Azure Windows VM with Database Partitioning - Complete Guide

Table of Contents

Part 1: Creating a Windows Virtual Machine in Azure

Prerequisites

- Active Azure subscription
- Azure account with appropriate permissions

Step 1: Sign in to Azure Portal

1. Navigate to https://portal.azure.com
2. Sign in with your Azure credentials

Step 2: Create a New Virtual Machine

1. **Navigate to Virtual Machines**
   - Click on "Create a resource" in the left sidebar
   - Search for "Virtual Machine"
   - Click "Create" → "Virtual Machine"
2. **Configure Basic Settings Subscription & Resource Group:**
   - Subscription: Select your subscription
   - Resource Group: Create new or select existing (e.g., "rg-ecommerce-demo")

   Instance Details:
   - Virtual machine name: `vm-windows-dataserver`
   - Region: Choose closest region (e.g., East US)
   - Availability options: No infrastructure redundancy required (for demo)
   - Security type: Standard

- Image: `Windows Server 2022 Datacenter: Azure Edition - x64 Gen2`
- Size: `Standard_D4s_v3` (4 vcpus, 16 GiB memory) - recommended for database work

Administrator Account:

- Username: `azureadmin` (or your preferred username)
- Password: Create a strong password (min 12 characters)
- Confirm password

Inbound Port Rules:

- Public inbound ports: Allow selected ports
- Select inbound ports: RDP (3389)

3. **Configure Disks**
   - Click "Next: Disks"
   - OS disk type: `Premium SSD` (recommended for database)
   - Encryption type: Default
   - Click "Next: Networking"
4. **Configure Networking**
   - Virtual network: Create new or use existing
   - Subnet: default
   - Public IP: Create new
   - NIC network security group: Basic
   - Public inbound ports: Allow RDP (3389)
   - Click "Next: Management"
5. **Management Settings**
   - Enable auto-shutdown (optional): Configure if desired
   - Click "Review + create"
6. **Review and Create**
   - Review all settings
   - Click "Create"
   - Wait 3-5 minutes for deployment to complete

Step 3: Connect to Your Windows VM

1. **Get Connection Details**
   - Navigate to your VM in Azure Portal

- Click "Connect" → "RDP"
- Click "Download RDP File"

2. **Connect via RDP**
    - Open the downloaded RDP file
    - Click "Connect"
    - Enter your username and password
    - Accept any certificate warnings
    - You're now connected to your Windows VM!

Part 2: Installing Azure Data Studio

Step 1: Download Azure Data Studio

1. **Open Browser in VM**
    - Open Microsoft Edge or Internet Explorer
    - Navigate to: https://docs.microsoft.com/en-us/sql/azure-data-studio/download
2. **Download Installer**
    - Click on "Windows User Installer (recommended)"
    - Save the file to Downloads folder

Step 2: Install Azure Data Studio

1. **Run Installer**
    - Navigate to Downloads folder
    - Double-click `azuredatastudio-windows-user-setup-<version>.exe`
2. **Installation Steps**
    - Accept the License Agreement
    - Choose installation location (default is fine)
    - Select "Create a desktop icon" (recommended)
    - Click "Install"
    - Wait for installation to complete
    - Click "Finish"

Step 3: Install SQL Server (for local testing)

1. **Download SQL Server Express**
    - Navigate to: https://www.microsoft.com/en-us/sql-server/sql-server-downloads
    - Click "Download now" under Express edition
    - Run the installer
2. **Install SQL Server**
    - Choose "Basic" installation type
    - Accept license terms
    - Choose installation location
    - Click "Install"
    - Wait for installation to complete
    - Note the connection string provided

Step 4: Configure Azure Data Studio Connection

1. **Launch Azure Data Studio**
    - Double-click the desktop icon
2. **Create New Connection**
    - Click "New Connection" or use Ctrl+Shift+C
    - Enter connection details:
        - Connection type: `Microsoft SQL Server`
        - Server: `localhost` or `.\SQLEXPRESS`
        - Authentication type: `Windows Authentication`
        - Database: `<Default>`
        - Server group: `<Default>`
    - Click "Connect"

Part 3: Understanding Database Partitioning

What is Partitioning?

Database partitioning divides large tables into smaller, manageable pieces called partitions. Each partition stores a subset of the data based on specific criteria.

Benefits of Partitioning

1. **Improved Query Performance**: Queries scan only relevant partitions
2. **Better Manageability**: Easier to manage large tables
3. **Enhanced Data Loading**: Load data into specific partitions
4. **Simplified Maintenance**: Perform maintenance on individual partitions
5. **Improved Scalability**: Distribute data across multiple filegroups

Types of Partitioning

1. **Range Partitioning**: Partition by value ranges (most common)
2. **List Partitioning**: Partition by specific value lists
3. **Hash Partitioning**: Partition using hash function
4. **Composite Partitioning**: Combination of methods

Partitioning Components in SQL Server

1. **Partition Function**: Defines how data is divided
2. **Partition Scheme**: Maps partitions to filegroups
3. **Partitioned Table**: Table using the partition scheme

Part 4: E-commerce Database Partitioning Example

Scenario

We'll create an e-commerce database with partitioned tables for:

- Orders (partitioned by order date - yearly)
- Order Items (partitioned by order date - yearly)
- Customer Activity Logs (partitioned by activity date - monthly)

Step 1: Create the E-commerce Database

sql

```
-- Create new database
CREATE DATABASE EcommerceDB;
GO

USE EcommerceDB;
GO
```

Step 2: Create Filegroups

Filegroups allow storing partitions on different physical storage.

sql

```sql
-- Add filegroups for yearly order partitions
ALTER DATABASE EcommerceDB
ADD FILEGROUP FG_Orders_2022;
GO

ALTER DATABASE EcommerceDB
ADD FILEGROUP FG_Orders_2023;
GO

ALTER DATABASE EcommerceDB
ADD FILEGROUP FG_Orders_2024;
GO

ALTER DATABASE EcommerceDB
ADD FILEGROUP FG_Orders_2025;
GO

ALTER DATABASE EcommerceDB
ADD FILEGROUP FG_Orders_Future;
GO

-- Add files to filegroups
ALTER DATABASE EcommerceDB
ADD FILE
(
    NAME = Orders_2022_Data,
    FILENAME = 'C:\SQLData\Orders_2022.ndf',
    SIZE = 100MB,
    FILEGROWTH = 50MB
) TO FILEGROUP FG_Orders_2022;
GO

ALTER DATABASE EcommerceDB
ADD FILE
(
    NAME = Orders_2023_Data,
    FILENAME = 'C:\SQLData\Orders_2023.ndf',
    SIZE = 100MB,
    FILEGROWTH = 50MB
) TO FILEGROUP FG_Orders_2023;
GO

ALTER DATABASE EcommerceDB
ADD FILE
(
    NAME = Orders_2024_Data,
```

```
    FILENAME = 'C:\SQLData\Orders_2024.ndf',
    SIZE = 100MB,
    FILEGROWTH = 50MB
) TO FILEGROUP FG_Orders_2024;
GO

ALTER DATABASE EcommerceDB
ADD FILE
(
    NAME = Orders_2025_Data,
    FILENAME = 'C:\SQLData\Orders_2025.ndf',
    SIZE = 100MB,
    FILEGROWTH = 50MB
) TO FILEGROUP FG_Orders_2025;
GO

ALTER DATABASE EcommerceDB
ADD FILE
(
    NAME = Orders_Future_Data,
    FILENAME = 'C:\SQLData\Orders_Future.ndf',
    SIZE = 100MB,
    FILEGROWTH = 50MB
) TO FILEGROUP FG_Orders_Future;
GO
```

Step 3: Create Partition Function

The partition function defines the boundary values for splitting data.

sql

```
-- Create partition function for orders by year
CREATE PARTITION FUNCTION PF_OrdersByYear (DATETIME2)
AS RANGE RIGHT FOR VALUES
(
    '2023-01-01',  -- Partition 1: < 2023-01-01
    '2024-01-01',  -- Partition 2: 2023-01-01 to < 2024-01-01
    '2025-01-01',  -- Partition 3: 2024-01-01 to < 2025-01-01
    '2026-01-01'   -- Partition 4: 2025-01-01 to < 2026-01-01
                   -- Partition 5: >= 2026-01-01
);
GO
```

Step 4: Create Partition Scheme

The partition scheme maps the partition function to filegroups.

```sql
-- Create partition scheme
CREATE PARTITION SCHEME PS_OrdersByYear
AS PARTITION PF_OrdersByYear
TO (FG_Orders_2022, FG_Orders_2023, FG_Orders_2024,
    FG_Orders_2025, FG_Orders_Future);
GO
```

Step 5: Create Base Tables

```sql
-- Create Customers table (not partitioned)
CREATE TABLE Customers
(
    CustomerID INT IDENTITY(1,1) PRIMARY KEY,
    FirstName NVARCHAR(50) NOT NULL,
    LastName NVARCHAR(50) NOT NULL,
    Email NVARCHAR(100) NOT NULL UNIQUE,
    Phone NVARCHAR(20),
    Address NVARCHAR(200),
    City NVARCHAR(50),
    State NVARCHAR(50),
    ZipCode NVARCHAR(10),
    Country NVARCHAR(50),
    CreatedDate DATETIME2 DEFAULT GETDATE()
);
GO

-- Create Products table (not partitioned)
CREATE TABLE Products
(
    ProductID INT IDENTITY(1,1) PRIMARY KEY,
    ProductName NVARCHAR(100) NOT NULL,
    CategoryID INT,
    Price DECIMAL(10,2) NOT NULL,
    StockQuantity INT NOT NULL,
    Description NVARCHAR(500),
    CreatedDate DATETIME2 DEFAULT GETDATE()
);
GO
```

Step 6: Create Partitioned Orders Table

sql

```
-- Create partitioned Orders table
CREATE TABLE Orders
(
    OrderID INT IDENTITY(1,1),
    CustomerID INT NOT NULL,
    OrderDate DATETIME2 NOT NULL,
    TotalAmount DECIMAL(10,2) NOT NULL,
    OrderStatus NVARCHAR(20) NOT NULL,
    ShippingAddress NVARCHAR(200),
    PaymentMethod NVARCHAR(50),
    CreatedDate DATETIME2 DEFAULT GETDATE(),

    CONSTRAINT PK_Orders PRIMARY KEY (OrderID, OrderDate),
    CONSTRAINT FK_Orders_Customers FOREIGN KEY (CustomerID)
        REFERENCES Customers(CustomerID)
) ON PS_OrdersByYear(OrderDate);
GO
```

Step 7: Create Partitioned OrderItems Table

sql

```
-- Create partitioned OrderItems table
CREATE TABLE OrderItems
(
    OrderItemID INT IDENTITY(1,1),
    OrderID INT NOT NULL,
    OrderDate DATETIME2 NOT NULL,
    ProductID INT NOT NULL,
    Quantity INT NOT NULL,
    UnitPrice DECIMAL(10,2) NOT NULL,
    Subtotal AS (Quantity * UnitPrice) PERSISTED,

    CONSTRAINT PK_OrderItems PRIMARY KEY (OrderItemID, OrderDate),
    CONSTRAINT FK_OrderItems_Orders FOREIGN KEY (OrderID, OrderDate)
        REFERENCES Orders(OrderID, OrderDate),
    CONSTRAINT FK_OrderItems_Products FOREIGN KEY (ProductID)
        REFERENCES Products(ProductID)
) ON PS_OrdersByYear(OrderDate);
GO
```

Step 8: Insert Sample Data

sql

```
-- Insert sample customers
INSERT INTO Customers (FirstName, LastName, Email, Phone, Address, City, State, ZipCode, Country)
VALUES
('John', 'Doe', 'john.doe@email.com', '555-0001', '123 Main St', 'New York', 'NY', '10001', 'USA'),
('Jane', 'Smith', 'jane.smith@email.com', '555-0002', '456 Oak Ave', 'Los Angeles', 'CA', '90001', 'USA'),
('Robert', 'Johnson', 'robert.j@email.com', '555-0003', '789 Pine Rd', 'Chicago', 'IL', '60601', 'USA'),
('Emily', 'Brown', 'emily.brown@email.com', '555-0004', '321 Elm St', 'Houston', 'TX', '77001', 'USA'),
('Michael', 'Davis', 'michael.d@email.com', '555-0005', '654 Maple Dr', 'Phoenix', 'AZ', '85001', 'USA');
GO

-- Insert sample products
INSERT INTO Products (ProductName, CategoryID, Price, StockQuantity, Description)
VALUES
('Laptop Computer', 1, 999.99, 50, 'High-performance laptop'),
('Wireless Mouse', 1, 29.99, 200, 'Ergonomic wireless mouse'),
('USB-C Cable', 1, 12.99, 500, 'Fast charging USB-C cable'),
('Bluetooth Headphones', 2, 79.99, 150, 'Noise-canceling headphones'),
('Smartphone Stand', 1, 19.99, 300, 'Adjustable phone stand'),
('Mechanical Keyboard', 1, 149.99, 100, 'RGB mechanical keyboard'),
('External SSD 1TB', 1, 129.99, 75, 'Portable solid state drive'),
('Webcam HD', 2, 89.99, 120, '1080p webcam'),
('Monitor 27"', 1, 299.99, 60, '4K monitor'),
('Desk Lamp LED', 3, 39.99, 200, 'Adjustable LED desk lamp');
GO

-- Insert orders across different years
-- Orders for 2022
INSERT INTO Orders (CustomerID, OrderDate, TotalAmount, OrderStatus, ShippingAddress, PaymentMethod)
VALUES
(1, '2022-03-15', 1029.98, 'Completed', '123 Main St, New York, NY', 'Credit Card'),
(2, '2022-06-20', 109.98, 'Completed', '456 Oak Ave, Los Angeles, CA', 'PayPal'),
(3, '2022-09-10', 299.99, 'Completed', '789 Pine Rd, Chicago, IL', 'Credit Card');
GO

-- Orders for 2023
INSERT INTO Orders (CustomerID, OrderDate, TotalAmount, OrderStatus, ShippingAddress, PaymentMethod)
VALUES
(1, '2023-01-10', 159.98, 'Completed', '123 Main St, New York, NY', 'Credit Card'),
(4, '2023-04-15', 449.97, 'Completed', '321 Elm St, Houston, TX', 'Debit Card'),
(5, '2023-07-22', 219.98, 'Completed', '654 Maple Dr, Phoenix, AZ', 'Credit Card'),
(2, '2023-10-05', 89.99, 'Completed', '456 Oak Ave, Los Angeles, CA', 'PayPal'),
(3, '2023-12-20', 999.99, 'Completed', '789 Pine Rd, Chicago, IL', 'Credit Card');
GO

-- Orders for 2024
INSERT INTO Orders (CustomerID, OrderDate, TotalAmount, OrderStatus, ShippingAddress, PaymentMethod)
```

```
VALUES
(1, '2024-02-14', 329.98, 'Completed', '123 Main St, New York, NY', 'Credit Card'),
(2, '2024-03-25', 179.97, 'Shipped', '456 Oak Ave, Los Angeles, CA', 'PayPal'),
(3, '2024-05-18', 1299.97, 'Completed', '789 Pine Rd, Chicago, IL', 'Credit Card'),
(4, '2024-08-09', 259.98, 'Completed', '321 Elm St, Houston, TX', 'Debit Card'),
(5, '2024-11-30', 129.99, 'Processing', '654 Maple Dr, Phoenix, AZ', 'Credit Card');
GO

-- Orders for 2025
INSERT INTO Orders (CustomerID, OrderDate, TotalAmount, OrderStatus, ShippingAddress, PaymentMethod)
VALUES
(1, '2025-01-03', 999.99, 'Processing', '123 Main St, New York, NY', 'Credit Card'),
(3, '2025-01-04', 89.99, 'Processing', '789 Pine Rd, Chicago, IL', 'PayPal');
GO

-- Insert order items for the orders
-- 2022 Order Items
INSERT INTO OrderItems (OrderID, OrderDate, ProductID, Quantity, UnitPrice)
VALUES
(1, '2022-03-15', 1, 1, 999.99),
(1, '2022-03-15', 2, 1, 29.99),
(2, '2022-06-20', 4, 1, 79.99),
(2, '2022-06-20', 2, 1, 29.99),
(3, '2022-09-10', 9, 1, 299.99);
GO

-- 2023 Order Items
INSERT INTO OrderItems (OrderID, OrderDate, ProductID, Quantity, UnitPrice)
VALUES
(4, '2023-01-10', 6, 1, 149.99),
(4, '2023-01-10', 3, 1, 12.99),
(5, '2023-04-15', 9, 1, 299.99),
(5, '2023-04-15', 6, 1, 149.99),
(6, '2023-07-22', 7, 1, 129.99),
(6, '2023-07-22', 8, 1, 89.99),
(7, '2023-10-05', 8, 1, 89.99),
(8, '2023-12-20', 1, 1, 999.99);
GO

-- 2024 Order Items
INSERT INTO OrderItems (OrderID, OrderDate, ProductID, Quantity, UnitPrice)
VALUES
(9, '2024-02-14', 9, 1, 299.99),
(9, '2024-02-14', 2, 1, 29.99),
(10, '2024-03-25', 6, 1, 149.99),
(10, '2024-03-25', 2, 1, 29.99),
(11, '2024-05-18', 1, 1, 999.99),
```

```
(11, '2024-05-18', 9, 1, 299.99),
(12, '2024-08-09', 7, 2, 129.99),
(13, '2024-11-30', 7, 1, 129.99);
GO

-- 2025 Order Items
INSERT INTO OrderItems (OrderID, OrderDate, ProductID, Quantity, UnitPrice)
VALUES
(14, '2025-01-03', 1, 1, 999.99),
(15, '2025-01-04', 8, 1, 89.99);
GO
```

Step 9: Query Partitioned Data

sql

```sql
-- View partition information
SELECT
    OBJECT_NAME(p.object_id) AS TableName,
    p.partition_number AS PartitionNumber,
    fg.name AS FileGroupName,
    p.rows AS RowCount,
    au.total_pages * 8 / 1024 AS TotalSpaceMB
FROM sys.partitions p
INNER JOIN sys.allocation_units au ON p.partition_id = au.container_id
INNER JOIN sys.filegroups fg ON au.data_space_id = fg.data_space_id
WHERE OBJECT_NAME(p.object_id) = 'Orders'
ORDER BY p.partition_number;
GO

-- Query specific partition (2024 orders)
-- This query will only scan the 2024 partition
SELECT
    o.OrderID,
    o.OrderDate,
    c.FirstName + ' ' + c.LastName AS CustomerName,
    o.TotalAmount,
    o.OrderStatus
FROM Orders o
INNER JOIN Customers c ON o.CustomerID = c.CustomerID
WHERE o.OrderDate >= '2024-01-01' AND o.OrderDate < '2025-01-01'
ORDER BY o.OrderDate DESC;
GO

-- Verify partition elimination using execution plan
-- Right-click and select "Display Estimated Execution Plan"
-- You should see "Partition: 3" indicating only one partition is scanned
```

Step 10: Useful Partition Management Queries

sql

```sql
-- 1. View all partition functions
SELECT
    pf.name AS PartitionFunction,
    pf.type_desc,
    pf.fanout AS NumberOfPartitions,
    pf.boundary_value_on_right
FROM sys.partition_functions pf;
GO

-- 2. View partition schemes
SELECT
    ps.name AS PartitionScheme,
    pf.name AS PartitionFunction,
    ds.name AS FileGroupName
FROM sys.partition_schemes ps
INNER JOIN sys.partition_functions pf ON ps.function_id = pf.function_id
INNER JOIN sys.destination_data_spaces dds ON ps.data_space_id = dds.partition_scheme_id
INNER JOIN sys.data_spaces ds ON dds.data_space_id = ds.data_space_id
ORDER BY ps.name, dds.destination_id;
GO

-- 3. View partition boundary values
SELECT
    pf.name AS PartitionFunction,
    prv.boundary_id,
    prv.value AS BoundaryValue
FROM sys.partition_functions pf
INNER JOIN sys.partition_range_values prv ON pf.function_id = prv.function_id
ORDER BY pf.name, prv.boundary_id;
GO

-- 4. Check which partition a specific date belongs to
SELECT $PARTITION.PF_OrdersByYear('2024-06-15') AS PartitionNumber;
GO

-- 5. Summary report by partition
SELECT
    $PARTITION.PF_OrdersByYear(OrderDate) AS PartitionNumber,
    YEAR(OrderDate) AS OrderYear,
    COUNT(*) AS OrderCount,
    SUM(TotalAmount) AS TotalRevenue,
    AVG(TotalAmount) AS AverageOrderValue
FROM Orders
GROUP BY $PARTITION.PF_OrdersByYear(OrderDate), YEAR(OrderDate)
```

```
ORDER BY PartitionNumber;
GO
```

Step 11: Partition Maintenance Operations

sql

```sql
-- Add a new partition for 2026
-- First, add a new filegroup and file
ALTER DATABASE EcommerceDB
ADD FILEGROUP FG_Orders_2026;
GO

ALTER DATABASE EcommerceDB
ADD FILE
(
    NAME = Orders_2026_Data,
    FILENAME = 'C:\SQLData\Orders_2026.ndf',
    SIZE = 100MB,
    FILEGROWTH = 50MB
) TO FILEGROUP FG_Orders_2026;
GO

-- Modify the partition scheme to use the new filegroup
ALTER PARTITION SCHEME PS_OrdersByYear
NEXT USED FG_Orders_2026;
GO

-- Split the partition to add new boundary
ALTER PARTITION FUNCTION PF_OrdersByYear()
SPLIT RANGE ('2027-01-01');
GO

-- Archive old data (switch out 2022 partition)
-- First create an archive table with same structure
CREATE TABLE Orders_Archive_2022
(
    OrderID INT,
    CustomerID INT NOT NULL,
    OrderDate DATETIME2 NOT NULL,
    TotalAmount DECIMAL(10,2) NOT NULL,
    OrderStatus NVARCHAR(20) NOT NULL,
    ShippingAddress NVARCHAR(200),
    PaymentMethod NVARCHAR(50),
    CreatedDate DATETIME2,

    CONSTRAINT PK_Orders_Archive_2022 PRIMARY KEY (OrderID, OrderDate)
) ON FG_Orders_2022;
GO

-- Switch partition 1 (2022 data) to archive table
ALTER TABLE Orders
SWITCH PARTITION 1 TO Orders_Archive_2022;
```

```
GO

-- Verify the switch
SELECT COUNT(*) AS ArchivedOrders FROM Orders_Archive_2022;
GO
```

Step 12: Performance Comparison

sql

```
-- Enable statistics for comparison
SET STATISTICS IO ON;
SET STATISTICS TIME ON;
GO

-- Query without partition elimination (scanning all partitions)
SELECT
    COUNT(*) AS TotalOrders,
    SUM(TotalAmount) AS TotalRevenue
FROM Orders;
GO

-- Query with partition elimination (scanning only one partition)
SELECT
    COUNT(*) AS TotalOrders,
    SUM(TotalAmount) AS TotalRevenue
FROM Orders
WHERE OrderDate >= '2024-01-01' AND OrderDate < '2025-01-01';
GO

SET STATISTICS IO OFF;
SET STATISTICS TIME OFF;
GO
```

Step 13: Best Practices

1. **Choose the Right Partition Key**: Select a column frequently used in WHERE clauses
2. **Align Partitions**: Keep related tables partitioned on the same key
3. **Monitor Partition Sizes**: Keep partitions relatively balanced
4. **Use Partition Elimination**: Write queries that filter on partition key
5. **Regular Maintenance**: Archive old partitions, add new ones as needed
6. **Index Strategy**: Create aligned indexes on partitioned tables

Step 14: Creating Indexes on Partitioned Tables

```sql
-- Create aligned index on Orders
CREATE NONCLUSTERED INDEX IX_Orders_CustomerID
ON Orders(CustomerID, OrderDate)
ON PS_OrdersByYear(OrderDate);
GO

-- Create aligned index on OrderItems
CREATE NONCLUSTERED INDEX IX_OrderItems_ProductID
ON OrderItems(ProductID, OrderDate)
ON PS_OrdersByYear(OrderDate);
GO

-- Create non-aligned index (for demonstration - usually avoided)
CREATE NONCLUSTERED INDEX IX_Orders_Status
ON Orders(OrderStatus)
ON [PRIMARY];  -- Not partitioned
GO
```

Troubleshooting Common Issues

Issue 1: File Path Errors

Problem: Error creating files in C:\SQLData
Solution: Create the directory first:

```sql
EXEC xp_cmdshell 'mkdir C:\SQLData';
GO
```

Or use a different path like your SQL Server data directory.

Issue 2: Partition Function Errors

Problem: "Partition function already exists" **Solution**: Drop and recreate:

```sql
```

```
-- First drop dependent objects
DROP TABLE IF EXISTS Orders;
DROP TABLE IF EXISTS OrderItems;
DROP PARTITION SCHEME PS_OrdersByYear;
DROP PARTITION FUNCTION PF_OrdersByYear;
GO
```

Issue 3: Cannot Switch Partition

Problem: Constraints prevent partition switch **Solution**: Ensure archive table has identical schema and constraints

---

Additional Resources

- **Azure Documentation**: https://docs.microsoft.com/azure
- **SQL Server Partitioning**: https://docs.microsoft.com/sql/relational-databases/partitions
- **Azure Data Studio**: https://docs.microsoft.com/sql/azure-data-studio
- **Performance Tuning**: https://docs.microsoft.com/sql/relational-databases/performance

---

Summary

You've now completed:
✅ Created a Windows VM in Azure
✅ Installed Azure Data Studio
✅ Understood database partitioning concepts
✅ Implemented a real-world e-commerce partitioning solution
✅ Learned partition maintenance operations
✅ Explored performance benefits

This setup provides a solid foundation for managing large-scale e-commerce databases with efficient data partitioning strategies.