

validacion-de-la-informacionipynb

November 23, 2023

#Herramientas para la visualización de datos masivos

Objetivo: El objetivo de este cuaderno es cargar y realizar la exploración inicial de los datos utilizando el lenguaje de programación Python.

Exploración de Datos

Indice

Cargar los Datos

Visualizar los Datos

Tipos de datos

Visualizar las estadísticas

Identificar datos faltantes

Explorar relaciones entre los datos

Graficar las estadísticas

Exportar los datos

##Cargar los datos

Existen varios formatos para un conjunto de datos, .csv, .json, .xlsx, etc. Los datos pueden ser almacenados en distintos lugares, ya sea localmente o en línea. En estas sección aprenderá a cargar un conjunto de datos en su cuaderno de python. En nuestro caso el conjunto de datos Automobile es de una fuente en línea en formato CSV (valores separados por coma). Usemos este conjunto como ejemplo para practicar la lectura de datos.

fuentes de datos: <https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.data>

tipo de datos: csv

Vamos a utilizar la librería Pandas de Python para realizar la lectura de archivos. Le ponemos un alias pd para que sea más fácil utilizarla:

```
[ ]: # Importar libreria requerida
import pandas as pd
import numpy as np
import os
```

Después del comando para importar, ahora tenemos acceso a una gran cantidad de clases y funciones predefinidas. Una forma en que pandas le permite trabajar con datos es con dataframes. Repasemos el proceso para pasar de un archivo de valores separados por comas (.csv) a un dataframe. Esta variable csv_path almacena la ruta de .csv, que se utiliza como argumento para la función read_csv. El resultado se almacena en el objeto df, esta es una forma corta común que se usa para una variable que se refiere a un dataframe de Pandas.

```
[ ]: # ruta de archivos
files = os.listdir("c:\\archivos\\proyecto")
os.chdir(r'C:\archivos\proyecto')

csv_path = 'ind_urgencias_final_2023_filtrado.txt'

# Read data from CSV file
df = pd.read_csv(csv_path,sep=";",header= None)
```

```
C:\Users\Victor.Gomez\AppData\Local\Temp\ipykernel_14576\2968785239.py:8:
DtypeWarning: Columns (10,12,17,18,19,21,24) have mixed types. Specify dtype
option on import or set low_memory=False.
df = pd.read_csv(csv_path,sep=";",header= None)
```

Visualizar los datos

Podemos utilizar el método dataframe.head() para examinar las primeras cinco filas del dataframe, se utiliza cuando el conjuntos de datos es muy grande y no queremos cargar todo:

```
[ ]: # Imprimir las primeras cinco filas de un dataframe
df.head()
```

```
[ ]:
```

	0	1	2	\
0	FECHA_LLEGADA	FECHA_TRIAGE	FECHA_INGRESO	
1	2023-01-01 01:20:23.853	2023-01-01 01:28:01.847	2023-01-01 01:29:41.210	
2	2023-01-01 01:29:46.050	2023-01-01 01:48:03.070	2023-01-01 01:49:40.973	
3	2023-01-01 03:15:35.623	2023-01-01 03:23:01.990	2023-01-01 03:23:39.793	
4	2023-01-01 05:54:53.563	2023-01-01 06:00:07.943	2023-01-01 06:02:07.320	

	3	4	5	\
0	FECHA_ATENCION	TIEMPO_DGTURNO_A_TRIAGE	TIEMPO_TRIAGE_A_INGRESO	
1	2023-01-01 02:00:07.590	0:07:38	0:01:40	
2	2023-01-01 02:02:53.663	0:18:17	0:01:37	
3	2023-01-01 03:30:21.233	0:07:26	0:00:38	
4	2023-01-01 06:26:17.050	0:05:14	0:02:00	

	6	7	8	\
0	TIEMPO_INGRESO_A_CONSULTA	TIEMPO_TOTAL	Tiempo_Minutos_Total	
1	0:30:26	0:39:44	39,73	
2	0:13:13	0:33:07	33,12	
3	0:06:42	0:14:46	14,77	
4	0:24:10	0:31:24	31,40	

	9	...	15	16	17	18	19	\
0	CENTRO_ATENCION	...	RÉGIMEN PACIENTE	NOMBRE_ENTIDAD	MEDICO	AÑO	MES	
1	TN	...	SUBSIDIADO	EPSS34	11065	2023	1	
2	ME	...	SUBSIDIADO	EPSS34	8861	2023	1	
3	UC	...	CONTRIBUTIVO	EPS002	5855	2023	1	
4	UC	...	SUBSIDIADO	EPSS34	11072	2023	1	

	20	21	22	23	24
0	DIA_SEMANA	HOOR	Turnos	TIME	DIA
1	DOMINGO	1	NOCHE	2023-01-01 01:20:23.853	1
2	DOMINGO	1	NOCHE	2023-01-01 01:29:46.050	1
3	DOMINGO	3	NOCHE	2023-01-01 03:15:35.623	1
4	DOMINGO	5	NOCHE	2023-01-01 05:54:53.563	1

[5 rows x 25 columns]

Después de leer el conjunto de datos podemos utilizar el método `dataframe.head(n)` para revisar las primeras `n` filas del dataframe; donde `n` es un entero. Al contrario de `dataframe.head(n)`, `dataframe.tail(n)` mostrará las `n` filas del final del dataframe.

AHORA TÚ:

Revise las ultimas 10 filas del dataframe “df”:

```
[ ]: # Escribe tu código aquí y presiona Shift+Enter para ejecutar
print("The last 10 rows of the dataframe\n")
df.tail(10)
```

The last 10 rows of the dataframe

```
[ ]:
82407 2023-09-17 18:12:40.660 2023-09-17 18:43:16.430
82408 2023-09-17 18:52:28.640 2023-09-17 19:48:17.070
82409 2023-09-17 19:57:56.033 2023-09-17 20:15:56.793
82410 2023-09-17 20:22:04.970 2023-09-17 20:36:16.703
82411 2023-09-17 21:22:59.137 2023-09-17 21:42:20.273
82412 2023-09-18 04:44:41.970 2023-09-18 04:53:22.553
82413 2023-09-18 06:17:00.573 2023-09-18 06:28:43.040
82414 2023-09-18 06:21:37.273 2023-09-18 07:00:57.420
82415 2023-09-18 06:25:33.483 2023-09-18 06:42:02.883
82416 2023-09-18 07:14:58.180 2023-09-18 07:30:50.643

82407 2023-09-17 18:48:36.403 2023-09-17 19:40:11.800 0:30:36 0:05:20
82408 2023-09-17 19:49:47.657 2023-09-17 20:32:54.070 0:55:49 0:01:30
82409 2023-09-17 20:28:57.590 2023-09-17 20:46:29.250 0:18:00 0:13:01
```

82410	2023-09-17	20:41:12.783	2023-09-17	22:00:50.287	0:14:12	0:04:56
82411	2023-09-17	21:51:04.433	2023-09-17	22:43:13.710	0:19:21	0:08:44
82412	2023-09-18	05:05:51.423	2023-09-18	06:09:35.867	0:08:41	0:12:29
82413	2023-09-18	06:35:38.213	2023-09-18	07:40:45.957	0:11:43	0:06:55
82414	2023-09-18	07:16:45.907	2023-09-18	08:27:27.337	0:39:20	0:15:48
82415	2023-09-18	06:51:35.970	2023-09-18	07:28:28.290	0:16:29	0:09:33
82416	2023-09-18	07:34:08.370	2023-09-18	07:49:18.440	0:15:52	0:03:18

	6	7	8	9	...	15	16	17	18	\
82407	0:51:35	1:27:31	87,52	ME	...	CONTRIBUTIVO	EPS005	120	2023	
82408	0:43:07	1:40:26	100,43	ME	...	SUBSIDIADO	EPSS17	120	2023	
82409	0:17:32	0:48:33	48,55	UC	...	SUBSIDIADO	EPSC34	3818	2023	
82410	1:19:38	1:38:46	98,77	ME	...	SUBSIDIADO	EPSS02	10786	2023	
82411	0:52:09	1:20:14	80,23	ME	...	VINCULADO	11000	10786	2023	
82412	1:03:44	1:24:54	84,90	ME	...	CONTRIBUTIVO	EPS017	7844	2023	
82413	1:05:07	1:23:45	83,75	ME	...	SUBSIDIADO	EPSC34	6204	2023	
82414	1:10:42	2:05:50	125,83	ME	...	CONTRIBUTIVO	EPS041	9951	2023	
82415	0:36:53	1:02:55	62,92	UB	...	SUBSIDIADO	EPSC34	4030	2023	
82416	0:15:10	0:34:20	34,33	TN	...	SUBSIDIADO	EPSC34	1239	2023	

	19	20	21	22		23	24
82407	9	DOMINGO	18	TARDE	2023-09-17	18:12:40.660	17
82408	9	DOMINGO	18	TARDE	2023-09-17	18:52:28.640	17
82409	9	DOMINGO	19	TARDE	2023-09-17	19:57:56.033	17
82410	9	DOMINGO	20	NOCHE	2023-09-17	20:22:04.970	17
82411	9	DOMINGO	21	NOCHE	2023-09-17	21:22:59.137	17
82412	9	LUNES	4	NOCHE	2023-09-18	04:44:41.970	18
82413	9	LUNES	6	NOCHE	2023-09-18	06:17:00.573	18
82414	9	LUNES	6	NOCHE	2023-09-18	06:21:37.273	18
82415	9	LUNES	6	NOCHE	2023-09-18	06:25:33.483	18
82416	9	LUNES	7	MAÑANA	2023-09-18	07:14:58.180	18

[10 rows x 25 columns]

###Añadir cabeceras

Observe el conjunto de datos; Pandas automaticamente establece la cabecera en un entero a partir de 0.

Para describir mejor nuestros datos podemos agregarle una cabecera, esta información esta disponible en: <https://archive.ics.uci.edu/ml/datasets/Automobile>

De este modo debemos agregar las cabeceras manualmente.

Primero creamos una lista headers que incluya todos los nombres de columna en orden. Despues usamos dataframe.columns = headers para reemplazar las cabeceras por la lista que hemos creado.

```
[ ]: # crear la lista headers
headers = [
    "FECHA_LLEGADA", "FECHA_TRIAGE", "FECHA_INGRESO", "FECHA_ATENCION", "TIEMPO_DGTURNO_A_TRIAGE",
```

```

        ↪ "CENTRO_ATENCION", "CLASIFICACION_TRIAGE", "PACIENTE_#_DOCUMENTO", "EDAD", "EDAD_RANGO", "SEXO",
        ↪ "PACIENTE", "NOMBRE_ENTIDAD", "MEDICO", "AÑO", "MES", "DIA_SEMANA", "HOUR", "Turnos", "TIME", "DIA"]
print("headers\n", headers)

```

```

headers
['FECHA_LLEGADA', 'FECHA_TRIAGE', 'FECHA_INGRESO', 'FECHA_ATENCION',
'TIEMPO_DGTURNO_A_TRIAGE', 'TIEMPO_TRIAGE_A_INGRESO',
'TIEMPO_INGRESO_A_CONSULTA', 'TIEMPO_TOTAL', 'Tiempo_Minutos_Total',
'CENTRO_ATENCION', 'CLASIFICACION_TRIAGE', 'PACIENTE_#_DOCUMENTO', 'EDAD',
'EDAD_RANGO', 'SEXO', 'RÉGIMEN PACIENTE', 'NOMBRE_ENTIDAD', 'MEDICO', 'AÑO',
'MES', 'DIA_SEMANA', 'HOUR', 'Turnos', 'TIME', 'DIA']

```

Remplazamos las cabeceras y volvemos a revisar nuestro dataframe:

```

[ ]: df.columns = headers
df.head()

```

Acceder a una columna y ver sus valores

Se accede a una columna especificando el nombre de la misma. Por ejemplo, puedes acceder a la columna symboling y a la columna body-style. Cada una de estas columnas es una serie de Pandas.

```

[ ]: x=df[["PACIENTE_#_DOCUMENTO"]]
x

```

```

[ ]:
    PACIENTE_#_DOCUMENTO
0    PACIENTE_#_DOCUMENTO
1              1007228378
2              1000003681
3              1007454009
4              1022997183
...
82412              1010242518
82413              93357619
82414              41372387
82415              1000807249
82416              4497020

```

[82417 rows x 1 columns]

```

[ ]: y=df[["Tiempo_Minutos_Total"]]
y

```

```

[ ]:
    Tiempo_Minutos_Total
0    Tiempo_Minutos_Total
1              39,73
2              33,12
3              14,77

```

```

4          31,40
...
82412      84,90
82413      83,75
82414     125,83
82415      62,92
82416      34,33

```

```
[82417 rows x 1 columns]
```

##Tipos de datos

Los datos se encuentran en una variedad de tipos. Los tipos principales almacenados en dataframes de Pandas son object, float, int, bool y datetime64. Para aprender mejor acerca de cada atributo es mejor para nosotros saber el tipo de dato de cada columna.

```
[ ]: #La función dtypes genera una tabla con el tipo de dato de cada columna
df.dtypes
```

```
[ ]: FECHA_LLEGADA      object
FECHA_TRIAGE          object
FECHA_INGRESO         object
FECHA_ATENCION        object
TIEMPO_DGTURNO_A_TRIAGE  object
TIEMPO_TRIAGE_A_INGRESO  object
TIEMPO_INGRESO_A_CONSULTA object
TIEMPO_TOTAL          object
Tiempo_Minutos_Total    object
CENTRO_ATENCION        object
CLASIFICACION_TRIAGE   object
PACIENTE_#_DOCUMENTO   object
EDAD                  object
EDAD_RANGO            object
SEXO                  object
RÉGIMEN PACIENTE      object
NOMBRE_ENTIDAD        object
MEDICO                object
AÑO                   object
MES                   object
DIA_SEMANA            object
HOUR                  object
Turnos                object
TIME                  object
DIA                   object
dtype: object
```

Tipo de dato de una columna específica

De esta forma podemos consultar cuál es el tipo de dato de una columna específica:

```
[ ]: #Separamos la columna en una dataframe llamado df_column
df_column=df[['Turnos']]
df_column.dtypes
```

```
[ ]: Turnos    object
dtype: object
```

Cambiar el tipo de dato de una columna específica

¿Cómo cambiar el tipo de dato de una columna específica? Cambiemos el tipo de datos de la columna Price que fue identificado como object y es un float.

```
[ ]: #utilizamos errors='coerce' para ignorar los datos faltantes
# df["price"] = pd.to_numeric(df["price"],errors='coerce')

df['Turnos'] = df['Turnos'].astype('category')
df['DIA_SEMANA'] = df['DIA_SEMANA'].astype('category')
df['CENTRO_ATENCION'] = df['CENTRO_ATENCION'].astype('category')
df.dtypes
```

```
[ ]: FECHA_LLEGADA          object
FECHA_TRIAGE              object
FECHA_INGRESO             object
FECHA_ATENCION            object
TIEMPO_DGTURNO_A_TRIAGE   object
TIEMPO_TRIAGE_A_INGRESO   object
TIEMPO_INGRESO_A_CONSULTA object
TIEMPO_TOTAL              object
Tiempo_Minutos_Total      object
CENTRO_ATENCION           category
CLASIFICACION_TRIAGE      object
PACIENTE_#_DOCUMENTO      object
EDAD                     object
EDAD_RANGO                object
SEXO                     object
RÉGIMEN PACIENTE          object
NOMBRE_ENTIDAD            object
MEDICO                   object
AÑO                      object
MES                      object
DIA_SEMANA                category
HOUR                     object
Turnos                   category
TIME                     object
DIA                      object
dtype: object
```

Como se muestra, se observa claramente que el tipo de dato de symboling y curb-weight es int64,

normalized-losses es object pero debería ser de tipo numérico, al igual que bore, etc. Estos tipos de datos pueden modificarse.

AHORA TÚ:

Cambie el tipo de datos de la columna “stroke”:

```
[ ]: # Escribe tu código aquí y presiona Shift+Enter para ejecutar

df['Tiempo_Total'] = df['Tiempo_Minutos_Total'].str.replace(',', '.',
↪ regex=True)
df['Tiempo_Total'] = pd.to_numeric(df['Tiempo_Total'], errors='coerce')
```

Visualizar las estadísticas

Este conjunto de datos es pequeño, pero si se quisiera saber la cantidad de atributos y de elementos que se tienen en el conjunto de datos, se puede utilizar la función dataframe.shape. Esta función visualiza primero el número de elementos y luego el número de atributos.

```
[ ]: df.shape
```

```
[ ]: (82417, 26)
```

Vamos a utilizar la función dataframe.describe para visualizar las estadísticas del conjunto de datos. Por defecto, la función dataframe.describe muestra las filas y columnas que contienen números.

Esto mostrará:

el recuento de esa variable

la media

la desviación estándar (std)

el valor mínimo

el IQR (rango intercuartil: 25%, 50% y 75%)

el valor máximo

```
[ ]: df.describe()
```

```
[ ]:      Tiempo_Total
count  82416.000000
mean    108.346035
std     132.899154
min       8.570000
25%     60.600000
50%     84.200000
75%    117.672500
max    2684.830000
```

Si se quisiera calcular la mediana de una variable en específico se puede de la siguiente manera:


```
[ ]: import matplotlib.pyplot as plt
import seaborn as sns
sns.distplot(df["Tiempo_Total"])
```

```
[ ]: df.head()
```

```
[ ]:
      FECHA_LLEGADA      FECHA_TRIAGE      FECHA_INGRESO \
1  2023-01-01 01:20:23.853  2023-01-01 01:28:01.847  2023-01-01 01:29:41.210
2  2023-01-01 01:29:46.050  2023-01-01 01:48:03.070  2023-01-01 01:49:40.973
3  2023-01-01 03:15:35.623  2023-01-01 03:23:01.990  2023-01-01 03:23:39.793
4  2023-01-01 05:54:53.563  2023-01-01 06:00:07.943  2023-01-01 06:02:07.320
5  2023-01-01 06:37:27.237  2023-01-01 07:52:31.687  2023-01-01 07:52:37.717
```

```
      FECHA_ATENCION TIEMPO_DGTURNNO_A_TRIAGE TIEMPO_TRIAGE_A_INGRESO \
1  2023-01-01 02:00:07.590      0:07:38      0:01:40
2  2023-01-01 02:02:53.663      0:18:17      0:01:37
3  2023-01-01 03:30:21.233      0:07:26      0:00:38
4  2023-01-01 06:26:17.050      0:05:14      0:02:00
5  2023-01-01 09:31:15.597      1:15:04      0:00:06
```

```
      TIEMPO_INGRESO_A_CONSULTA TIEMPO_TOTAL Tiempo_Minutos_Total CENTRO_ATENCION \
1      0:30:26      0:39:44      39,73      TN
2      0:13:13      0:33:07      33,12      ME
3      0:06:42      0:14:46      14,77      UC
4      0:24:10      0:31:24      31,40      UC
5      1:38:38      2:53:48      173,80      TN
```

```
... NOMBRE_ENTIDAD MEDICO  AÑO MES DIA_SEMANA HOUR Turnos \
1 ...      EPSS34  11065  2023  1  DOMINGO  1  NOCHE
2 ...      EPSS34   8861  2023  1  DOMINGO  1  NOCHE
3 ...      EPS002   5855  2023  1  DOMINGO  3  NOCHE
4 ...      EPSS34  11072  2023  1  DOMINGO  5  NOCHE
5 ...      EPSS34   1239  2023  1  DOMINGO  6  NOCHE
```

```
      TIME DIA Tiempo_Total
1  2023-01-01 01:20:23.853  1      39.73
2  2023-01-01 01:29:46.050  1      33.12
3  2023-01-01 03:15:35.623  1      14.77
4  2023-01-01 05:54:53.563  1      31.40
5  2023-01-01 06:37:27.237  1     173.80
```

[5 rows x 26 columns]

```
[ ]: df = df.drop(0)
```

```
[ ]: #Muestra la mediana para los atributos "length" y "compression-ratio"
median= df[['Tiempo_Total']].median()
```

```
median
```

```
[ ]: Tiempo_Total      84.2  
dtype: float64
```

Por defecto la función solo muestra atributos que son numéricos. Es posible hacer que la función describe funcione también para las columnas de tipo object. Para permitir un resumen de todas las columnas, podríamos añadir un argumento include="all" entre los paréntesis de la función describe.

```
[ ]: #unique, top y frequency ("único, superior y frecuencia").  
#df.describe(include="object")  
df.describe(include="all")
```

```
[ ]:          FECHA_LLEGADA          FECHA_TRIAGE \  
count                82416                82416  
unique                80382                80388  
top    2023-08-06 14:31:20.657  2023-08-04 11:13:29.903  
freq                   4                   4  
mean                  NaN                  NaN  
std                   NaN                  NaN  
min                   NaN                  NaN  
25%                   NaN                  NaN  
50%                   NaN                  NaN  
75%                   NaN                  NaN  
max                   NaN                  NaN
```

```
          FECHA_INGRESO          FECHA_ATENCION \  
count                82416                82416  
unique                80387                82414  
top    2023-05-05 16:30:25.897  2023-03-29 21:49:44.440  
freq                   4                   2  
mean                  NaN                  NaN  
std                   NaN                  NaN  
min                   NaN                  NaN  
25%                   NaN                  NaN  
50%                   NaN                  NaN  
75%                   NaN                  NaN  
max                   NaN                  NaN
```

```
          TIEMPO_DGTURNO_A_TRIAGE  TIEMPO_TRIAGE_A_INGRESO \  
count                82416                82416  
unique                5981                3433  
top                   0:10:06                0:01:08  
freq                   72                218  
mean                  NaN                  NaN  
std                   NaN                  NaN  
min                   NaN                  NaN
```

25%	NaN	NaN
50%	NaN	NaN
75%	NaN	NaN
max	NaN	NaN

	TIEMPO_INGRESO_A_CONSULTA	TIEMPO_TOTAL	Tiempo_Minutos_Total \
count	82416	82416	82416
unique	12074	13866	14141
top	0:33:27	1:13:04	73,07
freq	37	28	28
mean	NaN	NaN	NaN
std	NaN	NaN	NaN
min	NaN	NaN	NaN
25%	NaN	NaN	NaN
50%	NaN	NaN	NaN
75%	NaN	NaN	NaN
max	NaN	NaN	NaN

	CENTRO_ATENCION	...	NOMBRE_ENTIDAD	MEDICO	AÑO	MES \
count	82416	...	82416	82416.0	82416.0	82416.0
unique	9	...	78	378.0	2.0	14.0
top	ME	...	EPSC34	3826.0	2023.0	7.0
freq	41221	...	45973	1464.0	49649.0	10108.0
mean	NaN	...	NaN	NaN	NaN	NaN
std	NaN	...	NaN	NaN	NaN	NaN
min	NaN	...	NaN	NaN	NaN	NaN
25%	NaN	...	NaN	NaN	NaN	NaN
50%	NaN	...	NaN	NaN	NaN	NaN
75%	NaN	...	NaN	NaN	NaN	NaN
max	NaN	...	NaN	NaN	NaN	NaN

	DIA_SEMANA	HOOR	Turnos	TIME	DIA \
count	82416	82416.0	82416	82416	82416.0
unique	7	48.0	3	80382	62.0
top	LUNES	10.0	MAÑANA	2023-08-06 14:31:20.657	8.0
freq	13390	3911.0	40363	4	2040.0
mean	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN

	Tiempo_Total
count	82416.000000
unique	NaN

```

top          NaN
freq         NaN
mean      108.346035
std       132.899154
min         8.570000
25%        60.600000
50%        84.200000
75%       117.672500
max       2684.830000

```

```
[11 rows x 26 columns]
```

Contar Valores

Una forma de resumir los datos categóricos es usando la función `value_counts`. Por ejemplo, en nuestro conjunto de datos, tenemos el lugar del motor (engine-location) como una variable categórica de frontal y trasero.

```
[ ]: drive_wheels_counts = df['Turnos'].value_counts().to_frame()
drive_wheels_counts
```

```
[ ]:
count
Turnos
MAÑANA 40363
TARDE  25052
NOCHE  17001
Turnos      0
```

Examinar los recuentos de valores de la ubicación del motor no sería una buena variable predictiva del precio. Esto se debe a que solo tenemos tres autos con motor trasero y 202 con motor delantero, este resultado es sesgado. Por lo tanto, no podemos sacar ninguna conclusión sobre la ubicación del motor.

AHORA TÚ:

Puede seleccionar las columnas de un dataframe indicando el nombre de cada una, por ejemplo, puede seleccionar tres columnas de la siguiente manera:

```
dataframe[['column 1',column 2', 'column 3']]
```

Donde “column” es el nombre de la columna se puede aplicar el método “`.describe()`” para obtener las estadísticas de aquellas columnas de la siguiente manera:

```
dataframe[['column 1',column 2', 'column 3']].describe()
```

Aplicar el método “`.describe()`” a las columnas ‘length’ y ‘compression-ratio’.

```
[ ]: # Escribe tu código aquí y presiona Shift+Enter para ejecutar
df[['DIA_SEMANA', 'Turnos', 'Tiempo_Total']].describe()
```

```
[ ]:      Tiempo_Total
count    82416.000000
mean      108.346035
std       132.899154
min        8.570000
25%       60.600000
50%       84.200000
75%      117.672500
max      2684.830000
```

##Identificar datos faltantes

Debemos visualizar nuestros datos e identificar el valor(es) que se está utilizando para los datos faltantes. Los valores faltantes pueden ser espacios vacíos, NA, n/a, -, 0, o cualquier otro valor que no es considerado correcto en esa columna.

```
[ ]: #Identifique cual es el valor que se está utilizando para los datos faltantes
      ↪ en el set de datos:
df.head(20)
```

```
[ ]:      FECHA_LLEGADA      FECHA_TRIAGE      FECHA_INGRESO \
1    2023-01-01 01:20:23.853  2023-01-01 01:28:01.847  2023-01-01 01:29:41.210
2    2023-01-01 01:29:46.050  2023-01-01 01:48:03.070  2023-01-01 01:49:40.973
3    2023-01-01 03:15:35.623  2023-01-01 03:23:01.990  2023-01-01 03:23:39.793
4    2023-01-01 05:54:53.563  2023-01-01 06:00:07.943  2023-01-01 06:02:07.320
5    2023-01-01 06:37:27.237  2023-01-01 07:52:31.687  2023-01-01 07:52:37.717
6    2023-01-01 07:09:46.950  2023-01-01 07:18:01.200  2023-01-01 07:20:03.720
7    2023-01-01 07:20:31.113  2023-01-01 07:27:36.230  2023-01-01 07:34:17.667
8    2023-01-01 07:53:52.963  2023-01-01 08:01:10.640  2023-01-01 08:03:13.710
9    2023-01-01 08:05:21.230  2023-01-01 08:53:07.870  2023-01-01 08:56:10.277
10   2023-01-01 09:24:15.530  2023-01-01 09:52:05.463  2023-01-01 09:59:19.673
11   2023-01-01 10:57:35.917  2023-01-01 11:26:37.567  2023-01-01 11:30:19.197
12   2023-01-01 14:34:37.470  2023-01-01 14:49:51.770  2023-01-01 15:02:04.030
13   2023-01-01 14:54:22.407  2023-01-01 15:13:27.167  2023-01-01 15:20:55.660
14   2023-01-01 14:56:15.033  2023-01-01 15:03:35.823  2023-01-01 15:04:17.987
15   2023-01-01 16:09:09.527  2023-01-01 16:14:42.000  2023-01-01 16:16:22.767
16   2023-01-01 18:07:03.087  2023-01-01 18:23:37.837  2023-01-01 18:24:11.420
17   2023-01-01 18:13:00.330  2023-01-01 18:31:06.813  2023-01-01 18:34:41.503
18   2023-01-02 03:34:19.883  2023-01-02 03:56:47.470  2023-01-02 03:57:22.443
19   2023-01-02 04:30:07.617  2023-01-02 05:36:52.657  2023-01-02 05:38:34.253
20   2023-01-02 06:27:33.233  2023-01-02 07:12:56.400  2023-01-02 07:13:07.473
```

```
      FECHA_ATENCION TIEMPO_DGTURN0_A_TRIAGE TIEMPO_TRIAGE_A_INGRESO \
1    2023-01-01 02:00:07.590      0:07:38      0:01:40
2    2023-01-01 02:02:53.663      0:18:17      0:01:37
3    2023-01-01 03:30:21.233      0:07:26      0:00:38
4    2023-01-01 06:26:17.050      0:05:14      0:02:00
5    2023-01-01 09:31:15.597      1:15:04      0:00:06
```

6	2023-01-01	07:48:42.197	0:08:15	0:02:02
7	2023-01-01	07:59:17.140	0:07:05	0:06:41
8	2023-01-01	08:43:09.917	0:07:18	0:02:03
9	2023-01-01	09:26:52.977	0:47:46	0:03:03
10	2023-01-01	10:38:12.297	0:27:50	0:07:14
11	2023-01-01	12:48:10.377	0:29:02	0:03:42
12	2023-01-01	16:33:36.400	0:15:14	0:12:13
13	2023-01-01	16:47:10.060	0:19:05	0:07:28
14	2023-01-01	16:12:09.400	0:07:20	0:00:42
15	2023-01-01	16:39:18.937	0:05:33	0:01:40
16	2023-01-01	19:33:28.253	0:16:34	0:00:34
17	2023-01-01	19:54:53.770	0:18:06	0:03:35
18	2023-01-02	04:49:21.190	0:22:28	0:00:35
19	2023-01-02	05:54:41.813	1:06:45	0:01:42
20	2023-01-02	07:21:55.423	0:45:23	0:00:11

	TIEMPO_INGRESO_A_CONSULTA	TIEMPO_TOTAL	Tiempo_Minutos_Total \
1	0:30:26	0:39:44	39,73
2	0:13:13	0:33:07	33,12
3	0:06:42	0:14:46	14,77
4	0:24:10	0:31:24	31,40
5	1:38:38	2:53:48	173,80
6	0:28:39	0:38:56	38,93
7	0:25:00	0:38:46	38,77
8	0:39:56	0:49:17	49,28
9	0:30:42	1:21:31	81,52
10	0:38:53	1:13:57	73,95
11	1:17:51	1:50:35	110,58
12	1:31:32	1:58:59	118,98
13	1:26:15	1:52:48	112,80
14	1:07:52	1:15:54	75,90
15	0:22:56	0:30:09	30,15
16	1:09:17	1:26:25	86,42
17	1:20:12	1:41:53	101,88
18	0:51:59	1:15:02	75,03
19	0:16:07	1:24:34	84,57
20	0:08:48	0:54:22	54,37

	CENTRO_ATENCION	...	NOMBRE_ENTIDAD	MEDICO	AÑO	MES	DIA_SEMANA	hour \
1	TN	...	EPSS34	11065	2023	1	DOMINGO	1
2	ME	...	EPSS34	8861	2023	1	DOMINGO	1
3	UC	...	EPS002	5855	2023	1	DOMINGO	3
4	UC	...	EPSS34	11072	2023	1	DOMINGO	5
5	TN	...	EPSS34	1239	2023	1	DOMINGO	6
6	VB	...	EPS005	10751	2023	1	DOMINGO	7
7	UC	...	EPSS34	11072	2023	1	DOMINGO	7
8	VB	...	EPS041	10795	2023	1	DOMINGO	7

9	UB	...	ESS207	7795	2023	1	DOMINGO	8
10	ME	...	EPSS34	120	2023	1	DOMINGO	9
11	ME	...	EPSS17	7551	2023	1	DOMINGO	10
12	ME	...	ESS024	6204	2023	1	DOMINGO	14
13	VB	...	EPSS34	3826	2023	1	DOMINGO	14
14	VB	...	EPSS34	10795	2023	1	DOMINGO	14
15	TN	...	EPSC34	1239	2023	1	DOMINGO	16
16	TN	...	EPSS34	10030	2023	1	DOMINGO	18
17	TN	...	EPSS05	10030	2023	1	DOMINGO	18
18	ME	...	EPS002	10938	2023	1	LUNES	3
19	JT	...	EPSS34	8373	2023	1	LUNES	4
20	ME	...	EPSC34	4610	2023	1	LUNES	6

	Turnos	TIME	DIA	Tiempo_Total
1	NOCHE	2023-01-01 01:20:23.853	1	39.73
2	NOCHE	2023-01-01 01:29:46.050	1	33.12
3	NOCHE	2023-01-01 03:15:35.623	1	14.77
4	NOCHE	2023-01-01 05:54:53.563	1	31.40
5	NOCHE	2023-01-01 06:37:27.237	1	173.80
6	MAÑANA	2023-01-01 07:09:46.950	1	38.93
7	MAÑANA	2023-01-01 07:20:31.113	1	38.77
8	MAÑANA	2023-01-01 07:53:52.963	1	49.28
9	MAÑANA	2023-01-01 08:05:21.230	1	81.52
10	MAÑANA	2023-01-01 09:24:15.530	1	73.95
11	MAÑANA	2023-01-01 10:57:35.917	1	110.58
12	TARDE	2023-01-01 14:34:37.470	1	118.98
13	TARDE	2023-01-01 14:54:22.407	1	112.80
14	TARDE	2023-01-01 14:56:15.033	1	75.90
15	TARDE	2023-01-01 16:09:09.527	1	30.15
16	TARDE	2023-01-01 18:07:03.087	1	86.42
17	TARDE	2023-01-01 18:13:00.330	1	101.88
18	NOCHE	2023-01-02 03:34:19.883	2	75.03
19	NOCHE	2023-01-02 04:30:07.617	2	84.57
20	NOCHE	2023-01-02 06:27:33.233	2	54.37

[20 rows x 26 columns]

Con la función `isnull` podemos saber cuantos datos faltantes identifica Python en nuestro set de datos.

```
[ ]: print(df.isnull().sum())
```

FECHA_LLEGADA	0
FECHA_TRIAGE	0
FECHA_INGRESO	0
FECHA_ATENCION	0
TIEMPO_DGTURNO_A_TRIAGE	0
TIEMPO_TRIAGE_A_INGRESO	0

```

TIEMPO_INGRESO_A_CONSULTA    0
TIEMPO_TOTAL                 0
Tiempo_Minutos_Total        0
CENTRO_ATENCION              0
CLASIFICACION_TRIAGE         0
PACIENTE_#_DOCUMENTO        0
EDAD                        0
EDAD_RANGO                   0
SEXO                         0
RÉGIMEN PACIENTE             0
NOMBRE_ENTIDAD               0
MEDICO                       0
AÑO                          0
MES                          0
DIA_SEMANA                   0
HOUR                         0
Turnos                       0
TIME                         0
DIA                          0
Tiempo_Total                 0
dtype: int64

```

Todavía Python no está identificando los datos faltantes en el conjunto de datos, sino que los está tratando como un valor correcto más. Para marcar los datos faltantes se realiza lo siguiente:

```

[ ]: #Realice una lista de los valores que son identificados como datos faltantes
     #No olvide al final volver a cargar las cabeceras
missing_values = ["?", "1"]
csv_path = 'ind_urgencias_final_2023_filtrado.txt'
df = pd.read_csv(csv_path, sep=";", header= None, na_values = missing_values)
df.head(20)

```

C:\Users\Victor.Gomez\AppData\Local\Temp\ipykernel_14576\2949932682.py:5:

DtypeWarning: Columns (10,12,17,18,19,21,24) have mixed types. Specify dtype option on import or set low_memory=False.

```
df = pd.read_csv(csv_path, sep=";", header= None, na_values = missing_values)
```

```

[ ]:
      0          1          2  \
0      FECHA_LLEGADA      FECHA_TRIAGE      FECHA_INGRESO
1  2023-01-01 01:20:23.853  2023-01-01 01:28:01.847  2023-01-01 01:29:41.210
2  2023-01-01 01:29:46.050  2023-01-01 01:48:03.070  2023-01-01 01:49:40.973
3  2023-01-01 03:15:35.623  2023-01-01 03:23:01.990  2023-01-01 03:23:39.793
4  2023-01-01 05:54:53.563  2023-01-01 06:00:07.943  2023-01-01 06:02:07.320
5  2023-01-01 06:37:27.237  2023-01-01 07:52:31.687  2023-01-01 07:52:37.717
6  2023-01-01 07:09:46.950  2023-01-01 07:18:01.200  2023-01-01 07:20:03.720
7  2023-01-01 07:20:31.113  2023-01-01 07:27:36.230  2023-01-01 07:34:17.667
8  2023-01-01 07:53:52.963  2023-01-01 08:01:10.640  2023-01-01 08:03:13.710
9  2023-01-01 08:05:21.230  2023-01-01 08:53:07.870  2023-01-01 08:56:10.277

```


10	2023-01-01 09:24:15.530	2023-01-01 09:52:05.463	2023-01-01 09:59:19.673
11	2023-01-01 10:57:35.917	2023-01-01 11:26:37.567	2023-01-01 11:30:19.197
12	2023-01-01 14:34:37.470	2023-01-01 14:49:51.770	2023-01-01 15:02:04.030
13	2023-01-01 14:54:22.407	2023-01-01 15:13:27.167	2023-01-01 15:20:55.660
14	2023-01-01 14:56:15.033	2023-01-01 15:03:35.823	2023-01-01 15:04:17.987
15	2023-01-01 16:09:09.527	2023-01-01 16:14:42.000	2023-01-01 16:16:22.767
16	2023-01-01 18:07:03.087	2023-01-01 18:23:37.837	2023-01-01 18:24:11.420
17	2023-01-01 18:13:00.330	2023-01-01 18:31:06.813	2023-01-01 18:34:41.503
18	2023-01-02 03:34:19.883	2023-01-02 03:56:47.470	2023-01-02 03:57:22.443
19	2023-01-02 04:30:07.617	2023-01-02 05:36:52.657	2023-01-02 05:38:34.253

	3	4	5 \
0	FECHA_ATENCION	TIEMPO_DGTURNO_A_TRIAGE	TIEMPO_TRIAGE_A_INGRESO
1	2023-01-01 02:00:07.590	0:07:38	0:01:40
2	2023-01-01 02:02:53.663	0:18:17	0:01:37
3	2023-01-01 03:30:21.233	0:07:26	0:00:38
4	2023-01-01 06:26:17.050	0:05:14	0:02:00
5	2023-01-01 09:31:15.597	1:15:04	0:00:06
6	2023-01-01 07:48:42.197	0:08:15	0:02:02
7	2023-01-01 07:59:17.140	0:07:05	0:06:41
8	2023-01-01 08:43:09.917	0:07:18	0:02:03
9	2023-01-01 09:26:52.977	0:47:46	0:03:03
10	2023-01-01 10:38:12.297	0:27:50	0:07:14
11	2023-01-01 12:48:10.377	0:29:02	0:03:42
12	2023-01-01 16:33:36.400	0:15:14	0:12:13
13	2023-01-01 16:47:10.060	0:19:05	0:07:28
14	2023-01-01 16:12:09.400	0:07:20	0:00:42
15	2023-01-01 16:39:18.937	0:05:33	0:01:40
16	2023-01-01 19:33:28.253	0:16:34	0:00:34
17	2023-01-01 19:54:53.770	0:18:06	0:03:35
18	2023-01-02 04:49:21.190	0:22:28	0:00:35
19	2023-01-02 05:54:41.813	1:06:45	0:01:42

	6	7	8 \
0	TIEMPO_INGRESO_A_CONSULTA	TIEMPO_TOTAL	Tiempo_Minutos_Total
1	0:30:26	0:39:44	39,73
2	0:13:13	0:33:07	33,12
3	0:06:42	0:14:46	14,77
4	0:24:10	0:31:24	31,40
5	1:38:38	2:53:48	173,80
6	0:28:39	0:38:56	38,93
7	0:25:00	0:38:46	38,77
8	0:39:56	0:49:17	49,28
9	0:30:42	1:21:31	81,52
10	0:38:53	1:13:57	73,95
11	1:17:51	1:50:35	110,58
12	1:31:32	1:58:59	118,98

13	1:26:15	1:52:48	112,80
14	1:07:52	1:15:54	75,90
15	0:22:56	0:30:09	30,15
16	1:09:17	1:26:25	86,42
17	1:20:12	1:41:53	101,88
18	0:51:59	1:15:02	75,03
19	0:16:07	1:24:34	84,57

	9	...	15	16	17	18 \
0	CENTRO_ATENCION	...	RÉGIMEN PACIENTE	NOMBRE_ENTIDAD	MEDICO	AÑO
1	TN	...	SUBSIDIADO	EPSS34	11065	2023
2	ME	...	SUBSIDIADO	EPSS34	8861	2023
3	UC	...	CONTRIBUTIVO	EPS002	5855	2023
4	UC	...	SUBSIDIADO	EPSS34	11072	2023
5	TN	...	SUBSIDIADO	EPSS34	1239	2023
6	VB	...	CONTRIBUTIVO	EPS005	10751	2023
7	UC	...	SUBSIDIADO	EPSS34	11072	2023
8	VB	...	CONTRIBUTIVO	EPS041	10795	2023
9	UB	...	SUBSIDIADO	ESS207	7795	2023
10	ME	...	SUBSIDIADO	EPSS34	120	2023
11	ME	...	CONTRIBUTIVO	EPSS17	7551	2023
12	ME	...	SUBSIDIADO	ESS024	6204	2023
13	VB	...	SUBSIDIADO	EPSS34	3826	2023
14	VB	...	SUBSIDIADO	EPSS34	10795	2023
15	TN	...	CONTRIBUTIVO	EPSC34	1239	2023
16	TN	...	SUBSIDIADO	EPSS34	10030	2023
17	TN	...	SUBSIDIADO	EPSS05	10030	2023
18	ME	...	CONTRIBUTIVO	EPS002	10938	2023
19	JT	...	SUBSIDIADO	EPSS34	8373	2023

	19	20	21	22	23	24
0	MES	DIA_SEMANA	HOURL	Turnos	TIME	DIA
1	1	DOMINGO	1	NOCHE	2023-01-01 01:20:23.853	1
2	1	DOMINGO	1	NOCHE	2023-01-01 01:29:46.050	1
3	1	DOMINGO	3	NOCHE	2023-01-01 03:15:35.623	1
4	1	DOMINGO	5	NOCHE	2023-01-01 05:54:53.563	1
5	1	DOMINGO	6	NOCHE	2023-01-01 06:37:27.237	1
6	1	DOMINGO	7	MAÑANA	2023-01-01 07:09:46.950	1
7	1	DOMINGO	7	MAÑANA	2023-01-01 07:20:31.113	1
8	1	DOMINGO	7	MAÑANA	2023-01-01 07:53:52.963	1
9	1	DOMINGO	8	MAÑANA	2023-01-01 08:05:21.230	1
10	1	DOMINGO	9	MAÑANA	2023-01-01 09:24:15.530	1
11	1	DOMINGO	10	MAÑANA	2023-01-01 10:57:35.917	1
12	1	DOMINGO	14	TARDE	2023-01-01 14:34:37.470	1
13	1	DOMINGO	14	TARDE	2023-01-01 14:54:22.407	1
14	1	DOMINGO	14	TARDE	2023-01-01 14:56:15.033	1
15	1	DOMINGO	16	TARDE	2023-01-01 16:09:09.527	1

16	1	DOMINGO	18	TARDE	2023-01-01	18:07:03.087	1
17	1	DOMINGO	18	TARDE	2023-01-01	18:13:00.330	1
18	1	LUNES	3	NOCHE	2023-01-02	03:34:19.883	2
19	1	LUNES	4	NOCHE	2023-01-02	04:30:07.617	2

[20 rows x 25 columns]

```
[ ]: # crear la lista headers

df = df.drop(0)
headers = [
    "FECHA_LLEGADA", "FECHA_TRIAGE", "FECHA_INGRESO", "FECHA_ATENCION", "TIEMPO_DGTURNO_A_TRIAGE",
    "CENTRO_ATENCION", "CLASIFICACION_TRIAGE", "PACIENTE_#_DOCUMENTO", "EDAD", "EDAD_RANGO", "SEXO",
    "PACIENTE", "NOMBRE_ENTIDAD", "MEDICO", "AÑO", "MES", "DIA_SEMANA", "HOUR", "Turnos", "TIME", "DIA"]
df.columns = headers
df.head()
```

```
[ ]:          FECHA_LLEGADA          FECHA_TRIAGE          FECHA_INGRESO \
1  2023-01-01 01:20:23.853  2023-01-01 01:28:01.847  2023-01-01 01:29:41.210
2  2023-01-01 01:29:46.050  2023-01-01 01:48:03.070  2023-01-01 01:49:40.973
3  2023-01-01 03:15:35.623  2023-01-01 03:23:01.990  2023-01-01 03:23:39.793
4  2023-01-01 05:54:53.563  2023-01-01 06:00:07.943  2023-01-01 06:02:07.320
5  2023-01-01 06:37:27.237  2023-01-01 07:52:31.687  2023-01-01 07:52:37.717
```

```
          FECHA_ATENCION  TIEMPO_DGTURNO_A_TRIAGE  TIEMPO_TRIAGE_A_INGRESO \
1  2023-01-01 02:00:07.590                0:07:38                0:01:40
2  2023-01-01 02:02:53.663                0:18:17                0:01:37
3  2023-01-01 03:30:21.233                0:07:26                0:00:38
4  2023-01-01 06:26:17.050                0:05:14                0:02:00
5  2023-01-01 09:31:15.597                1:15:04                0:00:06
```

```
TIEMPO_INGRESO_A_CONSULTA  TIEMPO_TOTAL  Tiempo_Minutos_Total  CENTRO_ATENCION \
1                0:30:26            0:39:44                39,73            TN
2                0:13:13            0:33:07                33,12            ME
3                0:06:42            0:14:46                14,77            UC
4                0:24:10            0:31:24                31,40            UC
5                1:38:38            2:53:48               173,80            TN
```

```
... RÉGIMEN PACIENTE  NOMBRE_ENTIDAD  MEDICO  AÑO  MES  DIA_SEMANA  HOUR \
1  ...      SUBSIDIADO      EPSS34  11065  2023   1   DOMINGO    1
2  ...      SUBSIDIADO      EPSS34   8861  2023   1   DOMINGO    1
3  ...    CONTRIBUTIVO      EPS002   5855  2023   1   DOMINGO    3
4  ...      SUBSIDIADO      EPSS34  11072  2023   1   DOMINGO    5
5  ...      SUBSIDIADO      EPSS34   1239  2023   1   DOMINGO    6
```

Turnos TIME DIA

```

1 NOCHE 2023-01-01 01:20:23.853 1
2 NOCHE 2023-01-01 01:29:46.050 1
3 NOCHE 2023-01-01 03:15:35.623 1
4 NOCHE 2023-01-01 05:54:53.563 1
5 NOCHE 2023-01-01 06:37:27.237 1

```

[5 rows x 25 columns]

Vuelva a ejecutar la sentencia `print(df.isnull().sum())` para visualizar los datos faltantes

```
[ ]: print(df.isnull().sum())
```

```

FECHA_LLEGADA          0
FECHA_TRIAGE           0
FECHA_INGRESO          0
FECHA_ATENCION         0
TIEMPO_DGTURNO_A_TRIAGE 0
TIEMPO_TRIAGE_A_INGRESO 0
TIEMPO_INGRESO_A_CONSULTA 0
TIEMPO_TOTAL           0
Tiempo_Minutos_Total   0
CENTRO_ATENCION        0
CLASIFICACION_TRIAGE   0
PACIENTE_#_DOCUMENTO   0
EDAD                  0
EDAD_RANGO            0
SEXO                  0
RÉGIMEN PACIENTE       0
NOMBRE_ENTIDAD         0
MEDICO                0
AÑO                   0
MES                   0
DIA_SEMANA            0
HOUR                  0
Turnos                0
TIME                  0
DIA                   0
dtype: int64

```

```
[ ]: df.dtypes
```

```

[ ]: FECHA_LLEGADA          object
      FECHA_TRIAGE          object
      FECHA_INGRESO         object
      FECHA_ATENCION        object
      TIEMPO_DGTURNO_A_TRIAGE object
      TIEMPO_TRIAGE_A_INGRESO object

```

```

TIEMPO_INGRESO_A_CONSULTA    object
TIEMPO_TOTAL                  object
Tiempo_Minutos_Total         object
CENTRO_ATENCION               category
CLASIFICACION_TRIAGE         object
PACIENTE_#_DOCUMENTO         object
EDAD                          object
EDAD_RANGO                    object
SEXO                          object
RÉGIMEN PACIENTE              object
NOMBRE_ENTIDAD                object
MEDICO                        object
AÑO                           object
MES                           int64
DIA_SEMANA                    category
HOUR                           int64
Turnos                        category
TIME                          object
DIA                           int64
Tiempo_Total                  float64
dtype: object

```

##Explorar relaciones entre los datos

Variables numéricas continuas:

Las variables numéricas continuas son variables que pueden contener cualquier valor dentro de cierto rango. Las variables numéricas continuas pueden tener el tipo int64 o float64. Una excelente manera de visualizar estas variables es mediante el uso de diagramas de dispersión con líneas ajustadas.

Para comenzar a comprender la relación (lineal) entre una variable individual y el precio. Podemos hacer esto usando regplot, que traza el diagrama de dispersión más la línea de regresión ajustada para los datos.

Vamos a importar las librerías Matplotlib y Seaborn para la visualización de datos. Les ponemos un alias plt y sns para que sea más fácil su uso:

```
[ ]: import matplotlib.pyplot as plt
import seaborn as sns
```

El gráfico seaborn.pairplot permite visualizar todas las variables numéricas y la combinación entre ellas. Pudiendo identificar facilmente si existen relaciones de dependencia entre algunas variables.

```
[ ]: #sns.pairplot(df)
df['Tiempo_Total'] = df['Tiempo_Minutos_Total'].str.replace(',', '.',
    ↪regex=True)
df['Tiempo_Total'] = pd.to_numeric(df['Tiempo_Total'], errors='coerce')
df['Turnos'] = df['Turnos'].astype('category')
df['DIA_SEMANA'] = df['DIA_SEMANA'].astype('category')
```

```

df['CENTRO_ATENCION'] = df['CENTRO_ATENCION'].astype('category')
df['MES'] = pd.to_numeric(df['MES'], errors='coerce')
df['HOUR'] = pd.to_numeric(df['HOUR'], errors='coerce')
df['DIA'] = pd.to_numeric(df['DIA'], errors='coerce')

sub_df = df[['Tiempo_Total', 'MES', 'DIA', 'HOUR']]

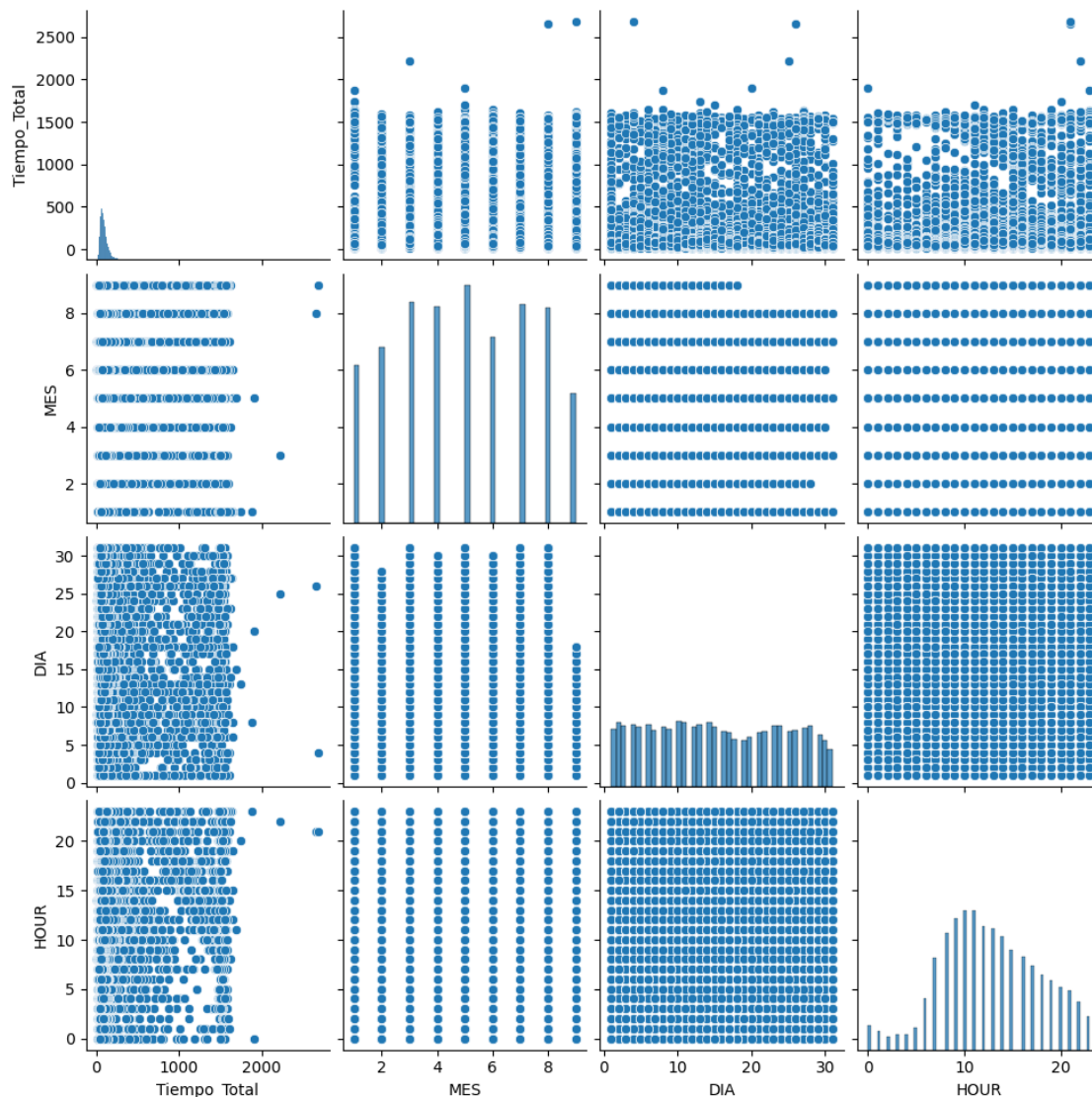
sns.pairplot(sub_df)

```

c:\Users\Victor.Gomez\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight

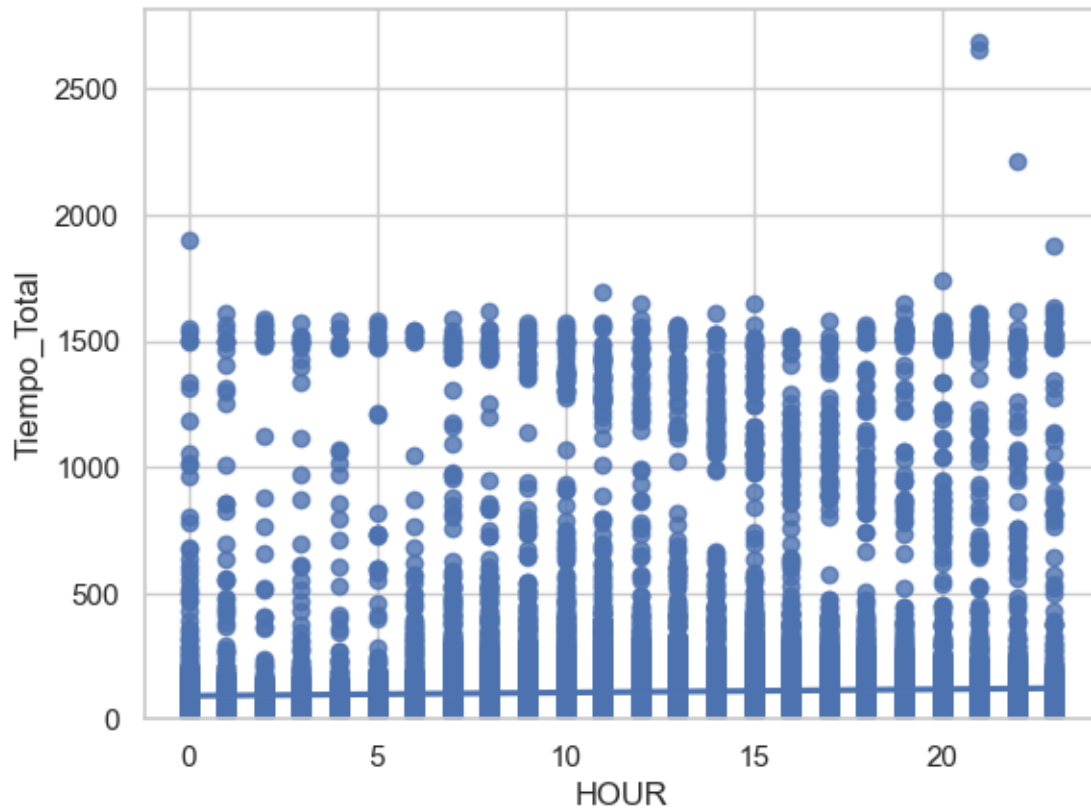
```
self._figure.tight_layout(*args, **kwargs)
```

```
[ ]: <seaborn.axisgrid.PairGrid at 0x1b6b1f7b010>
```

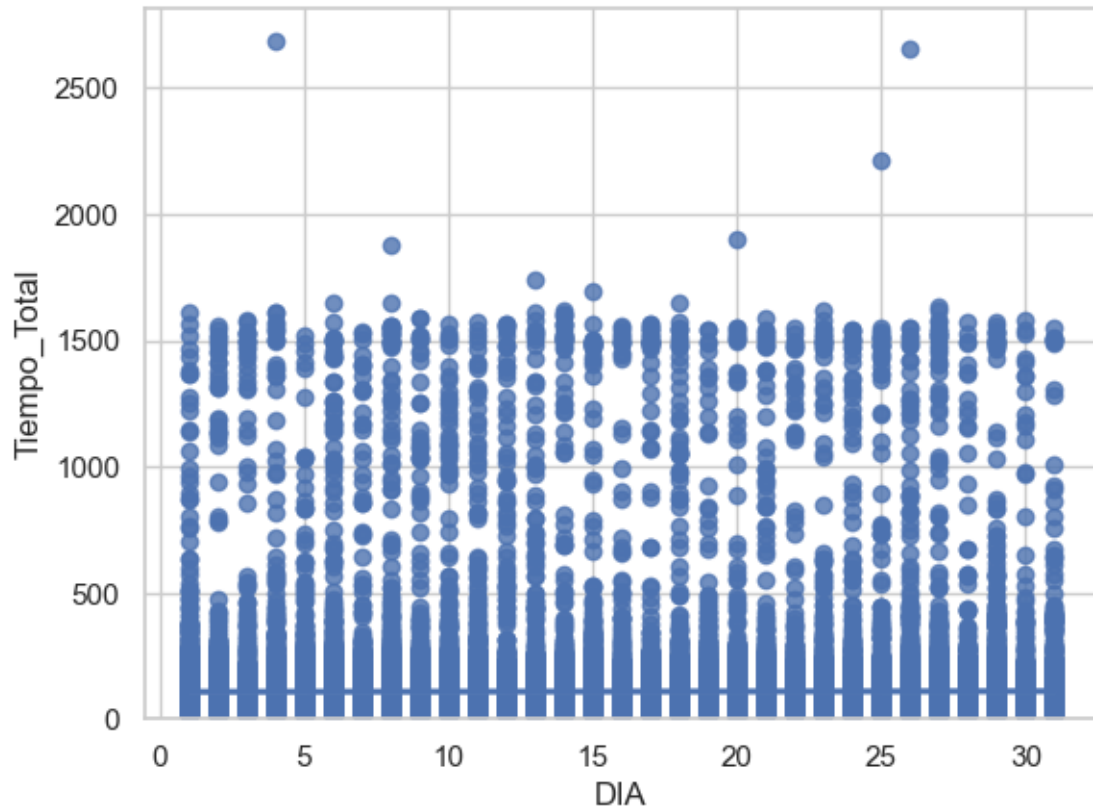


Visualicemos el diagrama de dispersión de tamaño del motor (engine-size) y precio (price)

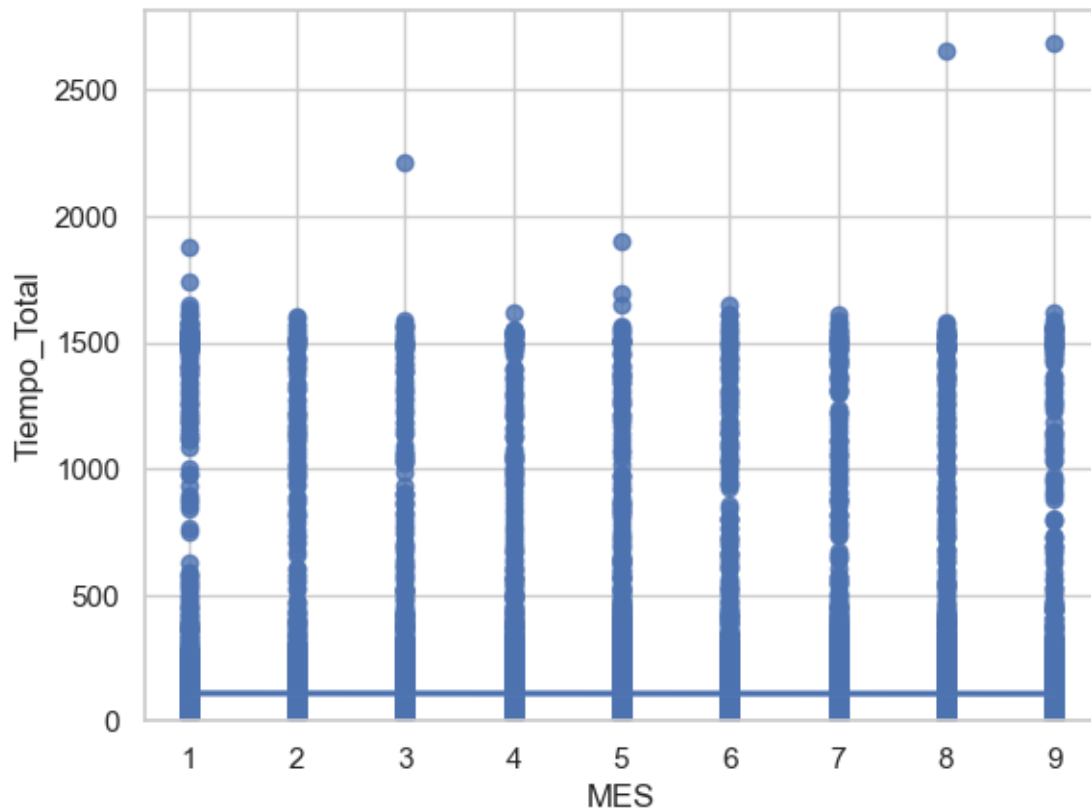
```
[ ]: # Luego, crea el gráfico de regresión
sns.set(style="whitegrid")
sns.regplot(x="HOUR", y="Tiempo_Total", data=df)
plt.ylim(0,)
plt.show()
```



```
[ ]: # Luego, crea el gráfico de regresión
sns.set(style="whitegrid")
sns.regplot(x="DIA", y="Tiempo_Total", data=df)
plt.ylim(0,)
plt.show()
```



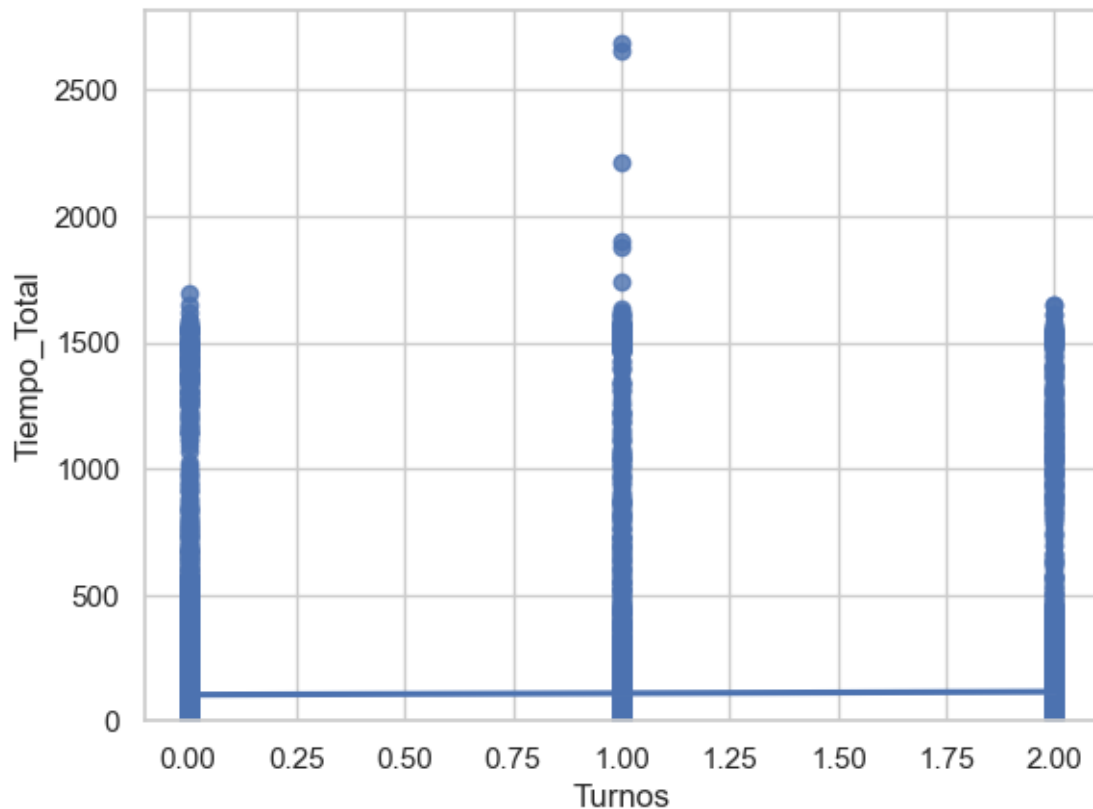
```
[ ]: # Luego, crea el gráfico de regresión
sns.set(style="whitegrid")
sns.regplot(x="MES", y="Tiempo_Total", data=df)
plt.ylim(0,)
plt.show()
```

```
[ ]: # Primero, asegúrate de que tus variables categóricas estén ordenadas
df['Turnos'] = df['Turnos'].astype('category')
df['Turnos'] = df['Turnos'].cat.as_ordered()

# Convierte la columna categórica en una variable numérica
df['Turnos'] = df['Turnos'].cat.codes

# Luego, crea el gráfico de regresión
sns.set(style="whitegrid")
sns.regplot(x="Turnos", y="Tiempo_Total", data=df)
plt.ylim(0,)
plt.show()
```



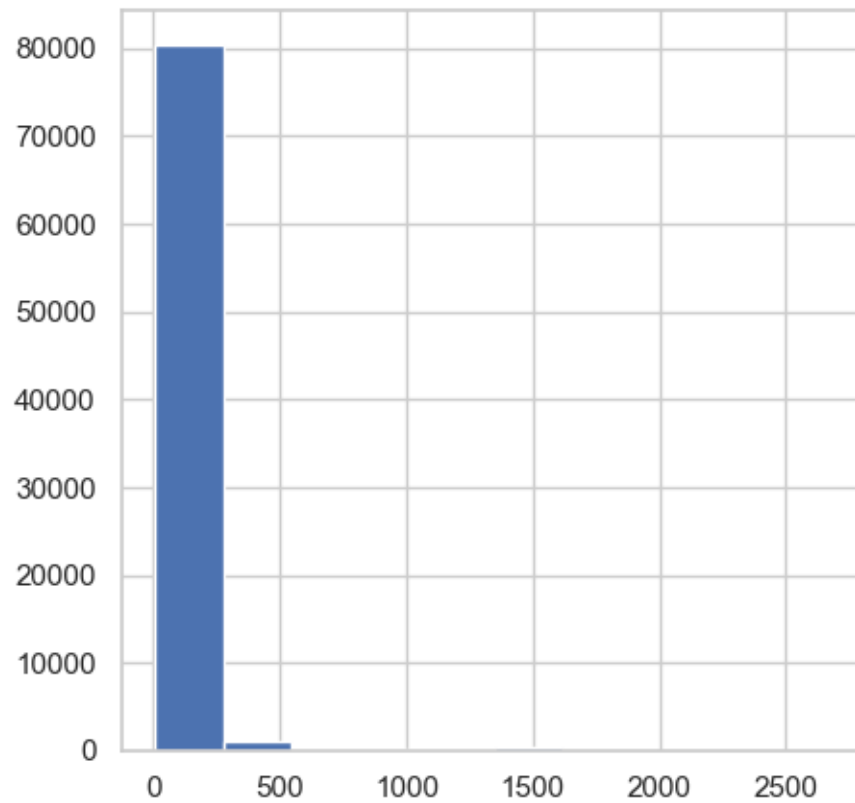
A medida que aumenta el tamaño del motor, aumenta el precio: esto indica una correlación directa positiva entre estas dos variables. El tamaño del motor parece ser un buen predictor de precio ya que la línea de regresión es casi una línea diagonal perfecta.

Visualicemos ahora highway-mpg y price. A medida que aumenta highway-mpg, el precio baja: esto indica una relación inversa/negativa entre estas dos variables. highway-mpg podría predecir el precio.

El gráfico de histograma también permite realizar un análisis sobre la variable numérica. Se utiliza el tipo de gráfico de Matplotlib hist sobre la variable age.

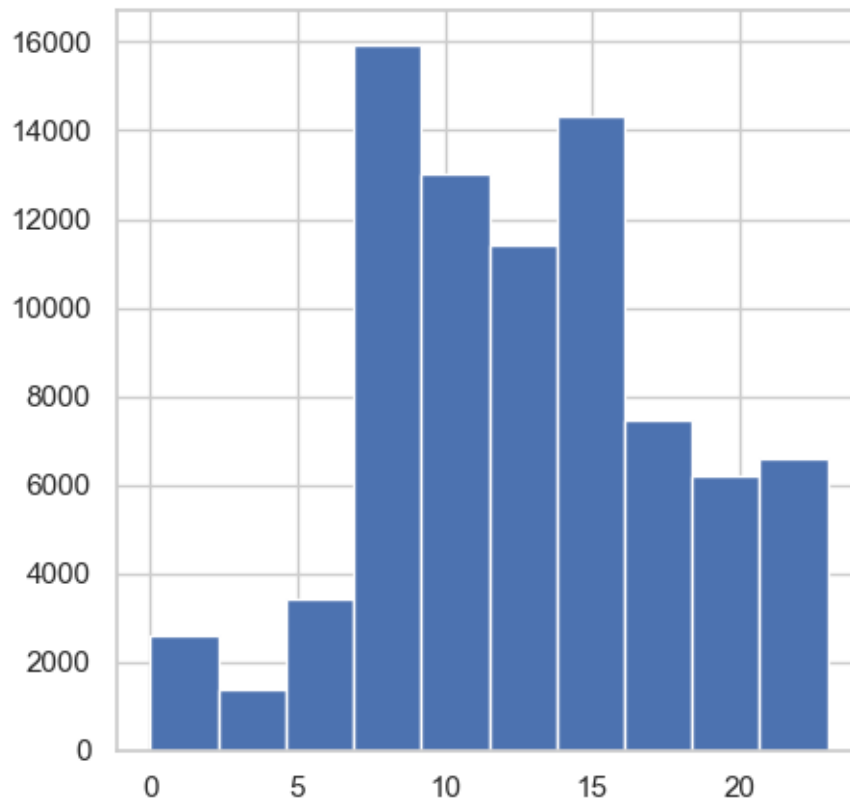
```
[ ]: df['Tiempo_Total'].hist(figsize = (5,5))
plt.show
```

```
[ ]: <function matplotlib.pyplot.show(close=None, block=None)>
```



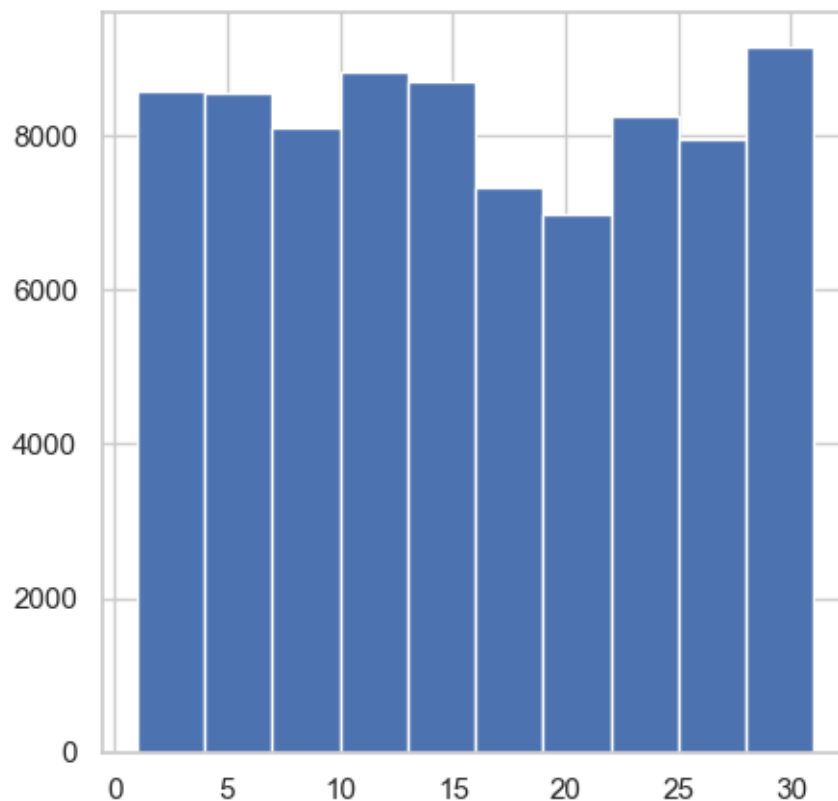
```
[ ]: df['HOUR'].hist(figsize = (5,5))  
plt.show
```

```
[ ]: <function matplotlib.pyplot.show(close=None, block=None)>
```



```
[ ]: df['DIA'].hist(figsize = (5,5))  
plt.show
```

```
[ ]: <function matplotlib.pyplot.show(close=None, block=None)>
```



Variables categóricas

Estas son variables que describen una ‘característica’ de una unidad de datos y se seleccionan de un pequeño grupo de categorías. Las variables categóricas pueden tener el tipo objeto o int64. Una buena forma de visualizar variables categóricas es mediante el uso de diagramas de caja.

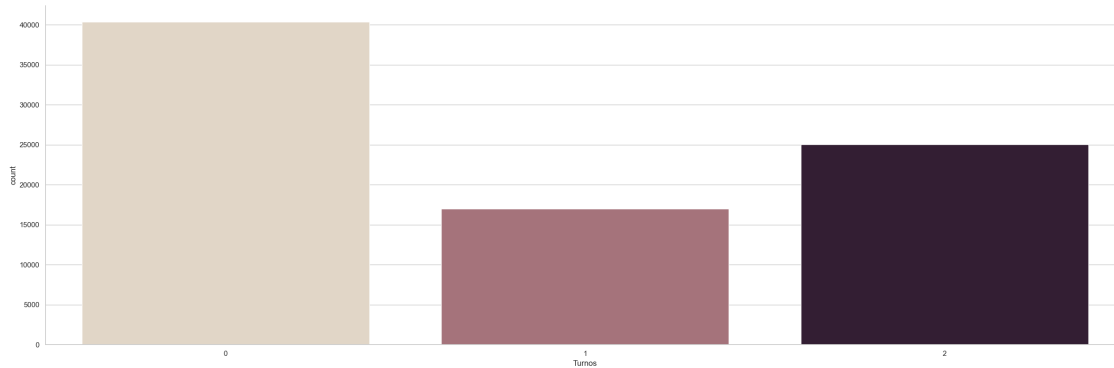
Mostremos un solo atributo, y contemos cuantos ejemplos hay de cada categoría. En este caso el atributo make que es la marca del carro:

```
[ ]: sns.set(style="whitegrid")
sns.catplot(x="Turnos", kind="count", palette="ch:.25", data=df, height = 8,
↪ aspect = 3)
```

```
c:\Users\Victor.Gomez\AppData\Local\Programs\Python\Python311\Lib\site-
packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to
tight
```

```
self._figure.tight_layout(*args, **kwargs)
```

```
[ ]: <seaborn.axisgrid.FacetGrid at 0x1b6b9ffa810>
```

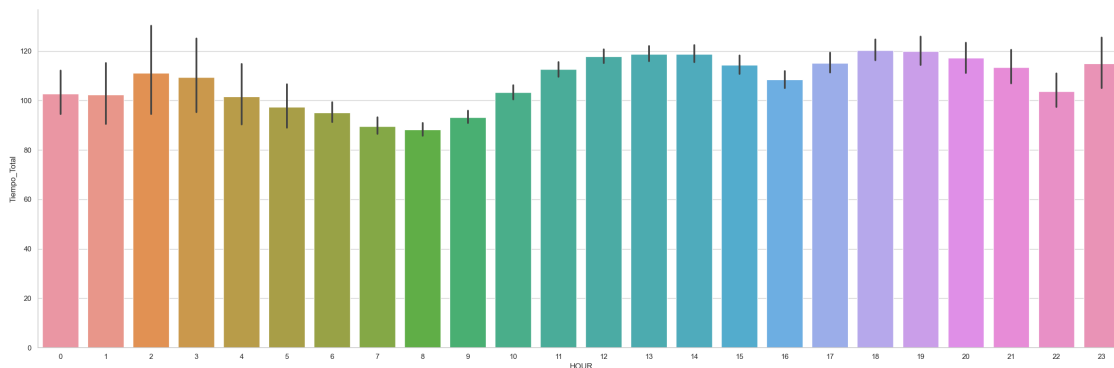


```
[ ]: sns.set(style="whitegrid")
sns.catplot(x="HOUR", y="Tiempo_Total", hue_order="class", kind="bar", data=df,
           height = 8, aspect = 3)
```

c:\Users\Victor.Gomez\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight

```
self._figure.tight_layout(*args, **kwargs)
```

```
[ ]: <seaborn.axisgrid.FacetGrid at 0x1b6ba008e50>
```



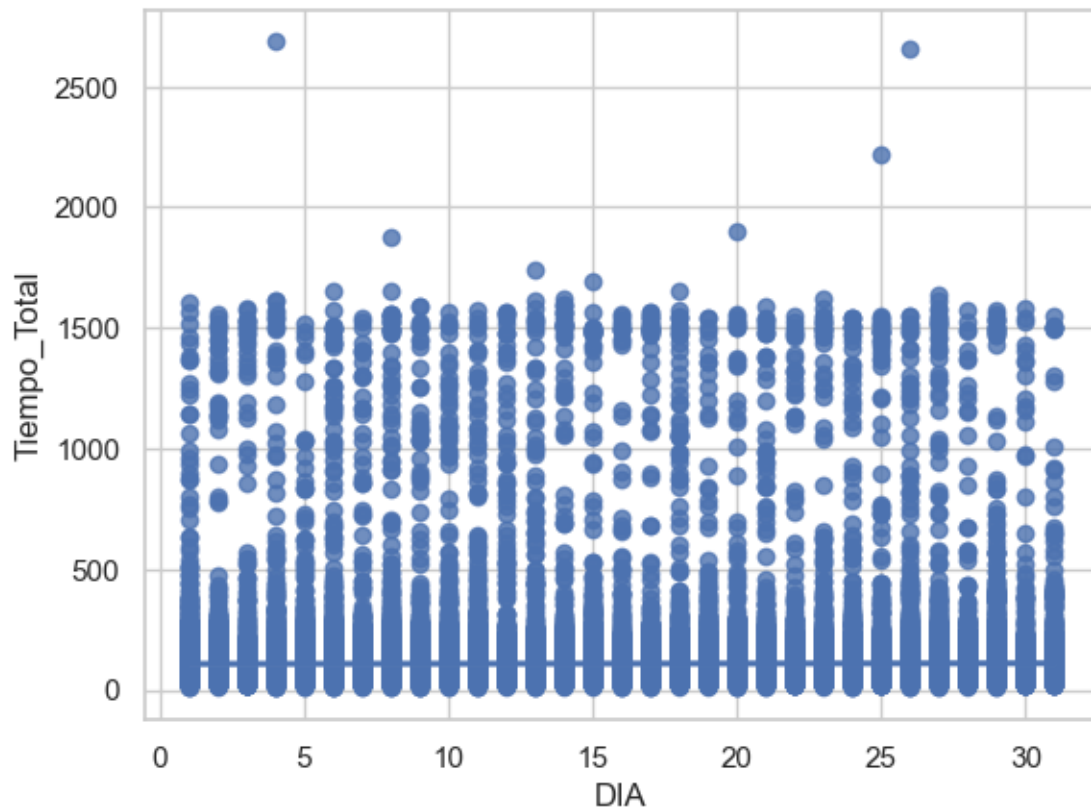
AHORA TÚ:

Visualice las variables stroke y price utilizando regplot. ¿Qué tipo de relación hay entre las variables?

```
[ ]: # Escribe tu código aquí y presiona Shift+Enter para ejecutar

sns.set(style="whitegrid")
sns.regplot(x="DIA", y="Tiempo_Total", data=df)
```

```
[ ]: <Axes: xlabel='DIA', ylabel='Tiempo_Total'>
```



Graficar las estadísticas

Podemos identificar la simetría de los datos utilizando gráficos de histograma:

```
[ ]: #Histograma del atributo "length"  
sns.distplot(df.Turnos)
```

C:\Users\Victor.Gomez\AppData\Local\Temp\ipykernel_14576\1764428100.py:2:
UserWarning:

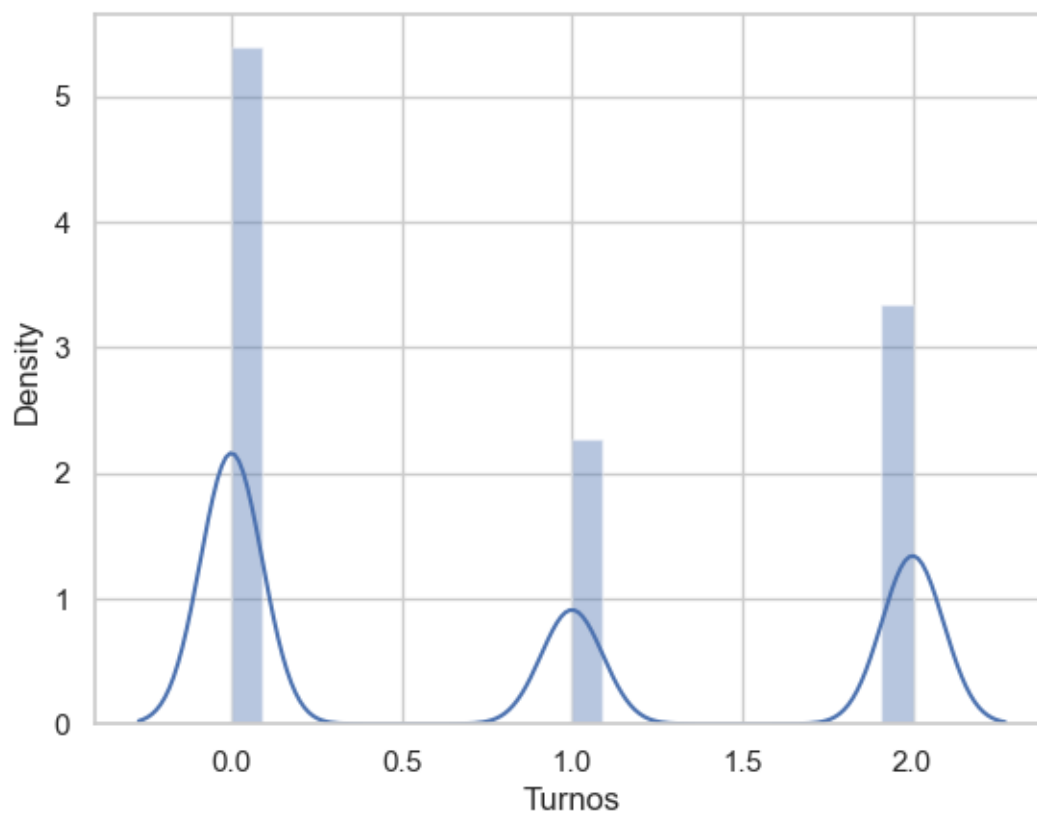
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
<https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

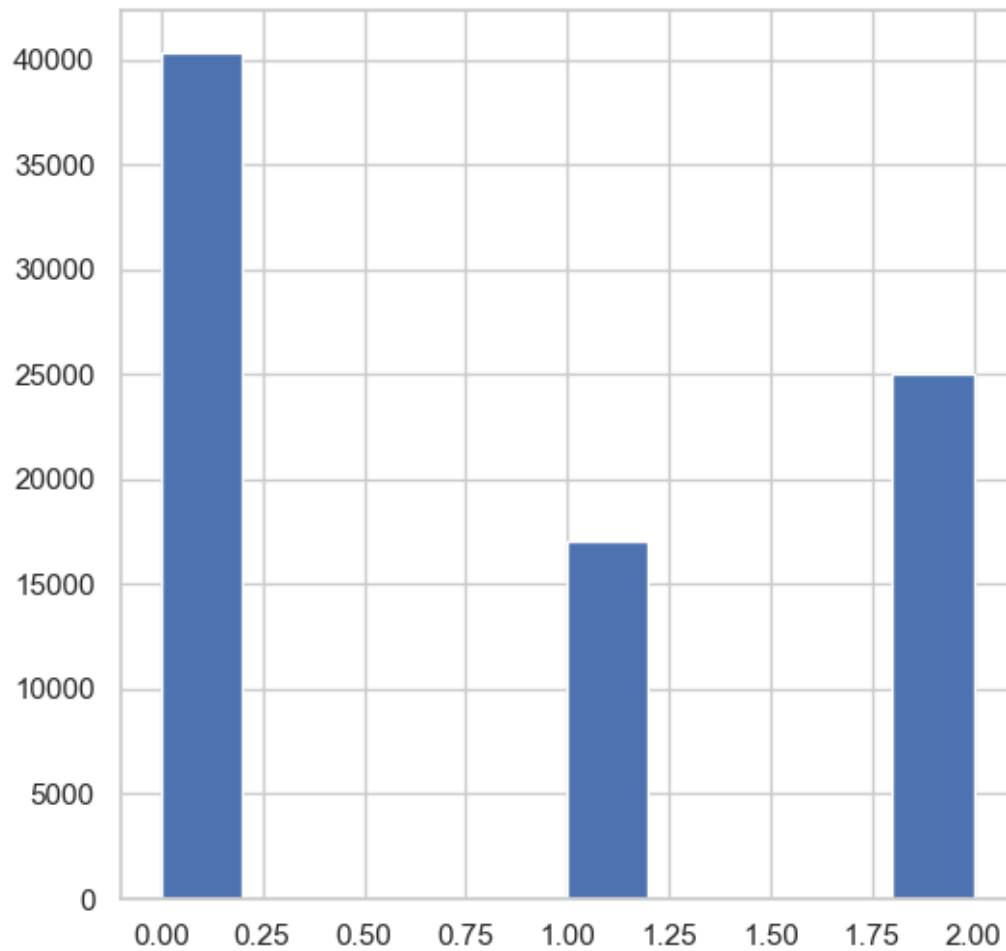
```
sns.distplot(df.Turnos)
```

```
[ ]: <Axes: xlabel='Turnos', ylabel='Density'>
```



```
[ ]: df['Turnos'].hist(figsize = (6,6))  
plt.show
```

```
[ ]: <function matplotlib.pyplot.show(close=None, block=None)>
```

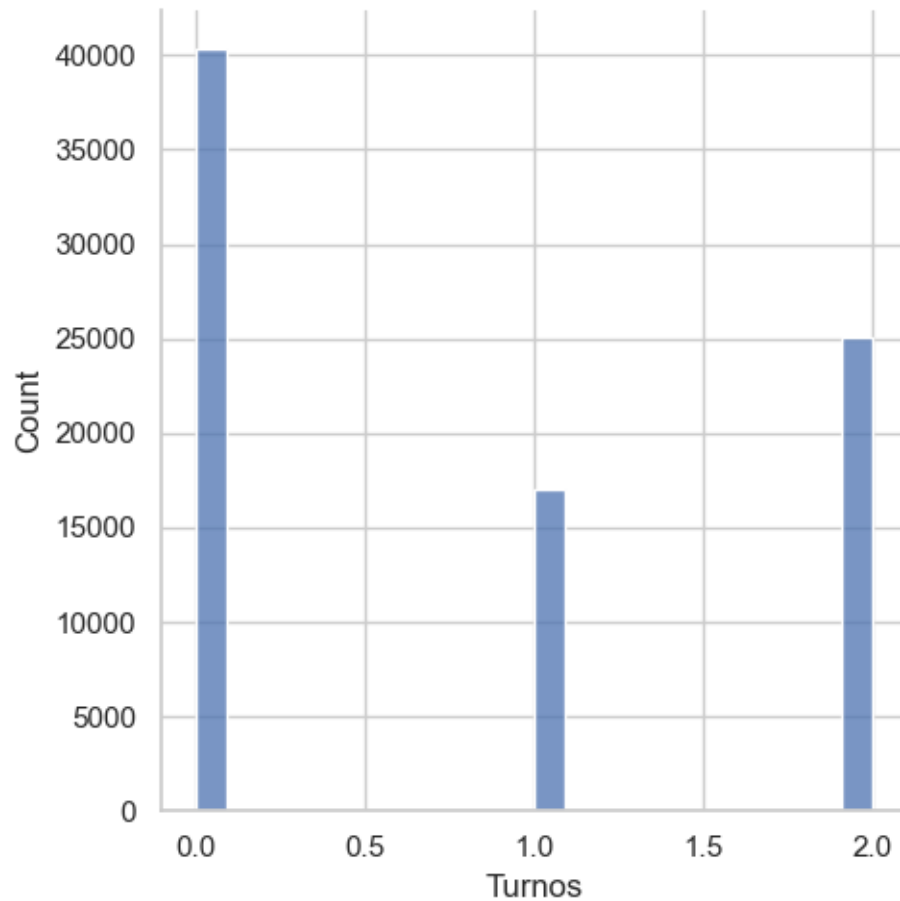



```
[ ]: sns.displot(df['Turnos'])
```

```
c:\Users\Victor.Gomez\AppData\Local\Programs\Python\Python311\Lib\site-  
packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to  
tight
```

```
    self._figure.tight_layout(*args, **kwargs)
```

```
[ ]: <seaborn.axisgrid.FacetGrid at 0x1b6ba034850>
```



```
[ ]: mean = df['Turnos'].mean()
median = df['Turnos'].median()
mode = df['Turnos'].mode()
skew = df['Turnos'].skew()
kurt = df['Turnos'].kurt()
print("La media es:", mean)
print("La mediana es:", median)
print("La moda es:", mode)
print("El sesgo es:", skew)
print("La kurtosis es:", kurt)
```

```
La media es: 0.8142229664142885
La mediana es: 1.0
La moda es: 0    0
Name: Turnos, dtype: int8
El sesgo es: 0.3684999729645169
La kurtosis es: -1.5835310691072397
```

AHORA TÚ:

Seleccione una variable y visualice su histograma. ¿Qué tipo de simetría tiene?

```
[ ]: # Escribe tu código aquí y presiona Shift+Enter para ejecutar
sns.distplot(df.HOUR)
```

C:\Users\Victor.Gomez\AppData\Local\Temp\ipykernel_14576\8436205.py:2:

UserWarning:

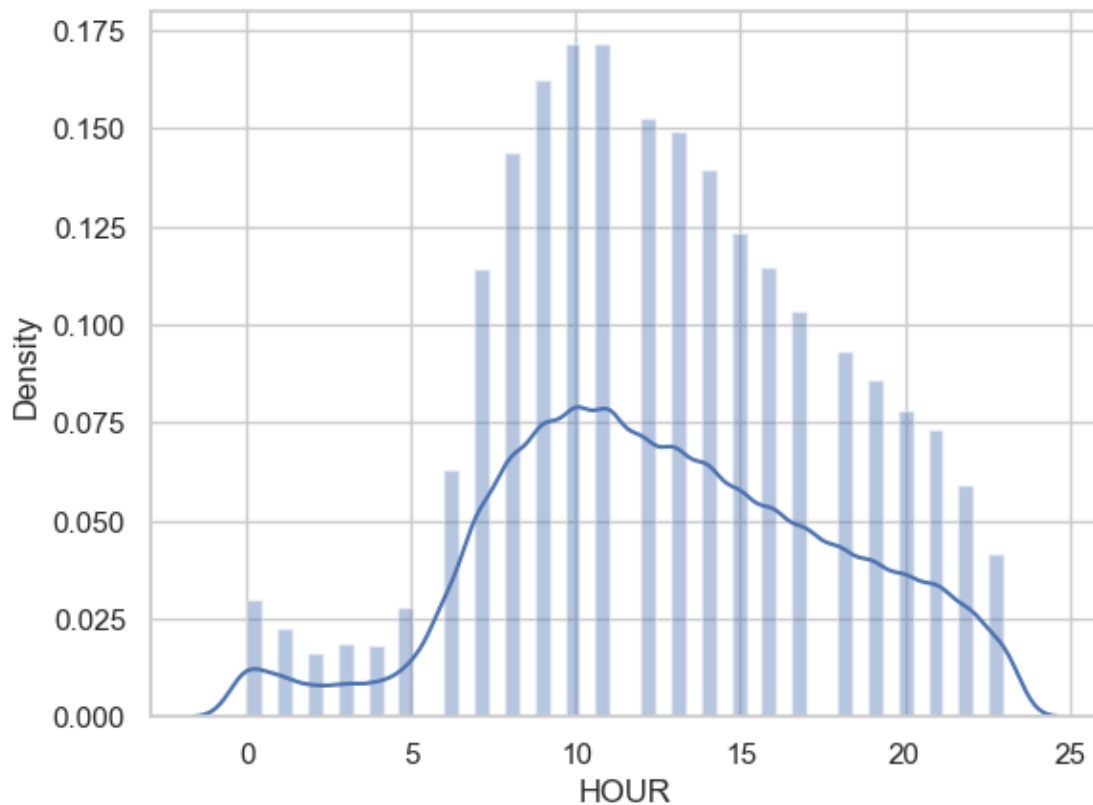
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(df.HOUR)
```

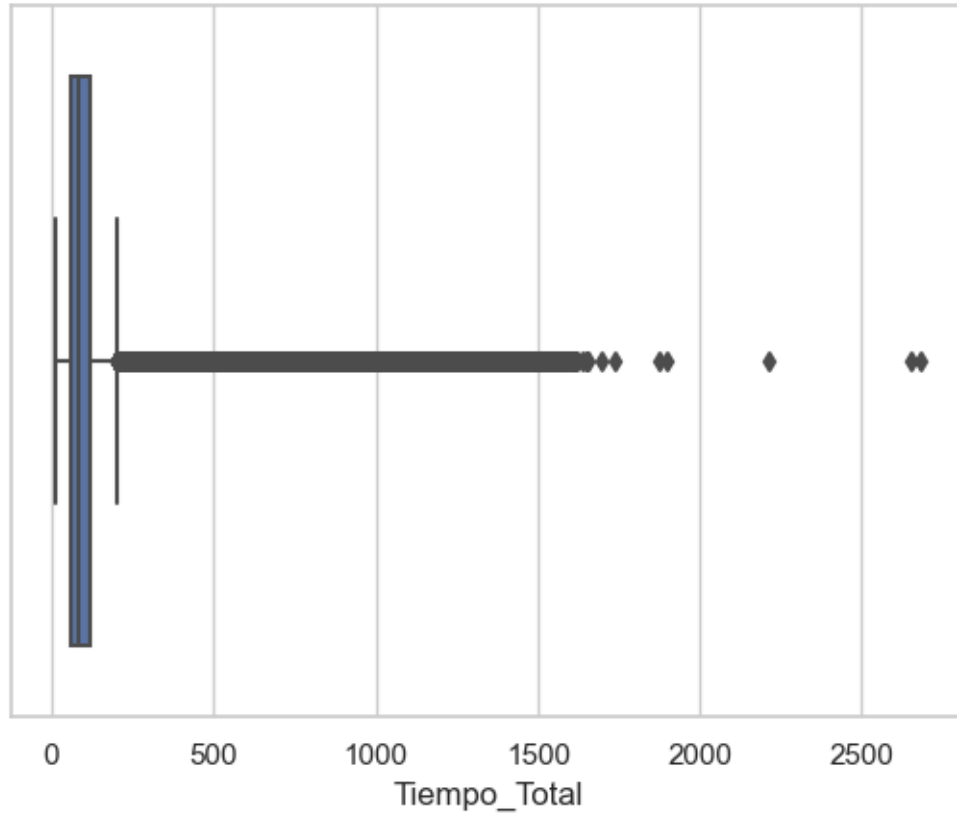
```
[ ]: <Axes: xlabel='HOUR', ylabel='Density'>
```



La dispersión de datos se puede comprobar también mediante los gráficos de tipo boxplot:

```
[ ]: sns.boxplot(x=df.Tiempo_Total)
```

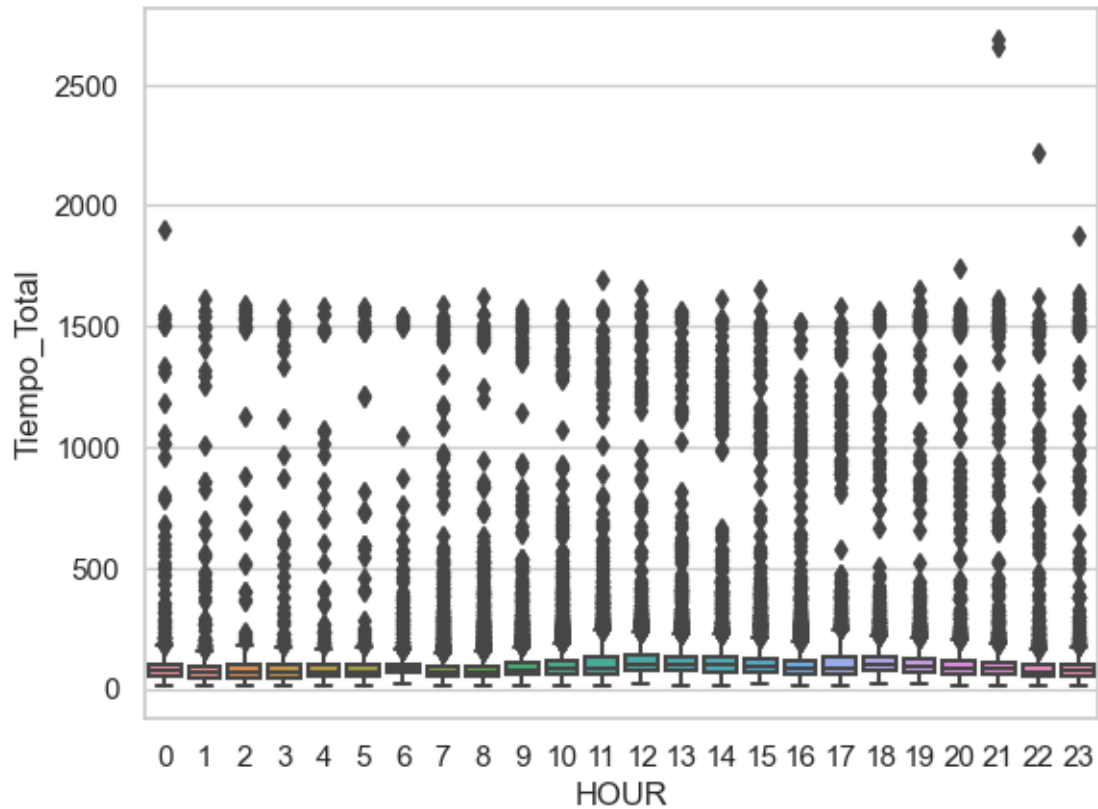
```
[ ]: <Axes: xlabel='Tiempo_Total'>
```



Representar más de una gráfico tipo boxplot permite comparar la dispersión de los datos al poder ver los resultados de forma conjunta. Veamos la relación entre body-style y price.

```
[ ]: sns.boxplot(x="HOURL", y="Tiempo_Total", data=df)
```

```
[ ]: <Axes: xlabel='HOURL', ylabel='Tiempo_Total'>
```



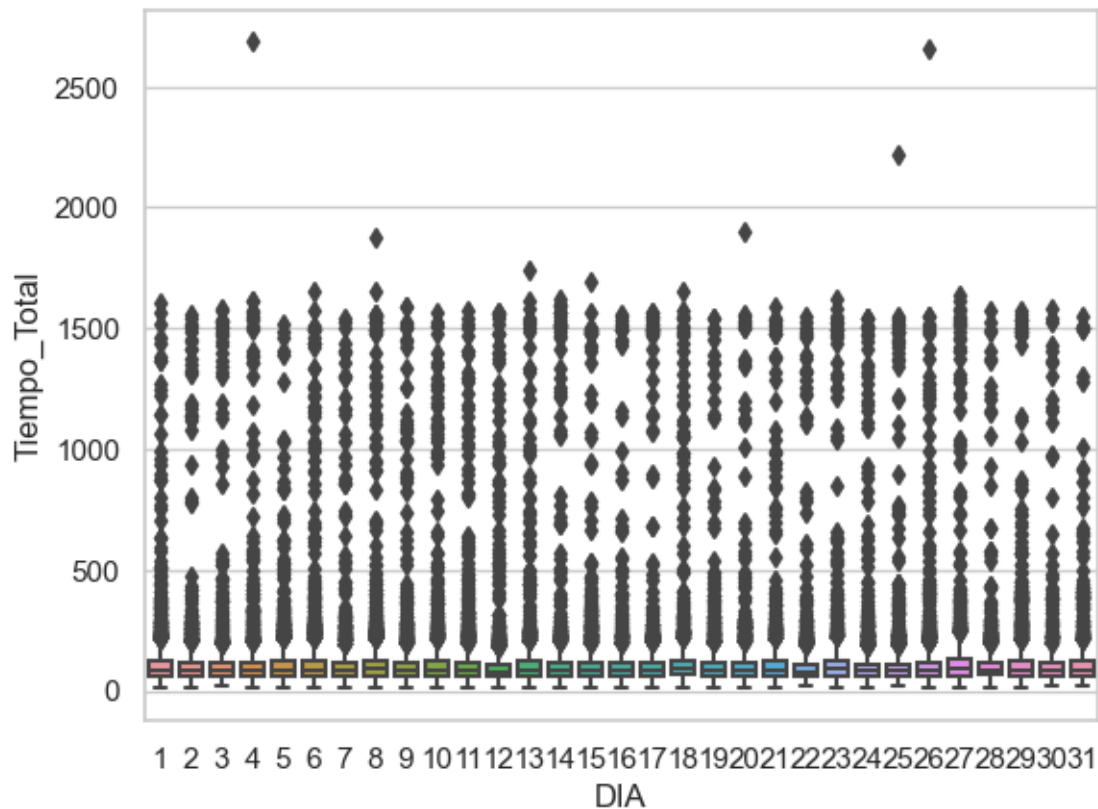
Vemos que las distribuciones de precios entre las diferentes categorías de estilo de cuerpo tienen una superposición significativa, por lo que el estilo de cuerpo no sería un buen predictor del precio.

AHORA TÚ:

Visualice las variables engine-location y price utilizando un gráfico de BoxPlot

```
[ ]: # Escribe tu código aquí y presiona Shift+Enter para ejecutar
sns.boxplot(x="DIA", y="Tiempo_Total", data=df)
```

```
[ ]: <Axes: xlabel='DIA', ylabel='Tiempo_Total'>
```



##Exportar los datos

De la misma forma, Pandas nos permite guardar el conjunto en formato CSV con el método `dataframe.to_csv()`, puede añadir la ruta al archivo y el nombre con comillas dentro de los corchetes.

Por ejemplo, si guarda el dataframe `df` como `automobile.csv` en su equipo local o en este caso dentro de Google Colab, podría usar la sintaxis siguiente:

```
[ ]: path="URGENCIAS.csv"
      df.to_csv(path)
```

Podemos leer y guardar con otros formatos y usar funciones similares a `pd.read_csv()` y `df.to_csv()` para otros formatos de datos, las funciones se muestran en la siguiente tabla:

Data Formate	Read	Save
csv	<code>pd.read_csv()</code>	<code>df.to_csv()</code>
json	<code>pd.read_json()</code>	<code>df.to_json()</code>
excel	<code>pd.read_excel()</code>	<code>df.to_excel()</code>
hdf	<code>pd.read_hdf()</code>	<code>df.to_hdf()</code>
sql	<code>pd.read_sql()</code>	<code>df.to_sql()</code>
...

##Links de ayuda interesantes:

Graficos Seaborn: <https://seaborn.pydata.org/examples/index.html>

Plotting with categorical data: <https://seaborn.pydata.org/tutorial/categorical.html>

Visualizing statistical relationships: <https://seaborn.pydata.org/tutorial/relational.html>

Funciones Generales y Ayuda Pandas: https://pandas.pydata.org/pandas-docs/stable/reference/general_functions.html

Ayuda de Pandas para función read_csv: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html?highlight=pandas%20read_csv#pandas.read_csv

How to Perform Exploratory Data Analysis with Seaborn: <https://towardsdatascience.com/how-to-perform-exploratory-data-analysis-with-seaborn-97e3413e841d>