EPAM’s Snowflake Hands-on Lab Report

## **Database creation**

A picture containing graphical user interface

Description automatically generated

## **Data loading**

Command like this executed for all files to put into stage:



Graphical user interface, text, application

Description automatically generated

List of files put into stage:

Graphical user interface, application, table

Description automatically generated

## **ELT Data Workflow**

**All the code for this task is in zip archive, it contains all SQL statements which were used.**

First, required schemas were created:

Graphical user interface, application

Description automatically generated

I suggest following transformation pipeline (added raw layer to store data as is):

stage -> raw (dirty data) -> core\_dwh (enhanced and cleansed data in 3NF) -> data\_mart

Following tasks have been created:

* dummy task which initiates whole pipeline
* for each table task which places data from stage to raw
* for each table task which takes data from raw, transform and clean it and put to core\_dwh
* task which uses stored procedure to add data to data\_mart layer once all data is in core\_dwh

As a result, following flow appeared:

Diagram

Description automatically generated

After executing ‘INITIATE\_PIPELINE’ task all consequent tables are filled accordingly step by step.

For filling RAW layer COPY INTO approach has been used, it understands which data has been already uploaded and adds only new data using mechanisms under the hood.

Table streams were used to insert only new values to tables when transfer from RAW to CORE\_DWH:

Graphical user interface, application

Description automatically generated

For filling DATA\_MART layer stored procedure executing multiple MERGE INTO statements was used. It is supposed to skip already filled values as well.

As a result, there are following tables and other objects in each layer:

Graphical user interface, application

Description automatically generated Graphical user interface, table

Description automatically generated with medium confidenceGraphical user interface, application

Description automatically generated

## **Snowflake & 3rd party tools**

I decided to analyze number of customers per country, to do this I created a view in using following query:

Text

Description automatically generated

Result:

Table

Description automatically generated

As it is seen, the most customers are in Vietnam, and least in China.

Let us visualize it using Power BI. First – connect to Snowflake and extract data from the created view and then create a dashboard.

For clear representation gradient showing number of customers has been created: green – the most customers, red – the least, which is completely correspond with result which we have seen in the created view.

Map

Description automatically generated

We can also use variety of DB tools such as DBeaver to connect to Snowflake and query data:

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generated

## **Snowflake SQL**

For benchmarking I decided to use heavy query which were not presented in provided file, to create a query which use multiple joins on tables containing most rows and take all columns using select \*:

Text

Description automatically generated

And used 2 different WH: X-Small and Medium and here are results.

For X-Small:

Diagram

Description automatically generated

For Medium:

Graphical user interface

Description automatically generated

As expected, medium-size cluster took much less time to execute query and provide results than x-small – 18 seconds instead of 30 seconds.

If do not keep “alter session set use\_cached\_result = false;” statement Snowflake will use cached result and return results instantly.

Regarding using SnowSQL CLI client, I used it already when put data to stages from local. (check 2 task)

## **Other Snowflake features**

## 

I learnt some useful Snowflake features:

* Object cloning allows to easily clone any objects: schemas, tables etc.

Graphical user interface, application, table

Description automatically generated with medium confidence

* Time travel allows to query some historical snapshot of the data within data retention period which is 1 day default and can be manually installed for objects up to 90 days.

Table

Description automatically generated

* Shares can be used for sharing some data between different accounts. Example:

