Implementing PCI-DSS Compliant MySQL Deployment with Real-Time Monitoring on Azure

Objectives:

- Ensure the MySQL database meets PCI-DSS compliance requirements.
- Implement secure MySQL installation and configuration.
- Create and manage users with appropriate access control.

Goals:

- Secure installation and configuration of MySQL on Ubuntu Server 22.04.
- Implement encryption for data at rest and in transit.
- Configure access controls and logging.
- Ensure regular updates and monitoring.

Azure Resource Requirements:

1. **Ubuntu Server 22.04 VM**:
    - Standard DS1_v2 (2vCPU, 8 GB RAM) for MySQL database.
    - Attached Standard SSD Managed Disk (30 GB).
2. **Azure Network Security Group (NSG)**:
    - Configure inbound and outbound rules to restrict access.
3. **Azure Key Vault**:
    - Store and manage sensitive information like encryption keys.
4. **Azure Monitor and Log Analytics**:
    - For monitoring and logging MySQL activities.
5. **Azure Security Center/Microsoft Defender for Cloud**:
    - For security assessments and recommendations.

MAPPING PCI-DSS requirements to MySQL database implementation

let's review the PCI-DSS V4.0 requirements and map each requirement to the MySQL database implementation phase for this project on Azure. PCI-DSS (Payment Card Industry Data Security Standard) is a set of security standards designed to ensure that all companies that accept, process, store, or transmit credit card information maintain a secure environment.

Here is a summarized mapping of each PCI-DSS V4.0 requirement to the MySQL database implementation phase on Azure:

PCI-DSS Requirement	Implementation Phase for MySQL on Azure
Requirement 1: Install and maintain a firewall configuration to protect cardholder data	Use NSGs to control traffic, configure firewalls
Requirement 2: Do not use vendor-supplied defaults for system passwords and other security parameters	Change default passwords, configure security parameters
Requirement 3: Protect stored cardholder data	Encrypt data at rest, implement data masking

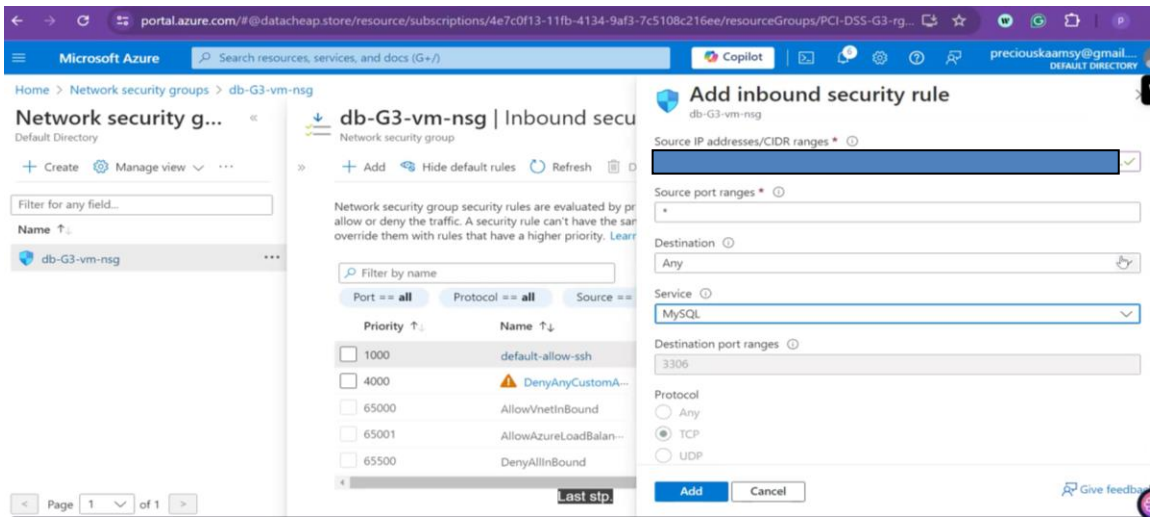
PCI-DSS Requirement	Implementation Phase for MySQL on Azure
Requirement 4: Encrypt transmission of cardholder data across open, public networks	Configure SSL/TLS for MySQL
Requirement 5: Protect all systems against malware and regularly update anti-virus software or programs	Regularly update MySQL and OS, use Azure Security Center
Requirement 6: Develop and maintain secure systems and applications	Apply MySQL patches, follow secure coding practices
Requirement 7: Restrict access to cardholder data by business need to know	Implement RBAC, use Azure AD
Requirement 8: Identify and authenticate access to system components	Enforce strong passwords, implement MFA
Requirement 9: Restrict physical access to cardholder data	Use Azure's physical security, secure backups
Requirement 10: Track and monitor all access to network resources and cardholder data	Enable MySQL logging, use Azure Monitor and Log Analytics
Requirement 11: Regularly test security systems and processes	Perform vulnerability scans, use Azure Security Center
Requirement 12: Maintain a policy that addresses information security for all personnel	Develop security policies, conduct security training

By following this mapping, We can ensure that your MySQL database implementation on Azure meets PCI-DSS V4.0 compliance requirements.

Step-by-Step Procedure:

Step 1: Provision Azure Resources

1. **Create Ubuntu VM**:
    - Navigate to the Azure portal.
    - Create a new resource group (PCI-DSS-RG).
    - Create a Virtual Net name: PCI-DSS-G3
    - Create subnet: PCI-Database
    - Create a new Ubuntu Server 22.04 VM (Standard DS1_v2) in RG-PCI-DSS and select PCI-Database SUBNET.
    - Attach a Standard SSD Managed Disk (30 GB).

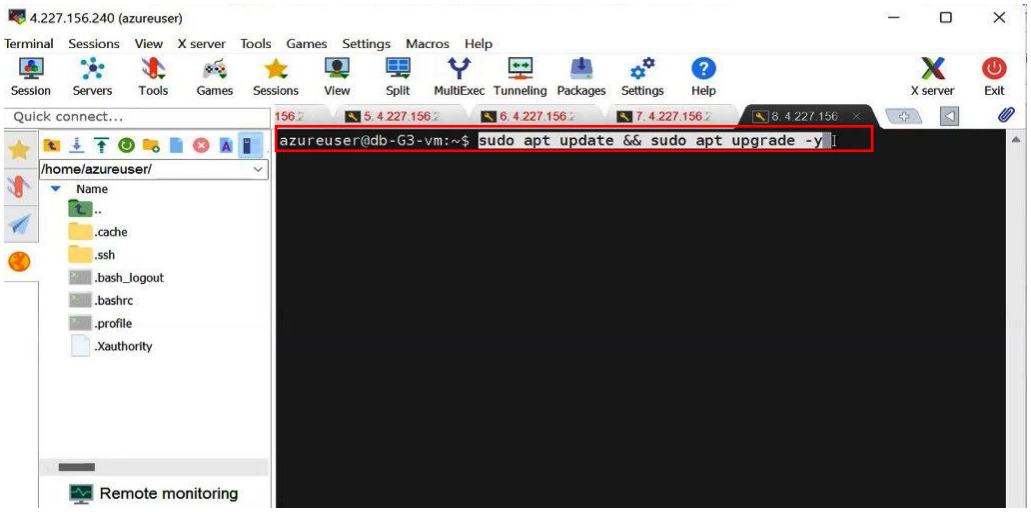


Step 2: Secure the VM

- Configure the VM with a Network Security Group (NSG) to allow SSH (port 22) and MySQL (port 3306) only from trusted IP addresses.

1. **Update the VM**:

   Use an SSH client to connect to your MySQL VM. Or **MOBAXTERM**
   sudo apt update && sudo apt upgrade –y



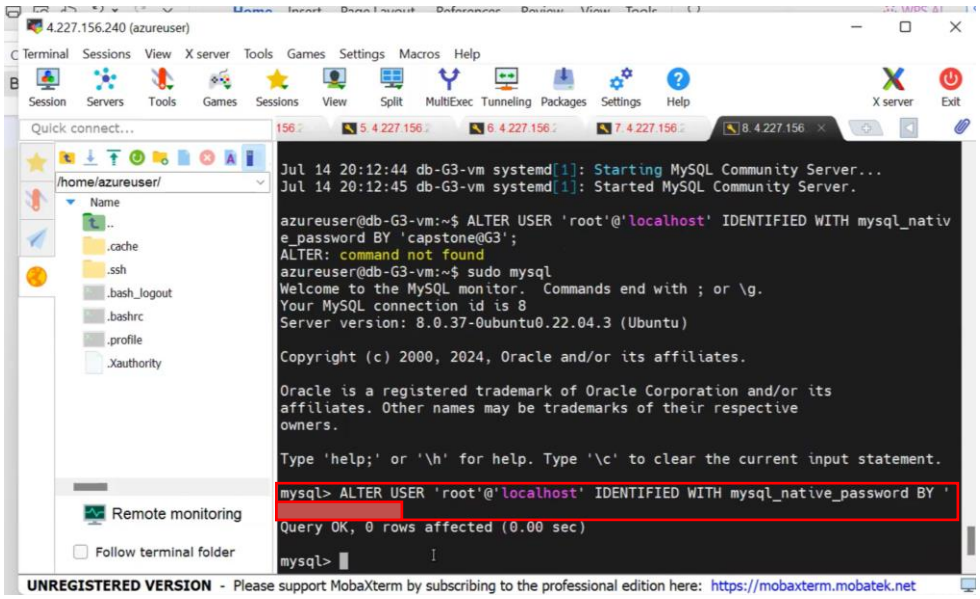
Step 3: Install MySQL

1. **Install MySQL Server**: or use Bash shell Script for auto-installation

   sudo apt install mysql-server –y

   "*Then run the following ALTER USER command to change the root user's authentication method to one that uses a password. The following example changes the authentication method to mysql_native_**password: as recommended by PCI-DSS V4.0***"

   ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '[redacted]';



To authenticate as the **root** MySQL user using a password, run this command:

Sudo mysql -u root -p

Then go back to using the default authentication method using this command:

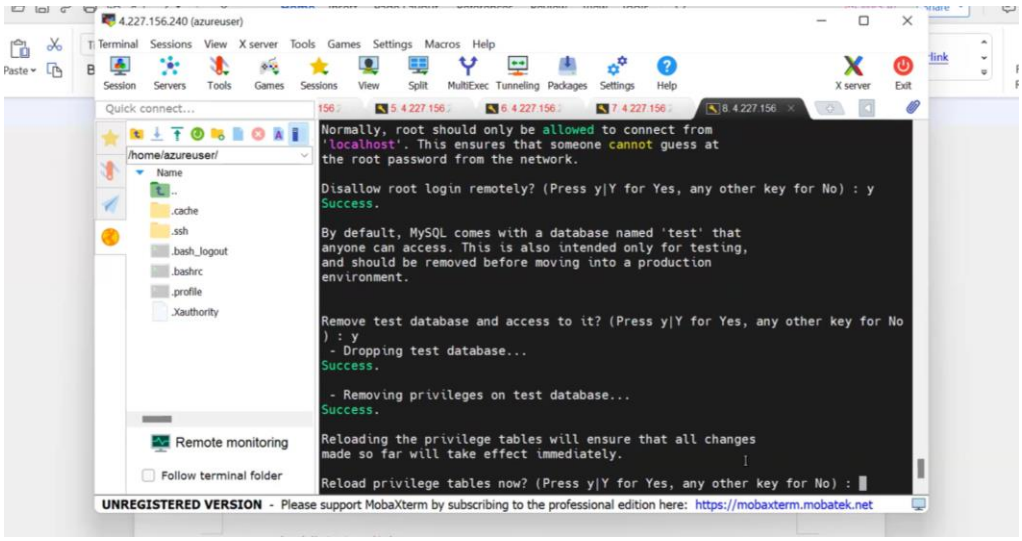
ALTER USER 'root'@'localhost' IDENTIFIED WITH auth_socket;

This will mean that you can once again connect to MySQL as your root **user using the sudo mysql command.**

2. **Secure MySQL Installation**:

sudo mysql_secure_installation

Follow the prompts to set a strong root password and remove unnecessary defaults. As recommended by PCI-DSS V4.0



Step 4: Configure MySQL for PCI-DSS Compliance

1. **Edit MySQL Configuration**:

sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf

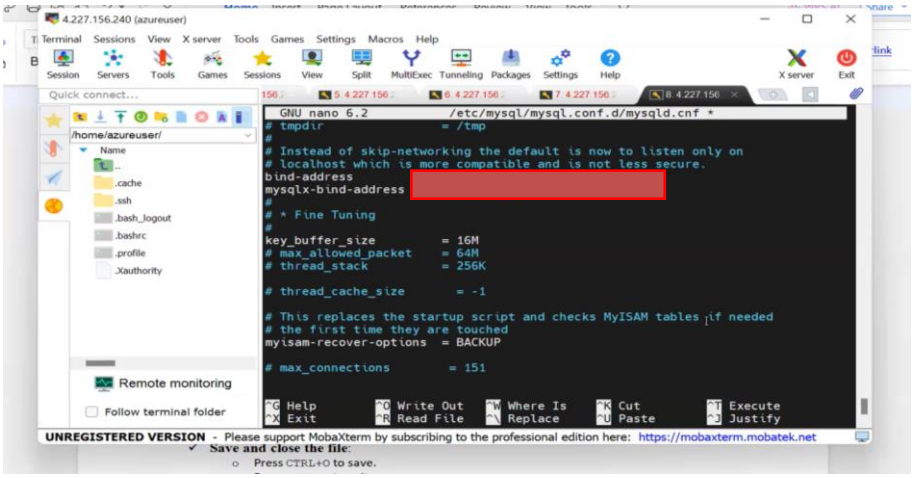
✓ **Find the bind-address directive**: Locate the loop back IP address:

bind-address = 127.0.0.1

✓ **Change the bind-address directive**: Modify the line to bind MySQL to a specific IP address (VM Public IP Address):

bind-address = [redacted]

- ✓ **Save and close the file**:
  - Press CTRL+O to save.
  - Press CTRL+X to exit.



Restart MySQL Service

sudo systemctl restart mysql.service

Verify MySQL Listening IPs and Ports

1. **Use netstat to check listening ports**:

   Sudo apt install net-tools

   sudo netstat -plunt | grep mysqld

Step 5: Create and Manage Users

1. **Login to MySQL**:

   sudo mysql -u root -p

2. **Create Users with Appropriate Privileges**:

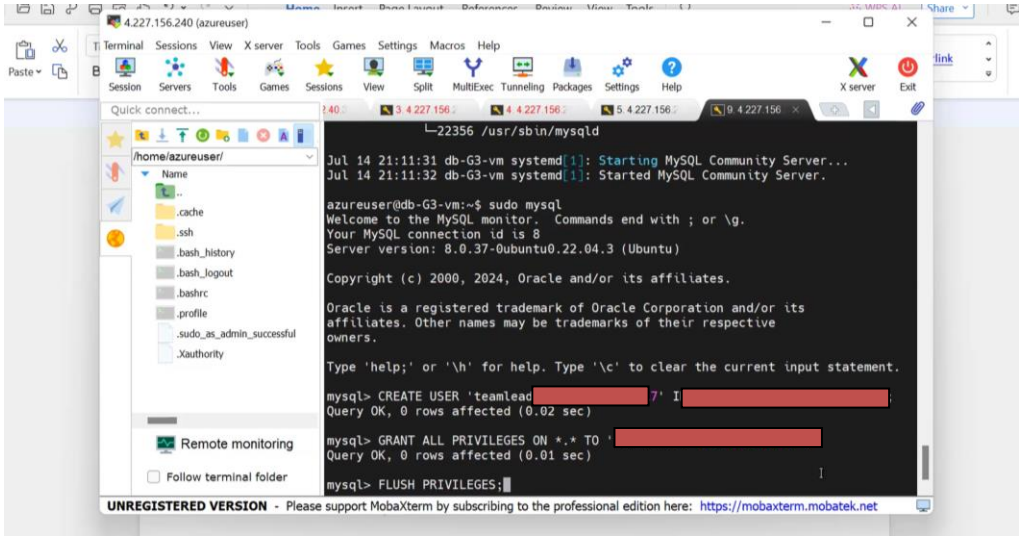
   CREATE USER 'richie'@'[redacted]
   GRANT SELECT, INSER[redacted]

   CREATE USER 'Prettyp[redacted]!';
   GRANT SELECT ON p[redacted]

   CREATE USER 'ju[redacted];
   GRANT ALL ON [redacted]

3. **Flush Privileges**:

   FLUSH PRIVILEGES;



Step 6: Enable Logging and Monitoring

Configure MySQL Logging:

Navigate to MYSQL Configuration file: ***sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf***

Enable Logging;
general_log = 1
general_log_file = /var/log/mysql/mysql.log
log_error = /var/log/mysql/error.log

Restart MySQL: sudo systemctl restart mysql.servic

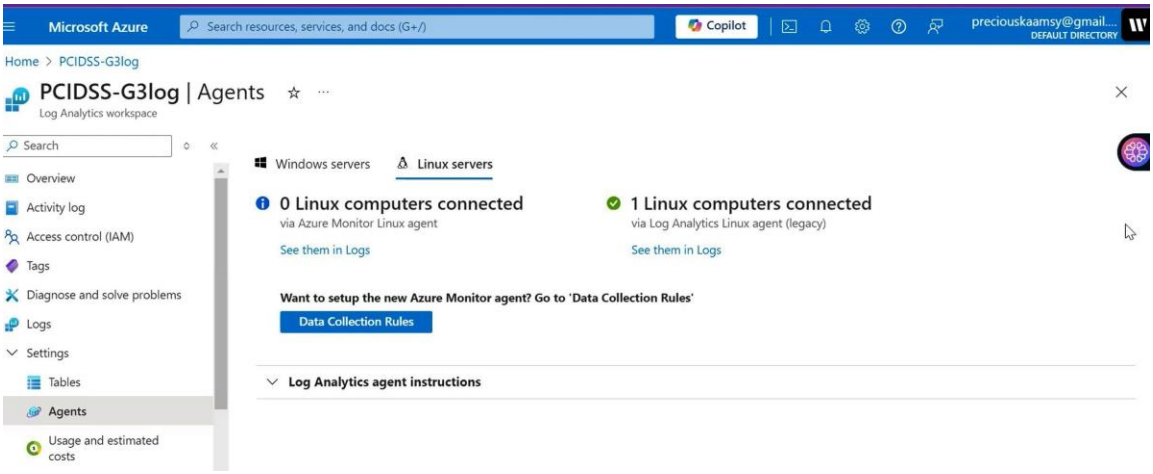
Create a Log Analytics Workspace

1. **Create a Log Analytics Workspace**:
   - In the left-hand menu, select **create a resource**.
   - Search for **Log Analytics Workspace** and select it.
   - Click **Create**.
   - Fill in the required details:
     - **Resource Group**: Select an existing one or create a new one.
     - **Name**: Provide a name for your Log Analytics Workspace.
     - **Region**: Select the region where you want to deploy the workspace.
   - Click **Review + create** and then **Create**.

Install the Log Analytics Agent on the MySQL VM

1. **Navigate to the VM**:
   - In the Azure Portal, go to **Virtual machines**.
   - Select your MySQL VM.
2. **Install the Log Analytics Agent**:
   - In the VM blade, select **Extensions + applications** under **Settings**.
   - Click **+ Add** and choose **Log Analytics agent**.
   - Select the **Log Analytics workspace** you created earlier.

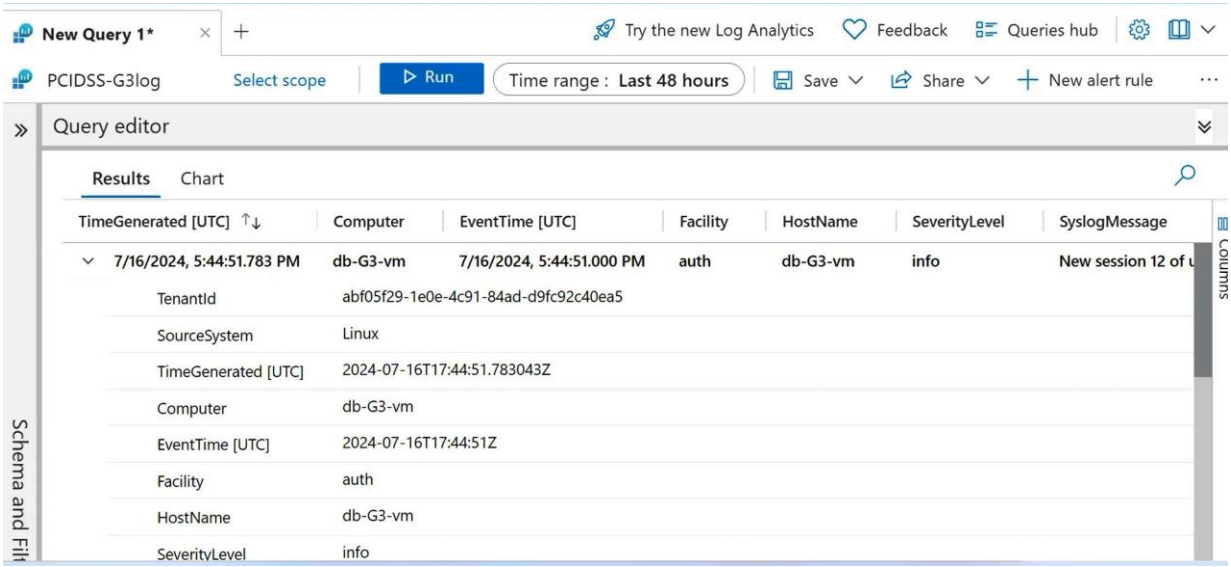
Click **Review + create** and then **Create** to install the agent



Verify Data Collection in Log Analytics Workspace

1. **Navigate to Log Analytics**:
    - In the Azure Portal, go to **Log Analytics workspaces**.
    - Select your workspace.
2. **Run a Log Query**:
    - In the workspace blade, select **Logs**.
    - Run a query to verify that MySQL logs are being collected. For example:

```
Syslog
| where TimeGenerated == 48Hrs
| limit 100
```



Step 8: Test MySQL Installation Using MySQL Workbench

1: Download and Install MySQL Workbench

1. **Download MySQL Workbench**:
   - Go to the MySQL Workbench download page.
2. **Install MySQL Workbench**:
   - Run the downloaded installer and follow the installation prompts.

2: Create a New Connection

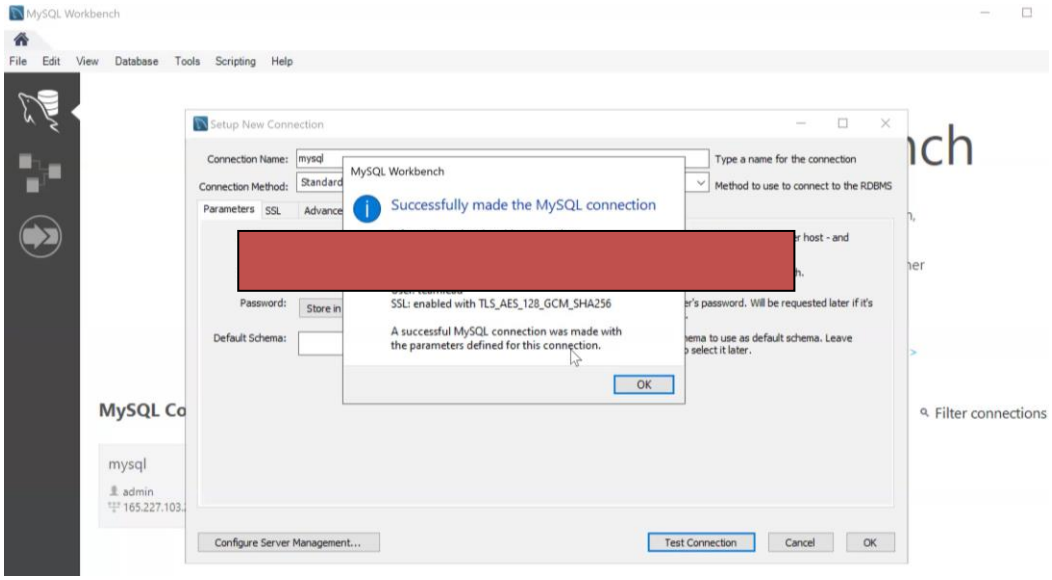
1. **Click on the + icon** in the MySQL Workbench home screen to create a new connection.
2. **Fill in the connection details**:
   - **Connection Name**: Enter a name for your connection (e.g., MySQL_DBG3).
   - **Connection Method**: Select Standard (TCP/IP).
   - **Hostname**: Enter the IP address of your MySQL server ([redacted]
   - **Port**: Enter the MySQL port (default is 3306).
   - **Username**: Enter your MySQL username (e.g., admin).
   - **Password**: Click on Store in Vault to enter and save your password.

3: Test the Connection

1. **Click on Test Connection**:
   - MySQL Workbench will attempt to connect to your MySQL server using the provided details.
2. **Verify the Connection**:
   - If the connection is successful, you will see a success message.
   - If there is an error, check the error message and verify your connection details, IP address, port, username, and password.

5: Save the Connection

1. **Click on OK** to save the connection if the test was successful.
2. **Your new connection** will now appear in the MySQL Workbench home screen under MySQL Connections.



6: Connect to MySQL Server

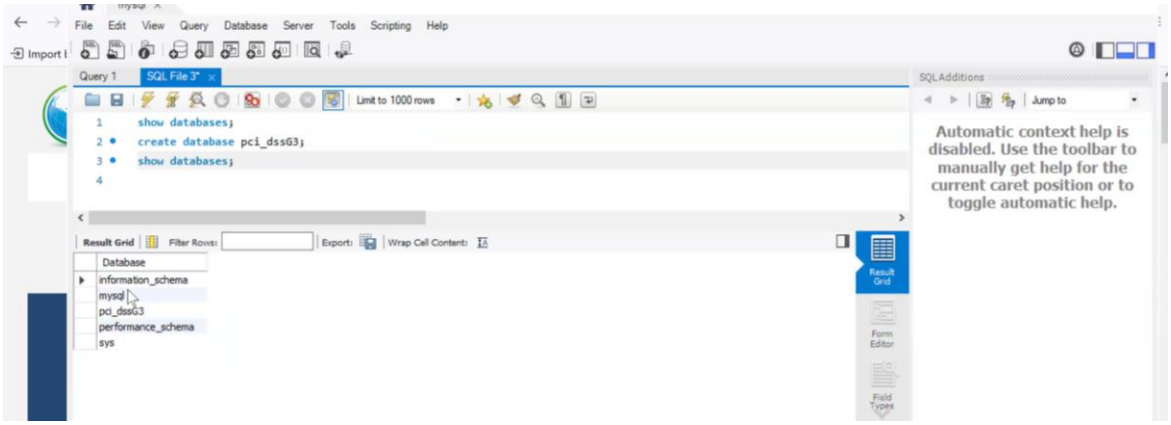
1. **Double-click on the saved connection**:
   - This will open a new SQL editor tab connected to your MySQL server.
2. **Verify the Connection**:
   - You should see the SQL editor where you can execute SQL queries.

7: Execute a Simple Query

1. **Execute a simple query** to verify the connection and database functionality:
   - In the SQL editor, type the following query:

SHOW DATABASES;

2. **Verify the Output**:
   - You should see a list of databases on your MySQL server in the results grid.



Step 7: Regular Maintenance and Updates

1. **Apply Regular Updates**:
   - Regularly update MySQL and the underlying operating system to the latest security patches.

sudo apt update && sudo apt upgrade -y
sudo systemctl restart mysql

2. **Backup and Disaster Recovery**:
   - Implement regular backups using Azure Backup.
   - Ensure backups are encrypted and stored securely.

Conclusion

This project not only strengthens the security posture of the MySQL deployment but also ensures that it aligns with industry best practices and regulatory requirements. The use of Azure's robust monitoring and security tools further enhances the ability to maintain a secure and compliant environment.

References:

https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-20-04

https://docs.microsoft.com/en-us/azure/azure-monitor/vm/monitor-virtual-machine

https://docs.microsoft.com/en-us/azure/azure-monitor/agents/log-analytics-agent

https://www.varonis.com/blog/pci-dss-requirements

https://www.auditboard.com/blog/pci-dss-requirements/

https://blog.rsisecurity.com/?s=pci+dss

https://docs-prv.pcisecuritystandards.org/PCI%20DSS/Standard/PCI-DSS-v4_0_1.pdf

https://learn.microsoft.com/en-us/azure/azure-monitor/agents/log-analytics-agent