employees.csv

emp_id,emp_name,department,job_title,salary,location,hire_date

101,Ramesh,IT,Data Engineer,95000,Bangalore,2019-06-15

102,Anita,HR,HR Executive,65000,Hyderabad,2020-02-10

103,John,Finance,Finance Manager,110000,New York,2016-09-01

104,Meena,IT,Senior Data Analyst,98000,Bangalore,2018-11-23

105,David,IT,Software Engineer,90000,San Francisco,2021-07-05

106,Priya,HR,HR Business Partner,72000,Hyderabad,2019-03-18

107,Michael,Finance,Account Director,120000,New York,2015-01-12

108,Sneha,IT,Junior Developer,78000,Bangalore,2022-08-09

109,Arjun,IT,Data Architect,115000,Bangalore,2017-05-30

110,Emily,Marketing,Marketing Manager,88000,Chicago,2018-10-17

---

departments.csv

dept_id,department,manager,budget,location

10,IT,Arjun Rao,500000,Bangalore

20,HR,Anita Sharma,200000,Hyderabad

30,Finance,John Smith,600000,New York

40,Marketing,Emily Wilson,300000,Chicago

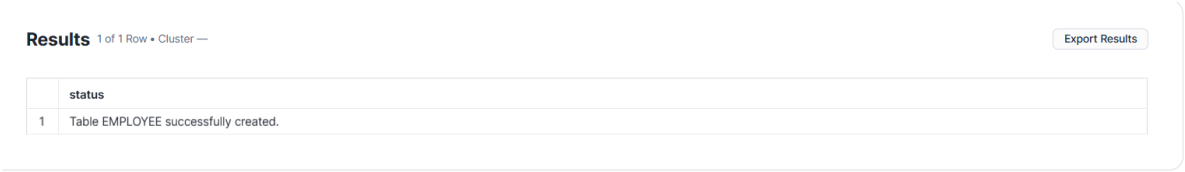
🧪 **Snowflake Cloning – Practical Assignments**

Important Rule for Students:
❌ No CTAS
❌ No data reload
✅ Only cloning & observation

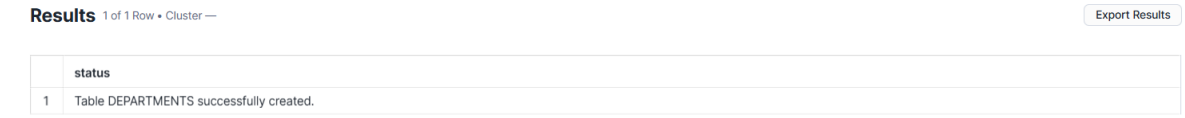
Assignment 1: Base Table Analysis

1.1) Load both datasets into Snowflake tables

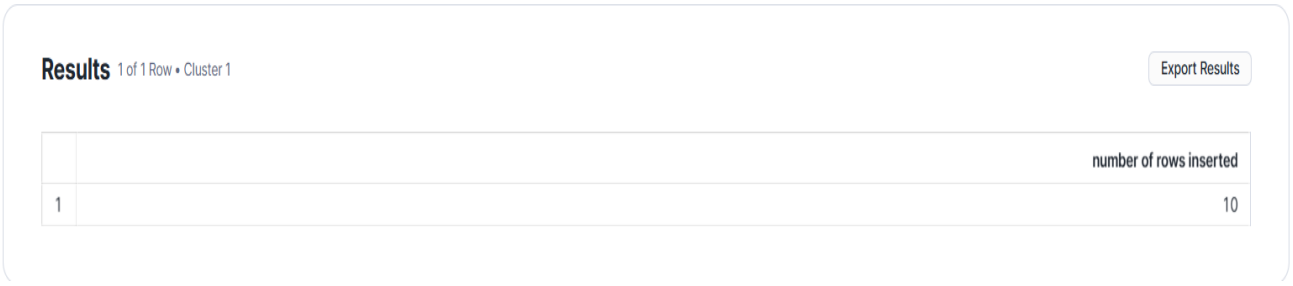
```
create table employee (emp_id int primary key, emp_name varchar (50),
department varchar (50),
job_title varchar (50),
salary int,location varchar (20), hire_date date);
```




```
create table departments
(dept_id int,department varchar (20),
manager varchar (50),
budget int,
location varchar (50));
```




```
INSERT INTO employee
(emp_id, emp_name, department, job_title, salary, location, hire_date)
VALUES
(101, 'Ramesh', 'IT', 'Data Engineer', 95000, 'Bangalore','2019-06-15'),
(102, 'Anita', 'HR', 'HR Executive', 65000, 'Hyderabad', '2020-02-10'),
(103, 'John', 'Finance', 'Finance Manager',110000, 'New York', '2016-09-01'),
(104, 'Meena', 'IT', 'Senior Data Analyst', 98000, 'Bangalore', '2018-11-23'),
(105, 'David', 'IT', 'Software Engineer', 90000, 'San Francisco', '2021-07-05'),
(106, 'Priya', 'HR', 'HR Business Partner', 72000, 'Hyderabad', '2019-03-18'),
(107, 'Michael', 'Finance', 'Account Director', 120000, 'New York', '2015-01-12'),
(108, 'Sneha', 'IT', 'Junior Developer', 78000, 'Bangalore', '2022-08-09'),
(109, 'Arjun', 'IT', 'Data Architect',115000, 'Bangalore', '2017-05-30'),
(110, 'Emily', 'Marketing', 'Marketing Manager', 88000, 'Chicago','2018-10-17');
```



```
INSERT INTO departments
(dept_id, department, manager, budget, location)
VALUES
(10, 'IT', 'Arjun Rao',500000, 'Bangalore'),
(20, 'HR', 'Anita Sharma', 200000, 'Hyderabad'),
(30, 'Finance', 'John Smith', 600000, 'New York'),
(40, 'Marketing', 'Emily Wilson', 300000, 'Chicago');
```

Results 1 of 1 Row • Cluster 1 Export Results

	number of rows inserted
1	4

1.2) Validate row counts and data distribution

```
select 'EMPLOYEE' AS TABLE_NAME, COUNT (*) AS ROW_COUNT FROM EMPLOYEE
UNION ALL
SELECT 'DEPARTMENTS' AS TABLE_NAME, COUNT (*) AS ROW_COUNT FROM DEPARTMENTS;
```

Results 2 of 2 Rows • Cluster 1 Export Results

	TABLE_NAME	ROW_COUNT
1	EMPLOYEE	10
2	DEPARTMENTS	4

```
SELECT
department, COUNT (*) AS emp_count,
AVG (salary) AS avg_salary
 FROM employee GROUP BY department   ORDER BY emp_count DESC;
```

Results 4 of 4 Rows • Cluster 1 Export Results

	DEPARTMENT	EMP_COUNT	AVG_SALARY
1	IT	5	95200.000000
2	HR	2	68500.000000
3	Finance	2	115000.000000
4	Marketing	1	88000.000000

select job_title,location,Avg(salary)as avg_salary from employee group by location,job_title,emp_name order by avg_salary Desc;

Results 10 of 10 Rows • Cluster 1 Export Results

	JOB_TITLE	LOCATION	AVG_SALARY
1	Account Director	New York	120000.000000
2	Data Architect	Bangalore	115000.000000
3	Finance Manager	New York	110000.000000
4	Senior Data Analyst	Bangalore	98000.000000
5	Data Engineer	Bangalore	95000.000000
6	Software Engineer	San Francisco	90000.000000
7	Marketing Manager	Chicago	88000.000000
8	Junior Developer	Bangalore	78000.000000
9	HR Business Partner	Hyderabad	72000.000000
10	HR Executive	Hyderabad	65000.000000

1.3) Identify candidate tables for cloning

Candidate table => the **original/source table** that we *choose* for cloning

Candidate tables were identified based on data availability, frequency of changes, business relevance, and suitability for observing Snowflake cloning behavior.

SELECT table_name, row_count

FROM TAB _ DB. INFORMATION _ SCHEMA.TABLES

WHERE table_schema = 'PUBLIC'

AND table_name IN ('EMPLOYEE', 'DEPARTMENTS');

INFORMATION_SCHEMA is a system schema that provides metadata about tables, schemas, and other database objects in Snowflake.

Res (14 hours ago) Results (just now)

Table Chart 2 rows 1.8s

	TABLE_NAME	ROW_COUNT
1	DEPARTMENTS	4
2	EMPLOYEE	10

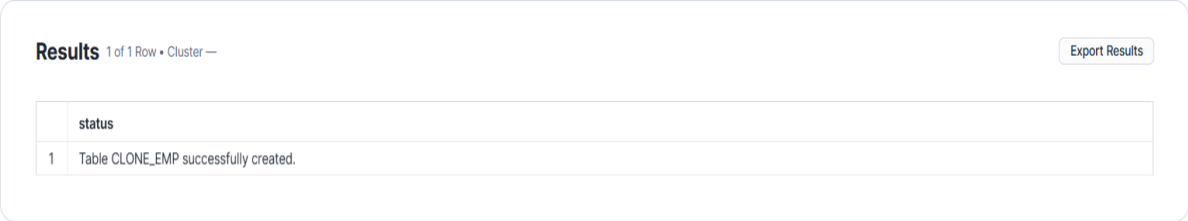
1.4) Explain why these tables are suitable for cloning experiments:

These tables are suitable for cloning because Employees supports frequent data changes while Departments represents stable reference data. Their size and business relevance make them ideal for demonstrating Snowflake's zero-copy cloning and isolation behavior.

Assignment 2: Table-Level Cloning

2.1) Clone the employee table

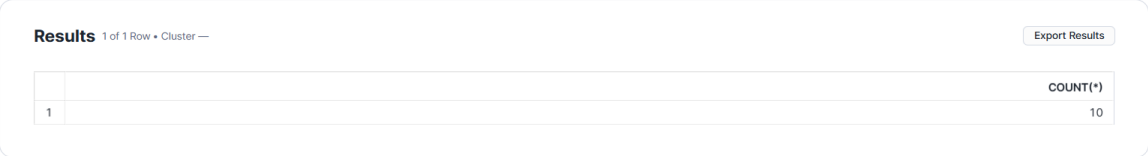
create table clone_emp clone employee;



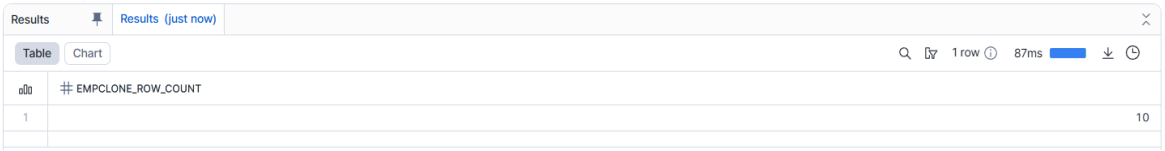
2.2) Compare:

Row count:

select count (*) from employee;



select count (*) from clone_emp;

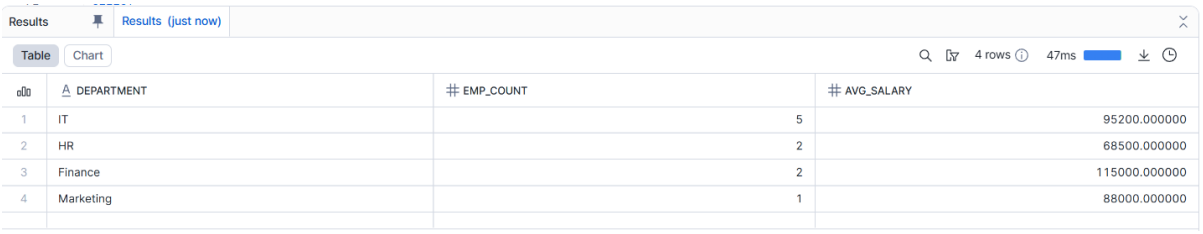


Query Perfomance:

SELECT department,COUNT(*) AS emp_count,AVG(salary) AS avg_salary

FROM employee GROUP BY department

ORDER BY emp_count DESC;



SELECT department,COUNT(*) AS emp_count,AVG(salary) AS avg_salary

FROM clone_emp GROUP BY department

ORDER BY emp_count DESC;

Results | Results (just now)

Table | Chart | 4 rows | 47ms

	DEPARTMENT	EMP_COUNT	AVG_SALARY
1	IT	5	95200.000000
2	HR	2	68500.000000
3	Finance	2	115000.000000
4	Marketing	1	88000.000000

Table | Chart | 133 rows | 4.1s

	QUERY_TEXT	EXECUTION_TIME	BYTES_SCANNED
	select department,avg(salary) as department_avgsal from employee group by department order by department_avgsal desc; 6.8% SELECT department, COUNT(*) AS emp_count, AVG(salary) AS avg_salary FROM employee GROUP BY department ORDER BY emp_co... 3.8% +64 more	0 1052	0 6144
1	SELECT department, COUNT(*) AS emp_count, AVG(salary) AS avg_salary FROM clone_emp GROUP BY department ORDER BY emp_count DESC;	1	0
2	SELECT department, COUNT(*) AS emp_count, AVG(salary) AS avg_salary FROM employee GROUP BY department ORDER BY emp_count DESC;	1	0

storage usage is checked to verify Snowflake's zero-copy cloning behavior. Identical storage values for the original and cloned tables confirm that no physical data is duplicated during clone creation. Storage increases only when data is modified, demonstrating the copy-on-write mechanism.

2.3) Explain why clone creation is instantaneous

Snowflake clone creation is instantaneous because only metadata is copied, not the actual data. Data is physically duplicated only when modifications occur through copy-on-write.

Assignment 3: Change Isolation Testing

3.1) Update salary data in the original employee table:

```
UPDATE employee

SET salary = salary + 1000

WHERE department = 'IT';
```

Results | Results (just now) | Download results

Table | Chart | 1 row | 1.8s

	number of rows updated	number of multi-joined rows updated
1	5	0

--validate

```
SELECT * FROM employee where DEPARTMENT='IT';
```

Results | Results (just now)

Table | Chart | 5 rows | 640ms

	EMP_ID	EMP_NAME	DEPARTMENT	JOB_TITLE	SALARY	LOCATION	HIRE_DATE
1	101	Ramesh	IT	Data Engineer	96000	Bangalore	2019-06-15
2	104	Meena	IT	Senior Data Analyst	99000	Bangalore	2018-11-23
3	105	David	IT	Software Engineer	91000	San Francisco	2021-07-05
4	108	Sneha	IT	Junior Developer	79000	Bangalore	2022-08-09
5	109	Arjun	IT	Data Architect	116000	Bangalore	2017-05-30

Eg:Before 95000 and after 96000

Select * from clone_emp WHERE DEPARTMENT='IT';

Results | Results (just now)

Table | Chart — 6 rows — 111ms

	EMP_ID	EMP_NAME	DEPARTMENT	JOB_TITLE	SALARY	LOCATION	HIRE_DATE
1	101	Ramesh	IT	Data Engineer	95000	Bangalore	2019-06-15
2	104	Meena	IT	Senior Data Analyst	98000	Bangalore	2018-11-23
3	105	David	IT	Software Engineer	90000	San Francisco	2021-07-05
4	108	Sneha	IT	Junior Developer	78000	Bangalore	2022-08-09
5	109	Arjun	IT	Data Architect	115000	Bangalore	2017-05-30
6	115	anuj	IT	Data Architect	115000	Bangalore	2017-05-30

Employee table IT department only changed

Its not affected in clone_emp

3.2) Delete records in the cloned table

DELETE FROM CLONE_EMP WHERE EMP_ID = 102;

Results 1 of 1 Row • Cluster 1 — Export Results

	number of rows deleted
1	1

Validate

select count(emp_id)from clone_emp where emp_id ='102' order by emp_id;

Results 1 of 1 Row • Cluster 1 — Export Results

	COUNT(EMP_ID)
1	0

SELECT count(emp_id) FROM employee where emp_id= '102' order by emp_id;

Results 1 of 1 Row • Cluster 1 — Export Results

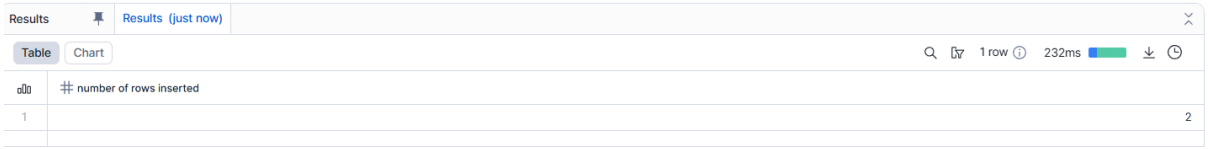
	COUNT(EMP_ID)
1	1

Once we delete the clone table data only deleted in clone data its not affected the primary table data.

3.3)Insert new records in the clone

insert into clone_emp values(115, 'anuj', 'IT', 'Data Architect', 115000, 'Bangalore', '2017-05-30'),

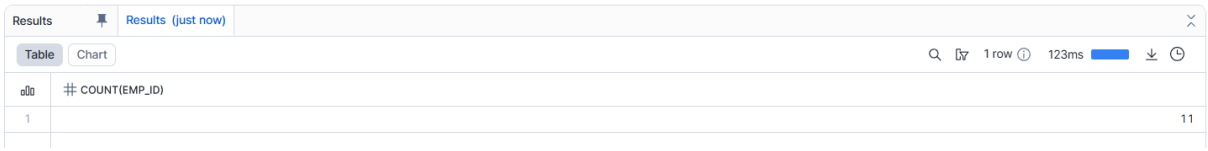
(116, 'miky', 'Marketing', 'Marketing Manager', 88000, 'Chicago', '2018-10-17');



--Validate

select count(emp_id) from employee;

select count(emp_id) from clone_emp;



I insert the data in clone table and successfully inserted but employee table is not affected.

3.3) Validate isolation between original and clone

Operation	Original	Clone
Updated in original table	Its Changed	Its not change
Updated in clone table	Its not affected	Its changed
Insert in original	Its inserted in original only	Not inserted in clone table
Insert in clone	its not inserted	Clone table only inserted
Delete in original	Only original table data is deleted	Clone table its not affected
Delete in clone	Its not changed	Its deleted

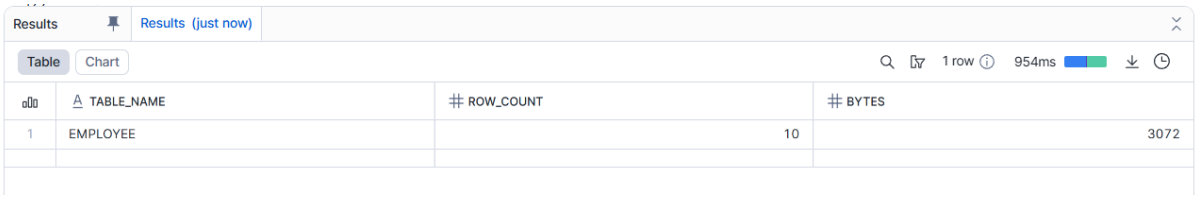
3.5) Explain backend copy-on-write behavior

Snowflake uses a copy-on-write mechanism for cloned objects. Initially, both the original and cloned tables share the same underlying storage. When a DML operation such as UPDATE, DELETE, or INSERT is performed, Snowflake creates new micro-partitions only for the modified data. Unchanged data continues to be shared. This ensures complete isolation between the original table and the clone while maintaining storage efficiency.

4: Storage Growth Exploration

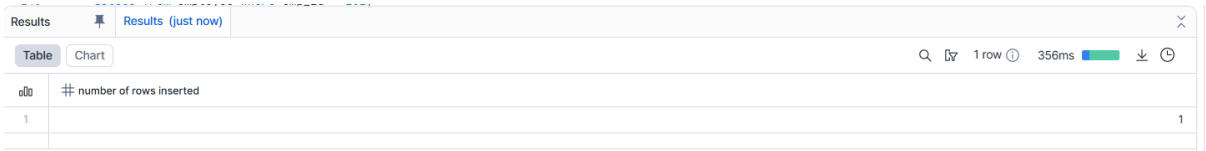
Track storage:

```
SELECT table_name,row_count,bytes

FROM TAB_DB. INFORMATION_SCHEMA.TABLES

WHERE table_schema = 'PUBLIC'

 AND table_name = 'EMPLOYEE';
```



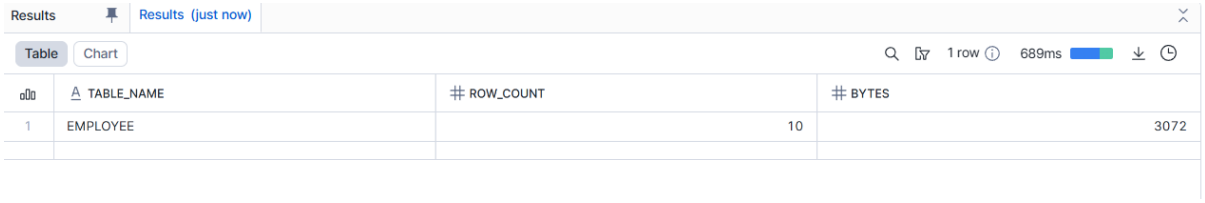
4.1) Perform DML operations incrementally:

```
insert into employee values (118, 'Emily', 'Marketing', 'Marketing Manager',88000, 'Chicago', '2018-10-17');

insert into employee values (119, 'Eoy', 'Marketing', 'Marketing Manager',88000, 'Chicago', '2018-10-17');
```



Performance Check:

```
SELECT table_name,row_count,bytes

FROM TAB_DB.INFORMATION_SCHEMA.TABLES

WHERE table_schema = 'PUBLIC'

 AND table_name = 'EMPLOYEE';
```



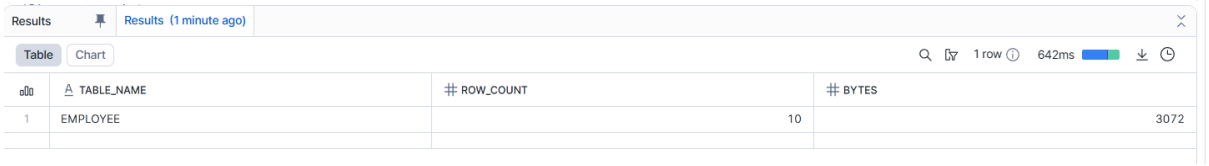
4.2)Track storage before and after each operation

Update the table:

```
update employee set salary= salary+10000 where emp_id =105;
```

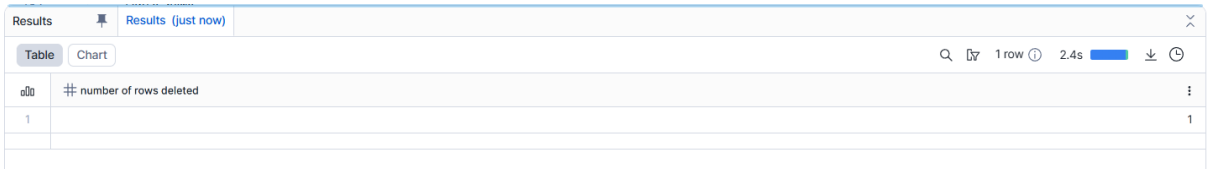
Track storage:

```
SELECT table_name,row_count, bytes
FROM TAB_DB.INFORMATION_SCHEMA.TABLES
WHERE table_schema = 'PUBLIC'
  AND table_name = 'EMPLOYEE';
```

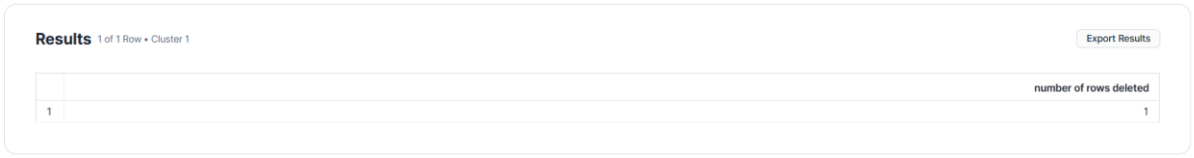


Delete the record in employee table:

```
delete from employee where emp_id = 102;
```

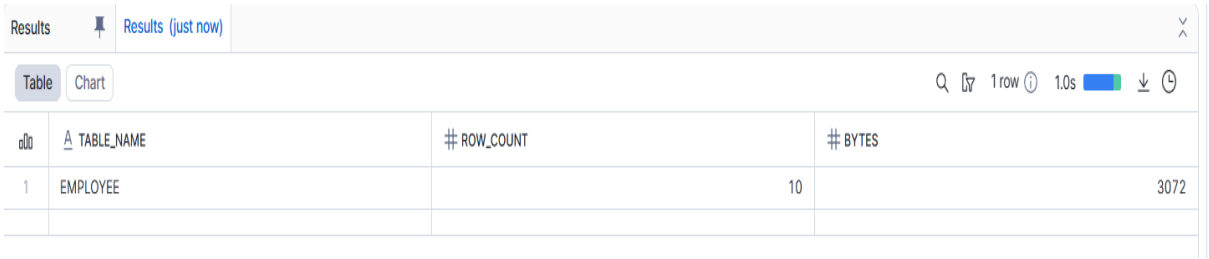



```
delete from employee where emp_id=103;
```



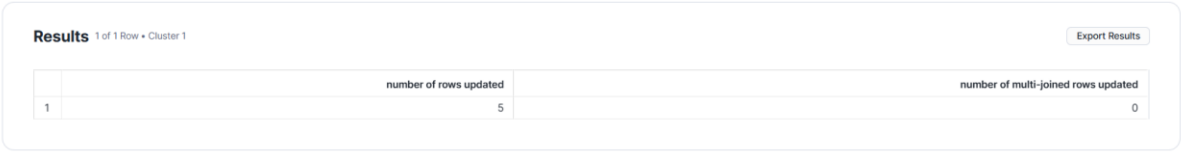
Track Storage:

```
SELECT table_name,row_count,bytes
FROM TAB_DB.INFORMATION_SCHEMA.TABLES
WHERE table_schema = 'PUBLIC'
AND table_name = 'EMPLOYEE';
```



Update the 5 Records:

```
UPDATE employee
SET salary = salary + 5000
WHERE emp_id IN (105, 106, 108, 109, 110);
```

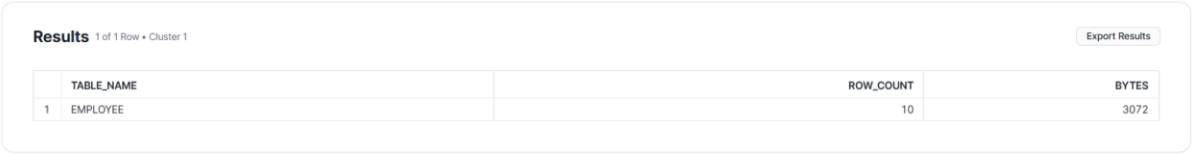


Results 1 of 1 Row • Cluster 1 — Export Results

	number of rows updated	number of multi-joined rows updated
1	5	0

Storage Track Progress:

```
SELECT table_name,row_count,
Bytes FROM TAB_DB.INFORMATION_SCHEMA.TABLES
WHERE table_schema = 'PUBLIC'
 AND table_name = 'EMPLOYEE';
```

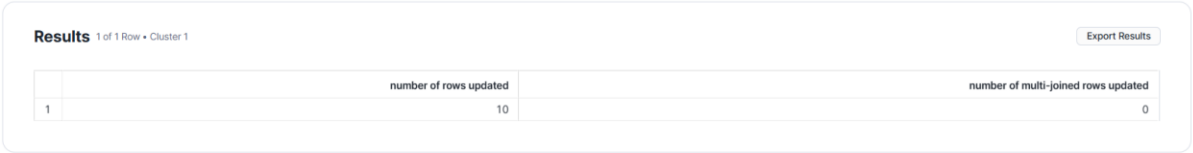


Results 1 of 1 Row • Cluster 1 — Export Results

	TABLE_NAME	ROW_COUNT	BYTES
1	EMPLOYEE	10	3072

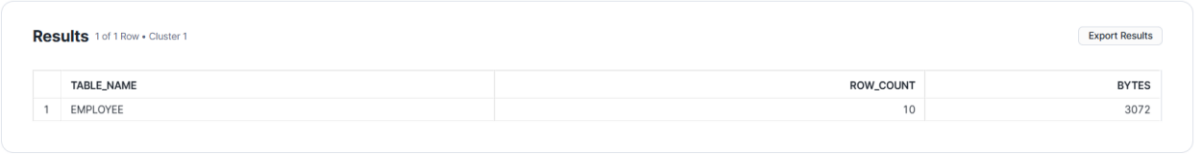
Update Employee Table:

```
 UPDATE employee
 SET salary = salary + 1000;
```



Results 1 of 1 Row • Cluster 1 — Export Results

	number of rows updated	number of multi-joined rows updated
1	10	0



Results 1 of 1 Row • Cluster 1 — Export Results

	TABLE_NAME	ROW_COUNT	BYTES
1	EMPLOYEE	10	3072

During the experiment, storage usage did not visibly increase because the table contained a very small dataset. Snowflake stores data in immutable micro-partitions, and small DML changes were absorbed within existing partitions without creating new ones.

4.3) Identify the exact moment storage increases

In Snowflake, storage growth occurs during UPDATE operations because the copy-on-write mechanism creates new micro-partitions instead of modifying existing data.

4.4) Explain Snowflake's physical storage strategy

Snowflake stores data in compressed, columnar **micro-partitions** with rich metadata.
Data is **immutable**, so changes create new micro-partitions instead of modifying existing ones.
This design enables **copy-on-write, zero-copy cloning, and Time Travel** with efficient storage use.

Assignment 5: Time Travel Cloning

5.1) Perform multiple updates on the original table

SELECT * FROM employee;

Results | Results (just now) | Table | Chart | 12 rows | 121ms

	EMP_ID (101–119)	EMP_NAME (Emily 16.7%, Eoy 16.7%, +8 more)	DEPARTMENT (IT 41.7%, Marketing 33.3%, +2 more)	JOB_TITLE (Marketing Manager 33.3%, Account Director 8.3%, +7 more)	SALARY (78000–121000)	LOCATION (Bangalore 33.3%, Chicago 33.3%, +3 more)	HIRE_DATE (1/12/2015–8/9/2022)
1	101	Ramesh	IT	Data Engineer	96000	Bangalore	2019-06-15
2	103	John	Finance	Finance Manager	111000	New York	2016-09-01
3	104	Meena	IT	Senior Data Analyst	99000	Bangalore	2018-11-23
4	107	Michael	Finance	Account Director	121000	New York	2015-01-12
5	105	David	IT	Software Engineer	106000	San Francisco	2021-07-05
6	106	Priya	HR	HR Business Partner	78000	Hyderabad	2019-03-18
7	108	Sneha	IT	Junior Developer	84000	Bangalore	2022-08-09

update employee set salary=salary+5000 where emp_id=101;

update employee set salary = salary +5000 where emp_id=104;

update employee set salary = salary +5000 where emp_id=119;

update employee set salary = salary +5000 where emp_id=110;

update employee set salary = salary +5000;

Results | Results (1 minute ago) | Table | Chart | 12 rows | 108ms

	EMP_ID (101–119)	EMP_NAME (Emily 16.7%, Eoy 16.7%, +8 more)	DEPARTMENT (IT 41.7%, Marketing 33.3%, +2 more)	JOB_TITLE (Marketing Manager 33.3%, Account Director 8.3%, +7 more)	SALARY (83000–126000)	LOCATION (Bangalore 33.3%, Chicago 33.3%, +3 more)	HIRE_DATE (1/12/2015–8/9/2022)
1	101	Ramesh	IT	Data Engineer	106000	Bangalore	2019-06-15
2	103	John	Finance	Finance Manager	116000	New York	2016-09-01
3	104	Meena	IT	Senior Data Analyst	109000	Bangalore	2018-11-23
4	107	Michael	Finance	Account Director	126000	New York	2015-01-12
5	105	David	IT	Software Engineer	111000	San Francisco	2021-07-05
6	106	Priya	HR	HR Business Partner	83000	Hyderabad	2019-03-18
7	108	Sneha	IT	Junior Developer	89000	Bangalore	2022-08-09

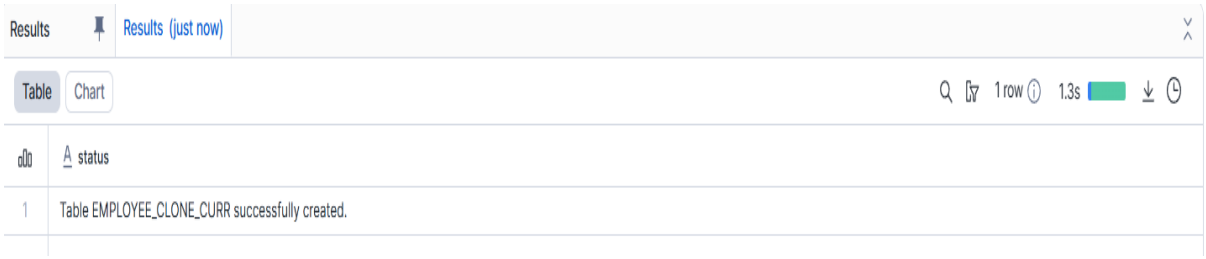
After Updated the multiple records

5.2)Create clones from:

Current state:

CREATE OR REPLACE TABLE employee_clone_curr --we update current records that clone table

CLONE employee;

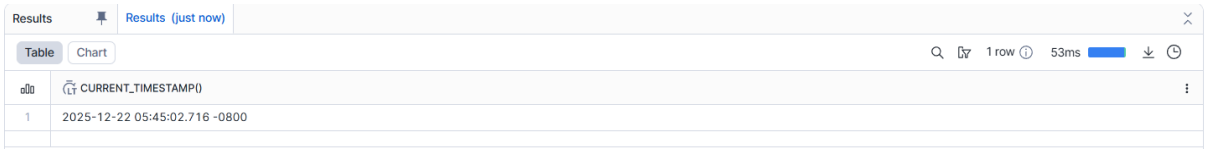


select*from employee_clone_curr;

Creating a clone from the current state means making an instant copy that reflects the latest data at the time of cloning.
It uses zero-copy cloning, so no data is physically duplicated and future changes stay isolated.

Past state (Time Travel)

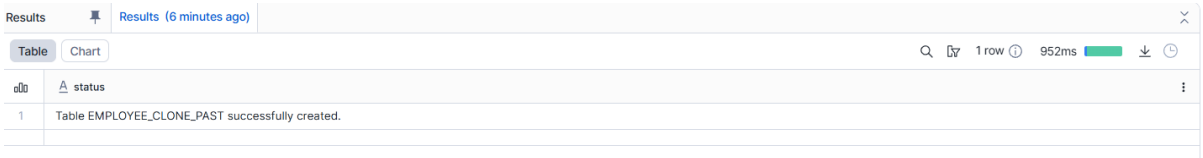
SELECT CURRENT_TIMESTAMP();



CREATE OR REPLACE TABLE employee_clone_past

CLONE employee

AT (TIMESTAMP => '2025-12-22 05:00:00'::TIMESTAMP_TZ);



Select * from employee_clone_past;

Results | Results (just now)

Table | Chart | 10 rows | 684ms

	EMP_ID	EMP_NAME	DEPARTMENT	JOB_TITLE	SALARY	LOCATION	HIRE_DATE
	101 – 110	Anita 10.0%, Arjun 10.0%, +8 more	IT 50.0%, Finance 20.0%, +2 more	Account Director 10.0%, Data Architect 10.0%, +8 more	65000 – 120000	Bangalore 40.0%, Hyderabad 20.0%, +3 more	1/12/2015 – 8/9/2022
1	101	Ramesh	IT	Data Engineer	95000	Bangalore	2019-06-15
2	102	Anita	HR	HR Executive	65000	Hyderabad	2020-02-10
3	103	John	Finance	Finance Manager	110000	New York	2016-09-01
4	104	Meena	IT	Senior Data Analyst	98000	Bangalore	2018-11-23
5	105	David	IT	Software Engineer	90000	San Francisco	2021-07-05
6	106	Priya	HR	HR Business Partner	72000	Hyderabad	2019-03-18

5.3) Compare data across versions

Original Table

SELECT * FROM employee ORDER BY emp_id;

Results | Results (just now)

Table | Chart | 12 rows | 30ms

	EMP_ID	EMP_NAME	DEPARTMENT	JOB_TITLE	SALARY	LOCATION	HIRE_DATE
	101 – 119	Emily 16.7%, Eoy 16.7%, +8 more	IT 41.7%, Marketing 33.3%, +2 more	Marketing Manager 33.3%, Account Director 8.3%, +7 more	83000 – 126000	Bangalore 33.3%, Chicago 33.3%, +3 more	1/12/2015 – 8/9/2022
1	101	Ramesh	IT	Data Engineer	106000	Bangalore	2019-06-15
2	103	John	Finance	Finance Manager	116000	New York	2016-09-01
3	104	Meena	IT	Senior Data Analyst	109000	Bangalore	2018-11-23
4	105	David	IT	Software Engineer	111000	San Francisco	2021-07-05
5	106	Priya	HR	HR Business Partner	83000	Hyderabad	2019-03-18
6	107	Michael	Finance	Account Director	126000	New York	2015-01-12
7	108	Sneha	IT	Junior Developer	89000	Bangalore	2022-08-09

SELECT * FROM employee_clone_curr ORDER BY emp_id;

Updated data its only available

Results | Results (just now)

Table | Chart | 12 rows | 27ms

	EMP_ID	EMP_NAME	DEPARTMENT	JOB_TITLE	SALARY	LOCATION	HIRE_DATE
	101 – 119	Emily 16.7%, Eoy 16.7%, +8 more	IT 41.7%, Marketing 33.3%, +2 more	Marketing Manager 33.3%, Account Director 8.3%, +7 more	83000 – 126000	Bangalore 33.3%, Chicago 33.3%, +3 more	1/12/2015 – 8/9/2022
1	101	Ramesh	IT	Data Engineer	106000	Bangalore	2019-06-15
2	103	John	Finance	Finance Manager	116000	New York	2016-09-01
3	104	Meena	IT	Senior Data Analyst	109000	Bangalore	2018-11-23
4	105	David	IT	Software Engineer	111000	San Francisco	2021-07-05
5	106	Priya	HR	HR Business Partner	83000	Hyderabad	2019-03-18
6	107	Michael	Finance	Account Director	126000	New York	2015-01-12
7	108	Sneha	IT	Junior Developer	89000	Bangalore	2022-08-09

SELECT * FROM employee_clone_past ORDER BY emp_id;

Before updated data its only using Timestamp

Results | Results (just now)

Table | Chart | 10 rows | 63ms

	EMP_ID	EMP_NAME	DEPARTMENT	JOB_TITLE	SALARY	LOCATION	HIRE_DATE
	101 – 110	Anita 10.0%, Arjun 10.0%, +8 more	IT 50.0%, Finance 20.0%, +2 more	Account Director 10.0%, Data Architect 10.0%, +8 more	65000 – 120000	Bangalore 40.0%, Hyderabad 20.0%, +3 more	1/12/2015 – 8/9/2022
1	101	Ramesh	IT	Data Engineer	95000	Bangalore	2019-06-15
2	102	Anita	HR	HR Executive	65000	Hyderabad	2020-02-10
3	103	John	Finance	Finance Manager	110000	New York	2016-09-01
4	104	Meena	IT	Senior Data Analyst	98000	Bangalore	2018-11-23
5	105	David	IT	Software Engineer	90000	San Francisco	2021-07-05
6	106	Priya	HR	HR Business Partner	72000	Hyderabad	2019-03-18
7	107	Michael	Finance	Account Director	120000	New York	2015-01-12

5.4) Explain real-world recovery use cases

Real-time Use Case 1: Accidental Update

Production-la mistake-aa salary update aagiduchu na:

Time Travel clone eduthu

Correct data recover pannalaam

Use Case 2: Data Validation

New transformation apply panna munna:

Old data clone eduthu

Compare pannalaam

Assignment 6: Schema-Level Cloning

6.1) Clone a schema containing employee and department tables

show tables in schema tab_db.public;

Results | Results (just now)

Table | Chart | 13 rows | 109ms

	created_on	name	database_name	schema_name	kind	comment	cluster_by	rows	bytes	owner
	12/18/2025 – 12/21/2025	CLON... 7.7%, CLON... 7.7%, +11 more	TAB_DB 100.0%	PUBLIC 100.0%	... 92.3%, ... 7.7%	100.0%	100.0%	4 – 12	15... – 30...	ACCOUNT... 100.
1	2025-12-21 04:13:26.150 -080(	CLONE_EMP	TAB_DB	PUBLIC	TABLE			11	3072	ACCOUNTADMIN
2	2025-12-19 03:12:45.429 -080(	CLONE_TAB	TAB_DB	PUBLIC	TABLE			6	2048	ACCOUNTADMIN
3	2025-12-18 21:50:36.239 -080(	CLONE_TB	TAB_DB	PUBLIC	TABLE			12	2560	ACCOUNTADMIN
4	2025-12-21 04:35:40.433 -080(	CLP	TAB_DB	PUBLIC	TABLE			10	3072	ACCOUNTADMIN
5	2025-12-21 03:26:06.105 -080(	DEPARTMENTS	TAB_DB	PUBLIC	TABLE			4	2560	ACCOUNTADMIN
6	2025-12-21 03:24:16.991 -080(	EMPLOYEE	TAB_DB	PUBLIC	TABLE			10	3072	ACCOUNTADMIN

create or replace schema tab_db.clone_pub_schema clone tab_db.public;

Results | Results (just now)

Table | Chart | 1 row | 5.3s

	status
1	Schema CLONE_PUB_SCHEMA successfully created.

SHOW TABLES IN SCHEMA TAB_DB.clone_pub_schema;

Table | Chart — 13 rows — 49ms

	created_on	name	database_name	schema_name	kind	comment	cluster_by	rows	bytes	ow
	12/22/2025 12/22/2025	CLON... 7.7% CLON... 7.7% +11 more	TAB_DB 100.0%	CLONE_PUB_S... 100.0%	... 92.3% ... 7.7%	100.0%	100.0%	4 12	15...30...	ACCO
1	2025-12-22 06:07:39.930 -0800	CLONE_EMP	TAB_DB	CLONE_PUB_SCHEMA	TABLE			11	3072	ACCO
2	2025-12-22 06:07:39.930 -0800	CLONE_TAB	TAB_DB	CLONE_PUB_SCHEMA	TABLE			6	2048	ACCO
3	2025-12-22 06:07:39.930 -0800	CLONE_TB	TAB_DB	CLONE_PUB_SCHEMA	TABLE			12	2560	ACCO
4	2025-12-22 06:07:39.930 -0800	CLP	TAB_DB	CLONE_PUB_SCHEMA	TABLE			10	3072	ACCO
5	2025-12-22 06:07:39.930 -0800	DEPARTMENTS	TAB_DB	CLONE_PUB_SCHEMA	TABLE			4	2560	ACCO
6	2025-12-22 06:07:39.930 -0800	EMPLOYEE	TAB_DB	CLONE_PUB_SCHEMA	TABLE			10	3072	ACCO

6.2)Modify tables in cloned schema

USE SCHEMA TAB_DB.clone_PUB_schema;

UPDATE departments

SET budget = budget+ 300000

WHERE dept_id = 20;

Results | Results (just now)

Table | Chart — 1 row — 1.7s

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

INSERT INTO departments VALUES (201, 'IT', 'Arjun Rao', 500000, 'Bangalore');

Results | Results (just now)

Table | Chart — 1 row — 1.1s

	number of rows inserted
1	1

select*from departments;

Results | Results (just now)

Table | Chart — 5 rows — 115ms

	DEPT_ID	DEPARTMENT	MANAGER	BUDGET	LOCATION
1	10	IT	Arjun Rao	500000	Bangalore
2	20	HR	Anita Sharma	500000	Hyderabad
3	30	Finance	John Smith	600000	New York
4	40	Marketing	Emily Wilson	300000	Chicago
5	201	IT	Arjun Rao	500000	Bangalore

6.3)Drop tables in original schema

DROP TABLE TAB_DB.PUBLIC.DEPARTMENTS;

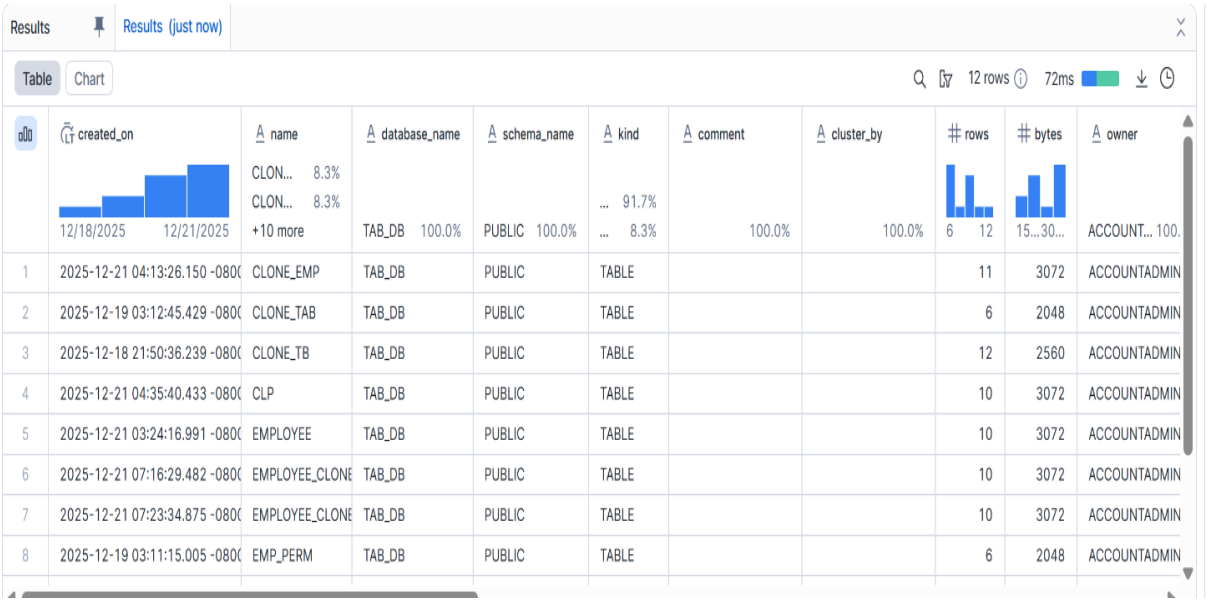
Results | Results (just now)

Table | Chart — 1 row — 89ms

	status
1	DEPARTMENTS successfully dropped.

SHOW TABLES IN SCHEMA TAB_DB.PUBLIC;

Its not show



	created_on	name	database_name	schema_name	kind	comment	cluster_by	rows	bytes	owner
	12/18/2025 12/21/2025	CLON... 8.3% CLON... 8.3% +10 more	TAB_DB 100.0%	PUBLIC 100.0%	... 91.7% ... 8.3%	100.0%	100.0%	6 12	15...30...	ACCOUNT... 100.
1	2025-12-21 04:13:26.150 -0800	CLONE_EMP	TAB_DB	PUBLIC	TABLE			11	3072	ACCOUNTADMIN
2	2025-12-19 03:12:45.429 -0800	CLONE_TAB	TAB_DB	PUBLIC	TABLE			6	2048	ACCOUNTADMIN
3	2025-12-18 21:50:36.239 -0800	CLONE_TB	TAB_DB	PUBLIC	TABLE			12	2560	ACCOUNTADMIN
4	2025-12-21 04:35:40.433 -0800	CLP	TAB_DB	PUBLIC	TABLE			10	3072	ACCOUNTADMIN
5	2025-12-21 03:24:16.991 -0800	EMPLOYEE	TAB_DB	PUBLIC	TABLE			10	3072	ACCOUNTADMIN
6	2025-12-21 07:16:29.482 -0800	EMPLOYEE_CLONE	TAB_DB	PUBLIC	TABLE			10	3072	ACCOUNTADMIN
7	2025-12-21 07:23:34.875 -0800	EMPLOYEE_CLONE	TAB_DB	PUBLIC	TABLE			10	3072	ACCOUNTADMIN
8	2025-12-19 03:11:15.005 -0800	EMP_PERM	TAB_DB	PUBLIC	TABLE			6	2048	ACCOUNTADMIN

SELECT * FROM TAB_DB.clone_pub_schema.departments;

	DEPT_ID	DEPARTMENT	MANAGER	BUDGET	LOCATION
1	10	IT	Arjun Rao	500000	Bangalore
2	20	HR	Anita Sharma	500000	Hyderabad
3	30	Finance	John Smith	600000	New York
4	40	Marketing	Emily Wilson	300000	Chicago
5	201	IT	Arjun Rao	500000	Bangalore

SELECT * FROM TAB_DB.public.departments;

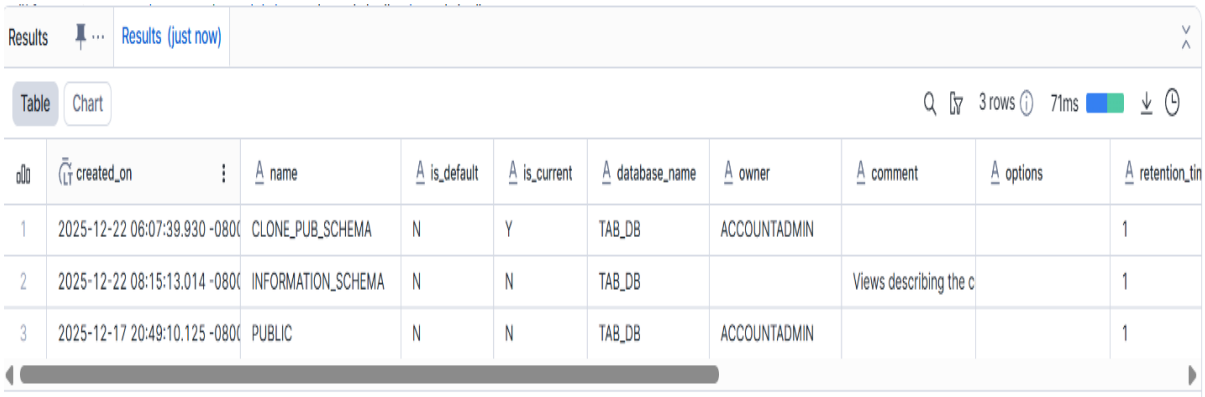
6.4) Observe behavior and explain metadata inheritance

Schema-level cloning in Snowflake creates a logical copy of all tables using metadata inheritance. The cloned schema initially shares the same underlying data. When data is modified in the cloned schema, Snowflake uses copy-on-write to create new micro-partitions. Dropping tables in the original schema does not impact the cloned schema because each clone maintains independent metadata references.

Assignment 7: Database-Level Cloning

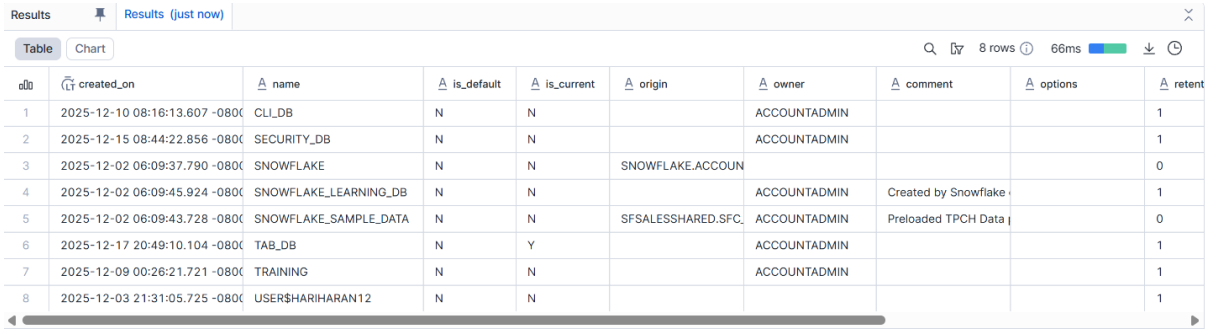
7.1) Clone the entire database

show schemas in database tab_db;



	created_on	name	is_default	is_current	database_name	owner	comment	options	retention_tin
1	2025-12-22 06:07:39.930 -080(	CLONE_PUB_SCHEMA	N	Y	TAB_DB	ACCOUNTADMIN			1
2	2025-12-22 08:15:13.014 -080(	INFORMATION_SCHEMA	N	N	TAB_DB		Views describing the c		1
3	2025-12-17 20:49:10.125 -080(	PUBLIC	N	N	TAB_DB	ACCOUNTADMIN			1

show databases;



	created_on	name	is_default	is_current	origin	owner	comment	options	retent
1	2025-12-10 08:16:13.607 -080(	CLI_DB	N	N		ACCOUNTADMIN			1
2	2025-12-15 08:44:22.856 -080(	SECURITY_DB	N	N		ACCOUNTADMIN			1
3	2025-12-02 06:09:37.790 -080(	SNOWFLAKE	N	N	SNOWFLAKE.ACCOUN				0
4	2025-12-02 06:09:45.924 -080(	SNOWFLAKE_LEARNING_DB	N	N		ACCOUNTADMIN	Created by Snowflake		1
5	2025-12-02 06:09:43.728 -080(	SNOWFLAKE_SAMPLE_DATA	N	N	SFSALESSHARED.SFC_	ACCOUNTADMIN	Preloaded TPCH Data		0
6	2025-12-17 20:49:10.104 -080(	TAB_DB	N	Y		ACCOUNTADMIN			1
7	2025-12-09 00:26:21.721 -080(	TRAINING	N	N		ACCOUNTADMIN			1
8	2025-12-03 21:31:05.725 -080(	USER$HARIHARAN12	N	N					1

create or replace database clone_tab_db clone tab_db;

	status
1	Database CLONE_TAB_DB successfully created.

7.2)Test query access and object availability

USE DATABASE clone_tab_db;

SELECT * FROM PUBLIC.EMPLOYEE;

Results | Results (7 minutes ago)

Table | Chart | 10 rows | 150ms

	EMP_ID	EMP_NAME	DEPARTMENT	JOB_TITLE	SALARY	LOCATION	HIRE_DATE
	101 – 119	Emily 20.0%, Arjun 10.0%, +7 more	IT 50.0%, Marketing 30.0%, +2 more	Marketing Manager 30.0%, Account Director 10.0%, +6 more	83000 – 127000	Bangalore 40.0%, Chicago 30.0%, +3 more	1/12/2015 – 8/9/2022
1	107	Michael	Finance	Account Director	126000	New York	2015-01-12
2	101	Ramesh	IT	Data Engineer	107000	Bangalore	2019-06-15
3	104	Meena	IT	Senior Data Analyst	110000	Bangalore	2018-11-23
4	106	Priya	HR	HR Business Partner	83000	Hyderabad	2019-03-18
5	105	David	IT	Software Engineer	112000	San Francisco	2021-07-05
6	108	Sneha	IT	Junior Developer	90000	Bangalore	2022-08-09
7	109	Arjun	IT	Data Architect	127000	Bangalore	2017-05-30

7.3) Perform DML on cloned database

UPDATE clone_tab_db.PUBLIC.EMPLOYEE

SET salary = salary + 3000 WHERE emp_id = 104;

Results | Results (just now)

Table | Chart | 1 row | 987ms

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

Clone database Its perform DML commends

insert into clone_tab_db.public.employee values(301, 'Ramesh', 'IT', 'Data Engineer', 95000, 'Bangalore', '2019-06-15');

Results | Results (just now)

Table | Chart | 1 row | 773ms

	number of rows inserted
1	1

SELECT * FROM TAB_DB.PUBLIC.EMPLOYEE WHERE emp_id = 104; --- 110000 primary database table is not updated

Results | Results (just now)

Table | Chart | 1 row | 71ms

	EMP_ID	EMP_NAME	DEPARTMENT	JOB_TITLE	SALARY	LOCATION	HIRE_DATE
1	104	Meena	IT	Senior Data Analyst	110000	Bangalore	2018-11-23

SELECT*FROM CLONE_TAB_DB.PUBLIC.EMPLOYEE WHERE EMP_ID=104

Results | Results (just now)

Table | Chart | 1 row | 1.4s

	EMP_ID	EMP_NAME	DEPARTMENT	JOB_TITLE	SALARY	LOCATION	HIRE_DATE
1	104	Meena	IT	Senior Data Analyst	113000	Bangalore	2018-11-23

--113000 ITS CHANGED SO CLONE DATABASE ITS CHANGED

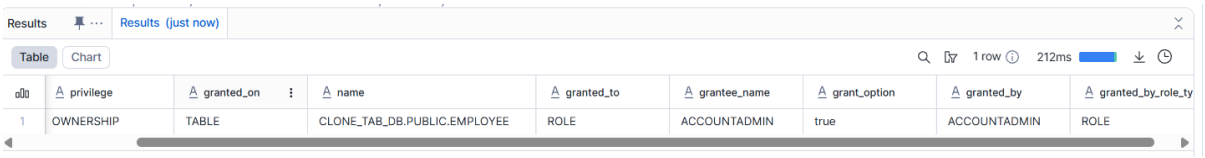
7.4)Explain how Snowflake isolates environments (Dev/Test/Prod)

Snowflake isolates Dev, Test, and Prod environments using database-level cloning. Each cloned database inherits metadata and shares the same underlying data initially. When DML operations are performed, Snowflake uses copy-on-write to isolate changes, ensuring that modifications in one environment do not impact others.

Assignment 8: Privileges & Security

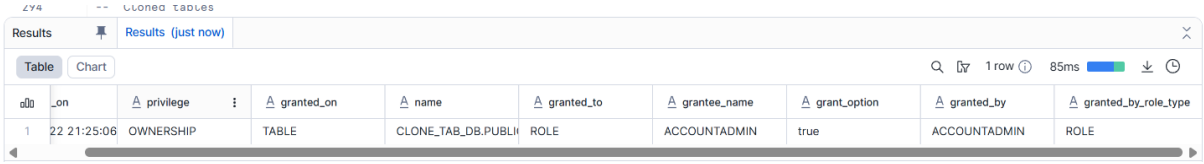
8.1)Analyze access privileges before cloning

SHOW GRANTS ON TABLE EMPLOYEE;



8.2) Observe privileges after cloning

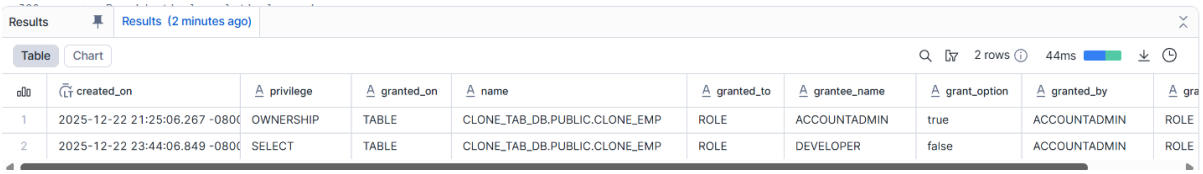
SHOW GRANTS ON TABLE CLONE_EMP;



8.3)Modify privileges on clone

SHOW GRANTS ON TABLE CLONE_EMP;

We checked the privileges on the EMPLOYEE table and saw which roles had access, like SELECT for the ANALYST role.
After cloning the table, those privileges were not copied automatically.
Only the owner role has full access on the clone, and any other access must be granted manually.
Changes made on the clone do not affect the original table's privileges.



8.4) Explain why Snowflake handles security this way

Snowflake handles security very strictly. Even though cloned tables share the same data, access privileges are not shared. This prevents unauthorized users from accessing cloned data. This design follows the least-privilege principle and helps keep production data secure.

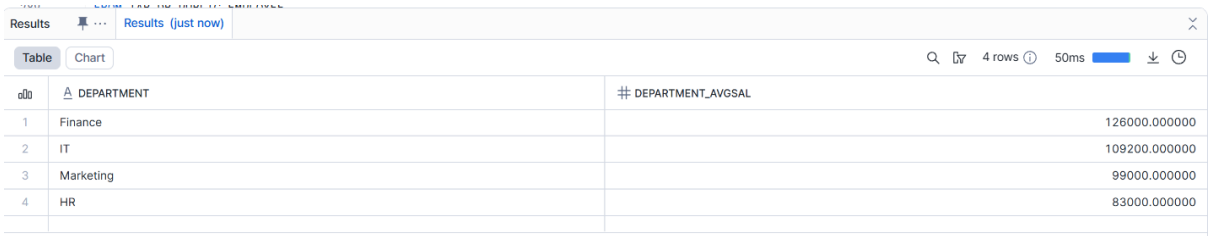
Assignment 9: Performance Comparison

9.1)Run identical analytical queries on:

Original tables:

use tab_db;

select department,avg(salary) as department_avgsal from employee group by department order by department_avgsal desc;



Results | Results (just now)

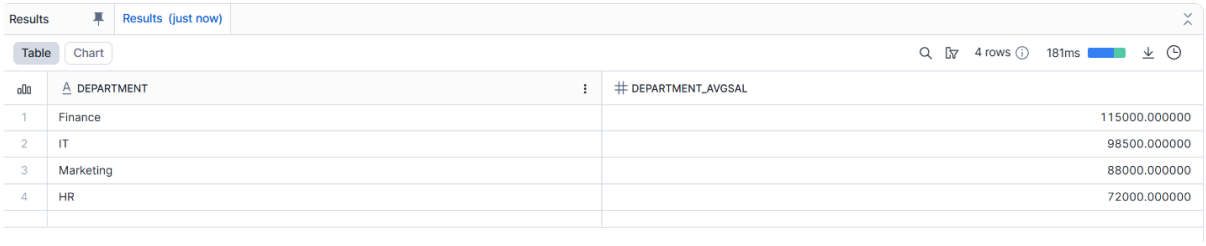
Table | Chart — 4 rows — 50ms

	DEPARTMENT	DEPARTMENT_AVGSAL
1	Finance	126000.000000
2	IT	109200.000000
3	Marketing	99000.000000
4	HR	83000.000000

Cloned tables:

use clone_tab_db;

select department,avg(salary) as department_avgsal from employee group by department order by department_avgsal desc;



Results | Results (just now)

Table | Chart — 4 rows — 181ms

	DEPARTMENT	DEPARTMENT_AVGSAL
1	Finance	115000.000000
2	IT	98500.000000
3	Marketing	88000.000000
4	HR	72000.000000

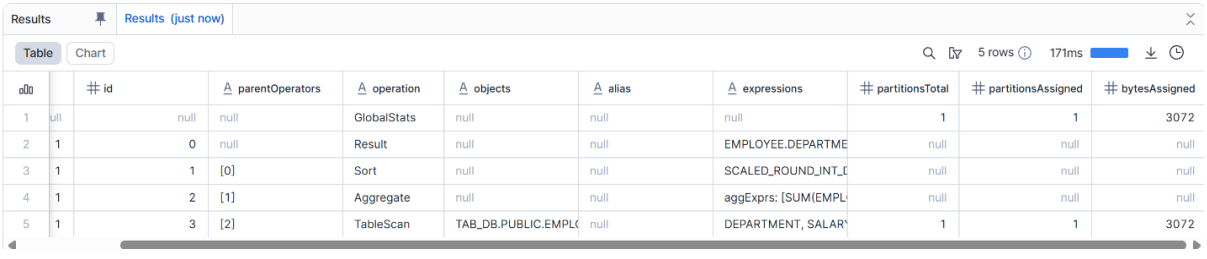
9.2)Compare execution plans:

EXPLAIN

SELECT department, AVG(salary)as department_avgsal

FROM TAB_DB.PUBLIC.EMPLOYEE

GROUP BY department order by department_avgsal desc;



Results | Results (just now)

Table | Chart — 5 rows — 171ms

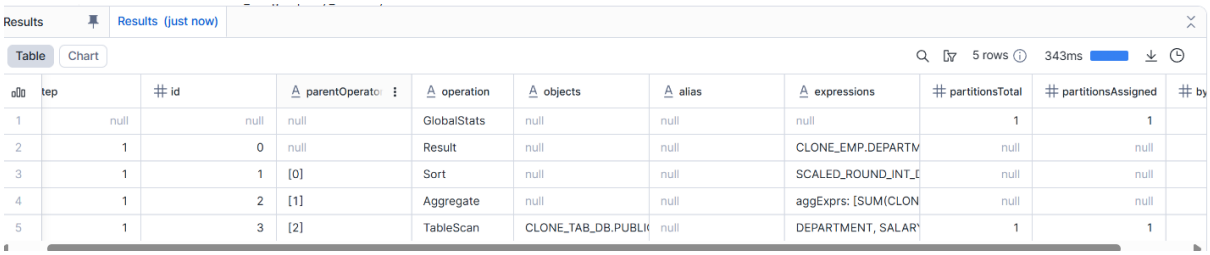
	id	parentOperators	operation	objects	alias	expressions	partitionsTotal	partitionsAssigned	bytesAssigned
1	null	null	GlobalStats	null	null	null	1	1	3072
2	0	null	Result	null	null	EMPLOYEE.DEPARTME	null	null	null
3	1	[0]	Sort	null	null	SCALED_ROUND_INT_D	null	null	null
4	2	[1]	Aggregate	null	null	aggExprs: [SUM(EMPL	null	null	null
5	3	[2]	TableScan	TAB_DB.PUBLIC.EMPLO	null	DEPARTMENT, SALAR	1	1	3072

EXPLAIN

SELECT department, AVG(salary)as department_avgsal

FROM clone_TAB_DB.PUBLIC.clone_EMP

GROUP BY department order by department_avgsal desc;



	tep	id	parentOperator	operation	objects	alias	expressions	partitionsTotal	partitionsAssigned	by
1	null	null	null	GlobalStats	null	null	null	1	1	
2	1	0	null	Result	null	null	CLONE_EMP.DEPARTM	null	null	
3	1	1	[0]	Sort	null	null	SCALED_ROUND_INT_D	null	null	
4	1	2	[1]	Aggregate	null	null	aggExprs: [SUM(CLON	null	null	
5	1	3	[2]	TableScan	CLONE_TAB_DB.PUBLI	null	DEPARTMENT, SALAR	1	1	

```
SELECT
    query_text,
    execution_time,
    bytes_scanned
FROM snowflake.account_usage.query_history
WHERE query_text ILIKE '%AVG(salary)%'
ORDER BY start_time DESC;
```

9.3)explain performance similarities/differences

Clone tables:

Same micro-partitions share pannum

Same metadata structure

Snowflake optimizer:

Same execution plan generate pannum

Assignment 10: Cleanup & Retention

10.1) Drop original objects

```
DROP TABLE TAB_DB.PUBLIC.EMPLOYEE;
```

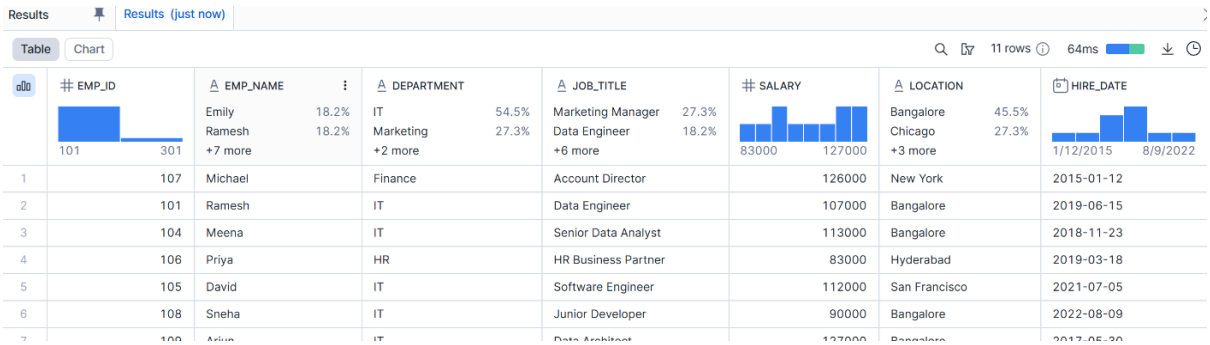
	status
1	EMPLOYEE successfully dropped.

select*from tab_db.public.employee; -----doesnt exist

10.2)Validate cloned object availability

SELECT * FROM CLONE_tab_db.PUBLIC.EMPLOYEE;

-- i droped original table but showing clone table no affect



	EMP_ID	EMP_NAME	DEPARTMENT	JOB_TITLE	SALARY	LOCATION	HIRE_DATE
	101 – 301	Emily 18.2%, Ramesh 18.2%, +7 more	IT 54.5%, Marketing 27.3%, +2 more	Marketing Manager 27.3%, Data Engineer 18.2%, +6 more	83000 – 127000	Bangalore 45.5%, Chicago 27.3%, +3 more	1/12/2015 – 8/9/2022
1	107	Michael	Finance	Account Director	126000	New York	2015-01-12
2	101	Ramesh	IT	Data Engineer	107000	Bangalore	2019-06-15
3	104	Meena	IT	Senior Data Analyst	113000	Bangalore	2018-11-23
4	106	Priya	HR	HR Business Partner	83000	Hyderabad	2019-03-18
5	105	David	IT	Software Engineer	112000	San Francisco	2021-07-05
6	108	Sneha	IT	Junior Developer	90000	Bangalore	2022-08-09

10.3) Drop clone and analyze Time Travel behavior

DROP TABLE CLONE_tab_db.PUBLIC.EMPLOYEE;

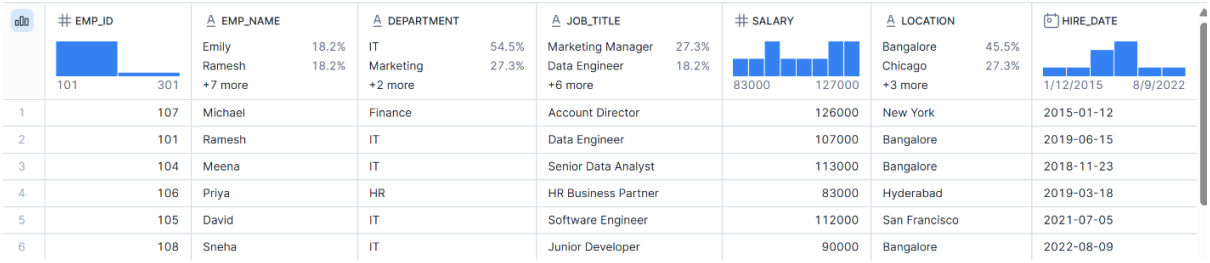
SELECT * FROM CLONE_tab_db.PUBLIC.EMPLOYEE; --- its showing doent exist

Recover Clone Using Time Travel (UNDROP)

undrop table clone_tab_db.public.employee;

--Verify Data After UNDROP:

SELECT * FROM CLONE_tab_db.PUBLIC.EMPLOYEE; ---- its showing output



	EMP_ID	EMP_NAME	DEPARTMENT	JOB_TITLE	SALARY	LOCATION	HIRE_DATE
	101 – 301	Emily 18.2% Ramesh 18.2% +7 more	IT 54.5% Marketing 27.3% +2 more	Marketing Manager 27.3% Data Engineer 18.2% +6 more	83000 – 127000	Bangalore 45.5% Chicago 27.3% +3 more	1/12/2015 – 8/9/2022
1	107	Michael	Finance	Account Director	126000	New York	2015-01-12
2	101	Ramesh	IT	Data Engineer	107000	Bangalore	2019-06-15
3	104	Meena	IT	Senior Data Analyst	113000	Bangalore	2018-11-23
4	106	Priya	HR	HR Business Partner	83000	Hyderabad	2019-03-18
5	105	David	IT	Software Engineer	112000	San Francisco	2021-07-05
6	108	Sneha	IT	Junior Developer	90000	Bangalore	2022-08-09

10.4) Explain lifecycle and retention rules

Dropped cloned objects can be recovered within the Time Travel period, after which data moves to Fail-safe and is eventually permanently deleted.