```
-- 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;
```