jupyter Dec_practice Last Checkpoint: 11/27/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Markdown

Business Problem Understanding:

The business problem for the "Insurance Premium Prediction" project involves predicting the insurance premiums for individuals based on several factors such as age, gender, etc.The goal is to develop a model that can accurately estimate insurance expenses, which can aid in pricing policies and better understanding the factors influencing premium costs. This predictive model can assist insurance companies in making more informed decisions about premium rates and risk assessment.

Import Required liabraries:

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Import CSV File:

In [2]:
```
df=pd.read_csv("insurance_project1.csv")
```

Data Exploration:

In [3]:
```
df.head()
```

Out[3]:

	age	sex	bmi	children	smoker	region	expenses
0	19	female	27.9	0	yes	southwest	16884.92
1	18	male	33.8	1	no	southeast	1725.55
2	28	male	33.0	3	no	southeast	4449.46
3	33	male	22.7	0	no	northwest	21984.47
4	32	male	28.9	0	no	northwest	3866.86

jupyter Dec_practice Last Checkpoint: 11/27/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Markdown

In [4]:
```
df.tail()
```

Out[4]:

	age	sex	bmi	children	smoker	region	expenses
1333	50	male	31.0	3	no	northwest	10600.55
1334	18	female	31.9	0	no	northeast	2205.98
1335	18	female	36.9	0	no	southeast	1629.83
1336	21	female	25.8	0	no	southwest	2007.95
1337	61	female	29.1	0	yes	northwest	29141.36

In [5]:
```
df.shape
```

Out[5]:
```
(1338, 7)
```

In [6]:
```
df.describe()
```

Out[6]:

	age	bmi	children	expenses
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.665471	1.094918	13270.422414
std	14.049960	6.098382	1.205493	12110.011240
min	18.000000	16.000000	0.000000	1121.870000
25%	27.000000	26.300000	0.000000	4740.287500
50%	39.000000	30.400000	1.000000	9382.030000
75%	51.000000	34.700000	2.000000	16639.915000
max	64.000000	53.100000	5.000000	63770.430000

In [7]:
```
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
```

jupyter Dec_practice Last Checkpoint: 11/27/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown

In [7]:
```
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   expenses  1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
expenses    0
dtype: int64
```

In [9]:
```
df.dtypes
```

Out[9]:
```
age           int64
sex          object
bmi         float64
children      int64
smoker       object
region       object
expenses    float64
dtype: object
```

In [10]:
```
df.columns.tolist()
```

Out[10]:
```
['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'expenses']
```

jupyter Dec_practice Last Checkpoint: 11/27/2023 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

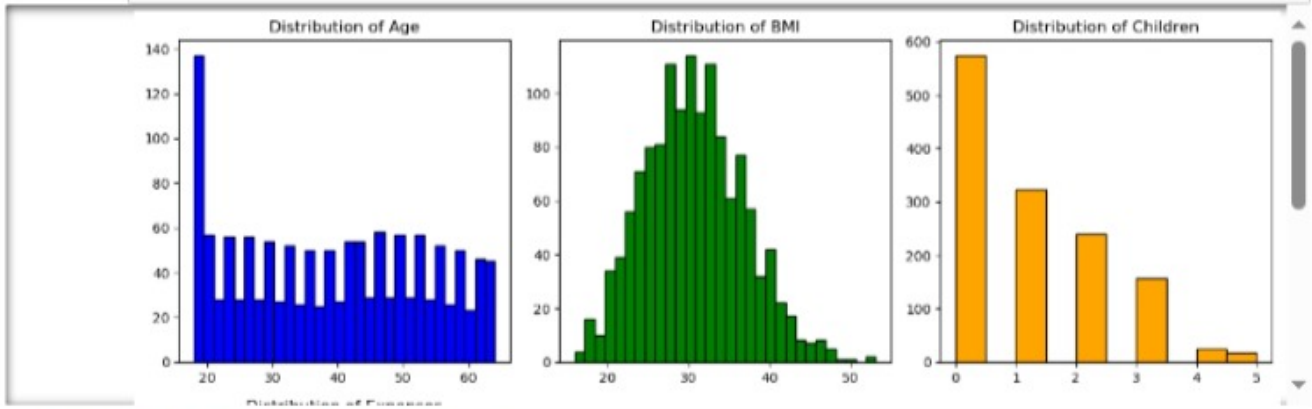
Not Trusted | Python 3 (ipykernel)

Run Markdown

Data Analysis:

- 1]EDA(Exploratory Data Analysis):

In [11]:
```
# Plot histograms for each numerical column
plt.figure(figsize=(12, 8))

plt.subplot(2, 3, 1)
plt.hist(df['age'], bins=30, color='blue', edgecolor='black')
plt.title('Distribution of Age')

plt.subplot(2, 3, 2)
plt.hist(df['bmi'], bins=30, color='green', edgecolor='black')
plt.title('Distribution of BMI')

plt.subplot(2, 3, 3)
plt.hist(df['children'], bins=10, color='orange', edgecolor='black')
plt.title('Distribution of Children')

plt.subplot(2, 3, 4)
plt.hist(df['expenses'], bins=30, color='red', edgecolor='black')
plt.title('Distribution of Expenses')

plt.tight_layout()
plt.show()
```



In [12]:
```
# Count the number of males and females
gender_counts = df['sex'].value_counts()

# Plot a bar chart
```

jupyter Dec_practice Last Checkpoint: 11/27/2023 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

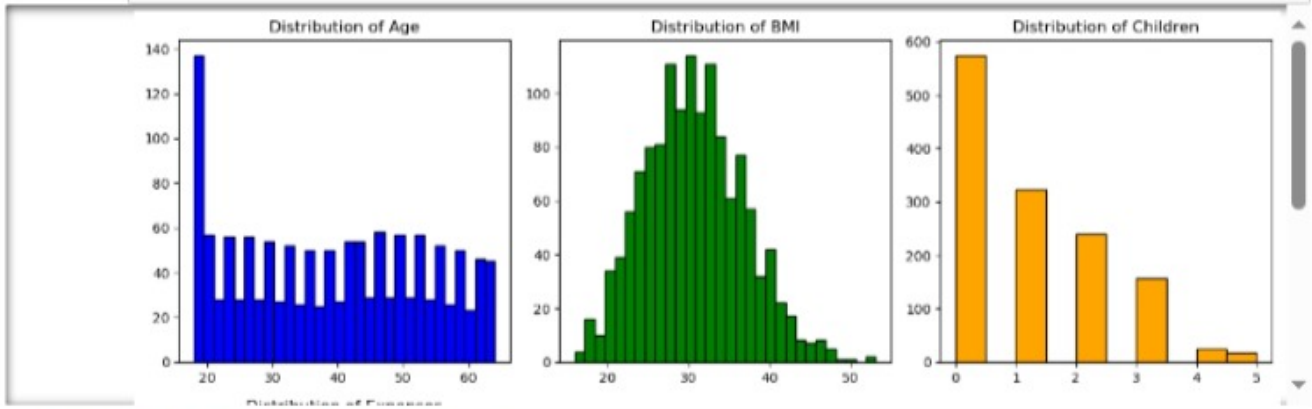
Not Trusted | Python 3 (ipykernel)

Run Markdown

Data Analysis:

- 1]EDA(Exploratory Data Analysis):

In [11]:
```
# Plot histograms for each numerical column
plt.figure(figsize=(12, 8))

plt.subplot(2, 3, 1)
plt.hist(df['age'], bins=30, color='blue', edgecolor='black')
plt.title('Distribution of Age')

plt.subplot(2, 3, 2)
plt.hist(df['bmi'], bins=30, color='green', edgecolor='black')
plt.title('Distribution of BMI')

plt.subplot(2, 3, 3)
plt.hist(df['children'], bins=10, color='orange', edgecolor='black')
plt.title('Distribution of Children')

plt.subplot(2, 3, 4)
plt.hist(df['expenses'], bins=30, color='red', edgecolor='black')
plt.title('Distribution of Expenses')

plt.tight_layout()
plt.show()
```



In [12]:
```
# Count the number of males and females
gender_counts = df['sex'].value_counts()

# Plot a bar chart
```

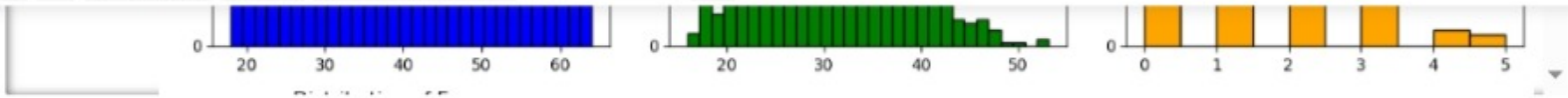
jupyter Dec_practice Last Checkpoint: 11/27/2023 (autosaved)

Logout

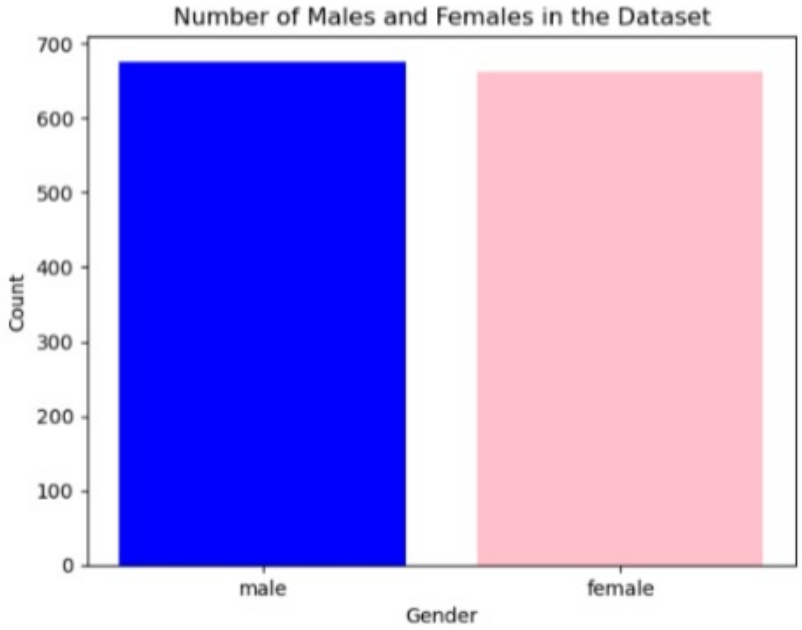
File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code



In [12]:
```
# Count the number of males and females
gender_counts = df['sex'].value_counts()

# Plot a bar chart
plt.bar(gender_counts.index, gender_counts, color=['blue', 'pink'])
plt.xlabel('Gender')
plt.ylabel('Count')
plt.title('Number of Males and Females in the Dataset')
plt.show()
```



jupyter Dec_practice Last Checkpoint: 11/27/2023 (autosaved)

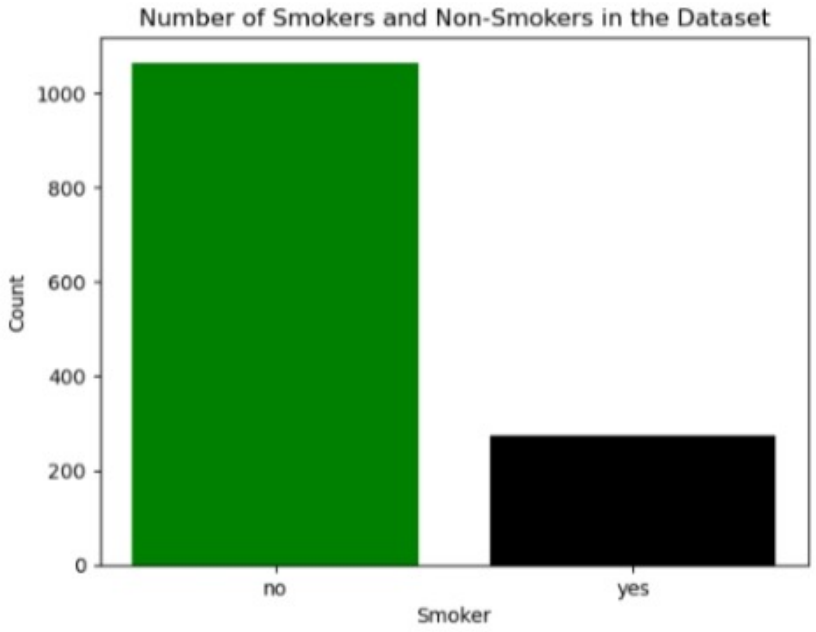
Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [13]:
```
# Count the number of smokers and non-smokers
smoker_counts = df['smoker'].value_counts()

# Plot a bar chart
plt.bar(smoker_counts.index, smoker_counts, color=['green', 'black'])
plt.xlabel('Smoker')
plt.ylabel('Count')
plt.title('Number of Smokers and Non-Smokers in the Dataset')
plt.show()
```



In [14]:
```
# Count the number of people in each region
region_counts = df['region'].value_counts()

# Plot a bar chart
```

jupyter Dec_practice Last Checkpoint: 11/27/2023 (autosaved)

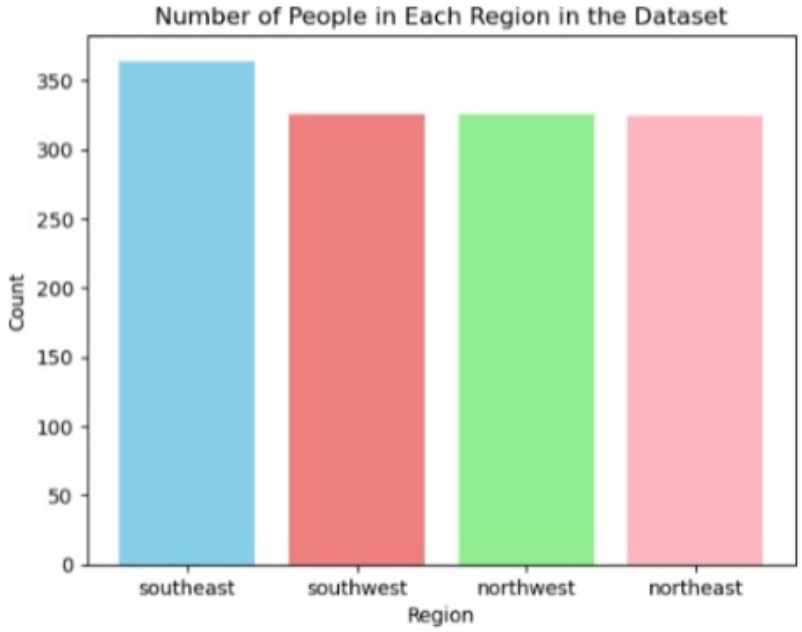
Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [14]:
```
# Count the number of people in each region
region_counts = df['region'].value_counts()

# Plot a bar chart
plt.bar(region_counts.index, region_counts, color=['skyblue', 'lightcoral', 'lightgreen', 'lightpink'])
plt.xlabel('Region')
plt.ylabel('Count')
plt.title('Number of People in Each Region in the Dataset')
plt.show()
```



In [15]:
```
# Plot pie chart for gender
gender_counts = df['sex'].value_counts()
plt.figure(figsize=(12, 5))
```

jupyter Dec_practice Last Checkpoint: 11/27/2023 (autosaved)

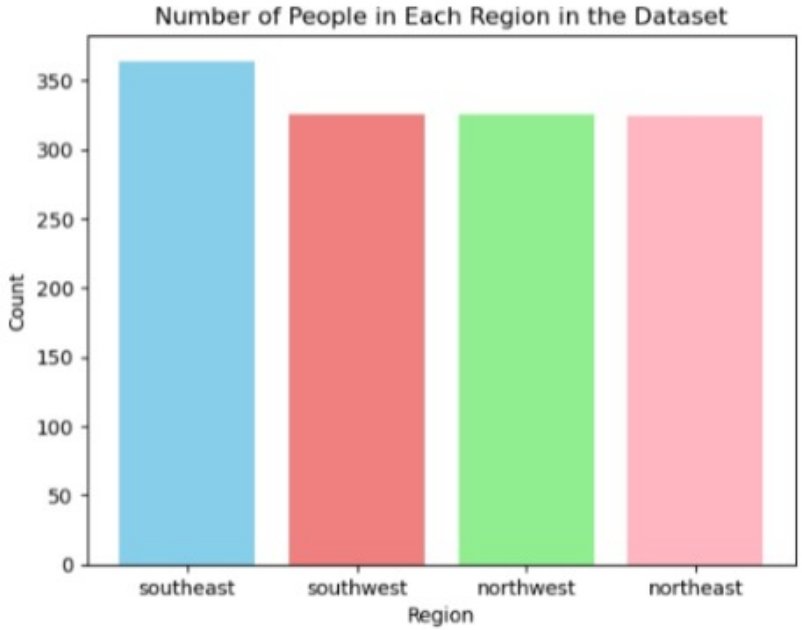
Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [14]:
```
# Count the number of people in each region
region_counts = df['region'].value_counts()

# Plot a bar chart
plt.bar(region_counts.index, region_counts, color=['skyblue', 'lightcoral', 'lightgreen', 'lightpink'])
plt.xlabel('Region')
plt.ylabel('Count')
plt.title('Number of People in Each Region in the Dataset')
plt.show()
```



In [15]:
```
# Plot pie chart for gender
gender_counts = df['sex'].value_counts()
plt.figure(figsize=(12, 5))
```

jupyter Dec_practice Last Checkpoint: 11/27/2023 (autosaved)

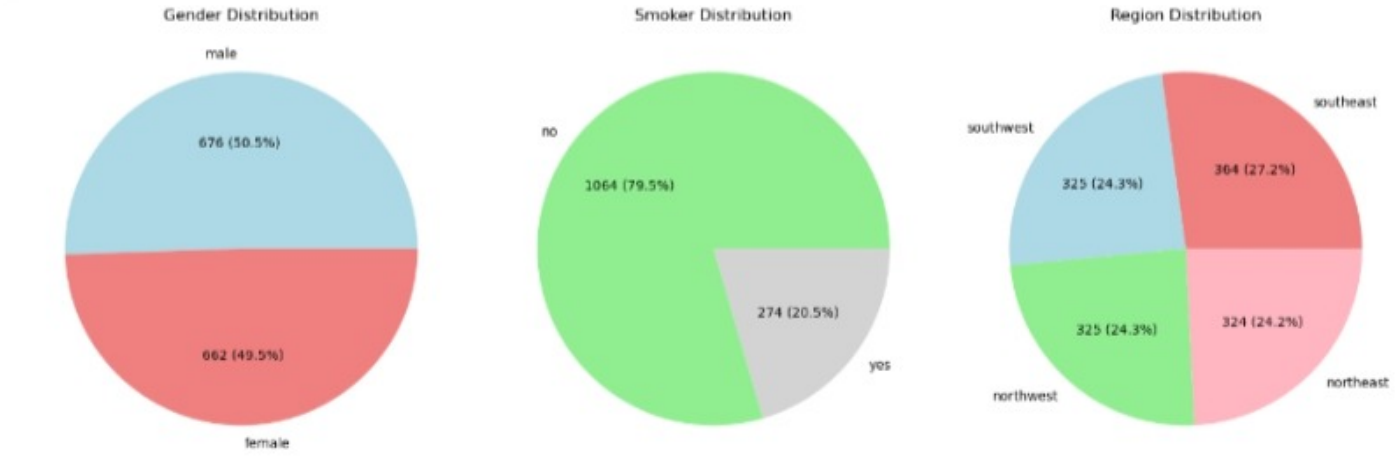
Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

```
plt.subplot(1, 3, 1)
plt.pie(gender_counts, labels=gender_counts.index, autopct=lambda p: '{:.0f} ({:.1f}%)'.format(p * sum(gender_counts) / 100, p),
plt.title('Gender Distribution')

# Plot pie chart for smokers
smoker_counts = df['smoker'].value_counts()

plt.subplot(1, 3, 2)
plt.pie(smoker_counts, labels=smoker_counts.index, autopct=lambda p: '{:.0f} ({:.1f}%)'.format(p * sum(smoker_counts) / 100, p),
plt.title('Smoker Distribution')

# Plot pie chart for region
region_counts = df['region'].value_counts()

plt.subplot(1, 3, 3)
plt.pie(region_counts, labels=region_counts.index, autopct=lambda p: '{:.0f} ({:.1f}%)'.format(p * sum(region_counts) / 100, p),
plt.title('Region Distribution')

plt.tight_layout()
plt.show()
```



jupyter Dec_practice Last Checkpoint: 11/27/2023 (autosaved)

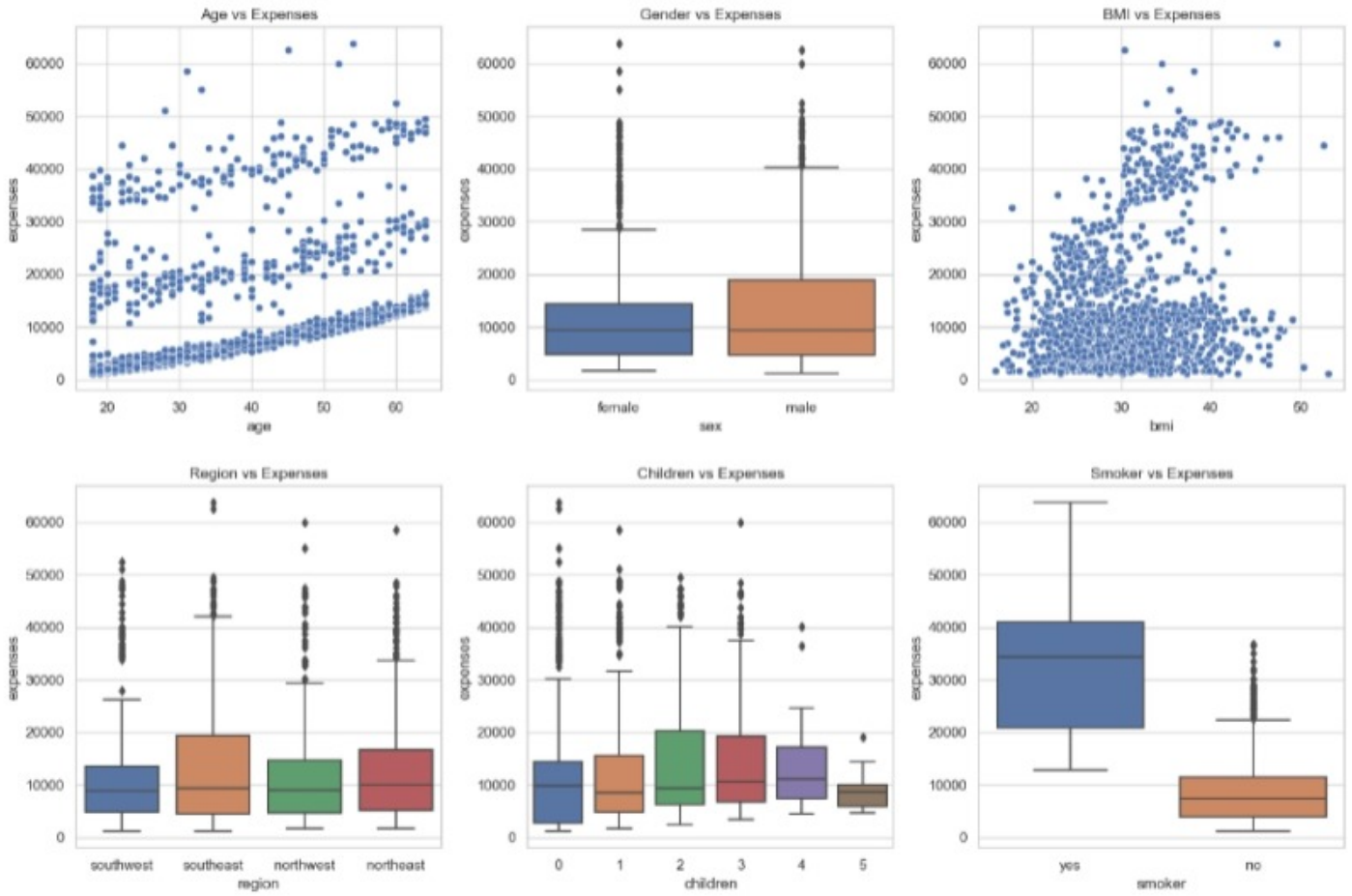
Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

```
plt.show()
```



In [18]:

jupyter Dec_practice Last Checkpoint: 11/27/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

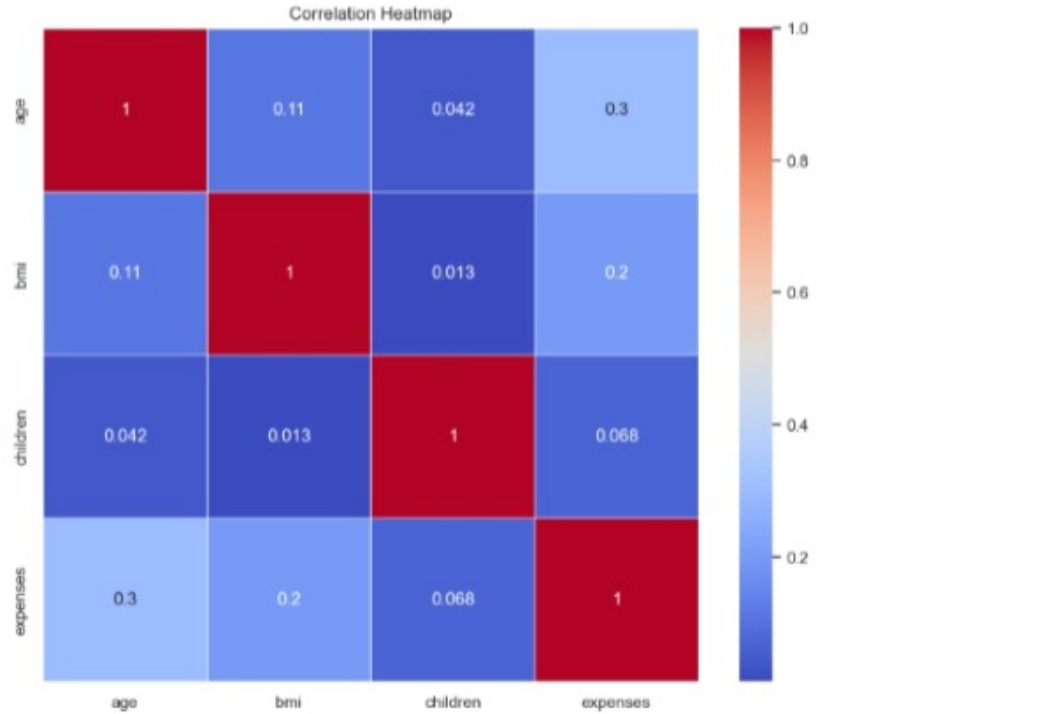
Run Code

region children smoker

In [18]:

```
# Calculate the correlation matrix
correlation_matrix = df.corr()

# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=.5)
plt.title('Correlation Heatmap')
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_4928\1011261691.py:2: FutureWarning: The default value of numeric_only in DataFram
e.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_
only to silence this warning.
  correlation_matrix = df.corr()
```



jupyter Dec_practice Last Checkpoint: 11/27/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

Data Preprocessing:

- 1]Datatype Handling:

In [19]:
```
df.dtypes
```

Out[19]:
```
age           int64
sex          object
bmi         float64
children      int64
smoker       object
region       object
expenses    float64
dtype: object
```

In [20]:
```
# One-hot encode 'sex', 'smoker', and 'region'
df = pd.get_dummies(df, columns=['sex', 'smoker', 'region'], drop_first=True)
```

In [21]:
```
df.head()
```

Out[21]:

	age	bmi	children	expenses	sex_male	smoker_yes	region_northwest	region_southeast	region_southwest
0	19	27.9	0	16884.92	0	1	0	0	1
1	18	33.8	1	1725.55	1	0	0	1	0
2	28	33.0	3	4449.46	1	0	0	1	0
3	33	22.7	0	21984.47	1	0	1	0	0
4	32	28.9	0	3866.86	1	0	1	0	0

In [22]:
```
# Extract the 'expenses' column
expenses_column = df['expenses']

# Drop the 'expenses' column from its current position
df = df.drop(columns=['expenses'])

# Add the 'expenses' column to the last position
df['expenses'] = expenses_column
```

In [23]:
```
df.head()
```

Out[23]:

age	bmi	children	sex_male	smoker_yes	region_northwest	region_southeast	region_southwest	expenses

jupyter Dec_practice Last Checkpoint: 11/27/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [22]:
```
# Extract the 'expenses' column
expenses_column = df['expenses']

# Drop the 'expenses' column from its current position
df = df.drop(columns=['expenses'])

# Add the 'expenses' column to the last position
df['expenses'] = expenses_column
```

In [23]:
```
df.head()
```

Out[23]:

	age	bmi	children	sex_male	smoker_yes	region_northwest	region_southeast	region_southwest	expenses
0	19	27.9	0	0	1	0	0	1	16884.92
1	18	33.8	1	1	0	0	1	0	1725.55
2	28	33.0	3	1	0	0	1	0	4449.46
3	33	22.7	0	1	0	1	0	0	21984.47
4	32	28.9	0	1	0	1	0	0	3866.86

- 2] Outlier Handling:

In [24]:
```
from scipy import stats
# Visualize box plots for all numerical columns
plt.figure(figsize=(15, 5))

plt.subplot(1, 4, 1)
sns.boxplot(x=df['age'])
plt.title('Age')

plt.subplot(1, 4, 2)
sns.boxplot(x=df['bmi'])
plt.title('BMI')

plt.subplot(1, 4, 3)
sns.boxplot(x=df['children'])
plt.title('Children')

plt.subplot(1, 4, 4)
sns.boxplot(x=df['expenses'])
plt.title('Expenses')

plt.show()
```

jupyter Dec_practice Last Checkpoint: 11/27/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

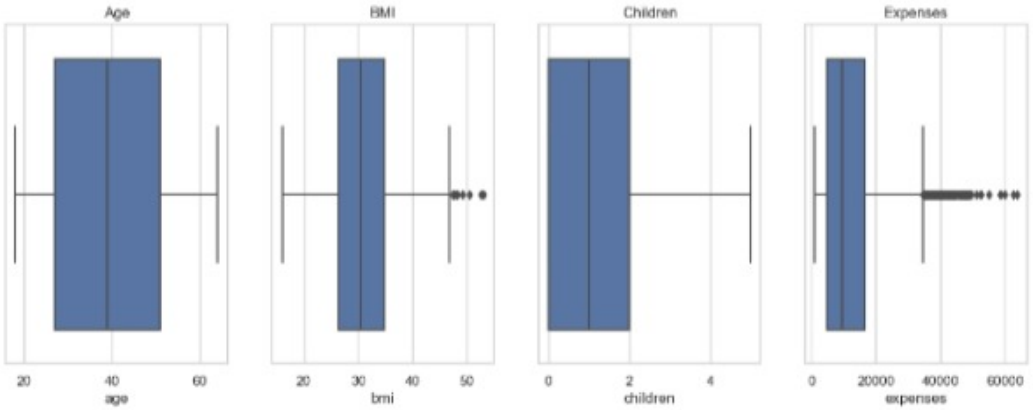
Not Trusted | Python 3 (ipykernel)

Run Code

- 2] Outlier Handling:

In [24]:

```
from scipy import stats
# Visualize box plots for all numerical columns
plt.figure(figsize=(15, 5))

plt.subplot(1, 4, 1)
sns.boxplot(x=df['age'])
plt.title('Age')

plt.subplot(1, 4, 2)
sns.boxplot(x=df['bmi'])
plt.title('BMI')

plt.subplot(1, 4, 3)
sns.boxplot(x=df['children'])
plt.title('Children')

plt.subplot(1, 4, 4)
sns.boxplot(x=df['expenses'])
plt.title('Expenses')

plt.show()

# Calculate z-scores for all numerical columns
numerical_columns = ['age', 'bmi', 'children', 'expenses']
z_scores = stats.zscore(df[numerical_columns])

# Define a threshold for z-scores
threshold = 3

# Identify and remove outliers based on z-scores
df_no_outliers = df[(z_scores < threshold).all(axis=1)]
```



jupyter Dec_practice Last Checkpoint: 11/27/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

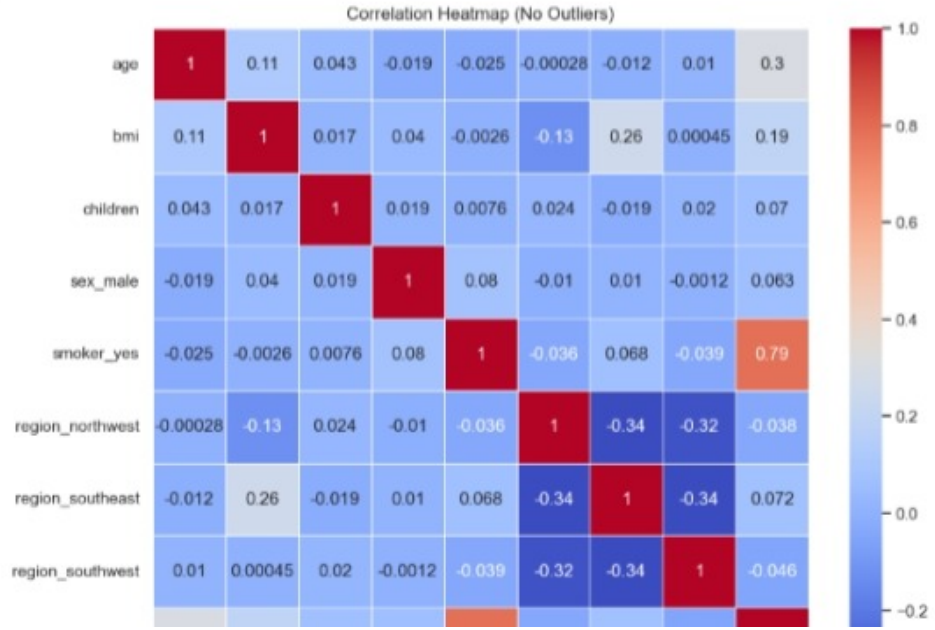
Run Code

In [25]:
```
# Calculate the IQR for numerical columns
Q1 = df[['age', 'bmi', 'children']].quantile(0.25)
Q3 = df[['age', 'bmi', 'children']].quantile(0.75)
IQR = Q3 - Q1

# Identify and remove outliers based on IQR
df_no_outliers = df[~((df[['age', 'bmi', 'children']] < (Q1 - 1.5 * IQR)) | (df[['age', 'bmi', 'children']] > (Q3 + 1.5 * IQR)))]
```

In [26]:
```
df_no_outliers.shape
```

Out[26]: (1329, 9)

In [27]:
```
# Calculate the correlation matrix for df_no_outliers
correlation_matrix_no_outliers = df_no_outliers.corr()

# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix_no_outliers, annot=True, cmap='coolwarm', linewidths=.5)
plt.title('Correlation Heatmap (No Outliers)')
plt.show()
```



jupyter Dec_practice Last Checkpoint: 11/27/2023 (autosaved)

Logout

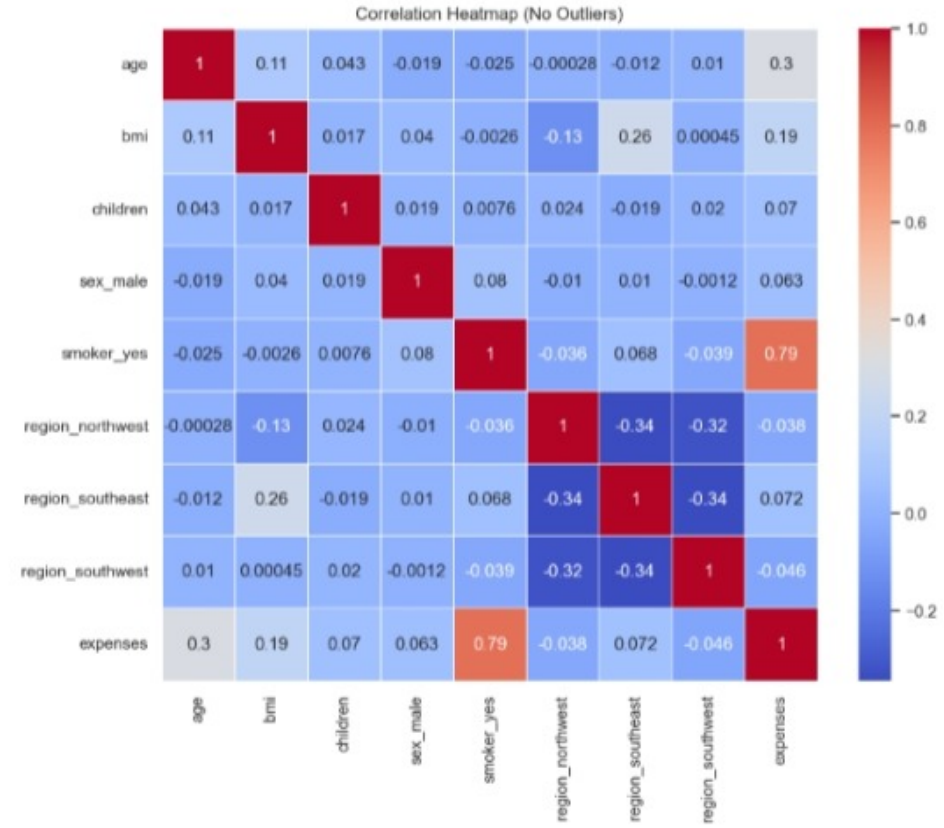
File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

Out[26]: (1329, 9)

In [27]:
```
# Calculate the correlation matrix for df_no_outliers
correlation_matrix_no_outliers = df_no_outliers.corr()

# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix_no_outliers, annot=True, cmap='coolwarm', linewidths=.5)
plt.title('Correlation Heatmap (No Outliers)')
plt.show()
```



Data Modelling:

- Train Test Split:

In [28]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score


# Assuming df_no_outliers is the DataFrame without outliers
X = df_no_outliers.drop('expenses', axis=1)
y = df_no_outliers['expenses']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [29]:
```
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(1063, 8) (266, 8) (1063,) (266,)
```

- 1] Linear Regression Model:

In [30]:
```
# Initialize the linear regression model
model = LinearRegression()

# Train the model on the training set
model.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = model.predict(X_test)

# Evaluate the model performance
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: 34498135.255386695
R-squared: 0.7671159829204874
```

In [31]:
```
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 34498135.255386695
```

In [32]:
```
r2 = r2_score(y_test, y_pred)
print(f'R-squared: {r2}')
```

```
R-squared: 0.7671159829204874
```