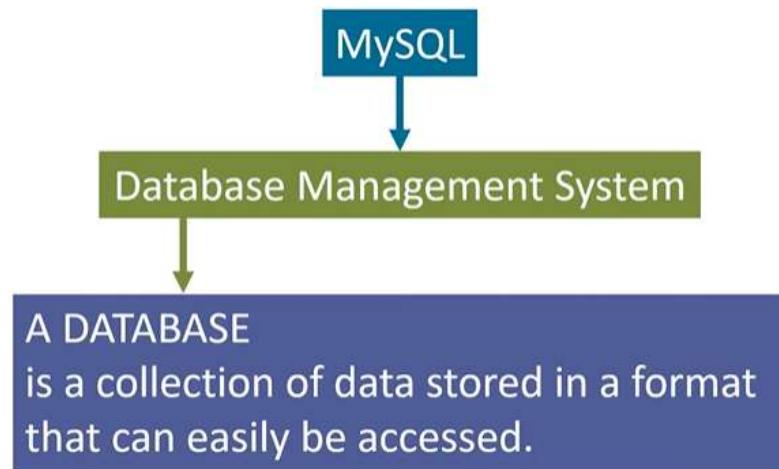


# 1. MySQL Introduction



## What is MySQL ?



## What is Database ?

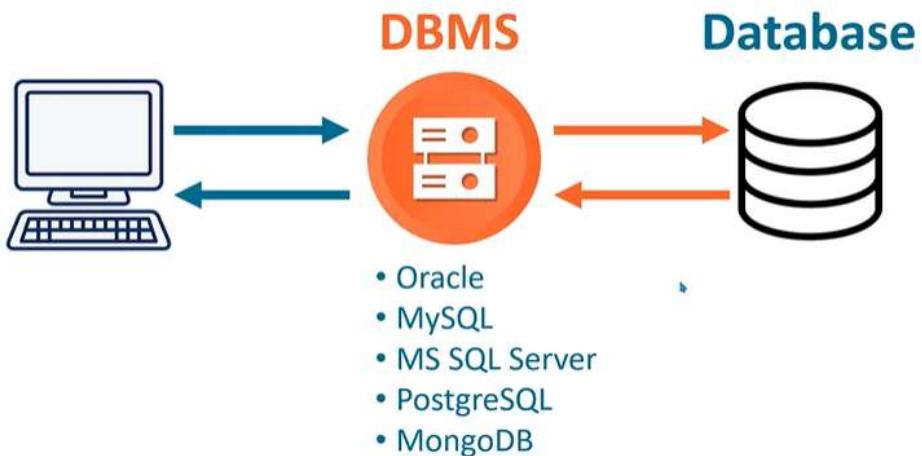
A form with fields for Name (text input), Age (text input), Gender (radio buttons for Male and Female), and a Submit button.

Collection of Data

Name	Age	Gender
Ram Kumar	21	Male
Salman Khan	22	Male
Meera Khan	21	Female
Sarita Kumari	21	Female
Anil Kapoor	22	Male



## How DBMS Works ?

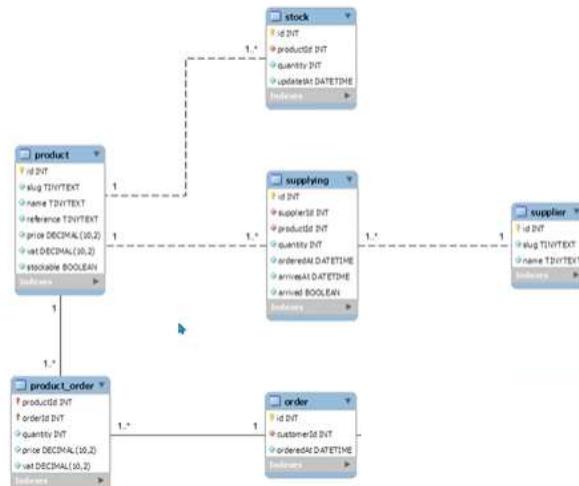


## Two type of Database :

### Relational

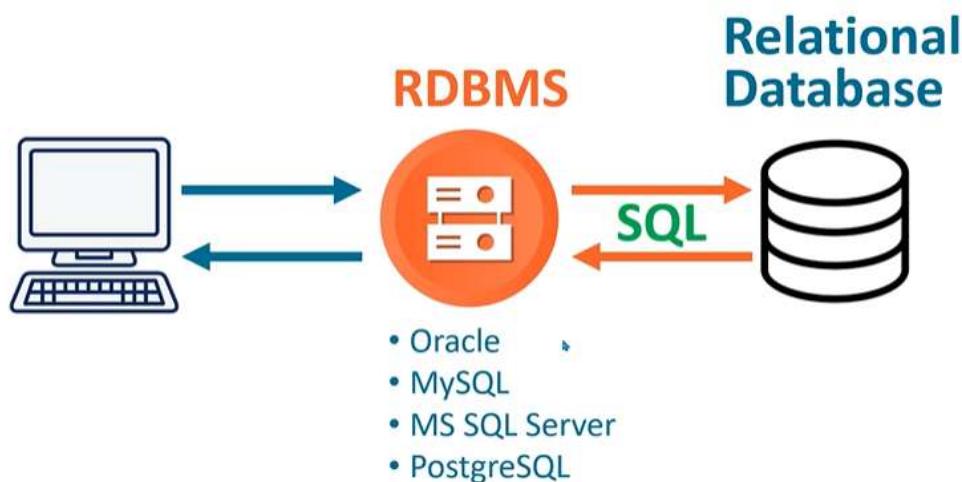
- Relational Database (RDBMS)
- RDBMS use SQL

### Structured Query Language





## Relational Database Management System



## Two type of Database :

### NoSQL

- NoSQL is not a Table-based Database.
- NoSQL databases are document based.
- MongoDB, Redis, Cassandra etc.



## Advantage of MySQL :

- Cross Platform
- Used with multiple languages (PHP, NodeJS, Python, C#).
- MySQL software is Open Source.
- MySQL is RDBMS.
- The MySQL Database Server is fast, reliable, scalable, and easy to use.
- MySQL Server works in client/server or embedded systems.



## Popular Websites Using MySQL

- Facebook
- Twitter
- Google
- Wikipedia
- Youtube
- Flickr
- Pinterest



## What we will learn in this course ?

- Create Database & Tables in database.
- Add Data in database.
- Update Data in database.
- Read data from database.
- Delete Data from database.

SQL Commands

## 2. MySQL Installation



## MySQL Installation Method 1 :

Students of PHP Course

- Install XAMPP, WAMP, MAMP
- MySQL Workbench



## MySQL Installation Method 2 :

- MySQL Community Server
- MySQL Workbench

### 3. MySQL Create Table



## How to create SQL Table ?

Name	Age	Gender
Ram Kumar	21	Male
Salman Khan	22	Male
Meera Khan	21	Female
Sarita Kumari	21	Female
Anil Kapoor	22	Male



## How to create SQL Table ?

Name = String

Age = Numeric

Gender = String

Column Names    Column Datatypes

Name	Age	Gender
Ram Kumar	21	Male
Salman Khan	22	Male
Meera Khan	21	Female
Sarita Kumari	21	Female
Anil Kapoor	22	Male

Table Name



## Create Table Syntax

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



## Datatypes in MySQL

3 Type of Category in Datatypes :

1. String

2. Numeric

3. Date and Time



## String Datatypes in MySQL

1. CHAR(size) **0 to 255**
2. VARCHAR(size) **0 to 65535**
3. BINARY(size)
4. VARBINARY(size)
5. TINYTEXT **255 characters**
6. TEXT(size) **65,535 bytes**
7. MEDIUMTEXT **16,777,215 characters**
8. LONGTEXT **4,294,967,295 characters**
9. TINYBLOB **255 bytes**
10. BLOB(size) **65,535 bytes**
11. MEDIUMBLOB **16,777,215 bytes**
12. LONGBLOB **4,294,967,295 bytes**
13. ENUM(val1, val2, val3, ...) **list up to 65535 values**
14. SET(val1, val2, val3, ...) **list up to 64 values**



## Numeric Datatypes in MySQL

1. BIT(size) **1 to 64**
2. TINYINT(size) **-128 to 127**
3. INT(size) **-2147483648 to 2147483647**
4. INTEGER(size)
5. SMALLINT(size) **-32768 to 32767**
6. MEDIUMINT(size) **-8388608 to 8388607**
7. BIGINT(size) **-9223372036854775808 to 9223372036854775807**
8. BOOL
9. BOOLEAN **0 / 1**
10. FLOAT(p)
11. DOUBLE(size, d) **255.568**
12. DECIMAL(size, d) **Size = 60 , d = 30**
13. DEC(size, d)



## Date & Time Datatypes in MySQL

1. DATE '1000-01-01' to '9999-12-31'
2. DATETIME(fsp) YYYY-MM-DD hh:mm:ss
3. TIMESTAMP(fsp)
4. TIME(fsp) hh:mm:ss
5. YEAR four-digit format : 1901

The screenshot shows the MySQL Workbench interface. The top bar displays "Local instance MySQL80". The menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. Below the menu is a toolbar with various icons. The left pane is titled "Navigator" and shows "SCHEMAS" with a search bar for "Filter objects". Under "SCHEMAS", there are three entries: dept, emp\_data, and personal. The personal schema is expanded, showing "Columns" with columns id, name, birth\_date, phone, and gender. It also shows "Indexes" and "Foreign Keys". The right pane is titled "SQL File 3\*" and contains the following SQL code:

```
3
4 • - create table personal(
5   id int,
6   name varchar(50),
7   birth_date date,
8   phone varchar(12),
9   gender varchar(1)
10 );
11
```

## 4. MySQL Insert



## How to Insert data in Tables with SQL ?

Name	Age	Gender
Ram Kumar	21	Male
Salman Khan	22	Male
Meera Khan	21	Female
Sarita Kumari	21	Female
Anil Kapoor	22	Male

```
INSERT INTO table_name ( column1, column2, ....)
VALUES ( value1, value2,....);
```

The screenshot shows the MySQL Workbench interface. The Navigator pane on the left displays the database schema with the following structure:

- SCHEMAS:
  - course
  - dept
  - emp\_data
  - personal
    - Columns:
      - id
      - name
      - birth\_date
      - phone
      - gender
    - Indexes
    - Foreign Keys
- Administration
- Schemas
- Information

The SQL Editor pane on the right contains the following SQL code:

```
1
2 • insert into personal(id, name, birth_date, phone, gender)
3   values (1, "Ram Kumar", "1990-07-15", "9999997771", "M");
4
5 • insert into personal(id, name, birth_date, phone, gender)
6   values (2, "Sita Kumari", "1991-07-15", "9995557771", "F");
7
8 • insert into personal(id, name, birth_date, phone, gender)
9   values (3, "Luxman Kumar", "1995-09-11", "9999997771", "M");
10
11 • insert into personal(id, name, birth_date, phone, gender)
12   values (4, "Bharat Kumar", "1994-08-12", "9995557771", "M");
13
```

## 5. MySQL INSERT Multiple Rows



## Insert Data in Tables with SQL :

Name	Age	Gender
Ram Kumar	21	Male
Salman Khan	22	Male
Meera Khan	21	Female
Sarita Kumari	21	Female
Anil Kapoor	22	Male

```
INSERT INTO table_name ( column1, column2, .... )
VALUES ( value1, value2,....);
```



## Insert Multiple Rows Syntax :

```
INSERT INTO table_name ( column1, column2, .... )
```

```
VALUES
```

```
( value1, value2,.... ),
```

```
( value1, value2,.... ),
```

```
( value1, value2,.... );
```

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database structure under the 'personal' schema, including tables like course, dept, emp\_data, and personal, along with its columns: id, name, birth\_date, phone, and gender. The right pane shows an SQL editor titled 'SQL File 3\*' with the following content:

```
1
2 •  insert into personal(id, name, birth_date, phone, gender)
3   values
4   (5, "Anil Kapur", "1965-07-15", "9999997771", "M"),
5   (6, "Juhi Chavla", "1970-11-5", "9995557771", "F"),
6   (7, "Madhuri Dikshit", "1971-09-11", "993337771", "F"),
7   (8, "Amitabh Bachchan", "1954-08-1", "9999999999", "M");
```

## 6. MySQL Constraints

Constraints meance restrictions



## List of Constraints in MySQL

- NOT NULL
- UNIQUE
- DEFAULT
- CHECK
- FOREIGN KEY
- PRIMARY KEY



## Data Table without Constraints

ID	Name	Age	Gender	Phone	City
1	Ram Kumar	17	Male	4022155	Agra
2	Salman Khan	19		4033244	Agra
3	Meera Khan	20	Female	4022155	Agra
	Sarita Kumari	18	Female	4066899	Agra
5	Anil Kapoor	19	Male	4188733	Agra



## Data Table without Constraints

ID	Name	Age	Gender	Phone	City
1	Ram Kumar	17	Male	4022155	Agra
2	Salman Khan	19		4033244	Agra
3	Meera Khan	20	Female	4022155	Agra
	Sarita Kumari	18	Female	4066899	Agra
5	Anil Kapoor	19	Male	4188733	Agra

↓  
NOT NULL  
↓  
UNIQUE

↓  
NOT NULL  
↓  
CHECK (age >= 18)

↓  
UNIQUE

↓  
DEFAULT 'Agra'



## Create Table Syntax

```
CREATE TABLE table_name (
    id INT NOT NULL UNIQUE,
    name VARCHAR(50) NOT NULL,
    age INT NOT NULL CHECK (age >= 18),
    gender VARCHAR(10) NOT NULL,
    phone VARCHAR(10) NOT NULL UNIQUE,
    city VARCHAR(10) NOT NULL DEFAULT 'Agra',
);
```

The screenshot shows the MySQL Workbench interface. On the left, the Navigator pane displays the database structure under the 'student' schema, including tables like 'personal', 'product', and 'Views'. The 'personal' table is selected, showing its columns: id, name, birth\_date, phone, and gender. The main area shows the 'Result Grid' containing 10 rows of data:

	id	name	birth_date	phone	gender
2	Sita Kumari	1991-07-15	9995557771	F	
3	Luxman Kumar	1995-09-11	9999997771	M	
4	Bharat Kumari	1994-08-12	9995557771	F	
5	Anil Kapur	1965-07-15	9999997771	M	
6	Juhi Chavla	1970-11-05	9995557771	F	
7	Madhuri Dikshit	1971-09-11	993337771	F	
8	Amitabh Bach...	1954-08-01	9999999999	M	
9	Amir Kapur	1965-07-15	9999997771	M	
10	Salman Khan	1965-07-15	1111111111	NULL	

The screenshot shows the MySQL Workbench Query Editor. The 'personal' table is selected in the Navigator. The Query Editor window displays the CREATE TABLE statement for the 'personal' table:

```
1 • CREATE TABLE personal(
2     id INT NOT NULL UNIQUE,
3     name VARCHAR(50) NOT NULL,
4     age INT NOT NULL CHECK (age >= 18),
5     gender VARCHAR(10) NOT NULL,
6     phone VARCHAR(10) NOT NULL UNIQUE,
7     city VARCHAR(15) NOT NULL DEFAULT 'Agra'
8 );
9
```

## 7. MySQL SELECT With WHERE Clause



## How to show data from Tables with SQL ?

Name	Age	Gender
Ram Kumar	19	Male
Salman Khan	22	Male
Meera Khan	21	Female
Sarita Kumari	18	Female
Anil Kapoor	22	Male

## SELECT Command



## SELECT Syntax

`SELECT column1, column2, column3, ....`

`FROM table_name;`

`SELECT *`

`FROM table_name;`

The screenshot shows the MySQL Workbench interface. The Navigator pane on the left displays the database schema, including the 'student' schema which contains a 'personal' table with columns: id, name, age, gender, phone, and city. The Query Editor pane at the top has the query: `1 • SELECT id, name, phone FROM personal;`. The Result Grid at the bottom shows the following data:

	id	name	phone
1	Ram Kumar	4022155	
2	Sarita Kumari	4034421	
3	Salman Khan	4056221	
4	Juhi Chawla	4089821	
5	Anil Kapoor	4025221	
6	John Abraham	4033776	

Using alias for column name

Navigator

**SCHEMAS**

- Filter objects
- phpmyadmin
- student**
- Tables
- personal
- Columns
- id
- name
- age
- gender
- phone
- city
- Indexes
- Foreign Keys
- Triggers
- Views
- Stored Procedures
- Functions

Administration Schemas Information

No object selected

Query 1 x

```
1 • SELECT id AS Id, name AS Student , phone AS Phone FROM personal;
```

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

	<b>Id</b>	<b>Student</b>	<b>Phone</b>
1	Ram Kumar	4022155	
2	Sarita Kumari	4034421	
3	Salman Khan	4056221	
4	Juhi Chawla	4089821	
5	Anil Kapoor	4025221	
6	John Abraham	4033776	

If column name is double that we need to use colon

Navigator

**SCHEMAS**

- Filter objects
- phpmyadmin
- student**
- Tables
- personal
- Columns
- id
- name
- age
- gender
- phone
- city
- Indexes
- Foreign Keys
- Triggers
- Views
- Stored Procedures
- Functions

Administration Schemas Information

No object selected

YouTube Home

Query 1 x

```
1 • SELECT id AS Id, name AS "Student Name" , phone AS Phone FROM personal;
```

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

	<b>Id</b>	<b>Student Name</b>	<b>Phone</b>
1	Ram Kumar	4022155	
2	Sarita Kumari	4034421	
3	Salman Khan	4056221	
4	Juhi Chawla	4089821	
5	Anil Kapoor	4025221	
6	John Abraham	4033776	

Conditional data checking

## MySQL SELECT Data with WHERE Clause

Name	Age	Gender	Name	Age	Gender
Ram Kumar	19	Male	Meera Khan	21	Female
Salman Khan	22	Male	Sarita Kumari	18	Female
Meera Khan	21	Female			
Sarita Kumari	18	Female			
Anil Kapoor	22	Male			

Name	Age	Gender
Ram Kumar	19	Male
Sarita Kumari	18	Female

## SELECT with WHERE Clause



# WHERE Comparison Operators

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<> Or !=	Not equal.
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

Navigator: Schemas

Query 1 ×

```
1 • SELECT * FROM personal
2 WHERE gender = "F";
```

Result Grid | Filter Rows: [ ] | Edit: [ ] | Export/Import: [ ] | Wrap Cell Content: [ ]

	id	name	age	gender	phone	city
▶	2	Sarita Kumari	21	F	4034421	Delhi
▶	4	Juhি Chawla	18	F	4089821	Bhopal
...						

Navigator: Schemas

Query 1 ×

```
1 • SELECT * FROM personal
2 WHERE age < 20;
```

Result Grid | Filter Rows: [ ] | Edit: [ ] | Export/Import: [ ] | Wrap Cell Content: [ ]

	id	name	age	gender	phone	city
▶	1	Ram Kumar	19	M	4022155	Agra
▶	4	Juhি Chawla	18	F	4089821	Bhopal
...						

No object selected

Navigator

**SCHEMAS**

Filter objects

- phpmyadmin
- student**
  - Tables
    - personal**
      - Columns
        - id
        - name
        - age
        - gender
        - phone
        - city
      - Indexes
      - Foreign Keys
      - Triggers
    - Views
    - Stored Procedures
    - Functions
- test

Administration Schemas Information

No object selected

Query 1

```
1 • SELECT * FROM personal
2 WHERE age <= 20;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	<b>id</b>	<b>name</b>	<b>age</b>	<b>gender</b>	<b>phone</b>	<b>city</b>
1	Ram Kumar	19	M	4022155	Agra	
3	Salman Khan	20	M	4056221	Agra	
4	Juhi Chawla	18	F	4089821	Bhopal	
*	HULL	HULL	HULL	HULL	HULL	

Navigator

**SCHEMAS**

Filter objects

- phpmyadmin
- student**
  - Tables
    - personal**
      - Columns
        - id
        - name
        - age
        - gender
        - phone
        - city
      - Indexes
      - Foreign Keys
      - Triggers
    - Views
    - Stored Procedures
    - Functions
- test

Administration Schemas Information

No object selected

Query 1

```
1 • SELECT * FROM personal
2 WHERE city != "Agra";
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	<b>id</b>	<b>name</b>	<b>age</b>	<b>gender</b>	<b>phone</b>	<b>city</b>
2	Sarita Kumari	21	F	4034421	Delhi	
4	Juhi Chawla	18	F	4089821	Bhopal	
6	John Abraham	21	M	4033776	Delhi	
*	HULL	HULL	HULL	HULL	HULL	

same with other sign

Navigator

**SCHEMAS**

Filter objects

- phpmyadmin
- student**
  - Tables
    - personal**
      - Columns
        - id
        - name
        - age
        - gender
        - phone
        - city
      - Indexes
      - Foreign Keys
      - Triggers
    - Views
    - Stored Procedures
    - Functions
- test

Administration Schemas Information

No object selected

Query 1

```
1 • SELECT * FROM personal
2 WHERE city <> "Agra";
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	<b>id</b>	<b>name</b>	<b>age</b>	<b>gender</b>	<b>phone</b>	<b>city</b>
2	Sarita Kumari	21	F	4034421	Delhi	
4	Juhi Chawla	18	F	4089821	Bhopal	
6	John Abraham	21	M	4033776	Delhi	
*	HULL	HULL	HULL	HULL	HULL	

## 8. MySQL AND, OR, NOT Operators



### SELECT Data with AND & OR Operators

WHERE Age  $\geq$  18 AND Age  $\leq$  21

Name	Age	Gender
Ram Kumar	19	Male
Salman Khan	22	Male
Meera Khan	21	Female
Sarita Kumari	18	Female
Anil Kapoor	22	Male

Name	Age	Gender
Ram Kumar	19	Male
Meera Khan	21	Female
Sarita Kumari	18	Female



### SELECT Data with AND & OR Operators

WHERE Age  $\geq$  18 AND Age  $\leq$  21

Name	Age	Gender
Ram Kumar	19	Male
Salman Khan	22	Male
Meera Khan	21	Female
Sarita Kumari	18	Female
Anil Kapoor	22	Male

Name	Age	Gender
Ram Kumar	19	Male
Meera Khan	21	Female
Sarita Kumari	18	Female

Name	Age	Gender
Meera Khan	21	Female
Sarita Kumari	18	Female

WHERE Age = 18 OR Age = 21



### SELECT with AND & OR Operator Syntax

SELECT column1, column2, column3, ....

FROM table\_name

WHERE condition1 AND condition2 AND condition3 ...;

SELECT column1, column2, column3, ....

FROM table\_name

WHERE condition1 OR condition2 OR condition3 ...;

## Practical Implementation

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

phpmyadmin

student

Tables

personal

Views

Stored Procedures

Functions

test

Query 1

1 • **SELECT \* FROM personal**

2 WHERE age >= 18 AND age <= 21;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

id	name	age	gender	phone	city
1	Ram Kumar	19	M	4022155	Agra
2	Sarita Kumari	21	F	4034421	Delhi
3	Salman Khan	20	M	4056221	Agra
4	Juhi Chawla	18	F	4089821	Bhopal
6	John Abraham	21	M	4033776	Delhi
NULL	NULL	NULL	NULL	NULL	NULL

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

phpmyadmin

student

Tables

personal

Views

Stored Procedures

Functions

test

Query 1

1 • **SELECT \* FROM personal**

2 WHERE age <= 20 AND gender = "M";

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

id	name	age	gender	phone	city
1	Ram Kumar	19	M	4022155	Agra
3	Salman Khan	20	M	4056221	Agra
NULL	NULL	NULL	NULL	NULL	NULL

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

student

Tables

personal

Views

Stored Procedures

Functions

test

Query 1

1 • **SELECT \* FROM personal**

2 WHERE age <= 20 AND gender = "M" AND city = "Agra";

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	id	name	age	gender	phone	city
>	1	Ram Kumar	19	M	4022155	Agra
	3	Salman Khan	20	M	4056221	Agra

No object selected

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

student

Tables

personal

Views

Stored Procedures

Functions

test

Query 1

1 • **SELECT \* FROM personal**

2 WHERE age <= 20 OR city = "Agra";

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	id	name	age	gender	phone	city
>	1	Ram Kumar	19	M	4022155	Agra
	3	Salman Khan	20	M	4056221	Agra
	4	Juhি Chawla	18	F	4089821	Bhopal
	5	Anil Kapoor	22	M	4025221	Agra

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

student

Tables

personal

Views

Stored Procedures

Functions

test

Query 1

1 • **SELECT \* FROM personal**

2 WHERE city = "Bhopal" OR city = "Agra";

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	<b>id</b>	<b>name</b>	<b>age</b>	<b>gender</b>	<b>phone</b>	<b>city</b>
▶	1	Ram Kumar	19	M	4022155	Agra
▶	3	Salman Khan	20	M	4056221	Agra
▶	4	Juhি Chawla	18	F	4089821	Bhopal
▶	5	Anil Kapoor	22	M	4025221	Agra

No object selected

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

student

Tables

personal

Views

Stored Procedures

Functions

test

Query 1

1 • **SELECT \* FROM personal**

2 WHERE (city = "Bhopal" OR city = "Agra") AND gender = "M";

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	<b>id</b>	<b>name</b>	<b>age</b>	<b>gender</b>	<b>phone</b>	<b>city</b>
▶	1	Ram Kumar	19	M	4022155	Agra
▶	3	Salman Khan	20	M	4056221	Agra
▶	5	Anil Kapoor	22	M	4025221	Agra

No object selected

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

student

Tables

personal

Views

Stored Procedures

Functions

test

Query 1

1 • **SELECT \* FROM personal**

2 WHERE NOT age >= 20;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	<b>id</b>	<b>name</b>	<b>age</b>	<b>gender</b>	<b>phone</b>	<b>city</b>
▶	1	Ram Kumar	19	M	4022155	Agra
▶	4	Juhি Chawla	18	F	4089821	Bhopal

## 9. MySQL IN Operator



### SELECT Data with IN Operator

Name	Age	Gender
Ram Kumar	19	Male
Salman Khan	22	Male
Meera Khan	21	Female
Sarita Kumari	18	Female
Anil Kapoor	22	Male

→

Name	Age	Gender
Meera Khan	21	Female
Sarita Kumari	18	Female

**WHERE Age = 18 OR Age = 21**



### SELECT Data with IN Operator

Name	Age	Gender
Ram Kumar	19	Male
Salman Khan	22	Male
Meera Khan	21	Female
Sarita Kumari	18	Female
Anil Kapoor	22	Male

→

Name	Age	Gender
Meera Khan	21	Female
Sarita Kumari	18	Female

**WHERE Age IN ( 18, 21 )**

**WHERE Age = 18 OR Age = 21**



### SELECT with IN Operator Syntax

**SELECT column1, column2, column3, ....**

**FROM table\_name**

**WHERE column\_name IN (value1, value2, ...);**

**SELECT column1, column2, column3, ....**

**FROM table\_name**

**WHERE column\_name NOT IN (value1, value2, ...);**

**personal**

```
1 • SELECT * FROM personal
2 WHERE age IN (18,21, 19);
```

	id	name	age	gender	phone	city
▶	1	Ram Kumar	19	M	4022155	Agra
2	Sarita Kumari	21	F	4034421	Delhi	
4	Juhi Chawla	18	F	4089821	Bhopal	
6	John Abraham	21	M	4033776	Delhi	

**personal**

```
1 • SELECT * FROM personal
2 WHERE age NOT IN (18,21, 19);
```

	id	name	age	gender	phone	city
▶	3	Salman Khan	20	M	4056221	Agra
5	Anil Kapoor	22	M	4025221	Agra	

**personal**

```
1 • SELECT * FROM personal
2 WHERE city IN ("Delhi", "Bhopal");
```

	id	name	age	gender	phone	city
▶	2	Sarita Kumari	21	F	4034421	Delhi
4	Juhi Chawla	18	F	4089821	Bhopal	
6	John Abraham	21	M	4033776	Delhi	

The screenshot shows the phpMyAdmin interface for a database named 'personal'. In the left sidebar, under 'SCHEMAS', the 'student' schema is selected, which contains a 'personal' table. The main area displays a query results grid. The query executed was:

```
1 • SELECT * FROM personal  
2 WHERE city NOT IN ("Delhi", "Bhopal");
```

The resulting grid shows three rows of data:

	id	name	age	gender	phone	city
▶	1	Ram Kumar	19	M	4022155	Agra
▶	3	Salman Khan	20	M	4056221	Agra
▶	5	Anil Kapoor	22	M	4025221	Agra

No object selected

The screenshot shows the phpMyAdmin interface for a database named 'personal'. In the left sidebar, under 'SCHEMAS', the 'student' schema is selected, which contains a 'personal' table. The main area displays a query results grid. The query executed was:

```
1 • SELECT * FROM personal  
2 WHERE id IN (1,3,4);
```

The resulting grid shows four rows of data, with the third row (id 3) highlighted by a blue cursor arrow:

	id	name	age	gender	phone	city
▶	1	Ram Kumar	19	M	4022155	Agra
▶	3	Salman Khan	20	M	4056221	Agra
▶	4	Juhি Chawla	18	F	4089821	Bhopal

No object selected

## 10. MySQL BETWEEN & NOT BETWEEN Operator



## SELECT Data with BETWEEN Operator

Student Table				WHERE Age BETWEEN 18 AND 20		
Name	Age	Gender	DOB	Name	Age	Gender
Ram Kumar	19	Male	1998-02-10	Ram Kumar	19	Male
Salman Khan	17	Male	1999-07-22	Meera Khan	19	Female
Meera Khan	19	Female	1998-05-11	Anil Kapoor	20	Male
Sarita Kumari	21	Female	1996-10-15			
Anil Kapoor	20	Male	1997-03-12			

January 1998 to June 1998

Name	Age	Gender
Ram Kumar	19	Male
Meera Khan	19	Female

WHERE DOB BETWEEN 1998-01-01 AND 1998-06-30



## SELECT with BETWEEN Operator Syntax

`SELECT column1, column2, column3, ....`

`FROM table_name`

`WHERE column_name BETWEEN value1 AND value2;`

`SELECT column1, column2, column3, ....`

`FROM table_name`

`WHERE column_name NOT BETWEEN value1 AND value2;`

## Practical

The screenshot shows the phpMyAdmin interface with the database 'student' selected. In the left sidebar, under the 'Tables' section, the 'personal' table is listed. The main area displays the results of the following SQL query:

```
1 • SELECT * FROM student.personal;
```

The results grid shows the following data:

	id	name	age	gender	phone	city
▶	1	Ram Kumar	19	M	4022155	Agra
▶	2	Sarita Kumari	21	F	4034421	Delhi
▶	3	Salman Khan	20	M	4056221	Agra
▶	4	Juhi Chawla	18	F	4089821	Bhopal
▶	5	Anil Kapoor	22	M	4025221	Agra
▶	6	John Abraham	21	M	4033776	Delhi

The screenshot shows the phpMyAdmin interface with the database 'student' selected. In the left sidebar, under the 'Tables' section, the 'personal' table is listed. The main area displays the results of the following SQL query:

```
1 • SELECT * FROM personal  
2 WHERE age BETWEEN 18 AND 20;
```

The results grid shows the following data, filtering rows where age is between 18 and 20:

	id	name	age	gender	phone	city
▶	1	Ram Kumar	19	M	4022155	Agra
▶	3	Salman Khan	20	M	4056221	Agra
▶	4	Juhi Chawla	18	F	4089821	Bhopal

Navigator: personal

SCHEMAS

Filter objects

student

Tables

personal

Views

Stored Procedures

Functions

test

Administration Schemas

Information

No object selected

1 • SELECT \* FROM personal  
2 WHERE age NOT BETWEEN 18 AND 20;

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content:

	id	name	age	gender	phone	city
▶	2	Sarita Kumari	21	F	4034421	Delhi
▶	5	Anil Kapoor	22	M	4025221	Agra
▶	6	John Abraham	21	M	4033776	Delhi
*	HULL	HULL	HULL	HULL	HULL	HULL

Navigator: personal

SCHEMAS

Filter objects

student

Tables

personal

Views

Stored Procedures

Functions

test

Administration Schemas

Information

No object selected

1 • SELECT \* FROM personal  
2 WHERE name BETWEEN "a" AND "k";

I

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content:

	id	name	age	gender	phone	city
▶	4	Juhi Chawla	18	F	4089821	Bhopal
▶	5	Anil Kapoor	22	M	4025221	Agra
▶	6	John Abraham	21	M	4033776	Delhi
*	HULL	HULL	HULL	HULL	HULL	HULL

same as above

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Navigator' and 'Schemas' sections, with 'personal' selected. The main area shows a query editor with the following SQL code:

```
1 • SELECT * FROM personal
2 WHERE name BETWEEN "anil" AND "kamal";
```

Below the query editor is a 'Result Grid' showing the results of the query. The grid has columns: id, name, age, gender, phone, and city. The data is as follows:

	id	name	age	gender	phone	city
▶	4	Juhi Chawla	18	F	4089821	Bhopal
▶	5	Anil Kapoor	22	M	4025221	Agra
▶	6	John Abraham	21	M	4033776	Delhi
*	HULL	HULL	HULL	HULL	HULL	HULL

No object selected

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Navigator' and 'Schemas' sections, with 'test' selected. The main area shows a query editor with the following SQL code:

```
1 • SELECT * FROM persons
2 WHERE birth_date BETWEEN "1995-01-01" AND "1995-06-30";
```

Below the query editor is a 'Result Grid' showing the results of the query. The grid has columns: id, name, and birth\_date. The data is as follows:

	id	name	birth_date
▶	1	Ram	1995-02-10
▶	4	Shoib	1995-04-21

## 11. MySQL LIKE Operator & Wildcards



## SELECT Data with LIKE Operator

Student Table

Name	Age	Gender
Ram Kumar	19	Male
Salman Khan	18	Male
Meera Khan	19	Female
Sarita Kumari	21	Female
Anil Kapoor	20	Male

WHERE Name LIKE "s%"

Name	Age	Gender
Salman Khan	18	Male
Sarita Kumari	21	Female

WildCard Characters :

% Percentage Sign : Represents zero, one, or multiple characters

\_ Underscore : Represents a single character



## LIKE Operator with Wildcard Patterns

Pattern	Description
LIKE 'a%'	Start with "a"
LIKE '%a'	End with "a"
LIKE '%am%'	Have "am" in any position
LIKE 'a%m'	Start with "a" and Ends with "m"
LIKE '_a%	"a" in the second position
LIKE '__a%	"a" in the third position
LIKE '_oy'	"o" in the second and "y" in the third position



## SELECT with LIKE Operator Syntax

`SELECT column1, column2, column3, ....`

`FROM table_name`

`WHERE column_name LIKE pattern;`

`SELECT column1, column2, column3, ....`

`FROM table_name`

`WHERE column_name NOT LIKE pattern;`

## Practical

The screenshot shows the phpMyAdmin interface for a database named 'student'. The 'personal' table is selected. A query window at the top displays the command: `1 • SELECT * FROM personal;`. Below it, a result grid shows the following data:

	id	name	age	gender	phone	city
1	1	Ram Kumar	19	M	4022155	Agra
2	2	Sarita Kumari	21	F	4034421	Delhi
3	3	Salman Khan	20	M	4056221	Agra
4	4	Juhi Chawla	18	F	4089821	Bhopal
5	5	Anil Kapoor	22	M	4025221	Agra
6	6	John Abraham	21	M	4033776	Delhi

The screenshot shows the phpMyAdmin interface for a database named 'student'. The 'personal' table is selected. Two queries are displayed in the query window: `1 • SELECT * FROM personal` and `2 WHERE name LIKE "S%";`. The second query is highlighted. Below it, a result grid shows the following data:

	id	name	age	gender	phone	city
1	2	Sarita Kumari	21	F	4034421	Delhi
2	3	Salman Khan	20	M	4056221	Agra

**personal**

1 • **SELECT \* FROM personal**  
2 **WHERE name LIKE "a%" ;**

	<b>id</b>	<b>name</b>	<b>age</b>	<b>gender</b>	<b>phone</b>	<b>city</b>
▶	5	Anil Kapoor	22	M	4025221	Agra
*	HULL	HULL	HULL	HULL	HULL	HULL

No object selected

**personal**

1 • **SELECT \* FROM personal**  
2 **WHERE name LIKE "%am%" ;**

	<b>id</b>	<b>name</b>	<b>age</b>	<b>gender</b>	<b>phone</b>	<b>city</b>
▶	1	Ram Kumar	19	M	4022155	Agra
*	6	John Abraham	21	M	4033776	Delhi
*	HULL	HULL	HULL	HULL	HULL	HULL

No object selected

personal

```
1 • SELECT * FROM personal
2 WHERE name LIKE "r%" OR name LIKE "s%";
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	id	name	age	gender	phone	city
▶	1	Ram Kumar	19	M	4022155	Agra
▶	2	Sarita Kumari	21	F	4034421	Delhi
▶	3	Salman Khan	20	M	4056221	Agra

No object selected

personal

```
1 • SELECT * FROM personal
2 WHERE name NOT LIKE "r%";
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	id	name	age	gender	phone	city
▶	2	Sarita Kumari	21	F	4034421	Delhi
▶	3	Salman Khan	20	M	4056221	Agra
▶	4	Juhi Chawla	18	F	4089821	Bhopal
▶	5	Anil Kapoor	22	M	4025221	Agra
▶	6	John Abraham	21	M	4033776	Delhi

personal

```
1 • SELECT * FROM personal
2 WHERE BINARY name LIKE "r%";
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	id	name	age	gender	phone	city
*	HULL	HULL	HULL	HULL	HULL	HULL

No object selected

The screenshot shows two separate sessions in the phpMyAdmin interface.

**Session 1:**

- Query 1: `SELECT * FROM personal`
- Query 2: `WHERE BINARY name LIKE "R%"`

**Result Grid:**

	id	name	age	gender	phone	city
1	1	Ram Kumar	19	M	4022155	Agra
*						

**No object selected**

**Session 2:**

- Query 1: `SELECT * FROM personal`
- Query 2: `WHERE name LIKE "s%n"`

**Result Grid:**

	id	name	age	gender	phone	city
3	3	Salman Khan	20	M	4056221	Agra
*						

**No object selected**

## 12. MySQL Regular Expression



## SELECT Data with Regular Expression

Student Table

Name	Age	Gender
Ram Kumar	19	Male
Salman Khan	18	Male
Meera Khan	19	Female
Sarita Kumari	21	Female
Anil Kapoor	20	Male

WHERE Name REGEXP "khan\$ | poor"

Name	Age	Gender
Salman Khan	18	Male
Meera Khan	19	Female
Anil Kapoor	20	Male



## Regular Expression Patterns with Description

Sign	Pattern	Description
^	'^ra'	Beginning of string
\$	'an\$'	End of string
[...]	'[rms]'	Any character listed between the square brackets
^ [...]	'^[rms]'	Begins with Any character listed between the square brackets
[a-z]	'[a-h]e'	Match with in the range
p1 p2 p3	'tom dick harry'	matches any of the patterns p1, p2, or p3



## SELECT with Regular Expression Syntax

SELECT column1, column2, column3, ....

FROM table\_name

WHERE column\_name REGEXP pattern;

## Practical

The screenshot shows the phpMyAdmin interface. The left sidebar displays the database structure under the 'student' schema, including the 'personal' table. The main query editor window contains the following SQL query:

```
1 • SELECT * FROM student.personal;
```

The results grid shows the following data:

	id	name	age	gender	phone	city
1	1	Ram Kumar	19	M	4022155	Agra
2	2	Sarita Kumari	21	F	4034421	Delhi
3	3	Salman Khan	20	M	4056221	Agra
4	4	Juhi Chawla	18	F	4089821	Bhopal
5	5	Anil Kapoor	22	M	4025221	Agra
6	6	John Abraham	21	M	4033776	Delhi

The screenshot shows the phpMyAdmin interface. The left sidebar displays the database structure under the 'student' schema, including the 'personal' table. The main query editor window contains the following SQL queries:

```
1 • SELECT * FROM personal  
2 WHERE name REGEXP 'ra';
```

The results grid shows the following data:

	id	name	age	gender	phone	city
1	1	Ram Kumar	19	M	4022155	Agra
6	6	John Abraham	21	M	4033776	Delhi

The screenshot shows the phpMyAdmin interface. The left sidebar displays the database structure under the 'student' schema, including the 'personal' table. The main query editor window contains the following SQL queries:

```
1 • SELECT * FROM personal  
2 WHERE name REGEXP 'man';
```

The results grid shows the following data:

	id	name	age	gender	phone	city
3	3	Salman Khan	20	M	4056221	Agra

personal personal

```

1 • SELECT * FROM personal
2 WHERE name REGEXP '^ra';

```

	id	name	age	gender	phone	city
1	1	Ram Kumar	19	M	4022155	Agra
*						

personal personal

```

1 • SELECT * FROM personal
2 WHERE name REGEXP 'an$';

```

	id	name	age	gender	phone	city
3	3	Salman Khan	20	M	4056221	Agra
*						

personal personal

```

1 • SELECT * FROM personal
2 WHERE name REGEXP 'ram|kapoor|khan';

```

	id	name	age	gender	phone	city
1	1	Ram Kumar	19	M	4022155	Agra
3	3	Salman Khan	20	M	4056221	Agra
5	5	Anil Kapoor	22	M	4025221	Agra
*						

no limit for "|" sign we can use as required

Navigator personal personal

SCHEMAS Filter objects

- phpmyadmin
- student**
- Tables personal
- Views
- Stored Procedures
- Functions

test

Administration Schemas Information

No object selected

1 • **SELECT \* FROM personal**

2 WHERE name REGEXP '^ram|poor|khan\$';

Result Grid | Filter Rows: [ ] Edit: Export/Import: Wrap Cell Content:

	<b>id</b>	<b>name</b>	<b>age</b>	<b>gender</b>	<b>phone</b>	<b>city</b>
▶	1	Ram Kumar	19	M	4022155	Agra
▶	3	Salman Khan	20	M	4056221	Agra
▶	5	Anil Kapoor	22	M	4025221	Agra
*	NULL	NULL	NULL	NULL	NULL	NULL

Navigator personal personal

SCHEMAS Filter objects

- phpmyadmin
- student**
- Tables personal
- Views
- Stored Procedures
- Functions

test

Administration Schemas Information

No object selected

1 • **SELECT \* FROM personal**

2 WHERE name REGEXP '[is]';

Result Grid | Filter Rows: [ ] Edit: Export/Import: Wrap Cell Content:

	<b>id</b>	<b>name</b>	<b>age</b>	<b>gender</b>	<b>phone</b>	<b>city</b>
▶	2	Sarita Kumari	21	F	4034421	Delhi
▶	3	Salman Khan	20	M	4056221	Agra
▶	4	Juh Chawla	18	F	4089821	Bhopal
▶	5	Anil Kapoor	22	M	4025221	Agra
*	NULL	NULL	NULL	NULL	NULL	NULL

Navigator personal personal

SCHEMAS

Filter objects

phpmyadmin

student

Tables

personal

Views

Stored Procedures

Functions

test

Limit to 1000 rows

1 • **SELECT \* FROM personal**

2 WHERE name REGEXP '[rm]a';

3

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content:

	id	name	age	gender	phone	city
▶	1	Ram Kumar	19	M	4022155	Agra
▶	2	Sarita Kumari	21	F	4034421	Delhi
▶	3	Salman Khan	20	M	4056221	Agra
▶	6	John Abraham	21	M	4033776	Delhi
	HULL	HULL	HULL	HULL	HULL	HULL

No object selected

Navigator personal personal

SCHEMAS

Filter objects

phpmyadmin

student

Tables

personal

Views

Stored Procedures

Functions

test

Limit to 1000 rows

1 • **SELECT \* FROM personal**

2 WHERE name REGEXP '[rmh]a';

3

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content:

	id	name	age	gender	phone	city
▶	1	Ram Kumar	19	M	4022155	Agra
▶	2	Sarita Kumari	21	F	4034421	Delhi
▶	3	Salman Khan	20	M	4056221	Agra
▶	4	Juhi Chawla	18	F	4089821	Bhopal
▶	6	John Abraham	21	M	4033776	Delhi
	HULL	HULL	HULL	HULL	HULL	HULL

No object selected

Navigator personal personal

SCHEMAS

Filter objects

phpmyadmin student

Tables personal

Views

Stored Procedures

Functions

test

Limit to 1000 rc ▾

1 • **SELECT \* FROM personal**

2 WHERE name REGEXP '^[\r\n]';

3

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content:

	id	name	age	gender	phone	city
▶	1	Ram Kumar	19	M	4022155	Agra
▶	2	Sarita Kumari	21	F	4034421	Delhi
▶	3	Salman Khan	20	M	4056221	Agra

No object selected

Navigator personal personal

SCHEMAS

Filter objects

phpmyadmin student

Tables personal

Views

Stored Procedures

Functions

test

Limit to 1000 rc ▾

1 • **SELECT \* FROM personal**

2 WHERE name REGEXP 'r[am]';

3

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content:

	id	name	age	gender	phone	city
▶	1	Ram Kumar	19	M	4022155	Agra
▶	6	John Abraham	21	M	4033776	Delhi

Navigator personal personal

SCHEMAS

Filter objects

phpmyadmin student

Tables personal

Views

Stored Procedures

Functions

test

Limit to 1000 rc ▾

1 • **SELECT \* FROM personal**

2 WHERE name REGEXP '[a-j]r';

3

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content:

	id	name	age	gender	phone	city
▶	1	Ram Kumar	19	M	4022155	Agra
▶	2	Sarita Kumari	21	F	4034421	Delhi
▶	6	John Abraham	21	M	4033776	Delhi

No object selected

# 13. MySQL ORDER BY & DISTINCT



## SELECT Data with ORDER BY

Student Table

Name	Age	Gender
Ram Kumar	19	Male
Salman Khan	18	Male
Meera Khan	19	Female
Sarita Kumari	21	Female
Anil Kapoor	20	Male

Ascending Order

Name	Age	Gender
Anil Kapoor	20	Male
Meera Khan	19	Female
Ram Kumar	19	Male
Salman Khan	18	Male
Sarita Kumari	21	Female

ORDER BY Name ASC

ORDER BY Name DESC



## SELECT with ORDER BY Syntax

SELECT column1, column2, column3, ....

FROM table\_name

ORDER BY column1, column2, .... ASC | DESC;

↑

## Practical

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator personal personal

SCHEMAS

personal

Tables personal Views Stored Procedures Functions

test

1 • `SELECT * FROM student.personal;`

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Contents: |

	<b>id</b>	<b>name</b>	<b>age</b>	<b>gender</b>	<b>phone</b>	<b>city</b>
▶	1	Ram Kumar	19	M	4022155	Agra
2	Sarita Kumari	21	F	4034421	Delhi	
3	Salman Khan	20	M	4056221	Agra	
4	Juhi Chawla	18	F	4089821	Bhopal	
5	Anil Kapoor	22	M	4025221	Agra	
6	John Abraham	21	M	4033776	Delhi	

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator personal personal

SCHEMAS

personal

Tables personal Views Stored Procedures Functions

test

1 • `SELECT * FROM personal`

2 • `ORDER BY name;`

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Contents: |

	<b>id</b>	<b>name</b>	<b>age</b>	<b>gender</b>	<b>phone</b>	<b>city</b>
▶	5	Anil Kapoor	22	M	4025221	Agra
6	John Abraham	21	M	4033776	Delhi	
4	Juhi Chawla	18	F	4089821	Bhopal	
1	Ram Kumar	19	M	4022155	Agra	
3	Salman Khan	20	M	4056221	Agra	
2	Sarita Kumari	21	F	4034421	Delhi	

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator personal personal

SCHEMAS

personal

1 • SELECT \* FROM personal  
2 ORDER BY name DESC;

	id	name	age	gender	phone	city
▶	2	Sarita Kumari	21	F	4034421	Delhi
3	Salman Khan	20	M	4056221	Agra	
1	Ram Kumar	19	M	4022155	Agra	
4	Juhi Chawla	18	F	4089821	Bhopal	
6	John Abraham	21	M	4033776	Delhi	
5	Anil Kapoor	22	M	4025221	Agra	

ascending and descending with condition

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator personal personal

SCHEMAS

personal

1 • SELECT \* FROM personal  
2 WHERE city = "Agra"  
3 ORDER BY name DESC;

	id	name	age	gender	phone	city
▶	3	Salman Khan	20	M	4056221	Agra
1	Ram Kumar	19	M	4022155	Agra	
5	Anil Kapoor	22	M	4025221	Agra	

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator personal personal

SCHEMAS Filter objects

- phpmyadmin
- student
  - Tables
    - personal
  - Views
  - Stored Procedures
  - Functions
- test

Administration Schemas Information

No object selected

```
1 • SELECT * FROM personal
2 WHERE city = "Agra"
3 ORDER BY name;
```

Result Grid | Filter Rows | Edit: Export/Import: Wrap Cell Content:

	id	name	age	gender	phone	city
5	Anil Kapoor	22	M	4025221	Agra	
1	Ram Kumar	19	M	4022155	Agra	
3	Salman Khan	20	M	4056221	Agra	

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator personal personal

SCHEMAS Filter objects

- phpmyadmin
- student
  - Tables
    - personal
  - Views
  - Stored Procedures
  - Functions
- test

Administration Schemas Information

No object selected

```
1 • SELECT * FROM personal
2 ORDER BY age;
```

Result Grid | Filter Rows | Edit: Export/Import: Wrap Cell Content:

	id	name	age	gender	phone	city
4	Juhi Chawla	18	F	4089821	Bhopal	
1	Ram Kumar	19	M	4022155	Agra	
3	Salman Khan	20	M	4056221	Agra	
2	Sarita Kumari	21	F	4034421	Delhi	
6	John Abraham	21	M	4033776	Delhi	
5	Anil Kapoor	22	M	4025221	Agra	

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator personal personal

SCHEMAS Filter objects

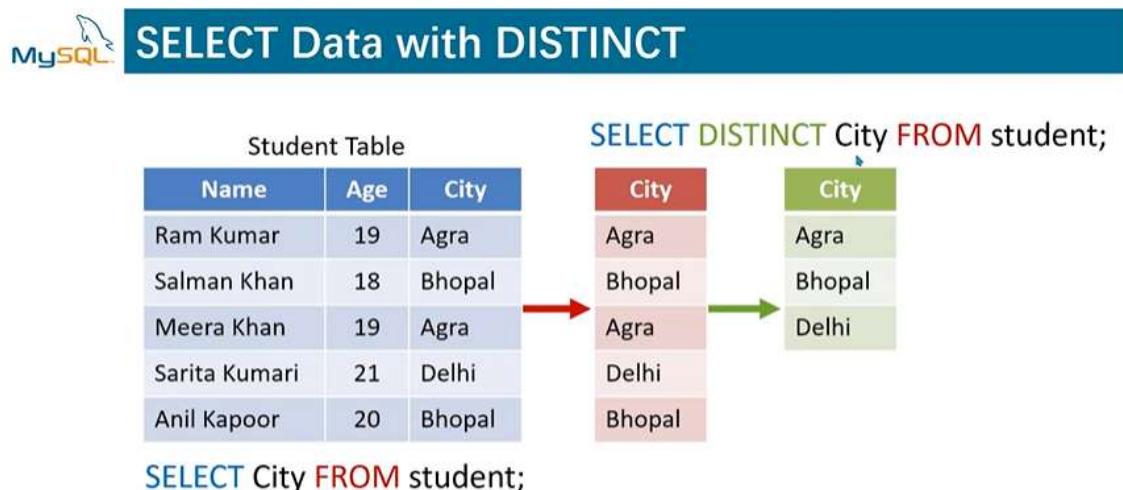
student Tables personal Views Stored Procedures Functions

personal

1 • `SELECT * FROM personal`  
2 `ORDER BY name,city;`

	id	name	age	gender	phone	city
5	Anil Kapoor	22	M	4025221	Agra	
6	John Abraham	21	M	4033776	Delhi	
4	Juhi Chawla	18	F	4089821	Bhopal	
1	Ram Kumar	19	M	4022155	Agra	
3	Salman Khan	20	M	4056221	Agra	
2	Sarita Kumari	21	F	4034421	Delhi	

Ordering with distinct(unique)



**MySQL** SELECT with DISTINCT Syntax

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

Navigator personal personal

SCHEMAS Filter objects

- phpmyadmin
- student
  - Tables
    - personal
  - Views
  - Stored Procedures
  - Functions
- test

Administration Schemas Information No object selected

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

1 • `SELECT * FROM personal;`

	id	name	age	gender	phone	city
>	1	Ram Kumar	19	M	4022155	Agra
>	2	Sarita Kumari	21	F	4034421	Delhi
>	3	Salman Khan	20	M	4056221	Agra
>	4	Juhi Chawla	18	F	4089821	Bhopal
>	5	Anil Kapoor	22	M	4025221	Agra
>	6	John Abraham	21	M	4033776	Delhi
*	HULL	HULL	HULL	HULL	HULL	HULL

Navigator personal personal

SCHEMAS Filter objects

- phpmyadmin
- student
  - Tables
    - personal
  - Views
  - Stored Procedures
  - Functions
- test

Administration Schemas Information No object selected

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

1 • `SELECT DISTINCT city FROM personal;`

city
> Agra
> Delhi
> Bhopal

Navigator personal personal

SCHEMAS Filter objects

- phpmyadmin
- student
  - Tables
    - personal
  - Views
  - Stored Procedures
  - Functions
- test

Administration Schemas Information No object selected

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

1 • `SELECT DISTINCT age FROM personal;`

age
> 19
> 21
> 20
> 18
> 22

distinct with order

The screenshot shows the phpMyAdmin interface. In the top navigation bar, the schema is set to 'personal'. Below it, the query '1 • SELECT DISTINCT age FROM personal ORDER BY age;' is displayed. The results are shown in a grid:

age
18
19
20
21
22

The 'age' column header is highlighted with a blue cursor. The results are ordered by age.

## 14. MySQL IS NULL & IS NOT NULL



## SELECT Data with IS NULL

Student Table

Name	Age	Gender
Ram Kumar	19	Male
Salman Khan	18	Male
Meera Khan		Female
Sarita Kumari	21	Female
Anil Kapoor	20	Male

WHERE Age IS NULL

Name	Age	Gender
Meera Khan		Female



## SELECT with IS NULL Syntax

**SELECT** column1, column2, column3, ....

**FROM** table\_name

**WHERE** column **IS NULL**;

**SELECT** column1, column2, column3, ....

**FROM** table\_name

**WHERE** column **IS NOT NULL**;

practical

personal persons

```
1 • SELECT * FROM test.persons;
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

	id	name	birth_date
1	1	Ram	1995-02-10
2	2	Madan	1995-11-03
3	3	Salman	1996-06-12
4	4	Shoib	1995-04-21
5	5	Juhi	1996-09-25
6	6	Raman	NULL
7	7		1996-08-10

personal persons

```
1 • SELECT * FROM persons  
2 WHERE birth_date IS NULL;
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

	id	name	birth_date
6	6	Raman	NULL

The screenshot shows the phpMyAdmin interface. The left sidebar displays the 'personal' schema with a single 'persons' table selected. The main area shows the results of the following SQL query:

```
1 • SELECT * FROM persons
2 WHERE name IS NULL;
```

The result grid displays one row of data:

	id	name	birth_date
▶	7	NULL	1996-08-10

The screenshot shows the phpMyAdmin interface. The left sidebar displays the 'personal' schema with a single 'persons' table selected. The main area shows the results of the following SQL query:

```
1 • SELECT * FROM persons
2 WHERE name IS NOT NULL;
```

The result grid displays six rows of data:

	id	name	birth_date
▶	1	Ram	1995-02-10
▶	2	Madan	1995-11-03
▶	3	Salman	1996-06-12
▶	4	Shoib	1995-04-21
▶	5	Juhí	1996-09-25
▶	6	Raman	NULL

## 15. MySQL LIMIT & OFFSET



## SELECT Data with LIMIT

Student Table

Name	Age	Gender
Ram Kumar	19	Male
Salman Khan	18	Male
Meera Khan	20	Female
Sarita Kumari	21	Female
Anil Kapoor	20	Male
Shahid Kapoor	19	Male
Virat Kohli	21	Male

LIMIT 3

Name	Age	Gender
Ram Kumar	19	Male
Salman Khan	18	Male
Meera Khan	20	Female

`SELECT * FROM student;`



## SELECT with LIMIT Syntax

`SELECT column1, column2, column3, ....`

`FROM table_name`

`WHERE condition`

`LIMIT number;`

## Practical

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator personal personal

SCHEMAS Filter objects

- phpmyadmin
- student
  - Tables
    - personal
  - Views
  - Stored Procedures
  - Functions
- test

Administration Schemas Information

No object selected

1 • `SELECT * FROM student.personal;`

	id	name	age	gender	phone	city
1	1	Ram Kumar	19	M	4022155	Agra
2	2	Sarita Kumari	21	F	4034421	Delhi
3	3	Salman Khan	20	M	4056221	Agra
4	4	Juhi Chawla	18	F	4089821	Bhopal
5	5	Anil Kapoor	22	M	4025221	Agra
6	6	John Abraham	21	M	4033776	Delhi
7	7	Shahid Kapoor	20		4022784	Agra

File Edit View Query Database Server Tools Scripting Help

Navigator personal personal

SCHEMAS Filter objects

- phpmyadmin
- student
  - Tables
    - personal
  - Views
  - Stored Procedures
  - Functions
- test

Administration Schemas Information

No object selected

1 • `SELECT * FROM personal`

2 • `LIMIT 3;`

	id	name	age	gender	phone	city
1	1	Ram Kumar	19	M	4022155	Agra
2	2	Sarita Kumari	21	F	4034421	Delhi
3	3	Salman Khan	20	M	4056221	Agra

The screenshot shows the phpMyAdmin interface with the 'personal' database selected. In the left sidebar, under 'SCHEMAS', the 'student' schema is expanded, showing the 'Tables' section which includes 'personal'. The main query editor window contains the following SQL code:

```
1 • SELECT * FROM personal
2 WHERE city = "Agra"
3 LIMIT 3;
```

The results grid displays three rows of data from the 'personal' table where the 'city' column is 'Agra':

	id	name	age	gender	phone	city
▶	1	Ram Kumar	19	M	4022155	Agra
▶	3	Salman Khan	20	M	4056221	Agra
▶	5	Anil Kapoor	22	M	4025221	Agra

No object selected

The screenshot shows the phpMyAdmin interface with the 'personal' database selected. In the left sidebar, under 'SCHEMAS', the 'student' schema is expanded, showing the 'Tables' section which includes 'personal'. The main query editor window contains the following SQL code:

```
1 • SELECT * FROM personal
2 WHERE city = "Agra"
3 ORDER BY name
4 LIMIT 3;
```

The results grid displays three rows of data from the 'personal' table where the 'city' column is 'Agra', ordered by 'name':

	id	name	age	gender	phone	city
▶	5	Anil Kapoor	22	M	4025221	Agra
▶	1	Ram Kumar	19	M	4022155	Agra
▶	3	Salman Khan	20	M	4056221	Agra

No object selected

limit with offset



## SELECT Data with LIMIT & OFFSET

Student Table

Name	Age	Gender
Ram Kumar	19	Male
Salman Khan	18	Male
Meera Khan	20	Female
Sarita Kumari	21	Female
Anil Kapoor	20	Male
Shahid Kapoor	19	Male
Virat Kohli	21	Male

**LIMIT 3**

Name	Age	Gender
Ram Kumar	19	Male
Salman Khan	18	Male
Meera Khan	20	Female

**LIMIT 3, 3**

↑      ↘

OFFSET    LIMIT NUMBER



## SELECT with LIMIT & OFFSET Syntax

```

SELECT column1, column2, column3, ....
FROM table_name
WHERE condition
LIMIT offset , number;
  
```

The screenshot shows the phpMyAdmin interface. In the left sidebar, under the 'personal' schema, the 'Tables' section is expanded, showing the 'personal' table. The main query editor window contains the following SQL code:

```

1 • SELECT * FROM personal
2   LIMIT 3, 3;
  
```

Below the query, the 'Result Grid' displays the following data:

id	name	age	gender	phone	city
4	Juhi Chawla	18	F	4089821	Bhopal
5	Anil Kapoor	22	M	4025221	Agra
6	John Abraham	21	M	4033776	Delhi

The status bar at the bottom left indicates "No object selected".

The screenshot shows the phpMyAdmin interface. In the top navigation bar, the schema is set to 'personal'. Below it, the 'Tables' section of the 'student' schema is selected, showing a table named 'personal'. A query is being run in the SQL tab:

```
1 • SELECT * FROM personal  
2 LIMIT 6, 3;
```

The results are displayed in a grid:

	id	name	age	gender	phone	city
▶	7	Shahid Kapoor	20	HULL	4022784	Agra
*	HULL	HULL	HULL	HULL	HULL	HULL

No object selected

The screenshot shows the phpMyAdmin interface. In the top navigation bar, the schema is set to 'personal'. Below it, the 'Tables' section of the 'student' schema is selected, showing a table named 'personal'. A query is being run in the SQL tab:

```
1 • SELECT * FROM personal  
2 LIMIT 0, 3;
```

The results are displayed in a grid:

	id	name	age	gender	phone	city
▶	1	Ram Kumar	19	M	4022155	Agra
2	Sarita Kumari	21	F	4034421	Delhi	
3	Salman Khan	20	M	4056221	Agra	
*	HULL	HULL	HULL	HULL	HULL	HULL

No object selected

## 16. MySQL Count, Sum, Min, Max, Avg



## SELECT Data with Aggregate Functions

Employee Table

Name	Age	Gender	Salary
Ram Kumar	19	Male	4500
Salman Khan	18	Male	5200
Meera Khan	20	Female	6000
Sarita Kumari	21	Female	8500
Anil Kapoor	20	Male	6300
Shahid Kapoor	19	Male	4800
Virat Kohli	21	Male	5700

COUNT(column\_name)

MAX(column\_name)

MIN(column\_name)

SUM(column\_name)

AVG(column\_name)



## SELECT with Aggregate Functions Syntax

SELECT COUNT(column\_name)

FROM table\_name

WHERE condition;

SELECT SUM(column\_name)

FROM table\_name

WHERE condition;

## Practical

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

personal personal

SCHEMAS Filter objects

phpmyadmin student

Tables personal Views Stored Procedures Functions

test

1 • `SELECT * FROM student.personal;`

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	id	name	percentage	age	gender	phone	city
1	1	Ram Kumar	45	19	M	4022155	Agra
2	2	Sarita Kumari	56	21	F	4034421	Delhi
3	3	Salman Khan	62	20	M	4056221	Agra
4	4	Jahi Chawla	47	18	F	4089821	Bhopal
5	5	Anil Kapoor	74	22	M	4025221	Agra
6	6	John Abraham	64	21	M	4033776	Delhi
7	7	Shahid Kapoor	52	20		4022784	Agra

File Edit View Query Database Server Tools Scripting Help

personal personal

SCHEMAS Filter objects

phpmyadmin student

Tables personal Views Stored Procedures Functions

test

1 • `SELECT COUNT(*) FROM personal;`

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

COUNT(*)
7

File Edit View Query Database Server Tools Scripting Help

personal personal

SCHEMAS Filter objects

phpmyadmin student

Tables personal Views Stored Procedures Functions

test

1 • `SELECT COUNT(DISTINCT city) FROM personal;`

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

COUNT(DISTINCT city)
3

Navigator personal personal

SCHEMAS  
Filter objects  
student  
Tables personal Views Stored Procedures Functions  
test

1 • `SELECT COUNT(DISTINCT city) AS Count FROM personal;`

Result Grid Filter Rows: Export: Wrap Cell Content:

Count
3

Administration Schemas

This screenshot shows the phpMyAdmin interface for a database named 'personal'. In the left sidebar, under the 'Tables' section, there is a table named 'personal'. The main query window displays the SQL command 'SELECT COUNT(DISTINCT city) AS Count FROM personal;'. The result grid shows a single row with the column 'Count' containing the value '3'. The 'Administration' tab is selected at the bottom.

Navigator personal personal

SCHEMAS  
Filter objects  
student  
Tables personal Views Stored Procedures Functions  
test

1 • `SELECT MAX(percentage) AS Percentage FROM personal;`

Result Grid Filter Rows: Export: Wrap Cell Content:

Percentage
74

This screenshot shows the phpMyAdmin interface for a database named 'personal'. In the left sidebar, under the 'Tables' section, there is a table named 'personal'. The main query window displays the SQL command 'SELECT MAX(percentage) AS Percentage FROM personal;'. The result grid shows a single row with the column 'Percentage' containing the value '74'. The 'Administration' tab is selected at the bottom.

Navigator personal personal

SCHEMAS  
Filter objects  
student  
Tables personal Views Stored Procedures Functions  
test

1 • `SELECT MIN(percentage) AS Percentage FROM personal;`

Result Grid Filter Rows: Export: Wrap Cell Content:

Percentage
45

This screenshot shows the phpMyAdmin interface for a database named 'personal'. In the left sidebar, under the 'Tables' section, there is a table named 'personal'. The main query window displays the SQL command 'SELECT MIN(percentage) AS Percentage FROM personal;'. The result grid shows a single row with the column 'Percentage' containing the value '45'. The 'Administration' tab is selected at the bottom.

The image shows two screenshots of MySQL Workbench. Both screenshots have a top navigation bar with File, Edit, View, Query, Database, Server, Tools, Scripting, and Help menus. The left sidebar is titled 'Navigator' and shows 'SCHEMAS' with 'personal' selected. Below it, 'Tables' are listed: 'personal' (selected), 'Views', 'Stored Procedures', and 'Functions'. The right pane contains a query editor and a result grid.

**Screenshot 1:**

```
1 • SELECT MIN(percentage) AS Percentage, name, city FROM personal;
```

Result Grid:

Percentage	name	city
45	Ram Kumar	Agra

**Screenshot 2:**

```
1 • SELECT AVG(percentage) AS Average FROM personal;
```

Result Grid:

Average
57.1429

## 17. MySQL UPDATE



## How to update data in Tables with SQL ?

Employee Table			
Name	Age	Gender	Salary
Ram Kumar	19	Male	4500
Salman Khan	18	Male	5200
Meera Khan	20	Female	6000
Sarita Kumari	21	Female	8500
Anil Kapoor	20	Male	6300

### UPDATE Command



## UPDATE Syntax

UPDATE *table\_name*

SET *column1\_name* = *value1*, *column2\_name* = *value2*, ...

WHERE *condition*;

### Practical

The screenshot shows the phpMyAdmin interface. The left sidebar displays the database structure with 'student' selected. The main area shows the results of the query 'SELECT \* FROM student.personal;'. The result grid contains the following data:

	<i>id</i>	<i>name</i>	<i>percentage</i>	<i>age</i>	<i>gender</i>	<i>phone</i>	<i>city</i>
1	Ram Kumar	45	19	M	4022144	Agra	
2	Sarita Kumari	55	22	F	4034421	Delhi	
3	Salman Khan	62	20	M	4056221	Agra	
4	Juhি Chawla	47	18	F	4089821	Bhopal	
5	Anil Kapoor	74	22	M	4025221	Agra	
6	John Abraham	64	21	M	4033776	Delhi	
7	Shahid Kapoor	52	20		4022784	Agra	

The screenshot shows the phpMyAdmin interface with the database 'personal' selected. In the left sidebar, under 'SCHEMAS', the 'student' schema is expanded, showing its tables, views, stored procedures, and functions. The 'Tables' section contains 'personal'. The main query editor window displays the following SQL code:

```
1 UPDATE personal
2 SET phone = "4055555"
3 WHERE id = 1;
```

The screenshot shows the phpMyAdmin interface with the database 'personal' selected. In the left sidebar, under 'SCHEMAS', the 'student' schema is expanded, showing its tables, views, stored procedures, and functions. The 'Tables' section contains 'personal'. The main query editor window displays the following SQL code:

```
1 • SELECT * FROM personal;
```

Below the query editor is a 'Result Grid' table with the following data:

	id	name	percentage	age	gender	phone	city
1	1	Ram Kumar	45	19	M	4055555	Agra
2	2	Sarita Kumari	55	22	F	4034421	Delhi
3	3	Salman Khan	62	20	M	4056221	Agra
4	4	Juhi Chawla	47	18	F	4089821	Bhopal
5	5	Anil Kapoor	74	22	M	4025221	Agra
6	6	John Abraham	64	21	M	4033776	Delhi
7	7	Shahid Kapoor	52	20		4022784	Agra

The screenshot shows the phpMyAdmin interface with the database 'personal' selected. In the left sidebar, under 'SCHEMAS', the 'student' schema is expanded, showing its tables, views, stored procedures, and functions. The 'Tables' section contains 'personal'. The main query editor window displays the following SQL code:

```
1 UPDATE personal
2 SET percentage = 66
3 WHERE id = 2;
```

personal personal

1 • `SELECT * FROM personal;`

	<b>id</b>	<b>name</b>	<b>percentage</b>	<b>age</b>	<b>gender</b>	<b>phone</b>	<b>city</b>
▶	1	Ram Kumar	45	19	M	4055555	Agra
	2	Sarita Kumari	66	22	F	4034421	Delhi
	3	Salman Khan	62	20	M	4056221	Agra
	4	Juhi Chawla	47	18	F	4089821	Bhopal
	5	Anil Kapoor	74	22	M	4025221	Agra
	6	John Abraham	64	21	M	4033776	Delhi
	7	Shahid Kapoor	52	20		4022784	Agra

update multiple columns

personal personal

1 `UPDATE personal`

2 `SET percentage = 55, age = 21`

3 `WHERE id = 2;`

personal personal

1 • `SELECT * FROM personal;`

	<b>id</b>	<b>name</b>	<b>percentage</b>	<b>age</b>	<b>gender</b>	<b>phone</b>	<b>city</b>
▶	1	Ram Kumar	45	19	M	4055555	Agra
	2	Sarita Kumari	55	21	F	4034421	Delhi
	3	Salman Khan	62	20	M	4056221	Agra
	4	Juhi Chawla	47	18	F	4089821	Bhopal
	5	Anil Kapoor	74	22	M	4025221	Agra
	6	John Abraham	64	21	M	4033776	Delhi
	7	Shahid Kapoor	52	20		4022784	Agra

update multiple rows

The screenshot shows the phpMyAdmin interface with the following details:

- Navigator:** Shows the database structure with the **student** schema selected.
- Query Editor:** Displays the following SQL code:
  - 1 • UPDATE personal
  - 2 SET age = 18
  - 3 WHERE id IN (2,3);
- Result Grid:** Shows the **personal** table with the following data:

	id	name	percentage	age	gender	phone	city
1	1	Ram Kumar	45	19	M	4055555	Agra
2	2	Sarita Kumari	55	18	F	4034421	Delhi
3	3	Salman Khan	62	18	M	4056221	Agra
4	4	Juhi Chawla	47	18	F	4089821	Bhopal
5	5	Anil Kapoor	74	22	M	4025221	Agra
6	6	John Abraham	64	21	M	4033776	Delhi
7	7	Shahid Kapoor	52	20		4022784	Agra

update without where clause

The screenshot shows the phpMyAdmin interface for a database named 'personal'. In the top query window, the following SQL statements are entered:

```
1 • UPDATE personal  
2   SET age = 19;
```

In the bottom query window, the following SQL statement is entered:

```
1 • SELECT * FROM personal;
```

The result grid displays the following data:

	id	name	percentage	age	gender	phone	city
1	1	Ram Kumar	45	19	M	4055555	Agra
2	2	Sarita Kumari	55	19	F	4034421	Delhi
3	3	Salman Khan	62	19	M	4056221	Agra
4	4	Juhi Chawla	47	19	F	4089821	Bhopal
5	5	Anil Kapoor	74	19	M	4025221	Agra
6	6	John Abraham	64	19	M	4033776	Delhi
7	7	Shahid Kapoor	52	19		4022784	Agra

## 18. MySQL COMMIT & ROLLBACK



## How to Rollback your work in MySQL ?

Employee Table

Id	Name	Age	Salary
1	Ram Kumar	19	4500
2	Salman Khan	18	5200
3	Sarita Kumari	21	8500
4	Anil Kapoor	20	6300

UPDATE employee  
SET Salary = 6000  
WHERE Id = 2;



option-1 with same command with changes option-2 rollback



## How to Rollback your work in MySQL ?

Employee Table

Id	Name	Age	Salary
1	Ram Kumar	19	4500
2	Salman Khan	18	5200
3	Sarita Kumari	21	8500
4	Anil Kapoor	20	6300

UPDATE employee  
SET Salary = 6000  
WHERE Id = 2;



**ROLLBACK;**

will rollback previous commands



## How to Rollback your work in MySQL ?

Employee Table

Id	Name	Age	Salary
1	Ram Kumar	19	4500
2	Salman Khan	18	5200
3	Sarita Kumari	21	8500
4	Anil Kapoor	20	6300

UPDATE employee  
SET Age = 22  
WHERE Id = 3;

UPDATE employee  
SET Salary = 6000  
WHERE Id = 2;



**ROLLBACK;**

so for that we have solution "commit" command



## How to Rollback your work in MySQL ?

Employee Table			
Id	Name	Age	Salary
1	Ram Kumar	19	4500
2	Salman Khan	18	5200
3	Sarita Kumari	21	8500
4	Anil Kapoor	20	6300

UPDATE employee

SET Age = 22

WHERE Id = 3;

**COMMIT;**

UPDATE employee

SET Salary = 6000

WHERE Id = 2;



**ROLLBACK;**



## COMMIT & ROLLBACK Works for :

- INSERT
- UPDATE
- DELETE

## Practical

The screenshot shows the MySQL Workbench interface. In the top-left, the Navigator pane displays the schema 'student' with its tables ('personal', 'Views', 'Stored Procedures', 'Functions'). The top-right pane contains a query editor with the SQL command: `1 • SELECT * FROM student.personal;`. Below the query is a result grid showing the following data:

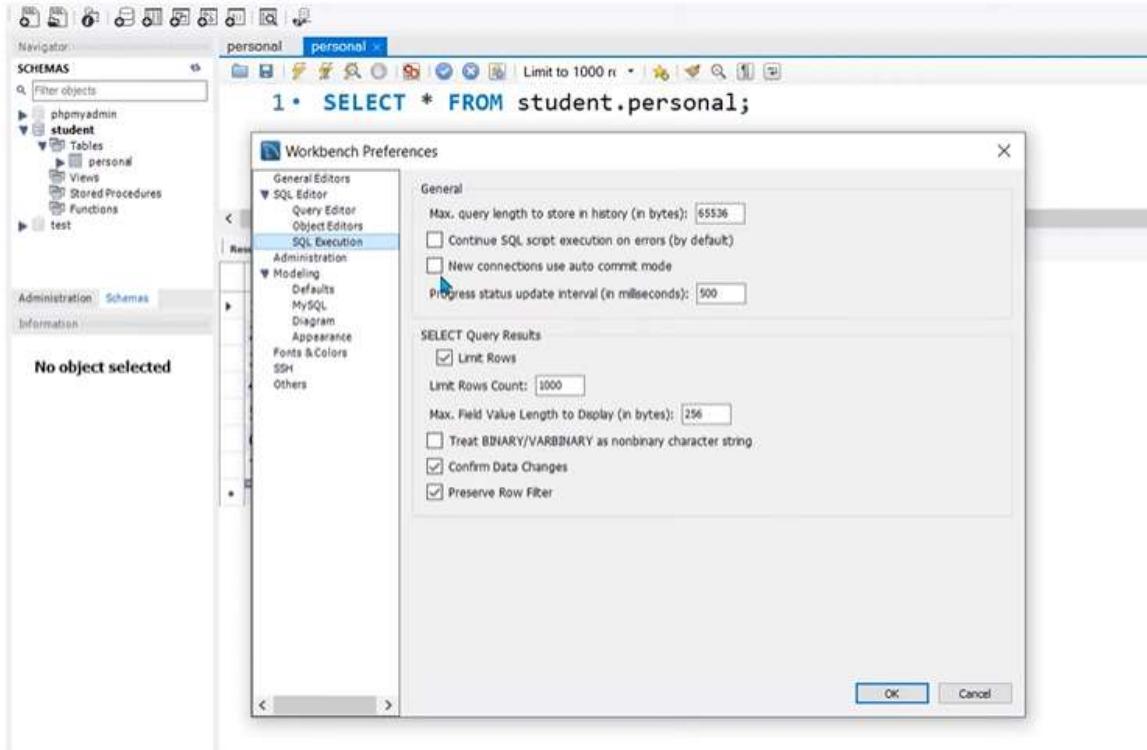
	id	name	percentage	age	gender	phone	city
1	1	Ram Kumar	45	19	M	4022144	Agra
2	2	Sarita Kumari	55	22	F	4027895	Delhi
3	3	Salman Khan	62	20	M	4022254	Agra
4	4	Juhi Chawla	47	18	F	4028524	Bhopal
5	5	Anil Kapoor	74	22	M	4027896	Agra
6	6	John Abraham	64	21	M	4022034	Delhi
7	7	Shahid Kapoor	52	20	M	4021469	Agra

Before start for commit and rollback we need make changes in mysql setting

The screenshot shows the MySQL Workbench interface with the 'Edit' menu open. The 'Preferences...' option is highlighted with a blue selection bar. To the right, a preview of the preferences window is shown, containing the following text:  
\* FROM S1  
percentage

At the bottom right of the main window, there is a note: "unchecked new connection".

use auto commit mode then restart mysql workbench



commit for we dont want changes in previous commands



Navigator personal personal

SCHEMAS Filter objects

phpmyadmin student Tables personal Views Stored Procedures Functions test

Administration Schemas Information

No object selected

```
1 • SELECT * FROM personal;
2
3 • COMMIT;
4
5 • UPDATE personal SET percentage = 60
6 WHERE id = 2;
```

Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:

	id	name	percentage	age	gender	phone	city
1	Ram Kumar	45	19	M	4022144	Agra	
2	Sarita Kumari	60	22	F	4027895	Delhi	
3	Salman Khan	62	20	M	4022254	Agra	
4	Juhি Chawla	47	18	F	4028524	Bhopal	
5	Anil Kapoor	74	22	M	4027896	Agra	
6	John Abraham	64	21	M	4022034	Delhi	
7	Shahid Kapoor	52	20	M	4021469	Agra	

rollback

Navigator personal personal

SCHEMAS Filter objects

phpmyadmin student Tables personal Views Stored Procedures Functions test

Administration Schemas Information

No object selected

```
1 • SELECT * FROM personal;
2
3 • COMMIT;
4
5 • UPDATE personal SET percentage = 60
6 WHERE id = 2;
7
8 • ROLLBACK;
```

personal personal

Limit to 1000 rows

```
1 • SELECT * FROM personal;
2
3 • COMMIT;
4
5 • UPDATE personal SET percentage = 60
6 WHERE id = 2;
7
8 • ROLLBACK;
```

Result Grid | Filter Rows | Edit: Export/Import: Wrap Cell Content:

	id	name	percentage	age	gender	phone	city
▶	1	Ram Kumar	45	19	M	4022144	Agra
	2	Sarita Kumari	55	22	F	4027895	Delhi
	3	Salman Khan	62	20	M	4022254	Agra
	4	Juhi Chawla	47	18	F	4028524	Bhopal
	5	Anil Kapoor	74	22	M	4027896	Agra
	6	John Abraham	64	21	M	4022034	Delhi
	7	Shahid Kapoor	52	20	M	4021469	Agra

personal personal

1 • `SELECT * FROM personal;`

2

3 • `COMMIT;`

4

5 • `UPDATE personal SET age = 20`

6 `WHERE id = 4;`

7

8 • `UPDATE personal SET percentage = 60`

9 `WHERE id = 2;`

10

11 • `ROLLBACK;`

No object selected

	id	name	percentage	age	gender	phone	city
1	1	Ram Kumar	45	19	M	4022144	Agra
2	2	Sarita Kumari	60	22	F	4027895	Delhi
3	3	Salman Khan	62	20	M	4022254	Agra
4	4	Juhি Chawla	47	20	F	4028524	Bhopal
5	5	Anil Kapoor	74	22	M	4027896	Agra
6	6	John Abraham	64	21	M	4022034	Delhi
7	7	Shahid Kapoor	52	20	M	4021469	Agra

Navigator personal personal

SCHEMAS Filter objects

phpmyadmin student

Tables personal Views Stored Procedures Functions

test

No object selected

```
1 • SELECT * FROM personal;
2
3 • COMMIT;
4
5 • UPDATE personal SET age = 20
6 WHERE id = 4;
7
8 • UPDATE personal SET percentage = 60
9 WHERE id = 2;
10
11 • ROLLBACK;
```

Result Grid Filter Rows Edit Export/Import Wrap Cell Content

	id	name	percentage	age	gender	phone	city
1	1	Ram Kumar	45	19	M	4022144	Agra
2	2	Sarita Kumari	55	22	F	4027895	Delhi
3	3	Salman Khan	62	20	M	4022254	Agra
4	4	Juhি Chawla	47	18	F	4028524	Bhopal
5	5	Anil Kapoor	74	22	M	4027896	Agra
6	6	John Abraham	64	21	M	4022034	Delhi
7	7	Shahid Kapoor	52	20	M	4021469	Agra

## 19. MySQL DELETE



## How to delete data from Tables with SQL ?

Employee Table

Name	Age	Gender	Salary
Ram Kumar	19	Male	4500
Salman Khan	18	Male	5200
Meera Khan	20	Female	6000
Sarita Kumari	21	Female	8500
Anil Kapoor	20	Male	6300



## DELETE Command



## DELETE Syntax

`DELETE FROM table_name`

`WHERE condition;`

if we use delete command without where then all data will be delete so need to take care

`DELETE FROM table_name;`

This screenshot shows a MySQL Workbench interface with three tabs labeled 'personal' at the top. The central pane displays a sequence of SQL statements:

```
1 • COMMIT;  
2  
3 • DELETE FROM personal;  
4  
5 • ROLLBACK;
```

This screenshot shows a MySQL Workbench interface with three tabs labeled 'personal' at the top. The central pane displays a single SQL statement:

```
1 • SELECT * FROM student.personal;
```

Below the statement is a 'Result Grid' window. The grid has a header row with columns: id, name, percentage, age, gender, phone, city. There is one data row below the header, indicated by an asterisk (\*).

	id	name	percentage	age	gender	phone	city
*							

## Practical

The screenshot shows the MySQL Workbench interface. In the top panel, the SQL tab contains the query:

```
1 • SELECT * FROM student.personal;
```

In the bottom panel, the Results Grid displays the following data:

	id	name	percentage	age	gender	phone	city
1	1	Ram Kumar	45	19	M	4022144	Agra
2	2	Sarita Kumari	55	22	F	4027895	Delhi
3	3	Salman Khan	62	20	M	4022254	Agra
4	4	Juhi Chawla	47	20	F	4028524	Bhopal
5	5	Anil Kapoor	74	22	M	4027896	Agra
6	6	John Abraham	64	21	M	4022034	Delhi
7	7	Shahid Kapoor	52	20	M	4021469	Agra

The screenshot shows the MySQL Workbench interface. In the top panel, the SQL tab contains the query:

```
1  
2 • DELETE FROM personal  
3 WHERE id = 4;
```

In the bottom panel, the Results Grid displays the same data as before, but the row for Juhi Chawla (id 4) is highlighted with a red border.

	id	name	percentage	age	gender	phone	city
1	1	Ram Kumar	45	19	M	4022144	Agra
2	2	Sarita Kumari	55	22	F	4027895	Delhi
3	3	Salman Khan	62	20	M	4022254	Agra
4	4	Juhi Chawla	47	20	F	4028524	Bhopal
5	5	Anil Kapoor	74	22	M	4027896	Agra
6	6	John Abraham	64	21	M	4022034	Delhi
7	7	Shahid Kapoor	52	20	M	4021469	Agra

```
1 • COMMIT;  
2  
3 • DELETE FROM personal  
4 WHERE id = 4;  
5  
6 • ROLLBACK; I
```

<b>id</b>	<b>name</b>	<b>percentage</b>	<b>age</b>	<b>gender</b>	<b>phone</b>	<b>city</b>
1	Ram Kumar	45	19	M	4022144	Agra
2	Sarita Kumari	55	22	F	4027895	Delhi
3	Salman Khan	62	20	M	4022254	Agra
4	Juhi Chawla	47	20	F	4028524	Bhopal
5	Anil Kapoor	74	22	M	4027896	Agra
6	John Abraham	64	21	M	4022034	Delhi
7	Shahid Kapoor	52	20	M	4021469	Agra

```
1 • SELECT * FROM student.personal;
```

The screenshot shows a MySQL Workbench interface with two panes. The top pane displays a transaction script:

```
1 • COMMIT;  
2  
3 • DELETE FROM personal  
4 WHERE gender = 'F';  
5  
6 • ROLLBACK;
```

The bottom pane shows the result of the query:

```
1 • SELECT * FROM student.personal;
```

The result grid displays the following data:

	id	name	percentage	age	gender	phone	city
▶	1	Ram Kumar	45	19	M	4022144	Agra
	3	Salman Khan	62	20	M	4022254	Agra
	5	Anil Kapoor	74	22	M	4027896	Agra
	6	John Abraham	64	21	M	4022034	Delhi
	7	Shahid Kapoor	52	20	M	4021469	Agra

## 20. MySQL PRIMARY KEY & FOREIGN KEY



## List of Constraints in MySQL

- NOT NULL
- UNIQUE
- DEFAULT
- CHECK
- PRIMARY KEY
- FOREIGN KEY



## What is PRIMARY KEY Constraint ?

- Primary key always has unique data.
- A primary key cannot have null value.
- A table can contain only one primary key constraint.

Student Table				
PRIMARY KEY	Id	Name	Age	City
• Unique Data	1	Ram Kumar	19	Agra
• No Null Value	2	Salman Khan	18	Bhopal
	3	Meera Khan	19	Agra
	4	Sarita Kumari	21	Delhi

option-1 for primary key



## Create Table with PRIMARY KEY Syntax

```
CREATE TABLE table_name (
    id INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    age INT NOT NULL,
    city VARCHAR(10) NOT NULL ,
    PRIMARY KEY (id)
);
```

option-2 for primary key (already exist table)



## Alter Table with PRIMARY KEY Syntax

ALTER TABLE `table_name`

ADD PRIMARY KEY (`id`);

Foreign Key



## What is FOREIGN KEY Constraint ?

- A FOREIGN KEY is a key used to link two tables together.
- A FOREIGN key in one table used to point PRIMARY key in another table.



## FOREIGN KEY in Table

Student Table

<b>Id</b>	<b>Name</b>	<b>Age</b>	<b>City</b>
1	Ram Kumar	19	1
2	Salman Khan	18	2
3	Meera Khan	19	1
4	Sarita Kumari	21	3

PRIMARY KEY

FOREIGN KEY

City Table

<b>Cid</b>	<b>City</b>
1	Agra
2	Bhopal
3	Delhi

PRIMARY KEY

Link

option-1



## Create Table with FOREIGN KEY Syntax

```
CREATE TABLE student(
    id INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    age INT NOT NULL,
    city VARCHAR(10) NOT NULL ,
    PRIMARY KEY (id),
    FOREIGN KEY (city) REFERENCES City (cid)
);
```

option-2



## Alter Table with FOREIGN KEY Syntax

```
ALTER TABLE table_name
ADD FOREIGN KEY (city) REFERENCES City (cid);
```

Practical

The screenshot shows two windows of MySQL Workbench. The top window displays the SQL code for creating a 'city' table:

```
CREATE TABLE city(
    cid INT NOT NULL AUTO_INCREMENT,
    cityname VARCHAR(50) NOT NULL,
    PRIMARY KEY (cid)
);
```

The bottom window shows the results of a query:

```
1. SELECT * FROM student.city;
```

The results grid shows the following data:

cid	cityname
*	

The screenshot shows the phpMyAdmin interface with the 'city' table selected in the 'personal' database. The left sidebar shows the 'Schemas' tree with 'student' expanded, revealing 'Tables', 'Views', 'Stored Procedures', and 'Functions'. The main area contains the following SQL code:

```
1  INSERT INTO city(cityname)
2  VALUES('Agra'),
3  ('Delhi'),
4  ('Bhopal'),
5  ('Jaipur'),
6  ('Noida');
```

The screenshot shows the phpMyAdmin interface with the 'city' table selected in the 'personal' database. The left sidebar shows the 'Schemas' tree with 'student' expanded. The main area contains the following SQL code:

```
1 • SELECT * FROM student.city;
```

Below the code, a result grid displays the data from the 'city' table:

cid	cityname
1	Agra
2	Delhi
3	Bhopal
4	Jaipur
5	Noida

The screenshot shows the phpMyAdmin interface with the 'personal' database selected. The left sidebar shows the 'Schemas' tree with 'student' expanded. The main area contains the following SQL code:

```
1  CREATE TABLE personal (
2      id INT NOT NULL,
3      name VARCHAR(50) NOT NULL,
4      percentage INT NOT NULL,
5      age INT NOT NULL,
6      gender VARCHAR(1) NOT NULL,
7      city INT NOT NULL,
8      PRIMARY KEY (id),
9      FOREIGN KEY (city) REFERENCES City (cid)
10 );
```

```
11  
12 • INSERT INTO personal (id, name, percentage, age, gender, city)  
13     VALUES (1, "Ram Kumar",45, 19, "M" , 1),  
14     (2, "Sarita Kumari",55, 22, "F" , 2),  
15     (3, "Salman Khan",62, 20, "M" , 1),  
16     (4, "Juhi Chawla",47, 18, "F" , 3),  
17     (5, "Anil Kapoor",74, 22, "M" , 1),  
18     (6, "John Abraham",64, 21, "M" , 2),  
19     (7, "Shahid Kapoor",52, 20, "M" , 1);  
20
```

The screenshot shows the MySQL Workbench interface. The top window displays the SQL query:

```
1 • SELECT * FROM student.personal;
```

The bottom window shows the resulting grid of data:

	id	name	percentage	age	gender	city
1	1	Ram Kumar	45	19	M	1
2	2	Sarita Kumari	55	22	F	2
3	3	Salman Khan	62	20	M	1
4	4	Juhi Chawla	47	18	F	3
5	5	Anil Kapoor	74	22	M	1
6	6	John Abraham	64	21	M	2
7	7	Shahid Kapoor	52	20	M	1

## 21. MySQL INNER JOIN

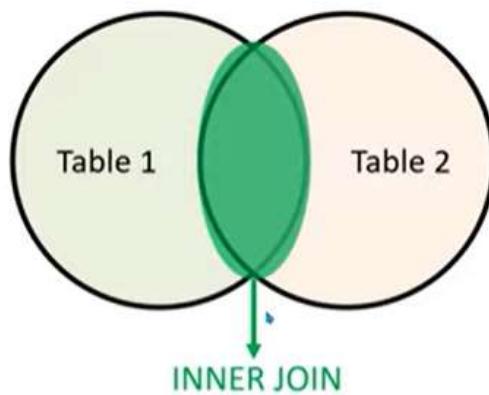


## Types of JOINS in MySQL

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- CROSS JOIN



## What is INNER JOIN ?



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



## JOIN Two Tables

### INNER JOIN



## INNER JOIN Syntax

`SELECT columns`

`FROM table1`

`INNER JOIN table2`

`ON table1.column_name = table2.column_name;`

↓  
`FOREIGN KEY`

↓  
`PRIMARY KEY`

Practical

Navigator personal personal city

SCHEMAS Filter objects

- phpmyadmin
- student
  - Tables
    - city
    - personal
  - Views
  - Stored Procedures
  - Functions
- test

Administration Schemas Information No object selected

1 • SELECT \* FROM student.personal;

	id	name	percentage	age	gender	city
▶	1	Ram Kumar	45	19	M	1
	2	Sarita Kumari	55	22	F	2
	3	Salman Khan	62	20	M	1
	4	Juhi Chawla	47	18	F	3
	5	Anil Kapoor	74	22	M	1
	6	John Abraham	64	21	M	2
	7	Shahid Kapoor	52	20	M	1

Navigator personal personal city

SCHEMAS Filter objects

- phpmyadmin
- student
  - Tables
    - city
    - personal
  - Views
  - Stored Procedures
  - Functions
- test

Administration Schemas Information No object selected

1 • SELECT \* FROM student.personal;

	id	name	percentage	age	gender	city
▶	1	Ram Kumar	45	19	M	1
	2	Sarita Kumari	55	22	F	2
	3	Salman Khan	62	20	M	1
	4	Juhi Chawla	47	18	F	3
	5	Anil Kapoor	74	22	M	1
	6	John Abraham	64	21	M	2
	7	Shahid Kapoor	52	20	M	1

Navigator personal personal city

SCHEMAS Filter objects

- phpmyadmin
- student**
  - Tables
    - city
    - personal
  - Views
  - Stored Procedures
  - Functions
- test

```

1   SELECT * FROM personal INNER JOIN city
2   ON personal.city = city.cid;
  
```

	id	name	percentage	age	gender	city	cid	cityname
>	1	Ram Kumar	45	19	M	1	1	Agra
	2	Sarita Kumari	55	22	F	2	2	Delhi
	3	Salman Khan	62	20	M	1	1	Agra
	4	Juhi Chawla	47	18	F	3	3	Bhopal
	5	Anil Kapoor	74	22	M	1	1	Agra
	6	John Abraham	64	21	M	2	2	Delhi
	7	Shahid Kapoor	52	20	M	1	1	Agra

Used alis for table name

Navigator personal personal city

SCHEMAS Filter objects

- phpmyadmin
- student**
  - Tables
    - city
    - personal
  - Views
  - Stored Procedures
  - Functions
- test

```

1   SELECT * FROM personal p INNER JOIN city c
2   ON p.city = c.cid;
  
```

	id	name	percentage	age	gender	city	cid	cityname
>	1	Ram Kumar	45	19	M	1	1	Agra
	2	Sarita Kumari	55	22	F	2	2	Delhi
	3	Salman Khan	62	20	M	1	1	Agra
	4	Juhi Chawla	47	18	F	3	3	Bhopal
	5	Anil Kapoor	74	22	M	1	1	Agra
	6	John Abraham	64	21	M	2	2	Delhi
	7	Shahid Kapoor	52	20	M	1	1	Agra

The screenshot shows the phpMyAdmin interface with the following details:

- Schemas:** personal, city
- Query:**

```

1  SELECT * FROM personal p INNER JOIN city c
2  ON p.city = c.cid;
    
```
- Result Grid:**

	id	name	percentage	age	gender	city	cid	cityname
1	1	Ram Kumar	45	19	M	1	1	Agra
2	2	Sarita Kumari	55	22	F	2	2	Delhi
3	3	Salman Khan	62	20	M	1	1	Agra
4	4	Juhি Chawla	47	18	F	3	3	Bhopal
5	5	Anil Kapoor	74	22	M	1	1	Agra
6	6	John Abraham	64	21	M	2	2	Delhi
7	7	Shahid Kapoor	52	20	M	1	1	Agra

we dont want whole table , just want required columns

The screenshot shows the phpMyAdmin interface with the following details:

- Schemas:** personal, city
- Query:**

```

1  SELECT p.id,p.name,p.percentage,p.age,p.gender,c.cityname
2  FROM personal p INNER JOIN city c
3  ON p.city = c.cid;
    
```
- Result Grid:**

	id	name	percentage	age	gender	cityname
1	1	Ram Kumar	45	19	M	Agra
2	2	Sarita Kumari	55	22	F	Delhi
3	3	Salman Khan	62	20	M	Agra
4	4	Juhি Chawla	47	18	F	Bhopal
5	5	Anil Kapoor	74	22	M	Agra
6	6	John Abraham	64	21	M	Delhi
7	7	Shahid Kapoor	52	20	M	Agra

using where clause with condition

The screenshot shows the phpMyAdmin interface with the following details:

- Schemas:** personal, city
- Query:**

```

1  SELECT p.id,p.name,p.percentage,p.age,p.gender,c.cityname
2  FROM personal p INNER JOIN city c
3  ON p.city = c.cid
4  WHERE c.cityname = "Agra";
    
```
- Result Grid:**

	id	name	percentage	age	gender	cityname
1	1	Ram Kumar	45	19	M	Agra
3	3	Salman Khan	62	20	M	Agra
5	5	Anil Kapoor	74	22	M	Agra
7	7	Shahid Kapoor	52	20	M	Agra

using order by clause

The screenshot shows the phpMyAdmin interface. The left sidebar displays the 'personal' schema with tables 'city' and 'personal'. The right pane shows a SQL query window with the following code:

```
1 SELECT p.id,p.name,p.percentage,p.age,p.gender,c.cityname
2 FROM personal p INNER JOIN city c
3 ON p.city = c.cid
4 WHERE c.cityname = "Agra"
5 ORDER BY p.name;
```

Below the query, a results grid displays the following data:

	id	name	percentage	age	gender	cityname
5	Anil Kapoor	74	22	M	Agra	
1	Ram Kumar	45	19	M	Agra	
3	Salman Khan	62	20	M	Agra	
7	Shahid Kapoor	52	20	M	Agra	

inner join and join both works same

The screenshot shows the phpMyAdmin interface. The left sidebar displays the 'personal' schema with tables 'city' and 'personal'. The right pane shows a SQL query window with the following code:

```
1 SELECT p.id,p.name,p.percentage,p.age,p.gender,c.cityname
2 FROM personal p JOIN city c
3 ON p.city = c.cid
4 WHERE c.cityname = "Agra"
5 ORDER BY p.name;
```

Below the query, a results grid displays the following data:

	id	name	percentage	age	gender	cityname
5	Anil Kapoor	74	22	M	Agra	
1	Ram Kumar	45	19	M	Agra	
3	Salman Khan	62	20	M	Agra	
7	Shahid Kapoor	52	20	M	Agra	

## 22. MySQL LEFT JOIN & RIGHT JOIN

### Left Join

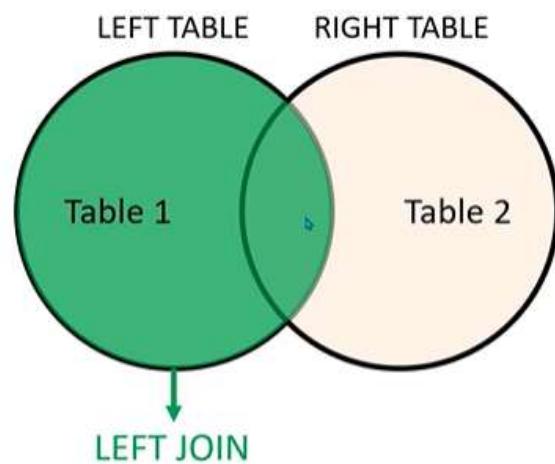


## Types of JOINS in MySQL

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- CROSS JOIN



### What is LEFT JOIN ?



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



## JOIN Two Tables

### LEFT JOIN

Student Table				City Table	
Id	Name	Age	City	Cid	City
1	Ram Kumar	19	1	1	Agra
2	Salman Khan	18	2	2	Bhopal
3	Meera Khan	19		3	Delhi
4	Sarita Kumari	21	3	4	Noida

Diagram illustrating a LEFT JOIN between Student Table and City Table:

- Match:** A green double-headed arrow labeled "Match" connects the "City" column in the Student Table to the "City" column in the City Table.
- FOREIGN KEY:** A red arrow labeled "FOREIGN KEY" points from the "City" column in the Student Table to the "City" column in the City Table.
- PRIMARY KEY:** A blue arrow labeled "PRIMARY KEY" points from the "City" column in the City Table to the "City" column in the Student Table.



## LEFT JOIN Syntax

SELECT columns

FROM table1

LEFT JOIN table2

ON table1.column\_name = table2.column\_name;

FOREIGN KEY

PRIMARY KEY

Practical

Table-1 (Personal)

The screenshot shows the phpMyAdmin interface with the 'personal' schema selected. A query window displays the result of the SQL command: `1 • SELECT * FROM student.personal;`. The result grid shows the following data:

	id	name	percentage	age	gender	city
1	1	Ram Kumar	45	19	M	1
2	2	Sarita Kumari	55	22	F	2
3	3	Salman Khan	62	20	M	1
4	4	Juhi Chawla	47	18	F	3
5	5	Anil Kapoor	74	22	M	1
6	6	John Abraham	64	21	M	2
7	7	Shahid Kapoor	52	20	M	1
*	HULL	HULL	HULL	HULL	HULL	HULL

Table-2 (City)

The screenshot shows the phpMyAdmin interface with the 'city' schema selected. A query window displays the result of the SQL command: `1 • SELECT * FROM student.city;`. The result grid shows the following data:

	cid	cityname
1	1	Agra
2	2	Delhi
3	3	Bhopal
4	4	Jaipur
5	5	Noida
*	HULL	HULL

The screenshot shows the phpMyAdmin interface with the 'personal' schema selected. In the SQL tab, the following query is run:

```
1 SELECT * FROM personal LEFT JOIN city
2 ON personal.city = city.cid;
```

The results are displayed in a grid:

	id	name	percentage	age	gender	city	cid	cityname
1	1	Ram Kumar	45	19	M	1	1	Agra
2	2	Sarita Kumari	55	22	F	2	2	Delhi
3	3	Salman Khan	62	20	M	1	1	Agra
4	4	Juhi Chawla	47	18	F	3	3	Bhopal
5	5	Anil Kapoor	74	22	M	1	1	Agra
6	6	John Abraham	64	21	M	2	2	Delhi
7	7	Shahid Kapoor	52	20	M	1	1	Agra

The screenshot shows the phpMyAdmin interface with the 'personal' schema selected. In the SQL tab, the following query is run:

```
1 • SELECT * FROM student.personal;
```

The results are displayed in a grid:

	id	name	percentage	age	gender	city
1	1	Ram Kumar	45	19	M	1
2	2	Sarita Kumari	55	22	F	2
3	3	Salman Khan	62	20	M	1
4	4	Juhi Chawla	47	18	F	3
5	5	Anil Kapoor	74	22	M	1
6	6	John Abraham	64	21	M	2
7	7	Shahid Kapoor	52	20	M	1

changing the inbuilt setting for not null constraint without command, because we want to apply left join command as one null value in personal table

The screenshot shows the phpMyAdmin interface. In the top navigation bar, the database is set to 'personal' and the table is 'personal'. A query window displays the result of the SQL command: `1 • SELECT * FROM student.personal;`. The result grid shows the following data:

	id	name	percentage	age	gender	city
1	Ram Kumar	45	19	M	1	
2	Sarita Kumari	55	22	F	2	
3	Salman Khan	62	20	M	1	
4	Juhi Chawla	47	18	F	3	
5	Anil Kapoor	74	22	M	1	
6	John Abraham	64	21	M	2	
7	Shahid Kapoor	52	20	M	1	

The screenshot shows the table structure for 'personal'. The table has the following columns:

- Column Name: id, Datatype: INT(11), PK: checked, NN: checked, UQ: checked, B: checked, UN: checked, ZF: checked, AI: checked, G: checked
- Column Name: name, Datatype: VARCHAR(50)
- Column Name: percentage, Datatype: INT(11)
- Column Name: age, Datatype: INT(11)
- Column Name: gender, Datatype: VARCHAR(1)
- Column Name: city, Datatype: INT(11)

The screenshot shows the 'Review SQL Script' dialog for the 'personal' table. The table name is 'personal'. The SQL script to be applied is:

```

1 ALTER TABLE `student`.`personal`
2 DROP FOREIGN KEY `personal_ibfk_1`;
3 ALTER TABLE `student`.`personal`
4 CHANGE COLUMN `city` `city` INT(11) NULL ;
5 ALTER TABLE `student`.`personal`
6 ADD CONSTRAINT `personal_ibfk_1`
7 FOREIGN KEY (`city`)
8 REFERENCES `student`.`city` (`cid`);
9

```

actually we want to change in city column for null value so for that through mysql tool we are deleting that value

Navigator personal personal city

SCHEMAS Filter objects

- phpmyadmin
- student
  - Tables
    - city
    - personal
  - Views
  - Stored Procedures
  - Functions
- test

Administration Schemas Information

Schema: student

1 • SELECT \* FROM student.personal;

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content: 15

id	name	percentage	age	gender	city
1	Ram Kumar	45	1		1
2	Sarita Kumari	55	22	F	2
3	Salman Khan	62	20	M	1
4	Juhi Chawla	47	18	F	3
5	Anil Kapoor	74	22	M	1
6	John Abraham	64	21	M	2
7	Shahid Kapoor	52	20	M	1
HULL	HULL	HULL	HULL	HULL	HULL

Navigator personal personal city

SCHEMAS Filter objects

- phpmyadmin
- student
  - Tables
    - city
    - personal
  - Views
  - Stored Procedures
  - Functions
- test

Administration Schemas Information

Schema: student

1 • SELECT \* FROM student.personal;

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content: 15

id	name	percentage
1	Ram Kumar	45
2	Sarita Kumari	55
3	Salman Khan	62
4	Juhi Chawla	47
5	Anil Kapoor	74
6	John Abraham	64
7	Shahid Kapoor	52
HULL	HULL	HULL

MySQL Workbench

! Apply Changes to Recordset

Autocommit is currently disabled and a transaction might be open.  
Recordset changes will be applied within that transaction and will be left uncommitted until you explicitly commit it manually.  
If you want it to be executed separately, click Cancel and commit the transaction first.

Apply Cancel

Navigator personal personal city personal - Table

**SCHEMAS**

- phpmyadmin
- student**
  - Tables
    - city
    - personal
  - Views
  - Stored Procedures
  - Functions
- test

1 **SELECT \* FROM personal LEFT JOIN city ON personal.city = city.cid;**

	id	name	percentage	age	gender	city	cid	cityname
>	1	Ram Kumar	45	19	M	1	1	Agra
	2	Sarita Kumari	55	22	F	2	2	Delhi
	3	Salman Khan	62	20	M	0		
	4	Juhি Chawla	47	18	F	3	3	Bhopal
	5	Anil Kapoor	74	22	M	1	1	Agra
	6	John Abraham	64	21	M	0		
	7	Shahid Kapoor	52	20	M	1	1	Agra

left join for specific columns

Navigator personal personal city personal - Table

**SCHEMAS**

- phpmyadmin
- student**
  - Tables
    - city
    - personal
  - Views
  - Stored Procedures
  - Functions
- test

1 **SELECT p.id, p.name,p.percentage,p.age,p.gender,c.cityname**  
 2 **FROM personal p LEFT JOIN city c**  
 3 **ON p.city = c.cid;**

	id	name	percentage	age	gender	cityname
>	1	Ram Kumar	45	19	M	Agra
	2	Sarita Kumari	55	22	F	Delhi
	3	Salman Khan	62	20	M	
	4	Juhি Chawla	47	18	F	Bhopal
	5	Anil Kapoor	74	22	M	Agra
	6	John Abraham	64	21	M	
	7	Shahid Kapoor	52	20	M	Agra

where condition for left join

Navigator personal personal city personal - Table

**SCHEMAS**

- phpmyadmin
- student**
  - Tables
    - city
    - personal
  - Views
  - Stored Procedures
  - Functions
- test

1 **SELECT p.id, p.name,p.percentage,p.age,p.gender,c.cityname**  
 2 **FROM personal p LEFT JOIN city c**  
 3 **ON p.city = c.cid**  
 4 **WHERE gender = "M";**

	id	name	percentage	age	gender	cityname
>	1	Ram Kumar	45	19	M	Agra
	3	Salman Khan	62	20	M	
	5	Anil Kapoor	74	22	M	Agra
	6	John Abraham	64	21	M	
	7	Shahid Kapoor	52	20	M	Agra

order by clause for left join

The screenshot shows the phpMyAdmin interface. The left sidebar displays the 'personal' schema with tables 'personal' and 'city'. The main area contains a SQL query:

```
1  SELECT p.id, p.name,p.percentage,p.age,p.gender,c.cityname
2  FROM personal p LEFT JOIN city c
3  ON p.city = c.cid
4  WHERE gender = "M"
5  ORDER BY name;
```

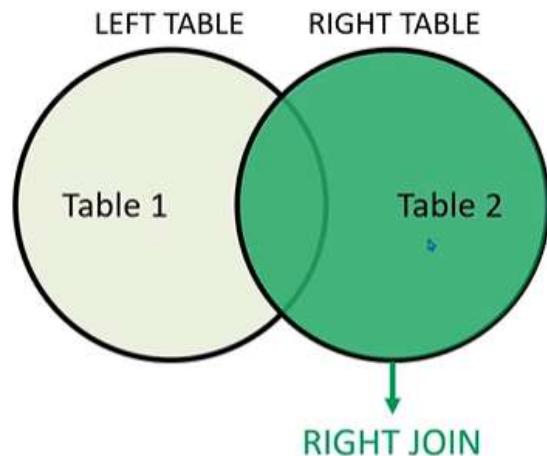
The results grid shows the following data:

	id	name	percentage	age	gender	cityname
5	Anil Kapoor	74	22	M	Agra	
6	John Abraham	64	21	M	Agra	
1	Ram Kumar	45	19	M	Agra	
3	Salman Khan	62	20	M	Agra	
7	Shahid Kapoor	52	20	M	Agra	

## Right Join



## What is RIGHT JOIN ?



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



## JOIN Two Tables

### RIGHT JOIN

Student Table				City Table	
Id	Name	Age	City	Cid	City
1	Ram Kumar	19	1	1	Agra
2	Salman Khan	18	2	2	Bhopal
3	Meera Khan	19		3	Delhi
4	Sarita Kumari	21	3	4	Noida

Match

FOREIGN KEY      PRIMARY KEY



## RIGHT JOIN Syntax

SELECT columns

FROM table1

RIGHT JOIN table2

ON table1.column\_name = table2.column\_name;

↓  
FOREIGN KEY

↓  
PRIMARY KEY

### Practical

The screenshot shows the phpMyAdmin interface. The left sidebar displays the 'student' schema with tables 'city' and 'personal'. The main area shows a SQL query:

```
1 SELECT p.id, p.name, p.percentage, p.age, p.gender, c.cityname
2 FROM personal p RIGHT JOIN city c
3 ON p.city = c.cid;
```

Below the query, the results are displayed in a grid:

	id	name	percentage	age	gender	cityname
1	Ram Kumar	45	19	M	Agra	
2	Sarita Kumari	55	22	F	Delhi	
4	Juhি Chawla	47	18	F	Bhopal	
5	Anil Kapoor	74	22	M	Agra	
7	Shahid Kapoor	52	20	M	Agra	
					Jaipur	
					Noida	

## 23. MySQL CROSS JOIN

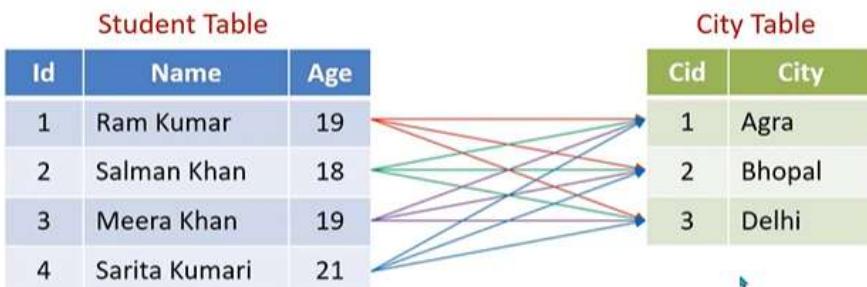


## Types of JOINS in MySQL

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- CROSS JOIN



## CROSS JOIN Two Tables





## CROSS JOIN Result

<b>Id</b>	<b>Name</b>	<b>Age</b>	<b>Cid</b>	<b>City</b>
1	Ram Kumar	19	1	Agra
2	Ram Kumar	19	2	Bhopal
3	Ram Kumar	19	3	Delhi
4	Salman Khan	18	1	Agra
5	Salman Khan	18	2	Bhopal
6	Salman Khan	18	3	Delhi
7	Meera Khan	19	1	Agra
8	Meera Khan	19	2	Bhopal
9	Meera Khan	19	3	Delhi
10	Sarita Kumari	21	1	Agra
11	Sarita Kumari	21	2	Bhopal
12	Sarita Kumari	21	3	Delhi



## CROSS JOIN Syntax

`SELECT columns`

`FROM table1`

`CROSS JOIN table2;`

Practical

Table 1

The screenshot shows the phpMyAdmin interface. The left sidebar displays the database structure under the 'personal' schema, including tables like 'city' and 'personal'. The main area shows the results of the query:

```
1 • SELECT * FROM student.personal;
```

	id	name	percentage	age	gender	city
1	1	Ram Kumar	45	19	M	1
2	2	Sarita Kumari	55	22	F	2
3	3	Salman Khan	62	20	M	1
4	4	Juhi Chawla	47	18	F	3
5	5	Anil Kapoor	74	22	M	1
6	6	John Abraham	64	21	M	2
7	7	Shahid Kapoor	52	20	M	1

Table 2

The screenshot shows the phpMyAdmin interface. The left sidebar displays the database structure under the 'personal' schema, including tables like 'city' and 'personal'. The main area shows the results of the query:

```
1 • SELECT * FROM student.city;
```

	cid	cityname
1	1	Agra
2	2	Delhi
3	3	Bhopal
4	4	Jaipur
5	5	Noida

Total 35 combinations

The screenshot shows the phpMyAdmin interface with the following details:

- Schemas:** personal, personal, city
- Query:**

```

1 • SELECT *
2 FROM personal CROSS JOIN city;
    
```
- Result Grid:**

	id	name	percentage	age	gender	city	cid	cityname
3	Salman Khan	62		20	F	3	5	Noida
5	Anil Kapoor	74		22	M	1	1	Agra
5	Anil Kapoor	74		22	M	1	2	Delhi
5	Anil Kapoor	74		22	M	1	3	Bhopal
5	Anil Kapoor	74		22	M	1	4	Jaipur
4	Juhি Chawla	47		18	F	3	5	Noida
4	Juhি Chawla	47		18	F	3	5	Noida
4	Juhি Chawla	47		18	F	3	5	Noida
5	Anil Kapoor	74		22	M	1	1	Agra
5	Anil Kapoor	74		22	M	1	2	Delhi
5	Anil Kapoor	74		22	M	1	3	Bhopal
5	Anil Kapoor	74		22	M	1	4	Jaipur
5	Anil Kapoor	74		22	M	1	5	Noida

Using specific columns

The screenshot shows the phpMyAdmin interface with the following details:

- Schemas:** personal, personal, city
- Query:**

```

1 • SELECT p.id,p.name,c.cityname
2 FROM personal p CROSS JOIN city c;
    
```
- Result Grid:**

	id	name	cityname
1	Ram Kumar	Agra	
1	Ram Kumar	Delhi	
1	Ram Kumar	Bhopal	
1	Ram Kumar	Jaipur	
1	Ram Kumar	Noida	
2	Sarita Kumari	Agra	
2	Sarita Kumari	Delhi	
2	Sarita Kumari	Bhopal	
2	Sarita Kumari	Jaipur	
2	Sarita Kumari	Noida	
3	Salman Khan	Agra	
3	Salman Khan	Delhi	
3	Salman Khan	Bhopal	

changing column names

The screenshot shows the phpMyAdmin interface. In the left sidebar, under the 'Schemas' section, the 'student' schema is selected, showing tables like 'city' and 'personal'. The main area displays a query results grid for the following SQL statement:

```
1 • SELECT p.id,p.name AS Name,c.cityname AS City
2   FROM personal p CROSS JOIN city c;
```

The result grid contains the following data:

	id	Name	City
1	1	Ram Kumar	Agra
1	1	Ram Kumar	Delhi
1	1	Ram Kumar	Bhopal
1	1	Ram Kumar	Jaipur
1	1	Ram Kumar	Noida
2	2	Sarita Kumari	Agra
2	2	Sarita Kumari	Delhi
2	2	Sarita Kumari	Bhopal

same result with another method

The screenshot shows the phpMyAdmin interface. In the left sidebar, under the 'Schemas' section, the 'student' schema is selected, showing tables like 'city' and 'personal'. The main area displays two separate SQL statements:

```
1 • SELECT p.id,p.name AS Name,c.cityname AS City
2   FROM personal p CROSS JOIN city c;
3
4 • SELECT p.id,p.name AS Name,c.cityname AS City
5   FROM personal p ,city c;
```

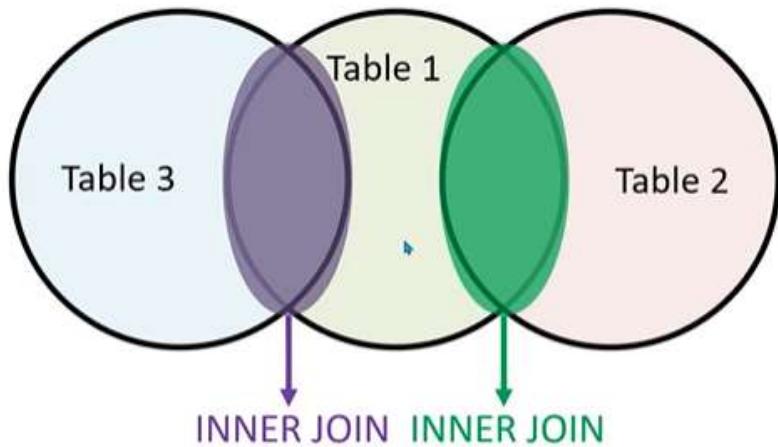
The result grid contains the same data as the previous screenshot:

	id	Name	City
1	1	Ram Kumar	Agra
1	1	Ram Kumar	Delhi
1	1	Ram Kumar	Bhopal
1	1	Ram Kumar	Jaipur
1	1	Ram Kumar	Noida
2	2	Sarita Kumari	Agra
2	2	Sarita Kumari	Delhi
2	2	Sarita Kumari	Bhopal

## 24. MySQL JOIN Multiple Tables



## How to JOIN Multiple Tables ?



## JOIN Three Tables

Student Table					City Table		Courses Table	
<b>Id</b>	<b>Name</b>	<b>Age</b>	<b>Courses</b>	<b>City</b>	<b>Cid</b>	<b>City</b>	<b>Crid</b>	<b>Course</b>
1	Ram Kumar	19	1	1	1	Agra	1	Btech
2	Salman Khan	18	3	2	2	Bhopal	2	BCA
3	Meera Khan	19	1	1	3	Delhi	3	BBA
4	Sarita Kumari	21	2	3	4	Noida		

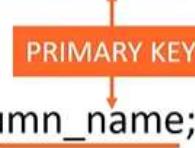
Diagram illustrating the JOIN operations:

- A red double-headed arrow labeled "JOIN" connects the Student Table and the City Table.
- A green double-headed arrow labeled "JOIN" connects the City Table and the Courses Table.



## INNER JOIN Syntax for Multiple Tables

```
SELECT columns  
FROM table1  
INNER JOIN table2  
ON table1.column_name = table2.column_name  
INNER JOIN table3  
ON table1.column_name = table3.column_name;
```



Practical

Navigator personal personal city courses

SCHEMAS phpmyadmin student Tables city courses personal Views Stored Procedures Functions test

1 • **SELECT \* FROM student.personal;**

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content:

	id	name	percentage	age	gender	city	courses
1	1	Ram Kumar	45	19	M	1	1
2	2	Sarita Kumari	55	22	F	2	2
3	3	Salman Khan	62	20	M	1	1
4	4	Juhি Chawla	47	18	F	3	1
5	5	Anil Kapoor	74	22	M	1	3
6	6	John Abraham	64	21	M	2	2
7	7	Shahid Kapoor	52	20	M	1	3
	HULL	HULL	HULL	HULL	HULL	HULL	HULL

Navigator personal personal city courses

SCHEMAS phpmyadmin student Tables city courses personal Views Stored Procedures Functions test

1 • **SELECT \* FROM student.city;**

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content:

	cid	cityname
1	1	Agra
2	2	Delhi
3	3	Bhopal
4	4	Jaipur
5	5	Noida
	HULL	HULL

The screenshot shows the phpMyAdmin interface. In the top navigation bar, the 'courses' tab is selected. Below it, a query window displays the following SQL code:

```
1 • SELECT * FROM student.courses;
```

The results are shown in a table titled 'Result Grid' with columns 'course\_id' and 'course\_name'. The data is as follows:

	course_id	course_name
1	1	Btech
2	2	BCA
3	3	MBA
*	MALE	MALE

joining first two tables

The screenshot shows the phpMyAdmin interface. In the top navigation bar, the 'personal' tab is selected. Below it, a query window displays the following SQL code:

```
1 SELECT * FROM personal p INNER JOIN city c  
2 ON p.city = c.cid;
```

The results are shown in a table titled 'Result Grid' with columns 'id', 'name', 'percentage', 'age', 'gender', 'city', 'courses', 'cid', and 'cityname'. The data is as follows:

	id	name	percentage	age	gender	city	courses	cid	cityname
1	1	Ram Kumar	45	19	M	1	1	1	Agra
2	2	Sarita Kumari	55	22	F	2	2	2	Delhi
3	3	Salman Khan	62	20	M	1	1	1	Agra
4	4	Juhi Chawla	47	18	F	3	1	3	Bhopal
5	5	Anil Kapoor	74	22	M	1	3	1	Agra
6	6	John Abraham	64	21	M	2	2	2	Delhi
7	7	Shahid Kapoor	52	20	M	1	3	1	Agra

joining first and third table along with previous result

The screenshot shows the MySQL Workbench interface. In the top-left pane, the 'Schemas' tree is visible, with 'personal' selected. The main area displays a query editor with the following SQL code:

```

1  SELECT * FROM
2  personal p INNER JOIN city c
3  ON p.city = c.cid
4  INNER JOIN courses cr
5  ON p.courses = cr.course_id;

```

Below the query editor is a 'Result Grid' table with the following data:

	id	name	percentage	age	gender	city	courses	cid	cityname	course_id	course_name
1	1	Ram Kumar	45	19	M	1	1	1	Agra	1	Btech
2	2	Sarita Kumari	55	22	F	2	2	2	Delhi	2	BCA
3	3	Salman Khan	62	20	M	1	1	1	Agra	1	Btech
4	4	Juhি Chawla	47	18	F	3	1	3	Bhopal	1	Btech
5	5	Anil Kapoor	74	22	M	1	3	1	Agra	3	MBA
6	6	John Abraham	64	21	M	2	2	2	Delhi	2	BCA
7	7	Shahid Kapoor	52	20	M	1	3	1	Agra	3	MBA

want to see specific columns

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```

1  SELECT p.id,p.name,p.percentage,p.age,p.gender,c.cityname,cr.course_name
2  FROM
3  personal p INNER JOIN city c
4  ON p.city = c.cid
5  INNER JOIN courses cr
6  ON p.courses = cr.course_id;

```

The resulting 'Result Grid' table displays the following data:

	id	name	percentage	age	gender	cityname	course_name
1	1	Ram Kumar	45	19	M	Agra	Btech
2	2	Sarita Kumari	55	22	F	Delhi	BCA
3	3	Salman Khan	62	20	M	Agra	Btech
4	4	Juhি Chawla	47	18	F	Bhopal	Btech
5	5	Anil Kapoor	74	22	M	Agra	MBA
6	6	John Abraham	64	21	M	Delhi	BCA
7	7	Shahid Kapoor	52	20	M	Agra	MBA

by using where condition

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```

1  SELECT p.id,p.name,p.percentage,p.age,p.gender,c.cityname,cr.course_name
2  FROM
3  personal p INNER JOIN city c
4  ON p.city = c.cid
5  INNER JOIN courses cr
6  ON p.courses = cr.course_id
7  WHERE c.cityname = "Agra";

```

The resulting 'Result Grid' table displays the following data, with the 'cityname' column highlighted in red:

	id	name	percentage	age	gender	cityname	course_name
1	1	Ram Kumar	45	19	M	Agra	Btech
3	3	Salman Khan	62	20	M	Agra	Btech
5	5	Anil Kapoor	74	22	M	Agra	MBA
7	7	Shahid Kapoor	52	20	M	Agra	MBA

# 25. MySQL GROUP BY & HAVING Clause



## GROUP BY Clause

Student Table

ID	Name	Age	City
1	Ram Kumar	19	1
2	Salman Khan	18	2
3	Meera Khan	19	1
4	Sarita Kumari	21	3

City Table

Cid	City	City	Total Stud
1	Agra	Agra	2
2	Bhopal	Bhopal	1
3	Delhi	Delhi	1
4	Noida		

### GROUP BY

The GROUP BY clause is used in conjunction with the  
SELECT statement and Aggregate functions  
to group rows together by common column values.



## SELECT with GROUP BY Syntax

SELECT columns

FROM table\_name

WHERE condition

GROUP BY column\_name(s);



## SELECT with GROUP BY Syntax

```
SELECT columns  
FROM table1 INNER JOIN table2  
ON table1.column_name = table2.column_name  
WHERE condition  
GROUP BY column_name(s);
```

Practical

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

personal personal city courses

1 • `SELECT * FROM student.personal;`

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	id	name	percentage	age	gender	city	courses
1	1	Ram Kumar	45	19	M	1	1
2	2	Sarita Kumari	55	22	F	2	2
3	3	Salman Khan	62	20	M	1	1
4	4	Juhi Chawla	47	18	F	3	1
5	5	Anil Kapoor	74	22	M	1	3
6	6	John Abraham	64	21	M	2	2
7	7	Shahid Kapoor	52	20	M	1	3

personal personal city courses

1 • `SELECT * FROM student.city;`

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	cid	cityname
1	1	Agra
2	2	Delhi
3	3	Bhopal
4	4	Jaipur
5	5	Noida

personal personal city courses

1 • **SELECT city, COUNT(city)**  
2 **FROM personal**  
3 **GROUP BY city;**  
4

Result Grid | Filter Rows: Export: Wrap Cell Content:

city	COUNT(city)
1	4
2	2
3	1

personal personal city courses

1 • **SELECT c.cityname, COUNT(p.city)**  
2 **FROM personal p INNER JOIN city c**  
3 **ON p.city = c.cid**  
4 **GROUP BY city;**

Result Grid | Filter Rows: Export: Wrap Cell Content:

cityname	COUNT(p.city)
Agra	4
Delhi	2
Bhopal	1

personal personal city courses

1 • **SELECT c.cityname, COUNT(p.city) AS Total**  
2 **FROM personal p INNER JOIN city c**  
3 **ON p.city = c.cid**  
4 **GROUP BY city;**  
5

Result Grid | Filter Rows: Export: Wrap Cell Content:

cityname	Total
Agra	4
Delhi	2
Bhopal	1

Navigator personal city courses

**SCHEMAS**

- personal
- city
- courses

Limit to 1000 rr | Filter Rows: | Export: | Wrap Cell Content: |

```

1 • SELECT c.cityname, COUNT(p.city) AS Total
2   FROM personal p INNER JOIN city c
3   ON p.city = c.cid
4   WHERE p.age >= 20
5   GROUP BY city;
6

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

cityname	Total
Agra	3
Delhi	2

No object selected

Navigator personal city courses

**SCHEMAS**

- personal
- city
- courses

Limit to 1000 rr | Filter Rows: | Export: | Wrap Cell Content: |

```

1 • SELECT c.cityname, COUNT(p.city) AS Total
2   FROM personal p INNER JOIN city c
3   ON p.city = c.cid |
4   GROUP BY city
5   ORDER BY COUNT(p.city);
6

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

cityname	Total
Bhopal	1
Delhi	2
Agra	4

No object selected

Navigator personal city courses

**SCHEMAS**

- personal
- city
- courses

Limit to 1000 rr | Filter Rows: | Export: | Wrap Cell Content: |

```

1 • SELECT c.cityname, COUNT(p.city) AS Total
2   FROM personal p INNER JOIN city c
3   ON p.city = c.cid
4   GROUP BY city
5   ORDER BY COUNT(p.city) DESC;
6

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

cityname	Total
Agra	4
Delhi	2
Bhopal	1

No object selected

Group by with having clause



## GROUP BY with HAVING Clause

Student Table

ID	Name	Age	City
1	Ram Kumar	19	1
2	Salman Khan	18	2
3	Meera Khan	19	1
4	Sarita Kumari	21	3

City Table

Cid	City
1	Agra
2	Bhopal
3	Delhi
4	Noida

City | Total Students

Agra	2
Bhopal	1
Delhi	1

Condition

HAVING Total Students > 5



## SELECT with GROUP BY & HAVING Syntax

```

SELECT columns
FROM table_name
GROUP BY column_name(s)
HAVING condition;
  
```

Practical

The screenshot shows the MySQL Workbench interface. In the top-left, the Navigator pane displays the database schema with 'personal' as the current database, containing 'personal', 'city', and 'courses' tables. Below the schema is a search bar labeled 'Filter objects'. The main area is a query editor with the following SQL code:

```

1 • SELECT c.cityname, COUNT(p.city) AS Total
2   FROM personal p INNER JOIN city c
3     ON p.city = c.cid
4   GROUP BY city
5   HAVING COUNT(p.city) > 3
6   ORDER BY COUNT(p.city) DESC;
  
```

Below the query editor is a results grid titled 'Result Grid' with the following data:

cityname	Total
Agra	4

## 26. MySQL SubQuery/Nested Query with EXISTS & NOT EXISTS



## What is SubQuery or Nested Query ?

Student Table			
<b>Id</b>	<b>Name</b>	<b>Age</b>	<b>Courses</b>
1	Ram Kumar	19	1
2	Salman Khan	18	3
3	Meera Khan	19	1
4	Sarita Kumari	21	2

Courses Table	
<b>Cid</b>	<b>Course</b>
1	Btech
2	BCA
3	BBA
	...

Name
Ram Kumar
Meera Khan



## SELECT with SubQuery Syntax

**SELECT** columns

**FROM** table1

**WHERE**

column = (**SELECT** columns **FROM** table2 **WHERE** condition);

UPGRADE

INSERT

DELETE

SELECT

Practical

MySQL Workbench

YahooBaba - Warning

File Edit View Query Database Server Tools Scripting Help

Navigator personal personal courses

SCHEMAS Filter objects

student Tables city courses personal

Views Stored Procedures Functions

test

No object selected

1 • `SELECT * FROM student.personal;`

	id	name	percentage	age	gender	city	courses
1	Ram Kumar	45	19	M	1	1	
2	Sarita Kumari	55	22	F	2	2	
3	Salman Khan	62	20	M	1	1	
4	Juhi Chawla	47	18	F	3	3	
5	Anil Kapoor	74	22	M	1	3	
6	John Abraham	64	21	M	2	2	
7	Shahid Kapoor	52	20	M	1	1	

personal personal courses

SCHEMAS Filter objects

student Tables city courses personal

Views Stored Procedures Functions

test

1 • `SELECT * FROM student.courses;`

	course_id	course_name
1	Btech	
2	BCA	
3	MBA	

personal personal courses

1 `SELECT name FROM personal`

2 `WHERE courses = (SELECT course_id FROM courses WHERE course_name = "MBA");`

3

	name
1	Juhi Chawla
2	Anil Kapoor

A screenshot of MySQL Workbench showing a query results grid. The query is:

```
1 SELECT name FROM personal
2 WHERE courses IN (SELECT course_id FROM courses WHERE course_name IN ("MBA","Btech"));
3
```

The results grid shows a single column named 'name' with the following data:

name
Ram Kumar
Salman Khan
Juhil Chawla
Anil Kapoor
Shahid Kapoor

## MySQL SELECT with EXISTS Syntax

SELECT columns

FROM table1

WHERE

EXISTS (SELECT columns FROM table2 WHERE condition);

If any Single Record Exists  
Then  
Parent command show results

## MySQL SELECT with NOT EXISTS Syntax

SELECT columns

FROM table1

WHERE

NOT EXISTS (SELECT columns FROM table2 WHERE condition);

If not any Single Record Exists  
Then  
Parent command show results

personal personal courses

```
1  SELECT name FROM personal
2  WHERE EXISTS (SELECT course_id FROM courses WHERE course_name IN ("MBA"));
3  |
```

Result Grid | Filter Rows: [ ] | Export: [ ] | Wrap Cell Content: [ ]

name
Ram Kumar
Sarita Kumari
Salman Khan
Juhি Chawla
Anil Kapoor
John Abraham
Shahid Kapoor

personal personal courses

```
1 • SELECT * FROM student.courses;
```

Result Grid | Filter Rows: [ ] | Edit: [ ] | Export/Import: [ ] | Wrap Cell Content: [ ]

course_id	course_name
1	Btech
2	BCA
3	MBA
*	

The image shows two screenshots of MySQL Workbench. Both screenshots have a top bar with tabs for 'personal', 'personal', and 'courses'. The first screenshot displays a query that selects names from the 'personal' table where there exists a course named 'Mtech' in the 'courses' table. The result grid shows a single column 'name' with one row containing the value 'name'. The second screenshot displays a query that selects names from the 'personal' table where there does not exist a course named 'Mtech' in the 'courses' table. The result grid shows a column 'name' with eight rows: Ram Kumar, Sarita Kumari, Salman Khan, Juhil Chawla, Anil Kapoor, John Abraham, and Shahid Kapoor.

```
1 SELECT name FROM personal
2 WHERE EXISTS (SELECT course_id FROM courses WHERE course_name IN ("Mtech"));
3
```

```
1 SELECT name FROM personal
2 WHERE NOT EXISTS (SELECT course_id FROM courses WHERE course_name IN ("Mtech"));
3
```

name
Ram Kumar
Sarita Kumari
Salman Khan
Juhil Chawla
Anil Kapoor
John Abraham
Shahid Kapoor

## 27. MySQL UNION & UNION ALL



## What is UNION & UNION ALL?

Student Table

<b>Id</b>	<b>Name</b>	<b>City</b>
1	Ram Kumar	Agra
2	Salman Khan	Delhi
3	Anil Kapoor	Bhopal

UNION ALL

<b>Name</b>	<b>City</b>
Ram Kumar	Agra
Salman Khan	Delhi
Meera Khan	Bhopal

Lecturer Table

<b>Id</b>	<b>Name</b>	<b>City</b>
1	Salim Khan	Agra
2	Ram Kumar	Delhi
3	Sarita	Agra

<b>Name</b>	<b>City</b>
Salim Khan	Agra
Ram Kumar	Delhi
Sarita	Agra



## What is UNION & UNION ALL?

Student Table

<b>Id</b>	<b>Name</b>	<b>City</b>
1	Ram Kumar	Agra
2	Salman Khan	Delhi
3	Anil Kapoor	Bhopal

UNION ALL

<b>Name</b>	<b>City</b>
Ram Kumar	Agra
Salman Khan	Delhi
Meera Khan	Bhopal

Lecturer Table

<b>Id</b>	<b>Name</b>	<b>City</b>
1	Salim Khan	Agra
2	Ram Kumar	Delhi
3	Sarita	Agra

UNION



<b>Name</b>	<b>City</b>
Ram Kumar	Delhi
Sarita	Agra



## UNION & UNION ALL Syntax

`SELECT column1, column2 FROM table1`

`UNION / UNION ALL`

`SELECT column1, column2 FROM table2;`

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

**personal students lecturers city courses**

1 • `SELECT * FROM student.students;`

	<b>id</b>	<b>name</b>	<b>age</b>	<b>gender</b>	<b>city</b>	<b>courses</b>
1	1	Ram Kumar	19	M	1	1
2	2	Sarita Kumari	22	F	2	2
3	3	Salman Khan	20	M	1	1
4	4	Juhi Chawla	18	F	3	3
5	5	Anil Kapoor	22	M	1	3
6	6	John Abraham	21	M	2	2
7	7	Shahid Kapoor	20	M	1	1

**personal students lecturers city courses**

1 • `SELECT * FROM student.lecturers;`

	<b>id</b>	<b>name</b>	<b>age</b>	<b>gender</b>	<b>city</b>	<b>courses</b>
1	1	Raj Kapoor	37	M	1	2
2	2	Sadhna	39	F	4	3
3	3	Ram Kumar	38	M	2	1
4	4	Salim Khan	45	M	3	2
5	5	Nagma	42	F	2	1

**Navigator**

**SCHEMAS**

- ▶ phmyadmin
- ◀ **student**
  - Tables
    - ▶ city
    - ▶ courses
    - ▶ lecturers
    - ▶ students
  - Views
  - Stored Procedures
  - Functions
- ▶ test

Administration Schemas Information

No object selected

**personal x students lecturers city courses**

1 **SELECT \* FROM students**  
 2 **UNION**  
 3 **SELECT \* FROM lecturers**

**Result Grid** | Filter Rows: Export: Wrap Cell Contents:

	<b>id</b>	<b>name</b>	<b>age</b>	<b>gender</b>	<b>city</b>	<b>courses</b>
▶	1	Ram Kumar	19	M	1	1
	2	Sarita Kumari	22	F	2	2
	3	Salman Khan	20	M	1	1
	4	Juhi Chawla	18	F	3	3
	5	Anil Kapoor	22	M	1	3
	6	John Abraham	21	M	2	2
	7	Shahid Kapoor	20	M	1	1
▶	1	Raj Kapoor	37	M	1	2
	2	Sadhna	39	F	4	3
	3	Ram Kumar	38	M	2	1
	4	Salim Khan	45	M	3	2
	5	Nagma	42	F	2	1

**Navigator**

**SCHEMAS**

- ▶ phmyadmin
- ◀ **student**
  - Tables
    - ▶ city
    - ▶ courses
    - ▶ lecturers
    - ▶ students
  - Views
  - Stored Procedures
  - Functions
- ▶ test

Administration Schemas Information

No object selected

**personal x students lecturers city courses**

1 **SELECT name FROM students**  
 2 **UNION**  
 3 **SELECT name FROM lecturers**

**Result Grid** | Filter Rows: Export: Wrap Cell Contents:

	<b>name</b>
▶	Ram Kumar
	Sarita Kumari
	Salman Khan
	Juhi Chawla
	Anil Kapoor
	John Abraham
	Shahid Kapoor
	Raj Kapoor
▶	Sadhna
	Salim Khan
	Nagma

Navigator

**SCHEMAS**

Filter objects

- phpmyadmin
- student**
  - Tables
    - city
    - courses
    - lecturers
    - students
  - Views
  - Stored Procedures
  - Functions
- test

Administration Schemas Information

No object selected

personal students lecturers city courses

1 **SELECT name FROM students**  
 2 **UNION ALL**  
 3 **SELECT name FROM lecturers**

	name
Ram Kumar	
Sarita Kumari	
Salman Khan	
Juhi Chawla	
Anil Kapoor	
John Abraham	
Shahid Kapoor	
Raj Kapoor	
Sadhna	
Ram Kumar	
Salim Khan	
Nagma	

Navigator

**SCHEMAS**

Filter objects

- phpmyadmin
- student**
  - Tables
    - city
    - courses
    - lecturers
    - students
  - Views
  - Stored Procedures
  - Functions
- test

Administration Schemas Information

No object selected

personal students lecturers city courses

1 **SELECT name,age FROM students**  
 2 **UNION ALL**  
 3 **SELECT name,age FROM lecturers**

	name	age
Ram Kumar	19	
Sarita Kumari	22	
Salman Khan	20	
Juhi Chawla	18	
Anil Kapoor	22	
John Abraham	21	
Shahid Kapoor	20	
Raj Kapoor	37	
Sadhna	39	
Ram Kumar	38	
Salim Khan	45	
Nagma	42	

**personal > students lecturers city courses**

```
1  SELECT name,age FROM students WHERE gender = "M"
2  UNION ALL
3  SELECT name,age FROM lecturers WHERE gender = "M"
```

Result Grid | Filter Rows: [ ] Export: Wrap Cell Content:

name	age
Ram Kumar	19
Salman Khan	20
Anil Kapoor	22
John Abraham	21
Shahid Kapoor	20
Raj Kapoor	37
Ram Kumar	38
Salim Khan	45

**personal > students lecturers city courses**

```
1  SELECT name,age FROM students WHERE gender = "M"
2  UNION ALL
3  SELECT name,age FROM lecturers WHERE gender = "F"
```

Result Grid | Filter Rows: [ ] Export: Wrap Cell Content:

name	age
Ram Kumar	19
Salman Khan	20
Anil Kapoor	22
John Abraham	21
Shahid Kapoor	20
Sadhna	39
Nagma	42

**personal > students lecturers city courses**

```
1  SELECT name,age FROM students
2  WHERE city = (SELECT cid FROM city WHERE cityname = "Delhi")
3  UNION ALL
4  SELECT name,age FROM lecturers
5  WHERE city = (SELECT cid FROM city WHERE cityname = "Delhi")
```

Result Grid | Filter Rows: [ ] Export: Wrap Cell Content:

name	age
Sarita Kumari	22
John Abraham	21
Ram Kumar	38
Nagma	42

```
personal > students lecturers city courses
File Edit View Insert Query Editor Help
1 • SELECT s.name,s.age , c.cityname
2   FROM students s INNER JOIN city c
3     ON s.city = c.cid
4   WHERE c.cityname = "Delhi"
5
```

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

	name	age	cityname
>	Sarita Kumari	22	Delhi
	John Abraham	21	Delhi

```
personal > students lecturers city courses
File Edit View Insert Query Editor Help
1 • SELECT s.name,s.age , c.cityname
2   FROM students s INNER JOIN city c
3     ON s.city = c.cid
4   WHERE c.cityname = "Delhi"
5 UNION ALL
6 SELECT l.name,l.age , ci.cityname
7   FROM lecturers l INNER JOIN city ci
8     ON l.city = ci.cid
9   WHERE ci.cityname = "Delhi";
```

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

	name	age	cityname
>	Sarita Kumari	22	Delhi
	John Abraham	21	Delhi
	Ram Kumar	38	Delhi
	Nagma	42	Delhi

## 28. MySQL IF & CASE Statement



## What is IF Clause ?

Student Table

<b>Id</b>	<b>Name</b>	<b>Percentage</b>
1	Ram Kumar	57
2	Salman Khan	28
3	Meera Khan	81
4	Sarita Kumari	45

City Table

<b>Id</b>	<b>Name</b>	<b>Percentage</b>	<b>Result</b>
1	Ram Kumar	57	PASS
2	Salman Khan	28	FAIL
3	Meera Khan	81	PASS
4	Sarita Kumari	45	PASS

→ Percentage  $\geq 33\%$  → Pass  
Percentage  $< 33\%$  → Fail



## SELECT with IF Clause Syntax

```
SELECT column1, column2,
       IF (Condition, TRUE Result, FALSE Result ) AS alias_name
    FROM table_name;
```

The screenshot shows the MySQL Workbench interface. On the left, the Navigator pane displays the 'students' schema with tables like city, courses, and students. The main area shows a query editor with the following SQL statement:

```
1 • SELECT * FROM student.students;
```

Below the query, the results are displayed in a grid:

	<b>id</b>	<b>name</b>	<b>percentage</b>	<b>age</b>	<b>gender</b>	<b>city</b>	<b>courses</b>
1	Ram Kumar	45	19	M	1	1	
2	Sarita Kumari	85	22	F	2	2	
3	Salman Khan	29	20	M	1	1	
4	Juhi Chawla	47	18	F	3	1	
5	Anil Kapoor	74	22	M	1	3	
6	John Abraham	64	21	M	2	2	
7	Shahid Kapoor	120	20	M	1	3	

The screenshot shows the MySQL Workbench interface. In the top-left, the Navigator pane displays the schema structure. The 'Schemas' section lists 'student' as the selected schema, which contains tables like 'city', 'courses', and 'students'. Below the schema list is a 'yb' entry. The top-right pane shows a SQL query window with the following code:

```

1 • SELECT id, name, percentage,
2 IF(percentage >= 33,"Pass","Fail") AS Result
3 FROM students;

```

The bottom-right pane displays the results of the query in a 'Result Grid'. The grid has columns: id, name, percentage, and Result. The data is as follows:

	id	name	percentage	Result
1	1	Ram Kumar	45	Pass
2	2	Sarita Kumari	85	Pass
3	3	Salman Khan	29	Fail
4	4	Juhি Chawla	47	Pass
5	5	Anil Kapoor	74	Pass
6	6	John Abraham	64	Pass
7	7	Shahid Kapoor	120	Pass

## MySQL What is CASE Clause ?

The diagram illustrates the relationship between the Student Table and the City Table. An arrow points from the Student Table to the City Table.

**Student Table**

Id	Name	Percentage
1	Ram Kumar	57
2	Salman Khan	28
3	Meera Khan	81
4	Sarita Kumari	45

**City Table**

Id	Name	Percentage	Grade
1	Ram Kumar	57	IInd Division
2	Salman Khan	28	FAIL
3	Meera Khan	81	Merit
4	Sarita Kumari	43	IIIrd Division

Percentage  $\geq 80$  AND Percentage  $\leq 100$  → Merit

Percentage  $\geq 60$  AND Percentage  $< 80$  → 1st Division

Percentage  $\geq 45$  AND Percentage  $< 60$  → IInd Division

Percentage  $\geq 33$  AND Percentage  $< 45$  → IIIrd Division



## SELECT with CASE Clause Syntax

SELECT column1, column2,

CASE

WHEN Condition1 THEN result1

WHEN Condition2 THEN result2

WHEN Condition3 THEN result3

ELSE result alias\_name

END AS alias\_name

FROM table\_name;

The screenshot shows the MySQL Workbench interface. On the left, the Navigator pane displays the 'SCHEMAS' section with 'student' selected. Under 'student', there are tables named 'city', 'courses', and 'students'. The 'students' table is currently selected. On the right, the main workspace shows a query editor window with the following SQL query:

```
1 • | SELECT * FROM student.students;
```

Below the query editor is a results grid titled 'Result Grid'. The grid displays the following data:

	id	name	percentage	age	gender	city	courses
1	1	Ram Kumar	45	19	M	1	1
2	2	Sarita Kumari	85	22	F	2	2
3	3	Salman Khan	29	20	M	1	1
4	4	Juhi Chawla	47	18	F	3	1
5	5	Anil Kapoor	74	22	M	1	3
6	6	John Abraham	64	21	M	2	2
7	7	Shahid Kapoor	120	20	M	1	3

Students > students

```
1 • SELECT id, name, percentage,
2 • CASE
3 •     WHEN percentage >= 80 AND percentage <= 100 THEN "Merit"
4 •     WHEN percentage >= 60 AND percentage < 80 THEN "Ist Division"
5 •     WHEN percentage >= 45 AND percentage < 60 THEN "IIInd Division"
6 •     WHEN percentage >= 33 AND percentage < 45 THEN "IIIInd Division"
7 •     WHEN percentage < 33 THEN "Fail"
8 •     ELSE "Not Correct %"
9 • END AS Grade
10 FROM students;
```

Result Grid | Filter Results: [ ] | Export: [ ] | Wrap Cell Content: [ ]

	id	name	percentage	Grade
1	1	Ram Kumar	45	IIInd Division
2	2	Sarita Kumari	85	Merit
3	3	Salman Khan	29	Fail
4	4	Juhি Chawla	47	IIInd Division
5	5	Anil Kapoor	74	Ist Division
6	6	John Abraham	64	Ist Division
7	7	Shahid Kapoor	120	Not Correct %

case with update

students x students

```

1 UPDATE students SET
2   percentage = (CASE id
3     WHEN 3 THEN 39
4     WHEN 7 THEN 62
5   END)
6 WHERE id IN (3, 7);

```

Result Grid | Filter Rows: [ ] | Export: | Wrap Cell Content: [ ]

	id	name	percentage	Grade
>	1	Ram Kumar	45	IInd Division
	2	Sarita Kumari	85	Merit
	3	Salman Khan	29	Fail
	4	Juhi Chawla	47	IInd Division
	5	Anil Kapoor	74	Ist Division
	6	John Abraham	64	Ist Division
	7	Shahid Kapoor	120	Not Correct %

students x students

```

1 • SELECT * FROM student.students;

```

Result Grid | Filter Rows: [ ] | Edit: | Export/Import: | Wrap Cell Content: [ ]

	id	name	percentage	age	gender	city	courses
>	1	Ram Kumar	45	19	M	1	1
	2	Sarita Kumari	85	22	F	2	2
	3	Salman Khan	39	20	M	1	1
	4	Juhi Chawla	47	18	F	3	1
	5	Anil Kapoor	74	22	M	1	3
	6	John Abraham	64	21	M	2	2
	7	Shahid Kapoor	62	20	M	1	3

## 29. MySQL Arithmetic Functions



### List of Arithmetic Functions in MySQL

- PI()
- ROUND()
- CEIL()
- FLOOR()
- POW()
- SQRT()
- RAND()
- ABS()
- SIGN()
- SIN()
- COS()
- TAN()
- ASIN()
- ACOS()
- ATAN()
- ATAN2()
- COT()
- RADIANS()



### MySQL Arithmetic Functions

ABS()	FLOOR()	CEIL()	ROUND()
-6.2	6.2	4.3	4.8



### MySQL Arithmetic Functions

$$\sqrt{4} = 2$$

POW(base, exp)

$$(4)^3$$

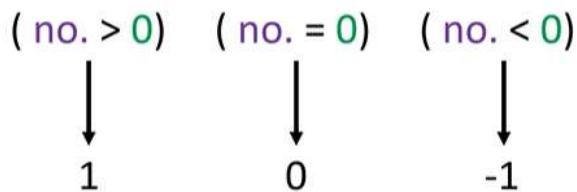
SQRT(number)

$$4 \times 4 \times 4 = 12$$



## MySQL Arithmetic Functions

### SIGN(number)



## MySQL Arithmetic Functions

### RAND()

Random Number  
Between 0 and 1

Practical

The screenshot shows the MySQL Workbench interface. The Navigator pane on the left displays the database schema, including the 'student' schema which contains tables like 'city', 'courses', and 'students'. The SQL Editor pane at the top contains the following query:

```
1  SELECT 5 + 6 AS Total;
```

The Result Grid pane at the bottom shows the output of the query:

Total
11

Navigator

SQL File 15\* × students

Limit to 1000 rows

1    **SELECT 15 / 6 AS Total;**

Result Grid | Filter Rows: [ ] | Export: [ ] | Wrap Cell Content: [ ]

Total
2.5000

Navigator

SQL File 15\* × students

Limit to 1000 rows

1    **SELECT 15 DIV 6 AS Total;**

I

Result Grid | Filter Rows: [ ] | Export: [ ] | Wrap Cell Content: [ ]

Total
3

Navigator

SCHEMAS

Filter objects

student

Tables

city

courses

students

Views

Stored Procedures

Functions

yb

Administration Schemas

Information

Schema: student

SQL File 15\* students

Limit to 1000 rr

```
1 SELECT id, name, percentage
2 FROM students;
```

Result Grid | Filter Rows: | Edit: | Export/Import: |

	id	name	percentage
1	1	Ram Kumar	45
2	2	Sarita Kumari	85
3	3	Salman Khan	39
4	4	Juhi Chawla	47
5	5	Anil Kapoor	74
6	6	John Abraham	64
7	7	Shahid Kapoor	62

Navigator

SCHEMAS

Filter objects

student

Tables

city

courses

students

Views

Stored Procedures

Functions

yb

Administration Schemas

Information

Schema: student

SQL File 15\* students

Limit to 1000 rr

```
1 SELECT id, name, (percentage + 5)
2 FROM students;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	id	name	(percentage + 5)
1	1	Ram Kumar	50
2	2	Sarita Kumari	90
3	3	Salman Khan	44
4	4	Juhi Chawla	52
5	5	Anil Kapoor	79
6	6	John Abraham	69
7	7	Shahid Kapoor	67

Navigator SQL File 15\* x students

SCHEMAS

Filter objects

student

Tables

city

courses

students

Views

Stored Procedures

Functions

yb

Administration Schemas

Information

Schema: student

```
1 SELECT id, name, (percentage * 5) AS "NEW Percentage"
2 FROM students;
```

Result Grid | Filter Rows: Export: Wrap Cell Contents:

	id	name	Percentage
1	1	Ram Kumar	225
2	2	Sarita Kumari	425
3	3	Salman Khan	195
4	4	Juhi Chawla	235
5	5	Anil Kapoor	370
6	6	John Abraham	320
7	7	Shahid Kapoor	310

4 • **SELECT PI();**

```
Result Grid | Filter Rows: Export:
```

PI()
3.141593

4 • **SELECT ROUND(4.51);**

```
Result Grid | Filter Rows: Export: Wrap Cell Contents:
```

ROUND(4.51)
5

4 • **SELECT ROUND(1234.987,2);**

```
Result Grid | Filter Rows: Export: Wrap Cell Contents:
```

ROUND(1234.987,2)
1234.99

4 • **SELECT CEIL(4.56);**

```
Result Grid | Filter Rows: Export: Wrap Cell Contents:
```

CEIL(4.56)
5

4 • **SELECT FLOOR(4.40);**

<	
Result Grid   Filter Rows: [ ] Export: [ ]   Wrap Cell Content: [ ]	
FLOOR(4.40)	
4	

4 • **SELECT POW(2,2);**

<	
Result Grid   Filter Rows: [ ] Export: [ ]   Wrap Cell Content: [ ]	
FLOOR(4.40)	
4	

**2 × 2 = 4**

4 • **SELECT POW(2,3);**

<	
Result Grid   Filter Rows: [ ] Export: [ ]   Wrap Cell Content: [ ]	
POW(2,3)	
8	

$$2 \times 2 \times 2 = 8$$

4 • **SELECT SQRT(5);**

<	
Result Grid   Filter Rows: [ ] Export: [ ]	
SQRT(5)	
2.23606797749979	

```
4 • SELECT ROUND(SQRT(5));
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content
ROUND(SQRT(5))				
2				

```
4 • SELECT RAND() * 100;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content
RAND() * 100				
59.51049081935593				

```
4 • SELECT ROUND(RAND() * 100);
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
ROUND(RAND() * 100)				
38				value between 7 to 12

```
4 • SELECT FLOOR(7 + (RAND() * 6));
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
FLOOR(7 + (RAND() * 6))				
12				

```

1   SELECT id, name, percentage, RAND()
2   FROM students;
3
4 •  SELECT FLOOR(1 + (RAND() * 5));

```

Result Grid | Filter Rows: [ ] | Export: [ ] | Wrap Cell Content: [ ]

	<b>id</b>	<b>name</b>	<b>percentage</b>	<b>RAND()</b>
▶	1	Ram Kumar	45	0.446604506528568
	2	Sarita Kumari	85	0.21245711130318895
	3	Salman Khan	39	0.7224720676638857
	4	Juhi Chawla	47	0.9749890351435068
	5	Anil Kapoor	74	0.7075290034595215
	6	John Abraham	64	0.6126779426007326
	7	Shahid Kapoor	62	0.9408027333587433

```

1   SELECT id, name, percentage
2   FROM students ORDER BY RAND();
3
4 •  SELECT FLOOR(1 + (RAND() * 5));

```

Result Grid | Filter Rows: [ ] | Edit: [ ] | Export/Import: [ ]

	<b>id</b>	<b>name</b>	<b>percentage</b>
▶	1	Ram Kumar	45
	3	Salman Khan	39
	6	John Abraham	64
	7	Shahid Kapoor	62
	4	Juhi Chawla	47
	2	Sarita Kumari	85
	5	Anil Kapoor	74
*	NULL	NULL	NULL

4 • `SELECT ABS(-56);`

Result Grid		Filter Rows:	Export:
	ABS(-56)		
▶	56		

4 • `SELECT SIGN(-25);`

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	SIGN(-25)			
▶	-1			

4 • `SELECT SIGN(-3.25);`

Result Grid		Filter Rows:	Export:	Wrap Cell
	SIGN(-3.25)			
▶	-1			

## 30. MySQL String Functions-Part1



## List of String Functions in MySQL

- `UPPER()` / `UCASE()`
- `LOWER()` / `LCASE()`
- `LENGTH()`
- `CHAR_LENGTH()`
- `CONCAT()`
- `CONCAT_WS()`
- `LTRIM()`
- `RTRIM()`
- `TRIM()`
- `POSITION()`
- `LOCATE()`
- `INSTR()`
- `SUBSTRING()` / `SUBSTR()`
- `MID()`
- `SUBSTRING_INDEX()`
- `LEFT()`
- `RIGHT()`
- `LPAD()`
- `RPAD()`
- `SPACE()`
- `REVERSE()`
- `REPEAT()`
- `REPLACE()`
- `STRCMP()`
- `FIELD()`
- `FIND_IN_SET()`
- `FORMAT()`
- `HEX(str)`

The screenshot shows the phpMyAdmin interface. In the left sidebar, under the 'SCHEMAS' section, the 'student' schema is selected. Inside 'student', there are tables named 'city', 'courses', and 'students'. The 'students' table is currently selected. In the main area, a SQL query is entered in the query editor:

```
1  SELECT * FROM student.students;
```

The result grid displays the following data:

	id	name	percentage	age	gender	city	courses
1	1	Ram Kumar	45	19	M	1	1
2	2	Santa Kumari	85	22	F	2	2
3	3	Salman Khan	39	20	M	1	1
4	4	Juhi Chawla	47	18	F	3	1
5	5	Anil Kapoor	74	22	M	1	3
6	6	John Abraham	64	21	M	2	2
7	7	Shahid Kapoor	62	20	M	1	3
	HULL	HULL	HULL	HULL	HULL	HULL	HULL

Practical

students x students

1 • `SELECT id, UPPER(name) AS Name , percentage`  
2 `FROM students;`  
3

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

	<b>id</b>	<b>Name</b>	<b>percentage</b>
1	1	RAM KUMAR	45
2	2	SARITA KUMARI	85
3	3	SALMAN KHAN	39
4	4	JUHI CHAWLA	47
5	5	ANIL KAPOOR	74
6	6	JOHN ABRAHAM	64
7	7	SHAHID KAPOOR	62

1 • `SELECT id, UCASE(name) AS Name , percentage`  
2 `FROM students;`  
3

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

	<b>id</b>	<b>Name</b>	<b>percentage</b>
1	1	RAM KUMAR	45
2	2	SARITA KUMARI	85
3	3	SALMAN KHAN	39
4	4	JUHI CHAWLA	47
5	5	ANIL KAPOOR	74
6	6	JOHN ABRAHAM	64
7	7	SHAHID KAPOOR	62

```
1 • SELECT id, LOWER(name) AS Name , percentage  
2   FROM students;  
3
```

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

	id	Name	percentage
▶	1	ram kumar	45
2	sarita kumari	85	
3	salman khan	39	
4	juhi chawla	47	
5	anil kapoor	74	
6	john abraham	64	
7	shahid kapoor	62	

```
1 • SELECT id, LCASE(name) AS Name , percentage  
2   FROM students;  
3
```

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

	id	Name	percentage
▶	1	ram kumar	45
2	sarita kumari	85	
3	salman khan	39	
4	juhi chawla	47	
5	anil kapoor	74	
6	john abraham	64	
7	shahid kapoor	62	

```
students x students  
1 • SELECT id, name ,CHARACTER_LENGTH(name) AS Characters  
2   FROM students;  
3
```

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

	id	name	Characters
▶	1	Ram Kumar	9
2	Sarita Kumari	13	
3	Salman Khan	11	
4	Juhi Chawla	11	
5	Anil Kapoor	11	
6	John Abraham	12	
7	Shahid Kapoor	13	

students x students

```
1 • SELECT id, name , LENGTH(name) AS Characters
2 FROM students;
3
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

	id	name	Characters
1	1	Ram Kumar	9
2	2	Sarita Kumari	13
3	3	Salman Khan	11
4	4	Juhi Chawla	11
5	5	Anil Kapoor	11
6	6	John Abraham	12
7	7	Shahid Kapoor	13

students x students

```
1 • SELECT id, CONCAT(name, percentage) AS Name
2 FROM students;
3
```

I

Result Grid | Filter Rows: Export: Wrap Cell Content:

	id	Name
1	1	Ram Kumar45
2	2	Sarita Kumari85
3	3	Salman Khan39
4	4	Juhi Chawla47
5	5	Anil Kapoor74
6	6	John Abraham64
7	7	Shahid Kapoor62

students x students

1 • **SELECT id, CONCAT(name, " ",percentage) AS Name**  
2   **FROM students;**  
3

Result Grid | Filter Rows: Export: Wrap Cell Content:

	id	Name
1	1	Ram Kumar 45
2	2	Sarita Kumari 85
3	3	Salman Khan 39
4	4	Juhi Chawla 47
5	5	Anil Kapoor 74
6	6	John Abraham 64
7	7	Shahid Kapoor 62

4 • **SELECT CONCAT("Yahoo", "Baba", "Youtube", "Channel") AS Name;**

Result Grid | Filter Rows: Export: Wrap Cell Content:

	Name
1	YahooBabaYoutubeChannel

4 • **SELECT CONCAT\_WS(" - ", "Baba", "Youtube", "Channel") AS Name;**

Result Grid | Filter Rows: Export: Wrap Cell Content:

	Name
1	Baba - Youtube - Channel

4 • **SELECT LTRIM(" Yahoo Baba ") AS Name;**

Result Grid | Filter Rows: Export: Wrap Cell Content:

	Name
1	Yahoo Baba

4 • **SELECT RTRIM(" Yahoo Baba ") AS Name;**

Result Grid | Filter Rows: Export: Wrap Cell Content:

	Name
1	Yahoo Baba

```
4 • SELECT TRIM("    Yahoo Baba      ") AS Name;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Name				

▶ Yahoo Baba

5

```
4 • SELECT POSITION("Baba" IN "Yahoo Baba") AS Name;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Name				

▶ 7

```
4 • SELECT POSITION("Baba" IN "Yahoo Baba Baba") AS Name;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Name				

▶ 7

```
4 • SELECT INSTR("Yahoo Baba Baba", "a") AS Name;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Name				

▶ 2

```
4 • SELECT LOCATE("a","Yahoo Baba Baba",3) AS Name;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Name				

▶ 8

## 31. MySQL String Functions - Part 2



## List of String Functions in MySQL

- UPPER() / UCASE()
- LOWER() / LCASE()
- LENGTH()
- CHAR\_LENGTH()
- CONCAT()
- CONCAT\_WS()
- LTRIM()
- RTRIM()
- TRIM()
- POSITION()
- LOCATE()
- INSTR()
- SUBSTRING() / SUBSTR()
- MID()
- SUBSTRING\_INDEX()
- LEFT()
- RIGHT()
- LPAD()
- RPAD()
- SPACE()
- REVERSE()
- REPEAT()
- REPLACE()
- STRCMP()
- FIELD()
- FIND\_IN\_SET()
- FORMAT()
- HEX(str)

The screenshot shows a MySQL Workbench interface. The title bar says "students x students". The toolbar includes icons for file operations, search, and database navigation. A dropdown menu says "Limit to 1000 rows". Below the toolbar, a query editor window contains the following SQL code:

```
1 • SELECT SUBSTRING("Yahoo Baba",3);
```

Below the query editor is a results grid. The first row has a header "Result Grid" and includes "Filter Rows", "Export", and "Wrap Cell Content" buttons. The results grid displays one row of data:

SUBSTRING("Yahoo Baba",3)
hoo Baba

students x students

Limit to 1000 rows | Filter Rows: | Export: | Wrap Cell Content: |

```
1 • SELECT SUBSTRING("Yahoo Baba",3,6) AS Name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Name
hoo Ba

students x students

Limit to 1000 rows | Filter Rows: | Export: | Wrap Cell Content: |

```
1 • SELECT SUBSTRING("Yahoo Baba",-6,3) AS Name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Name
o Ba

students x students

1 • `SELECT SUBSTR("Yahoo Baba",-6,3) AS Name;`

Result Grid | Filter Rows: Export: Wrap Cell Content:

Name
o B

students x students

1 `SELECT SUBSTRING_INDEX("www.yahoobaba.net",".",1) AS Name;`

Result Grid | Filter Rows: Export: Wrap Cell Content:

Name
www

students x students

1 `SELECT SUBSTRING_INDEX("www.yahoobaba.net","o",2) AS Name;`

Result Grid | Filter Rows: Export: Wrap Cell Content:

Name
www.yaho

students x students

1    **SELECT LEFT("Yahoo Baba",5) AS Name;**

Result Grid | Filter Rows: [ ] Export: Wrap Cell Content:

Name
Yahoo

students x students

1    **SELECT RIGHT("Yahoo Baba",5) AS Name;**

Result Grid | Filter Rows: [ ] Export: Wrap Cell Content:

Name
Baba

students x students

1    **SELECT RPAD("Yahoo Baba",20,"-") AS Name;**

Result Grid | Filter Rows: [ ] Export: Wrap Cell Content:

Name
Yahoo Baba-----

students × students

1    **SELECT LPAD("Yahoo Baba",20,"\*") AS Name;**

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

Name
*****Yahoo Baba

students × students

1    **SELECT SPACE(100) AS Name;**

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

Name
...

students × students

1    **SELECT REVERSE("Yahoo Baba") AS Name;**

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

Name
abaB oohaY

students x students

Limit to 1000 r ▾ ⚡ Filter Rows! Export: Wrap Cell Content: ↴

```
1 SELECT REPLACE("Yahoo Baba", "Baba", "Wow") AS Name;
```

Result Grid | Filter Rows! Export: Wrap Cell Content: ↴

Name
Yahoo Wow

students x students

Limit to 1000 r ▾ ⚡ Filter Rows! Export: Wrap Cell Content: ↴

```
1 SELECT REPLACE("Yahoo Baba Yoo Baba", "Baba", "Wow") AS Name;
```

Result Grid | Filter Rows! Export: Wrap Cell Content: ↴

Name
Yahoo Wow Yoo Wow

students x students

Limit to 1000 r ▾ ⚡ Filter Rows! Export: Wrap Cell Content: ↴

```
1 SELECT STRCMP("Yahoo Baba", "yahoo ") AS Name;
```

Result Grid | Filter Rows! Export: Wrap Cell Content: ↴

Name
1

students x students

1    **SELECT** STRCMP("Yahoo ", "yahoo baba") **AS** Name;

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Name	-1			

students x students

1    **SELECT** FIELD("a", "X", "A", "k") **AS** Name;

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Name	2			

students x students

1    **SELECT** FIELD(5, 0 , 1, 2 , 3, 4 ,5) **AS** Name;

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Name	6			

students x students

```
1  SELECT FIELD("ram", "Ram", "Mohan", "Shyam") AS Name;
```

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

Name
1

students x students

```
1  SELECT FIND_IN_set("ram", "Ram, Mohan, Shyam") AS Name;
```

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

Name
1

students x students

```
1  SELECT FIND_IN_set("Mohan", "Ram, Mohan, Shyam") AS Name;
```

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

Name
0

students x students

```
1  SELECT FIND_IN_set("Mohan", "Ram, Mohan, Shyam") AS Name;
```

I

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

Name
2

The image shows two separate MySQL query windows in a tool like MySQL Workbench.

**Query 1:**

```
1 SELECT FORMAT(255.3568,2) AS Value;
```

**Result:**

Value
255.36

**Query 2:**

```
1 SELECT HEX("Yahoo Baba") AS Value;
```

**Result:**

Value
5961686F6F2042616261

## 32. MySQL Date Functions - Part 1



## List of Date Functions in MySQL

- CURDATE
- CURRENT\_DATE
- SYSDATE
- NOW
- LAST\_DAY
- DAY
- DAYNAME
- DAYOFMONTH
- DAYOFWEEK
- DAYOFYEAR
- WEEK
- WEEKDAY
- WEEKOFYEAR
- YEAR
- YEARWEEK
- EXTRACT
- DATE\_ADD
- ADDDATE
- MAKEDATE
- DATE\_SUB
- SUBDATE
- DATEDIFF
- TO\_DAYS
- FROM\_DAYS
- PERIOD\_ADD
- PERIOD\_DIFF
- DATE\_FORMAT
- STR\_TO\_DATE

The screenshot shows the phpMyAdmin interface with three separate query windows. The top window displays the result of a query using CURRENT\_DATE(). The middle window displays the result of a query using CURDATE(). Both results show the date 2019-12-12.

**Top Window:**

```
1 SELECT CURRENT_DATE();
```

CURRENT_DATE()
2019-12-12

**Middle Window:**

```
1 SELECT CURDATE();
```

CURDATE()
2019-12-12

**Bottom Window:**

```
1 SELECT CURDATE();
```

CURDATE()
2019-12-12

The screenshot shows the Oracle SQL Developer interface. A query window titled "students" contains the SQL statement: "1 SELECT SYSDATE();". The result grid shows one row with the value "SYSDATE()" and its corresponding timestamp "2019-12-12 04:56:03".

```
1   SELECT SYSDATE();
```

	SYSDATE()
▶	2019-12-12 04:56:03

The screenshot shows the Oracle SQL Developer interface. A query window titled "students" contains the SQL statement: "1 SELECT NOW();". The result grid shows one row with the value "NOW()" and its corresponding timestamp "2019-12-12 04:56:33".

```
1   SELECT NOW();
```

	NOW()
▶	2019-12-12 04:56:33

The screenshot shows the Oracle SQL Developer interface. A query window titled "students" contains the SQL statement: "1 SELECT DATE('2019-10-15 09:34:21');". The result grid shows one row with the value "DATE('2019-10-15 09:34:21')". Below it, another row shows the timestamp "2019-10-15".

```
1   SELECT DATE('2019-10-15 09:34:21');
```

	DATE('2019-10-15 09:34:21')
▶	2019-10-15

students x students

1    **SELECT DATE("2019-10-15 09:34:21") AS DATE;**

The screenshot shows the MySQL Workbench interface with a single query window. The query is: `SELECT DATE("2019-10-15 09:34:21") AS DATE;`. The result grid displays one row with the column name 'DATE' and the value '2019-10-15'.

DATE
2019-10-15

students x students

1    **SELECT MONTH("2019-10-15 09:34:21") AS DATE;**

The screenshot shows the MySQL Workbench interface with a single query window. The query is: `SELECT MONTH("2019-10-15 09:34:21") AS DATE;`. The result grid displays one row with the column name 'DATE' and the value '10'.

DATE
10

students x students

1    **SELECT MONTHNAME("2019-10-15 09:34:21") AS DATE;**

The screenshot shows the MySQL Workbench interface with a single query window. The query is: `SELECT MONTHNAME("2019-10-15 09:34:21") AS DATE;`. The result grid displays one row with the column name 'DATE' and the value 'October'.

DATE
October

students x students

1    **SELECT DAYNAME("2019-10-15 09:34:21") AS DATE;**

The screenshot shows the MySQL Workbench interface with a single query window. The query is: `SELECT DAYNAME("2019-10-15 09:34:21") AS DATE;`. The result grid displays one row with the column name 'DATE' and the value 'Wednesday'.

DATE
Wednesday

students x students

1    **SELECT DAYNAME("2019-10-15 09:34:21") AS DATE;**

The screenshot shows the MySQL Workbench interface with a single query window. The query is: `SELECT DAYNAME("2019-10-15 09:34:21") AS DATE;`. The result grid displays one row with the column name 'DATE' and the value 'Wednesday'.

DATE
Wednesday

students x students

1    **SELECT YEAR("2019-10-15 09:34:21") AS DATE;**

I

< Result Grid | Filter Rows: Export: Wrap Cell Contents: □

DATE
2019

students x students

1 • **SELECT id, name, dob, YEAR(dob) AS Year FROM student.students;**

2

3 • **SELECT YEAR("2019-10-15 09:34:21") AS DATE;**

< Result Grid | Filter Rows: Export: Wrap Cell Contents: □

id	name	dob	Year
1	Ram Kumar	2000-05-10	2000
2	Sarita Kumari	1997-02-03	1997
3	Salman Khan	1999-11-12	1999
4	Juhi Chawla	2001-07-16	2001
5	Anil Kapoor	1997-01-03	1997
6	John Abraham	1998-08-10	1998
7	Shahid Kapoor	1999-12-08	1999

3 • **SELECT QUARTER("2019-10-15 09:34:21") AS DATE;**

I

< Result Grid | Filter Rows: Export: Wrap Cell Contents: □

DATE
4

```
3 • SELECT DAYNAME("2019-03-15 09:34:21") AS DATE;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
DATE	Friday			

```
3 • SELECT DAYOFWEEK("2019-03-15 09:34:21") AS DATE;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
DATE	6			

```
3 • SELECT DAYOFYEAR("2019-06-15 09:34:21") AS DATE;
```

I

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
DATE	166			

```
3 • SELECT WEEK("2019-03-15 09:34:21") AS DATE;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
DATE	10			

```
3 • SELECT WEEKDAY("2019-03-15 09:34:21") AS DATE;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
DATE	4			

```
3 • SELECT YEARWEEK("2019-06-15 09:34:21") AS DATE;
```

Result Grid	
DATE	
201923	

```
3 • SELECT LAST_DAY("2019-06-15 09:34:21") AS DATE;
```

Result Grid	
DATE	
2019-06-30	

```
3 • SELECT EXTRACT(MONTH FROM "2019-06-15 09:34:21") AS DATE;
```

Result Grid	
DATE	
2	



## List of Parameters with Extract Function

- MICROSECOND
- SECOND
- MINUTE
- HOUR
- DAY
- WEEK
- MONTH
- QUARTER
- YEAR
- SECOND\_MICROSECOND
- MINUTE\_MICROSECOND
- MINUTE\_SECOND
- HOUR\_MICROSECOND
- HOUR\_SECOND
- HOUR\_MINUTE
- DAY\_MICROSECOND
- DAY\_SECOND
- DAY\_MINUTE
- DAY\_HOUR
- YEAR\_MONTH

```
3 • SELECT EXTRACT(MINUTE FROM "2019-02-15 09:34:21") AS DATE;
```

Result Grid	
	DATE
▶	34

```
3 • SELECT EXTRACT(HOUR_MINUTE FROM "2019-02-15 09:34:21") AS DATE;
```

Result Grid	
	DATE
▶	934

## 33. MySQL Date Functions - Part 2



### List of Date Functions in MySQL

- CURDATE
- CURRENT\_DATE
- SYSDATE
- NOW
- LAST\_DAY
- DAY
- DAYNAME
- DAYOFMONTH
- DAYOFWEEK
- DAYOFYEAR
- WEEK
- WEEKDAY
- WEEKOFYEAR
- YEAR
- YEARWEEK
- EXTRACT
- DATE\_ADD
- ADDDATE
- MAKEDATE
- DATE\_SUB
- SUBDATE
- DATEDIFF
- TO\_DAYS
- FROM\_DAYS
- PERIOD\_ADD
- PERIOD\_DIFF
- DATE\_FORMAT
- STR\_TO\_DATE

students	
	students
1	SELECT ADDDATE("2019-06-15", INTERVAL 10 DAY) AS Date;
	Date
▶	2019-06-25

students > students

```
1 SELECT ADDDATE("2019-06-15", INTERVAL 1 MONTH) AS Date;
```

Date
2019-07-15

- ## MySQL List of addunits
- MICROSECOND
  - SECOND
  - MINUTE
  - HOUR
  - DAY
  - WEEK
  - MONTH
  - QUARTER
  - YEAR
  - SECOND\_MICROSECOND
  - MINUTE\_MICROSECOND
  - MINUTE\_SECOND
  - HOUR\_MICROSECOND
  - HOUR\_SECOND
  - HOUR\_MINUTE
  - DAY\_MICROSECOND
  - DAY\_SECOND
  - DAY\_MINUTE
  - DAY\_HOUR
  - YEAR\_MONTH

students > students

```
1 SELECT ADDDATE("2019-06-15", INTERVAL 5000 MINUTE) AS Date;
```

Date
2019-06-18 11:20:00

students x students

```
1 SELECT DATE_ADD("2019-06-15", INTERVAL 5000 MINUTE) AS Date;
```

Result Grid	
Filter Rows:	
Date	2019-06-18 11:20:00

```
3 • SELECT MAKEDATE(2016, 3);
```

Result Grid	
Filter Rows:	
MAKEDATE(2016, 3)	2016-01-03

students x students

```
1 SELECT SUBDATE("2019-06-15", INTERVAL 1 DAY) AS Date;
```

Result Grid	
Filter Rows:	
Date	2019-06-14

students x students

```
1 SELECT DATEDIFF("2019-06-15", "2019-02-10") AS Date;
```

Result Grid	
Filter Rows:	
Date	125

students x students

```
1  SELECT FROM_DAYS("685000") AS Date;
2
3
```

<

Result Grid | Filter Rows: Export: Wrap Cell Content:

Date
1875-06-20

students x students

```
1  SELECT PERIOD_ADD("685000", 5) AS Date;
2
3
```

<

Result Grid | Filter Rows: Export: Wrap Cell Content:

Date
685005

students x students

```
1  SELECT PERIOD_DIFF("685000", "695000") AS Date;
2
3
```

<

Result Grid | Filter Rows: Export: Wrap Cell Content:

Date
-1200

students x students

```
1  SELECT DATE_FORMAT("2019-06-15", "%Y") AS Date;
2
3 |
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Date
2019



## List of Date Format

25/03/2020

### Date Format

Day

Month

Year

Week

%d (01 to 31)

%M (January)

%Y (2019)

%a (Mon)

%e (0 to 31)

%b (Jan to Dec)

%y (19)

%W (Monday)

%D (st, nd, rd or th)

%m (00 to 12)

%w (0 to 6)

%j (001 to 366)

%c (0 to 12)



## List of Time Format

02:30:27:00 PM

### Time Format

Hour

Minutes

Seconds

Microseconds

%h (01 to 12)

%i (00 to 59)

%s (00 to 59)

%f (000000 to 999999)

%H (00 to 23)

%g (1 to 12)

%G (0 to 23)

Meridiem

%p (AM or PM)

```
students x students
1  SELECT DATE_FORMAT("2019-06-15", "%d/%b/%Y") AS Date;
2
3
```

```
Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]
Date
15/Jun/2019
```

```
students x students
1  SELECT DATE_FORMAT("2019-06-15", "%d-%c-%y") AS Date;
2
3
```

```
Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]
Date
15-6-19
```

```
students x students
1  SELECT DATE_FORMAT("2019-06-15 02:30:50:20", "%d-%c-%y, %h:%i") AS Date;
2
3
```

```
Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]
Date
15-6-19, 02:30
```

```
students x students
1  SELECT STR_TO_DATE("July 10 2019", "%M %d %Y") AS Date;
2
3
```

```
Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]
Date
2019-07-10
```

## 34. MySQL Time Functions



## List of Time Functions in MySQL

- CURTIME()
- CURRENT\_TIMESTAMP
- LOCALTIME
- LOCALTIMESTAMP
- TIMESTAMP
- TIME
- TIMEDIFF
- HOUR
- MINUTE
- SECOND
- MICROSECOND
- ADDTIME
- SUBTIME
- MAKETIME
- TIME\_FORMAT
- SEC\_TO\_TIME
- TIME\_TO\_SEC

The screenshot shows the MySQL Workbench interface with a single query window titled 'students'. The query 'SELECT CURRENT\_TIME();' has been run, and the result is displayed in a grid:

CURRENT_TIME()
03:34:52

students x students

1    **SELECT CURTIME();**

<

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

CURTIME()
03:35:30

◀

students x students

1    **SELECT CURRENT\_TIMESTAMP();**

<

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

CURRENT_TIMESTAMP()
2019-12-16 03:36:13

students x students

1    SELECT LOCALTIME();

< | Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

LOCALTIME()
▶ 2019-12-16 03:37:42

students x students

1    SELECT LOCALTIMESTAMP();

< | Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

LOCALTIMESTAMP()
▶ 2019-12-16 03:38:12

students x students

1    **SELECT TIME("2019-06-15 13:15:20") AS Time;**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Time				

13:15:20

students x students

1    **SELECT MICROSECOND("2019-06-15 13:15:20") AS Time;**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Time				

0

students x students

1    **SELECT TIMEDIFF("18:10:11","13:15:20") AS Time;**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Time				

04:54:51

students x students

1    **SELECT ADDTIME("2019-06-15 05:30:20.000001","5:5.000003") AS Time;**

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Time				

2019-06-15 10:35:20.000004

students x students

```
1  SELECT ADDTIME("2019-06-15 05:30:20.000001","5 2:10:5.000003") AS Time;
```

I

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

Time
2019-06-20 07:40:25.000004

students x students

```
1  SELECT SUBTIME("2019-06-15 05:30:20.000001","5 2:10:5.000003") AS Time;
```

I

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

Time
2019-06-10 03:20:14.999998

students x students

```
1  SELECT MAKETIME(16,15,20) AS Time;
```

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

Time
16:15:20

students x students

```
1  SELECT TIMESTAMP("2019-06-15","13:15:20") AS Time;
```

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

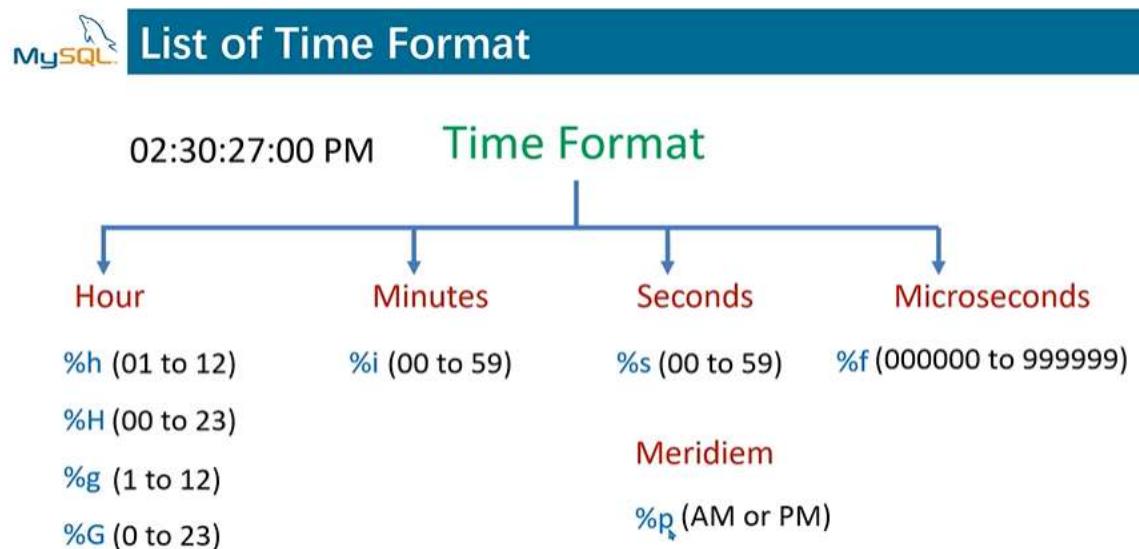
Time
2019-06-15 13:15:20

students > students

```
1  SELECT TIME_FORMAT("13:15:20", "%H") AS Time;
```

Result Grid | Filter Rows: [ ] | Export: [ ] | Wrap Cell Content: [ ]

Time
13



students > students

```
1  SELECT TIME_FORMAT("13:15:20", "%H %i %s") AS Time;
```

Result Grid | Filter Rows: [ ] | Export: [ ] | Wrap Cell Content: [ ]

Time
13 15 20

students x students

```
1  SELECT TIME_FORMAT("13:15:20", "%H-%i-%s") AS Time;
```

Time
13-15-20

students x students

```
1  SELECT TIME_FORMAT("13:15:20", "%H-%i-%s %p") AS Time;
```

Time
13-15-20 PM

students x students

```
1  SELECT TIME_TO_SEC("13:15:20") AS Time;
```

Time
47720

students x students

1    **SELECT SEC\_TO\_TIME("1") AS Time;**

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

Time
00:00:01.000000

students x students

1    **SELECT SEC\_TO\_TIME("5454") AS Time;**

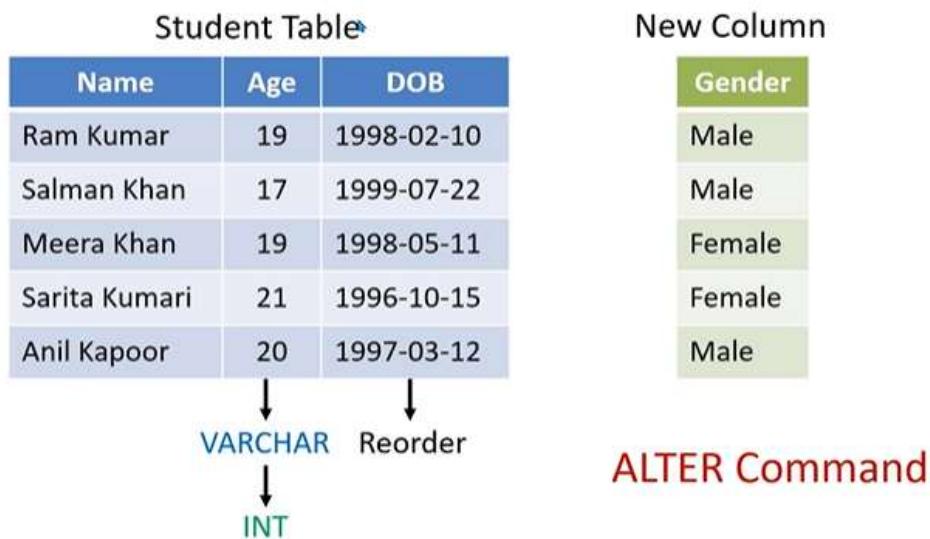
Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

Time
01:30:54.000000

## 35. MySQL ALTER



## How to Modify Tables in Database ?



## Features of MySQL ALTER Command :

- Add Column in a table
- Changing Data Type of a Column
- Change Column Name
- Adding Constraints to a Column
- Changing Column Position
- Delete Column
- Renaming Tables



## ALTER Syntax

Add Column

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

Modify Column

```
ALTER TABLE table_name  
MODIFY column_name datatype;
```

Delete Column

```
ALTER TABLE table_name  
DROP COLUMN column_name datatype;
```



## ALTER Syntax

Rename Column

```
ALTER TABLE table_name  
CHANGE column_name New_name datatype;
```

Rename Table

```
ALTER TABLE table_name  
RENAME new_table_name;
```

The screenshot shows the phpMyAdmin interface. The left sidebar shows the 'student' schema with tables like 'city', 'courses', and 'students'. The main area shows a query editor with the SQL command: `1 • SELECT * FROM student.students;`. Below it is a result grid displaying data from the 'students' table:

	id	name	percentage	dob	age	gender	city	courses
1	1	Ram Kumar	45	2000-05-10	19	M	1	1
2	2	Sarita Kumari	85	1997-02-03	22	F	2	2
3	3	Salman Khan	39	1999-11-12	20	M	1	1
4	4	Juhi Chawla	47	2001-07-16	18	F	3	1
5	5	Anil Kapoor	74	1997-01-03	22	M	1	3
6	6	John Abraham	64	1998-08-10	21	M	2	2
7	7	Shahid Kapoor	62	1999-12-08	20	M	1	3

we are adding new column

The screenshot shows the phpMyAdmin interface with the following details:

- Schemas:** student
- Tables:** city, courses, students
- Query:** 1 • `SELECT * FROM student.students;`
- Result Grid:** Displays 7 rows of data from the students table.

	id	name	percentage	dob	age	gender	city	courses	Email
▶	1	Ram Kumar	45	2000-05-10	19	M	1	1	
▶	2	Sarita Kumari	85	1997-02-03	22	F	2	2	
▶	3	Salman Khan	39	1999-11-12	20	M	1	1	
▶	4	Juhি Chawla	47	2001-07-16	18	F	3	1	
▶	5	Anil Kapoor	74	1997-01-03	22	M	1	3	
▶	6	John Abraham	64	1998-08-10	21	M	2	2	
▶	7	Shahid Kapoor	62	1999-12-08	20	M	1	3	

## reordering of column

The screenshot shows the phpMyAdmin interface with the following details:

- Schemas:** student
- Tables:** city, courses, students, views
- Current Table:** students
- Query:** 1 • `SELECT * FROM student.students;`
- Result Grid:** Displays 8 rows of data from the students table.

	<b>id</b>	<b>name</b>	<b>Email</b>	<b>percentage</b>	<b>dob</b>	<b>age</b>	<b>gender</b>	<b>city</b>	<b>courses</b>
1	1	Ram Kumar	HULL	45	2000-05-10	19	M	1	1
2	2	Sarita Kumari	HULL	85	1997-02-03	22	F	2	2
3	3	Salman Khan	HULL	39	1999-11-12	20	M	1	1
4	4	Juhি Chawla	HULL	47	2001-07-16	18	F	3	1
5	5	Anil Kapoor	HULL	74	1997-01-03	22	M	1	3
6	6	John Abraham	HULL	64	1998-08-10	21	M	2	2
7	7	Shahid Kapoor	HULL	62	1999-12-08	20	M	1	3

## checking datatype

Table Name: students  
 Charset/Collation: utf8mb4 utf8mb4\_0900\_ai\_ci

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
<input checked="" type="checkbox"/> id	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
<input checked="" type="checkbox"/> name	VARCHAR(50)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
<input checked="" type="checkbox"/> Email	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
<input checked="" type="checkbox"/> percentage	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input checked="" type="checkbox"/> dob	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input checked="" type="checkbox"/> age	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input checked="" type="checkbox"/> gender	VARCHAR(1)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input checked="" type="checkbox"/> city	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input checked="" type="checkbox"/> courses	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

changing datatype

1 • `SELECT * FROM student.students;`

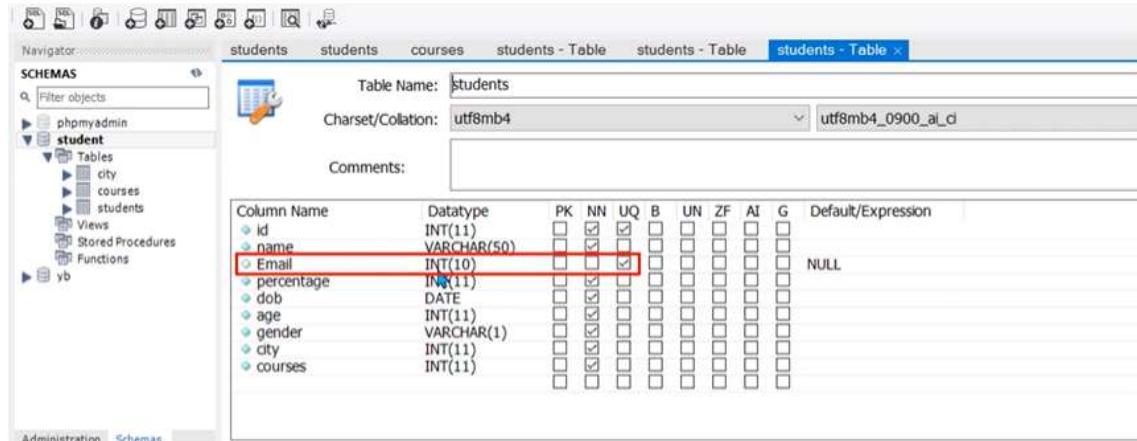
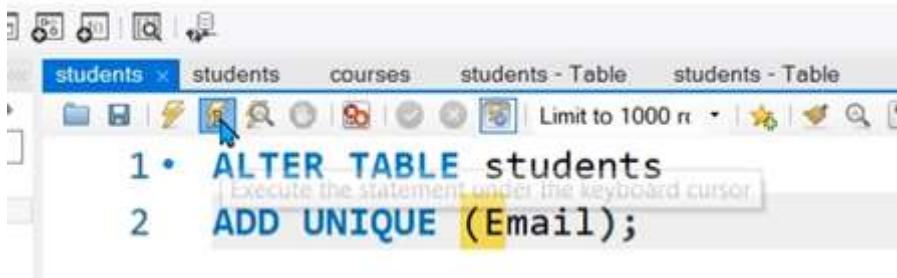
	id	name	Email	percentage	dob	age	gender	city	courses
1	1	Ram Kumar	NULL	45	2000-05-10	19	M	1	1
2	2	Sarita Kumari	NULL	85	1997-02-03	22	F	2	2
3	3	Salman Khan	NULL	39	1999-11-12	20	M	1	1
4	4	Juhি Chawla	NULL	47	2001-07-16	18	F	3	1
5	5	Anil Kapoor	NULL	74	1997-01-03	22	M	1	3
6	6	John Abraham	NULL	64	1998-08-10	21	M	2	2
7	7	Shahid Kapoor	NULL	62	1999-12-08	20	M	1	3

Table Name: students  
 Charset/Collation: utf8mb4 utf8mb4\_0900\_ai\_ci

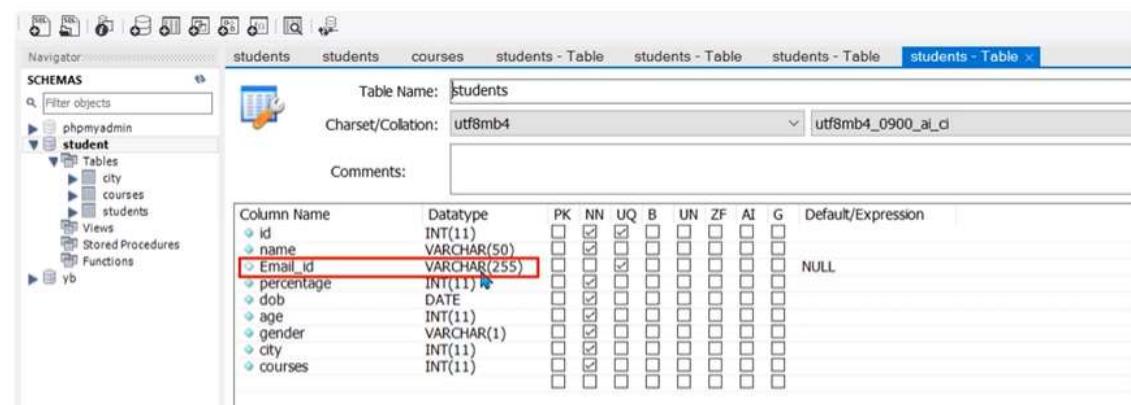
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
<input checked="" type="checkbox"/> id	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
<input checked="" type="checkbox"/> name	VARCHAR(50)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
<input checked="" type="checkbox"/> Email	INT(10)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
<input checked="" type="checkbox"/> percentage	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input checked="" type="checkbox"/> dob	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input checked="" type="checkbox"/> age	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input checked="" type="checkbox"/> gender	VARCHAR(1)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input checked="" type="checkbox"/> city	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input checked="" type="checkbox"/> courses	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

adding unique constant in email column



changing name of column



```
1 • ALTER TABLE students
2 • DROP COLUMN Email_id;
```

drop column

```
1 • SELECT * FROM student.students;
```

	id	name	percentage	dob	age	gender	city	courses
1	1	Ram Kumar	45	2000-05-10	19	M	1	1
2	2	Sarita Kumari	85	1997-02-03	22	F	2	2
3	3	Salman Khan	39	1999-11-12	20	M	1	1
4	4	Juhi Chawla	47	2001-07-16	18	F	3	1
5	5	Anil Kapoor	74	1997-01-03	22	M	1	3
6	6	John Abraham	64	1998-08-10	21	M	2	2
7	7	Shahid Kapoor	62	1999-12-08	20	M	1	3

```
1 • ALTER TABLE students
2 • RENAME studentss;
```

rename of column

The screenshot shows the MySQL Workbench interface with a query editor containing the SQL command:

```
1 • SELECT * FROM student.studentss;
```

The results are displayed in a grid:

	<b>id</b>	<b>name</b>	<b>percentage</b>	<b>dob</b>	<b>age</b>	<b>gender</b>	<b>city</b>	<b>courses</b>
▶	1	Ram Kumar	45	2000-05-10	19	M	1	1
	2	Sarita Kumari	85	1997-02-03	22	F	2	2
	3	Salman Khan	39	1999-11-12	20	M	1	1
	4	Juhি Chawla	47	2001-07-16	18	F	3	1
	5	Anil Kapoor	74	1997-01-03	22	M	1	3
	6	John Abraham	64	1998-08-10	21	M	2	2
	7	Shahid Kapoor	62	1999-12-08	20	M	1	3

problem of autoincrement

The screenshot shows the MySQL Workbench interface with a query editor containing the SQL command:

```
1 • SELECT * FROM student.courses;
```

The results are displayed in a grid:

	<b>course_id</b>	<b>course_name</b>
	1	Btech
	2	BCA
	3	MBA
	15	BA
▶	16	BCOM

The screenshot shows the MySQL Workbench interface with a query result grid for the courses table:

	<b>course_id</b>	<b>course_name</b>
	1	Btech
	2	BCA
	3	MBA
	15	BA
▶	16	BCOM

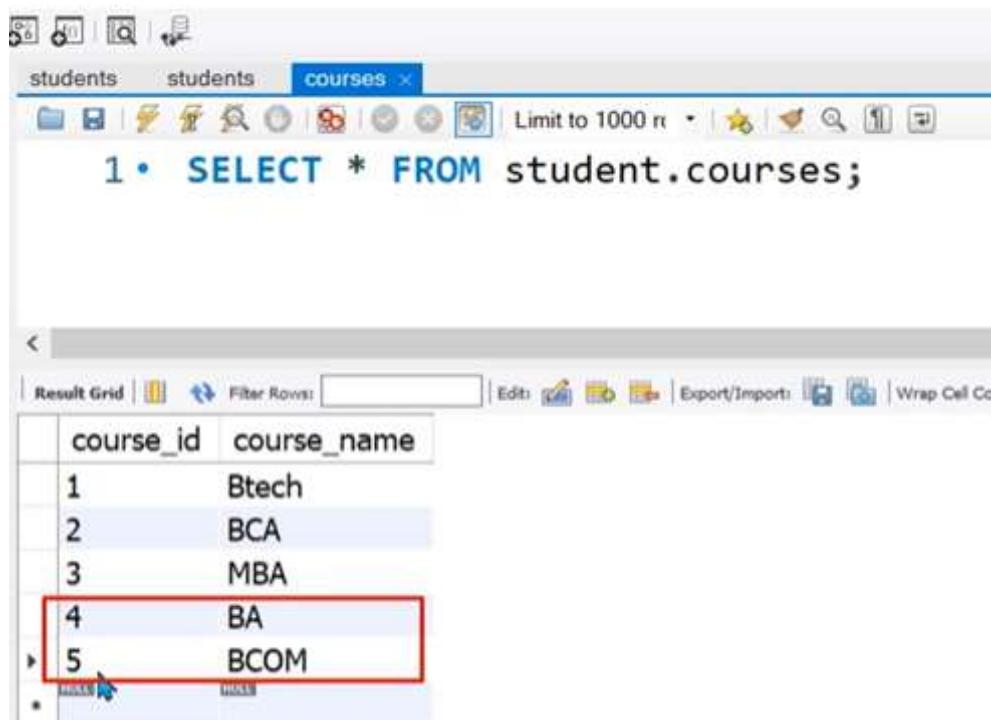
solution through

alter with autoincrement



students    students    courses

1 • ALTER TABLE courses  
2 AUTO\_INCREMENT = 4;



students    students    courses

1 • SELECT \* FROM student.courses;

course_id	course_name
1	Btech
2	BCA
3	MBA
4	BA
5	BCOM

## 36. MySQL DROP & TRUNCATE Table



## MySQL DROP & Truncate

Student Table

Name	Age	Gender
Ram Kumar	19	Male
Salman Khan	22	Male
Meera Khan	21	Female
Sarita Kumari	18	Female
Anil Kapoor	22	Male

DROP Command

Student Table

Name	Age	Gender

TRUNCATE Command



## DROP & TRUNCATE Syntax

`DROP TABLE table_name;`

`TRUNCATE TABLE table_name;`

1 • `SELECT * FROM student.courses;`

course_id	course_name
1	Btech
2	BCA
3	MBA
4	BA
5	BCOM

students students courses

1 • `SELECT * FROM student.courses;`

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	course_id	course_name
1	1	Btech
2	2	BCA
3	3	MBA
4	4	BA
5	5	BCOM

File Server Tools Scripting Help

students students courses

1 • `TRUNCATE TABLE courses;`

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	course_id	course_name
*	NULL	NULL



use Server Tools Scripting Help

ents students courses

1 DROP TABLE courses;



Database Server Tools Scripting Help

students students courses

1 • SELECT \* FROM student.courses;

## 37. MySQL VIEW



## MySQL INNER JOIN Syntax

Notepad file

```
SELECT columns  
FROM student  
INNER JOIN city  
ON student.city = city.cid;
```

Save in Database → VIEW Command



## VIEW Syntax

```
CREATE VIEW view_name  
AS  
SELECT columns  
FROM student  
INNER JOIN city  
ON student.city = city.cid;
```

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

student

Tables

city

courses

students

Views

Stored Procedures

Functions

Administration Schemas Information

No object selected

students students courses

1 • SELECT \* FROM students;

Result Grid | Filter Rows: Limit to 1000 n | Edit | Export/Import | Wrap Cell Content: [x]

	id	name	percentage	dob	age	gender	city	courses
>	1	Ram Kumar	45	2000-05-10	19	M	1	1
>	2	Sarita Kumari	85	1997-02-03	22	F	2	2
>	3	Salman Khan	39	1999-11-12	20	M	1	1
>	4	Juhi Chawla	47	2001-07-16	18	F	3	1
>	5	Anil Kapoor	74	1997-01-03	22	M	1	3
>	6	John Abraham	64	1998-08-10	21	M	2	2
>	7	Shahid Kapoor	62	1999-12-08	20	M	1	3

students students courses

1 • SELECT \* FROM students;

Result Grid | Filter Rows: Limit to 1000 n | Edit | Export/Import | Wrap Cell Content: [x]

	id	name	percentage	dob	age	gender	city	courses
>	1	Ram Kumar	45	2000-05-10	19	M	1	1
>	2	Sarita Kumari	85	1997-02-03	22	F	2	2
>	3	Salman Khan	39	1999-11-12	20	M	1	1
>	4	Juhi Chawla	47	2001-07-16	18	F	3	1
>	5	Anil Kapoor	74	1997-01-03	22	M	1	3
>	6	John Abraham	64	1998-08-10	21	M	2	2
>	7	Shahid Kapoor	62	1999-12-08	20	M	1	3

Result Grid | Filter Rows: Limit to 1000 n | Edit | Export/Import | Wrap Cell Content: [x]

	id	name	percentage	dob	age	gender	city	courses
>	1	Ram Kumar	45	2000-05-10	19	M	1	1
>	2	Sarita Kumari	85	1997-02-03	22	F	2	2
>	3	Salman Khan	39	1999-11-12	20	M	1	1
>	4	Juhi Chawla	47	2001-07-16	18	F	3	1
>	5	Anil Kapoor	74	1997-01-03	22	M	1	3
>	6	John Abraham	64	1998-08-10	21	M	2	2
>	7	Shahid Kapoor	62	1999-12-08	20	M	1	3

```

1  SELECT id, name, course_name FROM
2  students s INNER JOIN courses c
3  ON s.courses = c.course_id;

```

Result Grid | Filter Rows:  Export: [CSV] | Wrap Cell Content: [X]

	id	name	course_name
1	1	Ram Kumar	Btech
2	2	Sarita Kumari	BCA
3	3	Salman Khan	Btech
4	4	Juhi Chawla	Btech
5	5	Anil Kapoor	MBA
6	6	John Abraham	BCA
7	7	Shahid Kapoor	MBA

we dont want to write

this query again and again so for that below query

```

1 • CREATE VIEW studentdata
2 AS
3 SELECT id, name, course_name FROM
4 students s INNER JOIN courses c
5 ON s.courses = c.course_id;

```

MySQL Workbench

File Edit View Query Database

**Navigator**

**SCHEMAS**

Filter objects

- phpmyadmin
- student**
  - Tables
    - city
    - courses
    - students
  - Views
    - studentdata**
  - Stored Procedures
  - Functions
- yb

**Result Grid** Filter Rows Export Wrap Cell Content

	<b>id</b>	<b>name</b>	<b>course_name</b>
1	1	Ram Kumar	Btech
2	2	Sarita Kumari	BCA
3	3	Salman Khan	Btech
4	4	Juhi Chawla	Btech
5	5	Anil Kapoor	MBA
6	6	John Abraham	BCA
7	7	Shahid Kapoor	MBA

If we want to change view command

**students** students courses city

Limit to 1000 n

```

1 • ALTER VIEW studentdata
2 AS
3 SELECT id, name, course_name, cityname FROM
4 students s INNER JOIN courses c
5 ON s.courses = c.course_id
6 INNER JOIN city ci
7 ON s.city = ci.cid;
8

```

**Result Grid** Filter Rows Export Wrap Cell Content

	<b>id</b>	<b>name</b>	<b>course_name</b>
1	1	Ram Kumar	Btech
3	3	Salman Khan	Btech
4	4	Juhi Chawla	Btech
2	2	Sarita Kumari	BCA
6	6	John Abraham	BCA
5	5	Anil Kapoor	MBA

students > students courses city

```

3   SELECT id, name, course_name, cityname FROM
4     students s INNER JOIN courses c
5       ON s.courses = c.course_id
6   INNER JOIN city ci
7     ON s.city = ci.cid;
8
9 • SELECT * FROM studentdata;

```

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Contents: [ ]

	<b>id</b>	<b>name</b>	<b>course_name</b>	<b>cityname</b>
>	1	Ram Kumar	Btech	Agra
	3	Salman Khan	Btech	Agra
	4	Juhi Chawla	Btech	Bhopal
	2	Sarita Kumari	BCA	Delhi
	6	John Abraham	BCA	Delhi
	5	Anil Kapoor	MBA	Agra

students > students courses city

```

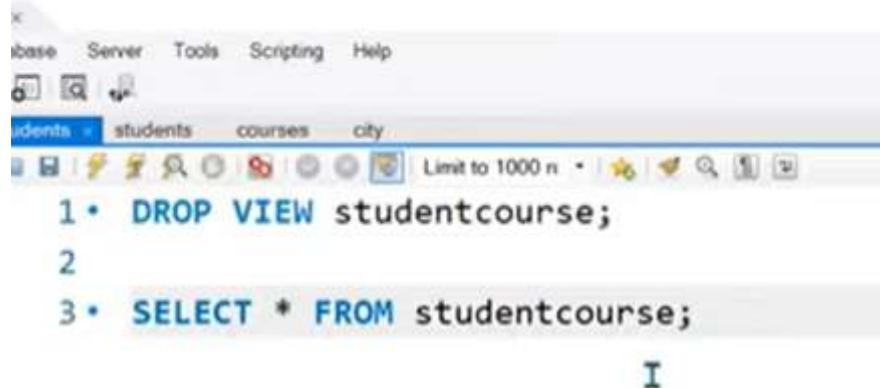
1 • RENAME TABLE studentdata
2   TO studentcourse;
3
4 • SELECT * FROM studentcourse;

```

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Contents: [ ]

	<b>id</b>	<b>name</b>	<b>course_name</b>	<b>cityname</b>
>	1	Ram Kumar	Btech	Agra
	3	Salman Khan	Btech	Agra
	4	Juhi Chawla	Btech	Bhopal
	2	Sarita Kumari	BCA	Delhi
	6	John Abraham	BCA	Delhi
	5	Anil Kapoor	MBA	Agra
	7	Shahid Kapoor	MBA	Agra

dropping view



```
1 • DROP VIEW studentcourse;
2
3 • SELECT * FROM studentcourse;
```

command



## Advantages & Disadvantages of VIEW :

### Advantages :

- Simplify complex query.
- Provides Extra layer of Security.

### Disadvantages :

- Performance Decreases.
- Dependency on Table.

## 38. MySQL INDEX



## What is Index ?

INDEX PAGE

Ch. No.	Chapter	Page No.
1	ABC	3
2	XYZ	15
3	EFG	23
4	IJK	29
5	RST	37

Student Table

Name	Age	Gender
Ram Kumar	19	Male
Salman Khan	22	Male
Meera Khan	21	Female
Sarita Kumari	18	Female
Anil Kapoor	22	Male



## INDEX Syntax

```
CREATE INDEX index_name  
ON table_name(column1, column2, column3, .... );
```

```
DROP INDEX index_name  
ON table_name;
```



## Guidelines of Index :

- Automatically creates the indexes for PRIMARY KEY and UNIQUE columns.
- Index columns that you frequently use to retrieve the data.
- Index columns that are used for joins to improve join performance.
- Avoid columns that contain too many NULL values.
- Small tables do not require indexes.

SQL File 8 students

1 • **SELECT \* FROM students;**

	<b>id</b>	<b>name</b>	<b>percentage</b>	<b>dob</b>	<b>age</b>	<b>gender</b>	<b>city</b>	<b>courses</b>
▶	1	Ram Kumar	45	2000-05-10	19	M	1	1
	2	Sarita Kumari	85	1997-02-03	22	F	2	2
	3	Salman Khan	39	1999-11-12	20	M	1	1
	4	Juhi Chawla	47	2001-07-16	18	F	3	1
	5	Anil Kapoor	74	1997-01-03	22	M	1	3
	6	John Abraham	64	1998-08-10	21	M	2	2
	7	Shahid Kapoor	62	1999-12-08	20	M	1	3

SQL File 8\* students

1   **SELECT \* FROM students**

2   **WHERE dob > "1999-01-01";** | I

3

4 • **CREATE INDEX studdob ON students (dob);**

	<b>id</b>	<b>name</b>	<b>percentage</b>	<b>dob</b>	<b>age</b>	<b>gender</b>	<b>city</b>	<b>courses</b>
▶	1	Ram Kumar	45	2000-05-10	19	M	1	1
	3	Salman Khan	39	1999-11-12	20	M	1	1
	4	Juhi Chawla	47	2001-07-16	18	F	3	1
	7	Shahid Kapoor	62	1999-12-08	20	M	1	3

SQL File 8\* students

```

1  SELECT * FROM students
2  WHERE dob > "1999-01-01";
3
4+ CREATE INDEX studdob ON students (dob);
5
6+ SHOW INDEX FROM students;

```

Result Grid | Filter Rows: Export | Map Cell Content: □

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment
students	0	id	1	id	A	4				BTREE		
students	1	studdob	1	dob	A	4				BTREE		

Navigator

**SCHEMAS**

Filter objects:

- phpmyadmin
- student**
  - Tables
    - city
    - courses
    - students** (highlighted)
  - Views
  - Stored Procedures
  - Functions
- yb

SQL File 8\* student.students

```

1  S
2  W
3
4+ C
5
6+ S

```

Navigator

**SCHEMAS**

Filter objects:

- phpmyadmin
- student**
  - Tables
    - city
    - courses
    - students**
  - Views
  - Stored Procedures
  - Functions
- yb

SQL File 8\* student.students

Info Columns Indexes Triggers Foreign keys Partitions Grants DDL

**Indexes in Table**

Key	Type	Unique	Columns
id	BTREE	YES	id
studdob	BTREE	NO	dob

**Index Details**

Key Name:  
Index Type:  
Allows NULL:  
Cardinality:  
Comment:  
User Comment:

**Columns in table**

Column	Type	Null...	Indexes
id	int(11)	NO	id
name	varchar(50)	NO	
percentage	int(11)	NO	
dob	date	NO	studdob
age	int(11)	NO	
gender	varchar(1)	NO	
city	int(11)	NO	
courses	int(11)	NO	

Administration Schemas Information

Schema: **student**

5

6 • **DROP INDEX studdob ON students;**

The screenshot shows the phpMyAdmin interface. On the left, the Navigator pane displays the 'SCHEMAS' section with 'student' selected. Under 'Tables', 'id' is highlighted. The main area shows the 'student.students' table with two tabs: 'Indexes' (selected) and 'Columns'. The 'Indexes' tab lists one index: 'id' (Type: BTREE, Unique: YES, Column: id). The 'Columns' tab lists seven columns: id, name, percentage, dob, age, gender, city, and courses. The 'Index Details' panel on the right is empty.

In [ ]: