Comprensión de los Datos

Diego Sebastian Oyervides Ortiz A00838810 9/19/2025

In [31]:
```
#importa librerías
import pandas as pd
import matplotlib.pyplot as plt
import sklearn
import seaborn as sns
import numpy as np
```

Descripción de Variables

Pregnancies: Número de embarazos. Variable numérica discreta.

Glucose: Concentración de glucosa plasmática a las 2 horas en una prueba oral de tolerancia. Variable numérica continua.

BloodPressure: Presión arterial diastólica en mmHg. Variable numérica continua.

SkinThickness: Espesor del pliegue cutáneo del tríceps en milímetros. Variable numérica continua.

Insulin: Insulina sérica en 2 horas (µU/ml). Variable numérica continua.

BMI: Índice de masa corporal (peso en kg / altura en m^2). Variable numérica continua.

DiabetesPedigreeFunction: Puntaje que estima la probabilidad de diabetes en función de antecedentes familiares. Variable numérica continua.

Age: Edad en años. Variable numérica discreta.

Outcome: Diagnóstico de diabetes (0 = No, 1 = Sí). Variable categórica binaria.

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

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

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

Out[3]: (768, 9)

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

```
diabetes.head(10)
```

Out[4]:

	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
5	5	116	74	0	0	25.6	0
6	3	78	50	32	88	31.0	0
7	10	115	0	0	0	35.3	0
8	2	197	70	45	543	30.5	0
9	8	125	96	0	0	0.0	0

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

Out[5]:

	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 [6]:
```
#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
diabetes.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 [7]:
```
#revisa cuántos valores únicos tiene cada atributo del archivo usando la función nu
diabetes.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 puede incluir -0
diabetes.describe()
```

Out[8]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Dial
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 [9]:
```
#Revisa Valores nulos con funcion isnull().sum()
diabetes.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-columna.unique(
diabetes.Insulin.unique()
```

Out[10]:
```
array([  0,  94, 168,  88, 543, 846, 175, 230,  83,  96, 235, 146, 115,
       140, 110, 245,  54, 192, 207,  70, 240,  82,  36,  23, 300, 342,
       304, 142, 128,  38, 100,  90, 270,  71, 125, 176,  48,  64, 228,
        76, 220,  40, 152,  18, 135, 495,  37,  51,  99, 145, 225,  49,
        50,  92, 325,  63, 284, 119, 204, 155, 485,  53, 114, 105, 285,
       156,  78, 130,  55,  58, 160, 210, 318,  44, 190, 280,  87, 271,
       129, 120, 478,  56,  32, 744, 370,  45, 194, 680, 402, 258, 375,
       150,  67,  57, 116, 278, 122, 545,  75,  74, 182, 360, 215, 184,
        42, 132, 148, 180, 205,  85, 231,  29,  68,  52, 255, 171,  73,
       108,  43, 167, 249, 293,  66, 465,  89, 158,  84,  72,  59,  81,
       196, 415, 275, 165, 579, 310,  61, 474, 170, 277,  60,  14,  95,
       237, 191, 328, 250, 480, 265, 193,  79,  86, 326, 188, 106,  65,
       166, 274,  77, 126, 330, 600, 185,  25,  41, 272, 321, 144,  15,
       183,  91,  46, 440, 159, 540, 200, 335, 387,  22, 291, 392, 178,
       127, 510,  16, 112])
```

In [90]:
```
diabetes['BMI'].unique()
```

Out[90]:
```
array([33.6, 26.6, 23.3, 28.1, 43.1, 25.6, 31. , 35.3, 30.5,  0. , 37.6,
       38. , 27.1, 30.1, 25.8, 30. , 45.8, 29.6, 43.3, 34.6, 39.3, 35.4,
       39.8, 29. , 36.6, 31.1, 39.4, 23.2, 22.2, 34.1, 36. , 31.6, 24.8,
       19.9, 27.6, 24. , 33.2, 32.9, 38.2, 37.1, 34. , 40.2, 22.7, 45.4,
       27.4, 42. , 29.7, 28. , 39.1, 19.4, 24.2, 24.4, 33.7, 34.7, 23. ,
       37.7, 46.8, 40.5, 41.5, 25. , 25.4, 32.8, 32.5, 42.7, 19.6, 28.9,
       28.6, 43.4, 35.1, 32. , 24.7, 32.6, 43.2, 22.4, 29.3, 24.6, 48.8,
       32.4, 38.5, 26.5, 19.1, 46.7, 23.8, 33.9, 20.4, 28.7, 49.7, 39. ,
       26.1, 22.5, 39.6, 29.5, 34.3, 37.4, 33.3, 31.2, 28.2, 53.2, 34.2,
       26.8, 55. , 42.9, 34.5, 27.9, 38.3, 21.1, 33.8, 30.8, 36.9, 39.5,
       27.3, 21.9, 40.6, 47.9, 50. , 25.2, 40.9, 37.2, 44.2, 29.9, 31.9,
       28.4, 43.5, 32.7, 67.1, 45. , 34.9, 27.7, 35.9, 22.6, 33.1, 30.4,
       52.3, 24.3, 22.9, 34.8, 30.9, 40.1, 23.9, 37.5, 35.5, 42.8, 42.6,
       41.8, 35.8, 37.8, 28.8, 23.6, 35.7, 36.7, 45.2, 44. , 46.2, 35. ,
       43.6, 44.1, 18.4, 29.2, 25.9, 32.1, 36.3, 40. , 25.1, 27.5, 45.6,
       27.8, 24.9, 25.3, 37.9, 27. , 26. , 38.7, 20.8, 36.1, 30.7, 32.3,
       52.9, 21. , 39.7, 25.5, 26.2, 19.3, 38.1, 23.5, 45.5, 23.1, 39.9,
       36.8, 21.8, 41. , 42.2, 34.4, 27.2, 36.5, 29.8, 39.2, 38.4, 36.2,
       48.3, 20. , 22.3, 45.7, 23.7, 22.1, 42.1, 42.4, 18.2, 26.4, 45.3,
       37. , 24.5, 32.2, 59.4, 21.2, 26.7, 30.2, 46.1, 41.3, 38.8, 35.2,
       42.3, 40.7, 46.5, 33.5, 37.3, 30.3, 26.3, 21.7, 36.4, 28.5, 26.9,
       38.6, 31.3, 19.5, 20.1, 40.8, 23.4, 28.3, 38.9, 57.3, 35.6, 49.6,
       44.6, 24.1, 44.5, 41.2, 49.3, 46.3])
```

Variables Cuantitativas

Medidas de tendencia central

In [12]:
```
#Edad
#Se puede obtener la media, mediana y moda para
mean_age = diabetes['Age'].mean()
median_age =diabetes['Age'].median()
mode_age = diabetes['Age'].mode()
print("Mean_age:",mean_age)
print("Median_age:",median_age)
print("Mode_age:",mode_age)
```

```
Mean_age: 33.240885416666664
Median_age: 29.0
Mode_age: 0    22
Name: Age, dtype: int64
```

Conclusiones: La edad promedio fue 33 La edad al centro es 29 La edad más repetida fue de 22

Variables

In [89]:
```
#Para conteo  de cada valor en una columna, en orden descendente usar función value
# nombreDataframe.columna.value_counts()
# nombreDataframe['columna'].value_counts()
diabetes.BMI.value_counts()
```

Out[89]:
```
BMI
32.0    13
31.6    12
31.2    12
0.0     11
32.4    10
        ..
49.6     1
24.1     1
41.2     1
49.3     1
46.3     1
Name: count, Length: 248, dtype: int64
```

Consulta

In [17]:
```
# df.iloc[i]: Accede a la fila en la posición i.
# Acceder a la primera fila
diabetes.iloc[0]
```

Out[17]:
```
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
Name: 0, dtype: float64
```

In [18]:
```
# Acceder a las dos primeras filas
diabetes.iloc[:2]
```

Out[18]:

	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 [19]:
```
#Seleccionar columnas, indicando entre corchetes [nombreColumna, nombreColumna]
diabetes[['BMI', 'Insulin']]
```

Out[19]:

	BMI	Insulin
0	33.6	0
1	26.6	0
2	23.3	0
3	28.1	94
4	43.1	168
...	...	...
763	32.9	180
764	36.8	0
765	26.2	112
766	30.1	0
767	30.4	0

768 rows × 2 columns

In [34]:
```
#Selección de filas [indicar dataframe[columna] operador valor]
sobrepeso = diabetes[diabetes['BMI']>=25]
```

In [35]:
```
sobrepeso.head()
```

Out[35]:

	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
3	1	89	66	23	94	28.1	0
4	0	137	40	35	168	43.1	2
5	5	116	74	0	0	25.6	0

In [22]:
```
#ordenar usando funcion sort_values(by=atributo, ascending=True/false)
sobrepeso.sort_values(by='DiabetesPedigreeFunction', ascending=False)
```

Out[22]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
445	0	180	78	63	14	59.4	
228	4	197	70	39	744	36.7	
4	0	137	40	35	168	43.1	
370	3	173	82	48	465	38.4	
45	0	180	66	39	0	42.0	
...	...	...	...	...	...	...	
135	2	125	60	20	140	33.8	
598	1	173	74	0	0	36.8	
149	2	90	70	17	0	27.3	
567	6	92	62	32	126	32.0	
268	0	102	52	0	0	25.1	

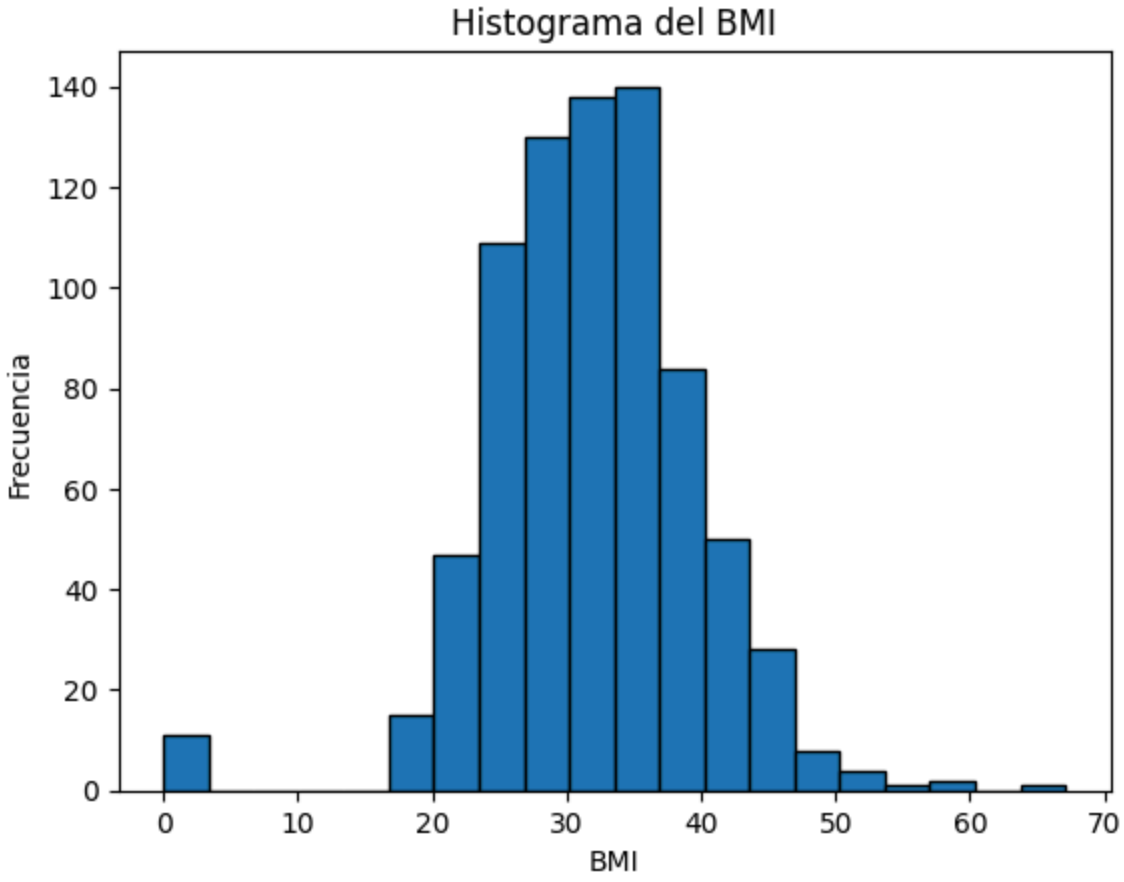
651 rows × 9 columns

In [23]:
```
#Agrupar por un atributo y calcular función de agregación utilizando groupby(atribu
sobrepeso.groupby('Outcome')['Glucose'].mean()
```

Out[23]:
```
Outcome
0    111.711735
1    140.980695
Name: Glucose, dtype: float64
```

Visuzalizacion de datos

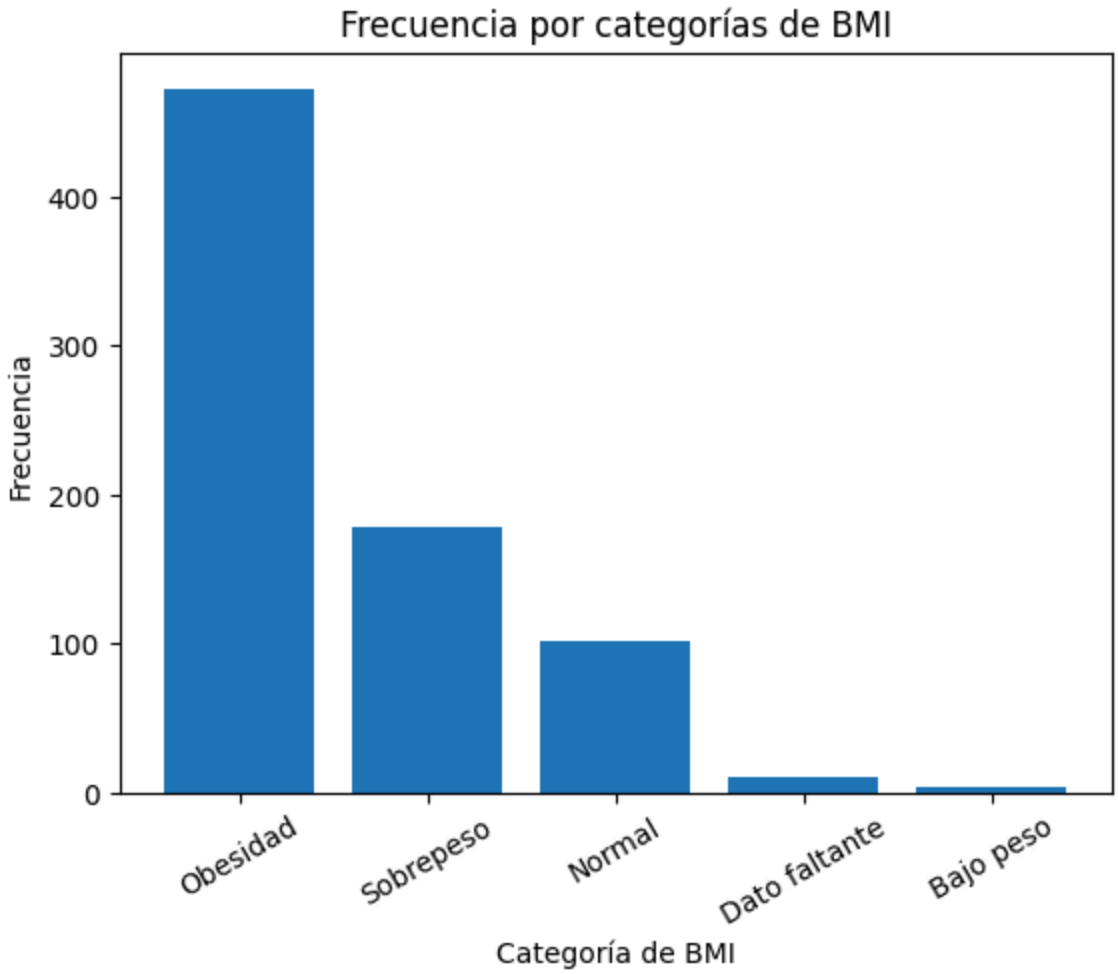
In [37]:
```
plt.hist(diabetes["BMI"], bins=20, edgecolor="black")
plt.title("Histograma del BMI")
plt.xlabel("BMI")
plt.ylabel("Frecuencia")
plt.show()
```



Aqui podemos ver donde se concentran los valores de los px dependiendo de su BMI. Podemos interpretar que la mayoria de los px tienen un BMI Mayor a 30, pero menor a 40

In [41]:
```
def clasificar_bmi(bmi):
    if bmi == 0:
        return "Dato faltante"
    elif bmi < 18.5:
        return "Bajo peso"
    elif 18.5 <= bmi < 25:
        return "Normal"
    elif 25 <= bmi < 30:
        return "Sobrepeso"
    else:
        return "Obesidad"
```

In [85]:
```
diabetes["BMI_Category"] = diabetes["BMI"].apply(clasificar_bmi)

conteo_categorias = diabetes["BMI_Category"].value_counts()

plt.bar(conteo_categorias.index, conteo_categorias.values)
plt.title("Frecuencia por categorías de BMI")
plt.xlabel("Categoría de BMI")
plt.ylabel("Frecuencia")
plt.xticks(rotation=30)
plt.show()
```



In [86]:
```
# Filtrar personas con obesidad y diabetes positiva
obesidad_diabetes = diabetes[(diabetes["BMI_Category"] == "Obesidad") & (diabetes["

print("Número de personas con obesidad y diabetes:", len(obesidad_diabetes))
```

```
Número de personas con obesidad y diabetes: 219
```

Para crear la grafica de barras primero cree una columna, usando una funcion que cree, donde clasifico si tienen obesidad, sobrepeso, normal, bajo peso, o dato faltante dependiendo de su BMI. Despues hago el conteo de las categorias y grafico estos numeros para asi conocer el numero de personas con cada diagnostico. Ya despues saco la variable de Obesidad del dataset y el outcome cuando es igual a uno. Y de misma manera veo cuando esta condicion es igual, hago el conteo y asi consigo el numero de personas con obesidad y diabetes.

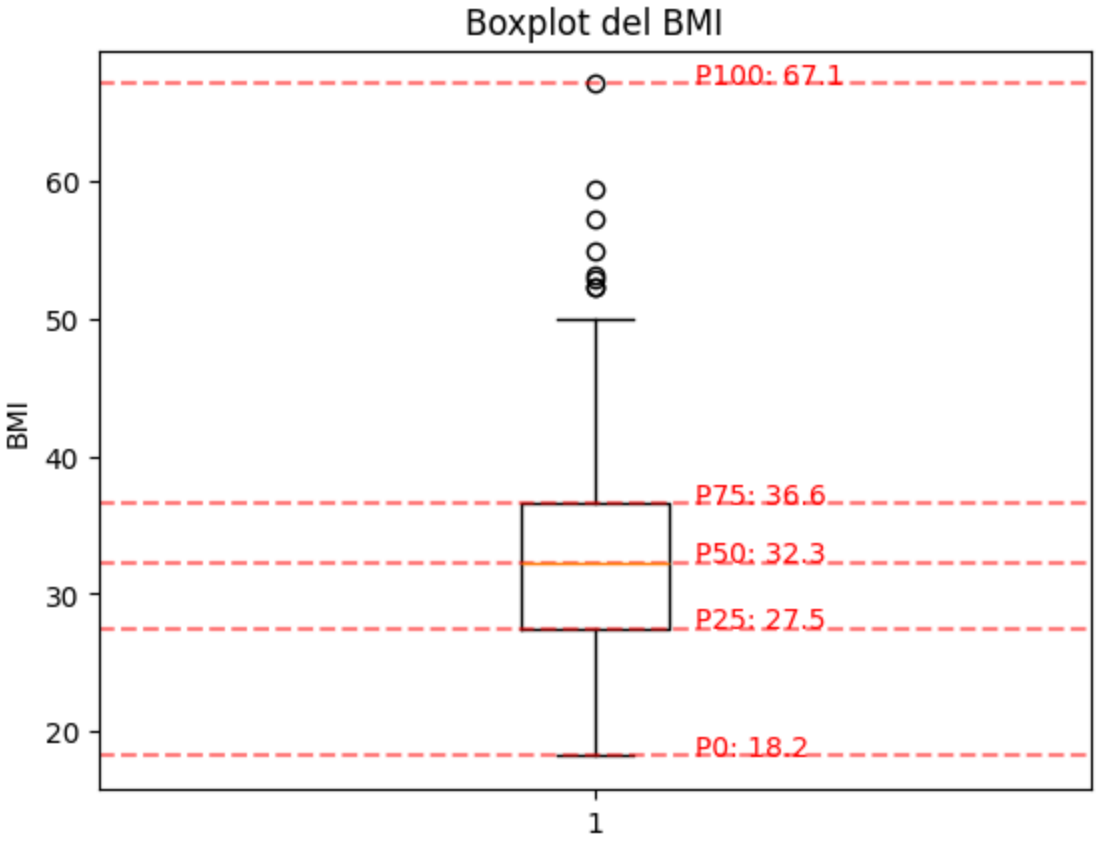
In [87]:
```
# Tabla cruzada entre BMI_Category y Outcome
tabla = pd.crosstab(diabetes["BMI_Category"], diabetes["Outcome"])

print(tabla)
```

```
Outcome          0    1
BMI_Category
Bajo peso        4    0
Dato faltante    9    2
Normal          95    7
Obesidad       253  219
Sobrepeso      139   40
```

Aqui solo veo que px dependiendo de su diagnostico de bmi tiene obesidad

In [49]:
```python
# Filtrar valores de BMI válidos (descartar 0 que indican faltante)
bmi_valid = diabetes["BMI"][diabetes["BMI"] > 0]

# Calcular percentiles
percentiles = np.percentile(bmi_valid, [0, 25, 50, 75, 100])

print("Percentiles del BMI:")
print(f"  Mínimo (0%): {percentiles[0]:.2f}")
print(f"  Q1 (25%): {percentiles[1]:.2f}")
print(f"  Mediana (50%): {percentiles[2]:.2f}")
print(f"  Q3 (75%): {percentiles[3]:.2f}")
print(f"  Máximo (100%): {percentiles[4]:.2f}")

# Boxplot con anotaciones
plt.boxplot(bmi_valid)
plt.title("Boxplot del BMI")
plt.ylabel("BMI")

# Agregar anotaciones de percentiles
for i, val in enumerate(percentiles):
    plt.axhline(val, color="red", linestyle="--", alpha=0.5)
    plt.text(1.1, val, f"P{i*25}: {val:.1f}", color="red")

plt.show()
```

```
Percentiles del BMI:
  Mínimo (0%): 18.20
  Q1 (25%): 27.50
  Mediana (50%): 32.30
  Q3 (75%): 36.60
  Máximo (100%): 67.10
```



EN la box plot observamos casos muy curiosos. Donde podemos observar como existe un valor bastante atipico dentro de nuestro analisis un px con BMI de 67.1. Vemos como solo el 25% de la poblacion se encuentra dentro de un rango de BMI s sin sobrepeso o obesidad.

Matriz de correlacion

In [78]:
```
variables_numericas = diabetes.select_dtypes(include='number')
matriz_correlacion = variables_numericas.corr().round(2)#round dar solo las decimal
```

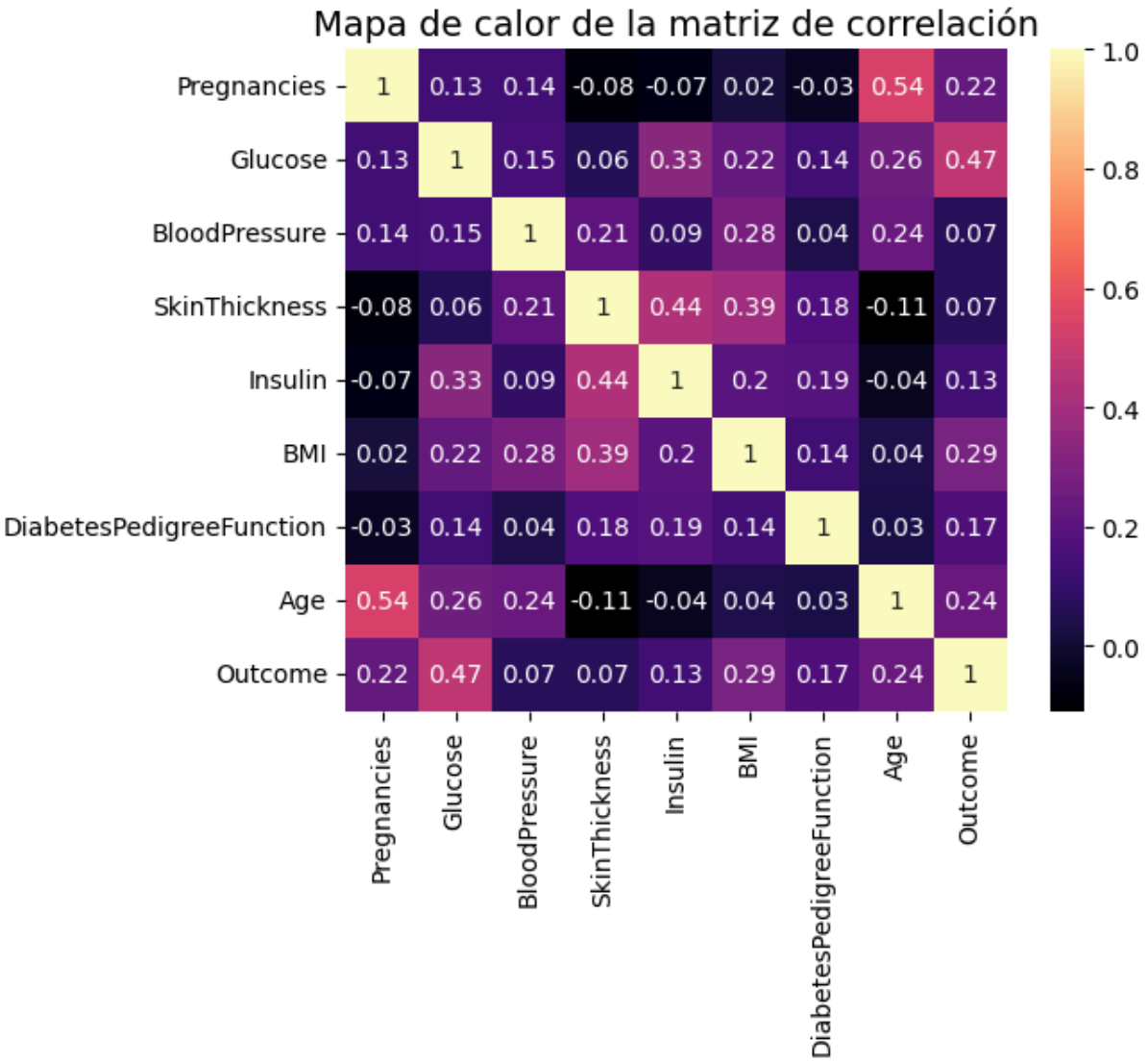
In [79]:
```
matriz_correlacion
```

Out[79]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BM
Pregnancies	1.00	0.13	0.14	-0.08	-0.07	0.0
Glucose	0.13	1.00	0.15	0.06	0.33	0.2
BloodPressure	0.14	0.15	1.00	0.21	0.09	0.2
SkinThickness	-0.08	0.06	0.21	1.00	0.44	0.3
Insulin	-0.07	0.33	0.09	0.44	1.00	0.2
BMI	0.02	0.22	0.28	0.39	0.20	1.0
DiabetesPedigreeFunction	-0.03	0.14	0.04	0.18	0.19	0.1
Age	0.54	0.26	0.24	-0.11	-0.04	0.0
Outcome	0.22	0.47	0.07	0.07	0.13	0.2

Mapa de calor

In [80]:
```
sns.heatmap(matriz_correlacion,
            annot=True,        # Muestra los valores dentro de las celd# Formato de
            cmap="magma",   # Paleta de colores
            cbar=True,         # Barra de colores
            square=True)       # Hace las celdas cuadradas

plt.title("Mapa de calor de la matriz de correlación", fontsize=14)

plt.show()
```



Para crear el heatmap tuve que pasar todas mis variables a numericas. Ya que cree una variable categorica para clasificar por BMI. Vemos que la variable de BMI es significativa dentro de el analisis para saber si un px tiene diabetes o no.

In [83]:
```
resumen = pd.DataFrame({
    "Mínimo": variables_numericas.min(),
    "Máximo": variables_numericas.max(),
    "Rango": variables_numericas.max() - variables_numericas.min()
})

print(resumen)
```

```
                           Mínimo  Máximo    Rango
Pregnancies                 0.000   17.00   17.000
Glucose                     0.000  199.00  199.000
BloodPressure               0.000  122.00  122.000
SkinThickness               0.000   99.00   99.000
Insulin                     0.000  846.00  846.000
BMI                         0.000   67.10   67.100
DiabetesPedigreeFunction    0.078    2.42    2.342
Age                        21.000   81.00   60.000
Outcome                     0.000    1.00    1.000
```

Analisis final

¿Hay alguna variable que no aporta información? Si tuvieras que eliminar variables, ¿cuáles quitarías y por qué?
Tal vez quitaria presion arterial, y skinthickness, pues estas variables muestran una correlacion casi despreciable con el outcome de si tienen diabetes o no. Asi mismo mi Si comparas el rango de las variables (min-max), ¿todas están en rangos similares? Describe sus rangos.
Creo que la mayoria tienen rangos distintos pues miden cosas muy diferentes, ahora bien. Los minimos de mis variables muestran que no hay datos no que el valor sea 0. A excepcion de edad y DiabetesPedigreeFunction
¿Existen variables que tengan datos atípicos? Describe cuáles si o no.
Dentro de mis variables existe un valor muy atipico donde vemos a un px con un BMI muy alta
¿Existe correlación alta entre variables? Describe algunas, indicando si es correlación positiva o negativa.

In [ ]: