Proyecto ETL NewsAPI

1.-Resumen

Para lograr el proyecto fue necesario ocupar las siguientes herramientas

Postman versión online(test de APIS)

Python 3.12.7 (lenguaje de programación para consumo de la api elegido)

MariaDB 10.4.32 (Capa de persistencia para los datos)

Drawio online (Herramienta para creación de diagramas)

3.-Explicación del código

-Imports necesarios para el proceso de datos

request.-provee funciones para conexion con APIS

json.-Permite convetir la respuesta de APIS en formato json

mysql.connector.-Provee el driver para conectar al motor de base de datos mysql

datetime.- Permite el trabajo con fechas, se ocupa para dar fomato a la fecha de noticias

```
####################
##1.-IMPORTS    ###
####################

import requests as r
import json as json
import mysql.connector
from datetime import datetime
```

-Conexion al motor de base de datos Mysql

```
#########################
##2.-CONEXION MYSQL    ###
#########################
conexion=mysql.connector.connect(user='root',password='',host='127.0.0.1',database='aplatam',port='3306')
cursor=conexion.cursor()
```

Se genera una conexión a un motor de base de datos local de pruebas y se genera un curso para ejecutar consultas.

-Conexión a api newsapi.org

```
##########################################
##3.-CONSUMO API E INSERCION DE DATOS    ###
##########################################
#step1#
args={'q':'bitcoin','apiKey':'ff9ab8a61e0b4c718d6edf43fe2eaa4a'} #se agrega diccionario de parametros
#url='https://newsapi.org/v2/everything?q=bitcoin&apiKey=ff9ab8a61e0b4c718d6edf43fe2eaa4a'
url='https://newsapi.org/v2/everything?'
response=r.get(url,params=args) #validar la url enviada
#print(response.url)
#step2#
if response.status_code==200:
    #content=response.content
    #print(content)
    #step3#
    response_json=json.loads(response.text)
    #step4#
    cursor=conexion.cursor()
```

Step 1

Se crea la variable args, que es un diccionario la cual contiene las variables que se envían por metodo get, a la url del api.

La variable response obtiene la respuesta de la api ejecutando el metodo get, incluyendo la url y los parametros de peticion.

Step 2

Se evalúa en un if el atributo status_code para saber si fue exitosa la petición.

Step 3

Cuando la respuesta es exitosa se convierte el atributo text del response a un objeto json.

Step 4

Se genera el cursor de la conexion a mysql para iniciar el insert de datos sobre el motor.

```
    #step 5#
    for field_dict in response_json['articles']:
        #print(field_dict['author'])
        #print(field_dict['source']['name'])
        #print(field_dict['publishedAt'])

        author=field_dict['author']
        if author is None:
            author='Sin author'
        source_name=field_dict['source']['name']
        published_date=datetime.strptime(field_dict['publishedAt'][:10],"%Y-%m-%d")
        #print(type(published_date)) se valida el tipo de dato

        nuevo_registro=(author,source_name,published_date) #se crea una tupla con los parametros del query

        query='''INSERT INTO noticias_temp (author,source_name,published_date)
                 VALUES (%s,%s,%s);'''
        cursor.execute(query,nuevo_registro)
```

Step 5

Se genera un diccionario con el objetivo json articles.

Se extrae el author de la noticia, si es null se genera el dato ‘Sin author’.

Se extrae el source_name de la noticia.

Se extrae published_date con el formato de fecha año-mes-dia.

Se genera una tupla del nuevo registro con lo datos antes extraidos.

Se preparar el query de insert.

Se ejecuta el insert.

```
#creación catálogo de datos
#step 6#
query='''INSERT INTO cat_source (source_name)
           SELECT DISTINCT source_name
           FROM aplatam.noticias_temp;'''
cursor.execute(query)
#step 7#
query='''INSERT INTO cat_author (author)
           SELECT DISTINCT author
           FROM aplatam.noticias_temp;'''
cursor.execute(query)
```

Step 6 y 7

Generan catalogos source y author

```
#step 8#
query='''INSERT INTO noticias (id_author,id_source_name,published_date)
         SELECT ca.id AS id_author
               ,cs.id AS id_source_name
               ,nt.published_date
           FROM aplatam.noticias_temp AS nt
             LEFT JOIN cat_author AS ca
                    ON nt.author=ca.author
             LEFT JOIN cat_source AS cs
                    ON nt.source_name=cs.source_name;'''
cursor.execute(query)
```

Step 8

Se crea y ejecuta consulta insert de tabla noticias normalizada

```
#step 9#
query='''INSERT INTO abt_noticias_resumen( id_author
                                          ,id_source_name
                                          ,total_news
                                          ,min_published_date
                                          ,max_published_date
                                          ,published_date)
         SELECT n.id_author
               ,n.id_source_name
               ,COUNT(*) AS total_news
               ,MIN(published_date) AS min_published_date
               ,MAX(published_date) AS max_published_date
               ,n.published_date
           FROM noticias AS n
           GROUP BY n.id_author
                   ,n.id_source_name
                   ,n.published_date;'''
cursor.execute(query)
```

Step 9

Se crea y ejecuta consulta insert de tabla de analítica básica de resumen de noticias, grado de granularidad published_date, author y source_name.

```
#step 10#
conexion.commit() #SE CONFIRMAN LOS CAMBIOS EN LA BASE DE DATOS
conexion.close()  #SE CIERRA LA CONEXION
```

Step 10

Se confirma cambios en la base de datos

Se cierra la conexión a mysql