

```
#Importing required data in a DataFrame
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('/content/drive/MyDrive/Dataset_ML/insurance.csv')
df
```

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

1338 rows × 8 columns

```
#print the first 5 observations
df.head()
```

	index	age	sex	bmi	children	smoker	region	charges
0	0	19	female	27.900	0	yes	southwest	16884.92400
1	1	18	male	33.770	1	no	southeast	1725.55230
2	2	28	male	33.000	3	no	southeast	4449.46200
3	3	33	male	22.705	0	no	northwest	21984.47061
4	4	32	male	28.880	0	no	northwest	3866.85520

```
#print the last 5 observations
df.tail()
```

	index	age	sex	bmi	children	smoker	region	charges
1333	1333	50	male	30.97	3	no	northwest	10600.5483
1334	1334	18	female	31.92	0	no	northeast	2205.9808



```
#print no of rows and columns
df.shape
```

```
(1338, 8)
```

```
#print columns heading
df.columns
```

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

```
#Each columns datatype
df.dtypes
```

```
index      int64
age         int64
sex         object
bmi         float64
children    int64
smoker      object
region      object
charges     float64
dtype: object
```

```
#checking duplicate data
df.duplicated().sum()
```

```
0
```

```
#Checking missing value
df.isna().sum()
```

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

```
#displaying column corresponding no-null count and datatype
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
```

```
Data columns (total 8 columns):
#   Column      Non-Null Count  Dtype
---  -
0   index        1338 non-null    int64
1   age           1338 non-null    int64
2   sex           1338 non-null    object
3   bmi           1338 non-null    float64
4   children      1338 non-null    int64
5   smoker        1338 non-null    object
6   region        1338 non-null    object
7   charges       1338 non-null    float64
dtypes: float64(2), int64(3), object(3)
memory usage: 83.8+ KB
```

```
df.describe()
```

	index	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000
mean	668.500000	39.207025	30.663397	1.094918	13270.422265
std	386.391641	14.049960	6.098187	1.205493	12110.011237
min	0.000000	18.000000	15.960000	0.000000	1121.873900
25%	334.250000	27.000000	26.296250	0.000000	4740.287150
50%	668.500000	39.000000	30.400000	1.000000	9382.033000
75%	1002.750000	51.000000	34.693750	2.000000	16639.912515
max	1337.000000	64.000000	53.130000	5.000000	63770.428010



```
#display count of unique values in a column
df.nunique()
```

```
index      1338
age         47
sex         2
bmi        548
children    6
smoker      2
region      4
charges    1337
dtype: int64
```

```
#drop unwanted columns
df=df.drop('index',axis=1)
df
```

	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

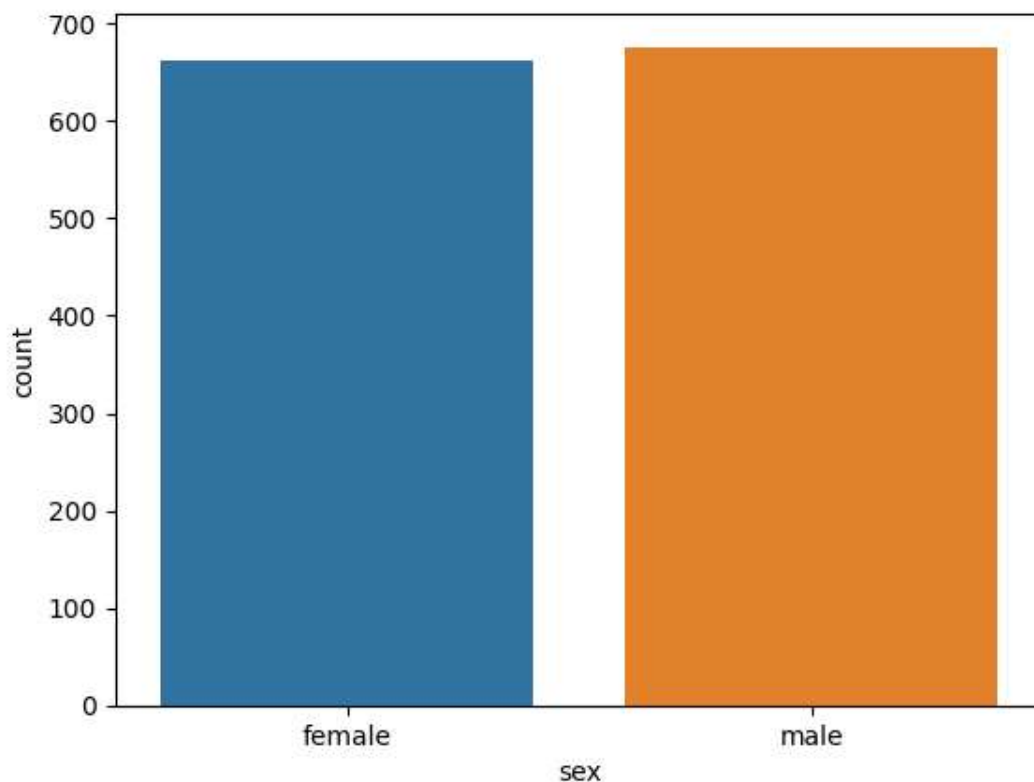


```
#display column values counts
sex=df['sex'].value_counts()
sex
```

```
male      676
female    662
Name: sex, dtype: int64
```

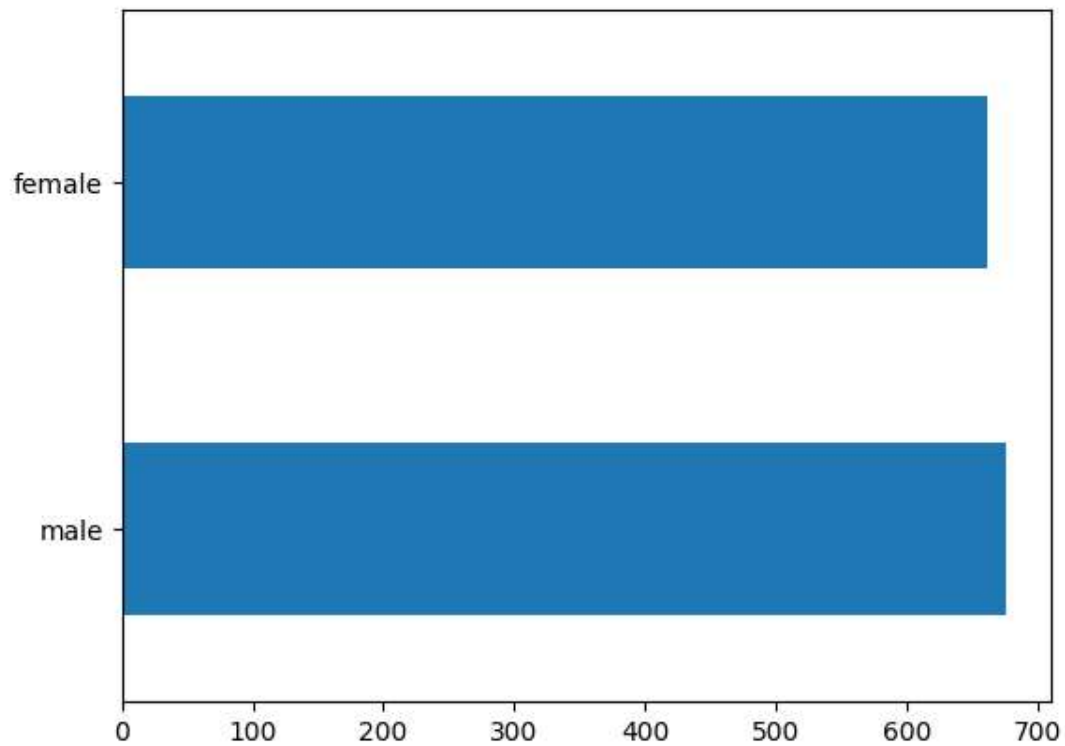
```
# SEABORN VISUALIZATION
sns.countplot(x='sex',data=df)
```

```
<Axes: xlabel='sex', ylabel='count'>
```



```
sex.plot(kind='barh')
```

<Axes: >



```
children=df['children'].value_counts()  
children
```

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

```
sns.countplot(x='children',data=df)
```

<Axes: xlabel='children', ylabel='count'>



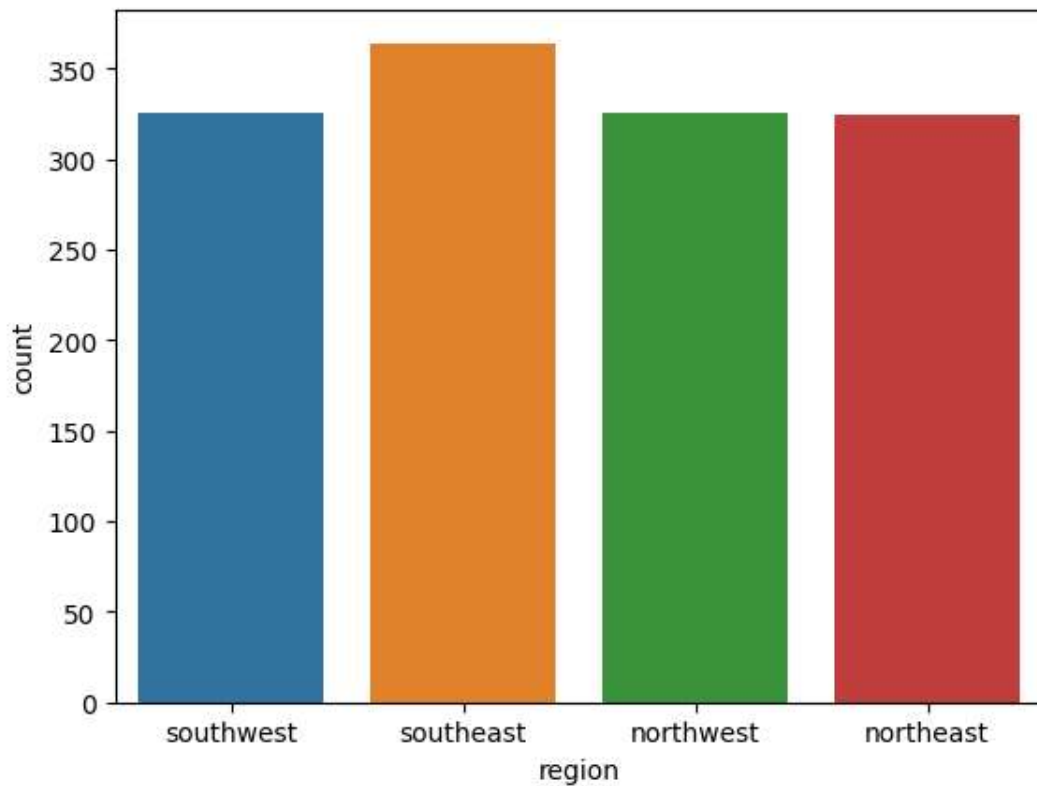
```
region=df['region'].value_counts()  
region
```

```
southeast    364  
southwest    325  
northwest    325  
northeast    324  
Name: region, dtype: int64
```



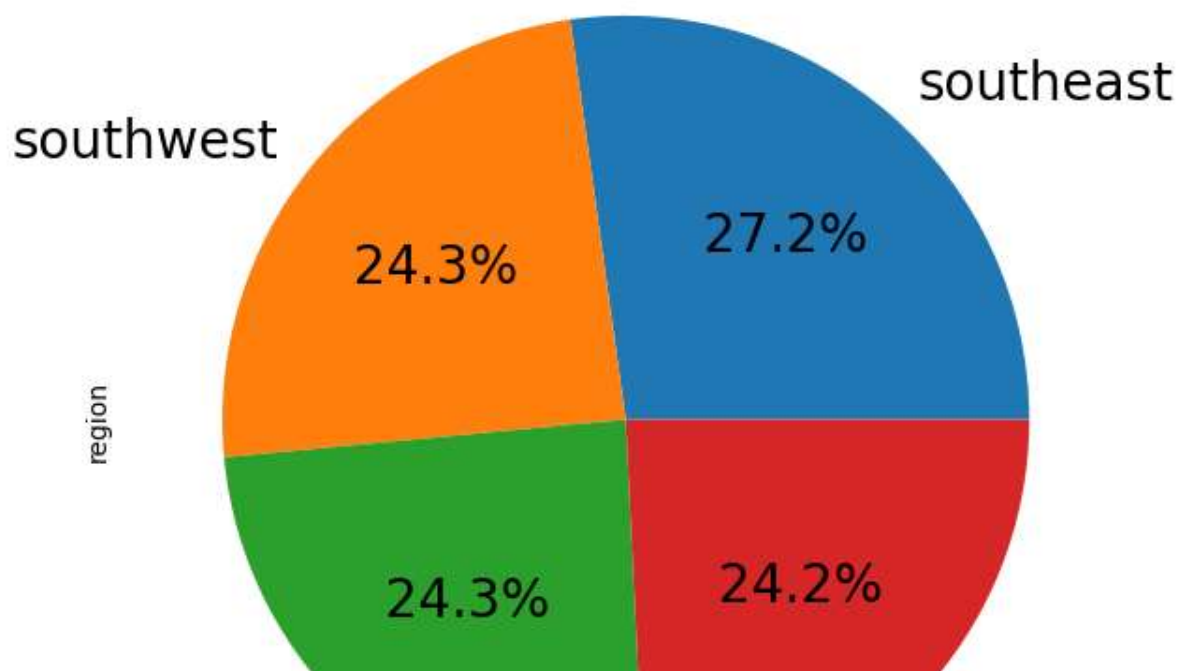
```
sns.countplot(x='region',data=df)
```

<Axes: xlabel='region', ylabel='count'>



```
plt.figure(figsize=(7,7))  
region.plot(kind='pie',fontsize=20,autopct='%1.1f%%')
```

<Axes: ylabel='region'>



```
#Encoding the column values by labelencoder
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['sex']=le.fit_transform(df['sex'])
df['smoker']=le.fit_transform(df['smoker'])
df['region']=le.fit_transform(df['region'])
df
```


	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

```
df.dtypes
```

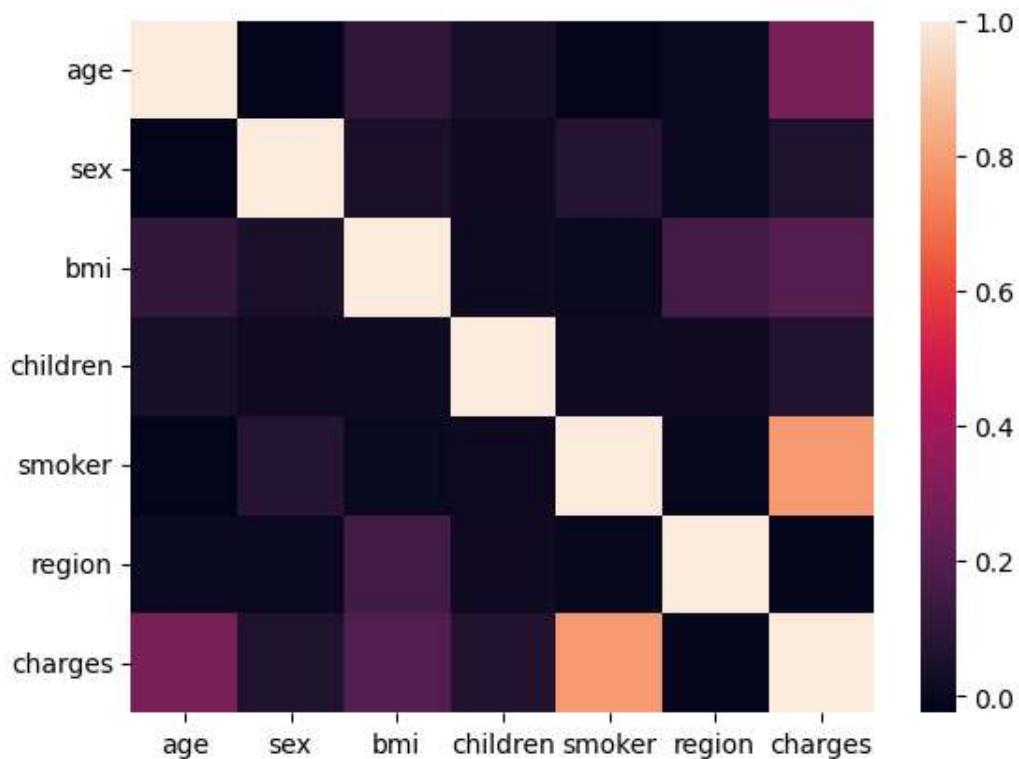
```
age          int64
sex          int64
bmi          float64
children     int64
smoker       int64
region       int64
charges      float64
dtype: object
```

```
df.corr()
```

	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	

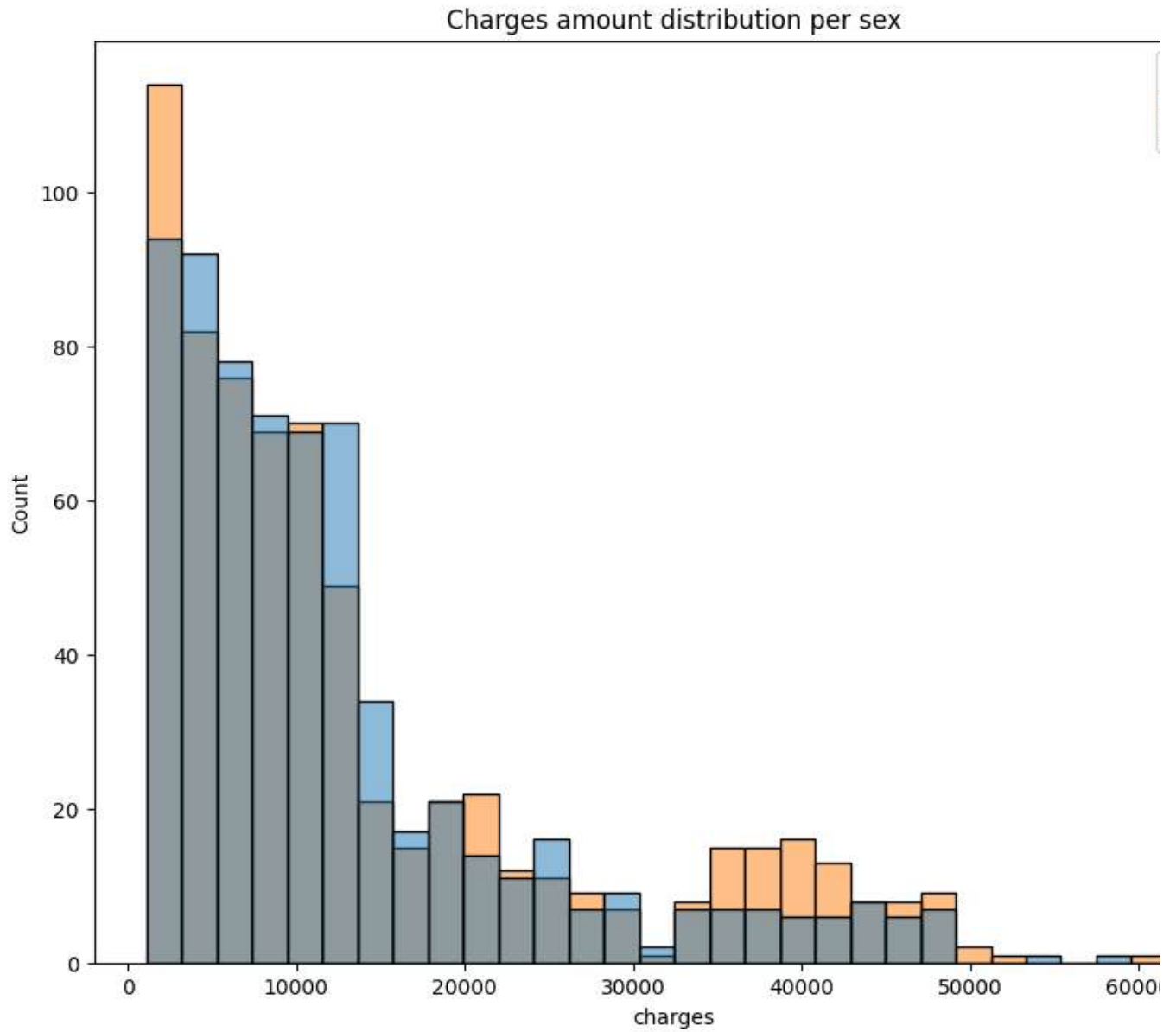
```
sns.heatmap(df.corr())
```

<Axes: >



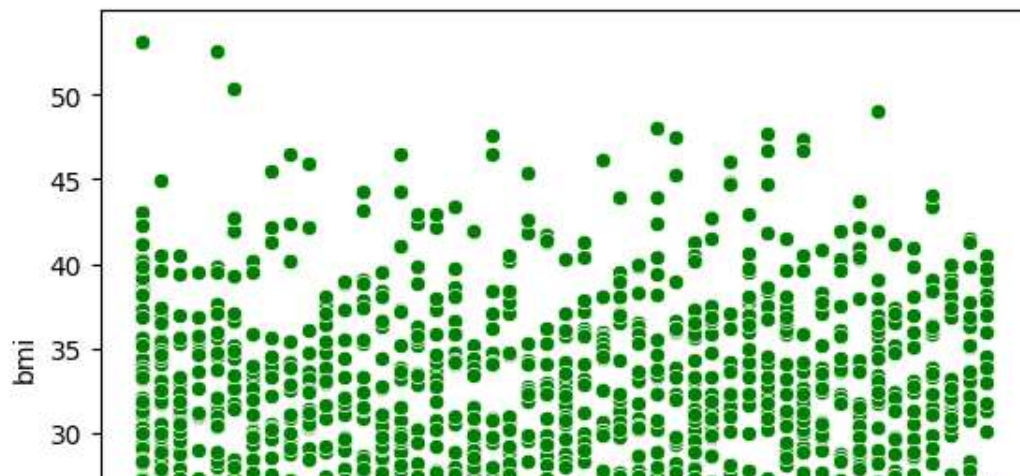

```
plt.figure(figsize=(10,8))
plt.title('Charges amount distribution per sex')
sns.histplot(df,x='charges',hue="sex")
```

<Axes: title={'center': 'Charges amount distribution per sex'}, xlabel='charges', ylabel='Count'



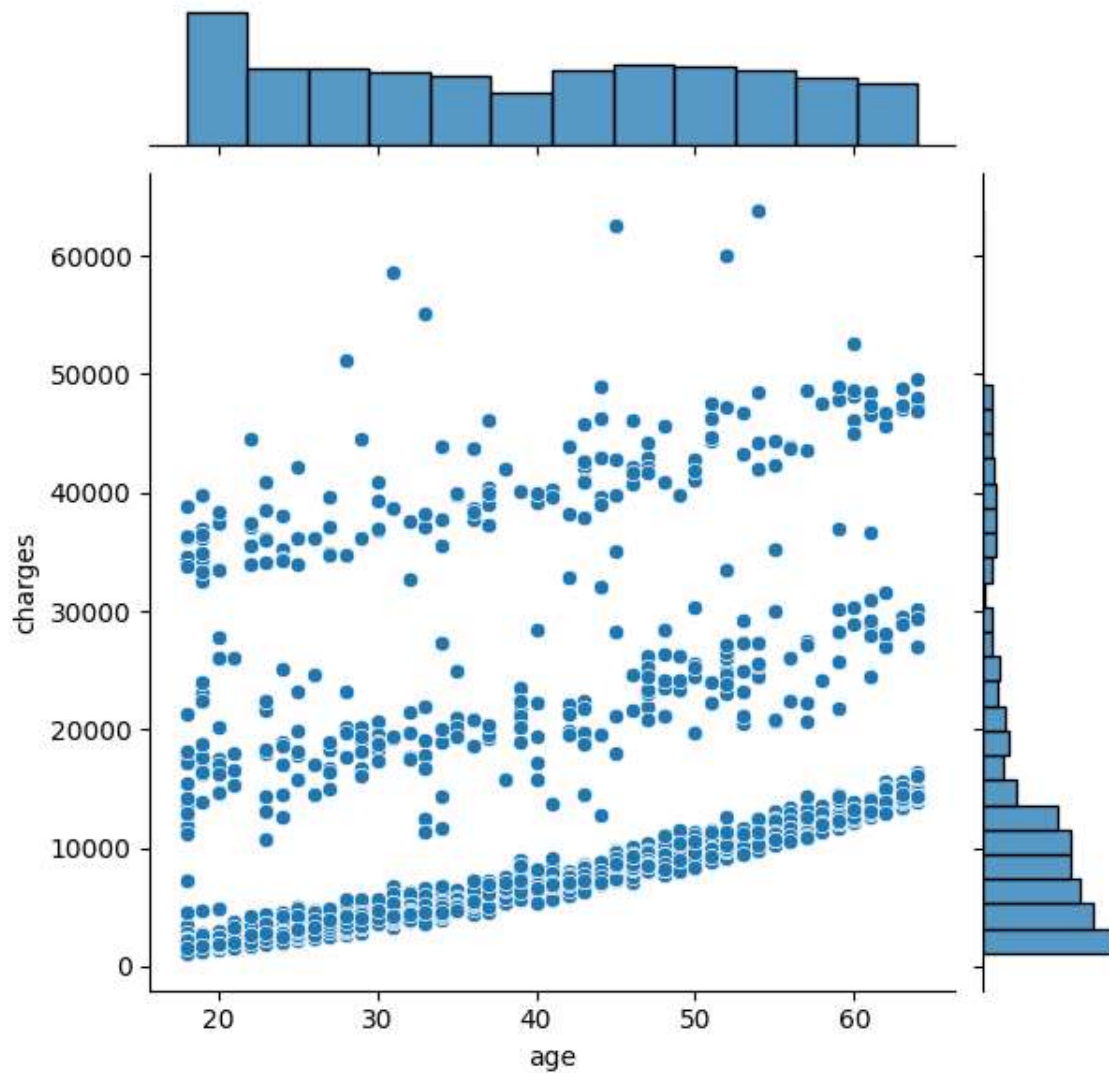
```
sns.scatterplot(data=df, x='age', y='bmi', color='green')
```

```
<Axes: xlabel='age', ylabel='bmi'>
```



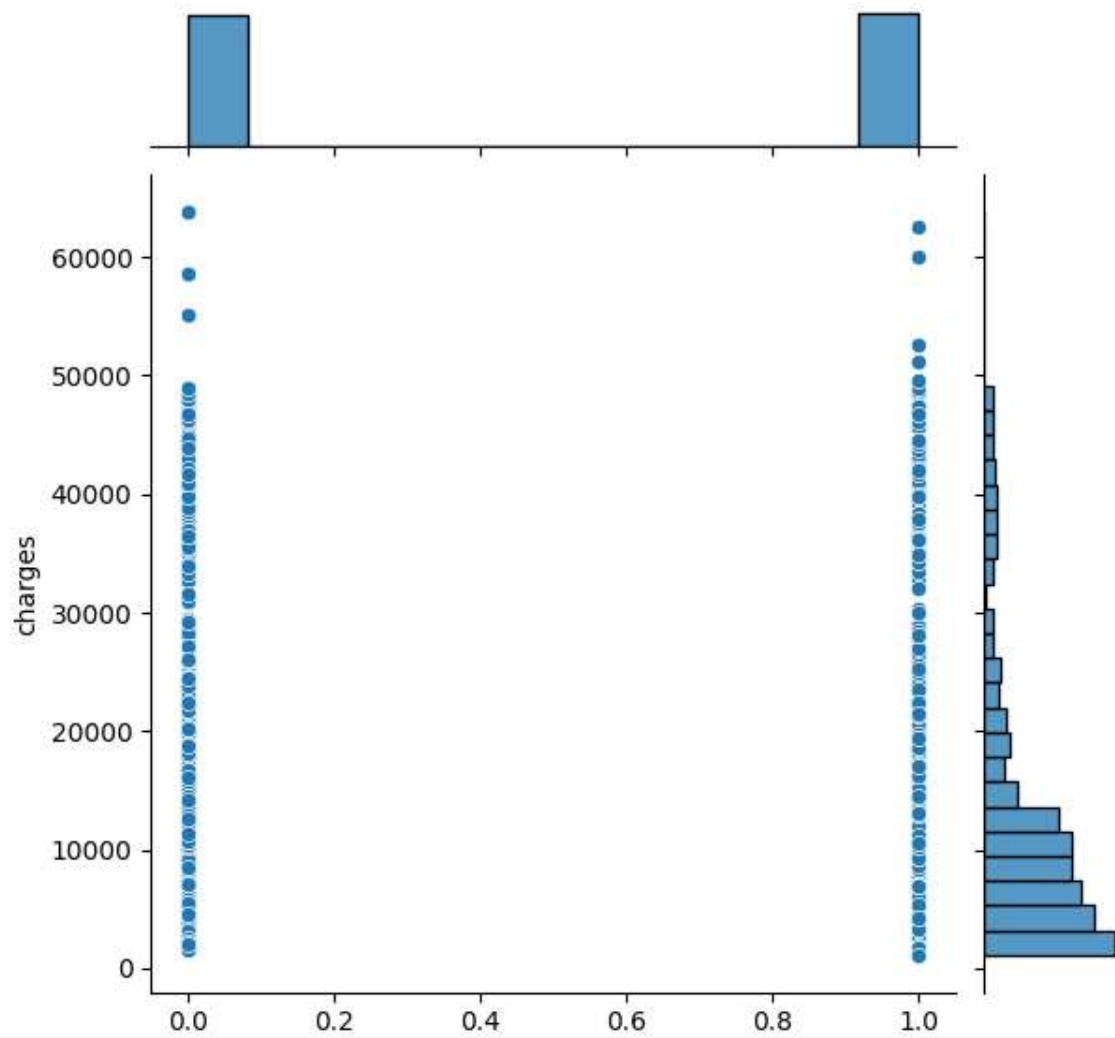
```
sns.jointplot(x='age', y='charges', data=df)
```

```
<seaborn.axisgrid.JointGrid at 0x7f3fb6567b50>
```



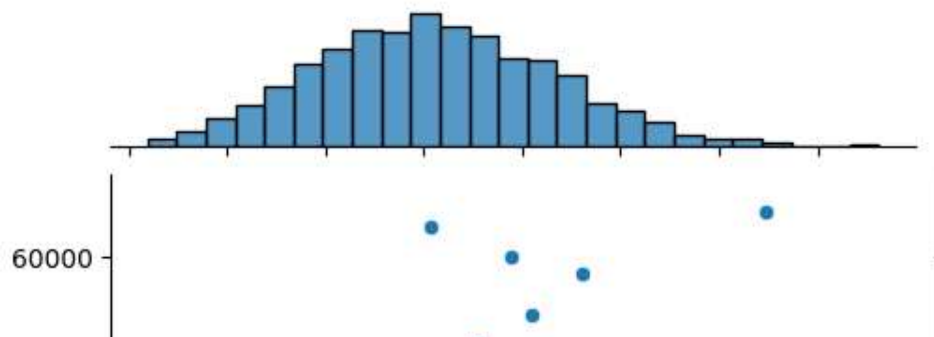
```
sns.jointplot(x='sex', y='charges', data=df)
```

<seaborn.axisgrid.JointGrid at 0x7f3fb63df700>



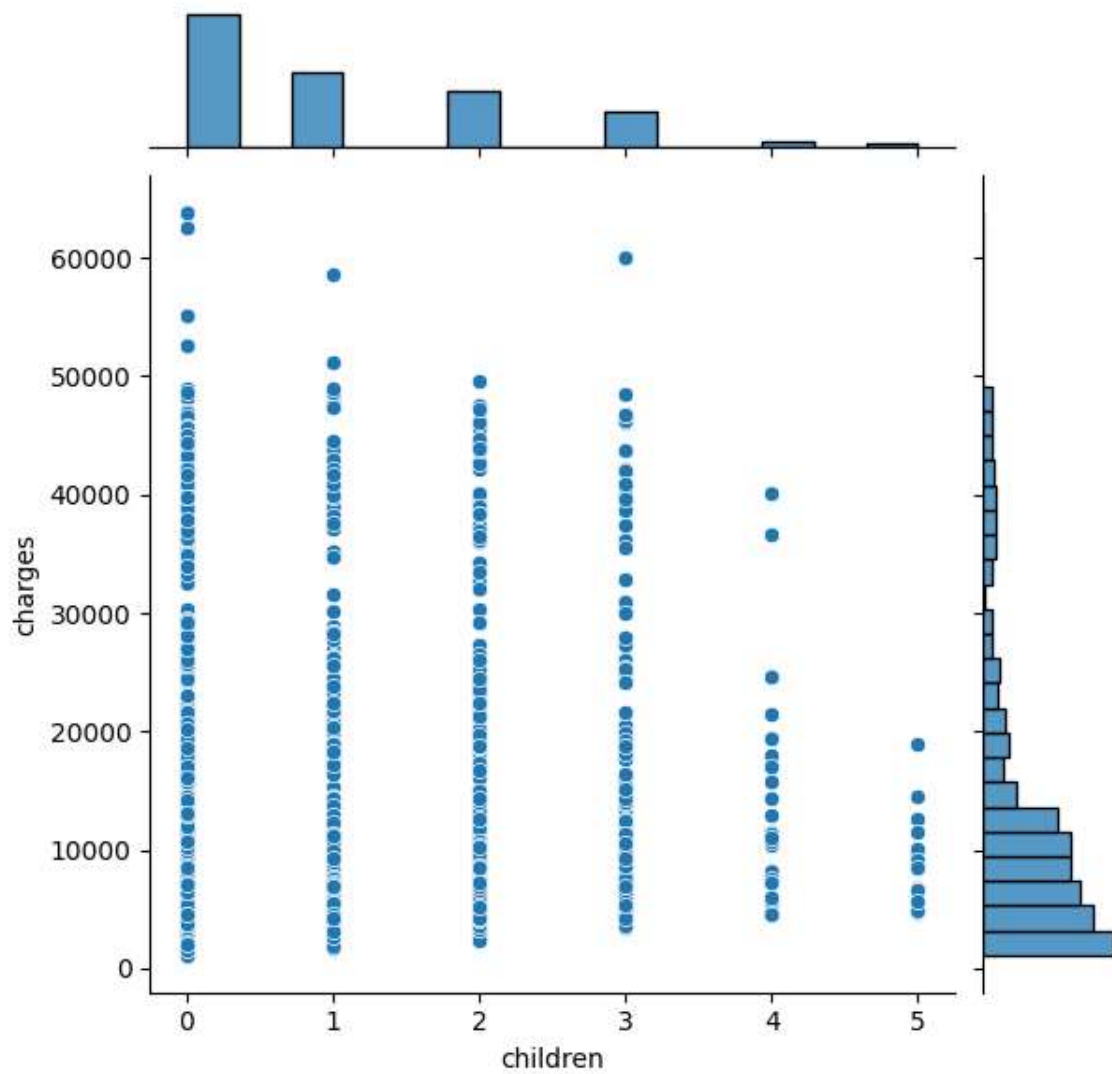
```
sns.jointplot(x='bmi', y='charges', data=df)
```

```
<seaborn.axisgrid.JointGrid at 0x7f3fba28a050>
```



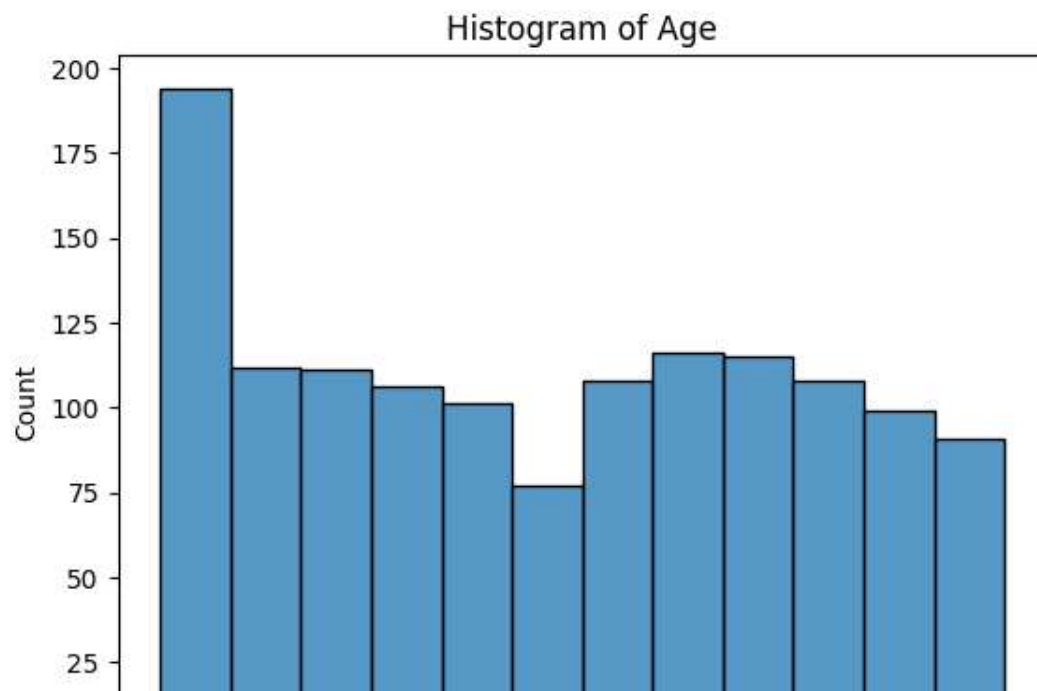
```
sns.jointplot(x='children', y='charges', data=df)
```

```
<seaborn.axisgrid.JointGrid at 0x7f3fba19f6d0>
```



```
sns.histplot(df['age'])  
plt.title('Histogram of Age')
```

```
Text(0.5, 1.0, 'Histogram of Age')
```

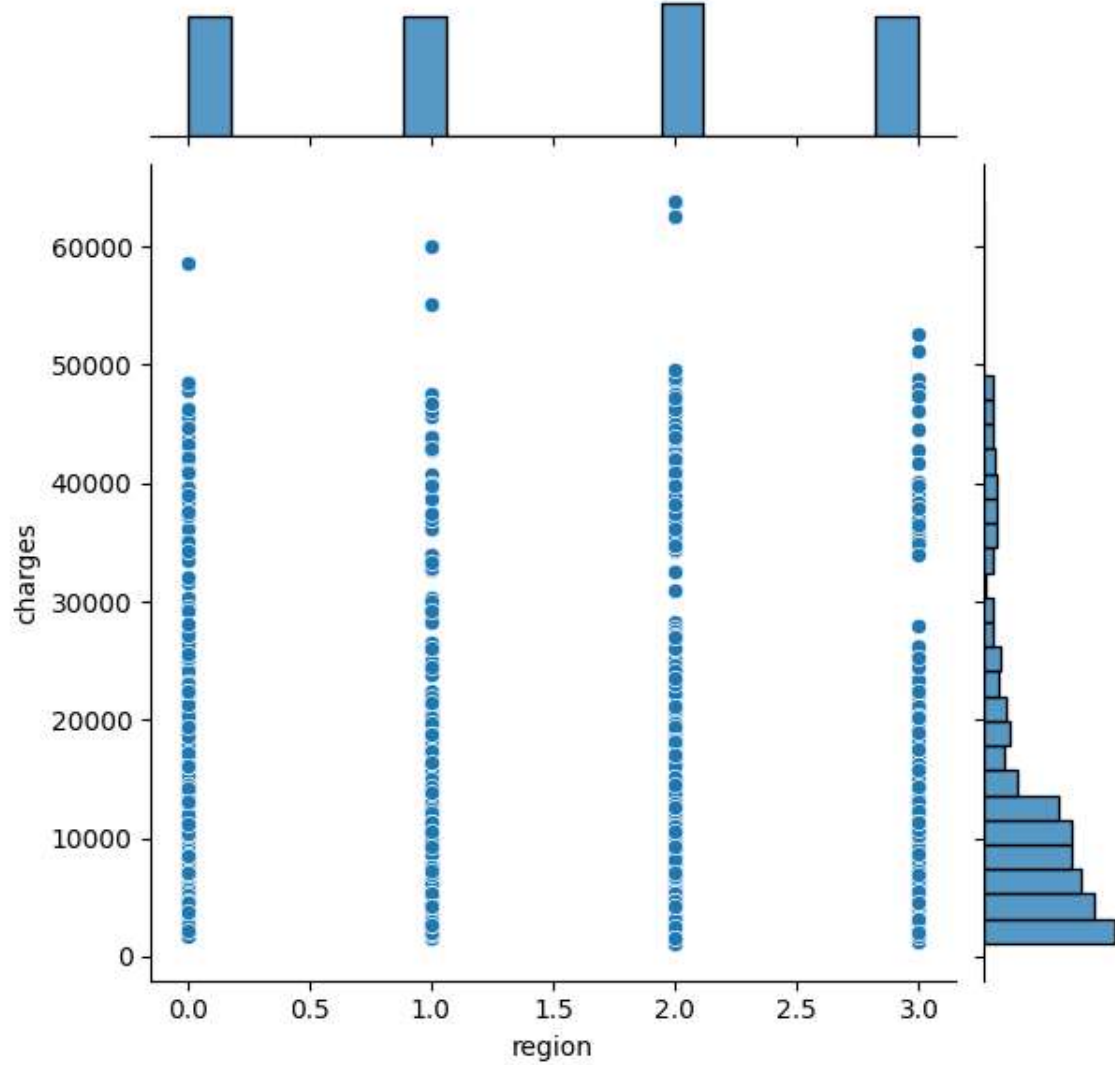


```
sns.jointplot(x='smoker', y='charges', data=df)
```

```
<seaborn.axisgrid.JointGrid at 0x7f3fb60e51e0>
```

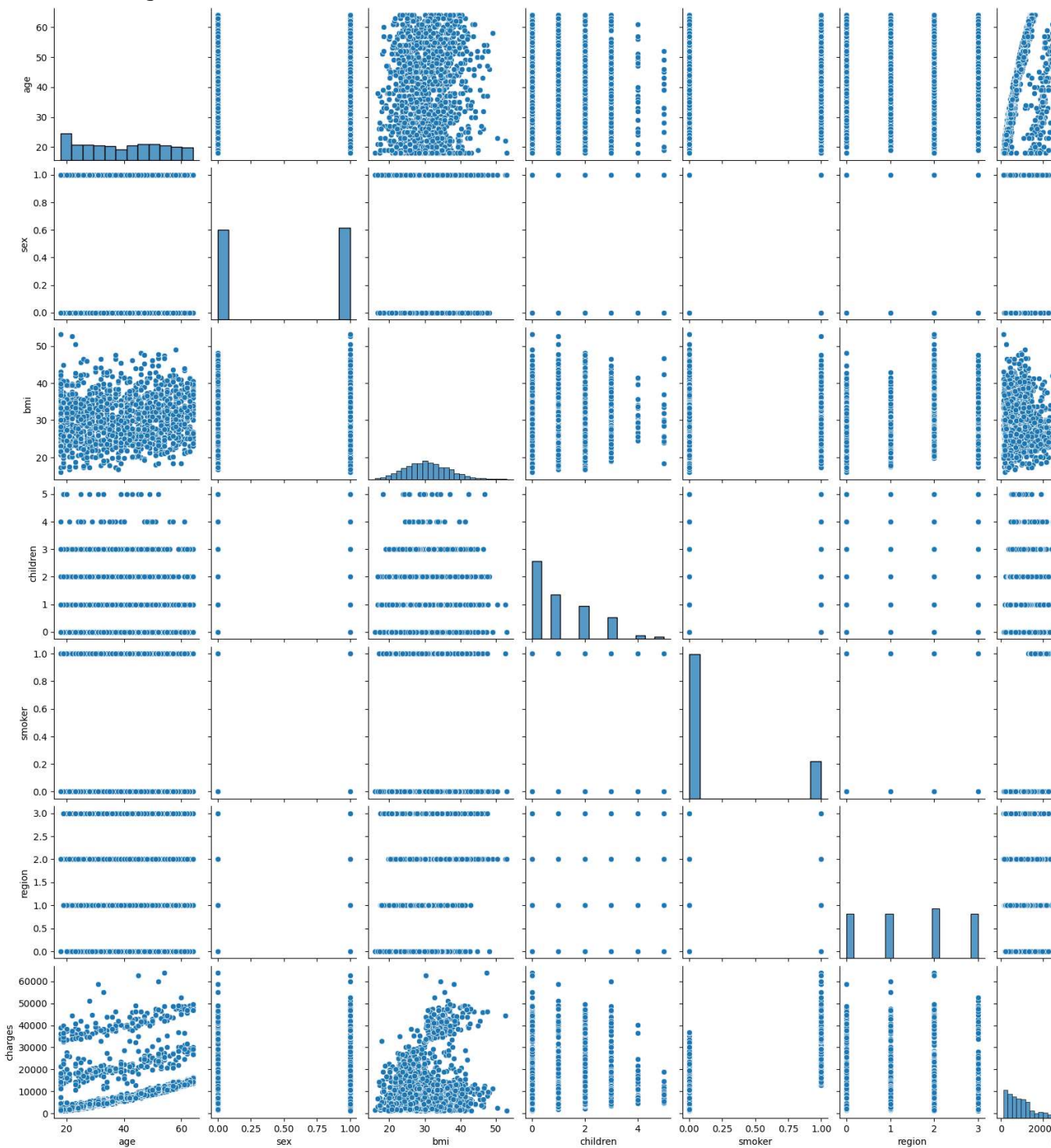
```
sns.jointplot(x='region', y='charges', data=df)
```

```
<seaborn.axisgrid.JointGrid at 0x7f3fb5b45ff0>
```



```
sns.pairplot(df)
```

<seaborn.axisgrid.PairGrid at 0x7f3fb5a2abc0>



```
#splitting data into input and output
x=df.drop('charges', axis=1).values
y=df['charges'].values
x
```

```
array([[19. ,  0. , 27.9 ,  0. ,  1. ,  3. ],
       [18. ,  1. , 33.77,  1. ,  0. ,  2. ],
       [28. ,  1. , 33. ,  3. ,  0. ,  2. ],
       ...,
       [18. ,  0. , 36.85,  0. ,  0. ,  2. ],
```

```
[21. , 0. , 25.8 , 0. , 0. , 3. ],  
[61. , 0. , 29.07, 0. , 1. , 1. ]])
```

```
#splitting into training and testing  
from sklearn.model_selection import train_test_split  
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)  
x_train
```

```
array([[61. , 0. , 31.16 , 0. , 0. , 1. ],  
       [46. , 1. , 27.6 , 0. , 0. , 3. ],  
       [54. , 0. , 31.9 , 3. , 0. , 2. ],  
       ...,  
       [58. , 1. , 25.175, 0. , 0. , 0. ],  
       [37. , 0. , 47.6 , 2. , 1. , 3. ],  
       [55. , 1. , 29.9 , 0. , 0. , 3. ]])
```

```
#preprocessing  
from sklearn.preprocessing import StandardScaler  
scaler=StandardScaler()  
scaler.fit(x_train)  
x_train=scaler.fit_transform(x_train)  
x_test=scaler.fit_transform(x_test)  
x_test
```

```
array([[ 0.41590161, -0.97542622, -0.91328391,  0.78934404, -0.49455237,  
        -1.48686599],  
       [-0.23268978, -0.97542622, -0.14666482, -0.89403157, -0.49455237,  
        -0.56016812],  
       [ 1.78515011, -0.97542622, -0.64271247, -0.89403157,  2.02203056,  
        -0.56016812],  
       ...,  
       [-1.52987257, -0.97542622, -0.43226801, -0.89403157, -0.49455237,  
        -1.48686599],  
       [ 1.35275585,  1.02519286,  0.8122024 , -0.89403157, -0.49455237,  
         0.36652976],  
       [-1.38574115,  1.02519286, -1.41566072, -0.05234377, -0.49455237,  
         1.29322763]])
```

```
#importing KNeighborsRegressor algorithm  
from sklearn.neighbors import KNeighborsRegressor  
kr_model=KNeighborsRegressor(n_neighbors=5)  
kr_model.fit(x_train,y_train)  
y_pred=kr_model.predict(x_test)  
y_pred
```


4656.57782 , 17231.720498, 18507.565466, 5554.01114 ,
 2419.49053 , 10939.079222, 6880.86507 , 42042.40241 ,
 2057.49774 , 28936.59906 , 2125.1372 , 1772.14282 ,
 14090.291566, 19199.137384, 1216.95898 , 11346.84806 ,
 5330.9212 , 32889.035338, 12621.99523 , 12844.769394,
 6924.31901 , 11831.919382, 41582.91025 , 1443.04368 ,
 13106.25306 , 40549.936768, 2444.19015 , 3936.7743 ,
 1713.94734 , 2057.49774 , 4879.04156 , 5488.06414 ,
 14165.417952, 5831.503738, 7516.616324, 7128.766 ,
 8083.450524, 13106.25306 , 7299.139544, 3936.7743 ,
 11058.7714 , 3950.33404 , 10323.83488 , 10294.151198,
 8348.43749 , 16851.233354, 21210.62118 , 45357.344518,
 15368.062834, 6757.66503 , 45158.429562, 8961.8956 ,
 9200.01641 , 12244.565872, 13277.284212, 9215.37292 ,
 12112.09587 , 11622.85018 , 5626.433834, 20377.536046,
 7255.9539 , 33192.65324 , 10026.352986, 12054.098622,
 10584.5644 , 16158.223932, 8456.08783 , 13021.030372,
 39457.187 , 2327.87621 , 12969.849432, 48077.66881 ,
 5725.492078, 5893.404426, 6191.08785 , 10136.514 ,
 16609.36492 , 12151.132154, 9457.75645 , 11835.27272 ,
 13951.784508, 2848.41898 , 1947.38988 , 37660.92983 ,
 29179.85038 , 8407.522444, 11717.552616, 18082.42362 ,
 15510.049066, 33999.96298 , 6392.27277 , 45357.344518,
 5724.069194, 1632.06584 , 5720.96728 , 5284.094022,
 17044.97833 , 11412.2838 , 1718.55756 , 15131.136452,
 13264.375514, 12412.49018 , 43406.685254, 12318.77439 ,
 33899.727452, 7219.9876 , 11997.2186 , 4601.64994 ,
 6453.67328 , 8308.2484 , 12717.42816 , 10018.8908 ,
 2336.75223 , 14708.037182, 12373.93055 , 11837.24074 ,
 6701.2988 , 21210.62118 , 8553.656666, 1711.88736 ,
 6156.09131 , 9545.6764 , 41616.34016 , 10099.1348 ,
 15899.376364, 7366.3048 , 9870.695222, 5069.5922 ,
 6392.27277 , 9310.05561 , 5600.67296 , 3750.99309 ,
 15616.186632, 11956.127392, 5769.6158 , 3515.89564 ,
 20740.275168, 4304.14631 , 40745.54989 , 36682.11139 ,
 43942.215622, 3695.217066, 7406.53778 , 9145.330014,
 3695.217066, 20185.63549 , 7621.07215 , 23543.896494,
 46994.937644, 5472.96164 , 11701.735 , 9601.062412,
 11634.932728, 17005.725986, 18698.73716 , 10602.18446 ,
 10528.812932, 6641.565494, 2457.54322 , 14614.878572,
 31916.180254, 14069.994986, 8298.479 , 2539.37692 ,
 9255.089174, 16190.148856, 13236.951794, 8731.6858 ,
 23437.9445 , 41157.4363 , 11701.735 , 7739.09495 ,
 12159.97171 , 13052.753262, 9946.889636, 3842.2775 ,
 7854.61843 , 4341.011156, 14868.954526, 5036.36592 ,
 12174.1164 , 11437.629724, 4190.09236 , 6249.6658 ,
 13479.058194, 1573.08676])

```
#Print the performance measurments
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
print('MAE is',mean_absolute_error(y_test,y_pred))
print('"ERROR percentage is',mean_absolute_percentage_error(y_test,y_pred))
print("MSE is",mean_squared_error(y_test,y_pred))
print('r2 score is',r2_score(y_test,y_pred))
```

MAE is 3247.8028918283585
"ERROR percentage is 0.38752417094116953
MSE is 26341783.423568398
r2 score is 0.8203450450890505

```
#importing LinearRegression algorithm
from sklearn.linear_model import LinearRegression
lr_model=LinearRegression()
lr_model.fit(x_train,y_train)
y_pred=lr_model.predict(x_test)
y_pred
```

```
1.25359981e+04,  5.04018189e+03,  1.80941994e+03,  3.29314735e+04,
2.54576992e+04,  1.78058043e+04,  2.71013602e+04,  1.05175454e+04,
3.80821082e+04, -3.73972648e+02,  7.21908690e+03,  7.59253270e+03,
```

```

5.75865511e+04, 5.46468429e+03, 8.91958308e+03, 8.48382797e+03,
3.30277370e+03, 3.23849170e+04, 7.38664825e+03, 2.92354911e+04,
3.67855810e+04, 7.22656543e+03, 1.30341510e+04, 9.88651168e+03,
8.63497999e+03, 1.29119460e+04, 3.10060371e+04, 1.70909378e+04,
1.18845254e+04, 4.18429763e+03, -1.19301448e+03, 1.24193800e+04,
3.14097311e+04, 1.32916319e+04, 1.13494450e+04, 7.82261784e+03,
3.74893887e+03, 7.17681754e+03, 7.32454981e+03, 1.07047098e+04,
3.45647879e+04, 3.97509161e+04, 1.20117615e+04, 8.51633709e+03,
1.66579524e+04, 1.57510602e+04, 9.52642046e+03, 9.04504686e+03,
9.05615515e+03, 2.73864361e+03, 1.01507490e+04, 4.11341233e+03,
1.07702258e+04, 1.60369007e+04, 6.83142865e+03, 2.11926630e+03,
1.47713078e+04, 1.53387334e+02])

```

```

#Print the performance measurments
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
print('MAE is',mean_absolute_error(y_test,y_pred))
print("ERROR percentage is",mean_absolute_percentage_error(y_test,y_pred))
print("MSE is",mean_squared_error(y_test,y_pred))
print('r2 score is',r2_score(y_test,y_pred))

```

```

MAE is 4132.993280433952
"ERROR percentage is 0.4363267999563344
MSE is 33823743.84386106
r2 score is 0.7693169411699159

```

```

#importing DecisionTreeRegression algorithm
from sklearn.tree import DecisionTreeRegressor
der_model=DecisionTreeRegressor()
der_model.fit(x_train,y_train)
y_pred=der_model.predict(x_test)
y_pred

```

1552.4697 , 9174.13565, 4058.1161 , 25065.4207 , 10197.7722 ,
 8124.4084 , 4058.71245, 6474.013 , 45702.02235, 1526.312 ,
 13012.20865, 36197.699 , 2850.68375, 3392.3652 , 1632.56445,
 2850.68375, 3847.674 , 4074.4537 , 10381.4787 , 1625.43375,
 1877.9294 , 23563.01618, 3353.284 , 11833.7823 , 2709.24395,
 3227.1211 , 12557.6053 , 2902.9065 , 25656.57526, 4661.28635,
 8162.71625, 13616.3586 , 20745.9891 , 60021.39897, 12224.35085,
 6555.07035, 48675.5177 , 9880.068 , 9500.57305, 20878.78443,
 8782.469 , 7418.522 , 12479.70895, 13887.9685 , 2007.945 ,
 12829.4551 , 6600.20595, 40103.89 , 6746.7425 , 9386.1613 ,
 25656.57526, 14410.9321 , 14478.33015, 25656.57526, 36837.467 ,
 3046.062 , 9549.5651 , 47462.894 , 4441.21315, 2850.68375,
 2203.73595, 11455.28 , 16776.30405, 10141.1362 , 14478.33015,
 10577.087 , 9630.397 , 3561.8889 , 2156.7518 , 39611.7577 ,
 62592.87309, 7265.7025 , 2304.0022 , 15518.18025, 12928.7911 ,
 37742.5757 , 21344.8467 , 45008.9555 , 5846.9176 , 1629.8335 ,
 5926.846 , 1842.519 , 14571.8908 , 13844.7972 , 1711.0268 ,
 27322.73386, 11353.2276 , 13224.05705, 43753.33705, 12269.68865,
 35069.37452, 7682.67 , 15555.18875, 4149.736 , 7518.02535,
 24603.04837, 14007.222 , 10577.087 , 2899.48935, 11365.952 ,
 13429.0354 , 11743.9341 , 4746.344 , 17352.6803 , 6571.02435,
 2395.17155, 3847.674 , 8798.593 , 41034.2214 , 8516.829 ,
 28923.13692, 6877.9801 , 4618.0799 , 4234.927 , 21344.8467 ,
 7518.02535, 5974.3847 , 1621.8827 , 11552.904 , 7419.4779 ,
 5926.846 , 3180.5101 , 8782.469 , 2498.4144 , 37742.5757 ,
 36149.4835 , 46255.1125 , 18955.22017, 7518.02535, 6358.77645,
 1877.9294 , 21880.82 , 7358.17565, 17663.1442 , 41676.0811 ,
 5240.765 , 13041.921 , 5729.0053 , 10156.7832 , 10269.46 ,
 21677.28345, 11093.6229 , 5729.0053 , 4134.08245, 1832.094 ,
 11658.11505, 19040.876 , 11945.1327 , 9144.565 , 2850.68375,
 16796.41194, 5966.8874 , 8539.671 , 8825.086 , 24667.419 ,
 47291.055 , 11455.28 , 7419.4779 , 13143.86485, 26392.26029,
 6571.544 , 18955.22017, 6796.86325, 3981.9768 , 7789.635 ,
 4134.08245, 11833.7823 , 9875.6804 , 3935.1799 , 2196.4732 ,
 11743.9341 , 1711.0268])

```

#Print the performance measurments
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
print('MAE is',mean_absolute_error(y_test,y_pred))
print('"ERROR percentage is',mean_absolute_percentage_error(y_test,y_pred))
print("MSE is",mean_squared_error(y_test,y_pred))
print('r2 score is',r2_score(y_test,y_pred))

```

```

MAE is 3201.0457401218905
"ERROR percentage is 0.37126675826119404
MSE is 45047823.9088777
r2 score is 0.6927670141747189

```

✓ 0s completed at 3:40 PM

