```
from tabulate import tabulate
import random
import json
import yaml
import time
import pandas as pd


from google.colab import drive
drive.mount('/content/drive')

ruta = "/content/drive/MyDrive/Actividad12"

import sys
sys.path.append(ruta)

import os

path = "/content/drive/MyDrive/Actividad12"
```

```
Mounted at /content/drive
```

```
datos = []

ApP = ["Ayon", "Dimas", "Gonzales", "Tello", "Dimas", "Ruvalcaba", "Cisnersos", "Valenzuela",
       "Arenas", "Olivas", "Cassas", "Sanchez", "De la luz", "Quintero", "Garcia"]

ApM = ["Ruiz", "Quinones", "Jimenez", "Perez", "Ayala", "Galicia", "Robles", "Gutierres", "Meraz",
       "Espinoza", "Crespo", "De la torre", "Nunes", "Galindo", "Peralta"]

sexo = ["Hombre", "Mujer"]

NombresH = ["Andres", "Ricardo", "Felipe", "Jose", "Eduardo", "Kevin", "Bryan", "Sergio", "Ramon",
            "Jair", "Daniel", "Yessi", "Jesus", "Gerardo", "Fernando"]

NombresM = ["Keyla", "Alicia", "Marisela", "Sofia", "Anna", "Daniela", "Sujey", "Melissa", "Karolina",
            "Silvia", "Maria", "Arlet", "Florinda", "Karely", "Karen"]


def aleatorio():
    i = 1
    while i <= 10:
        genero = random.choice(sexo)
        if genero == "Mujer":
            datos.append({"ID": random.randrange(300000, 399999), "Edad": random.randrange(18, 60),
                          "Nombre": random.choice(NombresM), "Apellido paterno": random.choice(ApP),
                          "Apellido materno": random.choice(ApM), "Sexo": "Mujer"})
        else:
            datos.append({"ID": random.randrange(300000, 399999), "Edad": random.randrange(18, 60),
                          "Nombre": random.choice(NombresH), "Apellido paterno": random.choice(ApP),
                          "Apellido materno": random.choice(ApM), "Sexo": "Hombre"})
        i = i + 1
    print("\nSe han generado los datos\n")

def generar_archivo_drive(datos, carpeta_destino, nombre_archivo, extension):
    try:
        ruta_archivo = os.path.join(carpeta_destino, f"{nombre_archivo}.{extension}")

        if extension == "excel":
            df = pd.DataFrame(datos)
            df.to_excel(ruta_archivo, index=False, header=True)
            print("Archivo de Excel generado y guardado en Google Drive.")
        elif extension == "txt":
            with open(ruta_archivo, "w") as file:
                tabla = tabulate(datos, headers="keys", tablefmt="grid")
                file.write(tabla)
            print("Archivo de texto generado y guardado en Google Drive.")
        elif extension == "csv":
            df = pd.DataFrame(datos)
            df.to_csv(ruta_archivo, index=False, header=True)
            print("Archivo CSV generado y guardado en Google Drive.")
        elif extension == "md":
            with open(ruta_archivo, "w") as f:
                f.write(tabulate(datos, headers="keys", tablefmt="pipe"))
            print("Archivo MarkDown generado y guardado en Google Drive.")
```

```
        except Exception as e:
            print(f"Error al generar el archivo en Google Drive: {e}")

    def cargar_archivo():
        try:
            nombre_archivo = input("Digite el nombre del archivo a cargar: ")
            ruta_archivo = os.path.join(path, nombre_archivo)

            with open(ruta_archivo, "r") as file:
                global datos
                extension = nombre_archivo.split(".")[-1]
                if extension == "json":
                    datos = json.load(file)
                    print("Archivo cargado con exito.")
                elif extension == "csv":
                    datos = pd.read_csv(file)
                    print("Archivo cargado con exito.")

        except FileNotFoundError:
            print("El archivo no se encontro.")
        except Exception as e:
            print(f"Error al cargar el archivo: {e}")

    def eliminar():
        try:
            delete = int(input("Digite el ID que desea borrar: "))
            for i in range(len(datos)):
                if delete == datos[i]["ID"]:
                    datos.pop(i)
                    print("\nSe eliminaron los datos\n")
                    return
            print("\nNo se encontro ninguna coincidencia para el ID proporcionado\n")
        except ValueError:
            print("\nEsto no es válido\n")


    def buscar():
        encontrado = False

        try:
            search = int(input("ID que desea buscar: "))
        except ValueError:
            print("\nVuelva a intentar ingresando un número valido\n")
            return

        for persona in datos:
            if search == persona["ID"]:
                print("\tDatos del trabajador")
                print("ID: ", persona["ID"])
                print("Nombre: ", persona["Nombre"])
                print("Apellido paterno: ", persona["Apellido paterno"])
                print("Apellido materno: ", persona["Apellido materno"])
                print("Edad: ", persona["Edad"])
                print("Sexo: ", persona["Sexo"])
                encontrado = True

        if not encontrado:
            print("\nNo se encontró ninguna coincidencia para el ID proporcionado\n")


    def buscar_appat():
        encontrado = False

        try:
            search = str(input("Apellido paterno que desea buscar: "))
        except ValueError:
            print("\nVuelva a intentar ingresando un apellido valido\n")
            return

        for persona in datos:
            if search == persona["Apellido paterno"]:
                print("\tDatos del trabajador")
                print("ID: ", persona["ID"])
                print("Nombre: ", persona["Nombre"])
                print("Apellido paterno: ", persona["Apellido paterno"])
                print("Apellido materno: ", persona["Apellido materno"])
                print("Edad: ", persona["Edad"])
```

```
            print("Sexo: ", persona["Sexo"])
            encontrado = True

    if not encontrado:
        print("\nNo se encontraron coincidencias para el apellido paterno proporcionado\n")


def ordenar():
    datos.sort(key=lambda x: x["ID"])
    print("Datos ordenados por ID.")


def deshacer():
    datos.clear()
    print("\nTodos los registros han sido eliminados\n")


def menu():
    while True:
        print("\tMENU")
        print("1.- Agregar (automático 10)")
        print("2.- Eliminar {ID}")
        print("3.- Imprimir lista (tabla)")
        print("4.- Buscar {ID}")
        print("5.- Buscar {appat} todas las coincidencias")
        print("6.- Ordenar {ID}")
        print("7.- Generar archivo {ID}")
        print("\ta) excel")
        print("\tb) txt")
        print("\tc) csv")
        print("\td) MarkDown")
        print("8.- Cargar archivo {ID}")
        print("9.- Imprimir archivo {ID}")
        print("10.- Borrar Toda la lista {ID}")
        print("0. SALIR")

        try:
            op = int(input("Digita una opcion: "))
            if op not in range(11):
                raise ValueError
        except ValueError:
            print("\nLa opción no es valida\n")
        else:
            if op == 1:
                print("\n")
                aleatorio()
                generar_archivo_drive(datos, path, "datos_generados", "csv")

            elif op == 2:
                print("\n")
                eliminar()

            elif op == 3:
                print("\n")
                print(tabulate(datos, headers="keys"))

            elif op == 4:
                print("\n")
                buscar()

            elif op == 5:
                print("\n")
                buscar_appat()

            elif op == 6:
                ordenar()

            elif op == 7:
                print("\n")
                generar_archivo_drive(datos, path, "archivo_generado", input("Digite la extension del archivo (excel, txt, csv, md): ").lower

            elif op == 8:
                print("\n")
                cargar_archivo()

            elif op == 9:
                print("\n")
                imprimir_archivo()
```

```
        elif op == 10:
            print("\n")
            deshacer()

        elif op == 0:
            break


menu()
```

```
         MENU
1.- Agregar (automático 10)
2.- Eliminar {ID}
3.- Imprimir lista (tabla)
4.- Buscar {ID}
5.- Buscar {appat} todas las coincidencias
6.- Ordenar {ID}
7.- Generar archivo {ID}
        a) excel
        b) txt
        c) csv
        d) MarkDown
8.- Cargar archivo {ID}
9.- Imprimir archivo {ID}
10.- Borrar Toda la lista {ID}
0. SALIR
Digita una opcion: 7


Digite la extension del archivo (excel, txt, csv, md): cvs
        MENU
1.- Agregar (automático 10)
2.- Eliminar {ID}
3.- Imprimir lista (tabla)
4.- Buscar {ID}
5.- Buscar {appat} todas las coincidencias
6.- Ordenar {ID}
7.- Generar archivo {ID}
        a) excel
        b) txt
        c) csv
        d) MarkDown
8.- Cargar archivo {ID}
9.- Imprimir archivo {ID}
10.- Borrar Toda la lista {ID}
0. SALIR
Digita una opcion: 7


Digite la extension del archivo (excel, txt, csv, md): md
Archivo MarkDown generado y guardado en Google Drive.
        MENU
1.- Agregar (automático 10)
2.- Eliminar {ID}
3.- Imprimir lista (tabla)
4.- Buscar {ID}
5.- Buscar {appat} todas las coincidencias
6.- Ordenar {ID}
7.- Generar archivo {ID}
        a) excel
        b) txt
        c) csv
        d) MarkDown
8.- Cargar archivo {ID}
9.- Imprimir archivo {ID}
10.- Borrar Toda la lista {ID}
0. SALIR
Digita una opcion: 0
```