**5. Implementing Access Control in Relational Database**

**AIM:**

To Implement access control in relational databases is crucial for ensuring the security and integrity of data.

**ALGORITHM:**

1. **Download** the latest MS SQL Server version from the official Microsoft website.
2. **Table Creation**: Define a table for storing customer data.
3. **Security Predicate Function**: Implement a function that filters rows based on the current user.
4. **Security Policy**: Apply the predicate function to enforce row-level security.
5. **Create Users**: Define users with login credentials.
6. **Grant Permissions**: Assign access permissions to the users for the table.
7. **Policy Enforcement**: The filter ensures that each user can only see or modify their own data based on their name.

**PROGRAM:**

**--Role-Based Access Control (RBAC)**

**--Row-Level Security**

**-- Create a new table with a security policy**

CREATE TABLE Customers1 (

CustomerID int,

Name varchar(50),

Email varchar(100)

);

**--Create a Security Predicate Function**

CREATE FUNCTION dbo.fn\_securitypredicate(@Name AS varchar(50))

RETURNS TABLE

WITH SCHEMABINDING

AS

RETURN SELECT 1 AS result

WHERE @Name = USER\_NAME();

**--Create a Security Policy**

CREATE SECURITY POLICY CustomerSecurityPolicy

ADD FILTER PREDICATE dbo.fn\_securitypredicate(Name) ON dbo.Customers

WITH (STATE = ON);

**--Create the User1**

CREATE LOGIN user1 WITH PASSWORD = 'root@123';

CREATE USER user1 FOR LOGIN user1;

**--Create the User2**

CREATE LOGIN user2 WITH PASSWORD = 'root@456';

CREATE USER user2 FOR LOGIN user2;

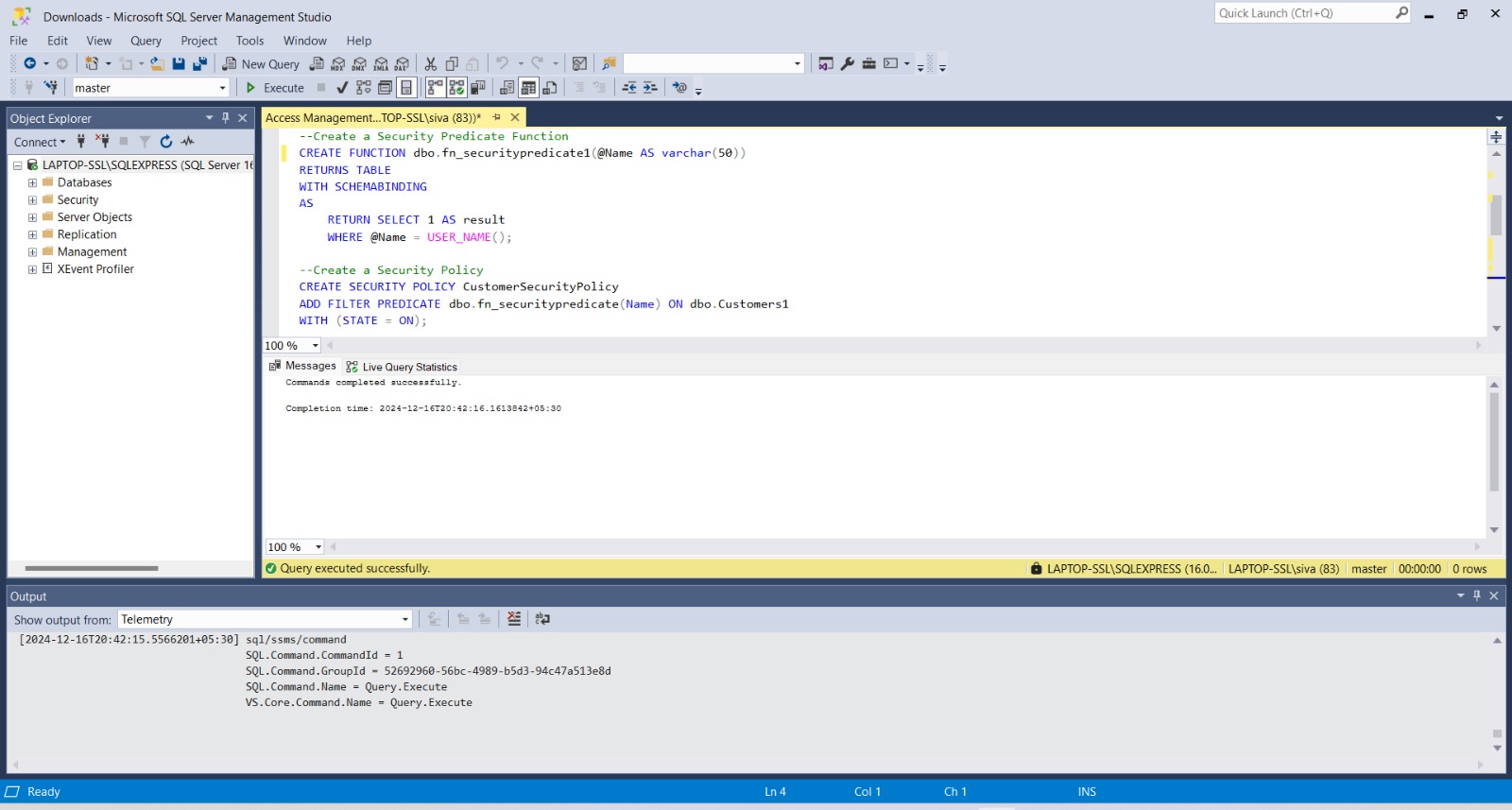
**--Grant Permissions**

GRANT SELECT, INSERT, UPDATE ON dbo.Customers TO user1;

**--Grant Permissions2**

GRANT SELECT, INSERT, UPDATE ON dbo.Customers TO user2;

**OUTPUT:**



**RESULT:**

Hence the implementation of access control in relational databases is crucial for ensuring the security and integrity of data is executed successfully.