

PANDAS VS POLARS VS SPARK VS DASK

Presentado por:

TANIA JULIET HURTADO RAMÍREZ

JUAN CAMILO SANCHEZ FERNANDEZ

Presentado a:

Ing ELIAS BUITRAGO BOLIVAR

Universidad ECCI

Ingeniería en Sistemas

Seminario Big Data & Gerencia de datos

Bogotá

2024

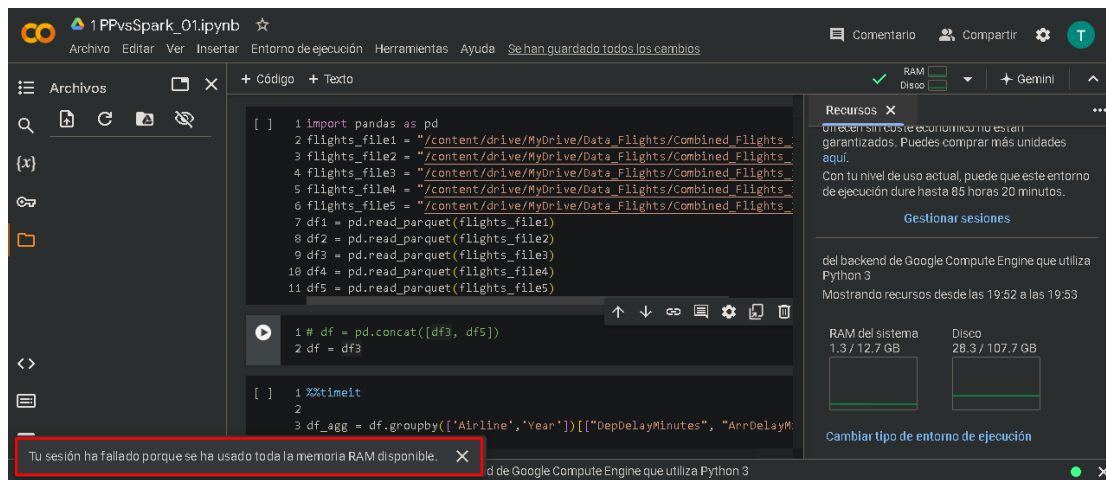
INTRODUCCIÓN

En este trabajo, realizaremos un experimento con las librerías Pandas, Polars, Spark y Dask utilizando cinco archivos de datos. El objetivo es determinar cuál de estas librerías es más eficiente en el procesamiento de los archivos.

DESARROLLO

Primero, subiremos los cinco archivos para observar cómo se comportan las distintas librerías.

PANDAS



The screenshot shows a Jupyter Notebook interface with a code editor on the left and a sidebar on the right. The code in the editor is as follows:

```
[ ] 1 import pandas as pd
2 flights_file1 = "/content/drive/MyDrive/Data_Flights/Combined_Flights_1.parquet"
3 flights_file2 = "/content/drive/MyDrive/Data_Flights/Combined_Flights_2.parquet"
4 flights_file3 = "/content/drive/MyDrive/Data_Flights/Combined_Flights_3.parquet"
5 flights_file4 = "/content/drive/MyDrive/Data_Flights/Combined_Flights_4.parquet"
6 flights_file5 = "/content/drive/MyDrive/Data_Flights/Combined_Flights_5.parquet"
7 df1 = pd.read_parquet(flights_file1)
8 df2 = pd.read_parquet(flights_file2)
9 df3 = pd.read_parquet(flights_file3)
10 df4 = pd.read_parquet(flights_file4)
11 df5 = pd.read_parquet(flights_file5)

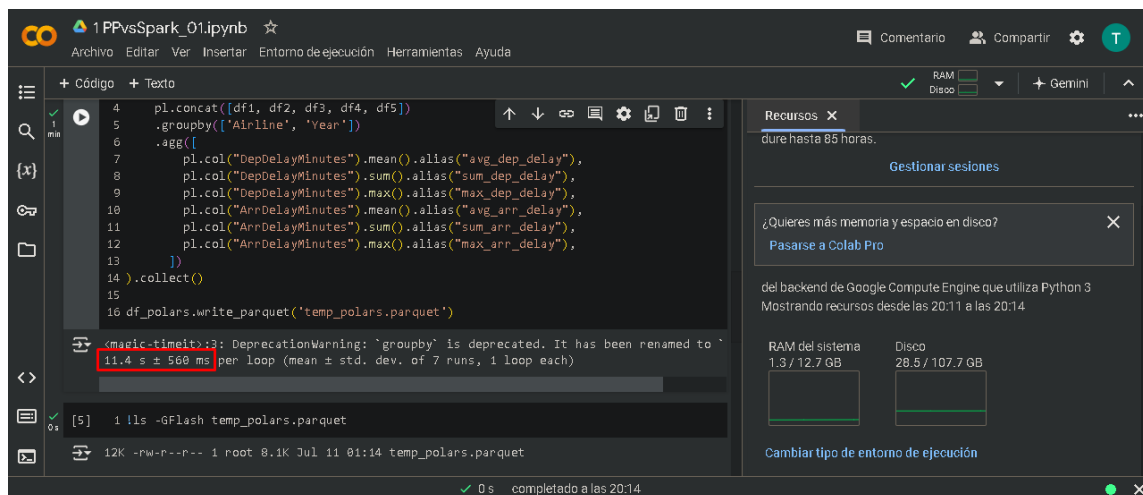
1 # df = pd.concat([df1, df2, df3, df4, df5])
2 df = df3

[ ] 1 %%timeit
2
3 df_agg = df.groupby(['Airline', 'Year'])['DepDelayMinutes', 'ArrDelayMinutes'].agg(['mean', 'sum', 'max'])
```

The output area shows a red error message: "Tu sesión ha fallado porque se ha usado toda la memoria RAM disponible." (Your session failed because all available RAM memory has been used).

The sidebar on the right shows system resources: RAM 1.3 / 12.7 GB and Disco 28.3 / 107.7 GB. It also includes a "Recursos" section with a warning about memory usage and a "Gestionar sesiones" button.

POLARS



The screenshot shows a Jupyter Notebook interface with a code editor on the left and a sidebar on the right. The code in the editor is as follows:

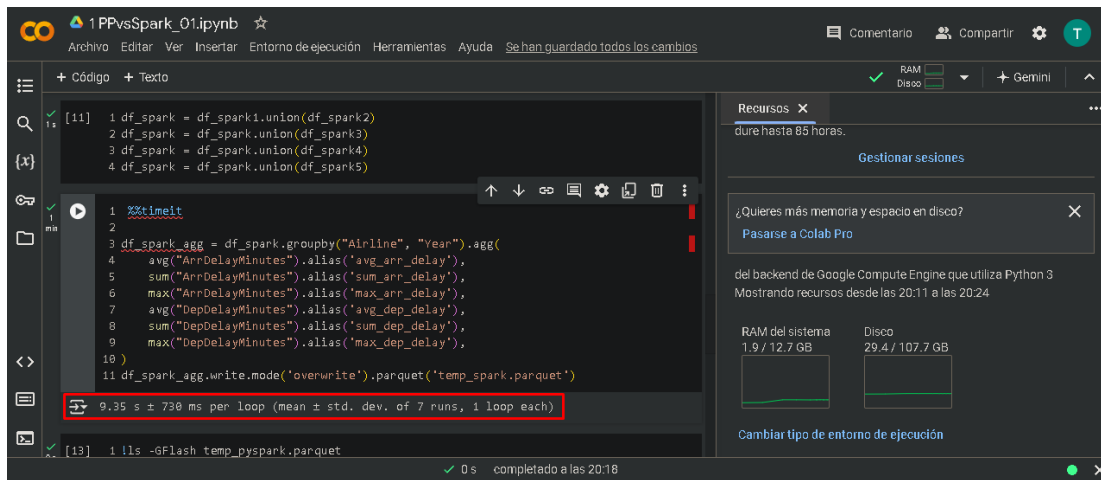
```
4 pl.concat([df1, df2, df3, df4, df5])
5 .groupby(['Airline', 'Year'])
6 .agg([
7     pl.col("DepDelayMinutes").mean().alias("avg_dep_delay"),
8     pl.col("DepDelayMinutes").sum().alias("sum_dep_delay"),
9     pl.col("DepDelayMinutes").max().alias("max_dep_delay"),
10    pl.col("ArrDelayMinutes").mean().alias("avg_arr_delay"),
11    pl.col("ArrDelayMinutes").sum().alias("sum_arr_delay"),
12    pl.col("ArrDelayMinutes").max().alias("max_arr_delay"),
13 ])
14 ).collect()
15
16 df_polars.write_parquet("temp_polars.parquet")

11.4 s ± 560 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

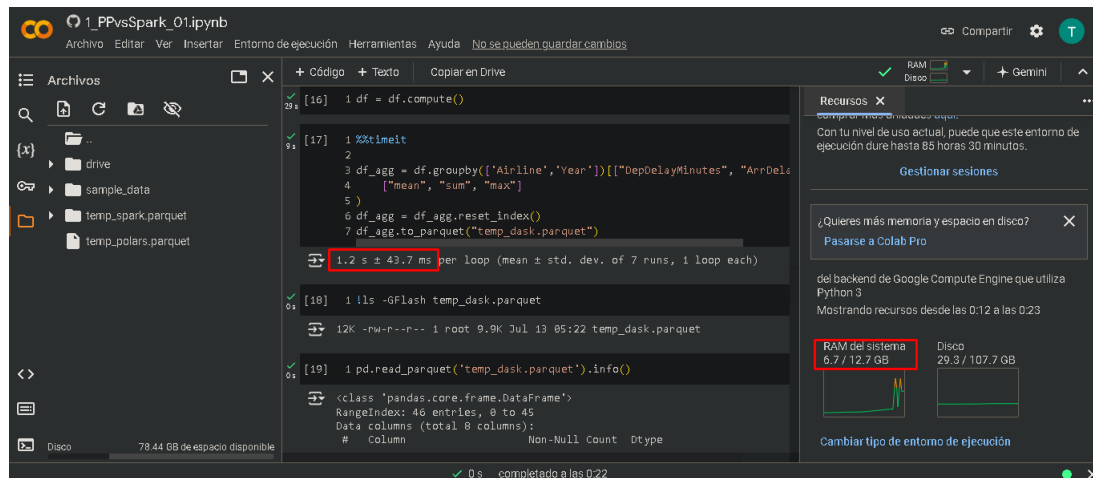
The output area shows the execution of the code, including a warning about the deprecated 'groupby' method and the final result of the aggregation.

The sidebar on the right shows system resources: RAM 1.3 / 12.7 GB and Disco 28.5 / 107.7 GB. It also includes a "Recursos" section with a warning about memory usage and a "Gestionar sesiones" button.

SPARK



DASK



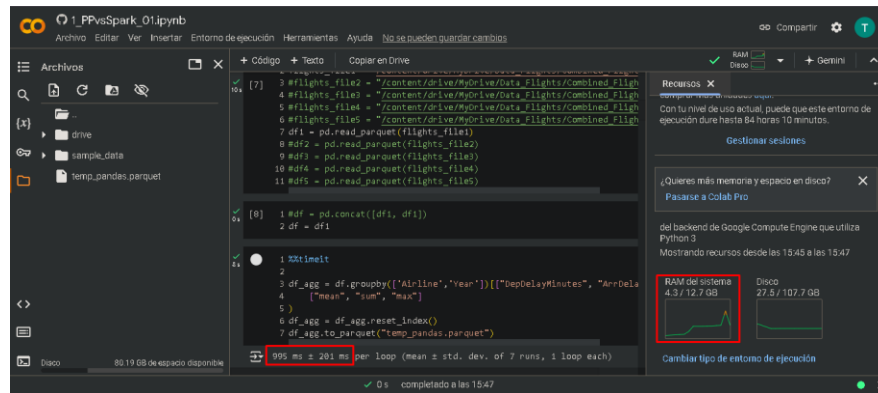
RESULTADOS

Con 5 archivos.

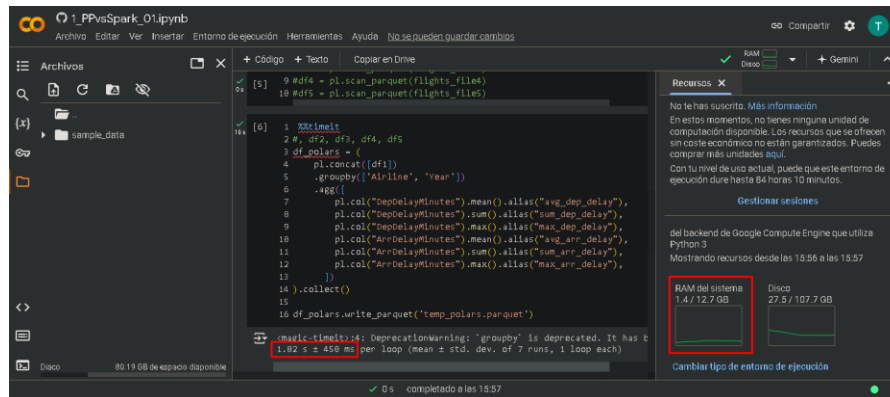
LIBRERIA	TIME	RAM
PANDAS	No ejecuta los 5 archivos.	No ejecuta los 5 archivos.
POLARS	11.7 s ± 570 ms	1.3 GB
SPARK	9.35 s ± 730 ms	1.9 GB
DASK	1.35 s ± 243 ms	6.7 GB

En la segunda prueba, ejecutaremos un solo archivo.

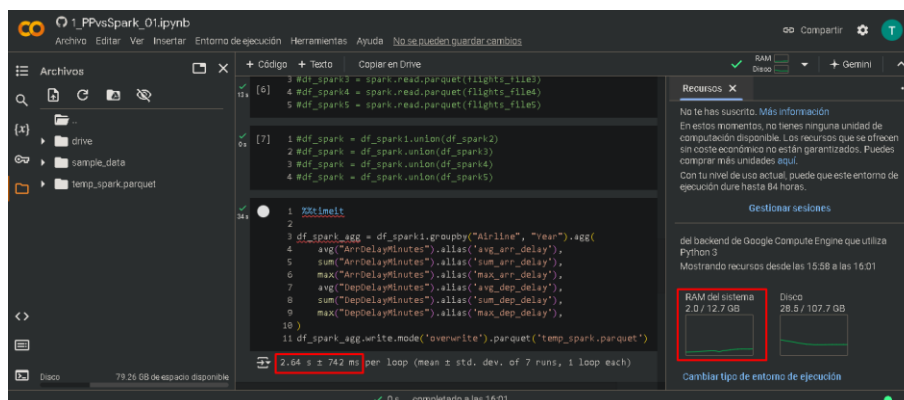
PANDAS



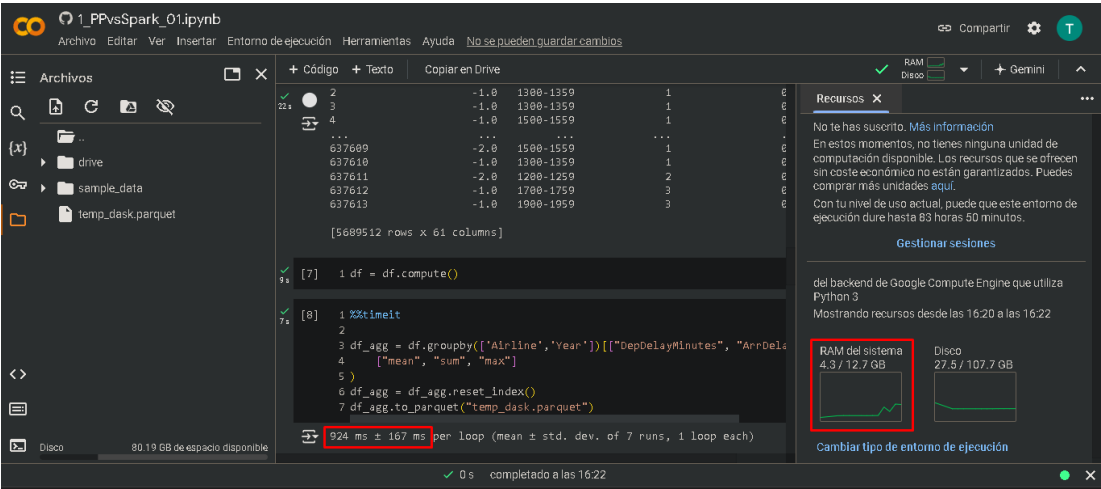
POLARS



SPARK



DASK



RESULTADOS

Con un archivo.

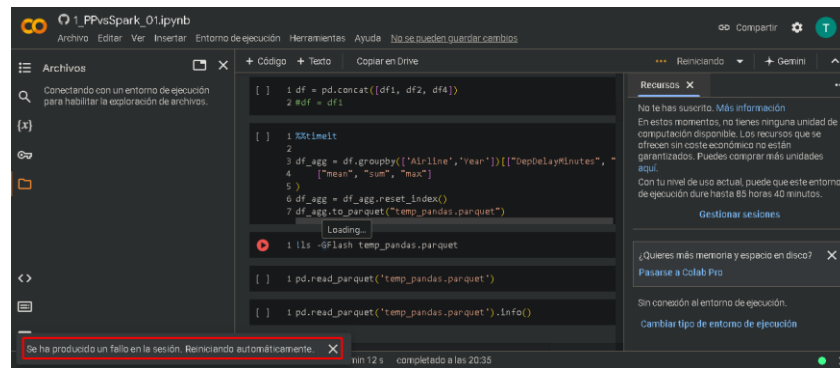
LIBRERIA	TIME	RAM
PANDAS	995 ms ± 201 ms	4.3 GB
POLARS	1.82 s ± 450 ms	1.4 GB
SPARK	2.64 s ± 742 ms	2.0 GB
DASK	924 ms ± 167 ms	4.3 GB

En la tercera prueba, subiremos los 3 archivos más pesados.

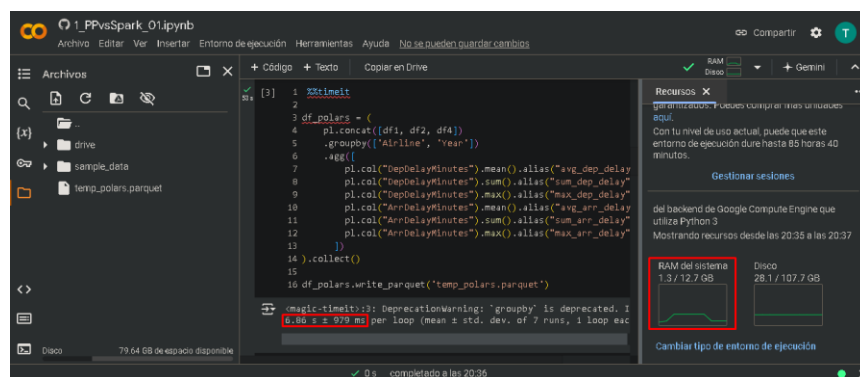
The screenshot shows a file explorer interface with a list of files in the 'Data_Flights' directory. The files are sorted by size, and the three largest files are highlighted with red boxes:

Nombre	Propietario	Última modificación	Tamaño de archivo
tablah.csv	yo	2 jul 2024	145 kB
Combined_Flights_2022.parquet	yo	2 jul 2024	142,7 MB
Combined_Flights_2021.parquet	yo	2 jul 2024	231,7 MB
Combined_Flights_2020.parquet	yo	2 jul 2024	174,6 MB
Combined_Flights_2019.parquet	yo	2 jul 2024	294,4 MB
Combined_Flights_2018.parquet	yo	2 jul 2024	215,3 MB

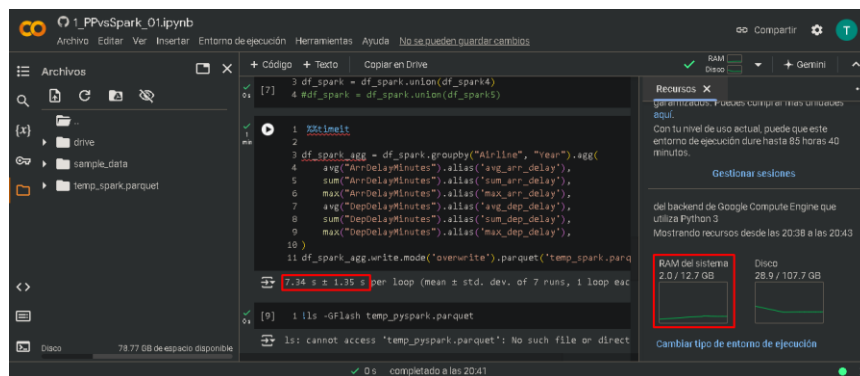
PANDAS



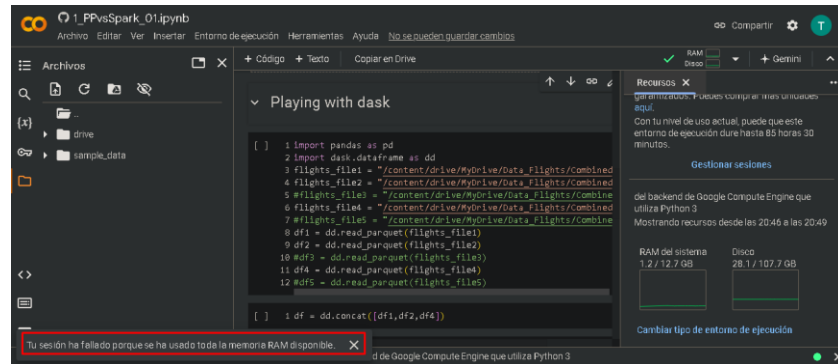
POLARS



SPARK



DASK



RESULTADOS

Con los tres archivos.

LIBRERIA	TIME	RAM
PANDAS	No ejecutó	No ejecutó
POLARS	6.86 s \pm 979 ms	1.3 GB
SPARK	7.34 s \pm 1.35 s	2.0 GB
DASK	No ejecutó	No ejecutó

CONCLUSIONES

- Manejo de grandes volúmenes de datos:
 - Pandas: No pudo procesar los cinco archivos ni los tres archivos más pesados, lo que indica limitaciones en el manejo de grandes volúmenes de datos.
 - Polars y Spark: Ambas librerías pudieron procesar los cinco archivos y los tres archivos más pesados. Sin embargo, Polars fue más eficiente en términos de uso de RAM (1.3 GB frente a 1.9 GB de Spark) y ligeramente más rápido en tiempo de procesamiento con cinco archivos (11.7 s frente a 9.35 s de Spark).
 - Dask: Fue el más rápido en la prueba con cinco archivos (1.35 s), pero consumió una cantidad significativa de RAM (6.7 GB). No pudo ejecutar los tres archivos más pesados.

- Eficiencia en el procesamiento de un solo archivo:
 - Pandas: Fue muy eficiente con un solo archivo (995 ms) pero consumió una cantidad considerable de RAM (4.3 GB).
 - Dask: Fue el más rápido (924 ms) y consumió la misma cantidad de RAM que Pandas (4.3 GB).
 - Polars y Spark: Polars (1.82 s, 1.4 GB) y Spark (2.64 s, 2.0 GB) fueron más lentos en comparación con Pandas y Dask, pero consumieron menos RAM.
- Consumo de RAM:
 - Polars: Se evidenció un uso de RAM más eficiente en comparación con las otras librerías, especialmente en las pruebas con múltiples archivos.
 - Dask: Aunque fue el más rápido en algunas pruebas, su alto consumo de RAM es una consideración importante, especialmente en entornos con recursos limitados.