# INFORME LABORATORIO 2

#### Gabriel Nicolas Aguilera Niño

## LIBRERÍA PANDAS

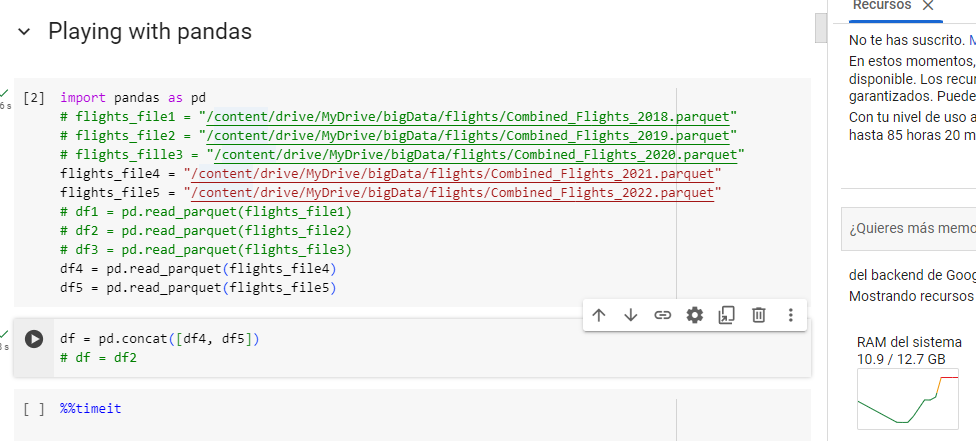
La idea es cargar la mayor cantidad de archivo y ver cuando puede consumir la RAM.

|  |  |  |
| --- | --- | --- |
| LIBRERÍA | ARCHIVOS CARGADOS | RAM UTILIZADA |
| Pandas Prueba 1 | Archivo 2 y 3 | 7.5 RAM / Concatenando Saturado |
| Pandas Prueba 2 | Archivo 4 y 5 | 6.1 RAM / Concatenando 10.9 RAM |
| Pandas Prueba 3 | Archivo 2 y 5 | 5.9 RAM / Concatenando 12.1 RAM |
| Pandas Prueba 4 | Archivos 3,4 y 5 | 10.8 RAM / Concatenando Saturado |
| Pandas Prueba 5 | Archivos 2,3 y 4 | Saturado Cargando los Archivos no se alcanza concatenar |
| Pandas Prueba 6 | Archivos 3 y 5 | 4.8 RAM / Concatenado 9.6 RAM |
| Pandas Prueba 7 | Archivo 2,3,4 y 5 | Saturado Cargando los Archivos no se alcanza concatenar |
| Pandas Prueba 8 | Archivos 1 y 2 | 7.9 RAM / Concatenando Saturado |
| Pandas Prueba 9 | Archivo 1 y 3 | 6.9 RAM / Concatenado 11.1 AM |

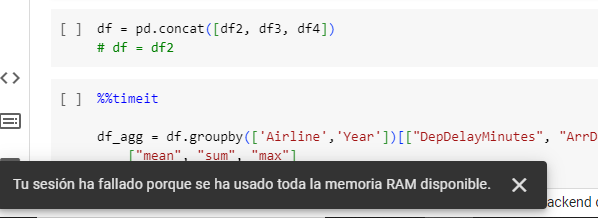
Hicimos varias pruebas con la librería de Pandas y podemos apreciar las siguientes conclusiones:

* Se pueden procesar de a dos (2) archivos dependiendo la cantidad de datos que exista en los mismos, como podemos apreciar en la prueba 6 se cargaron 2 archivos 3 y 5 el los cuales consumió 9.6 RAM en total.
* Si Cargamos de ha 3 archivos se satura completamente en la carga de los mismos y dependiendo el peso no se pueden Concatenar.
* Como se puede apreciar en la prueba 2 se cargaron 2 archivos, pero dependiendo la cantidad de datos ah procesar la maquina se saturo al momento de concatenar.

### Prueba 2



### Prueba 5

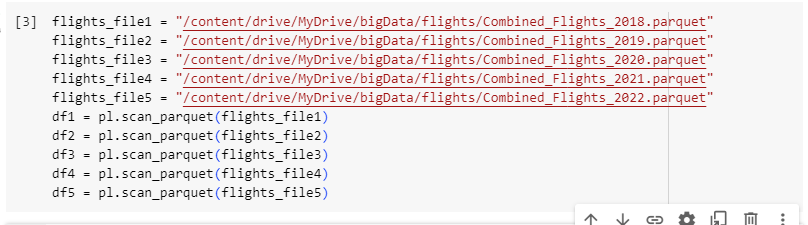


### Prueba 6



## LIBRERÍA POLARS

A pesar que con Polars gastamos mucho menos recursos con la maquina puede que sea un poco más complicado la agrupación y concatenación de los datos a diferencia de Pandas.



Con POLARS podemos procesar en 1 solo script los 5 archivos de datos sin problema alguno y solo utilizamos un total de 3.7 de RAM

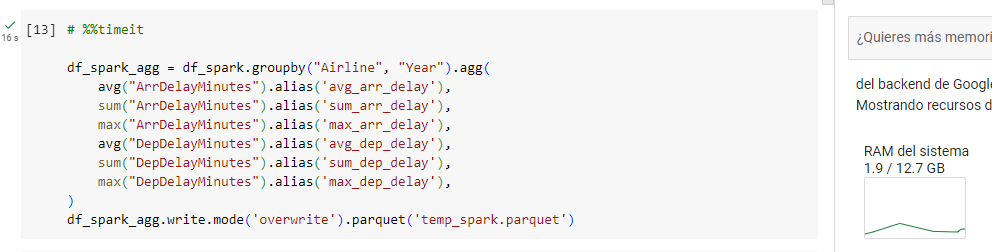


## LIBRERÍA PYSPARK

Si con Polars podíamos procesar los 5 archivos en uno solo script, en pySpark también podemos hacerlo a diferencia que su script es un poco mas complicado en la llamada de los datos.



Si con Polars usamos 3.7 de RAM en recursos de la maquina con pySpark usamos solo 1.9 de RAM de hecho no utilizamos casi recursos, pero si debemos hacer más largo el script.

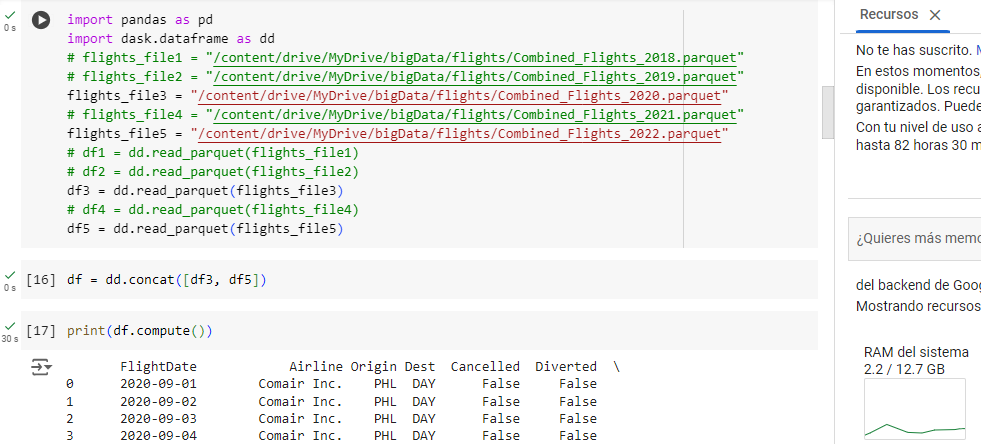


## LIBRERÍA DASK

Con dask utilizamos la misma estructura de Pandas a diferencia que importamos un nuevo parámetro llamado **dask.dataframe** para ellos haremos las mismas pruebas que hicimos con la librería pandas a ver si evidenciamos algún cambio en los recursos de la máquina.

|  |  |  |
| --- | --- | --- |
| LIBRERÍA | CARGA DE ARCHIVOS | RAM UTILIZADA |
| Pandas Prueba 1 | Archivo 2 y 3 |  |
| Pandas Prueba 2 | Archivo 4 y 5 | Aunque esta vez si pudo concatenar, tuvo un pico de 10.6 de RAM para luego bajar a 1.4 de RAM, puede llegar a colgarse |
| Pandas Prueba 3 | Archivo 2 y 5 |  |
| Pandas Prueba 4 | Archivos 3,4 y 5 |  |
| Pandas Prueba 5 | Archivos 2,3 y 4 | Saturado Completamente |
| Pandas Prueba 6 | Archivos 3 y 5 | Mejoro considerablemente de 9.6 de RAM paso solo ha utilizar 2.2 de RAM |
| Pandas Prueba 7 | Archivo 2,3,4 y 5 |  |

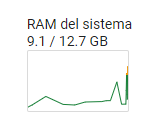
### Prueba 6



### Prueba 2

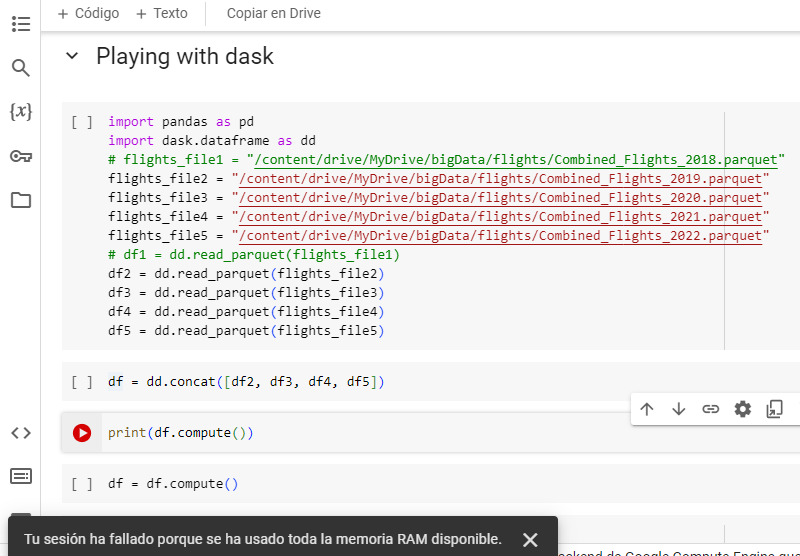


Podemos evidenciar que se ve un pico altísimo en recursos, pero termina bajando nuevamente al 1.4



Terminamos de hacer las pruebas, pero nos enfocamos mas en las 3 pruebas en las que sacamos conclusiones en la librería de Pandas, que en algunas si hubo cierto cambio mas que todo en la carga y en la concatenación de los datos en este caso utilizar dask favoreció mas que utilizar solo pandas.

Con dask cargar los archivos y concatenarlos es mas eficiente, pero al momento de procesarlos aun sigue gastando demasiado recursos de la maquina



## CONCLUSIÓN

Después de hacer este experimento comparando las librerías de Python para cargar datos sería más eficiente utilizar POLARS ya que podemos procesar los 5 archivos al mismo tiempo utilizando un poco de RAM y a pesar que su script es un poco largo, pero no es tan largo como el de pySpark.