Green University of Bangladesh

Department of Computer Science and Engineering(CSE)

Faculty of Sciences and Engineering

Semester: (Summer, Year:2021), B.Sc. in CSE (Day)

LAB REPORT NO 3

Course Title: Database System Lab

Course Code: CSE 210 Section:PC- 193 D

<u>Student Details</u>

Name		ID
1.	Azmary Akter	193902019

Lab Date : 10 July, 2021
Submission Date : 7 August, 2021
Course Teacher's Name : Md. Moshiur Rahman

<u>Lab Report Status</u>

Marks: ………………………………… Signature:....................
Comments:... Date:.............................

1. TITLE OF THE LAB EXPERIMENT

Here we will implement

- □ **Using different relational operators**
- □ **And, or, not (logical operator)**
- □ **Limit, Order BY, IN, Not in, Between, Not between**
- □ **Like operator**

2. OBJECTIVES/AIM [1]

Our goal is to implement all the Mysql topics properly.

3. PROCEDURE / ANALYSIS / DESIGN [2]

- At first we implemented all Query Using different relational operators Like ,
  - ➔ greater than or equal,
  - ➔ <>
  - ➔ String Match with condition.
  - ➔ String match with logical operators.

- Then We Updated data information by using And, or, not (logical operator).
- After that we have discussed Limit, Order BY, IN, Not in, Between, Not between.
- Lastly we Implement Like operator by query for searching anything.

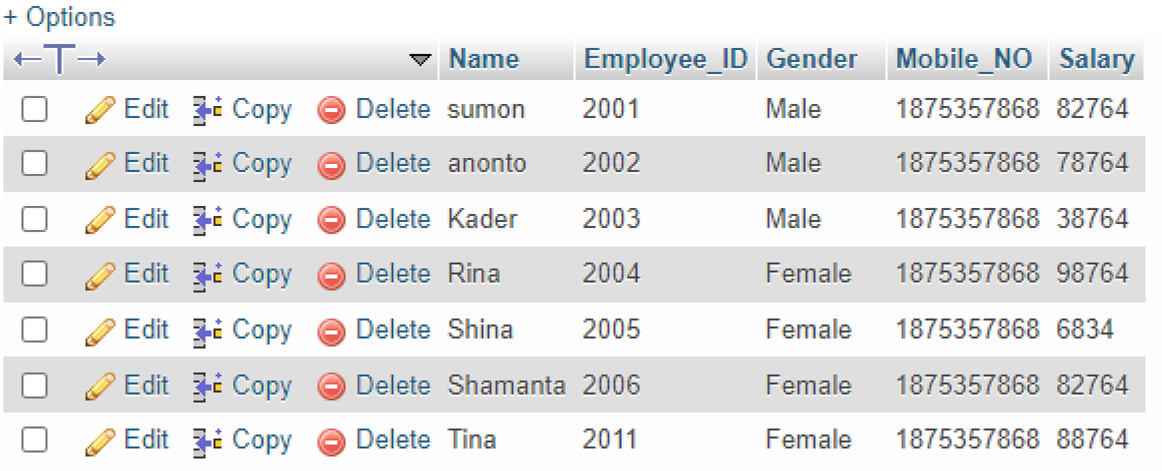
4. IMPLEMENTATION [2]

□ Limit, Order BY, IN, Not in, Between, Not between

- ### Implementation of relational operators

Relational operators are used for comparing numbers and strings. If a string is compared to a number, MySQL will try to convert the string to a number. If a TIMESTAMP column is compared to a string or a number, MySQL will attempt to convert the string or number to a timestamp value. If it’s unsuccessful at converting the other value to a timestamp, it will convert the TIMESTAMP column’s value to a string or a number. TIME and DATE columns are compared to other values as strings.

Before :



+ Options

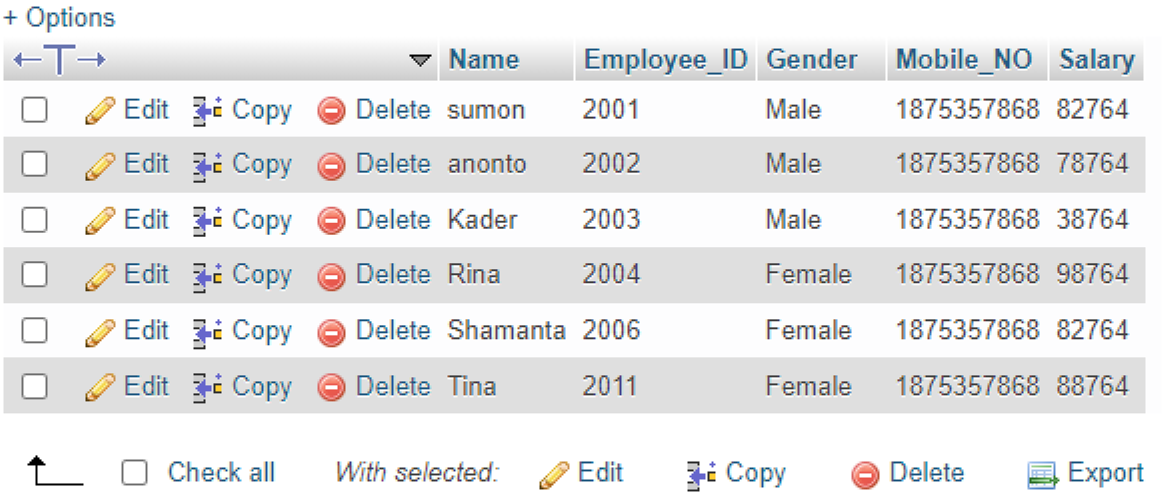
				Name	Employee_ID	Gender	Mobile_NO	Salary
□	Edit	Copy	Delete	sumon	2001	Male	1875357868	82764
□	Edit	Copy	Delete	anonto	2002	Male	1875357868	78764
□	Edit	Copy	Delete	Kader	2003	Male	1875357868	38764
□	Edit	Copy	Delete	Rina	2004	Female	1875357868	98764
□	Edit	Copy	Delete	Shina	2005	Female	1875357868	6834
□	Edit	Copy	Delete	Shamanta	2006	Female	1875357868	82764
□	Edit	Copy	Delete	Tina	2011	Female	1875357868	88764

- **Query of range(greater than or equal):**

SELECT * FROM `employee_information` WHERE Salary >= 7000;

(This query will show us those salary which is greater than or equal to the 7000)

After :

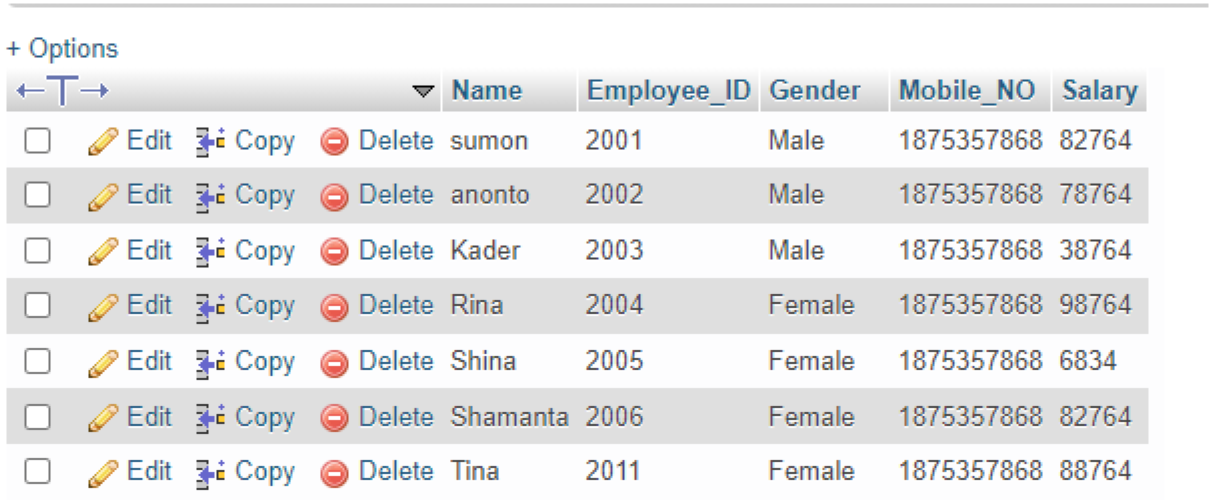


+ Options

				Name	Employee_ID	Gender	Mobile_NO	Salary
□	Edit	Copy	Delete	sumon	2001	Male	1875357868	82764
□	Edit	Copy	Delete	anonto	2002	Male	1875357868	78764
□	Edit	Copy	Delete	Kader	2003	Male	1875357868	38764
□	Edit	Copy	Delete	Rina	2004	Female	1875357868	98764
□	Edit	Copy	Delete	Shamanta	2006	Female	1875357868	82764
□	Edit	Copy	Delete	Tina	2011	Female	1875357868	88764

- **Query of range(<>):**

Before :



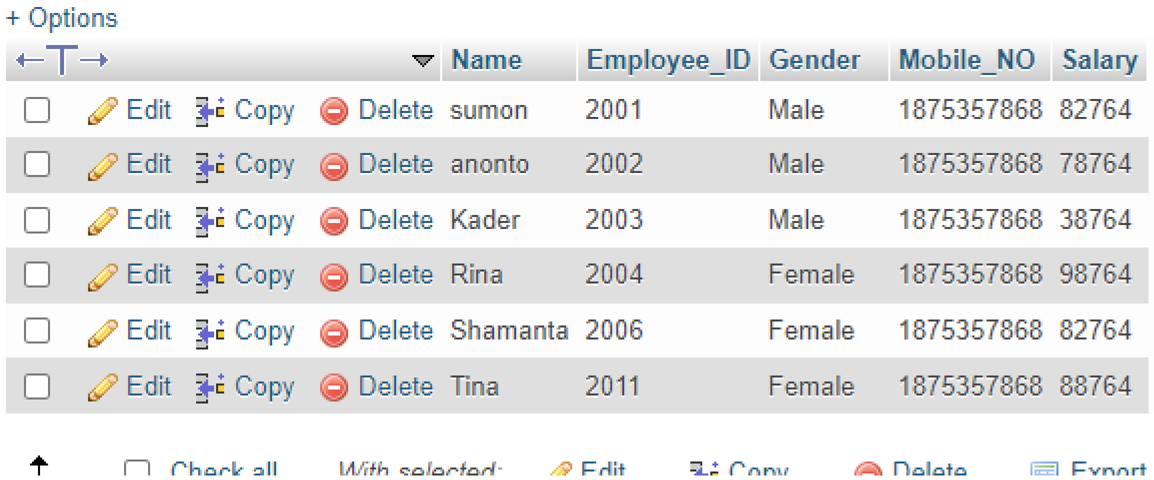
+ Options

←T→			▼	Name	Employee_ID	Gender	Mobile_NO	Salary
□	Edit	Copy	Delete	sumon	2001	Male	1875357868	82764
□	Edit	Copy	Delete	anonto	2002	Male	1875357868	78764
□	Edit	Copy	Delete	Kader	2003	Male	1875357868	38764
□	Edit	Copy	Delete	Rina	2004	Female	1875357868	98764
□	Edit	Copy	Delete	Shina	2005	Female	1875357868	6834
□	Edit	Copy	Delete	Shamanta	2006	Female	1875357868	82764
□	Edit	Copy	Delete	Tina	2011	Female	1875357868	88764

➔ **Query of range(<>):**

SELECT * FROM `employee_information` WHERE Salary <> 6834;

After :

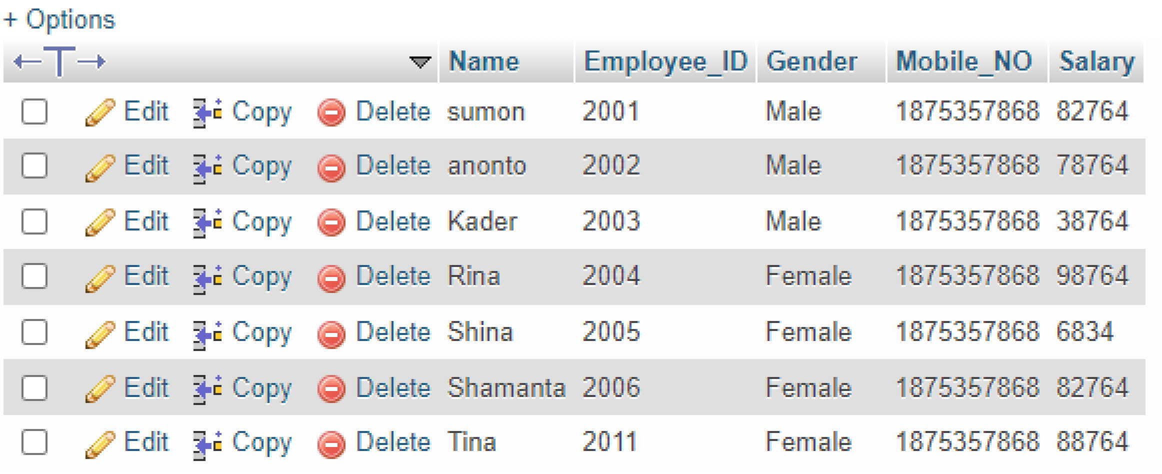


+ Options

←T→			▼	Name	Employee_ID	Gender	Mobile_NO	Salary
□	Edit	Copy	Delete	sumon	2001	Male	1875357868	82764
□	Edit	Copy	Delete	anonto	2002	Male	1875357868	78764
□	Edit	Copy	Delete	Kader	2003	Male	1875357868	38764
□	Edit	Copy	Delete	Rina	2004	Female	1875357868	98764
□	Edit	Copy	Delete	Shamanta	2006	Female	1875357868	82764
□	Edit	Copy	Delete	Tina	2011	Female	1875357868	88764

- **Query of range(=) :**

Before :

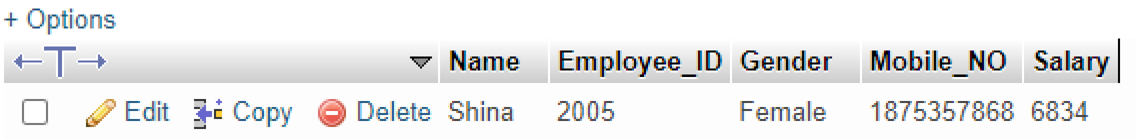


				Name	Employee_ID	Gender	Mobile_NO	Salary
□	Edit	Copy	Delete	sumon	2001	Male	1875357868	82764
□	Edit	Copy	Delete	anonto	2002	Male	1875357868	78764
□	Edit	Copy	Delete	Kader	2003	Male	1875357868	38764
□	Edit	Copy	Delete	Rina	2004	Female	1875357868	98764
□	Edit	Copy	Delete	Shina	2005	Female	1875357868	6834
□	Edit	Copy	Delete	Shamanta	2006	Female	1875357868	82764
□	Edit	Copy	Delete	Tina	2011	Female	1875357868	88764

➔ **Query of range(=) :**

SELECT * FROM `employee_information` WHERE Salary = 6834;

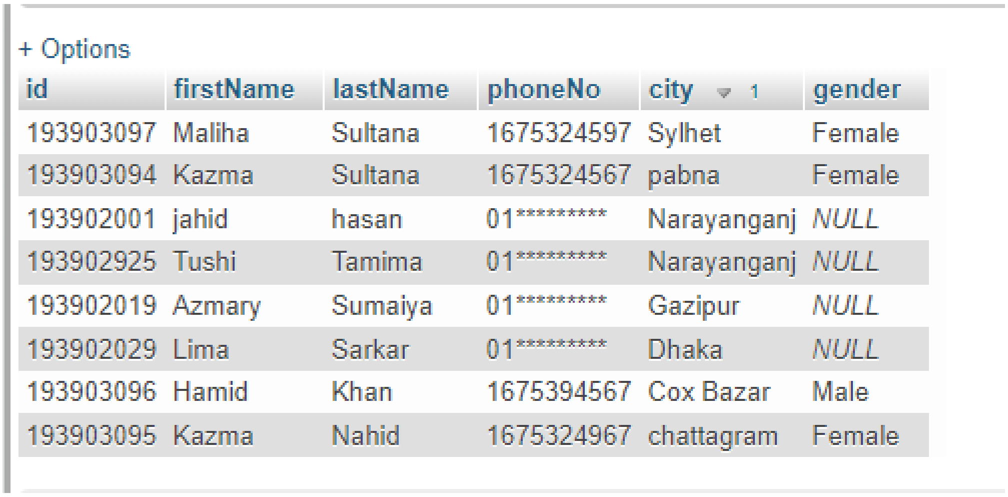
After :



				Name	Employee_ID	Gender	Mobile_NO	Salary
□	Edit	Copy	Delete	Shina	2005	Female	1875357868	6834

- **Query of String match with And :**

Before:

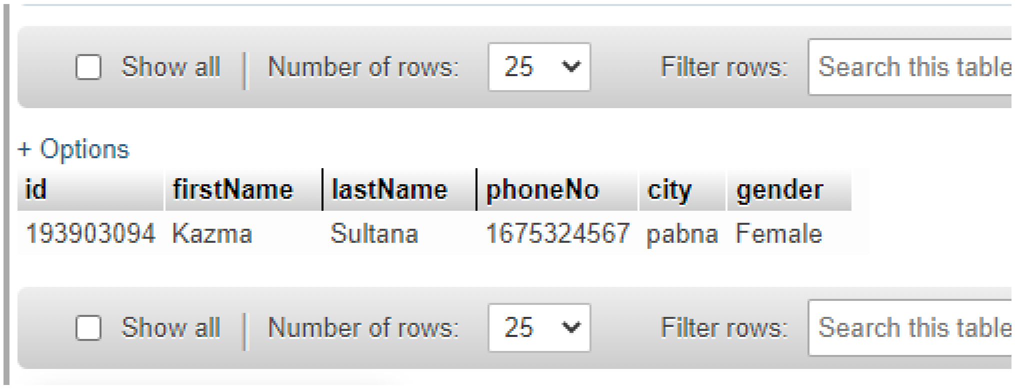


id	firstName	lastName	phoneNo	city ▼ 1	gender
193903097	Maliha	Sultana	1675324597	Sylhet	Female
193903094	Kazma	Sultana	1675324567	pabna	Female
193902001	jahid	hasan	01*********	Narayanganj	*NULL*
193902925	Tushi	Tamima	01*********	Narayanganj	*NULL*
193902019	Azmary	Sumaiya	01*********	Gazipur	*NULL*
193902029	Lima	Sarkar	01*********	Dhaka	*NULL*
193903096	Hamid	Khan	1675394567	Cox Bazar	Male
193903095	Kazma	Nahid	1675324967	chattagram	Female

➔ Query of String match :

SELECT * FROM `info` WHERE firstName = "Kazma" and lastName = "Sultana";

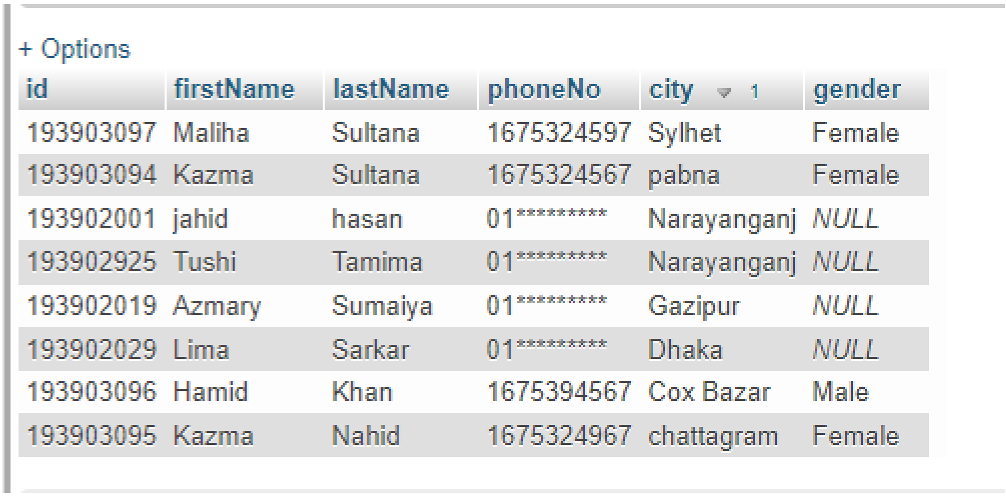
After :



id	firstName	lastName	phoneNo	city	gender
193903094	Kazma	Sultana	1675324567	pabna	Female

- **Query of String match with OR:**

Before:

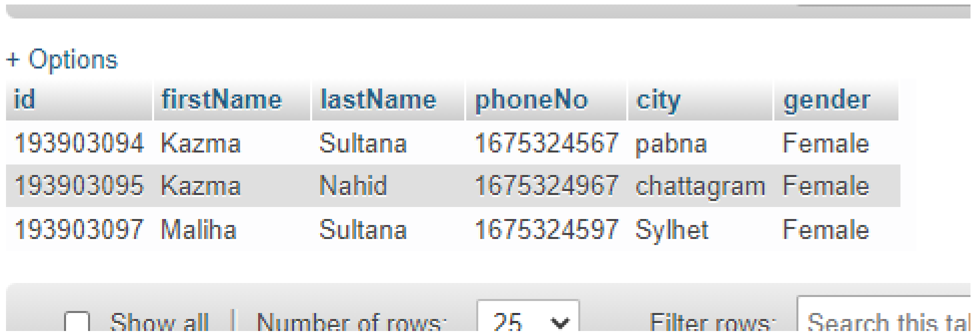


id	firstName	lastName	phoneNo	city 1	gender
193903097	Maliha	Sultana	1675324597	Sylhet	Female
193903094	Kazma	Sultana	1675324567	pabna	Female
193902001	jahid	hasan	01*********	Narayanganj	*NULL*
193902925	Tushi	Tamima	01*********	Narayanganj	*NULL*
193902019	Azmary	Sumaiya	01*********	Gazipur	*NULL*
193902029	Lima	Sarkar	01*********	Dhaka	*NULL*
193903096	Hamid	Khan	1675394567	Cox Bazar	Male
193903095	Kazma	Nahid	1675324967	chattagram	Female

➔ Query of String match :

SELECT * FROM `info` WHERE firstName = "Kazma" OR lastName = "Sultana";

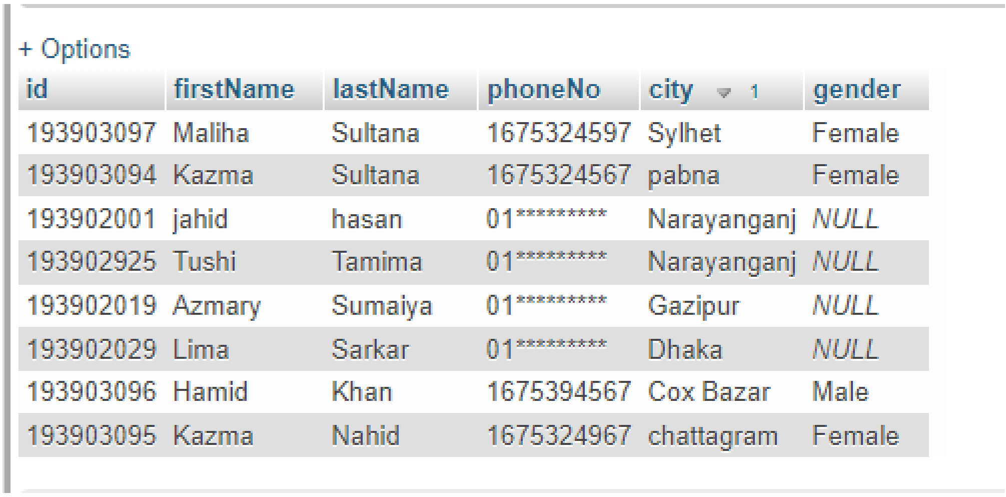
After:



id	firstName	lastName	phoneNo	city	gender
193903094	Kazma	Sultana	1675324567	pabna	Female
193903095	Kazma	Nahid	1675324967	chattagram	Female
193903097	Maliha	Sultana	1675324597	Sylhet	Female

- **Query of String match on 3 condition :**

Before :

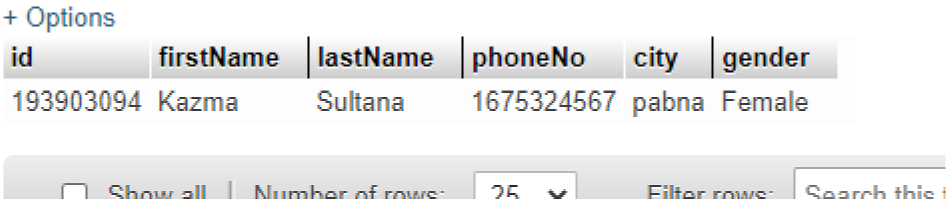


id	firstName	lastName	phoneNo	city ▼ 1	gender
193903097	Maliha	Sultana	1675324597	Sylhet	Female
193903094	Kazma	Sultana	1675324567	pabna	Female
193902001	jahid	hasan	01*********	Narayanganj	*NULL*
193902925	Tushi	Tamima	01*********	Narayanganj	*NULL*
193902019	Azmary	Sumaiya	01*********	Gazipur	*NULL*
193902029	Lima	Sarkar	01*********	Dhaka	*NULL*
193903096	Hamid	Khan	1675394567	Cox Bazar	Male
193903095	Kazma	Nahid	1675324967	chattagram	Female

➔ **Query :**

SELECT * FROM `info` WHERE (firstName = "Kazma" OR lastName = "Sultana") and city = "pabna";

After :



id	firstName	lastName	phoneNo	city	gender
193903094	Kazma	Sultana	1675324567	pabna	Female

☐ Implementation of And, or, not (logical operator) in MySQL

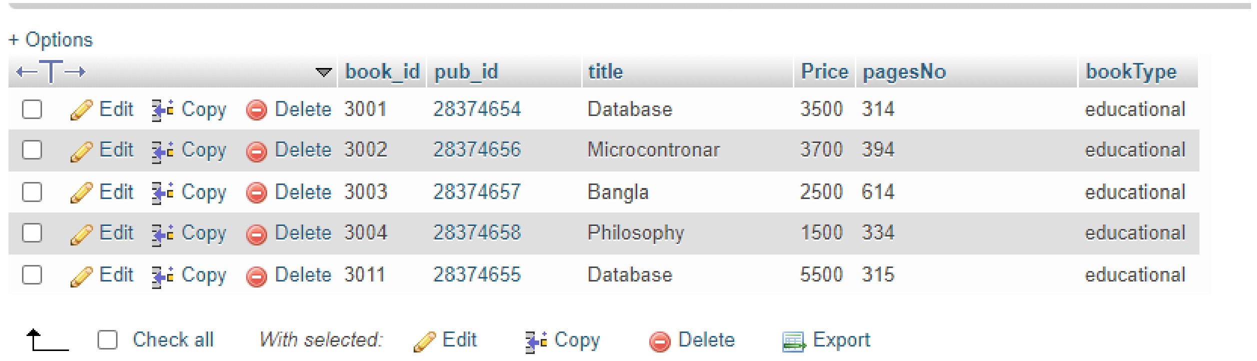
Why we use And, or, not (logical operator) in MySQL :

The AND and OR operators are used to filter records based on more than one condition: The AND operator displays a record if all the conditions separated by AND are TRUE. The OR operator displays a record if any of the conditions separated by OR is TRUE.

- **AND:**

MySQL logical AND operator compares two expressions and returns true if both of the expressions are true.

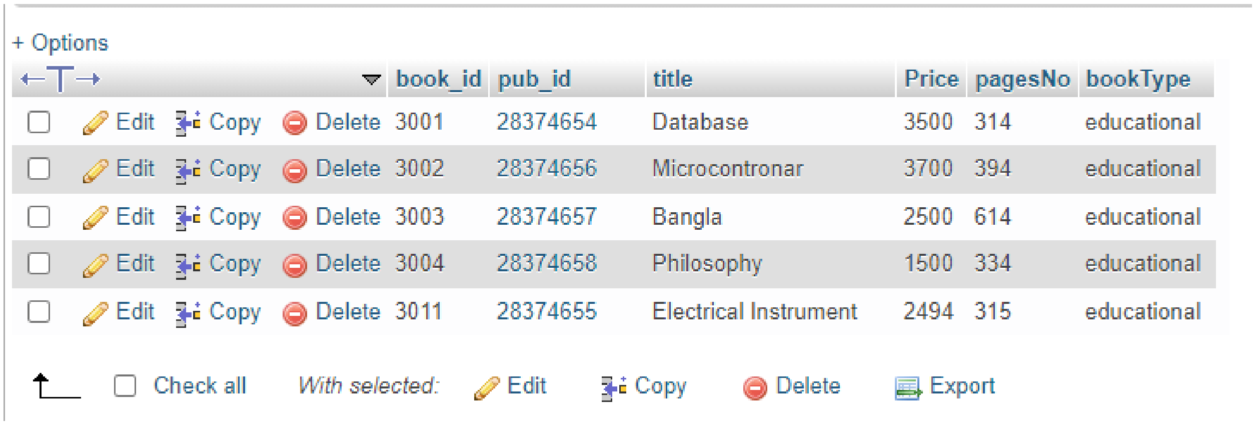
Before updating:



				book_id	pub_id	title	Price	pagesNo	bookType
☐	Edit	Copy	Delete	3001	28374654	Database	3500	314	educational
☐	Edit	Copy	Delete	3002	28374656	Microcontronar	3700	394	educational
☐	Edit	Copy	Delete	3003	28374657	Bangla	2500	614	educational
☐	Edit	Copy	Delete	3004	28374658	Philosophy	1500	334	educational
☐	Edit	Copy	Delete	3011	28374655	Database	5500	315	educational

```
Update book
SET title = "Electrical Instrument" , Price= 2494
WHERE title = "Database" AND pagesNo = 315;
```

After updating:

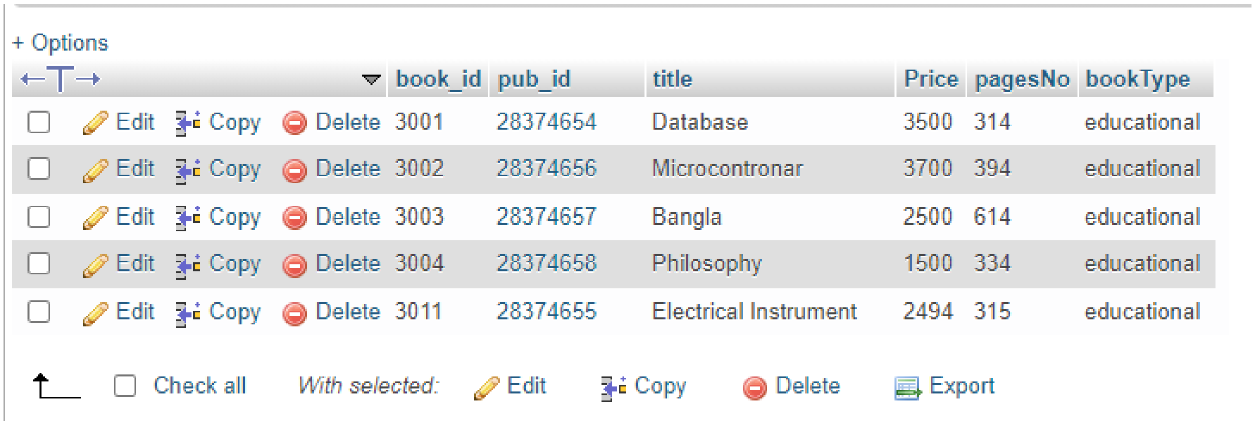


				book_id	pub_id	title	Price	pagesNo	bookType
☐	Edit	Copy	Delete	3001	28374654	Database	3500	314	educational
☐	Edit	Copy	Delete	3002	28374656	Microcontronar	3700	394	educational
☐	Edit	Copy	Delete	3003	28374657	Bangla	2500	614	educational
☐	Edit	Copy	Delete	3004	28374658	Philosophy	1500	334	educational
☐	Edit	Copy	Delete	3011	28374655	Electrical Instrument	2494	315	educational

- **OR :**

MySQL OR operator compares two expressions and returns TRUE if either of the expressions is TRUE.

Before updating:



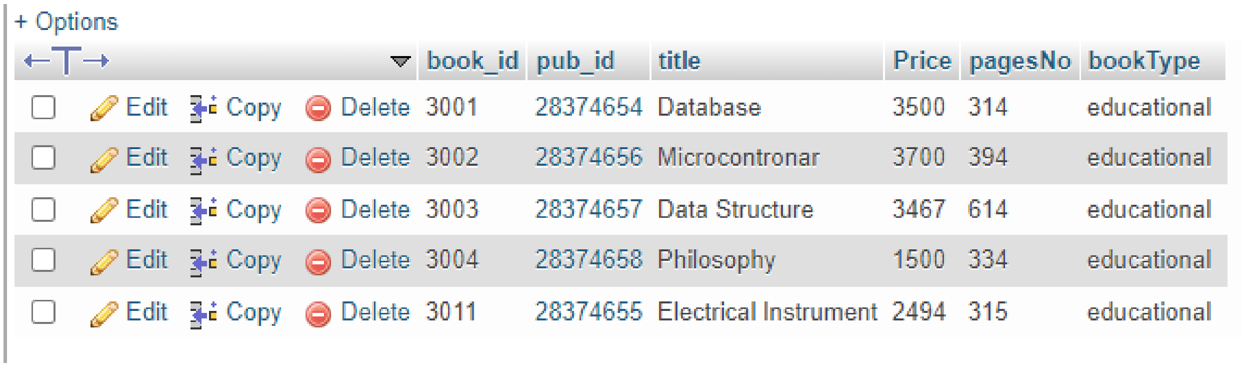
+ Options

←T→			▼	book_id	pub_id	title	Price	pagesNo	bookType
□	Edit	Copy	Delete	3001	28374654	Database	3500	314	educational
□	Edit	Copy	Delete	3002	28374656	Microcontronar	3700	394	educational
□	Edit	Copy	Delete	3003	28374657	Bangla	2500	614	educational
□	Edit	Copy	Delete	3004	28374658	Philosophy	1500	334	educational
□	Edit	Copy	Delete	3011	28374655	Electrical Instrument	2494	315	educational

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

Update book
SET title = "Data Structure" , Price= 3467
WHERE title = "Bangla" OR pagesNo = 614;

After updating:



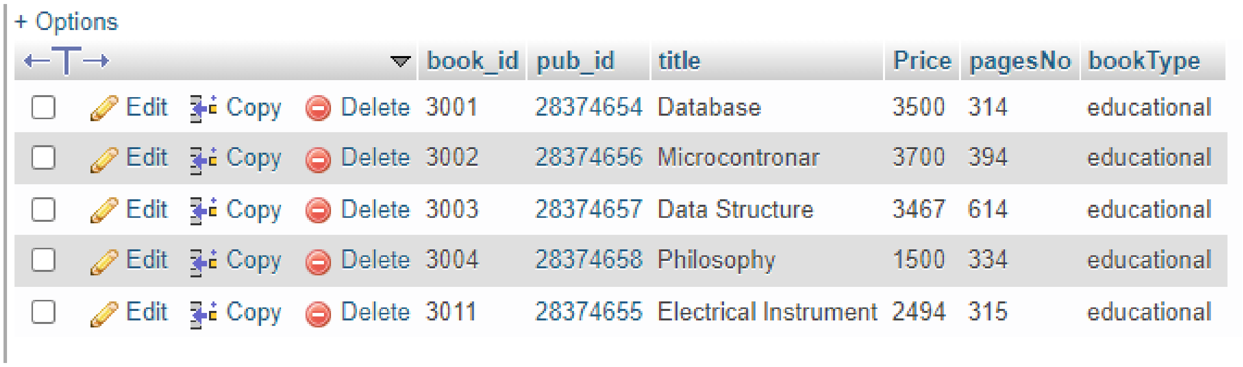
+ Options

←T→			▼	book_id	pub_id	title	Price	pagesNo	bookType
□	Edit	Copy	Delete	3001	28374654	Database	3500	314	educational
□	Edit	Copy	Delete	3002	28374656	Microcontronar	3700	394	educational
□	Edit	Copy	Delete	3003	28374657	Data Structure	3467	614	educational
□	Edit	Copy	Delete	3004	28374658	Philosophy	1500	334	educational
□	Edit	Copy	Delete	3011	28374655	Electrical Instrument	2494	315	educational

- **NOT**

The MySQL NOT Condition (also called the NOT Operator) is used to negate a condition in a SELECT, INSERT, UPDATE, or DELETE statement.

Before updating:

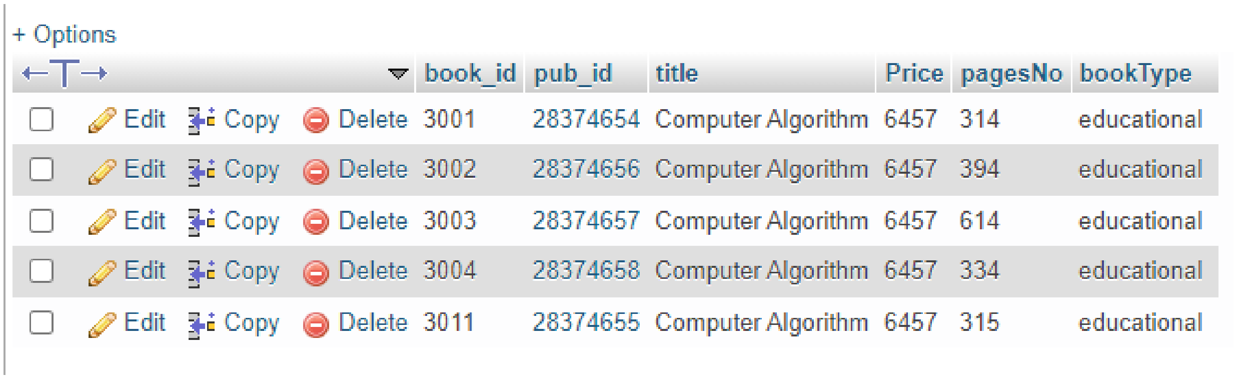


+ Options

←T→			▼	book_id	pub_id	title	Price	pagesNo	bookType
□	Edit	Copy	Delete	3001	28374654	Database	3500	314	educational
□	Edit	Copy	Delete	3002	28374656	Microcontronar	3700	394	educational
□	Edit	Copy	Delete	3003	28374657	Data Structure	3467	614	educational
□	Edit	Copy	Delete	3004	28374658	Philosophy	1500	334	educational
□	Edit	Copy	Delete	3011	28374655	Electrical Instrument	2494	315	educational

```
Update book
SET title = "Computer Algorithm" , Price= 6457
WHERE NOT book_id =  3004;
```

After updating:



+ Options

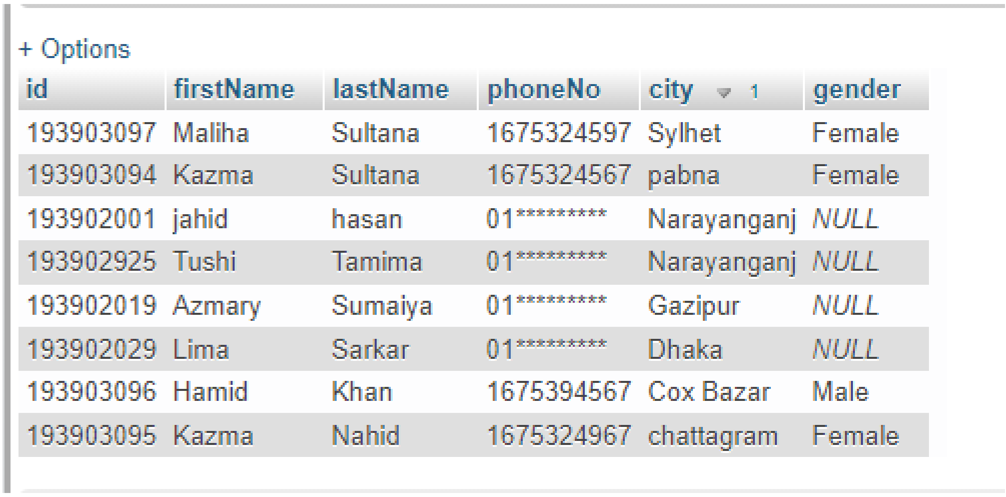
←T→			▼	book_id	pub_id	title	Price	pagesNo	bookType
□	Edit	Copy	Delete	3001	28374654	Computer Algorithm	6457	314	educational
□	Edit	Copy	Delete	3002	28374656	Computer Algorithm	6457	394	educational
□	Edit	Copy	Delete	3003	28374657	Computer Algorithm	6457	614	educational
□	Edit	Copy	Delete	3004	28374658	Computer Algorithm	6457	334	educational
□	Edit	Copy	Delete	3011	28374655	Computer Algorithm	6457	315	educational

□ **Implementing limit, Order BY, IN, Not in, Between, Not between in MYSQL.**

- **Limit:**

The SQL SELECT LIMIT statement is used to retrieve records from one or more tables in a database and limit the number of records returned based on a limit value.

Before:

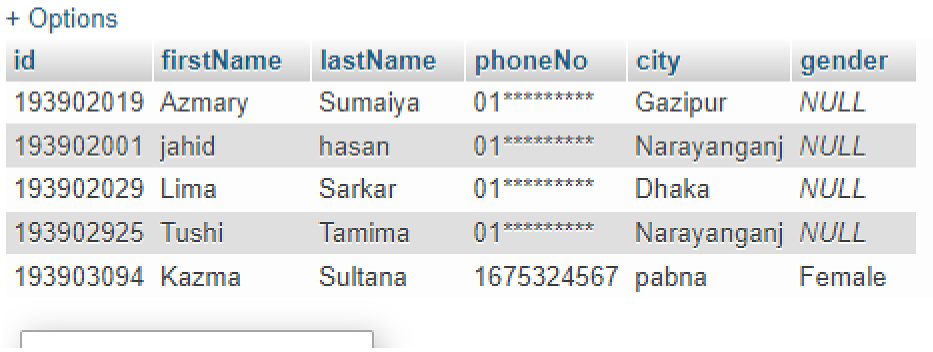


id	firstName	lastName	phoneNo	city ▾ 1	gender
193903097	Maliha	Sultana	1675324597	Sylhet	Female
193903094	Kazma	Sultana	1675324567	pabna	Female
193902001	jahid	hasan	01*********	Narayanganj	*NULL*
193902925	Tushi	Tamima	01*********	Narayanganj	*NULL*
193902019	Azmary	Sumaiya	01*********	Gazipur	*NULL*
193902029	Lima	Sarkar	01*********	Dhaka	*NULL*
193903096	Hamid	Khan	1675394567	Cox Bazar	Male
193903095	Kazma	Nahid	1675324967	chattagram	Female

➔ **Query :**

SELECT * FROM `info` LIMIT 5 ;

After:



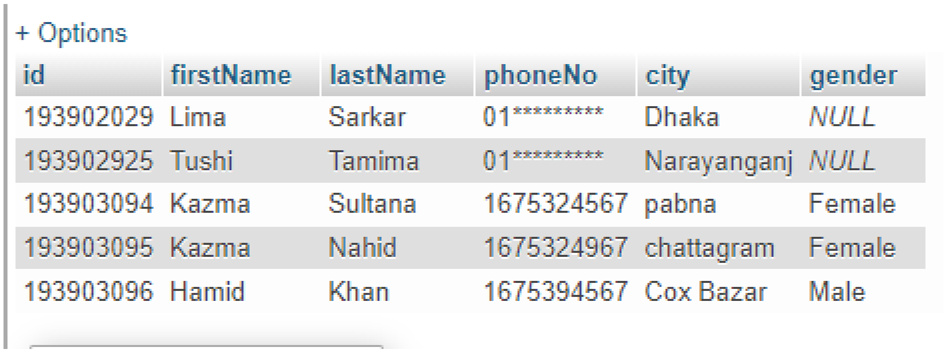
id	firstName	lastName	phoneNo	city	gender
193902019	Azmary	Sumaiya	01*********	Gazipur	*NULL*
193902001	jahid	hasan	01*********	Narayanganj	*NULL*
193902029	Lima	Sarkar	01*********	Dhaka	*NULL*
193902925	Tushi	Tamima	01*********	Narayanganj	*NULL*
193903094	Kazma	Sultana	1675324567	pabna	Female

➔ **Query :**

SELECT * FROM `info` LIMIT 2, 5 ;

(This limit 2,5 means that this query will ignore 1st 2 row then show next 5 rows .)

After :



id	firstName	lastName	phoneNo	city	gender
193902029	Lima	Sarkar	01*********	Dhaka	*NULL*
193902925	Tushi	Tamima	01*********	Narayanganj	*NULL*
193903094	Kazma	Sultana	1675324567	pabna	Female
193903095	Kazma	Nahid	1675324967	chattagram	Female
193903096	Hamid	Khan	1675394567	Cox Bazar	Male

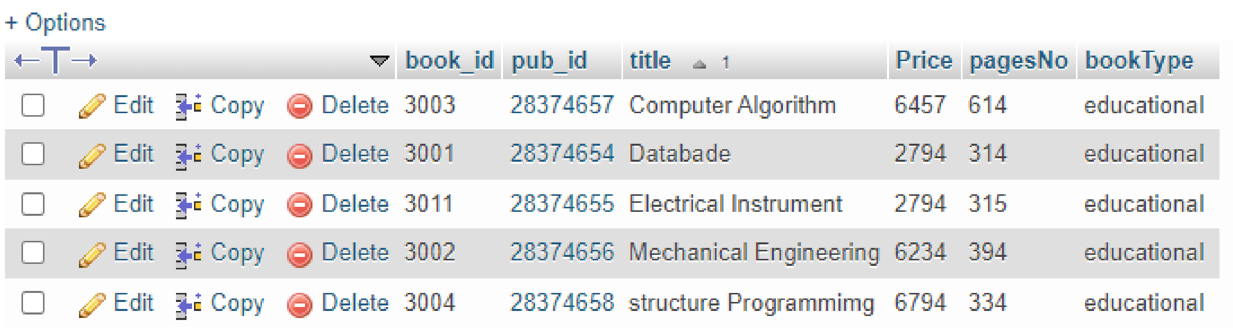
- ## Order BY:

The ORDER BY statement in sql is used to sort the fetched data in either ascending or descending according to one or more columns. By default ORDER BY sorts the data in ascending order. We can use the keyword DESC to sort the data in descending order and the keyword ASC to sort in ascending order.

➔ Query:

SELECT * FROM `book` ORDER by title;

Output :

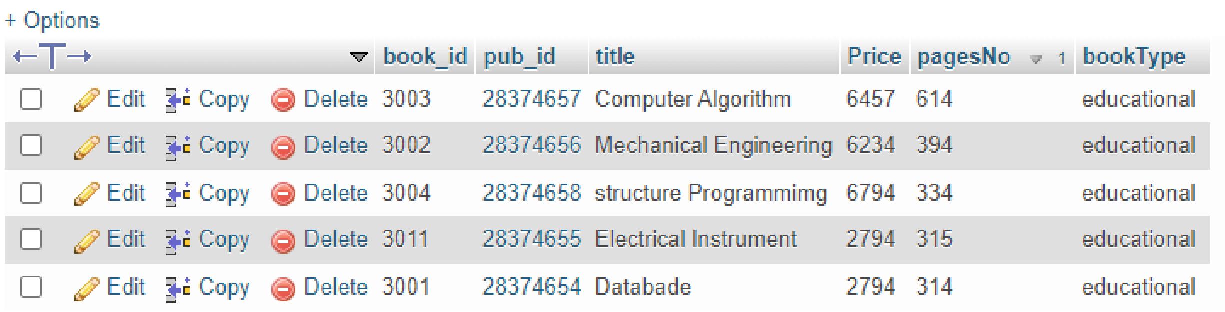


				book_id	pub_id	title ▲ 1	Price	pagesNo	bookType
□	Edit	Copy	Delete	3003	28374657	Computer Algorithm	6457	614	educational
□	Edit	Copy	Delete	3001	28374654	Databade	2794	314	educational
□	Edit	Copy	Delete	3011	28374655	Electrical Instrument	2794	315	educational
□	Edit	Copy	Delete	3002	28374656	Mechanical Engineering	6234	394	educational
□	Edit	Copy	Delete	3004	28374658	structure Programmimg	6794	334	educational

➔ Query:

SELECT * FROM `book` ORDER by pagesNo DESC;

Output :



	book_id	pub_id	title	Price	pagesNo	bookType
□ Edit Copy Delete	3003	28374657	Computer Algorithm	6457	614	educational
□ Edit Copy Delete	3002	28374656	Mechanical Engineering	6234	394	educational
□ Edit Copy Delete	3004	28374658	structure Programmimg	6794	334	educational
□ Edit Copy Delete	3011	28374655	Electrical Instrument	2794	315	educational
□ Edit Copy Delete	3001	28374654	Databade	2794	314	educational

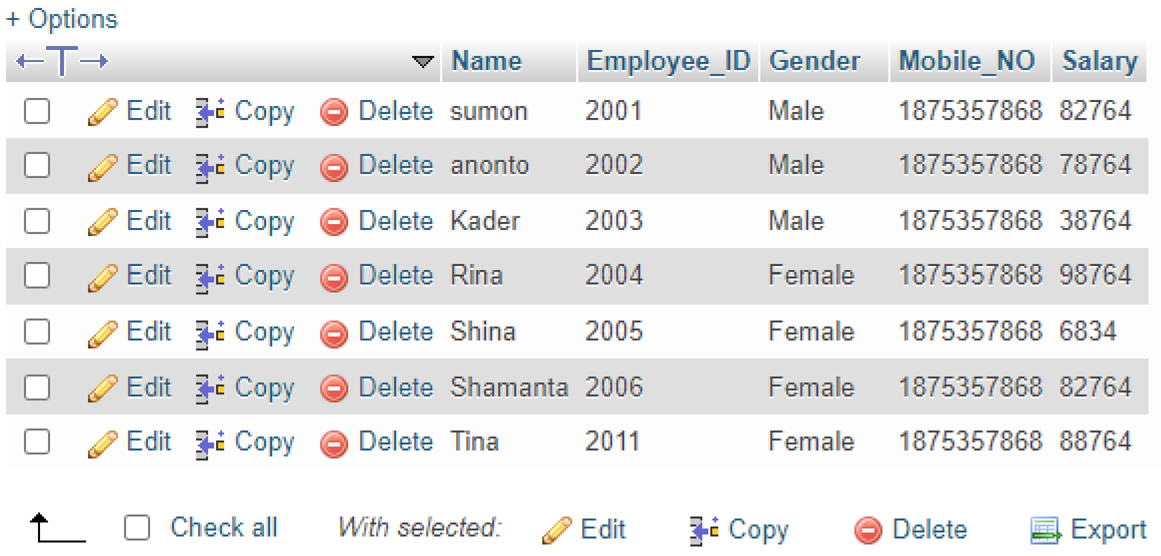
- **IN :**

The SQL IN condition (sometimes called the IN operator) allows us to easily test if an expression matches any value in a list of values.

➔ **Query :**

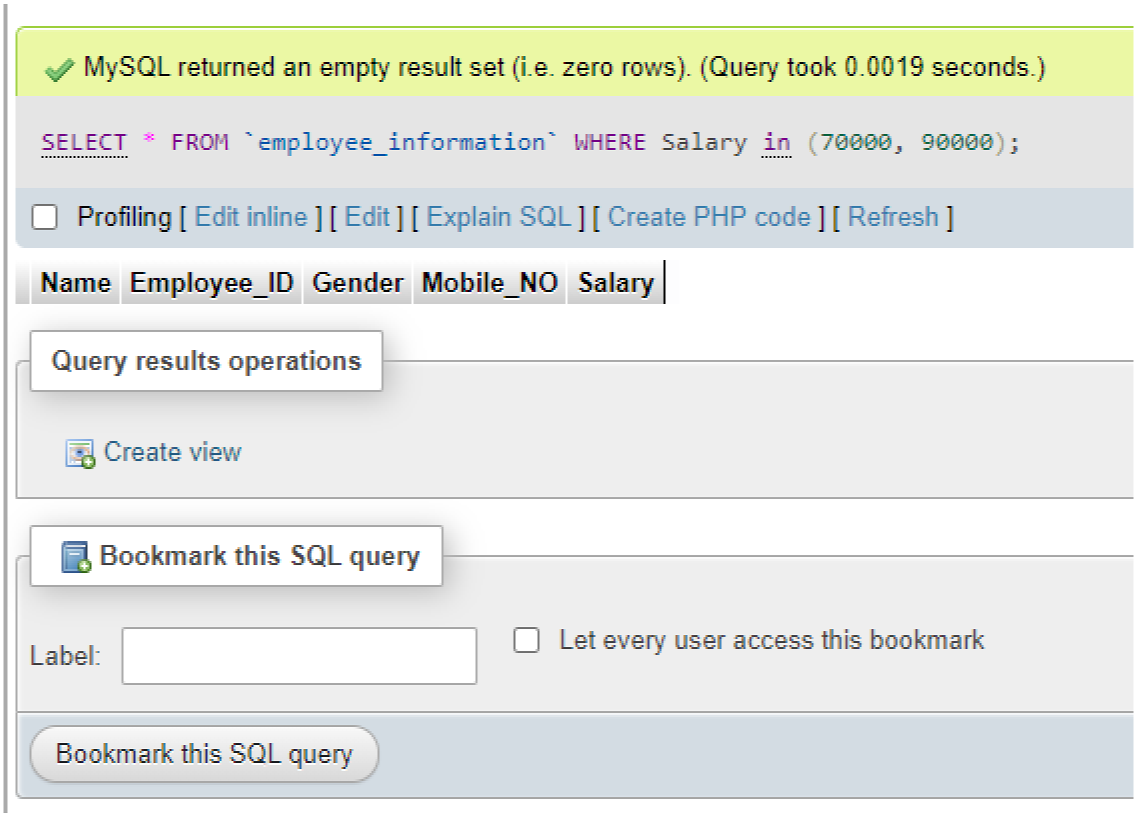
SELECT * FROM `employee_information` WHERE Salary in (70000, 90000);

Information Set :



	Name	Employee_ID	Gender	Mobile_NO	Salary
□ Edit Copy Delete	sumon	2001	Male	1875357868	82764
□ Edit Copy Delete	anonto	2002	Male	1875357868	78764
□ Edit Copy Delete	Kader	2003	Male	1875357868	38764
□ Edit Copy Delete	Rina	2004	Female	1875357868	98764
□ Edit Copy Delete	Shina	2005	Female	1875357868	6834
□ Edit Copy Delete	Shamanta	2006	Female	1875357868	82764
□ Edit Copy Delete	Tina	2011	Female	1875357868	88764

Output :



✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0019 seconds.)

```
SELECT * FROM `employee_information` WHERE Salary in (70000, 90000);
```

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

Name	Employee_ID	Gender	Mobile_NO	Salary

Query results operations
Create view
Bookmark this SQL query
Label:
□ Let every user access this bookmark
Bookmark this SQL query

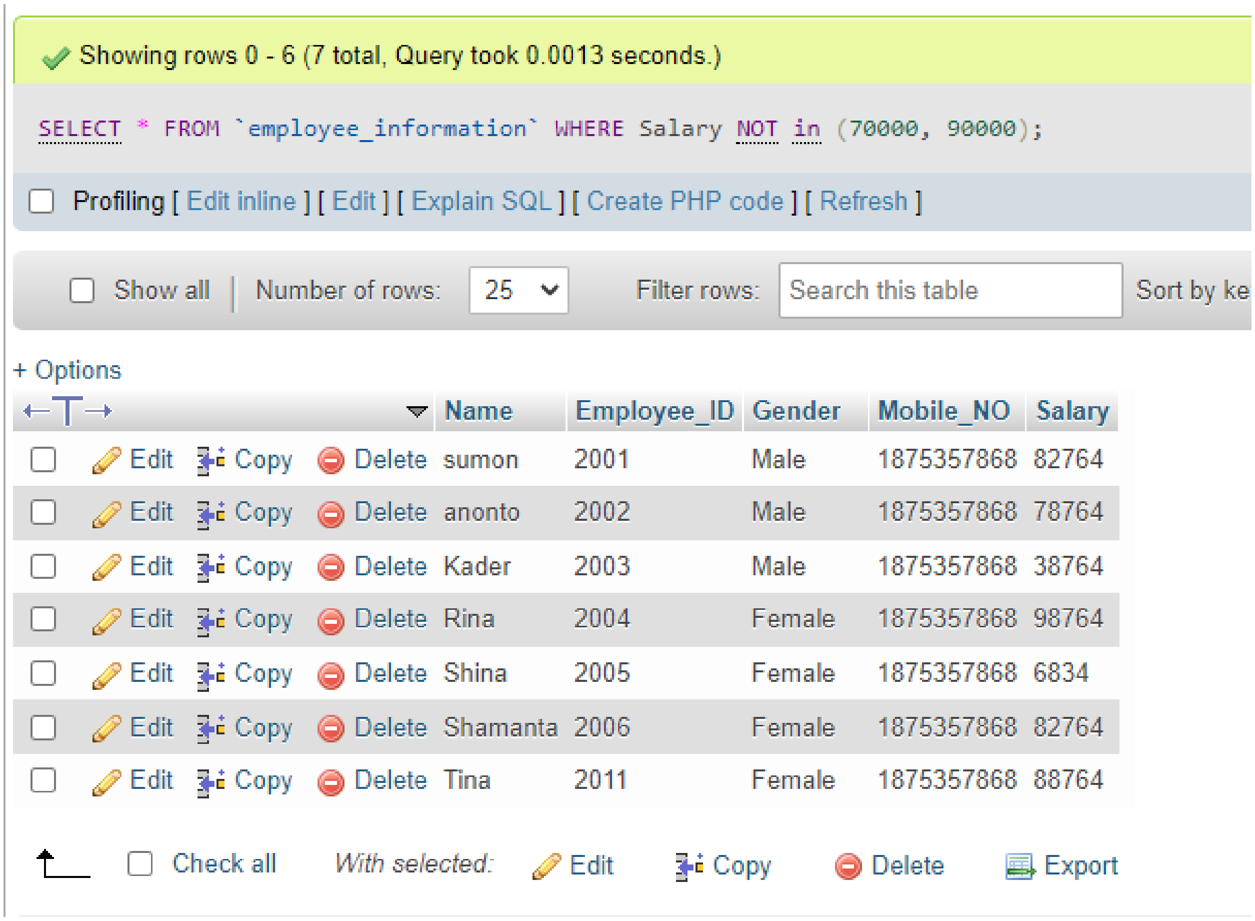
- **NOT IN :**

SQL NOT IN operator is used to filter the result if the values that are mentioned as part of the IN operator are not satisfied.

➔ **Query :**

SELECT * FROM `employee_information` WHERE Salary NOT in (70000, 90000);

Output :



✓ Showing rows 0 - 6 (7 total, Query took 0.0013 seconds.)

```
SELECT * FROM `employee_information` WHERE Salary NOT in (70000, 90000);
```

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

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

+ Options

	Name	Employee_ID	Gender	Mobile_NO	Salary
□ Edit Copy Delete	sumon	2001	Male	1875357868	82764
□ Edit Copy Delete	anonto	2002	Male	1875357868	78764
□ Edit Copy Delete	Kader	2003	Male	1875357868	38764
□ Edit Copy Delete	Rina	2004	Female	1875357868	98764
□ Edit Copy Delete	Shina	2005	Female	1875357868	6834
□ Edit Copy Delete	Shamanta	2006	Female	1875357868	82764
□ Edit Copy Delete	Tina	2011	Female	1875357868	88764

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

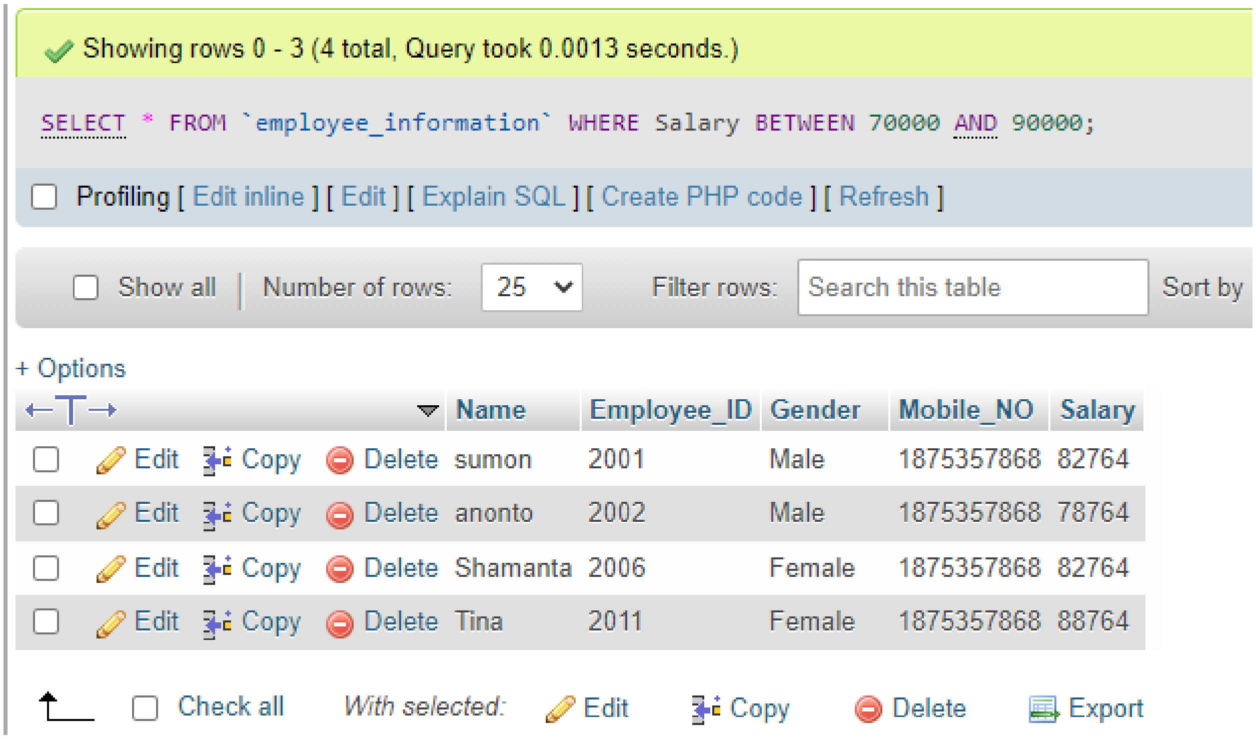
- **Between :**

The SQL BETWEEN condition allows you to easily test if an expression is within a range of values (inclusive).

➔ **Query :**

SELECT * FROM `employee_information` WHERE Salary BETWEEN 70000 AND 90000;

Output :



Showing rows 0 - 3 (4 total, Query took 0.0013 seconds.)

```
SELECT * FROM `employee_information` WHERE Salary BETWEEN 70000 AND 90000;
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
□ Show all
Number of rows: 25
Filter rows: Search this table
Sort by
+ Options
□ Check all
With selected:
Edit
Copy
Delete
Export

	Name	Employee_ID	Gender	Mobile_NO	Salary
□ Edit Copy Delete	sumon	2001	Male	1875357868	82764
□ Edit Copy Delete	anonto	2002	Male	1875357868	78764
□ Edit Copy Delete	Shamanta	2006	Female	1875357868	82764
□ Edit Copy Delete	Tina	2011	Female	1875357868	88764

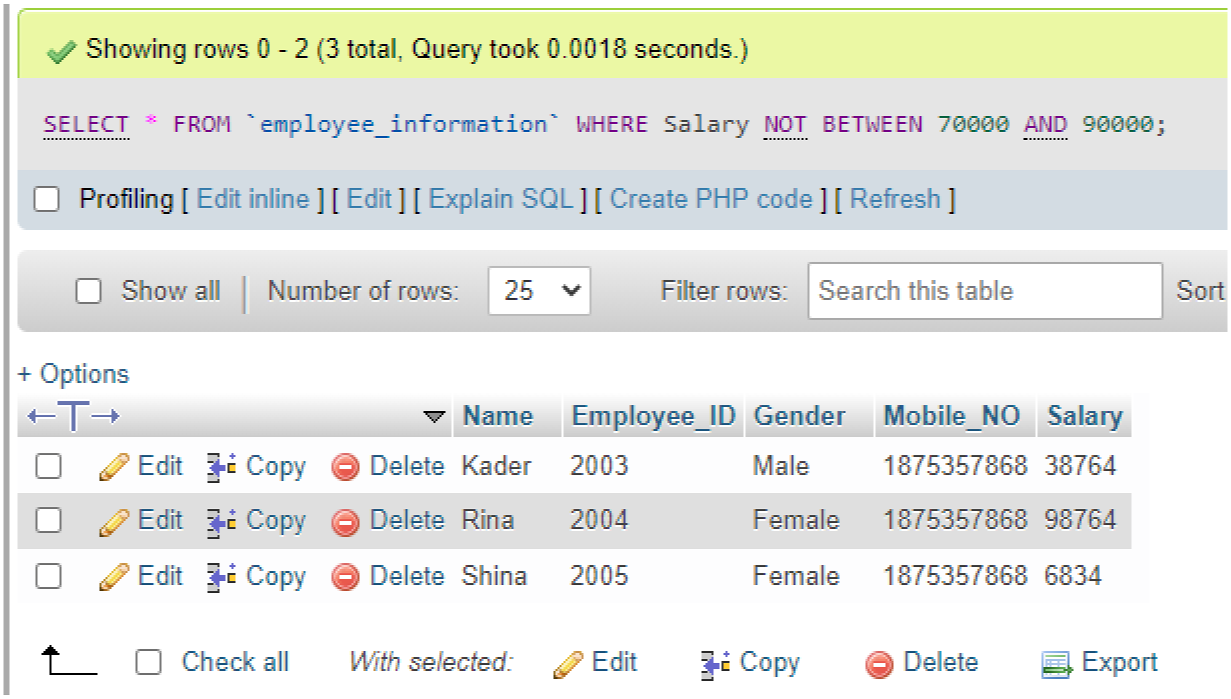
NOT Between :

The SQL NOT BETWEEN operator is used for getting the values as part of the result set which is outside of the range specified by the BETWEEN operator.

➔ Query :

SELECT * FROM `employee_information` WHERE Salary NOT BETWEEN 70000 AND 90000;

Output :



✓ Showing rows 0 - 2 (3 total, Query took 0.0018 seconds.)

```
SELECT * FROM `employee_information` WHERE Salary NOT BETWEEN 70000 AND 90000;
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
□ Show all | Number of rows: 25 Filter rows: Search this table Sort
+ Options

	Name	Employee_ID	Gender	Mobile_NO	Salary
□ Edit Copy Delete	Kader	2003	Male	1875357868	38764
□ Edit Copy Delete	Rina	2004	Female	1875357868	98764
□ Edit Copy Delete	Shina	2005	Female	1875357868	6834

□ Check all With selected: Edit Copy Delete Export

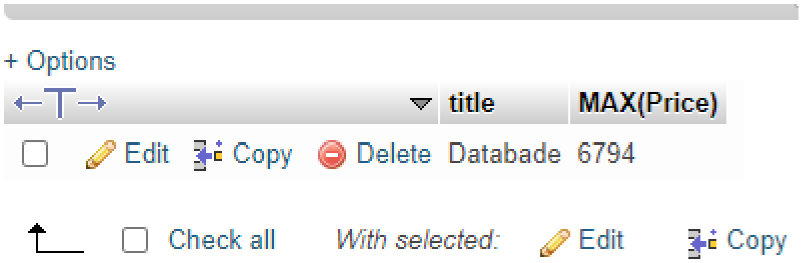
- **MAX:**

Usage. SQL MAX() aggregate function is used to return the maximum value from the provided numerical expression or the highest value in the collating sequence from the provided character expression.

➔ **Query of MAX:**

SELECT title, MAX(Price) FROM `book`

Output :



	title	MAX(Price)
□ Edit Copy Delete	Databade	6794

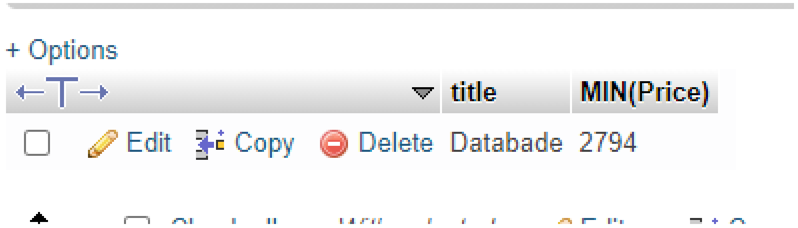
- **MIN :**

The aggregate function SQL MIN() is used to find the minimum value or lowest value of a column or expression. This function is useful to determine the smallest of all selected values of a column.

➔ **Query of MIN:**

SELECT title, MIN(Price) FROM `book`

Output:



title	MIN(Price)
Databade	2794

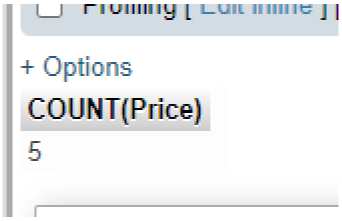
- **Count:**

The SQL COUNT() function returns the number of rows in a table satisfying the criteria specified in the WHERE clause. It sets the number of rows or non NULL column values.

➔ **Query of Count :**

SELECT COUNT(Price) FROM `book` WHERE bookType = "educational";

Output :



COUNT(Price)
5

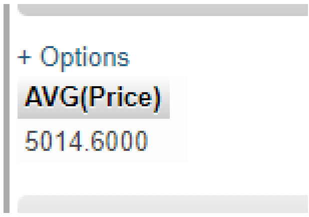
- **Average :**

SQL Server AVG() function is an aggregate function that returns the average value of a group. In this syntax: ALL instructs the AVG() function to take all values for calculation.

➔ **Query of Average:**

SELECT AVG(Price) FROM `book` WHERE bookType = "educational";

Output:



AVG(Price)
5014.6000

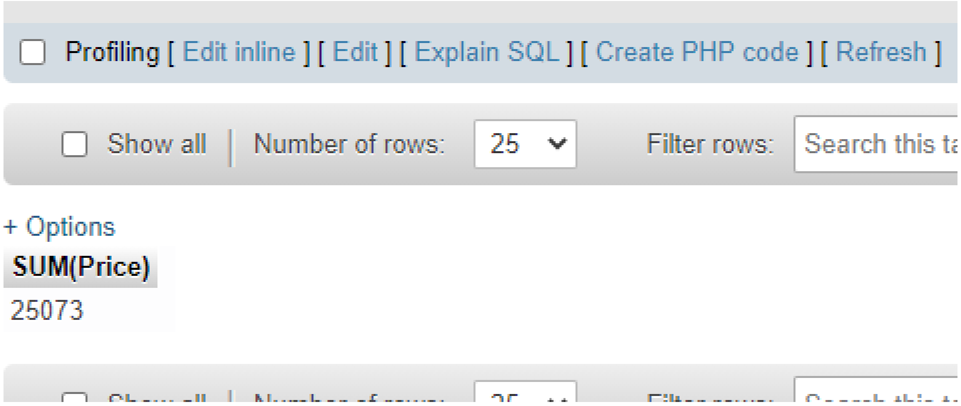
- **SUM:**

The SQL Server SUM() function is an aggregate function that calculates the sum of all or distinct values in an expression. In this syntax: ALL instructs the SUM() function to return the sum of all values including duplicates.

→ **Query of Sum:**

SELECT SUM (Price) FROM `book` WHERE bookType = "educational";

Output:


□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
□ Show all | Number of rows: 25
Filter rows: Search this ta
+ Options

SUM(Price)
25073

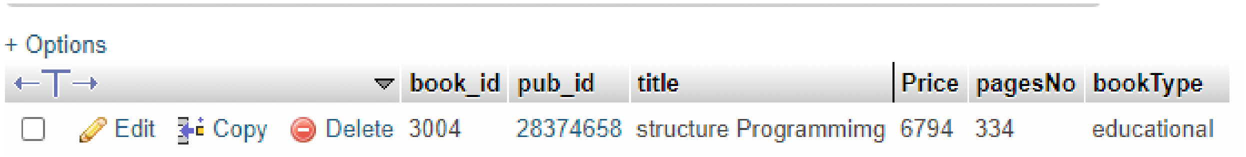
□ LIKE :

The SQL Server LIKE is a logical operator that determines if a character string matches a specified pattern. A pattern may include regular characters and wildcard characters. The LIKE operator is used in the WHERE clause of the SELECT , UPDATE , and DELETE statements to filter rows based on pattern matching.

➔ Query of LIKE:

```
SELECT * FROM `book` WHERE title like "st%" ;
```

Output :



←T→ ▼	book_id	pub_id	title	Price	pagesNo	bookType
□ Edit Copy Delete	3004	28374658	structure Programmimg	6794	334	educational

.

6. ANALYSIS AND DISCUSSION [2]

At first I implemented all Query Using different relational operators Like, greater than or equal, <>, String Match with condition, String match with logical operators. Then I Updated data information by using And, or, not (logical operator). After that I have discussed Limit, Order BY, IN, Not in, Between, Not between. Here Limit And Order By works properly but when I tried to implement IN, NOT IN, BETWEEN, NOT BETWEEN operators then it doesn't work properly. I have faced some difficulties. Lastly I Implement Like operator by query for searching anything.

7. SUMMARY:

In this lab report I tried to implement all the topics that I learned from the previous 210 lab class. Here is a simple thing that if we just remember the keywords and how to initialize them then everyone will be able to handle a database and make a database without any hesitation.If we know the proper querying system then our database management will be more easy to handle.