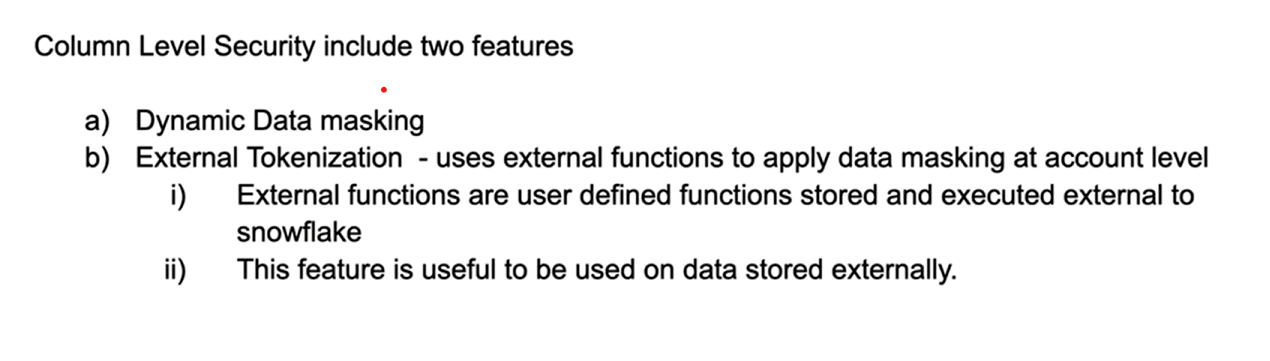
**DYNAMIC DATA MASKING**

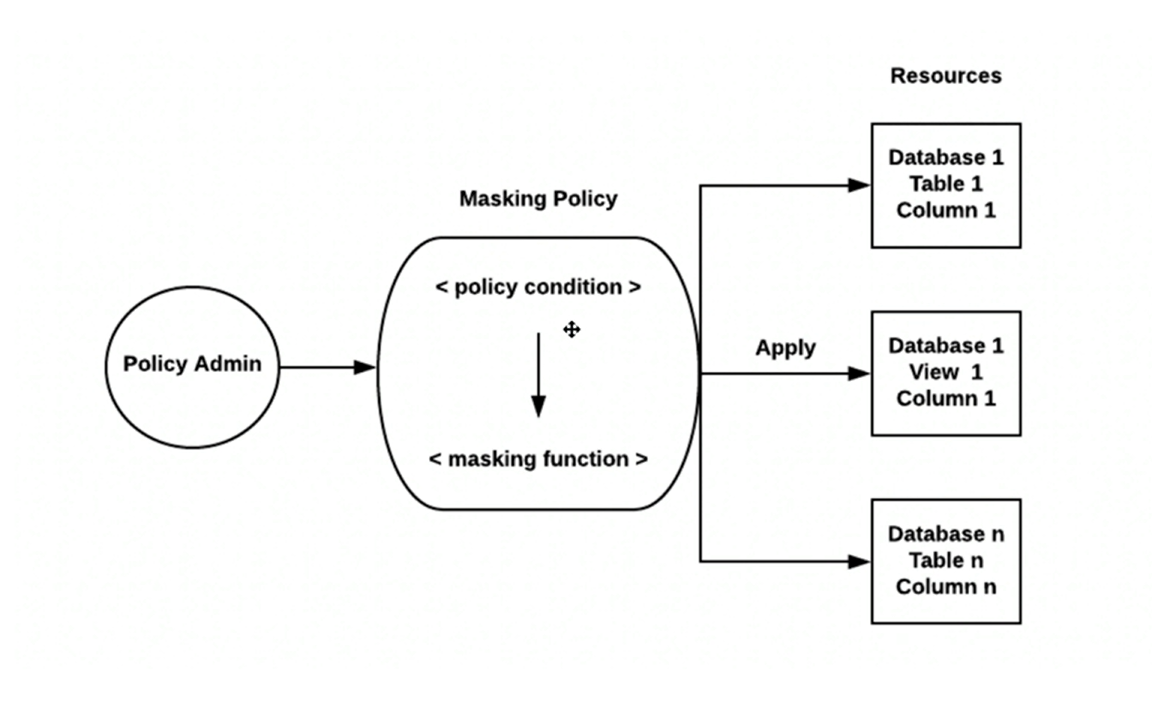
Dynamic data masking (comes under column level security) is masking the data and showing the right data to the right users with correct role.

So, here are the following:

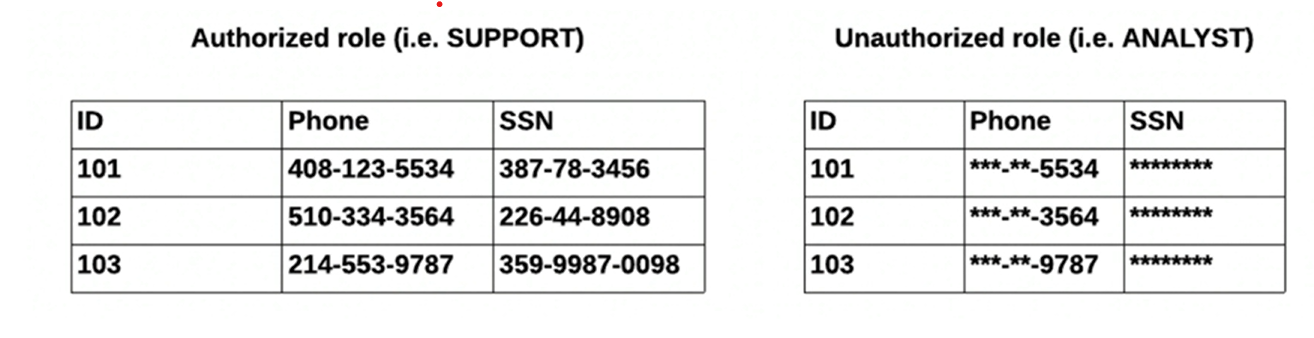
****

Dynamic data masking is a schema level object which we can create in snowflake and apply it in our tables, views and at column level

**General overview:**

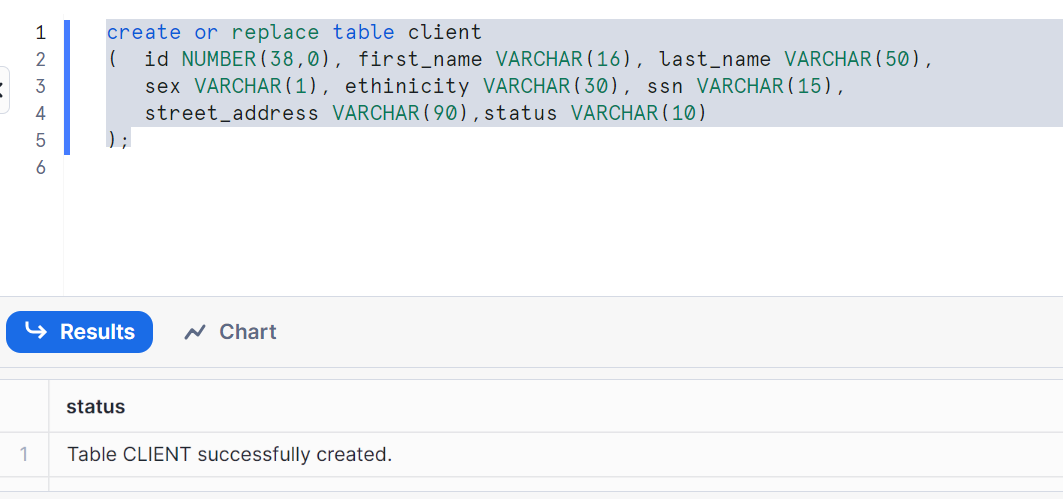


Here is the example output that we need to mask

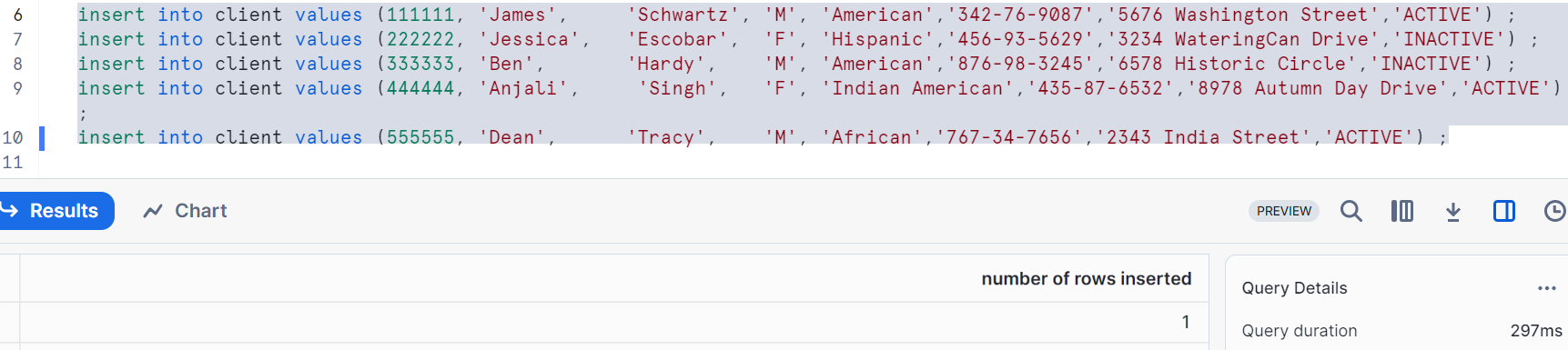


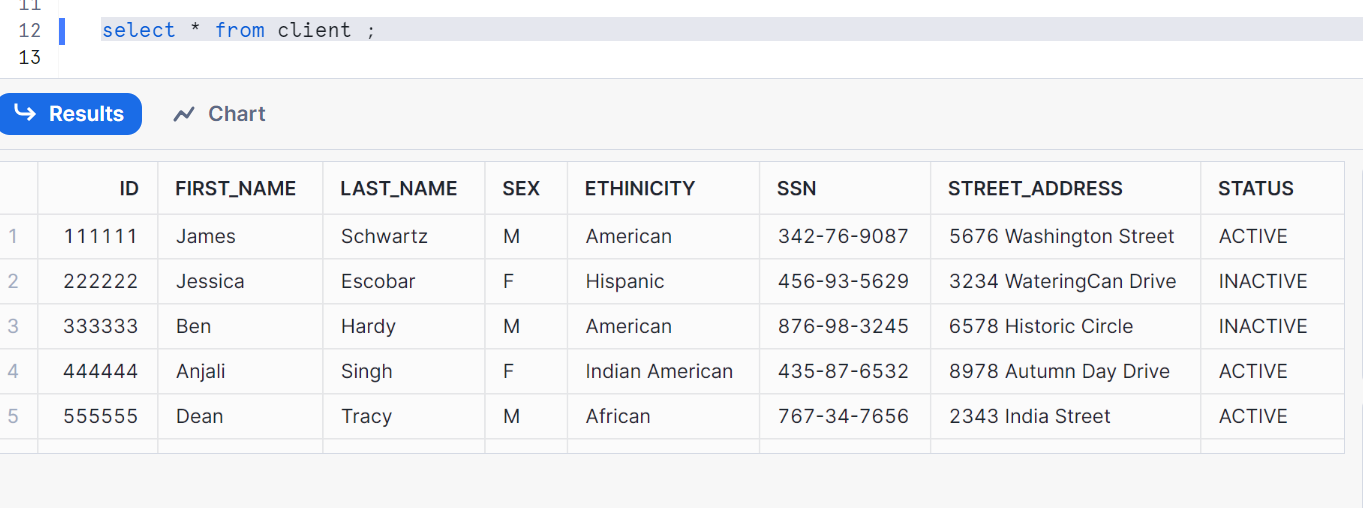
Let’s do some hand’s on

Create a table

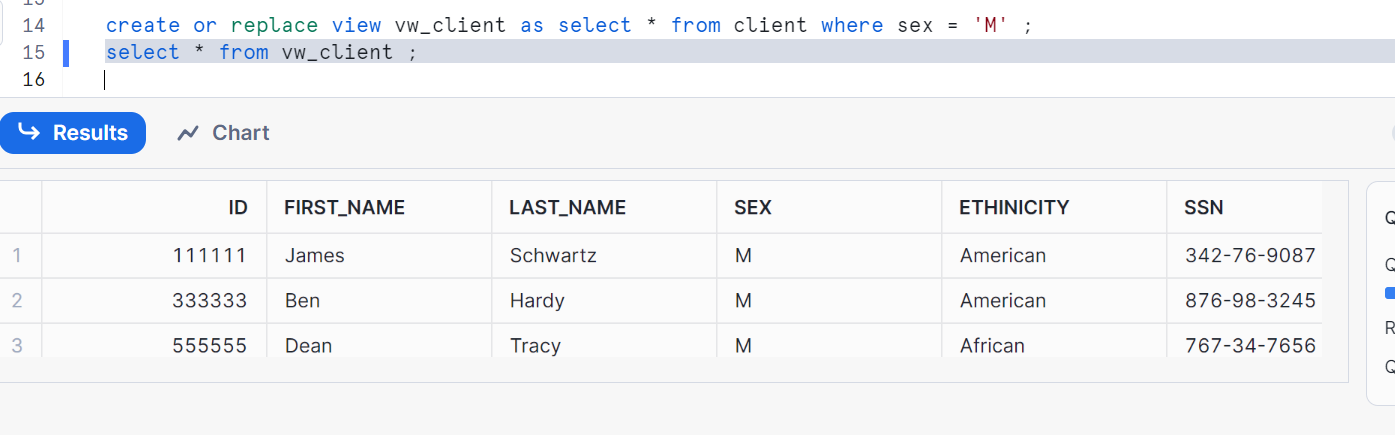


Insert some data

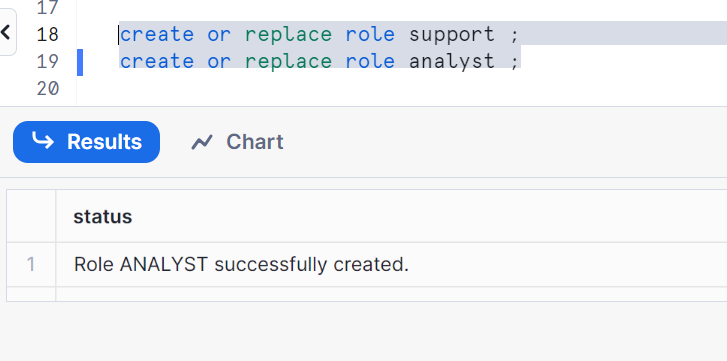




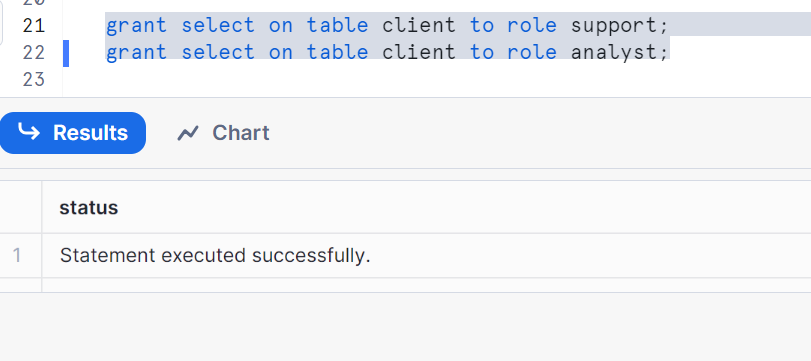
Create a View also to demonstrate the data masking



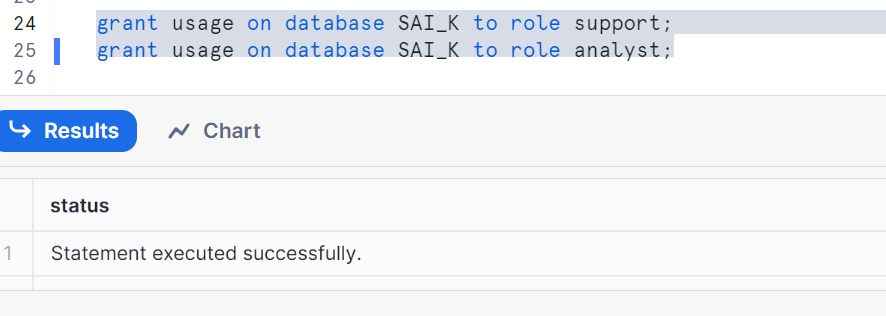
Create roles: support and analyst

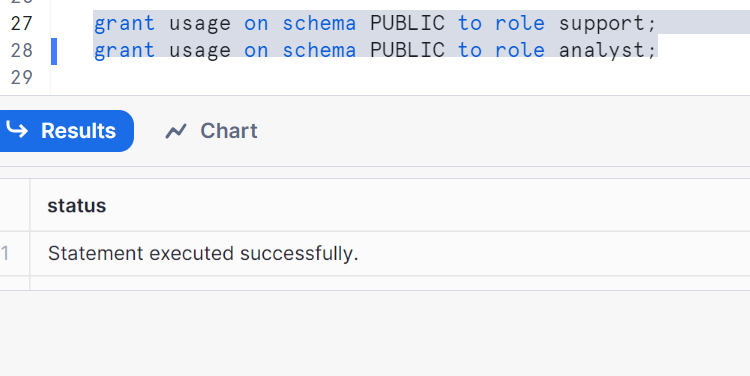


Give grants to the roles (note: make sure you are in account admin)

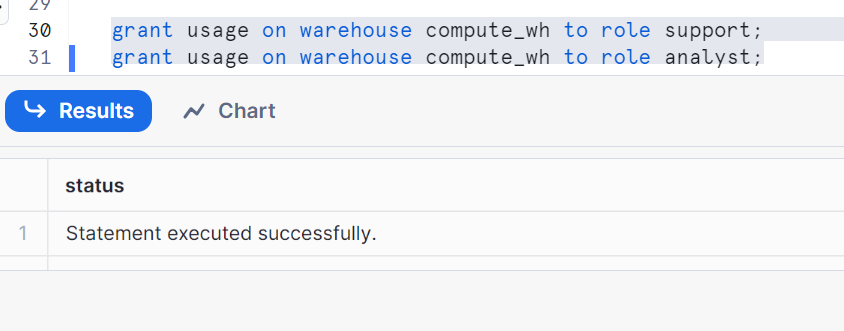


Grant schema usage to roles

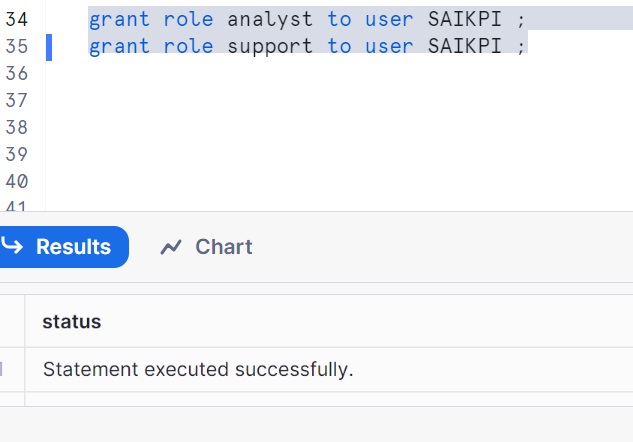




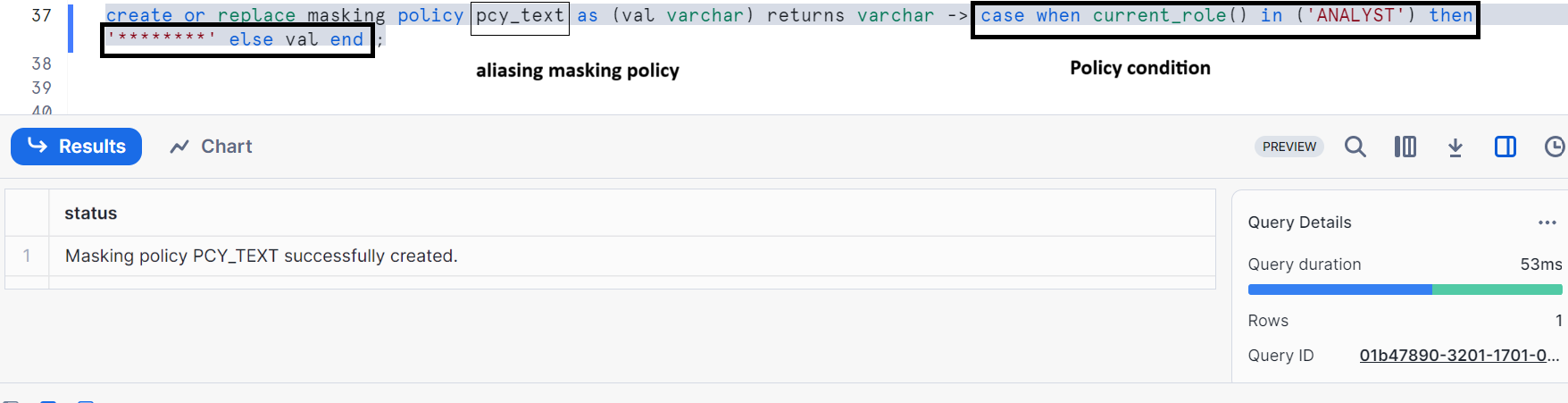
Grant usage on warehouse to roles

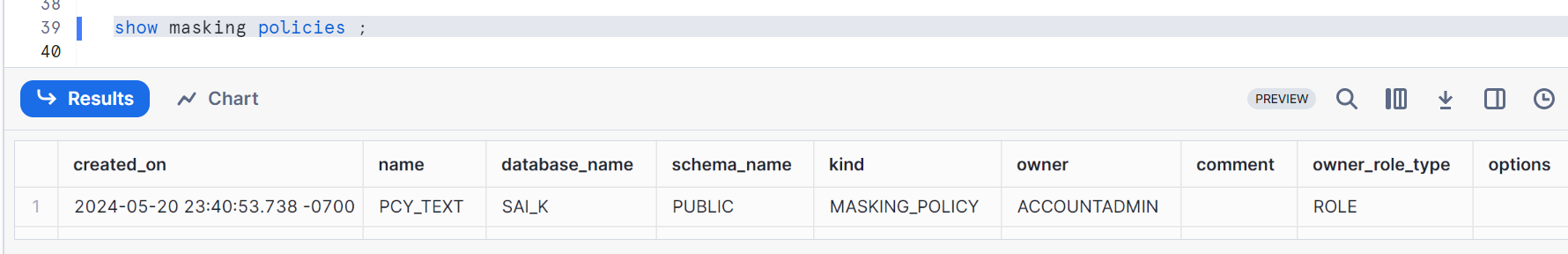


Now user should have these roles

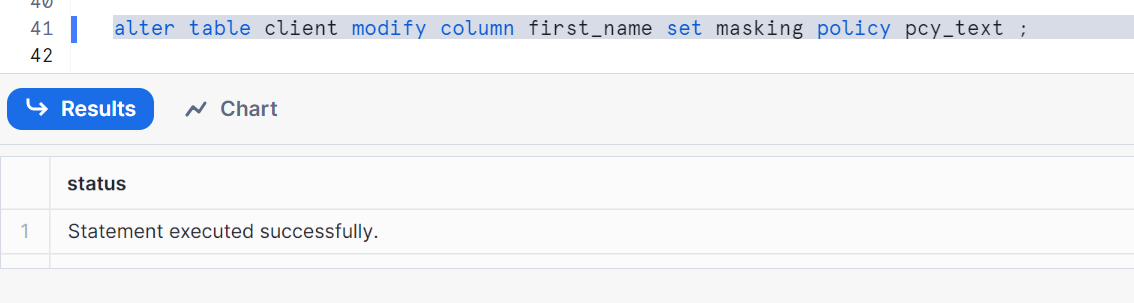


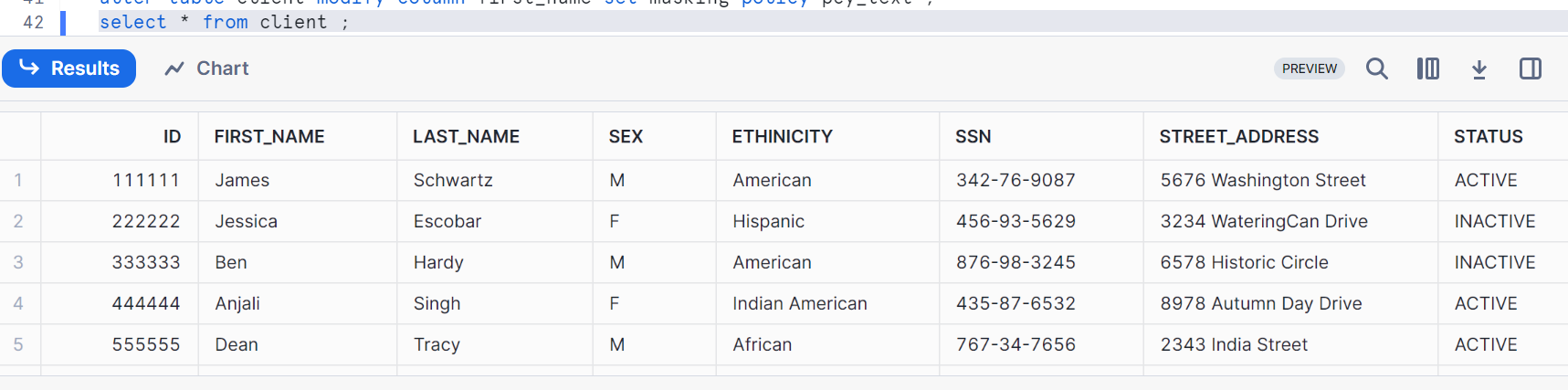
Create a simple masking policy



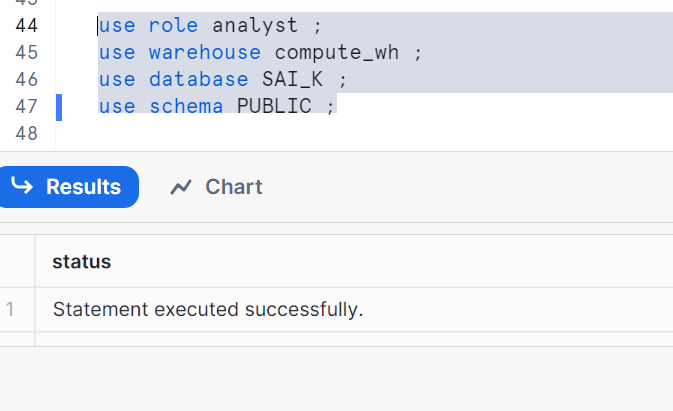


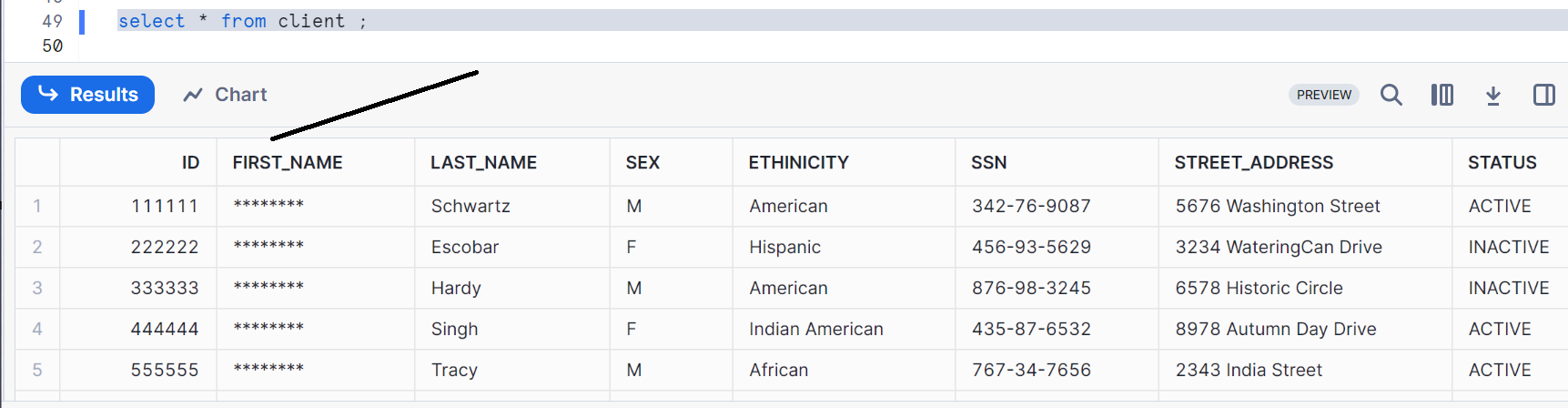
Now, apply this policy on a table column



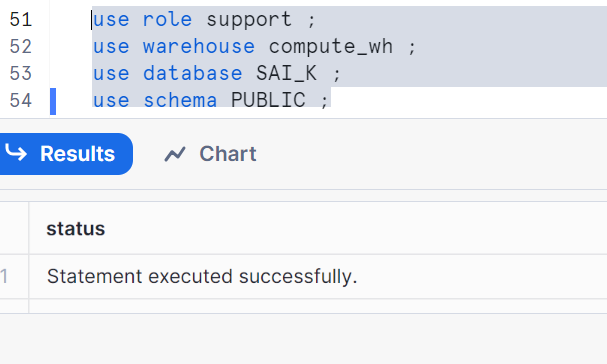


Set the role as ANALYST

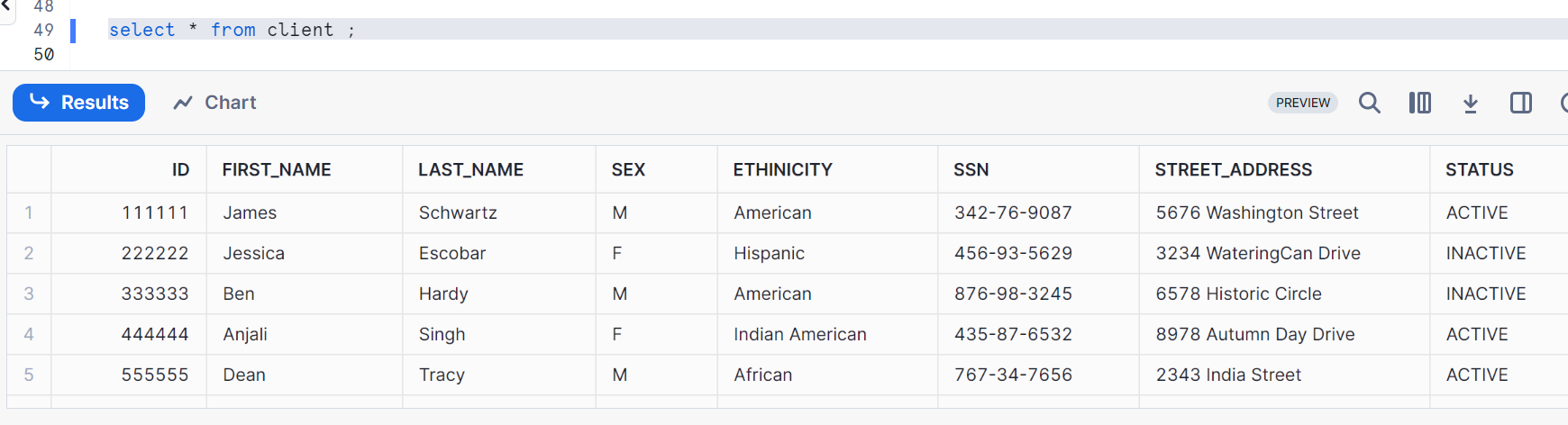




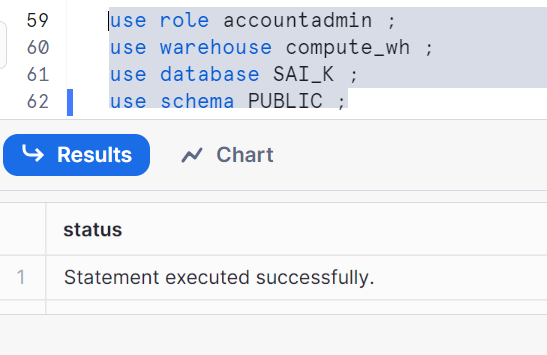
Set the role as support

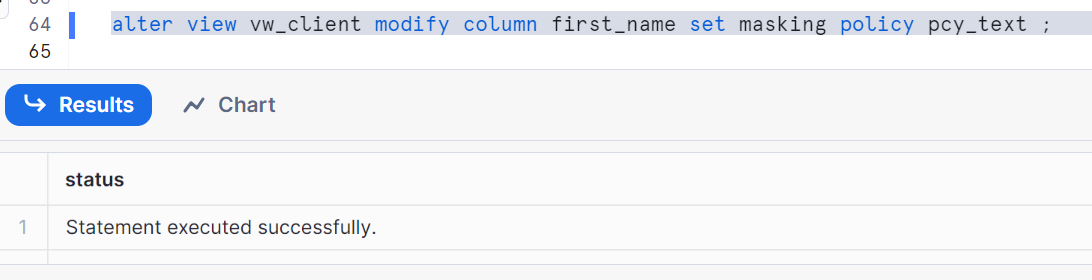


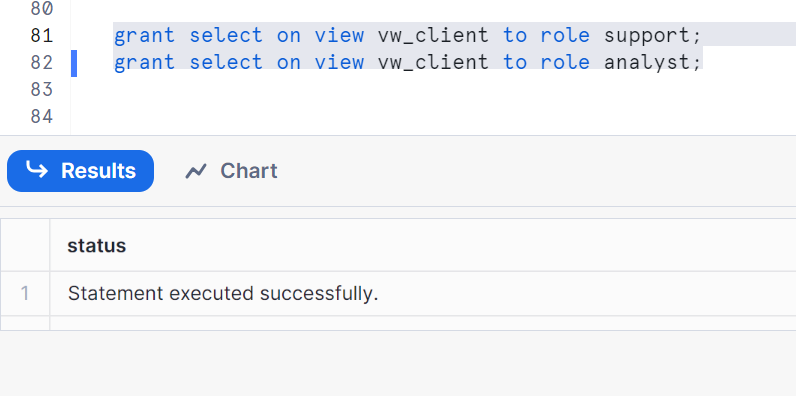
As it’s not authorized, the data will not mask



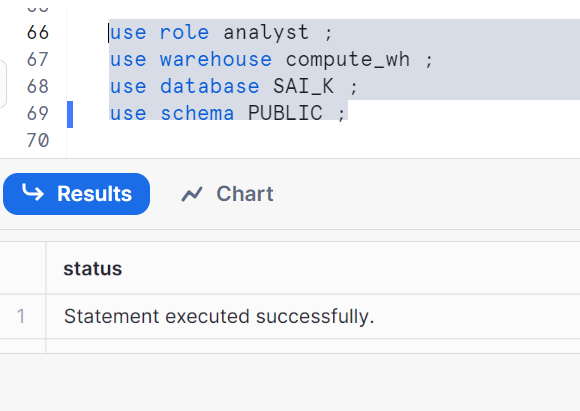
Now let’s see on view column

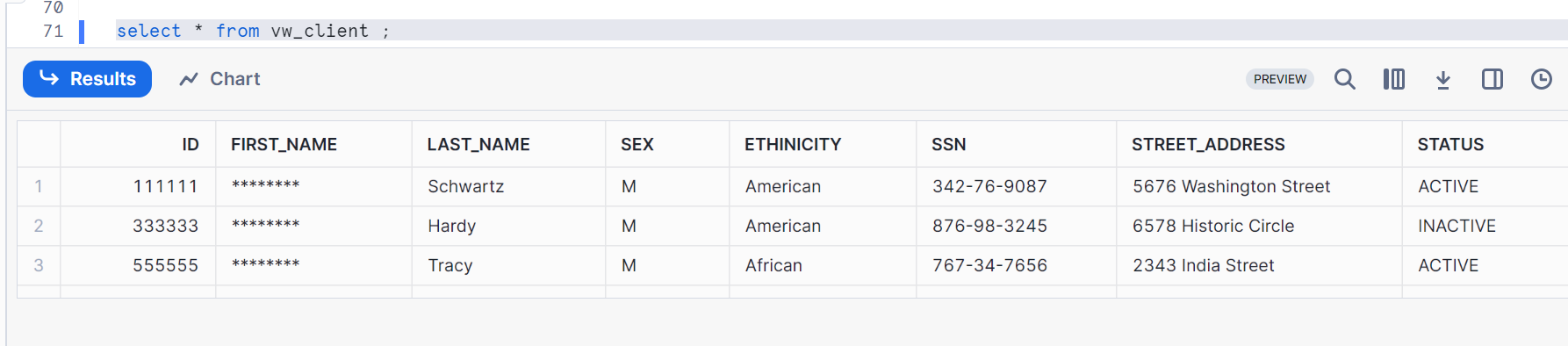




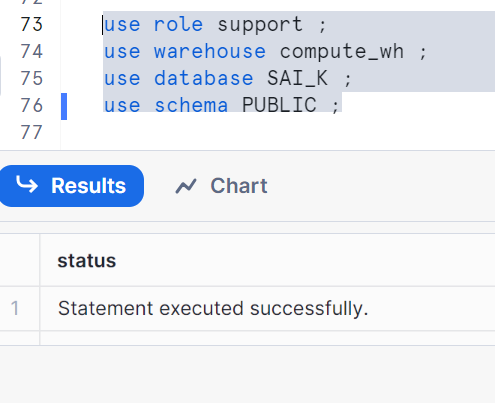


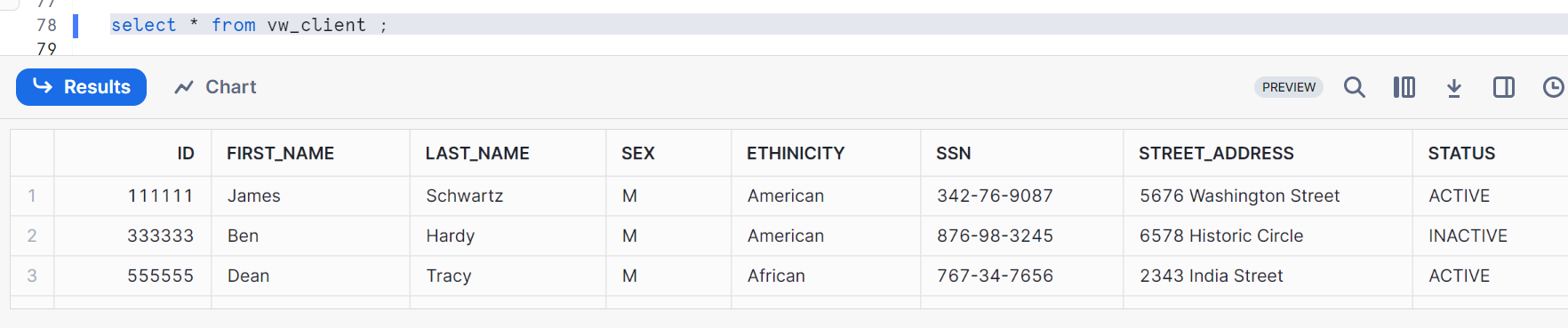
Set role to see data





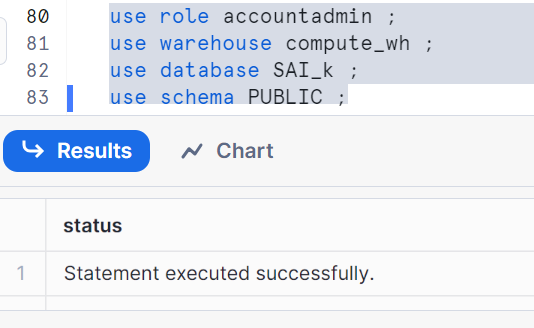
Set support role





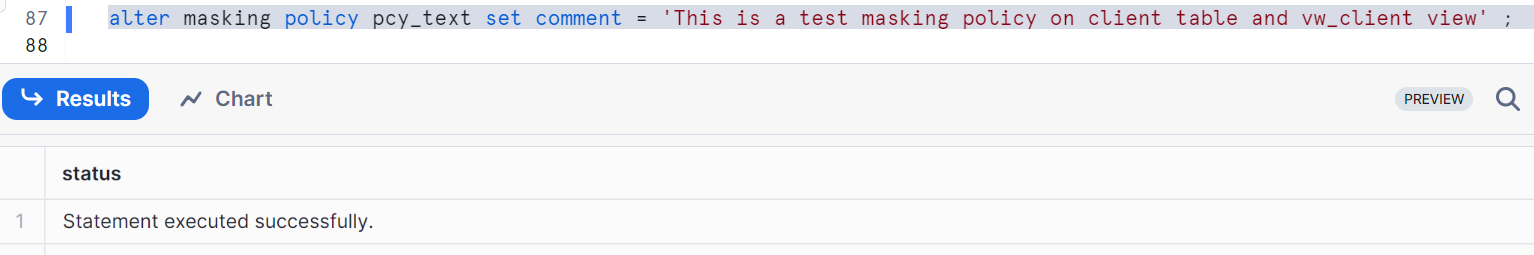
**Modify Masking Policies:**

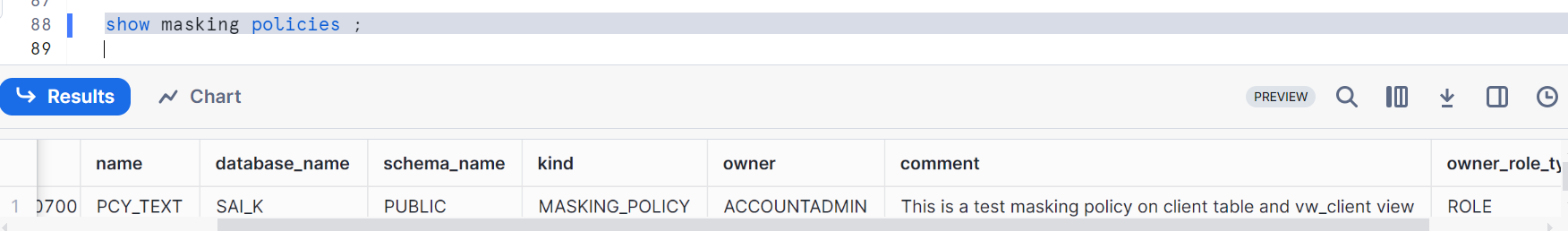
Let’s use account admin

****

****

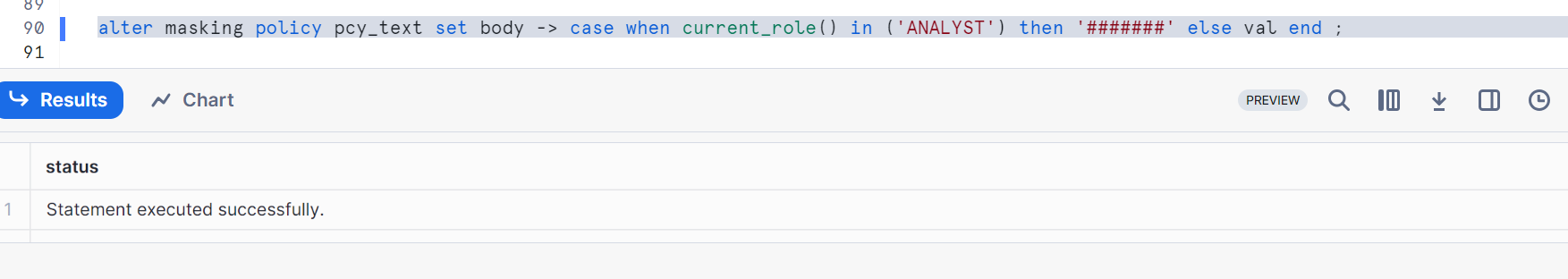
Altering comment in the masking policy

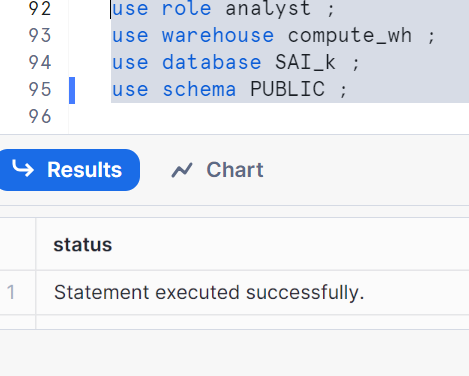
****

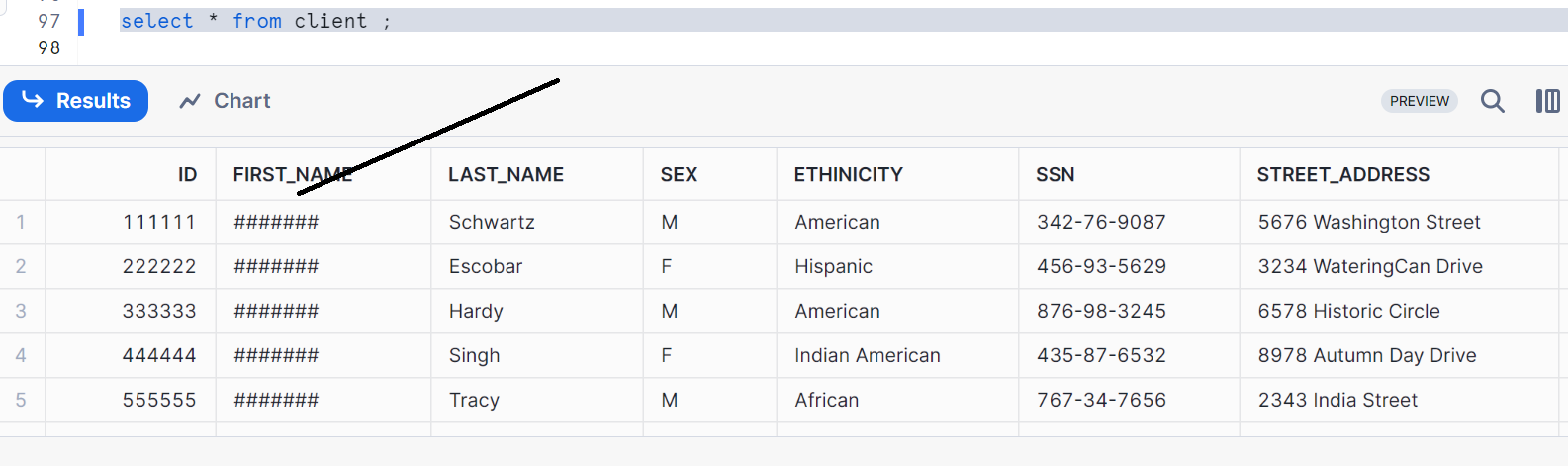
****

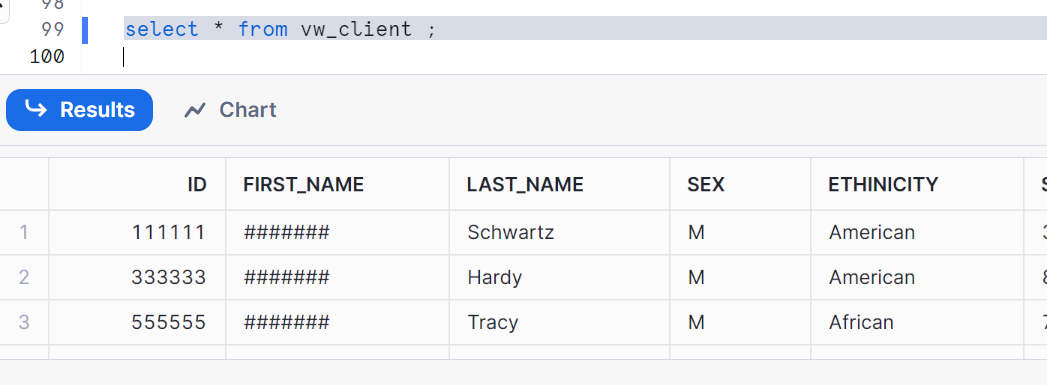
It doesn’t affect any even if you go to other roles (please check by going into other roles)

Now, let’s change the masking policy in account admin

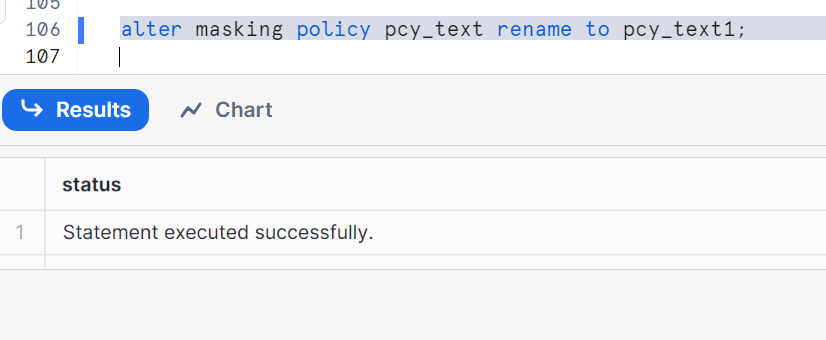
****

****

****

****

Now, let’s go to account admin and rename masking policy

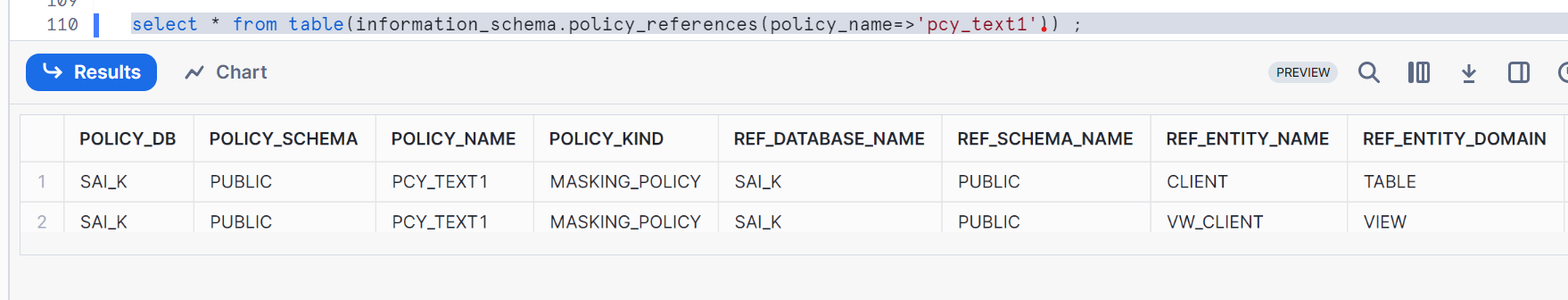




Even if you rename the masking policy, nothing will be affected (please check by changing the role and check whether data is masked or not)

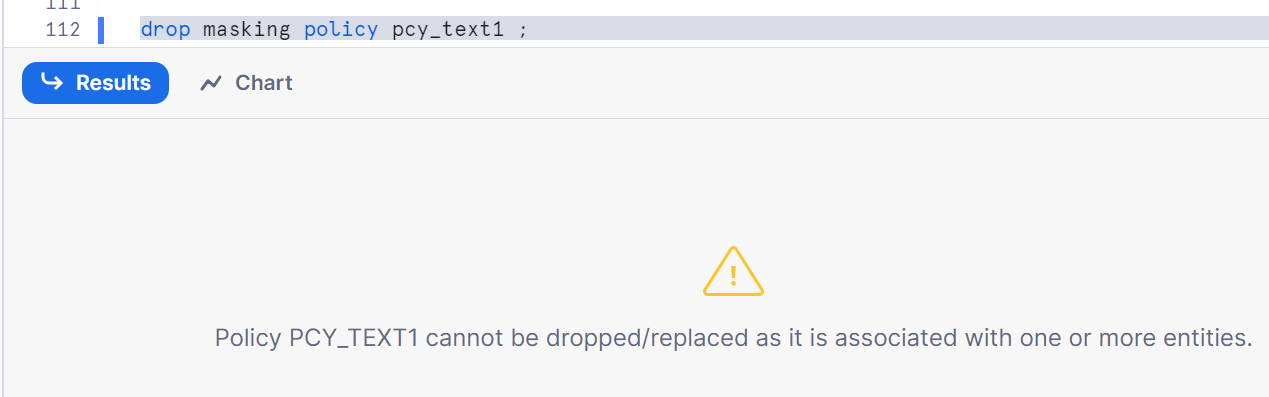
**How to check which columns and table the masking policy is applied to??**

Please follow the below syntax to check

****

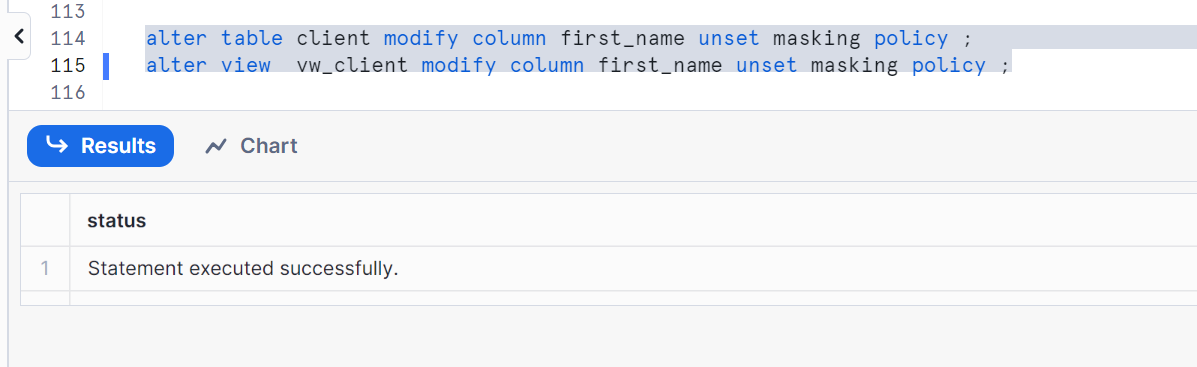
**Drop Masking Policies, Multiple columns policies and Nested policies:**

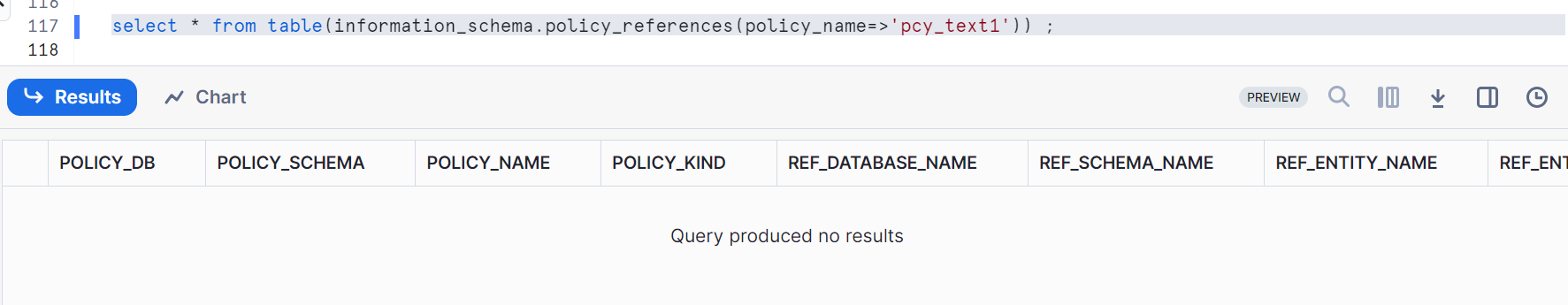
If you use the below syntax to drop, it doesn’t workout

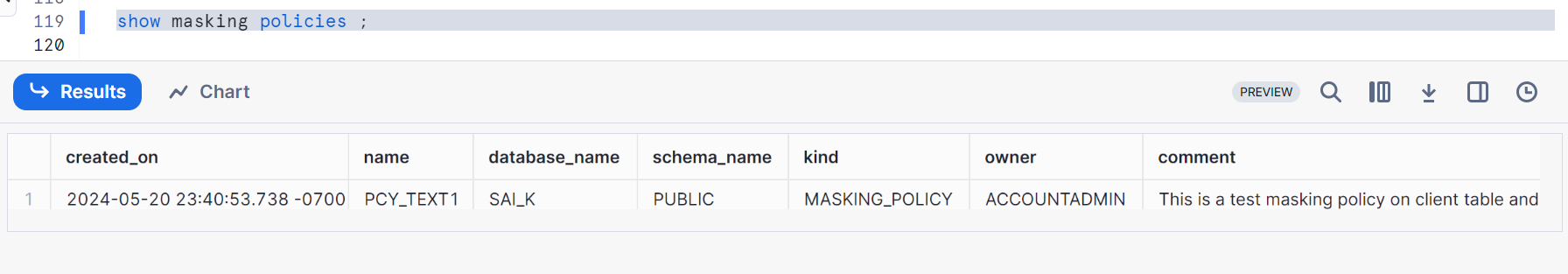


Hence we need to unset it first, we check the objects on which the policy is currently applied

Let’s unset for both table and view

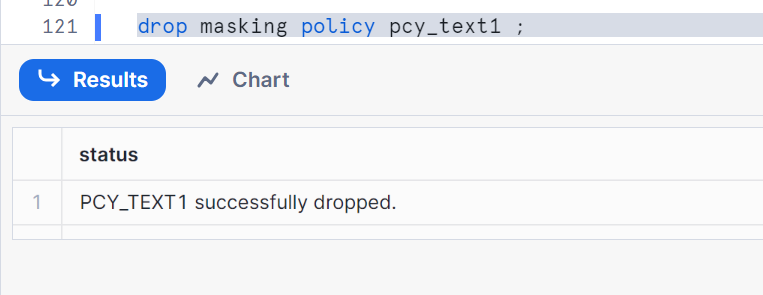


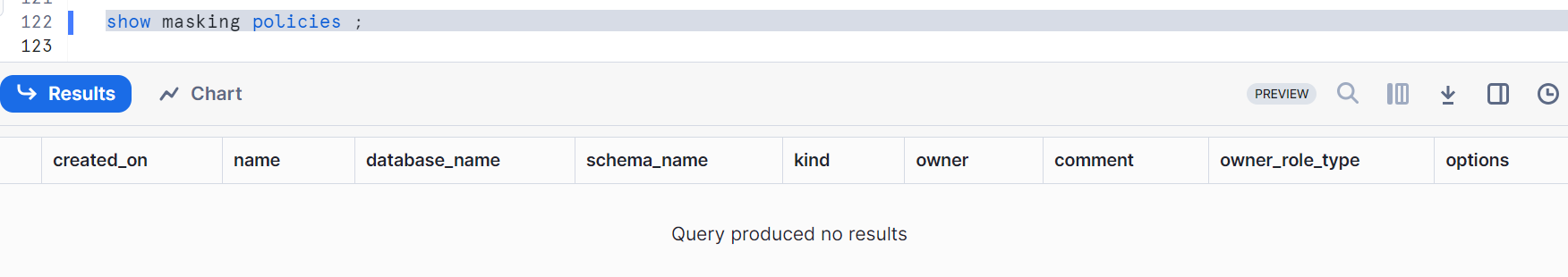




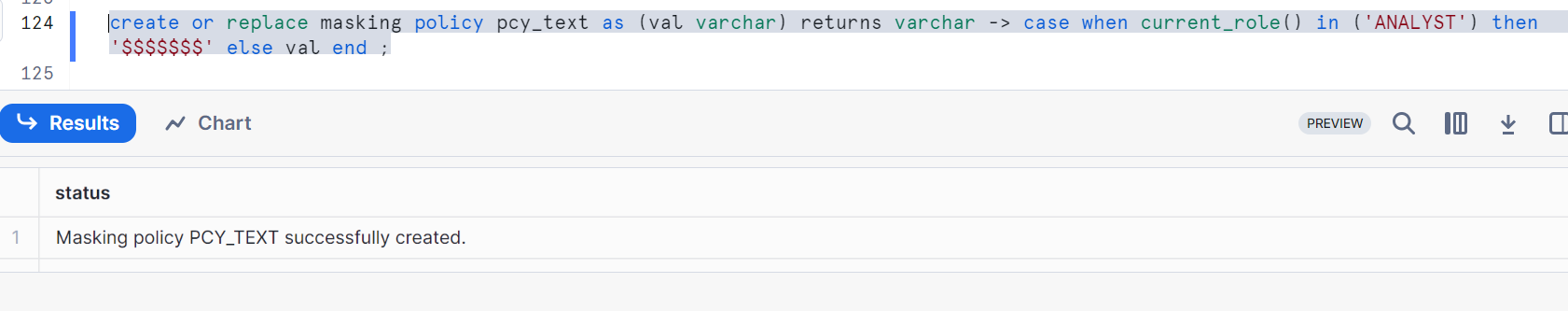
We still have masking policy

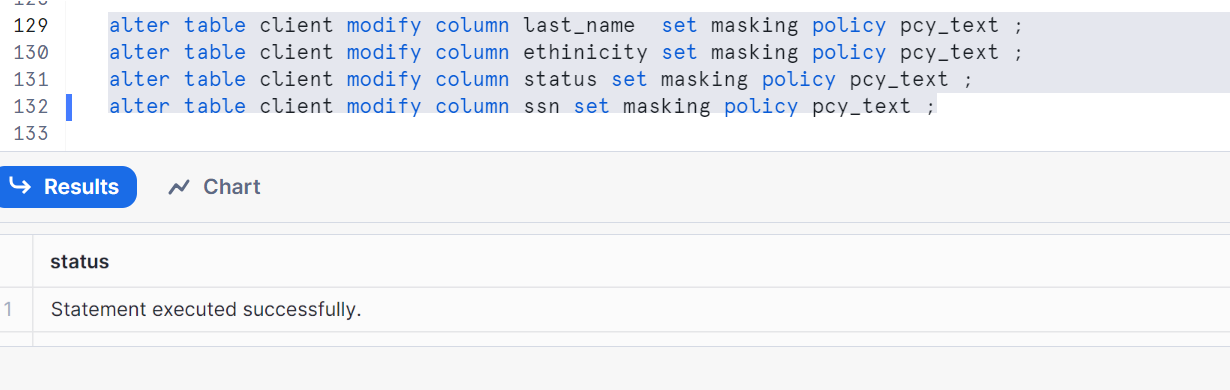
So, now we can drop the policy by below syntax

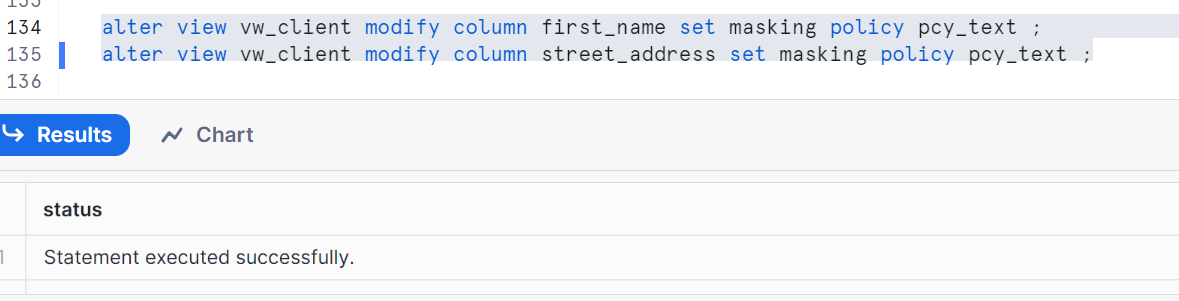


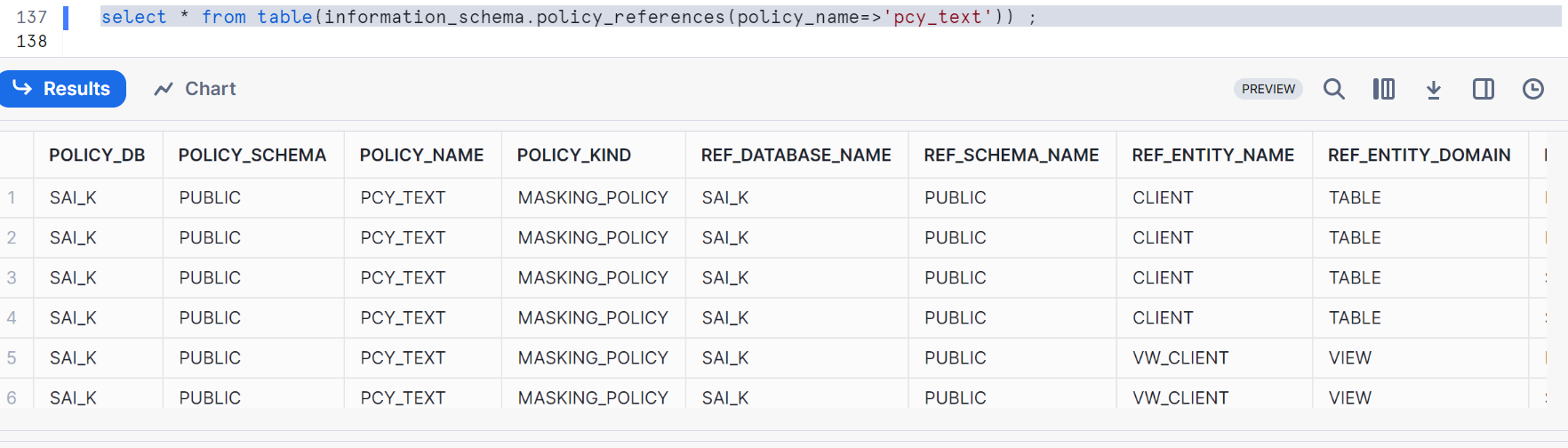


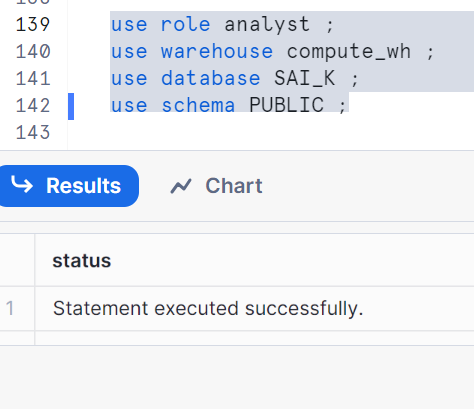
**Recreating the masking policies and apply on multiple columns for tables and views**

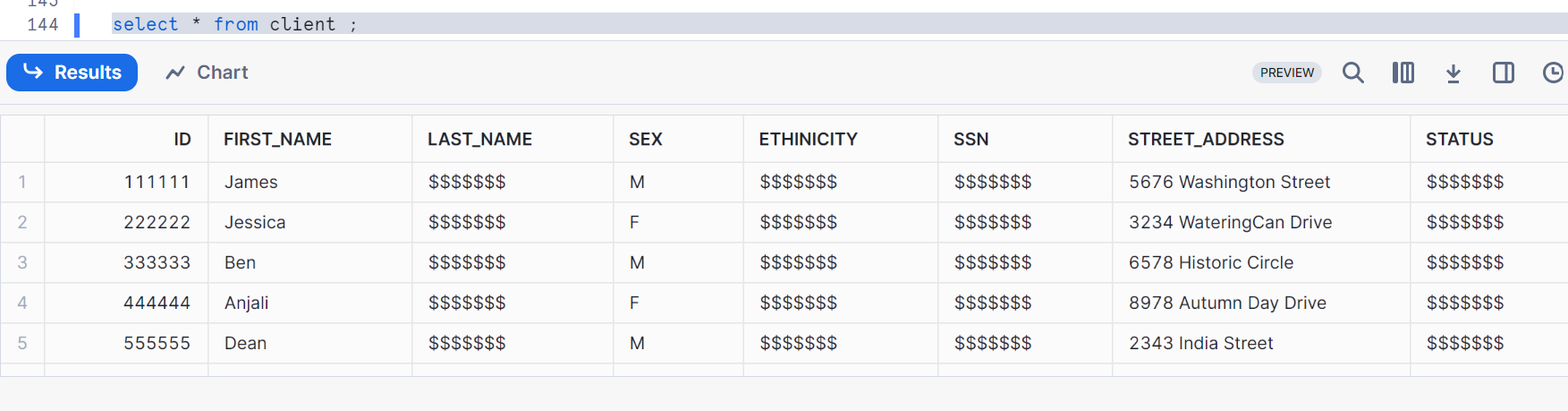
****

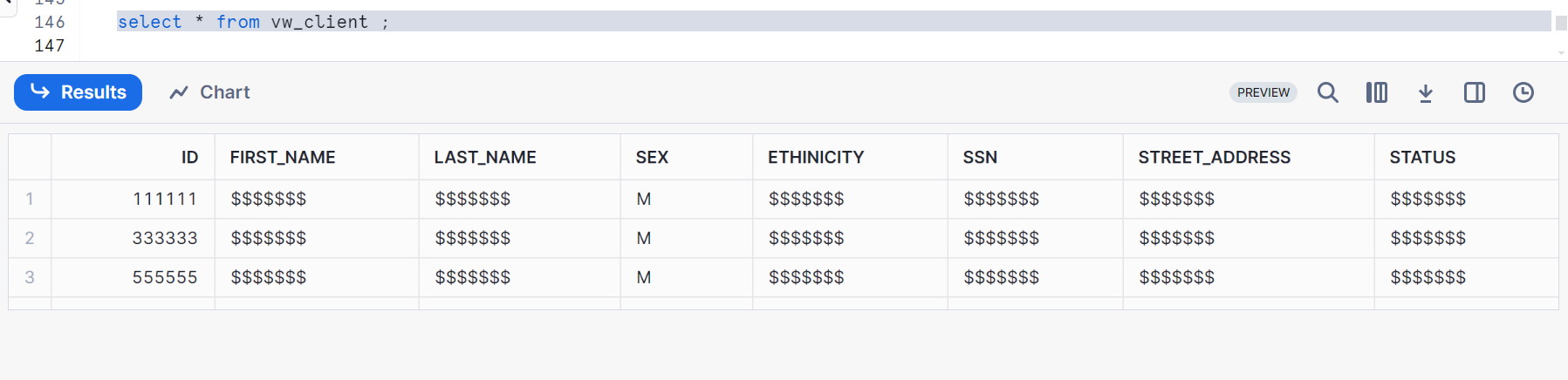
****





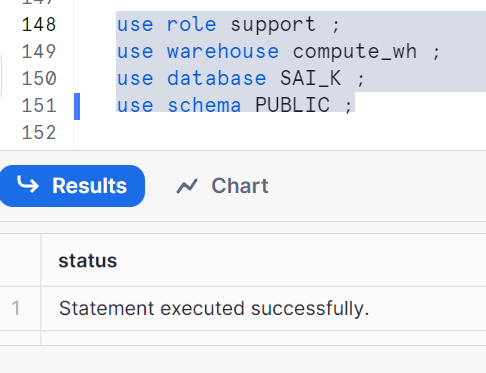


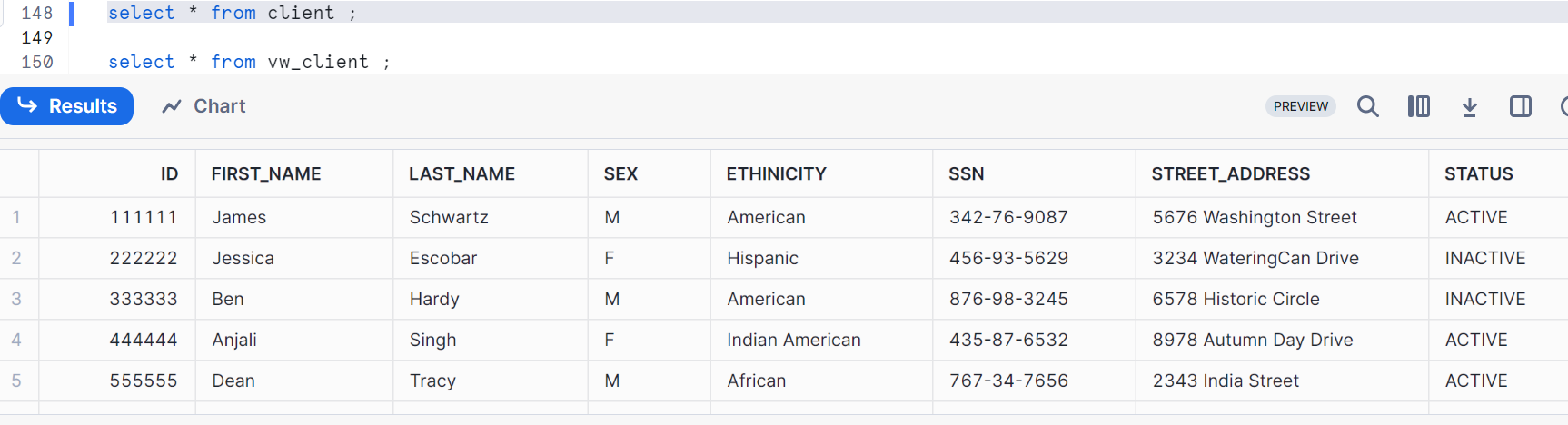
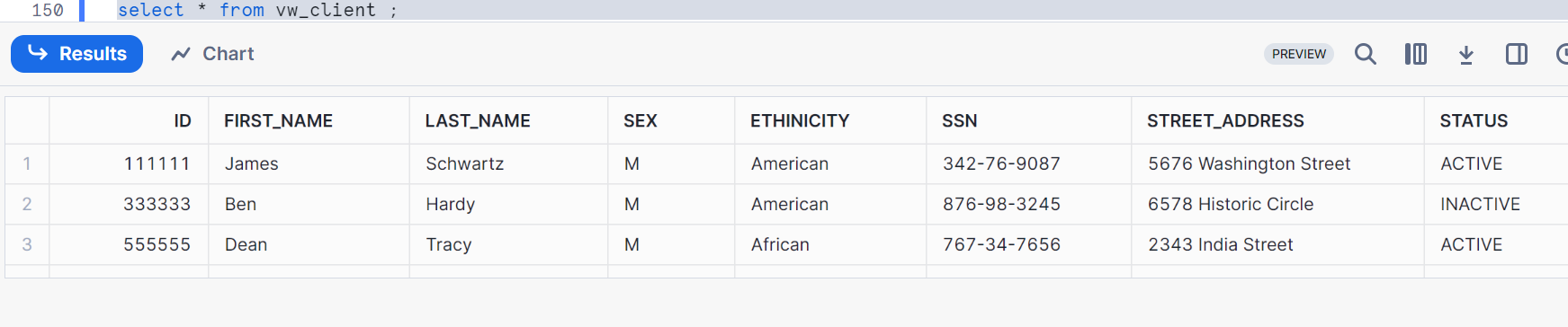




So, as we applied masking policy for two columns in view but we got for 6 columns. While creating a view, we used the table client and hence due to this, masking policy got inherited. Hence, it is called nested masking

Change role and check data

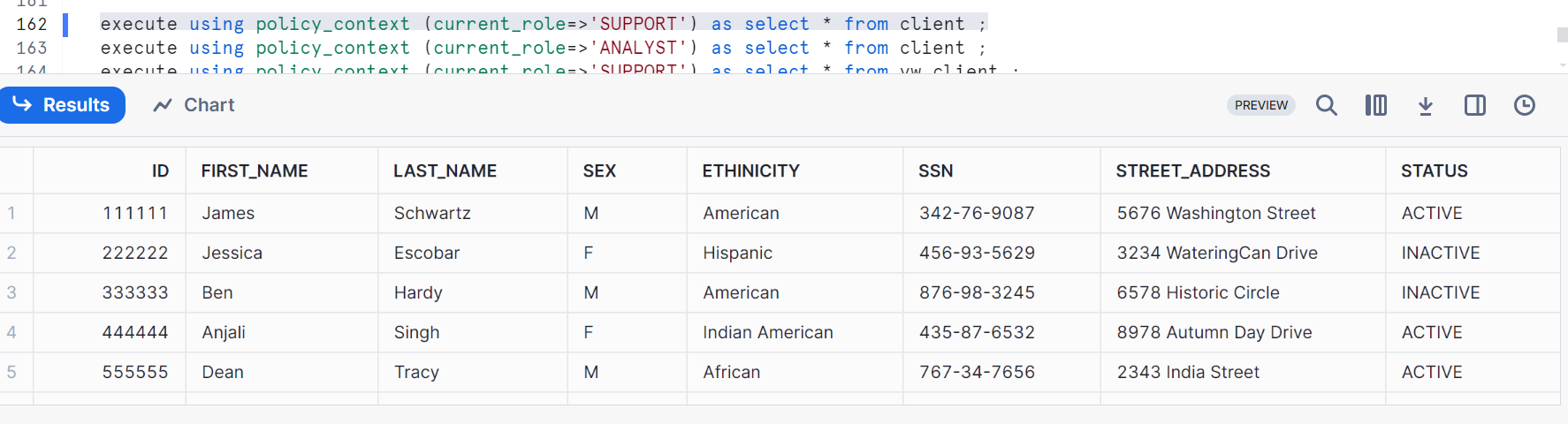


**Policy context:**

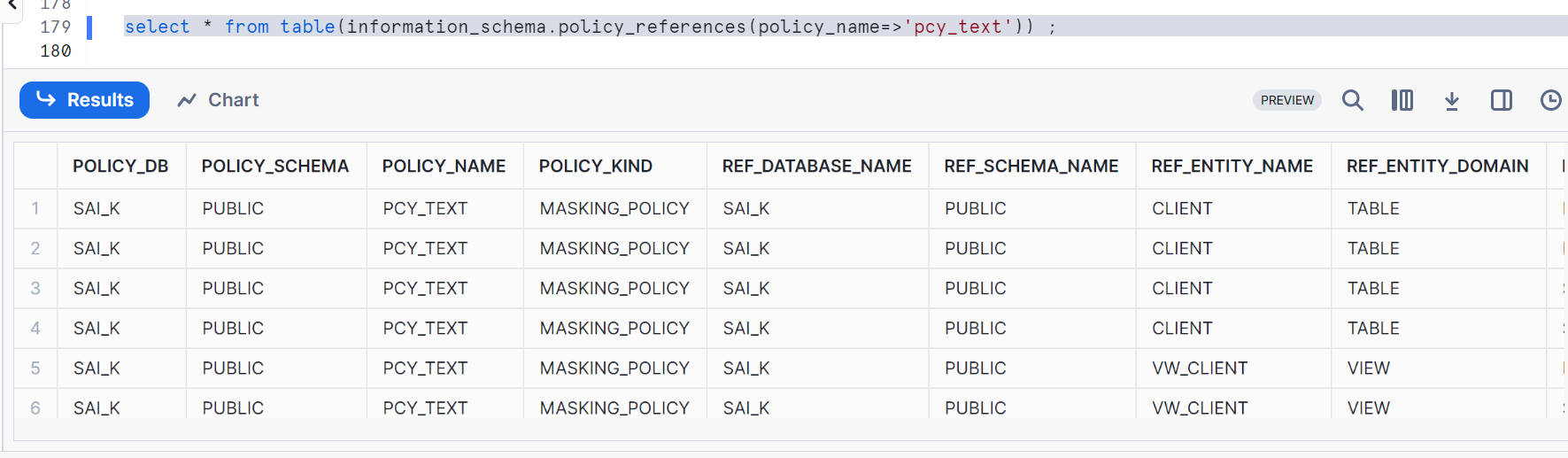
We have been setting the roles before doing a select to validate the masking policy, policy context simulates the query results

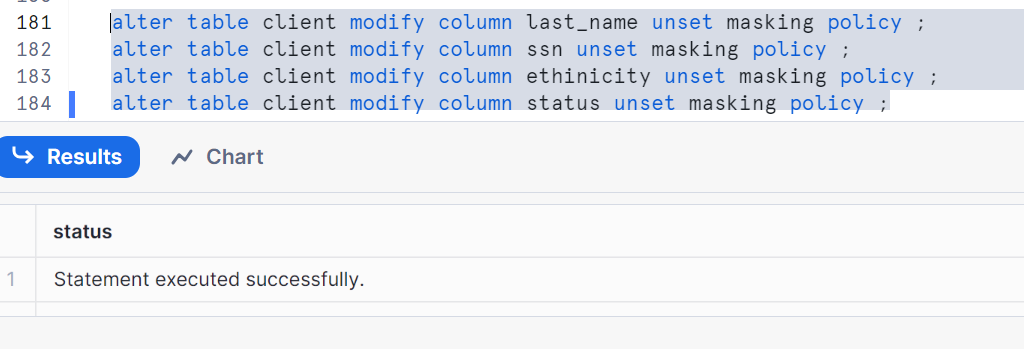
In simple words, being in account admin and operating in another role

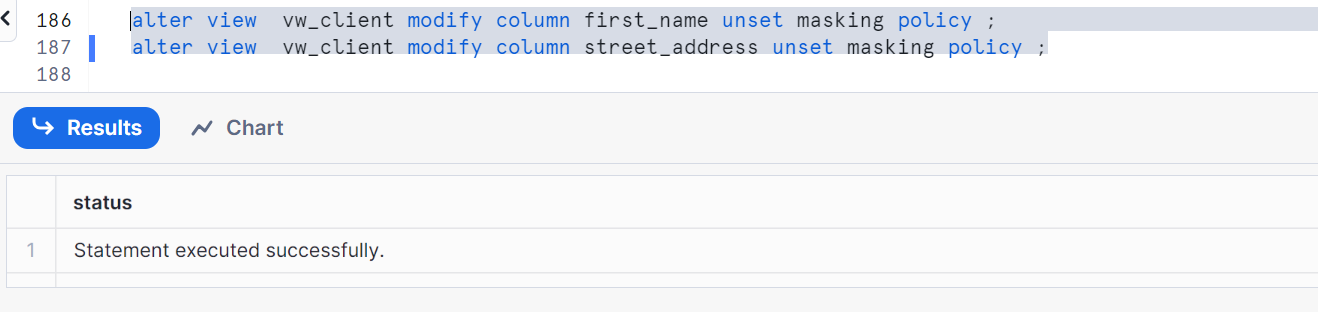




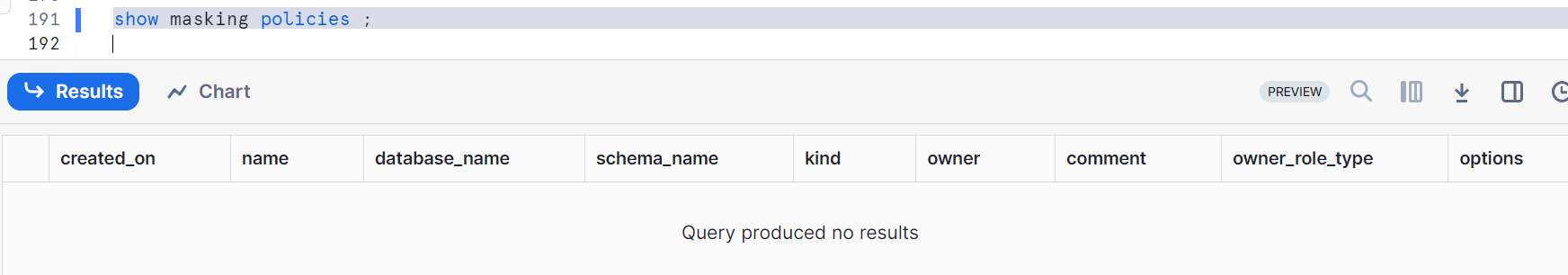
Let’s drop the masking policy





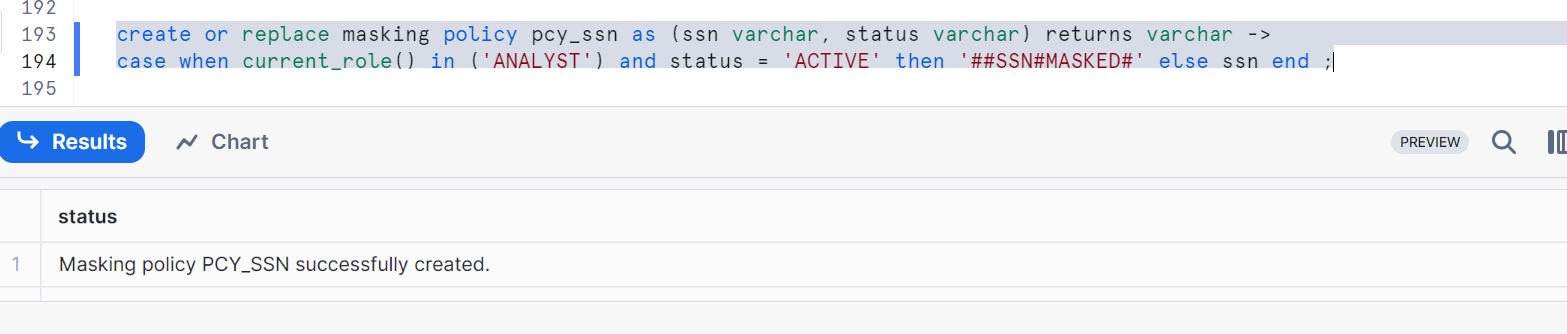






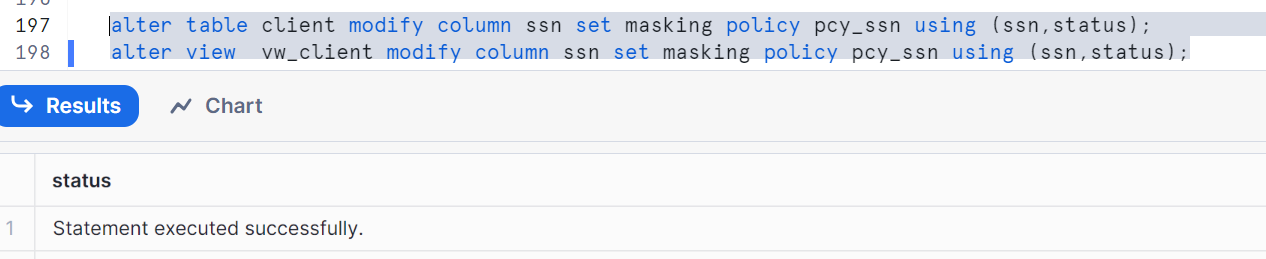
**Creating conditional masking policy:**

Masking will be based on a condition  
here with an example



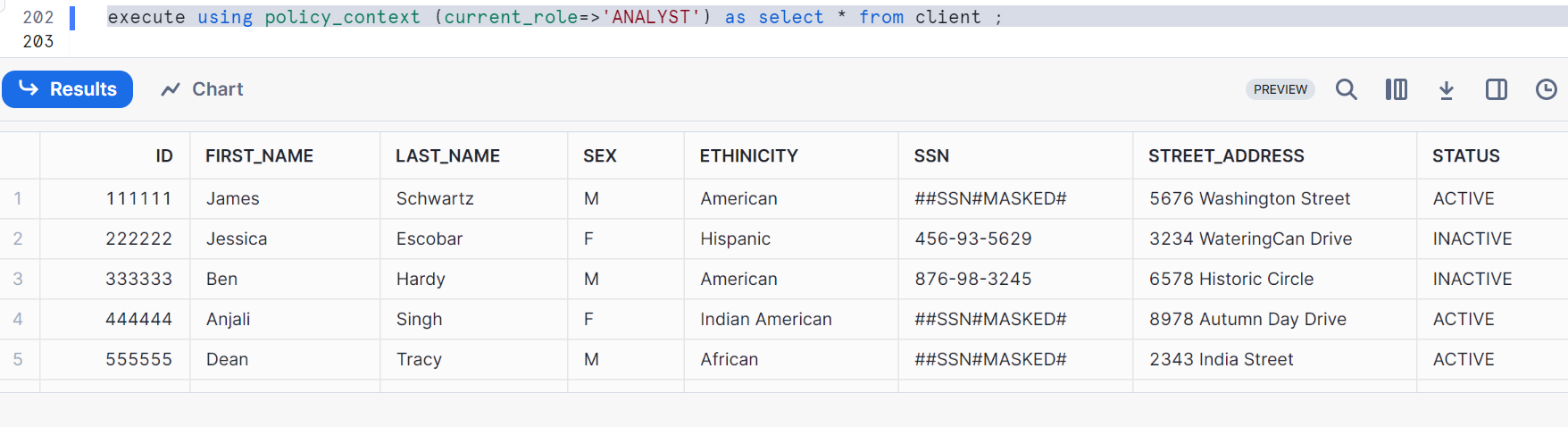


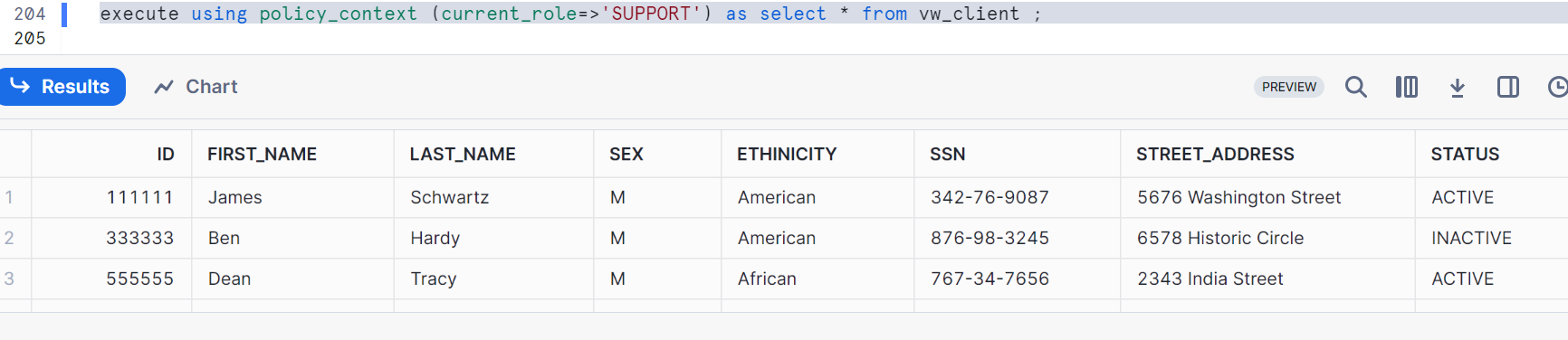
Pay attention to the using clause while applying this policy

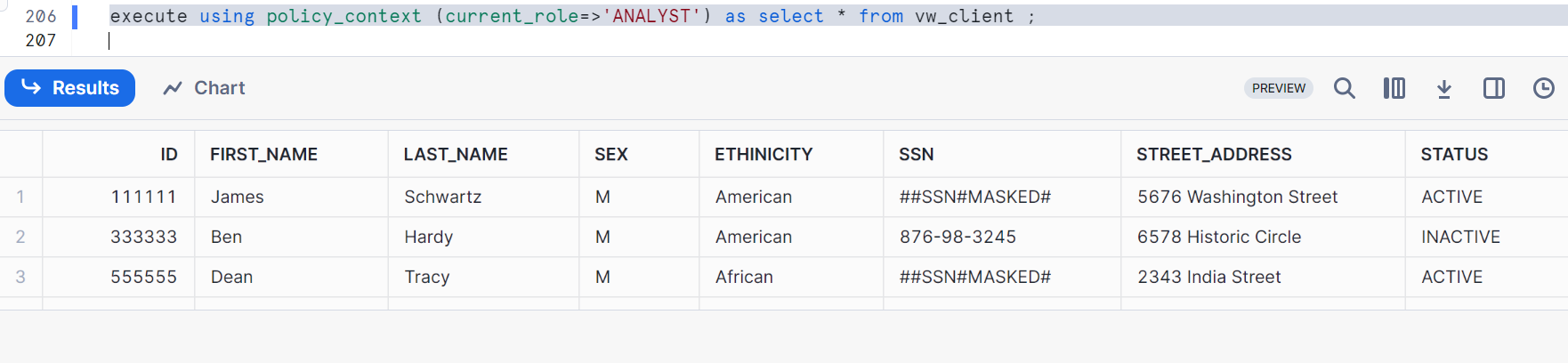


Let’s validate the policy









Clean up:

Drop table client;

Drop view vw\_client;

Drop masking policy pcy\_ssn;

Show masking policies;