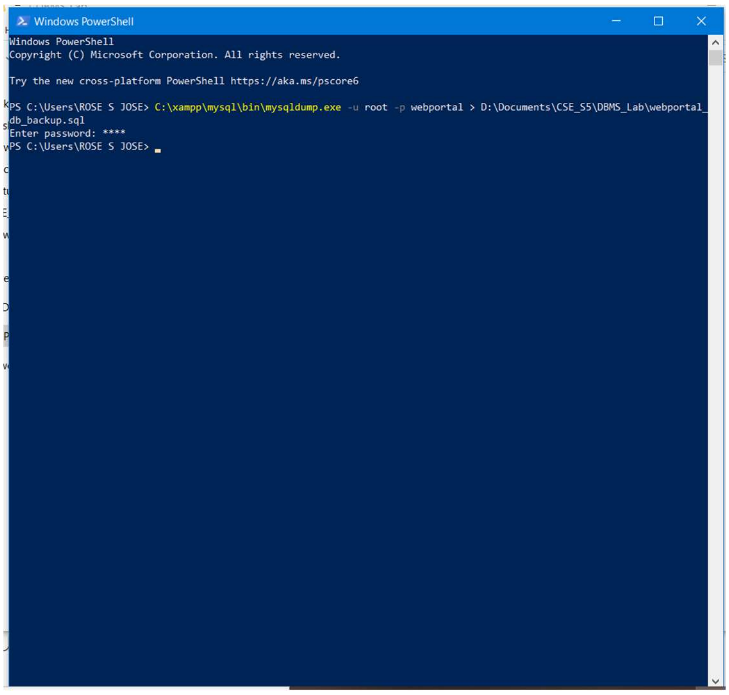


✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.5084 seconds.)

```
DROP DATABASE db;
```

28. BACKUP DB

The command in Windows PowerShell Create a backup for the database.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\ROSE S JOSE> C:\xampp\mysql\bin\mysqldump.exe -u root -p webportal > D:\Documents\CSE_S5\DBMS_Lab\webportal_
db_backup.sql
Enter password: ****
PS C:\Users\ROSE S JOSE>
```