

- **Año:** [2024]
- **Alumno/a:** [ANTONELLA ROBIOLIO]
- **Legajo:** [42904775]

✓ Pandas

A continuación, cada celda va a pedir algo distinto. Por favor, realizarlo con la menor cantidad de lineas posibles y con NumPy.

Importar pandas con el alias pd e imprimir la versión instalada.


```
import pandas as pd
```

Crear la siguiente tabla como un dataframe de Pandas donde cada linea represente un diccionario.

<i>Index</i>	nombre	edad	dni
9	Brown, James	43	30123444
3	Hamkel, Louis V.	29	44555666
7	Baptista, Carlos	28	43120111

```
data = [
    {"Index": 9, "nombre": "Brown, James", "edad": 43, "dni": 30123444},
    {"Index": 3, "nombre": "Hamkel, Louis V.", "edad": 29, "dni": 44555666},
    {"Index": 7, "nombre": "Baptista, Carlos", "edad": 28, "dni": 43120111}
]
```

```
df = pd.DataFrame(data).set_index("Index")
df
```




	nombre	edad	dni
Index			
9	Brown, James	43	30123444
3	Hamkel, Louis V.	29	44555666
7	Baptista, Carlos	28	43120111

Crear la siguiente tabla como un dataframe de Pandas donde todas las lineas esten dentro de un solo diccionario.

<i>Index</i>	nombre	edad	dni
9	Brown, James	43	30123444
3	Hamkel, Louis V.	29	44555666
7	Baptista, Carlos	28	43120111

```
data = {
    "Index": [9, 3, 7],
    "nombre": ["Brown, James", "Hamkel, Louis V.", "Baptista, Carlos"],
    "edad": [43, 29, 28],
    "dni": [30123444, 44555666, 43120111]
}
```

```
df = pd.DataFrame(data).set_index("Index")
df
```



	nombre	edad	dni
Index			
9	Brown, James	43	30123444
3	Hamkel, Louis V.	29	44555666
7	Baptista, Carlos	28	43120111

Crear la siguiente tabla como un dataframe de Pandas donde se usen unicamente listas.

<i>Index</i>	nombre	edad	dni
9	Brown, James	43	30123444
3	Hamkel, Louis V.	29	44555666
7	Baptista, Carlos	28	43120111

```
indices = [9, 3, 7]
nombres = ["Brown, James", "Hamkel, Louis V.", "Baptista, Carlos"]
edades = [43, 29, 28]
```

dnis = [30123444, 44555666, 43120111]

```
df = pd.DataFrame({
    "Index": indices,
    "nombre": nombres,
    "edad": edades,
    "dni": dnis
}).set_index("Index")
```

df


Crear la siguiente tabla como un dataframe de Pandas donde se usen unicamente Series .

<i>Index</i>	nombre	edad	dni
9	Brown, James	43	30123444
3	Hamkel, Louis V.	29	44555666
7	Baptista, Carlos	28	43120111

```
indices = pd.Series([9, 3, 7], name="Index")
nombres = pd.Series(["Brown, James", "Hamkel, Louis V.", "Baptista, Carlos"], name="nombre")
edades = pd.Series([43, 29, 28], name="edad")
dnis = pd.Series([30123444, 44555666, 43120111], name="dni")
```

```
df = pd.DataFrame({
    "Index": indices,
    "nombre": nombres,
    "edad": edades,
    "dni": dnis
}).set_index("Index")
```

df



	nombre	edad	dni
Index			
9	Brown, James	43	30123444
3	Hamkel, Louis V.	29	44555666
7	Baptista, Carlos	28	43120111

Reutilice cualquiera de los dataframe hechos anteriormente pero agregue la columna fecha con el tipo de dato relacionado a fechas.


<i>Index</i>	nombre	edad	dni	fecha
9	Brown, James	43	30123444	12/08/1981
3	Hamkel, Louis V.	29	44555666	10/04/1995
7	Baptista, Carlos	28	43120111	28/05/1996

```
data = [
    {"Index": 9, "nombre": "Brown, James", "edad": 43, "dni": 30123444},
    {"Index": 3, "nombre": "Hamkel, Louis V.", "edad": 29, "dni": 44555666},
    {"Index": 7, "nombre": "Baptista, Carlos", "edad": 28, "dni": 43120111}
]
```

```
df = pd.DataFrame(data).set_index("Index")
```

```
# Agregar la columna de fechas
df["fecha"] = ["12/08/1981", "10/04/1995", "28/05/1996"]
```

print(df)



	nombre	edad	dni	fecha
Index				
9	Brown, James	43	30123444	12/08/1981
3	Hamkel, Louis V.	29	44555666	10/04/1995
7	Baptista, Carlos	28	43120111	28/05/1996

Ejecute la siguiente celda. Se va a descargar un archivo llamado u.user .

```
!wget https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.user
```

```
--2024-11-19 16:55:44-- https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.user
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.110.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 22667 (22K) [text/plain]
Saving to: 'u.user'

u.user          100%[=====>]  22.14K  --.-KB/s   in 0.007s
```

Lea el archivo con pandas y muestre las primeras 5 filas.

```
df_users = pd.read_csv('u.user', delimiter='|')
print(df_users.head())
```

```

↗
  user_id  age gender occupation zip_code
0         1   24     M  technician   85711
1         2   53     F    other    94043
2         3   23     M    writer   32067
3         4   24     M  technician  43537
4         5   33     F    other   15213
```

Utilice la columna `user_id` como índice y saque dicha columna del dataframe

```
df_users.set_index('user_id', inplace=True)
print(df_users.head())
```

```

↗
  age gender occupation zip_code
user_id
1      24     M  technician   85711
2      53     F    other    94043
3      23     M    writer   32067
4      24     M  technician  43537
5      33     F    other   15213
```

¿Cuántas categorías de trabajos hay?

```
num_categorias_trabajos = df_users['occupation'].nunique()

print("Número de categorías de trabajos:", num_categorias_trabajos)
```

```
↗ Número de categorías de trabajos: 21
```

Reporte el porcentaje de personas que tiene cada ocupación.

```
porcentaje_ocupacion = (df_users['occupation'].value_counts(normalize=True) * 100).round(2)

print("Porcentaje de personas por ocupación:")
print(porcentaje_ocupacion)
```

```

↗ Porcentaje de personas por ocupación:
occupation
student      20.78
other        11.13
educator     10.07
administrator  8.38
engineer      7.10
programmer    7.00
librarian     5.41
writer        4.77
executive     3.39
scientist     3.29
artist        2.97
technician    2.86
marketing     2.76
entertainment 1.91
healthcare    1.70
retired       1.48
lawyer        1.27
salesman      1.27
none          0.95
homemaker     0.74
doctor        0.74
Name: proportion, dtype: float64
```

Reporte el promedio de edad de los estudiantes usando indexeo booleano.

```
edad_promedio_estudiantes = df_users[df_users['occupation'] == 'student']['age'].mean()

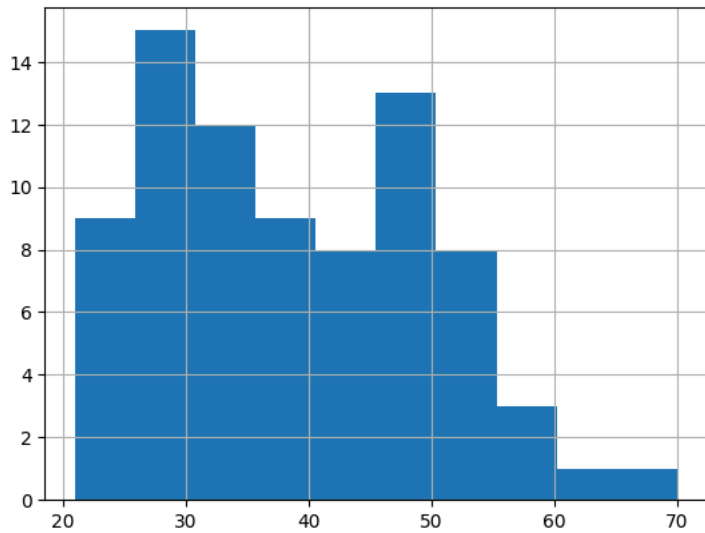
print("Promedio de edad de los estudiantes:", round(edad_promedio_estudiantes, 2))
```

```
↗ Promedio de edad de los estudiantes: 22.08
```

Mostrar, con una sola línea y sin importar `matplotlib`, un histograma de las edades de las personas que son administradores.

```
df_users[df_users['occupation'] == 'administrator']['age'].hist()
```

<Axes: >



Reemplace, sin usar for, en la columna gender F por female y M por male.

```
df_users['gender'] = df_users['gender'].replace({'F': 'female', 'M': 'male'})
print(df_users.head())
```

```
user_id  age  gender  occupation  zip_code
1         24   male   technician    85711
2         53 female    other       94043
3         23   male    writer       32067
4         24   male   technician    43537
5         33 female    other       15213
```

✓ Yahoo! Finance

Vamos a analizar acciones. La siguiente línea accede a Yahoo Finance y devuelve un DataFrame con los valores de la acción cada día desde el 1980.

```
import yfinance as yf
dataframe = yf.download('AAPL', start="1980-01-01", end="2030-01-01")
```

[*****100%*****] 1 of 1 completed

¿Cuál es el registro más viejo? Imprimirlo.

```
registro_mas_viejo = dataframe.head(1)
```

```
print("Registro más viejo:")
print(registro_mas_viejo)
```

```
Registro más viejo:
Price      Adj Close      Close      High      Low      Open  \
Ticker      AAPL      AAPL      AAPL      AAPL      AAPL
Date
1980-12-12 00:00:00+00:00  0.098834  0.128348  0.128906  0.128348  0.128348

Price      Volume
Ticker      AAPL
Date
1980-12-12 00:00:00+00:00  469033600
```

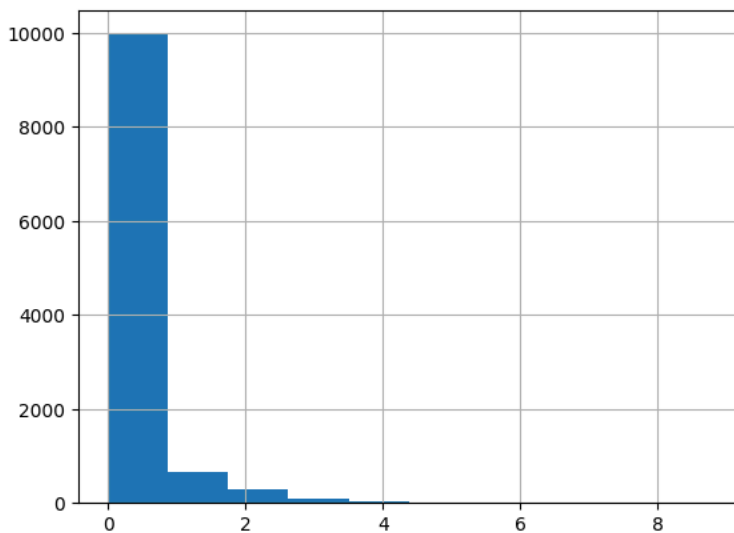
Cree la columna Average tal que

$$Average = \frac{High - Low}{2}$$

y muestre con un histograma dicha columna.

```
dataframe['Average'] = (dataframe['High'] - dataframe['Low']) / 2
dataframe['Average'].hist()
```

↵ <Axes: >



Con matplotlib, muestre como Average fue evolucionando *al final de cada año*.

```
import matplotlib.pyplot as plt
dataframe['Year'] = dataframe.index.year
average_anual = dataframe.groupby('Year')['Average'].last()
plt.plot(average_anual.index, average_anual.values)
plt.xlabel('Año')
plt.ylabel('Average')
plt.title('Evolución de Average al final de cada año')
plt.grid(True)
plt.show()
```

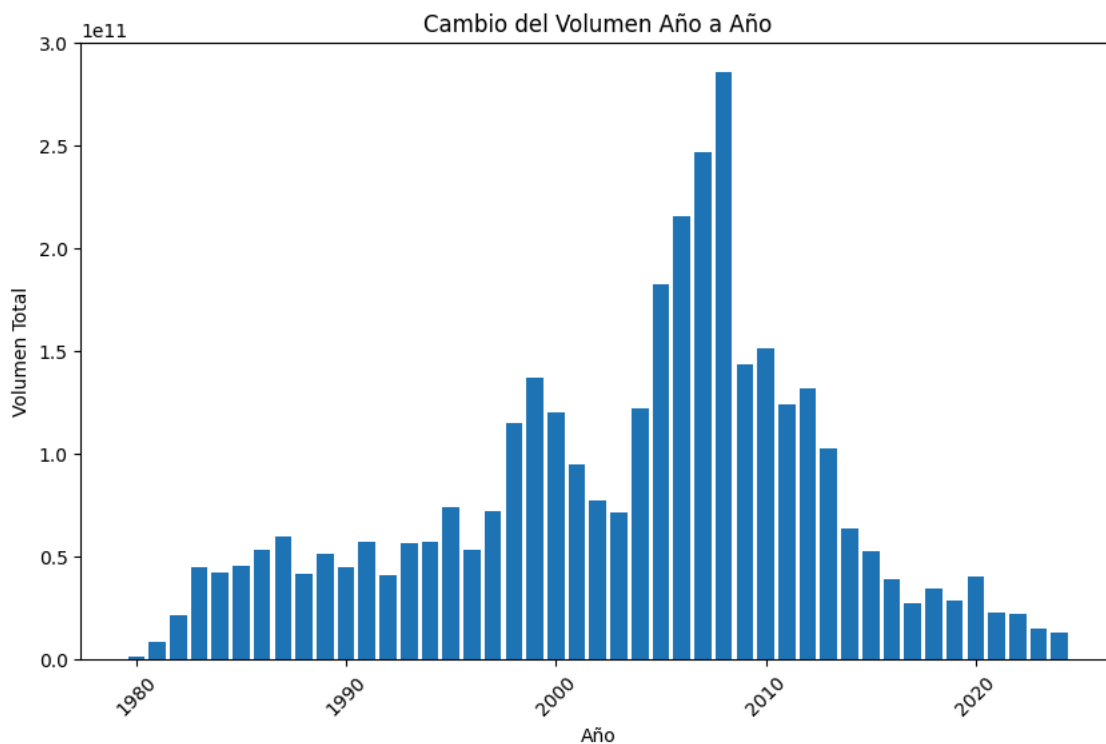
↵



Muestre con un gráfico de barras, como el volumen fue cambiando *año a año*.

```
import matplotlib.pyplot as plt

dataframe['Year'] = dataframe.index.year
volumen_anual = dataframe.groupby('Year')['Volume'].sum()
plt.figure(figsize=(10, 6))
plt.bar(volumen_anual.index, volumen_anual.values.ravel())
plt.xlabel('Año')
plt.ylabel('Volumen Total')
plt.title('Cambio del Volumen Año a Año')
plt.xticks(rotation=45)
plt.show()
```



✓ Cancelaciones y Delays de vuelos del 2015

Creese una cuenta en Kaggle e importe los archivos del dataset del siguiente link: <https://www.kaggle.com/datasets/usdot/flight-delays>. Cree los dataframes airlines, airports, y flights apartir de esos archivos.

```
import kagglehub
import pandas as pd
path = kagglehub.dataset_download("usdot/flight-delays")
```

```
airlines_file = f"{path}/airlines.csv"
airports_file = f"{path}/airports.csv"
flights_file = f"{path}/flights.csv"
```

```
airlines = pd.read_csv(airlines_file)
airports = pd.read_csv(airports_file)
flights = pd.read_csv(flights_file)
```

```
print("Airlines DataFrame:")
print(airlines.head())
```

```
print("\nAirports DataFrame:")
print(airports.head())
```

```
print("\nFlights DataFrame:")
print(flights.head())
```

```
<ipython-input-39-c388f5f65c16>:11: DtypeWarning: Columns (7,8) have mixed types. Specify dtype option on import or set low_memory=False
flights = pd.read_csv(flights_file)
```

Airlines DataFrame:

	IATA_CODE	AIRLINE
0	UA	United Air Lines Inc.
1	AA	American Airlines Inc.
2	US	US Airways Inc.
3	F9	Frontier Airlines Inc.
4	B6	JetBlue Airways

Airports DataFrame:

	IATA_CODE	AIRPORT	CITY	STATE	COUNTRY	\
0	ABE	Lehigh Valley International Airport	Allentown	PA	USA	
1	ABI	Abilene Regional Airport	Abilene	TX	USA	
2	ABQ	Albuquerque International Sunport	Albuquerque	NM	USA	
3	ABR	Aberdeen Regional Airport	Aberdeen	SD	USA	
4	ABY	Southwest Georgia Regional Airport	Albany	GA	USA	

	LATITUDE	LONGITUDE
0	40.65236	-75.44040
1	32.41132	-99.68190
2	35.04022	-106.60919
3	45.44906	-98.42183
4	31.53552	-84.19447

```

Flights DataFrame:
  YEAR  MONTH  DAY  DAY_OF_WEEK  AIRLINE  FLIGHT_NUMBER  TAIL_NUMBER \
0  2015      1    1           4      AS             98      N407AS
1  2015      1    1           4      AA            2336      N3KUAA
2  2015      1    1           4      US             840      N171US
3  2015      1    1           4      AA            258      N3HYAA
4  2015      1    1           4      AS             135      N527AS

  ORIGIN_AIRPORT  DESTINATION_AIRPORT  SCHEDULED_DEPARTURE  ...  ARRIVAL_TIME \
0      ANC          SEA              5  ...      408.0
1      LAX          PBI             10  ...      741.0
2      SFO          CLT             20  ...      811.0
3      LAX          MIA             20  ...      756.0
4      SEA          ANC             25  ...      259.0

  ARRIVAL_DELAY  DIVERTED  CANCELLED  CANCELLATION_REASON  AIR_SYSTEM_DELAY \
0      -22.0          0          0              NaN          NaN
1       -9.0          0          0              NaN          NaN
2       5.0          0          0              NaN          NaN
3      -9.0          0          0              NaN          NaN
4     -21.0          0          0              NaN          NaN

  SECURITY_DELAY  AIRLINE_DELAY  LATE_AIRCRAFT_DELAY  WEATHER_DELAY
0          NaN          NaN          NaN          NaN
1          NaN          NaN          NaN          NaN
2          NaN          NaN          NaN          NaN
3          NaN          NaN          NaN          NaN
4          NaN          NaN          NaN          NaN

[5 rows x 31 columns]

```

Combine (*join*) las tablas airlines, airports, y flights en una sola tabla.

```

flights_combined = flights.merge(airlines, how='left', left_on='AIRLINE', right_on='IATA_CODE')
flights_combined = flights_combined.merge(airports, how='left', left_on='ORIGIN_AIRPORT', right_on='IATA_CODE')
print("Tabla combinada:")
print(flights_combined.head())

```

```

➡ Tabla combinada:
  YEAR  MONTH  DAY  DAY_OF_WEEK  AIRLINE_x  FLIGHT_NUMBER  TAIL_NUMBER \
0  2015      1    1           4      AS             98      N407AS
1  2015      1    1           4      AA            2336      N3KUAA
2  2015      1    1           4      US             840      N171US
3  2015      1    1           4      AA            258      N3HYAA
4  2015      1    1           4      AS             135      N527AS

  ORIGIN_AIRPORT  DESTINATION_AIRPORT  SCHEDULED_DEPARTURE  ...  WEATHER_DELAY \
0      ANC          SEA              5  ...      NaN
1      LAX          PBI             10  ...      NaN
2      SFO          CLT             20  ...      NaN
3      LAX          MIA             20  ...      NaN
4      SEA          ANC             25  ...      NaN

  IATA_CODE_x  AIRLINE_y  IATA_CODE_y \
0      AS  Alaska Airlines Inc.      ANC
1      AA  American Airlines Inc.      LAX
2      US      US Airways Inc.      SFO
3      AA  American Airlines Inc.      LAX
4      AS  Alaska Airlines Inc.      SEA

  AIRPORT  CITY  STATE  COUNTRY \
0  Ted Stevens Anchorage International Airport  Anchorage  AK  USA
1      Los Angeles International Airport  Los Angeles  CA  USA
2      San Francisco International Airport  San Francisco  CA  USA
3      Los Angeles International Airport  Los Angeles  CA  USA
4      Seattle-Tacoma International Airport  Seattle  WA  USA

  LATITUDE  LONGITUDE
0  61.17432 -149.99619
1  33.94254 -118.40807
2  37.61900 -122.37484
3  33.94254 -118.40807
4  47.44898 -122.30931

[5 rows x 40 columns]

```

¿Cuántos vuelos fueron al aeropuerto JFK?

```

vuelos_a_JFK = flights[flights['DESTINATION_AIRPORT'] == 'JFK'].shape[0]

print(f"Número de vuelos al aeropuerto JFK: {vuelos_a_JFK}")

```

➡ Número de vuelos al aeropuerto JFK: 93809

¿Cuántos vuelos hizo la aerolínea AA?

```
vuelos_AA = flights[flights['AIRLINE'] == 'AA'].shape[0]
```

```
print(f"Número de vuelos realizados por la aerolínea AA: {vuelos_AA}")
```

```
➤ Número de vuelos realizados por la aerolínea AA: 725984
```

¿Que aerolíneas (las primeras 5) tuvo la menor cantidad de vuelos con atrasos? Imprimirlas.

```
vuelos_con_retraso = flights[flights['DEPARTURE_DELAY'] > 0]
retrasos_por_aerolinea = vuelos_con_retraso['AIRLINE'].value_counts()
menor_retraso_aerolineas = retrasos_por_aerolinea.nsmallest(5)
```

```
print("Las 5 aerolíneas con la menor cantidad de vuelos con retrasos:")
print(menor_retraso_aerolineas)
```

```
➤ Las 5 aerolíneas con la menor cantidad de vuelos con retrasos:
AIRLINE
HA      20146
VX      23379
F9      34893
AS      43566
NK      52089
Name: count, dtype: int64
```

¿Que aerolíneas (las primeras 5) tuvo la mayor cantidad de vuelos con atrasos? Imprimirlas.

```
vuelos_con_retraso = flights[flights['DEPARTURE_DELAY'] > 0]
retrasos_por_aerolinea = vuelos_con_retraso['AIRLINE'].value_counts()
mayor_retraso_aerolineas = retrasos_por_aerolinea.head(5)
```

```
print("Las 5 aerolíneas con la mayor cantidad de vuelos con retrasos:")
print(mayor_retraso_aerolineas)
```

```
➤ Las 5 aerolíneas con la mayor cantidad de vuelos con retrasos:
AIRLINE
WN      566807
DL      282463
UA      256550
AA      245904
OO      171572
Name: count, dtype: int64
```

Haga un resumen de las razones por la cual los vuelos se atrasan.

```
retraso_columns = ['WEATHER_DELAY', 'SECURITY_DELAY', 'LATE_AIRCRAFT_DELAY']
razones_retraso = flights[retraso_columns].sum()
```

```
print("Resumen de las razones por las cuales los vuelos se atrasan:")
print(razones_retraso)
```

```
➤ Resumen de las razones por las cuales los vuelos se atrasan:
WEATHER_DELAY      3100233.0
SECURITY_DELAY      80985.0
LATE_AIRCRAFT_DELAY 24961931.0
dtype: float64
```

Compruebe si hay columnas con celdas vacías.

```
columnas_con_celdas_vacias = flights.isnull().sum()
print("Columnas con celdas vacías:")
print(columnas_con_celdas_vacias[columnas_con_celdas_vacias > 0])
```

```
➤ Columnas con celdas vacías:
TAIL_NUMBER      14721
DEPARTURE_TIME    86153
DEPARTURE_DELAY   86153
TAXI_OUT          89047
WHEELS_OFF        89047
SCHEDULED_TIME      6
ELAPSED_TIME     105071
AIR_TIME          105071
WHEELS_ON         92513
TAXI_IN           92513
ARRIVAL_TIME      92513
ARRIVAL_DELAY     105071
CANCELLATION_REASON 5729195
AIR_SYSTEM_DELAY  4755640
SECURITY_DELAY    4755640
AIRLINE_DELAY     4755640
LATE_AIRCRAFT_DELAY 4755640
```



```
WEATHER_DELAY      4755640
dtype: int64
```

Haga una imputación de datos COMPLETA del dataframe. Pueden escoger cualquier estrategia y no necesariamente todas las columnas deben seguir la misma estrategia.

```
import pandas as pd
# Imputación de columnas numéricas
for column in flights.select_dtypes(include='number').columns:
    flights[column] = flights[column].fillna(flights[column].median())

# Imputación de columnas categóricas
for column in flights.select_dtypes(include='object').columns:
    flights[column] = flights[column].fillna(flights[column].mode()[0])

# Imputación específica para las columnas de retraso
retraso_columnas = ['WEATHER_DELAY', 'SECURITY_DELAY', 'LATE_AIRCRAFT_DELAY']
for column in retraso_columnas:
    if column in flights.columns:
        flights[column] = flights[column].fillna(0)

print("Imputación de datos completa realizada.")
```

🔄 Imputación de datos completa realizada.