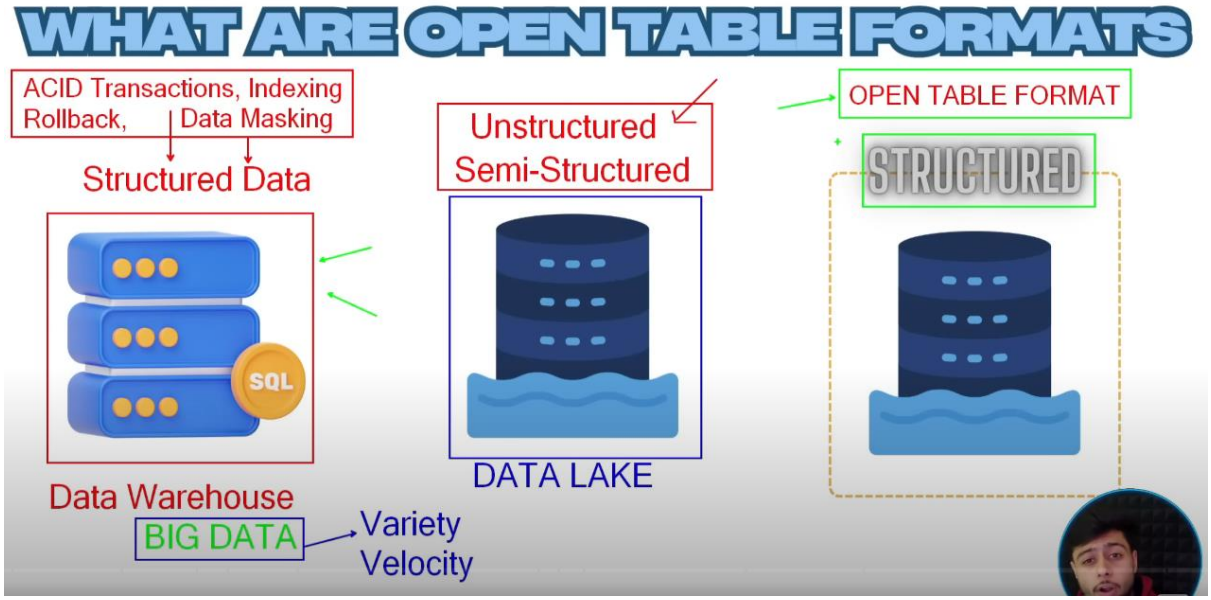
Open Table Format

AWS people primarily uses Apache Iceberg, concept wise exactly same but they have different optimization technique.

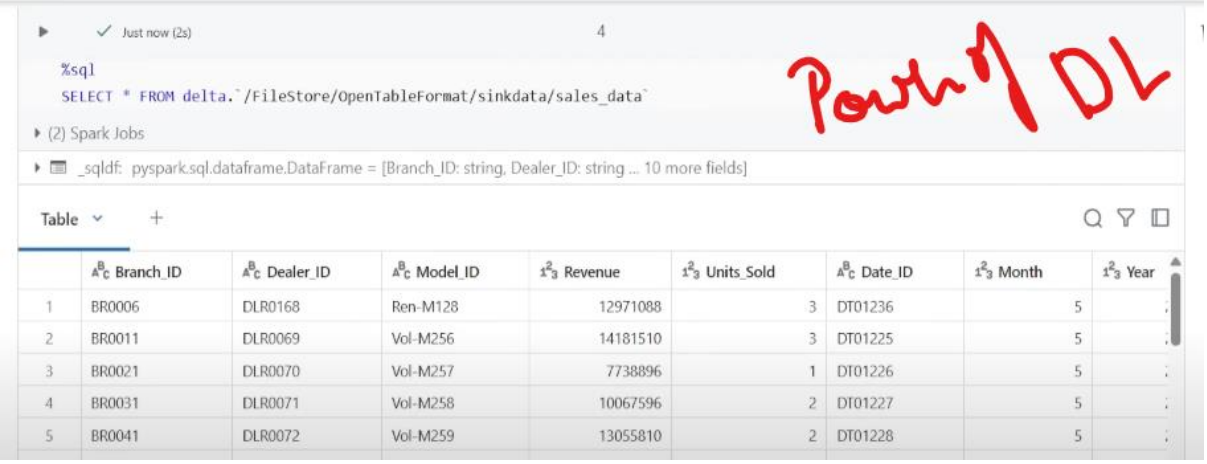


Structured data used to be kept in OLAP database i.e DWH

But with the rise of Big Data, varieties, velocity & volume of data, we moved to Data Lake, but Data Lake doesnot has ACID, Indexing, Rollback, Time Travel and Data Masking features which are there in DWH.

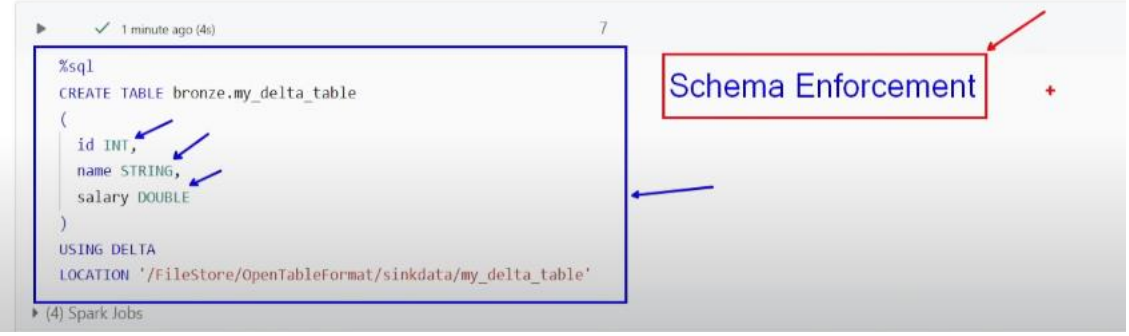
Open Table Format → behaves like a Table format, it use same Data Lake but add a metadata/structured layer on top of that.

For delta path, don't ever go to file level, just write till folder level



	Branch_ID	Dealer_ID	Model_ID	Revenue	Units_Sold	Date_ID	Month	Year
1	BR0006	DLR0168	Ren-M128	12971088	3	DT01236	5	
2	BR0011	DLR0069	Vol-M256	14181510	3	DT01225	5	
3	BR0021	DLR0070	Vol-M257	7738896	1	DT01226	5	
4	BR0031	DLR0071	Vol-M258	10067596	2	DT01227	5	
5	BR0041	DLR0072	Vol-M259	13055810	2	DT01228	5	

Without even creating a table we can display the data in Parquet file in format of table.

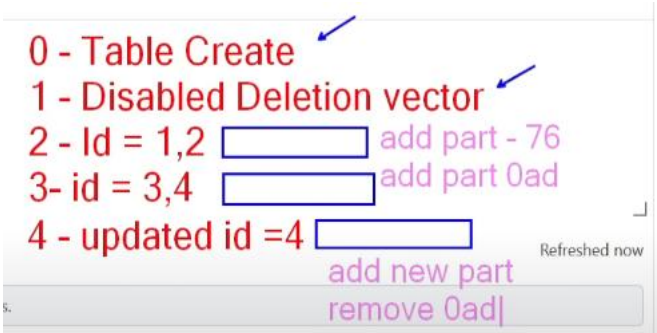


We have explicitly defined the datatypes, will not accept any data with some other formats, this is known as Schema Enforcement.

Deletion Vector

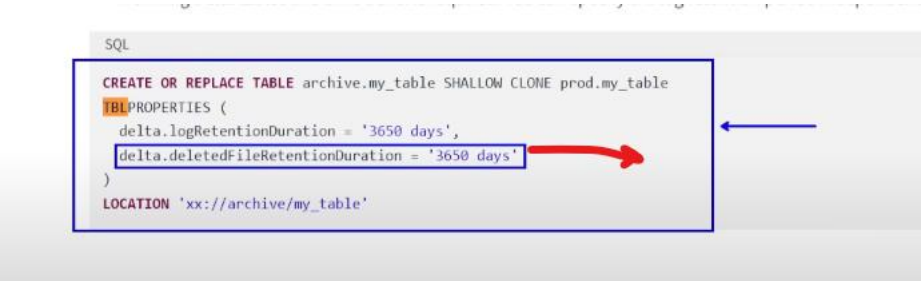
Insertion of data in table will create same no of json files with DV as well as without DV.

Now in case of DML, without DV



Operation 0 table has been created, 1→disabling DV, 2→ inserting data for id=1,2(part file named 76 got added); 3→ inserting data for id=3,4(part file named 0ad got added); 4→ updating data for id=4→ Now 0ad partition will be removed as it contains id=3(Soft Delete/Tombstoning), it will create a new partition and push all the data of the soft deleted partition along with the updated changes.

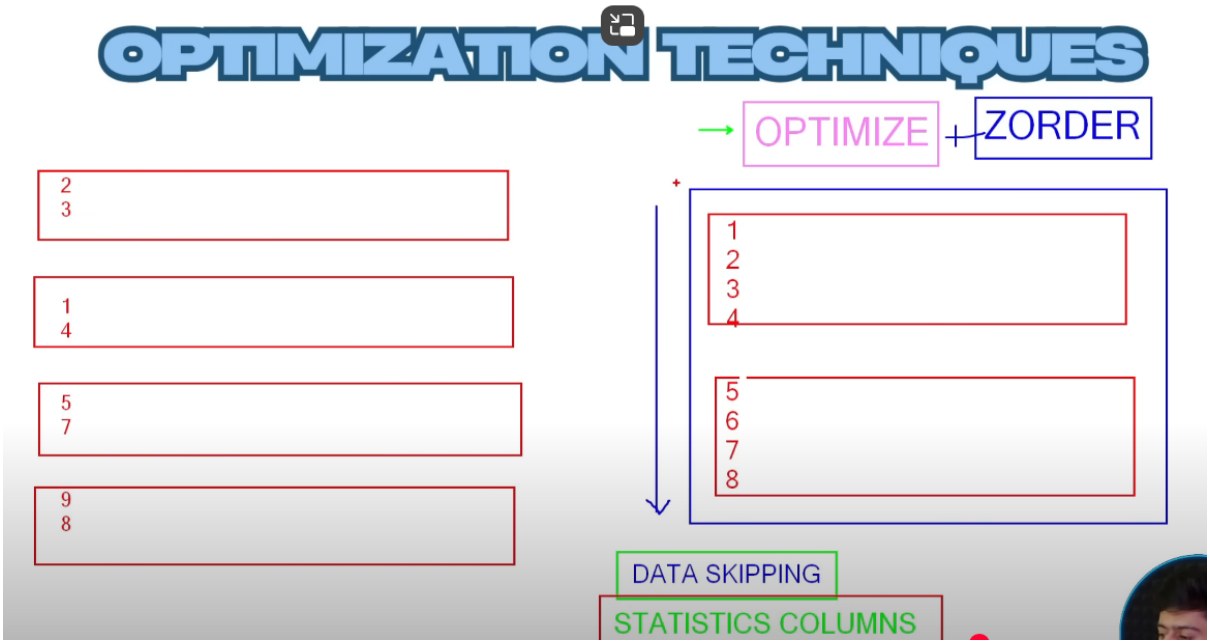
But what's the need of creating new partitions instead of direct change in existing partitions?
Because of Data Versioning & Time Travel



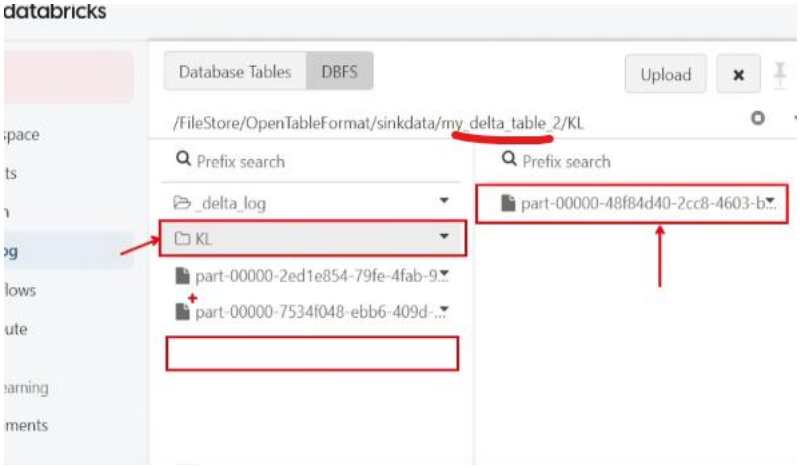
To change default value of deleted file Retention Period of 7 days to defined period.

Metadata level changes

Adding a column in Delta Lake will not add the column in the Parquet file but add the column in metadata/Delta log.(that means new json will be created but no new parquet file)
Same goes for reordering of the column.



ZORDER BY → use data/partition skipping
ZORDER BY basically uses Column Statistics (store statistics of first 32 columns →hold min, max and avg value), make sure the ZORDER BY column should be present in first 32 columns of the dataframe



Earlier the optimised file used to be kept in the delta log folder itself along with other files, but now a new folder KL gets created which holds the optimised files, it is basically a coalesced version of the 2 files. And the new json for this OPTIMIZE operation will mention remove the 2 partitions & add the KL optimized partition.

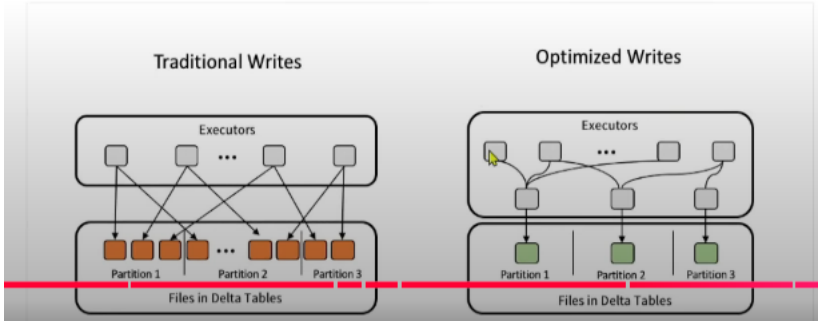
OPTIMIZE WRITE

Optimized writes for Delta Lake on Azure Databricks

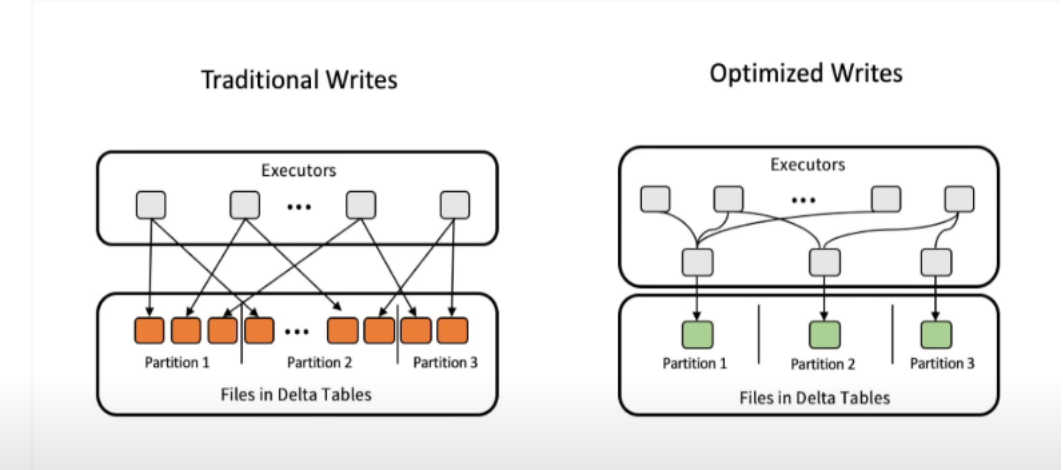
Optimized writes improve file size as data is written and benefit subsequent reads on the table.

Optimized writes are most effective for partitioned tables, as they reduce the number of small files written to each partition. Writing fewer large files is more efficient than writing many small files, but you might still see an increase in write latency because data is shuffled before being written.

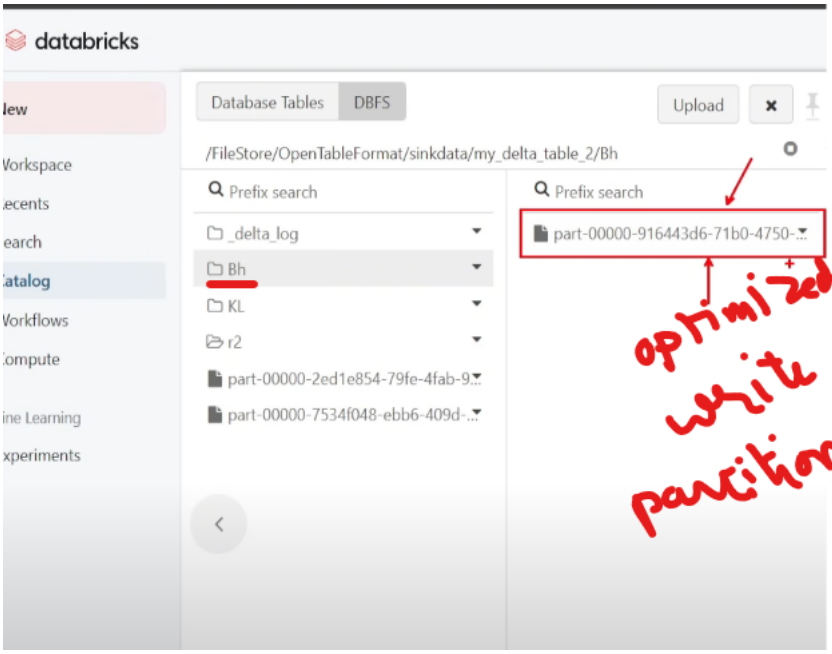
The following image demonstrates how optimized writes works:



The following image demonstrates how optimized writes works:

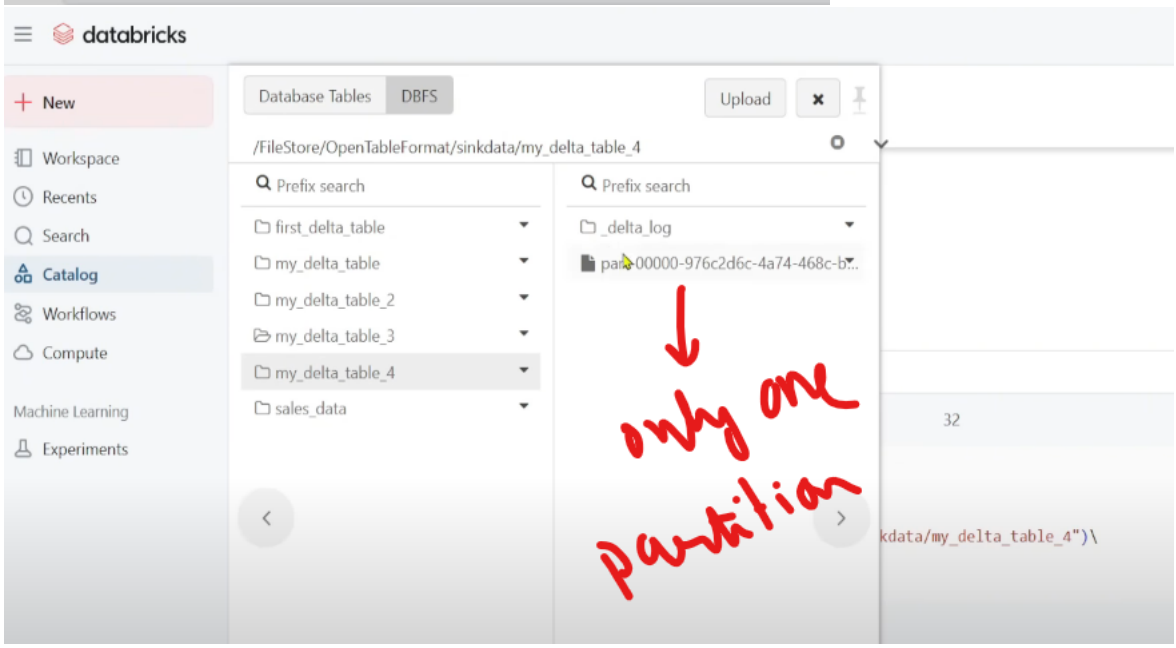


So perviously we can coalesce the files after being written, but now with we can coalesce the files before being written. Before, Executors directly writes partitions, but now instead of writing partitions directly, Executors will talk to each other & they will just coalesce the partitions and then will write it, so will get lesser number of partitions



So while writing the file, use OptimizeWrite but also ensure to write OPTIMIZE commands to coalesce the no of partitions because even if the data comes in the most optimized format(can be thousands of partitions created throughout the day) still you need to reduce the number of partitions at the end of the day.

Lets check it on a fresh table

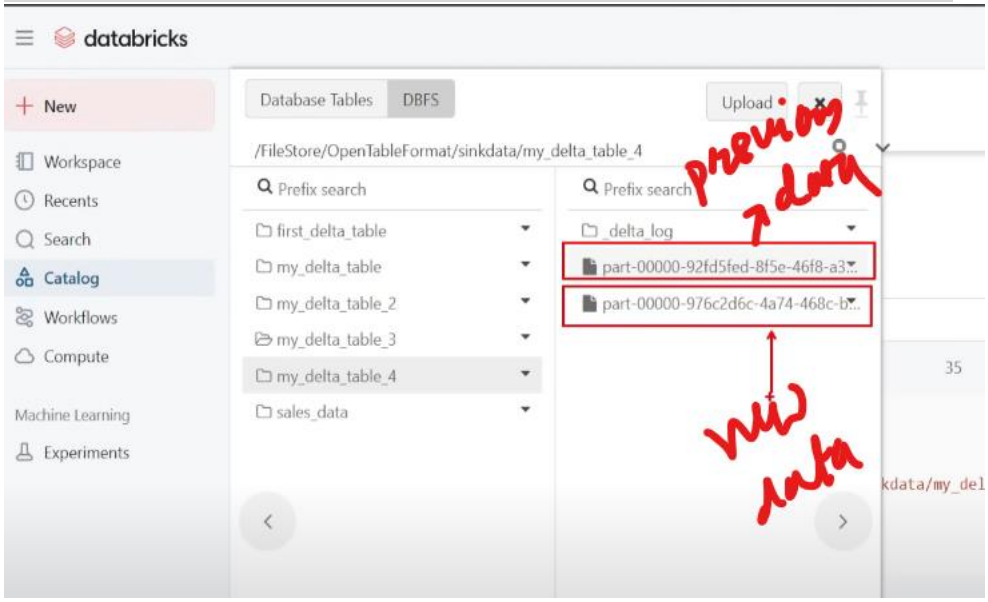


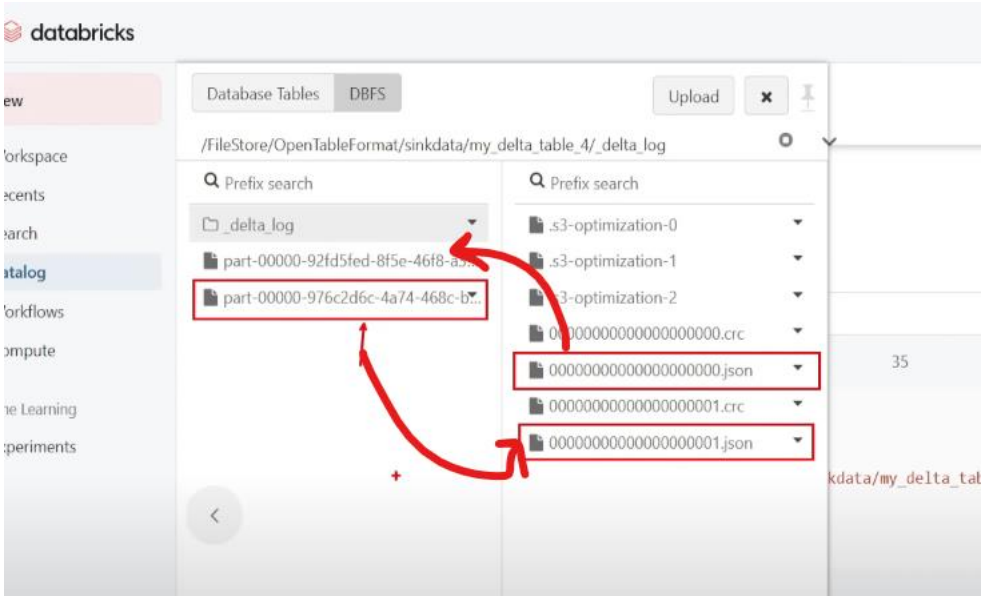
Now lets test one scenario

Create a new dataframe

Now try to overwrite the existing data in the table with the data in the dataframe

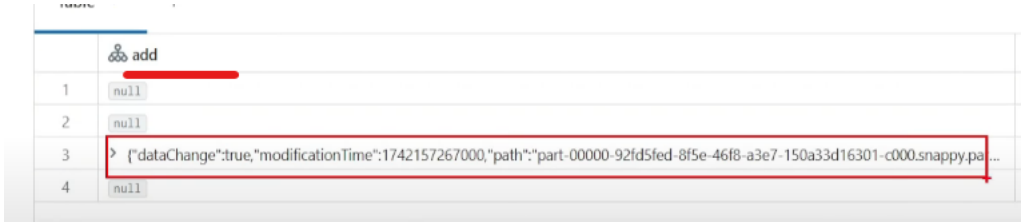
Even if we are overwrite the data in the table, we cant do the overwrite of schema in Delta Lake (In Spark Data Lake we can do it as it overwrites files along with schema)



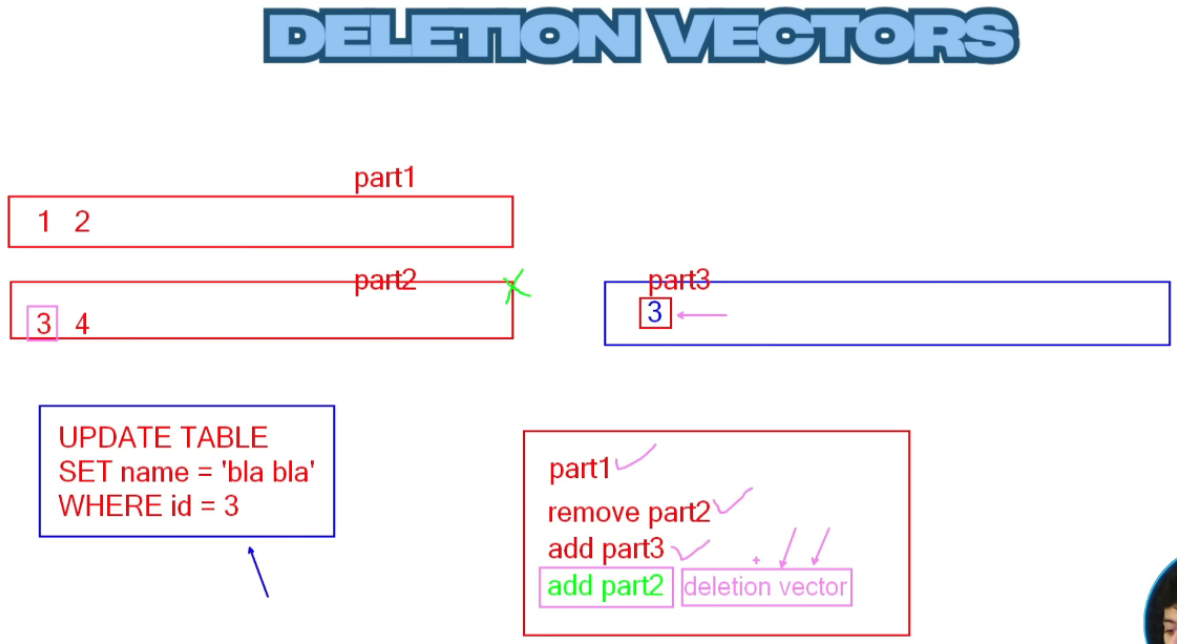


In Delta Lake, even if we overwrite the file, we can see the previous file because of Time Travel feature.

Check the recent json



Deletion Vectors



At the end it will coalesce the partitions using OPTIMIZE

Structured Streaming