In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

Columns

age: age of primary beneficiary

sex: insurance contractor gender, female, male

bmi: BMI is Body Mass Index is a ratio of weight and height. It can be defined as the weight in kilograms, divided by the square of the height in metres. This ratio is used to classify whether a person is underweight, overweight or obese.

children: Number of children covered by health insurance.

smoker: Smoking

region: the beneficiary's residential area in the US, northeast, southeast, southwest, northwest.

charges: Individual medical costs billed by health insurance

1 df=pd.read_csv("insurance.csv")

Problem statement

There is a insurance company and this company wants to create system that can predict what is the medical insurance cost of a person.

In [2]:
```
df=pd.read_csv("insurance.csv")
```

In [3]:
```
df
```

Out[3]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	male	30.970	3	no	northwest	10600.54830
1334	18	female	31.920	0	no	northeast	2205.98080
1335	18	female	36.850	0	no	southeast	1629.83350
1336	21	female	25.800	0	no	southwest	2007.94500
1337	61	female	29.070	0	yes	northwest	29141.36030

1338 rows × 7 columns

In [4]:
```
df.shape
```

Out[4]:
```
(1338, 7)
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dtype
='object')
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1338 non-null   int64  
 1   sex       1338 non-null   object 
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64  
 4   smoker    1338 non-null   object 
 5   region    1338 non-null   object 
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [8]:
```
df.duplicated().sum()
```

Out[8]:
```
1
```

In [9]:
```
df[df.duplicated(keep=False)]
```

Out[9]:

	age	sex	bmi	children	smoker	region	charges
195	19	male	30.59	0	no	northwest	1639.5631
581	19	male	30.59	0	no	northwest	1639.5631

In [10]:
```
df2=df.drop_duplicates()  ## use this function for drop duplicates value
```

In [11]:
```
df2
```

Out[11]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	male	30.970	3	no	northwest	10600.54830
1334	18	female	31.920	0	no	northeast	2205.98080
1335	18	female	36.850	0	no	southeast	1629.83350
1336	21	female	25.800	0	no	southwest	2007.94500
1337	61	female	29.070	0	yes	northwest	29141.36030

1337 rows × 7 columns

In [12]:
```
df2.shape
```

Out[12]:
```
(1337, 7)
```

In [13]:
```
df2.describe()
```

Out[13]:

	age	bmi	children	charges
count	1337.000000	1337.000000	1337.000000	1337.000000
mean	39.222139	30.663452	1.095737	13279.121487
std	14.044333	6.100468	1.205571	12110.359656
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.290000	0.000000	4746.344000
50%	39.000000	30.400000	1.000000	9386.161300
75%	51.000000	34.700000	2.000000	16657.717450
max	64.000000	53.130000	5.000000	63770.428010

Analyse Age colum

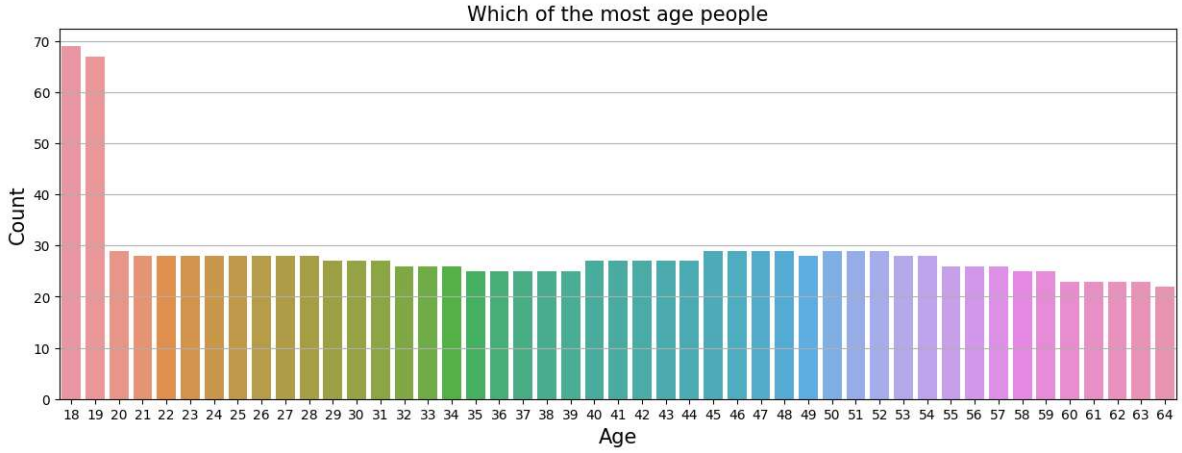
In [14]:
```
df2["age"].unique()
```

Out[14]:
```
array([19, 18, 28, 33, 32, 31, 46, 37, 60, 25, 62, 23, 56, 27, 52, 30, 34,
       59, 63, 55, 22, 26, 35, 24, 41, 38, 36, 21, 48, 40, 58, 53, 43, 64,
       20, 61, 44, 57, 29, 45, 54, 49, 47, 51, 42, 50, 39], dtype=int64)
```

In [15]:
```
df2.age.value_counts()
```

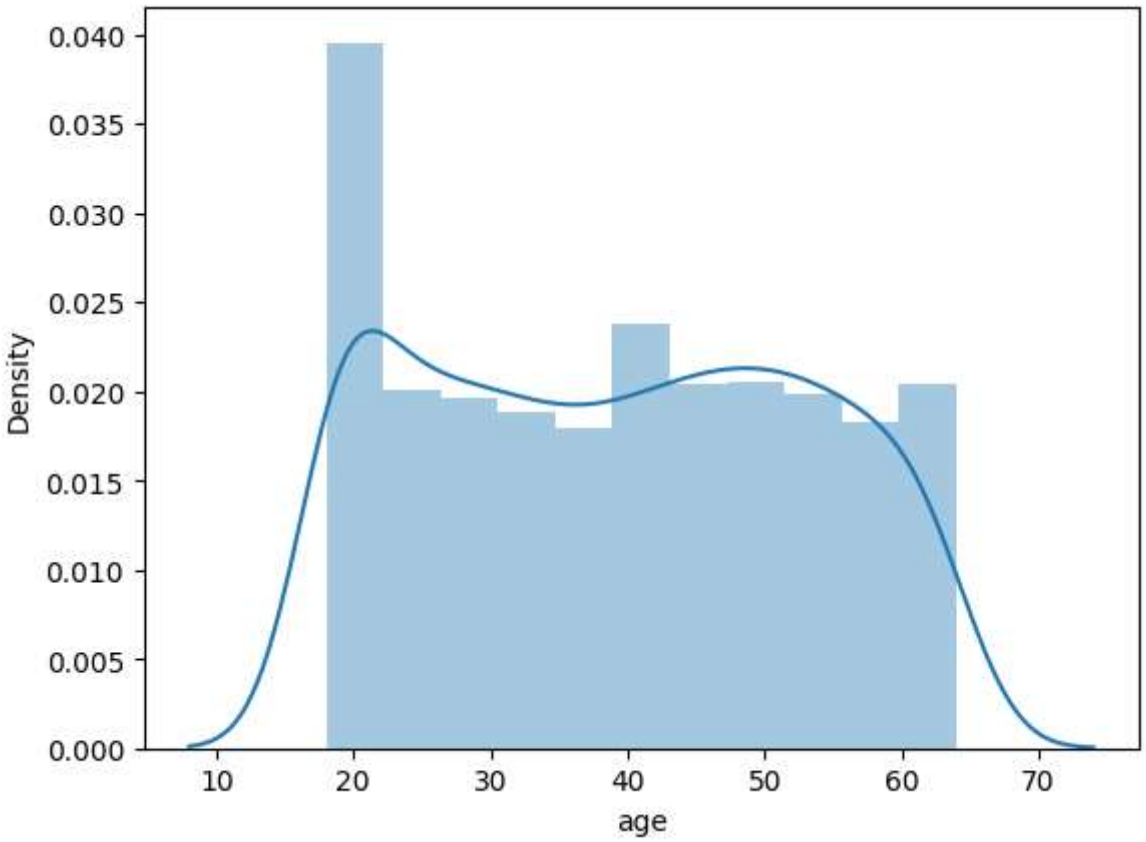
Out[15]:
```
age
18    69
19    67
50    29
51    29
47    29
46    29
45    29
20    29
48    29
52    29
22    28
49    28
54    28
53    28
21    28
26    28
24    28
25    28
28    28
27    28
23    28
43    27
29    27
30    27
41    27
42    27
44    27
31    27
40    27
32    26
33    26
56    26
34    26
55    26
57    26
37    25
59    25
58    25
36    25
38    25
35    25
39    25
61    23
60    23
63    23
62    23
64    22
Name: count, dtype: int64
```

In [90]:
```
# que = Which age group has more people ?

plt.figure(figsize=(15,5))
sns.countplot(x="age",data=df2)
plt.title("Which of the most age people",fontsize=15)
plt.xlabel("Age",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
```



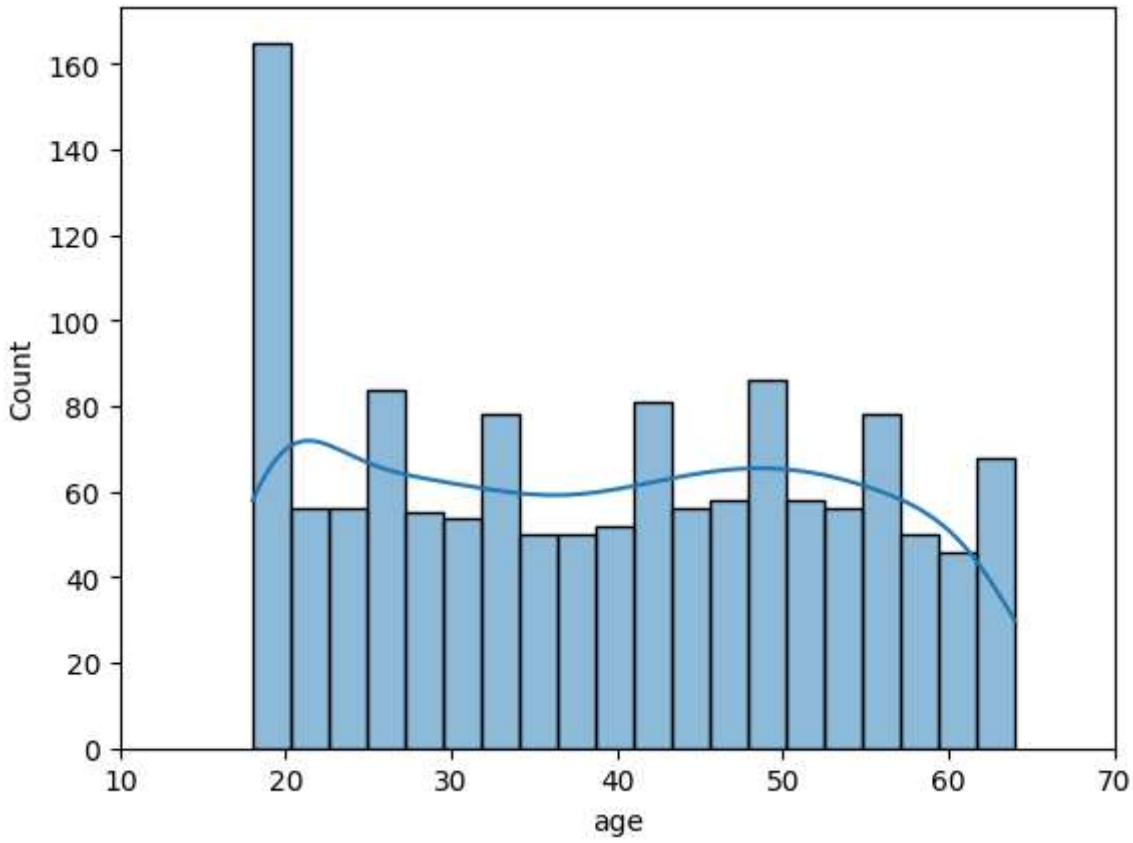
In [17]:
```
## distplot for Age column

sns.distplot(df2["age"])
```

Out[17]: <Axes: xlabel='age', ylabel='Density'>



In [18]:
```
## Histogram for Age column

sns.histplot(df2["age"], kde = True, bins = 20)
plt.xlim([10,70])
```

Out[18]: (10.0, 70.0)



Observation:

* People of 18 and 19 age are the most

Analyse sex colum

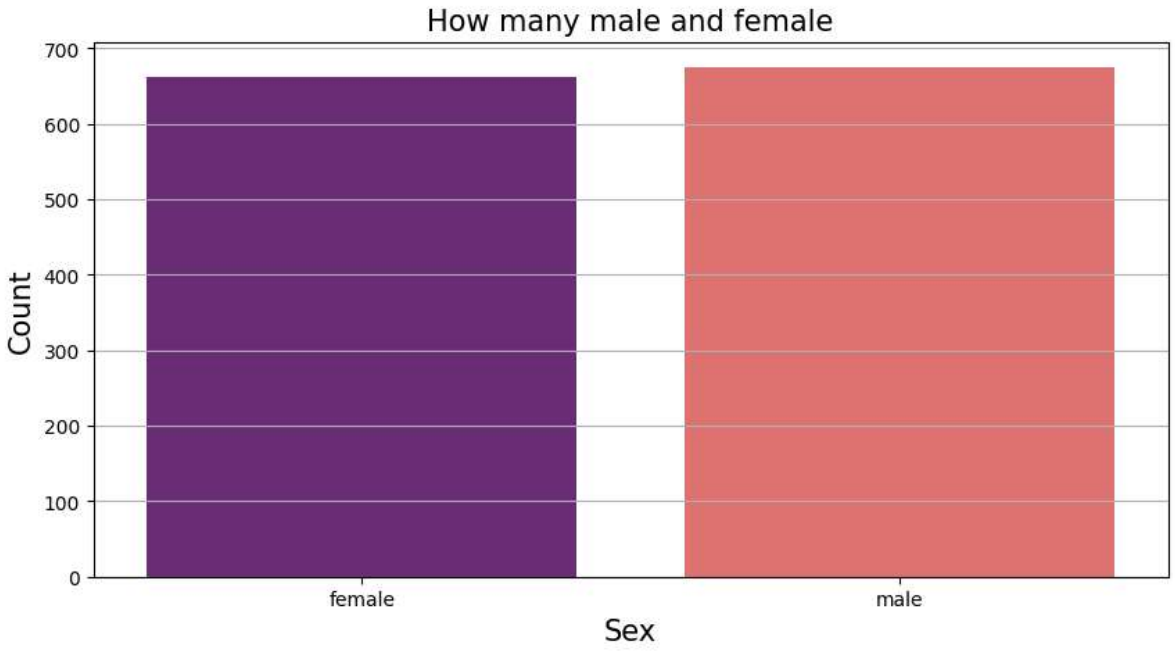
In [19]:
```
df["sex"].unique()
```

Out[19]: array(['female', 'male'], dtype=object)

In [20]:
```
df.sex.value_counts()
```

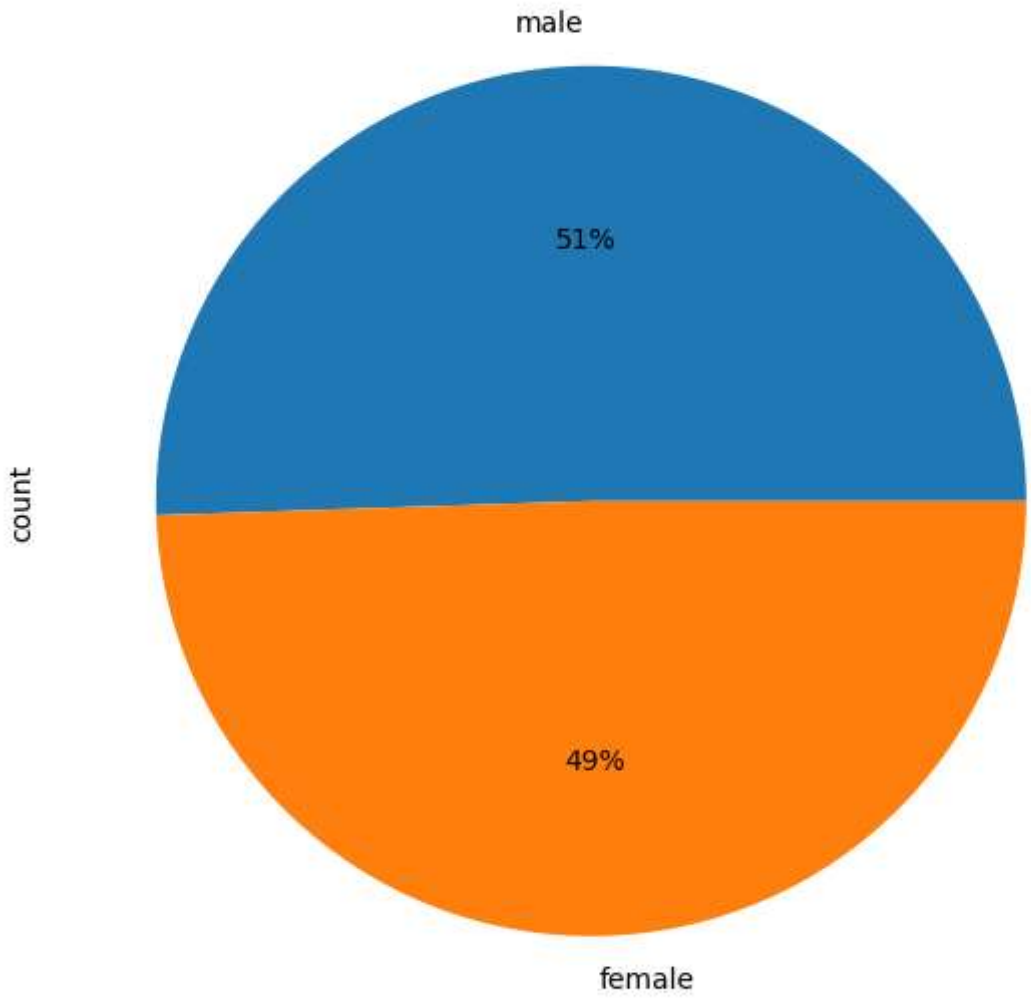
Out[20]:
```
sex
male      676
female    662
Name: count, dtype: int64
```

In [21]:
```
plt.figure(figsize=(10,5))
sns.countplot(x="sex",data=df2,palette="magma")
plt.title("How many male and female",fontsize=15)
plt.xlabel("Sex",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
```



In [22]:
```
plt.figure(figsize=(7,10))
df["sex"].value_counts().plot.pie(autopct="%.0f%%")
```

Out[22]: <Axes: ylabel='count'>



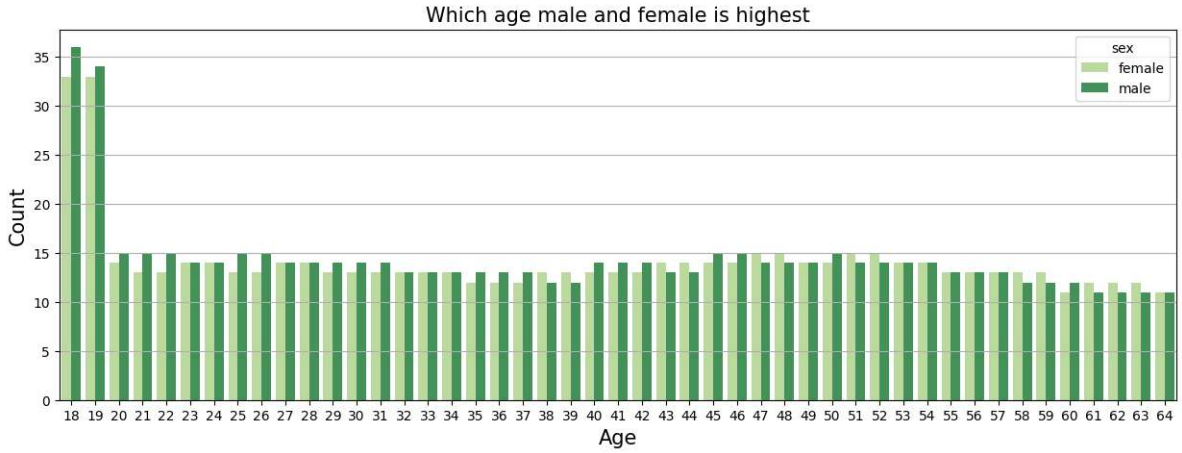
Observation:

* There is more male than female
* The percentage of male is 51
* The percentage of frmale is 49

Compare colum age and sex

Compare relation between in which age highest male and female

In [23]:
```
plt.figure(figsize=(15,5))
sns.countplot(x="age",hue="sex",data=df2,palette="YlGn")
plt.title("Which age male and female is highest",fontsize=15)
plt.xlabel("Age",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
```



Observation:

* Most of male and female are 18 to 19 years old.

Analyse bmi colum

In [24]:
```
df["bmi"].unique()
```

Out[24]:
```
array([27.9  , 33.77 , 33.   , 22.705, 28.88 , 25.74 , 33.44 , 27.74 ,
       29.83 , 25.84 , 26.22 , 26.29 , 34.4  , 39.82 , 42.13 , 24.6  ,
       30.78 , 23.845, 40.3  , 35.3  , 36.005, 32.4  , 34.1  , 31.92 ,
       28.025, 27.72 , 23.085, 32.775, 17.385, 36.3  , 35.6  , 26.315,
       28.6  , 28.31 , 36.4  , 20.425, 32.965, 20.8  , 36.67 , 39.9  ,
       26.6  , 36.63 , 21.78 , 30.8  , 37.05 , 37.3  , 38.665, 34.77 ,
       24.53 , 35.2  , 35.625, 33.63 , 28.   , 34.43 , 28.69 , 36.955,
       31.825, 31.68 , 22.88 , 37.335, 27.36 , 33.66 , 24.7  , 25.935,
       22.42 , 28.9  , 39.1  , 36.19 , 23.98 , 24.75 , 28.5  , 28.1  ,
       32.01 , 27.4  , 34.01 , 29.59 , 35.53 , 39.805, 26.885, 38.285,
       37.62 , 41.23 , 34.8  , 22.895, 31.16 , 27.2  , 26.98 , 39.49 ,
       24.795, 31.3  , 38.28 , 19.95 , 19.3  , 31.6  , 25.46 , 30.115,
       29.92 , 27.5  , 28.4  , 30.875, 27.94 , 35.09 , 29.7  , 35.72 ,
       32.205, 28.595, 49.06 , 27.17 , 23.37 , 37.1  , 23.75 , 28.975,
       31.35 , 33.915, 28.785, 28.3  , 37.4  , 17.765, 34.7  , 26.505,
       22.04 , 35.9  , 25.555, 28.05 , 25.175, 31.9  , 36.   , 32.49 ,
       25.3  , 29.735, 38.83 , 30.495, 37.73 , 37.43 , 24.13 , 37.145,
       39.52 , 24.42 , 27.83 , 36.85 , 39.6  , 29.8  , 29.64 , 28.215,
       37.   , 33.155, 18.905, 41.47 , 30.3  , 15.96 , 33.345, 37.7  ,
       27.835, 29.2  , 26.41 , 30.69 , 41.895, 30.9  , 32.2  , 32.11 ,
       31.57 , 26.2  , 30.59 , 32.8  , 18.05 , 39.33 , 32.23 , 24.035,
       36.08 , 22.3  , 26.4  , 31.8  , 26.73 , 23.1  , 23.21 , 33.7  ,
       33.25 , 24.64 , 33.88 , 38.06 , 41.91 , 31.635, 36.195, 17.8  ,
       24.51 , 22.22 , 38.39 , 29.07 , 22.135, 26.8  , 30.02 , 35.86 ,
       20.9  , 17.29 , 34.21 , 25.365, 40.15 , 24.415, 25.2  , 26.84 ,
       24.32 , 42.35 , 19.8  , 32.395, 30.2  , 29.37 , 34.2  , 27.455,
       27.55 , 20.615, 24.3  , 31.79 , 21.56 , 28.12 , 40.565, 27.645,
       31.2  , 26.62 , 48.07 , 36.765, 33.4  , 45.54 , 28.82 , 22.99 ,
       27.7  , 25.41 , 34.39 , 22.61 , 37.51 , 38.   , 33.33 , 34.865,
       33.06 , 35.97 , 31.4  , 25.27 , 40.945, 34.105, 36.48 , 33.8  ,
       36.7  , 36.385, 34.5  , 32.3  , 27.6  , 29.26 , 35.75 , 23.18 ,
       25.6  , 35.245, 43.89 , 20.79 , 30.5  , 21.7  , 21.89 , 24.985,
       32.015, 30.4  , 21.09 , 22.23 , 32.9  , 24.89 , 31.46 , 17.955,
       30.685, 43.34 , 39.05 , 30.21 , 31.445, 19.855, 31.02 , 38.17 ,
       20.6  , 47.52 , 20.4  , 38.38 , 24.31 , 23.6  , 21.12 , 30.03 ,
       17.48 , 20.235, 17.195, 23.9  , 35.15 , 35.64 , 22.6  , 39.16 ,
       27.265, 29.165, 16.815, 33.1  , 26.9  , 33.11 , 31.73 , 46.75 ,
       29.45 , 32.68 , 33.5  , 43.01 , 36.52 , 26.695, 25.65 , 29.6  ,
       38.6  , 23.4  , 46.53 , 30.14 , 30.   , 38.095, 28.38 , 28.7  ,
       33.82 , 24.09 , 32.67 , 25.1  , 32.56 , 41.325, 39.5  , 34.3  ,
       31.065, 21.47 , 25.08 , 43.4  , 25.7  , 27.93 , 39.2  , 26.03 ,
       30.25 , 28.93 , 35.7  , 35.31 , 31.   , 44.22 , 26.07 , 25.8  ,
       39.425, 40.48 , 38.9  , 47.41 , 35.435, 46.7  , 46.2  , 21.4  ,
       23.8  , 44.77 , 32.12 , 29.1  , 37.29 , 43.12 , 36.86 , 34.295,
       23.465, 45.43 , 23.65 , 20.7  , 28.27 , 35.91 , 29.   , 19.57 ,
       31.13 , 21.85 , 40.26 , 33.725, 29.48 , 32.6  , 37.525, 23.655,
       37.8  , 19.   , 21.3  , 33.535, 42.46 , 38.95 , 36.1  , 29.3  ,
       39.7  , 38.19 , 42.4  , 34.96 , 42.68 , 31.54 , 29.81 , 21.375,
       40.81 , 17.4  , 20.3  , 18.5  , 26.125, 41.69 , 24.1  , 36.2  ,
       40.185, 39.27 , 34.87 , 44.745, 29.545, 23.54 , 40.47 , 40.66 ,
       36.6  , 35.4  , 27.075, 28.405, 21.755, 40.28 , 30.1  , 32.1  ,
       23.7  , 35.5  , 29.15 , 27.   , 37.905, 22.77 , 22.8  , 34.58 ,
       27.1  , 19.475, 26.7  , 34.32 , 24.4  , 41.14 , 22.515, 41.8  ,
       26.18 , 42.24 , 26.51 , 35.815, 41.42 , 36.575, 42.94 , 21.01 ,
       24.225, 17.67 , 31.5  , 31.1  , 32.78 , 32.45 , 50.38 , 47.6  ,
       25.4  , 29.9  , 43.7  , 24.86 , 28.8  , 29.5  , 29.04 , 38.94 ,
       44.   , 20.045, 40.92 , 35.1  , 29.355, 32.585, 32.34 , 39.8  ,
```

```
24.605, 33.99 , 28.2  , 25.   , 33.2  , 23.2  , 20.1  , 32.5  ,
37.18 , 46.09 , 39.93 , 35.8  , 31.255, 18.335, 42.9  , 26.79 ,
39.615, 25.9  , 25.745, 28.16 , 23.56 , 40.5  , 35.42 , 39.995,
34.675, 20.52 , 23.275, 36.29 , 32.7  , 19.19 , 20.13 , 23.32 ,
45.32 , 34.6  , 18.715, 21.565, 23.   , 37.07 , 52.58 , 42.655,
21.66 , 32.   , 18.3  , 47.74 , 22.1  , 19.095, 31.24 , 29.925,
20.35 , 25.85 , 42.75 , 18.6  , 23.87 , 45.9  , 21.5  , 30.305,
44.88 , 41.1  , 40.37 , 28.49 , 33.55 , 40.375, 27.28 , 17.86 ,
33.3  , 39.14 , 21.945, 24.97 , 23.94 , 34.485, 21.8  , 23.3  ,
36.96 , 21.28 , 29.4  , 27.3  , 37.9  , 37.715, 23.76 , 25.52 ,
27.61 , 27.06 , 39.4  , 34.9  , 22.   , 30.36 , 27.8  , 53.13 ,
39.71 , 32.87 , 44.7  , 30.97 ])
```

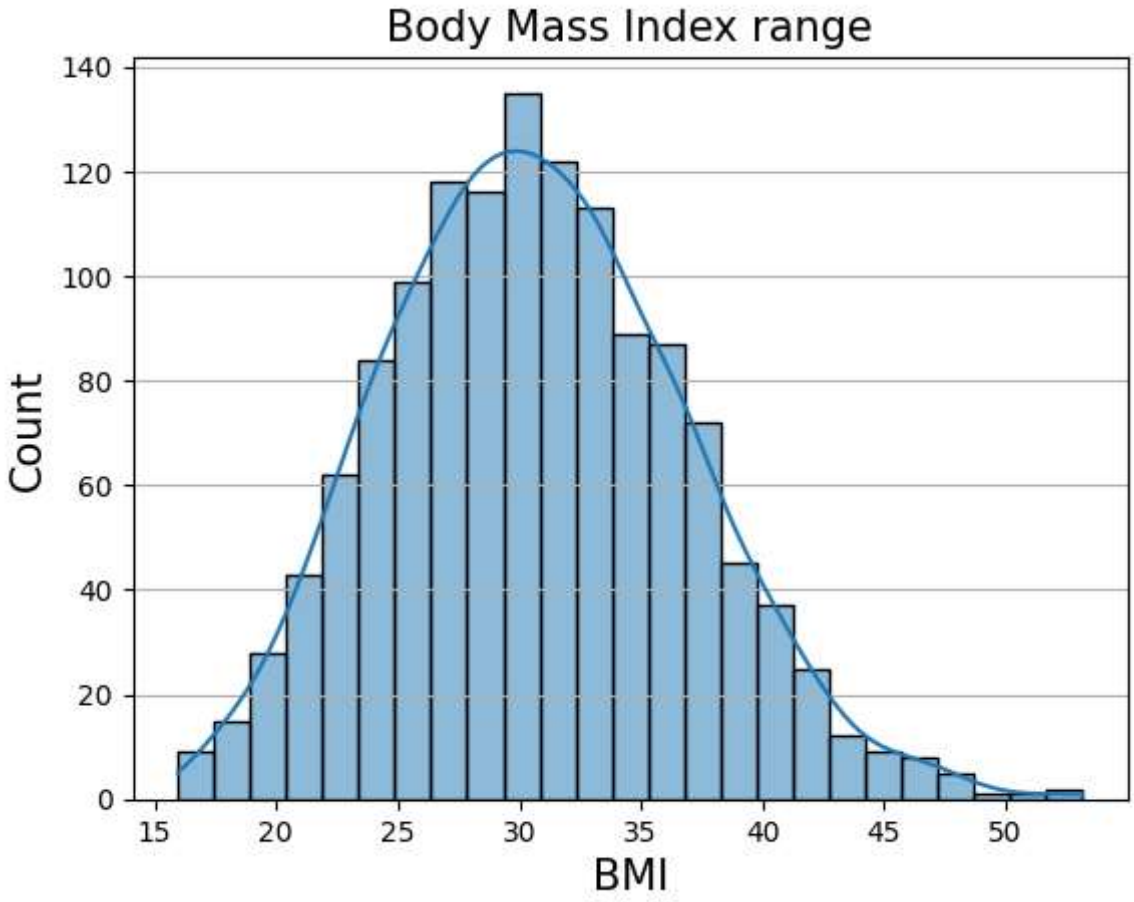
In [25]:
```
df.bmi.value_counts()
```

Out[25]:
```
bmi
32.300    13
28.310     9
30.495     8
30.875     8
31.350     8
          ..
46.200     1
23.800     1
44.770     1
32.120     1
30.970     1
Name: count, Length: 548, dtype: int64
```

In [26]:
```
df["bmi"].min(),df["bmi"].max()
```

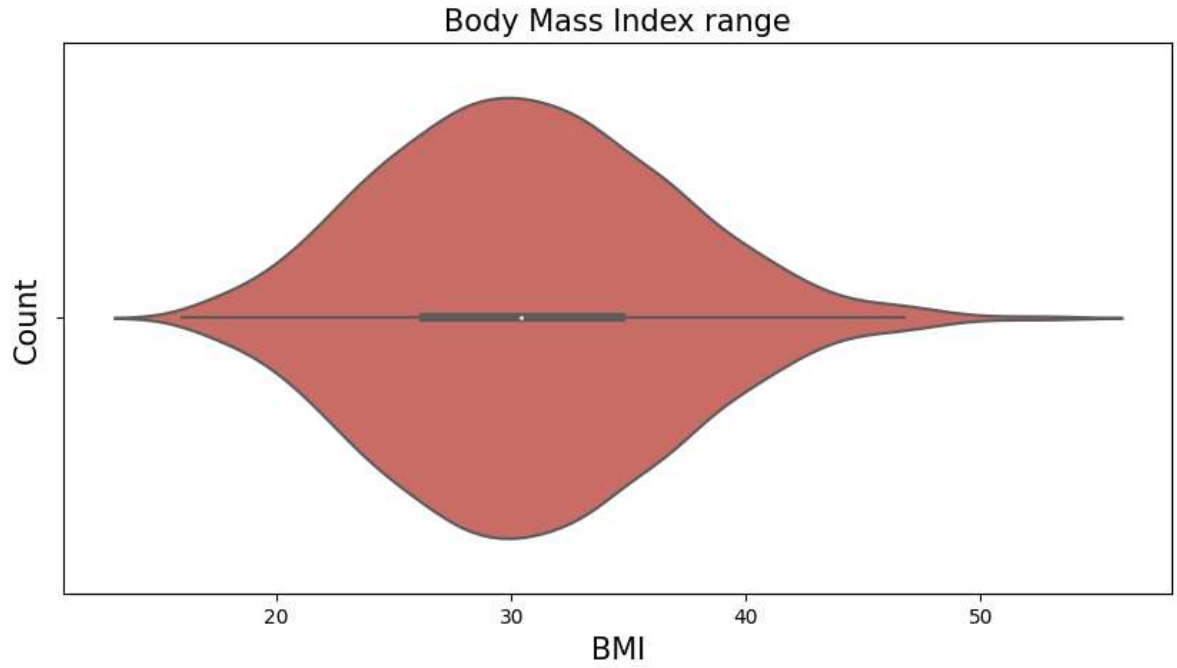
Out[26]:
```
(15.96, 53.13)
```

In [27]:
```
## Using histogram for bmi colum

sns.histplot(df2["bmi"],kde=True)
plt.title("Body Mass Index range",fontsize=15)
plt.xlabel("BMI",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
```



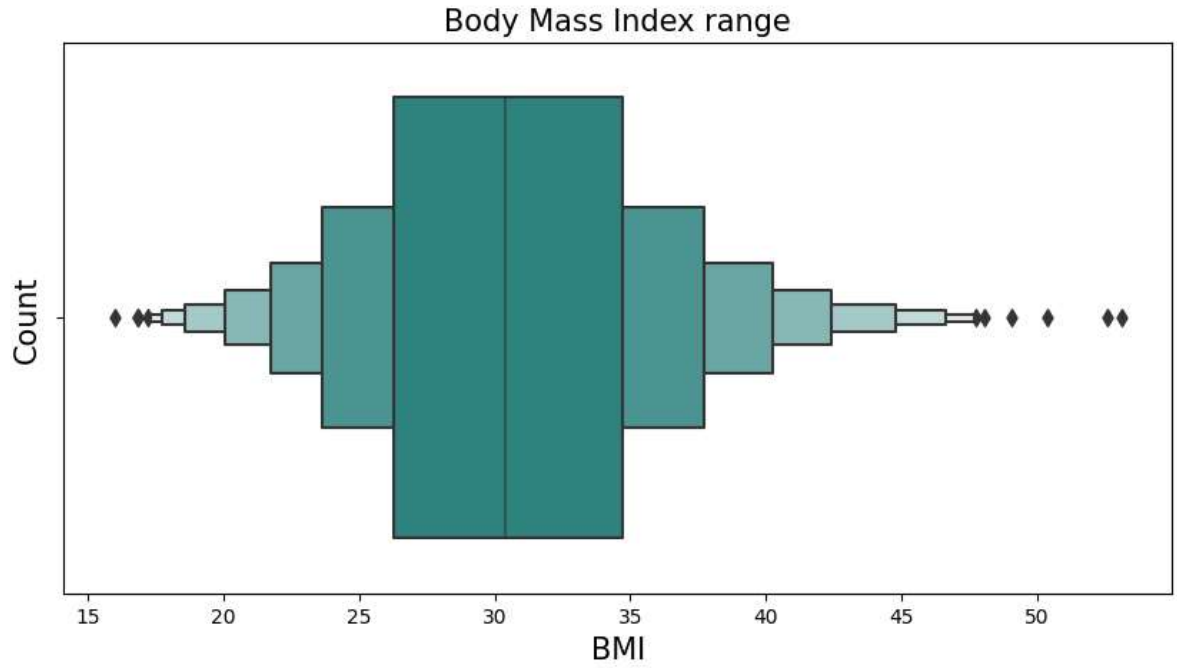
In [28]:
```
## Using Violinplot for bmi colum

plt.figure(figsize=(10,5))
sns.violinplot(x="bmi",data=df2,palette = "hls")
plt.title("Body Mass Index range",fontsize=15)
plt.xlabel("BMI",fontsize=15)
plt.ylabel("Count",fontsize=15)
```

Out[28]:
```
Text(0, 0.5, 'Count')
```



In [29]:
```
## Using Boxenplot for bmi colum

plt.figure(figsize=(10,5))
sns.boxenplot(x="bmi",data=df2,palette = "viridis")
plt.title("Body Mass Index range",fontsize=15)
plt.xlabel("BMI",fontsize=15)
plt.ylabel("Count",fontsize=15)
```

Out[29]:
```
Text(0, 0.5, 'Count')
```



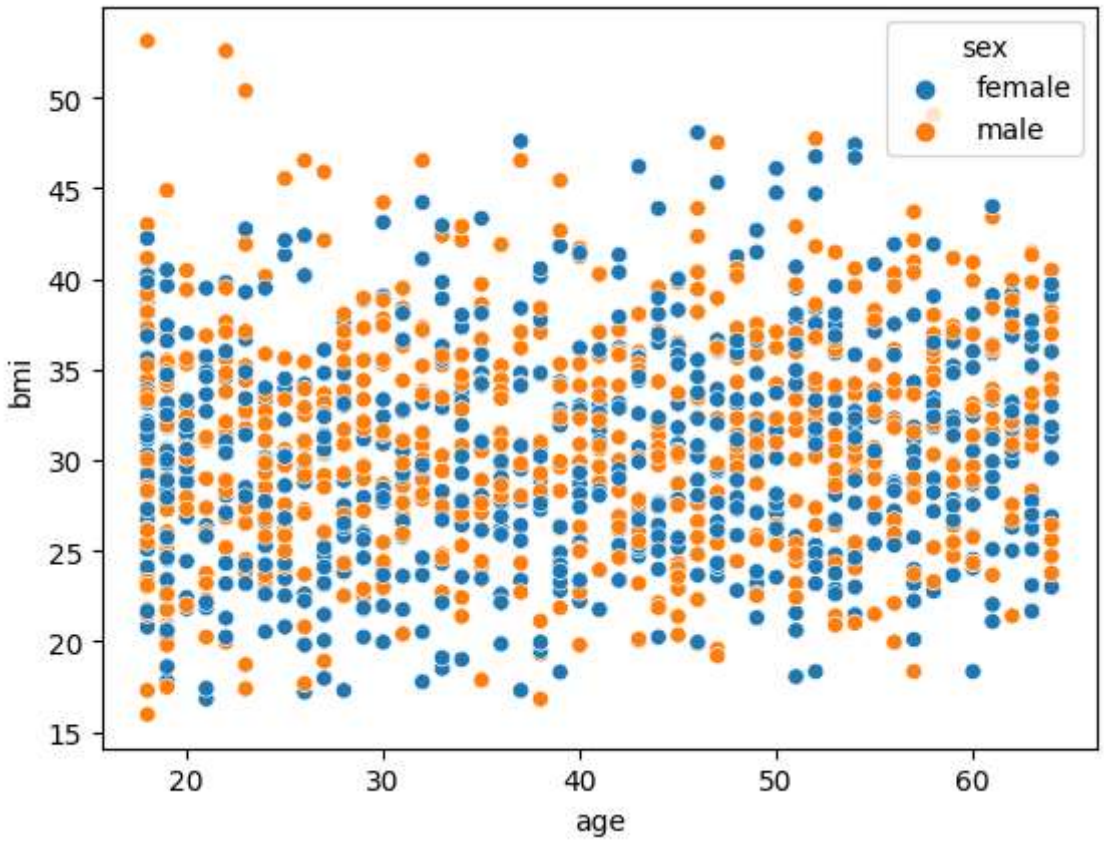
Observation:

* Most of the Body Mass Index (bmi) in between 25 to 35

Compare colum bmi , sex and age

In [30]:
```
sns.scatterplot(x="age",y="bmi",hue="sex",data=df2)
```

Out[30]:
```
<Axes: xlabel='age', ylabel='bmi'>
```



Analyse children colum

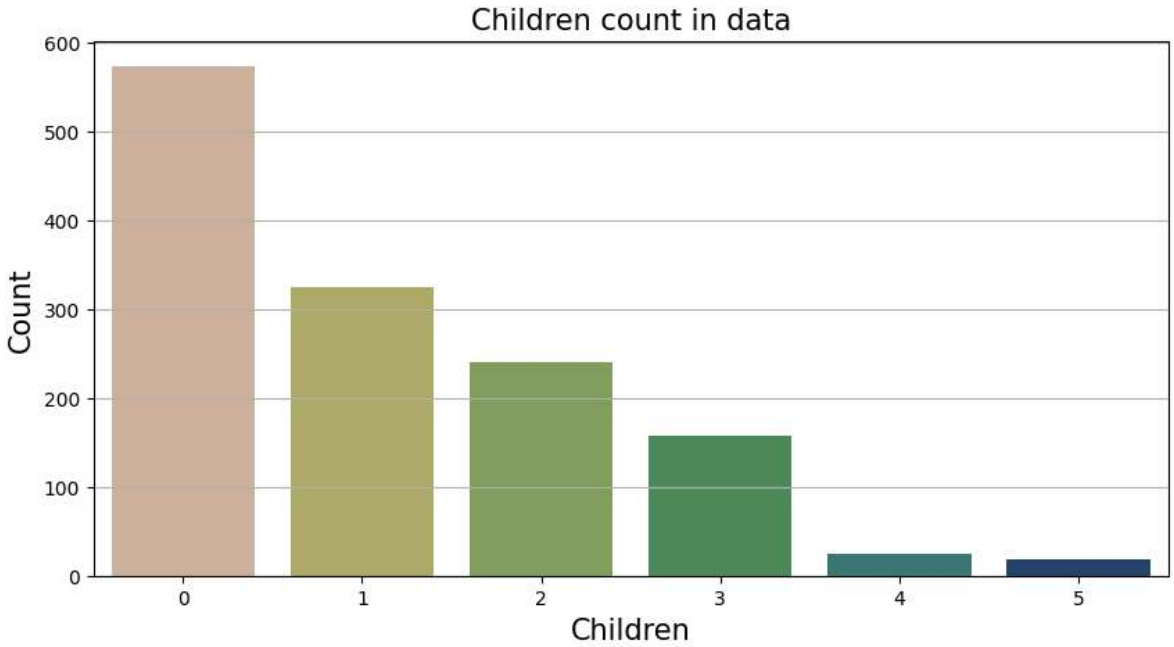
In [31]:
```
df["children"].unique()
```

Out[31]:
```
array([0, 1, 3, 2, 5, 4], dtype=int64)
```

In [32]:
```
df.children.value_counts()
```

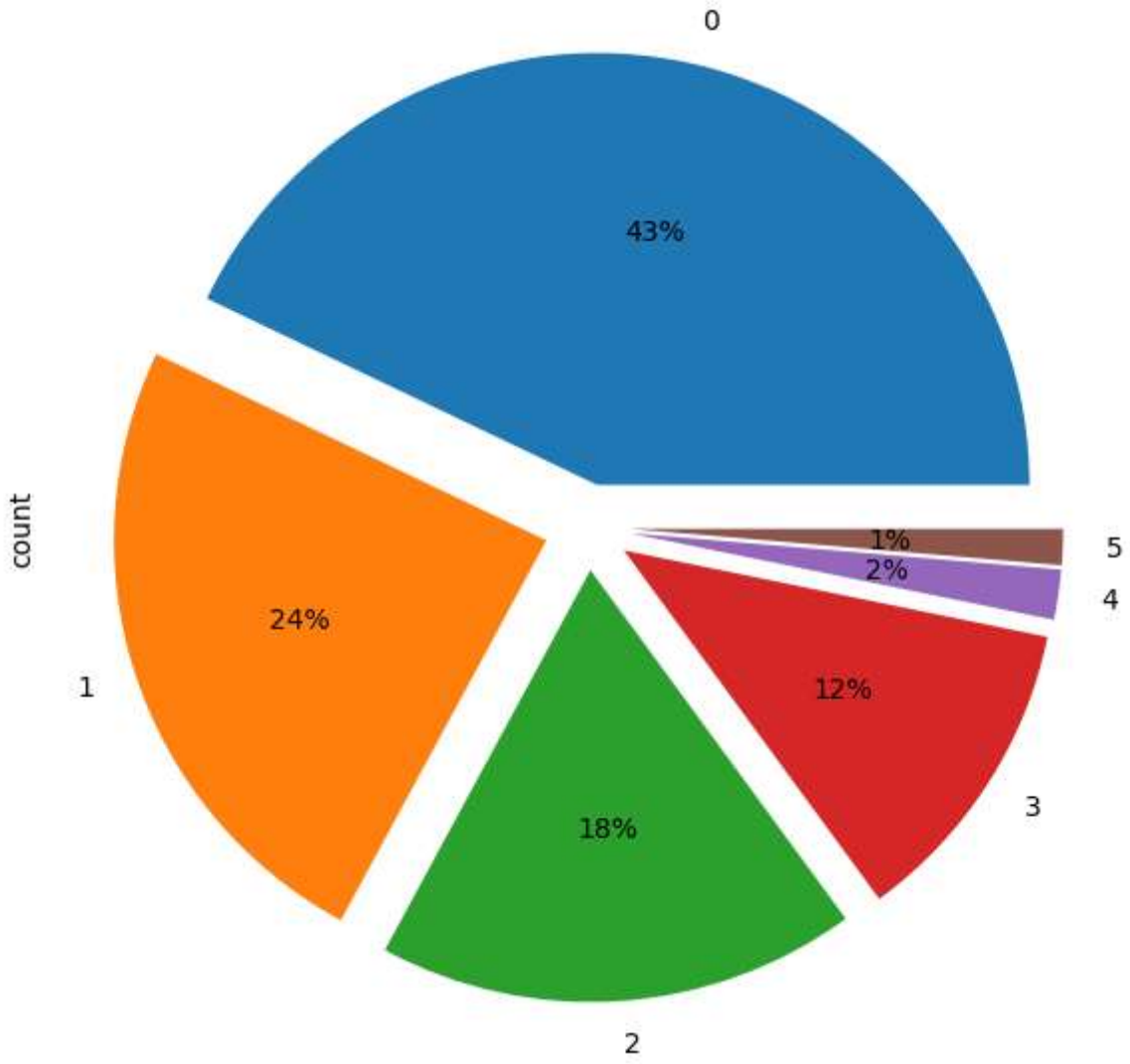
Out[32]:
```
children
0    574
1    324
2    240
3    157
4     25
5     18
Name: count, dtype: int64
```

In [33]:
```
## countplot for childern colum

plt.figure(figsize=(10,5))
sns.countplot(x="children",data=df2,palette="gist_earth_r")
plt.title("Children count in data",fontsize=15)
plt.xlabel("Children",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
```



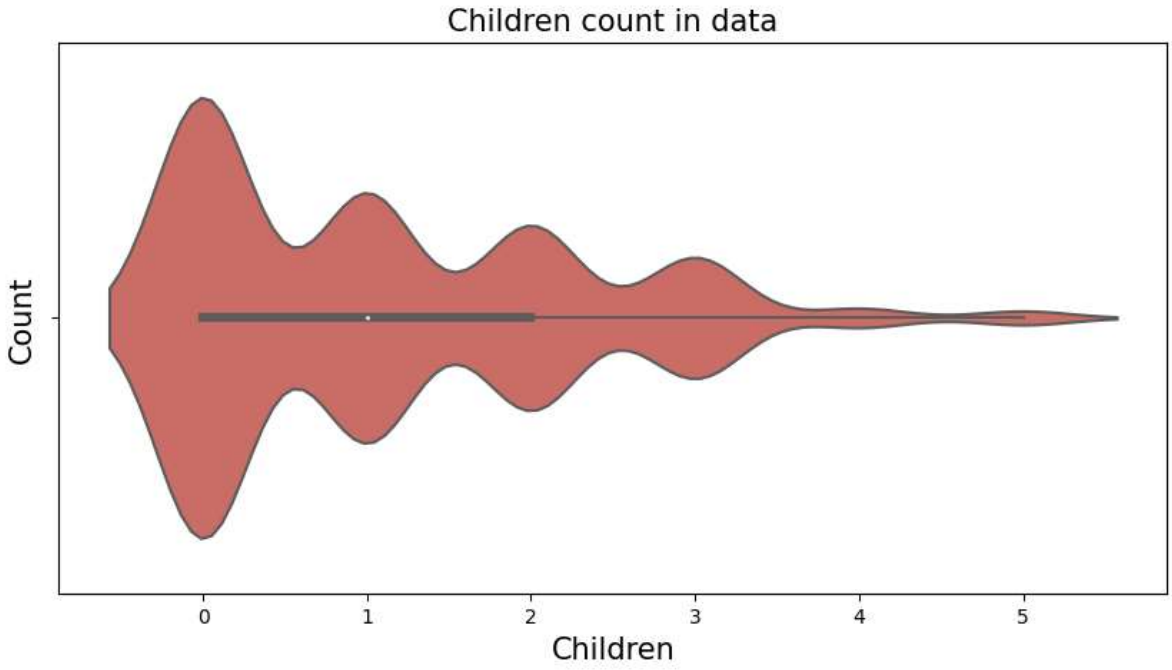
In [34]:
```
## piechart for childern colum

plt.figure(figsize=(7,15))
df["children"].value_counts().plot.pie(autopct="%.0f%%",explode=(.1,.1,.1,.1,.
```

Out[34]: <Axes: ylabel='count'>



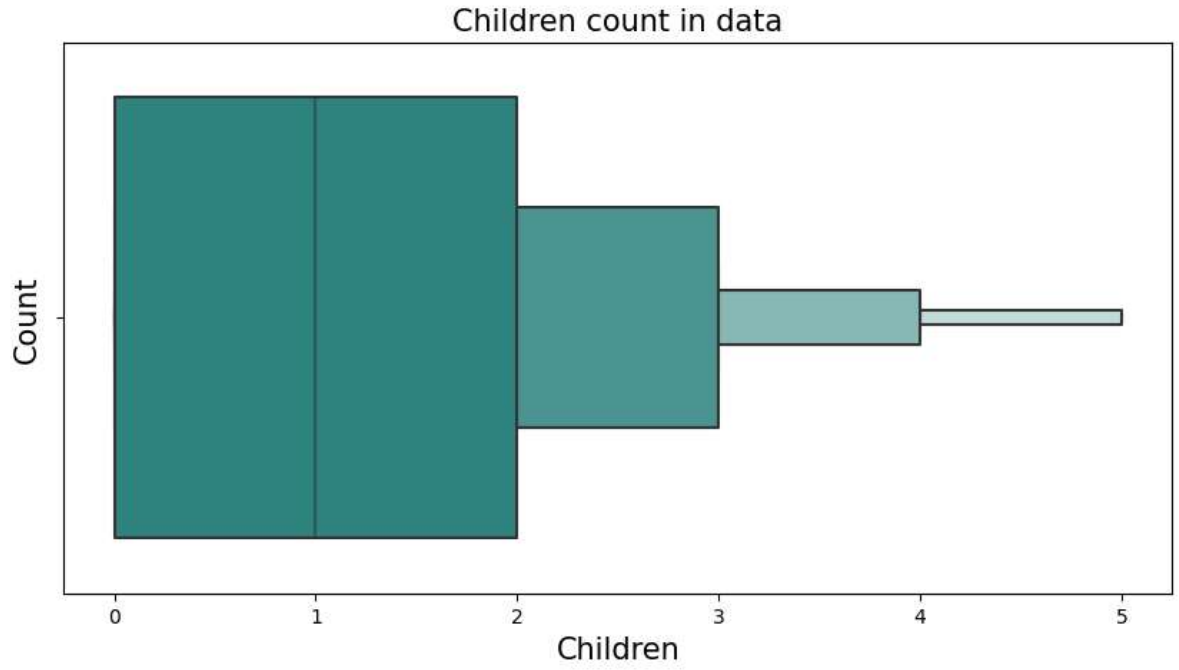
In [35]:
```
## violinplot for childern colum

plt.figure(figsize=(10,5))
sns.violinplot(x="children",data=df2,palette = "hls")
plt.title("Children count in data",fontsize=15)
plt.xlabel("Children",fontsize=15)
plt.ylabel("Count",fontsize=15)
```

Out[35]: Text(0, 0.5, 'Count')



In [36]:
```
## boxenplot for childern colum

plt.figure(figsize=(10,5))
sns.boxenplot(x="children",data=df2,palette = "viridis")
plt.title("Children count in data",fontsize=15)
plt.xlabel("Children",fontsize=15)
plt.ylabel("Count",fontsize=15)
```

Out[36]: Text(0, 0.5, 'Count')



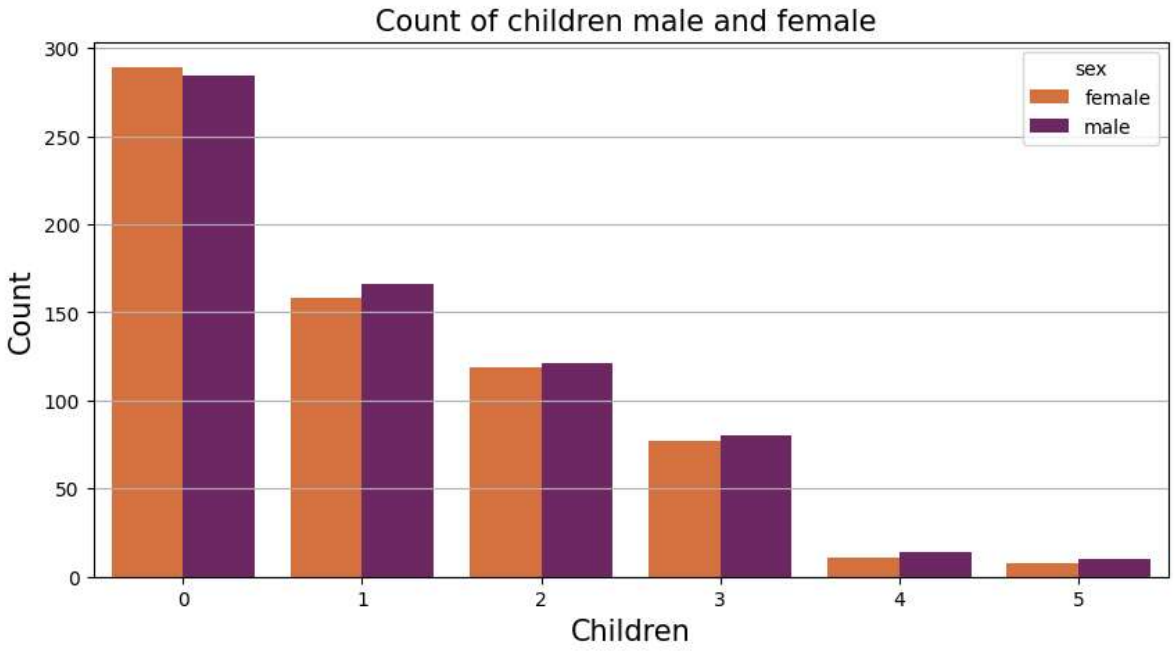
Observation:

* Most of the 0 to 2 childrens covered by health insurance.
* 42% 1 or 2 childrens covered by health insurance.
* 43% childrens are not covered in health insurance.

Compare colum children and sex

Compare relation between Which children is male and female

In [89]:
```
plt.figure(figsize=(10,5))
sns.countplot(x="children",hue="sex",data=df2,palette="inferno_r")
plt.title("Count of children male and female ",fontsize=15)
plt.xlabel("Children",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis='y')
```



Observation:

* There are 0 male and female children who covered by health insurance count is more than 250.
* There are at least 1 male and female children who covered by health insurance count is more than 150.
* There are 2 male and female children who covered by health insurance count is more than 100.

Analyse smoker colum

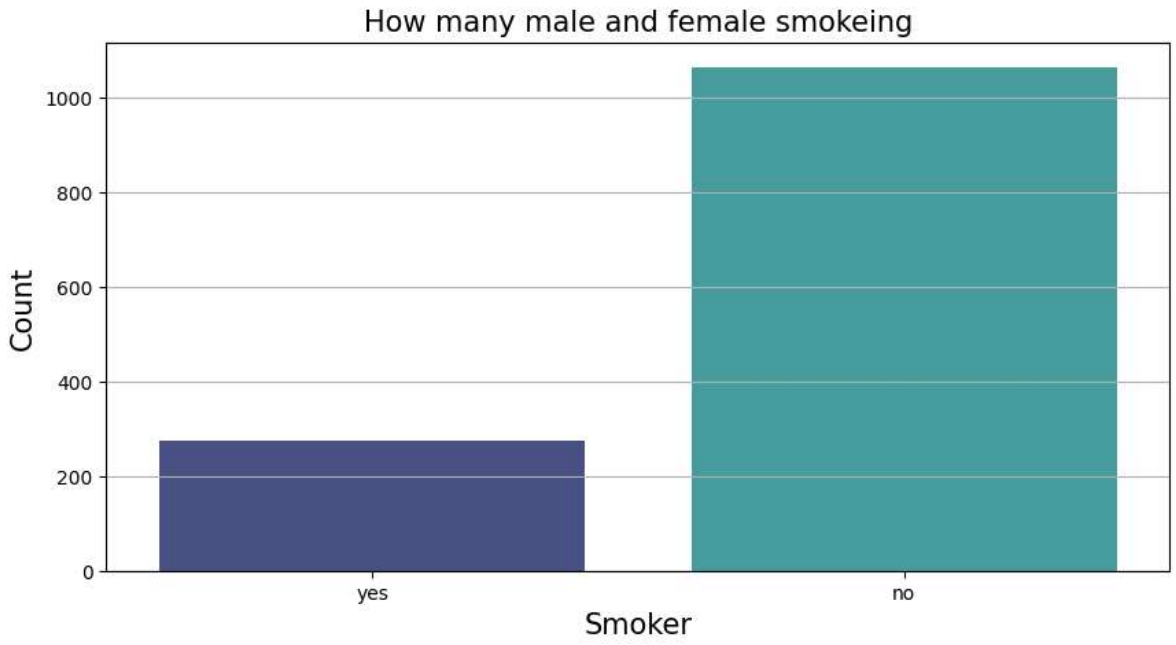
In [38]:
```
df["smoker"].unique()
```

Out[38]:
```
array(['yes', 'no'], dtype=object)
```

In [39]:
```
df.smoker.value_counts()
```

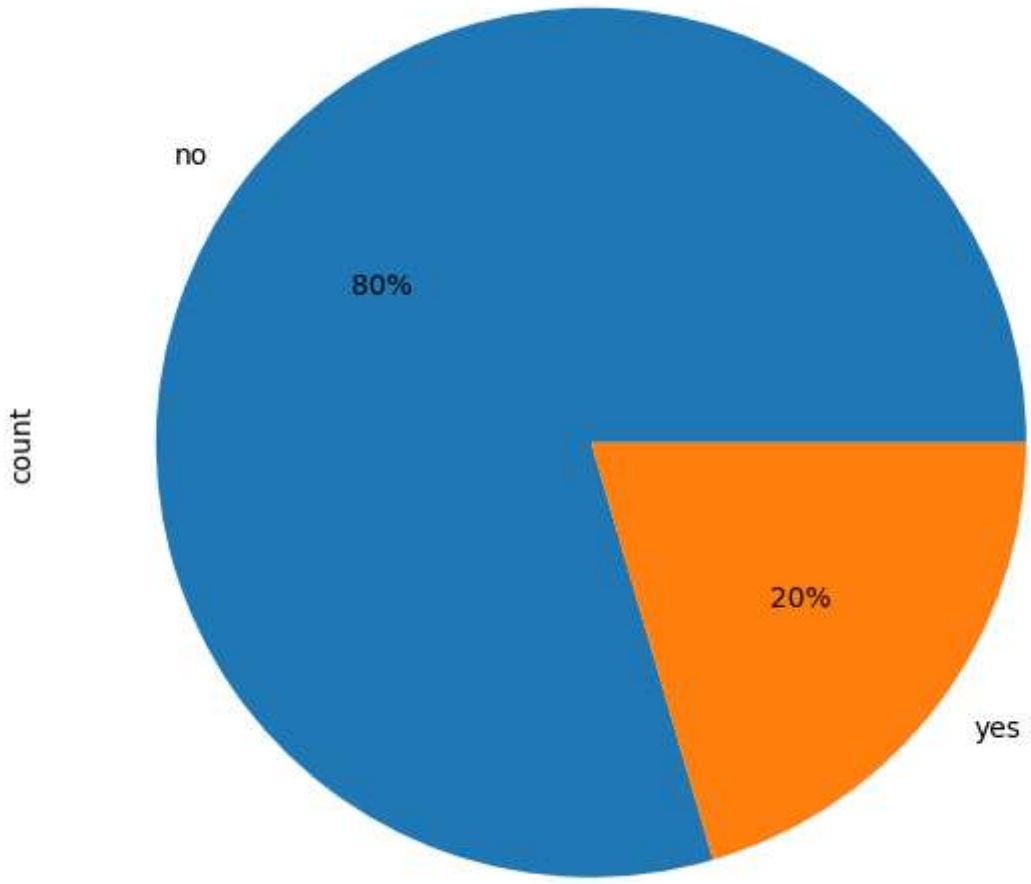
Out[39]:
```
smoker
no     1064
yes     274
Name: count, dtype: int64
```

In [40]:
```
## countplot for smoker colum

plt.figure(figsize=(10,5))
sns.countplot(x="smoker",data=df2,palette="mako")
plt.title("How many male and female smokeing",fontsize=15)
plt.xlabel("Smoker",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
```

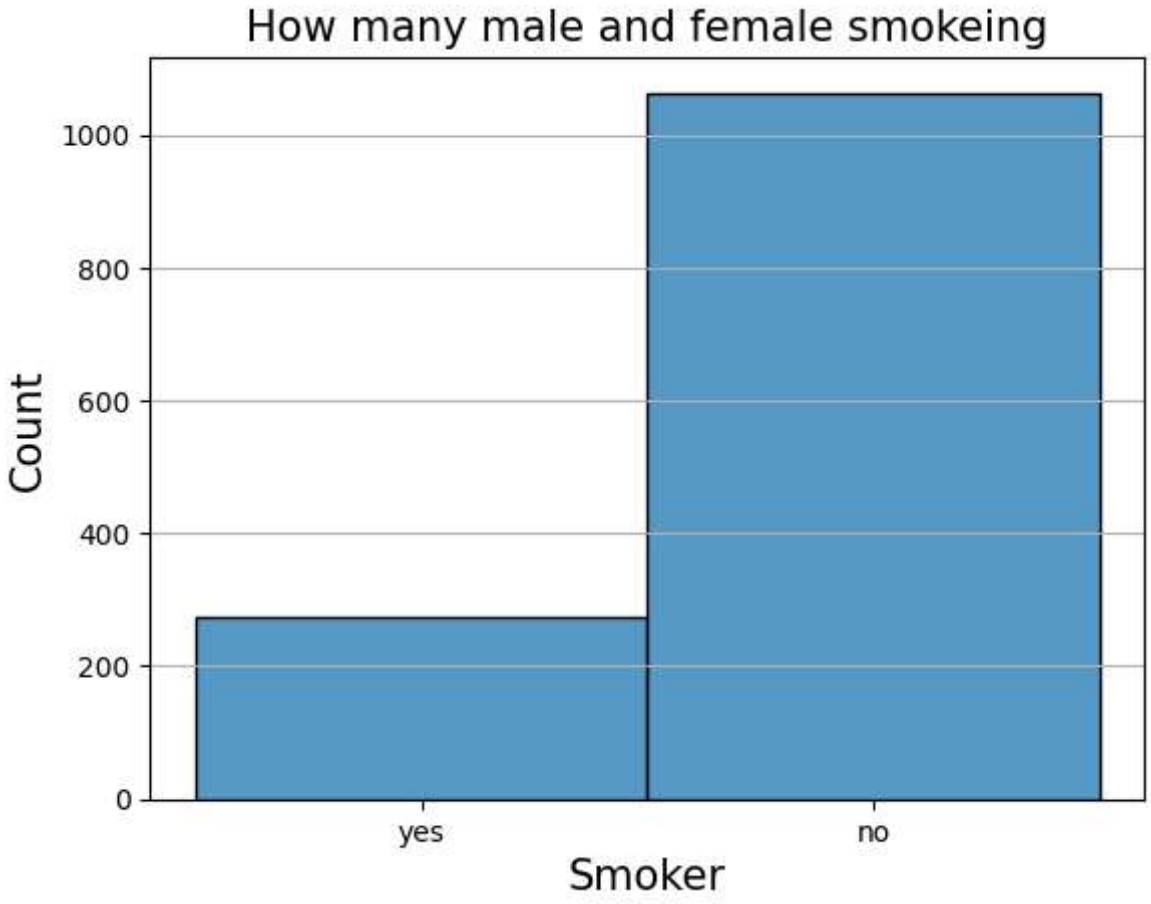


In [41]:
```
## piechart for smoker colum

plt.figure(figsize=(7,15))
df["smoker"].value_counts().plot.pie(autopct="%.0f%%")
```

Out[41]: <Axes: ylabel='count'>



In [42]:
```
## histplot for smoker colum

sns.histplot(x="smoker",data=df2)
plt.title("How many male and female smokeing",fontsize=15)
plt.xlabel("Smoker",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
```



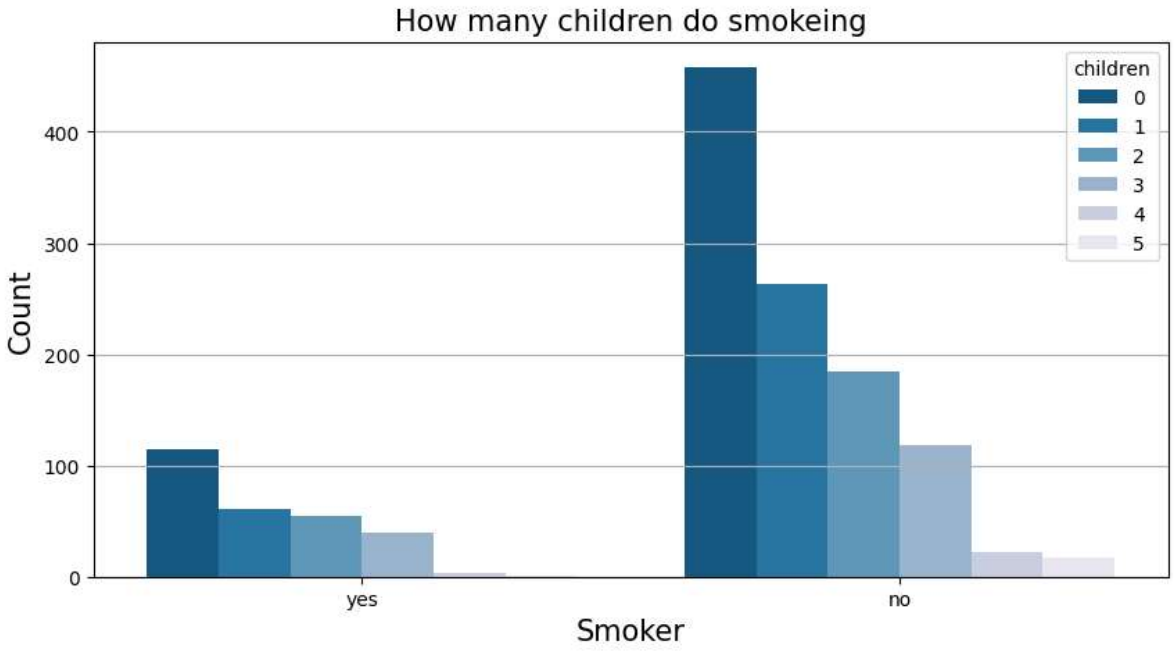
Observation:

* From above graph show.
* Most of the people is not smoker.
* 80% people is not smoker.
* 20% people is smoker.

Compare colum smoker and children

Compare relation between Which children is male and female

In [43]:
```
plt.figure(figsize=(10,5))
sns.countplot(x="smoker",hue="children",data=df2,palette="PuBu_r")
plt.title("How many children do smokeing",fontsize=15)
plt.xlabel("Smoker",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
```

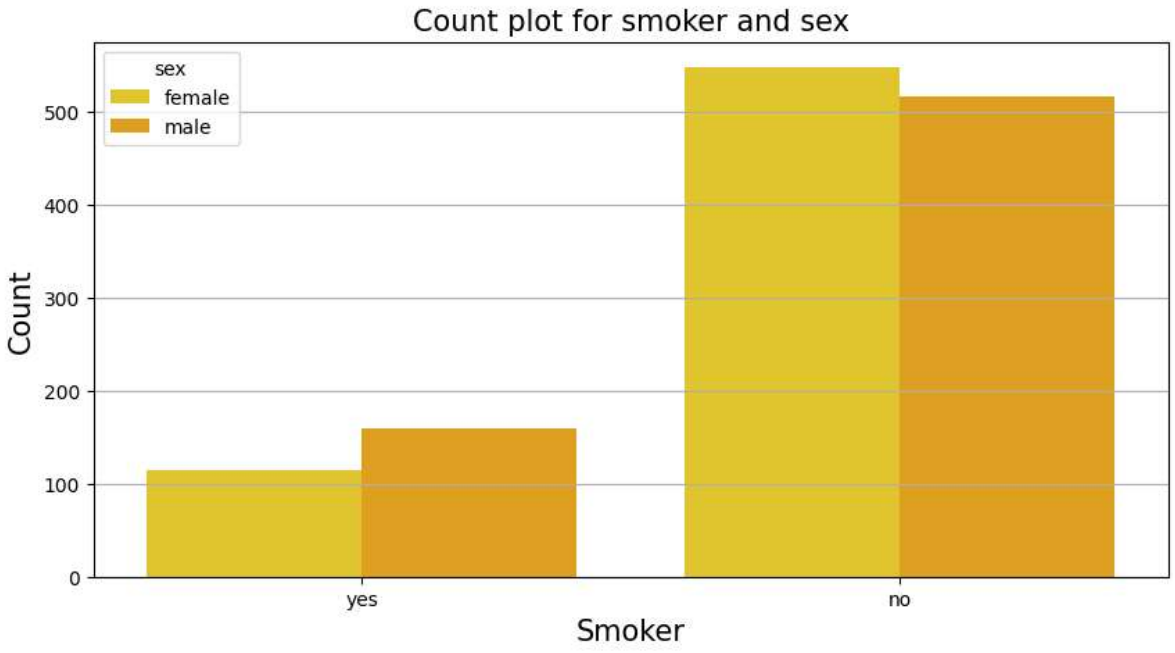


Observation:

* Most of the Children who are covered in insurance do not smoke .

Compare colum smoker and sex

In [44]:
```
plt.figure(figsize=(10,5))
sns.countplot(x="smoker",hue="sex",data=df2,palette="Wistia")
plt.title("Count plot for smoker and sex",fontsize=15)
plt.xlabel("Smoker",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
```



Observation:

* More than 500 male and female are not smoker.
* More than 100 male and female are smoker.

Analyse region colum

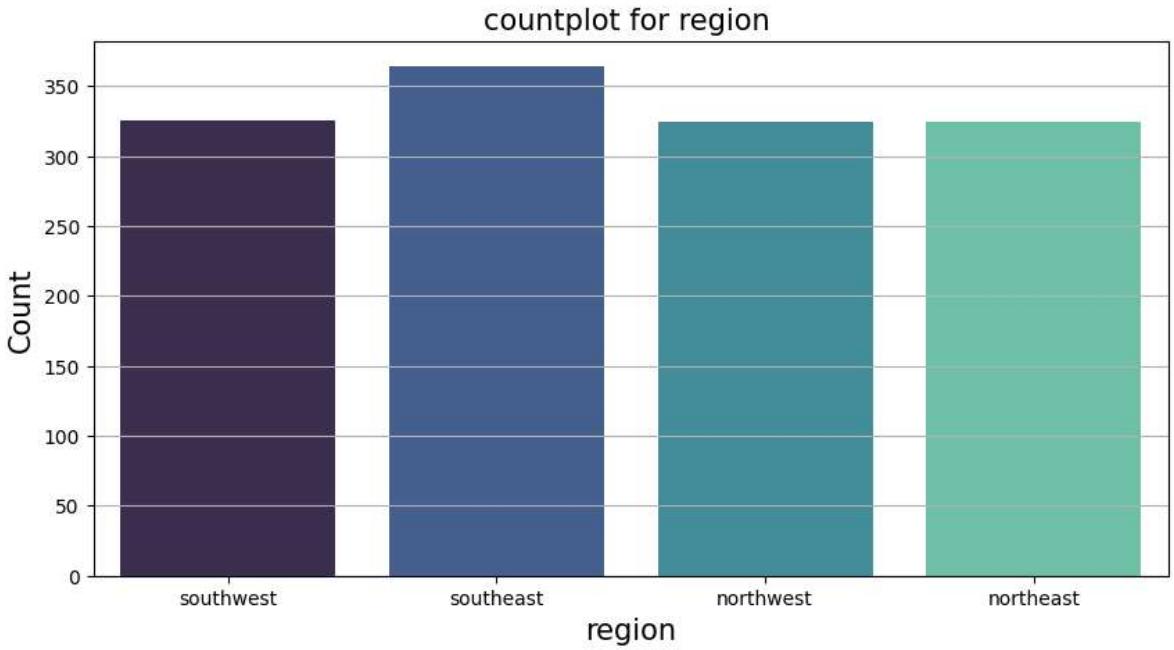
In [45]:
```
df["region"].unique()
```

Out[45]:
```
array(['southwest', 'southeast', 'northwest', 'northeast'], dtype=object)
```

In [46]:
```
df.region.value_counts()
```

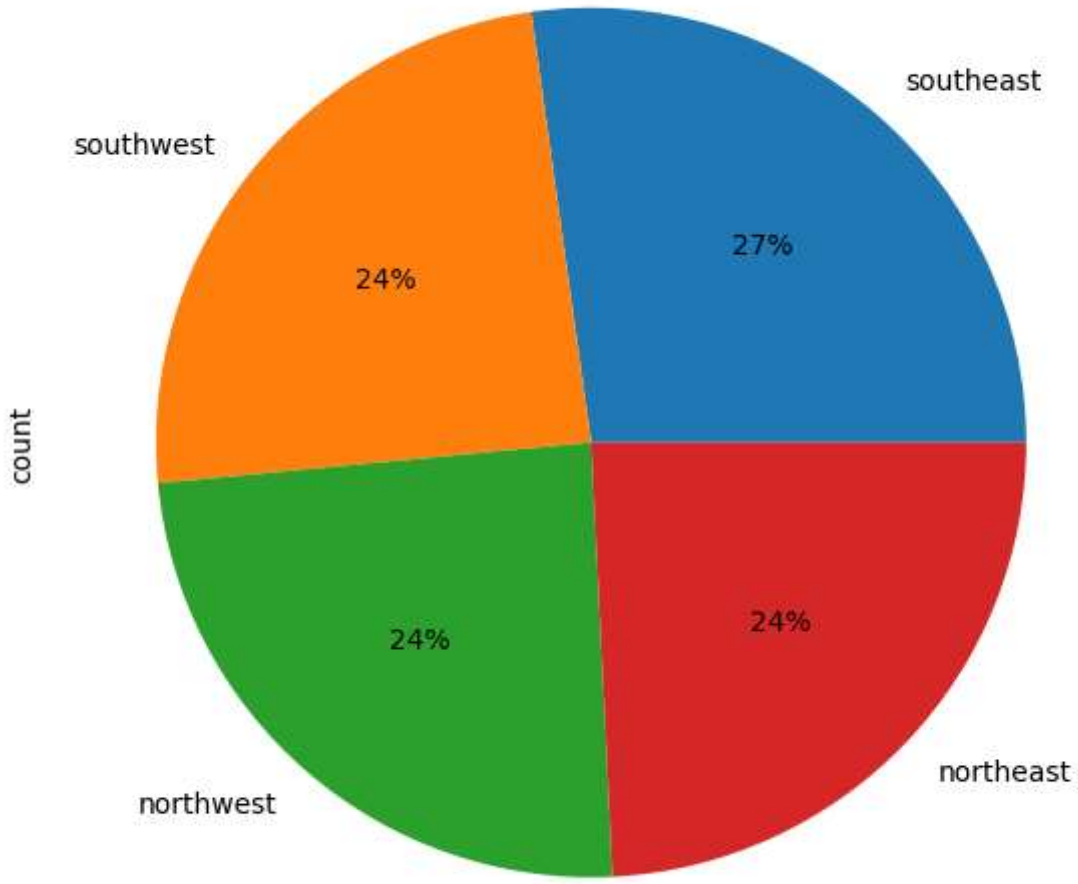
Out[46]:
```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

In [47]:
```
## countplot for region colum

plt.figure(figsize=(10,5))
sns.countplot(x="region",data=df2,palette="mako")
plt.title("countplot for region ",fontsize=15)
plt.xlabel("region",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
```

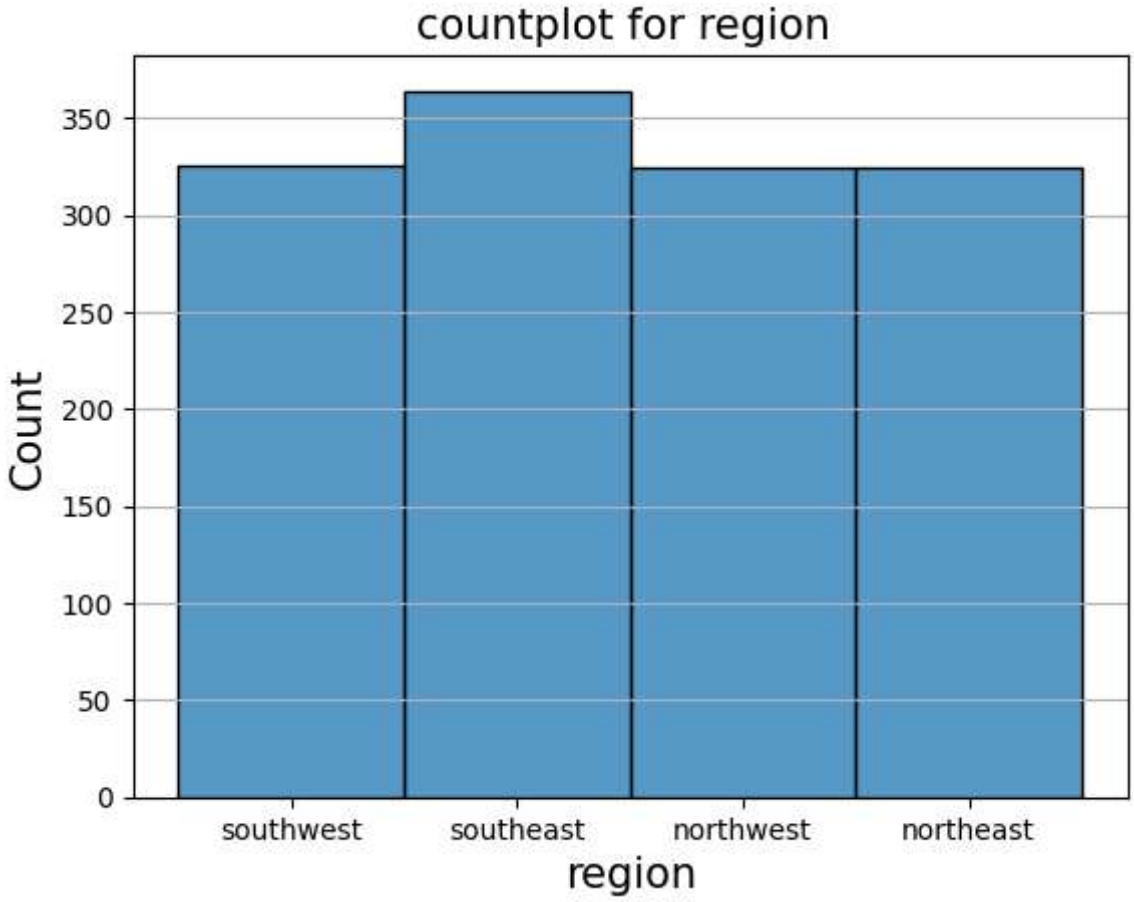


In [48]:
```
## piechart for region colum

plt.figure(figsize=(7,15))
df["region"].value_counts().plot.pie(autopct="%.0f%%")
```

Out[48]: <Axes: ylabel='count'>



In [49]:
```
## histplot for region colum

sns.histplot(x="region",data=df2)
plt.title("countplot for region ",fontsize=15)
plt.xlabel("region",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
```

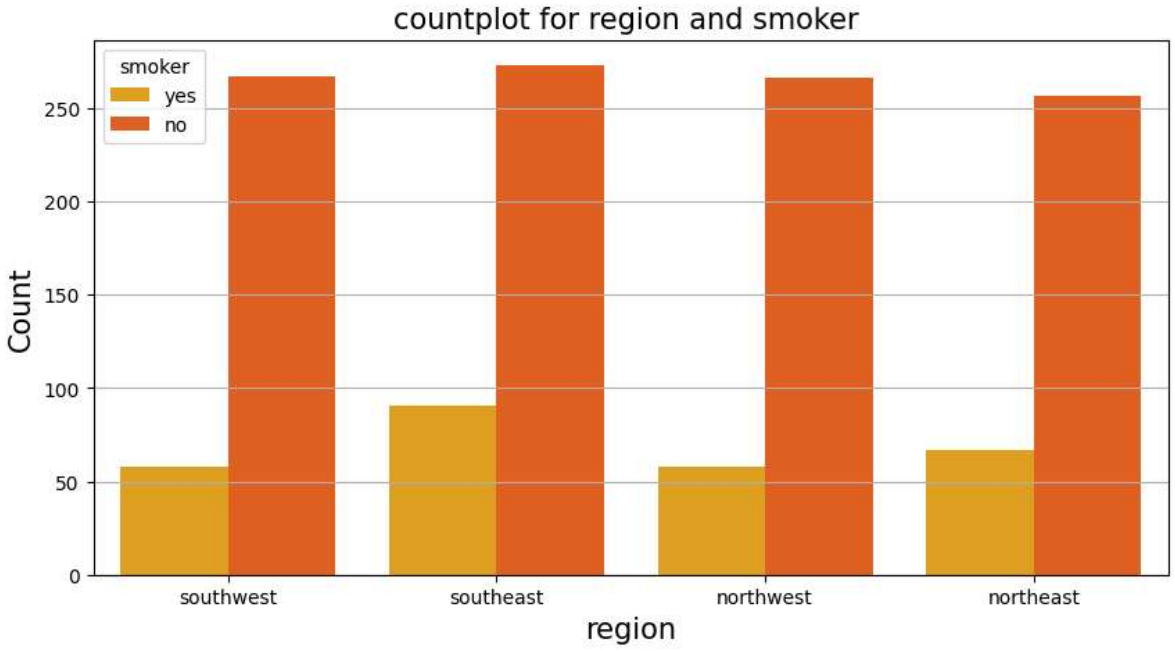


Observation:

* Most of health insurance holder came from southeast of us.
* 27% health insurance holder came from southeast of us.
* 24% health insurance holder came from southwest of us.
* 24% health insurance holder came from northwest of us.
* 24% health insurance holder came from northeast of us.

Compare colum region and smoker

In [50]:
```
plt.figure(figsize=(10,5))
sns.countplot(x="region",hue="smoker",data=df2,palette="autumn_r")
plt.title("countplot for region and smoker ",fontsize=15)
plt.xlabel("region",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
```

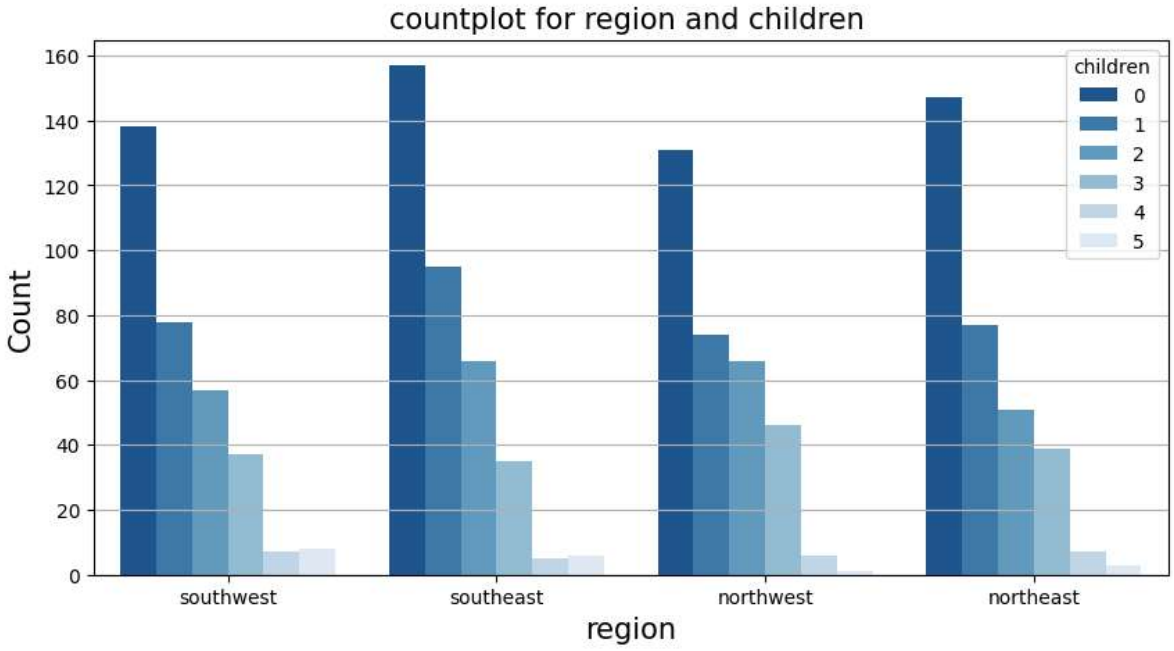


Observation:

* There are over 250 people in all the four regions who do not smoke.
* There are over 50 people in all the four regions who are smoke.

Compare colum region and children

In [51]:
```
plt.figure(figsize=(10,5))
sns.countplot(x="region",hue="children",data=df2,palette="Blues_r")
plt.title("countplot for region and children ",fontsize=15)
plt.xlabel("region",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
```

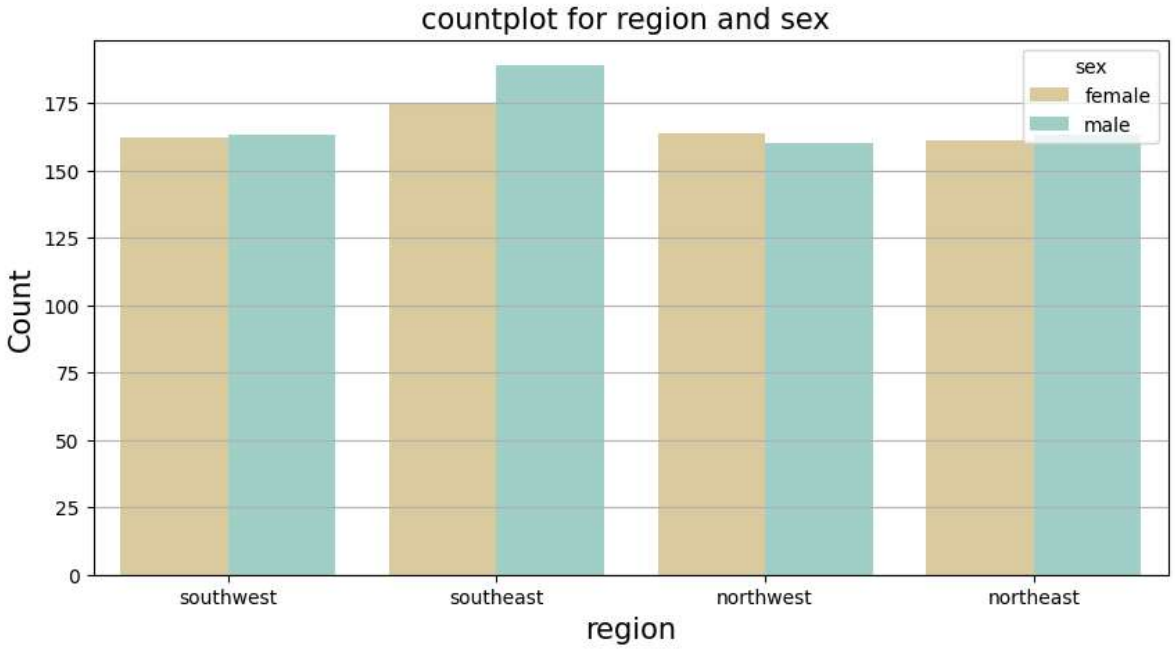


Observation:

* Southest is the region where most of children corved by health insurance.

Compare colum region and sex

In [52]:
```
plt.figure(figsize=(10,5))
sns.countplot(x="region",hue="sex",data=df2,palette="BrBG")
plt.title("countplot for region and sex ",fontsize=15)
plt.xlabel("region",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
```



Observation:

* Most of the male and female come from SouthEast region.
* Equal numbers of male and female come from the other three regions.

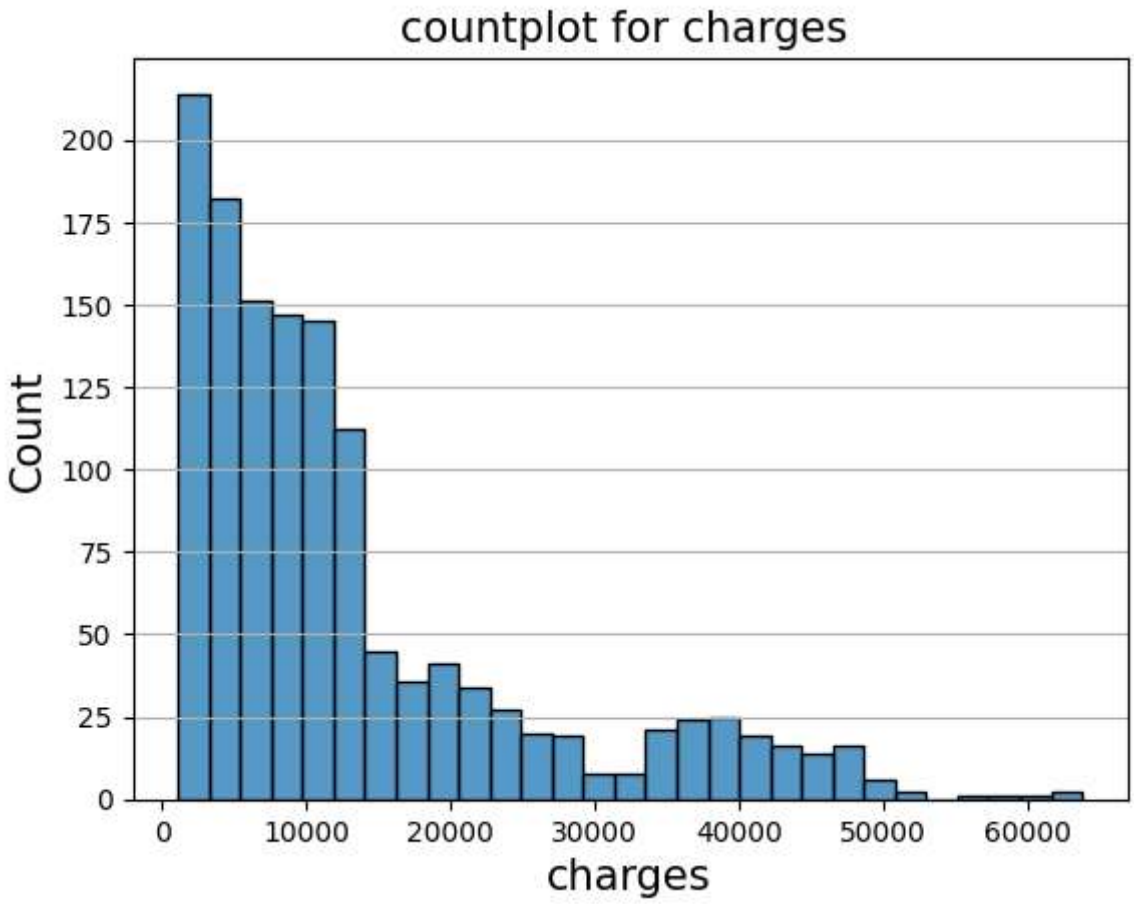
In [53]:
```
df["charges"].unique()
```

Out[53]:
```
array([16884.924 ,  1725.5523,  4449.462 , ...,  1629.8335,  2007.945 ,
       29141.3603])
```

In [54]:
```
df.charges.value_counts()
```

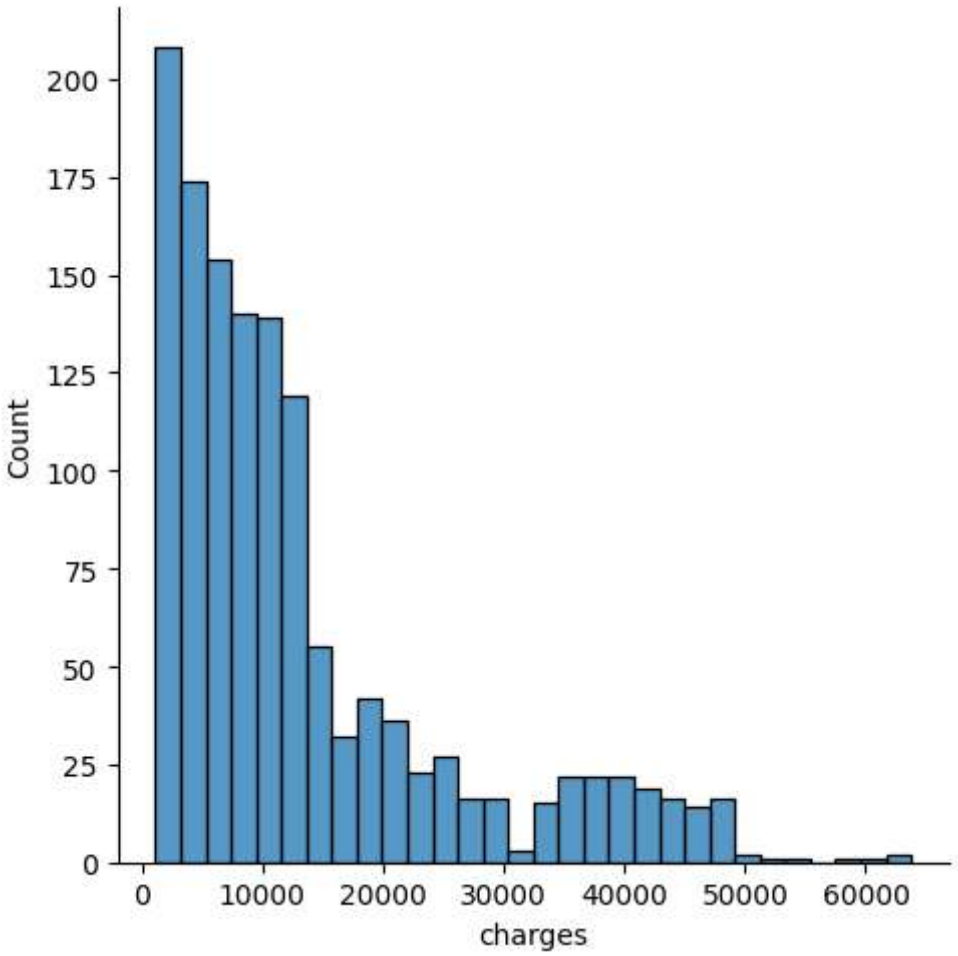
Out[54]:
```
charges
1639.56310     2
16884.92400    1
29330.98315    1
2221.56445     1
19798.05455    1
              ..
7345.08400     1
26109.32905    1
28287.89766    1
1149.39590     1
29141.36030    1
Name: count, Length: 1337, dtype: int64
```

In [55]:
```
sns.histplot(x="charges",data=df2)
plt.title("countplot for charges ",fontsize=15)
plt.xlabel("charges",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
```



In [56]:
```
sns.displot(df["charges"])
```

Out[56]:
```
<seaborn.axisgrid.FacetGrid at 0x16e13f018d0>
```



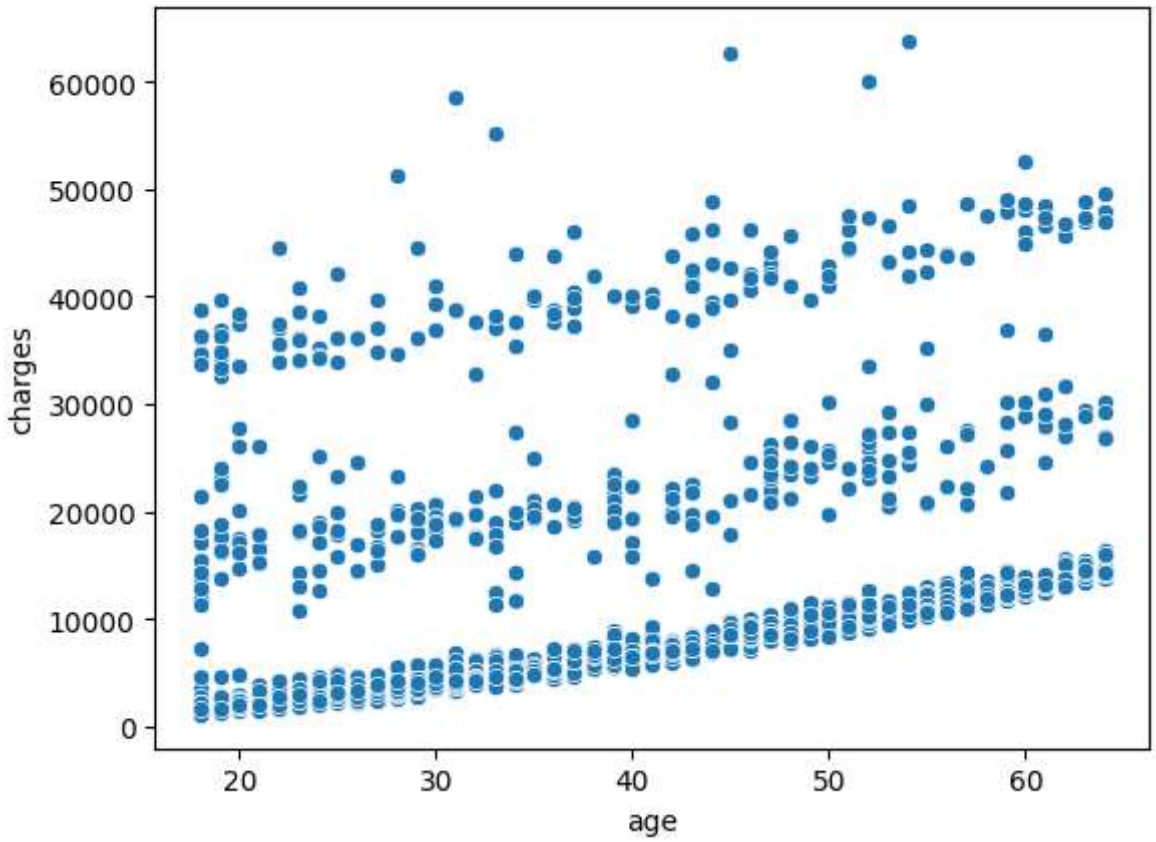
Observation:

```
* Most of the charge in between 1000 to 15000
```

Compare colum charges and age

In [57]:
```
sns.scatterplot(x="age",y="charges",data=df2)
```

Out[57]:
```
<Axes: xlabel='age', ylabel='charges'>
```



Observation:

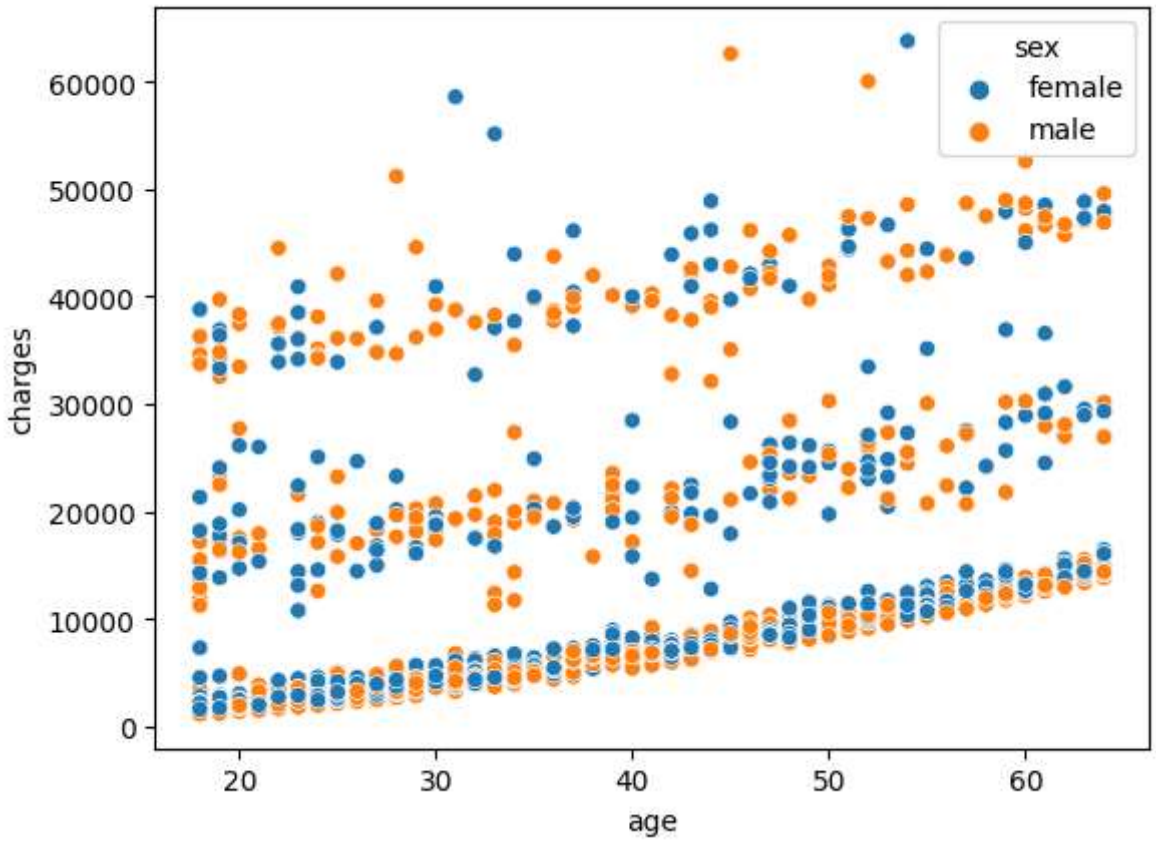
```
  * The cost of insurance for children between the age of 0 to 20 yea
rs falls between 1000 to 5000.
```

Compare charges and age colum with categorical colum sex,smoker and region

Compare colum charges , age and sex

In [58]:
```
sns.scatterplot(x="age",y="charges",hue="sex",data=df2)
```

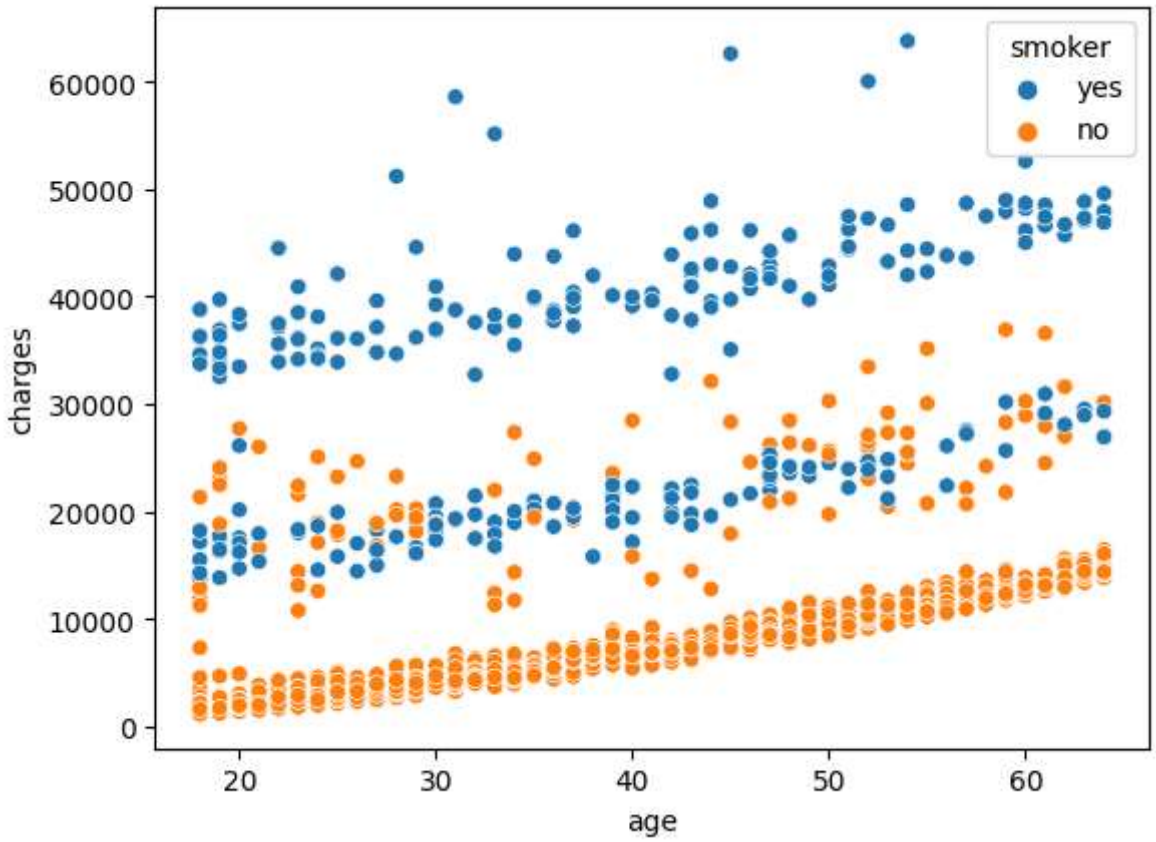
Out[58]:
```
<Axes: xlabel='age', ylabel='charges'>
```



Compare colum charges , age and smoker

In [59]:
```
sns.scatterplot(x="age",y="charges",hue="smoker",data=df2)
```

Out[59]:
```
<Axes: xlabel='age', ylabel='charges'>
```



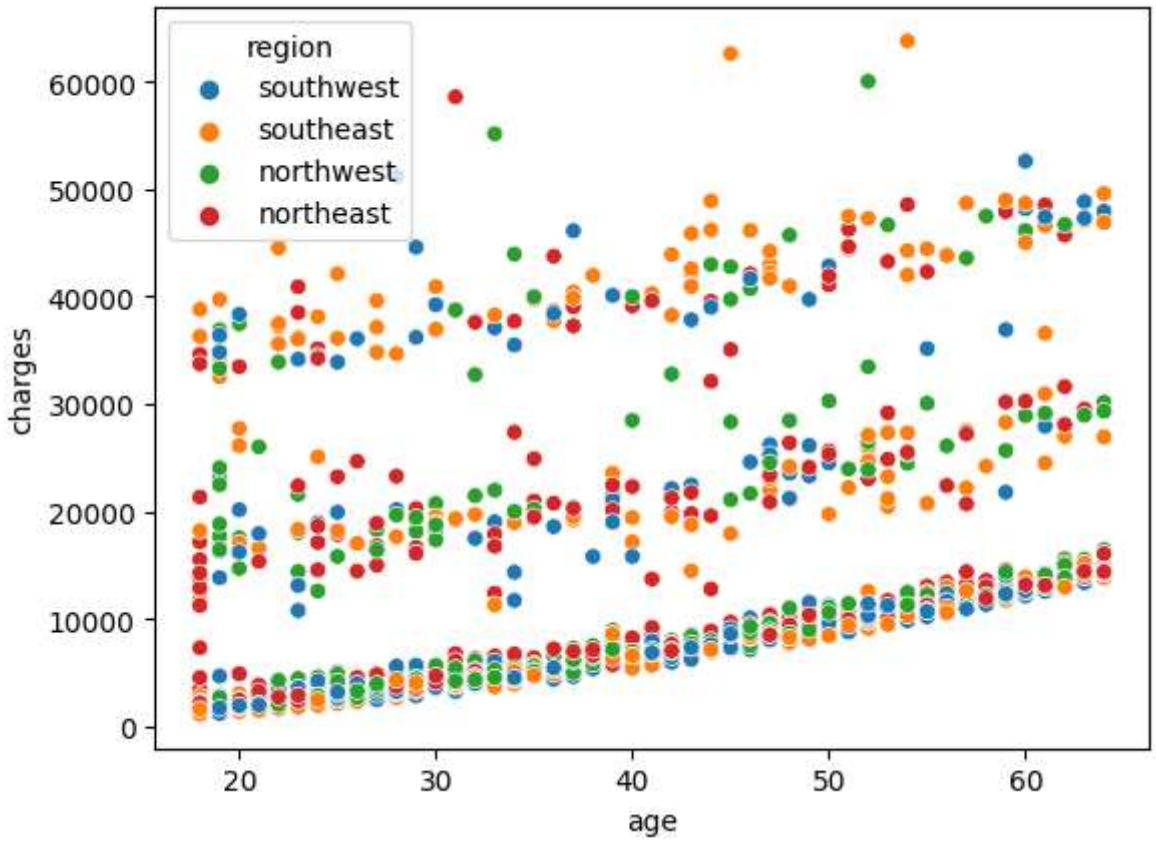
Observation:

* Those who smoke the most, age vise their insurance price falls between 30000 to 60000.
* For those who do not smoke, the price of insurance comes between 1000 to 10000.

Compare colum charges , age and region

In [60]:
```
sns.scatterplot(x="age",y="charges",hue="region",data=df2)
```

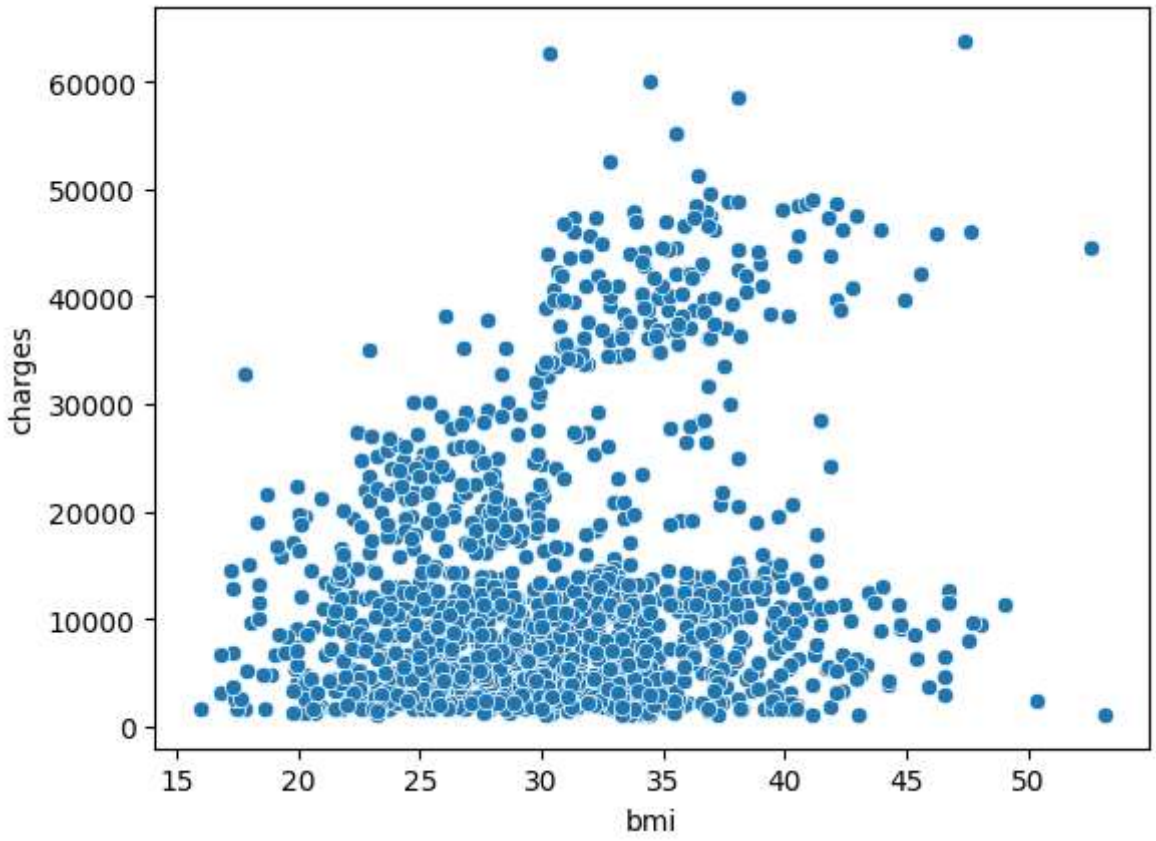
Out[60]:
```
<Axes: xlabel='age', ylabel='charges'>
```



Compare colum charges and bmi

In [61]:
```
sns.scatterplot(x="bmi",y="charges",data=df2)
```

Out[61]:
```
<Axes: xlabel='bmi', ylabel='charges'>
```



Observation:

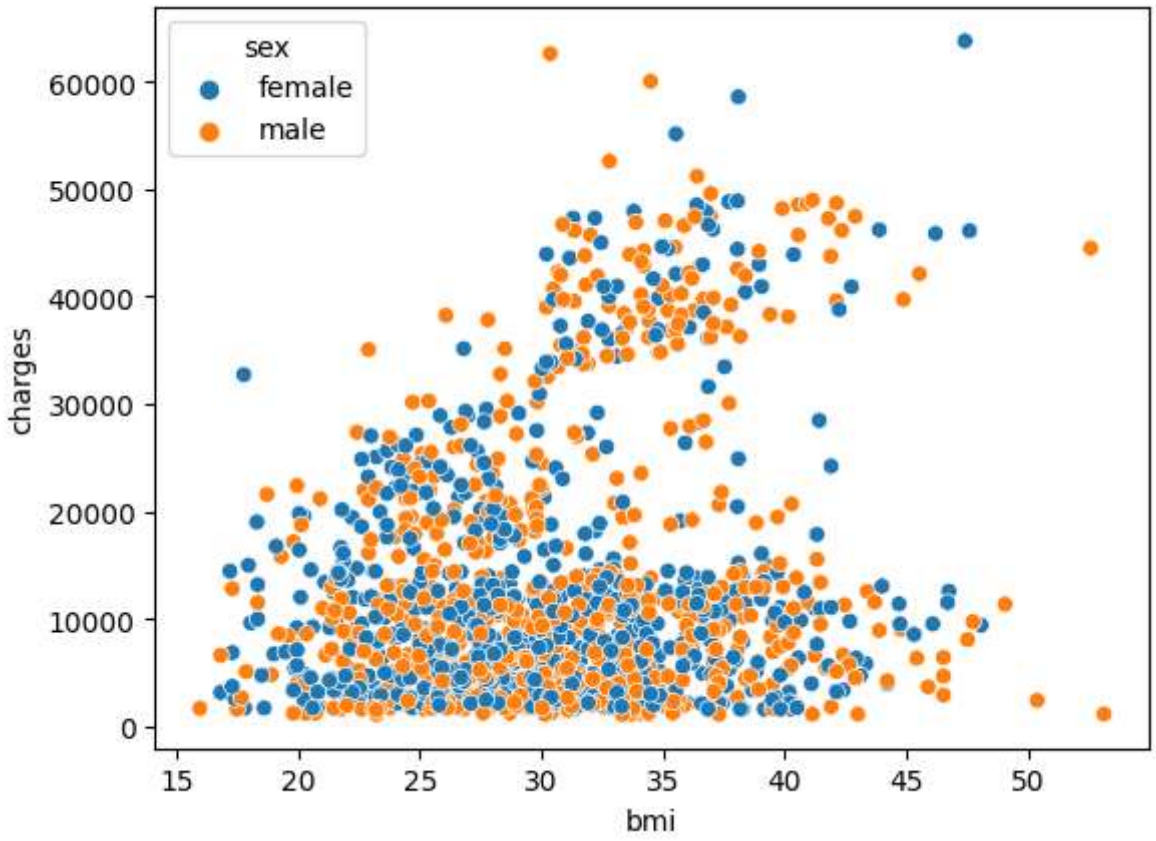
```
* Whose weight is in the range of 25 to 35 it means they are underwei
ght. The price of their insurance comes between 1000 to 15000.
```

Compare charges and bmi colum with categorical colum sex,smoker and region

Compare colum charges , bmi and sex

In [62]:
```
sns.scatterplot(x="bmi",y="charges",hue="sex",data=df2)
```

Out[62]:
```
<Axes: xlabel='bmi', ylabel='charges'>
```



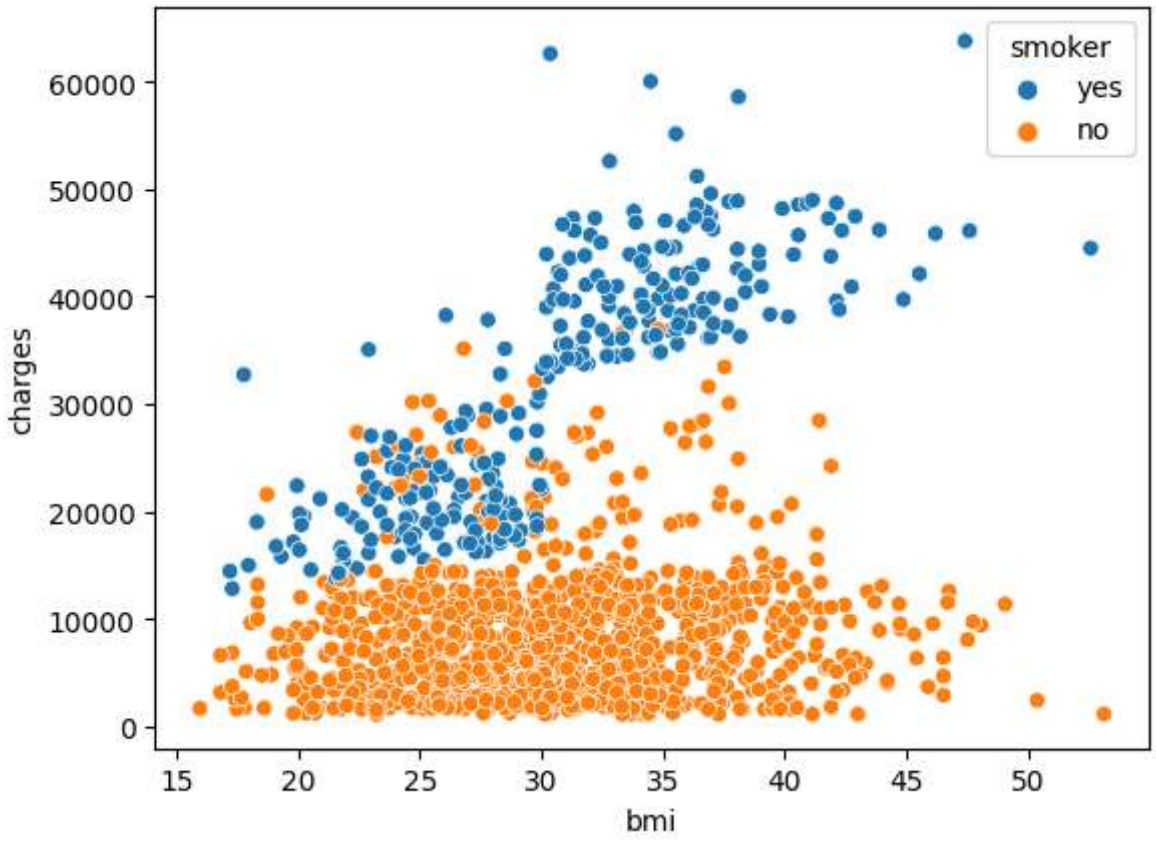
Observation:

- most of male and female the weight is in the range of 25 to 35 it means they are underweight. The price of their insurance comes between 1000 to 15000.
- male and female the weight is in the range of 33 to 50 it means they are overweight. The price of their insurance comes between 30000 to 60000.

Compare colum charges , bmi and smoker

In [63]:
```
sns.scatterplot(x="bmi",y="charges",hue="smoker",data=df2)
```

Out[63]:
```
<Axes: xlabel='bmi', ylabel='charges'>
```



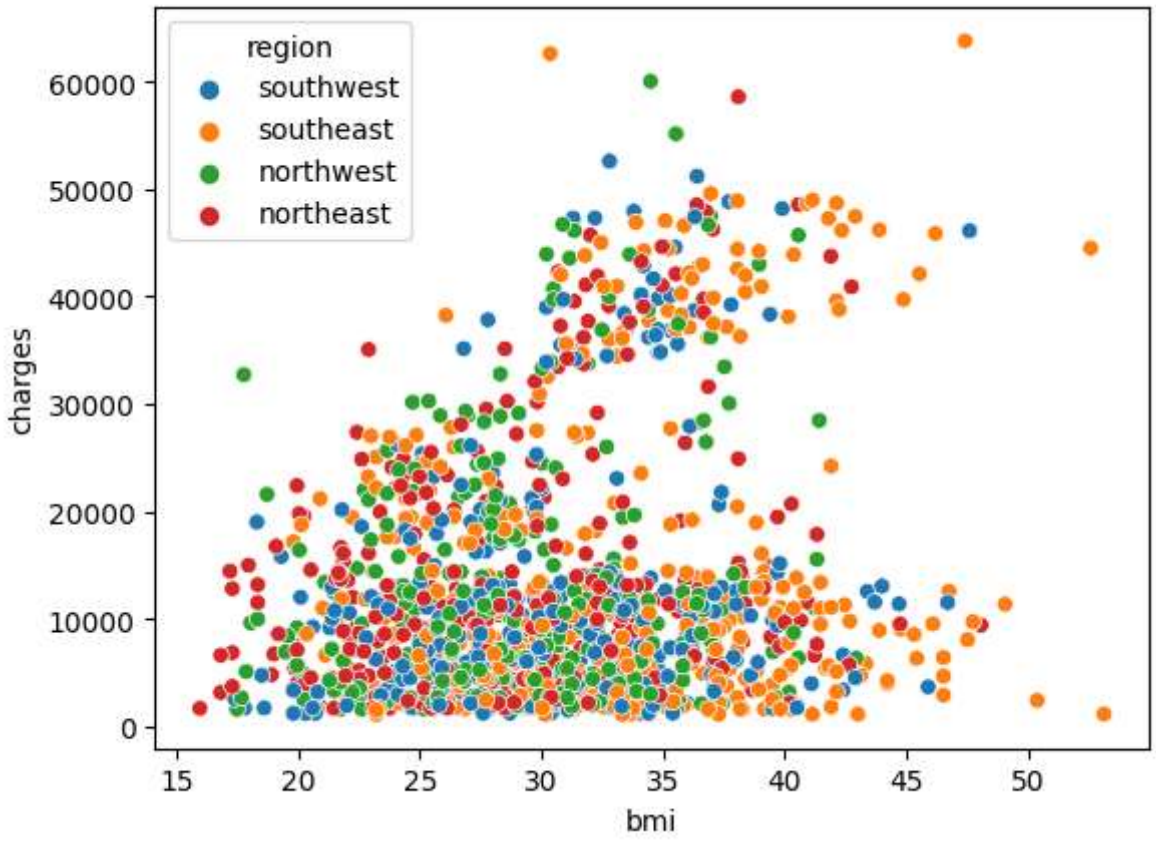
Observation:

* People who do not smoke and their weight range is between 25 to 35 it means they are underweight.The price of their insurance comes from 1000 to 10000.
* People who smoke and their weight range is between 18 to 50.The price of their insurance comes from 15000 to 60000.

Compare colum charges , bmi and region

In [64]:
```
sns.scatterplot(x="bmi",y="charges",hue="region",data=df2)
```

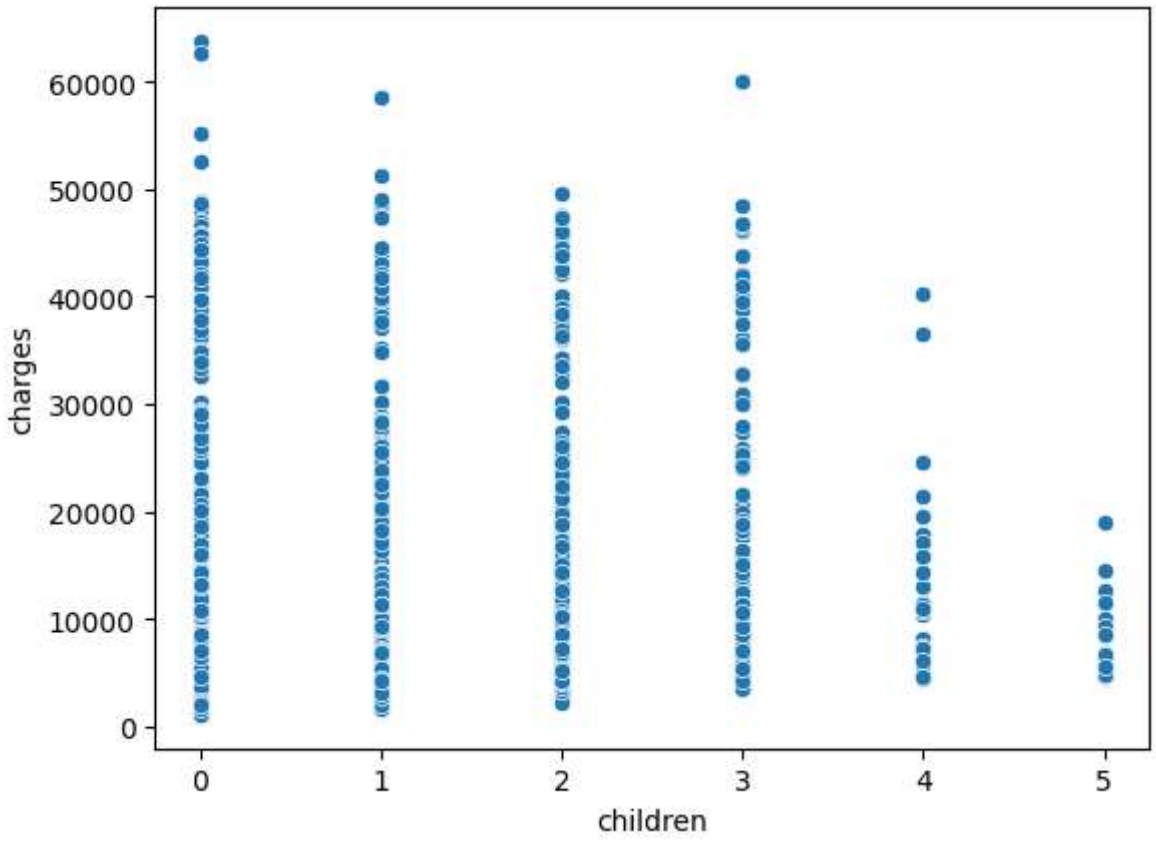
Out[64]:
```
<Axes: xlabel='bmi', ylabel='charges'>
```



Compare colum charges and children

In [65]:
```
sns.scatterplot(x="children",y="charges",data=df2)
```

Out[65]:
```
<Axes: xlabel='children', ylabel='charges'>
```

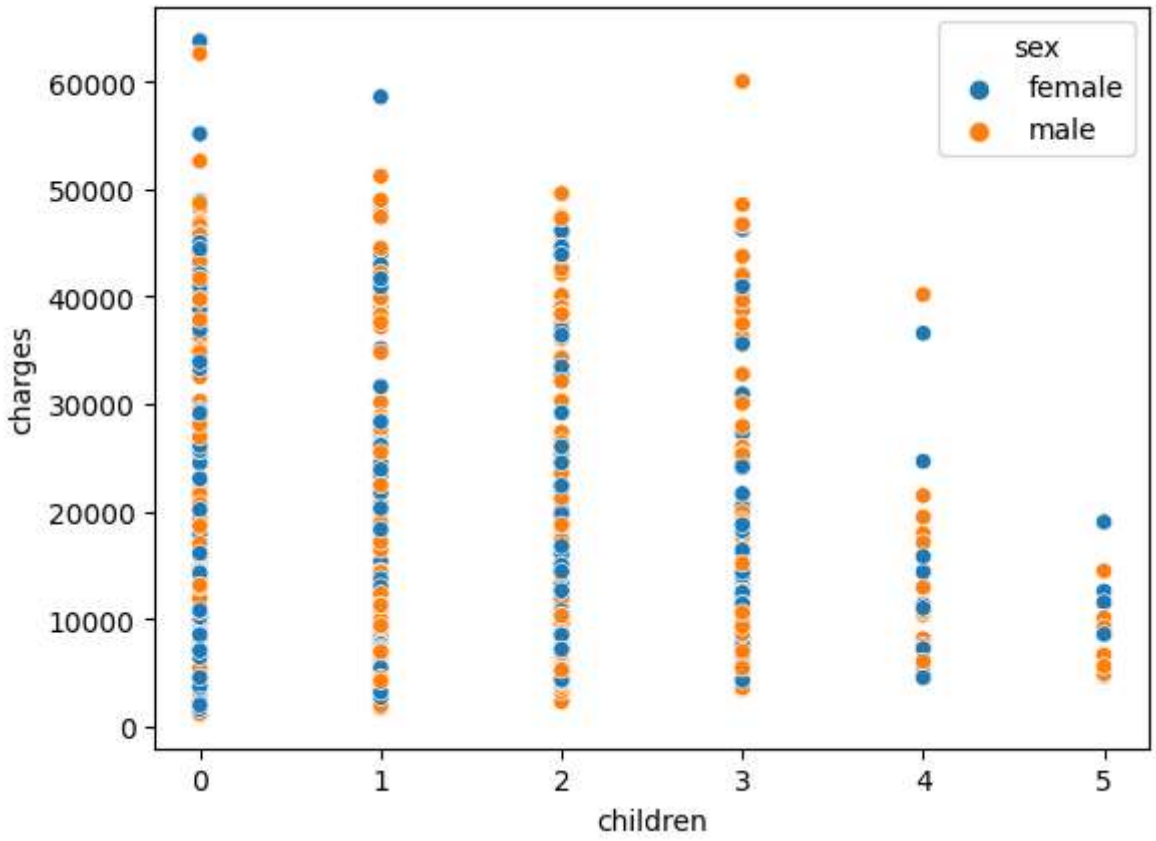


Compare charges and children colum with categorical colum sex,smoker and region

Compare colum charges , children and sex

In [66]:
```
sns.scatterplot(x="children",y="charges",hue="sex",data=df2)
```

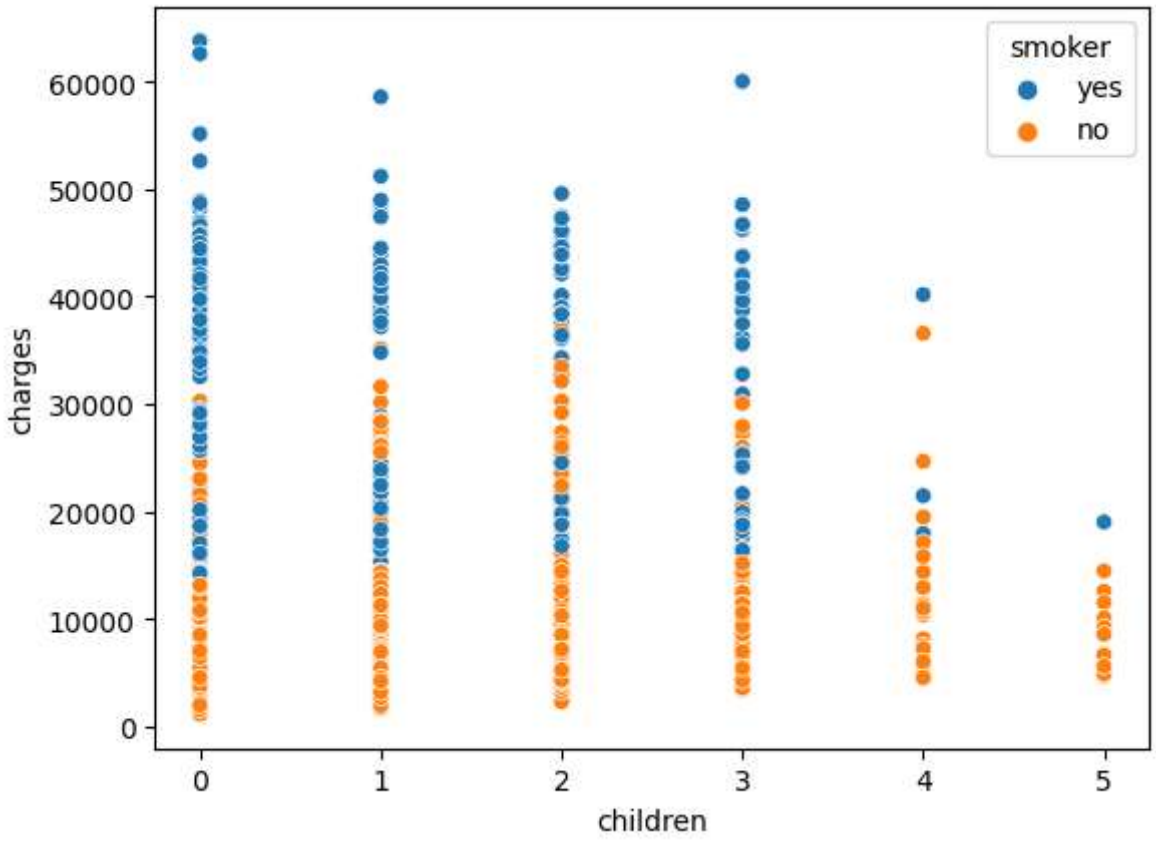
Out[66]:
```
<Axes: xlabel='children', ylabel='charges'>
```



Compare colum charges , children and smoker

In [67]:
```
sns.scatterplot(x="children",y="charges",hue="smoker",data=df2)
```

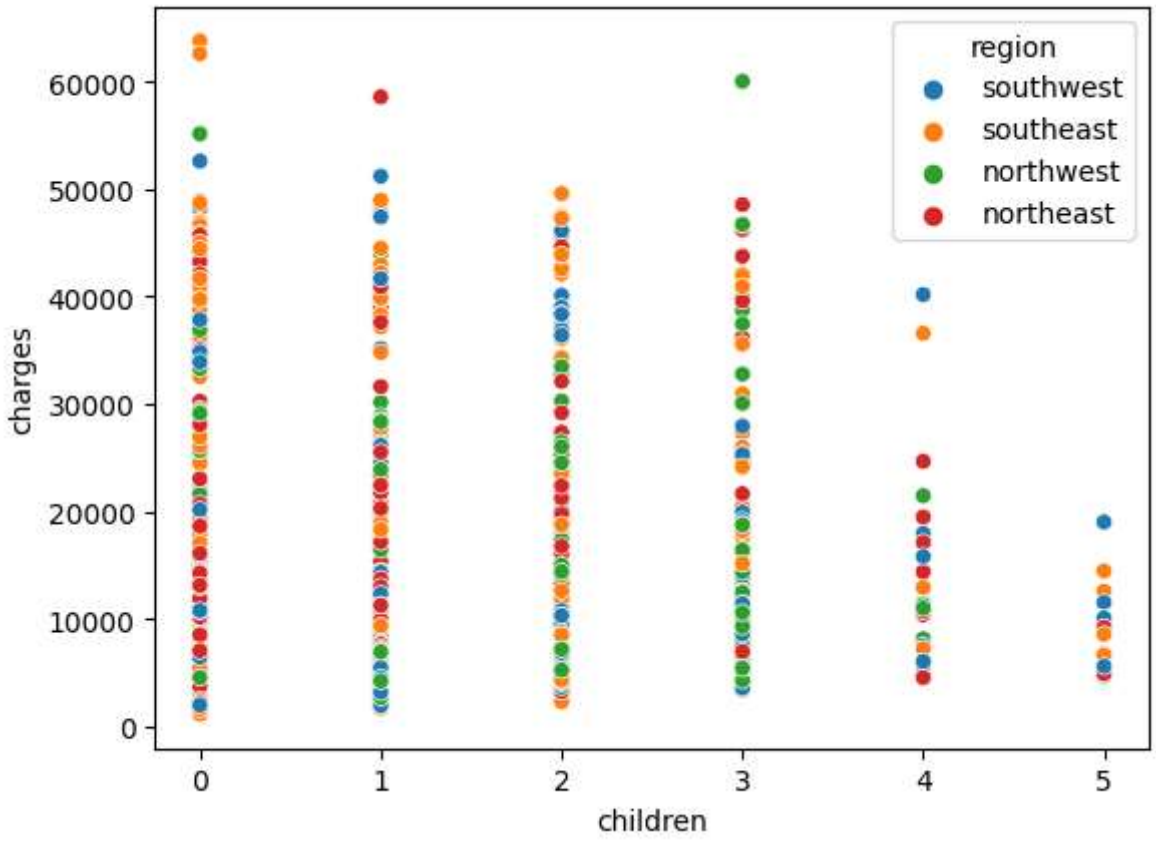
Out[67]:
```
<Axes: xlabel='children', ylabel='charges'>
```



Compare colum charges , children and region

In [68]:
```
sns.scatterplot(x="children",y="charges",hue="region",data=df2)
```

Out[68]:
```
<Axes: xlabel='children', ylabel='charges'>
```



Finding the correlation of data

In [69]:
```
A = df2.select_dtypes(include='number')
A
```

Out[69]:

	age	bmi	children	charges
0	19	27.900	0	16884.92400
1	18	33.770	1	1725.55230
2	28	33.000	3	4449.46200
3	33	22.705	0	21984.47061
4	32	28.880	0	3866.85520
...	...	...	...	...
1333	50	30.970	3	10600.54830
1334	18	31.920	0	2205.98080
1335	18	36.850	0	1629.83350
1336	21	25.800	0	2007.94500
1337	61	29.070	0	29141.36030

1337 rows × 4 columns

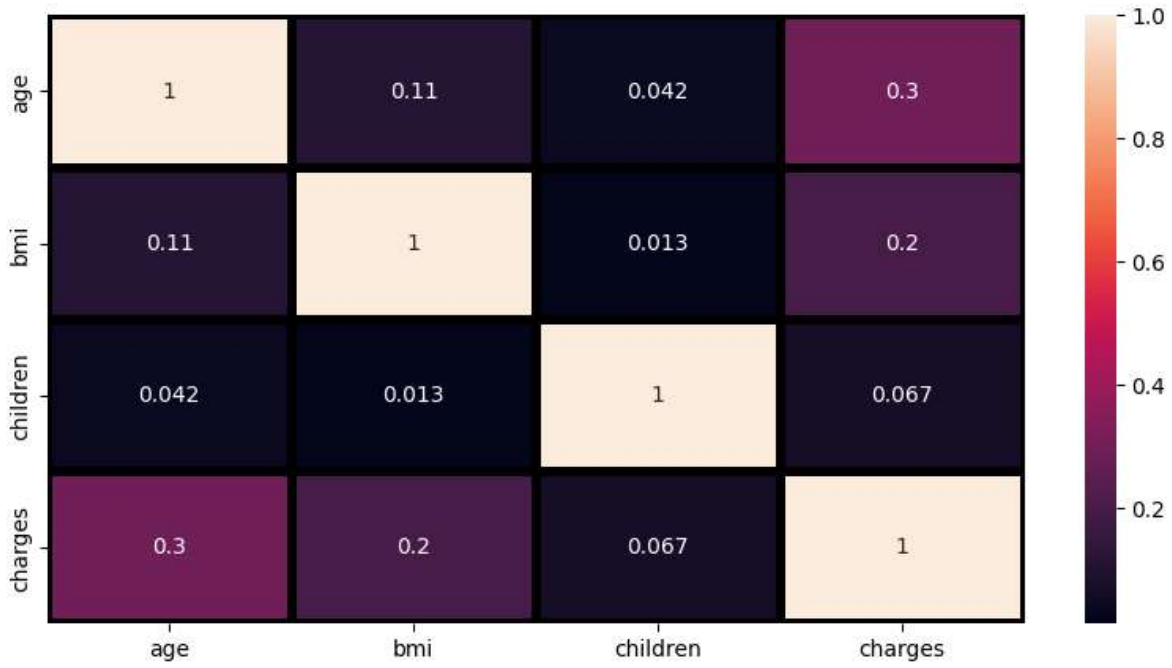
In [70]:
```
correlation=A.corr()
correlation
```

Out[70]:

	age	bmi	children	charges
age	1.000000	0.109344	0.041536	0.298308
bmi	0.109344	1.000000	0.012755	0.198401
children	0.041536	0.012755	1.000000	0.067389
charges	0.298308	0.198401	0.067389	1.000000

In [71]:
```
plt.figure(figsize=(10,5))
sns.heatmap(A.corr(),annot=True,linewidths=4,linecolor="k")
```

Out[71]: <Axes: >



Feature Engineering

- We use here one hot encoding.
- Because there is multiple independent colum with categorical value.

In [72]:
```
## Convert categorical value into numerical value

dataset = pd.get_dummies(df2,dtype=int)
```

In [73]:
```
dataset
```

Out[73]:

	age	bmi	children	charges	sex_female	sex_male	smoker_no	smoker_yes	regio
0	19	27.900	0	16884.92400	1	0	0	1	
1	18	33.770	1	1725.55230	0	1	1	0	
2	28	33.000	3	4449.46200	0	1	1	0	
3	33	22.705	0	21984.47061	0	1	1	0	
4	32	28.880	0	3866.85520	0	1	1	0	
...	...	...	...	...	...	...	...	...	
1333	50	30.970	3	10600.54830	0	1	1	0	
1334	18	31.920	0	2205.98080	1	0	1	0	
1335	18	36.850	0	1629.83350	1	0	1	0	
1336	21	25.800	0	2007.94500	1	0	1	0	
1337	61	29.070	0	29141.36030	1	0	0	1	

1337 rows × 12 columns

In [74]:
```
X = dataset.drop(columns=["charges",'sex_female','smoker_yes','region_northeas
y = dataset["charges"]
```

Spliting data into train and test set

In [100]:
```
from sklearn.model_selection import train_test_split
```

In [101]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)
```

In [102]:
```
X_train.shape
```

Out[102]: (1069, 8)

In [103]:
```
y_train.shape
```

Out[103]: (1069,)

In [104]:
```
X_test.shape
```

Out[104]: (268, 8)

In [105]:
```
y_test.shape
```

Out[105]: (268,)

In [106]:
```
from sklearn.linear_model import LinearRegression
```

In [107]:
```
reg=LinearRegression()
```

In [108]:
```
reg.fit(X_train,y_train)
```

Out[108]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [113]:
```
reg.predict(X_test)
```

Out[113]:
```
array([27310.13246147, 31874.375918  ,  1772.58604182, 11672.70525832,
        6369.58383052, 12728.53859948,  8240.28923199, 11181.79333017,
       12365.33010095,  2121.46693317, 31719.81384571,  2207.99927156,
        9312.96758174,  5493.14300928, 29927.18485085,  9928.95635317,
        5421.05698383, 13744.71035372,  7679.49769252,  5889.78408538,
       -1236.11091331, 15804.49065819, 32965.39339172, 33770.70069303,
       10382.11428828,  2027.01746697, 13121.62679313,  5807.96273679,
       25076.42694083,  9939.17018091,  2151.01049456,  2249.06069688,
       25842.4650258 ,  9085.76778468, 32303.77020093,  3711.13365441,
       33345.81912536,  6923.50493996, 10837.37111815,   337.12306688,
       10708.19455445, 12261.29195471, 26920.51982611,  1799.33803538,
       35613.7829867 ,  2691.62265446, 32125.68706326,  2666.78388552,
       11187.00809774, 23867.92339842,  6692.97075569, 36753.62467385,
       34316.40307165, 35423.60401401,  8758.76927965, 37169.91510944,
        4971.00283475, 14971.38878651, 38615.18637653, 15187.3027871 ,
        9384.7500578 , 10348.08045042,  1359.23258823,  4830.49789776,
        8094.70603833,  5786.25728678,  2323.21378964, 13284.52459209,
       29777.51959459,  6163.70956197,  8109.83195231, 12737.02761337,
       14240.8835295 ,  8901.72318809, 12984.37898324, 24025.2773948 ,
```

In [110]:
```
from sklearn.metrics import r2_score
```

In [111]:
```
r2_score(y_test,reg.predict(X_test))*100
```

Out[111]:
```
76.70815205141135
```

In [ ]:
```

```