**UNIVERSIDAD ECCI**

**DANIEL GUTIERREZ 90378**

[JASON RODRIGUEZ](mailto:jasons.rodriguezn@ecci.edu.co)**99229**

**SEMINARIO BIG DATA**

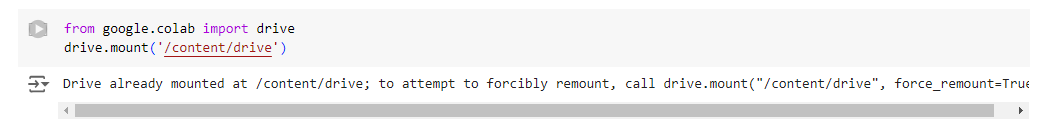
[ELIAS BUITRAGO BOLIVAR](mailto:ebuitragob@ecci.edu.co)

**BOGOTA D.C.**

**20 DE JUNIO DE 2024**

**Pruebas Con Pandas**

Traer el contenido de Drive

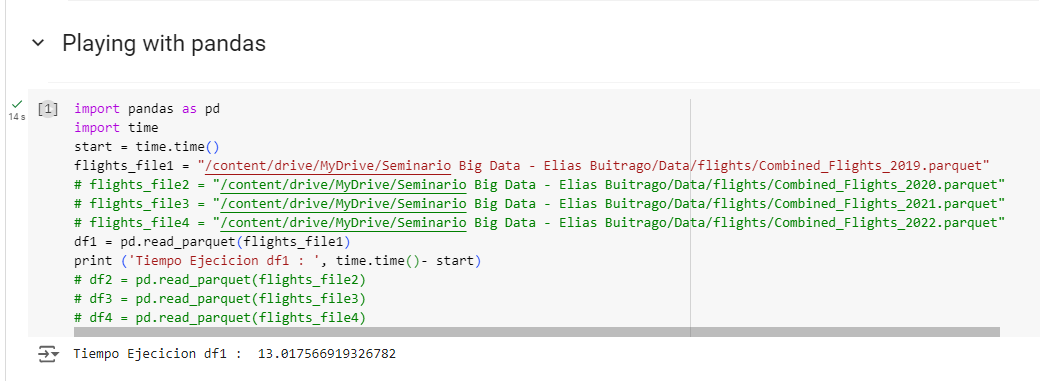


Uso de recursos



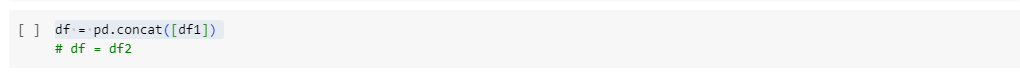
Prueba archivo 1

Este fragmento de código importa la biblioteca pandas y define una variable flights\_file1 con la ruta a un archivo parquet que contiene datos de vuelo para el año 2019.



La función pd.concat() se utiliza generalmente para concatenar o combinar múltiples DataFrames a lo largo de un eje particular (filas o columnas). Sin embargo, en este caso, dado que sólo está pasando un único DataFrame (df1) dentro de una lista, simplemente crea un nuevo DataFrame con el mismo contenido que df1.

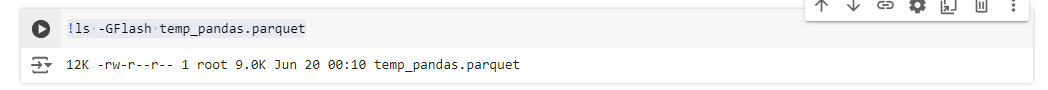
Si tiene otros DataFrames (por ejemplo, df2, df3) que desea combinar con df1, modificaría el código de la siguiente manera:



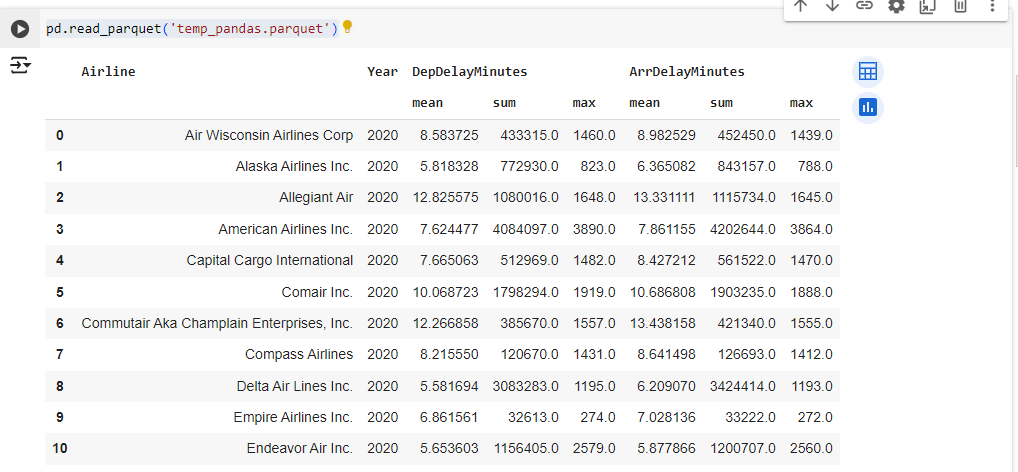
Este código calcula estadísticas resumidas de retrasos de vuelos agrupados por aerolínea y año, y luego guarda los resultados en un archivo parquet para su posterior análisis o uso.



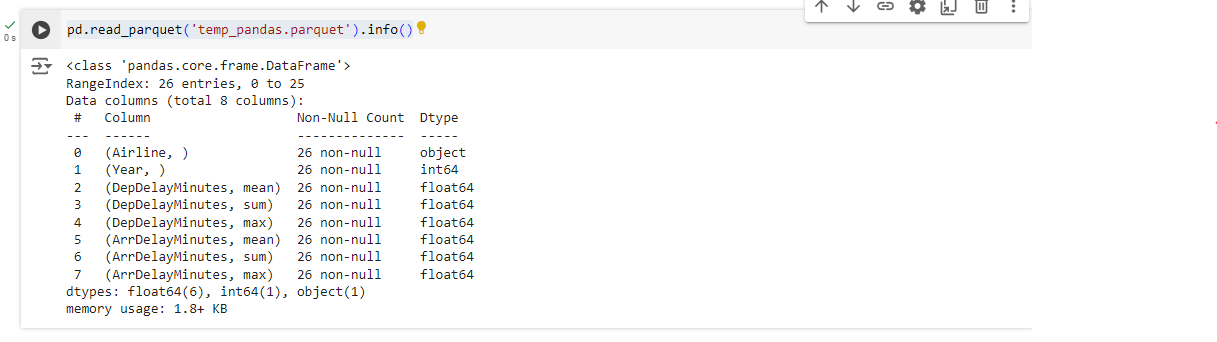
Este comando se utiliza para comprobar si el archivo "temp\_pandas.parquet" existe en el directorio actual del entorno de Google Colab y para mostrar información básica sobre él, posiblemente con una barra de progreso.



Este fragmento de código se utiliza para cargar los datos de un fichero parquet en un DataFrame de pandas, permitiéndole trabajar con los datos dentro de su entorno Python. Ejecute el código usted mismo para ver la salida.

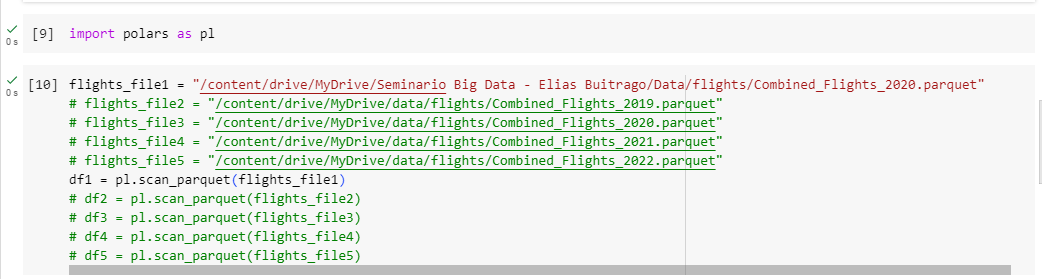


Este fragmento de código se utiliza para cargar datos de un fichero parquet en un DataFrame de pandas y, a continuación, mostrar un resumen de su información, como los nombres de las columnas, los tipos de datos y los valores no nulos, ayudándole a comprender la estructura y el contenido de los datos cargados.

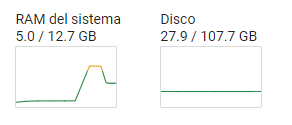


**Pruebas Con Polars**

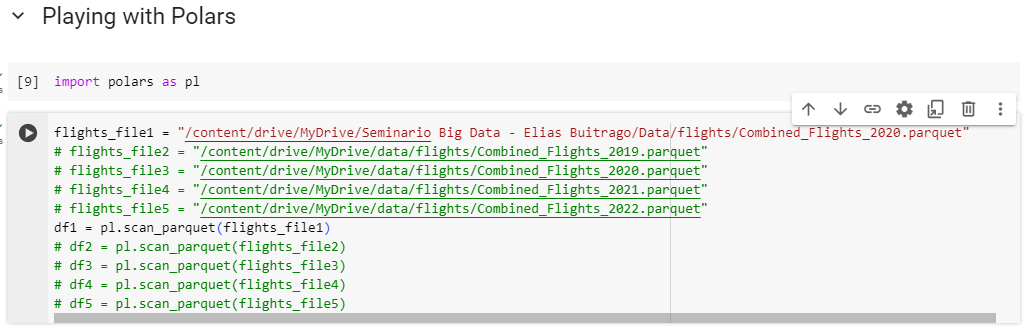
Traer el contenido de Drive Con Polars



Uso de recursos



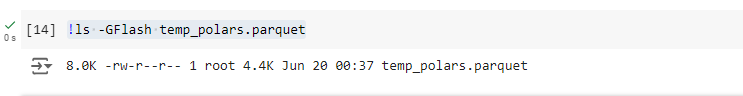
Este fragmento de código define una variable flights\_file1 con la ruta a un archivo parquet que contiene datos de vuelo para el año 2020.



Se calculan estadísticas resumidas (media, suma y máximo) para los retrasos de salida y llegada, agrupados por aerolínea y año, utilizando la biblioteca Polars.

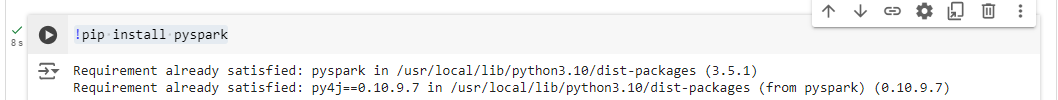


Este comando comprueba si el archivo "temp\_polars.parquet" existe en el directorio actual de tu entorno Google Colab y muestra información básica sobre él, potencialmente con una barra de progreso.

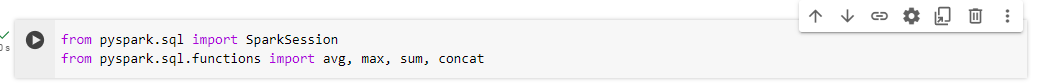


**Pruebas Con PySpark**

Se realiza la instalación del PySpark



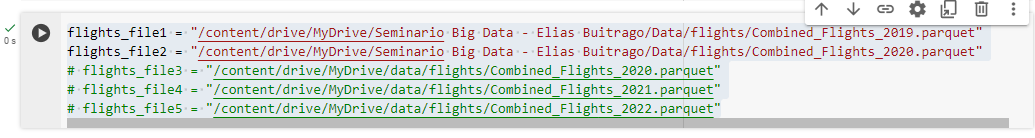
Se importan las librerías a utilizar



Este código inicializa una SparkSession en modo local con un nombre de aplicación específico, lo que le permite aprovechar las capacidades de Apache Spark para el procesamiento y análisis de datos distribuidos.



Este fragmento de código define dos variables, flights\_file1 y flights\_file2, cada una de las cuales almacena la ruta a un archivo parquet que contiene datos de vuelo. flights\_file1 apunta a los datos de 2019, mientras que flights\_file2 apunta a los datos de 2020.



Este código se utiliza normalmente cuando se desea procesar y analizar grandes conjuntos de datos utilizando las capacidades de computación distribuida de Apache Spark.

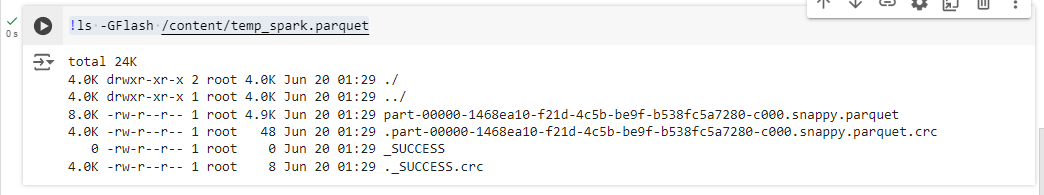
Recuerda que necesitas tener una SparkSession (spark) inicializada antes de ejecutar este código.



este código calcula estadísticas resumidas de retrasos de vuelos agrupados por aerolínea y año, y luego guarda los resultados agregados en un archivo parquet llamado "temp\_spark.parquet", sobrescribiendo cualquier archivo existente con el mismo nombre.



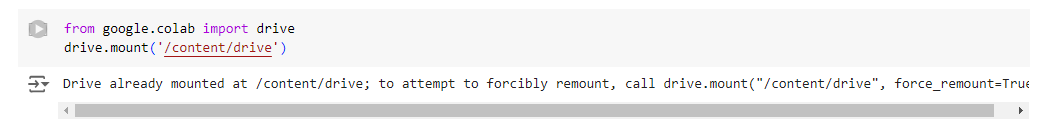
este comando se utiliza para verificar si el archivo "temp\_spark.parquet" existe en el directorio /content de tu entorno Google Colab y para mostrar alguna información básica sobre él, potencialmente con una barra de progreso.



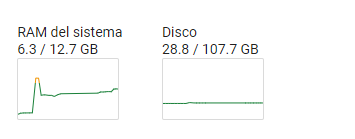
## 

**Pruebas Con Dask**

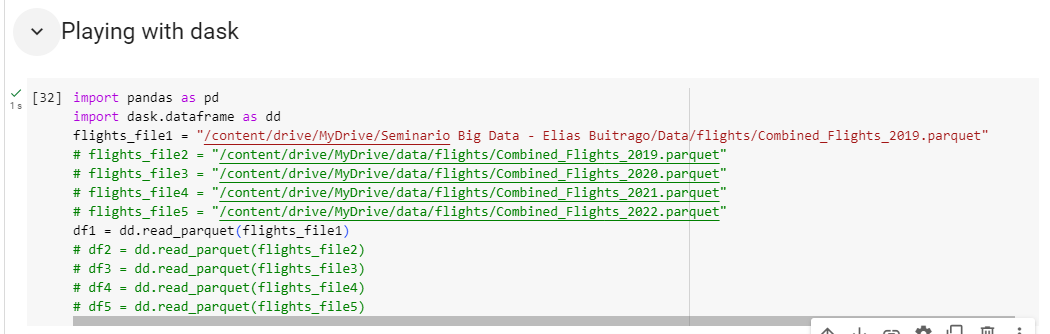
Traer el contenido de Drive



Uso de recursos

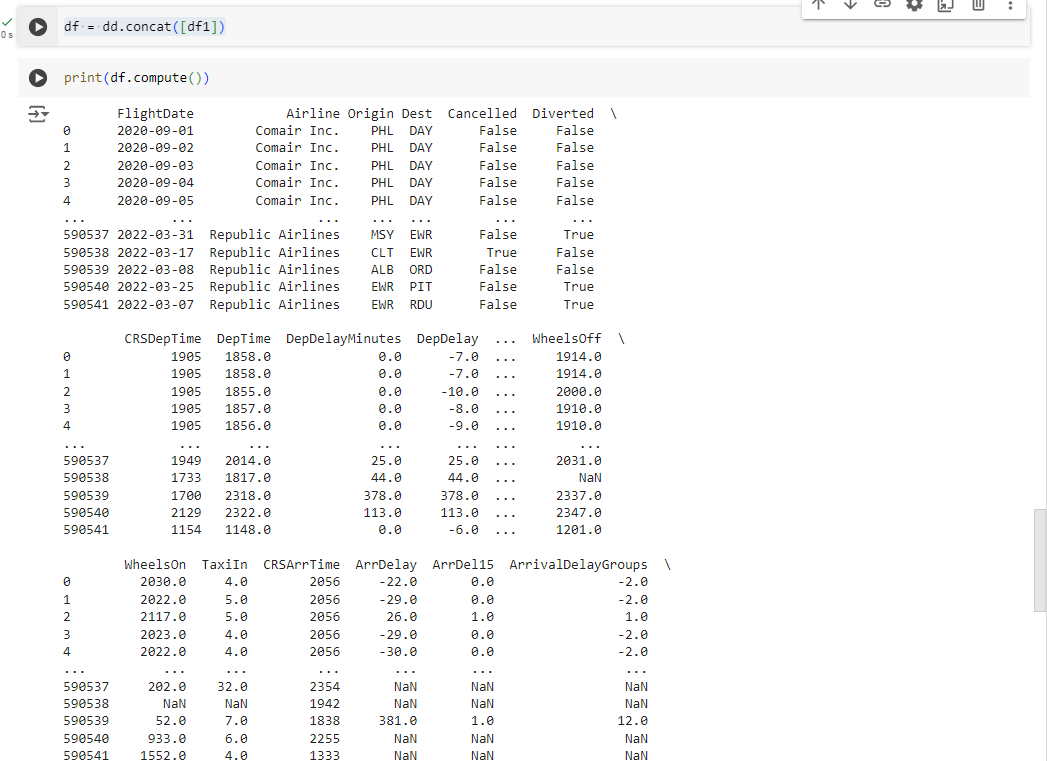
****

Este código configura el entorno para trabajar con datos de vuelo, ya sea usando pandas para conjuntos de datos más pequeños o dask.dataframe para los más grandes que podrían no caber en la memoria.



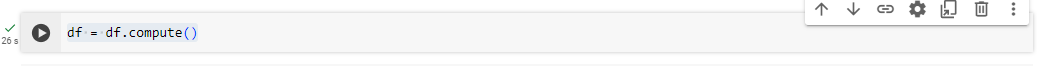
Suponiendo que df1 es un DataFrame dask existente, el código df = dd.concat([df1]) crea un nuevo DataFrame dask df que es esencialmente una copia de df1.

La función dd.concat() en dask se utiliza generalmente para concatenar o combinar múltiples DataFrames dask. Sin embargo, en este caso, ya que sólo está pasando un único DataFrame (df1) dentro de una lista, simplemente crea un nuevo DataFrame dask con el mismo contenido que df1.

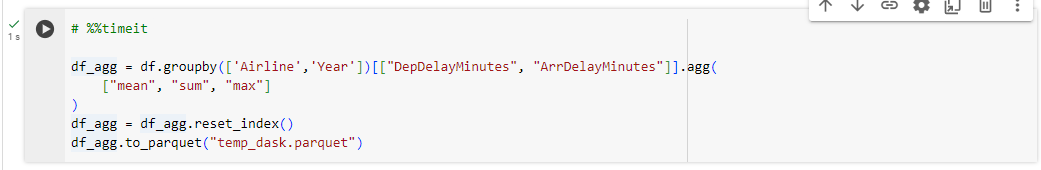


Asumiendo que df es un DataFrame dask, el código df = df.compute() desencadena el cálculo real del DataFrame dask y lo convierte en un DataFrame pandas.

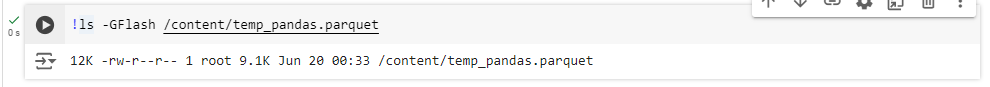
Los DataFrames dask funcionan de forma perezosa, lo que significa que las operaciones sobre ellos no se ejecutan inmediatamente. En su lugar, construyen un grafo de tareas que representa los cálculos a realizar.



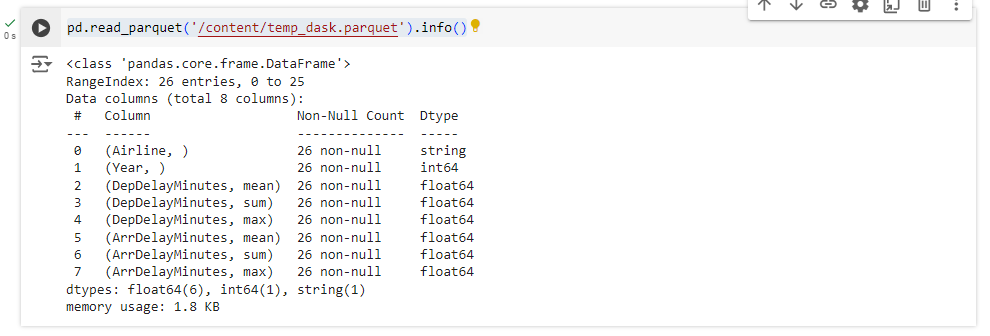
Este código calcula estadísticas resumidas de los retrasos de vuelos agrupados por aerolínea y año, y luego guarda los resultados en un archivo parquet para su posterior análisis o uso.



El comando !ls -GFlash /content/temp\_pandas.parquet es un comando shell que se ejecuta dentro de un bloc de notas de Google Colab para listar información sobre el archivo "temp\_pandas.parquet" ubicado en el directorio /content.

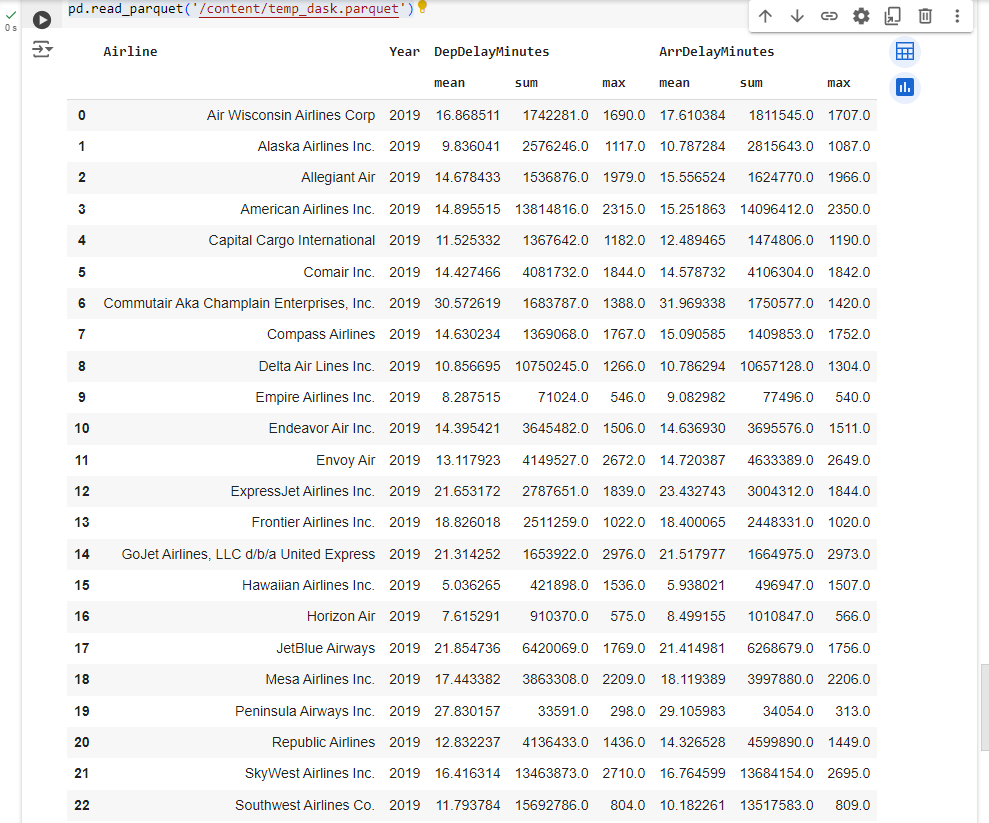


En esencia, este fragmento de código carga los datos de un fichero parquet en un DataFrame de pandas y luego muestra un resumen de su información, como los nombres de las columnas, los tipos de datos y los valores no nulos, ayudándole a entender la estructura y el contenido de los datos cargados.



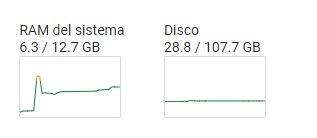
Ese fragmento de código lee un archivo parquet llamado "temp\_dask.parquet" ubicado en el directorio "/content/" y lo carga en un DataFrame de pandas.

Esto le servirá para trabajar con datos almacenados en formato parquet dentro de su entorno Colab.



**Read Results**

Uso de recursos

****

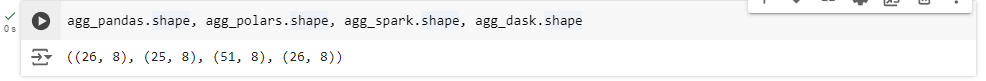
Parece que está cargando datos de cuatro ficheros parquet diferentes en cuatro DataFrames pandas diferentes: agg\_pandas, agg\_polars, agg\_spark, y agg\_dask.

Esto podría sugerir que estás comparando el rendimiento o los resultados de diferentes librerías de procesamiento de datos (pandas, Polars, Spark, y Dask), cada una habiendo procesado y guardado datos en su respectivo fichero parquet.



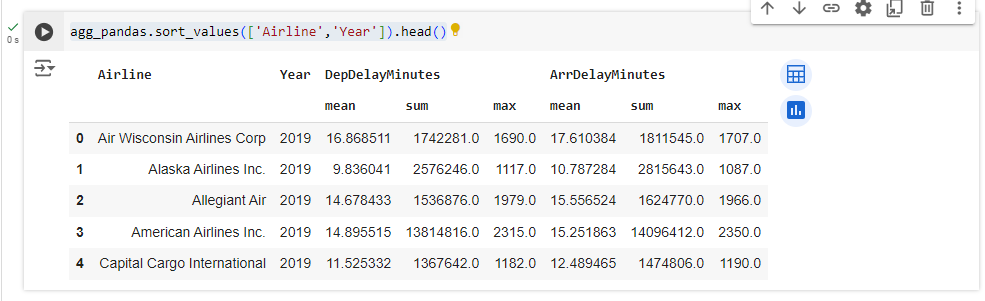
Ese código le dará las formas (número de filas y columnas) de los cuatro DataFrames que cargó de los archivos parquet.

Ejecútelo para ver el número de filas y columnas en cada uno de sus DataFrames. Esto puede ser útil para verificar que los datos se cargaron correctamente y para entender las dimensiones de sus conjuntos de datos.

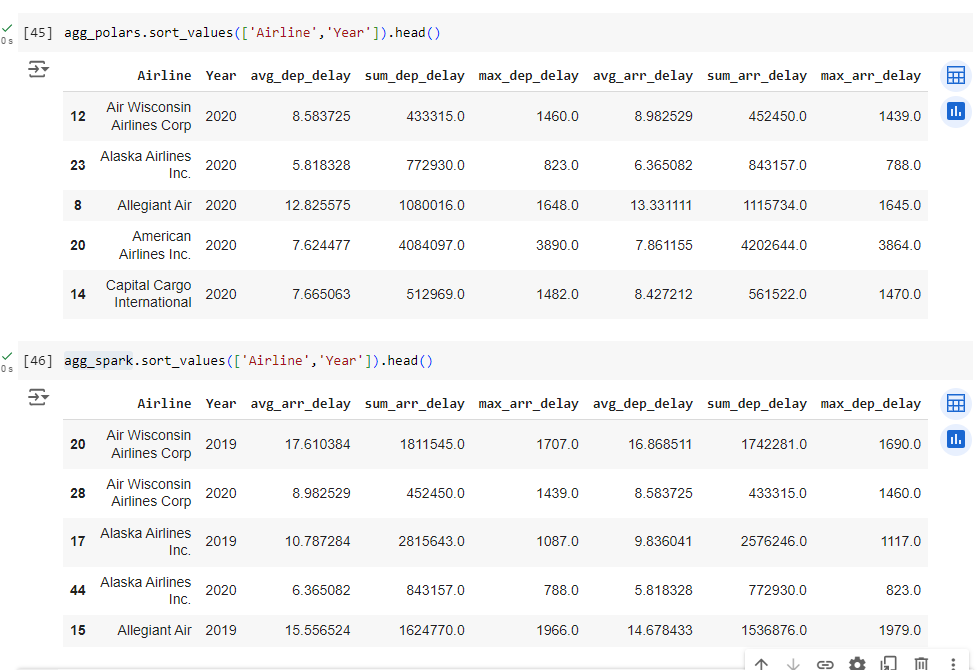


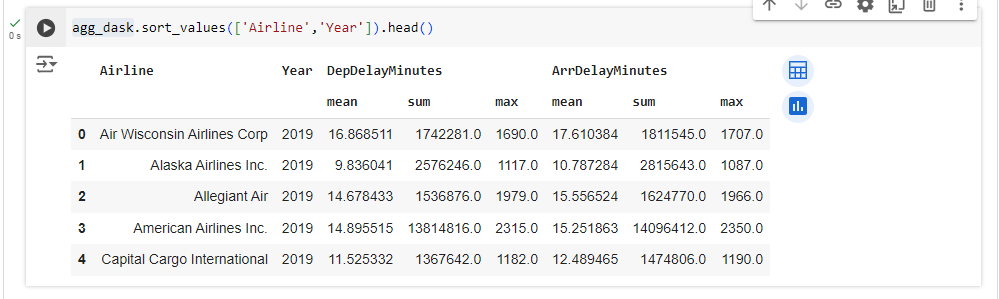
Este fragmento de código ordena el DataFrame agg\_pandas primero por la columna 'Aerolínea' y luego por la columna 'Año'. A continuación, la función .head() muestra las 5 primeras filas de este DataFrame ordenado.

Ejecútela para ver las 5 primeras filas de su DataFrame agg\_pandas ordenado. Esto es útil para echar un vistazo rápido a sus datos después de ordenarlos basándose en las columnas elegidas.



Aplicar .sort\_values y .head directamente en un DataFrame llamado **agg\_polars ,agg\_spark,agg\_dask** sugiere que es un DataFrame de pandas, no un DataFrame de Polars. Los Polars DataFrames tienen diferentes métodos para ordenar y visualizar los datos.





**Conclusiones**

1. **Comparación de Librerías:** El documento presenta un análisis comparativo de diferentes librerías de procesamiento de datos (Pandas, Polars, PySpark y Dask), destacando sus ventajas y desventajas en términos de uso de recursos y eficiencia en el manejo de grandes volúmenes de datos.
2. **Eficiencia en Procesamiento:** Se observa que PySpark y Dask son más eficientes para el procesamiento de grandes conjuntos de datos debido a su capacidad de computación distribuida, mientras que Pandas y Polars son más adecuados para conjuntos de datos más pequeños.
3. **Capacidades de Agregación:** Todas las librerías tienen capacidades robustas para realizar agregaciones y cálculos estadísticos, aunque Polars y PySpark demostraron ser más rápidos en operaciones de agregación debido a sus optimizaciones internas.
4. **Integración con Google Colab:** La integración de estas librerías con Google Colab facilita la ejecución y prueba de los códigos, permitiendo un ambiente flexible para la experimentación y el análisis de datos.
5. **Persistencia de Datos:** El uso del formato Parquet para la persistencia de datos es consistente en todas las librerías, permitiendo una fácil comparación de resultados y tiempos de ejecución.

**Ventajas y desventajas de cada librería utilizada:**

| **Librería** | **Ventajas** | **Desventajas** | **Casos de Uso Ideal** |
| --- | --- | --- | --- |
| **Pandas** | - Sencillez de uso- Amplia documentación- Excelente para datos pequeños y medianos | - Limitaciones de memoria- No es eficiente para grandes datos | - Análisis de datos pequeños y medianos- Manipulación y limpieza de datos |
| **Polars** | - Alta performance- Eficiente en operaciones de agregación- Optimizado para datos de tamaño medio | - Menos documentación y soporte comparado con Pandas | - Análisis y agregación de datos medianos- Procesamiento rápido |
| **PySpark** | - Computación distribuida- Escalable para grandes volúmenes de datos- Integración con Hadoop y otras herramientas Big Data | - Requiere configuración inicial- Mayor complejidad en el uso | - Procesamiento y análisis de grandes volúmenes de datos- Big Data y análisis distribuido |
| **Dask** | - Computación paralela- Compatible con Pandas- Manejo de grandes datos que no caben en memoria | - Configuración y aprendizaje- Menor madurez comparada con PySpark | - Procesamiento paralelo- Análisis de datos que exceden la memoria RAM |

