



Sri Lanka Institute of Information Technology

Assignment

Implementing Security features in OS & DB

IE3062 - Data and Operating Systems Security

Individual Assignment

Submitted by:

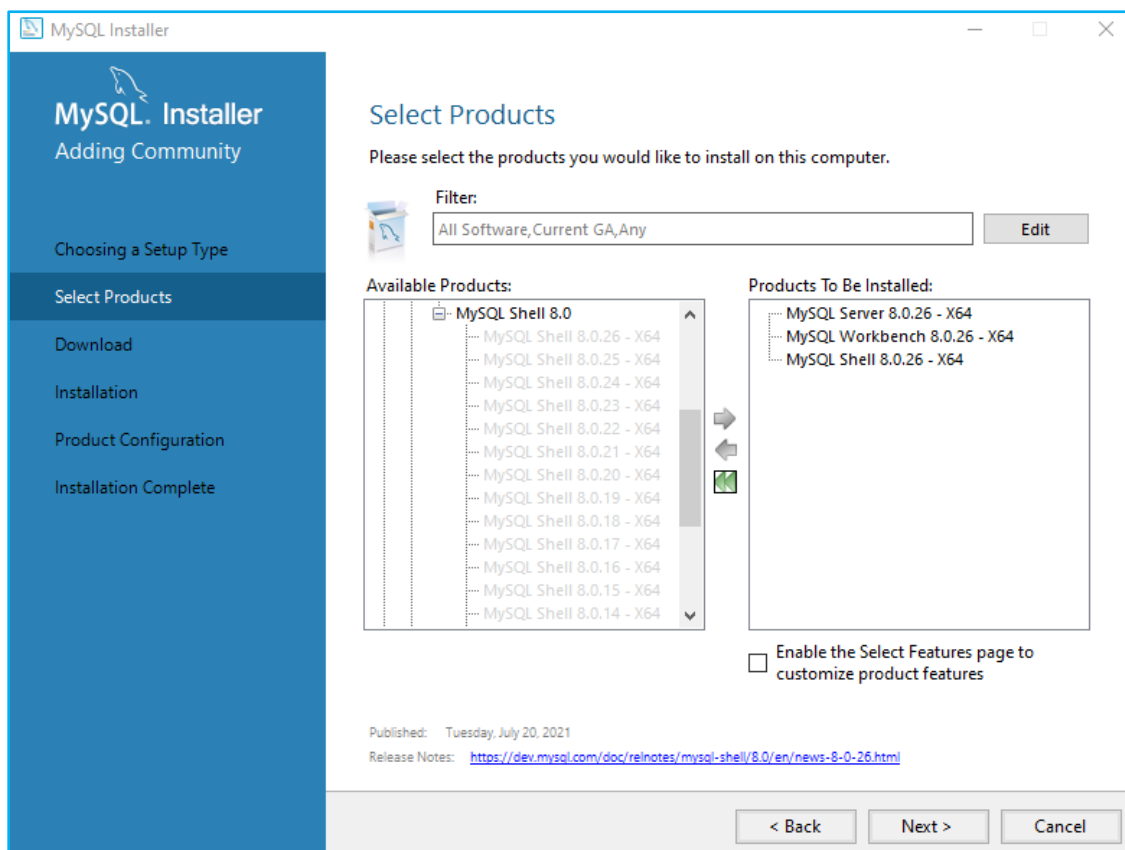
Student Registration Number	Student Name
IT19029146	Eranda H.P.D

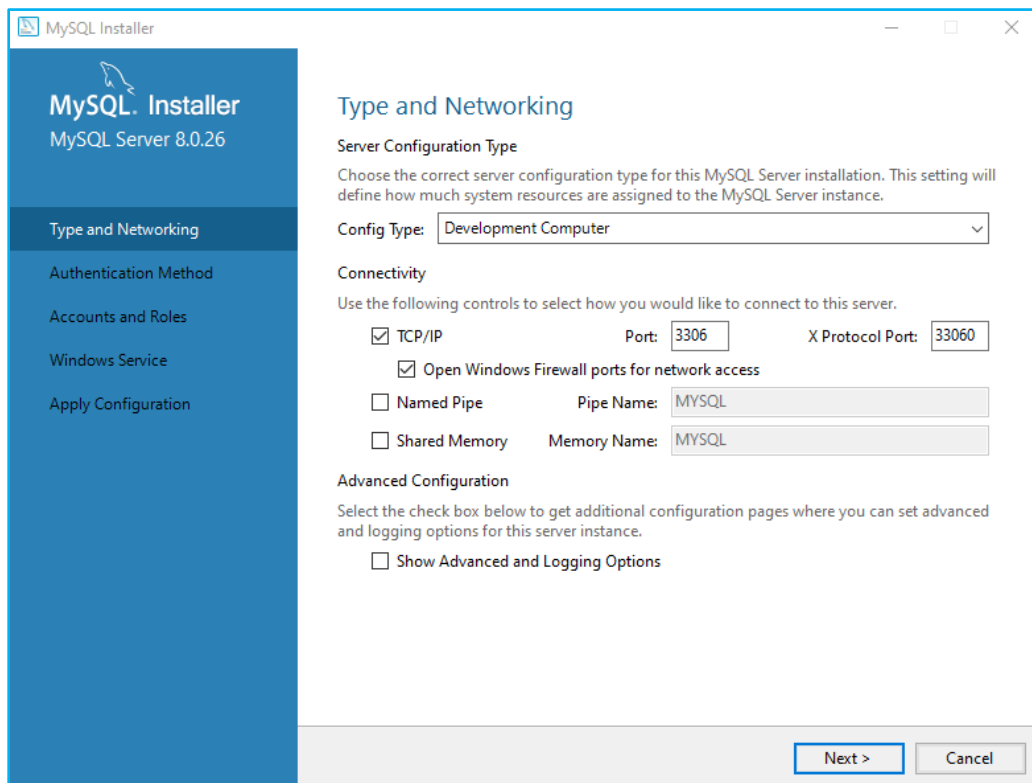
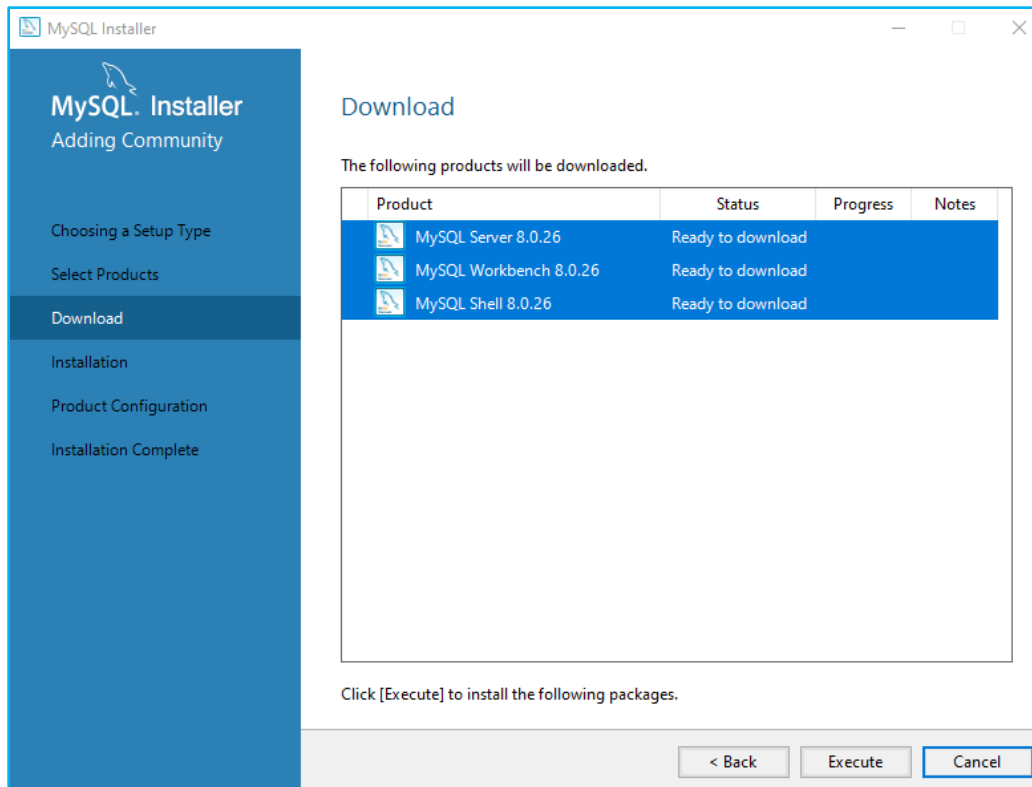
16/10/2021
Date of submission

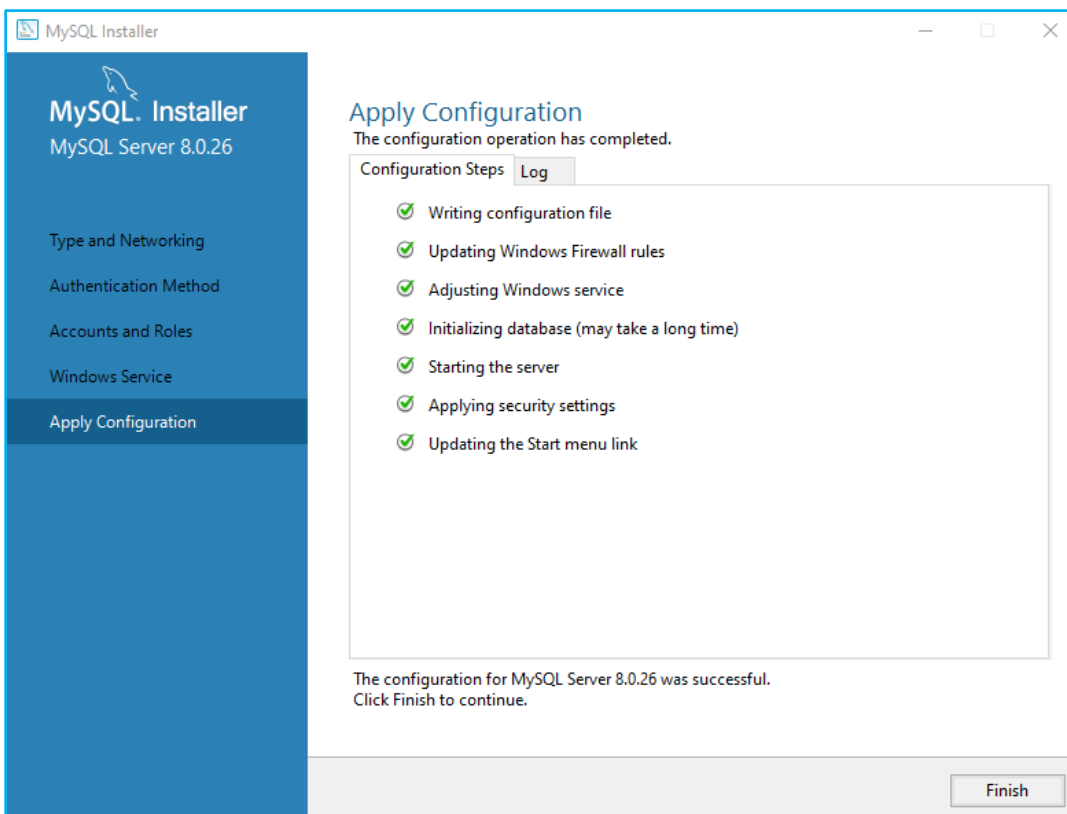
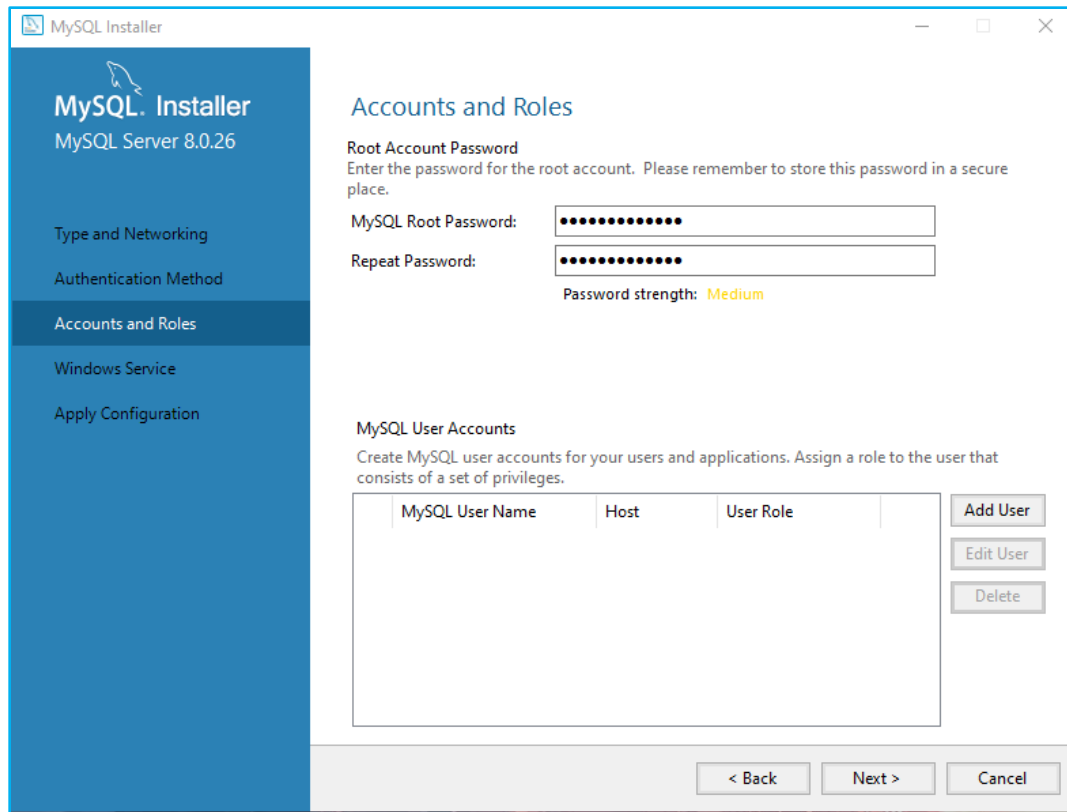
Security Check List :-

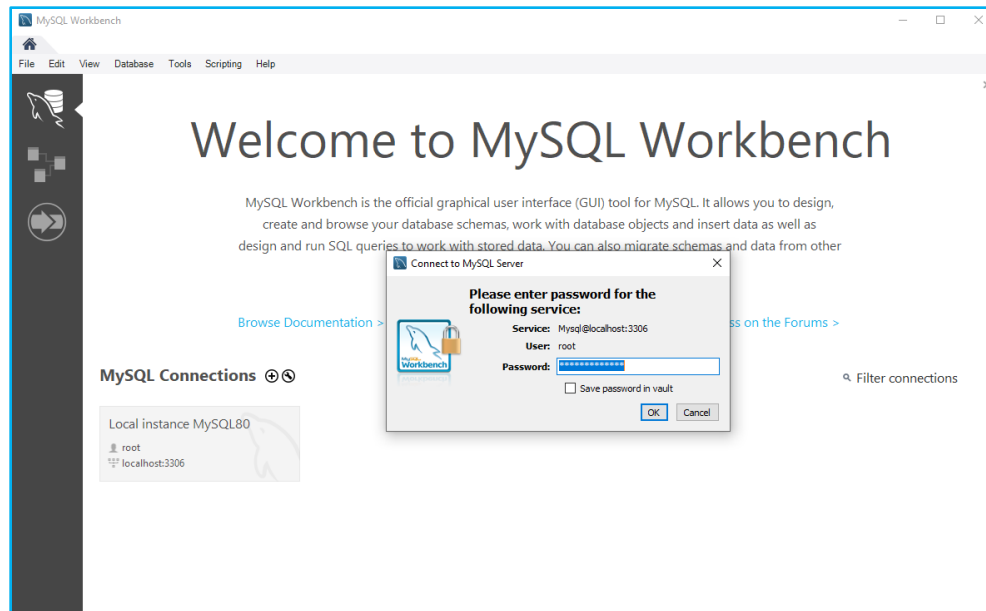
In this assignment, in order to enable the security features which are provided in the checklist, we were given the opportunity to choose a suitable operating system as per our wish. So, I have chosen **Windows Operating System** as my host operating system in order to perform this assignment. As the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

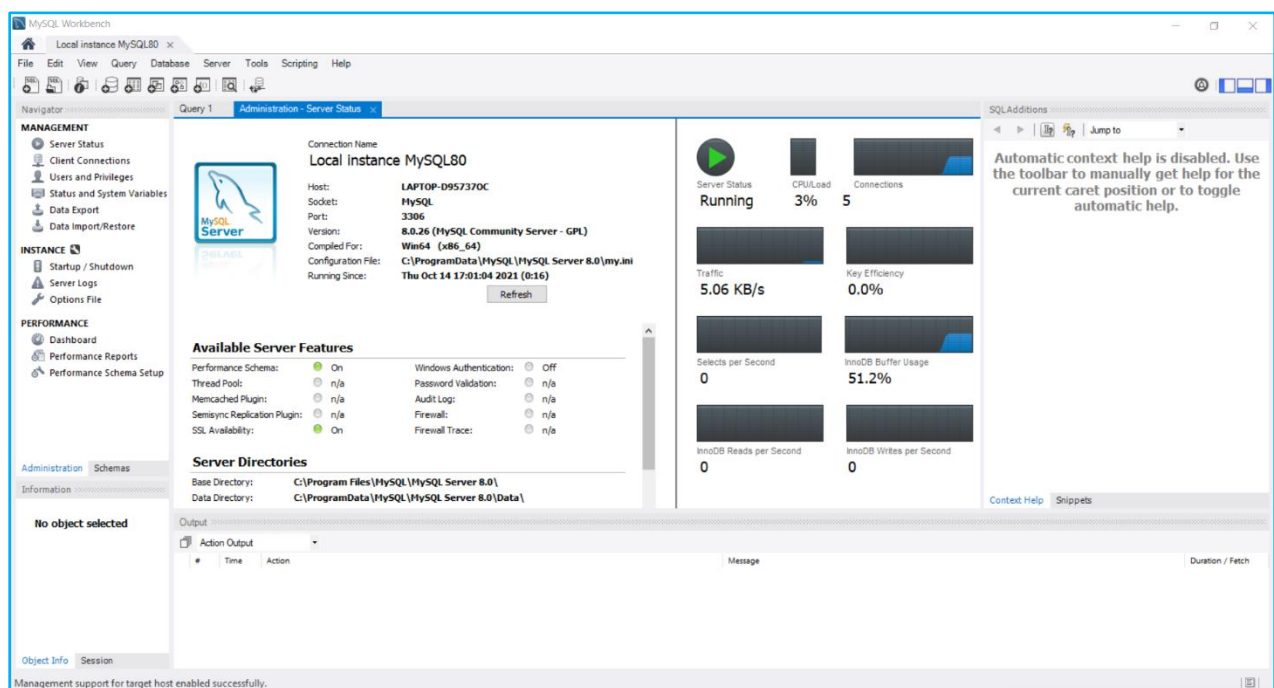








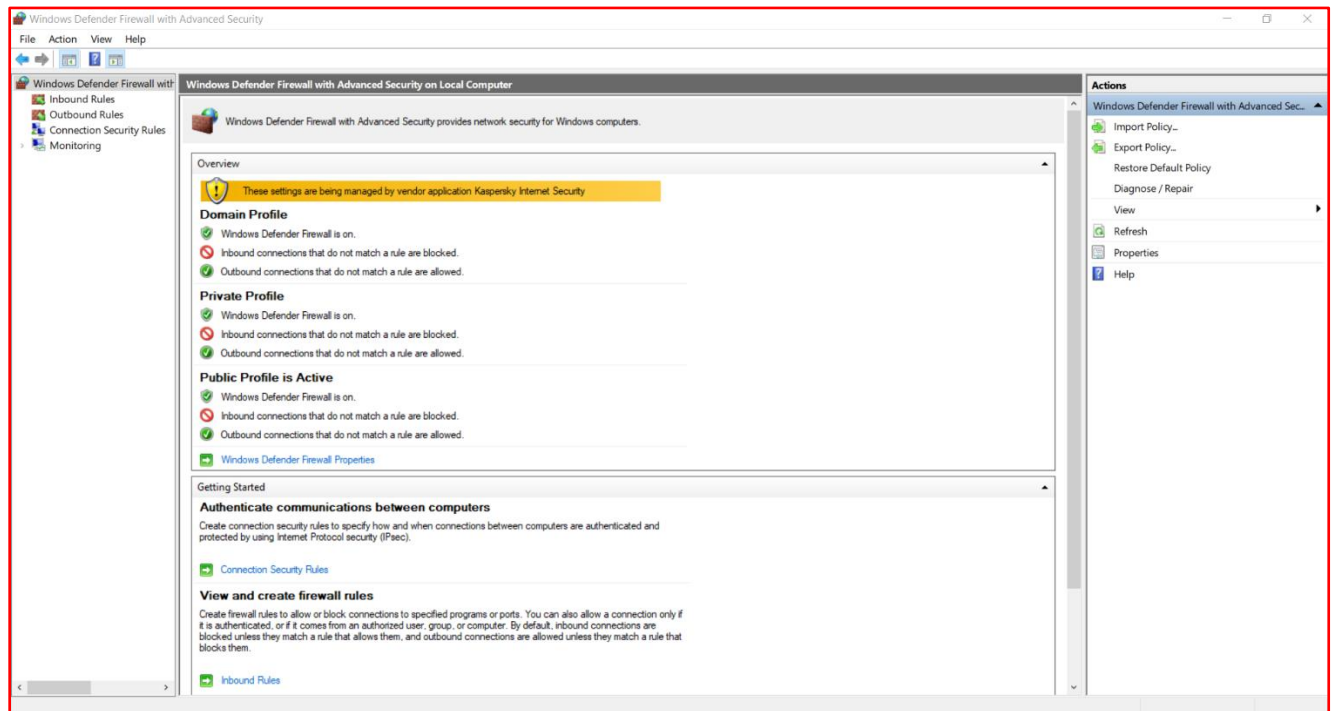
So, as you can see in the above image, I have successfully created a local database connection from the MySQL Workbench.



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

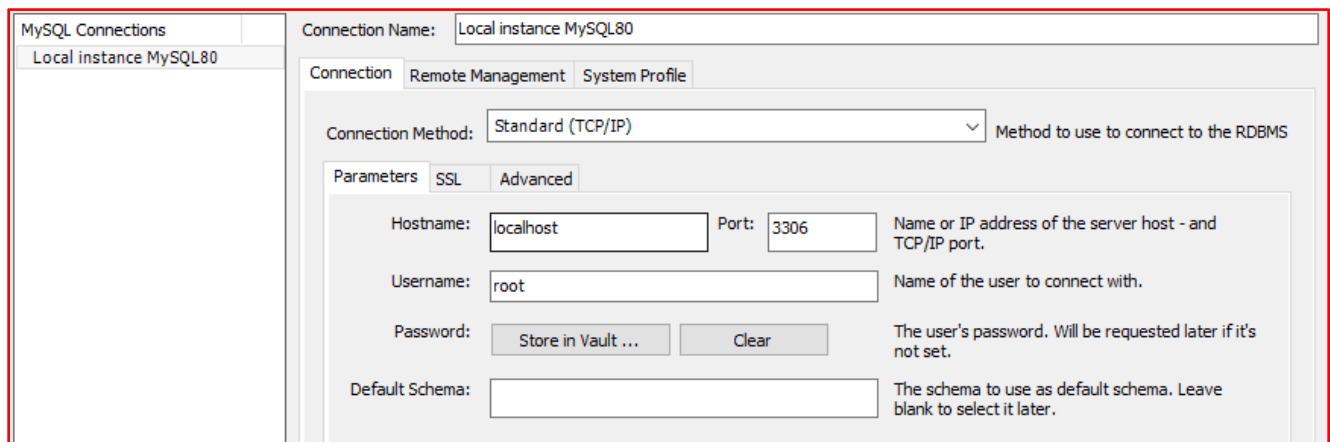
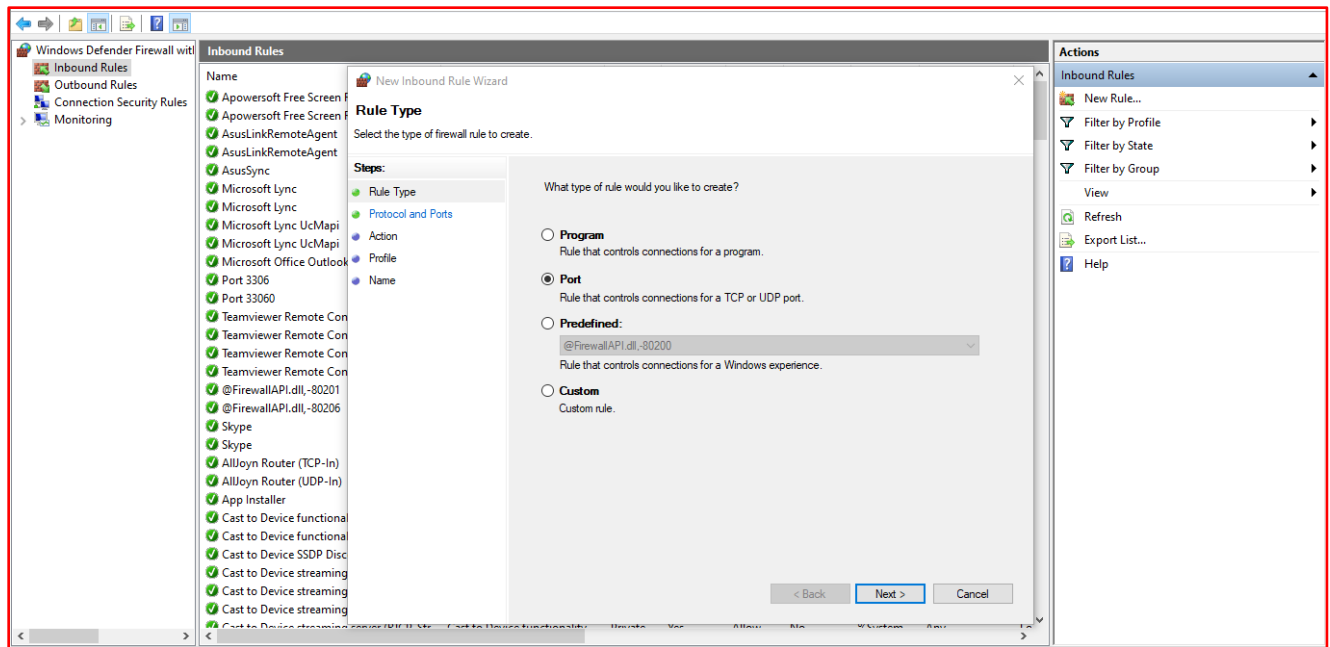
Since this is not the MySQL Enterprise version, it does not come with an inbuilt firewall for the database server. However, in order to configure the firewall rules, I have used the firewall settings on **Windows Defender Firewall** in my local computer.



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.

New Inbound Rule Wizard

Protocol and Ports

Specify the protocols and ports to which this rule applies.

Steps:

- Rule Type
- Protocol and Ports
- Action
- Profile
- Name

Does this rule apply to TCP or UDP?

☒ TCP
☐ UDP

Does this rule apply to all local ports or specific local ports?

☐ All local ports
☒ Specific local ports:
Example: 80, 443, 5000-5010

< Back Next > Cancel

New Inbound Rule Wizard

Action

Specify the action to be taken when a connection matches the conditions specified in the rule.

Steps:

- Rule Type
- Protocol and Ports
- Action
- Profile
- Name

What action should be taken when a connection matches the specified conditions?

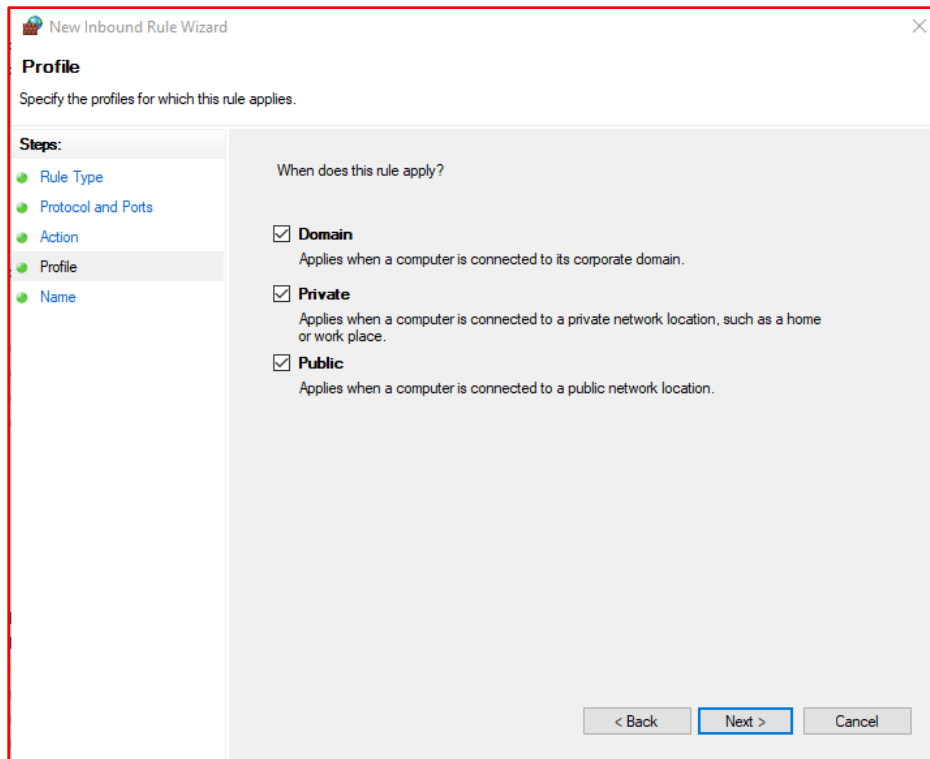
☒ **Allow the connection**
This includes connections that are protected with IPsec as well as those are not.

☐ **Allow the connection if it is secure**
This includes only connections that have been authenticated by using IPsec. Connections will be secured using the settings in IPsec properties and rules in the Connection Security Rule node.

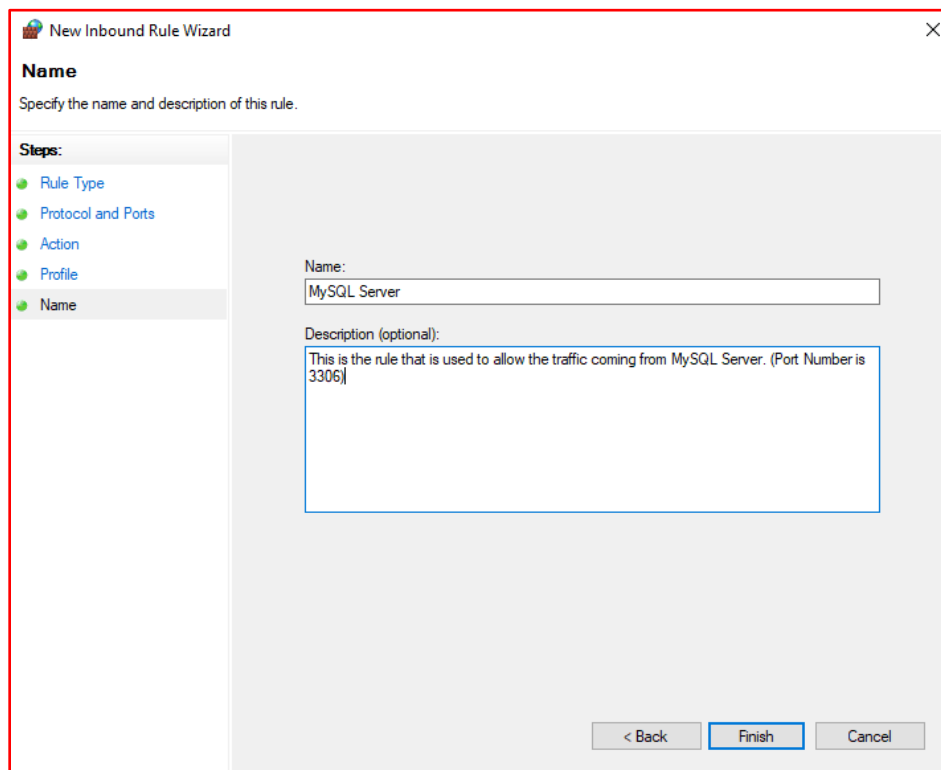
☐ **Block the connection**

< Back Next > Cancel

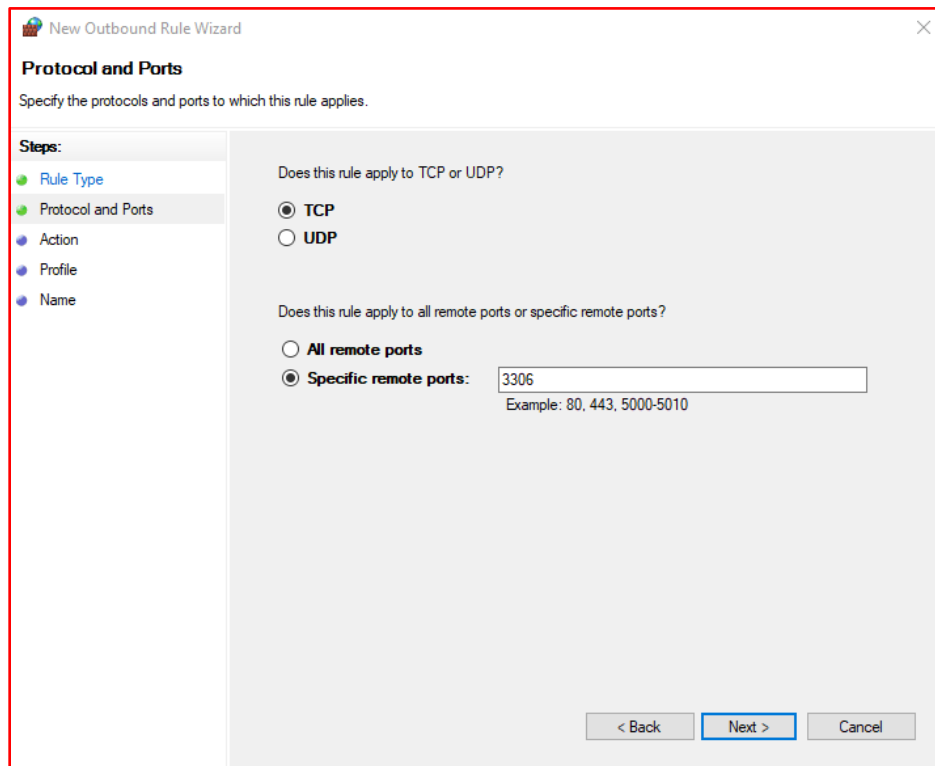
After that, you need to define **“For which network environments this rule needs to be applied to.”**



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

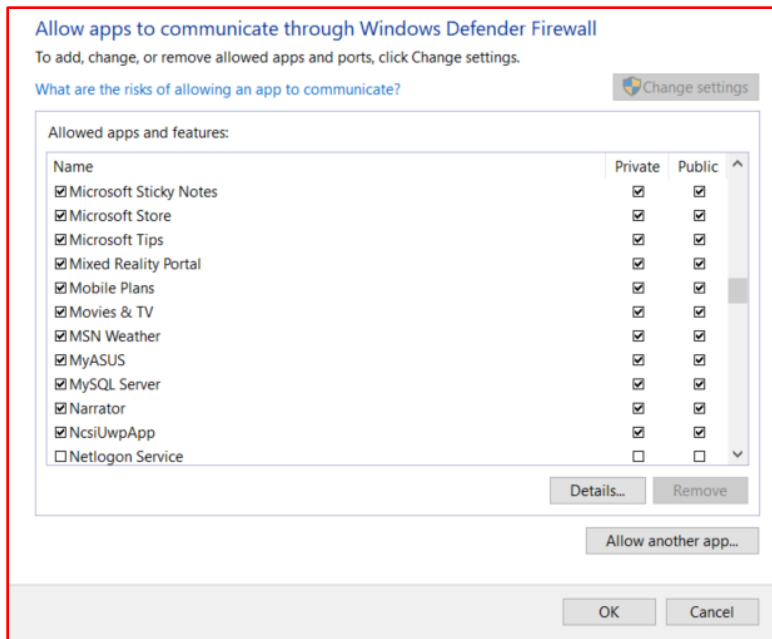


Now, let's move towards configuring the outbound firewall rules. In here also, you need follow the same procedure just like in configuring the inbound rules.



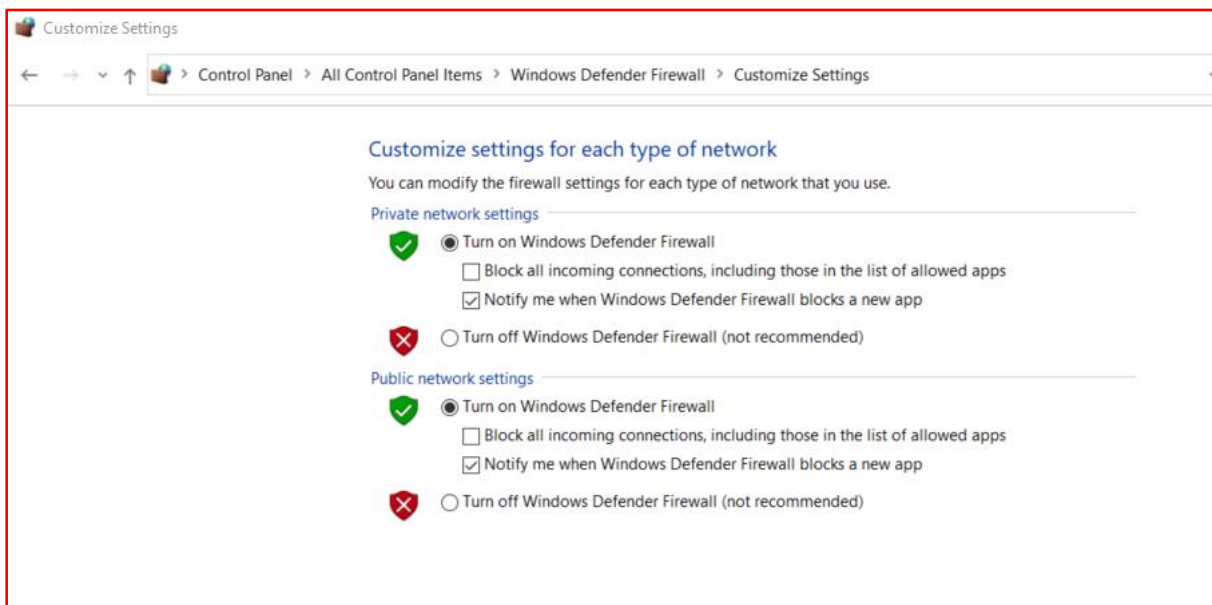
Furthermore, rather than only defining the firewall rules, you also need to manually allow the MySQL Database Server to transfer the traffic. You can perform it by selecting on the **“Allow an app or feature through Windows Defender Firewall”** option from the Windows Defender firewall settings.





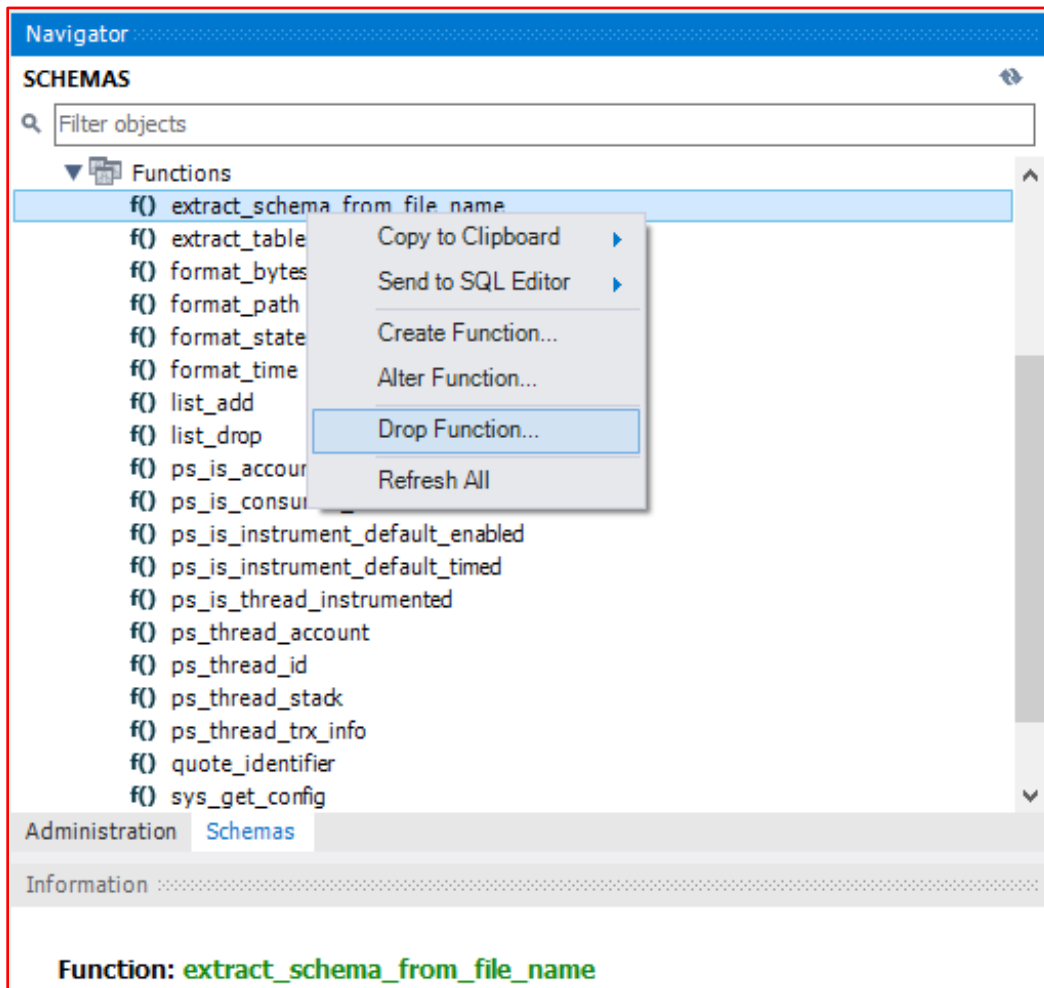
As it is demonstrated in the above image, you can perform it by selecting **Change Settings → Allow Another app → Browse the location of the app**.

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.

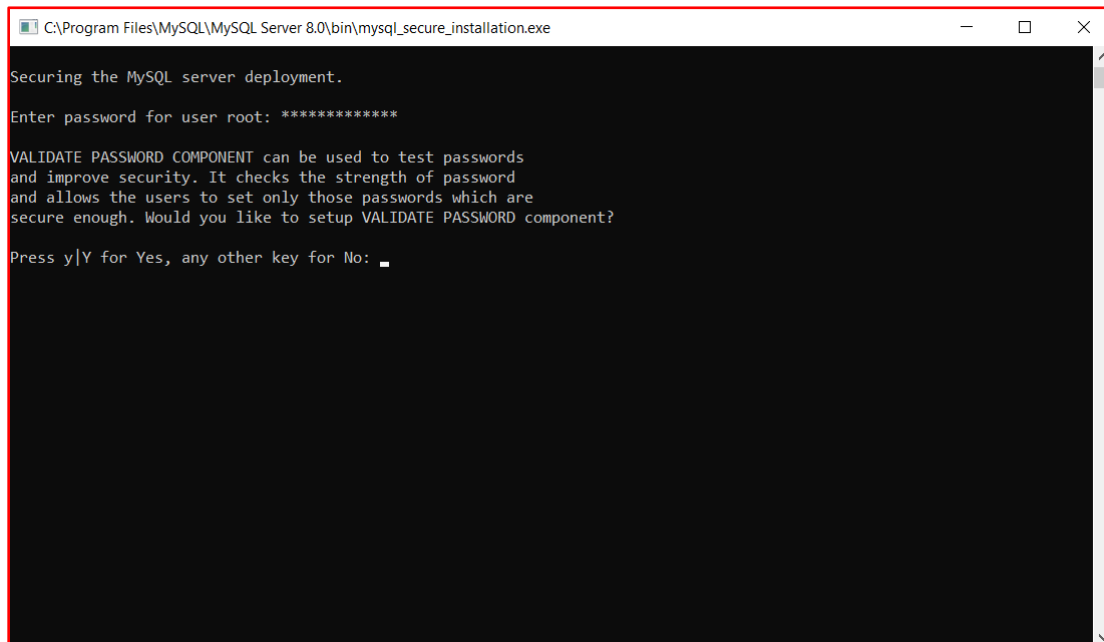


So, from this drop-down list, I have removed the following unnecessary functions.

- 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.



```
C:\Program Files\MySQL\MySQL Server 8.0\bin\mysql_secure_installation.exe

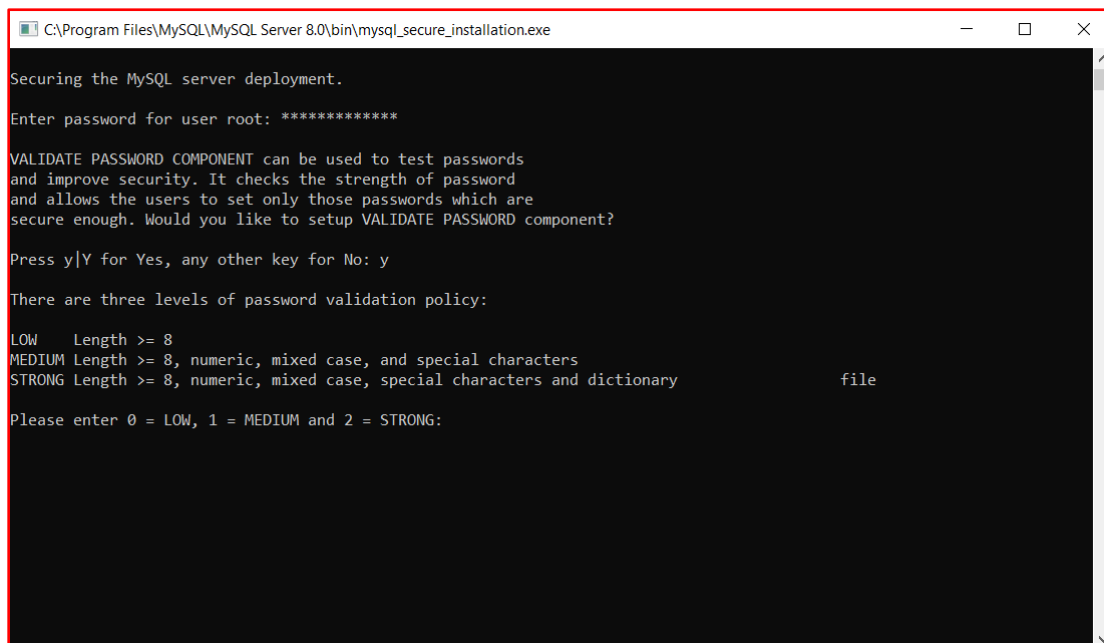
Securing the MySQL server deployment.

Enter password for user root: *****

VALIDATE PASSWORD COMPONENT can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD component?

Press y|Y for Yes, any other key for No: _
```

In here, you need to input ‘y’ in here.



```
C:\Program Files\MySQL\MySQL Server 8.0\bin\mysql_secure_installation.exe

Securing the MySQL server deployment.

Enter password for user root: *****

VALIDATE PASSWORD COMPONENT can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD component?

Press y|Y for Yes, any other key for No: y

There are three levels of password validation policy:

LOW    Length >= 8
MEDIUM Length >= 8, numeric, mixed case, and special characters
STRONG Length >= 8, numeric, mixed case, special characters and dictionary file

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG:
```

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.
```

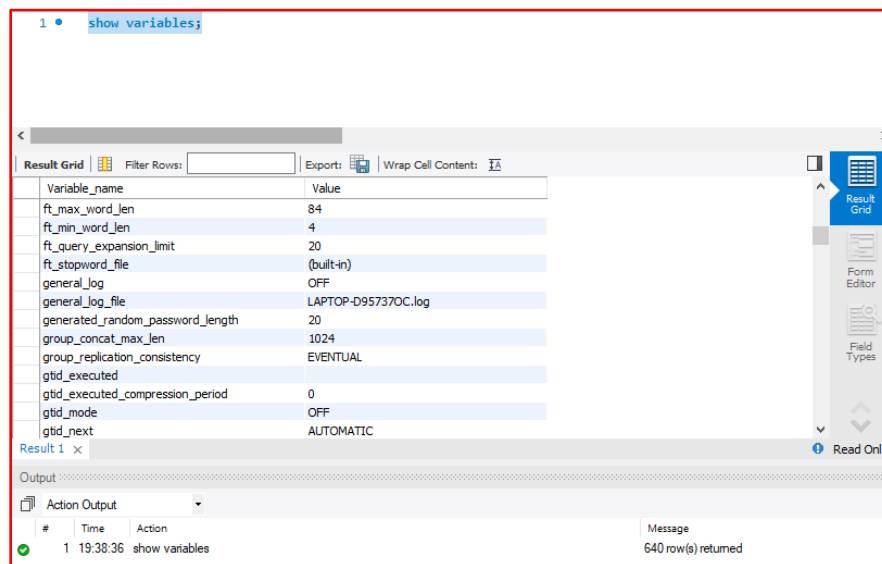
```
- Removing privileges on test database...
Success.
```

```
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.



```
1 • show variables;
```

Variable_name	Value
ft_max_word_len	84
ft_min_word_len	4
ft_query_expansion_limit	20
ft_stopword_file	(built-in)
general_log	OFF
general_log_file	LAPTOP-D957370C.log
generated_random_password_length	20
group_concat_max_len	1024
group_replication_consistency	EVENTUAL
gtid_executed	
gtid_executed_compression_period	0
gtid_mode	OFF
gtid_next	AUTOMATIC

Result 1 x

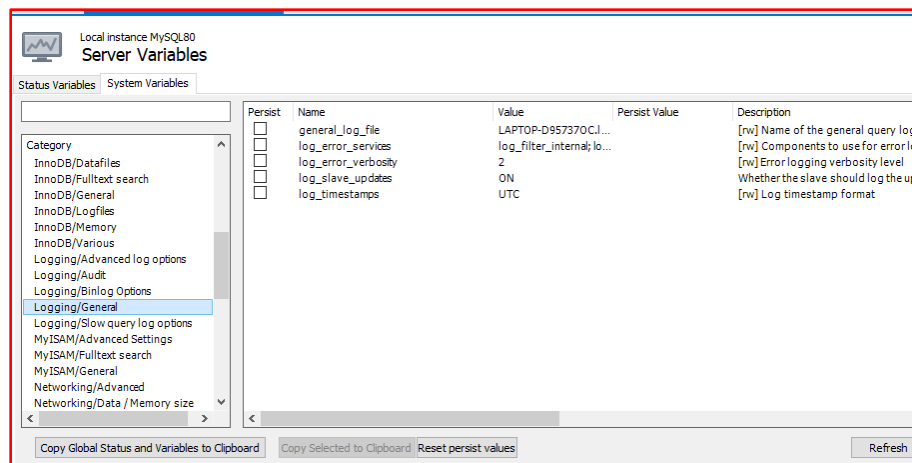
Output

Action Output

#	Time	Action	Message
1	19:38:36	show variables	640 row(s) returned

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';
```



Local instance MySQL80

Server Variables

Status Variables System Variables

Persist	Name	Value	Persist Value	Description
<input type="checkbox"/>	general_log_file	LAPTOP-D957370C.log		[rw] Name of the general query log file
<input type="checkbox"/>	log_error_services	log_filter_internal;log...		[rw] Components to use for error log
<input type="checkbox"/>	log_error_verbosity	2		[rw] Error logging verbosity level
<input type="checkbox"/>	log_slave_updates	ON		Whether the slave should log the updates
<input type="checkbox"/>	log_timestamps	UTC		[rw] Log timestamp format

Category

- InnoDB/Datafiles
- InnoDB/Fulltext search
- InnoDB/General
- InnoDB/Logfiles
- InnoDB/Memory
- InnoDB/Various
- Logging/Advanced log options
- Logging/Audit
- Logging/Binlog Options
- Logging/General
- Logging/Slow query log options
- MyISAM/Advanced Settings
- MyISAM/Fulltext search
- MyISAM/General
- Networking/Advanced
- Networking/Data / Memory size

Copy Global Status and Variables to Clipboard Copy Selected to Clipboard Reset persist values Refresh

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.

Client Connections

Local instance MySQL80

Threads Connected: 5 Threads Running: 2 Threads Created: 5 Threads Cached: 0 Rejected (over limit): 0

Total Connections: 33 Connection Limit: 151 Aborted Clients: 0 Aborted Connections: 4 Errors: 0

Id	User	Host	DB	Command	Time	State	Threa...	Type	Name	Par...
5	event_sched...	localhost	None	Daemon	10027	Waiting on e...	47	FOREGROUND	thread/sql/e...	
7	None	None	None	Daemon	10027	Suspending	50	FOREGROUND	thread/sql/c...	
14	root	localhost	None	Sleep	602	None	58	FOREGROUND	thread/sql/o...	
30	root	localhost	None	Sleep	54	None	74	FOREGROUND	thread/sql/o...	
31	root	localhost	None	Sleep	45	None	75	FOREGROUND	thread/sql/o...	
32	root	localhost	None	Query	0	executing	76	FOREGROUND	thread/sql/o...	
33	root	localhost	None	Sleep	0	None	77	FOREGROUND	thread/sql/o...	

Refresh Rate: Don't Refresh Kill Query(s) Kill Connection(s) Refresh

☐ Hide sleeping connections ☒ Hide background threads ☒ Don't load full thread info Show Details

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

Manage Server Connections

MySQL Connections

Local instance MySQL80

Connection Name: Local instance MySQL80

Connection Remote Management System Profile

☒ Do not use remote management

☐ Native Windows remote management (only available on Windows)

☐ SSH login based management

Hostname: localhost Port:

Username: DILSHAN

Password: Store in Vault ... Remove from Vault

☐ Authenticate Using SSH Key

SSH Key Path: Browse

New Delete Duplicate Move Up Move Down Test Connection Close

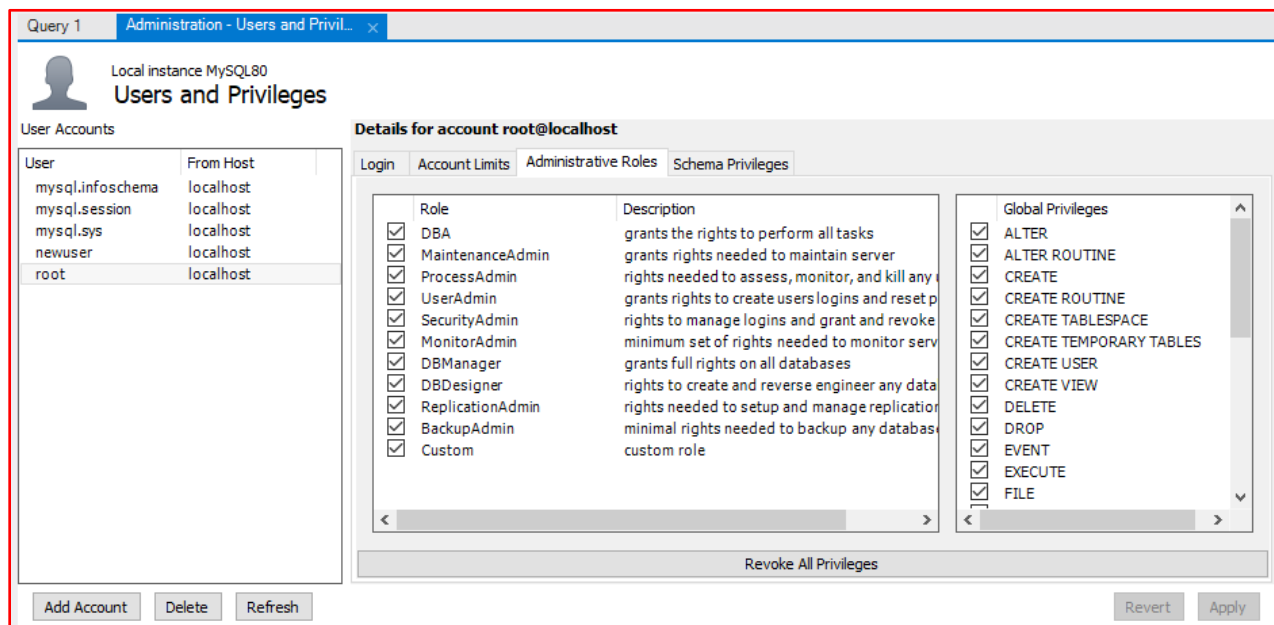
4) Maintain individual login credentials for the people who access the workstation and to perform administrative tasks of the database.

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

Local instance MySQL80
Users and Privileges

User Accounts

User	From Host
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
newuser	localhost
root	localhost

Details for account **mysql.sys@localhost**

Login Account Limits Administrative Roles Schema Privileges

Role	Description
<input type="checkbox"/> DBA	grants the rights to perform all tasks
<input checked="" type="checkbox"/> MaintenanceAdmin	grants rights needed to maintain server
<input checked="" type="checkbox"/> ProcessAdmin	rights needed to assess, monitor, and kill any user process
<input checked="" type="checkbox"/> UserAdmin	grants rights to create users logins and reset passwords
<input checked="" type="checkbox"/> SecurityAdmin	rights to manage logins and grant and revoke server authentication
<input type="checkbox"/> MonitorAdmin	minimum set of rights needed to monitor server
<input checked="" type="checkbox"/> DBManager	grants full rights on all databases
<input checked="" type="checkbox"/> DBDesigner	rights to create and reverse engineer any database schema
<input type="checkbox"/> ReplicationAdmin	rights needed to setup and manage replication
<input checked="" type="checkbox"/> BackupAdmin	minimal rights needed to backup any database

Global Privileges

- ☒ ALTER
- ☒ ALTER ROUTINE
- ☒ CREATE
- ☒ CREATE ROUTINE
- ☒ CREATE TABLESPACE
- ☒ CREATE TEMPORARY TABLES
- ☒ CREATE USER
- ☒ CREATE VIEW
- ☒ DELETE
- ☒ DROP
- ☒ EVENT
- ☐ EXECUTE
- ☐ FILE

Revoke All Privileges

Add Account Delete Refresh Revert Apply

iii. System User

Local instance MySQL80
Users and Privileges

User Accounts

User	From Host
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
newuser	localhost
root	localhost
system	localhost

Details for account **system@localhost**

Login Account Limits Administrative Roles Schema Privileges

Max. Queries: Number of queries the account can execute within one hour.

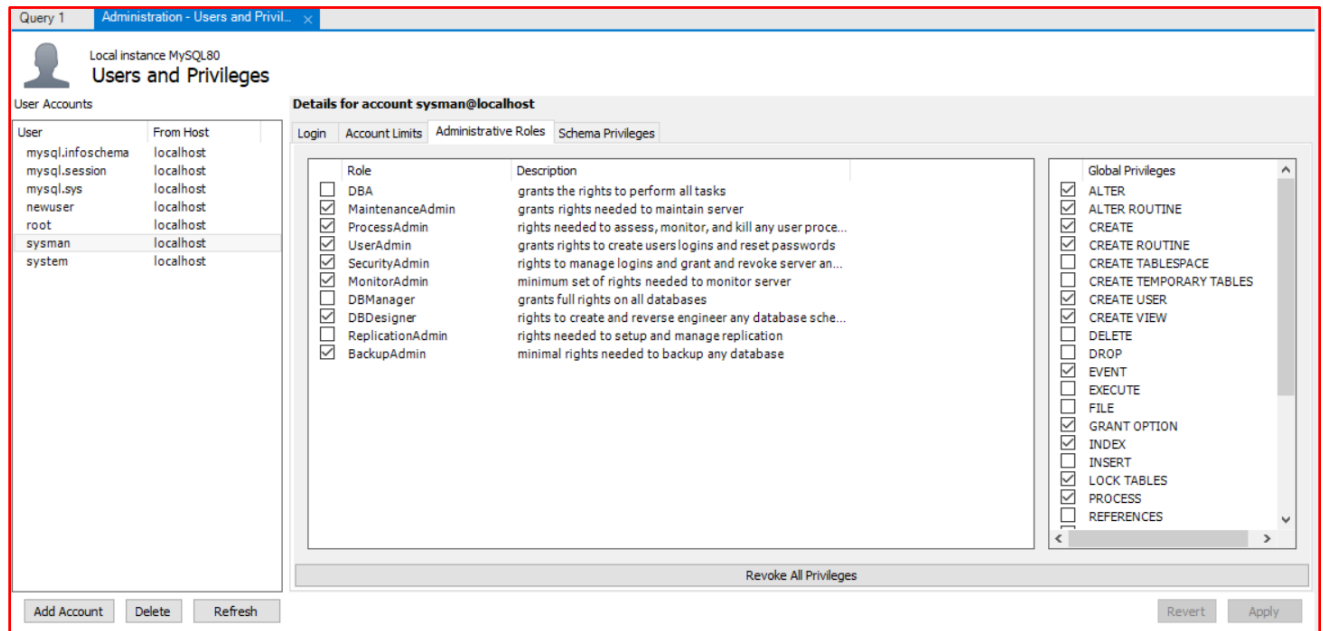
Max. Updates: Number of updates the account can execute within one hour.

Max. Connections: The number of times the account can connect to the server per hour.

Concurrent Connections: The number of simultaneous connections to the server the account can have.

Add Account Delete Refresh Revert Apply

iv. Sysman User



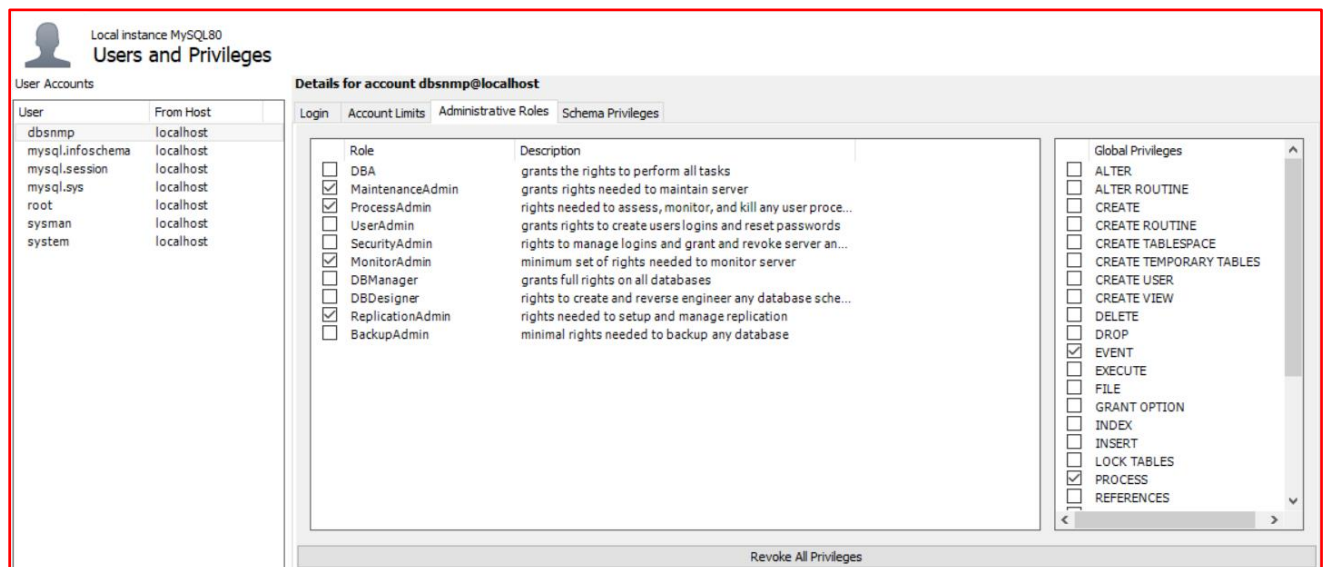
The screenshot shows the 'Users and Privileges' window for a local instance of MySQL 8.0. The 'User Accounts' table on the left lists several users, with 'sysman' selected. The 'Details for account sysman@localhost' pane shows the 'Administrative Roles' tab, where various roles are checked, including DBA, MaintenanceAdmin, ProcessAdmin, UserAdmin, SecurityAdmin, MonitorAdmin, DBManager, DBDesigner, ReplicationAdmin, and BackupAdmin. The 'Global Privileges' list on the right includes ALTER, ALTER ROUTINE, CREATE, CREATE ROUTINE, CREATE TABLESPACE, CREATE TEMPORARY TABLES, CREATE USER, CREATE VIEW, DELETE, DROP, EVENT, EXECUTE, FILE, GRANT OPTION, INDEX, INSERT, LOCK TABLES, PROCESS, and REFERENCES. Buttons at the bottom include 'Add Account', 'Delete', 'Refresh', 'Revoke All Privileges', 'Revert', and 'Apply'.

User	From Host
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
newuser	localhost
root	localhost
sysman	localhost
system	localhost

Role	Description
<input type="checkbox"/> DBA	grants the rights to perform all tasks
<input checked="" type="checkbox"/> MaintenanceAdmin	grants rights needed to maintain server
<input checked="" type="checkbox"/> ProcessAdmin	rights needed to assess, monitor, and kill any user process
<input checked="" type="checkbox"/> UserAdmin	grants rights to create users logins and reset passwords
<input checked="" type="checkbox"/> SecurityAdmin	rights to manage logins and grant and revoke server privileges
<input checked="" type="checkbox"/> MonitorAdmin	minimum set of rights needed to monitor server
<input checked="" type="checkbox"/> DBManager	grants full rights on all databases
<input checked="" type="checkbox"/> DBDesigner	rights to create and reverse engineer any database schema
<input checked="" type="checkbox"/> ReplicationAdmin	rights needed to setup and manage replication
<input checked="" type="checkbox"/> BackupAdmin	minimal rights needed to backup any database

Global Privileges
<input checked="" type="checkbox"/> ALTER
<input checked="" type="checkbox"/> ALTER ROUTINE
<input checked="" type="checkbox"/> CREATE
<input checked="" type="checkbox"/> CREATE ROUTINE
<input checked="" type="checkbox"/> CREATE TABLESPACE
<input checked="" type="checkbox"/> CREATE TEMPORARY TABLES
<input checked="" type="checkbox"/> CREATE USER
<input checked="" type="checkbox"/> CREATE VIEW
<input checked="" type="checkbox"/> DELETE
<input checked="" type="checkbox"/> DROP
<input checked="" type="checkbox"/> EVENT
<input checked="" type="checkbox"/> EXECUTE
<input checked="" type="checkbox"/> FILE
<input checked="" type="checkbox"/> GRANT OPTION
<input checked="" type="checkbox"/> INDEX
<input checked="" type="checkbox"/> INSERT
<input checked="" type="checkbox"/> LOCK TABLES
<input checked="" type="checkbox"/> PROCESS
<input checked="" type="checkbox"/> REFERENCES

v. Dbsnmp User



The screenshot shows the 'Users and Privileges' window for a local instance of MySQL 8.0. The 'User Accounts' table on the left lists several users, with 'dbsnmp' selected. The 'Details for account dbsnmp@localhost' pane shows the 'Administrative Roles' tab, where various roles are checked, including MaintenanceAdmin, ProcessAdmin, UserAdmin, SecurityAdmin, MonitorAdmin, DBManager, DBDesigner, ReplicationAdmin, and BackupAdmin. The 'Global Privileges' list on the right includes ALTER, ALTER ROUTINE, CREATE, CREATE ROUTINE, CREATE TABLESPACE, CREATE TEMPORARY TABLES, CREATE USER, CREATE VIEW, DELETE, DROP, EVENT, EXECUTE, FILE, GRANT OPTION, INDEX, INSERT, LOCK TABLES, PROCESS, and REFERENCES. Buttons at the bottom include 'Revoke All Privileges', 'Revert', and 'Apply'.

User	From Host
dbsnmp	localhost
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
root	localhost
sysman	localhost
system	localhost

Role	Description
<input type="checkbox"/> DBA	grants the rights to perform all tasks
<input checked="" type="checkbox"/> MaintenanceAdmin	grants rights needed to maintain server
<input checked="" type="checkbox"/> ProcessAdmin	rights needed to assess, monitor, and kill any user process
<input type="checkbox"/> UserAdmin	grants rights to create users logins and reset passwords
<input type="checkbox"/> SecurityAdmin	rights to manage logins and grant and revoke server privileges
<input checked="" type="checkbox"/> MonitorAdmin	minimum set of rights needed to monitor server
<input checked="" type="checkbox"/> DBManager	grants full rights on all databases
<input type="checkbox"/> DBDesigner	rights to create and reverse engineer any database schema
<input checked="" type="checkbox"/> ReplicationAdmin	rights needed to setup and manage replication
<input checked="" type="checkbox"/> BackupAdmin	minimal rights needed to backup any database

Global Privileges
<input type="checkbox"/> ALTER
<input type="checkbox"/> ALTER ROUTINE
<input type="checkbox"/> CREATE
<input type="checkbox"/> CREATE ROUTINE
<input type="checkbox"/> CREATE TABLESPACE
<input type="checkbox"/> CREATE TEMPORARY TABLES
<input type="checkbox"/> CREATE USER
<input type="checkbox"/> CREATE VIEW
<input type="checkbox"/> DELETE
<input type="checkbox"/> DROP
<input checked="" type="checkbox"/> EVENT
<input checked="" type="checkbox"/> EXECUTE
<input type="checkbox"/> FILE
<input type="checkbox"/> GRANT OPTION
<input type="checkbox"/> INDEX
<input type="checkbox"/> INSERT
<input type="checkbox"/> LOCK TABLES
<input checked="" type="checkbox"/> PROCESS
<input type="checkbox"/> REFERENCES

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)
```

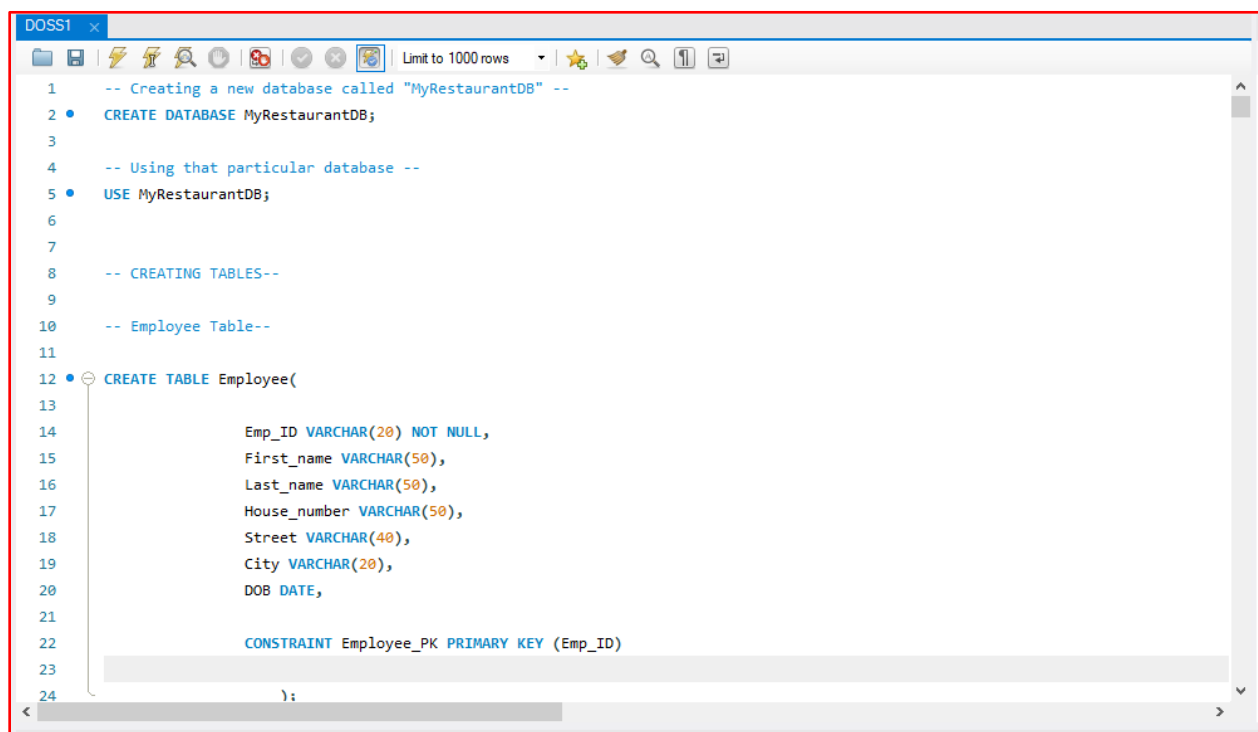
```
mysql> FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.00 sec)
```

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.



The screenshot shows a SQL query editor window titled "DOSS1". The editor contains the following SQL code:

```
1  -- Creating a new database called "MyRestaurantDB" --
2  • CREATE DATABASE MyRestaurantDB;
3
4  -- Using that particular database --
5  • USE MyRestaurantDB;
6
7
8  -- CREATING TABLES--
9
10 -- Employee Table--
11
12 • CREATE TABLE Employee(
13
14     Emp_ID VARCHAR(20) NOT NULL,
15     First_name VARCHAR(50),
16     Last_name VARCHAR(50),
17     House_number VARCHAR(50),
18     Street VARCHAR(40),
19     City VARCHAR(20),
20     DOB DATE,
21
22     CONSTRAINT Employee_PK PRIMARY KEY (Emp_ID)
23
24 );
```

```

405 -- Customer Table--
406
407 • INSERT INTO Customer (Customer_ID, First_name, Last_name, House_number, Street, City, Payment_ID, Cart_ID, Emp_ID) VALUES ('CUS0
408 • INSERT INTO Customer (Customer_ID, First_name, Last_name, House_number, Street, City, Payment_ID, Cart_ID, Emp_ID) VALUES ('CUS0
409 • INSERT INTO Customer (Customer_ID, First_name, Last_name, House_number, Street, City, Payment_ID, Cart_ID, Emp_ID) VALUES ('CUS0
410 • INSERT INTO Customer (Customer_ID, First_name, Last_name, House_number, Street, City, Payment_ID, Cart_ID, Emp_ID) VALUES ('CUS0
411 • INSERT INTO Customer (Customer_ID, First_name, Last_name, House_number, Street, City, Payment_ID, Cart_ID, Emp_ID) VALUES ('CUS0
412
413
414 -- User_account Table --
415
416 • INSERT INTO User_account (Username,password,Customer_ID) VALUES('jayani','jayani123','CUS001');
417 • INSERT INTO User_account (Username,password,Customer_ID) VALUES('samadhi','samadhi123','CUS002');
418 • INSERT INTO User_account (Username,password,Customer_ID) VALUES('eranda','eranda123','CUS003');
419 • INSERT INTO User_account (Username,password,Customer_ID) VALUES('tharindu','tharindu123','CUS004');
420 • INSERT INTO User_account (Username,password,Customer_ID) VALUES('chamodhi','chamodhi123','CUS005');
421
422
423 -- Orders Table --
424
425 • INSERT INTO Orders(Order_ID,Order_Date,Delivery_Date,Total_Price,Order_Status,Customer_ID,Cart_ID,Emp_ID) VALUES('001','2019-09-
426 • INSERT INTO Orders(Order_ID,Order_Date,Delivery_Date,Total_Price,Order_Status,Customer_ID,Cart_ID,Emp_ID) VALUES('002','2019-08-
427 • INSERT INTO Orders(Order_ID,Order_Date,Delivery_Date,Total_Price,Order_Status,Customer_ID,Cart_ID,Emp_ID) VALUES('003','2019-08-
428 • 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: | Exports: | Wrap Cell Contents: |

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 x | Read Only

DOSS1* x | Limit to 1000 rows |

```

1 • select * from Employee;
2
3 -- Creating a new database called "MyRestaurantDB" --
4 • CREATE DATABASE MyRestaurantDB;
5

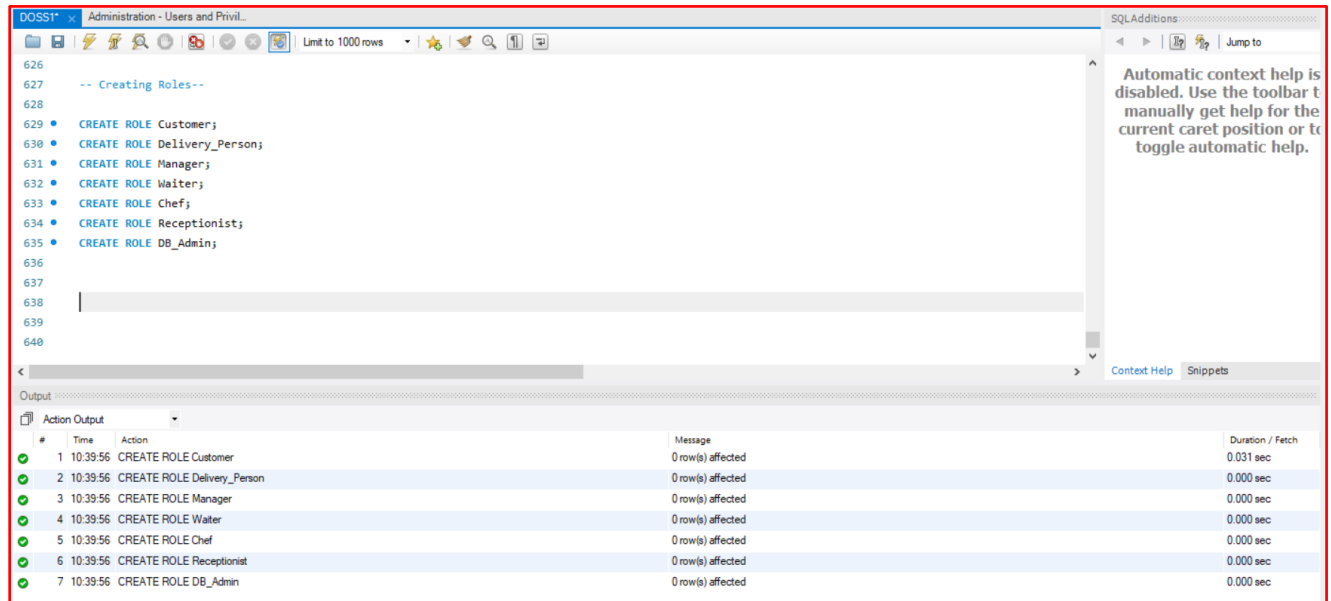
```

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 x | Apply | Revert

Step 1 :- Creating Roles

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



The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a SQL script with the following commands:

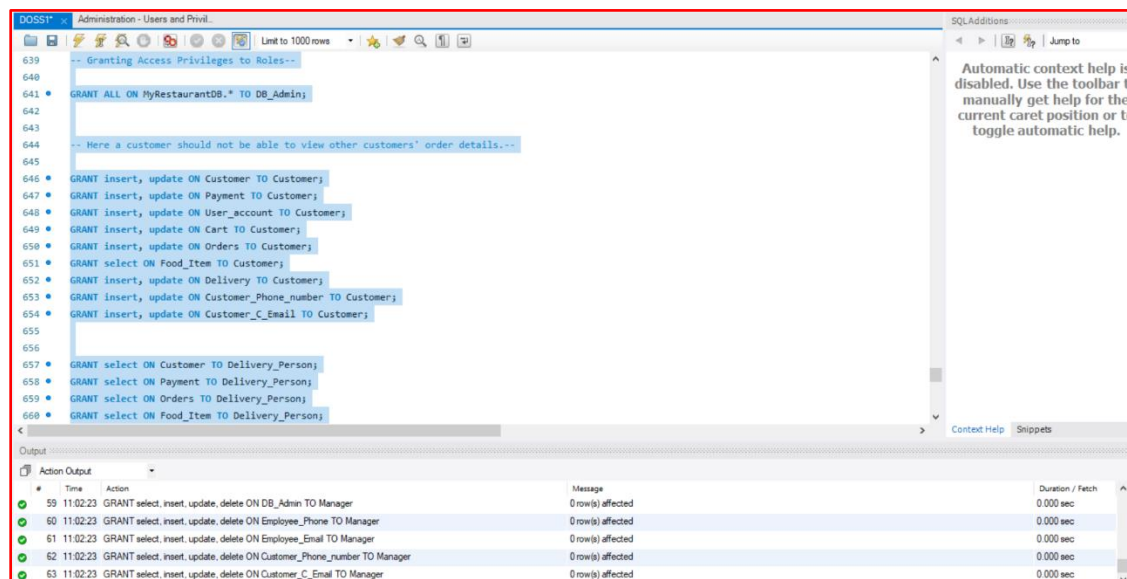
```
-- Creating Roles--
CREATE ROLE Customer;
CREATE ROLE Delivery_Person;
CREATE ROLE Manager;
CREATE ROLE Waiter;
CREATE ROLE Chef;
CREATE ROLE Receptionist;
CREATE ROLE DB_Admin;
```

The bottom pane shows the 'Output' window with the following results:

#	Time	Action	Message	Duration / Fetch
1	10:39:56	CREATE ROLE Customer	0 row(s) affected	0.031 sec
2	10:39:56	CREATE ROLE Delivery_Person	0 row(s) affected	0.000 sec
3	10:39:56	CREATE ROLE Manager	0 row(s) affected	0.000 sec
4	10:39:56	CREATE ROLE Waiter	0 row(s) affected	0.000 sec
5	10:39:56	CREATE ROLE Chef	0 row(s) affected	0.000 sec
6	10:39:56	CREATE ROLE Receptionist	0 row(s) affected	0.000 sec
7	10:39:56	CREATE ROLE DB_Admin	0 row(s) affected	0.000 sec

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.



The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a SQL script with the following commands:

```
-- Granting Access Privileges to Roles--
GRANT ALL ON MyRestaurantDB.* TO DB_Admin;

-- Here a customer should not be able to view other customers' order details--
GRANT insert, update ON Customer TO Customer;
GRANT insert, update ON Payment TO Customer;
GRANT insert, update ON User_account TO Customer;
GRANT insert, update ON Cart TO Customer;
GRANT insert, update ON Orders TO Customer;
GRANT select ON Food_Item TO Customer;
GRANT insert, update ON Delivery TO Customer;
GRANT insert, update ON Customer_Phone_number TO Customer;
GRANT insert, update ON Customer_C_Email TO Customer;

GRANT select ON Customer TO Delivery_Person;
GRANT select ON Payment TO Delivery_Person;
GRANT select ON Orders TO Delivery_Person;
GRANT select ON Food_Item TO Delivery_Person;
```

The bottom pane shows the 'Output' window with the following results:

#	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

DOSST Administration - Users and Privil.

SQLAdditions

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

```

669 • GRANT select ON Orders TO Waiter;
670 • GRANT select ON Food_Item TO Waiter;
671 • GRANT select ON Waiter TO Waiter;
672 • GRANT select ON Employee_Phone TO Waiter;
673
674
675 • GRANT select ON Customer TO Chef;
676 • GRANT select ON Orders TO Chef;
677 • GRANT select ON Cart TO Chef;
678 • GRANT select ON Chef TO Chef;
679 • GRANT select ON Employee_Phone TO Chef;
680
681
682 • GRANT select ON Customer TO Receptionist;
683 • GRANT select ON Payment TO Receptionist;
684 • GRANT select ON Cart TO Receptionist;
685 • GRANT select ON Orders TO Receptionist;
686 • GRANT select ON Food_Item TO Receptionist;
687 • GRANT select ON Delivery TO Receptionist;
688 • GRANT select ON Employee TO Receptionist;
689 • GRANT select ON Delivery_Person TO Receptionist;
690 • GRANT select ON Manager TO Receptionist;

```

Output

Action Output

#	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

DOSST Administration - Users and Privil.

SQLAdditions

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

```

696 • GRANT select ON Employee_Email TO Receptionist;
697 • GRANT select ON Customer_Phone_number TO Receptionist;
698 • GRANT select ON Customer_C_Email TO Receptionist;
699
700
701 • GRANT select, insert, update, delete ON Customer TO Manager;
702 • GRANT select, insert, update, delete ON Payment TO Manager;
703 • GRANT select, insert, update, delete ON Cart TO Manager;
704 • GRANT select, insert, update, delete ON Orders TO Manager;
705 • GRANT select, insert, update, delete ON Food_Item TO Manager;
706 • GRANT select, insert, update, delete ON Delivery TO Manager;
707 • GRANT select, insert, update, delete ON Employee TO Manager;
708 • GRANT select, insert, update, delete ON Delivery_Person TO Manager;
709 • GRANT select, insert, update, delete ON Manager TO Manager;
710 • GRANT select, insert, update, delete ON Waiter TO Manager;
711 • GRANT select, insert, update, delete ON Chef TO Manager;
712 • GRANT select, insert, update, delete ON Receptionist TO Manager;
713 • GRANT select, insert, update, delete ON DB_Admin TO Manager;
714 • GRANT select, insert, update, delete ON Employee_Phone TO Manager;
715 • GRANT select, insert, update, delete ON Employee_Email TO Manager;
716 • GRANT select, insert, update, delete ON Customer_Phone_number TO Manager;
717 • GRANT select, insert, update, delete ON Customer_C_Email TO Manager;

```

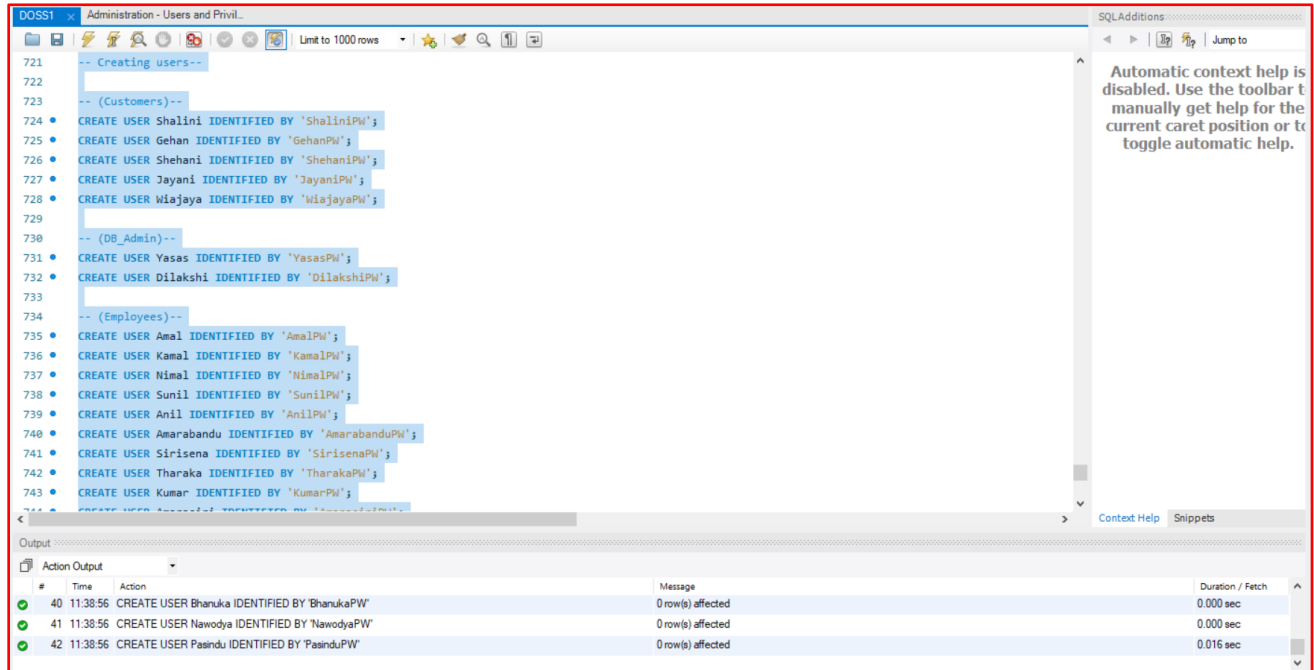
Output

Action Output

#	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

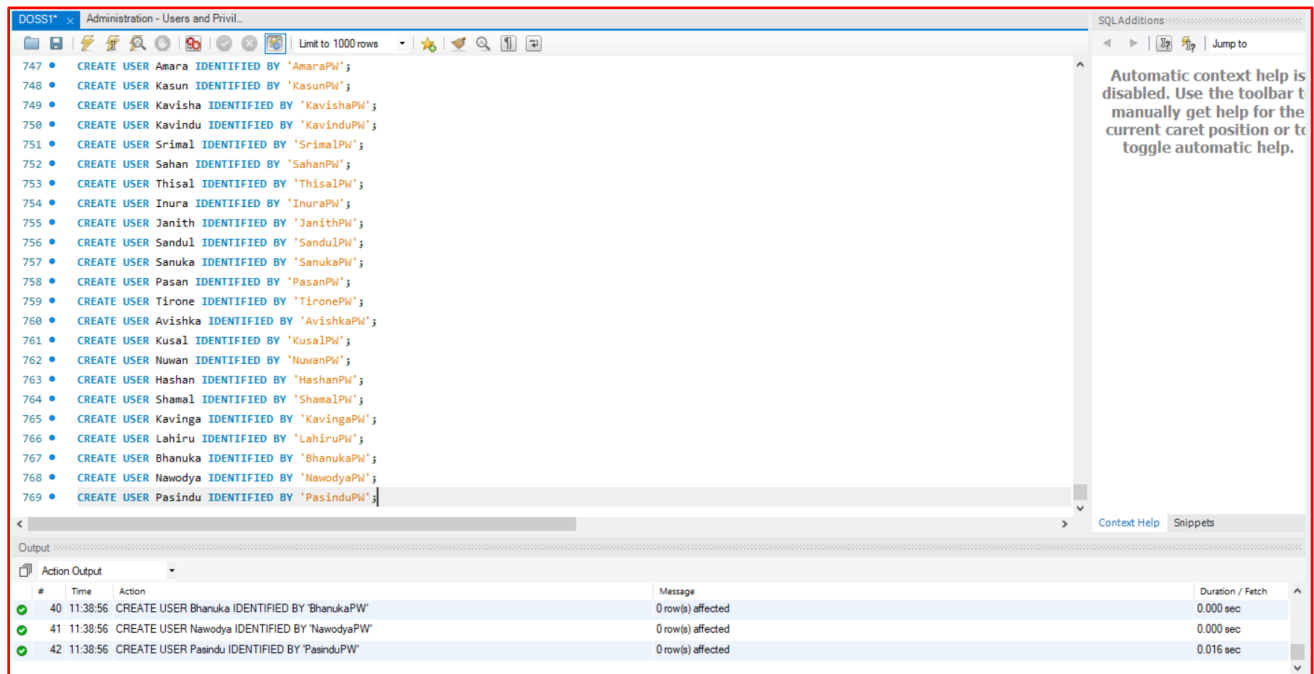
In the next step, you need to create some users with their respective account passwords. So that, we can use those users later onwards to assign them with the necessary roles that we have created in the previous step.



The screenshot shows the SQL Developer interface with a script titled "Administration - Users and Priv...". The script contains SQL commands to create users for three groups: Customers, DB_Admin, and Employees. The output window at the bottom shows the execution results for three rows.

```
-- Creating users--
-- (Customers)--
724 CREATE USER Shalini IDENTIFIED BY 'ShaliniPW';
725 CREATE USER Gehan IDENTIFIED BY 'GehanPW';
726 CREATE USER Shehani IDENTIFIED BY 'ShehaniPW';
727 CREATE USER Jayani IDENTIFIED BY 'JayaniPW';
728 CREATE USER Wajaya IDENTIFIED BY 'WajayaPW';
729
-- (DB_Admin)--
731 CREATE USER Yasas IDENTIFIED BY 'YasasPW';
732 CREATE USER Dilakshi IDENTIFIED BY 'DilakshiPW';
733
-- (Employees)--
735 CREATE USER Amal IDENTIFIED BY 'AmalPW';
736 CREATE USER Kamal IDENTIFIED BY 'KamalPW';
737 CREATE USER Nimal IDENTIFIED BY 'NimalPW';
738 CREATE USER Sunil IDENTIFIED BY 'SunilPW';
739 CREATE USER Anil IDENTIFIED BY 'AnilPW';
740 CREATE USER Amarabandu IDENTIFIED BY 'AmarabanduPW';
741 CREATE USER Sirisena IDENTIFIED BY 'SirisenaPW';
742 CREATE USER Tharaka IDENTIFIED BY 'TharakaPW';
743 CREATE USER Kumar IDENTIFIED BY 'KumarPW';
744 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



The screenshot shows the SQL Developer interface with a script titled "Administration - Users and Priv...". The script continues with SQL commands to create users for various roles. The output window at the bottom shows the execution results for three rows.

```
747 CREATE USER Amara IDENTIFIED BY 'AmaraPW';
748 CREATE USER Kasun IDENTIFIED BY 'KasunPW';
749 CREATE USER Kavisha IDENTIFIED BY 'KavishaPW';
750 CREATE USER Kavindu IDENTIFIED BY 'KavinduPW';
751 CREATE USER Srimal IDENTIFIED BY 'SrimalPW';
752 CREATE USER Sahan IDENTIFIED BY 'SahanPW';
753 CREATE USER Thisal IDENTIFIED BY 'ThisalPW';
754 CREATE USER Inura IDENTIFIED BY 'InuraPW';
755 CREATE USER Janith IDENTIFIED BY 'JanithPW';
756 CREATE USER Sandul IDENTIFIED BY 'SandulPW';
757 CREATE USER Sanuka IDENTIFIED BY 'SanukaPW';
758 CREATE USER Pasan IDENTIFIED BY 'PasanPW';
759 CREATE USER Tirone IDENTIFIED BY 'TironePW';
760 CREATE USER Avishka IDENTIFIED BY 'AvishkaPW';
761 CREATE USER Kusal IDENTIFIED BY 'KusalPW';
762 CREATE USER Nuwan IDENTIFIED BY 'NuwanPW';
763 CREATE USER Hashan IDENTIFIED BY 'HashanPW';
764 CREATE USER Shamal IDENTIFIED BY 'ShamalPW';
765 CREATE USER Kavinga IDENTIFIED BY 'KavingaPW';
766 CREATE USER Lahiru IDENTIFIED BY 'LahiruPW';
767 CREATE USER Bhanuka IDENTIFIED BY 'BhanukaPW';
768 CREATE USER Nawodya IDENTIFIED BY 'NawodyaPW';
769 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.

SQL Developer - Administration - Users and Privileges

-- Assigning the roles to users--

```
773 GRANT Customer TO Shalini;
774
775 GRANT Customer TO Gehan;
776
777 GRANT Customer TO Shehani;
778
779 GRANT Customer TO Jayani;
780
781 GRANT Customer TO Wiajaya;
782
783
784 GRANT DB_Admin TO Yasa;
785
786 GRANT DB_Admin TO Dilakshi;
787
788
789 GRANT Manager TO Kavisha;
790
791 GRANT Manager TO Kavindu;
792
793 GRANT Manager TO Srimal;
794
795 GRANT Manager TO Sahan;
796
797 GRANT Manager TO Thisal;
798
799
800 GRANT Delivery_Person TO Nimal;
801
802 GRANT Delivery_Person TO Amarabandu;
803
804 GRANT Delivery_Person TO Sirisen;
805
806 GRANT Delivery_Person TO Tharaka;
807
808 GRANT Delivery_Person TO Kumar;
809
810
811
812
813
```

Output

#	Time	Action	Message	Duration / Fetch
30	11:42:17	GRANT Receptionist TO Eshan	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

SQL Developer - Administration - Users and Privileges

```
791 GRANT Delivery_Person TO Amarabandu;
792
793 GRANT Delivery_Person TO Sirisen;
794
795 GRANT Delivery_Person TO Tharaka;
796
797 GRANT Delivery_Person TO Kumar;
798
799
800 GRANT Waiter TO Inura;
801
802 GRANT Waiter TO Janith;
803
804 GRANT Waiter TO Sandul;
805
806 GRANT Waiter TO Sanuka;
807
808 GRANT Waiter TO Pasan;
809
810
811 GRANT Chef TO Tirone;
812
813 GRANT Chef TO Avishka;
814
815 GRANT Chef TO Kusal;
816
817 GRANT Chef TO Nuwan;
818
819 GRANT Chef TO Hashan;
820
821 GRANT Chef TO Shamal;
822
823
824 GRANT Receptionist TO Thushal;
825
826 GRANT Receptionist TO Eshan;
827
828 GRANT Receptionist TO Amara;
829
830 GRANT Receptionist TO Kasun;
831
832
```

Output

#	Time	Action	Message	Duration / Fetch
30	11:42:17	GRANT Receptionist TO Eshan	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.

```
815 -- Storing user account passwords securely --
816
817 CREATE TABLE User_Credentials(
818
819     Username VARCHAR(255) NOT NULL,
820     Password VARCHAR(255) NOT NULL,
821
822     CONSTRAINT pk_credentials PRIMARY KEY (Username)
823
824 );
```

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

```
826 -- Method 1 :- MD5 Hashing --
827
828 INSERT INTO User_Credentials(Username, Password) VALUES ('Amal_J', md5('P455w0R$d1'));
829 INSERT INTO User_Credentials(Username, Password) VALUES ('Kamal_G', md5('P455w0R$d2'));
830 INSERT INTO User_Credentials(Username, Password) VALUES ('Nimal_L', md5('P455w0R$d3'));
831 INSERT INTO User_Credentials(Username, Password) VALUES ('Sunil_N', md5('P455w0R$d4'));
832 INSERT INTO User_Credentials(Username, Password) VALUES ('Anil_S', md5('P455w0R$d5'));
833 INSERT INTO User_Credentials(Username, Password) VALUES ('Amarabandu_R', md5('P455w0R$d6'));
834 INSERT INTO User_Credentials(Username, Password) VALUES ('Sirisena_F', md5('P455w0R$d7'));
835 INSERT INTO User_Credentials(Username, Password) VALUES ('Tharaka_K', md5('P455w0R$d8'));
```

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

```
837 -- Method 2 :- SHA1 Hashing --
838
839 • INSERT INTO User_Credentials(Username, Password) VALUES ('Kumar_D', sha1('P455w0R$d9'));
840 • INSERT INTO User_Credentials(Username, Password) VALUES ('Amarasiri_P', sha1('P455w0R$d10'));
841 • INSERT INTO User_Credentials(Username, Password) VALUES ('Thushal_K', sha1('P455w0R$d11'));
842 • INSERT INTO User_Credentials(Username, Password) VALUES ('Eishan_D', sha1('P455w0R$d12'));
843 • INSERT INTO User_Credentials(Username, Password) VALUES ('Amara_N', sha1('P455w0R$d1E'));
844 • INSERT INTO User_Credentials(Username, Password) VALUES ('Kasun_L', sha1('P455w0R$d14'));
845 • INSERT INTO User_Credentials(Username, Password) VALUES ('Kavisha_S', sha1('P455w0R$d15'));
846 • INSERT INTO User_Credentials(Username, Password) VALUES ('Kavindu_D', sha1('P455w0R$d16'));
847
```

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\$d15'))	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

```
847 -- Method 3 :- AES Encryption --
848
849 • INSERT INTO User_Credentials(Username, Password) VALUES ('Srimal_K', aes_encrypt('P455w0R$d17', 'secret'));
850 • INSERT INTO User_Credentials(Username, Password) VALUES ('Sahan_A', aes_encrypt('P455w0R$d18', 'secret'));
851 • INSERT INTO User_Credentials(Username, Password) VALUES ('Thisal_P', aes_encrypt('P455w0R$d19', 'secret'));
852 • INSERT INTO User_Credentials(Username, Password) VALUES ('Inura_K', aes_encrypt('P455w0R$d20', 'secret'));
853 • INSERT INTO User_Credentials(Username, Password) VALUES ('Janith_M', aes_encrypt('P455w0R$d21', 'secret'));
854 • INSERT INTO User_Credentials(Username, Password) VALUES ('Sandul_L', aes_encrypt('P455w0R$d22', 'secret'));
855 • INSERT INTO User_Credentials(Username, Password) VALUES ('Sanuka_L', aes_encrypt('P455w0R$d23', 'secret'));
856 • INSERT INTO User_Credentials(Username, Password) VALUES ('Pasan_M', aes_encrypt('P455w0R$d24', 'secret'));
```

iv. DES Encryption

```
855 -- Method 4 :- DES Encryption --
856
857 • INSERT INTO User_Credentials(Username, Password) VALUES ('Tirone_L', des_encrypt('P455w0R$d25', 'secret'));
858 • INSERT INTO User_Credentials(Username, Password) VALUES ('Avishka_S', des_encrypt('P455w0R$d26', 'secret'));
859 • INSERT INTO User_Credentials(Username, Password) VALUES ('Kusal_M', des_encrypt('P455w0R$d27', 'secret'));
860 • INSERT INTO User_Credentials(Username, Password) VALUES ('Nuwan_A', des_encrypt('P455w0R$d28', 'secret'));
861 • INSERT INTO User_Credentials(Username, Password) VALUES ('Hashan_T', des_encrypt('P455w0R$d29', 'secret'));
862 • INSERT INTO User_Credentials(Username, Password) VALUES ('Shamal_K', des_encrypt('P455w0R$d30', 'secret'));
863 • INSERT INTO User_Credentials(Username, Password) VALUES ('Kavinga_Y', des_encrypt('P455w0R$d31', 'secret'));
864 • INSERT INTO User_Credentials(Username, Password) VALUES ('Pasindu_N', des_encrypt('P455w0R$d32', 'secret'));
```

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

The screenshot shows a database query tool interface. At the top, a SQL query is entered: `SELECT * FROM User_Credentials;`. Below the query, a table of results is displayed. The table has two columns: 'Username' and 'Password'. There are 16 rows of data, each containing a username and a long, hashed password string. The interface includes a toolbar with options like 'Result Grid', 'Filter Rows', 'Edit', 'Export/Import', and 'Wrap Cell Content'. On the right side, there are buttons for 'Result Grid', 'Form Editor', 'Field Types', and 'Query Stats'. At the bottom, an 'Output' section shows the execution details: 'Action Output' with a green checkmark, 'Time' 09:42:38, 'Action' `SELECT * FROM User_Credentials LIMIT 0, 1000`, and 'Message' '16 row(s) returned'.

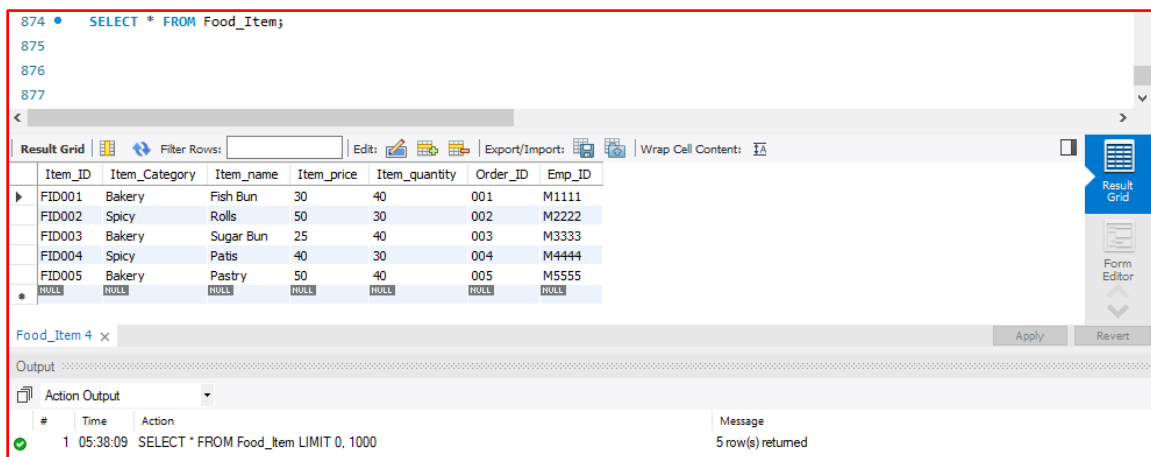
Username	Password
Amal_J	81d8bbb0576cdf7572b642605ece524
Amara_N	f97942f728be0b0aa349a2e6f8091b04d0fc4262
Amarabandu_R	34a42cdb047f79d4ed04e5ea871da07c
Amarasiri_P	a0c8bf2053d4cb94bb05d82b182798f43583aa31
Anil_S	f50767f0a8a319c0fae40259e370633c
Eishan_D	761ff786c87360928ff42b322087f537d828129d
Kamal_G	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

Output: Action Output
1 Time 09:42:38 Action SELECT * FROM User_Credentials LIMIT 0, 1000 Message 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.



```
874 • SELECT * FROM Food_Item;
```

Item_ID	Item_Category	Item_name	Item_price	Item_quantity	Order_ID	Emp_ID
FID001	Bakery	Fish Bun	30	40	001	M1111
FID002	Spicy	Rolls	50	30	002	M2222
FID003	Bakery	Sugar Bun	25	40	003	M3333
FID004	Spicy	Patis	40	30	004	M4444
FID005	Bakery	Pastry	50	40	005	M5555
NULL	NULL	NULL	NULL	NULL	NULL	NULL

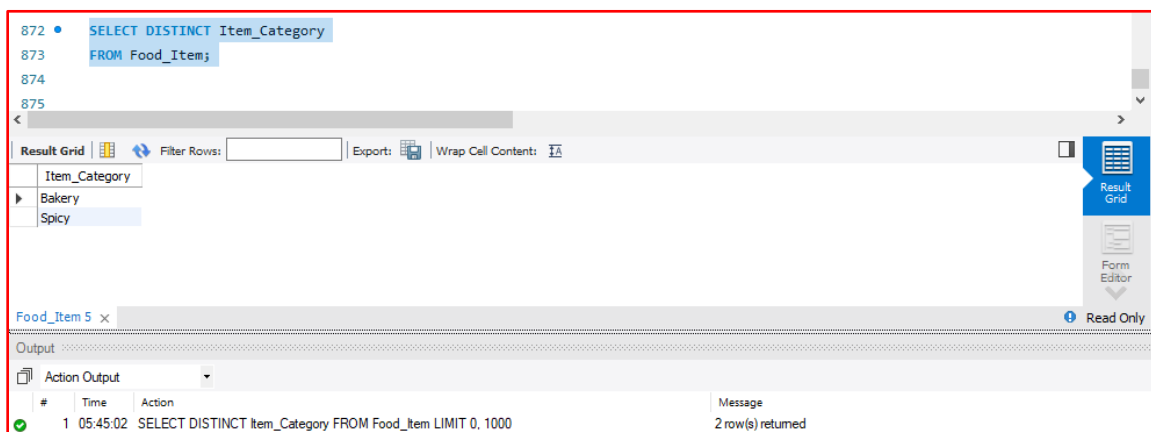
Food_Item 4 x

Output

Action Output

#	Time	Action	Message
1	05:38:09	SELECT * FROM Food_Item LIMIT 0, 1000	5 row(s) returned

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.



```
872 • SELECT DISTINCT Item_Category
```

```
873 FROM Food_Item;
```

Item_Category
Bakery
Spicy

Food_Item 5 x

Output

Action Output

#	Time	Action	Message
1	05:45:02	SELECT DISTINCT Item_Category FROM Food_Item LIMIT 0, 1000	2 row(s) returned

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

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

The screenshot displays a database query editor interface. At the top, a SQL query is entered in a text area:

```
875 • SELECT COUNT(DISTINCT Item_Category) AS 'Number of Food Categories'
876 FROM Food_Item;
877
878
```

Below the query editor, the 'Result Grid' is visible, showing a single row of results:

Number of Food Categories
2

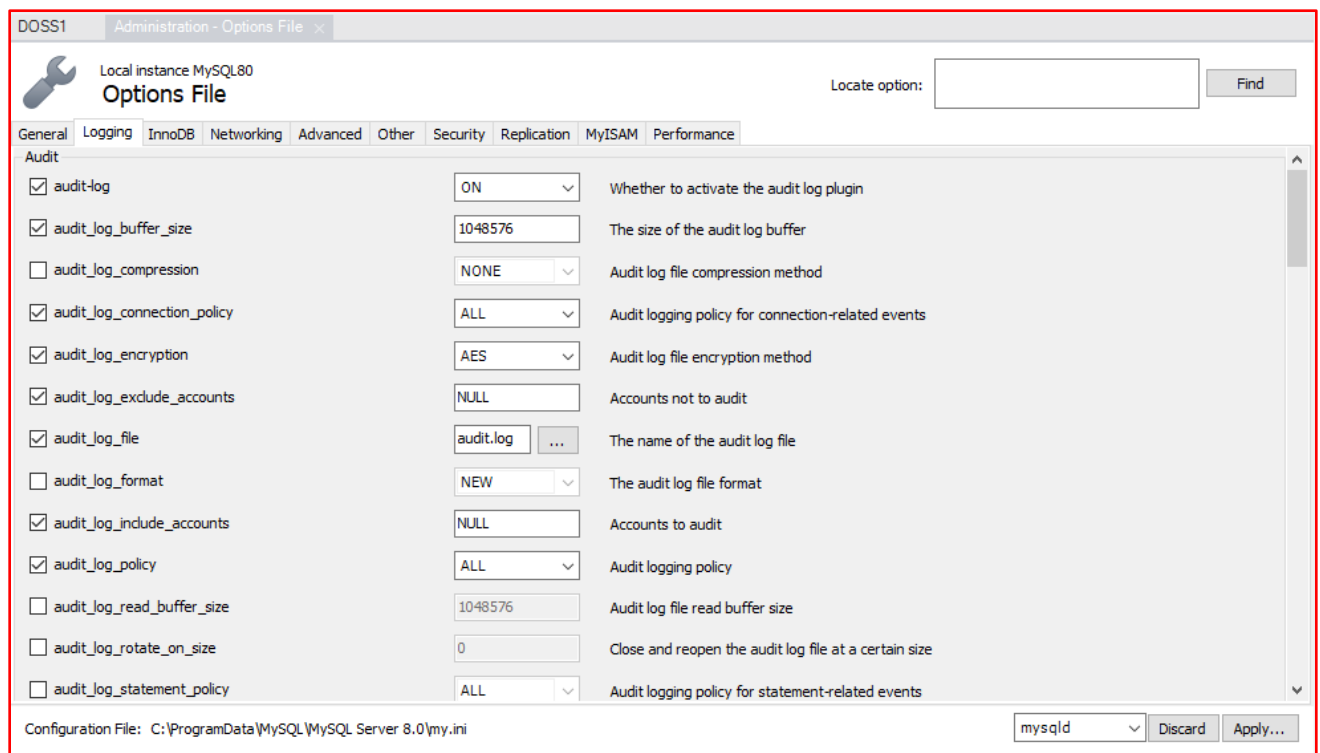
On the right side of the interface, there are buttons for 'Result Grid', 'Form Editor', and 'Read Only'. Below the result grid, the 'Output' section is visible, showing a log of the executed query:

#	Time	Action	Message
1	05:51:49	SELECT COUNT(DISTINCT Item_Category) AS 'Number of Food Categories' FROM Food_Item LIMIT...	1 row(s) returned

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.