Practica Subway

INDICE

- ✔ OBJETIVO
- ✔ INTRODUCCION
- ✔ PIPELINE
- ✔ EXTRACCION
- ✔ TRANSFORMACIÓN
- ✔ CARGA
- ✔ CODIGO
- ✔ OBJETOS

Practica Subway

OBJETIVO

En la práctica del modulo de ETL se solicita los siguientes puntos:

Descripción del proyecto
Este proyecto consiste en ayudar a una tienda que vende utensilios, ingredientes y especias internacionales a planificar dónde ubicar sus nuevas tiendas en su llegada a España. Para ello tendrás que diseñar un proceso ETL desde la extracción de los datos, pasando por su limpieza y terminando en su almacenamiento Este proyecto se centra en buena parte en utilizar Técnicas de ETL.

Objetivos

- ✔ Utiliza el CSV proporcionado y límpialo. Presta atención al resultado final que debería ser una tabla que contenga la siguiente información; id, cp, provincia, ciudad, provincia_local (con el nombre en el idioma local, por ejemplo en catalán), ciudad_local (con el nombre en el idioma local, por ejemplo en catalán)

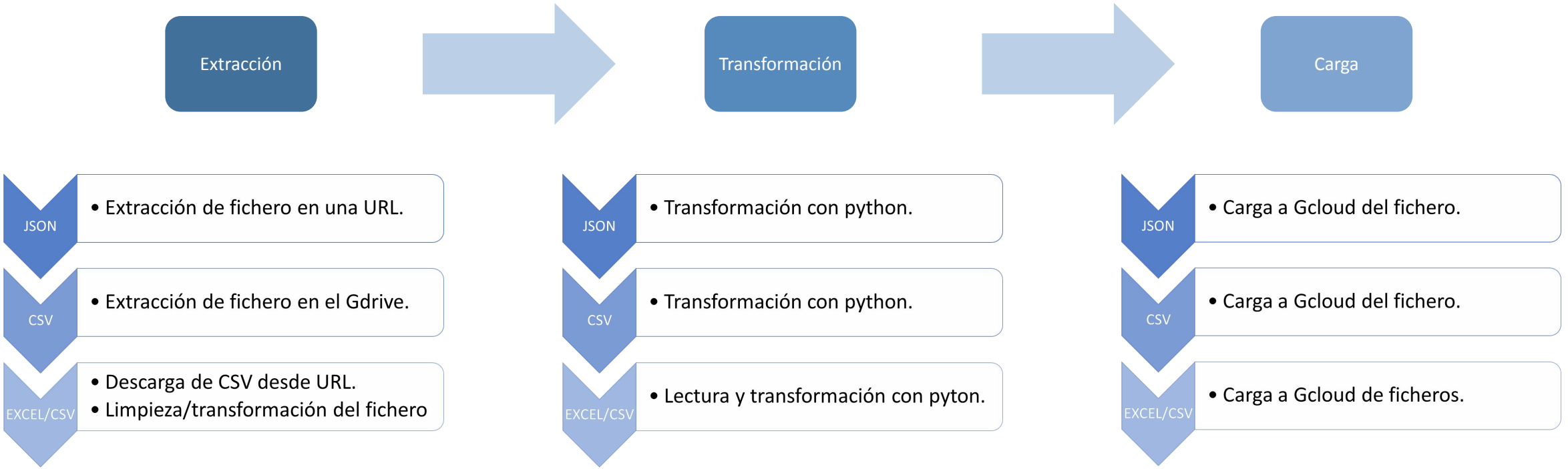
- ✔ La empresa se quiere centrar en un primer momento en Girona, por lo que es necesario obtener la población de todas las ciudades en la provincia. Esta información está disponible públicamente en https://servicios.ine.es/wstempus/js/es/DATOS_TABLA/33791?tip=AM. Tienes que limpiar los datos que obtienes para obtener un JSON y posteriormente crear una tabla que contiene la siguiente información: id, población, origen, y una columna para cada año del que hay datos.

- ✔ Ahora debes encontrar una forma de extraer los datos de datos.gob.es que contienen esa misma información pero para todas aquellas regiones donde el formato de datos JSON/html está disponible. Accediendo a https://datos.gob.es/es/catalogo encontrarás un icono para descargar todo el catálogo de datos disponible. Busca la forma de limpiar los datos en ese Excel (puedes usar excel) y genera un archivo excel/csv que contenga sólo las filas de los datos que contengan en el título "Población por sexo, municipios y nacionalidad". Una vez que lo tengas, serializa el proceso para obtener los datos en todas las URL en el mismo formato del punto anterior.

Practica Subway

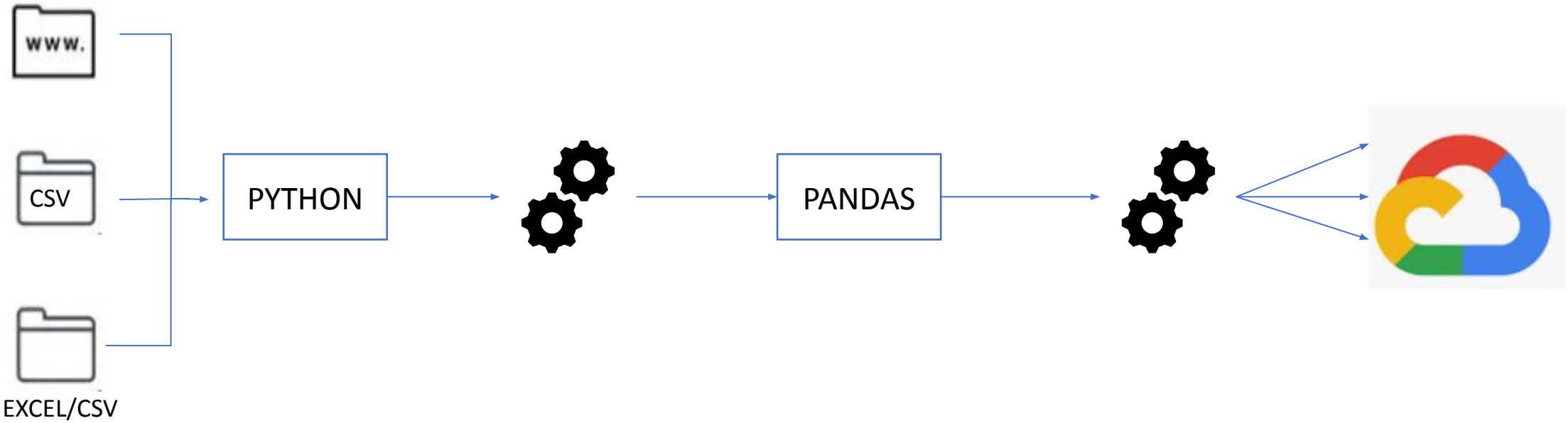
INTRODUCCIÓN

La práctica se ha realizado mediante COLAB con lenguaje PYTHON.
Para el desarrollo de la práctica se ha fijado el siguiente flujo:



Practica Subway

PIPE LINE



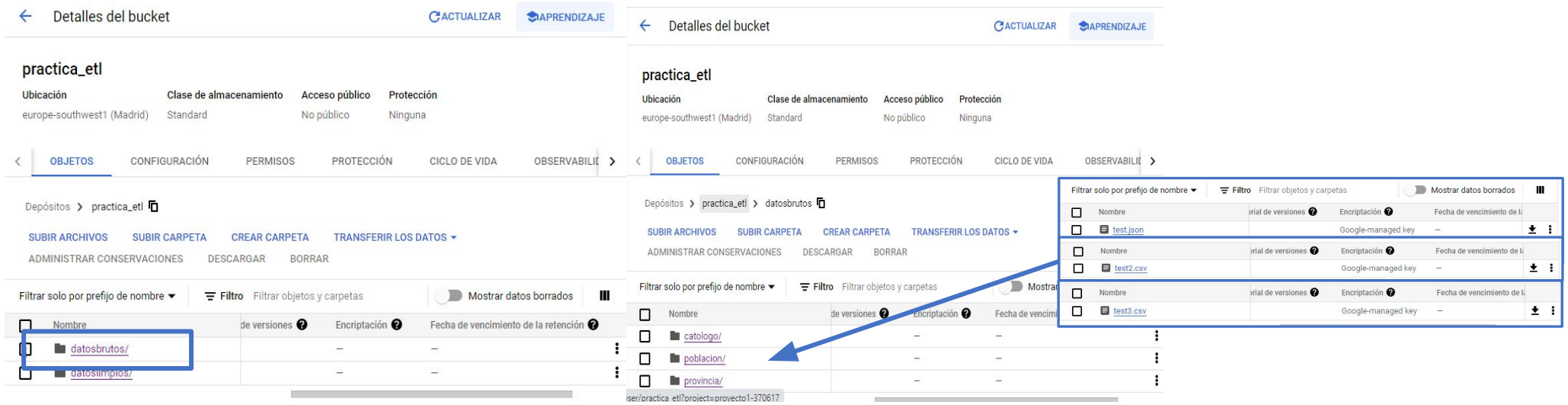
Practica Subway

EXTRACCIÓN:

✔ El objetivo de este paso de "Extracción" es recopilar los datos sin procesar de la fuente correcta, en nuestro caso las fuentes han sido un CSV proporcionado por el cliente y dos paginas web:

https://servicios.ine.es/wstempus/js/es/DATOS_TABLA/33791?tip=AM.
https://datos.gob.es/es/catalogo. (En este caso hemos descargado y limpiado un archivo Excel, hasta quedarnos sólo con las filas de los datos que contengan en el título "Población por sexo, municipios y nacionalidad".

Todos los datos han sido guardados en la carpeta de datos brutos en el Google Cloud.



Practica Subway

EXTRACCIÓN:

Para cargar los ficheros en bruto en Google Cloud, hemos utilizado Python con los siguientes pasos:

Definimos una función de guardado en Google Cloud.

```
def saveGcloud(_data, _name, _ruta):

    client = storage.Client()
    bucket = client.bucket('practica_etl')

    blob = bucket.blob(_ruta + _name)
    blob.upload_from_string(_data)

    print('Subido correctamente')
```

Se ha realizado una lectura de los orígenes:

- En el caso de la URL lo hemos transformado en JSON.
- Para el fichero CSV no se ha transformado
- En el fichero Excel/CSV no se ha transformado

JSON

```
url = 'https://servicios.ine.es/wstempus/js/es/DATOS_TABLA/33791?tip=AM'

payload = requests.get(url)
raw_json = payload.json()

json_text = json.dumps(raw_json, indent = 4)


miCarpeta = 'datosbrutos/poblacion/'


archivo = 'test.json'


saveGcloud(json_text, archivo, miCarpeta)
```

CSV

```
csv_txt = open('/content/drive/MyDrive/Proyecto_final_ETL/TFA '
    '- Proyecto 02 - Postal codes2.csv', 'rt', encoding='utf-8').read()


raw2= csv_txt.split('\n')


miCarpeta = 'datosbrutos/provincia/'


archivo = 'test2.csv'


saveGcloud(json_text , archivo, miCarpeta)
```

EXCEL/CSV

```
csv_txt = open('/content/drive/MyDrive/Proyecto_final_ETL/URL_JSON.csv'
    , 'rt', encoding='utf-8').read()


raw3= csv_txt.split('\n')


miCarpeta = 'datosbrutos/catologo/'


archivo = 'test3.csv'


saveGcloud(json_text , archivo, miCarpeta)
```

Después se han cargado al Google Cloud con la función definida al principio.

Practica Subway

TRANSFORMACIÓN:

El objetivo de este paso de “Transformación" es aplicar la manipulación y transformación de datos para mejorar la calidad de los mismos, por lo que a continuación explicaremos todos los pasos que hemos realizado en dicha Transformación.

Proceso 1:

1. Leemos la URL https://servicios.ine.es/wstempus/js/es/DATOS_TABLA/33791?tip=AM, la convertimos en un diccionario, después normalizamos en un dataframe de años y otro de variables y los unimos en uno solo.
2. Nos quedamos con las columnas necesarias (código, Municipio, País, sexo y años) y eliminamos el resto.
3. Como no se nos pide ningún dato relacionado con sexo nos quedamos con los datos agregados que son los que tienen valor **Total**.
4. Eliminamos la columna sexo y provincia y nos quedamos con las columnas código, Municipio, País, y años por columnas, siendo País el origen.
5. Eliminamos las filas donde Municipio y País sean vacíos, así como eliminamos las filas donde el valor de origen sea extranjero, ya que el valor extranjero es el total de la suma de todos los países excepto España.
6. Por último cambiamos los nulos con 0 y como resultado obtenemos un dataframe con las columnas y los datos necesarias.

Practica Subway

TRANSFORMACIÓN:

Proceso 2:

1. Leemos el CSV aportado por el cliente y lo transformamos en un dataframe cuyas columnas son provincia, población y código postal.
2. Separamos la columna provincia en otras 2, obteniendo una columna “provincia_local” y otra “Provincia”.
3. Se repite la misma operación con la columna población, obteniendo dos columnas que se han denominado “ciudad_local” y “ciudad_trf”.
4. A continuación como tenemos valores en “provincia_loc” pero no tenemos valores en “Provincia” (porque al separar la columna con la “/” las provincias con mismo idioma solo vienen con un valor) se ha usado *un numpy para comparar el valor de la “Provincia” y si es vacía asignarle el valor de la columna “provincia_loc”*.
5. Repetimos el paso anterior pero en este caso para población.
6. Eliminamos las columnas que no necesitamos y obtenemos un dataframe con las columnas “Cod. Postal”, “Provincia_loc”, “Ciudad_loc”, “Provincia” y “Población”.

Practica Subway

TRANSFORMACIÓN:

Proceso 3:

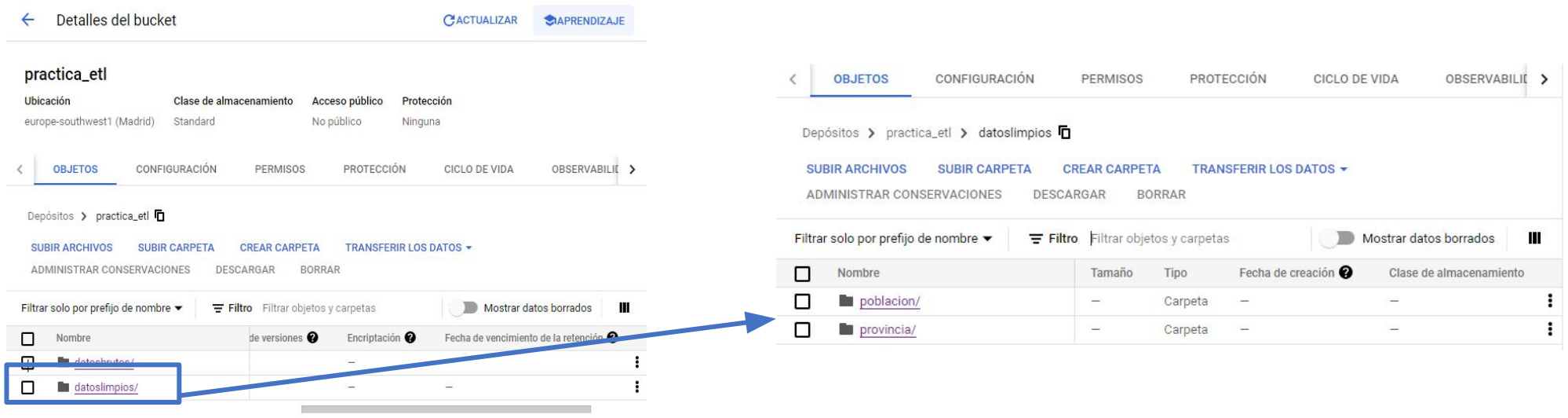
1. Importamos el CSV que es una lista de URLs y ejecutamos un next(reader) para saltarnos la cabecera
2. Ejecutamos un “for” para que lea cada línea de la lista que corresponde a una URL del INEN. Como no sabemos la cantidad de registros que podría haber, vamos a cargar un fichero por URL en GCloud. A la fila le asignamos una variable para leer cada URL.
3. Convertimos los datos leídos de la URL (que es un JSON) en un diccionario.
4. Hacemos los mismos pasos que hemos realizado en el proceso 1, donde también teníamos los datos procedente de una URL, con la diferencia de que en este caso tenemos una columna más que es total edad, que filtraremos por el valor “**todas las edades**”, valor que agrega la información para no duplicarla.
5. Al final realizaremos la comprobación de uno de los dataframe que hemos generado a partir de una URL.

Practica Subway

CARGA:

La carga de datos es el último paso del proceso ETL. El destino final de los datos procesados ha sido almacenado en archivos JSON que hemos subido a la nube en la carpeta de datos limpios.

Los datos se han almacenado dentro del BUCKET “practica_etl” en la carpeta “datos limpios”, como los datos del CSV son distintos a los datos del HTML y los datos transformados desde el fichero EXCEL/CSV se han creado dos carpetas “Población” y “Provincia” (El primero para el CSV y el segundo para el resto).

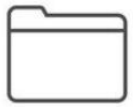


Practica Subway

CARGA:

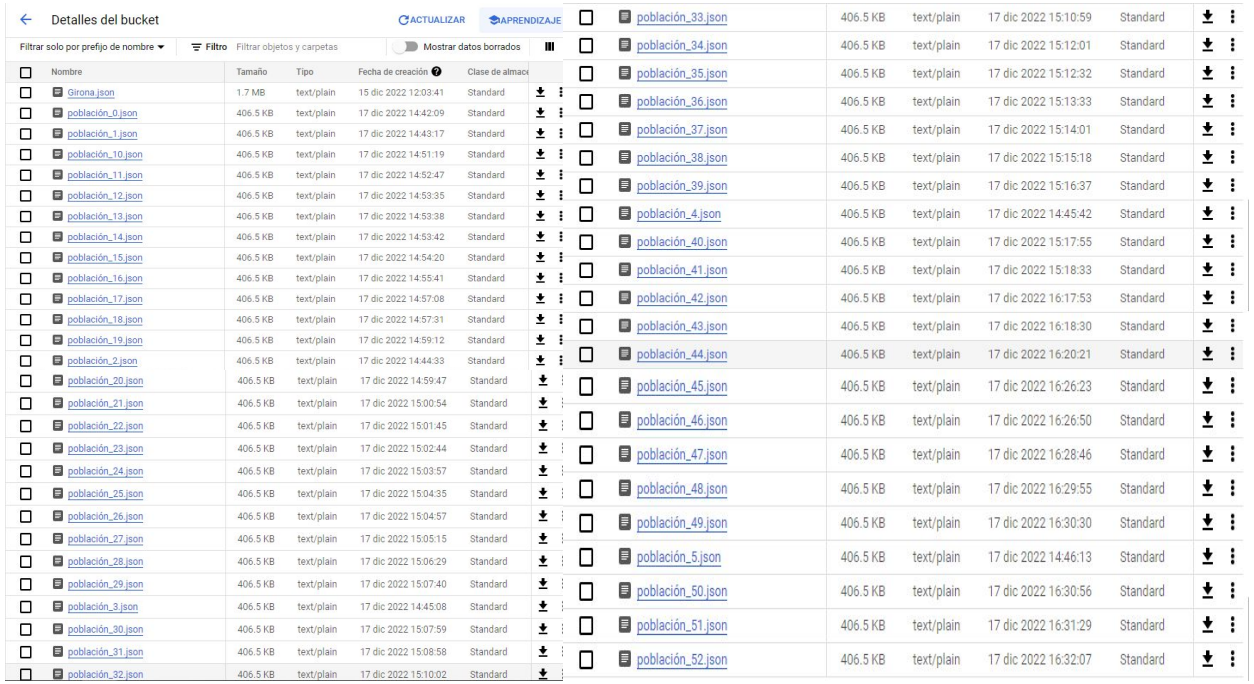
La carga de datos es el último paso del proceso ETL. El destino final de los datos procesados ha sido almacenado en archivos JSON que hemos subido a la nube en la carpeta de datos limpios.

Población



Girona.json

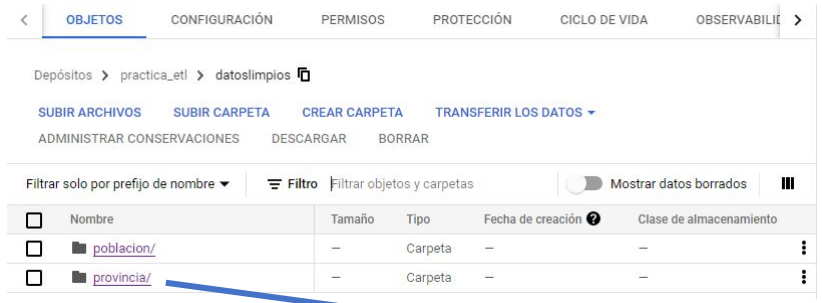
Provincia_0.json

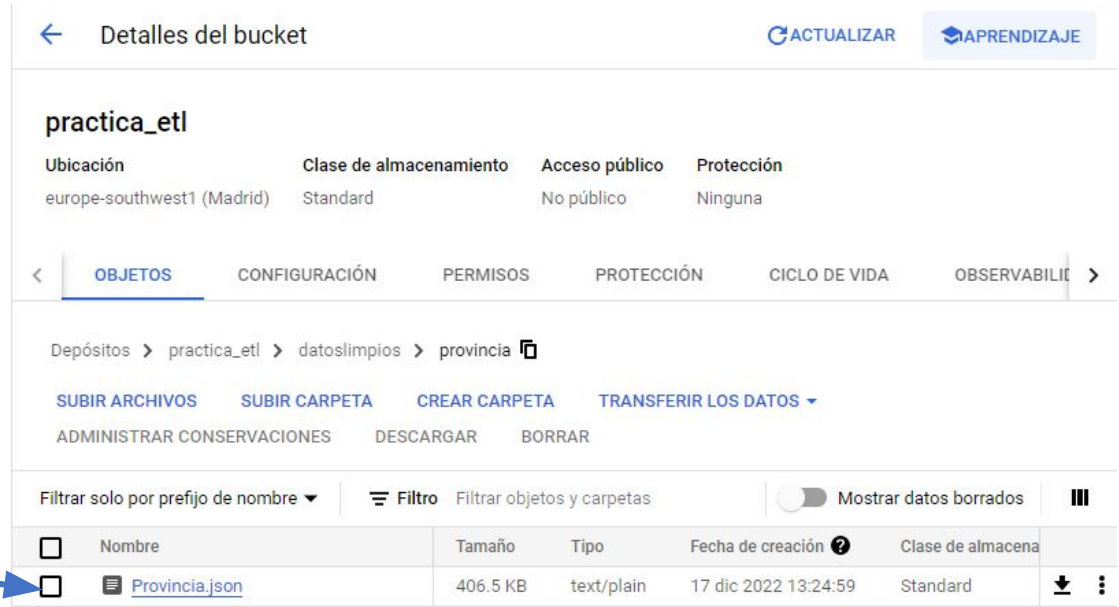


Practica Subway

CARGA:

Provincia





Provincia.json

Practica Subway

CÓDIGO:

El código desarrollado para realizar los pasos está en el documento COLAB de Google que se adjunta en la practica, en el viene todo lo explicado en los puntos anteriores.

Los puntos a destacar son:

- JSON, la normalización de los datos y la conversión a columnas de los ejercicios:

```
df_datosanyo_a = pd.json_normalize(data, record_path =['Data'], meta=['COD'])
df_datosvari_a = pd.json_normalize(data, record_path =['MetaData'], meta=['COD'])
```

```
df_datosanyo = df_datosanyo_a.pivot(index = 'COD', columns= 'Anyo', values = 'Valor' )
df_datosvari = df_datosvari_a.pivot(index = 'COD', columns= 'Variable.Nombre', values = 'Nombre' )
```

```
dftot = pd.merge(df_datosvari,df_datosanyo, left_on='COD', right_on='COD')
```

Practica Subway

CÓDIGO:

- JSON, el filtrado de los datos por sexo:

```
dflimp1=dflimp[dflimp['Sexo']=='Total']
```

- CSV, la creación de los datos locales y datos no locales.

```
prov=df_raw["provincia"].str.split('/', expand=True)
```

```
prov.columns = ['prov_loc', 'Provincia']
```

```
prov.columns
```

```
Index(['prov_loc', 'Provincia'], dtype='object')
```

```
df = pd.concat([df_raw, prov], axis=1)
```

- CSV, relleno de datos para las columnas no locales:

```
import numpy as np
```

```
df1['provNueva']= np.where(df1['Provincia'].isnull(), df1['prov_loc'], df1['Provincia'] )
```

```
df1['poblNueva']= np.where(df1['ciudad_trf'].isnull(), df1['ciudad_loc'], df1['ciudad_trf'] )
```

Practica Subway

CÓDIGO:

- EXCEL/CSV, lectura del las URLs para convertirlas en diccionario:

```
cont = 0
import csv
#Datos de origen, fichero CSV con filas de URLS del INEN.
with open('/content/drive/MyDrive/Proyecto_final_ETL/URL_JSON.csv', newline='') as File:
    reader = csv.reader(File)
    next(reader, None)
    for row in reader:
            print(cont)
            archivo = 'población_' + str(cont) + '.json'
            url1= row[0]


            json_1 = requests.get(url1).text

            data = json.loads(json_1)
```

- EXCEL/CSV, filtrado por “Totales Edad”:

```
df_datosvari_2 = df_datosvari.reindex(columns=['Municipios','Sexo','Nacionalidad','Totales de edad'])
dftot = pd.merge(df_datosvari_2,df_datosanyo, left_on='COD', right_on='COD')

dflimp = dftot

dflimp1=dflimp[dflimp['Sexo']=='Total']
dflimp1=dflimp1[dflimp1['Totales de edad']=='Todas las edades']

dflimp1.drop(['Sexo','Totales de edad'], axis = 1, inplace=True)
```

Practica Subway

OBJETOS:

Los objetos utilizados en cada parte son los siguientes:

Objetos practica

JSON

OBJETO	TIPO
json_1	OBJETO
data	LISTA DE DIC
df_datosanyo_a	DF
df_datosvari_a	DF
df_datosanyo	DF
df_datosvari	DF
df_datosvari_2	DF
dftot	DF
dflimp	DF
dflimp1	DF
dflimp2	DF
dflimp3	DF
dflimp4	DF
dflimp5	DF
jslimp5	TXTJSON

CSV

OBJETO	TIPO
df_raw	DF
prov	DF
prov.columns	OBJETO
df	DF
pobl	DF
pobl.columns	OBJETO
df1	DF
df1['provNueva']	DF
df1['poblNueva']	DF
df1limp	TXTJSON

EXCEL/CSV

OBJETO	TIPO
reader	LISTA
row	OBJETO
json_1	OBJETO
data	LISTA DE DICCIONARIOS.
df_datosanyo_a	DF
df_datosvari_a	DF
df_datosanyo	DF
df_datosvari	DF
df_datosvari_2	DF
dftot	DF
dflimp	DF
dflimp1	DF
dflimp2	DF
dflimp3	DF
dflimp4	DF
jslimp4	TXTJSON