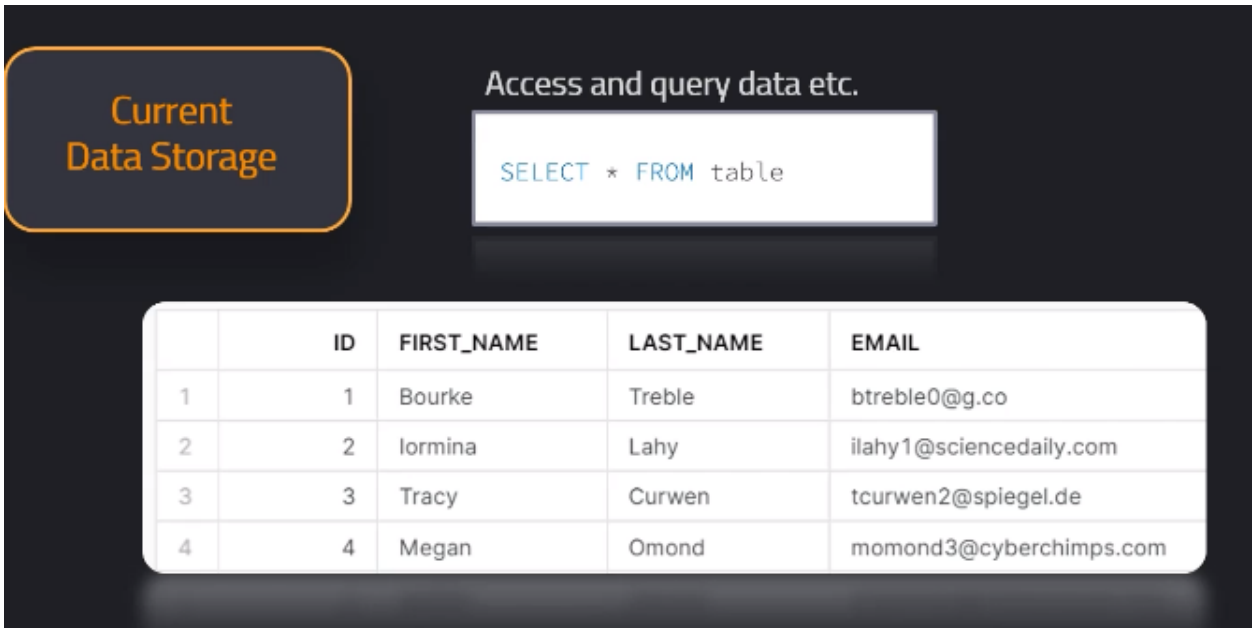
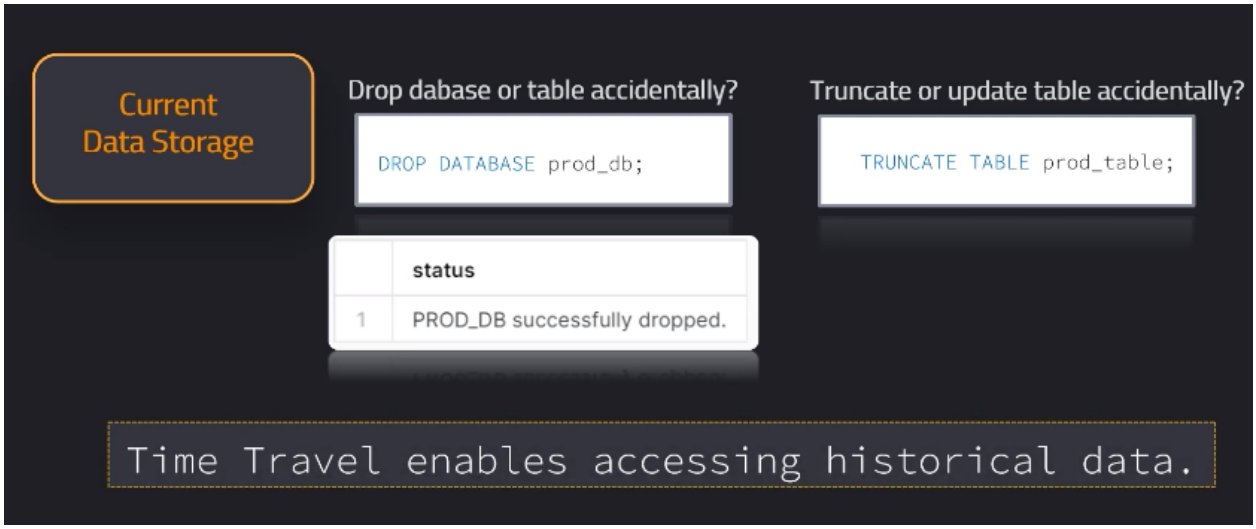
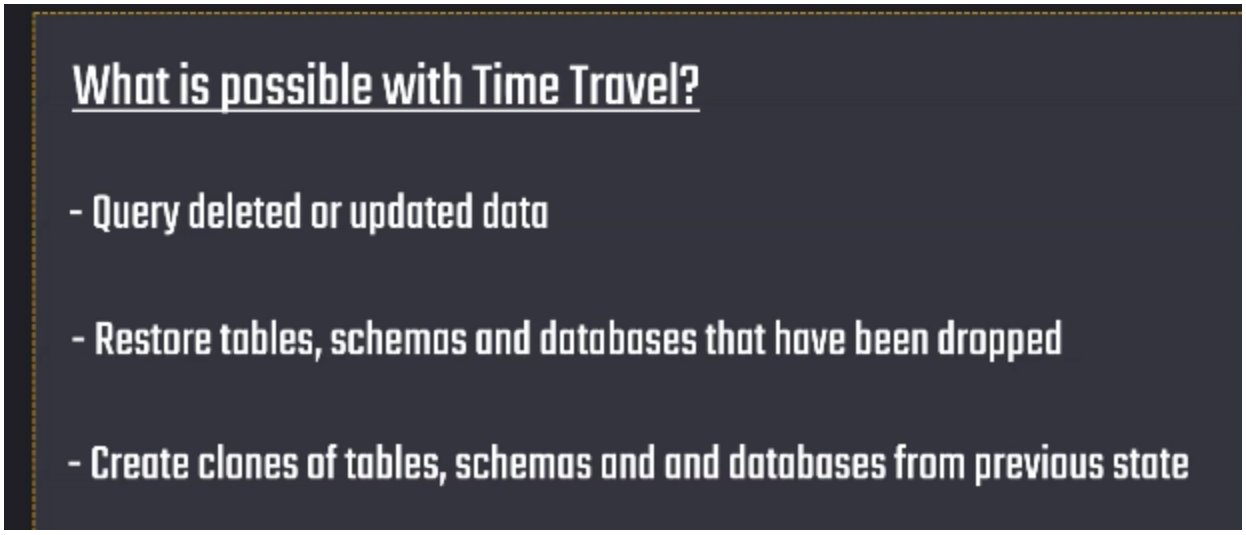
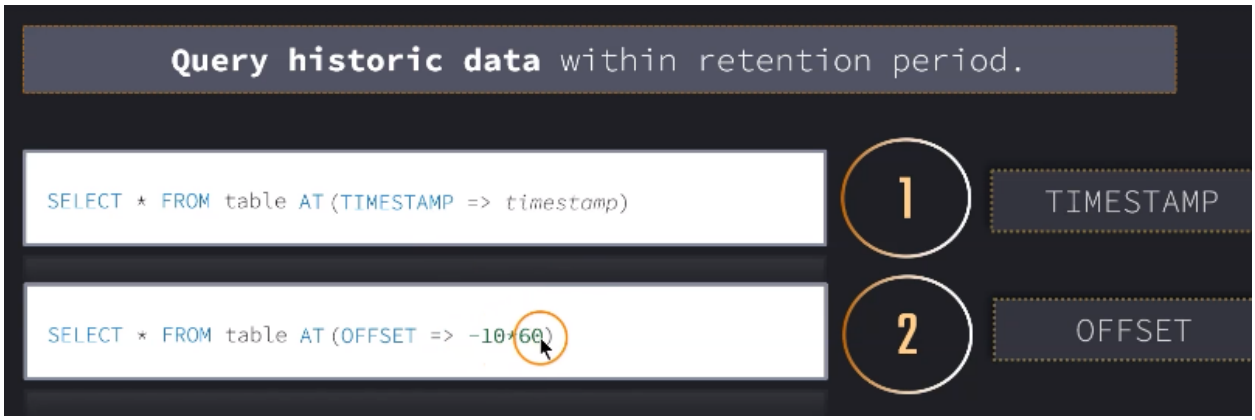
Time Travel

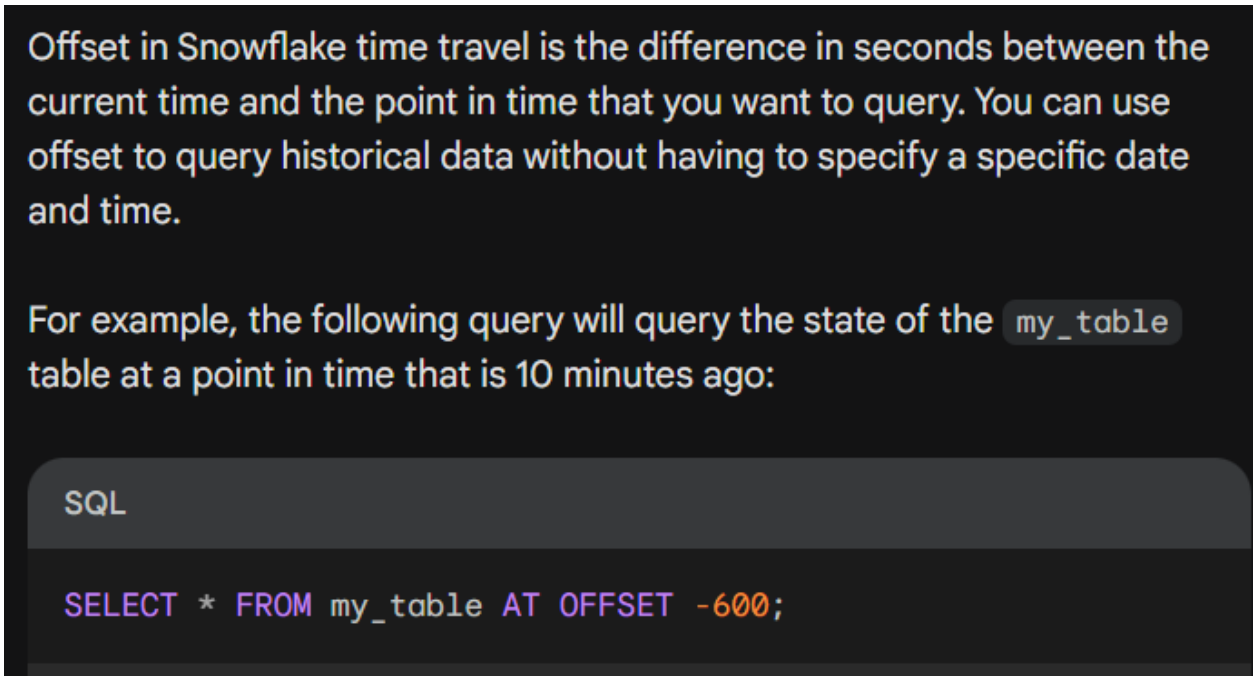
1. Data Protection LifeCycle
2. So far what we have know is…we have a data storage amd we can query data using commands
3. 

4. What if by mistake we dropped our database…then all our tables will be lost
5. 

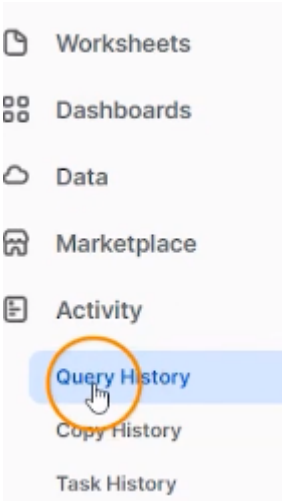
6. So if we have updated some table or if we have deleted some data, we can still query this deleted data, for example, by using timestamps in the past or query IDs

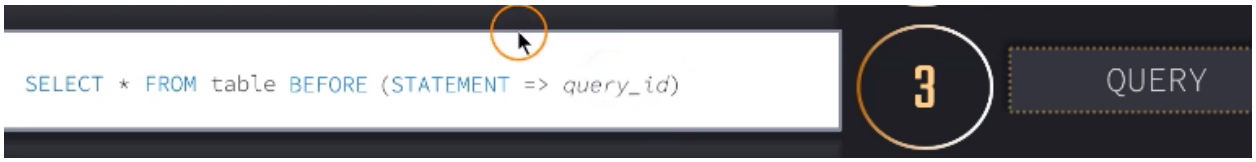
7. 
What is possible with Time Travel?
- Query deleted or updated data
- Restore tables, schemas and databases that have been dropped
- Create clones of tables, schemas and and databases from previous state

8. 
Query historic data within retention period.
SELECT * FROM table AT (TIMESTAMP => timestamp)
1
TIMESTAMP
SELECT * FROM table AT (OFFSET => -10*60)
2
OFFSET

9. 
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;

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



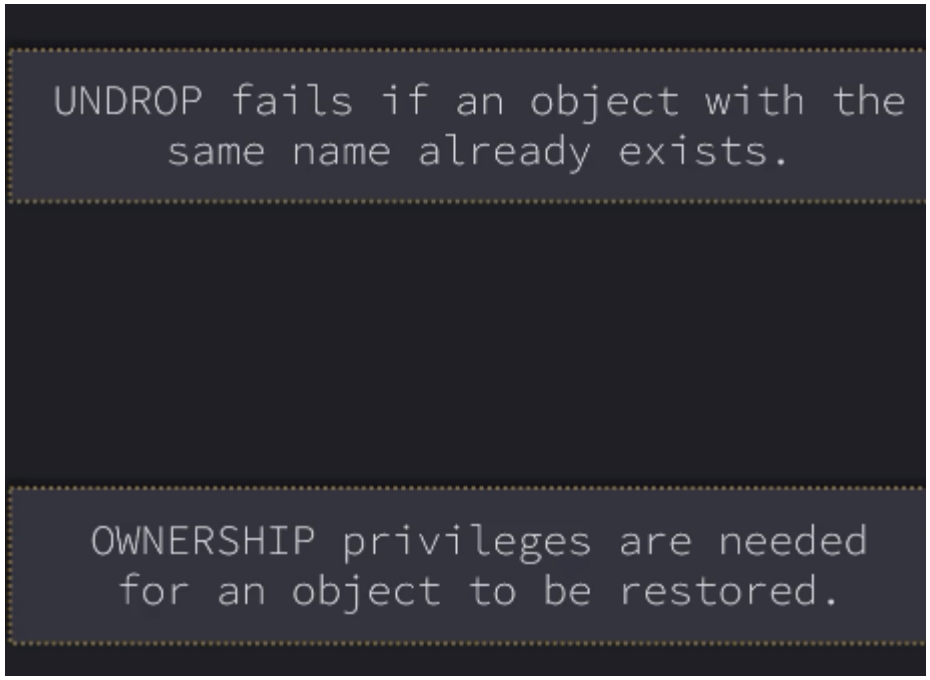


If you need to access the query ID of the last query that was executed, use the global variable `SQLID`.

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

11.

12. Some rules

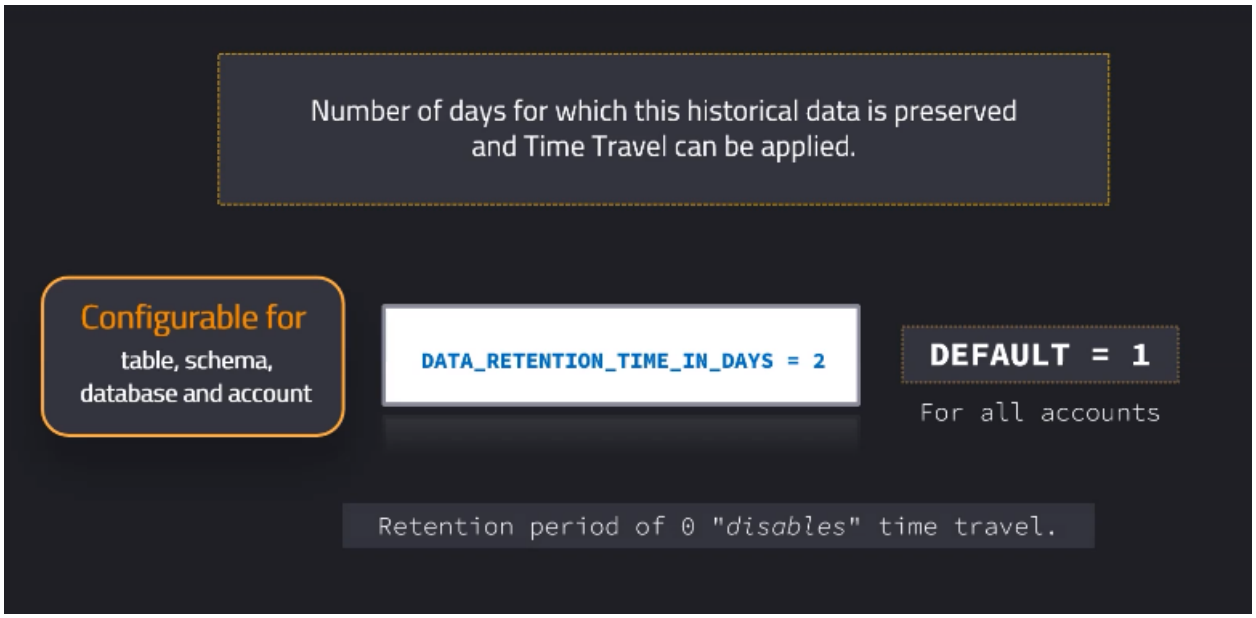


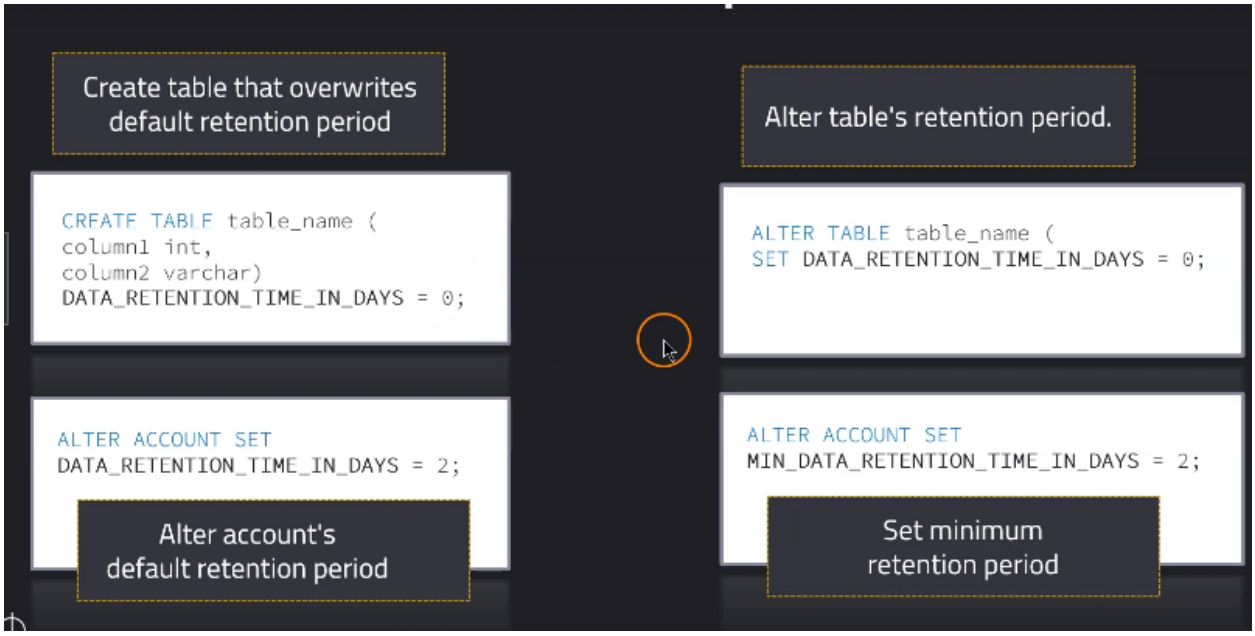
Hands On

1. Refer online and files

Retention Period

1.



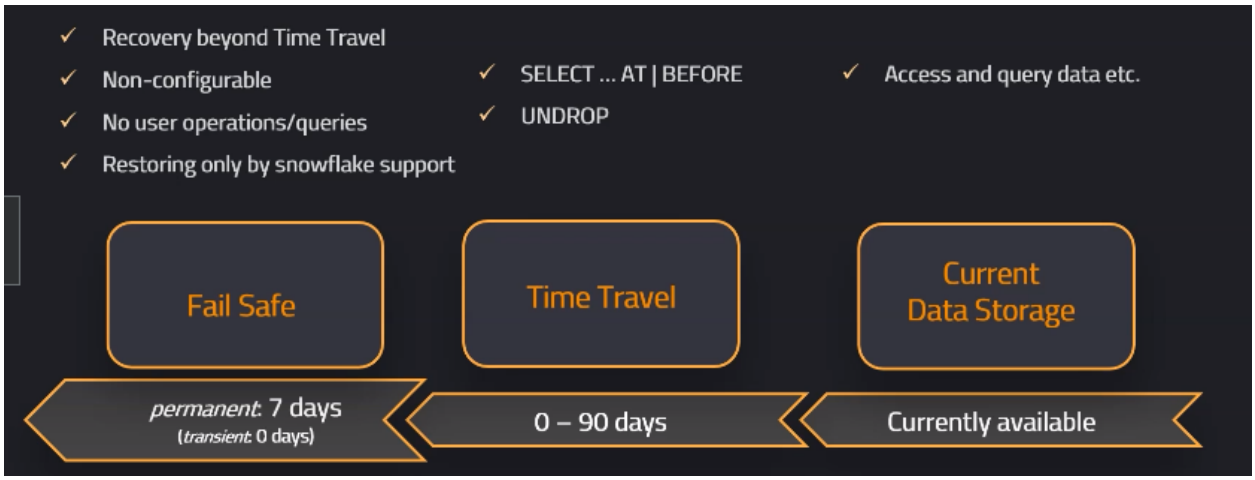
2. Retention period …by default is 1 day …we can extend it by using data_retention…seepic
3. If set retention period = 0 for a table..then we cannot use time travel on that table
4. 

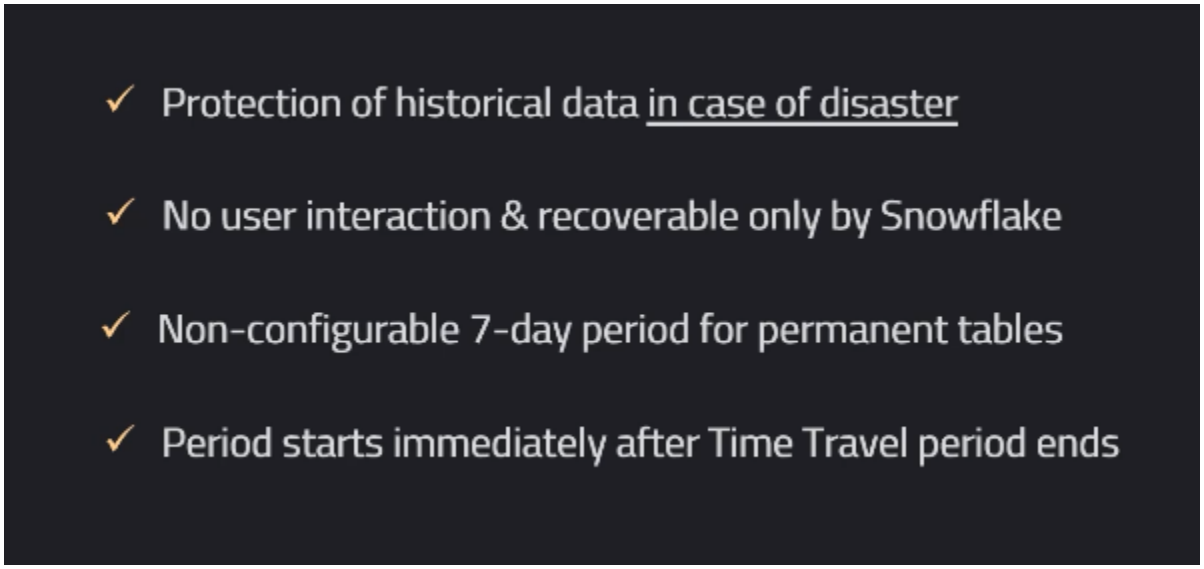
5. Suppose on account level..if we have min_data_reten = 2…then for tables in this account..if we set data_reten = 0..then it wont work…if set higher than min_retention ..then it works
6. 

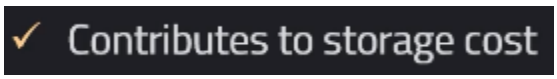
7. Hands on refer file

Fail Safe

1. 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. 

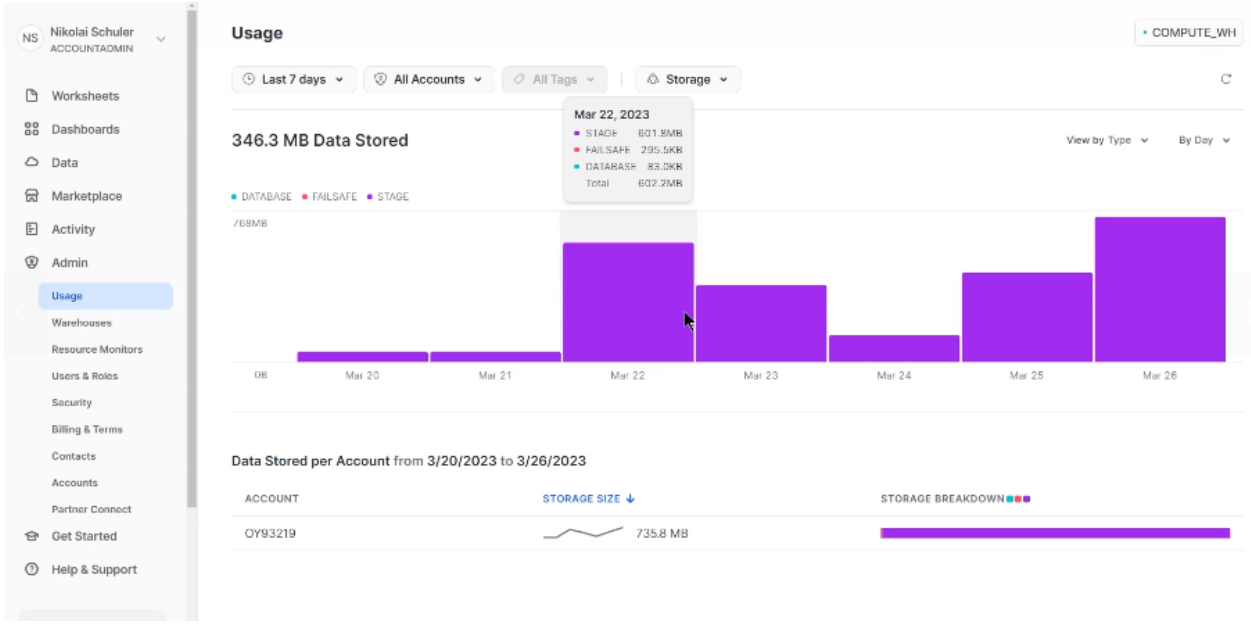
5. 



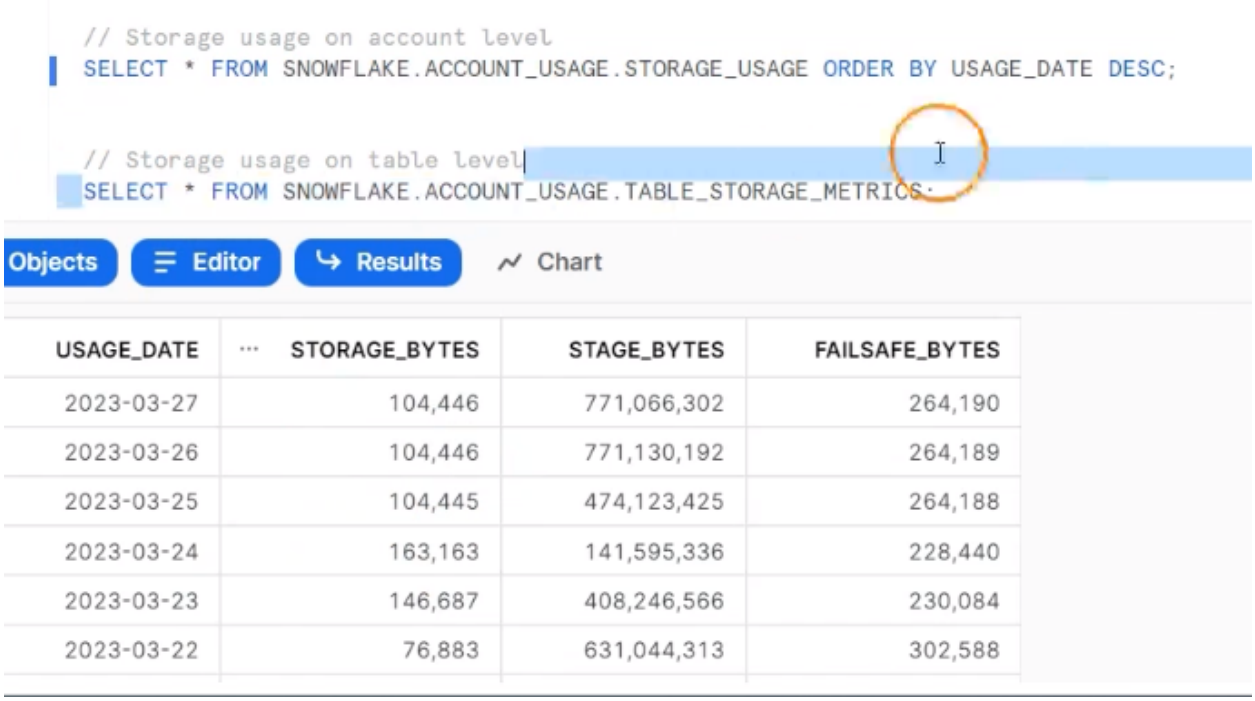
Storage costs

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



USAGE_DATE	STORAGE_BYTES	STAGE_BYTES	FAILSAFE_BYTES
2023-03-27	104,446	771,066,302	264,190
2023-03-26	104,446	771,130,192	264,189
2023-03-25	104,445	474,123,425	264,188
2023-03-24	163,163	141,595,336	228,440
2023-03-23	146,687	408,246,566	230,084
2023-03-22	76,883	631,044,313	302,588

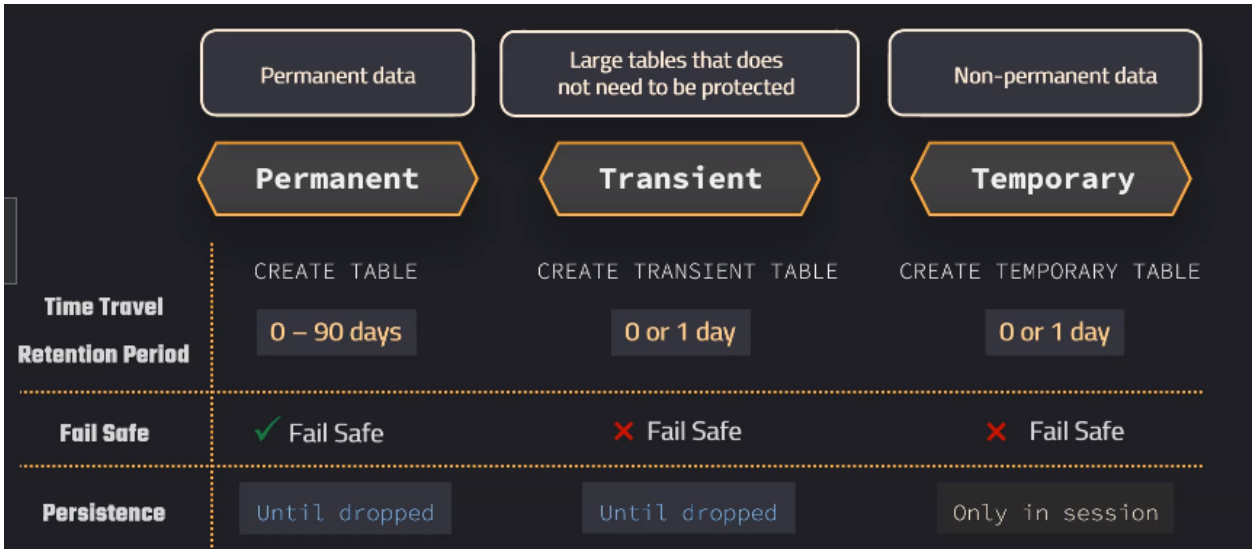
5. But it does not provide….details for time travel
6. To get detailed usage on tables level use

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

7. 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

1. 

2. We can decide which table we want…and thus by we can save storage costs
3.