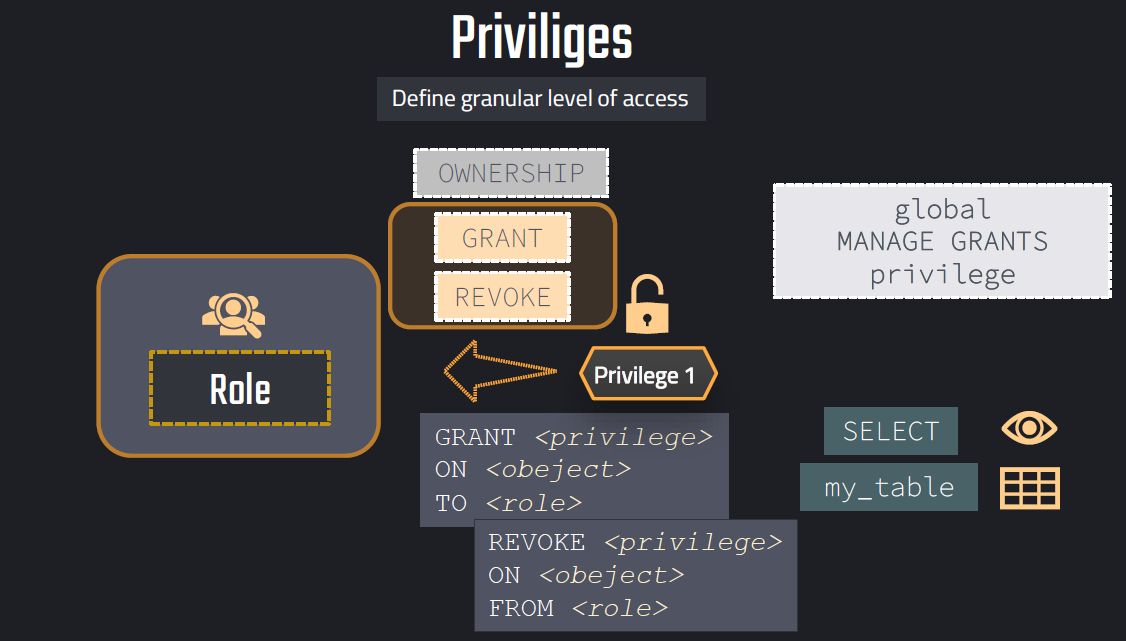
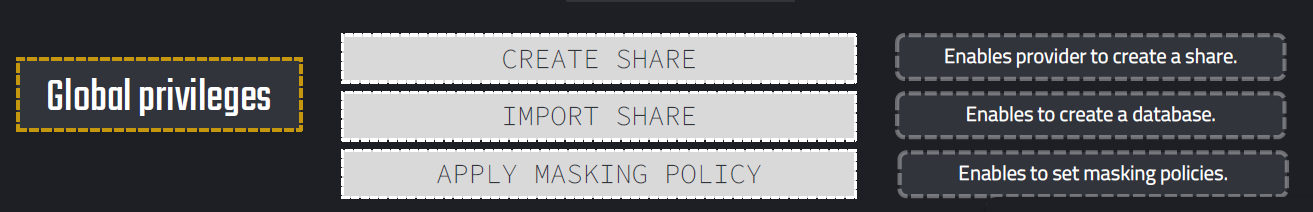
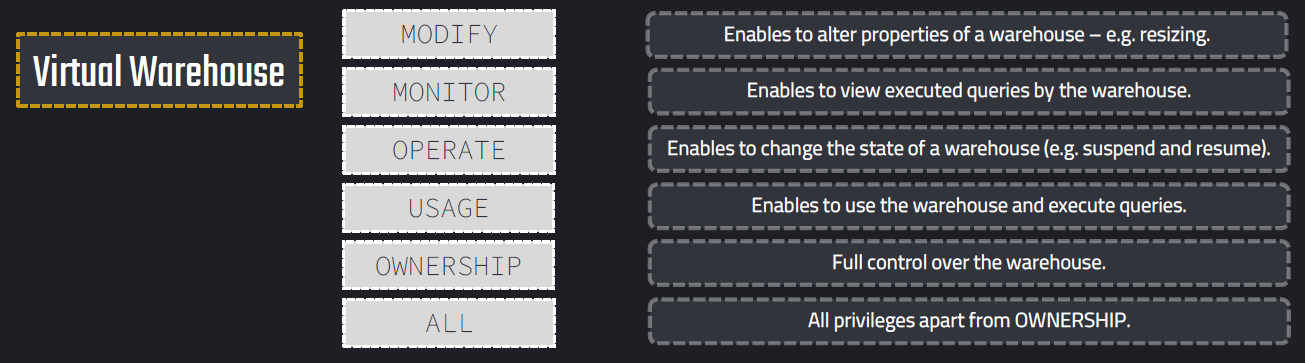


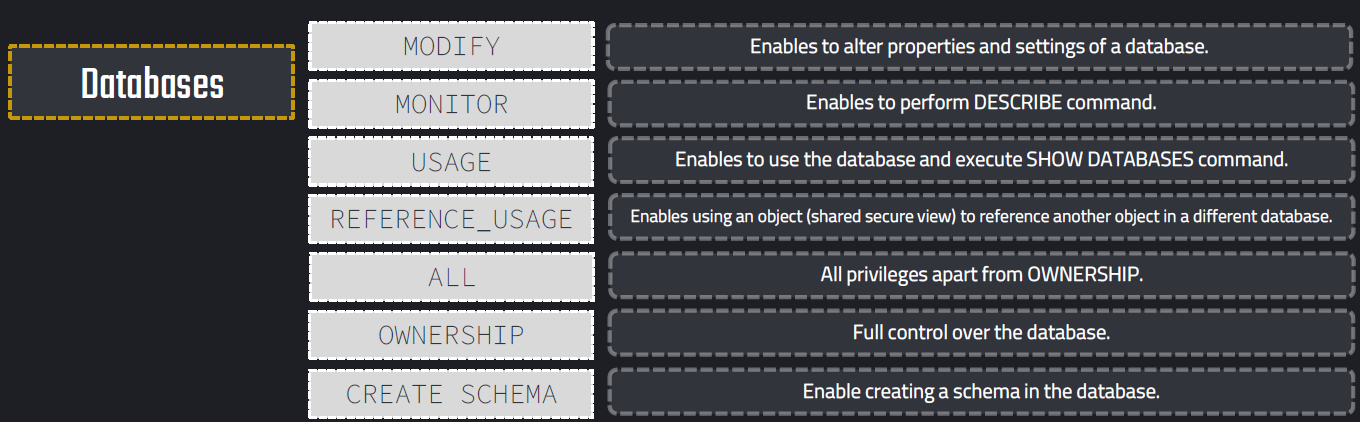
**Quick Cheat Sheet**

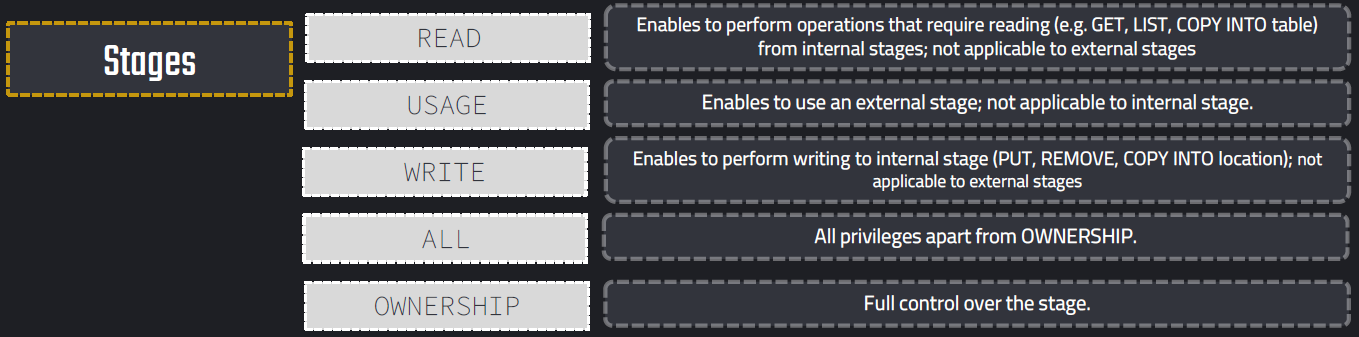
* **USERADMIN**: Creates/Manages USERS
* **SECURITYADMIN**: Grants ROLES and privileges
* **SYSADMIN**: Manages DATA objects
* **ACCOUNTADMIN**: Admin of admins
* **ORGADMIN**: The CEO of the Snowflake empire
* **PUBLIC**: Everyone’s backup role

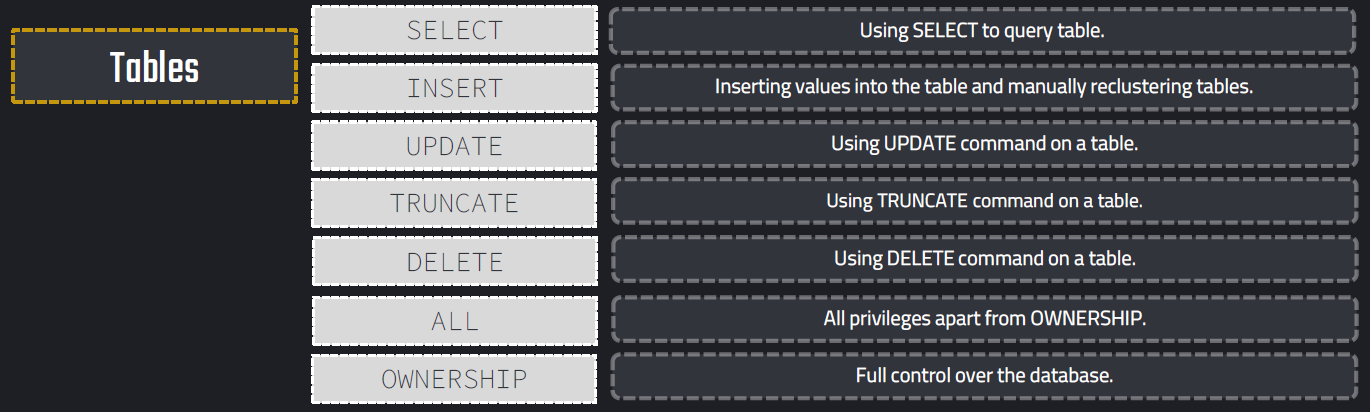












**Example:**

SHOW ROLES;

-- Set up the marketing database

USE ROLE SYSADMIN;

CREATE DATABASE MARKETING;

CREATE SCHEMA SALES;

CREATE TABLE MARKETING.SALES.CAMPAIGN (ID INT, COST NUMERIC,SALES\_AMOUNT NUMERIC);

-- Use USERADMIN or SECURITYADMIN to set up role

USE ROLE SECURITYADMIN;

-- Create role and user

CREATE ROLE MARKETING\_ADMIN;

CREATE USER INITIAL\_USER

PASSWORD = 'AbC201§#';

-- Assign user

GRANT ROLE MARKETING\_ADMIN TO USER INITIAL\_USER;

-- Assign role to SYSADMIN

GRANT ROLE MARKETING\_ADMIN TO ROLE SYSADMIN;

-- Grant privileges

GRANT USAGE ON DATABASE MARKETING TO ROLE MARKETING\_ADMIN;

GRANT USAGE ON SCHEMA MARKETING.SALES TO ROLE MARKETING\_ADMIN;

GRANT SELECT ON TABLE MARKETING.SALES.CAMPAIGN TO ROLE MARKETING\_ADMIN;

GRANT INSERT ON TABLE MARKETING.SALES.CAMPAIGN TO ROLE MARKETING\_ADMIN;

-- Assign warehouse

GRANT USAGE ON WAREHOUSE COMPUTE\_WH TO ROLE MARKETING\_ADMIN;

-- Create tables

GRANT CREATE TABLES ON SCHEMA MARKETING.SALES TO ROLE MARKETING\_ADMIN;

-- SELECT on all tables

GRANT SELECT ON ALL TABLES IN SCHEMA MARKETING.SALES TO ROLE MARKETING\_ADMIN;

-- SELECT on all future tables

GRANT SELECT ON FUTURE TABLES IN SCHEMA MARKETING.SALES TO ROLE MARKETING\_ADMIN;

-- Revoke USAGE on database

REVOKE USAGE ON SCHEMA MARKETING.SALES FROM ROLE MARKETING\_ADMIN;

GRANT OWNERSHIP ON SCHEMA MARKETING.SALES TO ROLE MARKETING\_ADMIN;

-- What are the privileges of the role?

SHOW GRANTS TO ROLE MARKETING\_ADMIN;

-- To whom was the role assigned?

SHOW GRANTS OF ROLE MARKETING\_ADMIN;

-- Drop USER and Database

USE ROLE SYSADMIN;

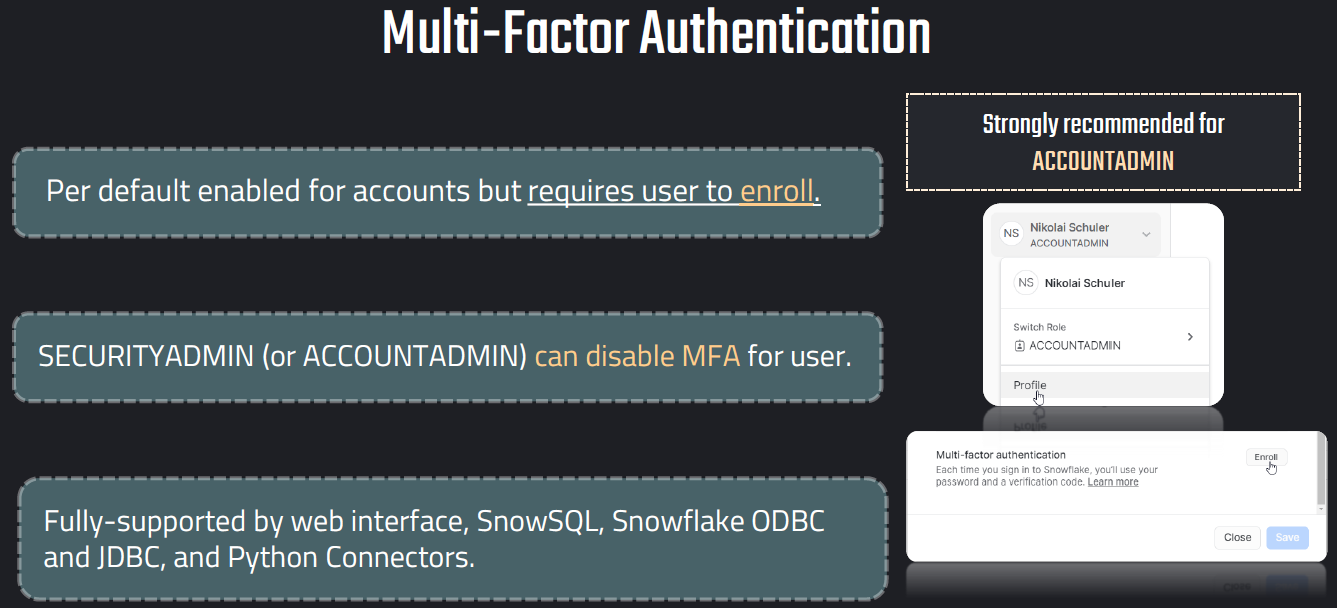
DROP DATABASE MARKETING;

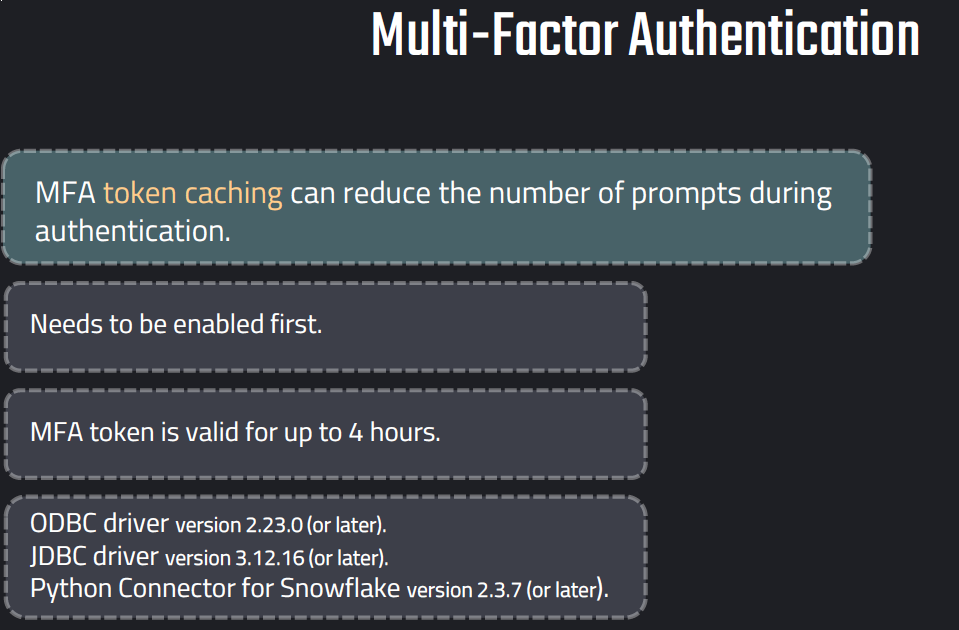
USE ROLE SECURITYADMIN;

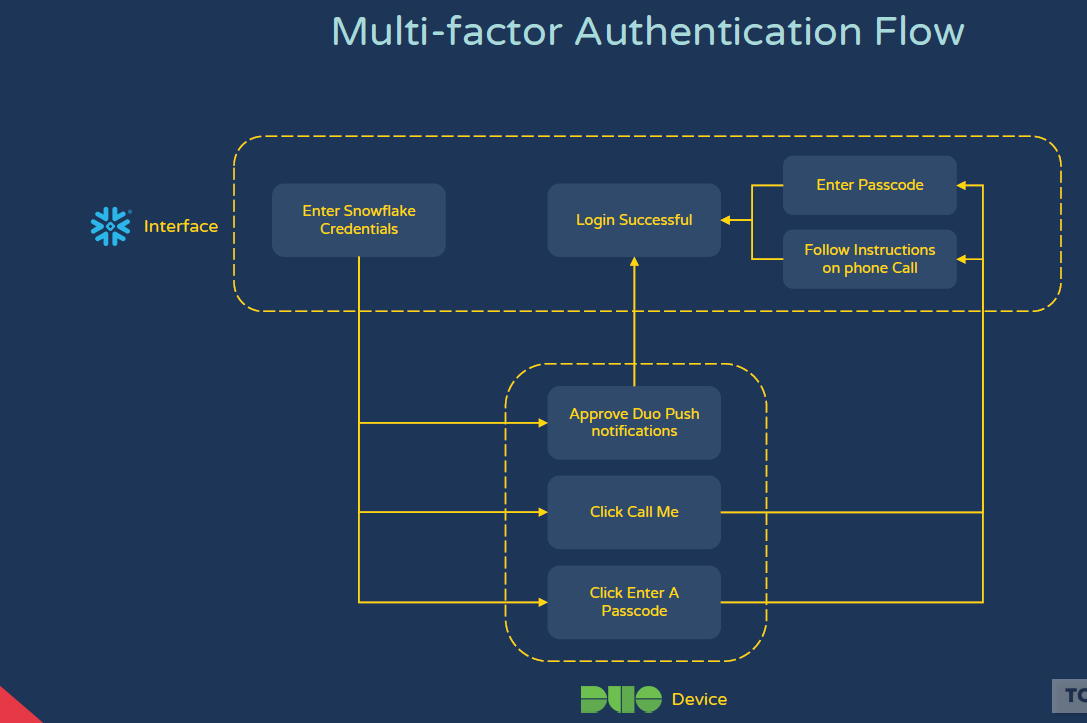
DROP ROLE MARKETING\_ADMIN;

DROP USER INITIAL\_USER;

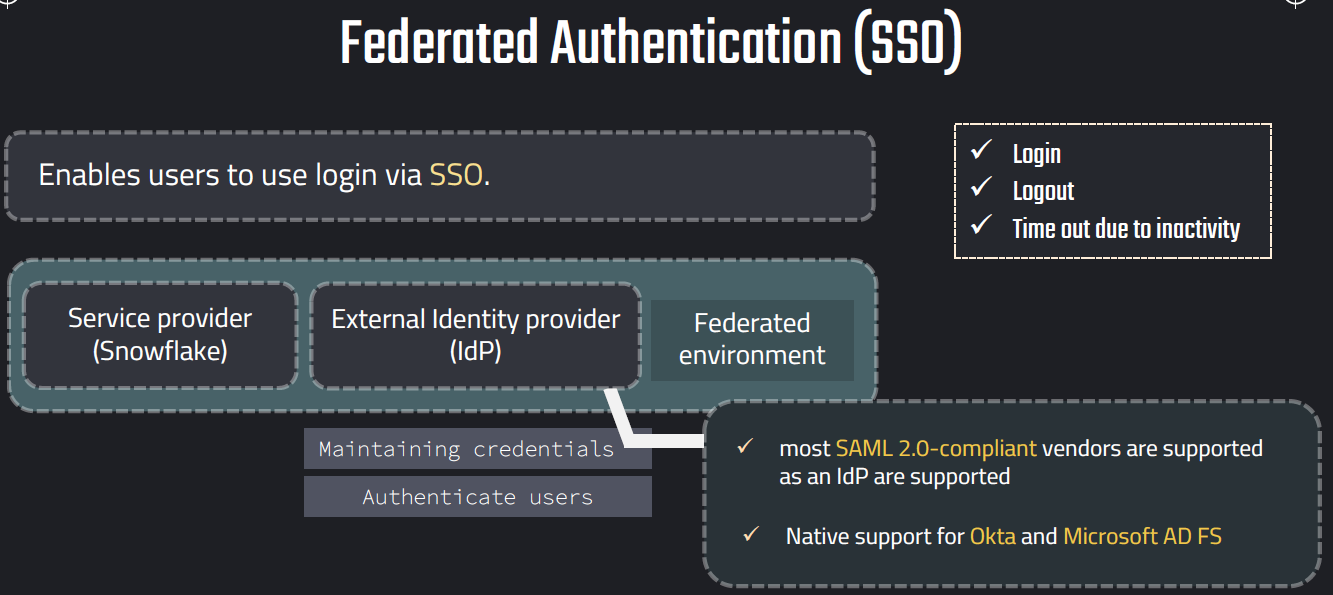


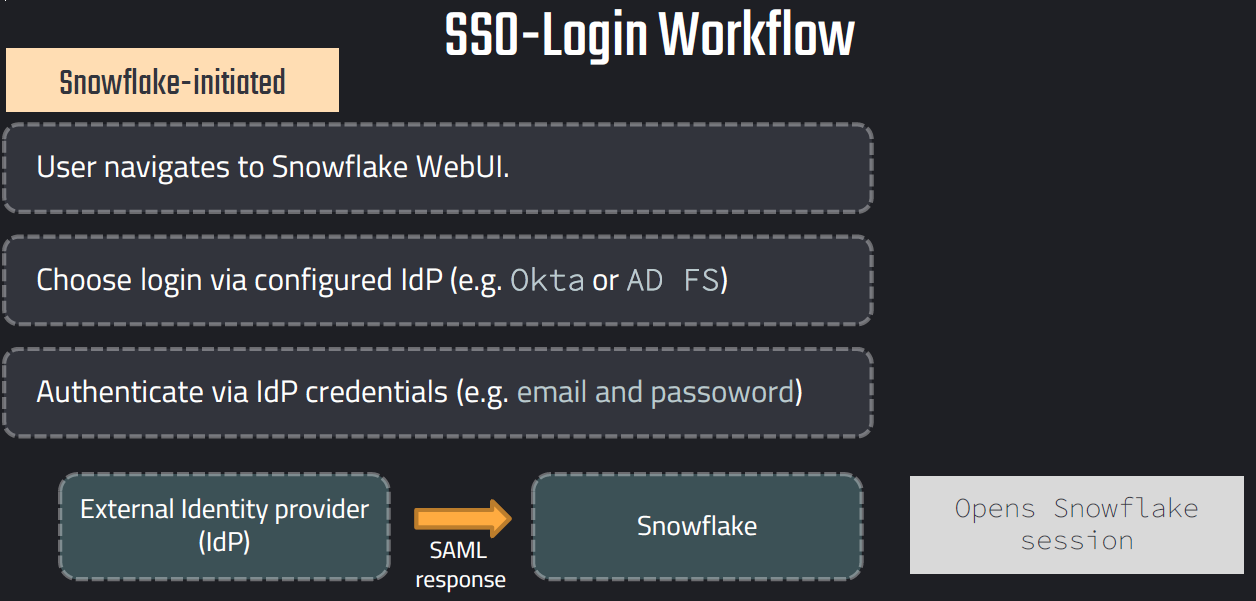


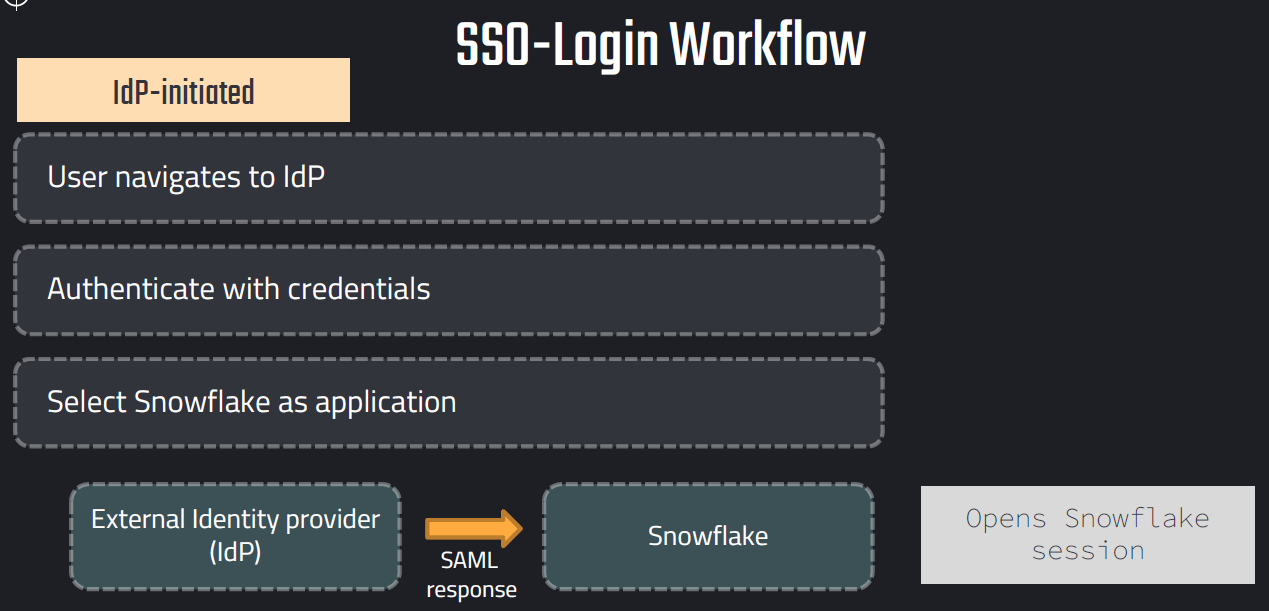


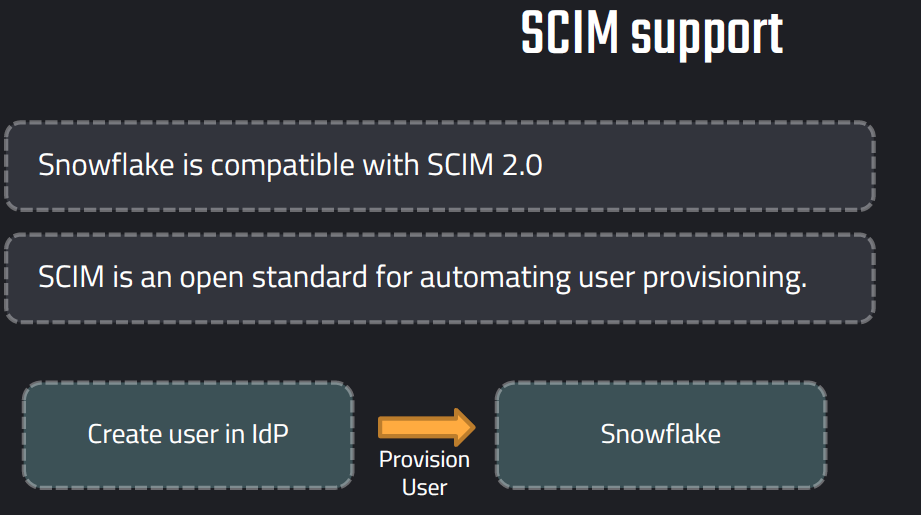


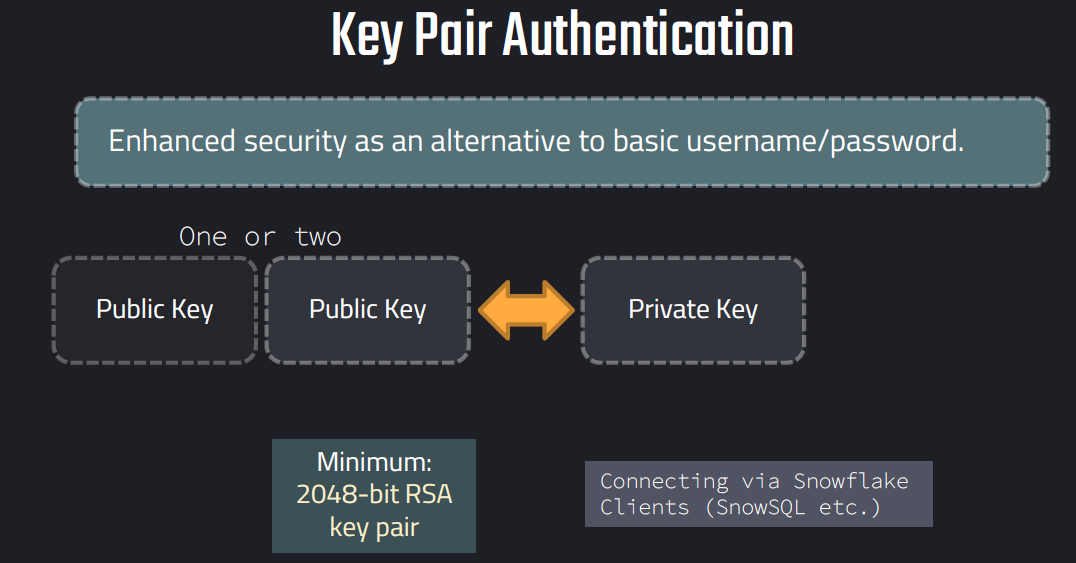


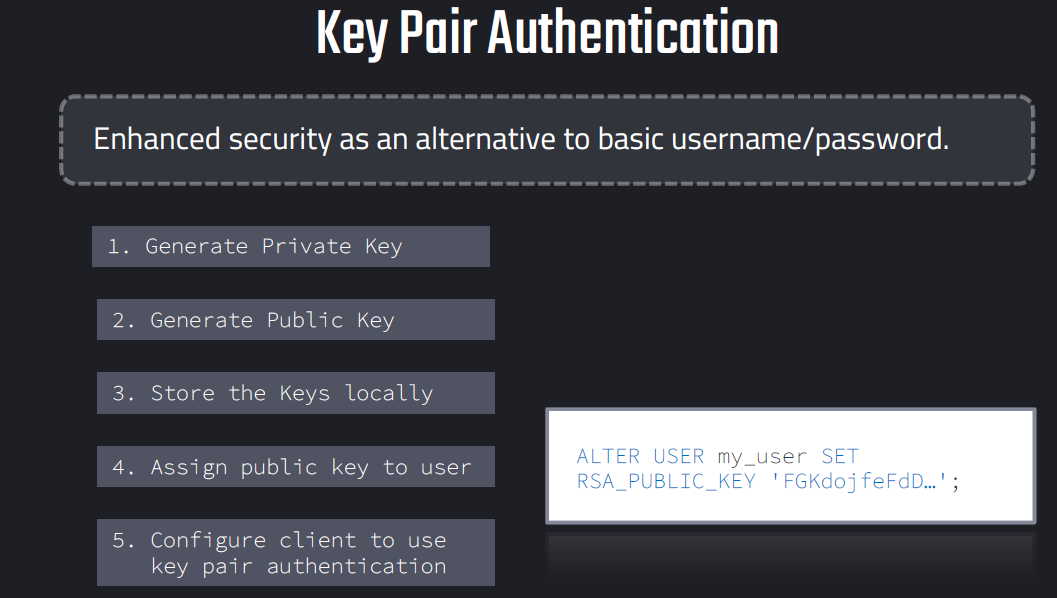




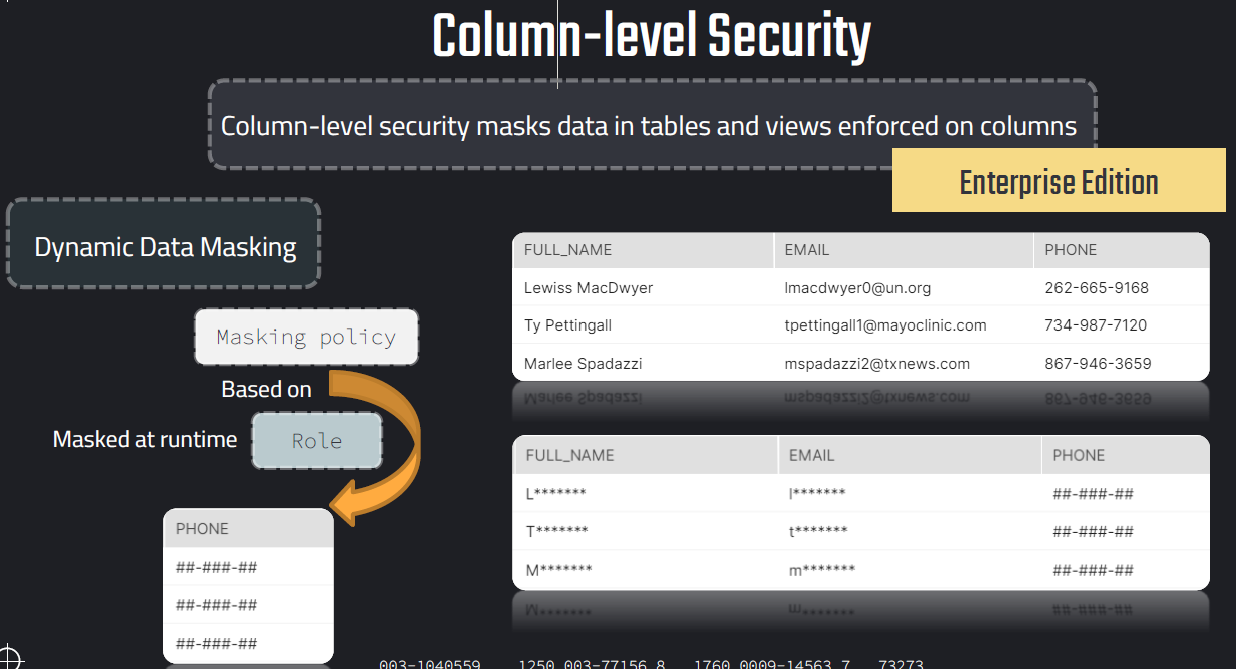


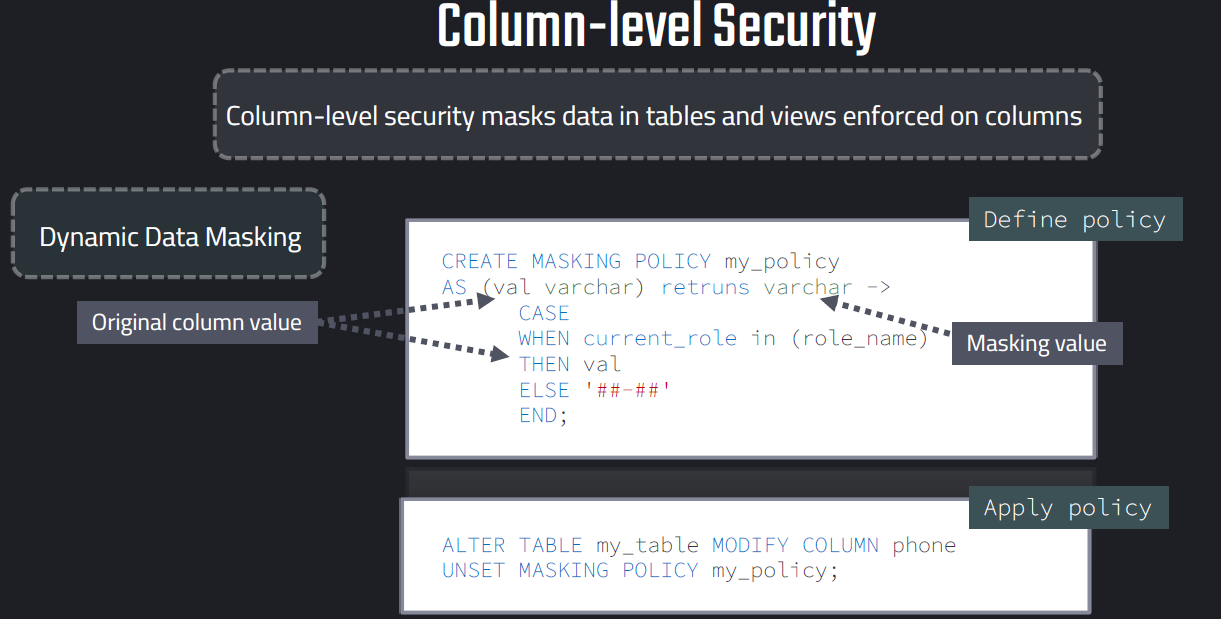


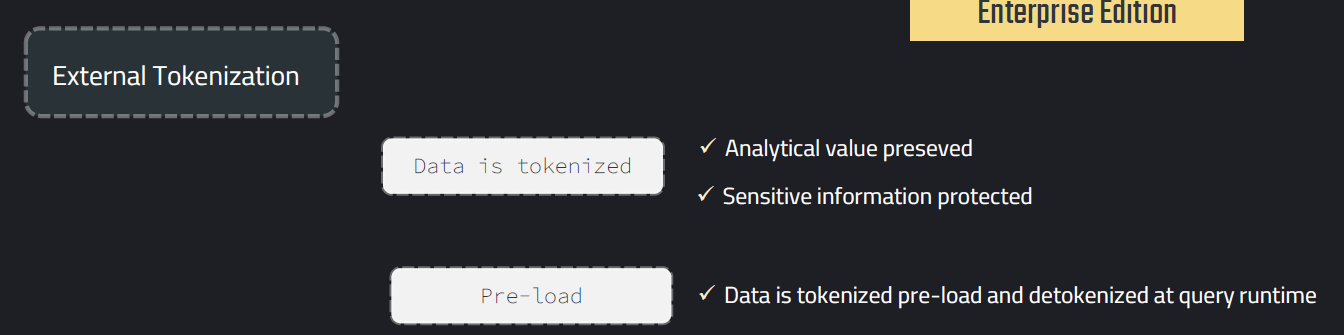






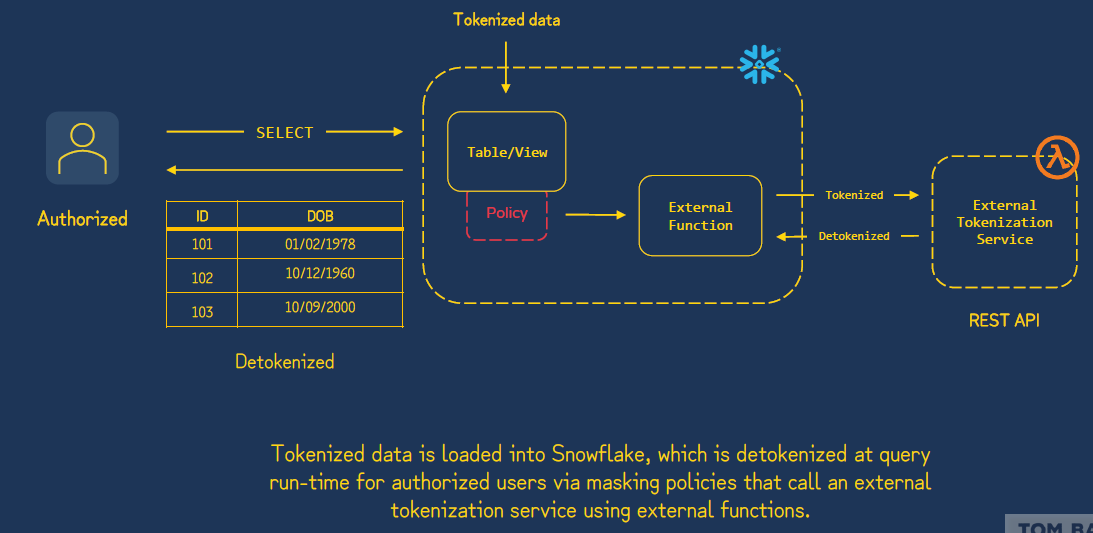






**High Level Flow**

1. Application sends sensitive value (ex: credit card number) to token provider.
2. Provider returns a **token**.
3. Token is stored/processed in Snowflake tables.
4. When needed for authorized use, an **external function** call detokenizes back.



Example:

//// DYNAMIC DATA MASKING ///////

USE ROLE ACCOUNTADMIN;

CREATE OR REPLACE DATABASE DEMO\_DB;

-- Prepare table --

create or replace table customers(

id number,

full\_name varchar,

email varchar,

phone varchar,

spent number,

create\_date DATE DEFAULT CURRENT\_DATE);

-- insert values in table --

insert into customers (id, full\_name, email,phone,spent)

values

(1,'Lewiss MacDwyer','lmacdwyer0@un.org','262-665-9168',140),

(2,'Ty Pettingall','tpettingall1@mayoclinic.com','734-987-7120',254),

(3,'Marlee Spadazzi','mspadazzi2@txnews.com','867-946-3659',120),

(4,'Heywood Tearney','htearney3@patch.com','563-853-8192',1230),

(5,'Odilia Seti','oseti4@globo.com','730-451-8637',143),

(6,'Meggie Washtell','mwashtell5@rediff.com','568-896-6138',600);

-- set up roles

CREATE OR REPLACE ROLE ANALYST\_MASKED;

CREATE OR REPLACE ROLE ANALYST\_FULL;

-- grant usage on database and schema

GRANT USAGE ON DATABASE DEMO\_DB TO ROLE ANALYST\_MASKED;

GRANT USAGE ON DATABASE DEMO\_DB TO ROLE ANALYST\_FULL;

-- grant usage on schema

GRANT USAGE ON SCHEMA DEMO\_DB.PUBLIC TO ROLE ANALYST\_MASKED;

GRANT USAGE ON SCHEMA DEMO\_DB.PUBLIC TO ROLE ANALYST\_FULL;

-- grant select on table

GRANT SELECT ON TABLE DEMO\_DB.PUBLIC.CUSTOMERS TO ROLE ANALYST\_MASKED;

GRANT SELECT ON TABLE DEMO\_DB.PUBLIC.CUSTOMERS TO ROLE ANALYST\_FULL;

-- grant warehouse access to roles

GRANT USAGE ON WAREHOUSE COMPUTE\_WH TO ROLE ANALYST\_MASKED;

GRANT USAGE ON WAREHOUSE COMPUTE\_WH TO ROLE ANALYST\_FULL;

-- assign roles to a user

GRANT ROLE ANALYST\_MASKED TO USER QJ74933;

GRANT ROLE ANALYST\_FULL TO USER QJ74933;

-- Set up masking policy

create or replace masking policy demo\_db.public.phone

as (val varchar) returns varchar ->

case

when current\_role() in ('ANALYST\_FULL', 'ACCOUNTADMIN') then val

else '##-###-##'

end;

-- Apply policy on a specific column

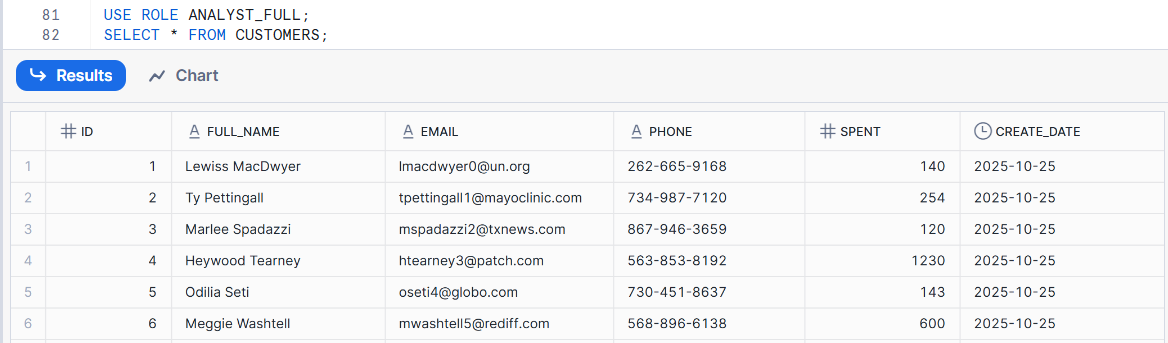
ALTER TABLE IF EXISTS demo\_db.public.CUSTOMERS MODIFY COLUMN phone

SET MASKING POLICY demo\_db.public.PHONE;

-- Validating policies

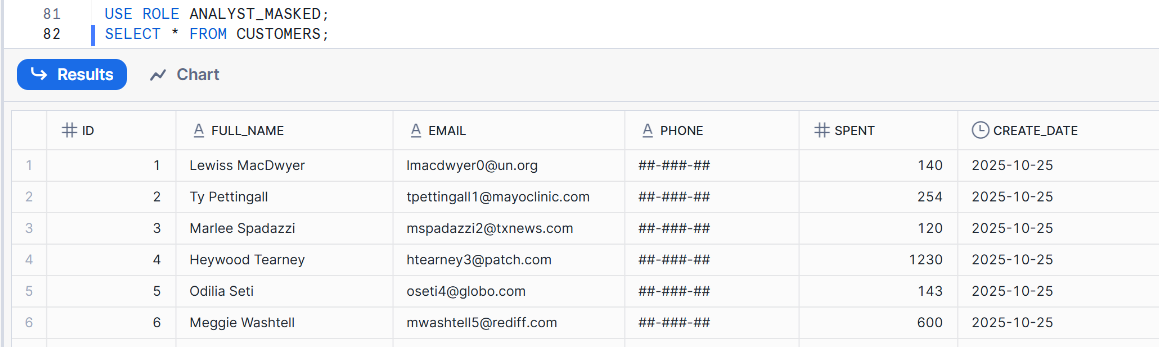
USE ROLE ANALYST\_FULL;

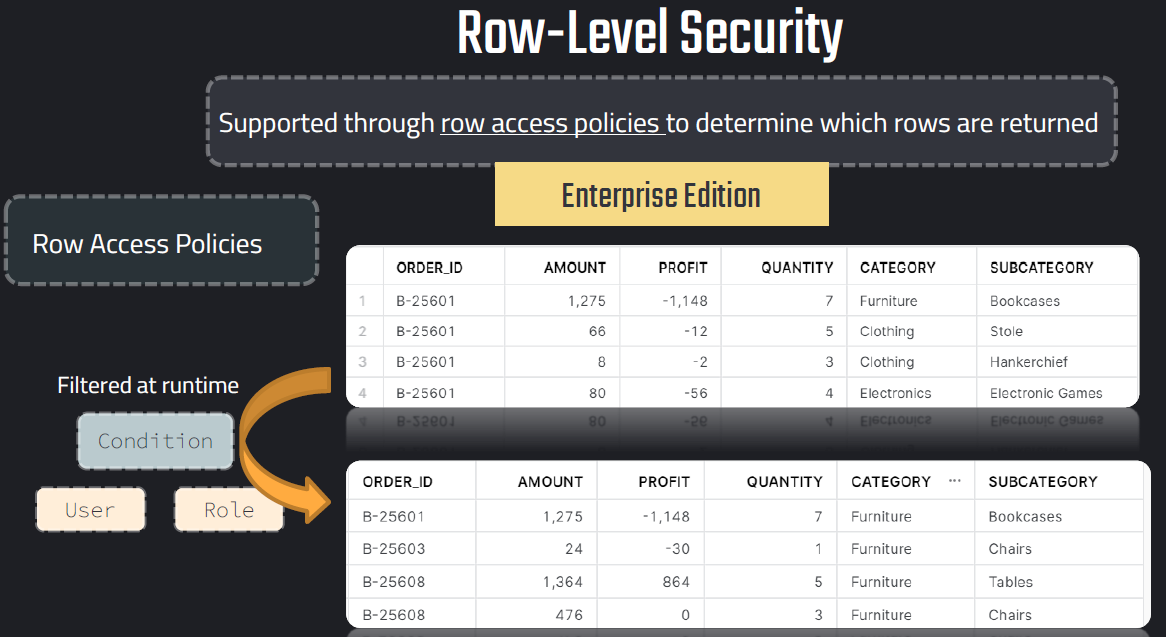
SELECT \* FROM CUSTOMERS;

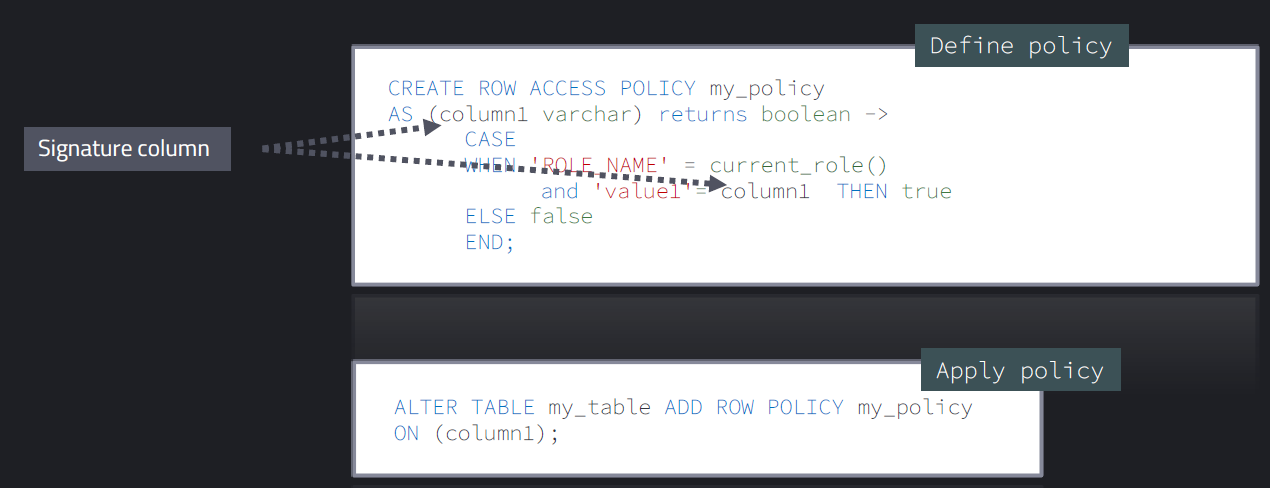


USE ROLE ANALYST\_MASKED;

SELECT \* FROM CUSTOMERS;







////// ROW ACCESS POLICIES ///////

-- Table of interest

SELECT \* FROM DATA\_S.PUBLIC.ORDERS;

-- Thinking of policy based on category

SELECT DISTINCT CATEGORY FROM DATA\_S.PUBLIC.ORDERS;

USE DATA\_S.PUBLIC;

USE ROLE ACCOUNTADMIN;

-- Set up role

create or replace role home\_manager;

grant usage on database data\_s to role home\_manager;

grant usage on schema data\_s.public to role home\_manager; -- wouldn't be necessary since it is public schema

grant select on table data\_s.public.orders to role home\_manager;

grant usage on warehouse compute\_wh to role home\_manager;

grant role home\_manager to user nikolai;

-- Table can be queried and all rows are visible

USE ROLE home\_manager;

SELECT \* FROM ORDERS;

-- Policy that role can see everything

USE ROLE accountadmin;

CREATE OR REPLACE ROW ACCESS POLICY category\_policy

AS (category varchar) RETURNS BOOLEAN ->

CASE WHEN 'HOME\_MANAGER' = current\_role() and 'Furniture'=category then true

else false

end;

-- roles are in ALL CAPS

SELECT current\_role();

DESC ROW ACCESS POLICY category\_policy;

-- Test policy (not working)

USE ROLE home\_manager;

SELECT DISTINCT CATEGORY FROM ORDERS;

-- Add policy to table

USE ROLE ACCOUNTADMIN;

ALTER TABLE data\_s.public.orders ADD ROW ACCESS POLICY category\_policy on (category);

-- Remove policy from table

ALTER TABLE data\_s.public.orders DROP ROW ACCESS POLICY category\_policy;

ALTER TABLE data\_s.public.orders DROP ALL ROW ACCESS POLICIES;

-- Test policy (working)

USE ROLE home\_manager;

SELECT \* FROM ORDERS;

SELECT current\_user();

-- Alter policy

USE ROLE ACCOUNTADMIN;

ALTER ROW ACCESS POLICY category\_policy

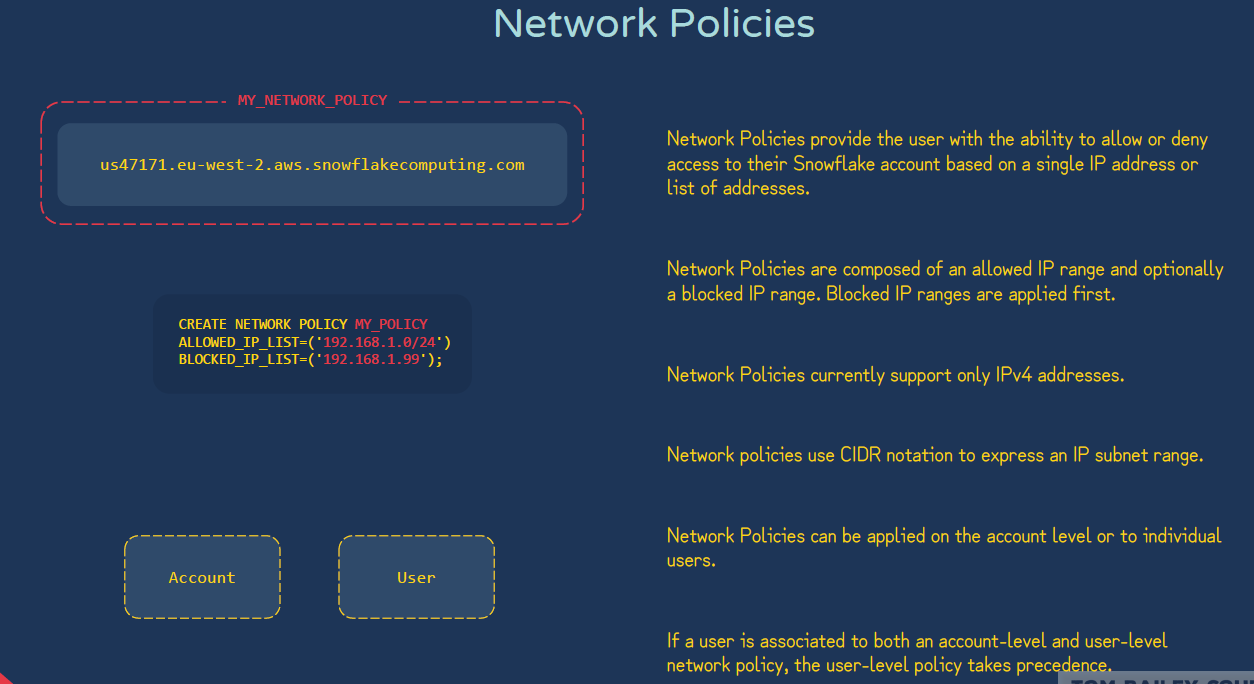
SET BODY ->

CASE WHEN 'NIKOLAI' = current\_user() and 'Furniture'=category then true

when 'ELECTRIC\_MANAGER' = current\_role() and 'Electronics'=category then true

else false

end;





| **CIDR Range Example** | **How Many IP Addresses It Covers** | **Typical Usage** |
| --- | --- | --- |
| 203.0.113.5/32 | 1 IP only | A single laptop or server |
| 203.0.113.0/24 | 256 IPs | Office network |
| 203.0.0.0/16 | 65,536 IPs | Large enterprise |
| 0.0.0.0/0 | The entire internet | Avoid unless you enjoy chaos |

