

Data bricks community edition

Loading a file to Data bricks

databricks

D

+ New

Workspace

Recents

Search

Catalog

Workflows

Compute

Machine Learning

Experiments

Add data >

Create New Table

Data source

Upload File S3

DBFS Target Directory ?
/FileStore/tables/ (optional) Select

Files uploaded to DBFS are accessible by everyone who has access to this workspace. [Learn more](#)

Files ?

Credit.csv ✓

0.3 KB
[Remove file](#)

✓ File uploaded to /FileStore/tables/Credit.csv

Create Table with UI Create Table in Notebook ?

▶

✓

17 hours ago (16s)

2

Python

🗑️

🔍

⋮

▼

```
# File location and type
file_location = "/FileStore/tables/Credit.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df = spark.read.format(file_type) \
    .option("inferSchema", infer_schema) \
    .option("header", first_row_is_header) \
    .option("sep", delimiter) \
    .load(file_location)

display(df)
```

▶ (3) Spark Jobs

▶ 📄 df: pyspark.sql.dataframe.DataFrame = [Credit_id: integer, Credit_Name: string ... 2 more fields]

▶

▼

✓

17 hours ago (16s)

2

Python

🗑️

🔍

⋮

▶ (3) Spark Jobs

▶ 📄 df: pyspark.sql.dataframe.DataFrame = [Credit_id: integer, Credit_Name: string ... 2 more fields]

Table ▼

+

🔍

🔍

📄

	123 Credit_id	A8C Credit_Name	A8C Credit_Type	123 Credit_Score
1	1	Deepthi	Master	989
2	2	Rahul	Visa	1124
3	3	Anjana	Visa	1124
4	4	Shriya	Master	765
5	1	Deepthi	Visa	678
6	2	Rahul	Master	1245
7	3	Anjana	Master	989
8	4	Shriya	Visa	765
9	1	Deepthi	Master	989
10	3	Anjana	Master	789

⌵ 10 rows | 16.44s runtime

Refreshed 17 hours ago

▶

✓

17 hours ago (<1s)

3

```
# Create a view or table

temp_table_name = "Credit_csv"

df.createOrReplaceTempView(temp_table_name)
```

▶

✓

17 hours ago (1s)

4

```
%sql

/* Query the created temp table in a SQL cell */

select * from `Credit_csv`
```

▶ (1) Spark Jobs

▶ 📄 _sql\$df: pyspark.sql.dataframe.DataFrame = [Credit_id: integer, Credit_Name: string ... 2 more fields]

▶ (1) Spark Jobs

▶ _sqldf: pyspark.sql.dataframe.DataFrame = [Credit_id: integer, Credit_Name: string ... 2 more fields]

Table

	Credit_id	Credit_Name	Credit_Type	Credit_Score
1	1	Deepthi	Master	989
2	2	Rahul	Visa	1124
3	3	Anjana	Visa	1124
4	4	Shriya	Master	765
5	1	Deepthi	Visa	678
6	2	Rahul	Master	1245
7	3	Anjana	Master	989
8	4	Shriya	Visa	765
9	1	Deepthi	Master	989
10	3	Anjana	Master	789

10 rows | 0.72s runtime Refreshed 17 hours ago

▶
▼
✓
17 hours ago (9s)
5
Python

```

# With this registered as a temp view, it will only be available to this particular notebook. If you'd like other
# users to be able to query this table, you can also create a table from the DataFrame.
# Once saved, this table will persist across cluster restarts as well as allow various users across different
# notebooks to query this data.
# To do so, choose your table name and uncomment the bottom line.

permanent_table_name = "Credit_csv"

df.write.format("parquet").saveAsTable(permanent_table_name)

```

▶ (1) Spark Jobs

1

▶ ▼ ✓ 17 hours ago (1s) 6 SQL

```
%sql
select * from default.credit_csv
```

▶ (1) Spark Jobs

▶ _sqldf: pyspark.sql.dataframe.DataFrame = [Credit_id: integer, Credit_Name: string ... 2 more fields]

Table

	Credit_id	Credit_Name	Credit_...	Credit_Score
1	1	Deepthi	Master	989
2	2	Rahul	Visa	1124
3	3	Anjana	Visa	1124
4	4	Shriya	Master	765
5	1	Deepthi	Visa	678
6	2	Rahul	Master	1245
7	3	Anjana	Master	989
8	4	Shriya	Visa	765
9	1	Deepthi	Master	989
10	3	Anjana	Master	789

10 rows | 1.42s runtime Refreshed 17 hours ago

Create your own data frames using sample data

17 hours ago (4s) 8

```
from pyspark.sql import Row
df1=[
    Row(Name="Deepthi",City="Toronto",Phno=2499791362),
    Row(Name="Rahul",City="Toronto",Phno=2499794567)
]
df2= spark.createDataFrame(df1)
display(df2)
```

▶ (3) Spark Jobs

▶ df2: pyspark.sql.dataframe.DataFrame = [Name: string, City: string ... 1 more field]

Table + 🔍 ⚙️ 📄

	^A _C Name	^A _C City	¹ ₃ Phno
1	Deepthi	Toronto	2499791362
2	Rahul	Toronto	2499794567

17 hours ago (16s) 9

```
df2.write.format("Delta").save("/FileStore/tables/Ncpl/employee")
```

▶ (6) Spark Jobs

17 hours ago (3s) 10 Python 🗑️ 📄 ⋮

```
dbf=spark.read.format("Delta").load("/FileStore/tables/Ncpl/employee")
display(dbf)
```

▶ (3) Spark Jobs

▶ dbf: pyspark.sql.dataframe.DataFrame = [Name: string, City: string ... 1 more field]

Table + 🔍 ⚙️ 📄

	^A _C Name	^A _C City	¹ ₃ Phno
1	Deepthi	Toronto	2499791362
2	Rahul	Toronto	2499794567

📄 2 rows | 2.92s runtime Refreshed 17 hours ago

Explore the delta table and version control mechanism

Added new data to data frame

Just now (9s) 8 Python

```

from pyspark.sql import Row
df1=[
    Row(Name="Deepthi",City="Toronto",Phno=2499791362),
    Row(Name="Rahul",City="Toronto",Phno=2499794567),
    Row(Name="Anjana",City="Sudbury",Phno=2496784567),
    Row(Name="Shirya",City="Brampton",Phno=4359794567)
]
df2= spark.createDataFrame(df1)
display(df2)

```

▶ (3) Spark Jobs

▶ df2: pyspark.sql.dataframe.DataFrame = [Name: string, City: string ... 1 more field]

Table + 🔍 🏠

	Name	City	Phno
1	Deepthi	Toronto	2499791362
2	Rahul	Toronto	2499794567
3	Anjana	Sudbury	2496784567
4	Shirya	Brampton	4359794567

Data is appended

Just now (32s) 9

```

df2.write.format("Delta").mode("append").save("/FileStore/tables/Ncpl/employee")

```

▶ (9) Spark Jobs

Just now (8s) 10 Python

```

dbf=spark.read.format("Delta").load("/FileStore/tables/Ncpl/employee")
display(dbf)

```

▶ (4) Spark Jobs

▶ dbf: pyspark.sql.dataframe.DataFrame = [Name: string, City: string ... 1 more field]

Table + 🔍 🏠

	Name	City	Phno
1	Deepthi	Toronto	2499791362
2	Deepthi	Toronto	2499791362
3	Shirya	Brampton	4359794567
4	Anjana	Sudbury	2496784567
5	Rahul	Toronto	2499794567
6	Rahul	Toronto	2499794567

Even after the data changed in data frame if we want first version of data we can get it by using versionAsOf

Just now (3s)

11

Python

```
dbf=spark.read.format("Delta").option("versionAsOf",0).load("/FileStore/tables/Ncpl/employee")
display(dbf)
```

(3) Spark Jobs

dbf: pyspark.sql.dataframe.DataFrame = [Name: string, City: string ... 1 more field]

Table

	Name	City	Phno
1	Deepthi	Toronto	2499791362
2	Rahul	Toronto	2499794567

2 rows | 2.61s runtime

Refreshed now

Just now (2s)

12

Python

```
dbf=spark.read.format("Delta").option("versionAsOf",1).load("/FileStore/tables/Ncpl/employee")
display(dbf)
```

(3) Spark Jobs

dbf: pyspark.sql.dataframe.DataFrame = [Name: string, City: string ... 1 more field]

Table

	Name	City	Phno
1	Deepthi	Toronto	2499791362
2	Deepthi	Toronto	2499791362
3	Shirya	Brampton	4359794567
4	Anjana	Sudbury	2496784567
5	Rahul	Toronto	2499794567
6	Rahul	Toronto	2499794567

6 rows | 2.22s runtime

Refreshed now

Adding new data and overwriting it

Just now (1s)

8

Python

```
from pyspark.sql import Row
df1=[
    Row(Name="Anjana",City="Sudbury",Phno=2496784567),
    Row(Name="Shirya",City="Brampton",Phno=4359794567)
]
df2= spark.createDataFrame(df1)
display(df2)
```

(3) Spark Jobs

df2: pyspark.sql.dataframe.DataFrame = [Name: string, City: string ... 1 more field]

Table

	Name	City	Phno
1	Anjana	Sudbury	2496784567
2	Shirya	Brampton	4359794567

```
df2.write.format("Delta").mode("overwrite").save("/FileStore/tables/Ncpl/employee")
```

▶ (6) Spark Jobs

```
dbf=spark.read.format("Delta").load("/FileStore/tables/Ncpl/employee")
display(dbf)
```

▶ (3) Spark Jobs

▶ dbf: pyspark.sql.dataframe.DataFrame = [Name: string, City: string ... 1 more field]

Table ▾ +

	^A _C Name	^A _C City	¹ ₃ Phno
1	Shirya	Brampton	4359794567
2	Anjana	Sudbury	2496784567

```
dbf=spark.read.format("Delta").option("versionAsOf",2).load("/FileStore/tables/Ncpl/employee")
display(dbf)
```

▶ (2) Spark Jobs

▶ dbf: pyspark.sql.dataframe.DataFrame = [Name: string, City: string ... 1 more field]

Table ▾ +

	^A _C Name	^A _C City	¹ ₃ Phno
1	Shirya	Brampton	4359794567
2	Anjana	Sudbury	2496784567

Even though we overwrite the data we can get previous version of data

```
dbf=spark.read.format("Delta").option("versionAsOf",1).load("/FileStore/tables/Ncpl/employee")
display(dbf)
```

▶ (4) Spark Jobs

▶ dbf: pyspark.sql.dataframe.DataFrame = [Name: string, City: string ... 1 more field]

Table ▾ +

	^A _C N...	^A _C City	¹ ₃ Phno
1	Deepthi	Toronto	2499791362
2	Deepthi	Toronto	2499791362
3	Shirya	Brampton	4359794567
4	Anjana	Sudbury	2496784567
5	Rahul	Toronto	2499794567
6	Rahul	Toronto	2499794567

↓ 6 rows | 2.99s runtime Refreshed now

