-- Q1: Create roles as per the below-mentioned hierarchy. Accountadmin

-- already exists in Snowflake.

create ROLE IDENTIFIER('"ADMIN"') COMMENT = ''

GRANT ROLE IDENTIFIER('"ADMIN"') TO ROLE IDENTIFIER('"ACCOUNTADMIN"')

create ROLE IDENTIFIER('"DEVELOPER"') COMMENT = ''

GRANT ROLE IDENTIFIER('"DEVELOPER"') TO ROLE IDENTIFIER('"ADMIN"')

create ROLE IDENTIFIER('"PII"') COMMENT = ''

GRANT ROLE IDENTIFIER('"PII"') TO ROLE IDENTIFIER('"ACCOUNTADMIN"')

-- Q2: Create an M-sized warehouse using the accountadmin role, name ->

-- assignment\_wh and use it for all the queries

create WAREHOUSE IDENTIFIER('"ASSIGNMENT\_WH"') COMMENT = '' WAREHOUSE\_SIZE = 'Medium' AUTO\_RESUME = true AUTO\_SUSPEND = 300 ENABLE\_QUERY\_ACCELERATION = false WAREHOUSE\_TYPE = 'STANDARD' MIN\_CLUSTER\_COUNT = 1 MAX\_CLUSTER\_COUNT = 1 SCALING\_POLICY = 'STANDARD'

-- Q3: Switch to the admin role

use role admin;

-- Q4: Create a database assignment\_db

CREATE DATABASE ASSIGNMENT\_DB;

-- Q5: Create a schema my\_schema

CREATE SCHEMA my\_schema;

USE SCHEMA MY\_SCHEMA;

-- Q6: Create a table using any sample csv. You can get 1 by googling for

-- sample csv’s. Preferably search for a sample employee dataset so that

-- you have PII related columns else you can consider any column as PII ( 5

-- ).

CREATE TABLE assignment\_db.my\_schema.employee (

employee\_id INT,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

email VARCHAR(100),

phone\_number VARCHAR(20),

hire\_date DATE,

salary DECIMAL(10,2),

inserted\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP(),

elt\_by VARCHAR(50) DEFAULT 'SnowSQL CLI',

file\_name VARCHAR(255)

);

-- Q7: Also, create a variant version of this dataset

CREATE table assignment\_db.my\_schema.employee\_variant (

employee\_id INT,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

email VARCHAR(100),

phone\_number VARCHAR(20),

hire\_date DATE,

salary DECIMAL(10,2),

inserted\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP(),

elt\_by VARCHAR(50) DEFAULT 'SnowSQL CLI',

file\_name VARCHAR(255)

);

-- Q8 and Q9: Load the file into an external and internal stage and copy into both tables

-- For internal staging, need to run commands on snowsql cli.

-- 8 part 1 and 9 part 1:

use ASSIGNMENT\_DB.my\_schema;

create or replace stage mystage;

use role ACCOUNTADMIN;

grant all privileges on stage mystage to role ADMIN;

use role admin;

put file:///Users/vanshtandon/Downloads/emp\_sampledata.csv @mystage;

copy into EMPLOYEE from @mystage/emp\_sampledata.csv file\_format = (type = csv skip\_header = 1);

select \* from EMPLOYEE limit 5;

-- 8 part 2 and 9 part 2: External Staging

--creating csv file format

create or replace file format assingment\_db.my\_schema.my\_csv\_format

type = csv

field\_delimiter = ','

skip\_header = 1

null\_if = ('NULL', 'null')

empty\_field\_as\_null = true;

-- creating a storage integration s3\_int2 using role accountadmin

create or replace storage integration s3\_int2 type = external\_stage storage\_provider= s3

enabled = true storage\_aws\_role\_arn='arn:aws:iam::737865507436:role/vantanrole'

storage\_allowed\_locations =('s3://assingmentbucket');

--using role accountadmin

grant ownership on integration s3\_int2 to role admin;

--creating external stage for s3 using role admin and loading to external stage;

create stage my\_external\_stage STORAGE\_INTEGRATION =s3\_int2

URL='s3://assingmentbucket/emp\_sampledata.csv' file\_format=my\_csv\_format;

-- loading from external stage to employee\_variant;

copy into employee\_variant from @my\_external\_stage;

--Q10: for staging the parquet file user1data.parquet; in terminal

--PUT

file:///Users/vanshtandon/Documents/Snowflake/userdata1.parquet

@mystage;

--creating new file type

create file format myparquetformat TYPE =parquet;

-- Q11: Select query and using inferschema

select \* from table (INFER\_SCHEMA (LOCATION =>'@mystage',FILE

\_FORMAT=>'myparquetformat'));

-- Q12: Add masking policy to the PII columns such that fields like email,

-- phone number, etc. show as \*\*masked\*\* to a user with the developer role.

-- If the role is PII the value of these columns should be visible

CREATE MASKING POLICY PII\_masking

AS (val STRING)

RETURNS STRING ->

CASE

WHEN CURRENT\_ROLE() = 'DEVELOPER' THEN '\*\*MASKED\*\*'

ELSE val

END;

-- Grant USAGE privilege on the masking policy to the developer role

ALTER TABLE employee MODIFY COLUMN EMAIL SET MASKING POLICY PII\_masking;

ALTER TABLE employee MODIFY COLUMN EMPLOYEE\_ID SET MASKING POLICY PII\_masking;

-- Grant SELECT privileges on the masked columns to the 'developer' role

GRANT SELECT(EMAIL) ON employee TO ROLE developer;

GRANT SELECT(EMPLOYEE\_ID) ON employee TO ROLE developer;