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

```
df=pd.read_csv('/content/insurance.csv')
```

```
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
```

```
df['sex'].value_counts()
```

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

```
df.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

```
df.isnull().sum()
```

```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
```

```
charges     0
dtype: int64
```

```
df.columns.unique()
```

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

```
df['sex'].unique()
```

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

```
df['region'].unique()
```

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

```
df.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

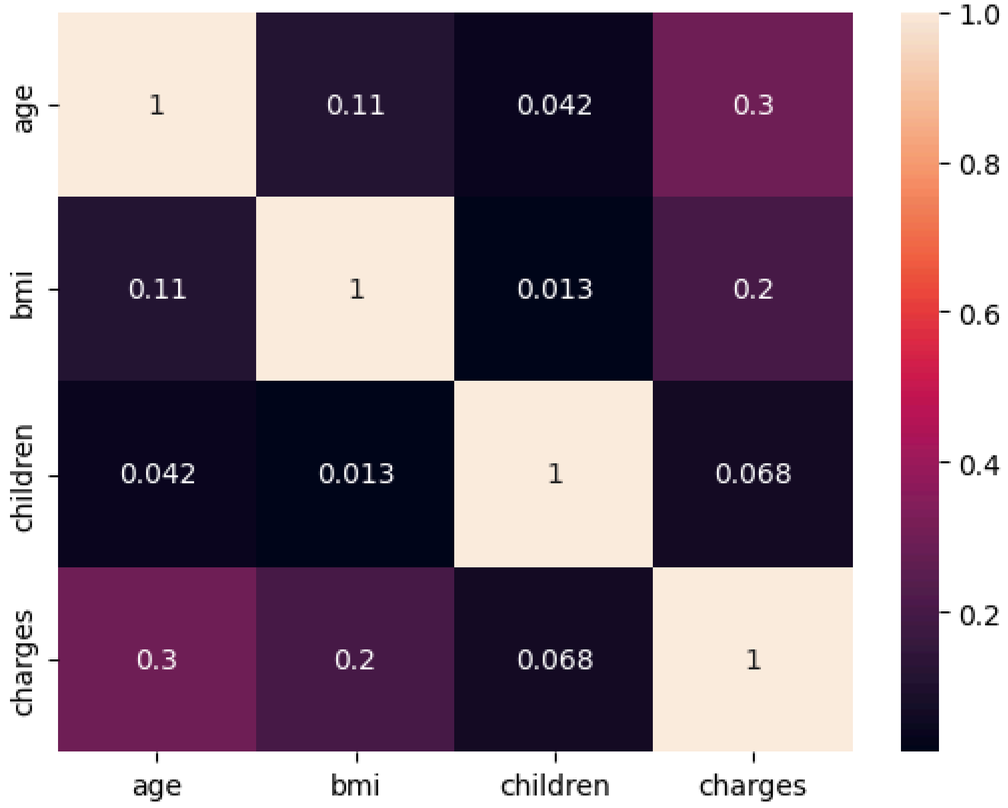
```
df.corr()
```

```
<ipython-input-23-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only i
  df.corr()
```

	age	bmi	children	charges
age	1.000000	0.109272	0.042469	0.299008
bmi	0.109272	1.000000	0.012759	0.198341
children	0.042469	0.012759	1.000000	0.067998
charges	0.299008	0.198341	0.067998	1.000000

```
sns.heatmap(df.corr(),annot=True)
```

```
<ipython-input-25-8df7bcac526d>:1: FutureWarning: The default value of numeric_only i
  sns.heatmap(df.corr(),annot=True)
<Axes: >
```

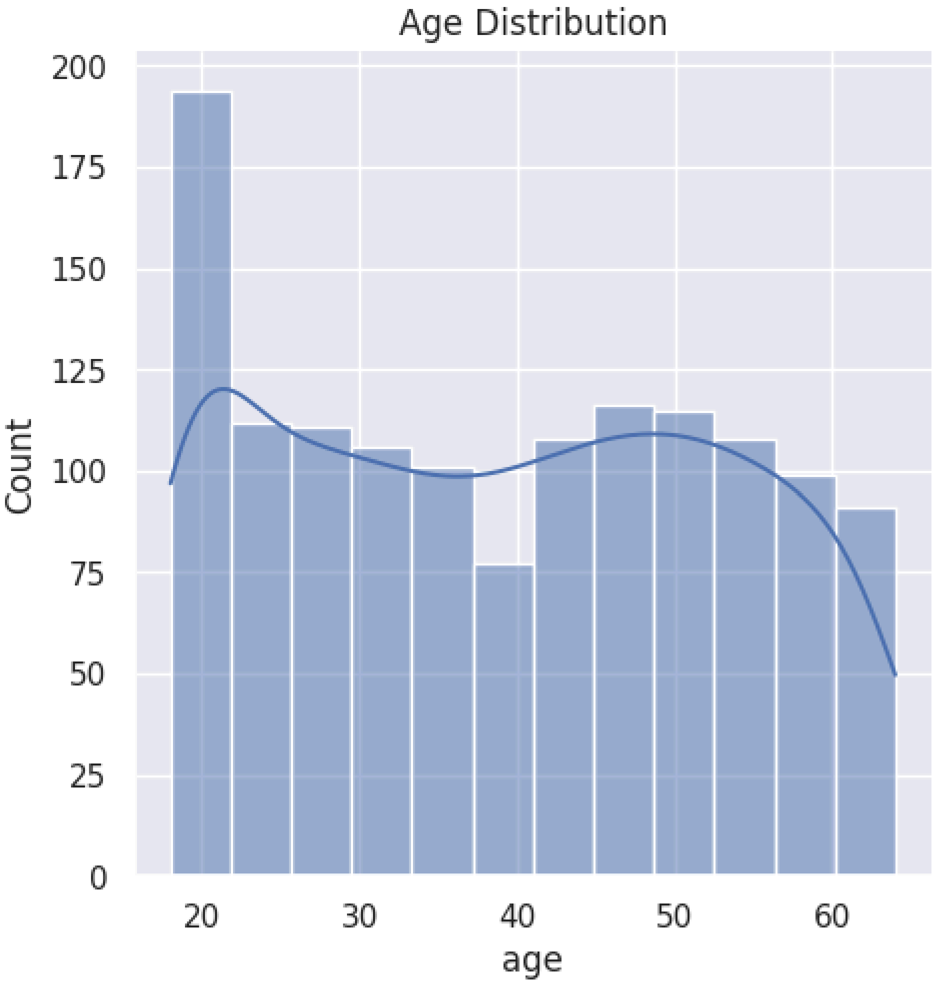


```
df.shape
```

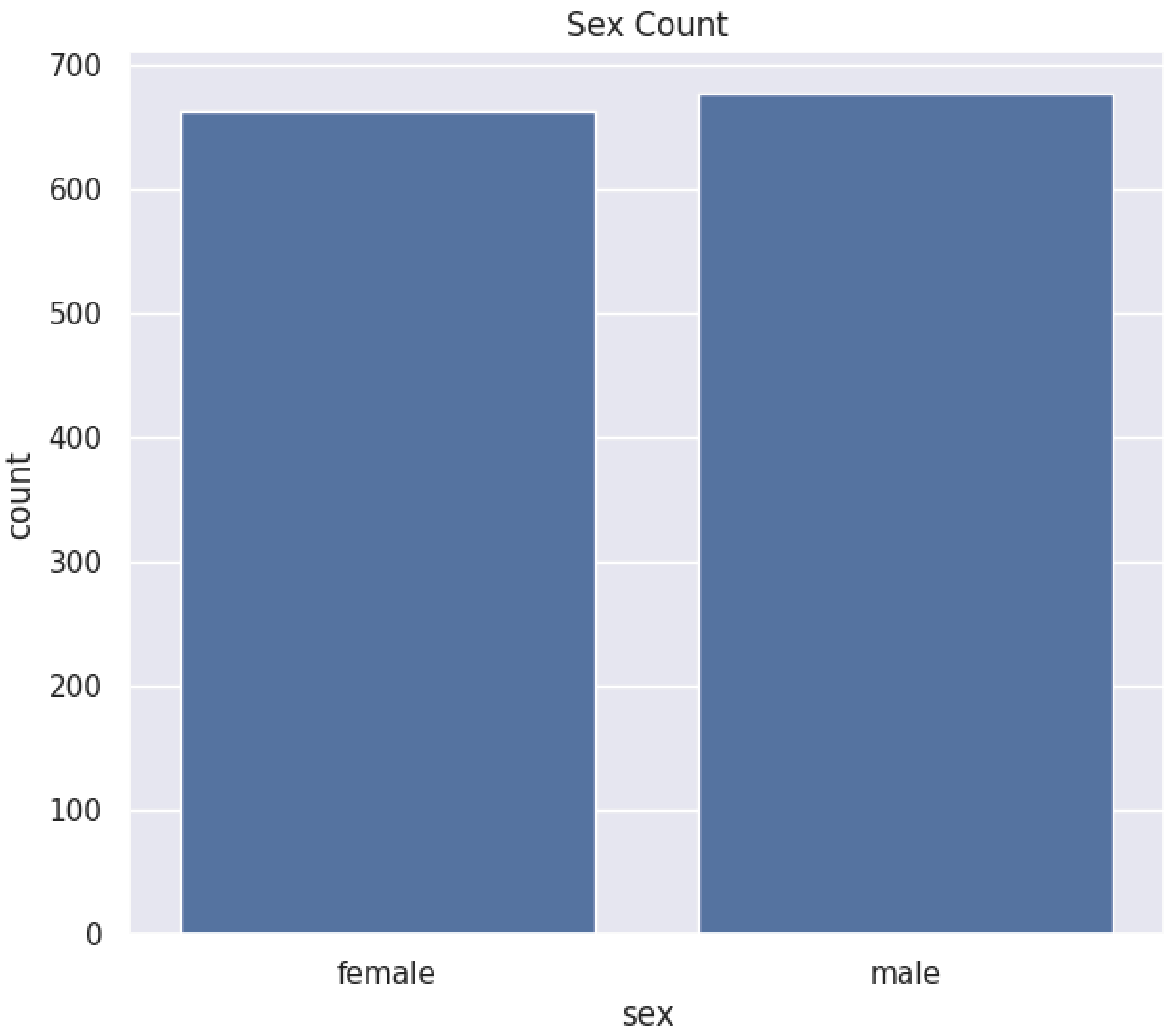
```
(1338, 7)
```

```
plt.figure(figsize=(10,6))
sns.displot(df['age'],kde=True)
plt.title('Age Distribution')
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```

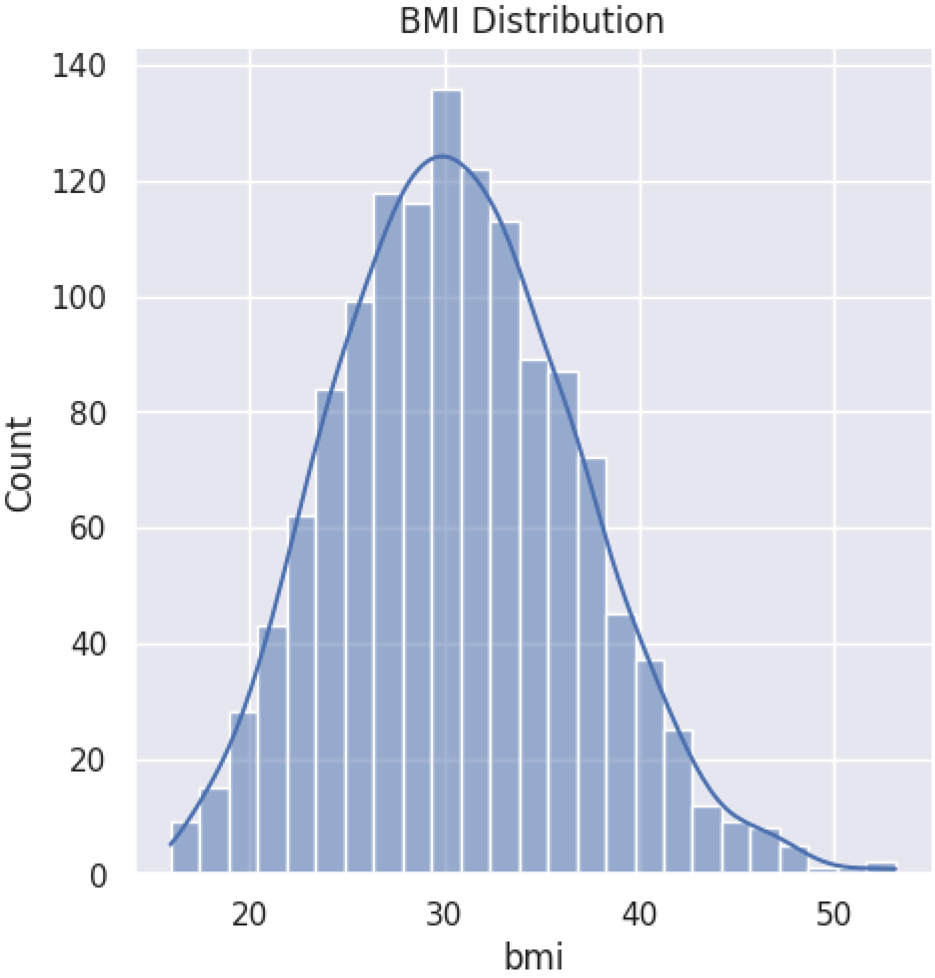


```
plt.figure(figsize=(7,6))
sns.countplot(x='sex',data=df)
plt.title('Sex Count')
plt.show()
```



```
plt.figure(figsize=(10,6))
sns.displot(df['bmi'],kde=True)
plt.title('BMI Distribution')
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```

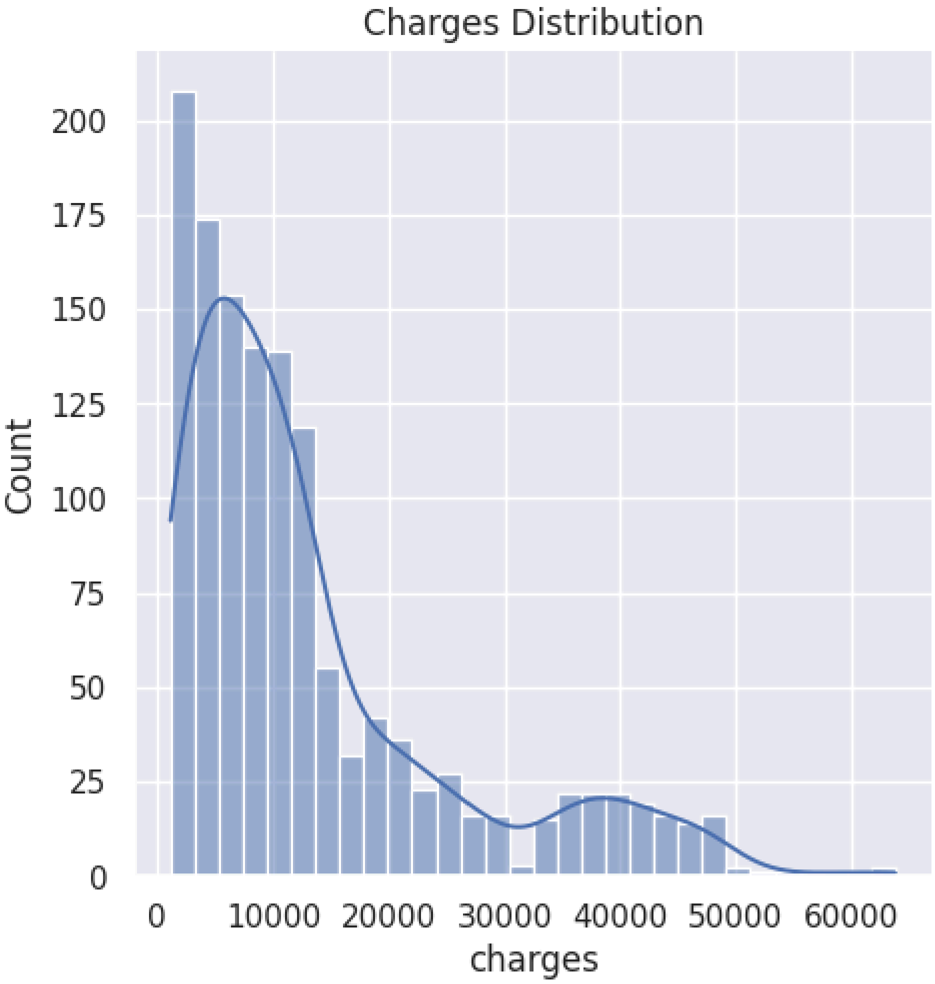


```
df['smoker'].value_counts()
```

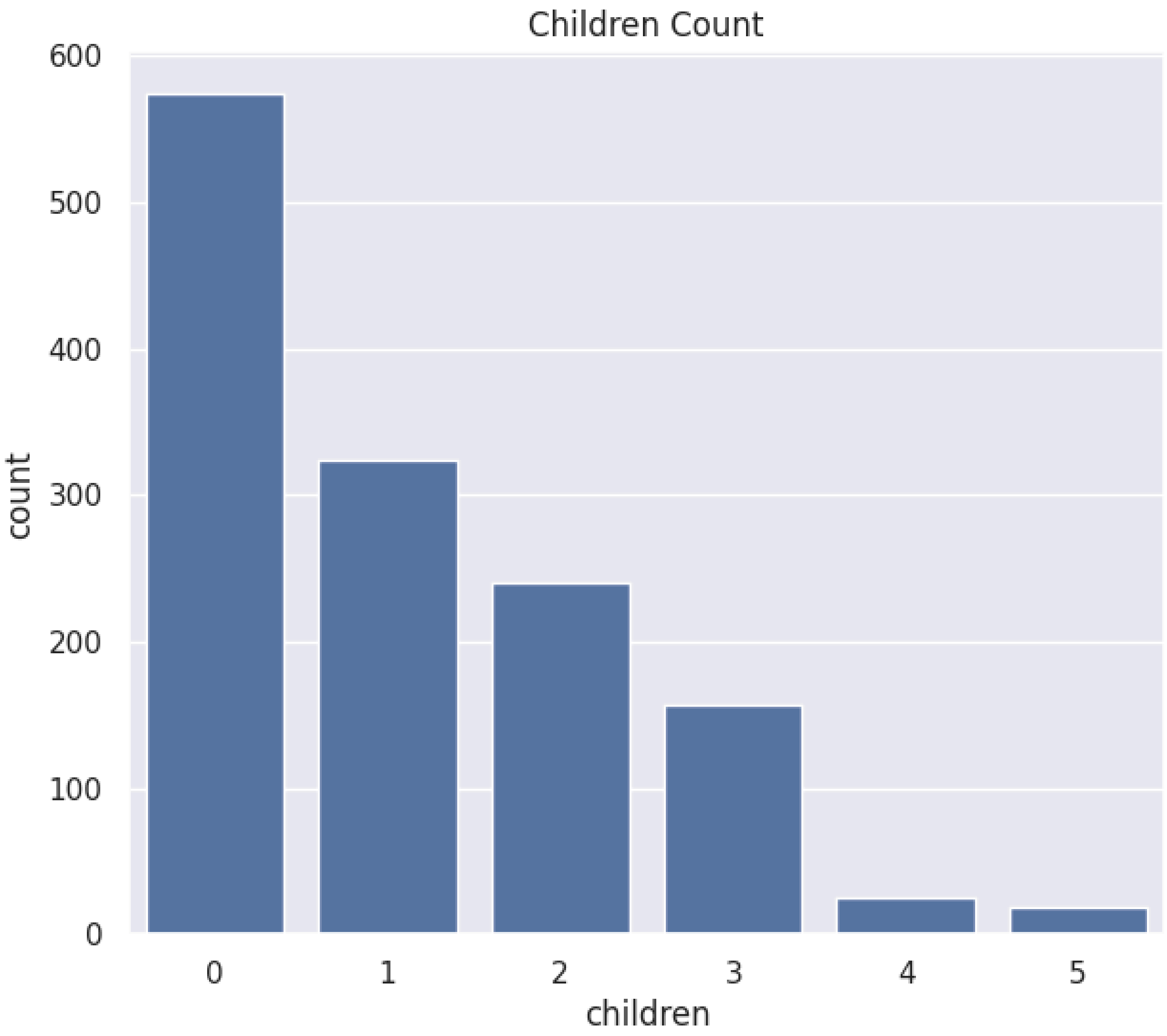
```
no     1064
yes     274
Name: smoker, dtype: int64
```

```
plt.figure(figsize=(10,6))
sns.displot(df['charges'],kde=True)
plt.title('Charges Distribution')
plt.show()
```

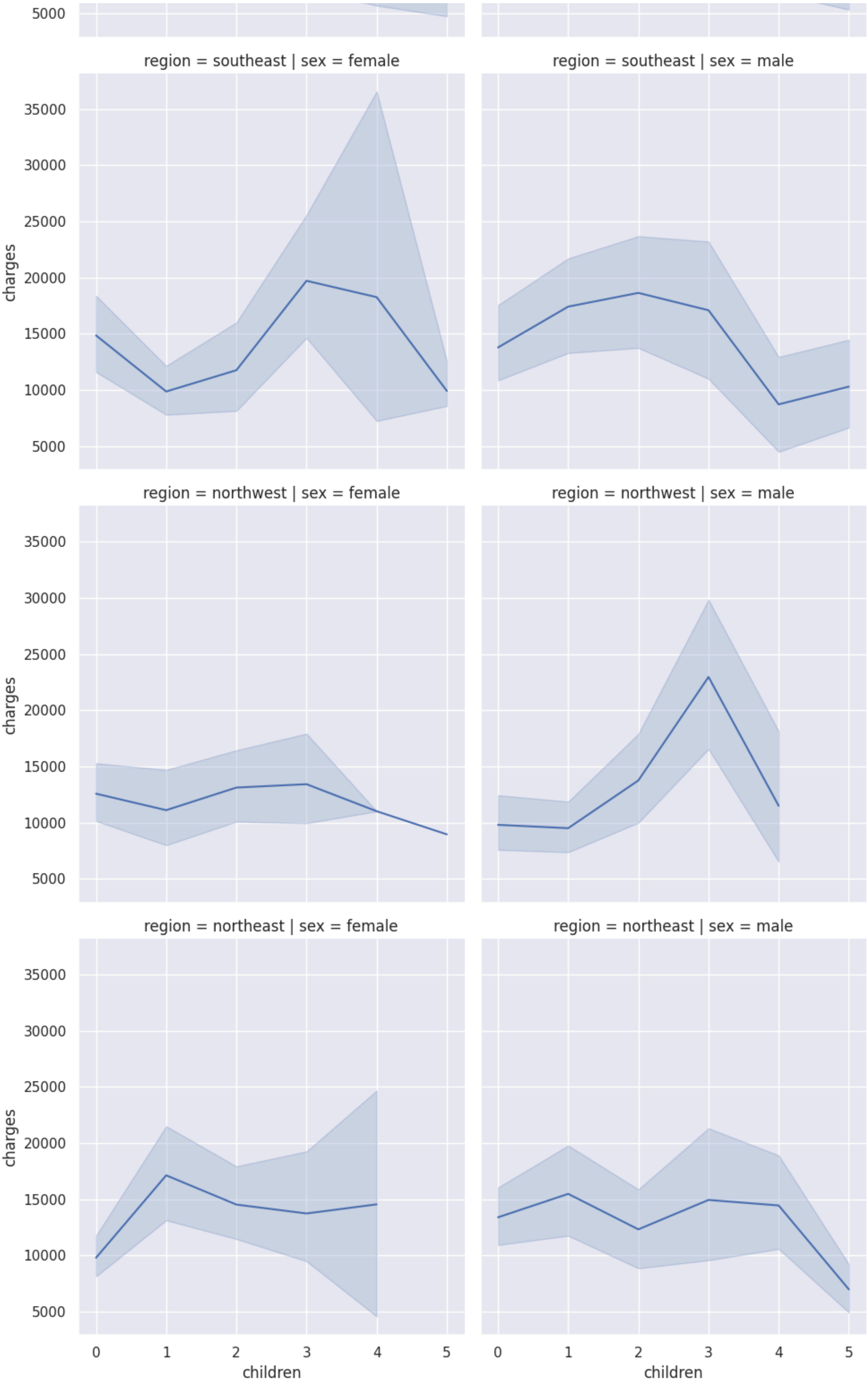
```
<Figure size 1000x600 with 0 Axes>
```



```
plt.figure(figsize=(7,6))
sns.countplot(x='children',data=df)
plt.title('Children Count')
plt.show()
```

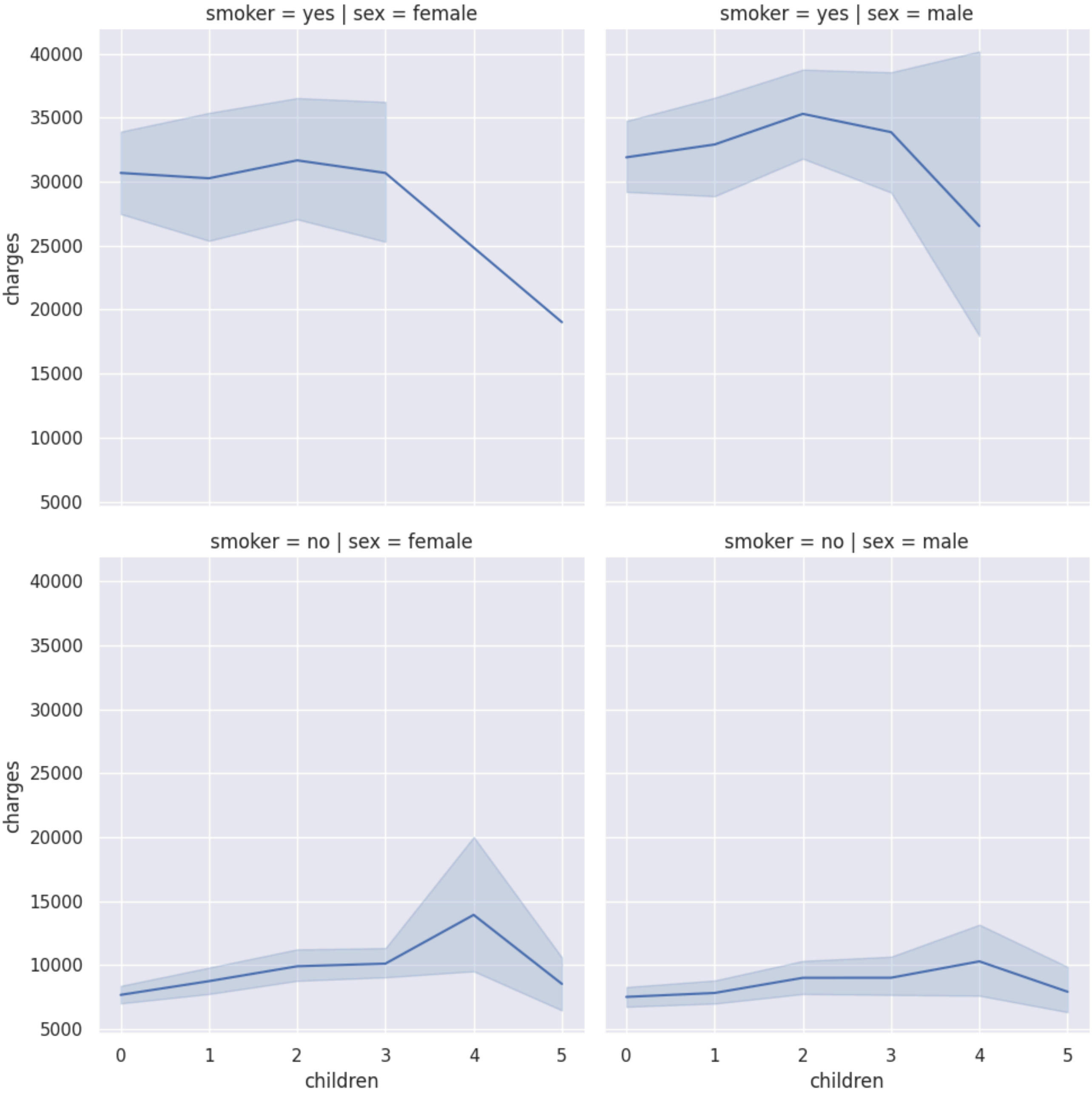


```
sns.relplot(kind='line',data=df,x='children',y='charges',col='sex',row='region')
```


5000
region = southeast | sex = female
region = southeast | sex = male
region = northwest | sex = female
region = northwest | sex = male
region = northeast | sex = female
region = northeast | sex = male
charges
35000
30000
25000
20000
15000
10000
5000
0
1
2
3
4
5
children

```
sns.relplot(kind='line',data=df,x='children',y='charges',col='sex',row='smoker')
```

```
<seaborn.axisgrid.FacetGrid at 0x7c54661436a0>
```



```
df.replace({'sex':{'male':0,'female':1}},inplace=True)
df.replace({'smoker':{'no':0,'yes':1}},inplace=True)
```

```
df['region'].str.strip(' ')
```

```
0       southwest
1       southeast
2       southeast
3       northwest
4       northwest
          ...
1333    northwest
1334    northeast
1335    southeast
1336    southwest
1337    northwest
Name: region, Length: 1338, dtype: object
```

```
df.replace({'region':{'southwest':0,'southeast':1,'northwest':2,'northeast':3}},inplace=T
```

```
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   int64
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64
 4   smoker    1338 non-null   int64
 5   region    1338 non-null   int64
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(5)
memory usage: 73.3 KB
```

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

```
1    364
0    325
2    325
3    324
Name: region, dtype: int64
```

```
X=df.drop('charges',axis=1)
y=df['charges']
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=42)
```

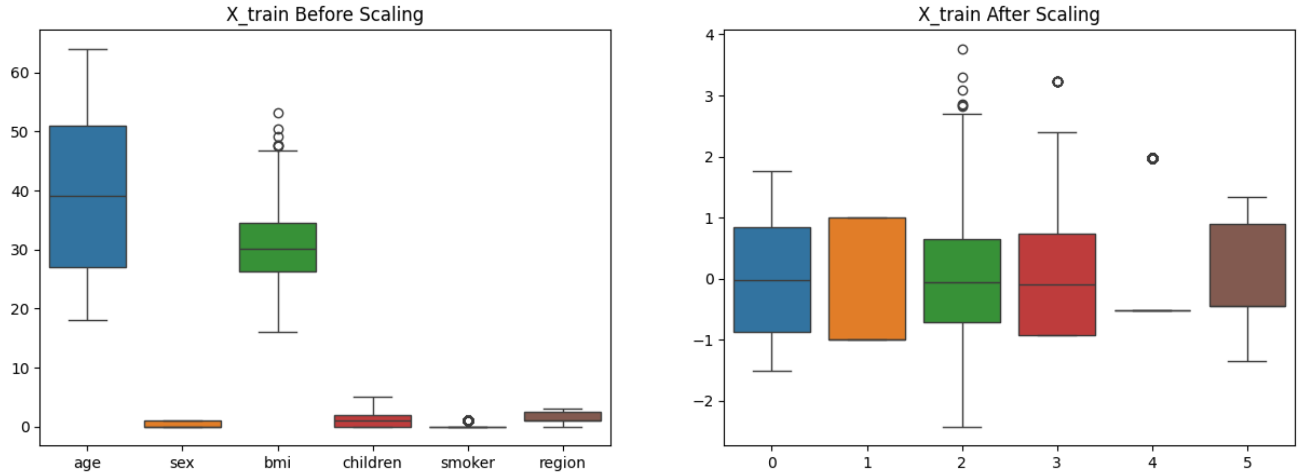
```
(X_train.shape,y_train.shape),(X_test.shape,y_test.shape)
```

```
(((1003, 6), (1003,)), ((335, 6), (335,)))
```

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_train_scaled=scaler.fit_transform(X_train)
X_test_scaled=scaler.transform(X_test)
```

```
plt.subplots(figsize=(15, 5))
plt.subplot(1, 2, 1)
sns.boxplot(data=X_train)
plt.title('X_train Before Scaling')
plt.subplot(1, 2, 2)
sns.boxplot(data=X_train_scaled)
plt.title('X_train After Scaling')
```

```
<ipython-input-34-41fb1d7ced73>:2: MatplotlibDeprecationWarning: Auto-removal of over
  plt.subplot(1, 2, 1)
Text(0.5, 1.0, 'X_train After Scaling')
```



```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
linreg=LinearRegression()
linreg.fit(X_train_scaled,y_train)
y_pred=linreg.predict(X_test_scaled)
mae=mean_absolute_error(y_test,y_pred)
score=r2_score(y_test,y_pred)
print("Mean absolute error", mae)
print("R2 Score", score)
```