//Row Access Policy v Dynamic Masking v Conditional Masking Sample Script

--initial setup of environment

use role sysadmin;

/\*create or replace warehouse wh\_access\_demo

with

warehouse\_size='XSMALL'

auto\_suspend = 600

auto\_resume = true

initially\_suspended=true;

use warehouse wh\_access\_demo;\*/

use warehouse DKWHXS;

/\*create or replace database db\_access\_demo;

use database db\_access\_demo;\*/

USE DATABASE USECASE\_DB;

--Create working area

--Create schema if not exists DATA\_GOVERNANCE\_DEMO;

USE SCHEMA USECASE\_DB.DATA\_GOVERNANCE\_DEMO;

--Now we're going to quickly demo the 3 different types of policy we can use to alter data access on tables

--First create a mock customer table

Create or replace Table customers

(

ID integer autoincrement (1,1),

CustNumber varchar(10),

FirstName varchar(100),

LastName varchar(100)

);

--Populate with some example data

Insert Into customers(CustNumber, FirstName, LastName)

Values

('A123456', 'John','Sullivan'),

('B123456', 'Jim','Corbett'),

('C123456', 'Bob','Fitzsimmons'),

('D123456', 'Jim','Jeffries'),

('E123456', 'Marvin','Hart'),

('F123456', 'Tommy','Burns'),

('G123456', 'Jack','Johnson'),

('H123456', 'Jess','Willard'),

('I123456', 'Jack','Dempsey'),

('J123456', 'Gene','Tunney');

select \* from customers;

--Now if we want to restrict visibility of the data to certain roles we can create a masking policy

--This policy will blank out a value when the role is not sysadmin

create or replace masking policy admin\_only as

(val string) returns string ->

CASE

WHEN current\_role() IN ('SYSADMIN') THEN val

ELSE '\*\*\*\*\*\*\*\*'

END;

--now we apply it to the columns in the table we want to blank out:

Alter Table customers

Modify Column CustNumber

Set masking policy admin\_only;

Alter Table customers

Modify Column FirstName

Set masking policy admin\_only;

Alter Table customers

Modify Column LastName

Set masking policy admin\_only;

DESC TABLE customers;

--We can see all customer details as we're sysadmin

Select \*

From customers;

--but if we switch to a different role:

use role accountadmin;

--they're now blanked out

USE SCHEMA USECASE\_DB.DATA\_GOVERNANCE\_DEMO;

Select \*

From customers;

use role sysadmin;

USE SCHEMA USECASE\_DB.DATA\_GOVERNANCE\_DEMO;

--Next scenario, the GDPR right to be forgotten

--we'll rebuild our table so we have a clean start

Create or replace Table customers

(

ID integer autoincrement (1,1),

CustNumber varchar(10),

FirstName varchar(100),

LastName varchar(100)

);

--Populate

Insert Into customers(CustNumber, FirstName, LastName)

Values

('A123456', 'John','Sullivan'),

('B123456', 'Jim','Corbett'),

('C123456', 'Bob','Fitzsimmons'),

('D123456', 'Jim','Jeffries'),

('E123456', 'Marvin','Hart'),

('F123456', 'Tommy','Burns'),

('G123456', 'Jack','Johnson'),

('H123456', 'Jess','Willard'),

('I123456', 'Jack','Dempsey'),

('J123456', 'Gene','Tunney');

--Bob has given notice that he wishes to be forgotten, his data should no longer be viewable by anyone

--so we delete Bob from our customer table

Delete From customers

Where CustNumber = 'C123456';

--Bob is now not visible when we query the table

Select \*

From customers;

--But he is still in time travel:

Select \*

From customers before(statement => last\_query\_id(-2));

--So as a secondary layer we will create a forgotten customers table using the anonymous internal ID

Create OR REPLACE Table forgotten

(

ID int

);

--And add Bob's ID to the table

Insert into forgotten Values(3);

--We can now set up a row access policy on the table to block forgotten records from returning, even with time travel

create or replace row access policy ForgottenCustomers as (ID int) returns boolean ->

not exists (

select 1

from forgotten f

where f.ID = ID

);

--And add it to our table

alter table customers add row access policy ForgottenCustomers on (ID);

--Now when we rerun the statement using time travel we do not get Bob returned:

Select \*

From customers before(statement => last\_query\_id(-7));

--For a production implementation you could hardcode the database and schema name in the policy

--This would ensure that even old cloned copies would continue to reference the up to date

--prod copy of the forgotten table ensuring that the record would be inaccessible across all clones

--Next scenario, Bob again wishes to be forgotten but this time we want to mask his details from everyone

--we can try and achieve this with conditional masking (private preview)

--again, let's reset our table

Create or replace Table customers

(

ID integer autoincrement (1,1),

CustNumber varchar(10),

FirstName varchar(100),

LastName varchar(100)

);

--Populate

Insert Into customers(CustNumber, FirstName, LastName)

Values

('A123456', 'John','Sullivan'),

('B123456', 'Jim','Corbett'),

('C123456', 'Bob','Fitzsimmons'),

('D123456', 'Jim','Jeffries'),

('E123456', 'Marvin','Hart'),

('F123456', 'Tommy','Burns'),

('G123456', 'Jack','Johnson'),

('H123456', 'Jess','Willard'),

('I123456', 'Jack','Dempsey'),

('J123456', 'Gene','Tunney');

--We can see all customers

Select \*

From customers;

--Bob still wishes to be forgotten so we will insert him into the forgotten table

Create or replace Table forgotten

(

ID int

);

Insert into forgotten Values(3);

--now we will set up conditional masking to hide all of Bob's details

--First we create the policy

create or replace masking policy masking\_forgotten\_customers as

(val string, ID int) returns varchar ->

case

when exists

(

select 1

from forgotten f

where f.ID = ID

)

then repeat('\*', 10) --Record found in forgotten table so mask

else val --otherwise return value

end;

--and add the policy to the relevant columns

Alter Table customers

Modify Column CustNumber

Set masking policy masking\_forgotten\_customers using (CustNumber, ID);

Alter Table customers

Modify Column FirstName

Set masking policy masking\_forgotten\_customers using (FirstName, ID);

Alter Table customers

Modify Column LastName

Set masking policy masking\_forgotten\_customers using (LastName, ID);

--Now when we try and select Bob will be masked even if we were to time travel back to before the record was added or the policy was created

USE ROLE SYSADMIN;

Select \*

From customers;

Select \*

From customers before(statement => last\_query\_id(-7));

--Tidy up

drop warehouse wh\_access\_demo;

drop database db\_access\_demo;