MySQL Setup

01 Install MySQL Server

```
sudo apt update
```

```
sudo apt install mysql-server
```

02 Start the MySQL Service

```
sudo service mysql start
```

- To enable MySQL to start on boot

```
sudo systemctl enable mysql
```

03 Secure MySQL Installation

- MySQL provides a security script to improve the security of your MySQL installation

```
sudo mysql_secure_installation
```

- You'll be prompted to set up a root password, remove anonymous users, disallow root login remotely, remove test databases, and reload privilege tables. It's recommended to answer 'Yes' to all prompts

04 Access MySQL from the Terminal

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

- `-u root` specifies the username (root)
- `-p` prompts you to enter the root password you set during installation

Access MySQL as the System Root User

- You need to log in to MySQL as the system's root user without requiring a password

```
sudo mysql
```

- This command should give you direct access to the MySQL shell

Change the Authentication Method for MySQL Root User

- Once inside the MySQL shell, change the root user's authentication method to the traditional password-based method

```
USE mysql;
```

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY
'your_password';
```

```
FLUSH PRIVILEGES;
```

```
exit;
```

- Replace `'your_password'` with a secure password of your choice

Reattempt Login with Password

- Now, try logging in again with the new password you set

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

- This process changes the root user's authentication method from `auth_socket` to `mysql_native_password`, allowing you to log in with a password
- After this, you should be able to access MySQL as the root user using the password you set

Optional: Revert Back to auth_socket

- If you want to revert to the `auth_socket` method

```
sudo mysql
```

```
USE mysql;
```

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH auth_socket;
```

```
FLUSH PRIVILEGES;
```

```
exit;
```

- This will revert the authentication method to the default `auth_socket`

05 Create a User and Grant Privileges (Optional)

- You can create a new MySQL user and grant privileges

```
CREATE USER 'newuser'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON *.* TO 'newuser'@'localhost' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

- Replace `'newuser'` with the desired username
- Replace `'password'` with the desired password

06 Access MySQL with the New User (Optional)

- You can now access MySQL using the new user account

```
mysql -u newuser -p
```

- Enter the password you set for the new user

07 Exit MySQL

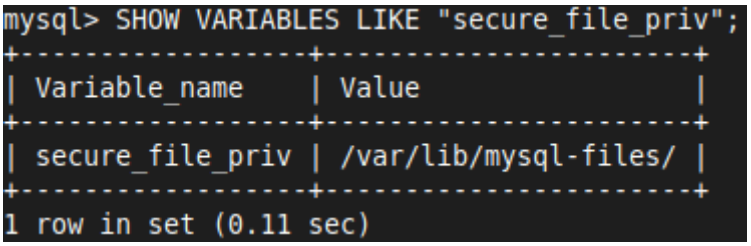
To exit the MySQL shell, type:

```
exit;
```

Database Initialization Using Bulk Import

- Using MySQL command-line
- **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

```
SHOW VARIABLES LIKE "secure_file_priv"
```



- Move the CSV file to `/var/lib.mysql-files/`

```
sudo cp table_name.csv /var/lib.mysql-files/
```

- Load data into table (table_name)

```
LOAD DATA INFILE '/var/lib.mysql-files/table_name.csv'
INTO TABLE table_name
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY "\n"
IGNORE 1 ROWS;
```

MySQL Backup

1. Using `mysqldump`

- The `mysqldump` utility is the most commonly used method to back up MySQL databases
- It creates a text file with SQL statements to recreate the database schema and insert the data

Basic Syntax:

```
mysqldump -u [username] -p [database_name] > [backup_file.sql]
```

Example:

```
mysqldump -u root -p DEMO > DEMO.sql
```

- The `backup_file.sql` (in this case, `DEMO.sql`) will be saved in the current working directory from which you run the `mysqldump` command
- If you want to specify a different directory, you can provide a full path to the file:

```
mysqldump -u root -p demo > /path/to/your/backup/DEMO.sql
```

2. Backing Up Multiple Databases

- To back up multiple databases, use the `--databases` option

```
mysqldump -u root -p --databases db1 db2 > multiple_databases_backup.sql
```

3. Backing Up All Databases

- To back up all databases on the MySQL server, use the `--all-databases` option

```
mysqldump -u root -p --all-databases > all_databases_backup.sql
```

4. Options for `mysqldump`

- `--single-transaction`: Ensures a consistent backup by using a single transaction
- `--routines`: Includes stored routines and functions in the backup
- `--triggers`: Includes triggers in the backup

MySQL Restore

1. Using `mysql`

- To restore a database from a backup file created by `mysqldump`, use the `mysql` command.

Basic Syntax:

```
mysql -u [username] -p [database_name] < [backup_file.sql]
```

Example:

```
mysql -u root -p DEMO < DEMO.sql
```

2. Restoring to a New Database:

- First create the database:

```
CREATE DATABASE new_database;
```

- Then restore it:

```
mysql -u root -p new_database < DEMO.sql
```

Tips

- **Compression:** For large databases, consider compressing the backup file to save disk space. For example, use gzip:

  ```
  mysqldump -u root -p demo | gzip > demo_backup.sql.gz
  ```

- To restore:

  ```
  gunzip < demo_backup.sql.gz | mysql -u root -p demo
  ```