

LABORATORIO N°2

LIBRERIAS PANDAS VS POLARS VS PYSPARK VS DASK

Andrés Felipe Esquivel Ruiz; Código 12967.

Nathaly Daniela Mejía Meléndez; Código 84588.

Universidad ECCI

Seminario Big Data

Elías Buitrago Bolívar

Julio de 2024

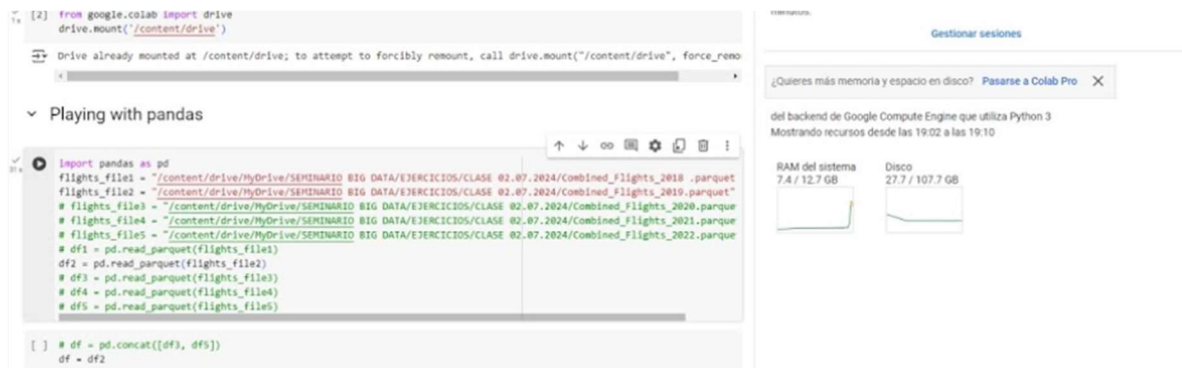
Descripción de la Actividad:

En el siguiente laboratorio realizamos la comparativa realizada entre las Librerías pandas vs polars vs pyspark vs dask Para lo cual utilizaremos la Data y códigos proporcionados por el ingeniero Elías Buitrago Bolívar, esto con el fin de estudiar y comparar diferentes herramientas para leer y manipular datos; para ser utilizado en la fase de comprensión de datos, con el fin de comprender el consumo de recursos utilizados por el servicio de procesamiento ofrecido por Google colab.

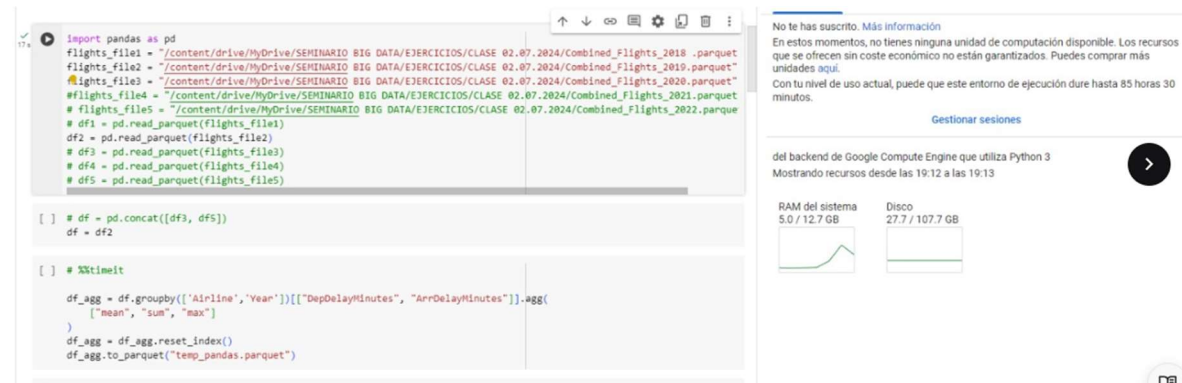
1. Jugando con Pandas:

Para la primera herramienta de lectura de datos utilizaremos la Librería PANDAS, en la cual corrimos los diferentes archivos de datos seccionados para evaluar el consumo de recursos, obteniendo los siguientes resultados:

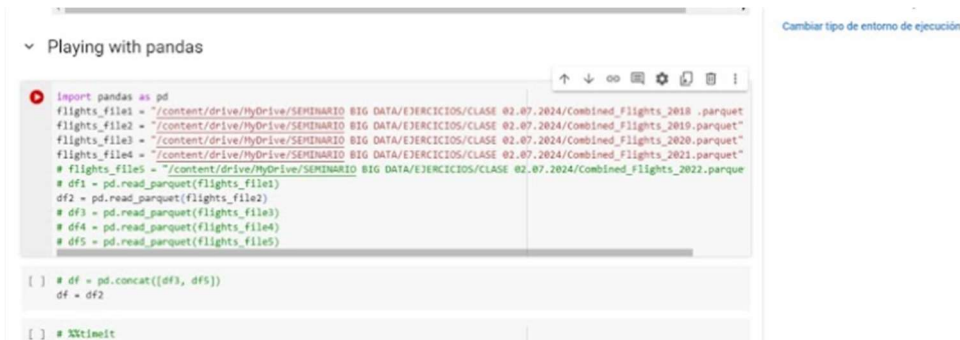
- Pandas con 2 archivos de datos al tiempo tuvo un consumo de 7.2 Gb de Ram de los recursos disponibles, luego de 31 segundos de Procesamiento el programa corrió satisfactoriamente.



- Pandas con 3 archivos de datos al tiempo tuvo un consumo de 7.4 Gb de Ram de los recursos disponibles, luego de 17 segundos de Procesamiento el programa corrió satisfactoriamente.



- Pandas con 4 archivos de datos al tiempo tuvo un colapso por Ram y el entorno de ejecución sufrió un reinicio



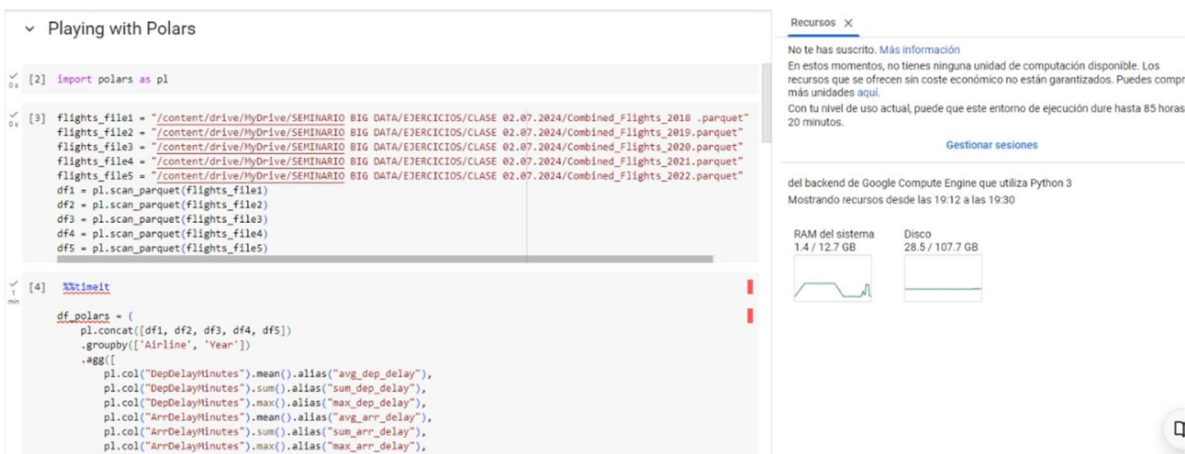
```
import pandas as pd
flights_file1 = "/content/drive/MyDrive/SEMINARIO BIG DATA/EJERCICIOS/CLASE 02.07.2024/Combined_Flights_2018.parquet"
flights_file2 = "/content/drive/MyDrive/SEMINARIO BIG DATA/EJERCICIOS/CLASE 02.07.2024/Combined_Flights_2019.parquet"
flights_file3 = "/content/drive/MyDrive/SEMINARIO BIG DATA/EJERCICIOS/CLASE 02.07.2024/Combined_Flights_2020.parquet"
flights_file4 = "/content/drive/MyDrive/SEMINARIO BIG DATA/EJERCICIOS/CLASE 02.07.2024/Combined_Flights_2021.parquet"
flights_file5 = "/content/drive/MyDrive/SEMINARIO BIG DATA/EJERCICIOS/CLASE 02.07.2024/Combined_Flights_2022.parquet"
df1 = pd.read_parquet(flights_file1)
df2 = pd.read_parquet(flights_file2)
df3 = pd.read_parquet(flights_file3)
df4 = pd.read_parquet(flights_file4)
df5 = pd.read_parquet(flights_file5)

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

%timeit
```

2. Jugando con Polars:

Para la segunda herramienta de lectura de datos utilizaremos la Librería POLARS, en la cual corrimos los 4 archivos de manera simultánea obteniendo un consumo de 4.7 Gb de Ram luego de 60 segundos de Procesamiento el programa corrió satisfactoriamente.



```
import polars as pl

flights_file1 = "/content/drive/MyDrive/SEMINARIO BIG DATA/EJERCICIOS/CLASE 02.07.2024/Combined_Flights_2018.parquet"
flights_file2 = "/content/drive/MyDrive/SEMINARIO BIG DATA/EJERCICIOS/CLASE 02.07.2024/Combined_Flights_2019.parquet"
flights_file3 = "/content/drive/MyDrive/SEMINARIO BIG DATA/EJERCICIOS/CLASE 02.07.2024/Combined_Flights_2020.parquet"
flights_file4 = "/content/drive/MyDrive/SEMINARIO BIG DATA/EJERCICIOS/CLASE 02.07.2024/Combined_Flights_2021.parquet"
flights_file5 = "/content/drive/MyDrive/SEMINARIO BIG DATA/EJERCICIOS/CLASE 02.07.2024/Combined_Flights_2022.parquet"
df1 = pl.scan_parquet(flights_file1)
df2 = pl.scan_parquet(flights_file2)
df3 = pl.scan_parquet(flights_file3)
df4 = pl.scan_parquet(flights_file4)
df5 = pl.scan_parquet(flights_file5)

df_polars = (
    pl.concat([df1, df2, df3, df4, df5])
    .groupby(['Airline', 'Year'])
    .agg([
        pl.col("DepDelayMinutes").mean().alias("avg_dep_delay"),
        pl.col("DepDelayMinutes").sum().alias("sum_dep_delay"),
        pl.col("DepDelayMinutes").max().alias("max_dep_delay"),
        pl.col("ArrDelayMinutes").mean().alias("avg_arr_delay"),
        pl.col("ArrDelayMinutes").sum().alias("sum_arr_delay"),
        pl.col("ArrDelayMinutes").max().alias("max_arr_delay"),
    ])
)
```

3. Jugando con PySpark:

Para la tercera herramienta de lectura de datos utilizaremos la Librería PySPARK, en la cual corrimos los 4 archivos de manera simultánea obteniendo un consumo mínimo de 1.7 Gb de Ram luego de 11 segundos de Procesamiento el programa corrió satisfactoriamente.

```
[10] spark = SparkSession.builder.master("local[1]").appName("airline-example").getOrCreate()

[11] flights_file1 = "/content/drive/MyDrive/SEMINARIO BIG DATA/PROFESOR ELIAS/EJERCICIOS/Pandas vs Pc
flights_file2 = "/content/drive/MyDrive/SEMINARIO BIG DATA/PROFESOR ELIAS/EJERCICIOS/Pandas vs Pc
flights_file3 = "/content/drive/MyDrive/SEMINARIO BIG DATA/PROFESOR ELIAS/EJERCICIOS/Pandas vs Pc
flights_file4 = "/content/drive/MyDrive/SEMINARIO BIG DATA/PROFESOR ELIAS/EJERCICIOS/Pandas vs Pc
flights_file5 = "/content/drive/MyDrive/SEMINARIO BIG DATA/PROFESOR ELIAS/EJERCICIOS/Pandas vs Pc

[12] df_spark1 = spark.read.parquet(flights_file1)
df_spark2 = spark.read.parquet(flights_file2)
df_spark3 = spark.read.parquet(flights_file3)
df_spark4 = spark.read.parquet(flights_file4)
df_spark5 = spark.read.parquet(flights_file5)

df_spark = df_spark1.union(df_spark2)
df_spark = df_spark.union(df_spark3)
df_spark = df_spark.union(df_spark4)
df_spark = df_spark.union(df_spark5)

[ ] %timeit
```

No tienes una suscripción. [Más información](#)
En este momento no tienes unidades de procesamiento disponibles. Los recursos que se ofrecen sin costo no están garantizados. Compra más unidades [aquí](#).
En tu nivel de uso actual, este tiempo de ejecución puede durar hasta 85 horas 20 minutos.

[Administrar sesiones](#)

¿Quieres más memoria y espacio en el disco? [Actualizar a Colab Pro](#)

del backend de Google Compute Engine en Python 3
Se muestran los recursos desde el 11:09 p.m. hasta el 11:39 p.m.

RAM de sistema 1.7 / 12.7 GB Disco 28.2 / 107.7 GB

[Cambiar tipo de entorno de ejecución](#)

4. Jugando con Dask:

Para la tercera herramienta de lectura de datos utilizaremos la Librería DASK, en la cual corrimos los 4 archivos de manera simultánea obteniendo un consumo máximo de 11.7 Gb de Ram luego de 23 segundos de Procesamiento el programa corrió satisfactoriamente.

```
Playing with dask

[21] import pandas as pd
import dask.dataframe as dd
flights_file1 = "/content/drive/MyDrive/SEMINARIO BIG DATA/PROFESOR ELIAS/EJERCICIOS/Pandas vs Pc
flights_file2 = "/content/drive/MyDrive/SEMINARIO BIG DATA/PROFESOR ELIAS/EJERCICIOS/Pandas vs Pc
flights_file3 = "/content/drive/MyDrive/SEMINARIO BIG DATA/PROFESOR ELIAS/EJERCICIOS/Pandas vs Pc
flights_file4 = "/content/drive/MyDrive/SEMINARIO BIG DATA/PROFESOR ELIAS/EJERCICIOS/Pandas vs Pc
flights_file5 = "/content/drive/MyDrive/SEMINARIO BIG DATA/PROFESOR ELIAS/EJERCICIOS/Pandas vs Pc
df1 = dd.read_parquet(flights_file1)
df2 = dd.read_parquet(flights_file2)
df3 = dd.read_parquet(flights_file3)
df4 = dd.read_parquet(flights_file4)
df5 = dd.read_parquet(flights_file5)

[22] df = dd.concat([df3, df5])

[23] print(df.compute())
```

No tienes una suscripción. [Más información](#)
En este momento no tienes unidades de procesamiento disponibles. Los recursos que se ofrecen sin costo no están garantizados. Compra más unidades [aquí](#).
En tu nivel de uso actual, este tiempo de ejecución puede durar hasta 85 horas 10 minutos.

[Administrar sesiones](#)

¿Quieres más memoria y espacio en el disco? [Actualizar a Colab Pro](#)

del backend de Google Compute Engine en Python 3
Se muestran los recursos desde el 11:47 p.m. hasta el 11:52 p.m.

RAM de sistema 6.9 / 12.7 GB Disco 29.2 / 107.7 GB

[Cambiar tipo de entorno de ejecución](#)

	FlightDate	Airline	Origin	Dest	Cancelled	Diverted	\
0	2020-09-01	Comair Inc.	PHL	DAY	False	False	

CONCLUSION:

Como conclusión podemos decir que la elección de la Biblioteca depende de las Necesidades específicas del proyecto, El tamaño de los datos y la capacidad de procesamiento disponible en el caso de Colab contamos con las 12Gb de Ram que ofrece en su versión Free, Cada una de las bibliotecas cuenta con sus fortalezas y debilidades y ya dependerá según los requisitos de análisis en cuestión, adicionalmente podemos concluir que PySpark y Dask están diseñados para escalar a grandes volúmenes de datos y múltiples nodos, lo cual es ideal para operaciones con grandes conjuntos de datos. Pandas y Polars están limitados a la capacidad de una sola máquina y su potencia de trabajo.