Medical cost Insurance prediction

In [44]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
```

In [2]:

```
df = pd.read_csv('medical_cost_insurance.csv')
```

In [4]:

```
df.head()
```

Out[4]:

	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 [5]:

```
df.shape
```

Out[5]:

```
(1338, 7)
```

In [8]:

```
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 [6]:

```
df.describe()
```

Out[6]:

	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 [7]:

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

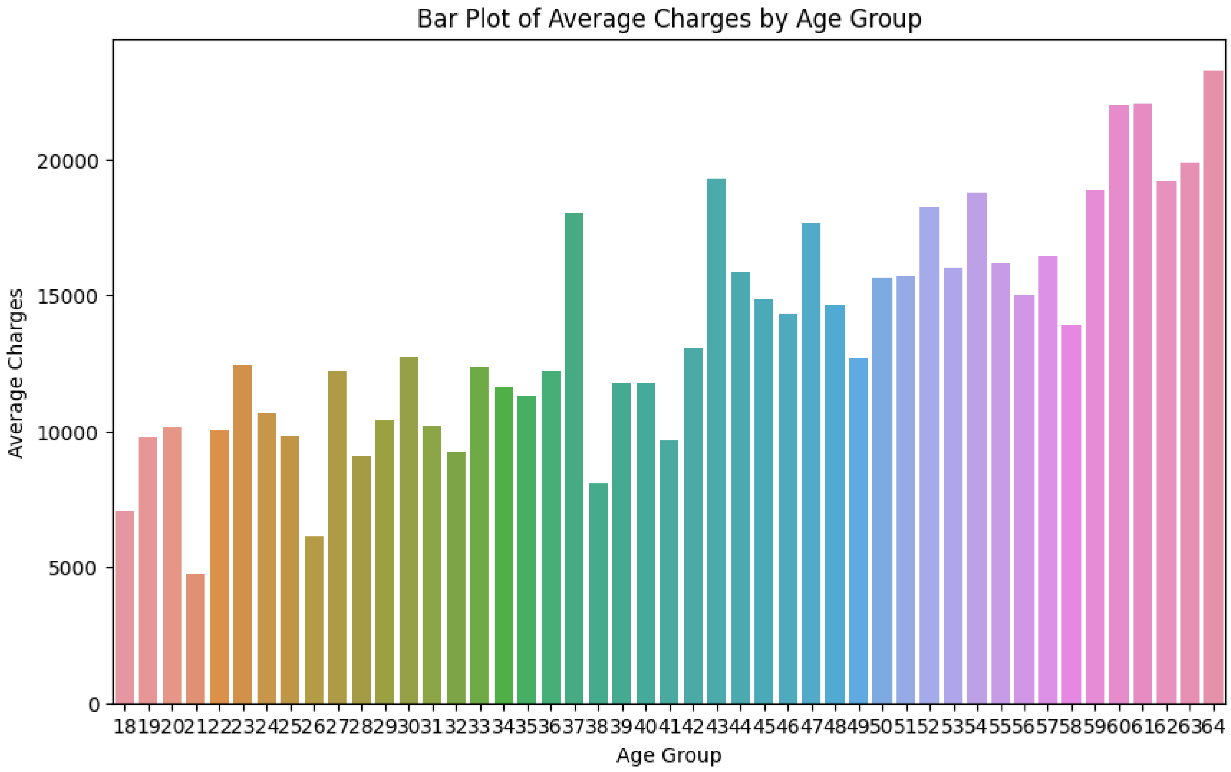
Out[7]:

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

In [10]:

```
plt.figure(figsize=(10, 6))
sns.barplot(x='age', y='charges', data=df, estimator=np.mean, ci=None)
plt.title('Bar Plot of Average Charges by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Average Charges')
plt.show()
```

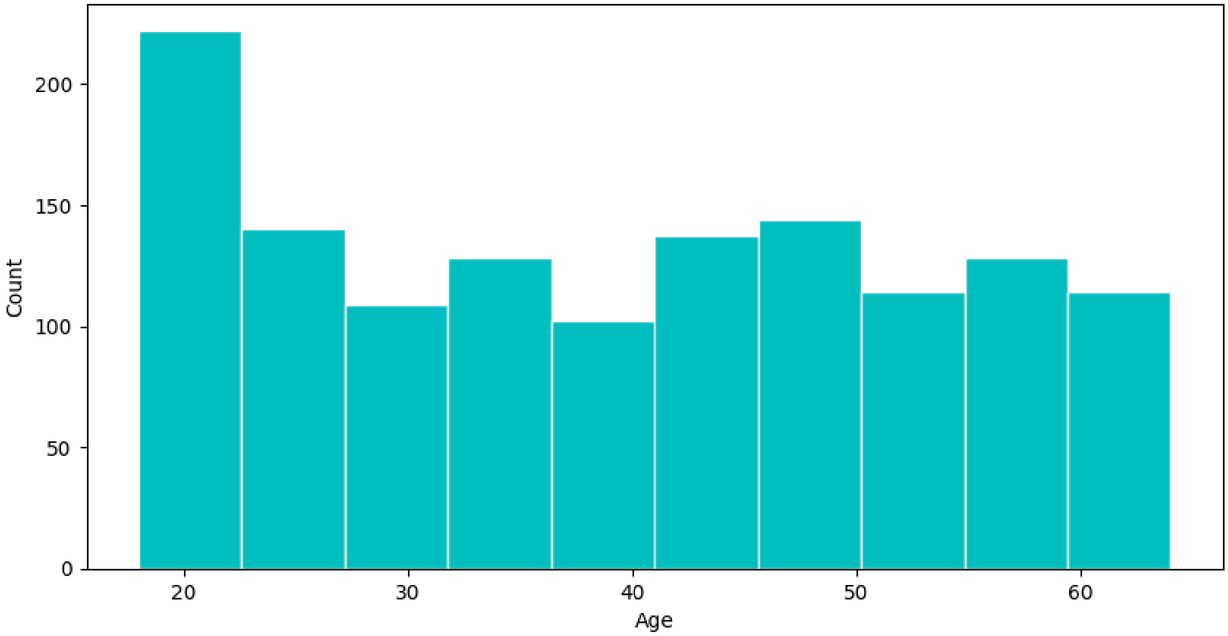
```
C:\Users\99Minds-1\AppData\Local\Temp\ipykernel_13464\2043728585.py:2: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='age', y='charges', data=df, estimator=np.mean, ci=None)
```



numerical variable plots

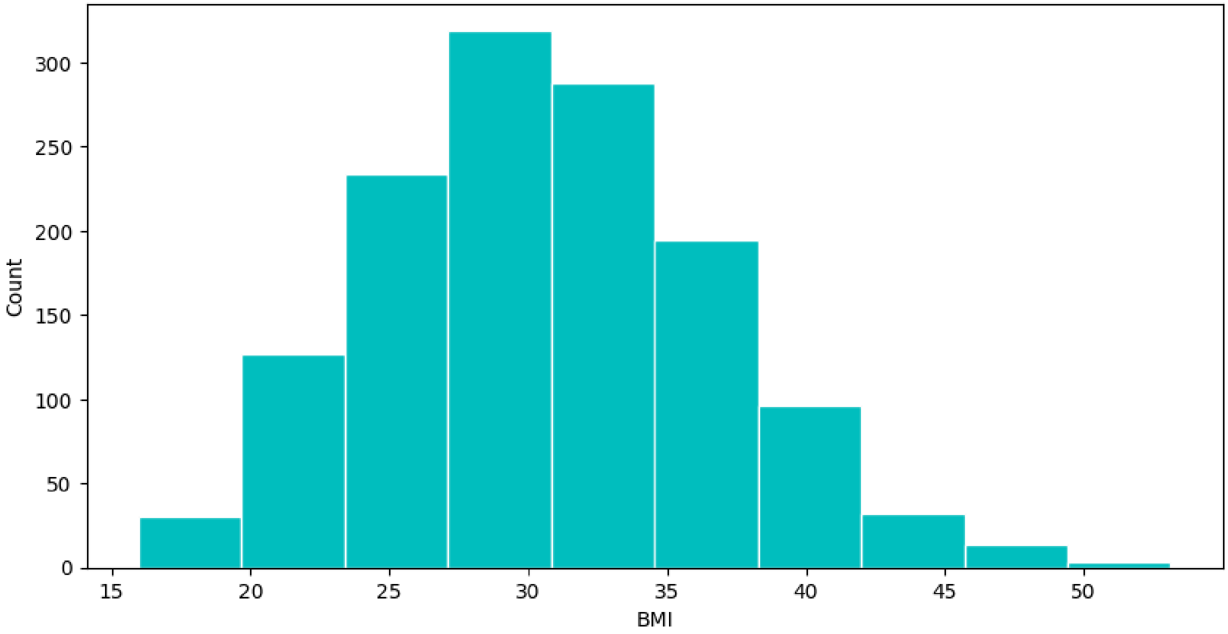
In [29]:

```
plt.figure(figsize=(10,5))
plt.hist(df['age'], edgecolor='white', label='d',color='c')
plt.xlabel("Age")
plt.ylabel("Count")
plt.title = ('Age Distrubtion in the Dataset')
```



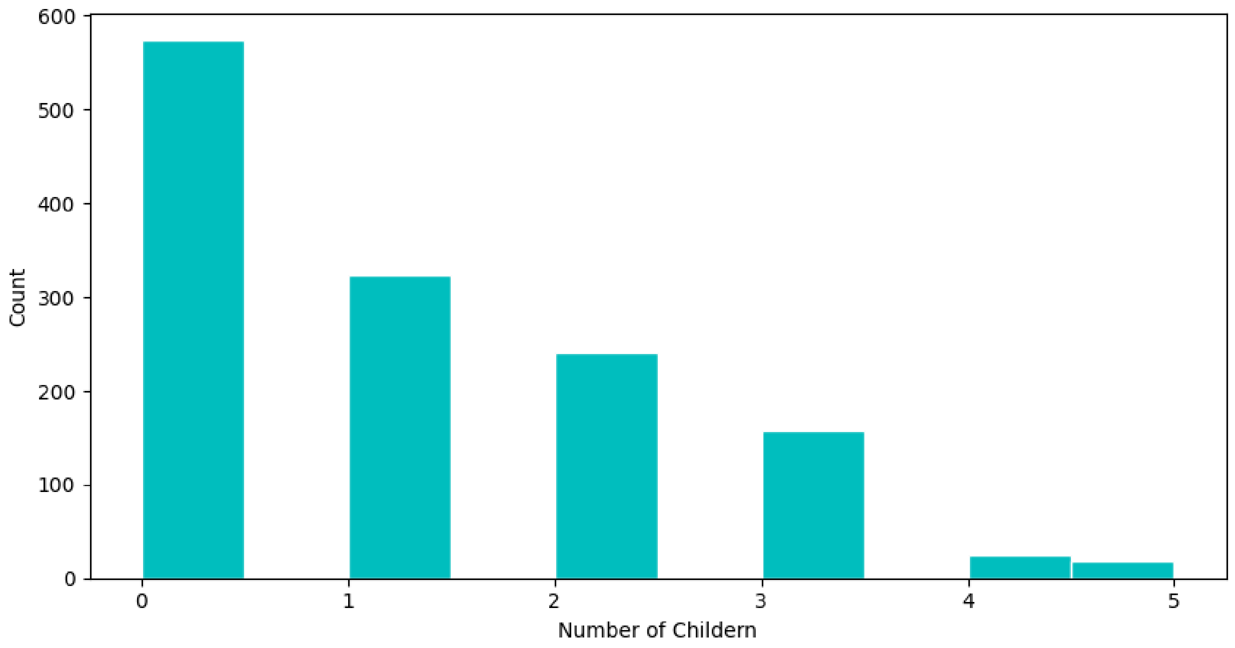
In [30]:

```
plt.figure(figsize=(10,5))
plt.hist(df['bmi'], edgecolor='white', label='d',color='c')
plt.xlabel("BMI")
plt.ylabel("Count")
plt.title = ('BMI Distrubtion in the Dataset')
```



In [31]:

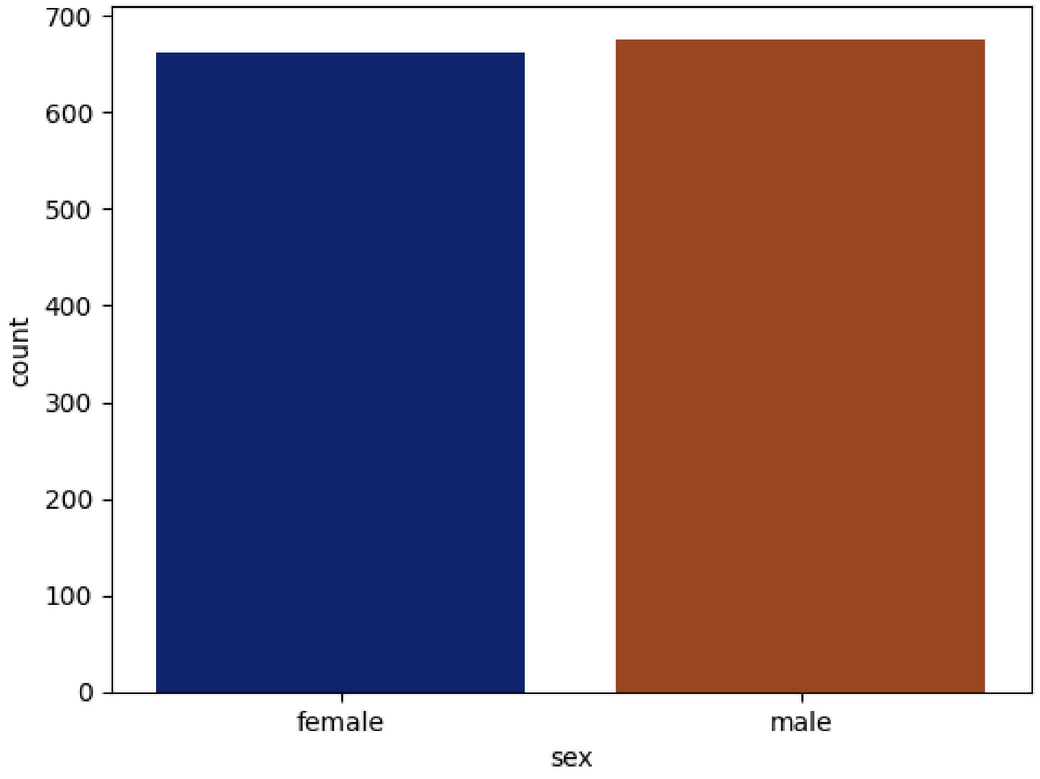
```
plt.figure(figsize=(10,5))
plt.hist(df['children'], edgecolor='white', label='d',color='c')
plt.xlabel("Number of Childern")
plt.ylabel("Count")
plt.title = ('Childern Distrubtion in the Dataset')
```



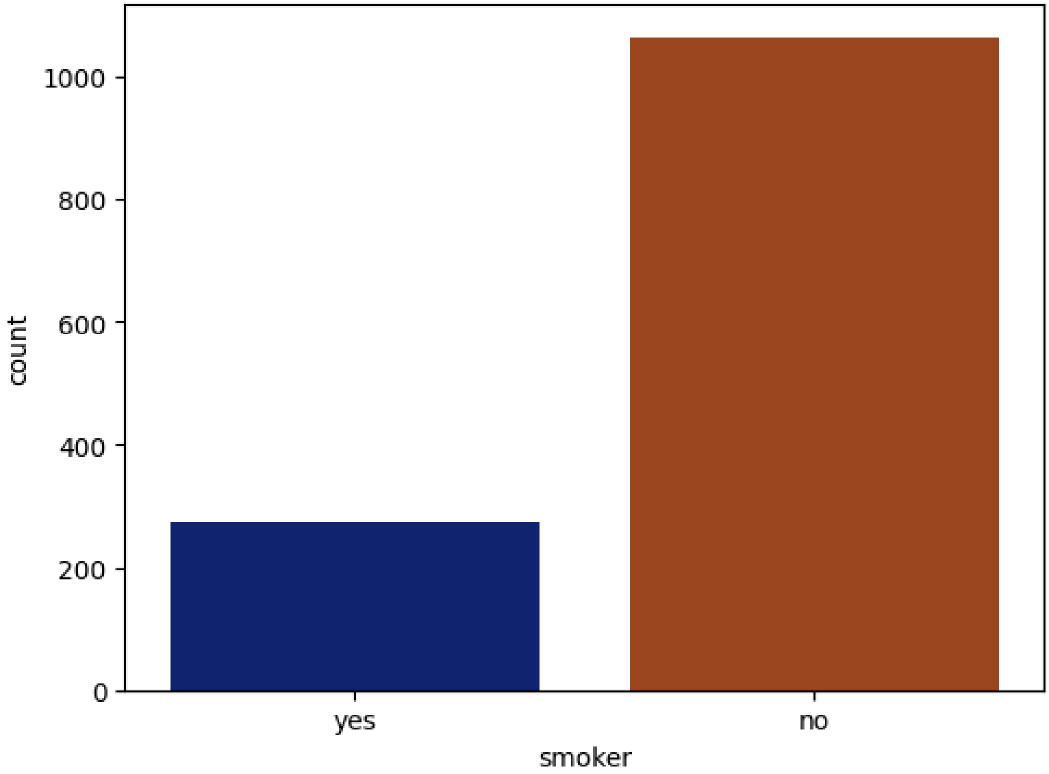
Categorical variable plots

In [32]:

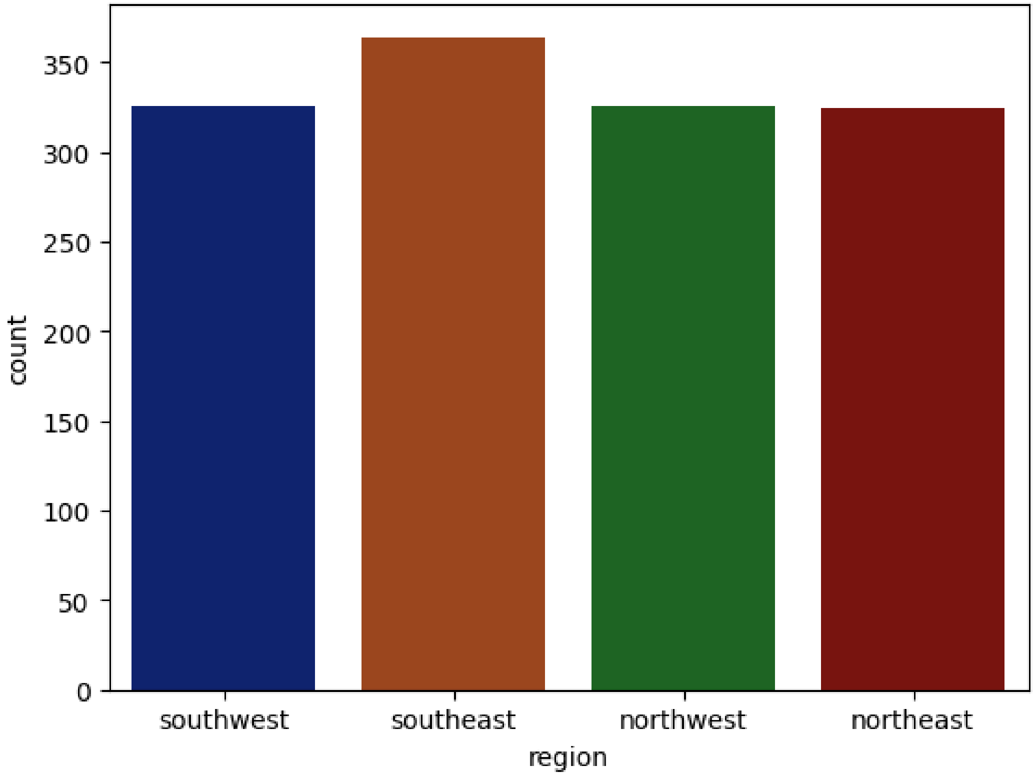
```
sex_plot =sns.countplot(x='sex',data=df,palette='dark')
```



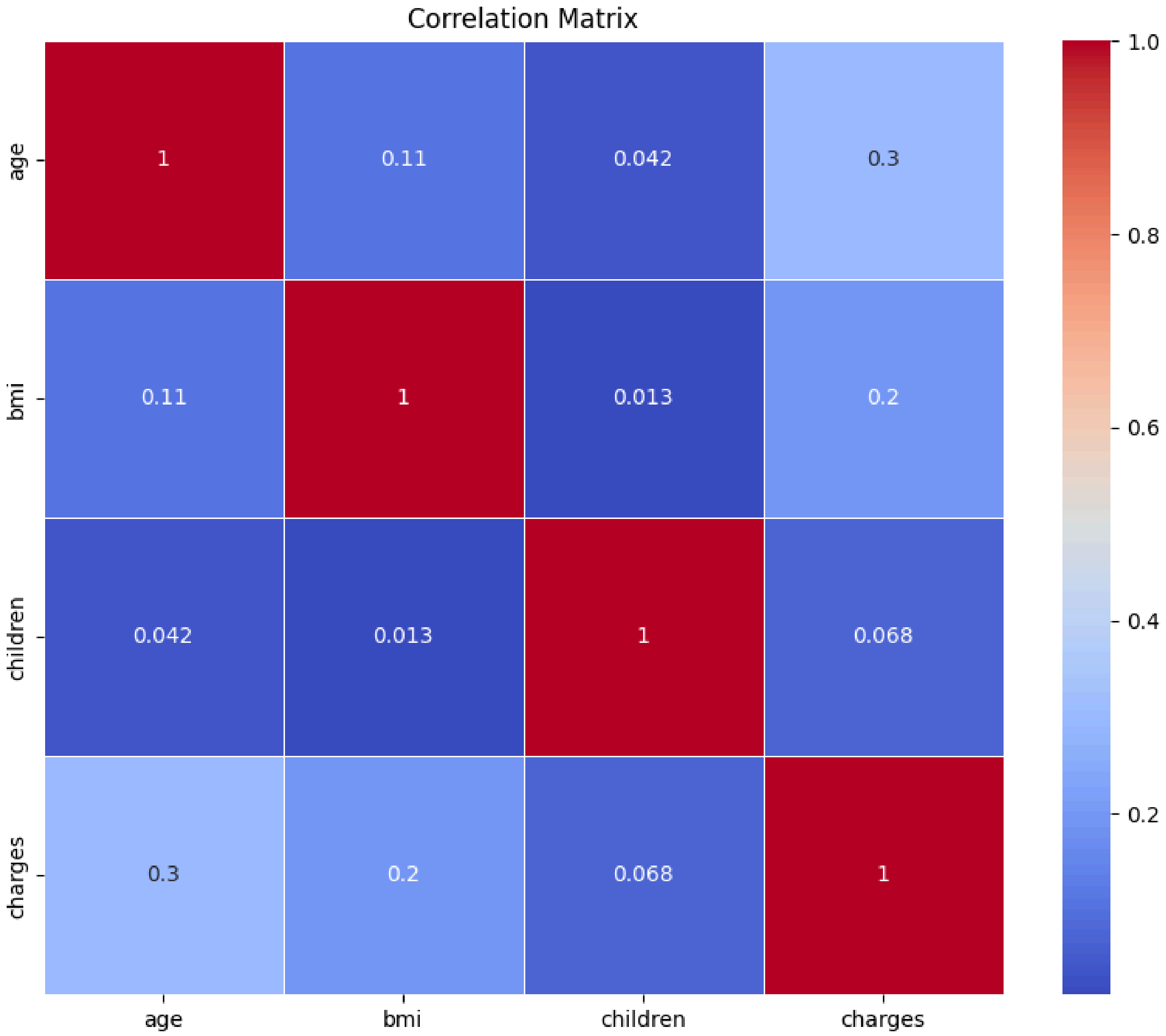
In [33]:
```
sex_plot =sns.countplot(x='smoker',data=df,palette='dark')
```



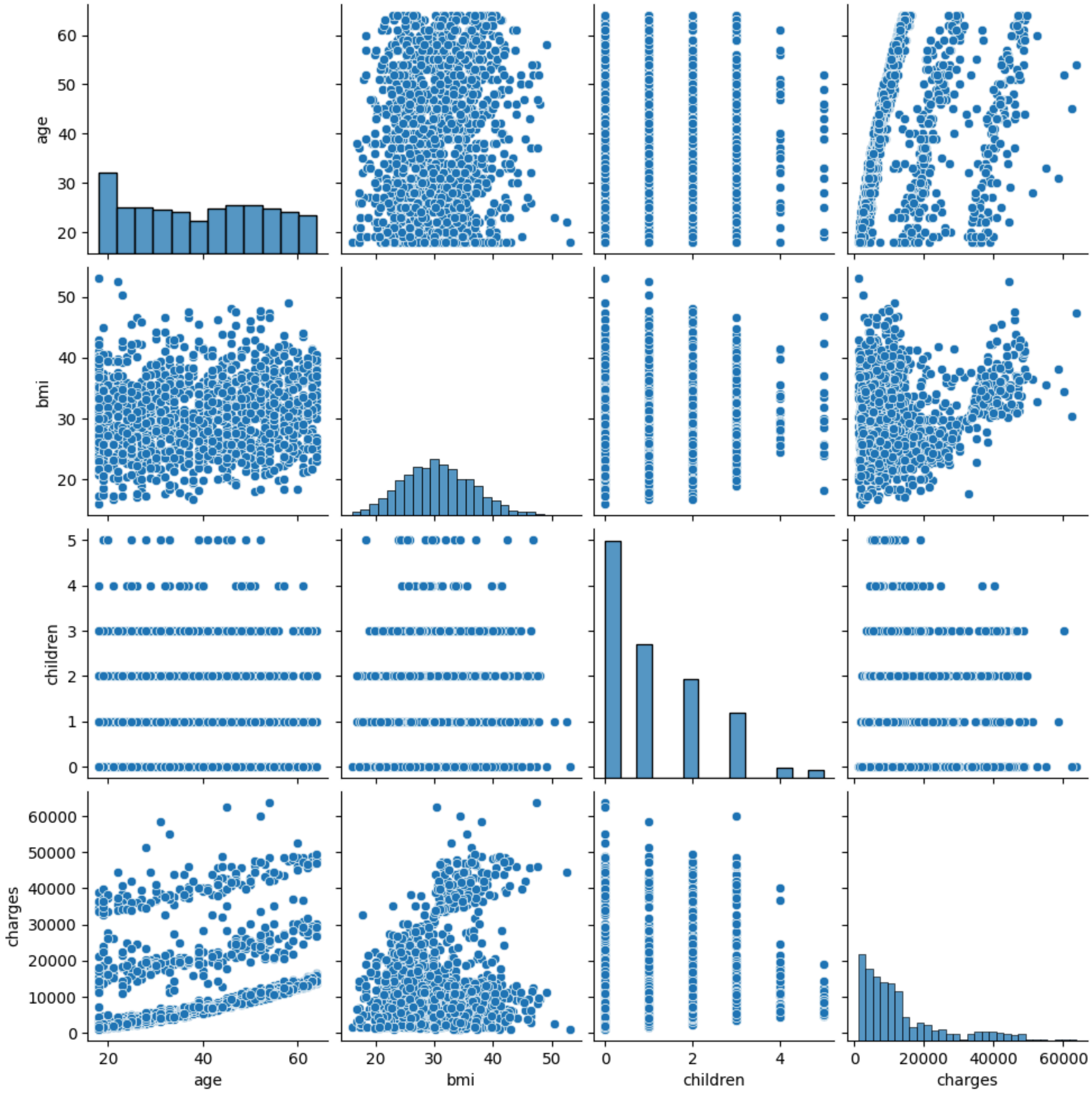
In [34]:
```
sex_plot =sns.countplot(x='region',data=df,palette='dark')
```



In [12]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(df[['age','bmi','children','charges']].corr(), annot=True, cmap='coolwarm',
plt.title('Correlation Matrix')
plt.show()
```



In [13]:
```
sns.pairplot(df)
plt.show()
```



In [20]:
```
# Encode categorical variables
df1 = pd.get_dummies(df, dtype=int)
```

In [21]:
```
df1
```

Out[21]:

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

1338 rows × 12 columns

In [25]:
```
# Define features and target
X = df1.drop('charges', axis=1)
y = df1['charges']
```

In [26]:
```
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42
```

In [27]:
```
model = LinearRegression()

# Train the model
model.fit(X_train, y_train)
```

Out[27]:
```
▾ LinearRegression
LinearRegression()
```

In [38]:
```
from sklearn.metrics import mean_absolute_error, r2_score

# Predict on the test set
y_pred = model.predict(X_test)

# Calculate metrics
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Absolute Error: {mae}')
print(f'R2 Score: {r2}')
```

```
Mean Absolute Error: 4181.194473753645
R2 Score: 0.7835929767120723
```

In [37]:

```
from xgboost import XGBRegressor
xgb = XGBRegressor(objective ='reg:squarederror', colsample_bytree = 0.3, learning_rate
                max_depth = 5, alpha = 10, n_estimators = 100)
xgb.fit(X_train, y_train)
```

Out[37]:

```
                                  XGBRegressor
XGBRegressor(alpha=10, base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.3, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.1, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=5, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
```

In [42]:

```
y_pred = xgb.predict(X_test)
```

In [45]:

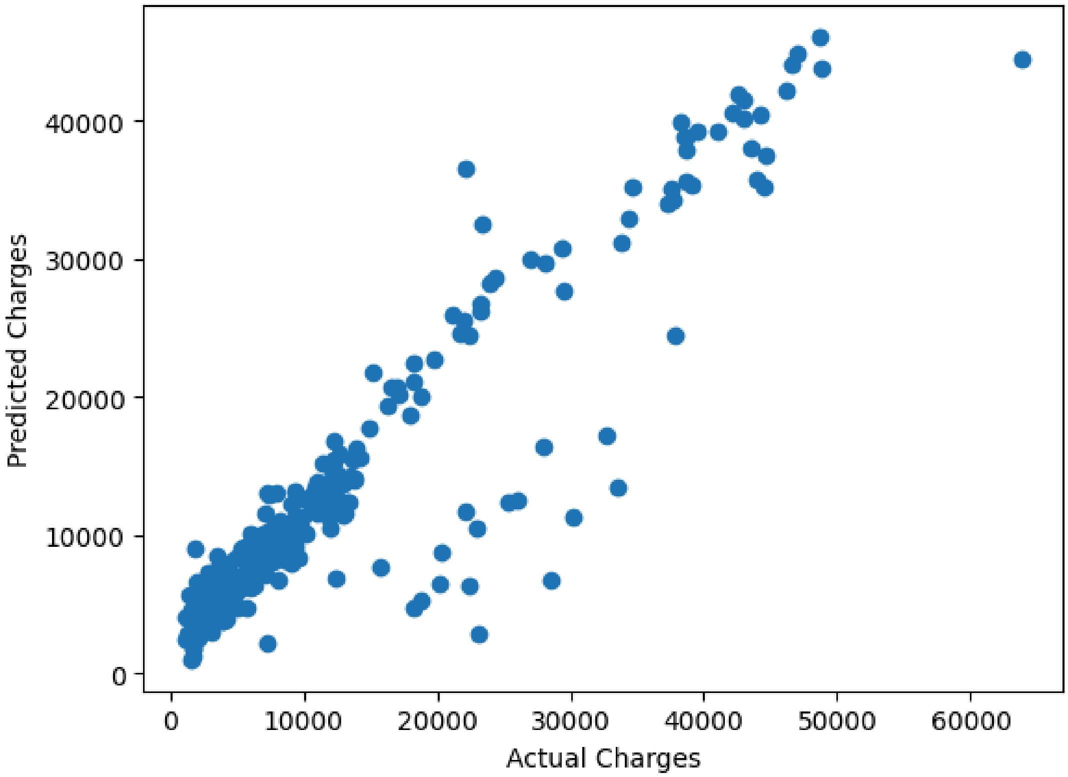
```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

```
Mean Squared Error: 22785810.99568766
R-squared: 0.8532302919531741
```

In [47]:

```
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Charges')
plt.ylabel('Predicted Charges')
plt.title('Actual vs Predicted Charges')
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[47], line 4
      2 plt.xlabel('Actual Charges')
      3 plt.ylabel('Predicted Charges')
----> 4 plt.title('Actual vs Predicted Charges')
      5 plt.show()

TypeError: 'str' object is not callable
```



Thankyou

In [ ]: