In [17]:
```
# Step 1: Import Libraries and Load the Dataset
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

# Load the dataset
df = pd.read_csv('insurance.csv')
print("Dataset loaded successfully!")
```

```
Dataset loaded successfully!
```

In [18]:
```
# Step 2: Check the shape of the data along with the data types of the column
print("--- Dataset Overview ---")
print("\nNumber of rows and columns:", df.shape)
print("\nInformation about the dataset:")
df.info()
print("\nFirst 5 rows of the dataset:")
print(df.head())
```

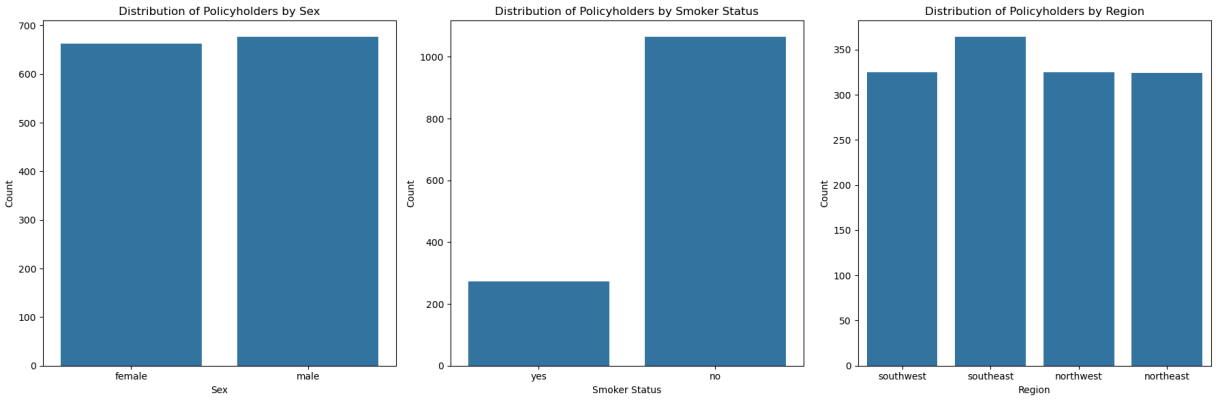
```
--- Dataset Overview ---

Number of rows and columns: (1338, 7)

Information about the dataset:
<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

First 5 rows of the dataset:
   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 [19]:
```
# Step 3: Check missing values in the dataset
print("--- Checking for Missing Values ---")
missing_values = df.isnull().sum()
missing_percentage = (df.isnull().sum() / len(df)) * 100
missing_df = pd.DataFrame({'Missing Count': missing_values, 'Percentage': missing_p
print(missing_df[missing_df['Missing Count'] > 0]) # Only print columns that have m
```

```
if missing_df['Missing Count'].sum() == 0:
    print("\nNo missing values found in the dataset. Data is clean!")
else:
    print("\nMissing values found. Review the report above to decide on imputation
    # If you find missing values, you would add imputation code here.
    # For example:
    # df['age'].fillna(df['age'].median(), inplace=True) # Fill 'age' with its medi
```

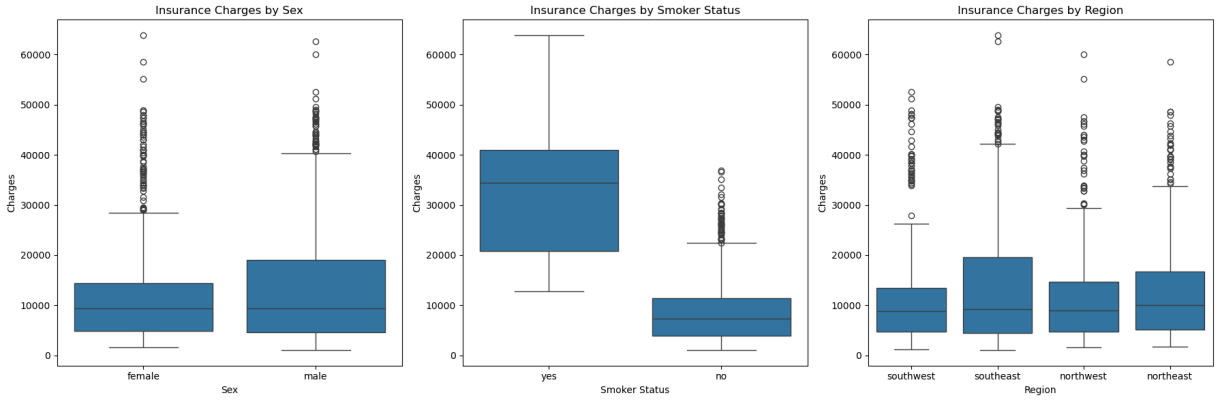
```
--- Checking for Missing Values ---
Empty DataFrame
Columns: [Missing Count, Percentage]
Index: []

No missing values found in the dataset. Data is clean!
```

In [20]:
```
# Step 4A: Count plots for categorical columns (Fixed by removing palette)
plt.figure(figsize=(18, 6))

plt.subplot(1, 3, 1) # 1 row, 3 columns, 1st plot
sns.countplot(x='sex', data=df) # Removed palette='viridis'
plt.title('Distribution of Policyholders by Sex')
plt.xlabel('Sex')
plt.ylabel('Count')

plt.subplot(1, 3, 2) # 1 row, 3 columns, 2nd plot
sns.countplot(x='smoker', data=df) # Removed palette='viridis'
plt.title('Distribution of Policyholders by Smoker Status')
plt.xlabel('Smoker Status')
plt.ylabel('Count')

plt.subplot(1, 3, 3) # 1 row, 3 columns, 3rd plot
sns.countplot(x='region', data=df) # Removed palette='viridis'
plt.title('Distribution of Policyholders by Region')
plt.xlabel('Region')
plt.ylabel('Count')

plt.tight_layout() # Adjust layout to prevent overlapping titles/labels
plt.show()
```

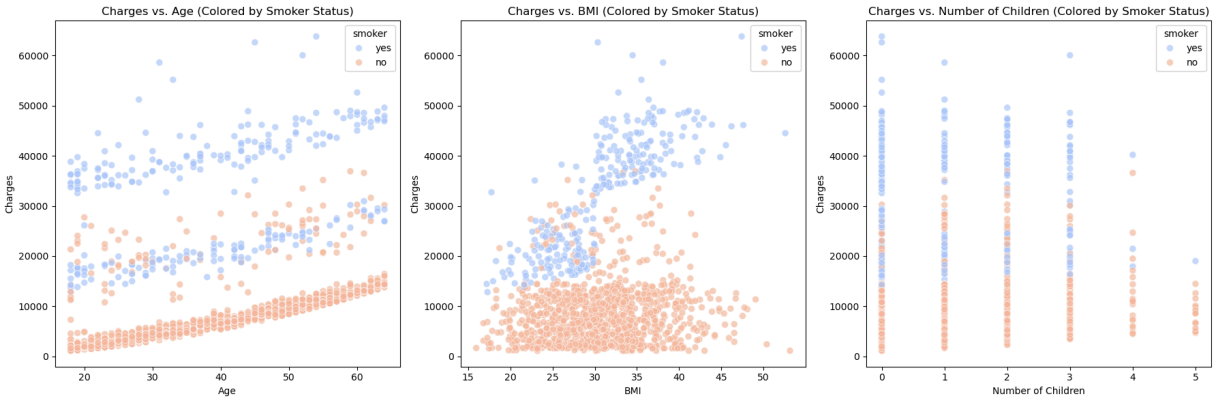


In [21]:
```
# Step 4B: Box plots of categorical columns vs. Charges (Fixed by removing palette)
plt.figure(figsize=(18, 6)) # Adjust figure size for better visualization

plt.subplot(1, 3, 1) # 1 row, 3 columns, 1st plot
sns.boxplot(x='sex', y='charges', data=df) # Removed palette='plasma'
```

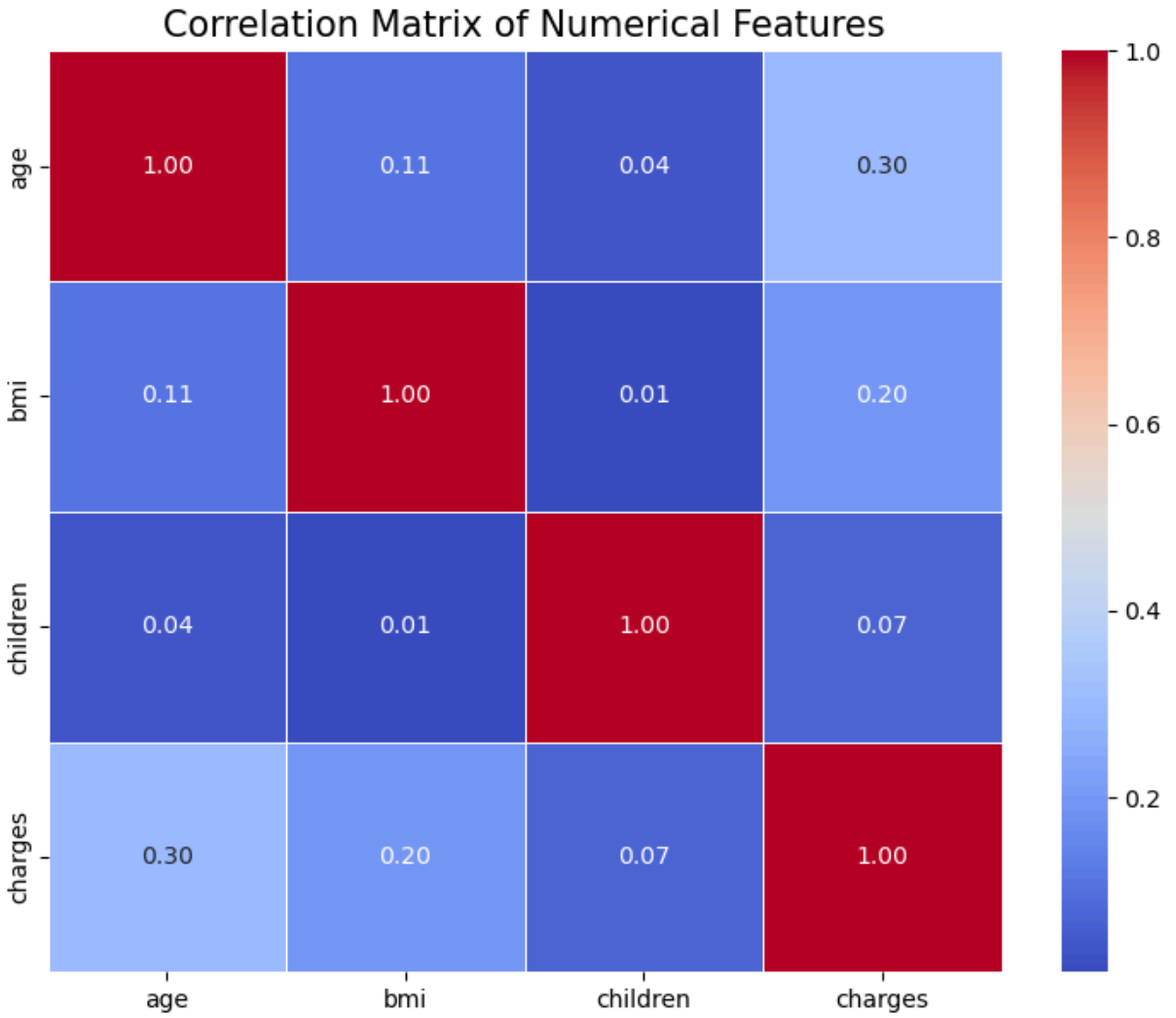
```
plt.title('Insurance Charges by Sex')
plt.xlabel('Sex')
plt.ylabel('Charges')

plt.subplot(1, 3, 2) # 1 row, 3 columns, 2nd plot
sns.boxplot(x='smoker', y='charges', data=df) # Removed palette='plasma'
plt.title('Insurance Charges by Smoker Status')
plt.xlabel('Smoker Status')
plt.ylabel('Charges')

plt.subplot(1, 3, 3) # 1 row, 3 columns, 3rd plot
sns.boxplot(x='region', y='charges', data=df) # Removed palette='plasma'
plt.title('Insurance Charges by Region')
plt.xlabel('Region')
plt.ylabel('Charges')

plt.tight_layout() # Adjust layout to prevent overlapping titles/labels
plt.show()
```



In [22]:
```
# Step 4C: Scatter plots of numerical columns vs. Charges (Fixed 'BMI' to 'bmi')
plt.figure(figsize=(18, 6)) # Adjust figure size for better visualization

plt.subplot(1, 3, 1) # 1 row, 3 columns, 1st plot
sns.scatterplot(x='age', y='charges', data=df, hue='smoker', palette='coolwarm', al
plt.title('Charges vs. Age (Colored by Smoker Status)')
plt.xlabel('Age')
plt.ylabel('Charges')

plt.subplot(1, 3, 2) # 1 row, 3 columns, 2nd plot
# --- FIX APPLIED HERE: Changed x='BMI' to x='bmi' ---
sns.scatterplot(x='bmi', y='charges', data=df, hue='smoker', palette='coolwarm', al
plt.title('Charges vs. BMI (Colored by Smoker Status)')
plt.xlabel('BMI') # Label can remain 'BMI' for readability in the plot
plt.ylabel('Charges')

plt.subplot(1, 3, 3) # 1 row, 3 columns, 3rd plot
sns.scatterplot(x='children', y='charges', data=df, hue='smoker', palette='coolwarm
plt.title('Charges vs. Number of Children (Colored by Smoker Status)')
plt.xlabel('Number of Children')
plt.ylabel('Charges')

plt.tight_layout() # Adjust layout to prevent overlapping titles/labels
plt.show()
```

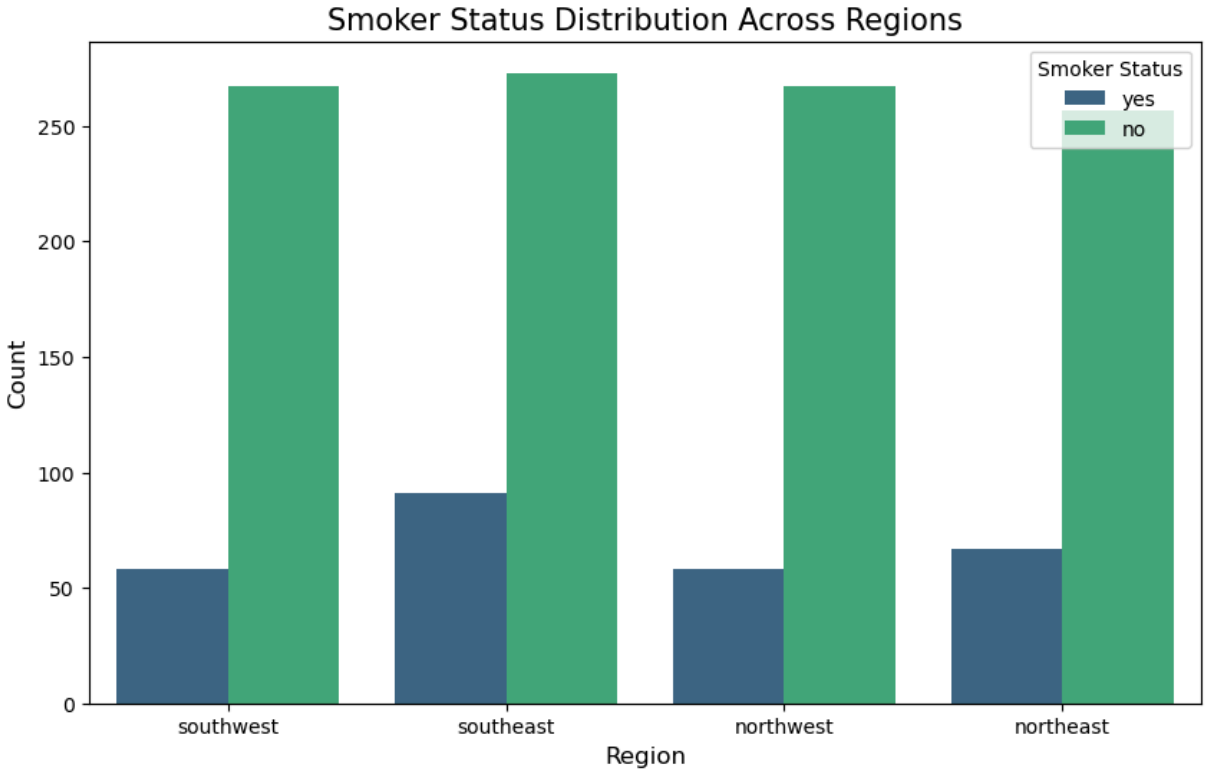


In [23]:
```
# Step 5A: Correlation Heatmap of Numerical Features
plt.figure(figsize=(9, 7)) # Adjust figure size for better visualization

# Select only numerical columns for correlation calculation
numerical_df = df.select_dtypes(include=np.number)
sns.heatmap(numerical_df.corr(), annot=True, cmap='coolwarm', fmt=".2f", linewidths
plt.title('Correlation Matrix of Numerical Features', fontsize=15)
plt.show()
```



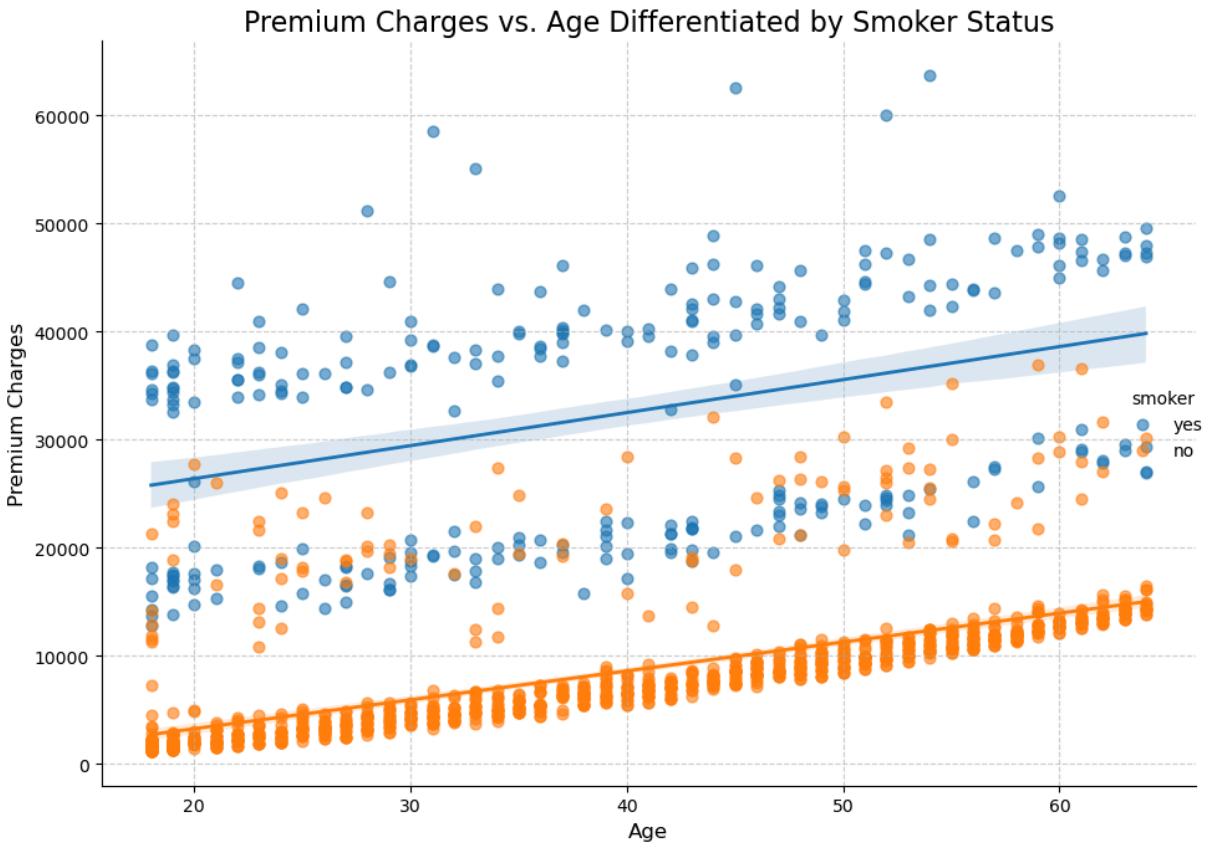
In [24]:
```
# Step 5B: Countplot of Smoker Status by Region
plt.figure(figsize=(10, 6)) # Adjust figure size for better visualization
```

```
sns.countplot(x='region', hue='smoker', data=df, palette='viridis') # Using 'viridi
plt.title('Smoker Status Distribution Across Regions', fontsize=15)
plt.xlabel('Region', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.legend(title='Smoker Status') # Adds a legend to explain the hue colors
plt.show()
```



In [25]:
```
# Step 6: Check if the number of premium charges for smokers or non-smokers is incr
# Using lmplot to visualize the relationship with separate regression lines for smo
plt.figure(figsize=(10, 7)) # Adjusting initial figure size, lmplot creates its own
sns.lmplot(x='age', y='charges', hue='smoker', data=df,
           height=7, aspect=1.3, ci=95,  # ci=95 shows 95% confidence interval
           scatter_kws={'alpha':0.6, 's': 40}, line_kws={'lw':2})
plt.title('Premium Charges vs. Age Differentiated by Smoker Status', fontsize=16)
plt.xlabel('Age', fontsize=12)
plt.ylabel('Premium Charges', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6) # Add a grid for better readability
plt.tight_layout() # Adjust layout
plt.show()
```

```
<Figure size 1000x700 with 0 Axes>
```


Premium Charges vs. Age Differentiated by Smoker Status
Premium Charges
Age
smoker
yes
no
0
10000
20000
30000
40000
50000
60000
20
30
40
50
60