Snowflake Peer Review

Akash's Approach

Building Roles, Databases and Schemas

- At first three roles are created: admin, developer, pii.
- Then developer role was granted to admin role which was further granted to accountadmin, pii was separately granted to accountadmin to build the specified hierarchy.
- Then a `MEDIUM` sized warehouse was created named `assignment_wh`. To give `admin` access to the warehouse, `GRANT USAGE` is used by accountadmin. Also, the `WITH GRANT OPTION` is mentioned to allow `admin` to grant the same privelege to other roles.
- Then to create database and schema, the respective priveleges of `CREATE DATABASE` and `CREATE SCHEMA` are granted to admin via accountadmin.
- Assignment_db database and schema named my_schema are created by admin.

Creating Stages and Tables

Via Internal Staging

- Created an internal stage - `local_upload_csv`. Uploaded a csv file(*employees.csv*) to the stage using `PUT`.
- Then made a file format(`employee_csv_format`) for the csv with `FIELD_OPTIONALLY_ENCLOSED_BY` set to `"`. This tells that some values might have comma in them but to not treat the comma as field separator if the value is surrounded by double quotes.
- Then created a table - `employees` and unloaded the csv file from the stage into it using `COPY INTO ... FROM (SELECT ...)`. Three other columns - `elt_ts` set to `DEFAULT CURRENT TIMESTAMP`, `elt_by` set to `DEFAULT 'LOCAL'` and `file_name` set to `METADATA$FILENAME` were also added while unloading.

Via External Staging

- First granted `CREATE INTEGRATION` to admin via accountadmin.
- Then created storage integration named `aws_integration` and assigned it to a stage `aws_external_upload`.
- Then a table `employees_external` is created in exactly the same way as in the internal stage (first table is created and the data is unloaded using same parameters)
- To create a variant version another table `employees_external_variant` is created. This is done via `CREATE TABLE employee_external_variant(col VARIANT) AS (SELECT ...)` command; along with that `PARSE_JSON` is used to parse and convert the stage data to a variant format.

Working with parquet

- First a file format `parquet_format` was created for the subsequent queries.
- The schema of parquet file is inferred using `INFER_SCHEMA`
- Then different nested objects were queried using the `$` syntax like `$1:continent` and `$1:country:city` in the `SELECT` query.

Masking

- A masking policy `mask_country` was created for the `country` column. The `employees` and `employees_external` were subsequently altered and the masking policy was set.
- Then permissions on warehouse, schema and tables were granted to developer and pii roles. `SELECT` was used to observe the masking effect.