Brief description of the data set and a summary of its attributes-

The following dataset used titled 'stroke prediction' is a dataset whhich covers the various parameters which might be considered while diagonosing stroke. The data contains 5110 observations with 12 attributes.

Data Dictionary

hypertension = High blood pressure

ever_married = marital status

work_type = Type of work

Residence_type = Environment where the subject lives.

avg_glucose_level = estimated average of your blood sugar (glucose) levels over a period of 2 to 3 months bmi Body mass index

```
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle datasets download -d fedesoriano/stroke-prediction-dataset
```

```
Downloading stroke-prediction-dataset.zip to /content
  0% 0.00/67.4k [00:00<?, ?B/s]
100% 67.4k/67.4k [00:00<00:00, 4.44MB/s]
```

```
!unzip stroke-prediction-dataset.zip
```

```
Archive:  stroke-prediction-dataset.zip
  inflating: healthcare-dataset-stroke-data.csv
```

```
%matplotlib inline
#Importing required libraries

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
path='/content/healthcare-dataset-stroke-data.csv'
df=pd.read_csv(path)
```

```
df.head()
type(df)
```

```
pandas.core.frame.DataFrame
```

```
type(df)
```

```
pandas.core.frame.DataFrame
```

```
df.info
print(type(df),df.info)
```

```
<class 'pandas.core.frame.DataFrame'> <bound method DataFrame.info of          id  gende
0      9046    Male  67.0  ...  36.6  formerly smoked      1
1     51676  Female  61.0  ...   NaN     never smoked      1
2     31112    Male  80.0  ...  32.5     never smoked      1
3     60182  Female  49.0  ...  34.4           smokes      1
4      1665  Female  79.0  ...  24.0     never smoked      1
...     ...     ...   ...  ...   ...              ...    ...
5105  18234  Female  80.0  ...   NaN     never smoked      0
5106  44873  Female  81.0  ...  40.0     never smoked      0
5107  19723  Female  35.0  ...  30.6     never smoked      0
5108  37544    Male  51.0  ...  25.6  formerly smoked      0
5109  44679  Female  44.0  ...  26.2          Unknown      0

[5110 rows x 12 columns]>
```

```
print(df.columns)
```

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

```
df.dtypes
```

```
id                   int64
gender              object
age                float64
hypertension         int64
heart_disease        int64
ever_married        object
work_type           object
Residence_type      object
avg_glucose_level  float64
bmi                float64
smoking_status      object
stroke               int64
dtype: object
```

```
df.describe()
```

	id	age	hypertension	heart_disease	avg_glucose_level	
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	4909.0(
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28.8(
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7.8(
min	67.000000	0.080000	0.000000	0.000000	55.120000	10.3(
25%	17741.250000	25.000000	0.000000	0.000000	77.245000	23.5(
50%	36932.000000	45.000000	0.000000	0.000000	91.885000	28.1(
75%	54682.000000	61.000000	0.000000	0.000000	114.090000	33.1(
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97.6(

Cleaning the data

```
df.isnull().any()
```

```
id                 False
gender             False
age                False
hypertension       False
heart_disease      False
ever_married       False
work_type          False
Residence_type     False
avg_glucose_level  False
bmi                 True
smoking_status     False
stroke             False
dtype: bool
```

```
df=df.dropna()
df.head()
df.isnull().any()
```

```
id                 False
gender             False
age                False
hypertension       False
heart_disease      False
ever_married       False
work_type          False
Residence_type     False
avg_glucose_level  False
bmi                False
smoking_status     False
```

```
stroke               False
dtype: bool
```

```
for col in ['hypertension','heart_disease','stroke']:
  df[col]=df[col].replace([0,1],['NO','YES'])
df
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Resider
0	9046	Male	67.0	NO	YES	Yes	Private	
2	31112	Male	80.0	NO	YES	Yes	Private	
3	60182	Female	49.0	NO	NO	Yes	Private	
4	1665	Female	79.0	YES	NO	Yes	Self-employed	
5	56669	Male	81.0	NO	NO	Yes	Private	
...	...	...	...	...	...	...	...	
5104	14180	Female	13.0	NO	NO	No	children	
5106	44873	Female	81.0	NO	NO	Yes	Self-employed	
5107	19723	Female	35.0	NO	NO	Yes	Self-employed	
5108	37544	Male	51.0	NO	NO	Yes	Private	
5109	44679	Female	44.0	NO	NO	Yes	Govt_iob	

```
#Target %
strokes= df[(df['stroke'] == "YES")]
no_strokes=df[(df['stroke'] == "NO")]
print(len(strokes))
len(no_strokes)
```

```
209
4700
```

```
df.drop_duplicates()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Resider
0	9046	Male	67.0	NO	YES	Yes	Private	
2	31112	Male	80.0	NO	YES	Yes	Private	
3	60182	Female	49.0	NO	NO	Yes	Private	
4	1665	Female	79.0	YES	NO	Yes	Self-employed	
5	56669	Male	81.0	NO	NO	Yes	Private	
...	...	...	...	...	...	...	...	
5104	14180	Female	13.0	NO	NO	No	children	

Classifying attributes

							Self-

```
categorical_cols=[]
numeric_cols=[]
for x in df.columns:
  if df[x].dtype=='object':
    categorical_cols.append(x)
  else:
    numeric_cols.append(x)
print(categorical_cols)

print(numeric_cols)
```

```
['gender', 'hypertension', 'heart_disease', 'ever_married', 'work_type', 'Residence_type
['id', 'age', 'avg_glucose_level', 'bmi']
```

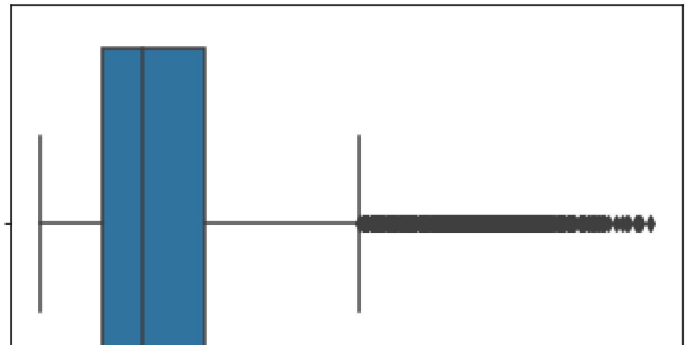
```
type(df)
```

```
pandas.core.frame.DataFrame
```

Outlier Analysis

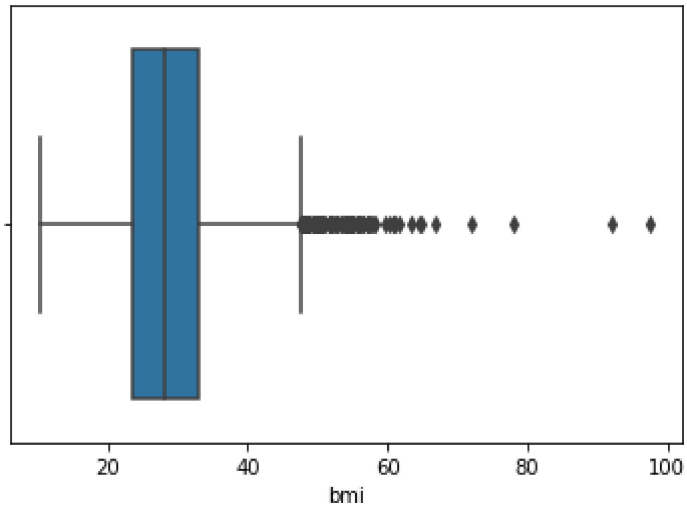
```
#Detecting outliers
sns.boxplot(x=df['avg_glucose_level'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7fcac5b510>
```



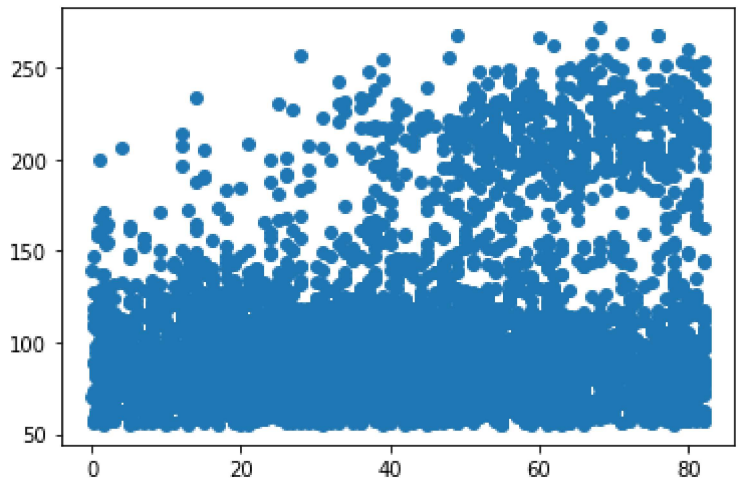
```
sns.boxplot(x=df['bmi'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7fc9bb7e10>
```



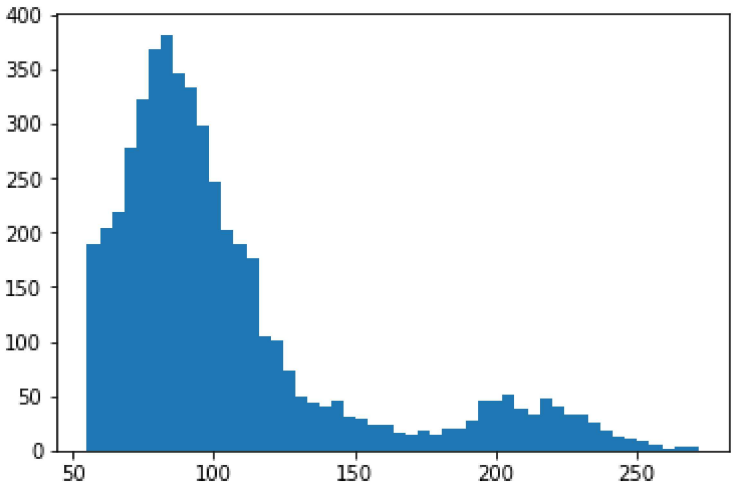

```
#scatter plot
ax=plt.axes()
plt.scatter(df.age,df.avg_glucose_level)
```

```
<matplotlib.collections.PathCollection at 0x7f94dcd31350>
```




```
plt.hist(df.avg_glucose_level,bins=50)
```

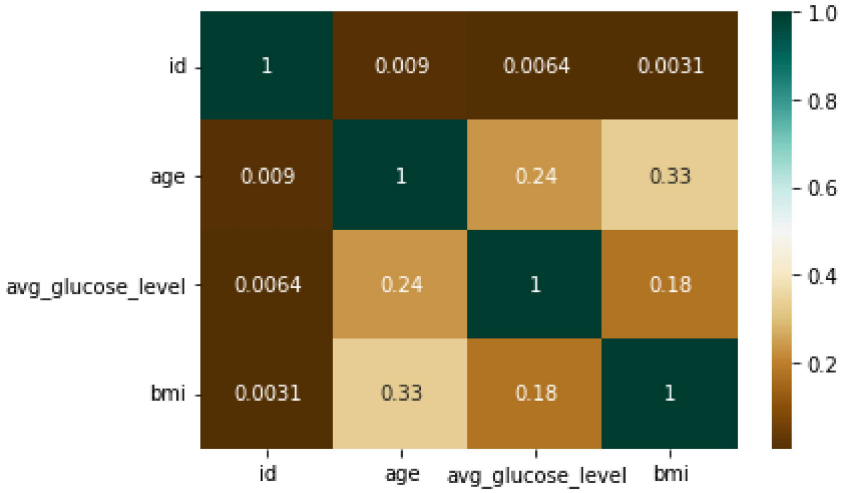
```
(array([189., 204., 220., 278., 322., 368., 382., 347., 334., 298., 246.,
        203., 190., 176., 104., 102.,  73.,  50.,  44.,  41.,  45.,  31.,
         29.,  24.,  23.,  16.,  15.,  18.,  14.,  20.,  20.,  28.,  45.,
         45.,  51.,  39.,  32.,  48.,  41.,  32.,  33.,  25.,  18.,  12.,
         11.,   9.,   5.,   2.,   3.,   4.]),
 array([ 55.12  ,  59.4524,  63.7848,  68.1172,  72.4496,  76.782 ,
         81.1144,  85.4468,  89.7792,  94.1116,  98.444 , 102.7764,
        107.1088, 111.4412, 115.7736, 120.106 , 124.4384, 128.7708,
        133.1032, 137.4356, 141.768 , 146.1004, 150.4328, 154.7652,
        159.0976, 163.43  , 167.7624, 172.0948, 176.4272, 180.7596,
        185.092 , 189.4244, 193.7568, 198.0892, 202.4216, 206.754 ,
        211.0864, 215.4188, 219.7512, 224.0836, 228.416 , 232.7484,
        237.0808, 241.4132, 245.7456, 250.078 , 254.4104, 258.7428,
        263.0752, 267.4076, 271.74  ]),
 <a list of 50 Patch objects>)
```



Providing insights regarding relationships between attributes

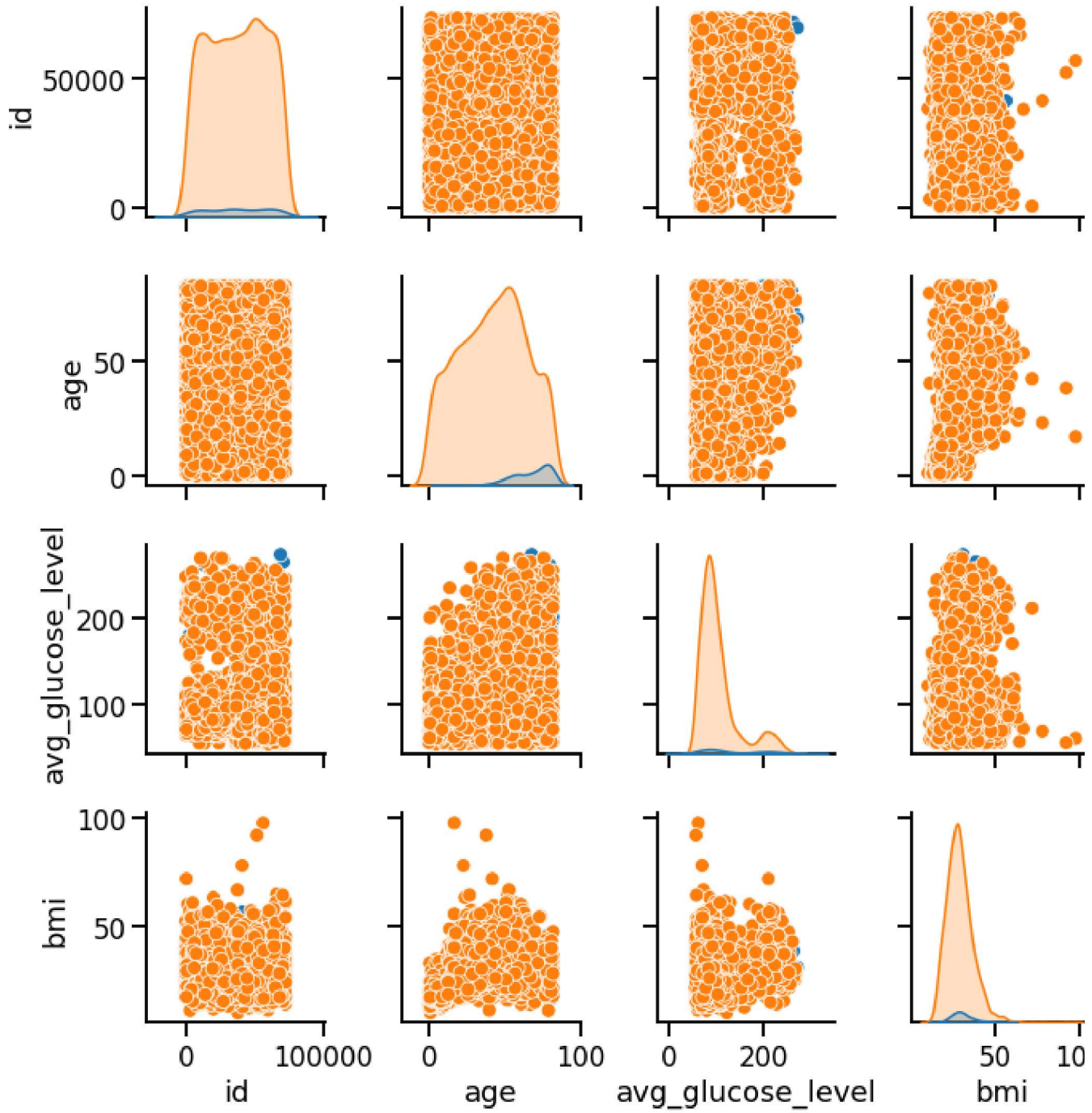
```
c=df.corr()
sns.heatmap(c,cmap="BrBG",annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f94dcbe5d90>
```



```
sns.set_context('talk')
sns.pairplot(df,hue='stroke')
```

```
<seaborn.axisgrid.PairGrid at 0x7f94dcb237d0>
```



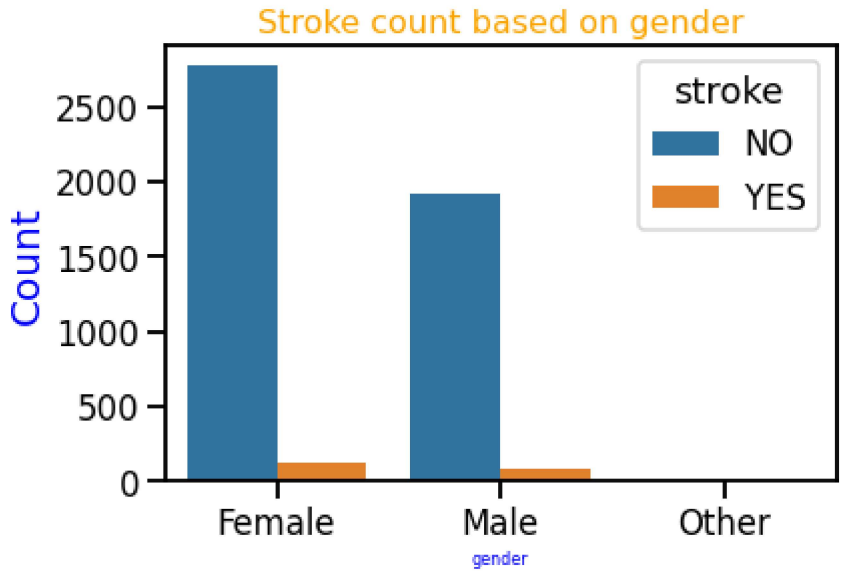

```
def bar_plot(data,x,y='id',hue='stroke',group='stroke' ,title=None):
    """
        function use to plot barplot by grouping the data

    """
    try:
        group=data.groupby([x,group],as_index=False)['id'].count()
        sns.barplot(data=group,x=x,y=y,hue=hue,)
```

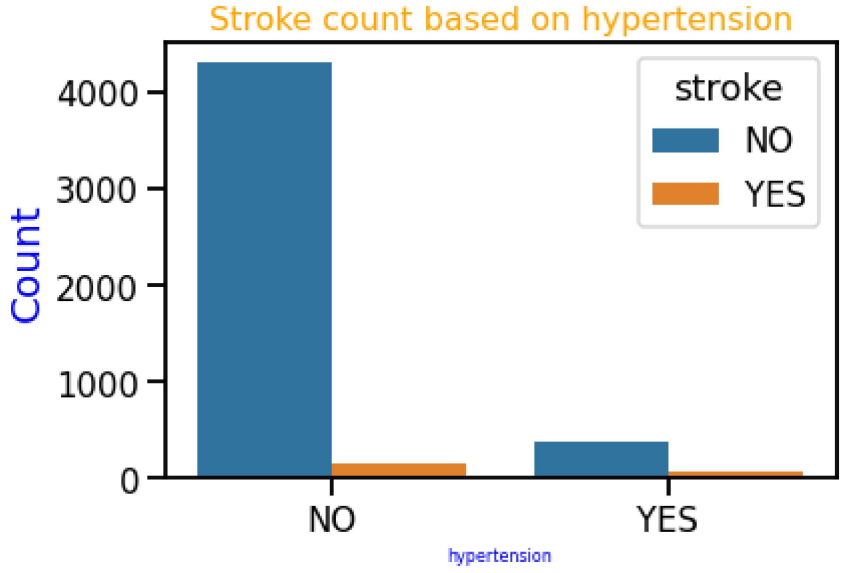
```
            plt.title("Stroke count based on "+ x , fontdict={'size' :16,'color':'orange'})
            plt.ylabel('Count',fontdict={'size':20,'color':'blue'})
            plt.xlabel(x,fontdict={'size':8,'color':'blue'})
            print('-'*100)
            return display(group) ,plt.show()
        except:
            pass

    for col in categorical_cols:
        bar_plot(df,col)
```

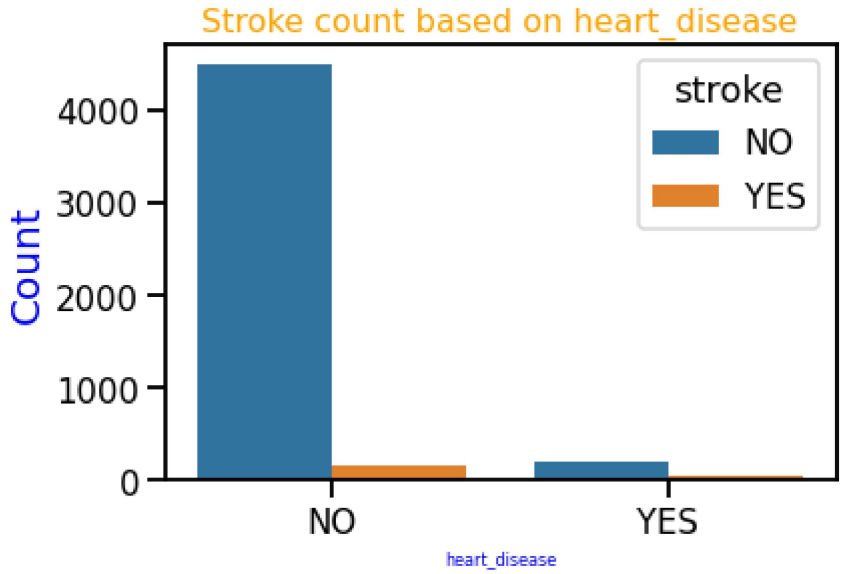
	gender	stroke	id
0	Female	NO	2777
1	Female	YES	120
2	Male	NO	1922
3	Male	YES	89
4	Other	NO	1



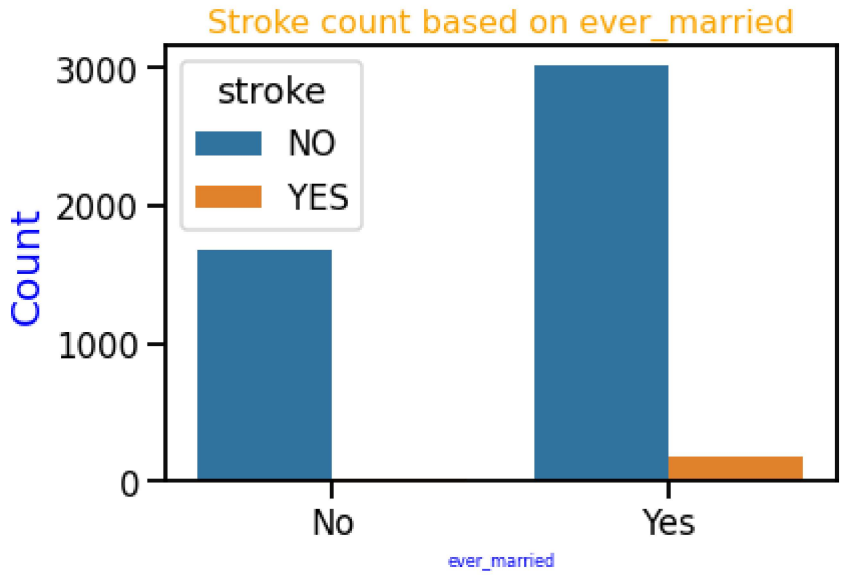
	hypertension	stroke	id
0	NO	NO	4309
1	NO	YES	149
2	YES	NO	391
3	YES	YES	60



	heart_disease	stroke	id
0	NO	NO	4497
1	NO	YES	169
2	YES	NO	203
3	YES	YES	40

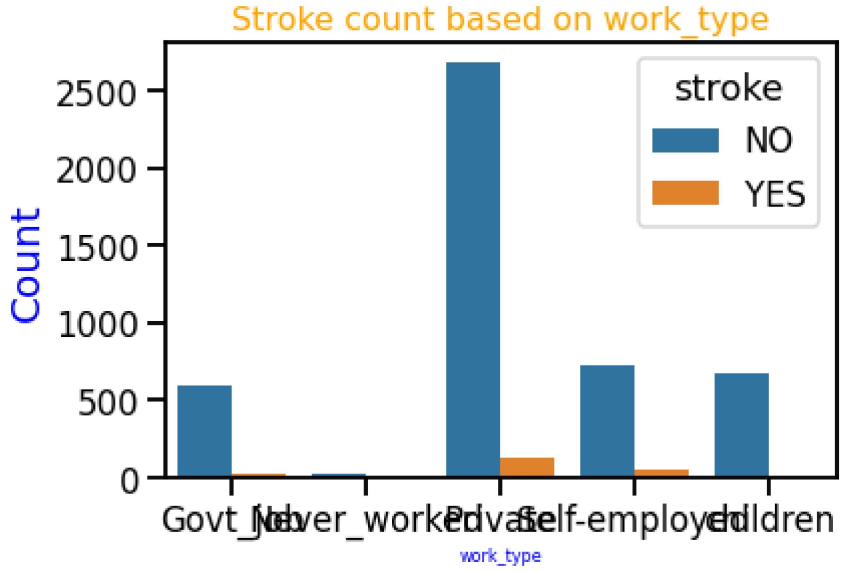


	ever_married	stroke	id
0	No	NO	1682
1	No	YES	23
2	Yes	NO	3018
3	Yes	YES	186

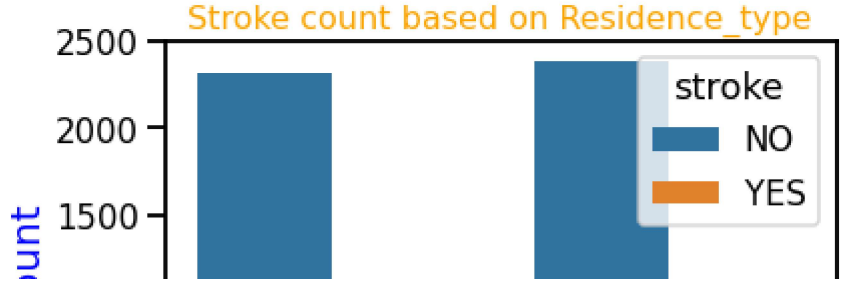


	work_type	stroke	id

0	Govt_job	NO	602
1	Govt_job	YES	28
2	Never_worked	NO	22
3	Private	NO	2684
4	Private	YES	127
5	Self-employed	NO	722
6	Self-employed	YES	53
7	children	NO	670
8	children	YES	1



	Residence_type	stroke	id
0	Rural	NO	2319
1	Rural	YES	100
2	Urban	NO	2381
3	Urban	YES	109

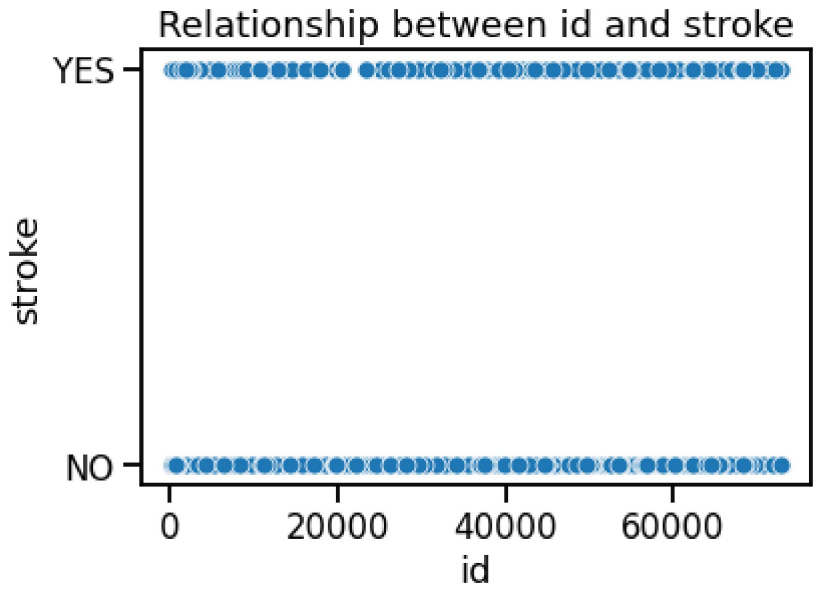


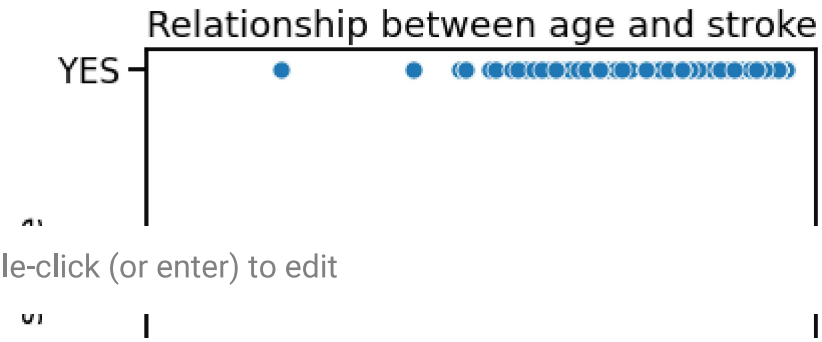

```
def scatter_graph(data,x,y='stroke',title=None):
  sns.scatterplot(data=df,x=x,y=y)
  plt.title("Relationship between "+x+" and "+y)
  print('-'*100*2)
  return plt.show()
```

```
    return plt.show()
```

Residence_type

```
for cols in numeric_cols:
  scatter_graph(df,cols)
```





Double-click (or enter) to edit

Observations

> Age is statistically related to having stroke i.e. more the age more prone you are to get a stroke.

> People in the bmi range of 20 to 40 are most susceptible to stroke.

> Residance area is also statistically related to stroke.

Hypothesis Testing

For age

Null- People above the age of 45 and above are prone to stroke

Alternate - Not prone to stroke

Alpha=0.01

For Average glucose level

Null- People having average glucose levels between 150-160 are less prone to strokes.

Alternate-There's no such thing

Alpha=0.01

For bmi

Null- People with bmi in range of 20-40 are more prone to stroke.

Alternate- There's no such thing everyone is prone to stroke.

Alpha = 0.01

For area

Null- People living in urban areas are prone to stroke.

Alternate- People living in urban areas are not prone to stroke.

Alpha = 0.01

```
import scipy as sp
```

```
strokes_df = df[df['stroke'] == "YES"]
no_strokes_df = df[df['stroke'] == "NO"]
```

```
sp.stats.shapiro(strokes_df['Residence_type'] == "Urban")
```

```
(0.6357665061950684, 6.935258354642178e-21)
```

```
sp.stats.ttest_ind(strokes_df['Residence_type'] == "Urban",no_strokes_df['Residence_type'] ==
```

```
Ttest_indResult(statistic=0.4219485281326302, pvalue=0.6734622116386637)
```

Alpha = 0.01

p value=0.67

Since pvalue > alpha

we fail to reject the null hypothesis which means yes urban people are more prone to have a stroke.

So there seems to be a statistical relationship between Residance area and heart stroke.