Module 5: Data privacy and protection

**Overview:**

In this module, you learn about the data privacy, security, and recovery features available in Azure SQL Data Warehouse. You will configure these features to prevent unauthorized users/logins from accessing confidential data, to monitor access to confidential columns, to restrict access to specific rows in the data warehouse, and to recover from user/application error in the data warehouse.

**Pre-requisites:**

* Azure SQL Data Warehouse
* SQL Server Management Studio
* Azure PowerShell

If you haven’t run through the earlier modules in the series, launch PowerShell and run the Module5Init.ps1 PowerShell script in the Module 5\Resources folder. This will configure your data warehouse with the pre-requisites needed to complete this lab.

**Login to your Azure subscription in PowerShell**

The first few sections of this module will use PowerShell to configure settings and values. If you haven’t previously logged into your Azure account, run the following to setup your PowerShell session.

# ------- Edit the variables below to set session-wide variables ---------

$subscriptionName = '<SubscriptionName>'

$participantNumber = <ParticipantNumber>

$resourceGroupName = '<ResourceGroupName>'

# Log into your Azure account

Connect-AzAccount -Subscription $SubscriptionName

Run the following to setup the variables you’ll use to configure Azure PowerShell commands in the lab:

Import-Module "C:\USGSdata\BruteForceSQL\Invoke-LoginAttack"

# ------- Setup module variables ---------

$serverName = 'usgsserver' + $participantNumber

$fullyQualifiedServerName = $serverName + '.database.windows.net'

$dataLakeAccountName = 'usgsdatalake' + $participantNumber

$blobStorageAccountName = 'usgsstaging' + $participantNumber

$securityScanResultsContainerName = 'vulnerabilityresults'

$dataWarehouseName = 'usgsdataset'

$restorePointLabel = 'preapplicationupgrade\_' + (Get-Date -Format FileDateTimeUniversal)

$restoredWarehouseName = 'usgsdataset\_restored'

$userListLocation = "C:\USGSdata\BruteForceSQL\userList.txt"

$passwordListLocation = "C:\USGSdata\BruteForceSQL\passwordList.txt"

$dbUserName = 'usgsadmin'

$dbPassword = 'P@ssword' + $participantNumber

$securePasswordString = ConvertTo-SecureString $dbPassword -AsPlainText -Force

$dbCredentials = New-Object System.Management.Automation.PSCredential($dbUserName, $securePasswordString)

**Create user-defined restore points before an application upgrade**

Azure SQL Data Warehouse includes automated database snapshots that can be leveraged to recover or copy a data warehouse to a previous state. These snapshots support an eight-hour recovery point objective (RPO) and are available to be used for 7 days. If you require a faster RPO or you require your snapshot to be available for longer than 7 days, you can manually trigger a snapshot to save the current database state. This is good practice to follow before and after large modifications to your data warehouse - it allows quicker recovery times in the event of any workload interruptions or user errors.  
In your existing PowerShell session

1. Create a custom restore point for your data warehouse. Note the creation date and time returned in the output

# Creates a custom restore point

New-AzSqlDatabaseRestorePoint -ResourceGroupName $resourceGroupName -ServerName $serverName -DatabaseName $dataWarehouseName -RestorePointLabel $restorePointLabel

1. Simulate data corruption in the application upgrade process by running the following command to drop a table in the data warehouse

# Drop a table in the warehouse to simulate an error in application upgrade

Invoke-Sqlcmd -Query "DROP TABLE STG.dimAreaProtectionCategory" -ServerInstance $fullyQualifiedServerName -Database $dataWarehouseName -Username $dbCredentials.UserName -Password $dbCredentials.GetNetworkCredential().Password

1. Confirm that the table does not exist anymore by running the following command

# Select from dropped table

Invoke-Sqlcmd -Query "SELECT TOP 10 \* FROM STG.dimAreaProtectionCategory" -ServerInstance $fullyQualifiedServerName -Database $dataWarehouseName -Username $dbCredentials.UserName -Password $dbCredentials.GetNetworkCredential().Password

1. Restore your data warehouse from the restore point you created earlier by running the commands below. Note: this will take some time to complete.

# List the last 10 database restore points

$restorePointList = ((Get-AzSqlDatabaseRestorePoints -ResourceGroupName $resourceGroupName -ServerName $serverName -DatabaseName ($dataWarehouseName)).RestorePointCreationDate)[-10..-1]

# Get restore point time

$targetTime = $restorePointList | Select-Object -Last 1

# Restore data warehouse to a new instance

$originDatabase = Get-AzSqlDatabase -ResourceGroupName $resourceGroupName -ServerName $serverName -DatabaseName $dataWarehouseName

$restoredDatabase = Restore-AzSqlDatabase -FromPointInTimeBackup -PointInTime $targetTime -ResourceGroupName $resourceGroupName -ServerName $serverName -TargetDatabaseName $restoredWarehouseName -ResourceId $originDatabase.ResourceID

1. Connect to the restored data warehouse instance and verify that the table exists. You can also login to the Azure portal to see the restored SQL data warehouse instance.

# Select from restored data warehouse

Invoke-Sqlcmd -Query "SELECT TOP 10 \* FROM STG.dimAreaProtectionCategory" -ServerInstance $fullyQualifiedServerName -Database $restoredWarehouseName -Username $dbCredentials.UserName -Password $dbCredentials.GetNetworkCredential().Password

1. You can pause or delete the restored data warehouse instance after you have finished exploring it.

**Enable Advanced Threat Protection on your Azure SQL Server**

SQL Advanced Threat Protection (ATP) is a unified package for advanced SQL security capabilities in your Azure SQL Server. It includes functionality for discovering and classifying sensitive data, surfacing and mitigating potential database vulnerabilities, and detecting anomalous activities that could indicate a threat to your database.

In your existing PowerShell session:

1. Turn on data discovery and classification on your Azure SQL Server

# Enable advanced threat protection on Azure SQL Server.

Note: this feature adds a $15 monthly cost per server to your Azure bill

Enable-AzSqlServerAdvancedThreatProtection -ResourceGroupName $resourceGroupName -ServerName $serverName

**Run a security vulnerability assessment on your data warehouse**

The Vulnerability Assessment service of the SQL ATP package allow you to discover, track, and fix potential security vulnerabilities in your data warehouse. It uses a knowledge base of rules based on Microsoft’s best practice guidelines and outlines the security issues that are the biggest risks to your data warehouse and its valuable data. In this section, you will run a vulnerability assessment to see what issues you should address.

In your existing PowerShell session:

1. Configure a storage account and container to store assessment results

# -- Run a security vulnerability assessment on your data warehouse ---

# Configure storage account for vulnerability assessment results

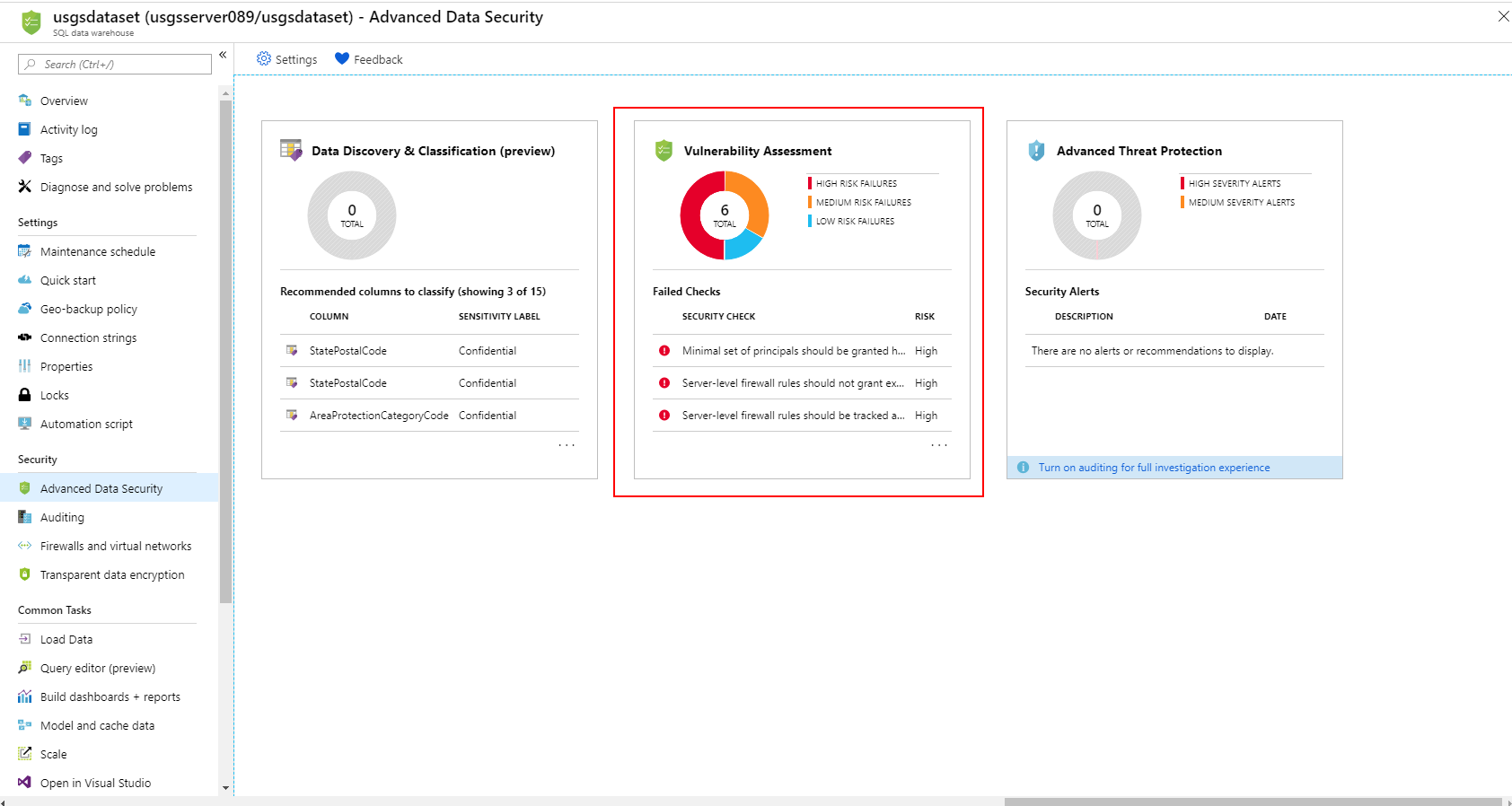
Update-AzSqlDatabaseVulnerabilityAssessmentSettings -ResourceGroupName $resourceGroupName -ServerName $serverName -DatabaseName $dataWarehouseName -StorageAccountName $blobStorageAccountName -ScanResultsContainerName $securityScanResultsContainerName -EmailAdmins $False

1. Run a security scan on the data warehouse. You will notice that there are some outstanding issues to fix on your data warehouse.

# Run a security scan on the data warehouse

$scanLabel = 'usgsdw-initialscan'

Start-AzSqlDatabaseVulnerabilityAssessmentScan -ResourceGroupName $resourceGroupName -ServerName $serverName -DatabaseName $dataWarehouseName -ScanId $scanLabel

1. Login to the Azure portal and navigate to your data warehouse. In the ‘Advanced Data Security’ pane underneath the security header, you will see the results of the Vulnerability assessment scan. Click into the ‘Vulnerability Assessment’ tile to get more details, remediation steps, and to set a baseline for the next security scan.  
   

**Configure threat detection capabilities for your data warehouse**

The threat detection capabilities of the SQL ATP package allow you to monitor and alert on anomalous activities on your data warehouse. The activities could range from a potential SQL injection attack, a brute force password attack, or even login from an unfamiliar SQL Server principal. In this section, you will launch a brute force password attack on your data warehouse to test the threat detection capabilities of the ATP package.  
  
In your existing PowerShell session

1. Configure threat detection for your data warehouse. Below you create policies for the entire server, so any new databases or data warehouses created inherit this configuration. Modify the ‘NotificationRecipientsEmail’ parameter below with your email address if you would like to receive an email for detected threats.

# Set an auditing policy on the warehouse. This is required to use threat detection capabilities

Set-AzSqlServerAuditing -State Enabled -ResourceGroupName $resourceGroupName -ServerName $serverName -StorageAccountName $blobStorageAccountName

# Set a threat detection policy. This can be configured to email the admins who manage the data warehouse

Set-AzSqlServerThreatDetectionPolicy -ResourceGroupName $resourceGroupName -ServerName $serverName -StorageAccountName $blobStorageAccountName -EmailAdmins $False -NotificationRecipientsEmail "admin@example.com"

1. Run a brute force password attack against your data warehouse.

# Run brute force login script to test threat detection policy

Invoke-LoginAttack -ComputerName $fullyQualifiedServerName -UserList $userListLocation -PasswordList $passwordListLocation -Service SQL -Verbose

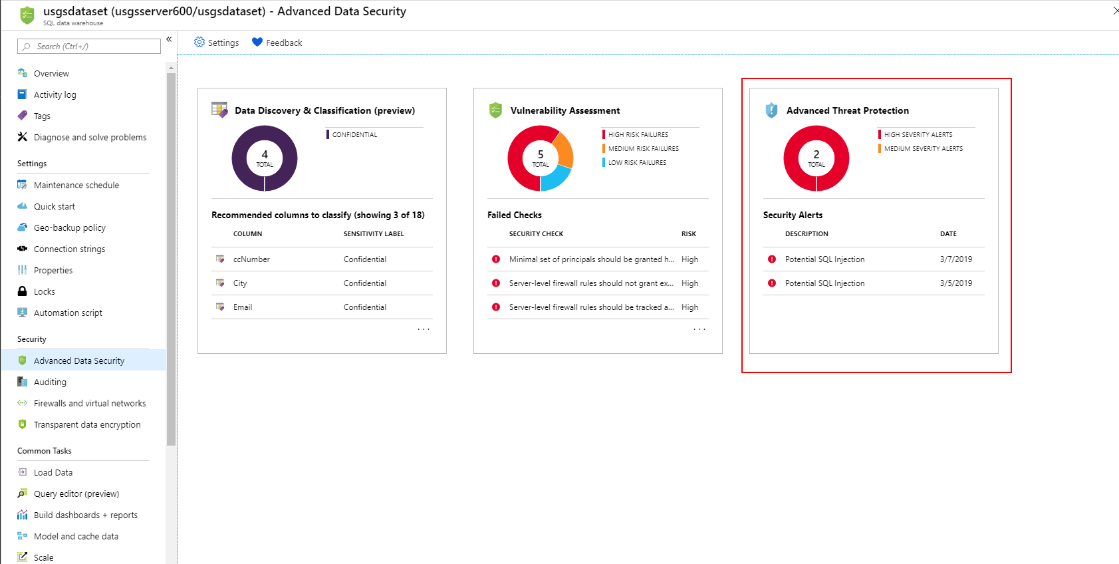
1. Run a SQL injection against your data warehouse.

# Run simple SQL injection attack against table in DW

$searchInput = "06031' OR 1=1--"

Invoke-Sqlcmd -Query "SELECT TOP 500 \* FROM prod.factWeatherMeasurements WHERE fipscountyCode = $fipsCounty OR ObservationTypeCode LIKE '%$searchInput%' OR StationId LIKE '%$searchInput%' OR ObservationSourceFlag LIKE '%$searchInput%'" -ServerInstance $fullyQualifiedServerName -Database $dataWarehouseName -Username $dbUserName -Password $dbPassword

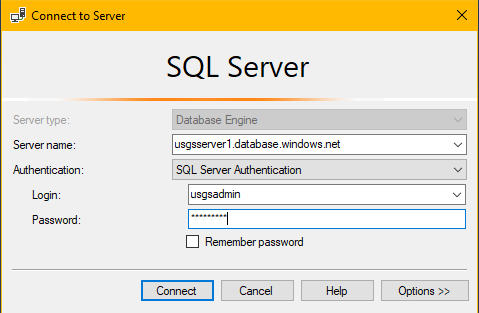
1. Login to the Azure portal and navigate to your data warehouse. In the ‘Advanced Threat Protection’ pane underneath the security header, you will see the security alerts generated from the brute force password attack. It can take up to 10 minutes for the alert to show up in the portal.



**Connect to your data warehouse in SQL Server Management Studio**

The rest of this module will use T-SQL code to configure data privacy for columns in your data warehouse. Follow the steps below to login to your data warehouse

In SQL Server Management Studio (SSMS):

1. Launch the Object Explorer and connect to your sample data warehouse with the credentials provided below – making sure to replace ‘##’ with your participant number:  
   Server name: usgsserver##.database.windows.net  
   Authentication: SQL Server Authentication  
   Username: usgsadmin  
   Password: P@ssword##  
   

**Restrict access to data records using row-level security**

One of the security capabilities of your data warehouse is the ability to restrict rows returned based on the identity of the user executing the query. In this section, you will configure a security policy that allows users to only see data for their states and jurisdiction.

In SQL Server Management Studio (SSMS):

Using the ‘usgsadmin’ session, right-click on your data warehouse and select ‘New query’ then run the following:

1. Create table to define access restrictions for users and roles

--- Create table to store access assignments for users and roles

CREATE TABLE dbo.userAccessAssignments

(

userName NVARCHAR(100) NULL,

stateFIPSAssignment NVARCHAR(2) NULL

)

WITH (DISTRIBUTION=ROUND\_ROBIN)

1. Insert values into access restrictions table

INSERT INTO dbo.userAccessAssignments

VALUES('CaliforniaForestryManager','06');

INSERT INTO dbo.userAccessAssignments

VALUES('CaliforniaForestryEmployee','06');

INSERT INTO dbo.userAccessAssignments

VALUES('TexasForestryEmployee','48');

INSERT INTO dbo.userAccessAssignments

VALUES('TexasForestryManager','48');

INSERT INTO dbo.userAccessAssignments

VALUES('usgsadmin','00');

1. Create predicate function and security policy that filters rows based on which states users can see.

--- Create predicate function that checks which states the current user can see

--- A state FIPS code assignment of 00, allows viewing data from all states

CREATE function [dbo].[fn\_RLSSecurityPredicate]

(@stateFipsCode AS VARCHAR(2))

RETURNS TABLE

WITH SCHEMABINDING

AS

RETURN

SELECT 1 AS PredicateResult

FROM dbo.userAccessAssignments a

WHERE (a.userName = SYSTEM\_USER) AND (a.stateFIPSAssignment = @stateFipsCode OR a.stateFIPSAssignment = '00')

GO

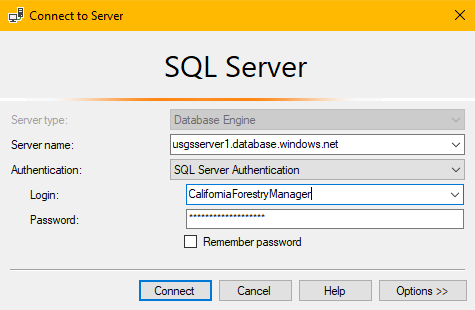
--- Create security policy for factFireEventsTable

CREATE SECURITY POLICY prod.factFireEventsFilter

ADD FILTER PREDICATE dbo.fn\_RLSSecurityPredicate(StateFipsCode)

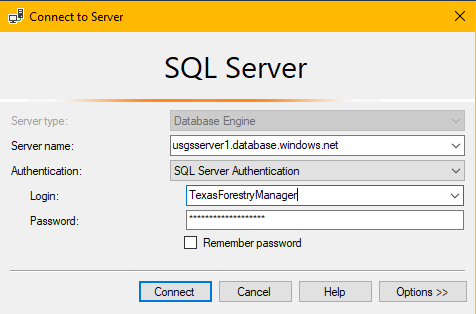
ON prod.factFireEvents

WITH (STATE=ON)

1. Login using credentials for the ‘CaliforniaForestryManager’ user - make sure to replace ‘##’ with the deployment number you were assigned at launch. Run the query below to test the security policy you created. You should only see records for California state.  
   Server name: usgsserver##.database.windows.net  
   Authentication: SQL Server Authentication  
   Username: CaliforniaForestryManager  
   Password: P@ssword##  
   

--Select top 100 records from fire events table

SELECT TOP 100 \* from factfireEvents

1. Login using credentials for the ‘TexasForestryManager’ user - make sure to replace ‘XX’ with the deployment number you were assigned at launch. Run the query below to test the security policy you created. You should only see records for Texas state.  
   Server name: usgsserver##.database.windows.net  
   Authentication: SQL Server Authentication  
   Username: TexasForestryManager  
   Password: P@ssword##  
   

--Select top 100 records from fire events table

SELECT TOP 100 \* from prod.factfireEvents

Bonus: configure Power BI report to access DW instance and login as TexasForestryManager/CaliforniaForestryManager to test out row-level security

**Discover, classify, and protect the sensitive data in your warehouse**Data Discovery & Classification (currently in preview) provides advanced capabilities built into your data warehouse for discovering, classifying, labeling and protecting the sensitive data in your warehouse. You can use this feature to help meet data privacy standards, alert on anomalous access to sensitive data, and control access to databases containing highly sensitive data.

In SQL Server Management Studio (SSMS):

Right-click on your data warehouse and select ‘New query’ then run the following:

1. View all columns with sensitivity classifications

--- View all columns in the database with sensitivity classifications

SELECT sys.schemas.name AS SchemaName,

sys.all\_objects.name AS TableName,

sys.all\_columns.name AS ColumnName,

[Label],

Label\_ID,

Information\_Type,

Information\_Type\_ID

FROM sys.sensitivity\_classifications

LEFT JOIN sys.all\_objects ON sys.sensitivity\_classifications.major\_id =

sys.all\_objects.object\_id

LEFT JOIN sys.all\_columns ON sys.sensitivity\_classifications.major\_id =

sys.all\_columns.object\_id

AND sys.sensitivity\_classifications.minor\_id = sys.all\_columns.column\_id

JOIN sys.schemas ON sys.all\_objects.schema\_id = sys.schemas.schema\_id

1. Configure and classify confidential columns for the access restrictions table you created in the previous section. This can also be done via the Azure portal by navigating to your Data Warehouse, the ‘Security’ Header, and finally the Data Discovery & Classification pane.

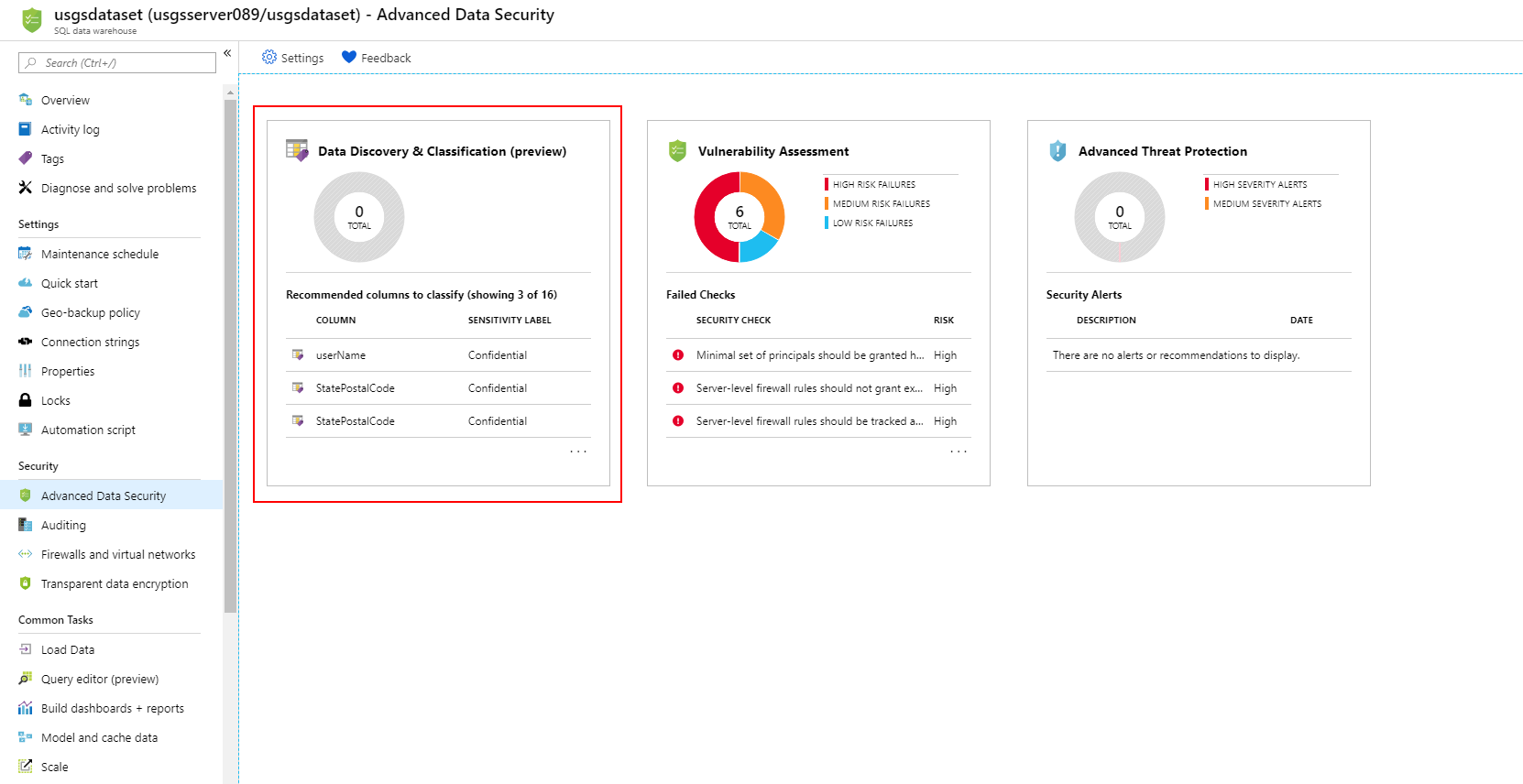
-- Classify columns in the factFireEvents table as confidential

ADD SENSITIVITY CLASSIFICATION

TO [dbo].userAccessAssignments.userName,

[dbo].userAccessAssignments.stateFIPSAssignment

WITH (LABEL='Confidential', INFORMATION\_TYPE='Classified')

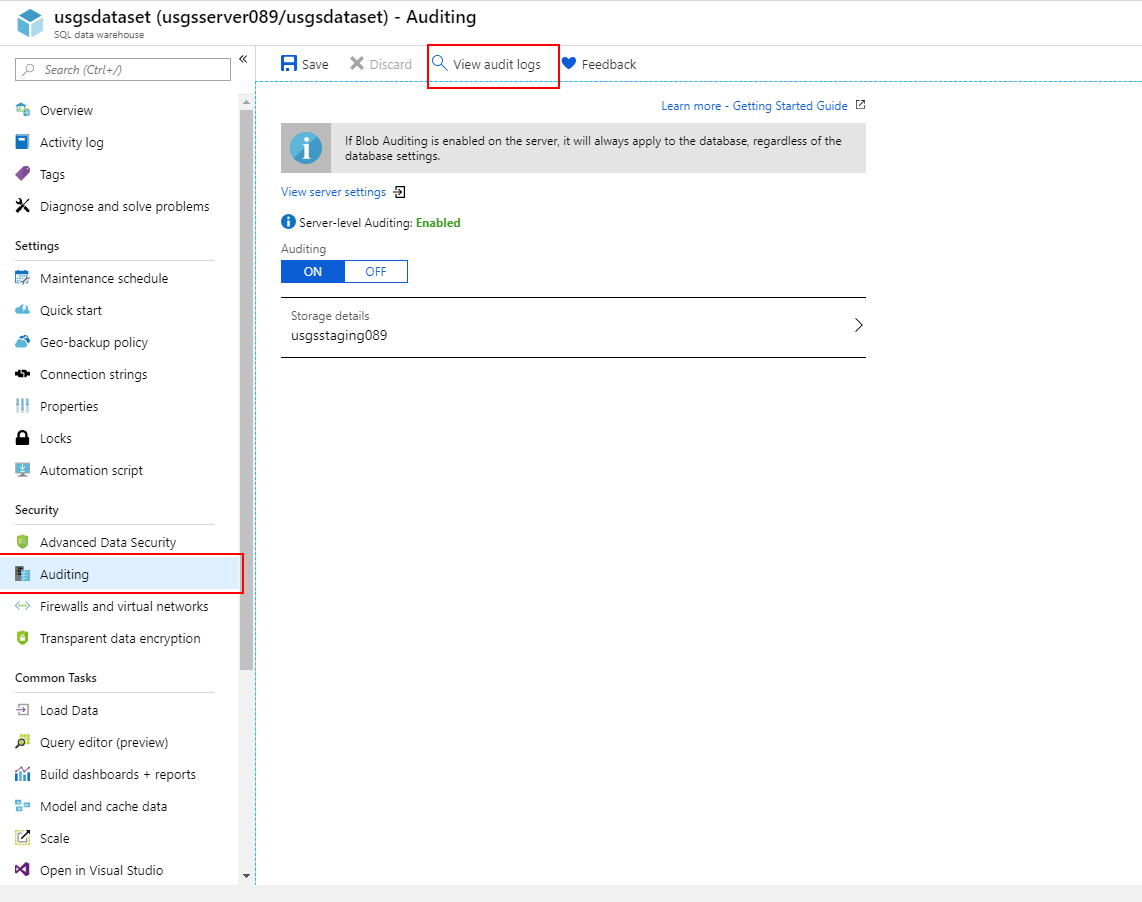


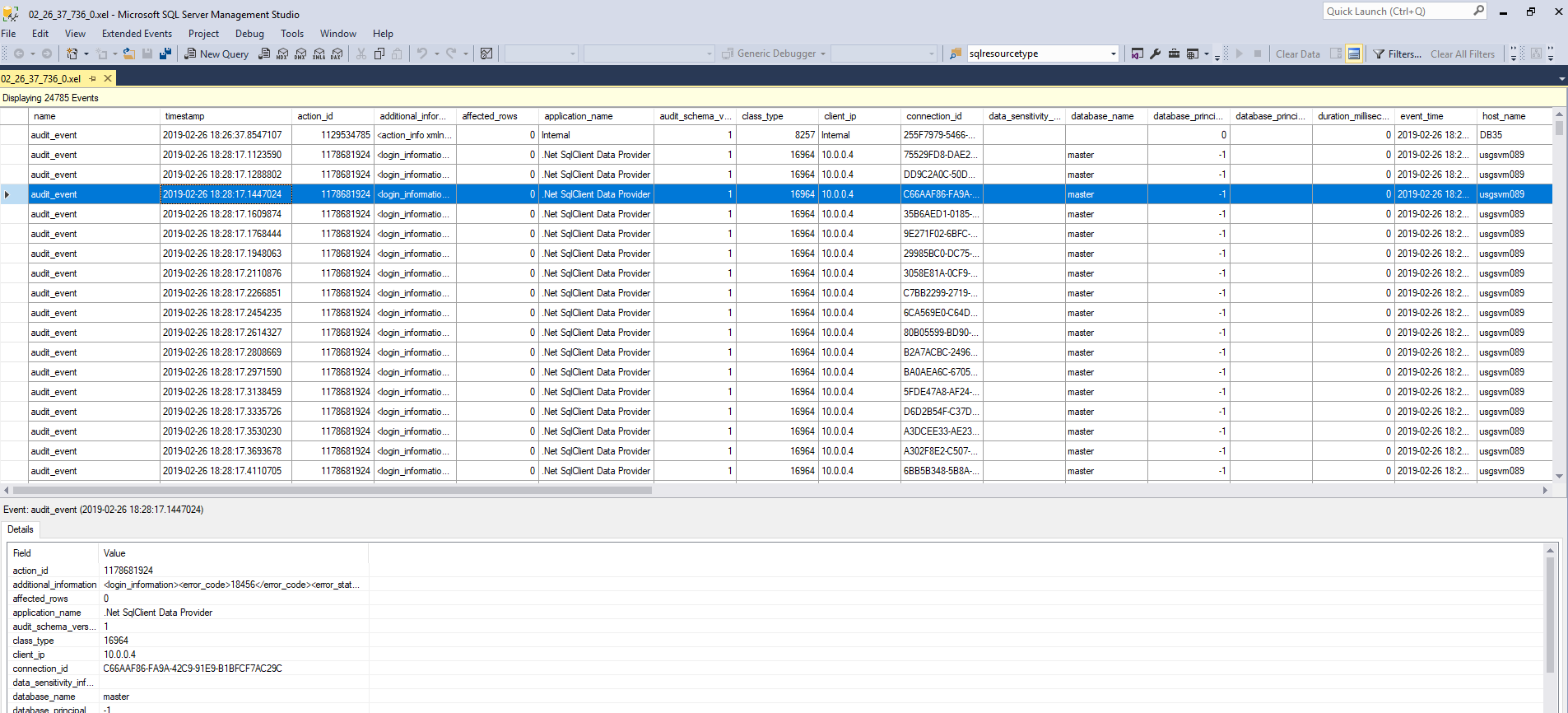
1. Run SQL query that targets these confidential columns.

-- Lookup 2014 fire events in the California area

SELECT \*

FROM dbo.userAccessAssignments

1. Login to the Azure portal and browse to your data warehouse instance. Navigate to the ‘Auditing’ pane under the security header and view the audit logs. 

1. To view queries targeting classified columns, you will need to use the Audit file viewer in SSMS.
   1. In the Azure portal, navigate to the storage account configured to store audit records for your data warehouse (usgsstaging##)
   2. Click on the ‘Blobs’ tile and then the ‘sqldbauditlogs’ container within it. Navigate to the folder containing the audit records for your warehouse: sqldbauditlogs > usgsserver## > usgsdataset >SqlDbAuditing\*
   3. Download the ‘.xel’ audit file to your local machine.
   4. Clicking on the file to open it in SSMS where you can find more details on the audit records.
   5. Add the ‘data\_sensitivity\_information’ column to the visible column list by right-clicking on an existing column, selecting ‘Choose columns’ and selecting ‘data\_sensitivity\_information’ 

**Restrict access to sensitive columns using column-level security**

In addition to monitoring access to sensitive columns in your data warehouse, you can restrict access to these columns. In this section, you will restrict access to classified columns to only users in the ‘manager’ role. The sample data warehouse comes configured with ‘manager’ and ‘employee’ database roles and two logins in each role.

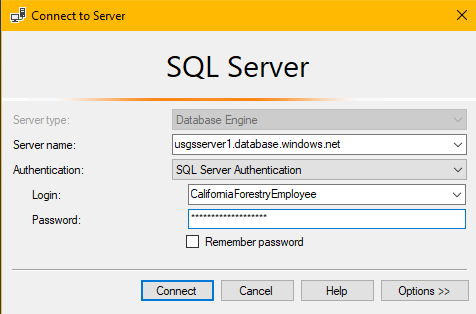
In SQL Server Management Studio (SSMS):

Right-click on your data warehouse and select ‘New query’ then run the following:

1. Restrict access to sensitive fire event columns for users in the employee role

-- Restrict access to sensitive columns for users in the employee role

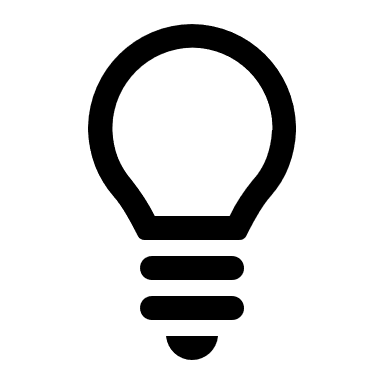
DENY SELECT ON prod.factFireEvents (USFSFireNumber, FireIncidentCode, AreaProtectionCategoryCode, FireManagementResponseCode, FireLongitudeDetail, FireLatitudeDetail) TO employee;

1. Login using credentials for the ‘CaliforniaForestryEmployee’ user. Make sure to replace ‘##’ with the deployment number you were assigned at launch.  
   Server name: usgsserver##.database.windows.net  
   Authentication: SQL Server Authentication  
   Username: CaliforniaForestryEmployee  
   Password: P@ssword##  
   
2. Execute the query below to test column-level security configuration. You should receive an error trying to select the confidential columns

-- Select the first 100 fire events in the database

SELECT TOP 100 \* from prod.factFireEvents

Bonus: configure Power BI report to access DW instance and login as CaliforniaForestryEmployee to test out column-level security

**Feedback**

Congratulations! You have completed the lab on data privacy and security for SQL DW.   
Was the module helpful? Any improvements you’d suggest?   
[Take this short 1 min survey to share your thoughts with us](https://forms.office.com/Pages/ResponsePage.aspx?id=v4j5cvGGr0GRqy180BHbR90qBpvyfe5Mkaxhz8QBg_hUOEhBTkE1WDJOSlYyWjhBUEhHTTVDWVlNRyQlQCN0PWcu).