EPAM’s Snowflake Hands-on Lab

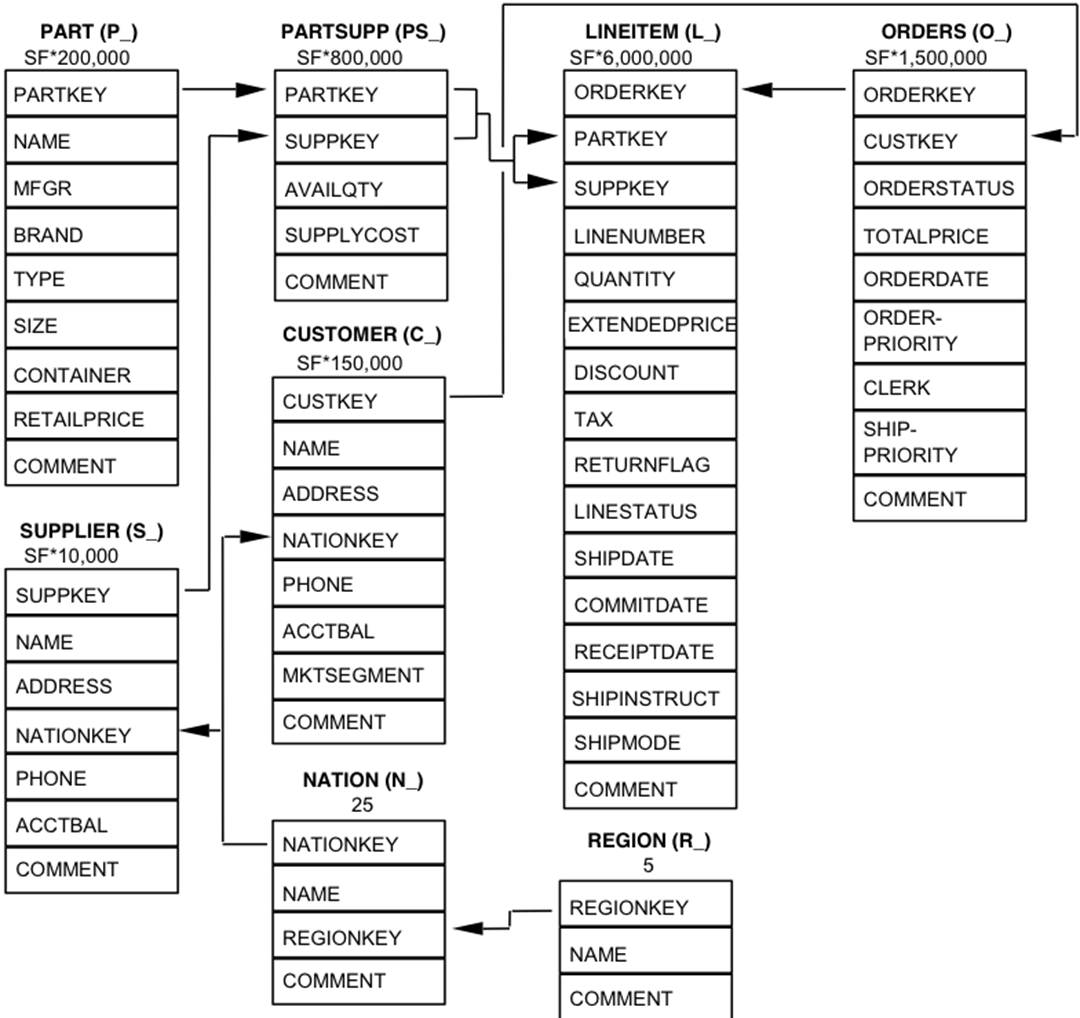
# Lab Overview

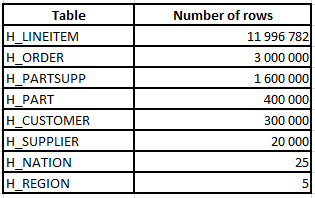
This Lab (prepared by your EPAM colleagues) offers a high-level description of the practical task for self-directed learning.

The target group for the Lab are DWBI engineers with experience in building Data Warehouses using other databases (Oracle, MS SQL, Teradata, etc.).

# Lab Data Set

Data set from [TPC-H benchmark](http://www.tpc.org/tpch/) is proposed for the Lab. TPC-H allows you to generate data for 8 tables. The data volume (in gigabytes) is defined by scale factor (SF). For the Lab purpose, you can [download](https://epam-my.sharepoint.com/:f:/p/maksim_krupenin/Ehu35G00GDVEt--YeQhrYc0BB63ueaqLIqU-zt4r9Dx4pQ?e=X12K0F) prepared in advance data set (2 GB of raw data, SF=2):





# In the shared folder, you can also find DDL script for the tables: *tpch\_ddl.sql*.

# Lab Description

Hands-on-lab is considered as completed if you score >= 60 points.

(Tasks 1, 8 – 5 points each, Tasks 2, 4, 5, 6 – 10 points each, Task 3 – 30 points, Task 7 – 20 points).

## Database creation

First, you need to create a separate database EPAM\_LAB in Snowflake.

## Data loading

In this step, you need to load Lab data set to an internal (Snowflake) or external stage. If you have an existing account in AWS/GCP/Azure cloud, an external stage would be preferable. Please note that you may need some data preparation steps before loading.

## ELT Data Workflow

Create two schemas in the DB you created before:

* CORE\_DWH
* DATA\_MART

Develop the following automated data workflow:

Stage -> CORE\_DWH -> DATA\_MART

Data in CORE\_DWH should be modeled according to 3NF (as is - no transformation). Star Schema is a target data model for DATA\_MART (data should be transformed accordingly).

The following Snowflake features should be used:

* Tasks
* Stored Procedures
* Tables Streams

*Note: No need to spend a lot of time on the modeling Star Schema implementing all nuances. You should rather focus on Snowflake’s capabilities.*

## Snowflake & 3rd party tools

When the data is loaded to DATA\_MART schema, connect Snowflake as a data source from any BI tool (Tableau, PowerBI, Qlik Sense, etc.) and create a simple dashboard.

Also, try connecting to Snowflake from any SQL editor (e.g. [DBeaver](https://dbeaver.io/)).

1. Snowflake SQL

From the shared folder you can also [download](https://epam-my.sharepoint.com/:f:/p/maksim_krupenin/Ehu35G00GDVEt--YeQhrYc0BB63ueaqLIqU-zt4r9Dx4pQ?e=X12K0F) the file with 22 TPC-H benchmarking queries (tpch\_benchmark\_queries.sql). Please note that the queries were modified to execute in AWS RedShift database, so some of them may require modifications for Snowflake. Use the queries to test how Snowflake works:

* Create several warehouses of different sizes and compare their performance.
* Test how Snowflake leverages different types of cash.
* Rewrite a couple of queries to execute on the Start Schema data model and compare performance (3NF vs Star Schema).
* Execute queries using SnowSQL (CLI Client).

## Other Snowflake features

Learn and test other interesting Snowflake features:

* Object Cloning
* Time Travel
* Data Sharing - share your DATA\_MART schema with a colleague who helps you with this Lab. Also, you may use a [Reader Account](https://docs.snowflake.com/en/user-guide/data-sharing-reader-create.html).

## Snowpipe

Automated incremental data loading using Snowpipe. Split lineitem & order files into several parts and simulate their sequential loading to stage buckets.

## Additional tasks

Connect your Snowflake account with partner applications available for a free trial (e.g. Fivetran, Periscope Data, Matillion in Partner Connect menu). Explore how selected tools work.

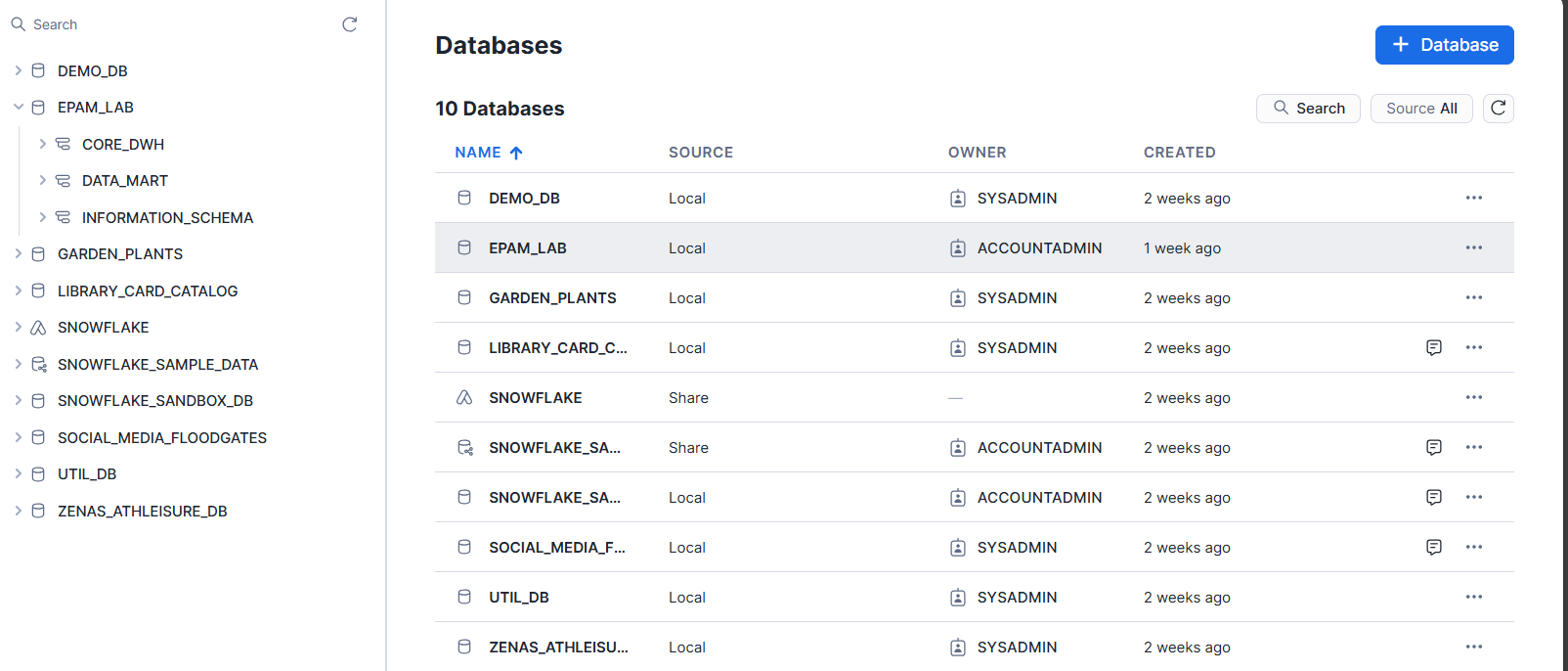
## Database creation.

use role accountadmin;

create database epam\_lab;

drop schema public;

use database epam\_lab;



## Data loading.

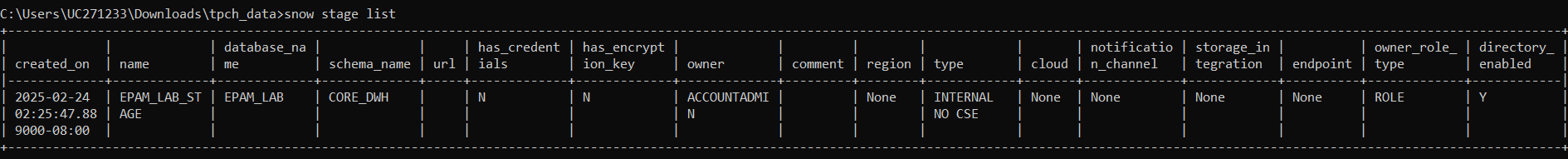
CREATE STAGE epam\_lab\_stage

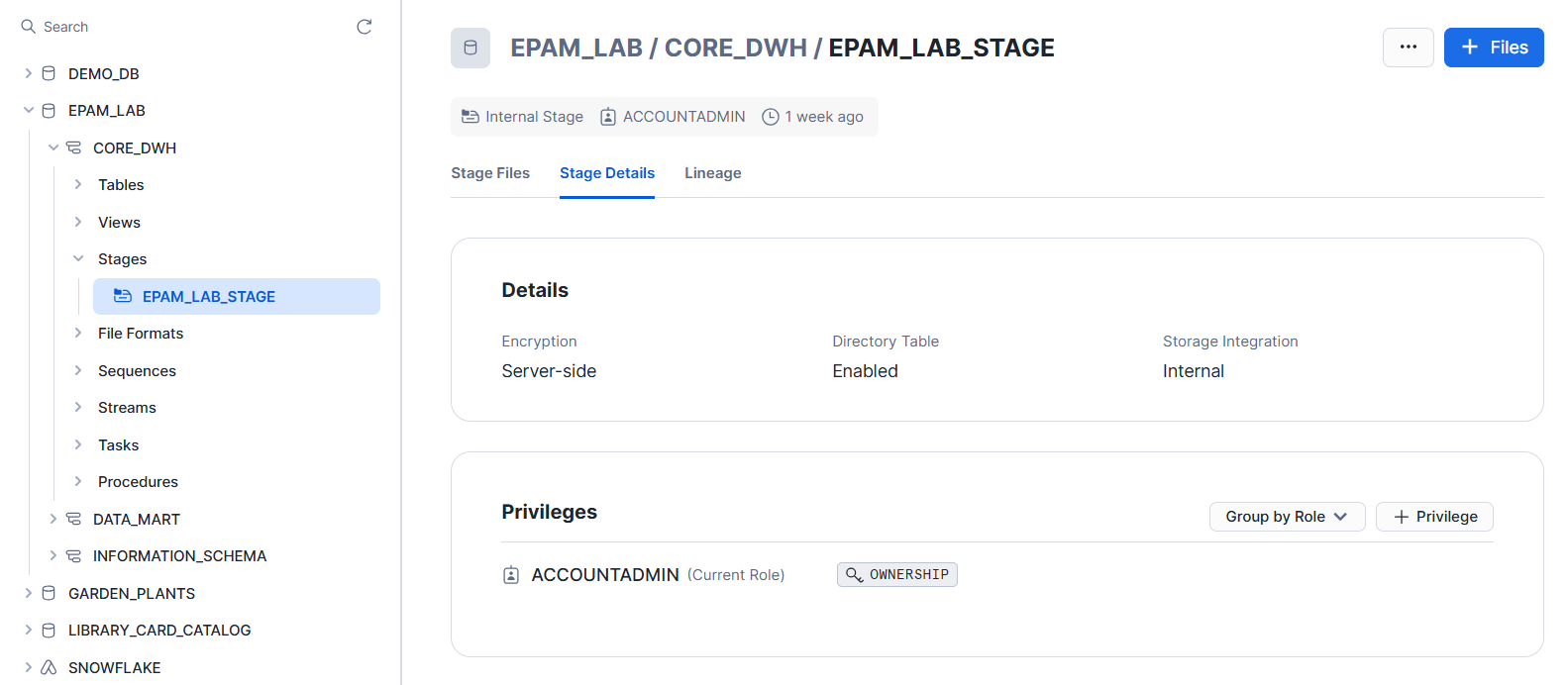
ENCRYPTION = (TYPE = 'SNOWFLAKE\_SSE');

ALTER STAGE epam\_lab\_stage

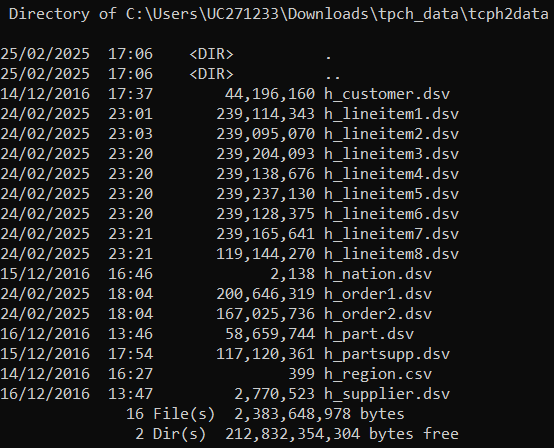
SET DIRECTORY = ( ENABLE = TRUE );

tpch\_data>snow stage list

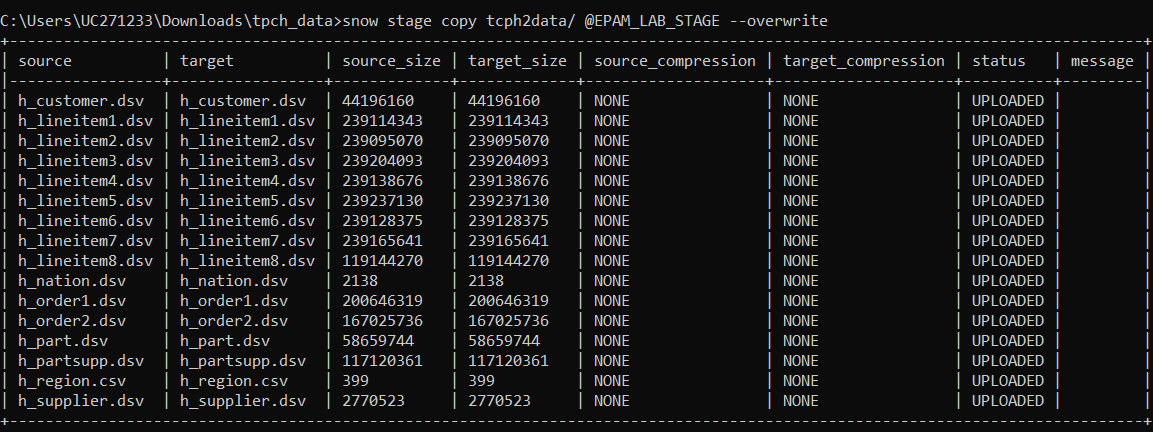




tpch\_data>dir tcph2data\\*



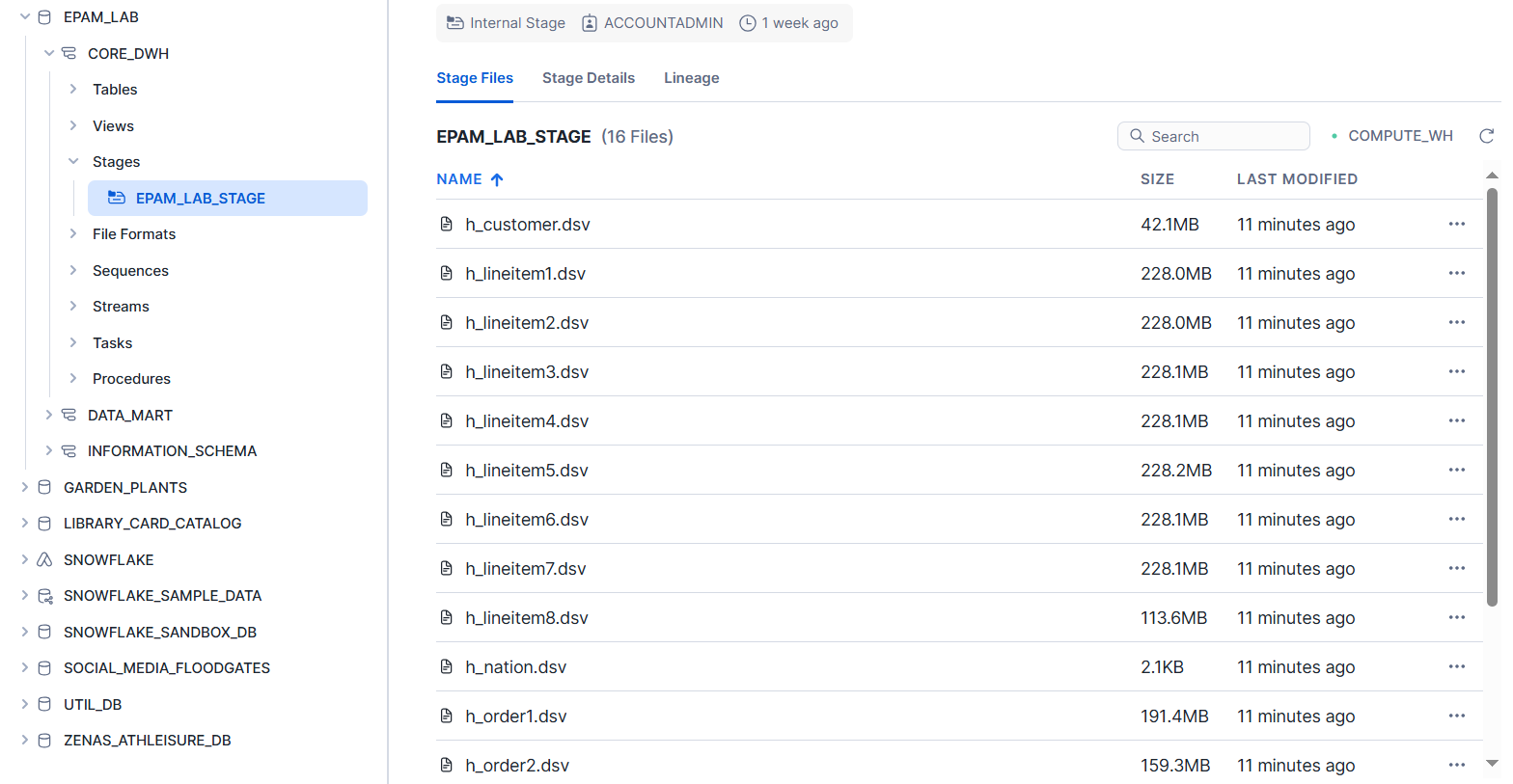
tpch\_data>snow stage copy tcph2data/ @EPAM\_LAB\_STAGE --overwrite



tpch\_data>snow stage list-files @EPAM\_LAB\_STAGE

A screen shot of a computer

AI-generated content may be incorrect.



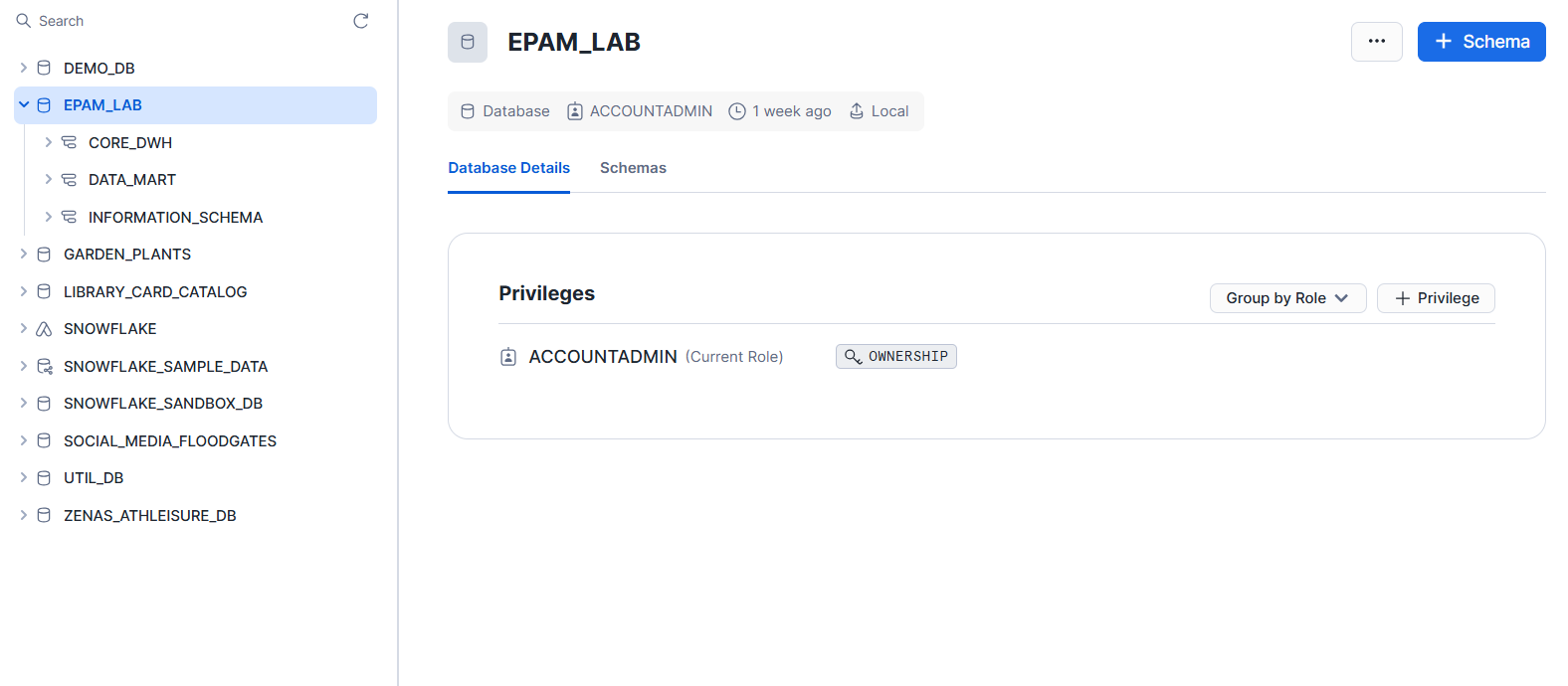
## ELT Data Workflow

* **Create schemas**

use database epam\_lab;

create schema core\_dwh;

create schema data\_mart;



* **Create format files**

use schema core\_dwh;

create or replace file format COMMCOLSEP\_ONEHEADROW

type = 'CSV'

field\_delimiter = ','

skip\_header = 1

field\_optionally\_enclosed\_by = '"'

trim\_space = TRUE;

create or replace file format PIPECOLSEP\_ONEHEADROW

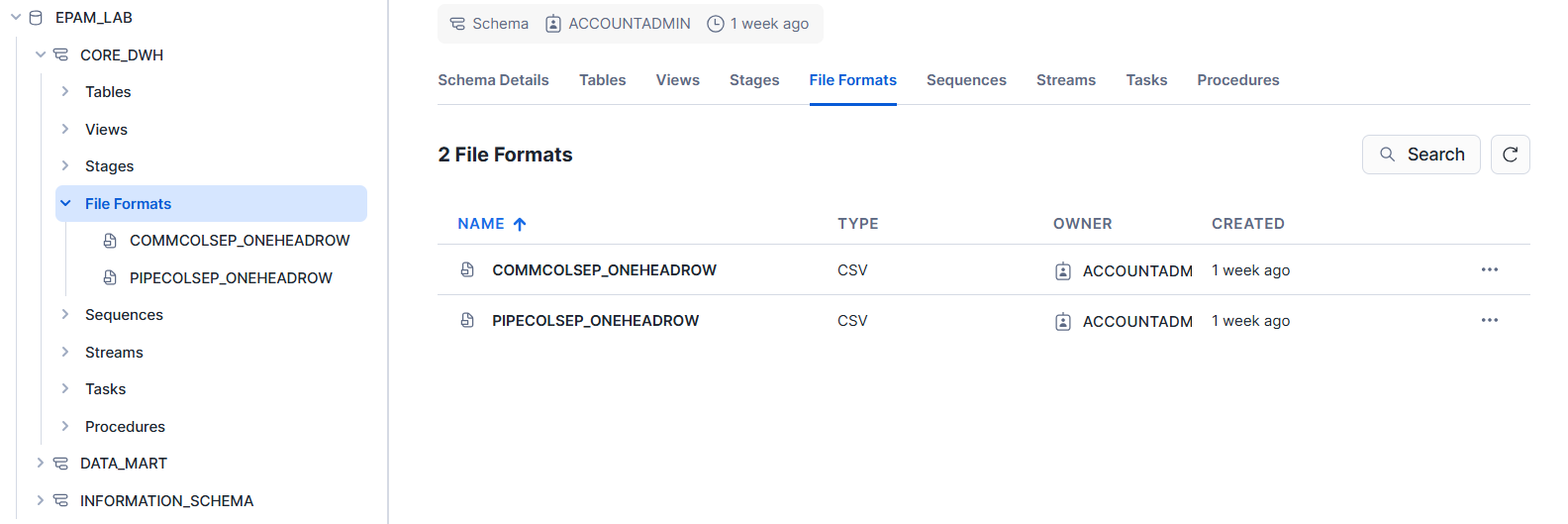
type = 'CSV'

field\_delimiter = '|'

skip\_header = 1

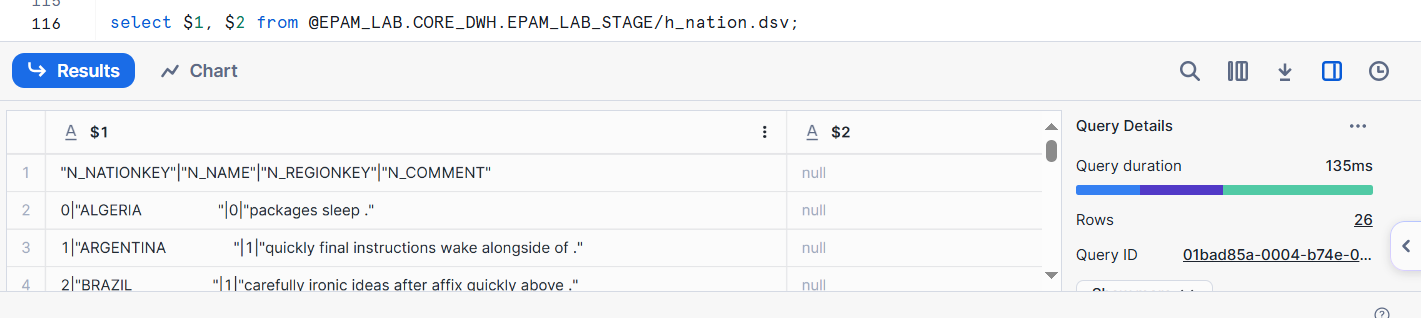
field\_optionally\_enclosed\_by = '"'

trim\_space = TRUE;



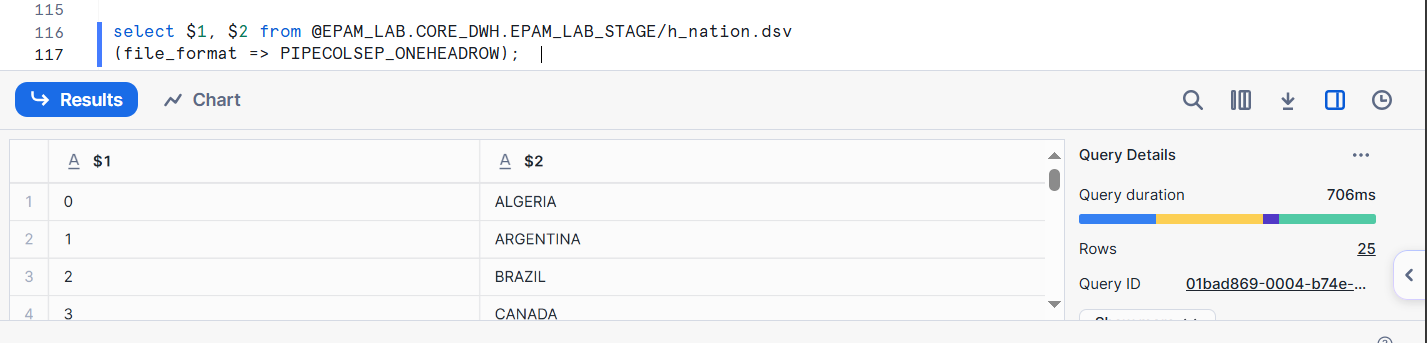
* **Check format files on stage files**

select $1, $2 from @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE/h\_nation.dsv;

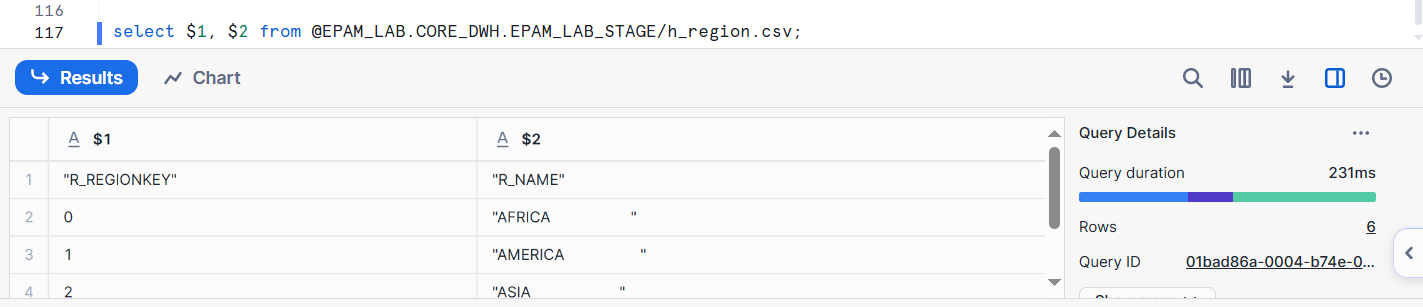


select $1, $2 from @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE/h\_nation.dsv

(file\_format => PIPECOLSEP\_ONEHEADROW);

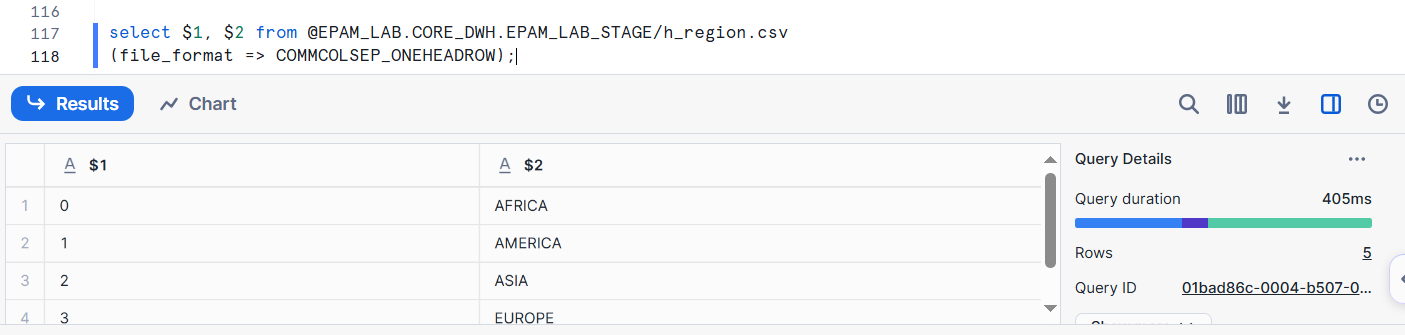


select $1, $2 from @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE/h\_region.csv;



select $1, $2 from @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE/h\_region.csv

(file\_format => COMMCOLSEP\_ONEHEADROW);



* **Create tables in CORE\_DWH schema.**

use schema core\_dwh;

drop table if exists region;

create table region

(

r\_regionkey NUMBER not null,

r\_name VARCHAR(25),

r\_comment VARCHAR(152)

);

drop table if exists nation;

create table nation

(

n\_nationkey NUMBER not null,

n\_name VARCHAR(27),

n\_regionkey NUMBER,

n\_comment VARCHAR(155)

);

drop table if exists supplier;

create table supplier

(

s\_suppkey NUMBER not null,

s\_name VARCHAR(25),

s\_address VARCHAR(40),

s\_nationkey NUMBER,

s\_phone VARCHAR(15),

s\_acctbal FLOAT,

s\_comment VARCHAR(101)

);

drop table if exists orders;

create table orders

(

o\_orderkey NUMBER not null,

o\_custkey NUMBER not null,

o\_orderstatus VARCHAR(1),

o\_totalprice FLOAT,

o\_orderdate DATE,

o\_prioritykey NUMBER(2),

o\_clerk VARCHAR(15),

o\_shippriority NUMBER(2),

o\_comment VARCHAR(79)

);

drop table if exists partsupp;

create table partsupp

(

ps\_partkey NUMBER not null,

ps\_suppkey NUMBER not null,

ps\_availqty NUMBER,

ps\_supplycost FLOAT not null,

ps\_comment VARCHAR(199)

);

drop table if exists part;

create table part

(

p\_partkey NUMBER not null,

p\_name VARCHAR(55),

p\_mnfgrkey NUMBER(5),

p\_brandkey NUMBER(5),

p\_pcktpkey NUMBER(5),

p\_size NUMBER,

p\_cntnrkey NUMBER(5),

p\_retailprice NUMBER,

p\_comment VARCHAR(23)

);

drop table if exists customer;

create table customer

(

c\_custkey NUMBER not null,

c\_name VARCHAR(25),

c\_address VARCHAR(40),

c\_nationkey NUMBER,

c\_phone VARCHAR(15),

c\_acctbal FLOAT,

c\_segmentkey NUMBER(5),

c\_comment VARCHAR(117)

);

drop table if exists lineitem;

create table lineitem

(

l\_orderkey NUMBER not null,

l\_partkey NUMBER not null,

l\_suppkey NUMBER not null,

l\_linenumber NUMBER not null,

l\_quantity NUMBER not null,

l\_extendedprice FLOAT not null,

l\_discount FLOAT not null,

l\_tax FLOAT not null,

l\_returnflag VARCHAR(1),

l\_linestatus VARCHAR(1),

l\_shipdate DATE,

l\_commitdate DATE,

l\_receiptdate DATE,

l\_shipinstruct VARCHAR(25),

l\_shipmode VARCHAR(10),

l\_comment VARCHAR(44)

);

drop table if exists ordpriority;

create table ordpriority

(

p\_prioritykey number(2) not null,

p\_priorityname varchar(80) not null

);

drop table if exists mktsegment;

create table mktsegment

(

m\_segmentkey number(5) not null,

m\_segmentname varchar(80) not null

);

drop table if exists brand;

create table brand

(

b\_brandkey number(5) not null,

b\_brandname varchar(80) not null

);

drop table if exists manufacture;

create table manufacture

(

m\_mnfgrkey number(5) not null,

m\_mnfgrname varchar(80) not null

);

drop table if exists container;

create table container

(

c\_cntnrkey number(5) not null,

c\_cntnrname varchar(80) not null

);

drop table if exists packagetype;

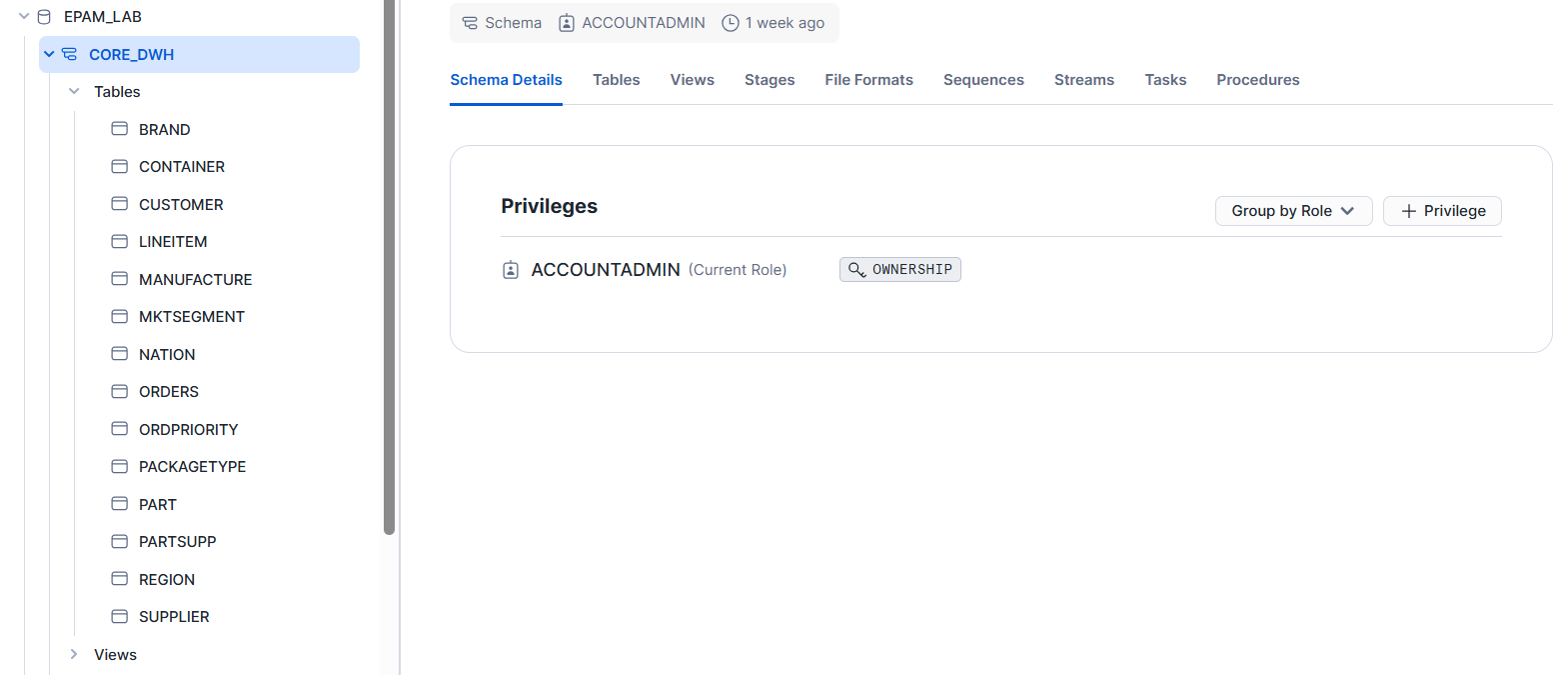
create table packagetype

(

p\_pcktpkey number(5) not null,

p\_pcktpname varchar(80) not null

);



* **Create Stage Views in CORE\_DWH schema.**

use database epam\_lab;

use schema core\_dwh;

create or replace view h\_mktsegment\_v as

select distinct trim($7) C\_MKTSEGMENT

from @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE/h\_customer.dsv

(file\_format => PIPECOLSEP\_ONEHEADROW )

order by C\_MKTSEGMENT;

create or replace view h\_customer\_v as

select C\_CUSTKEY, C\_NAME, C\_ADDRESS, C\_NATIONKEY, C\_PHONE, C\_ACCTBAL, M\_SEGMENTKEY, C\_COMMENT

from

(

select $1 C\_CUSTKEY, $2 C\_NAME, $3 C\_ADDRESS, $4 C\_NATIONKEY,

$5 C\_PHONE, replace($6, ',', '.') C\_ACCTBAL, trim($7) C\_MKTSEGMENT, $8 C\_COMMENT

from @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE/h\_customer.dsv(file\_format => PIPECOLSEP\_ONEHEADROW)

) r inner join mktsegment

on M\_SEGMENTNAME = C\_MKTSEGMENT

order by C\_CUSTKEY;

create or replace view h\_supplier\_v as

select $1 S\_SUPPKEY, $2 S\_NAME, $3 S\_ADDRESS, $4 S\_NATIONKEY,

$5 S\_PHONE, replace($6, ',', '.') S\_ACCTBAL, $7 S\_COMMENT

from @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE/h\_supplier.dsv

(file\_format => PIPECOLSEP\_ONEHEADROW );

create or replace view h\_order\_v as

select $1 O\_ORDERKEY, $2 O\_CUSTKEY, $3 O\_ORDERSTATUS, replace($4, ',', '.') O\_TOTALPRICE,

date($5, 'dd.mm.yy') O\_ORDERDATE, substring($6,1, position('-' in $6)-1) O\_PRIORITYKEY,

$7 O\_CLERK, $8 O\_SHIPPRIORITY, $9 O\_COMMENT

from @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE

( file\_format => PIPECOLSEP\_ONEHEADROW, pattern=>'h\_order.\*[.]dsv' );

create or replace view h\_ordpriority\_v as

select distinct substring($6,1, position('-' in $6)-1) O\_PRIORITYKEY,

substring($6,position('-' in $6)+1) O\_PRIORITYNAME

from @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE

( file\_format => PIPECOLSEP\_ONEHEADROW, pattern=>'h\_order.\*[.]dsv' );

create or replace view h\_lineitem\_v as

select $1 L\_ORDERKEY, $2 L\_PARTKEY, $3 L\_SUPPKEY, $4 L\_LINENUMBER, $5 L\_QUANTITY,

replace($6, ',', '.') L\_EXTENDEDPRICE, replace($7, ',', '.') L\_DISCOUNT,

replace($8, ',', '.') L\_TAX, $9 L\_RETURNFLAG, $10 L\_LINESTATUS, date($11, 'dd.mm.yy') L\_SHIPDATE,

date($12, 'dd.mm.yy') L\_COMMITDATE, date($13, 'dd.mm.yy') L\_RECEIPTDATE,

$14 L\_SHIPINSTRUCT, $15 L\_SHIPMODE, $16 L\_COMMENT

from @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE

( file\_format => PIPECOLSEP\_ONEHEADROW, pattern=>'h\_lineitem.\*[.]dsv' );

create or replace view h\_partsupp\_v as

select $1 PS\_PARTKEY, $2 PS\_SUPPKEY, $3 PS\_AVAILQTY,

replace($4, ',', '.') PS\_SUPPLYCOST, $5 PS\_COMMENT

from @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE

( file\_format => PIPECOLSEP\_ONEHEADROW, pattern=>'h\_partsupp.\*[.]dsv' );

create or replace view h\_mfrgr\_v as

select distinct trim($3) m\_mfrgrname

from @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE

( file\_format => PIPECOLSEP\_ONEHEADROW, pattern=>'h\_part.dsv' );

create or replace view h\_brand\_v as

select distinct trim($4) b\_brandname

from @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE

( file\_format => PIPECOLSEP\_ONEHEADROW, pattern=>'h\_part.dsv' );

create or replace view h\_container\_v as

select distinct trim($7) c\_cntrname

from @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE

( file\_format => PIPECOLSEP\_ONEHEADROW, pattern=>'h\_part.dsv' );

create or replace view h\_package\_v as

select distinct trim($5) p\_packname

from @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE

( file\_format => PIPECOLSEP\_ONEHEADROW, pattern=>'h\_part.dsv' );

create or replace view h\_part\_v as

select P\_PARTKEY, P\_NAME, M\_MNFGRKEY, B\_BRANDKEY, P\_PCKTPKEY, P\_SIZE,

C\_CNTNRKEY, P\_RETAILPRICE, P\_COMMENT

from

(

select $1 P\_PARTKEY, trim($2) P\_NAME, trim($3) P\_MFGR, trim($4) P\_BRAND, trim($5) P\_TYPE,

$6 P\_SIZE, trim($7) P\_CONTAINER, $8 P\_RETAILPRICE, $9 P\_COMMENT

from @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE

( file\_format => PIPECOLSEP\_ONEHEADROW, pattern=>'h\_part.dsv' )

)

inner join manufacture on m\_mnfgrname = P\_MFGR

inner join brand on b\_brandname = P\_BRAND

inner join packagetype on p\_pcktpname = P\_TYPE

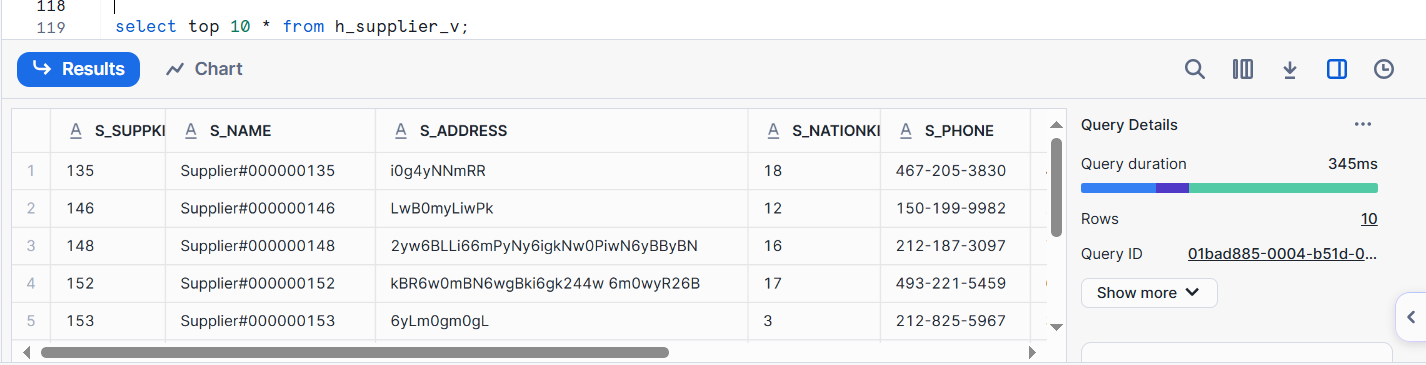
inner join container on c\_cntnrname = P\_CONTAINER;



select top 10 \* from h\_order\_v;



select top 10 \* from h\_supplier\_v;



* **Create load procedures in CORE\_DWH.**

create or replace procedure sp\_load\_csv (table\_name varchar, file\_name varchar)

returns integer not null

language sql

as

$$

begin

COPY INTO identifier(:table\_name)

FROM @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE

PATTERN = :file\_name

FILE\_FORMAT = ( format\_name = COMMCOLSEP\_ONEHEADROW TRIM\_SPACE = TRUE)

FORCE = TRUE;

return (select count(\*) from identifier(:table\_name));

end;

$$;

create or replace procedure sp\_load\_dsv (table\_name varchar, file\_name varchar)

returns integer not null

language sql

as

$$

begin

COPY INTO identifier(:table\_name)

FROM @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE

PATTERN = :file\_name

FILE\_FORMAT = ( format\_name = PIPECOLSEP\_ONEHEADROW TRIM\_SPACE = TRUE)

FORCE = TRUE;

return (select count(\*) from identifier(:table\_name));

end;

$$;

create or replace procedure sp\_load\_view (table\_name varchar, view\_name varchar)

returns integer not null

language sql

as

$$

begin

truncate table identifier(:table\_name);

INSERT INTO identifier(:table\_name)

SELECT \* FROM identifier(:view\_name);

return (select count(\*) from identifier(:table\_name));

end;

$$;

create or replace procedure sp\_load\_region()

returns integer not null

language sql

as

$$

BEGIN

truncate table epam\_lab.core\_dwh.region;

copy into region

from (select $1::int, trim($2::text), trim($3::text)

from @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE)

files = ('h\_region.csv')

file\_format = ( format\_name=COMMCOLSEP\_ONEHEADROW );

return SQLROWCOUNT;

END;

$$;

create or replace procedure sp\_load\_nation()

returns integer not null

language sql

as

$$

BEGIN

truncate table epam\_lab.core\_dwh.nation;

copy into nation

from (select $1::int, trim($2::text), $3::int, trim($4::text)

from @EPAM\_LAB.CORE\_DWH.EPAM\_LAB\_STAGE)

files = ('h\_nation.dsv')

file\_format = ( format\_name=PIPECOLSEP\_ONEHEADROW );

return SQLROWCOUNT;

END;

$$;

create or replace procedure sp\_load\_customer()

returns integer not null

language sql

as

$$

DECLARE

row\_count INTEGER DEFAULT 0;

BEGIN

truncate table epam\_lab.core\_dwh.mktsegment;

insert into epam\_lab.core\_dwh.mktsegment

select ROW\_NUMBER() OVER (order by c\_mktsegment), c\_mktsegment

from epam\_lab.core\_dwh.h\_mktsegment\_v;

call sp\_load\_view(

table\_name => 'customer',

view\_name => 'h\_customer\_v') into row\_count;

return row\_count;

END;

$$;

create or replace procedure sp\_load\_order()

returns integer not null

language sql

as

$$

DECLARE

row\_count INTEGER DEFAULT 0;

BEGIN

truncate table epam\_lab.core\_dwh.ordpriority;

insert into epam\_lab.core\_dwh.ordpriority

select \* from epam\_lab.core\_dwh.h\_ordpriority\_v;

call sp\_load\_view(

table\_name => 'orders',

view\_name => 'h\_order\_v') into row\_count;

return row\_count;

END;

$$;

create or replace procedure sp\_load\_part()

returns integer not null

language sql

as

$$

DECLARE

row\_count INTEGER DEFAULT 0;

BEGIN

truncate table epam\_lab.core\_dwh.brand;

insert into epam\_lab.core\_dwh.brand

select ROW\_NUMBER() OVER (order by b\_brandname), b\_brandname

from epam\_lab.core\_dwh.h\_brand\_v;

truncate table epam\_lab.core\_dwh.container;

insert into epam\_lab.core\_dwh.container

select ROW\_NUMBER() OVER (order by c\_cntrname), c\_cntrname

from epam\_lab.core\_dwh.h\_container\_v;

truncate table epam\_lab.core\_dwh.manufacture;

insert into epam\_lab.core\_dwh.manufacture

select ROW\_NUMBER() OVER (order by m\_mfrgrname), m\_mfrgrname

from epam\_lab.core\_dwh.h\_mfrgr\_v;

truncate table epam\_lab.core\_dwh.packagetype;

insert into epam\_lab.core\_dwh.packagetype

select ROW\_NUMBER() OVER (order by p\_packname), p\_packname

from epam\_lab.core\_dwh.h\_package\_v;

call sp\_load\_view(

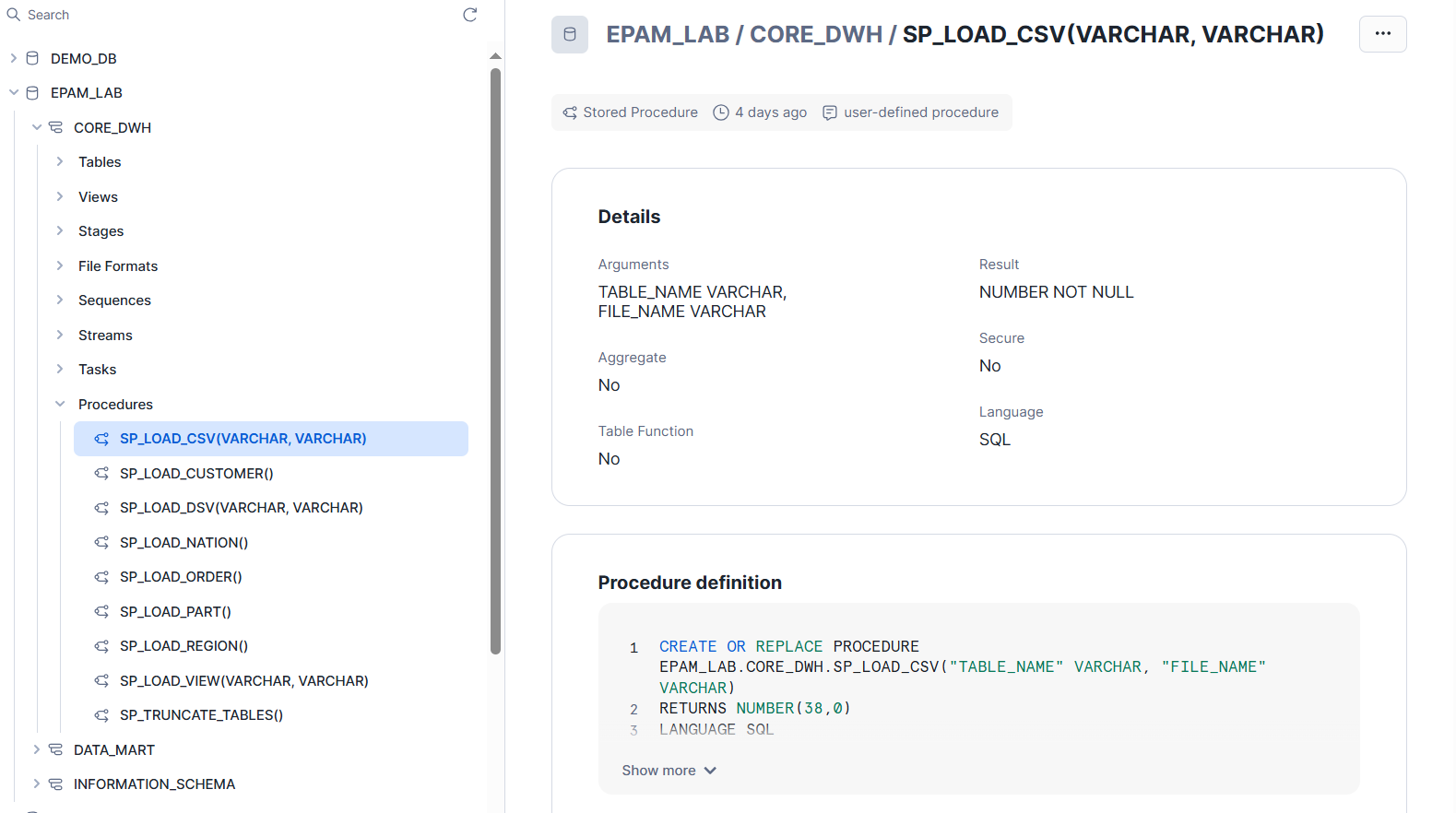
table\_name => 'part',

view\_name => 'h\_part\_v') into row\_count;

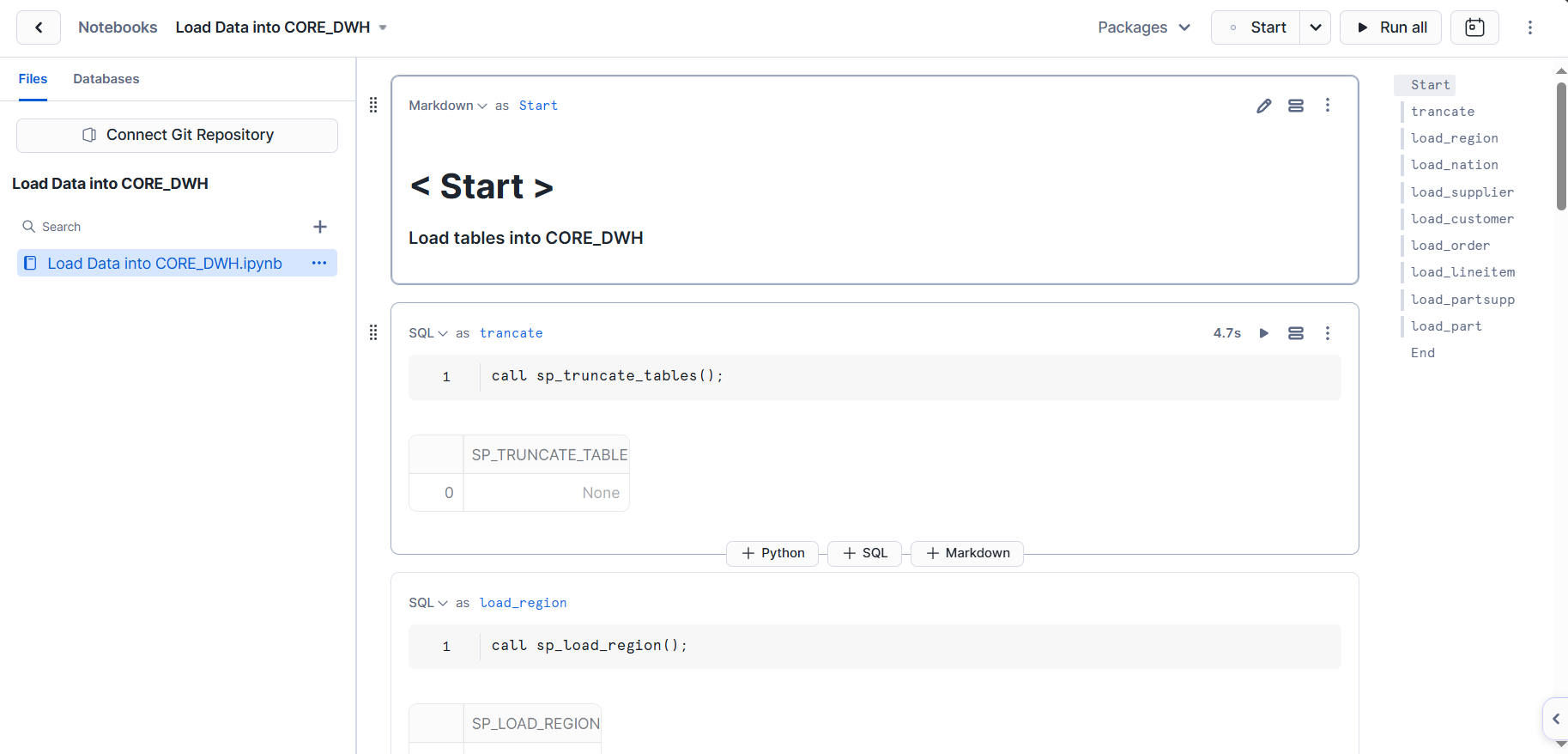
return row\_count;

END;

$$;



* **Data loading into CORE\_DWH.**



call sp\_truncate\_tables();

call sp\_load\_region();

call sp\_load\_nation();

call sp\_load\_view(

table\_name => 'supplier',

view\_name => 'h\_supplier\_v');

call sp\_load\_customer();

call sp\_load\_order();

call sp\_load\_view(

table\_name => 'lineitem',

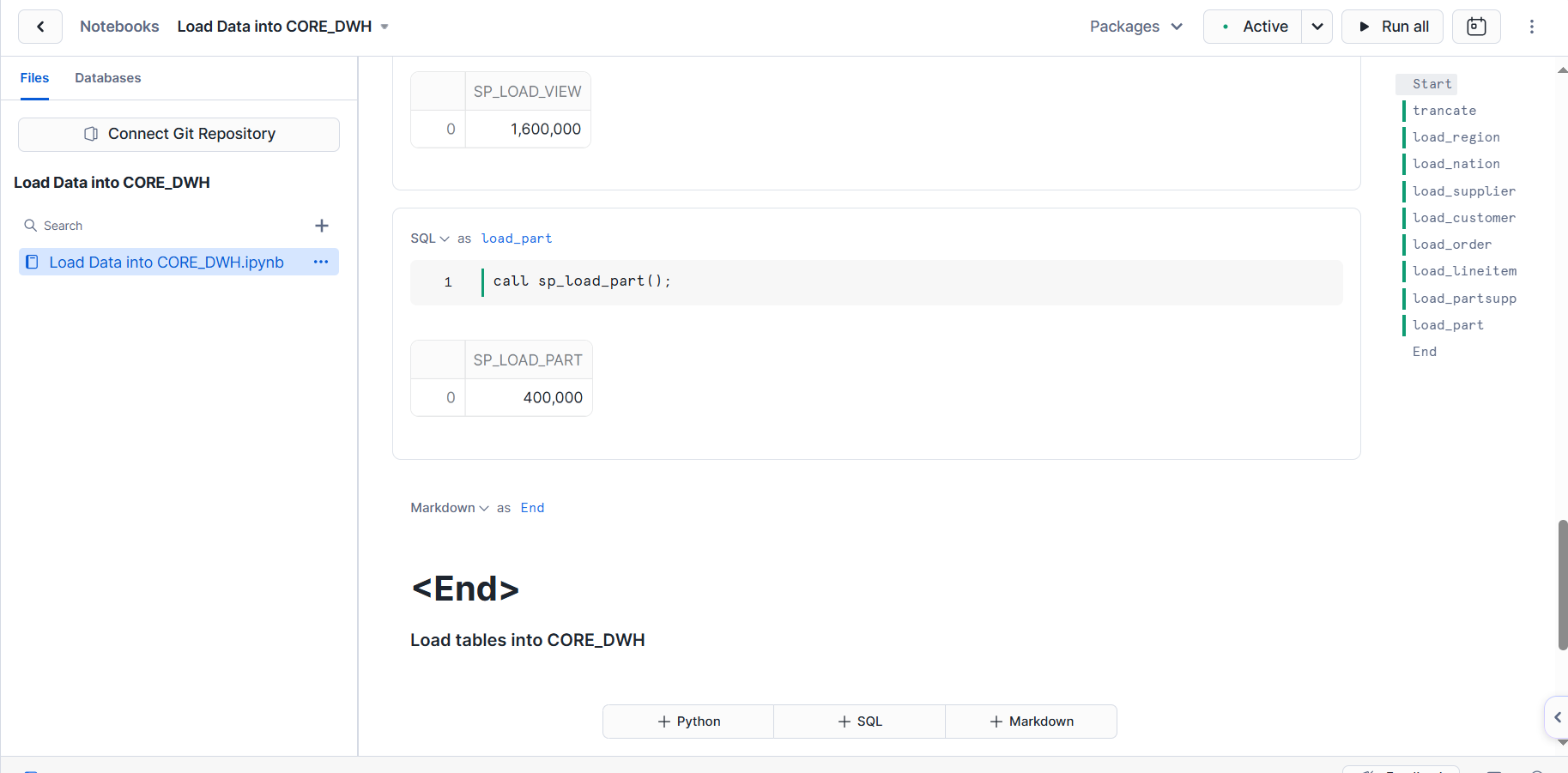
view\_name => 'h\_lineitem\_v');

call sp\_load\_view(

table\_name => 'partsupp',

view\_name => 'h\_partsupp\_v');

call sp\_load\_part();



* **Check loaded data**

select source, row\_num, expected,

case when row\_num = expected then 'OK' else 'Error' end passed

from

(

SELECT 'H\_LINEITEM' as source, count(\*) as row\_num, 11996782 expected

FROM epam\_lab.core\_dwh.lineitem

UNION ALL

SELECT 'H\_ORDER', count(\*), 3000000

FROM epam\_lab.core\_dwh.orders

UNION ALL

SELECT 'H\_PARTSUPP', count(\*), 1600000

FROM epam\_lab.core\_dwh.partsupp

UNION ALL

SELECT 'H\_PART', count(\*), 400000

FROM epam\_lab.core\_dwh.part

UNION ALL

SELECT 'H\_CUSTOMER', count(\*), 300000

FROM epam\_lab.core\_dwh.customer

UNION ALL

SELECT 'H\_SUPPLIER', count(\*), 20000

FROM epam\_lab.core\_dwh.supplier

UNION ALL

SELECT 'H\_NATION', count(\*), 25

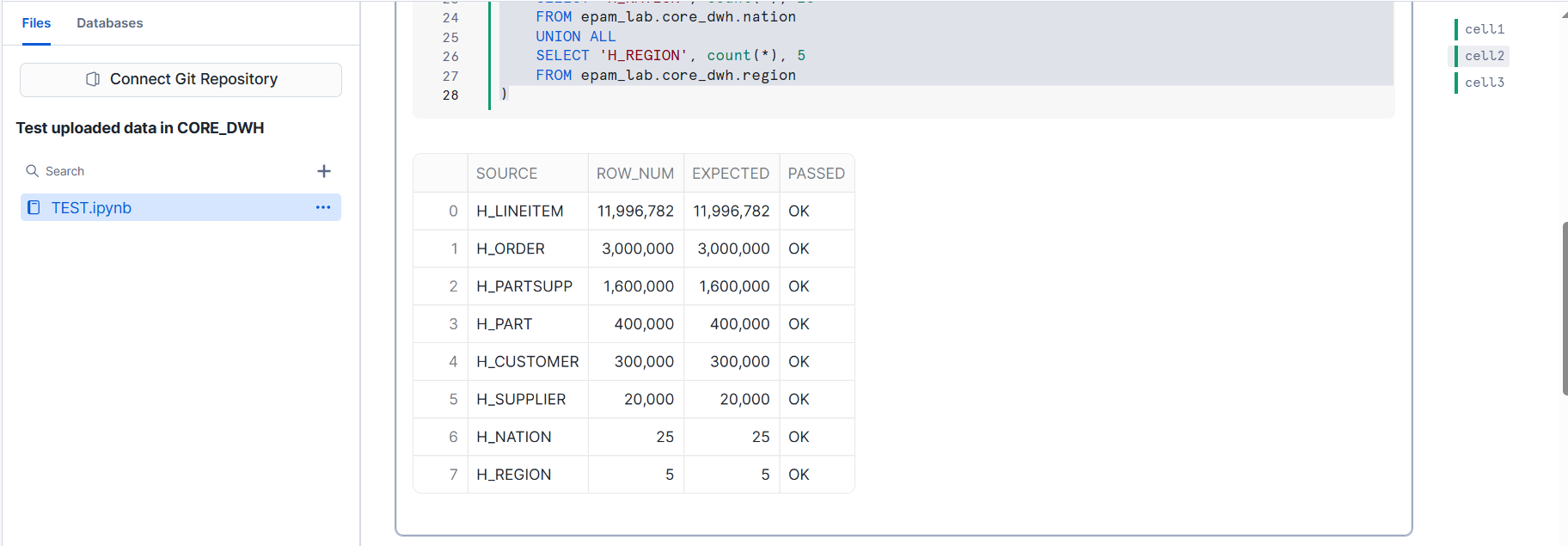
FROM epam\_lab.core\_dwh.nation

UNION ALL

SELECT 'H\_REGION', count(\*), 5

FROM epam\_lab.core\_dwh.region

);



A screenshot of a computer

AI-generated content may be incorrect.

* **Create DataMart tables**

create or replace TABLE EPAM\_LAB.DATA\_MART.DIM\_BRAND

(

B\_BRANDKEY NUMBER(5,0),

B\_BRANDNAME VARCHAR(80)

);

create or replace TABLE EPAM\_LAB.DATA\_MART.DIM\_CALENDAR

(

C\_CALNKEY VARCHAR(16777216),

C\_CALNDATE DATE,

C\_YEARNUM NUMBER(4,0),

C\_QUARTER NUMBER(2,0),

C\_MONTHNUM NUMBER(2,0),

C\_MONTHNAME VARCHAR(3),

C\_WEEKNUM NUMBER(2,0),

C\_DAYWEEK NUMBER(2,0),

C\_DAYNAME VARCHAR(3)

);

create or replace TABLE EPAM\_LAB.DATA\_MART.DIM\_CONTAINER

(

C\_CNTNRKEY NUMBER(5,0),

C\_CNTNRNAME VARCHAR(80)

);

create or replace TABLE EPAM\_LAB.DATA\_MART.DIM\_CUSTOMER

(

C\_CUSTKEY NUMBER(38,0),

C\_NAME VARCHAR(25),

C\_ADDRESS VARCHAR(40),

UPDATE\_TIMESTAMP TIMESTAMP\_LTZ(9) default CURRENT\_TIMESTAMP()::TIMESTAMP\_LTZ

);

create or replace TABLE EPAM\_LAB.DATA\_MART.DIM\_CUST\_UPD

(

C\_CUSTKEY NUMBER(38,0),

C\_NAME VARCHAR(25),

C\_ADDRESS VARCHAR(40),

START\_TIME TIMESTAMP\_NTZ(9),

END\_TIME TIMESTAMP\_NTZ(9),

CURRENT\_FLAG NUMBER(38,0)

);

create or replace TABLE EPAM\_LAB.DATA\_MART.DIM\_MKTSEGMENT

(

M\_SEGMENTKEY NUMBER(5,0),

M\_SEGMENTNAME VARCHAR(80)

);

create or replace TABLE EPAM\_LAB.DATA\_MART.DIM\_NATION

(

N\_NATIONKEY NUMBER(38,0),

N\_NAME VARCHAR(27),

N\_REGIONKEY NUMBER(38,0),

N\_COMMENT VARCHAR(155)

);

create or replace TABLE EPAM\_LAB.DATA\_MART.DIM\_ORDPRIORITY

(

P\_PRIORITYKEY NUMBER(2,0),

P\_PRIORITYNAME VARCHAR(80)  
);

create or replace TABLE EPAM\_LAB.DATA\_MART.DIM\_PACKAGE

(

P\_PCKTPKEY NUMBER(5,0),

P\_PCKTPNAME VARCHAR(80)

);

create or replace TABLE EPAM\_LAB.DATA\_MART.DIM\_REGION

(

R\_REGIONKEY NUMBER(38,0),

R\_NAME VARCHAR(25),

R\_COMMENT VARCHAR(152)

);

create or replace TABLE EPAM\_LAB.DATA\_MART.FACT\_ORDERS (

C\_CUSTKEY NUMBER(38,0),

C\_SEGMENTKEY NUMBER(5,0),

C\_NATIONKEY NUMBER(38,0),

C\_REGIONKEY NUMBER(38,0),

O\_ORDERKEY NUMBER(38,0),

O\_ORDDTKEY NUMBER(38,0),

O\_PRIORITYKEY NUMBER(2,0),

O\_SHIPPRIORITY NUMBER(2,0),

L\_SHPDTKEY NUMBER(38,0),

L\_SHIPMODE VARCHAR(10),

L\_RECDTKEY NUMBER(38,0),

L\_COMDTKEY NUMBER(38,0),

L\_EXTENDEDPRICE FLOAT,

L\_DISCOUNT FLOAT,

L\_QUANTITY NUMBER(38,0),

L\_RETURNFLAG VARCHAR(1),

L\_LINESTATUS VARCHAR(1),

S\_NATIONKEY NUMBER(38,0),

S\_REGIONKEY NUMBER(38,0),

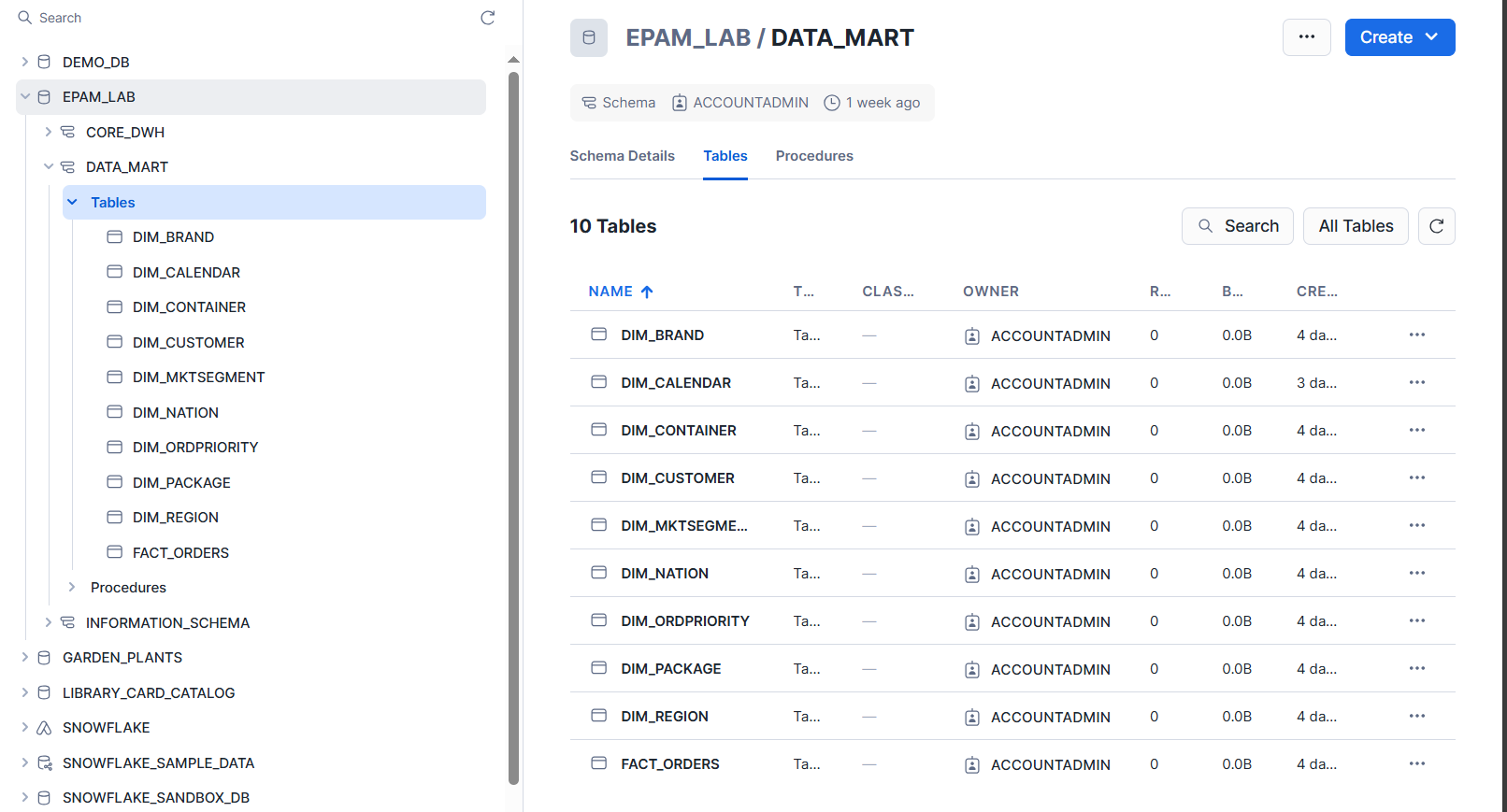
P\_BRANDKEY NUMBER(5,0),

P\_PCKTPKEY NUMBER(5,0),

P\_CNTNRKEY NUMBER(5,0),

P\_SIZE NUMBER(38,0)

);



* **Create Stream on table.**

create or replace stream CUST\_DELTA\_S on table DIM\_CUSTOMER;

--slow changing dimension table for Customer

create or replace TABLE EPAM\_LAB.DATA\_MART.DIM\_CUST\_UPD

(

C\_CUSTKEY NUMBER(38,0),

C\_NAME VARCHAR(25),

C\_ADDRESS VARCHAR(40),

start\_time TIMESTAMP\_NTZ,

end\_time TIMESTAMP\_NTZ,

current\_flag INT

);

--Slow Changing Dimension presentation for Customer changes

CREATE OR REPLACE VIEW cust\_change\_v AS

SELECT c\_custkey, c\_name, c\_address, start\_time, end\_time, current\_flag, dml\_type

FROM (

-- Logic for new insertions

SELECT c\_custkey, c\_name, c\_address, update\_timestamp AS start\_time,

LAG(update\_timestamp) OVER (PARTITION BY c\_custkey ORDER BY update\_timestamp DESC) AS end\_time\_raw,

CASE

WHEN end\_time\_raw IS NULL THEN '9999-12-31'::TIMESTAMP\_NTZ

ELSE end\_time\_raw

END AS end\_time,

CASE

WHEN end\_time\_raw IS NULL THEN 1

ELSE 0

END AS current\_flag,

'I' AS dml\_type

FROM cust\_delta\_s

WHERE metadata$action = 'INSERT' AND metadata$isupdate = 'FALSE'

UNION ALL

-- Logic for updates

SELECT c\_custkey, c\_name, c\_address, update\_timestamp AS start\_time,

LAG(update\_timestamp) OVER (PARTITION BY c\_custkey ORDER BY update\_timestamp DESC) AS end\_time\_raw,

CASE

WHEN end\_time\_raw IS NULL THEN '9999-12-31'::TIMESTAMP\_NTZ

ELSE end\_time\_raw

END AS end\_time,

CASE

WHEN end\_time\_raw IS NULL THEN 1

ELSE 0

END AS current\_flag,

CASE WHEN current\_flag = 1 then 'I' else 'U' end AS dml\_type

FROM cust\_delta\_s

WHERE metadata$isupdate = 'TRUE'

UNION ALL

-- Logic for deletions

SELECT c\_custkey, c\_name, c\_address, start\_time, CURRENT\_TIMESTAMP()::TIMESTAMP\_NTZ end\_time\_raw,

end\_time\_raw end\_time, null current\_flag, 'D' AS dml\_type

FROM dim\_cust\_upd h

WHERE h.c\_custkey IN (SELECT DISTINCT c\_custkey FROM cust\_delta\_s

WHERE metadata$action = 'DELETE'

AND metadata$isupdate = 'FALSE')

AND h.current\_flag = 1

);

CREATE OR REPLACE TASK dim\_cust\_upd\_scd

WAREHOUSE = compute\_wh

SCHEDULE = '5 minutes'

WHEN SYSTEM$STREAM\_HAS\_DATA('cust\_delta\_s')

AS

MERGE INTO dim\_cust\_upd h

USING cust\_change\_v cd ON h.c\_custkey = cd.c\_custkey

AND h.start\_time = cd.start\_time

WHEN MATCHED AND cd.dml\_type = 'U'

THEN UPDATE

SET h.end\_time = cd.end\_time,

h.current\_flag = 0

WHEN MATCHED AND cd.dml\_type = 'D'

THEN UPDATE

SET h.end\_time = cd.end\_time,

h.current\_flag = 0

WHEN NOT MATCHED AND cd.dml\_type = 'I'

THEN INSERT (c\_custkey, c\_name, c\_address, start\_time, end\_time, current\_flag)

VALUES (cd.c\_custkey, cd.c\_name, cd.c\_address, cd.start\_time, cd.end\_time,

cd.current\_flag);

--Check stream

insert into DIM\_CUSTOMER (C\_CUSTKEY, C\_NAME, C\_ADDRESS) values (11, 'Cust 11', 'addr');

insert into DIM\_CUSTOMER (C\_CUSTKEY, C\_NAME, C\_ADDRESS) values (12, 'Cust 12', 'addr');

insert into DIM\_CUSTOMER (C\_CUSTKEY, C\_NAME, C\_ADDRESS) values (13, 'Cust 13', 'addr');

select \* from cust\_delta\_s;

A screenshot of a computer

AI-generated content may be incorrect.

select \* from cust\_change\_v;

A screenshot of a computer

AI-generated content may be incorrect.

ALTER TASK IF EXISTS dim\_cust\_upd\_scd RESUME;

A screenshot of a computer

AI-generated content may be incorrect.

select \* from dim\_cust\_upd

order by c\_custkey;

A screenshot of a computer

AI-generated content may be incorrect.

select top 10 \* from DIM\_CUSTOMER;

A screenshot of a computer

AI-generated content may be incorrect.

update dim\_customer

set c\_address = 'new address is 11', update\_timestamp = CURRENT\_TIMESTAMP()::TIMESTAMP\_LTZ

where c\_custkey = 11;

delete from dim\_customer

where c\_custkey = 12;

select \* from cust\_delta\_s;

A screenshot of a computer screen

AI-generated content may be incorrect.

select \* from cust\_change\_v;

A screenshot of a computer

AI-generated content may be incorrect.

select \* from dim\_cust\_upd order by c\_custkey;

A screenshot of a computer

AI-generated content may be incorrect.

truncate table dim\_customer;

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* **Create DataMart loading procedures.**

use schema data\_mart;

create or replace procedure sp\_truncate\_dtmart()

returns integer not null

language sql

as

$$

begin

TRUNCATE TABLE EPAM\_LAB.DATA\_MART.DIM\_NATION;

TRUNCATE TABLE EPAM\_LAB.DATA\_MART.DIM\_REGION;

TRUNCATE TABLE EPAM\_LAB.DATA\_MART.DIM\_MKTSEGMENT;;

TRUNCATE TABLE EPAM\_LAB.DATA\_MART.DIM\_CALENDAR;

TRUNCATE TABLE EPAM\_LAB.DATA\_MART.DIM\_CUSTOMER;

TRUNCATE TABLE EPAM\_LAB.DATA\_MART.DIM\_PACKAGE;

TRUNCATE TABLE EPAM\_LAB.DATA\_MART.DIM\_CONTAINER;

TRUNCATE TABLE EPAM\_LAB.DATA\_MART.DIM\_ORDPRIORITY;

TRUNCATE TABLE EPAM\_LAB.DATA\_MART.DIM\_BRAND;

TRUNCATE TABLE EPAM\_LAB.DATA\_MART.FACT\_ORDERS;

end;

$$;

create or replace procedure sp\_load\_calendar()

returns integer not null

language sql

as

$$

begin

insert into dim\_calendar

with period as

(

select min(iff(l\_orderkey is null, o\_orderdate,

least(o\_orderdate, l\_shipdate, l\_receiptdate))) start\_date,

max(iff(l\_orderkey is null, o\_orderdate,

greatest(o\_orderdate, l\_shipdate, l\_receiptdate))) end\_date

from epam\_lab.core\_dwh.orders

left join epam\_lab.core\_dwh.lineitem

on l\_orderkey = o\_orderkey

)

select

to\_char(date(start\_date) + value::int, 'yyyymmdd') c\_calnkey,

date(start\_date) + value::int as c\_calndate,

year(c\_calndate) c\_yearnum,

quarter(c\_calndate) c\_quarter,

month(c\_calndate) c\_monthnum,

monthname(c\_calndate) c\_monthname,

week(c\_calndate) c\_weeknum,

day(c\_calndate) c\_dayweek,

dayname(c\_calndate) c\_dayname

from period,

table(flatten(array\_generate\_range(0, datediff('day', start\_date, end\_date) + 1)))

order by c\_calndate;

return SQLROWCOUNT;

end;

$$;

create or replace procedure sp\_load\_dim()

returns integer not null

language sql

as

$$

declare

row\_count INTEGER DEFAULT 0;

begin

insert into dim\_mktsegment

select \* from epam\_lab.core\_dwh.mktsegment;

row\_count := row\_count + SQLROWCOUNT;

insert into dim\_brand

select \* from epam\_lab.core\_dwh.brand;

row\_count := row\_count + SQLROWCOUNT;

insert into dim\_package

select \* from epam\_lab.core\_dwh.packagetype;

row\_count := row\_count + SQLROWCOUNT;

insert into dim\_container

select \* from epam\_lab.core\_dwh.container;

row\_count := row\_count + SQLROWCOUNT;

insert into dim\_nation

select \* from epam\_lab.core\_dwh.nation;

row\_count := row\_count + SQLROWCOUNT;

insert into dim\_region

select \* from epam\_lab.core\_dwh.region;

row\_count := row\_count + SQLROWCOUNT;

insert into dim\_ordpriority

select \* from epam\_lab.core\_dwh.ordpriority;

row\_count := row\_count + SQLROWCOUNT;

insert into dim\_customer (c\_custkey, c\_name, c\_address)

select c\_custkey, c\_name, c\_address

from epam\_lab.core\_dwh.customer;

row\_count := row\_count + SQLROWCOUNT;

return row\_count;

end;

$$;

create or replace procedure sp\_load\_facts()

returns integer not null

language sql

as

$$

begin

insert into epam\_lab.data\_mart.fact\_orders

select

c\_custkey,

c\_segmentkey,

c\_nationkey,

r1.r\_regionkey c\_regionkey,

o\_orderkey,

--o\_orderdate,

to\_char(o\_orderdate, 'yyyymmdd')::int o\_orddtkey,

o\_prioritykey,

o\_shippriority,

--l\_shipdate,

to\_char(l\_shipdate , 'yyyymmdd')::int l\_shpdtkey,

l\_shipmode,

--l\_receiptdate,

to\_char(l\_receiptdate , 'yyyymmdd')::int l\_recdtkey,

to\_char(l\_commitdate, 'yyyymmdd')::int l\_comdtkey,

l\_extendedprice,

l\_discount,

l\_quantity,

l\_returnflag,

l\_linestatus,

s\_nationkey,

r2.r\_regionkey s\_regionkey,

p\_brandkey,

p\_pcktpkey,

p\_cntnrkey,

p\_size

from

epam\_lab.core\_dwh.customer,

epam\_lab.core\_dwh.orders,

epam\_lab.core\_dwh.lineitem,

epam\_lab.core\_dwh.supplier,

epam\_lab.core\_dwh.part,

epam\_lab.core\_dwh.nation n1,

epam\_lab.core\_dwh.nation n2,

epam\_lab.core\_dwh.region r1,

epam\_lab.core\_dwh.region r2

where

c\_custkey = o\_custkey

and l\_orderkey = o\_orderkey

and l\_suppkey = s\_suppkey

and p\_partkey = l\_partkey

and c\_nationkey = n1.n\_nationkey

and n1.n\_regionkey = r1.r\_regionkey

and s\_nationkey = n2.n\_nationkey

and n2.n\_regionkey = r2.r\_regionkey;

return SQLROWCOUNT;

end;

$$;

A screenshot of a computer

AI-generated content may be incorrect.

* **Datamart Loading.**

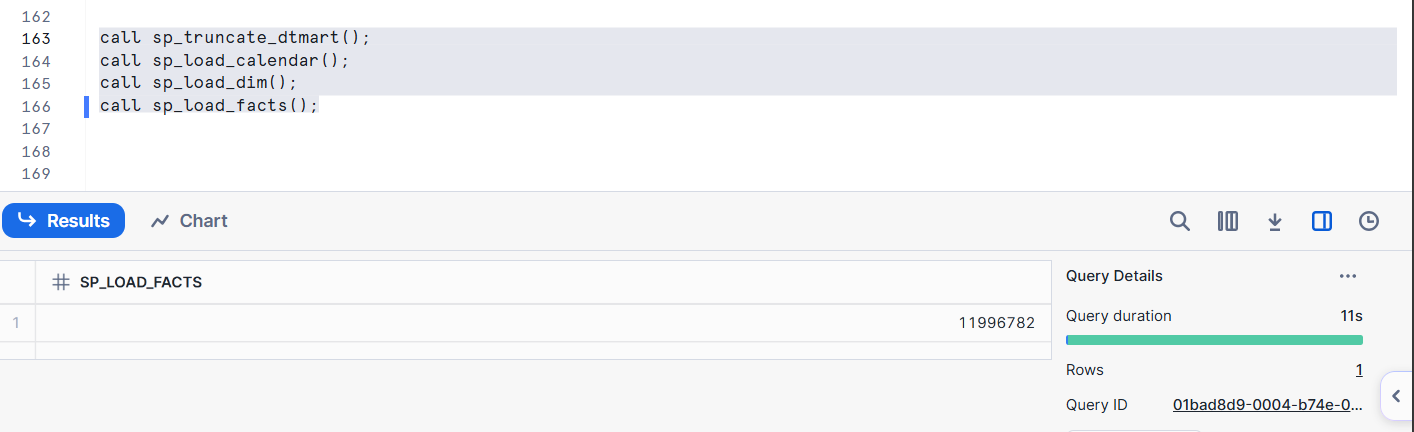
use schema data\_mart;

call sp\_truncate\_dtmart();

call sp\_load\_calendar();

call sp\_load\_dim();

call sp\_load\_facts();



* **Create loading/truncate tasks.**

**--For CORE\_DWH**

use schema core\_dwh;

CREATE OR REPLACE TASK load\_epam\_lab\_core\_dw

USER\_TASK\_MANAGED\_INITIAL\_WAREHOUSE\_SIZE = 'XSMALL'

SCHEDULE = 'USING CRON 0 11 \* \* \* Europe/London'

AS

begin

call sp\_truncate\_tables();

call sp\_load\_region();

call sp\_load\_nation();

call sp\_load\_view(

table\_name => 'supplier',

view\_name => 'h\_supplier\_v');

call sp\_load\_customer();

call sp\_load\_order();

call sp\_load\_view(

table\_name => 'lineitem',

view\_name => 'h\_lineitem\_v');

call sp\_load\_view(

table\_name => 'partsupp',

view\_name => 'h\_partsupp\_v');

call sp\_load\_part();

end;

CREATE OR REPLACE TASK clear\_epam\_lab\_stage

USER\_TASK\_MANAGED\_INITIAL\_WAREHOUSE\_SIZE = 'XSMALL'

SCHEDULE = 'USING CRON 0 20 \* \* \* Europe/London'

AS

remove @EPAM\_LAB\_STAGE;

CREATE OR REPLACE TASK clear\_epam\_lab\_tables

USER\_TASK\_MANAGED\_INITIAL\_WAREHOUSE\_SIZE = 'XSMALL'

SCHEDULE = 'USING CRON 0 21 \* \* \* Europe/London'

AS

call sp\_truncate\_tables();

A screenshot of a computer

AI-generated content may be incorrect.

**--For DATA\_MART**

use schema data\_mart;

CREATE OR REPLACE TASK load\_epam\_lab\_data\_mart

USER\_TASK\_MANAGED\_INITIAL\_WAREHOUSE\_SIZE = 'XSMALL'

SCHEDULE = 'USING CRON 15 11 \* \* \* Europe/London'

AS

begin

call sp\_load\_calendar();

call sp\_load\_dim();

call sp\_load\_facts();

end;

CREATE OR REPLACE TASK clear\_epam\_lab\_dtmart

USER\_TASK\_MANAGED\_INITIAL\_WAREHOUSE\_SIZE = 'XSMALL'

SCHEDULE = 'USING CRON 0 21 \* \* \* Europe/London'

AS

call sp\_truncate\_dtmart();

A screenshot of a computer

AI-generated content may be incorrect.

## Snowflake & 3rd party tools

No ability to install third party tools in LSEG laptop! Prohibited by LSEG Policy.

## Snowflake SQL

* **Create several warehouses of different sizes and compare their performance**

A screenshot of a computer

AI-generated content may be incorrect.

use database snowflake\_sample\_data;

use schema tpch\_sf100;

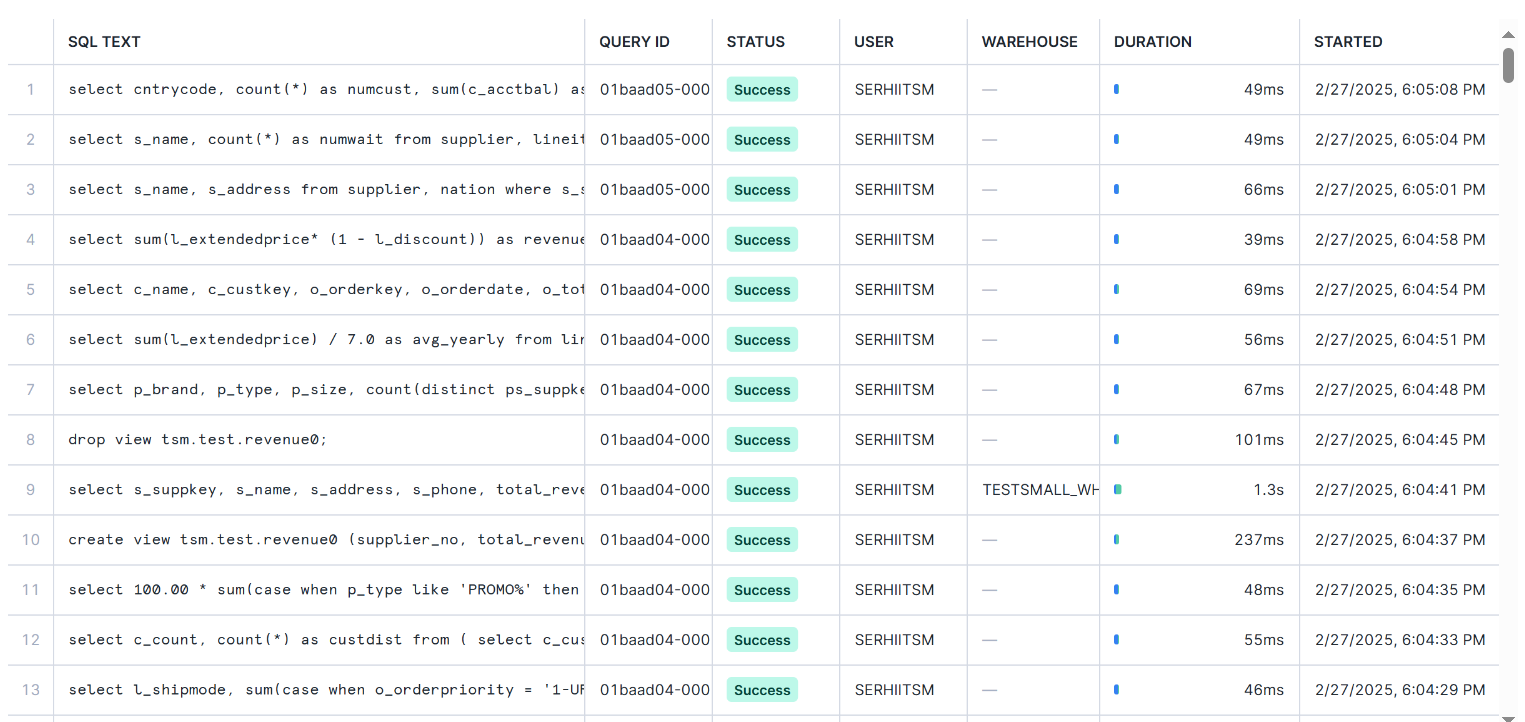
use warehouse compute\_wh;



use database snowflake\_sample\_data;

use schema tpch\_sf100;

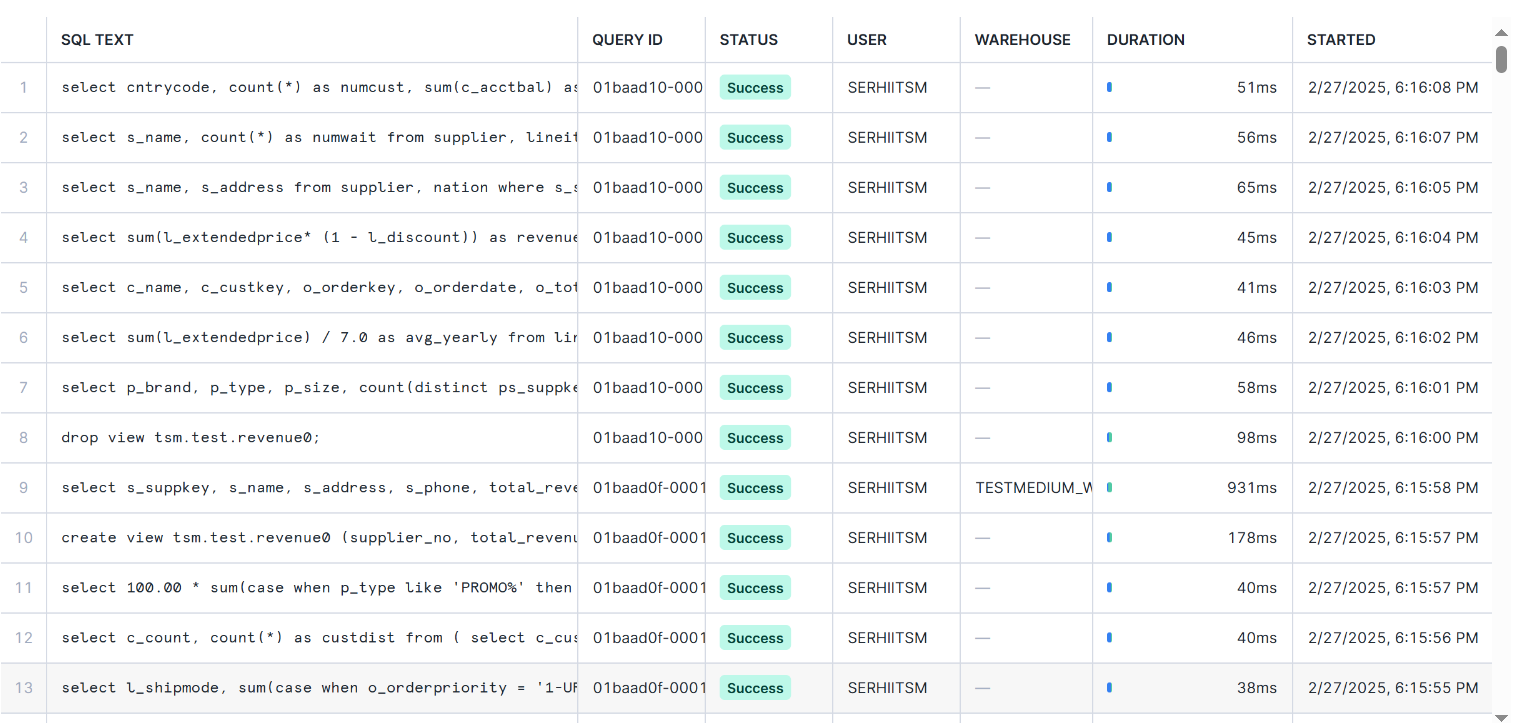
use warehouse testsmall\_wh;



use database snowflake\_sample\_data;

use schema tpch\_sf100;

use warehouse testmedium\_wh;



* **Test how Snowflake leverages different types of cash.**

--Sql query for testing

select cntrycode,

count(\*) as numcust,

sum(c\_acctbal) as totacctbal

from ( select substring(c\_phone, 1, 2) as cntrycode,

c\_acctbal

from customer

where substring(c\_phone, 1, 2) in

('14', '21', '24', '33', '28', '17', '10')

and c\_acctbal > (

select avg(c\_acctbal)

from customer

where c\_acctbal > 0.00

and substring(c\_phone, 1, 2) in

('14', '21', '24', '33', '28', '17', '10')

)

and not exists (

select \* from orders

where o\_custkey = c\_custkey)

) as custsale

group by cntrycode

order by cntrycode;

ALTER SESSION SET USE\_CACHED\_RESULT = FALSE;

ALTER WAREHOUSE COMPUTE\_WH SUSPEND;

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

ALTER SESSION SET USE\_CACHED\_RESULT = TRUE;

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

ALTER SESSION SET USE\_CACHED\_RESULT = FALSE;

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* **Rewrite a couple of queries to execute on the Start Schema data model and compare performance (3NF vs Star Schema).**

**--Query 1 in CORE\_DWH**

select l\_orderkey odrerkey,

sum(l\_extendedprice \* (1 - l\_discount)) as revenue,

o\_orderdate orderdate,

o\_shippriority shippriority

from epam\_lab.core\_dwh.customer,

epam\_lab.core\_dwh.orders,

epam\_lab.core\_dwh.lineitem,

epam\_lab.core\_dwh.mktsegment

where c\_custkey = o\_custkey

and l\_orderkey = o\_orderkey

and m\_segmentkey = c\_segmentkey

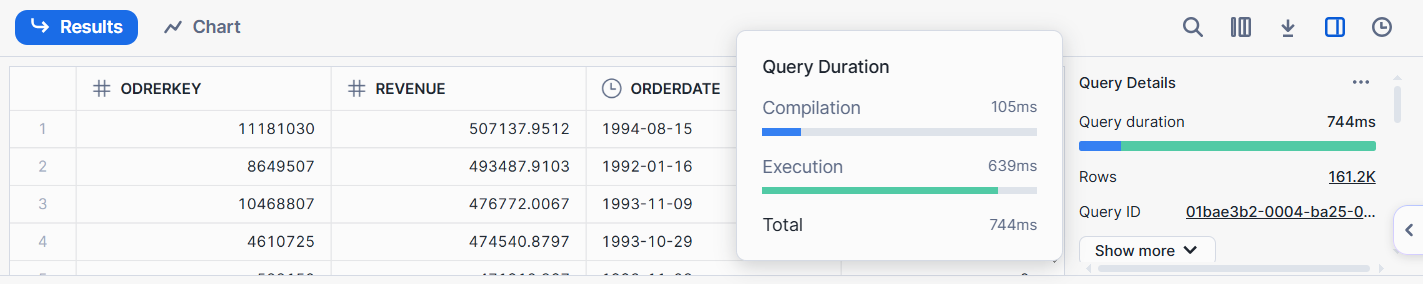
and o\_orderdate < date '1995-03-01'

and l\_shipdate > date '1995-03-01'

and m\_segmentname = 'AUTOMOBILE'

group by l\_orderkey, o\_orderdate, o\_shippriority

order by revenue desc, orderdate;



**--Query 1 in DATA\_MART**

select o\_orderkey l\_orderkey,

sum(l\_extendedprice \* (1 - l\_discount)) as revenue,

c1.c\_calndate o\_orderdate,

o\_shippriority

from fact\_orders f,

dim\_calendar c1,

dim\_mktsegment

where o\_orddtkey = c1.c\_calnkey

and o\_orddtkey < 19950301

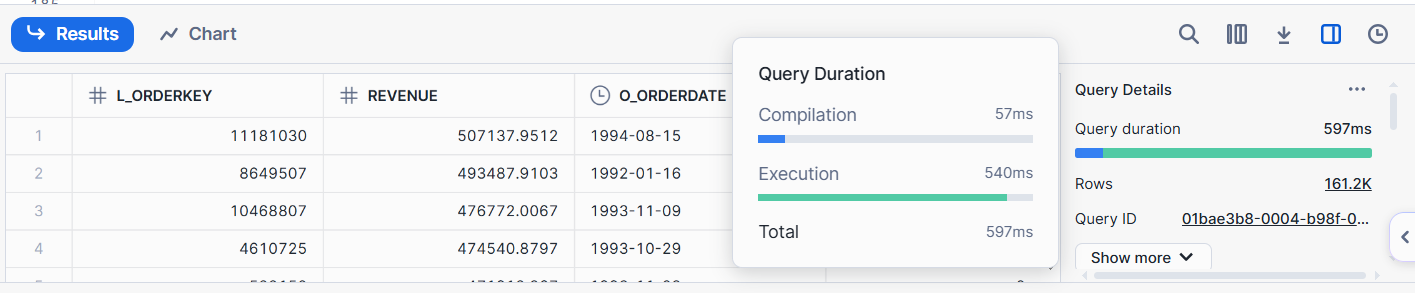
and l\_shpdtkey > 19950301

and m\_segmentkey = c\_segmentkey

and m\_segmentname = 'AUTOMOBILE'

group by l\_orderkey, o\_orderdate, o\_shippriority

order by revenue desc, o\_orderdate;



**--Query 2 in CORE\_DWH**

select p\_priorityname o\_orderpriority,

count(\*) as order\_count

from epam\_lab.core\_dwh.orders,

epam\_lab.core\_dwh.ordpriority

where o\_orderdate >= date '1993-10-01'

and p\_prioritykey = o\_prioritykey

and o\_orderdate < add\_months(date '1993-10-01',3)

and exists (

select \* from epam\_lab.core\_dwh.lineitem

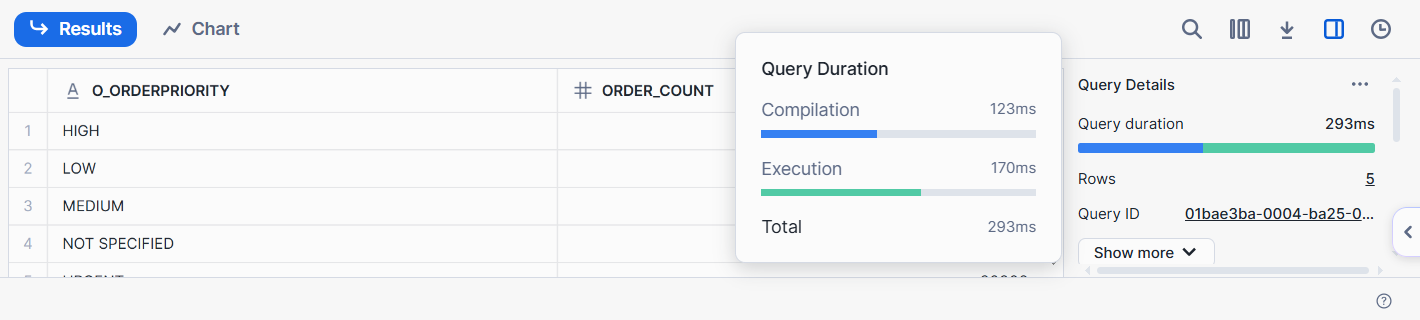
where l\_orderkey = o\_orderkey

and l\_commitdate < l\_receiptdate

)

group by o\_orderpriority

order by o\_orderpriority;



**--Query 2 in DATA\_MART**

select p\_priorityname o\_orderpriority,

count(\*) as order\_count

from fact\_orders, dim\_calendar, dim\_ordpriority

where o\_orddtkey = c\_calnkey

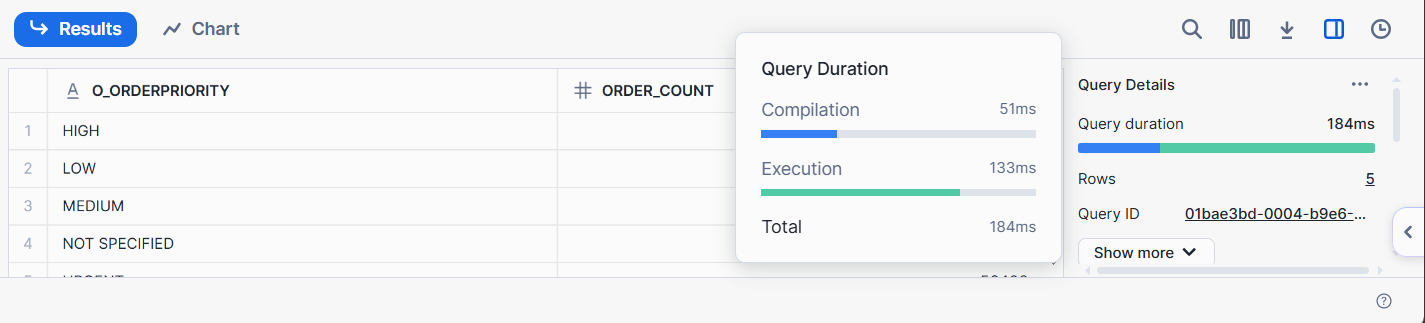
and c\_yearnum = 1993 and c\_quarter = 4

and l\_comdtkey < l\_recdtkey

and p\_prioritykey = o\_prioritykey

group by o\_orderpriority

order by o\_orderpriority;



**--Query 3 in CORE\_DWH**

select n\_name,

sum(l\_extendedprice \* (1 - l\_discount)) as revenue

from epam\_lab.core\_dwh.customer,

epam\_lab.core\_dwh.orders,

epam\_lab.core\_dwh.lineitem,

epam\_lab.core\_dwh.supplier,

epam\_lab.core\_dwh.nation,

epam\_lab.core\_dwh.region

where c\_custkey = o\_custkey

and l\_orderkey = o\_orderkey

and o\_orderdate >= date '1997-01-01'

and o\_orderdate < add\_months(date '1997-01-01',12)

and l\_suppkey = s\_suppkey

and c\_nationkey = s\_nationkey

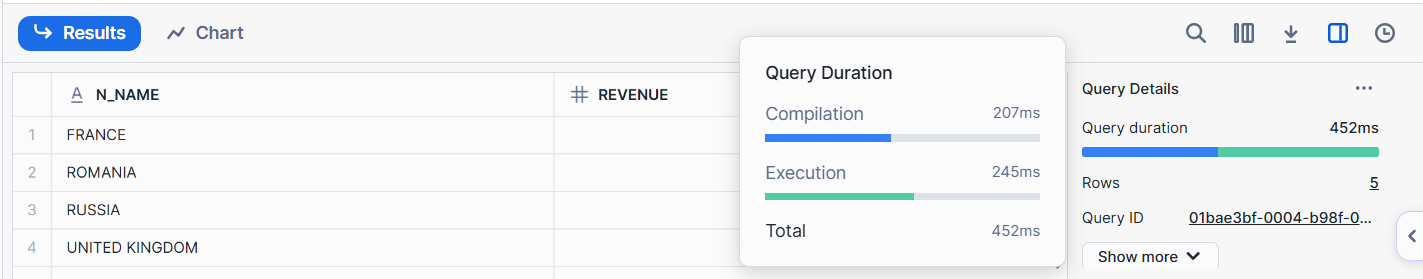
and s\_nationkey = n\_nationkey

and n\_regionkey = r\_regionkey

and r\_name = 'EUROPE'

group by n\_name

order by revenue desc;



**--Query 3 in DATA\_MART**

select n\_name,

sum(l\_extendedprice \* (1 - l\_discount)) as revenue

from fact\_orders, dim\_calendar, dim\_region, dim\_nation

where o\_orddtkey = c\_calnkey

and r\_regionkey = s\_regionkey

and c\_nationkey = s\_nationkey

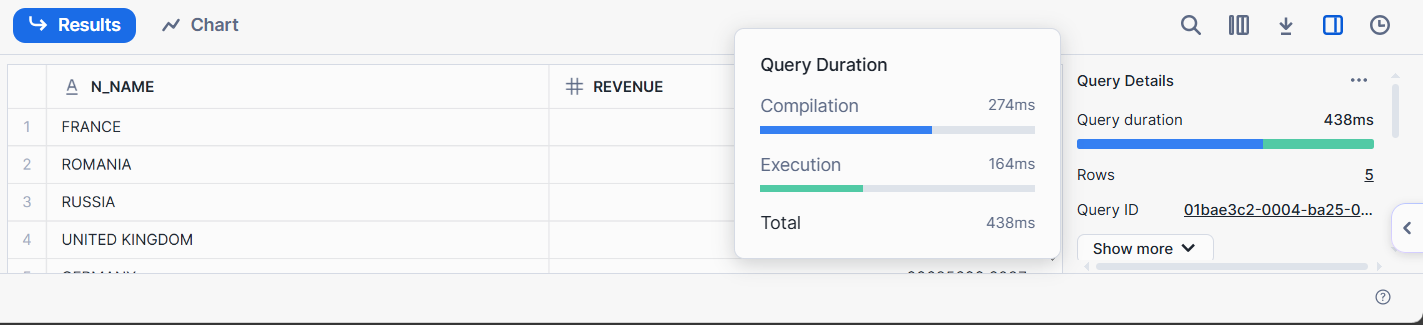
and s\_nationkey = n\_nationkey

and r\_name = 'EUROPE'

and c\_yearnum = 1997

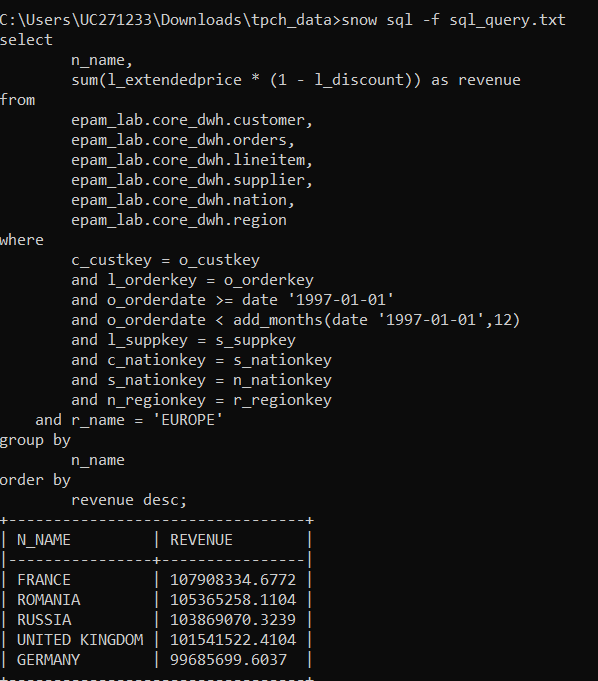
group by n\_name

order by revenue desc;



* **Execute queries using SnowSQL (CLI Client).**

tpch\_data>snow sql -f sql\_query.txt



tpch\_data>snow sql -q "select c\_custkey, c\_name, c\_address, c\_phone from customer where c\_custkey between 1 and 20 limit 10;"

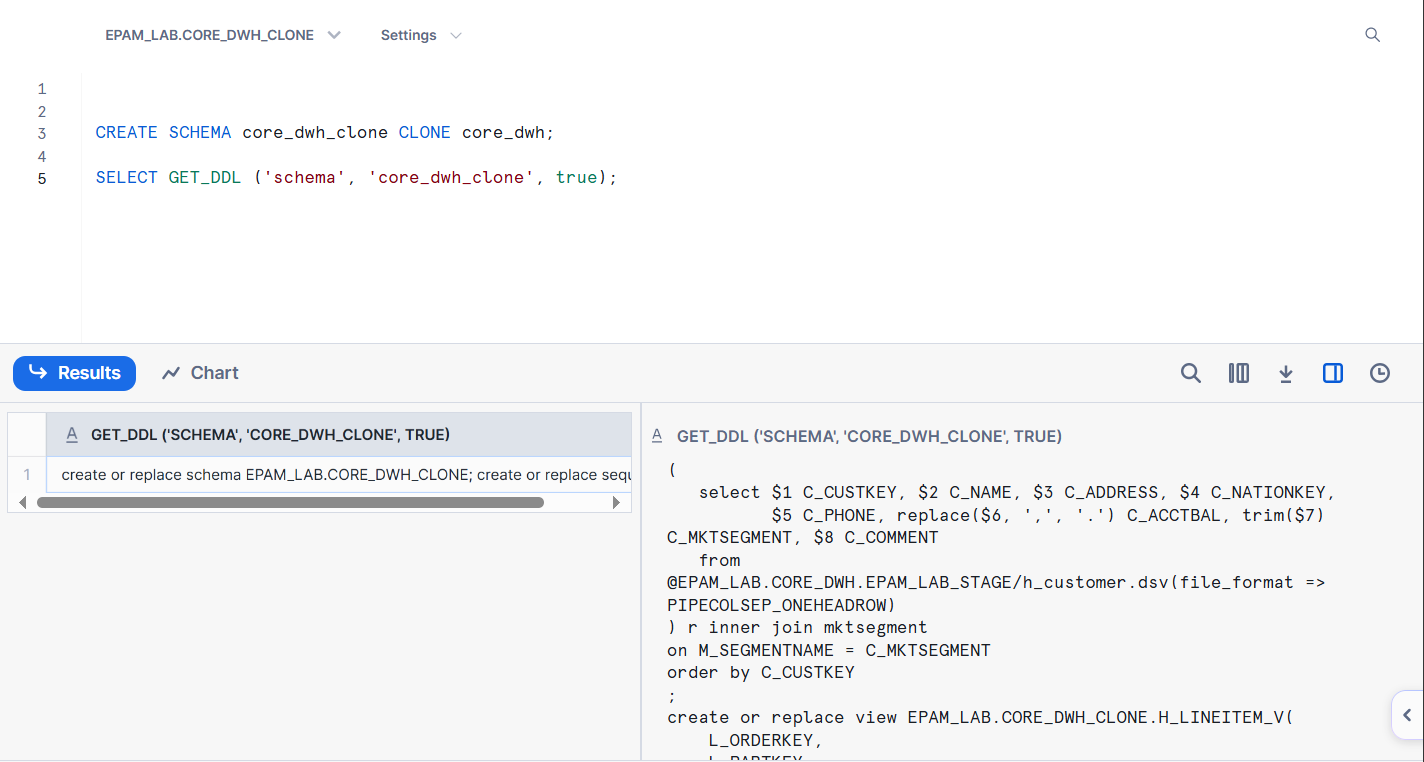


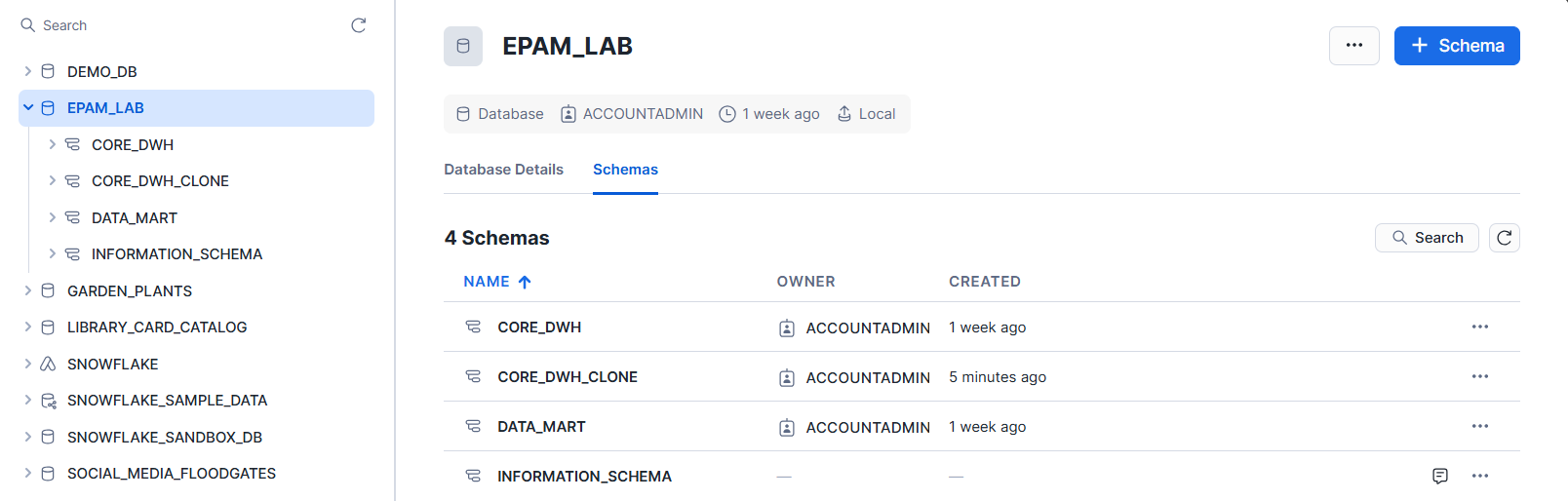
## Other Snowflake features

* **Object Cloning**

CREATE SCHEMA core\_dwh\_clone CLONE core\_dwh;

SELECT GET\_DDL ('schema', 'core\_dwh\_clone', true);





select \* from epam\_lab.core\_dwh.brand



select \* from epam\_lab.core\_dwh\_clone.brand

A screenshot of a computer

AI-generated content may be incorrect.

--Chang data in clone schema

insert into epam\_lab.core\_dwh\_clone.brand

values (6, 'Brand#66');

update epam\_lab.core\_dwh\_clone.brand

set b\_brandname = 'Brand#11'

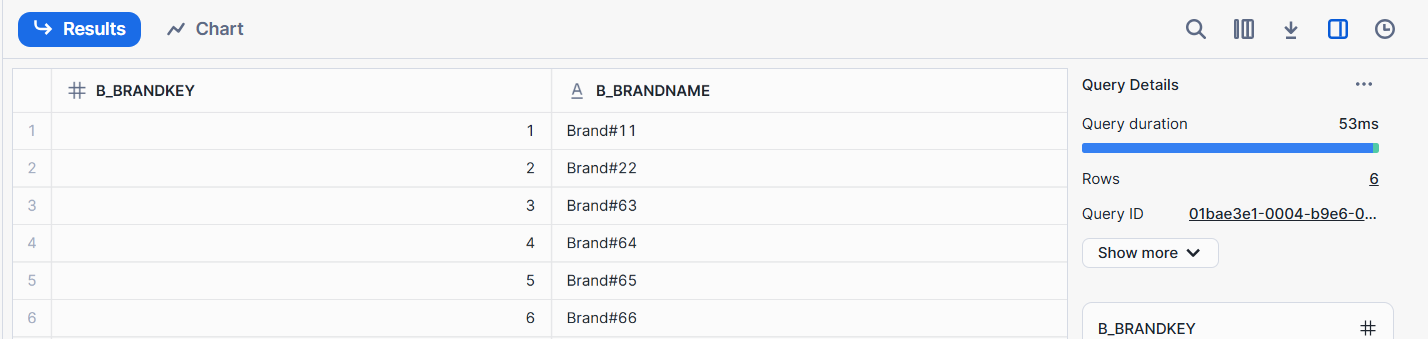
where b\_brandkey = 1;

update epam\_lab.core\_dwh\_clone.brand

set b\_brandname = 'Brand#22'

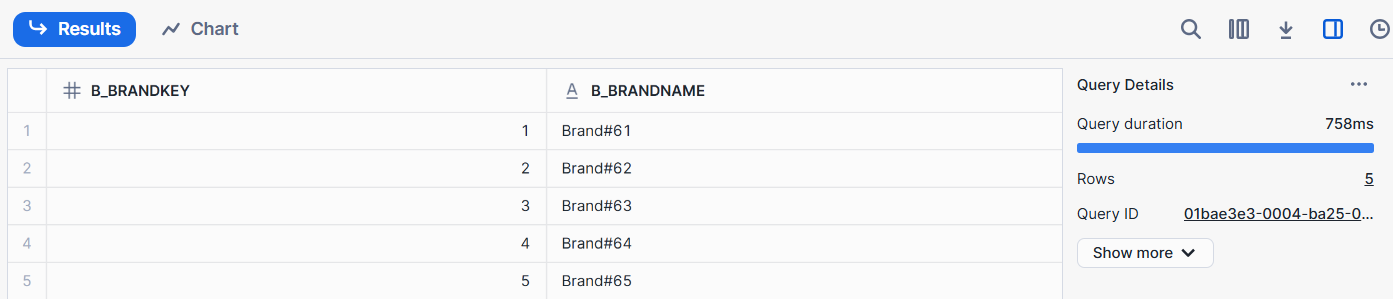
where b\_brandkey = 2;

select \* from epam\_lab.core\_dwh\_clone.brand



--Data is not changed in source.

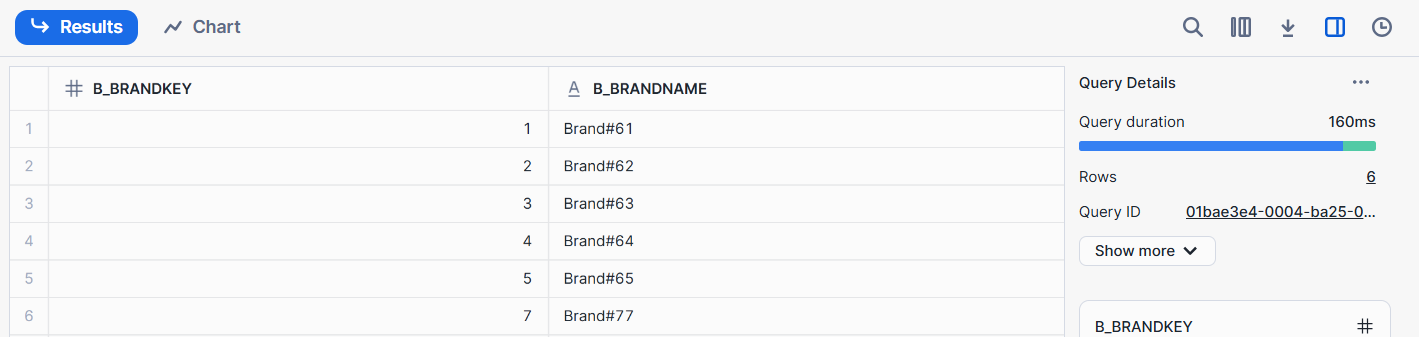
select \* from epam\_lab.core\_dwh.brand;



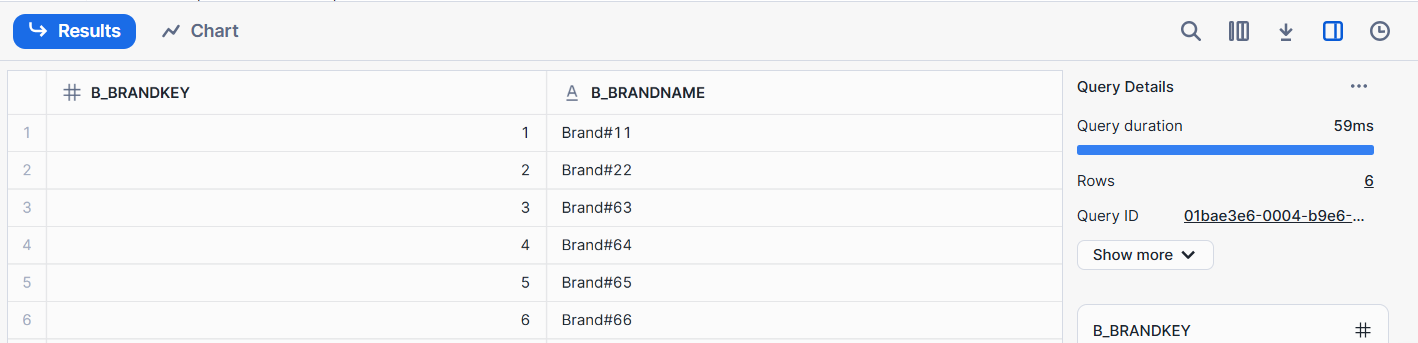
--Add a new record in source.

insert into epam\_lab.core\_dwh.brand

values (7, 'Brand#77');



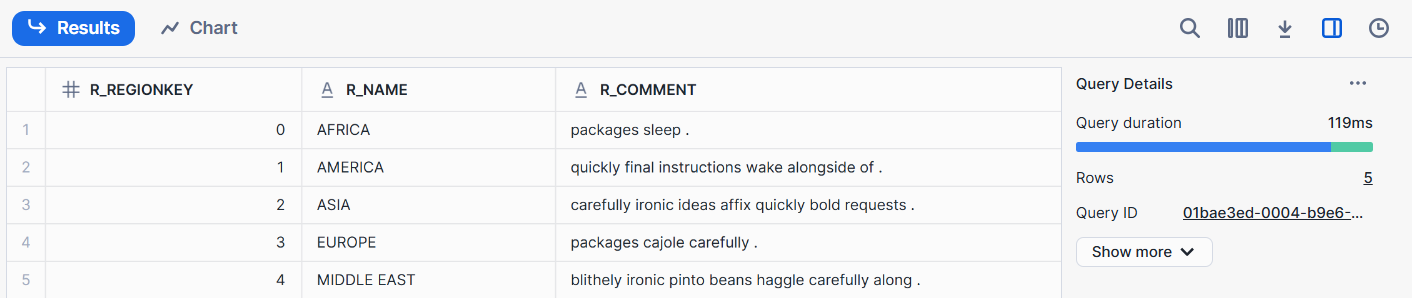
select \* from epam\_lab.core\_dwh\_clone.brand;



* **Time travel**

ALTER TABLE epam\_lab.core\_dwh.region SET DATA\_RETENTION\_TIME\_IN\_DAYS=1;

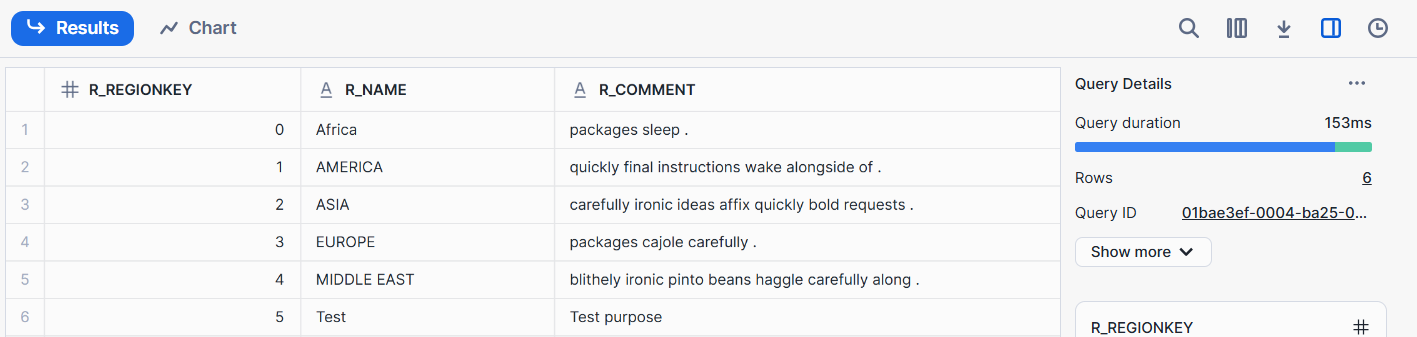
select \* from epam\_lab.core\_dwh.region;



insert into epam\_lab.core\_dwh.region values (5, 'Test', 'Test purpose');

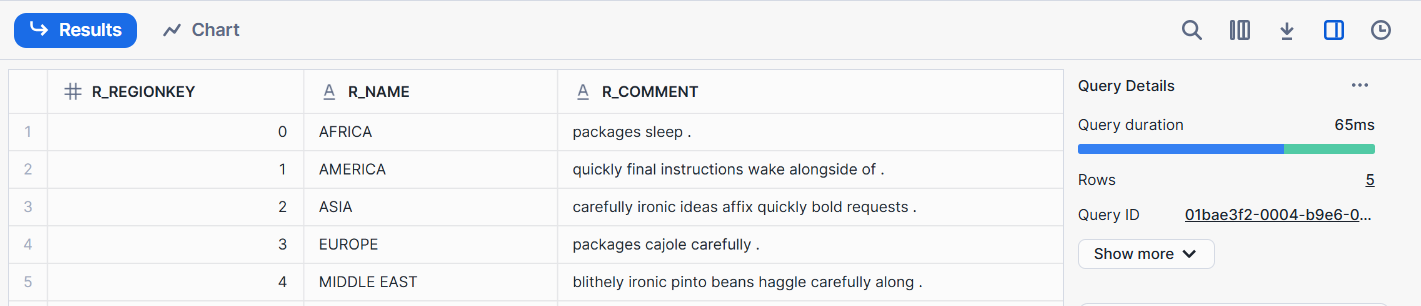
update epam\_lab.core\_dwh.region

set R\_NAME = 'Africa' where r\_regionkey = 0;

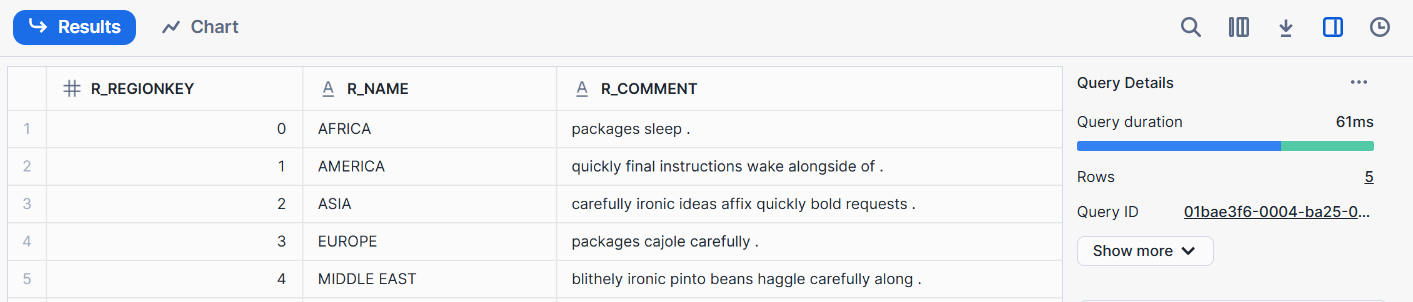


select getdate() -- 2025-03-09 03:24:35.849 -0700

SELECT \* FROM epam\_lab.core\_dwh.region BEFORE(TIMESTAMP => '2025-03-09 03:10:35.849 -0700'::timestamp\_tz);



SELECT \* FROM epam\_lab.core\_dwh.region AT(OFFSET => -60\*11);



SELECT \* FROM epam\_lab.core\_dwh.region AT(OFFSET => -60\*8);

A screenshot of a computer

AI-generated content may be incorrect.

A close-up of a computer screen

AI-generated content may be incorrect.

* **Data Sharing**

use role accountadmin;

create or replace share epam\_lab\_demo;

grant usage on database epam\_lab to share epam\_lab\_demo;

grant usage on schema epam\_lab.data\_mart to share epam\_lab\_demo;

grant select on all tables in schema epam\_lab.data\_mart to share epam\_lab\_demo;

alter share epam\_lab\_demo add accounts = RWXFUUQ.YZ51351;

--Consumer

use role accountadmin;

show shares;

create database epam\_lab\_demo from share SYNHUMY.ZBB74720;

## Snowpipe

create or replace TABLE TSM.TEST.PIPE\_TEST (

ID NUMBER(5,0),

NAME VARCHAR(80),

SEGMENT NUMBER(5,0)

);

create or replace file format PIPECOLSEP\_ONEHEADROW

type = 'CSV'

field\_delimiter = '|'

skip\_header = 1

field\_optionally\_enclosed\_by = '"'

trim\_space = TRUE;

create or replace storage integration snowpipe\_az\_integration

type = external\_stage

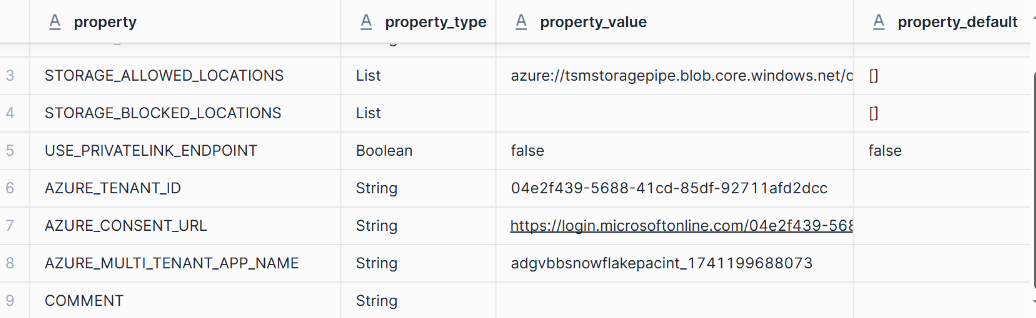
storage\_provider = azure

enabled = true

azure\_tenant\_id = '04e2f439-5688-41cd-85df-92711afd2dcc'

storage\_allowed\_locations = ('azure://tsmstoragepipe.blob.core.windows.net/ordercsv');

desc storage integration snowpipe\_az\_integration;



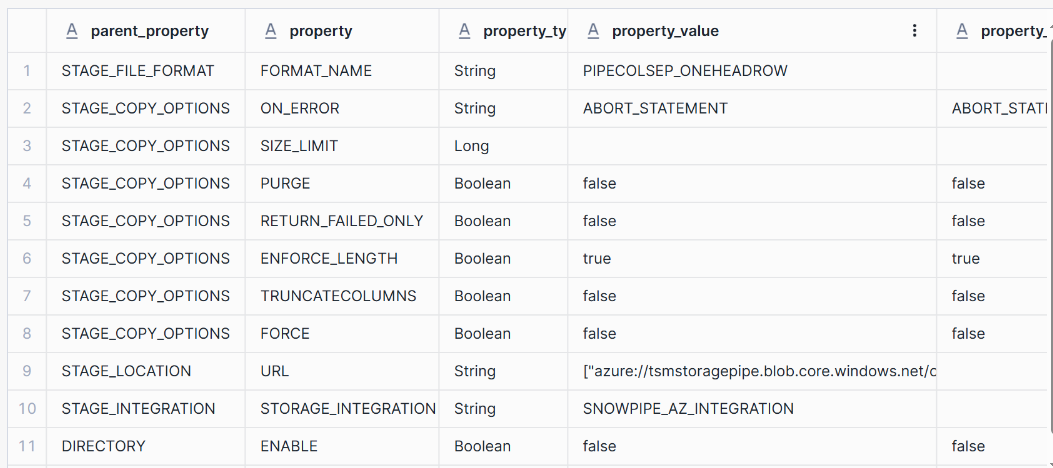
create or replace stage az\_stage

storage\_integration = snowpipe\_az\_integration

url = 'azure://tsmstoragepipe.blob.core.windows.net/ordercsv'

file\_format = ( format\_name=PIPECOLSEP\_ONEHEADROW );

describe stage az\_stage;



CREATE OR REPLACE NOTIFICATION INTEGRATION snowpipe\_event

ENABLED = TRUE

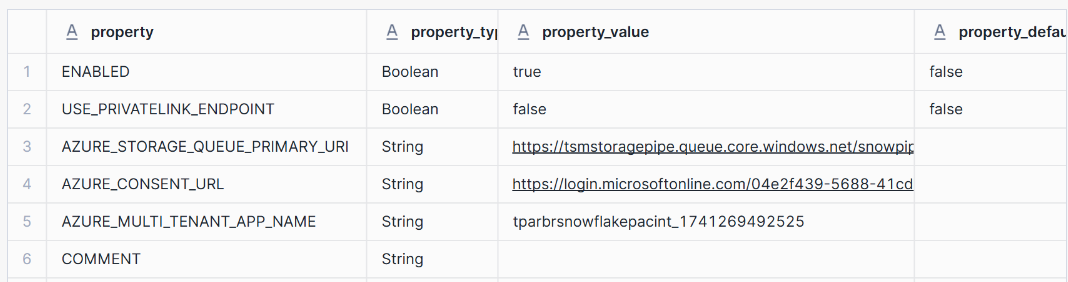
TYPE = QUEUE

NOTIFICATION\_PROVIDER = AZURE\_STORAGE\_QUEUE

AZURE\_STORAGE\_QUEUE\_PRIMARY\_URI = 'https://tsmstoragepipe.queue.core.windows.net/snowpipequeue'

AZURE\_TENANT\_ID = '04e2f439-5688-41cd-85df-92711afd2dcc';

DESCRIBE NOTIFICATION INTEGRATION snowpipe\_event;



CREATE or replace PIPE test\_pipe

auto\_ingest = true integration = 'SNOWPIPE\_EVENT'

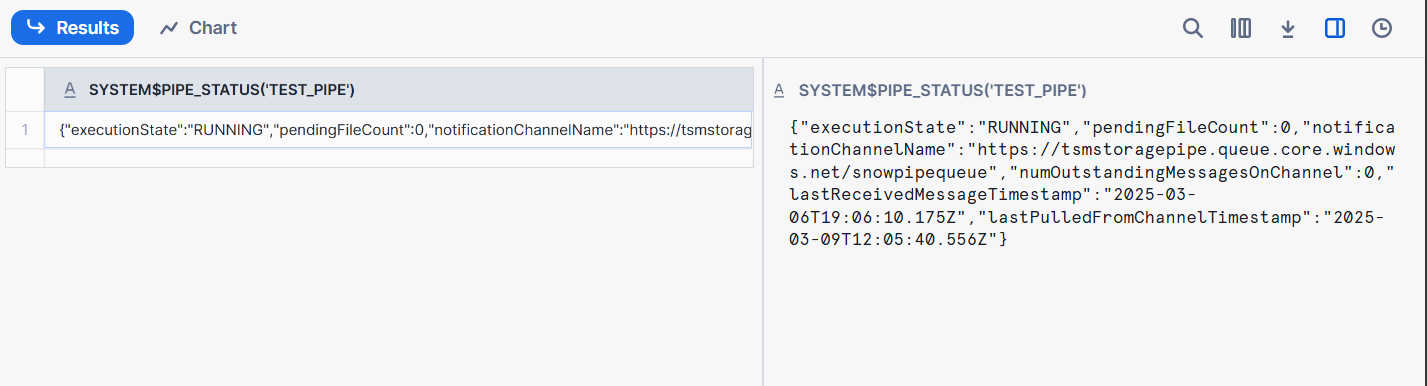
AS

COPY INTO tsm.test.pipe\_test

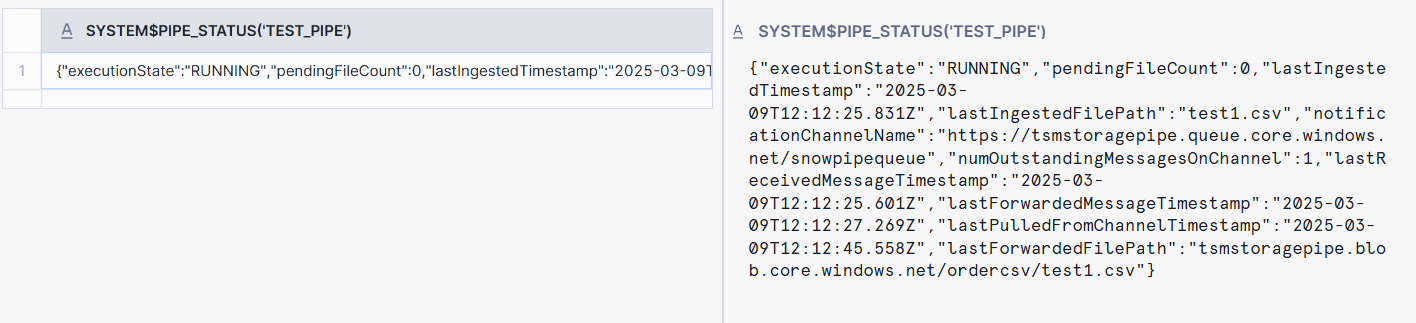
FROM @az\_stage

FILE\_FORMAT = ( format\_name=PIPECOLSEP\_ONEHEADROW );

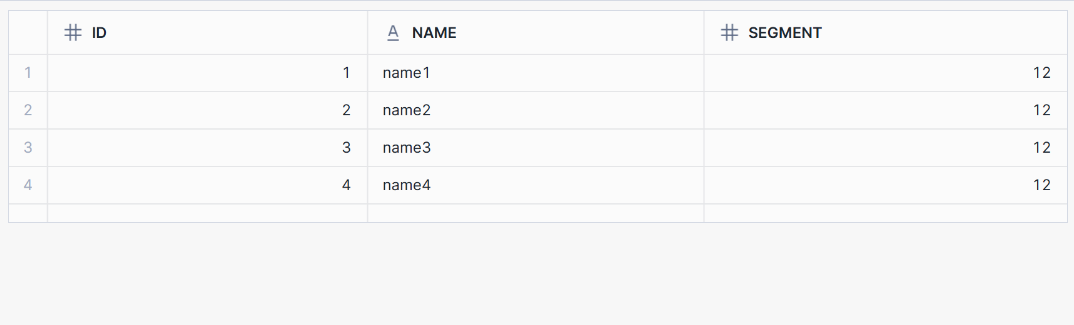
SELECT SYSTEM$PIPE\_STATUS('test\_pipe');



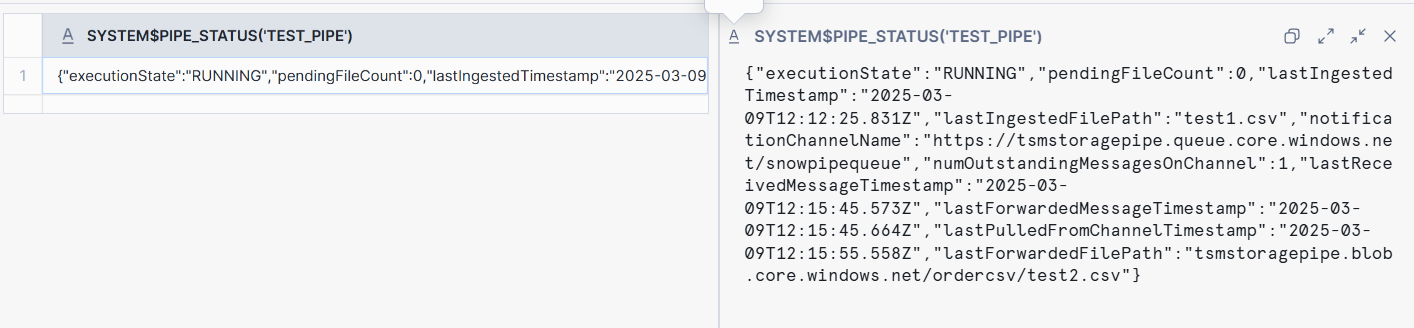
--Uploaded file test1.csv



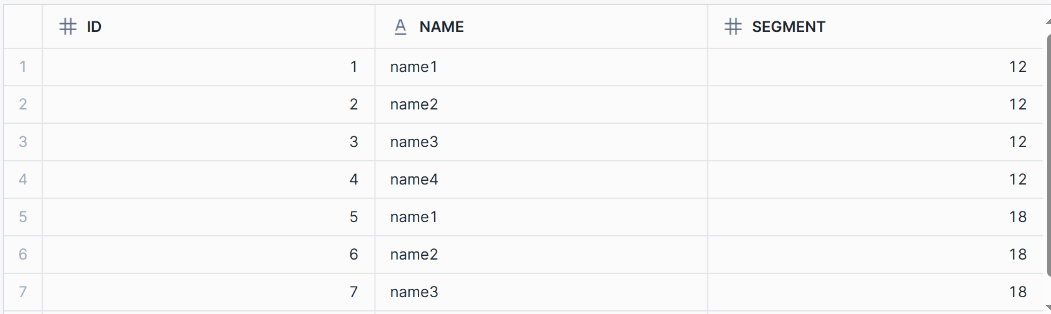
select \* from tsm.test.pipe\_test;



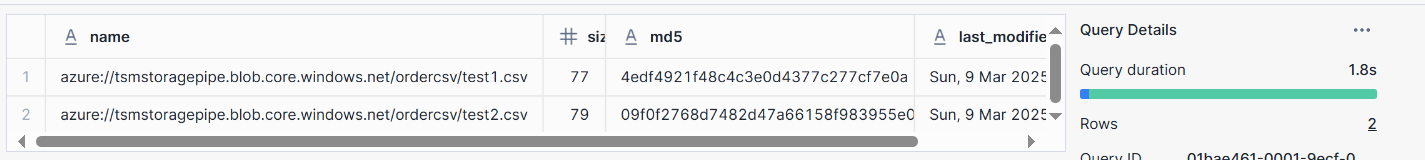
--Uploaded file test2.csv



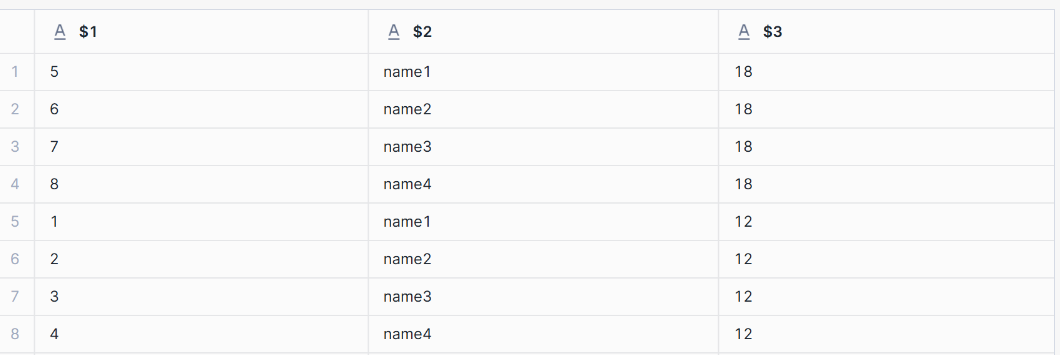
select \* from tsm.test.pipe\_test;



list @az\_stage;



select $1, $2, $3 from @az\_stage;



A screenshot of a web page

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.