Laboratorio 2

## Presentado por: Javier Ricardo Valderrama Gonzalez - 98001, Julián David Cifuentes Gonzalez - 100765

# Objetivo del Laboratorio

El objetivo principal de este laboratorio es comparar el desempeño de diferentes librerías de Python utilizadas para la carga y manipulación de datos tabulares. Las librerías en cuestión son Pandas, Polars, Spark y Dask. Se busca identificar cuál de estas librerías es más eficiente en términos de manejo de memoria y capacidad de procesamiento de datos en Google Colaboratory.

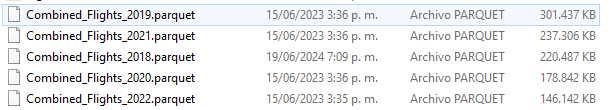
# Introducción

En este laboratorio se evaluará la eficiencia en el procesamiento básico de datos tabulares utilizando las librerías mencionadas. El propósito es determinar cuál puede manejar la mayor cantidad de datos sin saturar la memoria RAM disponible y evitar reinicios del entorno de Google Colaboratory. Los estudiantes deberán experimentar con cada librería y el conjunto de datos propuesto para encontrar la más eficiente en términos de carga de datos y uso de memoria.

# Desarrollo

En primer lugar, probaremos cargar archivos con la librería panda.

* A continuación, podemos observar el peso de los archivos con los cuales realizaremos la carga de la información, donde el archivo más pesado que podemos encontrar es el del año 2019 y el menos pesado es el del año 2022.



* Al crear el primer cuaderno en colab y ejecutar el primer comando podemos observar que la instancia de nuestro ambiente inicia con 1.2 de RAM y un disco de 28.1.

Código:

from google.colab import drive

drive.mount('/content/drive')

Interfaz de usuario gráfica

Descripción generada automáticamente

* Para la primera prueba vamos a cargar todos los archivos (df1, df2, df3, df4, df5), teniendo como resultado la saturación de la memoria RAM al momento de hacer la lectura del archivo 4.
  + El archivo df1 queda en un consumo de 7.1 GB de RAM.

Interfaz de usuario gráfica

Descripción generada automáticamente

* + El archivo df2 queda en un consumo de 8.3 GB de RAM.

Interfaz de usuario gráfica

Descripción generada automáticamente

* + El archivo df3 queda en un consumo de 10.2 GB de RAM.

Interfaz de usuario gráfica

Descripción generada automáticamente

* + El archivo df4 genera el desbordamiento de la memoria RAM.

Imagen de la pantalla de un celular de un mensaje en letras blancas

Descripción generada automáticamente con confianza baja

* Para la segunda prueba vamos a cargar todos los archivos (df1, df2, df4), teniendo como resultado la saturación de la memoria RAM al momento de hacer la lectura del archivo 4.
  + El archivo df1 queda en un consumo de 6.8 GB de RAM.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

* + El archivo df2 queda en un consumo de 9.8 GB de RAM.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

* + El archivo df4 genera el desbordamiento de la memoria RAM.

Imagen de la pantalla de un celular de un mensaje en letras blancas

Descripción generada automáticamente con confianza baja

* Para la segunda prueba vamos a cargar todos los archivos (df1, df2, df3), teniendo como resultado que los 3 archivos se cargan.
  + El archivo df1 queda en un consumo de 7.0 GB de RAM.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

* + El archivo df2 queda en un consumo de 7.0 GB de RAM.

Interfaz de usuario gráfica

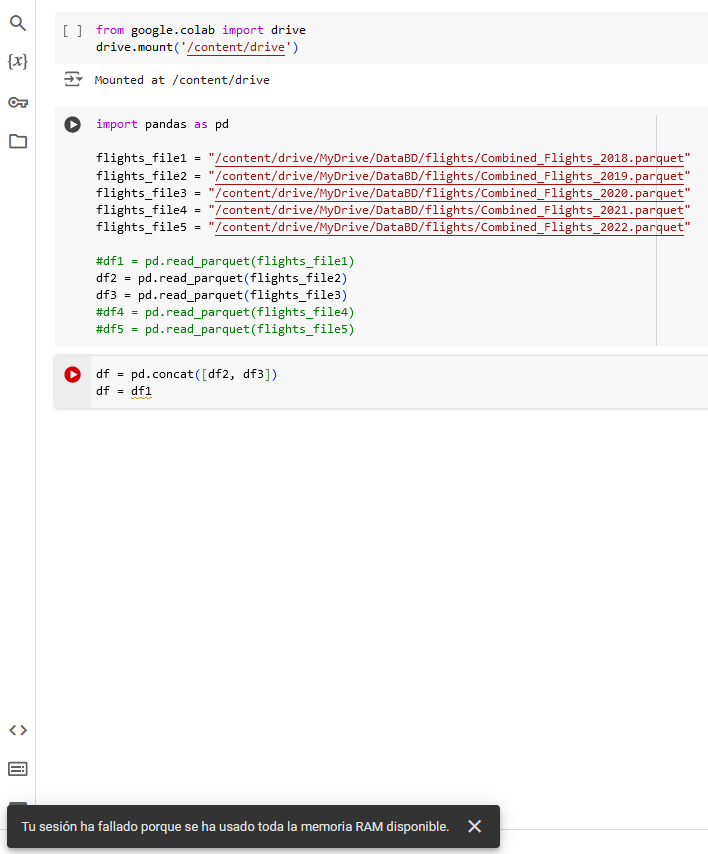
Descripción generada automáticamente

* + El archivo df3 queda en un consumo de 7.0 GB de RAM.

Interfaz de usuario gráfica

Descripción generada automáticamente

* Como Segunda Fase se intenta realizar los cargues necesarios para hacer las concatenaciones y tenemos los siguientes resultados.
  + Se desborda la ram al tratar de hacer concatenaciones de a 2 archivos el cual da error cuando intentamos concatenar los pares:
    - df1 - df2
    - df2 - df3
    - df2 - df4
  + El resto de pares concatenaron de forma correcta sin embargo nos deja un porcentaje altísimo de utilización de RAM.



En segundo lugar, probaremos cargar archivos con la librería Polars.

* Al crear el primer cuaderno en colab y ejecutar el primer comando podemos observar que la instancia de nuestro ambiente inicia con 1.2 de RAM y un disco de 28.1.

Código:

import polars as pl

Interfaz de usuario gráfica

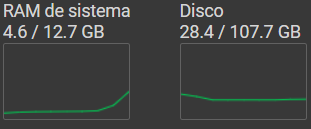
Descripción generada automáticamente

* Para la primera prueba vamos a cargar todos los archivos (df1, df2, df3, df4, df5), teniendo como resultado el cargue de los documentos en 0s y sin afectar el rendimiento de la memoria RAM en grandes escalas.

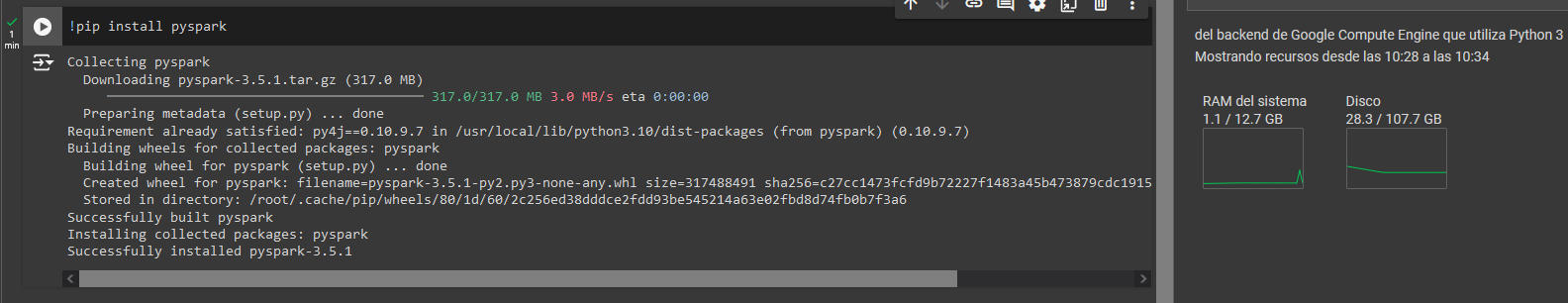
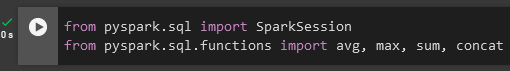
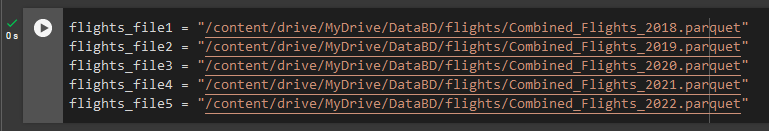
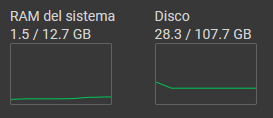
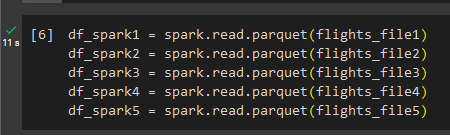
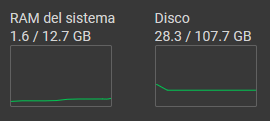
Interfaz de usuario gráfica

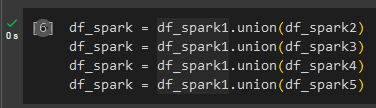
Descripción generada automáticamente

* Para la combinación de los archivos observamos que se realiza con todos y vemos que el uso de la memoria RAM se sube a 4.6 pero en ningún momento genera un desbordamiento como sucede con la librería de Pandas

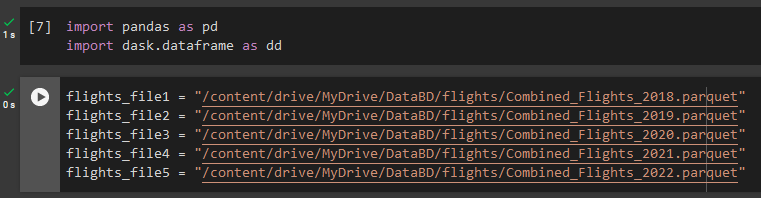
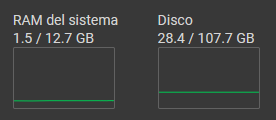
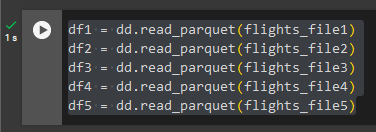


En tercer lugar, probaremos cargar archivos con la librería PySpark.

* Al momento de realizar la instalación no tenemos mucho saturamiento de memoria un poquito de disco y tarda menos de 1 minuto y la instancia de la libreria lo hace sin ningun fallo.  
    
    
    
  
* En el cargue de archivos Combined\_Flights no presenta ningún error al igual que polar no tiene ninguna demora y sin saturación de memoria RAM o de Disco  
    
    
  
* En la lectura de los archivos de igual manera es muy rápido y no satura la RAM en lo cual solo tuvo un aumento de 1.5 a 1.6 en RAM   
    
  
* Al intentar hacer diferentes combinaciones de unión o concatenación de archivos en PySpark funcionan todas las combinaciones de manera correcta sin mucho consumo de ram y de forma rápida.



En último lugar, probaremos cargar archivos con la librería Dask.

* Al importar la librería y llamar los archivos no tenemos ningún tipo de complicación ni en ram ni en disco carga de manera rápida y sin dificultades.  
    
  
* al intentar leer los archivos con la librería dask lo hace muy rápido sólo tarda 1 segundo en leer los 5 archivos que hemos manejado a lo largo del informe, por lo tanto no presenta saturación ni la Ram se expone a saturaciones.  
  
* Todas las combinaciones de concatenación las hace muy rápidas al punto que intentamos hacer la concatenación de los 5 archivos en un solo archivo y resultó muy rápido no tiene ningún contratiempo y da una respuesta correcta como lo vemos en la imagen.  
    
  