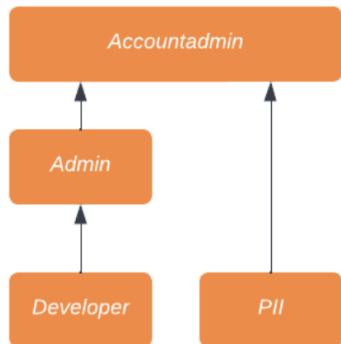


Name: **Md. Owais Ashraf**  
 Employee ID: **TAS205**

---

**PS 1: Create roles as per the below-mentioned hierarchy.**



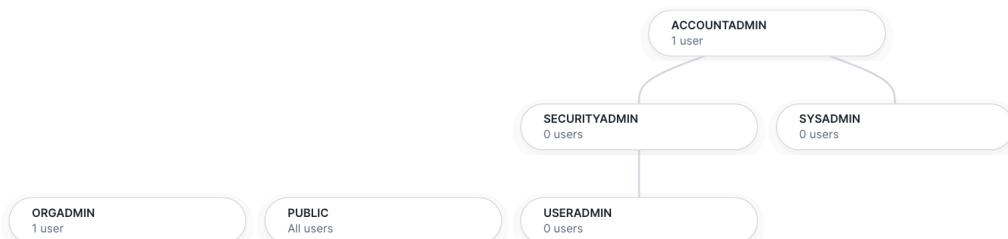
Also given that Accountadmin already exists in the hierarchy.

Solution :-

1. Entering SnowSQL CLI

```
[owais@Mds-MacBook-Air ~ % snowsql -a uo09544.ap-southeast-1 -u OWAIS
[Password:
 * SnowSQL * v1.2.32
 Type SQL statements or !help
 OWAIS#COMPUTE_WH@(no database).(no schema)>
```

Looking into pre-existing architecture :-



**ACCOUNTADMIN:** This is the top-tier role that has complete control over all aspects of the Snowflake account, including user and role management, billing, and configuration settings.

**SECURITYADMIN:** This role specializes in the security aspects, such as managing access permissions and security settings. It's a focused role that does not cover broader administrative tasks.

**SYSADMIN:** The SysAdmin role is tasked with the administration of system-level objects like databases and warehouses. It does not encompass user or security administration.

**USERADMIN:** Dedicated to user management, this role can create and manage users and their roles but does not have access to the broader account or system administration.

## 2. Adding roles as per problem statement

```
OWAIS#COMPUTE_WH@(no database).(no schema)>CREATE ROLE Admin;
GRANT ROLE Admin TO ROLE Accountadmin;

+-----+
| status
+-----+
| Role ADMIN successfully created.
+-----+
1 Row(s) produced. Time Elapsed: 0.194s

+-----+
| status
+-----+
| Statement executed successfully.
+-----+
1 Row(s) produced. Time Elapsed: 0.154s

OWAIS#COMPUTE_WH@(no database).(no schema)>CREATE ROLE Developer;
GRANT ROLE Developer TO ROLE Admin;

+-----+
| status
+-----+
| Role DEVELOPER successfully created.
+-----+
1 Row(s) produced. Time Elapsed: 0.274s

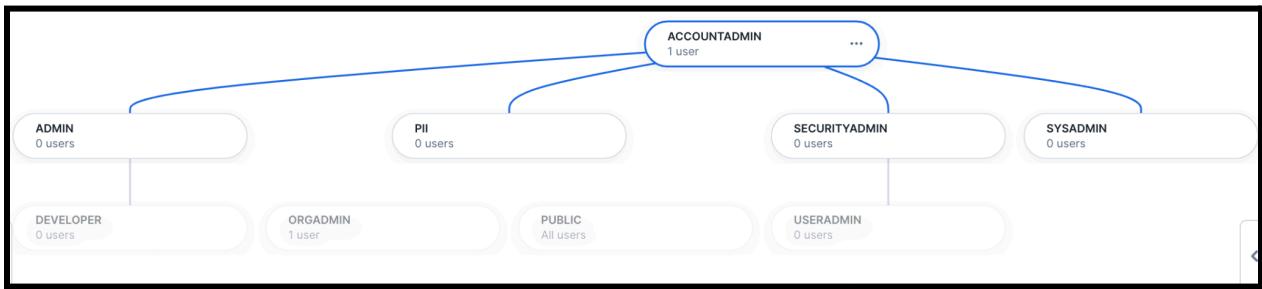
+-----+
| status
+-----+
| Statement executed successfully.
+-----+
1 Row(s) produced. Time Elapsed: 0.146s

OWAIS#COMPUTE_WH@(no database).(no schema)>CREATE ROLE PII;
GRANT ROLE PII TO ROLE Accountadmin;

+-----+
| status
+-----+
| Role PII successfully created.
+-----+
1 Row(s) produced. Time Elapsed: 0.169s

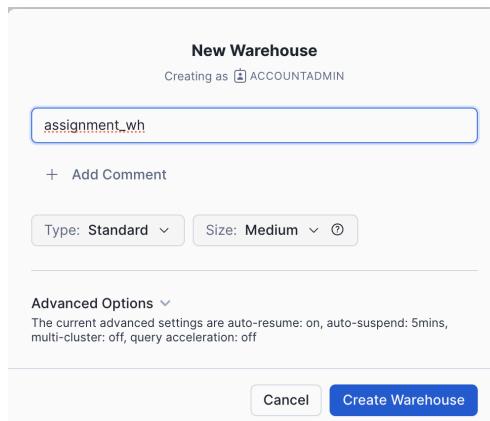
+-----+
| status
+-----+
| Statement executed successfully.
+-----+
1 Row(s) produced. Time Elapsed: 0.139s
```

Updated hierarchy:-



**2. Create an M-sized warehouse using the accountadmin role, name -> assignment\_wh and use it for all the queries.**

**Using UI :-**



**Using Snowflake Worksheet / SnowSQL CLI :-**

```
1 USE ROLE ACCOUNTADMIN;
2 CREATE WAREHOUSE IF NOT EXISTS ASSIGNMENT_WH
3 WITH WAREHOUSE_SIZE = 'MEDIUM';
4
```

status

1	Warehouse ASSIGNMENT_WH successfully created.
---	---

Query Details

- Query duration 63ms
- Rows 1
- Query ID 01b40346-0000-a140-...

### 3. Switch to the admin role

The screenshot shows a database interface with the following details:

- Session Information:** ADMIN • COMPUTE\_WH
- Code Area:** No Database selected, Settings dropdown.
- Query:** USE ROLE ADMIN;
- Results:** status  
1 Statement executed successfully.
- Query Details:** PREVIEW, Query duration: 55ms, Rows: 1, Query ID: 01b4034e-0000-a110-...

### 4. Create a database assignment\_db

Before creation of the database we will have to grant permission to ADMIN role of creating database and warehouse.

The screenshot shows a database interface with the following details:

- Session Information:** ACCOUNTADMIN • ASSIGNMENT\_WH
- Code Area:** ASSIGNMENT\_DB.PUBLIC, Settings dropdown.
- Query:** GRANT USAGE ON WAREHOUSE assignment\_wh TO ROLE admin;  
GRANT CREATE DATABASE ON ACCOUNT TO ROLE admin;
- Results:** status  
1 Statement executed successfully.
- Query Details:** PREVIEW, Query duration: 24ms

The screenshot shows a database interface with the following details:

- Session Information:** ADMIN • ASSIGNMENT\_WH
- Code Area:** ASSIGNMENT\_DB.PUBLIC, Settings dropdown.
- Query:** USE ROLE ADMIN;  
Create or replace DATABASE assignment\_db;
- Results:** status  
1 Database ASSIGNMENT\_DB successfully created.
- Query Details:** PREVIEW, Query duration: 79ms

## 5. Create a schema my\_schema

The screenshot shows a database interface with the following details:

- Top navigation bar: ADMIN, ASSIGNMENT\_WH, Share, and a dropdown menu.
- Schema selection: ASSIGNMENT\_DB.MY\_SCHEMA.
- Code editor area:

```
15
16  CREATE SCHEMA MY_SCHEMA;
17  USE SCHEMA MY_SCHEMA;
18
19
20
```
- Results tab selected, showing a single row:

status
1 Statement executed successfully.
- Query Details panel on the right:
  - Query duration: 21ms

**6. 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.**

The screenshot shows a database interface with the following details:

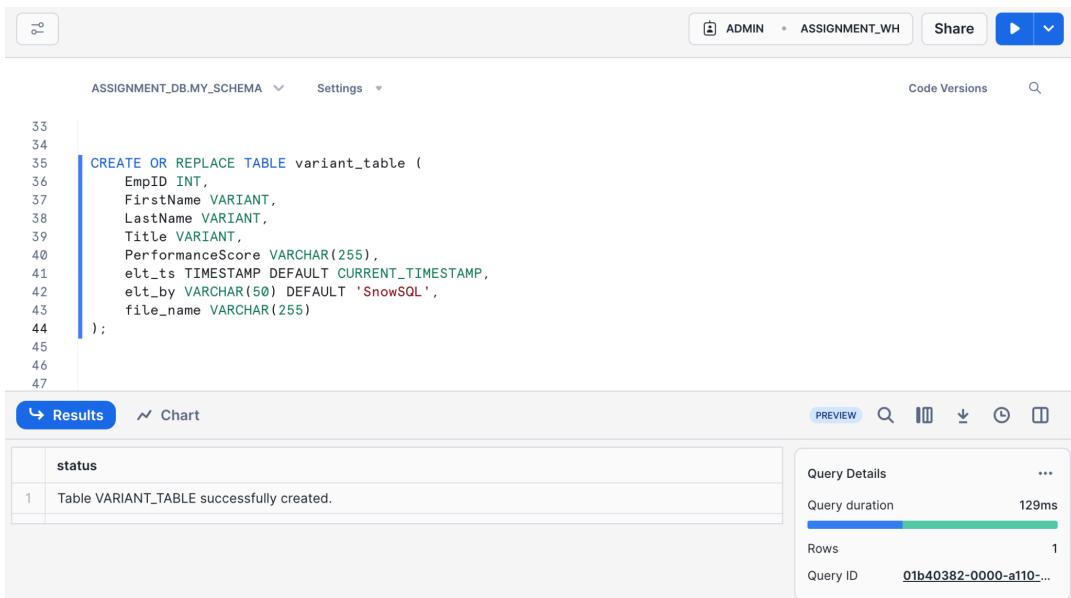
- Top navigation bar: ADMIN, ASSIGNMENT\_WH, Share, and a dropdown menu.
- Schema selection: ASSIGNMENT\_DB.MY\_SCHEMA.
- Code editor area:

```
18
19  CREATE or replace TABLE Employees (
20      EmpID INT PRIMARY KEY,
21      FirstName VARCHAR(255),
22      LastName VARCHAR(255),
23      StartDate DATE,
24      ExitDate DATE,
25      Title VARCHAR(255),
26      Supervisor VARCHAR(255),
27      ADEmail VARCHAR(255),
28      DOB DATE,
29      PerformanceScore VARCHAR(255),
30      elt_ts TIMESTAMP DEFAULT CURRENT_DATE,
31      elt_by VARCHAR(50) DEFAULT 'SnowSQL',
32      file_name VARCHAR(255)
33 );
34
35
```
- Results tab selected, showing a single row:

status
1 Table EMPLOYEES successfully created.
- Query Details panel on the right:
  - Query duration: 142ms

Dataset Link: [Click](#)

## 7. Create a Variant Table of the dataset.



The screenshot shows the Snowflake SQL editor interface. The top navigation bar includes 'ADMIN', 'ASSIGNMENT\_WH', 'Share', and a play/pause button. The schema dropdown is set to 'ASSIGNMENT\_DB.MY\_SCHEMA'. The code area contains the following SQL:

```
33
34
35 CREATE OR REPLACE TABLE variant_table (
36     EmpID INT,
37     FirstName VARIANT,
38     LastName VARIANT,
39     Title VARIANT,
40     PerformanceScore VARCHAR(255),
41     elt_ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
42     elt_by VARCHAR(50) DEFAULT 'SnowSQL',
43     file_name VARCHAR(255)
44 );
45
46
47
```

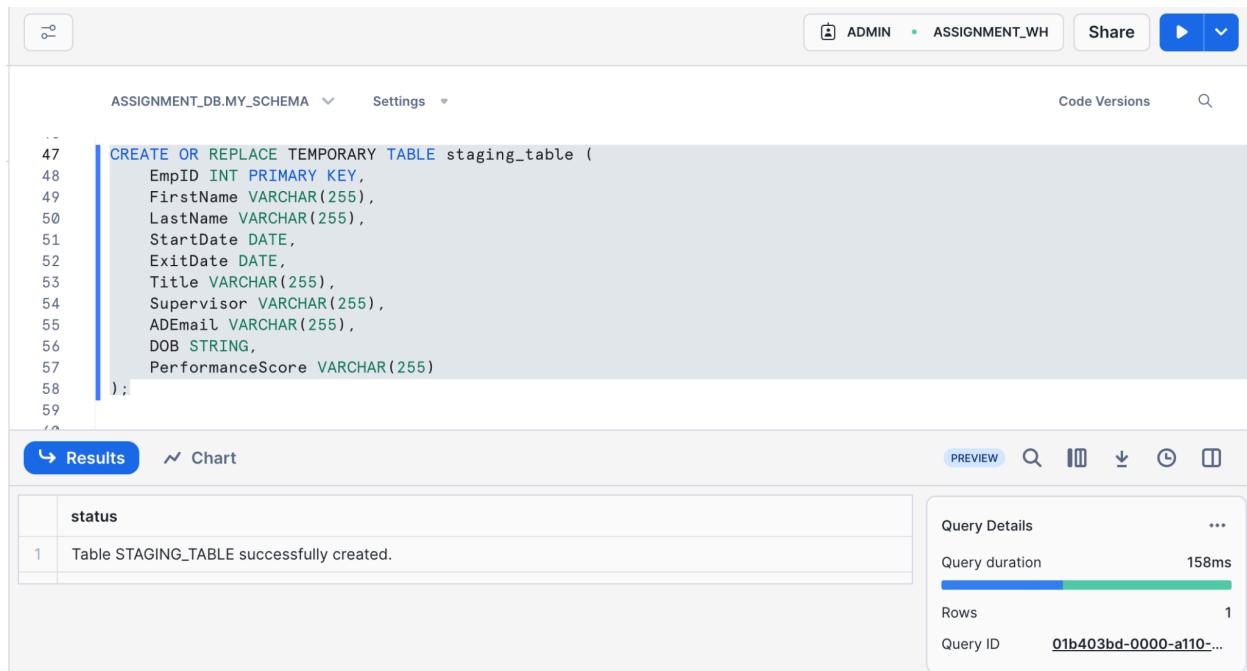
The results tab shows a single row in the 'status' table:

	status
1	Table VARIANT_TABLE successfully created.

Query Details panel on the right:

- Query duration: 129ms
- Rows: 1
- Query ID: 01b40382-0000-a110-...

In order to load data into the variant table, we need a staging table with the same schema as the original table.



The screenshot shows the Snowflake SQL editor interface. The top navigation bar includes 'ADMIN', 'ASSIGNMENT\_WH', 'Share', and a play/pause button. The schema dropdown is set to 'ASSIGNMENT\_DB.MY\_SCHEMA'. The code area contains the following SQL:

```
47
48
49 CREATE OR REPLACE TEMPORARY TABLE staging_table (
50     EmpID INT PRIMARY KEY,
51     FirstName VARCHAR(255),
52     LastName VARCHAR(255),
53     StartDate DATE,
54     ExitDate DATE,
55     Title VARCHAR(255),
56     Supervisor VARCHAR(255),
57     ADEmail VARCHAR(255),
58     DOB STRING,
59     PerformanceScore VARCHAR(255)
60 );
```

The results tab shows a single row in the 'status' table:

	status
1	Table STAGING_TABLE successfully created.

Query Details panel on the right:

- Query duration: 158ms
- Rows: 1
- Query ID: 01b403bd-0000-a110-...

## Creating staging area and file format

The screenshot shows the Snowflake SQL interface. The top navigation bar includes 'ADMIN', 'ASSIGNMENT\_WH', 'Share', and a play button. Below the navigation is a dropdown for 'ASSIGNMENT\_DB.MY\_SCHEMA' and a 'Settings' menu. On the right are 'Code Versions' and a search icon.

```

64 CREATE OR REPLACE FILE FORMAT my_csv_format
65 type = 'csv',
66 field_optionally_enclosed_by = """",
67 field_delimiter=',',
68 skip_header=1,
69 empty_field_as_null = TRUE,
70 null_if = ('N/A'),
71 TRIM_SPACE = true
72 error_on_column_count_mismatch=False;
73 |
74 CREATE OR REPLACE STAGE staging
75 FILE_FORMAT = my_csv_format;
76
  
```

The results tab shows a single row of output:

status	
1	Stage area STAGING successfully created.

On the right, the 'Query Details' panel displays:

- Query duration: 140ms
- Rows: 1
- Query ID: 01b403c1-0000-a140...

Using PUT command now to fetch data from local storage to snowflake schema (internal staging)

```

OWS#ASSIGNMENT_WH@ASSIGNMENT_DB.MY_SCHEMA>PUT file:///Users/owais/Downloads/employee_dataset.csv @staging;
+-----+-----+-----+-----+-----+-----+-----+-----+
| source | target           | source_size | target_size | source_compression | target_compression | status    | message   |
+-----+-----+-----+-----+-----+-----+-----+-----+
| employee_dataset.csv | employee_dataset.csv.gz | 779998 | 193024 | NONE | GZIP | UPLOADED |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 Row(s) produced. Time Elapsed: 1.701s
  
```

Now copying data from staging area to our temporary table

The screenshot shows the Snowflake SQL interface with the same top navigation and schema dropdown as before.

```

77
78 COPY INTO STAGING_TABLE
79 FROM @staging
80 FILE_FORMAT = my_csv_format
81 ON_ERROR = 'CONTINUE';
82
83 | SELECT * FROM STAGING_TABLE LIMIT 10;
84
85
  
```

The results tab shows the copied data in a table:

	EMPID	FIRSTNAME	LASTNAME	STARTDATE	EXITDATE	TITLE	SUPERVISOR	ADEMAIL	DOB
1	3433	Latia	Costa	2022-04-06	2023-07-03	Area Sales Manager	Jacob Braun	latia.costa@bilearner.com	01-07-1
2	3434	Sharlene	Terry	2020-11-06	2023-01-29	Area Sales Manager	Tracy Marquez	sharlene.terry@bilearner.com	07-03-1
3	3436	Joseph	Martins	2022-01-21	2023-06-29	Area Sales Manager	George Jenkins	joseph.martins@bilearner.com	11-11-1
4	3438	Dheepa	Nguyen	2018-08-10	2019-11-04	Area Sales Manager	Brian Miller	dheepa.nguyen@bilearner.com	06-04-1
5	3439	Bartholemew	Khemmich	2022-05-25	2022-11-27	Area Sales Manager	Charles Parks	bartholemew.khemmich@bilearner.com	24-11-1
6	3440	Xana	Potts	2019-12-05	2023-02-17	Area Sales Manager	Gregory Walker	xana.potts@bilearner.com	06-11-1
7	3441	Prater	Jeremy	2019-04-28	null	Area Sales Manager	Tyler Lewis	prater.jeremy@bilearner.com	21-11-1
8	3442	Kaylah	Moon	2019-07-09	2022-06-16	Area Sales Manager	Ashley Scott	kaylah.moon@bilearner.com	24-11-1
9	3443	Kristen	Tate	2021-04-05	2023-05-12	Area Sales Manager	Lauren Jones	kristen.tate@bilearner.com	08-04-1
10	3444	Bobby	Rodgers	2021-11-28	2022-02-04	Area Sales Manager	Matthew Jackson	bobby.rodgers@bilearner.com	15-11-1

On the right, the 'Query Details' panel displays:

- Query duration: 102ms
- Rows: 10
- Query ID: 01b403f5-0000-a110-0...

Below the table, there are two charts:

- An 'EMPID' distribution chart showing values 3433 and 3444.
- A 'FIRSTNAME' distribution chart showing 100% filled.

## Now, we have to bring data into our variant table

```

ASSIGNMENT_DB.MY_SCHEMA ▾ Settings ▾ Code Versions ▾

88
89     INSERT INTO variant_table (
90         EmpID,
91         FullName,
92         Duration,
93         Title,
94         Supervisor,
95         Contact,
96         DOB,
97         PerformanceScore,
98         elt_by,
99         file_name
100    )
101
102    SELECT
103        EmpID,
104        OBJECT_CONSTRUCT(
105            'First Name', FirstName,
106            'Last Name', LastName
107        ) AS FullName,
108        OBJECT_CONSTRUCT(
109            'Start Date', StartDate,
110            'Exit Date', ExitDate
111        ) AS Duration,
112        Title,
113        Supervisor,
114        OBJECT_CONSTRUCT(
115            'AD Email', ADEmail
116        ) AS Contact,
117        DOB,
118        PerformanceScore,
119        'SnowSQL', -- Assuming the elt_by is hardcoded for this example
120        'employees_dataset.csv' -- Assuming a single source file for simplicity
121
122    FROM
123        STAGING_TABLE;

```

The columns we want to be paired together as one json file under one parent column, we will use OBJECT\_CONSTRUCT method and alias it on the variant table column name.

		number of rows inserted
1		6024

Select \* from variant\_table;

	EMPID	FULLNAME	DURATION	TITLE	SUPERV	CONTACT	DOB	PERFO
1	3433	{ "First Name": "Latia", "Last Name": "Costa"}	{ "Exit Date": "2023-07-03", "Start Date": "2022-04-06"}	Area S	Jacob Br	{ "AD Email": "latia.costa@b	01-07-1942	Exceeds
2	3434	{ "First Name": "Sharlene", "Last Name": "Terry"}	{ "Exit Date": "2023-01-29", "Start Date": "2020-11-06"}	Area S	Tracy Mc	{ "AD Email": "sharlene.terry@	07-03-1957	Fully Met
3	3436	{ "First Name": "Joseph", "Last Name": "Martins"}	{ "Exit Date": "2023-06-29", "Start Date": "2022-01-21"}	Area S	George C	{ "AD Email": "joseph.martin@	11-11-1949	Fully Met
4	3438	{ "First Name": "Dheepa", "Last Name": "Nguyen"}	{ "Exit Date": "2019-11-04", "Start Date": "2018-08-10"}	Area S	Brian Mil	{ "AD Email": "dheepa.nguye	06-04-1948	Fully Met
5	3439	{ "First Name": "Bartholemew", "Last Name": "Khen"}	{ "Exit Date": "2022-11-27", "Start Date": "2022-05-25"}	Area S	Charles I	{ "AD Email": "bartholemew.	24-11-1981	Fully Met
6	3440	{ "First Name": "Xana", "Last Name": "Potts"}	{ "Exit Date": "2023-02-17", "Start Date": "2019-12-05"}	Area S	Gregory	{ "AD Email": "xana.potts@	06-11-1951	Fully Met
7	3441	{ "First Name": "Prater", "Last Name": "Jeremy"}	{ "Start Date": "2019-04-28"}	Area S	Tyler Lev	{ "AD Email": "prater.jeremy@	21-11-1989	Exceeds
8	3442	{ "First Name": "Kaylah", "Last Name": "Moon"}	{ "Exit Date": "2022-06-16", "Start Date": "2019-07-09"}	Area S	Ashley S	{ "AD Email": "kaylah.moon@	24-11-1952	Exceeds
9	3443	{ "First Name": "Kristen", "Last Name": "Tate"}	{ "Exit Date": "2023-05-12", "Start Date": "2021-04-05"}	Area S	Lauren J	{ "AD Email": "kristen.tate@	08-04-1994	Fully Met
10	3444	{ "First Name": "Bobby", "Last Name": "Rodgers"}	{ "Exit Date": "2022-02-04", "Start Date": "2021-11-28"}	Area S	Matthew	{ "AD Email": "bobby.rodger@	15-11-1983	Fully Met
11	3446	{ "First Name": "Hector", "Last Name": "Dalton"}	{ "Start Date": "2021-08-24"}	Area S	Sydney I	{ "AD Email": "hector.dalton@	01-05-1996	Exceeds
12	3447	{ "First Name": "Mariela", "Last Name": "Schultz"}	{ "Exit Date": "2023-06-18", "Start Date": "2020-05-26"}	Area S	Michelle	{ "AD Email": "mariela.schult	17-02-1964	Fully Met
13	3448	{ "First Name": "Angela", "Last Name": "Molina"}	{ "Exit Date": "2020-11-06", "Start Date": "2019-10-11"}	Area S	Patricia	{ "AD Email": "angela.molina@	12-05-1958	Fully Met
14	3449	{ "First Name": "Gerald", "Last Name": "Preston"}	{ "Exit Date": "2023-05-27", "Start Date": "2023-05-10"}	Area S	Ashley R	{ "AD Email": "gerald.presto	18-09-1992	Fully Met
15	3450	{ "First Name": "Reilly", "Last Name": "Moyer"}	{ "Exit Date": "2022-12-04", "Start Date": "2020-09-01"}	Area S	Stanley I	{ "AD Email": "reilly.moyer@	11-08-1994	Exceeds
16	3451	{ "First Name": "Carlee", "Last Name": "French"}	{ "Exit Date": "2022-11-11", "Start Date": "2021-02-18"}	Area S	Michael	{ "AD Email": "carlee.frenc	15-01-1968	Exceeds
17	3455	{ "First Name": "Charity", "Last Name": "Miranda"}	{ "Exit Date": "2022-07-05", "Start Date": "2021-06-29"}	Area S	Daniel Ro	{ "AD Email": "charity.mirand	18-01-1999	Exceeds
18	3458	{ "First Name": "Cory", "Last Name": "Robinson"}	{ "Exit Date": "2023-05-24", "Start Date": "2022-04-28"}	Area S	Elizabeth	{ "AD Email": "cory.robinson@	12-08-1996	Needs
19	3459	{ "First Name": "Sanya", "Last Name": "Yu"}	{ "Exit Date": "2022-06-21", "Start Date": "2021-04-18"}	Area S	Erin Baile	{ "AD Email": "sanya.yu@bil	09-02-1944	Exceeds
20	3460	{ "First Name": "Alisa", "Last Name": "James"}	{ "Start Date": "2020-02-19"}	Area S	Dennis H	{ "AD Email": "alisa.james@t	10-02-1944	Fully Met
21	3461	{ "First Name": "Lincoln", "Last Name": "Compton"}	{ "Exit Date": "2021-10-01", "Start Date": "2019-07-18"}	Area S	Tammy C	{ "AD Email": "lincoln.compt	29-12-1997	Needs
22	3462	{ "First Name": "Ailiana", "Last Name": "Nolan"}	{ "Exit Date": "2021-09-17", "Start Date": "2018-09-13"}	Area S	Brianna I	{ "AD Email": "ailiana.nolan@	09-08-1942	Needs
23	3463	{ "First Name": "Kayden", "Last Name": "Dodson"}	{ "Exit Date": "2021-07-18", "Start Date": "2020-04-30"}	Area S	Jessica I	{ "AD Email": "kayden.dodso	21-06-1951	Fully Met

## 8. Load the file into an external and internal stage

### 1. Internal Staging: Stores data files internally within Snowflake

- Firstly, we will be creating a file format(although we have already created it before too), to configure our data, like its type, delimiter and the first line is header.

```
[OWAIS#ASSIGNMENT_WH@ASSIGNMENT_DB.MY_SCHEMA>CREATE OR REPLACE FILE FORMAT my_format
  TYPE = 'CSV'
  FIELD_DELIMITER = ','
  SKIP_HEADER= 1;
+-----+
| status
+-----+
| File format MY_FORMAT successfully created.
+-----+
1 Row(s) produced. Time Elapsed: 0.168s
```

- Creating Staging area, configured with our created file format, to load the data

```
[OWAIS#ASSIGNMENT_WH@ASSIGNMENT_DB.MY_SCHEMA>CREATE OR REPLACE STAGE staging_area FILE_FORMAT=my_format;
+-----+
| status
+-----+
| Stage area STAGING_AREA successfully created.
+-----+
1 Row(s) produced. Time Elapsed: 0.159s
```

- PUT Command used to locate the data file in our system and put/load it on the stage we created.

```
[OWAIS#ASSIGNMENT_WH@ASSIGNMENT_DB.MY_SCHEMA>PUT file:///Users/owais/Documents/employee_dataset.csv @staging_area;
+-----+-----+-----+-----+-----+-----+-----+-----+
| source      | target          | source_size | target_size | source_compression | target_compression | status    | message   |
+-----+-----+-----+-----+-----+-----+-----+-----+
| employee_dataset.csv | employee_dataset.csv.gz | 366918 | 104320 | NONE | GZIP | UPLOADED |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 Row(s) produced. Time Elapsed: 1.058s
```

## 9. Loading data to our stage using the “copy into” command.

### 1. Loading via internal Staging

Now, since we already have data loaded into our stage, we will need a table where we will be storing our data, providing column names & structure to the csv file.

```
OWAIS#ASSIGNMENT_WH@ASSIGNMENT_DB.MY_SCHEMA>CREATE OR REPLACE TABLE emp_table (
  EmpID INT PRIMARY KEY,
  FirstName VARCHAR(255),
  LastName VARCHAR(255),
  StartDate DATE,
  ExitDate DATE,
  Title VARCHAR(255),
  Supervisor VARCHAR(255),
  ADEmail VARCHAR(255),
  DOB STRING,
  PerformanceScore VARCHAR(255),
  elt_ts TIMESTAMP,
  elt_by VARCHAR(255),
  file_name VARCHAR(255)
);

+-----+
| status |
+-----+
| Table EMP_TABLE successfully created. |
+-----+
1 Row(s) produced. Time Elapsed: 0.454s
```

- After we have the table, we will use the “COPY INTO” command which serves as a pipeline to bring staged data to our existing table in the warehouse.

```
OWAIS#ASSIGNMENT_WH@ASSIGNMENT_DB.MY_SCHEMA>COPY INTO emp_table
  FROM (
    SELECT
      $1,
      $2,
      $3,
      TO_DATE($4, 'DD-Mon-YYYY'),
      TO_DATE($5, 'DD-Mon-YYYY'),
      $6,
      $7,
      $8,
      $9,
      $10,
      CURRENT_TIMESTAMP(),
      'AppName',
      METADATA$FILENAME
    FROM @staging_area/employee_dataset.csv
  )
  FILE_FORMAT = (FORMAT_NAME = 'my_format');

+-----+-----+-----+-----+-----+
| file           | status | rows_parsed | rows_loaded | error_limit | errors_seen |
| name          |       |             |             |             |             |
+-----+-----+-----+-----+-----+
| staging_area/employee_dataset.csv.gz | LOADED |      3000 |        3000 |          1 |          0 |
+-----+-----+-----+-----+-----+
1 Row(s) produced. Time Elapsed: 0.794s
```

## Here we can see data being pipelined to our Snowflake.

EMPID	FIRSTNAME	LASTNAME	STARTDATE	EXITDATE	TITLE	SUPERVISOR	ADEMAIL	DOB	PERFORMANCESCORE	ELT_TS	ELT_BY	FILE_NAME
3427	Uriah	Bridges	0019-09-20	null	Production Technician I	Peter O'Neill	uriyah.bridges@bilearner.com	07-10-1969	Fully Meets	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz
3428	Paula	Small	0023-02-11	null	Production Technician I	Renee McCormick	paula.small@bilearner.com	30-08-1965	Fully Meets	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz
3429	Edward	Buck	0018-12-10	null	Area Sales Manager	Crystal Walker	edward.buck@bilearner.com	06-10-1991	Fully Meets	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz
3430	Michael	Riordan	0021-06-21	null	Area Sales Manager	Rebekah Wright	michael.riordan@bilearner.com	04-04-1998	Fully Meets	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz
3431	Jasmine	Onque	0019-06-29	null	Area Sales Manager	Jason Kim	jasmine.onque@bilearner.com	29-08-1969	Fully Meets	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz
3432	Maruk	Fraval	0020-01-17	null	Area Sales Manager	Sheri Campos	maruk.fraval@bilearner.com	03-04-1949	Fully Meets	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz
3433	Lata	Costa	0022-04-06	0023-07-03	Area Sales Manager	Jacobo Braun	lata.costa@bilearner.com	01-07-1942	Exceeds	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz
3434	Sharlene	Terry	0020-11-06	0023-01-29	Area Sales Manager	Tracy Marquez	sharlene.terry@bilearner.com	07-03-1957	Fully Meets	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz
3435	Jac	McKinzie	0018-08-18	null	Area Sales Manager	Sharon Becker	jac.mckinzie@bilearner.com	15-05-1974	Exceeds	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz
3436	Joseph	Martins	0022-01-21	0023-06-29	Area Sales Manager	George Jenkins	joseph.martins@bilearner.com	11-11-1949	Fully Meets	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz
3437	Myriam	Givens	0023-08-04	null	Area Sales Manager	Troy White	myriam.givens@bilearner.com	26-01-1964	Fully Meets	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz
3438	Dheepa	Nguyen	0018-08-10	0019-11-04	Area Sales Manager	Brian Miller	dheepa.nguyen@bilearner.com	06-04-1948	Fully Meets	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz
3439	Bartholemew	Khemmich	0022-05-25	0022-11-27	Area Sales Manager	Charles Parks	bartholemew.khemmich@bilearner.com	24-11-1981	Fully Meets	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz
3440	Xana	Potts	0019-12-05	0023-02-17	Area Sales Manager	Gregory Walker	xana.potts@bilearner.com	06-11-1951	Fully Meets	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz
3441	Prater	Jeremy	0019-04-28	null	Area Sales Manager	Tyler Lewis	prater.jeremy@bilearner.com	21-11-1989	Exceeds	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz
3442	Kaylah	Moon	0019-07-09	0022-06-16	Area Sales Manager	Ashley Scott	kaylah.moon@bilearner.com	24-11-1952	Exceeds	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz
3443	Kristen	Tate	0021-04-05	0023-05-12	Area Sales Manager	Lauren Jones	kristen.tate@bilearner.com	08-04-1994	Fully Meets	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz
3444	Bobby	Rodgers	0021-11-28	0022-02-04	Area Sales Manager	Matthew Jackson	bobby.rodgers@bilearner.com	15-11-1983	Fully Meets	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz
3445	Reid	Park	0021-01-16	null	Area Sales Manager	Michelle Mitchell	reid.park@bilearner.com	07-12-1985	Exceeds	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz
3446	Hector	Dalton	0021-08-24	null	Area Sales Manager	Sydney French	hector.dalton@bilearner.com	01-05-1996	Exceeds	2024-05-27 14:52:47.338	TAS205	employee_dataset.csv.gz

## 8. B. Loading via External Stage

Our dataset is currently in Amazon S3 Bucket.

The screenshot shows the AWS CloudWatch Metrics interface. At the top, a green banner indicates "Upload succeeded" with the message "View details below." Below this, a note says "The information below will no longer be available after you navigate away from this page." The main section is titled "Summary" and shows the destination as "s3://tas-205". It displays two rows: "Succeeded" with "1 file, 358.3 KB (100.00%)" and "Failed" with "0 files, 0 B (0%)". Below this is a tabbed section with "Files and folders" selected, showing a table of one file: "employee\_dataset.csv" (text/csv, 358.3 KB, Succeeded). The bottom of the screen includes standard AWS navigation links like CloudShell, Feedback, and Copyright information.

But to access our bucket, we need to create an IAM role assigning to a user with access to read files from s3 bucket.

The screenshot shows the AWS IAM interface. The first part is the "s3user" info page, which displays basic user details: ARN (arn:aws:siam::590183764903:user/s3user), Console access (Disabled), and Access key 1 (Create access key). It also shows the creation date (June 03, 2024) and last sign-in. Below this is the "Permissions" tab, which lists attached policies: "AmazonS3FullAccess". The second part is the "assignment\_role\_s3" info page, which shows the role's ARN (arn:aws:iam::590183764903:role/assignment\_role\_s3), creation date (May 03, 2024), and maximum session duration (1 hour). It also has a "Permissions" tab listing the same "AmazonS3FullAccess" policy.

We will have to create integration from AWS S3 to Snowflake but integration is allowed using ACCOUNTADMIN and SYSADMIN roles only. Hence :-

1. First we shift to role ACCOUNTADMIN
2. Create Integration with S3 bucket using ARN of the role we made with s3 bucket access.
3. Grant usage permission of that integration to role ADMIN.

```
OWAIS#ASSIGNMENT_WH@ASSIGNMENT_DB.MY_SCHEMA>use ROLE ACCOUNTADMIN;
+-----+
| status |
+-----+
| Statement executed successfully. |
+-----+
1 Row(s) produced. Time Elapsed: 0.196s
OWAIS#ASSIGNMENT_WH@ASSIGNMENT_DB.MY_SCHEMA>CREATE OR REPLACE STORAGE INTEGRATION s3_to_snowflake
    type = external_stage
    storage_provider=s3
    enabled = true
    storage_aws_role_arn = "arn:aws:iam::590183764903:role/assignment_role_s3"
    storage_allowed_locations = ('s3://tas-205/');
+-----+
| status |
+-----+
| Integration S3_TO_SNOWFLAKE successfully created. |
+-----+
1 Row(s) produced. Time Elapsed: 0.179s
OWAIS#ASSIGNMENT_WH@ASSIGNMENT_DB.MY_SCHEMA>GRANT USAGE ON INTEGRATION s3_to_snowflake TO ROLE ADMIN;
+-----+
| status |
+-----+
| Statement executed successfully. |
+-----+
1 Row(s) produced. Time Elapsed: 0.161s
OWAIS#ASSIGNMENT_WH@ASSIGNMENT_DB.MY_SCHEMA>use ROLE ADMIN;
+-----+
| status |
+-----+
| Statement executed successfully. |
+-----+
1 Row(s) produced. Time Elapsed: 0.146s
```

Next, we create our external stage from s3 data source URL with our file format over the created storage integration.

```
OWAIS#ASSIGNMENT_WH@ASSIGNMENT_DB.MY_SCHEMA>CREATE OR REPLACE STAGE external_stage
    STORAGE_INTEGRATION = s3_to_snowflake
    URL = 's3://tas-205/employee_dataset.csv'
    file_format = new_format;
+-----+
| status |
+-----+
| Stage area EXTERNAL_STAGE successfully created. |
+-----+
1 Row(s) produced. Time Elapsed: 2.193s
```

## 9. B. Loading data via External Loading

To grant our stage to fetch data from s3, we will need to configure trust policy of our S3 role with **STORAGE\_AWS\_IAM\_USER\_ARN** & **STORAGE\_AWS\_EXTERNAL\_ID**

[OWAIS#ASSIGNMENT_WH@ASSIGNMENT_DB.MY_SCHEMA>desc integration s3_to_snowflake;			
property	property_type	property_value	property_default
ENABLED	Boolean	true	false
STORAGE_PROVIDER	String	S3	
STORAGE_ALLOWED_LOCATIONS	List	s3://tas-205/	[]
STORAGE_BLOCKED_LOCATIONS	List		[]
STORAGE_AWS_IAM_USER_ARN	String	arn:aws:iam::058264540845:user/2r0n0000-s	
STORAGE_AWS_ROLE_ARN	String	arn:aws:iam::590183764903:role/assignment_role_s3	
STORAGE_AWS_EXTERNAL_ID	String	BL39191_SFCRole=2_J6d9G0w/vWnq/Glqpq8x5koocdc=	
COMMENT	String		
8 Row(s) produced. Time Elapsed: 1.128s			

Trusted entities

Entities that can assume this role under specified conditions.

Edit trust policy

```
1 [ {  
2     "Version": "2012-10-17",  
3     "Statement": [  
4         {  
5             "Effect": "Allow",  
6             "Principal": {  
7                 "AWS": "arn:aws:iam::058264540845:user/2r0n0000-s"  
8             },  
9             "Action": "sts:AssumeRole",  
10            "Condition": {  
11                "StringEquals": {  
12                    "sts:ExternalId": "BL39191_SFCRole=2_J6d9G0w/vWnq/Glqpq8x5koocdc="  
13                }  
14            }  
15        }  
16    ]  
17 }
```

Next Step is to create the table we will be loading our data in

```
34 CREATE OR REPLACE TABLE ext_emp_table (  
35     EmpID INT PRIMARY KEY,  
36     FirstName VARCHAR(255),  
37     LastName VARCHAR(255),  
38     StartDate DATE,  
39     ExitDate DATE,  
40     Title VARCHAR(255),  
41     Supervisor VARCHAR(255),  
42     ADEmail VARCHAR(255),  
43     DOB STRING,  
44     PerformanceScore VARCHAR(255),  
45     elt_ts TIMESTAMP,  
46     elt_by VARCHAR(255),  
47     file_name VARCHAR(255)  
48 );
```

Results

status
1 Table EXT_EMP_TABLE successfully created.

Query Details

- Query duration 185ms
- Rows 1
- Query ID 01b4bfe7-3201-1a21-0...

## Similar COPY INTO command to load data in the table.

```

49
50   COPY INTO ext_emp_table
51     FROM (
52       SELECT
53         $1,
54         $2,
55         $3,
56         TO_DATE($4, 'DD-Mon-YYYY'),
57         TO_DATE($5, 'DD-Mon-YYYY'),
58         $6,
59         $7,
60         $8,
61         $9,
62         $10,
63         CURRENT_TIMESTAMP(),
64         'TAS205',
65         METADATA$FILENAME
66       FROM @stage_area/employee_dataset.csv
67     )
68   FILE_FORMAT = (FORMAT_NAME = 'new_format');
69
70

```

	file	status	rows_parsed	rows_loaded	error_limit	errors_seen	first_error	first_error
1	stage_area/employee_dataset.csv.gz	LOADED	3000	3000	1	0	null	

Query Details
...

Query duration
876ms

	EMPID	FIRSTNAME	LASTNAME	STARTDATE	EXITDATE	TITLE	SUPERVISOR	ADEMAIL
1	3427	Uriah	Bridges	0019-09-20	null	Production Technician I	Peter Oneill	uriah.bridges@bilearn.com
2	3428	Paula	Small	0023-02-11	null	Production Technician I	Renee McCormick	paula.small@bilearn.com
3	3429	Edward	Buck	0018-12-10	null	Area Sales Manager	Crystal Walker	edward.buck@bilearn.com
4	3430	Michael	Riordan	0021-06-21	null	Area Sales Manager	Rebekah Wright	michael.riordan@bilearn.com
5	3431	Jasmine	Onque	0019-06-29	null	Area Sales Manager	Jason Kim	jasmine.onque@bilearn.com
6	3432	Maruk	Fraval	0020-01-17	null	Area Sales Manager	Sheri Campos	maruk.fraval@bilearn.com
7	3433	Latia	Costa	0022-04-06	0023-07-03	Area Sales Manager	Jacob Braun	latia.costa@bilearn.com
8	3434	Sharlene	Terry	0020-11-06	0023-01-29	Area Sales Manager	Tracy Marquez	sharlene.terry@bilearn.com
9	3435	Jac	McKinzie	0018-08-18	null	Area Sales Manager	Sharon Becker	jac.mckinzie@bilearn.com
10	3436	Joseph	Martins	0022-01-21	0023-06-29	Area Sales Manager	George Jenkins	joseph.martins@bilearn.com
11	3437	Myriam	Givens	0023-08-04	null	Area Sales Manager	Troy White	myriam.givens@bilearn.com
12	3438	Dheepa	Nguyen	0018-08-10	0019-11-04	Area Sales Manager	Brian Miller	dheepa.nguyen@bilearn.com
13	3439	Bartholemew	Khemmich	0022-05-25	0022-11-27	Area Sales Manager	Charles Parks	bartholemew.khemmich@bilearn.com
14	3440	Xana	Potts	0019-12-05	0023-02-17	Area Sales Manager	Gregory Walker	xana.potts@bilearn.com
15	3441	Prater	Jeremy	0019-04-28	null	Area Sales Manager	Tyler Lewis	prater.jeremy@bilearn.com
16	3442	Kaylah	Moon	0019-07-09	0022-06-16	Area Sales Manager	Ashley Scott	kaylah.moon@bilearn.com
17	3443	Kristen	Tate	0021-04-05	0023-05-12	Area Sales Manager	Lauren Jones	kristen.tate@bilearn.com
18	3444	Bobby	Rodgers	0021-11-28	0022-02-04	Area Sales Manager	Matthew Jackson	bobby.rodgers@bilearn.com
19	3445	Reid	Park	0021-01-16	null	Area Sales Manager	Michelle Mitchell	reid.park@bilearn.com
20	3446	Hector	Dalton	0021-09-24	null	Area Sales Manager	Sudanov French	hector.dalton@bilearn.com

Query Details
...

Query duration
380ms

Rows
20

Query ID
01b4bfed-3201-1a21-0...

EMPID
#

3427
3446

FIRSTNAME
A

100% filled

LASTNAME
A

100% filled

STARTDATE
(L)

100% filled

EXITDATE
(L)

45% filled
55% null

## 10. Upload any parquet file to the stage location and infer the schema of the file

```
17  
20 CREATE OR REPLACE FILE FORMAT my_parquet_format  
21   TYPE = 'PARQUET'  
22   SNAPPY_COMPRESSION = TRUE;  
23  
24 CREATE OR REPLACE STAGE PARQUET_STAGE FILE_FORMAT = my_parquet_format;  
25  
26 SELECT * FROM TABLE(  
27   infer_schema( location=>'@parquet_stage/test1.parquet',  
28     file_format=>'my_parquet_format')  
29 );  
30
```

↳ Results    ↵ Chart

	COLUMN_NAME	TYPE	NULLABLE	EXPRESSION	FILENAMES	ORDER_ID
1	ID	NUMBER(38, 0)	TRUE	\$1:ID::NUMBER(38, 0)	test1.parquet	0
2	NAME	TEXT	TRUE	\$1:NAME::TEXT	test1.parquet	1
3	PHONE NUMBER	TEXT	TRUE	\$1:PHONE NUMBER::TEXT	test1.parquet	2
4	EMAIL	TEXT	TRUE	\$1:EMAIL::TEXT	test1.parquet	3
5	DESIGNATION	TEXT	TRUE	\$1:DESIGNATION::TEXT	test1.parquet	4
6	EXPERIENCE	NUMBER(38, 0)	TRUE	\$1:EXPERIENCE::NUMBER(38, 0)	test1.parquet	5

## Parquet File Source: Generated via python code

```
PARQ.PY > ...  
1 import pandas as pd  
2 import pyarrow as pa  
3 import pyarrow.parquet as pq  
4  
5 # Create a DataFrame with the specified columns and 10 entries  
6 data = {  
7   'ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],  
8   'NAME': ['Alice', 'Bob', 'Charlie', 'David', 'Eva', 'Fiona', 'George', 'Hannah', 'Ian', 'Jane'],  
9   'PHONE NUMBER': ['1234567890', '0987654321', '1122334455', '1233214567', '9876543210', '8765432109', '7654321098', '6543210987',  
10  'EMAIL': [  
11    'alice@example.com', 'bob@example.com', 'charlie@example.com',  
12    'david@example.com', 'eva@example.com', 'fiona@example.com',  
13    'george@example.com', 'hannah@example.com', 'ian@example.com', 'jane@example.com'  
14  ],  
15  'DESIGNATION': ['Manager', 'Analyst', 'Clerk', 'Lead', 'Director', 'Coordinator', 'Officer', 'Supervisor', 'Engineer', 'Developer'],  
16  'EXPERIENCE': [5, 3, 10, 2, 12, 6, 8, 1, 9, 7]  
17 }  
18 df = pd.DataFrame(data)  
19  
20 # Convert DataFrame to a Table and write to Parquet  
21 table = pa.Table.from_pandas(df)  
22 pq.write_table(table, 'test1.parquet')
```

## 11. Run a select query on the staged parquet file without loading it to a snowflake table.

The screenshot shows a Snowflake query editor interface. At the top, there are tabs for 'ASSIGNMENT\_DB.MY\_SCHEMA' and 'Settings'. On the right, there are buttons for 'Code Versions' and a magnifying glass icon. The code area contains the following SQL:

```
30
31  SELECT * FROM @parquet_stage/test1.parquet (FILE_FORMAT => 'my_parquet_format');
32
33
34
35
```

Below the code, there are two tabs: 'Results' (which is selected) and 'Chart'. The results pane displays 10 rows of data:

\$1
{ "DESIGNATION": "Manager", "EMAIL": "alice@example.com", "EXPERIENCE": 5, "ID": 1, "NAME": "Alice", "PHONE NUMBER": "1234567890"}
{ "DESIGNATION": "Analyst", "EMAIL": "bob@example.com", "EXPERIENCE": 3, "ID": 2, "NAME": "Bob", "PHONE NUMBER": "0987654321"}
{ "DESIGNATION": "Clerk", "EMAIL": "charlie@example.com", "EXPERIENCE": 10, "ID": 3, "NAME": "Charlie", "PHONE NUMBER": "11223344"}
{ "DESIGNATION": "Lead", "EMAIL": "david@example.com", "EXPERIENCE": 2, "ID": 4, "NAME": "David", "PHONE NUMBER": "1233214567"}
{ "DESIGNATION": "Director", "EMAIL": "eva@example.com", "EXPERIENCE": 12, "ID": 5, "NAME": "Eva", "PHONE NUMBER": "9876543210"}
{ "DESIGNATION": "Coordinator", "EMAIL": "fiona@example.com", "EXPERIENCE": 6, "ID": 6, "NAME": "Fiona", "PHONE NUMBER": "876543210"}
{ "DESIGNATION": "Officer", "EMAIL": "george@example.com", "EXPERIENCE": 8, "ID": 7, "NAME": "George", "PHONE NUMBER": "76543210"}
{ "DESIGNATION": "Supervisor", "EMAIL": "hannah@example.com", "EXPERIENCE": 1, "ID": 8, "NAME": "Hannah", "PHONE NUMBER": "6543"}
{ "DESIGNATION": "Engineer", "EMAIL": "ian@example.com", "EXPERIENCE": 9, "ID": 9, "NAME": "Ian", "PHONE NUMBER": "5432109876"}
{ "DESIGNATION": "Developer", "EMAIL": "jane@example.com", "EXPERIENCE": 7, "ID": 10, "NAME": "Jane", "PHONE NUMBER": "4321098765"}

On the right side of the results pane, there is a 'Query Details' panel with the following information:

- Query duration: 217ms
- Rows: 10
- Query ID: 01b54124-3201-1f73-00...

Below the details panel, there is a preview section showing the first few columns of the data:

\$1	[ ]
DESIGNATION	10
EMAIL	10
EXPERIENCE	10

There is also a link '+ 3 more' at the bottom of the preview section.

**12. 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**

```

48
49 USE ROLE ACCOUNTADMIN;
50
51 GRANT USAGE ON DATABASE ASSIGNMENT_DB TO ROLE PII;
52 GRANT USAGE ON DATABASE ASSIGNMENT_DB TO ROLE DEVELOPER;
53 GRANT USAGE ON SCHEMA ASSIGNMENT_DB.MY_SCHEMA TO ROLE PII;
54 GRANT USAGE ON SCHEMA ASSIGNMENT_DB.MY_SCHEMA TO ROLE DEVELOPER;
55 GRANT SELECT ON ALL TABLES IN SCHEMA ASSIGNMENT_DB.MY_SCHEMA TO ROLE PII;
56 GRANT SELECT ON ALL TABLES IN SCHEMA ASSIGNMENT_DB.MY_SCHEMA TO ROLE DEVELOPER;
57 GRANT USAGE ON WAREHOUSE ASSIGNMENT_WH TO ROLE PII;
58 GRANT USAGE ON WAREHOUSE ASSIGNMENT_WH TO ROLE DEVELOPER;
59
60
61 -- Grants the ability to insert, update, delete, and select data from tables
62 GRANT INSERT, UPDATE, DELETE, SELECT ON ALL TABLES IN SCHEMA MY_SCHEMA TO ROLE developer;
63
64 use role developer;
65 -- Masking policy for email
66 CREATE OR REPLACE MASKING POLICY email_masking_policy AS (val STRING) RETURNS STRING ->
67     CASE
68         WHEN CURRENT_ROLE() IN ('PII') THEN val
69         ELSE '****masked****'
70     END;
71
72 -- Masking policy for phone number
73 CREATE OR REPLACE MASKING POLICY phone_number_masking_policy AS (val STRING) RETURNS STRING ->
74     CASE
75         WHEN CURRENT_ROLE() IN ('PII') THEN val
76         ELSE '****masked****'
77     END;
78
79 -- Apply masking policy to the email column
80 ALTER TABLE my_table MODIFY COLUMN email SET MASKING POLICY email_masking_policy;
81
82 -- Apply masking policy to the phone_number column
83 ALTER TABLE my_table MODIFY COLUMN phone_number SET MASKING POLICY phone_number_masking_policy;
84
85
86 | select * from my_table;
87

```

Results

ID	NAME	PHONE_NUMBER	EMAIL	DESIGNATION	EXPERIENCE
1	Alice	****masked****	****masked****	Manager	5
2	Bob	****masked****	****masked****	Analyst	3
3	Charlie	****masked****	****masked****	Clerk	10
4	David	****masked****	****masked****	Lead	2
5	Eva	****masked****	****masked****	Director	12
6	Fiona	****masked****	****masked****	Coordinator	6
7	George	****masked****	****masked****	Officer	8
8	Hannah	****masked****	****masked****	Supervisor	1
9	Ian	****masked****	****masked****	Engineer	9
10	Jane	****masked****	****masked****	Developer	7

## Using Role PII:

```
89  use role pii;
90  select * from my_table;
91
92
```

↳ Results    ↵ Chart

	ID	NAME	PHONE_NUMBER	EMAIL	DESIGNATION	EXPERIENCE
1	1	Alice	null	alice@example.com	Manager	5
2	2	Bob	null	bob@example.com	Analyst	3
3	3	Charlie	null	charlie@example.com	Clerk	10
4	4	David	null	david@example.com	Lead	2
5	5	Eva	null	eva@example.com	Director	12
6	6	Fiona	null	fiona@example.com	Coordinator	6
7	7	George	null	george@example.com	Officer	8
8	8	Hannah	null	hannah@example.com	Supervisor	1
9	9	Ian	null	ian@example.com	Engineer	9
10	10	Jane	null	jane@example.com	Developer	7

- The End -