

Snowflake Time Travel, Cloning and Cache Assignment

1. Time Travel :-

- Time travel is Retention time that means the time we can travel back into the past in snowflake this depends on the edition we use.
- This is basically the property for objects, for example if we take table, this is specifying for how much time we can travel back for this object.
- So, the default value here is usually set to one.
- So, if the table has its retention period property, set to one then we can travel back for 24 hours for this table.

Now I am taking a sample table called Products :-

Creating a table called products mentioning retention time as 30 days.

```
CREATE TABLE products (  
  product_id INT PRIMARY KEY,  
  product_name VARCHAR(100),  
  category VARCHAR(50),  
  manufacturer VARCHAR(100),  
  price INT) data_retention_time_in_days = 30;
```

The screenshot displays the Snowflake web interface. The top navigation bar includes the URL 'app.snowflake.com/nywaxdr/lo50872/w8BjgAbcBsk#query' and various icons. The left sidebar shows a 'Databases' tab with a tree view of objects, including 'PRODUCTS'. The main area shows the SQL query: `CREATE TABLE products (product_id INT PRIMARY KEY, product_name VARCHAR(100), category VARCHAR(50), manufacturer VARCHAR(100), price INT) data_retention_time_in_days = 30;`. Below the query, the 'Results' tab is active, showing a single row with the status 'Table PRODUCTS successfully created.' The right sidebar displays 'Query Details' with a duration of 215ms, 1 row, and a query ID of 01b05999-3200-ef60-...

Now I have inserted the data into the table with 5 records and now I am using show tables query and the retention time can be seen here

VEDA_DB.VEDA_SH1

```

13 11, 'Laptop', 'Electronics', 'ABC Electronics', 12000,
14 (2, 'Smartphone', 'Electronics', 'XYZ Tech', 800),
15 (3, 'Coffee Maker', 'Appliances', 'Home Appliances Co.', 50),
16 (4, 'Running Shoes', 'Footwear', 'SportGear Inc.', 80),
17 (5, 'Desk Chair', 'Furniture', 'Comfort Furnishings', 150);
18
19
20
21 //select statement to retrieve the data
22 select * from products;
23
24 show tables;
25
26
27
28
29
30
31
32
33
34

```

#	name	database_name	schema_name	kind	comment	cluster_by	rows	bytes	owner	retention_time
2	COUNTRY_TABLE	VEDA_DB	VEDA_SH1	TABLE			249	4,095	SYSADMIN	1
3	CREATE_DAY_TABLE_USING_LOAD_WIZARD	VEDA_DB	VEDA_SH1	TABLE			32	2,048	SYSADMIN	1
4	DAY_TABLE	VEDA_DB	VEDA_SH1	TABLE			31	1,538	ACCOUNTADMIN	1
5	DAY_TABLE1	VEDA_DB	VEDA_SH1	TABLE			0	0	SYSADMIN	1
6	EXTERNAL_TABLE	VEDA_DB	VEDA_SH1	TABLE			0	0	MENTOR	1
7	PERMANENT_TABLE	VEDA_DB	VEDA_SH1	TABLE			0	0	MENTEES	1
8	PERMANENT_TABLE2	VEDA_DB	VEDA_SH1	TABLE			93	1,538	SYSADMIN	1
9	PRODUCTS	VEDA_DB	VEDA_SH1	TABLE			5	2,560	SYSADMIN	30
10	TRANSIENT_TABLE	VEDA_DB	VEDA_SH1	TRANSIENT			0	0	MENTEES	1
11	TRANSIENT_TABLE_1	VEDA_DB	VEDA_SH1	TABLE			31	1,538	SYSADMIN	1

Now I am dropping the table products using drop table command

VEDA_DB.VEDA_SH1

```

19
20
21 //select statement to retrieve the data
22 select * from products;
23
24 show tables;
25
26 drop table products;
27
28
29
30
31
32
33
34
35
36

```

status
1 PRODUCTS successfully dropped.

Query Details

Query duration 80ms

Rows 1

Query ID 01b059a9-3200-ef60-...

Checking whether the table is dropped or not

VEDA_DB.VEDA_SH1

```

23
24
25 show tables;
26
27 drop table products;
28
29 undrop table products;
30
31 select * from products;
32
33
34
35
36
37
38
39
40

```

002003 (42502): SQL compilation error:

Object 'PRODUCTS' does not exist or not authorized.

Query Details

Query duration 36ms

Rows 0

Query ID 01b059b3-3200-ef60-...

Now I am trying to undrop the table as the retention time is set to 30 days so the table is restored back.

Query :-

drop table products;

undrop table products;

The screenshot shows a database interface with a sidebar on the left containing a tree view of databases and tables. The main area displays a SQL query editor with the following code:

```
19
20
21 //select statement to retrieve the data
22 select * from products;
23
24 show tables;
25
26 drop table products;
27
28 undrop table products;
29
30
31
32
33
34
35
36
```

Below the query editor, the 'Results' tab is active, showing a single row with the status: 'Table PRODUCTS successfully restored.' The 'Query Details' panel on the right indicates a query duration of 69ms, 1 row, and a query ID of 01b059aa-3200-ef18-0...

As I have used undrop the data is restored back and I have listed the data here

The screenshot shows the same database interface, but the SQL query editor now contains the following code:

```
23
24
25 show tables;
26
27 drop table products;
28
29 undrop table products;
30
31 select * from products;
32
33
34
35
36
37
38
39
40
```

The 'Results' tab is active, displaying a table with 5 rows of product data:

	PRODUCT_ID	PRODUCT_NAME	CATEGORY	MANUFACTURER	PRICE
2	2	Smartphone	Electronics	XYZ Tech	800
3	3	Coffee Maker	Appliances	Home Appliances Co.	50
4	4	Running Shoes	Footwear	SportsGear Inc.	80
5	5	Desk Chair	Furniture	Comfort Furnishings	150

The 'Query Details' panel on the right indicates a query duration of 30ms, 5 rows, and a query ID of 01b059ae-3200-ef5f-0...

Even if by mistake if we have updated something wrong then also, we can roll back the previous data.

Here I am updating the product table so in product table I want to change the price of coffee maker to 200 but instead of this I have updated price of all products to 200 by mistake so all the rows have been updated with price 200

The screenshot shows a database interface with a sidebar on the left containing a tree view of databases and tables. The main area displays a SQL query in a text editor. The query is as follows:

```

23
24 // for retention time
25 show tables;
26
27 drop table products;
28
29 undrop table products;
30 //retrieving the data
31 select * from products;
32
33 // update statement to update the price of coffe maker to 200 but by mistake updated all the products with price 200
34 update products set price = 200;
35 select * from products;
36
37
38
39
40

```

Below the query editor, the 'Results' tab is active, showing a table with 5 rows and 6 columns: PRODUCT_ID, PRODUCT_NAME, CATEGORY, MANUFACTURER, and PRICE. The data is as follows:

PRODUCT_ID	PRODUCT_NAME	CATEGORY	MANUFACTURER	PRICE
1	Laptop	Electronics	ABC Electronics	200
2	Smartphone	Electronics	XYZ Tech	200
3	Coffee Maker	Appliances	Home Appliances Co.	200
4	Running Shoes	Footwear	SportsGear Inc.	200

On the right side of the results table, there is a 'Query Details' panel showing: Query duration: 130ms, Rows: 5, and Query ID: 01b059bd-3200-ef5b-0...

- Here I can travel back using before

I have gone to query history and got the query id from there

The screenshot shows a 'Query History' interface. The left sidebar contains a menu with options like Worksheets, Dashboards, Streamlit, Apps, Data, Marketplace, Activity, Query History (selected), Copy History, Task History, Dynamic Tables, Admin, and Help & Support. The main area displays details for a specific query with ID '01b059bd-3200-ef5b-0006-c22a0003108a'. The query is marked as 'Success' and took 840ms to execute. The 'SQL Text' section shows the query: 'update products set price = 200;'. The 'Query Details' panel on the right shows: Start Time: 11/15/2023, 11:35 PM, End Time: 11/15/2023, 11:35 PM, Warehouse Size: X-Small, Query ID: 01b059bd-3200-ef5b-0006-c22a0003108a, Client Driver: Go 1.1.5, and Session ID: 1902335504867334.

Now in the syntax of time travel I have placed the query id there

we can use before and at commands.

Using before command so it means before this statement whatever the data present in the table, so that need to be retrieved

Query :-

select * from products before (statement => '01b059bd-3200-ef5b-0006-c22a0003108a');

The screenshot shows a Snowflake SQL interface. On the left, a sidebar lists databases and tables. The main area displays a SQL query with line numbers 27 to 44. The query includes a `drop table products;` statement, followed by `undrop table products;`, a comment `//retrieving the data`, a `select * from products;` statement, a comment `// update statement to update the price of coffe maker to 200 but by mistake updated all the products with price 200`, an `update products set price = 200;` statement, another `select * from products;` statement, a comment `//replacing with query id using before`, and a `select * from products before (statement => '01b059bd-3200-ef5b-0006-c22a0003108a');` statement. Below the query, a 'Results' tab is active, showing a table with 4 rows and 6 columns: `PRODUCT_ID`, `PRODUCT_NAME`, `CATEGORY`, `MANUFACTURER`, `PRICE`, and an empty column. The rows contain data for a Laptop, Smartphone, Coffee Maker, and Running Shoes. To the right of the table, a 'Query Details' panel shows 'Query duration' as 1.2s, 'Rows' as 5, and 'Query ID' as '01b059bd-3200-ef5b-0006-c22a0003108a'.

- Now in other scenario I am using timestamp in which we will be able to access the time travel data.

For this I need to give the exact timestamp.

Like I want to access the data present in table before that timestamp

Select records from the 'student' table before a specific timestamp

1. First here I am using the query so that it sets the session time zone to UTC (Coordinated Universal Time).

Query :-

alter session set timezone = "UTC";

2. This query retrieves the current timestamp in the session's time zone.

Query :-

select current_timestamp;

3. This query retrieves records from the 'student' table as they existed before the specified timestamp. "before and at" both are used for time travel.

Query :-

SELECT * FROM student_1 at (TIMESTAMP => '2023-11-16 05:04:00.230'::timestamp);

SELECT * FROM student_1 before (TIMESTAMP => '2023-11-16 05:04:00.230'::timestamp);

Result is :-

1. I have created a student table and inserted 5 records and retrieved the data.

The screenshot shows a database interface with a sidebar on the left containing a search bar and a list of databases: SNOWFLAKE, SNOWFLAKE_SAMPLE_DATA, TEST1_DB, TEST2_DB, and VEDA_DB. The main area displays a SQL script for creating a table named student_2 and inserting five records. The script is as follows:

```
48 //creating a table student
49
50 create table student_2(
51 id int,
52 name varchar(200)
53 )
54
55 //inserting values in student table
56 INSERT INTO student_2(id, name)
57 VALUES
58 (1, 'veda'),
59 (2, 'sita'),
60 (3, 'ravi'),
61 (4, 'rishi'),
62 (5, 'harsha');
63
64 | select * from student_2;
65
```

Below the script, the results of the query are displayed in a table:

	ID	NAME
1	1	veda
2	2	sita
3	3	ravi
4	4	rishi

Query Details:

Query duration	155ms
Rows	5
Query ID	01b05c5b-3200-ef18-0...

25 Queries:

Query	Status	Time	Details
10:34:30 AM	1.3s	select * from student_2	
10:34:29 AM	25ms	select current_time...	
10:34:28 AM	44ms	alter session set t...	
10:34:03 AM	1.7s	update student_1 se...	

2. I have got the current timestamp now

The screenshot shows the same database interface as before, but with a different SQL script. The script is as follows:

```
60 (3, 'ravi'),
61 (4, 'rishi'),
62 (5, 'harsha');
63
64 select * from student_2;
65
66 //using timestamp
67 alter session set timezone = "UTC";
68 select current_timestamp;
69 //update query
70 update student_2 set id =2;
71
72 | select * from student_2;
73
74
75
76 // not executing
77 SELECT * FROM student_1 at (TIMESTAMP => '2023-11-16 05:04:00.230':timestamp);
```

Below the script, the results of the query are displayed in a table:

...	CURRENT_TIMESTAMP
1	2023-11-16 05:17:34.283 +0000

Query Details:

Query duration	111ms
Rows	5
Query ID	01b05c5f-3200-ef18-0...

25 Queries:

Query	Status	Time	Details
10:34:30 AM	1.3s	select * from student_2	
10:34:29 AM	25ms	select current_time...	
10:34:28 AM	44ms	alter session set t...	
10:34:03 AM	1.7s	update student_1 se...	

3. Now I am updating all the rows with id is 2

The screenshot shows the same database interface as before, but with a different SQL script. The script is as follows:

```
60 (3, 'ravi'),
61 (4, 'rishi'),
62 (5, 'harsha');
63
64 select * from student_2;
65
66 //using timestamp
67 alter session set timezone = "UTC";
68 select current_timestamp;
69 //update query
70 update student_2 set id =2;
71
72 | select * from student_2;
73
74
75
76 // not executing
77 SELECT * FROM student_1 at (TIMESTAMP => '2023-11-16 05:04:00.230':timestamp);
```

Below the script, the results of the query are displayed in a table:

	ID	NAME
1	2	veda
2	2	sita
3	2	ravi
4	2	rishi

Query Details:

Query duration	111ms
Rows	5
Query ID	01b05c5f-3200-ef18-0...

25 Queries:

Query	Status	Time	Details
10:34:30 AM	1.3s	select * from student_2	
10:34:29 AM	25ms	select current_time...	
10:34:28 AM	44ms	alter session set t...	
10:34:03 AM	1.7s	update student_1 se...	

4. Now I am using at to retrieve the data back

The screenshot shows the Snowflake SQL Editor interface. The left sidebar displays the database structure with 'VEDA_DB' selected. The main editor contains the following SQL code:

```
66 // using timestamp
67 alter session set timezone = "UTC";
68 select current_timestamp;
69 //update query
70 update student_2 set id =2;
71
72 select * from student_2;
73
74
75 // using at
76 SELECT * FROM student_2 at (TIMESTAMP => '2023-11-16 05:17:34.283'::timestamp);
77
78
79
80
81
82
83
84
```

The 'Results' tab is active, showing a table with 4 rows and 2 columns: ID and NAME.

	ID	NAME
1	1	veda
2	2	sita
3	3	ravi
4	4	rishi

Query Details: Query duration 63ms, Rows 5, Query ID 01b05c65-3200-ef60-...

25 Queries: Thu Nov 16 2023, 10:55:28 AM, 10:54:55 AM

5. Now we need to replace the table

The screenshot shows the Snowflake SQL Editor interface. The left sidebar displays the database structure with 'VEDA_DB' selected. The main editor contains the following SQL code:

```
72 select * from student_2;
73
74
75 // using at
76 SELECT * FROM student_2 at (TIMESTAMP => '2023-11-16 05:17:34.283'::timestamp);
77
78 //replace the table
79 create or replace table student_2 as
80 (SELECT * FROM student_2 at (TIMESTAMP => '2023-11-16 05:17:34.283'::timestamp));
81
82
83 //cloning
84 create table employee
85 (id int,
86 first_name varchar(200))
```

The 'Results' tab is active, showing a single row with the status: 'Table STUDENT_2 successfully created.'

status
1 Table STUDENT_2 successfully created.

Query Details: Rows 1, Query ID 01b05c72-3200-ef8e-0...

status: 100% filled

6. I got the previous data now in the student table

The screenshot shows the Snowflake SQL Editor interface. The left sidebar displays the database structure with 'VEDA_DB' selected. The main editor contains the following SQL code:

```
72 select * from student_2;
73
74
75 // using at
76 SELECT * FROM student_2 at (TIMESTAMP => '2023-11-16 05:17:34.283'::timestamp);
77
78 //replace the table
79 create or replace table student_2 as
80 (SELECT * FROM student_2 at (TIMESTAMP => '2023-11-16 05:17:34.283'::timestamp));
81
82 select * from student_2;
83
84 //cloning
85 create table employee
```

The 'Results' tab is active, showing a table with 4 rows and 2 columns: ID and NAME.

	ID	NAME
1	1	veda
2	2	sita
3	3	ravi
4	4	rishi

Query Details: 25 Queries, Status All, 2:52:41 PM 1.5s, 11:52:50 AM 1.4s

We can use offset also now

The screenshot shows the Snowflake SQL Editor interface. The query editor contains the following SQL code:

```
// using at
SELECT * FROM student_2 at (TIMESTAMP => '2023-11-16 05:17:34.283'::timestamp);

select * from student_2 at (offset => -60*5);

//replace the table
create or replace table student_2 as
(SELECT * FROM student_2 at (TIMESTAMP => '2023-11-16 05:17:34.283'::timestamp));
```

The Results tab is active, displaying a table with 4 rows and 2 columns: ID and NAME.

ID	NAME
1	veda
2	sita
3	ravi
4	rishi

The right sidebar shows the query history with 25 queries. The current query is highlighted, showing it was executed at 2:56:46 PM on Nov 16, 2023, with a status of 'All' and a duration of 1.4s.

2. Cloning :-

I have created a table called employee and inserted the values

The screenshot shows the Snowflake SQL Editor interface. The query editor contains the following SQL code:

```
//creating
create table employee
(id int,
first_name varchar(200),
last_name varchar(200),
gender varchar(20),
address varchar(200)
);

insert into employee
values
(1, 'veda', 'nelanti', 'female', 'hyderabad'),
(2, 'vishnu', 'vardhan', 'male', 'hyderabad'),
(3, 'rohith', 'gorre', 'male', 'warangal'),
(4, 'saisri', 'bhavi', 'female', 'bangalore');

select * from employee;
```

The Results tab is active, displaying a table with 4 rows and 5 columns: ID, FIRST_NAME, LAST_NAME, GENDER, and ADDRESS.

ID	FIRST_NAME	LAST_NAME	GENDER	ADDRESS
1	veda	nelanti	female	hyderabad
2	vishnu	vardhan	male	hyderabad
3	rohith	gorre	male	warangal
4	saisri	bhavi	female	bangalore

The right sidebar shows the query history with 25 queries. The current query is highlighted, showing it was executed at 2:59:46 PM on Nov 16, 2023, with a status of 'All' and a duration of 240ms.

Now I have created employee_clone table

The screenshot shows the Snowflake SQL Editor interface. The query editor contains the following SQL code:

```
(3, 'rohith', 'gorre', 'male', 'warangal'),
(4, 'saisri', 'bhavi', 'female', 'bangalore');

select * from employee;

create table employee_clone CLONE employee;

select * from employee_clone;

insert into employee
values (5, 'devi', 'nagadurga', 'female', 'karimnagar');

insert into employee_clone
values (6, 'teja', 'sriramoju', 'female', 'warangal');

//caching
select * from employee;
```

The Results tab is active, displaying a table with 4 rows and 5 columns: ID, FIRST_NAME, LAST_NAME, GENDER, and ADDRESS.

ID	FIRST_NAME	LAST_NAME	GENDER	ADDRESS
1	veda	nelanti	female	hyderabad
2	vishnu	vardhan	male	hyderabad
3	rohith	gorre	male	warangal
4	saisri	bhavi	female	bangalore

The right sidebar shows the query history with 25 queries. The current query is highlighted, showing it was executed at 3:01:18 PM on Nov 16, 2023, with a status of 'All' and a duration of 305ms.

Here I have added a other record in employee table

The screenshot shows a database interface with a sidebar on the left containing a search bar and a list of databases: SNOWFLAKE, SNOWFLAKE_SAMPLE_DATA, TEST1_DB, TEST2_DB, and VEDA_DB. The main area displays a SQL query in the VEDA_DB.VEDA_SH1 schema:

```
111 select * from employee_clone;
112
113 insert into employee
114 values (5, 'devi','nagadurga','female', 'karimnagar');
115
116 select * from employee;
117
118
119
120
121
122
123
124
125
126
127 insert into employee_clone
```

Below the query editor, the 'Results' tab shows a table with 5 columns: ID, FIRST_NAME, LAST_NAME, GENDER, and ADDRESS. The table contains 5 rows of data:

ID	FIRST_NAME	LAST_NAME	GENDER	ADDRESS
2	vishnu	vardhan	male	hyderabad
3	rohith	gorre	male	warangal
4	saisri	bhavi	female	bangalore
5	devi	nagadurga	female	karimnagar

The '25 Queries' panel on the right shows a list of queries with their execution times and status.

The record is not added in clone table

The screenshot shows a database interface with a sidebar on the left containing a search bar and a list of databases: SNOWFLAKE, SNOWFLAKE_SAMPLE_DATA, TEST1_DB, TEST2_DB, and VEDA_DB. The main area displays a SQL query in the VEDA_DB.VEDA_SH1 schema:

```
115 select * from employee;
116
117
118
119 select * from employee_clone;
120
121
122
123
124
125
126
127
128 insert into employee_clone
129 values (6, 'teja','sriramoju','female','warangal');
130
131 //caching
132 select * from employee;
```

Below the query editor, the 'Results' tab shows a table with 5 columns: ID, FIRST_NAME, LAST_NAME, GENDER, and ADDRESS. The table contains 4 rows of data:

ID	FIRST_NAME	LAST_NAME	GENDER	ADDRESS
1	veda	nelanti	female	hyderabad
2	vishnu	vardhan	male	hyderabad
3	rohith	gorre	male	warangal
4	saisri	bhavi	female	bangalore

The '25 Queries' panel on the right shows a list of queries with their execution times and status.

Now I have added the one record in clone table

The screenshot shows a database interface with a sidebar on the left containing a search bar and a list of databases: SNOWFLAKE, SNOWFLAKE_SAMPLE_DATA, TEST1_DB, TEST2_DB, and VEDA_DB. The main area displays a SQL query in the VEDA_DB.VEDA_SH1 schema:

```
109 create table employee_clone CLONE employee;
110
111 select * from employee_clone;
112
113 insert into employee
114 values (5, 'devi','nagadurga','female', 'karimnagar');
115
116 select * from employee;
117
118 select * from employee_clone;
119
120 insert into employee_clone
121 values (6, 'teja','sriramoju','female','warangal');
122
123 select * from employee_clone;
124
125
126
```

Below the query editor, the 'Results' tab shows a table with 5 columns: ID, FIRST_NAME, LAST_NAME, GENDER, and ADDRESS. The table contains 5 rows of data:

ID	FIRST_NAME	LAST_NAME	GENDER	ADDRESS	
2	vishnu	vardhan	male	hyderabad	
3	rohith	gorre	male	warangal	
4	saisri	bhavi	female	bangalore	
5	6	teja	sriramoju	female	warangal

The '25 Queries' panel on the right shows a list of queries with their execution times and status.

When I query to get the data from employee table then the record which has been added in employee_clone table is not there in employee so main table and clone table is not dependant on each other

The screenshot shows a database interface with a sidebar on the left containing a tree view of databases and tables. The main area displays SQL queries and their results. The queries are as follows:

```
109 create table employee_clone CLONE employee;
110
111 select * from employee_clone;
112
113 insert into employee
114 values (5, 'devi', 'nagadurga', 'female', 'karimnagar');
115
116 select * from employee;
117
118 select * from employee_clone;
119
120 insert into employee_clone
121 values (6, 'teja', 'sriramoju', 'female', 'warangal');
122 select * from employee_clone;
```

The results section shows a table with 5 rows and 6 columns: ID, FIRST_NAME, LAST_NAME, GENDER, ADDRESS. The data is as follows:

ID	FIRST_NAME	LAST_NAME	GENDER	ADDRESS
2	vishnu	vardhan	male	hyderabad
3	rohith	gorre	male	warangal
4	saisri	bhavi	female	bangalore
5	devi	nagadurga	female	karimnagar

The right sidebar shows a list of queries with their execution times and status.

Now we can also clone the schema

The screenshot shows a database interface with a sidebar on the left containing a tree view of databases and tables. The main area displays SQL queries and their results. The queries are as follows:

```
115 select * from employee;
116
117 select * from employee_clone;
118
119 insert into employee_clone
120 values (6, 'teja', 'sriramoju', 'female', 'warangal');
121
122 select * from employee_clone;
123
124 //cloning the schema
125 create or replace schema clone_ved_sh_schema clone veda_sh1;
126
127 create schema clone_schema clone veda_sh;
128
129
130
131 //caching
132 select * from employee;
```

The results section shows a table with 1 row and 1 column: status. The data is as follows:

status
Schema CLONE_VED_SH_SCHEMA successfully created.

The right sidebar shows a list of queries with their execution times and status.

we can clone the database too

The screenshot shows a database interface with a sidebar on the left containing a tree view of databases and tables. The main area displays SQL queries and their results. The queries are as follows:

```
121 values (6, 'teja', 'sriramoju', 'female', 'warangal');
122 select * from employee_clone;
123
124 //cloning the schema
125 create or replace schema clone_ved_sh_schema clone veda_sh1;
126
127 create schema clone_schema clone veda_sh;
128
129 create database clone_database clone test1_db;
130
131
132
133
134
135
136 //caching
137 select * from employee;
```

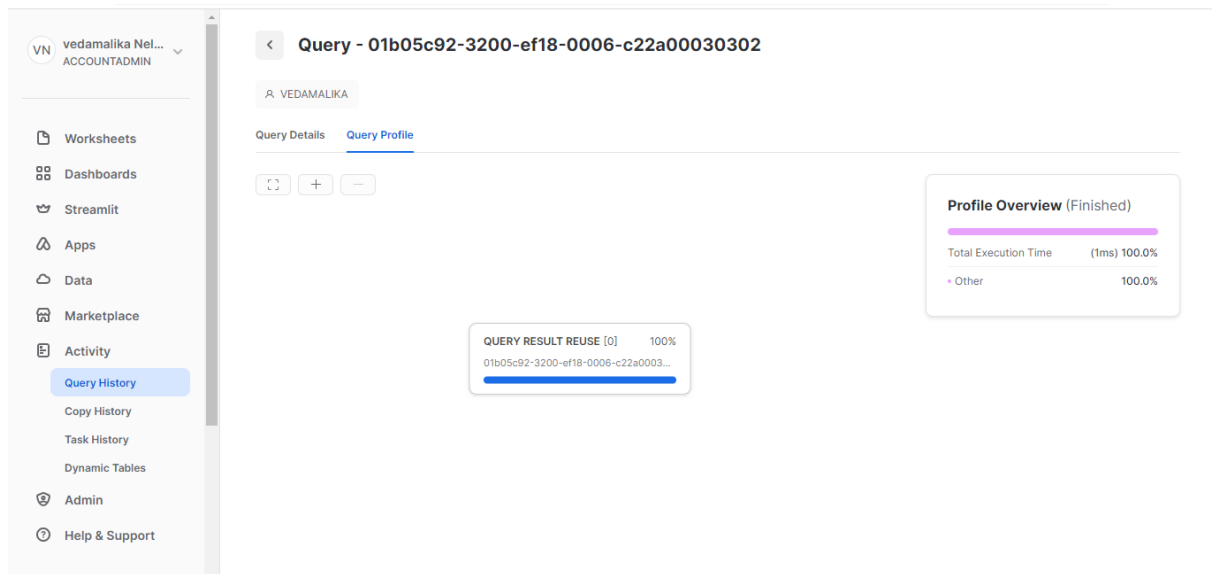
The results section shows a table with 1 row and 1 column: status. The data is as follows:

status
Database CLONE_DATABASE successfully created.

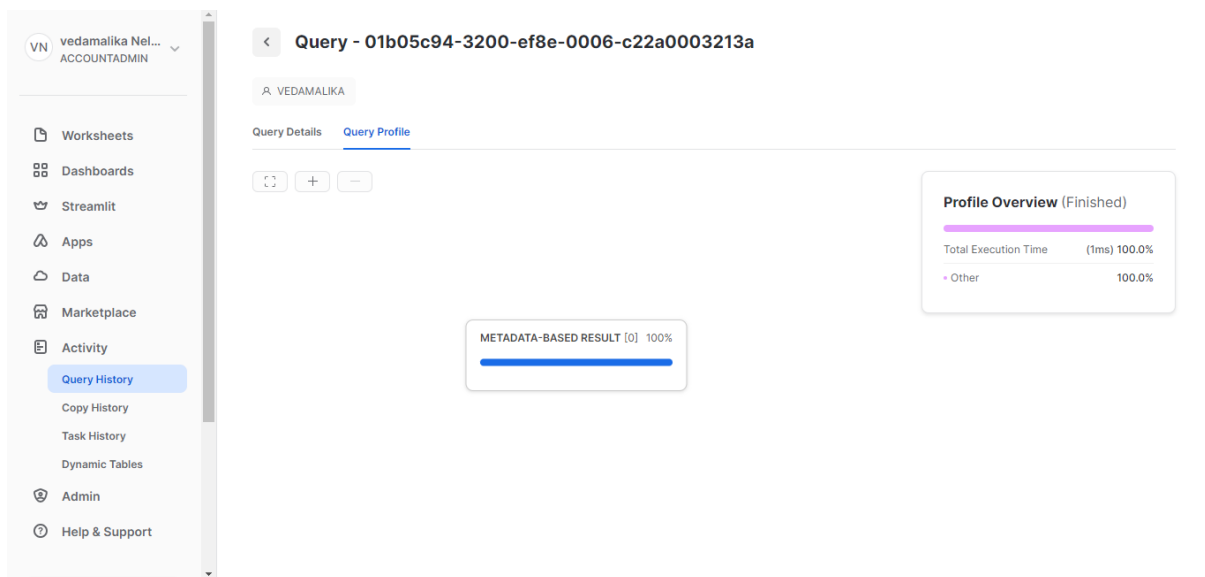
The right sidebar shows a list of queries with their execution times and status.

3. Cache :-

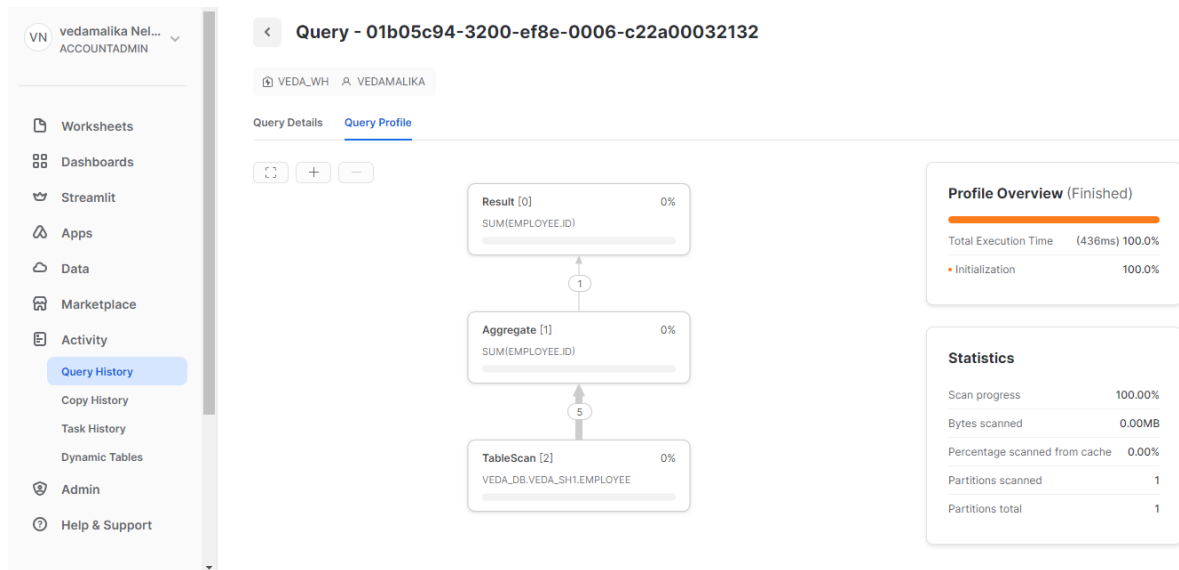
We have result cache



We have meta data cache



For sum aggregate function from the table scan the result has been achieved



From table scan we have got the result

