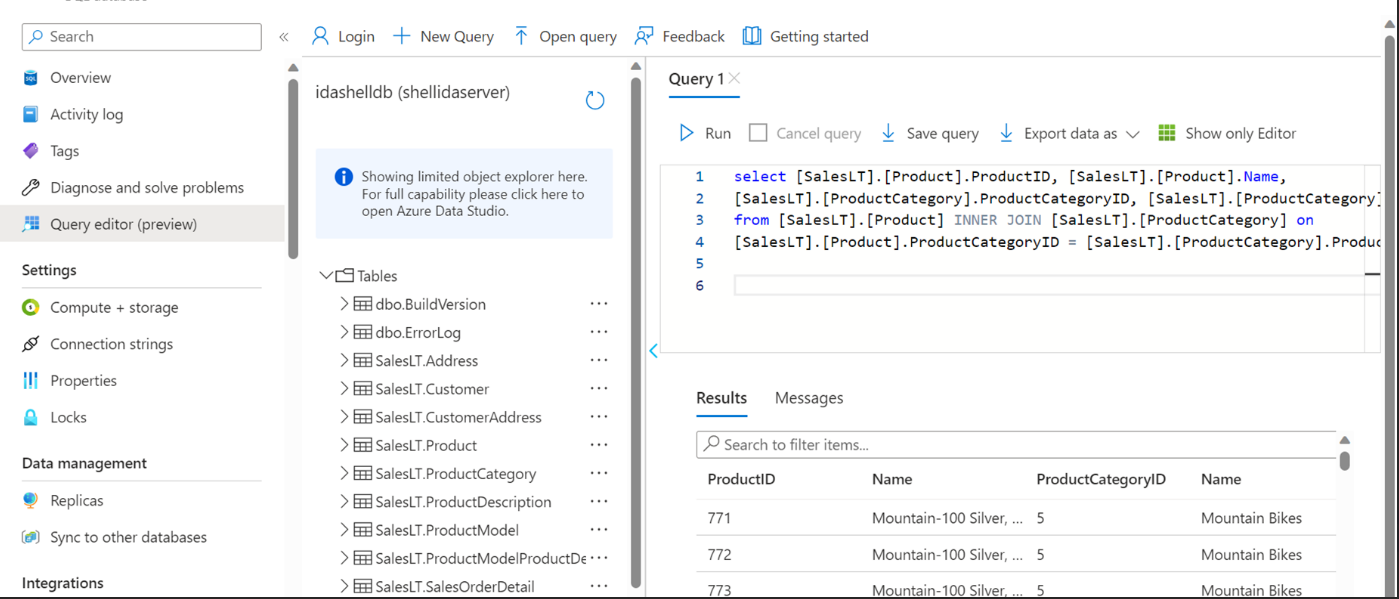
Custom Training

Day 4

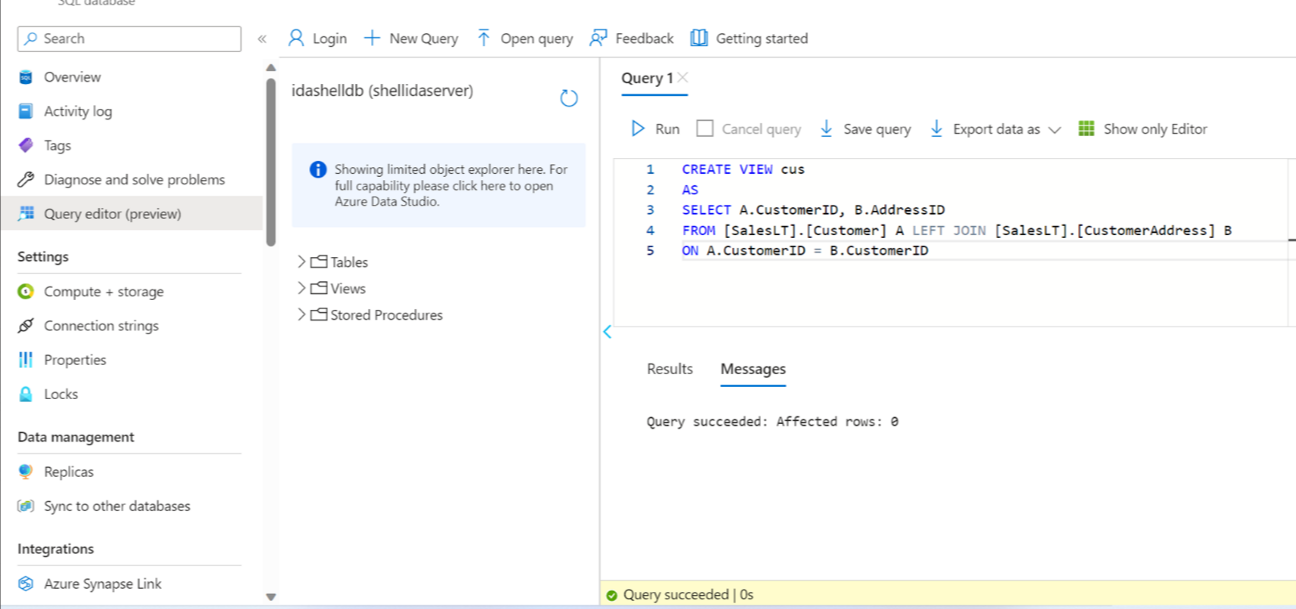
Creation of Elastic pool: It is used when we know one of the databases will occupy less space.

Joins:

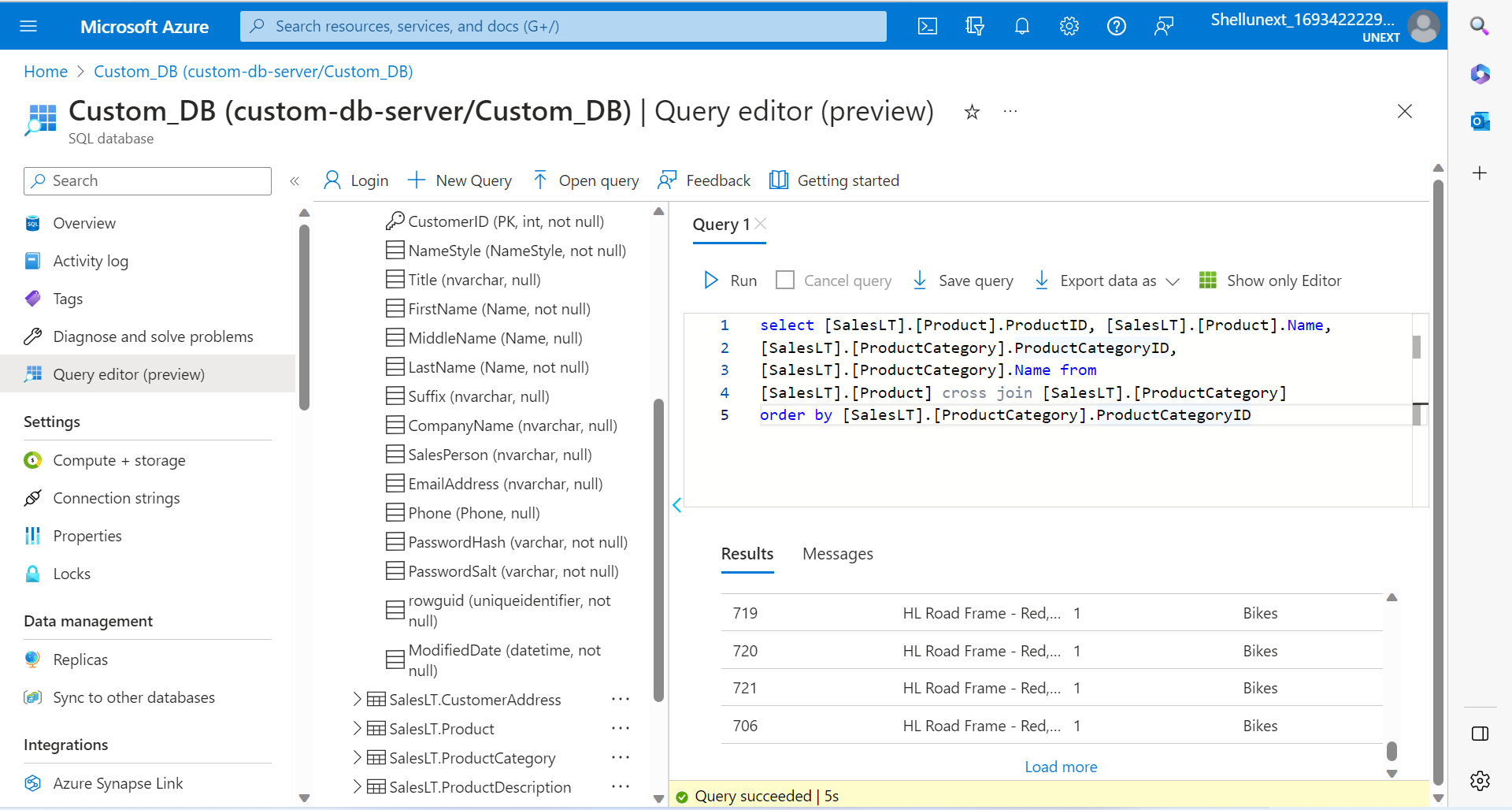
1. Inner Join: Returns only matched records from the joining tables.

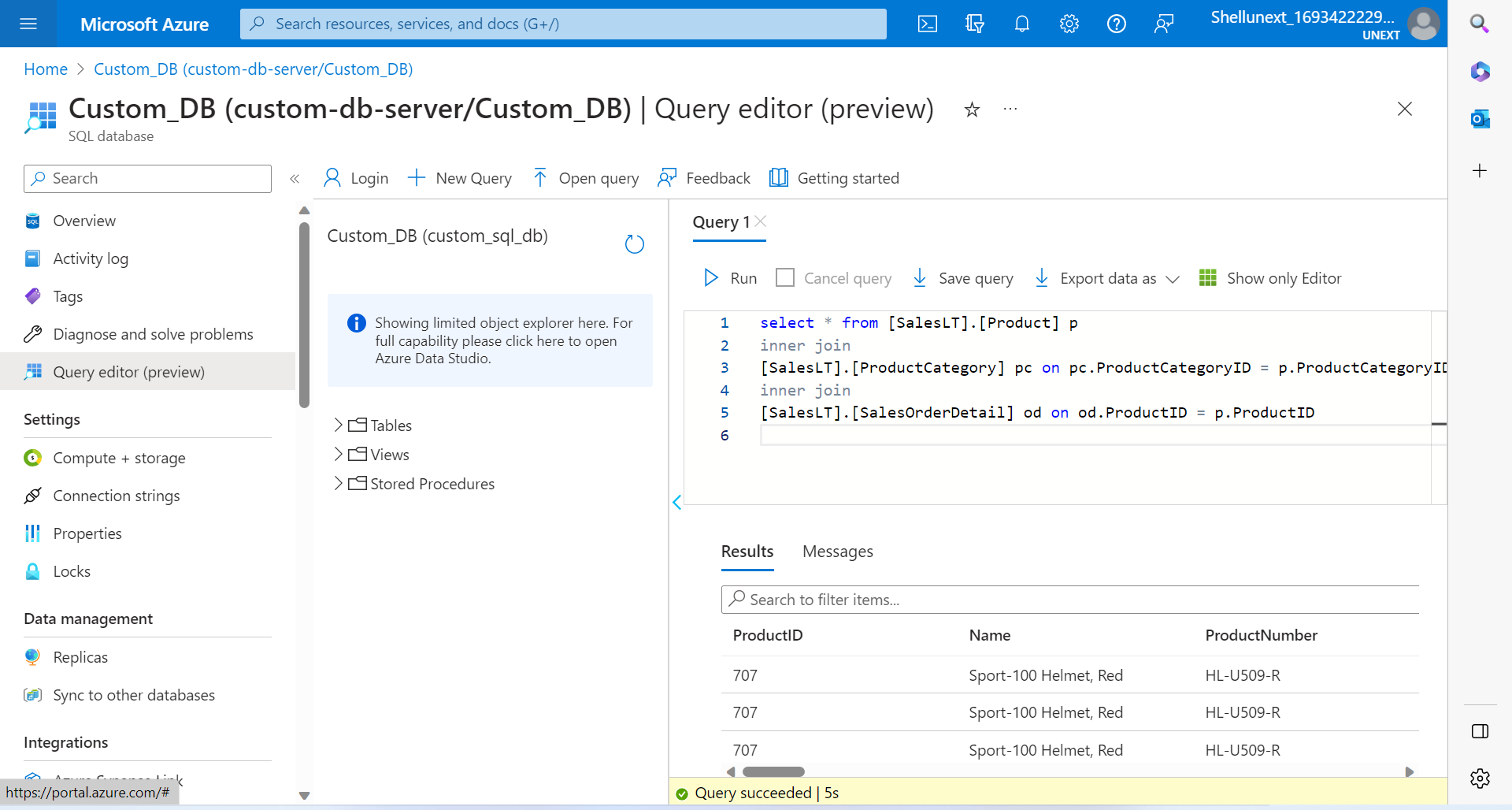


1. Left Join: Returns all the records from the left table.

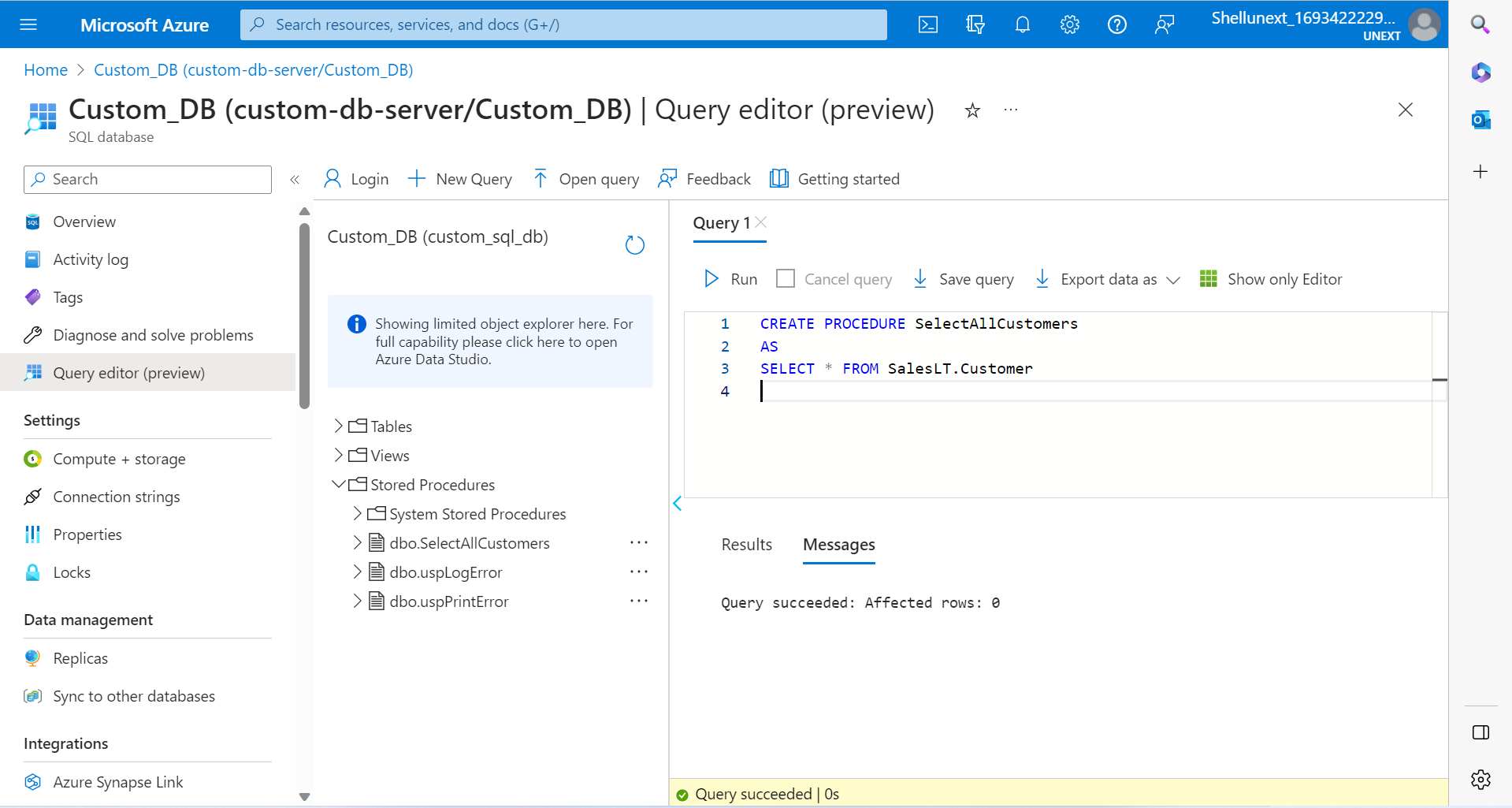


1. Right Join: Returns all the records from the right table.
2. Self Join: Joins with the same table only. Ex: Employee and Manager
3. Full Outer Join: Returns all the records from both the tables.
4. Cross Join: Returns the cartesian product of the records from both the tables.

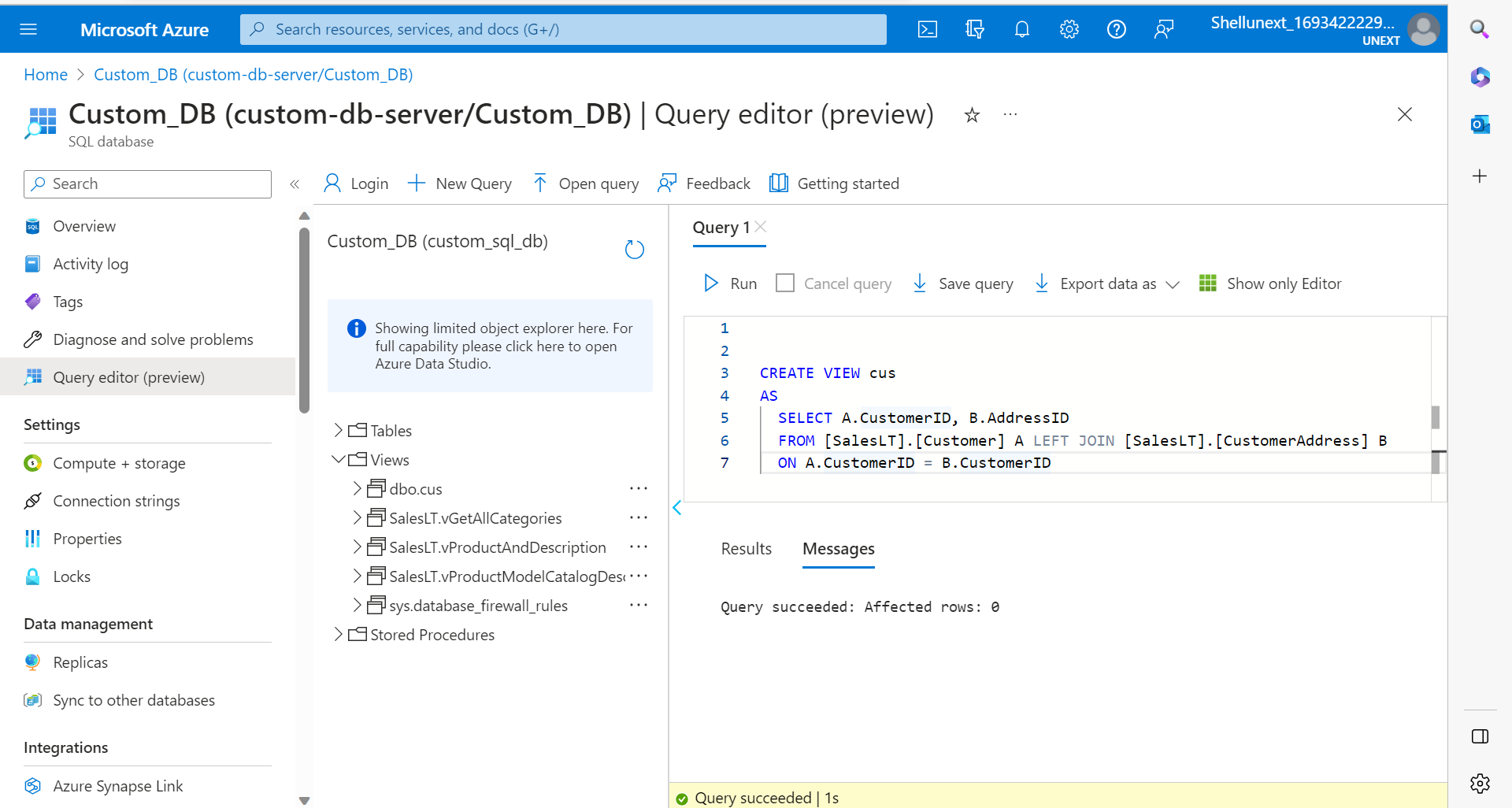




Stored Procedure: To store a lengthy and complex query we can use stored procedure. We can write parameterized stored procedure. We can put transaction to maintain the atomicity of the queries performed. We also put temporary table in the stored procedure.



Views: View of the table used in cases when limited columns should be available to an user.



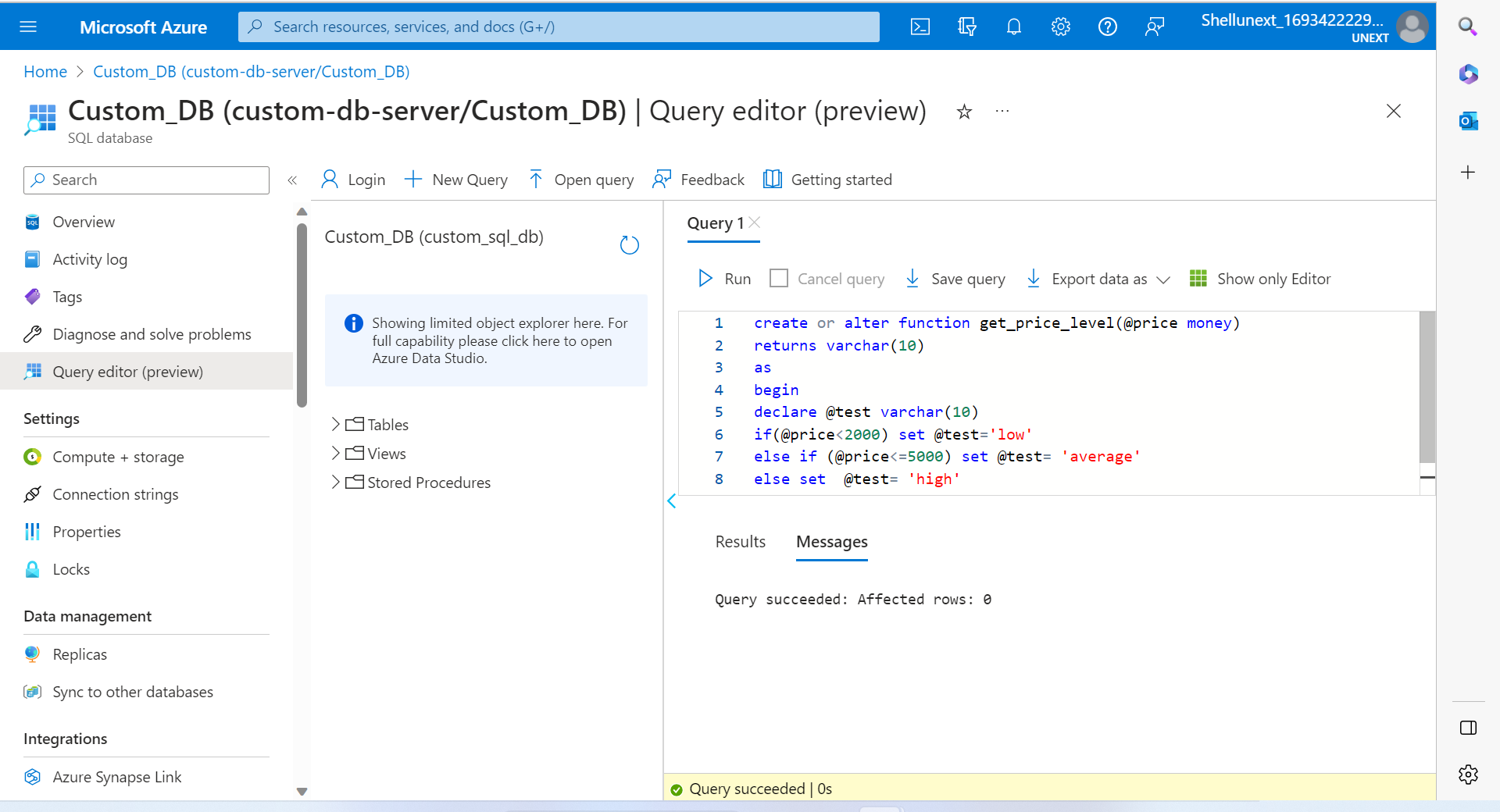
Temporary Table: Table which is created during a session and gets automatically created at the end of the session.

Select \* into #testtmp from tablename;

Select \* from #testtmp;

We run both the statements at once because it is considered as one session.

Functions: Piece of code which is going to decrease the code overhead and increase code reusability.

Sum, Avg etc. are system functions. We can create our own user-defined functions. 

2 types:

Scalar : Sum, Avg which gives one single value.

Table Values: Gives output in the form of rows and columns.

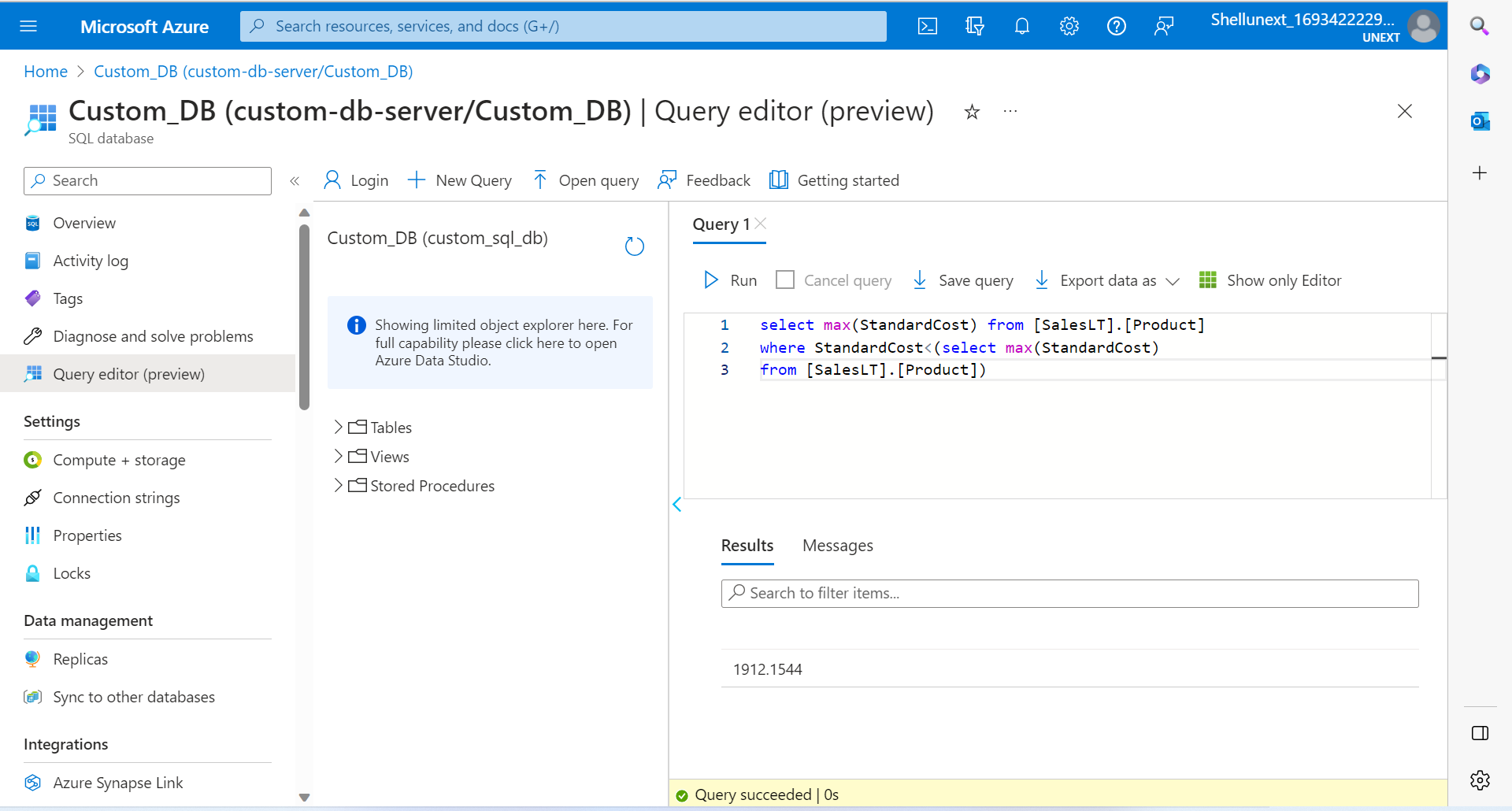
#scalar function  
CREATE FUNCTION dbo.GetSalaryCategory(@salary DECIMAL(10, 2)) RETURNS NVARCHAR(10)  
AS  
BEGIN  
    DECLARE @category NVARCHAR(10)  
  
    IF @salary > 2000  
        SET @category = 'High'  
    ELSE IF @salary < 1000  
        SET @category = 'Low'  
    ELSE  
        SET @category = 'NA'  
  
    RETURN @category  
END;  
GO

CREATE TABLE Employee (  
    EmployeeID INT PRIMARY KEY,  
    FirstName NVARCHAR(50),  
    LastName NVARCHAR(50),  
    Salary DECIMAL(10, 2),  
    SalaryCategory NVARCHAR(10)  
);

INSERT INTO Employee (EmployeeID, FirstName, LastName, Salary, SalaryCategory)  
VALUES  
    (1, 'John', 'Doe', 2500.00, dbo.GetSalaryCategory(2500.00)),  
    (2, 'Jane', 'Smith', 800.00, dbo.GetSalaryCategory(800.00)),  
    (3, 'Bob', 'Johnson', 1500.00, dbo.GetSalaryCategory(1500.00)),  
    (4, 'Alice', 'Williams', 3000.00, dbo.GetSalaryCategory(3000.00));

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Tabled Valued FUcntion  
CREATE FUNCTION dbo.GetEmployeesBySalaryRange (@minSalary DECIMAL(10, 2), @maxSalary DECIMAL(10, 2))  
RETURNS TABLE  
AS  
RETURN (  
    SELECT EmployeeID, FirstName, LastName, Salary  
    FROM Employee  
    WHERE Salary BETWEEN @minSalary AND @maxSalary  
);

Sub-Query: Query inside a query. Sub-Query in executed first, then the main query.



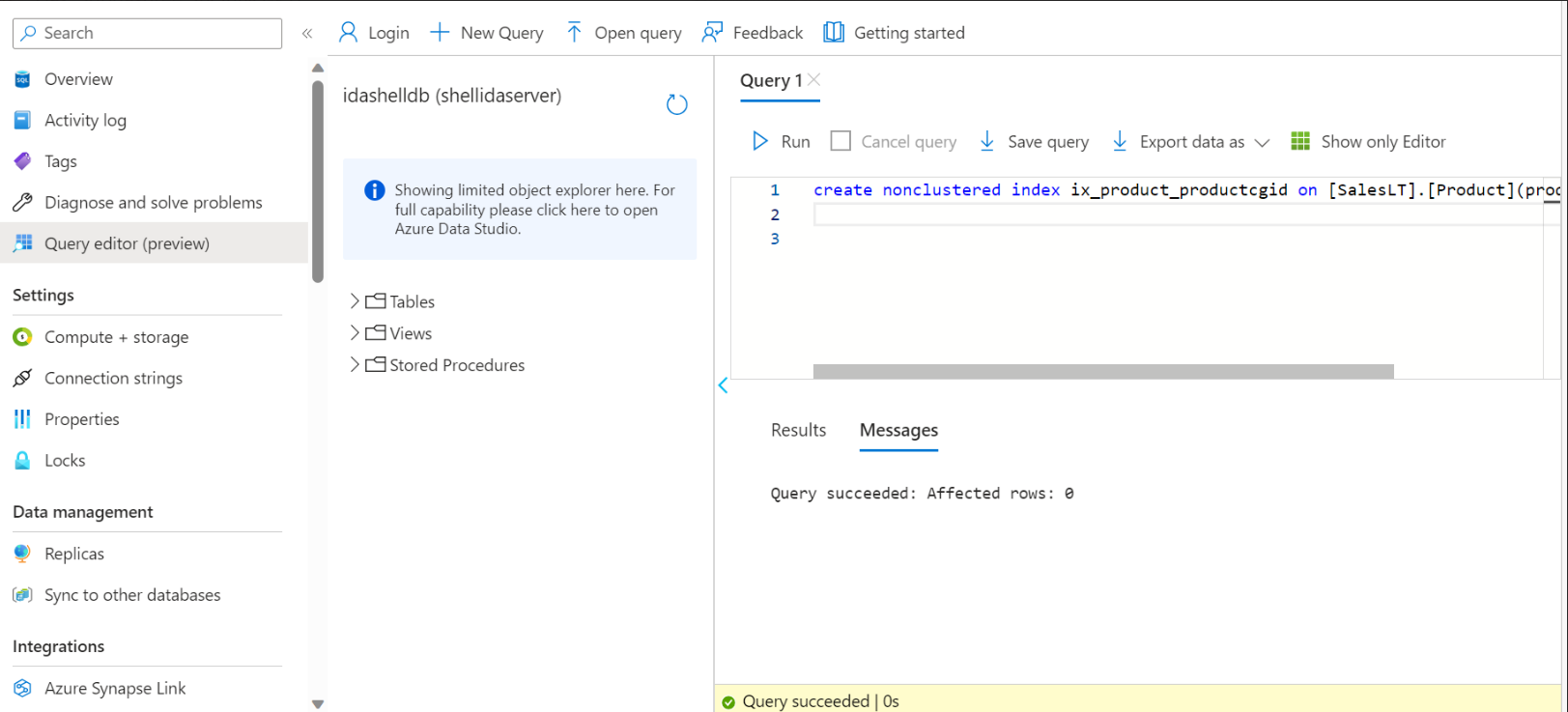
Union and Union All: Joins two table vertically. Should have same number of columns, same column names and same data type. Union all will accept duplicates and Union will remove the duplicates.

Intersect: Only common records are returned.

Index: 2 fetch data faster.

3 types:

1. Clustered index: On primary key only, it orders the rows and creates clusters with pointers
2. Non-Clustered Index: It will not order rows but creates groups with pointers.



1. Column Stored Index: We apply on a column to fetch data faster from a column.

Heap is default that means no index.