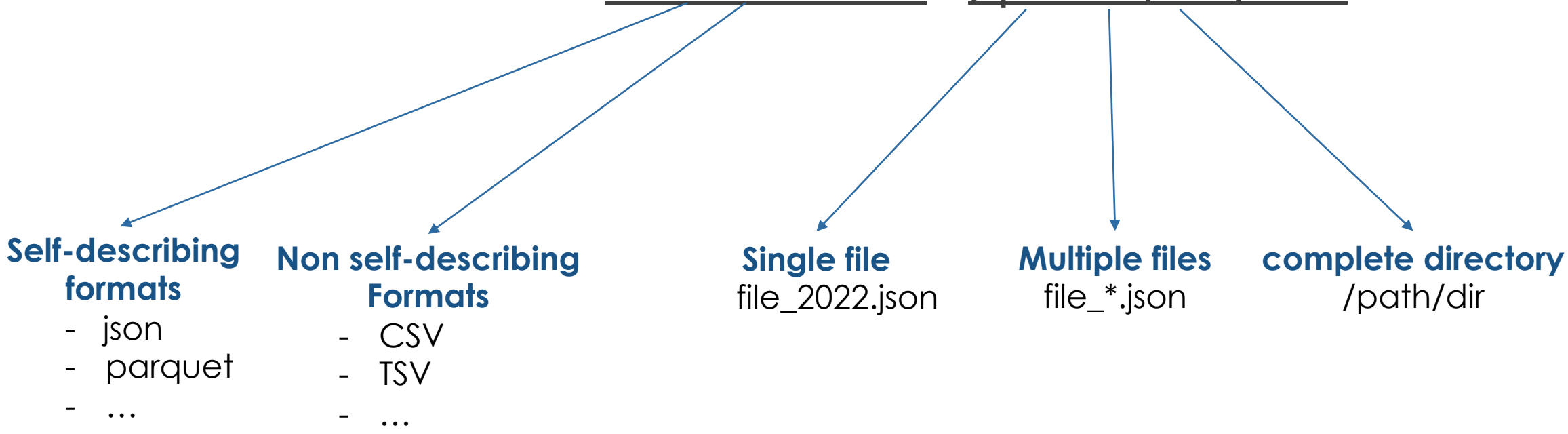
Querying Files

Learning Objectives

- Querying data files directly
- Extract files as raw contents
- Configure options of external sources
- Use CTAS statements to create Delta Lake tables

Querying Files Directly

SELECT * FROM file_format.`/path/to/file`



Example: JSON

```
SELECT * FROM json.`/path/file_name.json`
```

Raw data

- Extract text files as raw strings
  - Text-based files (JSON, CSV, TSV, and TXT formats)
  - SELECT * FROM **text**.`/path/to/file`
- Extract files as raw bytes
  - Images or unstructured data
  - SELECT * FROM **binaryFile**.`/path/to/file`

CTAS: Registering Tables from Files

- **CREATE TABLE** table_name

  AS SELECT * FROM file_format.`/path/to/file`

- Automatically infer schema information from query results
  - Do **Not** support manual schema declaration.
  - Useful for external data ingestion with well-defined schema

- Do **Not** support file options

Registering Tables on External Data Sources

- **CREATE TABLE** table_name
  (col_name1 col_type1, ...)

  USING data_source

  OPTIONS (key1 = val1, key2 = val2, ...)

  LOCATION = path

- External table
- Non-Delta table!

Example: CSV

- **CREATE TABLE** table_name
  (col_name1 col_type1, ...)
  USING CSV
  OPTIONS (header = "true",
  delimiter = ";")
  LOCATION = path

Example: Database

- **CREATE TABLE** table_name
  (col_name1 col_type1, ...)
  USING JDBC
  OPTIONS (url = "jdbc:sqlite://hostname:port",
  dbtable = "database.table",
  user = "username",
  password = ”pwd”)

Limitation

- It’s Not Delta table!
- We can not expect the performance guarantees associated with Delta Lake and Lakehouse
- Having a huge database table

Solution

- **CREATE TEMP VIEW** temp_view_name (col_name1 col_type1, ...)
  USING data_source
  OPTIONS (key1 = “val1”, key2 = “val2”, ..., path = “/path/to/files”)

- **CREATE TABLE** table_name
  AS SELECT * FROM temp_view_name