```
-- Creating the database for the assignment
CREATE DATABASE ASSIGNMENT_DB;

-- Creating a virtual warehouse to handle compute resources
CREATE WAREHOUSE assignment_wh;

-- Setting the created warehouse as the active warehouse
USE WAREHOUSE assignment_wh;

-- Creating and setting the schema within the warehouse
USE SCHEMA myschema;
CREATE SCHEMA MYSCHEMA;

-- Creating roles with specific access levels for different user
types
CREATE ROLE ADMIN;
CREATE ROLE DEVELOPER;
CREATE ROLE PII;

-- Granting higher-level roles to manage account-level permissions
GRANT ROLE ACCOUNTADMIN TO ROLE ADMIN;
GRANT ROLE ACCOUNTADMIN TO ROLE PII;
GRANT ROLE ADMIN TO ROLE DEVELOPER;

-- Assigning roles to the user (shashankreddy) for access control
GRANT ROLE ADMIN TO USER shashankreddy;
GRANT ROLE PII TO USER shashankreddy;
GRANT ROLE DEVELOPER TO USER shashankreddy;

-- Creating the table structure for storing employee data
CREATE TABLE employees2_csv (
    EMPLOYEE_ID INT PRIMARY KEY, -- Unique identifier for each
employee
    FIRST_NAME VARCHAR(50), -- Employee's first name
    LAST_NAME VARCHAR(50), -- Employee's last name
    EMAIL VARCHAR(100), -- Employee's email address
    PHONE_NUMBER VARCHAR(20), -- Employee's phone number
    HIRE_DATE DATE, -- Date the employee was hired
    JOB_ID VARCHAR(10), -- Identifier for the employee's job
    SALARY DECIMAL(10, 2), -- Employee's salary
    COMMISSION_PCT DECIMAL(5, 2), -- Commission percentage for the
employee
    MANAGER_ID INT, -- Identifier for the employee's manager
    DEPARTMENT_ID INT, -- Identifier for the employee's department
    ADDRESS VARCHAR(255) -- Employee's address
);

-- Creating an internal stage to store raw employee data files
CREATE OR REPLACE STAGE employees_data_int_stage;

-- Listing files in the internal stage to confirm the presence of
data files
ls @employees_data_int_stage;

-- Creating a file format for CSV files with specific configurations
CREATE OR REPLACE FILE FORMAT infer_csv_format
    TYPE = CSV
    COMPRESSION = GZIP
    FIELD_DELIMITER = ','
    PARSE_HEADER = TRUE
    DATE_FORMAT = 'YYYY-MM-DD'
    FIELD_OPTIONALLY_ENCLOSED_BY = '"';

-- Inferring the schema from a CSV file in the internal stage
SELECT * FROM TABLE(INFER_SCHEMA(
    LOCATION => '@employees_data_int_stage/employees2.csv.gz',
    FILE_FORMAT => 'infer_csv_format'
));

-- Creating the table using the inferred schema from the CSV file
CREATE OR REPLACE TABLE employees2_csv AS
SELECT *
FROM TABLE (
    INFER_SCHEMA(
        LOCATION => '@employees_data_int_stage/employees2.csv.gz',
        FILE_FORMAT => 'infer_csv_format'
    )
);

-- Verifying the generated DDL for the table to ensure it matches
expectations
SELECT GET_DDL('table', 'employees2_csv');

-- Adding additional columns to the table for tracking metadata
ALTER TABLE employees2_csv ADD COLUMN elt_by VARCHAR(100); -- Added
for identifying the data source
ALTER TABLE employees2_csv ADD COLUMN elt_ts TIMESTAMP_LTZ; -- Added
to track timestamp of data loading
ALTER TABLE employees2_csv ADD COLUMN file_name VARCHAR(100); --
Added to track file name
```

```
-- Creating a CSV file format for reading the employee data (for
loading purposes)
CREATE OR REPLACE FILE FORMAT read_csv_format
    TYPE = CSV
    FIELD_DELIMITER = ','
    SKIP_HEADER = 1
    FIELD_OPTIONALLY_ENCLOSED_BY = '"'
    EMPTY_FIELD_AS_NULL = TRUE;

-- Describing the structure of the table to confirm column
definitions
DESCRIBE TABLE employees2_csv;

-- Copying data from the CSV file in the internal stage into the
table
COPY INTO employees2_csv (
    EMPLOYEE_ID,
    FIRST_NAME,
    LAST_NAME,
    EMAIL,
    PHONE_NUMBER,
    HIRE_DATE,
    JOB_ID,
    SALARY,
    COMMISSION_PCT,
    MANAGER_ID,
    DEPARTMENT_ID,
    ADDRESS,
    ELT_BY,
    ELT_TS,
    FILE_NAME
)
FROM (
    SELECT $1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12,
           'my_app_name' AS ELT_BY,
           CURRENT_TIMESTAMP AS ELT_TS,
           METADATA$FILENAME AS FILE_NAME
    FROM '@employees_data_int_stage/employees2.csv.gz'
    (file_format => 'read_csv_format')
)
ON_ERROR = SKIP_FILE;  -- Skipping the file in case of errors

-- Verifying the data in the table to ensure successful data loading
SELECT * FROM employees2_csv;
```

```
-- Creating a variant table for storing employee data as a variant
CREATE OR REPLACE TABLE employees_variant(
    employee_data VARIANT -- Column to store employee data in variant
format
);

-- Inserting data into the variant table from the employee CSV table
INSERT INTO employees_variant (
    SELECT TO_VARIANT(OBJECT_CONSTRUCT(*))
    FROM employees2_csv
);

-- Verifying the variant table data to ensure successful data
insertion
SELECT * FROM employees_variant;

-- Creating storage integration with an external S3 bucket for file
storage
CREATE OR REPLACE STORAGE INTEGRATION s3_integration
    TYPE = EXTERNAL_STAGE
    STORAGE_PROVIDER = 'S3'
    STORAGE_AWS_ROLE_ARN = 'arn:aws:iam::619071335973:role/ak_role'
    ENABLED = TRUE
    STORAGE_ALLOWED_LOCATIONS = ('s3://mykbucket/my_folder/');

-- Describing the integration to confirm configuration details
DESCRIBE INTEGRATION s3_integration;

-- Creating an external stage for the S3 bucket to read data from
CREATE OR REPLACE STAGE employees_data_ext_stage
    URL = 's3://mykbucket/my_folder/'
    STORAGE_INTEGRATION = s3_integration;

-- Listing files in the external stage to confirm the presence of
data files
list @employees_data_ext_stage;

-- Creating a table for storing the external data from S3
CREATE OR REPLACE TABLE employees_external (
    EMPLOYEE_ID NUMBER(3, 0),
    FIRST_NAME VARCHAR(16777216),
    LAST_NAME VARCHAR(16777216),
    EMAIL VARCHAR(16777216),
    PHONE_NUMBER VARCHAR(16777216),
    HIRE_DATE DATE,
    JOB_ID VARCHAR(16777216),
    SALARY NUMBER(5, 0),
    COMMISSION_PCT NUMBER(3, 2),
    MANAGER_ID NUMBER(3, 0),
    DEPARTMENT_ID NUMBER(3, 0),
    ADDRESS VARCHAR(16777216),
    elt_by VARCHAR(100),
    elt_ts TIMESTAMP_LTZ,
    file_name VARCHAR(100)
);

-- Copying data from the external stage (S3) into the created table
COPY INTO employees_external
FROM (
    SELECT $1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12,
           'my_app_name' AS ELT_BY,
           CURRENT_TIMESTAMP AS ELT_TS,
           METADATA$FILENAME AS FILE_NAME
    FROM '@employees_data_int_stage/employees2.csv.gz'
)
FILE_FORMAT = (skip_header = 1, field_optionally_enclosed_by = '"')
ON_ERROR = SKIP_FILE;

-- Verifying the data from the external table to ensure successful
data loading
SELECT * FROM employees_external;

-- Creating a file format for reading Parquet files
CREATE OR REPLACE FILE FORMAT infer_parquet_format
    TYPE = PARQUET
    COMPRESSION = AUTO
    USE_LOGICAL_TYPE = TRUE
    TRIM_SPACE = TRUE
    REPLACE_INVALID_CHARACTERS = TRUE
    NULL_IF = ('\N', 'NULL', 'NUL', '');

-- Inferring schema for the Parquet file in the external stage
SELECT * FROM TABLE(INFER_SCHEMA(
    LOCATION => '@employees_data_ext_stage/titanic.parquet',
    FILE_FORMAT => 'infer_parquet_format',
    MAX_RECORDS_PER_FILE => 10
));

-- Reading the data from the Parquet file in the external stage
```

```
SELECT *,
    'my_app_name' AS ELT_BY,
    CURRENT_TIMESTAMP AS ELT_TS,
    METADATA$FILENAME AS FILE_NAME
FROM '@employees_data_ext_stage/titanic.parquet'
(FILE_FORMAT => infer_parquet_format);

-- Creating a masking policy for Personally Identifiable Information
(PII)
CREATE OR REPLACE MASKING POLICY pii_mask
    AS (val STRING)
    RETURNS STRING ->
        CASE
            WHEN current_role() IN ('DEVELOPER') THEN '**masked**'
-- Masked for developers
            ELSE val  -- Unmasked for other roles
        END;

-- Applying the masking policy to sensitive columns in the employees
table
ALTER TABLE IF EXISTS employees_csv MODIFY COLUMN email SET MASKING
POLICY pii_mask;
ALTER TABLE IF EXISTS employees_csv MODIFY COLUMN address SET MASKING
POLICY pii_mask;
ALTER TABLE IF EXISTS employees_csv MODIFY COLUMN phone_number SET
MASKING POLICY pii_mask;

-- Switching to the DEVELOPER role to test the masking policy
USE ROLE DEVELOPER;

-- Verifying the data for a developer (should see masked values)
SELECT * FROM employees2_csv;
```