Sri Lanka Institute of Information Technology

Assignment

Implementing Security features in OS & DB

IE3062 - Data and Operating Systems Security

Individual Assignment

Submitted by:

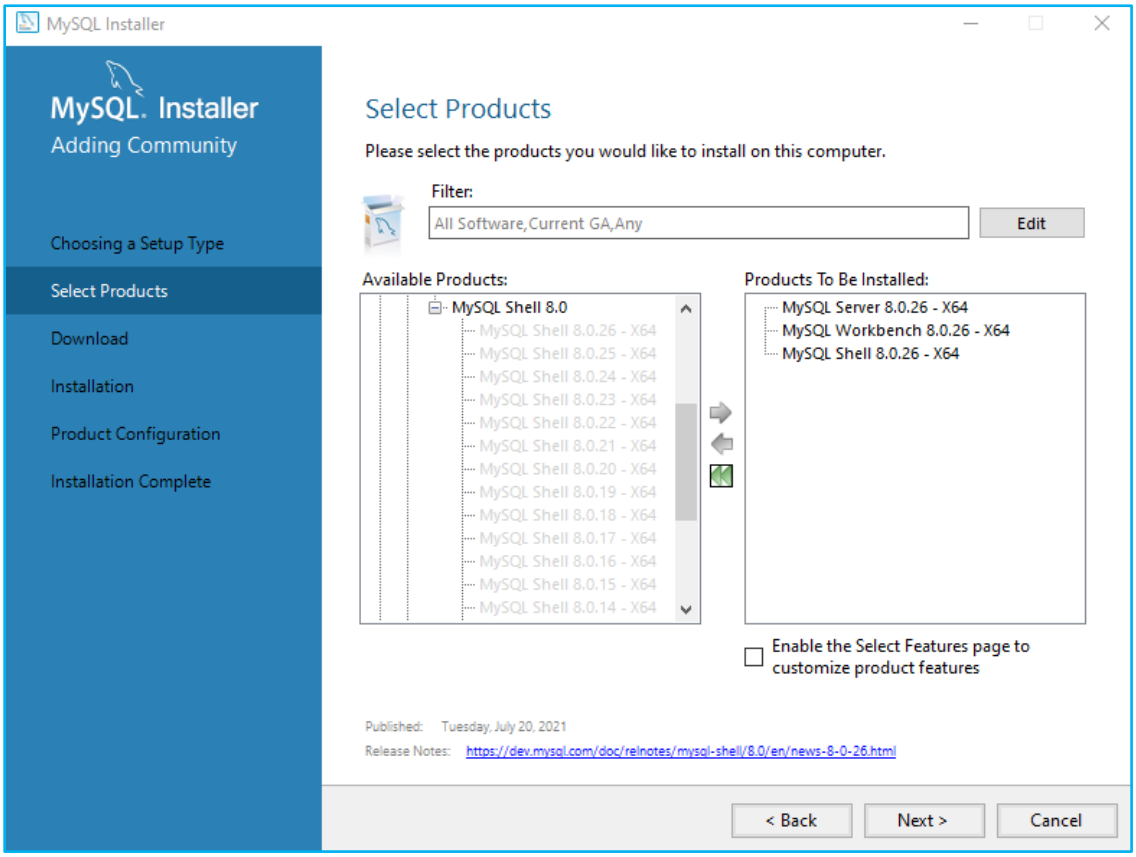
Student Registration Number	Student Name
IT19029146	Eranda H.P.D

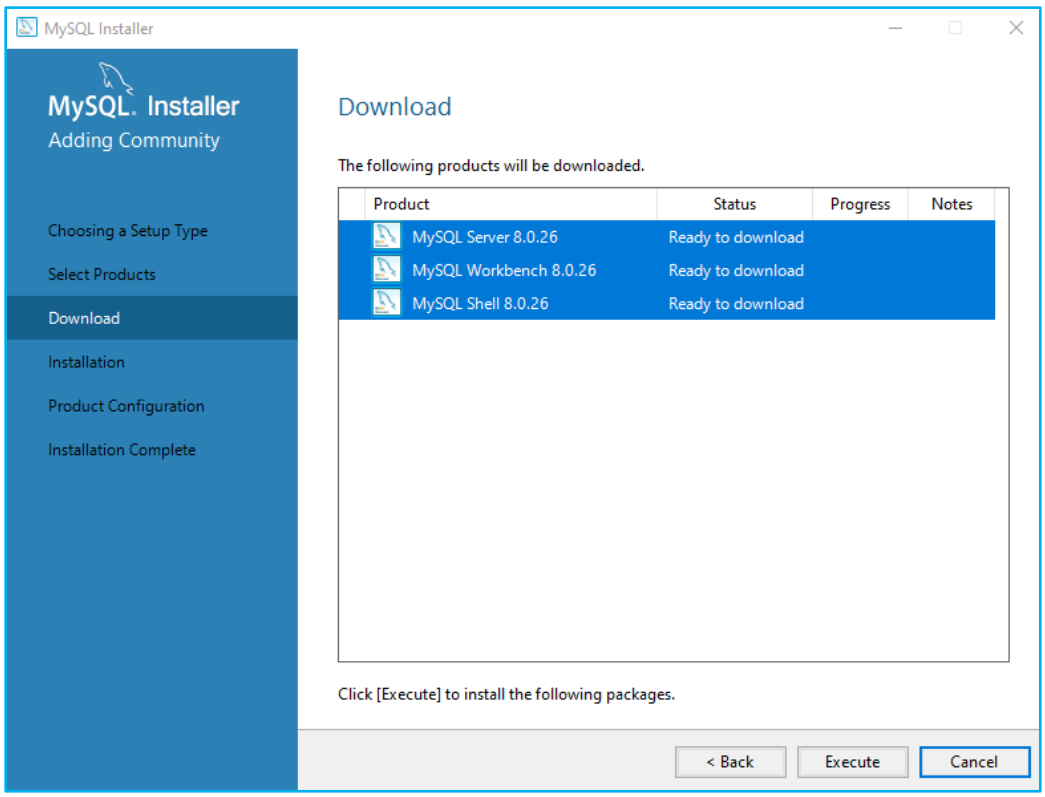
16/10/2021
Date of submission

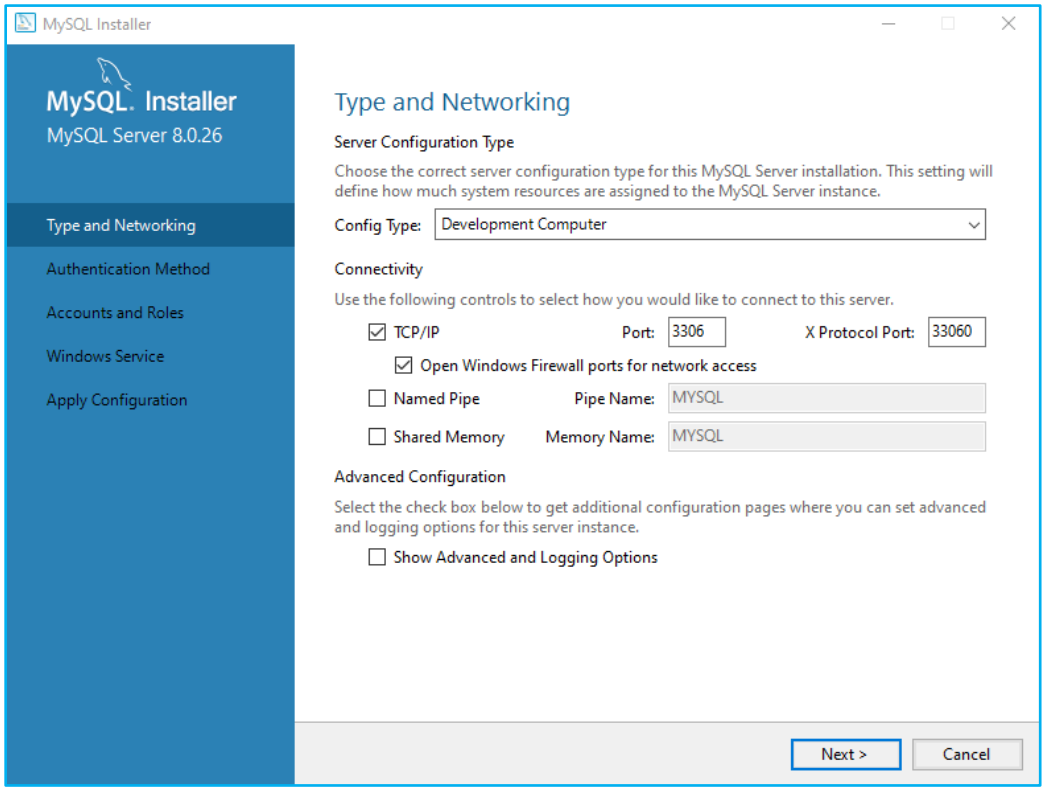
Security Check List :-

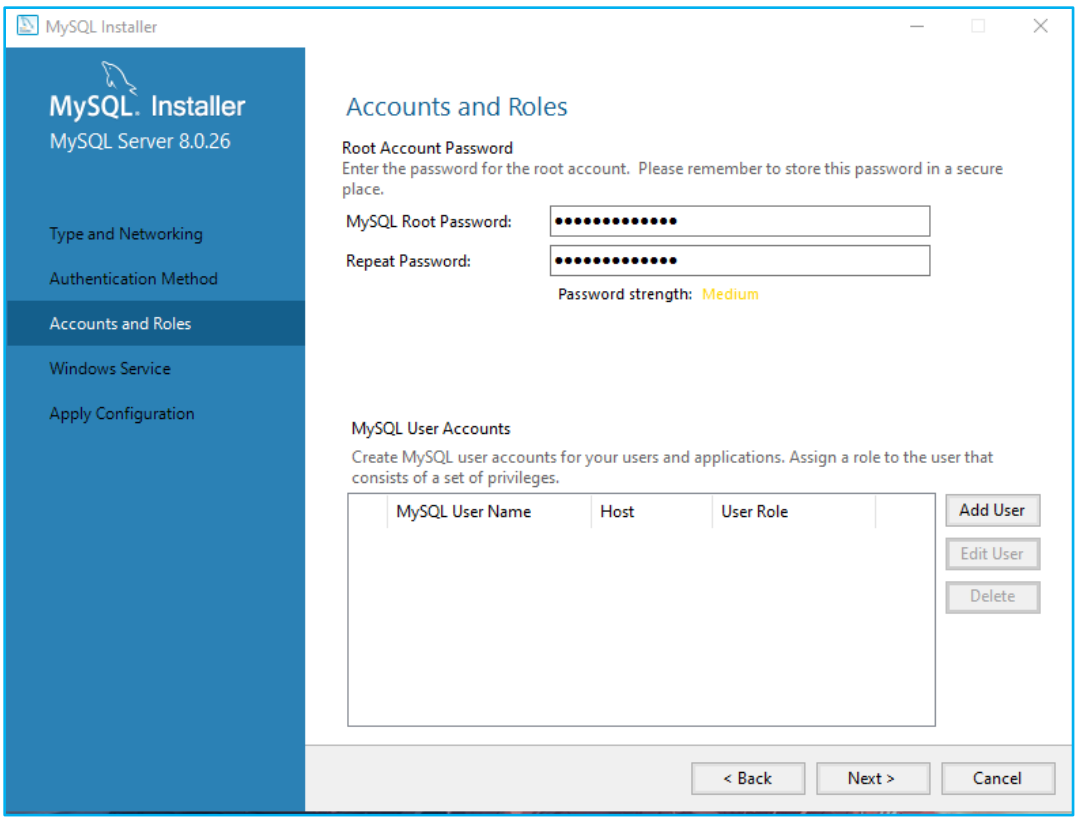
In this assignment, in order to enable the security features which are provided in the checklist, we were given the opportunity to choose a suitable operating system as per our wish. So, I have chosen **Windows Operating System** as my host operating system in order to perform this assignment. As the target database system, I have chosen the **MySQL** database system.

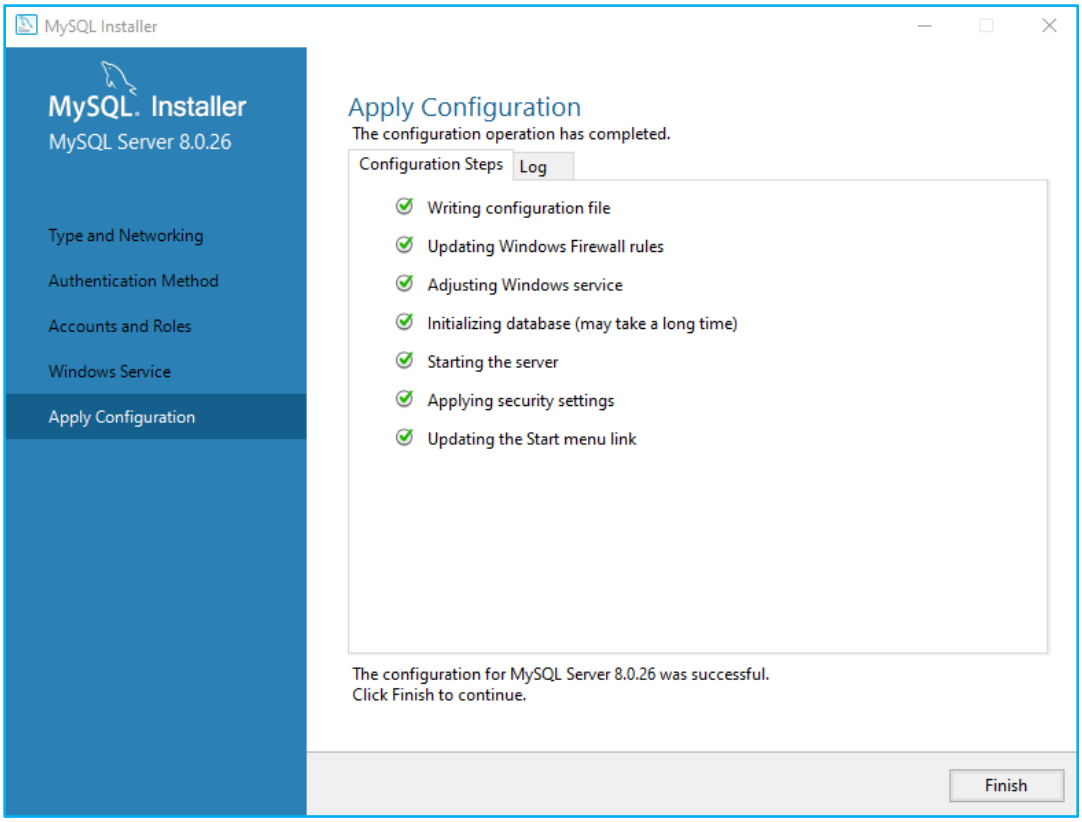
You can download the MySQL Community version for free from their official website itself. So, before anything else, let's see the installation process of the MySQL Server in my Windows host machine. When installing this particular database, I have chosen the custom installation setup.

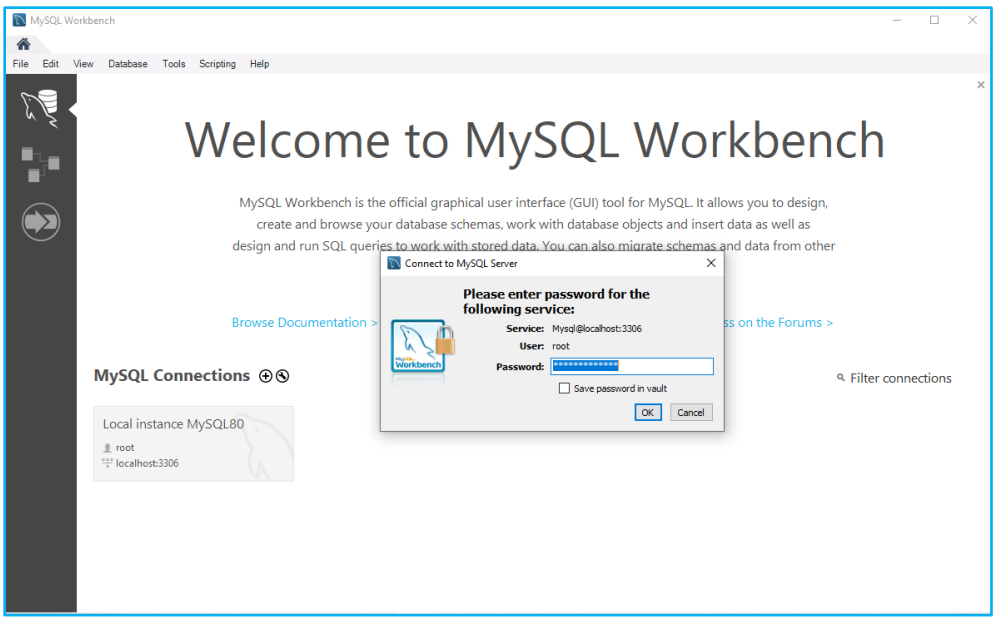



MySQL Installer
MySQL. Installer
Adding Community
Choosing a Setup Type
Select Products
Download
Installation
Product Configuration
Installation Complete
Download
The following products will be downloaded.
Product
Status
Progress
Notes
MySQL Server 8.0.26
Ready to download
MySQL Workbench 8.0.26
Ready to download
MySQL Shell 8.0.26
Ready to download
Click [Execute] to install the following packages.
< Back
Execute
Cancel

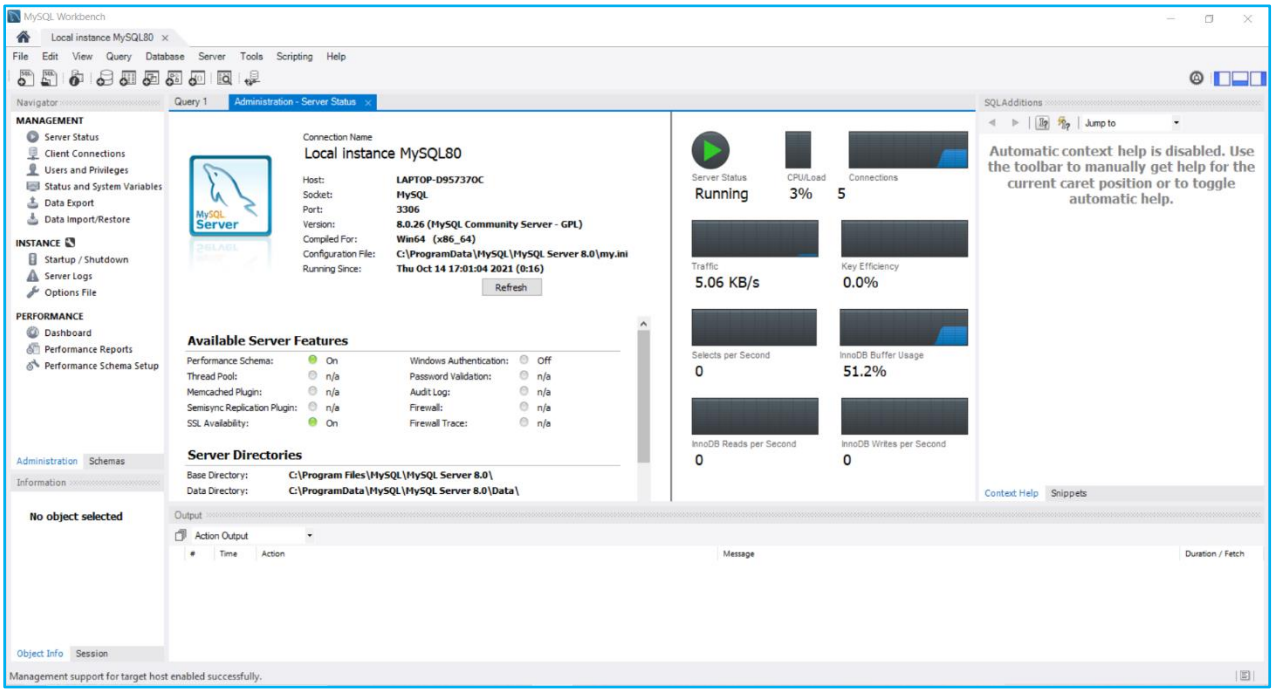

MySQL Installer
MySQL. Installer
MySQL Server 8.0.26
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Apply Configuration
Type and Networking
Server Configuration Type
Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance.
Config Type:
Development Computer
Connectivity
Use the following controls to select how you would like to connect to this server.
TCP/IP
Port:
3306
X Protocol Port:
33060
Open Windows Firewall ports for network access
Named Pipe
Pipe Name:
MYSQL
Shared Memory
Memory Name:
MYSQL
Advanced Configuration
Select the check box below to get additional configuration pages where you can set advanced and logging options for this server instance.
Show Advanced and Logging Options
Next >
Cancel


MySQL Installer
MySQL. Installer
MySQL Server 8.0.26
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Apply Configuration
Accounts and Roles
Root Account Password
Enter the password for the root account. Please remember to store this password in a secure place.
MySQL Root Password:
Repeat Password:
Password strength: Medium
MySQL User Accounts
Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.
MySQL User Name
Host
User Role
Add User
Edit User
Delete
< Back
Next >
Cancel


MySQL Installer
MySQL. Installer
MySQL Server 8.0.26
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Apply Configuration
Apply Configuration
The configuration operation has completed.
Configuration Steps
Log
Writing configuration file
Updating Windows Firewall rules
Adjusting Windows service
Initializing database (may take a long time)
Starting the server
Applying security settings
Updating the Start menu link
The configuration for MySQL Server 8.0.26 was successful.
Click Finish to continue.
Finish



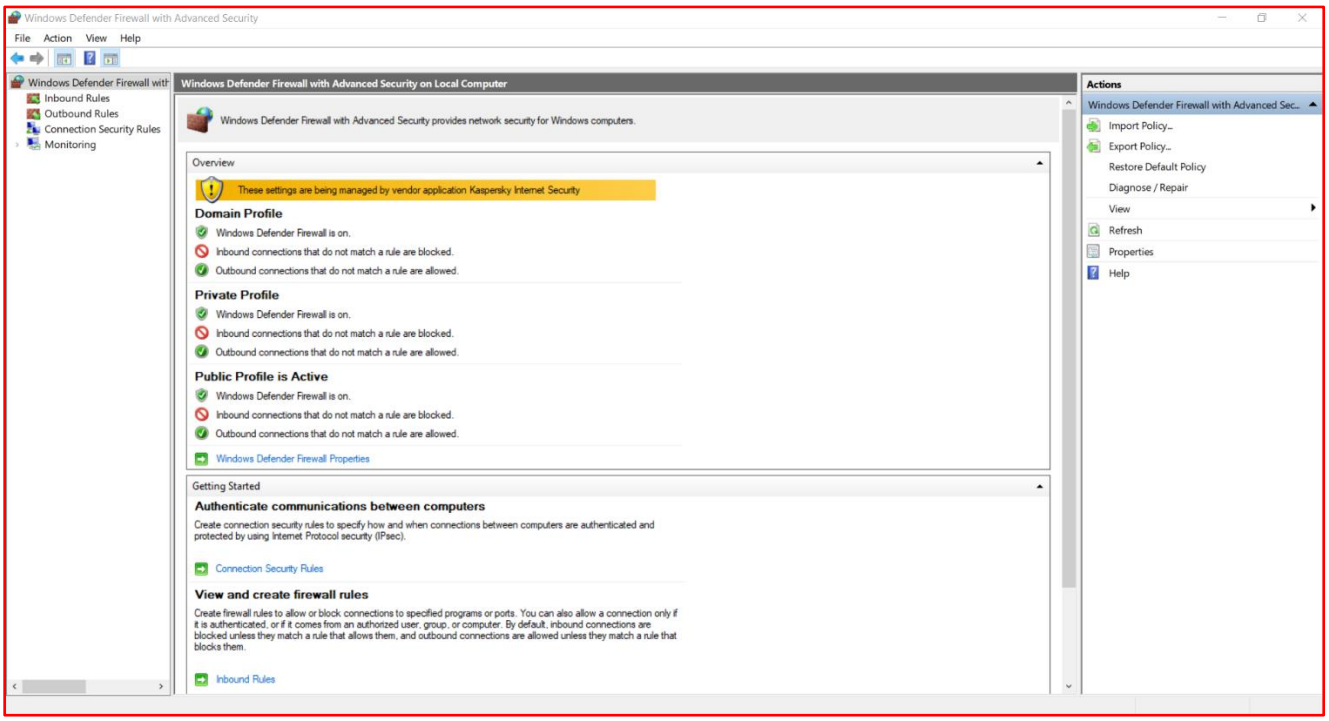
So, as you can see in the above image, I have successfully created a local database connection from the MySQL Workbench.



In here, when you go to the **Navigator → Server Status** Tab, you are able to witness the real-time status of the database server.

1) Firewall for database servers

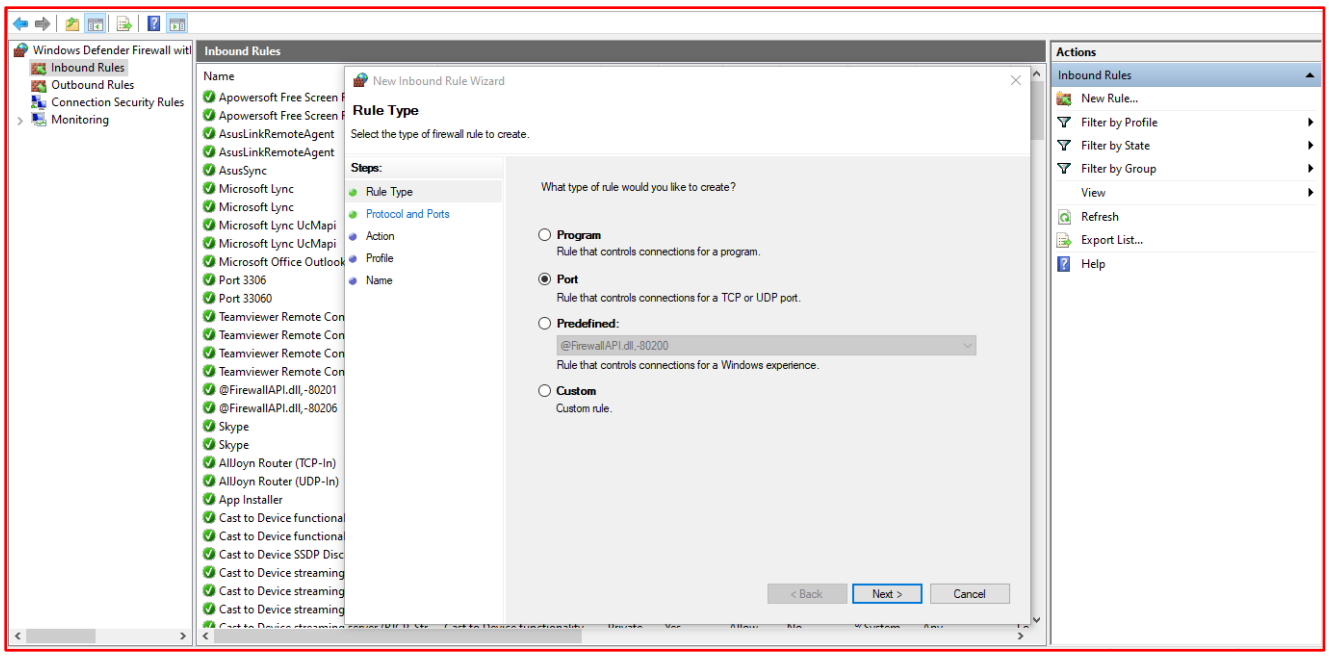
Since this is not the MySQL Enterprise version, it does not come with an inbuilt firewall for the database server. However, in order to configure the firewall rules, I have used the firewall settings on **Windows Defender Firewall** in my local computer.

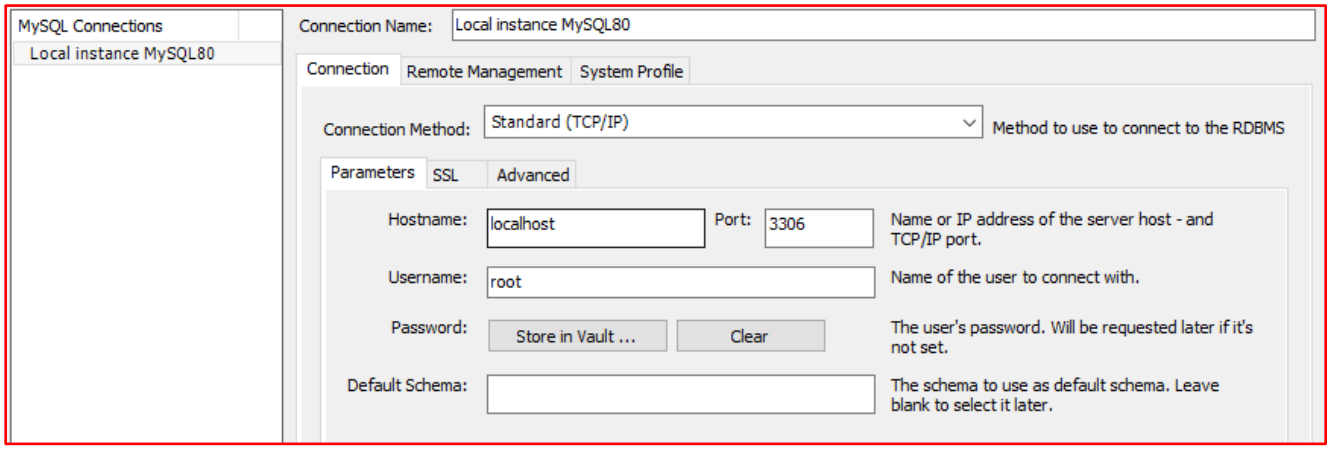


In this particular windows defender firewall settings, let's configure the inbound rules first.

So, in here, I'm going to configure inbound firewall rules in order to allow the MySQL application to access its database server.

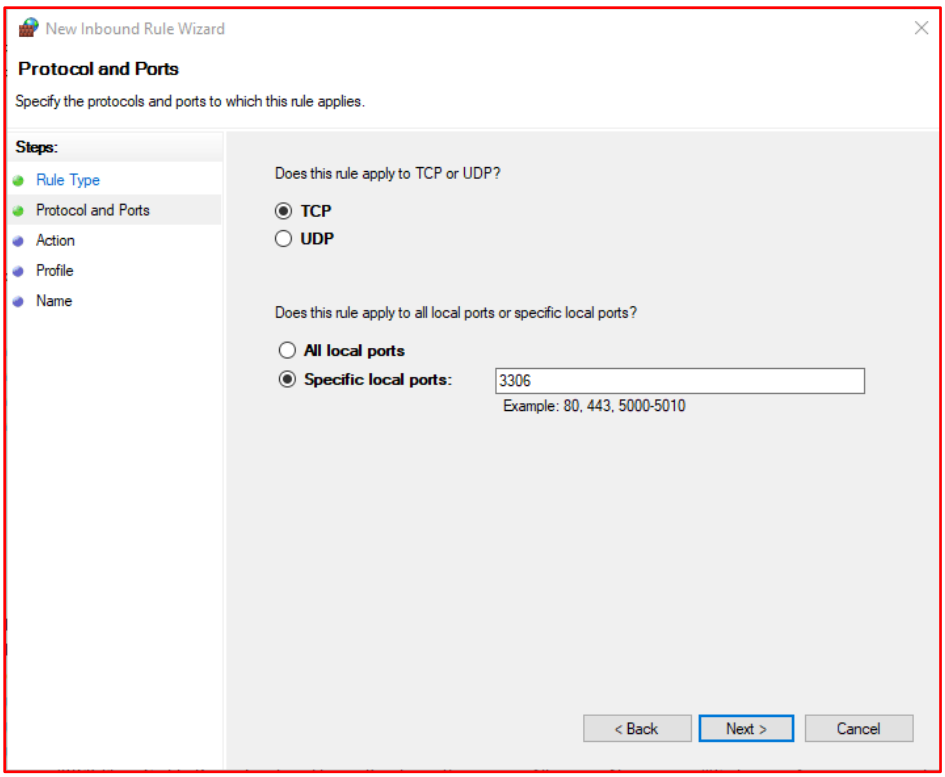
So, in here I am going to specify the ports that are required to control the TCP/UDP connections.

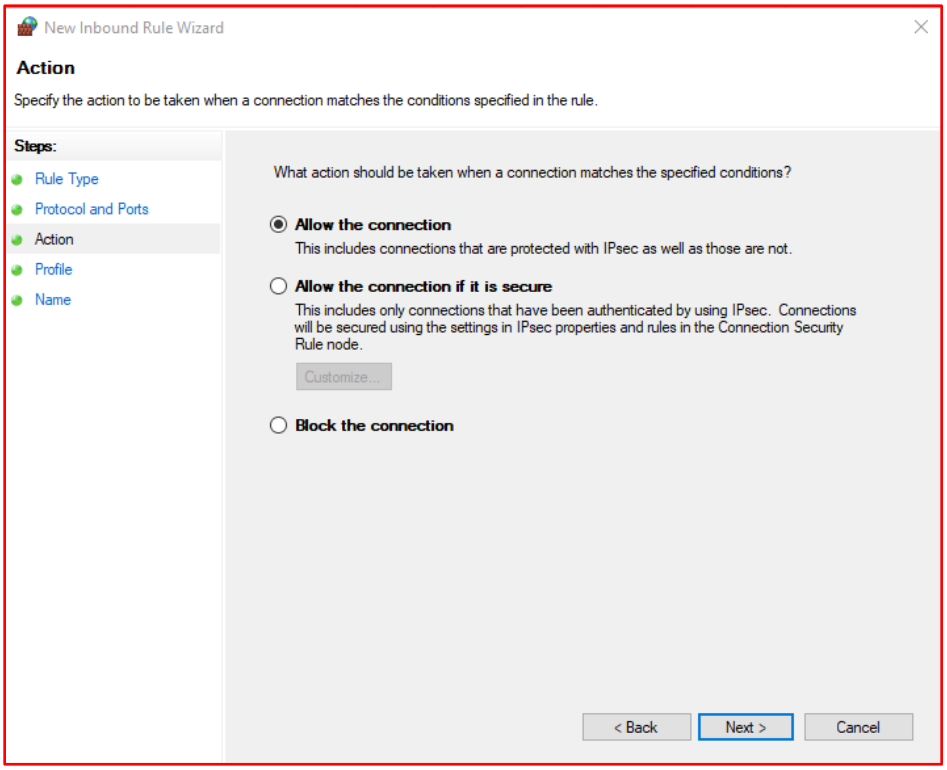




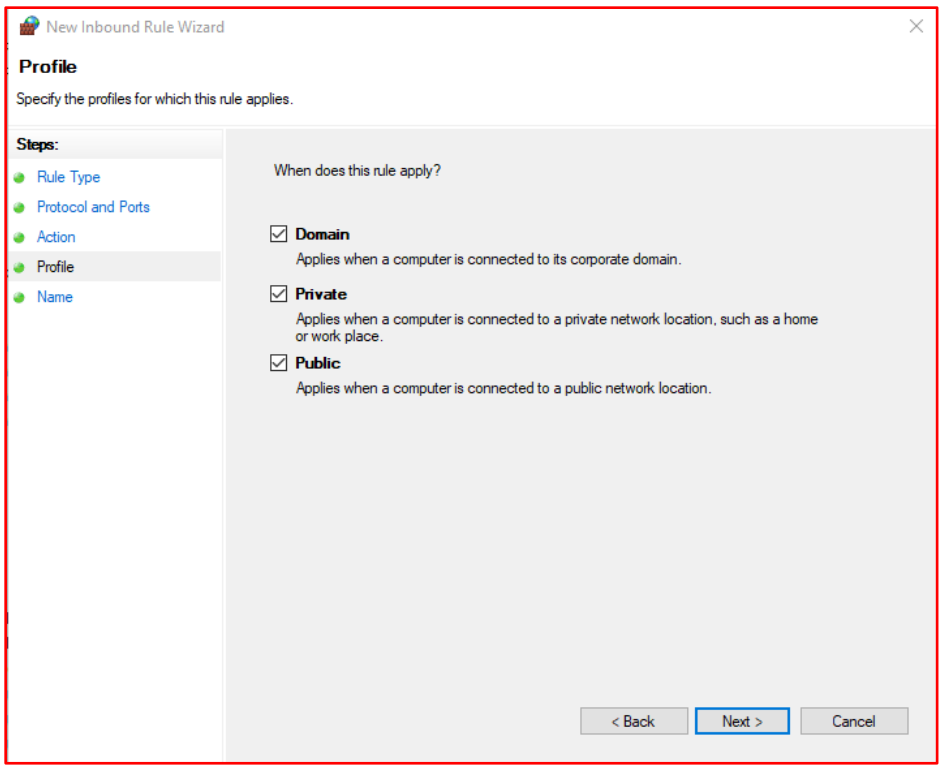
Since the specific port number for the MySQL Local connection is **3306**, I am going to use that port number when defining this firewall rule.

Then I need to choose **"Allow the connection"** option in order to allow the traffic for the MySQL database server.

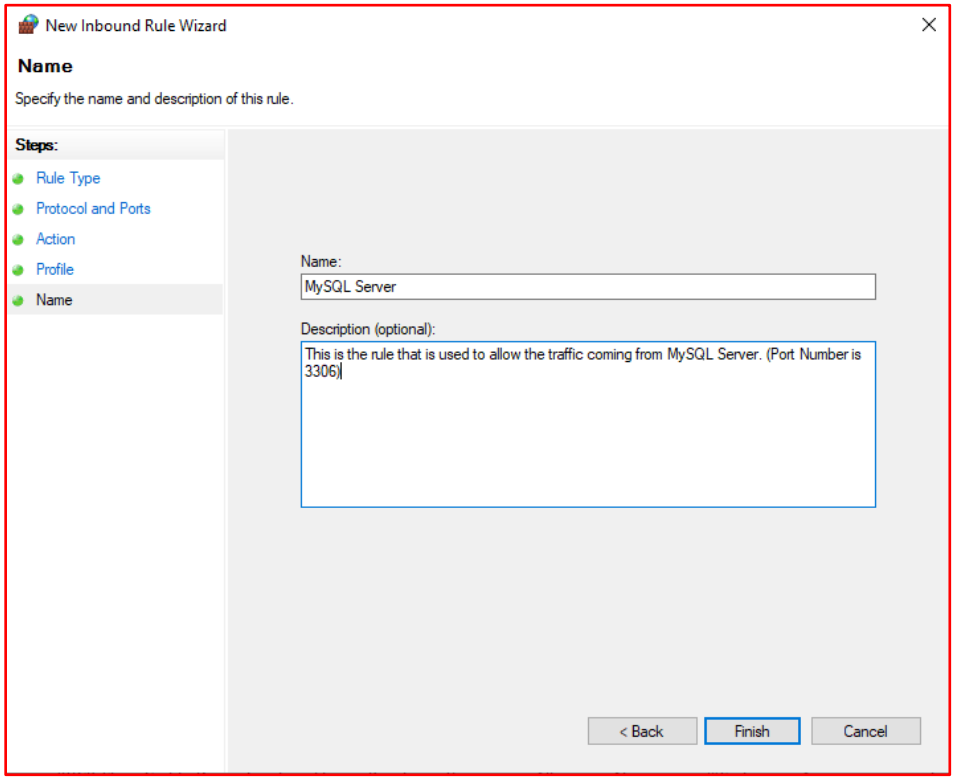




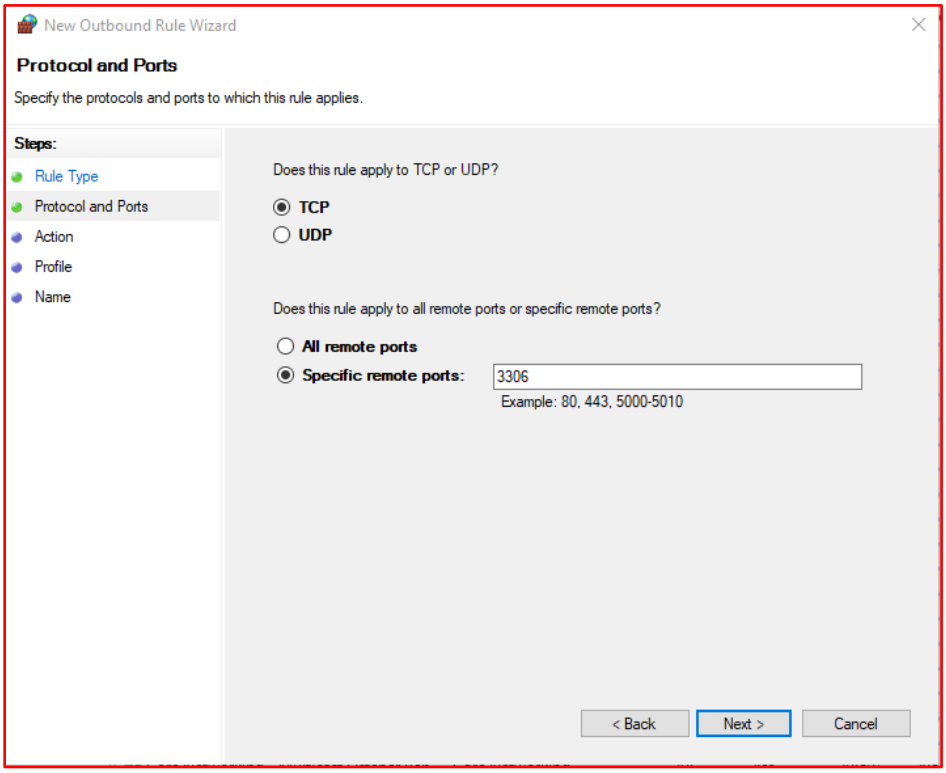
After that, you need to define **"For which network environments this rule needs to be applied to."**



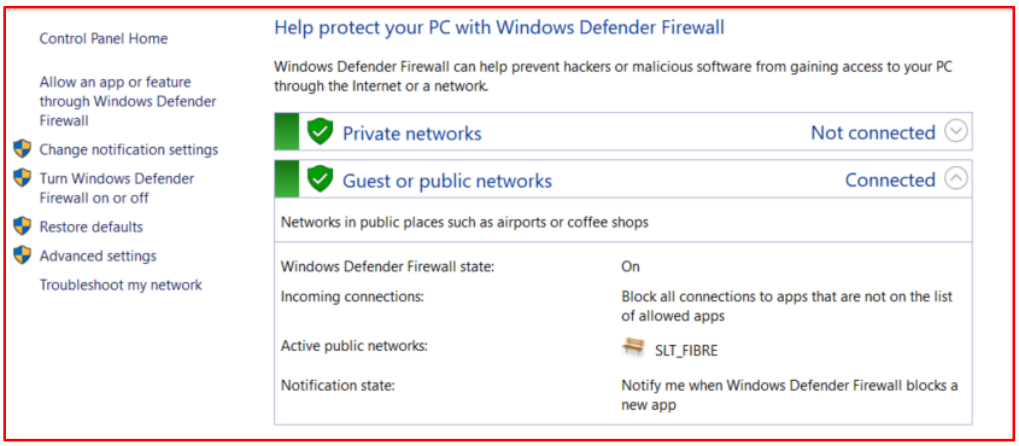
Then as the final step of configuring the inbound rule, you need to provide a **Name** & a **Description** for your particular firewall rule.

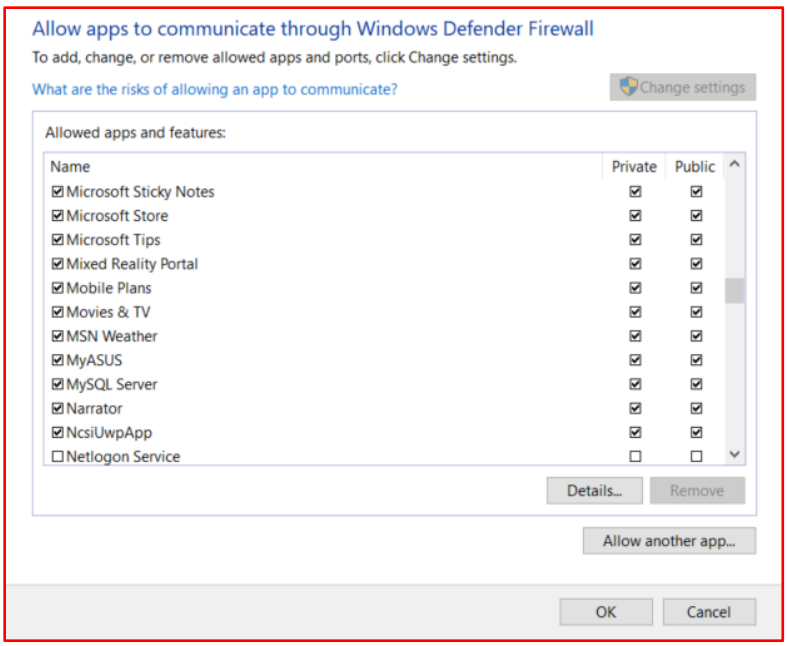


Now, let's move towards configuring the outbound firewall rules. In here also, you need follow the same procedure just like in configuring the inbound rules.



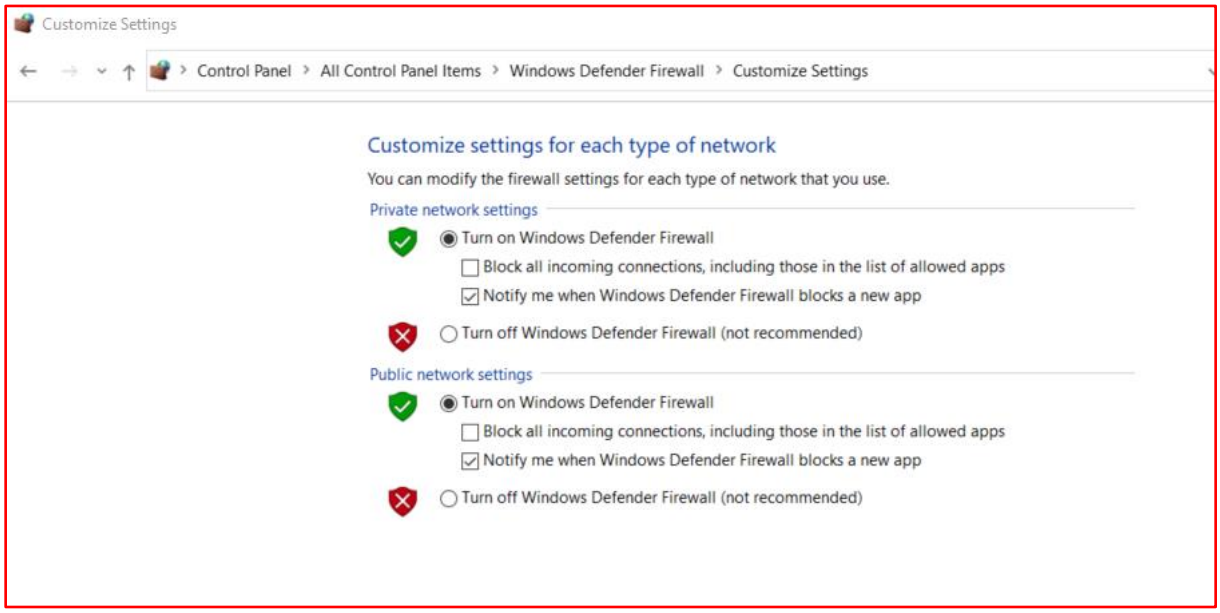
Furthermore, rather than only defining the firewall rules, you also need to manually allow the MySQL Database Server to transfer the traffic. You can perform it by selecting on the **"Allow an app or feature through Windows Defender Firewall"** option from the Windows Defender firewall settings.





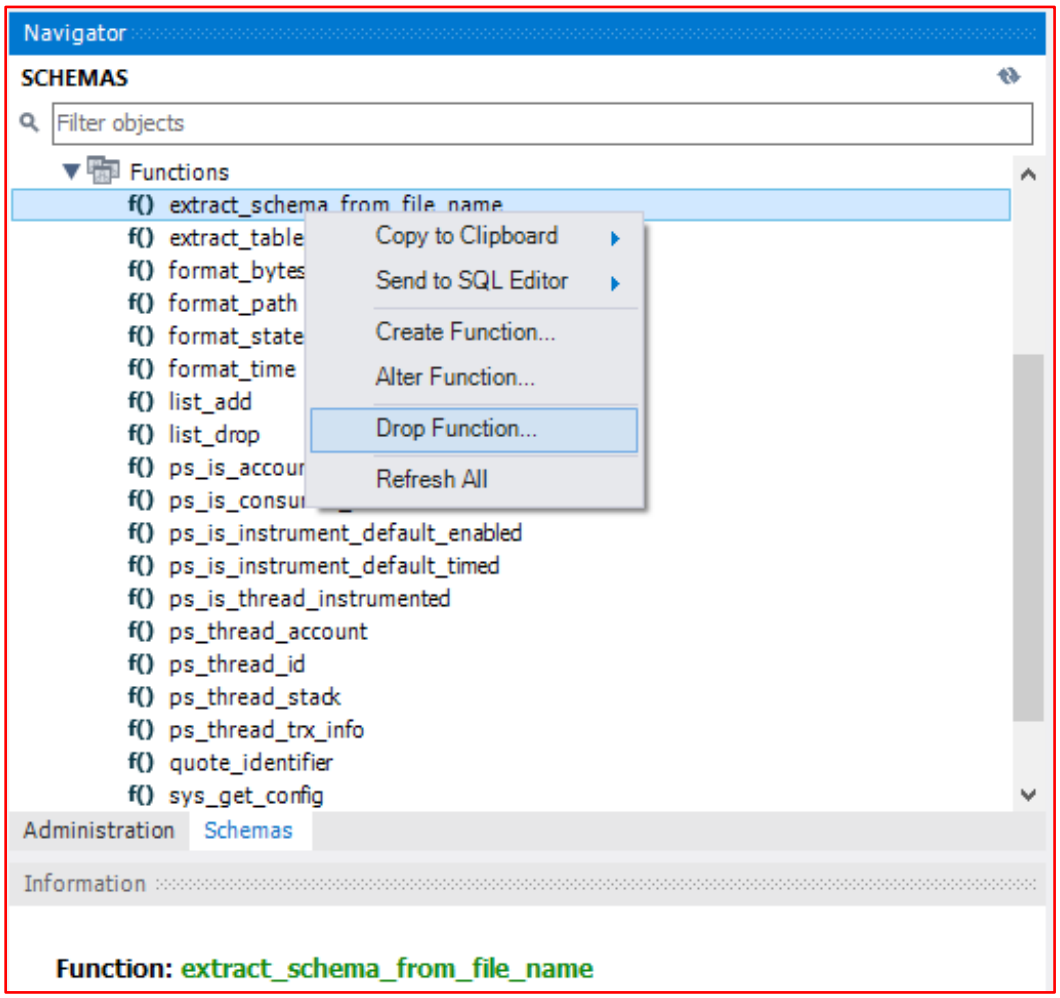
As it is demonstrated in the above image, you can perform it by selecting **Change Settings → Allow Another app → Browse the location of the app.**

Furthermore, you can enable notifications from the Windows Defender Firewall settings. By doing this it automatically sends a notification to the system administrator when a rule is modified in the firewall.



2) Database Software

In here, in order to remove the unnecessary functions, 1st you need to navigate to **Schema** tab. Then you need to click on the **Functions** drop down menu. From that drop down list, you can remove the unnecessary functions as per your consent.

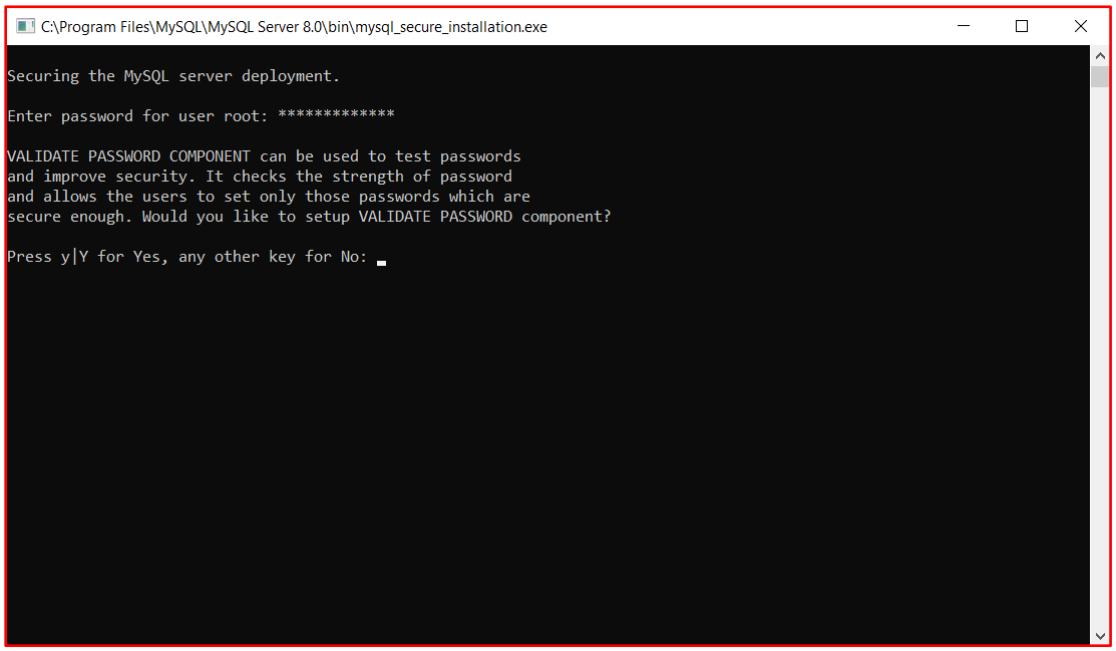


So, from this drop-down list, I have removed the following unnecessary functions.

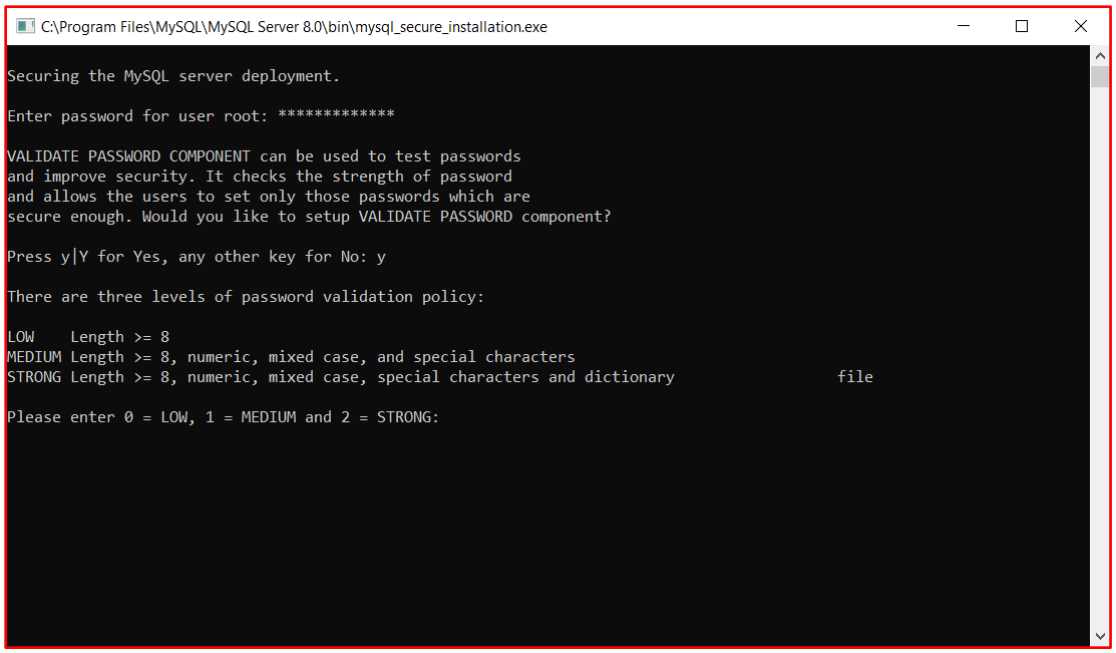
- ps_is_instrument_default_enabled
- ps_is_instrument_default_timed
- ps_is_thread_instrumented
- ps_thread_trx_info
- version_minor

Now, let's move on to changing the default passwords.

In order to perform this first you need to open up the **"mysql_secure_installation.exe"** CLI tool. You can browse it from **C:\Program Files\MySQL\MySQL Server 8.0\bin** location.



In here, you need to input **'y'** in here.



In here, you need to enter **“2”** in order to select the **STRONG** level.

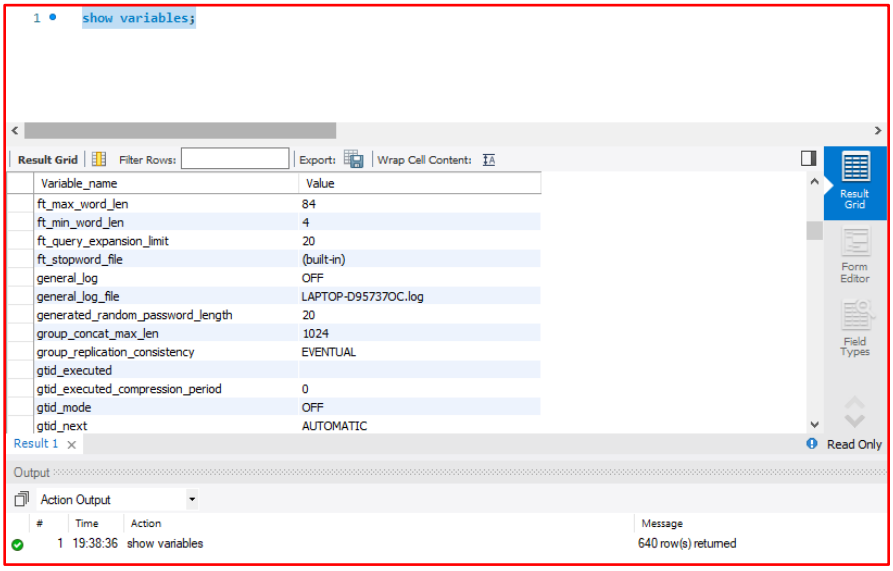
By doing so, you get the opportunity to remove unnecessary accounts.

After that I have changed my default root password to a much stronger password. Furthermore, I have disallowed the remote root login as well and also removed the test database & access for it.

```
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.


Normally, root should only be allowed to connect from
'localhost'. This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : y
Success.

By default, MySQL comes with a database named 'test' that
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.


Remove test database and access to it? (Press y|Y for Yes, any other key for No) : y
 - Dropping test database...
Success.

 - Removing privileges on test database...
Success.

Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) :
```

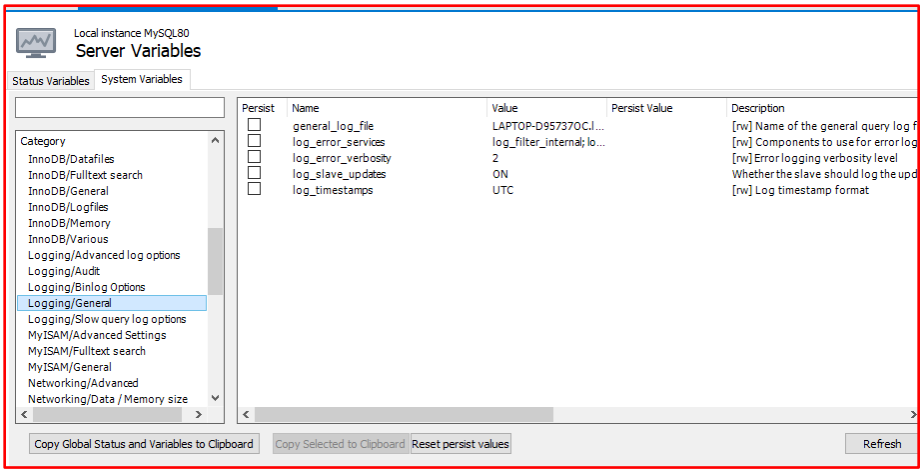
3) Maintain the log records of accessing to the database and maintain the minimum access privileges to the existing servers and applications.

When you give the query **“show variables”**, you are able to see all the variables that are being used in the MySQL Database Server. Inside those variables, you are able to find out the default location of the log file.

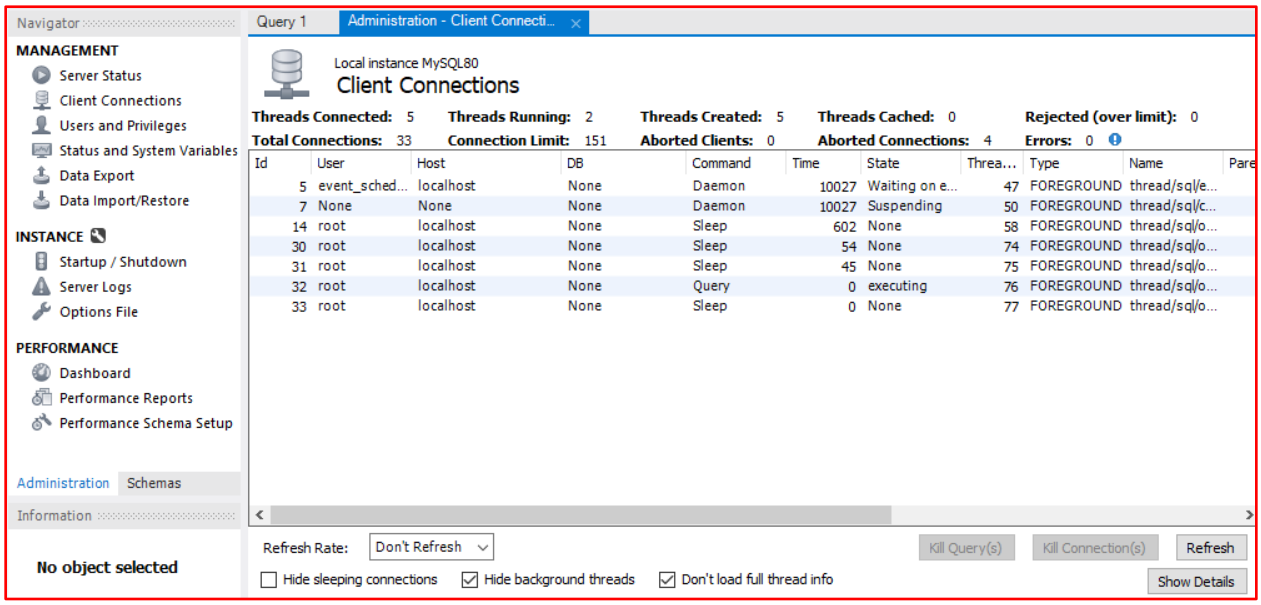


In order to turn on the **general_log** feature, you need to provide the following SQL Query as shown below.

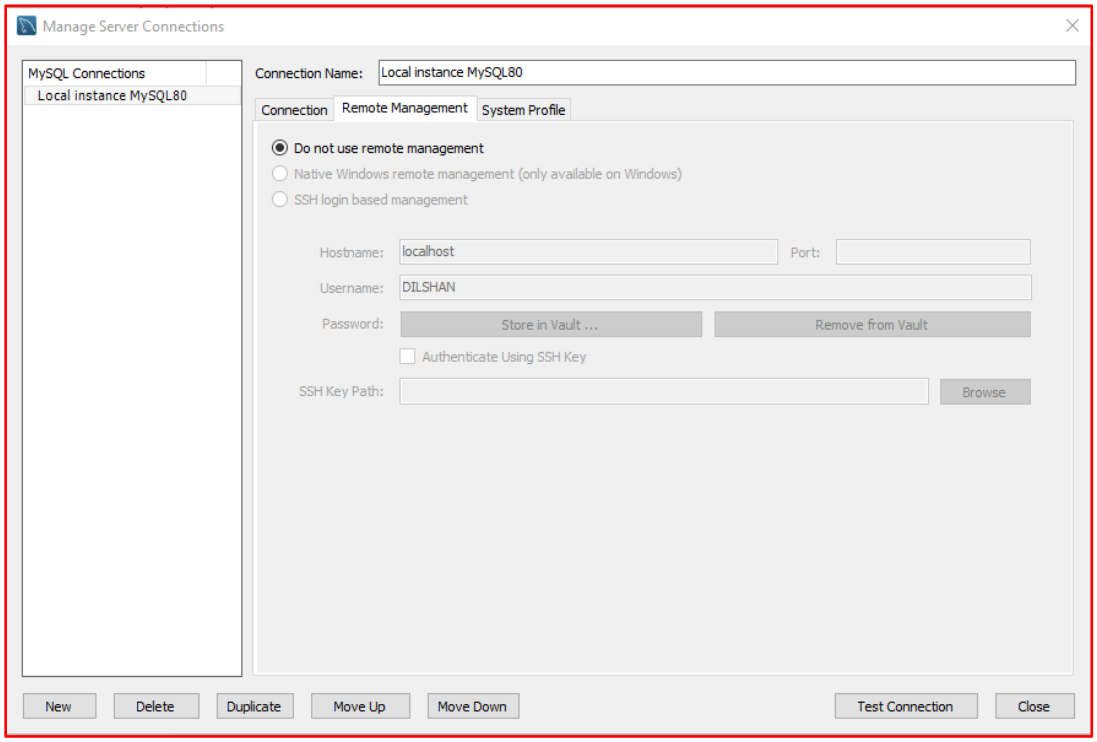
```
1 •   SET GLOBAL general_log = 'ON';
```



Furthermore, if you move towards the **Client Connections** option in the **Management** Tab, you are able to see all the necessary related information about the connections.



Additionally, you can change the sever connection settings for not allowing to use remote management.



4) Maintain individual login credentials for the people who access the workstation and to perform administrative tasks of the database.

When a particular database is considered, mainly there are 4 accounts that are allowed to perform administrative tasks on the database.

i. **SYS →** *Perform all kinds of administrative functions.*

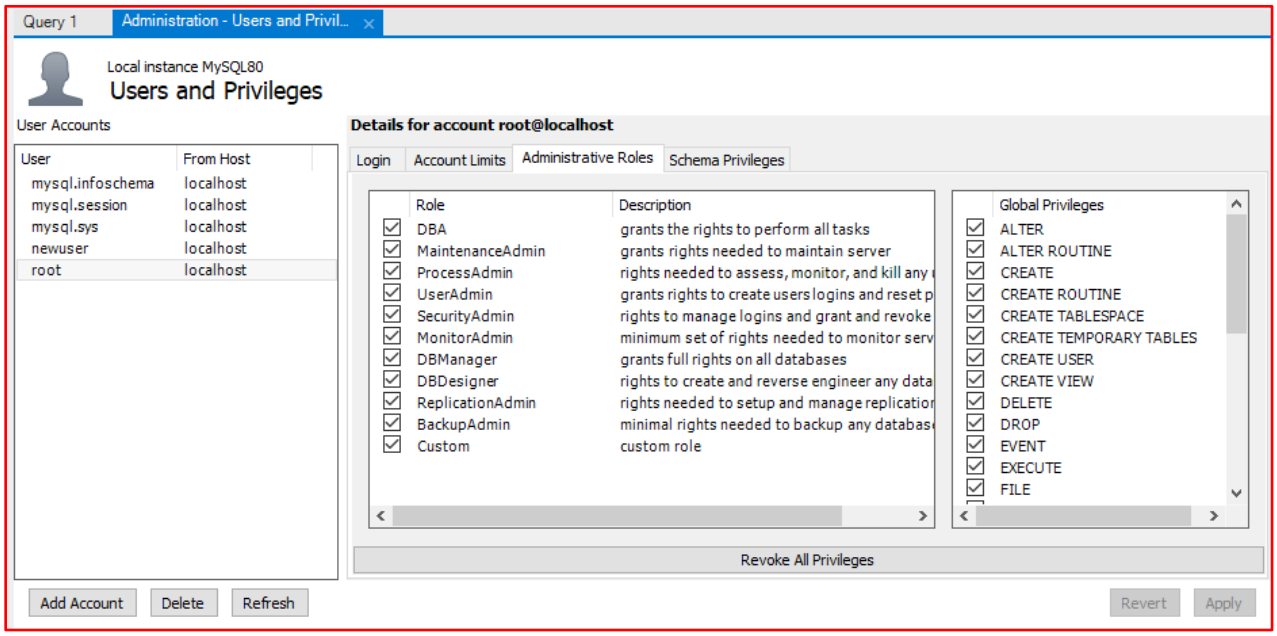
ii. **SYSTEM →** *Perform administrative functions except for* ***Backup & Recovery*** *and* ***Database Upgrade.***

iii. **SYSMAN →** *Set up and administer Enterprise Manager*

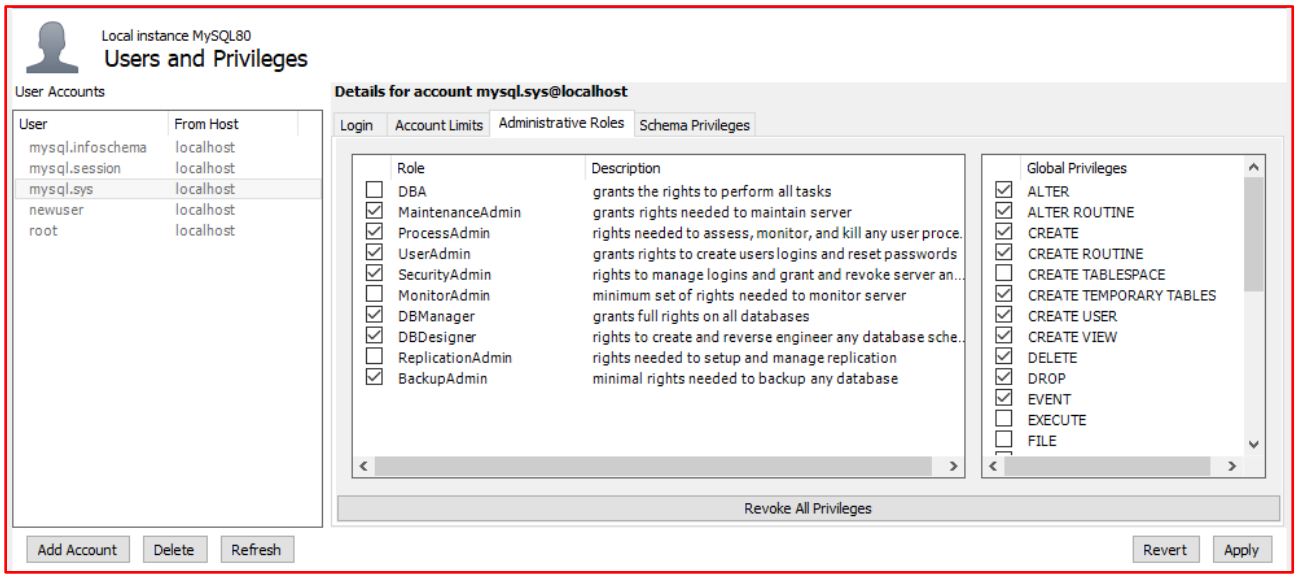
iv. **DBSNMP →** *Monitor & manage the database.*

Now, let's see what privileges those above-mentioned accounts possess.

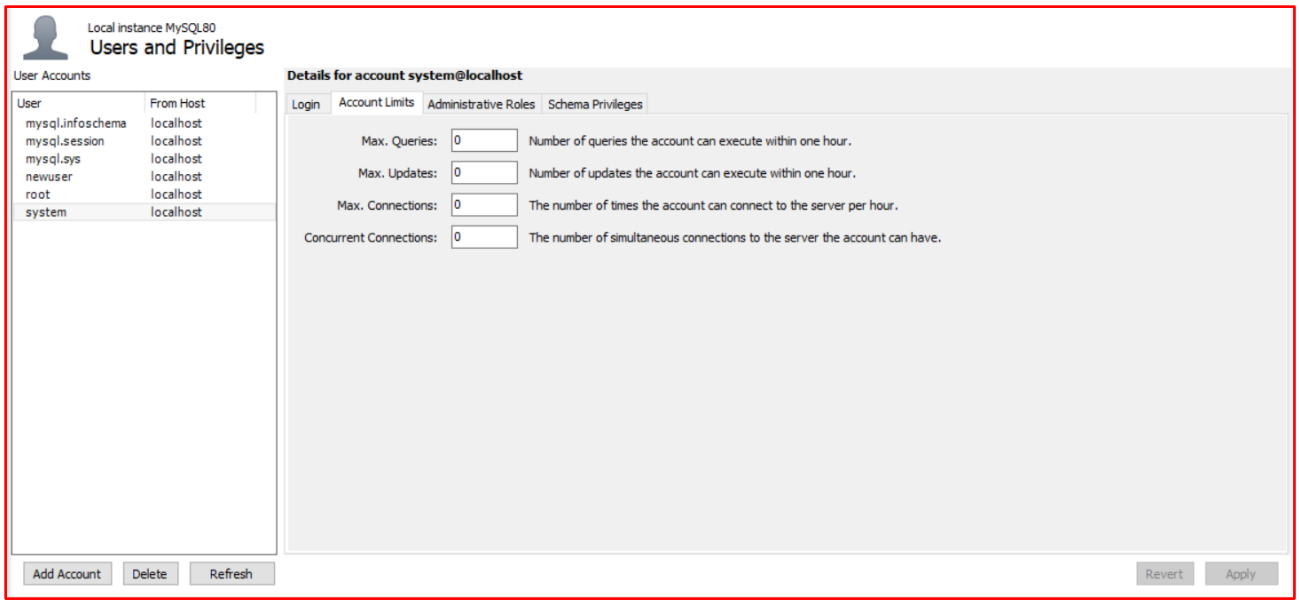
i. Root user



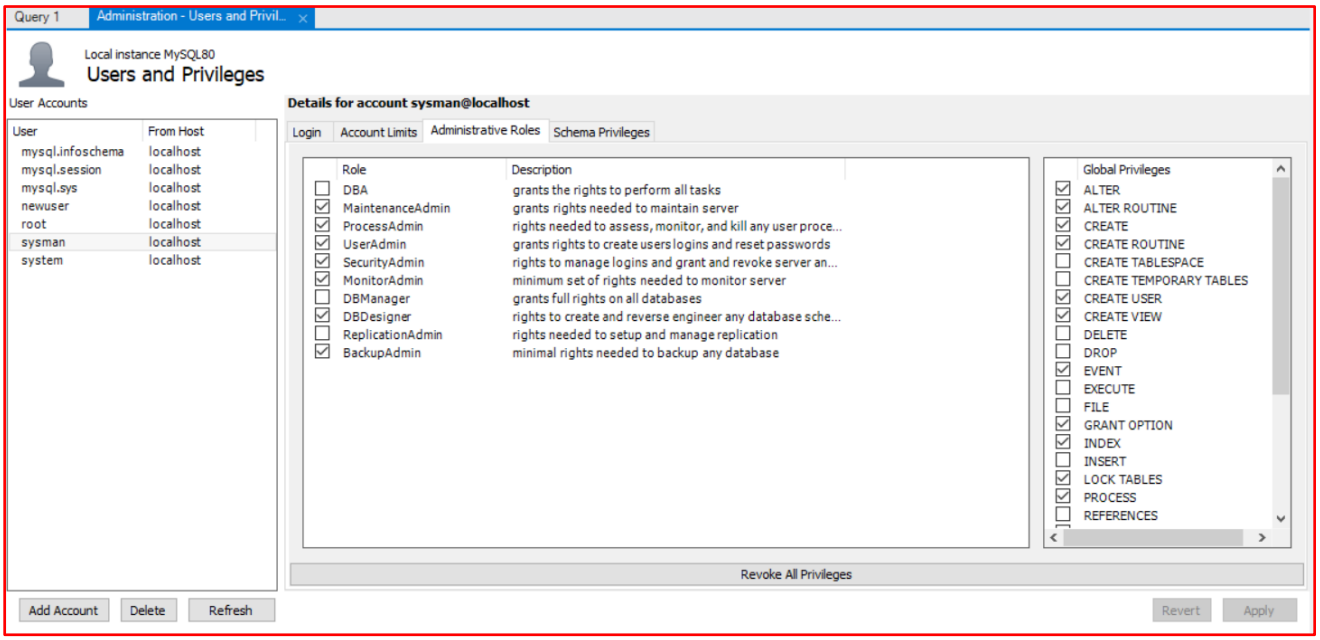
ii. Sys User



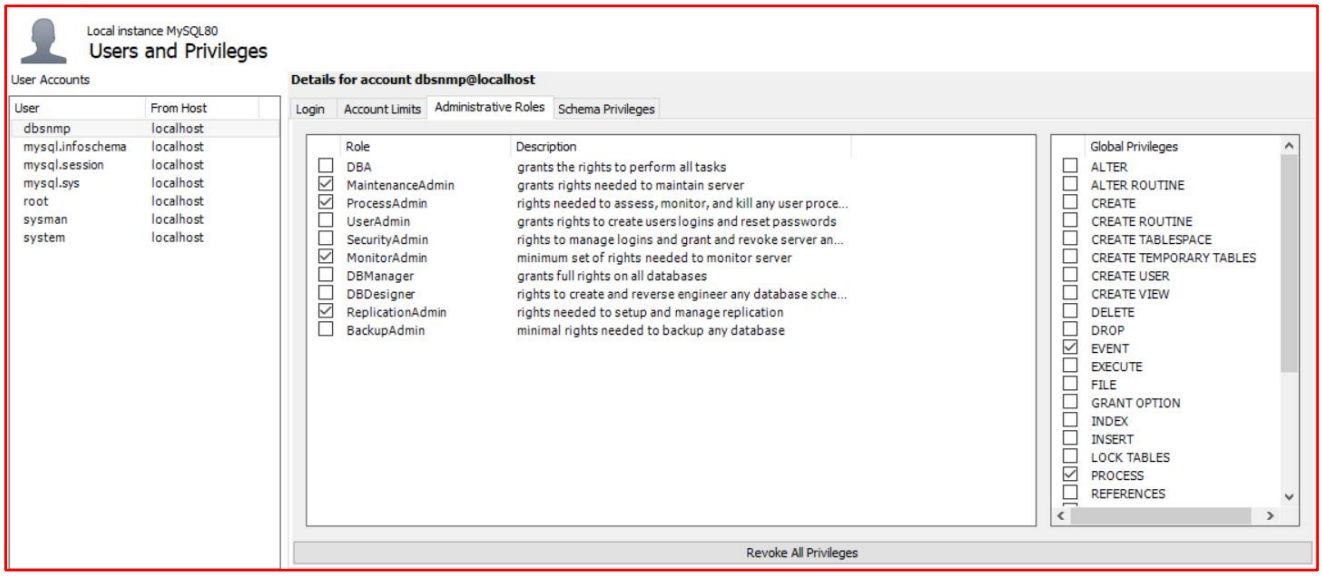
iii. System User



iv. Sysman User



v. Dbsnmp User



Now, let's change the login credentials of those administrative accounts. In here, I'm going to change all the other administrative accounts' passwords except for the root account.

```
mysql> ALTER USER 'mysql.sys'@'localhost' IDENTIFIED BY 'pa55w0rd';
Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> ALTER USER 'mysql.session'@'localhost' IDENTIFIED BY 'pa55w0rd2';
Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> ALTER USER 'system'@'localhost' IDENTIFIED BY 'pa55w0rd3';
Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER USER 'sysman'@'localhost' IDENTIFIED BY 'pa55w0rd4';
Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER USER 'dbsnmp'@'localhost' IDENTIFIED BY 'pa55w0rd4';
Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

5) **Grant minimal permissions that necessary for the people according to their job role in the database.**

6) **Permissions should be managed through roles or groups and not by direct grants to User IDs where possible.**

Before moving directly towards those 2 above questions, I have created a sample database of my own. The name of the database is **"MyRestaurantDB"**.

Since I am unable to show the entire database tables & records in here, following is a **sample set of** the SQL queries that I have used, the records that I have inserted and the tables I have created.

```sql
-- Creating a new database called "MyRestaurantDB" --
CREATE DATABASE MyRestaurantDB;

-- Using that particular database --
USE MyRestaurantDB;


-- CREATING TABLES--

-- Employee Table--

CREATE TABLE Employee(

                Emp_ID VARCHAR(20) NOT NULL,
                First_name VARCHAR(50),
                Last_name VARCHAR(50),
                House_number VARCHAR(50),
                Street VARCHAR(40),
                City VARCHAR(20),
                DOB DATE,

                CONSTRAINT Employee_PK PRIMARY KEY (Emp_ID)

                    );
```

```
-- Customer Table--

INSERT INTO Customer (Customer_ID, First_name, Last_name, House_number, Street, City, Payment_ID, Cart_ID, Emp_ID) VALUES ('CUS0
INSERT INTO Customer (Customer_ID, First_name, Last_name, House_number, Street, City, Payment_ID, Cart_ID, Emp_ID) VALUES ('CUS0
INSERT INTO Customer (Customer_ID, First_name, Last_name, House_number, Street, City, Payment_ID, Cart_ID, Emp_ID) VALUES ('CUS0
INSERT INTO Customer (Customer_ID, First_name, Last_name, House_number, Street, City, Payment_ID, Cart_ID, Emp_ID) VALUES ('CUS0
INSERT INTO Customer (Customer_ID, First_name, Last_name, House_number, Street, City, Payment_ID, Cart_ID, Emp_ID) VALUES ('CUS0


-- User_account Table --

INSERT INTO User_account (Username,password,Customer_ID) VALUES('jayani','jayani123','CUS001');
INSERT INTO User_account (Username,password,Customer_ID) VALUES('samadhi','samadhi123','CUS002');
INSERT INTO User_account (Username,password,Customer_ID) VALUES('eranda','eranda123','CUS003');
INSERT INTO User_account (Username,password,Customer_ID) VALUES('tharindu','tharindu123','CUS004');
INSERT INTO User_account (Username,password,Customer_ID) VALUES('chamodhi','chamodhi123','CUS005');


-- Orders Table --

INSERT INTO Orders(Order_ID,Order_Date,Delivery_Date,Total_Price,Order_Status,Customer_ID,Cart_ID,Emp_ID) VALUES('001','2019-09-
INSERT INTO Orders(Order_ID,Order_Date,Delivery_Date,Total_Price,Order_Status,Customer_ID,Cart_ID,Emp_ID) VALUES('002','2019-08-
INSERT INTO Orders(Order_ID,Order_Date,Delivery_Date,Total_Price,Order_Status,Customer_ID,Cart_ID,Emp_ID) VALUES('003','2019-08-
INSERT INTO Orders(Order_ID,Order_Date,Delivery_Date,Total_Price,Order_Status,Customer_ID,Cart_ID,Emp_ID) VALUES('004','2019-10-
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Tables_in_myrestaurantdb
customer_c_email
customer_phone_number
db_admin
delivery
delivery_food_item
delivery_person
employee
employee_email
employee_phone
food_item
manager
orders
payment
receptionist
user_account
user_credentials
waiter
waiter_food_item

Result 16 | Read Only

DOSS1* | Limit to 1000 rows

```
select * from Employee;

-- Creating a new database called "MyRestaurantDB" --
CREATE DATABASE MyRestaurantDB;
```

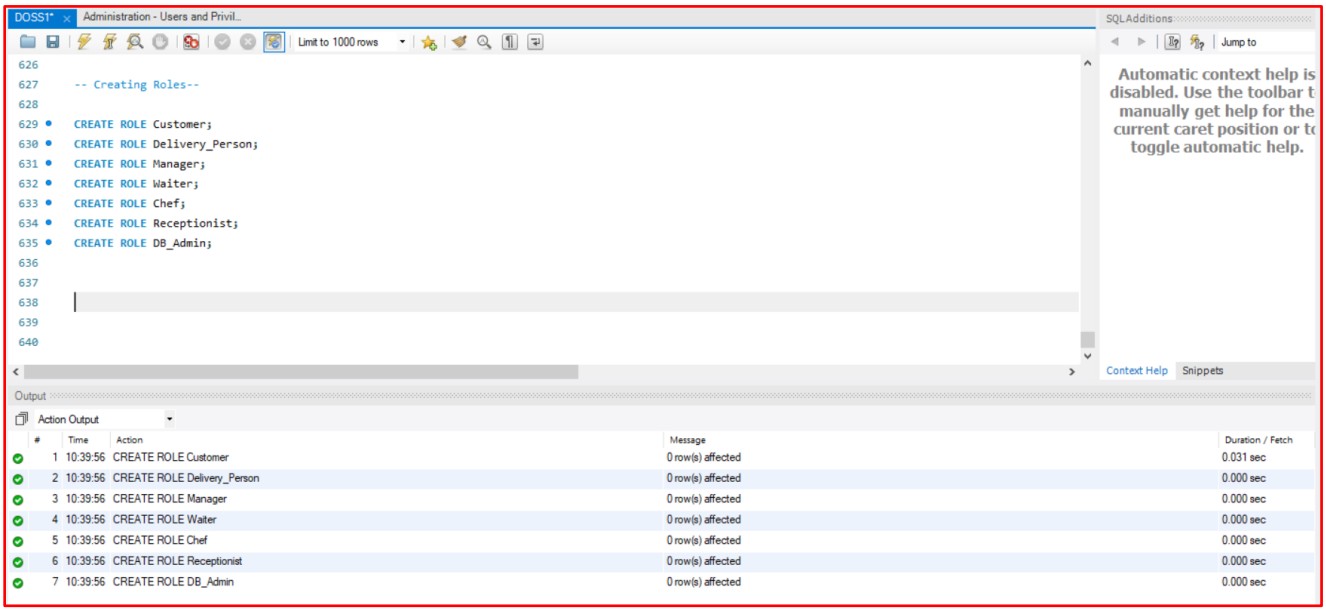
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Emp_ID	First_name	Last_name	House_number	Street	City	DOB
M1050	Eishan	Dinuka	14/A	3rd Lane	Gampaha	1973-06-20
M1078	Hashan	Perera	13/B	2nd Lane	Kandy	1994-04-05
M1111	Kavinga	Munidasa	13/A	1st Lane	Colombo	1991-08-22
M1211	Inura	Kumara	1/A	3rd Lane	Nuwaraeliya	1985-06-20
M1234	Amal	Perera	10/A	1st Lane	Gampaha	1975-01-11
M1235	Tirone	Silva	5/B	2nd Lane	Rathnapura	1990-08-08
M1311	Janith	Sankalpa	13/C	6th Lane	Colombo	1986-08-22
M1334	Kavinga	Herath	12/A	1st Lane	Colombo	1992-08-22
M1357	Nimal	Kumara	15/A	3rd Lane	Gampaha	1976-06-20
M1358	Avishka	Fernando	5/A	1st Lane	Hambanthota	1991-05-05
M1411	Sandul	Lakshan	2/B	8th Lane	Mathara	1987-03-14
M1434	Lahiru	Ambegoda	13/B	2nd Lane	Ampara	1994-03-14
M1511	Sanuka	Wickramas...	4/B	1st Lane	Galle	1988-05-23
M1534	Bhanuka	Udawatte	12/B	1st Lane	Kelaniya	1985-05-23
M1611	Pasan	Herath	3/A	3rd Lane	Polonnaruwa	1989-02-28
M1634	Nawodya	Soysa	12/C	10th Lane	Rathnapura	1983-02-28
M1750	Kasun	Kalhara	16/B	2nd Lane	Kaluthara	1973-03-14
M1950	Amara	Perera	15/C	5th Lane	Colombo	1979-08-22
M20	Dilakshi	Sandumini	3/A	3rd Lane	Polonnaruwa	1975-02-28

Employee 17 | Apply | Revert

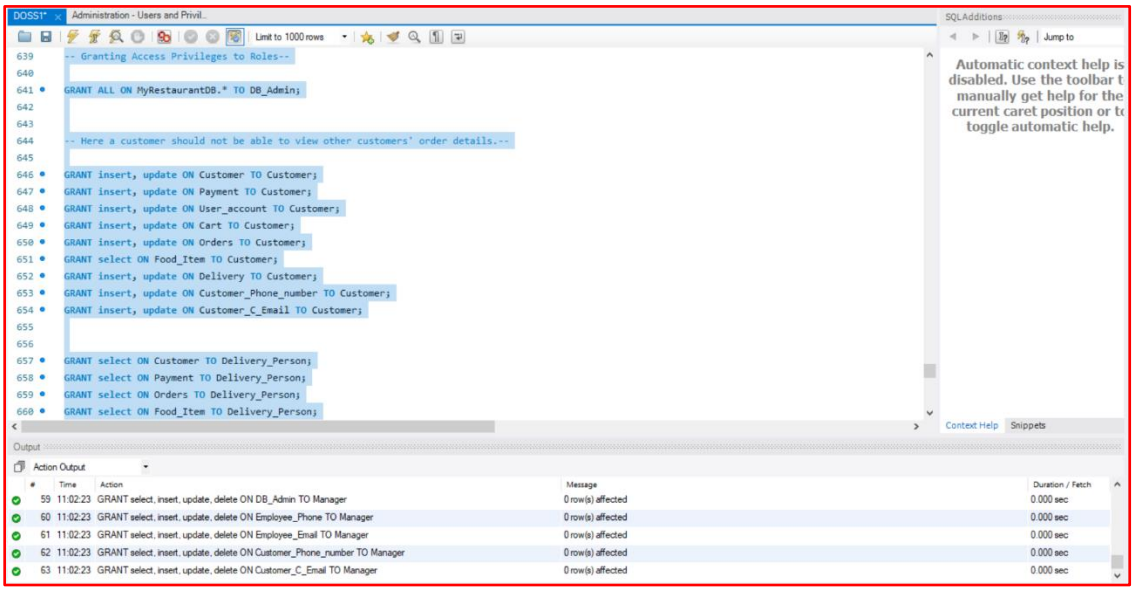
Step 1 :- Creating Roles

As the very 1st step, I have created 7 different roles for the database that I have created.



Step 2 :- Granting Access Privileges to Roles

As the next step, you need to grant the necessary access privileges **for the roles** you have created before.



```
GRANT select ON Orders TO Waiter;
GRANT select ON Food_Item TO Waiter;
GRANT select ON Waiter TO Waiter;
GRANT select ON Employee_Phone TO Waiter;


GRANT select ON Customer TO Chef;
GRANT select ON Orders TO Chef;
GRANT select ON Cart TO Chef;
GRANT select ON Chef TO Chef;
GRANT select ON Employee_Phone TO Chef;


GRANT select ON Customer TO Receptionist;
GRANT select ON Payment TO Receptionist;
GRANT select ON Cart TO Receptionist;
GRANT select ON Orders TO Receptionist;
GRANT select ON Food_Item TO Receptionist;
GRANT select ON Delivery TO Receptionist;
GRANT select ON Employee TO Receptionist;
GRANT select ON Delivery_Person TO Receptionist;
GRANT select ON Manager TO Receptionist;
```

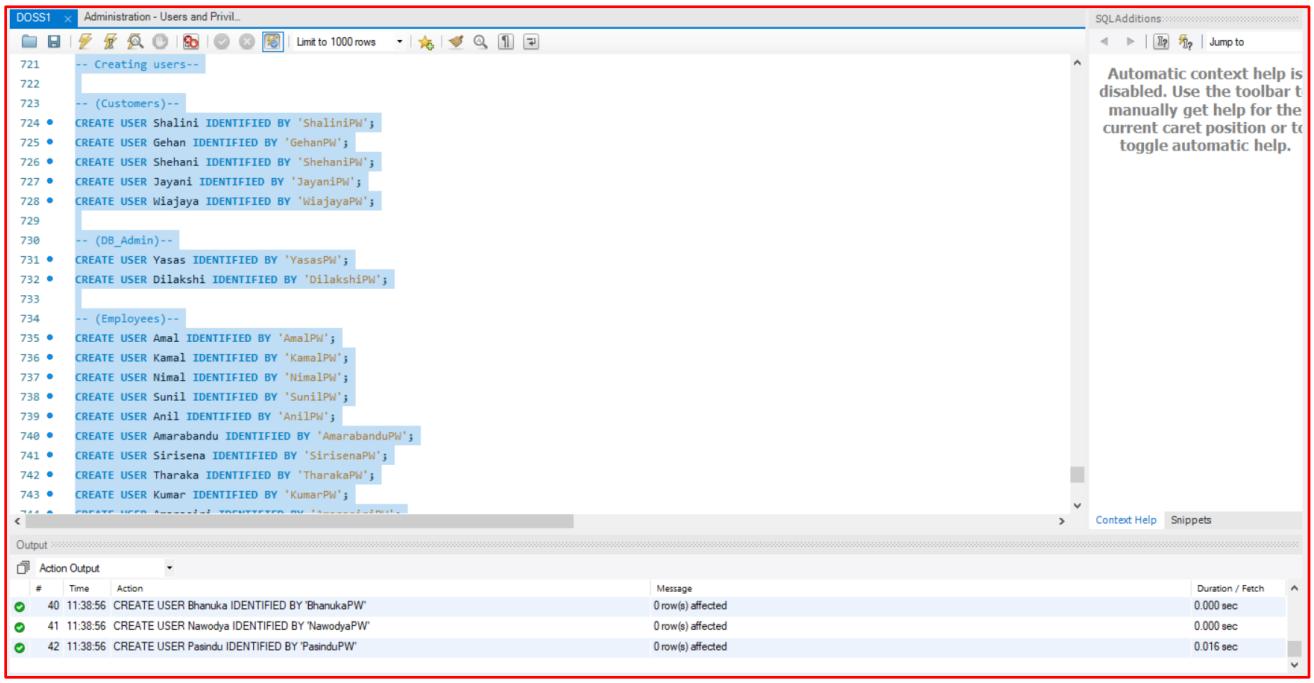
#	Time	Action	Message	Duration / Fetch
59	11:02:23	GRANT select, insert, update, delete ON DB_Admin TO Manager	0 row(s) affected	0.000 sec
60	11:02:23	GRANT select, insert, update, delete ON Employee_Phone TO Manager	0 row(s) affected	0.000 sec
61	11:02:23	GRANT select, insert, update, delete ON Employee_Email TO Manager	0 row(s) affected	0.000 sec
62	11:02:23	GRANT select, insert, update, delete ON Customer_Phone_number TO Manager	0 row(s) affected	0.000 sec
63	11:02:23	GRANT select, insert, update, delete ON Customer_C_Email TO Manager	0 row(s) affected	0.000 sec

```
GRANT select ON Employee_Email TO Receptionist;
GRANT select ON Customer_Phone_number TO Receptionist;
GRANT select ON Customer_C_Email TO Receptionist;


GRANT select, insert, update, delete ON Customer TO Manager;
GRANT select, insert, update, delete ON Payment TO Manager;
GRANT select, insert, update, delete ON Cart TO Manager;
GRANT select, insert, update, delete ON Orders TO Manager;
GRANT select, insert, update, delete ON Food_Item TO Manager;
GRANT select, insert, update, delete ON Delivery TO Manager;
GRANT select, insert, update, delete ON Employee TO Manager;
GRANT select, insert, update, delete ON Delivery_Person TO Manager;
GRANT select, insert, update, delete ON Manager TO Manager;
GRANT select, insert, update, delete ON Waiter TO Manager;
GRANT select, insert, update, delete ON Chef TO Manager;
GRANT select, insert, update, delete ON Receptionist TO Manager;
GRANT select, insert, update, delete ON DB_Admin TO Manager;
GRANT select, insert, update, delete ON Employee_Phone TO Manager;
GRANT select, insert, update, delete ON Employee_Email TO Manager;
GRANT select, insert, update, delete ON Customer_Phone_number TO Manager;
GRANT select, insert, update, delete ON Customer_C_Email TO Manager;
```

#	Time	Action	Message	Duration / Fetch
59	11:02:23	GRANT select, insert, update, delete ON DB_Admin TO Manager	0 row(s) affected	0.000 sec
60	11:02:23	GRANT select, insert, update, delete ON Employee_Phone TO Manager	0 row(s) affected	0.000 sec
61	11:02:23	GRANT select, insert, update, delete ON Employee_Email TO Manager	0 row(s) affected	0.000 sec
62	11:02:23	GRANT select, insert, update, delete ON Customer_Phone_number TO Manager	0 row(s) affected	0.000 sec
63	11:02:23	GRANT select, insert, update, delete ON Customer_C_Email TO Manager	0 row(s) affected	0.000 sec

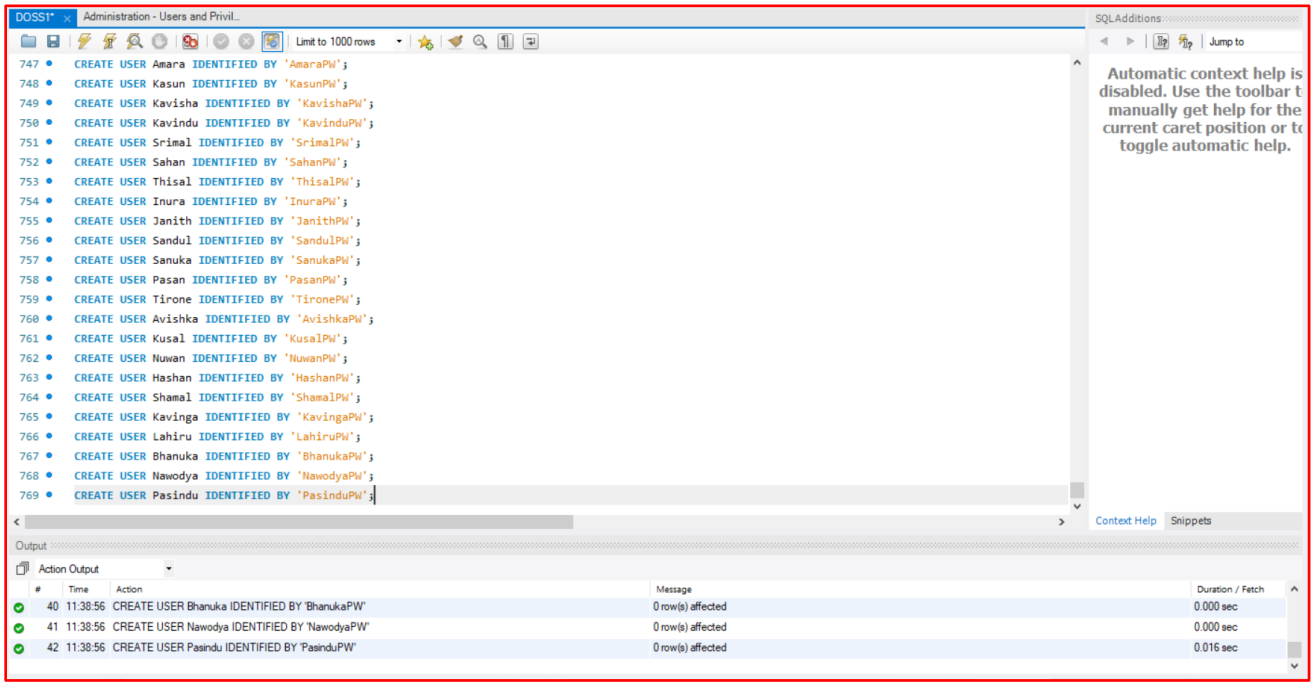
Step 3 :- Creating Users

In the next step, you need to create some users with their respective account passwords. So that, we can use those users later onwards to assign them with the necessary roles that we have created in the previous step.



```
-- Creating users--

-- (Customers)--
CREATE USER Shalini IDENTIFIED BY 'ShaliniPW';
CREATE USER Gehan IDENTIFIED BY 'GehanPW';
CREATE USER Shehani IDENTIFIED BY 'ShehaniPW';
CREATE USER Jayani IDENTIFIED BY 'JayaniPW';
CREATE USER Wiajaya IDENTIFIED BY 'WiajayaPW';

-- (DB_Admin)--
CREATE USER Yasas IDENTIFIED BY 'YasasPW';
CREATE USER Dilakshi IDENTIFIED BY 'DilakshiPW';

-- (Employees)--
CREATE USER Amal IDENTIFIED BY 'AmalPW';
CREATE USER Kamal IDENTIFIED BY 'KamalPW';
CREATE USER Nimal IDENTIFIED BY 'NimalPW';
CREATE USER Sunil IDENTIFIED BY 'SunilPW';
CREATE USER Anil IDENTIFIED BY 'AnilPW';
CREATE USER Amarabandu IDENTIFIED BY 'AmarabanduPW';
CREATE USER Sirisena IDENTIFIED BY 'SirisenaPW';
CREATE USER Tharaka IDENTIFIED BY 'TharakaPW';
CREATE USER Kumar IDENTIFIED BY 'KumarPW';
```

#	Time	Action	Message	Duration / Fetch
40	11:38:56	CREATE USER Bhanuka IDENTIFIED BY 'BhanukaPW'	0 row(s) affected	0.000 sec
41	11:38:56	CREATE USER Nawodya IDENTIFIED BY 'NawodyaPW'	0 row(s) affected	0.000 sec
42	11:38:56	CREATE USER Pasindu IDENTIFIED BY 'PasinduPW'	0 row(s) affected	0.016 sec

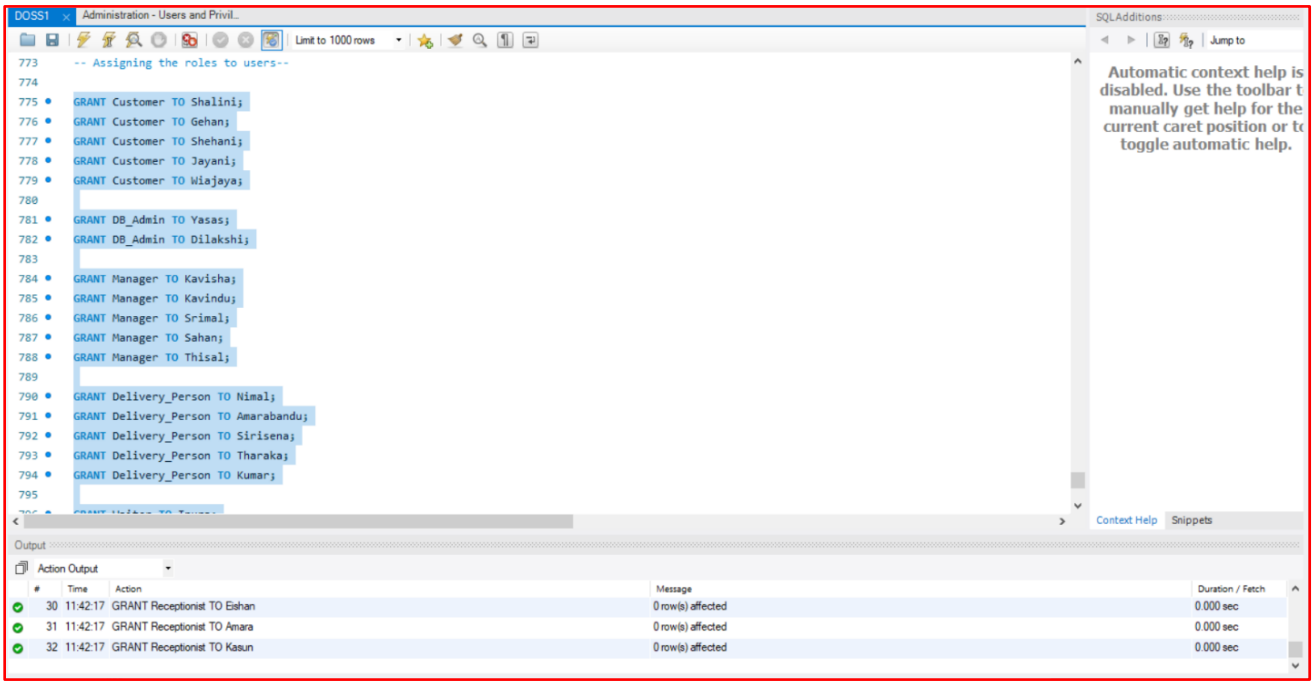


```
CREATE USER Amara IDENTIFIED BY 'AmaraPW';
CREATE USER Kasun IDENTIFIED BY 'KasunPW';
CREATE USER Kavisha IDENTIFIED BY 'KavishaPW';
CREATE USER Kavindu IDENTIFIED BY 'KavinduPW';
CREATE USER Srimal IDENTIFIED BY 'SrimalPW';
CREATE USER Sahan IDENTIFIED BY 'SahanPW';
CREATE USER Thisal IDENTIFIED BY 'ThisalPW';
CREATE USER Inura IDENTIFIED BY 'InuraPW';
CREATE USER Janith IDENTIFIED BY 'JanithPW';
CREATE USER Sandul IDENTIFIED BY 'SandulPW';
CREATE USER Sanuka IDENTIFIED BY 'SanukaPW';
CREATE USER Pasan IDENTIFIED BY 'PasanPW';
CREATE USER Tirone IDENTIFIED BY 'TironePW';
CREATE USER Avishka IDENTIFIED BY 'AvishkaPW';
CREATE USER Kusal IDENTIFIED BY 'KusalPW';
CREATE USER Nuwan IDENTIFIED BY 'NuwanPW';
CREATE USER Hashan IDENTIFIED BY 'HashanPW';
CREATE USER Shamal IDENTIFIED BY 'ShamalPW';
CREATE USER Kavinga IDENTIFIED BY 'KavingaPW';
CREATE USER Lahiru IDENTIFIED BY 'LahiruPW';
CREATE USER Bhanuka IDENTIFIED BY 'BhanukaPW';
CREATE USER Nawodya IDENTIFIED BY 'NawodyaPW';
CREATE USER Pasindu IDENTIFIED BY 'PasinduPW';
```

#	Time	Action	Message	Duration / Fetch
40	11:38:56	CREATE USER Bhanuka IDENTIFIED BY 'BhanukaPW'	0 row(s) affected	0.000 sec
41	11:38:56	CREATE USER Nawodya IDENTIFIED BY 'NawodyaPW'	0 row(s) affected	0.000 sec
42	11:38:56	CREATE USER Pasindu IDENTIFIED BY 'PasinduPW'	0 row(s) affected	0.016 sec

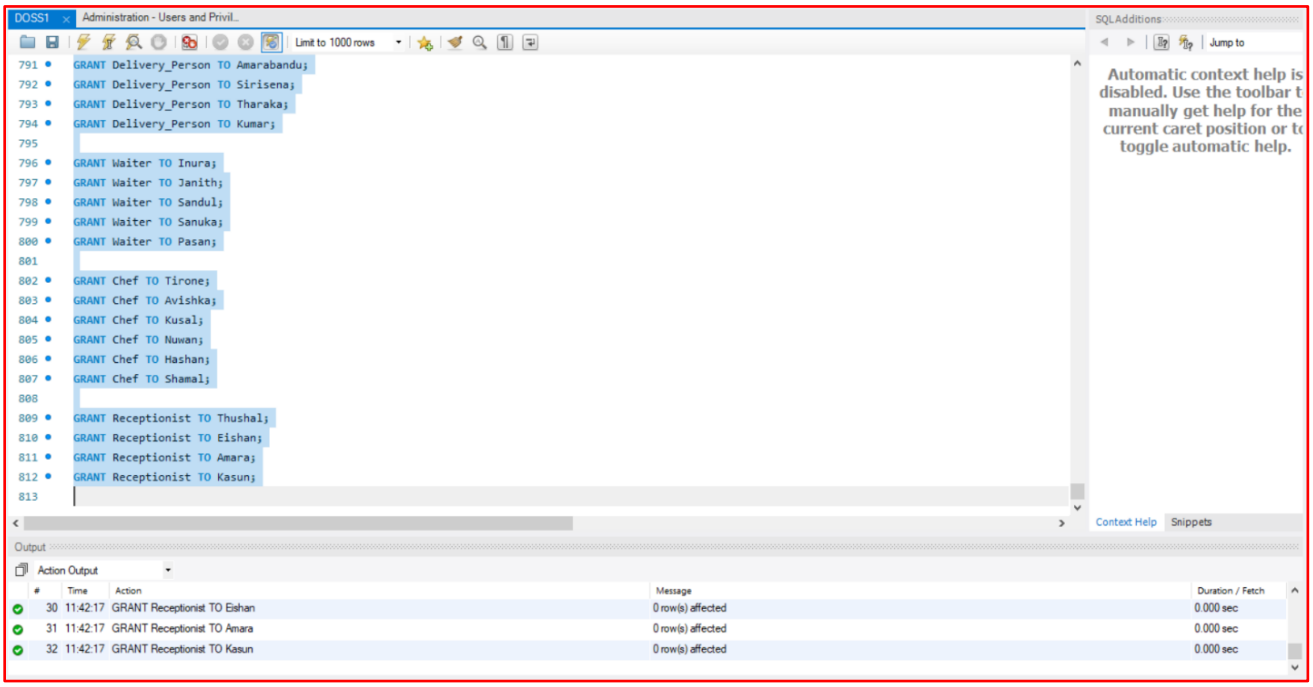
Step 4 :- Assigning the Roles to Users

The final step is to assign the above users with necessary roles.



```
-- Assigning the roles to users--

GRANT Customer TO Shalini;
GRANT Customer TO Gehan;
GRANT Customer TO Shehani;
GRANT Customer TO Jayani;
GRANT Customer TO Wiajaya;

GRANT DB_Admin TO Yasas;
GRANT DB_Admin TO Dilakshi;

GRANT Manager TO Kavisha;
GRANT Manager TO Kavindu;
GRANT Manager TO Srimal;
GRANT Manager TO Sahan;
GRANT Manager TO Thisal;

GRANT Delivery_Person TO Nimal;
GRANT Delivery_Person TO Amarabandu;
GRANT Delivery_Person TO Sirisena;
GRANT Delivery_Person TO Tharaka;
GRANT Delivery_Person TO Kumar;
```

#	Time	Action	Message	Duration / Fetch
30	11:42:17	GRANT Receptionist TO Eishan	0 row(s) affected	0.000 sec
31	11:42:17	GRANT Receptionist TO Amara	0 row(s) affected	0.000 sec
32	11:42:17	GRANT Receptionist TO Kasun	0 row(s) affected	0.000 sec



```
GRANT Delivery_Person TO Amarabandu;
GRANT Delivery_Person TO Sirisena;
GRANT Delivery_Person TO Tharaka;
GRANT Delivery_Person TO Kumar;

GRANT Waiter TO Inura;
GRANT Waiter TO Janith;
GRANT Waiter TO Sandul;
GRANT Waiter TO Sanuka;
GRANT Waiter TO Pasan;

GRANT Chef TO Tirone;
GRANT Chef TO Avishka;
GRANT Chef TO Kusal;
GRANT Chef TO Nuwan;
GRANT Chef TO Hashan;
GRANT Chef TO Shamal;

GRANT Receptionist TO Thushal;
GRANT Receptionist TO Eishan;
GRANT Receptionist TO Amara;
GRANT Receptionist TO Kasun;
```

#	Time	Action	Message	Duration / Fetch
30	11:42:17	GRANT Receptionist TO Eishan	0 row(s) affected	0.000 sec
31	11:42:17	GRANT Receptionist TO Amara	0 row(s) affected	0.000 sec
32	11:42:17	GRANT Receptionist TO Kasun	0 row(s) affected	0.000 sec

7) Manage to use strong password and follow secure methods to preserve the stored passwords.

In here, as the very 1st step, I have created a table to store the user credentials. It consists with the username & the password of the users.

```
-- Storing user account passwords securely --

CREATE TABLE User_Credentials(

            Username VARCHAR(255) NOT NULL,
            Password VARCHAR(255) NOT NULL,

            CONSTRAINT pk_credentials PRIMARY KEY (Username)

                        );
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	13:44:00	CREATE TABLE User_Credentials(Username VARCHAR(255) NOT NULL, Password VARCHAR(255) NOT NULL, ...	0 row(s) affected	0.031 sec

For all the users, I have assigned them with strong user passwords. Furthermore I have used several cryptographic mechanisms to secure the stored passwords.

i. MD5 Hashing

```
-- Method 1 :- MD5 Hashing --

INSERT INTO User_Credentials(Username, Password) VALUES ('Amal_J', md5('P455w0R$d1'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Kamal_G', md5('P455w0R$d2'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Nimal_L', md5('P455w0R$d3'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Sunil_N', md5('P455w0R$d4'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Anil_S', md5('P455w0R$d5'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Amarabandu_R', md5('P455w0R$d6'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Sirisena_F', md5('P455w0R$d7'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Tharaka_K', md5('P455w0R$d8'));
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
6	13:45:04	INSERT INTO User_Credentials(Username, Password) VALUES ('Amarabandu_R', md5('P455w0R$d6'))	1 row(s) affected	0.000 sec
7	13:45:04	INSERT INTO User_Credentials(Username, Password) VALUES ('Sirisena_F', md5('P455w0R$d7'))	1 row(s) affected	0.000 sec
8	13:45:04	INSERT INTO User_Credentials(Username, Password) VALUES ('Tharaka_K', md5('P455w0R$d8'))	1 row(s) affected	0.000 sec

ii. SHA1 Hashing

```
-- Method 2 :- SHA1 Hashing --

INSERT INTO User_Credentials(Username, Password) VALUES ('Kumar_D', sha1('P455w0R$d9'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Amarasiri_P', sha1('P455w0R$d1o'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Thushal_K', sha1('P455w0R$d11'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Eishan_D', sha1('P455w0R$d12'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Amara_N', sha1('P455w0R$d1E'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Kasun_L', sha1('P455w0R$d14'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Kavisha_S', sha1('P455w0R$d1S'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Kavindu_D', sha1('P455w0R$d16'));
```

Context Help | Snippets

Output

Action Output

#	Time	Action	Message	Duration / Fetch
5	13:59:55	INSERT INTO User_Credentials(Username, Password) VALUES ('Amara_N', sha1('P455w0R$d1E'))	1 row(s) affected	0.000 sec
6	13:59:55	INSERT INTO User_Credentials(Username, Password) VALUES ('Kasun_L', sha1('P455w0R$d14'))	1 row(s) affected	0.000 sec
7	13:59:55	INSERT INTO User_Credentials(Username, Password) VALUES ('Kavisha_S', sha1('P455w0R$d1S'))	1 row(s) affected	0.000 sec
8	13:59:55	INSERT INTO User_Credentials(Username, Password) VALUES ('Kavindu_D', sha1('P455w0R$d16'))	1 row(s) affected	0.000 sec

iii. AES Encryption

```
-- Method 3 :- AES Encryption --

INSERT INTO User_Credentials(Username, Password) VALUES ('Srimal_K', aes_encrypt('P455w0R$d17', 'secret'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Sahan_A', aes_encrypt('P455w0R$d18', 'secret'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Thisal_P', aes_encrypt('P455w0R$d19', 'secret'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Inura_K', aes_encrypt('P455w0R$d20', 'secret'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Janith_M', aes_encrypt('P455w0R$d21', 'secret'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Sandul_L', aes_encrypt('P455w0R$d22', 'secret'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Sanuka_L', aes_encrypt('P455w0R$d23', 'secret'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Pasan_M', aes_encrypt('P455w0R$d24', 'secret'));
```

iv. DES Encryption

```
-- Method 4 :- DES Encryption --

INSERT INTO User_Credentials(Username, Password) VALUES ('Tirone_L', des_encrypt('P455w0R$d25', 'secret'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Avishka_S', des_encrypt('P455w0R$d26', 'secret'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Kusal_M', des_encrypt('P455w0R$d27', 'secret'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Nuwan_A', des_encrypt('P455w0R$d28', 'secret'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Hashan_T', des_encrypt('P455w0R$d29', 'secret'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Shamal_K', des_encrypt('P455w0R$d30', 'secret'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Kavinga_Y', des_encrypt('P455w0R$d31', 'secret'));
INSERT INTO User_Credentials(Username, Password) VALUES ('Pasindu_N', des_encrypt('P455w0R$d32', 'secret'));
```

Since now we have used multiple cryptographic mechanisms to secure the passwords, let's see the output of it.

```
867 •   SELECT * FROM User_Credentials;
868
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Username	Password
Amal_J	81d8bbb0576cdbf7572b642605ece524
Amara_N	f97942f728be0b0aa349a2e6f8091b04d0fc4262
Amarabandu_R	34a42cdb047f79d4ed04e5ea871da07c
Amarasiri_P	a0c8bf2053d4cb94bb05d82b182798f43583aa31
Anil_S	f50767f0a8a319c0fae40259e370633c
Eishan_D	761ff786c87360928ff42b322087f537d828129d
Kamal_G	e13e94f42f23c305d7a2be73c70e0912
Kasun_L	5f415c7bee0f74c2f590886fd55b77af40530e60
Kavindu_D	5e16721367c18687027fe1ccf4cd35dc5fb44891
Kavisha_S	bf535fd538062253171dae00c5e7627000170db7
Kumar_D	323ee034cec26936cd3c7f37198243e82d32e890
Nimal_L	7f4f922f1a4518acca6157be932ff35e
Sirisena_F	9f0ab8b1f2c8e8a1466afc258615943c
Sunil_N	3c98a29c84ee09267e55cdde611addb9
Tharaka_K	9e4e4901416259ec8b79b6322c090941
Thushal_K	8571d750d1ee30948d4db31ca0400995dffb3f0b

Result Grid | Form Editor | Field Types | Query Stats

_Credentials 1 × Apply Revert

Output

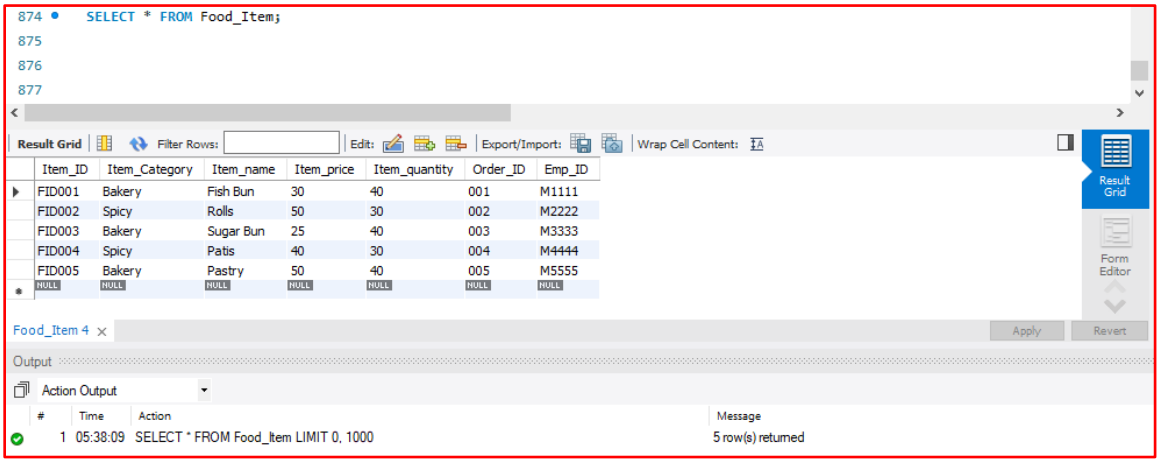
Action Output

#	Time	Action	Message
1	09:42:38	SELECT * FROM User_Credentials LIMIT 0, 1000	16 row(s) returned

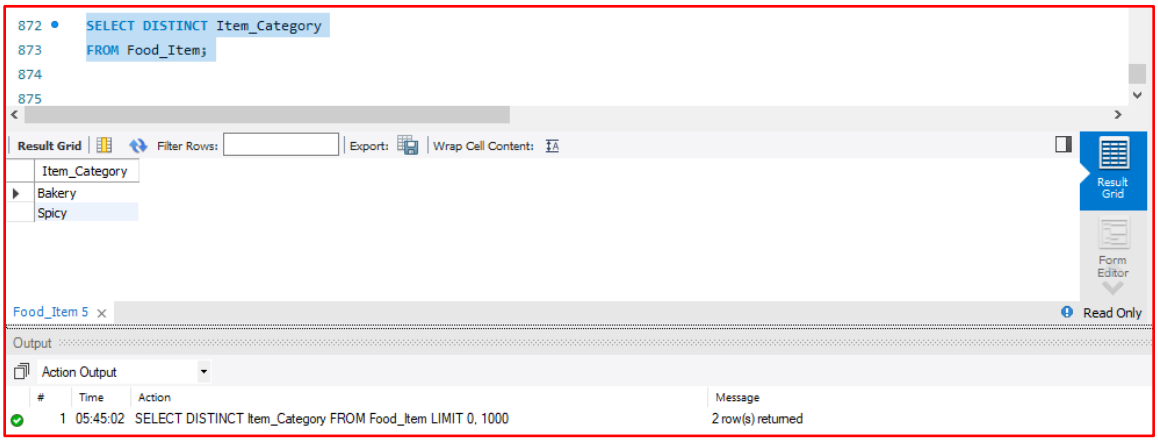
8) Prevent from redundancy of the stored records of the database.

Data redundancy is the duplication of data in a table or database. This may result in consuming more memory & storage area than usual. Usually, solving the redundancy of data is considered as the very 1st step towards analyzing large amounts of data.

So, the only solution to prevent from redundancy is the **Normalization.** Normalization is basically a set of guidelines that is used to reduce the redundant data while designing the database. So, you can use the keyword **"DISTINCT"** to remove the duplicate rows. In order to demonstrate this situation, I'm going to use the **"Food_Item"** table in my database.

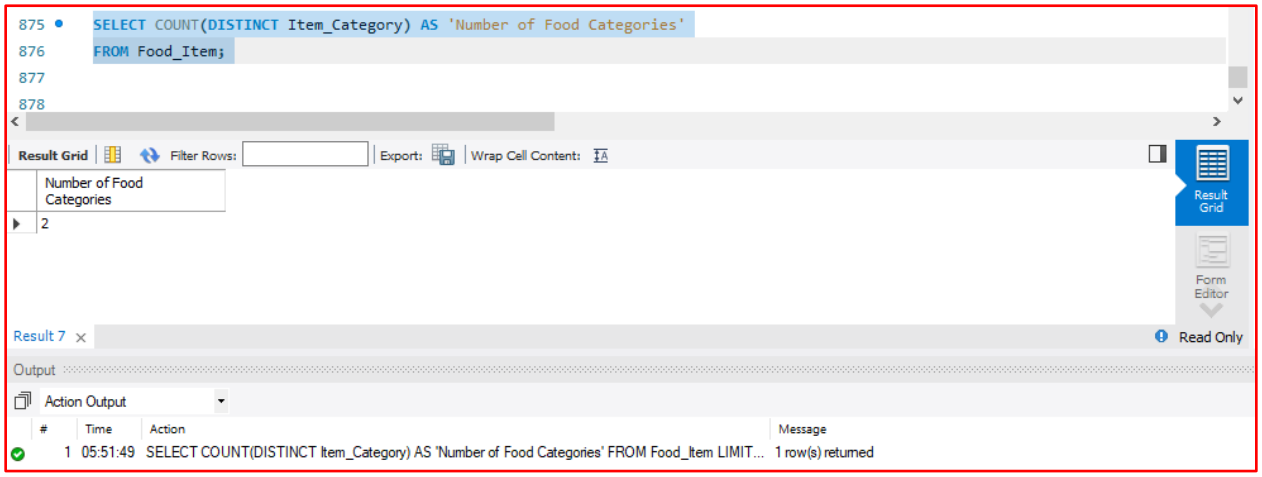


So, as you can see in the above image, under the **"Item_Category"** column, there are several redundant data that are available. In order to prevent this from happening, you need to use the keyword **"DISTINCT"** with the **SELECT** statement.



Furthermore, you can use the DISTINCT syntax with aggregate functions as well.

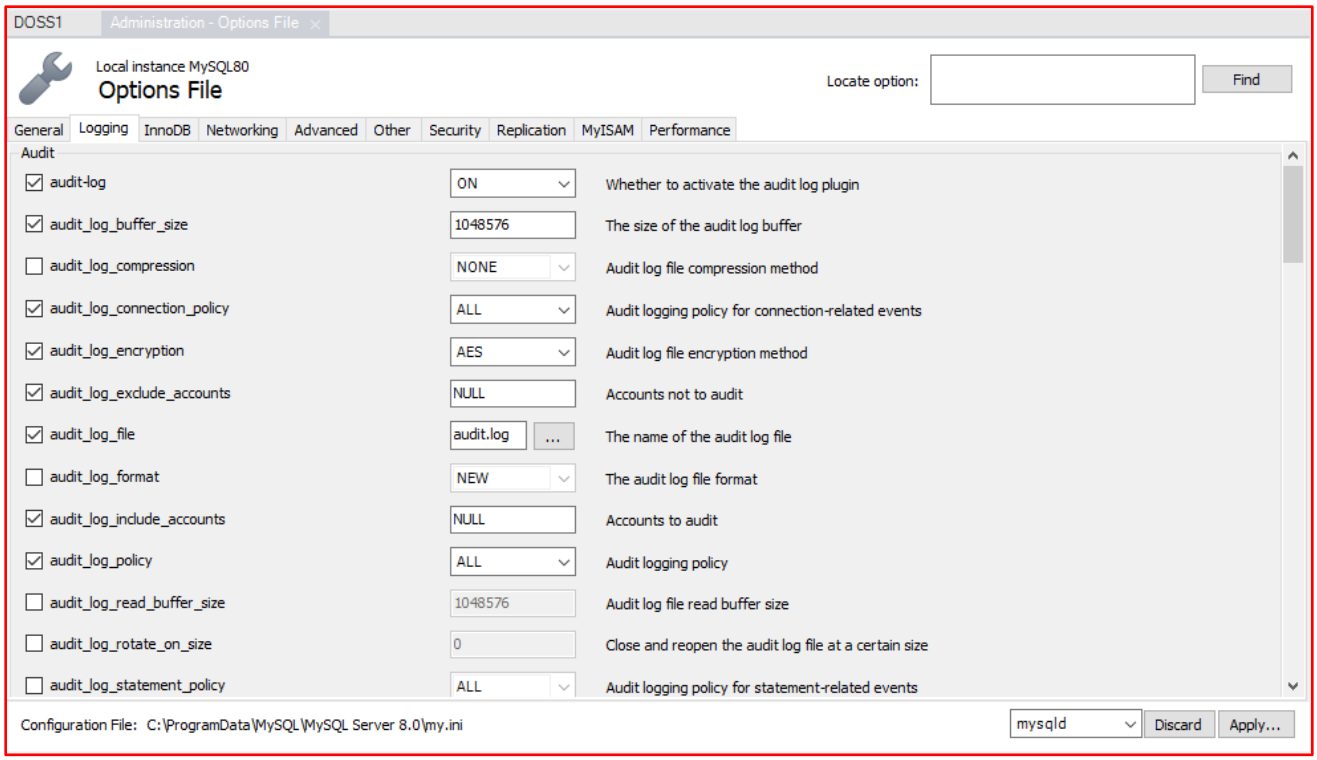
Ex : - **COUNT, AVG, MIN, MAX, SUM**



9) Turn on Auditing where technically possible for the database objects with protected data.

In here, you are able to turn on the auditing options by referring to **Navigator → Instance → Options File** tab.

After that you need to go to the **Logging** Tab in order to turn on the auditing features.



As you can see in the above image, there are multiple auditing options that are available for you to enable. So, you can choose to turn on any of the above audit options as per your consent.

10) Discuss how manage the implemented database backup and recovery.

When it comes to the MySQL, basically there are there 2 types of backups.

i. Logical Backups

- Usually, the logical backups save information represented as logical database structure **(CREATE Statements)** and content **(INSERT Statements)**. In most of the cases, this type of backups are suitable for smaller amounts of data where you might edit the data values or table structure or recreate the data on a different machine architecture.

ii. Physical Backups

- In here, both database and transaction logs should be backed up on order to restore/recover the database up to the point of failure. This type of backups consists with raw copies of the directories and files that store database contents. In most of the cases, the File or filegroup backup strategy can be used if the databases to be backed up are very large databases **(VLDBs)** that are partitioned among multiple files. Physical backups are suitable specially for large, important databases that need to be recovered quickly in case of an emergency.

Rather than those above-mentioned 2 types of database backups, there are other types of database backups as well such as **Online backups, Offline Backups, Local Backups, Remote Backups Snapshot Backups, Incremental Backups** etc.

When designing a proper backup planning mechanism, you need to concentrate on the following facts as well.

i. Establish a strategy for handling VLDB backups
ii. Establish an appropriate backup schedule and window
iii. Decide where to store backups
iv. Develop a backup retention policy

After designing a strong backup plan, you need to follow the below points in order to manage it properly.

i. Automate the backups → *Use maintenance plans*

ii. Monitor the backups → *Receive fail alerts through emails or sms*

iii. Review backup logs and catalogs

iv. Validate the backups → *Verify whether they are performing the backups correctly*

v. Set up dependencies → *You need to backup the database for multiple storage device in a sequential manner (Disks & Tape backups)*

vi. Test the backup restoration process → *Whether it is effective or not*

vii. Perform annual restore testing for audit purposes

When it comes to the database recovery, a proper Disaster Recovery Plan needs to be maintained. It's the responsibility of the Database Administrator to ensure that databases are included as a key element in the company's overall DRP.

Finally, proper OS & Database backup and recovery tools needs to be kept up to date with the latest version. Those backup & recovery tools have to be fully ready in case of an actual restore event.