/\* ==========================================================

EDIT HERE if needed (only if your names differ)

----------------------------------------------------------

Protected table : dbo.pcode

Filter column : State (NVARCHAR/CHAR recommended)

========================================================== \*/

/\* 0) Peek at data (optional) \*/

SELECT TOP (5) State, \*

FROM dbo.pcode;

/\* 1) Mapping table: who can see what state(s) \*/

IF OBJECT\_ID(N'dbo.UserStateAccess', N'U') IS NULL

BEGIN

CREATE TABLE dbo.UserStateAccess

(

UserName NVARCHAR(256) NOT NULL, -- store lower-case upn/login

State NVARCHAR(50) NOT NULL, -- store UPPER state codes/names

AccessLevel NVARCHAR(50) NULL, -- 'Reader','Manager','Admin'

CONSTRAINT PK\_UserStateAccess PRIMARY KEY (UserName, State)

);

CREATE INDEX IX\_UserStateAccess\_User ON dbo.UserStateAccess(UserName, State);

END;

/\* Seed test users

- Use UPPER for State values to keep comparisons consistent.

- Replace 'CA','NM' with real states present in dbo.pcode. \*/

TRUNCATE TABLE dbo.UserStateAccess;

INSERT dbo.UserStateAccess(UserName, State, AccessLevel)

VALUES

(N'alice@company.com', N'CA', N'Reader'), -- sees California only

(N'bob@company.com', N'NM', N'Reader'), -- sees New Mexico only

(N'manager@company.com', N'CA', N'Manager'), -- example: multiple states

(N'manager@company.com', N'NM', N'Manager'),

(N'superuser@company.com', N'\*', N'Admin'); -- Admin row (see-all)

/\* 2) Predicate function: allow if user has matching State OR is Admin \*/

IF OBJECT\_ID(N'dbo.fn\_rls\_state', N'IF') IS NOT NULL

DROP FUNCTION dbo.fn\_rls\_state;

GO

CREATE FUNCTION dbo.fn\_rls\_state (@State NVARCHAR(50))

RETURNS TABLE

WITH SCHEMABINDING

AS

RETURN

SELECT 1 AS AllowRow

WHERE

/\* Normalize both sides:

- Map session identity to lower-case

- Compare State in UPPER \*/

EXISTS (

SELECT 1

FROM dbo.UserStateAccess a

WHERE UPPER(a.State) = UPPER(@State)

AND a.UserName = LOWER(

COALESCE(

CAST(SESSION\_CONTEXT(N'upn') AS NVARCHAR(256)),

SUSER\_SNAME()

)

)

)

/\* Admin can see everything (AccessLevel = 'Admin') \*/

OR EXISTS (

SELECT 1

FROM dbo.UserStateAccess a

WHERE a.AccessLevel = N'Admin'

AND a.UserName = LOWER(

COALESCE(

CAST(SESSION\_CONTEXT(N'upn') AS NVARCHAR(256)),

SUSER\_SNAME()

)

)

);

GO

/\* 3) Security policy: filter + block on dbo.pcode(State) \*/

IF OBJECT\_ID(N'dbo.rls\_state\_policy', N'SP') IS NOT NULL

DROP SECURITY POLICY dbo.rls\_state\_policy;

GO

CREATE SECURITY POLICY dbo.rls\_state\_policy

ADD FILTER PREDICATE dbo.fn\_rls\_state(State) ON dbo.pcode,

ADD BLOCK PREDICATE dbo.fn\_rls\_state(State) ON dbo.pcode

WITH (STATE = ON);

GO

/\* 4) Tests (no user creation needed; use SESSION\_CONTEXT) \*/

-- Alice -> should only see CA rows

EXEC sys.sp\_set\_session\_context @key=N'upn', @value=N'alice@company.com';

SELECT DISTINCT State FROM dbo.pcode ORDER BY State;

SELECT COUNT(\*) AS RowsAlice FROM dbo.pcode;

EXEC sys.sp\_set\_session\_context @key=N'upn', @value=NULL;

-- Bob -> should only see NM rows

EXEC sys.sp\_set\_session\_context @key=N'upn', @value=N'bob@company.com';

SELECT DISTINCT State FROM dbo.pcode ORDER BY State;

SELECT COUNT(\*) AS RowsBob FROM dbo.pcode;

EXEC sys.sp\_set\_session\_context @key=N'upn', @value=NULL;

-- Manager -> should see CA and NM (both rows)

EXEC sys.sp\_set\_session\_context @key=N'upn', @value=N'manager@company.com';

SELECT DISTINCT State FROM dbo.pcode ORDER BY State;

SELECT COUNT(\*) AS RowsManager FROM dbo.pcode;

EXEC sys.sp\_set\_session\_context @key=N'upn', @value=NULL;

-- Superuser (Admin) -> should see ALL states

EXEC sys.sp\_set\_session\_context @key=N'upn', @value=N'superuser@company.com';

SELECT DISTINCT State FROM dbo.pcode ORDER BY State;

SELECT COUNT(\*) AS RowsAdmin FROM dbo.pcode;

EXEC sys.sp\_set\_session\_context @key=N'upn', @value=NULL;

ALTER SECURITY POLICY dbo.rls\_state\_policy WITH (STATE = OFF);

DROP SECURITY POLICY dbo.rls\_state\_policy;

DROP FUNCTION dbo.fn\_rls\_state;

DROP TABLE dbo.UserStateAccess;