MySQL Workbench

- MySQL Workbench is a powerful visual tool for database design, development, and administration
- It provides an intuitive interface for database architects, developers, and DBAs to interact with MySQL databases
- With Workbench, you can create databases, tables, rows, and perform various operations such as altering and deleting tables and rows

1. Initial Setup Steps

- **Download and Install MySQL Workbench**
  - Download MySQL Workbench from the official MySQL website
  - Follow the installation instructions for your operating system (Windows, macOS, Linux)
- **Connecting to a MySQL Server**
  - **Example Scenario:** Let's say you have MySQL installed locally and want to connect to it
  - **Open MySQL Workbench :** Launch MySQL Workbench from your applications
  - **Connect to Database**
    - **Hostname:** `localhost`
    - **Port:** `3306`
    - **Username:** `root` (or another username you have set up)
    - **Password:** Enter your MySQL password
    - Click "OK" to connect to the database server

2. Creating a Database

Example Scenario: You want to create a database named `school_db`

1. **Create a New Schema**

   - **Right-click on "Schemas"** in the Navigator panel and select **"Create Schema"**
   - **Schema Name:** Enter `school_db`
   - **Click Apply**

   You'll see an SQL script like this:

```
CREATE SCHEMA `school_db` ;
```

- **Click Apply** again, and then **Finish**

3. Creating Tables

Example Scenario: You want to create a table named `students` in the `school_db` database with columns for `student_id`, `name`, `age`, and `class`

1. **Select the Database**
   - Expand the `school_db` schema
2. **Create a New Table**
   - **Right-click on "Tables"** and select **"Create Table."**.
   - **Table Name:** Enter `students`
3. **Define Columns**
   - **Column 1**
     - **Name:** `student_id`
     - **Data Type:** `INT`
     - **Constraints:** Check **PK** (Primary Key) and **NN** (Not Null), and enable **AI** (Auto Increment)
   - **Column 2**
     - **Name:** `name`
     - **Data Type:** `VARCHAR(100)`
     - **Constraints:** Check **NN**
   - **Column 3**
     - **Name:** `age`
     - **Data Type:** `INT`
   - **Column 4**
     - **Name:** `class`
     - **Data Type:** `VARCHAR(10)`
4. **Apply Changes**
   - Click "Apply"
   - The SQL script will look like this:

```
CREATE TABLE `school_db`.`students` (
  `student_id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(100) NOT NULL,
  `age` INT NULL,
  `class` VARCHAR(10) NULL,
  PRIMARY KEY (`student_id`));
```

   - Click "Apply" and then "Finish"

4. Inserting Rows

Example Scenario: Insert a few records into the `students` table

1. **Open the Table**
   - Right-click on the `students` table and select **"Select Rows - Limit 1000"**
2. **Insert Data**
   - In the result grid, add the following data
     - **Row 1:**
       - `name`: `John Doe`
       - `age`: `15`
       - `class`: `10`
     - **Row 2:**
       - `name`: `Jane Smith`
       - `age`: `14`
       - `class`: `9`
3. **Apply Changes**
   - Click the **"Apply"** button
   - The SQL script will look like this:

```
INSERT INTO `school_db`.`students` (`name`, `age`, `class`) VALUES ('John
Doe', 15, '10');
INSERT INTO `school_db`.`students` (`name`, `age`, `class`) VALUES ('Jane
Smith', 14, '9');
```

   - Click "Apply" and then "Finish"

5. Altering Tables

Example Scenario: You want to add a `gender` column to the `students` table

1. **Select the Table to Alter**
   - Right-click on the `students` table and select **"Alter Table"**
2. **Modify Table Structure**
   - Add a new column
     - **Column Name:** `gender`
     - **Data Type:** `VARCHAR(10)`
   - Click "Apply"

   The SQL script will look like this:

```
ALTER TABLE `school_db`.`students`
ADD COLUMN `gender` VARCHAR(10) NULL AFTER `class`;
```

- Click "Apply" and then "Finish"

6. Deleting Rows

Example Scenario: You want to delete the record of `John Doe` from the `students` table

1. **Open the Table**
   - Right-click on the `students` table and select **"Select Rows - Limit 1000"**
2. **Delete Data**
   - In the result grid, find the row with `John`.
   - Right-click on the left-most part of the row and select **"Delete Row(s)"**
   - Click "Apply"

The SQL script will look like this:

```
DELETE FROM `school_db`.`students` WHERE `student_id` = 1;
```

- Click "Apply" and then "Finish"

7. Deleting Tables

Example Scenario: You want to delete the `students` table from the `school_db` database

1. **Select the Table**
   - Right-click on the `students` table
2. **Delete the Table**
   - Select **"Drop Table"** from the context menu
   - Click "Apply"

The SQL script will look like this:

```
DROP TABLE `school_db`.`students`;
```

- Click "Apply" and then "Finish"

Database Initialization Using Bulk Import in MySQL Workbench

- **Bulk Import** in MySQL Workbench allows you to quickly load large amounts of data into a database table from a file, such as a CSV or SQL dump file
- This is particularly useful for initializing a database with pre-existing data or migrating data from another system

Steps to Perform Bulk Import in MySQL Workbench

Let's assume you have a CSV file named `students.csv` with the following content:

```
student_id,name,age,class,gender
1,John Doe,15,10,Male
2,Jane Smith,14,9,Female
3,Tom Brown,16,11,Male
4,Emily White,13,8,Female
```

1. Prepare the Table for Import

- Before importing data, you need a table in your MySQL database that matches the structure of your CSV file

Example
Create the `students` table in the `school_db` database with the appropriate columns

1. **Create the Table (if not already created)**

```
CREATE TABLE `school_db`.`students` (
    `student_id` INT NOT NULL,
    `name` VARCHAR(100) NOT NULL,
    `age` INT NULL,
    `class` VARCHAR(10) NULL,
    `gender` VARCHAR(10) NULL,
    PRIMARY KEY (`student_id`)
);
```

   - You can execute this SQL command directly in MySQL Workbench by selecting your database (`school_db`), opening a new SQL tab, pasting the above script, and running it

2. Initiate the Bulk Import Process

1. **Open MySQL Workbench** and connect to your MySQL server
2. **Select the Database**
   - In the **Navigator** panel, select the `school_db` database by expanding the **Schemas** section and clicking on `school_db`
3. **Right-Click on the Table**
   - Right-click on the `students` table and select **"Table Data Import Wizard"**
4. **Choose the File to Import**
   - In the **"File to Import"** dialog, browse to the location of your CSV file (`students.csv`), select it, and click **"Next"**
5. **Configure Import Settings**:
   - **Data Source:** Ensure the **Input File Type** is set to `CSV`
   - **Field Separator:** Typically, this is a comma `,` for CSV files
   - **Line Separator:** Defaults to `\n` (new line)
   - **Enclosed by:** Set this to a quotation mark `"` if your data fields are enclosed in quotes
   - Click **"Next"**
6. **Map the Columns**
   - In the **"Column Mappings"** step, make sure each column in the CSV file is mapped correctly to the corresponding column in the `students` table. MySQL Workbench usually does this automatically, but you should verify it
   - If the first row in your CSV contains headers (as in this example), ensure the **"First Row Contains Column Names"** option is checked
   - Click **"Next"**
7. **Import Data**:
   - Click **"Next"** to start the import process. MySQL Workbench will import the data from your CSV file into the `students` table
   - A progress bar will show the status of the import
   - Once the import is complete, click **"Finish"**
8. **Verify the Import**:
   - Right-click on the `students` table and select **"Select Rows - Limit 1000"** to view the imported data and ensure everything has been loaded correctly