**Snowflake's dynamic data masking**

To mask sensitive data in Snowflake using dbt, you can leverage Snowflake's **dynamic data masking** feature in combination with dbt's capabilities. Here's how you can approach it:

**1. Create a Masking Policy in Snowflake**

First, create a masking policy in Snowflake that defines how the sensitive data will be masked. For example, to mask a credit card number:

CREATE MASKING POLICY mask\_credit\_card

AS (val STRING)

RETURNS STRING ->

CASE

WHEN CURRENT\_ROLE() IN ('FULL\_ACCESS\_ROLE') THEN val

ELSE 'XXXX-XXXX-XXXX-' || RIGHT(val, 4)

END;

This policy will show the full credit card number to users with the FULL\_ACCESS\_ROLE and mask it for others.

**2. Apply the Masking Policy to a Column**

You apply the masking policy to a column when creating or altering a table. For instance:

CREATE OR REPLACE TABLE customers (

customer\_id STRING,

credit\_card STRING MASKING POLICY mask\_credit\_card

);

**3. Integrate with dbt Models**

In your dbt project, you can apply the masking policy while creating or altering tables. Here's an example of how to do this:

* **Create a dbt model** to apply the masking policy.

Example: models/customers.sql

WITH base AS (

SELECT

customer\_id,

credit\_card

FROM {{ source('raw', 'customers') }}

)

SELECT

customer\_id,

credit\_card

FROM base

When this model runs, it will create or replace the table customers with the credit card column being automatically masked by the policy you created earlier.

**4. Ensure Roles and Access Control**

Ensure that the roles you assign to users have the correct level of access. Users with roles that don't have the FULL\_ACCESS\_ROLE will see the masked values.

This ensures that the masking policy works as intended while documenting the model and testing its integrity.

**Additional Considerations**

* If you need to mask more complex data types, you can modify the masking policies accordingly.
* Dynamic data masking is useful for controlling data access without modifying the underlying data.
* dbt doesn't directly manage Snowflake's masking policies, but it can interact with them by creating or altering tables/views that use them.