Comprensión de los Datos

A01384654

Angela Monserrat Hernandez Lomas

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Pregnancies Number of Pregnancies: Numérica Discreta
Glucose Plasma Glucose Concentration: Numérica Continua
BloodPressure Diastolic Blood Pressure (mm Hg): Numérica Continua
SkinThickness Triceps Skin Fold Thickness (mm): Numérica Continua
Insulin 2-Hour Serum Insulin (mu U/ml): Numérica Continua
BMI Body Mass Index (weight in kg / height in m^2): Numérica Continua
DiabetesPedigreeFunction Diabetes Pedigree Function: Numérica Continua
Age Age (years): Numérica Discreta
Outcome Class Variable (0 = No Diabetes; 1 = Diabetes): Categórica

In [2]:
```
#lee archivo csv
df = pd.read_csv('diabetes.csv')
```

Descripción de Variables

In [3]:
```
#Usa función shape para revisar el total de renglones y columnas
df.shape
```

Out[3]:
```
(768, 9)
```

In [4]:
```
#Revisa los primeros 5 renglones del dataset usando la función head()
df.head(4)
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabetes
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	

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

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabet
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 [6]:
```
#Revisa la información mas completa del conjunto de datos usando la fu
#Muestra el total de datos, las columnas y su tipo correspondiente, di
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
```

Ninguna variable contiene valores nulos

In [7]:
```
#revisa cuántos valores únicos tiene cada atributo del archivo usando
df.nunique()
```

Out[7]:
```
Pregnancies                 17
Glucose                    136
BloodPressure               47
SkinThickness               51
Insulin                    186
BMI                        248
DiabetesPedigreeFunction   517
Age                         52
Outcome                      2
dtype: int64
```

Exploración de Datos

In [8]:
```
#utiliza la función describe() para obtener estadística básica. se pue
df.describe()
```

Out[8]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	
count	768.000000	768.000000	768.000000	768.000000	768.000000	76
mean	3.845052	120.894531	69.105469	20.536458	79.799479	
std	3.369578	31.972618	19.355807	15.952218	115.244002	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	3
75%	6.000000	140.250000	80.000000	32.000000	127.250000	3
max	17.000000	199.000000	122.000000	99.000000	846.000000	

In [9]:
```
#Revisa Valores nulos con funcion isnull().sum()
df.isnull().sum()
```

Out[9]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [10]:
```
#Revisar valores únicos por columna usando función unique(): nombre-co
df.Pregnancies.unique()
```

Out[10]:
```
array([ 6,  1,  8,  0,  5,  3, 10,  2,  4,  7,  9, 11, 13, 15, 17, 1
2, 14])
```

In [11]:
```
df.Glucose.unique()
```

Out[11]:
```
array([148,  85, 183,  89, 137, 116,  78, 115, 197, 125, 110, 168, 13
9,
       189, 166, 100, 118, 107, 103, 126,  99, 196, 119, 143, 147,  9
7,
       145, 117, 109, 158,  88,  92, 122, 138, 102,  90, 111, 180, 13
3,
       106, 171, 159, 146,  71, 105, 101, 176, 150,  73, 187,  84,  4
4,
       141, 114,  95, 129,  79,   0,  62, 131, 112, 113,  74,  83, 13
6,
        80, 123,  81, 134, 142, 144,  93, 163, 151,  96, 155,  76, 16
0,
       124, 162, 132, 120, 173, 170, 128, 108, 154,  57, 156, 153, 18
8,
       152, 104,  87,  75, 179, 130, 194, 181, 135, 184, 140, 177, 16
4,
        91, 165,  86, 193, 191, 161, 167,  77, 182, 157, 178,  61,  9
8,
       127,  82,  72, 172,  94, 175, 195,  68, 186, 198, 121,  67, 17
4,
       199,  56, 169, 149,  65, 190])
```

In [12]:
```
df.BloodPressure.unique()
```

Out[12]:
```
array([ 72,  66,  64,  40,  74,  50,   0,  70,  96,  92,  80,  60,  8
4,
        30,  88,  90,  94,  76,  82,  75,  58,  78,  68, 110,  56,  6
2,
        85,  86,  48,  44,  65, 108,  55, 122,  54,  52,  98, 104,  9
5,
        46, 102, 100,  61,  24,  38, 106, 114])
```

In [13]:
```
df.SkinThickness.unique()
```

Out[13]:
```
array([35, 29,  0, 23, 32, 45, 19, 47, 38, 30, 41, 33, 26, 15, 36, 1
1, 31,
       37, 42, 25, 18, 24, 39, 27, 21, 34, 10, 60, 13, 20, 22, 28, 5
4, 40,
       51, 56, 14, 17, 50, 44, 12, 46, 16,  7, 52, 43, 48,  8, 49, 6
3, 99])
```

In [14]:
```
df.Insulin.unique()
```

Out[14]:
```
array([  0,  94, 168,  88, 543, 846, 175, 230,  83,  96, 235, 146, 11
5,
       140, 110, 245,  54, 192, 207,  70, 240,  82,  36,  23, 300, 34
2,
       304, 142, 128,  38, 100,  90, 270,  71, 125, 176,  48,  64, 22
8,
        76, 220,  40, 152,  18, 135, 495,  37,  51,  99, 145, 225,  4
9,
        50,  92, 325,  63, 284, 119, 204, 155, 485,  53, 114, 105, 28
5,
       156,  78, 130,  55,  58, 160, 210, 318,  44, 190, 280,  87, 27
1,
       129, 120, 478,  56,  32, 744, 370,  45, 194, 680, 402, 258, 37
5,
       150,  67,  57, 116, 278, 122, 545,  75,  74, 182, 360, 215, 18
4,
        42, 132, 148, 180, 205,  85, 231,  29,  68,  52, 255, 171,  7
3,
       108,  43, 167, 249, 293,  66, 465,  89, 158,  84,  72,  59,  8
1,
       196, 415, 275, 165, 579, 310,  61, 474, 170, 277,  60,  14,  9
5,
       237, 191, 328, 250, 480, 265, 193,  79,  86, 326, 188, 106,  6
5,
       166, 274,  77, 126, 330, 600, 185,  25,  41, 272, 321, 144,  1
5,
       183,  91,  46, 440, 159, 540, 200, 335, 387,  22, 291, 392, 17
8,
       127, 510,  16, 112])
```

In [15]:
```
df.Outcome.unique()
```

Out[15]:
```
array([1, 0])
```

In [16]:
```
df.DiabetesPedigreeFunction.unique()
```

Out[16]:
```
array([0.627, 0.351, 0.672, 0.167, 2.288, 0.201, 0.248, 0.134, 0.158,
       0.232, 0.191, 0.537, 1.441, 0.398, 0.587, 0.484, 0.551, 0.254,
       0.183, 0.529, 0.704, 0.388, 0.451, 0.263, 0.205, 0.257, 0.487,
       0.245, 0.337, 0.546, 0.851, 0.267, 0.188, 0.512, 0.966, 0.42 ,
       0.665, 0.503, 1.39 , 0.271, 0.696, 0.235, 0.721, 0.294, 1.893,
       0.564, 0.586, 0.344, 0.305, 0.491, 0.526, 0.342, 0.467, 0.718,
       0.962, 1.781, 0.173, 0.304, 0.27 , 0.699, 0.258, 0.203, 0.855,
       0.845, 0.334, 0.189, 0.867, 0.411, 0.583, 0.231, 0.396, 0.14 ,
       0.391, 0.37 , 0.307, 0.102, 0.767, 0.237, 0.227, 0.698, 0.178,
       0.324, 0.153, 0.165, 0.443, 0.261, 0.277, 0.761, 0.255, 0.13 ,
       0.323, 0.356, 0.325, 1.222, 0.179, 0.262, 0.283, 0.93 , 0.801,
       0.207, 0.287, 0.336, 0.247, 0.199, 0.543, 0.192, 0.588, 0.539,
       0.22 , 0.654, 0.223, 0.759, 0.26 , 0.404, 0.186, 0.278, 0.496,
       0.452, 0.403, 0.741, 0.361, 1.114, 0.457, 0.647, 0.088, 0.597,
       0.532, 0.703, 0.159, 0.268, 0.286, 0.318, 0.272, 0.572, 0.096,
       1.4  , 0.218, 0.085, 0.399, 0.432, 1.189, 0.687, 0.137, 0.637,
```

```
       0.833, 0.229, 0.817, 0.204, 0.368, 0.743, 0.722, 0.256, 0.709,
       0.471, 0.495, 0.18 , 0.542, 0.773, 0.678, 0.719, 0.382, 0.319,
       0.19 , 0.956, 0.084, 0.725, 0.299, 0.244, 0.745, 0.615, 1.321,
       0.64 , 0.142, 0.374, 0.383, 0.578, 0.136, 0.395, 0.187, 0.905,
       0.15 , 0.874, 0.236, 0.787, 0.407, 0.605, 0.151, 0.289, 0.355,
       0.29 , 0.375, 0.164, 0.431, 0.742, 0.514, 0.464, 1.224, 1.072,
       0.805, 0.209, 0.666, 0.101, 0.198, 0.652, 2.329, 0.089, 0.645,
       0.238, 0.394, 0.293, 0.479, 0.686, 0.831, 0.582, 0.446, 0.402,
       1.318, 0.329, 1.213, 0.427, 0.282, 0.143, 0.38 , 0.284, 0.249,
       0.926, 0.557, 0.092, 0.655, 1.353, 0.612, 0.2  , 0.226, 0.997,
       0.933, 1.101, 0.078, 0.24 , 1.136, 0.128, 0.422, 0.251, 0.677,
       0.296, 0.454, 0.744, 0.881, 0.28 , 0.259, 0.619, 0.808, 0.34 ,
       0.434, 0.757, 0.613, 0.692, 0.52 , 0.412, 0.84 , 0.839, 0.156,
       0.215, 0.326, 1.391, 0.875, 0.313, 0.433, 0.626, 1.127, 0.315,
       0.345, 0.129, 0.527, 0.197, 0.731, 0.148, 0.123, 0.127, 0.122,
       1.476, 0.166, 0.932, 0.343, 0.893, 0.331, 0.472, 0.673, 0.389,
       0.485, 0.349, 0.279, 0.346, 0.252, 0.243, 0.58 , 0.559, 0.302,
       0.569, 0.378, 0.385, 0.499, 0.306, 0.234, 2.137, 1.731, 0.545,
       0.225, 0.816, 0.528, 0.509, 1.021, 0.821, 0.947, 1.268, 0.221,
       0.66 , 0.239, 0.949, 0.444, 0.463, 0.803, 1.6  , 0.944, 0.196,
       0.241, 0.161, 0.135, 0.376, 1.191, 0.702, 0.674, 1.076, 0.534,
       1.095, 0.554, 0.624, 0.219, 0.507, 0.561, 0.421, 0.516, 0.264,
       0.328, 0.233, 0.108, 1.138, 0.147, 0.727, 0.435, 0.497, 0.23 ,
       0.955, 2.42 , 0.658, 0.33 , 0.51 , 0.285, 0.415, 0.381, 0.832,
       0.498, 0.212, 0.364, 1.001, 0.46 , 0.733, 0.416, 0.705, 1.022,
       0.269, 0.6  , 0.571, 0.607, 0.17 , 0.21 , 0.126, 0.711, 0.466,
       0.162, 0.419, 0.63 , 0.365, 0.536, 1.159, 0.629, 0.292, 0.145,
       1.144, 0.174, 0.547, 0.163, 0.738, 0.314, 0.968, 0.409, 0.297,
       0.525, 0.154, 0.771, 0.107, 0.493, 0.717, 0.917, 0.501, 1.251,
       0.735, 0.804, 0.661, 0.549, 0.825, 0.423, 1.034, 0.16 , 0.341,
       0.68 , 0.591, 0.3  , 0.121, 0.502, 0.401, 0.601, 0.748, 0.338,
       0.43 , 0.892, 0.813, 0.693, 0.575, 0.371, 0.206, 0.417, 1.154,
       0.925, 0.175, 1.699, 0.682, 0.194, 0.4  , 0.1  , 1.258, 0.482,
       0.138, 0.593, 0.878, 0.157, 1.282, 0.141, 0.246, 1.698, 1.461,
       0.347, 0.362, 0.393, 0.144, 0.732, 0.115, 0.465, 0.649, 0.871,
       0.149, 0.695, 0.303, 0.61 , 0.73 , 0.447, 0.455, 0.133, 0.155,
       1.162, 1.292, 0.182, 1.394, 0.217, 0.631, 0.88 , 0.614, 0.332,
       0.366, 0.181, 0.828, 0.335, 0.856, 0.886, 0.439, 0.253, 0.598,
       0.904, 0.483, 0.565, 0.118, 0.177, 0.176, 0.295, 0.441, 0.352,
       0.826, 0.97 , 0.595, 0.317, 0.265, 0.646, 0.426, 0.56 , 0.515,
       0.453, 0.785, 0.734, 1.174, 0.488, 0.358, 1.096, 0.408, 1.182,
       0.222, 1.057, 0.766, 0.171])
```

In [17]:
```
df.Age.unique()
```

Out[17]:
```
array([50, 31, 32, 21, 33, 30, 26, 29, 53, 54, 34, 57, 59, 51, 27, 4
1, 43,
       22, 38, 60, 28, 45, 35, 46, 56, 37, 48, 40, 25, 24, 58, 42, 4
4, 39,
       36, 23, 61, 69, 62, 55, 65, 47, 52, 66, 49, 63, 67, 72, 81, 6
4, 70,
       68])
```

Variables Cuantitativas

Medidas de tendencia central

In [18]:
```
#Se puede obtener la media, mediana y moda para
mean_glucose = df['Glucose'].mean()
median_glucose =df['Glucose'].median()
mode_glucose = df['Glucose'].mode()
print("Mean_glucose:",mean_glucose)
print("Median_glucose:",median_glucose)
print("Mode_glucose:",mode_glucose)

mean_BMI = df['BMI'].mean()
median_BMI =df['BMI'].median()
mode_BMI = df['BMI'].mode()
print("Mean_BMI:",mean_BMI)
print("Median_BMI:",median_BMI)
print("Mode_BMI:",mode_BMI)
```

```
Mean_glucose: 120.89453125
Median_glucose: 117.0
Mode_glucose: 0     99
1    100
Name: Glucose, dtype: int64
Mean_BMI: 31.992578124999998
Median_BMI: 32.0
Mode_BMI: 0    32.0
Name: BMI, dtype: float64
```

Conclusiones: La glucosa promedio fue 120.89 La glucosa al centro (mediana) fue 117 La glucosa más repetida fue de 99 y 100

El BMI promedio fue 31.99 El BMI al centro (mediana) fue 32 El BMI más repetido fue de 32

Variables Categóricas

In [19]:
```
#Para conteo  de cada valor en una columna, en orden descendente usar
# nombreDataframe.columna.value_counts()
# nombreDataframe['columna'].value_counts()
df.Outcome.value_counts()
```

Out[19]:
```
Outcome
0    500
1    268
Name: count, dtype: int64
```

In [20]:
```
# Revisa conteo de Outcome
print("Conteo Outcome:")
print(df['Outcome'].value_counts())
```

```
Conteo Outcome:
Outcome
0    500
1    268
Name: count, dtype: int64
```

In [21]:
```
# Crear columna para clasificar glucosa
df['GlucoseLevel'] = pd.cut(df['Glucose'], bins=[0, 99, 125, 200], lab

# Mostrar total por cada nivel
print(df['GlucoseLevel'].value_counts())
```

```
GlucoseLevel
Alto      297
Normal    274
Bajo      192
Name: count, dtype: int64
```

In [22]:
```
# Crear columna que combine BMI y Outcome
df['BMI_Category'] = pd.cut(df['BMI'], bins=[0, 18.5, 25, 30, 100], la

# Ver distribución
print(df['BMI_Category'].value_counts())
```

```
BMI_Category
Obeso        465
Sobrepeso    180
Normal       108
Bajo           4
Name: count, dtype: int64
```

In [23]:
```
df
```

Out[23]:

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

768 rows × 11 columns

Consulta

In [24]:
```
# df.iloc[i]: Accede a la fila en la posición i.
# Acceder a la primera fila
df.iloc[0]
```

Out[24]:
```
Pregnancies                 6
Glucose                   148
BloodPressure              72
SkinThickness              35
Insulin                     0
BMI                      33.6
DiabetesPedigreeFunction    0.627
Age                        50
Outcome                     1
GlucoseLevel             Alto
BMI_Category            Obeso
Name: 0, dtype: object
```

In [25]:
```
# Acceder a las dos primeras filas
df.iloc[0:2]
```

Out[25]:

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

In [26]:
```
#Seleccionar columnas, indicando entre corchetes [nombreColumna, nombr
df[['Glucose', 'BMI', 'Outcome']].head(10)  # Muestra las primeras 10
```

Out[26]:

	Glucose	BMI	Outcome
0	148	33.6	1
1	85	26.6	0
2	183	23.3	1
3	89	28.1	0
4	137	43.1	1
5	116	25.6	0
6	78	31.0	1
7	115	35.3	0
8	197	30.5	1
9	125	0.0	1

In [27]:
```
#Selección de filas [indicar dataframe[columna] operador valor]
# Outcome = 1 (personas con diabetes)
df[df['Outcome'] == 1][['Glucose', 'BMI', 'Outcome']]
```

Out[27]:

	Glucose	BMI	Outcome
0	148	33.6	1
2	183	23.3	1
4	137	43.1	1
6	78	31.0	1
8	197	30.5	1
...	...	...	...
755	128	36.5	1
757	123	36.3	1
759	190	35.5	1
761	170	44.0	1
766	126	30.1	1

268 rows × 3 columns

In [28]:
```
#ordenar usando funcion sort_values(by=atributo, ascending=True/false)
df.sort_values(by='Glucose', ascending=True)[['Glucose', 'BMI', 'Outco
```

Out[28]:

	Glucose	BMI	Outcome
75	0	24.7	0
502	0	39.0	1
349	0	41.0	1
342	0	32.0	0
182	0	27.7	0

Monstrara las 5 personas con nivel de glucosa más bajo.

In [29]:
```
df.sort_values(by='BMI', ascending=False)[['Glucose', 'BMI', 'Outcome'
```

Out[29]:

	Glucose	BMI	Outcome
177	129	67.1	1
445	180	59.4	1
673	123	57.3	0
125	88	55.0	1
120	162	53.2	1

Mostrara las 5 personas con BMI más alto.

In [30]:
```
#Agrupar por un atributo y calcular función de agregación utilizando g

df.groupby('Outcome')[['Glucose', 'BMI']].max()
```

Out[30]:

Outcome	Glucose	BMI
0	197	57.3
1	199	67.1

Nos dira el máximo valor de Glucose y BMI por Outcome

In [31]:
```
# Crear un subconjunto con Glucose > 140
df_high_glucose = df[df['Glucose'] > 140]
df_high_glucose[['Glucose', 'BMI', 'Outcome']].head(10)
```

Out[31]:

	Glucose	BMI	Outcome
0	148	33.6	1
2	183	23.3	1
8	197	30.5	1
11	168	38.0	1
13	189	30.1	1
14	166	25.8	1
22	196	39.8	1
24	143	36.6	1
26	147	39.4	1
28	145	22.2	0

Nos mostrara las primeras 10 personas que cuenten con glucosa mayor a 140, su BMI y si tienen o no tienen diabetes

In [32]:
```
# Crear un subconjunto con BMI > 30
df_high_bmi = df[df['BMI'] > 30]
df_high_bmi[['Glucose', 'BMI', 'Outcome']].head(10)
```

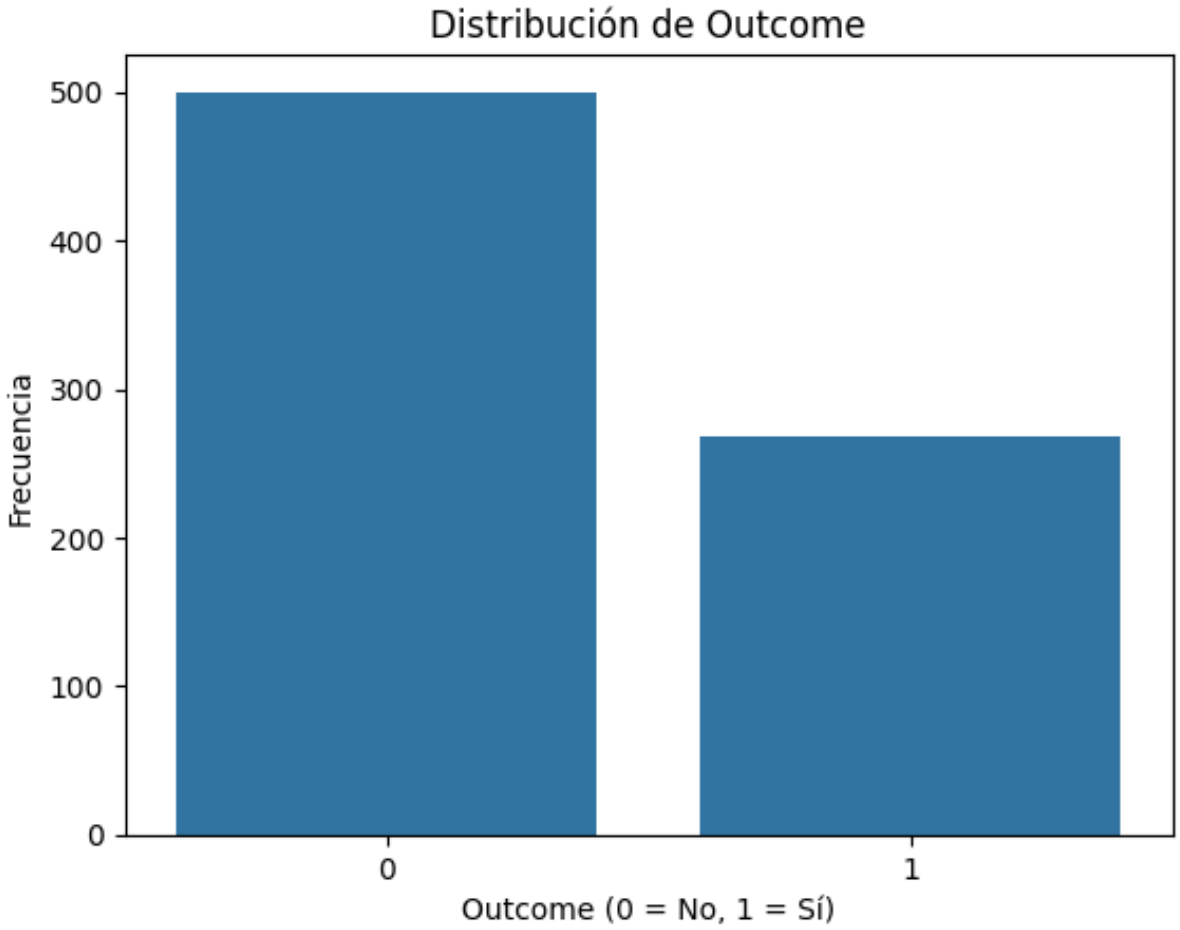
Out[32]:

	Glucose	BMI	Outcome
0	148	33.6	1
4	137	43.1	1
6	78	31.0	1
7	115	35.3	0
8	197	30.5	1
10	110	37.6	0
11	168	38.0	1
13	189	30.1	1
16	118	45.8	1
18	103	43.3	0

Nos mostrara las primeras 10 personas con BMI mayor a 30, su glucosa y si tienen o no diabetes

Visualización y Análisis de Datos

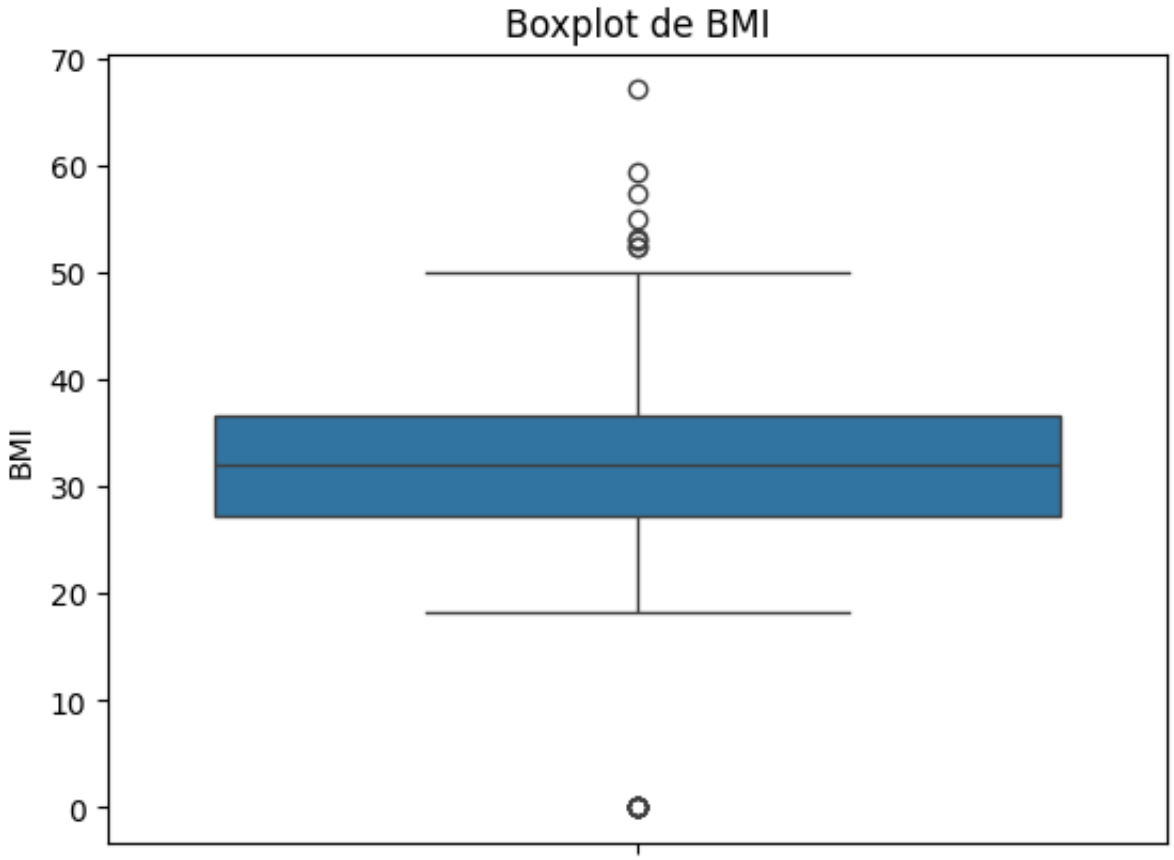
In [47]:
```
sns.countplot(x='Outcome', data=df)
plt.title('Distribución de Outcome')
plt.xlabel('Outcome (0 = No, 1 = Sí)')
plt.ylabel('Frecuencia')
plt.show()
```

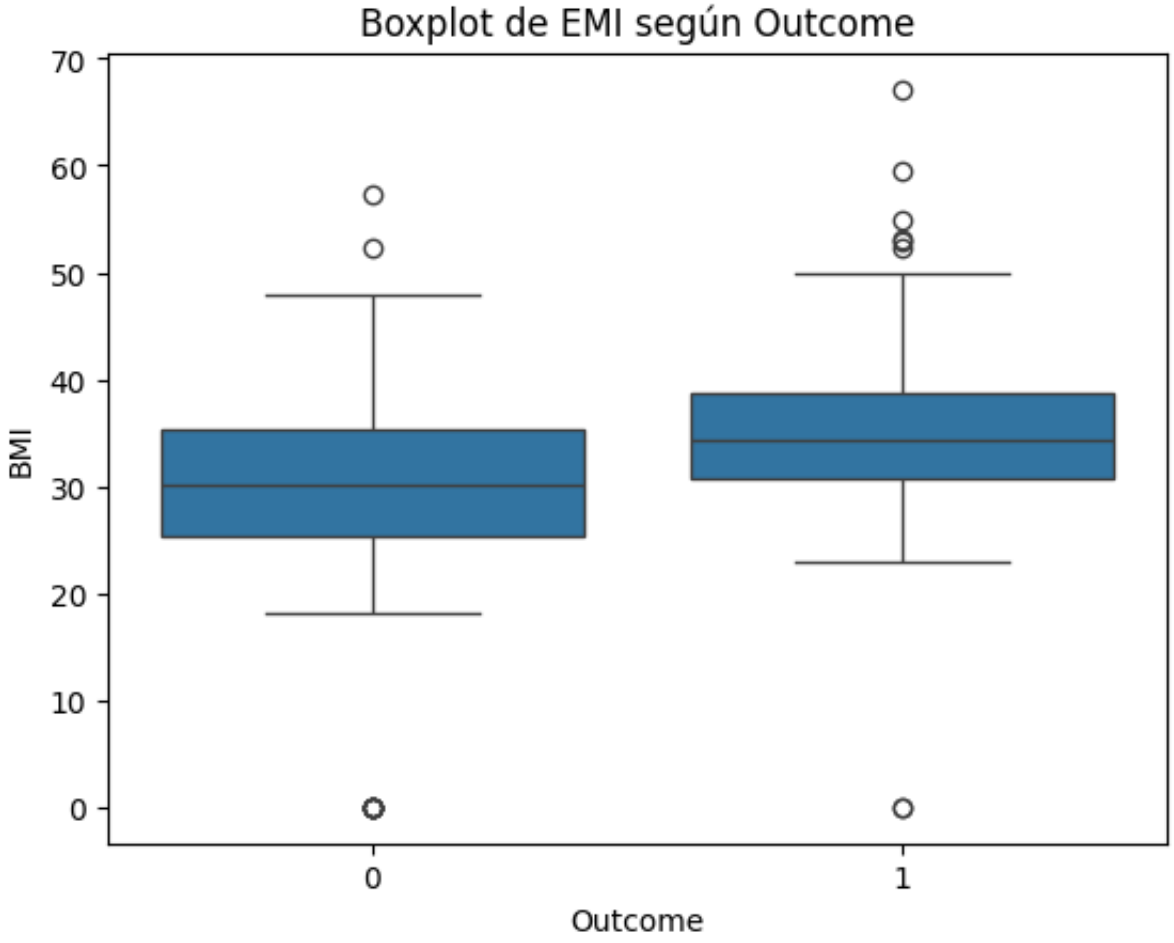


hay más del doble de casos sin diabetes que con diabetes

BMI

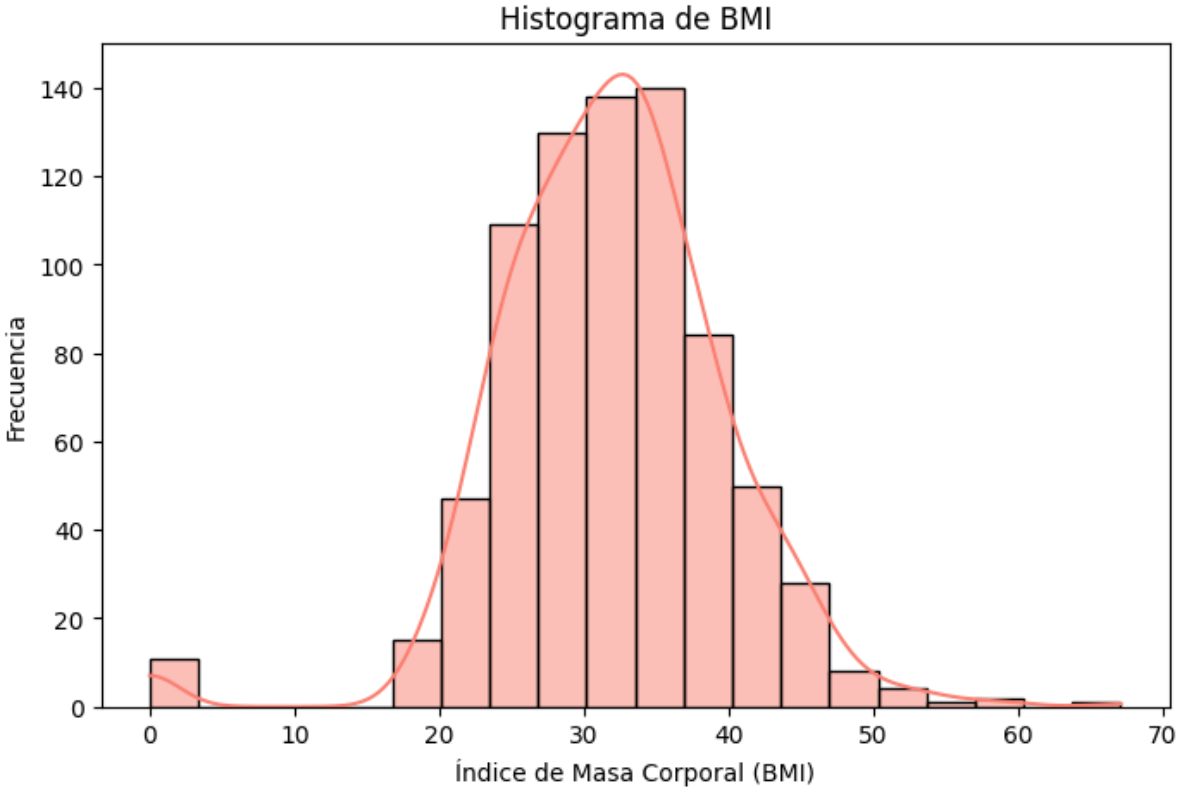
In [54]:
```
df['BMI'] = df['BMI'].fillna(df['BMI'].median())

sns.boxplot(y='BMI', data=df)
plt.title('Boxplot de BMI')
plt.show()

sns.boxplot(x='Outcome', y='BMI', data=df)
plt.title('Boxplot de EMI según Outcome')
plt.xlabel('Outcome')
plt.ylabel('BMI')
plt.show()
```


Boxplot de BMI
BMI
70
60
50
40
30
20
10
0


Boxplot de EMI según Outcome
BMI
70
60
50
40
30
20
10
0
0
1
Outcome

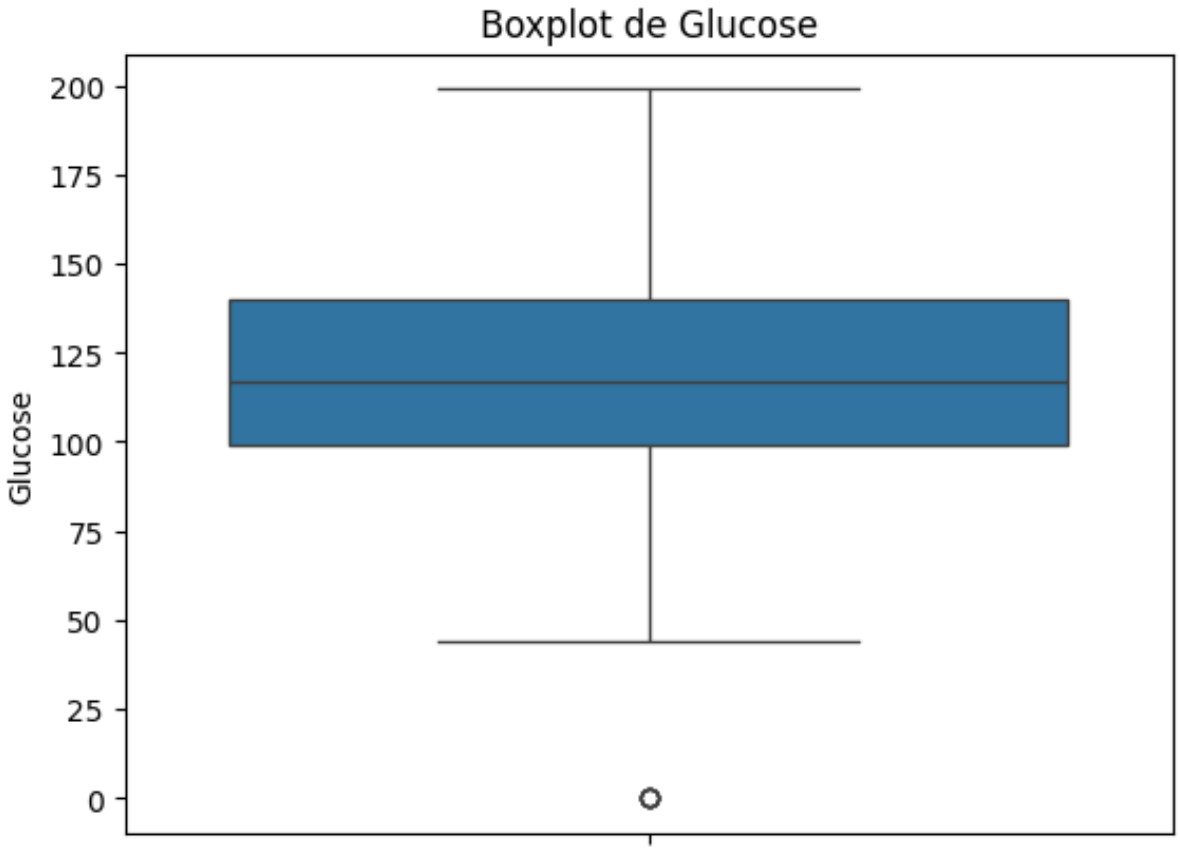
La mayoría de los valores están entre 25 y 35 es decir, en rangos de sobrepeso u obesidad. Se puede decir que la población tiende a tener sobrepeso, lo cual puede influir en la prevalencia de diabetes. Ademas se pueden visualizar valores atipicos por encima de 50

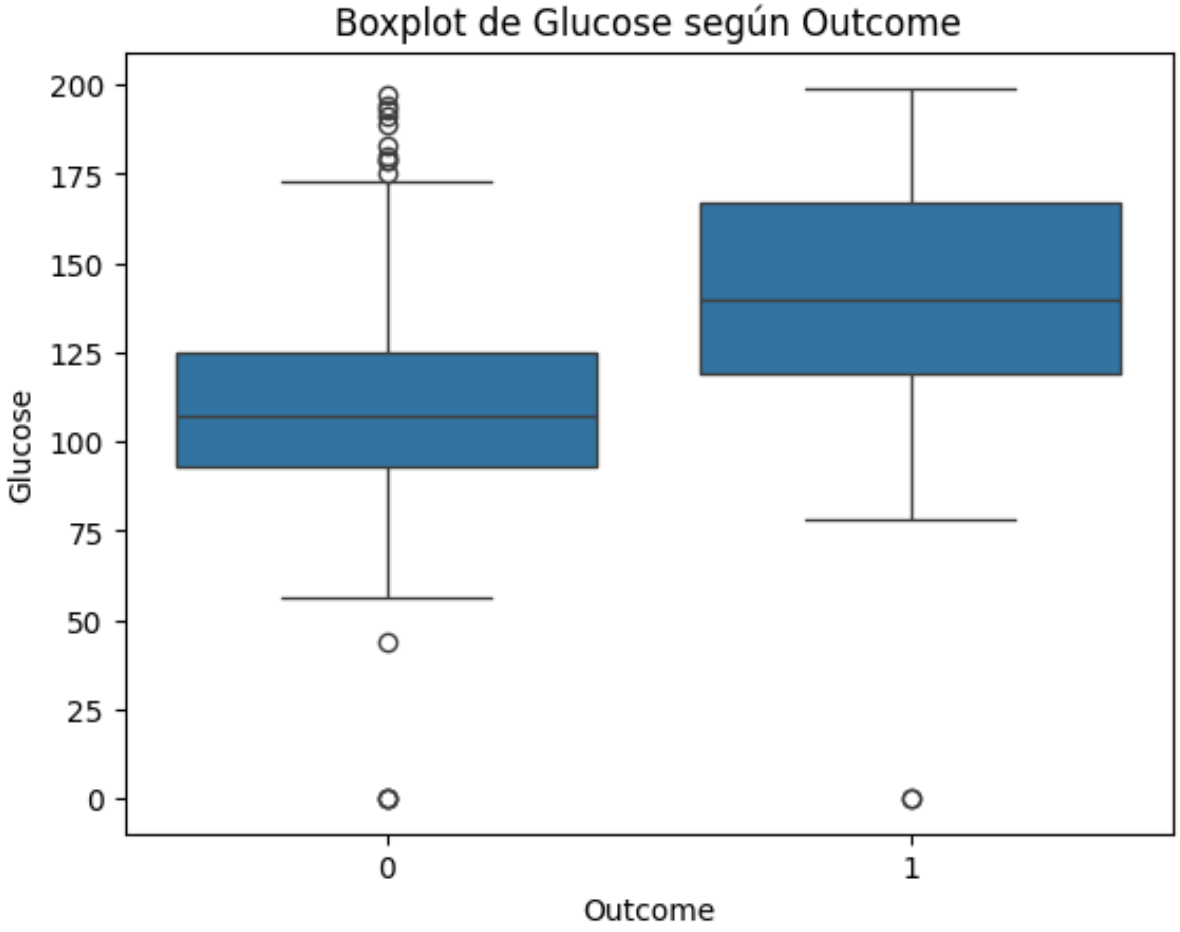
In [43]:
```
plt.figure(figsize=(8,5))
sns.histplot(df['BMI'], bins=20, kde=True, color='salmon')
plt.title('Histograma de BMI')
plt.xlabel('Índice de Masa Corporal (BMI)')
plt.ylabel('Frecuencia')
plt.show()
```



Glucose

In [55]:
```
df['Glucose'] = df['Glucose'].fillna(df['Glucose'].median())

sns.boxplot(y='Glucose', data=df)
plt.title('Boxplot de Glucose')
plt.show()

sns.boxplot(x='Outcome', y='Glucose', data=df)
plt.title('Boxplot de Glucose según Outcome')
plt.xlabel('Outcome')
plt.ylabel('Glucose')
plt.show()
```

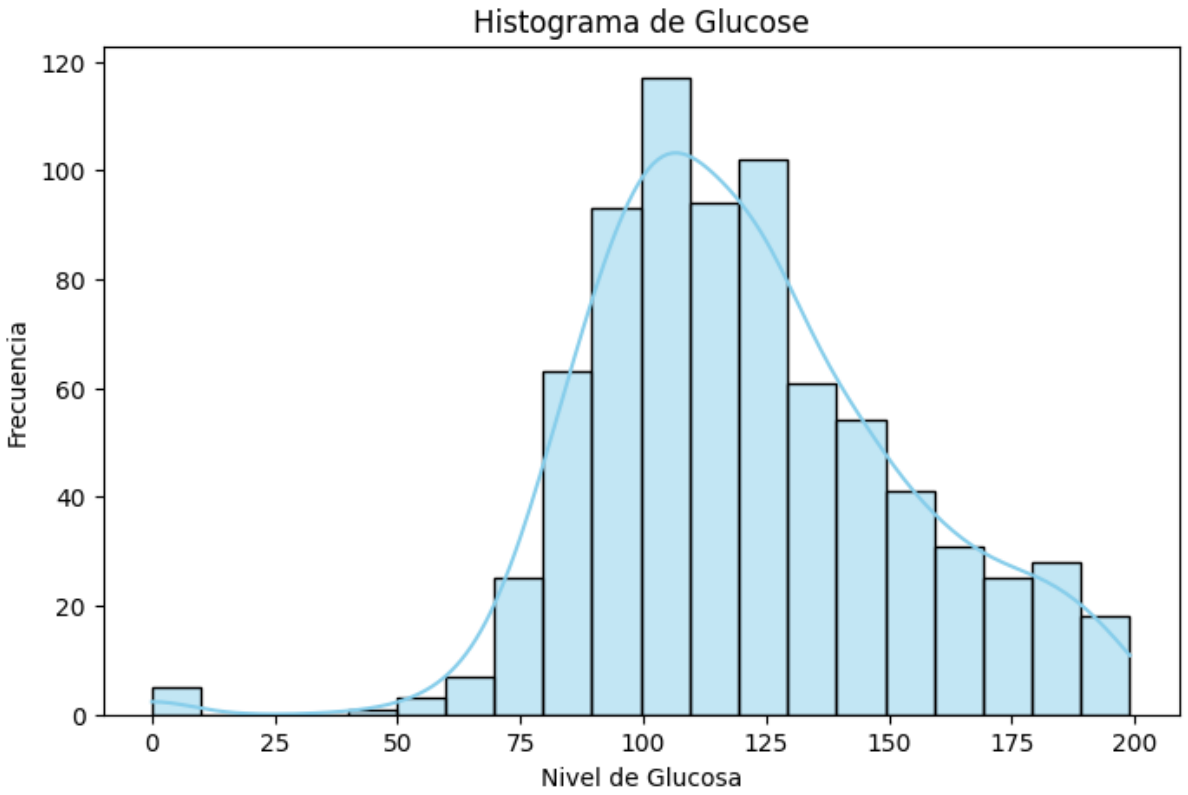




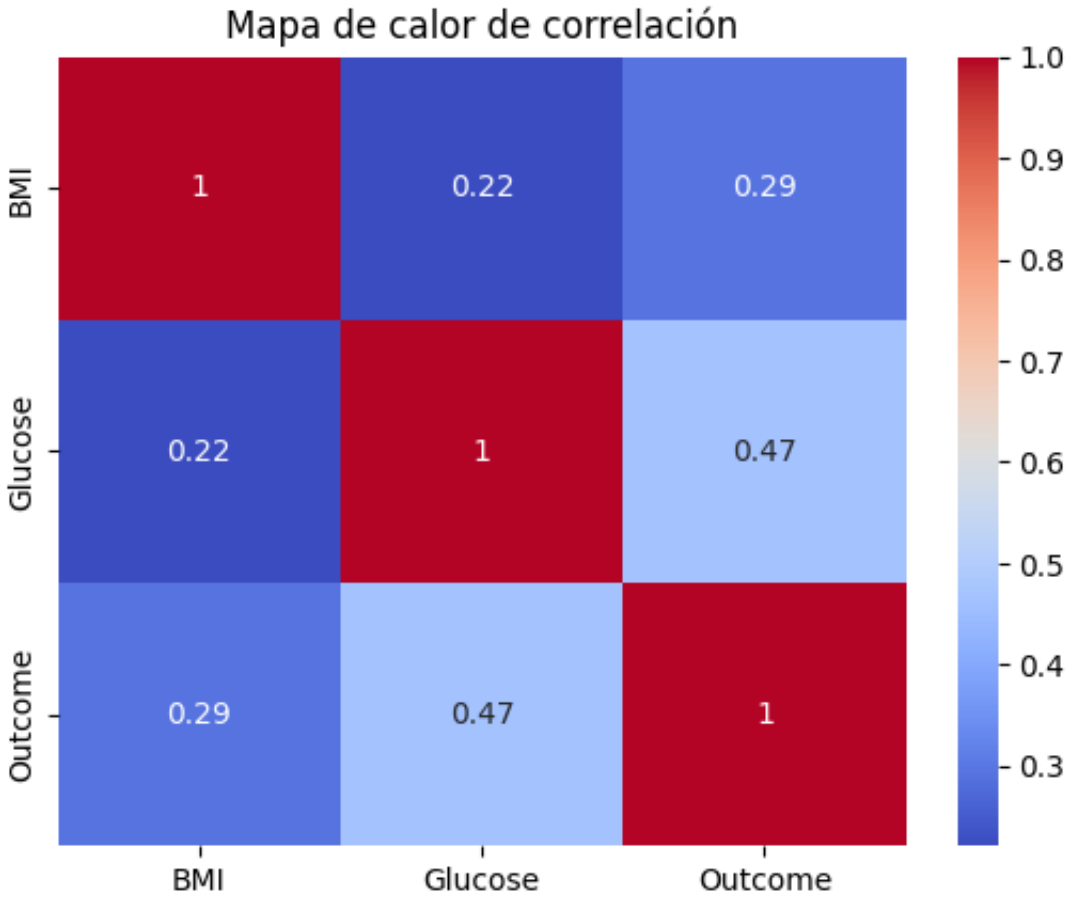
La mediana está alrededor de 120 mg/dL. Se pueden observar algunos ceros que

quiza son errores. La distribución está sesgada a la derecha porque hay mas valores hacia arriba lo que puede decirnos que hay más individuos con niveles normales y pocos con niveles muy altos. Se puede inferir que niveles altos de glucosa se asocian con mayor probabilidad de diabetes

In [45]:
```
plt.figure(figsize=(8,5))
sns.histplot(df['Glucose'], bins=20, kde=True, color='skyblue')
plt.title('Histograma de Glucose')
plt.xlabel('Nivel de Glucosa')
plt.ylabel('Frecuencia')
plt.show()
```

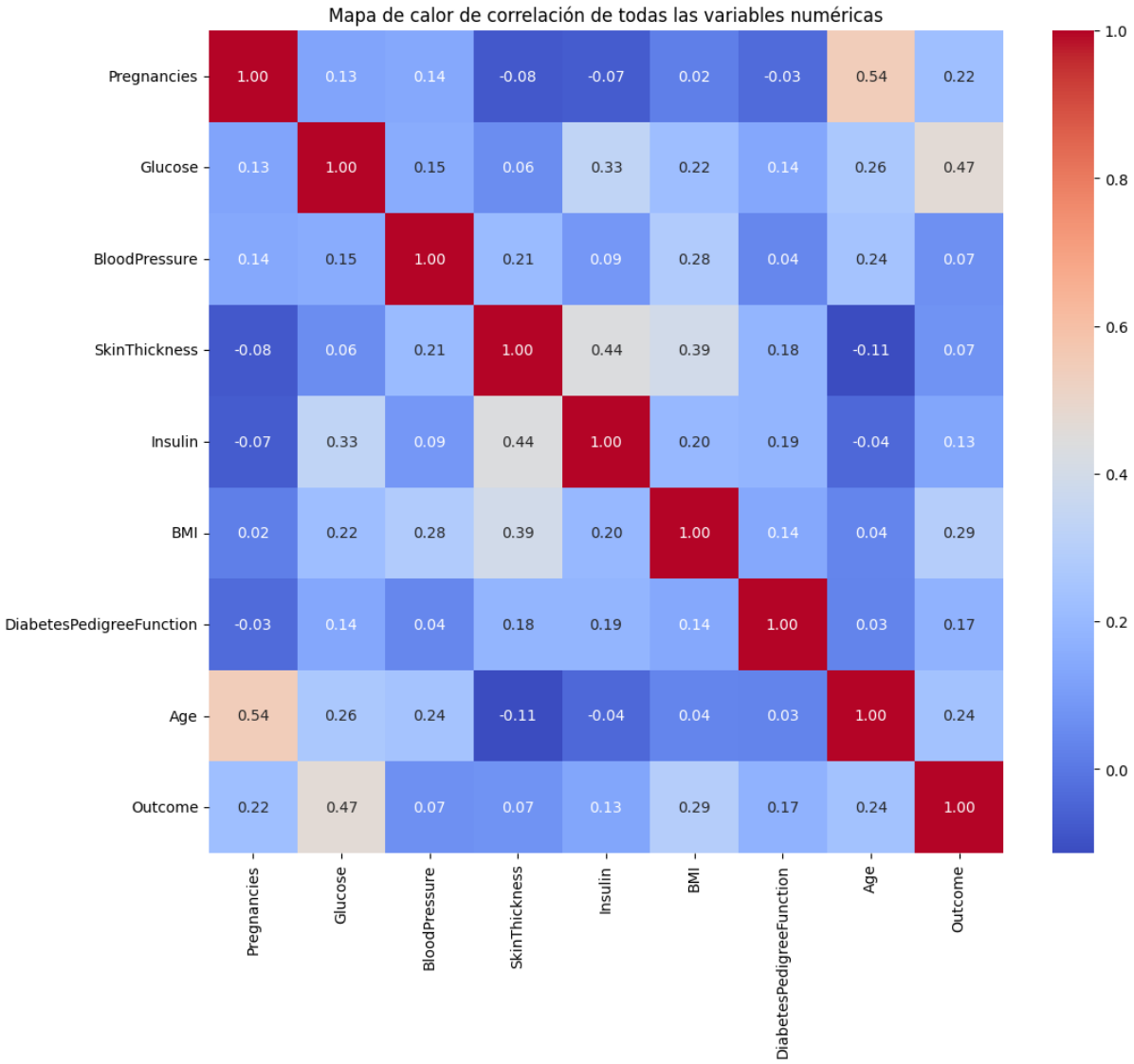


In [38]:
```
corr = df[['BMI', 'Glucose', 'Outcome']].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Mapa de calor de correlación')
plt.show()
```



La glucosa es la variable más relevante para diabetes, seguida de BMI. Se puede ver que BMI y Glucose están algo correlacionadas pero no tan significativamente

In [41]:
```
df_numericas = df.select_dtypes(include=['int64', 'float64'])
corr_total = df_numericas.corr()

# Heatmap de todas las variables
plt.figure(figsize=(12,10))
sns.heatmap(corr_total, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Mapa de calor de correlación de todas las variables numéric
plt.show()
```



Se puede observar con una relacion de 0.47 que a mayor glucosa mas probabilidad hay de que se tenga diabetes la cual es la variable mas relevante. Le sigue BMI con una correlación de 0.29 lo que nos dice que un IMC más alto también se asocia con mayor riesgo. Variables como Age (0.24) y Pregnancies (0.22) muestran correlaciones positivas bajas, indicando que personas mayores y con más embarazos tienen un riesgo ligeramente mayor de diabetes, pero estas variables tienen un impacto menor en outcome

¿Hay alguna variable que no aporta información? Si tuvieras que eliminar variables, ¿cuáles quitarías y por qué? Considero que glucosa y BMI son de las variables que mas aportan informacion por lo que eliminaria las demas variables como BloodPressure, SkinThickness y DiabetesPedigreeFunction que tienen correlaciones muy bajas con Outcome

Si comparas el rango de las variables (min-max), ¿todas están en rangos similares? Describe sus rangos.