MySQL TUTORIAL

About the Tutorial

SQL is a database computer language designed to retrieve and manage data in relational databases. SQL stands for Structured Query Language.This tutorial gives you a quick overview of SQL.It covers most of the topics needed to understand the basics of SQL and how SQL works.

Audience

This tutorial is intended for beginners to help you understand the basic and advanced concepts associated with the SQL language. This tutorial will give you a good understanding of the various components of SQL and give good examples.

Prerequisites

Before we start with the various types of examples in this tutorial,we assume that you already know what a database is, specifically an RDBMS, and what a computer programming language is.

Compile/Execute SQL Programs

Here for SQL programs we're using XAMPP. There are many more options for SQL like ORACLE and so on.To be concerned with our own comfort zone ,we are using XAMPP software.Before doing SQL programming directly ,we need simple

knowledge of MYSQL.If anyone need any help of database programs,this link(MYSQL-Administration) would help them out.

Contents

About the Tutorial..... 02
Audience..... 02
Prerequisites..... 02
Compile/Execute SQL Programs..... 02
Table of Contents.....03-05
SQL-Overview.....05
What is SQL?.....05
SQL Process.....05-06
SQL Commands.....07
Database.....08
What is Data?.....08
What is Database?.....08
What is a Database Management System (DBMS)?.....09
SHOW,CREATE AND DELETE DATABASE.....09-14
SQL — RDBMS Concepts.....14-15
What is RDBMS?.....14
What is a NULL value?.....14
SQL Constraints.....14-15

TABLE.....15
What is a table?.....15
What is a field?.....15
What is a Record or a Row?.....15-16
What is a column?.....16
Data Type.....17
CREATE,RENAME AND DROP TABLE.....17-19
DATA INSERTION.....22
SELECTSTATEMENT.....27-31
DISTINCT,LIMIT,Keyword.....31-38
ORDER BY keyword 38-41
Arithmetic Operator.....42-45

WHERE Clause.....45-46
Distinct Clause.....47-48
RELATIONAL Operator.....49
BETWEEN Operator.....49-51.
Logical Operator.....51-52
And ,Or,Not operator.....52-58
In operator 58-61
Like operator 62-63
As Keyword 63-65
Upper(),Lower() and concat Function.....65-71
Aggregate Functions.....71
Count () 72-75
Sum ().....75-76.
Avg () 76-77
Max().....77-78
Min ().....79
Group By Clause.....80-81
Delete Statement 81-82
Alter Statement.....82
Add column 82-83
Drop column.....83-84
Update Statement 84-86
Index Statement.....86
Create index statement 86-87
Show index statement 87
Drop Index statement 87-88
Foriegn key.....88
Auto Increment Keyword 88-91
Joining Table 91-93
Join Clause.....93-95
Inner Join 95-97
Left join 97-98
Right Join 99-100
Union Operator.....101-103
Union all.....103-104
Having Clause.....104-105
Exists operator.....105-106
Any Operator.....107-108
Select Into Statement.....108-109
Case statement.....109-110
VIEW STATEMENT.....111-113
Update view 113
Date Function..... 114

Now function.....114
Curdate function..... ...114
Curtime Function.....114
Date function.....114-115
Day Name 115-116
Add Date.....116-117

SQL – Overview ::::

SQL is a language for database mining;it includes database creation, deletion, row retrieval, row modification, etc. SQL is an ANSI (American National Standards Institute) standard language,but there are different versions of the SQL language.

What is SQL?

SQL is a structured query language, a computer language for storing, manipulating, and retrieving data stored in relational databases. SQL is the standard language for relational database systems. All relational database management systems (RDMS) such as MySQL, MS Access, Oracle, Sybase, Informix, Postgres, and SQL Server use SQL as the default database language.

SQL Process

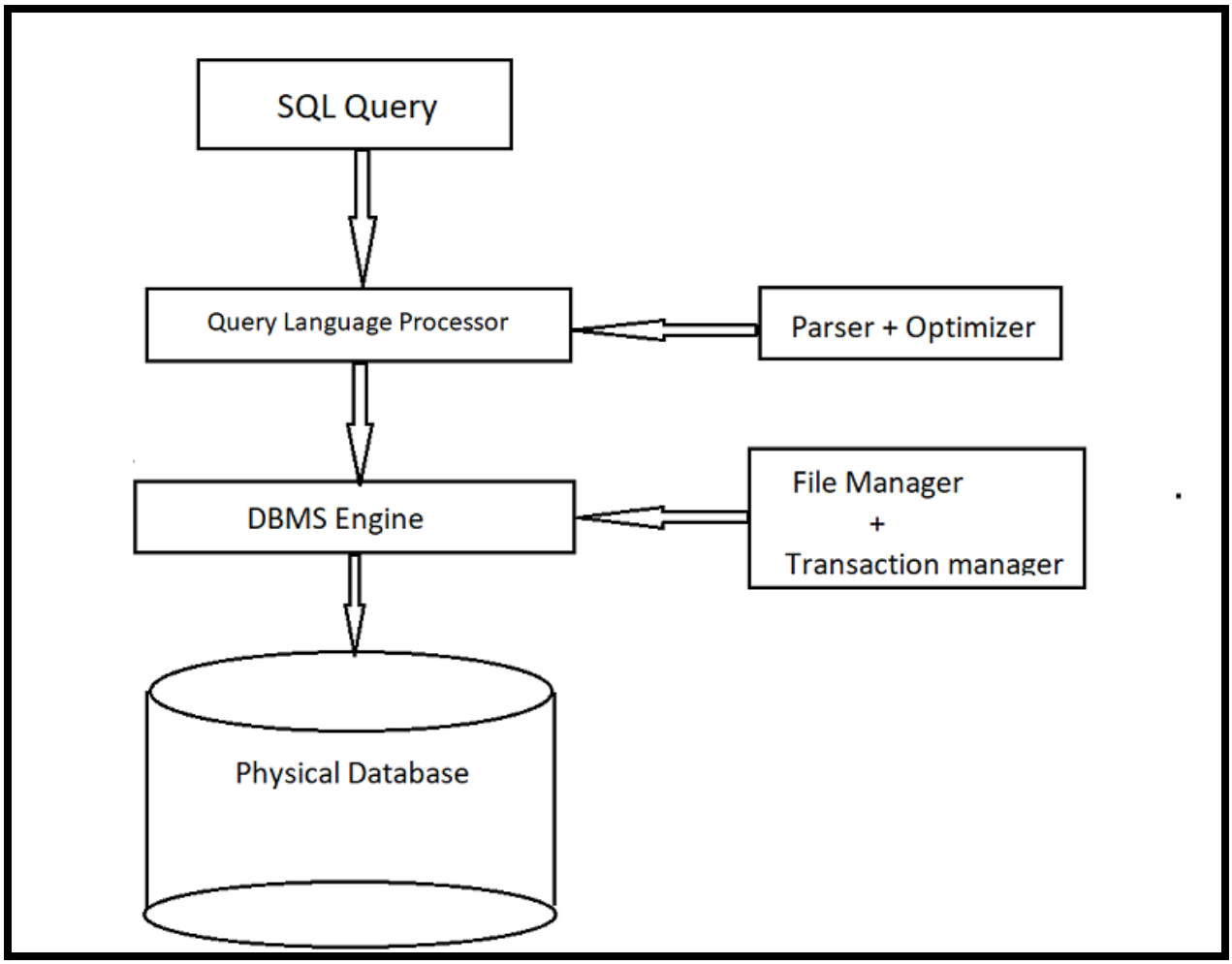
When you execute a SQL command against an RDBMS, the system determines the best way to execute the request, and the SQL engine determines how to interpret the task.This process involves various components.

These components are –

- Query dispatcher
- Optimization engine
- Traditional query engine
- SQL query engine etc.

The traditional query engine handles all non-SQL queries, but the SQL query engine does not handle logical files.

Below is a simple diagram showing the SQL architecture:



SQL Commands ::::

Standard SQL commands for interacting with relational databases are CREATE, SELECT, INSERT,UPDATE, DELETE, and DELETE. These commands can be divided into the following groups according to their type:

DDL - Data Definition Language:

Command	Description
CREATE	Creates a new table, a view of a table, or other object in the database
ALTER	Modifies an existing database object, such as a table
DROP	Deletes an entire table, a view of a table or other objects in the database.

DML - Data Manipulation Language:

Command	Description
SELECT	Retrieves certain records from one or more tables
INSERT	Creates a record.
UPDATE	Modifies records.
DELETE	Deletes records.

Database ::::

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A **database** is usually controlled by a database management system (DBMS).

What is Data?

In simple words, data can be facts related to any object in consideration. For example, your name, age, height, weight, etc. are some data related to you. A picture, image, file, pdf, etc. can also be considered data.

Database management system

A database is a systematic collection of data. They support electronic storage and manipulation of data. Databases make data management easy.

Let us discuss a database example:

An online telephone directory uses a database to store data of people, phone numbers, and other contact details. Your electricity service provider uses a database to manage billing, client-related issues, handle fault data, etc.

Let us also consider Facebook. It needs to store, manipulate, and present data related to members, their friends, member activities, messages, advertisements, and a lot more. We can provide a countless number of examples for the usage of databases.

Database Management System

Database Management System (DBMS) is a collection of programs that enable its users to access databases, manipulate data, report, and represent data. It also helps to control access to the database. Database Management Systems are not a new concept and, as such, had been first implemented in the 1960s.Charles Bachman's Integrated Data Store (IDS) is said to be the first DBMS in history. With time databases, technologies evolved a lot, while usage and expected functionalities of databases increased immensely.

☐ CREATE DATABASE :

First, we will learn how to use the "XAMPP" PHPMyadmin CREATE DATABASE statement to create a new database on a MySQL database server. The **CREATE DATABASE** statement is used to create a new SQL database.

➢ Syntax:

CREATE DATABASE ***databasename*****;**

INPUT FORMAT:

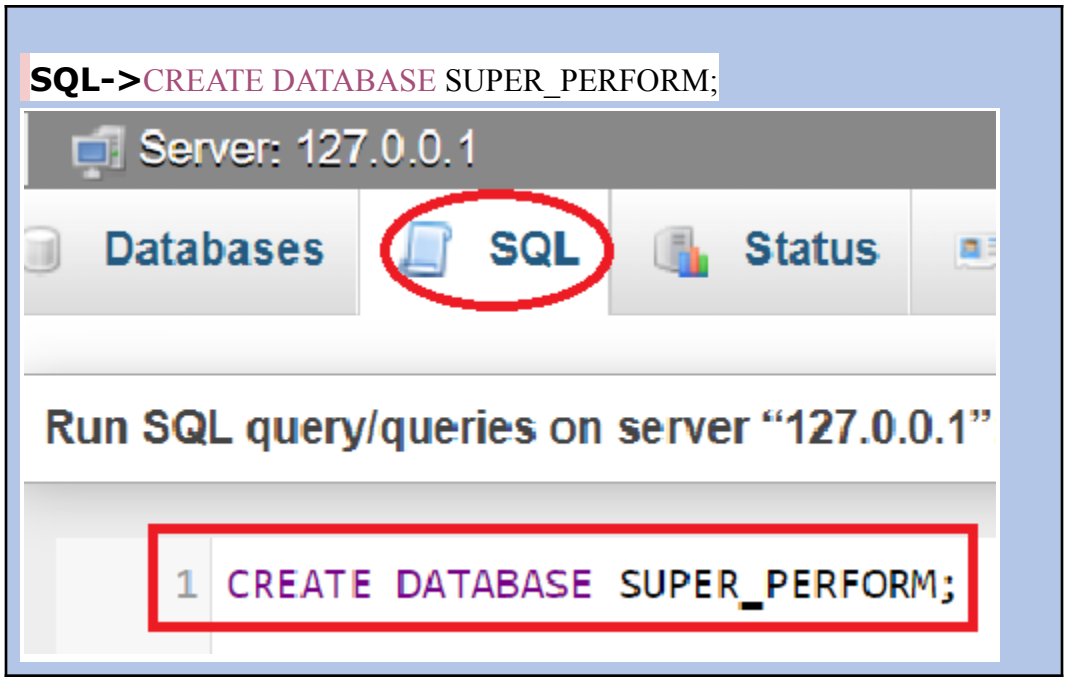


Fig:01

OUTPUT DATABASES:

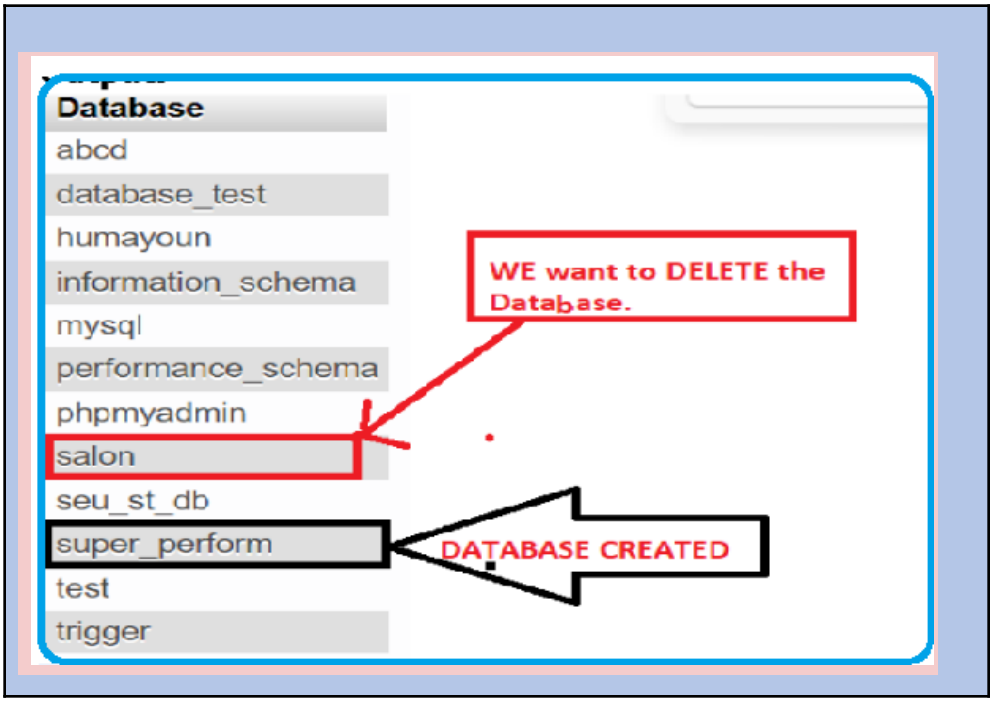


Fig:02

❑SHOW DATABASES :

To display the current databases available on the server using the SHOW DATABASES statement. This step is optional.

➢ Syntax :

SHOW DATABASES;

OUTPUT DATABASE:

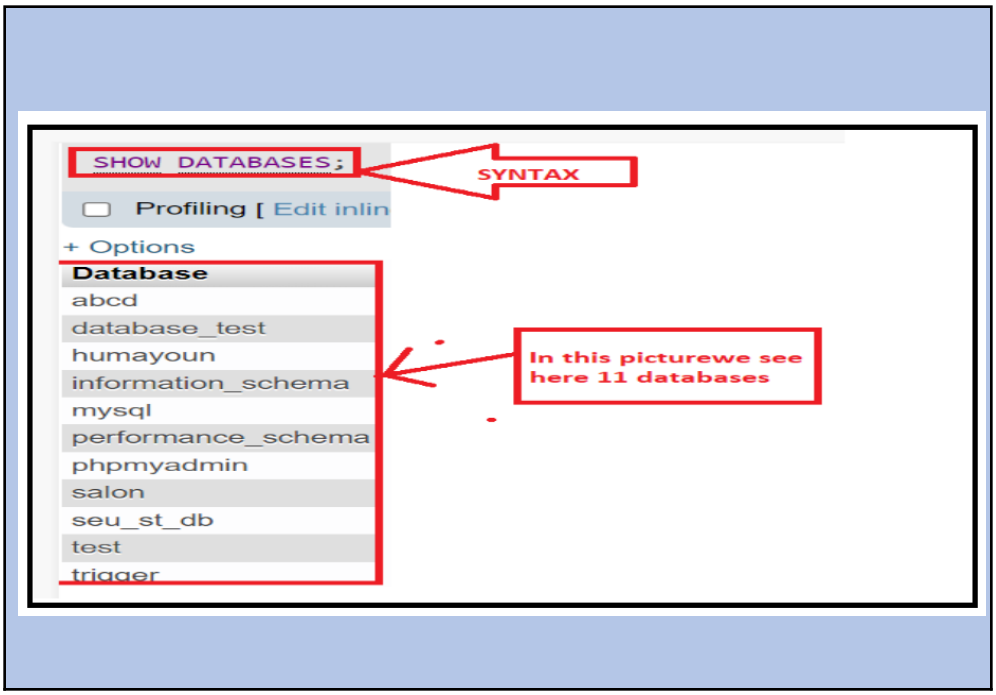


Fig-4

We see the picture SUPER_PERFORM Database has been created.

☐ DELETE DATABASE:

If we want to **delete** any databases from the database list,we will use simply delete databases sql command.

➢ **Syntex:**

DROP DATABASE *databasename*;

INPUT FORMAT:

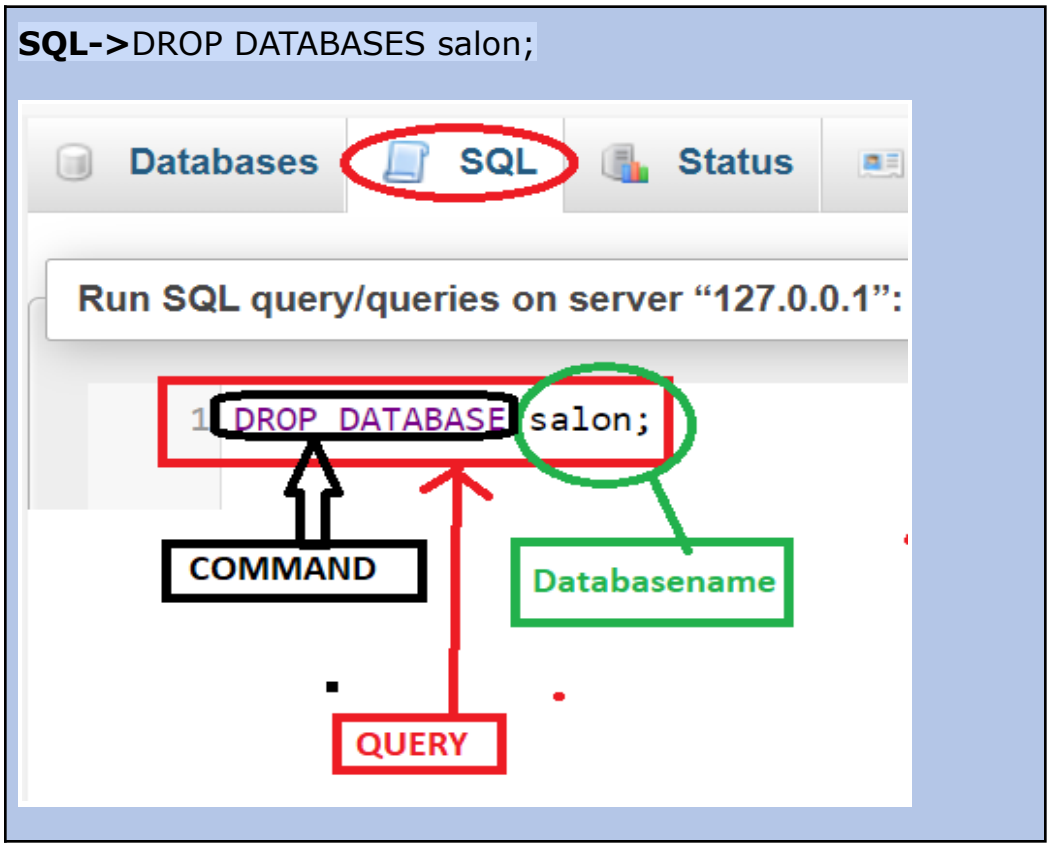


Fig:05

OUTPUT DATABASE:

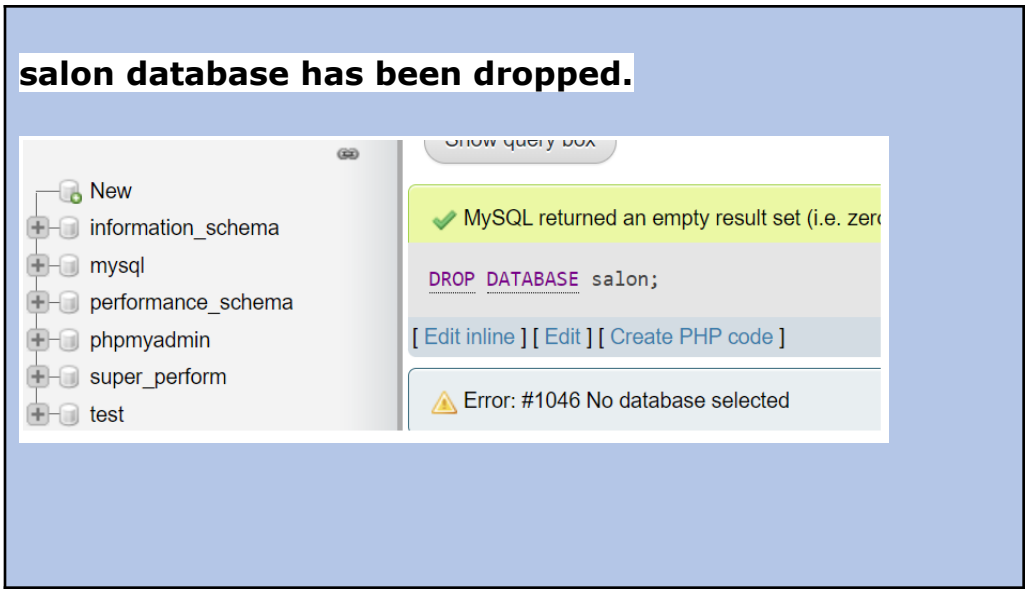


Fig-6

SQL – RDBMS Concepts

What is RDBMS?

Relational Database Management System (RDBMS) is an acronym for Relational Database Management System. SQL and all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access are built on top of RDBMS.

What is the Null value?

In a table, a NULL value is a value in a field that appears to be blank, implying that a field with a NULL value has no value. It's critical to grasp the difference between a NULL value and a zero value or a field with spaces. A field with a NULL value has been left blank throughout the record generation process.

SQL Constraints

Constraints are rules that are applied to data columns in a table. These are used to restrict the types of information that can be entered into a table. This guarantees that the data in the database is accurate and reliable.Constraints can be applied at the column or table level. Table level restrictions are applied to the entire table, whereas column level constraints are only applied to one column.

The following are some of the most regularly used SQL constraints:

- NOT NULL Constraint: This constraint prevents a column from having a NULL value.
- DEFAULT Constraint: When no value is supplied for a column, it defaults to that value.
- UNIQUE Constraint: Ensures that each value in a column is unique.
- PRIMARY KEY: Each row/record in a database table is uniquely identified by the primary key.
- FOREIGN KEY: Identifies a row/record in any other database table in a unique way.
- CHECK Constraint: The CHECK constraint guarantees that all values in a column meet particular criteria.
- INDEX: Used to easily construct and retrieve data from a database.
-

❑TABLE :

In an RDBMS, data is stored in database objects known as **tables**. This table, which has several columns and rows, is essentially a collection of connected data entries. In a relational database, a table is the most popular and basic form of data storage. A CUSTOMERS table is demonstrated in the following program:

Id	Name	Gender	Age	CGPA	City
101	Humayoun Kbir	Male	21	3.80	Rangpur
102	Naimur Rashid	Male	21	3.70	Feni
103	Abul Hasnat	Male	20	3.73	Jamalpur
104	Sanjana Akrher	Female	20	3.85	Dhaka

What is a field?

Every table is divided into fields, which are smaller entities. Id, NAME, Gender,AGE, CGPA, and City are the fields in the STUDENT table. A field is a table column that is used to store special information about each entry in the database.

Id	Name	Gender	Age	CGPA	City

What is a Record or a Row?

Each individual element in a table is referred to as a record, often known as a row of data. In the STUDENT table, for example, there are seven records. A single row of data or record from the STUDENT table is shown below:

101	Humayoun Kbir	Male	21	3.80	Rangpur

What is a column?

In a table, a **column** is a vertical item that holds all information linked with a certain field. A column in the STUDENT table, for example, is ID, which represents a location description and would look like this:

Id
101
102
103
104
105
106

Id
107
108

Data Types:

The following table summarizes the most commonly used data types supported by MySQL.

Data Type	Description
INT	Stores numeric values in the range of -2147483648 to 2147483647 .int 2 or 4 bytes
DECIMAL	Stores decimal values with exact precision.
CHAR	Stores fixed-length strings with a maximum size of 255 characters.
VARCHAR	Stores variable-length strings with a maximum size of 65,535 characters.
TEXT	Stores strings with a maximum size of 65,535 characters.
DATE	Stores date values in the YYYY-MM-DD format.
DATETIM	Stores combined date/time values in the YYYY-MM-DD HH:MM:SS format.
TIMESTAMP	Stores timestamp values. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:01' UTC).

☐CREATE TABLE :

To build a new table in a database, use the CREATE TABLE statement.In Fig-3,we create a database SUPER_PERFORM. Now ,we create a table whose name is STUDENT in this database.

➢ Syntax:

```
CREATE TABLE table_name (
        column1 datatype(size),
        column2 datatype(size),
        column3 datatype(size),
  ....
);
```

INPUT FORMAT:

```
SQL->CREATE TABLE STUDENT
(
  Id int NOT NULL,
  Name varchar(15),
  Gender varchar(10),
  Age int(5),
  CGPA double(3,2),
  City varchar(15),
  PRIMARY KEY(Id)
);
```

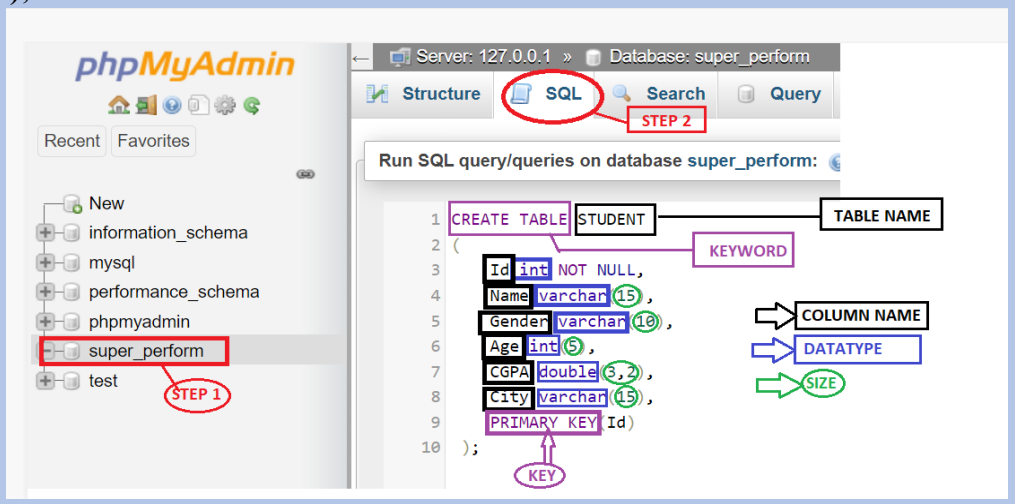


Fig-8

OUTPUT TABLE: **STUDENT** table has been created.

OUTPUT DATABASE :

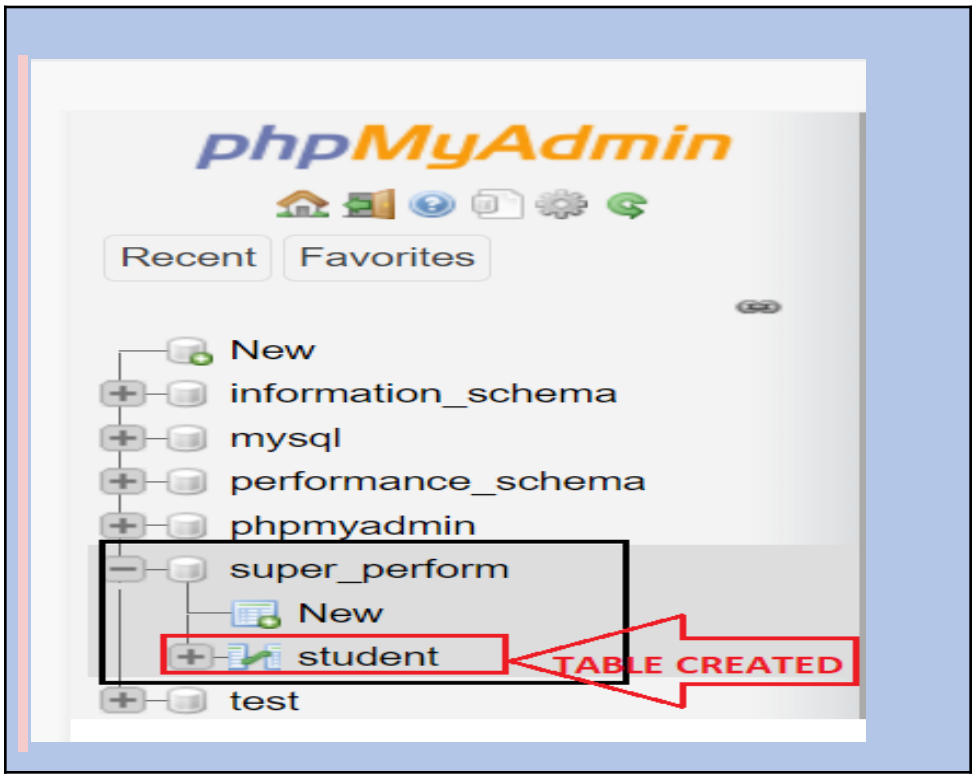


Fig-9

☐ SHOW TABLES:

If we want to see the TABLE,we have to use the show table query.

➢ Syntex:

```
SHOW TABLES;
```

Output Format:

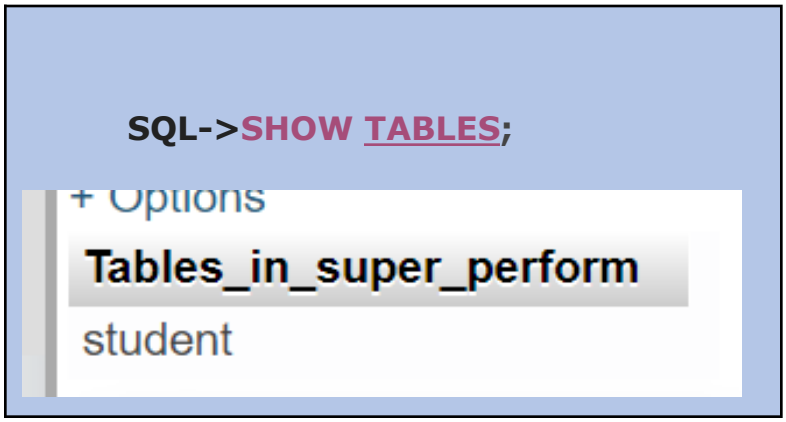


Fig-10

☐ RENAME TABLE:

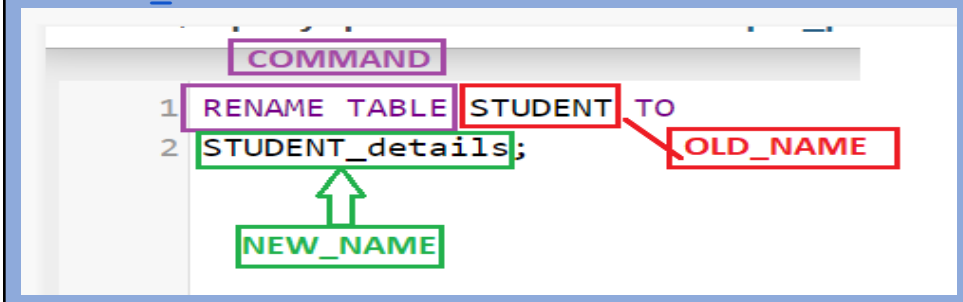
Table can be renamed with the RENAME command.It simply changes the name of the databases.

➢ Syntax:

```
RENAME TABLE old_name TO new_name;
```

INPUT FORMAT:

```
SQL->RENAME TABLE STUDENT TO
STUDENT_details;
```



OUTPUT DATABASE:

```
SQL->SHOW TABLES;
```

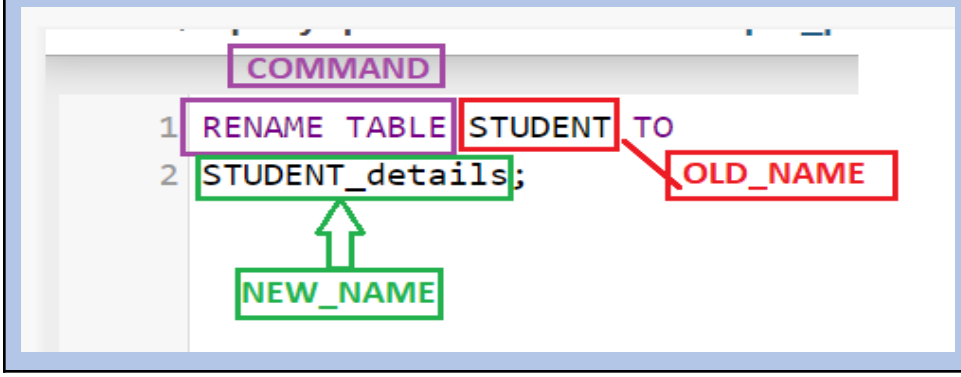


Fig-11

❑ DROP DATABASES:

Tables can be deleted with the DROP command.After this query ,no table will be shown

➢ Syntax:

```
DROP TABLE database_name;
```

INPUT FORMAT:

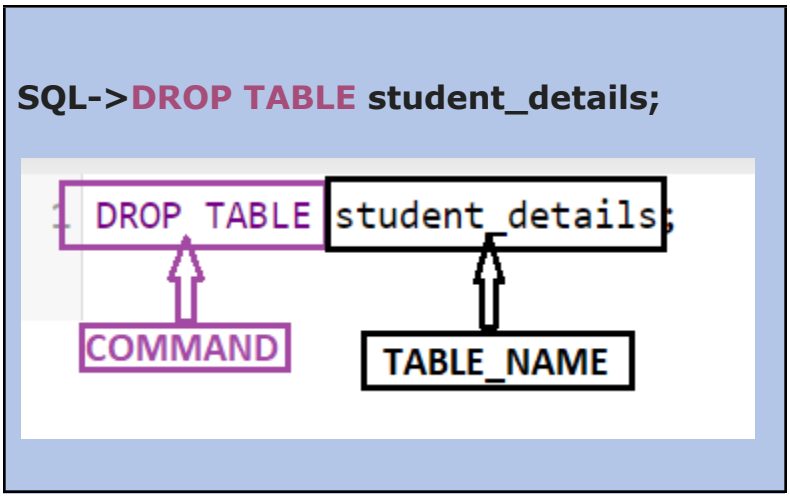


Fig-13

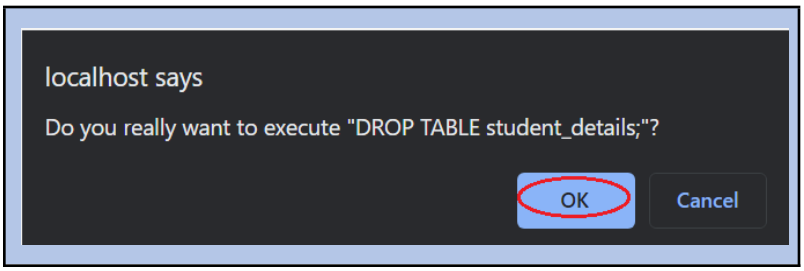


Fig-14

OUTPUT DATABASE:

We can't see any table in the super_perform database.

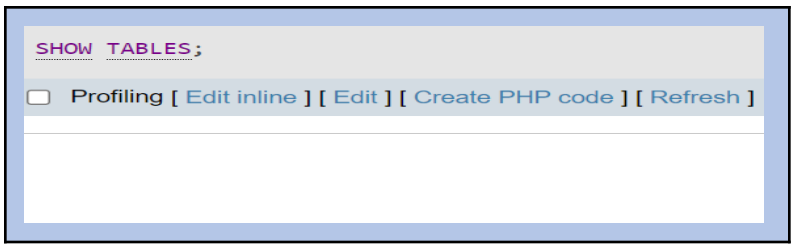


Fig-15

❑DATA INSERTION

After creating a table,**INSERT INTO** statements are used to insert data.It is possible to write the I**NSERT INTO s**tatement in two ways:

- Specify both the column names and the values to be inserted.
- If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the INSERT INTO syntax would be as follows.

➢ Syntax:

```
INSERT INTO table_name (column1, column2, column3,
...columnN)
VALUES (value1, value2, value3, ...valueN);
```

OR

```
INSERT INTO table_name VALUES (value1, value2,
value3, ...valueN);
```

Input Format :

```
SQL->INSERT INTO student_detailes
(Id,Name,Gender,Age,CGPA,City)
VALUES(101,'Humayoun Kobir','Male',21,3.80,'Rangpur');
```

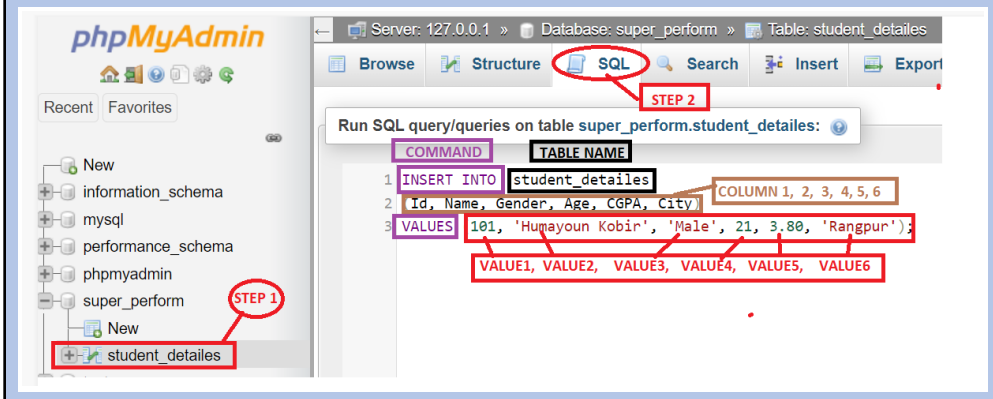


Fig-16

OR,another way:

```
SQL->INSERT INTO student_detailes
VALUES(102,'Naimur Rashid','Male',21,3.70,'Feni') ;
```

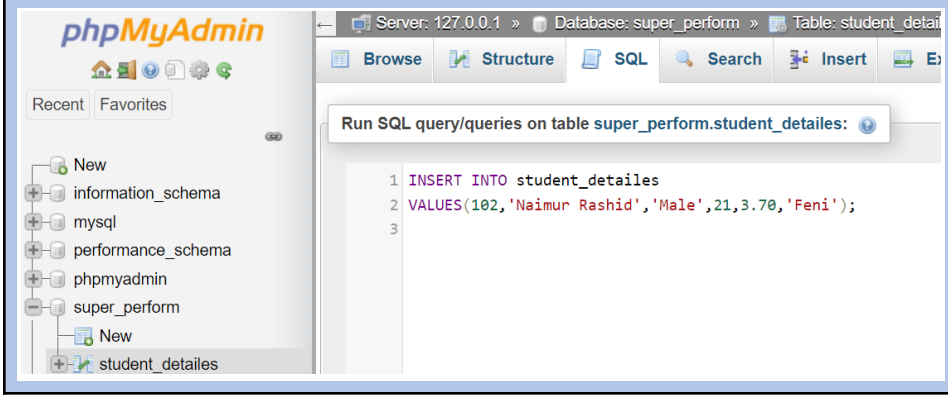


Fig-17

Another INPUT Format :

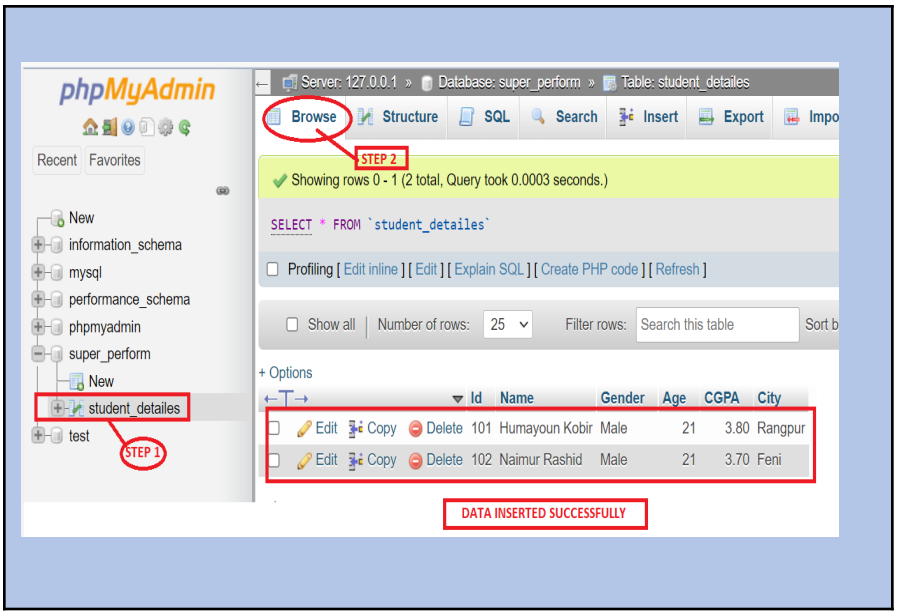


Fig-18

SQL-> **INSERT INTO student_detailes(Id,Name,Gender,Age,CGPA,City)VALUES**
(103,'AbulHasnat','Male',20,3.73,'Jamalpur'),
(104,'Sanjana Akther','Female',20,3.85,'Dhaka'),
(105,'Abdullah al Mamun','Male',22,3.50,'Gopalgjanj'),
(106,'Farzana Afroz','Female',23,3.65,'Jashore'),
(107,'Al Imran Munna','Male',25,3.30,'Khulna'),
(108,'Yeasin Arafat','Male',20,3.70,'Cumilla');

```
1  INSERT INTO student_detailes
2  (Id,Name,Gender,Age,CGPA,City)
3  VALUES
4  (103,'Abul Hasnat','Male',20,3.73,'Jamalpur'),
5  (104,'Sanjana Akther','Female',20,3.85,'Dhaka'),
6  (105,'Abdullah al Mamun','Male',22,3.50,'Gopalgjanj'),
7  (106,'Farzana Afroz','Female',23,3.65,'Jashore'),
8  (107,'Al Imran Munna','Male',25,3.30,'Khulna'),
9  (108,'Yeasin Arafat','Male',20,3.70,'Cumilla');
10
```

Fig-19

Output database:

Id	Name	Gender	Age	CGPA	City
101	Humayoun Kobir	Male	21	3.80	Rangpur
102	Naimur Rashid	Male	21	3.70	Feni
103	Abul Hasnat	Male	20	3.73	Jamalpur
104	Sanjana Akther	Female	20	3.85	Dhaka
105	Abdullah al Mam	Male	22	3.50	Gopalgjanj
106	Farzana Afroz	Female	23	3.65	Jashore
107	Al Imran Munna	Male	25	3.30	Khulna
108	Yeasin Arafat	Male	20	3.70	Cumilla

Fig-20

OUTPUT TABLE:

Id	Name	Gender	Age	CGPA	City
101	Humayoun Kbir	Male	21	3.80	Rangpur
102	Naimur Rashid	Male	21	3.70	Feni
103	Abul Hasnat	Male	20	3.73	Jamalpur
104	Sanjana Akrher	Female	20	3.85	Dhaka
105	Abdullah Al Mam	Male	22	3.50	Gopalganj
106	Farzana Afroz	Female	23	3.65	Jashore
107	All Imran Munna	Male	25	3.30	Khulna
108	Yeasin Arafat	Male	20	3.70	Comilla

☐ SELECT STATEMENT:

The required information can be found from the table with the help of a SELECT statement.SQL SELECT Statement is used to **fetch** the data from a database table which returns data in the form of a result table. These result tables are called **result-sets**.There are a lot of varieties of **select** commands in SQL

➢ Syntax:

SELECT ***column1, column2, ...***
FROM ***table_name*****;**

Here, column1, column2, ... are the field names of the table we want to select data from.

+

INPUT FORMAT:

```
SQL->SELECT Id,Name,CGPA FROM student_detailes;
```

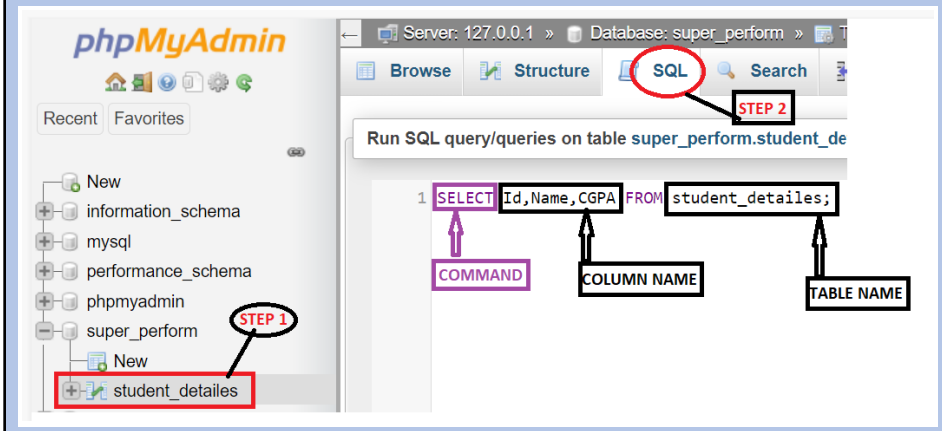


Fig-21

OUTPUT TABLE:

Id	Name	CGPA
101	Humayoun Kbir	3.80
102	Naimur Rashid	3.70
103	Abul Hasnat	3.73
104	Sanjana Akrher	3.85
105	Abdullah Al Mam	3.50
106	Farzana Afroz	3.65
107	All Imran Munna	3.30
108	Yeasin Arafat	3.70

OUTPUT DATABASE:

Id	Name	CGPA
101	Humayoun Kobir	3.80
102	Naimur Rashid	3.70
103	Abul Hasnat	3.73
104	Sanjana Akther	3.85
105	Abdullah al Mam	3.50
106	Farzana Afroz	3.65
107	Al Imran Munna	3.30
108	Yeasin Arafat	3.70

Fig:22

If you want to select all the fields available in the table, use the following syntax:

➢ **Syntax:**

```
SELECT * FROM table_name;
```

Input Format :

```
SQL->SELECT * FROM student_detailes;
```

```
1 SELECT * FROM student_detailes;
```

Fig-23

Output :

```
SELECT * FROM student_detailes;
```

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

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

Options

			Id	Name	Gender	Age	CGPA	City
Edit	Copy	Delete	101	Humayoun Kobir	Male	21	3.80	Rangpur
Edit	Copy	Delete	102	Naimur Rashid	Male	21	3.70	Feni
Edit	Copy	Delete	103	Abul Hasnat	Male	20	3.73	Jamalpur
Edit	Copy	Delete	104	Sanjana Akther	Female	20	3.85	Dhaka
Edit	Copy	Delete	105	Abdullah al Mam	Male	22	3.50	Gopalgjanj
Edit	Copy	Delete	106	Farzana Afroz	Female	23	3.65	Jashore
Edit	Copy	Delete	107	Al Imran Munna	Male	25	3.30	Khulna
Edit	Copy	Delete	108	Yeasin Arafat	Male	20	3.70	Cumilla

Fig-24

OUTPUT TABLE:

Id	Name	Gender	Age	Cgpa	City
101	Humayoun Kobir	Male	21	3.80	Rangpur
102	Naimur Rashid	Male	21	3.70	Feni
103	Abul Hasnat	Male	20	3.73	Jamalpur
104	Sanjana Akther	Female	20	3.85	Dhaka
105	Abdullah al Mam	Male	22	3.50	Gopalgonj
106	Farzana Afroz	Female	23	3.65	Jashore
107	Al Imran Munna	Male	25	3.30	Khulna
108	Yeasin Arafat	Male	20	3.70	Comilla

❑ DISTINCT Keyword:

The **SELECT DISTINCT** statement is used to return only distinct (different) values.Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

➢ **Syntax:**

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

INPUT FORMAT:

```
SQL->SELECT City FROM student_detailes;
```

```
1 SELECT City FROM student_detailes;
```

Fig-25

OUTPUT TABLE:

City
Rangpur
Feni
Jamalpur
Gopalgonj
Jashore
Rangpur
Comilla

OUTPUT DATABASES:

Fig-26

But,If we use the **DISTINCT** Keyword.Rangpur show one time.

INPUT FORMAT:

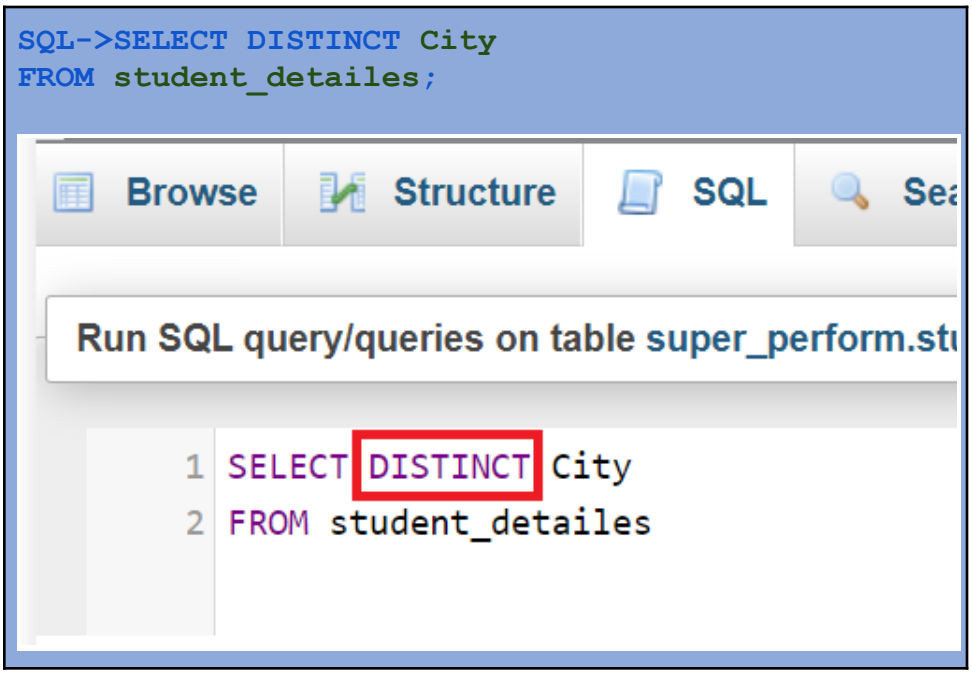


Fig-27

OUTPUT DATABASE:

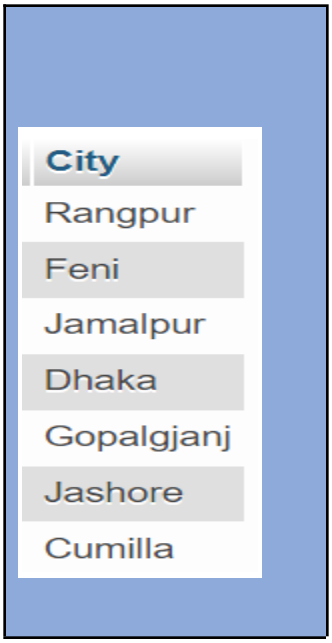


Fig-28

OUTPUT TABLE:

City
Rangpur
Feni
Jamalpur
Dhaka
Gopalgonj
Jashore
Comilla

☐ LIMIT:

The SQL LIMIT clause constrains the number of rows returned by a SELECT statement.

➢ Syntax:

```
SELECT  column1, column2, ...
FROM table_name
LIMIT conditions;
```

In fig-24, we see the total number of rows is 8.but now we want to see the 1st 5 rows only then we can use the LIMIT keyword.

INPUT FORMAT:

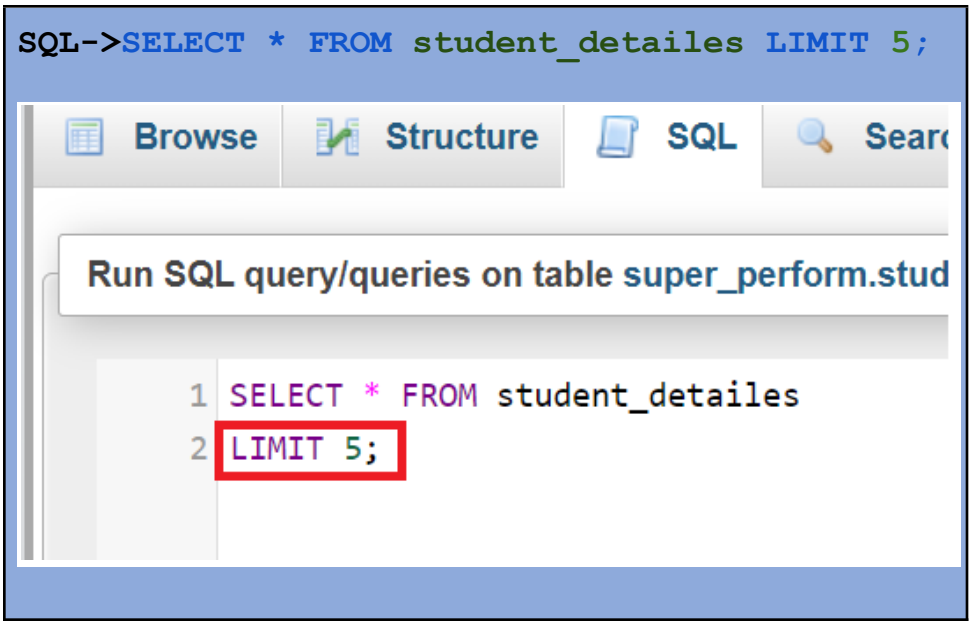


Fig-29

OUTPUT DATABASE:

Id	Name	Gender	Age	CGPA	City
101	Humayoun Kobir	Male	21	3.80	Rangpur
102	Naimur Rashid	Male	21	3.70	Feni
103	Abul Hasnat	Male	20	3.73	Jamalpur
104	Sanjana Akther	Female	20	3.85	Dhaka
105	Abdullah al Mam	Male	22	3.50	Gopalgjanj

Fig-30

OUTPUT TABLE:

Id	Name	Gender	Age	CGPA	City
101	Humayoun Kbir	Male	21	3.80	Rangpur
102	Naimur Rashid	Male	21	3.70	Feni
103	Abul Hasnat	Male	20	3.73	Jamalpur
104	Sanjana Akrher	Female	20	3.85	Dhaka
105	Abdullah Al Mam	Male	22	3.50	Gopalganj

Now ,we want to see the last 5 rows.

INPUT FORMAT:

```
SQL->SELECT * FROM student_detailes
LIMIT 3,5;
```

```
1 SELECT * FROM student_detailes
2 LIMIT 3, 5;
```

LAST 5 ROW SHOWN

1st 3 ROW NOT SHOW

Fig-31

OUTPUT DATABASE:

Id	Name	Gender	Age	CGPA	City
104	Sanjana Akther	Female	20	3.85	Dhaka
105	Abdullah al Mam	Male	22	3.50	Gopalgjanj
106	Farzana Afroz	Female	23	3.65	Jashore
107	Al Imran Munna	Male	25	3.30	Rangpur
108	Yeasin Arafat	Male	20	3.70	Cumilla

Fig-32

OUTPUT TABLE:

Id	Name	Gender	Age	CGPA	City
104	Sanjana Akther	Female	20	3.85	Dhaka
105	Abbdullah Al Mam	Male	22	3.50	Gopalgonj
106	Farzana Afroj	Female	23	3.65	Jashore
107	Al Imran Munna	Make	25	3.30	Rangpur
108	Yeasin Arafat	Male	20	3.70	Comilla

❑ORDER BY Clause

The **ORDER BY** keyword is used to sort the result-set in ascending or descending order.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;**

INPUT FORMAT:

```
SQL->SELECT Name
FROM student_detailes
ORDER BY Name;
```

```
1 SELECT Name
2 FROM student_detailes
3 ORDER BY Name;
```

Fig-33

OUTPUT DATABASES:

Name 1
Abdullah al Mam
Abul Hasnat
Al Imran Munna
Farzana Afroz
Humayoun Kobir
Naimur Rashid
Sanjana Akther
Yeasin Arafat

Fig-34

OUTPUT TABLE:

Name
Abdullah al Mamun
Abul Hasnat
Al Imran Munna
Farzana Afroz
Humayoun Kobir
Naimur Rashid

Sanjana Akther
Yeasin Arafat

INPUT FORMAT:

```
SQL->SELECT Id,Age,CGPA,Name
FROM student_detailes
ORDER BY Name DESC;
```

Run SQL query/queries on table super_perform.student_det

```
1 SELECT Id,Age,CGPA,Name
2 FROM student_detailes
3 ORDER BY Name DESC;
```

KEYWORD

CONDITION

FIG-35

Output database:

Id	Age	CGPA	Name 1
108	20	3.70	Yeasin Arafat
104	20	3.85	Sanjana Akther
102	21	3.70	Naimur Rashid
101	21	3.80	Humayoun Kobir
106	23	3.65	Farzana Afroz
107	25	3.30	Al Imran Munna
103	20	3.73	Abul Hasnat
105	22	3.50	Abdullah al Mam

Fig-36

OUTPUT TABLE:

Id	Age	CGPA	Name
108	20	3.70	Yeasin Arafat
104	20	3.85	Sanjana Akther
102	21	3.70	Naimur Rashid
101	21	3.80	Humayoun Kobir
106	23	3.65	Farzana Afroz
107	25	3.30	Al Imran Munna
103	20	3.73	Abul Hasnat
105	22	3.50	Abdullah al Mamun

Arithmetic Operators:

Arithmetic operators can perform arithmetic operations on numeric operands involved.Arithmetic operators are addition(+), subtraction(-), multiplication(*) and division(/). The + and - operators can also be used in date arithmetic.

Operator	Meaning	Operates on
+ (Add)	Addition	Numeric value
- (Subtract)	Subtraction	Numeric value
* (Multiply)	Multiplication	Numeric value
/ (Divide)	Division	Numeric value
% (Modulo)	Returns the integer remainder of a division. For example, 17 % 5 = 2 because the remainder of 17 divided by 5 is 2.	Numeric value

INPUT FORMAT:

```
SQL->SELECT
(11111111111111+878982949898983);
```

```
SELECT (11111111111111+878982949898983);
```

Output DATABASE:

```
+ Options
(11111111111111+878982949898983)
                          890094061010094
```

INPUT FORMAT:

```
SQL->SELECT (22222293-445325);
SELECT (22222293-445325);
```

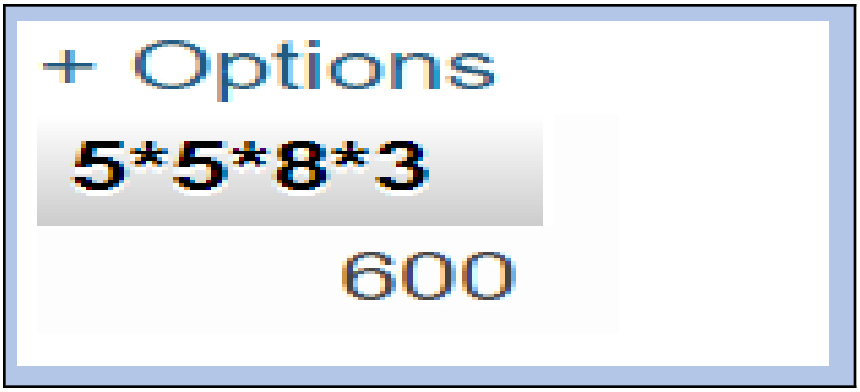
Output Database:

```
+ Options
(22222293-445325)
                          21776968
```

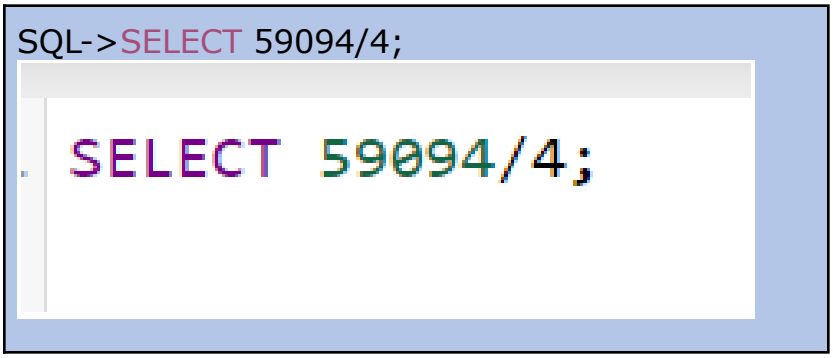
INPUT FORMAT:

```
SQL->SELECT 5*5*8*3;
SELECT 5*5*8*3;
```

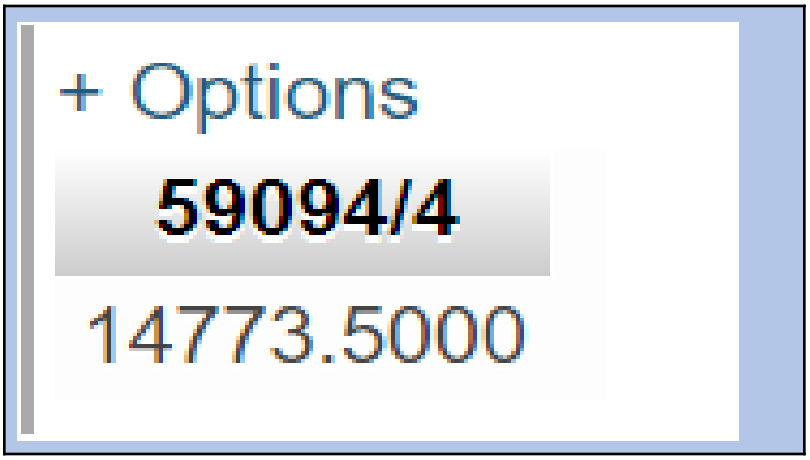
Output Database :



INPUT Format:



Output database:



Input format:

```
SQL-> SELECT 137%8;
```

```
1 SELECT 137%8;
```

Output database:

+ Options

137%8
1

❑WHERE Clause

The SQL WHERE clause is used to specify a condition while fetching the data from a single table or by joining with multiple tables. If the given condition is satisfied, then only it returns a specific value from the table. You should use the WHERE clause to filter the records and fetch only the necessary records.

➢ **Syntax:**

SELECT ***column1, column2, ...***
FROM ***table_name***
WHERE ***condition;***

INPUT FORMAT:

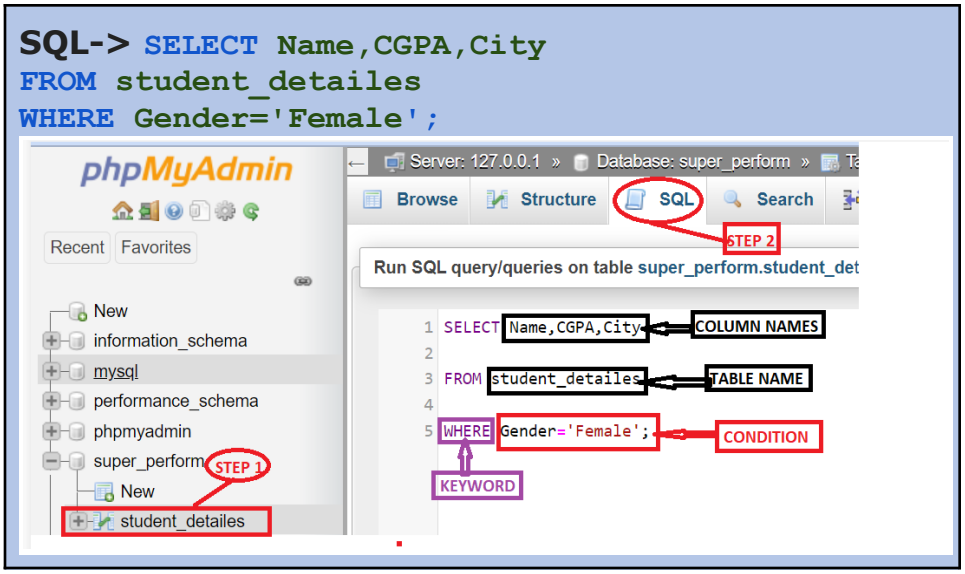
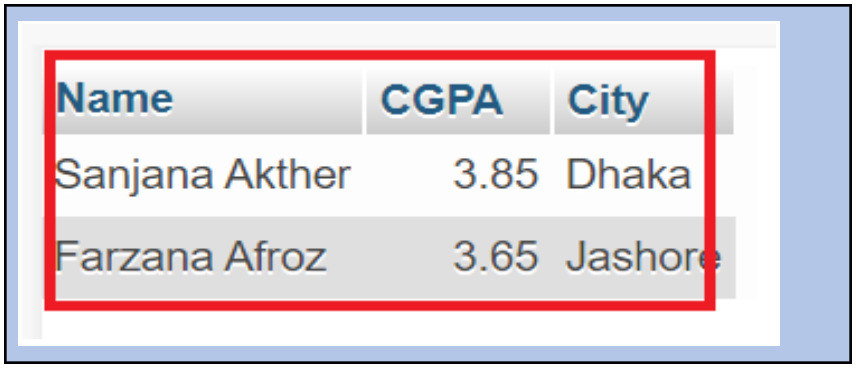


Fig-37

OUTPUT DATABASE:



Name	CGPA	City
Sanjana Akther	3.85	Dhaka
Farzana Afroz	3.65	Jashore

Fig-38

OUTPUT Table:

Name	CGPA	City
Sanjana Akther	3.85	Dhaka
Farzana Afroz	3.65	Jashore

❑DISTINCT CLAUSE:

The SQL **DISTINCT** keyword is used in conjunction with the SELECT statement to eliminate all the duplicate records and fetch only unique records.There may be a situation when you have multiple duplicate records in a table. While fetching such records, it makes more sense to fetch only unique records instead of fetching duplicate records.

INPUT FORMAT:

```
SQL-> SELECT DISTINCT City
FROM student_detailes
WHERE Gender='Male'
ORDER BY City DESC;
```

```
1 SELECT DISTINCT City
2 FROM student_detailes
3 WHERE Gender='Male'
4 ORDER BY City DESC;
```

Fig-39

OUTPUT DATABASE:

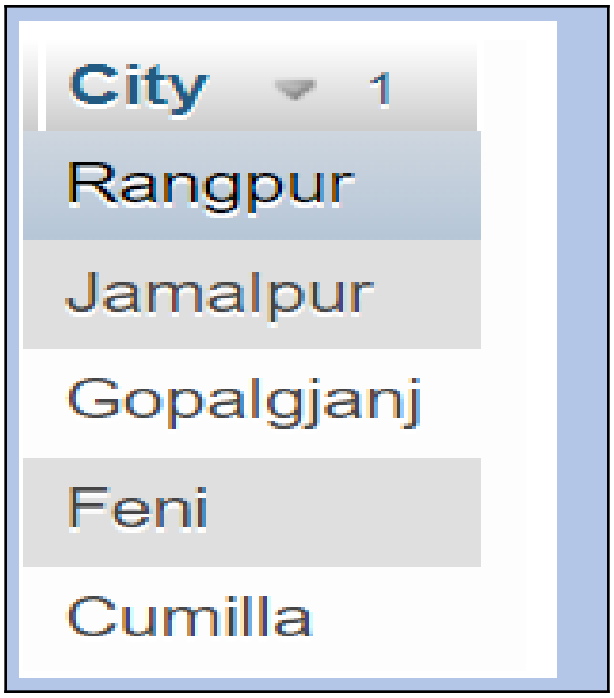


Fig-40

OUTPUT TABLE:

City
Rangpur
Jamalpur
Gopalgonj
Feni
Comilla

RELATIONAL Operator

Relational operators are important for making decisions. They allow us to compare numeric and char (chars are treated like numbers in C++) values to determine if one is greater than, less than, equal to, or not equal to another.Relational operators are binary meaning they require two operands.Relational operators have left to right associativity. Left to right associativity means that when two operators of the same precedence are adjacent, the leftmost operator is evaluated first.

Relational Operators	Meaning
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

❑ BETWEEN Operator

The **BETWEEN** operator selects values within a given range. The values can be numbers, text, or dates.The **BETWEEN** operator is inclusive: begin and end values are included.

➢ Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND
value2;
```

INPUT FORMAT:

```
SQL-> SELECT Id,Name,CGPA
FROM student_detailes
WHERE Id BETWEEN 102 AND 105;
```

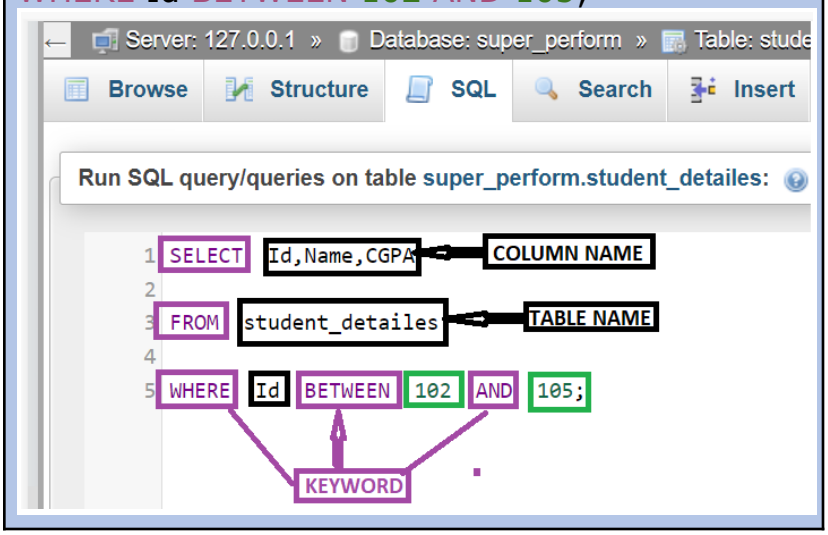
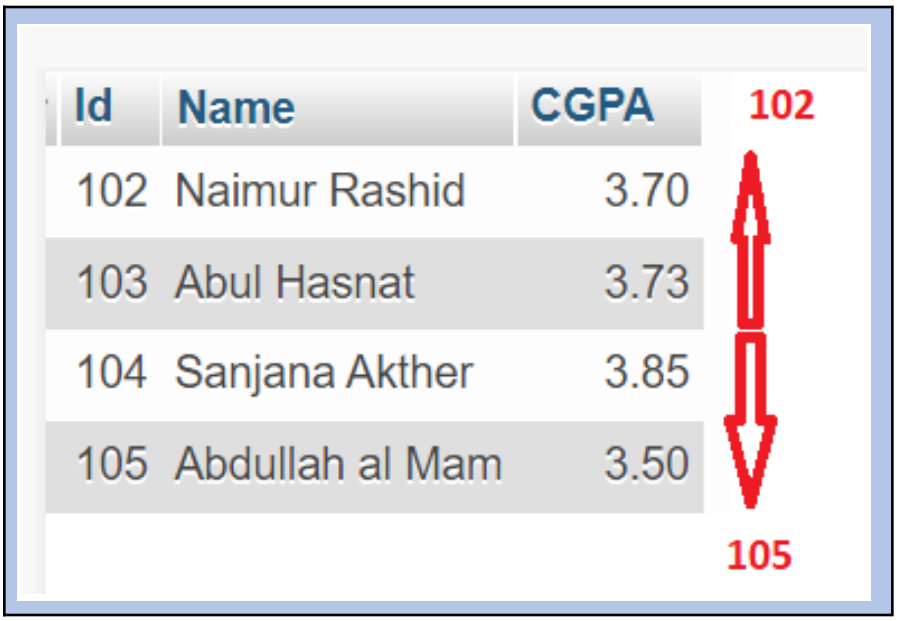


Fig-41

OUTPUT DATABASE:



Id	Name	CGPA
102	Naimur Rashid	3.70
103	Abul Hasnat	3.73
104	Sanjana Akther	3.85
105	Abdullah al Mam	3.50

Fig-42

OUTPUT TABLE:

Id	Name	CGPA
102	Naimur Rashid	3.70
103	Abul Hasnat	3.73
104	Sanjana Akther	3.85
105	Abdullah al Mamun	3.50

☐LOGICAL Operator

Logical operators test for the truth of some condition. Logical operators, like comparison operators, return a Boolean data type with a value of TRUE, FALSE, or UNKNOWN.

Operator	**Description**
ALL	TRUE if all of the subquery values meet the condition
AND	TRUE if all the conditions separated by AND is TRUE
ANY	TRUE if any of the subquery values meet the condition
BETWEEN	TRUE if the operand is within the range of comparisons
EXISTS	TRUE if the subquery returns one or more records
IN	TRUE if the operand is equal to one of a list of

	expressions
LIKE	TRUE if the operand matches a pattern
NOT	Displays a record if the condition(s) is NOT TRUE
OR	TRUE if any of the conditions separated by OR is TRUE
SOME	TRUE if any of the subquery values meet the condition

AND,OR,NOT Operator

The WHERE clause can be combined with AND, OR, and NOT operators.

The AND and OR operators are used to filter records based on more than one condition:

☐ OR OPERATOR:

The OR operator displays a record if any of the conditions separated by OR is TRUE. The **OR** command is used with WHERE to include rows where either condition is true.

➢ Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

INPUT FORMAT:

SQL-> SELECT *
FROM student_detailes
WHERE Gender='Male' OR CGPA>=3.70;

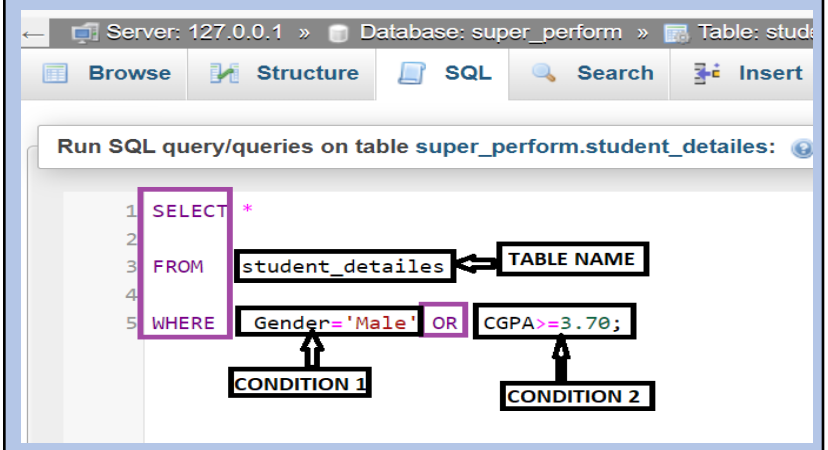


Fig-43

OUTPUT DATABASE:

Id	Name	Gender	Age	CGPA	City
101	Humayoun Kobir	Male	21	3.80	Rangpur
102	Naimur Rashid	Male	21	3.70	Feni
103	Abul Hasnat	Male	20	3.73	Jamalpur
104	Sanjana Akther	Female	20	3.85	Dhaka
105	Abdullah al Mam	Male	22	3.50	Gopalgjanj
107	Al Imran Munna	Male	25	3.30	Rangpur
108	Yeasin Arafat	Male	20	3.70	Cumilla

Fig-44

OUTPUT TABLE:

Id	Name	Gender	Age	CGPA	City
101	Humayoun Kobir	Male	21	3.80	Rangpur
102	Naimur Rashid	Male	21	3.70	Feni
103	Abul Hasnat	Male	20	3.73	Jamalpur
104	Sanjana Akther	Female	20	3.85	Dhaka
105	Abdullah al Mam	Male	22	3.50	Gopalganj
107	Al Imran Munna	Male	25	3.3.	Rangpur
108	Yeasin Arafat	Male	20	3.70	Comilla

☐ AND OPERATOR:

The AND operator displays a record if all the conditions separated by AND are TRUE.The AND command is used with WHERE to only include rows where both conditions are **true.**

➢ Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3
...;
```

Input Format :

```
SQL-> SELECT *
FROM student_detailes
WHERE Gender='Male' AND CGPA>=3.70;
```

```
1 SELECT *
2 FROM student_detailes
3 WHERE Gender='Male' AND CGPA>=3.70;
```

CONDITION 1 CONDITION 2

Fig-45

Output DATABASE:

Id	Name	Gender	Age	CGPA	City
101	Humayoun Kobir	Male	21	3.80	Rangpur
102	Naimur Rashid	Male	21	3.70	Feni
103	Abul Hasnat	Male	20	3.73	Jamalpur
108	Yeasin Arafat	Male	20	3.70	Cumilla

Fig-46

OUTPUT TABLE:

Id	Name	Gender	Age	CGPA	City
101	Humayoun Kobir	Male	21	3.80	Rangpur
102	Naimur Rashid	Male	21	3.70	Feni
103	Abul Hasnat	Male	20	3.73	Jamalpur
108	Yeasin Arafat	Male	20	3.70	Comilla

☐ NOT OPERATOR:

The NOT operator displays a record if the condition(s) is NOT TRUE.The **NOT** command is used with WHERE to only include rows where a condition is not true.

➢ Syntax:

SELECT ***column1, column2, ...***
FROM ***table_name***
WHERE NOT ***condition;***

Input Format :

```
SQL-> SELECT *
FROM student_detailes
WHERE NOT CGPA>=3.70;
```

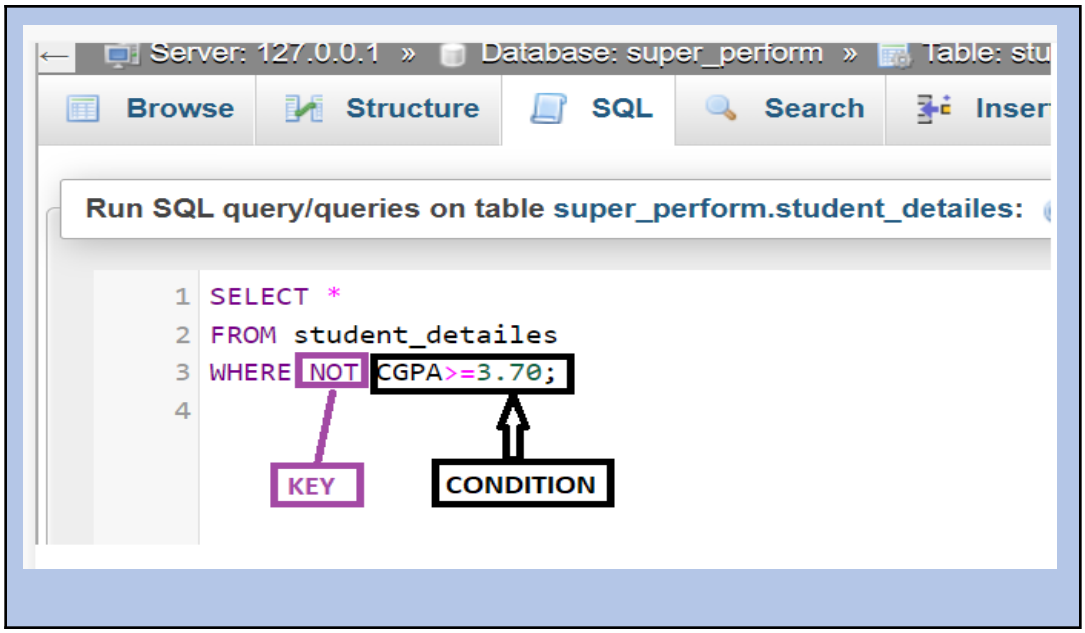
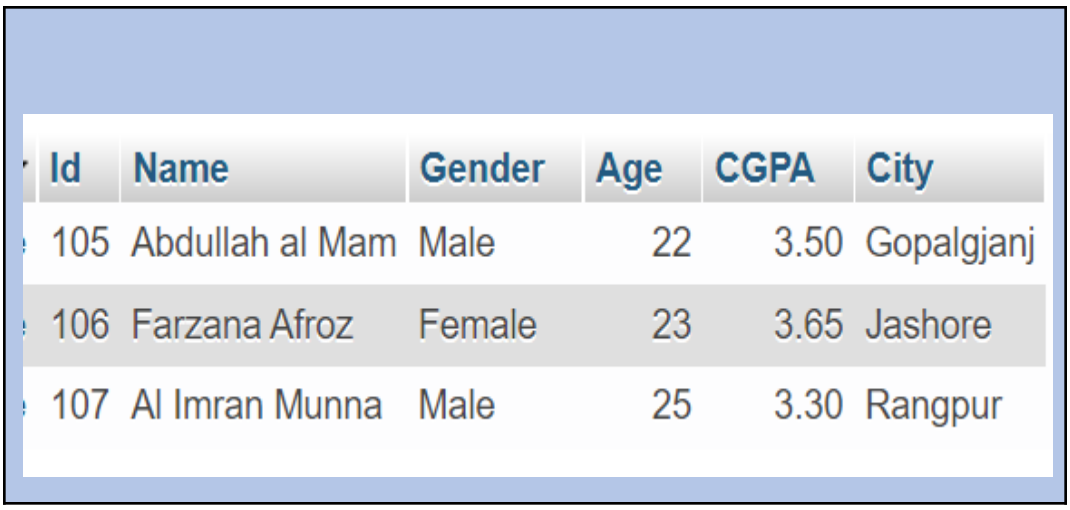


FIG-47

Output:



Id	Name	Gender	Age	CGPA	City
105	Abdullah al Mam	Male	22	3.50	Gopalgjanj
106	Farzana Afroz	Female	23	3.65	Jashore
107	Al Imran Munna	Male	25	3.30	Rangpur

Fig-48

OUTPUT TABLE:

Id	Name	Gender	Age	CGPA	City
105	Abdullah al Mamun	Male	22	3.50	Gopalganj
106	Farzana Afroz	Female	23	3.65	Jashore
107	Al Imran Munna	Male	25	3.30	Rangpur

☐ IN OPERATOR:

The IN operator allows you to specify multiple values in a WHERE clause.The IN operator is a shorthand for multiple OR conditions.

➢ Syntax:

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name* **IN (***value1*, *value2*, ...**);**

INPUT FORMAT:

```
SQL-> SELECT *
FROM student_detailes
WHERE City='Rangpur'
```

Browse | Structure | SQL | Search

Run SQL query/queries on table super_perform.student_det

```
1 SELECT *
2 FROM student_detailes
3 WHERE City='Rangpur'
4     OR
5      City='Dhaka'
6     OR
7      City='jamalpur';
```

Fig-49

OUTPUT DATABASE:

Id	Name	Gender	Age	CGPA	City
101	Humayoun Kobir	Male	21	3.80	Rangpur
103	Abul Hasnat	Male	20	3.73	Jamalpur
104	Sanjana Akther	Female	20	3.85	Dhaka
107	Al Imran Munna	Male	25	3.30	Rangpur

Fig-50

OUTPUT TABLE:

Id	Name	Gender	Age	CGPA	City
101	Humayoun kobir	Male	21	3.80	Rangpur
103	Abul HAsnat	Male	20	3.73	Jamalpur
104	Sanjana Akther	Female	20	3.85	Dhaka
107	Al Imran Munna	Male	25	3.30	Rangpur

But,If we use **IN Operator** it gives the same result.

INPUT FORMAT:

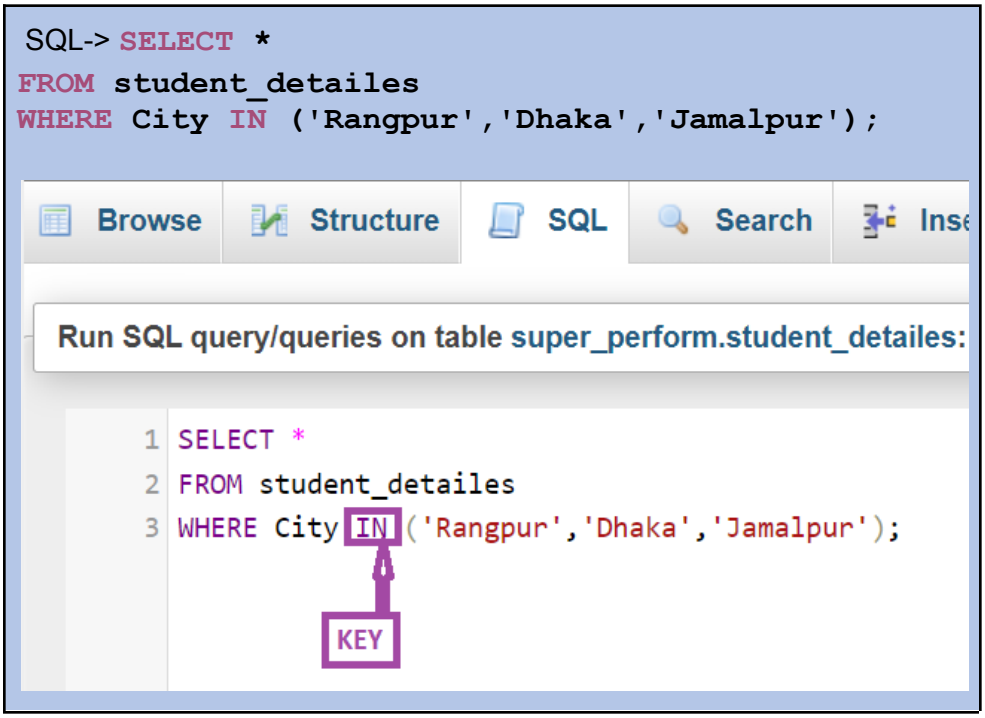


Fig-51

OUTPUT DATABASE:

Id	Name	Gender	Age	CGPA	City
101	Humayoun Kobir	Male	21	3.80	Rangpur
103	Abul Hasnat	Male	20	3.73	Jamalpur
104	Sanjana Akther	Female	20	3.85	Dhaka
107	Al Imran Munna	Male	25	3.30	Rangpur

Fig-52

OUTPUT TABLE:

Id	Name	Gender	Age	CGPA	City
101	Humayoun Kobir	Male	21	3.80	Rangpur
103	Abul Hasnat	Male	20	3.73	Jamalpur
104	Sanjana Akther	Female	20	3.85	Dhaka
107	Al Imran Munna	Male	25	3.30	Rangpur

☐ LIKE

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

There are two wildcards often used in conjunction with the LIKE operator:

- The percent sign (%) represents zero, one, or multiple characters
- The underscore sign (_) represents one, single character

➢ Syntax:

SELECT ***column1, column2, ...***
FROM ***table_name***
WHERE ***columnN*** **LIKE** ***pattern*****;**

INPUT FORMAT:

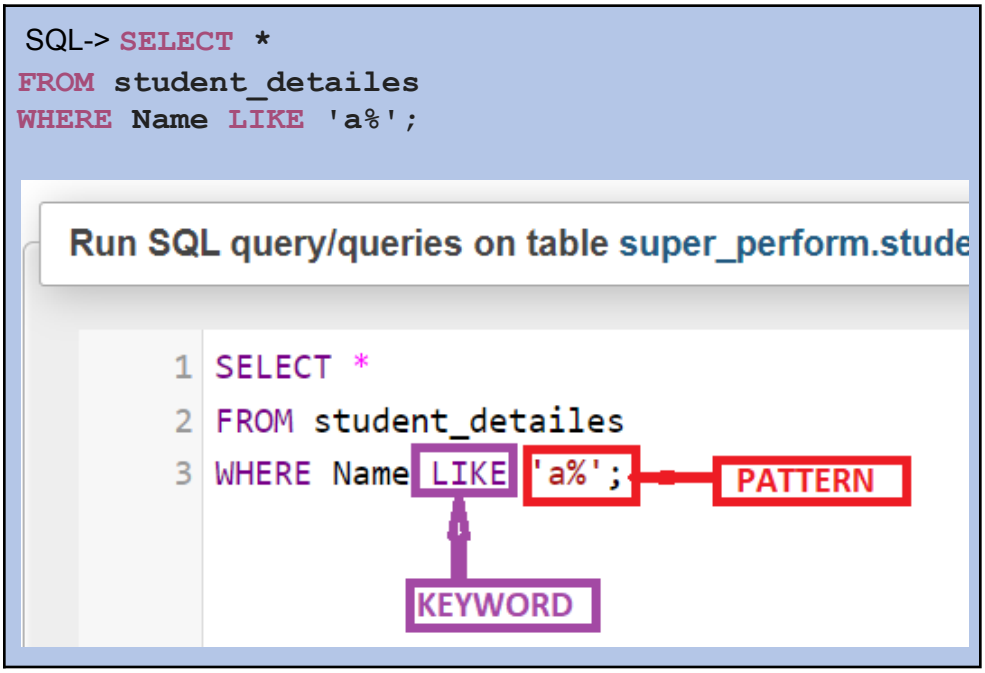


Fig-53

OUTPUT DATABASE:

	Id	Name	Gender	Age	CGPA	City
te	103	Abul Hasnat	Male	20	3.73	Jamalpur
te	105	Abdullah al Mam	Male	22	3.50	Gopalgjanj
te	107	Al Imran Munna	Male	25	3.30	Rangpur

Fig-54

OUTPUT TABLE:

Id	Name	Gender	Age	CGPA	City
103	Abul Hasnat	Male	20	3.73	Jamalpur
105	Abdullah al Mam	Male	22	3.50	Gopalganj
107	Al Imran Munna	Male	25	3.30	Rangpur

AS Keyword

SQL aliases are used to give a table, or a column in a table, a temporary name.Aliases are often used to make column names more readable.An alias only exists for the duration of that query.An alias is created with the AS keyword.

➢ **Syntax:**

SELECT *column_name* **AS** *alias_name*
FROM *table_name;*

INPUT FORMAT:

```
SQL-> SELECT Id AS ROLL,Name AS 'First Name'
FROM student_detailes;
```

```
1 SELECT Id AS ROLL,Name AS 'First Name'
2 FROM student_detailes;
```

Here we use 2 AS
1.Id AS ROLL
2.Name AS 'First Name'

Fig-55

OUTPUT DATABASE:

ROLL	First Name
101	Humayoun Kobir
102	Naimur Rashid
103	Abul Hasnat
104	Sanjana Akther
105	Abdullah al Mam
106	Farzana Afroz
107	Al Imran Munna
108	Yeasin Arafat

ROLL=Id
First Name= Name

Fig-56

OUTPUT TABLE:

Roll	First Name
101	Humayoun Kobir
102	Naimur Rashid
103	Abul Hasnat
104	Sanjana Akther
105	Abdullah al Mamun
106	Farzana Afroz
107	Al Imran Munna
108	Yeasin Arafat

UPPER() ,LOWER() AND CONCAT Function

☐ UPPER:

The **UPPER() function** converts a string to upper-case.It's the opposite of lower case function.

➢ **Syntax:**

```
SELECT UPPER('Anything')
FROM tablename;
```

INPUT FORMAT:

```
SQL-> SELECT UPPER(Name)
FROM student_detailes;
```

Run SQL query/queries on table super_perform.stu

```
1 SELECT UPPER(Name)
2 FROM student_detailes;
```

All Name convert Upper case

Fig-57

OUTPUT DATABASE:

UPPER(Name)
HUMAYOUN KOBIR
NAIMUR RASHID
ABUL HASNAT
SANJANA AKTHER
ABDULLAH AL MAM
FARZANA AFROZ
AL IMRAN MUNNA
YEASIN ARAFAT

Upper case Name

Fig-58

- We can see in the output, all names are converted to upper-case.

OUTPUT TABLE:

Upper(Name)
HUMAYOUN KOBIR

NAIMUR RASHID
ABUL HASNAT
SANJANA AKTHER
ABDULLAH AL MAMUN
FARZANA AFROZ
AL IMRAN MUNNA
YEASIN ARAFAT

☐ LOWER function:

It converts the text to upper-case. Totally opposite of upper case function.

➢ Syntax:

```
SELECT LOWER('Anything')
FROM tablename;
```

INPUT FORMAT:

```
SQL-> SELECT LOWER(City) AS 'LOWERCASE CITY NAME'
FROM student_detailes;
```

```
1 SELECT LOWER(City) AS 'LOWERCASE CITY NAME'
2
3 FROM student_detailes;
```

FIELD NAME WILL CHANGE FOR AS KEYWORD

ALL CITY NAME CONVERT LOWER CASE

Fig-59

OUTPUT DATABASE:

LOWERCASE CITY NAME
rangpur
feni
jamalpur
dhaka
gopalgjan
jashore
rangpur
cumilla

AS Keyword Use

ALL city name convert lower case

Fig-60

OUTPUT TABLE:

LOWERCASE CITY NAME
rangpur
feni
jamalpur
dhaka
gopalganj
jashore
rangpur
comilla

☐ CONCAT Function:

MySQL CONCAT() function is used to add two or more strings.There may be one or more arguments.Returns the string that results from concatenating the arguments.Returns a **nonbinary string,** if all arguments are nonbinary strings.Returns a binary string, if the arguments include any binary strings.If the argument is numeric, it is converted to its equivalent nonbinary string form.Returns NULL if any argument is NULL.

➢ Syntax:

```
SELECT CONCAT(string1….string2..string3)
FROM tablename;
```

INPUT FORMAT:

```
SQL-> SELECT CONCAT(Name,' is ',Age,' year old. ')
FROM student_detailes;
```

FUNCTION

```
1 SELECT CONCAT(Name,' is ',Age,' year old. ')
2 FROM student_detailes;
```

COLUMN/FIELD/STRING

Fig-61

OUTPUT DATABASE:

+ Options

CONCAT(Name,' is ',Age,' year old. ')
Humayoun Kobir is 21 year old.
Naimur Rashid is 21 year old.
Abul Hasnat is 20 year old.
Sanjana Akther is 20 year old.
Abdullah al Mam is 22 year old.
Farzana Afroz is 23 year old.
Al Imran Munna is 25 year old.
Yeasin Arafat is 20 year old.

It give a full sentance

Fig-62

OUTPUT TABLE:

CONCAT(Name, 'is',Age, 'year old.')
Humayoun kobir is 21 year old
Naimur Rashid is 21 year old
Abul Hasnat is 20 year old
Sanjana Akther is 20 year old
Abdullah al Mamun is 21 year old
Farzana Afroz is 22 year old
Al Imran Munna is 25 year old
Yeasin Arafat is 20 year old

☐Aggregate functions:

In database management, an **aggregate function** is a function where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning.

Various Aggregate Functions:

- ➔ Count()
- ➔ Sum()
- ➔ Avg()
- ➔ Min()
- ➔ Max()

☐ COUNT() Function:

The COUNT() function returns the number of rows that matches a specified criterion/returns the total number of values in a given column.

➢ Syntax:

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

INPUT FORMAT:

Id	Name	Gender	Age	CGPA	City
101	Humayoun Kobir	Male	21	3.80	Rangpur
102	Naimur Rashid	Male	21	3.70	Feni
103	Abul Hasnat	Male	20	3.73	Jamalpur
104	Sanjana Akther	Female	20	3.85	Dhaka
105	Abdullah al Mam	Male	22	3.50	Gopalgjanj
106	Farzana Afroz	Female	23	3.65	Jashore
107	Al Imran Munna	Male	25	3.30	Rangpur
108	Yeasin Arafat	Male	20	3.70	Cumilla
109	Rokon	Male	NULL	3.00	Rangpur

Age is NULL

Fig-63

OUTPUT TABLE:

Id	Name	Gender	Age	CGPA	City
101	Humayoun Kobir	Male	21	3.80	Rangpur
102	Naimur Rashid	Male	21	3.70	Feni
103	Abul Hasnat	Male	20	3.73	Jamalpur

Id	Name	Gender	Age	CGPA	City
104	Sanjana Akther	Female	20	3.85	Dhaka
105	Abdullah al Mam	Male	22	3.50	Gopalganj
106	Farzana Afroz	Female	23	3.65	Jashore
107	Al Imran Munna	Mlae	25	3.30	Rangpur
108	Yeasin Arafat	Male	20	3.70	Comilla
109	Rokon	Male	Null	3.00	Rangpur

Fig-63 ,we see the last row whose name is Rokon ,age is null .If we count the age column.Count shows 8 rows.

INPUT FORMAT:

```
SQL->SELECT COUNT(Age)
FROM student_detailes;
```

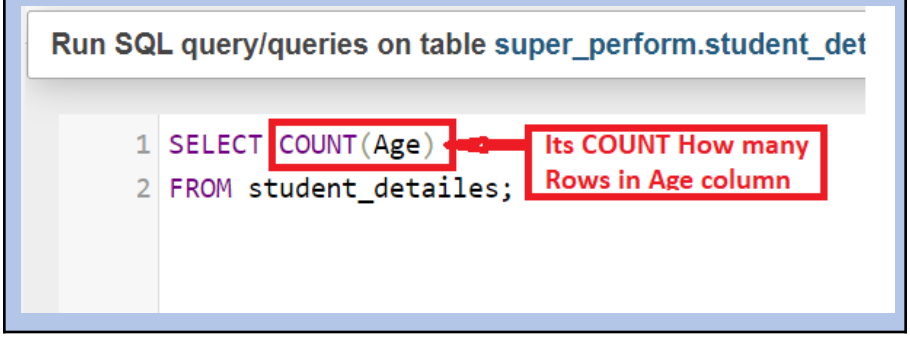


Fig-64

Output DATABASE:

COUNT(Age)
8

Fig-65

We can see the 8 rows in the Age column.because last row age is Null.But if we use COUNT(*) the output will geven 9 rows.

☐ COUNT(*) function:

It returns the number of rows in a table.If there is any null value for the **"Count"** targeted column,It will count that column also. So, for the exact column count ,we'll use **count(*)** function.

➢ Syntax:

```
SELECT COUNT(*)
FROM table_name
WHERE condition;
```

INPUT FORMAT:

```
SQL-> SELECT COUNT(*)
FROM student_detailes;
```

Run SQL query/queries on table super_perform.stu

```
1 SELECT COUNT(*)
2 FROM student_detailes;
3
```

Its count How many row in the table

Fig-66

Output TABLE:

COUNT(*)
9

Fig-67

☐ SUM() function:

The SQL **SUM function** is used to return the sum of an expression in a SELECT statement.It actually does add the value of the column of targeted query.

➢ Syntax:

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

INPUT FORMAT:

```
SQL->SELECT SUM(CGPA)
FROM student_detailes
WHERE age>=21 AND CGPA>3.50;
```

```
1 SELECT SUM(CGPA)
2
3 FROM student_detailes
4
5 WHERE age>=21 AND CGPA>3.50;
```

Its SUM CGPA Whose Age grather than 21 and cgpa is grater than 3.50

Fig-68

Output DATABASE:

SUM(CGPA)
11.15

☐ AVG() Function:

The **AVG() function** returns the average value of an expression.It is noted that **NULL** values are ignored.

➢ Syntax:

SELECT AVG(*column_name*)

```
FROM table_name
WHERE condition;
```

INPUT FORMAT:

```
SQL->SELECT AVG(AGE) AS 'STUDENT AGE AVERAGE'
FROM student_detailes
WHERE CGPA>3.70;
```

```
1 SELECT AVG(Age) AS 'STUDENT AGE AVERAGE'
2
3 FROM student_detailes
4
5 WHERE CGPA>3.70;
```

Function

Condition

OUTPUT DATABASE:

STUDENT AGE AVERAGE
20.3333

AS Use

Average Age

☐ MAX() FUNCTION:

The **MAX() function** returns the maximum value in a set of values.It is opposite of MIN() Function.

➢ **Syntax:**

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

INPUT FORMAT:

```
SQL-> SELECT MAX(CGPA) FROM student_detailes;
```

```
1 SELECT MAX(CGPA)
2 FROM student_detailes;
```

Function

Output DATABASE:

MAX(CGPA)
3.85

☐ MIN() FUNCTION:

The **MIN() function** returns the minimum value in a set of values.It displays the lowest value of a column.

➢ Syntax:

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

INPUT FORMAT:

```
SQL->SELECT MIN(CGPA) FROM student_detailes;
```

```
1 SELECT MIN(CGPA)
2 FROM student_detailes;
```

OUTPUT DATABASE:

MIN(CGPA)
3.00

☐ GROUP BY Clause:

The **SQL GROUP** BY clause is used in collaboration with the SELECT statement to arrange identical data into groups.The **GROUP** BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.

➢ Syntax:

SELECT ***Column,group_Function(column_name)***

FROM ***table_name***

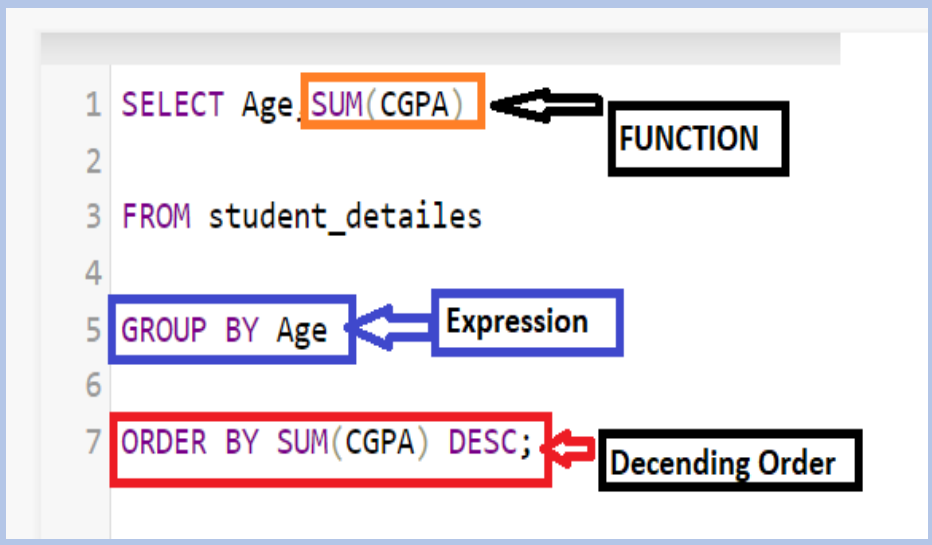
WHERE ***condition***

[GROUP BY ***expression*****]**

[ORDER BY ***Column*****];**

INPUT FORMAT:

```
SQL-> SELECT Age,SUM(CGPA)
FROM student_detailes
GROUP BY Age
ORDER BY SUM(CGPA) DESC;
```



OUTPUT DATABASE:

Age	SUM(CGPA) 1
20	11.28
21	7.50
23	3.65
22	3.50
25	3.30
NULL	3.00

OUTPUT TABLE:

Age	SUM(CGPA)
20	11.25
21	7.50
23	3.65
22	3.50
25	3.30
Null	3.00

☐ DELETE STATEMENT:

The **DELETE** statement is used to delete existing records in a table.It removes the

mentioned record or row.It is also a matter of concern that **delete operation** is a valuable operation.Because once it is **deleted**, we can't get back that valuable data.

➢ **Syntax:**

DELETE FROM ***table_name*** **WHERE** ***condition*****;**

INPUT FORMAT:

```
SQL->DELETE FROM student_detailes
WHERE Id=108;
```

Output TABLE:

Id	Name	Gender	Age	CGPA	City
101	Humayoun Kobir	Male	21	3.80	Rangpur
102	Naimur Rashid	Male	21	3.70	Feni
103	Abul Hasnat	Male	20	3.73	Jamalpur
104	Sanjana Akrher	Female	20	3.85	Dhaka
105	Abdullah Al Mam	Male	22	3.50	Gopalganj
106	Farzana Afroz	Female	23	3.65	Jashore
107	All Imran Munna	Male	25	3.30	Khulna
109	Rokon	Male	NULL	3.00	Rangpur

❑ ALTER STATEMENT

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.
The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

➢ **Syntax:**

```
ALTER TABLE table_name
ADD column_name datatype;
```

INPUT FORMAT:

```
SQL-> ALTER TABLE student_detailes
ADD Department varchar(15);
```

Output TABLE:

Id	Name	Gender	Age	CGPA	City	Department
101	Humayoun Kobir	Male	21	3.80	Rangpur	NULL
102	Naimur Rashid	Male	21	3.70	Feni	NULL
103	Abul Hasnat	Male	20	3.73	Jamalpur	NULL
104	Sanjana Akrher	Female	20	3.85	Dhaka	NULL
105	Abdullah Al Mam	Male	22	3.50	Gopalganj	NULL
106	Farzana Afroz	Female	23	3.65	Jashore	NULL
107	All Imran Munna	Male	25	3.30	Khulna	NULL
109	Rokon	Male	NULL	3.00	Rangpur	NULL

☐ ALTER TABLE - DROP COLUMN:

➢ **Syntax:**

```
ALTER TABLE table_name
DROP column_name;
```

INPUT FORMAT:

```
SQL-> ALTER TABLE student_detailes
DROP City,CGPA;
```

Output TABLE:

Id	Name	Gender	Age	CGPA	Department
101	Humayoun Kobir	Male	21	3.80	NULL
102	Naimur Rashid	Male	21	3.70	NULL
103	Abul Hasnat	Male	20	3.73	NULL
104	Sanjana Akrher	Female	20	3.85	NULL
105	Abdullah Al Mam	Male	22	3.50	NULL
106	Farzana Afroz	Female	23	3.65	NULL
107	All Imran Munna	Male	25	3.30	NULL
109	Rokon	Male	NULL	3.00	NULL

☐ UPDATE STATEMENT

The UPDATE statement is used to modify the existing records in a table.If any value needs to be **edited** or we want to update a value or change any entry ,it could be updated by this query.

➢ Syntax:

UPDATE ***table_name***
SET ***column1 = value1, column2 = value2, ...***

WHERE ***condition*****;**

INPUT FORMAT:

```
SQL->UPDATE student_detailes
SET Age=19
WHERE Id=109;
```

Output TABLE:

Id	Name	Gender	Age	CGPA	Department
101	Humayoun Kobir	Male	21	3.80	NULL
102	Naimur Rashid	Male	21	3.70	NULL
103	Abul Hasnat	Male	20	3.73	NULL
104	Sanjana Akrher	Female	20	3.85	NULL
105	Abdullah Al Mam	Male	22	3.50	NULL
106	Farzana Afroz	Female	23	3.65	NULL
107	All Imran Munna	Male	25	3.30	NULL
109	Rokon	Male	19	3.00	NULL

INPUT FORMAT:

```
SQL->
UPDATE student_detailes SET Department=CSE WHERE
Id=101;
UPDATE student_detailes SET Department=BBA WHERE
Id=102;
UPDATE student_detailes SET Department=EEE WHERE
Id=103;
UPDATE student_detailes SET Department=CSE WHERE
Id=104;
UPDATE student_detailes SET Department=BBA WHERE
```

```
Id=105;
UPDATE student_detailes SET Department=EEE WHERE
Id=106;
UPDATE student_detailes SET Department=BBA WHERE
Id=107;
UPDATE student_detailes SET Department=CSE WHERE
Id=109;
```

OUTPUT TABLE:

Id	Name	Gender	Age	CGPA	Department
101	Humayoun Kobir	Male	21	3.80	CSE
102	Naimur Rashid	Male	21	3.70	BBA
103	Abul Hasnat	Male	20	3.73	EEE
104	Sanjana Akrher	Female	20	3.85	CSE
105	Abdullah Al Mam	Male	22	3.50	BBA
106	Farzana Afroz	Female	23	3.65	EEE
107	All Imran Munna	Male	25	3.30	BBA
109	Rokon	Male	19	3.00	CSE

☐ INDEX Statement:

CREATE INDEX Statement:

The `CREATE INDEX` statement is used to create indexes in tables.Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches/queries.

Note: Updating a table with indexes takes more time than updating a table without (because the indexes also need an update). So, only create indexes on columns that will be frequently searched against.

➢ **Syntax:**

CREATE INDEX ***index_name***
ON ***table_name*** **(*column1*, *column2*, ...);**

INPUT FORMAT:

```
SQL-> CREATE INDEX STD_Indexing
ON student_detailes(Department,Name,CGPA);
```

SHOW INDEX Statement:

➢ **Syntax:**

SHOW INDEX FROM ***table_name*** **;**

```
SQL-> CREATE INDEX FROM student_detailes;
```

DROP INDEX Statement

The `DROP INDEX` statement is used to delete an index in a table.

➢ **Syntax:**

SHOW INDEX FROM ***table_name*** **;**

FOREIGN Key:

A **foreign key** is a key used to link two tables together. This is sometimes called a referencing key.Foreign Key is a column or a combination of columns whose values match a Primary Key in a different table.The relationship between 2 tables matches the Primary Key in one of the tables with a **Foreign Key** in the second table.
If a table has a primary key defined on any field(s), then you can not have two records having the same value of that field(s).

☐ AUTO INCREMENT Keyword:

Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.Often this is the primary key field that we would like to be created automatically every time a new record is inserted.

➢ Syntax:

```
CREATE TABLE table_name (
        column1 datatype(size) AUTO _INCREMENT,
        column2 datatype(size),
        column3 datatype(size),
         PRIMARY KEY(COLUMNname)
         FOREIGN KEY(COLUMNname)
         REFERENCES TABLE 1(Columnname)

);
```

INPUT FORMAT:

```
SQL-> CREATE TABLE Exam_date
(
   Reg_Number INT NOT NULL AUTO_INCREMENT,
   STD_Id INT, Exam_date DATETIME,
   PRIMARY KEY(Reg_Number),
   FOREIGN KEY(STD_Id)
   REFERENCES student_detailes(Id)
);
```

OUTPUT TABLE:

Reg_Number	Std_Id	Exam_date

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
Reg_Number	INT(11)	NO	PRIMARY KEY	None	AUTO_INCREMENT
Std_Id	INT(11)	YES		NULL	
Exam_date	DATE	YES		NULL	

INPUT FORMAT(2):

INSERT Table 2

```
SQL-> INSERT INTO `exam_date` (`Reg_Number`, `STD_Id`,
`Exam_date`)
VALUES
('13176165', '101', '2022-06-21 9:45:00'),
( '109', '2022-06-20 13:30:00'),
 ( '104', '2022-06-25 16:00:00'),
 ( '107', '2022-06-27 15:15:00'),
 ( '102', '2022-06-20 11:00:00'),
 ( '103', '2022-06-23 14:00:00'),
 ( '105', '2022-06-29 13:30:00'),
 ( '106', '2022-06-27 08:0:00');
```

OUTPUT FORMAT:

Reg_Number	STD_Id	Exam_date
13176165	101	2022-06-21 09:45:00
13176166	109	2022-06-20 13:30:00
13176167	104	2022-06-25 16:00:00
13176168	107	2022-06-27 15:15:00
13176169	102	2022-06-20 11:00:00
13176170	103	2022-06-23 14:00:00
13176171	105	2022-06-29 13:30:00
13176172	106	2022-06-27 08:00:00

INPUT FORMAT:

```
SQL-> ALTER TABLE `exam_date` CHANGE `STD_Id` `Id`
INT(11);
```

OUTPUT TABLE:

Reg_Number	Id	Exam_date
13176165	101	2022-06-21 09:45:00
13176166	109	2022-06-20 13:30:00
13176167	104	2022-06-25 16:00:00
13176168	107	2022-06-27 15:15:00
13176169	102	2022-06-20 11:00:00

13176170	103	2022-06-23 14:00:00
13176171	105	2022-06-29 13:30:00
13176172	106	2022-06-27 08:00:00

OUTPUT TABLE:

TABLE 1:(STUDENT)

Id	Name	Gender	Age	CGPA	Department
101	Humayoun Kobir	Male	21	3.80	CSE
102	Naimur Rashid	Male	21	3.70	BBA
103	Abul Hasnat	Male	20	3.73	EEE
104	Sanjana Akrher	Female	20	3.85	CSE
105	Abdullah Al Mam	Male	22	3.50	BBA
106	Farzana Afroz	Female	23	3.65	EEE
107	All Imran Munna	Male	25	3.30	BBA
109	Rokon	Male	19	3.00	CSE

TABLE 2:(Exam_Routine)

❑JOINING TABLE:

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

Different Types of SQL JOINs

Here are the different types of the JOINs in SQL:

- (INNER) JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
- RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
- FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table

INPUT FORMAT:

```
SQL-> SELECT student_detailes.Id,Reg_Number,
Name,Gender,Department,CGPA,exam_date
FROM student_detailes,exam_date
WHERE student_detailes.Id = exam_date.Id;
```

Output DATABASE:

Id	Reg_Number	Name	Gender	Department	CGPA	exam_date
101	13176165	Humayoun Kobir	Male	CSE	3.80	2022-06-21 09:45:00
109	13176166	Rokon	Male	CSE	3.00	2022-06-20 13:30:00
104	13176167	Sanjana Akther	Female	CSE	3.85	2022-06-25 16:00:00
107	13176168	Al Imran Munna	Male	BBA	3.30	2022-06-27 15:15:00
102	13176169	Naimur Rashid	Male	BBA	3.70	2022-06-20 11:00:00
103	13176170	Abul Hasnat	Male	EEE	3.73	2022-06-23 14:00:00
105	13176171	Abdullah al Mam	Male	BBA	3.50	2022-06-29 13:30:00
106	13176172	Farzana Afroz	Female	EEE	3.65	2022-06-27 08:00:00

OUTPUT TABLE:

Id	Reg_Number	Name	Gender	Department	CGPA	Exam_date
101	13176165	Humayoun Kobir	Male	CSE	3.80	2022-06-21 09:45:00
109	13176166	Rokon	Male	CSE	3.00	2022-06-20 13:30:00

104	13176167	Sanjana Akther	Female	CSE	3.85	2022-06-25 16:00:00
107	13176168	Al Imran Munna	Male	BBA	3.30	2022-06-27 15:15:00
102	13176169	Naimur Rashid	Male	BBA	3.70	2022-06-20 11:00:00
103	13176170	Abul Hasnat	Male	EEE	3.73	2022-06-23 14:00:00
105	13176171	Abdullah All Mam	Male	BBA	3.50	2022-06-29 13:30:00
106	13176172	Farzana Afroz	Female	EEE	3.65	2022-06-27 08:00:00

☐ JOIN Clause:

JOIN is an SQL clause used to query and access data from multiple tables, based on logical relationships between those tables.In other words,**JOINS** indicate how SQL Server should use data from one table to select the rows from another table.

Different types of JOINS in SQL Server

- INNER JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- SELF JOIN
- CROSS JOIN

INPUT FORMAT:

```
SQL-> SELECT std.Id,exam.Reg_Number,
std.Department,exam.Exam_date
FROM student_detailes AS std JOIN exam_date AS
exam
ON std.Id = exam.Id;
```

OUTPUT DATABASE:

Id	Reg_Number	Department	Exam_date
101	13176165	CSE	2022-06-21 09:45:00
109	13176166	CSE	2022-06-20 13:30:00
104	13176167	CSE	2022-06-25 16:00:00
107	13176168	BBA	2022-06-27 15:15:00
102	13176169	BBA	2022-06-20 11:00:00
103	13176170	EEE	2022-06-23 14:00:00
105	13176171	BBA	2022-06-29 13:30:00
106	13176172	EEE	2022-06-27 08:00:00

OUTPUT TABLE:

Id	Reg_Number	Department	Exam_date
101	13176165	CSE	2022-06-21 09:45:00
109	13176166	CSE	2022-06-20 13:30:00
104	13176167	CSE	2022-06-25 16:00:00
107	13176168	BBA	2022-06-27 15:15:00
102	13176169	BBA	2022-06-20 11:00:00
103	13176170	EEE	2022-06-23 14:00:00

105	13176171	BBA	2022-06-29 13:30:00
106	13176172	EEE	2022-06-27 08:00:00

☐ INNER JOIN:

The **INNER JOIN** keyword selects records that have matching values in both tables. The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns. If there are records in the “Student_details" table that do not have matches in "Exam_date", no records will not be shown!

Table-1:

Id	Name	Gender	Age	CGPA	Department
101	Humayoun Kobir	Male	21	3.80	CSE
102	Naimur Rashid	Male	21	3.70	BBA
103	Abul Hasnat	Male	20	3.73	EEE
104	Sanjana Akrher	Female	20	3.85	CSE
105	Abdullah Al Mam	Male	22	3.50	BBA
106	Farzana Afroz	Female	23	3.65	EEE
107	All Imran Munna	Male	25	3.30	BBA
109	Rokon	Male	19	3.00	CSE

Table 2:

Reg_Number	Id	Exam_date

13176165	101	2022-06-21 09:45:00
13176166	109	2022-06-20 13:30:00
13176167	104	2022-06-25 16:00:00
13176168	107	2022-06-27 15:15:00

➢ **Syntax:**

SELECT *column_name(s)*
FROM *table1*
INNER JOIN *table2*
ON *table1.column_name = table2.column_name*;

INPUT FORMAT:

```
SQL-> SELECT std.Id,exam.Reg_Number,
std.Department,exam.Exam_date
FROM student_detailes AS std INNER JOIN exam_date AS
exam
ON std.Id = exam.Id;
```

OUTPUT DATABASE:

Id	Reg_Number	Department	Exam_date
101	13176165	CSE	2022-06-21 09:45:00
109	13176166	CSE	2022-06-20 13:30:00
104	13176167	CSE	2022-06-25 16:00:00
107	13176168	BBA	2022-06-27 15:15:00

OUTPUT TABLE:

Id	Reg_Number	Department	Exam_date
101	13176165	CSE	2022-06-21 09:45:00
109	13176166	CSE	2022-06-20 13:30:00
104	13176167	CSE	2022-06-25 16:00:00
107	13176168	BBA	2022-06-27 15:15:00

☐ LEFT JOIN:

The **LEFT JOIN** keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

➢ Syntax:

SELECT *column_name(s)*
FROM *table1*
LEFT JOIN *table2*
ON *table1.column_name = table2.column_name*;

INPUT FORMAT:

```
SQL->SELECT std.Id,exam.Reg_Number,
std.Department,exam.Exam_date
FROM student_detailes AS std LEFT JOIN
exam_date AS exam
ON std.Id = exam.Id;
```

OUTPUT DATABASE:

Id	Reg_Number	Department	Exam_date
101	13176165	CSE	2022-06-21 09:45:00
109	13176166	CSE	2022-06-20 13:30:00
104	13176167	CSE	2022-06-25 16:00:00
107	13176168	BBA	2022-06-27 15:15:00
102	NULL	BBA	NULL
105	NULL	BBA	NULL
103	NULL	EEE	NULL
106	NULL	EEE	NULL

OUTPUT TABLE:

Id	Reg_Number	Department	Exam_date
101	13176165	CSE	2022-06-21 09:45:00
109	13176166	CSE	2022-06-20 13:30:00
104	13176167	CSE	2022-06-25 16:00:00
107	13176168	BBA	2022-06-27 15:15:00
102	NULL	BBA	NULL
103	NULL	EEE	NULL
105	NULL	BBA	NULL
106	NULL	EEE	NULL

☐ RIGHT JOIN

The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match

➢ Syntax:

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

TABLE-02:

Reg_Number	Id	Exam_date
13176165	101	2022-06-21 09:45:00
13176166	109	2022-06-20 13:30:00
13176167	104	2022-06-25 16:00:00
13176168	NULL	2022-06-27 15:15:00

INPUT FORMAT:

```
SQL-> SELECT std.Id,exam.Reg_Number,
std.Department,exam.Exam_date
FROM student_detailes AS std RIGHT JOIN
exam_date AS exam
ON std.Id = exam.Id;
```

OUTPUT DATABASE:

Options

Id	Reg_Number	Department	Exam_date
101	13176165	CSE	2022-06-21 09:45:00
109	13176166	CSE	2022-06-20 13:30:00
104	13176167	CSE	2022-06-25 16:00:00
NULL	13176168	NULL	2022-06-27 15:15:00

OUTPUT TABLE:

Id	Reg_Number	Department	Exam_date
101	13176165	CSE	2022-06-21 09:45:00
109	13176166	CSE	2022-06-20 13:30:00
104	13176167	CSE	2022-06-25 16:00:00
NULL	13176168	NULL	2022-06-27 15:150:00

➢ **Syntax:**

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]

EXCEPT

SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

UNION Operator

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

- Every SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in every SELECT statement must also be in the same order

➢ **Syntax:**

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

INPUT FORMAT:

```
SQL-> CREATE TABLE STUDENT_2
(
   Id int NOT NULL,
   Name varchar(15),
   Gender varchar(10),
   Age int(5),
   CGPA double(3,2),
   Department varchar(15),
   PRIMARY KEY(Id)
);
```

Output TABLE:

Id	Name	Gender	Age	CGPA	Department

INPUT FORMAT:

```
SQL-> INSERT INTO
student_2(Id,Name,Gender,Age,CGPA,Department)VALUE
S
(108,'Naveed Rahman','Male',24,3.90,'CSE'),
(109,'Rokon','Male',19,3.00,'CSE'),
(110,'Mamun','Male',22,3.50,'EEE');
```

OUTPUT TABLE:

Id	Name	Gender	Age	CGPA	Department
108	Naveed Rahman	Male	21	3.80	CSE
109	Rokon	Male	19	3.00	CSE
110	Mamun	Male	22	3.50	EEE

INPUT FORMAT:

```
SQL-> SELECT Id,Name,CGPA FROM student_detailes
UNION
SELECT Id,Name,CGPA FROM student_2;
```

Output TABLE:

Id	Name	Gender	Age	CGPA	Department
101	Humayoun Kobir	Male	21	3.80	CSE
102	Naimur Rashid	Male	21	3.70	BBA
103	Abul Hasnat	Male	20	3.73	EEE
104	Sanjana Akrher	Female	20	3.85	CSE

Id	Name	Gender	Age	CGPA	Department
105	Abdullah Al Mam	Male	22	3.50	BBA
106	Farzana Afroz	Female	23	3.65	EEE
107	All Imran Munna	Male	25	3.30	BBA
109	Rokon	Male	19	3.00	CSE
108	Naveed Rahman	Male	21	3.80	CSE
110	Mamun	Male	22	3.50	EEE

❑ UNION ALL:

The UNION operator selects only distinct values by default. To allow duplicate values, we use UNION ALL.

➢ Syntax:

SELECT ***column_name(s)*** **FROM** ***table1***
UNION ALL
SELECT ***column_name(s)*** **FROM** ***table2*****;**

INPUT FORMAT:

```
SQL-> SELECT Id,Name,CGPA FROM
student_detailes
UNION ALL
SELECT Id,Name,CGPA FROM student_2;
```

OUTPUT TABLE:

Id	Name	Gender	Age	CGPA	Department
101	Humayoun Kobir	Male	21	3.80	CSE
102	Naimur Rashid	Male	21	3.70	BBA

Id	Name	Gender	Age	CGPA	Department
103	Abul Hasnat	Male	20	3.73	EEE
104	Sanjana Akrher	Female	20	3.85	CSE
105	Abdullah Al Mam	Male	22	3.50	BBA
106	Farzana Afroz	Female	23	3.65	EEE
107	All Imran Munna	Male	25	3.30	BBA
109	Rokon	Male	19	3.00	CSE
108	Naveed Rahman	Male	21	3.80	CSE
109	Rokon	Male	19	3.00	CSE
110	Mamun	Male	22	3.50	EEE

☐ HAVING Clause:

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

➢ Syntax:

SELECT ***column_name(s)***
FROM ***table_name***
WHERE ***condition***
GROUP BY ***column_name(s)***
HAVING ***condition***

INPUT FORMAT:

```
SQL->SELECT Name,SUM(CGPA),Department
FROM student_detailes
```

```
GROUP BY Name
HAVING SUM(CGPA) >3.50
ORDER BY SUM(CGPA);
```

OUTPUT DATABASE:

Name	SUM(CGPA) ▲ 1	Department
Farzana Afroz	3.65	EEE
Naimur Rashid	3.70	BBA
Abul Hasnat	3.73	EEE
Humayoun Kobir	3.80	CSE
Sanjana Akther	3.85	CSE

OUTPUT TABLE:

Name	SUM(CGPA)	Department
Farzana Afroz	3.65	EEE
Naimur Rashid	3.70	BBA
Abul Hasnat	3.73	EEE
Humayoun Kobir	3.80	CSE
Sanjana Akther	3.85	CSE

☐ EXISTS Operator:

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.

➢ **Syntax:**

SELECT ***column_name(s)***
FROM ***table_name***
WHERE EXISTS
(SELECT ***column_name*** **FROM** ***table_name*** **WHERE** ***condition*****);**

INPUT FORMAT:

```
SQL-> SELECT Name,CGPA
FROM student_detailes
WHERE EXISTS (SELECT Reg_Number FROM exam_date
              WHERE exam_date.Id =
student_detailes.Id)
```

OUTPUT DATABASE:

Name	CGPA
Humayoun Kobir	3.80
Sanjana Akther	3.85
Rokon	3.00

OUTPUT TABLE:

Name	SUM(CGPA)
Humayoun Kobir	3.80
Sanjana Akther	3.85
Rokon	3.00

☐ ANY Operator:

The **ANY** operator:

- returns a boolean value as a result
- returns TRUE if ANY of the subquery values meet the condition

ANY means that the condition will be true if the operation is true for any of the values in the range.

➢ Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ANY
  (SELECT column_name
  FROM table_name
  WHERE condition);
```

INPUT FORMAT:

```
SQL-> SELECT Reg_Number,Exam_date
FROM exam_date
WHERE Id = ANY
  (SELECT Id
  FROM student_detailes
  WHERE Department = 'CSE');
```

OUTPUT DATABASE:

Reg_Number	Exam_date
13176165	2022-06-21 09:45:00
13176166	2022-06-20 13:30:00
13176167	2022-06-25 16:00:00

OUTPUT TABLE:

Reg_Number	Exam_date
13176165	2022-06-21 09:45:00
13176166	2022-06-20 13:30:00
13176167	2022-06-25 16:00:00

☐ SELECT INTO Statement:

The SELECT INTO statement copies data from one table into a new table.

➢ **Syntax:**

```
SELECT column1, column2, column3, ...
INTO new_table [IN external_db]
FROM oldtable
WHERE condition;
```

☐ CASE Statement:

The **CASE statement** goes through conditions and returns a value when the first condition is met (like an if-then-else statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause.If there is no ELSE part and no conditions are true, it returns NULL.

➢ **Syntax:**

```
SELECT Column(N),
CASE
  WHEN Condition THEN 'Statement'
  ELSE 'Statement"
END AS  tempname
FROM Table_name
```

INPUT FORMAT:

```
SQL-> SELECT Name, CGPA,
CASE
   WHEN CGPA > 3.75 THEN 'The CGPA is greater than 3.75'
   WHEN CGPA >3.50 AND CGPA<3.75 THEN 'The CGPA is under
than 3.75'
   WHEN CGPA =3.50 THEN 'The CGPA is 3.50'
   ELSE 'The CGPA is under 3.50'
END AS RESULT
FROM student_detailes;
```

OUTPUT DATABASE:

Name	CGPA	RESULT
Humayoun Kobir	3.80	The CGPA is greater than 3.75
Naimur Rashid	3.70	The CGPA is under than 3.75
Abul Hasnat	3.73	The CGPA is under than 3.75
Sanjana Akther	3.85	The CGPA is greater than 3.75
Abdullah al Mam	3.50	The CGPA is 3.50
Farzana Afroz	3.65	The CGPA is under than 3.75
Al Imran Munna	3.30	The CGPA is under 3.50
Rokon	3.00	The CGPA is under 3.50

Output TABLE:

Name	CGPA	RESULT
Humayoun Kobir	3.80	The CGPA is greater than 3.75
Naimur Rashid	3.70	The CGPA is under than 3.75
Abul Hasnat	3.73	The CGPA is under than 3.75
Sanjana Akrher	3.85	The CGPA is greater than 3.75
Abdullah Al Mam	3.50	The CGPA is 3.50
Farzana Afroz	3.65	The CGPA is under than 3.75
All Imran Munna	3.30	The CGPA is under 3.50
Rokon	3.00	The CGPA is under 3.50

☐VIEW Statement

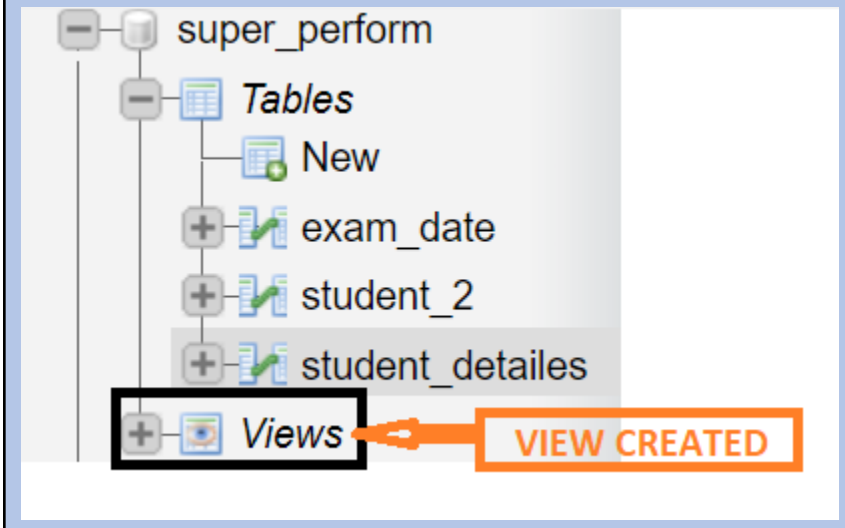
In SQL, a view is a virtual table based on the result-set of an SQL statement.A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.You can add **SQL statements** and functions to a view and present the data as if the data were coming from one single table.

➢ Syntax:

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

INPUT FORMAT:

```
SQL-> CREATE VIEW student_View AS
SELECT Id,Name,Department
FROM student_detailes;
```



➢ Syntax:

```
SELECT *
FROM viewName;
```

OUTPUT DATABASE:

SQL->`SELECT * FROM student_view;`

Id	Name	Department
101	Humayoun Kobir	CSE
102	Naimur Rashid	BBA
103	Abul Hasnat	EEE
104	Sanjana Akther	CSE
105	Abdullah al Mam	BBA
106	Farzana Afroz	EEE
107	Al Imran Munna	BBA
109	Rokon	CSE

OUTPUT TABLE:

Id	Name	Department
101	Humayoun Kobir	CSE
102	Naimur Rashid	BBA
103	Abul Hasnat	EEE
104	Sanjana Akrher	CSE
105	Abdullah Al Mam	BBA
106	Farzana Afroz	EEE
107	All Imran Munna	BBA
109	Rokon	CSE

UPDATE VIEW

➢ **Syntax:**

UPDATE ***viewName***
SET ***ColumnName=Value***
WHERE ***Condition;***

OUTPUT DATABASE:

```
SQL->UPDATE student_view
SET Department = 'CSE'
WHERE Id = 103;
```

Id	Name	Department
101	Humayoun Kobir	CSE
102	Naimur Rashid	BBA
103	Abul Hasnat	CSE
104	Sanjana Akther	CSE
105	Abdullah al Mam	BBA
106	Farzana Afroz	EEE
107	Al Imran Munna	BBA
109	Rokon	CSE

☐ Date Function

As long as your data contains only the date portion, your queries will work as expected. However, if a time portion is involved, it gets complicated.

Before talking about the complications of querying for dates, we will look at the most important built-in functions for working with dates.

Most important built-in date functions:

Function	Description
NOW()	Returns the current date and time
CURDATE()	Returns the current date
CURTIME()	Returns the current time
DATE()	Extracts the date part of a date or date/time expression

OUTPUT DATABASE: [CURDATE()]

```
SQL->SELECT CURDATE() AS TODAY;
```

TODAY
2022-06-19

OUTPUT DATABASE:[NOW()]

```
SQL->SELECT NOW();
```

NOW()
2022-06-19 16:14:03

OUTPUT DATABASE:[CURTIME()]

```
SQL->SELECT CURTIME();
```

CURTIME()
16:17:22

Table 2:

Reg_Number	Id	Exam_date
13176165	101	2022-06-21 09:45:00
13176166	109	2022-06-20 13:30:00
13176167	104	2022-06-25 16:00:00
13176168	107	2022-06-27 15:15:00

DAYNAME():The DAYNAME() function returns the weekday name for a given date

➢ **Syntax:**

DAYNAME(date)

INPUT FORMAT:

```
SQL->SELECT Exam_date,DAYNAME(Exam_date)
FROM exam_date;
```

OUTPUT DATABASE:

Exam_date	DAYNAME(Exam_date)
2022-06-21 09:45:00	Tuesday
2022-06-20 13:30:00	Monday
2022-06-25 16:00:00	Saturday
2022-06-27 15:15:00	Monday

OUTPUT TABLE:

Exam_date	DATENAME(Exam_date)
2022-06-21 09:45:00	Tuesday
2022-06-20 13:30:00	Monday
2022-06-25 16:00:00	Saturday
2022-06-27 15:15:00	Monday

ADDDATE(): The ADDDATE() function adds a time/date interval to a date and then returns the date.

➢ **Syntax:**

ADDDATE(*date*, INTERVAL *value addunit*)

INPUT FORMAT:

```
SQL->SELECT Exam_date,ADDDATE(Exam_date,INTERVAL 4 DAY)
FROM exam_date;
```

OUTPUT DATABASE:

Exam_date	ADDDATE(Exam_date,INTERVAL 4 DAY)
2022-06-21 09:45:00	2022-06-25 09:45:00
2022-06-20 13:30:00	2022-06-24 13:30:00
2022-06-25 16:00:00	2022-06-29 16:00:00
2022-06-27 15:15:00	2022-07-01 15:15:00

OUTPUT TABLE:

Exam_date	ADDDATE(Exam_date,INTERVAL 4 DAY)
2022-06-21 09:45:00	2022-06-25 09:45:00
2022-06-20 13:30:00	2022-06-24 13:30:00
2022-06-25 16:00:00	2022-06-29 16:00:00
2022-06-27 15:15:00	2022-07-01 15:15:00

CONCLUSION:

In this tutorial,we've covered almost all SQL commands within 3 tables and implemented them.It’s our pleasure to work on this and hope that it would be very appreciative for a desirable audience.

----------X----------