Comprensión de los Datos

In [64]:
```
"""
Jorge Eduardo García Pereda

A01230894

ITC
"""
```

Out[64]:
```
'\nJorge Eduardo García Pereda\n\nA01230894\n\nITC\n'
```

In [67]:
```
#importa librerías
import pandas as pd
```

Descripción de Variables

Pclass Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd): Categórica Nominal survival Survival (0 = No; 1 = Yes)
name Name
sex Sex
age Age
sibsp Number of Siblings/Spouses Aboard
parch Number of Parents/Children Aboard
ticket Ticket Number
fare Passenger Fare (British pound)
cabin Cabin
embarked Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)
boat Lifeboat
body Body Identification Number
home.dest Home/Destination

Ejemplo: Crear un objeto DataFrame con base en un archivo .csv

In [17]:
```
#lee archivo csv
df = pd.read_csv('diabetes.csv')
```

In [18]:
```
#Usa función shape para revisar el total de renglones y columnas
df.shape
```

Out[18]:
```
(768, 9)
```

In [19]:
```
#Revisa los primeros 5 renglones del dataset usando la función head()
df.head(5)
```

Out[19]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148	72	35	0	33.6	0
1	1	85	66	29	0	26.6	0
2	8	183	64	0	0	23.3	0
3	1	89	66	23	94	28.1	0
4	0	137	40	35	168	43.1	2

In [20]:
```
#Revisa los últimos 5 renglones del dataset usando la función tail()
df.tail(5)
```

Out[20]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

In [21]:
```
#Revisa la información mas completa del conjunto de datos usando la función info()
#Muestra el total de datos, las columnas y su tipo correspondiente, dice si contien
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [22]:
```
#revisa cuántos valores únicos tiene cada atributo del archivo usando la función nu
df.nunique()
```

Out[22]:
```
Pregnancies                 17
Glucose                    136
BloodPressure               47
SkinThickness               51
Insulin                    186
BMI                        248
DiabetesPedigreeFunction   517
Age                         52
Outcome                      2
dtype: int64
```

Exploración de Datos

In [23]:
```
#utiliza la función describe() para obtener estadística básica. se puede incluir -0
df.describe()
```

Out[23]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Dia
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [24]:
```
#Revisa Valores nulos con funcion isnull().sum()
df.isnull().sum()
```

Out[24]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [25]:
```
#Revisar valores únicos por columna usando función unique(): nombre-columna.unique(
df.Pregnancies.unique(), df.Glucose.unique()
```

Out[25]:
```
(array([ 6,  1,  8,  0,  5,  3, 10,  2,  4,  7,  9, 11, 13, 15, 17, 12, 14]),
 array([148,  85, 183,  89, 137, 116,  78, 115, 197, 125, 110, 168, 139,
        189, 166, 100, 118, 107, 103, 126,  99, 196, 119, 143, 147,  97,
        145, 117, 109, 158,  88,  92, 122, 138, 102,  90, 111, 180, 133,
        106, 171, 159, 146,  71, 105, 101, 176, 150,  73, 187,  84,  44,
        141, 114,  95, 129,  79,   0,  62, 131, 112, 113,  74,  83, 136,
         80, 123,  81, 134, 142, 144,  93, 163, 151,  96, 155,  76, 160,
        124, 162, 132, 120, 173, 170, 128, 108, 154,  57, 156, 153, 188,
        152, 104,  87,  75, 179, 130, 194, 181, 135, 184, 140, 177, 164,
         91, 165,  86, 193, 191, 161, 167,  77, 182, 157, 178,  61,  98,
        127,  82,  72, 172,  94, 175, 195,  68, 186, 198, 121,  67, 174,
        199,  56, 169, 149,  65, 190]))
```

Variables Cuantitativas

Medidas de tendencia central

In [49]:
```
variables = ["Pregnancies", "Glucose", "Outcome"]
stats = diabetes[variables].describe().T
print(stats)
```

```
             count        mean         std  min   25%    50%     75%    max
Pregnancies  768.0    3.845052    3.369578  0.0   1.0    3.0    6.00   17.0
Glucose      768.0  120.894531   31.972618  0.0  99.0  117.0  140.25  199.0
Outcome      768.0    0.348958    0.476951  0.0   0.0    0.0    1.00    1.0
```

In [50]:
```
# Pregnancies
mean_preg = diabetes['Pregnancies'].mean()
median_preg = diabetes['Pregnancies'].median()
mode_preg = diabetes['Pregnancies'].mode()[0]

# Glucose
mean_gluc = diabetes['Glucose'].mean()
median_gluc = diabetes['Glucose'].median()
mode_gluc = diabetes['Glucose'].mode()[0]

# Outcome (proporción)
mean_out = diabetes['Outcome'].mean()

print("Media embarazos:", mean_preg)
print("Mediana embarazos:", median_preg)
print("Moda embarazos:", mode_preg)
print("\nMedia glucosa:", mean_gluc)
print("Mediana glucosa:", median_gluc)
print("Moda glucosa:", mode_gluc)
print("\nProporción de diabéticas:", mean_out)
```

```
Media embarazos: 3.8450520833333335
Mediana embarazos: 3.0
Moda embarazos: 1

Media glucosa: 120.89453125
Mediana glucosa: 117.0
Moda glucosa: 99

Proporción de diabéticas: 0.3489583333333333
```

In [59]:
```
print("""Conclusión:

La media de embarazos es aproximadamente 3.85, la mediana es 3 y la moda es 0.
La media de glucosa es aproximadamente 120.89 mg/dL, la mediana es 117 y la moda es
La media de la variable Outcome es 0.34
""")
```

```
Conclusión:

La media de embarazos es aproximadamente 3.85, la mediana es 3 y la moda es 0.
La media de glucosa es aproximadamente 120.89 mg/dL, la mediana es 117 y la moda es
0.
La media de la variable Outcome es 0.34
```

In [54]:
```
diabetes[diabetes['Glucose'] > 140].head(),
```

Out[54]:
```
(    Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
 0             6      148             72             35        0  33.6
 2             8      183             64              0        0  23.3
 8             2      197             70             45      543  30.5
 11           10      168             74              0        0  38.0
 13            1      189             60             23      846  30.1

     DiabetesPedigreeFunction  Age  Outcome
 0                      0.627   50        1
 2                      0.672   32        1
 8                      0.158   53        1
 11                     0.537   34        1
 13                     0.398   59        1  ,)
```

In [55]:
```
diabetes[(diabetes['Outcome'] == 1) & (diabetes['Pregnancies'] > 3)].head()
```

Out[55]:

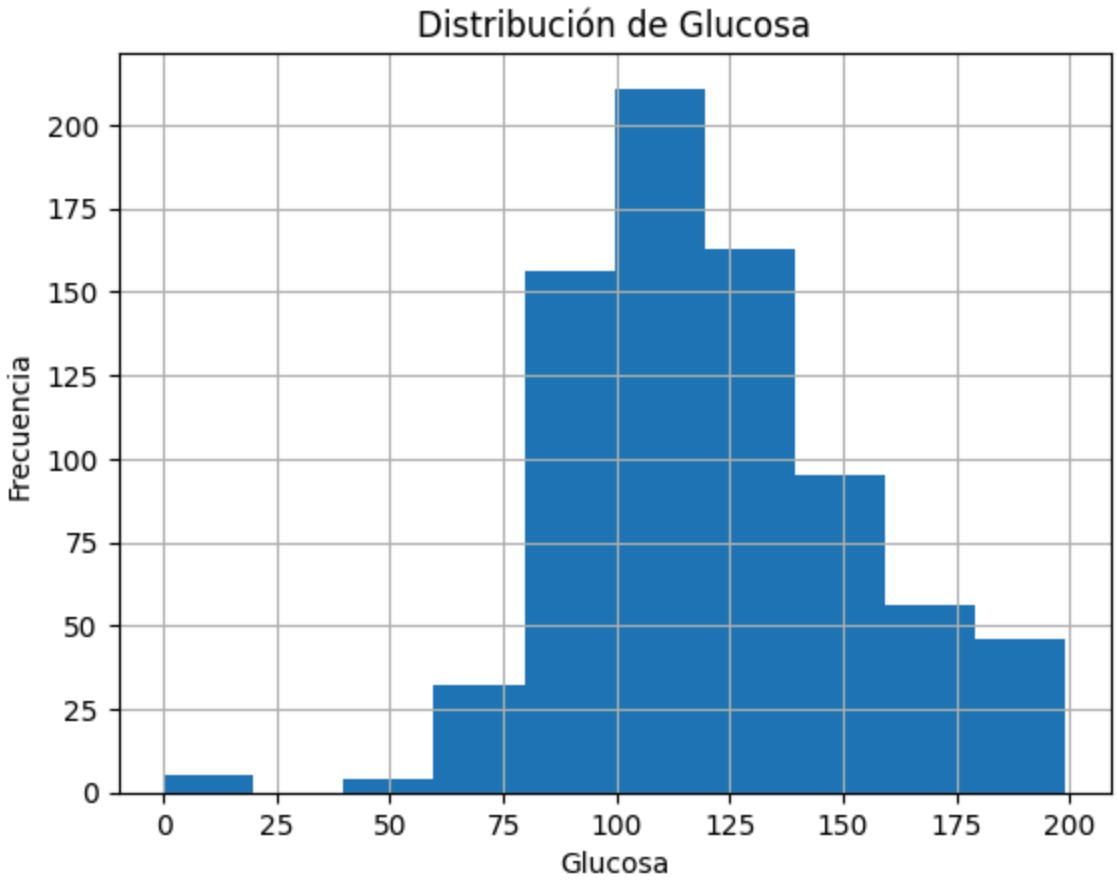
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFur
0	6	148	72	35	0	33.6	
2	8	183	64	0	0	23.3	
9	8	125	96	0	0	0.0	
11	10	168	74	0	0	38.0	
14	5	166	72	19	175	25.8	

In [56]:
```
diabetes.groupby('Outcome')['Glucose'].mean()
```

Out[56]:
```
Outcome
0    109.980000
1    141.257463
Name: Glucose, dtype: float64
```

In [68]:
```
import matplotlib.pyplot as plt

diabetes['Glucose'].hist()
plt.title("Distribución de Glucosa")
plt.xlabel("Glucosa")
plt.ylabel("Frecuencia")
plt.show()
```



Variables Categóricas

In [48]:
```
#Para conteo  de cada valor en una columna, en orden descendente usar función value
# nombreDataframe.columna.value_counts()
# nombreDataframe['columna'].value_counts()
df.Pregnancies.value_counts(), df.Glucose.value_counts()
```

Out[48]:
```
(Pregnancies
 1     135
 0     111
 2     103
 3      75
 4      68
 5      57
 6      50
 7      45
 8      38
 9      28
 10     24
 11     11
 13     10
 12      9
 14      2
 17      1
 15      1
 Name: count, dtype: int64,
 Glucose
 99     17
 100    17
 111    14
 125    14
 129    14
        ..
 56      1
 169     1
 149     1
 65      1
 190     1
 Name: count, Length: 136, dtype: int64)
```

In [34]:
```
#Revisa conteo de varias columnas
df[['Pregnancies', 'Glucose']].value_counts()
```

Out[34]:
```
Pregnancies  Glucose
0            102        5
2            112        5
             100        5
1            97         5
2            108        5
                       ..
13           152        1
             153        1
             158        1
14           100        1
             175        1
Name: count, Length: 536, dtype: int64
```

In [46]:
```
# Crear variable familySize que incluya la suma de las columnas SibSp y Parch
df['Relación Embarazo-Glucosa'] = df['Pregnancies'] + df['Glucose'] + 1

# Mostrar el total por cada tamaño de familia
df['Relación Embarazo-Glucosa'].value_counts()
```

Out[46]:
```
Relación Embarazo-Glucosa
115    20
132    18
103    17
102    15
130    14
       ..
193     1
199     1
68      1
59      1
170     1
Name: count, Length: 140, dtype: int64
```

Consulta

In [37]:
```
# df.iloc[i]: Accede a la fila en la posición i.
# Acceder a la primera fila
df.iloc[0]
```

Out[37]:
```
Pregnancies                 6.000
Glucose                   148.000
BloodPressure              72.000
SkinThickness              35.000
Insulin                     0.000
BMI                        33.600
DiabetesPedigreeFunction    0.627
Age                        50.000
Outcome                     1.000
familySize                155.000
Name: 0, dtype: float64
```

In [38]:
```
# Acceder a las dos primeras filas
df.iloc[[0,1]]
```

Out[38]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148	72	35	0	33.6	0
1	1	85	66	29	0	26.6	0

In [40]:
```
#Seleccionar columnas, indicando entre corchetes [nombreColumna, nombreColumna]
df[['Pregnancies', 'Glucose']]
```

Out[40]:

	Pregnancies	Glucose
0	6	148
1	1	85
2	8	183
3	1	89
4	0	137
...	...	...
763	10	101
764	2	122
765	5	121
766	1	126
767	1	93

768 rows × 2 columns

In [41]:
```
#Selección de filas [indicar dataframe[columna] operador valor]
df[df['Pregnancies'] == 1].head()
```

Out[41]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFur
1	1	85	66	29	0	26.6	
3	1	89	66	23	94	28.1	
13	1	189	60	23	846	30.1	
18	1	103	30	38	83	43.3	
19	1	115	70	30	96	34.6	

In [43]:
```
#ordenar usando funcion sort_values(by=atributo, ascending=True/false)
df.sort_values(by='Pregnancies', ascending=True).head(), df.sort_values(by='Glucose
```

Out[43]:
```
(     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
 16             0      118             84             47      230  45.8
 736            0      126             86             27      120  27.4
 713            0      134             58             20      291  26.4
 727            0      141             84             26        0  32.4
 681            0      162             76             36        0  49.6

      DiabetesPedigreeFunction  Age  Outcome  familySize
 16                      0.551   31        1         119
 736                     0.515   21        0         127
 713                     0.352   21        0         135
 727                     0.433   22        0         142
 681                     0.364   26        1         163  ,
      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
 661            1      199             76             43        0  42.9
 561            0      198             66             32      274  41.3
 579            2      197             70             99        0  34.7
 228            4      197             70             39      744  36.7
 8              2      197             70             45      543  30.5

      DiabetesPedigreeFunction  Age  Outcome  familySize
 661                     1.394   22        1         201
 561                     0.502   28        1         199
 579                     0.575   62        1         200
 228                     2.329   31        0         202
 8                       0.158   53        1         200  )
```

In [44]:
```
#Agrupar por un atributo y calcular función de agregación utilizando groupby(atribu
df.groupby('Pregnancies')['Glucose'].mean()
```

Out[44]:
```
Pregnancies
0     123.000000
1     112.748148
2     110.796117
3     123.586667
4     125.117647
5     118.859649
6     120.800000
7     136.444444
8     131.736842
9     131.392857
10    120.916667
11    126.454545
12    113.555556
13    125.500000
14    137.500000
15    136.000000
17    163.000000
Name: Glucose, dtype: float64
```

Crea un subconjunto de **titanic** para el costo mayor a 500

In [45]:
```
# Glucosa arriba de 100
df_boletos_caros = df[df['Glucose'] > 100]
df_boletos_caros.head()
```

Out[45]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148	72	35	0	33.6	0
2	8	183	64	0	0	23.3	0
4	0	137	40	35	168	43.1	2
5	5	116	74	0	0	25.6	0
7	10	115	0	0	0	35.3	0

In [ ]: