Back up and restore your Azure SQL database

900 XP

41 min • Module • 8 Units

V V V V W 4.6 (232) Rate it

Beginner Solutions Architect Azure

Protect the data in your Azure SQL database and recover from data loss or corruption with backup and restore.

In this module, you will:

- Configure backup and retention of an Azure SQL database
- Restore an Azure SQL database

Start

Prerequisites

- Basic knowledge of the Azure SQL Database service
- Basic knowledge of Azure PowerShell

This module is part of these learning paths

Architect migration, business continuity, and disaster recovery in Azure

Introduction
2 min

Back up Azure SQL Database
7 min

Exercise - Configure backups for Azure SQL Database 7 min

Retain backup history with long-term retention policies 5 min

Exercise - Configure long-term retention policies
7 min

Recover data by restoring an Azure SQL database
5 min

Exercise - Recover data by restoring an Azure SQL database 7 min

Summary
1 min

200 XP

Back up Azure SQL Database

7 minutes

The retail organization that you work for uses Azure SQL Database to store the relational data for its enterprise resource planning (ERP) system. The company uses this system for all its accounting, customer relationship management, sales management, and corporate governance procedures. If this data is lost, the business will suffer huge losses and might even have to cease operations.

The board has given you responsibility for the protection of this data. You want to be sure that if a disaster happens, you can restore all the data up to the failure within 3 hours.

Here, you'll learn about Azure SQL Database backups and how to use them effectively.

Storage for Azure SQL Database backups

SQL Database automatically creates database backups. The backups are kept for 7 to 35 days. The retention time depends on the purchasing model and the service tier that you choose when you create your database. When the backups are complete, they're stored as blobs in a read-access geo-redundant storage (RA-GRS) account in your Azure subscription. To help protect against a datacenter outage, they're replicated to a paired datacenter.

Azure SQL Database uses SQL Server technology to make these types of backups:

- **Full backups** In a full backup, everything in the database and the transaction logs is backed up. SQL Database makes a full backup once a week.
- **Differential backups** In a differential backup, everything that changed since the last full backup is backed up. SQL Database makes a differential backup every 12 hours.
- **Transactional backups** In a transactional backup, the contents of the transaction logs are backed up. SQL Database makes a transaction log backup every 5 to 10 minutes. Transactional backups enable administrators to restore up to a specific time, which includes the moment before data was mistakenly deleted.

You can use these backups to:

- Restore an existing database.
- Restore a deleted database up to the time when it was deleted.
- Restore the database to an alternative location or region.
- Restore a database from a long-term backup by using long-term retention (LTR).

When a failure occurs, you might lose changes from up to 5 minutes ago, if the live transaction logs are lost. If the transaction logs are intact, you can restore up to the moment that the failure occurred.

Backups and service tiers

The default backup retention period is set to 7 days when you create a database. Later, you can change that period to a duration from 0 to 35 days. When you create a database by using the purchasing model based on Data Transaction Units (DTUs), the default retention period for that database depends on the service tier:

Service tier	Default retention period
Basic	1 week
Standard	5 weeks
Premium	5 weeks

How often do backups happen?

There are backups for point-in-time restore, and there are backups for long-term retention.

SQL databases fully support point-in-time restore. They automatically create full backups, differential backups, and transaction log backups. The first full backup is scheduled as soon as the database is created. It usually finishes within 30 minutes, but it might take longer if the database is of significant size.

After the first full backup, all further backups are scheduled automatically and managed silently in the background. The SQL Database service determines the exact timing of all database backups, because it balances the overall system workload. You can't change or disable the backup jobs.

Full backups for LTR are kept up to 10 years in Azure Blob storage accounts. You can configure the LTR policy to perform automatic weekly full backups. The storage of LTR backups depends on the frequency and the retention period that you choose.

Storage costs

Microsoft provides Azure backup services to create all-encompassing backups with a predictable pricing system that lets you easily keep track of any Azure data backup costs. Pricing for backing up SQL Server is based on paying for both instance costs (the data getting protected) and storage costs each month.

By default, 7 days of automated backups of your databases are copied to RA-GRS standard blob storage. The storage is used by weekly full backups, daily differential backups, and transaction log backups copied every 5 minutes. The size of the transaction log depends on the rate of change of the database.

A minimum storage amount equal to 100 percent of database size is provided at no extra charge. Additional consumption of backup storage is charged in gigabytes per month.

Benefits of using Azure backups

Azure backups offer the following benefits:

- You can reduce your infrastructure costs, because there are minimal upfront costs and minimal operational expenses.
- You can use a range of features to help ensure that your data is backed up, secure, and stored in a separate location from your database.
- You can store three copies of your data in three different locations in the primary Azure datacenter. You can store another three copies in an alternative remote Azure datacenter. This arrangement protects against all but the most severe disasters.
- Your data is encrypted before it leaves the source database, whether it's in transit or held in the Azure backup vault.

Check your knowledge

1. You've created a new database in Azure SQL Database. When will the first full backup run?

- At midnight in the local region
- Immediately

  Azure schedules a full backup to run as soon as the database is created.
- When you manually run the backup

2. Where are SQL Server backups stored, by default?

- In the on-premises location that you specify
- In a locally redundant storage account
- In a read-access geo-redundant storage account

  The default location for backups is a read-access geo-redundant storage account that's replicated to a paired datacenter.

Next unit: Exercise - Configure backups for Azure SQL Database

Continue

100 XP

Exercise - Configure backups for Azure SQL Database

7 minutes

This module requires a sandbox to completeA **sandbox** gives you access to Azure resources. Your Azure subscription will not be charged. The sandbox may only be used to complete training on Microsoft Learn. Use for any other reason is prohibited, and may result in permanent loss of access to the sandbox.

Activate sandbox

Although the default Azure SQL Database configuration includes automated backups, most organizations will modify the default setup to tailor it to their needs.

Now that you have planned a comprehensive backup strategy for Azure SQL Database and your company's ERP system, it's time to implement it.

Here, you'll create a database in Azure and then configure backups. You'll set the retention to 28 days, to ensure that you have 4 weeks of backups retained in accordance with your policy. You'll also add some content to the database.

Create a SQL Database server and database

Let's use the Azure CLI to create a SQL Database server and a database.

1. Run the following commands in Azure Cloud Shell to set up some variables for creation of the SQL Database server.

```
PowerShell
Copy

$serverName = "ERPServer-$(Get-Random)"
$location = $(Get-AzResourceGroup -ResourceGroupName [sandbox resource group name ]).location
$sqlAdmin = Get-Credential -credential dbadmin
```

   This step creates a server name with a random number at the end to ensure that it's globally unique. We'll refer to the server name `ERPServer-NNNN` through the exercises, but replace this with the name of your server that's generated here.

   This step also sets the location for your server to the location of the resource group. Finally, it sets the credentials that you'll use to access the database server. When you're prompted, enter a complex password of your choice.

2. Run the `New-AzSqlServer` command to create a SQL Database server to store the database.

```
PowerShell
Copy

New-AzSqlServer `
    -ResourceGroupName [sandbox resource group name ] `
    -Location $location `
    -ServerName $serverName `
    -SqlAdministratorCredentials $sqlAdmin
```

3. Run the `New-AzSqlDatabase` command to create a database.

```
PowerShell
Copy

New-AzSqlDatabase `
    -ResourceGroupName [sandbox resource group name ] `
    -ServerName $serverName `
    -DatabaseName sql-erp-db
```

Configure the database retention policy

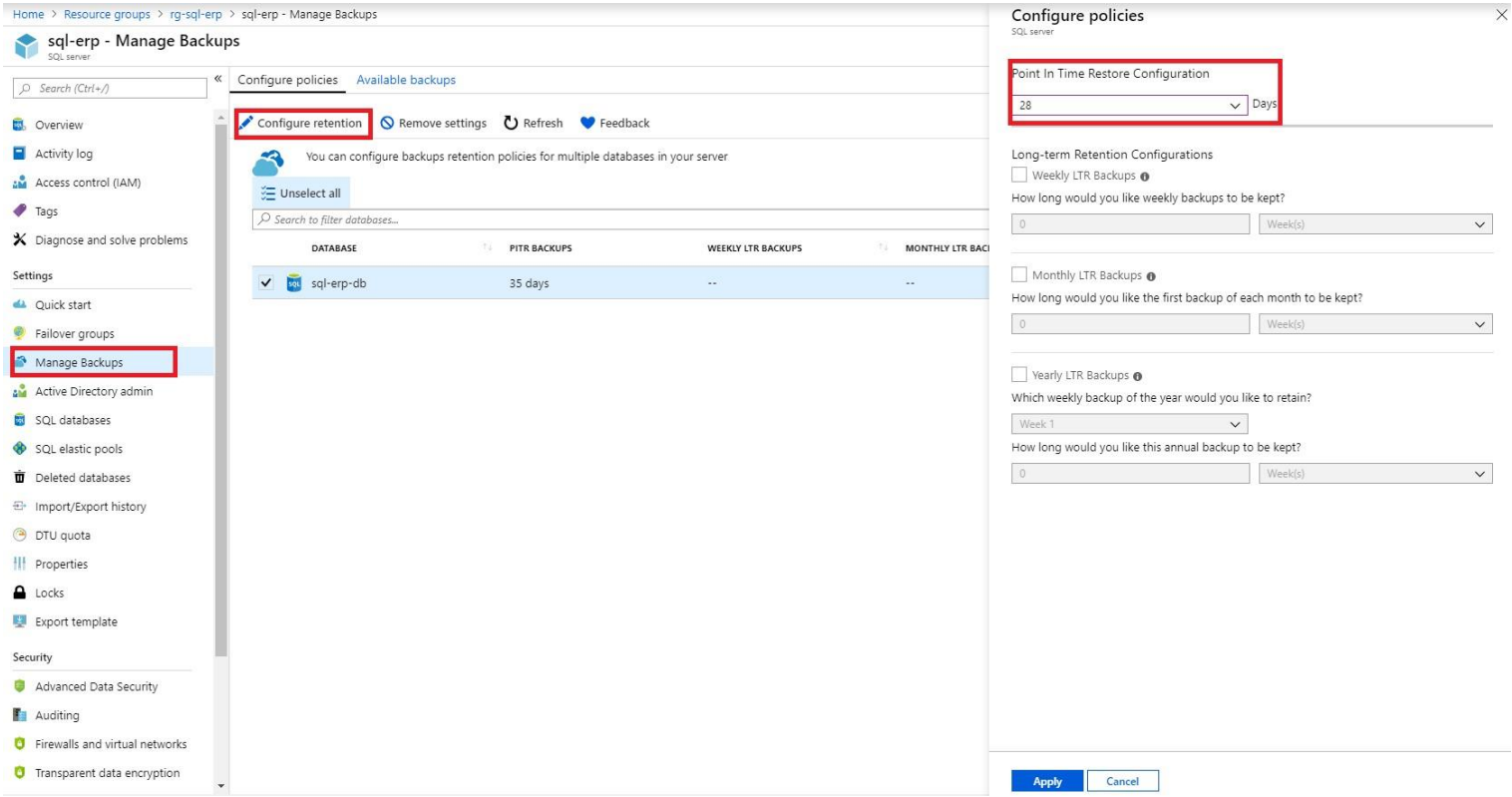
In the portal, you can examine the default retention policy and adapt it to your needs.

1. On the Azure portal menu or from the **Home** page, select **All resources**, and then select the **ERPServer-NNNN** database server that you created.

2. On the left in the **Settings** section, select **Manage Backups**

3. On the **Configure policies** tab, select the **sql-erp-db** database, and then select **Configure retention**

4. In the **Point In Time Restore Configuration** drop-down list, select **28**.



5. Select **Apply**, and then select **Yes**.

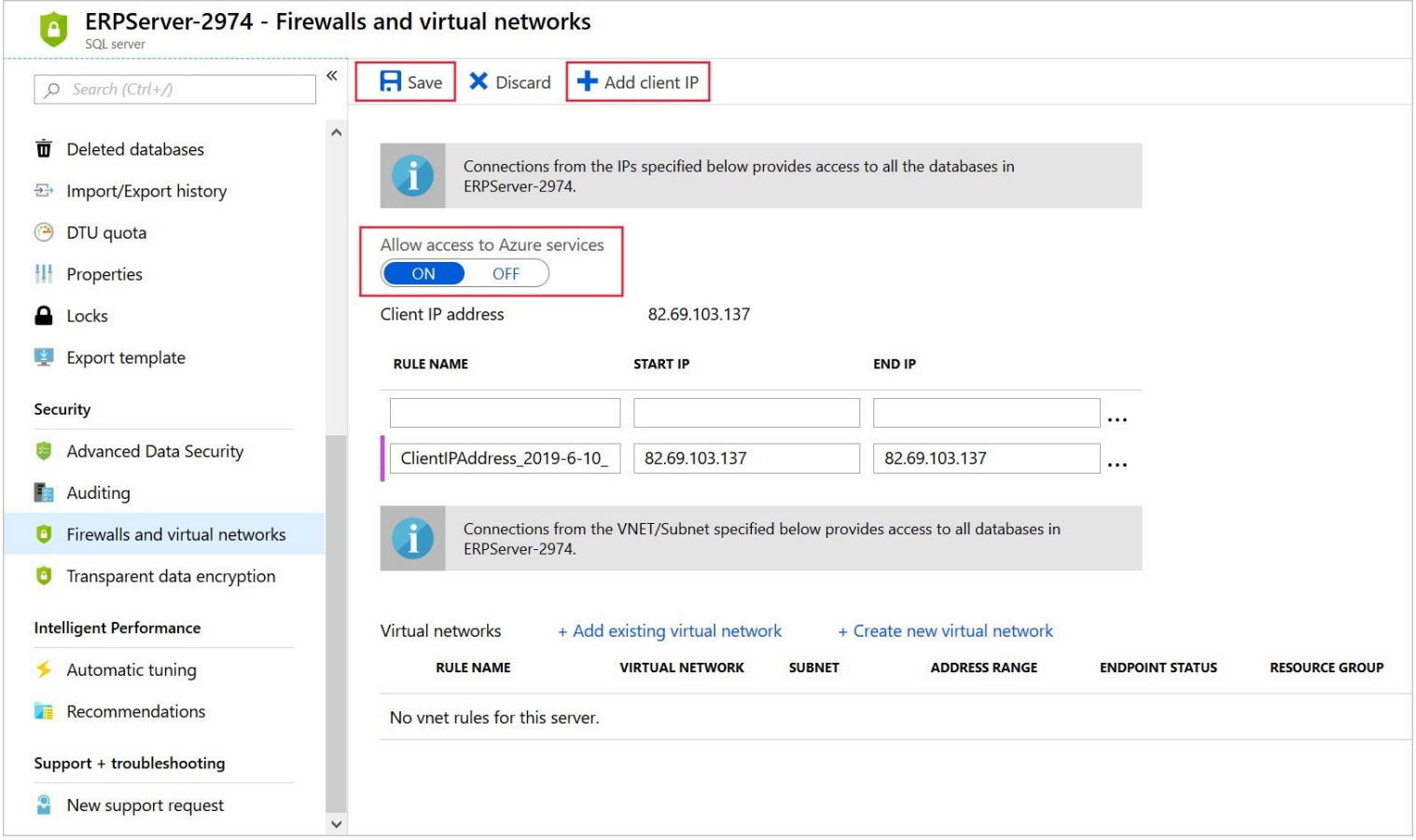
Allow network access to the database server

By default, Azure SQL Database blocks network access to the server. Let's enable both your IP address and Azure services to access the server so that we can run queries from Cloud Shell and the Azure portal. By adding your IP address, you can also connect directly from your local device.

1. On the left in the **Security** section, select **Firewalls and virtual networks**

2. At the top of the page, select **Add client IP**.

3. Under **Allow access to Azure services**, select **ON**.



4. Select **Save**. When the rule is saved, select **OK**.

Add data to the database

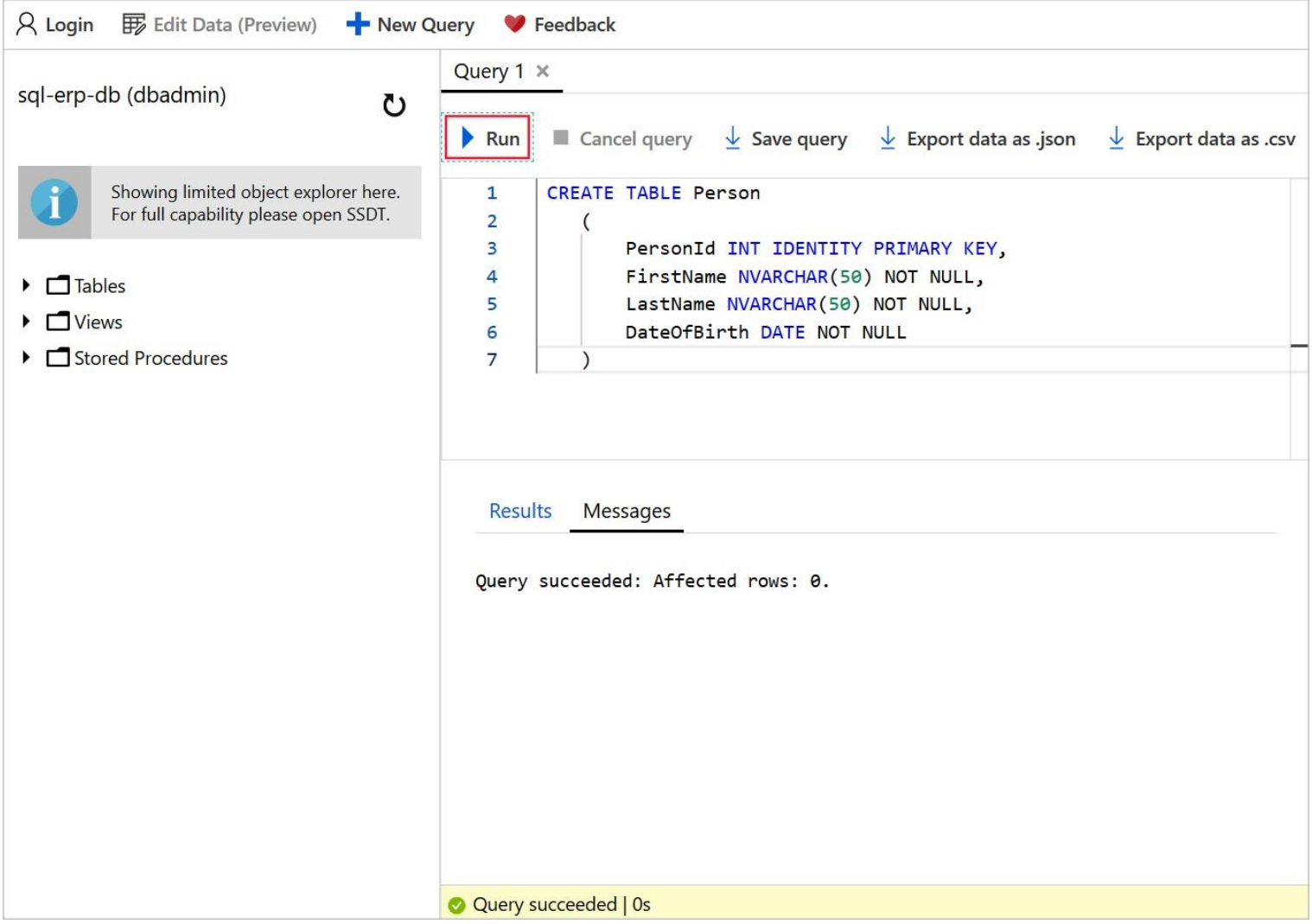
Now let's add a table and a sample record to the database. It's helpful to have some data in the database to validate that our backups and restores work later in the module.

1. In the **Settings** section, select **SQL databases**, and then select **sql-erp-database**
2. Select **Query editor**, and then sign in with the **dbadmin** credentials and the password that you specified for this account.
3. To create a table, in the **Query 1** window, enter this SQL command, and then select **Run**.

SQL

Copy

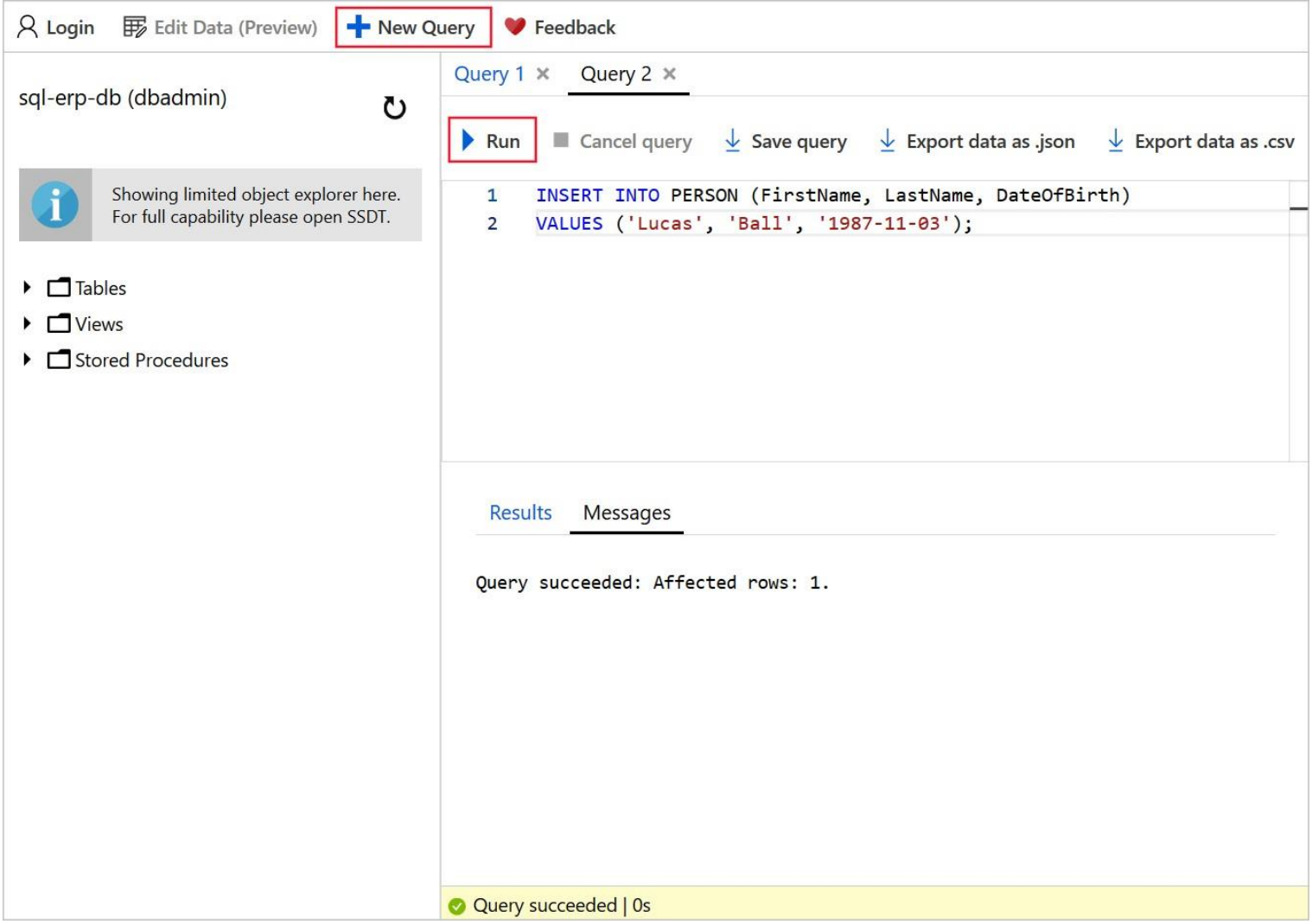
```
CREATE TABLE Person
(
    PersonId INT IDENTITY PRIMARY KEY,
    FirstName NVARCHAR(50) NOT NULL,
    LastName NVARCHAR(50) NOT NULL,
    DateOfBirth DATE NOT NULL
)
```



4. To add a record, select**+ New Query**. In the **Query 2** window, enter this SQL command, and then selec**Run**.

SQL Copy

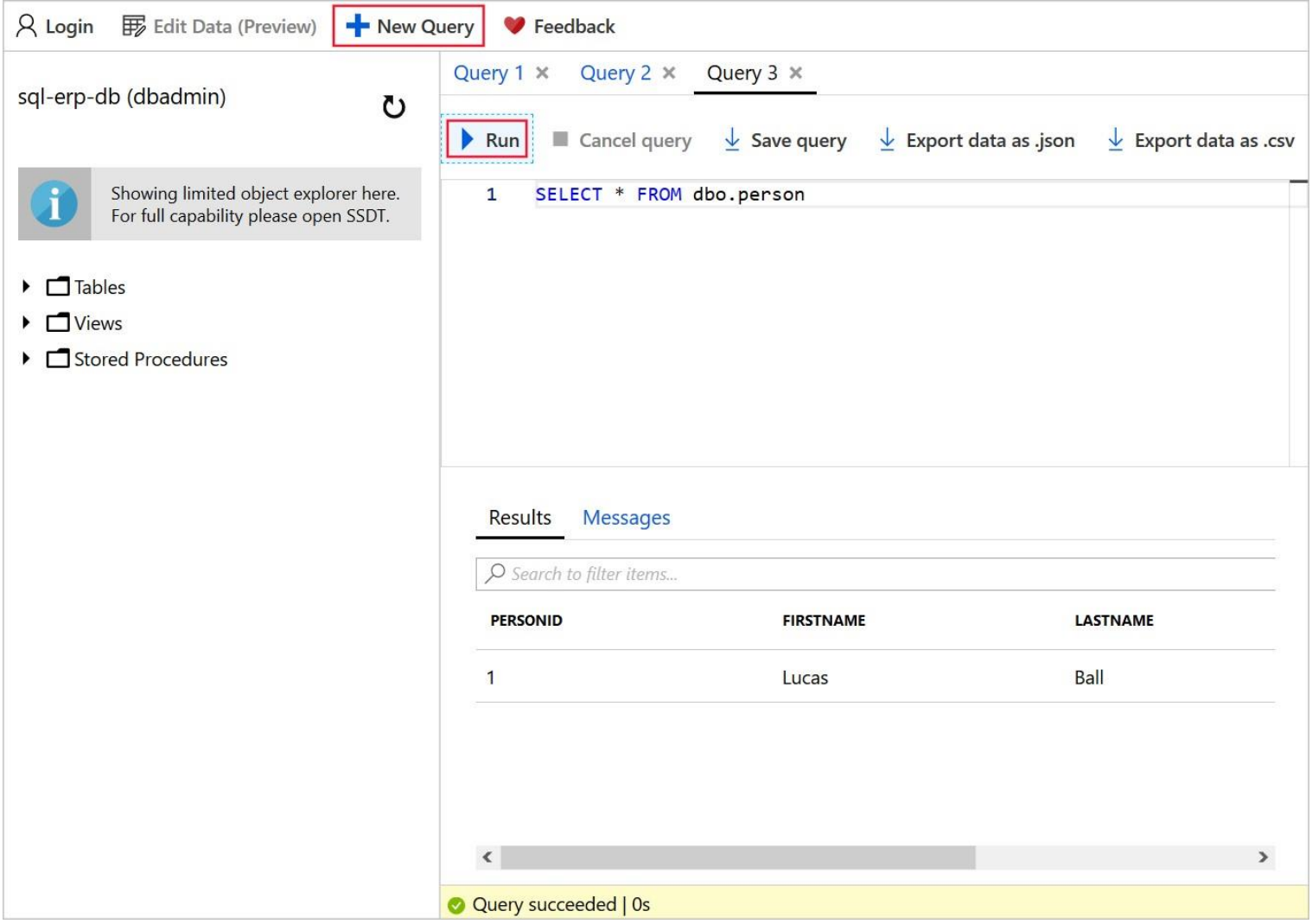
```
INSERT INTO PERSON (FirstName, LastName, DateOfBirth)
VALUES ('Lucas', 'Ball', '1987-11-03');
```



5. To query the database, select **+ New Query**. In the **Query 3** window, enter this SQL command, and then select **Run**.

```SQL
SELECT * FROM dbo.Person
```

The **Results** window displays the record for Lucas Ball.



You now have an Azure SQL database that's populated with data. And you've set up a retention policy to ensure that you have 4 weeks of backups immediately available for restore.

Next unit: Retain backup history with long-term retention policies

Continue

Exercise - Configure long-term retention policies

7 minutes

This module requires a sandbox to completeA **sandbox** gives you access to Azure resources. Your Azure subscription will not be charged. The sandbox may only be used to complete training on Microsoft Learn. Use for any other reason is prohibited, and may result in permanent loss of access to the sandbox.

Activate sandbox

Your retail organization must comply with data protection regulations in your jurisdiction. You need to keep all data for 2 years, and you want to keep one backup each month for 6 months. You've been asked to configure a long-term retention policy in Azure SQL Database to meet these requirements.

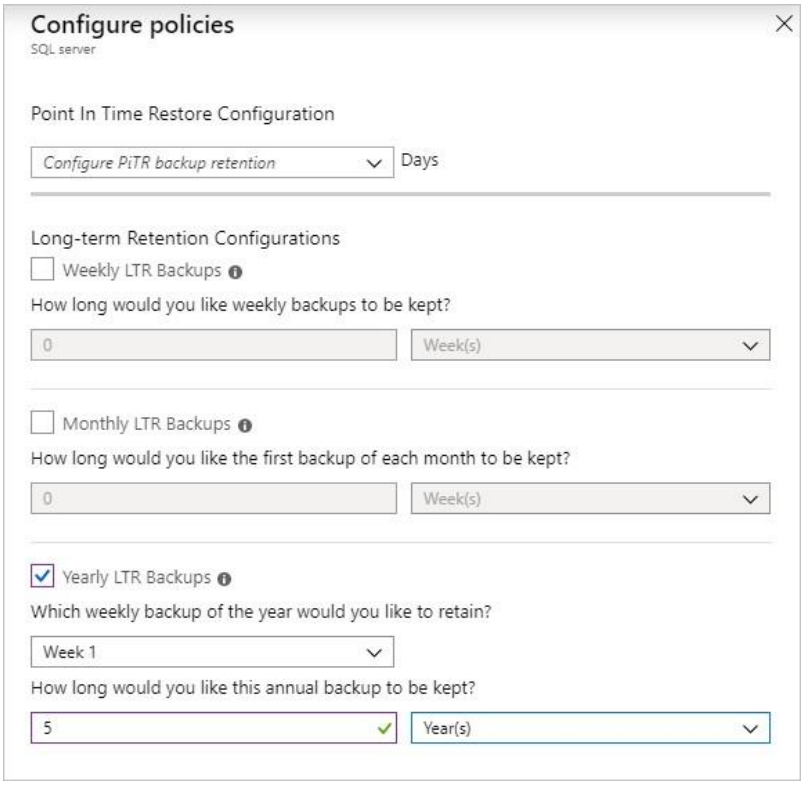
Here, you'll use the Azure portal to set up a policy and then check it in PowerShell. You need to set up the following retention policy to meet your regulatory requirements:

Weekly	8 weeks
Monthly	12 months
Yearly	5 years

Use the Azure portal to configure long-term retention

Let's start by configuring the 5-year retention by using the portal.

1. On the Azure portal menu or from the **Home** page, select **All resources**, and then select **ERPServer-NNNN**
2. Under **Settings**, select **Manage Backups**
3. In the list of databases, select **sql-erp-db**, and then select **Configure retention**
4. Select **Yearly LTR Backups**
5. In the **How long would you like this annual backup to be kept?** controls, enter the value **5** and select **Year(s)**.



6. Select **Apply**, and then select **Yes**.

Use PowerShell to configure long-term retention

You can also configure long-term retention policies by using PowerShell. Let's configure the remainder of the policy this way.

1. In Azure Cloud Shell, run this command to set a variable to the value of your SQL Server instance.

```
PowerShell
$sqlserver =Get-AzSqlServer
```

2. To view long-term retention policies for the database server, run this command.

```
PowerShell
Get-AzSqlDatabase  `
    -ResourceGroupName  [sandbox resource group name ] `
    -ServerName  $sqlserver .ServerName `
    |  Get-AzSqlDatabaseLongTermRetentionPolicy
```

   This step will output the retention policies for all databases on the server. In this case, you should see one policy for the **master** database, and one policy for the **sql-erp-db** database.

3. Run this command to view the long-term retention policy for the **sql-erp-db** database.

```
PowerShell
Get-AzSqlDatabaseBackupLongTermRetentionPolicy   `
    -ServerName  $sqlserver .ServerName `
    -DatabaseName  sql-erp -db `
    -ResourceGroupName  [sandbox resource group name ]
```

4. Now let's configure the rest of the policy to meet the requirements specified earlier. To configure a long-term retention policy via PowerShell, run this command.

```
PowerShell
Set-AzSqlDatabaseBackupLongTermRetentionPolicy   `
    -ServerName  $sqlserver .ServerName `
    -DatabaseName  sql-erp -db `
    -ResourceGroupName  [sandbox resource group name ] `
    -WeeklyRetention  P8W `
    -MonthlyRetention  P12M `
    -YearlyRetention  P5Y `
    -WeekOfYear  1
```

5. To check that the new policy has been applied, run this command again.

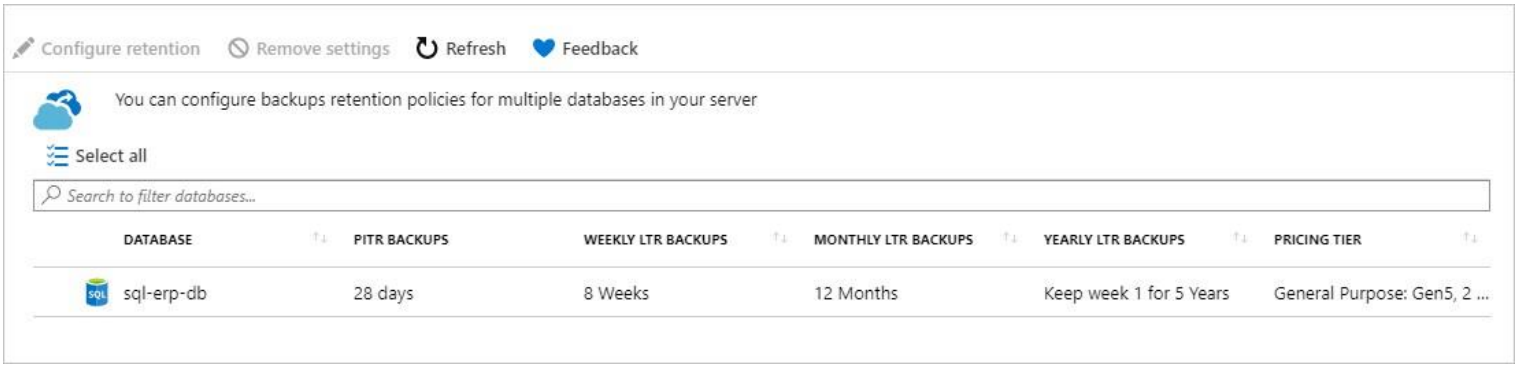
```
powerShell
Get-AzSqlDatabaseBackupLongTermRetentionPolicy   `
    -ServerName  $sqlserver .ServerName `
    -DatabaseName  sql-erp -db `
    -ResourceGroupName  [sandbox resource group name ]
```

   You should see the following policy configured. It enables a weekly retention of 8 weeks, a monthly retention of 12 months, and a yearly retention of 5 years for the first backup of the year.

```
output
ResourceGroupName : [sandbox resource group name]
ServerName        : erpserver-25078
DatabaseName      : sql-erp-db
WeeklyRetention   : P8W
MonthlyRetention  : P12M
YearlyRetention   : P5Y
WeekOfYear        : 1
Location          :
```

6. You can also confirm this in the portal. Open the Azure portal menu or from the **Home** page, select **All resources**, and then select **ERPServer**.

7. Under **Settings**, select **Manage Backups** In the list of databases, check the **sql-erp-db** long-term retention properties.



You've now configured a retention policy and validated that the policy meets your organizational and regulatory requirements.

Next unit: Recover data by restoring an Azure SQL database

Continue

Exercise - Recover data by restoring an Azure SQL database

7 minutes

This module requires a sandbox to completeA **sandbox** gives you access to Azure resources. Your Azure subscription will not be charged. The sandbox may only be used to complete training on Microsoft Learn. Use for any other reason is prohibited, and may result in permanent loss of access to the sandbox.

Activate sandbox

Trial restores are a key component of any disaster recovery strategy.

You want to familiarize yourself with the steps to restore a backed-up database to a specific point in time, in case it becomes necessary. You also want to investigate how long a restore operation will take so you can plan for this in your guidance for your organization.

Here, you'll perform a restore of from automated Azure SQL Database backups.

Confirm that backups are active

It can take up to 15 minutes for the first successful backup to finish. We need to make sure that backups are running before we continue the exercise.

1. Run the following PowerShell command in Azure Cloud Shell to validate that continuous backups are running.

```
PowerShell
Copy
Get-AzSqlDatabaseRestorePoint  `
    -ResourceGroupName  [sandbox resource group name ] `
    -DatabaseName  sql-erp-db `
    -ServerName  $sqlserver.ServerName
```

You should see output similar to the following if your backups are running. If the command returns no value, a backup hasn't started yet. Rerun this command in a couple of minutes.

```
output
Copy
ResourceGroupName        : [sandbox resource group name]
ServerName               : ERPServer-53903
DatabaseName             : sql-erp-db
Location                 : East US
RestorePointType         : CONTINUOUS
RestorePointCreationDate :
EarliestRestoreDate      : 9/24/19 4:21:21 PM
RestorePointLabel        :
```

`RestorePointType` is `CONTINUOUS`, indicating that backups are automatically happening. `EarliestRestoreDate` indicates the timestamp of the first backup. With backups in place, let's continue the exercise.

Drop a table from the database

Let's start by simulating a mistaken database modification.

1. On the Azure portal menu or from the **Home** page, select **All resources**, and then select the **sql-erp-db** database.
2. Select **Query editor**, and then sign in with the **dbadmin** user and the password that you specified for this account.
3. Let's drop the **Person** table that we created earlier. In a new query window, run this command.

```
SQL
Copy
DROP TABLE Person
```

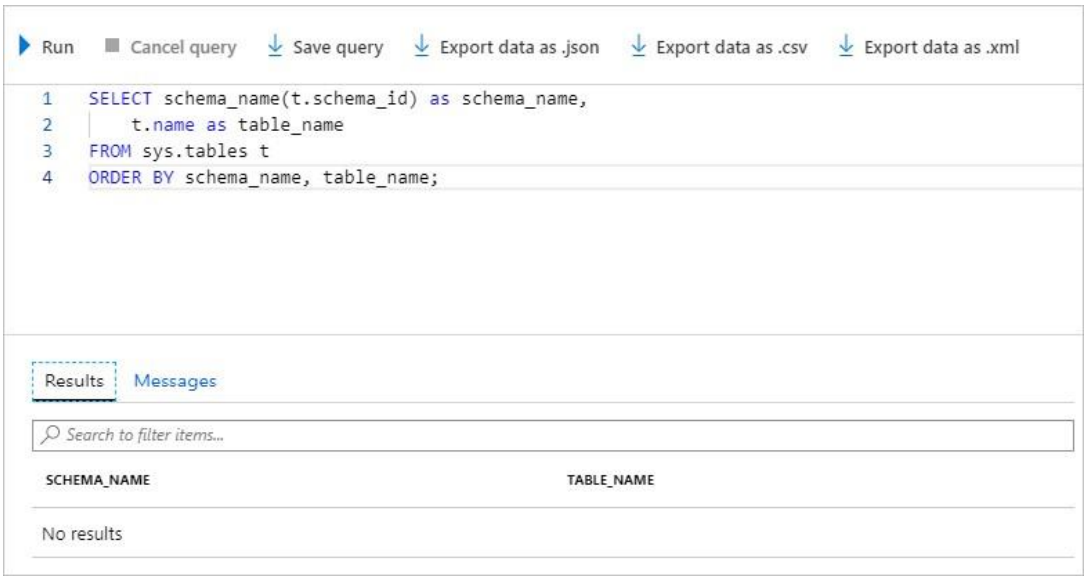
4. To check the tables in the database, select **New Query**. Then in the **Query 2** window, run this command to list all tables in the database.

SQL

Copy

```
SELECT schema_name(t.schema_id)  as schema_name,
    t.name  as table_name
FROM sys.tables t
ORDER BY schema_name, table_name;
```

You should see **No results** returned, because we deleted the **Person** table.

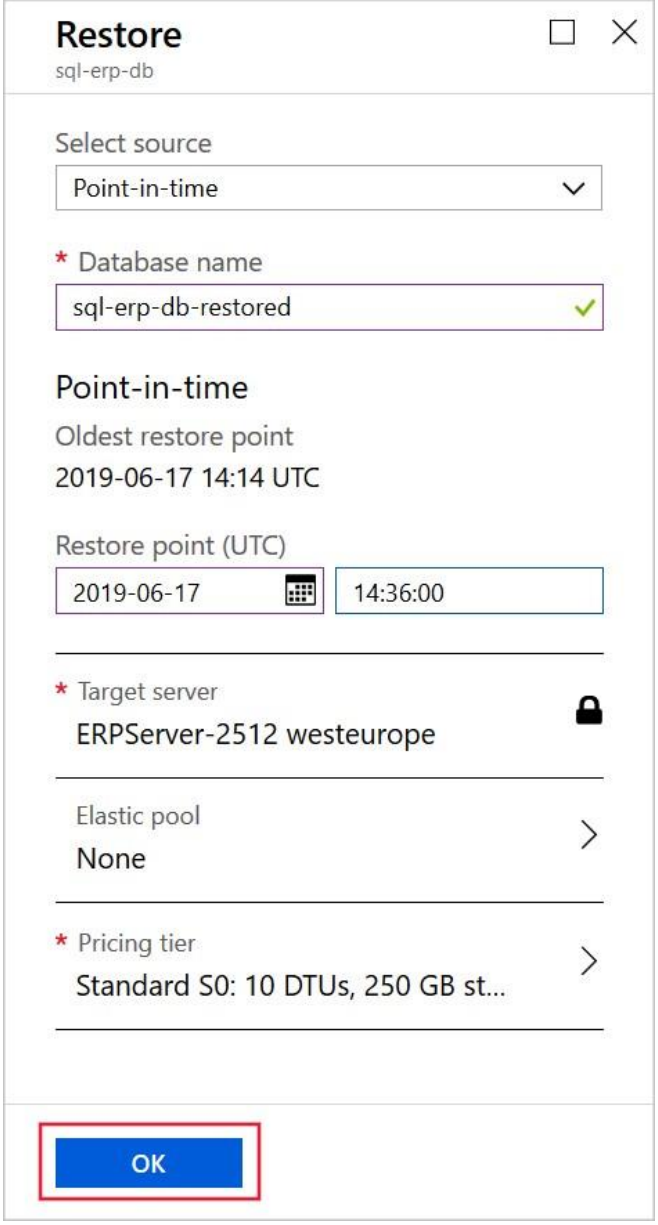


Run a point-in-time restore

The **Person** table was mistakenly deleted. Now, let's restore the database to its previous state.

1. On the Azure portal menu or from the **Home** page, select **All resources**, and then select the **sql-erp-db** database.
2. At the top of the **Overview** page, select **Restore**.
3. Complete the **Restore** page with these values, and then select **OK**.

Setting	Value
Select source	Point-in-time
Database name	sql-erp-db-restored
Restore point	Select a time 10 minutes ago, before you dropped the **Person** table
Target server	ERPServer
Elastic pool	None
Pricing tier	Default value



The database restore will take several minutes.

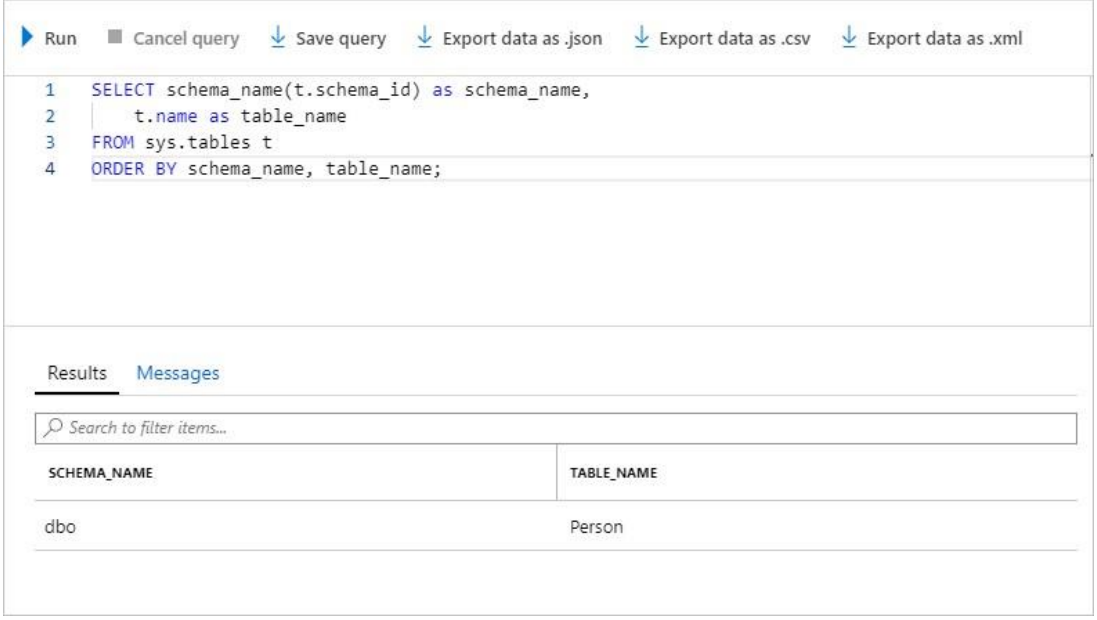
View the restored database

The restored database should contain the **Person** table. You can check that in the portal.

1. In the Azure portal menu or from the **Home** page, select **All resources**, and then select the **sql-erp-db-restored** database.
2. Select **Query editor**, and then sign in with the **dbadmin** user and the password that you specified for this account.
3. To check the tables in the database, in the **Query 1** window, run this command.

```sql
SELECT schema_name(t.schema_id)  as schema_name,
    t.name as table_name
FROM sys.tables t
ORDER BY schema_name, table_name;
```

The **Person** table should now be present.



4. Confirm that the data is in the table by running this command.

```
SQL
Copy
SELECT * FROM Person
```

You should see the data that you entered previously.

You've now seen how you can restore a database if something unintended happens to the data. You've familiarized yourself with the restore process. You can now assure your organization that your backup and restore procedures are properly defined.

Next unit: Summary

Continue

100 XP

Introduction

2 minutes

Imagine you work for a retail organization that has recently moved its enterprise resource planning (ERP) database to Azure SQL Database. This database holds critical operational data, and it must be protected from data loss or corruption. You need to set up backup and retention policies to ensure your organization can recover from disasters. You must be familiar with the steps and options to recover, in case the need arises.

In this module, you'll see how to configure backup and long-term data retention policies. You'll also learn about database restore procedures.

Learning objectives

At the end of this module, you will be able to:

- Configure backup and retention of a single Azure SQL database.
- Restore a single Azure SQL database.

Prerequisites

- Basic knowledge of the Azure SQL Database service
- Basic knowledge of Azure PowerShell

Next unit: Back up Azure SQL Database

Continue

Recover data by restoring an Azure SQL database

5 minutes

Testing and validating the restore capability and procedures is a critical piece of a recovery strategy. By testing the restore process, you validate that your backups are successful. You also familiarize yourself with the process and options available for recovering a database. This helps to ensure a quick and successfu recovery of data when needed.

Here, you'll learn how to restore an Azure SQL Server database from automated database backups.

What you can restore

Automated backups in Azure SQL Database copy databases to blobs in read-access geo-redundant storage (RA-GRS) accounts on the schedule that you specify. If you want to restore one of these backups, you must create a new database to contain the restored data. You can't restore a database over an existing database.

You can create the database on the same server where the backup was taken or on another server with these options:

- Create a new database on the same SQL Database server recovered to a specified point in time within the retention period.
- Create a database on the same SQL Database server recovered to the deletion time for a deleted database.
- Create a new database on any SQL Database server in the same region recovered to the point of the most recent backups.
- Create a new database on any SQL Database server in any other region recovered to the point of the most recent replicated backups.

How restore works

To complete a restore, Azure copies the database from the storage account to the Azure SQL Database server that you specify. In a point-in-time restore, SQL Database follows that by applying transaction logs to the restored database, up to the time you chose.

The length of this process varies widely. It depends on the database size, the transaction logs, network bandwidth, and the number of concurrent restore operations. Most restore operations finish in less than 12 hours.

The only way to know how long your restore operation takes is to perform a trial restore. It's good idea to perform trial restores occasionally to time them and to ensure your complete backup and restore strategy works as you expect.

Perform a point-in-time restore

You can perform database restores by using the Azure portal, PowerShell, or the Azure CLI. If you're performing a point-in-time restore on the original SQL Database server, you can choose:

- **Database replacement** If you want to replace the original database with the restored one, make sure you specify the same compute size and service tier as the original. Then rename the original database and give the restored database the original name by using T-SQL `ALTER DATABASE` commands.
- **Data recovery**. If you want to retrieve data from the restored database to mitigate an error, you don't need to rename the original and restored databases. Instead, use T-SQL commands to extract the data that you need from the restored database. Then insert the data into the original database.

Both of these options begin with the restoration of a database backup from storage. To recover in the Azure portal, select the **Restore** button on the database overview page, and then specify the time to restore to.

In PowerShell, use the `Restore-AzSqlDatabase` cmdlet to execute restorations. In the Azure CLI, use the `az sql db restore` command.

Restore a deleted database

If a database is mistakenly deleted, you can restore it from backup to the deletion time by using the Azure portal or PowerShell.

In the portal, go to the database server's **Overview** page. Then, in the **Operations** area, select **Deleted databases** You can specify a point in time up to the deletion, and then select **OK** to recover.

Perform a geo-restore

Azure SQL Database automatically replicates backed-up databases to datacenters in other regions. If the database in the original region is unavailable, for example because of a datacenter outage, you can restore from one of these replicated backup copies. You can restore up to the point in time when the backup was made. The latest backup might not have fully replicated to your region, so you might lose some recent changes.

To perform a geo-restore in the Azure portal, add a new database to an Azure SQL Database server. Then in the **Select source** drop-down list, select **Backup**, and choose the backup to restore from.

Next unit: Exercise - Recover data by restoring an Azure SQL database

Continue

Retain backup history with long-term retention policies

5 minutes

Companies need to keep backups for months or years for regular administrative protection, such as to restore accidentally deleted data.

For example, data protection laws in at least one country where your retail organization operates require you to keep records of all customer transactions for 5 years. You need to ensure that data in Azure SQL Database, which underpins your enterprise resource planning (ERP) system, is kept for at least that long.

Here, you'll learn about long-term retention policies in Azure SQL Database and how to use them when you need backups to be kept for more than 35 days.

Long-term backup retention policies

Azure SQL Database automatic backups remain available to restore for up to 35 days. This period is enough for the purposes of day-to-day administration. But sometimes you might need to retain data for longer periods. For example, data protection regulations in your local jurisdiction might require you to keep backups for several years.

For these requirements, use the long-term retention (LTR) feature. This way, you can store Azure SQL Database backups in read-access geo-redundant storage (RA-GRS) blobs for up to 10 years. If you need access to any backup in LTR, you can restore it as a new database by using either the Azure portal or PowerShell.

How SQL Database long-term retention works

LTR takes the backups that are automatically made for point-in-time recovery and copies them to different blobs. This copy operation runs in the background at low priority to ensure that there's no impact on performance.

These backups don't happen by default. You must configure a policy to start and manage them.

How to write a long-term retention policy

The long-term retention policy sets how frequently an automatic backup will be copied for long-term retention. You specify this frequency with letters:

- `W`: Specifies that one full backup each week will be coped to long-term retention.
- `M`: Specifies that one full backup from the first week of each month will be copied to long-term retention.
- `Y`: Specifies that one full backup each year will be coped to long-term retention.

If you use `Y` for yearly backups, you can specify the week of the year when that backup is copied by using the `WeekOfYear` parameter.

For each policy letter, you use numbers to indicate how long the backup should be retained. For example, to keep the weekly backup for 10 weeks, use `W=10`. To keep the annual backup for 3 years, use `Y=3`.

Example long-term retention policies

You can combine weekly, monthly, and yearly retention values to create a flexible policy. For example:

- `W=0, M=0, Y=5, WeekOfYear=3`

  This policy retains the third full backup of each year for 5 years.

- `W=0, M=10, Y=0`

  This policy retains the first full backup of each month for 10 months.

- `W=12, M=0, Y=0`

  This policy retains each weekly full backup for 12 weeks.

- `W=4, M=12, Y=10, WeekOfYear=1`

  This policy retains each weekly backup for 4 weeks. It also retains the first full backup of each month for 12 months. The first full backup taken in the first week of each year is retained for 10 years.

Setting retention policies in PowerShell

In PowerShell, you can examine a long-term retention policy by using this command:

```PowerShell
Get-AzSqlDatabase `
    -ResourceGroupName <ResourceGroupName> `
    -ServerName <ServerName> `
    | Get-AzSqlDatabaseLongTermRetentionPolicy
```

To configure the policy, use the `Set-AzSqlDatabaseBackupLongTermRetentionPolicy` cmdlet. When you specify these policies in PowerShell, you must use ISO 8601 duration values. For example, to specify the `W=10` policy, pass the string `P10W` to the `-WeeklyRetention` parameter. To specify the `Y=3` policy, pass the string `P3Y` to the `--YearlyRetention` parameter.

```PowerShell
Set-AzSqlDatabaseBackupLongTermRetentionPolicy   `
    -ServerName <ServerName> `
    -DatabaseName <DatabaseName> `
    -ResourceGroupName <ResourceGroupName> `
    -WeeklyRetention P10W `
    -YearlyRetention P3Y `
    -WeekOfYear 1
```

Next unit: Exercise - Configure long-term retention policies

Continue