

## Time Travel

1. Data Protection LifeCycle
2. So far what we have know is...we have a data storage amd we can query data using commands

The diagram illustrates the current state of data storage. On the left, a dark blue rounded rectangle contains the text "Current Data Storage" in orange. To its right, a white box contains the SQL command `SELECT * FROM table`. Below these, a table displays employee data.

	ID	FIRST_NAME	LAST_NAME	EMAIL
1	1	Bourke	Treble	btreble0@g.co
2	2	Iormina	Lahy	ilahy1@sciencedaily.com
3	3	Tracy	Curwen	tcurwen2@spiegel.de
4	4	Megan	Omond	momond3@cyberchimps.com

- 3.
4. What if by mistake we dropped our database...then all our tables will be lost

The diagram shows potential database errors. On the left, a dark blue rounded rectangle contains the text "Current Data Storage" in orange. To its right, two white boxes contain SQL commands: `DROP DATABASE prod_db;` and `TRUNCATE TABLE prod_table;`. Below these, a table shows a status message.

	status
1	PROD_DB successfully dropped.

Time Travel enables accessing historical data.

- 5.
6. So if we have updated some table or if we have deleted some data, we can still query this deleted data, for example, by using timestamps in the past or query IDs

## What is possible with Time Travel?

- Query deleted or updated data
- Restore tables, schemas and databases that have been dropped
- Create clones of tables, schemas and databases from previous state

7.

Query historic data within retention period.

```
SELECT * FROM table AT (TIMESTAMP => timestamp)
```

1

TIMESTAMP

```
SELECT * FROM table AT (OFFSET => -10*60)
```

2

OFFSET

8.

Offset in Snowflake time travel is the difference in seconds between the current time and the point in time that you want to query. You can use offset to query historical data without having to specify a specific date and time.

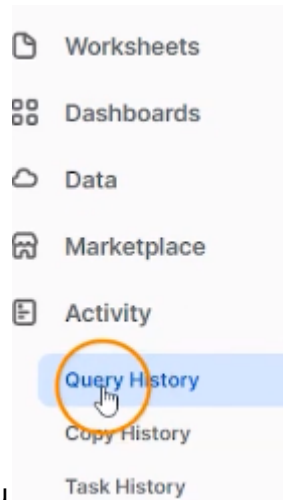
For example, the following query will query the state of the `my_table` table at a point in time that is 10 minutes ago:

SQL

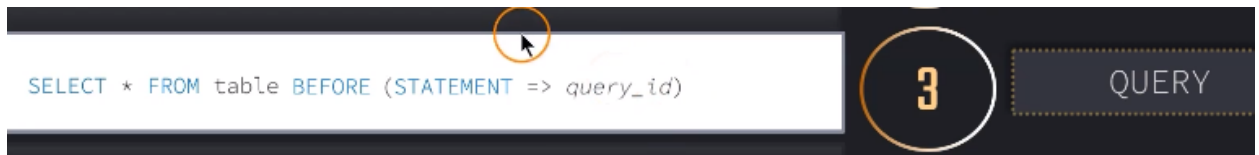
```
SELECT * FROM my_table AT OFFSET -600;
```

9.

10. We can use query id...which generates when we execute a query...we can get query



id..from query historu



If you need to access the query ID of the last query that was executed, use the global variable `SQLID`.

#### Note

If no query was executed, the default value of `SQLID` is NULL.

11.



UNDROP fails if an object with the same name already exists.

OWNERSHIP privileges are needed for an object to be restored.

## 12. Some rules

### Hands On

1. Refer online and files

### Retention Period

Number of days for which this historical data is preserved and Time Travel can be applied.

**Configurable for**  
table, schema,  
database and account

**DATA\_RETENTION\_TIME\_IN\_DAYS = 2**

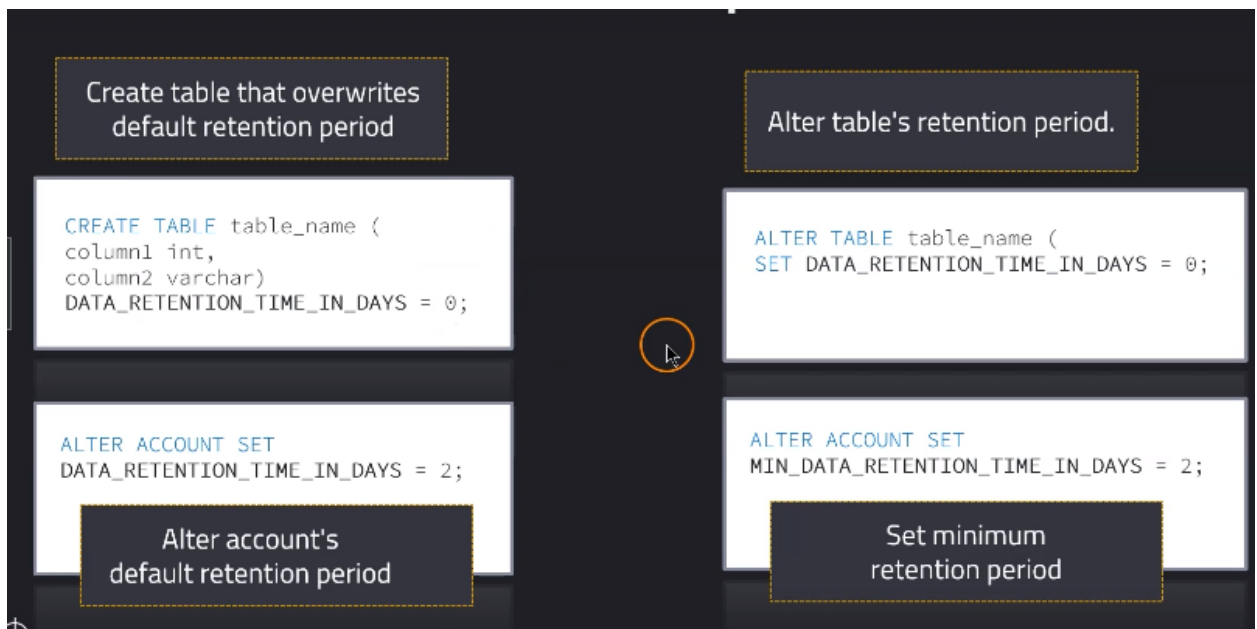
**DEFAULT = 1**

For all accounts

Retention period of 0 "disables" time travel.

- 1.

- Retention period ...by default is 1 day ...we can extend it by using data\_retention...seepic
- If set retention period = 0 for a table..then we cannot use time travel on that table



- 
- 
- 
- 
- Suppose on account level..if we have min\_data\_reten = 2...then for tables in this account..if we set data\_reten = 0..then it wont work...if set higher than min\_retention ..then it works

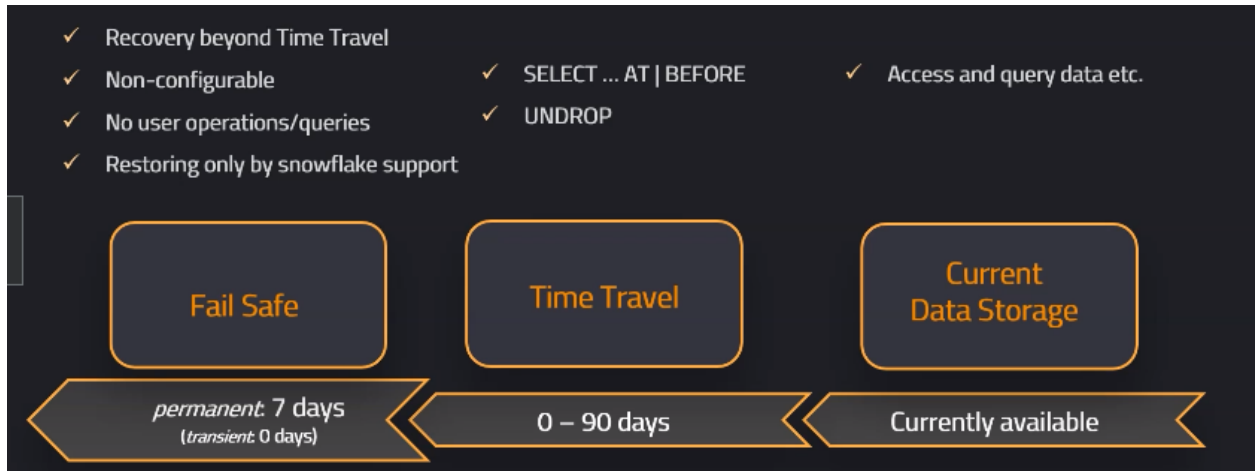


- 
- 
- 
- 
- 
- 
- Hands on refer file

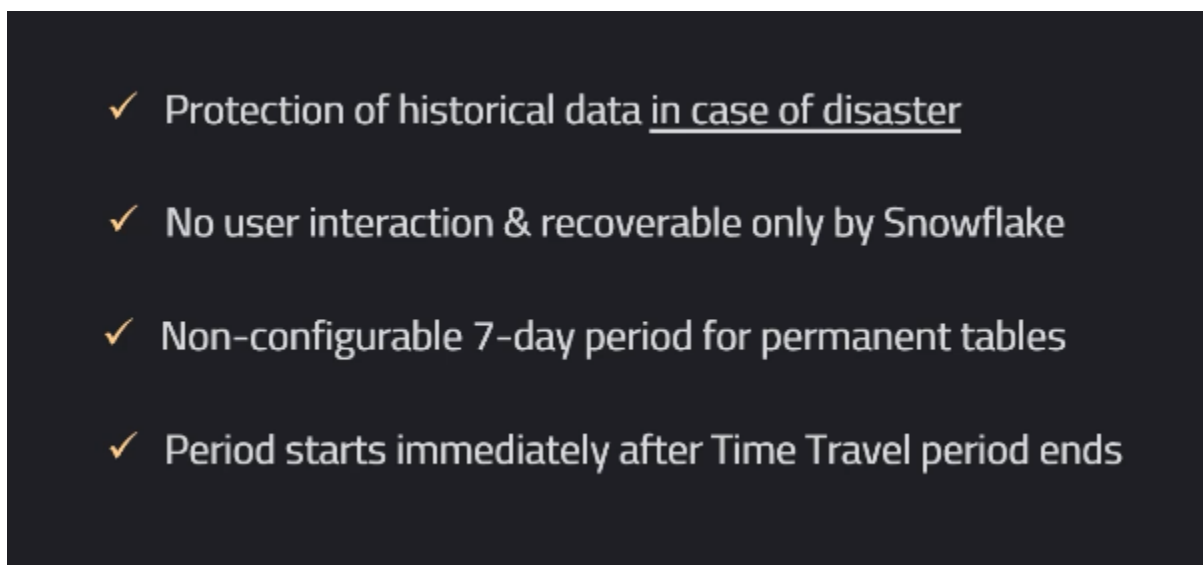
## Fail Safe

- What if theres some kind of disaster..and we r not able to recover data..even with time travel?

2. In this case, we, on top of time travel, also have failsafe.
3. So on top of this time travel retention period, we have seven days of failsafe, which is non configurable and it will always be seven days for the standard permanent tables.



4.



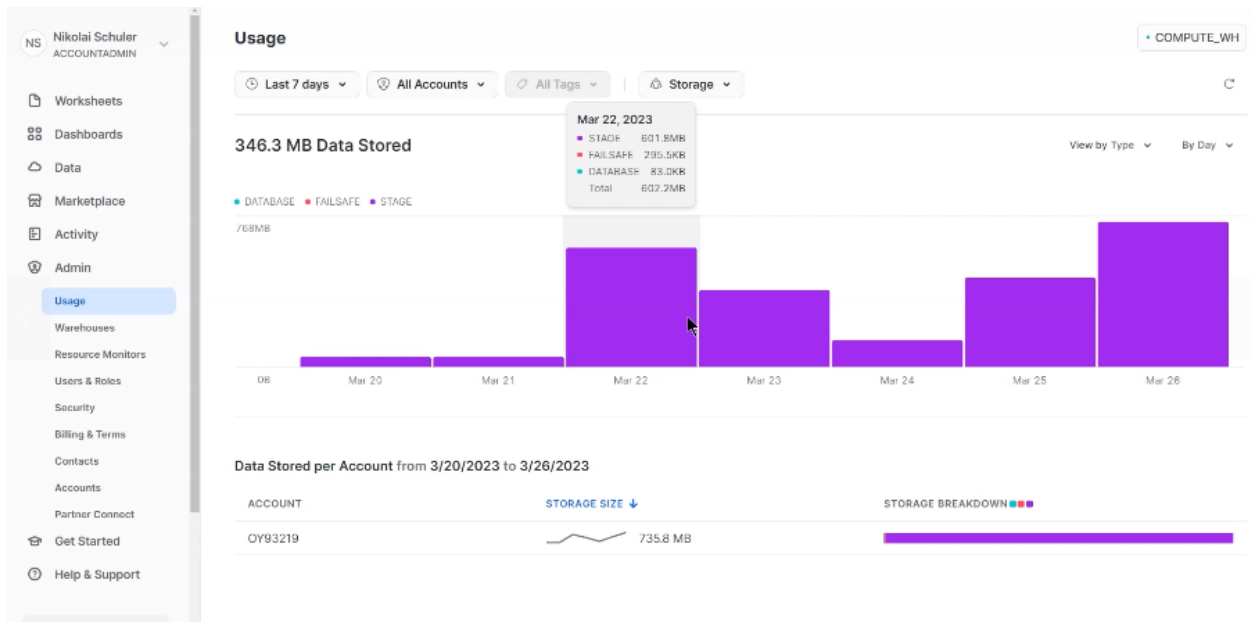
5.

✓ Contributes to storage cost

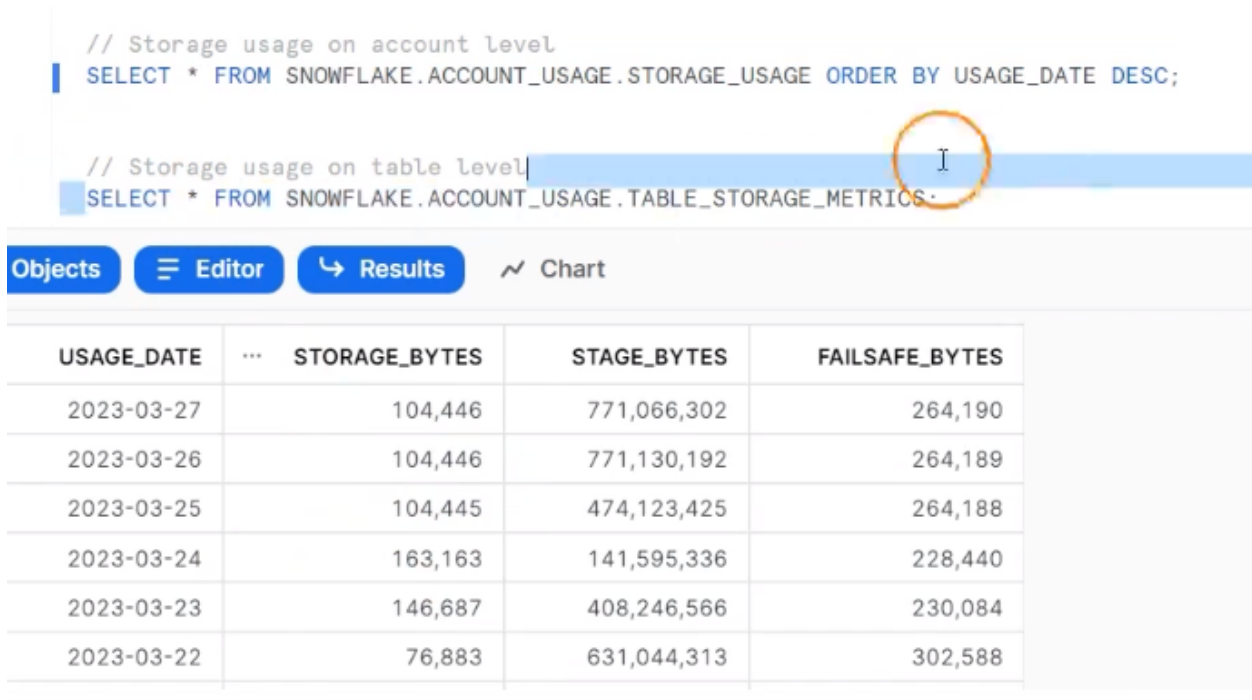
## Storage costs

1. Lets look at costs incurred for time travel
2. We can analyze this costs in two ways..

3. First is thru snowflake interface



4. We can also analyze costs using queries



5. But it does not provide....details for time travel

6. To get detailed usage on tables level use

```
// Storage usage on table level
SELECT * FROM SNOWFLAKE.ACCOUNT_USAGE.TABLE_STORAGE_METRICS;
```

- We can run this query ..to get more readability on storage costs table level

```
// Storage usage on table level - formatted
SELECT ID,
       TABLE_NAME,
       TABLE_SCHEMA,
       TABLE_CATALOG,
       ACTIVE_BYTES / (1024*1024*1024) AS STORAGE_USED_GB,
       TIME_TRAVEL_BYTES / (1024*1024*1024) AS TIME_TRAVEL_STORAGE_USED_GB,
       FAILSAFE_BYTES / (1024*1024*1024) AS FAILSAFE_GB
FROM SNOWFLAKE.ACCOUNT_USAGE.TABLE_STORAGE_METRICS
ORDER BY STORAGE_USED_GB DESC, TIME_TRAVEL_STORAGE_USED_GB DESC;
```

## Table types

	Permanent data	Large tables that does not need to be protected	Non-permanent data
	Permanent	Transient	Temporary
Time Travel Retention Period	CREATE TABLE 0 – 90 days	CREATE TRANSIENT TABLE 0 or 1 day	CREATE TEMPORARY TABLE 0 or 1 day
Fail Safe	✓ Fail Safe	✗ Fail Safe	✗ Fail Safe
Persistence	Until dropped	Until dropped	Only in session

- 
- We can decide which table we want...and thus by we can save storage costs

- ✓ Types are also available for other database objects

*If database is transient all included objects are transient.*

Table	Stages
Schema	Database

- ✓ For temporary table no naming conflicts with permanent/transient tables

*Other tables will be effectively hidden! Relevant for time travel.*

*Not visible to other users!*

- ✓ Not possible to change type of object for existing object

-



