

SQL Moderation Hack

Secure Your Data with Azure SQL DB Labs Step-by-step

V3

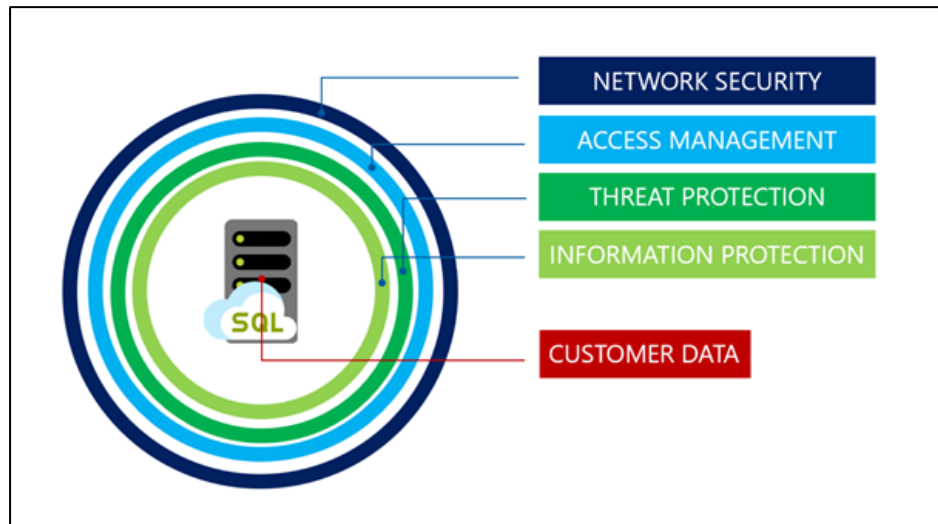
Table of Contents

Introduction	2
Azure SQL Database & Team VM Login Details	3
LAB 1: Auditing for Azure SQL Managed Instance	4
Auditing.....	4
Confirm Auditing is enabled at the Logical SQL Server Level	5
Create an Audit Specification for tracking queries against specific tables	7
LAB 2: Data Discovery & Classification.....	8
Data Discovery & Classification.....	8
Viewing Data Classification Recommendations.....	9
Query the Audit Log Directly using TSQL	16
LAB 3 Part 1: Azure Defender for SQL – Vulnerability Assessment	17
Vulnerability Assessment.....	17
LAB 3 Part 2: Azure Defender for SQL – Advanced Threat Protection.....	28
Advanced Threat Protection	30
LAB 4: Information Protection using Dynamic Data Masking.....	36
LAB 5: End-to-end encryption using Always Encrypted.....	40
Optional Lab – Use Randomized Encryption with Always Encrypted	49

Introduction

This hands-on lab will introduce you to the layered security model available when running databases in Azure. The activities within this hands-on lab will progress from the outer security layers that protect the perimeter of Azure SQL through to the inner layers that protect the information contained within the data.

Because SQL Managed Instance always runs in a private network the Network Security layer has already been implemented at the vNet level. Equally we have already defined and implemented Azure AD and SQL Server logins, roles and permissions so the Access Management tier has also been pre-built.



So this lab will focus on the Threat Protection, Information Protection and Customer Data layers of the security model and how these are implemented in Azure SQL Managed Instance through:

- Review and configuring auditing within Azure SQL Managed Instance
- Using Data Discovery & Classification
- Azure Defender for SQL
 - Vulnerability Assessment
 - Advanced Threat Protection
- Information protection & encryption
 - Dynamic Data Masking
 - Always Encrypted

Azure SQL Database & Team VM Login Details

All the labs run against the TEAMXX_TenantDataDb that you migrated earlier using either SQL Server Management Studio or the Azure Portal.

Your Win10 VM (vm-TEAMXX) login credentials are also a member of SQL Server sysadmin role.

Username	localhost\DemoUser
Password	Demo@pass1234567

The Azure Portal credentials are those that your proctor will supply.

LAB 1: Auditing for Azure SQL Managed Instance

Auditing

For Azure SQL Managed Instance auditing is enabled at the server level and tracks events at both the server and database level (depending on your audit configuration).

The events are then written to a centralized log stored outside of the Managed Instance environment.

The log can be stored in either:

- A file in Azure Storage Account
- Log Analytics Workspace (a special centralized log storage location for logs from all Azure services)
- Azure Event Hub (an Azure native message queue where streaming messages can be consumed in real-time)
- Or any combination of the 3

More details on auditing in SQL Managed Instance can be found here:

[SQL Managed Instance auditing - Azure SQL Managed Instance | Microsoft Docs](#)

(PTO)

SQL Modernisation Open Hack

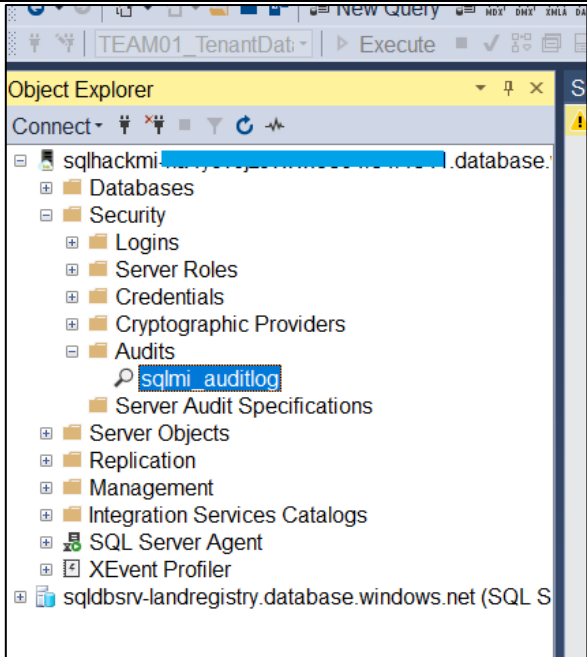
Confirm Auditing is enabled at the Logical SQL Server Level

Because auditing is switched on at the server level we have already done this within the lab environment and configured it to write to a log file held in an Azure Storage account.

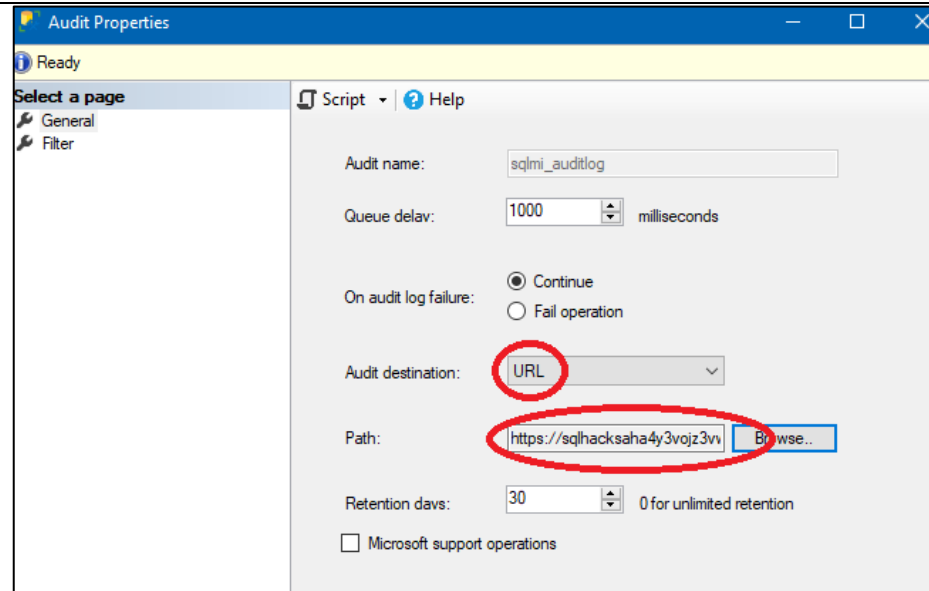
For more information on setting up auditing on SQL Managed Instance see this documentation:

[SQL Managed Instance auditing - Azure SQL Managed Instance | Microsoft Docs](#)

But for now let's confirm that auditing is enabled.

Narrative	Screenshot/Code	Notes
<p>1. On your team Win10 VM open SQL Server Management Studio, connect to the shared SQL Managed Instance and expand the Security\Audits folder</p>	 A screenshot of the SQL Server Enterprise Manager interface. The 'Object Explorer' pane on the left shows the hierarchy of the SQL Managed Instance. The 'Security' folder is expanded, and the 'Audits' folder is also expanded. Under 'Audits', the 'sqlmi_auditlog' folder is highlighted with a blue selection box. The top of the window shows the 'Connect' bar with the instance name 'TEAM01_TenantData' and the 'Execute' button.	

2. Right-click on the **sqlmi_auditlog** and look at it's properties. Notice that the Audit Destination is set to URL and the Path points to our shared Storage Account.



(PTO)

[Create an Audit Specification for tracking queries against specific tables](#)

Although a physical Audit Log has been created and enabled, to actually capture events we need to create Audit Specifications.

Audit Specifications define what actions and operations are audited and at what level. It is quite normal to have separate Audit Specifications for each database as well as at the server level depending on what activities you want to track. All these specifications will write to the same Audit Log.

Narrative	Screenshot/Code	Notes
1. On your team Win10 VM open SQL Server Management Studio, connect to the shared SQL Managed Instance and open a new query window to your TEAMXX_TenantDataDb		
2. Run this query to create an Audit Spec that will monitor all SELECT queries run against all tables in the SalesLT schema:	<pre>--RUN AGAINST YOUR TEAM'S [TenantDataDb]: USE [TEAMXX_TenantDataDb]; CREATE DATABASE AUDIT SPECIFICATION audit_sensitive_data FOR SERVER AUDIT [sqlmi_auditlog] ADD (SELECT ON Schema::SalesLT BY public) WITH (STATE = ON)</pre>	For more information on Audit Specification see this documentation: Create Server Audit & Server Audit Specification - SQL Server Microsoft Docs

We'll return to the Audit Logs later to see what it has captured for us.

LAB 2: Data Discovery & Classification

Data Discovery & Classification

Data Discovery & Classification is a built-in capability for discovering, classifying, labelling and protecting sensitive data in databases. It can be used to support many use cases including financial, healthcare, personally identifiable (PII) data and help meet data privacy standards and regulatory compliance.

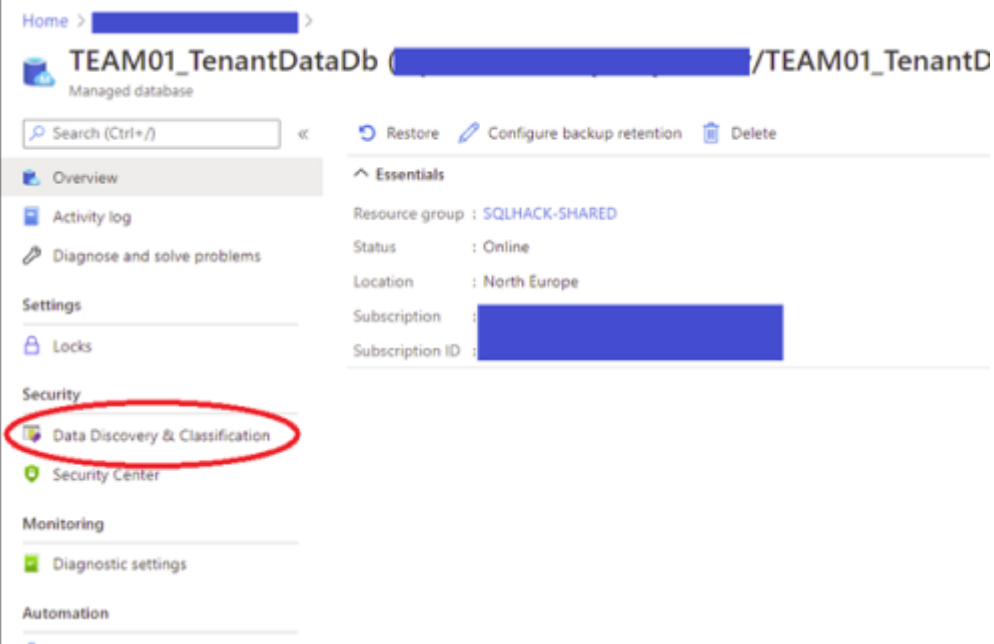
More information on Data Discovery & Classification can be found here:

<https://docs.microsoft.com/en-us/azure/azure-sql/database/data-discovery-and-classification-overview>

(PTO)

Viewing Data Classification Recommendations

Whenever a database is deployed or schema changes are made to an existing database, the Data Discovery & Classification engine automatically performs a scan to identify columns that may potentially contain sensitive data.

Narrative	Screenshot/Code	Notes
<p>1. Within the Azure Portal navigate to the shared Azure SQL Managed Instance screen. Scroll down to the list of databases and click on your teams TEAMXX_TenantDataDb database.</p>		
<p>2. On the blade on the left, under the Security section click “Data Discovery & Classification”</p>		

The Data Discovery and Classification **Overview** shows that no data classifications have been made but based on the automatic classification scan there are a number of potential data classification recommendations as shown at the top of the report:

3. Click the blue information bar (highlighted in yellow) to view the data classification recommendations

The screenshot displays the Microsoft Azure portal interface for the 'SQLDB-team-01 (azuresqlserver-team-01/SQLDB-team-01)' resource. The left-hand navigation pane lists various services, with 'Data Discovery & Classification' selected. The main content area shows the 'Overview' tab, which includes a summary of data classification results. A blue information bar at the top states: 'We have found 15 columns with classification recommendations'. Below this, the 'Overview' section displays three key metrics: 'Classified columns' (0), 'Tables containing sensitive data' (0), and 'Unique information types' (0). Two donut charts are also present: 'Label distribution' and 'Information type distribution', both showing 0 columns.

Microsoft Azure Search resources, services, and docs (G+)

Home > sql-hack-team-01 > SQLDB-team-01 (azuresqlserver-team-01/SQLDB-team-01)

SQLDB-team-01 (azuresqlserver-team-01/SQLDB-team-01) | Data Discovery & Classification

SQL database

Search (Ctrl+ /)

Export Configure Feedback

We have found 15 columns with classification recommendations →

SQL Data Classification provides manual labeling via T-SQL commands. For advanced classification capabilities, use [Azure Purview](#).

Overview Classification

Classified columns 0

Tables containing sensitive data 0

Unique information types 0

Label distribution

Information type distribution

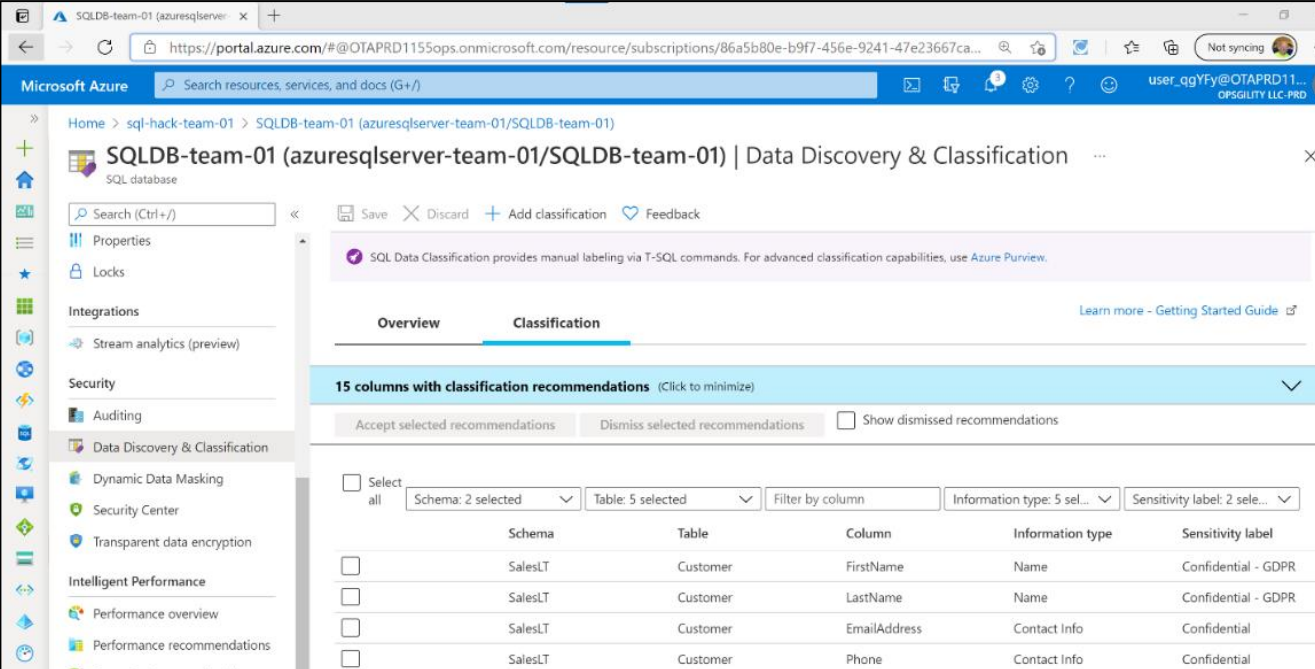
0 COLUMNS

0 COLUMNS

SQL Modernisation Open Hack

The recommendations show the name of the schema, table and column with intelligent information type classification and sensitivity recommendations.

As can be seen the **Customer** table in the **SalesLT** schema contains the columns **FirstName** and **LastName**. The initial data classification scan has identified that the **Information type** of these columns from a data classification perspective is **Name** and the **Sensitivity Label** for these columns is recommended to be **Confidential – GDPR**.



Microsoft Azure portal interface showing the Data Discovery & Classification section for a SQL database. The interface displays 15 columns with classification recommendations. The 'Classification' tab is active, showing a table with columns: Schema, Table, Column, Information type, and Sensitivity label. The table lists four columns from the SalesLT schema: FirstName, LastName, EmailAddress, and Phone. The first two are classified as 'Name' with a 'Confidential - GDPR' sensitivity label, while the other two are classified as 'Contact Info' with a 'Confidential' sensitivity label.

	Schema	Table	Column	Information type	Sensitivity label
<input type="checkbox"/>	SalesLT	Customer	FirstName	Name	Confidential - GDPR
<input type="checkbox"/>	SalesLT	Customer	LastName	Name	Confidential - GDPR
<input type="checkbox"/>	SalesLT	Customer	EmailAddress	Contact Info	Confidential
<input type="checkbox"/>	SalesLT	Customer	Phone	Contact Info	Confidential

4. Select the **FirstName** and **LastName** classification recommendations by selecting the recommendation rows, click **Accept selected recommendations** and then click **Save**.

5. Click the **Overview** tab on the Data Discovery & Classification report to look at the saved data classifications.





There are now two columns classified from the Customer table with the information type of **Name** and the sensitivity label **Confidential – GDPR**.


The screenshot shows the Microsoft Azure portal interface for the 'SQLDB-team-01' resource. The 'Data Discovery & Classification' section is open, and the 'Classification' tab is selected. A banner indicates that SQL Data Classification provides manual labeling via T-SQL commands. Below this, the 'Overview' and 'Classification' tabs are visible, with 'Classification' being the active tab. A summary bar shows '15 columns with classification recommendations'. Below this, there are buttons for 'Accept selected recommendations', 'Dismiss selected recommendations', and a checkbox for 'Show dismissed recommendations'. A table lists the columns with their classification details:

Select	Schema	Table	Column	Information type	Sensitivity label
<input checked="" type="checkbox"/>	SalesLT	Customer	FirstName	Name	Confidential - GDPR
<input checked="" type="checkbox"/>	SalesLT	Customer	LastName	Name	Confidential - GDPR
<input type="checkbox"/>	SalesLT	Customer	EmailAddress	Contact Info	Confidential

Now let's add a custom data classification which is not based on the auto recommendations.

6. Switch back to the **Classification** tab at the top of the report click **" + Add classification "**.

 Save  Discard  Add classification  Feedback

 SQL Data Classification provides manual labeling via T-SQL commands. For advanced classification capabilities, use [Azure Purview](#).

Overview

Classification

2 classified columns

Schema: 1 selected

Table: 1 selected

Filter by column

Schema	Table	Column
SalesLT	Customer	LastName
SalesLT	Customer	FirstName

7. On the **Add Classification** blade on the far right of the screen set the following values and then click **Add Classification** and then **Save** to save your new classification.

8. Click the **Overview** tab to look at the saved data classifications.

Schema name:	SalesLT
Table name:	Product
Column name:	ListPrice
Information type:	Financial
Sensitivity Label:	Highly Confidential
<i>Click</i>	Add Classification
<i>Click</i>	Save

Add classification ×

Schema name *

SalesLT ▼

Table name *

Product ▼

Column name *

ListPrice (money) ▼

Information type

Financial ▼

Sensitivity label

Highly Confidential ▼

Add classification

Cancel

<p>9. Open SQL Server Management Studio, connect to the shared SQL Managed Instance and open a new TSQL query window connected to your TEAMXX_TenantDataDb database</p> <p>10. Run the SELECT statements opposite against your TEAMXX_TenantDataDb database.</p>	<pre>-- 1 Data Discovery & Classification SELECT c.FirstName ,c.LastName ,c.* FROM SalesLT.Customer c; SELECT p.ListPrice FROM SalesLT.Product p;</pre>	
<p>Nothing out of the ordinary happens - two simple result sets should be returned containing the FirstName, LastName and ListPrice columns.</p>	<p><i>REMEMBER: Data Discovery and Classification is not a security mechanism – it's a data tagging and management tool.</i></p>	
<p>Now let's see how classifying columns is actually very useful when used in conjunction with the SQL Auditing that we configured earlier.</p>		

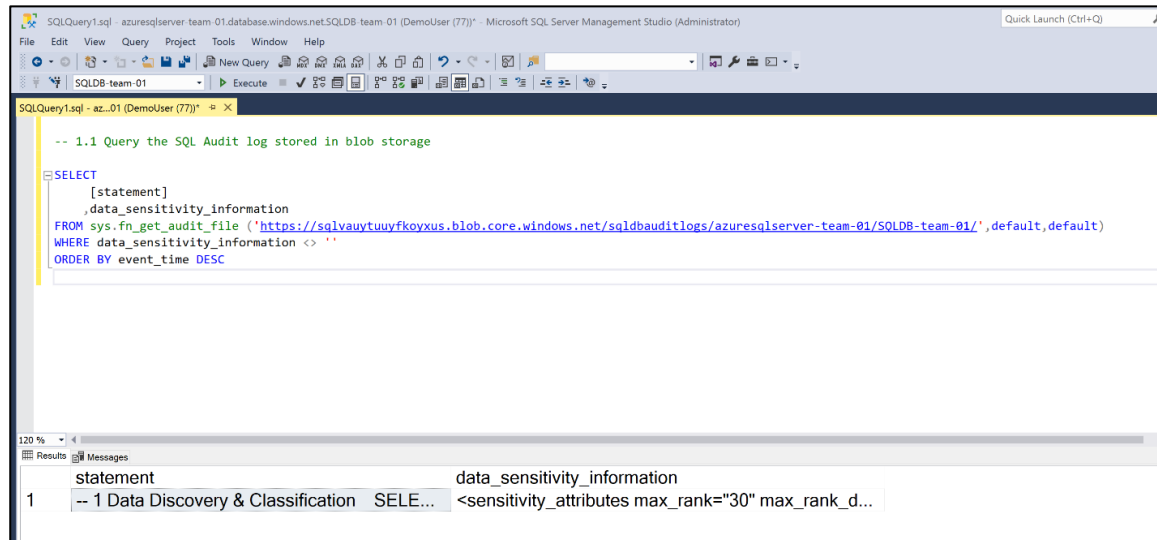
(PTO)

Query the Audit Log Directly using TSQL

Although we can view audit logs though Management Studio, it is also possible to query them directly using TSQL and a system function called `sys.fn_get_audit_file` to see what has been captured.

Narrative	Screenshot/Code	Notes
<p>1. Open a new query window in SQL Server Management Studio on the team VM. Copy the SQL statement opposite and paste it into the query window.</p>	<div data-bbox="658 368 2029 528" style="background-color: yellow; border: 1px solid black; padding: 10px; text-align: center;"> <p>NOTE - Before running the query change the Storage Account details to the URL for the sqlmi_auditlog folder in the shared storage account. Also change the database name in the WHERE clause.</p> </div> <pre data-bbox="658 600 1881 767"> --Query the SQL Audit log stored in blob storage SELECT * FROM sys.fn_get_audit_file ('<<URL FOR THE STORAGE ACCOUNT auditlogs...sqlmi_auditlog FOLDER>>/<<sqlmi- name>>',default,default) WHERE database_name = 'TEAMxx_TenantDataDb' </pre> <p data-bbox="658 810 1267 839">Example (with URL and MI name, change to yours):</p> <pre data-bbox="658 842 1697 962"> SELECT * FROM sys.fn_get_audit_file ('https://sqlhacksa7crx7fckhvzmu.blob.core.windows.net/auditlogs/sqlhackmi- 7crx7fckhvzmu',default,default) WHERE database_name = 'TEAM01_TenantDataDb' </pre>	

2. Once the query has been modified and executed, review the columns:
- statement
 - data_sensitivity_information



The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The top pane displays a SQL query that retrieves audit log data from a blob storage location. The query is as follows:

```
-- 1.1 Query the SQL Audit log stored in blob storage
SELECT
    [statement]
    ,data_sensitivity_information
FROM sys.fn_get_audit_file ('https://sqlvaudytyuufkoyxus.blob.core.windows.net/sqldbauditlogs/azuresqlserver-team-01/SOLDB-team-01/', default, default)
WHERE data_sensitivity_information <> ''
ORDER BY event_time DESC
```

The bottom pane shows the results of the query execution. The results are displayed in a table with two columns: 'statement' and 'data_sensitivity_information'. The first row of results is:

	statement	data_sensitivity_information
1	-- 1 Data Discovery & Classification SELE...	<sensitivity_attributes max_rank="30" max_rank_d...

LAB 3 Part 1: Azure Defender for SQL – Vulnerability Assessment

When provisioning an Azure SQL Managed Instance or an Azure SQL Database logical server there is the option to enable the security feature Azure Defender for SQL.

This security feature offers two security components:

- Vulnerability Assessments
- Advanced Threat Protection

This first part of the lab will focus on Vulnerability Assessments, Part 2 will deal with Advanced Threat Protection.

Vulnerability Assessment

A Vulnerability Assessment is an output position (or report) from a vulnerability scan.

SQL Modernisation Open Hack

A Vulnerability Assessment scan is the application of SQL Server best practices based on a rules engine, the goal being to improve the security posture of your Azure SQL Managed Instance or Azure SQL Database. The first scan will produce the initial vulnerability scan baseline. The first scan happens automatically once a database is deployed.

More details on Azure SQL vulnerability assessments can be found here:

<https://docs.microsoft.com/en-us/azure/azure-sql/database/sql-vulnerability-assessment>

(PTO)

3. In the **TEAMXX_TenantDataDB** database screen. On the left hand blade click **Microsoft Defender for Cloud** in the Setting section
4. If you see the Defender for Cloud upgrade screen, click “Skip” or “Remind me later” or click on the breadcrumb to go one level back

5. Scroll down the “Security” screen to the bottom and click the “**View additional recommendations in Vulnerability Assessment >**” link

managed resources

Search (Ctrl+ /)

Overview

Activity log

Diagnose and solve problems

Settings

Locks

Security

Data Discovery & Classification

Microsoft Defender for Cloud

Monitoring

Diagnostic settings

Automation

Tasks (preview)

Export template

Support + troubleshooting

Resource health

New Support Request

Recommendations

Defender for Cloud continuously monitors the configuration of your SQL Servers to identify potential security vulnerabilities and recommends actions to mitigate the

✓
✓
✓

No recommendations to display

There are no security recommendations for this resource

[View all recommendations in Defender for Cloud](#)

Security incidents and alerts

Defender for Cloud uses advanced analytics and global threat intelligence to alert you to malicious activity. Alerts displayed below are from the past 21 days.

[Check for alerts on this resource in Microsoft Defender for Cloud >](#)

Vulnerability assessment findings

ID	Security Check	↑↓	A
VA2129	Changes to signed modules should be authorized		T
VA2108	Minimal set of principals should be members of fixed high impact database roles		T
VA1102	The Trustworthy bit should be disabled on all databases except MSDB		T
VA1244	Orphaned users should be removed from SQL server databases		T

[View additional findings in Vulnerability Assessment >](#)

The “Vulnerability Assessment” page can be used to run a scan, view scan history and will show the number of checks that have been passed and failed for the last scan with failed checks listed in the table below.

- Run a scan if prompted to do so which should only take a few minutes.

Vulnerability Assessment
(azuresqlserver-team-01/SQLDB-team-01)

Scan | Export Scan Results | Scan History | Feedback

SQL Vulnerability Assessment rules have been updated. This may impact your scan results. [Learn more](#) →

Total failing checks: **5** (red X) | Total passing checks: **29** (green checkmark)

Risk summary:
 High Risk: 2 (red bar)
 Medium: 2 (orange bar)
 Low Risk: 1 (blue bar)

Last scan time: Tue, 06 Apr 2021 19:42:02 UTC

[Learn more](#) | [SQL Vulnerability Assessment](#)

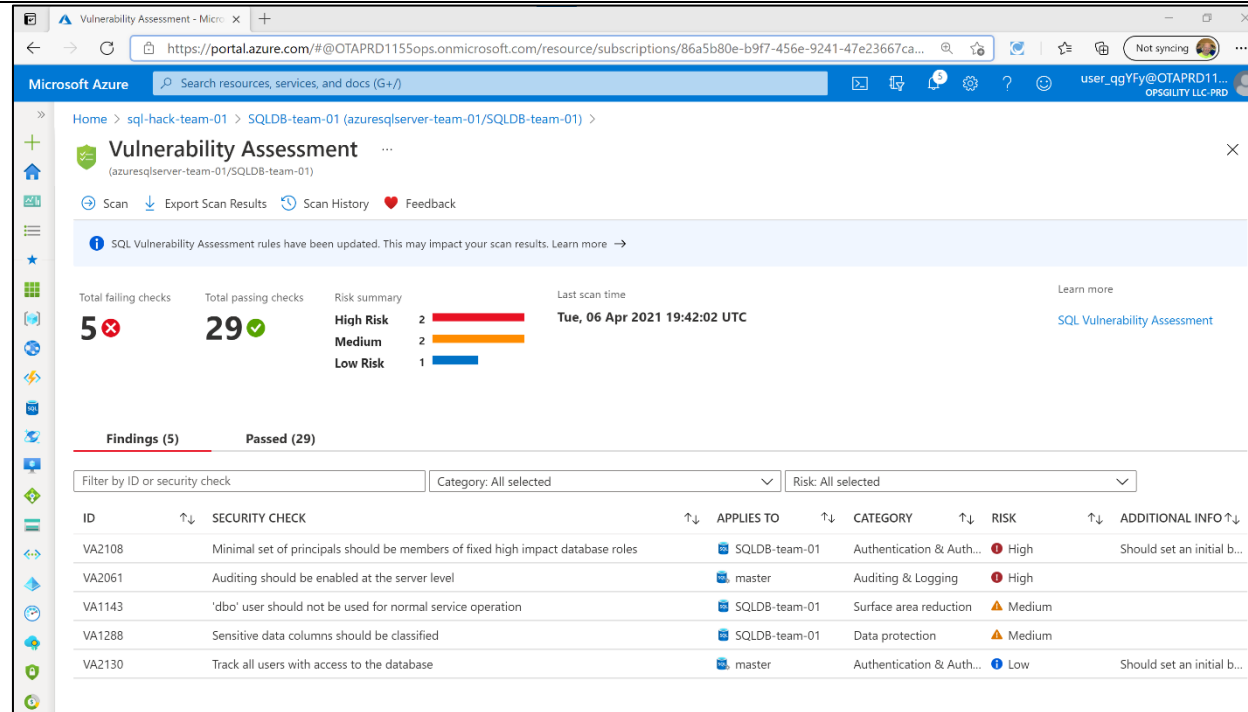
Findings (5) | **Passed (29)**

Filter by ID or security check | Category: All selected | Risk: All selected

ID	SECURITY CHECK	APPLIES TO	CATEGORY	RISK	ADDITIONAL INFO
VA2108	Minimal set of principals should be members of fixed high impact database roles	SQLDB-team-01	Authentication & Auth...	High	Should set an initial b...
VA2061	Auditing should be enabled at the server level	master	Auditing & Logging	High	
VA1143	'dbo' user should not be used for normal service operation	SQLDB-team-01	Surface area reduction	Medium	
VA1288	Sensitive data columns should be classified	SQLDB-team-01	Data protection	Medium	
VA2130	Track all users with access to the database	master	Authentication & Auth...	Low	Should set an initial b...

SQL Modernisation Open Hack

7. Review the lists of passed and failed checks. Notice that the report is specific to database you ran the scan for but does also include events against the system database and therefore flag server configuration issues.



Vulnerability Assessment
(azuresqlserver-team-01/SQLDB-team-01)

Scan | Export Scan Results | Scan History | Feedback

SQL Vulnerability Assessment rules have been updated. This may impact your scan results. [Learn more](#) →

Total failing checks: **5** (with red X icon) | Total passing checks: **29** (with green checkmark icon)

Risk summary:
 High Risk: 2 (red bar)
 Medium: 2 (orange bar)
 Low Risk: 1 (blue bar)

Last scan time: Tue, 06 Apr 2021 19:42:02 UTC

[Learn more](#) | [SQL Vulnerability Assessment](#)

Findings (5) | **Passed (29)**

Filter by ID or security check | Category: All selected | Risk: All selected

ID	SECURITY CHECK	APPLIES TO	CATEGORY	RISK	ADDITIONAL INFO
VA2108	Minimal set of principals should be members of fixed high impact database roles	SQLDB-team-01	Authentication & Auth...	High	Should set an initial b...
VA2061	Auditing should be enabled at the server level	master	Auditing & Logging	High	
VA1143	'dbo' user should not be used for normal service operation	SQLDB-team-01	Surface area reduction	Medium	
VA1288	Sensitive data columns should be classified	SQLDB-team-01	Data protection	Medium	
VA2130	Track all users with access to the database	master	Authentication & Auth...	Low	Should set an initial b...

8. In the **Findings** tab, which lists the failed checks, click on finding:

ID	Security Check
VA1256	User CLR assemblies should not be defined in the database

9. Note the detailed report lists the rule's details, the offending CLRAs and a remediation script to remove them.


However, because these 2 CLRAs are an integral part of our migrated legacy application we need to keep them.

But equally we don't want them to be continuously flagged as an issue in the Vulnerability Assessment reports. To do this we can add exceptions to the Vulnerability Assessment's "baseline" position.

VA1256 - User CLR assemblies should not be defined in the database ...

[✓ Approve as Baseline](#)
[✗ Clear Baseline](#)

STATUS

 **FAIL**

APPLIES TO

TEAM20_TenantDataDb

DESCRIPTION

CLR assemblies can be used to execute arbitrary code on SQL Server process. This rule checks that there are no user-defined CLR assemblies in the database

IMPACT

Using CLR assemblies can bring a security flaw to the SQL Server instance and to all other network resources accessible from it

BENCHMARK REFERENCES

- FedRAMP

RULE QUERY

```
SELECT name AS [Assembly] FROM sys.assemblies WHERE is_user_defined != 0
```

MICROSOFT RECOMMENDATION

Empty Set

RESULTS

In Baseline	Assembly
✗	CLRUFDS
✗	Database1

REMEDIATION

Drop assemblies from the affected databases

```
DROP ASSEMBLY [CLRUFDS]
DROP ASSEMBLY [Database1]
```


10. On the details page for V1256 click **Add all results as baseline** and select **Yes** in the warning message.

Approving as the baseline will update the Vulnerability Assessment rules engine to accept the current CLR Assemblies as allowable and set a new baseline position for the rule.

[Home](#) > [TEAM01_TenantDataDb \(sqlhackmi-7crx7fckhvmu/TEAM01_TenantDataDb\)](#) | [Microsoft Defender for Cloud](#) > [TEAM01_TenantDataDb \(sqlh](#)

VA1256 - User CLR assemblies should not be defined in the database ...

Severity
High

Status
Unhealthy

Scan time
10/18/2022

Description

CLR assemblies can be used to execute arbitrary code on SQL Server process. This rule checks that there are no user-defined CLR assemblies in the database

Impact

Using CLR assemblies can bring a security flaw to the SQL Server instance and to all other network resources accessible from it

Benchmark

- FedRAMP

Remediation

Drop assemblies from the affected databases

- `DROP ASSEMBLY [CLRUFDS]`
- `DROP ASSEMBLY [Database1]`

i Exercise standard precautions when using the suggested remediation script on prod

Query and results ⓘ

- `SELECT name AS [Assembly] FROM sys.assemblies WHERE`

Add all results as baseline

Remove all from baseline

Status	Assembly
Not in Baseline	Database1
Not in Baseline	CLRUFDS

11. Navigate back to the Vulnerability Assessment summary page by clicking the rightmost breadcrumb link at the top of the portal page

Home > TEAM01_TenantDataDb (sqlhackmi-7crx7fckhvzmu/TEAM01_TenantDataDb) | Microsoft Defender for Cloud > TEAM01_TenantDataDb (sqlhackmi-7crx7fckhvzmu/TEAM01_TenantDataDb)

VA1256 - User CLR assemblies should not be defined in the database ...

Severity **High** Status **Unhealthy** Scan time 10/18/2022

Description

CLR assemblies can be used to execute arbitrary code on SQL Server process. This rule checks that there are no user-defined CLR assemblies in the database

Impact

Using CLR assemblies can bring a security flaw to the SQL Server instance and to all other network resources accessible from it

Benchmark

- FedRAMP

Remediation

Drop assemblies from the affected databases

- DROP ASSEMBLY [CLRUFDS]
- DROP ASSEMBLY [Database1]

Exercise standard precautions when using the suggested remediation script on production environments

Query and results

```
1 SELECT name AS [Assembly] FROM sys.assemblies WHERE is_user_defined = 1
```

Add all results as baseline Remove all from baseline

Status	Assembly
Not in Baseline	Database1
Not in Baseline	CLRUFDS

12. Once back at the Vulnerability Assessment summary page there will be a warning that the baseline has been updated and a

Home > sqlhackmi-laalnideoycte > TEAM20_TenantDataDb (sqlhackmi-laalnideoycte/TEAM20_TenantDataDb) >

Vulnerability Assessment ...

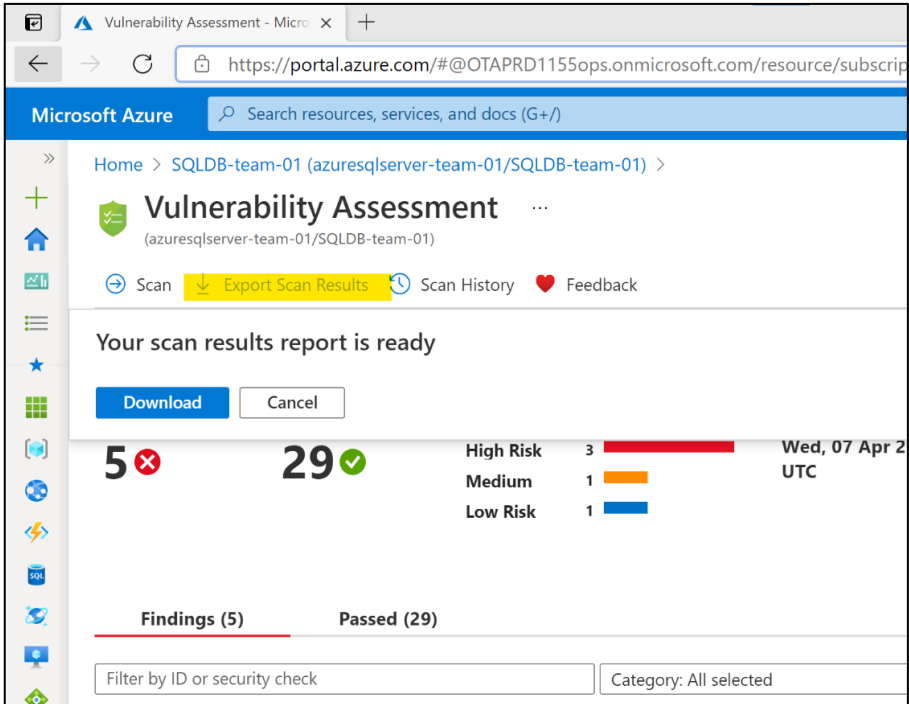
(sqlhackmi-laalnideoycte/TEAM20_TenantDataDb)

Scan Export Scan Results Scan History Feedback

There are pending baseline changes. Run a new scan to see updated results.

Total failing checks **17** Total passing checks **92** Risk summary Last scan time

High Risk	9		Fri, 04 Jun 2021 13:32:18 UTC
Medium	4		
Low Risk	4		

<p>new scan is needed.</p>										
<p>13. Click the Scan button to run a manual scan which will take a about a minute. Once the scan completes the finding VA1256 will be removed from the Findings list.</p> <p>When making changes to a Vulnerability Assessment baseline it may be necessary for compliance reasons to export a Scan Findings report to show the</p>	<div data-bbox="665 633 1570 1339"><p>The screenshot shows the Microsoft Azure Vulnerability Assessment interface. The breadcrumb path is 'Home > SQLDB-team-01 (azuresqlserver-team-01/SQLDB-team-01) >'. The main heading is 'Vulnerability Assessment' with the subtext '(azuresqlserver-team-01/SQLDB-team-01)'. Below this are buttons for 'Scan', 'Export Scan Results' (highlighted in yellow), 'Scan History', and 'Feedback'. A message states 'Your scan results report is ready' with 'Download' and 'Cancel' buttons. The results summary shows 5 findings (5 red X icons) and 29 passed items (29 green checkmark icons). A risk breakdown table is displayed:</p><table><tr><td>High Risk</td><td>3</td><td>Red bar</td></tr><tr><td>Medium</td><td>1</td><td>Orange bar</td></tr><tr><td>Low Risk</td><td>1</td><td>Blue bar</td></tr></table><p>The date and time 'Wed, 07 Apr 2020 12:00:00 UTC' are shown. At the bottom, there are filters for 'Findings (5)' and 'Passed (29)', a search bar 'Filter by ID or security check', and a category dropdown 'Category: All selected'.</p></div> <div data-bbox="1749 646 2002 857"><p>NOTE: Excel is <i>*not*</i> installed on your lab VMs so you will have to copy the report to your own desktop to have a look at it.</p></div>	High Risk	3	Red bar	Medium	1	Orange bar	Low Risk	1	Blue bar
High Risk	3	Red bar								
Medium	1	Orange bar								
Low Risk	1	Blue bar								

<p>security posture of the Azure SQL Database in relation to the amended baseline.</p> <p>To export the results of a scan to reflect the current baseline click “Export Scan Results” at the top of the portal screen:</p>		
---	--	--

LAB 3 Part 2: Azure Defender for SQL – Advanced Threat Protection

The other security component of Azure Defender for SQL is Advanced Threat Protection.

Advanced Threat Protection provides a layer of security that can detect and respond to potential threats as they occur by providing security alerts on anomalous activities. Alerts can be generated based on suspicious database activities, potential vulnerabilities, and SQL injection attacks, as well as anomalous database access and queries patterns.

More information in Azure Defender for SQL – Advanced Threat Protection can be found here:

<https://docs.microsoft.com/en-us/azure/azure-sql/database/threat-detection-overview>

(PTO)

Advanced Threat Protection

Narrative	Screenshot/Code	Notes
1. In the Azure portal navigate to the shared SQL Managed Instance.		
2. Scroll down the Overview screen until you see the list of databases and click on your TEAMXX_TenantDataDB database.		

3. In the **TEAMXX_TenantDataDB** database screen, on the left-hand blade click **Microsoft Defender for Cloud** in the Security section

4. Scroll down to the **Security incidents and alerts** heading – note no incidents or alerts are listed:

The screenshot shows the Microsoft Defender for Cloud interface for a managed database named **TEAM03_TenantDataDb (sqlhackmi-ha4y3vojz3vww/TEAM03_)**. The left-hand navigation pane lists various sections: Overview, Activity log, Diagnose and solve problems, Settings (with sub-items Locks, Security, and Monitoring), Automation (with sub-items Tasks (preview) and Export template), and Support + troubleshooting (with sub-items Resource health and New Support Request). The **Security** section is currently selected. The main content area displays the heading **Security incidents and alerts** (highlighted in yellow) and a message stating "Defender for Cloud continuously monitors the configuration of your SQL Serv". Below this, there is a link to "Check for alerts on this resource in Microsoft Defender for Cloud >". The **Vulnerability assessment findings** section is also visible, showing a table with columns for ID, Security Check, and a description. The first entry has ID VA2129 and the description "Changes to signed moduli".

TEAM03_TenantDataDb (sqlhackmi-ha4y3vojz3vww/TEAM03_)
Managed database

Search (Ctrl+ /) « Recommendations

Defender for Cloud continuously monitors the configuration of your SQL Serv

Overview
Activity log
Diagnose and solve problems

Settings

Locks

Security

Data Discovery & Classification

Microsoft Defender for Cloud

Monitoring

Dagnostic settings

Automation

Tasks (preview)
Export template

Support + troubleshooting

Resource health
New Support Request

Security incidents and alerts

Defender for Cloud uses advanced analytics and global threat intelligence to :

Check for alerts on this resource in Microsoft Defender for Cloud >

Vulnerability assessment findings

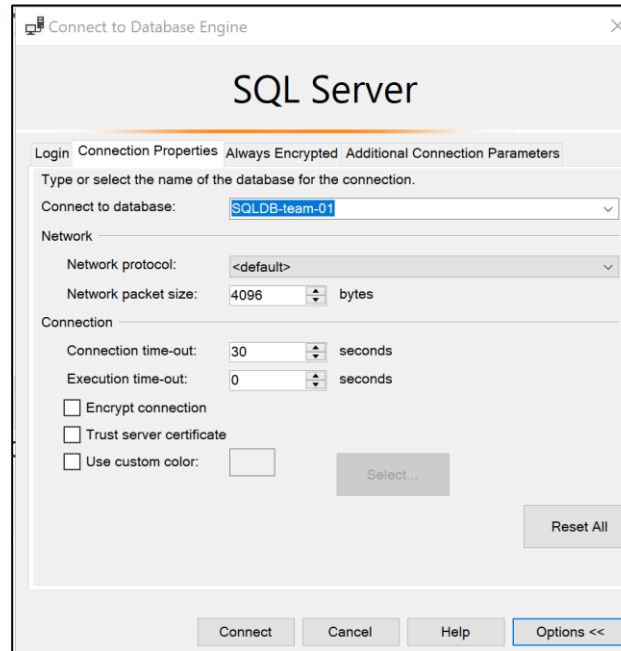
ID	Security Check
VA2129	Changes to signed moduli

<p>5. On the team VM, open a new query window in SQL Server Management Studio connected to your TEAMXX_TenantDataDB database.</p>		
<p>6. To simulate a potential SQL injection query copy the following SELECT into the new query window BUT DON'T RUN IT YET:</p> <p>(PTO)</p>	<pre>--Advanced Threat Protection SELECT * FROM sys.databases WHERE database_id like '' or 1 = 1 -- ' and family = 'test1';</pre>	<p>Notice that the logic in the WHERE clause will always equate to true and the positioning of single-quotes including in the comment represents a potential SQL injection vulnerability</p>

7. Before running the query change the connection properties as show opposite using the **Query\Connection\Change Connection...** menu in SSMS.

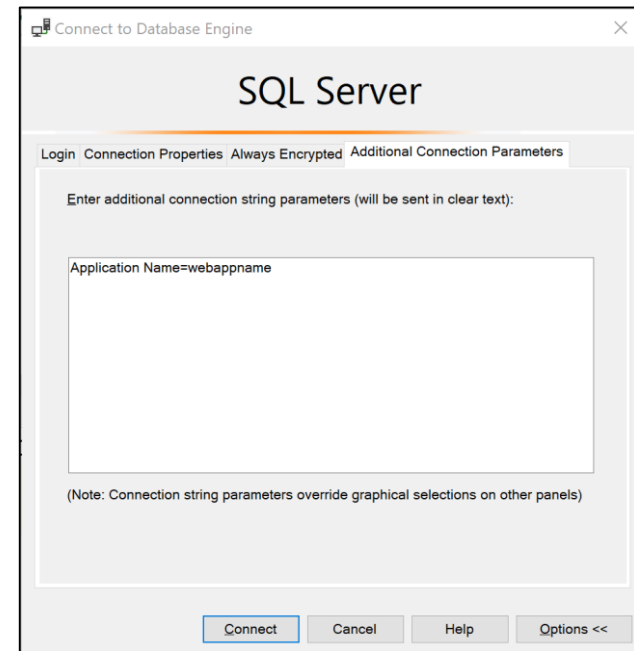
8. Click **Connect**

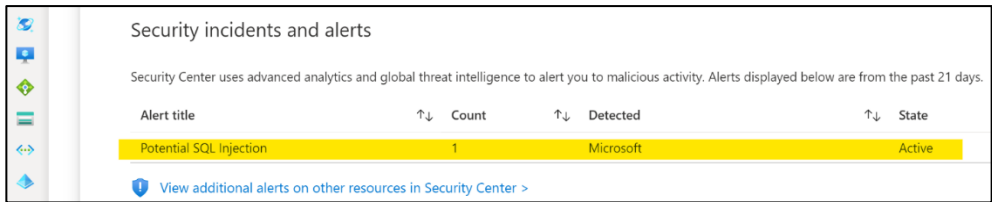
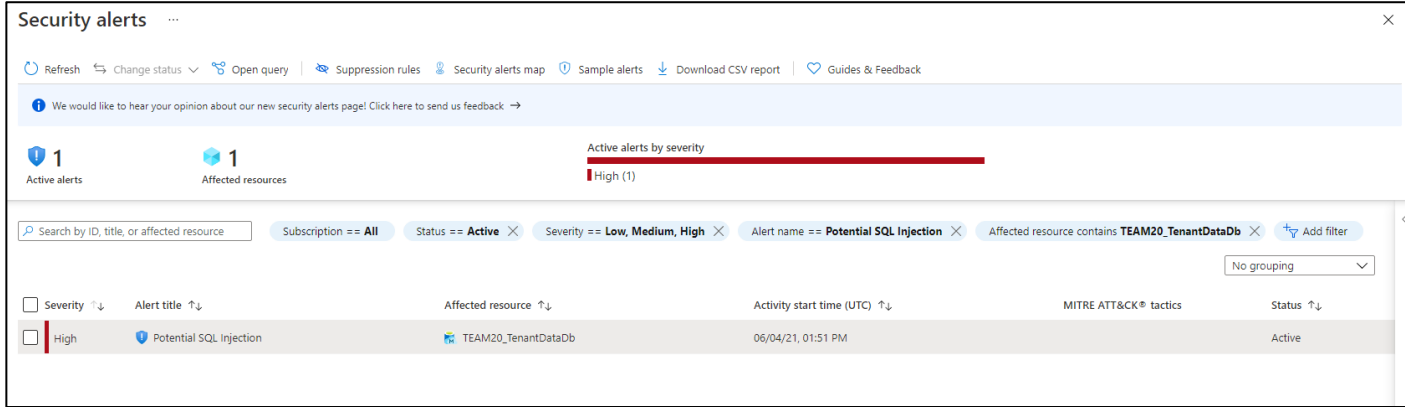
Specify the name of the team Azure SQL Database:
TEAMXX_TenantDataDB



On "Additional Connection Parameters" add a connection string option to specify the application name:

Application Name=webappname



<p>9. Run the query.</p> <p>It will return a list of databases on the server.</p>		
<p>10. Back in the Azure Portal Security Center screen, after a few minutes an Alert should be generated (notice this can take more than 10 minutes):</p>		<p>NOTE: It might take up to 10mins for the alert to appear in the portal</p>
<p>11. Once the Later appears click on it to see the details.</p> <p>Depending on the progress of other teams you may see multiple entries in the details table.</p>		

<p>12. Try clicking on the Alert.</p> <p>Note that you can drill further into the alert to see more details, get explanations and links to documentation on the alert and even advice on how negate and remediate the problem.</p>		
--	--	--

LAB 4: Information Protection using Dynamic Data Masking

In this lab we'll cover information protection. These capabilities offer inner security layers that can be used to protect data. We'll explore how the Dynamic Data Masking feature is part of SQL Server's "secure by default" posture and can protect information by limiting sensitive data exposure by masking it to non-privileged users thus helping to prevent unauthorized access.

Administrators can designate how much of the sensitive data to reveal with minimal impact on the application layer. It's a SQL policy-based security feature (meaning permissions are applied using DDL statements) that hides the sensitive data in the result set of a query over designated database fields, while the data in the database remains unchanged.

For example, a service representative at a call center might identify a caller by confirming several characters of their email address, but the complete email address shouldn't be revealed to the service representative. A masking rule can be defined that masks all the email address in the result set of any query.

More details on Dynamic Data Masking can be found here:

<https://docs.microsoft.com/en-us/azure/azure-sql/database/dynamic-data-masking-overview>

NOTE: As of March 2021 the UNMASK permission can now be applied on a column-by-column basis as per [this announcement](#). Although the documentation is yet to reflect this change there a [Channel 9 video](#) that explains this update.

(PTO)

Narrative	Screenshot/Code	Notes
For databases hosted on SQL Managed Instance, Dynamic Data Masking needs to be configured via TSQL.		
1. In SQL Server Management Studio open a new query window connected to your TEAMXX_TenantDataDB database on the SQL managed instance		
<p>2. To mask the email address column in the customer table we need to change the column definition by running this SQL:</p> <p>Note that we used a built-in email masking function.</p> <p>For details on built in masking functions and how to create custom masks see this documentation: Dynamic Data</p>	<pre>-- Replace XX with your team number USE TEAMXX_TenantDataDb; -- Alter column definition to mask [EmailAddress] data ALTER TABLE [SalesLT].[Customer] ALTER COLUMN [EmailAddress] VARCHAR(50) MASKED WITH (FUNCTION = 'email()');</pre>	

<p>Masking - SQL Server Microsoft Docs</p> <p>Now lets test the masking to see what affect it has.</p>	
<p>1. Open a new query window in SQL Server Management Studio. Copy the SQL statements below and paste them into the query window.</p> <p>One of the main advantages of Dynamic Data Masking is that because the masking/unmasking is performed by the SQL Server engine, masked data will appear masked in *any* client application without the need to make application code changes.</p>	<div data-bbox="667 443 1630 517"> <p>NOTE: The statements below are separate steps - run each step individually and look at the results after each one.</p> </div> <pre data-bbox="680 555 1608 1366"> USE TEAMXX_TenantDataDb; -- Replace XX with your team number -- STEP 1: SELECT performed by a member of db_owner or sysadmin role to show plain text SELECT TOP 100 c.EmailAddress, c.* FROM SalesLT.Customer c; -- STEP 2: Create new database user and give them SELECT permission on [Customer] CREATE USER TestUser WITHOUT LOGIN; GRANT SELECT ON [SalesLT].[Customer] to TestUser; -- STEP 3: SELECT columns from the customer table as the test user EXECUTE AS USER = 'TestUser'; SELECT TOP 100 c.EmailAddress, c.* FROM SalesLT.Customer c; REVERT; -- STEP 4: Grant unmask privilege to the test user GRANT UNMASK TO TestUser; -- STEP 5: Select from the table again as test user </pre> <p>Note that once defined a mask is applied by default – you have to be assigned the UNMASK permission to see any masked data as plain text. But UNMASK is a global permission that applies to *all* masked columns – you can't mask/unmask columns individually.</p>

	<pre>EXECUTE AS USER = 'TestUser'; SELECT c.EmailAddress, c.* FROM SalesLT.Customer c; REVERT;</pre>	
--	--	--

LAB 5: End-to-end encryption using Always Encrypted

THIS LAB CURRENTLY HAS AN ISSUE IN THE DEMO ENIRONMENT. DO NOT USE

In this lab we'll explore data encryption using Always Encrypted.

Always Encrypted is a feature designed to protect sensitive data stored in SQL Server or Azure SQL databases. Always Encrypted allows client applications to encrypt sensitive data and never reveal the encryption keys to the database engine. As a result, Always Encrypted provides a separation between those who own the data and can view it, and those who manage the data but should have no access.

Always Encrypted means just that - even SQL Server can't read data protected by Always Encrypted – only client applications that have access to the encryption keys can unencrypt the data

More details about Always Encrypted can be found here:

<https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/always-encrypted-database-engine?view=sql-server-ver15>

(PTO)

SQL Modernisation Open Hack

The Always Encrypted wizard in Management Studio generates the column encryption keys and places them in an external secure store such as Key Vault where other applications (with the correct permissions) can access them. Once the data is encrypted and the keys are in the secure store SQL Server and Management Studio themselves cannot access them meaning that querying encrypted data using Management Studio simply returns the encrypted has values. The thing to remember is that the data is ***always*** encrypted including in transmission until a client application unencrypts it locally. We shall see the effect of this later in the lab.

First let's encrypt a column, generate the encryption keys and store them in Azure Key Vault.

Narrative	Screenshot/Code	Notes
<p>1. In SQL Server Management Studio right-click on your TEAMXX_TenantDataDB database for the SQL Managed Instance and choose Tasks\Encrypt Columns...</p> <p>This will launch the Always Encrypted wizard</p>		
<p>2. Read the notes on the Introduction page and click Next</p> <p><i>(PTO)</i></p>		

3. Select the **SalesLT.Customer**
Phone column and set the
Encryption Type drop down to
Deterministic

Click **Next**

Always Encrypted

Column Selection

Introduction

Column Selection

Master Key Configuration

Run Settings

Summary

Results

Search column name...

☒ Apply one key to all checked columns: CEK_Auto1 (New)

Encryption Type ⓘ Encryption Key ⓘ

Name	State	Encryption Type	Encryption Key
dbo.Users			
dbo.UserTransactions			
SalesLT.Customer			
CustomerID			
NameStyle			
Title			
FirstName			
MiddleName			
LastName			
Suffix			
CompanyName			
SalesPerson			
EmailAddress			
Phone	*	Choose Type...	CEK_Auto1 (New)
PasswordHash			
PasswordSalt			
rowguid			
ModifiedDate			
SalesLT.Product			

☐ Show affected columns only

< Previous Next > Cancel

4. On the Master Key Configuration screen set the "Select column master key" to "Auto generate column master key".

5. For the Select Key Store Provider option select **Azure Key Vault** and click "Sign In..."

6. An Azure sign-in dialog will be presented, enter the **team's Azure login** (SQLHACK_TEAMxx@YYYY.onmicrosoft.com) to sign-in to the labs Azure subscription. When sign-in is completed the lab subscription and pre-built Azure Key Vault should be pre-populated in the drop downs.

Click **Next**

Always Encrypted

Master Key Configuration

Introduction
Column Selection
Master Key Configuration
Run Settings
Summary
Results

Help

To generate a new column encryption key, a column master key must be selected to protect it. The column master key is stored outside of the database.

Select column master key:
Auto generate column master key

Select the key store provider:
☐ Windows certificate store
☒ Azure Key Vault

You are signed in as user_0A491@OTAPRD621ops.onmicrosoft.com. [Change user](#)

Select a subscription to use:
OTA-PRD-621 (1c4ed14a-54a3-4513-a4ce-c5f0002f4ca5)

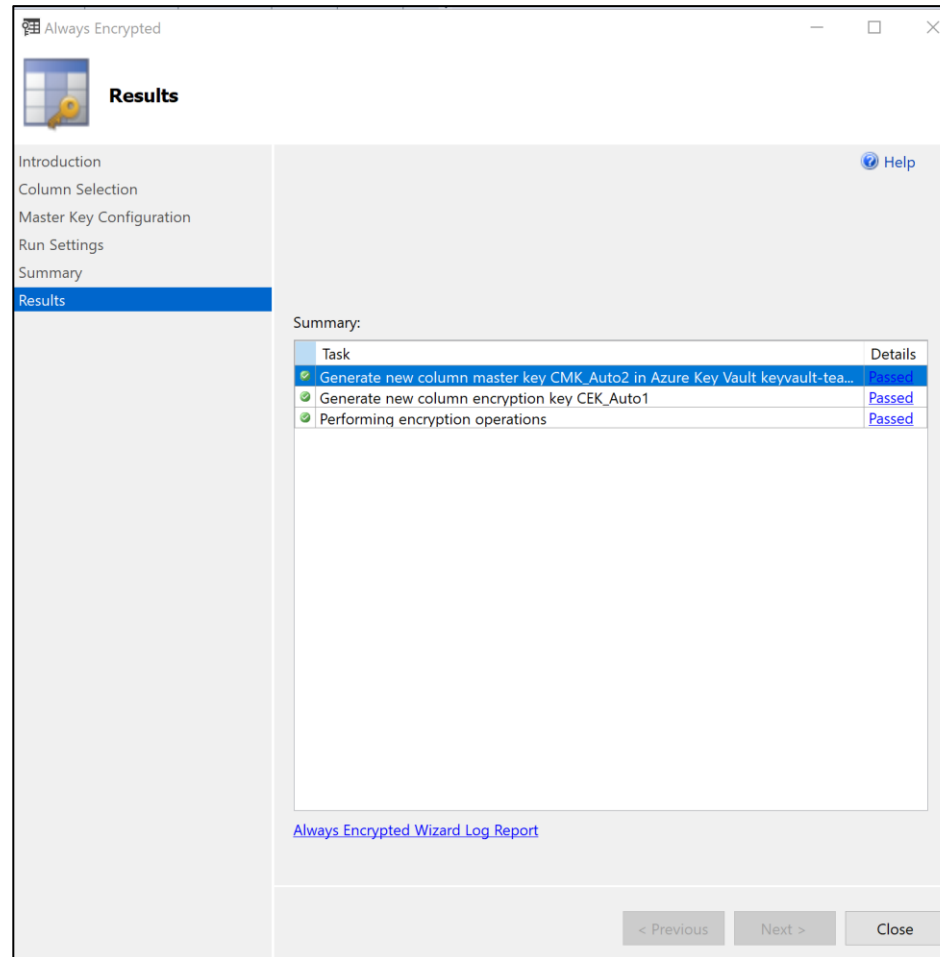
Select an Azure Key Vault:
sqlhack-keyvault-32352

< Previous Next > Cancel

7. On the Run Settings dialog leave **“Proceed to finish now”** selected, click **Next** and then **Finish** on the Summary page.

8. A final Azure subscription sign-in will be presented, select the team’s lab login.

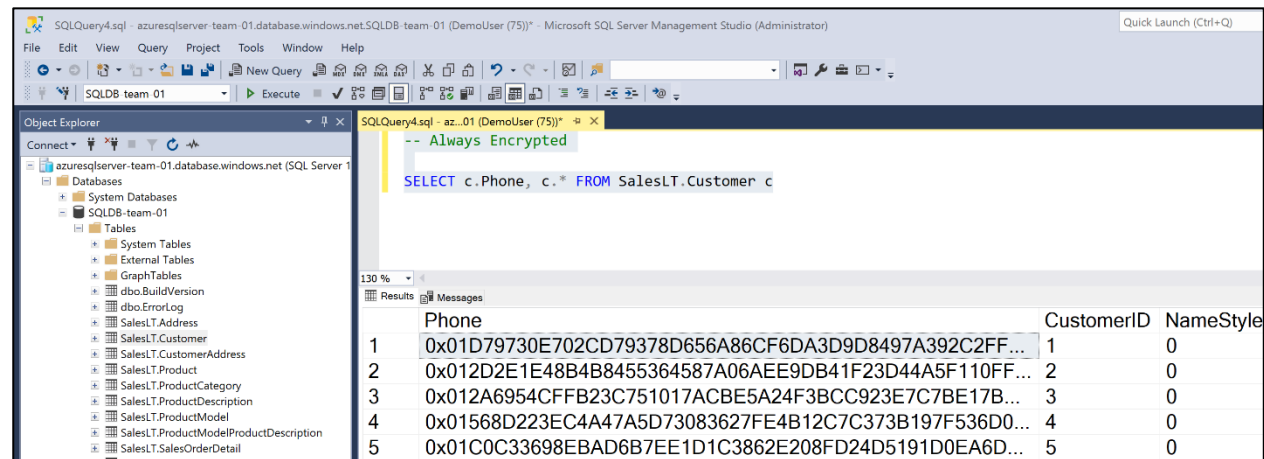
Once the three steps for the encryption process have passed, click **Close**.



The **Phone** column in the **Customer** table should now be encrypted using deterministic encryption.

9. In a query window in SQL Server Management Studio copy and run the following SELECT statement to confirm the state of the column:

```
-- Always Encrypted
USE [TEAMXX_TenantDataDb]; -- Replace XX with your Team number
SELECT c.Phone, c.* FROM SalesLT.Customer c;
```

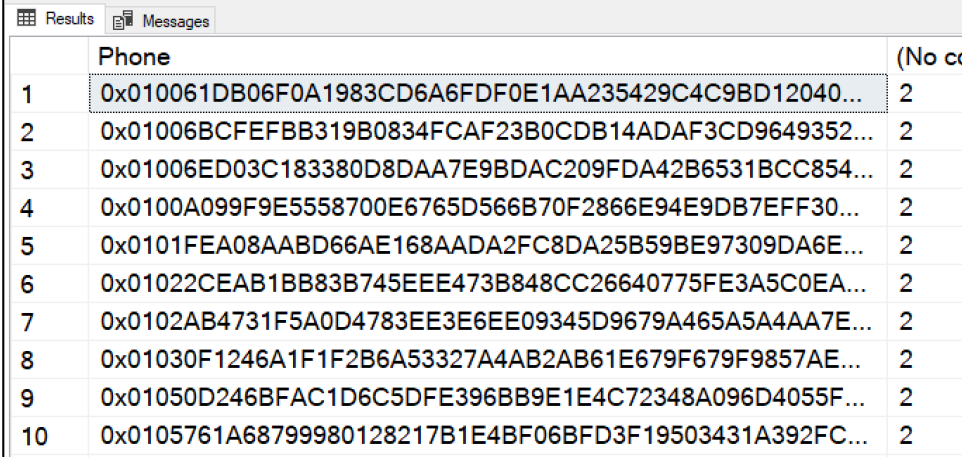


The **Phone** column data is now stored and processed in an encrypted state within the database – regardless of your permissions in SQL Server you will not be able to see the plain text values.

To decrypt the data an application must have access to the key in Key Vault and use it to locally decrypt the encrypted data that SQL Server will return in response to a query.

10. Look in the **Keys** section of the Azure Key Vault (in the **SQLHACK-SHARED** Resource Group) you will see the encryption keys stored there. Note the Key names are not user friendly but can't be altered as the Always Encrypted wizard sets this name.

Best practice for Key Vault is to have a separate Key Vault per app/service per environment. In that case we'd have separate Key Vaults per TeamXX databases making Key management simpler.

Narrative	Screenshot/Code	Notes																																	
<p>During the data encryption process we selected deterministic as the encryption type for encrypting the Phone column. Deterministic encryption means that a specific input value (in this case a phone number string of digits) will always generate the same encrypted has value.</p> <p>Columns that are encrypted with deterministic encryption can therefore be used for equality operations, joins and group by operations. However a determined hacker may be able to work out the recurring hash patterns and thus decrypt the data.</p> <p>Randomized encryption, as its name suggests, means that a specific value will generate different hash values whenever it is encrypted. This is obviously more secure than Deterministic encryption because there is no recurring pattern for a hacker to discern, but it also means equality operators can't be used effectively preventing SQL Server from many of the basic search, join and group by functions upon columns encrypted with Randomized encryption.</p>																																			
<p>11. Now that the Phone column has been encrypted using deterministic encryption, copy the SELECT statement opposite and run it in a query window:</p>	<pre>USE [TEAMXX_TenantDataDb]; -- Replace XX with your Team number --Deterministic encryption allows equality & grouping operations SELECT Phone, COUNT(*) FROM SalesLT.Customer GROUP BY Phone HAVING COUNT(Phone) > 1</pre>																																		
<p>You should see a result set like this:</p>	 <p>The screenshot shows a SQL Server query results window with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with 10 rows. The first column is labeled 'Phone' and the second column is labeled '(No co'. Each row contains a long hexadecimal string representing an encrypted phone number and the value '2'.</p> <table border="1"> <thead> <tr> <th></th> <th>Phone</th> <th>(No co</th> </tr> </thead> <tbody> <tr><td>1</td><td>0x010061DB06F0A1983CD6A6FDF0E1AA235429C4C9BD12040...</td><td>2</td></tr> <tr><td>2</td><td>0x01006BCFEFBB319B0834FCAF23B0CDB14ADAF3CD9649352...</td><td>2</td></tr> <tr><td>3</td><td>0x01006ED03C183380D8DAA7E9BDAC209FDA42B6531BCC854...</td><td>2</td></tr> <tr><td>4</td><td>0x0100A099F9E5558700E6765D566B70F2866E94E9DB7EFF30...</td><td>2</td></tr> <tr><td>5</td><td>0x0101FEA08AABD66AE168AADA2FC8DA25B59BE97309DA6E...</td><td>2</td></tr> <tr><td>6</td><td>0x01022CEAB1BB83B745EEE473B848CC26640775FE3A5C0EA...</td><td>2</td></tr> <tr><td>7</td><td>0x0102AB4731F5A0D4783EE3E6EE09345D9679A465A5A4AA7E...</td><td>2</td></tr> <tr><td>8</td><td>0x01030F1246A1F1F2B6A53327A4AB2AB61E679F679F9857AE...</td><td>2</td></tr> <tr><td>9</td><td>0x01050D246BFAC1D6C5DFE396BB9E1E4C72348A096D4055F...</td><td>2</td></tr> <tr><td>10</td><td>0x0105761A68799980128217B1E4BF06BFD3F19503431A392FC...</td><td>2</td></tr> </tbody> </table>		Phone	(No co	1	0x010061DB06F0A1983CD6A6FDF0E1AA235429C4C9BD12040...	2	2	0x01006BCFEFBB319B0834FCAF23B0CDB14ADAF3CD9649352...	2	3	0x01006ED03C183380D8DAA7E9BDAC209FDA42B6531BCC854...	2	4	0x0100A099F9E5558700E6765D566B70F2866E94E9DB7EFF30...	2	5	0x0101FEA08AABD66AE168AADA2FC8DA25B59BE97309DA6E...	2	6	0x01022CEAB1BB83B745EEE473B848CC26640775FE3A5C0EA...	2	7	0x0102AB4731F5A0D4783EE3E6EE09345D9679A465A5A4AA7E...	2	8	0x01030F1246A1F1F2B6A53327A4AB2AB61E679F679F9857AE...	2	9	0x01050D246BFAC1D6C5DFE396BB9E1E4C72348A096D4055F...	2	10	0x0105761A68799980128217B1E4BF06BFD3F19503431A392FC...	2	<p>Can you explain the results?</p>
	Phone	(No co																																	
1	0x010061DB06F0A1983CD6A6FDF0E1AA235429C4C9BD12040...	2																																	
2	0x01006BCFEFBB319B0834FCAF23B0CDB14ADAF3CD9649352...	2																																	
3	0x01006ED03C183380D8DAA7E9BDAC209FDA42B6531BCC854...	2																																	
4	0x0100A099F9E5558700E6765D566B70F2866E94E9DB7EFF30...	2																																	
5	0x0101FEA08AABD66AE168AADA2FC8DA25B59BE97309DA6E...	2																																	
6	0x01022CEAB1BB83B745EEE473B848CC26640775FE3A5C0EA...	2																																	
7	0x0102AB4731F5A0D4783EE3E6EE09345D9679A465A5A4AA7E...	2																																	
8	0x01030F1246A1F1F2B6A53327A4AB2AB61E679F679F9857AE...	2																																	
9	0x01050D246BFAC1D6C5DFE396BB9E1E4C72348A096D4055F...	2																																	
10	0x0105761A68799980128217B1E4BF06BFD3F19503431A392FC...	2																																	

<p>Simply put, the Customer table contains duplicate customer phone number values.</p> <p>This can be shown using the SELECT statement opposite, copy the statement and run it:</p>	<pre>USE [TEAMXX_TenantDataDb]; -- Replace XX with your Team number -- Customers with duplicate encrypted phone numbers SELECT c.phone, c.* FROM SalesLT.Customer c INNER JOIN (SELECT Phone FROM SalesLT.Customer GROUP BY Phone HAVING COUNT(Phone) > 1) p ON c.Phone = p.Phone ORDER BY c.LastName;</pre>																																																																																											
<p>Note the customer names appearing multiple times with different CustomerIds <i>but the same Phone hash values</i>.</p>	<div><div>Results</div><table><tr><th></th><th>phone</th><th>CustomerID</th><th>NameStyle</th><th>Title</th><th>FirstName</th><th>MiddleName</th><th>LastName</th><th>Suffix</th></tr><tr><td>1</td><td>0x013604AD103DCE1C6CD0D249AB0DBED5EE2FBCA184DFDE...</td><td>582</td><td>0</td><td>Ms.</td><td>Catherine</td><td>R.</td><td>Abel</td><td>NU</td></tr><tr><td>2</td><td>0x013604AD103DCE1C6CD0D249AB0DBED5EE2FBCA184DFDE...</td><td>29485</td><td>0</td><td>Ms.</td><td>Catherine</td><td>R.</td><td>Abel</td><td>NU</td></tr><tr><td>3</td><td>0x011111996E68545BE3A925407550653E771B7A1863FF7CBE8...</td><td>29486</td><td>0</td><td>Ms.</td><td>Kim</td><td>NULL</td><td>Abercrombie</td><td>NU</td></tr><tr><td>4</td><td>0x011111996E68545BE3A925407550653E771B7A1863FF7CBE8...</td><td>579</td><td>0</td><td>Ms.</td><td>Kim</td><td>NULL</td><td>Abercrombie</td><td>NU</td></tr><tr><td>5</td><td>0x014960DBA325133081C0A27ED4C7C0B74E6F51D1D627C466...</td><td>544</td><td>0</td><td>Mr.</td><td>Jay</td><td>NULL</td><td>Adams</td><td>NU</td></tr><tr><td>6</td><td>0x014960DBA325133081C0A27ED4C7C0B74E6F51D1D627C466...</td><td>29492</td><td>0</td><td>Mr.</td><td>Jay</td><td>NULL</td><td>Adams</td><td>NU</td></tr><tr><td>7</td><td>0x01CA36E69E9BF295E59FA05D04A2CAFF64677313BC240818...</td><td>491</td><td>0</td><td>Ms.</td><td>Frances</td><td>B.</td><td>Adams</td><td>NU</td></tr><tr><td>8</td><td>0x01CA36E69E9BF295E59FA05D04A2CAFF64677313BC240818...</td><td>29489</td><td>0</td><td>Ms.</td><td>Frances</td><td>B.</td><td>Adams</td><td>NU</td></tr><tr><td>9</td><td>0x01A0FE8D1157E812420475871A26B93CDE41BC150A82C2ED4...</td><td>29494</td><td>0</td><td>Mr.</td><td>Samuel</td><td>N</td><td>Aspelli</td><td>NU</td></tr></table></div>		phone	CustomerID	NameStyle	Title	FirstName	MiddleName	LastName	Suffix	1	0x013604AD103DCE1C6CD0D249AB0DBED5EE2FBCA184DFDE...	582	0	Ms.	Catherine	R.	Abel	NU	2	0x013604AD103DCE1C6CD0D249AB0DBED5EE2FBCA184DFDE...	29485	0	Ms.	Catherine	R.	Abel	NU	3	0x011111996E68545BE3A925407550653E771B7A1863FF7CBE8...	29486	0	Ms.	Kim	NULL	Abercrombie	NU	4	0x011111996E68545BE3A925407550653E771B7A1863FF7CBE8...	579	0	Ms.	Kim	NULL	Abercrombie	NU	5	0x014960DBA325133081C0A27ED4C7C0B74E6F51D1D627C466...	544	0	Mr.	Jay	NULL	Adams	NU	6	0x014960DBA325133081C0A27ED4C7C0B74E6F51D1D627C466...	29492	0	Mr.	Jay	NULL	Adams	NU	7	0x01CA36E69E9BF295E59FA05D04A2CAFF64677313BC240818...	491	0	Ms.	Frances	B.	Adams	NU	8	0x01CA36E69E9BF295E59FA05D04A2CAFF64677313BC240818...	29489	0	Ms.	Frances	B.	Adams	NU	9	0x01A0FE8D1157E812420475871A26B93CDE41BC150A82C2ED4...	29494	0	Mr.	Samuel	N	Aspelli	NU	
	phone	CustomerID	NameStyle	Title	FirstName	MiddleName	LastName	Suffix																																																																																				
1	0x013604AD103DCE1C6CD0D249AB0DBED5EE2FBCA184DFDE...	582	0	Ms.	Catherine	R.	Abel	NU																																																																																				
2	0x013604AD103DCE1C6CD0D249AB0DBED5EE2FBCA184DFDE...	29485	0	Ms.	Catherine	R.	Abel	NU																																																																																				
3	0x011111996E68545BE3A925407550653E771B7A1863FF7CBE8...	29486	0	Ms.	Kim	NULL	Abercrombie	NU																																																																																				
4	0x011111996E68545BE3A925407550653E771B7A1863FF7CBE8...	579	0	Ms.	Kim	NULL	Abercrombie	NU																																																																																				
5	0x014960DBA325133081C0A27ED4C7C0B74E6F51D1D627C466...	544	0	Mr.	Jay	NULL	Adams	NU																																																																																				
6	0x014960DBA325133081C0A27ED4C7C0B74E6F51D1D627C466...	29492	0	Mr.	Jay	NULL	Adams	NU																																																																																				
7	0x01CA36E69E9BF295E59FA05D04A2CAFF64677313BC240818...	491	0	Ms.	Frances	B.	Adams	NU																																																																																				
8	0x01CA36E69E9BF295E59FA05D04A2CAFF64677313BC240818...	29489	0	Ms.	Frances	B.	Adams	NU																																																																																				
9	0x01A0FE8D1157E812420475871A26B93CDE41BC150A82C2ED4...	29494	0	Mr.	Samuel	N	Aspelli	NU																																																																																				
<p>There are several ways to search for the non-duplicate customer rows in the Customer table, one approach is to simply modify the previous query to show the non-duplicate customer rows (based on the assumption that a single phone number represents a single customer):</p>	<pre>USE [TEAMXX_TenantDataDb]; -- Replace XX with your Team number -- Customer without duplicate encrypted phone numbers SELECT c.phone, c.* FROM SalesLT.Customer c INNER JOIN (SELECT Phone FROM SalesLT.Customer GROUP BY Phone HAVING COUNT(Phone) = 1) p ON c.Phone = p.Phone ORDER BY c.LastName;</pre>																																																																																											

SQL Modernisation Open Hack

Running the query above should produce a result set as shown below:

Results Messages									
	phone	CustomerID	NameStyle	Title	FirstName	MiddleName	LastName	Suffix	CompanyName
1	0x013311A77DDE397400207B1D0D4AA808A72851BC3F582A6...	294	0	Mr.	Alberto	F.	Baltazar	NULL	Flawless Bike
2	0x0150CEA9208CF5F91590478B3E10E77AA6F16A010355B4D1...	419	0	Mr.	Darrell	M.	Banks	NULL	Exertion Acti
3	0x01CDE1A10DC7D376F60469E0E54A2C3D196A85FEC1C93E8...	416	0	Ms.	Angela	NULL	Barbariol	NULL	Impromptu T
4	0x012C5ED510B102E673C1022AE9D19B45C4908BA08189B34...	204	0	Mr.	Ido	NULL	Ben-Sachar	NULL	Traction Tire
5	0x019371EDA028343E6631905CF2EB8A34BDDBAE1625D99AB...	565	0	Mr.	Richard	M.	Bentley	Jr.	Metallic Pain
6	0x01E05FE8D7874E78BEED98B14651A25FB0802BC21FD917C...	402	0	Ms.	Marian	M.	Berch	NULL	Pro Sporting
7	0x01C550DB29C734B5D1C7DBA9D1C949FA37EEB93F8A9B43...	329	0	NULL	Andreas	NULL	Berglund	NULL	Unicycles, B
8	0x01329536FDA780EDB56C3C4F6B1FE4473613A7F48B73EB6...	507	0	Mr.	Mary	NULL	Bishop	NULL	Global Sport
9	0x019AE155A29B5AB0BE148693CEBCC137C822B5D7E72925B...	174	0	Mr.	Michael	L.	Bohling	NULL	Toy Manufac
10	0x013B10F5DF048F77E1CAF76509FA5AC5066E48DC614C09E...	347	0	Ms.	Ingrid	K.	Burkhardt	NULL	Bike Univers
11	0x0128933A81A4C99F8B88EC1D4F43AB5E00BB1B4DE42E30B...	415	0	Mr.	Greg	NULL	Chapman	NULL	Imaginary To
12	0x01E8C1F39DDFFE53FF4205BADA45FB2B5552F8D14CCFFE4...	192	0	Mr.	Hao	NULL	Chen	NULL	Nuts and Bo
13	0x01CEA477679DE09495A6C69A33158E6E01147B4443730906...	132	0	Ms.	Teanna	M.	Cobb	NULL	Another Spo
14	0x012077A31B6CBD7AE0B9D751FE59E4C47518B8A0CC31E50...	371	0	Mr.	Bruno	NULL	Costa Da Silva	NULL	Custom Sale

(PTO)

Optional Lab – Use Randomized Encryption with Always Encrypted

In this optional activity we'll use Randomized encryption to encrypt a new column that we'll add to the Customer table.

Narrative	Screenshot/Code	Notes
<p>1. In a query window in SQL Server Management Studio copy and run the following TSQL statements to add and populate a new column in the Customer table.</p>	<pre>USE [TEAMXX_TenantDataDb]; -- Replace XX with your Team number -- Add a new column to the customer table ALTER TABLE [SalesLT].[Customer] ADD LastNameEncrypted NVARCHAR(50) NULL; GO -- Copy plain text [LastName] values into the new column UPDATE SalesLT.Customer SET LastNameEncrypted = LastName; SELECT LastNameEncrypted, * FROM SalesLT.Customer; -- Show the plain text values can obviously be used in grouping SELECT LastNameEncrypted, COUNT(LastNameEncrypted) FROM SalesLT.Customer GROUP BY LastNameEncrypted HAVING COUNT(LastNameEncrypted) > 1;</pre>	
<p>2. In the Object Explorer window right-click at the TEAMXX_TenantDataDb level "Task\Encrypt Column..."</p>		

3. Run the rough the Always Encrypted wizard once again but this time encrypt the new **LastNameEncrypted** column and set the Encryption Type to **Randomized**

Name	State	Encryption Type	Encryption Key
dbo.UserTransactions			
SalesLT.Customer			
CustomerID			
NameStyle			
Title			
FirstName			
MiddleName			
LastName			
Suffix			
CompanyName			
SalesPerson			
EmailAddress			
Phone		Deterministic	CEK_Auto1
PasswordHash			
PasswordSalt			
rowguid			
ModifiedDate			
LastNameEncrypted		Randomized	CEK_Auto1
SalesLT.Product			

☐ Show affected columns only

< Previous **Next >** Cancel

4. Accept all the remaining settings in the encryption dialogs by clicking **Next** and eventually **Finish**. In the final encryption dialog an Azure subscription login dialog may be presented to access the team Azure Key Vault instance. If so enter the team Azure login and password.

5. Once the new [LastNameEncrypted] column has been successfully encrypted using Randomized encryption copy the SELECT statement below and run it in a query window

```
USE [TEAMXX_TenantDataDb]; -- Replace XX with your Team number

-- Same values now have different hash values
SELECT LastNameEncrypted, LastName
FROM SalesLT.Customer
ORDER BY LastName;
```

	Results	Messages
	LastNameEncrypted	LastName
1	0x01C09FA219F5494A22F868D2C7C75215DDFA7515BCF3C8872...	Abel
2	0x0147EB900FAAACA1D9E5ACB09BDDA0744F110370DFABB16...	Abel
3	0x01BF7A49BB15F3114D06A5BAD00F77CD347F8CC3E65C1394...	Abercrombie
4	0x01162860F7C08D46C5EB7BDACDE8ACA178B57CE5366D07D...	Abercrombie
5	0x01A27051931AA5AD146C133A7788DDD4212D31EB99CA6FF8...	Adams
6	0x01643C6FB91A08C8A84EAAAC1A81DCED8A5FD96B370501DB...	Adams
7	0x01163C7173A437AA5122AF527B27C47373F7B359CFA879011...	Adams
8	0x01D2B6F8732B836F80D6D6E8790D81E01C4A37B07C8FFEC0...	Adams
9	0x01ABB54537FD97DAC82B97A1471F05C7EBA9E2AC1FC3F3AA...	Agcaoli
10	0x01CE1F978F64F6482CCE9F867A28C083C88E3051EF9E160A4...	Agcaoli

Observe that all the encryption hash values of the LastName column are unique.

(PTO)

6. Run the opposite SELECT statement to show the number of unique Randomized hash values that now exist for each LastName value:

The results will look like this:

```
USE [TEAMXX_TenantDataDb]; -- Replace XX with your Team number

-- Count different hash values for the same LastName text
SELECT LastName, COUNT(LastNameEncrypted) AS UniqueHashValues
FROM SalesLT.Customer
GROUP BY LastName
ORDER BY UniqueHashValues DESC;
```

	LastName	UniqueHashValues
1	Miller	10
2	Johnson	8
3	Lee	6
4	Li	6
5	Liu	6
6	Kim	6
7	Brown	6
8	Evans	6
9	Thompson	6
10	Sullivan	4
11	Vargas	4
12	Burnett	4
13	Brooks	4

Whilst more useful from a SQL functionality point of view, data encrypted with a Deterministic algorithm could result in some decerning of information through ad-hoc querying. We may not be able to see the encrypted Last Names but we can see which records share the most popular hash values and therefore we might make intelligent guesses as to what the actual value might be.

So from a security perspective using the Randomized encryption algorithm is much more secure but this is done at the sacrifice of even basic SQL functionality.