جامعة القاهرة الجديدة التكنولوجية
NEW CAIRO TECHNOLOGICAL UNIVERSITY
كلية تكنولوجيا الصناعة والطاقة

# PEARSON BTEC International Standards Verifier

## ICT Program

Dr. Amany AbdElSamea Saeed

Lec. 2 MySQL-Basics

February 2024

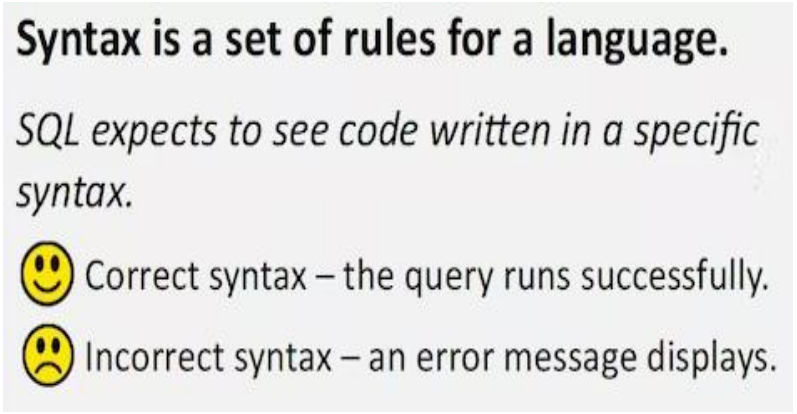
# Outline



- MySQL Syntax
- Connecting to and Disconnecting from the Server
- MYSQL Basic Queries
- Creating and Using a Database
- Creating and Using table

# MySQL Syntax

- SQL is followed by unique set of rules and guidelines called Syntax.
- All the SQL statements start with any of the keywords like SELECT, INSERT, UPDATE, DELETE, ALTER, DROP, CREATE, USE, SHOW and all the statements end with a semicolon(;).
- Important point to be noted is that SQL is not case sensitive, which means SELECT and select have same meaning in SQL statements.
- The great thing about everything you do in MySQL is that the "code" is very easy for humans to read, as opposed to harder programming languages like C or C++.



# MYSQL Connection

# MySQL Connection

- A user can connect with the database server, whether on the same machine or remote locations.

- MySQL Connection Types:
  - Command-line client
  - MySQL Workbench
  - PHP Script

# Connection Using command-line client

- To connect to the server, you usually need to provide a MySQL user name and password when you invoke mysql .
- If the server runs on a machine other than the one where you log in, you must also specify a host name.

```
$> mysql -h host -u user -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 25338 to server version: 8.0.32-standard
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>
```

- If you are logging in on the same machine that MySQL is running on simply use the following:

```
$> mysql -u user -p
```

- After you have connected successfully, you can disconnect any time by typing QUIT (or \q) at the mysql> prompt, You can also terminate the session by issuing an EXIT statement or (under Unix) by typing Ctrl-D.

```
mysql> QUIT
Bye
```

# Set Password to MySQL Root

- we can also set the initial password using the following command

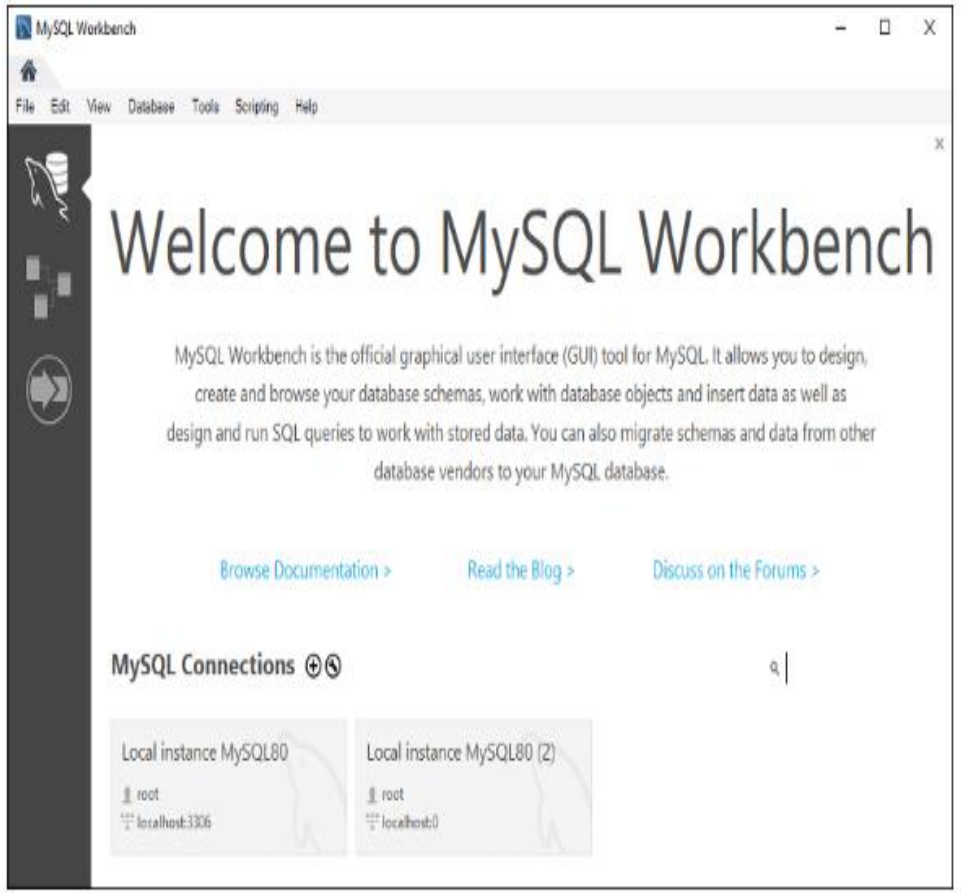
```
mysql> mysql -u root password "new_password";
```

- Reset Password
  - We can also change the existing password using the SET PASSWORD statement. However, we can only do so after logging in to the user account using the existing password.

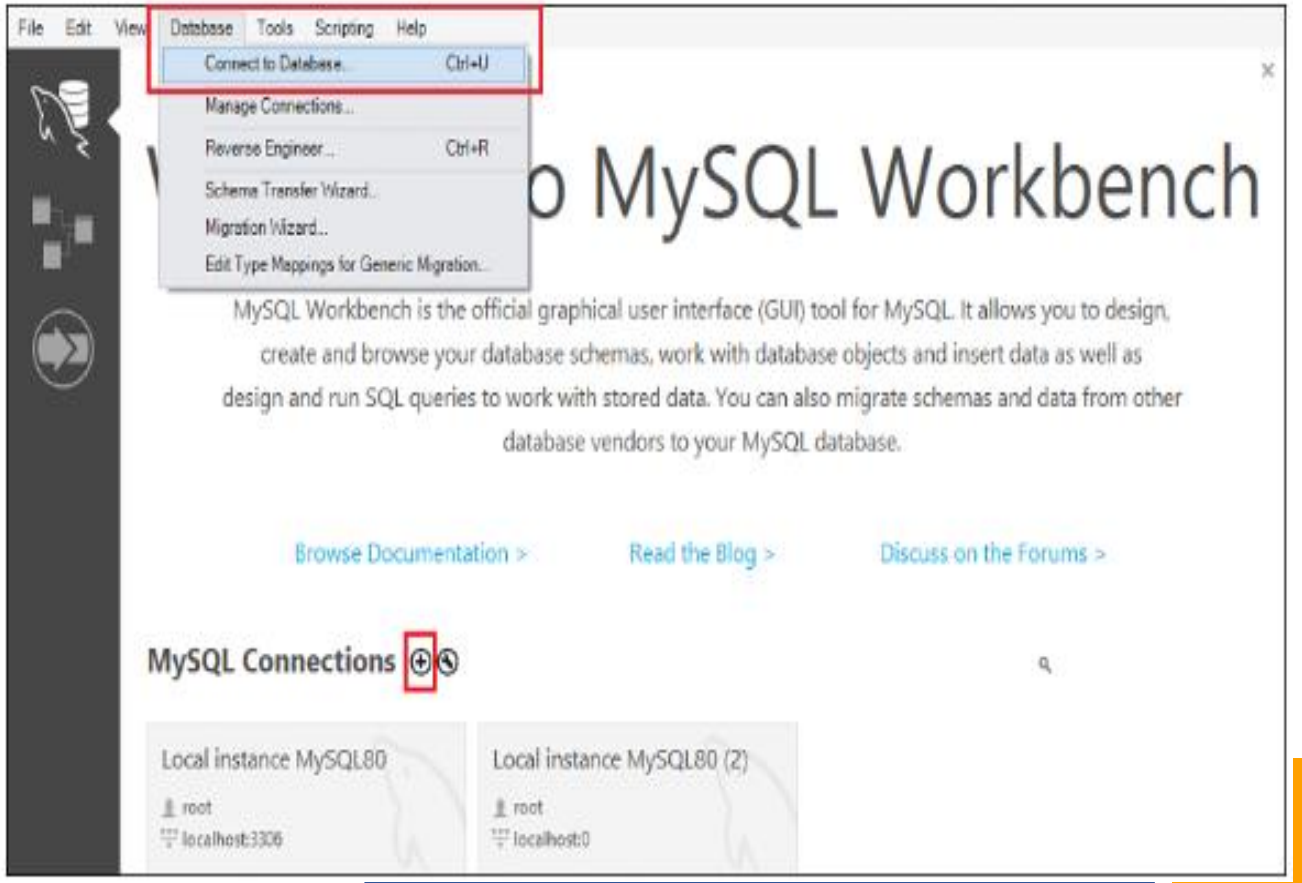
```
mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('password_name');
FLUSH PRIVILEGES;
```

# Connection Using MYSQL Workbench

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

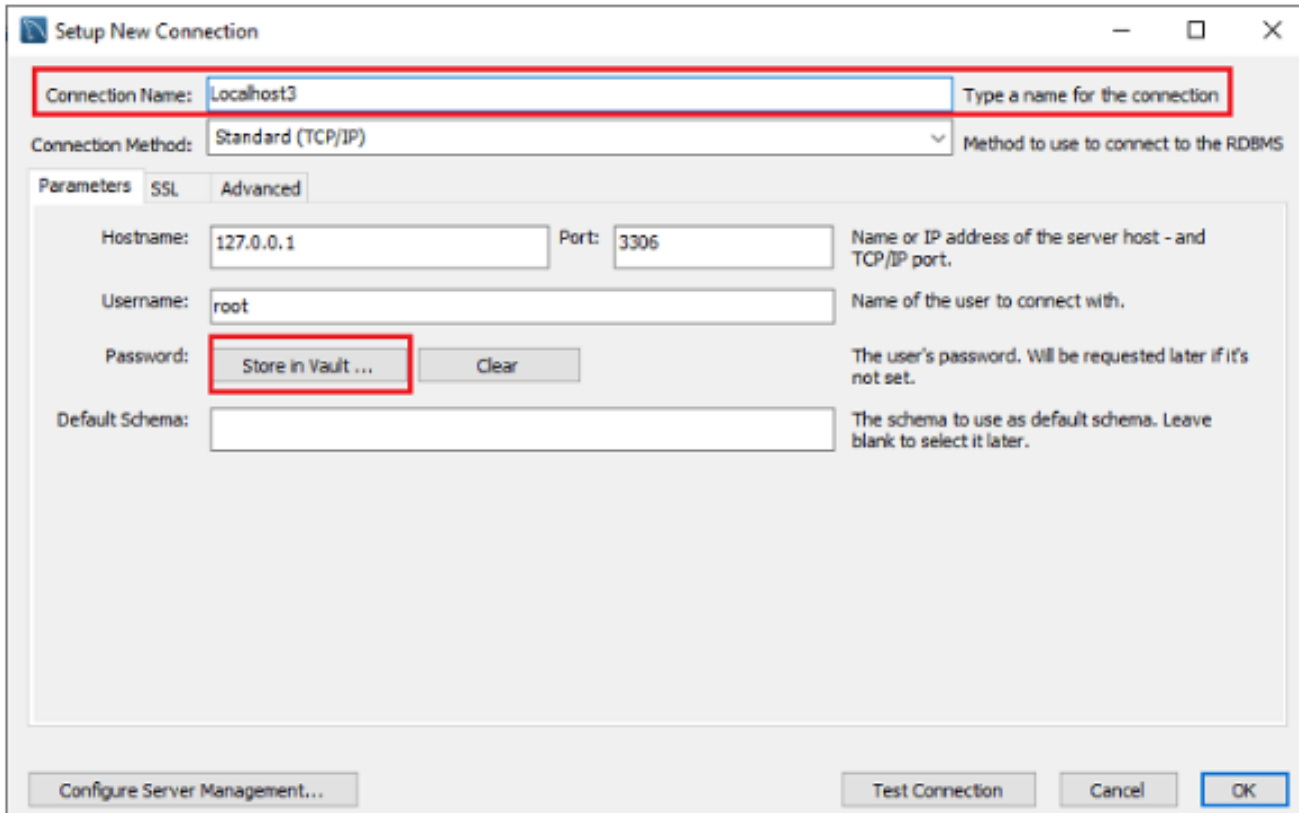



- **Step 2**: Navigate to the menu bar, click on the **'Database'** and choose **Connect to Database** option or press the **CTRL+U** command. We can also connect with the database server by just clicking the **plus (+) button** located next to the MySQL Connection

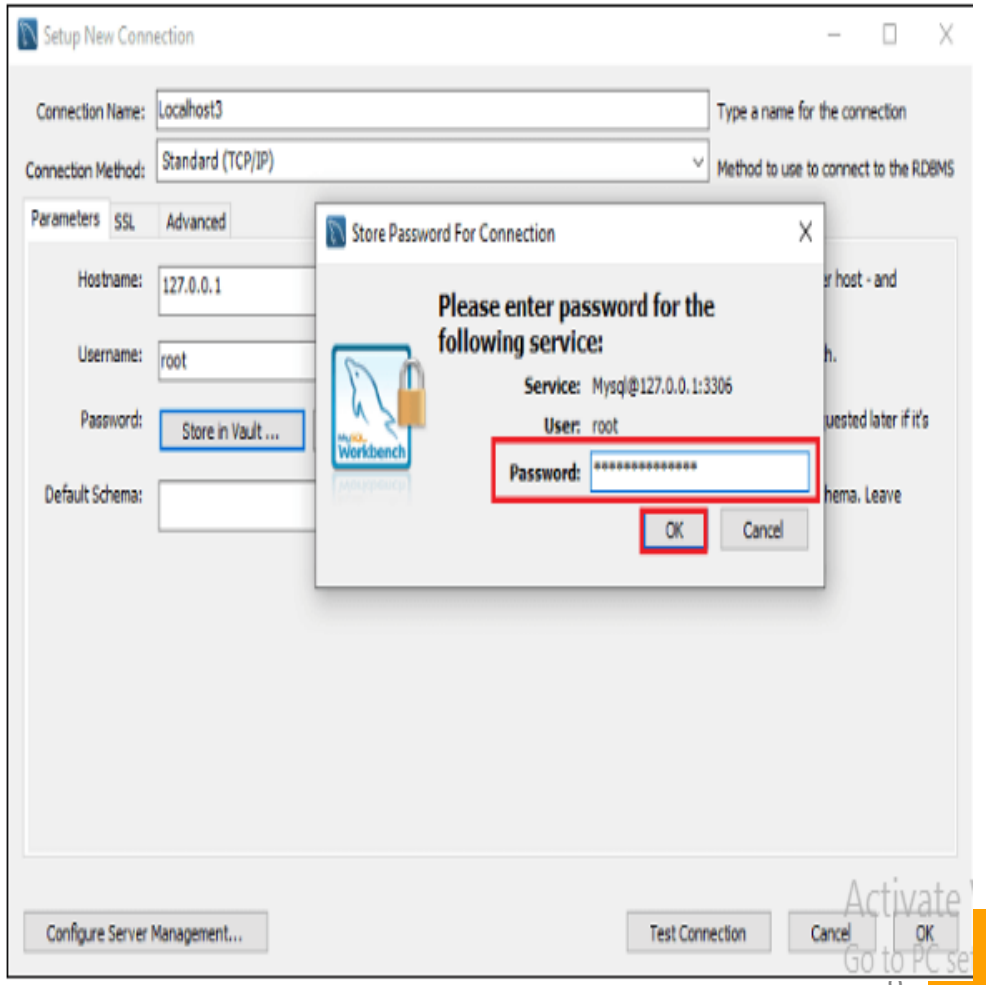


# Connection Using MYSQL Workbench

- **Step3**: 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.

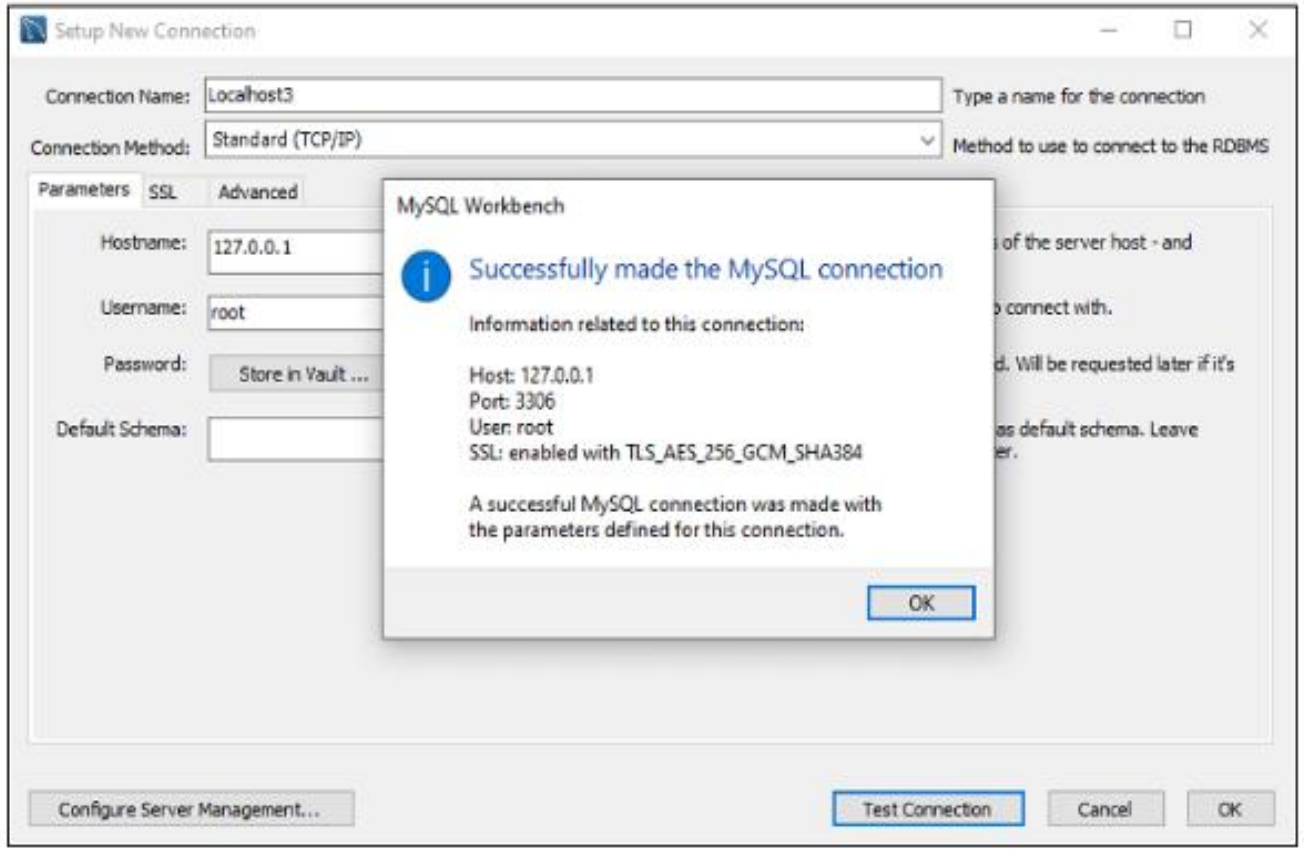



- **Step 4**: We will get a new window to write the password and click the **OK** button.

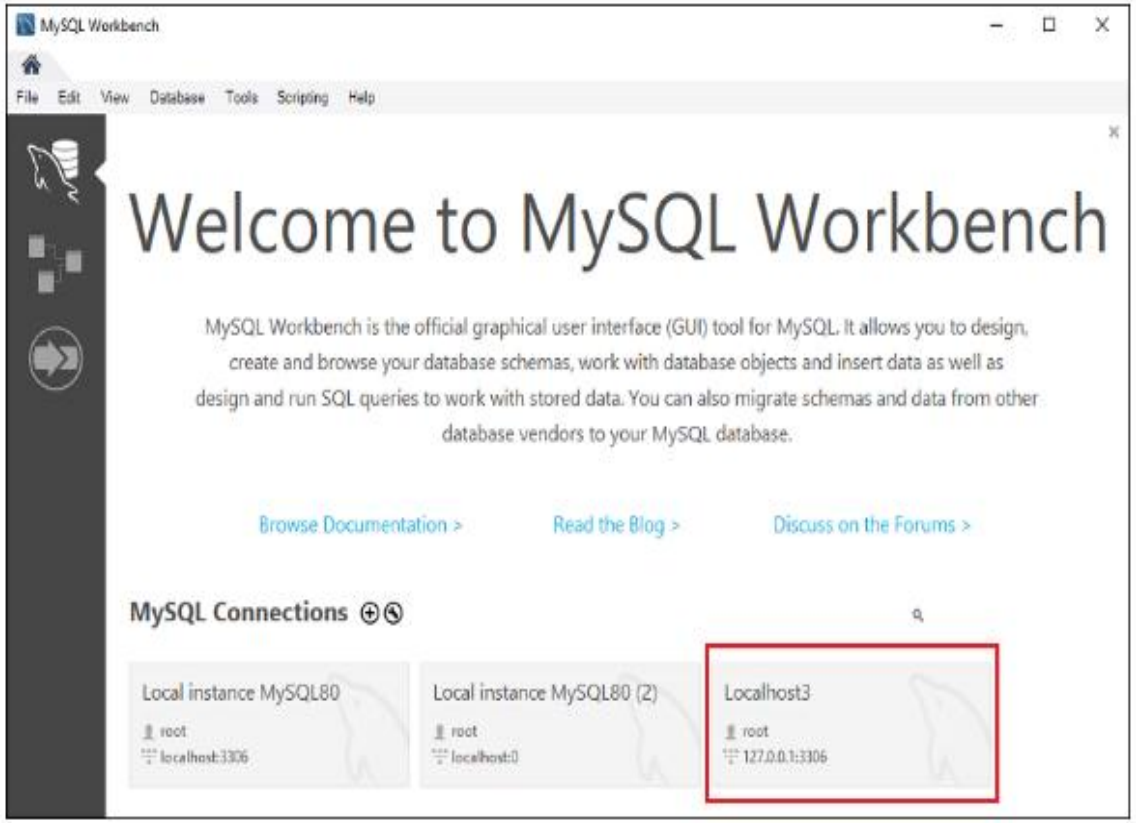


# Connection Using MYSQL Workbench

- **Step 5**: 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.



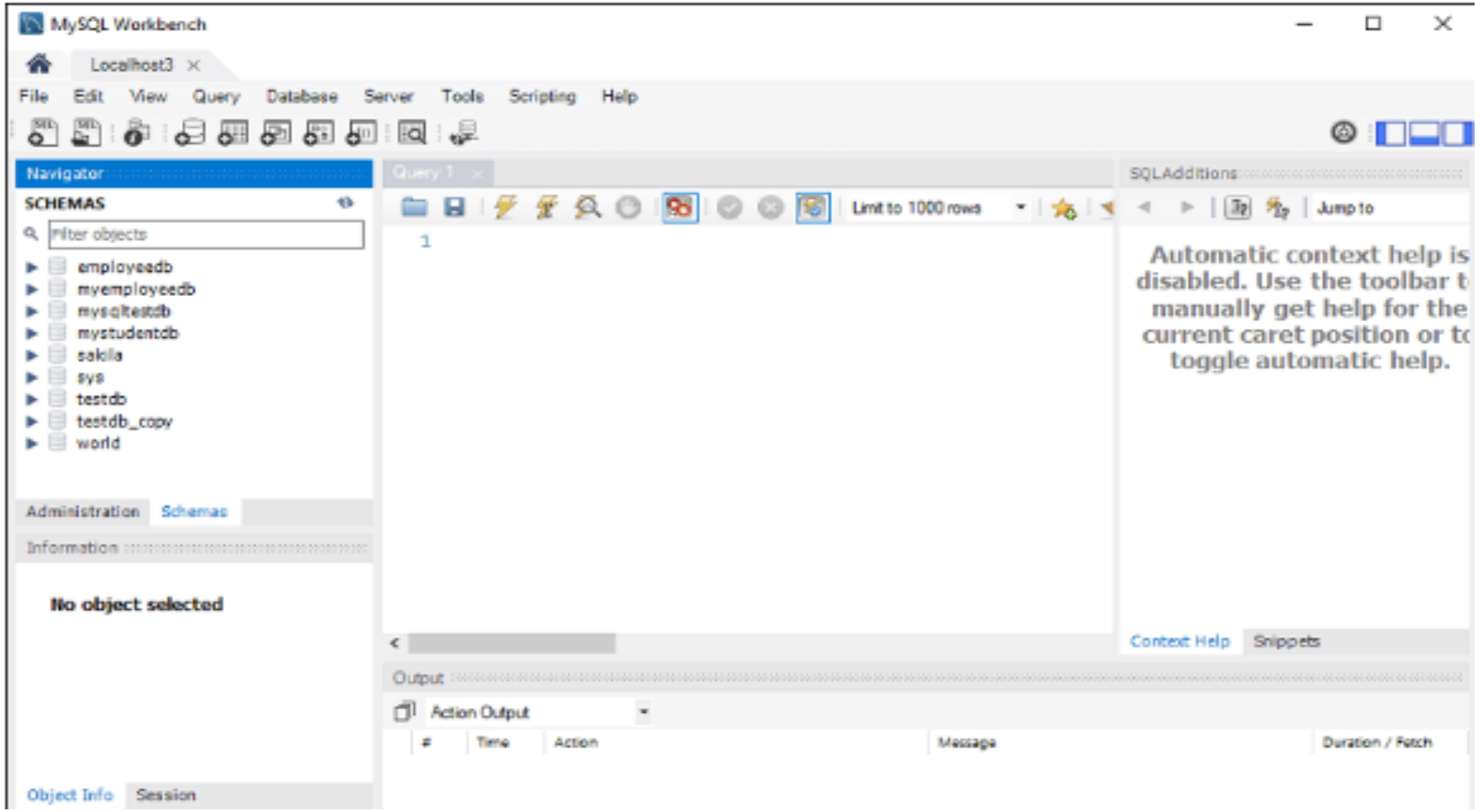

- **Step 6**: Again, click on the **OK** button for saving connection setup. After finishing all the setup, we can see this connection under **MySQL Connections**



# Connection Using MYSQL Workbench



- **Step 7**: Now, we can click this newly created connection that displays the current schemas and a pane for entering queries:



# Connection Using PHP Script

- The simplest way to connect with the MySQL database server using the PHP script is to use the **mysql_connect() function**. This function needs **five parameters** and returns the MySQL link identifier when the connection becomes successful. If the connection is failed, it returns **FALSE**.
- **Syntax:**

```
connection mysql_connect(server, user, passwordd, new_link, client_flag);
```

- If we want **to disconnect from the MySQL database server**, we can use another PHP function named **mysql_close().**

```
bool mysql_close ( resource $link_identifier );
```

```
$sql = "RENAME TABLE old_table_name TO new_table_name";
$mysqli->query($sql);
```

# MYSQL Database

# Create Database

- Syntax:

```
Mysql> CREATE DATABASE  mydatabase;
```

- *You will get an error if you run the CREATE DATABASE statement without specifying IF NOT EXISTS and the database already exists. So it's better to use the IF NOT EXISTS clause to prevent errors.*

```
Mysql> CREATE DATABASE IF NOT EXISTS mydatabase;
```

# Show and Use Command

- Use the SHOW statement to find out what databases currently exist on the server:

```
mysql> SHOW DATABASES;
+-----------+
| Database  |
+-----------+
| mysql     |
| test      |
| tmp       |
+-----------+
```

- If the test database exists, try to access it:

```
mysql> USE test
Database changed
```

- USE, like QUIT, does not require a semicolon. (You can terminate such statements with a semicolon if you like; it does no harm.)

# Delete Database

- Now, suppose after you create your database, you realize that you have typed the name wrongly. There is no easy way to rename a database in MySQL. What you can do is create a new database and delete the old database.

```
mysql> DROP DATABASE [IF EXISTS] name_of_database;
```

# MySQL - Database Export

- The **mysqldump** command-line tool is used in MySQL to create backups of databases. It can be used to back up an entire database, specific tables, or even specific rows based of a table.
- Following is the syntax of mysqldump command to export a database :

```
$ mysqldump -u username -p database_name > output_file.sql
```

- Exporting only Specific Tables in Database

```
$ mysqldump -u username -p database_name table1 table2 ... > output_file.sql
```

- Exporting all Databases in a Host

```
$ mysqldump -u root -p --all-databases > database_dump.sql
```

# MySQL - Database Import

- We can import the backup data into an MySQL database using the mysql command-line tool. It takes the username, database name, and the backup file with the data.
- Syntax

```
$ mysql -u username -p new_database_name < dumpfile_path
```

- **username**: This is the MySQL username to use when connecting to the MySQL server.
- **new_database_name**: The name of the database where you want to import the data.
- **dumpfile_path**: It is the path of the backup file. The data will be imported from this file.
- **<**: This symbol **imports** the data from the file named output_file_path.

# MYSQL Tables

# CREATE TABLE Statement

- The CREATE TABLE statement is used to create tables in MYSQL database.
- Syntax

```
CREATE TABLE [IF NOT EXISTS] table_name(
column1 datatype,
column2 datatype,
column3 datatype,
.....
columnN datatype, );
```

```
mysql> CREATE TABLE pet (
    -> name VARCHAR(20),
    -> owner VARCHAR(20),
    -> species VARCHAR(20),
    -> sex CHAR(1),
    -> birth DATE, death DATE);
Query OK, 0 rows affected (0.04 sec)
```

# Showing Tables

- To verify that the table has been created:
- Syntax

```
SHOW TABLES;
```

- SHOW TABLES with FULL modifier
- In MySQL, we use the optional **FULL** modifier along with the SHOW TABLES command to display a second output column that contains additional information about the tables present in a database, such as their types: **BASE TABLE** for a table, **VIEW** for a view, or **SYSTEM VIEW** for an INFORMATION_SCHEMA table.

```
SHOW FULL TABLES;
```

- SHOW TABLES in different Database

```
SHOW TABLES IN testdb2;
```

- We can also perform the above operation using the SHOW TABLES command with **FROM** clause.

```
SHOW TABLES FROM testdb2;
```

# Describe Tables

- Describing a MySQL table refers to retrieving its definition or structure. When we describe a table, it basically includes the fields present, their datatypes, and if any constraints defined on them.

- We can get the information about the table structure using the following SQL statements −

  - DESCRIBE Statement
  - DESC Statement
  - SHOW COLUMNS Statement
  - EXPLAIN Statement

# DESCRIBE Statement

- Syntax:

```
mysql> DESCRIBE table_name [col_name | wild];
```

- To view a table structure, use the DESCRIBE command:

```
mysql> DESCRIBE pet;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(20) | YES  |     | NULL    |       |
| owner   | varchar(20) | YES  |     | NULL    |       |
| species | varchar(20) | YES  |     | NULL    |       |
| sex     | char(1)     | YES  |     | NULL    |       |
| birth   | date        | YES  |     | NULL    |       |
| death   | date        | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
```

- Describing a specific column

```
mysql> DESCRIBE pet name;
```

# DESC Statement

- We can also retrieve the table information using the MySQL DESC statement instead of DESCRIBE. They both give the same results, so DESC is just a shortcut for DESCRIBE statement.

- Syntax:

```
mysql> DESC table_name [col_name | wild];
```

- Describing a specific column

```
mysql> DESC CUSTOMERS NAME;
```

# SHOW COLUMNS Statement

- The MySQL SHOW COLUMNS Statement is used to display the information of all the columns present in a table.
- Syntax:

```
mysql> SHOW COLUMNS FROM table_name;
```

- Example

```
mysql> SHOW COLUMNS FROM CUSTOMERS;
```

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| ID | int | NO | PRI | NULL | auto_increment |
| NAME | varchar(20) | NO | | NULL | |
| AGE | int | NO | | NULL | |
| ADDRESS | char(25) | YES | | NULL | |
| SALARY | decimal(18,2) | YES | | NULL | |

# EXPLAIN Statement

- The MySQL EXPLAIN Statement is a synonym of DESCRIBE Statement which retrieves the information of a table's structure such as column names, column data types, and constraints (if any).
- Syntax:

```
mysql> EXPLAIN table_name;
```

- Example

```
mysql> EXPLAIN CUSTOMERS;
```

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| ID | int | NO | PRI | NULL | auto_increment |
| NAME | varchar(20) | NO | | NULL | |
| AGE | int | NO | | NULL | |
| ADDRESS | char(25) | YES | | NULL | |
| SALARY | decimal(18,2) | YES | | NULL | |

# Rename Tables

- The MySQL RENAME TABLE statement is used to rename an existing table in a database with another name.
- Syntax

```
mysql> RENAME TABLE table_name TO new_name;
```

- Renaming Multiple Tables
- Syntax

```
Mysql>RENAME TABLE old_table1 TO new_table1, old_table2 TO new_table2,
old_table3 TO new_table3;
```

```
MysqlRENAME TABLE Cust1 TO Buyer1, Cust2 TO Buyer2, Cust3 TO Buyer3;
```

# Truncate Table

- The MySQL **TRUNCATE TABLE** statement is used to delete only the data of an existing table, but not the table.

- *You can delete a table using the DROP TABLE command, but be careful because it completely erases both data and the table's structure from the database.*

- Syntax

```
mysql> TRUNCATE TABLE table_name;
```

# Temporary Tables

- The **Temporary Tables** are the tables that are created in a database to store data temporarily. These tables will be automatically deleted once the current client session is terminated or ends.
- Temporary tables were introduced in MySQL version 3.23 or later.
- Creating Temporary Tables in MySQL  Dropping Temporary Tables in MySQL

```
mysql> CREATE TEMPORARY TABLE CUSTOMERS(
ID INT NOT NULL,
NAME VARCHAR (20) NOT NULL,
AGE INT NOT NULL,
ADDRESS CHAR (25) ,
SALARY DECIMAL (18, 2),
PRIMARY KEY (ID)
);
```

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

# Deleting a Table

- To delete an entire table,
  - use the DROP TABLE command:

```
Mysql> DROP Table pet;
Query OK, 0 rows affected (0.02 sec)
```

# ALTER Command

- The MySQL **ALTER** command is used to modify the structure of an existing table. It allows you to make various changes, such as adding, deleting, or modify columns within the table.
- Additionally, the ALTER command is also used to add and drop different constraints associated with an existing table.
- Syntax

```
mysql> ALTER TABLE table_name [alter_option ...];
```

- Dropping a Column
  - To drop a column in an existing table, we use the ALTER TABLE command with **DROP** clause.

```
mysql> ALTER TABLE CUSTOMERS DROP ID;
```

Note: A DROP clause will not work if the column is the only one left in the table.

# ALTER Command cont.,

- Adding a Column
  - To add a new column into an existing table, we use **ADD** keyword with the ALTER TABLE command.
  - Example

  In the following query, we are adding a column named ID into an existing table CUSTOMERS.

```
mysql> ALTER TABLE CUSTOMERS ADD ID INT;
```

- Repositioning a Column
  - If we want a column to be placed at a specific position within the table, we can use **FIRST** to make it the first column or **AFTER col_name** to indicate that the new column should be positioned after the **col_name**.

```
mysql> ALTER TABLE CUSTOMERS ADD ID INT FIRST;
```

```
mysql> ALTER TABLE CUSTOMERS ADD ID INT AFTER NAME;
```

# ALTER Command cont.,

- Altering a Column Definition or a Name
  - In MySQL, to change a column's definition, we use **MODIFY** or **CHANGE** clause in conjunction with the ALTER command

```
mysql> ALTER TABLE CUSTOMERS MODIFY NAME INT;
```

```
mysql> ALTER TABLE CUSTOMERS MODIFY ID VARCHAR(20);
```

- Altering a Column's Default Value
  - In MySQL, we can change a default value for any column by using the **DEFAULT** constraint with ALTER command.

```
mysql> ALTER TABLE CUSTOMERS ALTER NAME SET DEFAULT 1000;
```

  - We can remove the default constraint from any column by using **DROP** clause along with the ALTER command.

```
mysql> ALTER TABLE CUSTOMERS ALTER NAME DROP DEFAULT;
```

# ALTER Command cont.,

- Altering (Renaming) a Table
  - To rename a table, use the **RENAME** option of the **ALTER TABLE** statement.
  - The following query renames the table named CUSTOMERS to BUYERS.

```
mysql> ALTER TABLE CUSTOMERS RENAME TO BUYERS;
```



# Thank you