

## PySpark Basics Result

ผลลัพธ์จากการ run ในไฟล์ PySpark Basics จะรวมไว้ในนี้ สามารถคลิกดูตามหัวข้อได้ (ในแอป docs คลิก 3 จุดขวาบน > Document Outline) บางหัวข้อที่ไม่ได้มีผลลัพธ์จะไม่ได้ปรากฏในนี้

### Create a DataFrame

Create a DataFrame with specified values

### Create a DataFrame with specified values

▶ ✅ 2 minutes ago (12s) 10

```
# แบบตั้งต้น พื้นฐาน
df_children = spark.createDataFrame(
    data = [("Mikhail", 15), ("Zaky", 13), ("Zoya", 8)],
    schema = ['name', 'age'])
display(df_children)
> See performance (1)
> df_children: pyspark.sql.connect.dataframe.DataFrame = [name: string, age: long]
```

Table +

	name	age
1	Mikhail	15
2	Zaky	13
3	Zoya	8

## Create a DataFrame from a table in Unity Catalog

Create a DataFrame from a table in Unity Catalog

```
# 从 Unity Catalog 中读取表
df_customer = spark.table('samples.tpch.customer')
display(df_customer)
> See performance (1)
```

df\_customer: pyspark.sql.connect.dataframe.DataFrame = [c\_custkey: long, c\_name: string ... 6 more fields]

	c_custkey	c_name	c_address	c_nationkey	c_phone	c_acctbal	c_mktseg
1	412445	Customer#0004124...	0QA83QjmbP6mA0Bkgf	21	31-421-403-4333	5358.33	BUILDING
2	412446	Customer#0004124...	5u8MSbyiC7J,7PuY4lvaq1JRbTCMKeNVaq	20	30-487-949-7942	9441.59	MACHINERY
3	412447	Customer#0004124...	HC4ZT62gKPgjr ceoaZgFOuniUogr7GO	7	17-797-466-6308	7868.75	AUTOMOBILE
4	412448	Customer#0004124...	hJok1MMrDgH	6	16-541-510-4964	6060.98	MACHINERY
5	412449	Customer#0004124...	zAt1nZNG01gOhIqgyDtDaS,Y0VSofZJsid	14	24-710-983-5536	4973.84	HOUSEHOLD
6	412450	Customer#0004124...	fUD6loGdtF	20	30-293-696-5047	4406.28	BUILDING
7	412451	Customer#0004124...	W2Ge0Qd8adH	20	30-590-724-6711	2290.38	BUILDING
8	412452	Customer#0004124...	lj4xiPleNEP1uR5p7H	10	20-492-590-3363	3426.64	AUTOMOBILE

## Create a DataFrame from an uploaded file

```
# 将此变量设置为您的完整卷文件路径
volume_file_path = "/Volumes/workspace/default/tutorial/rows.csv"

df_csv = (spark.read
    .format("csv")
    .option("header", True)
    .option("inferSchema", True)
    .load(volume_file_path)
)
display(df_csv)
> See performance (1)
```

df\_csv: pyspark.sql.connect.dataframe.DataFrame = [Year: integer, First Name: string ... 3 more fields]

	Year	First Name	County	Sex	Count
1	2022	OLIVIA	Albany	F	16
2	2022	AMELIA	Albany	F	15
3	2022	AVERY	Albany	F	12
4	2022	EMMA	Albany	F	11
5	2022	CHARLOTTE	Albany	F	11
6	2022	CHLOE	Albany	F	11
7	2022	SOPHIA	Albany	F	8

# Create a DataFrame from a JSON response

The screenshot shows a Jupyter Notebook cell with the following code:

```
# focus แค่ส่วนของ products นะ
products = response.json()["products"]

# ในข้อมูล จะมีส่วนที่ Spark ไม่สามารถจัดได้ร่างเรียกว่าใน Data Type แบบไหน เพราะบางที่ในส่วนนั้นข้อมูลยาวบ้าง สั้นบ้าง
# แล้วจึงการขอ默ดังนี้ก่อน
for p in products:
    p["price"] = float(p["price"])
    p["rating"] = float(p["rating"])
    p["discountPercentage"] = float(p["discountPercentage"])

# สร้าง DataFrame จาก JSON ดังกล่าว
df_json = spark.createDataFrame(products)
display(df_json)
```

Below the code, there is a table preview:

	availabilityStatus	brand	category	description
1	In Stock	Essence	beauty	> The Essence Mascara Lash Princess is a popular mascara known for its volumizing and lengthening effects. Achieve... > The Eyeshadow Palette with Mirror offers a versatile range of eyeshadow shades for creating stunning eye looks. > The Powder Canister is a finely milled setting powder designed to set makeup and control shine. With a lightwe...
2	In Stock	Glamour Beauty	beauty	> The Red Lipstick is a classic and bold choice for adding a pop of color to your lips. With a creamy and pigment...
3	In Stock	Velvet Touch	beauty	> The Powder Canister is a finely milled setting powder designed to set makeup and control shine. With a lightwe...
4	In Stock	Chic Cosmetics	beauty	> The Red Lipstick is a classic and bold choice for adding a pop of color to your lips. With a creamy and pigment...

(30 rows)

## Select a JSON field or object

The screenshot shows a Jupyter Notebook cell with the following code:

```
# เลือก field มาดู เช่น เรากด reviews ของสินค้า
display(df_json.select(df_json["reviews"]))
```

Below the code, there is a tree view of the selected JSON data:

- reviews
  - array
    - 0:
      - reviewerEmail: "eleanor.collins@x.dummyjson.com"
      - reviewerName: "Eleanor Collins"
      - comment: "Would not recommend!"
      - date: "2025-04-30T09:41:02.053Z"
      - rating: "3"
    - 1: {"reviewerEmail": "lucas.gordon@x.dummyjson.com", "reviewerName": "Lucas Gordon", "comment": "Very satisfied!", "date": "2025-04-30T09:41:02.053Z", "rating": "4"}
    - 2: {"reviewerEmail": "eleanor.collins@x.dummyjson.com", "reviewerName": "Eleanor Collins", "comment": "Highly impressed!", "date": "2025-04-30T09:41:02.053Z", "rating": "5"}

2 > [{"reviewerEmail": "savannah.gomez@x.dummyjson.com", "reviewerName": "Savannah Gomez", "comment": "Great product!", "date": "2025-04-30T09:41:02.053Z", "rating": "5"}]

3 > [{"reviewerEmail": "alexander.jones@x.dummyjson.com", "reviewerName": "Alexander Jones", "comment": "Would buy again!", "date": "2025-04-30T09:41:02.053Z", "rating": "5"}]

4 > [{"reviewerEmail": "liam.garcia@x.dummyjson.com", "reviewerName": "Liam Garcia", "comment": "Great product!", "date": "2025-04-30T09:41:02.053Z", "rating": "5"}]

5 > [{"reviewerEmail": "benjamin.wilson@x.dummyjson.com", "reviewerName": "Benjamin Wilson", "comment": "Poor quality!", "date": "2025-04-30T09:41:02.053Z", "rating": "1"}]

6 > [{"reviewerEmail": "layla.young@x.dummyjson.com", "reviewerName": "Layla Young", "comment": "Very disappointed!", "date": "2025-04-30T09:41:02.053Z", "rating": "1"}]

▶ ✓ 1 minute ago (1s) 23

```
# เลือก reviewerName ที่อยู่ใน index ที่ 1 ของรีวิวทั้งหมด ในสินค้าแต่ละชิ้น
display(df_json.select(
    df_json["reviews"][1]["reviewerName"],
    df_json["reviews"][1]["rating"],
    df_json["reviews"][1]["comment"]
))
```

> See performance (1)

Table +

	<code>reviews[1][reviewerName]</code>	<code>reviews[1][rating]</code>	<code>reviews[1][comment]</code>
1	Lucas Gordon	4	Very satisfied!
2	Christian Perez	4	Awesome product!
3	Elijah Cruz	5	Highly impressed!
4	Ruby Andrews	5	Great product!
5	Liam Smith	5	Great product!
6	Daniel Cook	4	Fast shipping!
7	Leah Henderson	5	Awesome product!
8	Penelope Harper	4	Great value for money!
9	Nolan Gonzalez	4	Highly recommended!
10	Daniel Cook	5	Very happy with my purchase!

## Create a DataFrame from a file

Create a DataFrame from a file 25 Python

▶ ✓ 2 minutes ago (1s)

```
# ดูว่าใน Databricks ไฟล์ Sample data จะมีไฟล์อะไร
display(dbutils.fs.ls('/databricks-datasets'))
```

> See performance (2)

Table +

	<code>path</code>	<code>name</code>	<code>size</code>	<code>modificationTime</code>
1	dbfs:/databricks-datasets/COVID/	COVID/	0	1765694991640
2	dbfs:/databricks-datasets/README.md	README.md	976	1596557781000
3	dbfs:/databricks-datasets/Rdatasets/	Rdatasets/	0	1765694991640
4	dbfs:/databricks-datasets/SPARK_README.md	SPARK_README.md	3359	1596557823000
5	dbfs:/databricks-datasets/adult/	adult/	0	1765694991640
6	dbfs:/databricks-datasets/airlines/	airlines/	0	1765694991641
7	dbfs:/databricks-datasets/amazon/	amazon/	0	1765694991641

หรือจะใช้วิธีอีกแบบคือ [Databricks CLI file system commands](#)

```
▶ ✓ 5 minutes ago (1s) 28
%fs ls '/databricks-datasets'
> See performance (2)

Table +
```

	A <sup>B</sup> C path	A <sup>B</sup> C name	A <sup>2</sup> <sub>3</sub> size	A <sup>2</sup> <sub>3</sub> modificationTime
1	dbfs:/databricks-datasets/COVID/	COVID/	0	1765694992444
2	dbfs:/databricks-datasets/README.md	README.md	976	1596557781000
3	dbfs:/databricks-datasets/Rdatasets/	Rdatasets/	0	1765694992444
4	dbfs:/databricks-datasets/SPARK_README.md	SPARK_README.md	3359	1596557823000
5	dbfs:/databricks-datasets/adult/	adult/	0	1765694992444

(ผลลัพธ์ใหม่อ่อนกัน)

```
▶ ✓ 01:49 PM (2s) 29
df_population = (spark.read
  .format("csv")
  .option("header", True)
  .option("inferSchema", True)
  .load("/databricks-datasets/samples/population-vs-price/data_geo.csv")
)
display(df_population)
> See performance (1)
> df_population: pyspark.sql.connect.DataFrame = [2014 rank: integer, City: string ... 4 more fields]

Table +
```

	A <sup>2</sup> <sub>3</sub> 2014 rank	A <sup>B</sup> C City	A <sup>B</sup> C State	A <sup>B</sup> C State Code	A <sup>2</sup> <sub>3</sub> 2014 Population estimate	1.2 2015 median sales price
1	101	Birmingham	Alabama	AL	212247	162.9
2	125	Huntsville	Alabama	AL	188226	157.7
3	122	Mobile	Alabama	AL	194675	122.5
4	114	Montgomery	Alabama	AL	200481	129
5	64	Anchorage[19]	Alaska	AK	301010	null
6	78	Chandler	Arizona	AZ	254276	null
7	86	Gilbert[20]	Arizona	AZ	239277	null
8	88	Glendale	Arizona	AZ	237517	null
9	38	Mesa	Arizona	AZ	464704	null
10	148	Peoria	Arizona	AZ	166934	null

# Transform data with DataFrames

## Column operations



A screenshot of a Jupyter Notebook cell. The cell has a blue play button icon and a green checkmark icon. The text "Just now (<1s)" is displayed next to the checkmark. The code in the cell is:

```
# all columns in DF
df_customer.columns
```

The output of the cell is:

```
[c_custkey',
 'c_name',
 'c_address',
 'c_nationkey',
 'c_phone',
 'c_acctbal',
 'c_mktsegment',
 'c_comment']
```

## Select columns

```
▶   ✓  2 minutes ago (<1s)

# ใช้ select และ col
from pyspark.sql.functions import col

df_customer.select(
    col("c_custkey"),
    col("c_acctbal")
)

DataFrame[c_custkey: bigint, c_acctbal: decimal(18,2)]
```

```
▶   ✓  01:49 PM (1s)

# ใช้ expr ที่สามารถรับ expression ในรูปแบบ string ได้
from pyspark.sql.functions import expr

df_customer.select(
    expr("c_custkey"),
    expr("c_acctbal")
)

DataFrame[c_custkey: bigint, c_acctbal: decimal(18,2)]
```

```
▶   ✓  01:49 PM (<1s)

# selectExpr ที่สามารถรับ SQL expressions ได้
df_customer.selectExpr(
    "c_custkey as key",
    "round(c_acctbal) as account_rounded"
)

DataFrame[key: bigint, account_rounded: decimal(17,0)]
```

```
▶   ✓  01:49 PM (<1s)

# select แบบใช้ string
df_customer.select(
    "c_custkey",
    "c_acctbal"
)

DataFrame[c_custkey: bigint, c_acctbal: decimal(18,2)]
```

ในการเลือกคอลัมน์จาก DataFrame ที่เราต้องการโดยเฉพาะ สามารถใช้ `[]` หรือ `[ ]` ได้ (`[ ]` ในสามารถใช้เลือกคอลัมน์ที่ชื่อต้นด้วยจำนวนเต็ม หรือคอลัมน์ที่มีช่องว่างหรืออักษรพิเศษได้) หรือซึ่งวายได้เมื่อ join dataframe ที่มีคอลัมน์เหมือนกัน

```
43
df_customer.select(
    df_customer["c_custkey"],
    df_customer["c_acctbal"]
)

DataFrame[c_custkey: bigint, c_acctbal: decimal(18,2)]
```

```
44
df_customer.select(
    df_customer.c_custkey,
    df_customer.c_acctbal
)

DataFrame[c_custkey: bigint, c_acctbal: decimal(18,2)]
```

## Create columns

```
48
df_customer_flag = df_customer.withColumn("balance_flag", col("c_acctbal") > 1000)
> df_customer_flag: pyspark.sql.connect.DataFrame = [c_custkey: long, c_name: string ... 7 more fields]
```

```
49
display(df_customer_flag)
> See performance (1)
```

Optimize

	c_acctbal	c_mktsegment	c_comment	balance_flag
1	5358.33	BUILDING	arefully blithely regular epi	true
2	9441.59	MACHINERY	sleep according to the fluffy even forges. fluffy careful packages after the ironic, silent deposi	true
3	7868.75	AUTOMOBILE	aggle blithely among the carefully express excus	true
4	6060.98	MACHINERY	ly silent requests boost slyly, express courts sleep according to the fluf	true
5	4973.84	HOUSEHOLD	refully final theodolites. final, slow excuses sleep quickly! quickly ironic idea	true
6	4406.28	BUILDING	refully final dolphins after the carefully bold packages sleep quickly express deposits. fluffy	true
7	2290.38	BUILDING	slow asymptotes will are carefully final packages. slyly regular fox	true
8	3426.64	AUTOMOBILE	deep chly after the sometimes even ideas. chly express theodolites dazzle furiously ironic dependenci	true

## Rename columns

```
▶  ✓ 01:49 PM (<1s) 51
df_customer_flag_renamed = df_customer_flag.withColumnRenamed("balance_flag", "balance_flag_renamed")
> df_customer_flag_renamed: pyspark.sql.connect.DataFrame = [c_custkey: long, c_name: string ... 7 more fields]

▶  ✓ 01:49 PM (1s) 52
display(df_customer_flag_renamed)
> See performance (1) Python ⚡ ⋮ Optimize

Table + 
+---+---+
| il | ABc c_mktsegment | ABc c_comment | ✓ balance_flag_renamed |
+---+---+
| 1 | 3358.33 BUILDING | arefully blithely regular epi | true |
| 2 | 3441.59 MACHINERY | sleep according to the fluffy even forges. fluffy careful packages after the ironic, silent deposi | true |
| 3 | 7868.75 AUTOMOBILE | aggle blithely among the carefully express excus | true |
| 4 | 5060.98 MACHINERY | ly silent requests boost stly. express courts sleep according to the fluf | true |
| 5 | 1973.84 HOUSEHOLD | refully final theodolites. final, slow excuses sleep quickly! quickly ironic idea | true |
| 6 | 1406.28 BUILDING | refully final dolphins after the carefully bold packages sleep quickly express deposits. fluffy | true |
+---+---+---+---+
```

```
▶  ✓ 01:49 PM (1s) 55
from pyspark.sql.functions import avg

df_segment_balance = df_customer.groupby("c_mktsegment").agg(
    avg(df_customer["c_acctbal"]).alias("avg_account_balance")
)

display(df_segment_balance)
> See performance (1)
> df_segment_balance: pyspark.sql.connect.DataFrame = [c_mktsegment: string, avg_account_balance: decimal(22,6)]
```

```
Table + 
+---+---+
| ABc c_mktsegment | .00 avg_account_balance |
+---+---+
| 1 | BUILDING | 4509.073485 |
| 2 | MACHINERY | 4494.998388 |
| 3 | AUTOMOBILE | 4495.681835 |
| 4 | HOUSEHOLD | 4508.414341 |
| 5 | FURNITURE | 4496.971036 |
+---+---+
```

↓ 5 rows | 1.22s runtime

^ ภาพบนนี้คือผลจากการเปลี่ยนชื่อของคอลัมน์ที่มาจากการทำ aggregate

## Cast column types

```
▶  ✓  01:49 PM (<1s) 58
from pyspark.sql.functions import col

df_casted = df_customer.withColumn("c_custkey", col("c_custkey").cast(StringType()))
print(type(df_casted))

▼ df_casted: pyspark.sql.connect.DataFrame
  c_custkey: string
  c_name: string
  c_address: string
  c_nationkey: long
  c_phone: string
  c_acctbal: decimal(18,2)
  c_mktsegment: string
  c_comment: string

<class 'pyspark.sql.connect.DataFrame'>
```

## Remove columns

```
▶  ✓  4 minutes ago (<1s) 61
df_customer_flag_renamed.drop("balance_flag_renamed")

DataFrame[c_custkey: bigint, c_name: string, c_address: string, c_nationkey: bigint, c_phone: string, c_acctbal: decimal(18,2), c_mktsegment: string,
c_comment: string]

+ Code + Text Assistant
```

```
▶  ✓  4 minutes ago (<1s) 62
df_customer_flag_renamed.drop("c_phone", "balance_flag_renamed")

DataFrame[c_custkey: bigint, c_name: string, c_address: string, c_nationkey: bigint, c_acctbal: decimal(18,2), c_mktsegment: string, c_comment: string]
```

```
▶  ✓  02:35 PM (1s) 68
display(df_customer.filter((col("c_nationkey") == 20) & (col("c_acctbal") > 1000)))
> See performance (1) Optimize
```

	c_nationkey	c_phone	c_acctbal	c_mktsegment	c_comment	
4	tPlylFTZe4Wzc2RkHFMQ	20	30-849-795-3196	4977.67	HOUSEHOLD	the regular accounts. furiously express frays boost car
5	y8iR	20	30-371-666-2149	2761.61	AUTOMOBILE	ithely? slyly final packages
6	lWyVvR	20	30-134-806-9402	6710.60	BUILDING	sits cajole quickly among the furiously final warthogs-
7	zfe76032Ei55	20	30-608-742-5671	3117.69	BUILDING	ly ironic theodolites. slyly bold accounts sleep daringly
8	6XpQC6xSVY4imtMpm	20	30-825-964-4062	5747.10	FURNITURE	foxes. regular, unusual accou
9	NUPhBqoOrV	20	30-166-422-9502	9705.61	BUILDING	along the ironic, final pinto beans. bold deposits are
10	/3detjlY7YH7mFRntVmFkjMhCuR5	20	30-653-321-3760	9536.17	FURNITURE	y special packages. slyly thin ideas wake carefully.
11	Tbeyonss11H0Tqs0CzMiCADD	20	30-833-575-4136	7200.15	FURNITURE	final requests cajole careful
12	iSAWHGVEBhF1IUas7or	20	30-426-585-9074	6342.12	HOUSEHOLD	. ironic packages cajole above the regular requests. car
13	HnaRynn9TWCGpmht4N1bS	20	30-484-261-6925	6998.56	FURNITURE	sits are along the carefully even accounts. fluffily final :
14	OKhD95BUffzi	20	30-493-308-1226	3226.24	HOUSEHOLD	nts. furiously even dugouts are. ironi

02:35 PM (<1s) 69 Python

```
# ໃຊ້ c_custkey ແລະ or operator ວິທີກຳເນົດເລືອກນາມ
df_filtered_customer = df_customer.filter((col("c_custkey") == 412446) | (col("c_custkey") == 412447))

> df_filtered_customer: pyspark.sql.connect.DataFrame = [c_custkey: long, c_name: string ... 6 more fields]
```

02:35 PM (1s) 70 Python

```
display(df_filtered_customer)
> See performance (1)
```

Table +

	c_custkey	c_name	c_address	c_nationkey	c_phone	c_acctbal	c_mktsegment
1	412446	Customer#0004124...	5u8MSbyiC7J,7PuY4lvaq1RbTCMKeNVqg	20	30-487-949-7942	9441.59	MACHINERY
2	412447	Customer#0004124...	HC4ZT62gKPgrjr ceoaZgFOunlUogr7GO	7	17-797-466-6308	7868.75	AUTOMOBILE

2 rows | 0.92s runtime Refreshed 15 minutes ago

## Append rows

02:35 PM (1s) 87 Python

```
df_appended_rows = df_that_one_customer.union(df_filtered_customer)

display(df_appended_rows)
> See performance (1)
```

df\_appended\_rows: pyspark.sql.connect.DataFrame = [c\_custkey: long, c\_name: string ... 6 more fields]

Table +

	c_custkey	c_name	c_address	c_nationkey	c_phone	c_acctbal	c_mktsegment
1	412449	Customer#0004124...	zAt1nZNG01gOhlggyDtDa S,Y0VSofZjs1...	14	24-710-983-5536	4973.84	HOUSEHOLD
2	412446	Customer#0004124...	5u8MSbyiC7J,7PuY4lvaq1RbTCMKeNVqg	20	30-487-949-7942	9441.59	MACHINERY
3	412447	Customer#0004124...	HC4ZT62gKPgrjr ceoaZgFOunlUogr7GO	7	17-797-466-6308	7868.75	AUTOMOBILE

3 rows | 1.00s runtime Refreshed 34 minutes ago

## Sort rows

89

```
▶ ✓ 02:35 PM (1s)
display(df_customer.orderBy(col("c_acctbal")))
> See performance (1)
```

Table +

	<sup>B</sup> c_name	<sup>B</sup> c_address	<sup>I<sub>3</sub></sup> c_nationkey	<sup>B</sup> c_phone	.00 c_acctbal	<sup>B</sup> c_mktsegment	<sup>B</sup> c_comment
1	Customer#00001488...	oJKqRZcWFE_9x7e2Z8kgdQq,h	6	16-860-910-6299	-999.99	FURNITURE	instructions caj...
2	Customer#00000540...	y6k6qXqWugY1z6AWSmipy	21	31-173-610-7413	-999.98	AUTOMOBILE	ckages. special...
3	Customer#00000070...	56UWrTFeF1evNB4m4BFud0,gyGjgVQldw9hB	4	14-356-473-2563	-999.95	MACHINERY	carefully unusu...
4	Customer#0002638...	7kBBykFnOxf	6	16-609-430-5195	-999.95	FURNITURE	excuses wake i...
5	Customer#0001688...	agf1v8GocZbZ zBx62292gUQPjBwdQGZ	1	11-851-335-7609	-999.95	BUILDING	fluffly special...
6	Customer#0003232...	WWKKNyZwaq9qEp,YHd9DfmnlbS	13	23-152-829-4985	-999.94	BUILDING	ts. even, pendin...
7	Customer#0001235...	mVNNNuSnobb	21	31-908-400-6759	-999.93	MACHINERY	nt deposits. req...
8	Customer#0006222...	agZp7,Y2GQdmzoWtBrFH0yeRVsz	23	33-545-236-9930	-999.92	MACHINERY	. blithely final...
9	Customer#0005701...	CKyqlw72qagOWxFwdvA199jCIDE	7	17-951-808-5613	-999.90	HOUSEHOLD	requests wake...
10	Customer#0007248...	MQRKLQ,RjIQVRfnmElvGGtqgg	13	23-455-962-4517	-999.90	FURNITURE	the furiously e...
11	Customer#0006128...	wO8pWban60PhOG 2JaC09y0LUVB5s45NK4Tzd	14	24-452-144-3866	-999.89	MACHINERY	eodolites eat fu...
12	Customer#0003875...	,N,1rzN oExmlAPRbTbTAZGfSihXATNztROgw	20	30-523-653-9696	-999.89	FURNITURE	ways even requ...
13	Customer#0004811...	VbSLw8lkCToFU	11	21-859-110-1522	-999.88	MACHINERY	slow theodolit...

90

```
▶ ✓ 02:35 PM (2s)
# ใช้ desc() เพื่อเรียงจากมากไปน้อย
display(df_customer.sort(col("c_custkey").desc()))
> See performance (1)
```

Table +

	<sup>I<sub>3</sub></sup> c_custkey	<sup>B</sup> c_name	<sup>B</sup> c_address	<sup>I<sub>3</sub></sup> c_nationkey	<sup>B</sup> c_phone	.00 c_acctbal	<sup>B</sup> c_mktsegment
1	750000	Customer#0007500...	WDCQSV9ExHw	14	24-810-310-6879	1289.08	AUTOMOBILE
2	749999	Customer#0007499...	F189mrzCqRwl	22	32-677-912-6847	631.78	BUILDING
3	749998	Customer#0007499...	b1Z,6uSx8TNeuTDHMO4	23	33-136-966-8926	3059.89	FURNITURE
4	749997	Customer#0007499...	FXaqJdiXlqy7fCR52OAAaNbFXIluxEli	5	15-846-165-9594	8733.15	MACHINERY
5	749996	Customer#0007499...	B489KiMGBSSNmLMjHSNhMinPTRK3J	4	14-362-620-3071	-505.93	AUTOMOBILE
6	749995	Customer#0007499...	Ys8exKLDd6N8poASbgMDSDphulZSzG fe0q	13	23-991-740-6907	6656.90	FURNITURE
7	749994	Customer#0007499...	XoqUeQf917pNpzYeZsZ4G	5	15-792-449-8157	-598.34	AUTOMOBILE
8	749993	Customer#0007499...	QMYqPzYClPqiMxoCxm4qis3UgRAUBrd	6	16-548-178-4993	4668.55	MACHINERY
9	749992	Customer#0007499...	fw3OEji3tutOR9	15	25-965-631-6868	6166.34	FURNITURE
10	749991	Customer#0007499...	1SR18vMAImvZyqY7Zx	23	33-418-559-5843	5782.90	AUTOMOBILE
11	749990	Customer#0007499...	YRYfpvBYKrgEQRhomA 8wYncFCWnD8m	2	12-242-933-4443	7927.63	BUILDING
12	749989	Customer#0007499...	SKKKoP,wk1OHhAN	4	14-925-634-8481	4697.86	AUTOMOBILE
13	749988	Customer#0007499...	Bdg5az7w E17rlZPchKuzE,dpMtU	10	20-681-689-3035	304.09	FURNITURE
14	749987	Customer#0007499...	zh6wfyPoZSYT8DwPqzRcm k	15	25-908-559-5601	144.78	FURNITURE

92

```
▶ ✓ 02:35 PM (<1s)
# เรียงมากกว่า 1 คอลัมน์
df_sorted = df_customer.orderBy(col("c_acctbal").desc(), col("c_custkey").asc())
df_sorted = df_customer.sort(col("c_acctbal").desc(), col("c_custkey").asc())
> df_sorted: pyspark.sql.connect.DataFrame = [c_custkey: long, c_name: string ... 6 more fields]
```

93

```
▶ ✓ 02:35 PM (1s)
display(df_sorted.limit(10))
> See performance (1)
```

Table +

	<sup>I<sub>3</sub></sup> c_custkey	<sup>B</sup> c_name	<sup>B</sup> c_address	<sup>I<sub>3</sub></sup> c_nationkey	<sup>B</sup> c_phone	.00 c_acctbal	<sup>B</sup> c_mktsegment
1	61453	Customer#0000614...	RxNgWcy15RZD4qOynyT3	15	25-819-925-1077	9999.99	BUILDING
2	508503	Customer#0005085...	EBhwnV5Ujs0VX	24	34-668-980-4098	9999.99	HOUSEHOLD
3	399453	Customer#0003994...	6jbggglUx8E	23	33-793-351-2867	9999.97	FURNITURE
4	69321	Customer#0000693...	ZKuxYWP7wMeeHT j5dAZ	15	25-347-101-1161	9999.96	AUTOMOBILE
5	242308	Customer#0002423...	U3vunijnOJ669ENHVrgiCb	22	32-977-688-8103	9999.96	MACHINERY
6	348586	Customer#0003485...	hQOvYfZD H,EH3aKL34h3zKS0kCqClyh9ZLBlo	15	25-358-183-6366	9999.96	HOUSEHOLD
7	652672	Customer#0006526...	Bn8ltMvKcxrdN0rxH1192lmi AAC	3	13-724-936-6840	9999.93	MACHINERY

# Join DataFrames

▶ 02:35 PM (4s) 96 Python ⚡ ⌂ ⌂

```
df_customer = spark.table('samples.tpch.customer')
df_order = spark.table('samples.tpch.orders')

df_joined = df_order.join(
    df_customer,
    on = df_order["o_custkey"] == df_customer["c_custkey"],
    how = "inner"
)

display(df_joined)
> See performance (1) Optimize
```

> df\_customer: pyspark.sql.connect.dataframe.DataFrame = [c\_custkey: long, c\_name: string ... 6 more fields]  
> df\_joined: pyspark.sql.connect.dataframe.DataFrame = [o\_orderkey: long, o\_custkey: long ... 15 more fields]  
> df\_order: pyspark.sql.connect.dataframe.DataFrame = [o\_orderkey: long, o\_custkey: long ... 7 more fields]

	Table	+	🔍	✖	☰	☰		
1	$\text{o\_orderkey}$	$\text{o\_custkey}$	$\text{o\_orderstatus}$	.00 $\text{o\_totalprice}$	$\text{o\_orderdate}$	$\text{o\_orderpriority}$	$\text{o\_clerk}$	$\text{o\_shippi}$
1	11419110	715561	F	104040.78	1992-03-09	1-URGENT	Clerk#000001175	
2	11421925	361792	F	211375.04	1994-10-31	4-NOT SPECIFIED	Clerk#000004690	
3	11426566	453949	F	141293.47	1993-12-10	2-HIGH	Clerk#000001528	
4	11429414	608180	F	55602.80	1992-01-23	1-URGENT	Clerk#000004083	
5	11430853	266785	O	214631.98	1996-10-23	5-LOW	Clerk#000002861	
6	11430882	660493	F	139133.98	1994-01-29	5-LOW	Clerk#000000171	

6555+ rows

## Aggregate data

▶ ▾ ✓ 02:35 PM (1s) 101

```
from pyspark.sql.functions import avg

# group by one column
df_segment_balance = df_customer.groupBy("c_mktsegment").agg(
    avg(df_customer["c_acctbal"])
)

display(df_segment_balance)
> See performance (1)
```

df\_segment\_balance: pyspark.sql.connect.DataFrame = [c\_mktsegment: string, avg(c\_acctbal): decimal(22,6)]

Table +

c_mktsegment	.00 avg(c_acctbal)
BUILDING	4509.073485
MACHINERY	4494.998388
AUTOMOBILE	4495.681835
HOUSEHOLD	4508.414341
FURNITURE	4496.971036

↓ ▾ 5 rows | 0.99s runtime

▶ ✓ 02:35 PM (1s) 102

```
# แยกตาม nation ด้วย
from pyspark.sql.functions import avg

# group by two columns
df_segment_nation_balance = df_customer.groupBy("c_mktsegment", "c_nationkey").agg(
    avg(df_customer["c_acctbal"])
)

display(df_segment_nation_balance)
> See performance (1)
```

df\_segment\_nation\_balance: pyspark.sql.connect.DataFrame = [c\_mktsegment: string, c\_nationkey: long ... 1 more field]

Table +

c_mktsegment	c_nationkey	.00 avg(c_acctbal)
BUILDING	21	4537.355840
MACHINERY	20	4506.505043
AUTOMOBILE	7	4457.661147
MACHINERY	6	4413.460338
HOUSEHOLD	14	4515.775535
BUILDING	20	4537.627599
AUTOMOBILE	10	4494.695059
MACHINERY	21	4500.694374

## Chaining calls

The screenshot shows a Jupyter Notebook cell with the following content:

```
from pyspark.sql.functions import count

df_chained = (
    df_order.filter(col("o_orderstatus") == "F") # 1. Filter
    .groupBy(col("o_orderpriority")) # 2. Group By
    .agg(count(col("o_orderkey")).alias("n_orders")) # 3. Aggregation
    .sort(col("n_orders").desc()) # 4. Sort
)

display(df_chained)
> See performance (1)
```

df\_chained: pyspark.sql.connect.DataFrame = [o\_orderpriority: string, n\_orders: long]

Table +

	<sup>B</sup> c o_orderpriority	<sup>I<sub>2</sub></sup> <sub>3</sub> n_orders
1	1-URGENT	731447
2	4-NOT SPECIFIED	731011
3	5-LOW	730923
4	3-MEDIUM	730832
5	2-HIGH	730288

↓ ▾ 5 rows | 1.29s runtime

# Visualize your DataFrame

02:35 PM (1s) 108 Python Optimize

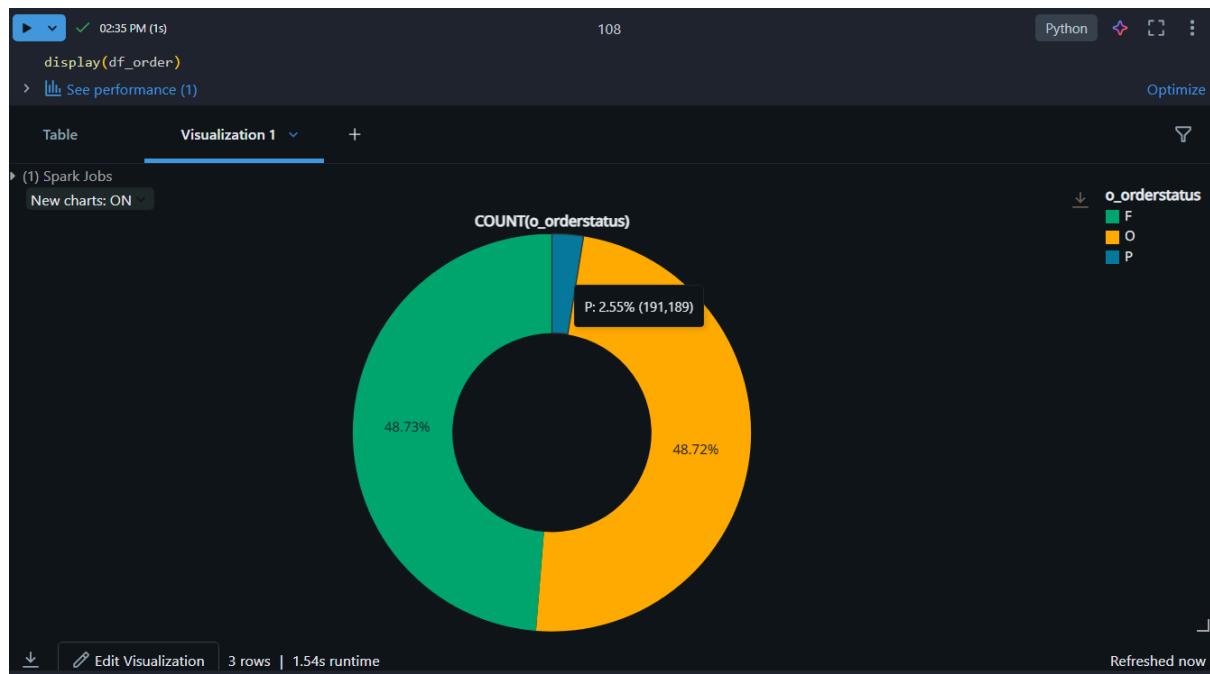
```
display(df_order)
> See performance (1)
```

Table Visualization 1 +

	$\text{o\_orderkey}$	$\text{o\_custkey}$	$\text{o\_orderstatus}$	$\text{o\_totalprice}$	$\text{o\_orderdate}$	$\text{o\_orderpriority}$	$\text{o\_clerk}$	$\text{o\_shippin}$
1	5611649	687736	O	51905.72	1996-06-04	5-LOW	Clerk#000002954	
2	5611650	513292	O	62845.59	1997-02-28	5-LOW	Clerk#000002092	
3	5611651	395308	F	226256.25	1992-03-05	1-URGENT	Clerk#000004123	
4	5611652	423847	O	141103.54	1995-08-07	3-MEDIUM	Clerk#000003975	
5	5611653	90844	O	157430.11	1996-08-14	5-LOW	Clerk#000004721	
6	5611654	540176	O	110800.10	1998-05-02	1-URGENT	Clerk#000004018	
7	5611655	733111	O	136877.11	1998-02-16	1-URGENT	Clerk#000006114	
8	5611680	366310	O	198245.15	1996-12-25	5-LOW	Clerk#000004898	
9	5611681	555686	F	240819.88	1992-03-19	4-NOT SPECIFIED	Clerk#000004636	
10	5611682	412018	F	203722.37	1992-09-13	3-MEDIUM	Clerk#000002208	
11	5611683	10091	O	148587.17	1996-01-07	5-LOW	Clerk#000003479	
12	5611684	88135	F	162936.96	1994-11-03	1-URGENT	Clerk#000002488	
13	5611685	16594	O	72678.09	1997-08-25	1-URGENT	Clerk#000000145	
14	5611686	617351	O	59154.35	1995-08-11	4-NOT SPECIFIED	Clerk#000003673	
15								

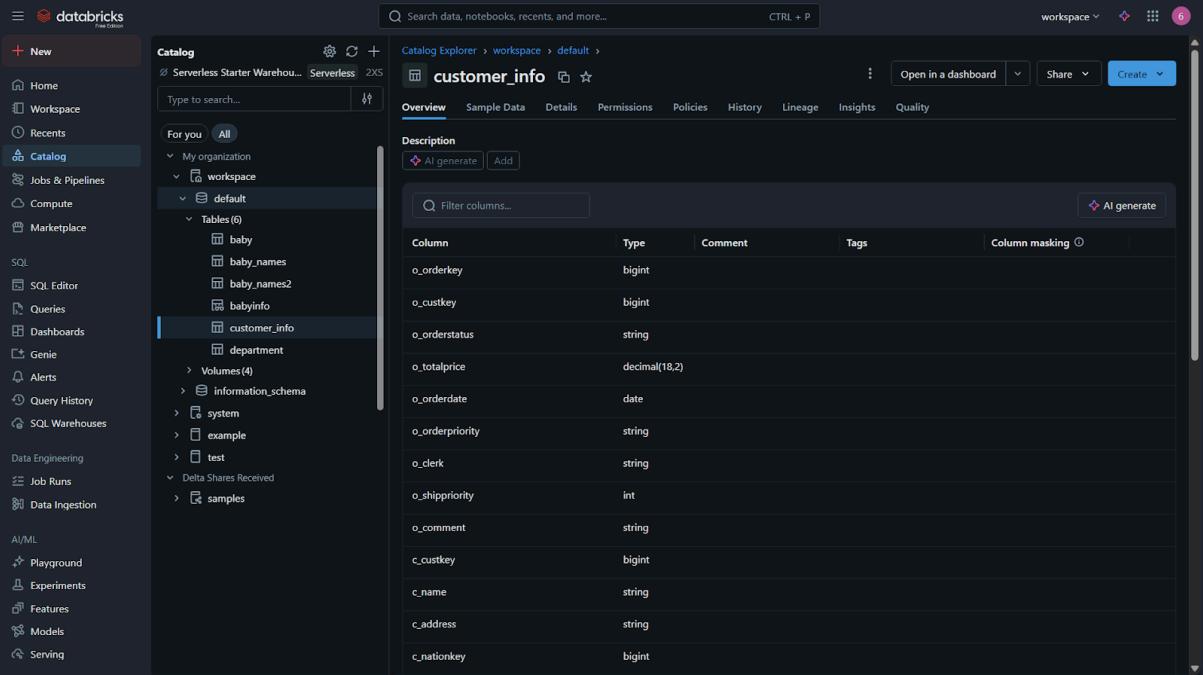
10,000+ rows | Truncated data | 1.21s runtime Refreshed 1 hour ago

ตัวอย่าง Viz



# Save your data

## Save your DataFrame as a table



The screenshot shows the Databricks Catalog Explorer interface. On the left, the sidebar navigation includes Home, Workspace, Recents, Catalog (selected), Jobs & Pipelines, Compute, Marketplace, SQL, SQL Editor, Queries, Dashboards, Genie, Alerts, Query History, and SQL Warehouses. Under Data Engineering, it lists Job Runs and Data Ingestion. Under AI/ML, it lists Playground, Experiments, Features, Models, and Serving.

The main area displays the Catalog Explorer for the workspace/default catalog. The 'customer\_info' table is selected. The table schema is shown in a table:

Column	Type	Comment	Tags	Column masking
<code>o_orderkey</code>	<code>bigint</code>			
<code>o_custkey</code>	<code>bigint</code>			
<code>o_orderstatus</code>	<code>string</code>			
<code>o_totalprice</code>	<code>decimal(18,2)</code>			
<code>o_orderdate</code>	<code>date</code>			
<code>o_orderpriority</code>	<code>string</code>			
<code>o_clerk</code>	<code>string</code>			
<code>o_shippriority</code>	<code>int</code>			
<code>o_comment</code>	<code>string</code>			
<code>c_custkey</code>	<code>bigint</code>			
<code>c_name</code>	<code>string</code>			
<code>c_address</code>	<code>string</code>			
<code>c_nationkey</code>	<code>bigint</code>			

# Write your DataFrame as CSV

The screenshot shows the Catalog Explorer interface. On the left, the navigation tree displays the 'Serverless Starter Warehouse' and 'Serverless' sections, with 'workspace' selected. Under 'workspace', 'default' is expanded, showing 'Tables(6)' and 'Volumes(4)'. One of the volumes, 'test\_n', is selected and expanded, revealing sub-directories like 'information\_schema', 'system', 'example', and 'test'. A folder named 'tutorial' is also visible under 'test\_n'. On the right, the 'tutorial' workspace is shown in more detail. The 'Overview' tab is selected, displaying the path '/Volumes/workspace/default/tutorial / cust\_info'. Below this is a file list table with columns 'Name', 'Size', and 'Last modified'. The table contains 10 entries, all of which are part files for a Delta table, starting with '\_SUCCESS', '\_committed', and '\_started', followed by 'part-00000' through 'part-00008'. All files are 0.00 B in size and were last modified 1 hour ago.

Name	Size	Last modified
_SUCCESS	0.00 B	1 hour ago
_committed_6925484220085621748	825.00 B	1 hour ago
_started_6925484220085621748	0.00 B	1 hour ago
part-00000-tid-6925484220085621748-7b0296a4-910d-42f	246.44 MB	1 hour ago
part-00001-tid-6925484220085621748-7b0296a4-910d-42f	247.30 MB	1 hour ago
part-00002-tid-6925484220085621748-7b0296a4-910d-42f	247.75 MB	1 hour ago
part-00003-tid-6925484220085621748-7b0296a4-910d-42f	248.18 MB	1 hour ago
part-00004-tid-6925484220085621748-7b0296a4-910d-42f	247.33 MB	1 hour ago
part-00005-tid-6925484220085621748-7b0296a4-910d-42f	246.27 MB	1 hour ago
part-00006-tid-6925484220085621748-7b0296a4-910d-42f	247.00 MB	1 hour ago
part-00007-tid-6925484220085621748-7b0296a4-910d-42f	246.53 MB	1 hour ago
part-00008-tid-6925484220085621748-7b0296a4-910d-42f	9.99 MB	1 hour ago

This screenshot shows the 'tutorial' workspace again, but with a different directory path: '/Volumes/workspace/default/tutorial / cust\_info\_csv\_ver'. The 'Overview' tab is selected. The file list table shows four entries: '\_SUCCESS', '\_committed\_5097961829365563432', '\_started\_5097961829365563432', and 'part-00000-tid-5097961829365563432-5e831edc-3ef0-492b'. The 'part' file is 1.94 GB in size and was last modified 2 hours ago. The other three files are 0.00 B in size and were last modified 2 hours ago.

Name	Size	Last modified
_SUCCESS	0.00 B	2 hours ago
_committed_5097961829365563432	113.00 B	2 hours ago
_started_5097961829365563432	0.00 B	2 hours ago
part-00000-tid-5097961829365563432-5e831edc-3ef0-492b	1.94 GB	2 hours ago