Introduction to Database

A **database** is an organized collection of data that is stored and accessed electronically. It allows for the efficient storage, retrieval, and management of data. Databases are essential in various applications, from small websites to large enterprise systems.

Here's a breakdown of the key concepts:

---

1. What is a Database?

A database is a system that helps store and manage large amounts of data. Instead of storing data in a flat file (like a spreadsheet or text file), a database uses structured formats, making it much easier to retrieve and manipulate data.

---

2. Types of Databases

There are different types of databases, but the most common are:

- **Relational Databases**: Store data in tables and use SQL (Structured Query Language) to query data. Examples include MySQL, PostgreSQL, and SQL Server.
- **NoSQL Databases**: Designed for handling large amounts of unstructured or semi-structured data. Examples include MongoDB, Cassandra, and Firebase.
- **In-memory Databases**: Data is stored in RAM for fast access. Examples include Redis and Memcached.

---

3. Database Components

- **Tables**: A table is the core component of a relational database. It consists of rows (records) and columns (fields). Each column in a table represents a data type (e.g., name, age, date of birth), and each row represents a single record.

Example:

pgsql

Copy

| ID | Name | Age |

|----|----------|-----|

| 1 | John | 25 |

| 2 | Alice | 30 |

- **Records**: A record (or row) in a table contains data for a specific entity, like a person or product.

- **Fields**: A field (or column) holds specific data about an attribute of an entity, such as a person's name or age.

---

4. Database Management Systems (DBMS)

A **DBMS** is software that manages databases. It provides an interface for users and applications to interact with the database. Some popular DBMS examples are:

- **MySQL**
- **PostgreSQL**
- **SQLite**
- **Oracle**
- **Microsoft SQL Server**

---

5. SQL (Structured Query Language)

SQL is a language used to communicate with relational databases. It helps users to create, read, update, and delete data. Here are the basic SQL operations:

- **SELECT**: Retrieve data from the database.
- **INSERT**: Add new data.
- **UPDATE**: Modify existing data.
- **DELETE**: Remove data.

SQL is essential for querying and interacting with relational databases.

---

6. Relationships Between Tables

In relational databases, tables can be linked to each other through relationships:

- **One-to-One Relationship**: A record in one table corresponds to one record in another table.
- **One-to-Many Relationship**: A record in one table can correspond to many records in another table.
- **Many-to-Many Relationship**: Records in one table can correspond to many records in another table, and vice versa. This is usually represented by a third table called a "junction table."

---

7. Key Concepts in Relational Databases

- **Primary Key**: A unique identifier for a record in a table. No two records can have the same primary key.

- **Foreign Key**: A field in one table that uniquely identifies a row in another table. It establishes the relationship between two tables.
- **Index**: A database object that improves the speed of data retrieval.

---

8. Why Are Databases Important?

- **Efficient Data Retrieval**: Databases are designed to handle large volumes of data and allow users to quickly retrieve, filter, and manipulate data.
- **Data Integrity**: A well-designed database ensures that data remains accurate and consistent.
- **Data Security**: Databases offer features like encryption, user access control, and backups to ensure data security.
- **Scalability**: Databases can grow with your application. Whether it’s a small database for a personal project or a large database for a global organization, databases can scale to meet your needs.

Conclusion

Databases are the backbone of modern data management. Whether you're analyzing customer data, building a web application, or running large-scale data analytics, understanding databases and how to interact with them using SQL is a fundamental skill.

Why SQL?

SQL (Structured Query Language) is a powerful and essential tool for managing and interacting with relational databases. Here are some reasons why SQL is widely used:

1. **Standardized Language**: SQL is a standardized language used across many different database systems (like MySQL, PostgreSQL, Oracle, and SQL Server). This means that learning SQL gives you a broad skillset that is transferable between systems.
2. **Efficient Data Management**: SQL allows for efficient querying, inserting, updating, and deleting of data. It's optimized for handling large datasets and is capable of performing complex operations like joins, aggregations, and filtering with relatively simple syntax.
3. **Data Integrity**: SQL databases offer strong consistency and integrity features. With concepts like ACID (Atomicity, Consistency, Isolation, Durability) compliance, SQL databases ensure that transactions are processed reliably.
4. **Data Retrieval**: SQL is designed specifically to retrieve data in a structured and organized way. You can use SQL queries to pull out exactly the data you need from large datasets, which is often faster than other methods.
5. **Scalability**: SQL databases are highly scalable and can manage huge volumes of data. While initially you may not need a lot of scalability, SQL systems can grow as your data and requirements increase.
6. **Security**: SQL allows for the definition of access controls, so users can be given specific permissions (read, write, etc.) to databases or tables. This helps protect sensitive data.
7. **Widely Used in Industry**: SQL is a fundamental skill in many industries such as finance, e-commerce, healthcare, and more. Its universality makes it an in-demand skill for data analysts, data scientists, backend developers, and others working with databases.
8. **Complex Queries Made Simple**: SQL lets you easily combine multiple tables (through JOINs) and perform complex queries (with aggregate functions like COUNT(), SUM(), AVG(), etc.), which would otherwise require complicated programming logic in other languages.
9. **Integration**: SQL can easily integrate with many programming languages (such as Python, Java, C#, etc.), making it versatile and useful in different environments.

Overall, SQL is a crucial tool for working with structured data, providing efficiency, scalability, and control over database management tasks. Whether you are building web applications, analyzing data, or working in data engineering, knowing SQL can significantly streamline your workflow.

Execution of an SQL Statement

The execution of an SQL statement typically involves a few key steps, where a client application or a user interacts with the database system. Here's a simplified breakdown of how an SQL statement is executed:

1. SQL Query Preparation

- The process starts when you write an SQL query. For example, a query might look like:

SELECT * FROM employees WHERE department = 'Sales';

2. Query Submission

- The SQL query is submitted to the **Database Management System (DBMS)** through a client interface. This could be a command-line interface (CLI), a graphical user interface (GUI), or even through an application that uses a programming language (like Python, Java, etc.).
- The client sends the SQL statement to the DBMS.

3. SQL Parsing

- The DBMS **parses** the SQL statement. In this stage, the DBMS checks for:
    - **Syntax errors**: Whether the query is written correctly.
    - **Semantics**: Whether the query makes sense (e.g., checking if the table employees exists and whether the department column is valid).
- If any issues are found, the DBMS will return an error message to the user.

4. Query Optimization

- Once the query passes the parsing phase, the **query optimizer** kicks in. The DBMS attempts to figure out the most efficient way to execute the query.
- The optimizer may decide on the **execution plan**, which could involve:
    - Choosing the most efficient index to use.
    - Deciding on the order in which to join tables if the query involves multiple tables.
    - Figuring out whether to use sorting, filtering, or other operations.

5. Execution Plan Generation

- Based on the optimization, the DBMS creates an **execution plan**. This plan contains a series of steps (operations) that the DBMS will execute to fulfill the query.
- This execution plan is crucial for improving performance, especially for large datasets.

6. Execution

- Now, the query is actually executed:
    - The DBMS retrieves the required data from the database using the execution plan.
    - It may involve accessing tables, indexes, or other data structures to get the data.
    - If the query involves updates (e.g., INSERT, UPDATE, or DELETE), the changes are made to the database.

7. Return Results (For SELECT Queries)

- If the SQL statement is a **SELECT** query, the DBMS sends the result (data) back to the client.
- The data is formatted in a way that the client can process and display it, typically in the form of rows and columns (like a table).

8. Commit or Rollback (For Data Modifications)

- If the query modifies data (for example, an INSERT, UPDATE, or DELETE), the DBMS may:
    - **Commit** the changes (save the data to the database).
    - If there was an error, the DBMS may **rollback** the transaction to revert the changes, ensuring the database's consistency.

9. Completion

- Once the results are returned or changes committed, the execution of the SQL query is complete.

Example Walkthrough: A Simple SQL Query

Let's walk through a simple SELECT query to better understand each step.

SQL Statement:

SELECT name, department FROM employees WHERE department = 'Sales';

Steps:

1. **Preparation**: The query selects name and department from the employees table, where the department is 'Sales'.
2. **Submission**: The query is submitted to the DBMS (e.g., through a SQL command line, a web interface, or an application).
3. **Parsing**: The DBMS checks the syntax to ensure the query is correct and that the employees table exists and has name and department columns.
4. **Optimization**: The optimizer determines the most efficient way to execute this query, possibly by using an index on the department column if one exists.
5. **Execution Plan**: The DBMS plans to scan the employees table and filter rows where department is 'Sales'.
6. **Execution**: The DBMS retrieves rows from the employees table where the department matches 'Sales' and returns them.
7. **Return Results**: The matching rows are sent back to the client, showing the name and department columns for employees in the 'Sales' department.
8. **Completion**: The query execution is complete, and the client can use the results as needed.

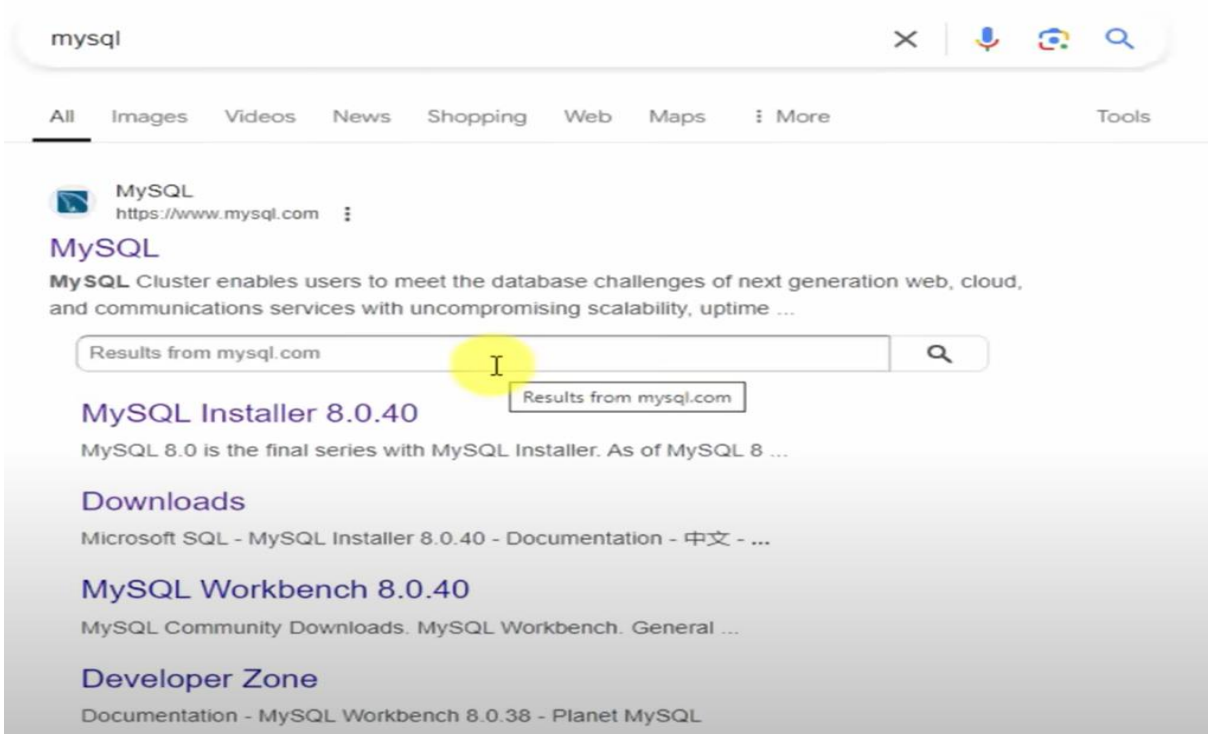
Tools and Interfaces for SQL Execution

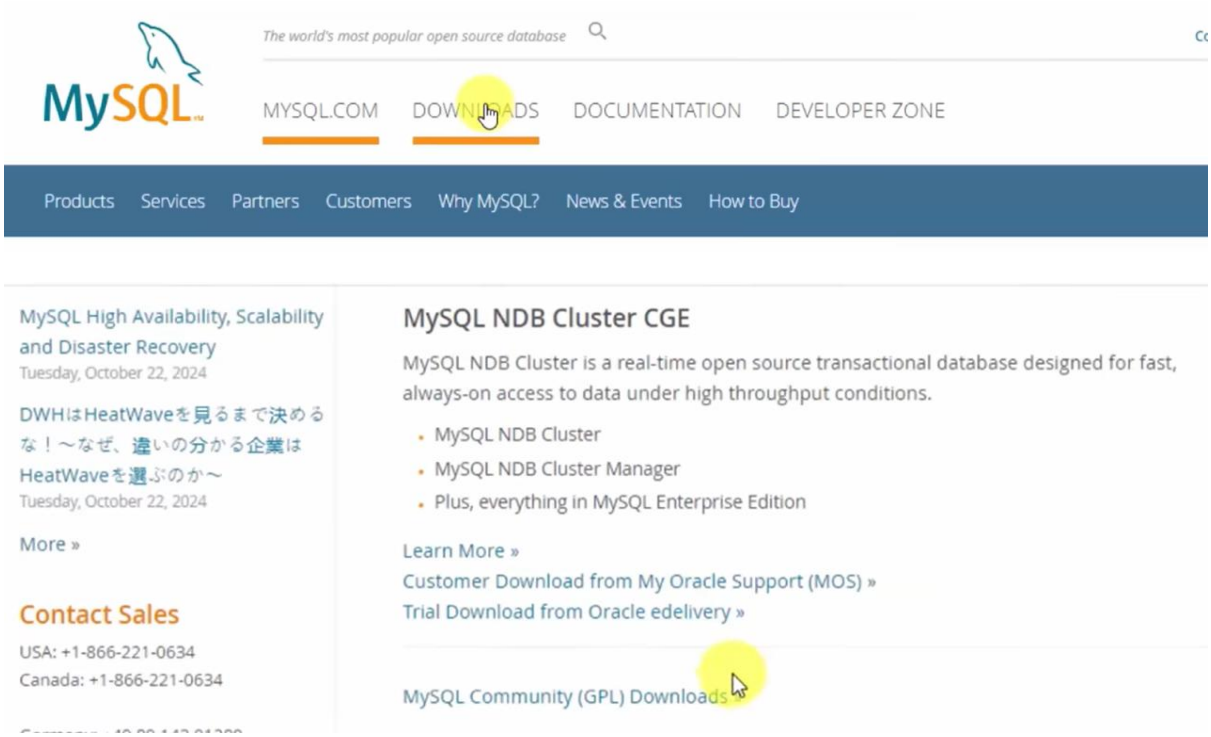
SQL queries can be executed using various tools and methods:

- **SQL Command Line**: Most databases (like MySQL, PostgreSQL, Oracle) come with a command-line interface (CLI) to directly input and execute SQL queries.
- **Graphical User Interfaces (GUIs)**: Tools like **MySQL Workbench**, **pgAdmin**, **DBeaver**, or **SQL Server Management Studio** provide a user-friendly interface to write and execute queries.
- **Programming Languages**: SQL can be executed from within programming languages (e.g., Python using libraries like sqlite3, psycopg2, or SQLAlchemy for database interaction).

In essence, SQL execution is a series of structured steps that ensure the database can handle requests efficiently and accurately.

Installing MySql





First, go to chrome and search **MySQL download** -> then click on the **MySQL** that show in the first image -> then go to **Download** -> then click on **MySQL Community (GPL) Downloads** -> little scroll down and click on **MySQL Installer for windows** in below image shown.

MySQL Community Downloads

- MySQL Yum Repository
- MySQL APT Repository
- MySQL SUSE Repository

- MySQL Community Server
- MySQL NDB Cluster
- MySQL Router
- MySQL Shell
- MySQL Operator
- MySQL NDB Operator
- MySQL Workbench

- MySQL Installer for Windows

- C API (libmysqlclient)
- Connector/C++
- Connector/J
- Connector/NET
- Connector/Node.js
- Connector/ODBC
- Connector/Python
- MySQL Native Driver for PHP

- MySQL Benchmark Tool
- Time zone description tables
- Download Archives

MySQL Enterprise Edition for Developers

Free for learning, developing, and prototyping.

Download Now »

MySQL Installer 8.0.40

Note: MySQL 8.0 is the final series with MySQL Installer. As of MySQL 8.1, use a MySQL product's MSI or Zip archive for installation. MySQL Server 8.1 and higher also bundle MySQL Configurator, a tool that helps configure MySQL Server.

Select Version:

8.0.40

Select Operating System:

Microsoft Windows

Windows (x86, 32-bit), MSI Installer	8.0.40	2.1M	Download
(mysql-installer-web-community-8.0.40.0.msi)	MD5: 42da0dc06ad328fe2451eeb3998fb016 | Signature		
Windows (x86, 32-bit), MSI Installer	8.0.40	306.4M	Download
(mysql-installer-community-8.0.40.0.msi)	MD5: 8c1bf3a205d5e191e36dc334a10f55d2 | Signature		

MySQL Community Downloads

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

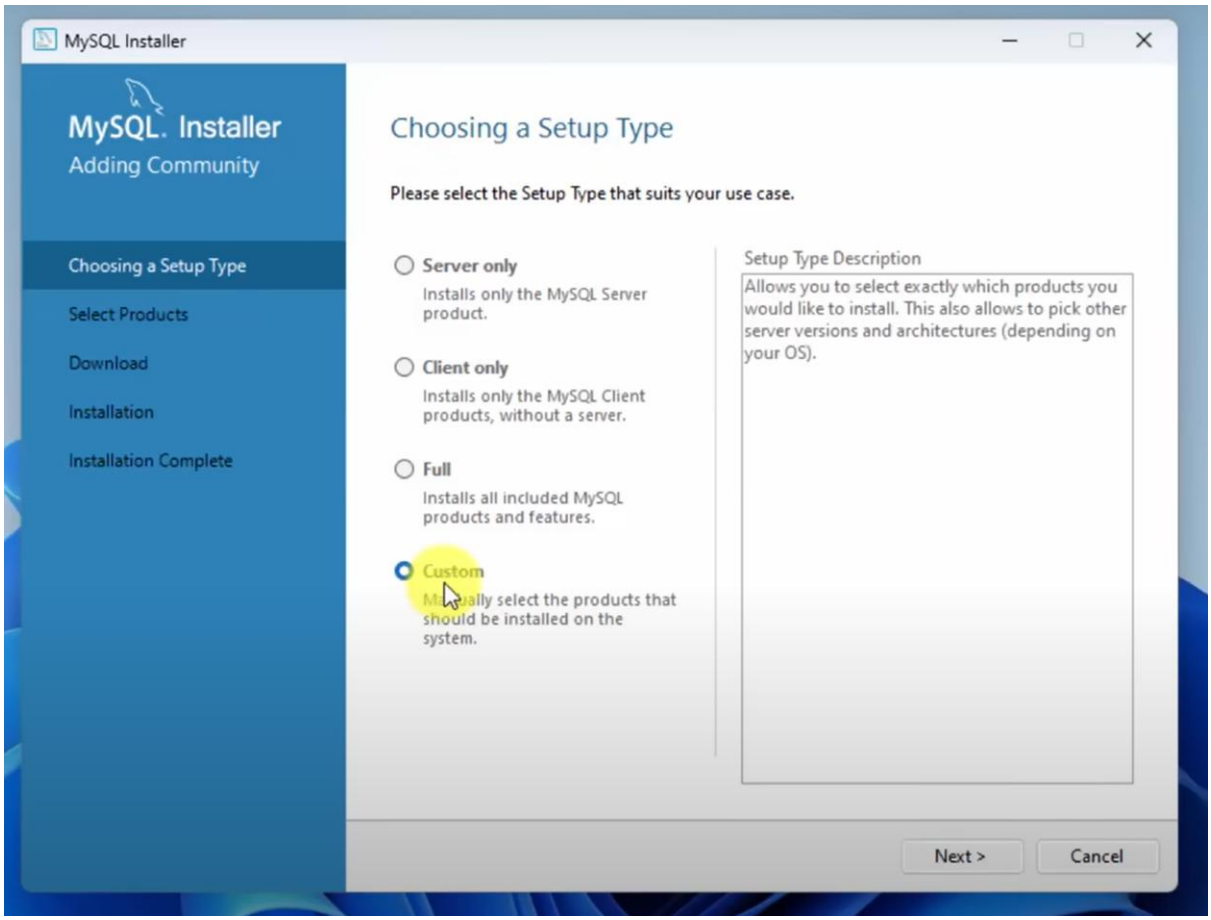
Login » using my Oracle Web account

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.

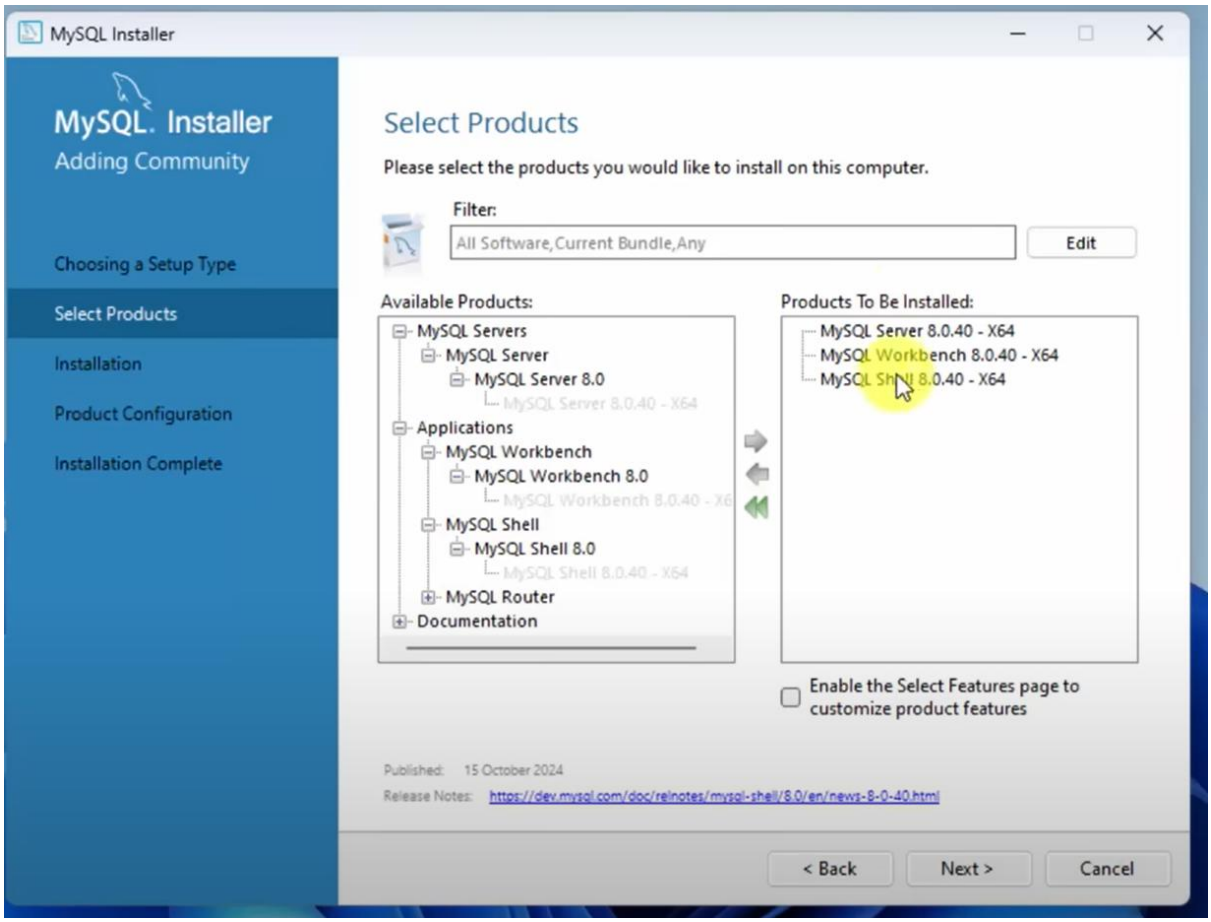
No thanks, just start my download.

Just double click on the MySQL file and click on Open



Here you need to select Setup Type

Select Custom -> Then Click on Next



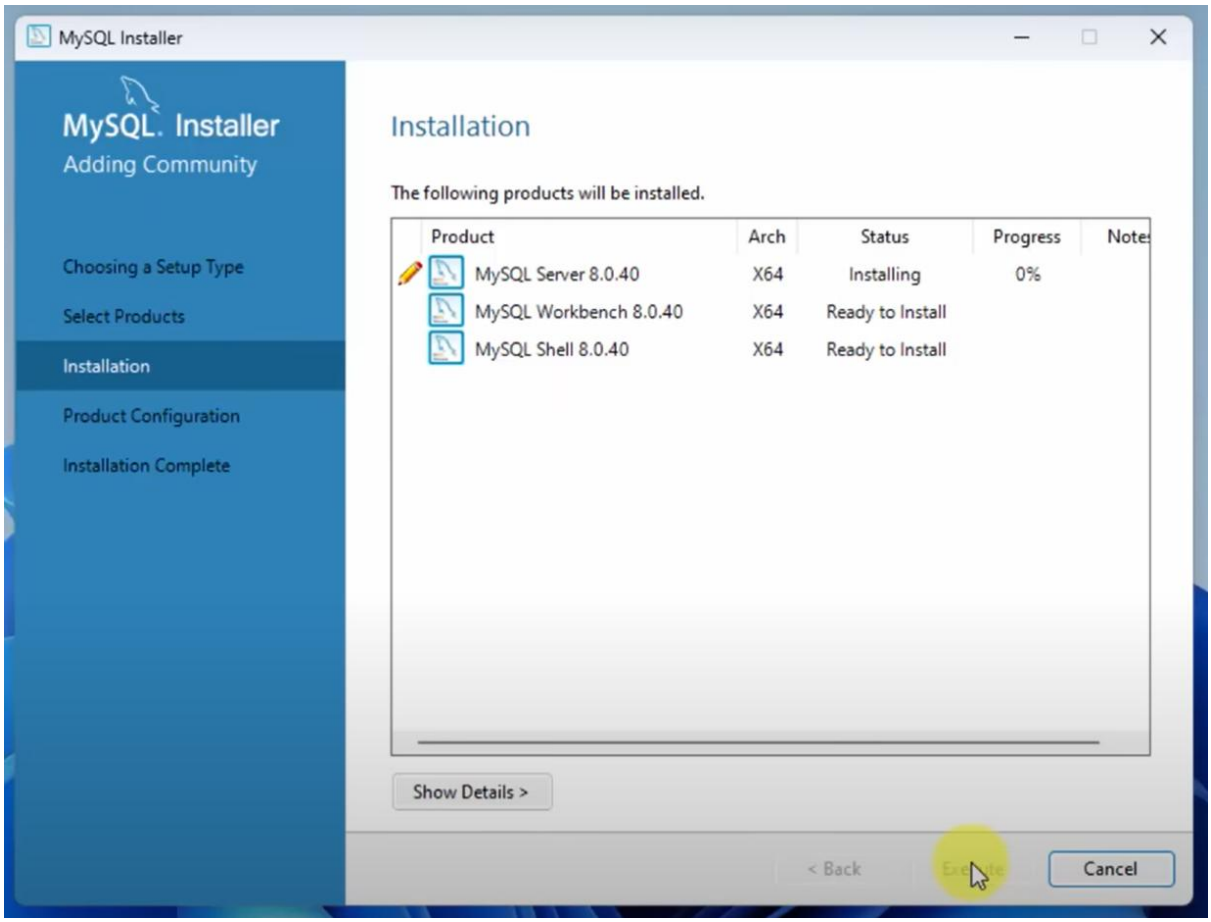
From Select Products

MySQL Server -> MySQL Server 8.0.40 – X64

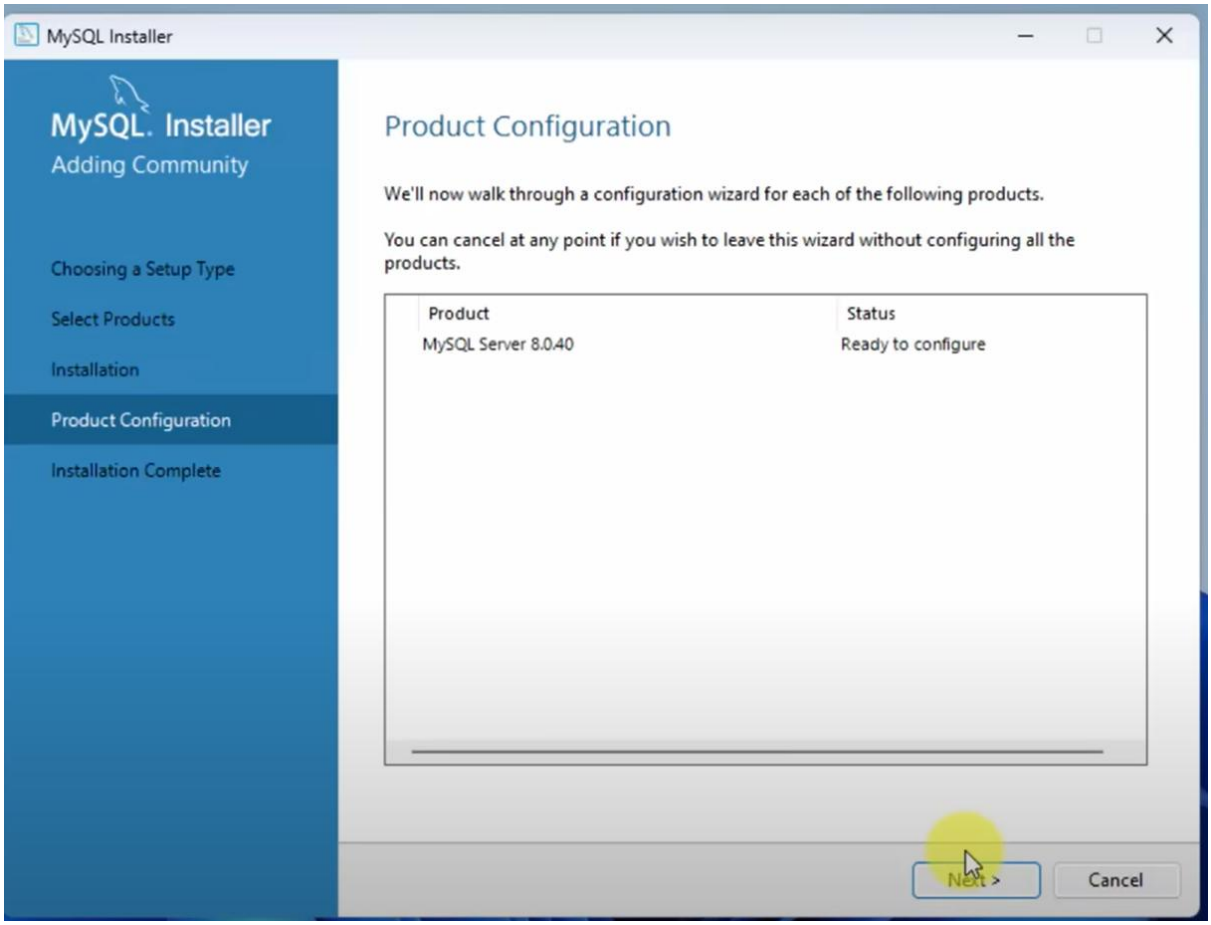
Applications -> MySQL Workbench -> MySQL Workbench 8.0 -> MySQL Workbench 8.0.40 – X64

And From MySQL Shell -> MySQL Shell 8.0 -> MySQL Shell 8.0.40 – X64

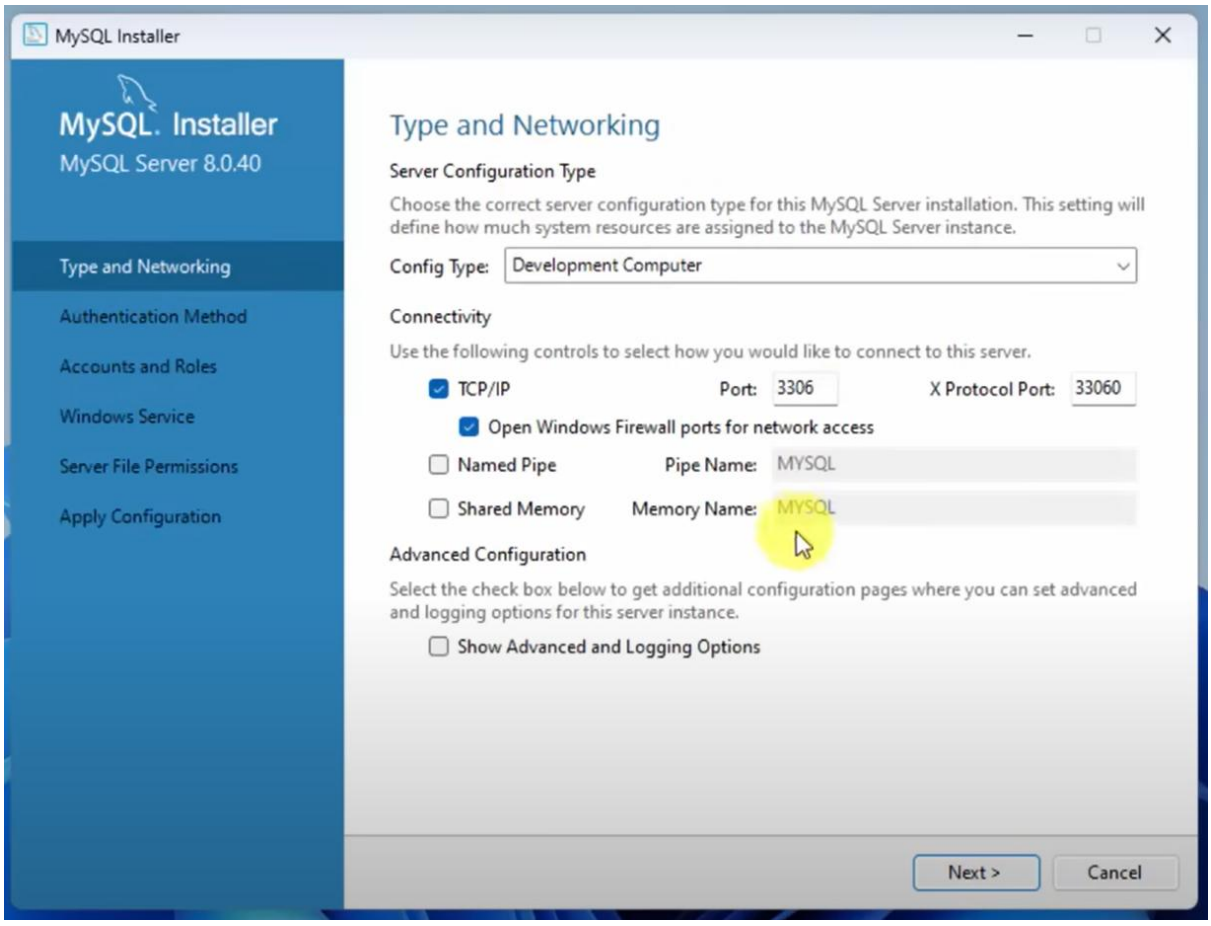
And Click on Next.



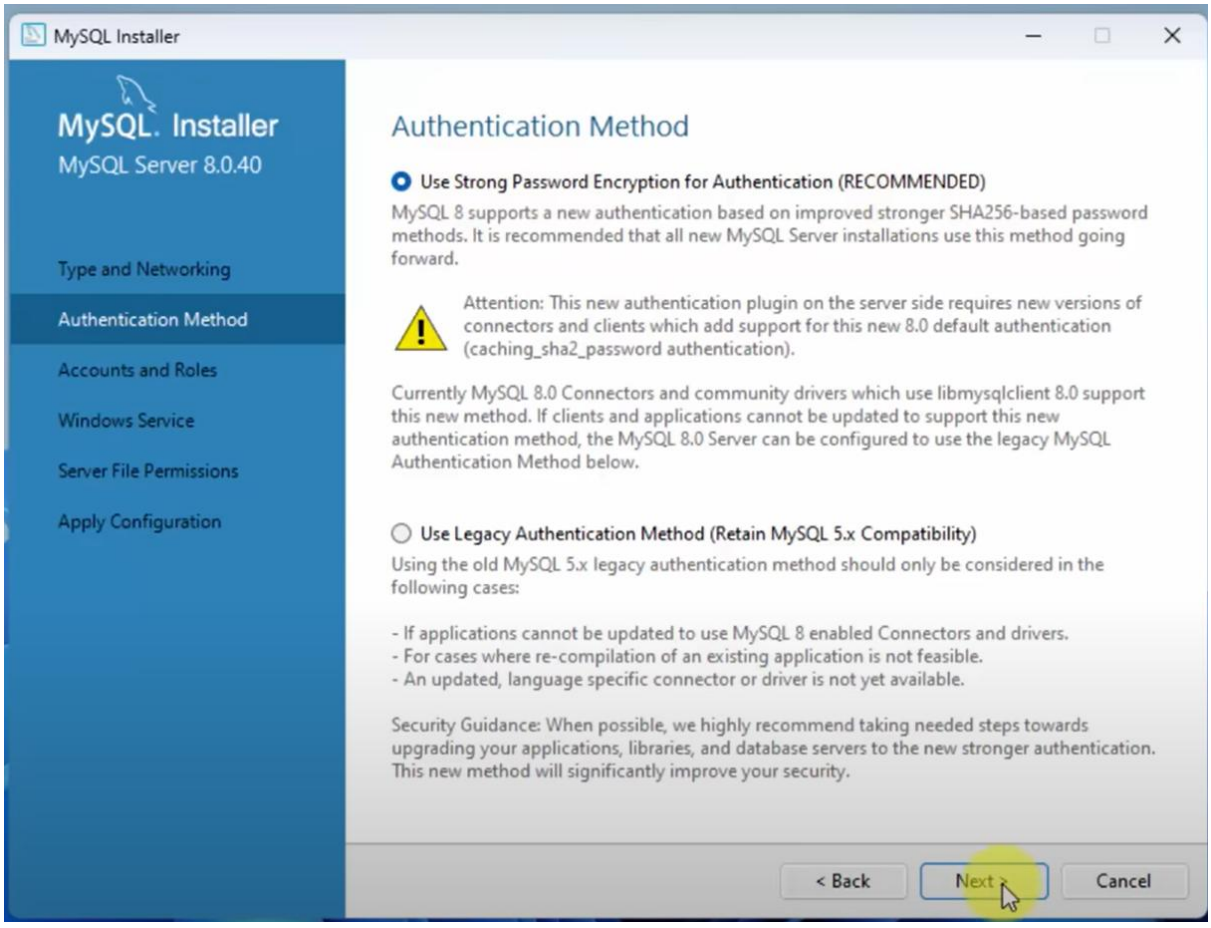
Click on Execute



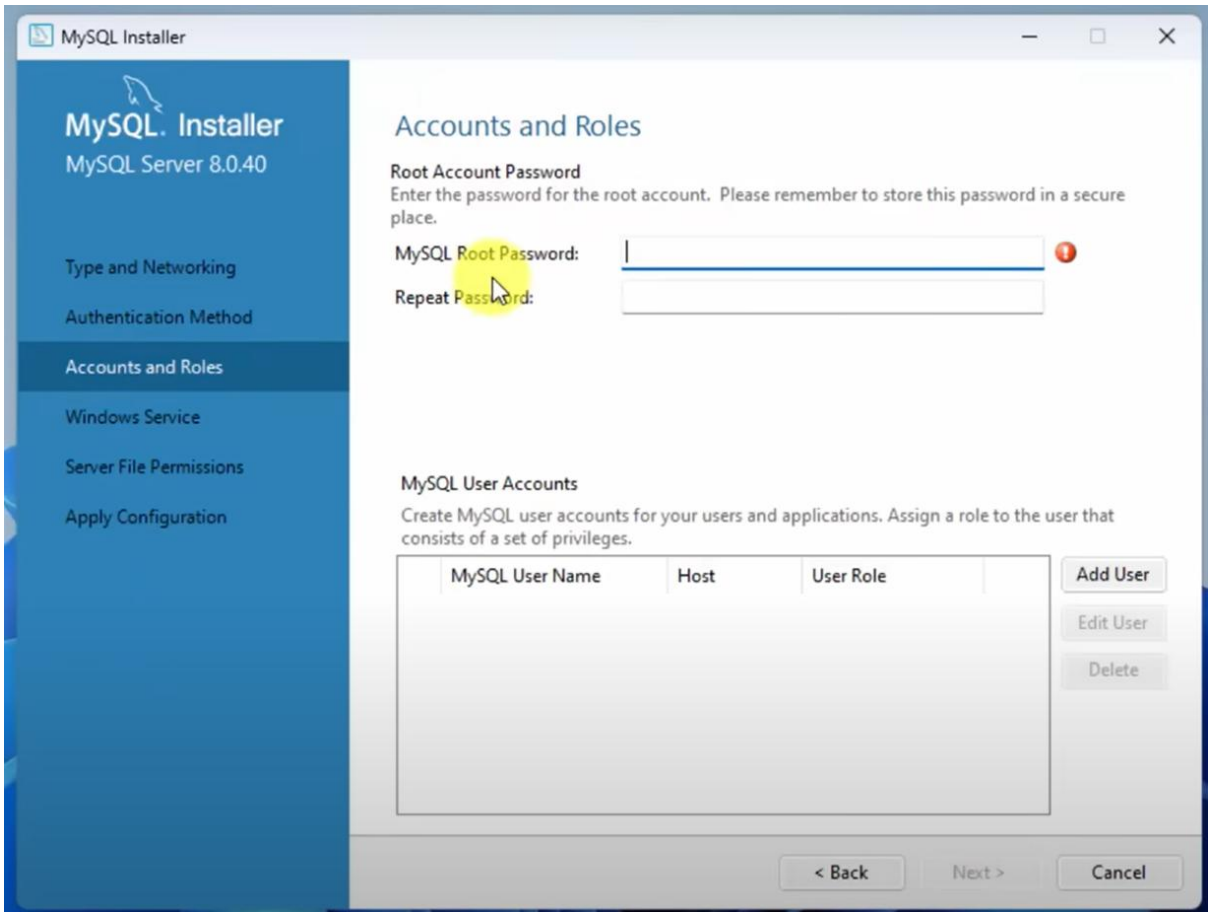
Here Click on Next



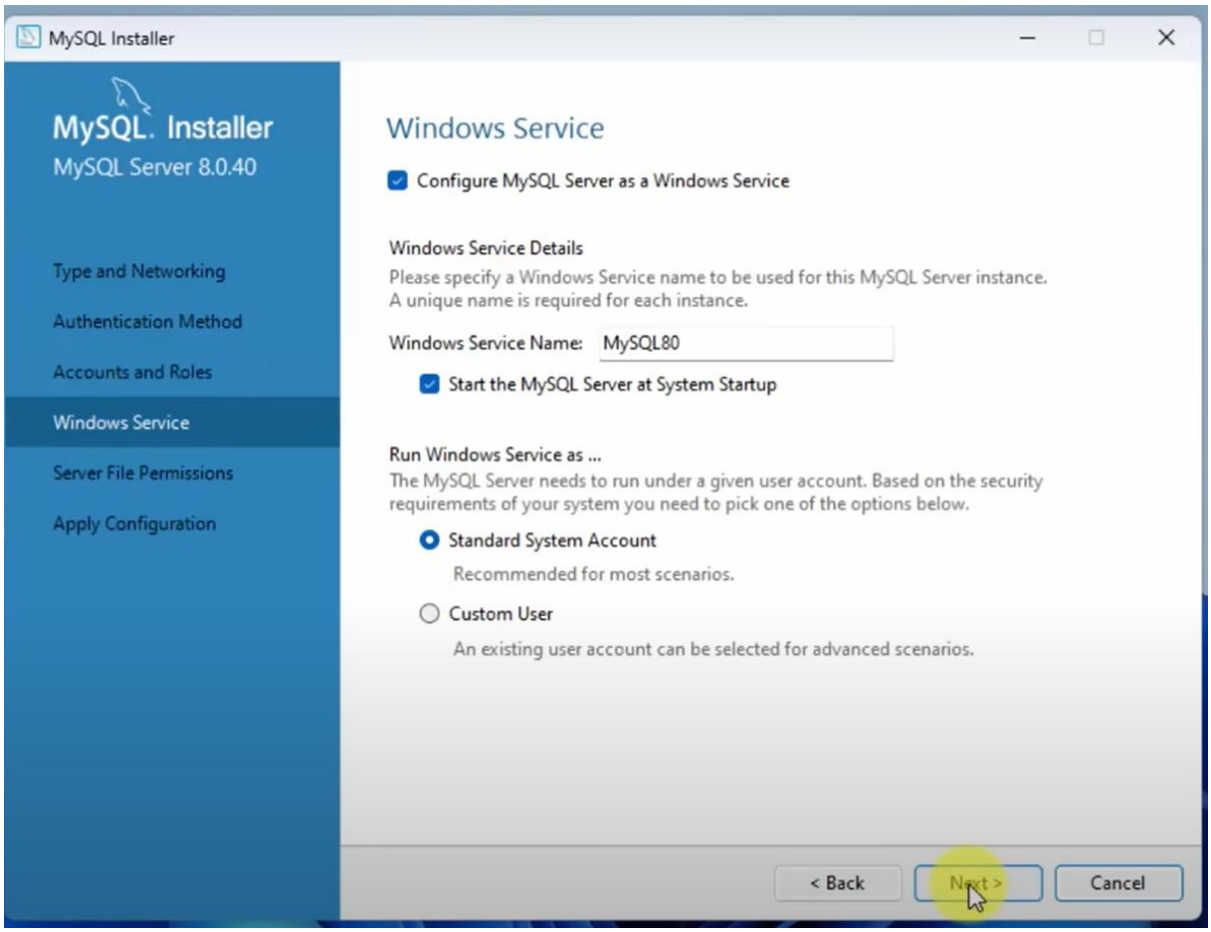
Take as it and click on next



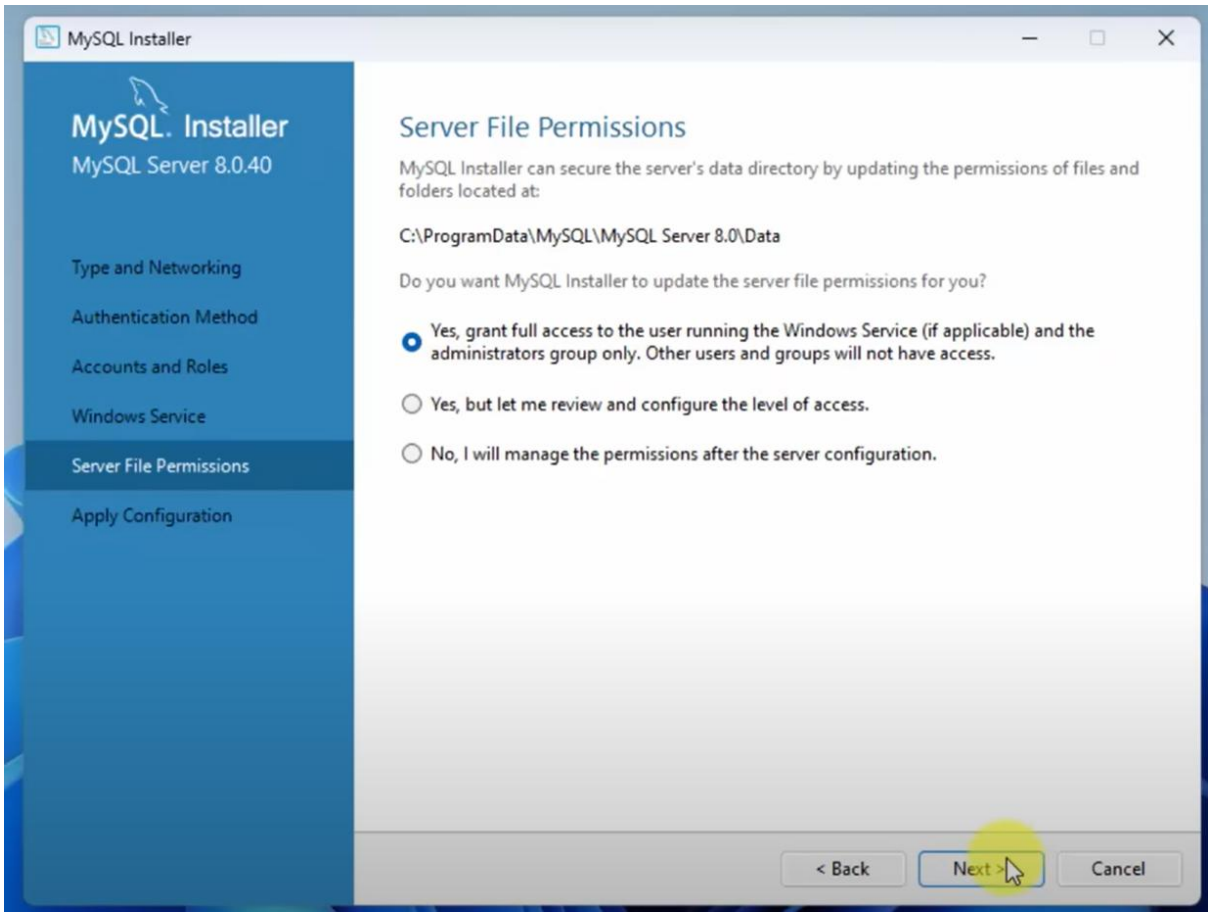
Take as it and click on next



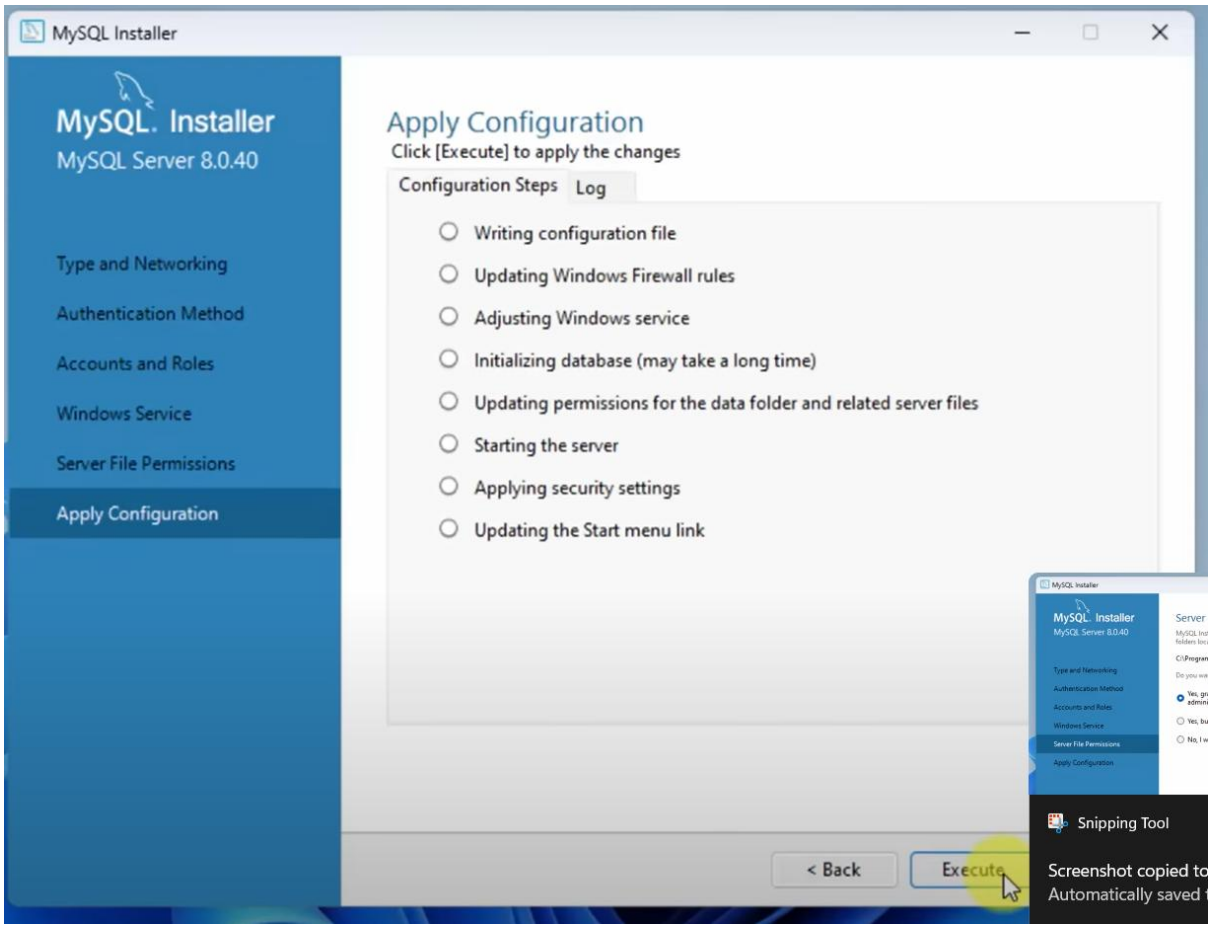
Here you have to set a password and click on next



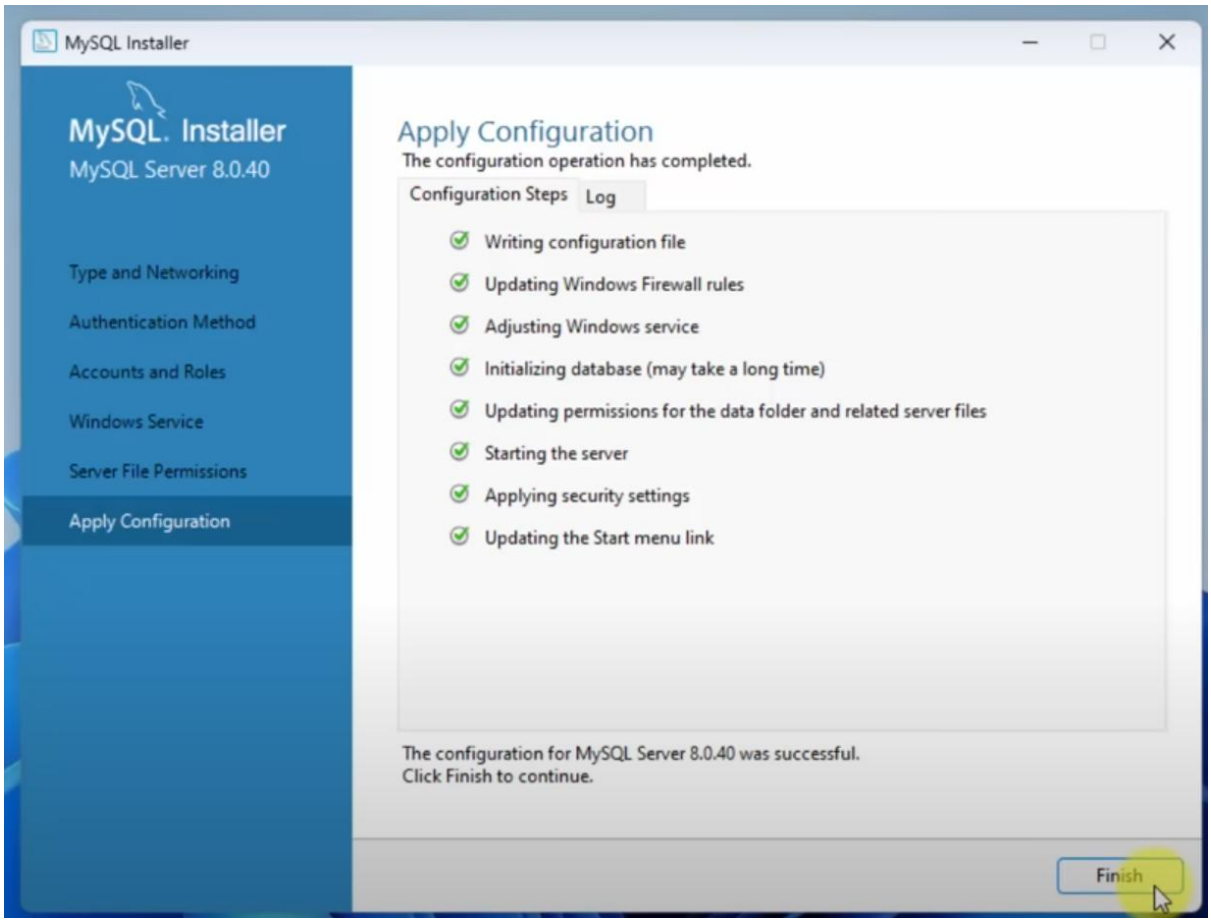
Take as it and click on next



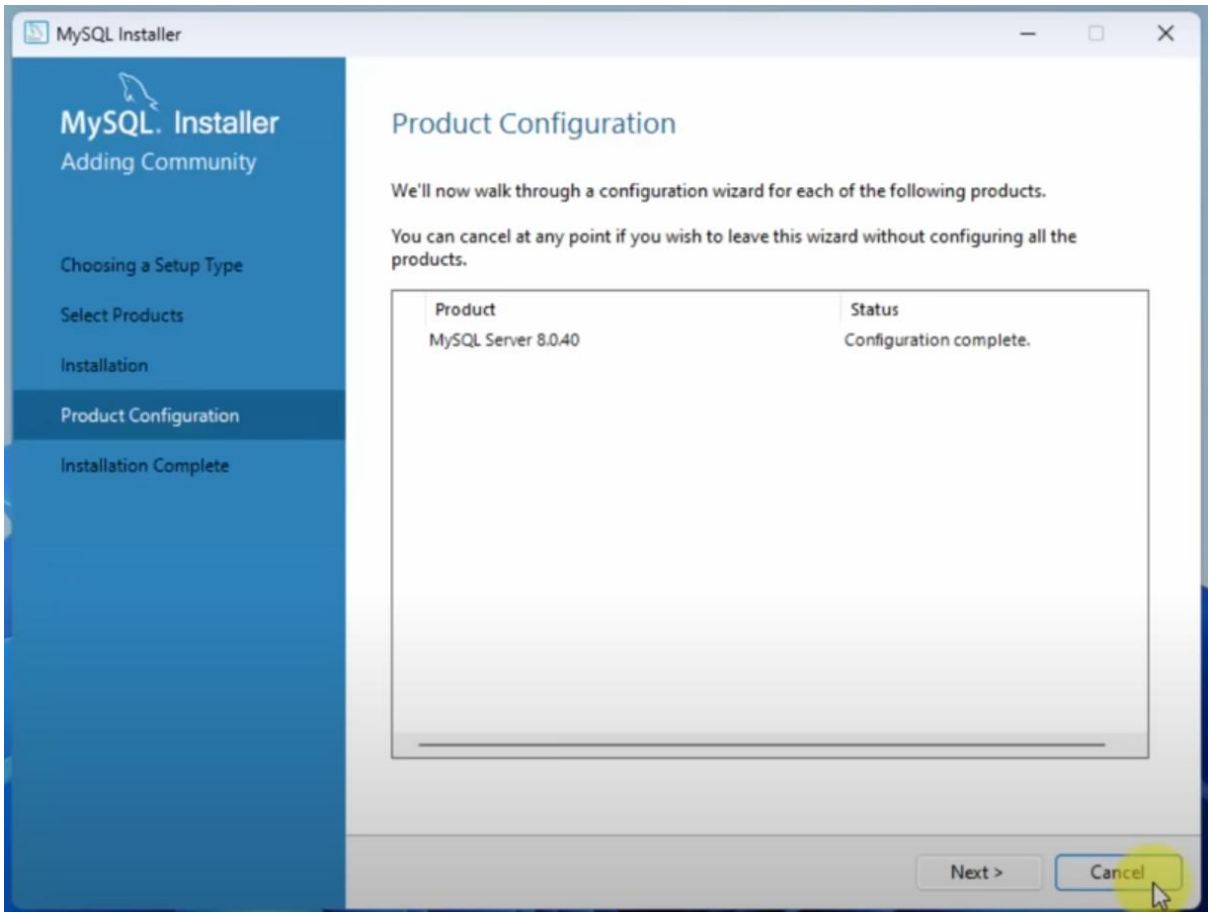
Take as it and click on next



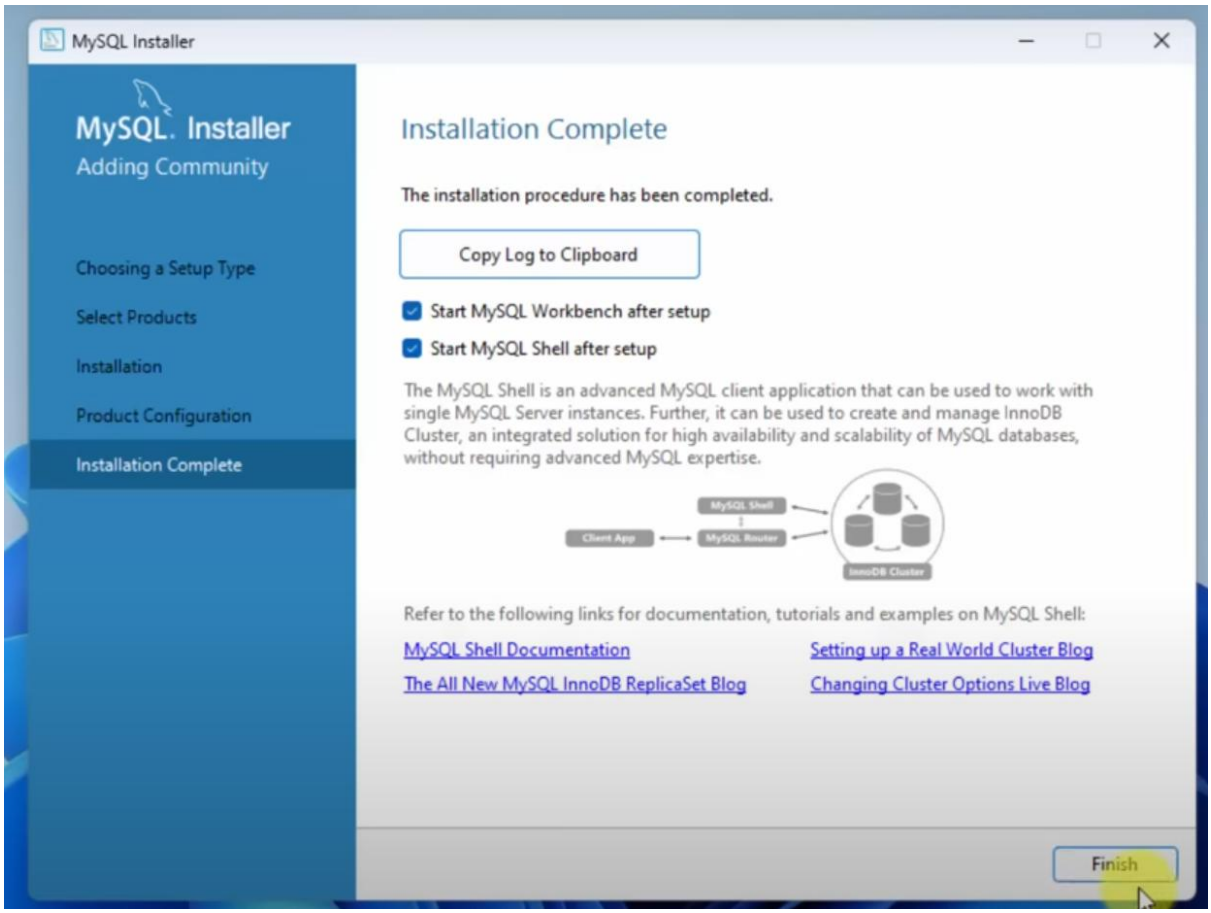
Click On Execute Button



Configuration done now click on finish



Click on next



Now installation is Complete. Click On Finish

Here there something different with your system. If the path is not set in your system environment then you have set the path.

Open file manager and go to C drive and follow this path.

C:\Program Files\MySQL\MySQL Server(version)\bin

Copy and search for Edit the system environment variable -> Environment Variables -> on system variable double click on path -> then paste the path here that you copy.

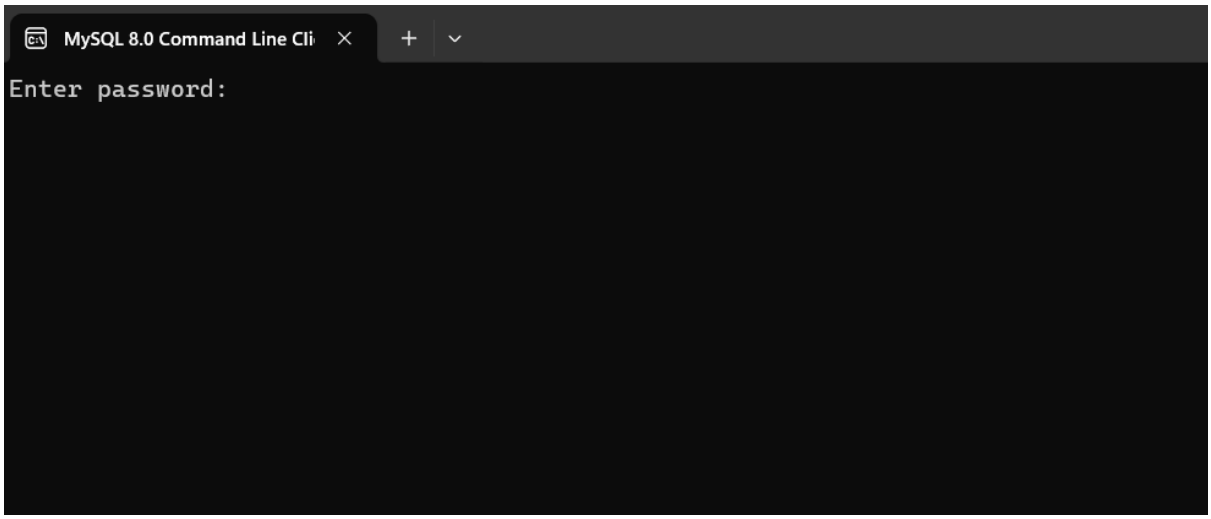
Then click ok -> ok -> and apply.

Importing IMDB Dataset

Here, I am working on **IMDB dataset**.

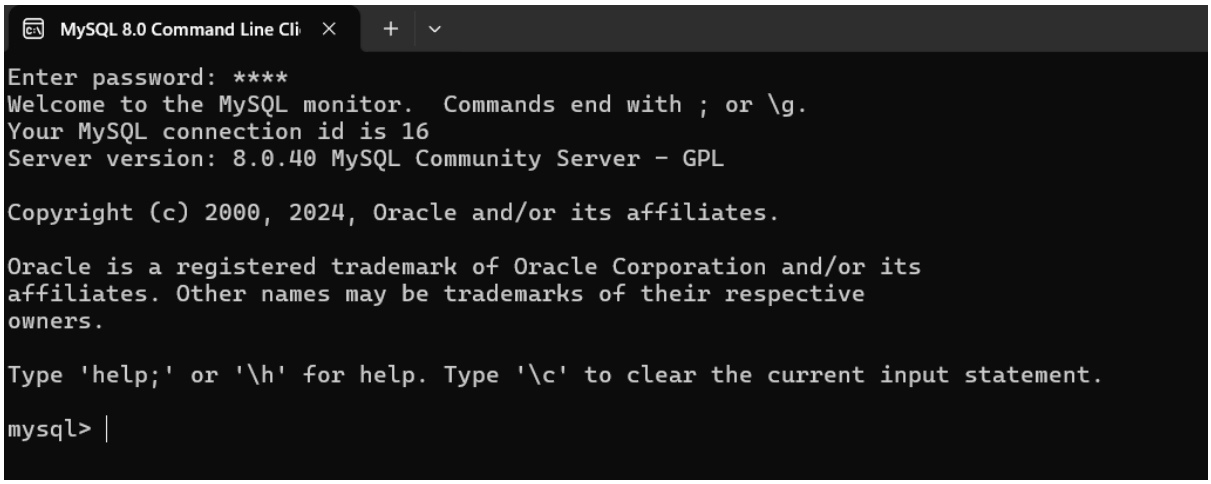
So, you can use MySQL CLC or MySQL Workbench. I am using Command Line Client of MySQL.

Open MySQL Command Line Client

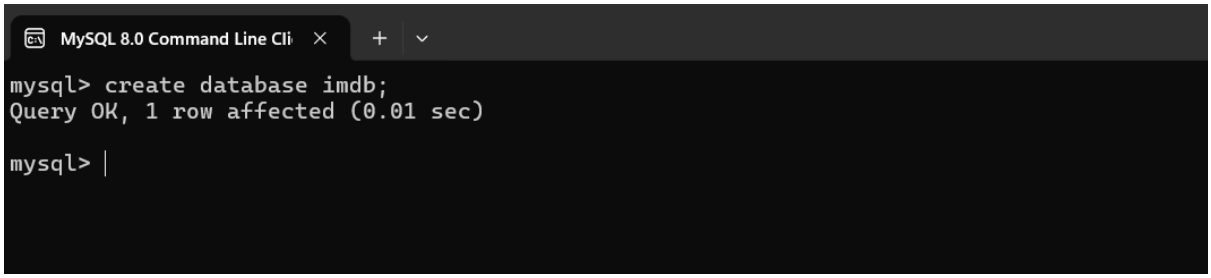


Type here the password that you had entered while doing configuration of MySQL.

Here my Pass is root.

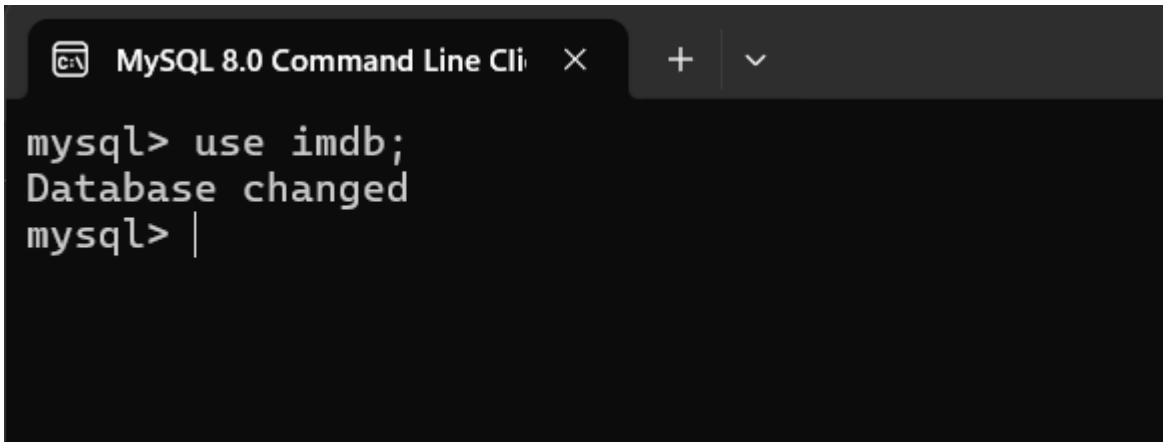


Is look like this when you enter the pass.



So, first we need a database.

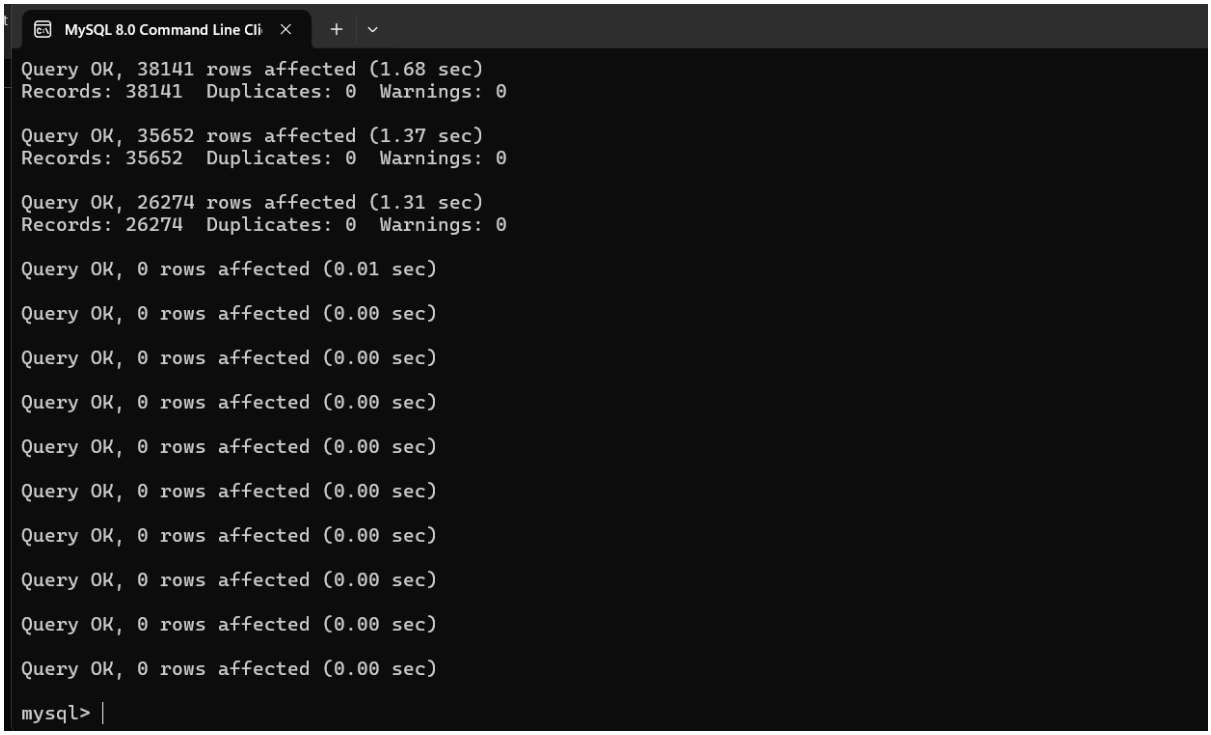
After creating database we need this databse in use




```
mysql> source C:\Users\Amol Thakare\Downloads\imdb.sql
```

After that we have to use imdb dataset

I already have a dataset of imdb so, I am directly take to the database using above command.



Data has been uploaded successfully.

Now, if you want to see the how many tables in the database then use show tables command.

```
mysql> show tables;
+------------------+
| Tables_in_imdb   |
+------------------+
| actors           |
| directors        |
| directors_genres |
| movies           |
| movies_directors |
| movies_genres    |
| roles            |
+------------------+
7 rows in set (0.01 sec)

mysql>
```

Here, I want to see what information is given in the tables.

Then I am using describe and particular table content.

Like **describe actors, describe directors** and so on.

```
mysql> describe actors;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| id         | int          | NO   | PRI | 0       |       |
| first_name | varchar(100) | YES  | MUL | NULL    |       |
| last_name  | varchar(100) | YES  | MUL | NULL    |       |
| gender     | char(1)      | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
4 rows in set (0.02 sec)

mysql> describe roles;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| actor_id | int          | NO   | PRI | NULL    |       |
| movie_id | int          | NO   | PRI | NULL    |       |
| role     | varchar(100) | NO   | PRI | NULL    |       |
+----------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> describe directors;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| id         | int          | NO   | PRI | 0       |       |
| first_name | varchar(100) | YES  | MUL | NULL    |       |
| last_name  | varchar(100) | YES  | MUL | NULL    |       |
+------------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> describe movies;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| id        | int          | NO   | PRI | 0       |       |
| name      | varchar(100) | YES  | MUL | NULL    |       |
| year      | int          | YES  |     | NULL    |       |
| rankscore | float        | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql>
```