

Load and transform data using DataFrames

[docs](#)

Dataframe

- DataFrame เป็นข้อมูลที่มี labeled รูปแบบ 2 มิติ
- อาจจะมองแบบ Spreadsheet, ตาราง SQL

Apache Spark DataFrames

- ในส่วนของ Apache Spark DataFrames มี function ที่มากมาย ช่วยในการทำ data analysis ได้ ex: select columns, filter, join, aggregate
- สร้างบน RDDs
- มีประสิทธิภาพสูง เพราะใช้ unified planning and optimization engine กับ Spark DF& Spark SQL ก็จะได้ประสิทธิภาพที่เกือบจะเหมือนกันในทุกภาษาที่ DataBricks รองรับ(Py, SQL, Scala, R)

Requirements

- Unity Catalog ต้องใช้งานได้ (default เราเมื่อยังไม่ได้)
- ต้องสร้าง Volume มาเก็บข้อมูลที่ใช้ใน tut นี้
- ต้องมีสิทธิ์เหล่านี้ใน Unity Catalog : READ VOLUME and WRITE VOLUME, use schema, use catalog or ALL PRIVILEGES สำหรับ Volume, Schema, Catalog ที่ใช้

Define variables and load CSV file

สร้าง Notebook ใหม่ > วิธีสร้าง Notebook เราเคยทำกันไปแล้ว

ใช้ volume เดิมที่เราเคยสร้างไว้ หรือจะสร้าง volume ใหม่ก็ได้ ในที่นี่สร้าง volume ใหม่ ซื้อ tutorial มารองรับไว้

จากนั้นกำหนดตัวแปร ประมาณนี้

- rename : <catalog_name>, <schema_name>, <volume_name> เป็นสิ่งที่เรามี ในที่นี่จะใช้ตามในภาพด้านล่างต่อไป
- <table_name> ใช้ชื่อแบบที่เราอยากรังสรรค์ ในที่นี่ ตั้งชื่อว่า baby

```

catalog = "<catalog_name>"
schema = "<schema_name>"
volume = "<volume_name>"
download_url = "https://health.data.ny.gov/api/views/jxy9-yhdk/rows.csv"
file_name = "rows.csv"
table_name = "<table_name>"
path_volume = "/Volumes/" + catalog + "/" + schema + "/" + volume
path_table = catalog + "." + schema
print(path_table) # Show the complete path
print(path_volume) # Show the complete path

```

```

1 catalog = "workspace"
2 schema = "default"
3 volume = "tutorial"
4 download_url = "https://health.data.ny.gov/api/views/jxy9-yhdk/rows.csv"
5 file_name = "rows.csv"
6 table_name = "baby"
7 path_volume = "/Volumes/" + catalog + "/" + schema + "/" + volume
8 path_table = catalog + "." + schema
9 print(path_table) # Show the complete path
10 print(path_volume) # Show the complete path

```

ต่อมา ใช้โค้ดด้านล่างเพื่อ copy ไฟล์ rows.csv จาก [source](#) มาเก็บไว้ใน Volume ที่เราเตรียมไว้

```
dbutils.fs.cp(f"{download_url}", f"{path_volume}/{file_name}")
```

Create a DataFrame

สร้าง DataFrame จากโค้ดนี้

```

data = [[2021, "test", "Albany", "M", 42]]
columns = ["Year", "First_Name", "County", "Sex", "Count"]

df1 = spark.createDataFrame(data, schema="Year int, First_Name STRING, County STRING, Sex STRING,
Count int")

```

```
display(df1)
```

รายละเอียด `display()` & `show()` command ที่อยู่ใน comment

The `display()` method is specific to Databricks notebooks and provides a richer visualization.
`df1.show()` The `show()` method is a part of the Apache Spark DataFrame API and provides basic visualization.

จะได้ผลลัพธ์แบบในรูปข้างล่าง

	Year	First Name	County	Sex	Count
1	2021	test	Albany	M	42

Load data into a DataFrame from CSV file

ในส่วนนี้เราจะ Load Data มาเป็น DataFrame โดยใช้ข้อมูลจาก CSV file ด้วยโค้ดข้างล่างนี้ ซึ่ง path เราได้กำหนดไปตั้งแต่ตอนกำหนดค่าตัวแปรตอนแรกแล้ว

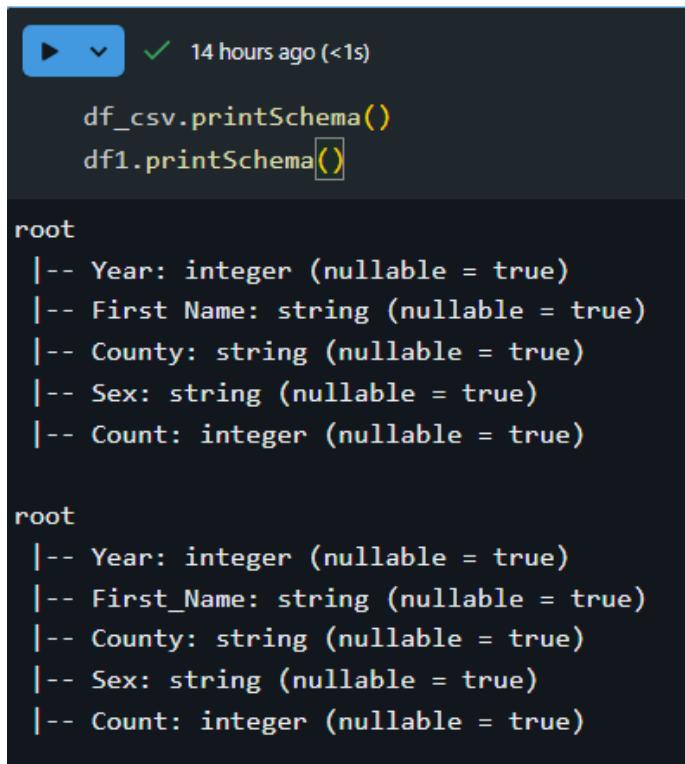
```
df_csv = spark.read.csv(f"{path_volume}/{file_name}",
    header=True,
    inferSchema=True,
    sep=",")
display(df_csv)
```

	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
8	2022	CORA	Albany	F	8
9	2022	MIA	Albany	F	7

View and interact with your DataFrame

ตรวจสอบ Schema ของ DF ทั้ง 2

```
df_csv.printSchema()  
df1.printSchema()
```

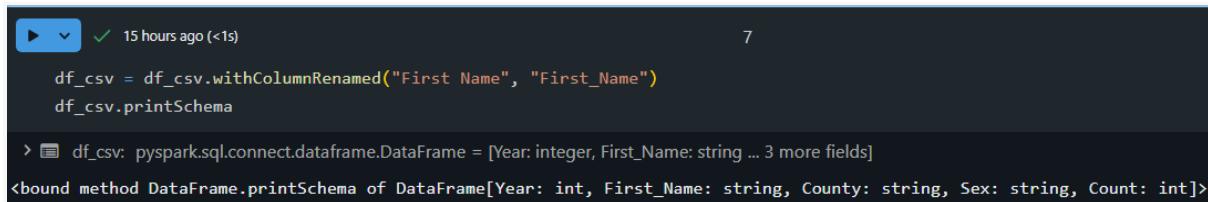


```
14 hours ago (<1s)  
  
df_csv.printSchema()  
df1.printSchema()  
  
root  
| -- Year: integer (nullable = true)  
| -- First Name: string (nullable = true)  
| -- County: string (nullable = true)  
| -- Sex: string (nullable = true)  
| -- Count: integer (nullable = true)  
  
root  
| -- Year: integer (nullable = true)  
| -- First_Name: string (nullable = true)  
| -- County: string (nullable = true)  
| -- Sex: string (nullable = true)  
| -- Count: integer (nullable = true)
```

จากรูปจะเห็นว่า Schema ของทั้ง 2 DF เกือบจะเหมือนกันแล้ว ยกเว้น First_Name field เราจะเปลี่ยนชื่อมันด้วย withColumnRenamed()

```
df_csv = df_csv.withColumnRenamed("First Name", "First_Name")  
df_csv.printSchema
```

จะได้ schema แบบนี้ (ถ้าไม่ได้ใส่ () ต่อท้าย)



```
15 hours ago (<1s) 7  
  
df_csv = df_csv.withColumnRenamed("First Name", "First_Name")  
df_csv.printSchema  
> df_csv: pyspark.sql.connect.DataFrame = [Year: integer, First_Name: string ... 3 more fields]  
<bound method DataFrame.printSchema of DataFrame[Year: int, First_Name: string, County: string, Sex: string, Count: int]>
```

รวม 2 DF ที่มี schema เหมือนกันด้วย union()

```
df = df1.union(df_csv)
display(df)
```

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

```
df = df1.union(df_csv)
display(df)
df.printSchema
```

Below the code, there is a link to "See performance (1)". The output section shows the resulting DataFrame:

	Year	First_Name	County	Sex	Count
1	2021	test	Albany	M	42
2	2022	OLIVIA	Albany	F	16
3	2022	AMELIA	Albany	F	15
4	2022	AVERY	Albany	F	12
5	2022	EMMA	Albany	F	11
6	2022	CHARLOTTE	Albany	F	11
7	2022	CHLOE	Albany	F	11
8	2022	SOPHIA	Albany	F	9

Filter rows in a DataFrame

.filter() method

ใช้ filter() เพื่อ filter หาแถวที่มี Count > 50

```
display(df.filter(df["Count"] > 50))
```

ผลลัพธ์ตามรูปข้างมือ

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

```
display(df.filter(df["Count"] > 50))
> See performance (1)
```

Below the code, there is a link to "See performance (1)". The output section shows the filtered DataFrame:

	Year	First_Name	County	Sex	Count
1	2022	SOPHIA	Bronx	F	65
2	2022	LUNA	Bronx	F	65
3	2022	EMMA	Bronx	F	60
4	2022	ISABELLA	Bronx	F	55
5	2022	LIAM	Bronx	M	156
6	2022	NOAH	Bronx	M	132
7	2022	ETHAN	Bronx	M	84
8	2022	JACOB	Bronx	M	68
9	2022	JAYDEN	Bronx	M	65
10	2022	AMIR	Bronx	M	64
11	2022	AIDEN	Bronx	M	59
12	2022	SEBASTIAN	Bronx	M	56
13	2022	DYLAN	Bronx	M	54
14	2022	MATTHEW	Bronx	M	54
15	2022	ESTHER	Kings	F	198

At the bottom, it says "5,815 rows | 1.43s runtime".

.where() method

ที่เราจะหาคือเมื่อกันกับหัวข้อ filter
แต่เปลี่ยนเป็น where

```
display(df.where(df["Count"] > 50))
```

ผลลัพธ์ตามภาพข้างมือ

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

```
display(df.where(df["Count"] > 50))
```

Below the code, there is a table output showing 15 rows of data. The columns are labeled Year, First_Name, County, Sex, and Count. The data includes names like SOPHIA, LUNA, EMMA, ISABELLA, LIAM, NOAH, ETHAN, JACOB, JAYDEN, AMIR, AIDEN, SEBASTIAN, DYLAN, MATTHEW, and ESTHER, along with their counts and other details.

	Year	First_Name	County	Sex	Count
1	2022	SOPHIA	Bronx	F	65
2	2022	LUNA	Bronx	F	65
3	2022	EMMA	Bronx	F	60
4	2022	ISABELLA	Bronx	F	55
5	2022	LIAM	Bronx	M	156
6	2022	NOAH	Bronx	M	132
7	2022	ETHAN	Bronx	M	84
8	2022	JACOB	Bronx	M	68
9	2022	JAYDEN	Bronx	M	65
10	2022	AMIR	Bronx	M	64
11	2022	AIDEN	Bronx	M	59
12	2022	SEBASTIAN	Bronx	M	56
13	2022	DYLAN	Bronx	M	54
14	2022	MATTHEW	Bronx	M	54
15	2022	ESTHER	Kings	F	198

Select columns from a DataFrame and order by frequency

ดูความถี่ของชื่อเด็กๆ ด้วยการใช้ select(), orderBy(), desc()

```
from pyspark.sql.functions import desc
display(df.select("First_Name", "Count").orderBy(desc("Count")))
```

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

```
from pyspark.sql.functions import desc
display(df.select("First_Name", "Count").orderBy(desc("Count")))
```

Below the code, there is a table output showing 9 rows of data. The columns are labeled First_Name and Count. The data includes names like DAVID, ETHAN, JAYDEN, and DAVID again, ordered by their frequency in descending order.

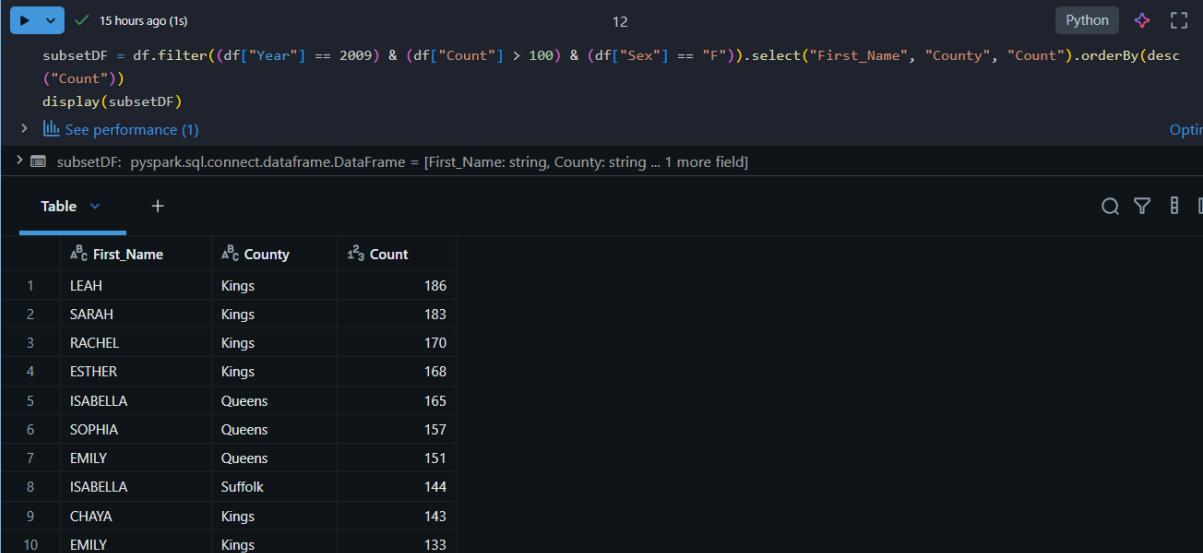
	First_Name	Count
1	DAVID	297
2	DAVID	287
3	ETHAN	273
4	DAVID	273
5	LIAM	272
6	DAVID	272
7	JAYDEN	268
8	JAYDEN	268
9	DAVID	267

Create a Subset DF

สร้าง Subset จาก DF ที่มีอยู่แล้ว ในที่นี่คือ สร้าง DF โดย filter ข้อมูลตามปี จำนวน และเพศ ใช้ select() เพื่อเลือกคอลัมน์ ใช้ orderBy() & desc เพื่อเรียงลำดับ DataFrame ใหม่ตามจำนวน จำนวนมากไปน้อย

```
subsetDF = df.filter((df["Year"] == 2009) & (df["Count"] > 100) & (df["Sex"] == "F")).select("First_Name", "County", "Count").orderBy(desc("Count"))
```

```
display(subsetDF)
```



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

```
subsetDF = df.filter((df["Year"] == 2009) & (df["Count"] > 100) & (df["Sex"] == "F")).select("First_Name", "County", "Count").orderBy(desc("Count"))
display(subsetDF)
> See performance (1)
subsetDF: pyspark.sql.connect.DataFrame = [First_Name: string, County: string ... 1 more field]
```

The cell has a status bar indicating "15 hours ago (1s)" and "12". The output area shows a table with the following data:

	First_Name	County	Count
1	LEAH	Kings	186
2	SARAH	Kings	183
3	RACHEL	Kings	170
4	ESTHER	Kings	168
5	ISABELLA	Queens	165
6	SOPHIA	Queens	157
7	EMILY	Queens	151
8	ISABELLA	Suffolk	144
9	CHAYA	Kings	143
10	EMILY	Kings	133

Save the DataFrame

Save DF to a Table

Code : df.write.mode("overwrite").saveAsTable(f"{path_table}.{table_name}")

Save เป็น Table เก็บไว้ สามารถแกะไปดู Table ตาม path ที่เรากำหนดไว้ตามปกติ ซึ่งตัวแปร path และชื่อ Table เราตั้งตัวแปรไว้แล้วตั้งแต่แรก

Save DF to JSON files

Code :

df.write.format("json").mode("overwrite").save("dbfs:/Volumes/workspace/default/tutorial/json_data/")

ที่ Highlight คือ Path ที่เราจะเก็บข้อมูล Json ไว้ ซึ่งตามใน docs จะใช้ /tmp/json_data ซึ่งเรามีมีสิทธิ์เข้าถึงส่วนนี้ เราเลยเปลี่ยน path เป็น volume ที่เราจะเก็บข้อมูลแทน ซึ่งในที่นี่เราใช้ volume = tutorial และเขียนใส่ folder = json_data

หน้าตาหลังจาก save จะเป็นประมาณนี้ ข้อมูลจะแบ่ง part เก็บตามที่ระบบจัดการให้

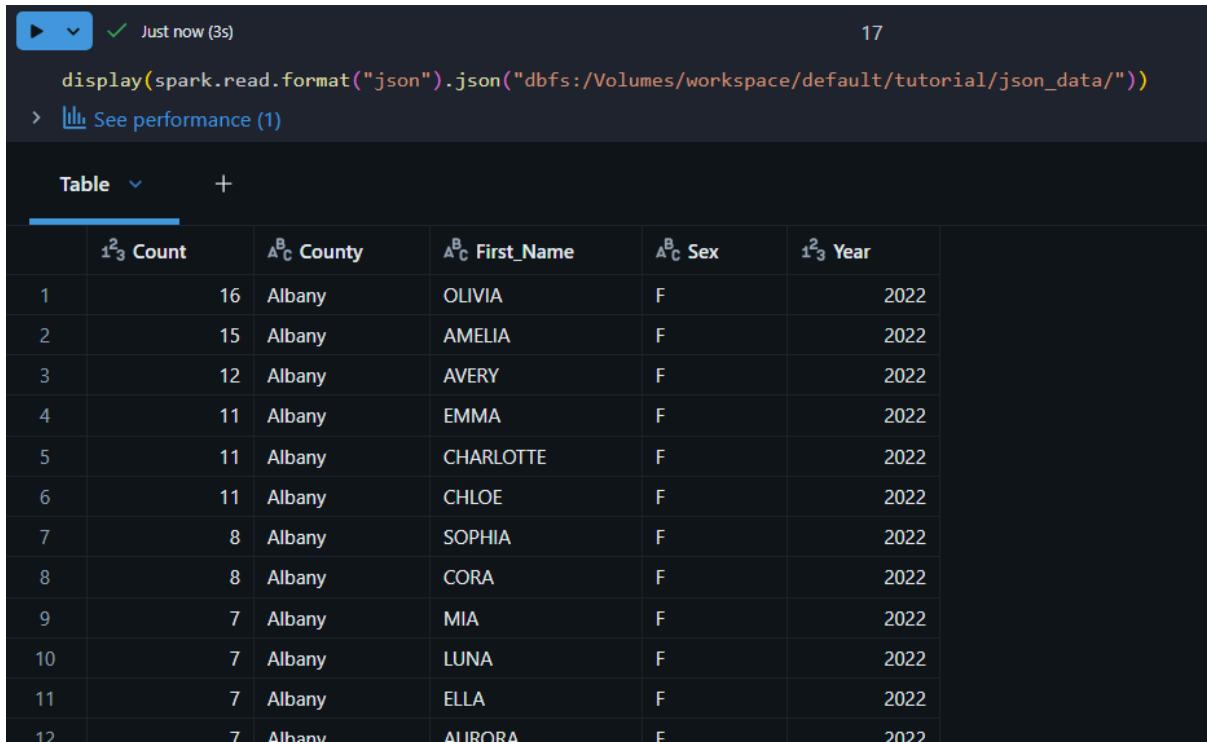
Name	Size	Last modified
_SUCCESS	0.00 B	16 hours ago
_committed_958380714305718895	380.00 B	16 hours ago
_started_958380714305718895	0.00 B	16 hours ago
part-00000-tid-958380714305718895-d413dd3b-73d0-4a88	73.00 B	16 hours ago
part-00001-tid-958380714305718895-d413dd3b-73d0-4a88	3.12 MB	16 hours ago
part-00002-tid-958380714305718895-d413dd3b-73d0-4a88	3.11 MB	16 hours ago
part-00003-tid-958380714305718895-d413dd3b-73d0-4a88	891.35 KB	16 hours ago

Read DF from JSON file

Code :

display(spark.read.format("json").json("dbfs:/Volumes/workspace/default/tutorial/json_data/"))
ตอนมันอ่านมันจะอ่านจาก folder ที่เราเก็บข้อมูลไว้ (ใส่ path ให้ถูก)

จากตัวอย่างตอนอ่านօกมาผลจะไม่เหมือนกับผลหลังจากเรา union ไป เเลยทำการเช็คจำนวนเควด้วย count() พบว่า จำนวนเควดเท่ากันกับ df ที่มี



	Count	County	First_Name	Sex	Year
1	16	Albany	OLIVIA	F	2022
2	15	Albany	AMELIA	F	2022
3	12	Albany	AVERY	F	2022
4	11	Albany	EMMA	F	2022
5	11	Albany	CHARLOTTE	F	2022
6	11	Albany	CHLOE	F	2022
7	8	Albany	SOPHIA	F	2022
8	8	Albany	CORA	F	2022
9	7	Albany	MIA	F	2022
10	7	Albany	LUNA	F	2022
11	7	Albany	ELLA	F	2022
12	7	Albany	AURORA	F	2022

ผลลัพธ์หลังจากอ่าน ตามภาพข้างบน ^

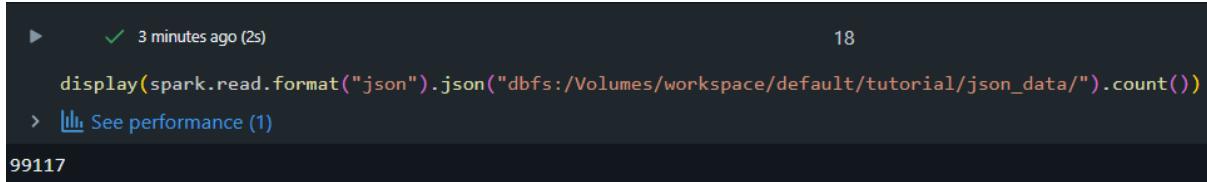
จำนวนเควด



5 minutes ago (1s)

```
df.count()
```

99117 < อันนี้คือหลังจาก union และยังไม่ได้ save ในรูปแบบ json



3 minutes ago (2s)

```
display(spark.read.format("json").json("dbfs:/Volumes/workspace/default/tutorial/json_data/")).count()
```

99117 ^ อันนี้คืออ่าน json จาก path ที่เรา save ไว้

Run SQL queries in PySpark, Scala, and R

ใช้ `selectExpr()` เพื่อรับ SQL expressions แบบ String
(ถ้า `select()` เฉยๆ จะรับแค่ string ที่เป็นชื่อคอลัมน์เท่านั้น)

ตัวอย่าง เช่น เราจะเลือก `col : Count` และ `County` แต่อยากแสดงผล `County` เป็นตัวพิมพ์ใหญ่
ทั้งหมด และเปลี่ยนให้ชื่อคอลัมน์ที่แสดงเป็น `big_name` จะใช้โค้ดดังนี้

Code : `display(df.selectExpr("Count", "upper(County) as big_name"))`

ผลลัพธ์จะเป็นประมาณนี้

```
▶  ✓  12:33 PM (1s)

display(df.selectExpr("Count", "upper(County) as big_name"))

> See performance (1)

Table +
```

Count	big_name
42	ALBANY
16	ALBANY
15	ALBANY
12	ALBANY
11	ALBANY
11	ALBANY
11	ALBANY
8	ALBANY
8	ALBANY
7	ALBANY
7	ALBANY
7	ALBANY

หรือจะใช้รูปแบบ `expr()` ร่วมกับการ `select()` ก็ได้เช่นกัน

```
from pyspark.sql.functions import expr *  
display(df.select("Count", expr("lower(County) as little_name")))
```

▶ ✓ 12:33 PM (1s) 20

```
from pyspark.sql.functions import expr  
display(df.select("Count", expr("lower(County) as little_name")))
```

› [See performance \(1\)](#)

Table +

	Count	little_name
1	42	albany
2	16	albany
3	15	albany
4	12	albany
5	11	albany
6	11	albany
7	11	albany
8	8	albany
9	8	albany
10	7	albany
11	7	albany

ยังไม่พอ หากต้องการใช้ SQL เลยก็ย่อได้ ด้วยคำสั่ง `spark.sql()`

```
code : display(spark.sql(f"SELECT * FROM {path_table}.{table_name}"))
```

▶ ✓ 12:33 PM (2s) 21

```
display(spark.sql(f"SELECT * FROM {path_table}.{table_name}"))
```

› [See performance \(1\)](#)

Table +

	Year	First_Name	County	Sex	Count
1	2022	OLIVIA	Albany	F	16
2	2022	AMELIA	Albany	F	15
3	2022	AVERY	Albany	F	12