1st February

Spark Dataframe

Handling Data types in SparK

focus on how Spark handles different data Types

dates
string

Data Type	Common Issues
String	Handling inconsistent cases, trimming spaces, and invalid encodings.
Number	Conversion issues, dealing with nulls, and handling precision for floats/doubles.
Date	Parsing formats, handling invalid dates, and timezone mismatches.
Boolean	Variations in input representation (`True/False`, `1/0`, `yes/no`).

customer_id	name	city	registration_date	amount_spent	is_active
1	John Doe	Bangalore	2023-01-15	152.75	True
2	Jane Smith	Delhi	2023-05-20	89.50	False
3	Robert Brown	Mumbai	InvalidDate	200.00	True
4	Linda White	Kolkata	2023-02-29		yes
5	Mike Green	Chennai	2023-08-10	NaN	1
6	Sarah Blue	Hyderabad	InvalidDate	300.40	No

Strings

numeric part

date ⟶ yyyy-mm-dd

Spark Table

Structured way to store and Query the data.

Data metadata / Schema

Spark table can be temporary or permanent

1. dataframe
2. Spark SQL
   ↳ Spark table

Type	Description
Temporary Table	Exists only for the duration of the session. Schema and data are stored in memory.
Global Temporary Table	Accessible across all sessions within the same application.
Persistent Table	Stored in a metastore (e.g., Hive), with data saved on disk (HDFS/S3). Accessible across sessions.

data = tbs

data

metadata

Spark
||
hive

Spark SQL ← higher level API

module in Spark use structured SQL Queries.

we can write SQL Queries on our Spark Table

Aspect	Managed Table	External Table
Ownership	Spark owns both data and metadata.	Spark owns only the metadata.
Data Location	Stored in the default warehouse directory.	Data resides at a user-defined external path.
Data Deletion	Dropping the table deletes both data and metadata.	Dropping the table deletes only metadata.
Insert Support	Supported	Supported
Update/Delete	Not supported in open-source Spark.	Not supported in open-source Spark.
Use Case	Suitable for single-user or Spark-managed data.	Suitable for shared, externally-managed data.

⇒ we normally don't do

Spark higher level APIs → SQL → Tables
⇒ dataframe

⇒ Caching ✓

Joins ✓

How can we create Spark dataframe ?

Method	Description	Example
`spark.read`	Reads data from external sources.	`spark.read.format("csv").load("/path")`
`spark.sql`	Executes SQL and returns a DataFrame.	`spark.sql("SELECT * FROM table")`
`spark.table`	Accesses an existing table.	`spark.table("customers")`
`spark.range`	Creates a single-column DataFrame with numbers in a range.	`spark.range(0, 10, 2)`
`spark.createDataFrame`	Converts local lists or RDDs to DataFrame.	`spark.createDataFrame(data, schema)`
`rdd.toDF()`	Converts an RDD to a DataFrame.	`rdd.toDF(["col1", "col2"])`
`spark.createDataFrame.toDF()`	Allows renaming columns after DataFrame creation.	`spark.createDataFrame(data).toDF("col1", "col2")`
Explicit Schema	Enforces a schema using `StructType` for better control.	`spark.createDataFrame(data, StructType([...]))`

Nested Data Structure

1. Complex Schema

→ Spark df operations
→ handling data
→ caching ☆