Context

Tracking read/write operations

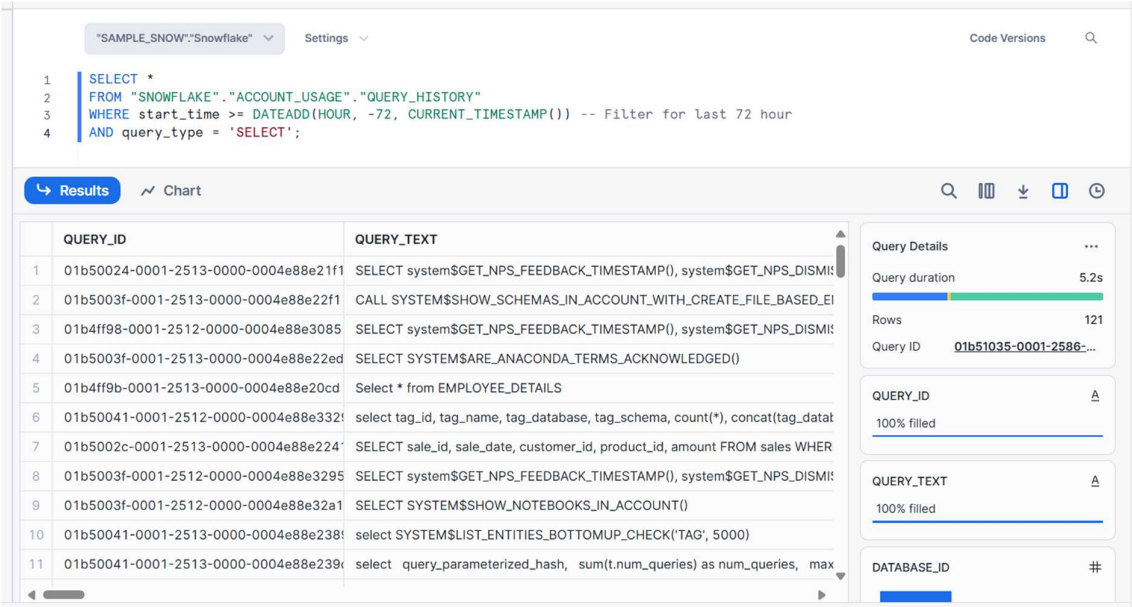
1. Querying QUERY_HISTORY view:
2. This view contains a record of all queries executed in your account.
3. You can filter queries based on their type (e.g., SELECT for read operations, INSERT/UPDATE/DELETE for write operations).
4. Query to find recent read operations.

Query: SELECT *

FROM "SNOWFLAKE"."ACCOUNT_USAGE"."QUERY_HISTORY"

WHERE start_time >= DATEADD(HOUR, -72, CURRENT_TIMESTAMP()) -- Filter for last 72 hour

AND query_type = 'SELECT';



5. Query to find recent write operations.

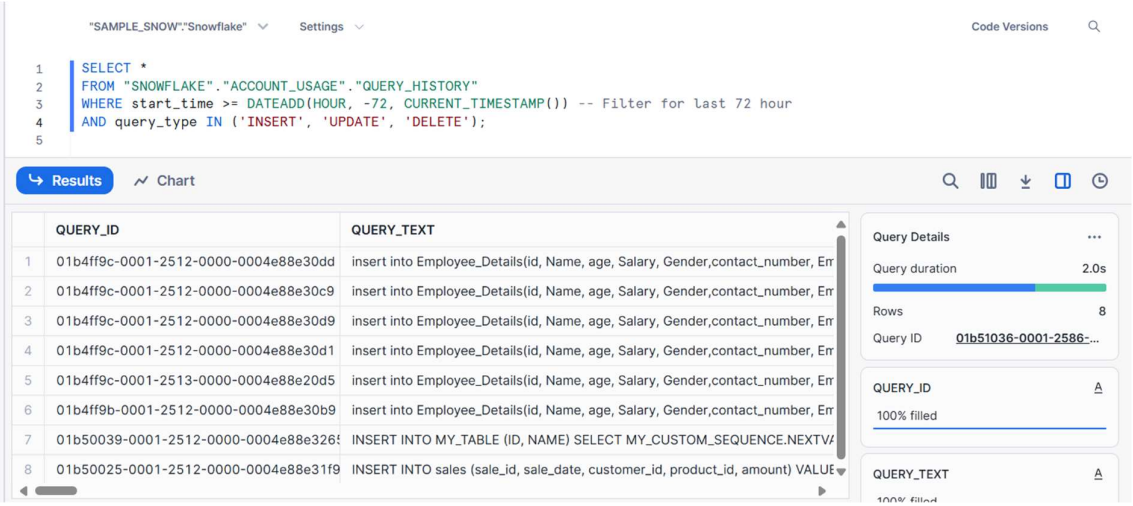
Query: SELECT *

FROM "SNOWFLAKE"."ACCOUNT_USAGE"."QUERY_HISTORY"

WHERE start_time >= DATEADD(HOUR, -72, CURRENT_TIMESTAMP()) -- Filter for last 72 hour

AND query_type IN ('INSERT', 'UPDATE', 'DELETE');

6. Check the data.

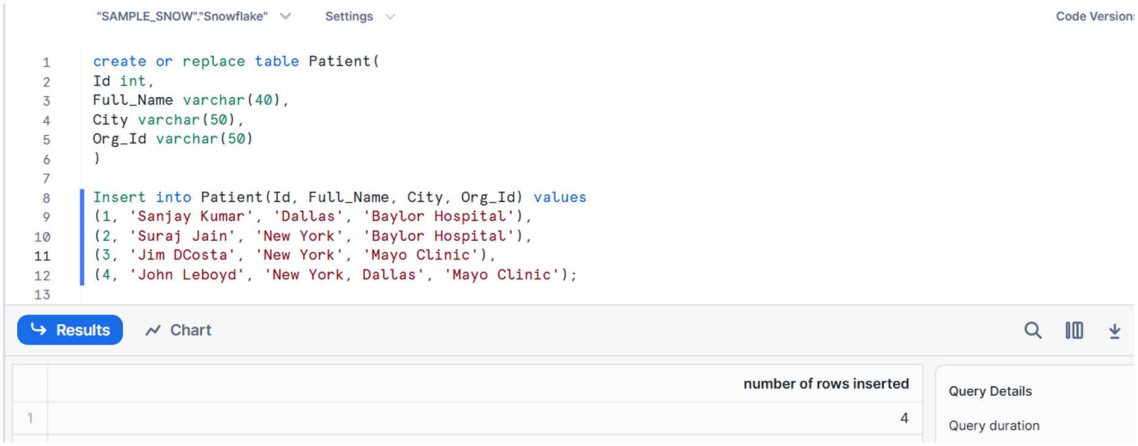


Secure View

1. Create a table and insert the data with the below query.

Query: create or replace table Patient(
Id int,
Full_Name varchar(40),
City varchar(50),
Org_Id varchar(50)
)

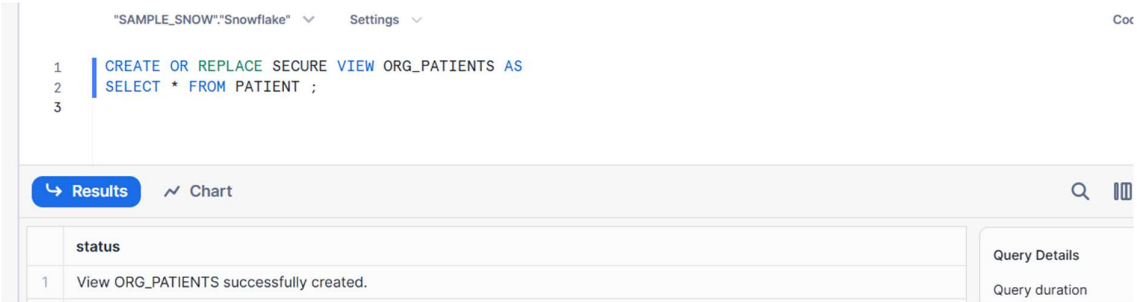
Insert into Patient(Id, Full_Name, City, Org_Id) values
(1, 'Sanjay Kumar', 'Dallas', 'Baylor Hospital'),
(2, 'Suraj Jain', 'New York', 'Baylor Hospital'),
(3, 'Jim DCosta', 'New York', 'Mayo Clinic'),
(4, 'John Leboyd', 'New York, Dallas', 'Mayo Clinic');



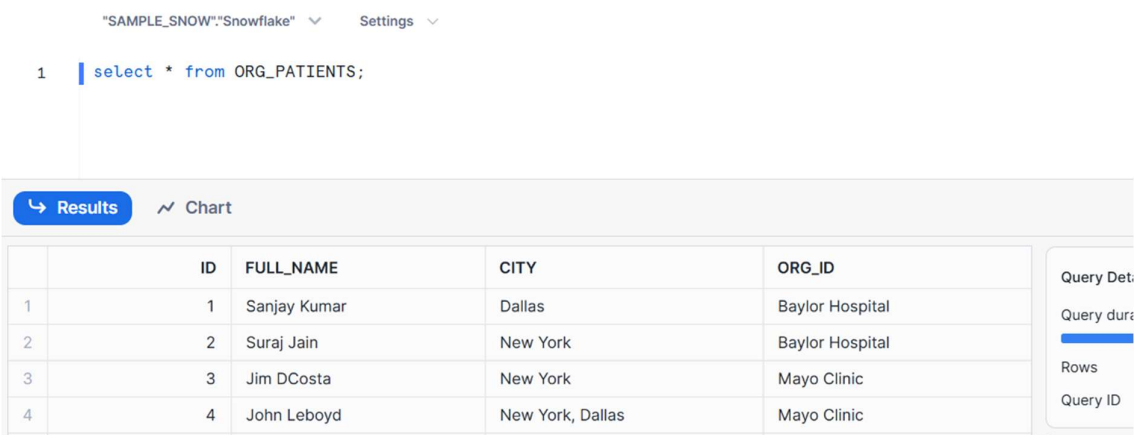
2. Create a secure view with the below query.

Query: CREATE OR REPLACE SECURE VIEW ORG_PATIENTS AS

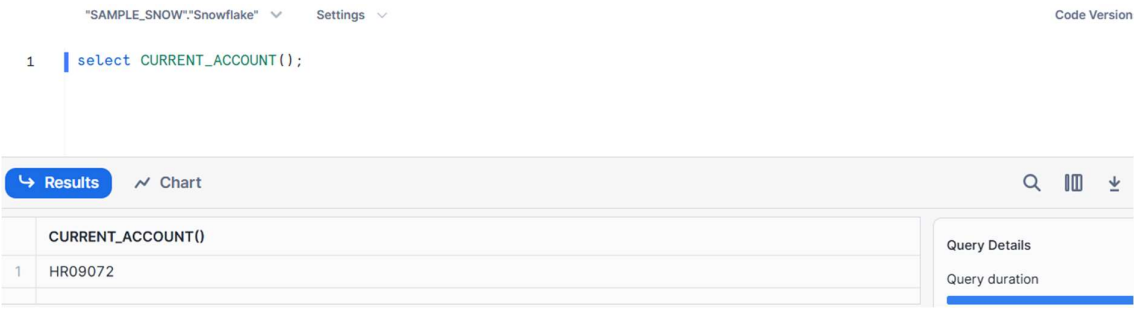
SELECT * FROM PATIENT ;



3. Check the view data.



4. Select the current account. Copy the account name.



5. Create a User Access table with the below query. Here give your account name for the Baylor Hospital.

Query: create table USER_ACCESS
(
  ORG_ID VARCHAR(50),
  ACCOUNT_ID varchar(50)
);

insert into USER_ACCESS values ('Baylor Hospital', HR09072);
insert into USER_ACCESS values ('Mayo Clinic','PO51568');

"SAMPLE_SNOW"."Snowflake" Settings Code Versions

```
Create table USER_ACCESS
(
    ORG_ID VARCHAR(50),
    ACCOUNT_ID varchar(50)
);

insert into USER_ACCESS values ('Baylor Hospital','HR09072');
insert into USER_ACCESS values ('Mayo Clinic','PO51568');
```

Results Chart

	number of rows inserted
1	1

Query Details
Query duration 43

6. Create a secure view using both the tables.

Query: CREATE OR REPLACE SECURE VIEW ORG_PATIENTS AS

SELECT FULL_NAME,CITY,USER_ACCESS.ORG_ID FROM patient

INNER JOIN USER_ACCESS ON PATIENT.ORG_ID=USER_ACCESS.ORG_ID

WHERE ACCOUNT_ID=CURRENT_ACCOUNT();

"SAMPLE_SNOW"."Snowflake" Settings Code Versions

```
CREATE OR REPLACE SECURE VIEW ORG_PATIENTS AS
SELECT FULL_NAME,CITY,USER_ACCESS.ORG_ID FROM patient
INNER JOIN USER_ACCESS ON PATIENT.ORG_ID=USER_ACCESS.ORG_ID
WHERE ACCOUNT_ID=CURRENT_ACCOUNT();
```

Results Chart

	status
1	View ORG_PATIENTS successfully created.

Query Details
Query duration 137ms

7. Now check the data.

"SAMPLE_SNOW"."Snowflake" Settings Code Versions

```
select * from ORG_PATIENTS;
```

Results Chart

	FULL_NAME	CITY	ORG_ID
1	Sanjay Kumar	Dallas	Baylor Hospital
2	Suraj Jain	New York	Baylor Hospital

Query Details
Query duration 777ms
Rows 2

Row-level Security

Row-Level Security is a security mechanism that limits the records returned from a database table based on the permissions provided to the currently logged-in user. Typically, this is done such that certain users can access only their data and are not permitted to view the data of other users.

Create a table to apply Row-Level Security

1. Let us consider a sample employee table as an example for the demonstration of row-level security using secure views.
2. The below SQL statements create a table named employees with required sample data in the HR schema of the analytics database.

Query: use role SYSADMIN;

```
create or replace database analytics;

create or replace schema analytics.hr;

create or replace table analytics.hr.employees(
   employee_id number,
   first_name varchar(50),
   last_name varchar(50),
   email varchar(50),
   hire_date date,
   country varchar(50)
);

INSERT INTO
analytics.hr.employees(employee_id,first_name,last_name,email,hire_date,country) VALUES
(100,'Steven','King','SKING@outlook.com','2013-06-17','US'),
(101,'Neena','Kochhar','NKOCHHAR@outlook.com','2015-09-21','US'),
(102,'Lex','De Haan','LDEHAAN@outlook.com','2011-01-13','US'),
(103,'Alexander','Hunold','AHUNOLD@outlook.com','2016-01-03','UK'),
(104,'Bruce','Ernst','BERNST@outlook.com','2017-05-21','UK'),
(105,'David','Austin','DAUSTIN@outlook.com','2015-06-25','UK'),
(106,'Valli','Pataballa','VPATABAL@outlook.com','2016-02-05','CA'),
(107,'Diana','Lorentz','DLORENTZ@outlook.com','2017-02-07','CA'),
(108,'Nancy','Greenberg','NGREENBE@outlook.com','2012-08-17','CA')
;
```

ANALYTICS.HR Settings

```
use role SYSADMIN;

create or replace database analytics;

create or replace schema analytics.hr;

create or replace table analytics.hr.employees(
    employee_id number,
    first_name varchar(50),
    last_name varchar(50),
    email varchar(50),
    hire_date date,
    country varchar(50)
);

INSERT INTO analytics.hr.employees(employee_id,first_name,last_name,email,hire_date,country) VALUES
(100,'Steven','King','SKING@outlook.com','2013-06-17','US'),
(101,'Neena','Kochhar','NKOCHHAR@outlook.com','2015-09-21','US'),
(102,'Lex','De Haan','LDEHAAN@outlook.com','2011-01-13','US'),
```

Results Chart

	number of rows inserted
1	9

Create a Role Mapping table

3. The below SQL statements create mapping table named role_mapping which stores the country and corresponding role to be assigned for the users of that country as shown below.

Query: use role SYSADMIN;

```
create or replace table analytics.hr.role_mapping(
   country varchar(50),
   role_name varchar(50)
);

INSERT INTO analytics.hr.role_mapping(country, role_name) VALUES
('US','DATA_ANALYST_ROLE_US'),
('UK','DATA_ANALYST_ROLE_UK'),
('CA','DATA_ANALYST_ROLE_CA')
;
```

ANALYTICS.HR Settings

```
use role SYSADMIN;

create or replace table analytics.hr.role_mapping(
    country varchar(50),
    role_name varchar(50)
);

INSERT INTO analytics.hr.role_mapping(country, role_name) VALUES
('US','DATA_ANALYST_ROLE_US'),
('UK','DATA_ANALYST_ROLE_UK'),
('CA','DATA_ANALYST_ROLE_CA')
;
```

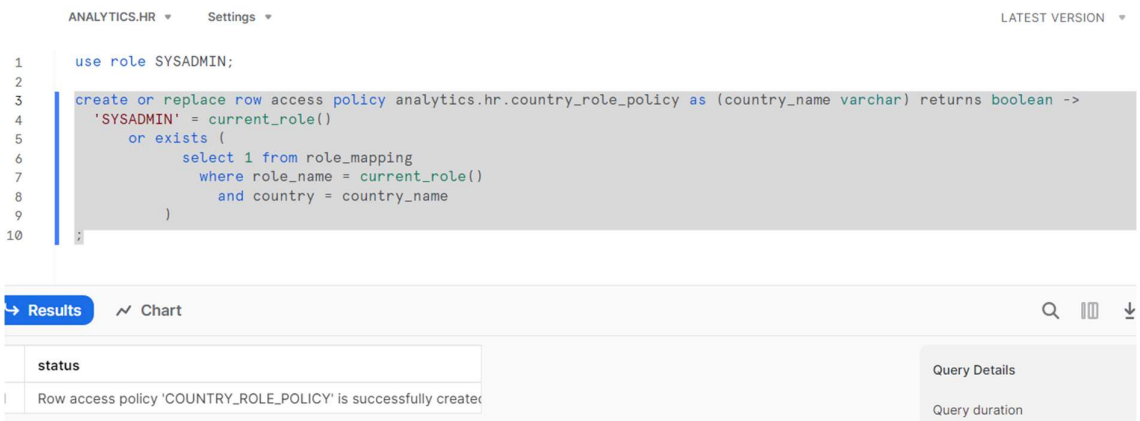
Results Chart

	number of rows inserted
1	3

Create a Row Access Policy

4. The below SQL statement creates a Row Access Policy with following two conditions.
5. User with SYSADMIN role can query all rows of the table.
6. User with DATA_ANALYST roles can query only rows belonging to their country based on the role mapping table.

```
Query: use role SYSADMIN;
create or replace row access policy analytics.hr.country_role_policy as (country_name
varchar) returns boolean ->
 'SYSADMIN' = current_role()
   or exists (
       select 1 from role_mapping
         where role_name = current_role()
          and country = country_name
      )
;
```



7. In the above statement:
8. **country_role_policy** specifies the name of the policy.
9. **country_name** is the signature of the row access policy which specifies the field and data type of the mapping table to which it links.
10. **returns boolean ->** specifies the application of the row access policy.
11. **'SYSADMIN' = current_role()** is the first condition of row access policy which allows users with SYSDAMIN role to view all rows of the table.
12. **or exists …** is the second condition of the row access policy expression which uses a subquery. The subquery requires the CURRENT_ROLE to be the custom role which specifies the country through role mapping table. This is used by row access policy to limit the rows to be returned for the query executed by user.

Add the Row Access Policy to a table

13. The below SQL statement adds the row access policy named country_role_policy to the table employees on country field.

Query: use role SYSADMIN;
alter table analytics.hr.employees
add row access policy analytics.hr.country_role_policy on (country);

ANALYTICS.HR Settings

```
use role SYSADMIN;

alter table analytics.hr.employees
add row access policy analytics.hr.country_role_policy on (country);
```

Results Chart

	status
1	Statement executed successfully.

Create Custom Roles and their Role Hierarchy

14. The below SQL statements creates custom roles mentioned in the role mapping table to assign to the users in later stage.

Query: use role SECURITYADMIN;

create or replace role DATA_ANALYST_ROLE_US;
create or replace role DATA_ANALYST_ROLE_UK;
create or replace role DATA_ANALYST_ROLE_CA;

ANALYTICS.HR Settings

```
use role SECURITYADMIN;

create or replace role DATA_ANALYST_ROLE_US;
create or replace role DATA_ANALYST_ROLE_UK;
create or replace role DATA_ANALYST_ROLE_CA;
```

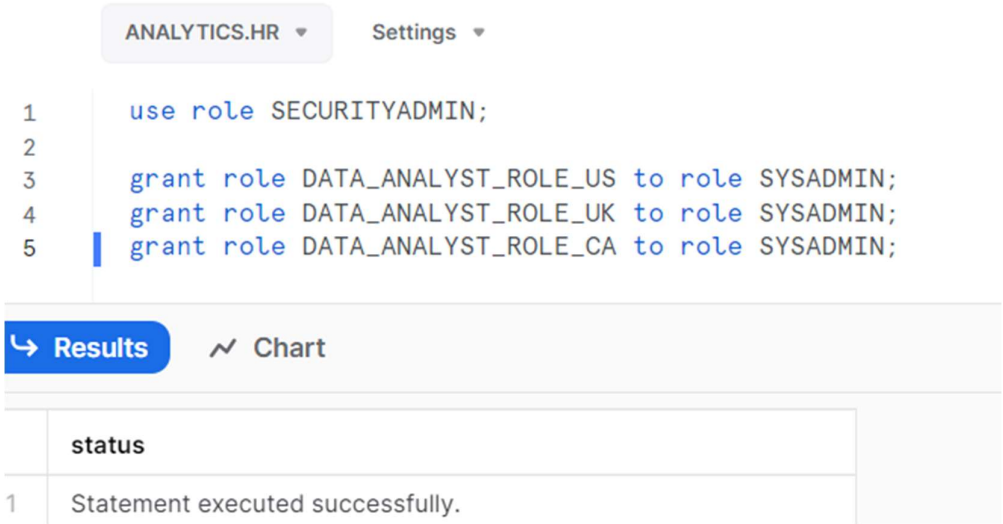
Results Chart

	status
1	Role DATA_ANALYST_ROLE_CA successfully created.

15. When the roles are created, they exist in isolation not allowing the other roles (even the roles which create and grant privileges to them) to access the objects created by them.
16. So, it is required to set up a role hierarchy for the custom roles we created.
17. The below SQL statements assigns the custom roles to the role SYSADMIN so that the SYSADMIN can inherit all the privileges assigned to custom role.

Query: use role SECURITYADMIN;

grant role DATA_ANALYST_ROLE_US to role SYSADMIN;
grant role DATA_ANALYST_ROLE_UK to role SYSADMIN;
grant role DATA_ANALYST_ROLE_CA to role SYSADMIN;



Grant SELECT privilege on table to custom roles

18. The below SQL statements grants usage privileges on database analytics and schema hr present inside it with only select privilege on all tables present inside them to the custom roles created.

Query: use role SYSADMIN;

grant usage on database analytics to role DATA_ANALYST_ROLE_US;
grant usage on schema analytics.hr to role DATA_ANALYST_ROLE_US;
grant select on all tables in schema analytics.hr to role DATA_ANALYST_ROLE_US;

grant usage on database analytics to role DATA_ANALYST_ROLE_UK;
grant usage on schema analytics.hr to role DATA_ANALYST_ROLE_UK;
grant select on all tables in schema analytics.hr to role DATA_ANALYST_ROLE_UK;

grant usage on database analytics to role DATA_ANALYST_ROLE_CA;
grant usage on schema analytics.hr to role DATA_ANALYST_ROLE_CA;
grant select on all tables in schema analytics.hr to role DATA_ANALYST_ROLE_CA;

ANALYTICS.HR Settings

```sql
use role SYSADMIN;

grant usage on database analytics to role DATA_ANALYST_ROLE_US;
grant usage on schema analytics.hr to role DATA_ANALYST_ROLE_US;
grant select on all tables in schema analytics.hr to role DATA_ANALYST_ROLE_US;

grant usage on database analytics to role DATA_ANALYST_ROLE_UK;
grant usage on schema analytics.hr to role DATA_ANALYST_ROLE_UK;
grant select on all tables in schema analytics.hr to role DATA_ANALYST_ROLE_UK;

grant usage on database analytics to role DATA_ANALYST_ROLE_CA;
grant usage on schema analytics.hr to role DATA_ANALYST_ROLE_CA;
grant select on all tables in schema analytics.hr to role DATA_ANALYST_ROLE_CA;
```

Results Chart

	status
1	Statement executed successfully. 2 objects affected.

Grant USAGE privilege on virtual warehouse to custom roles

19. The below SQL statements provides usage privileges on warehouse compute_wh to the custom roles to query tables.

Query: use role ACCOUNTADMIN;

grant usage on warehouse compute_wh to role DATA_ANALYST_ROLE_US;
grant usage on warehouse compute_wh to role DATA_ANALYST_ROLE_UK;
grant usage on warehouse compute_wh to role DATA_ANALYST_ROLE_CA;

ANALYTICS.HR Settings

```sql
use role ACCOUNTADMIN;

grant usage on warehouse compute_wh to role DATA_ANALYST_ROLE_US;
grant usage on warehouse compute_wh to role DATA_ANALYST_ROLE_UK;
grant usage on warehouse compute_wh to role DATA_ANALYST_ROLE_CA;
```

Results Chart

status
Statement executed successfully.

Assign Custom Roles to Users

20. In my case I have only one user so, I am only assigning only DATA_ANALYST_ROLE_US to my use.
21. If you have multiple users you can assign for other roles also.

Query: use role SECURITYADMIN;

grant role DATA_ANALYST_ROLE_US to user GISULSNOWFLAKE;

ANALYTICS.HR Settings

```
use role SECURITYADMIN;

grant role DATA_ANALYST_ROLE_US to user GISULSNOWFLAKE;
```

Results Chart

status
Statement executed successfully.

22. Let us verify the data returned for user when queried on the same table.
23. The below image shows that for user with role DATA_ANALYST_ROLE_US when queried on the table employees, the data returned is only from country US.

Query: Use role DATA_ANALYST_ROLE_US;

Select * from analytics.hr.employees;

ANALYTICS.HR Settings

```
use role DATA_ANALYST_ROLE_US;

select * from analytics.hr.employees;
```

Results Chart

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	COUNTRY
100	Steven	King	SKING@outlook.com	2013-06-17	US
101	Neena	Kochhar	NKOCHHAR@outlook.com	2015-09-21	US
102	Lex	De Haan	LDEHAAN@outlook.com	2011-01-13	US

24. The below image shows that when the user with role SYSADMIN queries on the table employees, all rows are returned.

Query: User role SYSADMIN;

Select * from analytics.hr.employees;

ANALYTICS.HR Settings

```
use role SYSADMIN;

select * from analytics.hr.employees;
```

Results Chart

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	COUNTRY
100	Steven	King	SKING@outlook.com	2013-06-17	US
101	Neena	Kochhar	NKOCHHAR@outlook.com	2015-09-21	US
102	Lex	De Haan	LDEHAAN@outlook.com	2011-01-13	US
103	Alexander	Hunold	AHUNOLD@outlook.com	2016-01-03	UK
104	Bruce	Ernst	BERNST@outlook.com	2017-05-21	UK
105	David	Austin	DAUSTIN@outlook.com	2015-06-25	UK
106	Valli	Pataballa	VPATABAL@outlook.com	2016-02-05	CA
107	Diana	Lorentz	DLORENTZ@outlook.com	2017-02-07	CA
108	Nancy	Greenberg	NGREENBE@outlook.com	2012-08-17	CA

25. The below SQL statement revokes all privileges on table role_mapping to the custom roles.

Query: use role SYSADMIN;

revoke all privileges on table analytics.hr.role_mapping from role DATA_ANALYST_ROLE_US;

ANALYTICS.HR Settings

```
use role SYSADMIN;

revoke all privileges on table analytics.hr.role_mapping from role DATA_ANALYST_ROLE_US;
```

Results Chart

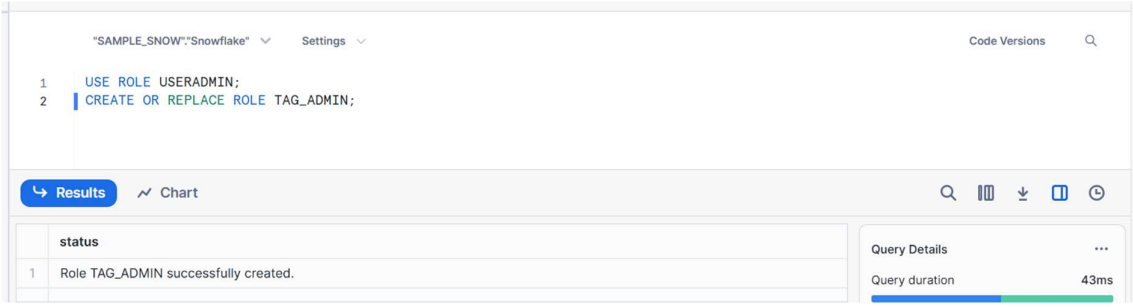
	status
1	Statement executed successfully.

Tagging

1. Create a Custom Role
2. The below SQL statement creates a custom role TAG_ADMIN in Snowflake.

Query: USE ROLE USERADMIN;

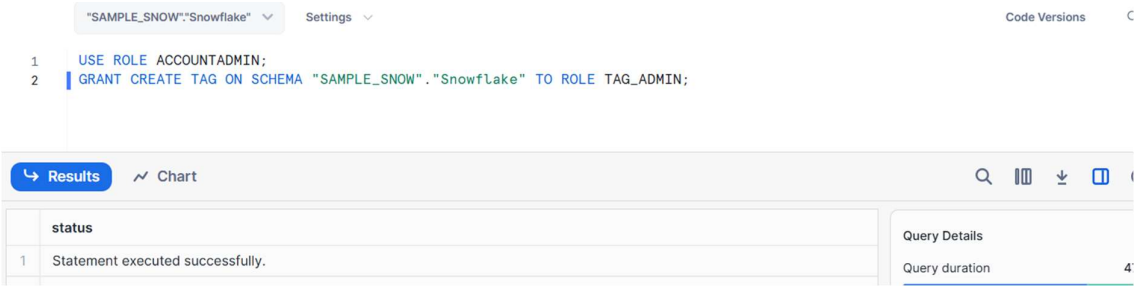
CREATE OR REPLACE ROLE TAG_ADMIN;



3. Assign Tagging Privileges to custom role
4. The below SQL statement grants privileges to create tags to the role TAG_ADMIN.

Query: USE ROLE ACCOUNTADMIN;

GRANT CREATE TAG ON SCHEMA "SAMPLE_SNOW"."Snowflake" TO ROLE TAG_ADMIN;



7. The below SQL statement grants privileges to apply tags on Snowflake objects to the role TAG_ADMIN.

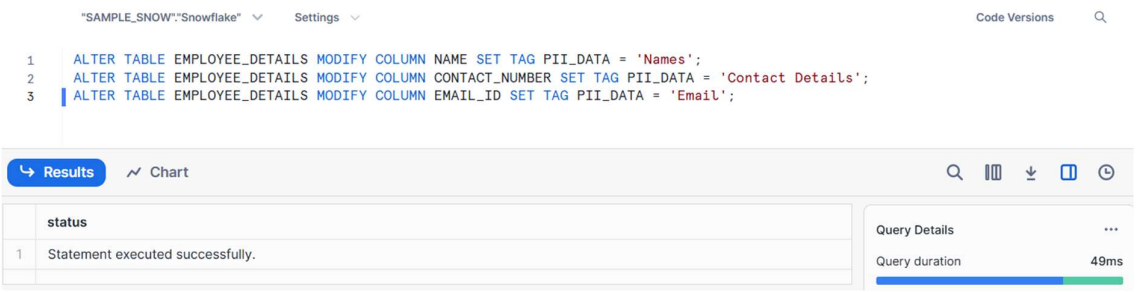
Query: GRANT APPLY TAG ON ACCOUNT TO ROLE TAG_ADMIN;

8. The below example shows creating a tag named PII_DATA with Names, Contact Details and Email as allowed values.

Query: CREATE OR REPLACE TAG PII_DATA ALLOWED_VALUES 'Names', 'Contact Details','Email';

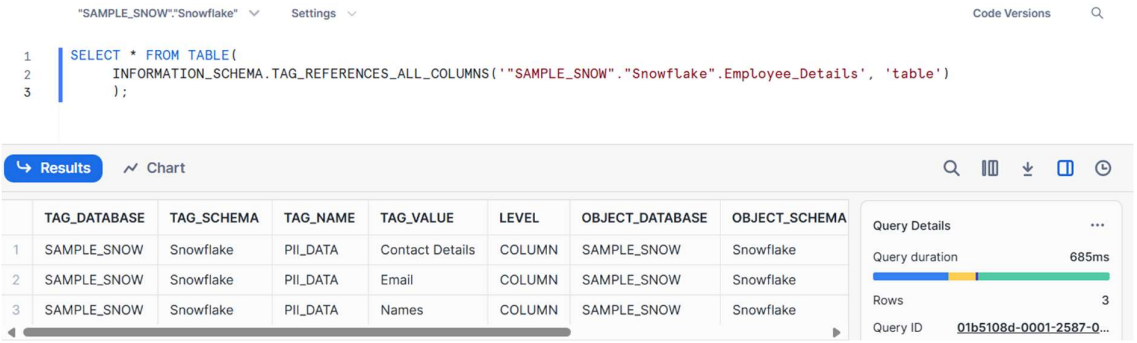
9. To set a tag on an existing column, use the ALTER TABLE … MODIFY COLUMN command for a table column
10. The below example assigns the tag PII_DATA on multiple columns with different tag values on the table named Employee_Details.

Query: ALTER TABLE EMPLOYEE_DETAILS MODIFY COLUMN NAME SET TAG PII_DATA = 'Names';
ALTER TABLE EMPLOYEE_DETAILS MODIFY COLUMN CONTACT_NUMBER SET TAG PII_DATA = 'Contact Details';
ALTER TABLE EMPLOYEE_DETAILS MODIFY COLUMN EMAIL_ID SET TAG PII_DATA = 'Email';



11. The below example helps in identifying all the tags assigned on the columns of the table EMPLOYEE_DETAILS.

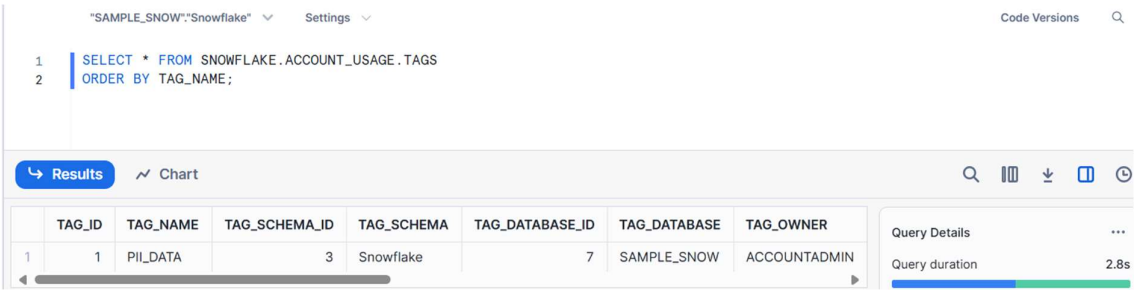
Query: SELECT * FROM TABLE(INFORMATION_SCHEMA.TAG_REFERENCES_ALL_COLUMNS('"SAMPLE_SNOW"."Snowflake". Employee_Details', 'table'));



12. The TAGS view in the Account Usage schema of Snowflake database provides information of all the tags in your Snowflake account including the deleted tags.

Query: SELECT * FROM SNOWFLAKE.ACCOUNT_USAGE.TAGS

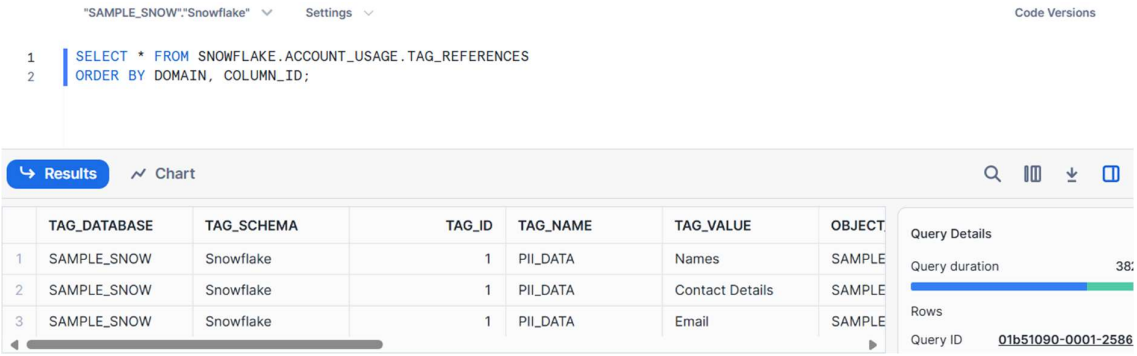
ORDER BY TAG_NAME;



13. The TAG_REFERENCES view in the Account Usage schema of Snowflake database provides information of all the objects in your Snowflake account that are assigned with a tag and a tag value.

Query: SELECT * FROM SNOWFLAKE.ACCOUNT_USAGE.TAG_REFERENCES

ORDER BY DOMAIN, COLUMN_ID;



14. The above output shows all the tags assigned at DATABASE, SCHEMA, TABLE and COLUMN levels at the account level.