BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA.

CU2.

Facultad de ciencias de la computación

ASIGNATURA:

Introducción a la ciencia de datos

DOCENTE:

JAIME ALEJANDRO ROMERO SIERRA

ALUMNO:

LUNA DAMIAN ANGEL GABRIEL

LINK DE GITHUB:

https://github.com/Kittypiupiu/entrega-limpieza-de-datos-angel

FECHA DE ENTREGA:

20/10/2025

Descripción general del conjunto de datos

El Diabetes Health Indicators Dataset es una base de datos desarrollado a partir de los registros de salud recopilados por el Centers for Disease Control and Prevention (CDC) en Estados Unidos.
Su principal objetivo es analizar y comprender los factores asociados con la aparición de la diabetes y la prediabetes en la población adulta.

Este dataset reúne información de miles de participantes, obtenida mediante encuestas nacionales que recogen tanto aspectos clínicos como de estilo de vida.
El conjunto contiene más de100 000 observaciones, donde cada fila representa a una persona y cada columna a una característica relacionada con su salud o su entorno sociodemográfico. Entre las variables más relevantes se incluyen indicadores como el índice de masa corporal (IMC), la presencia de presión arterial alta, colesterol elevado, nivel de actividad física, consumo de tabaco y alcohol, así como condiciones médicas previas, como enfermedades cardíacas.

También incorpora variables demográficas importantes, como la edad, el sexo, el nivel educativo, los ingresos económicos y la raza o grupo étnico.
El valor central del dataset es una variable binaria o categórica que indica si la persona tiene diabetes, se encuentra en estado de prediabetes o no padece la enfermedad, lo que permite su uso en tareas de clasificación y predicción médica.
El Diabetes Health Indicators Dataset ofrece una visión integral del estado de salud de una población amplia y diversa, constituyéndose como una herramienta útil para el análisis estadístico, la educación en ciencia de datos y el desarrollo de estrategias preventivas orientadas a mejorar la calidad de vida y reducir la incidencia de la diabetes.

Significado de cada columna de la base de datos:

Columna	Descripción / Significado
age	Edad del individuo (en años o en rangos de edad).
gender	Sexo o género del participante (por ejemplo: masculino, femenino, otro).
ethnicity	Grupo étnico o raza al que pertenece la persona (por ejemplo: blanca, afroamericana, hispana, asiática, etc.).
education_level	Nivel máximo de estudios alcanzado (por ejemplo: primaria, secundaria, universidad, posgrado).
income_level	Nivel de ingresos del hogar o del individuo, generalmente expresado en rangos (bajo, medio, alto).
employment_status	Situación laboral actual (empleado, desempleado, jubilado, estudiante, etc.).
smoking_status	Indica si la persona fuma actualmente, fumó en el pasado o nunca ha fumado.

Columna	Descripción / Significado
alcohol_consumption_per_week	Cantidad promedio de consumo de alcohol por semana (en unidades o bebidas estándar).
physical_activity_minutes_per_week	Minutos de actividad física moderada o vigorosa realizados semanalmente.
diet_score	Puntuación general de calidad de la dieta, basada en hábitos alimenticios saludables.
sleep_hours_per_day	Promedio de horas de sueño que la persona tiene por día.
screen_time_hours_per_day	Tiempo promedio frente a pantallas (televisión, computadora, celular) al día.
family_history_diabetes	Indica si existen antecedentes familiares de diabetes (sí/no).
hypertension_history	Antecedentes personales de hipertensión arterial (sí/no).
cardiovascular_history	Indica si la persona ha tenido enfermedades cardiovasculares previas (sí/no).
bmi	Índice de Masa Corporal (Body Mass Index), calculado como peso (kg) / altura² (m^2).
waist_to_hip_ratio	Relación entre la circunferencia de cintura y la de cadera, indicador de distribución de grasa corporal.
systolic_bp	Presión arterial sistólica (valor superior, medido en mmHg).
diastolic_bp	Presión arterial diastólica (valor inferior, medido en mmHg).
heart_rate	Frecuencia cardíaca en reposo (latidos por minuto).
cholesterol_total	Concentración total de colesterol en sangre (mg/dL).
hdl_cholesterol	Colesterol de lipoproteínas de alta densidad (HDL, “colesterol bueno”).
ldl_cholesterol	Colesterol de lipoproteínas de baja densidad (LDL, “colesterol malo”).
triglycerides	Nivel de triglicéridos en sangre (mg/dL).
glucose_fasting	Nivel de glucosa en ayunas (mg/dL).
glucose_postprandial	Nivel de glucosa en sangre después de comer (mg/dL).
insulin_level	Concentración de insulina en sangre (μU/mL o pmol/L).
hba1c	Porcentaje de hemoglobina glicosilada (HbA1c), que refleja el promedio de glucosa en sangre en los últimos 2–3 meses.
diabetes_risk_score	Puntaje estimado del riesgo de desarrollar diabetes, calculado a partir de los indicadores de salud.
diabetes_stage	Etapa o nivel de avance de la diabetes (por ejemplo: no diabético, prediabético, diabético).
diagnosed_diabetes	Indica si la persona ha sido diagnosticada clínicamente con diabetes por un profesional de salud (sí/no).

Proceso de limpieza de datos:

la base de datos es sobre indicadores de riesgo de diabetes que contiene una muestra de 100000 datos de diferentes personas mayores de edad,
la base de datos contiene distintos valores nulos y extraños que tienen que ser limpiados, a continuacion se muestra el paso a paso de como limpie la base de datos

markdown

importar librerias y cargar base de datos sucia

```
import pandas as pd
df = pd.read_csv("C:/Users/aaron/Downloads/df_sucio.csv")
df.head()
```

[33] Python

	age	gender	ethnicity	education_level	income_level	employment_status	smoking_status	alcohol_consumption_per_week	physical_activity_minutes_per_week	diet_score	...	hdl_cholesterol	ldl_c
0	58	Male	Asian	Highschool	Lower-Middle	Employed	Never	0.0	215.0	5.7	...	41.0	
1	48	Female	White	Highschool	Middle	Employed	Former	1.0	143.0	NaN	...	55.0	
2	60	Male	Hispanic	Highschool	Middle	Unemployed	Never	1.0	57.0	6.4	...	NaN	
3	74	Female	Black	Highschool	Low	Retired	Never	0.0	49.0	3.4	...	50.0	
4	NaN	Male	White	Graduate	Middle	Retired	Never	1.0	109.0	7.2	...	52.0	

5 rows × 31 columns

se verifica el tipo de dato registrado en cada columna para realizar el cambio de tipo en caso de ser necesario

```
df.info()
```

[34] Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100428 entries, 0 to 100427
Data columns (total 31 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   age                               97416 non-null  object
 1   gender                            97416 non-null  object
 2   ethnicity                         97416 non-null  object
 3   education_level                   97416 non-null  object
 4   income_level                      97416 non-null  object
 5   employment_status                 97416 non-null  object
 6   smoking_status                    97416 non-null  object
 7   alcohol_consumption_per_week      97416 non-null  float64
 8   physical_activity_minutes_per_week  97416 non-null  float64
 9   diet_score                        97416 non-null  float64
 10  sleep_hours_per_day               97416 non-null  float64
 11  screen_time_hours_per_day         97416 non-null  float64
 12  family_history_diabetes           97416 non-null  float64
 13  hypertension_history              97416 non-null  float64
 14  cardiovascular_history            97416 non-null  object
 15  bmi                               97416 non-null  object
 16  waist_to_hip_ratio                95475 non-null  float64
 17  systolic_bp                       95474 non-null  float64
 18  diastolic_bp                      97416 non-null  object
 19  heart_rate                        97416 non-null  object
...
 29  diabetes_stage                    97416 non-null  object
 30  diagnosed_diabetes                97416 non-null  object
dtypes: float64(13), object(18)
```

se verifican las columnas con las que se cuenta

```
df.columns
```

[35] Python

```
Index(['age', 'gender', 'ethnicity', 'education_level', 'income_level',
       'employment_status', 'smoking_status', 'alcohol_consumption_per_week',
       'physical_activity_minutes_per_week', 'diet_score',
       'sleep_hours_per_day', 'screen_time_hours_per_day',
       'family_history_diabetes', 'hypertension_history',
       'cardiovascular_history', 'bmi', 'waist_to_hip_ratio', 'systolic_bp',
       'diastolic_bp', 'heart_rate', 'cholesterol_total', 'hdl_cholesterol',
       'ldl_cholesterol', 'triglycerides', 'glucose_fasting',
       'glucose_postprandial', 'insulin_level', 'hba1c', 'diabetes_risk_score',
       'diabetes_stage', 'diagnosed_diabetes'],
      dtype='object')
```

conteo de datos nulos por columna

```
[36] df.isnull().sum()
```
Python

```
age                                  3012
gender                               3012
ethnicity                            3012
education_level                      3012
income_level                         3012
employment_status                    3012
smoking_status                       3012
alcohol_consumption_per_week         3012
physical_activity_minutes_per_week   3012
diet_score                           3012
sleep_hours_per_day                  3012
screen_time_hours_per_day            3012
family_history_diabetes              3012
hypertension_history                 3012
cardiovascular_history               3012
bmi                                  3012
waist_to_hip_ratio                   4953
systolic_bp                          4954
diastolic_bp                         3012
heart_rate                           3012
cholesterol_total                    3012
hdl_cholesterol                      3012
ldl_cholesterol                      3012
triglycerides                        3012
glucose_fasting                      3012
...
hba1c                                3012
diabetes_risk_score                  3012
diabetes_stage                       3012
diagnosed_diabetes                   3012
dtype: int64
```
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

tamaño del dataframe original

```
[37] df.shape
```
Python

```
(100428, 31)
```

visualizacion de datos duplicados

```
[38] df.duplicated().sum()
```
Python

```
27
```

como solo son 27 valores duplicados, los elimmino directamente del dataframe original

```
[39] df= df.drop_duplicates()
```
Python

se comprueba que se eliminaron

```
[40] df.duplicated().sum()
```

Python

```
0
```

se muestran los valores unicos por columna en el dataframe

```
[41] for i in df:
         print(f"los valores unicos en la columna {i} son: ", df[i].unique())
         print()
```

Python

```
los valores unicos en la columna age son:  ['58' '48' '60' '74' nan '46' '75' '62' '42' '59' '43' '54' '19' '22' '41'
 '34' '55' '35' '27' '73' '51' '52' '32' '56' '40' '45' '80' '50' '33'
 '63' '30' '53' '26' '38' '67' '44' '39' '66' '65' '37' '47' '72' '49'
 '18' '89' '31' '68' '28' '85' '25' 'Auto%#' '71' '24' '29' '61' '69' '57'
 '36' '90' '64' '87' '20' '84' '78' '76' '70' '83' '79' '23' '81' '77'
 '21' '86' '82' '88']

los valores unicos en la columna gender son:  ['Male' 'Female' nan 'Other']

los valores unicos en la columna ethnicity son:  ['Asian' 'White' 'Hispanic' 'Black' nan 'Other']

los valores unicos en la columna education_level son:  ['Highschool' 'Graduate' 'Postgraduate' 'No formal' nan]

los valores unicos en la columna income_level son:  ['Lower-Middle' 'Middle' 'Low' 'Upper-Middle' nan 'High' 'Auto%#']

los valores unicos en la columna employment_status son:  ['Employed' 'Unemployed' 'Retired' nan 'Student']

los valores unicos en la columna income_level son:  ['Lower-Middle' 'Middle' 'Low' 'Upper-Middle' nan 'High' 'Auto%#']

los valores unicos en la columna employment_status son:  ['Employed' 'Unemployed' 'Retired' nan 'Student']

los valores unicos en la columna smoking_status son:  ['Never' 'Former' 'Current' nan]

los valores unicos en la columna alcohol_consumption_per_week son:  [ 0.  1.  2.  3.  6.  5.  4. nan  9.  8.  7. 10.]

los valores unicos en la columna physical_activity_minutes_per_week son:  [215. 143.  57.  49. 109. 124.  53.  75. 114.  86. 118. 167.  nan 105.
  99.  31.  54.  14. 164. 128. 147. 241. 185.  78. 148.  88.  29. 115.
 182. 106. 113.  51. 170.  72. 133.  34.  59. 123.  71.  91.  17. 175.
  30.  39.  77. 121. 240.  80. 188.  85.  56.  61.  47.  62. 180. 135.
...
los valores unicos en la columna diabetes_stage son:  ['Type 2' 'No Diabetes' nan 'Pre-Diabetes' 'Auto%#' 'Gestational' 'Type 1']

los valores unicos en la columna diagnosed_diabetes son:  ['1' '0' nan 'Auto%#']

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

se hace el conteo de los valores incongruentes que en este caso es la palabra Auto%#

```
[11] for nombre in df:
         print(f"En la columna {nombre} los datos incongruentes son: {df[df[nombre] == 'Auto%#'].shape[0]}")
```

✓ 0.2s

Python

```
En la columna age los datos incongruentes son: 1942
En la columna gender los datos incongruentes son: 0
En la columna ethnicity los datos incongruentes son: 0
En la columna education_level los datos incongruentes son: 0
En la columna income_level los datos incongruentes son: 1937
En la columna employment_status los datos incongruentes son: 0
En la columna smoking_status los datos incongruentes son: 0
```

```
En la columna alcohol_consumption_per_week los datos incongruentes son: 0
En la columna physical_activity_minutes_per_week los datos incongruentes son: 0
En la columna diet_score los datos incongruentes son: 0
En la columna sleep_hours_per_day los datos incongruentes son: 0
En la columna screen_time_hours_per_day los datos incongruentes son: 0
En la columna family_history_diabetes los datos incongruentes son: 0
En la columna hypertension_history los datos incongruentes son: 0
En la columna cardiovascular_history los datos incongruentes son: 1959
En la columna bmi los datos incongruentes son: 1953
En la columna waist_to_hip_ratio los datos incongruentes son: 0
En la columna systolic_bp los datos incongruentes son: 0
En la columna diastolic_bp los datos incongruentes son: 1935
En la columna heart_rate los datos incongruentes son: 1950
En la columna cholesterol_total los datos incongruentes son: 0
En la columna hdl_cholesterol los datos incongruentes son: 0
En la columna ldl_cholesterol los datos incongruentes son: 1965
En la columna triglycerides los datos incongruentes son: 1962
En la columna glucose_fasting los datos incongruentes son: 0
...
En la columna hba1c los datos incongruentes son: 0
En la columna diabetes_risk_score los datos incongruentes son: 1949
En la columna diabetes_stage los datos incongruentes son: 1945
En la columna diagnosed_diabetes los datos incongruentes son: 1956
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

se crea un alista de las columnas con valores incongruentes para despues eliminarlos y poder hacer el cambio de tipo de variable en las columnas

```
[12]
lista_con_incon= ["age", "income_level", "cardiovascular_history", "bmi", "diastolic_bp", "heart_rate", "ldl_cholesterol", "triglycerides", "glucose_postprandial", "insulin_level"
                  , "diabetes_stage", "diagnosed_diabetes","diabetes_risk_score"]
```

✓ 0.0s Python

para eliminar los valores incongruentes los transformé a valores nulos y esos valores nulos los cambié por el valor inmediato anterior

```
[13]
for columna in lista_con_incon:
    df[columna] = df[columna].replace("Auto%#", None)
    df[columna] = df[columna].fillna(method='bfill')
```

✓ 0.2s Python

```
C:\Users\aaron\AppData\Local\Temp\ipykernel_22608\1824905169.py:3: FutureWarning: Series.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.b
  df[columna] = df[columna].fillna(method='bfill')
```

se comprueba que se eliminaron los datos incongruentes

```
[14]
for nombre in df:
    print(f"En la columna {nombre} los datos incongruentes son: {df[df[nombre] == 'Auto%#'].shape[0]}")
```

✓ 0.1s Python

```
En la columna age los datos incongruentes son: 0
En la columna gender los datos incongruentes son: 0
En la columna ethnicity los datos incongruentes son: 0
En la columna education_level los datos incongruentes son: 0
En la columna income_level los datos incongruentes son: 0
En la columna employment_status los datos incongruentes son: 0
En la columna smoking_status los datos incongruentes son: 0
En la columna alcohol_consumption_per_week los datos incongruentes son: 0
En la columna physical_activity_minutes_per_week los datos incongruentes son: 0
En la columna diet_score los datos incongruentes son: 0
En la columna sleep_hours_per_day los datos incongruentes son: 0
```

```
En la columna sleep_hours_per_day los datos incongruentes son: 0
En la columna screen_time_hours_per_day los datos incongruentes son: 0
En la columna family_history_diabetes los datos incongruentes son: 0
En la columna hypertension_history los datos incongruentes son: 0
En la columna cardiovascular_history los datos incongruentes son: 0
En la columna bmi los datos incongruentes son: 0
En la columna waist_to_hip_ratio los datos incongruentes son: 0
En la columna systolic_bp los datos incongruentes son: 0
En la columna diastolic_bp los datos incongruentes son: 0
En la columna heart_rate los datos incongruentes son: 0
En la columna cholesterol_total los datos incongruentes son: 0
En la columna hdl_cholesterol los datos incongruentes son: 0
En la columna ldl_cholesterol los datos incongruentes son: 0
En la columna triglycerides los datos incongruentes son: 0
En la columna glucose_fasting los datos incongruentes son: 0
...
En la columna hba1c los datos incongruentes son: 0
En la columna diabetes_risk_score los datos incongruentes son: 0
En la columna diabetes_stage los datos incongruentes son: 0
En la columna diagnosed_diabetes los datos incongruentes son: 0
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

ahora utilizando el mismo metodo, relleno los valores nulos en las demás columnas

```
[15]
for colum in df:
    df[colum]=df[colum].fillna(method='bfill')
```

✓ 0.2s Python

```
C:\Users\aaron\AppData\Local\Temp\ipykernel_22608\1307558410.py:2: FutureWarning: Series.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.b
  df[colum]=df[colum].fillna(method='bfill')
```

se comprueba que ya no hay valores nulos en ninguna columna

```
[16]
df.isnull().sum()
```

✓ 0.0s Python

```
age                                0
gender                             0
ethnicity                          0
education_level                    0
income_level                       0
employment_status                  0
smoking_status                     0
alcohol_consumption_per_week       0
physical_activity_minutes_per_week 0
diet_score                         0
sleep_hours_per_day                0
screen_time_hours_per_day          0
family_history_diabetes            0
hypertension_history               0
cardiovascular_history             0
bmi                                0
waist_to_hip_ratio                 0
systolic_bp                        0
diastolic_bp                       0
heart_rate                         0
cholesterol_total                  0
hdl_cholesterol                    0
ldl_cholesterol                    0
triglycerides                      0
glucose_fasting                    0
...
hba1c                              0
diabetes_risk_score                1
```

como el bfill no rellenó un valor en la columna "diabetes_risk_score", utilice el ffill

```
[17] df["diabetes_risk_score"]=df["diabetes_risk_score"].fillna(method='ffill')
```

✓ 0.0s Python

```
C:\Users\aaron\AppData\Local\Temp\ipykernel_22608\2686900603.py:1: FutureWarning: Series.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.b
  df["diabetes_risk_score"]=df["diabetes_risk_score"].fillna(method='ffill')
```

se comprueba que esta vez ya no hay valores nulos

```
[18] df.isnull().sum()
```

✓ 0.1s Python

```
age                                0
gender                             0
ethnicity                          0
education_level                    0
income_level                       0
employment_status                  0
smoking_status                     0
alcohol_consumption_per_week       0
physical_activity_minutes_per_week 0
diet_score                         0
sleep_hours_per_day                0
screen_time_hours_per_day          0
family_history_diabetes            0
hypertension_history               0
cardiovascular_history             0
bmi                                0
waist_to_hip_ratio                 0
cardiovascular_history             0
bmi                                0
waist_to_hip_ratio                 0
systolic_bp                        0
diastolic_bp                       0
heart_rate                         0
cholesterol_total                  0
hdl_cholesterol                    0
ldl_cholesterol                    0
triglycerides                      0
glucose_fasting                    0
...
hba1c                              0
diabetes_risk_score                0
diabetes_stage                     0
diagnosed_diabetes                 0
dtype: int64
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

ahora traduciré las columnas a español

```
[19] traduccion_columnas = {
        'age': 'edad',
        'gender': 'género',
        'ethnicity': 'etnia',
        'education_level': 'nivel_educativo',
        'income_level': 'nivel_ingresos',
        'employment_status': 'estado_laboral',
        'smoking_status': 'hábito_fumar',
        'alcohol_consumption_per_week': 'consumo_alcohol_semana',
        'physical_activity_minutes_per_week': 'actividad_física_minutos_semana',
        'diet_score': 'puntaje_dieta',
        'sleep_hours_per_day': 'horas_sueño_por_día',
        'screen_time_hours_per_day': 'horas_pantalla_día',
        'family_history_diabetes': 'antecedente_familiar_diabetes',
        'hypertension_history': 'antecedente_hipertensión',
        'cardiovascular_history': 'antecedente_cardiovascular',
        'bmi': 'imc',
        'waist_to_hip_ratio': 'relación_cintura_cadera',
        'systolic_bp': 'presión_sistólica',
        'diastolic_bp': 'presión_diastólica',
        'heart_rate': 'frecuencia_cardiaca',
        'cholesterol_total': 'colesterol_total',
        'hdl_cholesterol': 'colesterol_hdl',
        'ldl_cholesterol': 'colesterol_ldl',
        'triglycerides': 'triglicéridos',
        'glucose_fasting': 'glucosa_ayuno',
        'glucose_postprandial': 'glucosa_postprandial',
        'insulin_level': 'nivel_insulina',
        'hba1c': 'hba1c',
        'diabetes_risk_score': 'puntaje_riesgo_diabetes',
        'diabetes_stage': 'etapa_diabetes',
        'diagnosed_diabetes': 'diabetes_diagnosticada'
    }

    df = df.rename(columns=traduccion_columnas)
```

✓ 0.0s Python

se comprueba que estan traducidas

```
df.columns
```

[20] ✓ 0.0s Python

```
Index(['edad', 'género', 'etnia', 'nivel_educativo', 'nivel_ingresos',
       'estado_laboral', 'hábito_fumar', 'consumo_alcohol_semana',
       'actividad_física_minutos_semana', 'puntaje_dieta',
       'horas_sueño_por_día', 'horas_pantalla_día',
       'antecedente_familiar_diabetes', 'antecedente_hipertensión',
       'antecedente_cardiovascular', 'imc', 'relación_cintura_cadera',
       'presión_sistólica', 'presión_diastólica', 'frecuencia_cardiaca',
       'colesterol_total', 'colesterol_hdl', 'colesterol_ldl', 'triglicéridos',
       'glucosa_ayuno', 'glucosa_postprandial', 'nivel_insulina', 'hba1c',
       'puntaje_riesgo_diabetes', 'etapa_diabetes', 'diabetes_diagnosticada'],
      dtype='object')
```

para terminar cambiaré el tipo de valores de las columnas que no corresponden primero haré los diccionarios que son para enteros y con decimales

```
colum_int= ['edad','consumo_alcohol_semana','actividad_física_minutos_semana','antecedente_familiar_diabetes', 'antecedente_hipertensión',
       'antecedente_cardiovascular','presión_sistólica', 'presión_diastólica', 'frecuencia_cardiaca','colesterol_total', 'colesterol_hdl', 'colesterol_ldl', 'triglicéridos',
       'glucosa_ayuno', 'glucosa_postprandial','diabetes_diagnosticada']

colum_float = ['puntaje_dieta','horas_sueño_por_día', 'horas_pantalla_día','imc', 'relación_cintura_cadera','nivel_insulina', 'hba1c',
       'puntaje_riesgo_diabetes']
```

[21] ✓ 0.0s Python

se hace el cambio de tipo de variable

markdown

```
for i in colum_int:
    df[i]=df[i].astype(int)

for j in colum_float:
    df[j]=df[j].astype(float)
```

[22] ✓ 0.1s Python

y se comprueba que se cambio el tipo de variable

```
df.info()
```

[23] ✓ 0.0s Python

```
<class 'pandas.core.frame.DataFrame'>
Index: 100401 entries, 0 to 100427
Data columns (total 31 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   edad                            100401 non-null  int32
 1   género                          100401 non-null  object
 2   etnia                           100401 non-null  object
 3   nivel_educativo                 100401 non-null  object
 4   nivel_ingresos                  100401 non-null  object
 5   estado_laboral                  100401 non-null  object
 6   hábito_fumar                    100401 non-null  object
 7   consumo_alcohol_semana          100401 non-null  int32
 8   actividad_física_minutos_semana  100401 non-null  int32
 9   puntaje_dieta                   100401 non-null  float64
 10  horas_sueño_por_día             100401 non-null  float64
 11  horas_pantalla_día              100401 non-null  float64
 12  antecedente_familiar_diabetes   100401 non-null  int32
 13  antecedente_hipertensión        100401 non-null  int32
 14  antecedente_cardiovascular      100401 non-null  int32
 15  imc                             100401 non-null  float64
 16  relación_cintura_cadera         100401 non-null  float64
 17  presión_sistólica               100401 non-null  int32
 18  presión_diastólica              100401 non-null  int32
 19  frecuencia_cardiaca             100401 non-null  int32
...
 29  etapa_diabetes                  100401 non-null  object
 30  diabetes_diagnosticada          100401 non-null  int32
dtypes: float64(8), int32(16), object(7)
memory usage: 18.4+ MB
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
df.shape
```

[24] ✓ 0.0s Python

```
(100401, 31)
```

se guarda la base de datos limpia

```
df.to_csv("Base_limpia.csv", index=False)
```

[25] ✓ 1.3s Python

Conclusiones:

Como problema principal, el 30% de los datos del dataframe contenían un valor nulo, por lo que eliminar dichos datos directamente por medio del comando “dropna” eliminaría gran parte de los datos, lo que produciría una mayor desviación al momento de intentar hacer predicciones, también varias columnas contenían palabras extrañas o incongurentes que impedían hacer un cambio de tipo de variable directamente al contener texto en columnas que debian ser numéricas, asimismo el dataframe presentaba valores duplicados.

Para solucionar dichos problemas recurrí primero a eliminar los valores dupplicados, que al ser muy pocos, decidí eliminarlos directamente, ya que no representaban una cantidad que afectara los resultados finales, con los valores incongruentes, primero reemplacé esos valores extraños a nulos y los valores nulos los rellené con el valor inmediato anterior de dicha columna, de esta manera me aseguraba de no crear mas incongruencias en las columnas, y mantuve el tamaño original del dataframe.