

Create a new notebook deltalake-demo-4

Use the deltalake_catalog created in the last session

If you are using a free edition then you can proceed with creating a managed table.
(if you are using azure databricks then you follow the steps shown in the video)

```
Just now (3s) 3

%sql

DROP TABLE IF EXISTS orders_managed;

CREATE OR REPLACE TABLE orders_managed (
  order_id BIGINT,
  sku STRING,
  product_name STRING,
  product_category STRING,
  qty INT,
  unit_price DECIMAL(10,2)
)
```

```
%sql
DESCRIBE HISTORY deltalake_catalog.default.orders_managed
See performance (1)

_sqldf: pyspark.sql.connect.dataframe.DataFrame = [version: long, timestamp: timestamp ... 13 more fields]

Table + 
1 00+00... 720133137164... technicalsupport@trendytech... CREATE OR REPLACE TABLE > {"partitionBy":[], "clusterBy":[], "description":null, "isManaged":true"}
```

```
%sql
INSERT INTO orders_managed VALUES
(1, 'SKU-1001', 'Wireless Mouse', 'Electronics', 2, 799.00),
(2, 'SKU-2001', 'Yoga Mat', 'Fitness', 1, 1199.00),
(3, 'SKU-3001', 'Notebook A5', 'Stationery', 5, 49.50),
(4, 'SKU-4001', 'Coffee Mug', 'Kitchen', 3, 299.00),
(5, 'SKU-5001', 'LED Bulb', 'Electronics', 4, 149.99);

> See performance (1)

▶ _sqldf: pyspark.sql.connect.DataFrame = [num_affected_rows: long, num_inserted_rows: long]

Table +
+-----+-----+
| 1  | num_affected_rows | num_inserted_rows |
+-----+-----+
| 1  | 5                | 5                |
+-----+-----+
```



```
%sql
DESCRIBE HISTORY deltalake_catalog.default.orders_managed
> See performance (1)

▶ _sqldf: pyspark.sql.connect.DataFrame = [version: long, timestamp: timestamp ... 13 more fields]

Table +
+-----+-----+-----+-----+-----+-----+
| 1  | timestamp       | userId          | userName        | operation      | operationParameters |
+-----+-----+-----+-----+-----+-----+
| 1  | 2025-09-09T09:25:59.000+00... | 720133137164... | technicalsupport@trendytech... | WRITE          | > {"mode": "Append", "statsOnLoad": true} |
| 2  | 2025-09-09T09:25:38.000+00... | 720133137164... | technicalsupport@trendytech... | CREATE OR REPLACE TABLE | > {"partitionBy":[], "clusterBy":[]}, "des
```

But unlike Premium/Pro editions, free editions sometimes don't populate the Location field in DESCRIBE EXTENDED.

The table is still a managed Delta table, just that the UI/SQL output doesn't show the path.

```
%sql
UPDATE deltalake_catalog.default.orders_managed
SET qty = 3
where order_id = 1;
> See performance (1)
```

▶ _sqldf: pyspark.sql.connect.DataFrame = [num_affected_rows: long]

Table +

	1 num_affected_rows
1	1

Just now (2s)

12

SQL 🗑️ ⚡ ⚡ ⚡

```
%sql
DESCRIBE HISTORY deltalake_catalog.default.orders_managed
> See performance (1)
```

Optimiz

▶ _sqldf: pyspark.sql.connect.DataFrame = [version: long, timestamp: timestamp ... 13 more fields]

Table +

Q Y H D

	1 version	1 timestamp	1 userId	1 userName	1 operation	1 operationParameter
1	3	2025-09-09T09:41:00.000+00:00	720133137164...	technicalsupport@trendytech....	OPTIMIZE	> {"predicate": "[]", "auto"}
2	2	2025-09-09T09:40:58.000+00:00	720133137164...	technicalsupport@trendytech....	UPDATE	> {"predicate": "[("order_id", "=> 1)]", "auto"}
3	1	2025-09-09T09:25:59.000+00:00	720133137164...	technicalsupport@trendytech....	WRITE	> {"mode": "Append", "staging"}
4	0	2025-09-09T09:25:38.000+00:00	720133137164...	technicalsupport@trendytech....	CREATE OR REPLACE TABLE	> {"partitionBy": "[]", "clu"}

Just now (2s)

14

SQL

```
%sql
INSERT OVERWRITE TABLE orders_managed VALUES
(1, 'SKU-1001', 'Wireless Mouse', 'Electronics', 3, 799.00),
(3, 'SKU-3001', 'Notebook A5', 'Stationery', 5, 49.50),
(5, 'SKU-5001', 'LED Bulb', 'Electronics', 4, 149.99);
> See performance (1)
```

▶ _sqldf: pyspark.sql.connect.DataFrame = [num_affected_rows: long, num_inserted_rows: long]

Table +

	1 num_affected_rows	1 num_inserted_rows
1	3	3

Just now (2s) 16 SQL

```
%sql
SELECT DISTINCT _metadata.file_path
FROM orders_managed;
> See performance (1)
```

_sqldf: pyspark.sql.connect.DataFrame = [file_path: string]

Table + C

	A ^B file_path
1	> s3://dbstorage-prod-uj0ub/uc/b1f08865-0a64-468d-9e44-5c2b3310926f/5f864d67-9999-41f0-8c1d-da085226bcfc/_unitystorage/ca...

Just now (1s) 17 SQL

```
%sql
describe history orders_managed;
> See performance (1) Optimize
```

_sqldf: pyspark.sql.connect.DataFrame = [version: long, timestamp: timestamp ... 13 more fields]

Table +

	A ^B version	timestamp	A ^B userId	A ^B userName	A ^B operation	operationParameter
1	4	2025-09-09T09:49:17.000+00:00	720133137164...	technicalsupport@trendytech...	WRITE	> {"mode":"Overwrite",
2	3	2025-09-09T09:41:00.000+00:00	720133137164...	technicalsupport@trendytech...	OPTIMIZE	> {"predicate": "[]", "autoC
3	2	2025-09-09T09:40:58.000+00:00	720133137164...	technicalsupport@trendytech...	UPDATE	> {"predicate": "\\"(order
4	1	2025-09-09T09:25:59.000+00:00	720133137164...	technicalsupport@trendytech...	WRITE	> {"mode":"Append", "s
5	0	2025-09-09T09:25:28.000+00:00	720133137164	technicalsupport@trendytech	CREATE OR REPLACE TABLE	> J"partitionBy": "[]", "re

Delta lake session 2:

Just now (1s) 18

```
%sql
select * from orders_managed;
```

_sqldf: pyspark.sql.connect.DataFrame = [order_id: long, sku: string ... 4 more fields]

Table + Q Y E □

	order_id	sku	product_name	product_category	qty	unit_price
1	1	SKU-1001	Wireless Mouse	Electronics	3	799.00
2	3	SKU-3001	Notebook A5	Stationery	5	49.50
3	5	SKU-5001	LED Bulb	Electronics	4	149.99

Just now (3s) 19

```
%sql
select * from orders_managed VERSION AS OF 1;
```

See performance (1) Optimize

_sqldf: pyspark.sql.connect.DataFrame = [order_id: long, sku: string ... 4 more fields]

Table + Q Y E □

	order_id	sku	product_name	product_category	qty	unit_price
1	1	SKU-1001	Wireless Mouse	Electronics	2	799.00
2	2	SKU-2001	Yoga Mat	Fitness	1	1199.00
3	3	SKU-3001	Notebook A5	Stationery	5	49.50
4	4	SKU-4001	Coffee Mug	Kitchen	3	299.00
5	5	SKU-5001	LED Bulb	Electronics	4	149.99

5 rows | 2.84s runtime Refreshed now

Python equivalent code for above sql code:

```
orders_df = spark.sql("SELECT * FROM orders_managed VERSION AS OF 2")
orders_df.show()
```

```
%sql
select * from orders_managed VERSION AS OF 2;
See performance (1)
```

Optimize

_sqldf: pyspark.sql.connect.DataFrame = [order_id: long, sku: string ... 4 more fields]

Table + Q Y E □

	order_id	sku	product_name	product_category	qty	unit_price
1	2	SKU-2001	Yoga Mat	Fitness	1	1199.00
2	3	SKU-3001	Notebook A5	Stationery	5	49.50
3	4	SKU-4001	Coffee Mug	Kitchen	3	299.00
4	5	SKU-5001	LED Bulb	Electronics	4	149.99
5	1	SKU-1001	Wireless Mouse	Electronics	3	799.00

5 rows | 2.83s runtime Refreshed now

```
%sql
select * from orders_managed@v2;
> See performance (1)
▶ _sqldf: pyspark.sql.connect.DataFrame = [order_id: long, sku: string ... 4 more fields]
```

Table + C

	order_id	sku	product_name	product_category	qty	unit_price
1	2	SKU-2001	Yoga Mat	Fitness	1	1199.00
2	3	SKU-3001	Notebook A5	Stationery	5	49.50
3	4	SKU-4001	Coffee Mug	Kitchen	3	299.00
4	5	SKU-5001	LED Bulb	Electronics	4	149.99
5	1	SKU-1001	Wireless Mouse	Electronics	3	799.00

5 rows | 2.55s runtime

```
orders_df = spark.sql("SELECT * FROM orders_managed VERSION AS OF 2")
orders_df.show()
```

sql
select * from orders_managed timestamp as of '2025-09-09T09:35:55.000+00:00';
> See performance (1)

	order_id	sku	product_name	product_category	qty	unit_price
1	1	SKU-1001	Wireless Mouse	Electronics	2	799.00
2	2	SKU-2001	Yoga Mat	Fitness	1	1199.00
3	3	SKU-3001	Notebook A5	Stationery	5	49.50
4	4	SKU-4001	Coffee Mug	Kitchen	3	299.00
5	5	SKU-5001	LED Bulb	Electronics	4	149.99

5 rows | 2.66s runtime Refresh

```
orders_df = spark.sql("""
    SELECT * FROM orders_managed TIMESTAMP AS OF '2025-09-09T09:35:55.000+00:00'
""")
orders_df.show()
```



Just now (1s)

22

```
df_latest = spark.read.table("orders_managed")
df_latest.show()
> See performance \(1\)
▶ df_latest: pyspark.sql.connect.dataframe.DataFrame = [order_id: long, sku: string ... 4 more fields]
+-----+-----+-----+-----+
|order_id|sku|product_name|product_category|qty|unit_price|
+-----+-----+-----+-----+
|1|SKU-1001|Wireless Mouse|Electronics|3|799.00|
|3|SKU-3001|Notebook A5|Stationery|5|49.50|
|5|SKU-5001|LED Bulb|Electronics|4|149.99|
+-----+-----+-----+-----+
```



Just now (1s)

23

```
df_v2 = spark.read.option("versionAsOf",2).table("orders_managed")
df_v2.show()
▶ df_v2: pyspark.sql.connect.dataframe.DataFrame = [order_id: long, sku: string ... 4 more fields]
+-----+-----+-----+-----+
|order_id|sku|product_name|product_category|qty|unit_price|
+-----+-----+-----+-----+
|2|SKU-2001|Yoga Mat|Fitness|1|1199.00|
|3|SKU-3001|Notebook A5|Stationery|5|49.50|
|4|SKU-4001|Coffee Mug|Kitchen|3|299.00|
|5|SKU-5001|LED Bulb|Electronics|4|149.99|
|1|SKU-1001|Wireless Mouse|Electronics|3|799.00|
+-----+-----+-----+-----+
```



Just now (1s)

24

```
df_v2 = spark.read.format("delta") \
.option("timestampAsOf",'2025-09-09T9:35:55.000+00:00') \
.table("orders_managed")

df_v2.show()
▶ df_v2: pyspark.sql.connect.dataframe.DataFrame = [order_id: long, sku: string ... 4 more fields]
+-----+-----+-----+-----+
|order_id|sku|product_name|product_category|qty|unit_price|
+-----+-----+-----+-----+
|1|SKU-1001|Wireless Mouse|Electronics|2|799.00|
|2|SKU-2001|Yoga Mat|Fitness|1|1199.00|
|3|SKU-3001|Notebook A5|Stationery|5|49.50|
|4|SKU-4001|Coffee Mug|Kitchen|3|299.00|
|5|SKU-5001|LED Bulb|Electronics|4|149.99|
+-----+-----+-----+-----+
```

Just now (2s) 25 SQL ⚡ ⚡ ⚡ ⚡

```
%sql
describe history orders_managed;
> See performance (1) Optimiz
```

_sqldf: pyspark.sql.connect.DataFrame = [version: long, timestamp: timestamp ... 13 more fields]

Table +

version	timestamp	userId	userName	operation	operationParameters
4	2025-09-09T09:49:17.000+00:00	720133137164...	technicalsupport@trendytech...	WRITE	> {"mode":"Overwrite",
3	2025-09-09T09:41:00.000+00:00	720133137164...	technicalsupport@trendytech...	OPTIMIZE	> {"predicate":[], "aut
2	2025-09-09T09:40:58.000+00:00	720133137164...	technicalsupport@trendytech...	UPDATE	> {"predicate":["\\(orde
1	2025-09-09T09:25:59.000+00:00	720133137164...	technicalsupport@trendytech...	WRITE	> {"mode":"Append", "
0	2025-09-09T09:25:30.000+00:00	720133137164...		CREATE OR REPLACE TABLE	> {"mode": "Append", "t

3 minutes ago (9s) 26 SQL ⚡ ⚡ ⚡ ⚡

```
%sql
RESTORE TABLE orders_managed TO VERSION AS OF 1;
```

_sqldf: pyspark.sql.connect.DataFrame = [table_size_after_restore: long, num_of_files_after_restore: long ... 4 more fields]

Table +

table_size_after_restore	num_of_files_after_restore	num_removed_files	num_restored_files	removed_files_size
1922		1	1	1

spark.sql("RESTORE TABLE orders_managed TO VERSION AS OF 1")

Just now (2s) 25 SQL ⚡ ⚡ ⚡ ⚡

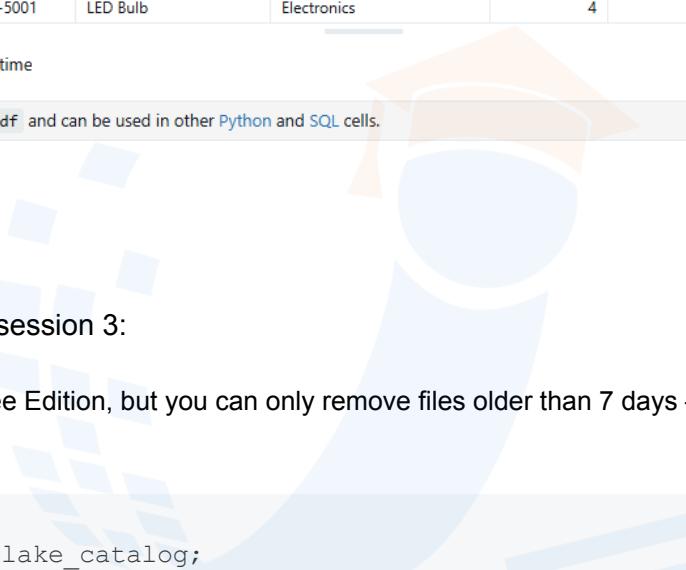
```
%sql
describe history orders_managed;
```

_sqldf: pyspark.sql.connect.DataFrame = [version: long, timestamp: timestamp ... 13 more fields]

Table +

version	timestamp	userId	userName	operation	operationParameters
5	2025-09-09T10:16:41.000+00:00	720133137164...	technicalsupport@trendytech...	RESTORE	> {"version":1, "times
4	2025-09-09T09:49:17.000+00:00	720133137164...	technicalsupport@trendytech...	WRITE	> {"mode":"Overwrite"
3	2025-09-09T09:41:00.000+00:00	720133137164...	technicalsupport@trendytech...	OPTIMIZE	> {"predicate":[], "aut
2	2025-09-09T09:40:58.000+00:00	720133137164...	technicalsupport@trendytech...	UPDATE	> {"predicate":["\\(orde
1	2025-09-09T09:25:59.000+00:00	720133137164...	technicalsupport@trendytech...	WRITE	> {"mode":"Append", "
0	2025-09-09T09:25:30.000+00:00	720133137164...		CREATE OR REPLACE TABLE	> {"mode": "Append", "t

Since we have restored to old version, we should see 5 records -



Just now (1s) 28 SQL ⚡ ⏺ ⏹ ⏷

```
%sql
select * from orders_managed;
```

_sqldf: pyspark.sql.connect.DataFrame = [order_id: long, sku: string ... 4 more fields]

	order_id	sku	product_name	product_category	qty	unit_price
1	1	SKU-1001	Wireless Mouse	Electronics	2	799.00
2	2	SKU-2001	Yoga Mat	Fitness	1	1199.00
3	3	SKU-3001	Notebook A5	Stationery	5	49.50
4	4	SKU-4001	Coffee Mug	Kitchen	3	299.00
5	5	SKU-5001	LED Bulb	Electronics	4	149.99

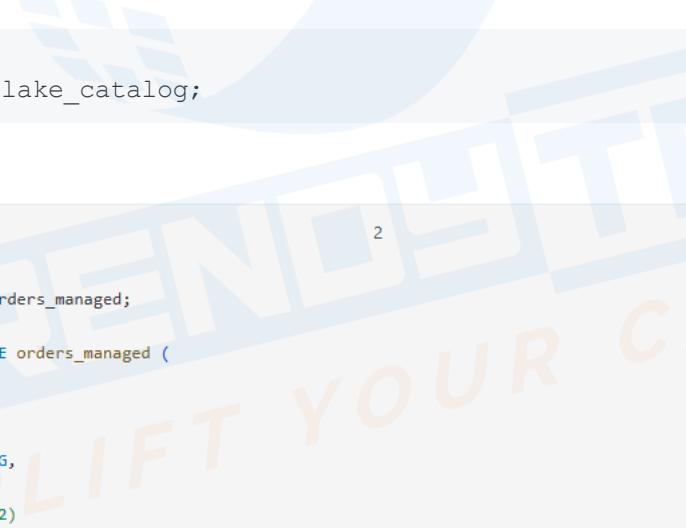
5 rows | 1.42s runtime Refreshed now

This result is stored as _sqldf and can be used in other Python and SQL cells.

Delta lake compute session 3:

VACUUM works in Free Edition, but you can only remove files older than 7 days — retention cannot be shortened

```
%sql
use catalog deltalake_catalog;
```



Just now (4s) 2

```
%sql
DROP TABLE IF EXISTS orders_managed;

CREATE OR REPLACE TABLE orders_managed (
    order_id BIGINT,
    sku STRING,
    product_name STRING,
    product_category STRING,
    qty INT,
    unit_price DECIMAL(10,2)
)
TBLPROPERTIES (
    delta.autoOptimize.optimizeWrite = false,
    delta.autoOptimize.autoCompact = false
)
```

See performance (2)

Pyspark equivalent code for above sql code:

```
# Drop table if it exists
spark.sql("DROP TABLE IF EXISTS orders_managed")
```

```

# Create or replace table with schema and TBLPROPERTIES
spark.sql("""
CREATE OR REPLACE TABLE orders_managed (
    order_id BIGINT,
    sku STRING,
    product_name STRING,
    product_category STRING,
    qty INT,
    unit_price DECIMAL(10,2)
)
TBLPROPERTIES (
    delta.autoOptimize.optimizeWrite = false,
    delta.autoOptimize.autoCompact = false
)
""")

```

Just now (2s) 3 SQL

```
%sql
SHOW TBLPROPERTIES orders_managed;
> See performance (1)
```

_sqldf: pyspark.sql.connect.DataFrame = [key: string, value: string]

	key	value
1	delta.autoOptimize.autoCompact	false
2	delta.autoOptimize.optimizeWri...	false
3	delta.enableDeletionVectors	true
4	delta.feature.appendOnly	supported
5	delta.feature.deletionVectors	supported
6	delta.feature.invariants	supported
7	delta.minReaderVersion	3
8	delta.minWriterVersion	7

8 rows | 1.76s runtime

```

props_df = spark.sql("SHOW TBLPROPERTIES orders_managed")
props_df.show(truncate=False)

```

Just now (6s) 4 SQL

```
%sql
INSERT INTO orders_managed VALUES
(1, 'SKU-1001', 'Wireless Mouse', 'Electronics', 2, 799.00),
(2, 'SKU-2001', 'Yoga Mat', 'Fitness', 1, 1199.00),
(3, 'SKU-3001', 'Notebook A5', 'Stationery', 5, 49.50),
(4, 'SKU-4001', 'Coffee Mug', 'Kitchen', 3, 299.00),
(5, 'SKU-5001', 'LED Bulb', 'Electronics', 4, 149.99);
> See performance (1)
```

_sqldf: pyspark.sql.connect.DataFrame = [num_affected_rows: long, num_inserted_rows: long]

Table +

	num_affected_rows	num_inserted_rows
1	5	5

Just now (3s) 5 SQL

```
%sql
DESCRIBE HISTORY deltalake_catalog.default.orders_managed
> See performance (1) Optimize
```

_sqldf: pyspark.sql.connect.DataFrame = [version: long, timestamp: timestamp ... 13 more fields]

Table +

	version	timestamp	userId	userName	operation	operationParameters
1	1	2025-09-11T08:48:50.000+00:00	720133137164...	technicalsupport@trendytech....	WRITE	> {"mode":"Append","stat...
2	0	2025-09-11T08:40:55.000+00:00	720133137164...	technicalsupport@trendytech....	CREATE OR REPLACE TABLE	> {"partitionBy":[],"cluste...

Just now (2s) 6

```
%sql
describe detail deltalake_catalog.default.orders_managed
> See performance (1) Optimize
```

_sqldf: pyspark.sql.connect.DataFrame = [format: string, id: string ... 15 more fields]

Table +

	format	id	name	description	location	created
1	delta	fb3ec147-5128-4bae-a1d4-4e103e84e5...	deltalake_catalog.default.orders_manag...	null		2025-09-11T08:40:55.000+00:00

8

```
%sql
UPDATE deltalake_catalog.default.orders_managed
SET qty = 3
WHERE order_id = 1;
> See performance \(1\)
```

_sqldf: pyspark.sql.connect.DataFrame = [num_affected_rows: long]

Table	+
1²₃ num_affected_rows	1
1	1

9

```
%sql
INSERT OVERWRITE TABLE orders_managed VALUES
(1, 'SKU-1001', 'Wireless Mouse', 'Electronics', 3, 799.00),
(3, 'SKU-3001', 'Notebook A5', 'Stationery', 5, 49.50),
(5, 'SKU-5001', 'LED Bulb', 'Electronics', 4, 149.99);
> See performance \(1\)
```

_sqldf: pyspark.sql.connect.DataFrame = [num_affected_rows: long, num_inserted_rows: long]

Table	+
1²₃ num_affected_rows	1²₃ num_inserted_rows
1	3

10

```
%sql
describe history orders_managed;
> See performance \(1\)
```

_sqldf: pyspark.sql.connect.DataFrame = [version: long, timestamp: timestamp ... 13 more fields]

Table	+ 				
1²₃ version	4 timestamp	A^B_C userId	A^B_C userName	A^B_C operation	operationParameter
1	2025-09-11T08:59:31.000+00:00	720133137164...	technicalsupport@trendytech....	WRITE	> {"mode":"Overwrite", "partitionColumn": "order_id", "dataFormat": "PARQUET"}
2	2025-09-11T08:57:35.000+00:00	720133137164...	technicalsupport@trendytech....	OPTIMIZE	> {"predicate": "order_id = 1", "autoOptimizeLevel": "INTERVAL", "partitionColumn": "order_id", "dataFormat": "PARQUET"}
3	2025-09-11T08:57:33.000+00:00	720133137164...	technicalsupport@trendytech....	UPDATE	> {"predicate": "order_id = 1", "partitionColumn": "order_id", "dataFormat": "PARQUET"}
4	2025-09-11T08:48:50.000+00:00	720133137164...	technicalsupport@trendytech....	WRITE	> {"mode":"Append", "partitionColumn": "order_id", "dataFormat": "PARQUET"}
5	2025-09-11T08:48:50.000+00:00	720133137164...	technicalsupport@trendytech....	CREATE OR REPLACE TABLE	> {"mode": "Overwrite", "partitionColumn": "order_id", "dataFormat": "PARQUET"}

You still see OPTIMIZE in the history because on serverless clusters, Databricks runs background compaction jobs automatically for performance, even if Auto Optimize is disabled

Just now (2s) 11 SQL ⚡ ⚡

```
%sql
SELECT DISTINCT _metadata.file_path FROM orders_managed;
> See performance (1)
```

_sqldf: pyspark.sql.connect.DataFrame = [file_path: string]

Table +

	file_path
1	> s3://dbstorage-prod-uj0ub/uc/b1f08865-0a64-468d-9e44-5c2b3310926f/5f864d67-9999-41f0-8c1d-da085226bcfc/_unitystorage/ca...

```
file_paths = spark.read.table("orders_managed") \
    .selectExpr("DISTINCT _metadata.file_path")
```

```
file_paths.show(truncate=False)
```

Just now (3s) 12 SQL ⚡ ⚡

```
%sql
SELECT DISTINCT _metadata.file_path FROM orders_managed@v2;
> See performance (1)
```

_sqldf: pyspark.sql.connect.DataFrame = [file_path: string]

Table +

	file_path
1	> s3://dbstorage-prod-uj0ub/uc/b1f08865-0a64-468d-9e44-5c2b3310926f/5f864d67-9999-41f0-8c1d-da085226bcfc/_unitystorage/ca...
2	> s3://dbstorage-prod-uj0ub/uc/b1f08865-0a64-468d-9e44-5c2b3310926f/5f864d67-9999-41f0-8c1d-da085226bcfc/_unitystorage/ca...

```
file_paths_df = spark.sql("""
SELECT DISTINCT _metadata.file_path
FROM orders_managed VERSION AS OF 2
""")
```

```
file_paths_df.show(truncate=False)
```

Just now (10s) 13

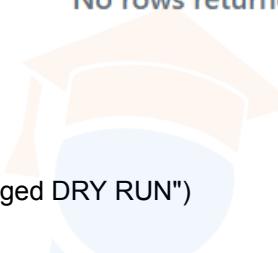
```
%sql
VACUUM orders_managed DRY RUN;
> See performance (1)

▶ _sqldf: pyspark.sql.connect.DataFrame = [path: string]

Table +
```

A	B	C	path

No rows returned



because there are no files created before 7 days

Pyspark equivalent code for above sql code:

```
vacuum_result = spark.sql("VACUUM orders_managed DRY RUN")
vacuum_result.show(truncate=False)
```

Last execution failed 14

```
1 %sql
2 VACUUM orders_managed retain 0 hours DRY RUN;
> See performance (1) ×1
```

i > requirement failed: Are you sure you would like to vacuum files with such a low retention period? If you have writers that are currently writing to this table, there is a risk that you may corrupt the state of your Delta table.

...

Last execution failed 14

```
1 %sql
2 VACUUM orders_managed retain 0 hours DRY RUN;
> See performance (1) ×1
```

i > requirement failed: Are you sure you would like to vacuum files with such a low retention period? If you have writers that are currently writing to this table, there is a risk that you may corrupt the state of your Delta table.

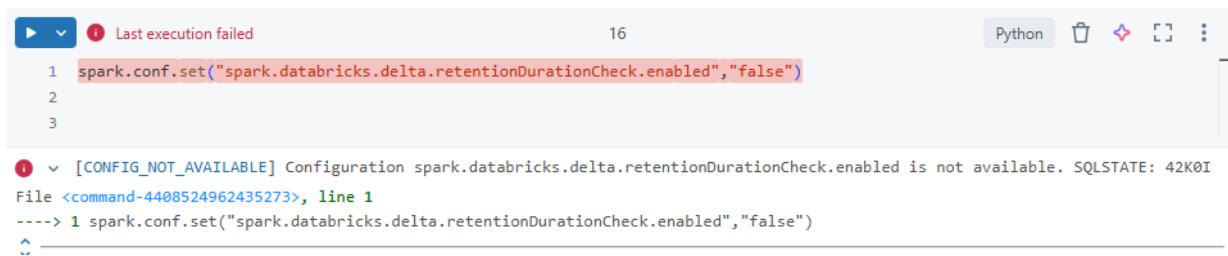
If you are certain that there are no operations being performed on this table, such as insert/upsert/delete/optimize, then you may turn off this check by setting:
spark.databricks.delta.retentionDurationCheck.enabled = false

If you are not sure, please use a value not less than "168 hours".

```
JVM stacktrace:
java.lang.IllegalArgumentException
    at scala.Predef$.require(Predef.scala:337)
    at com.databricks.sql.transaction.tahoe.commands.VacuumCommand$.checkRetentionPeriodSafety(VacuumCommand.scala:130)
```

Pyspark equivalent code for above sql code:

```
vacuum_result = spark.sql("VACUUM orders_managed RETAIN 0 HOURS DRY RUN")
vacuum_result.show(truncate=False)
```



```
1 spark.conf.set("spark.databricks.delta.retentionDurationCheck.enabled","false")
2
3
4 [CONFIG_NOT_AVAILABLE] Configuration spark.databricks.delta.retentionDurationCheck.enabled is not available. SQLSTATE: 42K01
File <command-4408524962435273>, line 1
--> 1 spark.conf.set("spark.databricks.delta.retentionDurationCheck.enabled","false")
```

Free edition of Databricks doesn't support changing certain Delta table configurations like spark.databricks.delta.retentionDurationCheck.enabled.

VACUUM works in Free Edition, but you can only remove files older than 7 days — retention cannot be shortened

Delta lake compute session 4: (need azure account)

```
%sql
use catalog deltalake_catalog;
```

```
%sql
DROP TABLE IF EXISTS orders_ext_01;

CREATE OR REPLACE TABLE orders_ext_01 (
order_id BIGINT,
sku STRING,
product_name STRING,
product_category STRING,
qty INT,
unit_price DECIMAL(10,2)
)
LOCATION
'abfss://externaldata@ttmystorageaccount001.dfs.core.windows.net/orders'
```

```
for i in range(100):    # 100 small commits – adjust smaller/larger as needed
    order_id = 1000 + i
    sku = f"SKU-BURST-{i}"
    name = f"Burst-{i}"
    category = "Demo"
    qty = 1
```

```
    price = float(i % 10 + 1) * 1.0
    sql = f"""
        INSERT INTO orders_ext_01 VALUES ({order_id}, '{sku}', '{name}',
        '{category}', {qty}, {price});
    """
    spark.sql(sql)
```

Above code is already in python

```
%sql  
DESCRIBE HISTORY orders_ext_01;
```

Pyspark:

```
checkpoint_path =  
"abfss://externaldata@ttmystorageaccount001.dfs.core.windows.net/orders/_delta_log/0000000000000000  
0072.checkpoint.parquet"
```

```
checkpoint_df = spark.read.format("parquet").load(checkpoint_path)
checkpoint_df.show(truncate=False)
```

```
%sql  
SELECT * FROM  
JSON.`abfss://externaldata@ttmystorageaccount001.dfs.core.windows.net/orders/_d  
elta log/00000000000000000000000000000073.00000000000000000000000000000078.compacted.json`
```

Pyspark:

```
compacted_path =  
"abfss://externaldata@ttmystorageaccount001.dfs.core.windows.net/orders/_delta_log/0000000000000000  
0073.0000000000000000078.compacted.json"
```

```
compacted_df = spark.read.json(compacted_path)  
compacted_df.show(truncate=False)
```

Session 5 - can follow steps followed by sir in the video

Session 6 - need azure account

Session 7 - can follow sir's video

Session 8:

You can create managed table using free edition -

▶ ✓ Just now (4s)

1

```
%sql  
use catalog deltalake_catalog;  
➤ See performance \(1\)
```

▶ ✓ Just now (2s)

2

```
%sql  
DROP TABLE IF EXISTS orders_ext_01;  
  
CREATE OR REPLACE TABLE orders_ext_01 (  
order_id BIGINT,  
sku STRING,  
product_name STRING,  
product_category STRING,  
qty INT,  
unit_price DECIMAL(10,2)  
)  
➤ See performance \(2\)
```

```
%sql  
INSERT INTO orders_ext_01 VALUES  
(1, 'SKU-1001', 'Wireless Mouse', 'Electronics', 2, 799.00),  
(2, 'SKU-2001', 'Yoga Mat', 'Fitness', 1, 1199.00),  
(3, 'SKU-3001', 'Notebook A5', 'Stationery', 5, 49.50),  
(4, 'SKU-4001', 'Coffee Mug', 'Kitchen', 3, 299.00),  
(5, 'SKU-5001', 'LED Bulb', 'Electronics', 4, 149.99);
```

```
%sql  
DESCRIBE FORMATTED orders_ext_01
```

```
%sql  
ALTER TABLE orders_ext_01  
ALTER COLUMN order_id SET NOT NULL;
```

> [See performance \(1\)](#)

Optimize

```
%sql  
DESCRIBE HISTORY orders_ext_01;  
> See performance \(1\)
```

Optimize

▶ [_sqldf: pyspark.sql.connect.DataFrame = \[version: long, timestamp: timestamp ... 13 more fields\]](#)

Table	+	Q	T	E	D
1 ² version	3 ³ timestamp	A ^B userId	A ^B userName	A ^B operation	operationParameters
1	3 2025-09-16T10:42:18.000+00:00	720133137164...	technicalsupport@trendytech...	CHANGE COLUMN	> {"column": "\\"name\\"", "mode": "Append", "st...
2	2 2025-09-16T10:41:12.000+00:00	720133137164...	technicalsupport@trendytech...	WRITE	> {"mode": "Append", "st...
3	1 2025-09-16T10:40:54.000+00:00	720133137164...	technicalsupport@trendytech...	WRITE	> {"mode": "Append", "st...
4	0 2025-09-16T10:40:08.000+00:00	720133137164...	technicalsupport@trendytech...	CREATE OR REPLACE TABLE	> {"partitionBy": "[]", "clust...

```
%sql  
ALTER TABLE orders_ext_01  
ADD CONSTRAINT chk_qty_price CHECK (qty > 0 AND unit_price > 0)  
> See performance \(1\)
```

Optimize

```
%sql  
DESCRIBE HISTORY orders_ext_01;  
> See performance \(1\)
```

Optimize

▶ [_sqldf: pyspark.sql.connect.DataFrame = \[version: long, timestamp: timestamp ... 13 more fields\]](#)

Table	+	Q	T	E	D
1 ² version	4 ³ timestamp	A ^B userId	A ^B userName	A ^B operation	operationParameters
1	4 2025-09-17T08:25:22.000+00:00	720133137164...	technicalsupport@trendytech...	ADD CONSTRAINT	> {"name": "chk_qty_price", "st...
2	3 2025-09-16T10:42:18.000+00:00	720133137164...	technicalsupport@trendytech...	CHANGE COLUMN	> {"column": "\\"name\\"", "mode": "Append", "st...
3	2 2025-09-16T10:41:12.000+00:00	720133137164...	technicalsupport@trendytech...	WRITE	> {"mode": "Append", "st...
4	1 2025-09-16T10:40:54.000+00:00	720133137164...	technicalsupport@trendytech...	WRITE	> {"mode": "Append", "st...

```
1 %sql  
2 INSERT INTO orders_ext_01 VALUES (6, 'SKU-6001', 'Faulty Item', 'Misc', 0, -10.00);  
> See performance \(1\)
```

Optimize

[DELTA_VIOLATE_CONSTRAINT_WITH_VALUES] CHECK constraint chk_qty_price ((qty > 0) AND (unit_price > 0)) violated by row with values:
- qty : 0
- unit_price : -10.00 SQLSTATE: 23001

[Diagnose error](#)

Assistant Quick Fix: ON ▾

```

▶ ✓ Just now (1s) 10
%sql
-- Create a temp source table (simulate an incoming batch with one bad row)
CREATE OR REPLACE TEMP VIEW incoming AS
SELECT * FROM VALUES
(2, 'SKU-2001', 'Yoga Mat - new', 'Fitness', 2, 1099.00),
(6, 'SKU-6001', 'Invalid Item', 'Misc', 0, -10.00)
AS t(order_id, sku, product_name, product_category, qty, unit_price);

> See performance \(1\)

```

The screenshot shows two sessions in a database interface:

- Session 10:** Contains the SQL code to create a temporary view named 'incoming' from a VALUES clause. The code includes two rows: (2, 'SKU-2001', 'Yoga Mat - new', 'Fitness', 2, 1099.00) and (6, 'SKU-6001', 'Invalid Item', 'Misc', 0, -10.00). A link to 'See performance (1)' is present.
- Session 11:** Contains a MERGE statement into a table named 'orders_ext_01'. The statement uses 'incoming' as the source and 'src' as the target. It includes conditions for WHEN MATCHED and WHEN NOT MATCHED. A note indicates a constraint violation: '[DELTA_VIOLATE_CONSTRAINT_WITH_VALUES] CHECK constraint chk_qty_price ((qty > 0) AND (unit_price > 0)) violated by row with values: - qty : 0 - unit_price : -10.00 SQLSTATE: 23001'. A 'Diagnose error' button is available.

```
%sql
select sum(qty) as total_qty from orders_ext_01;
update orders_ext_01 set qty = qty + 10 where order_id = 1
```

Session 9:

Sir has used serverless, you can follow the same steps.

Python equivalent code for the sql in session 9 are given below -

```
%sql
DROP TABLE IF EXISTS orders_ext_01;

CREATE OR REPLACE TABLE orders_ext_01 (
order_id BIGINT,
sku STRING,
product_name STRING,
product_category STRING,
qty INT,
```

```
unit_price DECIMAL(10, 2)
)

Pyspark:
# Drop table if it exists
spark.sql("DROP TABLE IF EXISTS orders_ext_01")

# Create or replace table with schema
spark.sql("""
CREATE OR REPLACE TABLE orders_ext_01 (
  order_id BIGINT,
  sku STRING,
  product_name STRING,
  product_category STRING,
  qty INT,
  unit_price DECIMAL(10,2)
)
""")
```

```
%sql
INSERT INTO orders_ext_01 VALUES
(1, 'SKU-1001', 'Wireless Mouse', 'Electronics', 2, 799.00),
(2, 'SKU-2001', 'Yoga Mat', 'Fitness', 1, 1199.00),
(3, 'SKU-3001', 'Notebook A5', 'Stationery', 5, 49.50),
(4, 'SKU-4001', 'Coffee Mug', 'Kitchen', 3, 299.00),
(5, 'SKU-5001', 'LED Bulb', 'Electronics', 4, 149.99);
```

```
Pyspark:
spark.sql("""
INSERT INTO orders_ext_01 VALUES
(1, 'SKU-1001', 'Wireless Mouse', 'Electronics', 2, 799.00),
(2, 'SKU-2001', 'Yoga Mat', 'Fitness', 1, 1199.00),
(3, 'SKU-3001', 'Notebook A5', 'Stationery', 5, 49.50),
(4, 'SKU-4001', 'Coffee Mug', 'Kitchen', 3, 299.00),
(5, 'SKU-5001', 'LED Bulb', 'Electronics', 4, 149.99)
""")
```

```
%sql
DESCRIBE FORMATTED orders_ext_01
```

```
Pyspark:
desc_df = spark.sql("DESCRIBE FORMATTED orders_ext_01")
desc_df.show(truncate=False)
```

```
%sql
ALTER TABLE orders_ext_01
ALTER COLUMN order_id SET NOT NULL;
```

```
Pyspark:
```

```
spark.sql("""
ALTER TABLE orders_ext_01
ALTER COLUMN order_id SET NOT NULL
""")
```

```
%sql
DESCRIBE HISTORY orders_ext_01;
```

Pyspark:

```
history_df = spark.sql("DESCRIBE HISTORY orders_ext_01")
history_df.show(truncate=False)
```

```
%sql
ALTER TABLE orders_ext_01
ADD CONSTRAINT chk_qty_price CHECK (qty > 0 AND unit_price > 0)
```

Pyspark:

```
spark.sql("""
ALTER TABLE orders_ext_01
ADD CONSTRAINT chk_qty_price CHECK (qty > 0 AND unit_price > 0)
""")
```

```
%sql
DESCRIBE HISTORY orders_ext_01;
```

Pyspark:

```
history_df = spark.sql("DESCRIBE HISTORY orders_ext_01")
history_df.show(truncate=False)
```

```
%sql
INSERT INTO orders_ext_01 VALUES (6, 'SKU-6001', 'Faulty Item', 'Misc', 0,
-10.00);
```

Pyspark:

```
spark.sql("""
    INSERT INTO orders_ext_01 VALUES
    (6, 'SKU-6001', 'Faulty Item', 'Misc', 0, -10.00)
""")
```

```
%sql
-- Create a temp source table (simulate an incoming batch with one bad row)
CREATE OR REPLACE TEMP VIEW incoming AS
SELECT * FROM VALUES
(2, 'SKU-2001', 'Yoga Mat - new', 'Fitness', 2, 1099.00),
(6, 'SKU-6001', 'Invalid Item', 'Misc', 0, -10.00)
AS t(order_id, sku, product_name, product_category, qty, unit_price);
```

Pyspark:

```
from pyspark.sql import Row
```

```

# Create DataFrame directly
incoming_data = [
    Row(2, "SKU-2001", "Yoga Mat - new", "Fitness", 2, 1099.00),
    Row(6, "SKU-6001", "Invalid Item", "Misc", 0, -10.00)    ]

incoming_df = spark.createDataFrame(incoming_data,
    ["order_id", "sku", "product_name", "product_category", "qty", "unit_price"])

# Register as temp view (so you can use it in SQL)
incoming_df.createOrReplaceTempView("incoming")

# Quick check
spark.sql("SELECT * FROM incoming").show()

```

```
%sql
MERGE INTO orders_ext_01 AS TARGET
USING incoming AS src
ON TARGET.order_id = src.order_id
WHEN MATCHED THEN UPDATE SET *
WHEN NOT MATCHED THEN INSERT *;
```

Pyspark:

```
# Filter only valid rows
valid_source = spark.table("incoming").filter("qty > 0 AND unit_price > 0")
```

```
# Perform merge
target = DeltaTable.forName(spark, "orders_ext_01")
```

```
( target.alias("TARGET")
    .merge(
        valid_source.alias("src"),
        "TARGET.order_id = src.order_id"
    )
    .whenMatchedUpdateAll()
    .whenNotMatchedInsertAll()
    .execute()
)
```

```
%sql
select sum(qty) as total_qty from orders_ext_01;
update orders_ext_01 set qty = qty + 10 where order_id = 1
```

Pyspark:

```
spark.sql("""
UPDATE orders_ext_01
SET qty = qty + 10
```

```
WHERE order_id = 1
""")
```

Session 10:

Sir has used serverless, you can follow the same steps.

Python equivalent code for the sql in session 10 are given below -

```
from decimal import Decimal
from pyspark.sql.types import StructType, StructField, LongType, StringType,
IntegerType, DecimalType

# Sample data with Decimal for unit_price
data = [
    (1, "SKU-1001", "Wireless Mouse", "Electronics", 2, Decimal("799.00")),
    (2, "SKU-2001", "Yoga Mat", "Fitness", 1, Decimal("1199.00")),
    (3, "SKU-3001", "Notebook A5", "Stationery", 5, Decimal("49.50")),
    (4, "SKU-4001", "Coffee Mug", "Kitchen", 3, Decimal("299.00")),
    (5, "SKU-5001", "LED Bulb", "Electronics", 4, Decimal("149.99"))
]

schema = StructType([
    StructField("order_id", LongType(), False),
    StructField("sku", StringType(), True),
    StructField("product_name", StringType(), True),
    StructField("product_category", StringType(), True),
    StructField("qty", IntegerType(), True),
    StructField("unit_price", DecimalType(10, 2), True)
])

df = spark.createDataFrame(data, schema=schema)

display(df)
df.printSchema()

volume_path = "/Volumes/deltalake_catalog/default/delta_volume1/ordersdata1"
df.write.format("delta").mode("overwrite").save(volume_path)
Is already in pyspark format

%sql
```

```
SELECT * FROM  
DELTA.`/Volumes/deltalake_catalog/default/delta_volume1/ordersdata1`  
  
Pyspark:  
df = spark.read.format("delta").load("/Volumes/deltalake_catalog/default/delta_volume1/ordersdata1")  
df.show()
```

```
%sql  
DESCRIBE HISTORY  
DELTA.`/Volumes/deltalake_catalog/default/delta_volume1/ordersdata1`  
  
Pyspark:  
history_df = spark.sql("")  
DESCRIBE HISTORY DELTA.`/Volumes/deltalake_catalog/default/delta_volume1/ordersdata1`  
""")  
  
history_df.show(truncate=False)
```

```
%sql  
ALTER TABLE  
DELTA.`/Volumes/deltalake_catalog/default/delta_volume1/ordersdata1`  
RENAME COLUMN qty TO quantity;  
  
Pyspark:  
spark.sql("")  
ALTER TABLE DELTA.`/Volumes/deltalake_catalog/default/delta_volume1/ordersdata1`  
RENAME COLUMN qty TO quantity  
""")
```

```
%sql  
ALTER TABLE  
DELTA.`/Volumes/deltalake_catalog/default/delta_volume1/ordersdata1`  
SET TBLPROPERTIES ('delta.columnMapping.mode' = 'name')  
  
Pyspark:  
spark.sql("")  
ALTER TABLE DELTA.`/Volumes/deltalake_catalog/default/delta_volume1/ordersdata1`  
SET TBLPROPERTIES ('delta.columnMapping.mode' = 'name')  
""")
```

```
%sql  
describe formatted  
DELTA.`/Volumes/deltalake_catalog/default/delta_volume1/ordersdata1`  
  
Pyspark:  
desc_df = spark.sql("")
```

```
DESCRIBE FORMATTED DELTA.`/Volumes/deltalake_catalog/default/delta_volume1/ordersdata1`  
""")  
desc_df.show(truncate=False)
```

```
%sql  
ALTER TABLE  
DELTA.`/Volumes/deltalake_catalog/default/delta_volume1/ordersdata1`  
DROP COLUMN product_category;
```

Pyspark:

```
spark.sql("""  
ALTER TABLE DELTA.`/Volumes/deltalake_catalog/default/delta_volume1/ordersdata1`  
DROP COLUMN product_category  
""")
```

