

## Laboratorio Numero 2

Luis Felipe Sanchez Sanchez - 90613  
Sebastian Morales Devia - 73487

Universidad ECCI  
Facultad de Ingeniería

Elias Buitrago Bolivar

Bogotá  
2024

Introducción

Al comparar diferentes bibliotecas en cuanto a su rendimiento y efectividad en el procesamiento de archivos .parquet, es necesario observar más de cerca con qué tipo de conjuntos de datos trabajan. En esta investigación, analizaremos las capacidades de varias bibliotecas conocidas para cargar y trabajar con archivos .parquet, considerando diferentes escenarios para cargas de archivos y distintos niveles de complejidad de datos. A partir de esta evaluación podemos identificar la mejor biblioteca en función de su velocidad, eficacia y capacidad para manejar grandes cantidades de datos en formato .parquet.

PANDAS

Prueba 1:

+ Código

+ Texto

6

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

6.766355

1929514.0

1503.0

52

Virgin America

2018

10.902274

187977.0

520.0

11.950971

204995.0

504.0

1

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: 3.4+ KB

Recursos

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 84 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 4:46 p.m. hasta el 4:51 p.m.

RAM de sistema  
11.1 / 12.7 GB

Disco  
27.8 / 107.7 GB

Cambiar tipo de entorno de ejecución

0 s se ejecutó 4:50 p.m.

Archivos usados: 1 y 3  
RAM: 11.1 / 12./ GB  
NOTAS: Ejecuta todas las líneas sin lio  
Ejecucción: 52 SEG

Prueba 2:

+ Código

+ Texto

1 from google.colab import drive

2 drive.mount('/content/drive')

Mounted at /content/drive

Playing with pandas

1 import pandas as pd

2 flights\_file1 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2018.parquet"

3 flights\_file2 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2019.parquet"

4 flights\_file3 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2020.parquet"

5 flights\_file4 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2021.parquet"

6 flights\_file5 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2022.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, df5])

2 df = df2

1 # %timeit

2

Tu sesión falló porque se usó toda la RAM disponible.

Conectado a del backend de Google Compute Engine en Python 3

Recursos

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 84 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

Uso de recursos desde 4:56 p.m.

RAM de sistema 1.0 / 12.7 GB

Disco 28.3 / 107.7 GB

Cambiar tipo de entorno de ejecución

Archivos usados: 1, 2 y 5

RAM: - / 12.7 GB

NOTAS: Desbordamiento en la sección de concatenación

Ejecucción: 39 SEG

### Prueba 3:

+ Código

+ Texto

1 import pandas as pd

2 #flights\_file1 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2018.parquet"

3 #flights\_file2 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2019.parquet"

4 flights\_file3 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2020.parquet"

5 flights\_file4 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2021.parquet"

6 #flights\_file5 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2022.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([df3, df4])

2 df = df2

1 # %timeit

2

3 df\_agg = df.groupby(['Airline', 'Year'])[['DepDelayMinutes', 'ArrDelayMinutes']].agg([

4 ["mean", "sum", "max"]

5 ])

6 df\_agg = df\_agg.reset\_index()

7 df\_agg.to\_parquet("temp\_pandas.parquet")

1 !ls -l temp\_pandas.parquet

12K -rw-r--r-- 1 root 11K Jul 3 21:49 temp\_pandas.parquet

1 pd.read\_parquet("temp\_pandas.parquet")

3 s se ejecutó 5:00 p.m.

Recursos

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 84 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 4:56 p.m. hasta el 5:00 p.m.

RAM de sistema 11.9 / 12.7 GB

Disco 28.5 / 107.7 GB

Cambiar tipo de entorno de ejecución

Archivos usados: 3 y 4

RAM: 11.7/ 12.7 GB

NOTAS: Se ejecutaron los comandos sin desbordamiento

Ejecucción: 48 SEG

### Prueba 4:



## Intento 2:

+ Código + Texto
RAM Disco
+ Gemini ↕

```
[1] 1 import polars as pl

[6] 1 #flights_file1 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2018.parquet"
2 flights_file2 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2019.parquet"
3 flights_file3 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2020.parquet"
4 flights_file4 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2021.parquet"
5 flights_file5 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2022.parquet"
6 df1 = pl.scan_parquet(flights_file1)
7 df2 = pl.scan_parquet(flights_file2)
8 df3 = pl.scan_parquet(flights_file3)
9 df4 = pl.scan_parquet(flights_file4)
10 df5 = pl.scan_parquet(flights_file5)

[7] 1 %timeit
2
3 df_polars = (
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')
```

< ⚙️ ⏮ ⏭ ⏯ ⚙️ 📄 🖨 ☰

```
1 ls -lGash temp_polars.parquet

12K -rw-r--r-- 1 root 8.1K Jul 3 23:31 temp_polars.parquet
```

**Recursos X**

No te has suscrito. [Mas informacion](#)

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 18:23 a las 18:32

RAM del sistema	Disco
1.4 / 12.7 GB	28.5 / 107.7 GB

[Cambiar tipo de entorno de ejecución](#)

✓ 0 s completado a las 18:32
●

Archivos usados: 2, 3 y 4

RAM: 1.4 / 12.7 GB

NOTAS: Ejecutó sin problemas

Ejecución: 60 SEG

### Intento 3:

+ Código + Texto
RAM Disco Gemini ↕

```
[1] 1 import polars as pl

[10] 1 flights_file1 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2018.parquet"
   2 #flights_file2 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2019.parquet"
   3 #flights_file3 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2020.parquet"
   4 #flights_file4 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2021.parquet"
   5 flights_file5 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2022.parquet"
   6 df1 = pl.scan_parquet(flights_file1)
   7 #df2 = pl.scan_parquet(flights_file2)
   8 #df3 = pl.scan_parquet(flights_file3)
   9 #df4 = pl.scan_parquet(flights_file4)
  10 df5 = pl.scan_parquet(flights_file5)

1 %timeit
2
3 df_polars = (
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")

... <magic-timetr>3: DeprecationWarning: 'groupby' is deprecated. It has been renamed to 'group_by'.

[12] 1 !ls -lGfash temp_polars.parquet

12K -rw-r--r-- 1 root 8.1K Jul 3 23:35 temp_polars.parquet
```

↑ ↓ 🔍 ⚙️ 🗑️ 📄

```
>>> magic-timetr>3: DeprecationWarning: 'groupby' is deprecated. It has been renamed to 'group_by'.
```

12K -rw-r--r-- 1 root 8.1K Jul 3 23:35 temp\_polars.parquet

Ejecución (32 s) <cell line: 1> > run\_cell\_magic() > run\_cell\_magic() > timeitr() > lambda> timeitr() > repeat() > timeitr() > inner() > collect()

### 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.

[Gestionar sesiones](#)

---

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

---

del backend de Google Compute Engine que utiliza Python 3

Mostrando recursos desde las 18:23 a las 18:36

RAM del sistema	Disco
3.9 / 12.7 GB	28.5 / 107.7 GB

[Cambiar tipo de entorno de ejecución](#)

Archivos usados: 1 y 5

RAM: 3.9/ 12.7 GB

NOTAS: Se queda procesando dicha línea por más de 1:30 min

Ejecución: +1:30 MIN

### Intento 4:

+ Código + Texto
✓ RAM Disco ▾ + Gemini ↗

---

### ▼ Playing with Polars

```
[1] 1 import polars as pl

2 #flights_file1 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2018.parquet"
3 #flights_file2 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2019.parquet"
4 flights_file3 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2020.parquet"
5 #flights_file4 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2021.parquet"
6 #flights_file5 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2022.parquet"
7 df1 = pl.scan_parquet(flights_file1)
8 df2 = pl.scan_parquet(flights_file2)
9 df3 = pl.scan_parquet(flights_file3)
10 df4 = pl.scan_parquet(flights_file4)
11 df5 = pl.scan_parquet(flights_file5)

12 %timeit
13 df_polars = (
14     pl.concat([df1, df2, df3, df4, df5])
15     .groupby(["Airline", "Year"])
16     .agg([
17         pl.col("DepDelayMinutes").mean().alias("avg_dep_delay"),
18         pl.col("DepDelayMinutes").sum().alias("sum_dep_delay"),
19         pl.col("DepDelayMinutes").max().alias("max_dep_delay"),
20         pl.col("ArrDelayMinutes").mean().alias("avg_arr_delay"),
21         pl.col("ArrDelayMinutes").sum().alias("sum_arr_delay"),
22         pl.col("ArrDelayMinutes").max().alias("max_arr_delay"),
23     ])
24 )
25 df_polars.collect()
26 df_polars.write_parquet("temp_polars.parquet")
```

<magic-timeit>3: DeprecationWarning: 'groupby' is deprecated. It has been renamed to 'group\_by'.  
9.45 s ± 739 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

✓ 1 min 16 s completado a las 18:39

**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.

[Gestionar sesiones](#)

---

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

del backend de Google Compute Engine que utiliza Python 3  
Mostrando recursos desde las 18:23 a las 18:39

RAM del sistema	Disco
1.4 / 12.7 GB	28.5 / 107.7 GB

[Cambiar tipo de entorno de ejecución](#)

# PYSPARK

## Intento 1:

+ Código

+ Texto

✓ [18] installing collected packages: pyspark

Successfully installed pyspark-3.5.1

✓ [19]

```
1 from pyspark.sql import SparkSession
2 from pyspark.sql.functions import avg, max, sum, concat
```

✓ [20]

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

✓ [21]

```
1 flights_file1 = "/content/drive/MyDrive/Colab Notebooks/Fallier/Combined_Flights_2018.parquet"
2 flights_file2 = "/content/drive/MyDrive/Colab Notebooks/Fallier/Combined_Flights_2019.parquet"
3 flights_file3 = "/content/drive/MyDrive/Colab Notebooks/Fallier/Combined_Flights_2020.parquet"
4 flights_file4 = "/content/drive/MyDrive/Colab Notebooks/Fallier/Combined_Flights_2021.parquet"
5 flights_file5 = "/content/drive/MyDrive/Colab Notebooks/Fallier/Combined_Flights_2022.parquet"
```

✓ [22]

```
1 df_spark1 = spark.read.parquet(flights_file1)
2 df_spark2 = spark.read.parquet(flights_file2)
3 df_spark3 = spark.read.parquet(flights_file3)
4 df_spark4 = spark.read.parquet(flights_file4)
5 df_spark5 = spark.read.parquet(flights_file5)
```

✓ [25]

```
1 df_spark = df_spark1 # Initialize df_spark with the first DataFrame
2 df_spark = df_spark.union(df_spark2)
3 df_spark = df_spark.union(df_spark3)
4 df_spark = df_spark.union(df_spark4)
5 df_spark = df_spark.union(df_spark5)
```

✓ [26]

```
1 %timeit
2
3 df_spark_agg = df_spark.groupby("Airline", "Year").agg(
4     avg("ArrDelayMinutes").alias("avg_arr_delay"),
5     sum("ArrDelayMinutes").alias("sum_arr_delay"),
6     max("ArrDelayMinutes").alias("max_arr_delay"),
7     avg("DepDelayMinutes").alias("avg_dep_delay"),
8     sum("DepDelayMinutes").alias("sum_dep_delay"),
9     max("DepDelayMinutes").alias("max_dep_delay"),
10 )
11 df_spark_agg.write.mode("overwrite").parquet("temp_spark.parquet")
```

4.97 s ± 715 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

✓ 0 s completado a las 18:55

✓ 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 82 horas 50 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 18:23 a las 18:55

RAM del sistema

2.1 / 12.7 GB

Disco

29.4 / 107.7 GB

[Cambiar tipo de entorno de ejecución](#)

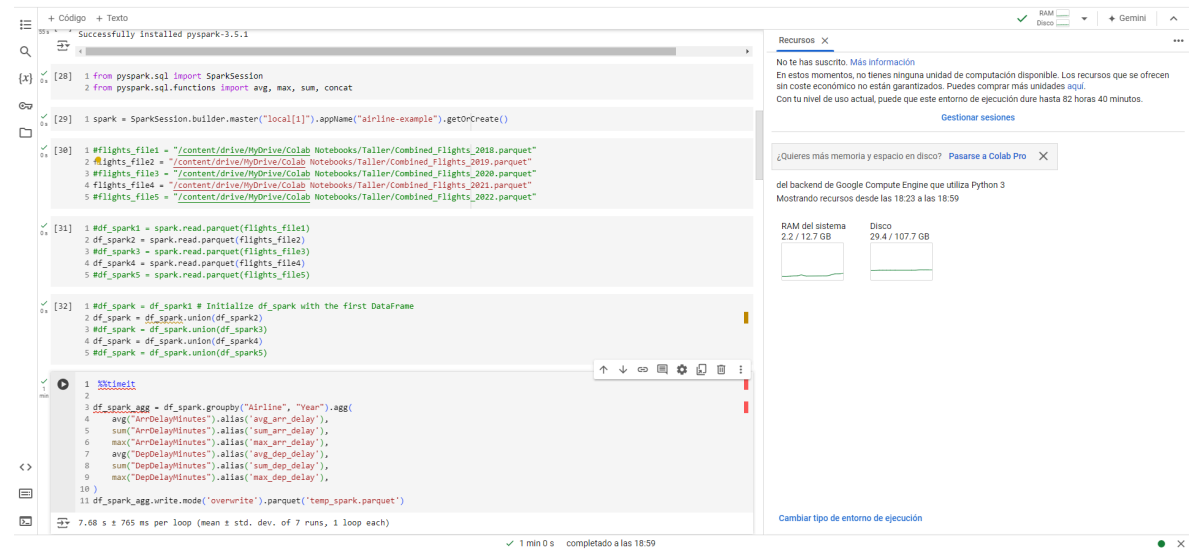
Archivos usados: 1, 3 y 5

RAM: 2.1 / 12.7 GB

NOTAS: Ejecución exitosa

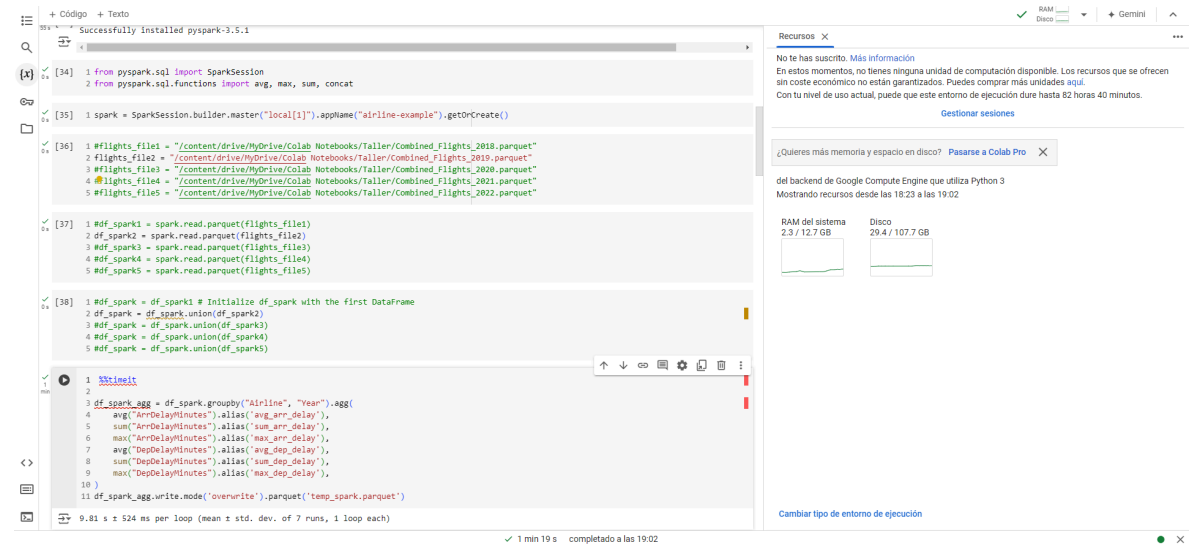
Ejecución: 1.81 MIN

## Intento 2:



Archivos usados: 2 y 4  
RAM: 2.2/ 12.7 GB  
NOTAS: Compilo sin ningún error  
Ejecución: 1 MIN

### Intento 3:



Archivos usados: 2  
RAM: 2.4/ 12.7 GB  
NOTAS: Ejecución sin problemas  
Ejecución: 1.10 MIN

### Intento 4:

+ Código+ Texto

Successfully Installed pyspark-3.5.1

1 from pyspark.sql import SparkSession

2 from pyspark.sql.functions import avg, max, sum, concat

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

1 flights\_file1 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2018.parquet"

2 flights\_file2 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2019.parquet"

3 flights\_file3 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2020.parquet"

4 flights\_file4 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2021.parquet"

5 flights\_files = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2022.parquet"

1 df\_spark1 = spark.read.parquet(flights\_file1)

2 df\_spark2 = spark.read.parquet(flights\_file2)

3 df\_spark3 = spark.read.parquet(flights\_file3)

4 df\_spark4 = spark.read.parquet(flights\_file4)

5 df\_spark5 = spark.read.parquet(flights\_files)

1 df\_spark = df\_spark1 # Initialize df\_spark with the first DataFrame

2 df\_spark = df\_spark.union(df\_spark2)

3 df\_spark = df\_spark.union(df\_spark3)

4 df\_spark = df\_spark.union(df\_spark4)

5 df\_spark = df\_spark.union(df\_spark5)

1 %timeit

2

3 df\_spark\_agg = df\_spark.groupby("Airline", "Year").agg(

4 avg("ArrDelayMinutes").alias("avg\_arr\_delay"),

5 sum("ArrDelayMinutes").alias("sum\_arr\_delay"),

6 max("ArrDelayMinutes").alias("max\_arr\_delay"),

7 avg("DepDelayMinutes").alias("avg\_dep\_delay"),

8 sum("DepDelayMinutes").alias("sum\_dep\_delay"),

9 max("DepDelayMinutes").alias("max\_dep\_delay"),

10 )

11 df\_spark\_agg.write.mode("overwrite").parquet("temp\_spark.parquet")

7.34 s ± 883 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

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 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 18:23 a las 19:07

RAM del sistema

2.4 / 12.7 GB

Disco

29.4 / 107.7 GB

Cambiar tipo de entorno de ejecución

58 s completado a las 19:07

Archivos usados: 1, 2, 3, 4 y 5  
RAM: 2.4/ 12.7 GB  
NOTAS: Exitosa la ejecución  
Ejecución: 59 SEG

DASK

Intento 1:

+ Código+ Texto

[ ] 48 celdas ocultas

Playing with dask

1 import pandas as pd

2 import dask.dataframe as dd

3 #flights\_file1 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2018.parquet"

4 #flights\_file2 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2019.parquet"

5 flights\_file3 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2020.parquet"

6 flights\_file4 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2021.parquet"

7 flights\_file5 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined\_Flights\_2022.parquet"

8 df1 = dd.read\_parquet(flights\_file1)

9 df2 = dd.read\_parquet(flights\_file2)

10 df3 = dd.read\_parquet(flights\_file3)

11 df4 = dd.read\_parquet(flights\_file4)

12 df5 = dd.read\_parquet(flights\_file5)

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

1 print(df.compute())

	FlightDate	Airline	Origin	Dest	Cancelled	Diverted	\
0	2020-09-01	Comair Inc.	PHL	DAY	False	False	
1	2020-09-02	Comair Inc.	PHL	DAY	False	False	
2	2020-09-03	Comair Inc.	PHL	DAY	False	False	
3	2020-09-04	Comair Inc.	PHL	DAY	False	False	
4	2020-09-05	Comair Inc.	PHL	DAY	False	False	
...	...	...	...	...	...	...	
590537	2022-03-31	Republic Airlines	HSV	ENR	False	True	
590538	2022-03-17	Republic Airlines	CLT	ENR	True	False	
590539	2022-03-08	Republic Airlines	ALB	ORD	False	False	
590540	2022-03-25	Republic Airlines	ENR	PIT	False	True	
590541	2022-03-07	Republic Airlines	ENR	RDU	False	True	

	CRSDepTime	DepTime	DepDelayMinutes	DepDelay	...	WheelsOff	\
0	1905	1858.0	0.0	-7.0	...	1914.0	
1	1905	1858.0	0.0	-7.0	...	1914.0	
2	1905	1855.0	0.0	-10.0	...	2000.0	
3	1905	1857.0	0.0	-8.0	...	1918.0	
4	1905	1856.0	0.0	-9.0	...	1910.0	

31 s completado a las 19:12

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 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 18:23 a las 19:12

RAM del sistema

9.9 / 12.7 GB

Disco

29.4 / 107.7 GB

Cambiar tipo de entorno de ejecución

31 s completado a las 19:12

Archivos usados: 3, 4 y 5  
RAM: 9.9/ 12.7 GB  
NOTAS: Se corta el la linea timeit  
Ejecución: 31 SEG

Intento 2:





+ Código + Texto

[ ] 4 8 celdas ocultas

Playing with dask

```
1 import pandas as pd
2 import dask.dataframe as dd
3 flights_file1 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2018.parquet"
4 flights_file2 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2019.parquet"
5 flights_file3 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2020.parquet"
6 flights_file4 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2021.parquet"
7 flights_file5 = "/content/drive/MyDrive/Colab Notebooks/Taller/Combined_Flights_2022.parquet"
8 df1 = dd.read_parquet(flights_file1)
9 df2 = dd.read_parquet(flights_file2)
10 df3 = dd.read_parquet(flights_file3)
11 df4 = dd.read_parquet(flights_file4)
12 df5 = dd.read_parquet(flights_file5)
```

+ Código + Texto

[13] 1 df = dd.concat([df1, df2, df3, df4, df5])

1 print(df.compute())

...

Recursos X AttributeError

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 85 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 19:16 a las 19:22

RAM del sistema  
11.9 / 12.7 GB

Disco  
29.4 / 107.7 GB

Cambiar tipo de entorno de ejecución

En ejecución (25 s) <cell line: 1> > compute() > compute() > get() > get\_async() > queue.get() > get() > wait()

Archivos usados: 1, 2, 3, 4 y 5  
RAM: 11.9/ 12.7 GB  
NOTAS: Desbordamiento con todos los archivos  
Ejecución: 40 SEG

## Conclusiones

- La librería pandas la mayoría de veces que se corrían varios datos si no colapsaba la RAM, quedaba al límite
- El tiempo de ejecución con pandas es promedio a todas las demás librerías, sin embargo, con pandas se rompe
- El correr 2 conjuntos de datos podría provocar que se caiga COLAB
- El correr con DASK varios datos puede provocar en promedio más de 10 GB de consumo por ejecución
- A pesar de las dificultades en la carga de datos y las limitaciones, pandas sigue siendo la opción usada para estas tareas
- Si no queremos exceder la RAM, es factible usar cualquiera que no sea pandas, ya que disminuye la cantidad recursos consumidos operativos