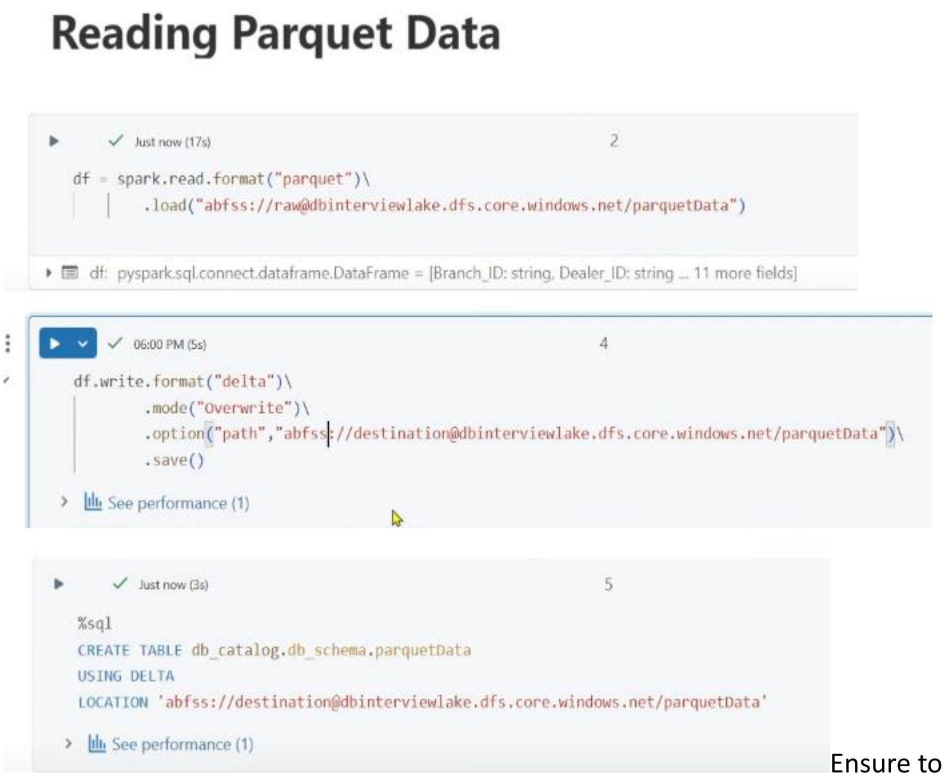
In Databricks Serverless, we cant see the Spark UI, we can only see the performance.

Q1) Read data from ADLS in parquet and write in delta format along with table



Reading Parquet Data

```
df = spark.read.format("parquet")\
        .load("abfss://raw@dbinterviewlake.dfs.core.windows.net/parquetData")
```

df: pyspark.sql.connect.dataframe.DataFrame = [Branch_ID: string, Dealer_ID: string ... 11 more fields]

```
df.write.format("delta")\
        .mode("Overwrite")\
        .option("path","abfss://destination@dbinterviewlake.dfs.core.windows.net/parquetData")\
        .save()
```

See performance (1)

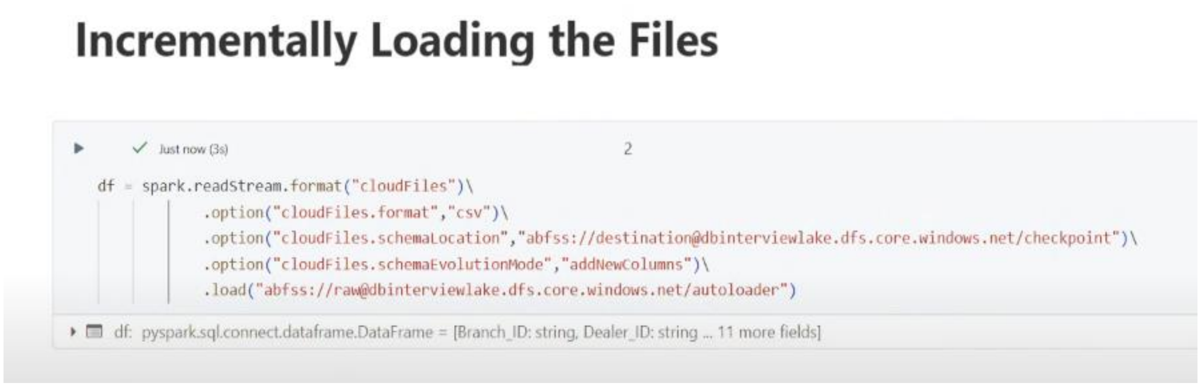
```
%sql
CREATE TABLE db_catalog.db_schema.parquetData
USING DELTA
LOCATION 'abfss://destination@dbinterviewlake.dfs.core.windows.net/parquetData'
```

See performance (1)

Ensure to give same path

You can also write saveAsTable instead of save to do in a single step

Q2) Incremental Data Loading → use Autoloader

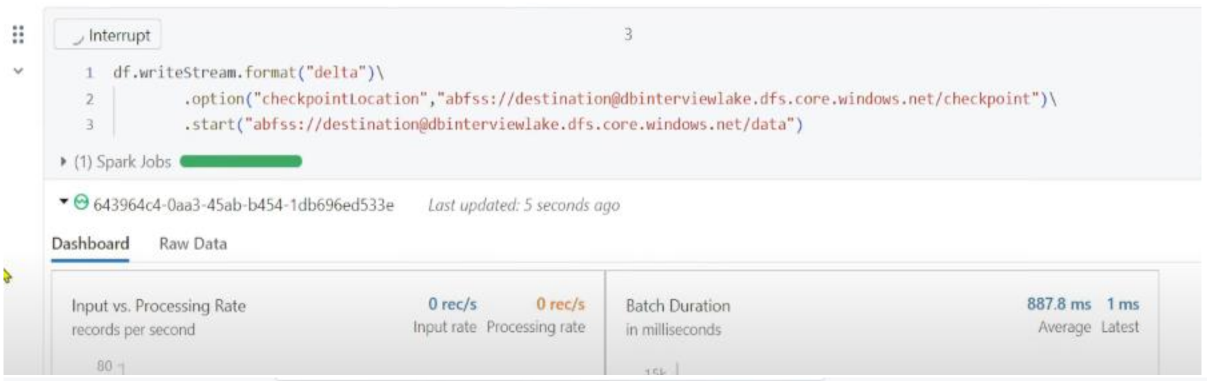


Incrementally Loading the Files

```
df = spark.readStream.format("cloudFiles")\
        .option("cloudFiles.format","csv")\
        .option("cloudFiles.schemaLocation","abfss://destination@dbinterviewlake.dfs.core.windows.net/checkpoint")\
        .option("cloudFiles.schemaEvolutionMode","addNewColumns")\
        .load("abfss://raw@dbinterviewlake.dfs.core.windows.net/autoloader")
```

df: pyspark.sql.connect.dataframe.DataFrame = [Branch_ID: string, Dealer_ID: string ... 11 more fields]

Schema Evolution happens in Autoloader by default, schemaEvolutionMode happens in Autoloader by default no need of explicitly mentioning it.



Interrupt

```
df.writeStream.format("delta")\
        .option("checkpointLocation","abfss://destination@dbinterviewlake.dfs.core.windows.net/checkpoint")\
        .start("abfss://destination@dbinterviewlake.dfs.core.windows.net/data")
```

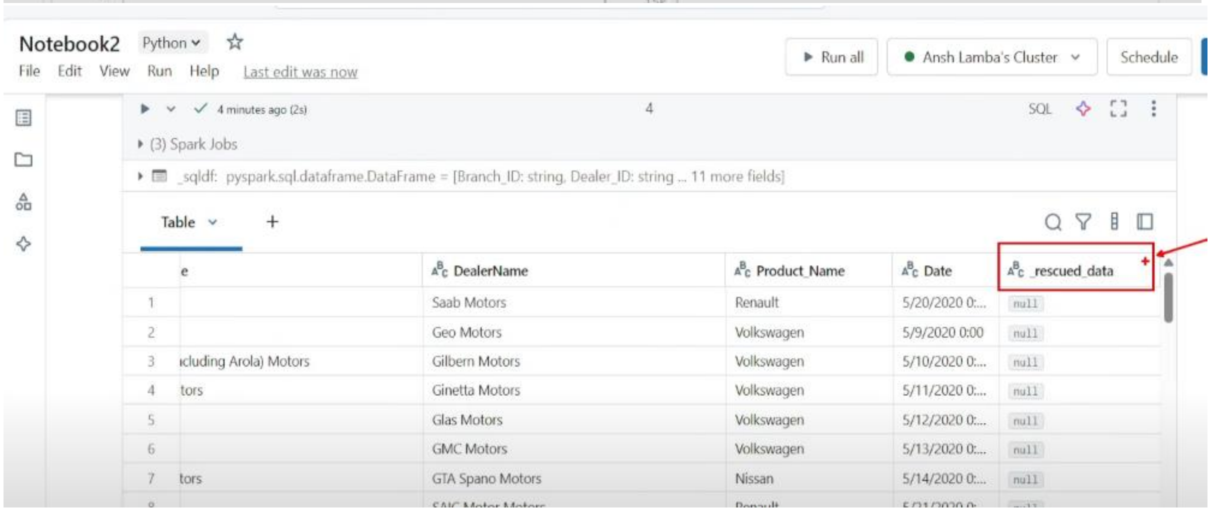
(1) Spark Jobs

643964c4-0aa3-45ab-b454-1db696ed533e   *Last updated: 5 seconds ago*

Dashboard   Raw Data

Input vs. Processing Rate, records per second: 0 rec/s Input rate, 0 rec/s Processing rate

Batch Duration in milliseconds: 887.8 ms Average, 1 ms Latest



Notebook2   Python

File Edit View Run Help   Last edit was now

Run all   Ansh Lamba's Cluster   Schedule

4 minutes ago (2s)

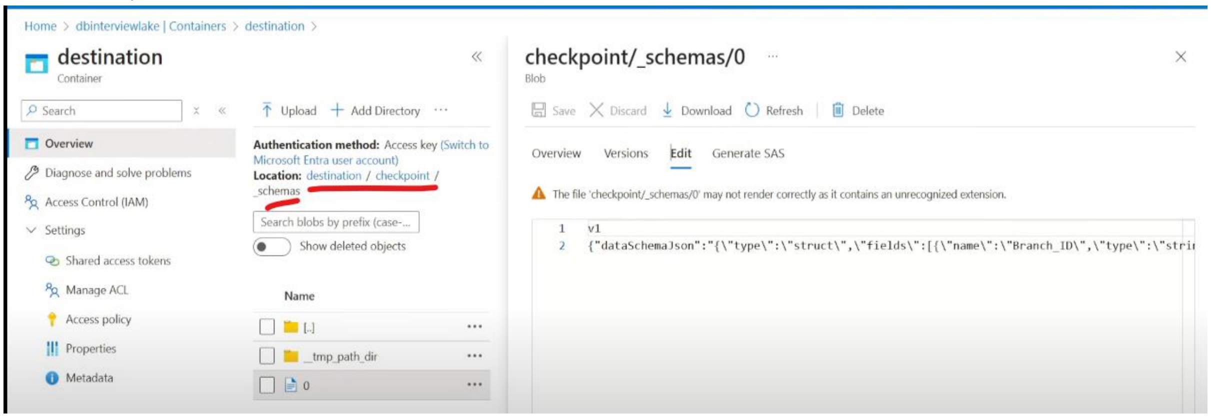
(3) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [Branch_ID: string, Dealer_ID: string ... 11 more fields]

Table

	e	DealerName	Product_Name	Date	_rescued_data
1		Saab Motors	Renault	5/20/2020 0:...	null
2		Geo Motors	Volkswagen	5/9/2020 0:00	null
3	ıcluding Arola) Motors	Gilbern Motors	Volkswagen	5/10/2020 0:...	null
4	tors	Ginetta Motors	Volkswagen	5/11/2020 0:...	null
5		Glas Motors	Volkswagen	5/12/2020 0:...	null
6		GMC Motors	Volkswagen	5/13/2020 0:...	null
7	tors	GTA Spano Motors	Nissan	5/14/2020 0:...	null

Schema is present in this location



Home > dbinterviewlake | Containers > destination >

destination
Container

Overview / Diagnose and solve problems / Access Control (IAM) / Settings / Shared access tokens / Manage ACL / Access policy / Properties / Metadata

Upload   Add Directory

Authentication method: Access key (Switch to Microsoft Entra user account)
Location: destination / checkpoint / _schemas

Show deleted objects

Name
[..]
_tmp_path_dir
0

checkpoint/_schemas/0
Blob

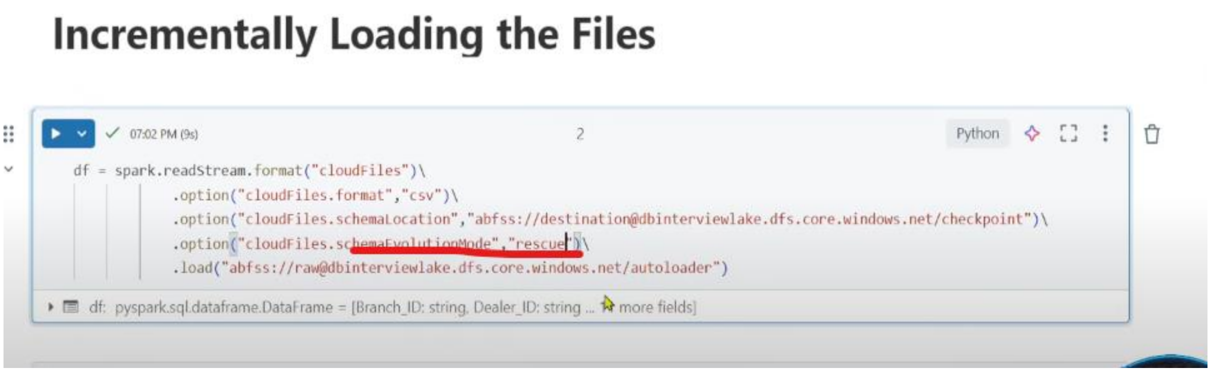
Save   Discard   Download   Refresh   Delete

Overview   Versions   Edit   Generate SAS

The file 'checkpoint/_schemas/0' may not render correctly as it contains an unrecognized extension.

```
v1
{"dataSchemaJson":"{\"type\":\"struct\",\"fields\":[{\"name\":\"Branch_ID\",\"type\":\"stri
```

Now lets say there is schema change,and you want to rescue those columns, in destination I don't want to add any further columns but I want to store those columns

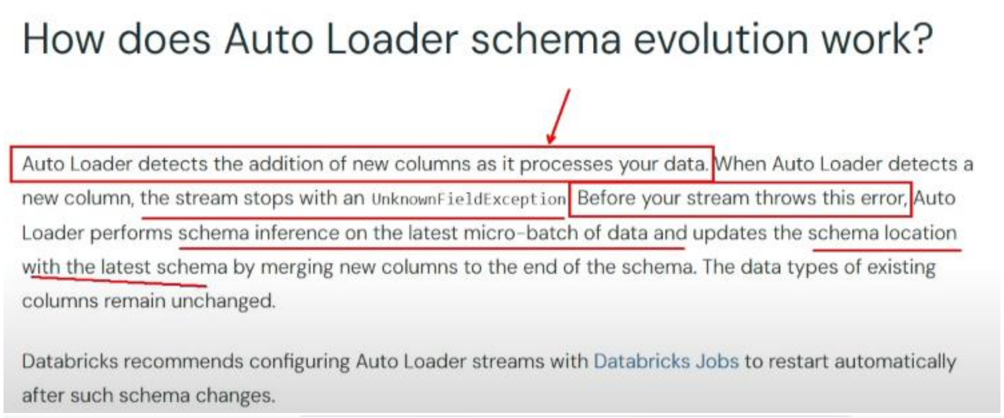


Incrementally Loading the Files

```
df = spark.readStream.format("cloudFiles")\
            .option("cloudFiles.format","csv")\
            .option("cloudFiles.schemaLocation","abfss://destination@dbinterviewlake.dfs.core.windows.net/checkpoint")\
            .option("cloudFiles.schemaEvolutionMode","rescue")\
            .load("abfss://raw@dbinterviewlake.dfs.core.windows.net/autoloader")
```

```
df: pyspark.sql.dataframe.DataFrame = [Branch_ID: string, Dealer_ID: string ... more fields]
```

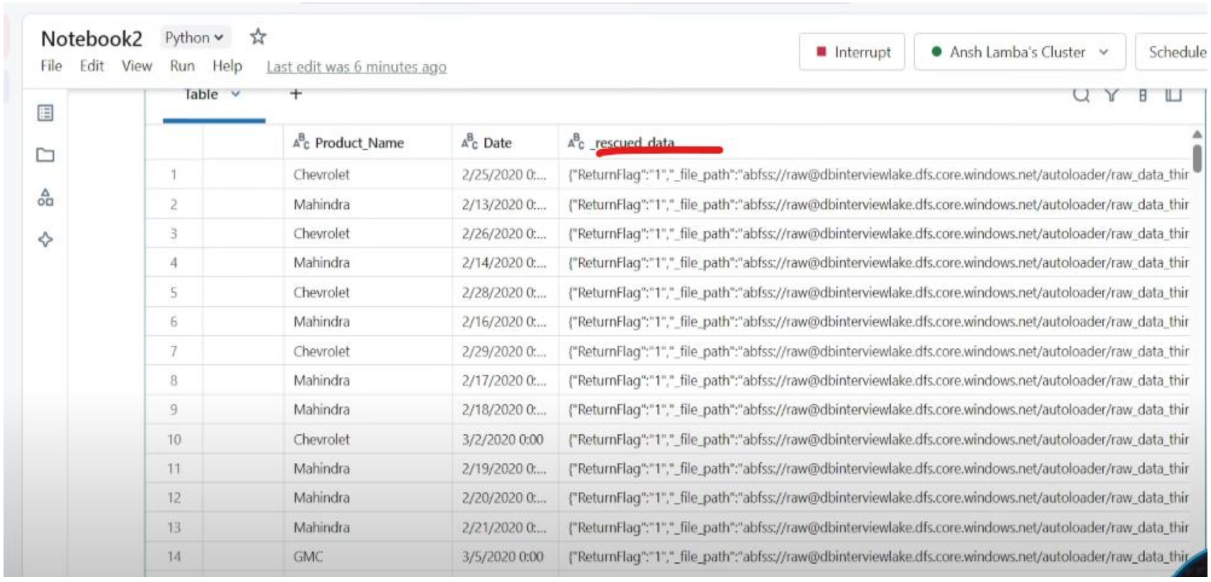
All the new columns can be found in rescued_data column



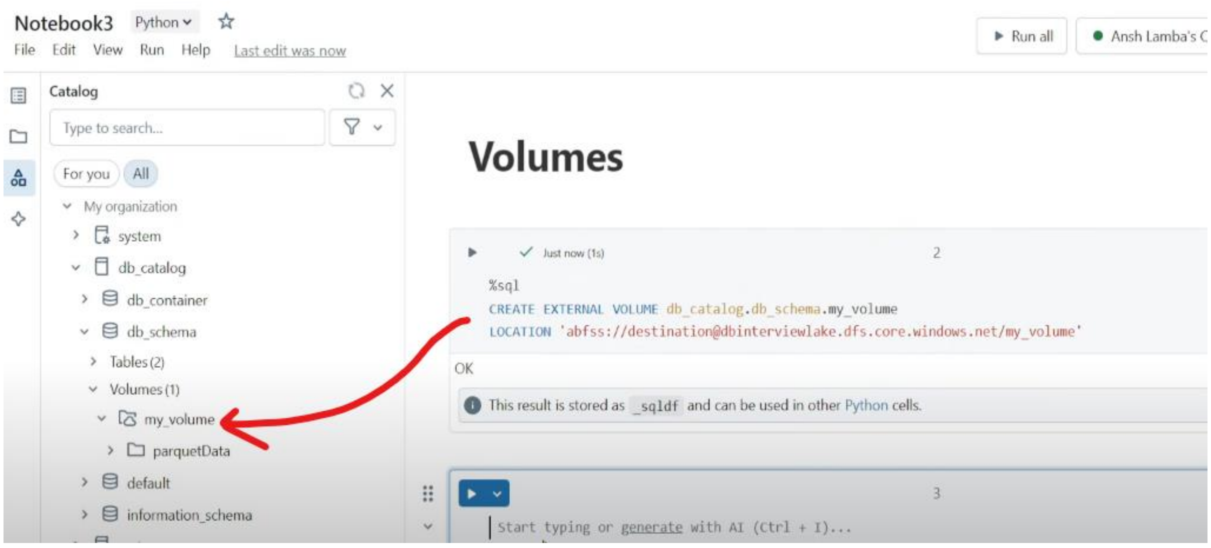
How does Auto Loader schema evolution work?

Auto Loader detects the addition of new columns as it processes your data. When Auto Loader detects a new column, the stream stops with an `UnknownFieldException`. Before your stream throws this error, Auto Loader performs schema inference on the latest micro-batch of data and updates the schema location with the latest schema by merging new columns to the end of the schema. The data types of existing columns remain unchanged.

Databricks recommends configuring Auto Loader streams with Databricks Jobs to restart automatically after such schema changes.



		Product_Name	Date	_rescued_data
1		Chevrolet	2/25/2020 0:...	{"ReturnFlag":"1","_file_path":"abfss://raw@dbinterviewlake.dfs.core.windows.net/autoloader/raw_data_thir
2		Mahindra	2/13/2020 0:...	{"ReturnFlag":"1","_file_path":"abfss://raw@dbinterviewlake.dfs.core.windows.net/autoloader/raw_data_thir
3		Chevrolet	2/26/2020 0:...	{"ReturnFlag":"1","_file_path":"abfss://raw@dbinterviewlake.dfs.core.windows.net/autoloader/raw_data_thir
4		Mahindra	2/14/2020 0:...	{"ReturnFlag":"1","_file_path":"abfss://raw@dbinterviewlake.dfs.core.windows.net/autoloader/raw_data_thir
5		Chevrolet	2/28/2020 0:...	{"ReturnFlag":"1","_file_path":"abfss://raw@dbinterviewlake.dfs.core.windows.net/autoloader/raw_data_thir
6		Mahindra	2/16/2020 0:...	{"ReturnFlag":"1","_file_path":"abfss://raw@dbinterviewlake.dfs.core.windows.net/autoloader/raw_data_thir
7		Chevrolet	2/29/2020 0:...	{"ReturnFlag":"1","_file_path":"abfss://raw@dbinterviewlake.dfs.core.windows.net/autoloader/raw_data_thir
8		Mahindra	2/17/2020 0:...	{"ReturnFlag":"1","_file_path":"abfss://raw@dbinterviewlake.dfs.core.windows.net/autoloader/raw_data_thir
9		Mahindra	2/18/2020 0:...	{"ReturnFlag":"1","_file_path":"abfss://raw@dbinterviewlake.dfs.core.windows.net/autoloader/raw_data_thir
10		Chevrolet	3/2/2020 0:00	{"ReturnFlag":"1","_file_path":"abfss://raw@dbinterviewlake.dfs.core.windows.net/autoloader/raw_data_thir
11		Mahindra	2/19/2020 0:...	{"ReturnFlag":"1","_file_path":"abfss://raw@dbinterviewlake.dfs.core.windows.net/autoloader/raw_data_thir
12		Mahindra	2/20/2020 0:...	{"ReturnFlag":"1","_file_path":"abfss://raw@dbinterviewlake.dfs.core.windows.net/autoloader/raw_data_thir
13		Mahindra	2/21/2020 0:...	{"ReturnFlag":"1","_file_path":"abfss://raw@dbinterviewlake.dfs.core.windows.net/autoloader/raw_data_thir
14		GMC	3/5/2020 0:00	{"ReturnFlag":"1","_file_path":"abfss://raw@dbinterviewlake.dfs.core.windows.net/autoloader/raw_data_thir



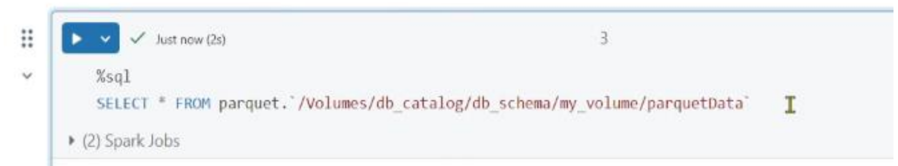
```
%sql
CREATE EXTERNAL VOLUME db_catalog.db_schema.my_volume
LOCATION 'abfss://destination@dbinterviewlake.dfs.core.windows.net/my_volume'
```

```
OK
```

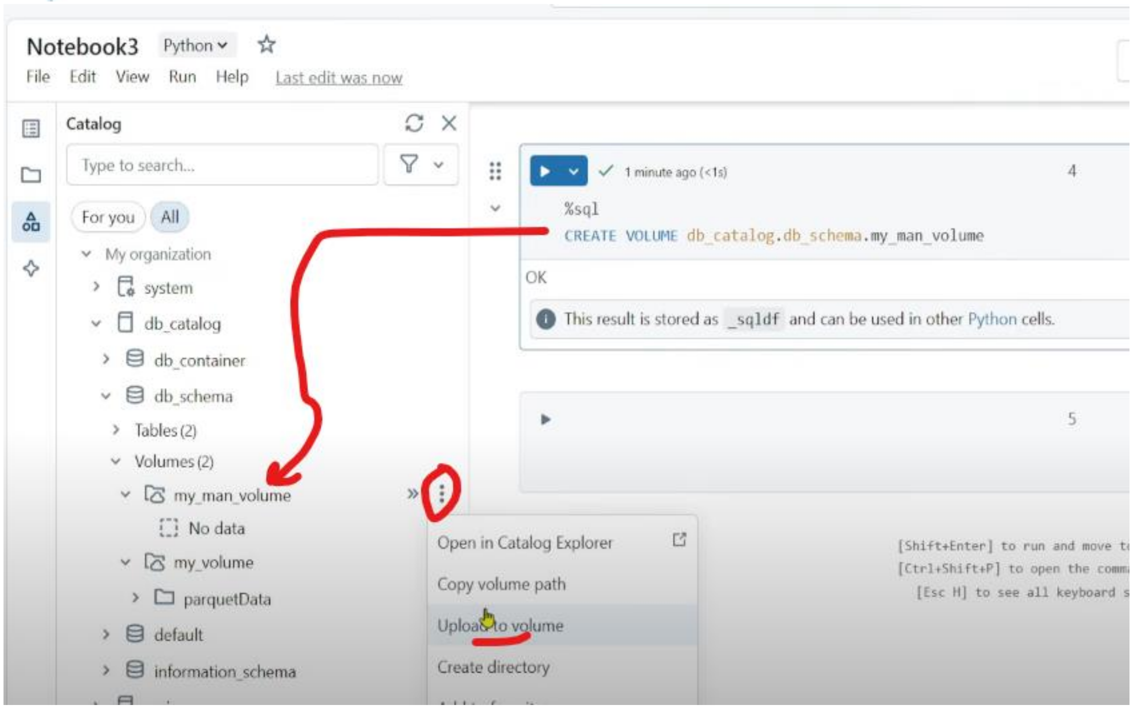
This result is stored as `_sqldf` and can be used in other Python cells.

```
Start typing or generate with AI (Ctrl + I)...
```

Volumes are like Files in Fabric LH



```
%sql
SELECT * FROM parquet.`/Volumes/db_catalog/db_schema/my_volume/parquetData`
```



```
%sql
CREATE VOLUME db_catalog.db_schema.my_man_volume
```

```
OK
```

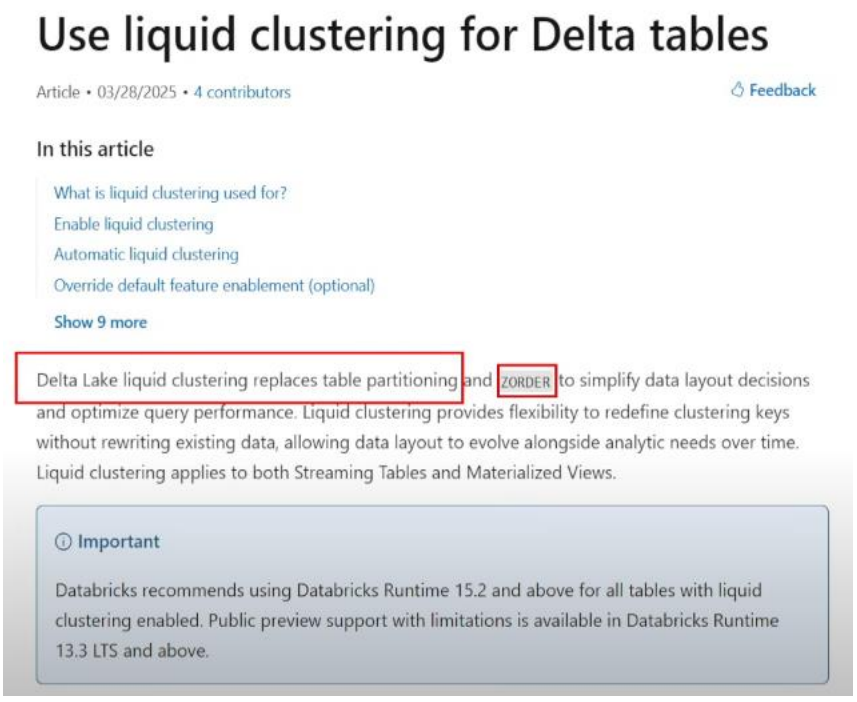
This result is stored as `_sqldf` and can be used in other Python cells.

```
[Shift+Enter] to run and move t
[Ctrl+Shift+P] to open the comm
[Esc H] to see all keyboard s
```

To insert data → Upload to Volume

Liquid Clustering

Liquid Clustering creates dynamic clusters on top of data, dynamic query behaviour



Use liquid clustering for Delta tables

Article • 03/28/2025 • 4 contributors

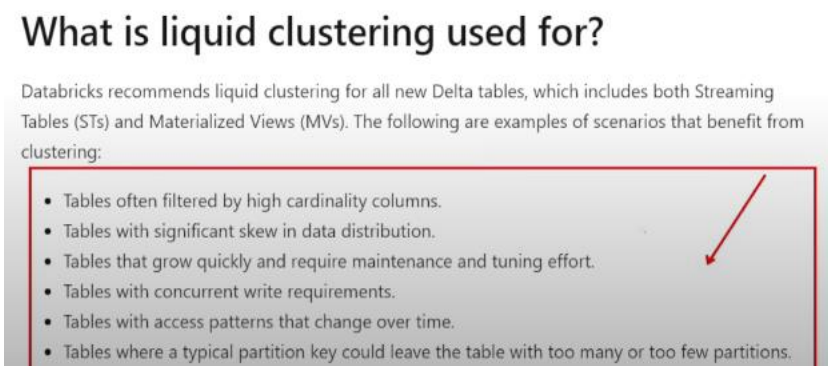
Feedback

In this article

Delta Lake liquid clustering replaces table partitioning and `ZORDER` to simplify data layout decisions and optimize query performance. Liquid clustering provides flexibility to redefine clustering keys without rewriting existing data, allowing data layout to evolve alongside analytic needs over time. Liquid clustering applies to both Streaming Tables and Materialized Views.

ⓘ Important

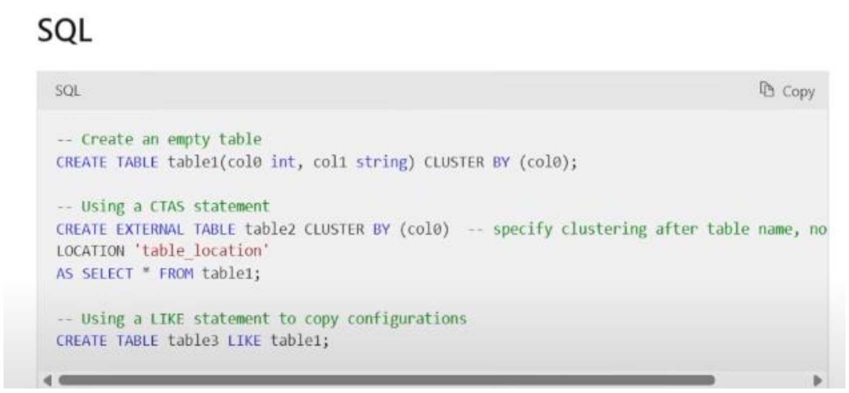
Databricks recommends using Databricks Runtime 15.2 and above for all tables with liquid clustering enabled. Public preview support with limitations is available in Databricks Runtime 13.3 LTS and above.



What is liquid clustering used for?

Databricks recommends liquid clustering for all new Delta tables, which includes both Streaming Tables (STs) and Materialized Views (MVs). The following are examples of scenarios that benefit from clustering:

- Tables often filtered by high cardinality columns.
- Tables with significant skew in data distribution.
- Tables that grow quickly and require maintenance and tuning effort.
- Tables with concurrent write requirements.
- Tables with access patterns that change over time.
- Tables where a typical partition key could leave the table with too many or too few partitions.



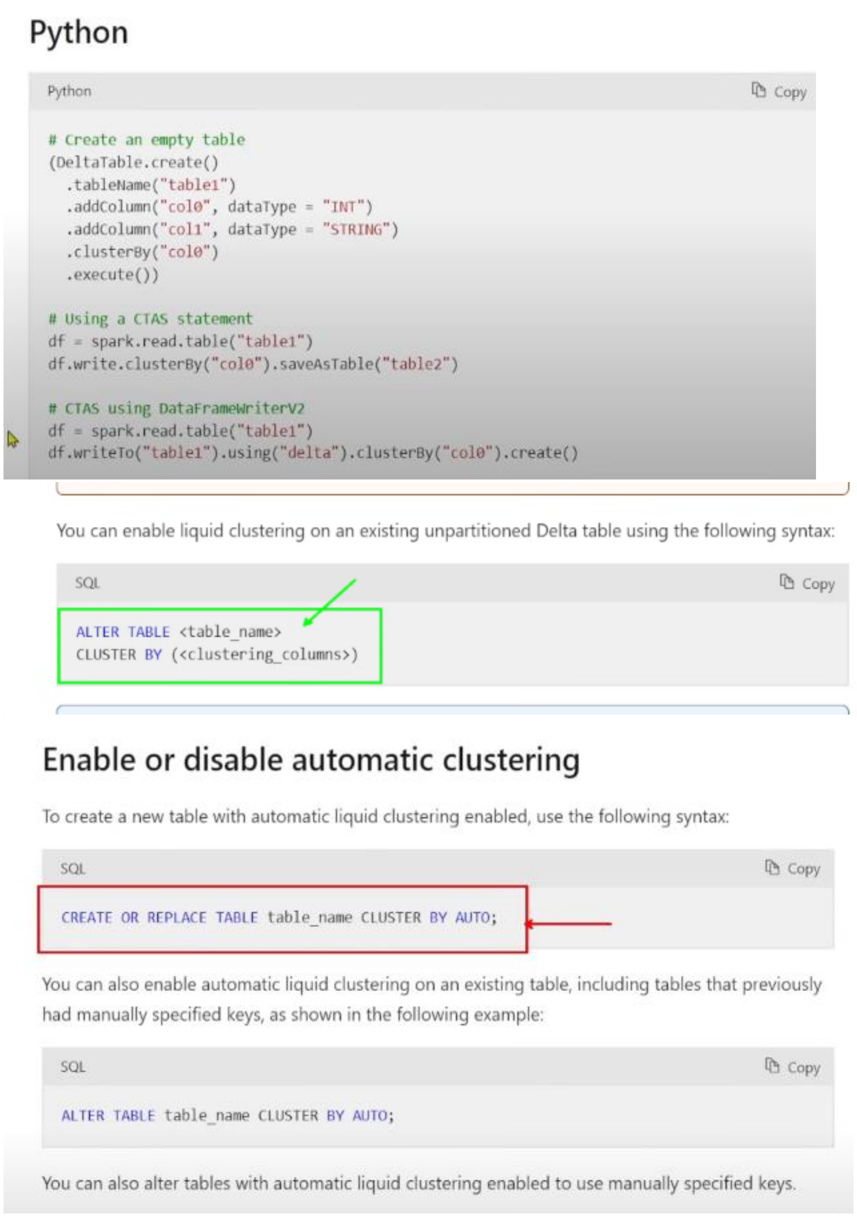
SQL

SQL Copy

```sql
-- Create an empty table
CREATE TABLE table1(col0 int, col1 string) CLUSTER BY (col0);

-- Using a CTAS statement
CREATE EXTERNAL TABLE table2 CLUSTER BY (col0)  -- specify clustering after table name, no
LOCATION 'table_location'
AS SELECT * FROM table1;

-- Using a LIKE statement to copy configurations
CREATE TABLE table3 LIKE table1;
```



Python

Python

```python
# Create an empty table
(DeltaTable.create()
  .tableName("table1")
  .addColumn("col0", dataType = "INT")
  .addColumn("col1", dataType = "STRING")
  .clusterBy("col0")
  .execute())

# Using a CTAS statement
df = spark.read.table("table1")
df.write.clusterBy("col0").saveAsTable("table2")

# CTAS using DataFrameWriterV2
df = spark.read.table("table1")
df.writeTo("table1").using("delta").clusterBy("col0").create()
```

You can enable liquid clustering on an existing unpartitioned Delta table using the following syntax:

SQL

```sql
ALTER TABLE <table_name>
CLUSTER BY (<clustering_columns>)
```

Enable or disable automatic clustering

To create a new table with automatic liquid clustering enabled, use the following syntax:

SQL

```sql
CREATE OR REPLACE TABLE table_name CLUSTER BY AUTO;
```

You can also enable automatic liquid clustering on an existing table, including tables that previously had manually specified keys, as shown in the following example:

SQL

```sql
ALTER TABLE table_name CLUSTER BY AUTO;
```

You can also alter tables with automatic liquid clustering enabled to use manually specified keys.