Welcome you all

MySQL

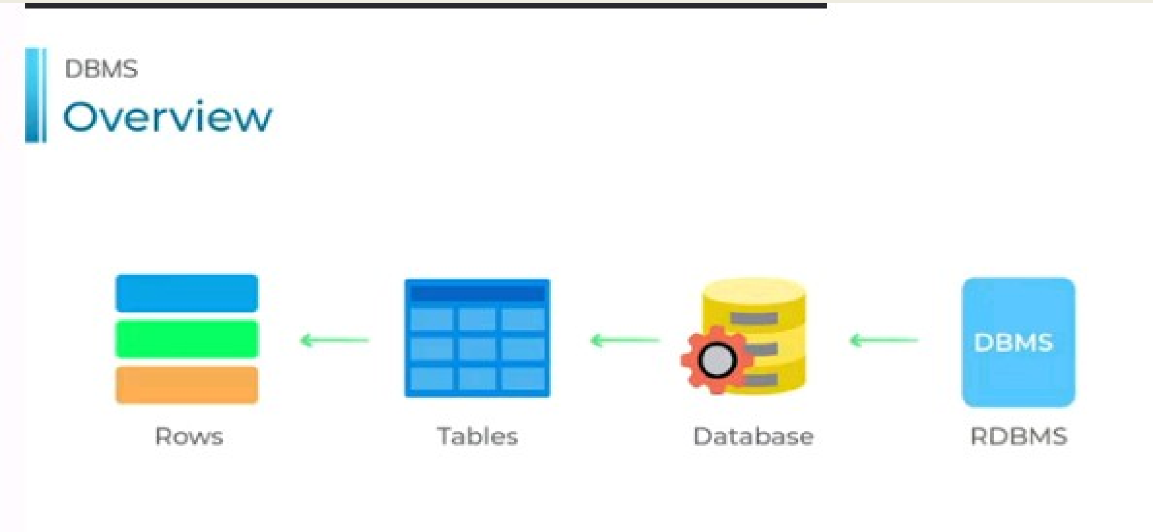
D.Sakthivel
Assistant Professor & Trainer,
KGiSL Micro College
KGiSL Campus, Coimbatore – 641 035.

Day 1

- RDBMS
- WHAT IS SQL?
  - DDL
  - DML
- MySQL
  - Data types
- **SELECT QUERY : WHERE CLAUSE - Comparison Operator**
  - Comparison Operator
  - LIKE OPERATOR
  - LOGICAL OPERATOR
  - BETWEEN AND IN
  - ORDERBY and DISTINCT
  - AGGREGATE FUNCTION
  - GROUP BY AND HAVING CLAUSE
  - OPERATORS
  - EXPRESSIONS IN QUERYING

RDBMS

- A Relation Database Management system (RDBMS) is **a database management system that is based on the relational model**.
- It has the following major components: ***Table, Record/Tuple/Row, Field, and Column/Attribute.***
- Examples of the most popular RDBMS are MYSQL, Oracle, IBM DB2, and Microsoft SQL Server database.



What is Relation Database?

- A relational database is a database divided into logical units called tables, where tables are related to one another within the database.
- Relational database allows data to be broken down into logical, smaller, and manageable units for easier maintenance and better performance.
- **Tables are related to one another through common keys or fields** in a relational database system.
- The desired data may exist in more than one table, you can easily join multiple tables together to get combined data set using a gle query.

- **Setting Up Work Environment for Practicing SQL**
- You can install a free, open-source DBMS.
- MySQL is the most popular and widely supported open-source database management system.
- It is very easy to download and use and available for both Windows and Linux (or UNIX) operating system.
- You can download it freely from here https://dev.mysql.com/downloads/mysql/

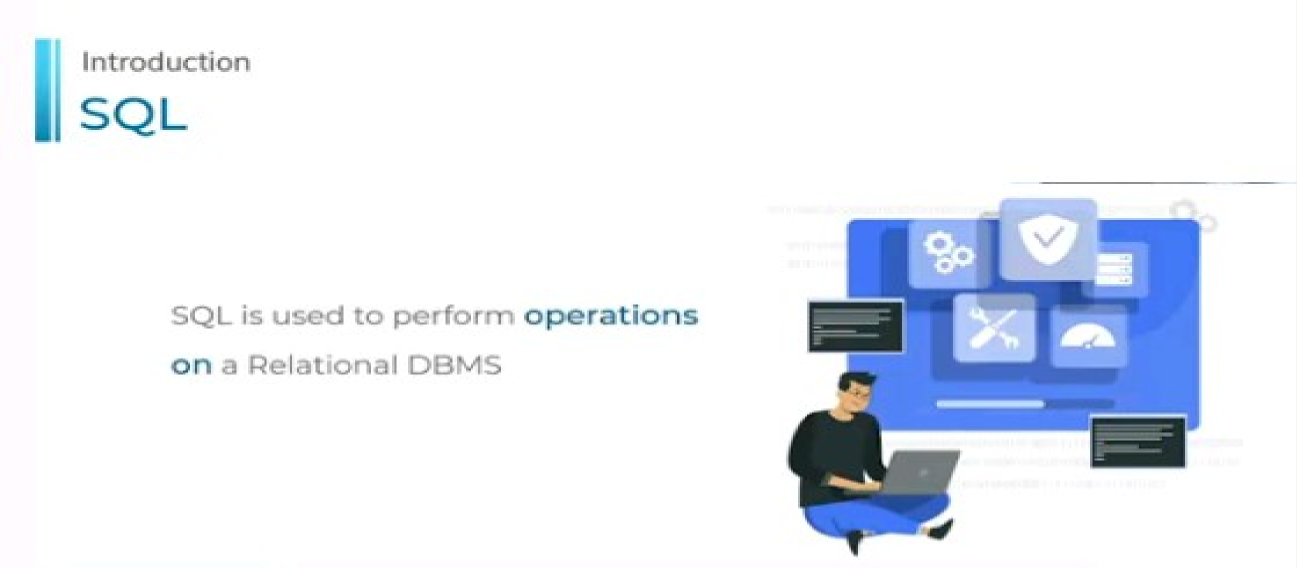
SQL(Structured Query Language)

- SQL is a **standard language designed for managing data in relational database management system.**
- SQL stands for Structured Query Language.
- SQL is a standard programming language specifically designed for storing, retrieving, managing or manipulating the data inside a relational database management system (RDBMS).
- SQL became an ISO standard in 1987.
- SQL is the most widely-implemented database language and supported by the popular relational database systems, like MySQL, SQL Server, and Oracle.
- However, some features of the SQL standard are implemented differently in different database systems.
- SQL was originally developed at IBM in the early 1970s. Initially it was called SEQUEL (Structured English Query Language) which was later changed to SQL (pronounced as S-Q-L).

SQL(Structured Query Language)

There are lot more things you can do with SQL:

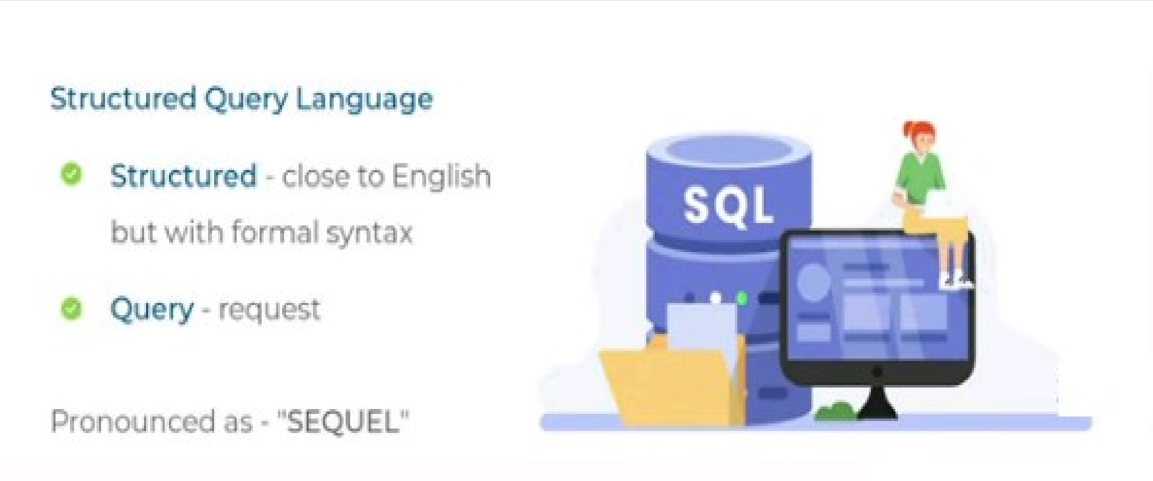
- You can create a database.
- You can create tables in a database.
- You can query or request information from a database.
- You can insert records in a database.
- You can update or modify records in a database.
- You can delete records from the database.
- You can set permissions or access control within the database for data security.
- You can create views to avoid typing frequently used complex queries.



Introduction

SQL

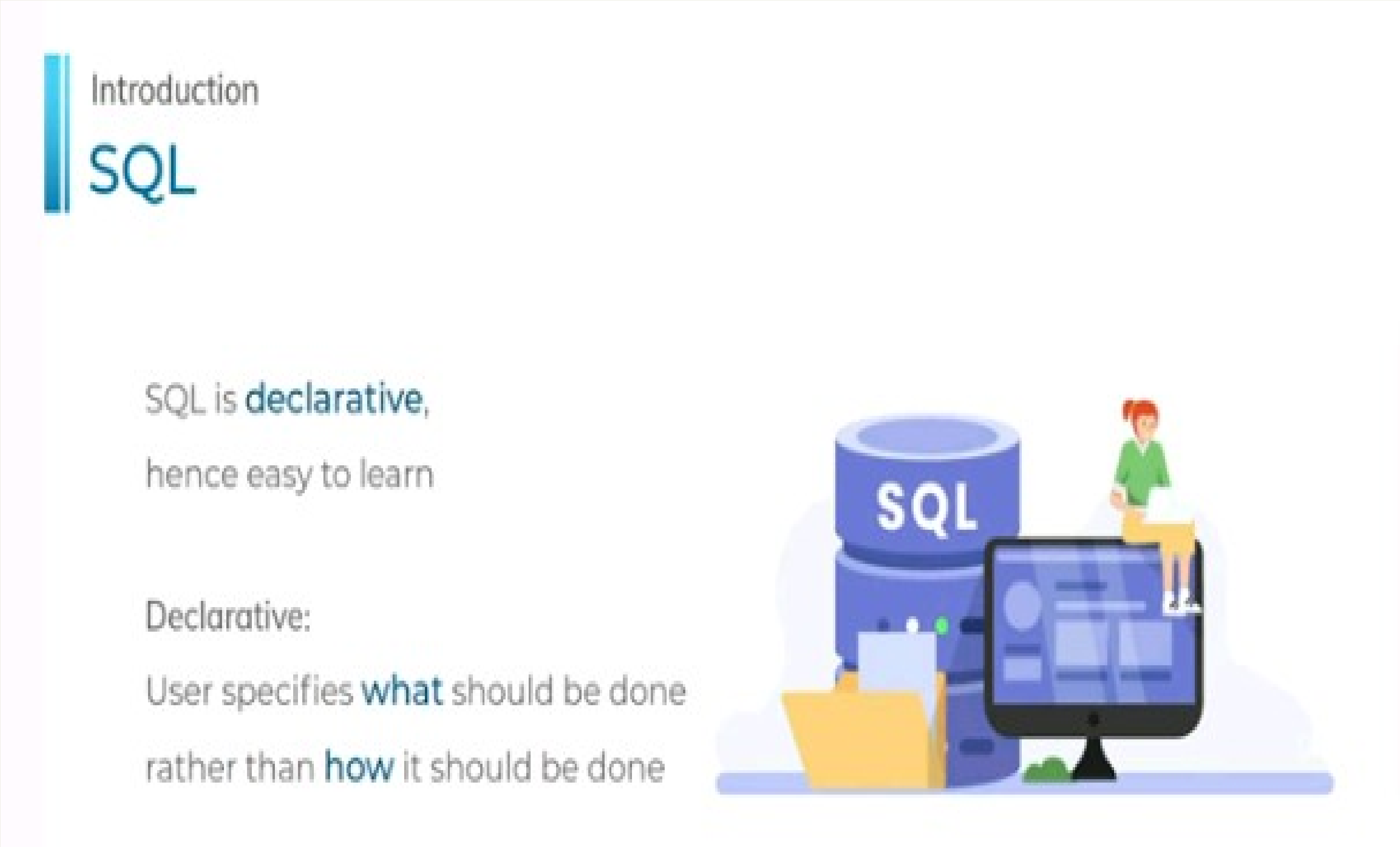
SQL is used to perform **operations on** a Relational DBMS



Structured Query Language

- **Structured** - close to English but with formal syntax
- **Query** - request

Pronounced as - "SEQUEL"



Introduction

SQL

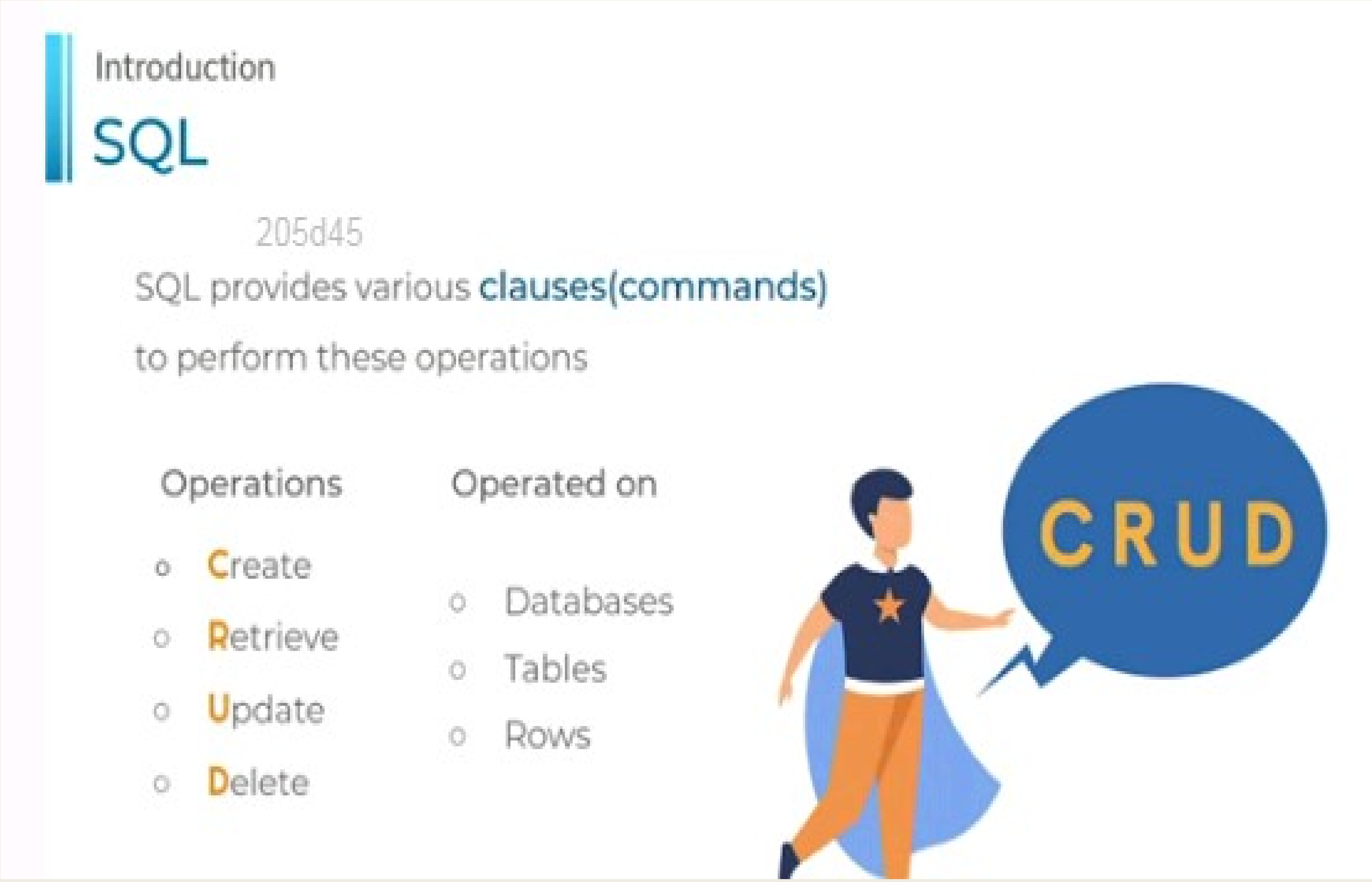
SQL is **declarative**,

hence easy to learn

Declarative:

User specifies **what** should be done

rather than **how** it should be done



Introduction

SQL

SQL provides various **clauses(commands)** to perform these operations

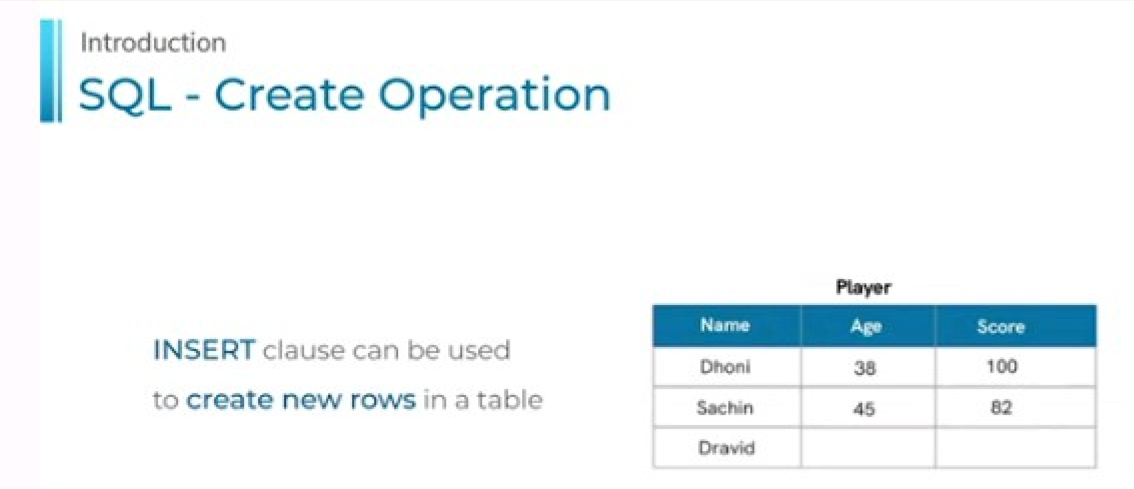
Operations

- Create
- Retrieve
- Update
- Delete

Operated on

- Databases
- Tables
- Rows

KGiSL MicrCollege



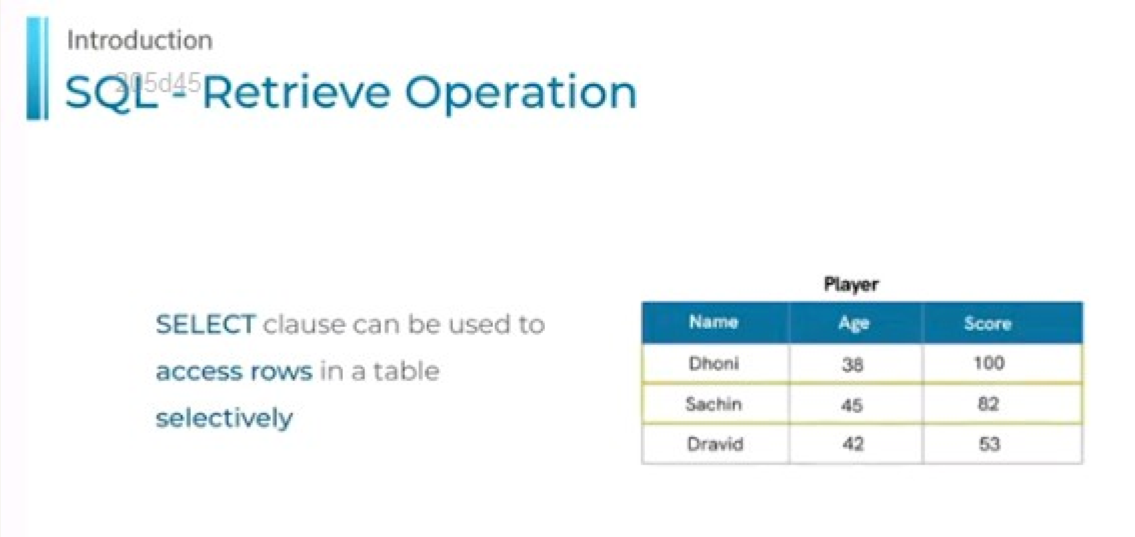
Introduction

SQL - Create Operation

INSERT clause can be used to **create new rows** in a table

Player

Name	Age	Score
Dhoni	38	100
Sachin	45	82
Dravid		



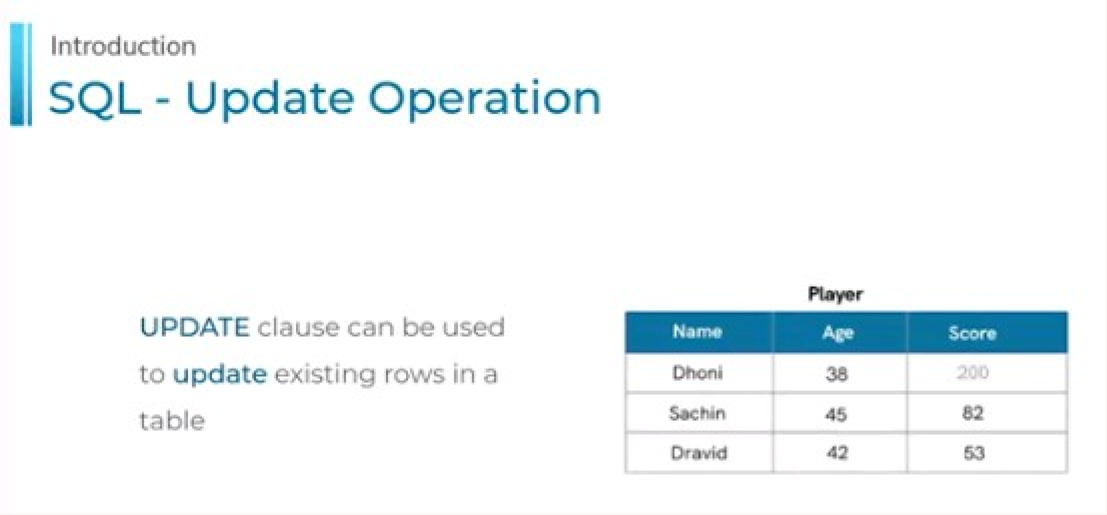
Introduction

SQL - Retrieve Operation

SELECT clause can be used to **access rows** in a table **selectively**

Player

Name	Age	Score
Dhoni	38	100
Sachin	45	82
Dravid	42	53



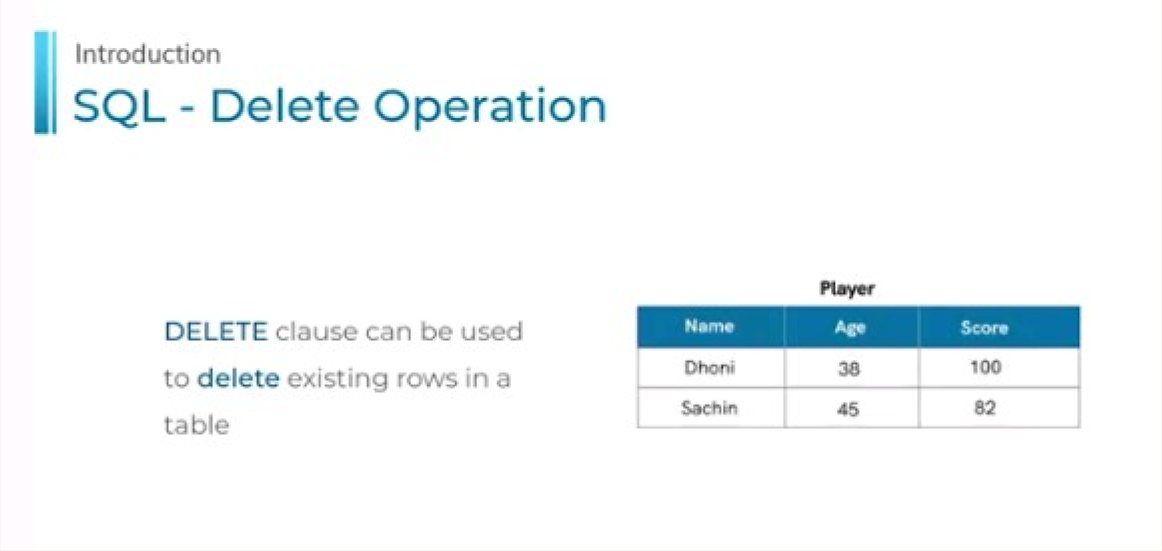
Introduction

SQL - Update Operation

UPDATE clause can be used to update existing rows in a table

Player

Name	Age	Score
Dhoni	38	200
Sachin	45	82
Dravid	42	53



Introduction

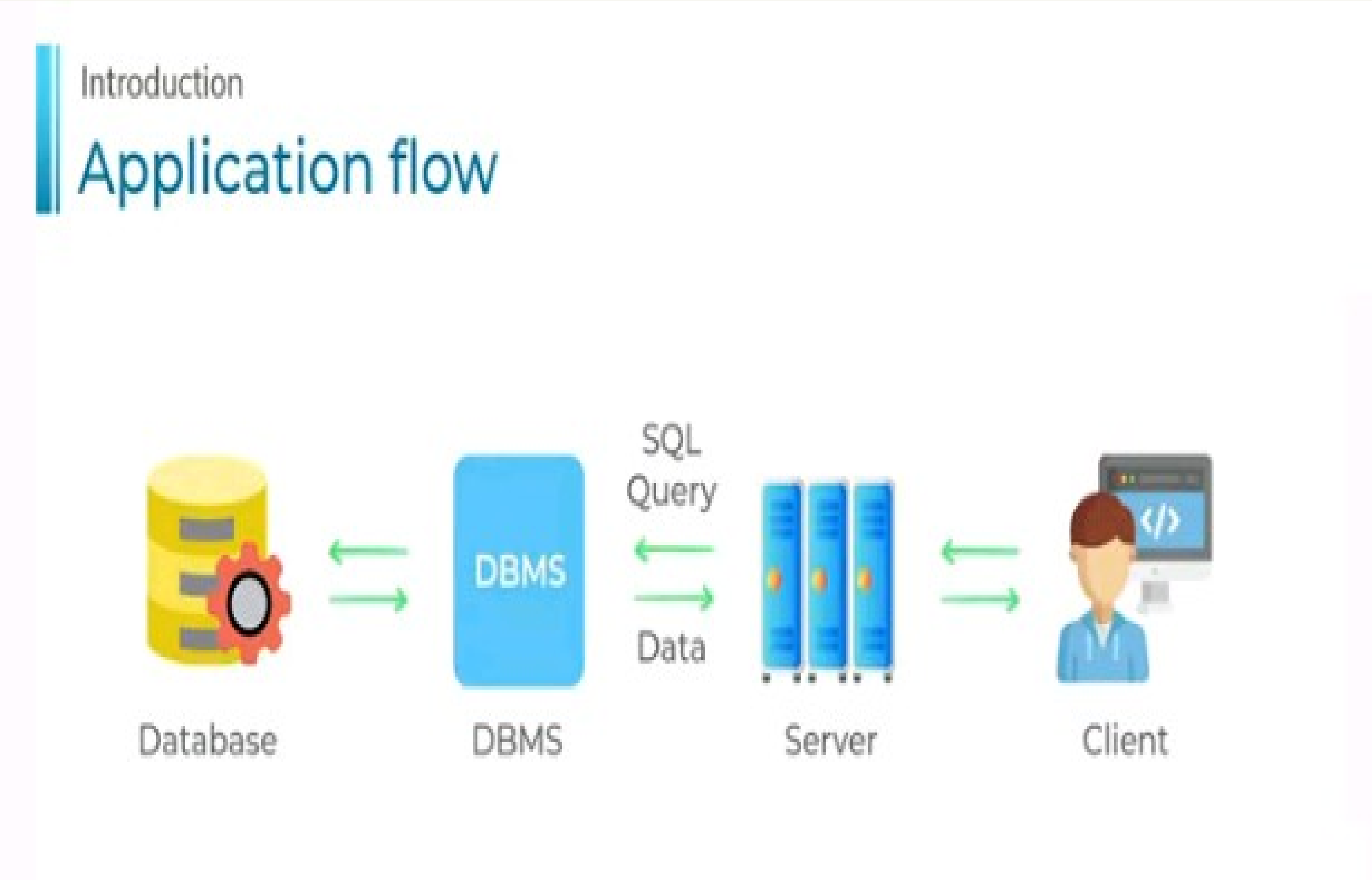
SQL - Delete Operation

DELETE clause can be used to delete existing rows in a table

Player

Name	Age	Score
Dhoni	38	100
Sachin	45	82

KGiSL MicroCollege



Introduction

Application flow

MySQL

Overview

- MySQL is a relational database management system based on the Structured Query Language,
- MySQL is the popular language for accessing and managing the records in the database.
- MySQL is open-source and free software under the GNU license. It is supported by **Oracle Company**.
- It is developed, marketed, and supported by **MySQL AB, a Swedish company**, and written in C programming language and C++ programming language.
- The project of MySQL was started in 1979 when MySQL's inventor **Michael Widenius** developed an in-house database tool called **UNIREG** for managing databases.

Data types

1. Numeric Data type

TINYINT : It is a very small integer that can be signed or unsigned. If signed, the allowable range is from **-128 to 127.** If unsigned, the allowable range is from 0 to 255. We can specify a **width of up to 4 digits**. It takes **1 byte for storage.**

SMALLINT: It is a small integer that can be signed or unsigned. If signed, the allowable range is from **-32768 to 32767**. If unsigned, the allowable range is from 0 to 65535. We can specify a width of up to **5 digits.** It requires **2 bytes for storage.**

MEDIUMINT : It is a medium-sized integer that can be signed or unsigned. If signed, the allowable range is from **-8388608 to 8388607**. If unsigned, the allowable range is from 0 to 16777215. We can specify a width of up to **9 digits**. It requires **3 bytes for storage.**

INT : It is a normal-sized integer that can be signed or unsigned. If signed, the allowable range is from **-2147483648 to 2147483647**. If unsigned, the allowable range is from 0 to

cify a width of up to **11 digits**. It

ge.

KGiSL MicrCollege

Data types

1. Numeric Data type

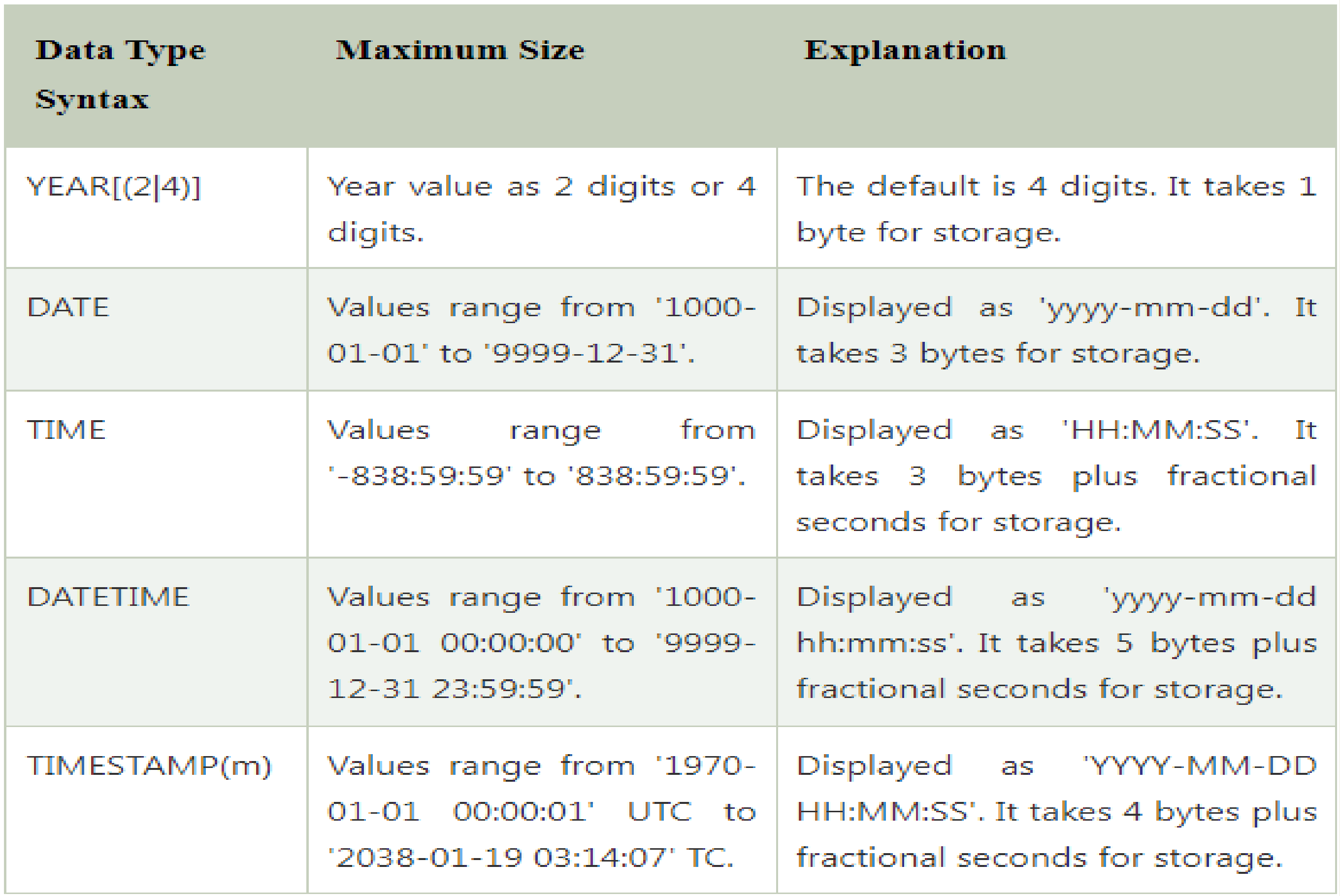
FLOAT(m,d): It is a floating-point number that cannot be unsigned. You can define the **display length (m) and the number of decimals (d).** This is not required and will default to 10,2, where 2 is the number of decimals, and **10 is the total number of digits** (including decimals). Decimal precision can go to **24 places for a float type**. It requires **2 bytes for storage.**

DOUBLE(m,d): It is a double-precision floating-point number that cannot be unsigned. This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to **53 places for a double**. It requires **8 bytes for storage.**

BOOL: It is used only for the true and false condition. It considered numeric value 1 as true and 0 as false. [illegible] to the BOOL.

Data types

2. Date and Time Data type



Data Type Syntax	Maximum Size	Explanation
YEAR[(2|4)]	Year value as 2 digits or 4 digits.	The default is 4 digits. It takes 1 byte for storage.
DATE	Values range from '1000-01-01' to '9999-12-31'.	Displayed as 'yyyy-mm-dd'. It takes 3 bytes for storage.
TIME	Values range from '-838:59:59' to '838:59:59'.	Displayed as 'HH:MM:SS'. It takes 3 bytes plus fractional seconds for storage.
DATETIME	Values range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.	Displayed as 'yyyy-mm-dd hh:mm:ss'. It takes 5 bytes plus fractional seconds for storage.
TIMESTAMP(m)	Values range from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' TC.	Displayed as 'YYYY-MM-DD HH:MM:SS'. It takes 4 bytes plus fractional seconds for storage.

Data types

2. String Data type

CHAR(size)	It can have a maximum size of 255 characters.	Here size is the number of characters to store. Fixed-length strings. Space padded on the right to equal size characters.
VARCHAR(size)	It can have a maximum size of 255 characters.	Here size is the number of characters to store. Variable-length string.
TEXT(size)	Maximum size of 65,535 characters.	Here size is the number of characters to store.

Data types

2. String Data type

CHAR(size)	It can have a maximum size of 255 characters.	Here size is the number of characters to store. Fixed-length strings. Space padded on the right to equal size characters.
VARCHAR(size)	It can have a maximum size of 255 characters.	Here size is the number of characters to store. Variable-length string.
TEXT(size)	Maximum size of 65,535 characters.	Here size is the number of characters to store.

Data types

Data Type	Description
INT	Stores numeric values in the range of -2147483648 to 2147483647
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.
DATETIME	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 Database:

Syntax:

```
CREATE DATABASE  database_name ;
```

Example:

```
mysql> CREATE DATABASE employeesdb;
```

Create Database:

Syntax:

```
CREATE DATABASE  database_name ;
```

Example:

```
mysql> CREATE DATABASE studentdb;
```

We can review the newly created database using the below query that returns the database name, character set, and collation of the database:

```
mysql> SHOW CREATE DATABASE studentdb;
```

- **To show all Database:**

```
mysql> SHOW DATABASES;
```

- **SELECT/Access Database**

Syntax:

```
USE database_name;
```

Example:

```
USE studentdb;
```

- **SHOW ALL TABLES PRESENT IN A DATABASE:**

```
mysql> SHOW TABLES;
```

SHOW VERSION OF MYSQL

```
mysql> select version();
```

SHOW CURRENT DATE:

```
mysql> select CURRENT_DATE;
```

To show all USERS:

```
mysql> select users();
```

SHOW CURRENT DATE AND TIME

```
mysql> select now();
```

SQL types of query language:

DDL – DATA DEFINITION LANGUAGE

Create, Alter, Truncate, Drop

DML – DATA MANIPULATION LANGUAGE

Select, Insert, Update, Delete

DCL – DATA CONTROL LANGUAGE – GRANT & REVOKE

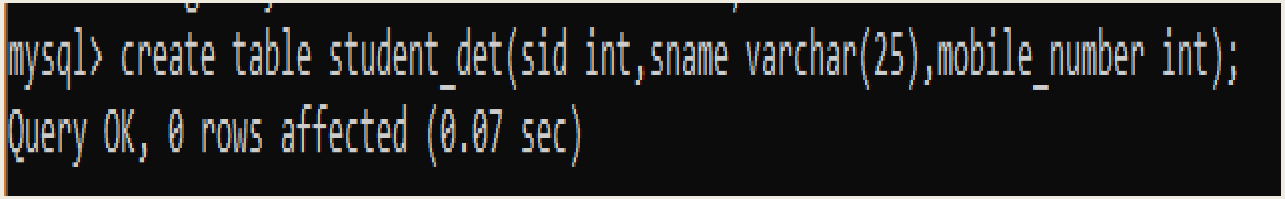
TCL – TRANSACTION CONTROL LANGAUGE –COMMIT & SAVE POINT

CREATE TABLE:

To Create a table with relevant fields

Syntax:

```
CREATE TABLE  table_name(
    column_definition1 DATATYPE,
    column_definition2 DATATYPE,
    ........,
    table_constraints
);
```

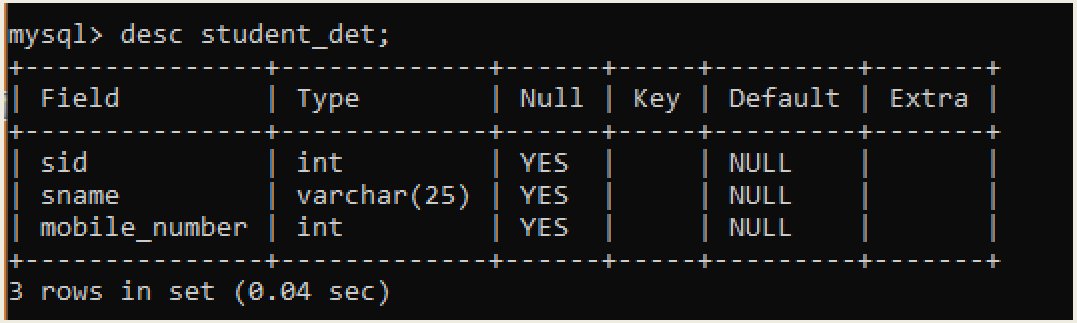


```
mysql> create table student_det(sid int,sname varchar(25),mobile_number int);
Query OK, 0 rows affected (0.07 sec)
```

To See the table structure:

Syntax:

mysql> DESC <TABLENAME>;



```
mysql> desc student_det;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| sid           | int         | YES  |     | NULL    |       |
| sname         | varchar(25) | YES  |     | NULL    |       |
| mobile_number | int         | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
3 rows in set (0.04 sec)
```

ALTER Table

To add new column in to the created table.

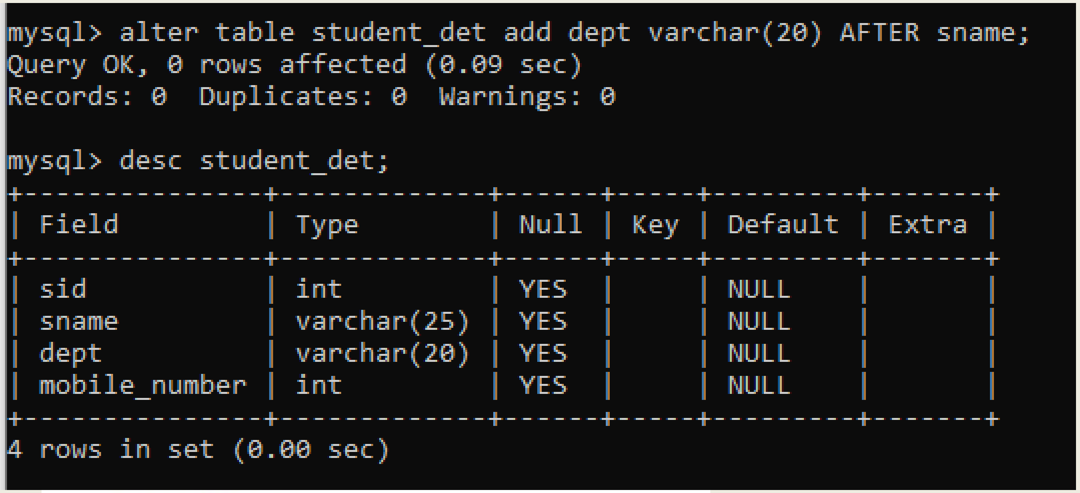
A) ADD

Syntax:

ALTER TABLE table_name

ADD new_column_name column_definition

[**FIRST** | **AFTER** column_name];

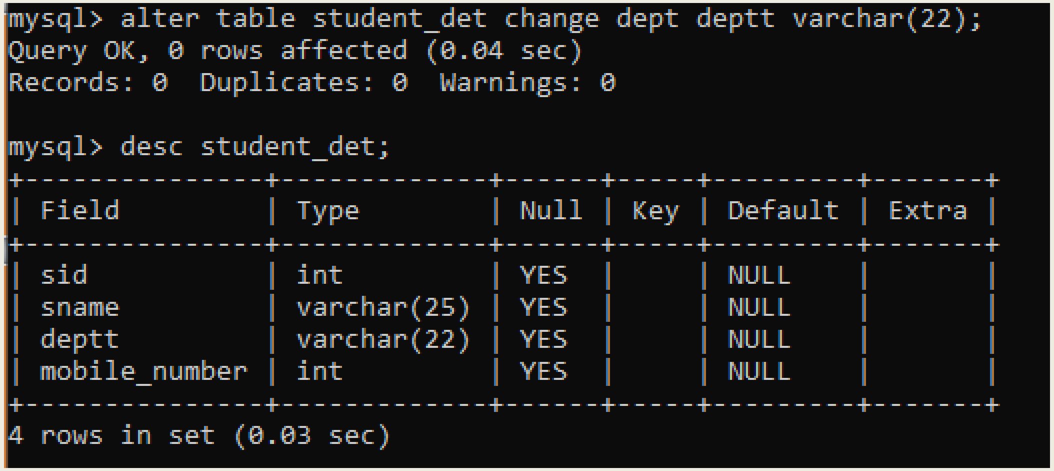


```
mysql> alter table student_det add dept varchar(20) AFTER sname;
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc student_det;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| sid           | int         | YES  |     | NULL    |       |
| sname         | varchar(25) | YES  |     | NULL    |       |
| dept          | varchar(20) | YES  |     | NULL    |       |
| mobile_number | int         | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

ALTER Table

B) CHANGE – TO rename the column in a table

ALTER TABLE table_name CHANGE old_column_name new_col_name Data Type;

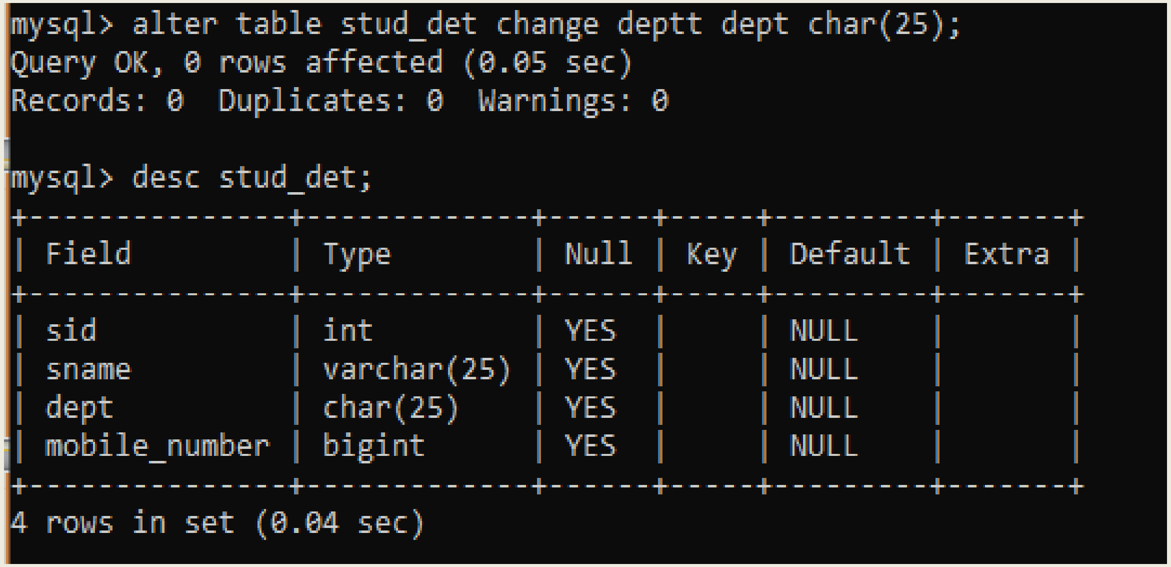


```
mysql> alter table student_det change dept deptt varchar(22);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc student_det;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| sid           | int         | YES  |     | NULL    |       |
| sname         | varchar(25) | YES  |     | NULL    |       |
| deptt         | varchar(22) | YES  |     | NULL    |       |
| mobile_number | int         | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
4 rows in set (0.03 sec)
```

ALTER Table

B) CHANGE – TO rename the column in a table

ALTER TABLE table_name CHANGE old_column_name new_col_name Data Type;



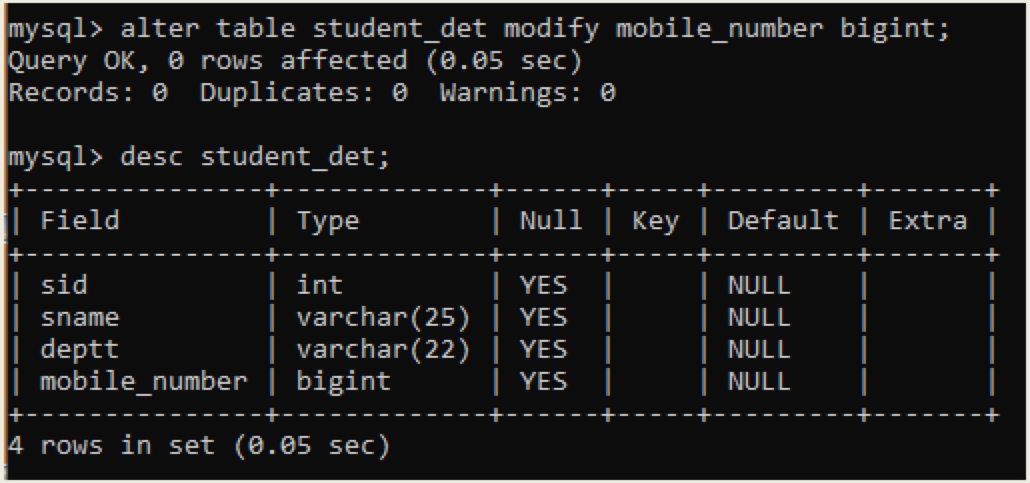
```
mysql> alter table stud_det change deptt dept char(25);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc stud_det;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| sid           | int         | YES  |     | NULL    |       |
| sname         | varchar(25) | YES  |     | NULL    |       |
| dept          | char(25)    | YES  |     | NULL    |       |
| mobile_number | bigint      | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
4 rows in set (0.04 sec)
```

ALTER Table

B) Modify –To change the data type of a column in a table.

Syntax:

ALTER TABLE *table_name*
MODIFY COLUMN *column_name datatype*;



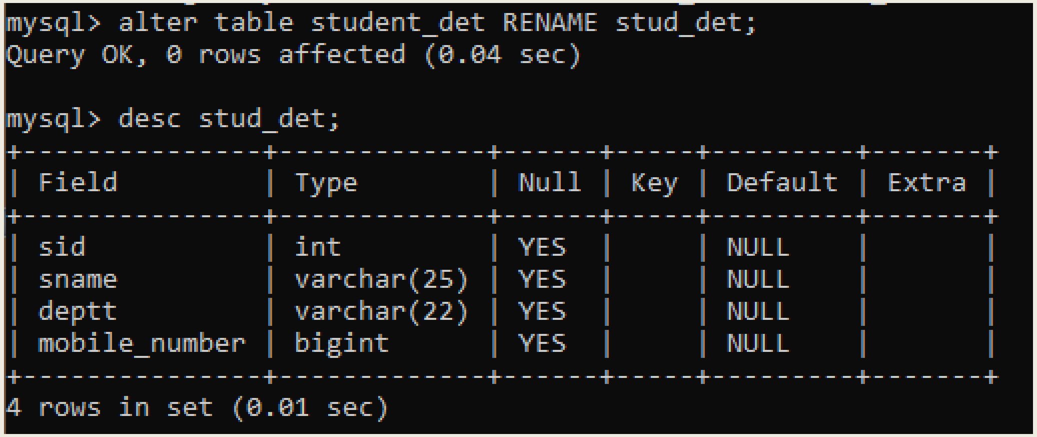
```
mysql> alter table student_det modify mobile_number bigint;
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc student_det;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| sid           | int         | YES  |     | NULL    |       |
| sname         | varchar(25) | YES  |     | NULL    |       |
| deptt         | varchar(22) | YES  |     | NULL    |       |
| mobile_number | bigint      | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
4 rows in set (0.05 sec)
```

❑Rename Table:

To Rename a existing table in MySQL.

SYNTAX:

ALTER TABLE old_table RENAME new_table;



```
mysql> alter table student_det RENAME stud_det;
Query OK, 0 rows affected (0.04 sec)

mysql> desc stud_det;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| sid           | int         | YES  |     | NULL    |       |
| sname         | varchar(25) | YES  |     | NULL    |       |
| deptt         | varchar(22) | YES  |     | NULL    |       |
| mobile_number | bigint      | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

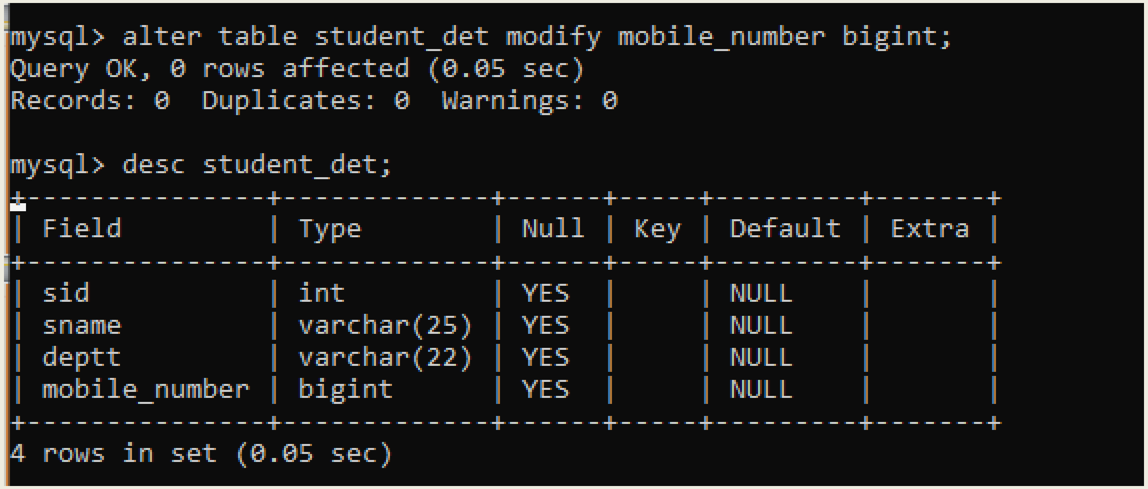
ALTER Table

B) MODIFY – To modify the size/change the data type of a column in a table.

ALTER TABLE table_name

MODIFY column_name column_definition

[**FIRST** | **AFTER** column_name];



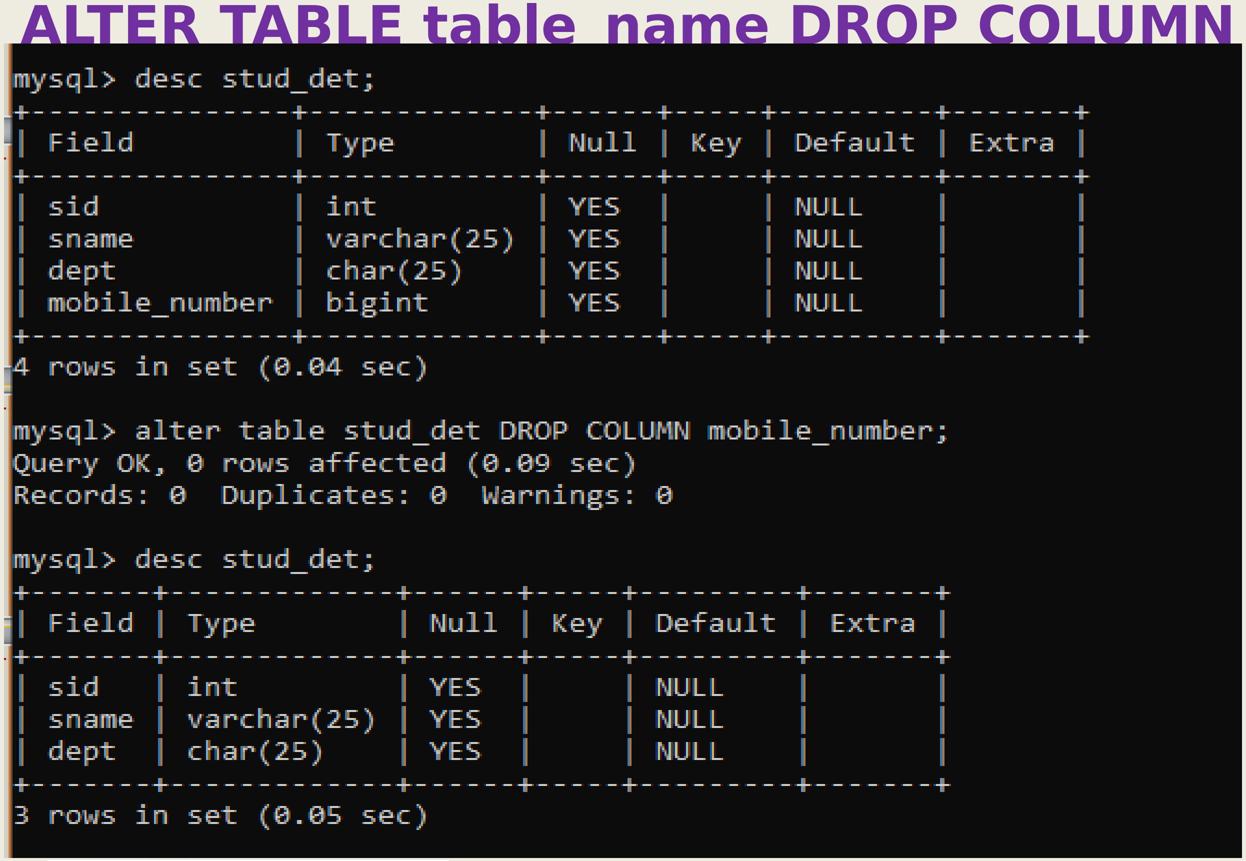
```
mysql> alter table student_det modify mobile_number bigint;
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc student_det;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| sid           | int         | YES  |     | NULL    |       |
| sname         | varchar(25) | YES  |     | NULL    |       |
| deptt         | varchar(22) | YES  |     | NULL    |       |
| mobile_number | bigint      | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
4 rows in set (0.05 sec)
```

ALTER Table

Drop column in table – To Delete column in a table.

Syntax

ALTER TABLE table_name DROP COLUMN



```
mysql> desc stud_det;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| sid           | int         | YES  |     | NULL    |       |
| sname         | varchar(25) | YES  |     | NULL    |       |
| dept          | char(25)    | YES  |     | NULL    |       |
| mobile_number | bigint      | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
4 rows in set (0.04 sec)

mysql> alter table stud_det DROP COLUMN mobile_number;
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc stud_det;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| sid   | int         | YES  |     | NULL    |       |
| sname | varchar(25) | YES  |     | NULL    |       |
| dept  | char(25)    | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.05 sec)
```

❑TRUNCATE Table:

The TRUNCATE statement in MySQL removes the complete data without removing its structure.

SYNTAX:

```
TRUNCATE [TABLE] table_name;
```

❑DROP Table:

Drop Table statement to delete the existing table.

SYNTAX:

```
mysql> DROP TABLE  table_name;
```

❑DROP COLUMN

SYNTAX:

ALTER TABLE table_name **DROP COLUMN** column_name;

❑CHANGE COLUMN NAME:

SYNTAX:

ALTER TABLE table_name

CHANGE **COLUMN** old_column_name new_column_name Data Type;

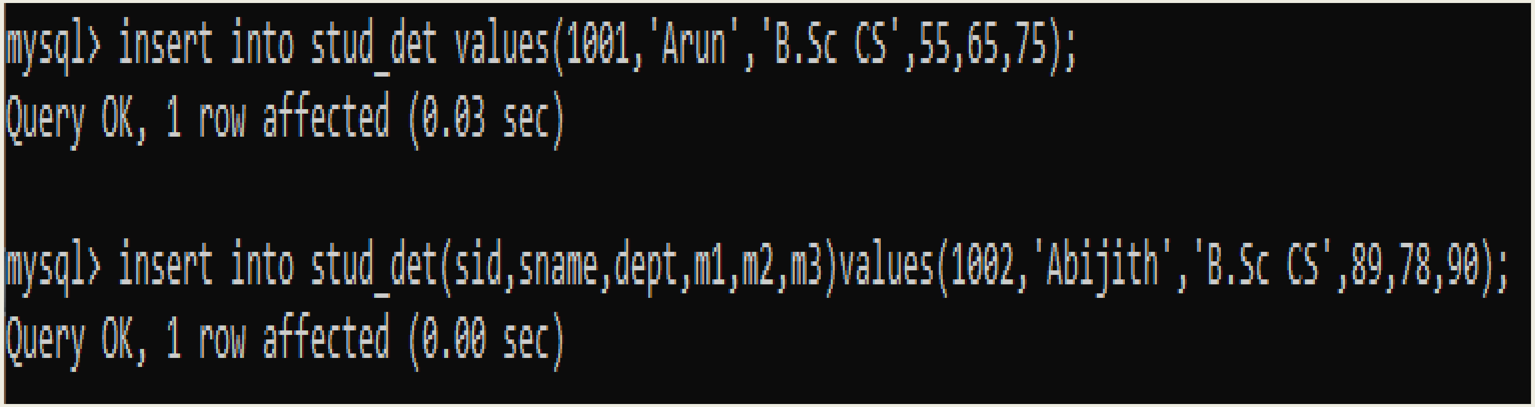
ALTER TABLE table_name

RENAME **COLUMN** old_column_name **TO** new_column_name;

❑ INSERT Statement

SYNTAX: - Single Row Insertion

INSERT INTO table_name (field1, field2,...fieldN)
VALUES
(value1, value2,...valueN);



```
mysql> insert into stud_det values(1001,'Arun','B.Sc CS',55,65,75);
Query OK, 1 row affected (0.03 sec)

mysql> insert into stud_det(sid,sname,dept,m1,m2,m3)values(1002,'Abijith','B.Sc CS',89,78,90);
Query OK, 1 row affected (0.00 sec)
```

❑ INSERT Statement

SYNTAX:

If we want to insert **multiple records** within a single command, use the following statement:

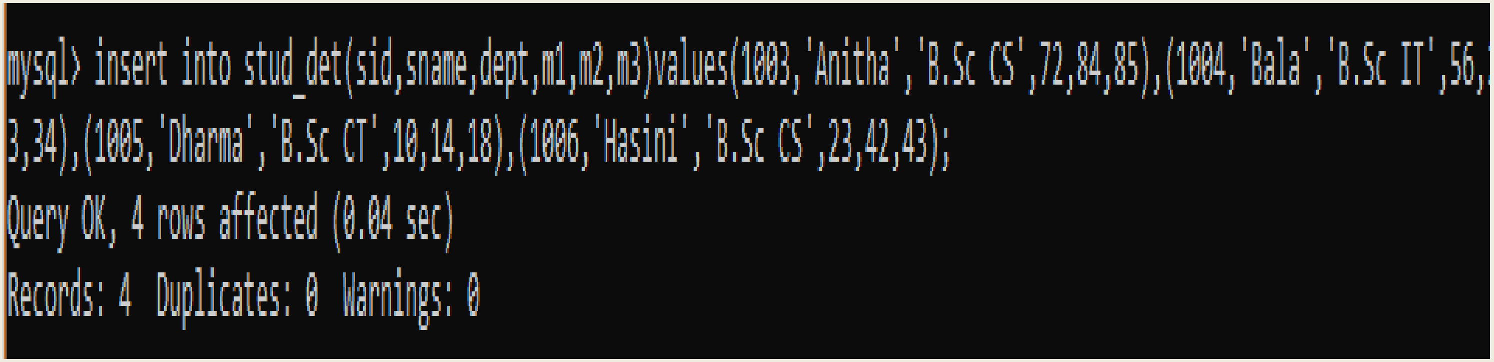
INSERT INTO table_name VALUES

(value1, value2,...valueN) ,

(value1, value2,...valueN),

....

(value1, value2,...valueN);



```
mysql> insert into stud_det(sid,sname,dept,m1,m2,m3)values(1003,'Anitha','B.Sc CS',72,84,85),(1004,'Bala','B.Sc IT',56,
3,34),(1005,'Dharma','B.Sc CT',10,14,18),(1006,'Hasini','B.Sc CS',23,42,43);
Query OK, 4 rows affected (0.04 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

❑ SELECT Statement

SYNTAX:

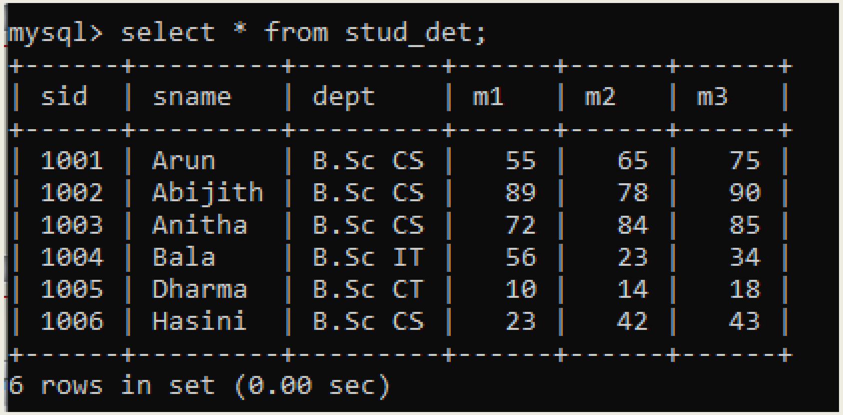
SELECT field_name1, field_name 2,... field_name
FROM table_name1, table_name2...
[**WHERE** condition]
[**GROUP BY** field_name(s)]
[**HAVING** condition]
[**ORDER BY** field_name(s)] ;

Selecting All Columns

SELECT *FROM table_name;

Selecting Specific Rows

SELECT * FROM table_nameWHERE condition;

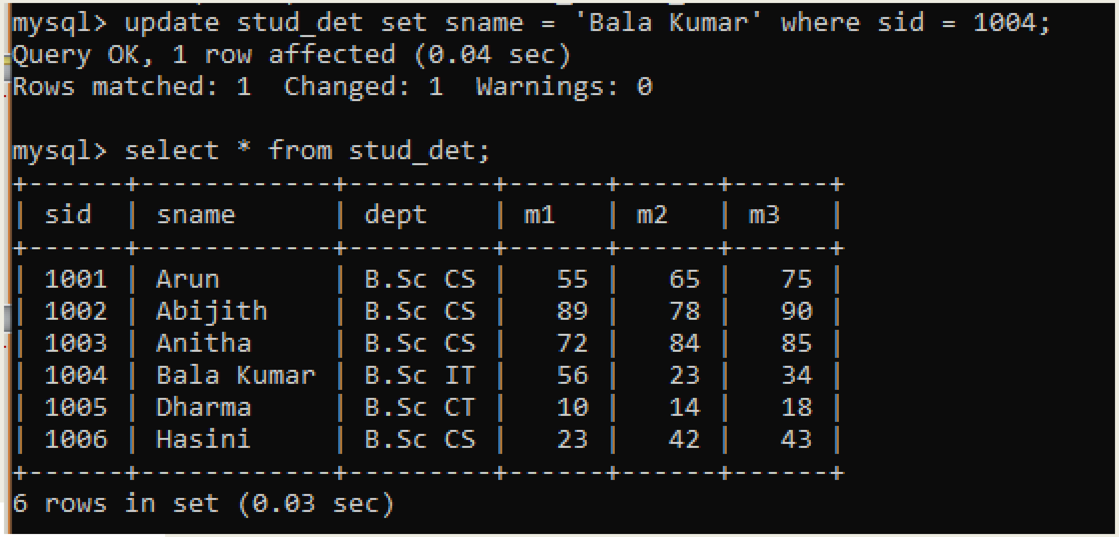


```
mysql> select * from stud_det;
+------+---------+---------+------+------+------+
| sid  | sname   | dept    | m1   | m2   | m3   |
+------+---------+---------+------+------+------+
| 1001 | Arun    | B.Sc CS |   55 |   65 |   75 |
| 1002 | Abijith | B.Sc CS |   89 |   78 |   90 |
| 1003 | Anitha  | B.Sc CS |   72 |   84 |   85 |
| 1004 | Bala    | B.Sc IT |   56 |   23 |   34 |
| 1005 | Dharma  | B.Sc CT |   10 |   14 |   18 |
| 1006 | Hasini  | B.Sc CS |   23 |   42 |   43 |
+------+---------+---------+------+------+------+
6 rows in set (0.00 sec)
```

❑ UPDATE QUERY

SYNTAX:

UPDATE table_name
SET column_name1 = new-value1,
        column_name2=new-value2, ...
[**WHERE** Clause]

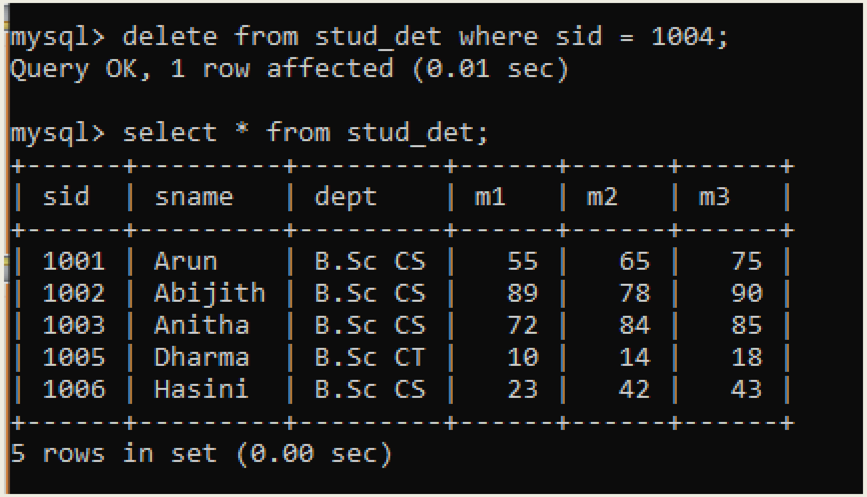


```
mysql> update stud_det set sname = 'Bala Kumar' where sid = 1004;
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from stud_det;
+------+------------+---------+------+------+------+
| sid  | sname      | dept    | m1   | m2   | m3   |
+------+------------+---------+------+------+------+
| 1001 | Arun       | B.Sc CS |   55 |   65 |   75 |
| 1002 | Abijith    | B.Sc CS |   89 |   78 |   90 |
| 1003 | Anitha     | B.Sc CS |   72 |   84 |   85 |
| 1004 | Bala Kumar | B.Sc IT |   56 |   23 |   34 |
| 1005 | Dharma     | B.Sc CT |   10 |   14 |   18 |
| 1006 | Hasini     | B.Sc CS |   23 |   42 |   43 |
+------+------------+---------+------+------+------+
6 rows in set (0.03 sec)
```

❑ DELETE QUERY

SYNTAX:

DELETE FROM table_name **WHERE** condition;



```
mysql> delete from stud_det where sid = 1004;
Query OK, 1 row affected (0.01 sec)

mysql> select * from stud_det;
+------+---------+---------+------+------+------+
| sid  | sname   | dept    | m1   | m2   | m3   |
+------+---------+---------+------+------+------+
| 1001 | Arun    | B.Sc CS |   55 |   65 |   75 |
| 1002 | Abijith | B.Sc CS |   89 |   78 |   90 |
| 1003 | Anitha  | B.Sc CS |   72 |   84 |   85 |
| 1005 | Dharma  | B.Sc CT |   10 |   14 |   18 |
| 1006 | Hasini  | B.Sc CS |   23 |   42 |   43 |
+------+---------+---------+------+------+------+
5 rows in set (0.00 sec)
```

SELECT QUERY - WHERE CLAUSE

- Comparison Operator
- LIKE OPERATOR
- LOGICAL OPERATOR
- BETWEEN AND IN
- ORDERBY and DISTINCT
- AGGREGATE FUNCTION
- GROUP BY AND HAVING CLAUSE
- OPERATORS
- SET OPERATIONS – UNION & UNION ALL

SQL WHERE Clause

Selecting Record Based on Condition

The WHERE clause is used with the SELECT, UPDATE, and DELETE.

Syntax

The WHERE clause is used with the SELECT statement to extract only those records that fulfill specified conditions. The basic syntax can be given with:

SELECT *column_list* FROM *table_name* WHERE *condition*;

Here, *column_list* are the names of columns/fields like *name*, *age*, *country* etc. of a database table whose values you want to fetch.

However, if you want to fetch the values of all the columns available in a table, you can use the following syntax:

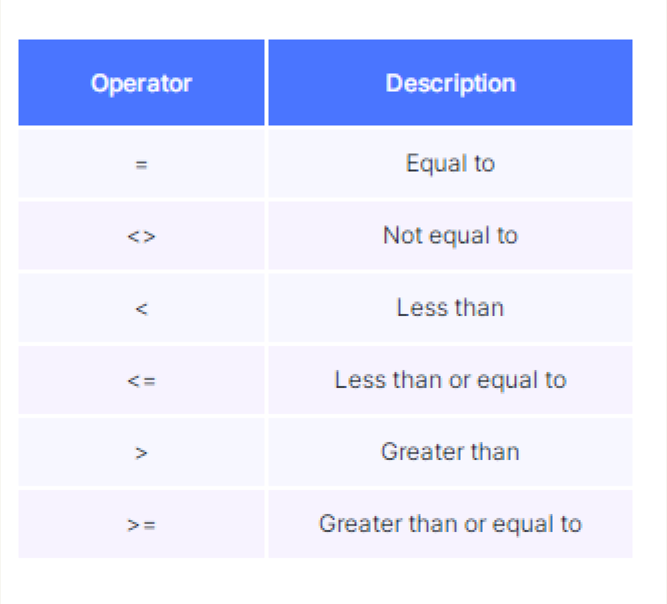
SELECT * FROM *table_name* WHERE *condition*;

Operators Allowed in WHERE Clause

SQL supports a number of different operators that can be used in `WHERE` clause, the most important ones are summarized in the following table.

Operator	Description	Example
=	Equal	WHERE id = 2
>	Greater than	WHERE age > 30
<	Less than	WHERE age < 18
>=	Greater than or equal	WHERE rating >= 4
<=	Less than or equal	WHERE price <= 100
LIKE	Simple pattern matching	WHERE name LIKE 'Dav'
IN	Check whether a specified value matches any value in a list or subquery	WHERE country IN ('USA', 'UK')
BETWEEN	Check whether a specified value is within a range of values	WHERE rating BETWEEN 3 AND 5

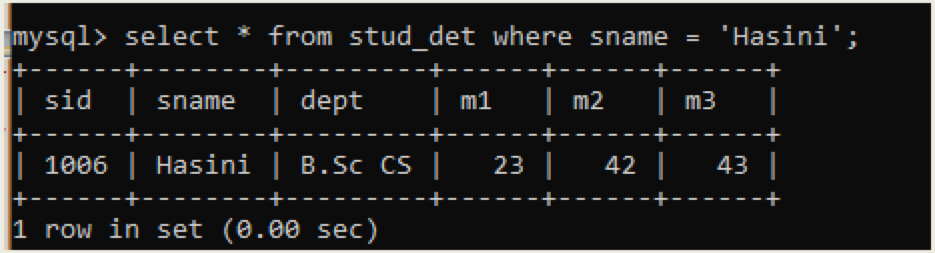
❑ COMPARISON OPERATOR:



Operator	Description
=	Equal to
<>	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

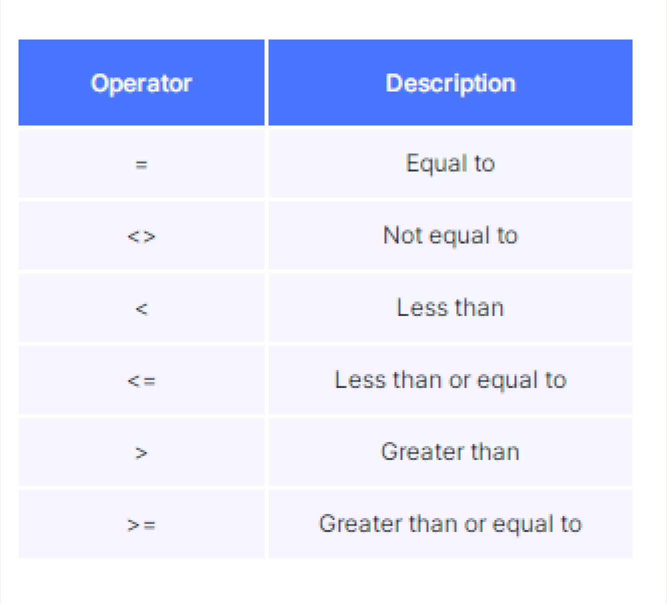
EXAMPLE:

SELECT *FROM stud_det WHERE sname = “Hasini";



```
mysql> select * from stud_det where sname = 'Hasini';
+------+--------+---------+------+------+------+
| sid  | sname  | dept    | m1   | m2   | m3   |
+------+--------+---------+------+------+------+
| 1006 | Hasini | B.Sc CS |   23 |   42 |   43 |
+------+--------+---------+------+------+------+
1 row in set (0.00 sec)
```

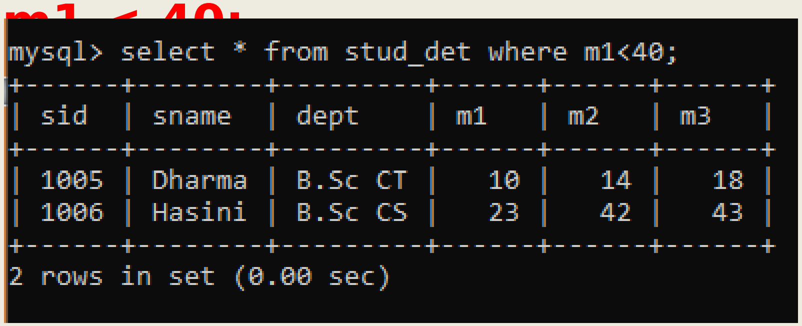
❑ COMPARISON OPERATOR:



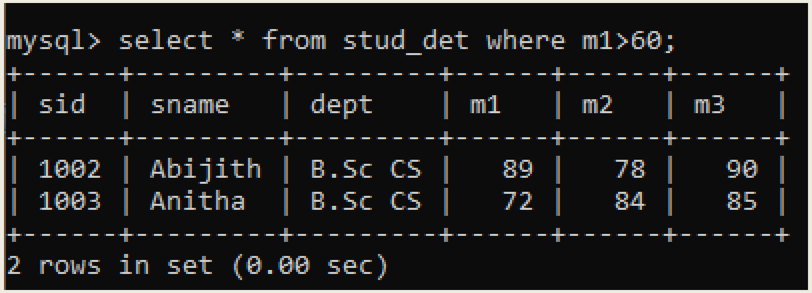
Operator	Description
=	Equal to
<>	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

EXAMPLE:

SELECT *FROM product WHERE m1 < 40;

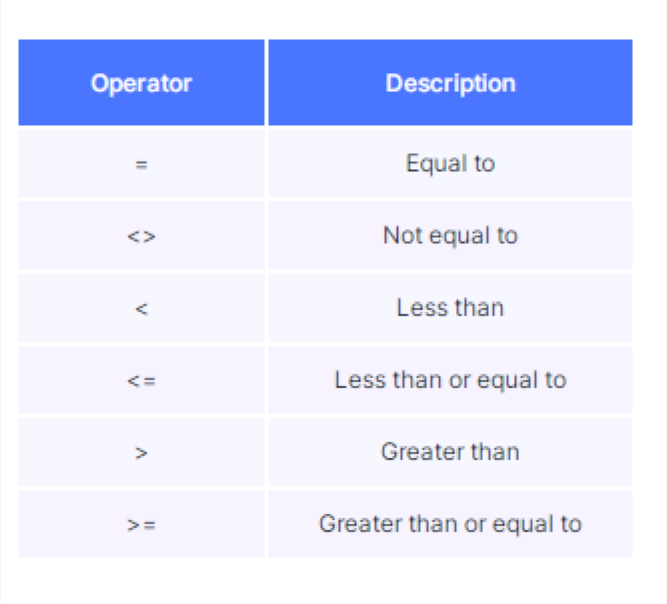


```
mysql> select * from stud_det where m1<40;
+------+--------+----------+------+------+------+
| sid  | sname  | dept     | m1   | m2   | m3   |
+------+--------+----------+------+------+------+
| 1005 | Dharma | B.Sc CT  |   10 |   14 |   18 |
| 1006 | Hasini | B.Sc CS  |   23 |   42 |   43 |
+------+--------+----------+------+------+------+
2 rows in set (0.00 sec)
```

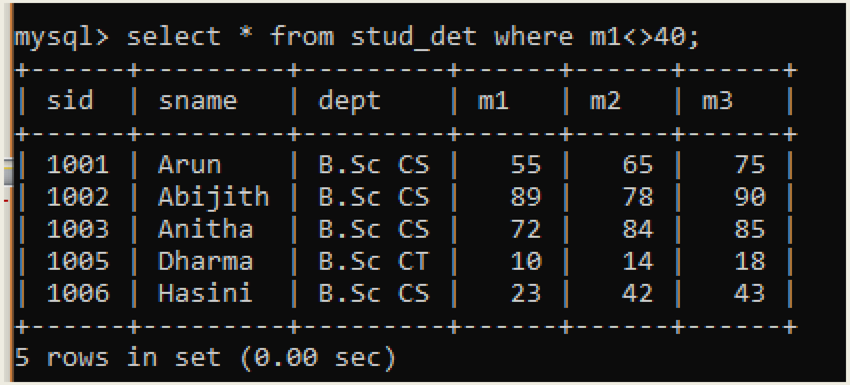


```
mysql> select * from stud_det where m1>60;
+------+----------+----------+------+------+------+
| sid  | sname    | dept     | m1   | m2   | m3   |
+------+----------+----------+------+------+------+
| 1002 | Abijith  | B.Sc CS  |   89 |   78 |   90 |
| 1003 | Anitha   | B.Sc CS  |   72 |   84 |   85 |
+------+----------+----------+------+------+------+
2 rows in set (0.00 sec)
```

❑ COMPARISON OPERATOR:



Operator	Description
=	Equal to
<>	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to



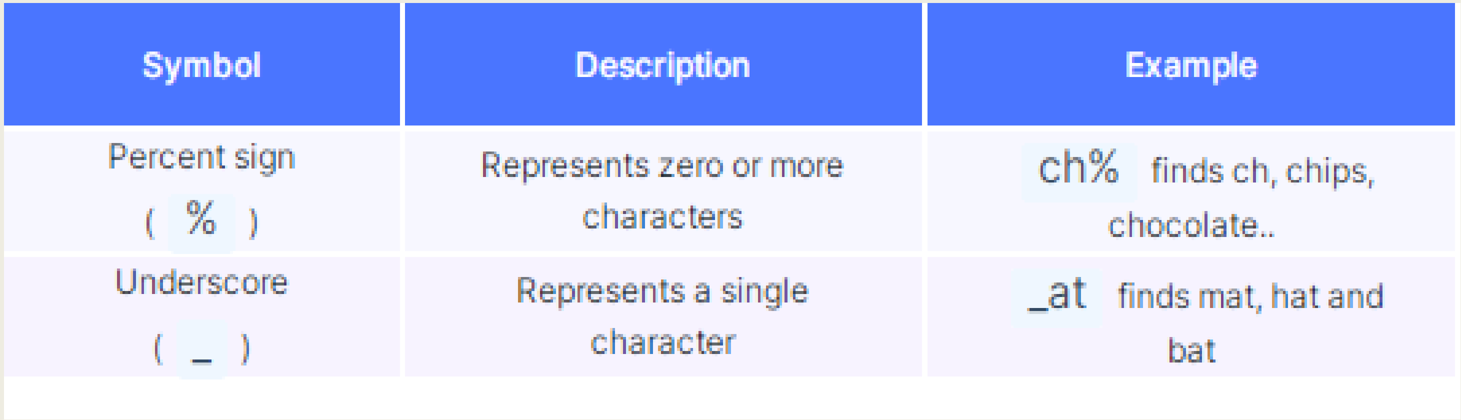
```
mysql> select * from stud_det where m1<>40;
+------+---------+---------+------+------+------+
| sid  | sname   | dept    | m1   | m2   | m3   |
+------+---------+---------+------+------+------+
| 1001 | Arun    | B.Sc CS |   55 |   65 |   75 |
| 1002 | Abijith | B.Sc CS |   89 |   78 |   90 |
| 1003 | Anitha  | B.Sc CS |   72 |   84 |   85 |
| 1005 | Dharma  | B.Sc CT |   10 |   14 |   18 |
| 1006 | Hasini  | B.Sc CS |   23 |   42 |   43 |
+------+---------+---------+------+------+------+
5 rows in set (0.00 sec)
```

❑ LIKE OPERATOR

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

LIKE Operator

LIKE operator is used to perform queries on strings. This operator is especially used in WHERE clause to retrieve all the rows that match the given pattern.



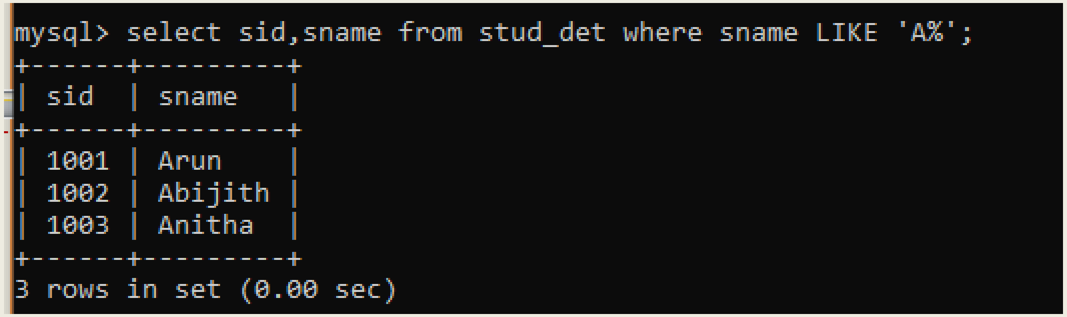
Symbol	Description	Example
Percent sign (%)	Represents zero or more characters	ch% finds ch, chips, chocolate..
Underscore (_)	Represents a single character	_at finds mat, hat and bat

❑ LIKE OPERATOR

❑Consider the case of e-commerce platforms. We generally search for the products on the basis of product name. But while searching, we need not enter the full name. For example, typing “mobiles” in a search bar will fetch thousands of results.

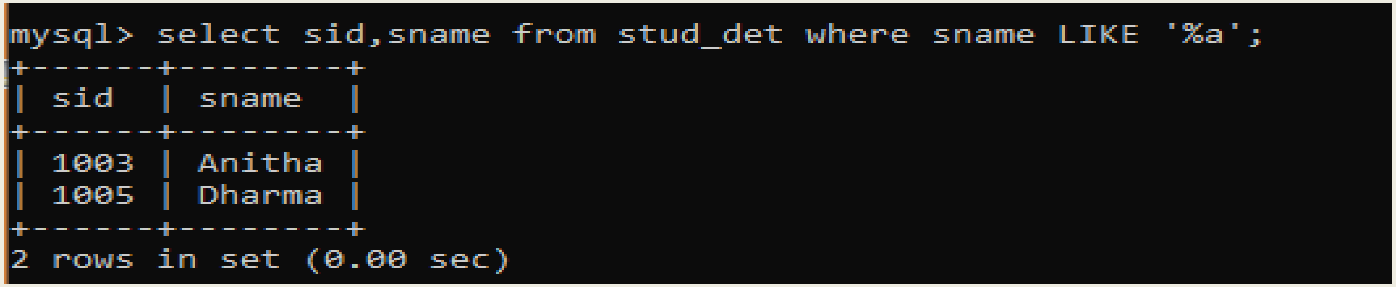
❑Syntax:

SELECT columnname1,columnname2 FROM Tablename WHERE columnname LIKE PATTERN;

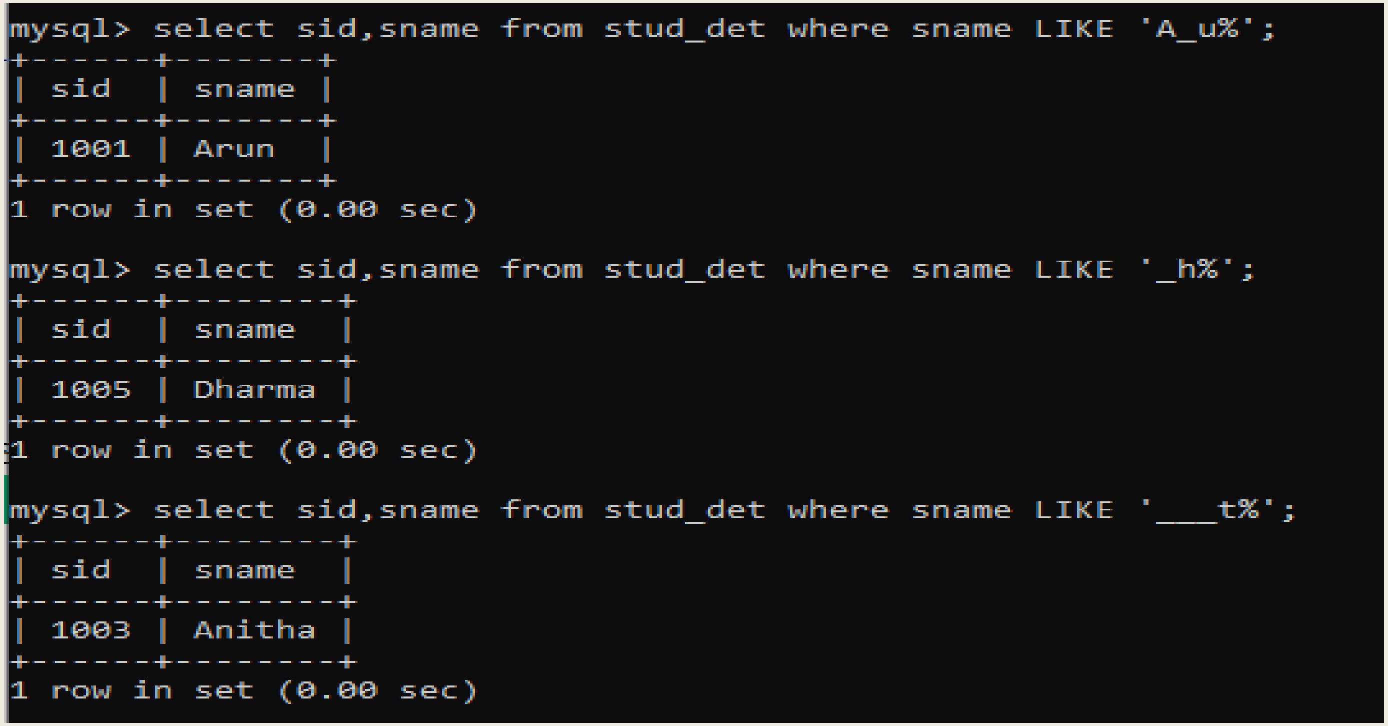


```
mysql> select sid,sname from stud_det where sname LIKE 'A%';
+------+---------+
| sid  | sname   |
+------+---------+
| 1001 | Arun    |
| 1002 | Abijith |
| 1003 | Anitha  |
+------+---------+
3 rows in set (0.00 sec)
```

❑ LIKE OPERATOR

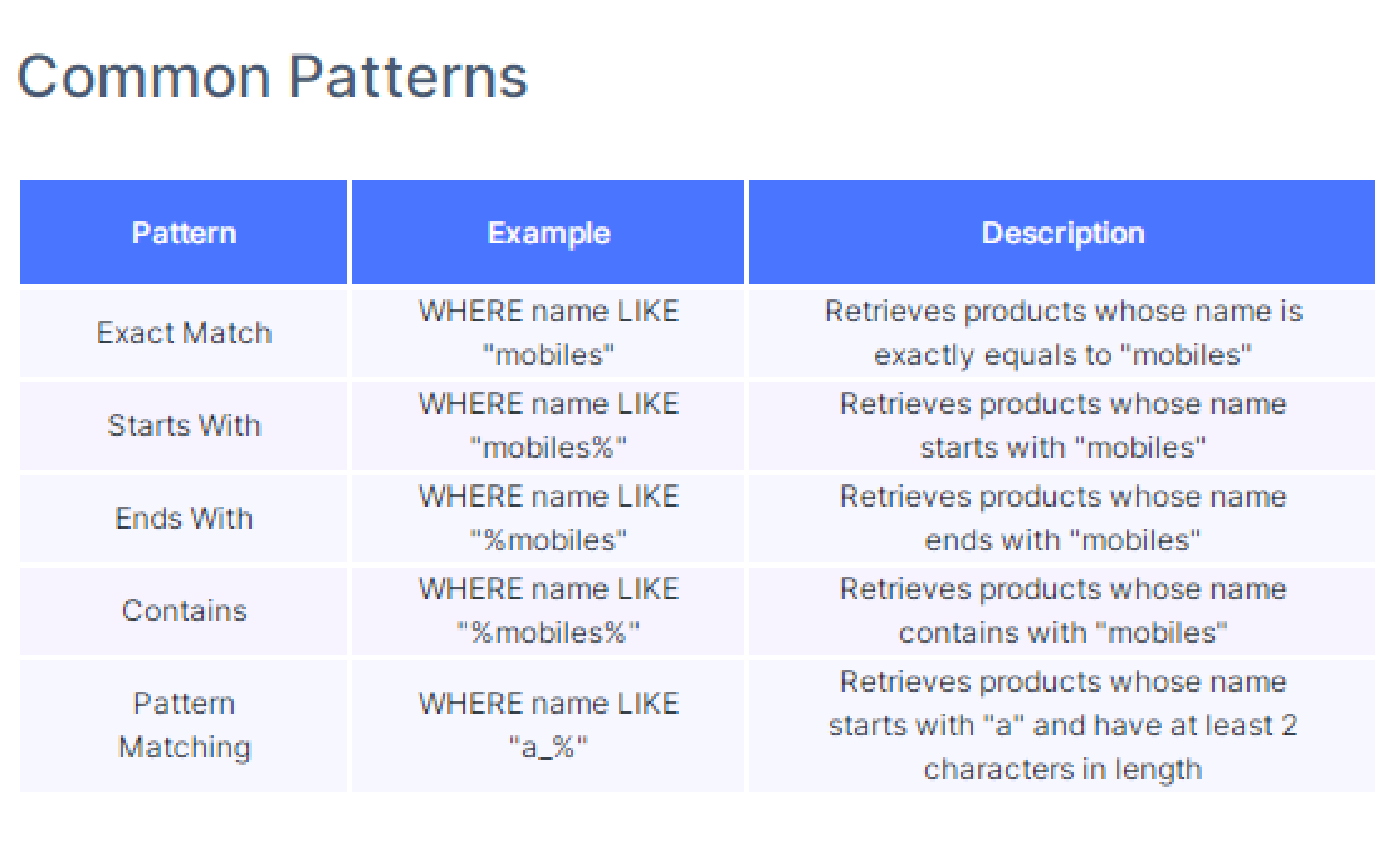


```
mysql> select sid,sname from stud_det where sname LIKE '%a';
+------+---------+
| sid  | sname   |
+------+---------+
| 1003 | Anitha  |
| 1005 | Dharma  |
+------+---------+
2 rows in set (0.00 sec)
```



```
mysql> select sid,sname from stud_det where sname LIKE 'A_u%';
+------+--------+
| sid  | sname  |
+------+--------+
| 1001 | Arun   |
+------+--------+
1 row in set (0.00 sec)

mysql> select sid,sname from stud_det where sname LIKE '_h%';
+------+---------+
| sid  | sname   |
+------+---------+
| 1005 | Dharma  |
+------+---------+
1 row in set (0.00 sec)

mysql> select sid,sname from stud_det where sname LIKE '___t%';
+------+---------+
| sid  | sname   |
+------+---------+
| 1003 | Anitha  |
+------+---------+
1 row in set (0.00 sec)
```

❑ String Operations



Common Patterns

Pattern	Example	Description
Exact Match	WHERE name LIKE "mobiles"	Retrieves products whose name is exactly equals to "mobiles"
Starts With	WHERE name LIKE "mobiles%"	Retrieves products whose name starts with "mobiles"
Ends With	WHERE name LIKE "%mobiles"	Retrieves products whose name ends with "mobiles"
Contains	WHERE name LIKE "%mobiles%"	Retrieves products whose name contains with "mobiles"
Pattern Matching	WHERE name LIKE "a_%"	Retrieves products whose name starts with "a" and have at least 2 characters in length

❑ String Operations

Syntax

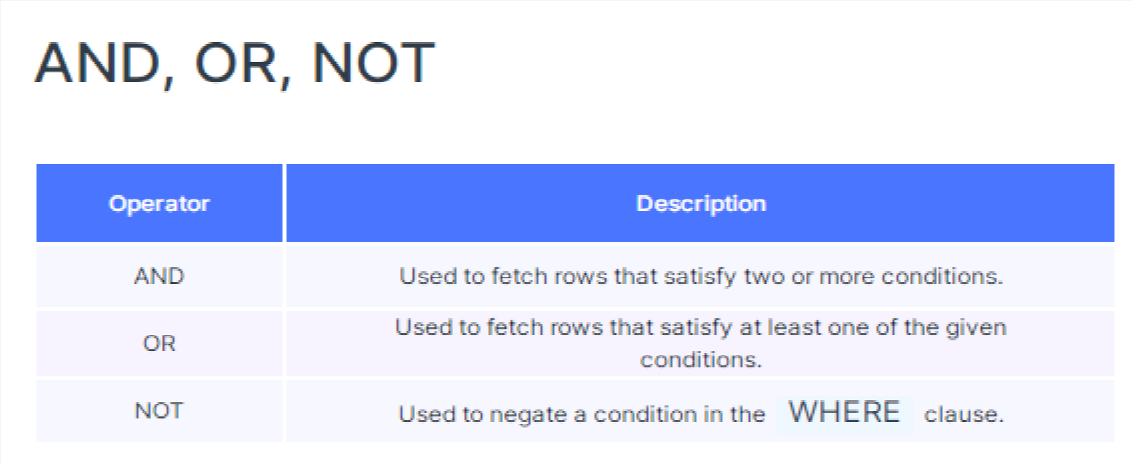
```
SELECT * FROM table_name WHERE c1 LIKE
matching_pattern;
```

EXAMPLE:

```
SELECT  *FROM  product WHERE  category LIKE
"Gadgets";
SELECT  *FROM  product WHERE  name LIKE "Bourbon%";
SELECT  *FROM  product WHERE  name LIKE "%Smart%";
```

Logical Operators

- ❑ If you want to combine more than one condition, then you need to use the Logical Operators in MySQL.
- ❑ The Logical Operators are used to check for the truthiness of some conditions.
- ❑ Logical operators return a Boolean data type with a value of TRUE, FALSE, or UNKNOWN.
- ❑ But in real-world scenarios, we often have to retrieve the data using several conditions at once.



AND, OR, NOT

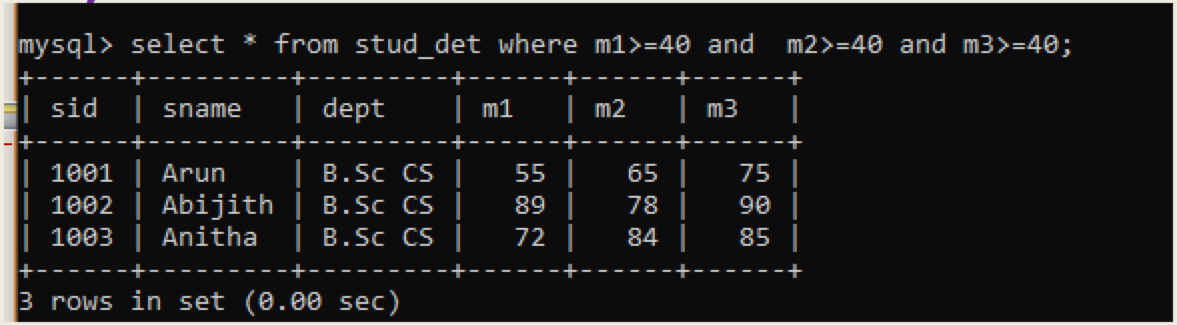
Operator	Description
AND	Used to fetch rows that satisfy two or more conditions.
OR	Used to fetch rows that satisfy at least one of the given conditions.
NOT	Used to negate a condition in the `WHERE` clause.

Logical Operators:

AND OPERATOR: The AND operator displays a record if all the conditions separated by AND are TRUE.

SYNTAX:

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



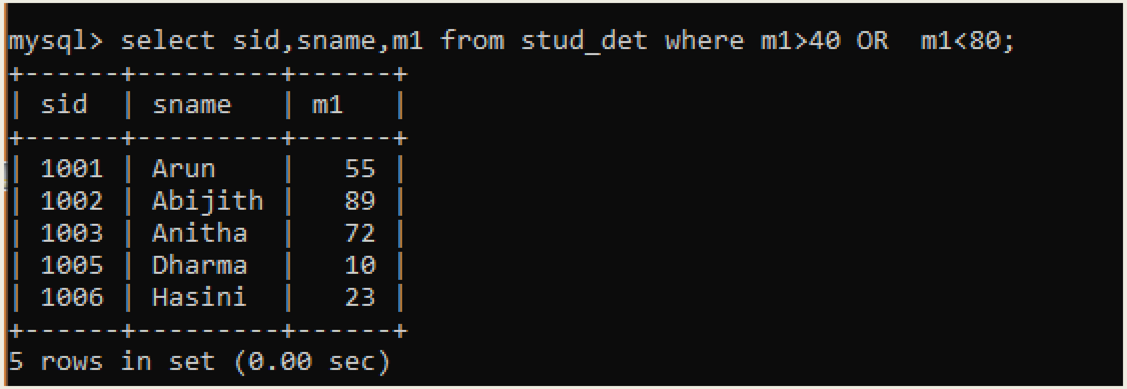
```
mysql> select * from stud_det where m1>=40 and  m2>=40 and m3>=40;
+------+---------+---------+------+------+------+
| sid  | sname   | dept    | m1   | m2   | m3   |
+------+---------+---------+------+------+------+
| 1001 | Arun    | B.Sc CS |   55 |   65 |   75 |
| 1002 | Abijith | B.Sc CS |   89 |   78 |   90 |
| 1003 | Anitha  | B.Sc CS |   72 |   84 |   85 |
+------+---------+---------+------+------+------+
3 rows in set (0.00 sec)
```

Logical Operators:

OR OPERATOR: The OR operator displays a record if any of the conditions separated by OR is TRUE.

SYNTAX:

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



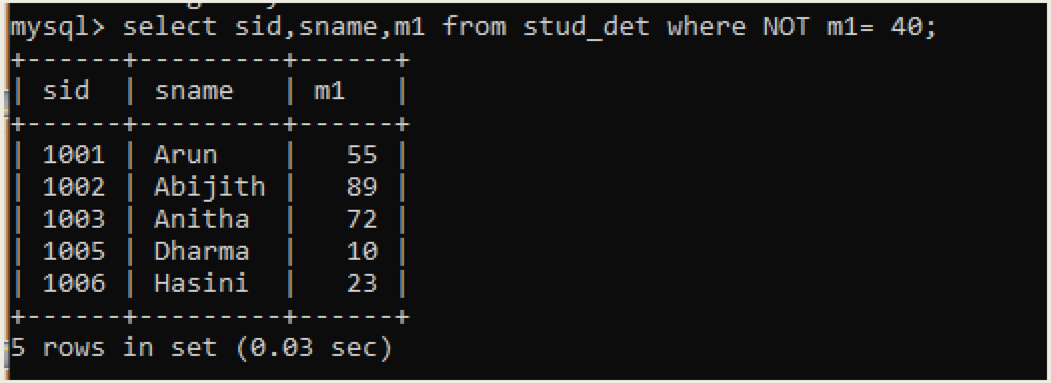
```
mysql> select sid,sname,m1 from stud_det where m1>40 OR  m1<80;
+------+---------+------+
| sid  | sname   | m1   |
+------+---------+------+
| 1001 | Arun    |   55 |
| 1002 | Abijith |   89 |
| 1003 | Anitha  |   72 |
| 1005 | Dharma  |   10 |
| 1006 | Hasini  |   23 |
+------+---------+------+
5 rows in set (0.00 sec)
```

Logical Operators:

NOT OPERATOR: The NOT operator displays a record if the condition(s) is NOT TRUE.

SYNTAX:

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



```
mysql> select sid,sname,m1 from stud_det where NOT m1= 40;
+------+---------+------+
| sid  | sname   | m1   |
+------+---------+------+
| 1001 | Arun    |   55 |
| 1002 | Abijith |   89 |
| 1003 | Anitha  |   72 |
| 1005 | Dharma  |   10 |
| 1006 | Hasini  |   23 |
+------+---------+------+
5 rows in set (0.03 sec)
```

❑ String Operations

Syntax

```
SELECT  *FROM  table_name WHERE  condition1
operator condition2  operator condition3  ...;
```

EXAMPLE:

```
SELECT  *FROM  product WHERE  category = "Clothing"
AND price <= 1000;

SELECT    *FROM    product WHERE    (brand = "Redmi"
AND rating > 4)    OR brand = "OnePlus";
```

IN Operators

We use the IN operator to check if a value is present in the list of values.

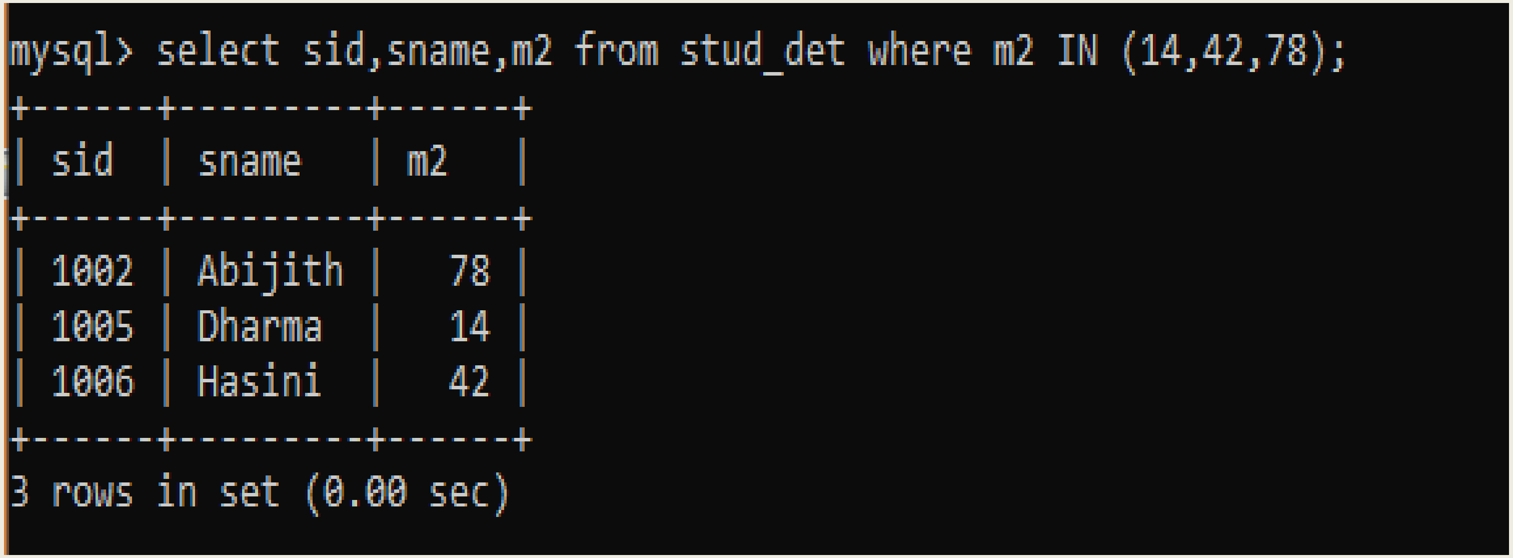
The IN operator allows you to specify multiple values in a WHERE clause.

The IN operator is a shorthand for multiple OR conditions.

IN Syntax

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

IN Operators



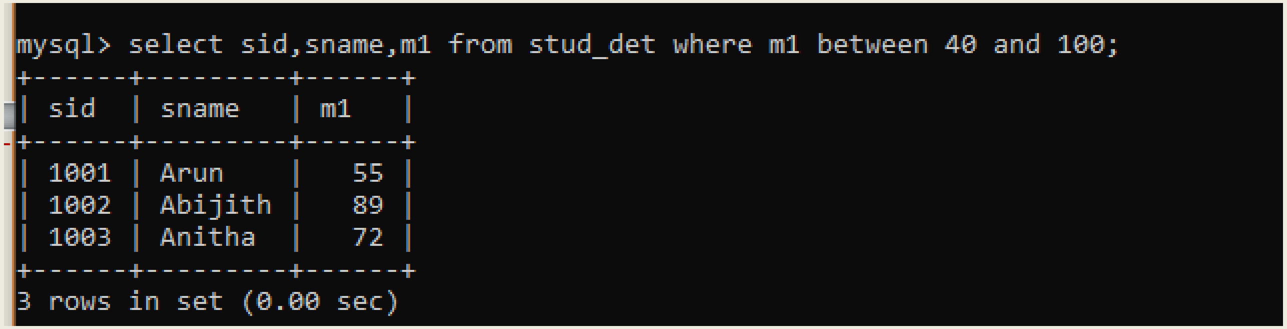
```
mysql> select sid,sname,m2 from stud_det where m2 IN (14,42,78);
+------+---------+------+
| sid  | sname   | m2   |
+------+---------+------+
| 1002 | Abijith |   78 |
| 1005 | Dharma  |   14 |
| 1006 | Hasini  |   42 |
+------+---------+------+
3 rows in set (0.00 sec)
```

BETWEEN Operators

- 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.

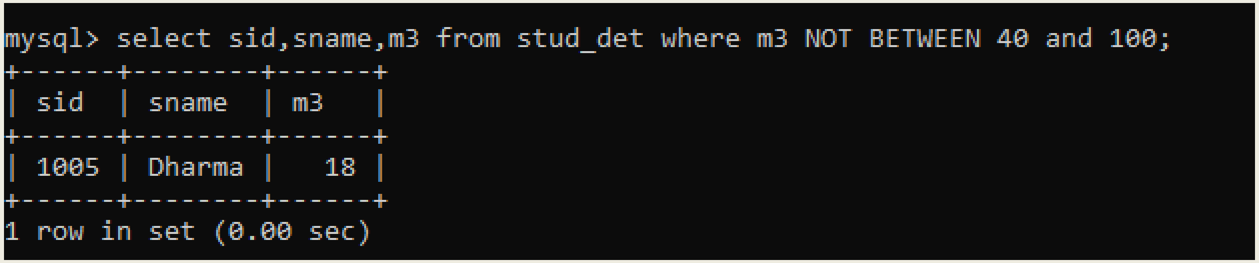
- Syntax:
- **SELECT *column_name(s)***
  FROM *table_name*
  WHERE *column_name* BETWEEN *value1* AND *value2;*

BETWEEN Operators



```
mysql> select sid,sname,m1 from stud_det where m1 between 40 and 100;
+------+----------+------+
| sid  | sname    | m1   |
+------+----------+------+
| 1001 | Arun     |   55 |
| 1002 | Abijith  |   89 |
| 1003 | Anitha   |   72 |
+------+----------+------+
3 rows in set (0.00 sec)
```

NOT BETWEEN Operators



```
mysql> select sid,sname,m3 from stud_det where m3 NOT BETWEEN 40 and 100;
+------+--------+------+
| sid  | sname  | m3   |
+------+--------+------+
| 1005 | Dharma |   18 |
+------+--------+------+
1 row in set (0.00 sec)
```

IN and BETWEEN Operators

Consider the case of a typical e-commerce scenario. Users generally search for the products that belong to a list of brands, or the products that lie within a particular price range.

In such scenarios, we use the IN operator to check if a value is present in the list of values. And, BETWEEN operator is used to check if a particular value exists in the given range.

❑ IN OPERATOR

Syntax

```
SELECT  *FROM  table_nameWHERE  c1 IN (v1, v2,..);
```

EXAMPLE:

```
SELECT  *FROM  product WHERE  brand IN ( "Puma", "Levi's", "Mufti", "Lee", "Denim");
```

❑ BETWEEN OPERATOR

Syntax:

```
SELECT  *FROM  table_name WHERE  c1
BETWEEN v1  AND v2;
```

EXAMPLE:

```
SELECT  name,  price,  brand FROM  product WHERE
 price BETWEEN 1000  AND 5000;
```

ORDER BY

- Generally when you use the SELECT statement to fetch data from a table, the rows in result set are not in any particular order.
- If you want your result set in a particular order, you can specify the ORDER BY clause at the end of the statement which tells the server how to sort the data returned by the query.

The default sorting order is ascending.

Syntax

The **ORDER BY clause is used to sort the data returned by a query in ascending or descending order**.

The basic syntax of this clause can be given with:

SELECT *column_list* FROM *table_name* ORDER B DESC;

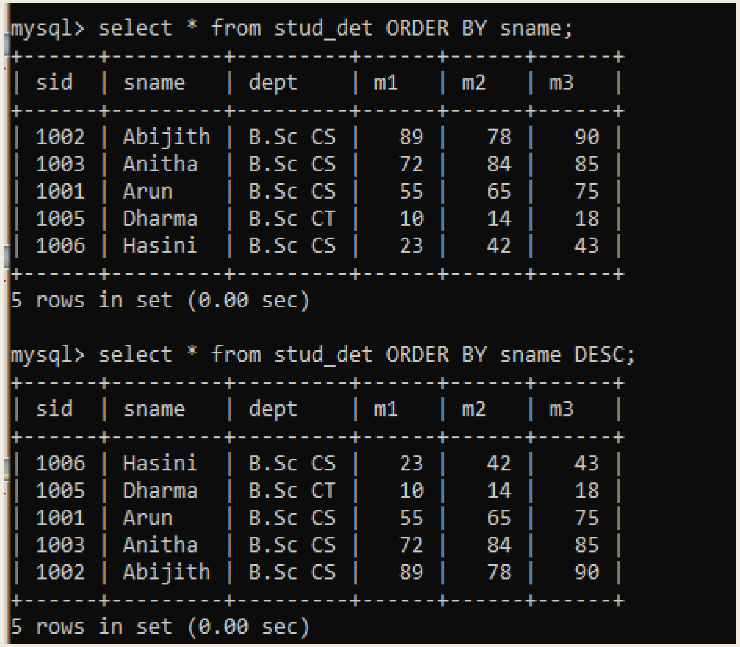
ORDER BY

- 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.

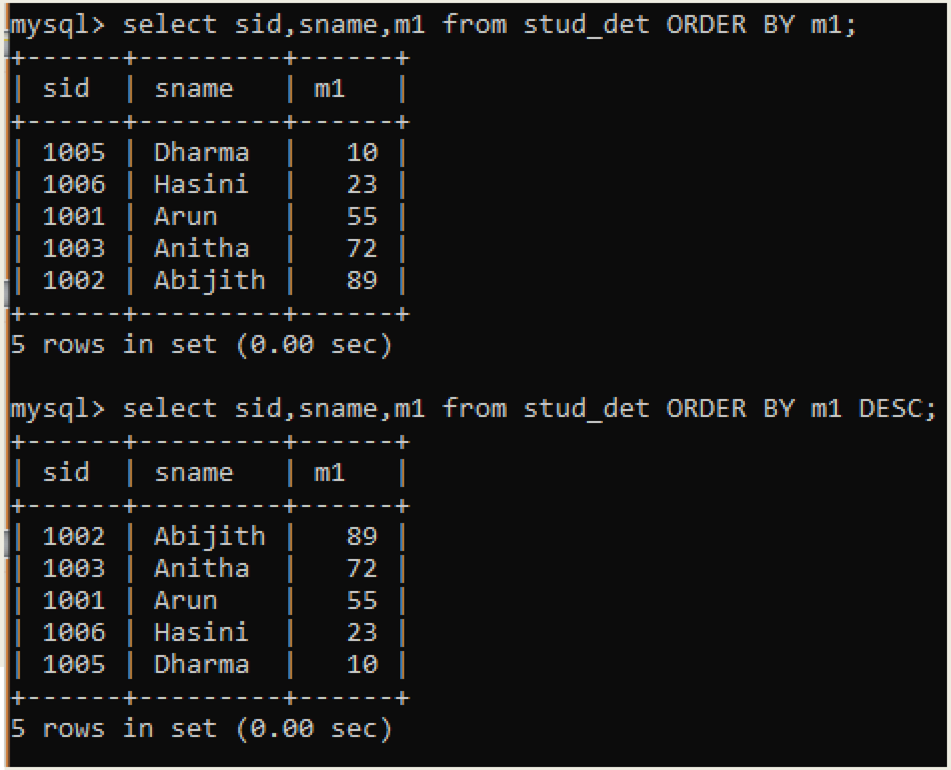
Syntax

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

ORDER BY



```
mysql> select * from stud_det ORDER BY sname;
+------+---------+---------+------+------+------+
| sid  | sname   | dept    | m1   | m2   | m3   |
+------+---------+---------+------+------+------+
| 1002 | Abijith | B.Sc CS |   89 |   78 |   90 |
| 1003 | Anitha  | B.Sc CS |   72 |   84 |   85 |
| 1001 | Arun    | B.Sc CS |   55 |   65 |   75 |
| 1005 | Dharma  | B.Sc CT |   10 |   14 |   18 |
| 1006 | Hasini  | B.Sc CS |   23 |   42 |   43 |
+------+---------+---------+------+------+------+
5 rows in set (0.00 sec)

mysql> select * from stud_det ORDER BY sname DESC;
+------+---------+---------+------+------+------+
| sid  | sname   | dept    | m1   | m2   | m3   |
+------+---------+---------+------+------+------+
| 1006 | Hasini  | B.Sc CS |   23 |   42 |   43 |
| 1005 | Dharma  | B.Sc CT |   10 |   14 |   18 |
| 1001 | Arun    | B.Sc CS |   55 |   65 |   75 |
| 1003 | Anitha  | B.Sc CS |   72 |   84 |   85 |
| 1002 | Abijith | B.Sc CS |   89 |   78 |   90 |
+------+---------+---------+------+------+------+
5 rows in set (0.00 sec)
```



```
mysql> select sid,sname,m1 from stud_det ORDER BY m1;
+------+---------+------+
| sid  | sname   | m1   |
+------+---------+------+
| 1005 | Dharma  |   10 |
| 1006 | Hasini  |   23 |
| 1001 | Arun    |   55 |
| 1003 | Anitha  |   72 |
| 1002 | Abijith |   89 |
+------+---------+------+
5 rows in set (0.00 sec)

mysql> select sid,sname,m1 from stud_det ORDER BY m1 DESC;
+------+---------+------+
| sid  | sname   | m1   |
+------+---------+------+
| 1002 | Abijith |   89 |
| 1003 | Anitha  |   72 |
| 1001 | Arun    |   55 |
| 1006 | Hasini  |   23 |
| 1005 | Dharma  |   10 |
+------+---------+------+
5 rows in set (0.00 sec)
```

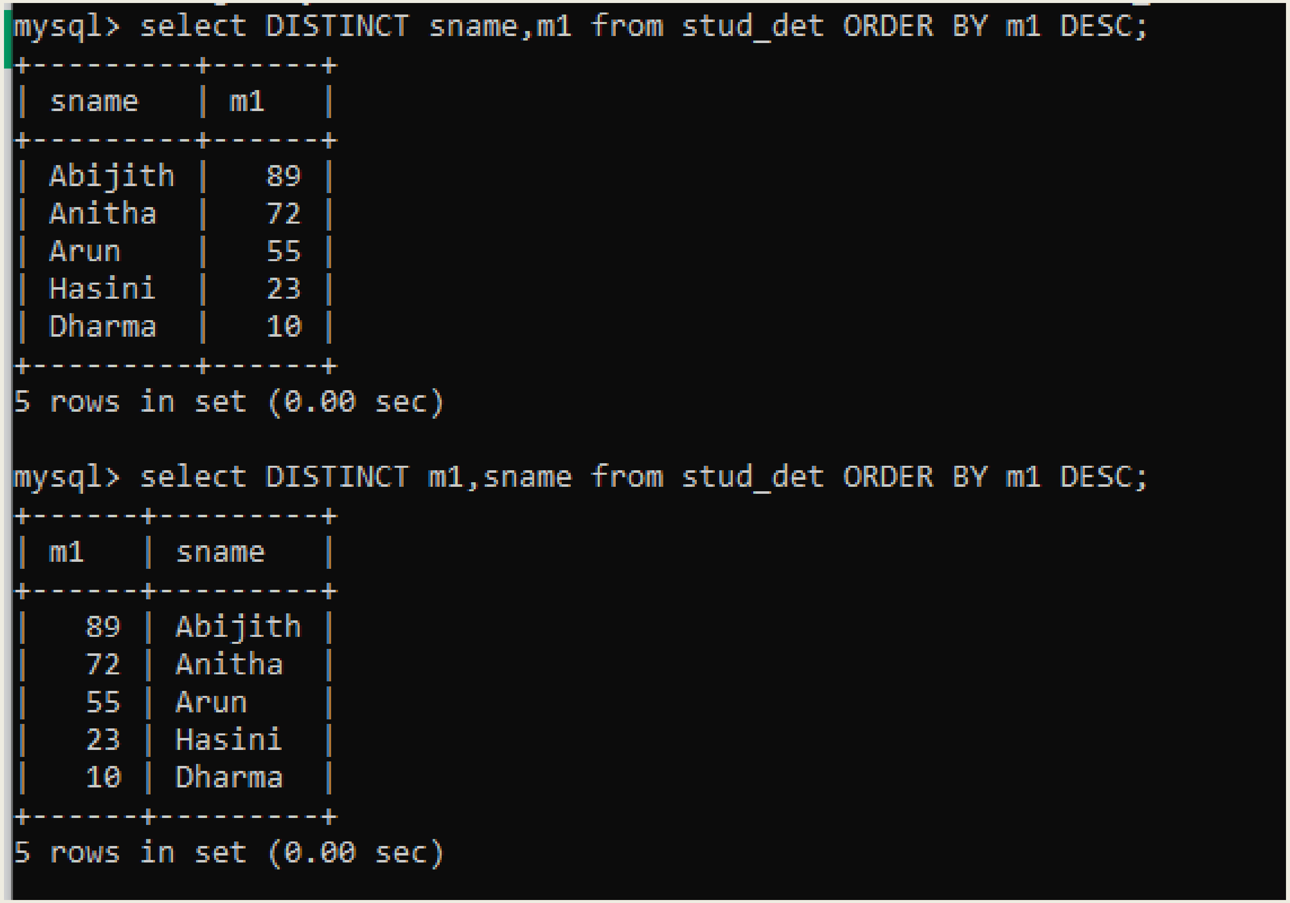
DISTINCT

- 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*;

DISTINCT

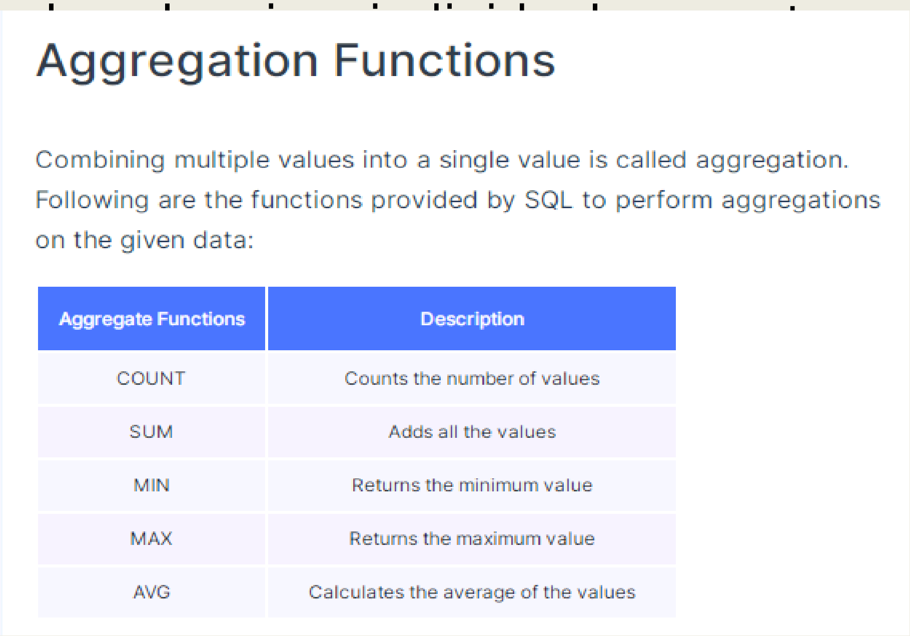


```
mysql> select DISTINCT sname,m1 from stud_det ORDER BY m1 DESC;
+---------+------+
| sname   | m1   |
+---------+------+
| Abijith |   89 |
| Anitha  |   72 |
| Arun    |   55 |
| Hasini  |   23 |
| Dharma  |   10 |
+---------+------+
5 rows in set (0.00 sec)

mysql> select DISTINCT m1,sname from stud_det ORDER BY m1 DESC;
+------+---------+
| m1   | sname   |
+------+---------+
|   89 | Abijith |
|   72 | Anitha  |
|   55 | Arun    |
|   23 | Hasini  |
|   10 | Dharma  |
+------+---------+
5 rows in set (0.00 sec)
```

Aggregations:

❖Consider the case of sports tournaments like cricket. Players' performances are analysed based on their batting average, maximum number of sixes hit, the least score in a tournament, etc.

❖We perform aggregations in such scenarios to combine multiple values into a sir
average score.



Aggregation Functions

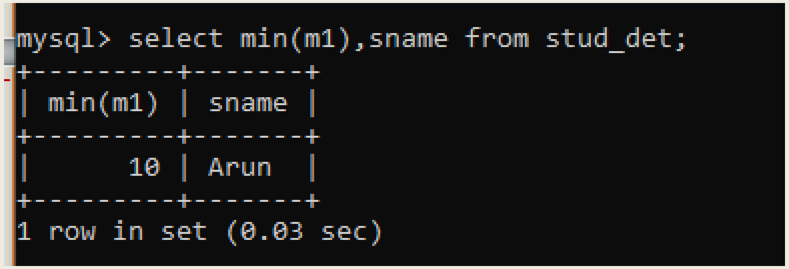
Combining multiple values into a single value is called aggregation. Following are the functions provided by SQL to perform aggregations on the given data:

Aggregate Functions	Description
COUNT	Counts the number of values
SUM	Adds all the values
MIN	Returns the minimum value
MAX	Returns the maximum value
AVG	Calculates the average of the values

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

Syntax:

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

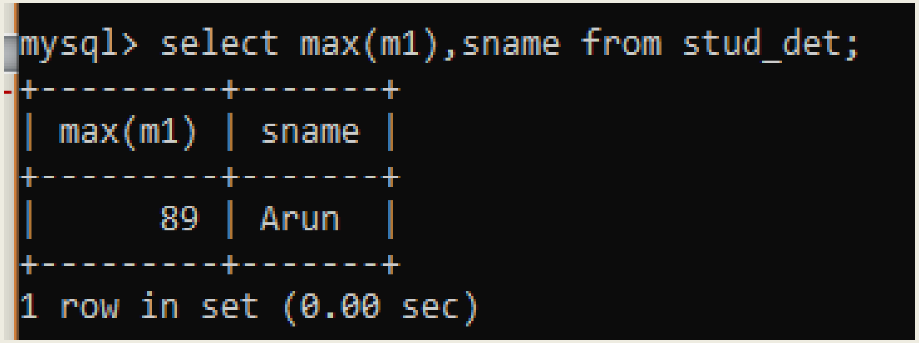


```
mysql> select min(m1),sname from stud_det;
+---------+-------+
| min(m1) | sname |
+---------+-------+
|      10 | Arun  |
+---------+-------+
1 row in set (0.03 sec)
```

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

Syntax:

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



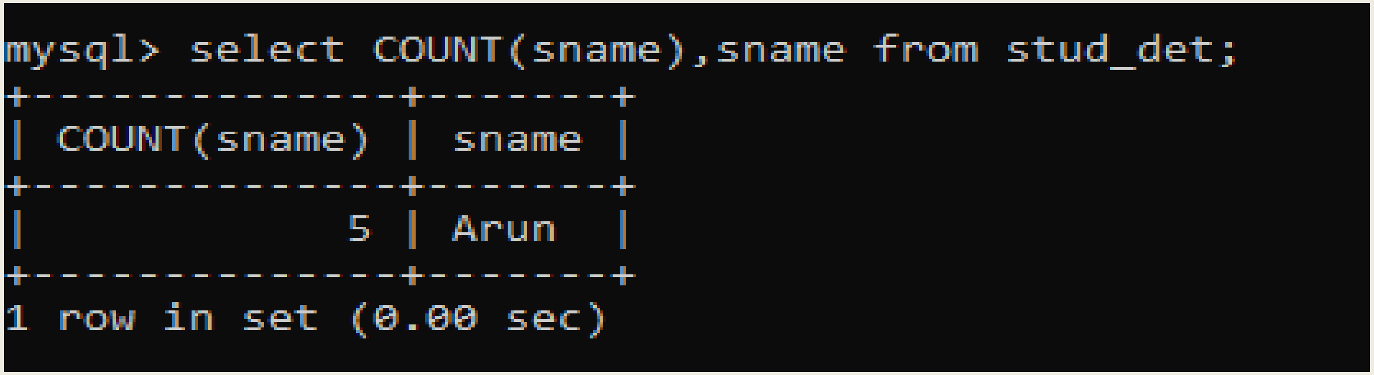
```
mysql> select max(m1),sname from stud_det;
+---------+-------+
| max(m1) | sname |
+---------+-------+
|      89 | Arun  |
+---------+-------+
1 row in set (0.00 sec)
```

COUNT():

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

SYNTAX:

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



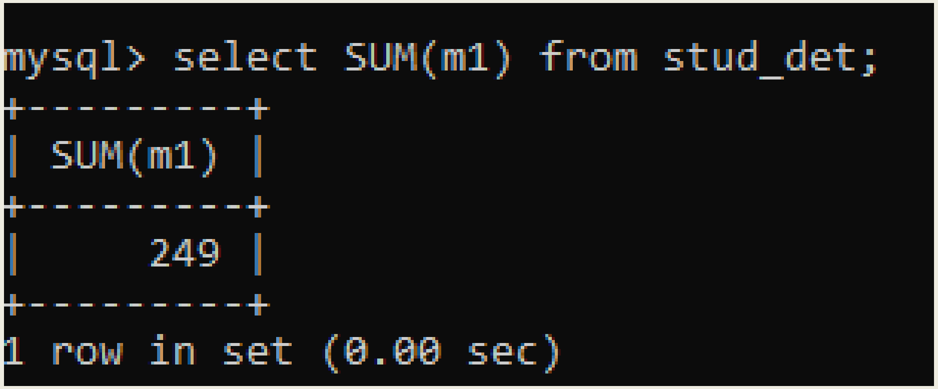
```
mysql> select COUNT(sname),sname from stud_det;
+--------------+-------+
| COUNT(sname) | sname |
+--------------+-------+
|            5 | Arun  |
+--------------+-------+
1 row in set (0.00 sec)
```

SUM():

The SUM() function returns the total sum of a numeric column. .

SYNTAX:

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



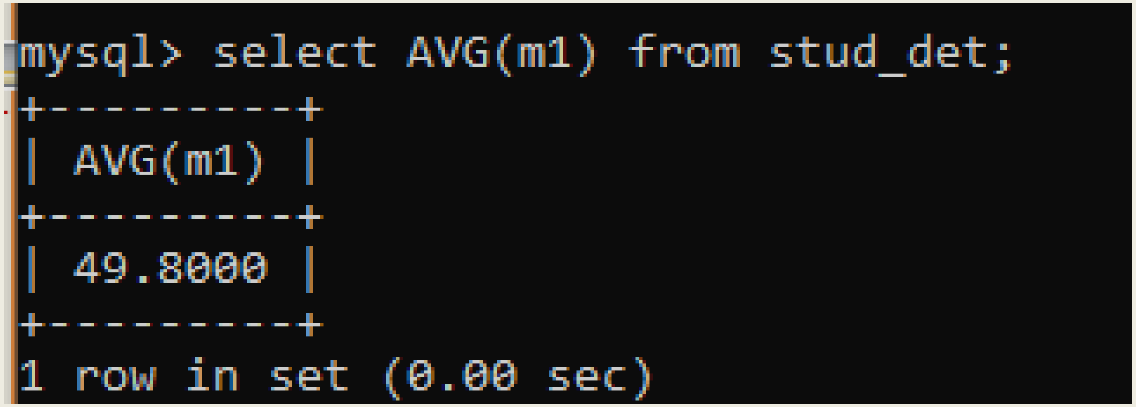
```
mysql> select SUM(m1) from stud_det;
+---------+
| SUM(m1) |
+---------+
|     249 |
+---------+
1 row in set (0.00 sec)
```

AVG():

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

SYNTAX:

SELECT AVG(*column_name*) FROM *table_name* WHERE *condition*;



```
mysql> select AVG(m1) from stud_det;
+----------+
| AVG(m1) |
+----------+
| 49.8000 |
+----------+
1 row in set (0.00 sec)
```

Syntax:

```
SELECT  aggregate_function(c1),
aggregate_function(c2) FROM  TABLE;
```

EXAMPLE:

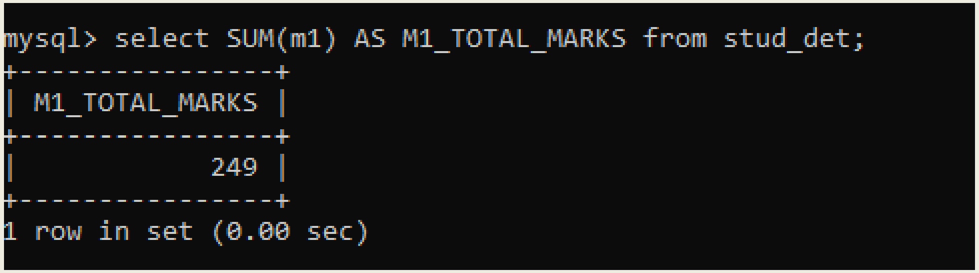
```
SELECT  SUM(score)FROM  player_match_detailsWHERE
name = "Ram";
ELECT  MAX(score),  MIN(score)FROM
player_match_detailsWHERE  year = 2011;
SELECT COUNT(*)     FROM player_match_details;
```

Alias

- 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;*



```
mysql> select SUM(m1) AS M1_TOTAL_MARKS from stud_det;
+-----------------+
| M1_TOTAL_MARKS |
+-----------------+
|             249 |
+-----------------+
1 row in set (0.00 sec)
```

Alias

Using the keyword AS , we can provide alternate temporary names to the columns in the output.

Syntax:

```
SELECT  c1 AS a1,  c2 AS a2,  ...FROM
table_name;
```

EXAMPLE:

```
SELECT  name AS player_name FROM
player_match_details;
SELECT  AVG(score) AS avg_score FROM
player_match_details;
```

GROUP BY

- The GROUP BY clause in SQL is used to group rows which have same values for the mentioned attributes.
- The MySQL GROUP BY Clause returns an aggregated data (value) by grouping one or more columns.
- It first groups the columns and then applies the aggregated functions on the remaining columns.
- To display the high-level or aggregated information, you have to use this MySQL Group by clause

Syntax:

```
SELECT c1, aggregate_function(c2)FROM table_name
GROUP BY c1;
```

MySQL GROUP BY

The syntax as:

SELECT [Column1],...[ColumnN], Aggregate Function(Column_Name) FROM [Source]
WHERE [Conditions] -- Optional GROUP BY [Column1],...[ColumnN]
ORDER BY Columns.

•**Column1…N:** Choose the columns from a table(s).

•**Aggregate Functions:** Use any of the aggregate functions. COUNT, SUM, AVG, AVG, MIN, MAX, STD, and VARIANCE are the functions that we can use.

•**Group By:** Columns that are not part of an Aggregate Function have to place after this.

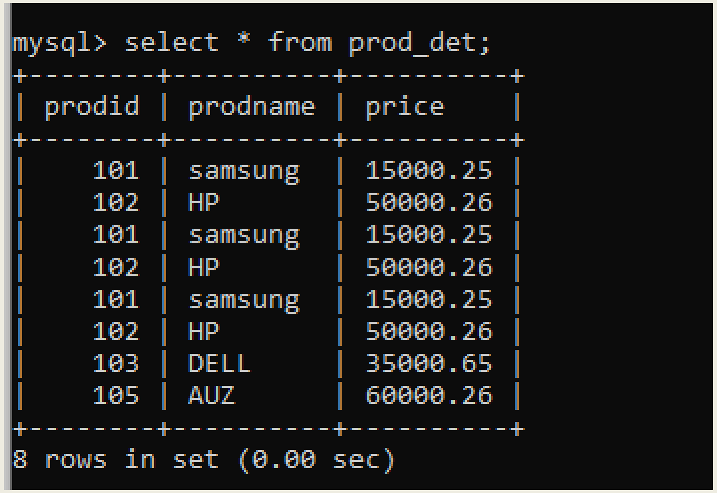
Group By:

- The GROUP BY statement groups rows that have the same values into summary rows, like "find the total marks of number of students in each subject".
- The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

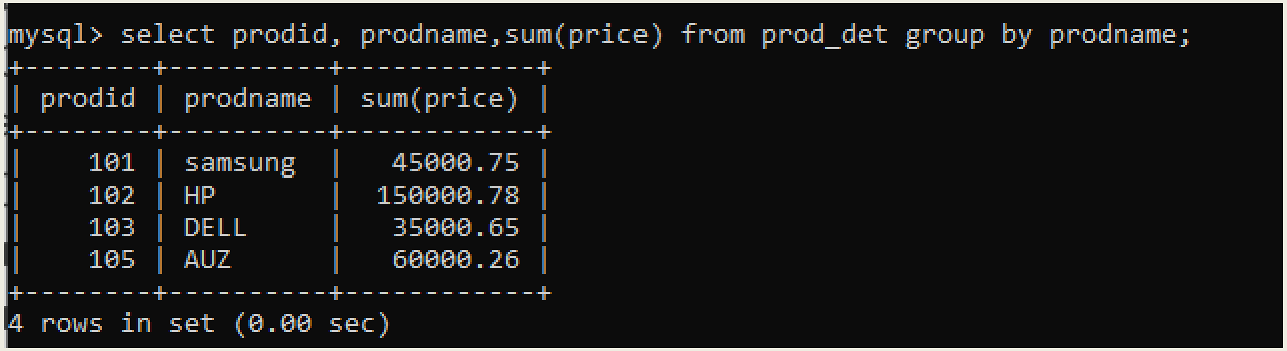
SYNTAX:

```
SELECT column_name(s) FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

MySQL GROUP BY CLAUSE

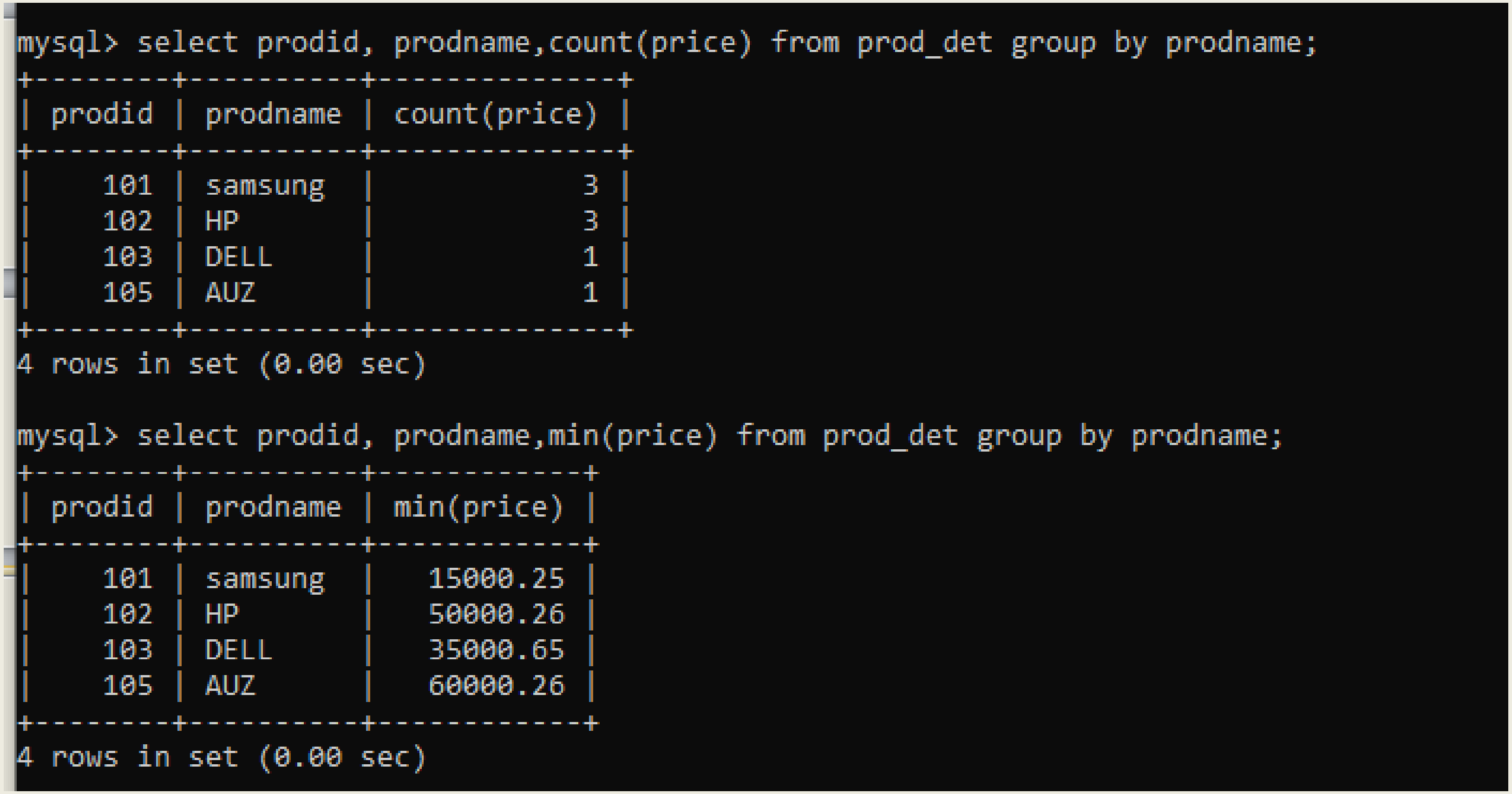


```
mysql> select * from prod_det;
+--------+----------+----------+
| prodid | prodname | price    |
+--------+----------+----------+
|    101 | samsung  | 15000.25 |
|    102 | HP       | 50000.26 |
|    101 | samsung  | 15000.25 |
|    102 | HP       | 50000.26 |
|    101 | samsung  | 15000.25 |
|    102 | HP       | 50000.26 |
|    103 | DELL     | 35000.65 |
|    105 | AUZ      | 60000.26 |
+--------+----------+----------+
8 rows in set (0.00 sec)
```



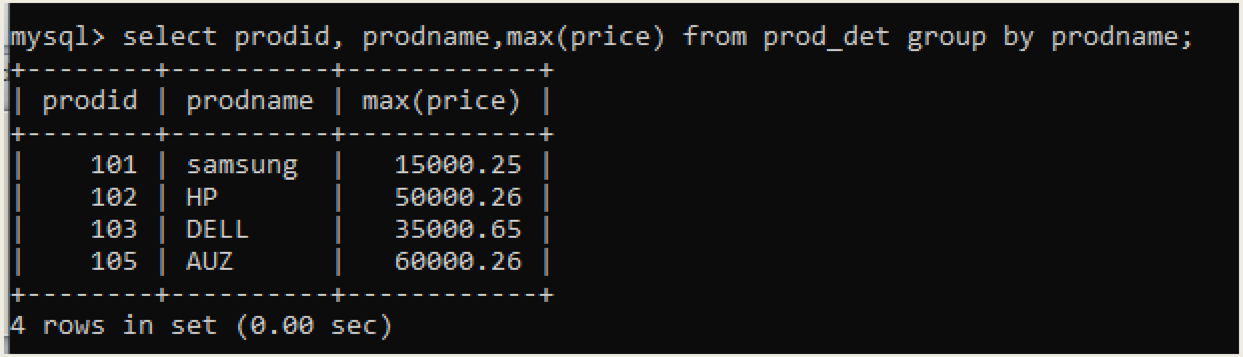
```
mysql> select prodid, prodname,sum(price) from prod_det group by prodname;
+--------+----------+------------+
| prodid | prodname | sum(price) |
+--------+----------+------------+
|    101 | samsung  |   45000.75 |
|    102 | HP       |  150000.78 |
|    103 | DELL     |   35000.65 |
|    105 | AUZ      |   60000.26 |
+--------+----------+------------+
4 rows in set (0.00 sec)
```

MySQL GROUP BY



```
mysql> select prodid, prodname,count(price) from prod_det group by prodname;
+--------+----------+--------------+
| prodid | prodname | count(price) |
+--------+----------+--------------+
|    101 | samsung  |            3 |
|    102 | HP       |            3 |
|    103 | DELL     |            1 |
|    105 | AUZ      |            1 |
+--------+----------+--------------+
4 rows in set (0.00 sec)

mysql> select prodid, prodname,min(price) from prod_det group by prodname;
+--------+----------+------------+
| prodid | prodname | min(price) |
+--------+----------+------------+
|    101 | samsung  |   15000.25 |
|    102 | HP       |   50000.26 |
|    103 | DELL     |   35000.65 |
|    105 | AUZ      |   60000.26 |
+--------+----------+------------+
4 rows in set (0.00 sec)
```

MySQL GROUP BY



```
mysql> select prodid, prodname,max(price) from prod_det group by prodname;
+--------+----------+------------+
| prodid | prodname | max(price) |
+--------+----------+------------+
|    101 | samsung  |   15000.25 |
|    102 | HP       |   50000.26 |
|    103 | DELL     |   35000.65 |
|    105 | AUZ      |   60000.26 |
+--------+----------+------------+
4 rows in set (0.00 sec)
```

HAVING

- HAVING clause is used to filter the resultant rows after the application of GROUP BY clause.
- The MySQL Having Clause restricts the number of records or rows returned by the Group By Clause.
- To use MySQL Having Clause, we have to use Group By.
- It is because the Having is applied after the Group by.

Syntax:

```
SELECT  c1,  c2,  aggregate_function(c1)FROM
table_nameGROUP BY   c1, c2HAVING   condition;
```

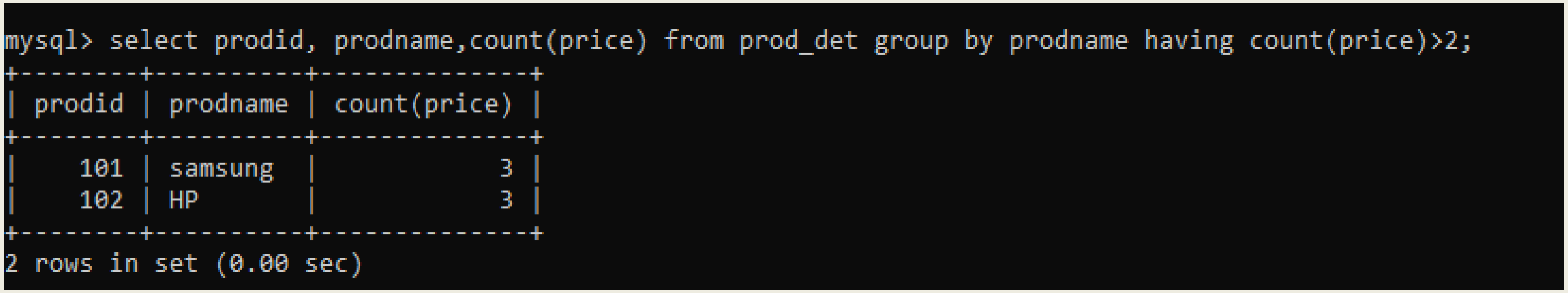
MySQL HAVING CLAUSE

Syntax:

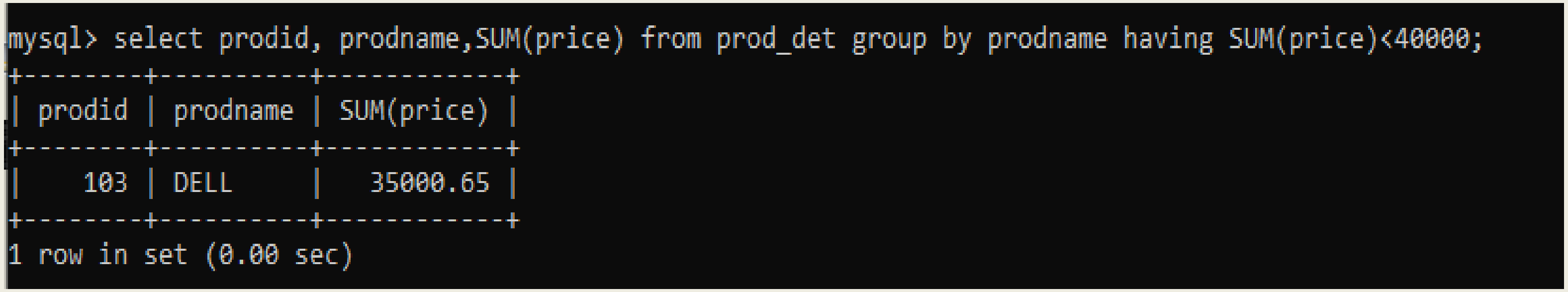
```
SELECT [Column1],...[ColumnN],
Aggregate_Function(Column_Name)
FROM [Source]
WHERE [Conditions] -- Optional
GROUP BY [Column1],...[ColumnN]
HAVING [Conditions] -- Condition is on Aggregate Function(Column_Name)
```

- **Column1…N:** Choose the columns from a table(s).
- **Aggregate Functions:** Use any of the aggregate functions. COUNT, SUM, AVG, AVG, MIN, MAX, STD, and VARIANCE are the functions that we can use.
- **Group By:** Columns that are not part of an Aggregate Function have to place after this Group by.
- **Having:** We can provide the Filters or apply Conditions on the Aggregated Data that we got from the Group By.

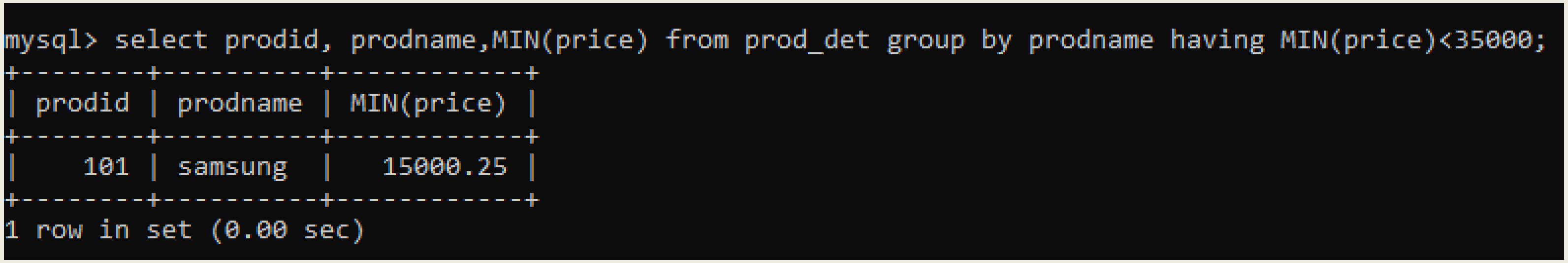
MySQL HAVING CLAUSE



```
mysql> select prodid, prodname,count(price) from prod_det group by prodname having count(price)>2;
+--------+----------+--------------+
| prodid | prodname | count(price) |
+--------+----------+--------------+
|    101 | samsung  |            3 |
|    102 | HP       |            3 |
+--------+----------+--------------+
2 rows in set (0.00 sec)
```



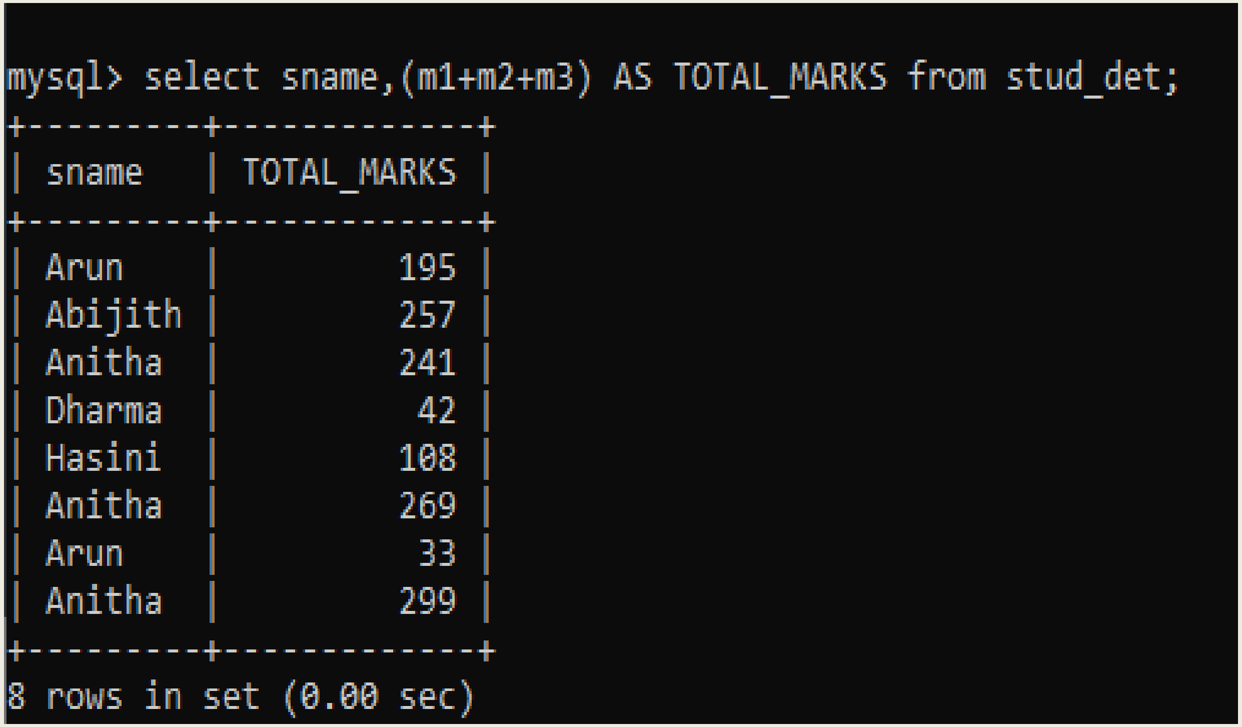
```
mysql> select prodid, prodname,SUM(price) from prod_det group by prodname having SUM(price)<40000;
+--------+----------+------------+
| prodid | prodname | SUM(price) |
+--------+----------+------------+
|    103 | DELL     |   35000.65 |
+--------+----------+------------+
1 row in set (0.00 sec)
```



```
mysql> select prodid, prodname,MIN(price) from prod_det group by prodname having MIN(price)<35000;
+--------+----------+------------+
| prodid | prodname | MIN(price) |
+--------+----------+------------+
|    101 | samsung  |   15000.25 |
+--------+----------+------------+
1 row in set (0.00 sec)
```

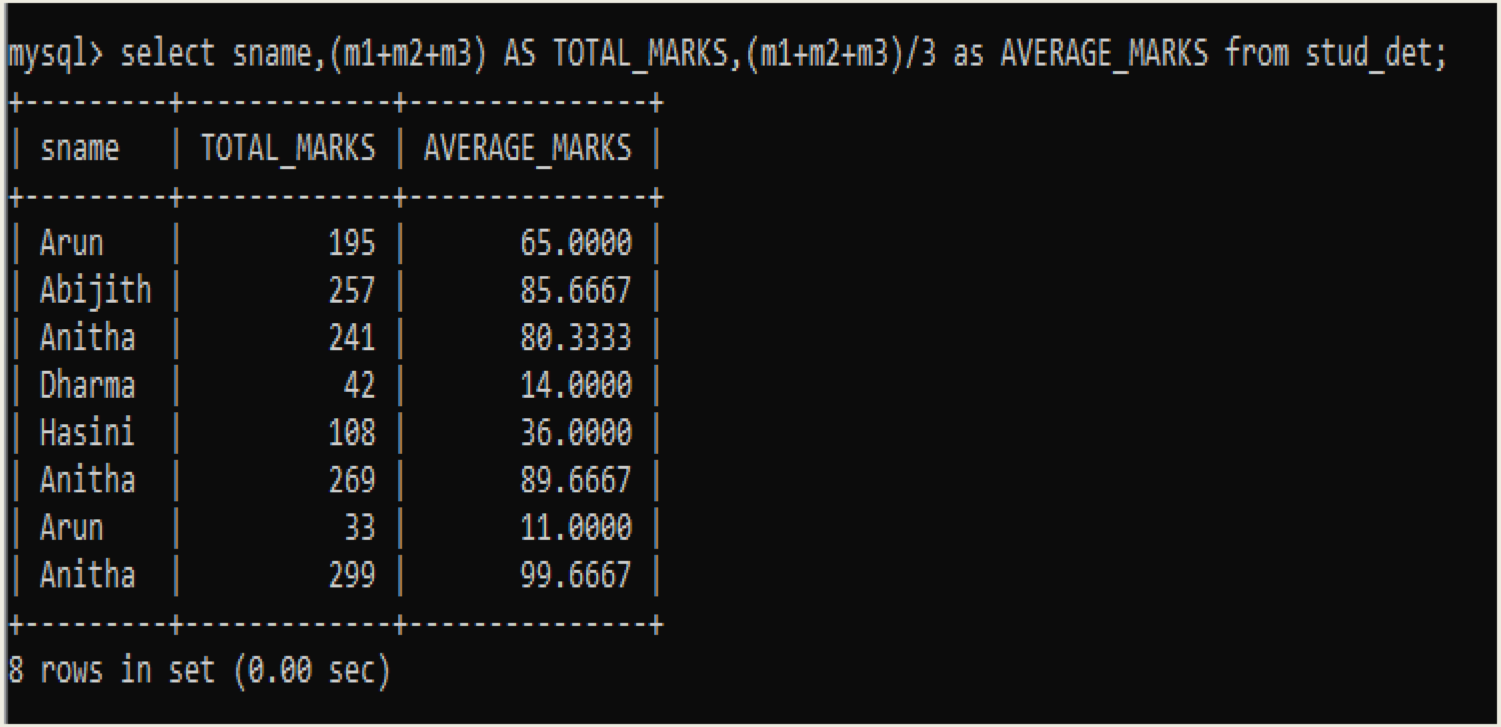
Expressions in Querying

We can write **expressions** in various SQL clauses.
Expressions can comprise of various data types like integers, floats, strings, datetime, etc.



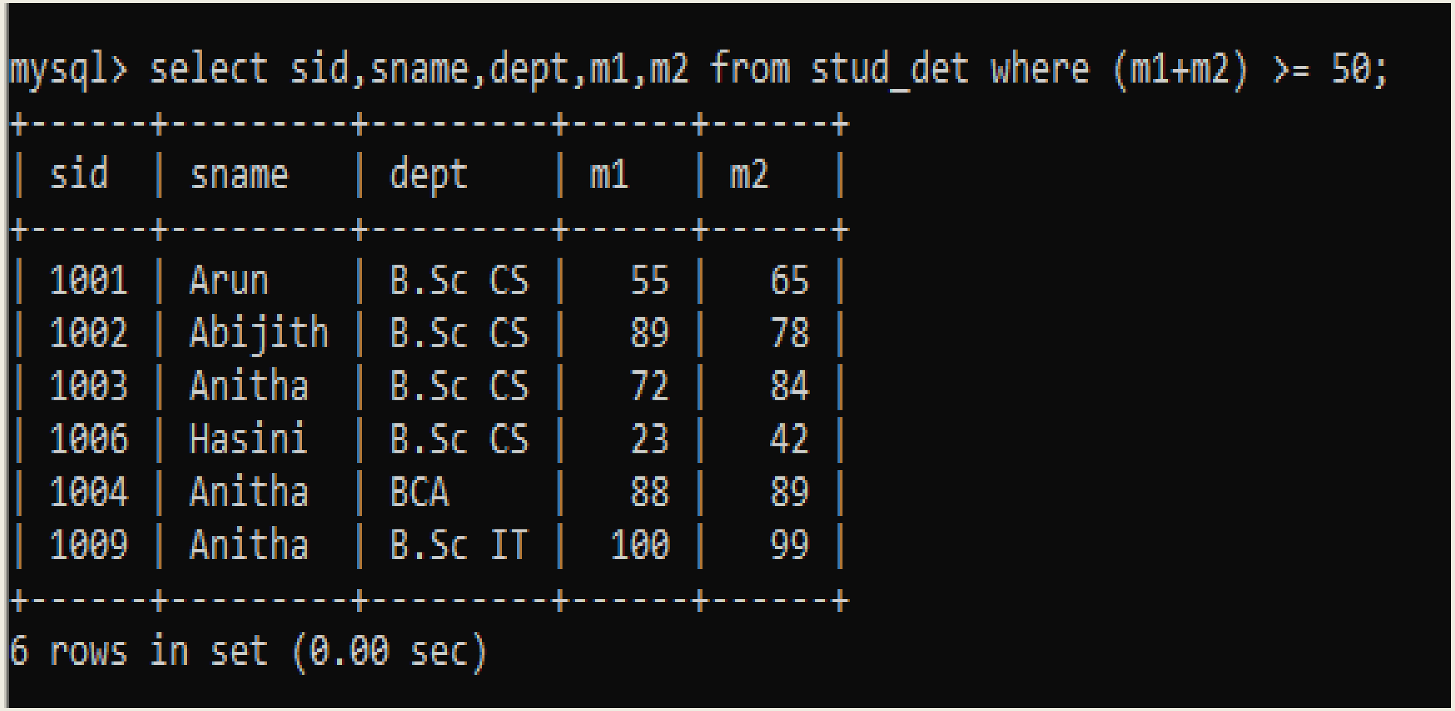
```
mysql> select sname,(m1+m2+m3) AS TOTAL_MARKS from stud_det;
+---------+-------------+
| sname   | TOTAL_MARKS |
+---------+-------------+
| Arun    |         195 |
| Abijith |         257 |
| Anitha  |         241 |
| Dharma  |          42 |
| Hasini  |         108 |
| Anitha  |         269 |
| Arun    |          33 |
| Anitha  |         299 |
+---------+-------------+
8 rows in set (0.00 sec)
```

Expressions in Querying



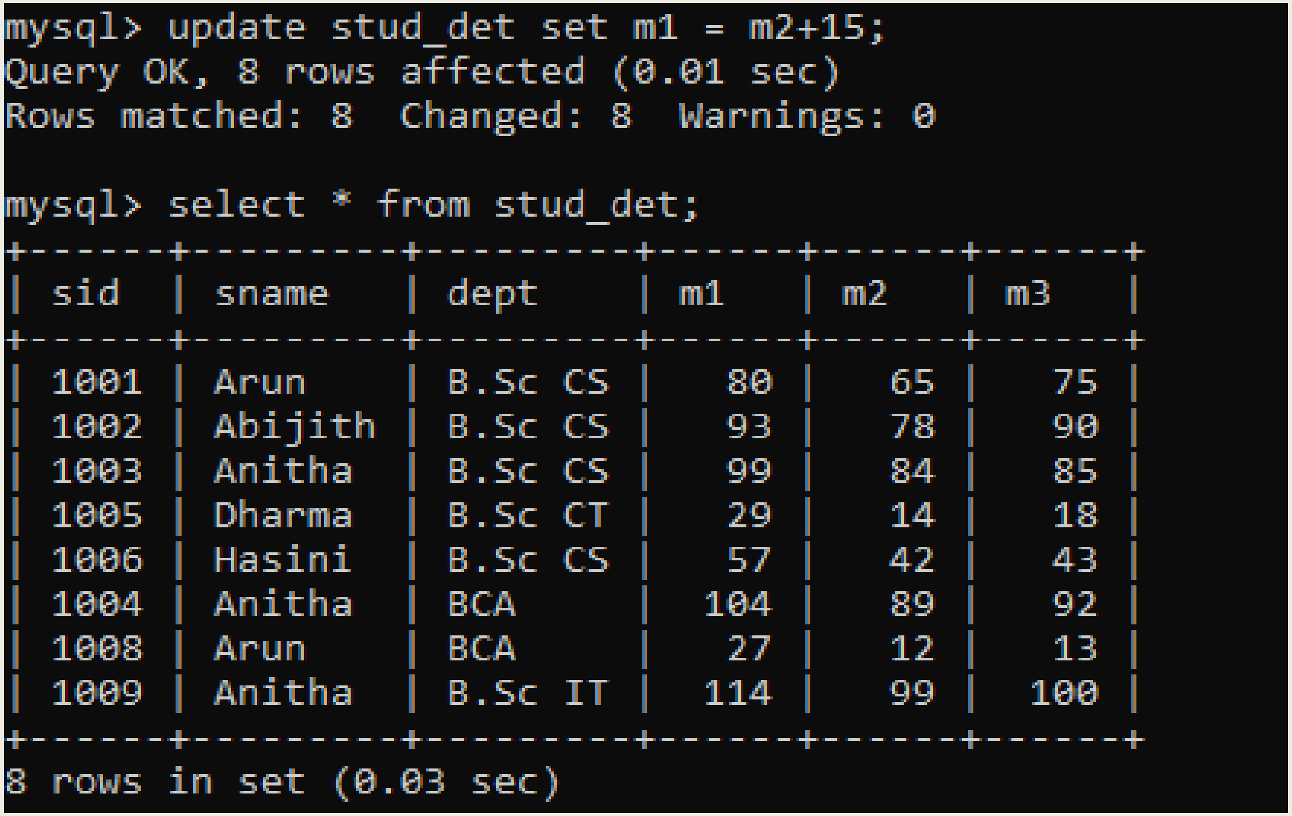
```
mysql> select sname,(m1+m2+m3) AS TOTAL_MARKS,(m1+m2+m3)/3 as AVERAGE_MARKS from stud_det;
+---------+-------------+---------------+
| sname   | TOTAL_MARKS | AVERAGE_MARKS |
+---------+-------------+---------------+
| Arun    |         195 |       65.0000 |
| Abijith |         257 |       85.6667 |
| Anitha  |         241 |       80.3333 |
| Dharma  |          42 |       14.0000 |
| Hasini  |         108 |       36.0000 |
| Anitha  |         269 |       89.6667 |
| Arun    |          33 |       11.0000 |
| Anitha  |         299 |       99.6667 |
+---------+-------------+---------------+
8 rows in set (0.00 sec)
```

Using Expressions in WHERE Clause



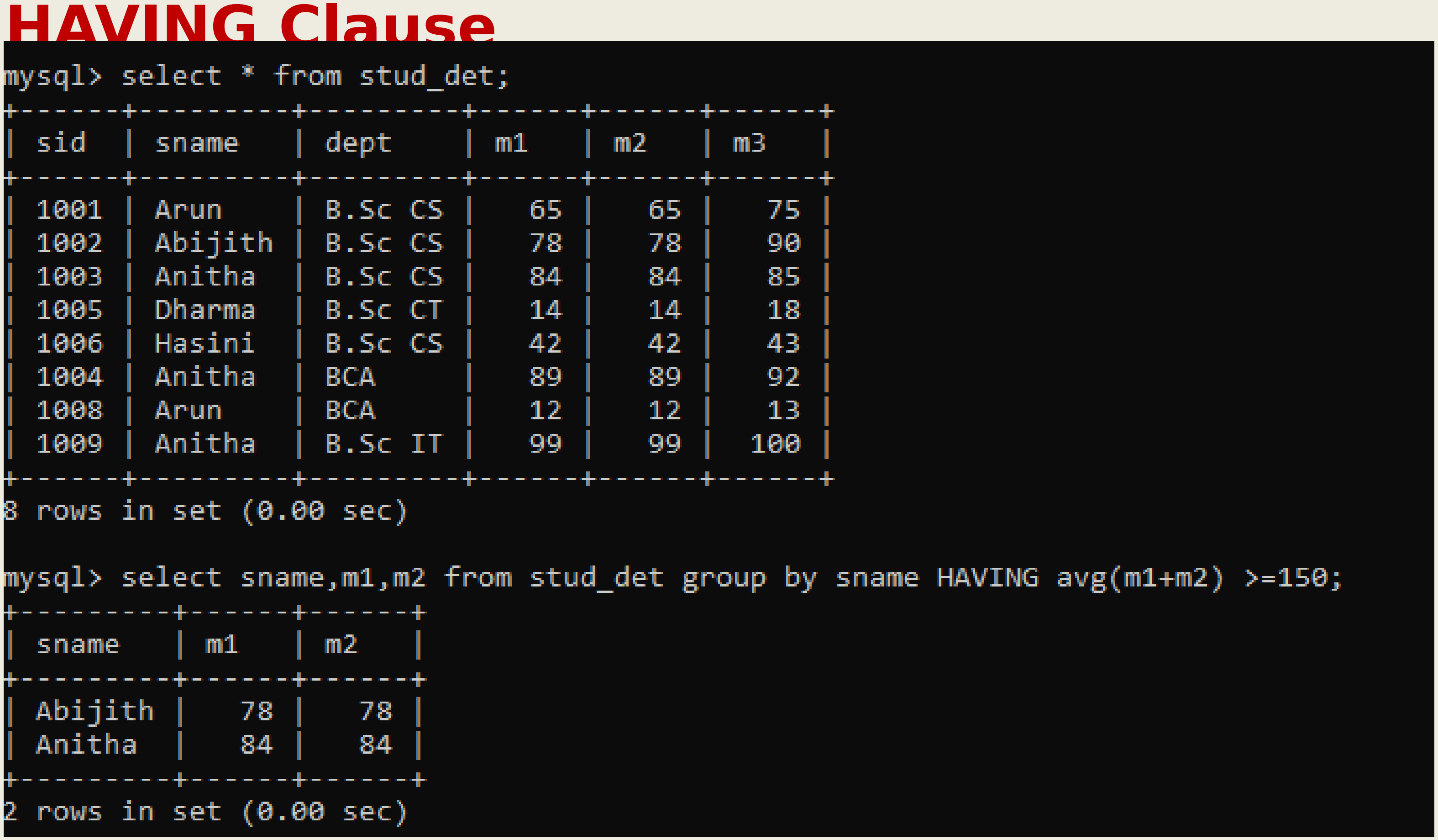
```
mysql> select sid,sname,dept,m1,m2 from stud_det where (m1+m2) >= 50;
+------+---------+---------+------+------+
| sid  | sname   | dept    | m1   | m2   |
+------+---------+---------+------+------+
| 1001 | Arun    | B.Sc CS |   55 |   65 |
| 1002 | Abijith | B.Sc CS |   89 |   78 |
| 1003 | Anitha  | B.Sc CS |   72 |   84 |
| 1006 | Hasini  | B.Sc CS |   23 |   42 |
| 1004 | Anitha  | BCA     |   88 |   89 |
| 1009 | Anitha  | B.Sc IT |  100 |   99 |
+------+---------+---------+------+------+
6 rows in set (0.00 sec)
```

Using Expressions in UPDATE Clause

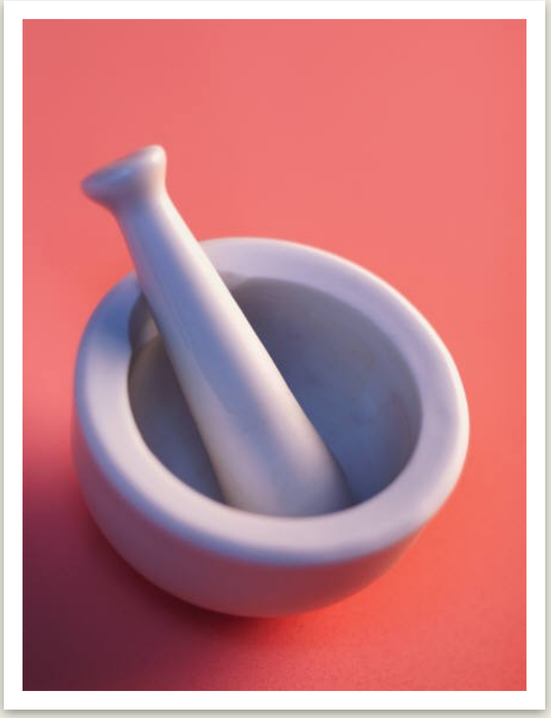


```
mysql> update stud_det set m1 = m2+15;
Query OK, 8 rows affected (0.01 sec)
Rows matched: 8  Changed: 8  Warnings: 0

mysql> select * from stud_det;
+------+---------+---------+------+------+------+
| sid  | sname   | dept    | m1   | m2   | m3   |
+------+---------+---------+------+------+------+
| 1001 | Arun    | B.Sc CS |   80 |   65 |   75 |
| 1002 | Abijith | B.Sc CS |   93 |   78 |   90 |
| 1003 | Anitha  | B.Sc CS |   99 |   84 |   85 |
| 1005 | Dharma  | B.Sc CT |   29 |   14 |   18 |
| 1006 | Hasini  | B.Sc CS |   57 |   42 |   43 |
| 1004 | Anitha  | BCA     |  104 |   89 |   92 |
| 1008 | Arun    | BCA     |   27 |   12 |   13 |
| 1009 | Anitha  | B.Sc IT |  114 |   99 |  100 |
+------+---------+---------+------+------+------+
8 rows in set (0.03 sec)
```

Expressions in HAVING Clause



```
mysql> select * from stud_det;
+------+---------+---------+------+------+------+
| sid  | sname   | dept    | m1   | m2   | m3   |
+------+---------+---------+------+------+------+
| 1001 | Arun    | B.Sc CS |   65 |   65 |   75 |
| 1002 | Abijith | B.Sc CS |   78 |   78 |   90 |
| 1003 | Anitha  | B.Sc CS |   84 |   84 |   85 |
| 1005 | Dharma  | B.Sc CT |   14 |   14 |   18 |
| 1006 | Hasini  | B.Sc CS |   42 |   42 |   43 |
| 1004 | Anitha  | BCA     |   89 |   89 |   92 |
| 1008 | Arun    | BCA     |   12 |   12 |   13 |
| 1009 | Anitha  | B.Sc IT |   99 |   99 |  100 |
+------+---------+---------+------+------+------+
8 rows in set (0.00 sec)

mysql> select sname,m1,m2 from stud_det group by sname HAVING avg(m1+m2) >=150;
+---------+------+------+
| sname   | m1   | m2   |
+---------+------+------+
| Abijith |   78 |   78 |
| Anitha  |   84 |   84 |
+---------+------+------+
2 rows in set (0.00 sec)
```



Thank You