11 Spark SQL : Managed vs External Tables

1. What is a Managed Table?

A **managed table** is a table where Spark/Hive **manages both metadata AND data**.

✔ Spark stores metadata in the metastore

✔ Spark stores data inside the warehouse directory

`<warehouse_dir>/<table_name>/`

✔ If you DROP the table → data is deleted

Spark removes both:

- the table entry from the metastore
- the actual data files from disk

✔ Format defaults to Parquet (unless specified)

How to create a Managed Table

```
spark = (
    SparkSession.builder
    .master("spark://spark-master:7077")
    .appName("dataframe_apis")
    .config("spark.sql.warehouse.dir", "/data/spark-warehouse")
    .config("spark.sql.catalogImplementation", "hive")
    .config("javax.jdo.option.ConnectionURL", "jdbc:derby:/data/metastore_db;create=true")
    .getOrCreate()
)

#sql
```

```
CREATE TABLE orders (
  id INT,
  amount DOUBLE
);
or
CREATE TABLE orders (
  id INT,
  amount DOUBLE
) USING PARQUET/CSV;

# using dataframe :
df.write.saveAsTable("orders")
df.write.format("csv").saveAsTable("orders_csv")

# without hive catalog spark will create table metadta with spark catalog w
hich stores metadata in memory

spark.sql("show extended orders")
```

When to use Managed Tables

Use managed tables when:

✔️ You want Spark to fully manage data lifecycle

✔️ You want simple SQL workflows

---

2. What is an External Table?

An **external table** is a table where:

✔️ Spark stores ONLY metadata in the metastore

✔️ Spark does NOT own the data

✔️ Data lives at the given LOCATION

✔️ DROP TABLE deletes only metadata, NOT data files

How to create an External Table

```
#sql
CREATE TABLE orders_ext (
  id INT,
  amount DOUBLE
)
USING PARQUET
LOCATION '/data/orders_external';

# using dataframe
df.write.format("parquet").save("/data/orders_external")

df.write \
  .format("parquet") \
  .option("path", "/data/orders_external/") \
  .saveAsTable("orders_ext")

spark.sql("""
CREATE TABLE orders_ext USING PARQUET
LOCATION '/data/orders_external'
""")

df_orders.write \
  .format("parquet") \
  .option("path", "/data/orders_external/") \
  .mode("append") \
  .saveAsTable("orders_ext5")
```

Important

If `LOCATION` /path is provided → table is always external.

When to use External Tables

Use external tables when:

✔️ Data is shared across many tools (Spark, Hive, Presto, etc.)

✔️ Data already exists and you just want to “register” it

✔️ You don’t want accidental DROP TABLE to delete data

✔️ You store data on S3 / ADLS / GCS

Key Differences (Very Important)

Feature	Managed Table	External Table
Who owns data?	Spark owns data	User owns data
Where is data stored?	Warehouse directory	User defined LOCATION
DROP TABLE	Deletes metadata + data	Deletes metadata only
Schema stored in	Metastore	Metastore
Data deletion responsibility	Spark	User
Default format	Parquet	None (depends on files)
Persistence	Yes	Yes
Good for big data lakes	❌	✔️

```
# existing databases
spark.sql("show databases").show()

# create new database
spark.sql("create database mydb")

spark.sql("use mydb")

# create a persistent managed table
spark.sql("create table mydb.orders_data  (customer_id int ,order_date strin
g,order_id int,sales double,state string, status string) ")

spark.sql("create table mydb.orders_data  (customer_id int ,order_date strin
g,order_id int,sales double,state string, status string  ) using csv")

# insert data
spark.sql("insert into mydb.orders_data select * from  global_temp.orders")

# see table details
```

```
spark.sql("describe mydb.orders_data").show()
spark.sql("describe extended mydb.orders_data").show(truncate = False)

spark.sql("drop table mydb.orders_data") → both data and metatadta will b
e deleted

# external table
spark.sql("create table orders_data  (order_id int,customer_id int, sales dou
ble, order_date string,status string, state string  ) using csv location '/data/s
park-warehouse/orders_c'")

spark.sql("truncate table orders_data") -- not allowed
spark.sql("drop table orders_data") -- drop the data
```

Note : you can not delete or update data ..only inserts and overwrite