Health Insurance Python Codes| Viz outputs

1.1 Import data libraries from pandas, NumPy and seaborn and matplotlib. Then load the and read the csv insurance data

Table: 1.1.Import of libraries

```
[2]: # importing libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

[3]: # loading and reading dataset
data = pd.read_csv('insurance.csv')
```

a) Checking the first five rows of the data. below is the output.

Table: 1.2. Five Rows of the Dataset

```
[5]: # data inspection by checking the first five rows of the data
print(data.head())

   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
```

b) Checking the last five rows of the dataset. Below is the output

Table: 1.3. Last Five Rows of the dataset

```
[14]: # data inspection by checking the last five rows of the data
print(data.tail())

      age     sex    bmi  children smoker     region     charges
1333   50    male  30.97         3     no  northwest  10600.5483
1334   18  female  31.92         0     no  northeast   2205.9808
1335   18  female  36.85         0     no  southeast   1629.8335
1336   21  female  25.80         0     no  southwest   2007.9450
1337   61  female  29.07         0    yes  northwest  29141.3603
```

- Checking the last five rows of the dataset

c) Checking how many rows and columns the dataset has ?

Table: 1.4 Data Sharp

```
[13]: # Chechking the data structure
print(f'Data structure: {data.shape}')
print(f'There are {data.shape[0]} rows in the dataset.')
print(f'and There are also {data.shape[1]} columns in the dataset.')

Data structure: (1338, 7)
There are 1338 rows in the dataset.
and There are also 7 columns in the dataset.
```

- Results showed that, there are about 1338 rows in the data and 7 columns.

1.5 Check the shape of the data along with the data types of the column

Table: 2.1 Shows the Data Types

```
[15]: # inspecting the data types in the dataset
      print(data.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
      None

[18]: # summary of the different types of datatypes present in the dataset
      data.dtypes.unique()

[18]: array([dtype('int64'), dtype('O'), dtype('float64')], dtype=object)

[19]: # Checking the individual columns in the dataset
      data.columns

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

[21]: # transforming the index into a series  and grouping by datatype
      Trans= data.columns.to_series().groupby(data.dtypes).groups
      print(Trans)

      {int64: ['age', 'children'], float64: ['bmi', 'charges'], object: ['sex', 'smoker', 'region']}
```

- line 15 results showed that we have two floot64(2), two int64 and three object datatypes.

1.6 Check for missing values in the dataset and using the appropriate measures to fill in the missing values

Table: 3.1 **Checking for Missing Values**

```
[23]: # checking for missing values
      print(data.isnull().sum())

      age         0
      sex         0
      bmi         0
      children    0
      smoker      0
      region      0
      charges     0
      dtype: int64
```

- The output results showed that, there are no missing values in the dataset.

2.Checking for duplicate values in the dataset

```
[37]: # checing for duplicate values in the dataset
      data.duplicated().sum()

[37]: 1

[38]: # droping the duplicated values in the dataset
      data=data.drop_duplicates()

[39]: # verifying if the duplicate values have been removed
      data.duplicated().sum()

[39]: 0
```

- There was one duplicated value in the dataset which was removed

3.0 Explore the relationship between the feature and target column using a count plot of categorical columns and a scatter plot of numerical columns

3.1. Describe gives the summary and distribution of the numerical attributes like age, bmi and charges
Table 5.1.1 Summary of Distributions

```
[24]: # performing summary stats
      print(data.describe())

               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
```

Output interpretation

The results shows the count of all the variable is 1338 for bmi , age, charge and children. the results also showed that the summary of the mean of age was (39.207) ,bmi (30.663) ,children(1.0949) and charges(13270.4222). the results also showed the min and max values of the age with minimum age (18)maximum (64) , bmi with minimum bmi bring (15.96)maximum (53.13), children with minimum child bring (0)maximum (5).the results also gives the standard deviation on how spaced the value are to the true value of the age(14.04) , bmi(6.098) , children(1.2) and charge(12110.01).

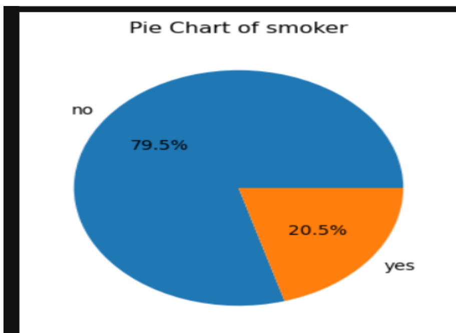
4. identifying the data variables which might be categorical in nature. Describe and explore these variables using appropriate tools.

Table 4.2: shows the different categorical data in the dataset.

```
sorted(data['sex'].unique())

['female', 'male']

sorted(data['region'].unique())

['northeast', 'northwest', 'southeast', 'southwest']
```

```
# Creating a function that return a pie chart for categorical variable

def pie_chart(x='smoker'):
    fig, ax = plt.subplots(figsize=(8, 6), subplot_kw=dict(aspect="equal"))
    s = data.groupby(x).size()
    data_values = s.values.tolist()
    labels = s.index.tolist()
    ax.pie(data_values, labels=labels, autopct='%1.1f%%')
    ax.set_title(f"Pie Chart of {x}")
    plt.show()

# Now call the function
print(pie_chart())
```

Table:5.5 : Show a pie chart for smoker of the insured people



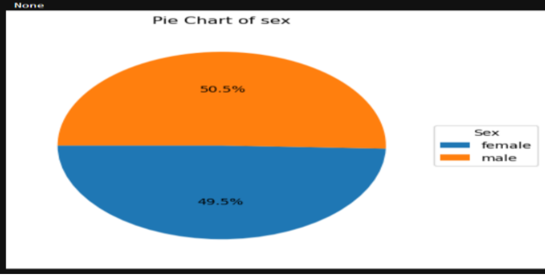
None

Table 5.6: Show a pie chart for Sex of the insured people

```python
# Creating a function that return a pie chart for categorical variable

def pie_chart(x='sex'):
    fig, ax = plt.subplots(figsize=(5, 6), subplot_kw=dict(aspect="equal"))
    s = data.groupby(x).size()
    data_values = s.values.tolist()
    wedges, texts, autotexts = ax.pie(data_values, autopct='%1.1f%%', startangle=180)
    labels = s.index.tolist()
    ax.set_title(f"Pie Chart of {x}")
    ax.legend(wedges, labels, title=x.capitalize(), loc='center left', bbox_to_anchor=(1, 0.5))
```

```python
[25]: # Now call the function
      print(pie_chart())
```

```
None
```



- The results showed that we had about 50.5% of the insured where males and 49.5% were Females.

Table:5.7 : Shows a bar chart for smokers by Sex

```python
sns.countplot(x= 'smoker', hue = 'sex', data = data )
plt.title('Gender')
print(plt.show)
```

```
<function show at 0x000001DB28044180>
```

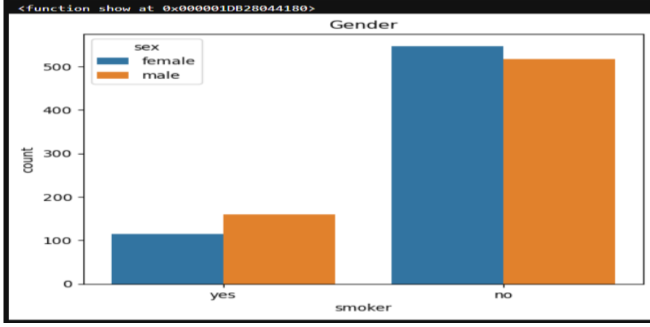
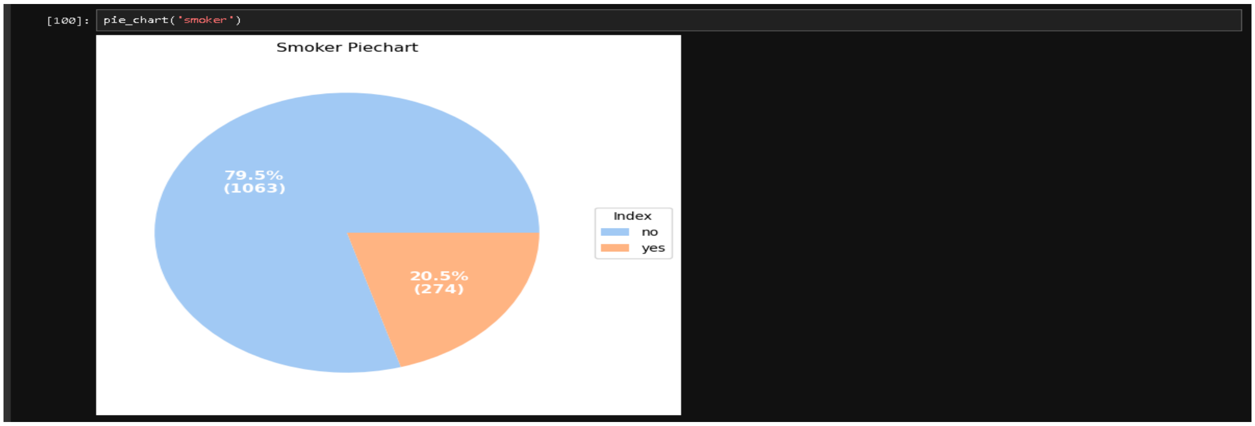


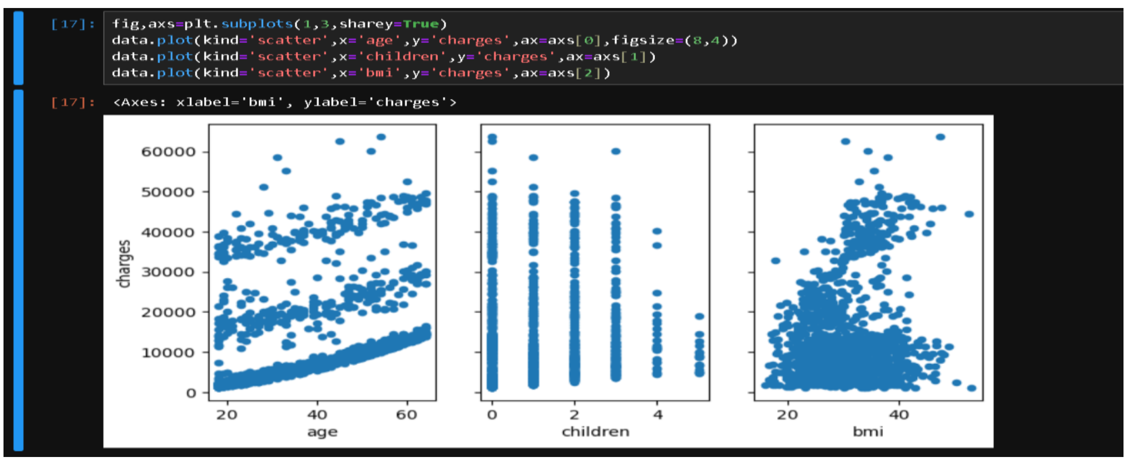
Table:5.9 Shows a pie chart for smokers by Sex



Results Interpretation

- From the total 1338 insured people about 274 (20.5%) are smokers and the rest are non-smokers.
- Among 274 smokers, proportion of males (159) are higher than females (115).
- The average insurance premium for smokers are significantly higher than non-smokers.

Table 5.3: shows the **scatter plot of numerical columns data**



Interpretation

- The results showed that the higher the bmi the higher the charger
- The higher the age the higher the charge.
  Table 4.4: shows the count plot of categorical columns

6. **Perform data visualization using plots of feature vs feature**

After reviewing the results of the data, we shall use the following steps to come up with the charts that will give appropriate data meaning.

a. We will use a density plot to show the distribution of the data.
b. We will also use the boxplot to show the output of the quantitative data in a way that shows comparisons between the variables.

c. The Boxplot and whisker plot to shows the distribution of quantitative data in a way that facilitates comparisons between variables. The box will show the quartiles of the dataset while the whiskers extend to show the rest of the distribution, except for points that are determined to be “outliers” using a method that is a function of the inter-quartile range.

d. the violin plot features a kernel density estimation of the underlying distribution.

e. The cumulative frequency also corresponds to the density distribution, indicates the central tendency of the data.

6.1 Perform data visualization using plots of feature vs feature

Figure 5.1: Show command for creating distplot, boxplots, violin plot, kdelot

```
def summary(x):
    x_min = data[x].min()
    x_max = data[x].max()
    Q1 = data[x].quantile(0.25)
    Q2 = data[x].quantile(0.50)
    Q3 = data[x].quantile(0.75)
    print(f'5 Point Summary of {x.capitalize()} Attribute:\n'
          f'{x.capitalize()}(min) : {x_min}\n'
          f'Q1                    : {Q1}\n'
          f'Q2(Median)            : {Q2}\n'
          f'Q3                    : {Q3}\n'
          f'{x.capitalize()}(max) : {x_max}')

    fig = plt.figure(figsize=(16, 10))
    plt.subplots_adjust(hspace = 0.6)
    sns.set_palette('pastel')

    plt.subplot(221)
    ax1 = sns.distplot(data[x], color = 'r')
    plt.title(f'{x.capitalize()} Density Distribution')

    plt.subplot(222)
    ax2 = sns.violinplot(x = data[x], palette = 'Accent', split = False)
    plt.title(f'{x.capitalize()} Violinplot')

    plt.subplot(223)
    ax2 = sns.boxplot(x=data[x], palette = 'cool', width=0.7, linewidth=0.6)
    plt.title(f'{x.capitalize()} Boxplot')

    plt.subplot(224)
    ax3 = sns.kdeplot(data[x], cumulative=True)
    plt.title(f'{x.capitalize()} Cumulative Density Distribution')

    plt.show()
```

Figure 5.2: Show the alignment of the charts

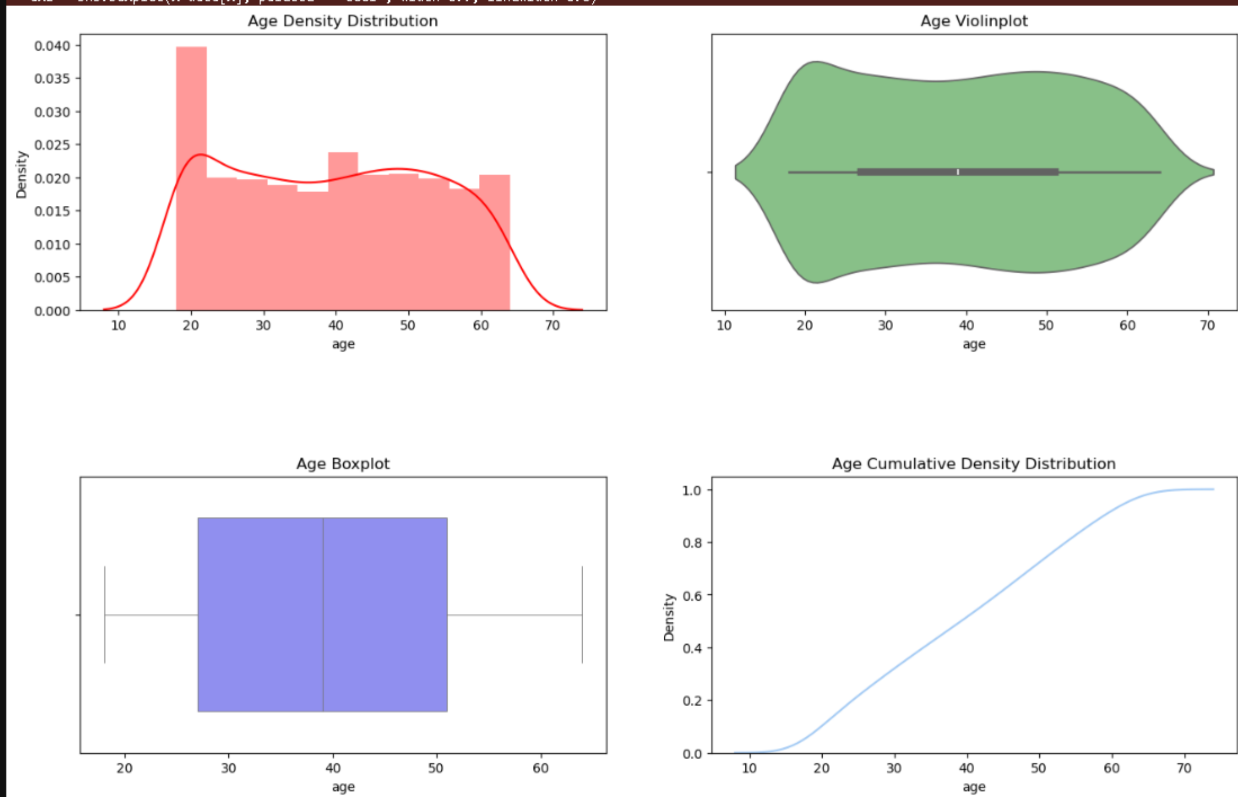
```python
def box_plot(x = 'bmi'):
    def add_values(bp, ax):
        """ This actually adds the numbers to the various points of the boxplots"""
        for element in ['whiskers', 'medians', 'caps']:
            for line in bp[element]:
                # Get the position of the element. y is the label you want
                (x_l, y),(x_r, _) = line.get_xydata()
                # Make sure datapoints exist
                # (I've been working with intervals, should not be problem for this case)
                if not np.isnan(y):
                    x_line_center = x_l + (x_r - x_l)/2
                    y_line_center = y  # Since it's a line and it's horisontal
                    # overlay the value:  on the line, from center to right
                    ax.text(x_line_center, y_line_center, # Position
                            '%.2f' % y, # Value (3f = 3 decimal float)
                            verticalalignment='center', # Centered vertically with line
                            fontsize=12, backgroundcolor="white")

    fig, axes = plt.subplots(1, figsize=(4, 8))

    red_diamond = dict(markerfacecolor='r', marker='D')

    bp_dict = data.boxplot(column = x,
                           grid=True,
                           figsize=(4, 8),
                           ax=axes,
                           vert = True,
                           notch=False,
                           widths = 0.7,
                           showmeans = True,
                           whis = 1.5,
                           flierprops = red_diamond,
                           boxprops= dict(linewidth=3.0, color='black'),
                           whiskerprops=dict(linewidth=3.0, color='black'),
                           return_type = 'dict')
```

```python
    add_values(bp_dict, axes)

    plt.title(f'{x.capitalize()} Boxplot', fontsize=16)
    plt.ylabel(f'{x.capitalize()}', fontsize=14)
    plt.show()

    skew = data[x].skew()
    Q1 = data[x].quantile(0.25)
    Q3 = data[x].quantile(0.75)
    IQR = Q3 - Q1
    total_outlier_num = ((data[x] < (Q1 - 1.5 * IQR)) | (data[x] > (Q3 + 1.5 * IQR))).sum()
    print(f'Mean {x.capitalize()} = {data[x].mean()}')
    print(f'Median {x.capitalize()} = {data[x].median()}')
    print(f'Skewness of {x}: {skew}.')
    print(f'Total number of outliers in {x} distribution: {total_outlier_num}.')
```

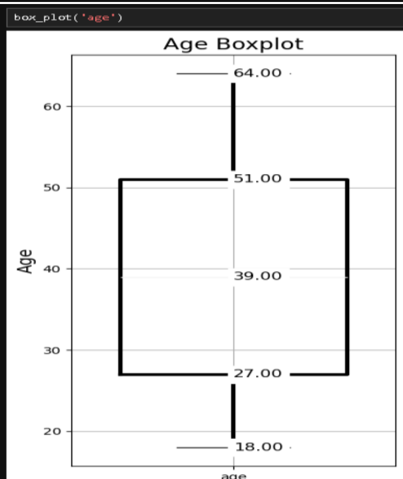
Table: 5.3 : Distribution of the Age

```
summary('age')

5 Point Summary of Age Attribute:
Age(min) : 18
Q1                    : 27.0
Q2(Median)            : 39.0
Q3                    : 51.0
Age(max) : 64
```

```
ax2 = sns.boxplot(x=data[x], palette = 'cool', width=0.7, linewidth=0.6)
```




```
[64]: box_plot('age')
```




```
Mean Age = 39.20702541106129
Median Age = 39.0
Skewness of age: 0.05567251565299186.
Total number of outliers in age distribution: 0.
```

Interpretation of the results

The box and whisker are the same on both sides. meaning the distribution is symmetric. The result also showed that the min age of the insurance cover was 18 year and the maximum age was 64.the quartile range are 25th percentiles(Q1) is 27,at percentile Q2 was 39.0 and the 75th percentile Q3 was (51).and there are no outliers in the data. Therefore, the age of the insured approximately follows a uniform distribution with Mean of 39.2 and Median of 39.0, and with lowest age being 18 and highest being 64. There are no outlier values in the Age distribution in the data.

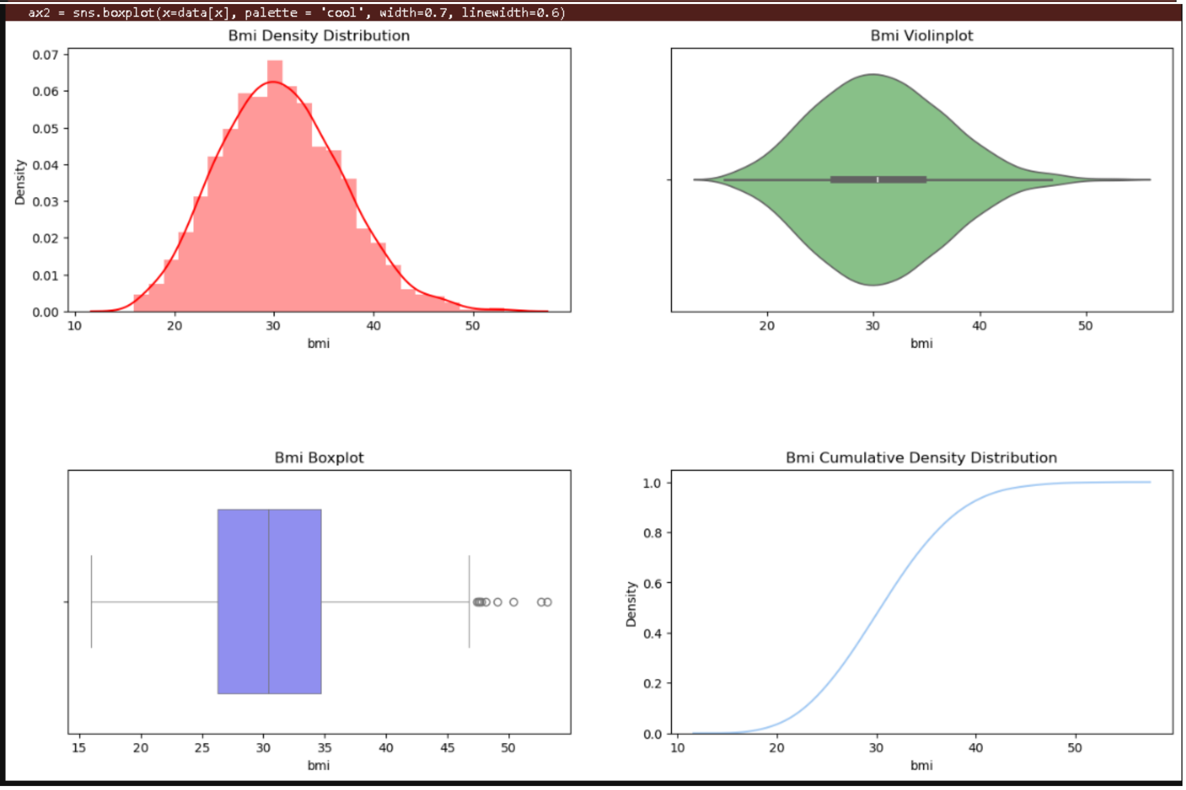
Table: 5.4 : Shows the maximum age of the insured.

```
[251]: # How many of the insured have the age of 64?
       data = data[data['age'] == data['age'].max()]
       print(df.head())
       print()
       print(f'Total number of insured people with the age of 64: {len(df)}.')

            age  sex    bmi  children  smoker  region      charges
       62    64    1  24.70         1       0       1  30166.61817
       94    64    0  31.30         2       1       3  47291.05500
       199   64    0  39.33         0       0       0  14901.51670
       328   64    0  33.80         1       1       3  47928.03000
       335   64    1  34.50         0       0       3  13822.80300

       Total number of insured people with the age of 64: 20.
```

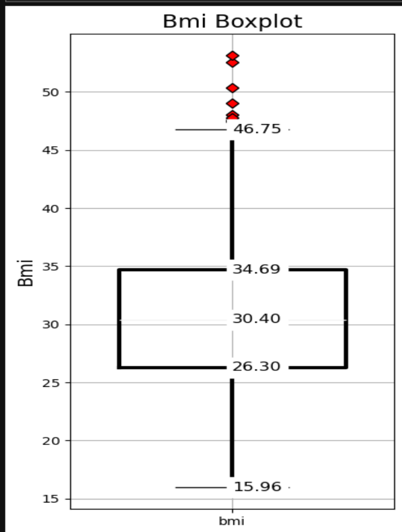
Table: 5.5 : Distribution of the ‘bmi’

```
summary('bmi')

5 Point Summary of Bmi Attribute:
Bmi(min) : 15.96
Q1                  : 26.29625
Q2(Median)          : 30.4
Q3                  : 34.69375
Bmi(max)  : 53.13
```

```
ax2 = sns.boxplot(x=data[x], palette = 'cool', width=0.7, linewidth=0.6)
```



```
[70]: box_plot('bmi')
```




```
Mean Bmi = 30.66339686098655
Median Bmi = 30.4
Skewness of bmi: 0.2840471105987448.
Total number of outliers in bmi distribution: 9.
```

```
[72]: # Who amonga the insured has the highest BMI what would be the charge?
data[data['bmi'] == data['bmi'].max()]
```

[72]:

	age	sex	bmi	children	smoker	region	charges
1317	18	male	53.13	0	no	southeast	1163.4627

Observations:
The BMI distribution of the Insured approximately follows a normal distribution with a Mean of 30.66 and Median of 30.4.
There are a total of 9 outlier values in the BMI distribution, all in the higher side. The highest BMI observed is 53.13.
The person with the highest BMI (least healthy, based on available data) is also one of the youngest (male, 18, non-smoker.) He is paying less premium than the mean, but significantly more than the median charges. This is in line with our basic understanding of underwriting rules.

6.0 Check if the number of premium charges for smokers or non-smokers is increasing as they are aging

Table 6.1.Distribution of the Charges

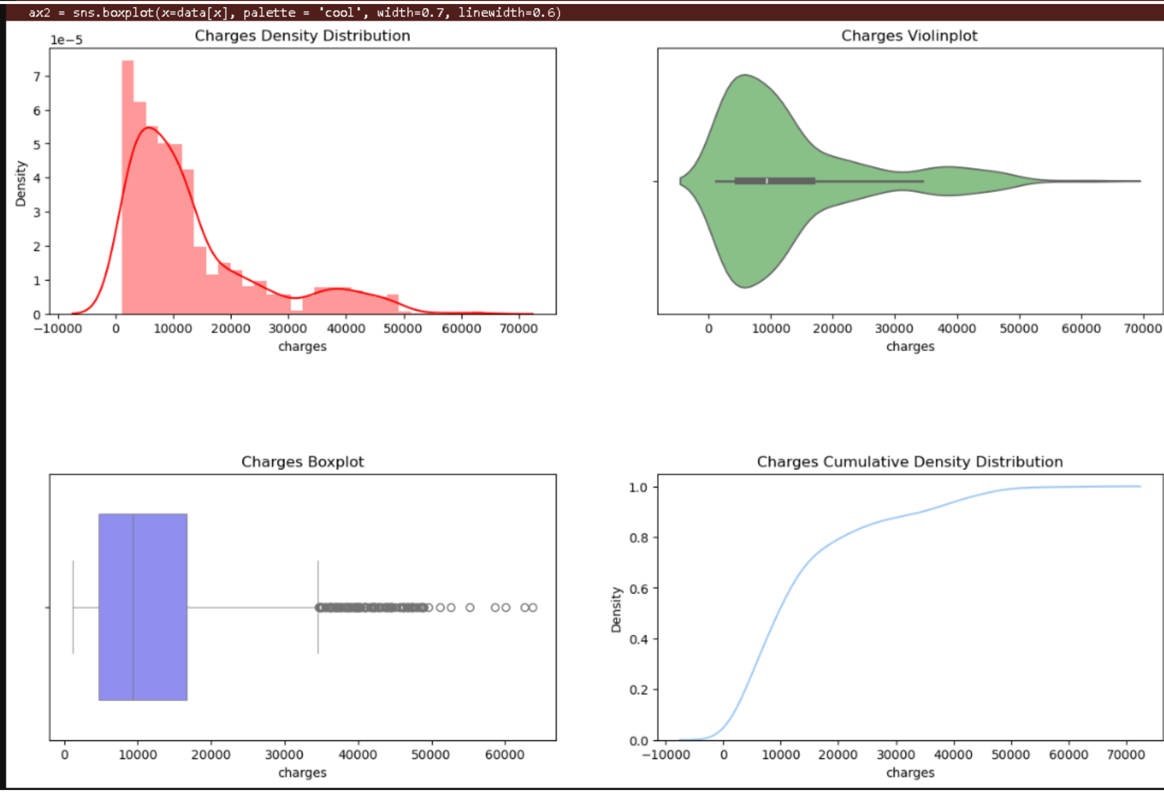
```
[74]: data['charges'].mean(), data['charges'].median()

[74]: (13270.422265141257, 9382.033)
```

- The mean charge was 13270.42 and the medium charge was 9383

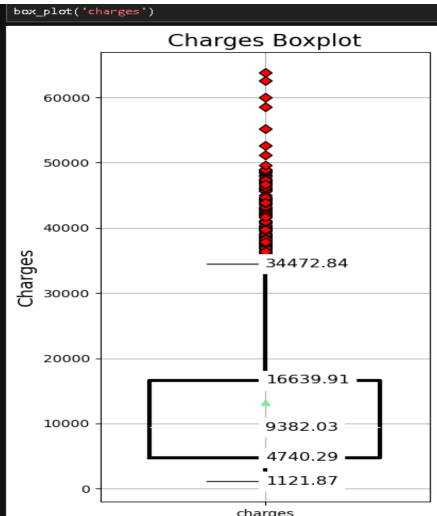
Table:6.2: Distribution of Charges

```
[78]: summary('charges')

5 Point Summary of Charges Attribute:
Charges(min) : 1121.8739
Q1               : 4740.28715
Q2(Median)       : 9382.033
Q3               : 16639.912515
Charges(max) : 63770.42801
```

```
ax2 = sns.boxplot(x=data[x], palette = 'cool', width=0.7, linewidth=0.6)
```




```
[80]: box_plot('charges')
```




```
Mean Charges = 13270.422265141257
Median Charges = 9382.033
Skewness of charges: 1.5158796580240388.
Total number of outliers in charges distribution: 139.
```

Table:6.2: Person with the Highest premium Charges

```
•[82]: # Who amonga people is paying the highest charges?
       data[data['charges'] == data['charges'].max()]
```

[82]:

	age	sex	bmi	children	smoker	region	charges
543	54	female	47.41	0	yes	southeast	63770.42801

Table:6.2.1 Person with the minimum premium Charges

```
[126]: data[data['charges']==data['charges'].min()]
```

[126]:

	age	sex	bmi	children	smoker	region	charges
940	18	male	23.21	0	no	southeast	1121.8739

Table:6.3: Person with the Bmi paying Less premium Charges

```
[84]: # Who is the insured with the highest BMI, and how does his charges compare to the rest?
      data[data['bmi'] == data['bmi'].max()]
```

[84]:

	age	sex	bmi	children	smoker	region	charges
1317	18	male	53.13	0	no	southeast	1163.4627

Table:6.4:Show a Person in the nonsmokers with the Highest bmi

```
[253]: # Who in the insured has the highest BMI, and how does his charges compare to the rest?
       data[data['bmi'] == data['bmi'].max()]
```

[253]:

	age	sex	bmi	children	smoker	region	charges
534	64	1	40.48	0	0	2	13831.1152

```
[86]: data['charges'].mean(), data['charges'].median()
[86]: (13270.422265141257, 9382.033)
```

Result interpretation

- From the total of 1338 data points, there are 139 outlier values in the distribution of charges, all in the higher side. The highest charges paid is 63770.42801.(table.6.4 and 6.2)
- The insured smoker charged with highest premium amount is a 54 years old female smoker with relatively high BMI of 47.41cm indicating obesity.(table.6.2)
- The person with the highest BMI is a male with 18 years and is a non-smoker. He is paying less premium charges than the mean, but significantly more than the median.
- The insured non - smoker with the highest bmi is a 64 years old non-smoker with relatively high BMI of 40.48 cm (indicating obesity).table.6.4
- The distribution of Charges for the Insured is heavily left skewed (median < mean) with a Mean of 13270.4223 and Median of 9382.033. The lowest charged amount is 1121.8739 and the highest charged amount is 63770.42801.table.6.2

Table:7.0. Distribution and summary of categorical

```python
# distribution of categorical variables: sex, smoker,region and children
```

```python
# Create a function that returns a Pie chart for categorical variable:
def pie_chart(x = 'smoker'):
    """
    Function creates a Pie chart for categorical variables.
    """
    fig, ax = plt.subplots(figsize=(8, 6), subplot_kw=dict(aspect="equal"))

    s = data.groupby(x).size()

    mydata_values = s.values.tolist()
    mydata_index = s.index.tolist()

    def func(pct, allvals):
        absolute = int(pct/100.*np.sum(allvals))
        return "{:.1f}%\n({:d})".format(pct, absolute)


    wedges, texts, autotexts = ax.pie(mydata_values, autopct=lambda pct: func(pct, mydata_values),
                                      textprops=dict(color="w"))

    ax.legend(wedges, mydata_index,
              title="Index",
              loc="center left",
              bbox_to_anchor=(1, 0, 0.5, 1))

    plt.setp(autotexts, size=12, weight="bold")

    ax.set_title(f'{x.capitalize()} Piechart')

    plt.show()
```

Table:7.1 : Shows the smokers by Sex

```
[262]: # Average premium charges for smokers significantly higher than non-smokers?
       data['charges'].groupby(data['smoker']).mean()

[262]: smoker
       0    15733.801755
       1    39283.060036
       Name: charges, dtype: float64
```

Table:7.2:Shows the smokers by Sex

```
[104]: data.groupby(['smoker', 'sex']).agg('count')
```

[104]:

smoker	sex	age	bmi	children	region	charges
no	female	547	547	547	547	547
	male	517	517	517	517	517
yes	female	115	115	115	115	115
	male	159	159	159	159	159

Table:7.3: Shows group of smokers, charges and Sex

```
[166]: data.groupby (['smoker','charges','age']).agg('max')
```

[166]:

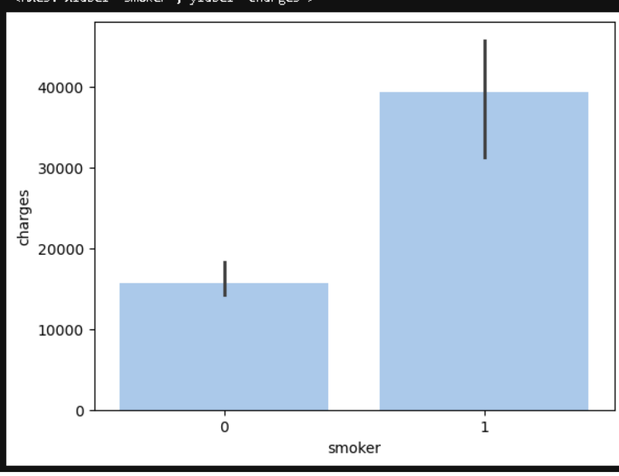
smoker	charges	age	sex	bmi	children	region
no	1121.87390	18	male	23.210	0	southeast
	1131.50660	18	male	30.140	0	southeast
	1135.94070	18	male	33.330	0	southeast
	1136.39940	18	male	33.660	0	southeast
	1137.01100	18	male	34.100	0	southeast
...	...	...	...	...	...	...
yes	55135.40209	33	female	35.530	0	northwest
	58571.07448	31	female	38.095	1	northeast
	60021.39897	52	male	34.485	3	northwest
	62592.87309	45	male	30.360	0	southeast
	63770.42801	54	female	47.410	0	southeast

1337 rows × 4 columns

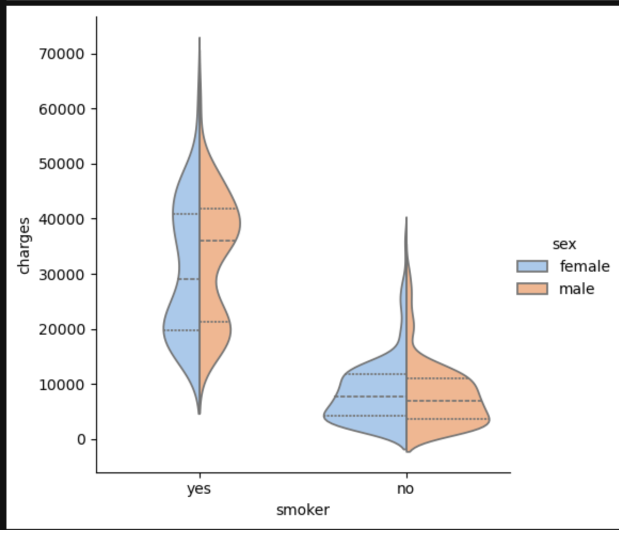
Table:7.4: Shows group of smokers by charges

```
[264]: # premium average charges for smokers are indeed significantly higher than non-smokers.
       sns.barplot(x = "smoker", y = "charges", data = data)
```

[264]: <Axes: xlabel='smoker', ylabel='charges'>



```
[108]: sns.catplot(x="smoker", y="charges", hue="sex",
                   kind="violin", inner="quartiles", split=True,
                   palette="pastel", data=data);
```



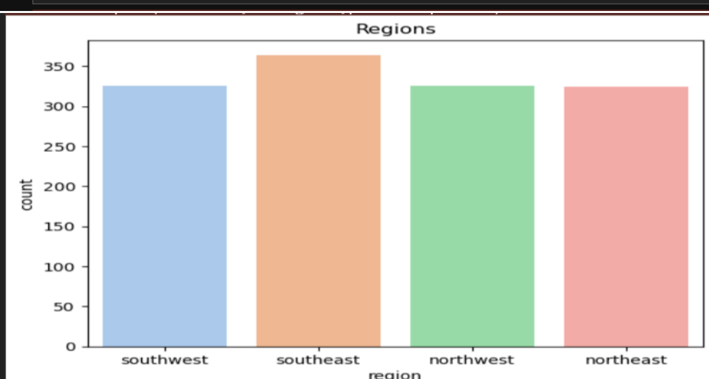

```
[110]: data.groupby(['smoker', 'sex']).agg('count')['age']

[110]: smoker  sex
       no      female    547
               male      517
       yes     female    115
               male      159
       Name: age, dtype: int64
```

Results Interpretation

- From the total 1338 insured people about 274 (20.5%) are smokers and the rest are non-smokers.
- Among 274 smokers, proportion of males (159) are higher than females (115).
- The average insurance premium for smokers is significantly higher than non-smokers.

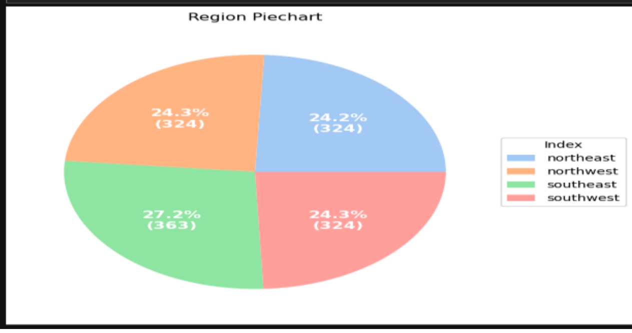
8.0 Show the bar chart of Region

```
[19]: sns.countplot(data=data,x='region',palette='pastel')
      plt.title('Regions')
      plt.show()
```



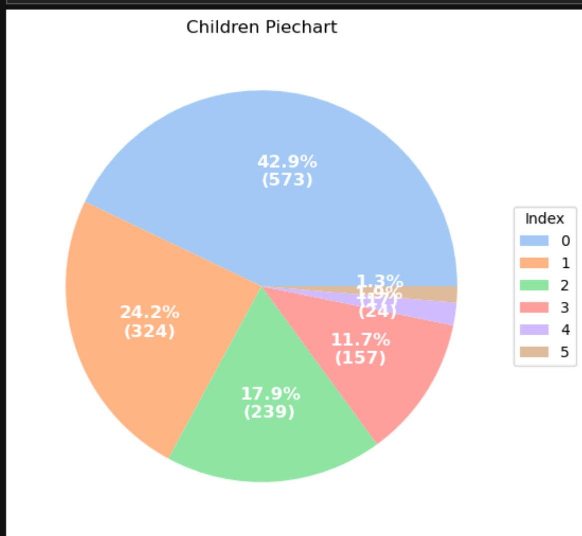
8.1 Show the a pie chart of Region

```
[117]: pie_chart('region')
```



9.1 Distribution of the numbers of children

```
[119]: pie_chart('children')
```




```
[121]: data.groupby(['children']).agg('count')['age']

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

Observation: In the dataset, approximately 85% (1138 / 1338) of the insured have less than 3 children

```
[124]: data.head()
```

[124]:

	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

10.1 Creating relationship in the dataset

[129]:
```
for x in ['sex', 'children', 'smoker', 'region']:
    data[x] = data[x].astype('category')

data.dtypes
```

[129]:
```
age            int64
sex         category
bmi          float64
children    category
smoker      category
region      category
charges      float64
dtype: object
```

```
cat_columns = data.select_dtypes(['category']).columns
cat_columns
```

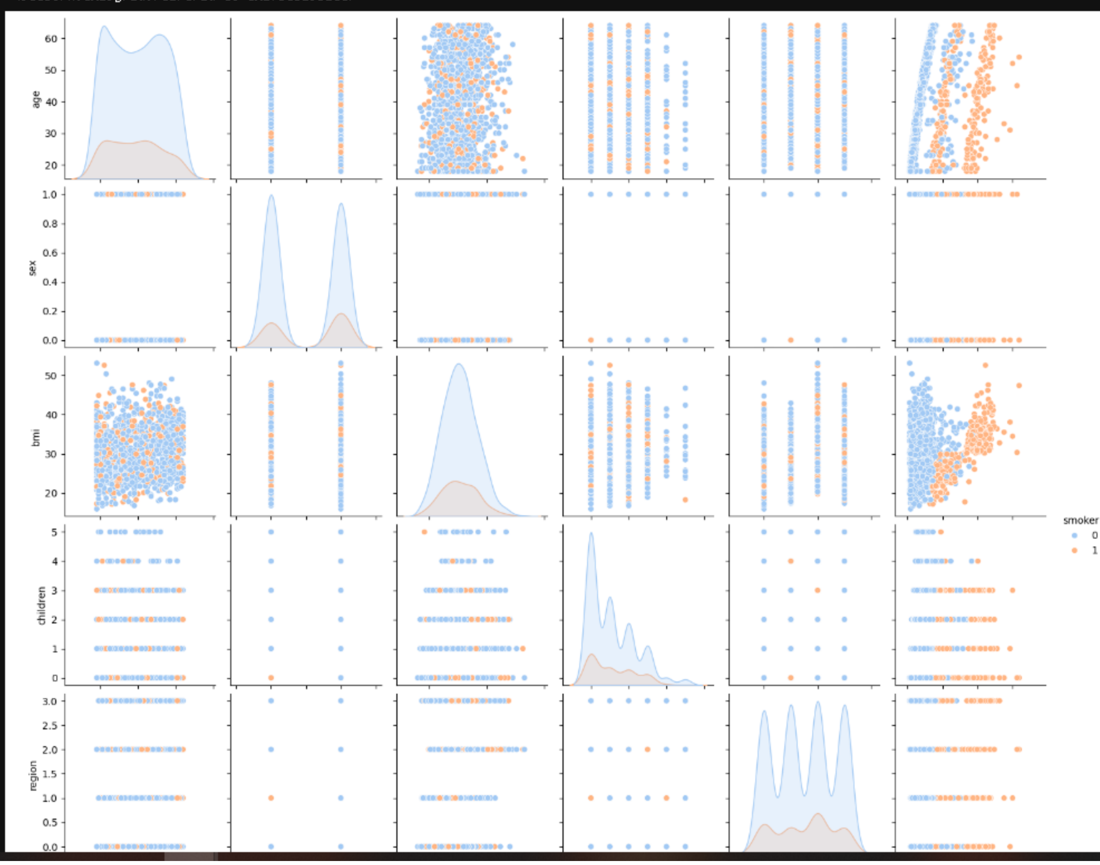
[135]:
```
# Finally, we transform the original columns by replacing the elements with their category codes:
data[cat_columns] = data[cat_columns].apply(lambda x: x.cat.codes)
data.head()
```

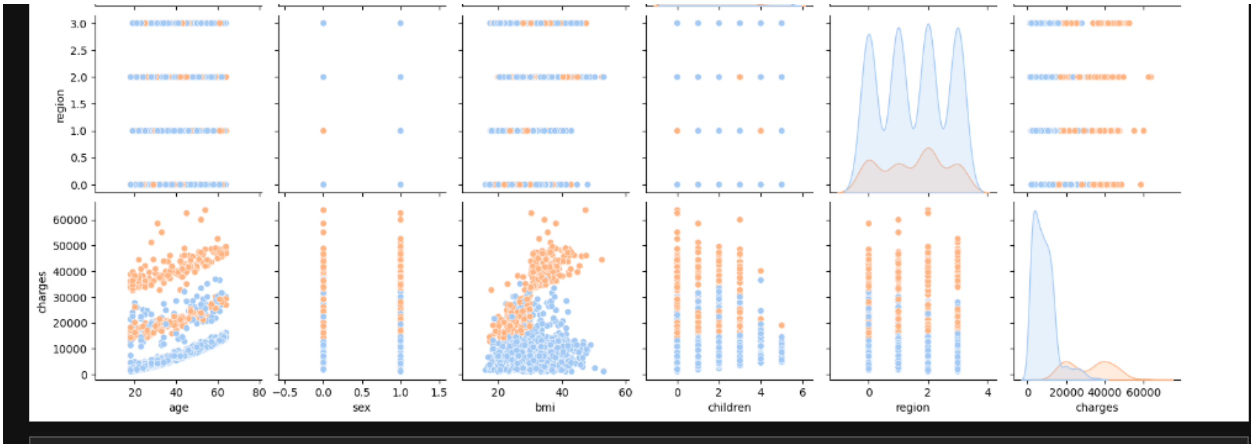
[135]:

	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

[137]:
```
# Now we can plot all columns of our dataset in a pairplot!
sns.pairplot(data, hue  = 'smoker')
```

[137]: <seaborn.axisgrid.PairGrid at 0x27bc32931a0>





Interpretation of results

- The results showed that with increased age, smoking increases .therefore, the higher the age the of smoker's the more premium charges paid.
- The higher the bmi in smokers the higher the charges.
- Smokers pay higher insurance premium than the non-smokers.

[143]:
```
data.plot(kind="scatter", x="age", y="charges",
    s=data["smoker"]*25, label="smoker", figsize=(14,10),
    c='bmi', cmap=plt.get_cmap("jet"), colorbar=True,
    sharex=False)
plt.legend()
```

[143]: `<matplotlib.legend.Legend at 0x27bc30ccdd0>`

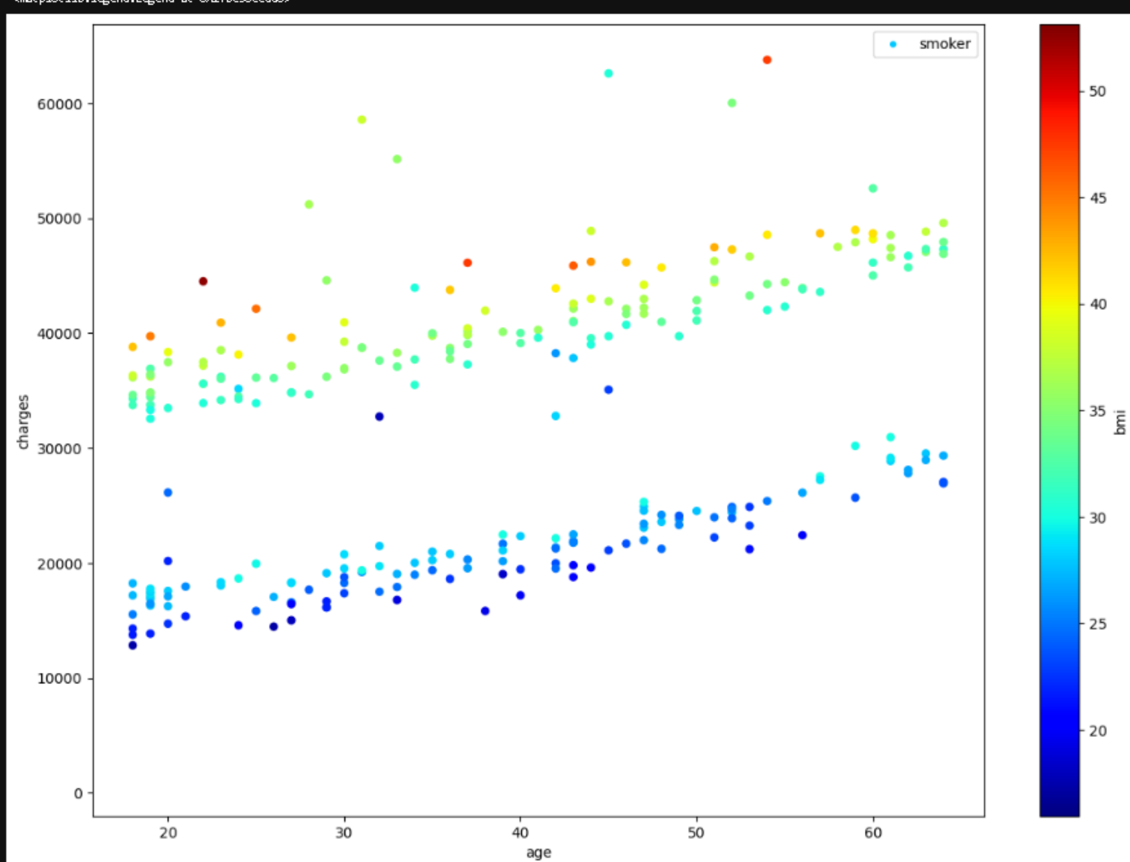


Table 11: Determining the relationship by correlation

Table 11.1 :Show a Heatmap

```
[148]: corr = data.corr()
       plt.figure(figsize=(10, 8))
       sns.heatmap(corr, annot=True, cmap = 'summer_r')

[148]: <Axes: >
```

- From the correlation heatmap, we can conclude that the premium charges show a strong positive correlation with smoking and a weak positive correlation with the Age and BMI of the insured people

Table 11.2 shows T test command and output

Null hypothesis -there is no relation between mean charges of smokers and nonsmokers

Alternative hypothesis- there is a relation between mean charges of smokers and nonsmokers

```
[266]: # Collect data and calculate the value of test statistic

[153]: smokers = data[data['smoker'] == 0]
       nonsmokers = data[data['smoker'] == 1]
       charge_smokers = smokers['charges']
       charge_nonsmokers = nonsmokers['charges']

       print(f'Number of smokers: {smokers.shape[0]}')
       print(f'Variance in charges of smokers: {np.var(charge_smokers)}')
       print(f'Number of non - smokers: {nonsmokers.shape[0]}')
       print(f'Variance in charges of non - smokers: {np.var(charge_nonsmokers)}')

       Number of smokers: 1064
       Variance in charges of smokers: 35891656.00316425
       Number of non - smokers: 274
       Variance in charges of non - smokers: 132721153.13625384
```

```
[155]: from scipy.stats import ttest_ind

       t_statistic, p_value = ttest_ind(charge_smokers, charge_nonsmokers, equal_var=False)
       print(f't_statistic: {t_statistic}\np_value: {p_value}')

       t_statistic: -32.751887766341824
       p_value: 5.88946444671698e-103
```

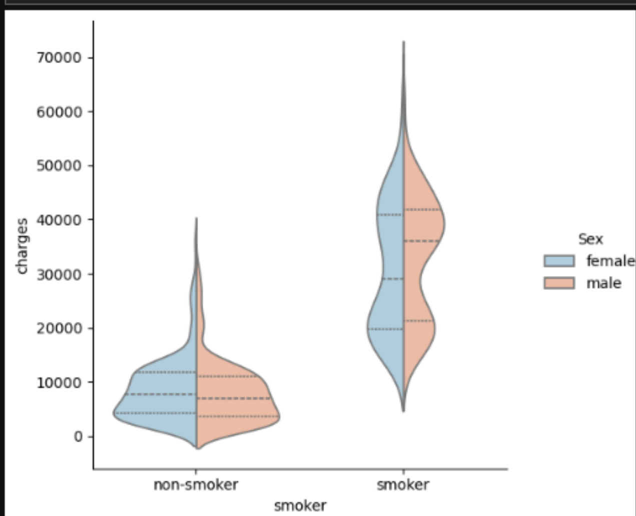
```
[270]: # Determine the probability associated with the test statistic under the null hypothesis using sampling distribution of the test statistic

[159]: print ("two-sample t-test p-value=", p_value)

two-sample t-test p-value= 5.889464444671698e-103
```

- We Reject the Null Hypothesis that state at At 5% significance level, the mean charges of smokers and non - smokers are not equal. Therefore, the charges of people who smoke differs significantly from the people who do not smoke.

Table 11.3 : data Collection and calculating of the value of t test statistic

```
[173]: smokers = data[data['smoker'] == 0]
nonsmokers = data[data['smoker'] == 1]
change_smokers = smokers['charges']
change_nonsmokers = nonsmokers['charges']

print(f'Number of smokers: {smokers.shape[0]}')
print(f'Variance in charges of smokers: {np.var(change_smokers)}')
print(f'Number of non - smokers: {nonsmokers.shape[0]}')
print(f'Variance in charges of non - smokers: {np.var(change_nonsmokers)}')

Number of smokers: 1064
Variance in charges of smokers: 35891656.00316425
Number of non - smokers: 274
Variance in charges of non - smokers: 132721153.13625384
```

```
[175]: #Visulizing the collected data:
g = sns.catplot(x="smoker", y="charges", hue="sex",
          kind="violin", inner="quartiles", split=True,
          palette="RdBu_r", data=data, legend_out = True);

xlabels = ['non-smoker', 'smoker']
g.set_xticklabels(xlabels)

new_title = 'Sex'
g._legend.set_title(new_title)
g._legend.set_bbox_to_anchor([1.1, 0.5])
# replace labels
new_labels = ['female', 'male']
for t, l in zip(g._legend.texts, new_labels): t.set_text(l)
```




```
[177]: from scipy.stats import ttest_ind

t_statistic, p_value = ttest_ind(change_smokers, change_nonsmokers, equal_var=False)
print(f't_statistic: {t_statistic}\np_value: {p_value}')

t_statistic: -32.751887766341824
p_value: 5.889464444671698e-103
```

```
[179]: Determine the probability associated with the test statistic under the null hypothesis using sampling distribution of the test statistic

[272]: print ("two-sample t-test p-value=", p_value)

two-sample t-test p-value= 0.08992430667834876
```

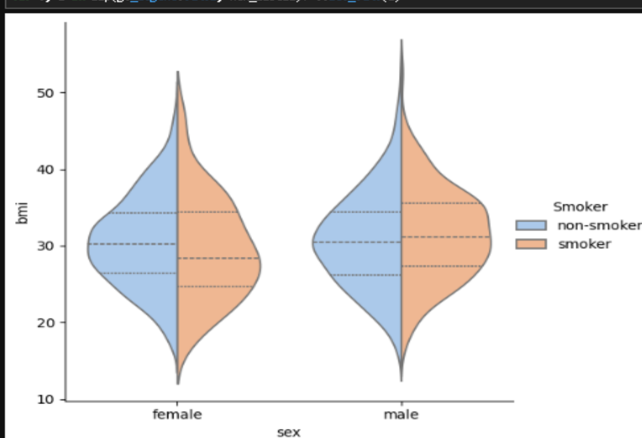
```
[187]: Compare the probability associated with the test statistic with level of significance specified
At 5% significance level, α = 0.05

[189]: p_value > 0.05

[189]: False
```

- We Reject the Null Hypothesis and that At 5% significance levels, the mean charges of smokers and non - smokers are not equal. That is say that the charges of people who smoke indeed differ significantly from the people who don't.

Table 11.3 : Does BMI of males differ significantly from that of females

```
[274]: # Collecting data and calculate the value of test statistic

[282]: males = data[data['sex'] == 1]
females = data[data['sex'] == 0]
bmi_males = males['bmi']
bmi_females = females['bmi']

print(f'Number of males: {males.shape[0]}')
print(f'Variance in BMI of males: {np.var(bmi_males)}')
print(f'Number of females: {females.shape[0]}')
print(f'Variance in BMI of females: {np.var(bmi_females)}')

Number of males: 676
Variance in BMI of males: 37.6491607363954
Number of females: 662
Variance in BMI of females: 36.49917703379856
```

```
[284]: #visualizing the collected data:
g = sns.catplot(x="sex", y="bmi", hue="smoker",
            kind="violin", inner="quartiles", split=True,
            palette="pastel", data=data, legend_out = True);

xlabels = ['female', 'male']
g.set_xticklabels(xlabels)

new_title = 'Smoker'
g._legend.set_title(new_title)
g._legend.set_bbox_to_anchor([1.1, 0.5])
# replace labels
new_labels = ['non-smoker', 'smoker']
for t, l in zip(g._legend.texts, new_labels): t.set_text(l)
```




```
[286]: from scipy.stats import ttest_ind

t_statistic, p_value = ttest_ind(bmi_males, bmi_females, equal_var=False)
print(f't_statistic: {t_statistic}\np_value: {p_value}')

t_statistic: 1.697027933124022
p_value: 0.08992430667834876
```

```
[208]: #4 Determine the probability associated with the test statistic under the null hypothesis using sampling distribution of the test statistic
[210]: print ("two-sample t-test p-value=", p_value)
two-sample t-test p-value= 0.08992430667834876
```

```
[214]: # Compare the probability associated with the test statistic with level of significance specified
# At 5% significance level, α = 0.05
[212]: p_value > 0.05
[212]: True
```

Results interpretation

We Fail to Reject the Null Hypothesis that at 5% significance level, the mean BMI of insured males and females are equal. Therefor the BMI of males do not differ significantly from that of females in our data

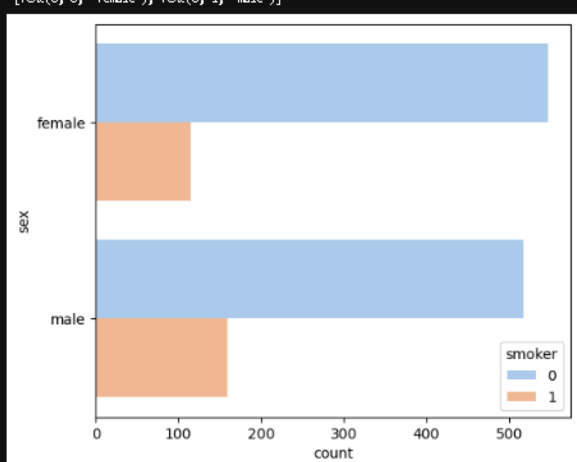
Table 12: Relation of smokers and nonsmokers by sex

Null hypothesis that BMI for 3 groups of women having no, one or two children respectively

Alternative Hypothesis that the Bmi for 3 groups of women does not have one or two children repectively

Table 12.1 The proportions of gender by smoker and non-smokers

```
[220]: # Visualization of the collected data:
plt.figure(figsize=(6,5))
chart = sns.countplot(y = 'sex', hue = 'smoker', data = data)
chart.set_yticklabels(['female', 'male'])
[220]: [Text(0, 0, 'female'), Text(0, 1, 'male')]
```



They are different proportions between smokers and non-smokers, however the results show that they are statistically significant.

```
[238]: data = data[data['children'] <= 2]
       female = df[df['sex'] == 0]
       female.head()
```

[238]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.90	0	1	3	16884.92400
5	31	0	25.74	0	0	2	3756.62160
6	46	0	33.44	1	0	2	8240.58960
9	60	0	25.84	0	0	1	28923.13692
11	62	0	26.29	0	1	2	27808.72510

```
[240]: # Visualizing the collected data:
       fig = plt.figure(figsize=(12, 8))
       box_plot = sns.boxplot(x = "children", y = "bmi", data = female, width = 0.5)

       medians = female.groupby(['children'])['bmi'].median().round(2)
       vertical_offset = female['bmi'].median() * 0.05 # offset from median for display

       medians
       for xtick in box_plot.get_xticks():
           box_plot.text(xtick, medians[xtick] + vertical_offset,medians[xtick],
                   horizontalalignment='center',color='w',weight='semibold')


       plt.title('BMI by No. of children')
       plt.show()
```

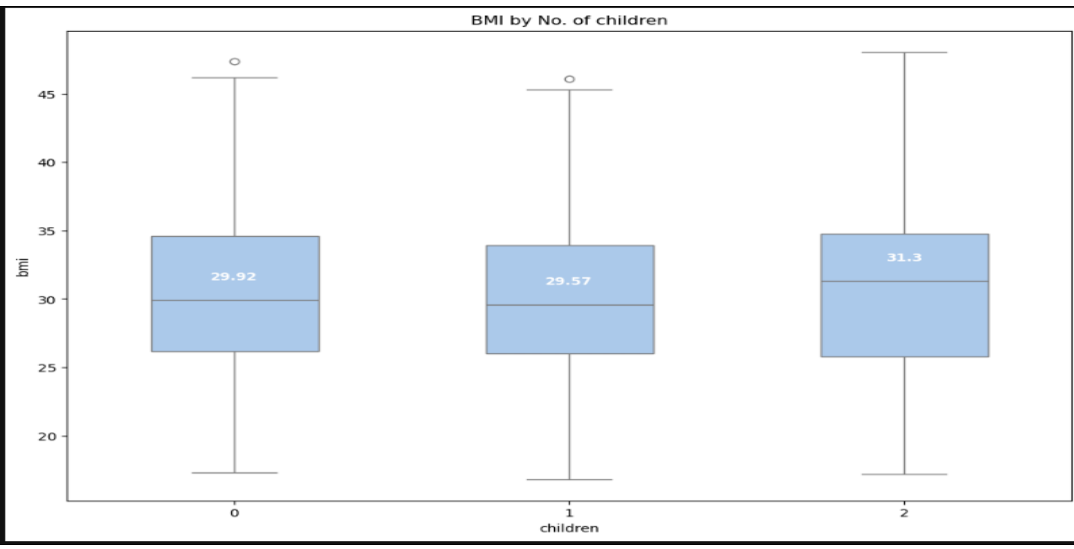


Table 13.0: The proportions of gender by smoker and non-smokers

Table 13.1. setting a statistical model.

```
[243]: import statsmodels.api as sm
       from statsmodels.formula.api import ols

       mod = ols('bmi ~ children', data = female).fit()
       aov_table = sm.stats.anova_lm(mod, typ=2)
       print(aov_table)

                    sum_sq     df         F    PR(>F)
       children     2.512982    1.0  0.068411  0.79376
       Residual  20717.738725  564.0       NaN       NaN
```

Table 13. 2 Analysis of variance using Anova

```
[245]: from statsmodels.stats.multicomp import pairwise_tukeyhsd
       print(pairwise_tukeyhsd(female['bmi'], female['children']))

       Multiple Comparison of Means - Tukey HSD, FWER=0.05
       ==================================================
       group1 group2 meandiff p-adj   lower  upper  reject
       --------------------------------------------------
            0      1  -0.3089 0.8641 -1.7185 1.1008  False
            0      2   0.2883 0.9003 -1.2636 1.8401  False
            1      2   0.5971 0.6961 -1.1322 2.3265  False
       --------------------------------------------------
```

Interpretation of results

We **Fail to Reject** the null hypothesis states that for BMI for 3 groups of women having no, one or two children respectively, mean BMI of all groups are equal. Therefore, the distribution of BMI across women with no children, one child and two children are the same.