In [1]:
```
import pandas as pd
df = pd.read_csv("insurance.csv")
df.head()
```

Out[1]:

	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 [2]:
```
print("Shape:", df.shape)
print("\nInfo:")
print(df.info())
print("\nSummary:")
print(df.describe())
```

```
Shape: (1338, 7)

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

Summary:
                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 [3]:
```
print("Missing values:\n", df.isnull().sum())
```

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

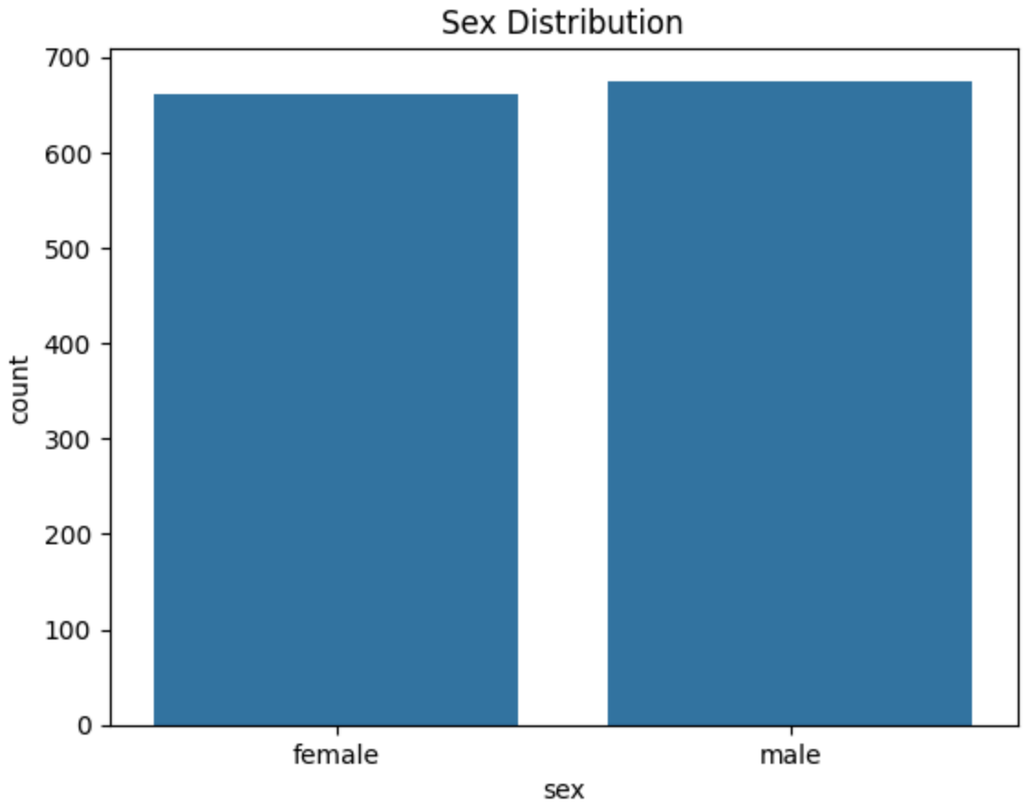
In [4]:
```
duplicates = df.duplicated().sum()
print("Number of duplicate rows:", duplicates)
```

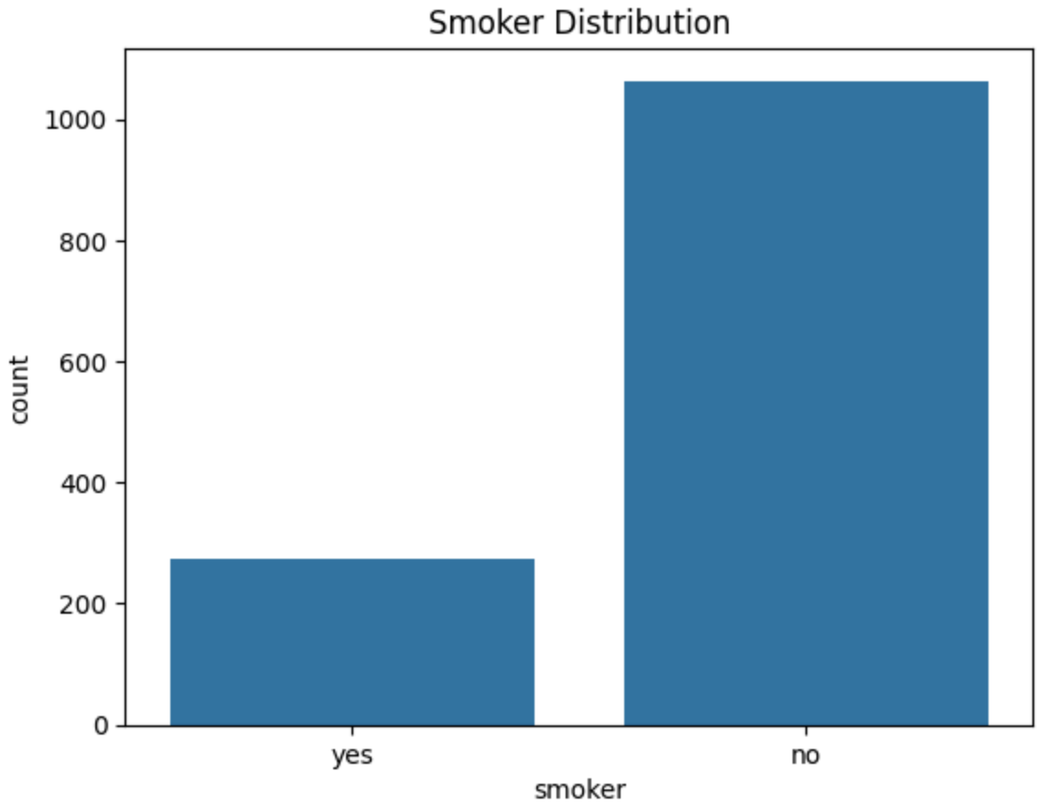
```
Number of duplicate rows: 1
```

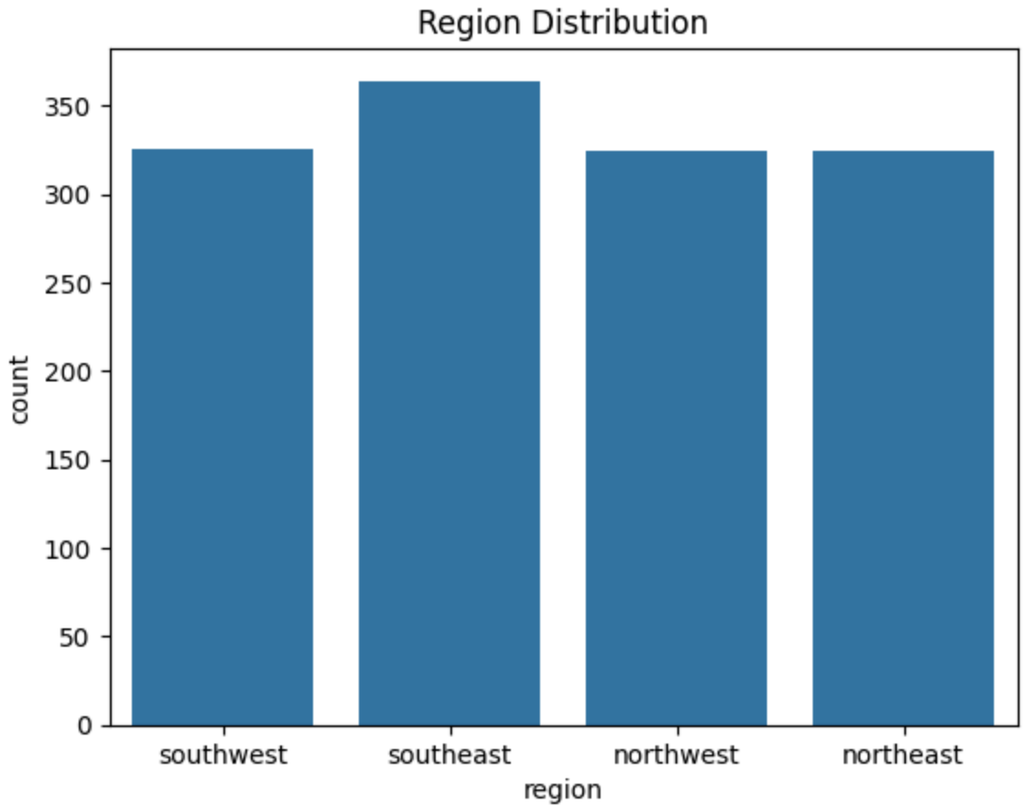
In [5]:
```
df = df.drop_duplicates()
print("Number of duplicate rows after removal:", df.duplicated().sum())
```

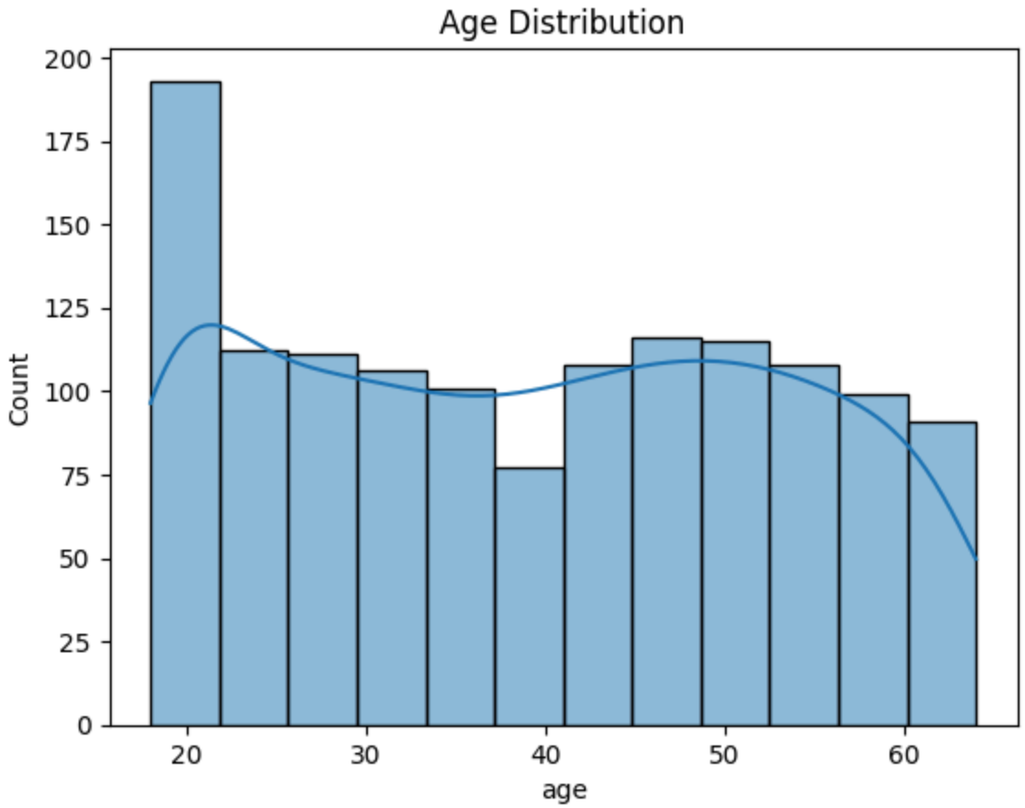
```
Number of duplicate rows after removal: 0
```

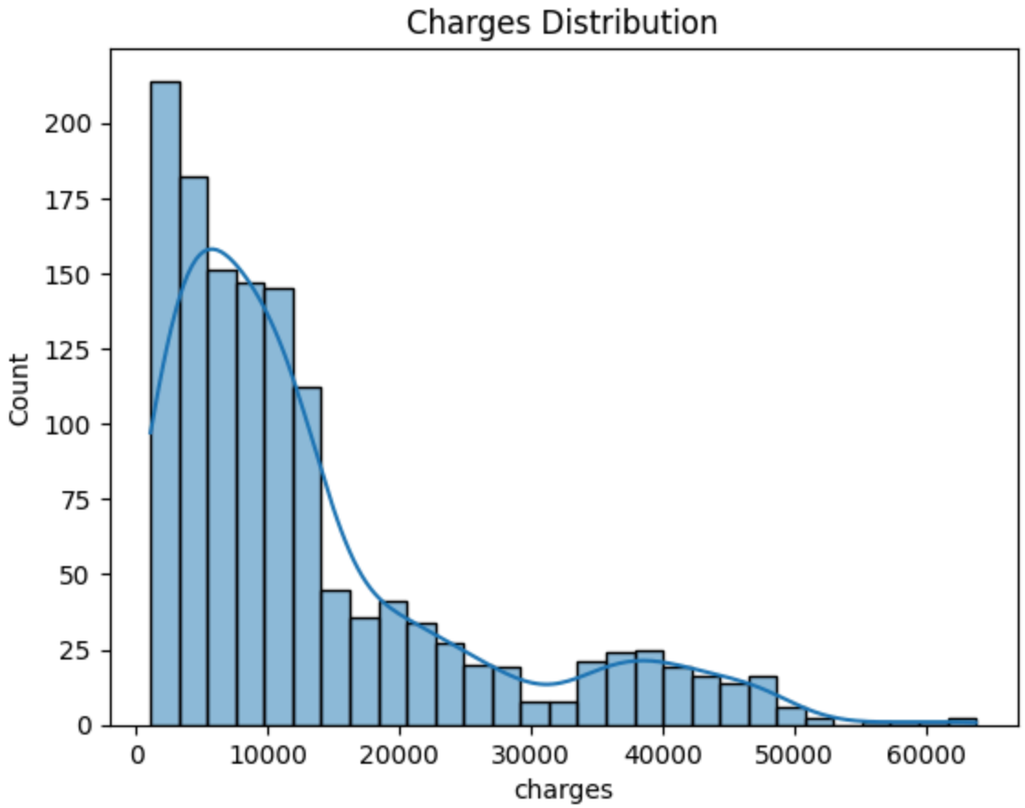
In [7]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Plot for categorical features
sns.countplot(x='sex', data=df)
plt.title('Sex Distribution')
plt.show()

sns.countplot(x='smoker', data=df)
plt.title('Smoker Distribution')
plt.show()

sns.countplot(x='region', data=df)
plt.title('Region Distribution')
plt.show()

# Plot for numerical features
sns.histplot(df['age'], kde=True)
plt.title('Age Distribution')
plt.show()

sns.histplot(df['charges'], kde=True)
plt.title('Charges Distribution')
plt.show()
```


Sex Distribution
700
600
500
400
300
200
100
0
count
female
male
sex

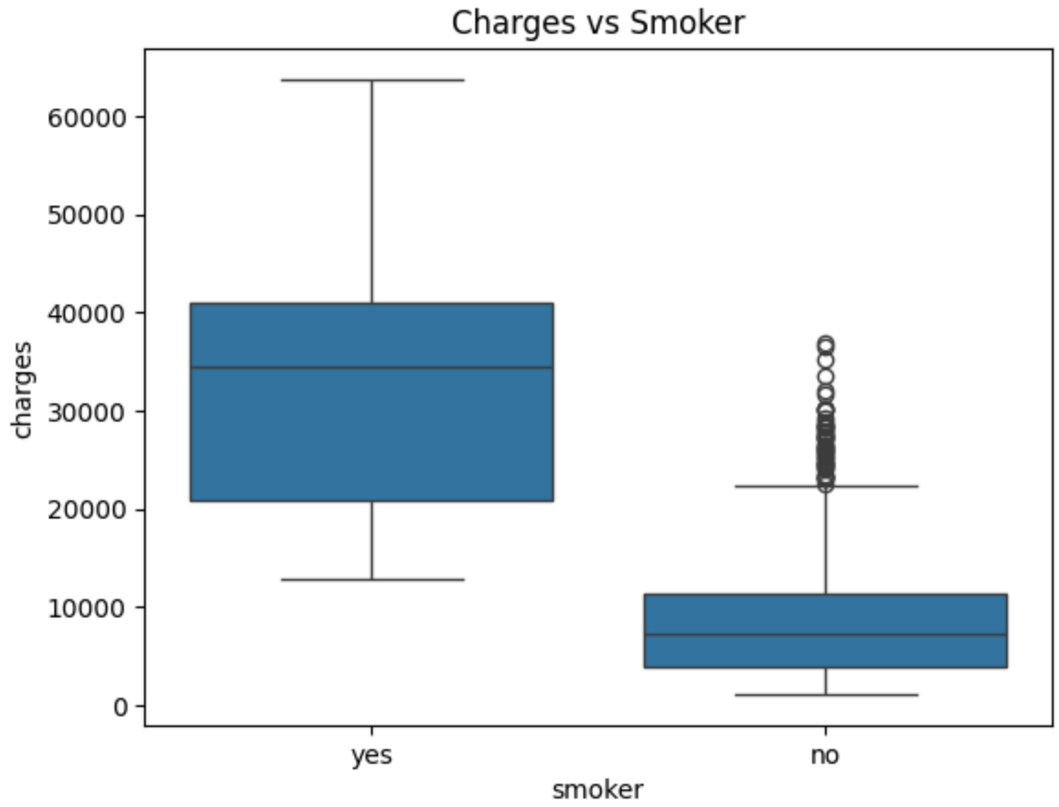

Smoker Distribution
1000
800
600
400
200
0
count
yes
no
smoker

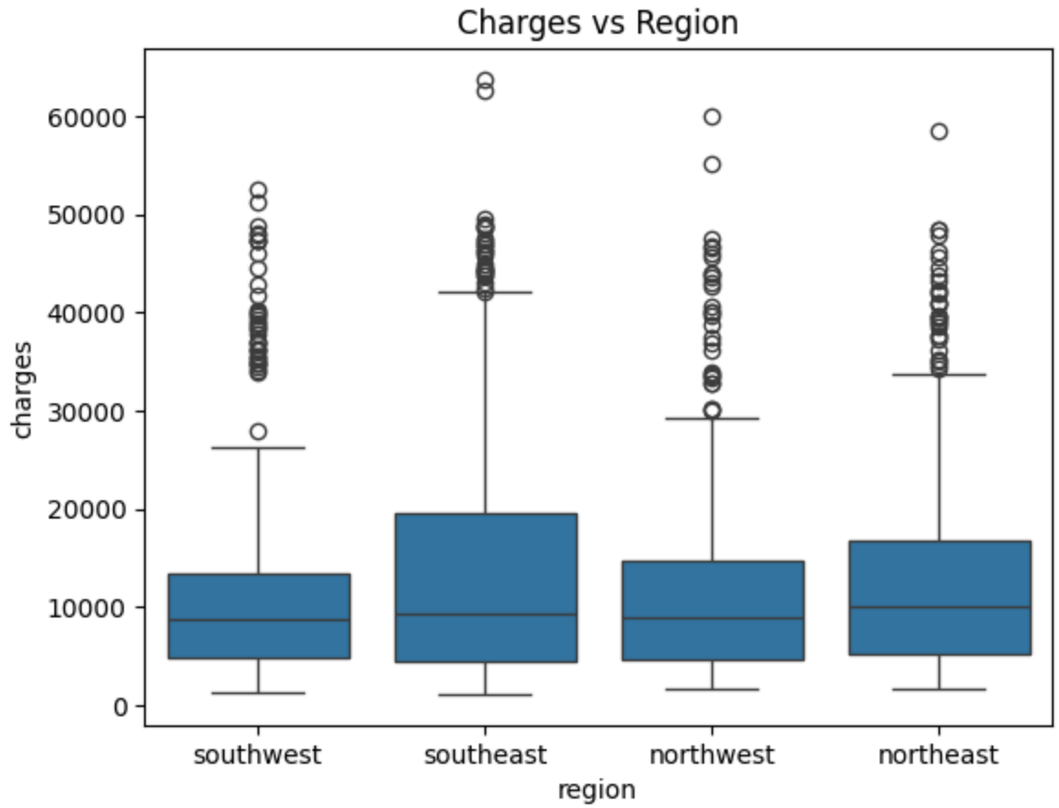

Region Distribution
count
0
50
100
150
200
250
300
350
southwest
southeast
northwest
northeast
region

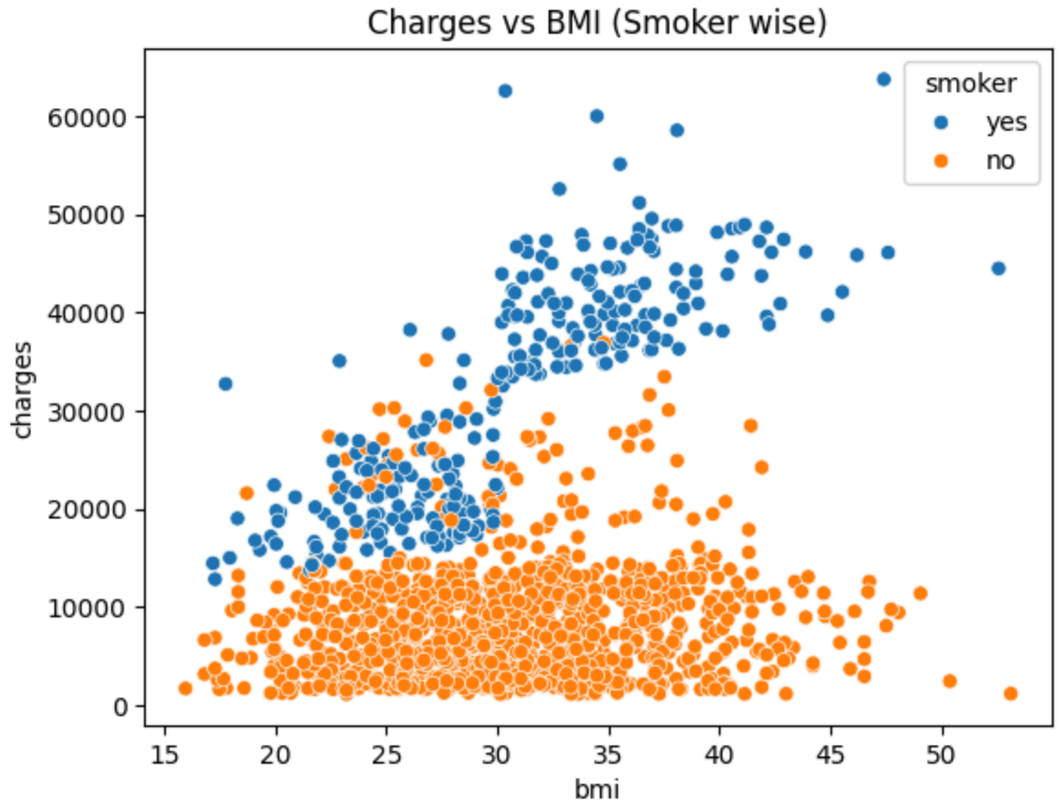

Age Distribution
Count
age
0
25
50
75
100
125
150
175
200
20
30
40
50
60



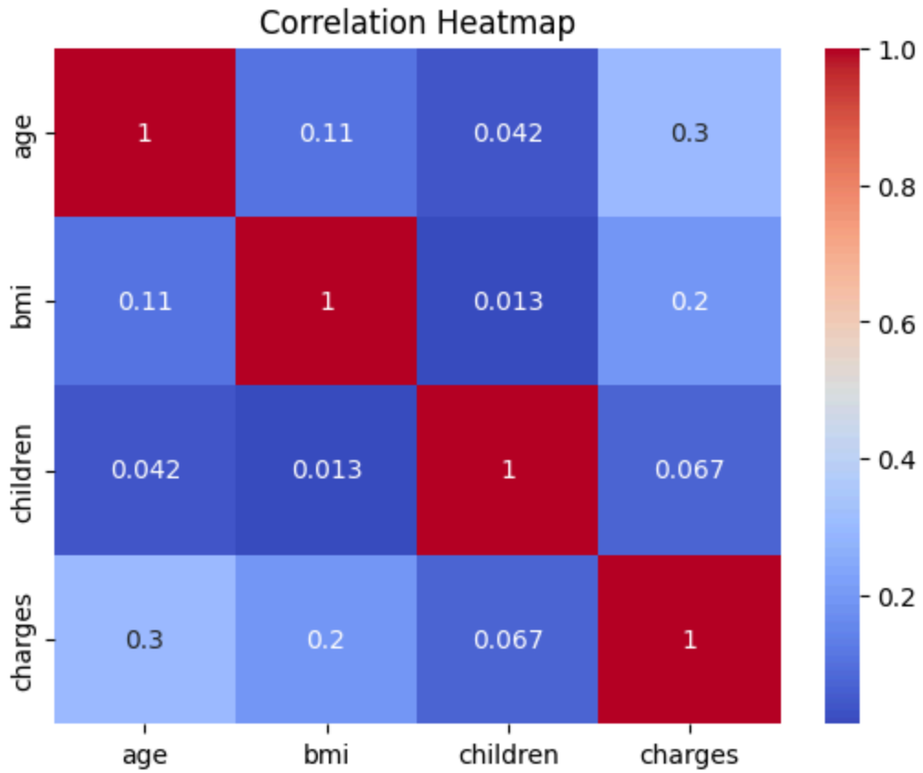
In [8]:
```
sns.boxplot(x='smoker', y='charges', data=df)
plt.title('Charges vs Smoker')
plt.show()

sns.boxplot(x='region', y='charges', data=df)
plt.title('Charges vs Region')
plt.show()

sns.scatterplot(x='bmi', y='charges', hue='smoker', data=df)
plt.title('Charges vs BMI (Smoker wise)')
plt.show()
```


Charges vs Smoker
charges
60000
50000
40000
30000
20000
10000
0
yes
no
smoker


Charges vs Region
charges
60000
50000
40000
30000
20000
10000
0
southwest
southeast
northwest
northeast
region



In [9]:
```
corr = df.corr(numeric_only=True)
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



In [10]:
```
df_encoded = pd.get_dummies(df, drop_first=True)
df_encoded.head()
```

Out[10]:

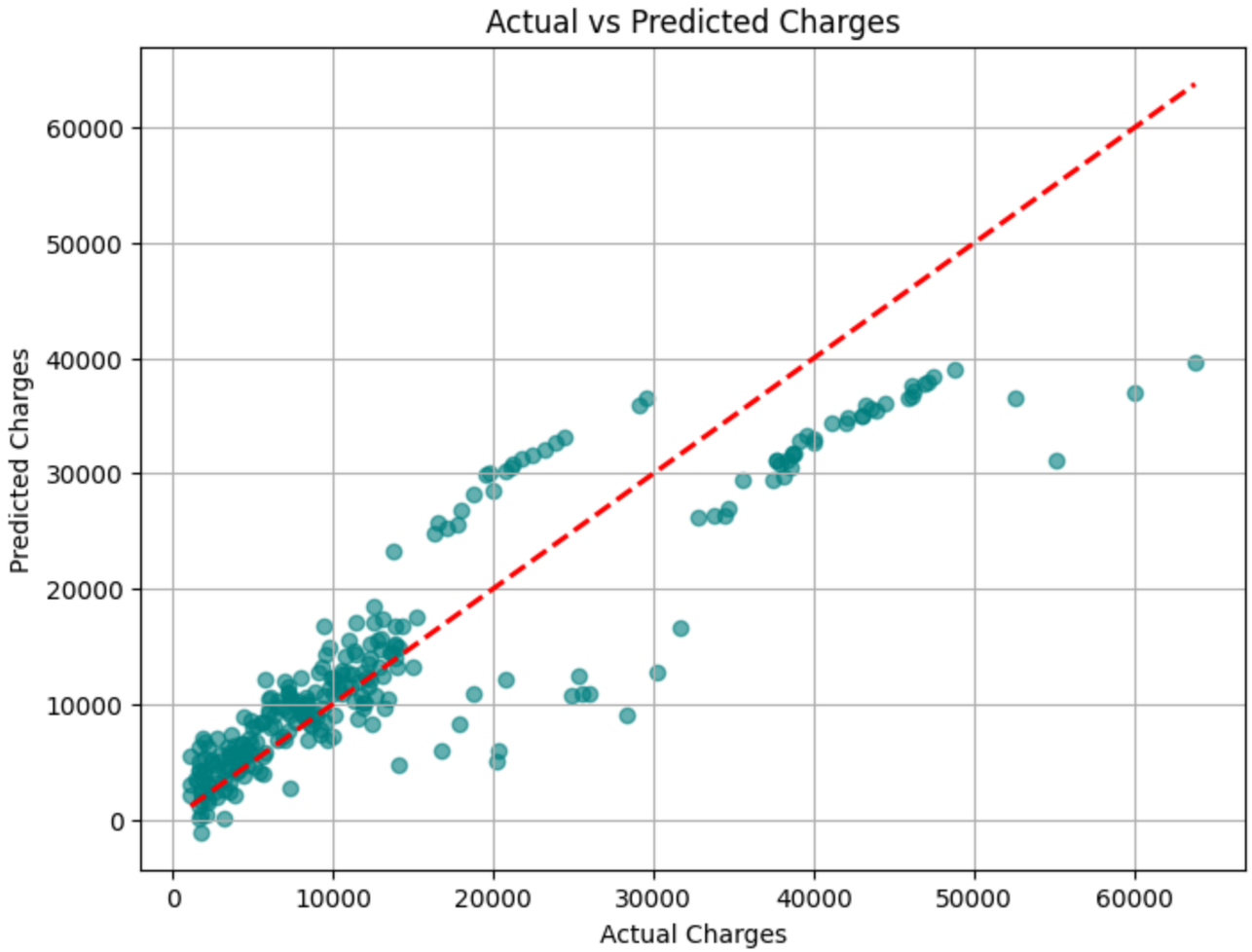
	age	bmi	children	charges	sex_male	smoker_yes	region_northwest	region_southeast	region_southwest
0	19	27.900	0	16884.92400	False	True	False	False	True
1	18	33.770	1	1725.55230	True	False	False	True	False
2	28	33.000	3	4449.46200	True	False	False	True	False
3	33	22.705	0	21984.47061	True	False	True	False	False
4	32	28.880	0	3866.85520	True	False	True	False	False

In [24]:
```
from sklearn.model_selection import train_test_split

X = df_encoded.drop('charges', axis=1)

y = df_encoded['charges']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [18]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

lr = LinearRegression()
lr.fit(X_train, y_train)

# Predict on test data
y_pred = lr.predict(X_test)

# Evaluate the model
print("R² Score:", r2_score(y_test, y_pred))
print("MSE:", mean_squared_error(y_test, y_pred))
```

```
R² Score: 0.8069287081198012
MSE: 35478020.67523559
```

In [21]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred, alpha=0.6, color='teal')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red', lw=2, linestyle='--')  # ideal line
plt.xlabel("Actual Charges")
plt.ylabel("Predicted Charges")
plt.title("Actual vs Predicted Charges")
plt.grid(True)
plt.show()
# What This Shows:
#Points on the red line = perfect predictions.

#Points above/below the red line = prediction error.
```


Actual vs Predicted Charges
Predicted Charges
Actual Charges
0
10000
20000
30000
40000
50000
60000

In [ ]: