DOCUMENTACIÓN SEMANA 2 PIA

Grupo 5

Métodos de extracción de datos y herramientas empleadas:

Los datos extraídos en nuestro script se enfocan en la opción elegida por el usuario.

```
menu = """
-------------------------------------Menú-------------------------------------
| 1. Determinar si un asteroide es potencialmente peligroso (se necesita ID) |
| 2. Consultar sobre NEOs (se necesita ID)                                   |
| 3. Consultar IDs en la base de datos                                       |
| 4. Salir                                                                   |
------------------------------------------------------------------------------
"""
```

1. En caso de que este elija la primera opción, se extrae del API de datos del asteroide consultado el valor correspondiente a la clave “is_potentially_hazardous_asteroid” (la cual indica si el asteroide causara daños significativos si este golpea la Tierra) y su nombre. Aquí se usó la herramienta del ciclo if para que en caso de que si fuera peligroso, imprimiera un mensaje acorde.

2. Si se elige la segunda opción, el usuario recibirá datos detallados sobre el asteroide en cuestión. En la función “consultar_datos” se crea un diccionario que guarda la información del nombre del asteroide, su ID, la magnitud absoluta H y su diámetro. Luego el programa le pregunta al usuario si desea guardar los datos en un archivo txt para futura consulta.

3. En cuanto a la tercera opción, se permite al usuario consultar un intervalo de IDs haciendo uso de la función “lista_ids”. La función “lista_ids” permite al usuario consultar un rango de IDs de asteroides en la base de datos de NEO de la NASA. El usuario debe ingresar un valor inicial y final para definir el intervalo de IDs que desea consultar. Luego, la función realiza una solicitud a la API para obtener los datos correspondientes a esos IDs.

Técnicas de limpieza aplicadas:

Se realizó una evaluación detallada de la estructura y consistencia de los datos proporcionados por la API de objetos cercanos a la Tierra (NEO) de la NASA. Al analizar las respuestas en formato JSON, se determinó que los datos están bien organizados y no presentan campos nulos, inconsistencias, redundancias ni ruido evidente.

Por esta razón, no fue necesario aplicar procesos de limpieza complejos como eliminación de duplicados, imputación de valores faltantes o transformación de formatos. Sin embargo, se llevó a cabo una validación manual de los campos relevantes para asegurar que los valores extraídos fueran los esperados.

Estructura de los datos optimizados y su diseño lógico:

Se utilizó un diccionario compuesto para mostrar los datos detallados de un asteroide determinado. Además, el programa le da la opción al consultante de guardar dicha información. En el futuro, buscaríamos ofrecer la opción de que el usuario tenga un archivo de los asteroides que guste donde se pueda agregar al archivo de texto (con un append()) la información de otros asteroides.

Cosas agregadas al programa:

```
class Manejo_de_api:
    # Función que conecta el programa con el API de NEO de la NASA
    # Determina si el asteroide es peligroso o no
    def determinar_peligro(API_KEY, asteroid_id):
        url= f"https://api.nasa.gov/neo/rest/v1/neo/{asteroid_id}?api_key={API_KEY}"
        respuesta= requests.get(url)
        if respuesta.status_code == 200:
            datos = respuesta.json()
            nombre = datos["name"]
            print(f"Nombre del asteroide {asteroid_id}: {nombre}")
            es_peligroso = datos["is_potentially_hazardous_asteroid"]
            if es_peligroso==True:
                print(f"\n{nombre} es un asteroide potencialmente peligroso.")
            else:
                print(f"\n{nombre} NO es un asteroide potencialmente peligroso.")
        else:
            print("Error: no se pudo conectar con la API.")

    # Esta función permite que el usuario visualice datos de un asteroide
    # Le da la opción de crear un archivo txt con los datos

    def consultar_datos(API_KEY, asteroid_id):
        url= f"https://api.nasa.gov/neo/rest/v1/neo/{asteroid_id}?api_key={API_KEY}"
        respuesta= requests.get(url)
        if respuesta.status_code == 200:
            datos = respuesta.json()
            # Se crea un diccionario
            datos_detallados = dict()
            datos_detallados["NOMBRE"]= datos["name"]
```

La diferencia principal entre el script anterior es que ahora las funciones se encuentran dentro de una clase. La razón de este cambio es que el menú iba a tener más opciones y consideramos que es mejor trabajar con las funciones si se encuentran dentro de un objeto.

Para garantizar la validez de las entradas del usuario, se implementó una validación con expresiones regulares. Se usó re.fullmatch(r"\d+", asteroid_id) para asegurarse de que el ID ingresado contenga únicamente dígitos, evitando errores en las solicitudes a la API. Aparte de esto. se incluyó una excepción extra para validar los intervalos de la opción 3.