Laboratorio No 2

Marco David Suarez Berdugo

Elías Buitrago Bolivar

Seminario Big Data y Gestión de Datos

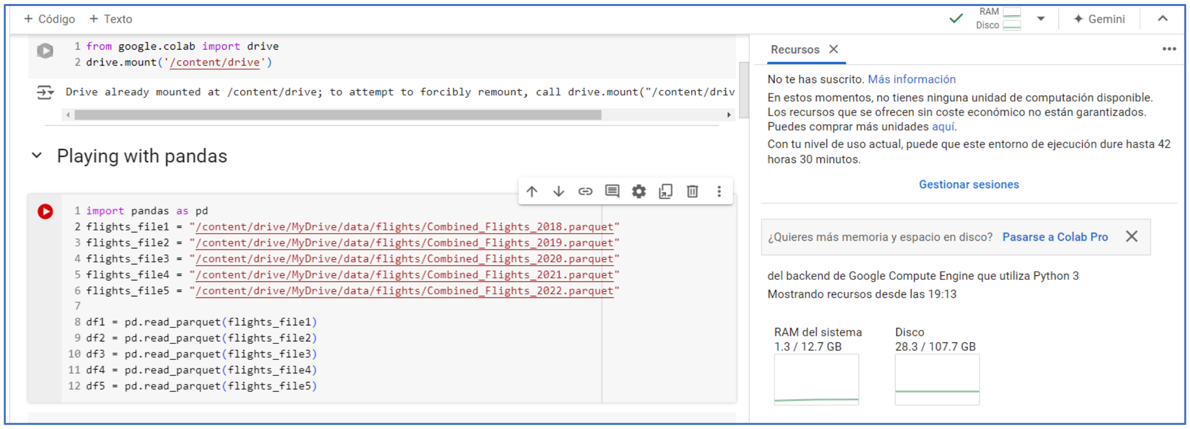
Universidad ECCI

Julio del 2024

Desarrollo

PANDAS

Con pandas se intenta cargar los 5 archivos resultando en un reinicio del servidor por consumo completo de la ram



```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/driv
```

Playing with pandas

```python
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)
```

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 30 minutos.

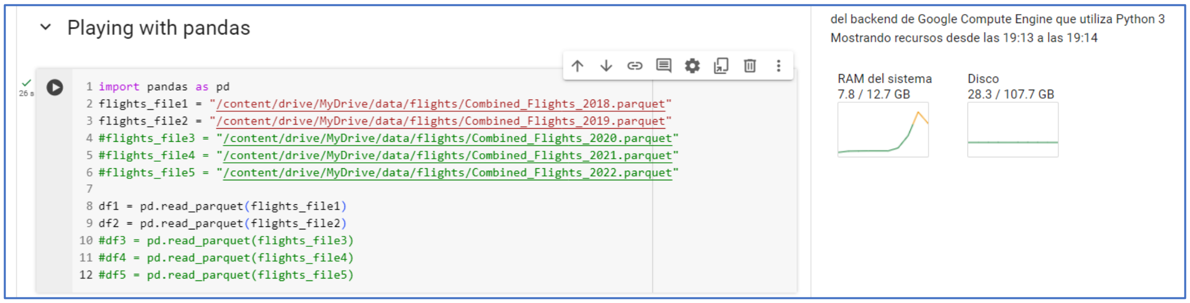
¿Quieres más memoria y espacio en disco? Pasarse a Colab Pro ✕

del backend de Google Compute Engine que utiliza Python 3
Mostrando recursos desde las 19:13

RAM del sistema
1.3 / 12.7 GB

Disco
28.3 / 107.7 GB

Se toma solo dos archivos de los 5, quedando procesados.



Playing with pandas

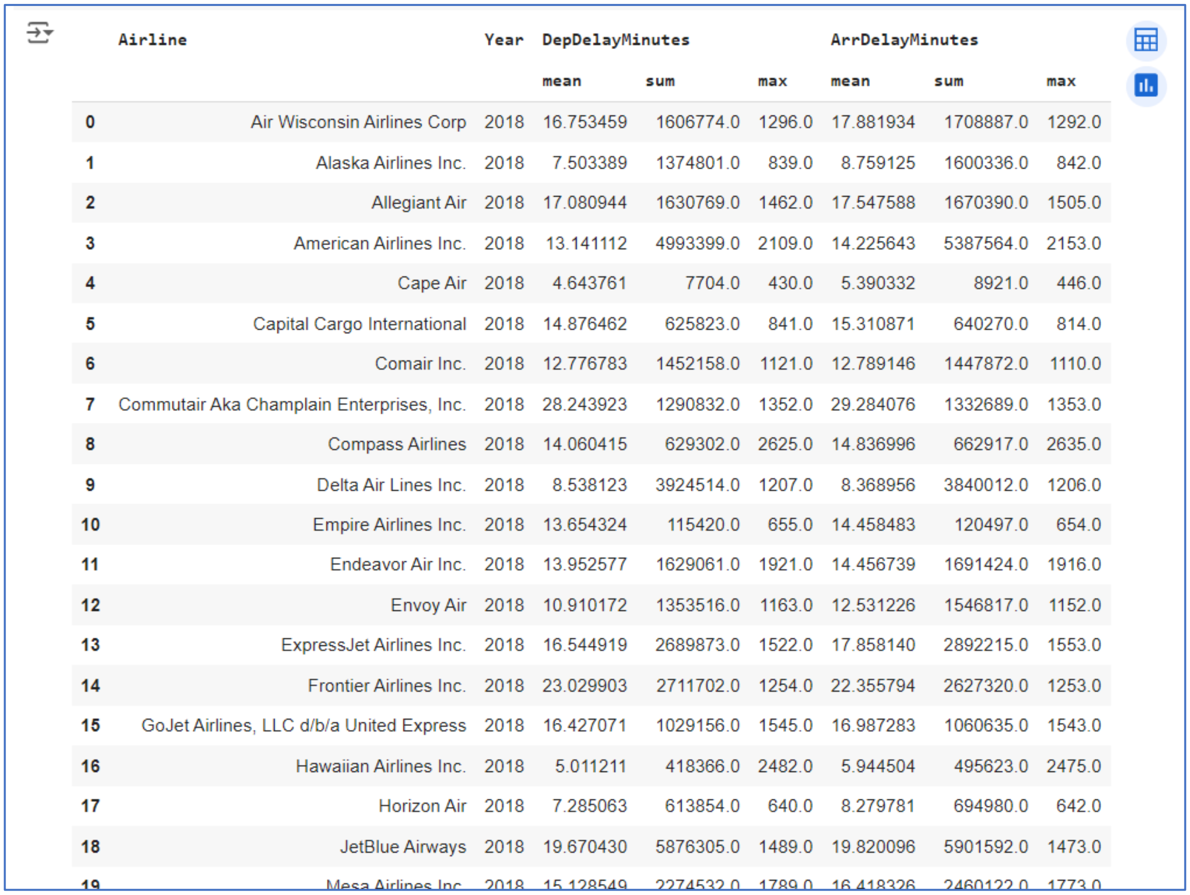
```python
import pandas as pd
flights_file1 = "/content/drive/MyDrive/data/flights/Combined_Flights_2018.parquet"
flights_file2 = "/content/drive/MyDrive/data/flights/Combined_Flights_2019.parquet"
#flights_file3 = "/content/drive/MyDrive/data/flights/Combined_Flights_2020.parquet"
#flights_file4 = "/content/drive/MyDrive/data/flights/Combined_Flights_2021.parquet"
#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)
```

del backend de Google Compute Engine que utiliza Python 3
Mostrando recursos desde las 19:13 a las 19:14

RAM del sistema
7.8 / 12.7 GB

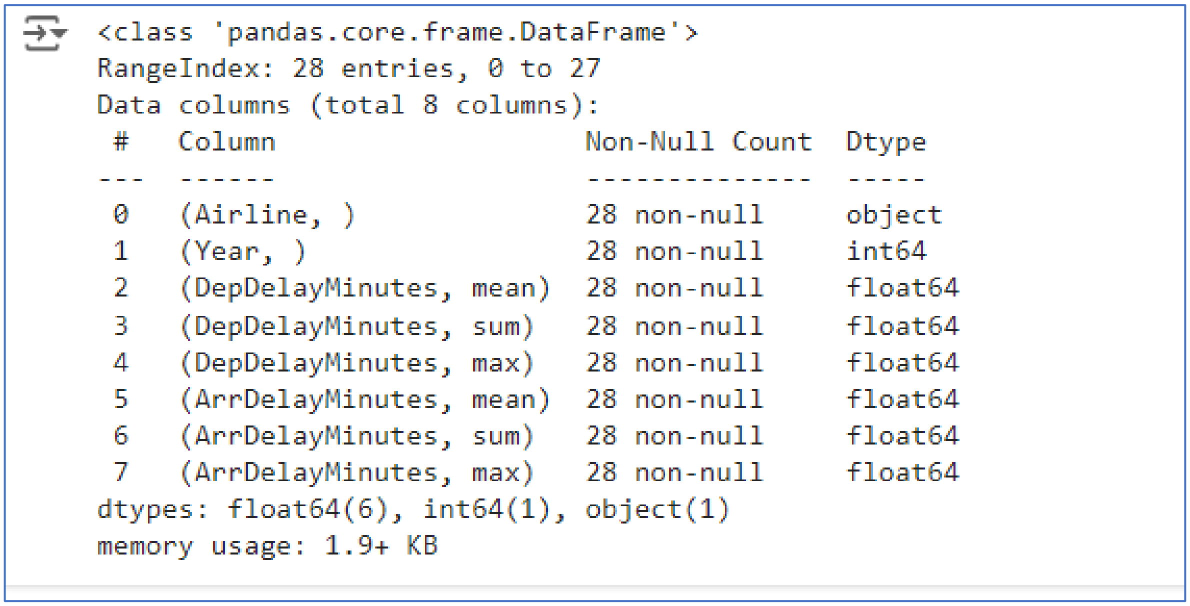
Disco
28.3 / 107.7 GB

Se obtiene la siguiente tabla:



	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	Alaska Airlines Inc.	2018	7.503389	1374801.0	839.0	8.759125	1600336.0	842.0
2	Allegiant Air	2018	17.080944	1630769.0	1462.0	17.547588	1670390.0	1505.0
3	American Airlines Inc.	2018	13.141112	4993399.0	2109.0	14.225643	5387564.0	2153.0
4	Cape Air	2018	4.643761	7704.0	430.0	5.390332	8921.0	446.0
5	Capital Cargo International	2018	14.876462	625823.0	841.0	15.310871	640270.0	814.0
6	Comair Inc.	2018	12.776783	1452158.0	1121.0	12.789146	1447872.0	1110.0
7	Commutair Aka Champlain Enterprises, Inc.	2018	28.243923	1290832.0	1352.0	29.284076	1332689.0	1353.0
8	Compass Airlines	2018	14.060415	629302.0	2625.0	14.836996	662917.0	2635.0
9	Delta Air Lines Inc.	2018	8.538123	3924514.0	1207.0	8.368956	3840012.0	1206.0
10	Empire Airlines Inc.	2018	13.654324	115420.0	655.0	14.458483	120497.0	654.0
11	Endeavor Air Inc.	2018	13.952577	1629061.0	1921.0	14.456739	1691424.0	1916.0
12	Envoy Air	2018	10.910172	1353516.0	1163.0	12.531226	1546817.0	1152.0
13	ExpressJet Airlines Inc.	2018	16.544919	2689873.0	1522.0	17.858140	2892215.0	1553.0
14	Frontier Airlines Inc.	2018	23.029903	2711702.0	1254.0	22.355794	2627320.0	1253.0
15	GoJet Airlines, LLC d/b/a United Express	2018	16.427071	1029156.0	1545.0	16.987283	1060635.0	1543.0
16	Hawaiian Airlines Inc.	2018	5.011211	418366.0	2482.0	5.944504	495623.0	2475.0
17	Horizon Air	2018	7.285063	613854.0	640.0	8.279781	694980.0	642.0
18	JetBlue Airways	2018	19.670430	5876305.0	1489.0	19.820096	5901592.0	1473.0
19	Mesa Airlines Inc.	2018	15.128549	2274532.0	1789.0	16.418326	2460122.0	1773.0

Con la siguiente información:

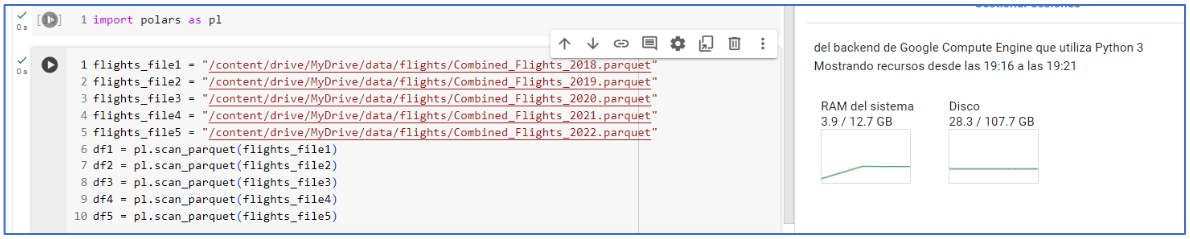


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28 entries, 0 to 27
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   (Airline, )                  28 non-null     object 
 1   (Year, )                     28 non-null     int64  
 2   (DepDelayMinutes, mean)      28 non-null     float64
 3   (DepDelayMinutes, sum)       28 non-null     float64
 4   (DepDelayMinutes, max)       28 non-null     float64
 5   (ArrDelayMinutes, mean)      28 non-null     float64
 6   (ArrDelayMinutes, sum)       28 non-null     float64
 7   (ArrDelayMinutes, max)       28 non-null     float64
dtypes: float64(6), int64(1), object(1)
memory usage: 1.9+ KB
```

Y el tiempo de ejecución fue:

POLARS

Se ejecutan los 5 archivos obteniendo que se cargan:



```
import polars as pl
```

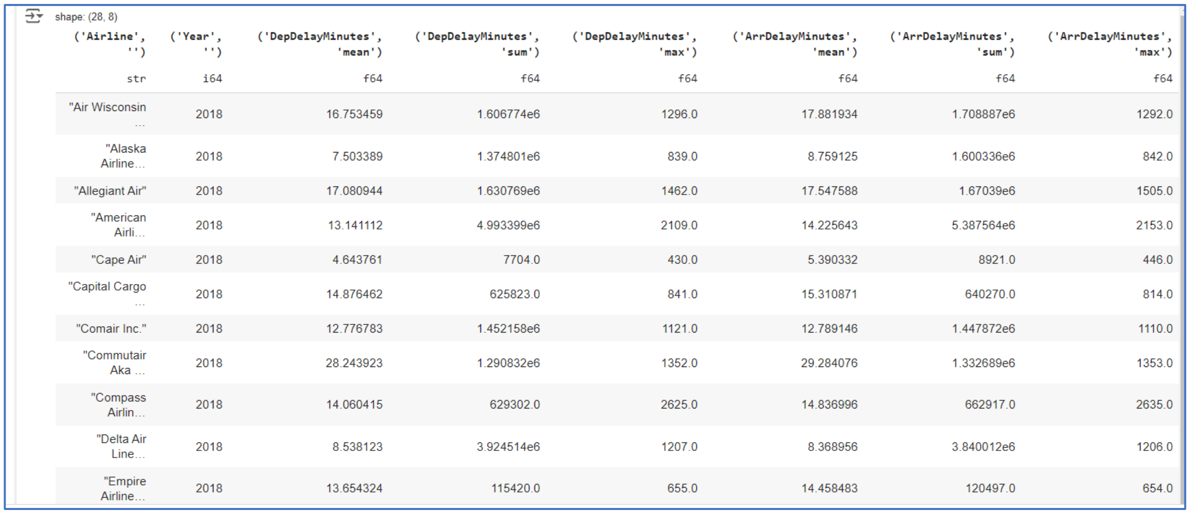
```
flights_file1 = "/content/drive/MyDrive/data/flights/Combined_Flights_2018.parquet"
flights_file2 = "/content/drive/MyDrive/data/flights/Combined_Flights_2019.parquet"
flights_file3 = "/content/drive/MyDrive/data/flights/Combined_Flights_2020.parquet"
flights_file4 = "/content/drive/MyDrive/data/flights/Combined_Flights_2021.parquet"
flights_file5 = "/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_file5)
```

del backend de Google Compute Engine que utiliza Python 3
Mostrando recursos desde las 19:16 a las 19:21

RAM del sistema
3.9 / 12.7 GB

Disco
28.3 / 107.7 GB

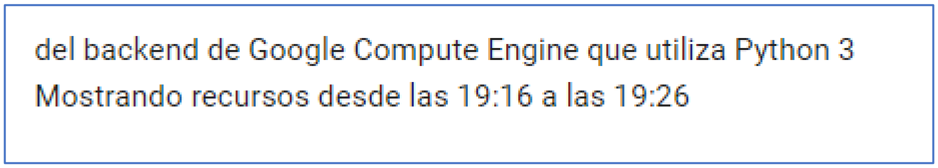
Obteniendo la siguiente tabla:



shape: (28, 8)

('Airline', '')	('Year', '')	('DepDelayMinutes', 'mean')	('DepDelayMinutes', 'sum')	('DepDelayMinutes', 'max')	('ArrDelayMinutes', 'mean')	('ArrDelayMinutes', 'sum')	('ArrDelayMinutes', 'max')
str	i64	f64	f64	f64	f64	f64	f64
"Air Wisconsin …	2018	16.753459	1.606774e6	1296.0	17.881934	1.708887e6	1292.0
"Alaska Airline…	2018	7.503389	1.374801e6	839.0	8.759125	1.600336e6	842.0
"Allegiant Air"	2018	17.080944	1.630769e6	1462.0	17.547588	1.67039e6	1505.0
"American Airli…	2018	13.141112	4.993399e6	2109.0	14.225643	5.387564e6	2153.0
"Cape Air"	2018	4.643761	7704.0	430.0	5.390332	8921.0	446.0
"Capital Cargo …	2018	14.876462	625823.0	841.0	15.310871	640270.0	814.0
"Comair Inc."	2018	12.776783	1.452158e6	1121.0	12.789146	1.447872e6	1110.0
"Commutair Aka …	2018	28.243923	1.290832e6	1352.0	29.284076	1.332689e6	1353.0
"Compass Airlin…	2018	14.060415	629302.0	2625.0	14.836996	662917.0	2635.0
"Delta Air Line…	2018	8.538123	3.924514e6	1207.0	8.368956	3.840012e6	1206.0
"Empire Airline…	2018	13.654324	115420.0	655.0	14.458483	120497.0	654.0

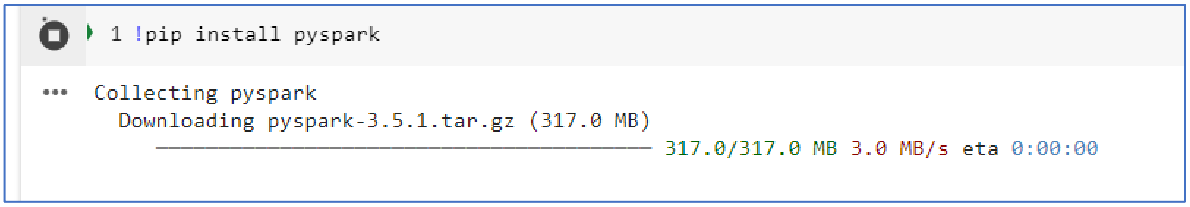
Y el tiempo de ejecución fue:



del backend de Google Compute Engine que utiliza Python 3
Mostrando recursos desde las 19:16 a las 19:26

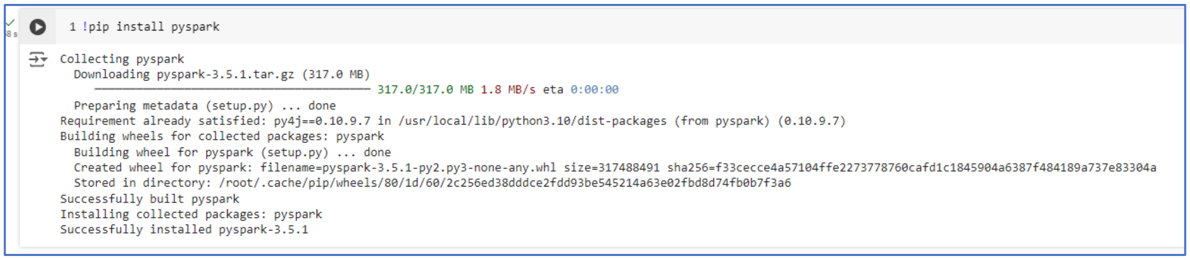
PYSPARK

Se instala pyspark



```
!pip install pyspark
```

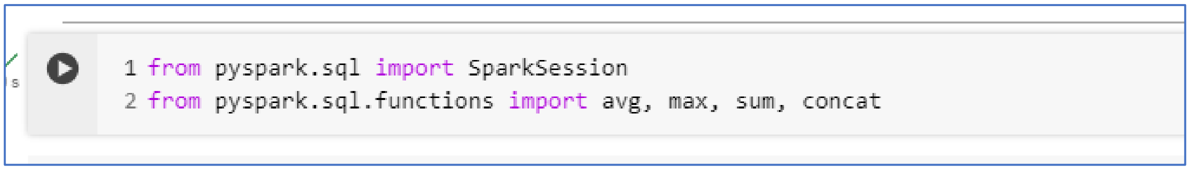
```
Collecting pyspark
  Downloading pyspark-3.5.1.tar.gz (317.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 317.0/317.0 MB 3.0 MB/s eta 0:00:00
```



```
!pip install pyspark
```

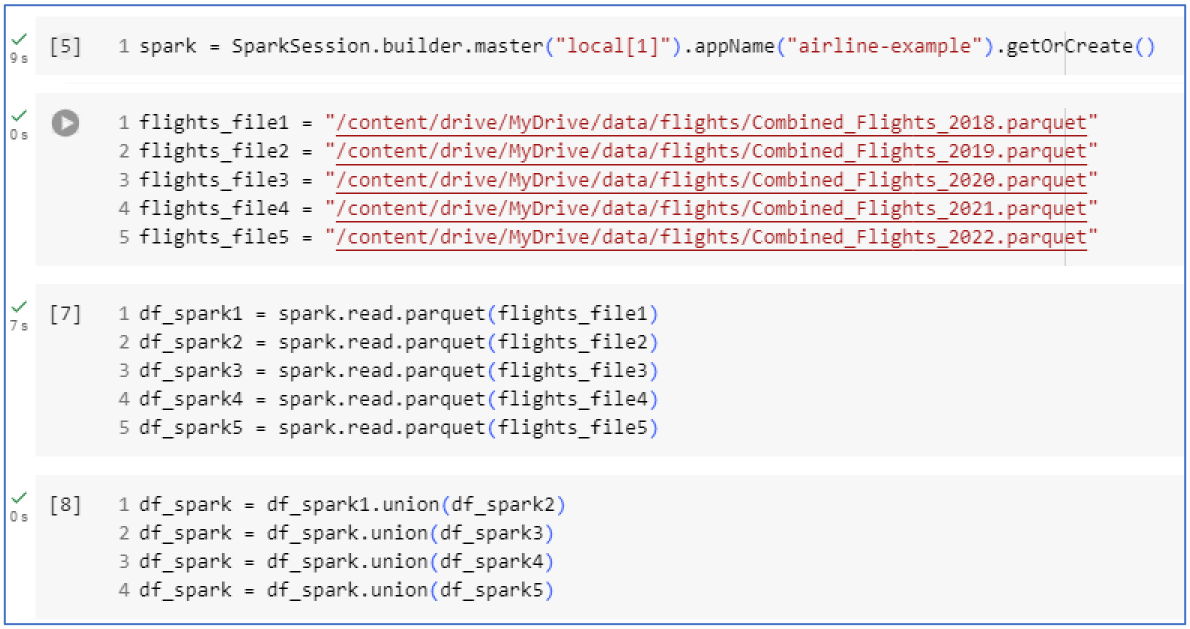
```
Collecting pyspark
  Downloading pyspark-3.5.1.tar.gz (317.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 317.0/317.0 MB 1.8 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
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=f33cecce4a57104ffe2273778760cafd1c1845904a6387f484189a737e83304a
  Stored in directory: /root/.cache/pip/wheels/80/1d/60/2c256ed38dddce2fdd93be545214a63e02fbd8d74fb0b7f3a6
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.1
```

Uso de librerías:



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

Se cargan los archivos:



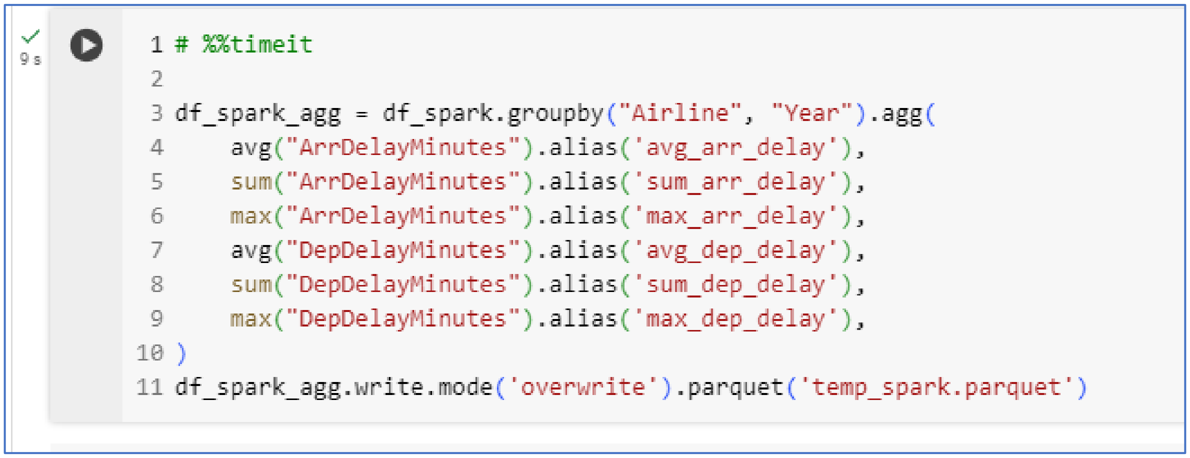
```python
spark = SparkSession.builder.master("local[1]").appName("airline-example").getOrCreate()
```

```python
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"
```

```python
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)
```

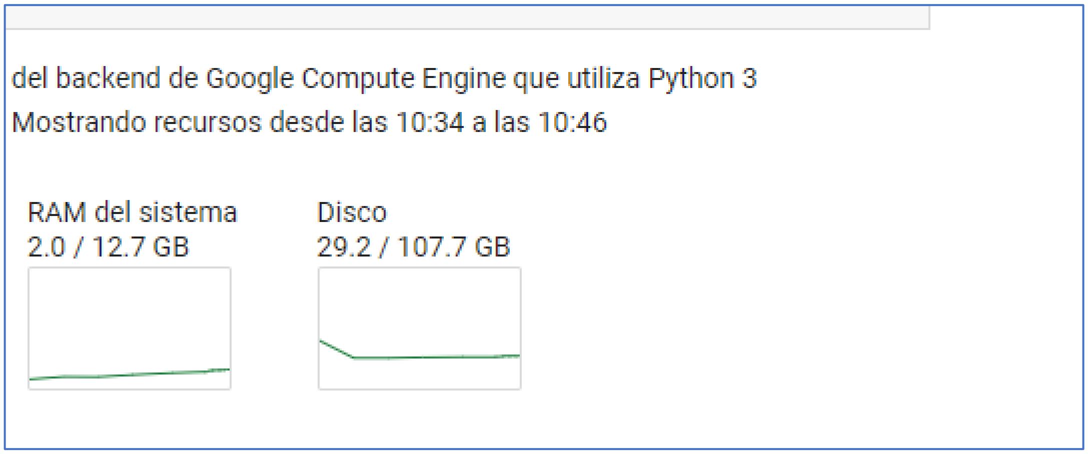
```python
df_spark = df_spark1.union(df_spark2)
df_spark = df_spark.union(df_spark3)
df_spark = df_spark.union(df_spark4)
df_spark = df_spark.union(df_spark5)
```

Se realizan los cálculos



```python
# %%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')
```

No se evidencia consumo completo de recursos



del backend de Google Compute Engine que utiliza Python 3

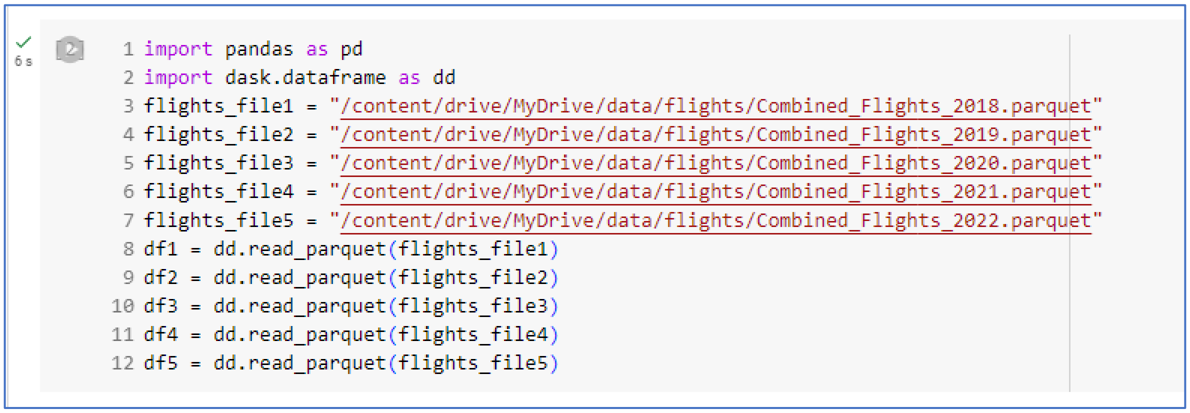
Mostrando recursos desde las 10:34 a las 10:46

RAM del sistema
2.0 / 12.7 GB

Disco
29.2 / 107.7 GB

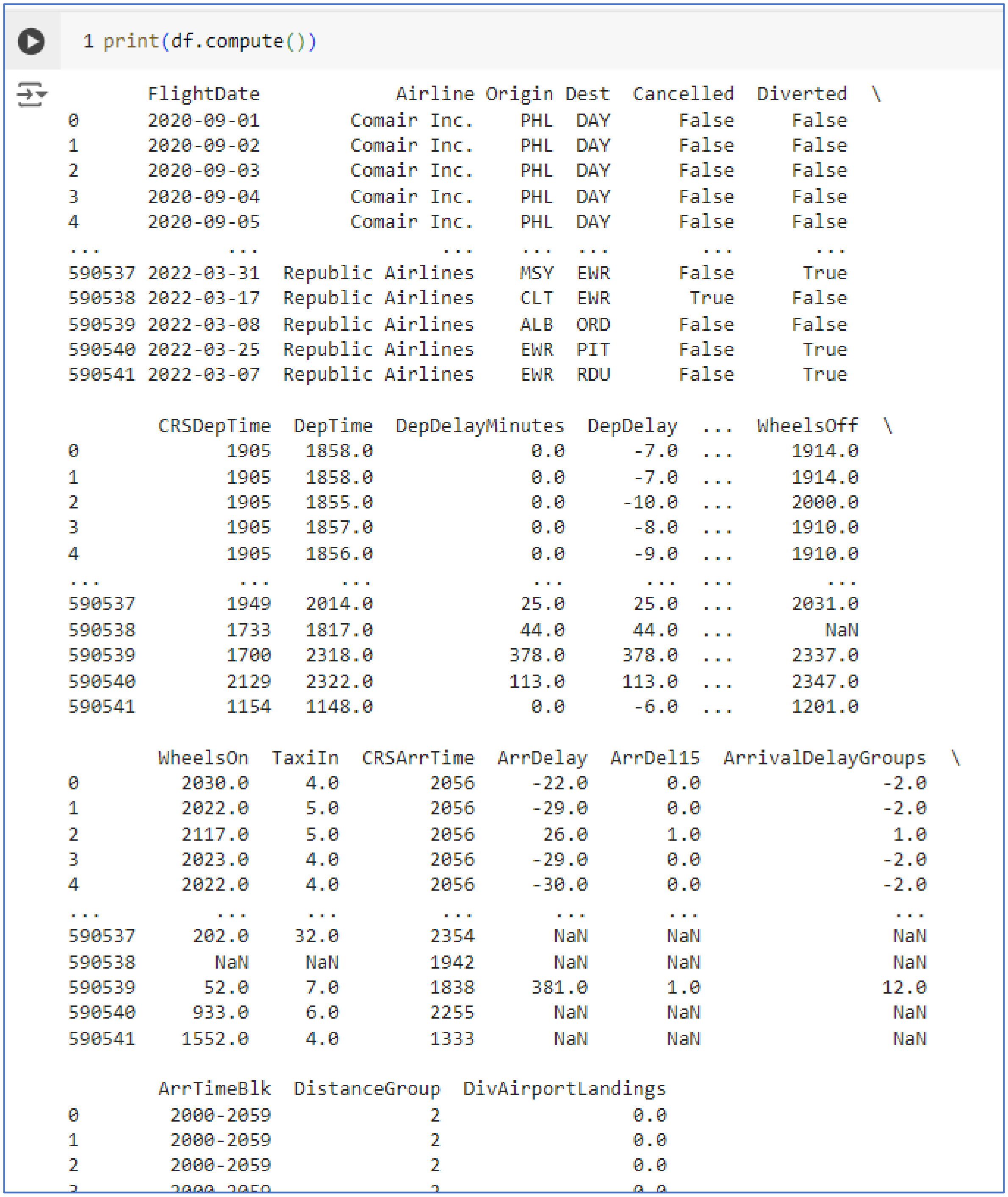
DASK

Se importan las librerías



```python
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)
```

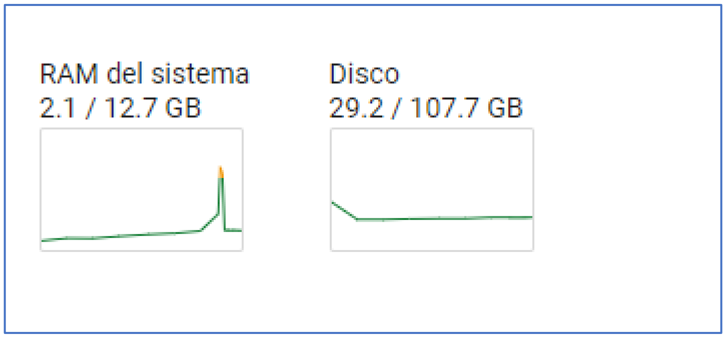
Validación de archivos



```
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    MSY  EWR      False      True   
590538   2022-03-17  Republic Airlines    CLT  EWR       True     False   
590539   2022-03-08  Republic Airlines    ALB  ORD      False     False   
590540   2022-03-25  Republic Airlines    EWR  PIT      False      True   
590541   2022-03-07  Republic Airlines    EWR  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  ...     1910.0   
4              1905   1856.0              0.0      -9.0  ...     1910.0   
...             ...      ...              ...       ...  ...        ...   
590537         1949   2014.0             25.0      25.0  ...     2031.0   
590538         1733   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          2030.0     4.0        2056     -22.0       0.0                -2.0   
1          2022.0     5.0        2056     -29.0       0.0                -2.0   
2          2117.0     5.0        2056      26.0       1.0                 1.0   
3          2023.0     4.0        2056     -29.0       0.0                -2.0   
4          2022.0     4.0        2056     -30.0       0.0                -2.0   
...           ...     ...         ...       ...       ...                 ...   
590537      202.0    32.0        2354       NaN       NaN                 NaN   
590538        NaN     NaN        1942       NaN       NaN                 NaN   
590539       52.0     7.0        1838     381.0       1.0                12.0   
590540      933.0     6.0        2255       NaN       NaN                 NaN   
590541     1552.0     4.0        1333       NaN       NaN                 NaN   

         ArrTimeBlk  DistanceGroup  DivAirportLandings  
0         2000-2059              2                 0.0  
1         2000-2059              2                 0.0  
2         2000-2059              2                 0.0  
3         2000-2059              2                 0.0  
```

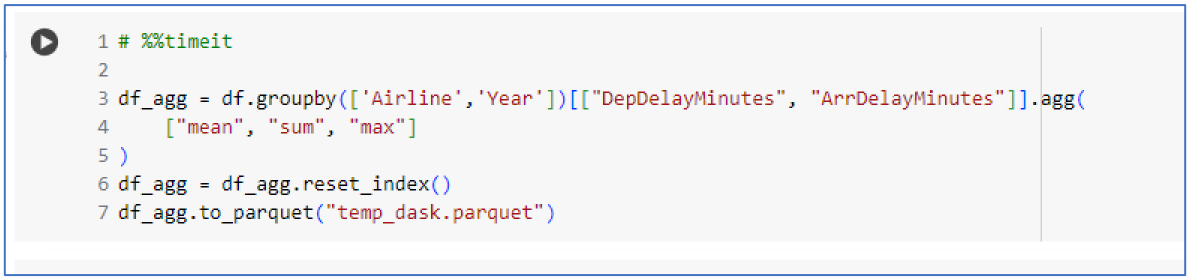
Vemos que se acerca a consumir la mayoría de recurso de ram



RAM del sistema
2.1 / 12.7 GB

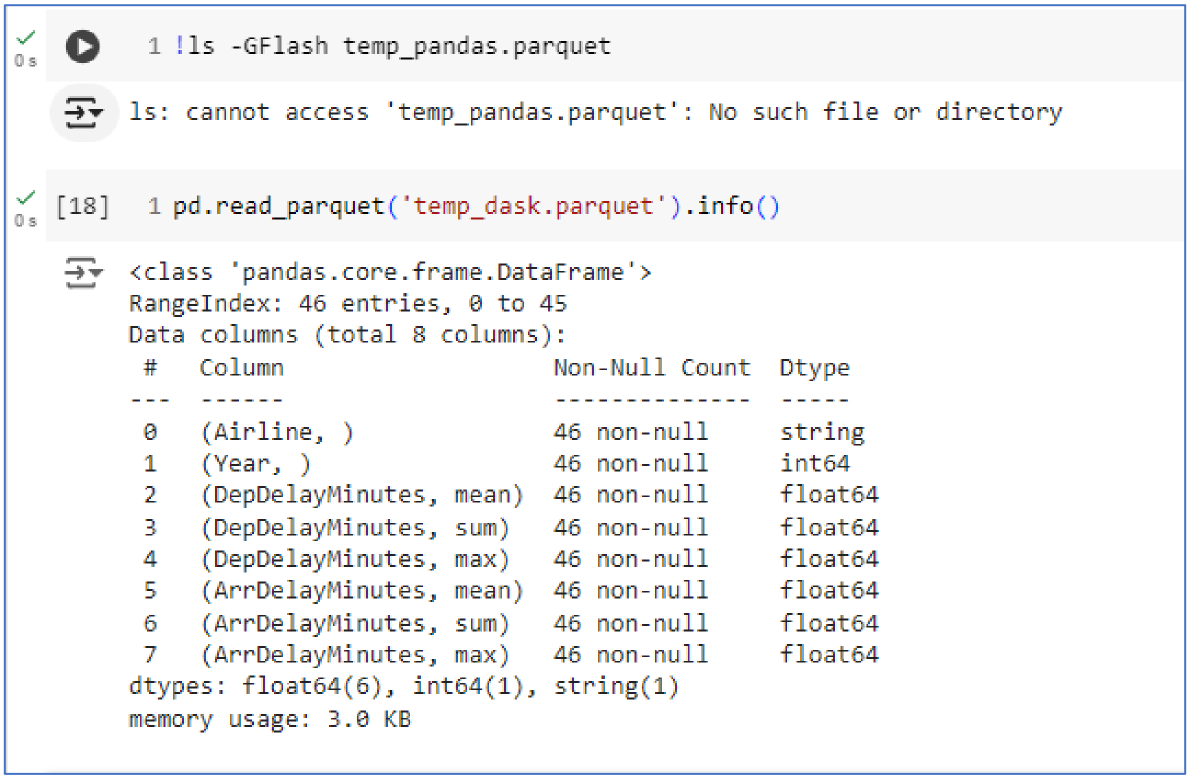
Disco
29.2 / 107.7 GB

Se realizan los agrupamientos



```python
# %%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")
```

Se carga el parquet



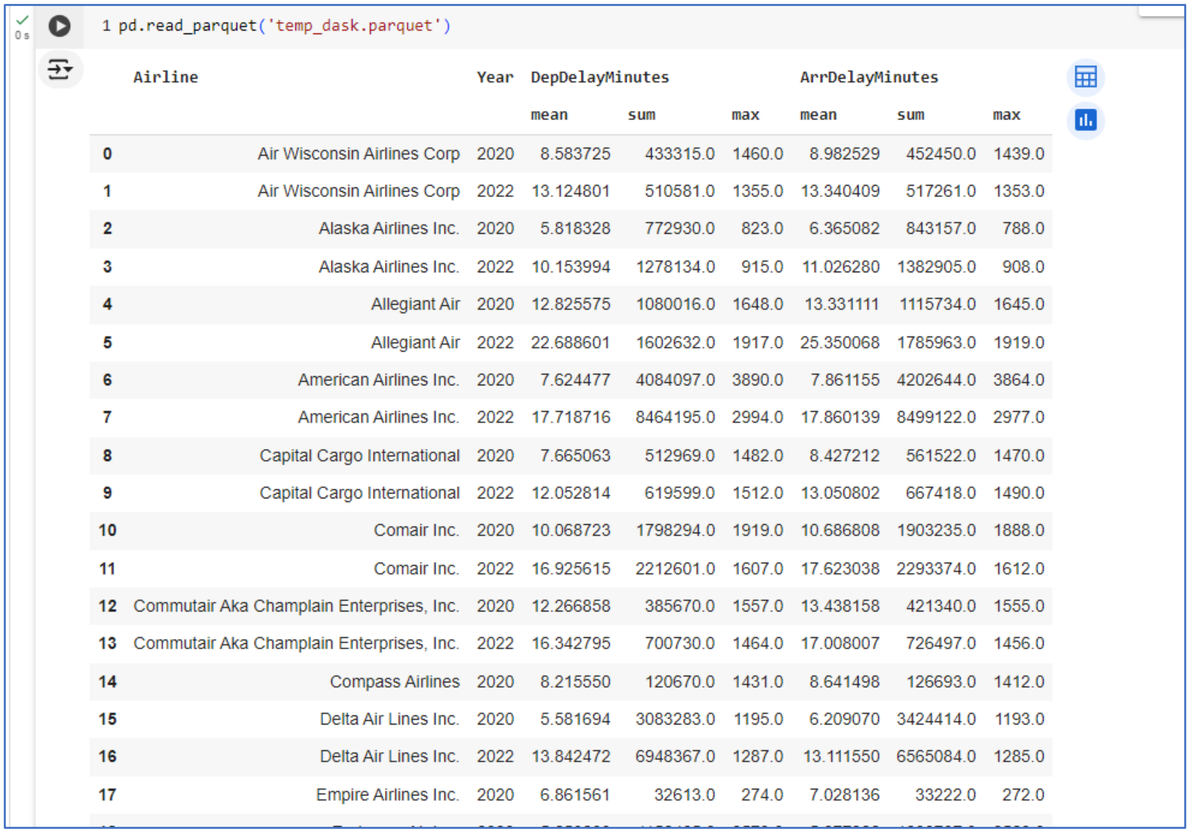
```
!ls -GFlash temp_pandas.parquet
```

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

```
[18] pd.read_parquet('temp_dask.parquet').info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46 entries, 0 to 45
Data columns (total 8 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   (Airline, )             46 non-null     string 
 1   (Year, )                46 non-null     int64  
 2   (DepDelayMinutes, mean) 46 non-null     float64
 3   (DepDelayMinutes, sum)  46 non-null     float64
 4   (DepDelayMinutes, max)  46 non-null     float64
 5   (ArrDelayMinutes, mean) 46 non-null     float64
 6   (ArrDelayMinutes, sum)  46 non-null     float64
 7   (ArrDelayMinutes, max)  46 non-null     float64
dtypes: float64(6), int64(1), string(1)
memory usage: 3.0 KB
```

Obteniendo lo siguiente

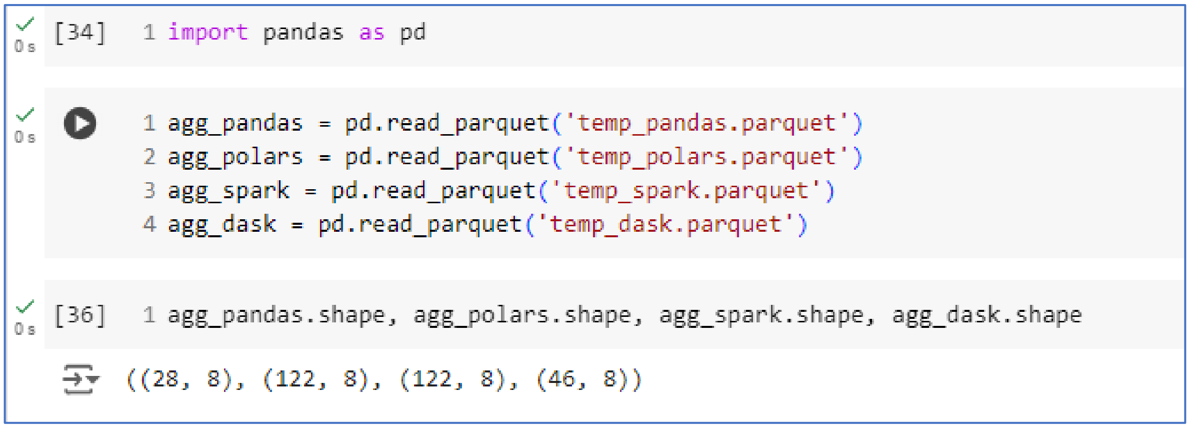


```python
pd.read_parquet('temp_dask.parquet')
```

	Airline	Year	DepDelayMinutes			ArrDelayMinutes		
			mean	sum	max	mean	sum	max
0	Air Wisconsin Airlines Corp	2020	8.583725	433315.0	1460.0	8.982529	452450.0	1439.0
1	Air Wisconsin Airlines Corp	2022	13.124801	510581.0	1355.0	13.340409	517261.0	1353.0
2	Alaska Airlines Inc.	2020	5.818328	772930.0	823.0	6.365082	843157.0	788.0
3	Alaska Airlines Inc.	2022	10.153994	1278134.0	915.0	11.026280	1382905.0	908.0
4	Allegiant Air	2020	12.825575	1080016.0	1648.0	13.331111	1115734.0	1645.0
5	Allegiant Air	2022	22.688601	1602632.0	1917.0	25.350068	1785963.0	1919.0
6	American Airlines Inc.	2020	7.624477	4084097.0	3890.0	7.861155	4202644.0	3864.0
7	American Airlines Inc.	2022	17.718716	8464195.0	2994.0	17.860139	8499122.0	2977.0
8	Capital Cargo International	2020	7.665063	512969.0	1482.0	8.427212	561522.0	1470.0
9	Capital Cargo International	2022	12.052814	619599.0	1512.0	13.050802	667418.0	1490.0
10	Comair Inc.	2020	10.068723	1798294.0	1919.0	10.686808	1903235.0	1888.0
11	Comair Inc.	2022	16.925615	2212601.0	1607.0	17.623038	2293374.0	1612.0
12	Commutair Aka Champlain Enterprises, Inc.	2020	12.266858	385670.0	1557.0	13.438158	421340.0	1555.0
13	Commutair Aka Champlain Enterprises, Inc.	2022	16.342795	700730.0	1464.0	17.008007	726497.0	1456.0
14	Compass Airlines	2020	8.215550	120670.0	1431.0	8.641498	126693.0	1412.0
15	Delta Air Lines Inc.	2020	5.581694	3083283.0	1195.0	6.209070	3424414.0	1193.0
16	Delta Air Lines Inc.	2022	13.842472	6948367.0	1287.0	13.111550	6565084.0	1285.0
17	Empire Airlines Inc.	2020	6.861561	32613.0	274.0	7.028136	33222.0	272.0

Resultados

Se localizan los temporales de cada uno



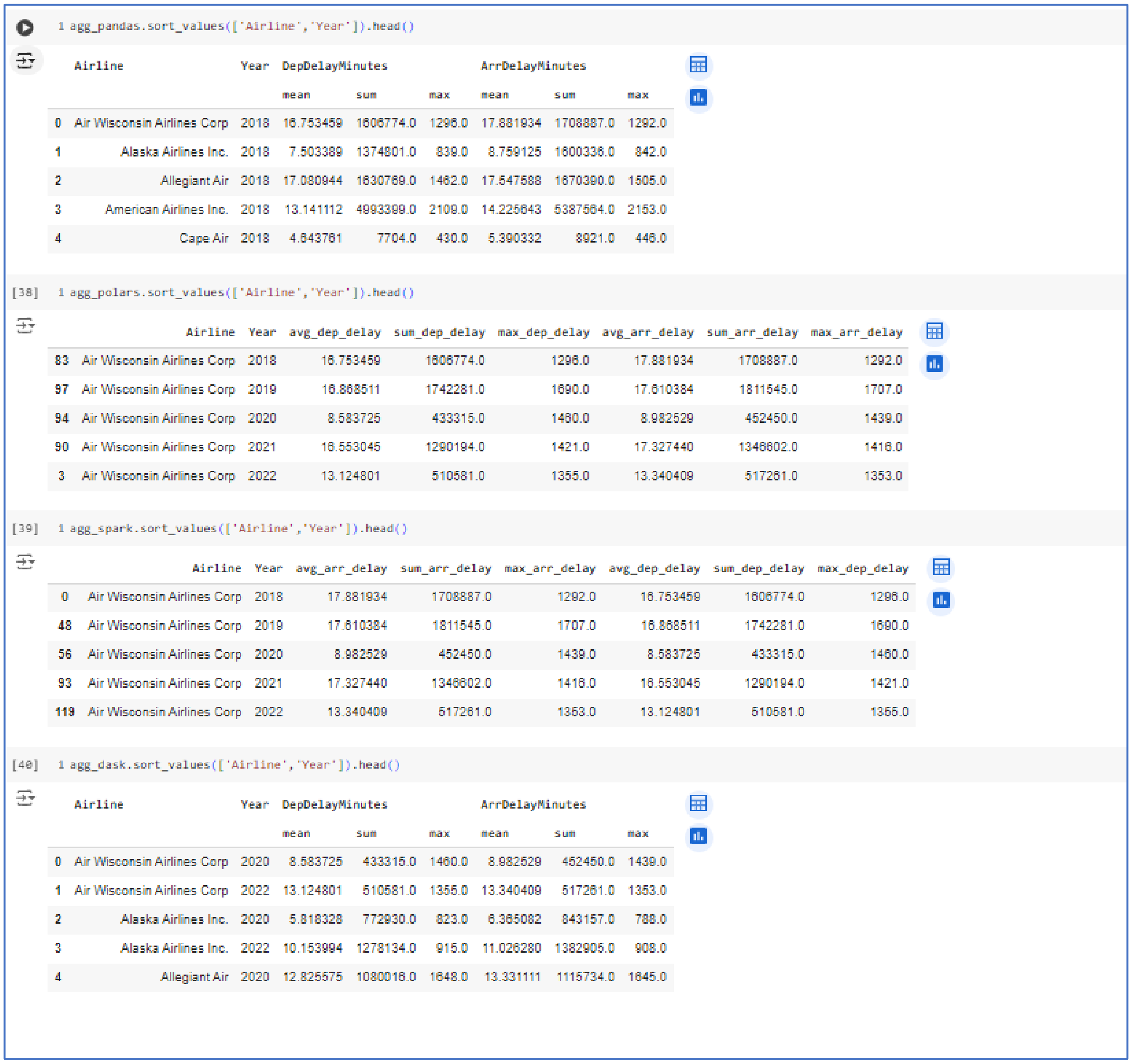
```python
[34]
import pandas as pd
```

```python
agg_pandas = pd.read_parquet('temp_pandas.parquet')
agg_polars = pd.read_parquet('temp_polars.parquet')
agg_spark = pd.read_parquet('temp_spark.parquet')
agg_dask = pd.read_parquet('temp_dask.parquet')
```

```python
[36]
agg_pandas.shape, agg_polars.shape, agg_spark.shape, agg_dask.shape
```

```
((28, 8), (122, 8), (122, 8), (46, 8))
```

Se muestran los resultados



```
agg_pandas.sort_values(['Airline','Year']).head()
```

	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	Alaska Airlines Inc.	2018	7.503389	1374801.0	839.0	8.759125	1600336.0	842.0
2	Allegiant Air	2018	17.080944	1630769.0	1462.0	17.547588	1670390.0	1505.0
3	American Airlines Inc.	2018	13.141112	4993399.0	2109.0	14.225643	5387564.0	2153.0
4	Cape Air	2018	4.643761	7704.0	430.0	5.390332	8921.0	446.0

[38]
```
agg_polars.sort_values(['Airline','Year']).head()
```

	Airline	Year	avg_dep_delay	sum_dep_delay	max_dep_delay	avg_arr_delay	sum_arr_delay	max_arr_delay
83	Air Wisconsin Airlines Corp	2018	16.753459	1606774.0	1296.0	17.881934	1708887.0	1292.0
97	Air Wisconsin Airlines Corp	2019	16.868511	1742281.0	1690.0	17.610384	1811545.0	1707.0
94	Air Wisconsin Airlines Corp	2020	8.583725	433315.0	1460.0	8.982529	452450.0	1439.0
90	Air Wisconsin Airlines Corp	2021	16.553045	1290194.0	1421.0	17.327440	1346602.0	1416.0
3	Air Wisconsin Airlines Corp	2022	13.124801	510581.0	1355.0	13.340409	517261.0	1353.0

[39]
```
agg_spark.sort_values(['Airline','Year']).head()
```

	Airline	Year	avg_arr_delay	sum_arr_delay	max_arr_delay	avg_dep_delay	sum_dep_delay	max_dep_delay
0	Air Wisconsin Airlines Corp	2018	17.881934	1708887.0	1292.0	16.753459	1606774.0	1296.0
48	Air Wisconsin Airlines Corp	2019	17.610384	1811545.0	1707.0	16.868511	1742281.0	1690.0
56	Air Wisconsin Airlines Corp	2020	8.982529	452450.0	1439.0	8.583725	433315.0	1460.0
93	Air Wisconsin Airlines Corp	2021	17.327440	1346602.0	1416.0	16.553045	1290194.0	1421.0
119	Air Wisconsin Airlines Corp	2022	13.340409	517261.0	1353.0	13.124801	510581.0	1355.0

[40]
```
agg_dask.sort_values(['Airline','Year']).head()
```

	Airline	Year	DepDelayMinutes			ArrDelayMinutes		
			mean	sum	max	mean	sum	max
0	Air Wisconsin Airlines Corp	2020	8.583725	433315.0	1460.0	8.982529	452450.0	1439.0
1	Air Wisconsin Airlines Corp	2022	13.124801	510581.0	1355.0	13.340409	517261.0	1353.0
2	Alaska Airlines Inc.	2020	5.818328	772930.0	823.0	6.365082	843157.0	788.0
3	Alaska Airlines Inc.	2022	10.153994	1278134.0	915.0	11.026280	1382905.0	908.0
4	Allegiant Air	2020	12.825575	1080016.0	1648.0	13.331111	1115734.0	1645.0

Conclusiones

En un entorno con 12 GB de RAM y 100 GB de disco, Polars y Dask son las opciones más eficientes para manejar grandes volúmenes de datos y operaciones complejas. Polars destaca por su rendimiento en memoria y procesamiento paralelo, mientras que Dask ofrece flexibilidad y escalabilidad, permitiendo el manejo de datos que exceden la memoria disponible. Pandas sigue siendo útil para conjuntos de datos más pequeños y análisis rápidos, y PySpark podría ser considerado si se trabaja en un entorno distribuido o se planifica escalar a un clúster en el futuro.