Partitioning Delta Lake Tables

Learning Objectives

▶ Partitioning definition

▶ How to partition Delta Lake tables

▶ Best practices

Partitioning

- A partition is a subset of rows that share the same value for predefined partitioning columns

partitioning column

id	name	month	year
1	Adam	9	2015
2	Sarah	11	2015
3	John	5	2000
4	Ali	9	2000
5	Emma	10	2000
6	Eric	1	2023
7	Sophia	3	2023

Partition 1
year=2015

Partition 2
year=2020

Partition 3
year=2023

Partitioning Delta Lake Tables

CREATE TABLE my_table (id INT, name STRING, year INT, month INT)
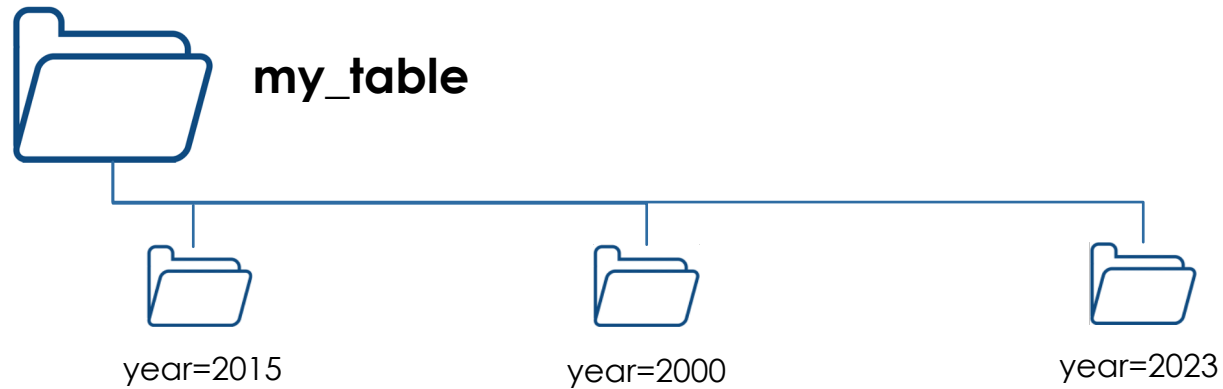
PARTITIONED BY (year)

my_table

Partitioning Delta Lake Tables

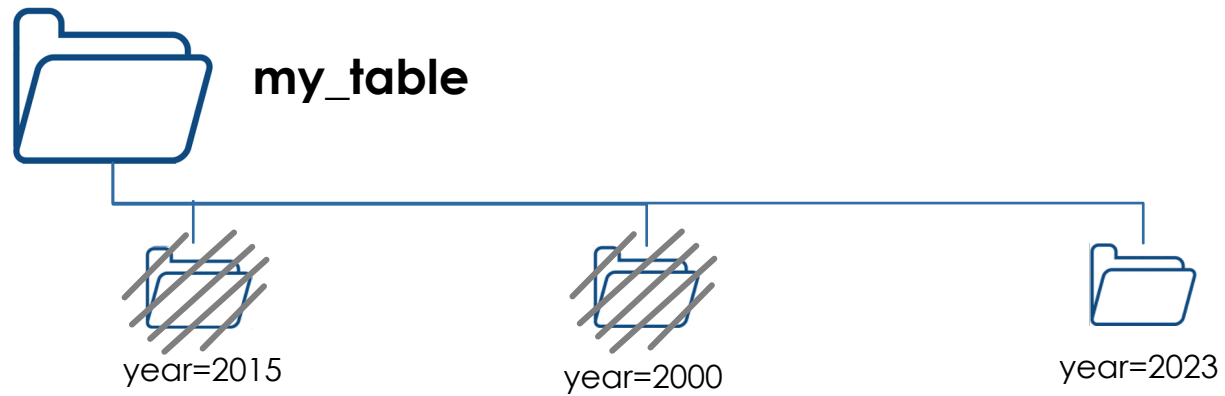**CREATE TABLE** my_table (id INT, name STRING, year INT, month INT)

PARTITIONED BY (year)
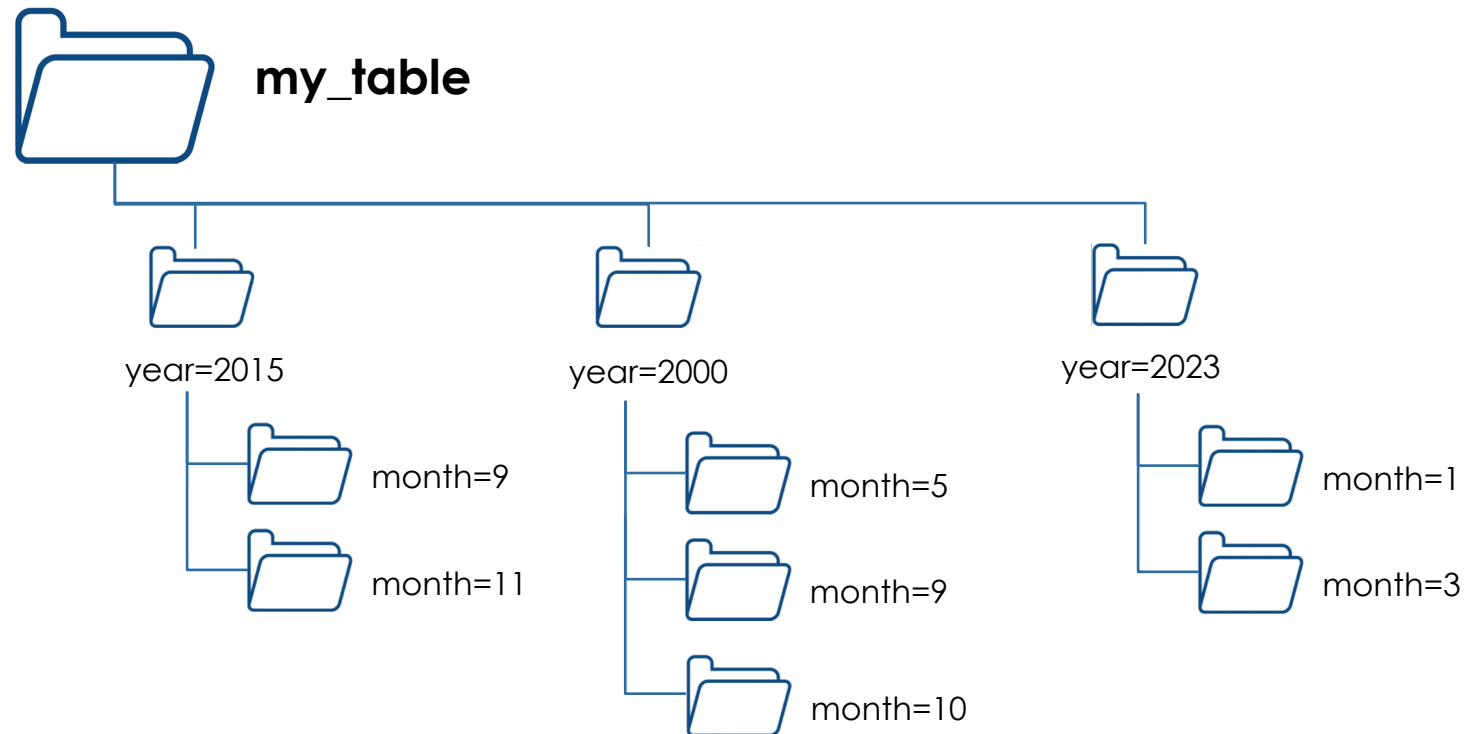
Partition Skipping

SELECT * FROM my_table

Where year = 2023

Multiple Partitioning Columns

CREATE TABLE my_table (id INT, name STRING, year INT, month INT)

PARTITIONED BY (year, month)

Choosing partition columns

1. Total values present in a column
   - ▶ Low cardinality fields should be used for partitioning

2. Total records share a given value for a column
   - ▶ Partitions should be at least 1 GB in size

3. Records with a given value will continue to arrive indefinitely
   - ▶ Datetime fields can be used for partitioning

Avoid Over-Partitioning

- ▶ Files cannot be combined or compacted across partition boundaries
  - ▶ Partitioned small tables increase storage costs and total number of files to scan.

- ▶ If most partitions < 1GB of data, the table is over-partitioned

- ▶ Data that is over-partitioned or incorrectly partitioned will suffer greatly
  - ▶ Lead to slowdowns for most general queries
  - ▶ Require a full rewrite of all data files to remedy

Deleting at Partition Boundaries
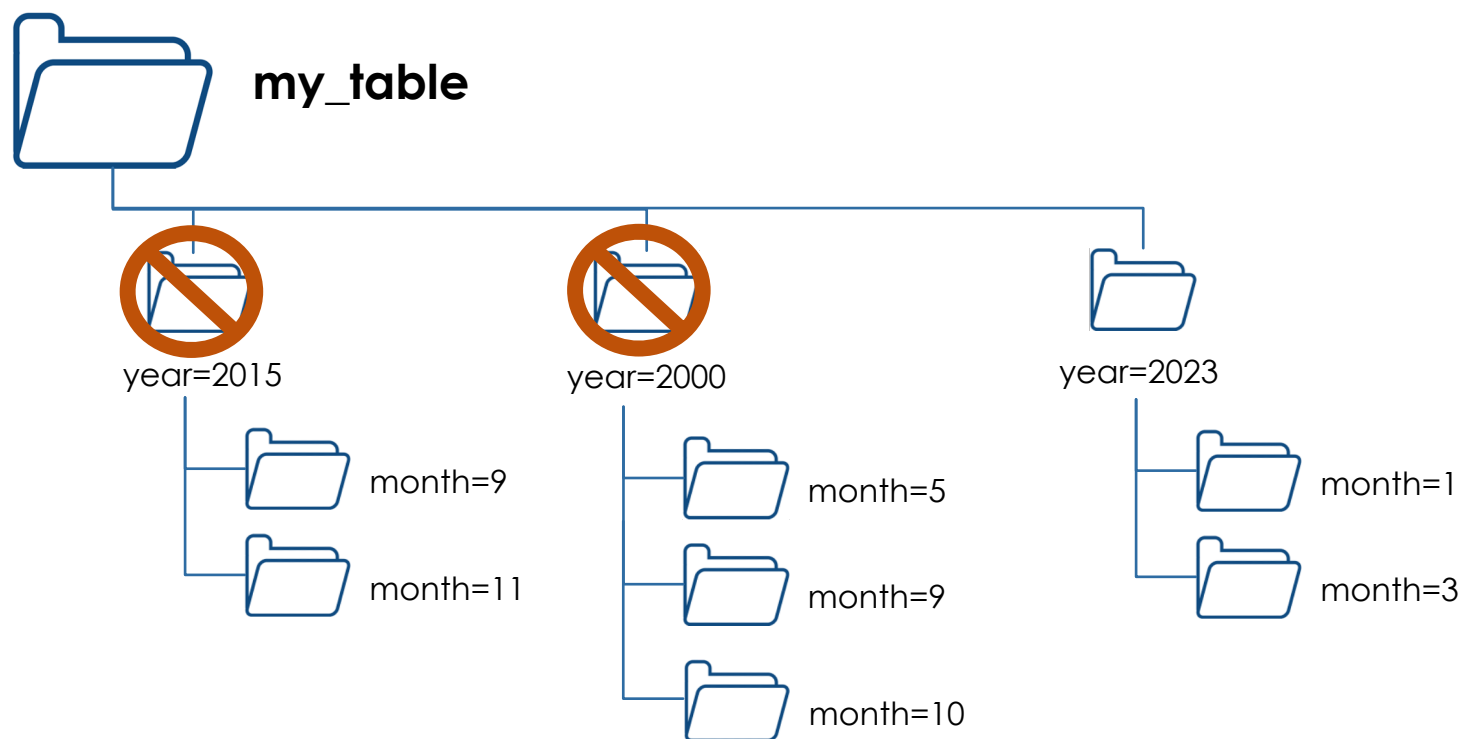
DELETE FROM my_table

Where year < 2023
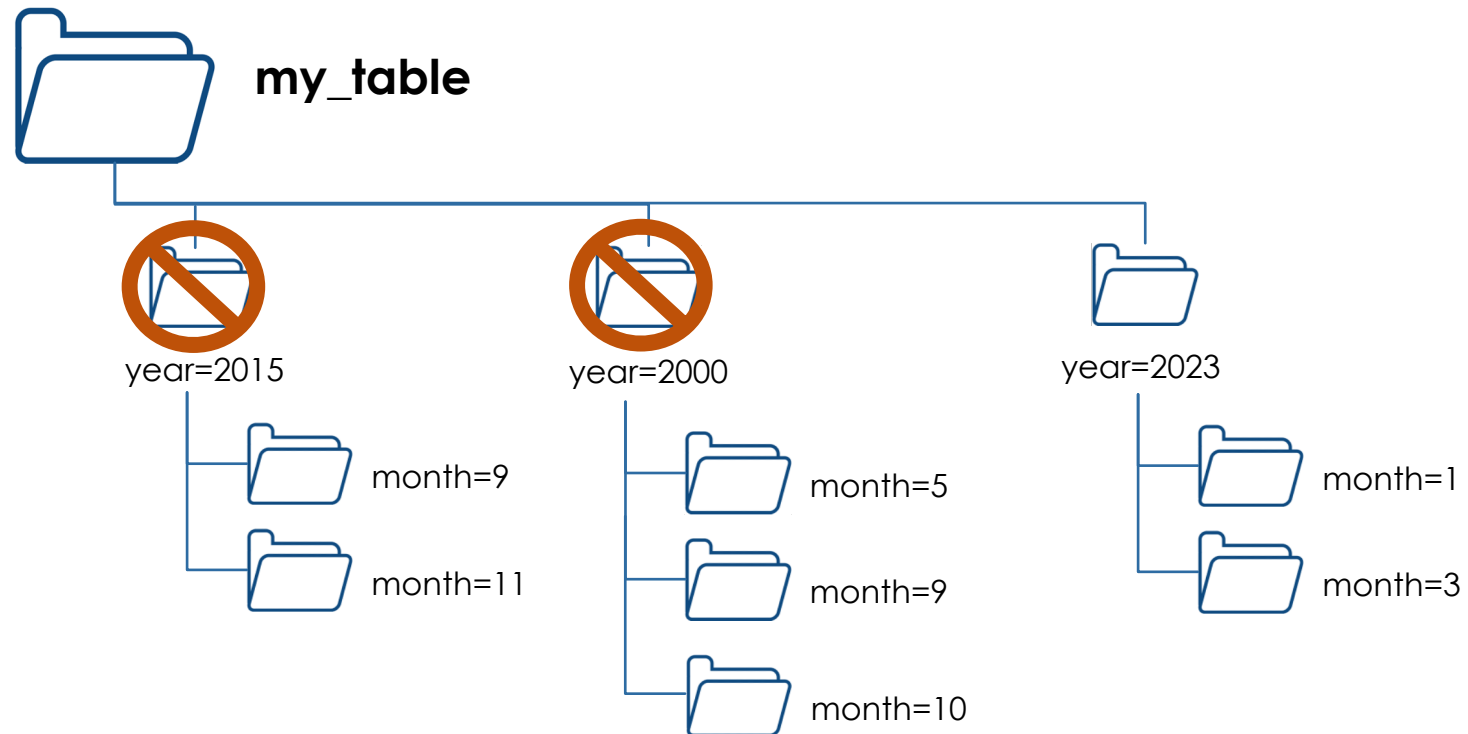
Deleting at Partition Boundaries

```
spark.readStream

    .option("ignoreDeletes", True)

    .table("my_table")
```

Deleting at Partition Boundaries

VACUUM my_table