AWS SDK FOR PANDAS DATA WRANGLER

Descargar fichero de datos

INTALACION

```
import awswrangler as wr
wr.__version__
```

```
import boto3
boto3.setup_default_session(region_name="us-east-1")
bucket='mycestebucket'
wr.s3.does_object_exist(f"s3://{bucket}/flights.csv")

```

```
boto3.setup_default_session(profile_name="default", region_name="us-east-1")
wr.s3.does_object_exist(f"s3://{bucket}/flights.csv")
```

S3

Gestion de objetos

Cargar y descargar objetos

```
localpath='data/csv/flights.csv'
path1 = f"s3://{bucket}/upload/flights.csv"
wr.s3.upload(local_file=localpath, path=path1)
wr.s3.download(local_file='./flights.csv', path=path1)
```

Listar

```
print('buckets',wr.s3.list_buckets())
print('dirs',wr.s3.list_directories(f's3://{bucket}/'))
print('buckeobjects',wr.s3.list_objects(f's3://{bucket}/'))
```

Copiar y borrar objetos

```
path1 = f"s3://{bucket}/upload/flights.csv"
wr.s3.copy_objects(
    paths=[path1],
    source_path=f"s3://{bucket}/upload/",
    target_path=f"s3://{bucket}/copied/",
    use_threads=True  #para paralelizar
)
```

```
path1 = f"s3://{bucket}/upload/flights.csv"
path2 = f"s3://{bucket}/copied/flights.csv"
wr.s3.delete_objects([path1, path2])  # Delete both objects
```

```
# List and delete files in the folder
s3_folder=f"s3://{bucket}/upload/"
wr.s3.delete_objects(paths=wr.s3.list_objects(path=s3_folder))
```

Ver los metadatos de los objetos

```
# Define S3 file path
s3_file = f"s3://{bucket}/csv/flights.csv"

describe=wr.s3.describe_objects(path=s3_file)
for k,v in describe[s3_file]['ResponseMetadata']['HTTPHeaders'].items():
    print(k,v)
```

Escritura y lectura de dataframes

```
#to get a localname protected visualization
#import getpass
#bucket = getpass.getpass()
```

```
import pandas as pd
localpath='data/csv/flights.csv'
df1= pd.read_csv(localpath)
display(df1.head())

path1 = f"s3://{bucket}/csv/flights.csv"
wr.s3.to_csv(df1, path1, index=False)
```

```
df1_read=wr.s3.read_csv([path1])
df1_read.head(5)
```

```
#partimos el csv en dos ficheros (el segundo con los ultimos dos registros del primero)
localpath1='data/csv/flights1.csv'
df1= pd.read_csv(localpath1)
localpath2='data/csv/flights2.csv'
df2= pd.read_csv(localpath2)
display(df2.head())

#leyendo multiples ficheros a la vez
path1 = f"s3://{bucket}/csv/flights1.csv"
path2 = f"s3://{bucket}/csv/flights2.csv"
wr.s3.to_csv(df1, path1, index=False)
wr.s3.to_csv(df2, path2, index=False)
```

```
13 df_read=wr.s3.read_csv([path1,path2])
14 df_read.tail(5)
```

EJERCICIOS

Ejercicio: Escribir y leer datos Parquet desde S3

Crear los datos de vuelos en un DataFrame, escribirlo en S3 en formato parquet. Listar objetos creados y despues leerlo en otro DataFrame y ver que esta correcto.

Prueba diversas opciones de la escritura:

- compresion, etc...
- escribir en multiples ficheros utilizando un numero maximo de filar por fichero o con particion for dia de la semana y hora del vuelo

Objetivo: Aprender a escribir y leer datos en formato parquet directamente en S3 usando AWS Data Wrangler y pandas.

```
1 # cargar fichero de datos flights en un DataFrame
2 # guardarlo en un directorio /parquet/ como un fichero .parquet -> comprimido, max_rows 100
3 # listar objetos que se han creado en el bucket
4 # descargar y leerlos de S3 en otro DataFrame
5
6 # probar particionar los datos al escribirlos por dia de la semana y hora
7
```

Ejercicio: Filtrar datos al leer de S3

Leer un archivo Parquet desde S3 y filtrar filas basadas en valores específicos de las particiones directamente durante la lectura.

Objetivo: Aprender a aplicar filtros durante la lectura para optimizar el manejo de datos.

Filtrando por las columnas de particion

```
1 # utilizar el atriburo partition_filter para importar
2 # en un DataFrame solo las filas que tengan dia de la semana 4
```

Leer solo unas columnas

```
1 # utilizar el atriburo columns para importar
2 # en un DataFrame solo las columnas de precio y distancia del vuelo que tengan dia de la semana 4
```

Ejercicio: Lectura de datos incremental

Cargar solo los nuevos archivos en un bucket S3 desde la última operación de procesamiento, basado en las marcas de tiempo de los archivos.

Cargar los ficheros `.csv` mas recientes de 1 semana

Objetivo: Trabajar con la carga incremental de datos para manejar flujos continuos.

```
1 # obtener todos los objetos que se hayan creado hace un determinado dia y hora
2 # leerlos cargarlos en un DataFrame
```

Ejercicio: copia recursiva de objetos entre buckets en paralelo

Copiar todos los archivos y subcarpetas de un folder S3 a otro, preservando la estructura.

Objetivo: Entender cómo mover grandes cantidades de datos entre carpetas de S3.

```
1 # copia los objetos del dir /parket/ en otro dir
2 # utilizando use_threads=True
```

Ejercicio: ETL pipeline

Extraer CSV de un dir en S3, tranformarlos y guardarlos en otro dir en formato parquet comprimidos y particionados.

Puede desplegarse en una lambda para que añada ficheros parquet cada vez que se sube un fichero CSV

Objetivo: Ejercicio encadenando funcionalidades en un pipeline ETL

```
1 # tu codigo
```