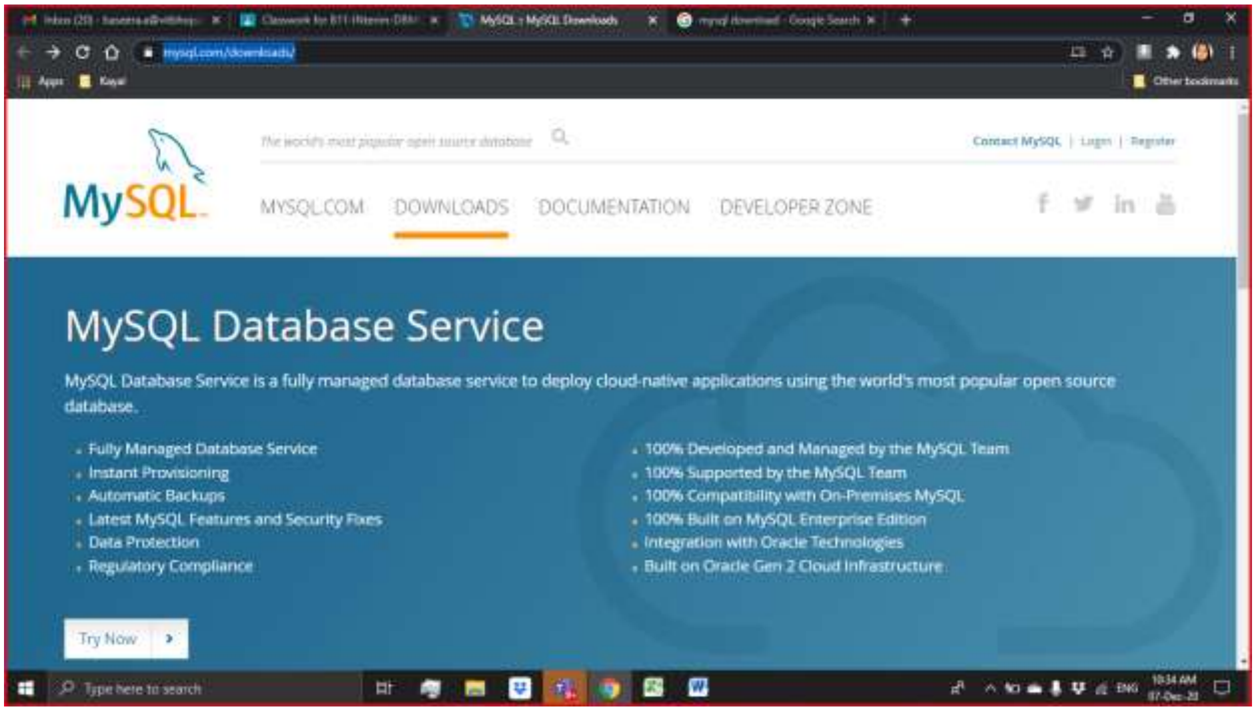
Make sure that your system is not installed mysql already !!!!!!!!!

Also make sure that your system or your default drive is not installed xamp already !!!

If it is installed you have to uninstall or install my sql in a separate drive !!!!

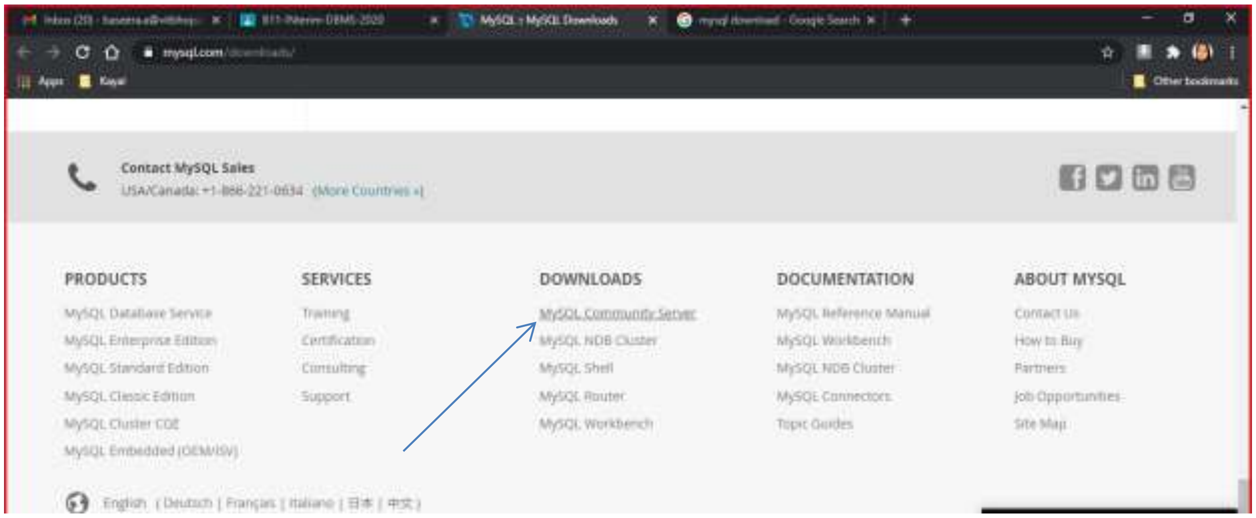
Steps to installation process of mysql – (open source)

1. Open any browser…. What ever you have…..
2. Go to official website -> “mysql.com” - select →downloads – tab - ---


MySQL Database Service

MySQL Database Service is a fully managed database service to deploy cloud-native applications using the world's most popular open source database.

- Fully Managed Database Service
- Instant Provisioning
- Automatic Backups
- Latest MySQL Features and Security Fixes
- Data Protection
- Regulatory Compliance
- 100% Developed and Managed by the MySQL Team
- 100% Supported by the MySQL Team
- 100% Compatibility with On-Premises MySQL
- 100% Built on MySQL Enterprise Edition
- Integration with Oracle Technologies
- Built on Oracle Gen 2 Cloud Infrastructure

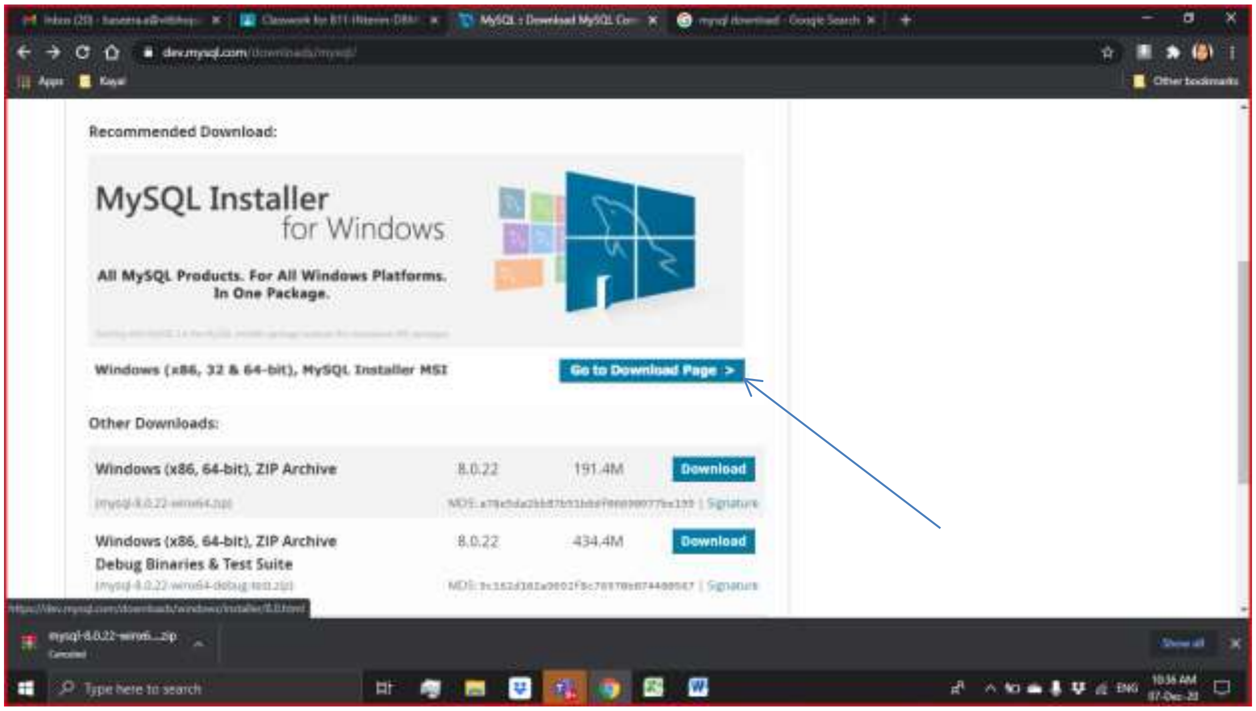


Contact MySQL Sales
USA/Canada: +1-866-221-0634 (More Countries »)

PRODUCTS	SERVICES	DOWNLOADS	DOCUMENTATION	ABOUT MYSQL
MySQL Database Service	Training	MySQL Community Server	MySQL Reference Manual	Contact Us
MySQL Enterprise Edition	Certification	MySQL NDB Cluster	MySQL Workbench	How to Buy
MySQL Standard Edition	Consulting	MySQL Shell	MySQL NDB Cluster	Partners
MySQL Classic Edition	Support	MySQL Router	MySQL Connectors	Job Opportunities
MySQL Cluster CGE		MySQL Workbench	Topic Guides	Site Map
MySQL Embedded (OEM/ISV)				

English (Deutsch | Français | Italiano | 日本 | 中文)

Select goto download page

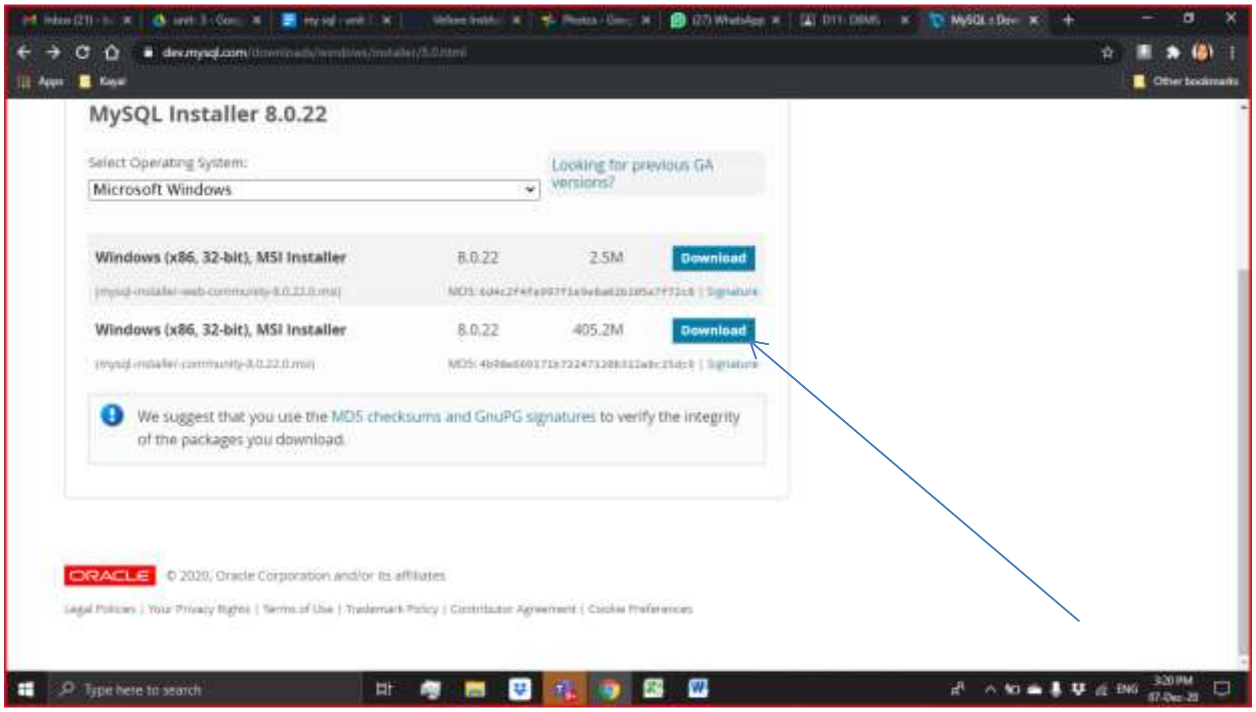


Recommended Download:

Windows (x86, 32 & 64-bit), MySQL Installer MSI — Go to Download Page >

Other Downloads:

Windows (x86, 64-bit), ZIP Archive (mysql-8.0.22-winx64.zip)	8.0.22	191.4M	Download
Windows (x86, 64-bit), ZIP Archive Debug Binaries & Test Suite (mysql-8.0.22-winx64-debug-test.zip)	8.0.22	434.4M	Download


MySQL Installer 8.0.22

Select Operating System:
Microsoft Windows

Looking for previous GA versions?

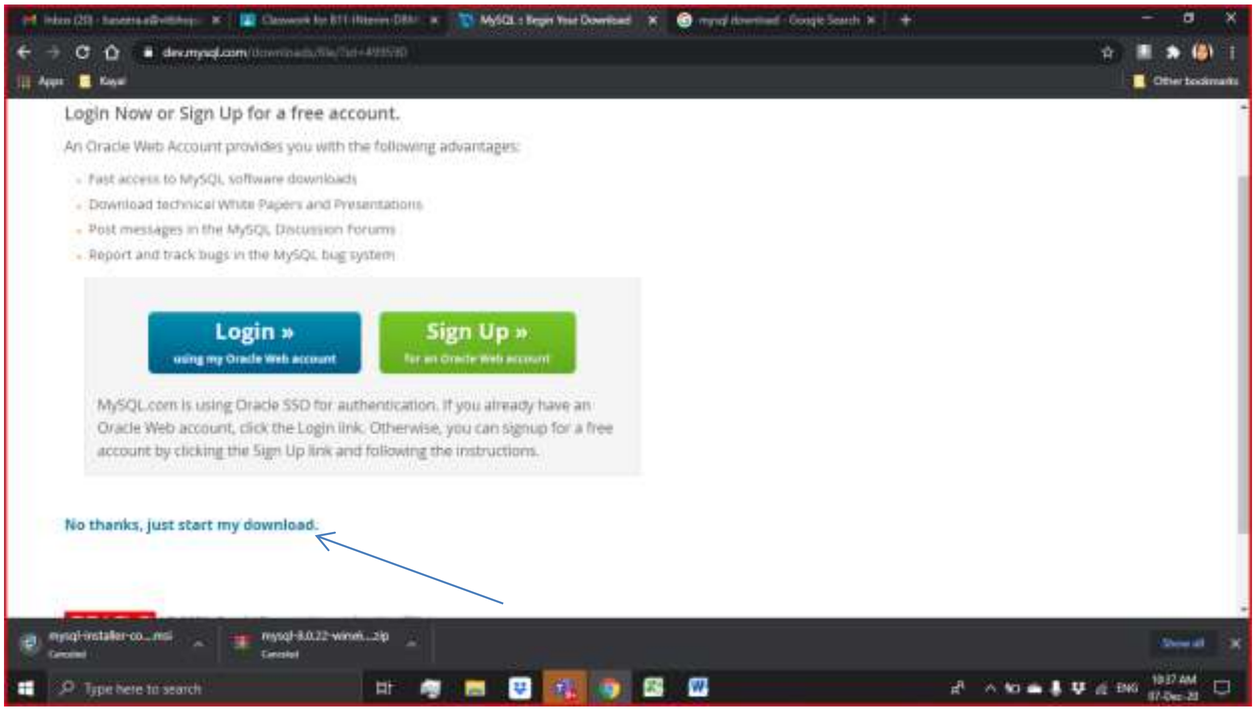
Windows (x86, 32-bit), MSI Installer	8.0.22	2.5M	Download
(mysql-installer-web-community-8.0.22.0.msi)	MD5: 6d4c2f4fe997f1a9ebad2b105a7f72c6 | Signature		
Windows (x86, 32-bit), MSI Installer	8.0.22	405.2M	Download
(mysql-installer-community-8.0.22.0.msi)	MD5: 4b98e660371b72247128b332a8c25dc8 | Signature		

We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

© 2020, Oracle Corporation and/or its affiliates

Legal Policies | Your Privacy Rights | Terms of Use | Trademark Policy | Contributor Agreement | Cookie Preferences

Select 2nd option


Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

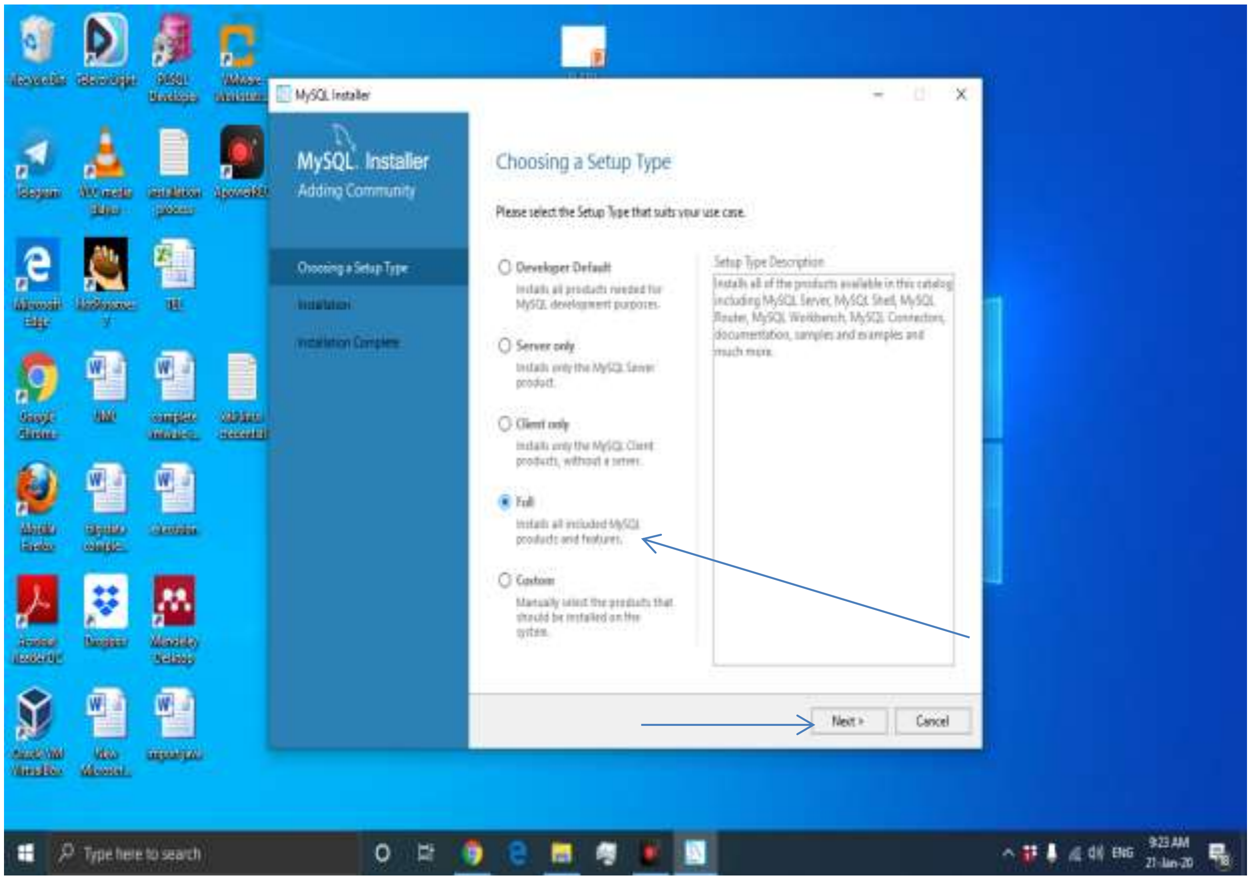
- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system

Login » using my Oracle Web account
Sign Up » for an Oracle Web account

MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

No thanks, just start my download.

Click - >   No thanks just start my download



MySQL Installer

MySQL. Installer
Adding Community

- Choosing a Setup Type
- Installation
- Installation Complete

Choosing a Setup Type

Please select the Setup Type that suits your use case.

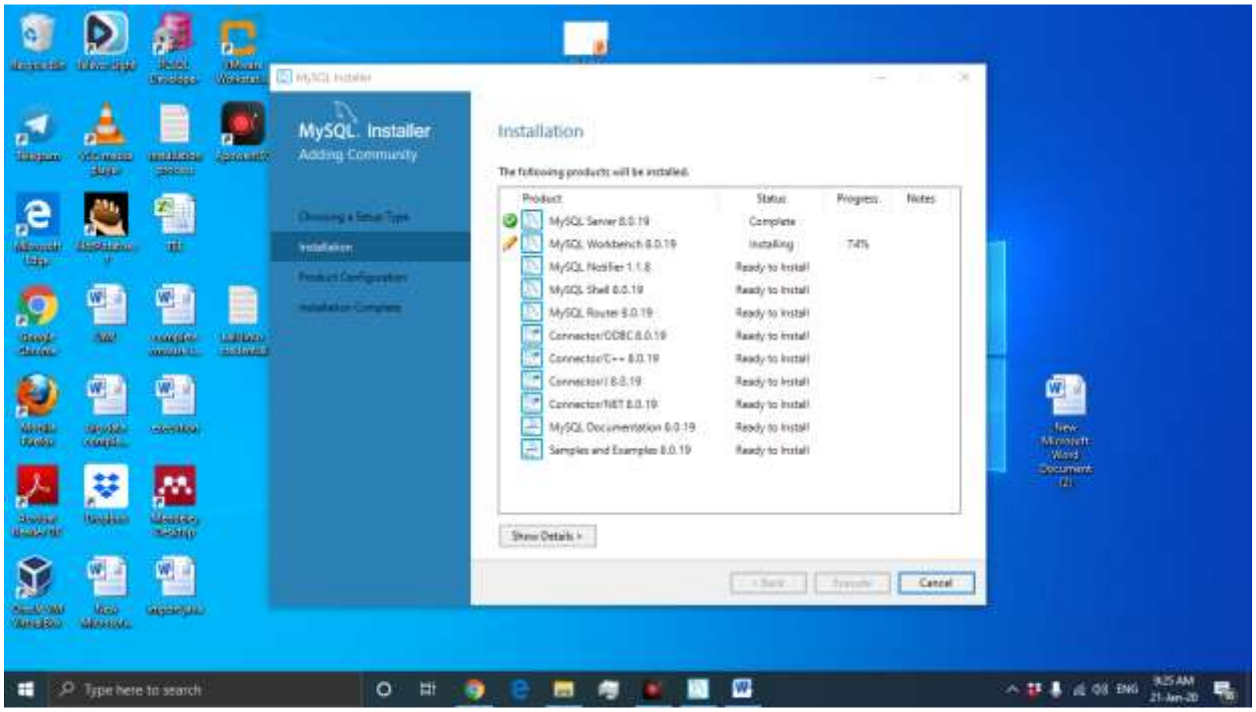
- ○ Developer Default
  Installs all products needed for MySQL development purposes.
- ○ Server only
  Installs only the MySQL Server product.
- ○ Client only
  Installs only the MySQL Client products, without a server.
- [✓ Full]
  Installs all included MySQL products and features.
- ○ Custom
  Manually select the products that should be installed on the system.

Setup Type Description

Installs all of the products available in this catalog including MySQL Server, MySQL Shell, MySQL Router, MySQL Workbench, MySQL Connectors, documentation, samples and examples and much more.

Yes

Continue

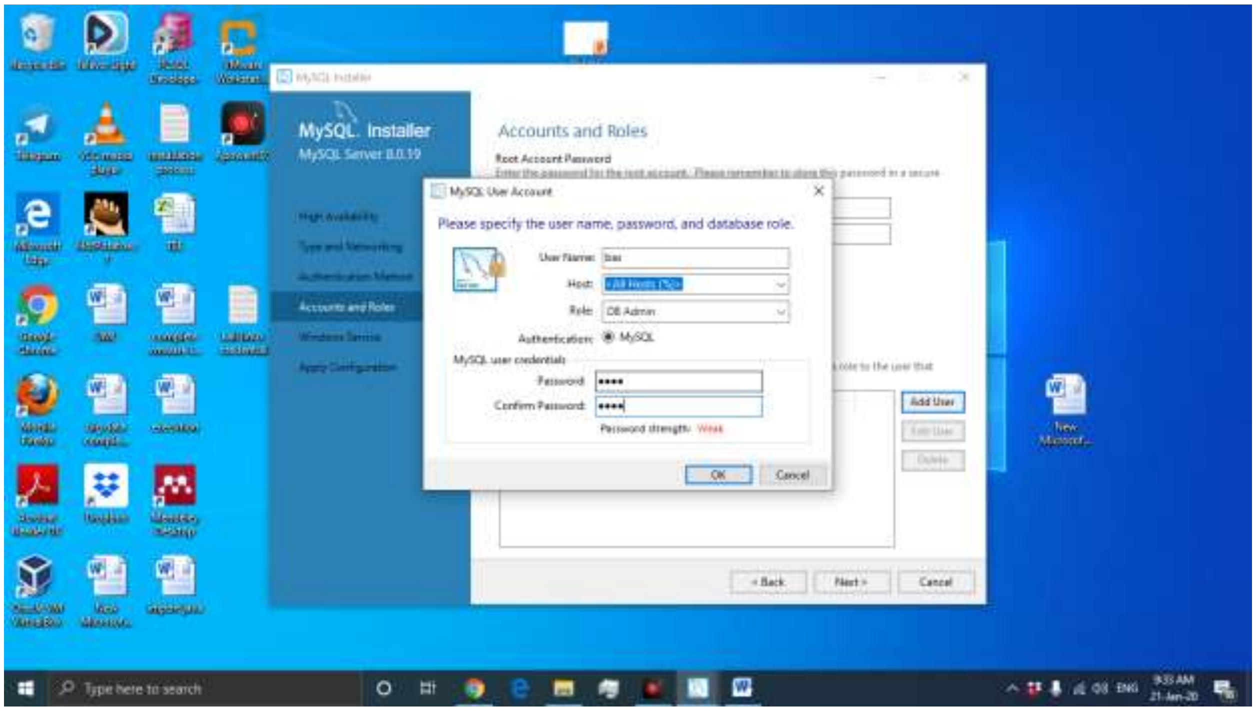

MySQL Installer
Adding Community

Choosing a Setup Type
Installation
Product Configuration
Installation Complete

Installation

The following products will be installed.

Product	Status	Progress	Notes
MySQL Server 8.0.19	Complete		
MySQL Workbench 8.0.19	Installing	74%	
MySQL Notifier 1.1.8	Ready to Install		
MySQL Shell 8.0.19	Ready to Install		
MySQL Router 8.0.19	Ready to Install		
Connector/ODBC 8.0.19	Ready to Install		
Connector/C++ 8.0.19	Ready to Install		
Connector/J 8.0.19	Ready to Install		
Connector/NET 8.0.19	Ready to Install		
MySQL Documentation 8.0.19	Ready to Install		
Samples and Examples 8.0.19	Ready to Install		


MySQL Installer
MySQL Server 8.0.19

High Availability
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Apply Configuration

Accounts and Roles

Root Account Password
Enter the password for the root account. Please remember to store this password in a secure

MySQL User Account

Please specify the user name, password, and database role.

User Name: bas
Host: <All Hosts (%)>
Role: DB Admin
Authentication: ● MySQL

MySQL user credentials
Password: ••••
Confirm Password: ••••
Password strength: Weak

MySQL is an open source database product that was created by MySQL AB, a company founded in 1995 in Sweden.

MySQL was started in 1979, when MySQL's inventor, Michael Widenius developed an in-house database tool called UNIREG for managing databases. After that UNIREG has been rewritten in several different languages and extended to handle big databases. After some time Michael Widenius contacted David Hughes, the author of mSQL, to see if Hughes would be interested in connecting mSQL to UNIREG's B+ ISAM handler to provide indexing to mSQL. That's the way MySQL came in existence.

History by year:

Year	Happenings
1995	MySQL AB founded by Michael Widenius (Monty), David Axmark and Allan Larsson in Sweden.
2000	MySQL goes open source and releases software under the terms of the GPL. Revenues dropped 80% as a result, and it took a year to make up for it.
2001	Marten Mickos elected CEO at age 38. Marten was the CEO of a number of nordic companies before joining MySQL, and comes with a sales and marketing background. 2 million active installations. Raised series a with undisclosed amount from Scandinavian venture capitalists. Estimated to be around $1 to $2 million.
2002	MySQL launched us headquarters in addition to Swedish headquarters. 3 million active users. Ended the year with $6.5 million in revenue with 1,000 paying customers.
2003	Raised a $19.5 million series b from benchmark capital and index ventures. 4 million active installations and over 30,000 downloads per day. Ended the year with $12 million in revenue.
2004	With the main revenue coming from the oem dual-licensing model, MySQL decides to move more into the enterprise market and to focus more on recurring revenue from end users rather than one-time licensing fees from their oem partners. Ended the year with $20 million in revenue.
2005	MySQL launched the MySQL network modeled after the Redhat network. the MySQL network is a subscription service targeted at end users that provides updates, alerts, notifications, and product-level support designed to make it easier for companies to manage hundreds of MySQL servers. MySQL 5 ships and includes many new features to go after enterprise users (e.g. stored procedures, triggers, views, cursors, distributed transactions, federated storage engines, etc.)

	Oracle buys innobase, the 4-person and a Finland's company behind MySQL's innodb storage backend, ended the year with $34 million in revenue based on 3400 customers.
2006	Marten Mickos confirms that oracle tried to buy MySQL. Oracle' CEO Larry Ellison commented: "we've spoken to them, in fact we've spoken to almost everyone. Are we interested? It's a tiny company. I think the revenues from MySQL are between $30 million and $40 million. Oracle's revenue next year is $15 billion." Oracle buys sleepycat, the company that provides MySQL with the Berkeley db transactional storage engine. Marten Mickos announces that they are making MySQL ready for an IPO in 2008 on an projected $100 million in revenues. 8 million active installations. MySQL has 320 employees in 25 countries, 70 percent of whom work from home, raised a $18 million series c based on a rumored valuation north of $300 million. MySQL is estimated to have a 33% market share measured in install base and 0.2% market share measured in revenue (the database market was a $15 billion market in 2006). Ended the year with $50 million in revenue.
2007	Ended the year with $75 million in revenue.
2008	Sun Microsystems acquired MySQL AB for approximately $1 billion. Michael Widenius (Monty) and David Axmark, two of MySQL AB's co-founders, begin to criticize Sun publicly and leave Sun shortly after.
2009	Marten Mickos leaves Sun and becomes entrepreneur-in-residence at Benchmark Capital. Sun has now lost the business and spiritual leaders that turned MySQL into a success. Sun Microsystems and Oracle announced that they have entered into a definitive agreement under which Oracle will acquire Sun common stock for $9.50 per share in cash. The transaction is valued at approximately $7.4 billion.

MySQL Features

- **Relational Database Management System (RDBMS):** MySQL is a relational database management system.
- **Easy to use:** MySQL is easy to use. You have to get only the basic knowledge of SQL. You can build and interact with MySQL with only a few simple SQL statements.
- **It is secure:** MySQL consist of a solid data security layer that protects sensitive data from intruders. Passwords are encrypted in MySQL.

- **Client/ Server Architecture:** MySQL follows a client /server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc.
- **Free to download:** MySQL is free to use and you can download it from MySQL official website.
- **It is scalable:** MySQL can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, you can increase this number to a theoretical limit of 8 TB of data.
- **Compatibale on many operating systems:** MySQL is compatible to run on many operating systems, like Novell NetWare, Windows* Linux*, many varieties of UNIX* (such as Sun* Solaris*, AIX, and DEC* UNIX), OS/2, FreeBSD*, and others. MySQL also provides a facility that the clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).
- **Allows roll-back:** MySQL allows transactions to be rolled back, commit and crash recovery.
- **High Performance:** MySQL is faster, more reliable and cheaper because of its unique storage engine architecture.
- **High Flexibility:** MySQL supports a large number of embedded applications which makes MySQL very flexible.
- **High Productivity:** MySQL uses Triggers, Stored procedures and views which allows the developer to give a higher productivity.

Disadvantages / Drawback of MySQL:

Following are the few disadvantages of MySQL:

- MySQL version less than 5.0 doesn't support ROLE, COMMIT and stored procedure.
- MySQL does not support a very large database size as efficiently.
- MySQL doesn't handle transactions very efficiently and it is prone to data corruption.
- MySQL is accused that it doesn't have a good developing and debugging tool compared to paid databases.
- MySQL doesn't support SQL check constraints.

Maximum size of 255 bytes.	
BLOB(size)	Maximum size of 65,535 bytes.

MEDIUMBLOB	Maximum size of 16,777,215 bytes.
LONGTEXT	Maximum size of 4gb or 4,294,967,295 characters.

Download MySQL

Follow these steps:

- Go to MySQL official website **http://www.mysql.com/downloads/**
- Choose the version number for MySQL community server which you want.

installing MySQL on Windows

Your downloaded MySQL is neatly packaged with an installer. Download the installer package, unzip it anywhere and run setup.exe.

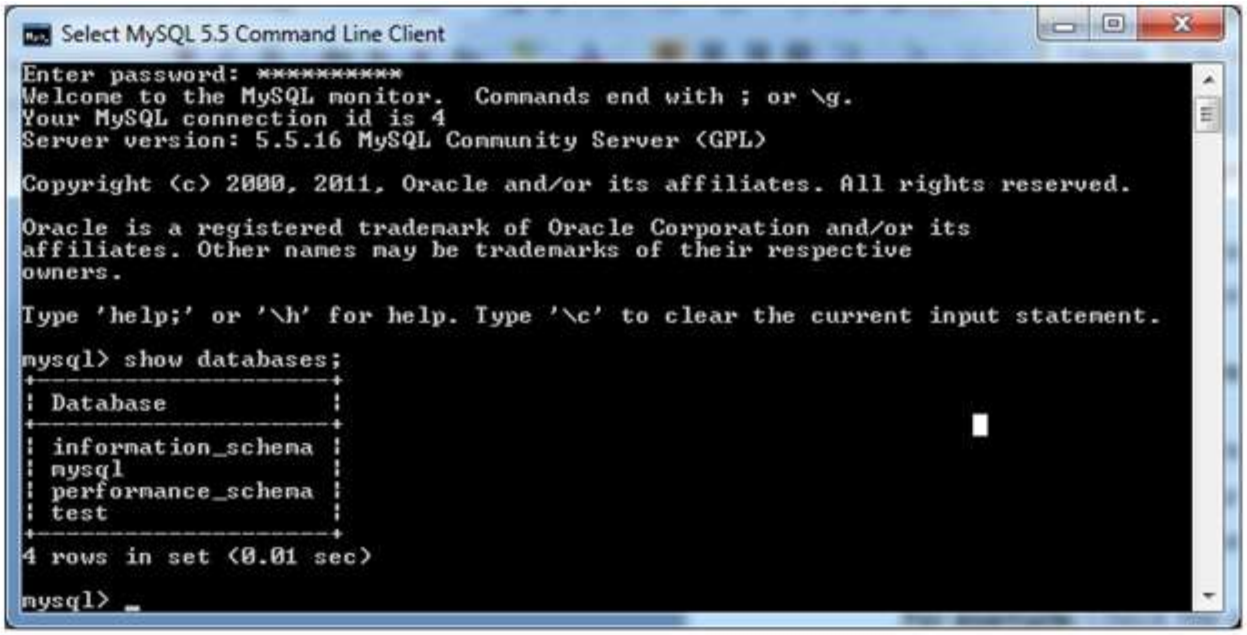
By default, this process will install everything under C:\mysql.

Verify MySQL installation

Once MySQL has been successfully installed, the base tables have been initialized, and the server has been started, you can verify its working via some simple tests.

Open your MySQL Command Line Client, it should be appeared with a mysql> prompt. If you have set any password, write your password here. Now, you are connected to the MySQL server and you can execute all the SQL command at mysql> prompt as follows:

For example: Check the already created databases with show databases command:


Select MySQL 5.5 Command Line Client

```
Enter password: **********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.5.16 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.01 sec)

mysql> _
```

What is Primary key

A primary key is a single field or combination of fields that contains a unique record. It must be filled. None of the field of primary key can contain a null value. A table can have only one primary key.

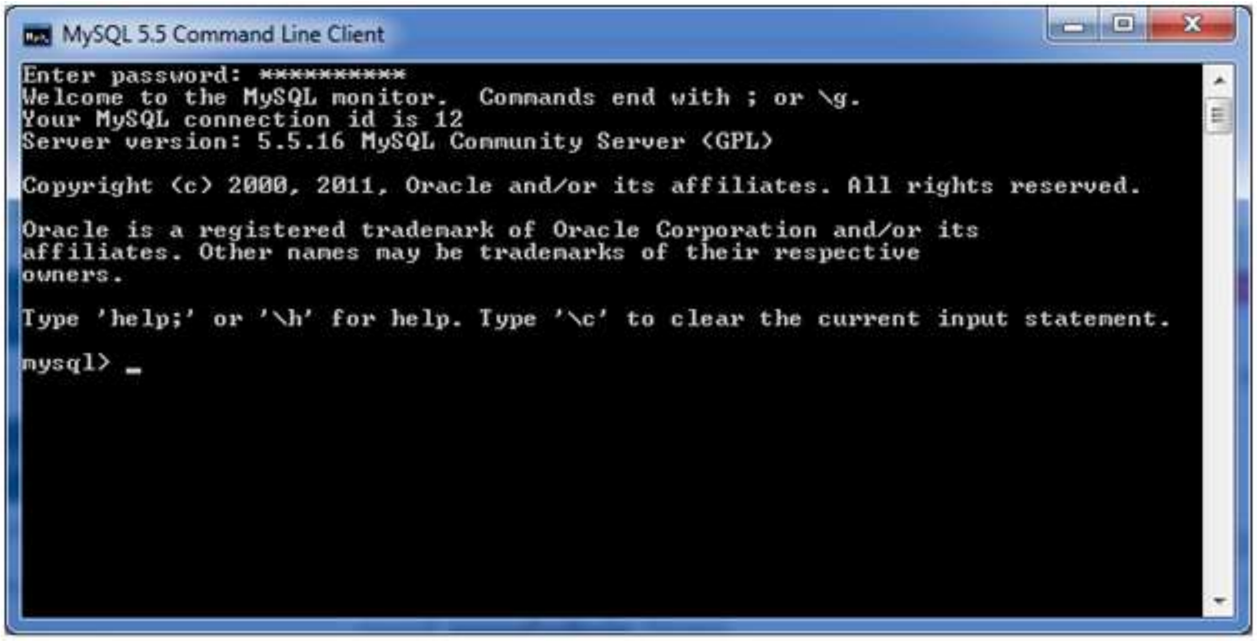
MySQL Create Database

- A database is a collection of data.
- MySQL allows us to store and retrieve the data from the database in a efficient way.
- In MySQL, we can create a database using the **CREATE DATABASE** statement.
- But, if database already exits, it throws an error.
- To avoid the error, we can use the **IF NOT EXISTS** option with the CREATE DATABASE statement.

You can create a MySQL database by using MySQL Command Line Client.

Open the MySQL console and write down password,

if you set one while installation. You will get the following:



```
Enter password: **********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.5.16 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> _
```

Now you are ready to create database.

Syntax:

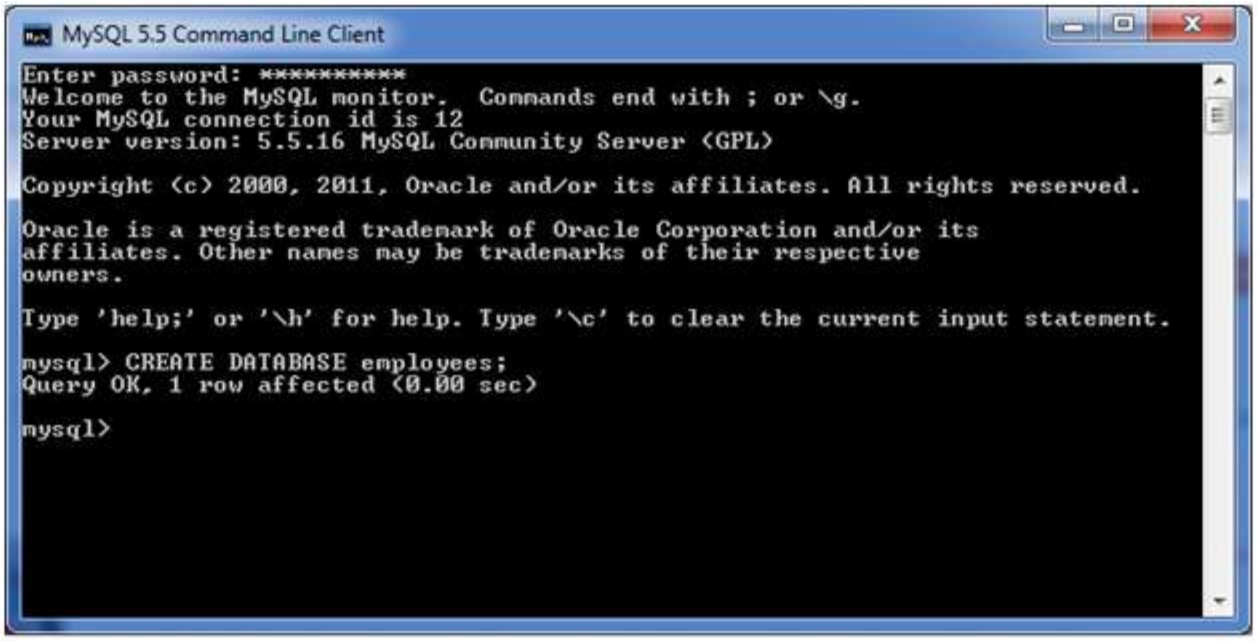
1. **CREATE DATABASE** database_name;

Example:

Let's take an example to create a database name "employees"

1. **CREATE DATABASE** employees;

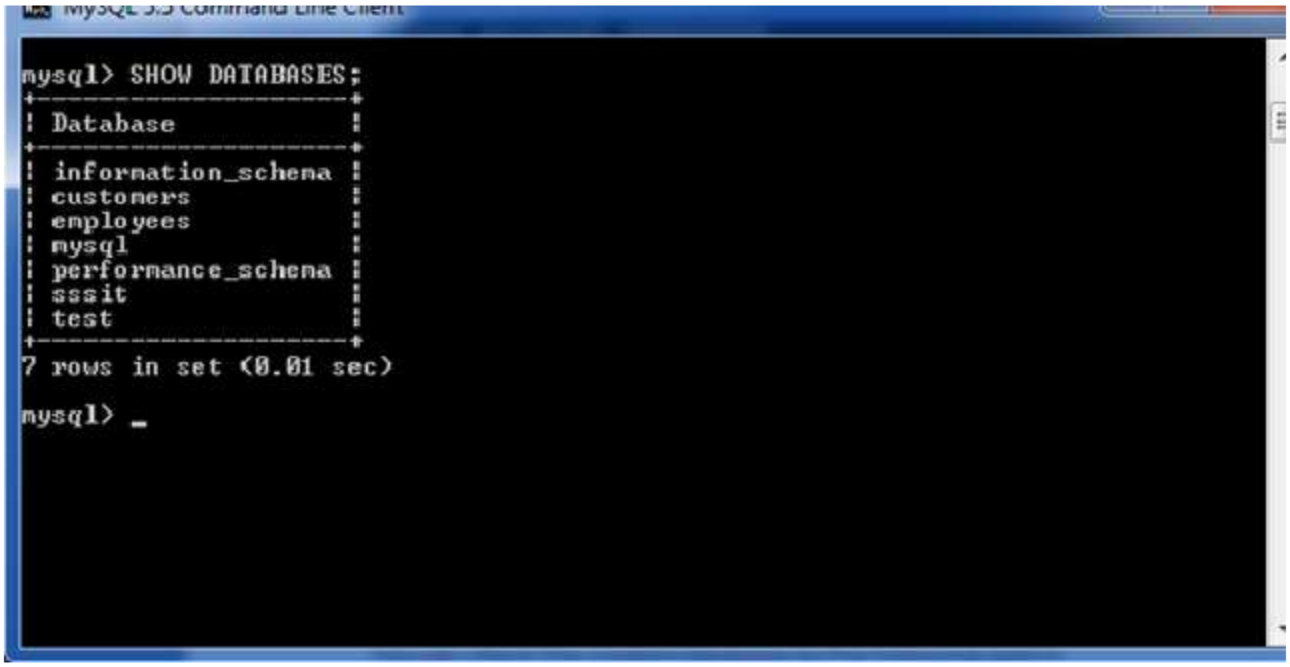
It will look like this:


```
Enter password: **********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.5.16 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE employees;
Query OK, 1 row affected (0.00 sec)

mysql>
```

You can check the created database by the following query:

1. SHOW DATABASES;

Output


```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| customers          |
| employees          |
| mysql              |
| performance_schema |
| sssit              |
| test               |
+--------------------+
7 rows in set (0.01 sec)

mysql> _
```

Here, you can see the all created databases.

MySQL SELECT Database

SELECT Database is used in MySQL to select a particular database to work with. This query is used when multiple databases are available with MySQL Server.

You can use SQL command **USE** to select a particular database.

Syntax:

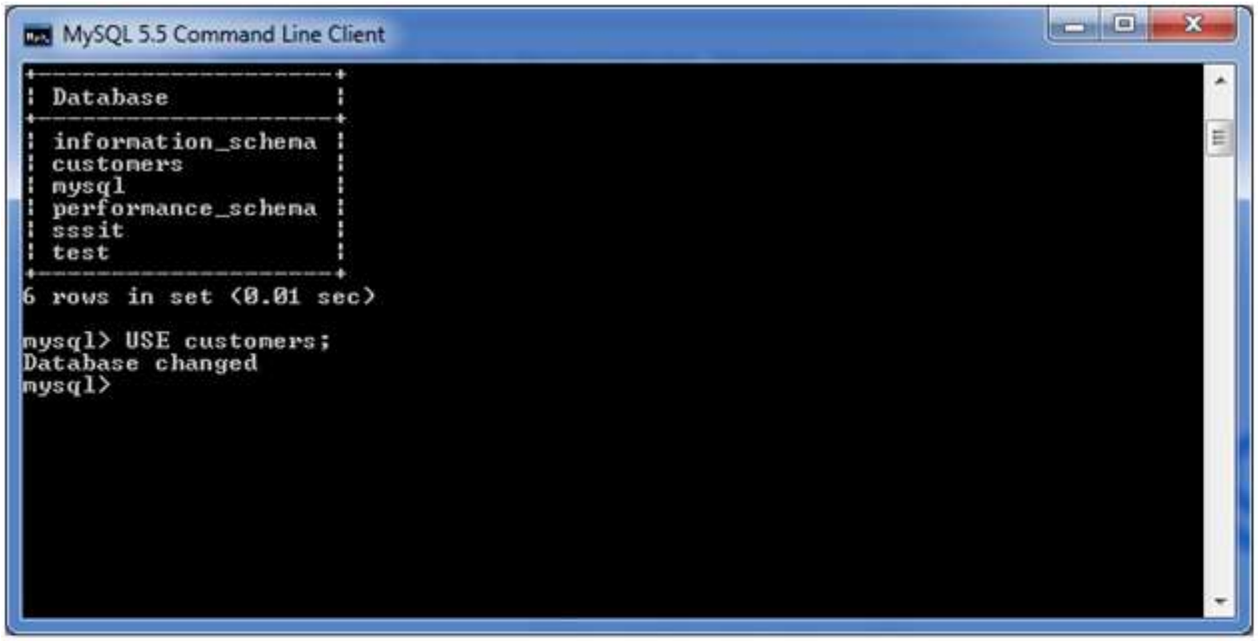
USE database_name;

Example:

Let's take an example to use a database name "customers".

1. USE customers;

It will look like this:



```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| customers          |
| mysql              |
| performance_schema |
| sssit              |
| test               |
+--------------------+
6 rows in set (0.01 sec)

mysql> USE customers;
Database changed
mysql>
```

Note: All the database names, table names and table fields name are case sensitive. You must have to use proper names while giving any SQL command.

MySQL Drop Database

- You can drop/delete/remove a MySQL database easily with the MySQL **DROP DATABASE** command.
- It deletes all the tables of the database along with the database permanently.

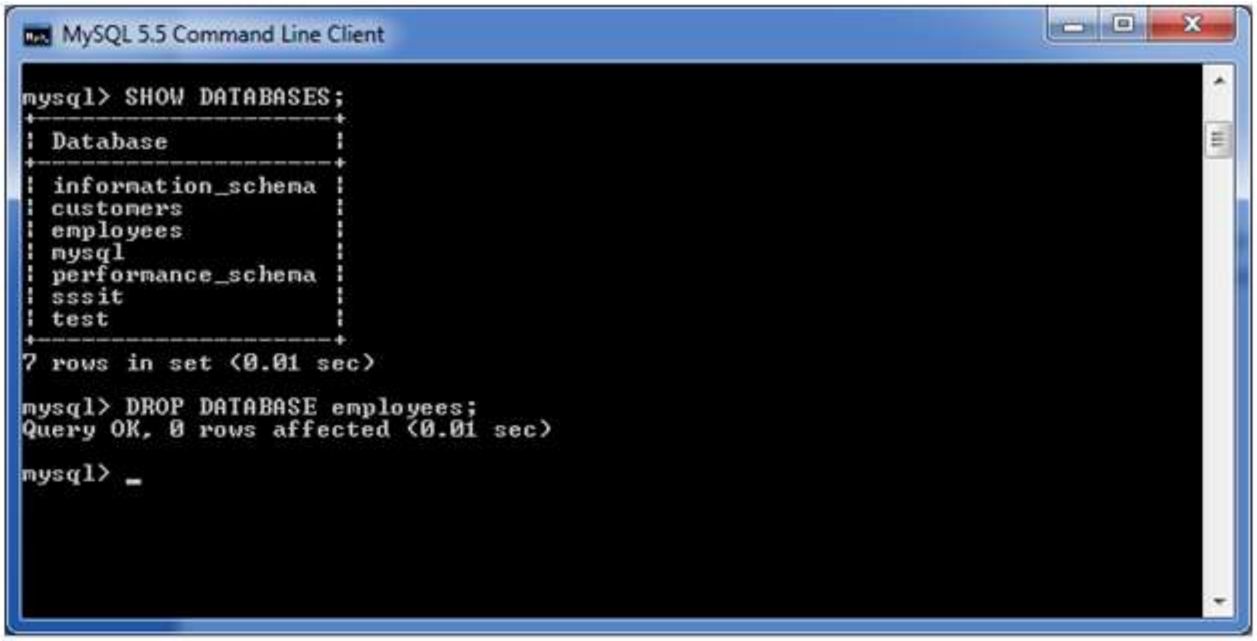
It throws an error, if the database is not available. We can use the **IF EXISTS** option with the DROP DATABASE statement. It returns the numbers of tables which are deleted through the

DROP DATABASE statement. We should be careful while deleting any database because we will loose all the data available in the database.

Syntax:

1. **DROP DATABASE** database_name;

Example:

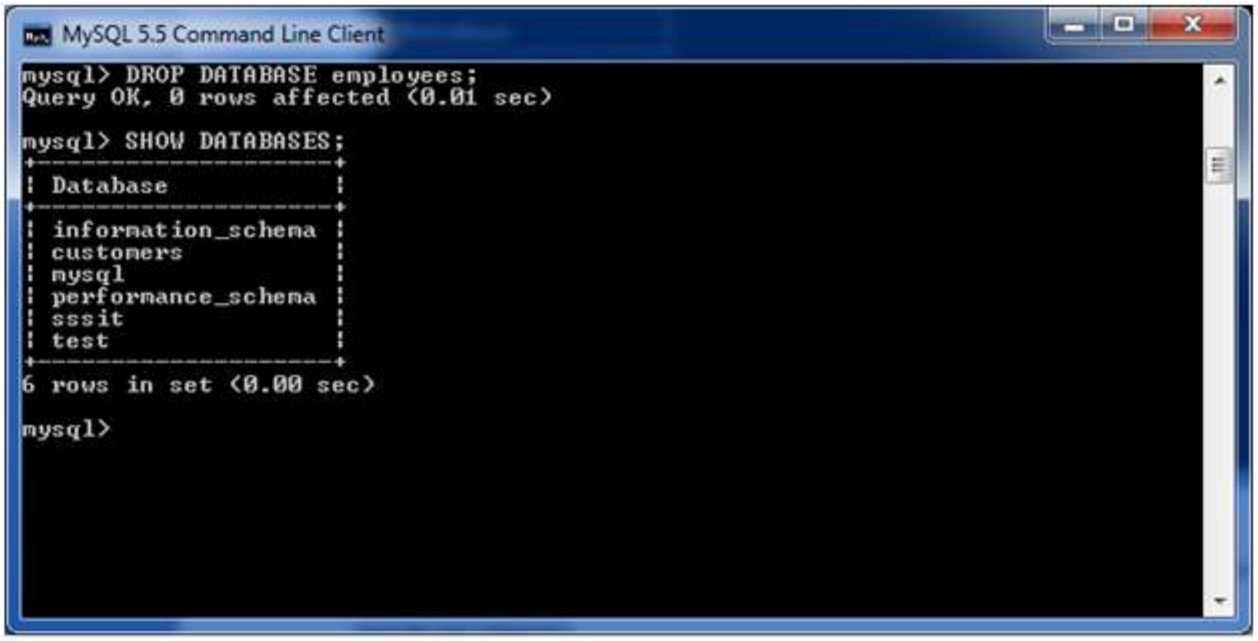


```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| customers          |
| employees          |
| mysql              |
| performance_schema |
| sssit              |
| test               |
+--------------------+
7 rows in set (0.01 sec)

mysql> DROP DATABASE employees;
Query OK, 0 rows affected (0.01 sec)

mysql> _
```

Now you can check that either your database is removed by executing the following query:

1. SHOW DATABASES;

Output:



```
mysql> DROP DATABASE employees;
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| customers          |
| mysql              |
| performance_schema |
| sssit              |
| test               |
+--------------------+
6 rows in set (0.00 sec)

mysql>
```

Here, you can see that the database "employees" is removed.

MySQL CREATE TABLE

The MySQL CREATE TABLE command is used to create a new table into the database. A table creation command requires three things:

- Name of the table
- Names of fields
- Definitions for each field

Syntax:

Following is a generic syntax for creating a MySQL table in the database.

```
CREATE TABLE table_name (column_name column_type...);
```

Example:

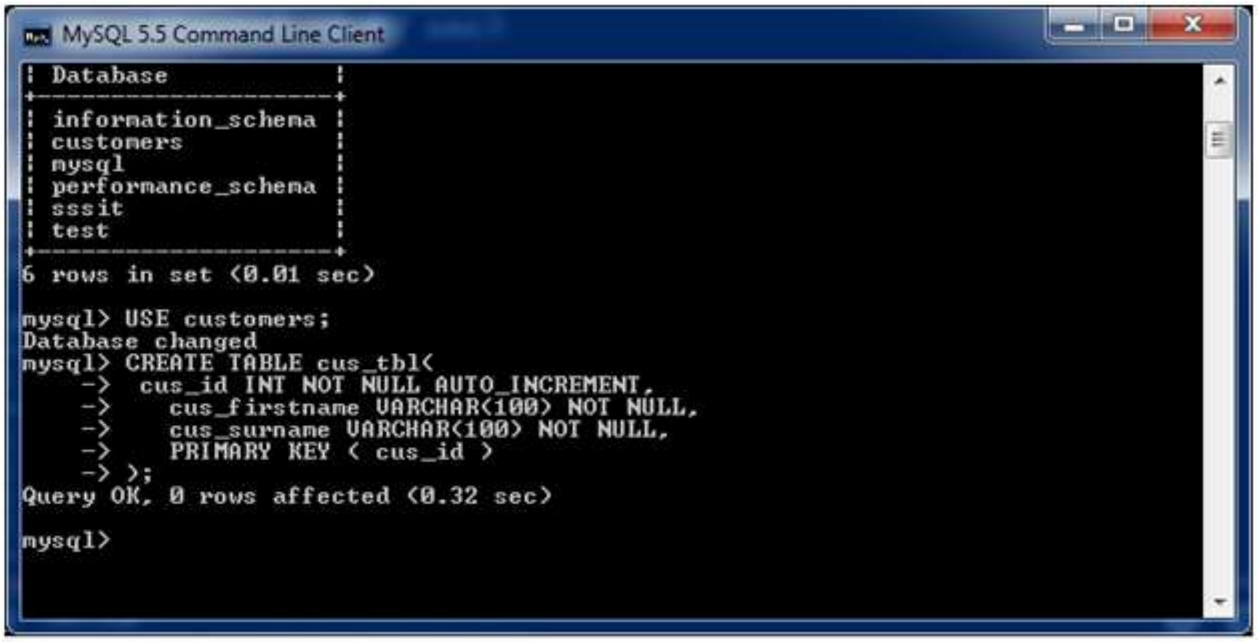
Here, we will create a table named "cus_tbl" in the database "customers".

```
CREATE TABLE cus_tbl(
  cus_id INT NOT NULL AUTO_INCREMENT,
  cus_firstname VARCHAR(100) NOT NULL,
  cus_surname VARCHAR(100) NOT NULL,
  PRIMARY KEY ( cus_id )
);
```

Note:

1. Here, NOT NULL is a field attribute and it is used because we don't want this field to be NULL. If you will try to create a record with NULL value, then MySQL will raise an error.
2. The field attribute AUTO_INCREMENT specifies MySQL to go ahead and add the next available number to the id field.PRIMARY KEY is used to define a column as primary key. You can use multiple columns separated by comma to define a primary key.

Visual representation of creating a MySQL table:

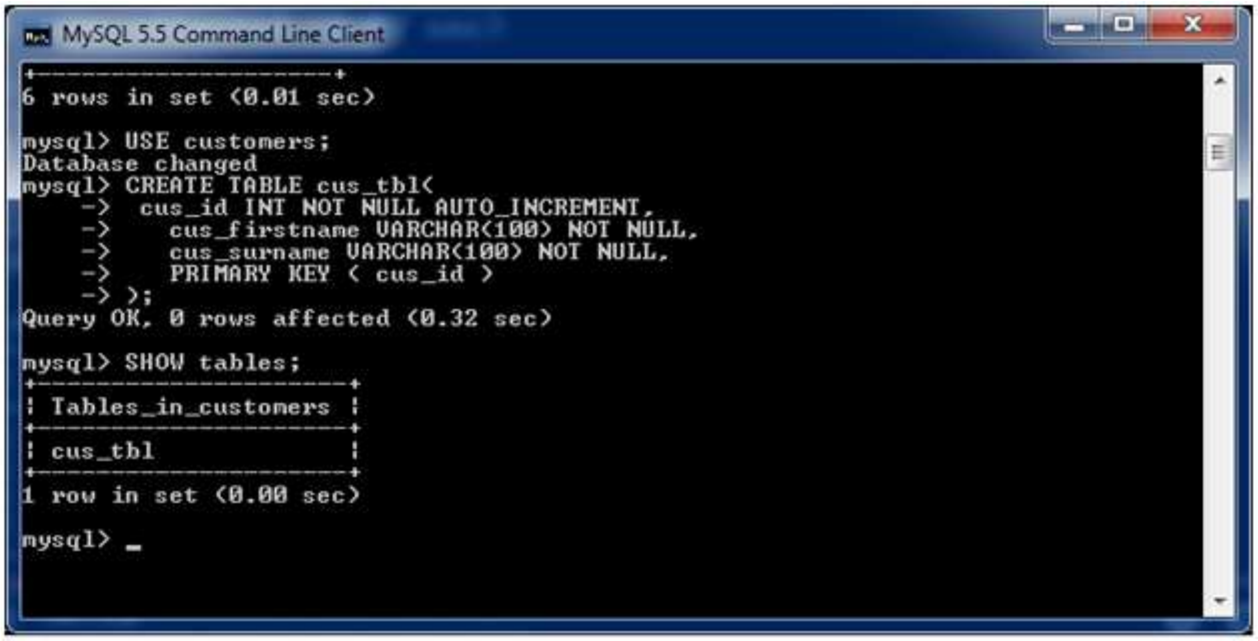


```
| Database           |
+--------------------+
| information_schema |
| customers          |
| mysql              |
| performance_schema |
| sssit              |
| test               |
+--------------------+
6 rows in set (0.01 sec)

mysql> USE customers;
Database changed
mysql> CREATE TABLE cus_tbl(
    ->  cus_id INT NOT NULL AUTO_INCREMENT,
    ->    cus_firstname VARCHAR(100) NOT NULL,
    ->    cus_surname VARCHAR(100) NOT NULL,
    ->    PRIMARY KEY ( cus_id )
    -> );
Query OK, 0 rows affected (0.32 sec)

mysql>
```

See the created table:

Use the following command to see the table already created:

1. SHOW tables;

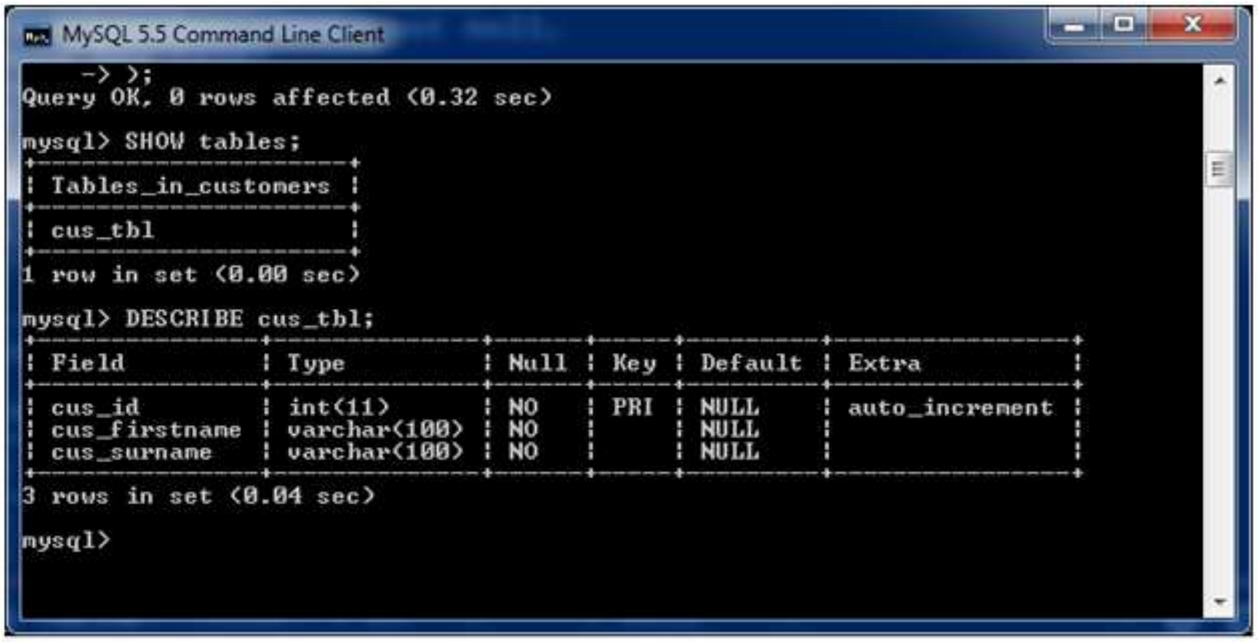


```
+----------------------+
6 rows in set (0.01 sec)

mysql> USE customers;
Database changed
mysql> CREATE TABLE cus_tbl(
    ->  cus_id INT NOT NULL AUTO_INCREMENT,
    ->    cus_firstname VARCHAR(100) NOT NULL,
    ->    cus_surname VARCHAR(100) NOT NULL,
    ->    PRIMARY KEY ( cus_id )
    -> );
Query OK, 0 rows affected (0.32 sec)

mysql> SHOW tables;
+----------------------+
| Tables_in_customers |
+----------------------+
| cus_tbl              |
+----------------------+
1 row in set (0.00 sec)

mysql> _
```

See the table structure:

Use the following command to see the table already created:

1. DESCRIBE cus_tbl;



```
    -> );
Query OK, 0 rows affected (0.32 sec)

mysql> SHOW tables;
+----------------------+
| Tables_in_customers |
+----------------------+
| cus_tbl              |
+----------------------+
1 row in set (0.00 sec)

mysql> DESCRIBE cus_tbl;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| cus_id        | int(11)      | NO   | PRI | NULL    | auto_increment |
| cus_firstname | varchar(100) | NO   |     | NULL    |                |
| cus_surname   | varchar(100) | NO   |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
3 rows in set (0.04 sec)

mysql>
```

MySQL ALTER Table

MySQL ALTER statement is used when you want to change the name of your table or any table field. It is also used to add or delete an existing column in a table.

The ALTER statement is always used with "ADD", "DROP" and "MODIFY" commands according to the situation.

1) ADD a column in the table

Syntax:

1. **ALTER TABLE** table_name
2. **ADD** new_column_name column_definition
3. [**FIRST** | **AFTER** column_name];

Parameters

table_name: It specifies the name of the table that you want to modify.

new_column_name: It specifies the name of the new column that you want to add to the table.

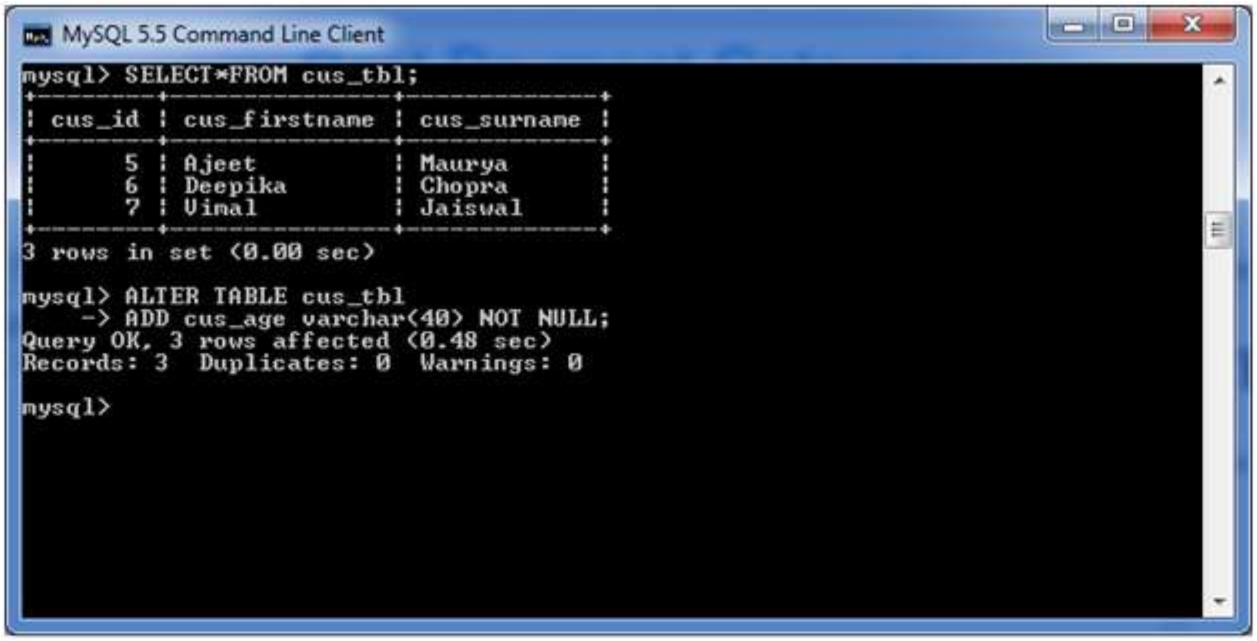
column_definition: It specifies the data type and definition of the column (NULL or NOT NULL, etc).

FIRST | AFTER column_name: It is optional. It tells MySQL where in the table to create the column. If this parameter is not specified, the new column will be added to the end of the table.

Example:
In this example, we add a new column "cus_age" in the existing table "cus_tbl".
Use the following query to do this:

1. ---
2. **ALTER TABLE** cus_tbl **ADD** cus_age **varchar**(40) NOT NULL;
3. **--**

Output:

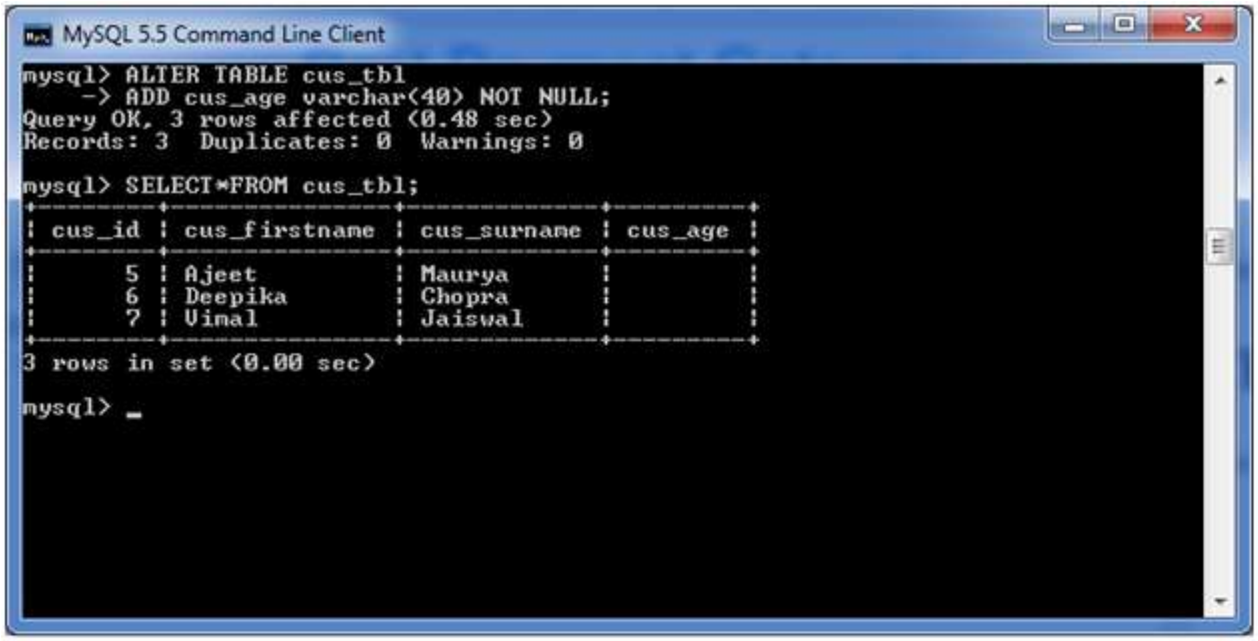


```
mysql> SELECT*FROM cus_tbl;
+--------+---------------+-------------+
| cus_id | cus_firstname | cus_surname |
+--------+---------------+-------------+
|      5 | Ajeet         | Maurya      |
|      6 | Deepika       | Chopra      |
|      7 | Vimal         | Jaiswal     |
+--------+---------------+-------------+
3 rows in set (0.00 sec)

mysql> ALTER TABLE cus_tbl
    -> ADD cus_age varchar(40) NOT NULL;
Query OK, 3 rows affected (0.48 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
```

See the recently added column:

1. **SELECT*** **FROM** cus_tbl;

Output:



```
mysql> ALTER TABLE cus_tbl
    -> ADD cus_age varchar(40) NOT NULL;
Query OK, 3 rows affected (0.48 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT*FROM cus_tbl;
+--------+---------------+-------------+---------+
| cus_id | cus_firstname | cus_surname | cus_age |
+--------+---------------+-------------+---------+
|      5 | Ajeet         | Maurya      |         |
|      6 | Deepika       | Chopra      |         |
|      7 | Vimal         | Jaiswal     |         |
+--------+---------------+-------------+---------+
3 rows in set (0.00 sec)

mysql> _
```

2) Add multiple columns in the table

Syntax:

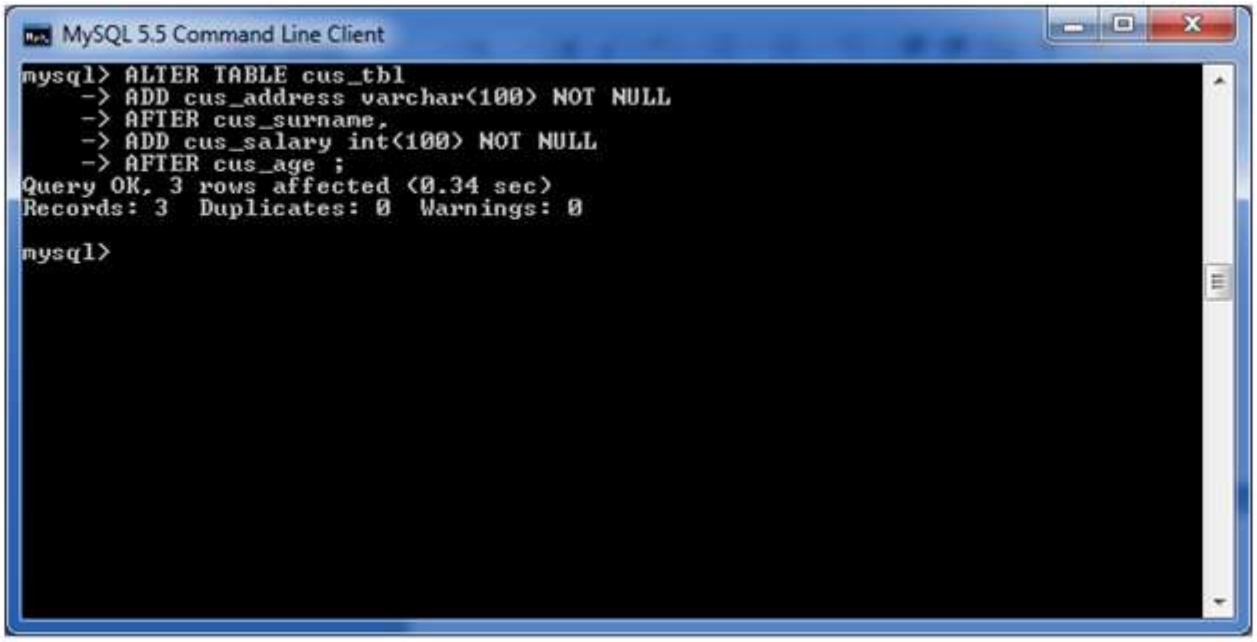
```
ALTER TABLE table_name
ADD new_column_name column_definition
[ FIRST | AFTER column_name ],
ADD new_column_name column_definition
[ FIRST | AFTER column_name ],
 ...
;
```

Example:

In this example, we add two new columns "cus_address", and cus_salary in the existing table "cus_tbl". cus_address is added after cus_surname column and cus_salary is added after cus_age column.

Use the following query to do this:

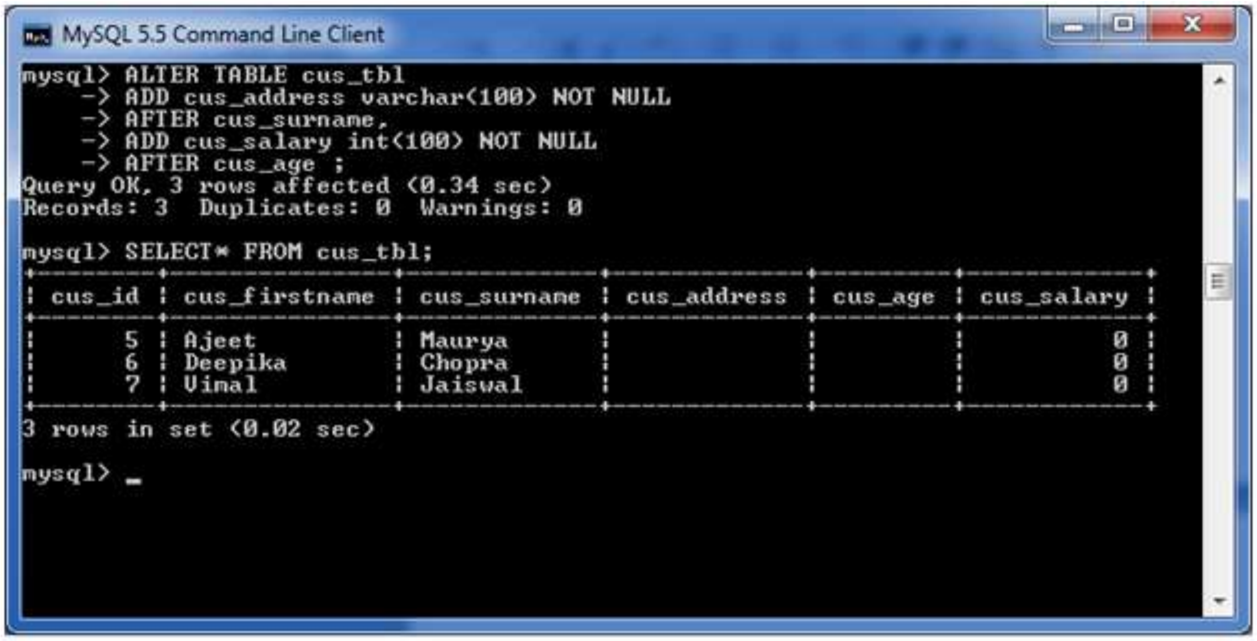
1. ALTER TABLE cus_tbl ADD cus_address varchar(100) NOT NULL
2. AFTER cus_surname, ADD cus_salary int(100) NOT NULL AFTER cus_age ;



```
mysql> ALTER TABLE cus_tbl
    -> ADD cus_address varchar(100) NOT NULL
    -> AFTER cus_surname,
    -> ADD cus_salary int(100) NOT NULL
    -> AFTER cus_age ;
Query OK, 3 rows affected (0.34 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
```

See the recently added columns:

1. **SELECT*** **FROM** cus_tbl;



```
mysql> ALTER TABLE cus_tbl
    -> ADD cus_address varchar(100) NOT NULL
    -> AFTER cus_surname,
    -> ADD cus_salary int(100) NOT NULL
    -> AFTER cus_age ;
Query OK, 3 rows affected (0.34 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM cus_tbl;
+--------+---------------+-------------+-------------+---------+------------+
| cus_id | cus_firstname | cus_surname | cus_address | cus_age | cus_salary |
+--------+---------------+-------------+-------------+---------+------------+
|      5 | Ajeet         | Maurya      |             |         |          0 |
|      6 | Deepika       | Chopra      |             |         |          0 |
|      7 | Vimal         | Jaiswal     |             |         |          0 |
+--------+---------------+-------------+-------------+---------+------------+
3 rows in set (0.02 sec)

mysql> _
```

3) MODIFY column in the table

The MODIFY command is used to change the column definition of the table.

Syntax:

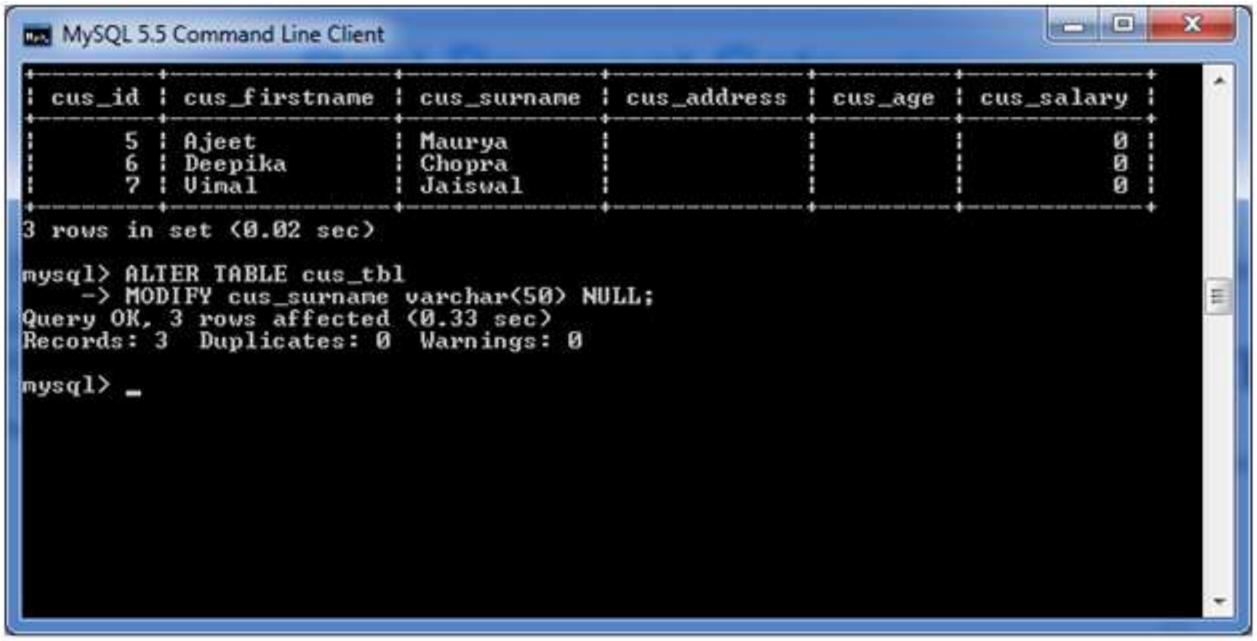
1. **ALTER TABLE** table_name
2. **MODIFY** column_name column_definition
3. [**FIRST** | **AFTER** column_name];

Example:

In this example, we modify the column cus_surname to be a data type of varchar(50) and force the column to allow NULL values.

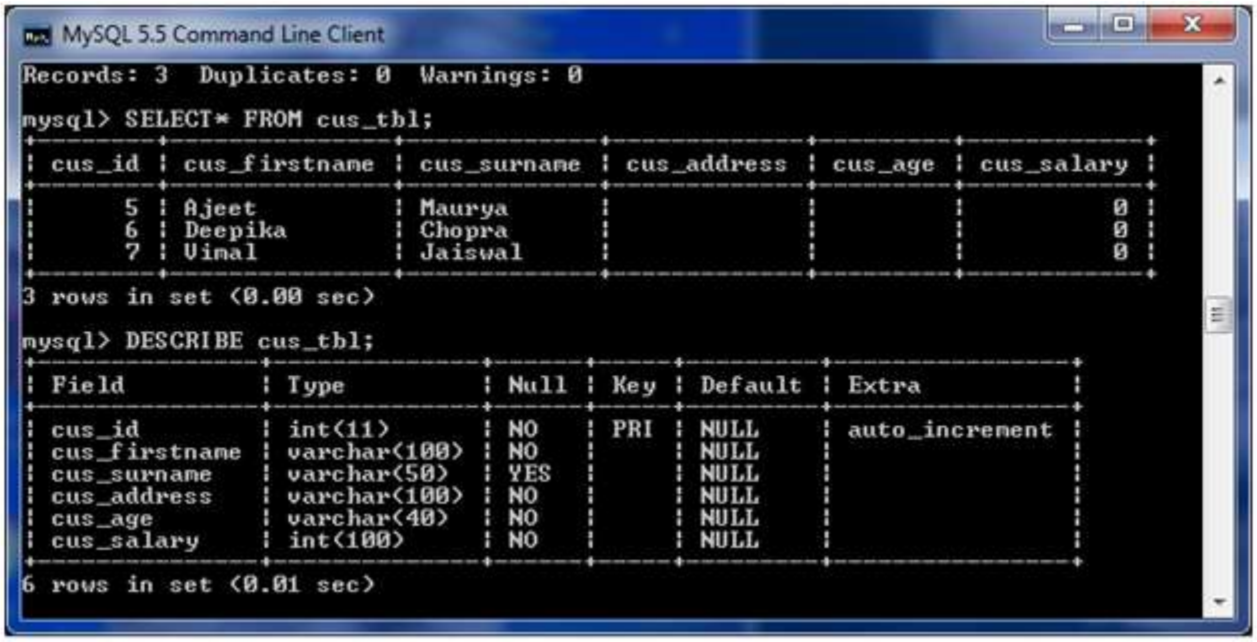
Use the following query to do this:

1. **ALTER TABLE** cus_tbl **MODIFY** cus_surname **varchar**(50) NULL;



```
+--------+---------------+-------------+-------------+---------+------------+
| cus_id | cus_firstname | cus_surname | cus_address | cus_age | cus_salary |
+--------+---------------+-------------+-------------+---------+------------+
|      5 | Ajeet         | Maurya      |             |         |          0 |
|      6 | Deepika       | Chopra      |             |         |          0 |
|      7 | Vimal         | Jaiswal     |             |         |          0 |
+--------+---------------+-------------+-------------+---------+------------+
3 rows in set (0.02 sec)

mysql> ALTER TABLE cus_tbl
    -> MODIFY cus_surname varchar(50) NULL;
Query OK, 3 rows affected (0.33 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> _
```

See the table structure:



```
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM cus_tbl;
+--------+---------------+-------------+-------------+---------+------------+
| cus_id | cus_firstname | cus_surname | cus_address | cus_age | cus_salary |
+--------+---------------+-------------+-------------+---------+------------+
|      5 | Ajeet         | Maurya      |             |         |          0 |
|      6 | Deepika       | Chopra      |             |         |          0 |
|      7 | Vimal         | Jaiswal     |             |         |          0 |
+--------+---------------+-------------+-------------+---------+------------+
3 rows in set (0.00 sec)

mysql> DESCRIBE cus_tbl;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| cus_id        | int(11)      | NO   | PRI | NULL    | auto_increment |
| cus_firstname | varchar(100) | NO   |     | NULL    |                |
| cus_surname   | varchar(50)  | YES  |     | NULL    |                |
| cus_address   | varchar(100) | NO   |     | NULL    |                |
| cus_age       | varchar(40)  | NO   |     | NULL    |                |
| cus_salary    | int(100)     | NO   |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
6 rows in set (0.01 sec)
```

4) DROP column in table

Syntax:

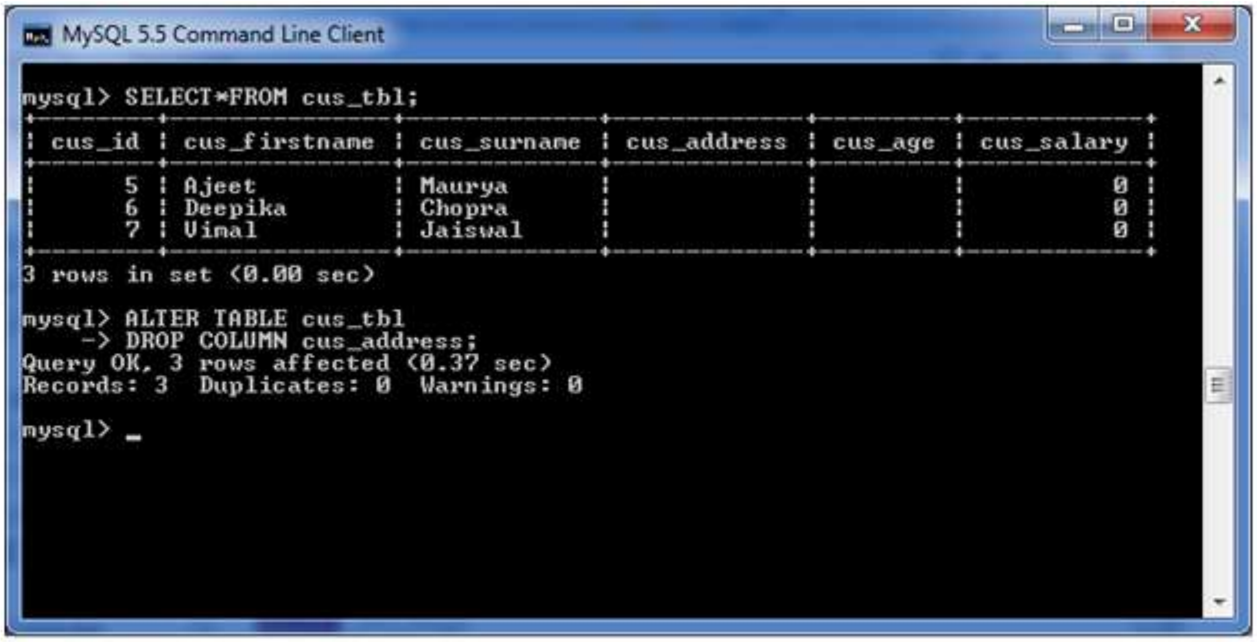
1. **ALTER TABLE** table_name **DROP COLUMN** column_name;

Let's take an example to drop the column name "cus_address" from the table "cus_tbl".

Use the following query to do this:

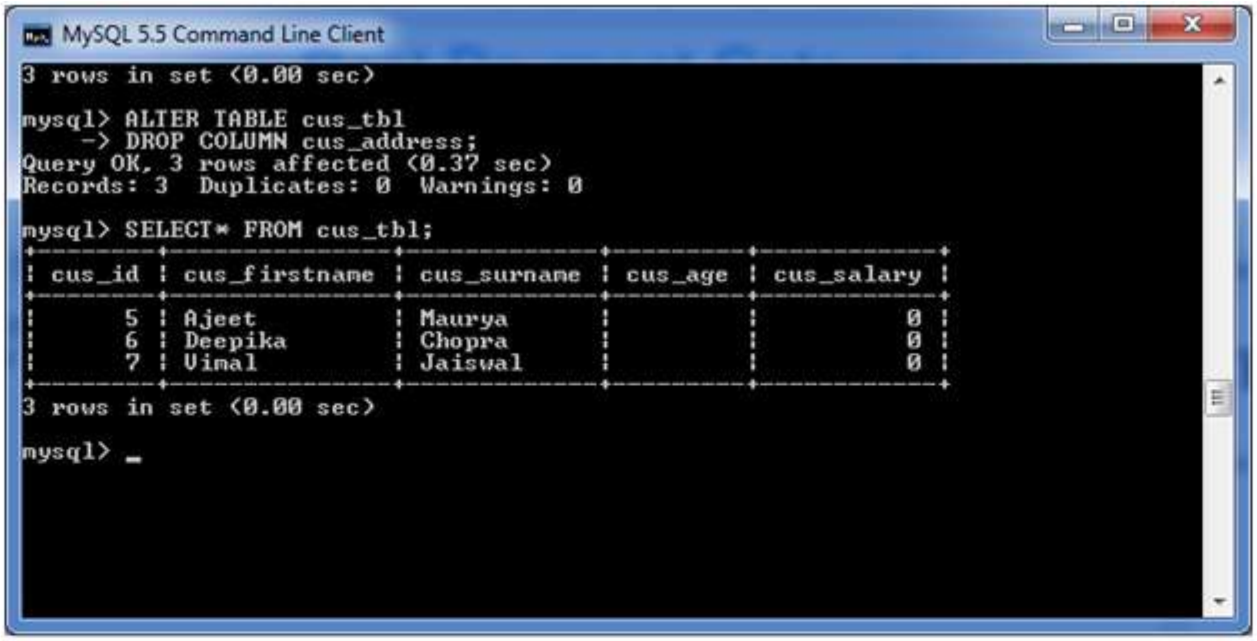
1. **ALTER TABLE** cus_tbl **DROP COLUMN** cus_address;

Output:



```
mysql> SELECT*FROM cus_tbl;
+--------+---------------+-------------+-------------+---------+------------+
| cus_id | cus_firstname | cus_surname | cus_address | cus_age | cus_salary |
+--------+---------------+-------------+-------------+---------+------------+
|      5 | Ajeet         | Maurya      |             |         |          0 |
|      6 | Deepika       | Chopra      |             |         |          0 |
|      7 | Vimal         | Jaiswal     |             |         |          0 |
+--------+---------------+-------------+-------------+---------+------------+
3 rows in set (0.00 sec)

mysql> ALTER TABLE cus_tbl
    -> DROP COLUMN cus_address;
Query OK, 3 rows affected (0.37 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> _
```

See the table structure:



```
3 rows in set (0.00 sec)

mysql> ALTER TABLE cus_tbl
    -> DROP COLUMN cus_address;
Query OK, 3 rows affected (0.37 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM cus_tbl;
+--------+---------------+-------------+---------+------------+
| cus_id | cus_firstname | cus_surname | cus_age | cus_salary |
+--------+---------------+-------------+---------+------------+
|      5 | Ajeet         | Maurya      |         |          0 |
|      6 | Deepika       | Chopra      |         |          0 |
|      7 | Vimal         | Jaiswal     |         |          0 |
+--------+---------------+-------------+---------+------------+
3 rows in set (0.00 sec)

mysql> _
```

5) RENAME column in table

Syntax:

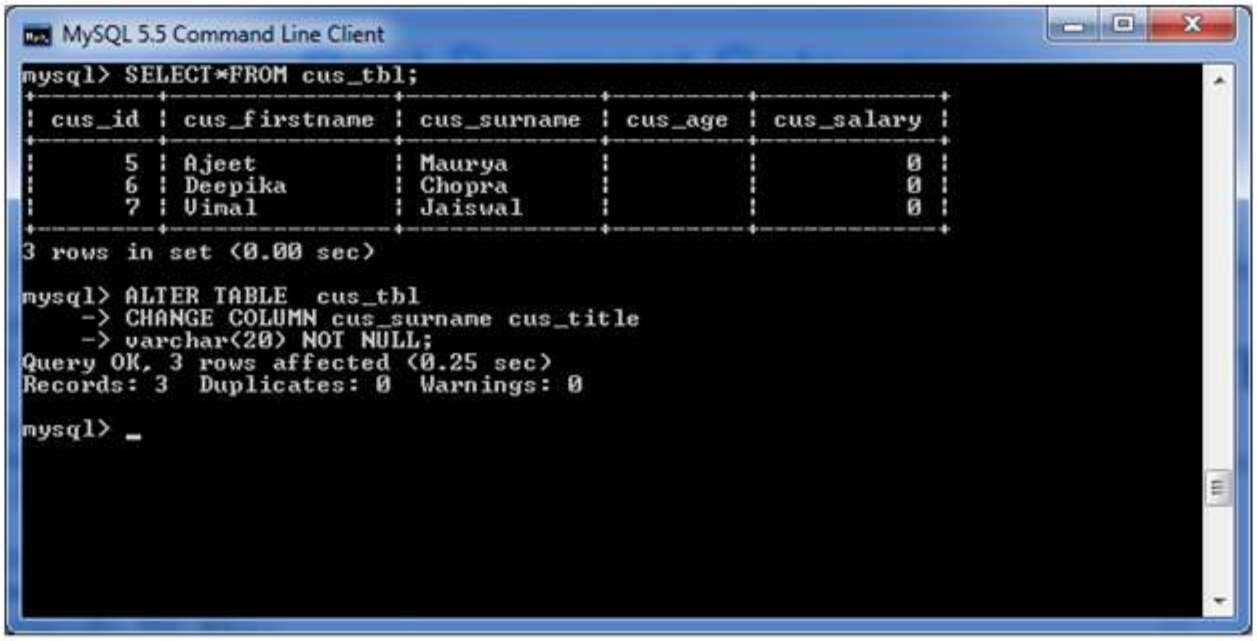
1. **ALTER TABLE** table_name
2. CHANGE **COLUMN** old_name new_name
3. column_definition
4. [**FIRST** | **AFTER** column_name]

Example:

In this example, we will change the column name "cus_surname" to "cus_title".
Use the following query to do this:

1. **ALTER TABLE** cus_tbl CHANGE **COLUMN** cus_surname cus_title **varchar**(20) NOT NULL;

Output:



```
mysql> SELECT*FROM cus_tbl;
+--------+---------------+-------------+---------+------------+
| cus_id | cus_firstname | cus_surname | cus_age | cus_salary |
+--------+---------------+-------------+---------+------------+
|      5 | Ajeet         | Maurya      |         |          0 |
|      6 | Deepika       | Chopra      |         |          0 |
|      7 | Vimal         | Jaiswal     |         |          0 |
+--------+---------------+-------------+---------+------------+
3 rows in set (0.00 sec)

mysql> ALTER TABLE  cus_tbl
    -> CHANGE COLUMN cus_surname cus_title
    -> varchar(20) NOT NULL;
Query OK, 3 rows affected (0.25 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> _
```

6) RENAME table

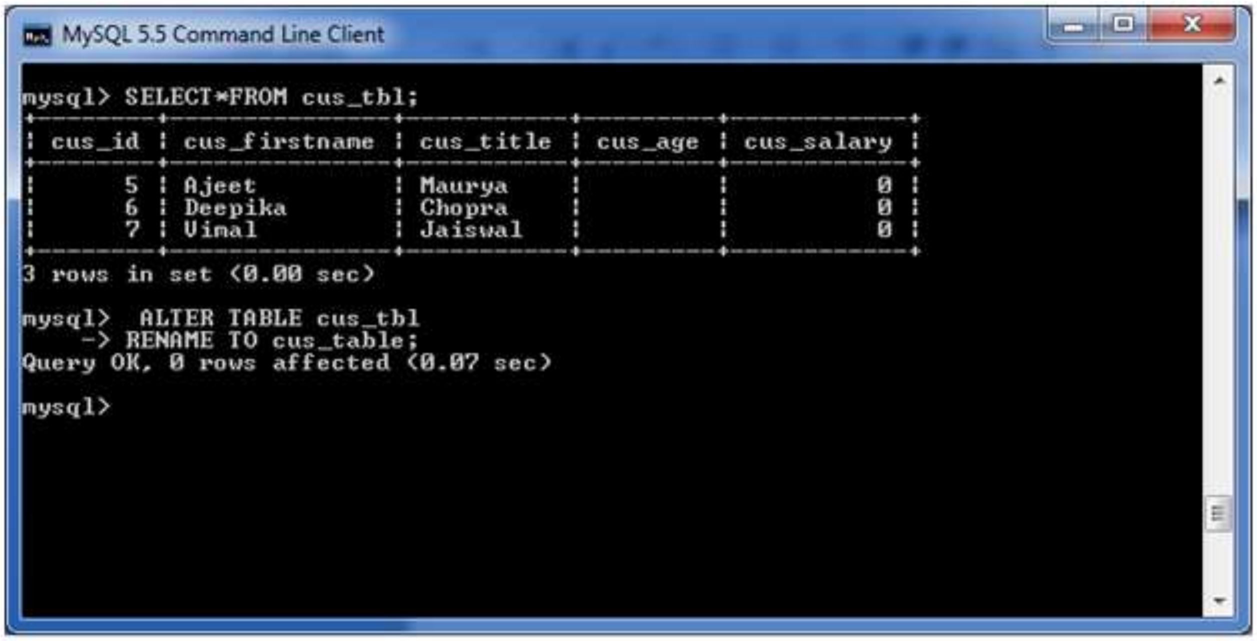
Syntax:

1. **ALTER TABLE** table_name RENAME **TO** new_table_name;

Example:
In this example, the table name cus_tbl is renamed as cus_table.

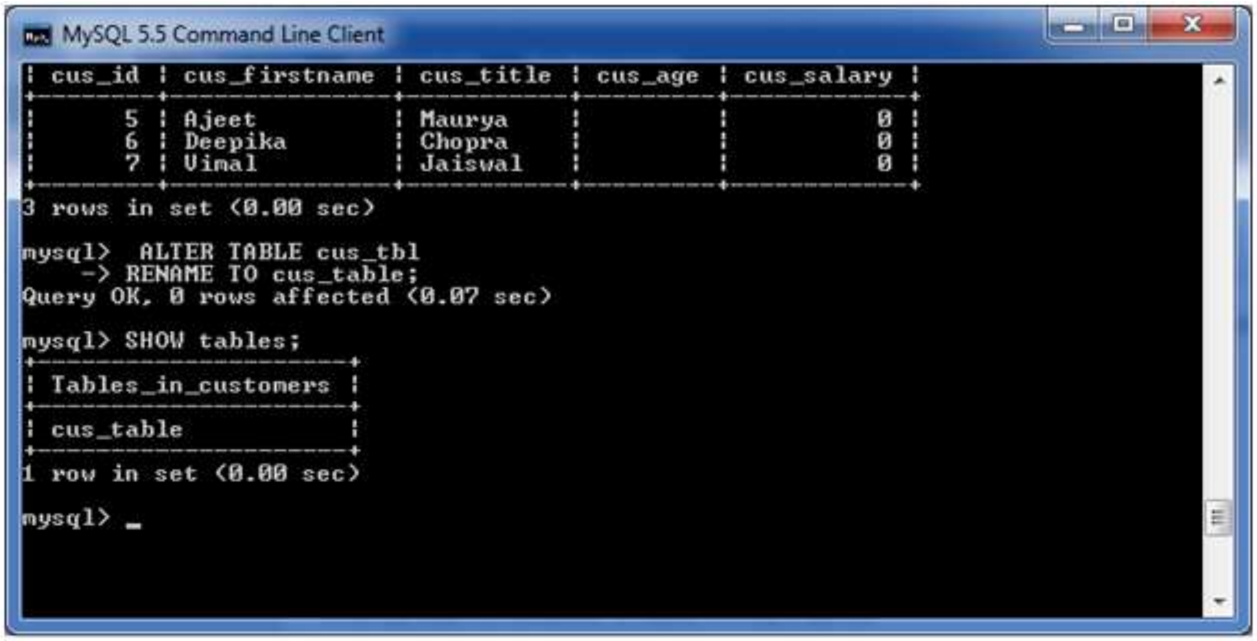
1. **ALTER TABLE** cus_tbl RENAME **TO** cus_table;

Output:



```
mysql> SELECT*FROM cus_tbl;
+--------+---------------+-----------+---------+------------+
| cus_id | cus_firstname | cus_title | cus_age | cus_salary |
+--------+---------------+-----------+---------+------------+
|      5 | Ajeet         | Maurya    |         |          0 |
|      6 | Deepika       | Chopra    |         |          0 |
|      7 | Vimal         | Jaiswal   |         |          0 |
+--------+---------------+-----------+---------+------------+
3 rows in set (0.00 sec)

mysql>  ALTER TABLE cus_tbl
    -> RENAME TO cus_table;
Query OK, 0 rows affected (0.07 sec)

mysql>
```

See the renamed table:



```
| cus_id | cus_firstname | cus_title | cus_age | cus_salary |
+--------+---------------+-----------+---------+------------+
|      5 | Ajeet         | Maurya    |         |          0 |
|      6 | Deepika       | Chopra    |         |          0 |
|      7 | Vimal         | Jaiswal   |         |          0 |
+--------+---------------+-----------+---------+------------+
3 rows in set (0.00 sec)

mysql>  ALTER TABLE cus_tbl
    -> RENAME TO cus_table;
Query OK, 0 rows affected (0.07 sec)

mysql> SHOW tables;
+---------------------+
| Tables_in_customers |
+---------------------+
| cus_table           |
+---------------------+
1 row in set (0.00 sec)

mysql> _
```

MySQL ALTER Table – summary

1) ADD a column in the table

2) Add multiple columns in the table

3) MODIFY column in the table

4) DROP column in table

5) RENAME column in table

6) RENAME table

===================================

MySQL INSERT Statement

MySQL INSERT statement is used to insert data in MySQL table within the database. We can insert single or multiple records using a single query in MySQL.

Syntax:

The SQL INSERT INTO command is used to insert data in MySQL table. Following is a generic syntax:

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

Field name is optional. If you want to specify partial values, field name is mandatory.

Syntax for all fields:

1. **INSERT INTO** table_name **VALUES** (value1, value2,...valueN);

MySQL INSERT Example 1: for all fields

If you have to store all the field values, either specify all field name or don't specify any field.

Example:

INSERT INTO emp **VALUES** (7, 'Sonoo', 40000);

Or,

INSERT INTO emp(id,**name**,salary) **VALUES** (7, 'Sonoo', 40000);

In such case, it is mandatory to specify field names.

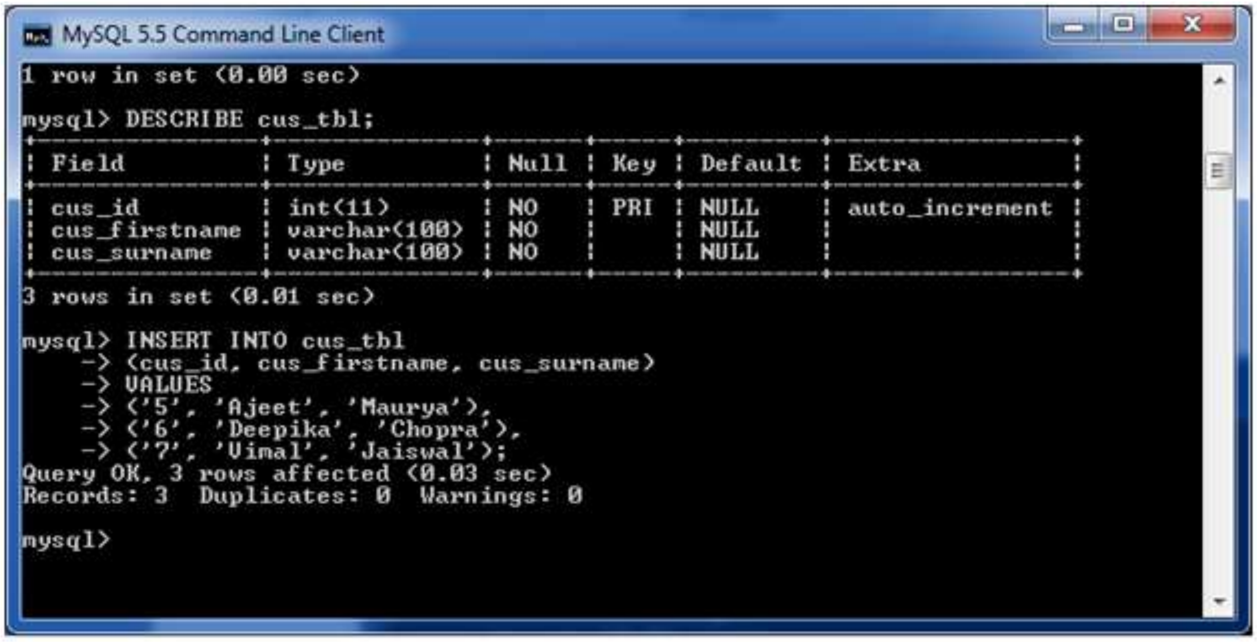
1. **INSERT INTO** emp(id,**name**) **VALUES** (7, 'Sonoo');

MySQL INSERT Example 3: inserting multiple records

Here, we are going to insert record in the "cus_tbl" table of "customers" database.

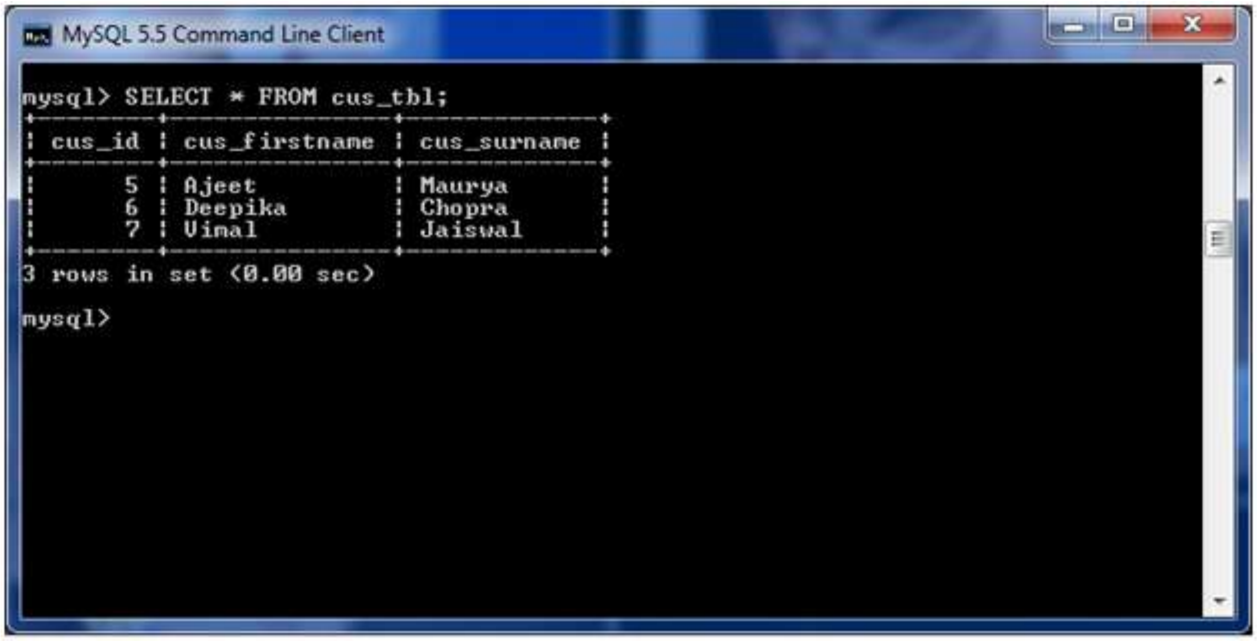
 INSERT INTO cus_tbl
(cus_id, cus_firstname, cus_surname)
VALUES
(5, 'Ajeet', 'Maurya'),
(6, 'Deepika', 'Chopra'),
(7, 'Vimal', 'Jaiswal');

Visual Representation:



```
1 row in set (0.00 sec)

mysql> DESCRIBE cus_tbl;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| cus_id        | int(11)      | NO   | PRI | NULL    | auto_increment |
| cus_firstname | varchar(100) | NO   |     | NULL    |                |
| cus_surname   | varchar(100) | NO   |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)

mysql> INSERT INTO cus_tbl
    -> (cus_id, cus_firstname, cus_surname)
    -> VALUES
    -> ('5', 'Ajeet', 'Maurya'),
    -> ('6', 'Deepika', 'Chopra'),
    -> ('7', 'Vimal', 'Jaiswal');
Query OK, 3 rows affected (0.03 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
```

See the data within the table by using the SELECT command:



```
mysql> SELECT * FROM cus_tbl;
+--------+---------------+-------------+
| cus_id | cus_firstname | cus_surname |
+--------+---------------+-------------+
|      5 | Ajeet         | Maurya      |
|      6 | Deepika       | Chopra      |
|      7 | Vimal         | Jaiswal     |
+--------+---------------+-------------+
3 rows in set (0.00 sec)

mysql>
```

MySQL UPDATE Query

MySQL UPDATE statement is used to update data of the MySQL table within the database. In real life scenario, records are changed over the period of time. So, we need to make changes in the values of the tables also. To do so, we need to use the UPDATE statement.

The **UPDATE** statement is used with the SET, and WHERE clauses. The **SET** clause is used to change the values of the specified column. We can update single or multiple columns at a time.

The **WHERE** clause is used to specify the condition, but it is optional.

Syntax:

Following is a generic syntax of UPDATE command to modify data into the MySQL table:

1. **UPDATE** table_name
2. **SET** field1=new-value1, field2=new-value2, ...
3. [**WHERE** Clause]

Note:

- One or more field can be updated altogether.
- Any condition can be specified by using WHERE clause.
- You can update values in a single table at a time.

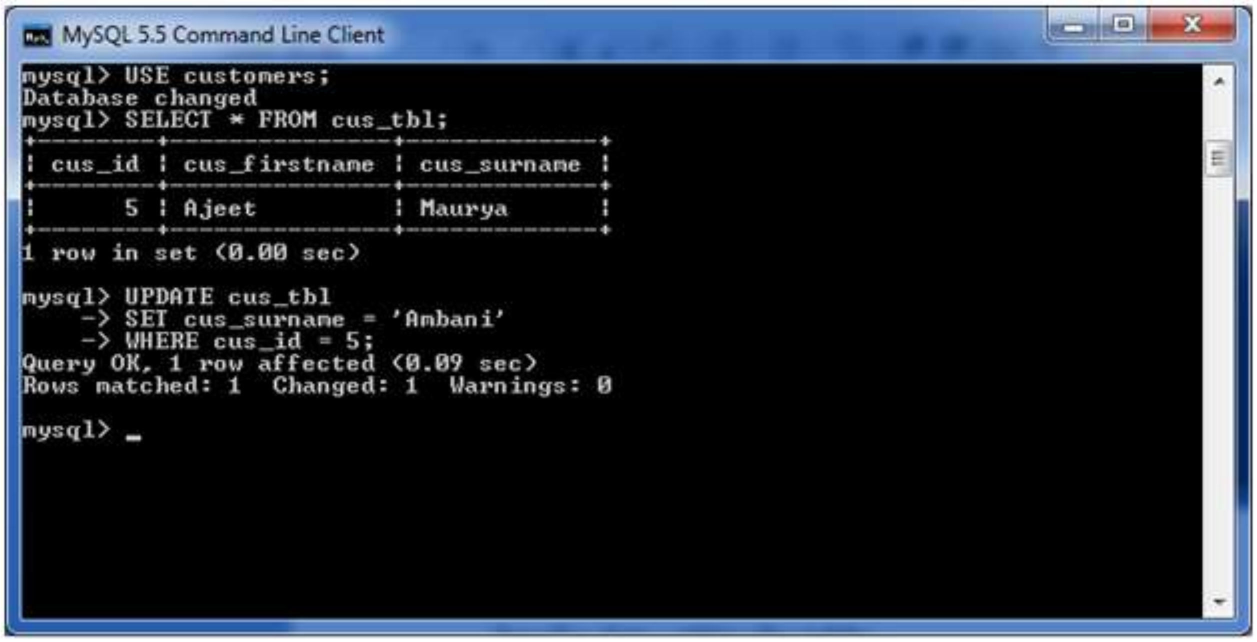
- WHERE clause is used to update selected rows in a table.

Example:

Here, we have a table "cus_tbl" within the database "customers". We are going to update the data within the table "cus_tbl".
This query will update cus_surname field for a record having cus_id as 5.

UPDATE cus_tbl
SET cus_surname = 'Ambani'
WHERE cus_id = 5;

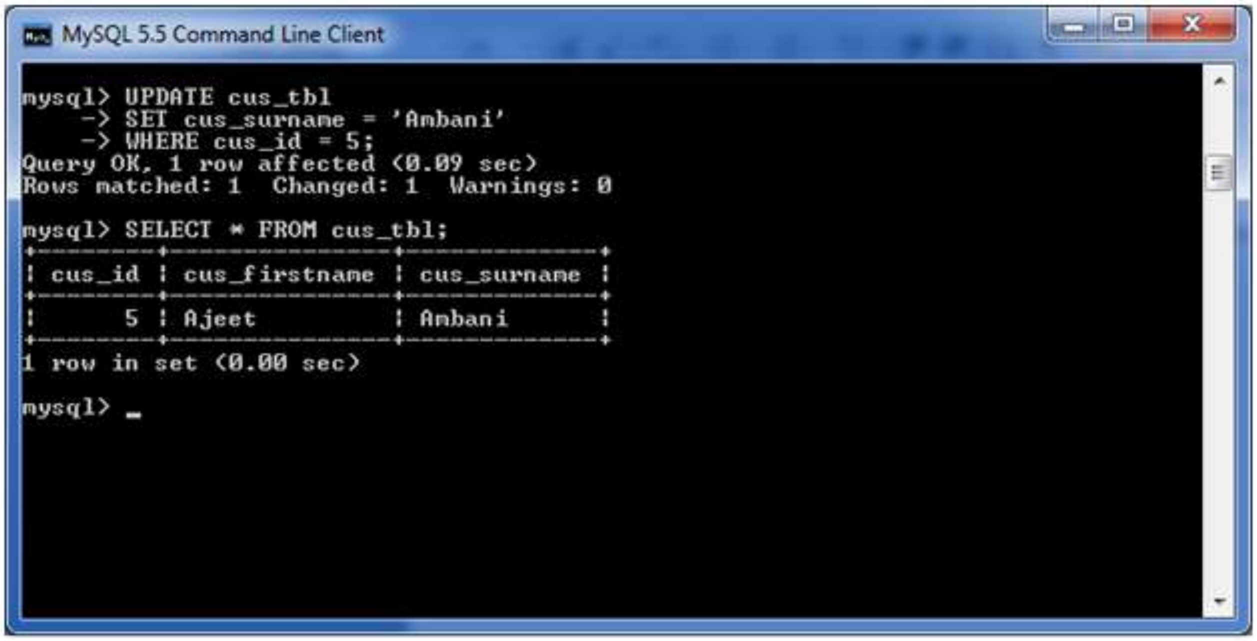
Visual Representation:



```
mysql> USE customers;
Database changed
mysql> SELECT * FROM cus_tbl;
+--------+---------------+-------------+
| cus_id | cus_firstname | cus_surname |
+--------+---------------+-------------+
|      5 | Ajeet         | Maurya      |
+--------+---------------+-------------+
1 row in set (0.00 sec)

mysql> UPDATE cus_tbl
    -> SET cus_surname = 'Ambani'
    -> WHERE cus_id = 5;
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> _
```

Output by SELECT query:

SELECT * **FROM** cus_tbl;



```
mysql> UPDATE cus_tbl
    -> SET cus_surname = 'Ambani'
    -> WHERE cus_id = 5;
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM cus_tbl;
+--------+---------------+-------------+
| cus_id | cus_firstname | cus_surname |
+--------+---------------+-------------+
|      5 | Ajeet         | Ambani      |
+--------+---------------+-------------+
1 row in set (0.00 sec)

mysql> _
```

Here, you can see that the table is updated as per your conditions.

MySQL DELETE Statement

MySQL DELETE statement is used to delete data from the MySQL table within the database. By using delete statement, we can delete records on the basis of conditions.

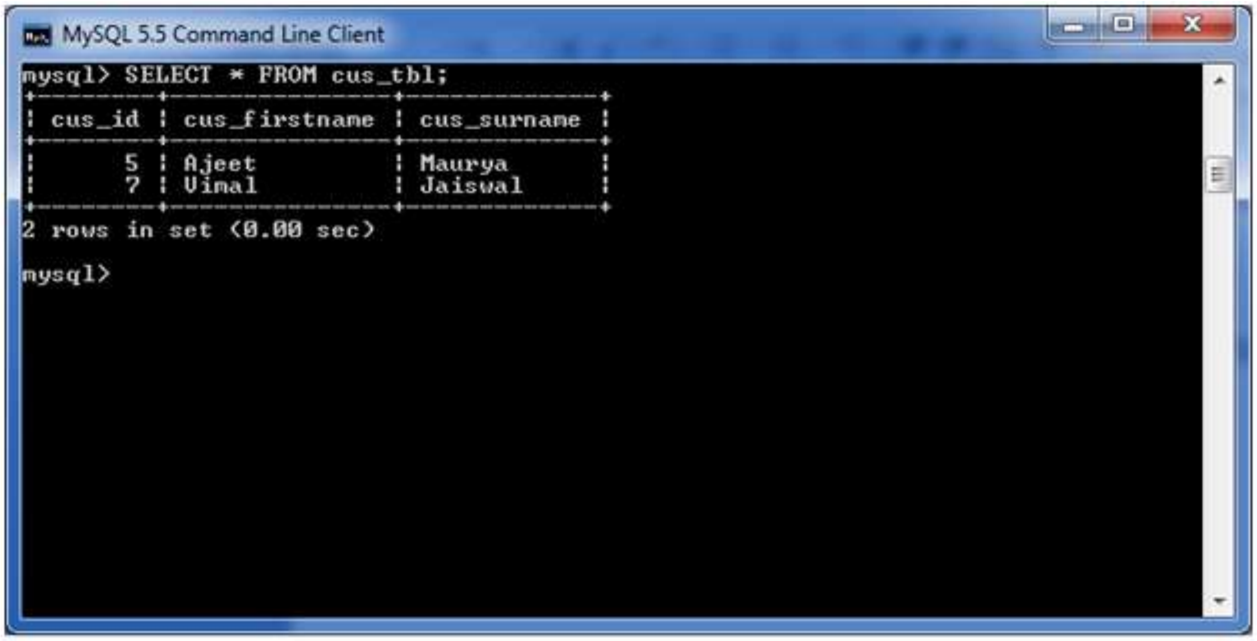
Syntax:

DELETE FROM table_name
WHERE
(Condition specified);

Example:

DELETE FROM cus_tbl
WHERE cus_id = 6;

Output:



```
mysql> SELECT * FROM cus_tbl;
+--------+---------------+-------------+
| cus_id | cus_firstname | cus_surname |
+--------+---------------+-------------+
|      5 | Ajeet         | Maurya      |
|      7 | Vimal         | Jaiswal     |
+--------+---------------+-------------+
2 rows in set (0.00 sec)

mysql>
```

MySQL SELECT Statement

The MySQL SELECT statement is used to fetch data from the one or more tables in MySQL. We can retrieve records of all fields or specified fields.

Syntax for specified fields:

1. **SELECT** expressions
2. **FROM** tables
3. [**WHERE** conditions];

Syntax for all fields:

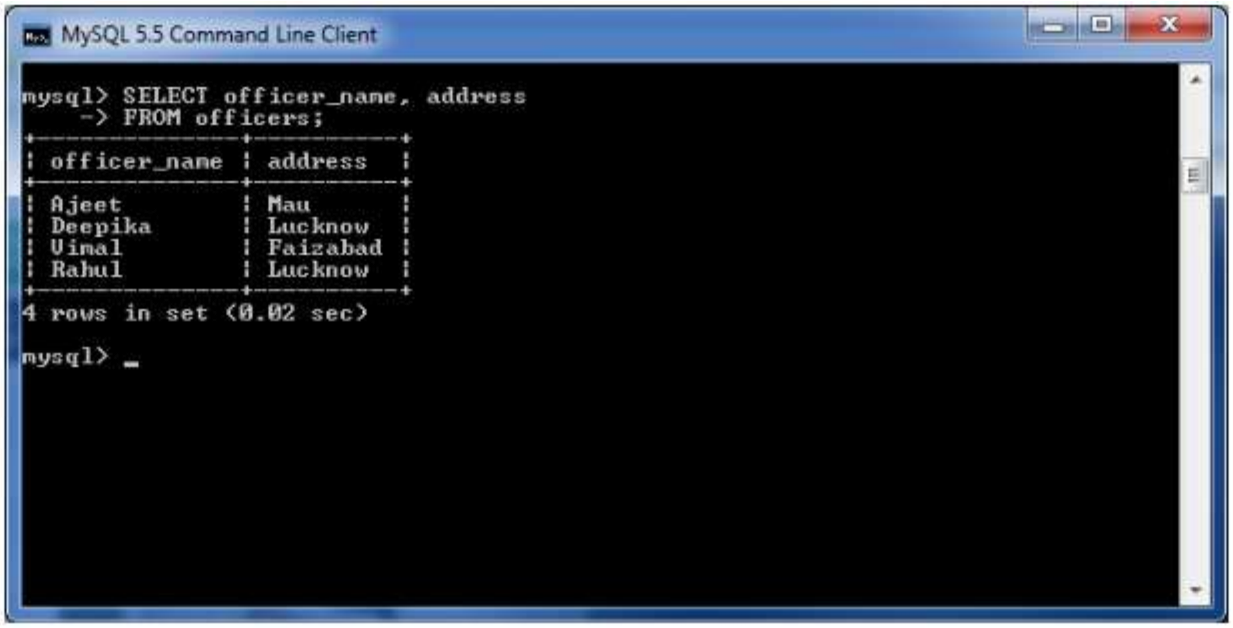
1. **SELECT** * **FROM** tables [**WHERE** conditions];

MySQL SELECT Example 1: for specified fields

In such case, it is mandatory to specify field names.

Example:

1. **SELECT** officer_name, address
2. **FROM** officers

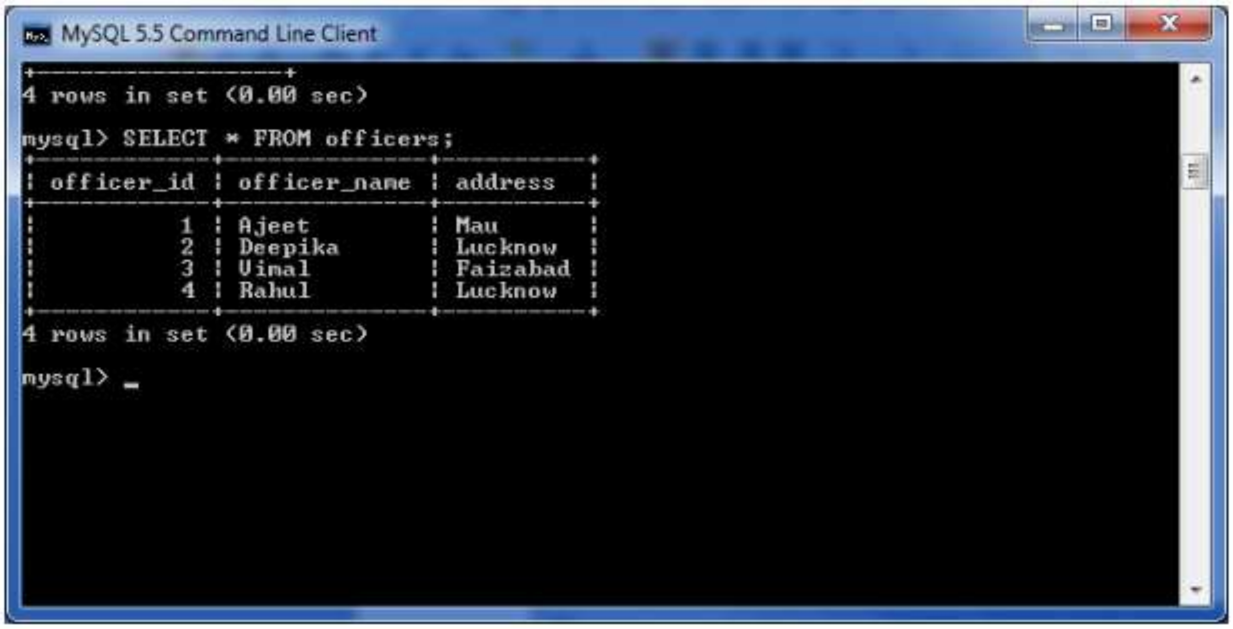


```
mysql> SELECT officer_name, address
    -> FROM officers;
+--------------+----------+
| officer_name | address  |
+--------------+----------+
| Ajeet        | Mau      |
| Deepika      | Lucknow  |
| Vimal        | Faizabad |
| Rahul        | Lucknow  |
+--------------+----------+
4 rows in set (0.02 sec)

mysql> _
```

MySQL SELECT Example 2: for all fields

In such case, we can specify either all fields or * (asterisk) symbol.

SELECT * **FROM** officers

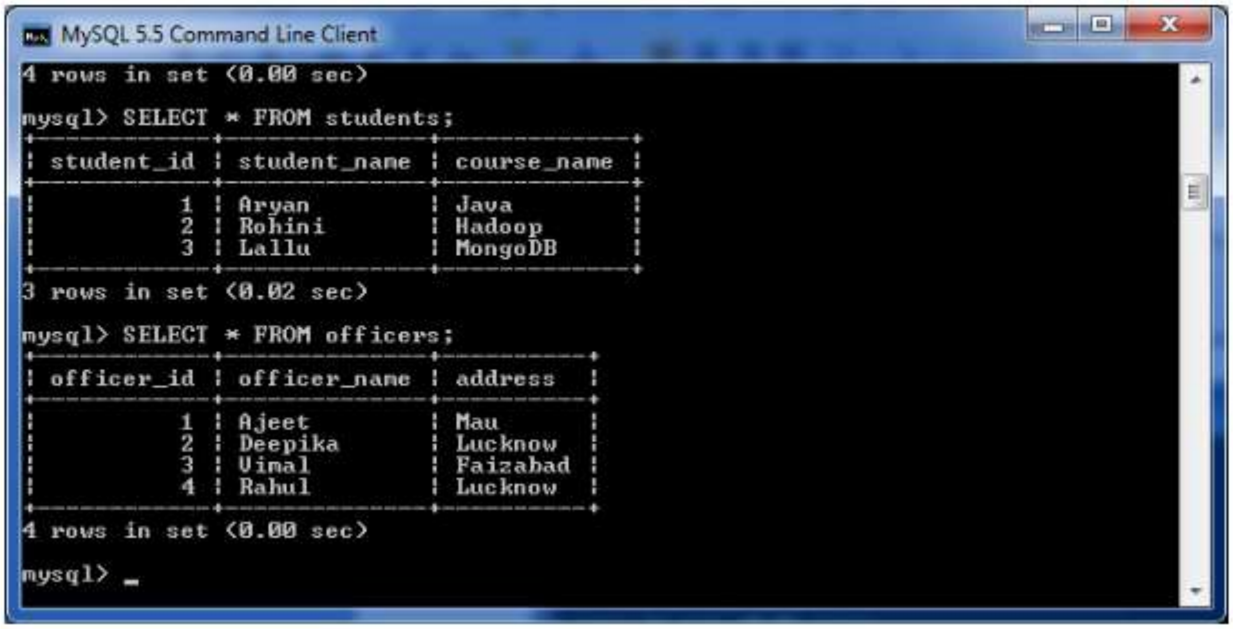


```
+--------------------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM officers;
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          1 | Ajeet        | Mau      |
|          2 | Deepika      | Lucknow  |
|          3 | Vimal        | Faizabad |
|          4 | Rahul        | Lucknow  |
+------------+--------------+----------+
4 rows in set (0.00 sec)

mysql> _
```

MySQL SELECT Example 3: from multiple tables

MySQL SELECT statement can also be used to retrieve records from multiple tables by using JOIN statement.

Let's take two tables "students" and "officers", having the following data.

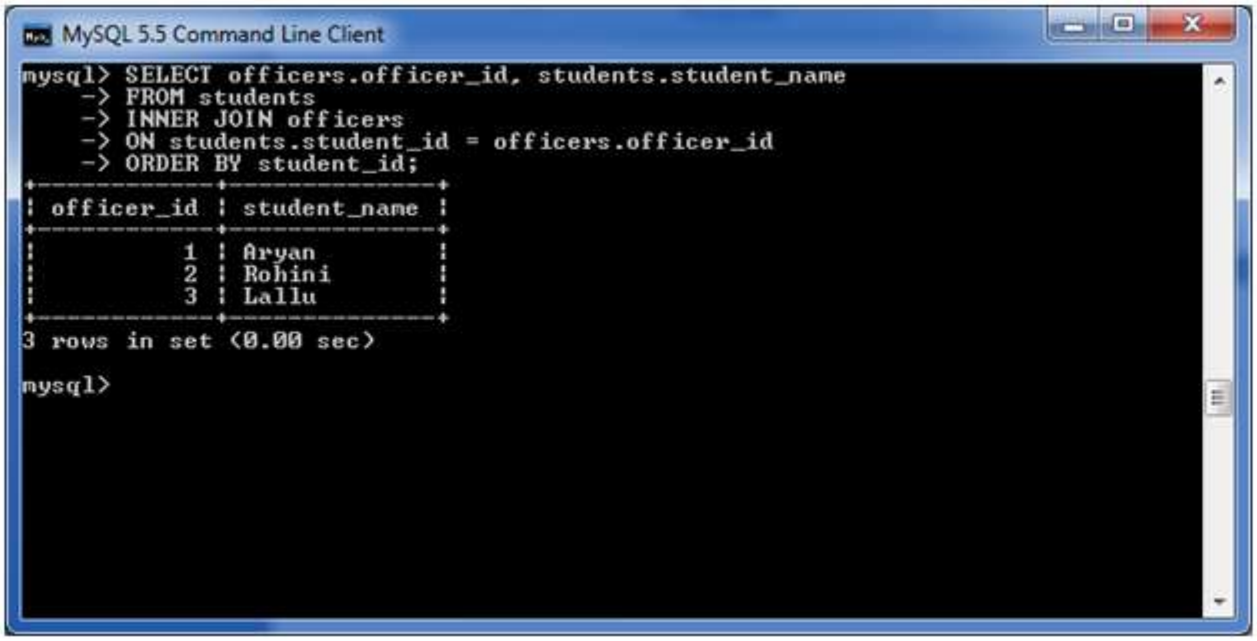


```
MySQL 5.5 Command Line Client
4 rows in set (0.00 sec)

mysql> SELECT * FROM students;
+------------+--------------+-------------+
| student_id | student_name | course_name |
+------------+--------------+-------------+
|          1 | Aryan        | Java        |
|          2 | Rohini       | Hadoop      |
|          3 | Lallu        | MongoDB     |
+------------+--------------+-------------+
3 rows in set (0.02 sec)

mysql> SELECT * FROM officers;
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          1 | Ajeet        | Mau      |
|          2 | Deepika      | Lucknow  |
|          3 | Vimal        | Faizabad |
|          4 | Rahul        | Lucknow  |
+------------+--------------+----------+
4 rows in set (0.00 sec)

mysql> _
```

Execute the following query:

SELECT officers.officer_id, students.student_name **FROM** students
INNER JOIN officers **ON** students.student_id = officers.officer_id
ORDER BY student_id;

Output:



```
MySQL 5.5 Command Line Client
mysql> SELECT officers.officer_id, students.student_name
    -> FROM students
    -> INNER JOIN officers
    -> ON students.student_id = officers.officer_id
    -> ORDER BY student_id;
+------------+--------------+
| officer_id | student_name |
+------------+--------------+
|          1 | Aryan        |
|          2 | Rohini       |
|          3 | Lallu        |
+------------+--------------+
3 rows in set (0.00 sec)

mysql>
```

MySQL TRUNCATE Table

MYSQL TRUNCATE statement removes the complete data without removing its structure.

The TRUNCATE TABLE statement is used when you want to delete the complete data from a table without removing the table structure.

Syntax:

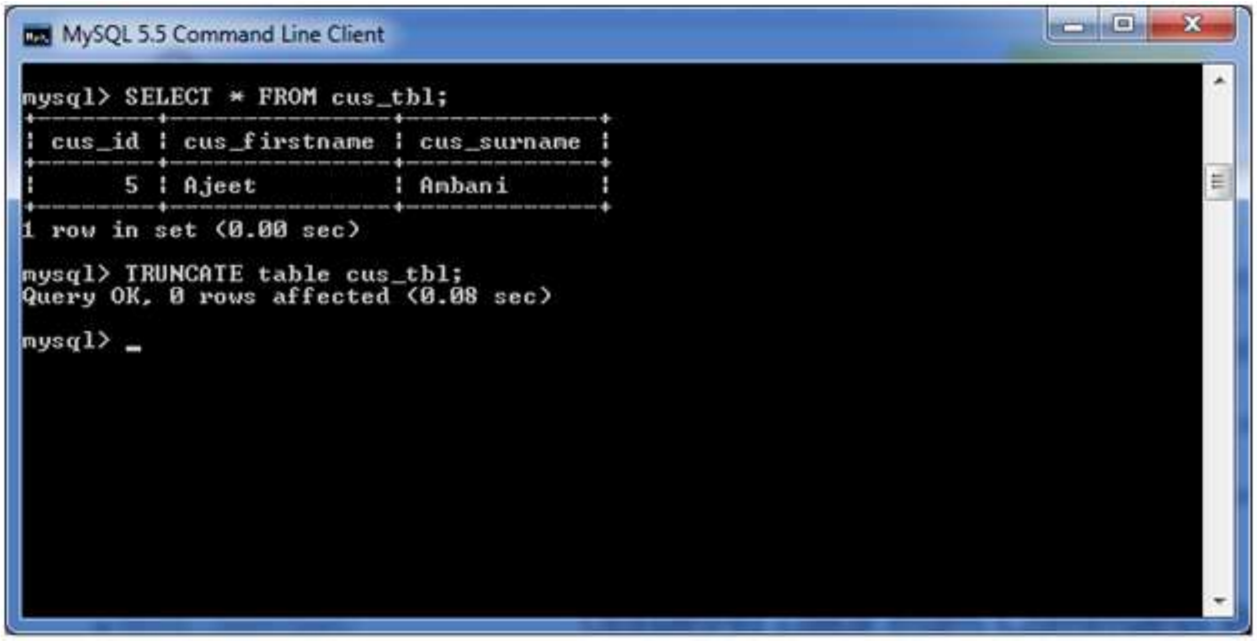
1. **TRUNCATE TABLE** table_name;

Example:

This example specifies how to truncate a table. In this example, we truncate the table "cus_tbl".

1. **TRUNCATE TABLE** cus_tbl;

Output:

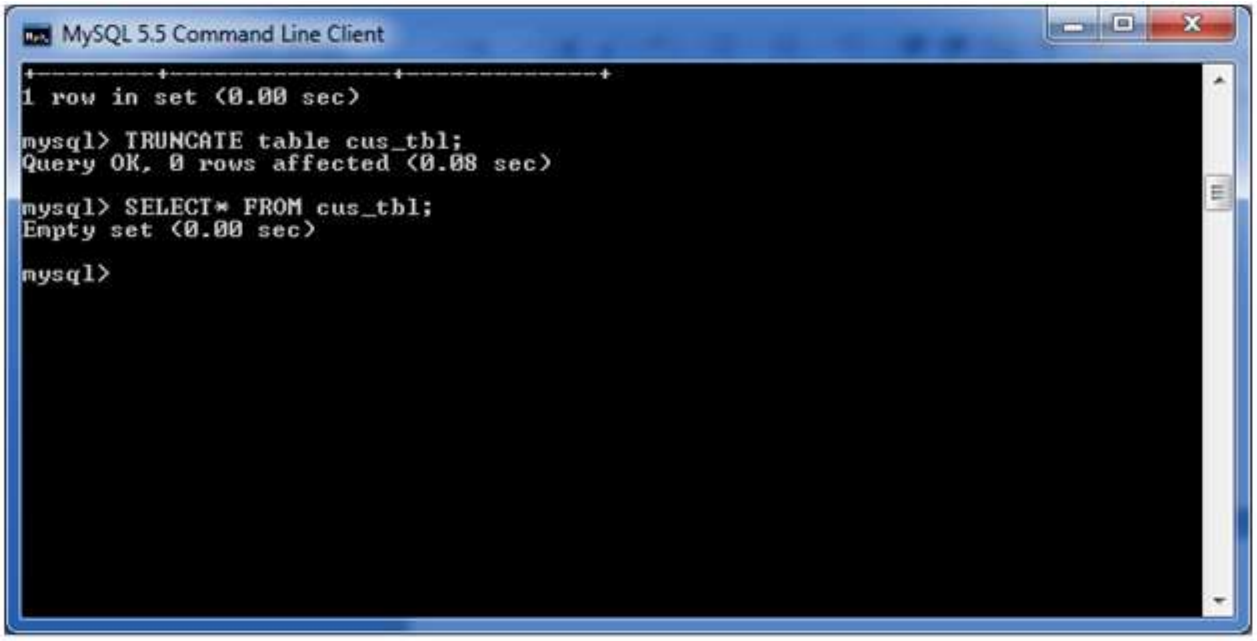


```
mysql> SELECT * FROM cus_tbl;
+--------+---------------+-------------+
| cus_id | cus_firstname | cus_surname |
+--------+---------------+-------------+
|      5 | Ajeet         | Ambani      |
+--------+---------------+-------------+
1 row in set (0.00 sec)

mysql> TRUNCATE table cus_tbl;
Query OK, 0 rows affected (0.08 sec)

mysql> _
```

See the table:

1. **SELECT*** **FROM** cus_tbl;

   Output:



MySQL 5.5 Command Line Client

```
+---------+-----------------+---------------+
1 row in set (0.00 sec)

mysql> TRUNCATE table cus_tbl;
Query OK, 0 rows affected (0.08 sec)

mysql> SELECT* FROM cus_tbl;
Empty set (0.00 sec)

mysql>
```

MySQL DROP Table

MYSQL DROP table statement removes the complete data with structure.

Syntax:

1. **DROP TABLE** table_name;

Example:

This example specifies how to drop a table. In this example, we are dropping the table "cus_tbl".

1. **DROP TABLE** cus_tbl;

MySQL TRUNCATE Table vs DROP Table

You can also use DROP TABLE command to delete complete table but it will remove complete table data and structure both. You need to re-create the table again if you have to store some data. But in the case of TRUNCATE TABLE, it removes only table data not structure. You don't need to re-create the table again because the table structure already exists.

MySQL View

In MySQL, View is a virtual table created by a query by joining one or more tables.

MySQL Create VIEW

A VIEW is created by SELECT statements. SELECT statements are used to take data from the source table to make a VIEW.

Syntax:

1. **CREATE** [OR REPLACE] **VIEW** view_name **AS**
2. **SELECT** columns
3. **FROM** tables
4. [**WHERE** conditions];   => if the condition is in square bracket – meaning its optional

Parameters:

OR REPLACE: It is optional. It is used when a VIEW already exist. If you do not specify this clause and the VIEW already exists, the CREATE VIEW statement will return an error.

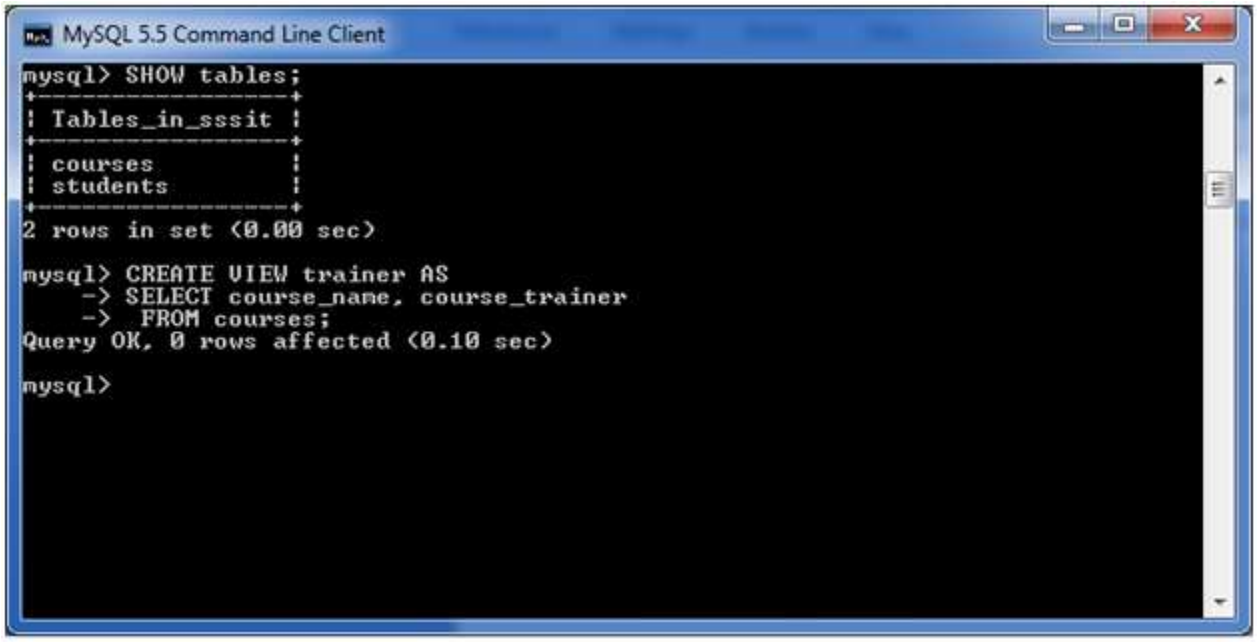
view_name: It specifies the name of the VIEW that you want to create in MySQL.

WHERE conditions: It is also optional. It specifies the conditions that must be met for the records to be included in the VIEW.

The following example will create a VIEW name "trainer". This is a virtual table made by taking data from the table "courses".

CREATE VIEW trainer **AS**

SELECT course_name, course_trainer
 FROM courses;


```
mysql> SHOW tables;
+------------------+
| Tables_in_sssit |
+------------------+
| courses          |
| students         |
+------------------+
2 rows in set (0.00 sec)

mysql> CREATE VIEW trainer AS
    -> SELECT course_name, course_trainer
    ->  FROM courses;
Query OK, 0 rows affected (0.10 sec)

mysql>
```

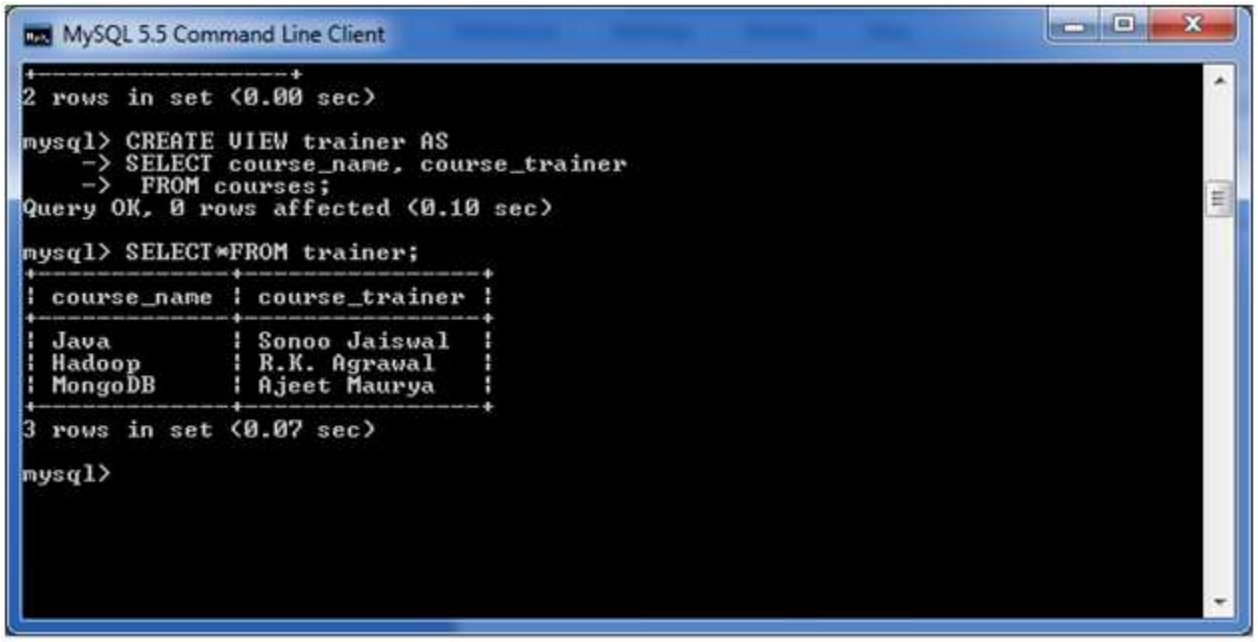
To see the created VIEW:

Syntax:

1. **SELECT** * **FROM** view_name;

Let's see how it looks the created VIEW:

1. **SELECT** * **FROM** trainer;


```
+------------------+
2 rows in set (0.00 sec)

mysql> CREATE VIEW trainer AS
    -> SELECT course_name, course_trainer
    ->  FROM courses;
Query OK, 0 rows affected (0.10 sec)

mysql> SELECT*FROM trainer;
+-------------+----------------+
| course_name | course_trainer |
+-------------+----------------+
| Java        | Sonoo Jaiswal  |
| Hadoop      | R.K. Agrawal   |
| MongoDB     | Ajeet Maurya   |
+-------------+----------------+
3 rows in set (0.07 sec)

mysql>
```

MySQL Update VIEW

In MYSQL, the ALTER VIEW statement is used to modify or update the already created VIEW without dropping it.

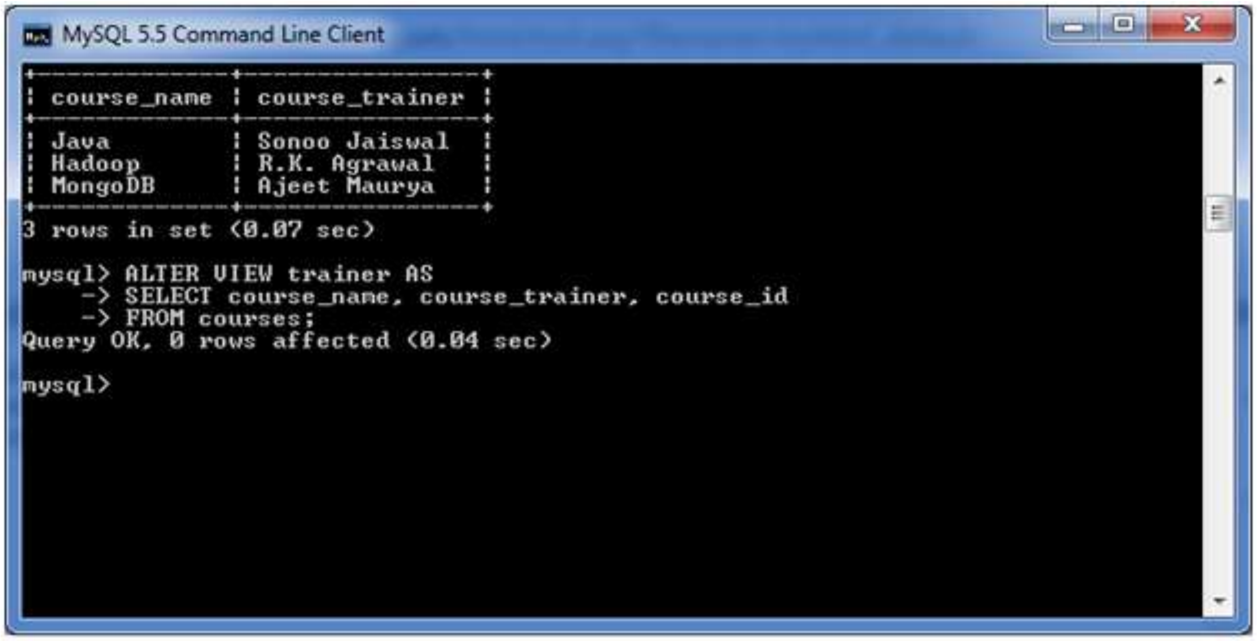
Syntax:

1. **ALTER VIEW** view_name **AS**
2. **SELECT** columns
3. **FROM table**
4. **WHERE** conditions;

Example:

The following example will alter the already created VIEW name "trainer" by adding a new column.

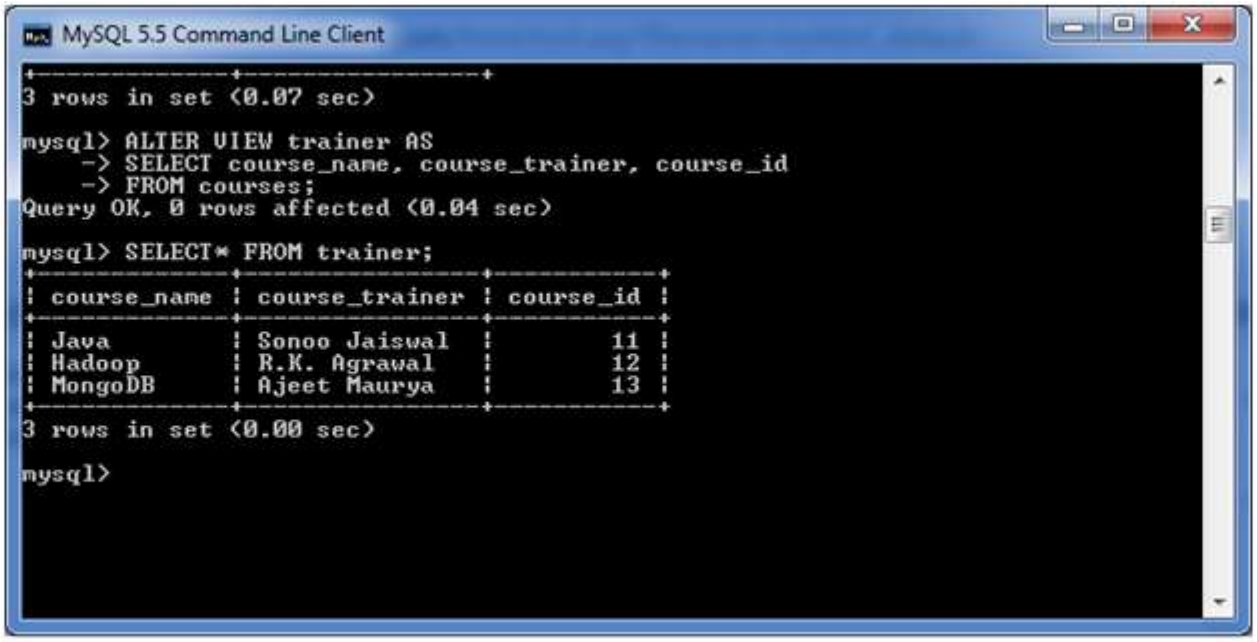
1. **ALTER VIEW** trainer **AS**
2. **SELECT** course_name, course_trainer, course_id
3. **FROM** courses;



```
MySQL 5.5 Command Line Client
+-------------+----------------+
| course_name | course_trainer |
+-------------+----------------+
| Java        | Sonoo Jaiswal  |
| Hadoop      | R.K. Agrawal   |
| MongoDB     | Ajeet Maurya   |
+-------------+----------------+
3 rows in set (0.07 sec)

mysql> ALTER VIEW trainer AS
    -> SELECT course_name, course_trainer, course_id
    -> FROM courses;
Query OK, 0 rows affected (0.04 sec)

mysql>
```

To see the altered VIEW:

1. **SELECT*FROM** trainer;



```
MySQL 5.5 Command Line Client
+-------------+----------------+
3 rows in set (0.07 sec)

mysql> ALTER VIEW trainer AS
    -> SELECT course_name, course_trainer, course_id
    -> FROM courses;
Query OK, 0 rows affected (0.04 sec)

mysql> SELECT* FROM trainer;
+-------------+----------------+-----------+
| course_name | course_trainer | course_id |
+-------------+----------------+-----------+
| Java        | Sonoo Jaiswal  |        11 |
| Hadoop      | R.K. Agrawal   |        12 |
| MongoDB     | Ajeet Maurya   |        13 |
+-------------+----------------+-----------+
3 rows in set (0.00 sec)

mysql>
```

MySQL Drop VIEW

You can drop the VIEW by using the DROP VIEW statement.

Syntax:

1. **DROP VIEW** [IF EXISTS] view_name;

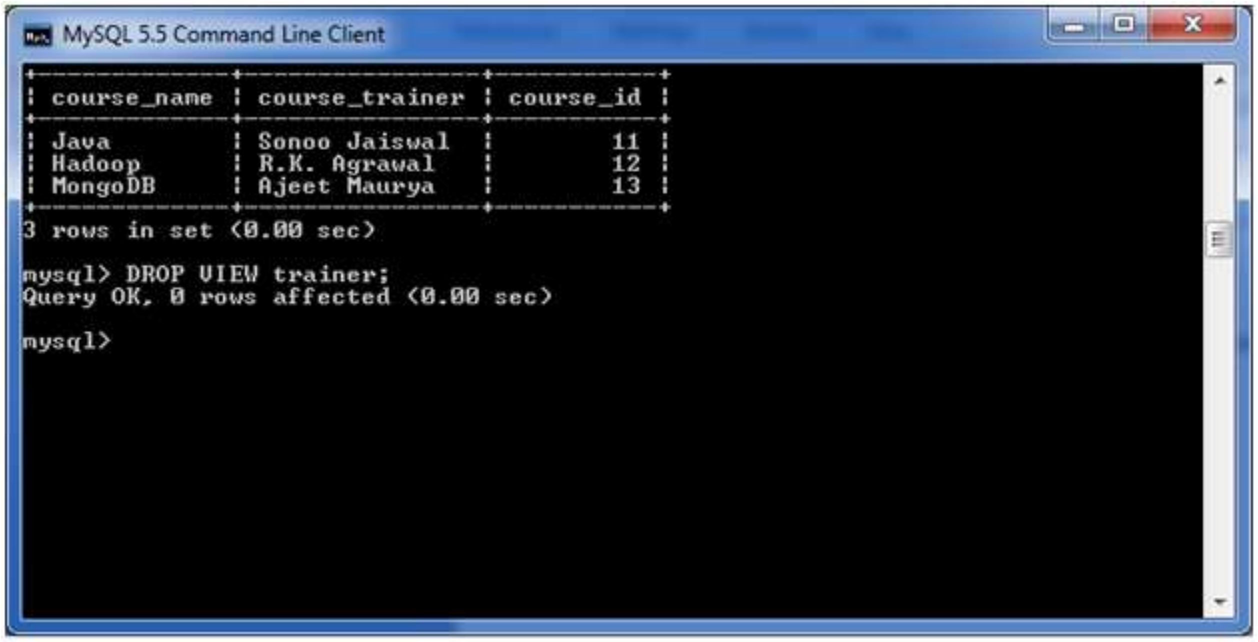
Parameters:

view_name: It specifies the name of the VIEW that you want to drop.

IF EXISTS: It is optional. If you do not specify this clause and the VIEW doesn't exist, the DROP VIEW statement will return an error.

Example:

1. **DROP VIEW** trainer;



```
+-------------+----------------+-----------+
| course_name | course_trainer | course_id |
+-------------+----------------+-----------+
| Java        | Sonoo Jaiswal  |        11 |
| Hadoop      | R.K. Agrawal   |        12 |
| MongoDB     | Ajeet Maurya   |        13 |
+-------------+----------------+-----------+
3 rows in set (0.00 sec)

mysql> DROP VIEW trainer;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

======================11 dec 2020

summary

MySQL Queries

A list of commonly used MySQL queries to create database, use database, create table, insert record, update record, delete record, select record, truncate table and drop table are given below.

1) MySQL Create Database

MySQL create database is used to create database. For example

1. **create database** db1;

2) MySQL Select/Use Database

MySQL use database is used to select database. For example

1. use db1;

3) MySQL Create Query

MySQL create query is used to create a table, view, procedure and function. For example:

1. **CREATE TABLE** customers (id **int**(10),**name varchar**(50),city **varchar**(50), **PRIMARY KEY** (id));

4) MySQL Alter Query

MySQL alter query is used to add, modify, delete or drop colums of a table. Let's see a query to add column in customers table:

1. **ALTER TABLE** customers **ADD** age **varchar**(50);

5) MySQL Insert Query

MySQL insert query is used to insert records into table. For example:

1. **insert into** customers **values**(101,'rahul','delhi');

6) MySQL Update Query

MySQL update query is used to update records of a table. For example:

1. **update** customers **set name**='bob', city='london' **where** id=101;

7) MySQL Delete Query

MySQL update query is used to delete records of a table from database. For example:

1. **delete from** customers **where** id=101;

8) MySQL Select Query

Oracle select query is used to fetch records from database. For example:

1. **SELECT** * **from** customers;

9) MySQL Truncate Table Query

MySQL update query is used to truncate or remove records of a table. It doesn't remove structure. For example:

1. **truncate table** customers;

10) MySQL Drop Query

MySQL drop query is used to drop a table, view or database. It removes structure and data of a table if you drop table. For example:

1. **drop table** customers;

From first

MySQL Data Types

Data Type Syntax	Description

A Data Type specifies a particular type of data, like integer, floating points, Boolean etc. It also identifies the possible values for that type, the operations that can be performed on that type and the way the values of that type are stored.

MySQL supports a lot number of SQL standard data types in various categories. It uses many different data types broken into mainly three categories: numeric, date and time, and string types.

INT	A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.
TINYINT	A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.
SMALLINT	A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.
MEDIUMINT	A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.
BIGINT	A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 20 digits.
FLOAT(m,d)	A floating-point number that cannot be unsigned. You can define the display length (m) and the number of decimals (d). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a float.
DOUBLE(m,d)	A double precision floating-point number that cannot be unsigned. You can define the display length (m) and the number of decimals (d). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a double. Real is a synonym for double.
DECIMAL(m,d)	An unpacked floating-point number that cannot be unsigned. In unpacked decimals, each decimal corresponds to one byte. Defining the display length (m) and the number of decimals (d) is required. Numeric is a synonym for decimal.

Numeric Data Type

Date and Time Data Type:

Data Type Syntax	Maximum Size	Explanation
DATE	Values range from '1000-01-01' to '9999-12-31'.	Displayed as 'y
DATETIME	Values range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.	Displayed as 'y hh:mm:ss'.
TIMESTAMP(m)	Values range from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' TC.	Displayed as 'Y HH:MM:SS'.
TIME	Values range from '-838:59:59' to '838:59:59'.	Displayed as 'H
YEAR[(2|4)]	Year value as 2 digits or 4 digits.	Default is 4 dig

String Data Types:

Data Type Syntax	Maximum Size	Explanation
CHAR(size)	Maximum size of 255 characters.	Where size is the number of characte length strings. Space padded on righ characters.
VARCHAR(size)	Maximum size of 255 characters.	Where size is the number of characte Variable-length string.
TINYTEXT(size)	Maximum size of 255	Where size is the number of characte

	characters.	
TEXT(size)	Maximum size of 65,535 characters.	Where size is the number of characte
MEDIUMTEXT(size)	Maximum size of 16,777,215 characters.	Where size is the number of characte
LONGTEXT(size)	Maximum size of 4GB or 4,294,967,295 characters.	Where size is the number of characte
BINARY(size)	Maximum size of 255 characters.	Where size is the number of binary c Fixed-length strings. Space padded c size characters. (introduced in MySQL 4.1.2)
VARBINARY(size)	Maximum size of 255 characters.	Where size is the number of characte Variable-length string. (introduced in MySQL 4.1.2)

Large Object Data Types (LOB) Data Types:

Data Type Syntax	Maximum Size
TINYBLOB	Maximum size of 255 bytes.
BLOB(size)	Maximum size of 65,535 bytes.
MEDIUMBLOB	Maximum size of 16,777,215 bytes.

LONGTEXT	Maximum size of 4gb or 4,294,967,295 characters.

MySQL Clauses

MySQL WHERE Clause

MySQL WHERE Clause is used with SELECT, INSERT, UPDATE and DELETE clause to filter the results. It specifies a specific position where you have to do the operation.

Syntax:

1. **WHERE** conditions;

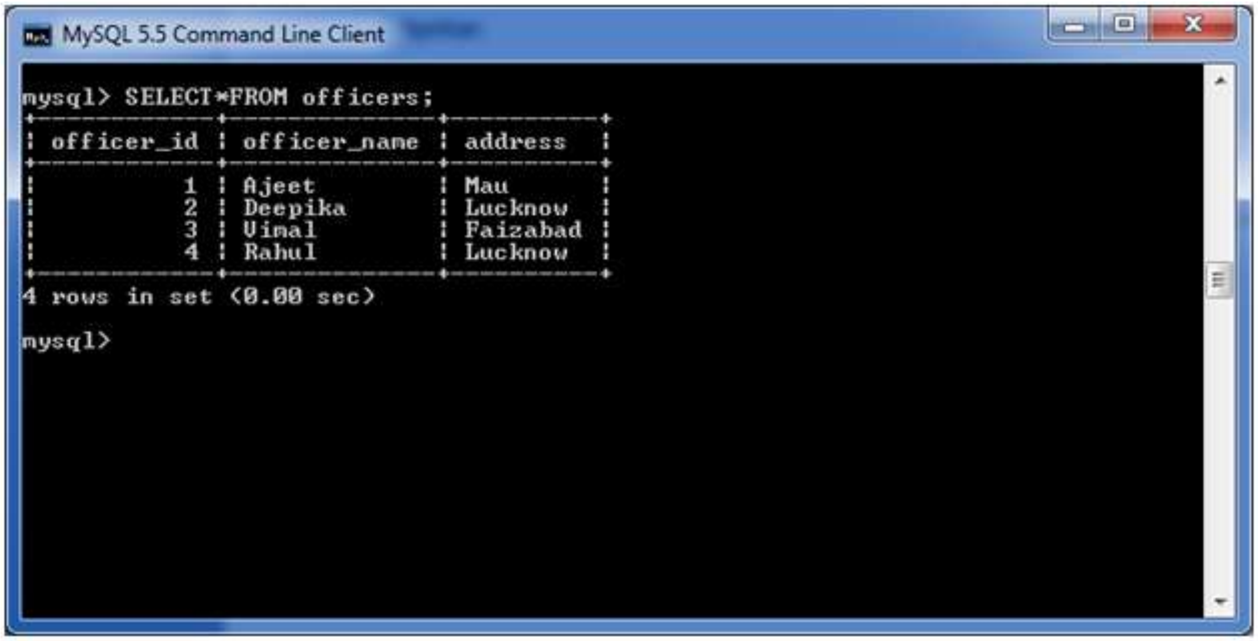
Parameter:

conditions: It specifies the conditions that must be fulfilled for records to be selected.

MySQL WHERE Clause with single condition

Let's take an example to retrieve data from a table "officers".

Table structure:

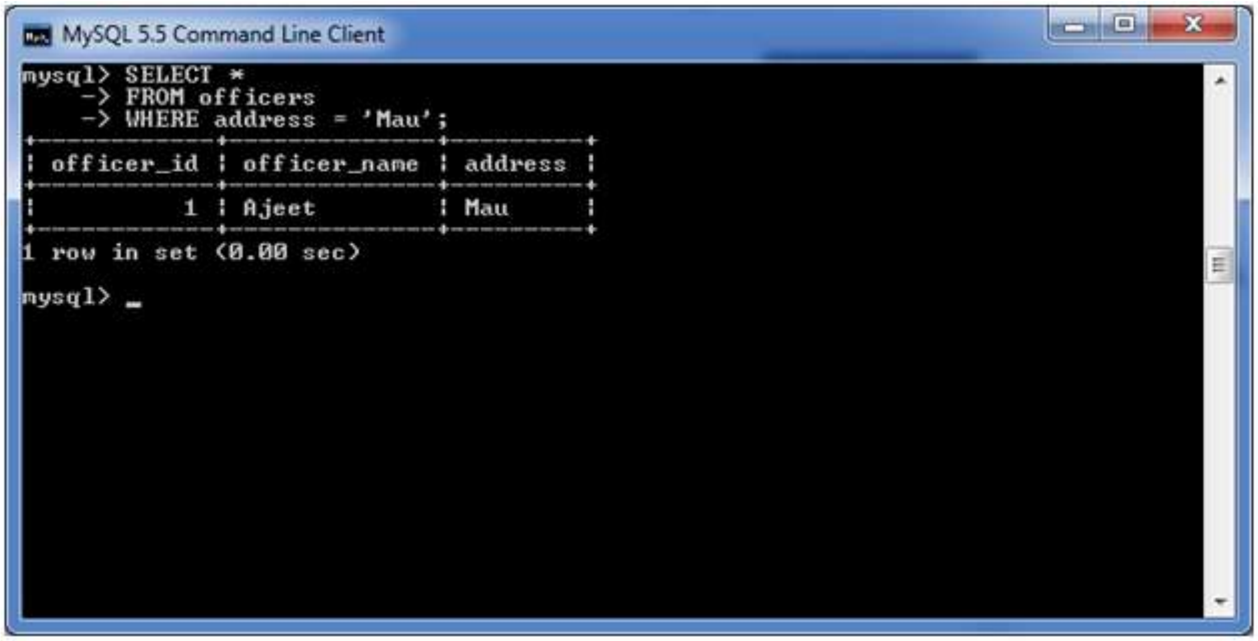


```
mysql> SELECT*FROM officers;
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          1 | Ajeet        | Mau      |
|          2 | Deepika      | Lucknow  |
|          3 | Vimal        | Faizabad |
|          4 | Rahul        | Lucknow  |
+------------+--------------+----------+
4 rows in set (0.00 sec)

mysql>
```

Execute this query:

1. **SELECT** * **FROM** officers **WHERE** address = 'Mau';

Output:



```
mysql> SELECT *
    -> FROM officers
    -> WHERE address = 'Mau';
+------------+--------------+---------+
| officer_id | officer_name | address |
+------------+--------------+---------+
|          1 | Ajeet        | Mau     |
+------------+--------------+---------+
1 row in set (0.00 sec)

mysql> _
```

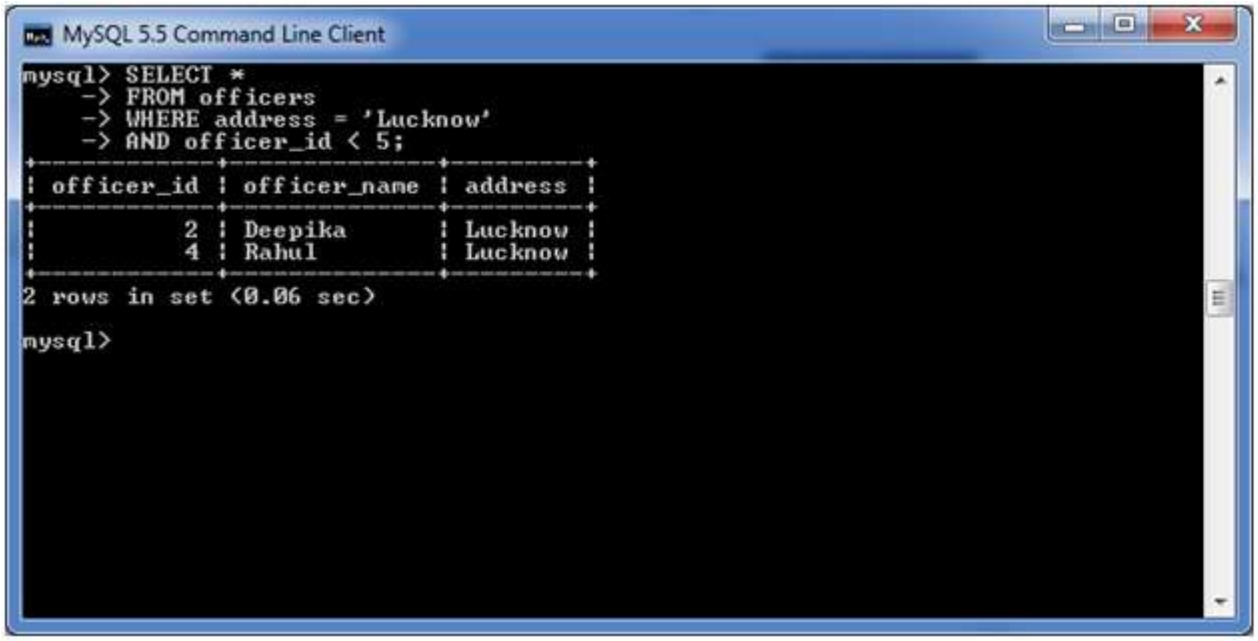
MySQL WHERE Clause with AND condition

In this example, we are retrieving data from the table "officers" with AND condition.

Execute the following query:

```
1. SELECT *  FROM officers  WHERE address = 'Lucknow'  AND officer_id < 5;
```

Output:



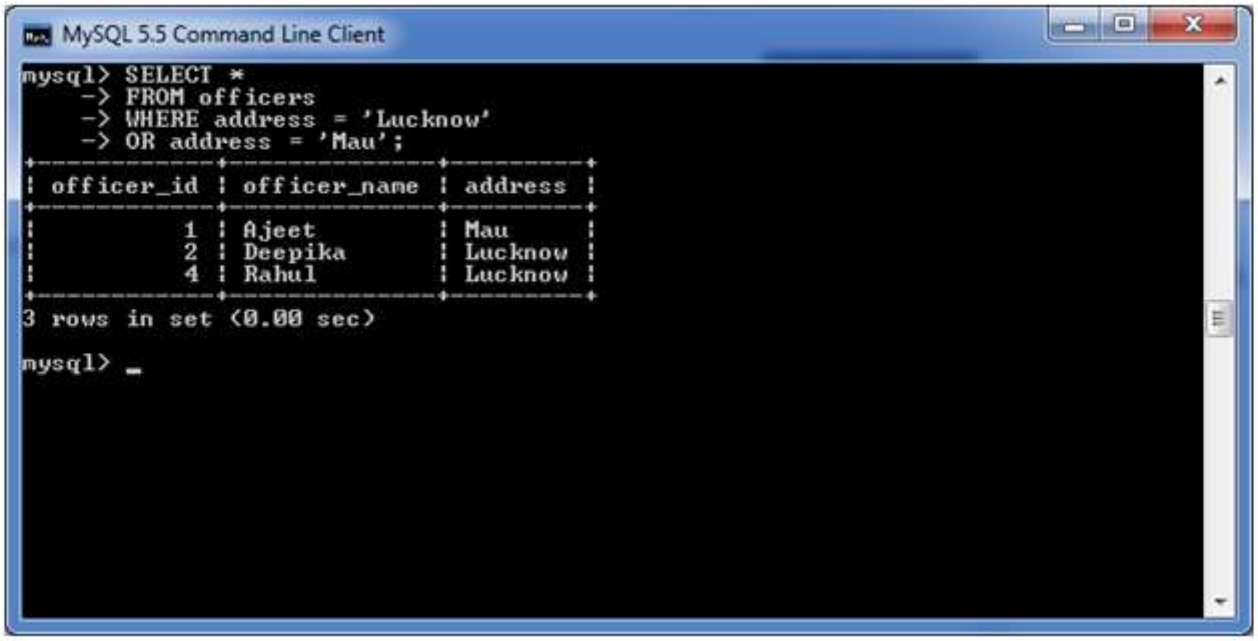
```
mysql> SELECT *
    -> FROM officers
    -> WHERE address = 'Lucknow'
    -> AND officer_id < 5;
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          2 | Deepika      | Lucknow  |
|          4 | Rahul        | Lucknow  |
+------------+--------------+----------+
2 rows in set (0.06 sec)

mysql>
```

WHERE Clause with OR condition

Execute the following query:

```
1. SELECT *  FROM officers  WHERE address = 'Lucknow'  OR address = 'Mau';
```

Output:



```
mysql> SELECT *
    -> FROM officers
    -> WHERE address = 'Lucknow'
    -> OR address = 'Mau';
+------------+--------------+---------+
| officer_id | officer_name | address |
+------------+--------------+---------+
|          1 | Ajeet        | Mau     |
|          2 | Deepika      | Lucknow |
|          4 | Rahul        | Lucknow |
+------------+--------------+---------+
3 rows in set (0.00 sec)

mysql> _
```

MySQL WHERE Clause with combination of AND & OR conditions

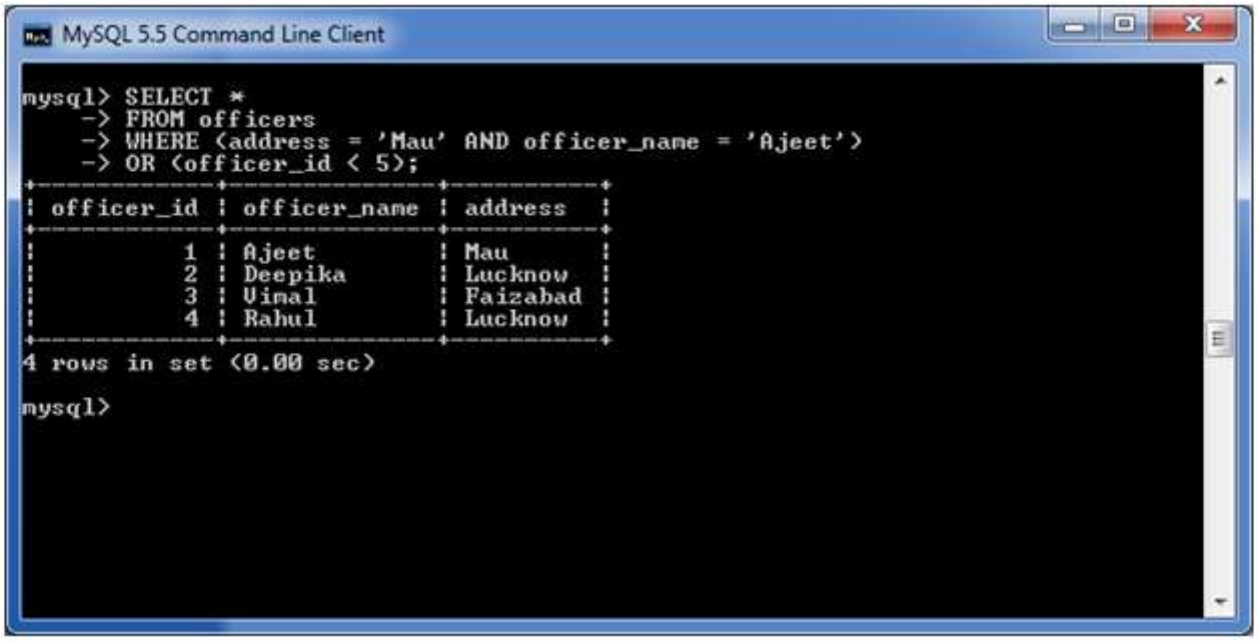
use the AND & OR conditions altogether with the WHERE clause.

See this example:

Execute the following query:

1. **SELECT** * **FROM** officers **WHERE** (address = 'Mau' AND officer_name = 'Ajeet') OR (officer_id < 5);

Output:



```
mysql> SELECT *
    -> FROM officers
    -> WHERE (address = 'Mau' AND officer_name = 'Ajeet')
    -> OR (officer_id < 5);
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          1 | Ajeet        | Mau      |
|          2 | Deepika      | Lucknow  |
|          3 | Vimal        | Faizabad |
|          4 | Rahul        | Lucknow  |
+------------+--------------+----------+
4 rows in set (0.00 sec)

mysql>
```

Officer_ID	**OFF_name**	**bg**	**gender**	**Place**
1001	Adamya	O positive	M	Haryana
1002	Vaishnavi	B positive	F	Rajastan
1003	Dhurv	A positive	M	Gwalior
1004	Aayush	B negative	M	Delhi
1005	rahul	AB positive	M	mumbai
1006	lavish	O positive	M	Haryana
1007	Boopathy	B positive	F	Rajastan
1008	Dhurv	A positive	M	Gwalior
1009	Aayush tiwari	B negative	M	Delhi
10010	rahul	AB positive	M	mumbai

Create the table with the above mentioned data

1) Add one more field called "pincode"
2) Update lavish place to "prayagraj"
3) Use set and change any field data
4) Change Rahul name as rakesh city as Haryana where id = 10010
5) Rahul whose id is 1005 remove from record
6) Use logical operations and or not and show the results
7) Show all the data where gender is Male and place is delhi
8) Show all the data where gender is Male and place is delhi or id is 1005 or 1004
9) Change aayush as nitish and location as delhi
10) Add value as sourav reg no .10244,location –rewa.bg A positive,

Adding extra !!! – Revision of 3rd unit - summary

Features of dbms

- It helps users to access data in the RDBMS system.
- It helps you to describe the data.
- It allows you to define the data in a database and manipulate that specific data.
- With the help of SQL, you can create and drop databases and tables.
- SQL offers you to use the function in a database, create a view, and stored procedure.
- You can set permissions on tables, procedures, and views.

Brief History of SQL

Here, are important landmarks from the history of SQL:

- 1970 - Dr. Edgar F. "Ted" Codd described a relational model for databases.
- 1974 - Structured Query Language appeared.
- 1978 - IBM released a product called System/R.
- 1986 - IBM developed the prototype of a relational database, which is standardized by ANSI.
- 1989- First ever version launched of SQL
- 1999 - SQL 3 launched with features like triggers, object-orientation, etc.
- SQL2003- window functions, XML-related features, etc.
- SQL2006- Support for XML Query Language
- SQL2011-improved support for temporal databases

Types of SQL

Here are five types of widely used SQL queries.

1. Data Definition Language (DDL)
2. Data Manipulation Language (DML)
3. Data Control Language(DCL)
4. Transaction Control Language(TCL)
5. Data Query Language (DQL)

What is DDL?

Data *Definition Language* helps you to define the database structure or schema.

Five types of DDL commands are:

CREATE

CREATE statements is used to define the database structure schema:

Syntax:

```
CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);
```

For example:

```
Create database university;
Create table students;
Create view for_students;
```

DROP

Drops commands *remove tables and databases* from RDBMS.

Syntax

```
DROP TABLE ;
```

For example:

```
Drop object_type object_name;
Drop database university;
Drop table student;
```

ALTER

Alters command allows you to alter the structure of the database.

Syntax:

To add a new column in the table

```
ALTER TABLE table_name ADD column_name COLUMN-definition;
```

To modify an existing column in the table:

```
ALTER TABLE MODIFY(COLUMN DEFINITION....);
```

For example:

```
Alter table tablename add subject varchar;
```

TRUNCATE:

This command used to delete all the rows from the table and free the space containing the table.

Syntax:

```
TRUNCATE TABLE table_name;
```

Example:

```
TRUNCATE table students;
```

DDL : =summary of stmts (CDAT)short form to remember the stmts.

1. **Create**
2. **Drop**
3. **Alter**
4. **Truncate**

What is Data Manipulation Language?

Data Manipulation Language (DML) allows you to modify the database instance by inserting, modifying, and deleting its data. It is responsible for performing all types of data modification in a database.

There are three basic constructs which allow database program and user to enter data and information are:

DML commands: (IDU)

1. INSERT
2. DELETE
3. UPDATE

INSERT:

This is a statement is a SQL query. This command is used to insert data into the row of a table.

Syntax:

```
INSERT INTO TABLE_NAME  (col1, col2, col3,.... col N)
VALUES (value1, value2, value3, .... valueN);
Or
INSERT INTO TABLE_NAME
VALUES (value1, value2, value3, .... valueN);
```

For example:

```
INSERT INTO students (RollNo, FIrstName, LastName) VALUES ('60', 'Tom', Erichsen');
```

UPDATE:

This command is used to update or *modify the value of a column in the table*.

Syntax:

```
UPDATE table_name SET [column_name1= value1,...column_nameN = valueN] [WHERE CONDITION]
```

For example:

```
UPDATE students
SET FirstName = 'Jhon', LastName= 'son'
WHERE StudID = 3;
```

DELETE:

This command is used *to remove one or more rows from a table*.

Syntax:

```
DELETE FROM table_name [WHERE condition];
```

For example:

```
DELETE FROM students
WHERE FirstName = 'Jhon';
```

What is DCL?

DCL (Data Control Language) includes commands like GRANT and REVOKE, which are useful to give "*rights & permissions."* Other permission controls parameters of the database system.

DCL commands: (GR)

Commands that come under DCL:

- Grant
- Revoke

Grant:

This command is *use to give user access privileges to a database*.

Syntax:

```
GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;
```

For example:

```
GRANT SELECT ON Users TO'rahul'@'localhost;
```

Revoke:

It is useful to *back permissions from the user.*

Syntax:

```
REVOKE privilege_nameON object_nameFROM {user_name |PUBLIC |role_name}
```

For example:

```
REVOKE SELECT, UPDATE ON student FROM BCA, MCA,Btech,mtech,phd;
```

What is TCL?

Transaction control language or TCL commands deal with the transaction within the database.

TCL –(CRS)

1. **Commit**
2. **Rollback**
3. **SAVEPOINT**

Commit

This command is used to *save all the transactions to the database.*

Syntax:

```
Commit;
```

For example:

```
DELETE FROM Students WHERE RollNo =25;
COMMIT;
```

Rollback

Rollback command allows you to *undo transactions that have not already been saved to the database.*

Syntax:

```
ROLLBACK;
```

Example:

```
DELETE FROM Students  WHERE RollNo =25;   rollback;
```

SAVEPOINT

This command helps you *to sets a savepoint within a transaction*.

Syntax:

```
SAVEPOINT SAVEPOINT_NAME;
```

Example:

```
SAVEPOINT RollNo;
```

What is DQL?

Data Query Language (DQL) is used to fetch the data from the database. It uses only one command:

SELECT:

This command helps you to select the attribute based on the condition described by the *WHERE clause.*

Syntax:

```
SELECT expressions FROM TABLES  WHERE conditions;
```

For example:

```
SELECT FirstName  FROM Student  WHERE RollNo > 15;
```

Summary:

- SQL is a database language designed for the retrieval and management of data in a relational database.
- It helps users to access data in the RDBMS system
- In the year 1974, the term Structured Query Language appeared
- Five types of SQL queries are
- 1) Data Definition Language (DDL)
- 2) Data Manipulation Language (DML)
- 3) Data Control Language(DCL)
- 4) Transaction Control Language(TCL) and,
- 5) Data Query Language (DQL)
- Data Definition Language(DDL) helps you to define the database structure or schema.
- Data Manipulation Language (DML) allows you to modify the database instance by inserting, modifying, and deleting its data.
- DCL (Data Control Language) includes commands like GRANT and REVOKE, which are useful to give "rights & permissions."
- Transaction control language or TCL commands deal with the transaction within the database.
- Data Query Language (DQL) is used to fetch the data from the database.

What is Join in DBMS?

Join in DBMS is a binary operation which allows you to combine join product and selection in one single statement. The goal of creating a join condition is that it helps you to combine the data from two or more DBMS tables. The tables in DBMS are associated using the primary key and foreign keys.

Types of Join

There are mainly two types of joins in DBMS:

1. Inner Joins: Theta, Natural, EQUI

2. Outer Join: Left, Right, Full

Inner Join

INNER JOIN is used to *return rows from both tables* which satisfy the given condition. It is the most widely used join operation and can be considered as a default join-type

An Inner join or equijoin is a comparator-based join which uses equality comparisons in the join-predicate. However, if you use other comparison operators like ">" it can't be called equijoin.

Inner Join further divided into three subtypes:

- Theta join
- Natural join
- EQUI join

Theta Join

THETA JOIN allows you to *merge two tables based on the condition* represented by theta. *Theta joins work for all comparison operators*. It is denoted by symbol **θ**. The general case of JOIN operation is called a Theta join.

Syntax:

```
A ⋈θ B
```

Theta join can use any conditions in the selection criteria.

Consider the following tables.

Table A		**Table B**	
column 1	**column 2**	**column 1**	**column 2**
1	1	1	1

1	2	1	3

For example:

```
A ⋈ A.column 2 >  B.column 2 (B)
```

A ⋈ A.column 2 > B.column 2 (B)	
column 1	**column 2**
1	2

EQUI Join

EQUI JOIN is done when a Theta join uses only the equivalence condition. EQUI join is the most difficult operation to implement efficiently in an RDBMS, and one reason why RDBMS have essential performance problems.

For example:

```
A ⋈ A.column 2 =  B.column 2 (B)
```

A ⋈ A.column 2 = B.column 2 (B)	
column 1	**column 2**
1	1

Natural Join (⋈)

NATURAL JOIN does not utilize any of the comparison operators. In this type of join, the attributes should have the same name and domain. In Natural Join, there should be at least one common attribute between two relations.

It performs selection forming equality on those attributes which appear in both relations and eliminates the duplicate attributes.

Example:

Consider the following two tables

C	
Num	**Square**
2	4
3	9

D	
Num	**Cube**
2	8
3	18

```
C ⋈ D
```

C ⋈ D		
Num	**Square**	**Cube**
2	4	8
3	9	18

Outer Join

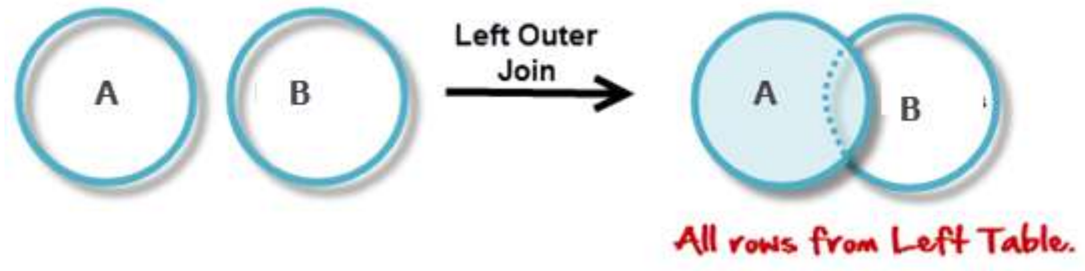
An **OUTER JOIN** doesn't require each record in the two join tables to have a matching record. In this type of join, the table retains each record even if no other matching record exists.

Three types of Outer Joins are:

- Left Outer Join
- Right Outer Join
- Full Outer Join

Left Outer Join (A ⟕ B)

LEFT JOIN returns all the rows from the table on the left even if no matching rows have been found in the table on the right. When no matching record found in the table on the right, NULL is returned.



Consider the following 2 Tables

A	
Num	**Square**
2	4
3	9

4	16

B	
Num	**Cube**
2	8
3	18
5	75

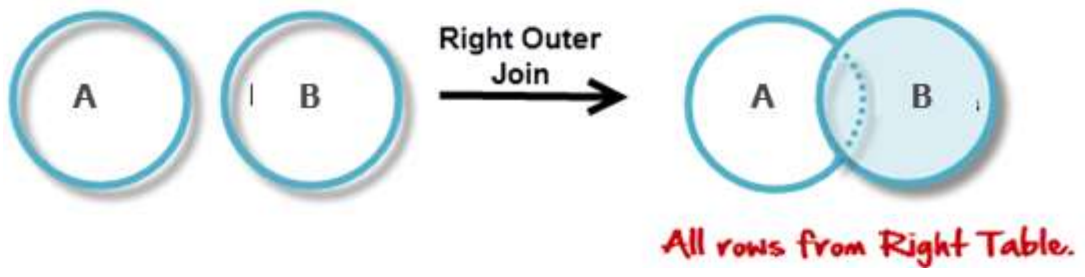
A ⟕ B

A ⋈ B		
Num	**Square**	**Cube**
2	4	8
3	9	18
4	16	-

Right Outer Join (A ⟖ B)

RIGHT JOIN returns all the columns from the table on the right even if no matching rows have been found in the table on the left. Where no matches have been found in the table on the left, NULL is returned. RIGHT outer JOIN is the opposite of LEFT JOIN

In our example, let's assume that you need to get the names of members and movies rented by them. Now we have a new member who has not rented any movie yet.



A ⟖ B

A ⋈ B		
Num	**Cube**	**Square**
2	8	4
3	18	9
5	75	-

Full Outer Join (A ⟗ B)

In a **FULL OUTER JOIN** , all tuples from both relations are included in the result, irrespective of the matching condition.

Example:

A ⟗ B

A ⋈ B		
Num	**Square**	**Cube**

2	4	8
3	9	18
4	16	-
5	-	75

Summary:

- There are mainly two types of joins in DBMS 1) Inner Join 2) Outer Join
- An inner join is the widely used join operation and can be considered as a default join-type.
- Inner Join is further divided into three subtypes: 1) Theta join 2) Natural join 3) EQUI join
- Theta Join allows you to merge two tables based on the condition represented by theta
- When a theta join uses only equivalence condition, it becomes an equi join.
- Natural join does not utilize any of the comparison operators.
- An outer join doesn't require each record in the two join tables to have a matching record.
- Outer Join is further divided into three subtypes are: 1)Left Outer Join 2) Right Outer Join 3) Full Outer Join
- The LEFT Outer Join returns all the rows from the table on the left, even if no matching rows have been found in the table on the right.
- The RIGHT Outer Join returns all the columns from the table on the right, even if no matching rows have been found in the table on the left.
- In a full outer join, all tuples from both relations are included in the result, irrespective of the matching condition.

What is Indexing?

Indexing is a data structure technique *which allows you to quickly retrieve records from a database file*. An Index is a small table having only two columns. The first column comprises a copy of the primary or candidate key of a table. Its second column contains a set of pointers for holding the address of the disk block where that specific key value stored.

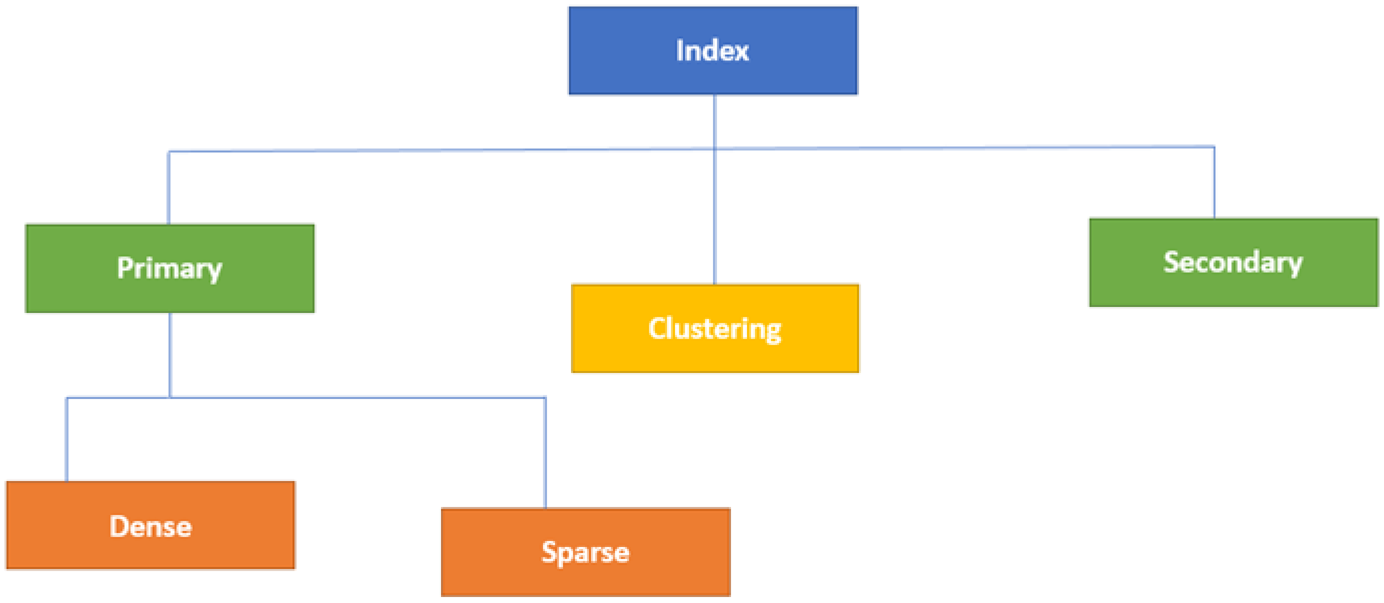
An index -

- Takes a search key as input
- Efficiently returns a collection of matching records.

  Types of Indexing

- Primary Index
- Secondary Index
- Clustering Index
- What is Multilevel Index?
- B-Tree Index
- Advantages of Indexing
- Disadvantages of Indexing

Types of Indexing



Type of Indexes in Database

Indexing in Database is defined based on its indexing attributes. Two main types of indexing methods are:

- Primary Indexing
- Secondary Indexing

Primary Index

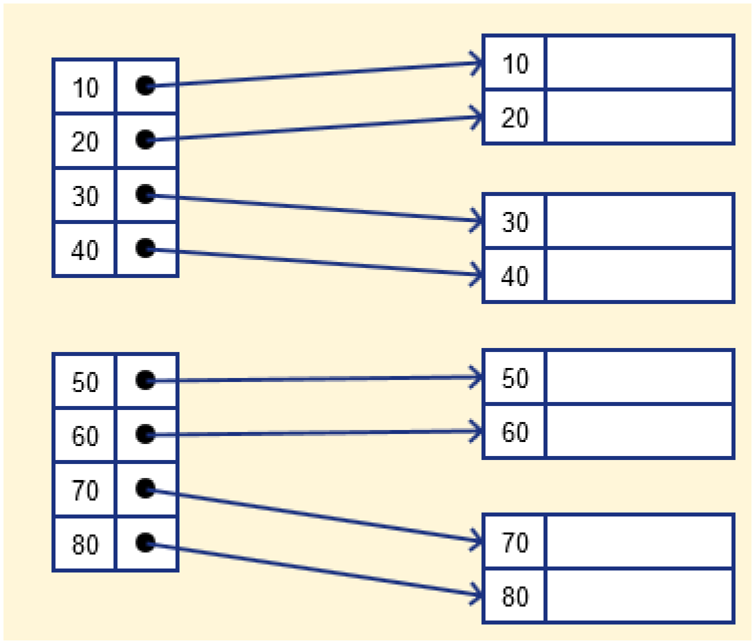
Primary Index is an *ordered file* which is ***fixed length size with two fields***. The first field is the same a primary key and second, filed is pointed to that specific data block. In the primary Index, there is *always one to one relationship* between the entries in the index table.

The primary Indexing in DBMS is also further divided into two types.

- Dense Index
- Sparse Index

Dense Index

In a dense index, a record is created for every search key valued in the database. This helps you to search faster but needs more space to store index records. In this Indexing, method records contain search key value and points to the real record on the disk.

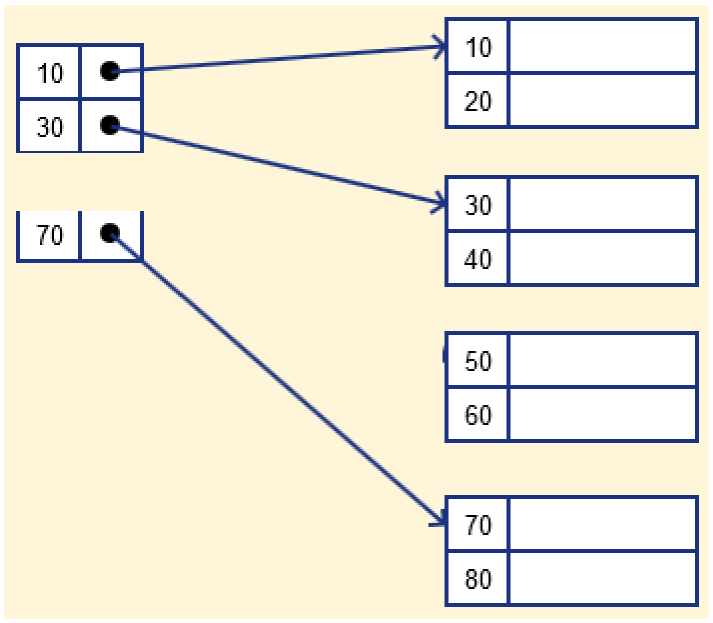


Sparse Index

It is an index record that *appears for only some of the values in the file*. Sparse Index helps you to resolve the issues of dense Indexing in DBMS. In this method of indexing technique, a range of index columns stores the same data block address, and when data needs to be retrieved, the block address will be fetched.

However, sparse Index stores index records for only some search-key values. It needs less space, less maintenance overhead for insertion, and deletions but It is slower compared to the dense Index for locating records.

Below is an database index Example of Sparse Index



Secondary Index

The secondary Index in DBMS can be *generated by a field which has a unique value for each record, and it should be a candidate key. It is also known as a non-clustering index.*

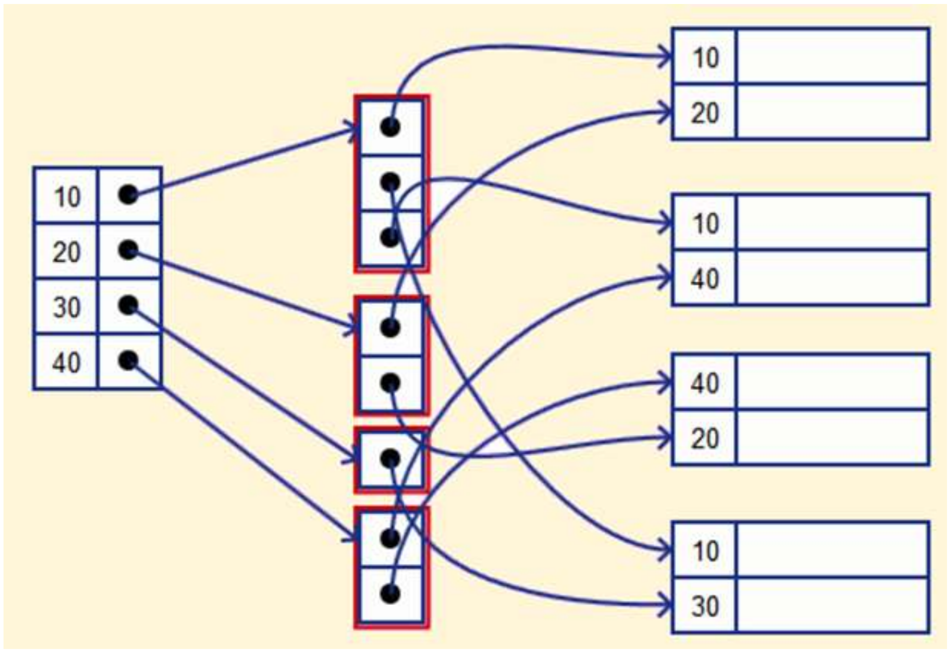
This two-level database indexing technique is used to reduce the mapping size of the first level. For the first level, a large range of numbers is selected because of this; the mapping size always remains small.

Example of secondary Indexing

Let's understand secondary indexing with a database index example:

In a bank account database, data is stored sequentially by acc_no; you may want to find all accounts in of a specific branch of ABC bank.

Here, you can have a secondary index in DBMS for every search-key. Index record is a record point to a bucket that contains pointers to all the records with their specific search-key value.



Clustering Index

In a clustered index, records *themselves are stored in the Index and not pointers*. Sometimes the Index is created on non-primary key columns which might not be unique for each record. In such a situation, you can group two or more columns to get the unique values and create an index which is called clustered Index. This also helps you to identify the record faster.

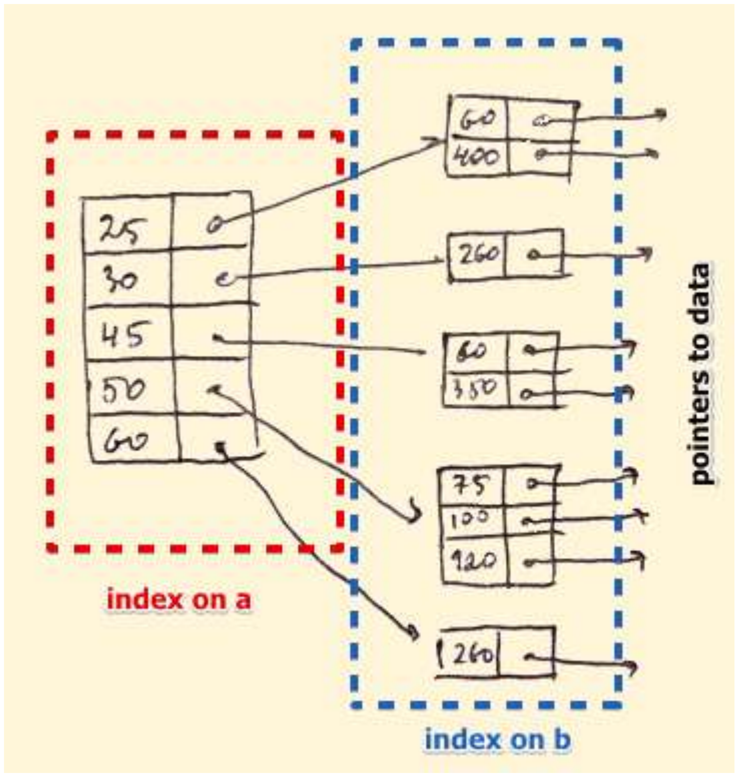
Example:

Let's assume that a company recruited many employees in various departments. In this case, clustering indexing in DBMS should be created for all employees who belong to the same dept.

It is considered in a single cluster, and index points point to the cluster as a whole. Here, Department _no is a non-unique key.

What is Multilevel Index?

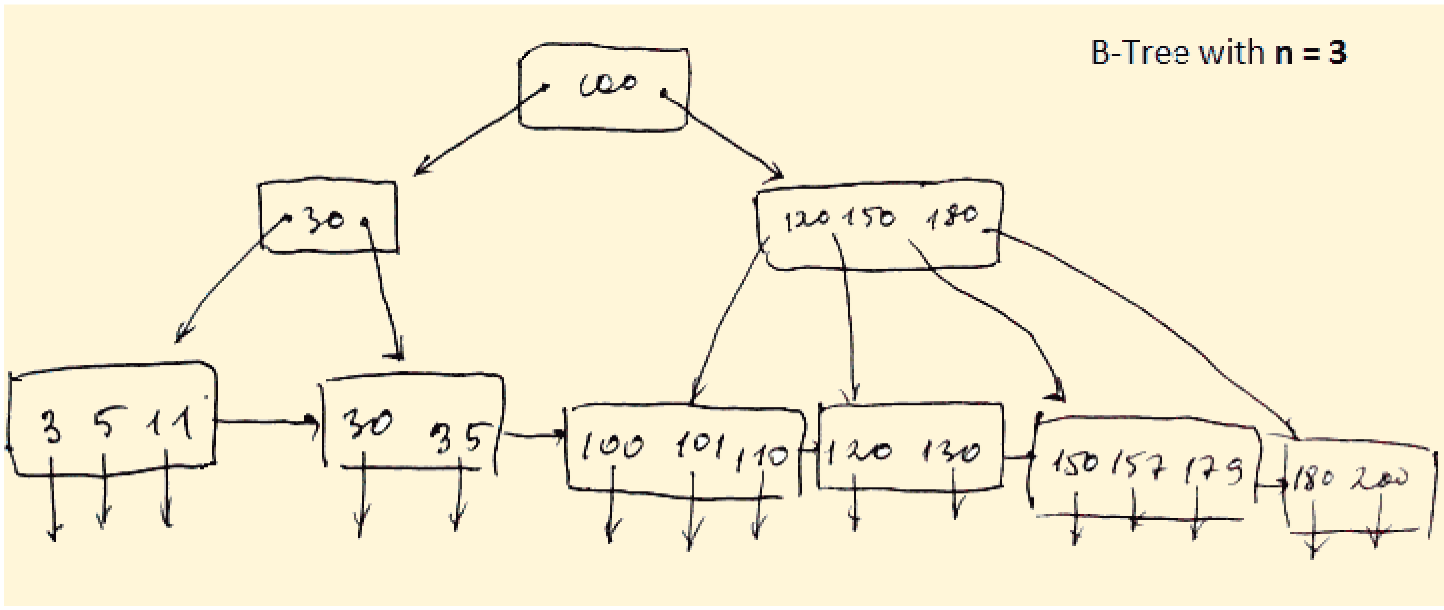
Multilevel Indexing in Database is created when a primary index does not fit in memory. In this type of indexing method, you can reduce the number of disk accesses to short any record and kept on a disk as a sequential file and create a sparse base on that file.



B-Tree Index

B-tree index is the widely used data structures for tree based indexing in DBMS. It is a multilevel format of tree based indexing in DBMS technique which has balanced binary search trees. All leaf nodes of the B tree signify actual data pointers.

Moreover, all leaf nodes are interlinked with a link list, which allows a B tree to support both random and sequential access.



- Lead nodes must have between 2 and 4 values.
- Every path from the root to leaf are mostly on an equal length.
- Non-leaf nodes apart from the root node have between 3 and 5 children nodes.
- Every node which is not a root or a leaf has between n/2] and n children.

Advantages of Indexing

Important pros/ advantage of Indexing are:

- It helps you to *reduce the total number of I/O operations* needed to retrieve that data, so you don't need to access a row in the database from an index structure.
- *Offers Faster search and retrieval of data to users.*
- Indexing also helps you to reduce tablespace as you don't need to link to a row in a table, as there is no need to store the ROWID in the Index. Thus you will able to reduce the tablespace.
- You can't sort data in the lead nodes as the value of the primary key classifies it.

Disadvantages of Indexing

Important drawbacks/cons of Indexing are:

- To perform the indexing database management system, you need a primary key on the table with a unique value.
- You can't perform any other indexes in Database on the Indexed data.
- You are not allowed to partition an index-organized table.
- SQL Indexing Decrease performance in INSERT, DELETE, and UPDATE query.

Summary:

- Indexing is a small table which is consist of two columns.
- Two main types of indexing methods are 1)Primary Indexing 2) Secondary Indexing.
- Primary Index is an ordered file which is fixed length size with two fields.
- The primary Indexing is also further divided into two types 1)Dense Index 2)Sparse Index.
- In a dense index, a record is created for every search key valued in the database.
- A sparse indexing method helps you to resolve the issues of dense Indexing.
- The secondary Index in DBMS is an indexing method whose search key specifies an order different from the sequential order of the file.
- Clustering index is defined as an order data file.
- Multilevel Indexing is created when a primary index does not fit in memory.
- The biggest benefit of Indexing is that it helps you to reduce the total number of I/O operations needed to retrieve that data.
- The biggest drawback to performing the indexing database management system, you need a primary key on the table with a unique value.

•

What is DBMS?

A DBMS is a software used to store and manage data. The DBMS was introduced during 1960's to store any data. It also offers manipulation of the data like insertion, deletion, and updating of the data.

DBMS system also performs the functions like defining, creating, revising and controlling the database. It is specially designed to create and maintain data and enable the individual business application to extract the desired data.

What is RDBMS?

Relational Database Management System (RDBMS) is an advanced version of a DBMS system. It came into existence during 1970's. RDBMS system also allows the organization to access data more efficiently then DBMS.

RDBMS is a software system which is used to store only data which need to be stored in the form of tables. In this kind of system, data is managed and stored in rows and columns which is known as tuples and attributes. RDBMS is a powerful data management system and is widely used across the world.

KEY DIFFERENCE

- DBMS stores data as a file whereas in RDBMS, data is stored in the form of tables.
- DBMS supports single users, while RDBMS supports multiple users.
- DBMS does not support client-server architecture but RDBMS supports client-server architecture.
- DBMS has low software and hardware requirements whereas RDBMS has higher hardware and software requirements.
- In DBMS, data redundancy is common while in RDBMS, keys and indexes do not allow data redundancy.

Difference between DBMS vs RDBMS

Parameter	DBMS	RDBMS
Storage	DBMS stores data as a file.	Data is stored in the form of

Parameter	DBMS	RDBMS
		tables.
Database structure	DBMS system, stores data in either a navigational or hierarchical form.	RDBMS uses a tabular structure where the headers are the column names, and the rows contain corresponding values
Number of Users	DBMS supports single user only.	It supports multiple users.
ACID	In a regular database, the data may not be stored following the ACID model. This can develop inconsistencies in the database.	Relational databases are harder to construct, but they are consistent and well structured. They obey ACID (Atomicity, Consistency, Isolation, Durability).
Type of program	It is the program for managing the databases on the computer networks and the system hard disks.	It is the database systems which are used for maintaining the relationships among the tables.
Hardware and software needs.	Low software and hardware needs.	Higher hardware and software need.
Integrity constraints	DBMS does not support the integrity constants. The integrity constants are not imposed at the file level.	RDBMS supports the integrity constraints at the schema level. Values beyond a defined range cannot be stored into the particular RDMS column.
Normalization	DBMS does not support Normalization	RDBMS can be Normalized.
Distributed Databases	DBMS does not support distributed database.	RBMS offers support for distributed databases.
Ideally suited	DBMS system mainly deals	RDMS is designed to handle a

Parameter	DBMS	RDBMS
for	with small quantity of data.	large amount of data.
Dr. E.F. Codd Rules	Dbms satisfy less than seven of Dr. E.F. Codd Rules	Dbms satisfy 8 to 10 Dr. E.F. Codd Rules
Client Server	DBMS does not support client-server architecture	RDBMS supports client-server architecture.
Data Fetching	Data fetching is slower for the complex and large amount of data.	Data fetching is rapid because of its relational approach.
Data Redundancy	Data redundancy is common in this model.	Keys and indexes do not allow Data redundancy.
Data Relationship	No relationship between data	Data is stored in the form of tables which are related to each other with the help of foreign keys.
Security	There is no security.	Multiple levels of security. Log files are created at OS, Command, and object level.
Data Access	Data elements need to access individually.	Data can be easily accessed using SQL query. Multiple data elements can be accessed at the same time.
Examples	Examples of DBMS are a file system, XML, Windows Registry, etc.	Example of RDBMS is MySQL, Oracle, SQL Server, etc.

What are Keys?

Keys are attribute that helps you to identify a row(tuple) in a relation(table). They allow you to find the relationship between two tables. Keys help you uniquely identify a row in a table by a combination of one or more columns in that table. The database key is also helpful for finding a unique record or row from the table.

What is Database Relationship?

The database relationship is associations between one or more tables that are created using join statements. It is used to efficiently retrieve data from the database. There are primarily three types of relationships 1) One-to-One, 2) One-to-many, 3) Many-to-many.

What is Primary Key?

A primary key constrain is a column or group of columns that uniquely identifies every row in the table of the relational database management system. It cannot be a duplicate, meaning the same value should not appear more than once in the table.

A table can have more than one primary key. Primary key can be defined at the column or the table level. If you create a composite primary key, it should be defined at the table level.

What is Foreign Key?

Foreign key is a column that creates a relationship between two tables. The purpose of the Foreign key is to maintain data integrity and allow navigation between two different instances of an entity. It acts as a cross-reference between two tables as it references the primary key of another table. Every relationship in the database should be supported by a foreign key.

KEY DIFFERENCES:

- A primary key constrain is a column that uniquely identifies every row in the table of the relational database management system, while foreign key is a column that creates a relationship between two tables.

- Primary Key never accepts null values whereas foreign key may accept multiple null values.
- You can have only a single primary key in a table while you can have multiple foreign keys in a table.
- The value of the primary key can't be removed from the parent table whereas the value of foreign key value can be removed from the child table.
- No two rows can have any identical values for a primary key on the other hand a foreign key can contain duplicate values.
- There is no limitation in inserting the values into the table column while inserting any value in the foreign key table, ensure that the value is present into a column of a primary key.

Why use Primary Key?

Here are the cons/benefits of using primary key:

- The main aim of the primary key is to identify each and every record in the database table.
- You can use a primary key when you do not allow someone to enter null values.
- If you delete or update records, the action you specified will be undertaken to make sure data integrity.
- Perform restrict operation to rejects delete or update operation for the parent table.
- Data are organized in a sequence of clustered index whenever you physically organize DBMS table.

Why use Foreign Key?

Here are the important reasons of using foreign key:

- Foreign keys help you to migrate entities using a primary key from the parent table.
- A foreign key enables you to link two or more tables together.
- It makes your database data consistent.
- A foreign key can be used to match a column or combination of columns with primary key in a parent table.
- SQL foreign key constraint is used to make sure the referential integrity of the data parent to match values in the child table.

Example of Primary Key

Syntax:

Below is the syntax of Primary Key:

```
CREATE TABLE <Table-Name>
(
Column1 datatype,
Column2 datatype,  PRIMARY KEY (Column-Name)
.
);
```

Here,

- Table_Name is the name of the table you have to create.
- Column_Name is the name of the column having the primary key.

Example:

StudID	Roll No	First Name	Last Name	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

In the above example, we have created a student table with columns like StudID, Roll No, First Name, Last Name, and Email. StudID is chosen as a primary key because it can uniquely identify other rows in the table.

Example of Foreign Key

Syntax:

Below is the syntax of Foreign Key:

```
CREATE TABLE <Table Name>(
column1    datatype,
column2    datatype,
constraint (name of constraint)
FOREIGN KEY [column1, column2...]
REFERENCES [primary key table name] (List of primary key table column) ...);
```

Here,

- The parameter Table Name indicates the name of the table that you are going to create.
- The parameters column1, column2… depicts the columns that need to be added to the table.
- Constraint denotes the name of constraint you are creating.
- References indicate a table with the primary key.

Example:

DeptCode	**DeptName**
001	Science
002	English
005	Computer

Teacher ID	**Fname**	**Lname**
B002	David	Warner
B017	Sara	Joseph
B009	Mike	Brunton

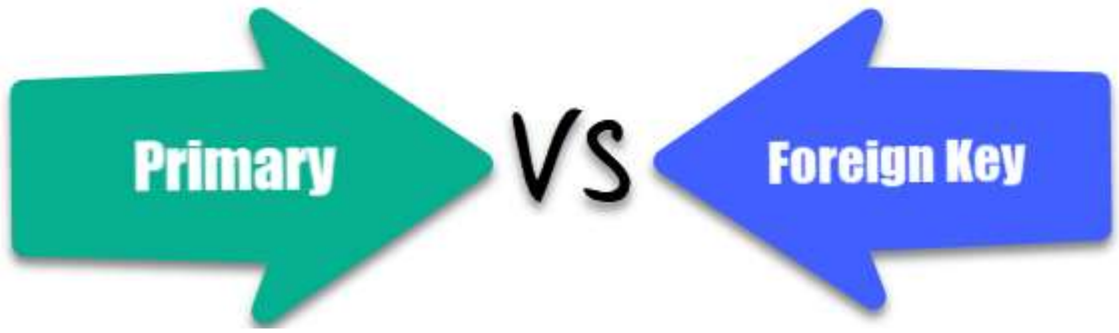
In the above example, we have two tables, a teacher and a department in a school. However, there is no way to see which search works in which department.

In this table, adding the foreign key in Deptcode to the Teacher name, we can create a relationship between the two tables.

Teacher ID	**DeptCode**	**Fname**	**Lname**
B002	002	David	Warner
B017	002	Sara	Joseph
B009	001	Mike	Brunton

This concept is also known as Referential Integrity.

Difference between Primary key and Foreign key



Here is the important difference between Primary key and Foreign key:

Primary Key	Foreign Key
A primary key constrain is a column or group of columns that uniquely identifies every row in the table of the relational database management system.	Foreign key is a column that creates a relationship between two tables.
It helps you to uniquely identify a record in the table.	It is a field in the table that is a primary key of another table.
Primary Key never accepts null values.	A foreign key may accept multiple null values.
The primary key is a clustered index, and data in the DBMS table are physically organized in the sequence of the clustered index.	A foreign key cannot automatically create an index, clustered, or non-clustered.
You can have the single Primary key in a table.	You can have multiple foreign keys in a table.
The value of the primary key can't be removed from the parent table.	The value of foreign key value can be removed from the child table.
You can define the primary key implicitly on	You cannot define foreign keys on

Primary Key	Foreign Key
the temporary tables.	the local or global temporary tables.
Primary key is a clustered index.	By default, it is not a clustered index.
No two rows can have any identical values for a primary key.	A foreign key can contain duplicate values.
There is no limitation in inserting the values into the table column.	While inserting any value in the foreign key table, ensure that the value is present into a column of a primary key.