🔹 Assignment 1: Permanent Table

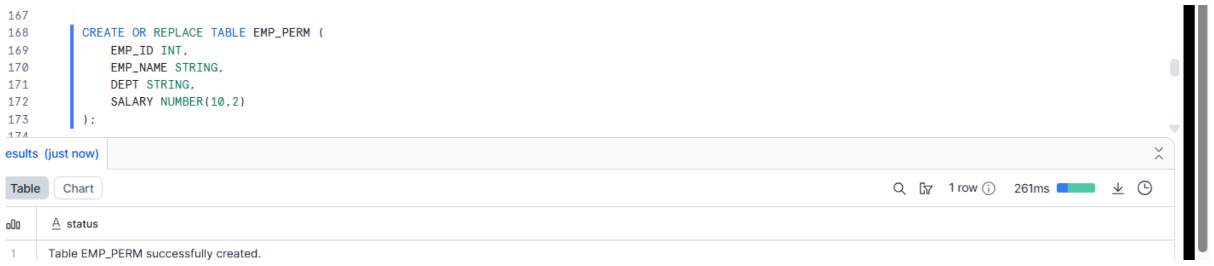
Goal: Test persistence, Time Travel & recovery

Tasks:

1. Create a permanent table EMP_PERM
2. Insert minimum 5 records
3. Logout and login again
4. Verify data persists
5. Delete 2 records
6. Query deleted data using **Time Travel**
7. Restore deleted data
8. Drop the table
9. Recover using UNDROP TABLE

*A **Permanent Table** is the **default table type in Snowflake** that **stores data permanently** until it is explicitly dropped.*

1. Create a permanent table EMP_PERM

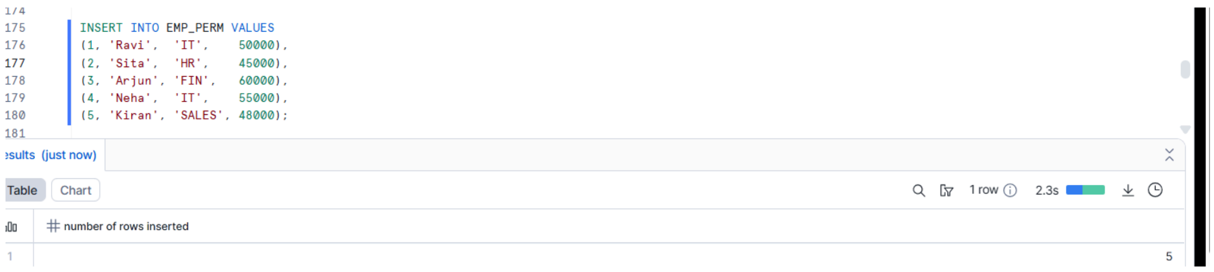


```
CREATE OR REPLACE TABLE EMP_PERM (
    EMP_ID INT,
    EMP_NAME STRING,
    DEPT STRING,
    SALARY NUMBER(10,2)
);
```

esults (just now)

```
status
1  Table EMP_PERM successfully created.
```

2. Insert minimum 5 records



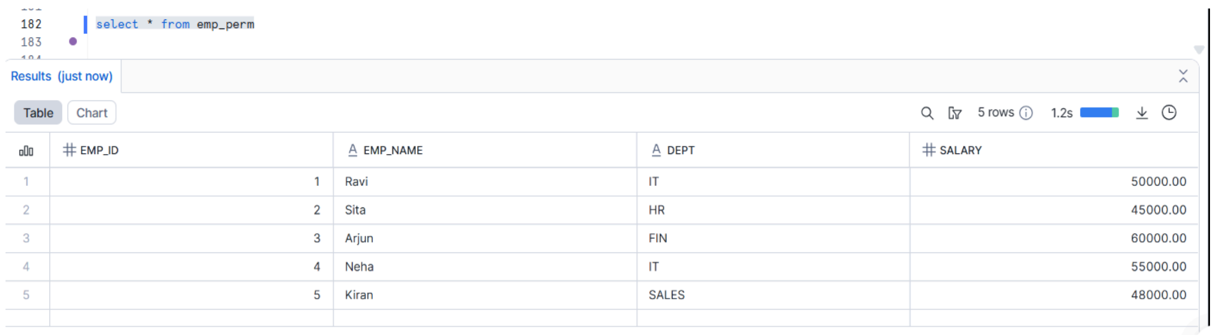
```
INSERT INTO EMP_PERM VALUES
(1, 'Ravi',  'IT',    50000),
(2, 'Sita',  'HR',    45000),
(3, 'Arjun', 'FIN',   60000),
(4, 'Neha',  'IT',    55000),
(5, 'Kiran', 'SALES', 48000);
```

esults (just now)

```
number of rows inserted
1                        5
```

Logout and login again

4. Verify data persists



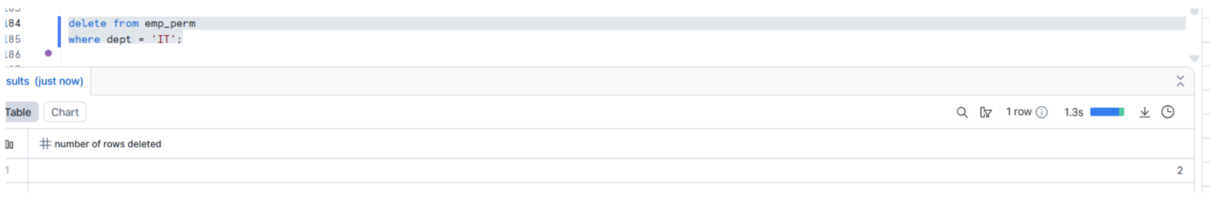
```
select * from emp_perm
```

Results (just now)

Table Chart — 5 rows 1.2s

	EMP_ID	EMP_NAME	DEPT	SALARY
1	1	Ravi	IT	50000.00
2	2	Sita	HR	45000.00
3	3	Arjun	FIN	60000.00
4	4	Neha	IT	55000.00
5	5	Kiran	SALES	48000.00

5. Delete 2 records

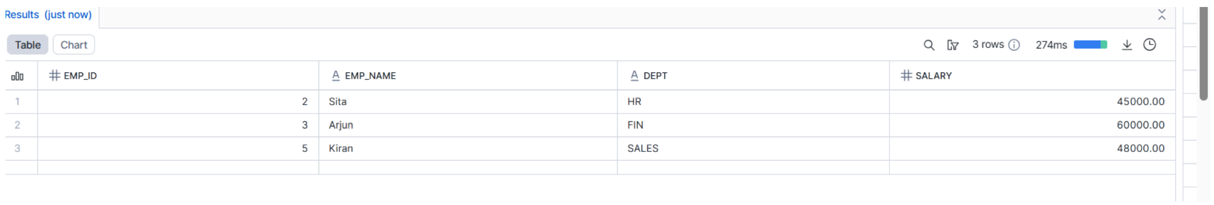


```
delete from emp_perm
where dept = 'IT';
```

sults (just now)

Table Chart — 1 row 1.3s

	number of rows deleted
1	2



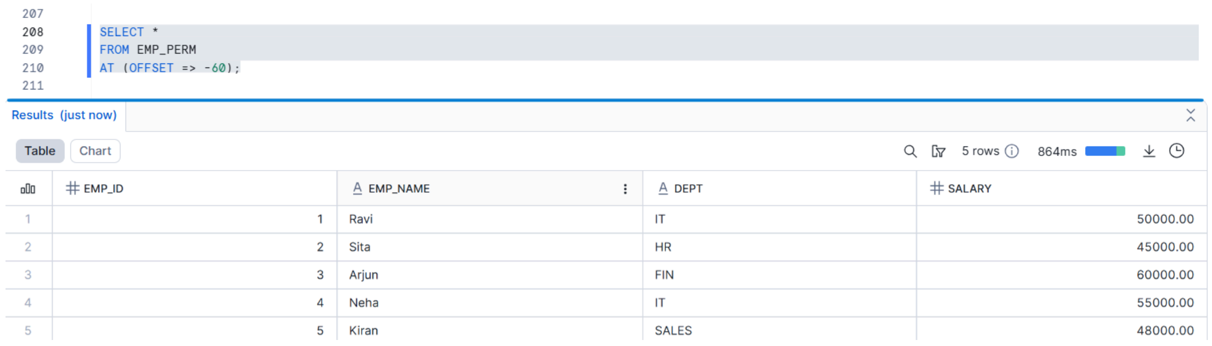
Results (just now)

Table Chart — 3 rows 274ms

	EMP_ID	EMP_NAME	DEPT	SALARY
1	2	Sita	HR	45000.00
2	3	Arjun	FIN	60000.00
3	5	Kiran	SALES	48000.00

6. Query deleted data using Time Travel

After deleting records from a permanent table, you can query the previous version of the data using Time Travel.



```
SELECT *
FROM EMP_PERM
AT (OFFSET => -60);
```

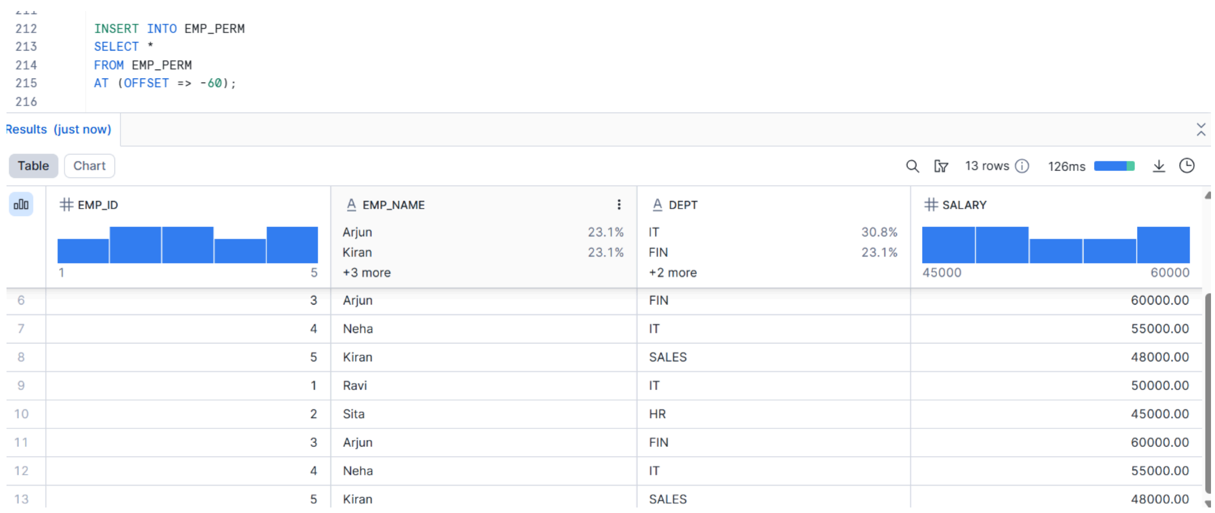
Results (just now)

Table Chart — 5 rows 864ms

	EMP_ID	EMP_NAME	DEPT	SALARY
1	1	Ravi	IT	50000.00
2	2	Sita	HR	45000.00
3	3	Arjun	FIN	60000.00
4	4	Neha	IT	55000.00
5	5	Kiran	SALES	48000.00

- *This returns the table data **as it was 60 seconds ago**, before the DELETE operation.*

7. Restore deleted data



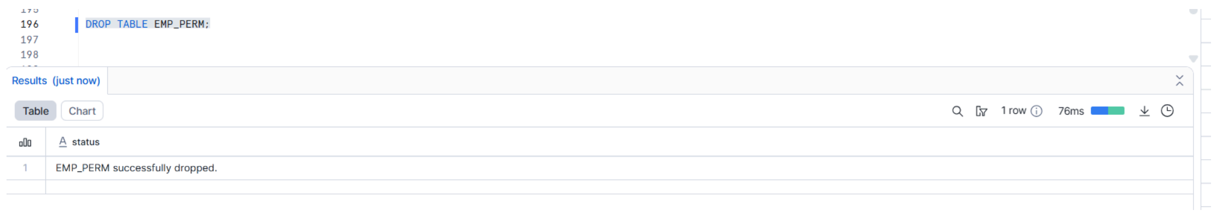
```
INSERT INTO EMP_PERM
SELECT *
FROM EMP_PERM
AT (OFFSET => -60);
```

Results (just now)

Table Chart 13 rows 126ms

	EMP_ID	EMP_NAME	DEPT	SALARY
	1 – 5	Arjun 23.1%, Kiran 23.1%, +3 more	IT 30.8%, FIN 23.1%, +2 more	45000 – 60000
6	3	Arjun	FIN	60000.00
7	4	Neha	IT	55000.00
8	5	Kiran	SALES	48000.00
9	1	Ravi	IT	50000.00
10	2	Sita	HR	45000.00
11	3	Arjun	FIN	60000.00
12	4	Neha	IT	55000.00
13	5	Kiran	SALES	48000.00

8. Drop the table



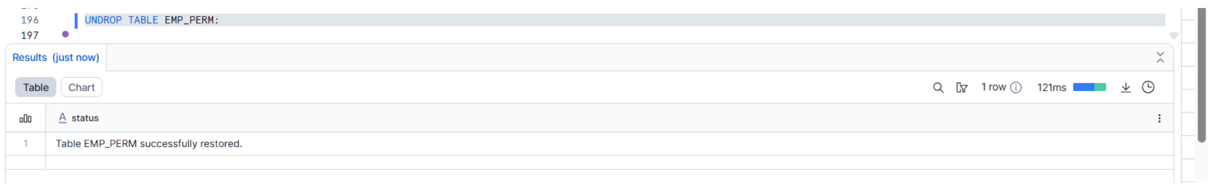
```
DROP TABLE EMP_PERM;
```

Results (just now)

Table Chart 1 row 76ms

	status
1	EMP_PERM successfully dropped.

9. Recover using UNDROP TABLE



```
UNDROP TABLE EMP_PERM;
```

Results (just now)

Table Chart 1 row 121ms

	status
1	Table EMP_PERM successfully restored.

🔹 Assignment 2: Temporary Table

Goal: Test session-based background cleanup

Tasks:

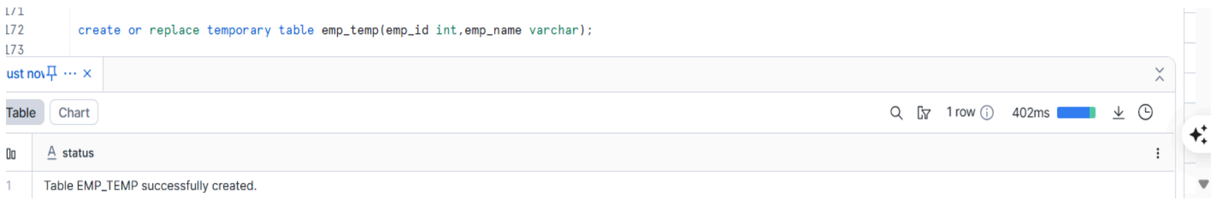
1. Create temporary table EMP_TEMP
2. Insert records
3. Query data successfully

4. End the session
5. Reconnect and query the table

Learning Outcome:
✔ Table automatically removed after session ends
✔ No Time Travel or Fail-safe

*A **Temporary Table** is a table that **exists only for the current session**.Once the session ends (logout / worksheet closed), the table is **automatically dropped**.*

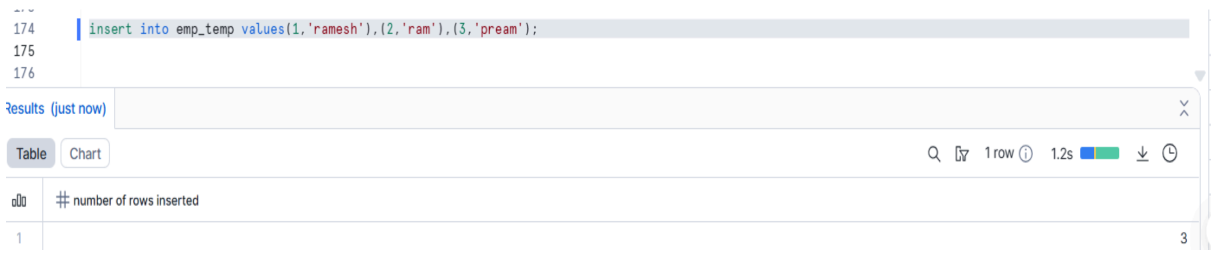
1. Create temporary table EMP_TEMP



```
create or replace temporary table emp_temp(emp_id int,emp_name varchar);
```

```
1 row   402ms
  status
1 Table EMP_TEMP successfully created.
```

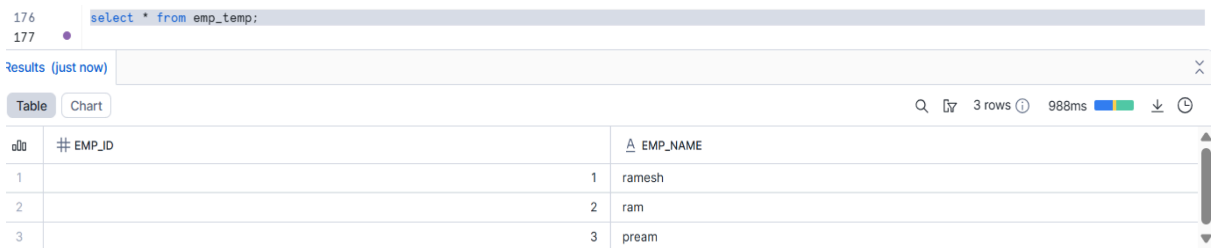
2. Insert records



```
insert into emp_temp values(1,'ramesh'),(2,'ram'),(3,'pream');
```

```
Results (just now)
1 row   1.2s
  number of rows inserted
1                       3
```

3. Query data successfully



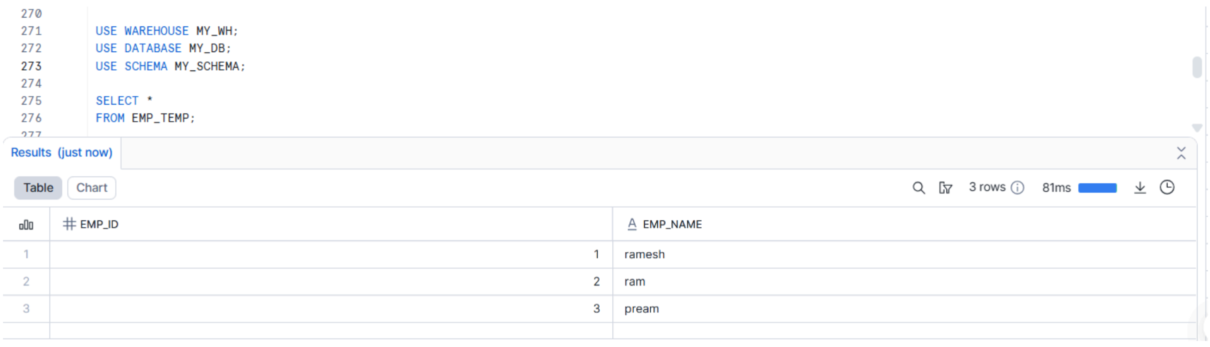
```
select * from emp_temp;
```

```
Results (just now)
3 rows   988ms
  EMP_ID   EMP_NAME
1      1   ramesh
2      2   ram
3      3   pream
```

4. End the session

   Close the worksheet
   Logout from Snowflake

5. Reconnect and query the table



```
USE WAREHOUSE MY_WH;
USE DATABASE MY_DB;
USE SCHEMA MY_SCHEMA;

SELECT *
FROM EMP_TEMP;
```

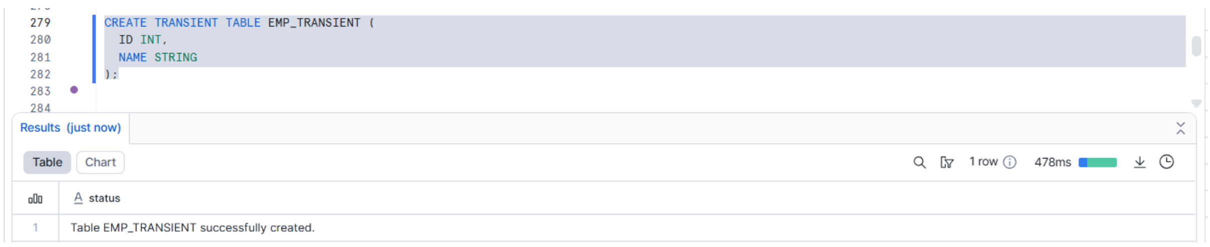
	EMP_ID	EMP_NAME
1	1	ramesh
2	2	ram
3	3	pream

🔹 Assignment 3: Transient Table

Goal: Test limited retention behavior

Tasks:

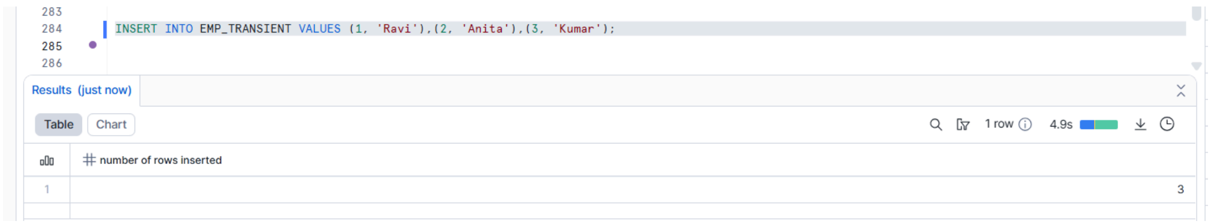
1. Create transient table EMP_TRANS



```
CREATE TRANSIENT TABLE EMP_TRANSIENT (
  ID INT,
  NAME STRING
);
```

	status
1	Table EMP_TRANSIENT successfully created.

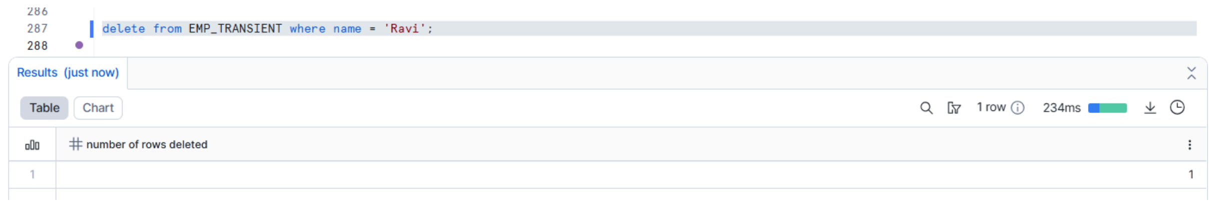
2. Insert records



```
INSERT INTO EMP_TRANSIENT VALUES (1, 'Ravi'),(2, 'Anita'),(3, 'Kumar');
```

	number of rows inserted
1	3

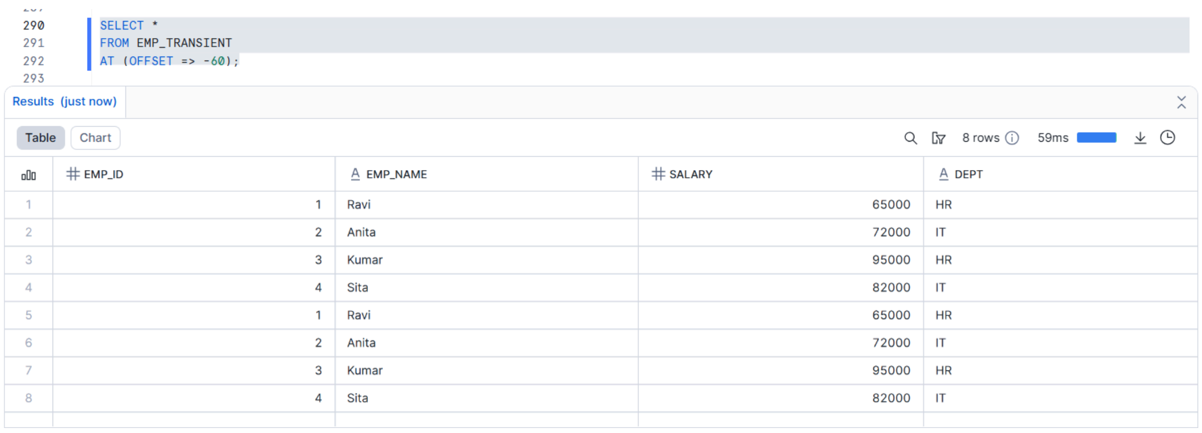
3. Delete records



```
delete from EMP_TRANSIENT where name = 'Ravi';
```

	number of rows deleted
1	1

4. Query data using Time Travel (within retention)



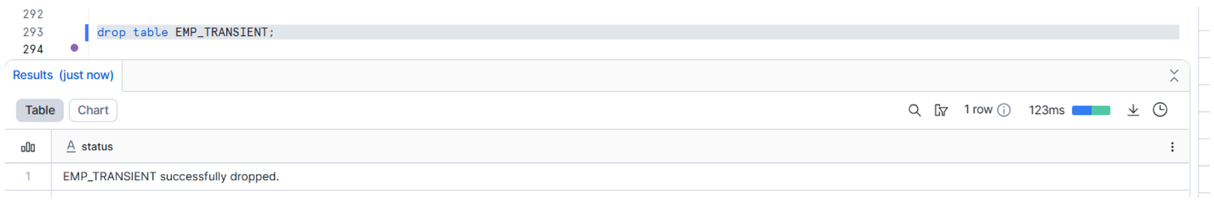
```
SELECT *
FROM EMP_TRANSIENT
AT (OFFSET => -60);
```

Results (just now)

Table Chart 8 rows 59ms

	EMP_ID	EMP_NAME	SALARY	DEPT
1	1	Ravi	65000	HR
2	2	Anita	72000	IT
3	3	Kumar	95000	HR
4	4	Sita	82000	IT
5	1	Ravi	65000	HR
6	2	Anita	72000	IT
7	3	Kumar	95000	HR
8	4	Sita	82000	IT

5. Drop the table



```
drop table EMP_TRANSIENT;
```

Results (just now)

Table Chart 1 row 123ms

	status
1	EMP_TRANSIENT successfully dropped.

6. Attempt recovery after retention period

Learning Outcome:

✔ Limited Time Travel

✔ No Fail-safe

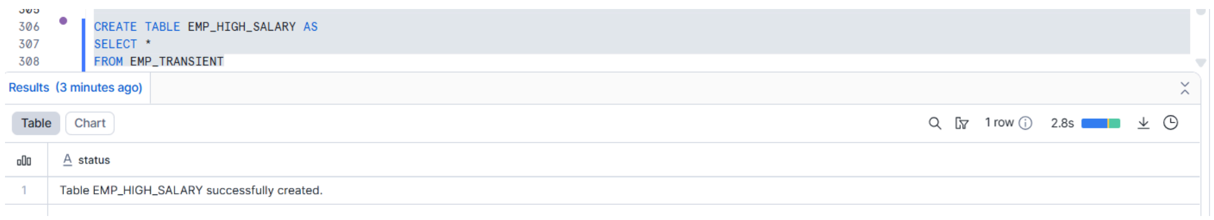
✔ Faster background purge

🔹 Assignment 4: CTAS (Create Table As Select)

Goal: Test background data transformation

Tasks:

1. Create table EMP_HIGH_SALARY using CTAS



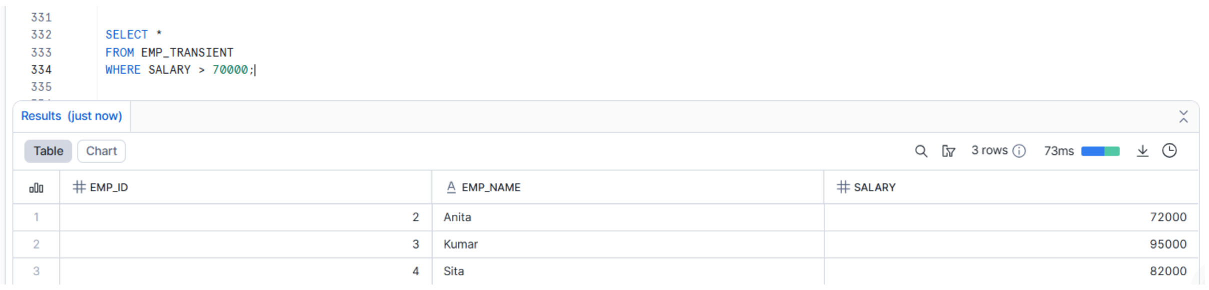
```
CREATE TABLE EMP_HIGH_SALARY AS
SELECT *
FROM EMP_TRANSIENT
```

Results (3 minutes ago)

Table Chart 1 row 2.8s

	status
1	Table EMP_HIGH_SALARY successfully created.

2. Filter salary > 70,000



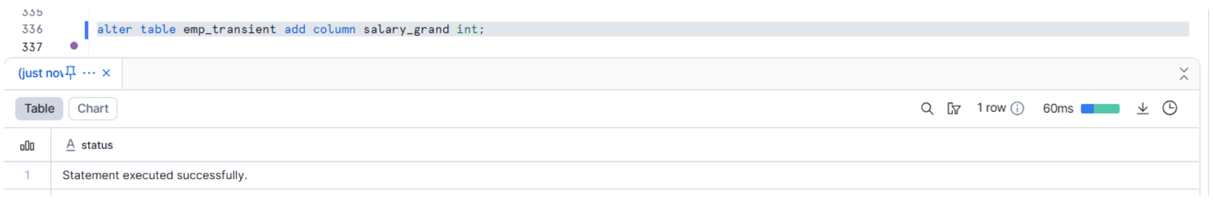
```
SELECT *
FROM EMP_TRANSIENT
WHERE SALARY > 70000;
```

Results (just now)

Table Chart 3 rows 73ms

	EMP_ID	EMP_NAME	SALARY
1	2	Anita	72000
2	3	Kumar	95000
3	4	Sita	82000

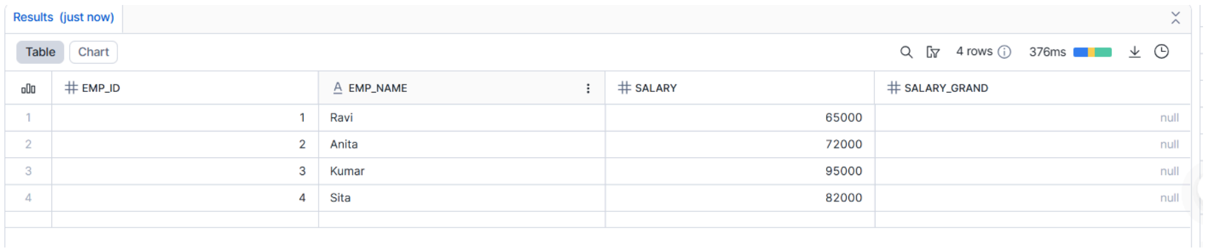
3. Add calculated column SALARY_GRADE



```
alter table emp_transient add column salary_grand int;
```

Table Chart 1 row 60ms

	status
1	Statement executed successfully.

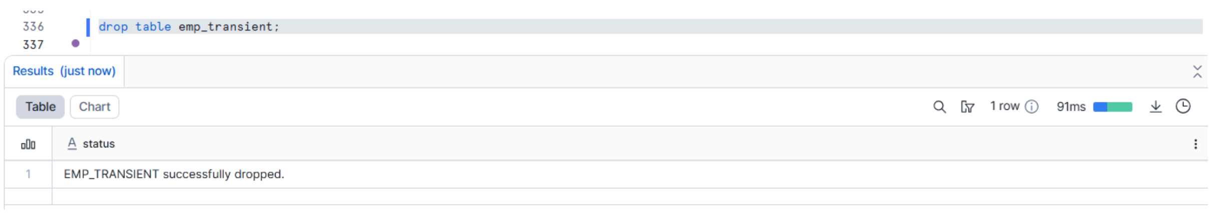


Results (just now)

Table Chart 4 rows 376ms

	EMP_ID	EMP_NAME	SALARY	SALARY_GRAND
1	1	Ravi	65000	null
2	2	Anita	72000	null
3	3	Kumar	95000	null
4	4	Sita	82000	null

4. Drop source table



```
drop table emp_transient;
```

Results (just now)

Table Chart 1 row 91ms

	status
1	EMP_TRANSIENT successfully dropped.

5. Verify CTAS table still exists

Learning Outcome:

✔ Independent table creation

✔ Background transformation execution

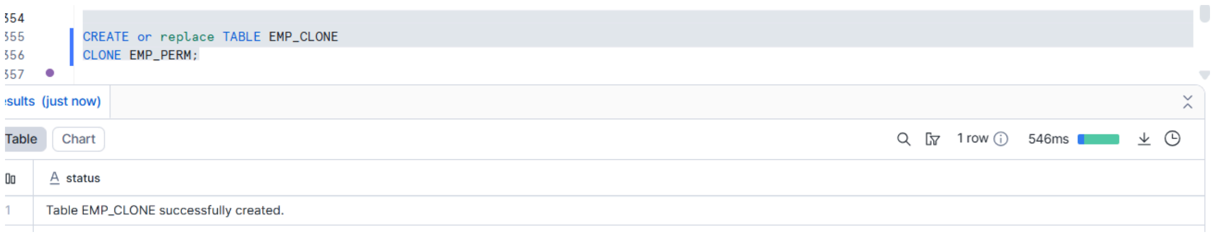
CLONE = fast, shares storage
CTAS = slow, copies all data

🔹 Assignment 5: Table Cloning

Goal: Test zero-copy cloning

Tasks:

1. Clone EMP_PERM as EMP_CLONE



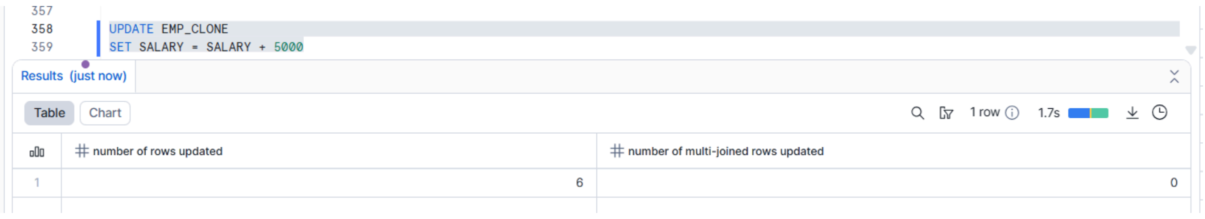
```
CREATE or replace TABLE EMP_CLONE
CLONE EMP_PERM;
```

sults (just now)

Table Chart 1 row 546ms

	status
1	Table EMP_CLONE successfully created.

2. Update data in cloned table



```
UPDATE EMP_CLONE
SET SALARY = SALARY + 5000
```

Results (just now)

Table Chart 1 row 1.7s

	number of rows updated	number of multi-joined rows updated
1	6	0

3. Compare original vs clone

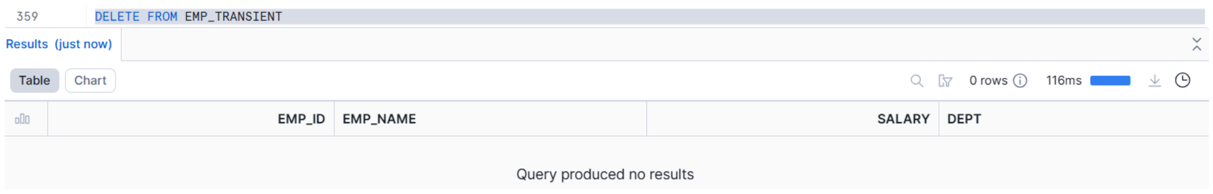
4. Explain storage behavior

🔹 Assignment 6: Time Travel Testing

Goal: Test point-in-time recovery

Tasks:

1. Perform DELETE on any table



```
DELETE FROM EMP_TRANSIENT
```

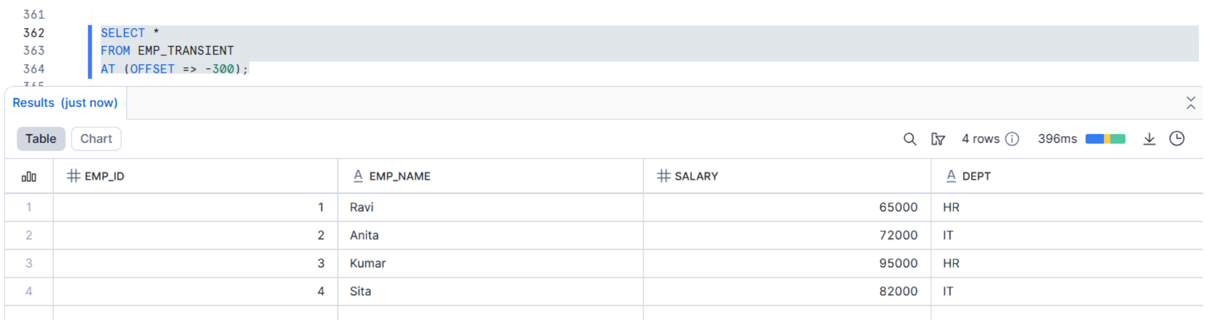
Results (just now)

Table Chart — 0 rows — 116ms

EMP_ID	EMP_NAME	SALARY	DEPT

Query produced no results

2. Query using:
   - AT OFFSET



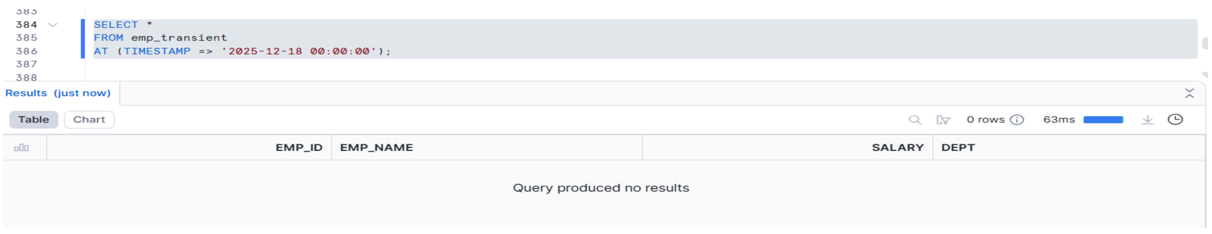
```
SELECT *
FROM EMP_TRANSIENT
AT (OFFSET => -300);
```

Results (just now)

Table Chart — 4 rows — 396ms

	EMP_ID	EMP_NAME	SALARY	DEPT
1	1	Ravi	65000	HR
2	2	Anita	72000	IT
3	3	Kumar	95000	HR
4	4	Sita	82000	IT

   - AT TIMESTAMP_Exactly when did it happen?



```
SELECT *
FROM emp_transient
AT (TIMESTAMP => '2025-12-18 00:00:00');
```

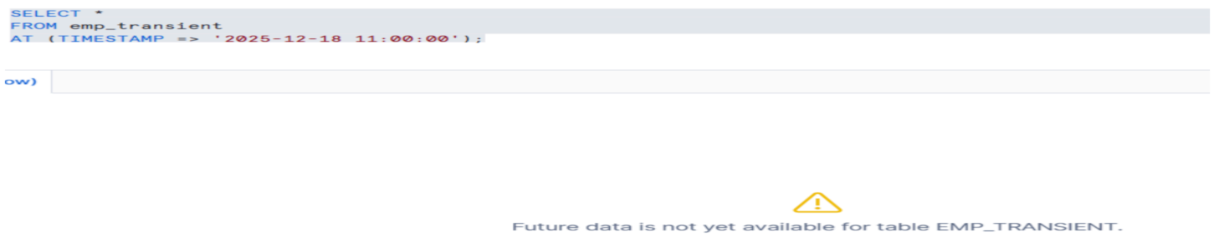
Results (just now)

Table Chart — 0 rows — 63ms

EMP_ID	EMP_NAME	SALARY	DEPT

Query produced no results

I changed the time:

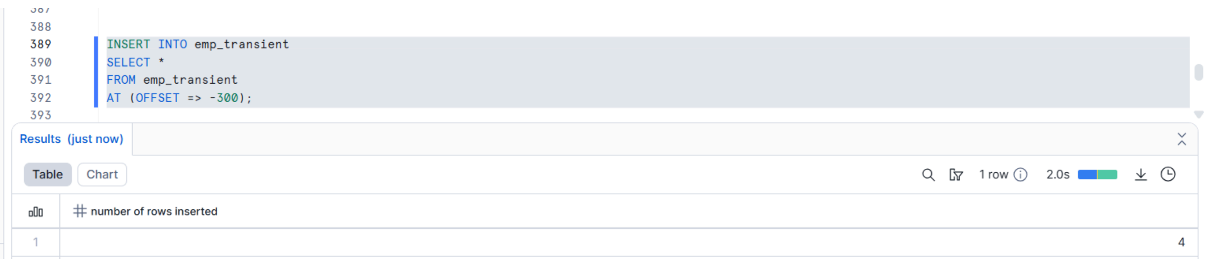


```
SELECT *
FROM emp_transient
AT (TIMESTAMP => '2025-12-18 11:00:00');
```

Future data is not yet available for table EMP_TRANSIENT.

3. Restore deleted data

Using OFFSET



```
INSERT INTO emp_transient
SELECT *
FROM emp_transient
AT (OFFSET => -300);
```

Results (just now)

number of rows inserted
4

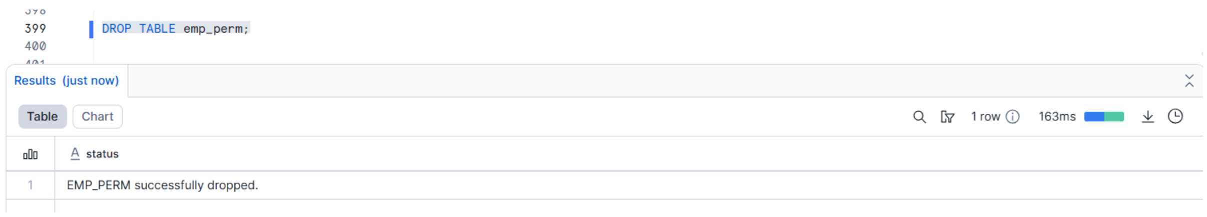
🔹 Assignment 7: Drop & Purge Behavior

Goal: Compare cleanup rules

Purge Behavior - deleted forever

Tasks:

1. Drop Permanent table and recover it



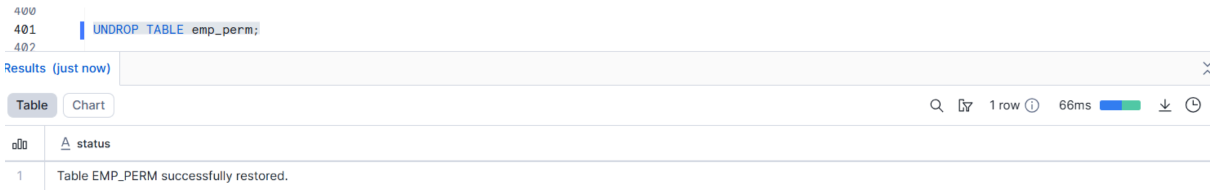
```
DROP TABLE emp_perm;
```

Results (just now)

Table | Chart | 1 row | 163ms

	status
1	EMP_PERM successfully dropped.

Recover it



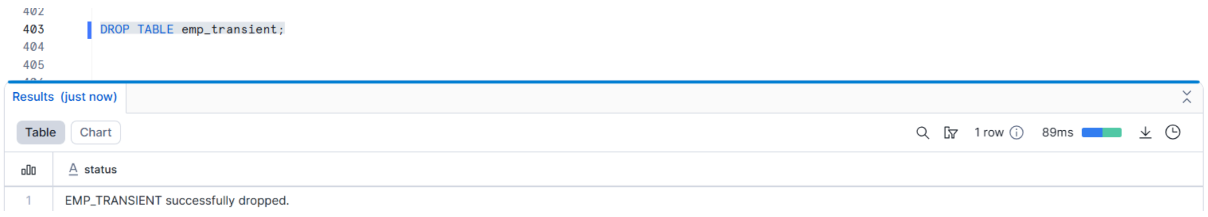
```
UNDROP TABLE emp_perm;
```

Results (just now)

Table | Chart | 1 row | 66ms

	status
1	Table EMP_PERM successfully restored.

2. Drop Transient table and attempt recovery



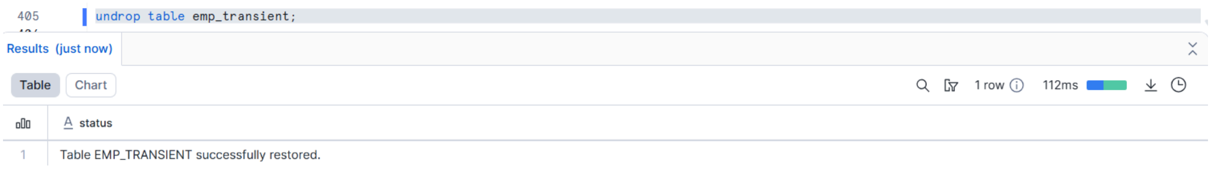
```
DROP TABLE emp_transient;
```

Results (just now)

Table | Chart | 1 row | 89ms

	status
1	EMP_TRANSIENT successfully dropped.

Attempt recovery



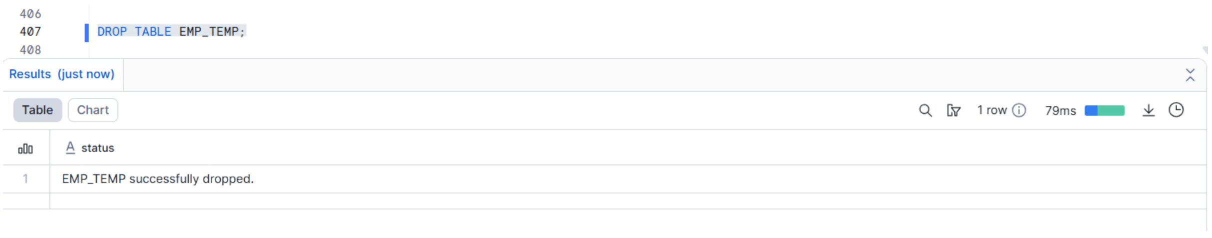
```
undrop table emp_transient;
```

Results (just now)

Table | Chart | 1 row | 112ms

	status
1	Table EMP_TRANSIENT successfully restored.

3. Drop Temporary table and verify immediate removal



```
DROP TABLE EMP_TEMP;
```

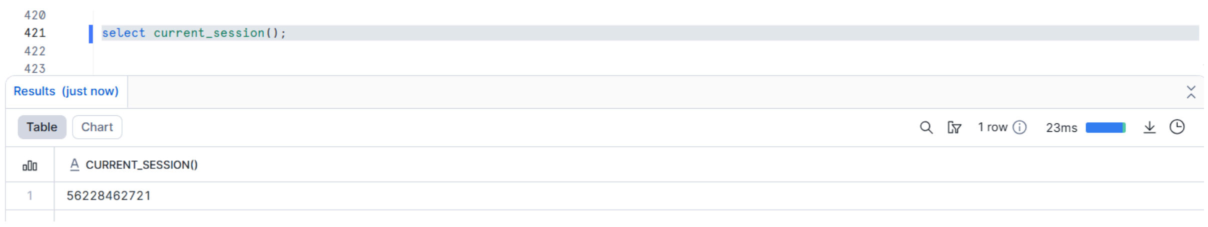
Results (just now)

Table | Chart | 1 row | 79ms

	status
1	EMP_TEMP successfully dropped.

.....Verify immediate removal :
By using another session_id .it didn’t work.

Old session_id



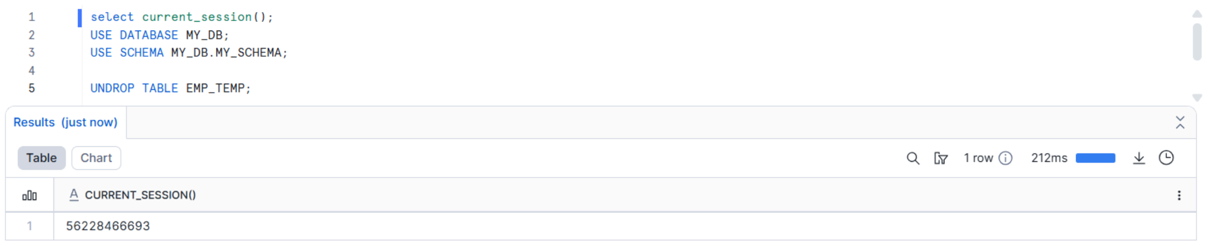
```
select current_session();
```

Results (just now)

Table | Chart — 1 row — 23ms

```
  CURRENT_SESSION()
1 56228462721
```

New session_id:



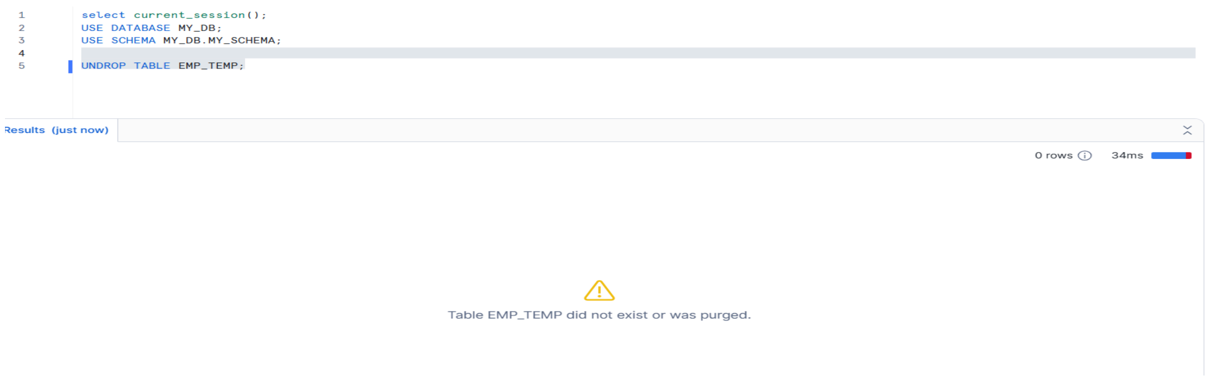
```
select current_session();
USE DATABASE MY_DB;
USE SCHEMA MY_DB.MY_SCHEMA;

UNDROP TABLE EMP_TEMP;
```

Results (just now)

Table | Chart — 1 row — 212ms

```
  CURRENT_SESSION()
1 56228466693
```

Final answer:



```
select current_session();
USE DATABASE MY_DB;
USE SCHEMA MY_DB.MY_SCHEMA;

UNDROP TABLE EMP_TEMP;
```

Results (just now)

0 rows — 34ms

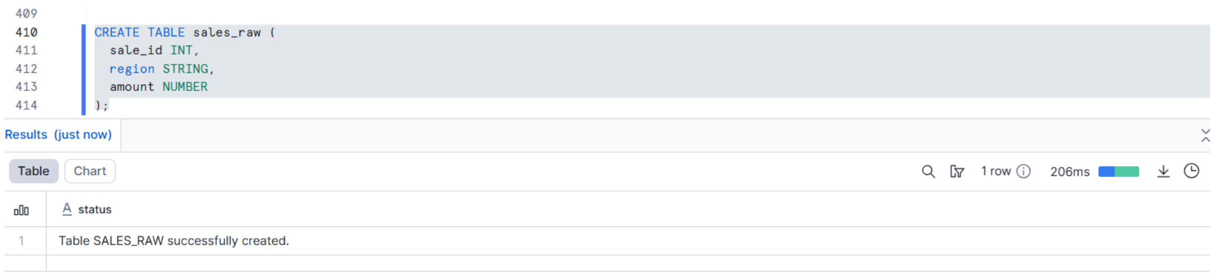
Table EMP_TEMP did not exist or was purged.

Assignment 9: Dynamic Tables

Goal: Test automated background refresh

A Dynamic Table is a table that automatically updates itself when the source data changes.

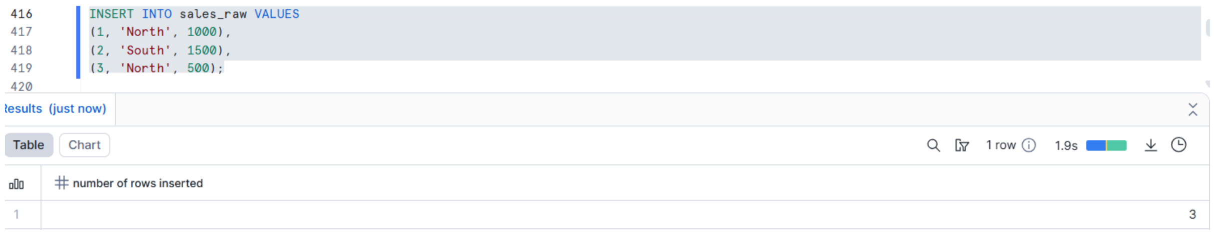
1. Create a source table SALES_RAW



```sql
CREATE TABLE sales_raw (
  sale_id INT,
  region STRING,
  amount NUMBER
);
```

```
status
1  Table SALES_RAW successfully created.
```

2. Insert sample sales data

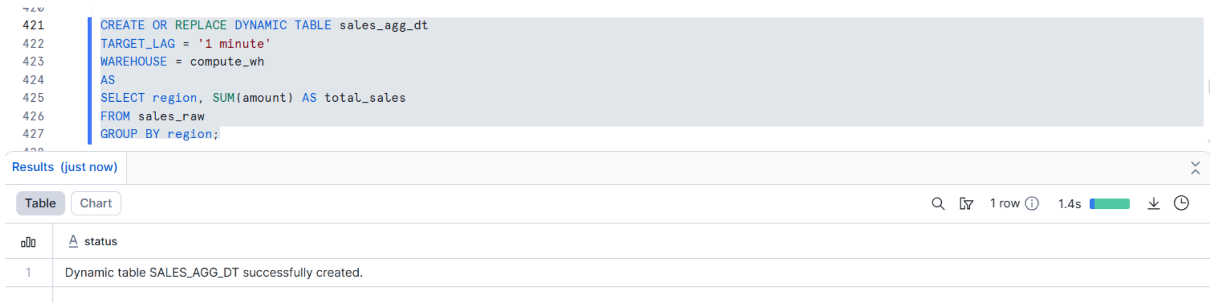


```sql
INSERT INTO sales_raw VALUES
(1, 'North', 1000),
(2, 'South', 1500),
(3, 'North', 500);
```

```
number of rows inserted
1                                  3
```

3. Create a **Dynamic Table** SALES_AGG_DT
   - Aggregate total sales by region

Target_lag *means* ***how long the dynamic table can wait before updating****.*



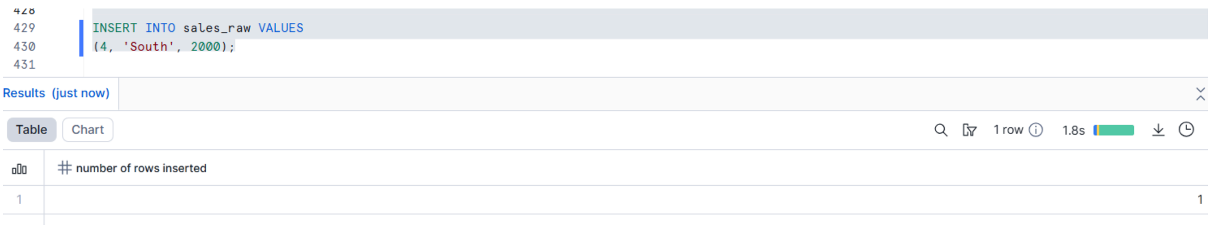
```sql
CREATE OR REPLACE DYNAMIC TABLE sales_agg_dt
TARGET_LAG = '1 minute'
WAREHOUSE = compute_wh
AS
SELECT region, SUM(amount) AS total_sales
FROM sales_raw
GROUP BY region;
```

```
status
1  Dynamic table SALES_AGG_DT successfully created.
```

4. Set refresh lag (e.g., 1 minute)

Refresh lag *is the waiting time before updated data shows in a table.*

5. Insert new data into SALES_RAW



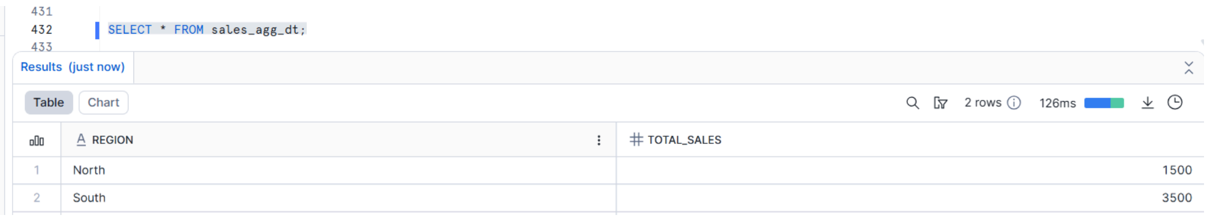
```
INSERT INTO sales_raw VALUES
(4, 'South', 2000);
```

Results (just now)

Table Chart 1 row 1.8s

	number of rows inserted
1	1

6. Observe automatic refresh



```
SELECT * FROM sales_agg_dt;
```

Results (just now)

Table Chart 2 rows 126ms

	REGION	TOTAL_SALES
1	North	1500
2	South	3500