Andrés Camilo Velásquez – 63111

Laboratorio comprensión de los datos

Objetivo Laboratorio

Comparar el desempeño de librerías de Python para carga y manipulación de datos tabulares.

Desarrollo Laboratorio

Librería Pandas

Primero comenzamos ejecutando la librería pandas con todos los paquetes y notamos que ocurre una saturación en la RAM al rededor del minuto 1:30.

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

```
Mounted at /content/drive
```

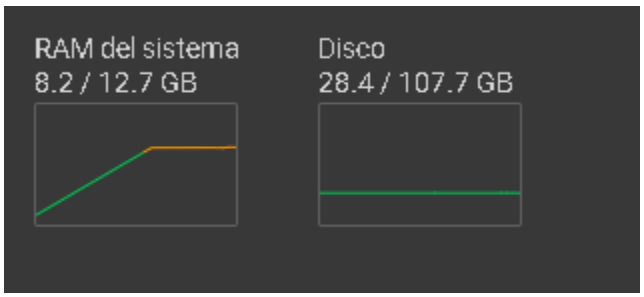
```
import pandas as pd
flights_file1 = "/content/drive/MyDrive/Combined_Flights_2018.parquet"
flights_file2 = "/content/drive/MyDrive/Combined_Flights_2019.parquet"
flights_file3 = "/content/drive/MyDrive/Combined_Flights_2020.parquet"
flights_file4 = "/content/drive/MyDrive/Combined_Flights_2021.parquet"
flights_file5 = "/content/drive/MyDrive/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)
```

Tu sesión ha fallado porque se ha usado toda la memoria RAM disponible.

Realizando pruebas nos damos cuenta que la mejor manera de ejecutar los paquetes es con los 3 más ligeros y luego los dos más pesados.

- Primero con los 3 más ligeros notamos un máximo de 8.2 GB de RAM en 24 segundos.

```python
import pandas as pd
flights_file1 = "/content/drive/MyDrive/Combined_Flights_2018.parquet"
#flights_file2 = "/content/drive/MyDrive/Combined_Flights_2019.parquet"
flights_file3 = "/content/drive/MyDrive/Combined_Flights_2020.parquet"
#flights_file4 = "/content/drive/MyDrive/Combined_Flights_2021.parquet"
flights_file5 = "/content/drive/MyDrive/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)
```




```python
[2] # df = pd.concat([df3, df5])
    df = df3
```

```python
[3] # %%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")
```

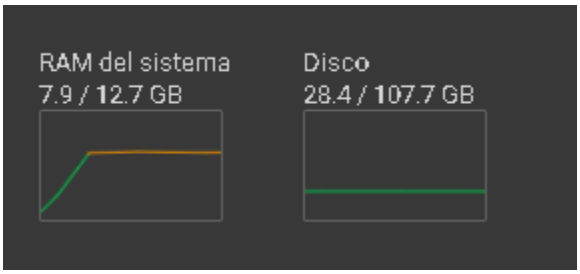
```python
[4] !ls -GFlash temp_pandas.parquet
```

```
12K -rw-r--r-- 1 root 9.0K Jul 17 17:18 temp_pandas.parquet
```

```python
[5] pd.read_parquet('temp_pandas.parquet')
```

- Segundo con los 2 paquetes más pesados notamos un máximo de 7.9 GB de RAM en 32 segundos, tomando mas tiempo incluso que con los 3 paquetes.

```
import pandas as pd
#flights_file1 = "/content/drive/MyDrive/Combined_Flights_2018.parquet"
flights_file2 = "/content/drive/MyDrive/Combined_Flights_2019.parquet"
#flights_file3 = "/content/drive/MyDrive/Combined_Flights_2020.parquet"
flights_file4 = "/content/drive/MyDrive/Combined_Flights_2021.parquet"
#flights_file5 = "/content/drive/MyDrive/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)
```



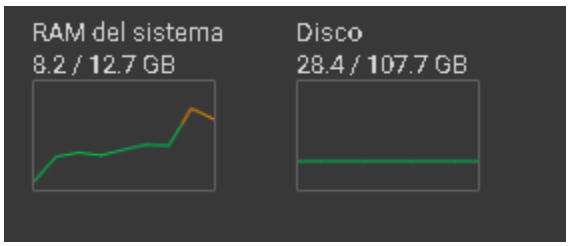
```python
pd.read_parquet('temp_pandas.parquet')
```

	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	Alaska Airlines Inc.	2020	5.818328	772930.0	823.0	6.365082	843157.0	788.0
2	Allegiant Air	2020	12.825575	1080016.0	1648.0	13.331111	1115734.0	1645.0
3	American Airlines Inc.	2020	7.624477	4084097.0	3890.0	7.861155	4202644.0	3864.0
4	Capital Cargo International	2020	7.665063	512969.0	1482.0	8.427212	561522.0	1470.0
5	Comair Inc.	2020	10.068723	1798294.0	1919.0	10.686808	1903235.0	1888.0
6	Commutair Aka Champlain Enterprises, Inc.	2020	12.266858	385670.0	1557.0	13.438158	421340.0	1555.0
7	Compass Airlines	2020	8.215550	120670.0	1431.0	8.641498	126693.0	1412.0
8	Delta Air Lines Inc.	2020	5.581694	3083283.0	1195.0	6.209070	3424414.0	1193.0
9	Empire Airlines Inc.	2020	6.861561	32613.0	274.0	7.028136	33222.0	272.0
10	Endeavor Air Inc.	2020	5.653603	1156405.0	2579.0	5.877866	1200707.0	2560.0
11	Envoy Air	2020	6.685444	1339382.0	2248.0	8.417206	1682490.0	2238.0
12	ExpressJet Airlines Inc.	2020	6.396004	308249.0	1357.0	7.443095	357797.0	1338.0
13	Frontier Airlines Inc.	2020	7.398021	640062.0	645.0	7.330406	633479.0	638.0
14	GoJet Airlines, LLC d/b/a United Express	2020	8.309550	304786.0	1408.0	8.336411	305196.0	1382.0
15	Hawaiian Airlines Inc.	2020	3.592000	137491.0	1484.0	4.092661	156487.0	1481.0
16	Horizon Air	2020	4.849894	445133.0	775.0	5.351232	489932.0	758.0
17	JetBlue Airways	2020	8.213825	1118838.0	1135.0	8.503231	1155334.0	1153.0
18	Mesa Airlines Inc.	2020	9.662425	1236887.0	1890.0	10.344427	1320704.0	1884.0
19	Republic Airlines	2020	5.224466	1093533.0	1352.0	5.824160	1216865.0	1381.0
20	SkyWest Airlines Inc.	2020	8.671988	4963057.0	2327.0	8.718970	4976457.0	2308.0
21	Southwest Airlines Co.	2020	4.116316	3637115.0	602.0	3.372148	2975631.0	597.0
22	Spirit Air Lines	2020	7.809889	1032194.0	1339.0	7.758486	1023290.0	1262.0
23	Trans States Airlines	2020	13.739364	235108.0	1390.0	14.784864	252008.0	1366.0
24	United Air Lines Inc.	2020	6.925499	1978518.0	1471.0	6.766355	1929514.0	1503.0

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 8 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   (Airline, )               25 non-null     object
 1   (Year, )                  25 non-null     int64
 2   (DepDelayMinutes, mean)   25 non-null     float64
 3   (DepDelayMinutes, sum)    25 non-null     float64
 4   (DepDelayMinutes, max)    25 non-null     float64
 5   (ArrDelayMinutes, mean)   25 non-null     float64
 6   (ArrDelayMinutes, sum)    25 non-null     float64
 7   (ArrDelayMinutes, max)    25 non-null     float64
dtypes: float64(6), int64(1), object(1)
memory usage: 1.7+ KB
```

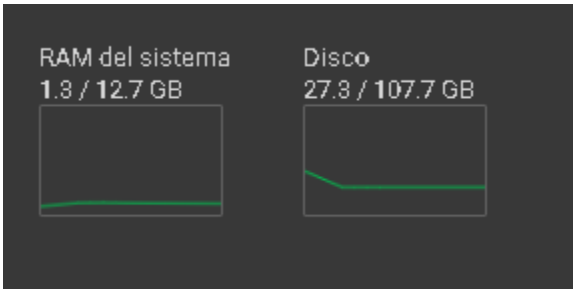
Y finalmente aumentando alrededor de 0.3 en Ram con el resto del código.



Librería Polars

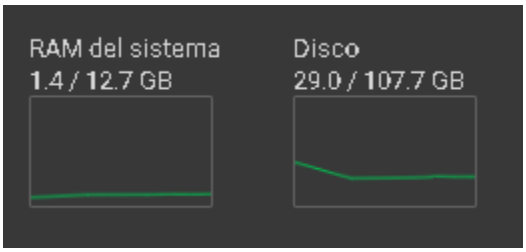
Primero comenzamos ejecutando la librería polars con todos los paquetes y notamos que ejecuta todos los paquetes en 2 segundos con un uso de ram de 1.3 GB.

```
[3] import polars as pl

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



Aunque notamos bastante tiempo en el resto del código con la librería de 1.32 segundos y un aumento de 0.1 GB de RAM.

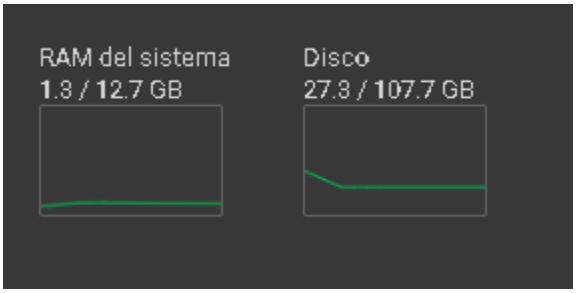
```
[5] %%timeit

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

    df_polars.write_parquet('temp_polars.parquet')
```



Librería Spark

Primero comenzamos ejecutando la librería polars con todos los paquetes y notamos que ejecuta todos los paquetes en menos de 1 segundo con un uso de ram de 1.3 GB.

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



Y con el resto del código ejecutando alrededor de 10 segundos con un aumento de 0.6 GB de RAM.

```
[11] 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)
```

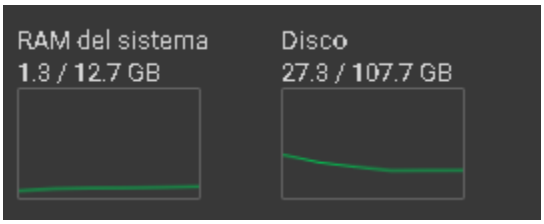
```
[12] df_spark = df_spark1.union(df_spark2)
     df_spark = df_spark.union(df_spark3)
     df_spark = df_spark.union(df_spark4)
     df_spark = df_spark.union(df_spark5)
```

```
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')
```

Librería Dask

Primero comenzamos ejecutando la librería polars con todos los paquetes y notamos que ejecuta todos los paquetes en 1 segundo con un uso de ram de 1.3 GB.

```
import pandas as pd
import dask.dataframe as dd
flights_file1 = "/content/drive/MyDrive/Combined_Flights_2018.parquet"
flights_file2 = "/content/drive/MyDrive/Combined_Flights_2019.parquet"
flights_file3 = "/content/drive/MyDrive/Combined_Flights_2020.parquet"
flights_file4 = "/content/drive/MyDrive/Combined_Flights_2021.parquet"
flights_file5 = "/content/drive/MyDrive/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)
```



Y con el resto del código ejecutando alrededor de 30 segundos con un aumento de 5.0 GB de RAM.

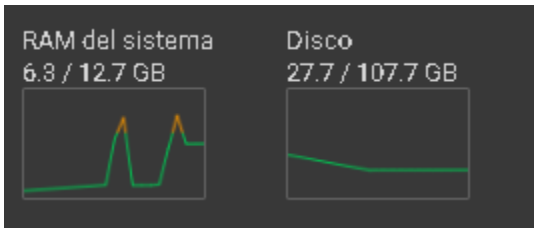
```
[5] df = dd.concat([df3, df5])

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
4        2000-2059              2                 0.0
...            ...            ...                 ...
590537   2300-2359              5                 1.0
```

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

```
[8] # %%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")
```

```
[9] !ls -GFlash temp_pandas.parquet

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

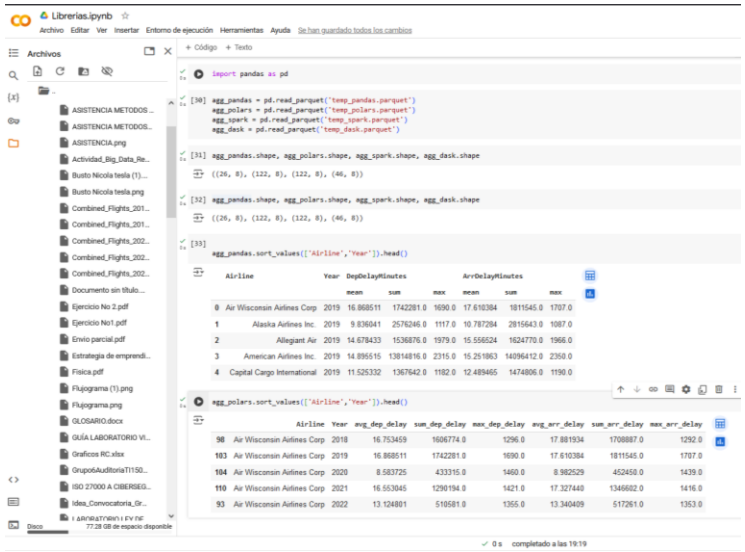
```
pd.read_parquet('temp_dask.parquet').info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 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
```

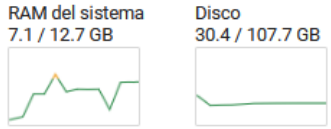



```
pd.read_parquet('temp_dask.parquet')
```

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
18	Endeavor Air Inc.	2020	5.653603	1156405.0	2579.0	5.877866	1200707.0	2560.0
19	Endeavor Air Inc.	2022	13.284602	1819220.0	1973.0	14.184149	1935952.0	1968.0

Final del código con las librerías





[33]
```
agg_pandas.sort_values(['Airline','Year']).head()
```

	Airline	Year	DepDelayMinutes			ArrDelayMinutes		
			mean	sum	max	mean	sum	max
0	Air Wisconsin Airlines Corp	2019	16.868511	1742281.0	1690.0	17.610384	1811545.0	1707.0
1	Alaska Airlines Inc.	2019	9.836041	2576246.0	1117.0	10.787284	2815643.0	1087.0
2	Allegiant Air	2019	14.678433	1536876.0	1979.0	15.556524	1624770.0	1966.0
3	American Airlines Inc.	2019	14.895515	13814816.0	2315.0	15.251863	14096412.0	2350.0
4	Capital Cargo International	2019	11.525332	1367642.0	1182.0	12.489465	1474806.0	1190.0

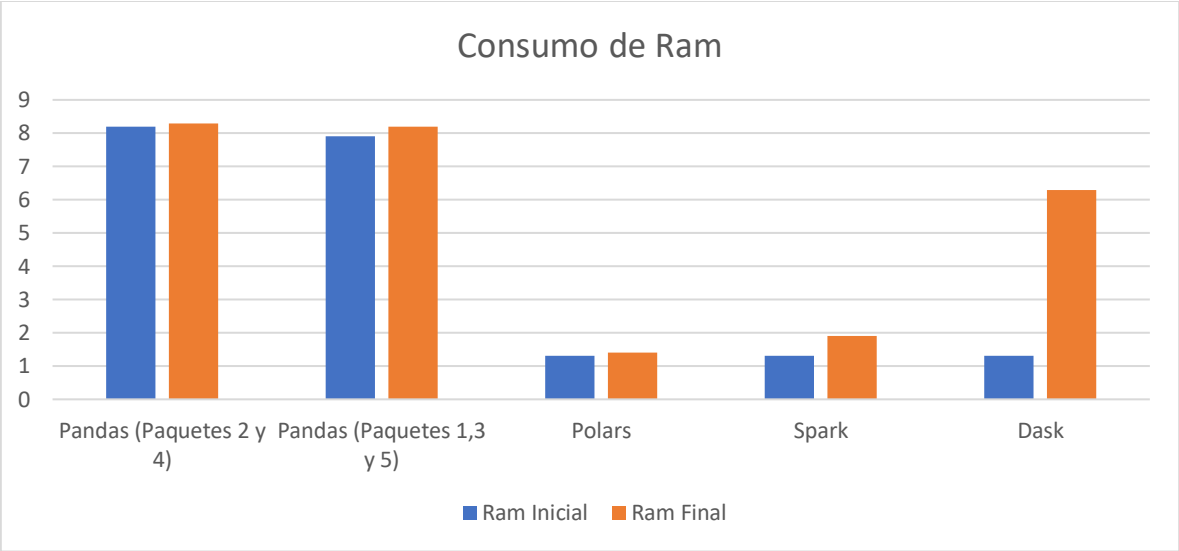
[34]
```
agg_polars.sort_values(['Airline','Year']).head()
```

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

```
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

Resultados



Conclusión

Luego de realizar las pruebas con las librerías de Pandas, Polars, Spark y Dask, Se concluye que:

- En cuanto carga de paquetes Polars, Spark son bastantes ligeras, cargan y ejecutan el código de manera rápida y efectiva, sin generar tanta carga de RAM.
- En cuanto a la librería Dask también es bastante rápida en la carga y ejecución, pero nos consume más RAM durante el proceso.
- En cuanto a la librería Pandas en cuanto a la carga de paquetes colapsaba la RAM, se realizó por partes para no saturar la RAM que nos provee Google Colabority, pero de todas maneras tiene un consumo de RAM bastante elevado a comparación de los demás.