In [220]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

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

Out[2]:

	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

In [ ]:
```
# upto here we are uploded "medical_cost_insurance.csv" to jupyter notebook.
# and make df as a instance of our insurance dataset.
```

In [4]:
```
df.shape
# here we finds the shape of our dataset, i.e it containing 1338 ROWS & 7 COLUMNS
```

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

In [6]:
```
df.columns
# here we finds the names of different 7 columns of the data set.
```

Out[6]:
```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dtype='object')
```

In [8]:
```
df.dtypes
# here can see that there are some different types of data is present in the given dataset like : [ int64, object, float64 ]
```

Out[8]:
```
age           int64
sex          object
bmi         float64
children      int64
smoker       object
region       object
charges     float64
dtype: object
```

In [9]:
```
df["age"].unique()
```

Out[9]:
```
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 [10]:
```
df["sex"].unique()
```

Out[10]:
```
array(['female', 'male'], dtype=object)
```

In [14]:
```
df["bmi"].unique()
```

Out[14]:
```
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 [15]:
```
df["children"].unique()
```

Out[15]:
```
array([0, 1, 3, 2, 5, 4], dtype=int64)
```

In [16]:
```
df["smoker"].unique()
```

Out[16]:
```
array(['yes', 'no'], dtype=object)
```

In [17]:
```
df["region"].unique()
```

Out[17]:
```
array(['southwest', 'southeast', 'northwest', 'northeast'], dtype=object)
```

In [18]:
```
df["charges"].unique()
```

Out[18]:
```
array([16884.924 ,  1725.5523,  4449.462 , ...,  1629.8335,  2007.945 ,
       29141.3603])
```

In [19]:
```
#  From the above analysis we can find that the column SEX, CHILDREN, SMOKER, REGION are the catgorical columns
```

In [21]:
```
df['sex'].nunique()
```

Out[21]:
```
2
```

In [22]:
```
df['children'].nunique()
```

Out[22]:
```
6
```

In [23]:
```
df['smoker'].nunique()
```

Out[23]:
```
2
```

In [24]:
```
df['region'].nunique()
```

Out[24]:
```
4
```

In [25]:
```
# here above we can see the no. of unique categories present in the categorical columns
```

In [27]:
```
df['sex'].value_counts()
```

Out[27]:
```
male      676
female    662
Name: sex, dtype: int64
```

In [ ]:
```
# in first categorical column "sex" there are 676 MALES & 662 FEMALES are presents,
#  so we observe that the quantity of male is slightly higher then the females
```

In [30]:
```
df['children'].value_counts()
```

Out[30]:
```
0    574
1    324
2    240
3    157
4     25
5     18
Name: children, dtype: int64
```

In [ ]:
```
#  here we find that the majority of the customers having 0,1,2, or 3 childerens
```

In [31]:
```
df['smoker'].value_counts()
```

Out[31]:
```
no     1064
yes     274
Name: smoker, dtype: int64
```

In [ ]:
```
#  the majority of the customers are NON-SMOKER
```

In [32]:
```
df['region'].value_counts()
```

Out[32]:
```
southeast    364
southwest    325
northwest    325
northeast    324
Name: region, dtype: int64
```

In [33]:
```
# there is slightly hike in the customers from "southeast" region and the rest of the regions having similar no. of customers.
```

In [ ]:
```
#  from the above analysis we finds that :-
# 1. "SEX" :- male customers are slightly higher the female customers.
# 2."CHILDREN" :- Majority of the customers having childers  0 > 1 > 2 > 3.
# 3. "SMOKER" :- Majority of the customers are NON-SMOKER
# 4. "REGION" :- the distribution of customers are mostly same, but there is slightly in "southeast" region.
```

In [34]:
```
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 [35]:
```
#  here from the above information we are getting that, initially we didn't see null values, we can also conform it by "heatmap"
# total no. of columns, different datatypes, memory usage etc.
```

In [36]:
```
df.describe()
```

Out[36]:

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

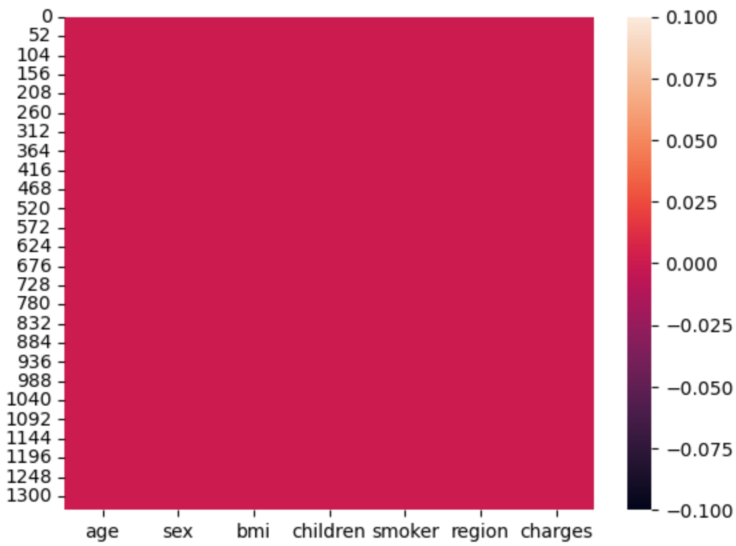
In [37]:
```
#  here we are getting information like count, mean,std,min,max,25%,50% and 75%
#  here we observe that there is huge difference between 75 percentile & max of "charges" column.
#  so from this we can assume that the presence of OUTLIERS is there in the "charges" column.
```

In [38]:
```
#  CHEKING NULL VALUES -->
```

In [39]:
```
df.isnull().sum()
```

Out[39]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

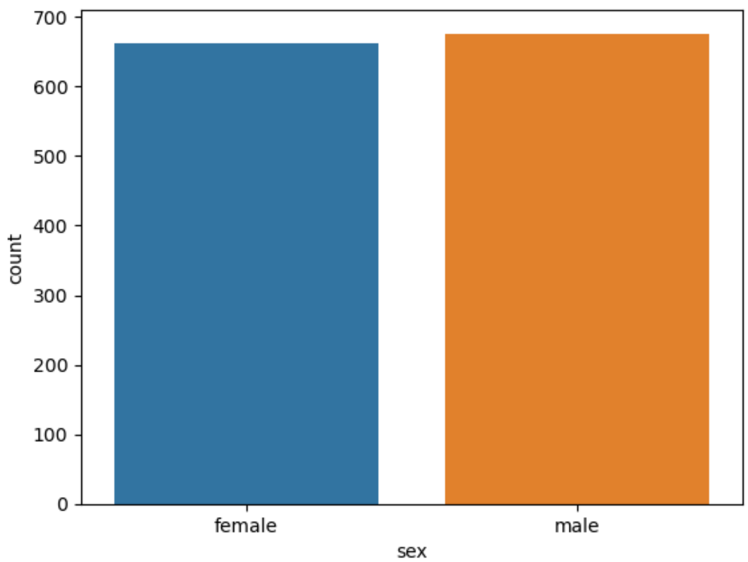
In [40]:
```
# here we can confrom that there is no null values are present in the dataset
```

In [41]:
```
sns.heatmap(df.isnull())
```
Out[41]:
```
<AxesSubplot:>
```


In [ ]:
```
#  here again conforming of absense of null values.
```
In [42]:
```
#  CORRELATION =====>>>
```
In [43]:
```
#  Now for doing some graphical representation we are seprating the cetagorical columnd and numerical columns.
```
In [44]:
```
categorical = []
numerical = []

# here above we are making two empty lsits and then we can put all the columns according to both categoris
```
In [46]:
```
df.dtypes
```
Out[46]:
```
age           int64
sex          object
bmi         float64
children      int64
smoker       object
region       object
charges     float64
dtype: object
```
In [47]:
```
for i in df.dtypes.index:
    if df.dtypes[i] == "object":
        categorical.append(i)
    else:
        numerical.append(i)
```
In [48]:
```
categorical

#  here we can see that the columns "sex", "smoker" and "region" whose type is object is inserted in the categorical list
```
Out[48]:
```
['sex', 'smoker', 'region']
```
In [49]:
```
numerical

#  and the rest of the columns age,bmi,children and charges , whose data type is int64 or float64 is inserted in the numerical co
```
Out[49]:
```
['age', 'bmi', 'children', 'charges']
```
In [ ]:
```
# NOW UPTO HERE WE ARE COMPLETED NON GRAPHICAL ANALYSIS, NOW WE HAVE TO SOME GRAPHICAL ANALYSIS ON THE DATASET TO SOME MORE FINDI
```
UNIVARIATE ANALYSIS ===

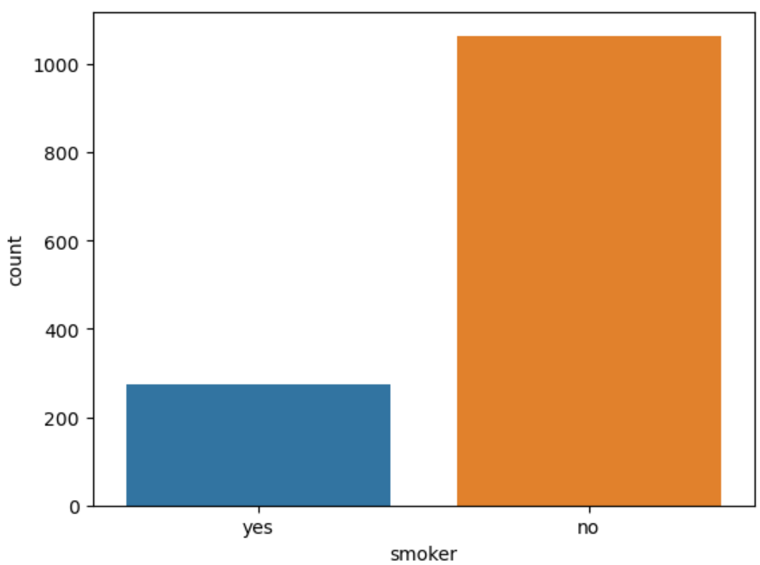
In [51]:
```
sns.countplot(x='sex', data=df)
# here we can see the slightly difference between male and femal customers.
```

Out[51]: <AxesSubplot:xlabel='sex', ylabel='count'>



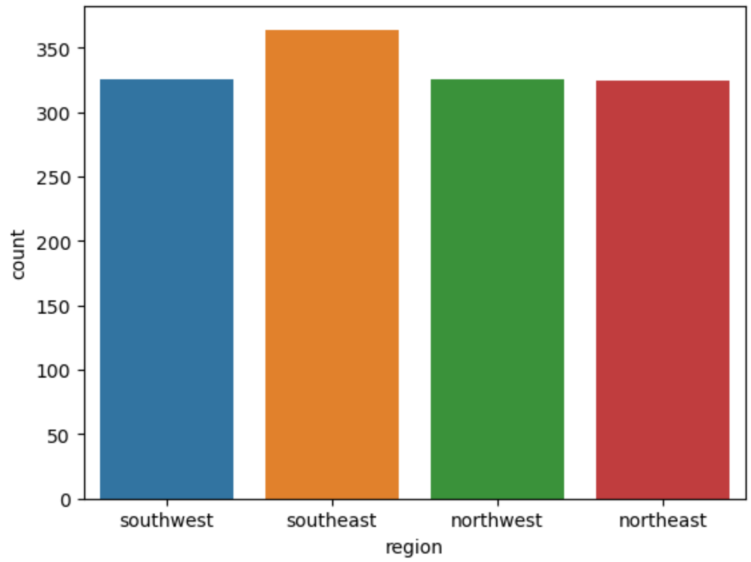
In [52]:
```
sns.countplot(x='smoker', data=df)

#there is huge differnce between the NON-SMOKER & SMOKER customers
```

Out[52]: <AxesSubplot:xlabel='smoker', ylabel='count'>



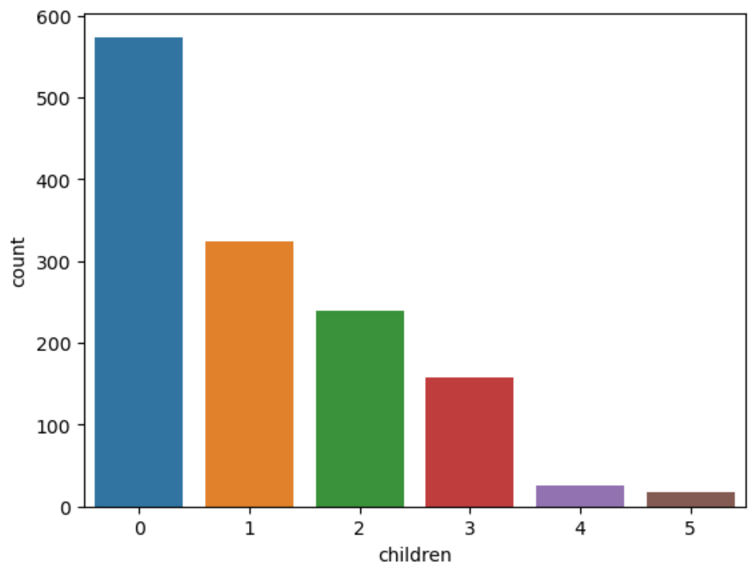
In [53]:
```
sns.countplot(x='region', data=df)

# the customer base is equally distributed , but there is a slight hike in "southeast" region
```

Out[53]: <AxesSubplot:xlabel='region', ylabel='count'>



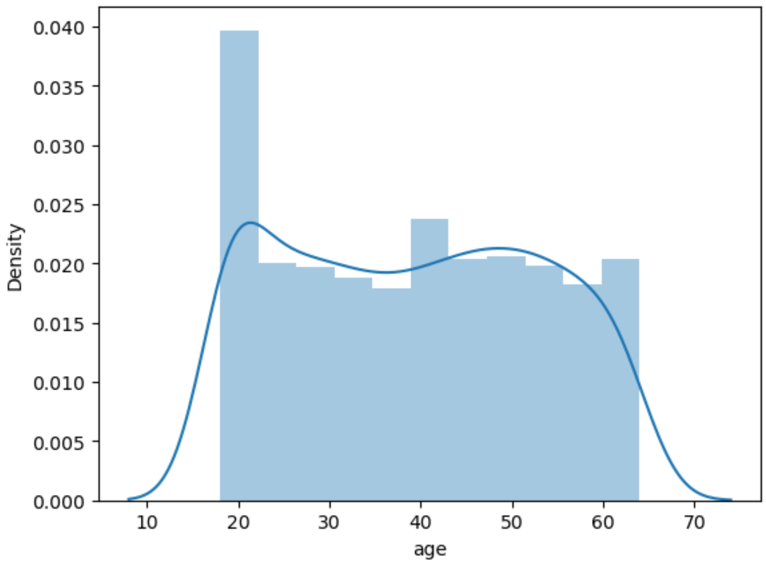
In [54]:
```
sns.countplot(x='children', data=df)

# the highest customers are having no childers, and the deacreasing towards 1 , 2, 3
#  there are only few customers whose having 4, 5 chidren.
```

Out[54]: <AxesSubplot:xlabel='children', ylabel='count'>



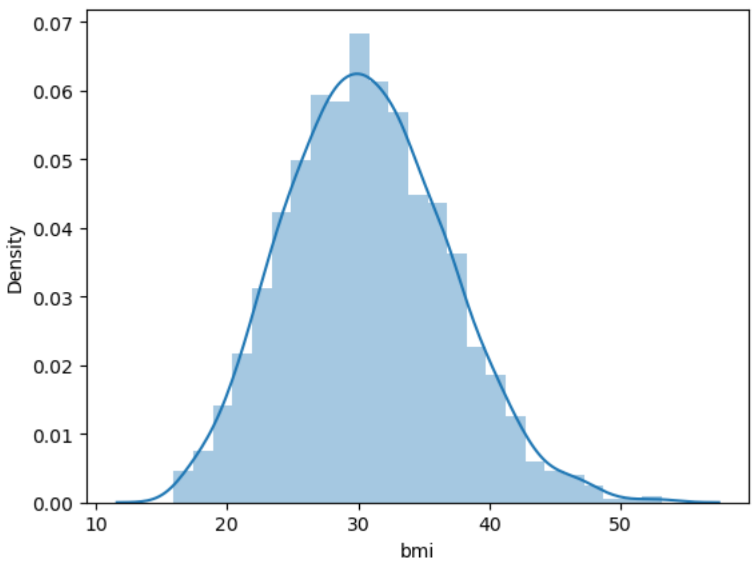
In [62]:
```
sns.distplot(df['age'])
# here we can see the age distribution of the customers
# there is a hike on the 20th year old customers,
# from this we can observe that in initial age the peoples are buying MEDICAL POLICIES
```

Out[62]: <AxesSubplot:xlabel='age', ylabel='Density'>



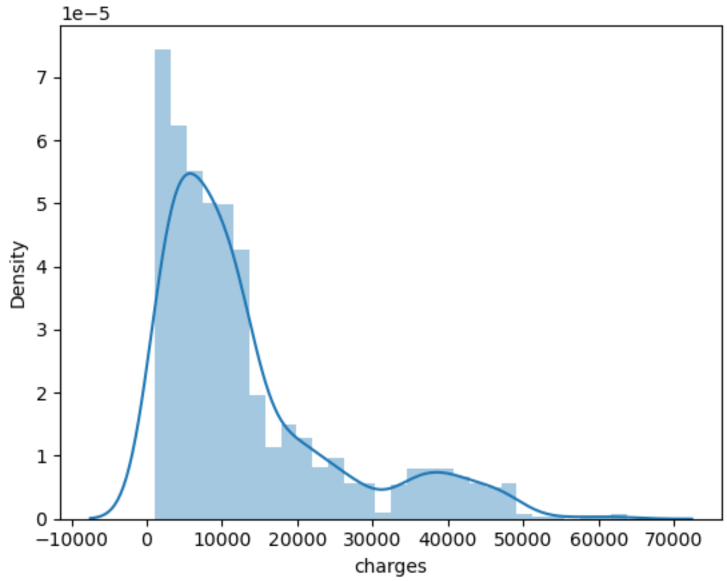
In [63]:
```
sns.distplot(df['bmi'])
# here the distribution of bmi (body mass index)
```

Out[63]: <AxesSubplot:xlabel='bmi', ylabel='Density'>



In [64]:
```
sns.distplot(df['charges'])
# below we can see that the majority of distribution of charges is lying beteween 10k to 15k
```

Out[64]: <AxesSubplot:xlabel='charges', ylabel='Density'>



In [65]:
```
# upto here we are completed our UNIVARIATE ANALYSIS for individual columns:
```

In [ ]:
```
#  now proceeding towards BIVARIATE ANALYSIS ====>>>>>
```

BIVARIATE ANALYSIS ==

In [66]:
```
categorical
```

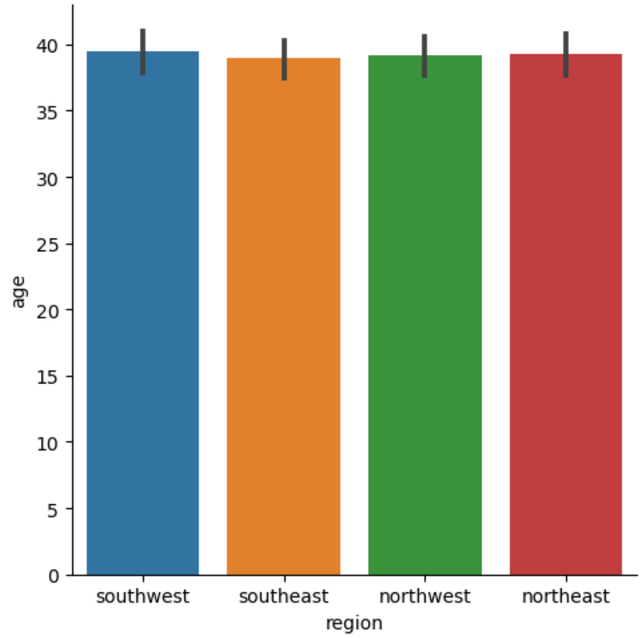
Out[66]: ['sex', 'smoker', 'region']

In [57]:
```
numerical
```

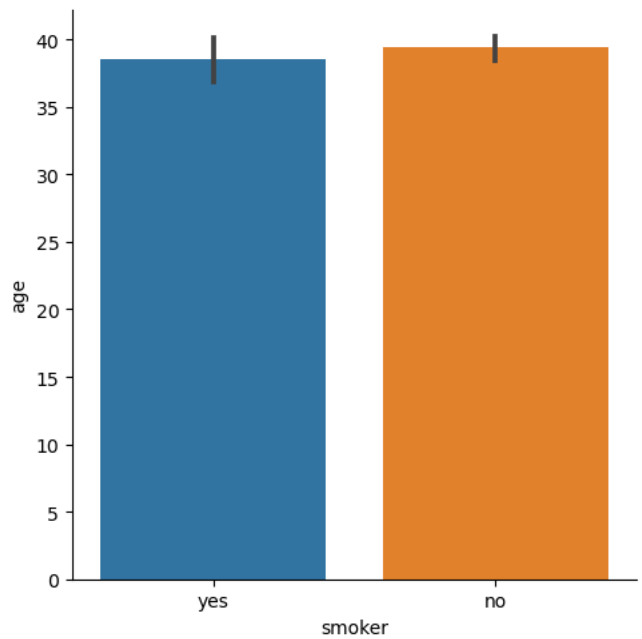
Out[57]: ['age', 'bmi', 'children', 'charges']

In [ ]:
```
# now here in BIVARIATE ANALYSIS, we analyse relation between two columns:
# for this first we are taking "sex" & "smoker"
```

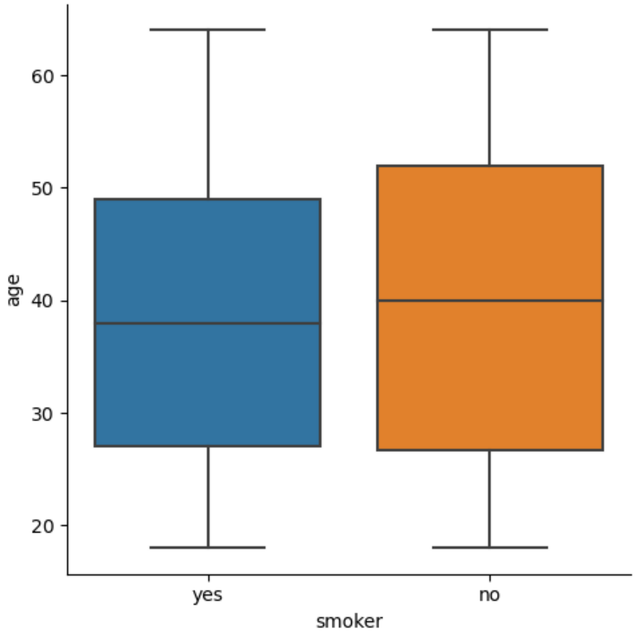
In [106]:
```
sns.catplot (x = 'region', y = 'age', data = df, kind = "bar")
plt.show()

# Here we can see the "region" wise age distribution of the customers.
```



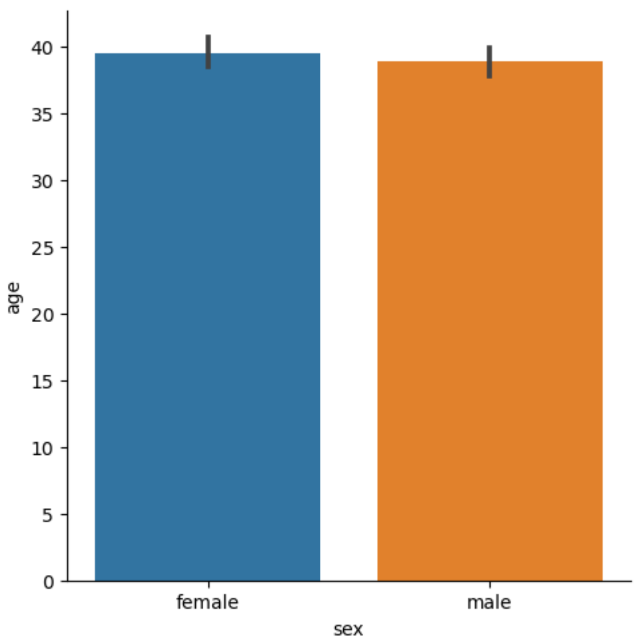
In [110]:
```
sns.catplot (x = 'smoker', y = 'age', data = df, kind = "bar")
plt.show()
# the age of smokers are mostly similar
```



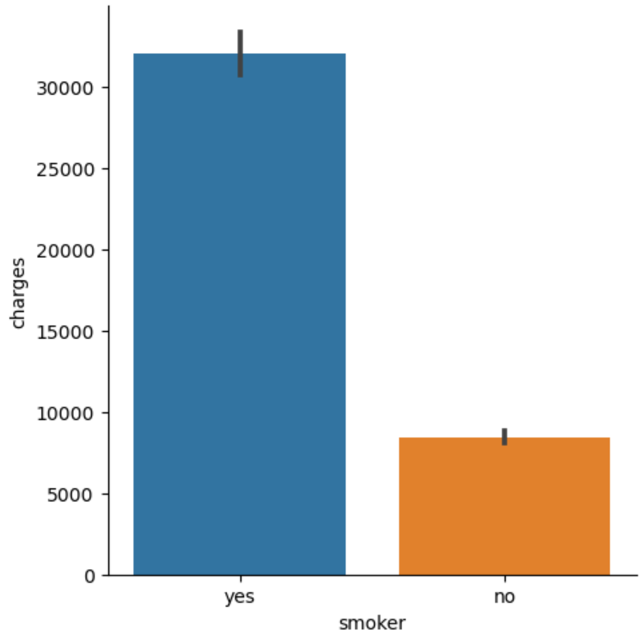
In [112]:
```
sns.catplot (x = 'smoker', y = 'age', data = df, kind = "box")
plt.show()
#  here we can see the minimum age of the "yes smoker" are below 20 which not good for their health and for insurance company als
#  the average age of "yes smoker" is lying between aprrox 28 to 50
```



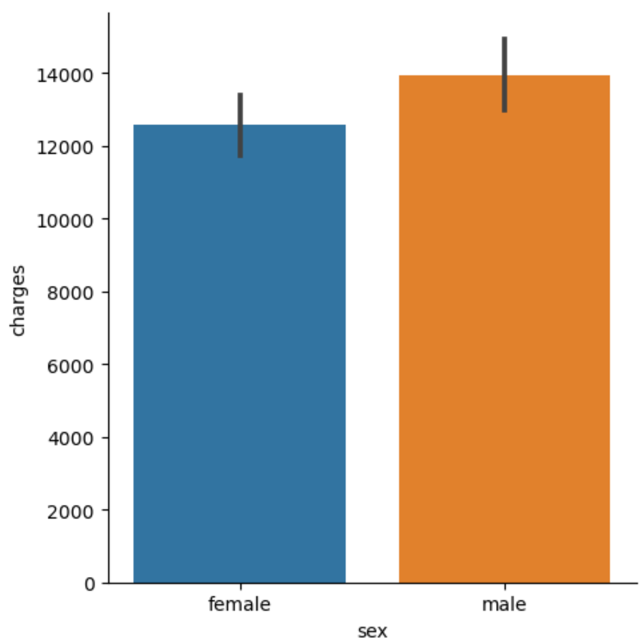
In [111]:
```
sns.catplot (x = 'sex', y = 'age', data = df, kind = "bar")
plt.show()
# the age distribution for male and female
```



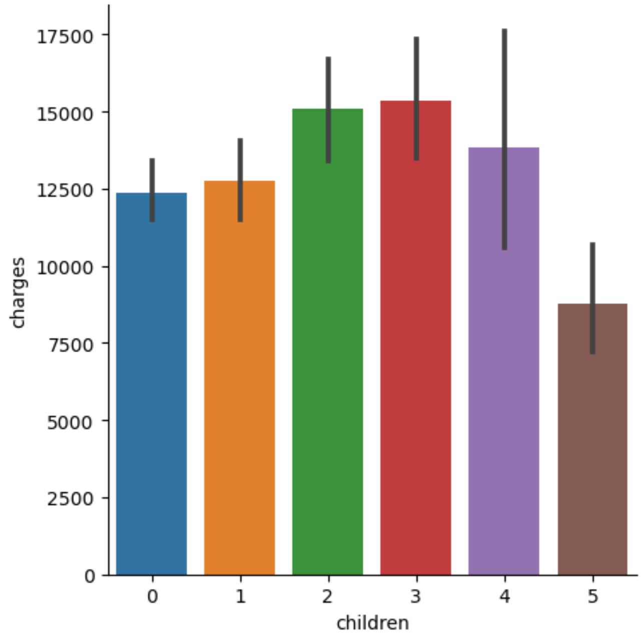
In [113]:
```
sns.catplot (x = 'smoker', y = 'charges', data = df, kind = "bar")
plt.show()
# here we can see that "YES SMOKERS" are paying verymuch high INSURANCE PREMIUM as compared to "NO SMOKER"
```



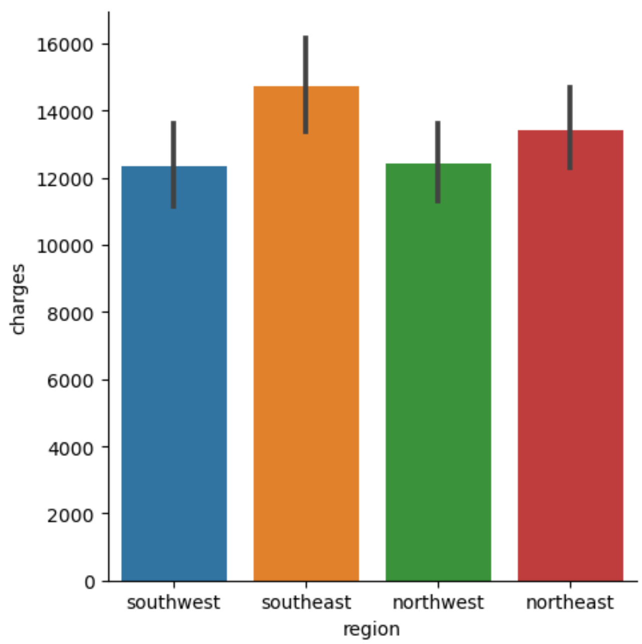
In [114]:
```
sns.catplot (x = 'sex', y = 'charges', data = df, kind = "bar")
plt.show()

#  the numbers of "male customers" are paying higher premium as compared to "female customers"
```



In [116]:
```
sns.catplot (x = 'children', y = 'charges', data = df, kind = "bar")
plt.show()

# the customers who are having children 2, 3 or 4 they are paying higher premium charges as compared to others.
```

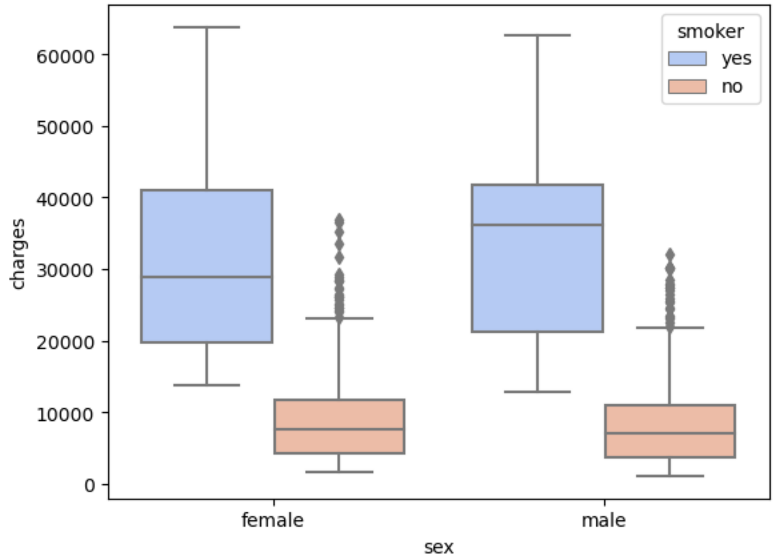


In [117]:
```
sns.catplot (x = 'region', y = 'charges', data = df, kind = "bar")
plt.show()

# the "south east" and "north east" customers are paying more premiumm as compared to others.
```




```
MULTI-VARIATE ANALYSIS =====================================================================================================
```

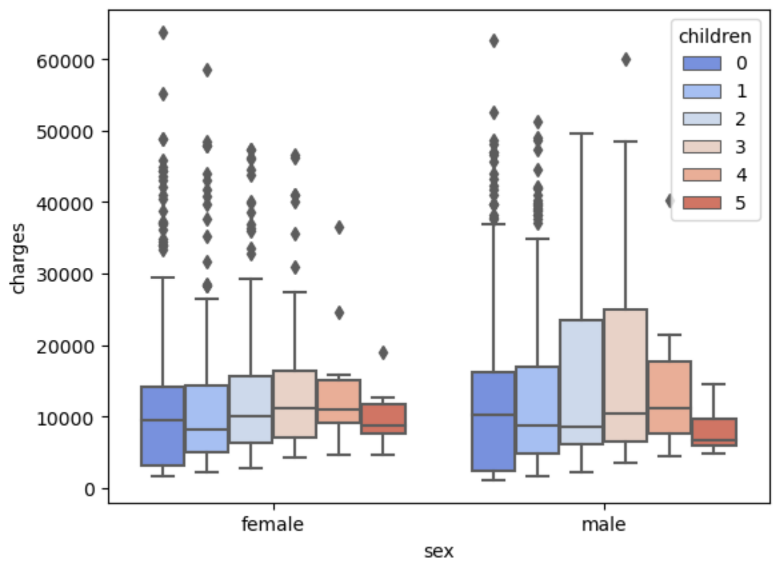
In [126]:
```
sns.boxplot (x= 'sex', y = 'charges', hue = 'smoker', data= df, palette = "coolwarm")
plt.show()

# here we can clearly see that (" female with yes smoker" and "male with yes smoker" are paying very higher premiums as comapred
# from this we can also conclude that the company is charging higher premium from the "yes smokers" wheather they are male or fem
```



In [127]:
```
categorical,numerical
```

Out[127]:
```
(['sex', 'smoker', 'region'], ['age', 'bmi', 'children', 'charges'])
```

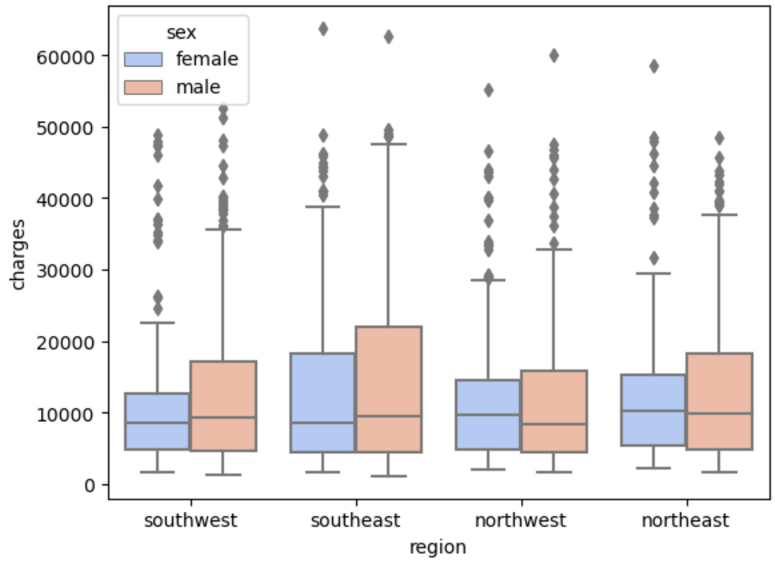
In [130]:
```
sns.boxplot (x= 'sex', y = 'charges', hue = 'children', data= df, palette = "coolwarm")
plt.show()
#  here from the below data we observ that the customers having 2-3 chidren are paying more premium
```



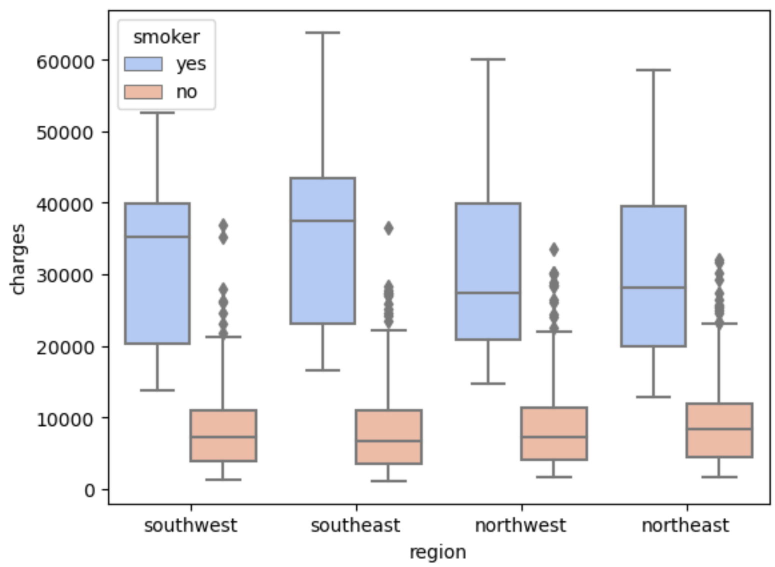
In [138]:
```
categorical, numerical
```

Out[138]:
```
(['sex', 'smoker', 'region'], ['age', 'bmi', 'children', 'charges'])
```

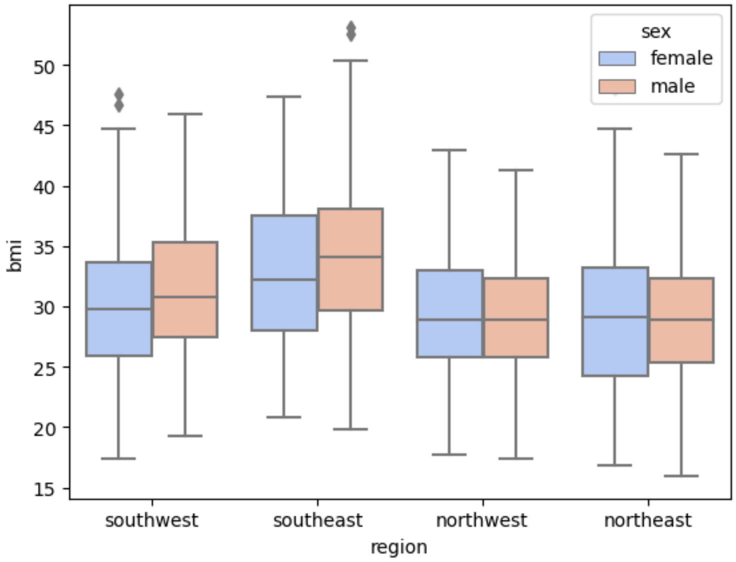
In [140]:
```
sns.boxplot (x= 'region', y = 'charges', hue = 'sex', data= df, palette = "coolwarm")
plt.show()
# here we can see that in all four regions "males" are paying more premium as compared to womens
```



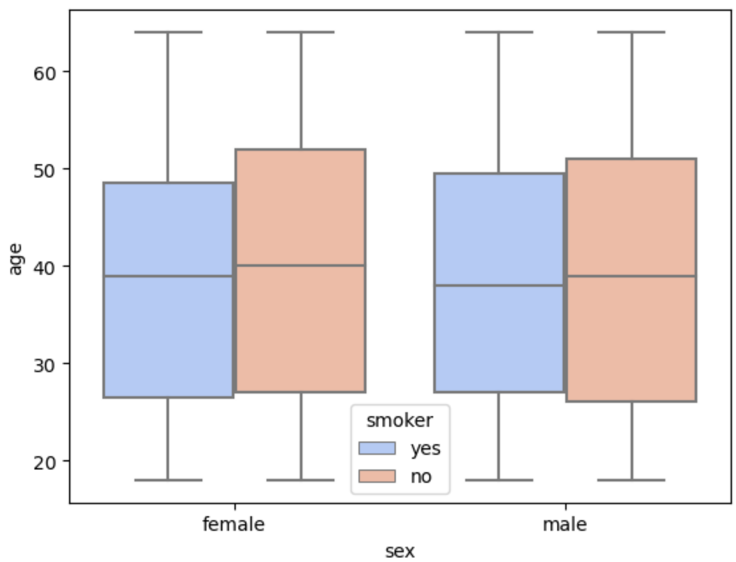
In [141]:
```
sns.boxplot (x= 'region', y = 'charges', hue = 'smoker', data= df, palette = "coolwarm")
plt.show()
# it is clear from the below plot that in all four regions the "yes smokers" are paying more premium as compared to others.
#  that means company charging more premium from "yes smokers" in all the regions
```



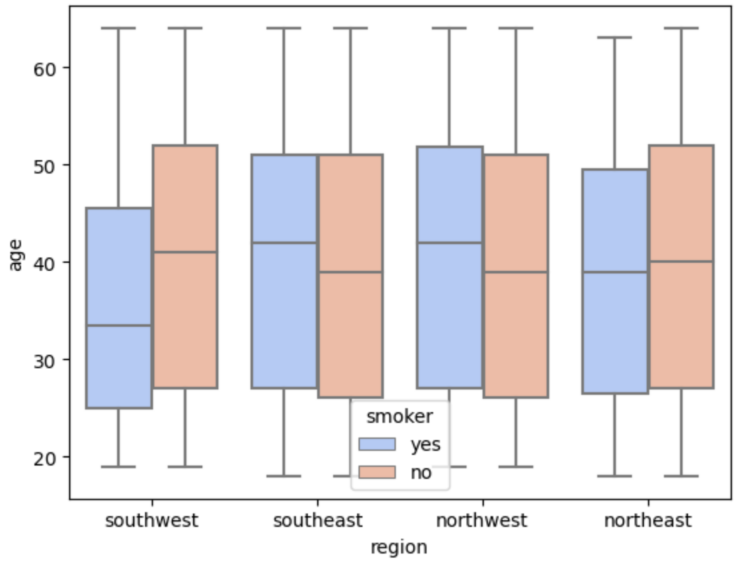
In [142]:
```
sns.boxplot (x= 'region', y = 'bmi', hue = 'sex', data= df, palette = "coolwarm")
plt.show()
# the "bmi" BODY MASS INDEX of SOUTHEAST REGION customrs are higher as compared to others.
```



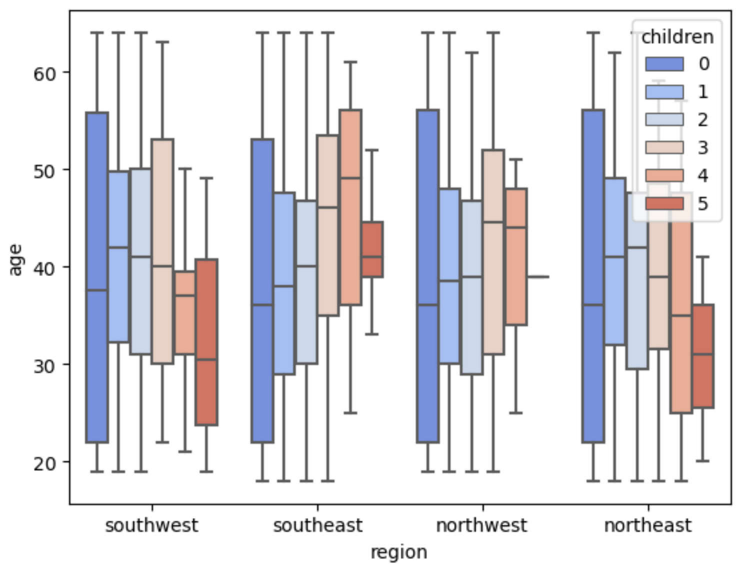
In [145]:
```
sns.boxplot (x= 'sex', y = 'age', hue = 'smoker', data= df, palette = "coolwarm")
plt.show()
# the 75 percentile age of the "yes smoker" wether they "male & female" is lesser as compared to "no smoker"
```



In [146]:
```
sns.boxplot (x= 'region', y = 'age', hue = 'smoker', data= df, palette = "coolwarm")
plt.show()
# the "yes smoker" age of 75th percentile of customers from "southwest & northeast" region customers is less as compared to "south
```



In [147]:
```
sns.boxplot (x= 'region', y = 'age', hue = 'children', data= df, palette = "coolwarm")
plt.show()

# here from below it is cleared that the majority of average customers are DONT HAVING CHILDREN, from this we can understood that
#  nowadays the people are aware to opt insurance in early age also.
```



In [ ]:
```
# here above we are completed our BIVARIATE & MULTI-VARIATE ANALYSIS==========================================================
```

CORRELATION =======>>>

In [149]:
```
# now we have to check is there any correlation between columns and the target (charges)
# but before applyin correlation we have to aplly encoding techniques
```

In [ ]:

In [80]:
```
df
```

Out[80]:

	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

In [ ]:
```
ENCODIND TECHNIQUES ====>>>>>>>>
```

In [151]:
```
categorical
# here we are having 3 categorical columns whose datatype is "object", so first we have to encode them .
```

Out[151]: ['sex', 'smoker', 'region']

In [153]:
```
# here we are apllying "LABEL ENCODER" for all the categorical columns:-
#  for which we have to import some libraries
```

In [154]:
```
from sklearn.preprocessing import LabelEncoder
```

In [155]:
```
le = LabelEncoder()
```

In [157]:
```
df["sex"] = le.fit_transform(df["sex"])
df

# here below you can see that the values of "sex column" has changed from " male / female" to "0 / 1"
#  here "0 = FEMALE" & "1 = MALE"
```

Out[157]:

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

1338 rows × 7 columns

In [ ]:
```
#  similarly we can apply the LABEL ENCODER on "smoker" and " region" also
```

In [159]:
```
df["smoker"] = le.fit_transform(df["smoker"])
df["region"] = le.fit_transform(df["region"])
df

#  now "smoker" and "region" columns are also changed from "object" to "numerical" datatype
```

Out[159]:

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

1338 rows × 7 columns

In [161]:
```
df.dtypes
# here you can see the change in data type of "sex" "smoker" and "region"
```

Out[161]:
```
age           int64
sex           int32
bmi         float64
children      int64
smoker        int64
region        int64
charges     float64
dtype: object
```

CORRELATIONS ====>>>

In [ ]:
```
# NOW WE CHECK THE CORRELAITON BETWEEN THE COLUMNS
```

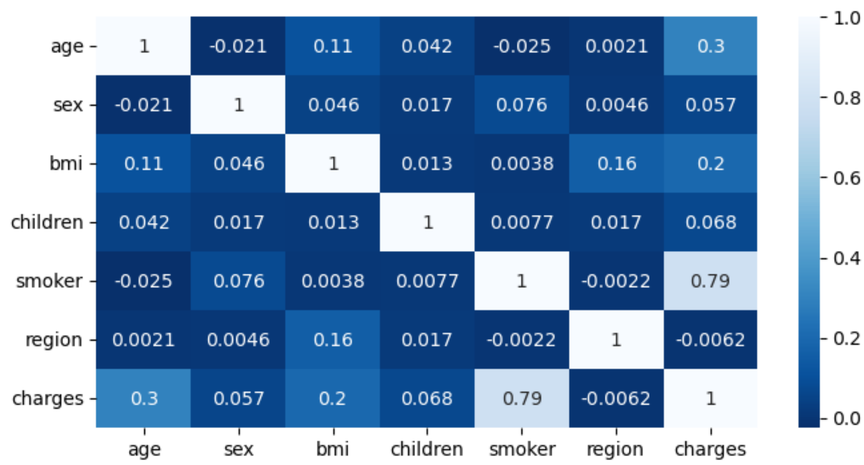
In [162]:
```
dfcor = df.corr()
dfcor
```

Out[162]:

	age	sex	bmi	children	smoker	region	charges
age	1.000000	-0.020856	0.109272	0.042469	-0.025019	0.002127	0.299008
sex	-0.020856	1.000000	0.046371	0.017163	0.076185	0.004588	0.057292
bmi	0.109272	0.046371	1.000000	0.012759	0.003750	0.157566	0.198341
children	0.042469	0.017163	0.012759	1.000000	0.007673	0.016569	0.067998
smoker	-0.025019	0.076185	0.003750	0.007673	1.000000	-0.002181	0.787251
region	0.002127	0.004588	0.157566	0.016569	-0.002181	1.000000	-0.006208
charges	0.299008	0.057292	0.198341	0.067998	0.787251	-0.006208	1.000000

In [166]:
```
plt.figure(figsize=(8,4))
sns.heatmap(dfcor,annot=True,cmap="Blues_r")
```

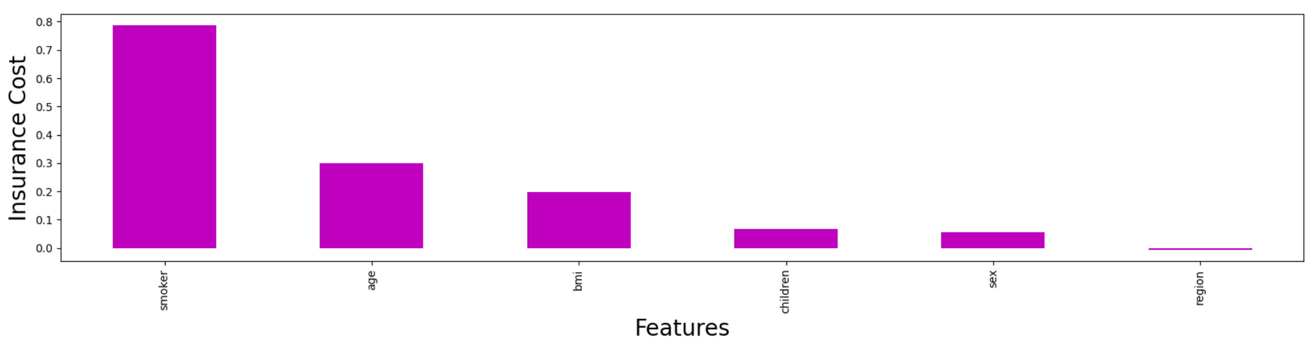
Out[166]: <AxesSubplot:>



In [ ]:
```
# here above in the plot we can see the correlation between " CHARGES & SMOKER"
```

In [ ]:
```
 '''' as from above now it is easy to identify the highly positive and negative correlated columns.
1. the colour from dark blue to white is showing the incresing the (+)ve correlation, white colour = highly (+)ve correlation.
2. the dark blue colour is showing the highly (-)ve correlation.
3. As from the heatmap we can finds that the correlation between "CHARGES" & "SMOKER" is very highly (+)ve. = .79
 ''''
```

In [168]:
```
plt.figure(figsize=(20,4))
df.corr()['charges'].sort_values(ascending=False).drop(['charges']).plot(kind='bar',color="m")
plt.xlabel('Features',fontsize=20)
plt.ylabel('Insurance Cost',fontsize =20)
plt.title=("Insurance Project")
plt.show()
```



In [ ]:
```
#  here from the above also it is cleared that that the INSURANCE PREMIUM CHARGES are HIGHLY CORRELATED with "SMOKER" & "AGE"
```

In [170]:
```
df.describe()
```

Out[170]:

	age	sex	bmi	children	smoker	region	charges
count	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	0.505232	30.663397	1.094918	0.204783	1.515695	13270.422265
std	14.049960	0.500160	6.098187	1.205493	0.403694	1.104885	12110.011237
min	18.000000	0.000000	15.960000	0.000000	0.000000	0.000000	1121.873900
25%	27.000000	0.000000	26.296250	0.000000	0.000000	1.000000	4740.287150
50%	39.000000	1.000000	30.400000	1.000000	0.000000	2.000000	9382.033000
75%	51.000000	1.000000	34.693750	2.000000	0.000000	2.000000	16639.912515
max	64.000000	1.000000	53.130000	5.000000	1.000000	3.000000	63770.428010

In [ ]:
```
''''' ".describe" is a very important function which gives a lot of following important information about the dataset:-

 1. here we can see the total no. counts present in each columns, i.e [1338] and all columns are showing the whole count that
    means there are no missing values in the given dataset.
 2. It also shows the MEAN, STANDARD DEVIATION of each column.
 3. It also provides us the MIN, MAXIMUM, 25%, 50% (median), 75% Values of every column.
 4. IMP POINT:- we know that IF THE MEAN > MEDIAN = data is Right Skewed
                        and  IF THE MEAN < median = data is left Skewed, so we can the skewness in the following columns:-
            (a) "charges" = mean < median ( left skewed)
            the rest of the columns are having very slight difference.
 5. IMP POINT :- here we also determines the "STANDARD DEVIATION" -
            so we can see that the "std" of "charges" is very high
            so from this we can conclude that there are some OUTLIERS are may present.
 6. As we can also see that the difference between "75 Percentile & Maximum" is very high in "charges", "age" & " bmi",
            so from this also we can assume that there maybe OUTLIERS are present in all 3 of the columns.
    ''''
```

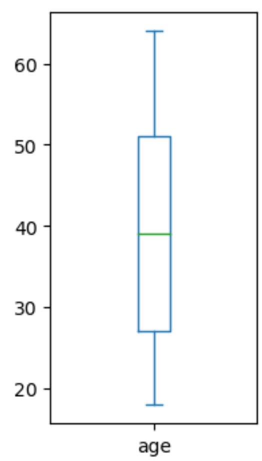
CHECKING FOR OUTLIERS ============>>>>>>>>>>>>>>

In [178]:
```
categorical,numerical
```

Out[178]: (['sex', 'smoker', 'region'], ['age', 'bmi', 'children', 'charges'])

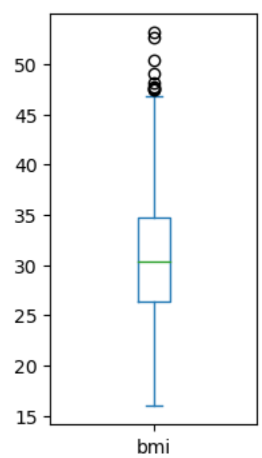
In [206]:
```
plt.figure(figsize=(2,4))
df['age'].plot.box()
# df.boxplot(['age'])
# there are no outliers in the "age" column
```

Out[206]: <AxesSubplot:>



In [207]:
```
plt.figure(figsize=(2,4))
df['bmi'].plot.box()
# df.boxplot(['bmi'])
# here we can see the presence of outliers in the "bmi" column but they are acceptable range and we dont need to remove them.
```

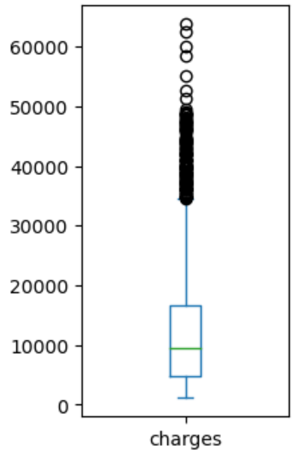
Out[207]: <AxesSubplot:>



In [ ]:

In [208]:
```
plt.figure(figsize=(2,4))
df['charges'].plot.box()
# df.boxplot(['charges'])
# here we can see there a numbers of outliers are present in the "charges" column
#  BUT CAN'T REMOVE OUTLIERS FROM THE TAGET COLUMN, and here "CHARGES" is our TARGET COLUMN.
```

Out[208]: <AxesSubplot:>



In [ ]:
```
# so the conclusion from the above is there is no presence of OUTLIERS in the columns which have to be removed from the data.
# so now we can procced further.
```

CHECKING FOR SKEWNESS =====>>>>>>

In [194]:
```
df.skew()
```

Out[194]:
```
age         0.055673
sex        -0.020951
bmi         0.284047
children    0.938380
smoker      1.464766
region     -0.038101
charges     1.515880
dtype: float64
```

In [ ]:
```
# here also we didn't a skewness in any column which have to remove.
```

SCALING OF DATASET ======>>>>>

In [195]:
```
df
#  here in our dataset we can see the differences between the numerical of various columns, which can be affect our model to
#  to predict the result, so first we have to scale our dataset before apllyin model.
```

Out[195]:

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

1338 rows × 7 columns

In [196]:
```
from sklearn.preprocessing import StandardScaler
```

In [198]:
```
st = StandardScaler()
```

In [ ]:
```
#  now here we have to remember one thing that we can aplly scaling techiniques on only our "FEATURES" not on "TARGET COLUMN"
```

In [202]:
```
x= df.iloc[:,:-1]
x.shape

#  "x" is our "features" column
```

Out[202]: (1338, 6)

In [203]:
```
y = df.iloc[:,-1]
y.shape

# "y" is our "target" column
```

Out[203]: (1338,)

In [ ]:
```
# here above we are splitting the data in 'x' & 'y'
#  where 'x' is the fetures & 'y' is our target column.
```

In [204]:
```
x= st.fit_transform(x)
x

# here we are applying scaling technique on all "features columns"
```

Out[204]:
```
array([[-1.43876426, -1.0105187 , -0.45332   , -0.90861367,  1.97058663,
         1.34390459],
       [-1.50996545,  0.98959079,  0.5096211 , -0.07876719, -0.5074631 ,
         0.43849455],
       [-0.79795355,  0.98959079,  0.38330685,  1.58092576, -0.5074631 ,
         0.43849455],
       ...,
       [-1.50996545, -1.0105187 ,  1.0148781 , -0.90861367, -0.5074631 ,
         0.43849455],
       [-1.29636188, -1.0105187 , -0.79781341, -0.90861367, -0.5074631 ,
         1.34390459],
       [ 1.55168573, -1.0105187 , -0.26138796, -0.90861367,  1.97058663,
        -0.46691549]])
```

In [205]:
```
xf= pd.DataFrame(data=x)
xf

# making datafame of scalling apllied data
```

Out[205]:

	0	1	2	3	4	5
0	-1.438764	-1.010519	-0.453320	-0.908614	1.970587	1.343905
1	-1.509965	0.989591	0.509621	-0.078767	-0.507463	0.438495
2	-0.797954	0.989591	0.383307	1.580926	-0.507463	0.438495
3	-0.441948	0.989591	-1.305531	-0.908614	-0.507463	-0.466915
4	-0.513149	0.989591	-0.292556	-0.908614	-0.507463	-0.466915
...	...	...	...	...	...	...
1333	0.768473	0.989591	0.050297	1.580926	-0.507463	-0.466915
1334	-1.509965	-1.010519	0.206139	-0.908614	-0.507463	-1.372326
1335	-1.509965	-1.010519	1.014878	-0.908614	-0.507463	0.438495
1336	-1.296362	-1.010519	-0.797813	-0.908614	-0.507463	1.343905
1337	1.551686	-1.010519	-0.261388	-0.908614	1.970587	-0.466915

1338 rows × 6 columns

In [235]:
```
column = ['age', 'sex', 'bmi', 'children', 'smoker', 'region']
```

In [236]:
```
xf.columns = column
```

In [237]:
```
xf
```

Out[237]:

	age	sex	bmi	children	smoker	region
0	-1.438764	-1.010519	-0.453320	-0.908614	1.970587	1.343905
1	-1.509965	0.989591	0.509621	-0.078767	-0.507463	0.438495
2	-0.797954	0.989591	0.383307	1.580926	-0.507463	0.438495
3	-0.441948	0.989591	-1.305531	-0.908614	-0.507463	-0.466915
4	-0.513149	0.989591	-0.292556	-0.908614	-0.507463	-0.466915
...	...	...	...	...	...	...
1333	0.768473	0.989591	0.050297	1.580926	-0.507463	-0.466915
1334	-1.509965	-1.010519	0.206139	-0.908614	-0.507463	-1.372326
1335	-1.509965	-1.010519	1.014878	-0.908614	-0.507463	0.438495
1336	-1.296362	-1.010519	-0.797813	-0.908614	-0.507463	1.343905
1337	1.551686	-1.010519	-0.261388	-0.908614	1.970587	-0.466915

1338 rows × 6 columns

```
FINDING MULICOLLINEARITY =====>>>>>
```

In [ ]:
```
# to find the multicollinearity between the features  and remove it we can use VIF (VARIANCE INFLATION FACTOR)
# we can not aplly VIF on the TARGET COLUMN
# for apllyin VIF we have to import some libraries as follows
```

In [209]:
```
import statsmodels.api as sm
from scipy import stats
from statsmodels .stats.outliers_influence import variance_inflation_factor
```

In [243]:
```
# here we are making "def function" for calculating VIF
def calc_vif(xf):
    vif = pd.DataFrame()
    vif["FETURES"] = xf.columns
    vif["VIF FACTOR"] = [variance_inflation_factor(xf.values,i) for i in range (xf.shape[1])]
    return (vif)
```

In [239]:
```
xf.shape
```

Out[239]:
```
(1338, 6)
```

In [244]:
```
calc_vif(xf)
```

Out[244]:

	FETURES	VIF FACTOR
0	age	1.015394
1	sex	1.008889
2	bmi	1.040608
3	children	1.002482
4	smoker	1.006466
5	region	1.025966

In [ ]:
```
# here above we can see that there is no multicollinearity between the columns , so we no need to remove mulitcollienearity
```

```
RESAMPLING TECHNIQUES =====>>>>>>
```

In [245]:
```
y.value_counts()
```

Out[245]:
```
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: charges, Length: 1337, dtype: int64
```

In [ ]:
```
# as we can see that our target column is not a categorical , it a floating data, se also no need apply RESAMPLIN TECHNIQUES to t
```

APPLYING ML MODEL ============>>>>>>>>

''''' NOW HERE WE CAN SEE THAT OUR TARGET/LABEL COLUMN IN NOT A CATEGORICAL DATA, IT IS HAVING FLOATING DATA, AND WHEN WE ARE HAVING "Y" (TARGET) IN DECIMAL FORM THEN WE CAN APPLY "LINEAR REGRESSION MODEL", SO HERE WE CAN APPLY LINEAR REGRESSION MODEL ON OUR DATASET TO PREDICT, "INSURANCE COSTS".''''

In [246]:
```
from sklearn.linear_model import LinearRegression
```

In [259]:
```
lr = LinearRegression()
```

In [250]:
```
xf.shape
```

Out[250]:
```
(1338, 6)
```

In [258]:
```
y.shape
```

Out[258]:
```
(1338,)
```

In [272]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import mean_squared_error
```

In [264]:
```
lr = LinearRegression()
```

In [265]:
```
 x_train,x_test,y_train,y_test = train_test_split(xf,y,test_size=0.20,random_state=42)
```

In [273]:
```
lr.fit(x_train,y_train)
print(lr.score(x_train,y_train))
lrpred = lr.predict(x_test)
mse = mean_squared_error(y_test, lrpred)
print(f"Mean Squared Error: {mse}")
```

```
0.7417049283233981
Mean Squared Error: 33635210.43117845
```

In [275]:
```
print("Coefficients:", lr.coef_)
print("Intercept:", lr.intercept_)
```

```
Coefficients: [ 3.61028043e+03 -9.39521400e+00  2.04689296e+03  5.12253132e+02
  9.54291505e+03 -2.99625864e+02]
Intercept: 13315.44519213977
```

In [ ]:
```
#  here i get some difficulties in apllying the model.
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [263]:
```
maxaccu = 0
maxrs = 0

for i in range(1,200):
    x_train,x_test,y_train,y_test = train_test_split(xf,y,test_size=0.20,random_state=i)
    lr = LinearRegression()
    lr.fit(x_train,y_train)
    pred = lr.predict(x_test)
    acc = accuracy_score(y_test,pred)

    if acc > maxaccu :
        maxaccu = acc
        maxrs = i

print ("Best accuracy is",maxaccu, "at random state", maxrs)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_6284\3644937284.py in <module>
      7     lr.fit(x_train,y_train)
      8     pred = lr.predict(x_test)
----> 9     acc = accuracy_score(y_test,pred)
     10 
     11     if acc > maxaccu :

~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py in accuracy_score(y_true, y_pred, normalize, sample_weight)
    209 
    210     # Compute accuracy for each possible representation
--> 211     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
    212     check_consistent_length(y_true, y_pred, sample_weight)
    213     if y_type.startswith("multilabel"):

~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py in _check_targets(y_true, y_pred)
    102     # No metrics support "multiclass-multioutput" format
    103     if y_type not in ["binary", "multiclass", "multilabel-indicator"]:
--> 104         raise ValueError("{0} is not supported".format(y_type))
    105 
    106     if y_type in ["binary", "multiclass"]:

ValueError: continuous is not supported
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: