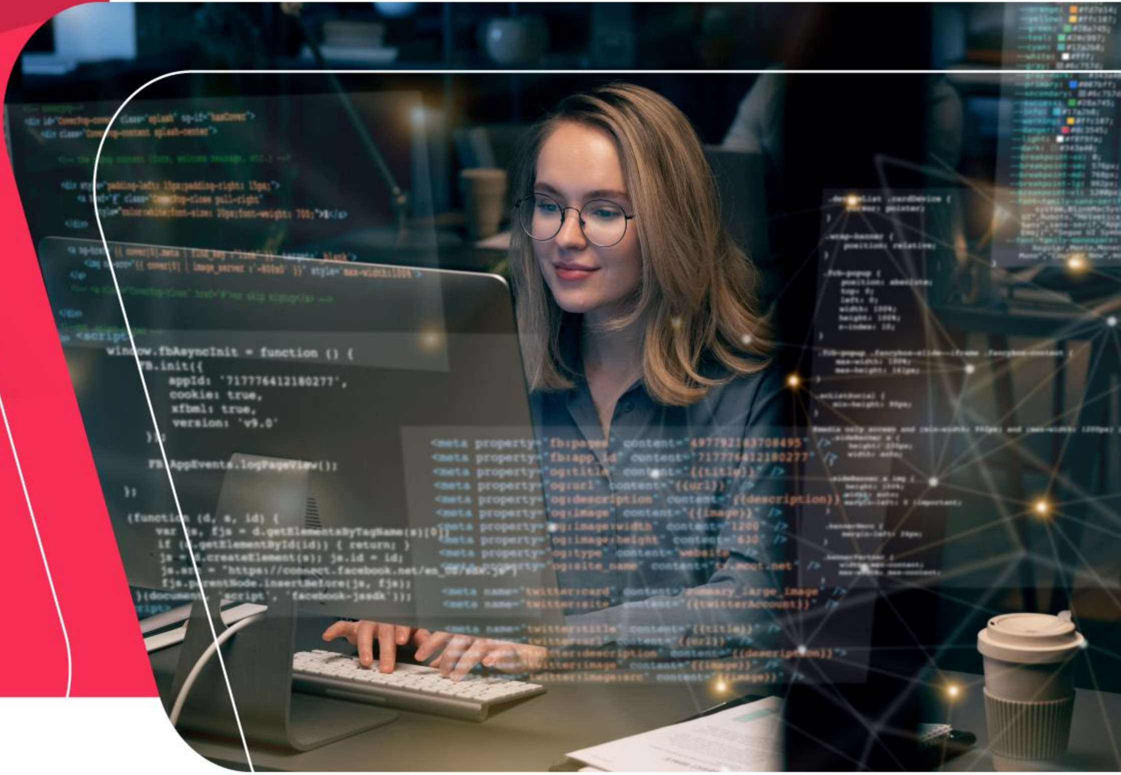
TIA-Academy

Java Bootcamp

Day 26



Technologies will be Use

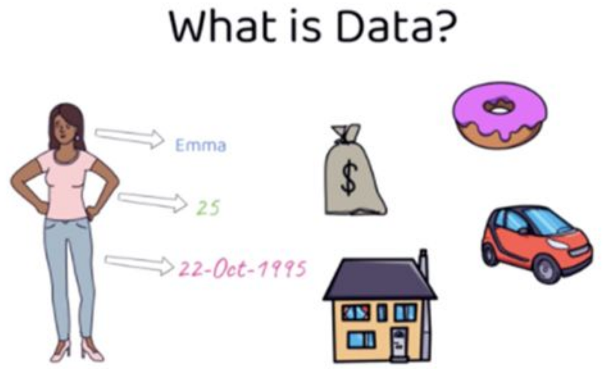
- JDK 8/**11**/15
- JRE 8/**11**/15
- **Intellij IDEA Community Edition**
- JAVA 3rd Party Library (Network, DB, etc)
- **MySQL Server Community**
- MySQL Workbench or SQLyog (Windows)

TIA-Academy

DATABASE

What is Data?

- Data can be anything and everything.
- Any information or fact can be considered as data.
- Your name, age, data of birth or any other information such as your house address, bank balance, vehicle you drive or even the food you eat can be considered as data.



What is Data?

- Details related to a School, technology, statistics and even mathematics can be considered as data and be stored in a database.
- Data can be in any form such as an image, file, voice recording, video or even a plain text etc.
- For a school, data can be information related to its teachers, students or the subjects they teach.
- To summarize, data can be anything and everything and it can be in any format.

TIA-Academy

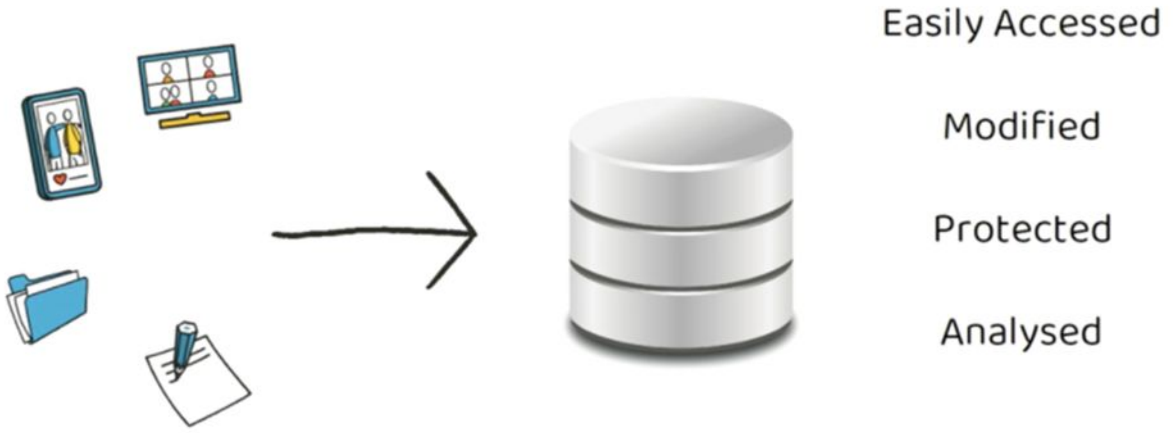
What is Database?

- Database can be considered as a container filled with data or information which is electronically stored in a computer system.
- Data in any form can be stored into the database.
- Purpose of storing data in a database is so that it can be easily accessed, modified, protected and analysed.

DATA
DATABASE
SERVER

What is Database?

- You may already be using database everyday without your knowledge because most of the websites or mobile apps that you use everyday such as Google, Instagram, WhatsApp, Facebook etc already use database to store their data.



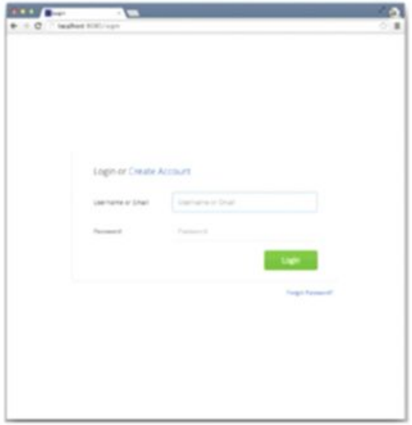
TIA-Academy

What is Database?

- As an example, let's say, for your **college demo project, you plan to develop a simple college timesheet web application**.
- Basically, this application should allow you to login through a URL and then have options to enter staff details, timesheet details, student attendance, student marks etc.
- **First thing this application will need is a database to store and retrieve data** hence you need to first install a database on your computer, which will connect to this timesheet application.
- So like any other software installed on your computer, **database software will also be allocated a specific location in your computer hard drive.** This is the location where the data from the database will be stored.

What is Database?

"College Timesheet Web Application"



Staff Details

Timesheet Details

Student Attendance

Student Marks

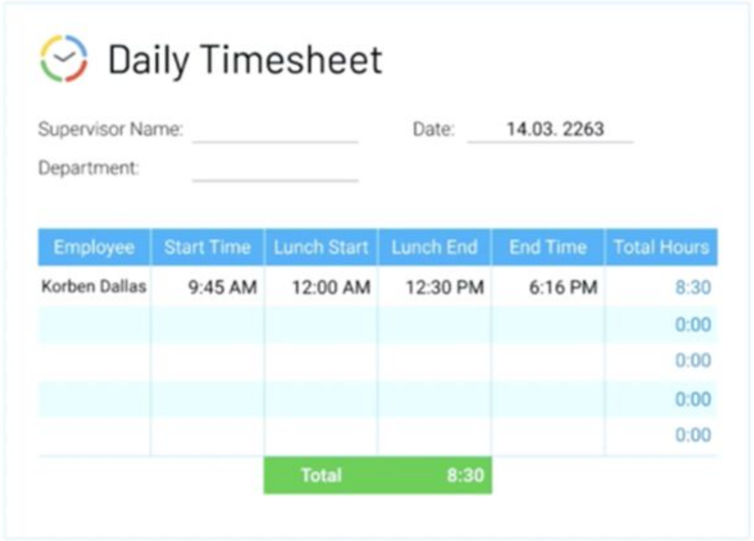
Etc...

/Library/PostgreSQL/12

D:\Database\Oracle19c

TIA-Academy

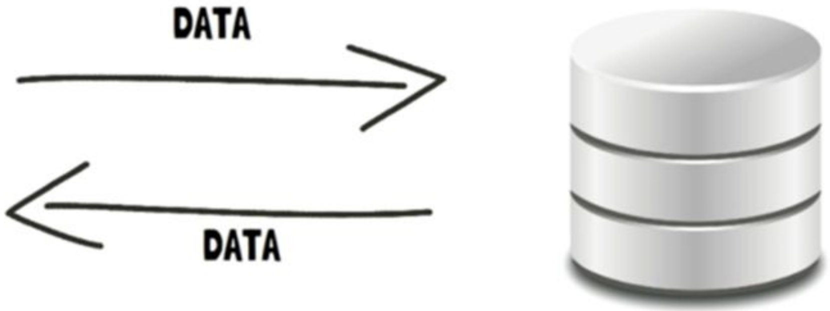
What is Database?



Daily Timesheet

Supervisor Name:		Date:	14.03. 2263
Department:			

Employee	Start Time	Lunch Start	Lunch End	End Time	Total Hours
Korben Dallas	9:45 AM	12:00 AM	12:30 PM	6:16 PM	8:30
					0:00
					0:00
					0:00
					0:00
		Total	8:30		



TIA-Academy

What is Database?

- This is a **simple example of how an application uses a small database** to store, modify and retrieve data.
- When it comes to bigger applications or **companies or government, they need huge database to store huge data** which may be installed on huge servers.

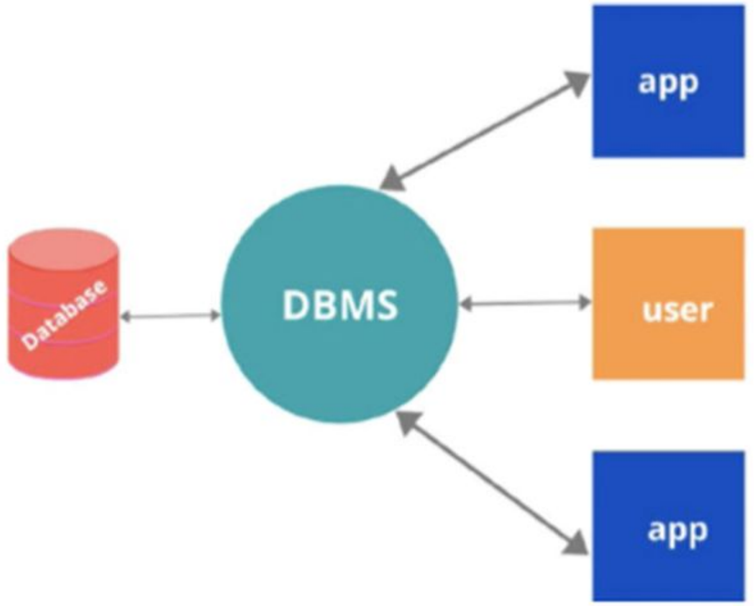
What is Database?

- Is Database same as DBMS?

DBMS ≠ DATABASE ?

TIA-Academy

What is DBMS?



Lot of people refer to database as DBMS but that's not entirely correct because database is just a container which stores data whereas Database Management System or DBMS is a software which is used to manage your database.

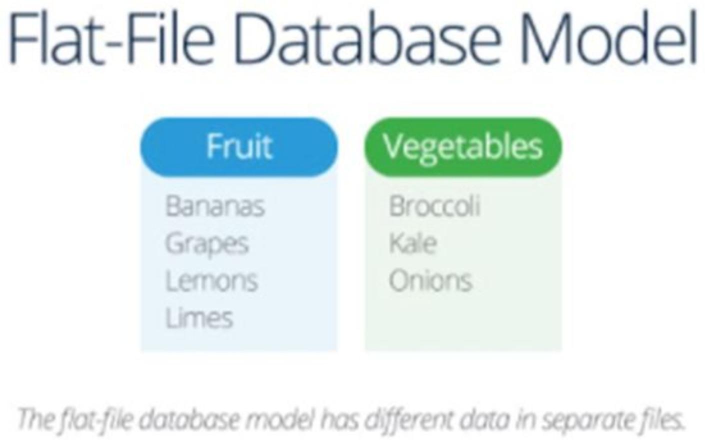
You need DBMS to interact with database, to store, modify, retrieve and protect data. DBMS is also required to create, modify and delete databases.

Users like you and me can use DBMS to enter commands in specific language to interact with Database.

Example of a DBMS is MySQL, PostgreSQL, MongoDB, Neo4j, Cassandra etc.

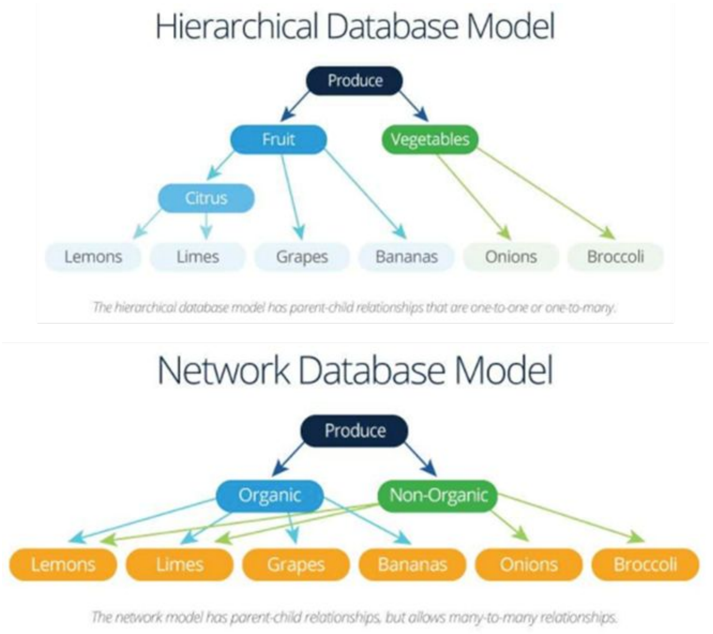
Evolution of Database

- Evolution of database started in **1960's when the first type of database were made which was the Flat File** Database.
- Here the data was stored in simple files such as CSV file or fixed length files etc.



TIA-Academy

Evolution of Database


Hierarchical Database Model
Produce
Fruit
Vegetables
Citrus
Lemons
Limes
Grapes
Bananas
Onions
Broccoli
The hierarchical database model has parent-child relationships that are one-to-one or one-to-many.
Network Database Model
Produce
Organic
Non-Organic
Lemons
Limes
Grapes
Bananas
Onions
Broccoli
The network model has parent-child relationships, but allows many-to-many relationships.

- Later on came the Hierarchical Database and then the Network database which stored data though parent child relation ship.
- But both Hierarchical and Network database were not capable of storing complex data relationships hence were soon replaced by Relational database.

TIA-Academy

Evolution of Database

- **Fast forward to 202X**,
- In today's world, there are mainly 2 popular database types.
  1. Relational Database
  2. Non-Relational Database (or NoSQL Database)
- As per the usage, **over 74% of database used today are relational database** but due to the immense raise in data usage over the past decade, mainly due to social media platforms, non relational database have become very popular.

TIA-Academy

Relational Database

- In a relational database, data is stored through **collection of tables**. **These tables are related to one another.**
- Each table consist of columns and rows. Each column has a name and a data type. Data type can be said as a data rule which is associated to every column. Only those data that satisfy these data rule can be inserted in the specific column.
- **A row can be treated as a record** which is formed by single or multiple columns.

TIA-Academy

Relational Database

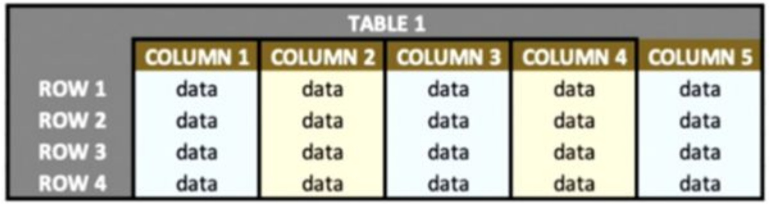
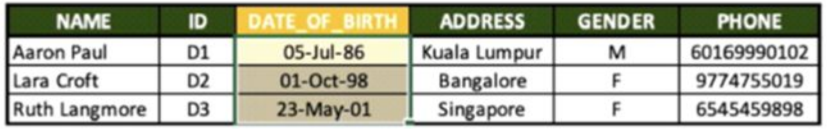


TABLE 1

	COLUMN 1	COLUMN 2	COLUMN 3	COLUMN 4	COLUMN 5
ROW 1	data	data	data	data	data
ROW 2	data	data	data	data	data
ROW 3	data	data	data	data	data
ROW 4	data	data	data	data	data



NAME	ID	DATE_OF_BIRTH	ADDRESS	GENDER	PHONE
Aaron Paul	D1	05-Jul-86	Kuala Lumpur	M	60169990102
Lara Croft	D2	01-Oct-98	Bangalore	F	9774755019
Ruth Langmore	D3	23-May-01	Singapore	F	6545459898

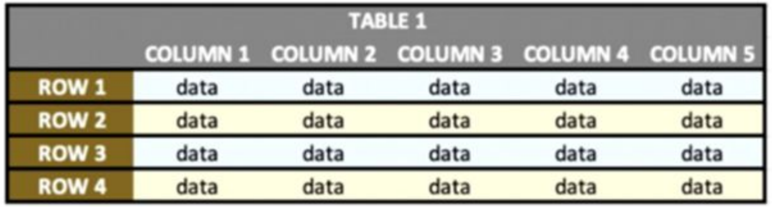
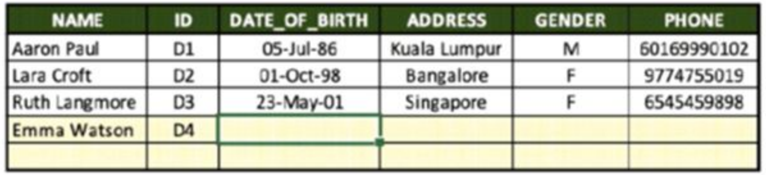


TABLE 1

	COLUMN 1	COLUMN 2	COLUMN 3	COLUMN 4	COLUMN 5
ROW 1	data	data	data	data	data
ROW 2	data	data	data	data	data
ROW 3	data	data	data	data	data
ROW 4	data	data	data	data	data



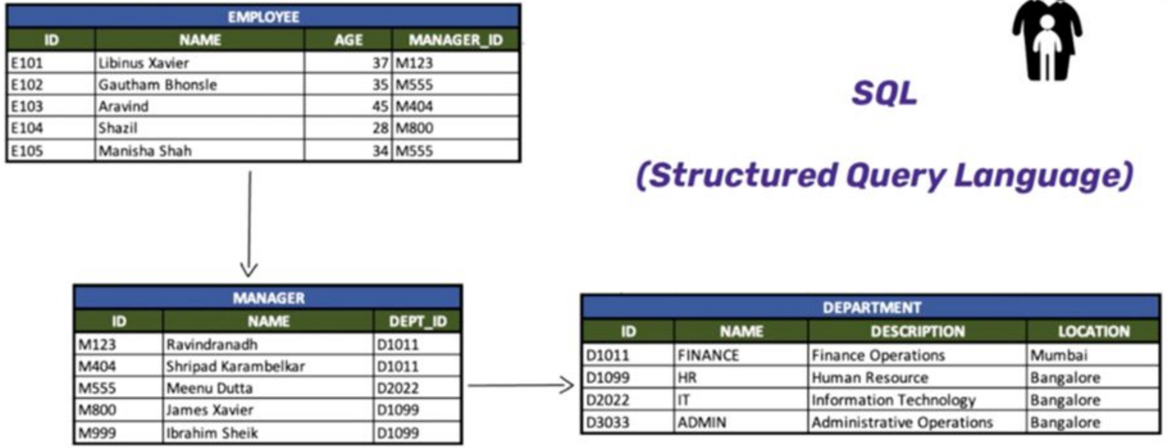
NAME	ID	DATE_OF_BIRTH	ADDRESS	GENDER	PHONE
Aaron Paul	D1	05-Jul-86	Kuala Lumpur	M	60169990102
Lara Croft	D2	01-Oct-98	Bangalore	F	9774755019
Ruth Langmore	D3	23-May-01	Singapore	F	6545459898
Emma Watson	D4				

Relational Database

- As an example, if we consider an **office database**, it may have information related to **Employees, Managers and Departments**. These details are stored in different tables. But these tables will be related to each other through certain columns.
- Here Employee and Manager table is related through the Manager_ID column which is present in both these tables. In a relational database, the foreign key constraint is used to form relations between different tables.
- Similarly Manager and Department table are related through dept id column.
- As you can see, employee and department table are not directly related to each other. But still it is possible to fetch data from employee table based on specific conditions from department table.

Relational Database

Office Database



EMPLOYEE			
ID	NAME	AGE	MANAGER_ID
E101	Libinus Xavier	37	M123
E102	Gautham Bhonsle	35	M555
E103	Aravind	45	M404
E104	Shazil	28	M800
E105	Manisha Shah	34	M555

MANAGER		
ID	NAME	DEPT_ID
M123	Ravindranadh	D1011
M404	Shripad Karambelkar	D1011
M555	Meenu Dutta	D2022
M800	James Xavier	D1099
M999	Ibrahim Sheik	D1099

DEPARTMENT			
ID	NAME	DESCRIPTION	LOCATION
D1011	FINANCE	Finance Operations	Mumbai
D1099	HR	Human Resource	Bangalore
D2022	IT	Information Technology	Bangalore
D3033	ADMIN	Administrative Operations	Bangalore

SQL

(Structured Query Language)

Oracle   Microsoft SQL Server   MySQL   PostgreSQL

TIA-Academy

Relational Database

- This is **how relational database works**, information is scattered across multiple tables which are related to one another. Hence using table relations, **it is possible to retrieve data from different tables**.
- In a relational database, **using DBMS, you can enter commands in specific language to store, retrieve and modify data.**
- **This specific language is SQL (Structured Query Language)**
- SQL is a programming language which follows a standard format for querying data across different relations database.
- Most of the financial institutions such as a Bank or Insurance companies use relational database.
- Examples of relational database are Oracle, MySQL, Microsoft SQL Server, PostgreSQL etc.

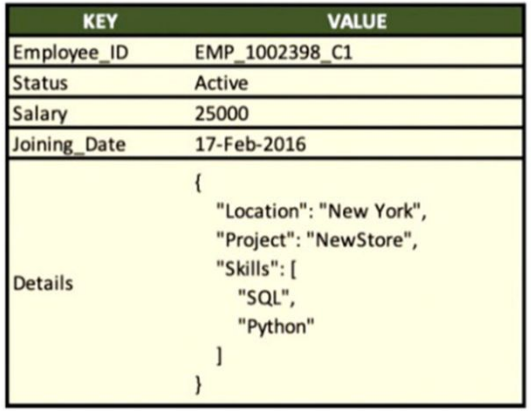
TIA-Academy

Non Relational Database

- When it comes to a non relation database there are several categories of database. Such as:
  - Key Value Store / Key Value Database
  - Document Database
  - Graph Database
  - Wide Column Database
  - Search Engine Database
  - Time Series Database
- Each of these database types store data differently and are useful for managing specific types of data.

Non Relational Database

Key-Value Database :

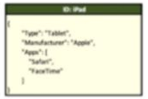


KEY	VALUE
Employee_ID	EMP_1002398_C1
Status	Active
Salary	25000
Joining_Date	17-Feb-2016

Details	{
	"Location": "New York",
	"Project": "NewStore",
	"Skills": [
	"SQL",
	"Python"
	]
	}

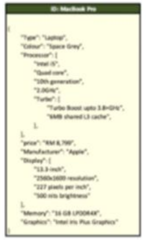
Document Database :

MongoDB, CouchDB etc...



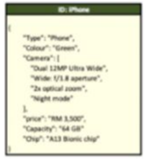
ID: iPad

```
{
    "Type": "Tablet",
    "Manufacturer": "Apple",
    "Apps": [
        "Safari",
        "FaceTime"
    ]
}
```



ID: MacBook Pro

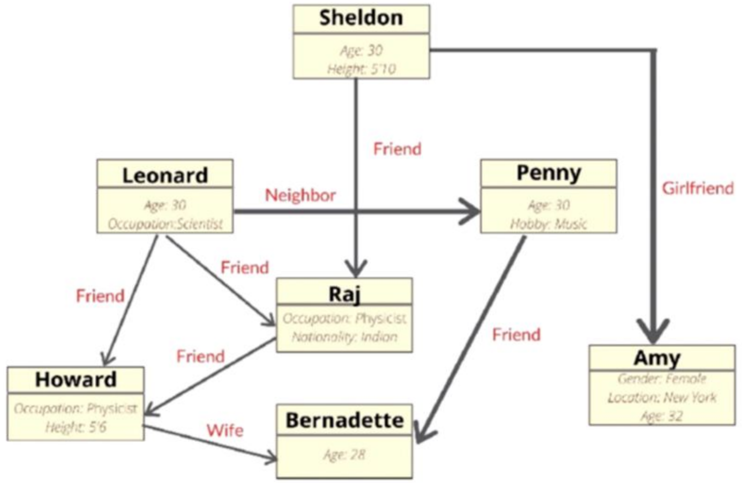
```
{
    "Type": "Laptop",
    "Colour": "Space Grey",
    "Processor": [
        "Intel i5",
        "Quad core",
        "10th generation",
        "2.0GHz",
        "Turbo": [
            "Turbo Boost upto 3.8GHz",
            "6MB shared L3 cache",
        ],
    ],
    "price": "RM 8,799",
    "Manufacturer": "Apple",
    "Display": [
        "13.3-inch",
        "2560x1600 resolution",
        "227 pixels per inch",
        "500 nits brightness"
    ],
    "Memory": "16 GB LPDDR4X",
    "Graphics": "Intel Iris Plus Graphics"
}
```



ID: iPhone

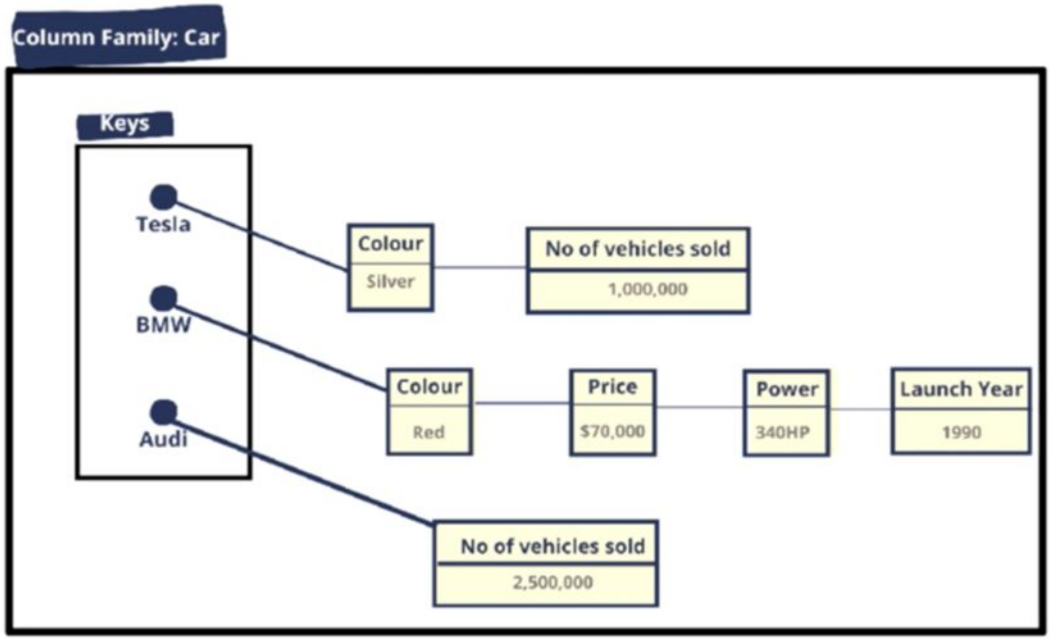
```
{
    "Type": "Phone",
    "Colour": "Green",
    "Camera": [
        "Dual 12MP Ultra Wide",
        "Wide: f/1.8 aperture",
        "2x optical zoom",
        "Night mode"
    ],
    "price": "RM 3,500",
    "Capacity": "64 GB"
    "Chip": "A13 Bionic chip"
}
```

Graph Database:



Non Relational Database

Wide Column Database



Java switch Statement

TIA-Academy

What is MySQL?

- MySQL is currently the most popular database management system software used for managing the relational database.
- It is open-source database software, which is supported by Oracle Company.
- It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database.
- It is commonly used for creating powerful and dynamic server-side or web-based enterprise applications.

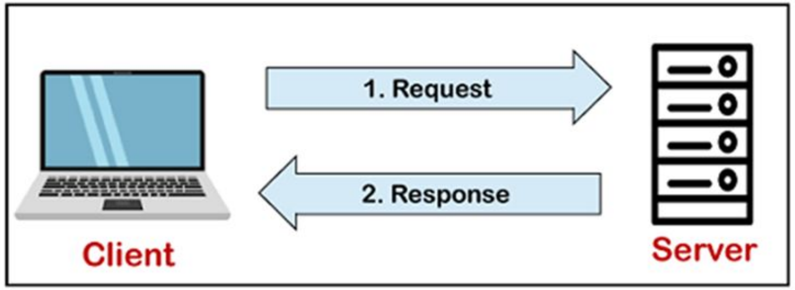
TIA-Academy

What is MySQL?

- MySQL supports many Operating Systems like Windows, Linux, MacOS, etc. with C, C++, and Java languages.
- MySQL is a Relational Database Management System (RDBMS) software that provides many things, which are as follows:
  - It allows us to implement database operations on tables, rows, columns, and indexes.
  - It defines the database relationship in the form of tables (collection of rows and columns), also known as relations.
  - It provides the Referential Integrity between rows or columns of various tables.

TIA-Academy

How MySQL Works?

- MySQL follows the working of Client-Server Architecture. This model is designed for the end-users called clients to access the resources from a central computer known as a server using network services.
- Here, the clients make requests through a graphical user interface (GUI), and the server will give the desired output as soon as the instructions are matched.
- 

same as the client-server model.

TIA-Academy

How MySQL Works?

- The core of the MySQL database is the MySQL Server. This server is available as a separate program and responsible for handling all the database instructions, statements, or commands. The working of MySQL database with MySQL Server are as follows:
  1. MySQL creates a database that allows you to build many tables to store and manipulate data and defining the relationship between each table.
  2. Clients make requests through the GUI screen or command prompt by using specific SQL expressions on MySQL.
  3. Finally, the server application will respond with the requested expressions and produce the desired result on the client-side.

TIA-Academy

How MySQL Works?

- A client can use any MySQL GUI. But, it is making sure that your GUI should be lighter and user-friendly to make your data management activities faster and easier.
- Some of the most widely used MySQL GUIs are **MySQL Workbench, SequelPro, DBVisualizer, and the Navicat DB Admin Tool.**
- Some GUIs are commercial, while some are free with limited functionality, and some are only compatible with MacOS. Thus, you can choose the GUI according to your needs.

TIA-Academy

MySQL Connection

- A connection is a computer science facility that allows the user to connect with the database server software.
- **A user can connect with the database server, whether on the same machine or *remote locations*.**
- Therefore, if we want to work with the database server to send commands and receive answers in the form of a result set, we need connections.

MySQL Connection

MySQL provides various ways to connect with the database server. **Once we have installed the MySQL server, we can connect it using any of the client programs that are listed below**:

1. Command-line client
2. **MySQL Workbench**

TIA-Academy

MySQL Server Connection Using CLI

- MySQL command-line client program provides interaction with the database server in an interactive and non-interactive mode.
- We can see this program in the **bin directory of the MySQL's installation folder**.
- We can open the MySQL command prompt by navigating to the bin directory of the MySQL's installation folder and type:

```
MySQL
```

TIA-Academy

MySQL Server Connection Using CLI

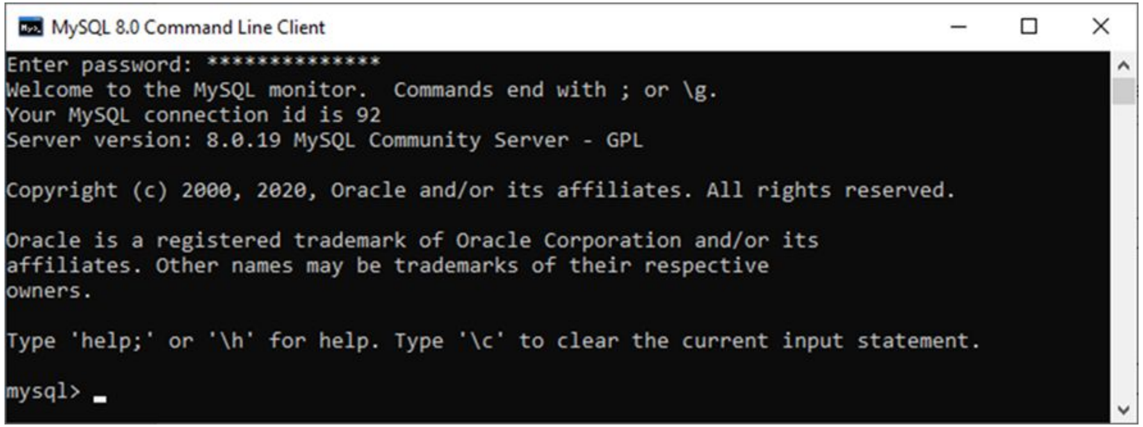
- If we find the MySQL program in the **PATH**, we can use the below command to connect to the MySQL Server:

```
mysql -u root -p
```

- In the syntax, the **-u root indicates** that we will connect to the MySQL server using the root user account and **-p** instructs MySQL to ask for a password.

MySQL Server Connection Using CLI

- Next, we need to type the password for the root user account and press **Enter**. If everything is correct, it should give the screen as follows:



MySQL 8.0 Command Line Client

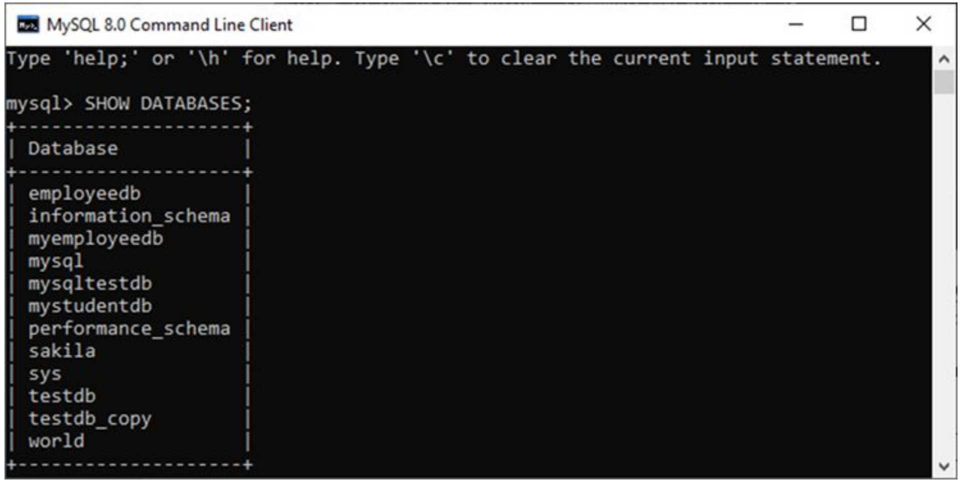
```
Enter password: **************
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 92
Server version: 8.0.19 MySQL Community Server - GPL

Copyright (c) 2000, 2020, 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> _
```

- This screen indicates that we have successfully connected with the MySQL database server, where we can send commands and receive answers in the form of a result set.

MySQL Server Connection Using CLI

- Suppose we want to display all databases available in the current server; we can use the command as follows:

```
mysql> SHOW DATABASES;
```



```
MySQL 8.0 Command Line Client
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| employeedb         |
| information_schema |
| myemployeedb       |
| mysql              |
| mysqltestdb        |
| mystudentdb        |
| performance_schema |
| sakila             |
| sys                |
| testdb             |
| testdb_copy        |
| world              |
+--------------------+
```

- If you want **to disconnect the opened MySQL database server**, you need to use the exit command.

```
mysql> EXIT;
```

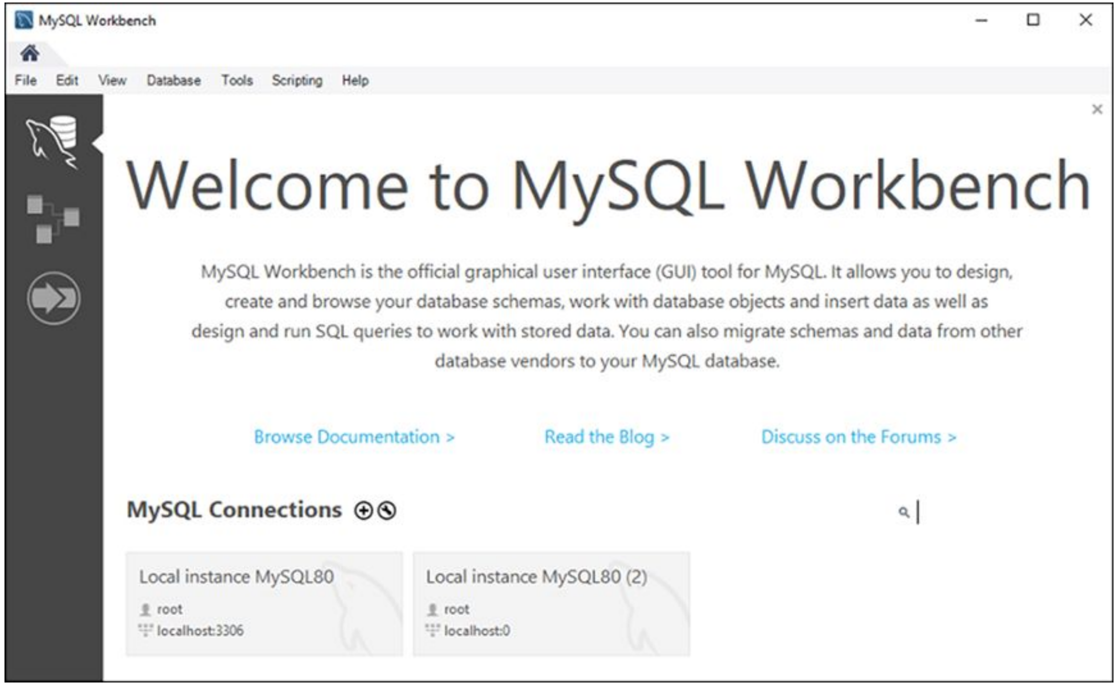
TIA-Academy

MYSQL WORKBENCH

TIA-Academy

Connect to Database Server Using MySQL Workbench

Step 1: Launch the MySQL Workbench. We should get the following screen:



Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

Browse Documentation >

Read the Blog >

Discuss on the Forums >

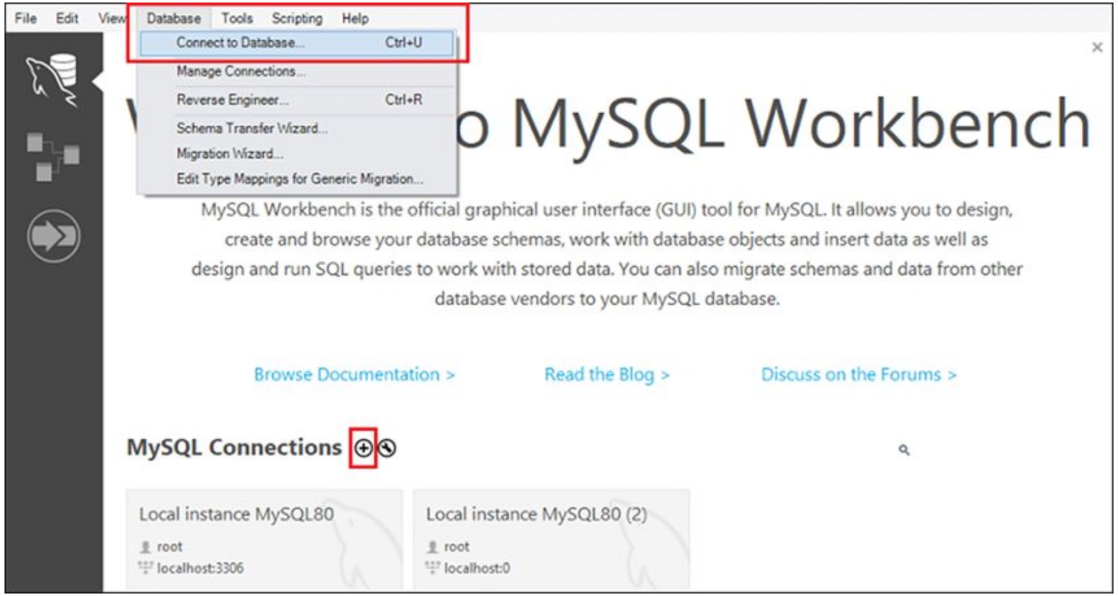
MySQL Connections

Local instance MySQL80
root
localhost:3306

Local instance MySQL80 (2)
root
localhost:0

TIA-Academy

Connect to Database Server Using MySQL Workbench



File Edit View Database Tools Scripting Help
Connect to Database... Ctrl+U
Manage Connections...
Reverse Engineer... Ctrl+R
Schema Transfer Wizard...
Migration Wizard...
Edit Type Mappings for Generic Migration...
o MySQL Workbench
Browse Documentation >
Read the Blog >
Discuss on the Forums >
MySQL Connections
Local instance MySQL80
root
localhost:3306
Local instance MySQL80 (2)
root
localhost:0

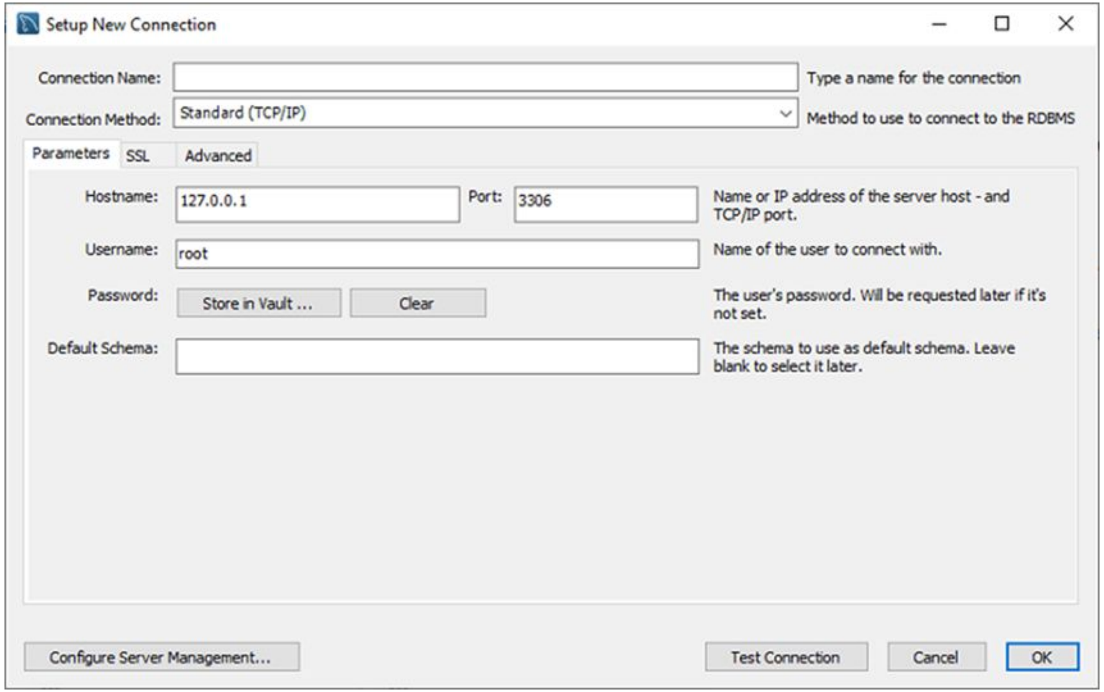
MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

Step 2: Navigate to the menu bar, click on the **'Database'** and choose **Connect to Database** option or press the **CTRL+U** command. We can also connect with the database server by just clicking the **plus (+) button** located next to the MySQL Connections.

TIA-Academy

Connect to Database Server Using MySQL Workbench

Step 3: After choosing any of the options, we will get the below screen:



Setup New Connection

Connection Name: Type a name for the connection

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Parameters | SSL | Advanced

Hostname: 127.0.0.1 Port: 3306 — Name or IP address of the server host - and TCP/IP port.

Username: root — Name of the user to connect with.

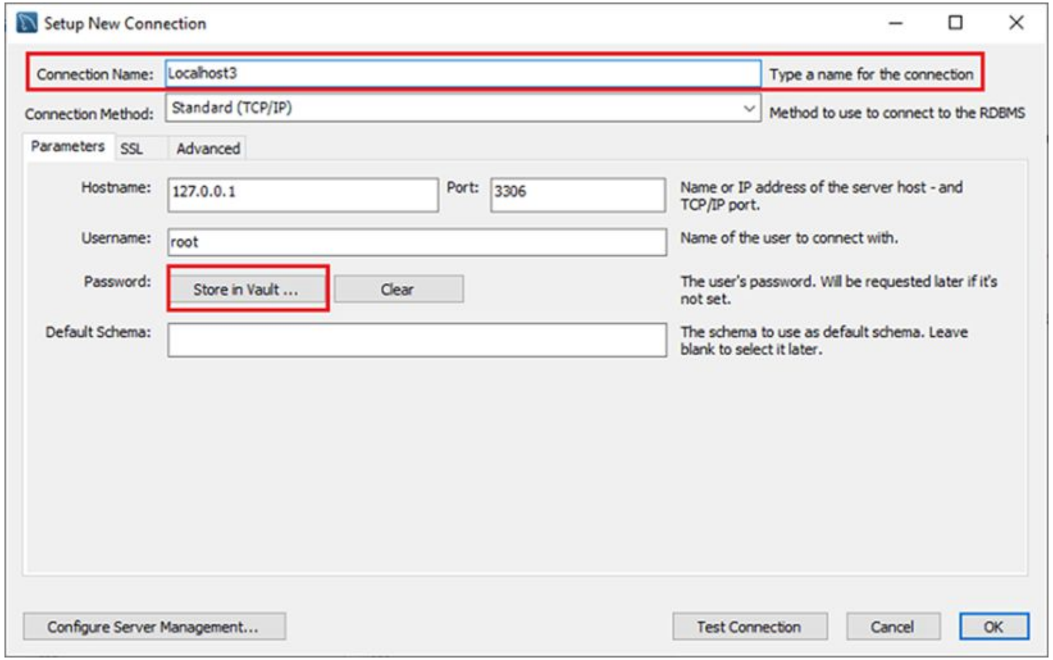
Password: Store in Vault ... Clear — The user's password. Will be requested later if it's not set.

Default Schema: — The schema to use as default schema. Leave blank to select it later.

Configure Server Management... Test Connection Cancel OK

TIA-Academy

Connect to Database Server Using MySQL Workbench


Setup New Connection

Connection Name: Localhost3 — Type a name for the connection

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname: 127.0.0.1 Port: 3306 — Name or IP address of the server host - and TCP/IP port.

Username: root — Name of the user to connect with.

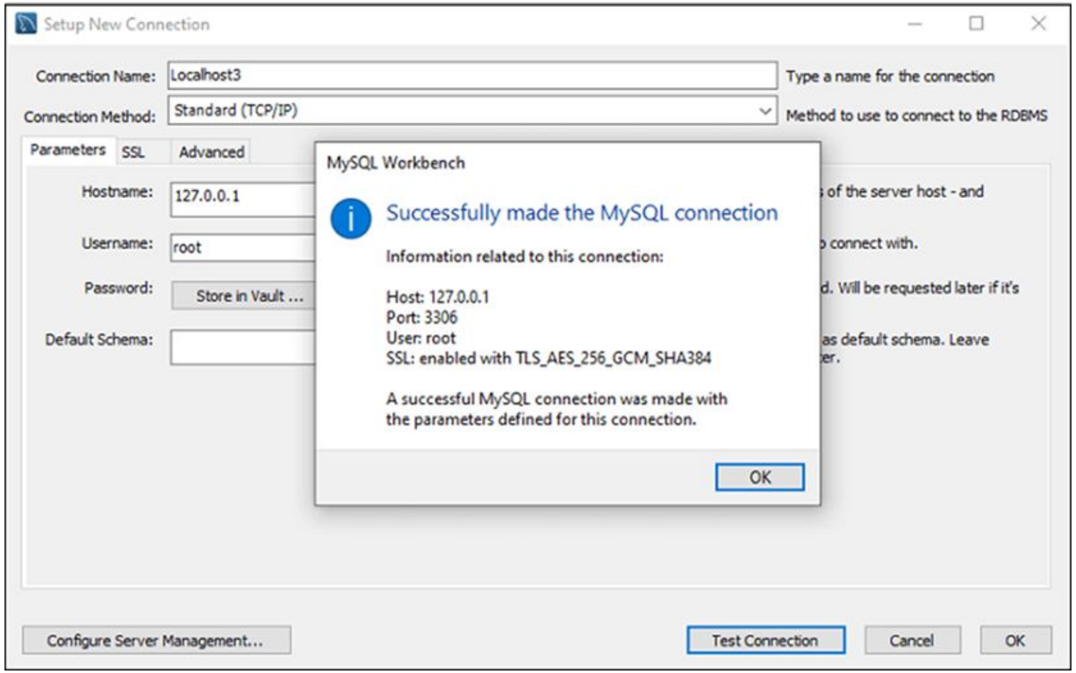
Password: — The user's password. Will be requested later if it's not set.

Default Schema: — The schema to use as default schema. Leave blank to select it later.

Step 4: Fill the box to create a connection, such as **connection name** and **username**, whatever you want. By default, the username is the **root**, but we can also change it with a different username in the Username textbox. After filling all boxes, click the **Store in Vault ... button** to write the password for the given user account.

TIA-Academy

Connect to Database Server Using MySQL Workbench



Connection Name: Localhost3 — Type a name for the connection

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname: 127.0.0.1 — of the server host - and

Username: root — connect with.

Password: Store in Vault ... — d. Will be requested later if it's

Default Schema: — as default schema. Leave er.

MySQL Workbench

Successfully made the MySQL connection

Information related to this connection:

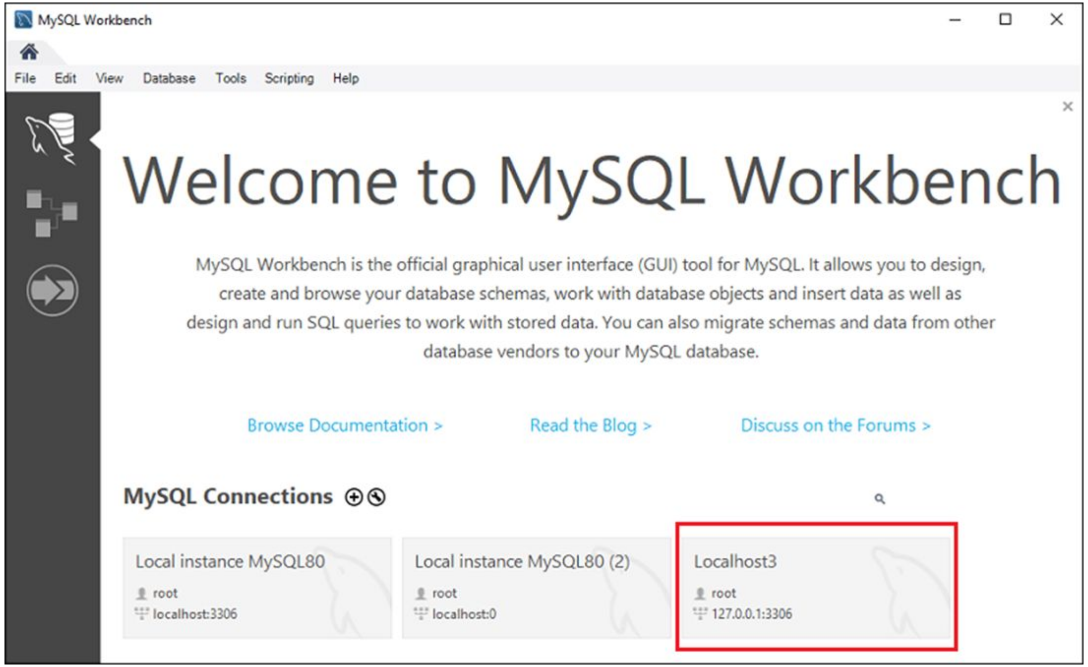
Host: 127.0.0.1
Port: 3306
User: root
SSL: enabled with TLS_AES_256_GCM_SHA384

A successful MySQL connection was made with the parameters defined for this connection.

OK

Step 6: After entering all the details, click on the **Test Connection** to test the database connectivity is successful or not. If the connection is successful, click on the **OK** button.

Connect to Database Server Using MySQL Workbench



Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

Browse Documentation > Read the Blog > Discuss on the Forums >

MySQL Connections

Local instance MySQL80
root
localhost:3306

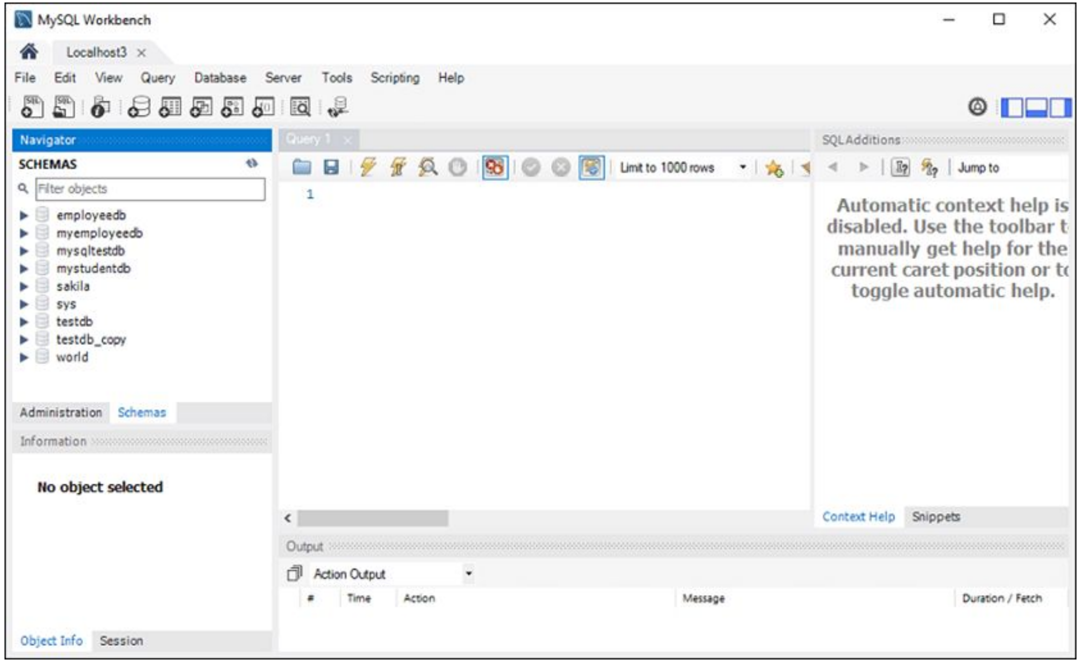
Local instance MySQL80 (2)
root
localhost:0

Localhost3
root
127.0.0.1:3306

Step 7: Again, click on the **OK** button for saving connection setup. After finishing all the setup, we can see this connection under **MySQL Connections** for connecting to the MySQL database server. See the side output where we have **Localhost3** connection name:

TIA-Academy

Connect to Database Server Using MySQL Workbench



Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or to toggle automatic help.

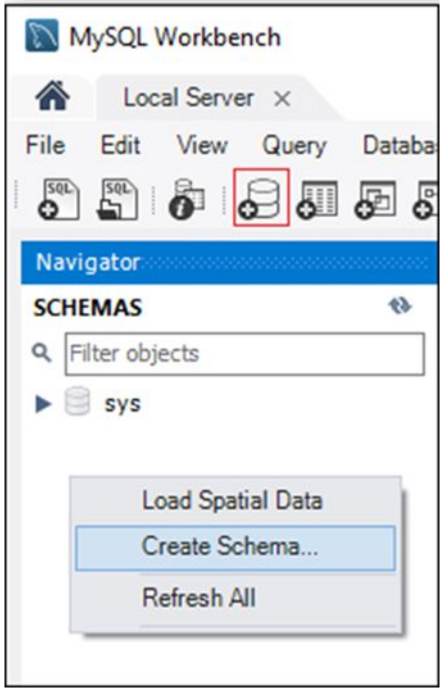
Step 8: Now, we can click this newly created connection that displays the current schemas and a pane for entering queries:

TIA-Academy

MySQL Workbench Create, Alter, Drop Database

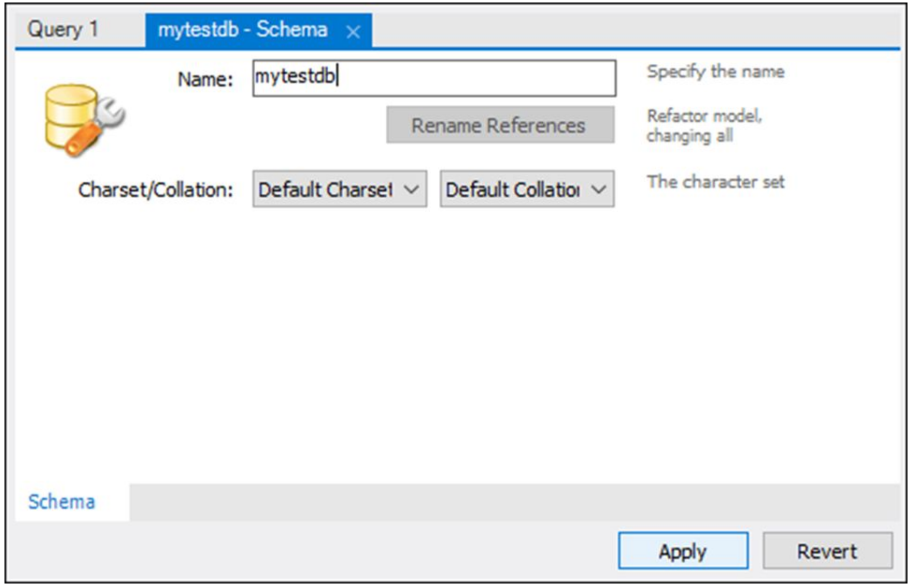
Create Database

1. Open the MySQL Workbench and logged in using username and password. Then, go to the Navigation tab and click on the **Schema menu**. Here, you can see all the previously created databases.
2. If you want to create a new database, right-click under the Schema menu and select **Create Schema** or click the database icon (red rectangle), as shown in the following screen.



TIA-Academy

MySQL Workbench Create, Alter, Drop Database



Name: mytestdb

Specify the name

Refactor model, changing all

Charset/Collation: Default Charset | Default Collation

The character set

3. The new Schema window screen open.

   Enter the new database name (for example, mytestdb) and use default **Collation**. Collation is used to store specific data characters, mainly useful for storing foreign languages. Now, click on the Apply button as shown in the screen below:

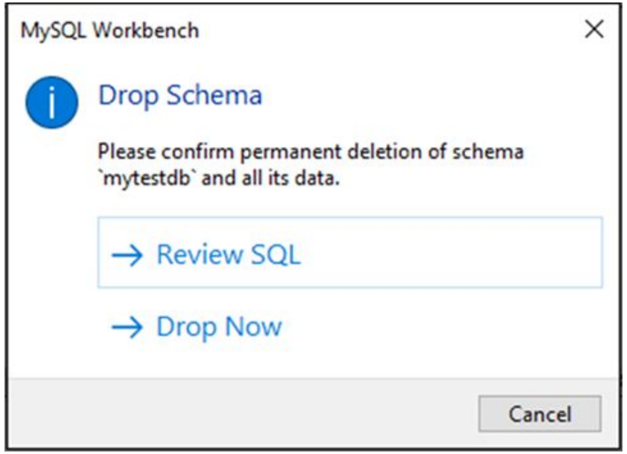
MySQL Workbench Create, Alter, Drop Database

4. A new popup window appears, click Apply->Finish button to create a new database.
5. After the successful creation of the database, you can see this new database in the Schema menu. If you do not see this, click on the refresh icon into the Schema menu.
6. If you want to see more information about the database, select mytestdb database, and click on the 'i' icon. The information window displays several options, like Table, Column, Functions, Users, and many more.
7. **MySQL Workbench does not provide an option to rename the database** name, but we can create, update, and delete the table and data rows from the database.

MySQL Workbench Create, Alter, Drop Database

Drop Database

1. To delete a database, you need to choose the database, right-click on it, and select the **Drop Schema** option. The following screen appears:
2. Select **Drop Now** option in the popup window and the database including table, data rows will be deleted from the database Server.


MySQL Workbench

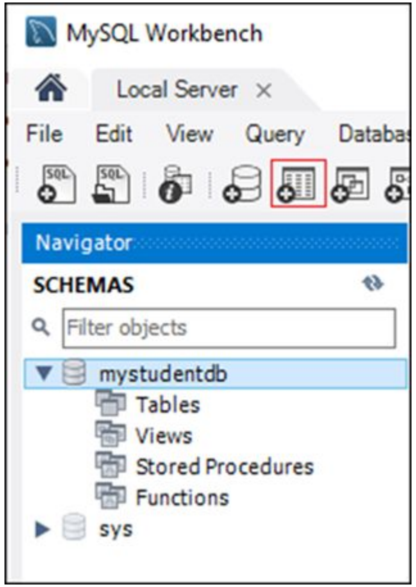
Drop Schema

Please confirm permanent deletion of schema `mytestdb` and all its data.

MySQL Workbench Create, Alter, Drop Table

Create Table

1. Open the MySQL Workbench and logged in using username and password. Then, go to the Navigation tab and click on the Schema menu. Here, you can see all the previously created databases. You can also create a new database.
2. Select the newly created database, double click on it, and you will get the sub-menu under the database. The sub-menu under the database are Tables, Views, Functions, and Stored Procedures, as shown in the below screen.



MySQL Workbench Create, Alter, Drop Table

3. Select Tables sub-menu, right-click on it and select **Create Table** option. You can also click on create a new table icon (shown in red rectangle) to create a table.
4. On the new table screen, you need to fill all the details to create a table. Here, we are going to enter the table name (for example, student) and use default collation and engine.

MySQL Workbench Create, Alter, Drop Table

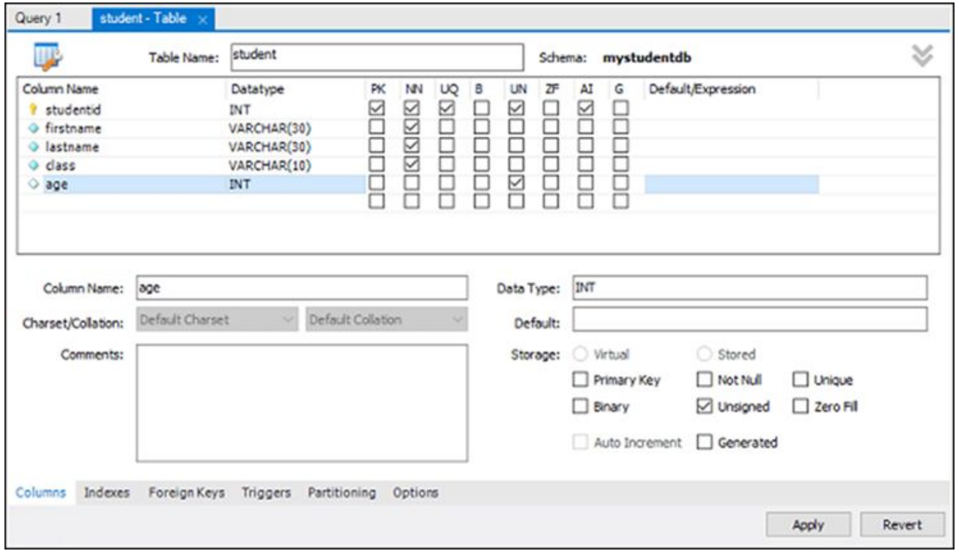


Table Name: student Schema: mystudentdb

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
studentid	INT	✓	✓	✓		✓		✓		
firstname	VARCHAR(30)		✓							
lastname	VARCHAR(30)		✓							
class	VARCHAR(10)		✓							
age	INT					✓				

Column Name: age Data Type: INT

Charset/Collation: Default Charset Default Collation Default:

Comments: Storage: Virtual Stored

Primary Key Not Null Unique

Binary [✓ Unsigned] Zero Fill

Auto Increment Generated

5. Click inside the middle window and fill the column details. Here, the column name contains many attributes such as Primary Key(PK), Not Null (NN), Unique Index (UI), Binary(B), Unsigned Data type(UN), Auto Incremental (AI), etc. The following screen explains it more clearly. After filling all the details, click on the **Apply** button

MySQL Workbench Create, Alter, Drop Table

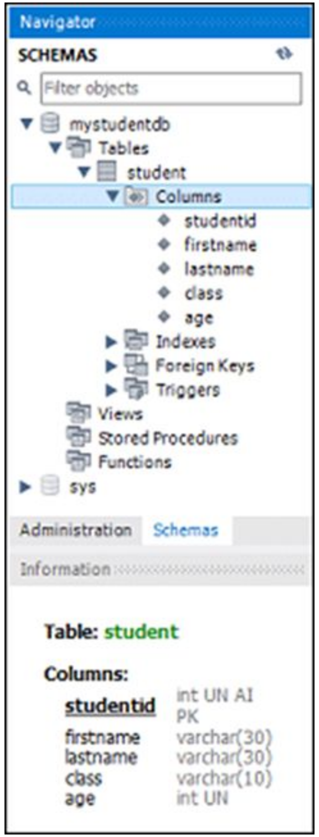


Table: student

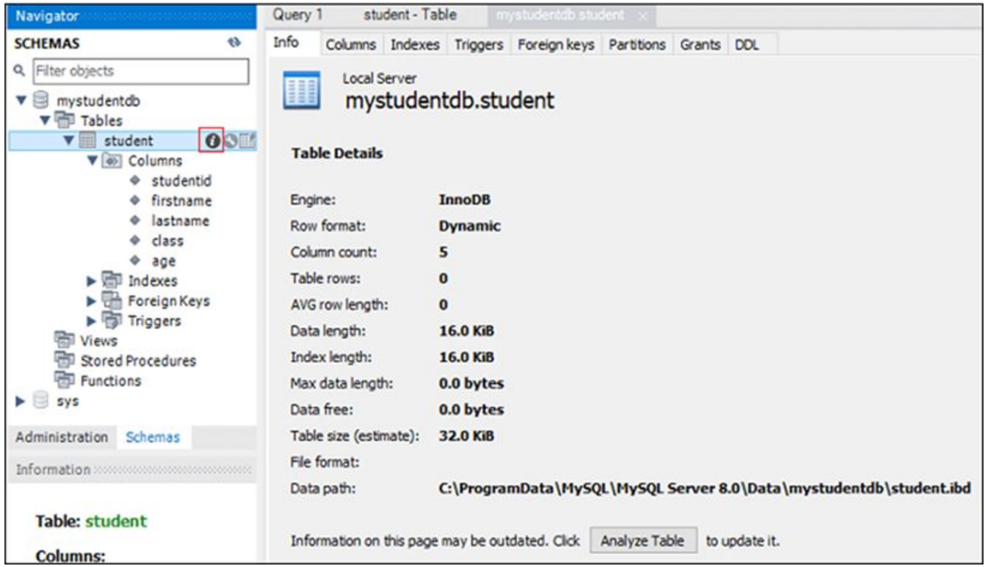
Columns:

studentid	int UN AI PK
firstname	varchar(30)
lastname	varchar(30)
class	varchar(10)
age	int UN

6. As soon as you click on the Apply button, it will open the SQL statement window. Again, click on the Apply button to execute the statement and Finish button to save the change.
7. Now, go to the Schema menu and select the database which contains the newly created table, as shown in the screen.

TIA-Academy

MySQL Workbench Create, Alter, Drop Table



Local Server
mystudentdb.student

Table Details

Engine:	InnoDB
Row format:	Dynamic
Column count:	5
Table rows:	0
AVG row length:	0
Data length:	16.0 KiB
Index length:	16.0 KiB
Max data length:	0.0 bytes
Data free:	0.0 bytes
Table size (estimate):	32.0 KiB
File format:	
Data path:	C:\ProgramData\MySQL\MySQL Server 8.0\Data\mystudentdb\student.ibd

Information on this page may be outdated. Click Analyze Table to update it.

Table: student

Columns:

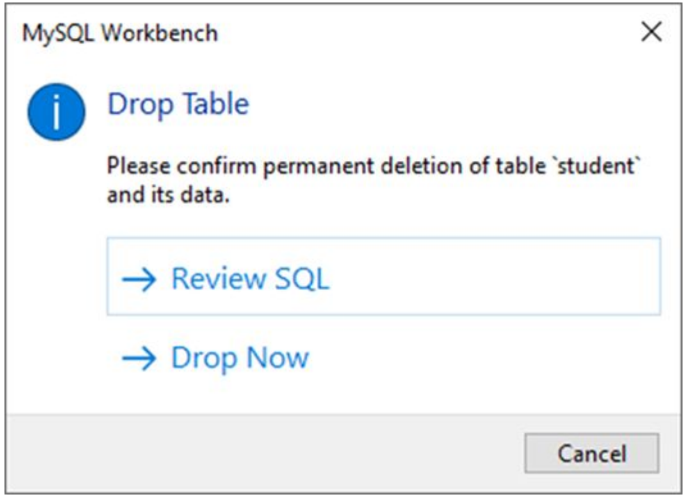
Alter Table

1. Select the table you want to modify, click on the 'i' icon, and you will get the following screen.
2. In the screen, you can modify the column name, data type, and other table settings.

MySQL Workbench Create, Alter, Drop Table

Drop a Table

1. To delete a table, you need to choose the table, right-click on it, and select the Drop Table option. The following screen appears:
2. Select **Drop Now** option in the popup window to delete the table from the database instantly.


MySQL Workbench

Drop Table

Please confirm permanent deletion of table `student` and its data.

TIA-Academy

MYSQL QUERIES

MySQL INSERT Statement

MySQL INSERT statement is used to store or add data in MySQL table within the database. We can perform insertion of records in two ways using a single query in MySQL:

1. Insert record in a single row
2. Insert record in multiple rows

MySQL INSERT Statement

- The below is generic syntax of **SQL INSERT INTO** command to insert a single record in MySQL table:

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

- In the above syntax, we first have to specify the table name and list of comma-separated columns. Second, we provide the list of values corresponding to columns name after the VALUES clause.

TIA-Academy

MySQL INSERT Statement

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

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

- In the above syntax, all rows should be separated by commas in the value fields.

TIA-Academy

MySQL INSERT Example

- Let us understand how INSERT statements work in MySQL with the help of multiple examples.
- First, create a table "**People**" in the database using the following command:

```
CREATE TABLE mystudentdb.people(
    id int NOT NULL AUTO_INCREMENT,
    name varchar(45) NOT NULL,
    occupation varchar(35) NOT NULL,
    age int,
    PRIMARY KEY (id)
);
```

TIA-Academy

MySQL INSERT Example

1. If we want to store single records for all fields, use the syntax as follows:

```
INSERT INTO people (id, name, occupation, age)
VALUES (101, 'Peter', 'Engineer', 32);
```

1. If we want to store multiple records, use the following statements where we can either specify all field names or don't specify any field.

```
INSERT INTO people VALUES
(102, 'Joseph', 'Developer', 30),
(103, 'Mike', 'Leader', 28),
(104, 'Stephen', 'Scientist', 45);
```

TIA-Academy

MySQL INSERT Example

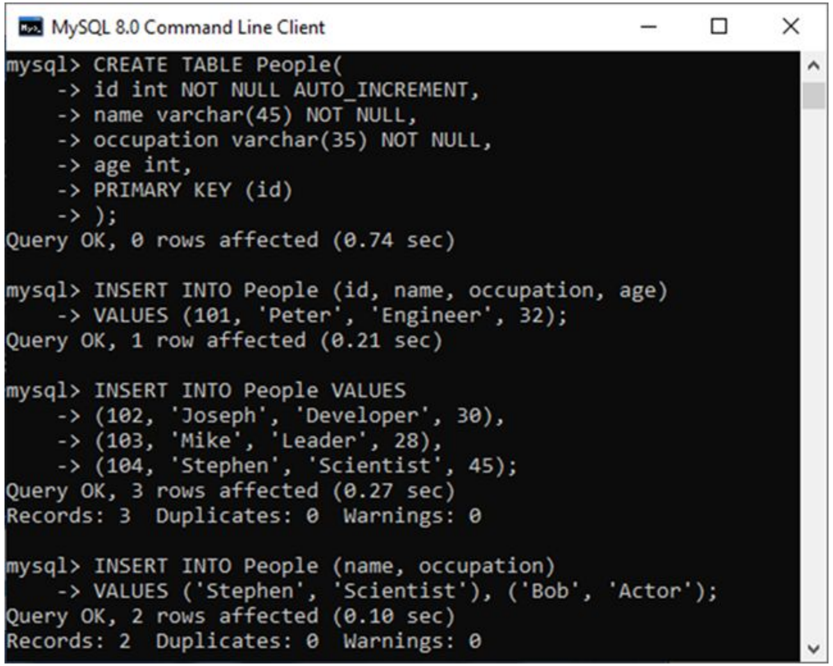
3. If we want to store records without giving all fields, we use the following **partial field** statements. In such case, it is mandatory to specify field names.

```
INSERT INTO People (name, occupation)
VALUES ('Stephen', 'Scientist'), ('Bob', 'Actor');
```

TIA-Academy

MySQL INSERT Example

- In the below output, we can see that all INSERT statements have successfully executed and stored the value in a table correctly.


MySQL 8.0 Command Line Client

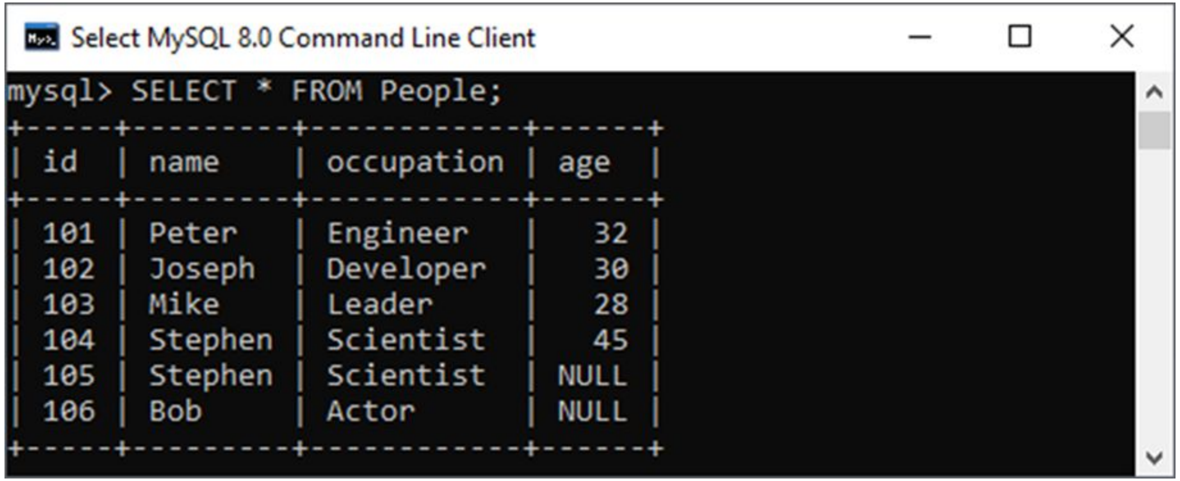
```
mysql> CREATE TABLE People(
    -> id int NOT NULL AUTO_INCREMENT,
    -> name varchar(45) NOT NULL,
    -> occupation varchar(35) NOT NULL,
    -> age int,
    -> PRIMARY KEY (id)
    -> );
Query OK, 0 rows affected (0.74 sec)

mysql> INSERT INTO People (id, name, occupation, age)
    -> VALUES (101, 'Peter', 'Engineer', 32);
Query OK, 1 row affected (0.21 sec)

mysql> INSERT INTO People VALUES
    -> (102, 'Joseph', 'Developer', 30),
    -> (103, 'Mike', 'Leader', 28),
    -> (104, 'Stephen', 'Scientist', 45);
Query OK, 3 rows affected (0.27 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO People (name, occupation)
    -> VALUES ('Stephen', 'Scientist'), ('Bob', 'Actor');
Query OK, 2 rows affected (0.10 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

TIA-Academy

MySQL INSERT Example

- We can use the below syntax to show the records of the **People** table:

```
mysql> SELECT * FROM People;
```



Select MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM People;
+-----+---------+------------+------+
| id  | name    | occupation | age  |
+-----+---------+------------+------+
| 101 | Peter   | Engineer   |   32 |
| 102 | Joseph  | Developer  |   30 |
| 103 | Mike    | Leader     |   28 |
| 104 | Stephen | Scientist  |   45 |
| 105 | Stephen | Scientist  | NULL |
| 106 | Bob     | Actor      | NULL |
+-----+---------+------------+------+
```

TIA-Academy

Inserting Date in MySQL Table:

- We can also use the INSERT STATEMENT to add the date in MySQL table. MySQL provides several data types for storing dates such as DATE, TIMESTAMP, DATETIME, and YEAR. The **default forma**t of the date in MySQL is **YYYY-MM-DD**.
- This format has the below descriptions:
  - **YYYY:** It represents the four-digit year, like 2020.
  - **MM:** It represents the two-digit month, like 01, 02, 03, and 12.
  - **DD:** It represents the two-digit day, like 01, 02, 03, and 31.

TIA-Academy

Inserting Date in MySQL Table:

- Following is the basic syntax to insert date in MySQL table yyyy-mm-dd:

```
INSERT INTO table_name (column_name, column_date)
VALUES ('DATE: Manual Date', '2008-7-04');
```

- If we want to insert a date in the mm/dd/yyyy format, it is required to use the below statement:

```
INSERT INTO table_name
VALUES (STR_TO_DATE(date_value, format_specifier));
```

TIA-Academy

MySQL UPDATE Query

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

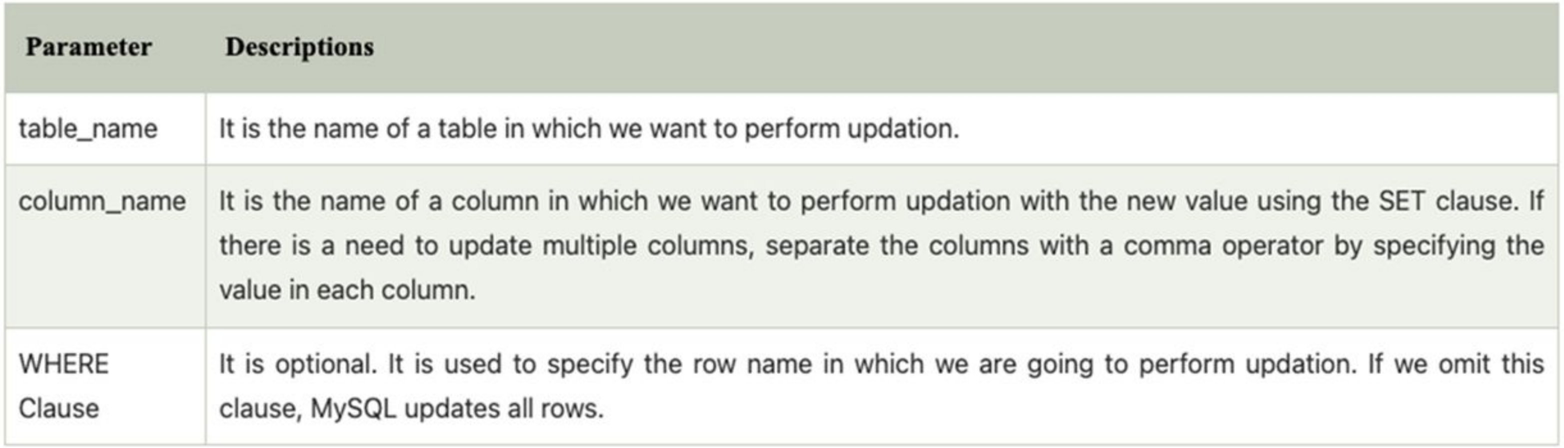
TIA-Academy

MySQL UPDATE Query

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

```
UPDATE table_name
SET column_name1 = new-value1,
        column_name2=new-value2, ...

[WHERE Clause]
```

MySQL UPDATE Query



Parameter	Descriptions
table_name	It is the name of a table in which we want to perform updation.
column_name	It is the name of a column in which we want to perform updation with the new value using the SET clause. If there is a need to update multiple columns, separate the columns with a comma operator by specifying the value in each column.
WHERE Clause	It is optional. It is used to specify the row name in which we are going to perform updation. If we omit this clause, MySQL updates all rows.

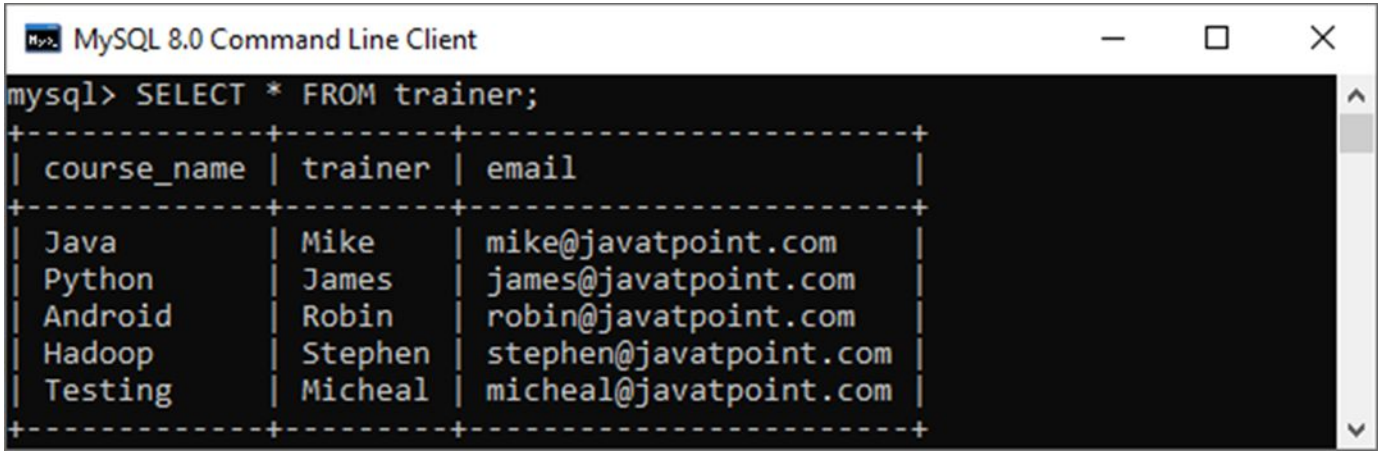
MySQL UPDATE Query

- This statement can update values in a single table at a time.
- We can update single or multiple columns altogether with this statement.
- Any condition can be specified by using the WHERE clause.
- WHERE clause is very important because sometimes we want to update only a single row, and if we omit this clause, it accidentally updates all rows of the table.

TIA-Academy

MySQL UPDATE Query

- Let us understand the UPDATE statement with the help of various examples. Suppose we have a table **"trainer"** within the **"testdb"** database. We are going to update the data within the "trainer" table.



MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM trainer;
+-------------+---------+------------------------+
| course_name | trainer | email                  |
+-------------+---------+------------------------+
| Java        | Mike    | mike@javatpoint.com    |
| Python      | James   | james@javatpoint.com   |
| Android     | Robin   | robin@javatpoint.com   |
| Hadoop      | Stephen | stephen@javatpoint.com |
| Testing     | Micheal | micheal@javatpoint.com |
+-------------+---------+------------------------+
```

MySQL UPDATE Query

Update Single Column

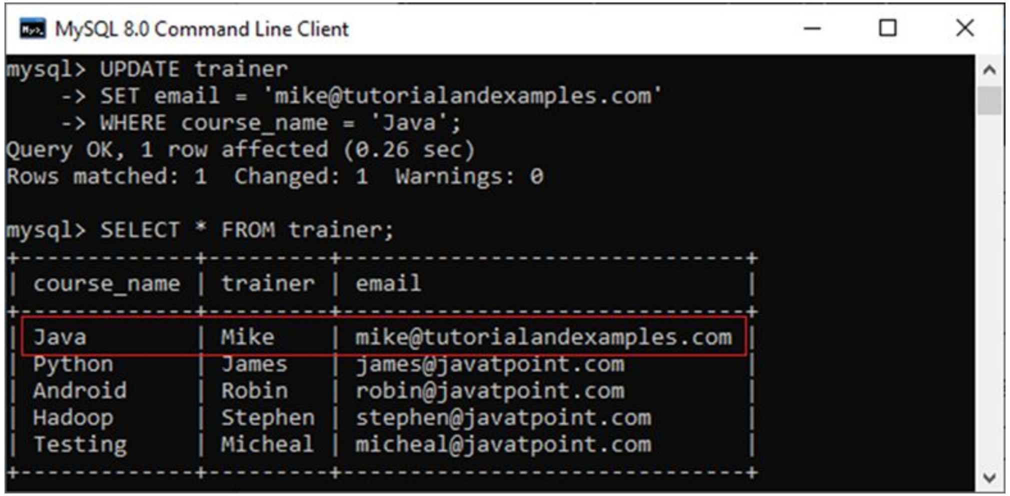
- This query will update the **email id of Java** course with the new id as follows:

```
UPDATE trainer
SET email = 'mike@tutorialandexamples.com'
WHERE course_name = 'Java';
```

MySQL UPDATE Query

- After successful execution, we will verify the table using the below statement:

```
SELECT * FROM trainer;
```



MySQL 8.0 Command Line Client

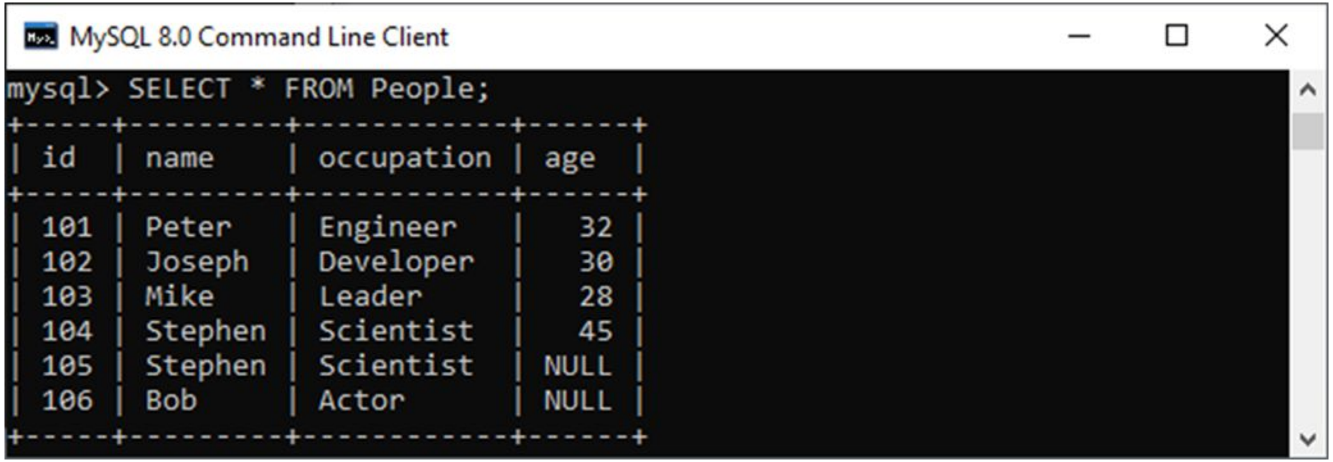
```
mysql> UPDATE trainer
    -> SET email = 'mike@tutorialandexamples.com'
    -> WHERE course_name = 'Java';
Query OK, 1 row affected (0.26 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM trainer;
+-------------+---------+------------------------------+
| course_name | trainer | email                        |
+-------------+---------+------------------------------+
| Java        | Mike    | mike@tutorialandexamples.com |
| Python      | James   | james@javatpoint.com         |
| Android     | Robin   | robin@javatpoint.com         |
| Hadoop      | Stephen | stephen@javatpoint.com       |
| Testing     | Micheal | micheal@javatpoint.com       |
+-------------+---------+------------------------------+
```

TIA-Academy

MySQL UPDATE Query

Update Multiple Column

- The UPDATE statement can also be used to update multiple columns by specifying a comma-separated list of columns. Suppose we have a table as :



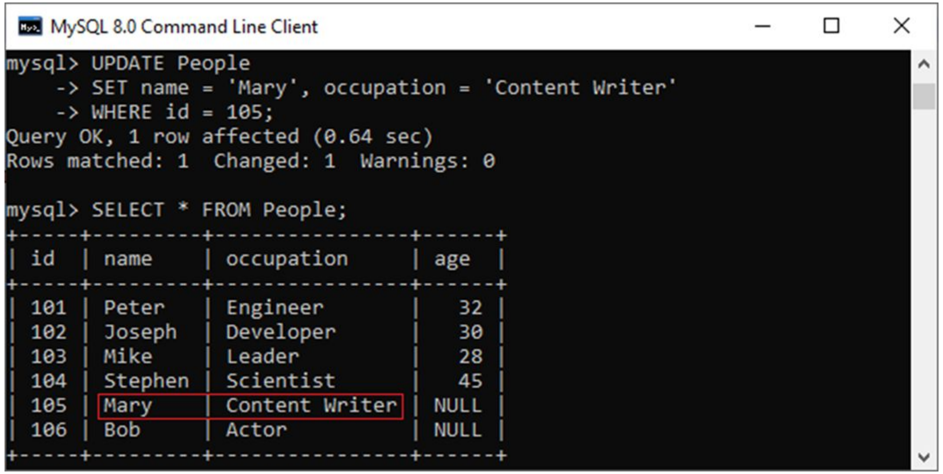
```
mysql> SELECT * FROM People;
+-----+---------+------------+------+
| id  | name    | occupation | age  |
+-----+---------+------------+------+
| 101 | Peter   | Engineer   |   32 |
| 102 | Joseph  | Developer  |   30 |
| 103 | Mike    | Leader     |   28 |
| 104 | Stephen | Scientist  |   45 |
| 105 | Stephen | Scientist  | NULL |
| 106 | Bob     | Actor      | NULL |
+-----+---------+------------+------+
```

TIA-Academy

MySQL UPDATE Query

- This statement explains will update the **name** and **occupation** whose **id = 105** in the **People** table as follows:

```
UPDATE People
SET name = 'Mary',
occupation = 'Content Writer'
WHERE id = 105;
```



MySQL 8.0 Command Line Client

```
mysql> UPDATE People
    -> SET name = 'Mary', occupation = 'Content Writer'
    -> WHERE id = 105;
Query OK, 1 row affected (0.64 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM People;
+-----+---------+----------------+------+
| id  | name    | occupation     | age  |
+-----+---------+----------------+------+
| 101 | Peter   | Engineer       |   32 |
| 102 | Joseph  | Developer      |   30 |
| 103 | Mike    | Leader         |   28 |
| 104 | Stephen | Scientist      |   45 |
| 105 | Mary    | Content Writer | NULL |
| 106 | Bob     | Actor          | NULL |
+-----+---------+----------------+------+
```

TIA-Academy

MySQL Delete Statement

- MySQL DELETE statement is used to remove records from the MySQL table that is no longer required in the database.
- **This query in MySQL deletes a full row from the table and produces the count of deleted rows**.
- It also allows us to delete more than one record from the table within a single query, which is beneficial while removing large numbers of records from a table.
- By using the delete statement, we can also remove data based on conditions.

TIA-Academy

MySQL Delete Statement

- **Once we delete the records using this query, we cannot recover it**.
- Therefore before deleting any records from the table, it is recommended to **create a backup of your database**.
- The database backups allow us to restore the data whenever we need it in the future.

MySQL Delete Statement

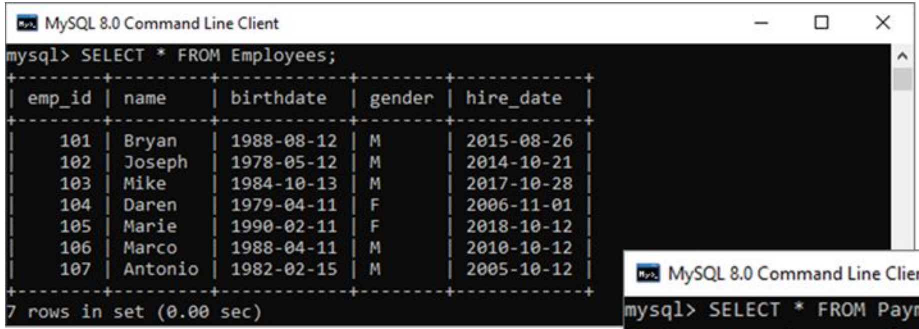
```
DELETE FROM table_name WHERE condition;
```

- In the above statement, we have to first specify the table name from which we want to delete data.
- Second, we have to specify the condition to delete records in the WHERE clause, which is optional. If we omit the WHERE clause into the statement, this query will **remove whole records from the database table**.
- If we want to delete records from multiple tables using a single DELETE query, we must add the **JOIN clause** with the DELETE statement.
- If we want to delete all records from a table without knowing the count of deleted rows, we must use the **TRUNCATE TABLE** statement that gives better performance.

TIA-Academy

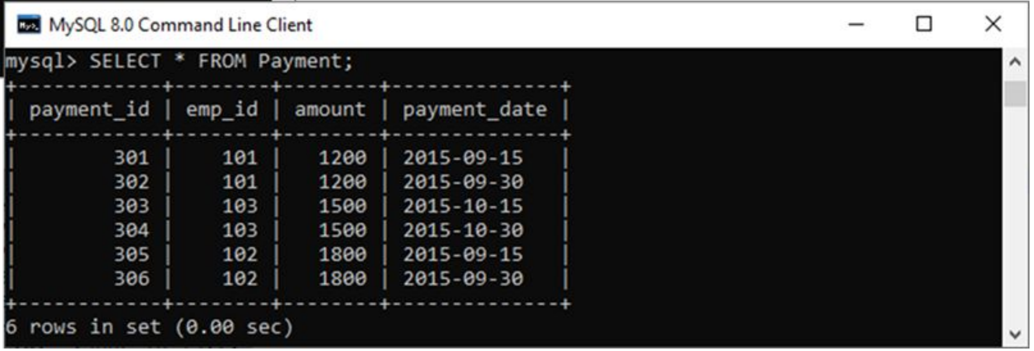
MySQL Delete Statement

- Here, we are going to use the **"Employees"** and **"Payment"** tables for the demonstration of the DELETE statement. Suppose the Employees and Payment tables contain the following data:



MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM Employees;
+--------+---------+------------+--------+------------+
| emp_id | name    | birthdate  | gender | hire_date  |
+--------+---------+------------+--------+------------+
|    101 | Bryan   | 1988-08-12 | M      | 2015-08-26 |
|    102 | Joseph  | 1978-05-12 | M      | 2014-10-21 |
|    103 | Mike    | 1984-10-13 | M      | 2017-10-28 |
|    104 | Daren   | 1979-04-11 | F      | 2006-11-01 |
|    105 | Marie   | 1990-02-11 | F      | 2018-10-12 |
|    106 | Marco   | 1988-04-11 | M      | 2010-10-12 |
|    107 | Antonio | 1982-02-15 | M      | 2005-10-12 |
+--------+---------+------------+--------+------------+
7 rows in set (0.00 sec)
```



MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM Payment;
+------------+--------+--------+--------------+
| payment_id | emp_id | amount | payment_date |
+------------+--------+--------+--------------+
|        301 |    101 |   1200 | 2015-09-15   |
|        302 |    101 |   1200 | 2015-09-30   |
|        303 |    103 |   1500 | 2015-10-15   |
|        304 |    103 |   1500 | 2015-10-30   |
|        305 |    102 |   1800 | 2015-09-15   |
|        306 |    102 |   1800 | 2015-09-30   |
+------------+--------+--------+--------------+
6 rows in set (0.00 sec)
```

MySQL Delete Statement

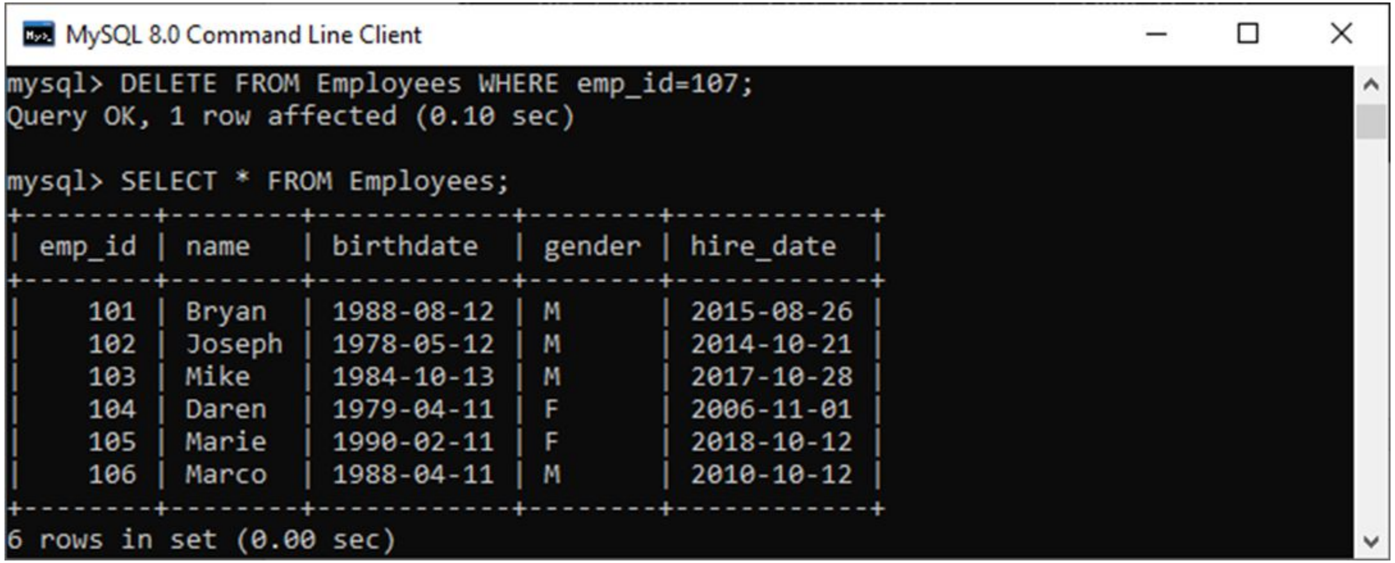
- If we want to delete an employee whose **emp_id is 107**, we should use the DELETE statement with the WHERE clause. See the below query:

```
mysql> DELETE FROM Employees WHERE emp_id=107;
```

TIA-Academy

MySQL Delete Statement

- After the execution of the query, it will return the output as below image. Once the record is deleted, verify the table using the SELECT statement:



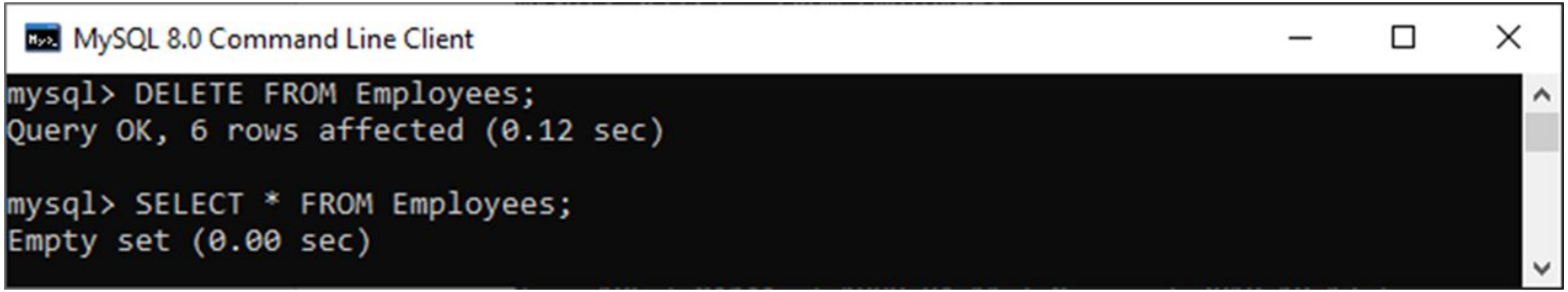
MySQL 8.0 Command Line Client

```
mysql> DELETE FROM Employees WHERE emp_id=107;
Query OK, 1 row affected (0.10 sec)

mysql> SELECT * FROM Employees;
+--------+--------+------------+--------+------------+
| emp_id | name   | birthdate  | gender | hire_date  |
+--------+--------+------------+--------+------------+
|    101 | Bryan  | 1988-08-12 | M      | 2015-08-26 |
|    102 | Joseph | 1978-05-12 | M      | 2014-10-21 |
|    103 | Mike   | 1984-10-13 | M      | 2017-10-28 |
|    104 | Daren  | 1979-04-11 | F      | 2006-11-01 |
|    105 | Marie  | 1990-02-11 | F      | 2018-10-12 |
|    106 | Marco  | 1988-04-11 | M      | 2010-10-12 |
+--------+--------+------------+--------+------------+
6 rows in set (0.00 sec)
```

TIA-Academy

MySQL Delete Statement Examples

- If we want to delete all records from the table, there is no need to use the WHERE clause with the DELETE statement. See the below code and output:



```
mysql> DELETE FROM Employees;
Query OK, 6 rows affected (0.12 sec)

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

- In the above output, we can see that after removing all rows, the Employees table will be empty. It means no records available in the selected table.

TIA-Academy

MySQL Select Statement

- The SELECT statement in MySQL is used to **fetch data from one or more tables**.
- We can retrieve records of all fields or specified fields that match specified criteria using this statement.
- It can also work with various scripting languages such as PHP, Ruby, and many more.

TIA-Academy

MySQL Select Statement

- It is the most commonly used SQL query. The general syntax of this statement to fetch data from tables are as follows:

```
SELECT field_name1, field_name 2,... field_nameN
FROM table_name1, table_name2...
[WHERE condition]
[GROUP BY field_name(s)]
[HAVING condition]
[ORDER BY field_name(s)]
[OFFSET M ][LIMIT N];
```

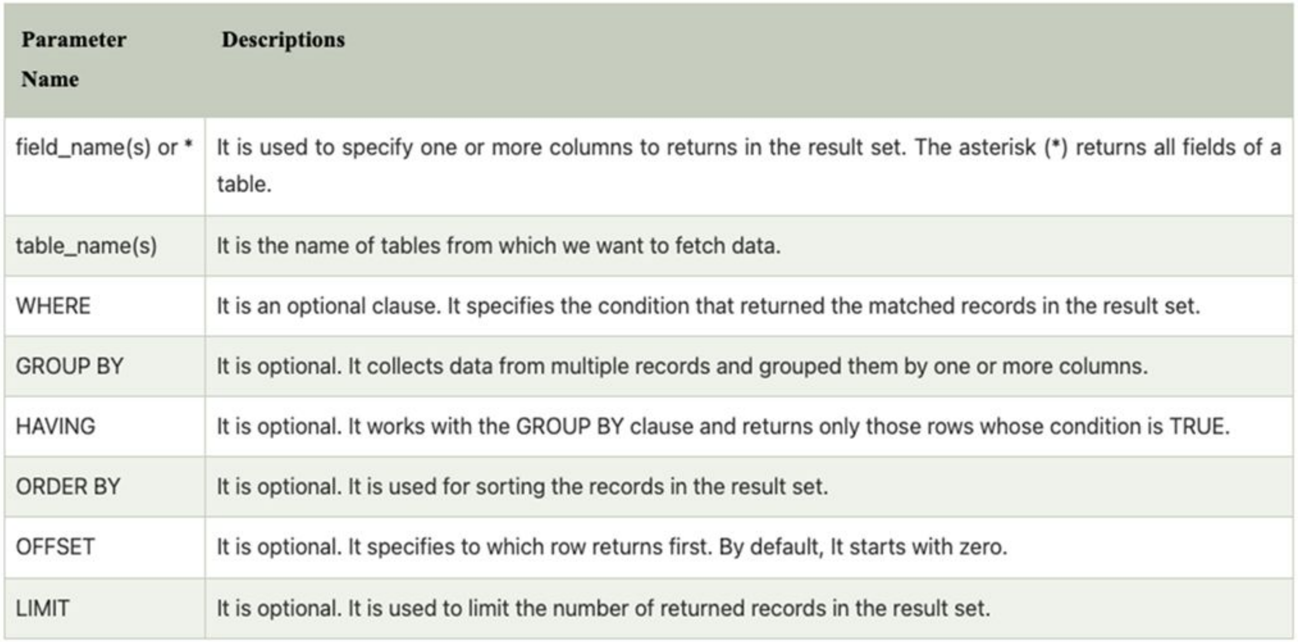
MySQL Select Statement

- Syntax for all fields:

```
SELECT * FROM tables [WHERE conditions]
[GROUP BY fieldName(s)]
[HAVING condition]
[ORDER BY fieldName(s)]
[OFFSET M ][LIMIT N];
```

MySQL Select Statement

- The SELECT statement uses the following parameters:

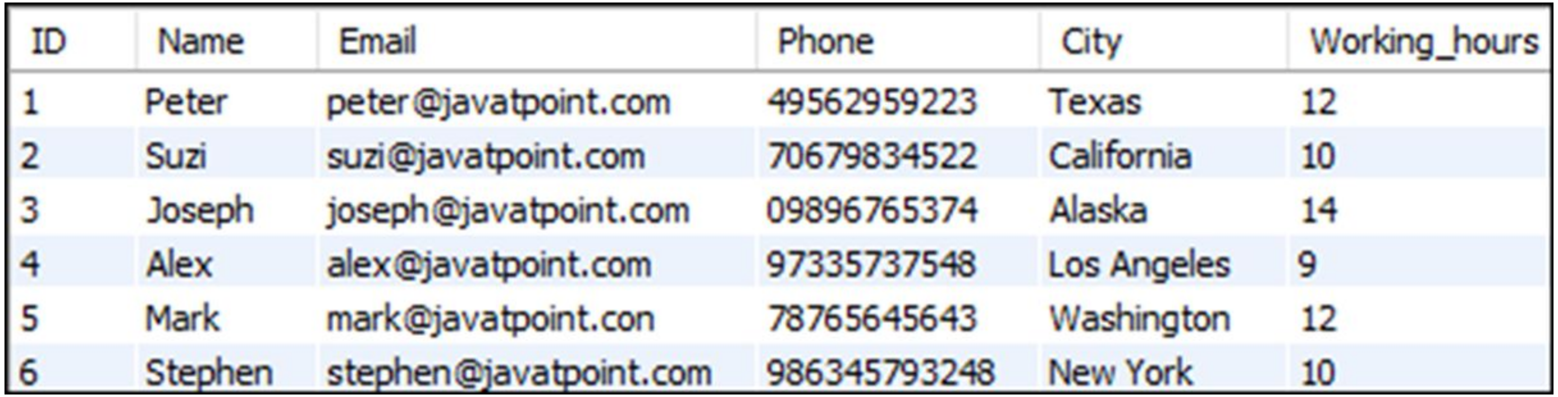


Parameter Name	Descriptions
field_name(s) or *	It is used to specify one or more columns to returns in the result set. The asterisk (*) returns all fields of a table.
table_name(s)	It is the name of tables from which we want to fetch data.
WHERE	It is an optional clause. It specifies the condition that returned the matched records in the result set.
GROUP BY	It is optional. It collects data from multiple records and grouped them by one or more columns.
HAVING	It is optional. It works with the GROUP BY clause and returns only those rows whose condition is TRUE.
ORDER BY	It is optional. It is used for sorting the records in the result set.
OFFSET	It is optional. It specifies to which row returns first. By default, It starts with zero.
LIMIT	It is optional. It is used to limit the number of returned records in the result set.

TIA-Academy

MySQL Select Statement Example

- Let us understand how SELECT command works in MySQL with the help of various examples.
- Suppose we have a table named **employee_detail** that contains the following data:

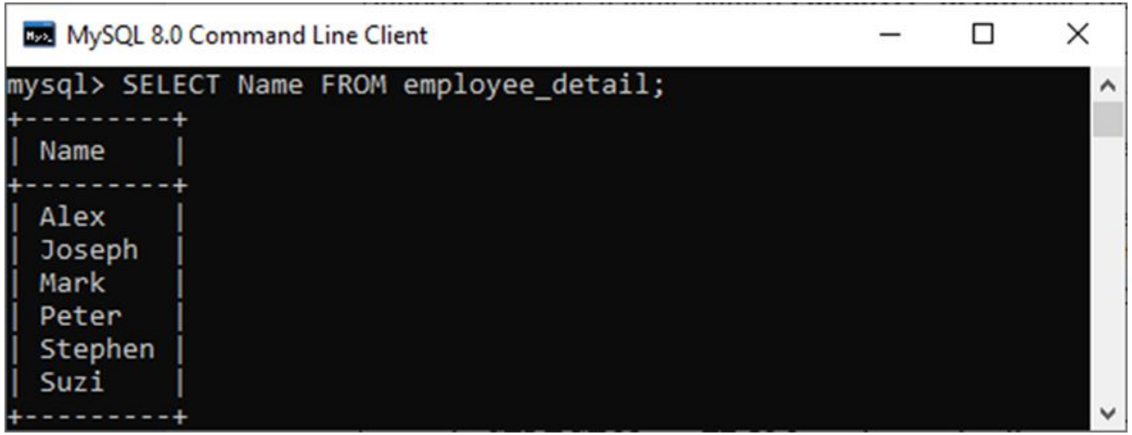


ID	Name	Email	Phone	City	Working_hours
1	Peter	peter@javatpoint.com	49562959223	Texas	12
2	Suzi	suzi@javatpoint.com	70679834522	California	10
3	Joseph	joseph@javatpoint.com	09896765374	Alaska	14
4	Alex	alex@javatpoint.com	97335737548	Los Angeles	9
5	Mark	mark@javatpoint.con	78765645643	Washington	12
6	Stephen	stephen@javatpoint.com	986345793248	New York	10

MySQL Select Statement Example

- If we want to retrieve a **single column from the table**, we need to execute the below query:

```
mysql> SELECT Name FROM employee_detail;
```


MySQL 8.0 Command Line Client

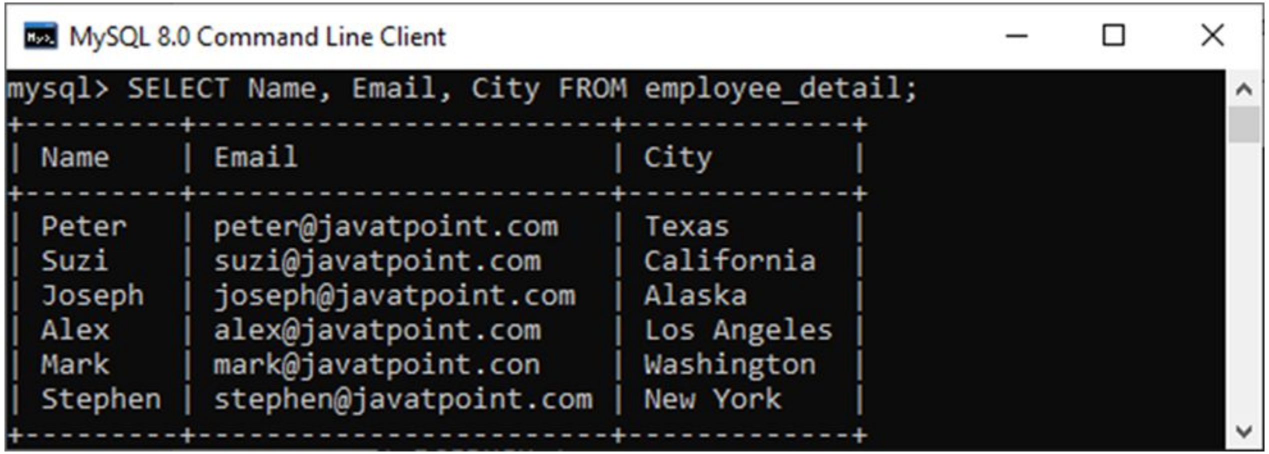
```
mysql> SELECT Name FROM employee_detail;
+---------+
| Name    |
+---------+
| Alex    |
| Joseph  |
| Mark    |
| Peter   |
| Stephen |
| Suzi    |
+---------+
```

TIA-Academy

MySQL Select Statement Example

- If we want to query **multiple columns from the table**, we need to execute the below query:

```
mysql> SELECT Name, Email, City FROM employee_detail;
```

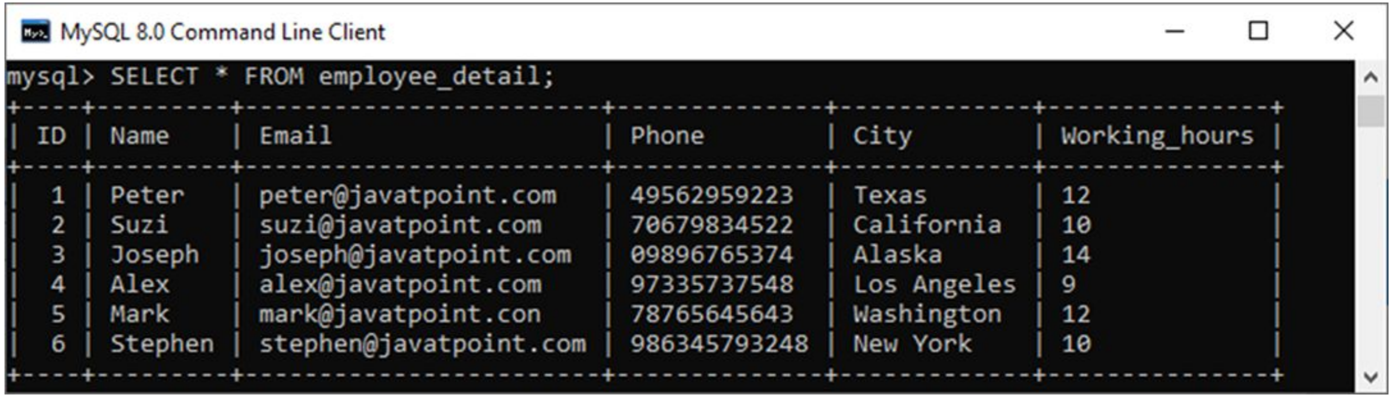


```
mysql> SELECT Name, Email, City FROM employee_detail;
+---------+-----------------------+-------------+
| Name    | Email                 | City        |
+---------+-----------------------+-------------+
| Peter   | peter@javatpoint.com  | Texas       |
| Suzi    | suzi@javatpoint.com   | California  |
| Joseph  | joseph@javatpoint.com | Alaska      |
| Alex    | alex@javatpoint.com   | Los Angeles |
| Mark    | mark@javatpoint.con   | Washington  |
| Stephen | stephen@javatpoint.com | New York   |
+---------+-----------------------+-------------+
```

MySQL Select Statement Example

- If we want to fetch data from **all columns of the table**, we need to use all column's names with the select statement. Specifying all column names is not convenient to the user, so MySQL uses an **asterisk** (*) to retrieve all column data as follows:

```
mysql> SELECT * FROM employee_detail;
```



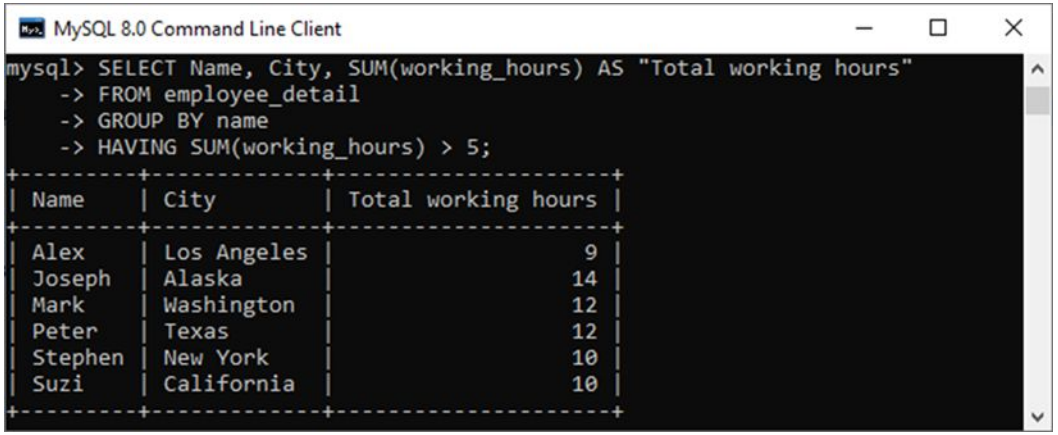
```
mysql> SELECT * FROM employee_detail;
+----+---------+------------------------+--------------+-------------+---------------+
| ID | Name    | Email                  | Phone        | City        | Working_hours |
+----+---------+------------------------+--------------+-------------+---------------+
|  1 | Peter   | peter@javatpoint.com   | 49562959223  | Texas       | 12            |
|  2 | Suzi    | suzi@javatpoint.com    | 70679834522  | California  | 10            |
|  3 | Joseph  | joseph@javatpoint.com  | 09896765374  | Alaska      | 14            |
|  4 | Alex    | alex@javatpoint.com    | 97335737548  | Los Angeles | 9             |
|  5 | Mark    | mark@javatpoint.con    | 78765645643  | Washington  | 12            |
|  6 | Stephen | stephen@javatpoint.com | 986345793248 | New York    | 10            |
+----+---------+------------------------+--------------+-------------+---------------+
```

TIA-Academy

MySQL Select Statement Example

- Here, we use the **SUM function** with the **HAVING** clause in the SELECT command to get the employee name, city, and total working hours.
- Also, it uses the **GROUP BY** clause to group them by the Name column:

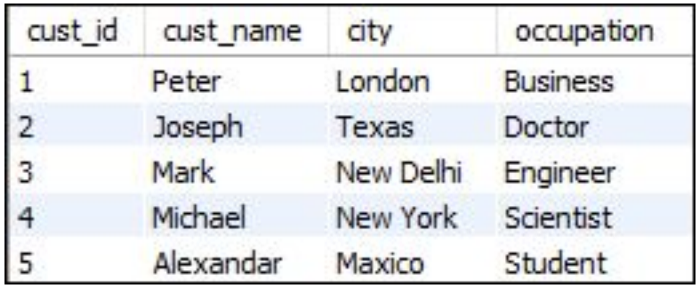
```
SELECT Name, City, SUM(working_hours) AS "Total working hours"
FROM employee_detail
GROUP BY Name
HAVING SUM(working_hours) > 5;
```


MySQL 8.0 Command Line Client

```
mysql> SELECT Name, City, SUM(working_hours) AS "Total working hours"
    -> FROM employee_detail
    -> GROUP BY name
    -> HAVING SUM(working_hours) > 5;
+---------+-------------+---------------------+
| Name    | City        | Total working hours |
+---------+-------------+---------------------+
| Alex    | Los Angeles |                   9 |
| Joseph  | Alaska      |                  14 |
| Mark    | Washington  |                  12 |
| Peter   | Texas       |                  12 |
| Stephen | New York    |                  10 |
| Suzi    | California  |                  10 |
+---------+-------------+---------------------+
```

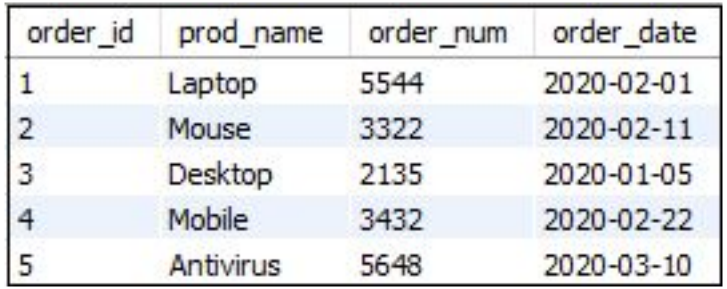
MySQL Select Statement Example

- MySQL SELECT statement can also be used to retrieve records from multiple tables by using a **JOIN statement**. Suppose we have a table named **"customer"** and **"orders"** that contains the following data:



cust_id	cust_name	city	occupation
1	Peter	London	Business
2	Joseph	Texas	Doctor
3	Mark	New Delhi	Engineer
4	Michael	New York	Scientist
5	Alexandar	Maxico	Student

Table: customer



order_id	prod_name	order_num	order_date
1	Laptop	5544	2020-02-01
2	Mouse	3322	2020-02-11
3	Desktop	2135	2020-01-05
4	Mobile	3432	2020-02-22
5	Antivirus	5648	2020-03-10

Table: orders

TIA-Academy

MySQL Select Statement Example

- Execute the following SQL statement that returns the matching records from both tables using the **INNER JOIN** query:

```
SELECT cust_name, city, order_num, order_date
FROM customer INNER JOIN orders
ON customer.cust_id = orders.order_id
WHERE order_date < '2020-04-30'
ORDER BY cust_name;
```

TIA-Academy

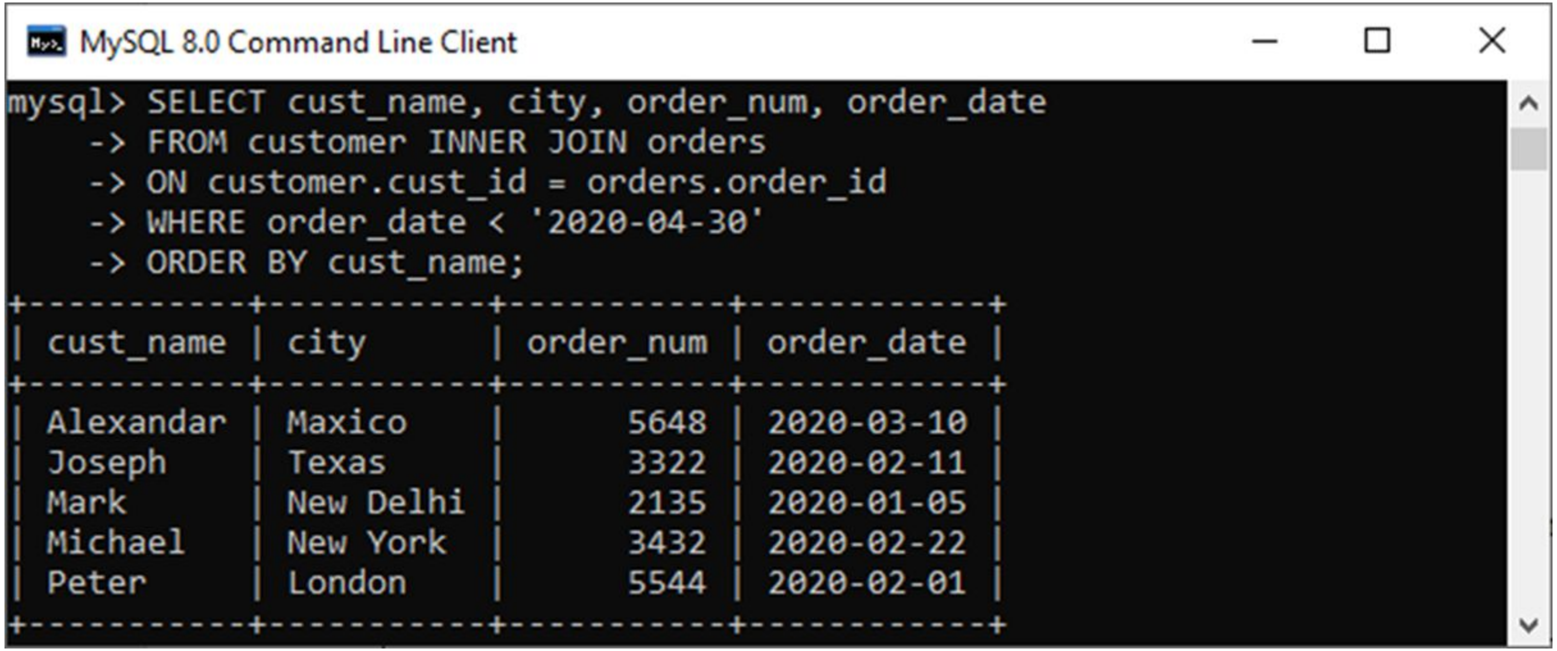
MySQL Select Statement Example

- Execute the following SQL statement that returns the matching records from both tables using the **INNER JOIN** query:

```
SELECT cust_name, city, order_num, order_date
FROM customer INNER JOIN orders
ON customer.cust_id = orders.order_id
WHERE order_date < '2020-04-30'
ORDER BY cust_name;
```

MySQL Select Statement Example

- After successful execution of the query, we will get the output as follows:



```
mysql> SELECT cust_name, city, order_num, order_date
    -> FROM customer INNER JOIN orders
    -> ON customer.cust_id = orders.order_id
    -> WHERE order_date < '2020-04-30'
    -> ORDER BY cust_name;
+-----------+-----------+-----------+------------+
| cust_name | city      | order_num | order_date |
+-----------+-----------+-----------+------------+
| Alexandar | Maxico    |      5648 | 2020-03-10 |
| Joseph    | Texas     |      3322 | 2020-02-11 |
| Mark      | New Delhi |      2135 | 2020-01-05 |
| Michael   | New York  |      3432 | 2020-02-22 |
| Peter     | London    |      5544 | 2020-02-01 |
+-----------+-----------+-----------+------------+
```

TIA-Academy

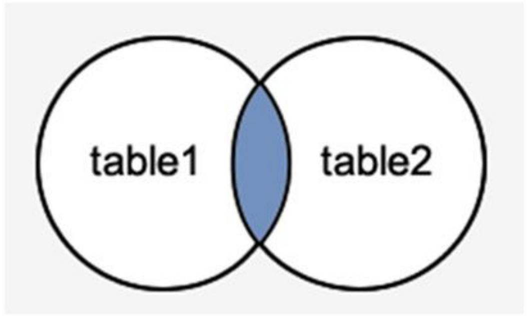
MySQL Joins

- MySQL JOINS are used with SELECT statement.
  - It is used to retrieve data from multiple tables.
  - It is performed whenever you need to fetch records from two or more tables.
- There are three types of MySQL joins:
  - MySQL INNER JOIN (or sometimes called simple join)
  - MySQL LEFT OUTER JOIN (or sometimes called LEFT JOIN)
  - MySQL RIGHT OUTER JOIN (or sometimes called RIGHT JOIN)

MySQL Inner JOIN (Simple Join)

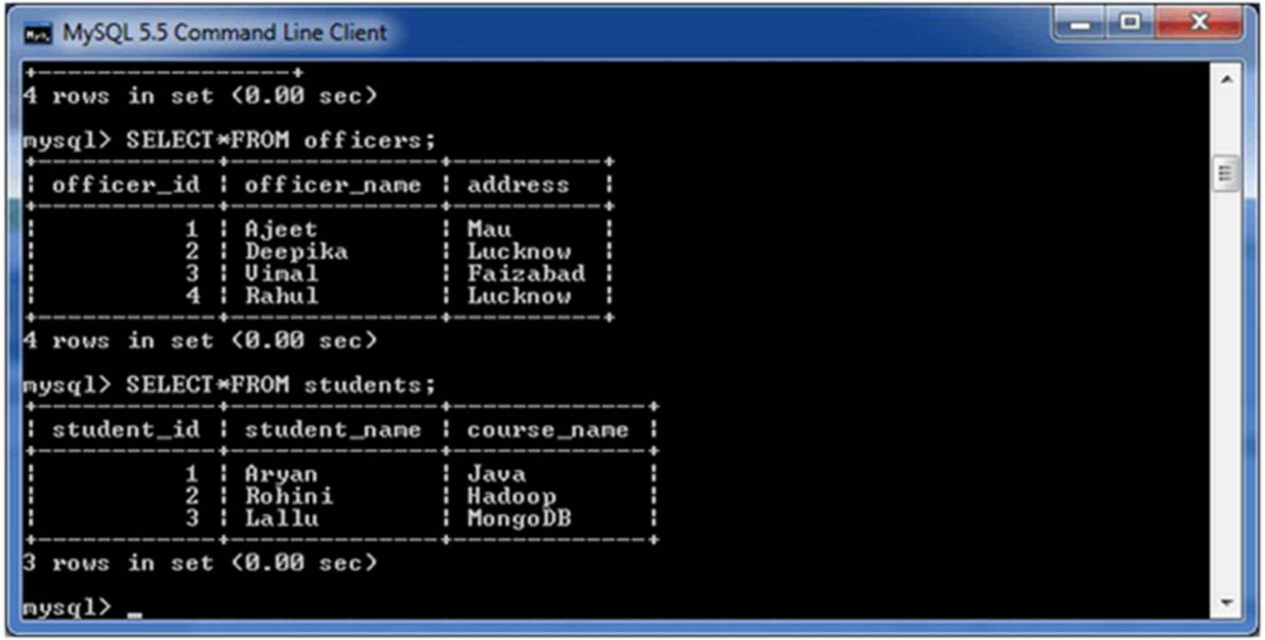
- The MySQL INNER JOIN is used to return all rows from multiple tables where the join condition is satisfied. It is the most common type of join.

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



MySQL Inner JOIN (Simple Join)

- Consider two tables "officers" and "students", having the following data.



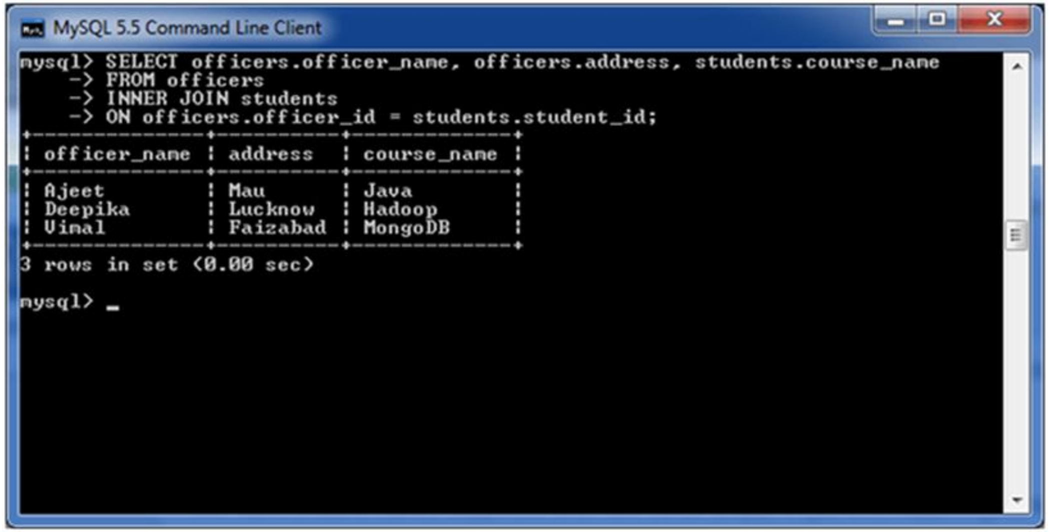
MySQL 5.5 Command Line Client

```
+------------------+
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> SELECT*FROM students;
+------------+--------------+-------------+
| student_id | student_name | course_name |
+------------+--------------+-------------+
|          1 | Aryan        | Java        |
|          2 | Rohini       | Hadoop      |
|          3 | Lallu        | MongoDB     |
+------------+--------------+-------------+
3 rows in set (0.00 sec)

mysql> _
```

TIA-Academy

MySQL Inner JOIN (Simple Join)

```
SELECT officers.officer_name, officers.address, students.course_name
FROM officers
INNER JOIN students
ON officers.officer_id = students.student_id;
```



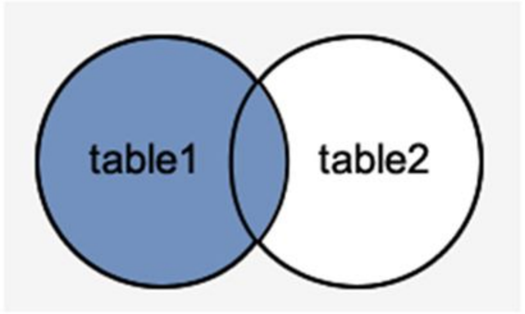
```
mysql> SELECT officers.officer_name, officers.address, students.course_name
    -> FROM officers
    -> INNER JOIN students
    -> ON officers.officer_id = students.student_id;
+--------------+----------+-------------+
| officer_name | address  | course_name |
+--------------+----------+-------------+
| Ajeet        | Mau      | Java        |
| Deepika      | Lucknow  | Hadoop      |
| Vimal        | Faizabad | MongoDB     |
+--------------+----------+-------------+
3 rows in set (0.00 sec)

mysql> _
```

TIA-Academy

MySQL Left Outer Join

- The LEFT OUTER JOIN returns all rows from the left hand table specified in the ON condition and only those rows from the other table where the join condition is fulfilled.

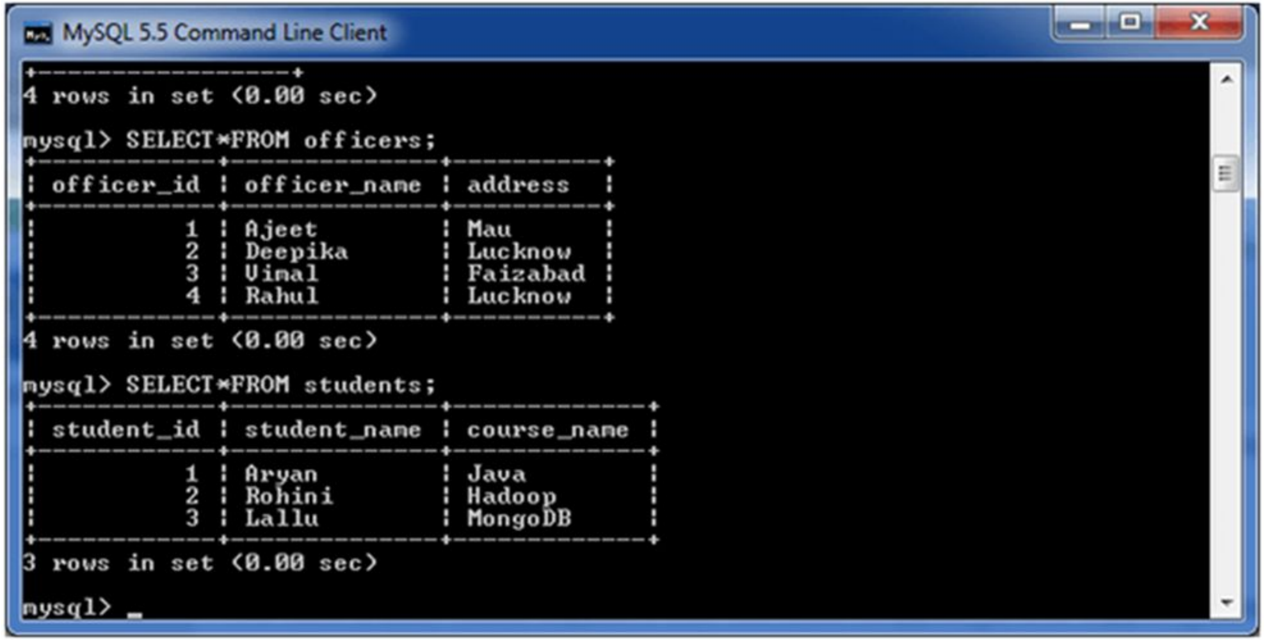
```
SELECT columns
FROM table1
LEFT [OUTER] JOIN table2
ON table1.column = table2.column;
```



TIA-Academy

MySQL Left Outer Join

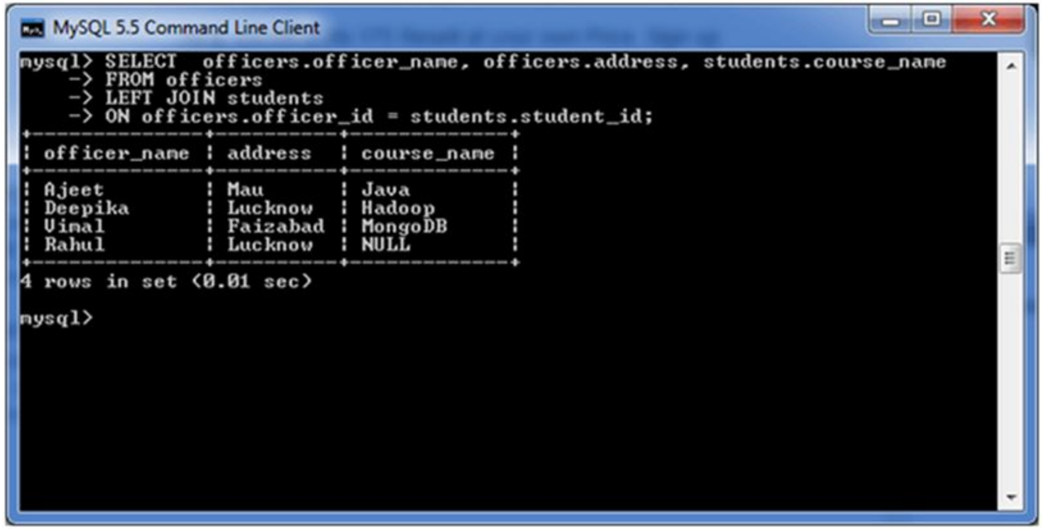
- Consider two tables "officers" and "students", having the following data.



```
+------------------+
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> SELECT*FROM students;
+------------+--------------+-------------+
| student_id | student_name | course_name |
+------------+--------------+-------------+
|          1 | Aryan        | Java        |
|          2 | Rohini       | Hadoop      |
|          3 | Lallu        | MongoDB     |
+------------+--------------+-------------+
3 rows in set (0.00 sec)

mysql> _
```

MySQL Left Outer Join

```
SELECT  officers.officer_name, officers.address, students.course_name
FROM officers
LEFT JOIN students
ON officers.officer_id = students.student_id;
```



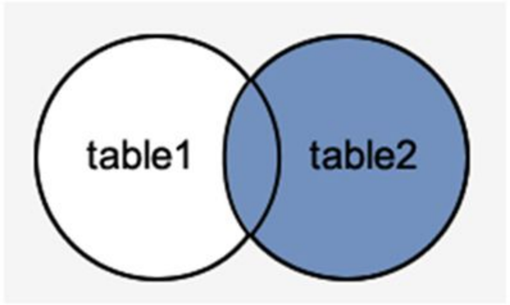
MySQL 5.5 Command Line Client

```
mysql> SELECT  officers.officer_name, officers.address, students.course_name
    -> FROM officers
    -> LEFT JOIN students
    -> ON officers.officer_id = students.student_id;
+--------------+----------+-------------+
| officer_name | address  | course_name |
+--------------+----------+-------------+
| Ajeet        | Mau      | Java        |
| Deepika      | Lucknow  | Hadoop      |
| Vimal        | Faizabad | MongoDB     |
| Rahul        | Lucknow  | NULL        |
+--------------+----------+-------------+
4 rows in set (0.01 sec)

mysql>
```

MySQL Right Outer Join

- The MySQL Right Outer Join returns all rows from the RIGHT-hand table specified in the ON condition and only those rows from the other table where he join condition is fulfilled.

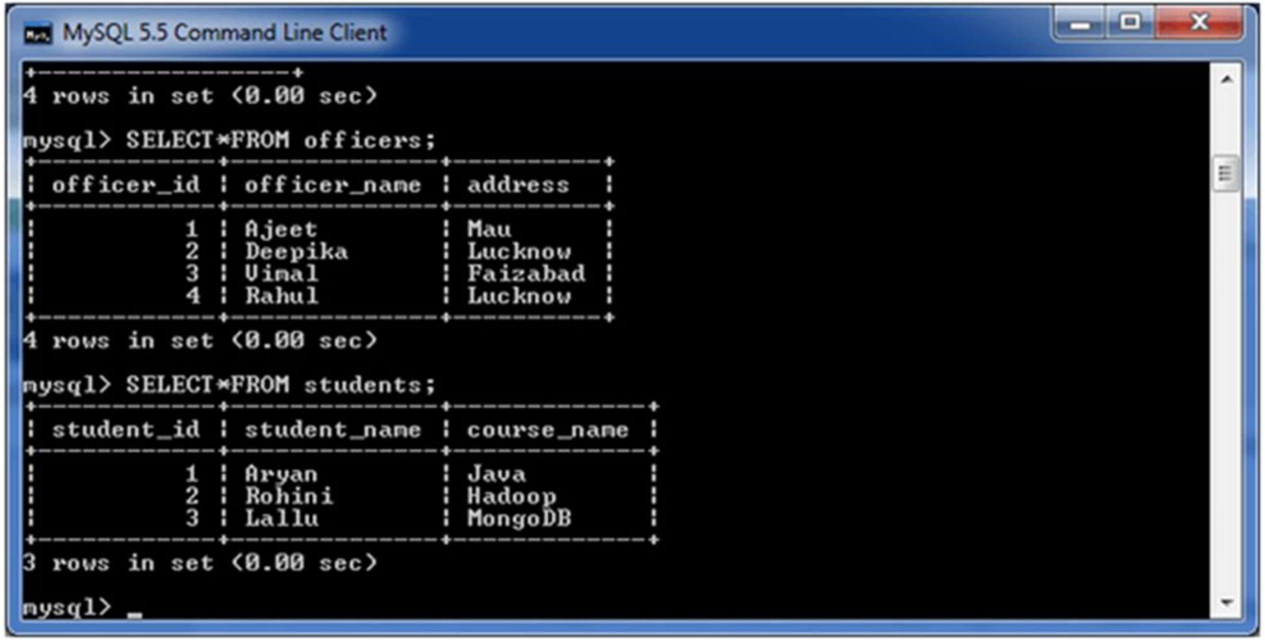
```
SELECT columns
FROM table1
RIGHT [OUTER] JOIN table2
ON table1.column = table2.column;
```



TIA-Academy

MySQL Right Outer Join

- Consider two tables "officers" and "students", having the following data.



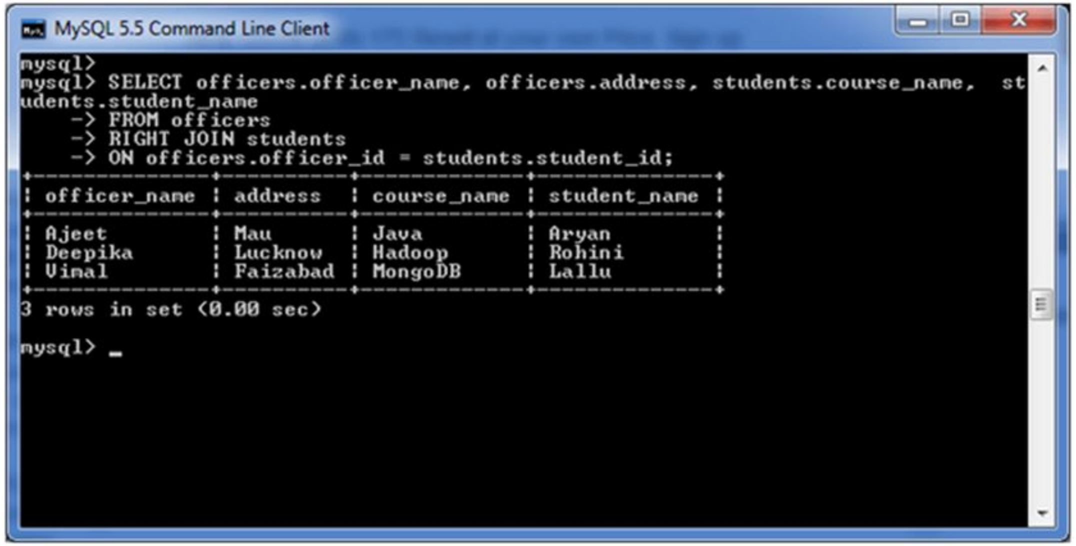
MySQL 5.5 Command Line Client

```
+-------------------+
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> SELECT*FROM students;
+------------+--------------+-------------+
| student_id | student_name | course_name |
+------------+--------------+-------------+
|          1 | Aryan        | Java        |
|          2 | Rohini       | Hadoop      |
|          3 | Lallu        | MongoDB     |
+------------+--------------+-------------+
3 rows in set (0.00 sec)

mysql> _
```

TIA-Academy

MySQL Right Outer Join

```
SELECT officers.officer_name, officers.address, students.course_name, students.student_name
FROM officers
RIGHT JOIN students
ON officers.officer_id = students.student_id;
```



MySQL 5.5 Command Line Client

```
mysql>
mysql> SELECT officers.officer_name, officers.address, students.course_name,  st
udents.student_name
    -> FROM officers
    -> RIGHT JOIN students
    -> ON officers.officer_id = students.student_id;
+--------------+----------+-------------+--------------+
| officer_name | address  | course_name | student_name |
+--------------+----------+-------------+--------------+
| Ajeet        | Mau      | Java        | Aryan        |
| Deepika      | Lucknow  | Hadoop      | Rohini       |
| Vimal        | Faizabad | MongoDB     | Lallu        |
+--------------+----------+-------------+--------------+
3 rows in set (0.00 sec)

mysql> _
```

TIA-Academy

ASSIGNMENT 05 (HOME ASSIGNMENT)

TIA-Academy

Thank You