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 ซึ่งช่วยในการทำ 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 เรามีอยู่แล้ว)
- 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 ในที่นี้สร้าง volume ใหม่ ชื่อ tutorial มารองรับไว้

จากนั้นกำหนดตัวแปร ประมาณนี้

- rename : <catalog_name>, <schema_name>, <volume_name> เป็นสิ่งที่เรามี ในที่นี้จะใช้ตามในภาพด้านล่างต่อไป
- <table_name> ใช้ชื่อแบบที่เราอยากตั้ง ในที่นี้ 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

```
catalog = "workspace"
schema = "default"
volume = "tutorial"
download_url = "https://health.data.ny.gov/api/views/jxy9-yhdk/rows.csv"
file_name = "rows.csv"
table_name = "baby"
path_volume = "/Volumes/" + catalog + "/" + schema + "/" + volume
path_table = catalog + "." + schema
print(path_table) # Show the complete path
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.

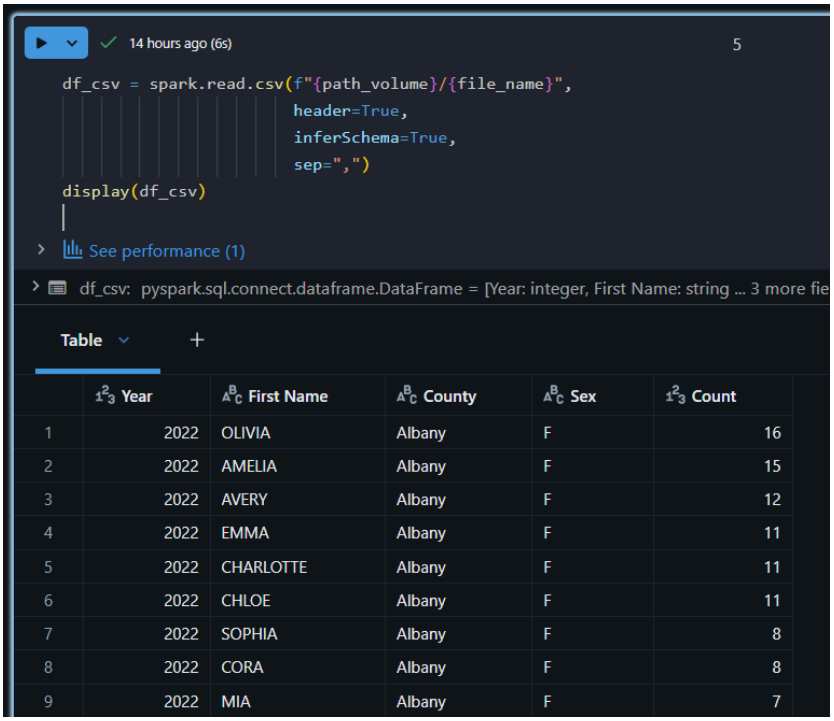
df1.show() The show() method is a part of the Apache Spark DataFrame API and provides basic visualization.

จะได้ผลลัพธ์แบบในรูปข้างล่าง

Load data into a DataFrame from CSV file

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

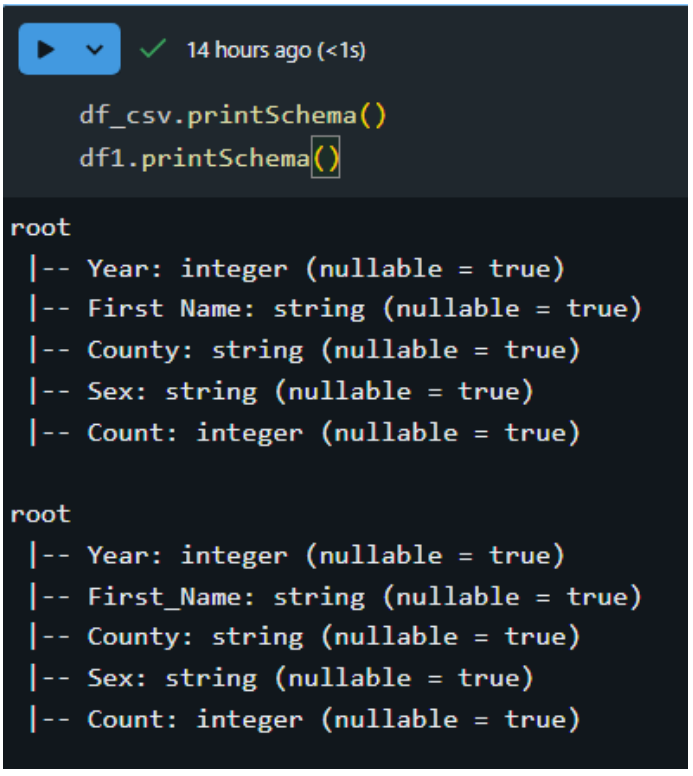
```
df_csv = spark.read.csv(f"{path_volume}/{file_name}",
    header=True,
    inferSchema=True,
    sep=",")
display(df_csv)
```



View and interact with your DataFrame

ตรวจสอบ Schema ของ DF ทั้ง 2

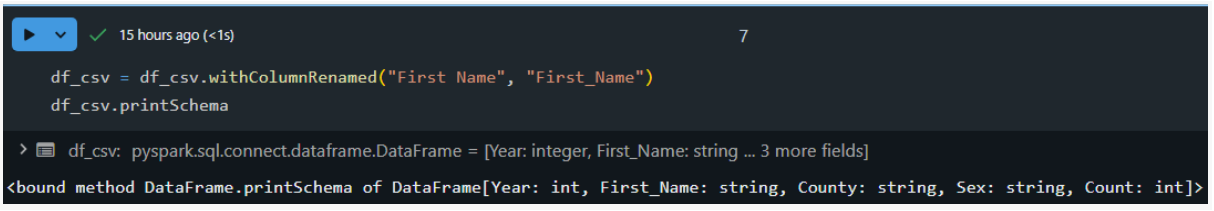
```
df_csv.printSchema()
df1.printSchema()
```



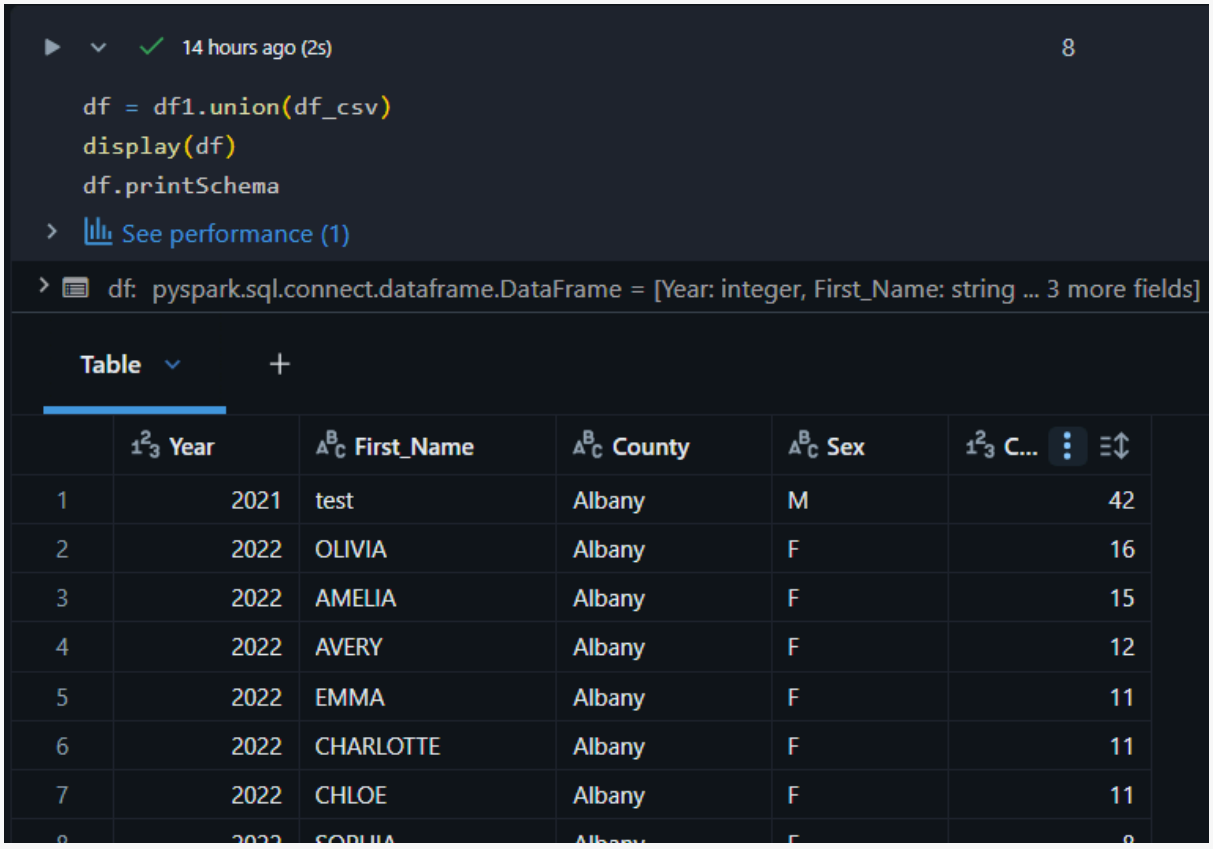
จากภาพจะเห็นว่า Schema ของทั้ง 2 DF เกือบจะเหมือนกันแล้ว ยกเว้น First_Name field เราจะเปลี่ยนชื่อมันด้วย `withColumnRenamed()`

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

จะได้ schema แบบนี้ (ถ้าไม่ได้ใส่ () ต่อท้าย)



รวม 2 DF ที่มี schema เหมือนกันด้วย union()
df = df1.union(df_csv)
display(df)



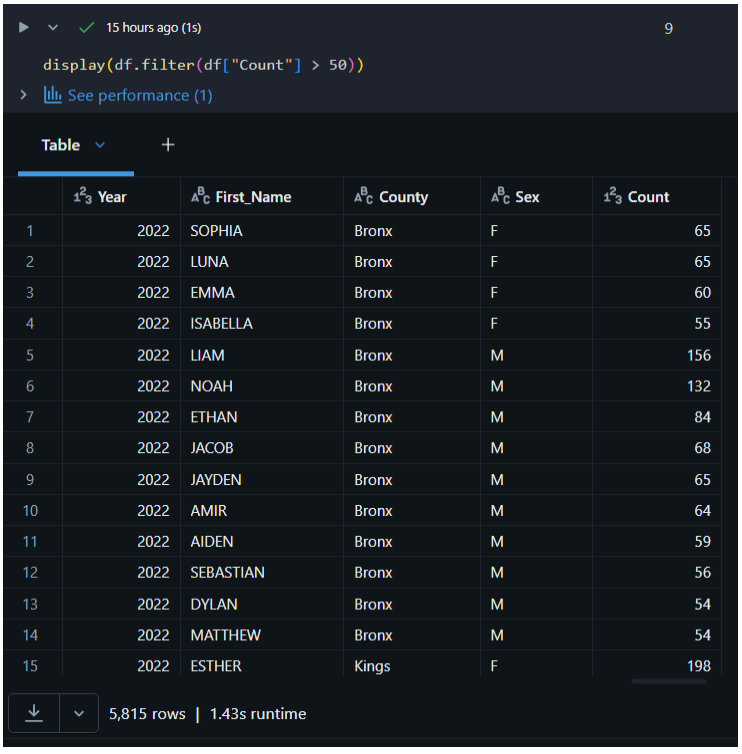
Filter rows in a DataFrame

.filter() method

ใช้ filter() เพื่อ filter หาแถวที่มี Count > 50

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

ผลลัพธ์ตามรูปขวามือ

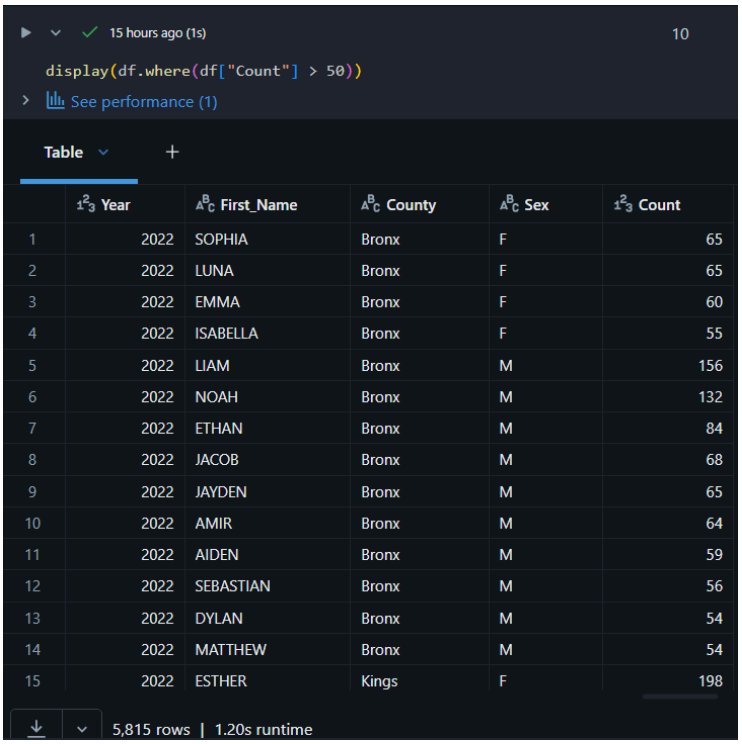


.where() method

ที่เราจะหาคือเหมือนกันกับหัวข้อ filter แต่เปลี่ยนเป็น where

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

ผลลัพธ์ตามภาพขวามือ

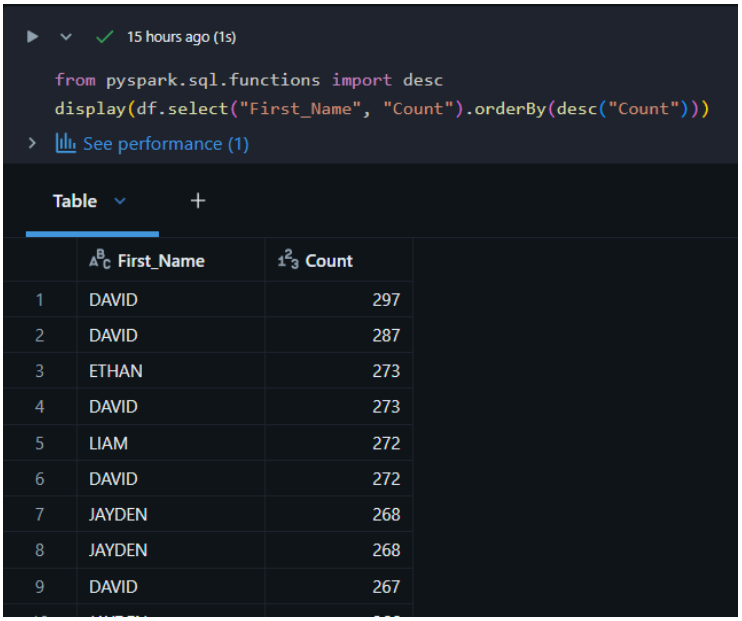


	Year	First_Name	County	Sex	Count
1	2022	SOPHIA	Bronx	F	65
2	2022	LUNA	Bronx	F	65
3	2022	EMMA	Bronx	F	60
4	2022	ISABELLA	Bronx	F	55
5	2022	LIAM	Bronx	M	156
6	2022	NOAH	Bronx	M	132
7	2022	ETHAN	Bronx	M	84
8	2022	JACOB	Bronx	M	68
9	2022	JAYDEN	Bronx	M	65
10	2022	AMIR	Bronx	M	64
11	2022	AIDEN	Bronx	M	59
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")))



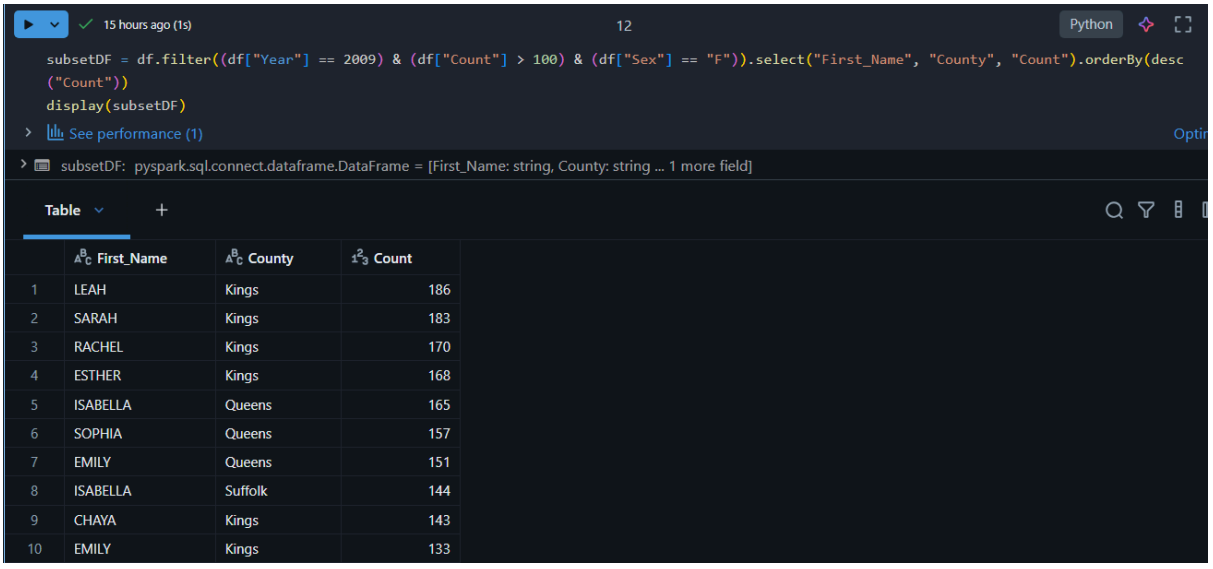
	First_Name	Count
1	DAVID	297
2	DAVID	287
3	ETHAN	273
4	DAVID	273
5	LIAM	272
6	DAVID	272
7	JAYDEN	268
8	JAYDEN	268
9	DAVID	267

Create a Subset DF

สร้าง Subset จาก DF ที่มีอยู่แล้ว ในที่นี้ คือ สร้าง DF โดย filter ข้อมูลตามปี จำนวน และเพศ ใช้ select() เพื่อเลือกคอลัมน์ ใช้ orderBy() & desc เพื่อเรียงลำดับ DataFrame ใหม่ตามจำนวนจากมากไปน้อย

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

display(subsetDF)
```



	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}")

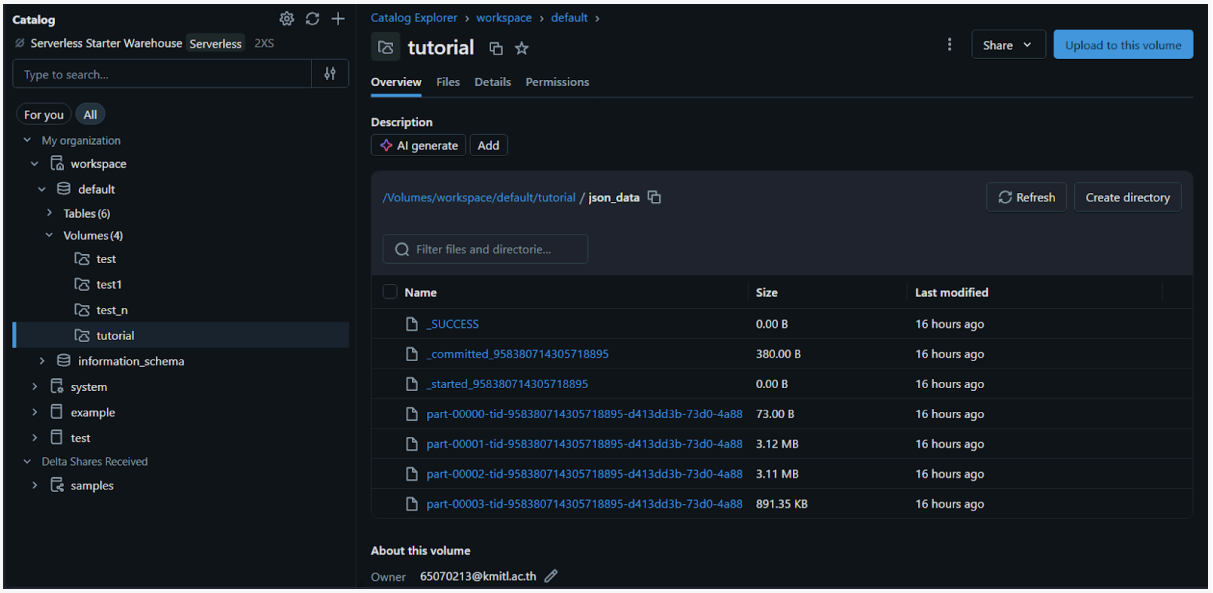
Save เป็น Table เก็บไว้ สามารถแวะไปดู Table ตาม path ที่เรากำหนดไว้ตามปกติ ซึ่งตัวแปร path และชื่อ Table เราตั้งตัวแปรไว้แล้วตั้งแต่แรก

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 ซึ่งเราไม่มีสิทธิ์เข้าถึงส่วนนี้ เราเลยเปลี่ยน path เป็น volume ที่เราจะเก็บข้อมูลแทน ซึ่งในที่นี้เราใช้ volume = tutorial และเขียนใส่ folder = json_data

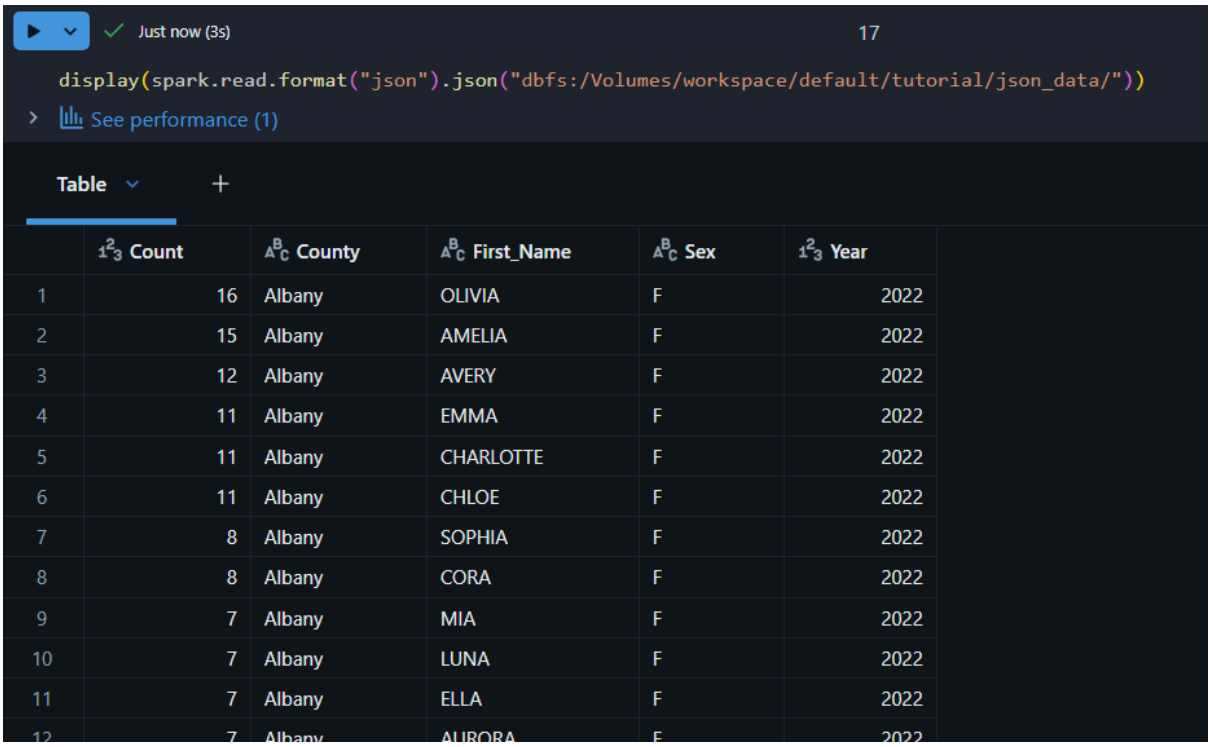
หน้าตาหลังจาก save จะเป็นประมาณนี้ ข้อมูลจะแบ่ง part เก็บตามที่ระบบจัดการให้



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)

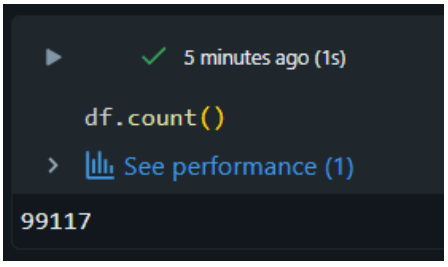
จากตัวอย่างตอนอ่านออกมาผลจะไม่เหมือนกับผลหลังจากเรา union ไป เลยทำการเช็คจำนวนแถวด้วย count() พบว่า จำนวนแถวเท่ากันกับ df ที่มี



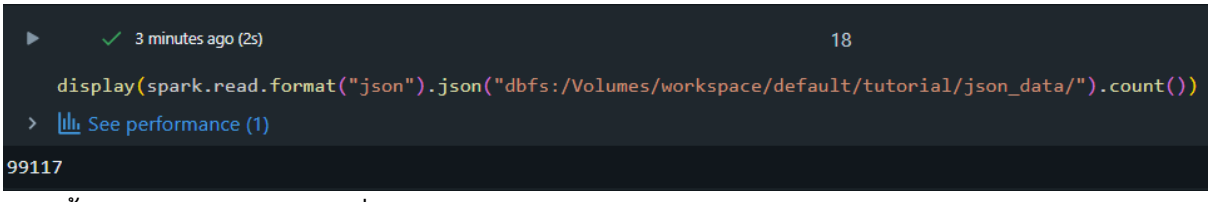
	Count	County	First_Name	Sex	Year
1	16	Albany	OLIVIA	F	2022
2	15	Albany	AMELIA	F	2022
3	12	Albany	AVERY	F	2022
4	11	Albany	EMMA	F	2022
5	11	Albany	CHARLOTTE	F	2022
6	11	Albany	CHLOE	F	2022
7	8	Albany	SOPHIA	F	2022
8	8	Albany	CORA	F	2022
9	7	Albany	MIA	F	2022
10	7	Albany	LUNA	F	2022
11	7	Albany	ELLA	F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

ผลลัพธ์หลังจากอ่าน ตามภาพข้างบน ^

จำนวนแถว



< อันนี้คือหลังจาก union และยังไม่ได้ save ในรูปแบบ json



^ อันนี้คืออ่าน json จาก path ที่เรา save ไว้

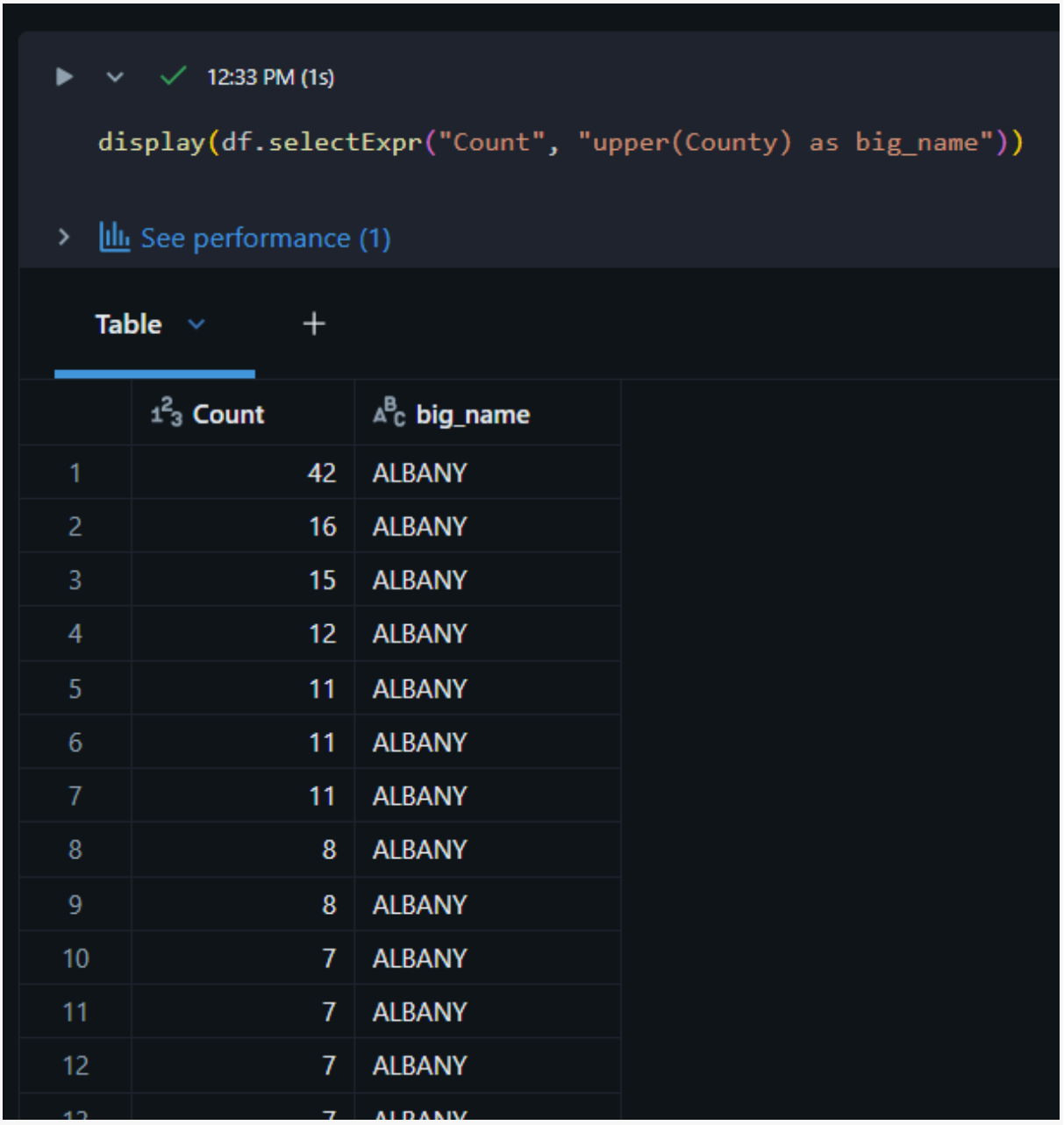
Run SQL queries in PySpark, Scala, and R

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

ตัวอย่าง เช่น เราจะเลือก col : Count และ County แต่อยากแสดงผล County เป็นตัวพิมพ์ใหญ่ทั้งหมด และเปลี่ยนให้ชื่อคอลัมน์ที่แสดงเป็น big_name จะใช้โค้ดดังนี้

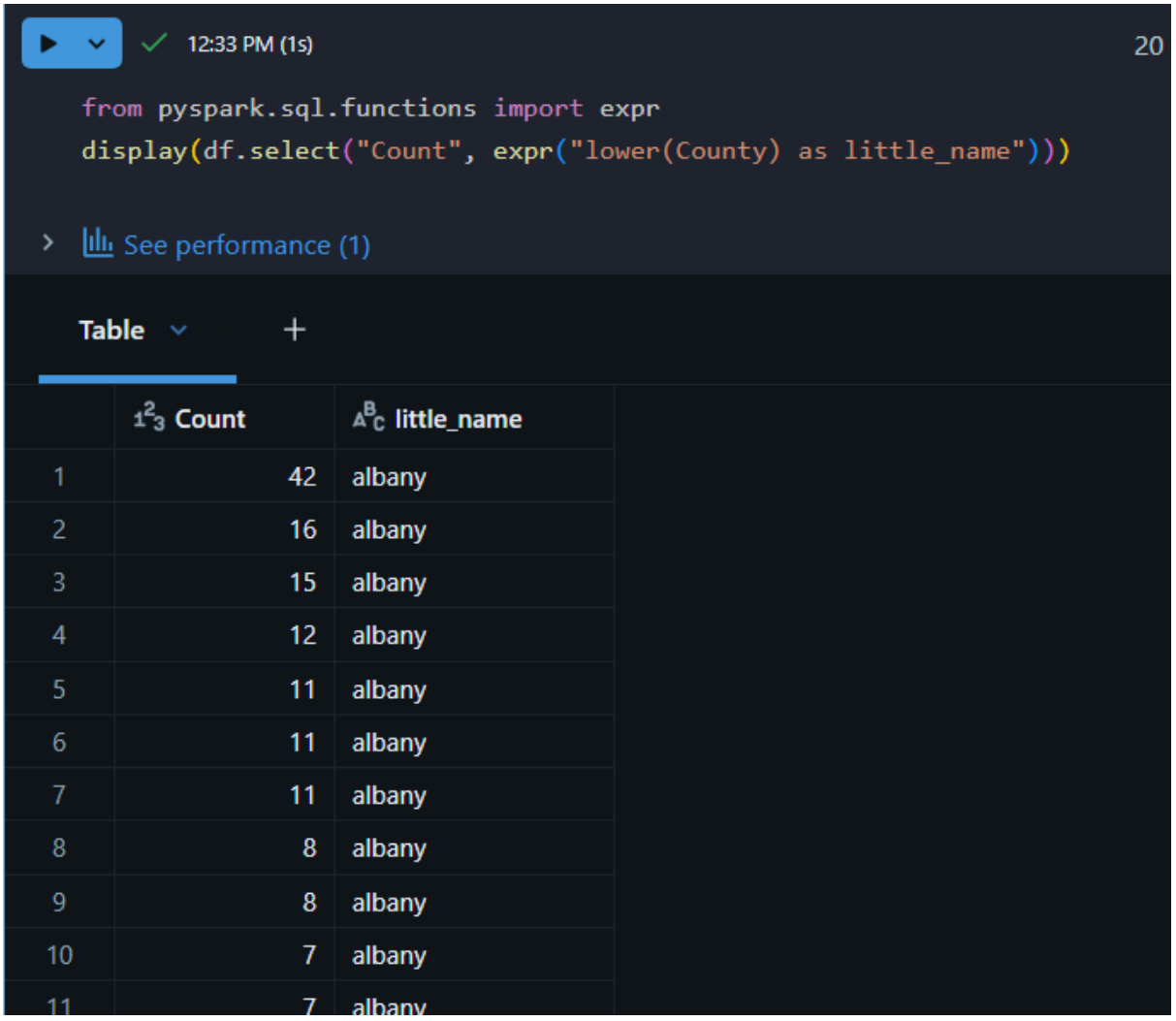
Code : display(df.selectExpr("Count", "upper(County) as big_name"))

ผลลัพธ์จะเป็นประมาณนี้



หรือจะใช้วิธี expr() ร่วมกับการ select() ก็ได้เช่นกัน

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



	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 เลยก็ย่อมได้ ด้วยคำสั่ง spark.sql()

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

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