Librerías

In [6]:
```
#Graficos
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
#Dataframes
import numpy as np
import pandas as pd
#Verifica valores perdidos
```

In [7]:

In [8]:

Lectura / importación de datos

In [9]:

In [10]:

In [11]:

Out[11]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type
0	9046	Male	67.0	0	1	Yes	Private	Urban
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural
2	31112	Male	80.0	0	1	Yes	Private	Rural
3	60182	Female	49.0	0	0	Yes	Private	Urban
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural
5	56669	Male	81.0	0	0	Yes	Private	Urban
6	53882	Male	74.0	1	1	Yes	Private	Rural
7	10434	Female	69.0	0	0	No	Private	Urban
8	27419	Female	59.0	0	0	Yes	Private	Rural
9	60491	Female	78.0	0	0	Yes	Private	Urban
10	12109	Female	81.0	1	0	Yes	Private	Rural
11	12095	Female	61.0	0	1	Yes	Govt_job	Rural
12	12175	Female	54.0	0	0	Yes	Private	Urban
13	8213	Male	78.0	0	1	Yes	Private	Urban
14	5317	Female	79.0	0	1	Yes	Private	Urban
15	58202	Female	50.0	1	0	Yes	Self-employed	Rural
16	56112	Male	64.0	0	1	Yes	Private	Urban
17	34120	Male	75.0	1	0	Yes	Private	Urban
18	27458	Female	60.0	0	0	No	Private	Urban
19	25226	Male	57.0	0	1	No	Govt_job	Urban

In [12]:

Out[12]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_typ
5100	68398	Male	82.0	1	0	Yes	Self-employed	Rura
5101	36901	Female	45.0	0	0	Yes	Private	Urba
5102	45010	Female	57.0	0	0	Yes	Private	Rura
5103	22127	Female	18.0	0	0	No	Private	Urba
5104	14180	Female	13.0	0	0	No	children	Rura
5105	18234	Female	80.0	1	0	Yes	Private	Urba
5106	44873	Female	81.0	0	0	Yes	Self-employed	Urba
5107	19723	Female	35.0	0	0	Yes	Self-employed	Rura
5108	37544	Male	51.0	0	0	Yes	Private	Rura
5109	44679	Female	44.0	0	0	Yes	Govt_job	Urba

In [13]:
```
#El método "describe" devuelve información estadística de los datos del da
#(de hecho, este método devuelve un dataframe).
#Esta información incluye el número de muestras, el valor medio,
#la desviación estándar, el valor mínimo, máximo, la mediana y los valores
```

Out[13]:

	id	age	hypertension	heart_disease	avg_glucose_level	bmi
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28.893237
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7.854067
min	67.000000	0.080000	0.000000	0.000000	55.120000	10.300000
25%	17741.250000	25.000000	0.000000	0.000000	77.245000	23.500000
50%	36932.000000	45.000000	0.000000	0.000000	91.885000	28.100000
75%	54682.000000	61.000000	0.000000	0.000000	114.090000	33.100000
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97.600000

In [14]:
```
# Filas y columnas dataframe
```

Out[14]: (5110, 12)

In [15]:
```
# Acceder al índice
```

Out[15]: RangeIndex(start=0, stop=5110, step=1)

In [16]:
```
# Columnas del data frame
```

Out[16]:
```
Index(['id', 'gender', 'age', 'hypertension', 'heart_disease', 'ever_married
',
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status', 'stroke'],
      dtype='object')
```

In [17]:
```
# función longitud para saber el número de columnas
```

Out[17]:
```
12
```

In [18]:
```
# Acceder a una columnas llamada gender
```

Out[18]:
```
0         Male
1       Female
2         Male
3       Female
4       Female
5         Male
6         Male
7       Female
8       Female
9       Female
10      Female
11      Female
12      Female
13        Male
14      Female
15      Female
16        Male
17        Male
18      Female
```

Limpiar datos

In [19]:

Out[19]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type
0	9046	Male	67.0	0	1	Yes	Private	Urban
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural
2	31112	Male	80.0	0	1	Yes	Private	Rural
3	60182	Female	49.0	0	0	Yes	Private	Urban
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural

In [20]:
```
1 # Tipos de variables de dataframe:
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 5110 non-null   int64  
 1   gender             5110 non-null   object 
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64  
 4   heart_disease      5110 non-null   int64  
 5   ever_married       5110 non-null   object 
 6   work_type          5110 non-null   object 
 7   Residence_type     5110 non-null   object 
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object 
 11  stroke             5110 non-null   int64  
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
```

In [21]:
```
1 # Cambiar datatype:
2 # cambiar una columna a objeto
3 data["id"] = data["id"].astype("object")
4 data["hypertension"]=data["hypertension"].astype("object")
5 data["heart_disease"]=data["heart_disease"].astype("object")
6 data["stroke"]=data["stroke"].astype("object")
7 # cambiar una columna a entero
```

In [22]:
```

```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 5110 non-null   object 
 1   gender             5110 non-null   object 
 2   age                5110 non-null   int64  
 3   hypertension       5110 non-null   object 
 4   heart_disease      5110 non-null   object 
 5   ever_married       5110 non-null   object 
 6   work_type          5110 non-null   object 
 7   Residence_type     5110 non-null   object 
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object 
 11  stroke             5110 non-null   object 
dtypes: float64(2), int64(1), object(9)
memory usage: 479.2+ KB
```

In [23]:

Out[23]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	a
0	9046	Male	67	0	1	Yes	Private	Urban	
1	51676	Female	61	0	0	Yes	Self-employed	Rural	
2	31112	Male	80	0	1	Yes	Private	Rural	
3	60182	Female	49	0	0	Yes	Private	Urban	
4	1665	Female	79	1	0	Yes	Self-employed	Rural	

In [24]:
```
# Ejemplo de cómo poner una columna como índice:
data_ejemplo = data.copy()
data_ejemplo = data_ejemplo.set_index('id')
```

```
C:\Users\Alumne_mati1\anaconda3\lib\site-packages\pandas\core\indexes\base.p
y:6982: FutureWarning: In a future version, the Index constructor will not in
fer numeric dtypes when passed object-dtype sequences (matching Series behavi
or)
  return Index(sequences[0], name=names)
```

Out[24]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_
id								
9046	Male	67	0	1	Yes	Private	Urban	
51676	Female	61	0	0	Yes	Self-employed	Rural	
31112	Male	80	0	1	Yes	Private	Rural	
60182	Female	49	0	0	Yes	Private	Urban	
1665	Female	79	1	0	Yes	Self-employed	Rural	

Buscar y corregir errores

Revisar datos anomalos

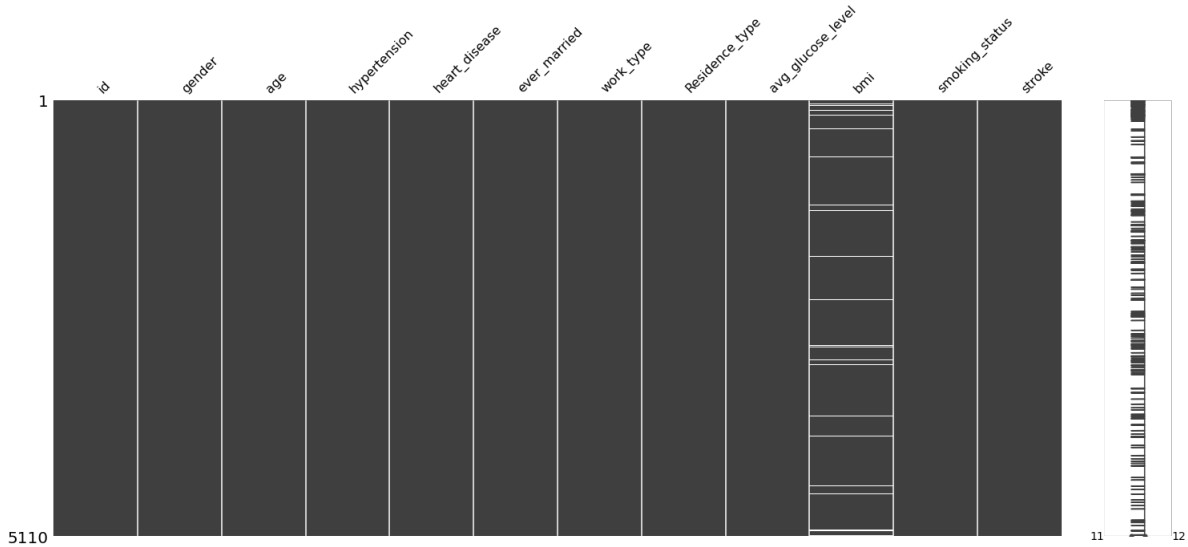
In [25]:
```
# identificar valores máximos y mínimos:
data["age"].min()
```

Out[25]: 82

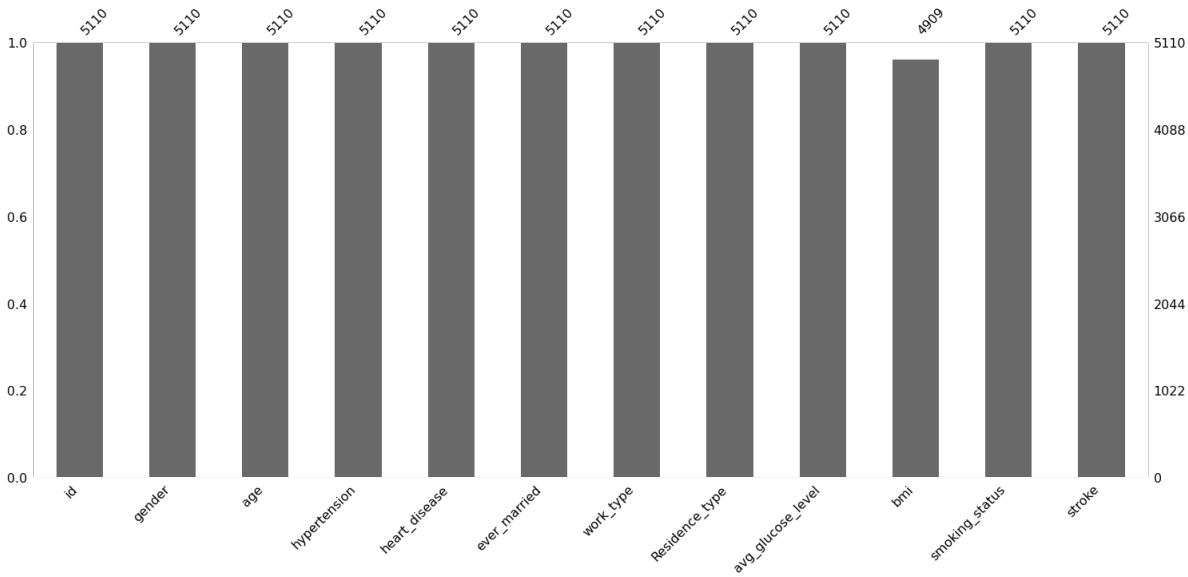
In [26]:
```
1 # identificar outliers(valores anomalos que se salen mucho del rango):
2 fig = px.box(data, y="avg_glucose_level")
```

Analizar nans(valores perdidos)

In [26]:
```
# Matriz de los nans:
msno.matrix(data)
```

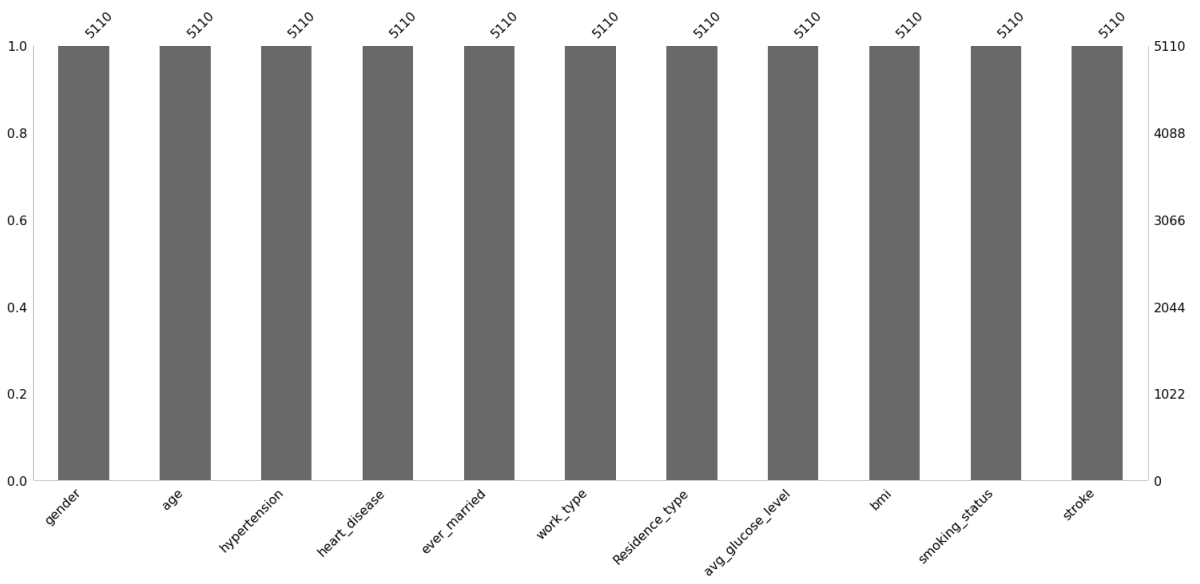


In [27]:
```
# Barplot de los nans
msno.bar(data)
```



In [28]:
```
#Tratar los nans:
# Opción 1 - eliminar los nans - dropna
```

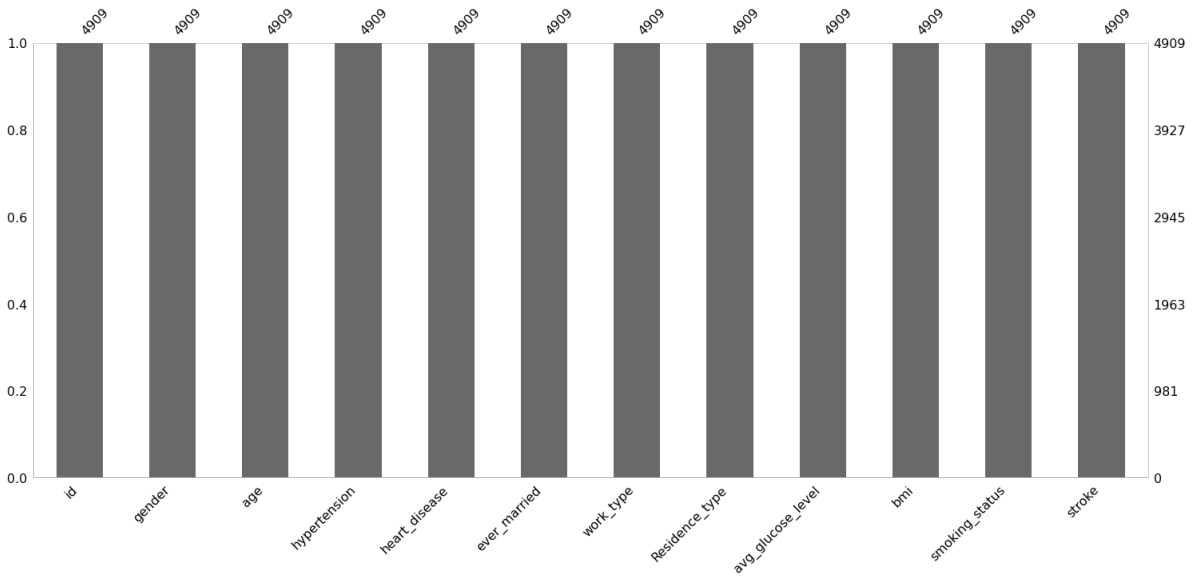
In [29]:
```
# Ejemplo de imputación con la media:
data_ejemplo["bmi"] = data_ejemplo["bmi"].fillna(data_ejemplo["bmi"].mean(
msno.bar(data_ejemplo)
```



In [30]:
```
# Eliminar los nans:
data = data.dropna()
```

Out[30]: (4909, 12)

In [31]:
```
msno.bar(data)
```



In [32]:
```
# Criterios para elminar:
# variable cuantitativa con datos iguales en casi todas las filas
# variable que tenga el 85% de nans
```

In [33]:
```
columnas_borrar = ["id"]
data = data.drop(columnas_borrar,axis = 1)
```

Out[33]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_gluc
0	Male	67	0	1	Yes	Private	Urban	
2	Male	80	0	1	Yes	Private	Rural	
3	Female	49	0	0	Yes	Private	Urban	
4	Female	79	1	0	Yes	Self-employed	Rural	
5	Male	81	0	0	Yes	Private	Urban	

Análisis de los datos

Analisis univariado variables cuantitativas

In [35]:
```

```

Out[35]:
```
Index(['gender', 'age', 'hypertension', 'heart_disease', 'ever_married',
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status', 'stroke'],
      dtype='object')
```

In [36]:
```

```

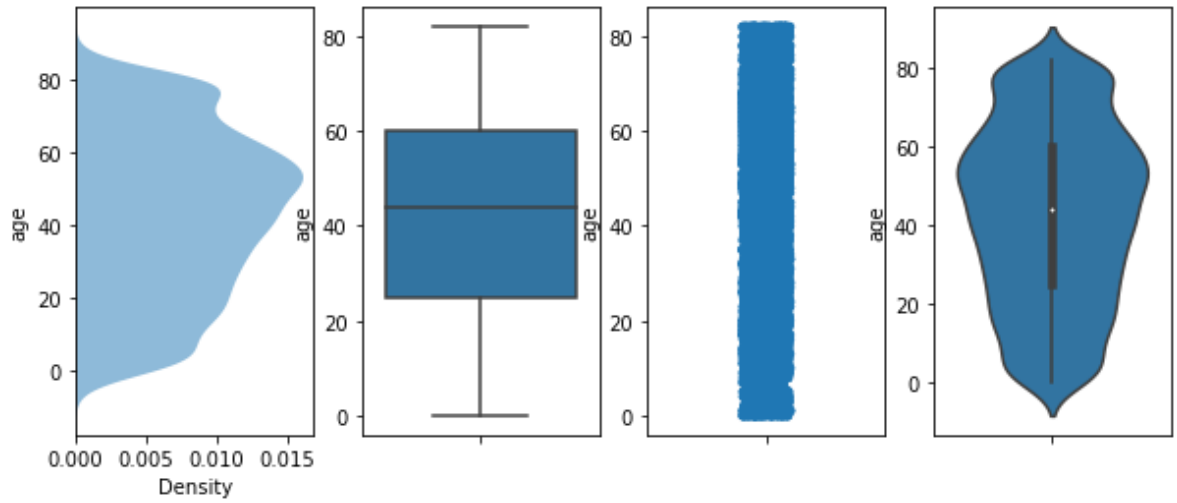
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4909 entries, 0 to 5109
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   gender             4909 non-null   object 
 1   age                4909 non-null   int64  
 2   hypertension       4909 non-null   object 
 3   heart_disease      4909 non-null   object 
 4   ever_married       4909 non-null   object 
 5   work_type          4909 non-null   object 
 6   Residence_type     4909 non-null   object 
 7   avg_glucose_level  4909 non-null   float64
 8   bmi                4909 non-null   float64
 9   smoking_status     4909 non-null   object 
 10  stroke             4909 non-null   object 
dtypes: float64(2), int64(1), object(8)
memory usage: 460.2+ KB
```

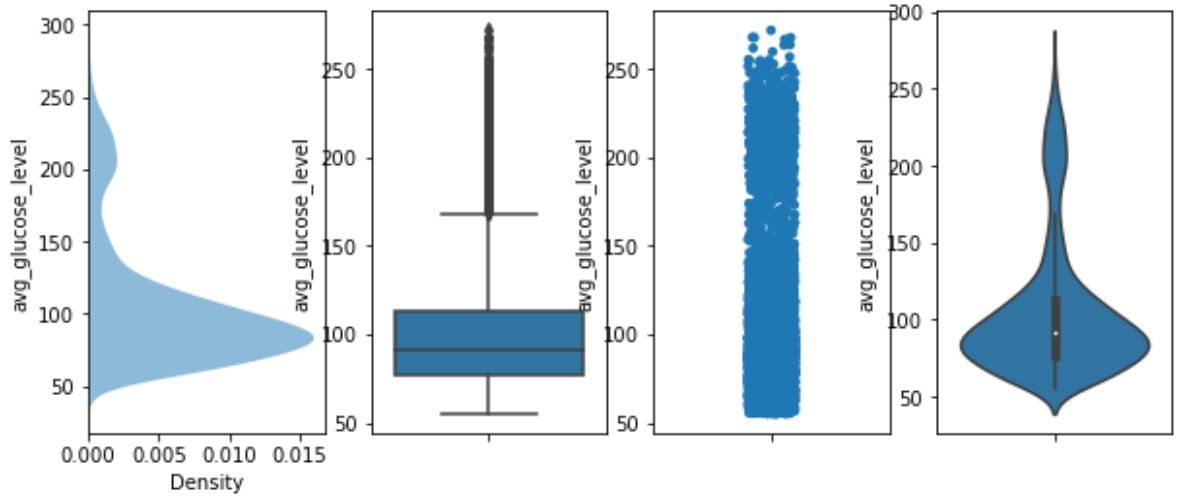
In [37]:
```
# Boxplot
# Density plot - histograma
# Mean plot - error plot
# Stripchart
```

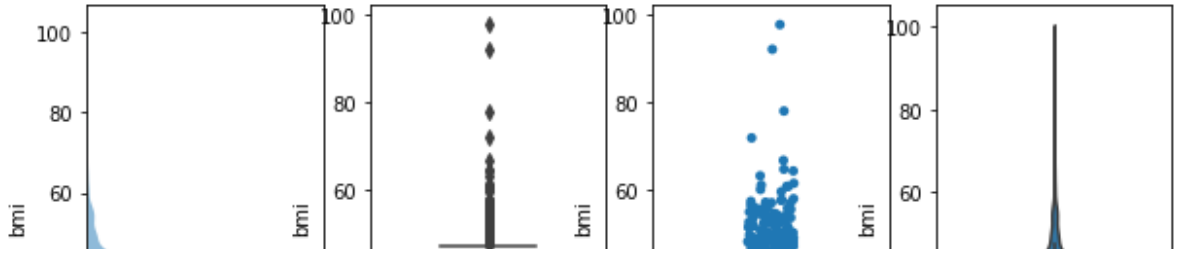
In [38]:

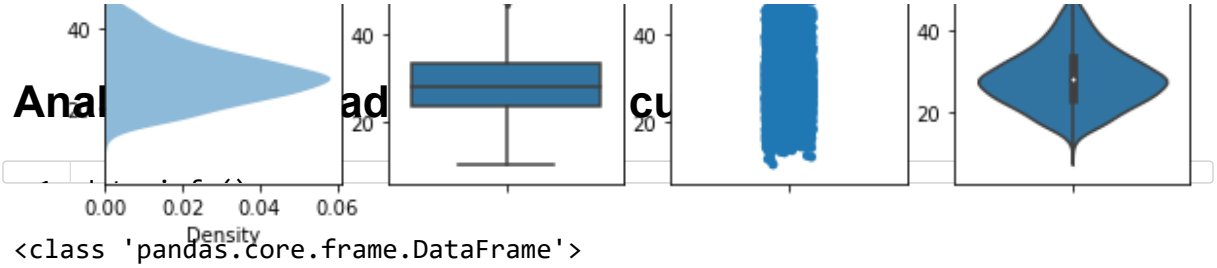
In [39]:

```
# Descripcion de todas las variables cuantitativas

for i in var_num:
    name_var = i
    fig, axs = plt.subplots(ncols=4, figsize=(10, 4))
    # Histograma de densidad
    sns.kdeplot(
        data=data, y=name_var,
        fill=True, common_norm=False, palette="crest",
        alpha=.5, linewidth=0, ax=axs[0])
    # Boxplot
    sns.boxplot(data=data, y=name_var,ax=axs[1])
    # Stripchart
    sns.stripplot(y= name_var,data=data,ax=axs[2])
    # Violinplot
    sns.violinplot(data=data, y=name_var,ax=axs[3])
```









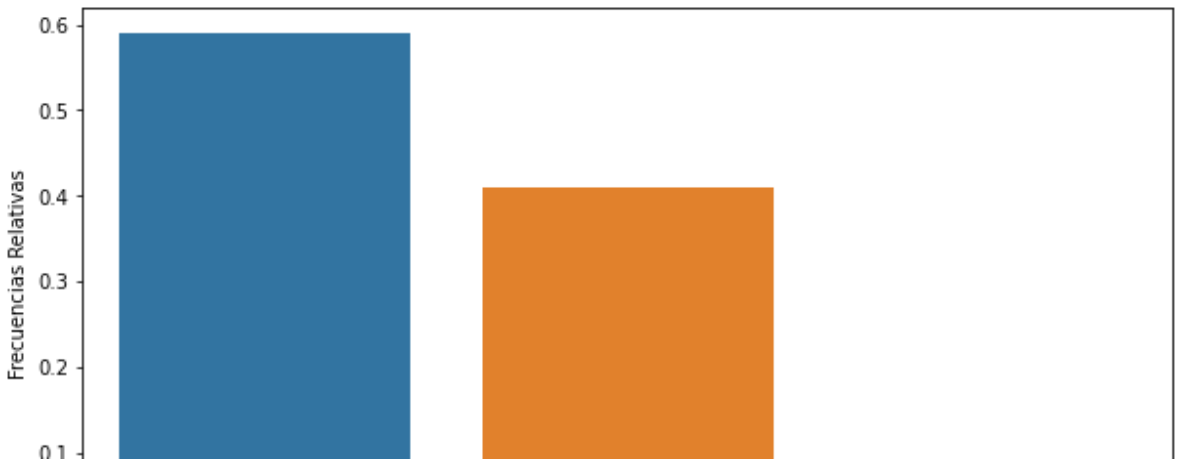
In [40]:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4909 entries, 0 to 5109
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   gender             4909 non-null   object 
 1   age                4909 non-null   int64  
 2   hypertension       4909 non-null   object 
 3   heart_disease      4909 non-null   object 
 4   ever_married       4909 non-null   object 
 5   work_type          4909 non-null   object 
 6   Residence_type     4909 non-null   object 
 7   avg_glucose_level  4909 non-null   float64
 8   bmi                4909 non-null   float64
 9   smoking_status     4909 non-null   object 
 10  stroke             4909 non-null   object 
dtypes: float64(2), int64(1), object(8)
memory usage: 589.3+ KB
```

In [41]:

```
# gráficos y tablas de frecuencias de todas las variables cualitativas:
#La frecuencia relativa es el cociente de la frecuencia absoluta y el núme
#Para calcular la frecuencia relativa de cada dato dividimos
#la frecuencia absoluta del dato entre el número total de datos.


var_cual = ['gender','hypertension', 'heart_disease', 'ever_married',
       'work_type', 'Residence_type',
       'smoking_status', 'stroke']

for i in var_cual:

    print("Variable = "+i)

    # 1. Tabla
    pct = pd.DataFrame(data[i].value_counts(normalize=True))
    pct.columns = ['Frecuencias Relativas']
    pct["Frecuencias Absolutas"] = data[i].value_counts()
    print(pct)

    # 2. Barplot
    sns.barplot(x=pct.index, y='Frecuencias Relativas', data=pct)
    plt.show()

    # 3. Pieplot
    frec_abs = data[i].value_counts() # grabo las frecuencias absolutas
    labels = data[i].value_counts().index # cojo los nombres de los grupos
    colors = sns.color_palette('pastel')[0:len(labels)] # defino los color
    plt.pie(frec_abs, labels = labels, colors = colors, autopct='%.0f%%')
```

```
Variable = gender
        Frecuencias Relativas  Frecuencias Absolutas
Female               0.590141                   2897
Male                 0.409656                   2011
Other                0.000204                      1
```



Diagramas de dispersión variables cuantitativas

In [42]:
```
#Los gráficos de dispersión se utilizan para mostrar relaciones.
#Para la correlación, los gráficos de dispersión ayudan a mostrar la fuerz
#Para la regresión, los gráficos de dispersión suelen incorporar una líneq
#En control de calidad, los gráficos de dispersión pueden con frecuencia i
```

In [43]:
```

```

Out[43]:
```
Index(['gender', 'age', 'hypertension', 'heart_disease', 'ever_married',
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status', 'stroke'],
      dtype='object')
```

Scatter 3D

In [27]:
```
# Diagramas de dispersión 3D de variables cuantitativas

z = 'avg_glucose_level'
x = 'bmi'
y = 'age'

fig = px.scatter_3d(data,x=x,y=y,z=z,size_max=0.1)
fig.update_traces(marker_size = 1)
```

Matrixplot

In [45]:

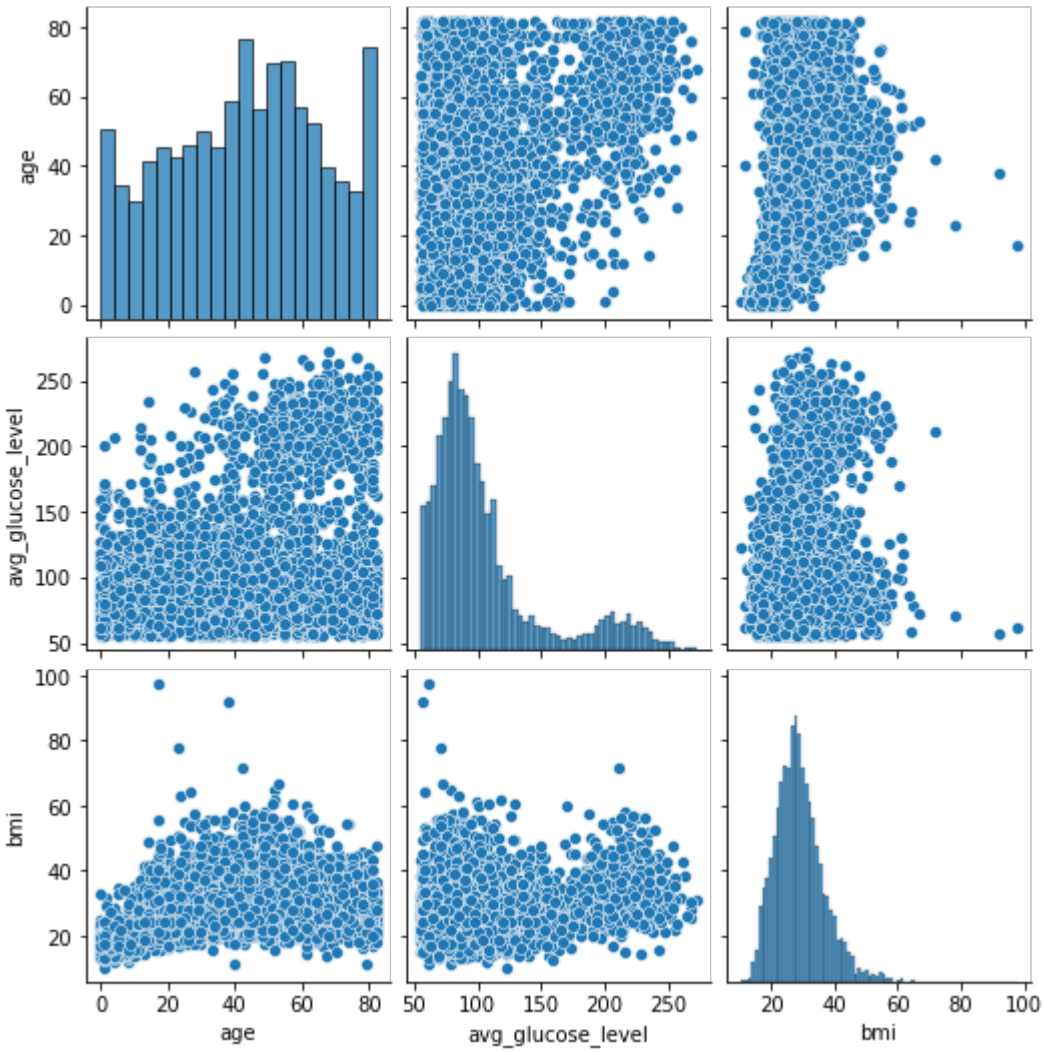
In [46]:

Out[46]:

	age	avg_glucose_level	bmi
0	67	228.69	36.6
2	80	105.92	32.5
3	49	171.23	34.4
4	79	174.12	24.0
5	81	186.21	29.0

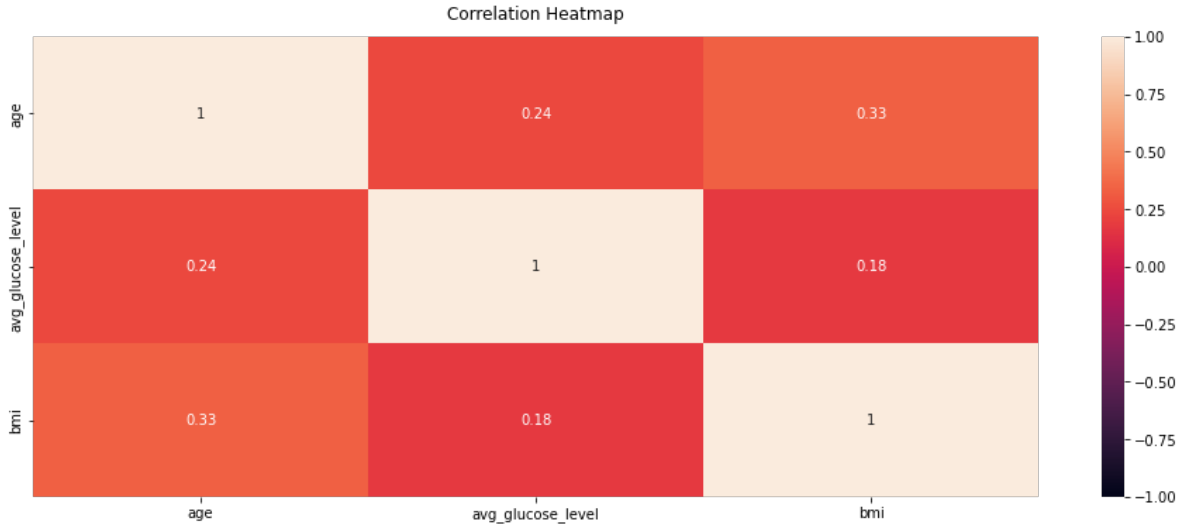
In [47]:

```
# Matrixplot

columnas_plot = ["age","avg_glucose_level","bmi"]
# seaborn
#sns.pairplot(data[var_num])
sns.pairplot(data[columnas_plot])
plt.show()
```



Correlograma

In [49]:
```
# Visualizacion de la correlación serial en datos que cambian con el tiemp
#(es decir , datos de series temporales ).

X = data[var_num]

plt.figure(figsize=(16, 6))
heatmap = sns.heatmap(X.corr(), vmin=-1, vmax=1, annot=True)

heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':12}, pad=12)
```



Diagramas variables cualitativas

In [50]:

Out[50]:
```
Index(['gender', 'age', 'hypertension', 'heart_disease', 'ever_married',
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status', 'stroke'],
      dtype='object')
```

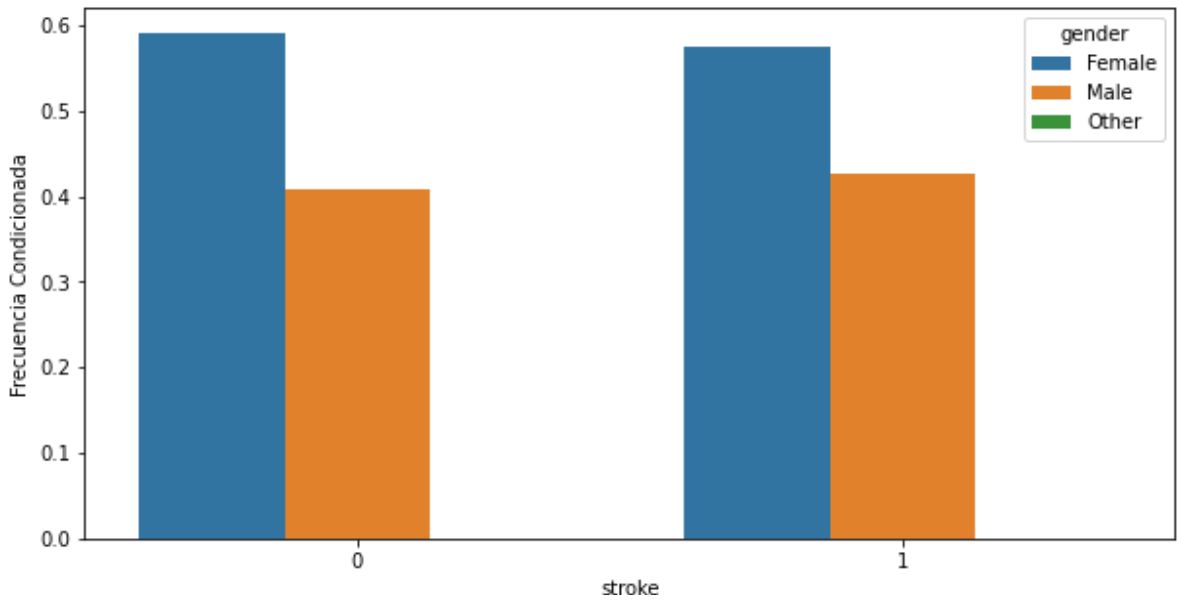
In [51]:

Out[51]:
```
['gender',
 'hypertension',
 'heart_disease',
 'ever_married',
 'work_type',
 'Residence_type',
 'smoking_status',
 'stroke']
```

In [52]:

```
# Diagramas de barras de dos factores de frecuencias condicionadas a los g
#(Son las frecuencias que posee una variable si sólo consideramos un valor
#distribución bidimensional X-Y. En la práctica se traduce a considerar só
#según el valor elegido.

x = 'stroke'
vec_y = ['gender',
 'hypertension',
 'heart_disease',
 'ever_married',
 'work_type',
 'Residence_type',
 'smoking_status']


for i in vec_y:

    y = i
    print(y + ' vs ' + x)
    pct2 = (data.groupby([x,y]).size() / data.groupby([x]).size()).reset_i
    # seaborn
    sns.barplot(x=x, hue=y, y='Frecuencia Condicionada', data=pct2)
    plt.show()
    print(pct2)
```

```
gender vs stroke
```



In [ ]: