

LABORATORIO 2

Integrantes: Cristhian Balaguera - Daniel Gordillo

LABORATORIO 2

SEMINARIO DE BIG DATA Y CIENCIAS DE DATOS

CRISTHIAN CAMILO BALAGUERA SOTO - 88913
DANIEL MAURICIO GORDILLO ALVAREZ - 89400

PROFESOR

ELIAS BUITRAGO BOLIVAR

UNIVERSIDAD ECCI

BOGOTÁ D.C., COLOMBIA

2024

Laboratorio 2**Introducción**

Este laboratorio tiene como objetivo evaluar la eficiencia de diversas librerías de Python para el procesamiento de las librerías: Pandas, Polars, Spark y Dask. El propósito es determinar cuál de estas librerías puede manejar la mayor cantidad de datos en Google Colaboratory sin que la memoria RAM se sature y provoque el reinicio del entorno. Se experimentaran con cada librería y el conjunto de datos proporcionado, identificando cuál es la más eficiente en términos de uso de memoria RAM y capacidad de procesamiento de archivos.

Método Pandas	# de prueba	Datos utilizados	Resultados
	Prueba 1	flights_file1 flights_file2 df1 df2	42 segundos Limite de RAM alcanzado en el paso 3
	Prueba 2	flights_file1 flights_file3 df1 df3	43 segundos Proceso ejecutado con éxito RAM : 11.4 Gb Disco : 28.1 Gb
	Prueba 3	flights_file2 flights_file4 df2 df4	47 segundos Limite alcanzado en el paso 2
	Prueba 4	flights_file2 flights_file5 df2 df5	43 segundos Limite alcanzado en el paso 3

Prueba 1

```
[ ] from google.colab import drive
drive.mount("/content/drive")

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

v Playing with pandas

[ ] import pandas as pd
flights_file1 = "/content/drive/MyDrive/data/flights/Combined_Flights_2018.parquet"
flights_file2 = "/content/drive/MyDrive/data/flights/Combined_Flights_2019.parquet"
flights_file3 = "/content/drive/MyDrive/data/flights/Combined_Flights_2020.parquet"
flights_file4 = "/content/drive/MyDrive/data/flights/Combined_Flights_2021.parquet"
flights_file5 = "/content/drive/MyDrive/data/flights/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])
df = df1

[ ] # %timeit
df_agg = df.groupby(['Airline', 'Year'])[['DepDelayMinutes', 'ArrDelayMinutes']].agg(
    ["mean", "sum", "max"]
)
df_agg = df_agg.reset_index()
df_agg.to_parquet("temp_pandas.parquet")

NameError                                Traceback (most recent call last)
<ipython-input-1-98409a323802> in <cell line: 3>()
      1 # %timeit
      2
----> 3 df_agg = df.groupby(['Airline', 'Year'])[['DepDelayMinutes', 'ArrDelayMinutes']].agg(
      4     "mean", "sum", "max"]
      5 )
```

Se ha producido un fallo en la sesión. Reiniciando automáticamente.

15 s completado a las 19:28

Prueba 2

```
[ ] pd.read_parquet("temp_pandas.parquet").info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53 entries, 0 to 52
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype
---  ---
 0   (Airline, )         53 non-null    object
 1   (Year, )            53 non-null    int64
 2   (DepDelayMinutes, mean) 53 non-null    float64
 3   (DepDelayMinutes, sum)  53 non-null    float64
 4   (DepDelayMinutes, max)  53 non-null    float64
 5   (ArrDelayMinutes, mean) 53 non-null    float64
 6   (ArrDelayMinutes, sum)  53 non-null    float64
 7   (ArrDelayMinutes, max)  53 non-null    float64
dtypes: float64(6), int64(1), object(1)
memory usage: 3.4k B
```

	Airline	Year	DepDelayMinutes (mean)	DepDelayMinutes (sum)	DepDelayMinutes (max)	ArrDelayMinutes (mean)	ArrDelayMinutes (sum)	ArrDelayMinutes (max)
38	Mesa Airlines Inc.	2020	9.662425	1236887.0	1890.0	10.344427	1320704.0	1884.0
39	Peninsula Airways Inc.	2018	14.184256	16397.0	470.0	15.569902	17485.0	437.0
40	Republic Airlines	2018	12.002701	2377663.0	1270.0	13.293485	2624772.0	1346.0
41	Republic Airlines	2020	5.224466	1093533.0	1302.0	5.824160	1216865.0	1381.0
42	SkyWest Airlines Inc.	2018	14.514365	7611841.0	2098.0	15.656154	8173859.0	2108.0
43	SkyWest Airlines Inc.	2020	8.671968	4963067.0	2327.0	8.718970	4976457.0	2308.0
44	Southwest Airlines Co.	2018	12.596239	16812158.0	713.0	11.119075	14806251.0	698.0
45	Southwest Airlines Co.	2020	4.116316	3637115.0	602.0	3.372148	2975631.0	597.0
46	Spirit Air Lines	2018	13.666850	2385002.0	1527.0	14.241442	2479848.0	1527.0
47	Spirit Air Lines	2020	7.809889	1032194.0	1339.0	7.758486	1023290.0	1262.0
48	Trans States Airlines	2018	21.379592	1389310.0	1519.0	22.335766	1444320.0	1537.0
49	Trans States Airlines	2020	13.739364	235108.0	1390.0	14.784864	252008.0	1366.0
50	United Air Lines Inc.	2018	13.614583	8398591.0	1431.0	14.477765	8902595.0	1429.0
51	United Air Lines Inc.	2020	6.925499	1978518.0	1471.0	6.766335	1929514.0	1503.0
52	Virgin America	2018	10.902274	187977.0	520.0	11.950971	204995.0	504.0

Prueba 3

1_PPvsSpark_01ipynb

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Se han guardado todos los cambios

Reiniciando + Gemini

Archivos Conectando con un entorno de ejecución para habilitar la exploración de archivos.

Playing with pandas

```
[ ] import pandas as pd
#flights_file1 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2018.parquet"
#flights_file2 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2019.parquet"
#flights_file3 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2020.parquet"
#flights_file4 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2021.parquet"
#flights_file5 = "/content/drive/MyDrive/data/Flights/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([df2,df4])
# df = df1

# %%timeit
df_agg = df.groupby(['Airline','Year'])[['DepDelayMinutes', 'ArrDelayMinutes']].agg(
    ["mean", "sum", "max"]
)
df_agg = df_agg.reset_index()
df_agg.to_parquet("temp_pandas.parquet")

[ ] !ls -l temp_pandas.parquet
12K -rw-r--r-- 1 root 11K Jun 20 00:35 temp_pandas.parquet

pd.read_parquet('temp_pandas.parquet')
```

	Airline	Year	DepDelayMinutes	ArrDelayMinutes				
			mean	sum	max	mean	sum	max
0	Air Wisconsin Airlines Corp	2018	16.753459	1606774.0	1296.0	17.881934	1708887.0	1292.0
1	Air Wisconsin Airlines Corp	2020	8.583725	433315.0	1460.0	8.982529	452450.0	1439.0
	Alaska Airlines Inc.	2018	7.503389	1374801.0	839.0	8.759125	1600336.0	842.0

Se ha producido un fallo en la sesión. Reiniciando automáticamente.

12 s completado a las 19:43

Recursos

No te has suscrito. Más información

En estos momentos, no tienes ninguna unidad de computación disponible. Los recursos que se ofrecen sin coste económico no están garantizados. Puedes comprar más unidades aquí.

Con tu nivel de uso actual, puede que este entorno de ejecución dure hasta 42 horas 10 minutos.

Gestionar sesiones

¿Quieres más memoria y espacio en disco?

Pasarse a Colab Pro

Sin conexión al entorno de ejecución.

Cambiar tipo de entorno de ejecución

Prueba 4

1_PPvsSpark_01ipynb

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda

Reiniciando + Gemini

Archivos Conectando con un entorno de ejecución para habilitar la exploración de archivos.

Playing with pandas

```
[ ] Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

[ ] import pandas as pd
#flights_file1 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2018.parquet"
#flights_file2 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2019.parquet"
#flights_file3 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2020.parquet"
#flights_file4 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2021.parquet"
#flights_file5 = "/content/drive/MyDrive/data/Flights/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([df2,df5])
# df = df1

# %%timeit
df_agg = df.groupby(['Airline','Year'])[['DepDelayMinutes', 'ArrDelayMinutes']].agg(
    ["mean", "sum", "max"]
)
df_agg = df_agg.reset_index()
df_agg.to_parquet("temp_pandas.parquet")

[ ] !ls -l temp_pandas.parquet
12K -rw-r--r-- 1 root 11K Jun 20 00:35 temp_pandas.parquet

pd.read_parquet('temp_pandas.parquet')
```

	Airline	Year	DepDelayMinutes	ArrDelayMinutes				
			mean	sum	max	mean	sum	max
	Air Wisconsin Airlines Corp	2018	16.753459	1606774.0	1296.0	17.881934	1708887.0	1292.0

Se ha producido un fallo en la sesión. Reiniciando automáticamente.

4 s completado a las 19:48

Recursos

No te has suscrito. Más información

En estos momentos, no tienes ninguna unidad de computación disponible. Los recursos que se ofrecen sin coste económico no están garantizados. Puedes comprar más unidades aquí.

Con tu nivel de uso actual, puede que este entorno de ejecución dure hasta 42 horas.

Gestionar sesiones

¿Quieres más memoria y espacio en disco?

Pasarse a Colab Pro

Sin conexión al entorno de ejecución.

Cambiar tipo de entorno de ejecución

Método Polars	# de prueba	Datos utilizados	Resultados
	Prueba 1	flights_file1 flights_file3 df1 df3	42 segundos Proceso ejecutado con éxito RAM: 1.4 Gb Disco: 28.5 Gb
	Prueba 2	flights_file2 flights_file3 df2 df3	44 segundos Proceso ejecutado con éxito RAM : 1.5 Gb Disco : 28.5 Gb
	Prueba 3	flights_file1 flights_file4 df1 df4	30 segundos Proceso ejecutado con éxito. RAM:1.5 Gb Disco: 28.5 Gb
	Prueba 4	flights_file3 flights_file5 df3 df5	24 segundos Proceso ejecutado con éxito. RAM:1.5 Gb Disco: 28.5 Gb

Prueba 1

Prueba 2

1_PPvsSpark_01.ipynb

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda

Archivos

drive

sample_data

temp_pandas.parquet

temp_polars.parquet

[1] import polars as pl

[2] flights_file1 = "/content/drive/MyDrive/data/flights/Combined_Flights_2018.parquet"
#flights_file2 = "/content/drive/MyDrive/data/flights/Combined_Flights_2019.parquet"
flights_file3 = "/content/drive/MyDrive/data/flights/Combined_Flights_2020.parquet"
#flights_file4 = "/content/drive/MyDrive/data/flights/Combined_Flights_2021.parquet"
#flights_files = "/content/drive/MyDrive/data/flights/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_files)

[5] %timeit
df_polars = (
 pl.concat([df1, df3])
 .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"),
])
).collect()

df_polars.write_parquet('temp_polars.parquet')

<magic-timeit>:3: DeprecationWarning: 'groupby' is deprecated. It has been renamed to 'group_by'.
3.56 s ± 780 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

[6] !ls -l temp_polars.parquet

8.0K -rw-r--r-- 1 root 5.7K Jun 20 00:55 temp_polars.parquet

RAM
Disco

Recursos

No te has suscrito. Más información
En estos momentos, no tienes ninguna unidad de computación disponible. Los recursos que se ofrecen sin coste económico no están garantizados. Puedes comprar más unidades aquí.
Con tu nivel de uso actual, puede que este entorno de ejecución dure hasta 42 horas.
Gestionar sesiones

¿Quieres más memoria y espacio en disco?
Pasarse a Colab Pro

del backend de Google Compute Engine que utiliza Python 3
Mostrando recursos desde las 19:48 a las 19:56

RAM del sistema
1.4 / 12.7 GB

Disco
28.5 / 107.7 GB

Cambiar tipo de entorno de ejecución

Disco 79.24 GB de espacio disponible

Playing with PySpark

0 s completado a las 19:55

LABORATORIO 2

Integrantes: Cristhian Balaguera - Daniel Gordillo

Archivos

drive

sample_data

temp_pandas.parquet

temp_polars.parquet

[1] import polars as pl

#flights_file1 = "/content/drive/MyDrive/data/flights/Combined_Flights_2018.parquet"

#flights_file2 = "/content/drive/MyDrive/data/flights/Combined_Flights_2019.parquet"

#flights_file3 = "/content/drive/MyDrive/data/flights/Combined_Flights_2020.parquet"

#flights_file4 = "/content/drive/MyDrive/data/flights/Combined_Flights_2021.parquet"

#flights_files = "/content/drive/MyDrive/data/flights/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_files)

%timeit

df_polars = (
 pl.concat([df2, df3])
 .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"),
])
 .collect()
)
df_polars.write_parquet('temp_polars.parquet')

%magic-timelt>:3: DeprecationWarning: 'groupby' is deprecated. It has been renamed to 'group_by'.
5.15 s ± 2.29 s per loop (mean ± std. dev. of 7 runs, 1 loop each)

!ls -l temp_polars.parquet

8.0K -rw-r--r-- 1 root 5.6K Jun 20 01:19 temp_polars.parquet

Playing with PySpark

RAM
Disco

+ Gemini

Recursos

No te has suscrito. Más información

En estos momentos, no tienes ninguna unidad de computación disponible. Los recursos que se ofrecen sin coste económico no están garantizados. Puedes comprar más unidades aquí.

Con tu nivel de uso actual, puede que este entorno de ejecución dure hasta 83 horas 40 minutos.

Gestionar sesiones

¿Quieres más memoria y espacio en disco?

Pasarse a Colab Pro

del backend de Google Compute Engine que utiliza Python 3

Mostrando recursos desde las 19:48 a las 20:19

RAM del sistema
1.5 / 12.7 GB

Disco
28.5 / 107.7 GB

Cambiar tipo de entorno de ejecución

Prueba 3

Archivos

drive

sample_data

temp_pandas.parquet

temp_polars.parquet

[1] import polars as pl

#flights_file1 = "/content/drive/MyDrive/data/flights/Combined_Flights_2018.parquet"

#flights_file2 = "/content/drive/MyDrive/data/flights/Combined_Flights_2019.parquet"

#flights_file3 = "/content/drive/MyDrive/data/flights/Combined_Flights_2020.parquet"

#flights_file4 = "/content/drive/MyDrive/data/flights/Combined_Flights_2021.parquet"

#flights_files = "/content/drive/MyDrive/data/flights/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_files)

%timeit

df_polars = (
 pl.concat([df1, df4])
 .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"),
])
 .collect()
)
df_polars.write_parquet('temp_polars.parquet')

%magic-timelt>:3: DeprecationWarning: 'groupby' is deprecated. It has been renamed to 'group_by'.
3.79 s ± 642 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

!ls -l temp_polars.parquet

8.0K -rw-r--r-- 1 root 5.6K Jun 20 01:25 temp_polars.parquet

Playing with Polars

RAM
Disco

+ Gemini

Recursos

No te has suscrito. Más información

En estos momentos, no tienes ninguna unidad de computación disponible. Los recursos que se ofrecen sin coste económico no están garantizados. Puedes comprar más unidades aquí.

Con tu nivel de uso actual, puede que este entorno de ejecución dure hasta 83 horas 40 minutos.

Gestionar sesiones

¿Quieres más memoria y espacio en disco?

Pasarse a Colab Pro

del backend de Google Compute Engine que utiliza Python 3

Mostrando recursos desde las 19:48 a las 20:25

RAM del sistema
1.5 / 12.7 GB

Disco
28.5 / 107.7 GB

Cambiar tipo de entorno de ejecución

Prueba 4

Archivos

drive

sample_data

temp_pandas.parquet

temp_polars.parquet

[1] import polars as pl

#flights_file1 = "/content/drive/MyDrive/data/flights/Combined_Flights_2018.parquet"

#flights_file2 = "/content/drive/MyDrive/data/flights/Combined_Flights_2019.parquet"

#flights_file3 = "/content/drive/MyDrive/data/flights/Combined_Flights_2020.parquet"

#flights_file4 = "/content/drive/MyDrive/data/flights/Combined_Flights_2021.parquet"

#flights_files = "/content/drive/MyDrive/data/flights/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_files)

%timeit

df_polars = (
 pl.concat([df3, 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"),
])
 .collect()
)
df_polars.write_parquet('temp_polars.parquet')

%magic-timelt>:3: DeprecationWarning: 'groupby' is deprecated. It has been renamed to 'group_by'.
2.82 s ± 573 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

!ls -l temp_polars.parquet

8.0K -rw-r--r-- 1 root 5.6K Jun 20 01:25 temp_polars.parquet

Playing with Polars

RAM
Disco

+ Gemini

Recursos

No te has suscrito. Más información

En estos momentos, no tienes ninguna unidad de computación disponible. Los recursos que se ofrecen sin coste económico no están garantizados. Puedes comprar más unidades aquí.

Con tu nivel de uso actual, puede que este entorno de ejecución dure hasta 83 horas 30 minutos.

Gestionar sesiones

¿Quieres más memoria y espacio en disco?

Pasarse a Colab Pro

del backend de Google Compute Engine que utiliza Python 3

Mostrando recursos desde las 19:48 a las 20:27

RAM del sistema
1.5 / 12.7 GB

Disco
28.5 / 107.7 GB

Cambiar tipo de entorno de ejecución

Método PySpark

Método PySpark	# de prueba	Datos utilizados	Resultados
	Prueba 1	<code>flights_file3</code> <code>flights_file4</code> <code>df3</code> <code>df4</code>	46 segundos Proceso ejecutado con éxito RAM: 2.0Gb Disco: 29.4 Gb
	Prueba 2	<code>flights_file4</code> <code>flights_file5</code> <code>df4</code> <code>df5</code>	29 segundos Proceso ejecutado con éxito RAM : 2.3 Gb Disco : 29.4 Gb
	Prueba 3	<code>flights_file1</code> <code>flights_file5</code> <code>df1</code> <code>df5</code>	26 segundos Proceso ejecutado con éxito RAM : 2.3 Gb Disco : 29.4 Gb
	Prueba 4	<code>flights_file2</code> <code>flights_file3</code> <code>df2</code> <code>df3</code>	33 segundos Proceso ejecutado con éxito RAM : 2.3 Gb Disco : 29.4 Gb

Prueba 1

The screenshot displays a Google Cloud Shell interface with three main panes:

- Left Pane (File Explorer):** Shows the file system structure. A directory named "temp_spark.parquet" has been created under the home directory.
- Middle Pane (Code Editor):** Contains two files:
 - `install.py`: A script to install PySpark. It uses `curl` to download the PySpark binary package from the official website and installs it locally.
 - `flight.py`: A PySpark application that reads flight data from Parquet files, groups it by airline and year, calculates average arrival and departure delays, and writes the results back to a Parquet file.
- Right Pane (Terminal):** Shows the execution output.
 - The first command (`python install.py`) successfully installs PySpark 3.5.1.
 - The second command (`python flight.py`) runs the PySpark application. The output shows the total number of records (24K) and the distribution of delay statistics across different airlines and years.

The bottom status bar indicates that the terminal session is completed at 10:43 AM.

Prueba 2

Archivos

drive

sample_data

temp_spark.parquet

temp_pandas.parquet

temp_polars.parquet

Playing with PySpark

```
[17] !pip install pyspark

Collecting pyspark
  Using cached pyspark-3.5.1.tar.gz (317.0 MB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: pyjy<0.10.0, >0.10.0 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.0)
Building wheel for collected packages: pyspark
Building wheel for pyspark (setup.py) ... done
Created wheel for pyspark: filename=pyspark-3.5.1-py2.py3-none-any.whl size=317488491 sha256=ee85d3d38e610735c1c1c1ffcc90ff0e234ace216f00f2ff0b0403a8
Stored in directory: /root/.cache/pip/wheels/00/16/00/2c25e6d3866cc2f693be543214403e0270b07470b07f3a6
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.1

[36] from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, max, sum, concat

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

[38] #flights_file1 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2018.parquet"
#flights_file2 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2019.parquet"
#flights_file3 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2020.parquet"
#flights_file4 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2021.parquet"
#flights_file5 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2022.parquet"

[39] #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)

[39] #df_spark = df_spark1.union(df_spark2)
#df_spark = df_spark.union(df_spark3)
#df_spark = df_spark4.union(df_spark5)
#df_spark = df_spark.union(df_spark5)

[40] df_spark.write.mode("overwrite").parquet("temp_spark.parquet")

3.57 s ± 549 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

[41] !ls -l temp_spark.parquet

total 24K
4.0K drwxr-xr-x 2 root 4.0K Jun 20 01:47 ./
4.0K drwxr-xr-x 1 root 4.0K Jun 20 01:47 ../
8.0K -rw-r--r-- 1 root 4.0K Jun 20 01:47 part-00000-c58573ed-466d-49a1-9ffd-322134ace8e3-c000.snappy.parquet
4.0K -rw-r--r-- 1 root 4.0K Jun 20 01:47 part-00000-c58573ed-466d-49a1-9ffd-322134ace8e3-c000.snappy.parquet.crc
0 -rw-r--r-- 1 root 0 Jun 20 01:47 SUCCESS
4.0K -rw-r--r-- 1 root 0 Jun 20 01:47 SUCCESS.crc
```

Comentario

Compartir

Recursos

No te has suscrito. Más información

En estos momentos, no tienes ninguna unidad de computación disponible. Los recursos que se ofrecen sin coste económico no están garantizados. Puedes comprar más unidades aquí.

Con tu nivel de uso actual, puede que este entorno de ejecución dure hasta 83 horas 10 minutos.

Gestionar sesiones

¿Quieres más memoria y espacio en disco?

Pasarse a Colab Pro

del backend de Google Compute Engine que utiliza Python 3

Mostrando recursos desde las 20:40 a las 20:47

RAM del sistema

2.4 / 12.7 GB

Disco

29.4 / 107.7 GB

Cambiar tipo de entorno de ejecución

Playing with dask

```
[42] import pandas as pd

[43] from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, max, sum, concat

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

[44] #flights_file1 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2018.parquet"
#flights_file2 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2019.parquet"
#flights_file3 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2020.parquet"
#flights_file4 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2021.parquet"
#flights_file5 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2022.parquet"

[45] #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)

[46] #df_spark = df_spark1.union(df_spark2)
#df_spark = df_spark.union(df_spark3)
#df_spark = df_spark4.union(df_spark5)
#df_spark = df_spark.union(df_spark5)

[47] df_spark.write.mode("overwrite").parquet("temp_spark.parquet")

3.29 s ± 677 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

[48] !ls -l temp_spark.parquet

total 24K
4.0K drwxr-xr-x 2 root 4.0K Jun 20 01:50 ./
4.0K drwxr-xr-x 1 root 4.0K Jun 20 01:50 ../
8.0K -rw-r--r-- 1 root 4.0K Jun 20 01:50 part-00000-4b0768a0-e949-42c6-b14d-87e3afdc0596-c000.snappy.parquet
4.0K -rw-r--r-- 1 root 4.0K Jun 20 01:50 part-00000-4b0768a0-e949-42c6-b14d-87e3afdc0596-c000.snappy.parquet.crc
0 -rw-r--r-- 1 root 0 Jun 20 01:50 SUCCESS
4.0K -rw-r--r-- 1 root 0 Jun 20 01:50 SUCCESS.crc
```

Prueba 3

Archivos

drive

sample_data

temp_spark.parquet

temp_pandas.parquet

temp_polars.parquet

Playing with PySpark

```
[42] !pip install pyspark

Collecting pyspark
  Using cached pyspark-3.5.1.tar.gz (317.0 MB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: pyjy<0.10.0, >0.10.0 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.0)
Building wheel for collected packages: pyspark
Building wheel for pyspark (setup.py) ... done
Created wheel for pyspark: filename=pyspark-3.5.1-py2.py3-none-any.whl size=317488491 sha256=ee85d3d38e610735c1c1c1ffcc90ff0e234ace216f00f2ff0b0403a8
Stored in directory: /root/.cache/pip/wheels/00/16/00/2c25e6d3866cc2f693be543214403e0270b07470b07f3a6
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.1

[42] from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, max, sum, concat

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

[44] #flights_file1 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2018.parquet"
#flights_file2 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2019.parquet"
#flights_file3 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2020.parquet"
#flights_file4 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2021.parquet"
#flights_file5 = "/content/drive/MyDrive/data/Flights/Combined_Flights_2022.parquet"

[45] #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)

[46] #df_spark = df_spark1.union(df_spark2)
#df_spark = df_spark.union(df_spark3)
#df_spark = df_spark4.union(df_spark5)
#df_spark = df_spark.union(df_spark5)

[47] df_spark.write.mode("overwrite").parquet("temp_spark.parquet")

3.29 s ± 677 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

[48] !ls -l temp_spark.parquet

total 24K
4.0K drwxr-xr-x 2 root 4.0K Jun 20 01:50 ./
4.0K drwxr-xr-x 1 root 4.0K Jun 20 01:50 ../
8.0K -rw-r--r-- 1 root 4.0K Jun 20 01:50 part-00000-4b0768a0-e949-42c6-b14d-87e3afdc0596-c000.snappy.parquet
4.0K -rw-r--r-- 1 root 4.0K Jun 20 01:50 part-00000-4b0768a0-e949-42c6-b14d-87e3afdc0596-c000.snappy.parquet.crc
0 -rw-r--r-- 1 root 0 Jun 20 01:50 SUCCESS
4.0K -rw-r--r-- 1 root 0 Jun 20 01:50 SUCCESS.crc
```

Comentario

Compartir

Recursos

No te has suscrito. Más información

En estos momentos, no tienes ninguna unidad de computación disponible. Los recursos que se ofrecen sin coste económico no están garantizados. Puedes comprar más unidades aquí.

Con tu nivel de uso actual, puede que este entorno de ejecución dure hasta 83 horas 10 minutos.

Gestionar sesiones

¿Quieres más memoria y espacio en disco?

Pasarse a Colab Pro

del backend de Google Compute Engine que utiliza Python 3

Mostrando recursos desde las 20:40 a las 20:50

RAM del sistema

2.3 / 12.7 GB

Disco

29.4 / 107.7 GB

Cambiar tipo de entorno de ejecución

Playing with dask

```
[47] !ls -l temp_spark.parquet

total 24K
4.0K drwxr-xr-x 2 root 4.0K Jun 20 01:50 ./
4.0K drwxr-xr-x 1 root 4.0K Jun 20 01:50 ../
8.0K -rw-r--r-- 1 root 4.0K Jun 20 01:50 part-00000-4b0768a0-e949-42c6-b14d-87e3afdc0596-c000.snappy.parquet
4.0K -rw-r--r-- 1 root 4.0K Jun 20 01:50 part-00000-4b0768a0-e949-42c6-b14d-87e3afdc0596-c000.snappy.parquet.crc
0 -rw-r--r-- 1 root 0 Jun 20 01:50 SUCCESS
4.0K -rw-r--r-- 1 root 0 Jun 20 01:50 SUCCESS.crc
```


Prueba 4

Archivos

1_PPvsSpark_01.ipynb

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda

+

Código

+

Texto

[x]

drive

sample_data

temp_spark.parquet

temp_pandas.parquet

temp_polars.parquet

Playing with PySpark

!pip install pyspark

Collecting pyspark
Using cached pyspark-3.5.1.tar.gz (317.0 MB)
Preparing metadata (setup.py) ... done
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
Building wheels for collected packages: pyspark
Building wheel for pyspark (setup.py) ... done
Created wheel for pyspark: filename=pyspark-3.5.1-py2.py3-none-any.whl size=317488491 sha256=ee85d3d38eb187325c1c1ffecf90ff85e234ace2716f9b9f2effb0d403a8
Stored in directory: /root/.cache/pip/wheels/80/1d/60/2c256ed38ddce2fd93be54214a63e02fbd8d74f0b07f3a6
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.1

[49] from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, max, sum, concat

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

[51] #flights_file1 = "/content/drive/MyDrive/data/flights/combined_flights_2018.parquet"
#flights_file2 = "/content/drive/MyDrive/data/flights/combined_flights_2019.parquet"
#flights_file3 = "/content/drive/MyDrive/data/flights/combined_flights_2020.parquet"
#flights_file4 = "/content/drive/MyDrive/data/flights/combined_flights_2021.parquet"
#flights_files = "/content/drive/MyDrive/data/flights/combined_flights_2022.parquet"

[52] #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_files)

[53] #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)

✓ 0 s completado a las 20:52

[54] %timeit

df_spark_agg = df_spark.groupby("Airline", "Year").agg(
 avg("ArrDelayMinutes").alias('avg_arr_delay'),
 sum("ArrDelayMinutes").alias('sum_arr_delay'),
 max("ArrDelayMinutes").alias('max_arr_delay'),
 avg("DepDelayMinutes").alias('avg_dep_delay'),
 sum("DepDelayMinutes").alias('sum_dep_delay'),
 max("DepDelayMinutes").alias('max_dep_delay'),
)
df_spark_agg.write.mode('overwrite').parquet('temp_spark.parquet')

4.26 s ± 961 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

!ls -l temp_spark.parquet

total 24K
4.0K drwxr-xr-x 2 root 4.0K Jun 20 01:52 ./
4.0K drwxr-xr-x 1 root 4.0K Jun 20 01:52 ../
0.0K -rw-r--r-- 1 root 4.0K Jun 20 01:52 part-00000-f2b7c1c7-7f3d-4ece-b8f6-56963775eed0-c000.snappy.parquet
4.0K -rw-r--r-- 1 root 48 Jun 20 01:52 _part-00000-f2b7c1c7-7f3d-4ece-b8f6-56963775eed0-c000.snappy.parquet.crc
0 -rw-r--r-- 1 root 0 Jun 20 01:52 _SUCCESS
4.0K -rw-r--r-- 1 root 8 Jun 20 01:52 _SUCCESS.crc

Playing with dask

✓ 0 s completado a las 20:52

RAM del sistema
2.4 / 12.7 GB

Disco
29.4 / 107.7 GB

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

del backend de Google Compute Engine que utiliza Python 3
Mostrando recursos desde las 20:40 a las 20:53

Cambiar tipo de entorno de ejecución

Recursos X

No te has suscrito. Más información
En estos momentos, no tienes ninguna unidad de computación disponible. Los recursos que se ofrecen sin coste económico no están garantizados. Puedes comprar más unidades aquí.
Con tu nivel de uso actual, puede que este entorno de ejecución dure hasta 83 horas 10 minutos.

Gestionar sesiones

Método Dask	# de prueba	Datos utilizados	Resultados
	Prueba 1	flights_file2 flights_file4 df3 df4	29 segundos Limite de RAM alcanzado en el paso 3
	Prueba 2	flights_file1 flights_file5 df4 df5	77 segundos Proceso ejecutado con éxito RAM : 6.4 Gb Disco : 28.3 Gb
	Prueba 3	flights_file2 flights_file5 df2 df5	51 segundos Proceso ejecutado con éxito RAM : 6.5 Gb Disco : 28.3 Gb
	Prueba 4	flights_file1 flights_file3 df1 df3	65 segundos Proceso ejecutado con éxito RAM : 7.1 Gb Disco : 28.5 Gb

Prueba 1

```

import pandas as pd
import dask.dataframe as dd

# flights_file1 = "/content/drive/MyDrive/data/flights/Combined Flights 2018.parquet"
flights_file2 = "/content/drive/MyDrive/data/flights/Combined Flights 2019.parquet"
# flights_file3 = "/content/drive/MyDrive/data/flights/Combined Flights 2020.parquet"
flights_file4 = "/content/drive/MyDrive/data/flights/Combined Flights 2021.parquet"
# flights_file5 = "/content/drive/MyDrive/data/flights/Combined Flights 2022.parquet"
# 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)

[ ] df = dd.concat([df2, df4])

[ ] print(df.compute())

[ ] df = df.compute()

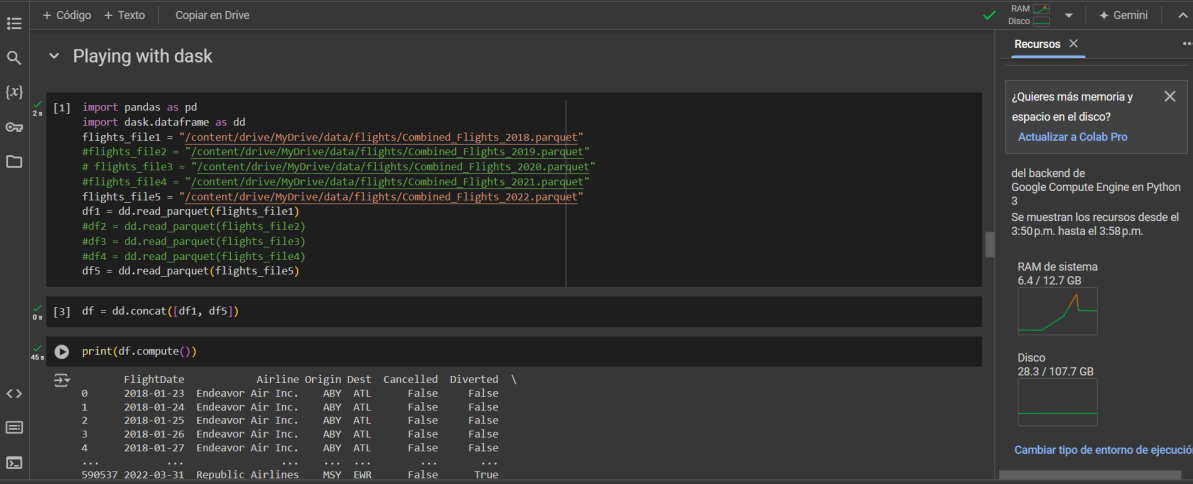
[ ] %%timeit
df_agg = df.groupby(['Airline', 'Year'])[['DepDelayMinutes', 'ArrDelayMinutes']].agg(
    ["mean", "sum", "max"]
)
df_agg = df_agg.reset_index()

```

La sesión falló. Reiniciando automáticamente.

28 s se ejecutó 3:21 p.m.

Prueba 2



Playing with dask

```
[1] import pandas as pd
import dask.dataframe as dd

flights_file1 = "/content/drive/MyDrive/data/flights/combined_Flights_2018.parquet"
# flights_file2 = "/content/drive/MyDrive/data/flights/combined_Flights_2019.parquet"
# flights_file3 = "/content/drive/MyDrive/data/flights/combined_Flights_2020.parquet"
# flights_file4 = "/content/drive/MyDrive/data/flights/combined_Flights_2021.parquet"
flights_file5 = "/content/drive/MyDrive/data/flights/combined_Flights_2022.parquet"
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)

[3] df = dd.concat([df1, df5])

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

	FlightDate	Airline	Origin	Dest	Cancelled	Diverted	\
0	2018-01-23	Endeavor Air Inc.	ABY	ATL	False	False	
1	2018-01-24	Endeavor Air Inc.	ABY	ATL	False	False	
2	2018-01-25	Endeavor Air Inc.	ABY	ATL	False	False	
3	2018-01-26	Endeavor Air Inc.	ABY	ATL	False	False	
4	2018-01-27	Endeavor Air Inc.	ABY	ATL	False	False	
...	
590537	2022-03-31	Republic Airlines	MSY	EMR	False	True	

0 s se ejecutó 3:56p.m.

Recursos

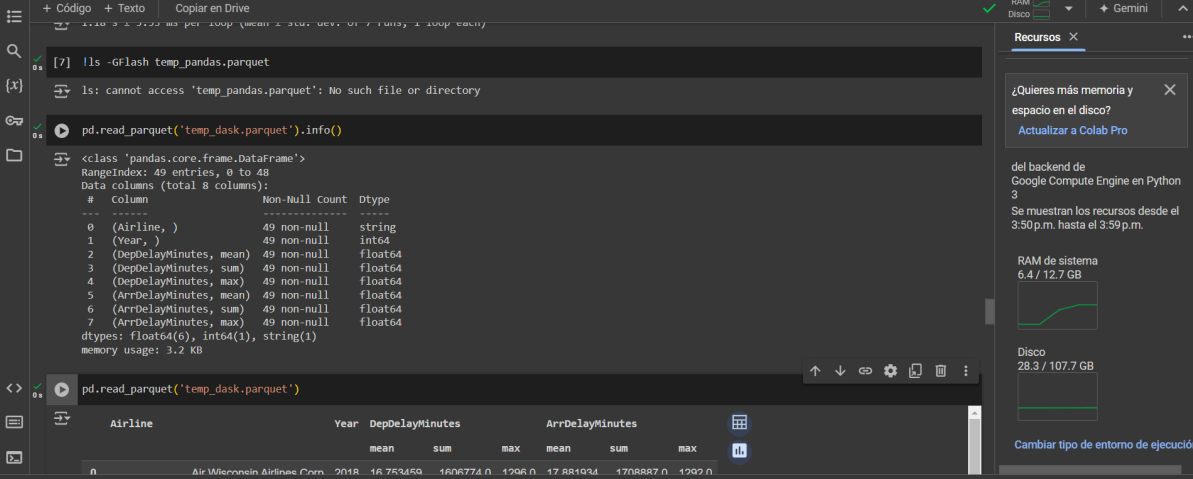
¿Quieres más memoria y espacio en el disco?
[Actualizar a Colab Pro](#)

del backend de Google Compute Engine en Python 3
Se muestran los recursos desde el 3:50 p.m. hasta el 3:58 p.m.

RAM de sistema
6.4 / 12.7 GB

Disco
28.3 / 107.7 GB

[Cambiar tipo de entorno de ejecución](#)



```
[7] !ls -l temp_pandas.parquet

ls: cannot access 'temp_pandas.parquet': No such file or directory

pd.read_parquet('temp_dask.parquet').info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49 entries, 0 to 48
Data columns (total 8 columns):
 #   column                                Non-Null Count  Dtype
---  ---
 0   (Airline, )                          49 non-null    string
 1   (Year, )                             49 non-null    int64
 2   (DepDelayMinutes, mean)              49 non-null    float64
 3   (DepDelayMinutes, sum)               49 non-null    float64
 4   (DepDelayMinutes, max)               49 non-null    float64
 5   (ArrDelayMinutes, mean)              49 non-null    float64
 6   (ArrDelayMinutes, sum)               49 non-null    float64
 7   (ArrDelayMinutes, max)               49 non-null    float64
dtypes: float64(6), int64(1), string(1)
memory usage: 3.2 KB
```

	Airline	Year	DepDelayMinutes	ArrDelayMinutes				
			mean	sum	max	mean	sum	max
0	Air Wisconsin Airlines Corp.	2018	16.753450	1606774.0	1206.0	17.881934	1708887.0	1202.0

0 s se ejecutó 3:56p.m.

Recursos

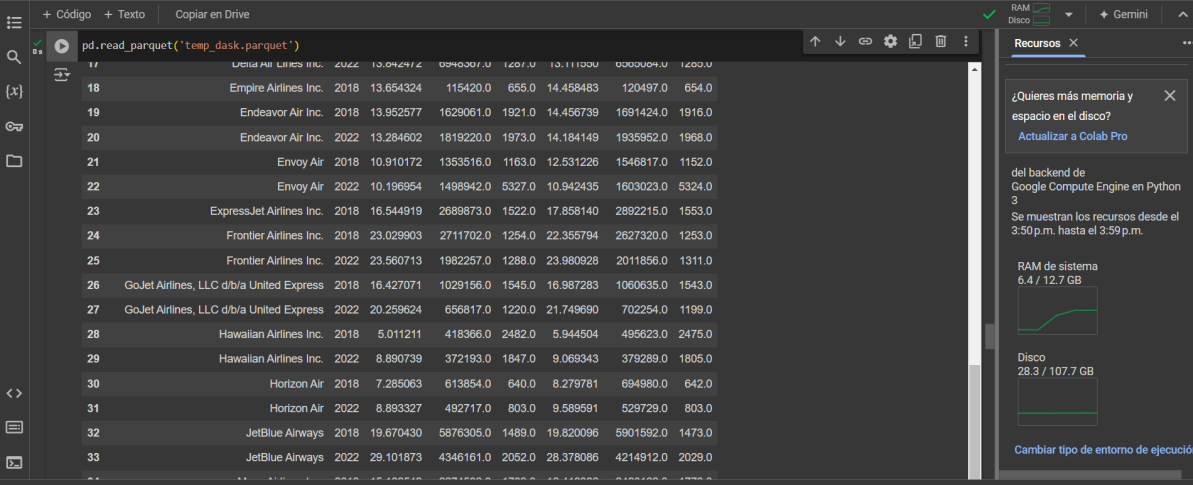
¿Quieres más memoria y espacio en el disco?
[Actualizar a Colab Pro](#)

del backend de Google Compute Engine en Python 3
Se muestran los recursos desde el 3:50 p.m. hasta el 3:59 p.m.

RAM de sistema
6.4 / 12.7 GB

Disco
28.3 / 107.7 GB

[Cambiar tipo de entorno de ejecución](#)



```
pd.read_parquet('temp_dask.parquet')
```

	Airline	Year	DepDelayMinutes	ArrDelayMinutes				
17	Delta Air Lines Inc.	2022	13.042472	1207.0	13.111000	1000004.0	1200.0	
18	Empire Airlines Inc.	2018	13.654324	115420.0	655.0	14.458483	120497.0	654.0
19	Endeavor Air Inc.	2018	13.952577	1629061.0	1921.0	14.456739	1691424.0	1916.0
20	Endeavor Air Inc.	2022	13.284602	1819220.0	1973.0	14.184149	1935952.0	1968.0
21	Envoy Air	2018	10.910172	1353516.0	1163.0	12.531226	1546817.0	1152.0
22	Envoy Air	2022	10.196954	1498942.0	5327.0	10.942435	1603023.0	5324.0
23	ExpressJet Airlines Inc.	2018	16.544919	2689873.0	1522.0	17.858140	2892215.0	1553.0
24	Frontier Airlines Inc.	2018	23.029903	2711702.0	1254.0	22.355794	2627320.0	1253.0
25	Frontier Airlines Inc.	2022	23.560713	1982257.0	1288.0	23.980928	2011856.0	1311.0
26	GoJet Airlines, LLC d/b/a United Express	2018	16.427071	1029156.0	1545.0	16.987283	1080635.0	1543.0
27	GoJet Airlines, LLC d/b/a United Express	2022	20.259624	656817.0	1220.0	21.749690	702254.0	1199.0
28	Hawaiian Airlines Inc.	2018	5.011211	418366.0	2482.0	5.944504	495623.0	2475.0
29	Hawaiian Airlines Inc.	2022	8.890739	372193.0	1847.0	9.069343	379289.0	1805.0
30	Horizon Air	2018	7.285063	613854.0	640.0	8.279781	694980.0	642.0
31	Horizon Air	2022	8.893327	492717.0	803.0	9.589591	529729.0	803.0
32	JetBlue Airways	2018	19.670430	5876305.0	1489.0	19.820096	5901592.0	1473.0
33	JetBlue Airways	2022	29.101873	4346161.0	2052.0	28.378086	4214812.0	2029.0

0 s se ejecutó 3:56p.m.

Recursos

¿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 3:50 p.m. hasta el 3:59 p.m.

RAM de sistema
6.4 / 12.7 GB

Disco
28.3 / 107.7 GB

[Cambiar tipo de entorno de ejecución](#)

Prueba 3

The screenshot shows a Jupyter Notebook titled "Playing with dask". The code defines five flight data files for the years 2018 through 2022 and reads them into Dask DataFrames. These are then concatenated into a single Dask DataFrame and computed.

```
import pandas as pd
import dask.dataframe as dd

# flights_file1 = "/content/drive/myDrive/data/flights/combined_Flights_2018.parquet"
# flights_file2 = "/content/drive/myDrive/data/flights/combined_Flights_2019.parquet"
# flights_file3 = "/content/drive/myDrive/data/flights/combined_Flights_2020.parquet"
# flights_file4 = "/content/drive/myDrive/data/flights/combined_Flights_2021.parquet"
# flights_file5 = "/content/drive/myDrive/data/flights/combined_Flights_2022.parquet"
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)

[12] df = dd.concat([df1, df5])

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

	FlightDate	Airline	Origin	Dest	Cancelled	Diverted	\
0	2018-01-23	Endeavor Air Inc.	ABY	ATL	False	False	
1	2018-01-24	Endeavor Air Inc.	ABY	ATL	False	False	
2	2018-01-25	Endeavor Air Inc.	ABY	ATL	False	False	
3	2018-01-26	Endeavor Air Inc.	ABY	ATL	False	False	
4	2018-01-27	Endeavor Air Inc.	ABY	ATL	False	False	
...
590537	2022-03-31	Republic Airlines	MSY	EWK	False	True	
590538	2022-03-31	Republic Airlines	CLT	EWK	True	False	
590539	2022-03-08	Republic Airlines	ALB	ORD	False	False	
590540	2022-03-25	Republic Airlines	EWK	PIT	False	True	
590541	2022-03-07	Republic Airlines	EWK	ROU	False	True	

The screenshot shows the continuation of the Jupyter Notebook. The code computes the Dask DataFrame and displays the results, including flight details and delay statistics.

```
print(df.compute())
```

	FlightDate	Airline	Origin	Dest	Cancelled	Diverted	\
0	2018-01-23	Endeavor Air Inc.	ABY	ATL	False	False	
1	2018-01-24	Endeavor Air Inc.	ABY	ATL	False	False	
2	2018-01-25	Endeavor Air Inc.	ABY	ATL	False	False	
3	2018-01-26	Endeavor Air Inc.	ABY	ATL	False	False	
4	2018-01-27	Endeavor Air Inc.	ABY	ATL	False	False	
...
590537	2022-03-31	Republic Airlines	MSY	EWK	False	True	
590538	2022-03-31	Republic Airlines	CLT	EWK	True	False	
590539	2022-03-08	Republic Airlines	ALB	ORD	False	False	
590540	2022-03-25	Republic Airlines	EWK	PIT	False	True	
590541	2022-03-07	Republic Airlines	EWK	ROU	False	True	

	CRSDepTime	DepTime	DepDelayMinutes	DepDelay	...	WheelsOff	\
0	1202	1157.0	0.0	-5.0	...	1211.0	
1	1202	1157.0	0.0	-5.0	...	1210.0	
2	1202	1153.0	0.0	-9.0	...	1211.0	
3	1202	1159.0	0.0	-12.0	...	1207.0	
4	1400	1355.0	0.0	-5.0	...	1412.0	
...
590537	1949	2014.0	25.0	25.0	...	2031.0	
590538	1723	1817.0	44.0	44.0	...	NaN	
590539	1700	2318.0	378.0	378.0	...	2337.0	
590540	2129	2322.0	113.0	113.0	...	2347.0	
590541	1154	1148.0	0.0	-6.0	...	1201.0	

	WheelsOn	TaxiIn	CRSArrTime	ArrDelay	ArrDel15	ArrivalDelayGroups	\
0	1249.0	7.0	1304	-8.0	0.0	-1.0	
1	1246.0	12.0	1304	-6.0	0.0	-1.0	
...

The screenshot shows the Jupyter Notebook performing a groupby aggregation on the flight data by airline and year, calculating mean and maximum delay minutes. The result is saved to a temporary parquet file.

```
[14] df = df.compute()

[15] %%timeit
df_agg = df.groupby(["Airline", "year"])[["DepDelayMinutes", "ArrDelayMinutes"]].agg(
    ["mean", "sum", "max"]
)
df_agg = df_agg.reset_index()
df_agg.to_parquet("temp_dask.parquet")
```

1.41 s ± 265 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

```
[19] !ls -l temp_dask.parquet
```

12K -rw-r--r-- 1 root 11K Jun 20 21:03 temp_dask.parquet

```
[16] pd.read_parquet("temp_dask.parquet").info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49 entries, 0 to 48
Data columns (total 8 columns):
#   Column                                Non-Null Count  Dtype
---  ---                                ---
0   (Airline, )                           49 non-null     string
1   (Year, )                              49 non-null     int64
2   (DepDelayMinutes, mean)               49 non-null     float64
3   (DepDelayMinutes, sum)                49 non-null     float64
4   (DepDelayMinutes, max)                49 non-null     float64
5   (ArrDelayMinutes, mean)               49 non-null     float64
6   (ArrDelayMinutes, sum)                49 non-null     float64
```

The screenshot shows the final data output from the Jupyter Notebook, displaying a detailed view of the flight data with columns for airline, year, flight date, and various delay metrics.

```
[18] pd.read_parquet("temp_dask.parquet")
```

	AIRLINE	YEAR	FLIGHTDATE	CRSDPTTIME	DEPTTIME	DEPDELAY	ARRPTTIME	ARRTIME	ARRDELAY	ARRDEL15	ARRIVALDELAYGROUPS
18	Empire Airlines Inc.	2018	13.054324	1154.20	655.0	14.458483	120497.0	654.0			
19	Endeavor Air Inc.	2018	13.952577	1629061.0	1921.0	14.456739	1691424.0	1916.0			
20	Endeavor Air Inc.	2022	13.284602	1819220.0	1973.0	14.184149	1935962.0	1968.0			
21	Envoy Air	2018	10.910172	1353516.0	1163.0	12.531226	1546817.0	1152.0			
22	Envoy Air	2022	10.196954	1468942.0	5327.0	10.942435	1603023.0	5324.0			
23	ExpressJet Airlines Inc.	2018	16.544919	2688873.0	1522.0	17.858140	2892215.0	1553.0			
24	Frontier Airlines Inc.	2018	23.029903	2711702.0	1254.0	22.355794	2627320.0	1253.0			
25	Frontier Airlines Inc.	2022	23.560713	1982257.0	1288.0	23.980628	2011856.0	1311.0			
26	GoJet Airlines, LLC d/b/a United Express	2018	16.427071	1029156.0	1545.0	16.987283	1060635.0	1543.0			
27	GoJet Airlines, LLC d/b/a United Express	2022	20.259624	656817.0	1220.0	21.749690	702254.0	1198.0			
28	Hawaiian Airlines Inc.	2018	5.011211	418366.0	2482.0	5.944504	495623.0	2475.0			
29	Hawaiian Airlines Inc.	2022	8.890739	372193.0	1847.0	9.069343	379289.0	1805.0			
30	Horizon Air	2018	7.285063	613854.0	640.0	8.279781	694880.0	642.0			
31	Horizon Air	2022	8.893327	492717.0	803.0	9.589591	529729.0	803.0			
32	JetBlue Airways	2018	19.670430	5876305.0	1489.0	19.820096	5901592.0	1473.0			
33	JetBlue Airways	2022	29.101873	4348161.0	2052.0	29.379096	4714812.0	2029.0			

Prueba 4

The first screenshot shows the initial code for reading flight data from four Parquet files (2018, 2019, 2020, 2021, 2022) and concatenating them into a single DataFrame. The output shows the first few rows of the concatenated data.

```
[21] import pandas as pd
import dask.dataframe as dd

flights_file1 = "/content/drive/MyDrive/data/flights/Combined_Flights_2018.parquet"
flights_file2 = "/content/drive/MyDrive/data/flights/Combined_Flights_2019.parquet"
flights_file3 = "/content/drive/MyDrive/data/flights/Combined_Flights_2020.parquet"
flights_file4 = "/content/drive/MyDrive/data/flights/Combined_Flights_2021.parquet"
flights_file5 = "/content/drive/MyDrive/data/flights/Combined_Flights_2022.parquet"

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([df1, df3])

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

	FlightDate	Airline	Origin	Dest	Cancelled	Diverted	\
0	2018-01-23	Endeavor Air Inc.	ABY	ATL	False	False	
1	2018-01-24	Endeavor Air Inc.	ABY	ATL	False	False	
2	2018-01-25	Endeavor Air Inc.	ABY	ATL	False	False	
3	2018-01-26	Endeavor Air Inc.	ABY	ATL	False	False	
4	2018-01-27	Endeavor Air Inc.	ABY	ATL	False	False	
...
331233	2020-04-01	Republic Airlines	CLT	DEN	True	False	

The second screenshot shows the code for aggregating the data by airline and year, calculating the mean, sum, and max of departure and arrival delays. The output shows the execution time and the file path of the saved Parquet file.

```
%%timeit
df_agg = df.groupby(['Airline', 'Year'])[['DepDelayMinutes', 'ArrDelayMinutes']].agg(
    ["mean", "sum", "max"]
)
df_agg = df_agg.reset_index()
df_agg.to_parquet("temp_dask.parquet")
```

1.34 s ± 67.2 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

```
[26] !ls -lG temp_dask.parquet
```

12K -rw-r--r-- 1 root 11K Jun 20 21:17 temp_dask.parquet

```
pd.read_parquet("temp_dask.parquet").info()
```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53 entries, 0 to 52
Data columns (total 8 columns):
column Non-Null Count Dtype ---
0 (Airline,) 53 non-null string
1 (Year,) 53 non-null int64
2 (DepDelayMinutes, mean) 53 non-null float64
3 (DepDelayMinutes, sum) 53 non-null float64
4 (DepDelayMinutes, max) 53 non-null float64
5 (ArrDelayMinutes, mean) 53 non-null float64
6 (ArrDelayMinutes, sum) 53 non-null float64
7 (ArrDelayMinutes, max) 53 non-null float64

The third screenshot shows the final aggregated data as a table.

	Airline	Year	DepDelayMinutes			ArrDelayMinutes		
			mean	sum	max	mean	sum	max
0	Air Wisconsin Airlines Corp	2018	16.753459	1606774.0	1296.0	17.881934	1708887.0	1292.0
1	Air Wisconsin Airlines Corp	2020	8.583725	433315.0	1460.0	8.982529	452450.0	1439.0
2	Alaska Airlines Inc.	2018	7.503389	1374801.0	839.0	8.759125	1600336.0	842.0
3	Alaska Airlines Inc.	2020	5.818328	772930.0	823.0	6.365082	843157.0	788.0
4	Allegiant Air	2018	17.080944	1630769.0	1462.0	17.547588	1670390.0	1505.0
5	Allegiant Air	2020	12.825575	1080016.0	1648.0	13.331111	1115734.0	1645.0
6	American Airlines Inc.	2018	13.141112	4993399.0	2109.0	14.225643	5387564.0	2153.0
7	American Airlines Inc.	2020	7.624477	4084097.0	3890.0	7.861155	4202644.0	3864.0
8	Cape Air	2018	4.643761	7704.0	430.0	5.390332	8921.0	446.0
9	Capital Cargo International	2018	14.876462	625823.0	841.0	15.310871	640270.0	814.0
10	Capital Cargo International	2020	7.665063	512969.0	1482.0	8.427212	561522.0	1470.0
11	Comair Inc.	2018	12.776783	1452158.0	1121.0	12.789146	1447872.0	1110.0
12	Comair Inc.	2020	10.068723	1798294.0	1919.0	10.686808	1903235.0	1888.0
13	Communtair Aka Champlain Enterprises, Inc.	2018	28.243923	1290832.0	1352.0	29.284076	1332689.0	1353.0

Conclusiones

Luego de realizar los análisis con cada una de las librerías y con varias combinaciones de datos, podemos realizar las siguientes conclusiones de este laboratorio:

- El método polars es el más flexible y sencillo con los datos ya que este trabaja con pocos recursos tanto en programación como en hardware
- El método pandas terminó siendo el que tuvo más complicaciones para realizar el análisis de los datos dado a que de las 4 pruebas, solo una pudo realizarse con éxito
- Los grupos de datos más flexibles de trabajar fueron los de 2018 (`flights_file1`) y 2020 (`flights_file1`)
- El grupo de datos con más inconvenientes para trabajar fueron los de 2019 (`flights_file2`) debido a que estos mismos eran los que hacían sobrepasar el límite de RAM con más frecuencia a comparación de los demás grupos de datos
- El método PySpark fue el método más equilibrado en cuanto a uso de RAM con un promedio entre las 2 Gb y eficiencia de manejo de datos. Esto está demostrado con el hecho de que los métodos Pandas y Dusk superan las 7 Gb de RAM y el método Polars maneja usos entre la 1 y 2 Gb pero su manejo de datos es muy sencillo.