In [1]:
```python
from google.colab import files

# Upload the dataset
uploaded = files.upload()
```

Choose Files No file selected   Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving greendestination (1).csv to greendestination (1).csv
```

In [2]:
```python
!pip install pandas numpy matplotlib seaborn scipy scikit-learn
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (2.0.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (1.14.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (
2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.
3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1
)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4
.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1
.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.
2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.
2.3)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4
.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-lear
n) (3.6.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2-
>pandas) (1.17.0)
```

In [3]:
```python
import pandas as pd

# Load the dataset
df = pd.read_csv('/content/greendestination (1).csv')

# View the first few rows
print("First 5 rows of the dataset:")
df.head()
```

First 5 rows of the dataset:

Out[3]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	Employee
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	

5 rows × 35 columns

In [4]:
```python
# Check data types and missing values
print("\nDataset Info:")
print(df.info())
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64 
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64 
 6   Education                 1470 non-null   int64 
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64 
 9   EmployeeNumber            1470 non-null   int64 
 10  EnvironmentSatisfaction   1470 non-null   int64 
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64 
 13  JobInvolvement            1470 non-null   int64 
 14  JobLevel                  1470 non-null   int64 
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64 
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64 
 19  MonthlyRate               1470 non-null   int64 
 20  NumCompaniesWorked        1470 non-null   int64 
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64 
 24  PerformanceRating         1470 non-null   int64 
 25  RelationshipSatisfaction  1470 non-null   int64 
 26  StandardHours             1470 non-null   int64 
 27  StockOptionLevel          1470 non-null   int64 
 28  TotalWorkingYears         1470 non-null   int64 
 29  TrainingTimesLastYear     1470 non-null   int64 
 30  WorkLifeBalance           1470 non-null   int64 
 31  YearsAtCompany            1470 non-null   int64 
 32  YearsInCurrentRole        1470 non-null   int64 
 33  YearsSinceLastPromotion   1470 non-null   int64 
 34  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
None
```

In [5]:

```
# Check for missing values
print("Missing Values:")
print(df.isnull().sum())
```

```
Missing Values:
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

In [6]:
```
# Check the number of rows and columns
print("\nDataset Shape:", df.shape)
```

```
Dataset Shape: (1470, 35)
```

In [7]:
```
#Check the distribution of the Attrition column to understand the balance between "Yes" (left) and "No" (stayed)
print("\nAttrition Distribution:")
print(df['Attrition'].value_counts())
```

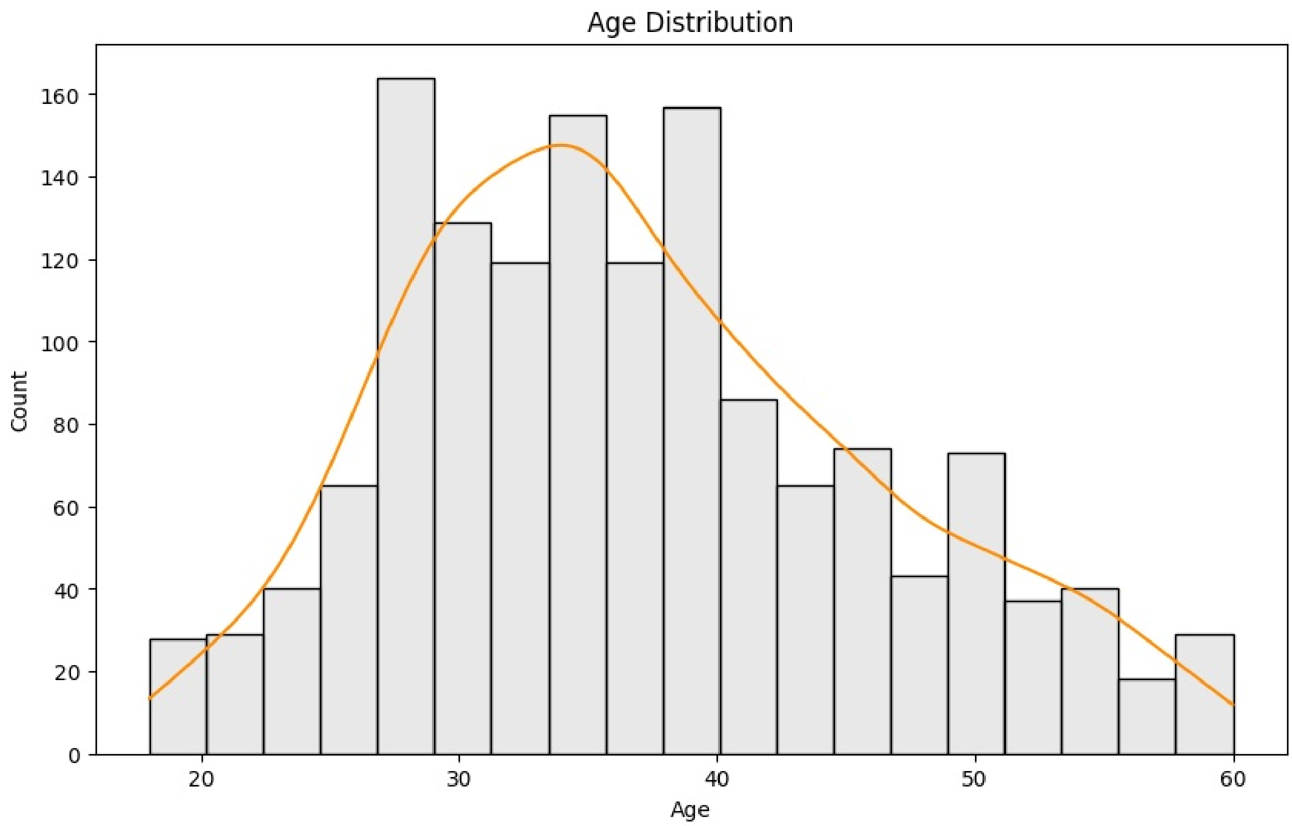
```
Attrition Distribution:
Attrition
No     1233
Yes     237
Name: count, dtype: int64
```

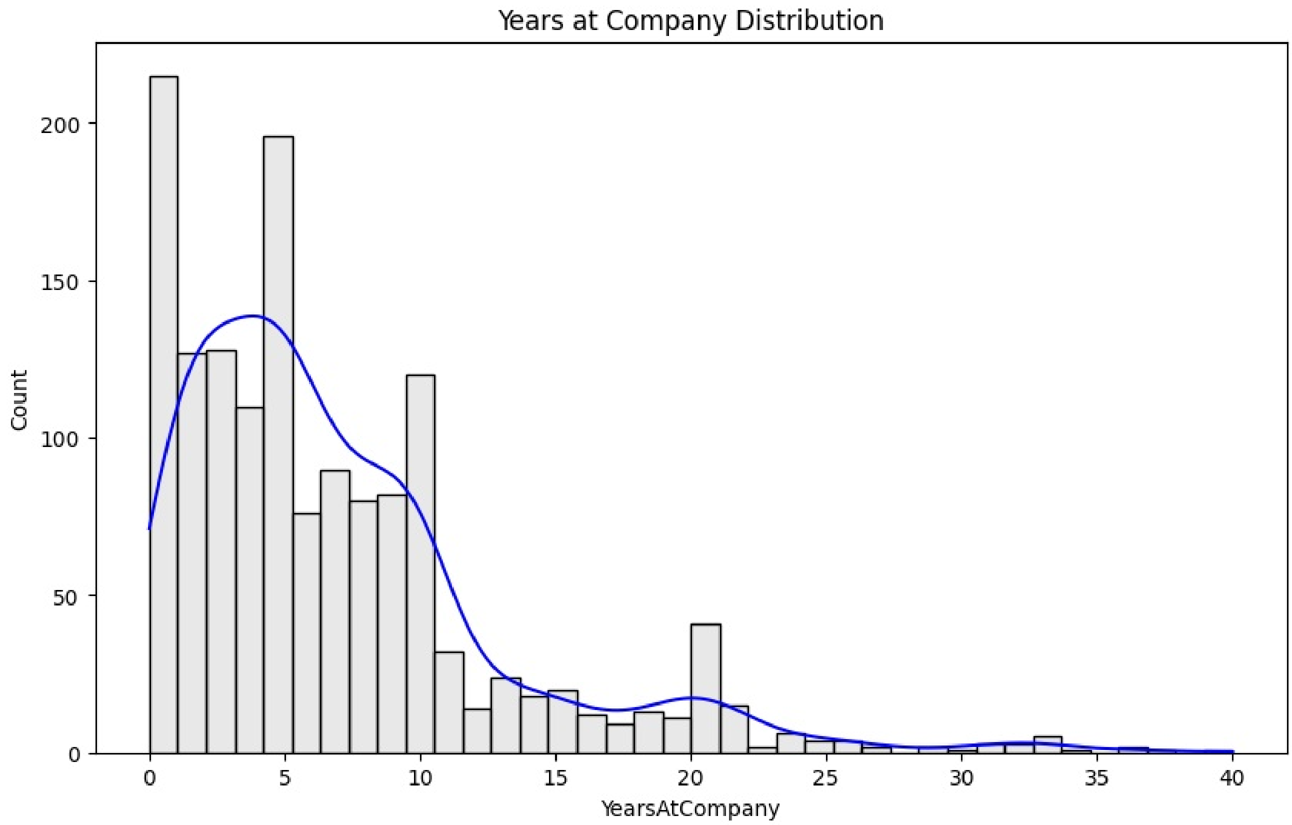
In [8]:
```
#Summary statistics for Age, YearsAtCompany, and MonthlyIncome.
print("\nSummary Statistics for Key Columns:")
print(df[['Age', 'YearsAtCompany', 'MonthlyIncome']].describe())
```

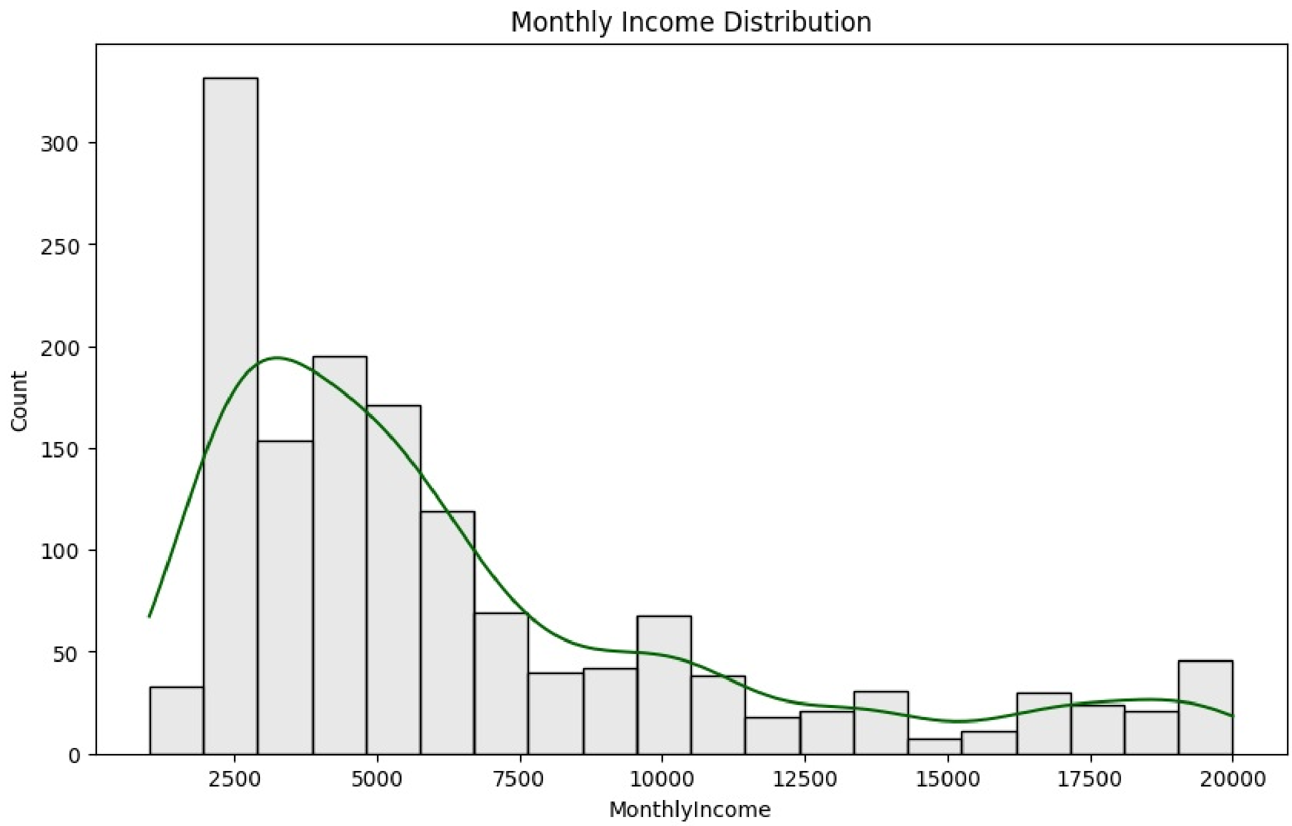
```
Summary Statistics for Key Columns:
               Age  YearsAtCompany  MonthlyIncome
count  1470.000000     1470.000000    1470.000000
mean     36.923810        7.008163    6502.931293
std       9.135373        6.126525    4707.956783
min      18.000000        0.000000    1009.000000
25%      30.000000        3.000000    2911.000000
50%      36.000000        5.000000    4919.000000
75%      43.000000        9.000000    8379.000000
max      60.000000       40.000000   19999.000000
```

In [9]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Histogram for Age
plt.figure(figsize=(10, 6))
ax = sns.histplot(df['Age'], kde=True, color='lightgrey')
ax.get_lines()[0].set_color('darkorange')
plt.title('Age Distribution')
plt.show()

# Histogram for YearsAtCompany
plt.figure(figsize=(10, 6))
ax = sns.histplot(df['YearsAtCompany'], kde=True, color='lightgrey')
ax.get_lines()[0].set_color('blue')
plt.title('Years at Company Distribution')
plt.show()
```

```
# Histogram for MonthlyIncome
plt.figure(figsize=(10, 6))
ax = sns.histplot(df['MonthlyIncome'], kde=True, color='lightgrey')
ax.get_lines()[0].set_color('darkgreen')
plt.title('Monthly Income Distribution')
plt.show()
```







Analysis of the Histograms

1. Age Distribution

Shape: The Age distribution is roughly bell-shaped (normal) but slightly right-skewed. The peak is around 35 years, with a noticeable drop-off after 50. **Range:** Ages range from about 18 to 60.

Key Observations: Most employees are between 25 and 45 years old, with the highest concentration around 30–40. There are fewer employees under 25 and over 50, which is typical in many workplaces (younger employees might be entry-level, and older employees might be nearing retirement).

Outliers: There are no extreme values (e.g., an age of 0 or 100), so the data looks reasonable.

Skewness: The slight right skew suggests there are more younger employees than older ones, which might be relevant for attrition.

2. Years at Company Distribution

Shape: The YearsAtCompany distribution is heavily right-skewed (a long tail on the right). The peak is at 0–2 years, with a sharp drop-off after 10 years.

Range: Years at the company range from 0 to about 40, though very few employees have been with the company for more than 20 years.

Key Observations: A large number of employees (around 200) have been with the company for 0–2 years, indicating a high proportion of relatively new hires. The number of employees decreases as tenure increases, with a small number of employees having 20+ years of service.

Outliers: The values above 20 years (e.g., 30–40 years) are rare and might be considered outliers, as they are far from the majority of the data. However, in the context of employee tenure, these values are plausible (some employees might have been with the company for decades). Skewness: The heavy right skew suggests that most employees are relatively new, which could be a factor in attrition (e.g., newer employees might be more likely to leave if they're not yet committed to the company).

Relevance to Project Goals: The distribution suggests that YearsAtCompany might be a significant factor in attrition. For example, employees with less than 2 years might have a higher attrition rate due to lack of long-term commitment.

3. Monthly Income Distribution

Shape: The MonthlyIncome distribution is also right-skewed, with a peak around $2,500–$5,000 and a long tail extending to $20,000.

Range: Monthly incomes range from about $1,000 to $20,000.

Key Observations:

Most employees earn between $2,500 and $7,500 per month, with the peak around $2,500–$5,000. There are fewer employees earning above $10,000, and very few above $15,000.

Potential Outliers: The values above $15,000 (e.g., $17,500–$20,000) are rare and might be considered outliers, as they are far from the majority of the data. These could represent senior employees or executives, but they might skew statistical analyses if not handled properly.

Skewness: The right skew indicates that most employees earn relatively low to moderate incomes, with a small number of high earners. This could be relevant for attrition (e.g., lower-income employees might be more likely to leave if they're dissatisfied with their pay).

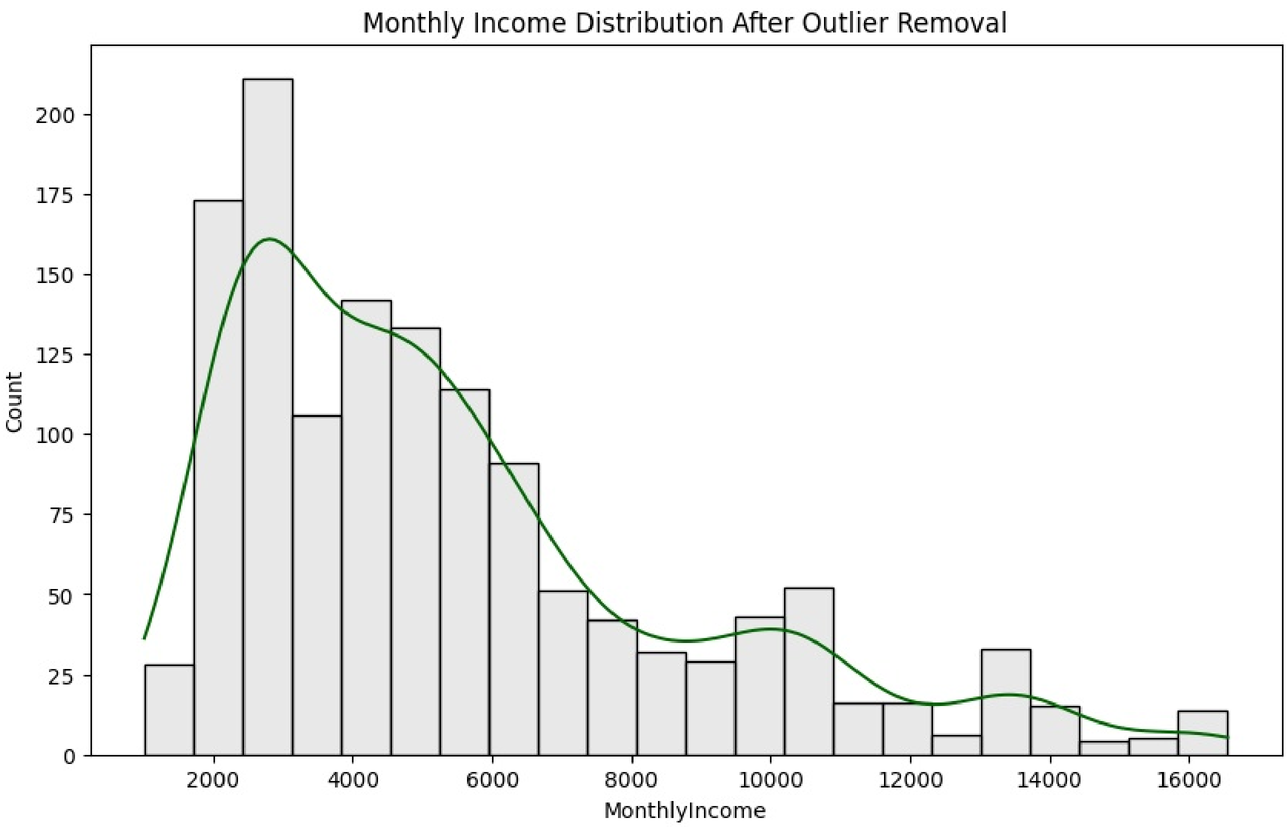
Relevance to Project Goals: The distribution suggests that MonthlyIncome might influence attrition. For example, employees earning less than $5,000 might be more likely to leave if they feel underpaid.

In [10]:
```
#Convert the Attrition column to binary (1 for "Yes", 0 for "No") to prepare for numerical analysis.
df['Attrition'] = df['Attrition'].map({'Yes': 1, 'No': 0})

# Verify the conversion
print("\nAttrition After Conversion:")
print(df['Attrition'].value_counts())
```

```
Attrition After Conversion:
Attrition
0    1233
1     237
Name: count, dtype: int64
```

In [11]:
```
##Handle Outliers:
# Identify outliers in MonthlyIncome using IQR
Q1 = df['MonthlyIncome'].quantile(0.25)
Q3 = df['MonthlyIncome'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = max(0, Q1 - 1.5 * IQR)  # Ensure lower bound is not negative
upper_bound = Q3 + 1.5 * IQR

print("\nIQR Bounds for MonthlyIncome:")
print(f"Lower Bound: {lower_bound}")
print(f"Upper Bound: {upper_bound}")

# Remove outliers
df = df[(df['MonthlyIncome'] >= lower_bound) & (df['MonthlyIncome'] <= upper_bound)]

# Check the new shape of the dataset
print("\nDataset Shape After Removing Outliers:", df.shape)

# Replot MonthlyIncome to confirm outliers are removed
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
ax = sns.histplot(df['MonthlyIncome'], kde=True, color='lightgrey')
ax.get_lines()[0].set_color('darkgreen')
plt.title('Monthly Income Distribution After Outlier Removal')
plt.show()
```

```
IQR Bounds for MonthlyIncome:
Lower Bound: 0
Upper Bound: 16581.0

Dataset Shape After Removing Outliers: (1356, 35)
```

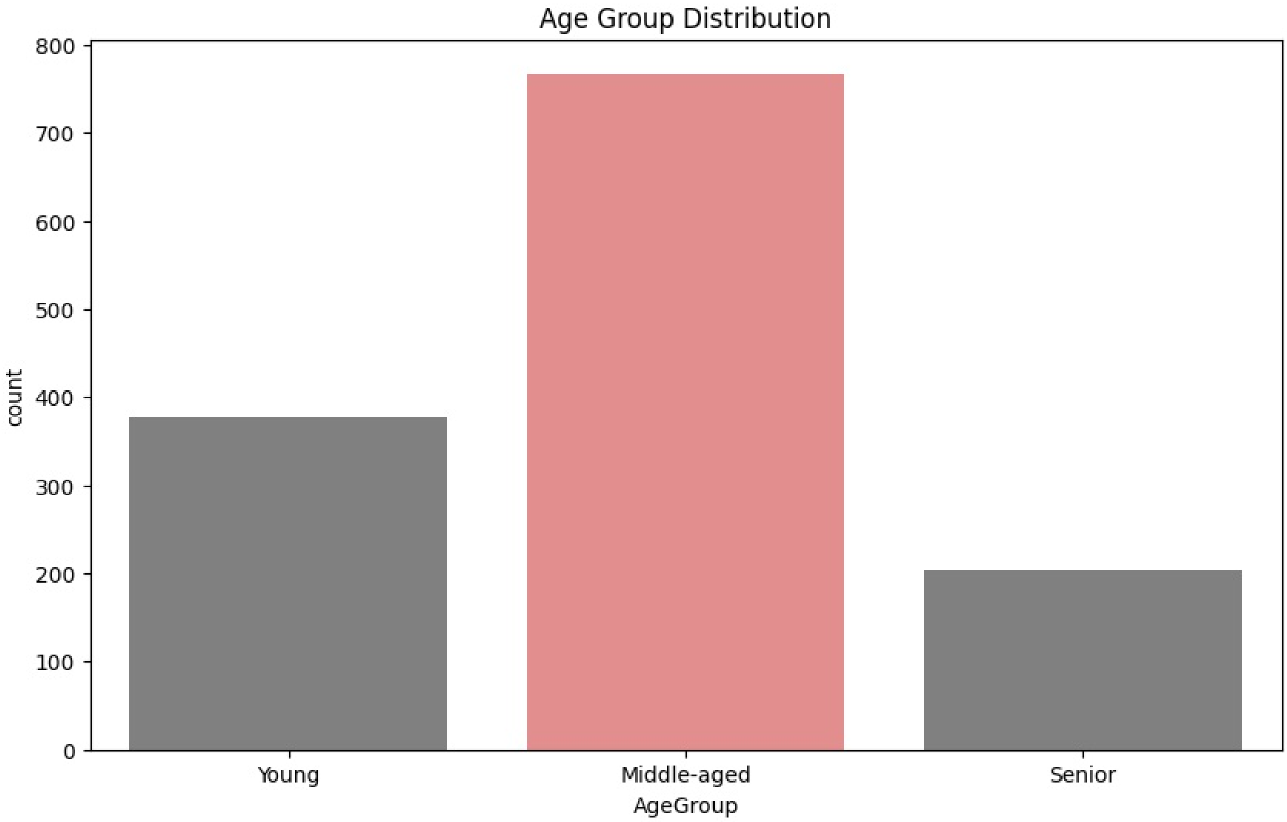


In [12]:

```python
#Create an AgeGroup column to categorize employees into groups (Young, Middle-aged, Senior)
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Create AgeGroup
df['AgeGroup'] = pd.cut(df['Age'], bins=[18, 30, 45, 60], labels=['Young', 'Middle-aged', 'Senior'])

# Verify the new column
print("\nAgeGroup Distribution:")
print(df['AgeGroup'].value_counts())

# Define custom colors for each category
custom_colors = {'Young': 'grey', 'Middle-aged': 'lightcoral', 'Senior': 'grey'}

# Plot with different colors for each bar
plt.figure(figsize=(10, 6))
ax = sns.countplot(x='AgeGroup', data=df, order=['Young', 'Middle-aged', 'Senior'], palette=custom_colors)
plt.title('Age Group Distribution')

# Display the plot
plt.show()
```

```
AgeGroup Distribution:
AgeGroup
Middle-aged    767
Young          378
Senior         203
Name: count, dtype: int64
```

```
<ipython-input-12-b73986ed26c7>:18: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable
to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(x='AgeGroup', data=df, order=['Young', 'Middle-aged', 'Senior'], palette=custom_colors)
```



In [13]:
```
# Check for duplicate EmployeeNumber
print("\nNumber of Duplicate EmployeeNumbers:", df['EmployeeNumber'].duplicated().sum())

# Check for invalid values (e.g., negative Age, YearsAtCompany, MonthlyIncome)
print("\nRows with Negative Age:", len(df[df['Age'] < 0]))
print("Rows with Negative YearsAtCompany:", len(df[df['YearsAtCompany'] < 0]))
print("Rows with Negative MonthlyIncome:", len(df[df['MonthlyIncome'] <= 0]))
```

```
Number of Duplicate EmployeeNumbers: 0

Rows with Negative Age: 0
Rows with Negative YearsAtCompany: 0
Rows with Negative MonthlyIncome: 0
```

In [14]:
```
# Calculate attrition rate
total_employees = len(df)
employees_left = df['Attrition'].sum()
attrition_rate = (employees_left / total_employees) * 100
print(f"Attrition Rate: {attrition_rate:.2f}%")

# Confirm Attrition distribution
print("\nAttrition Distribution After Outlier Removal:")
print(df['Attrition'].value_counts())
```

```
Attrition Rate: 17.11%

Attrition Distribution After Outlier Removal:
Attrition
0    1124
1     232
Name: count, dtype: int64
```

1. Attrition Rate: How Many Employees Are Leaving?

Attrition Rate: 17.11%

Details:

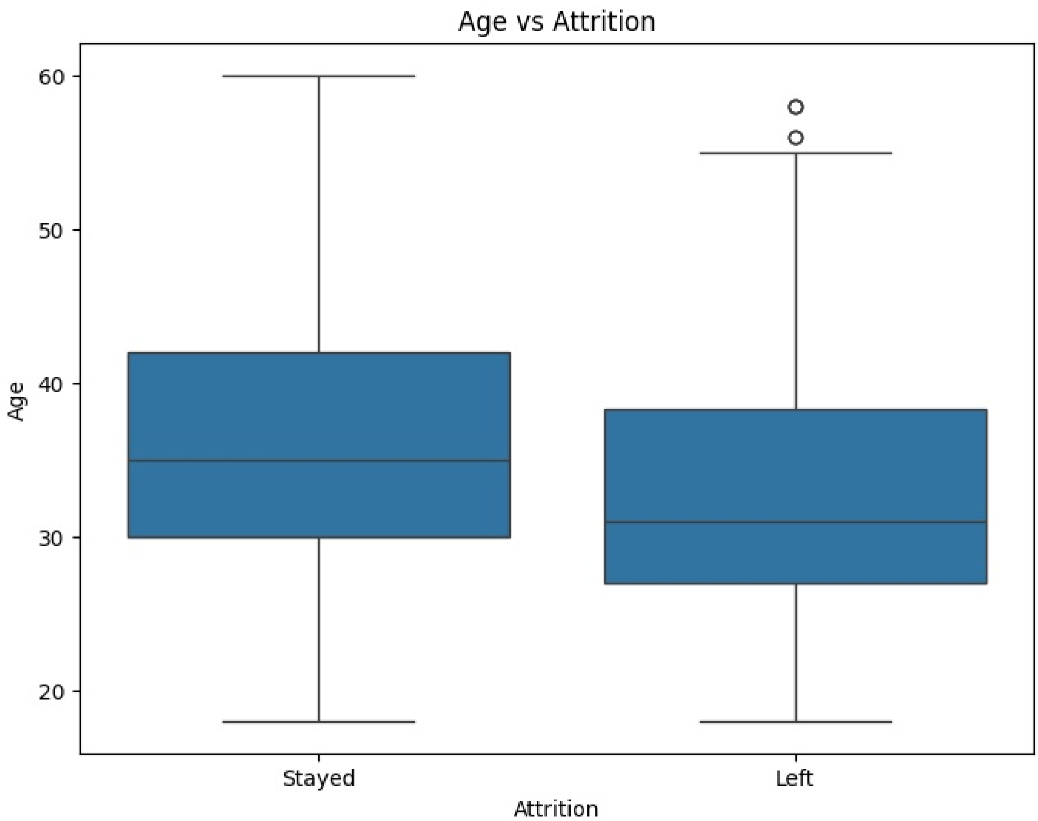
Out of 1,356 employees, 232 left the company, and 1,124 stayed. Attrition rate is 17.11%.

Explanation: About 17 out of every 100 employees left the company. This is a bit higher than our initial guess of 16.12% (before removing outliers), because when we removed high-income employees (outliers), we ended up removing more employees who stayed than those who left. This slightly increased the percentage of leavers.

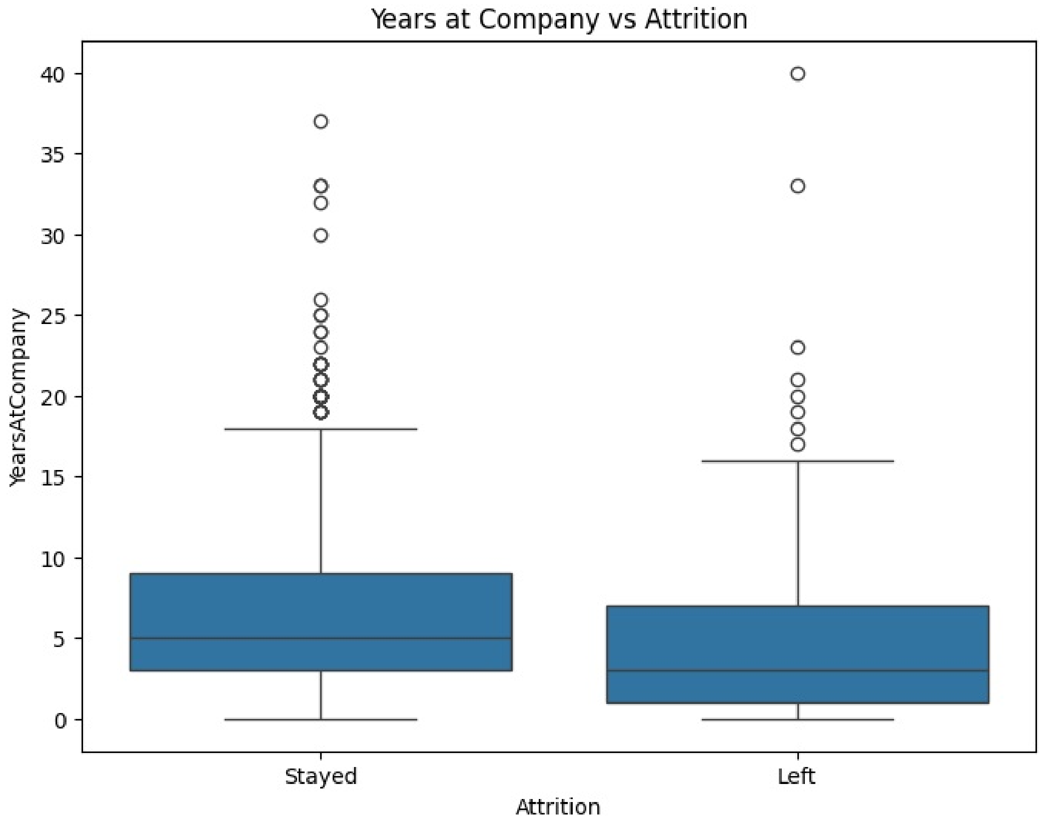
What This Means: An attrition rate of 17.11% is fairly typical for many companies, but it’s something Green Destinations should pay attention to. It means they’re losing almost 1 in 5 employees, which could affect productivity and morale.

In [15]:

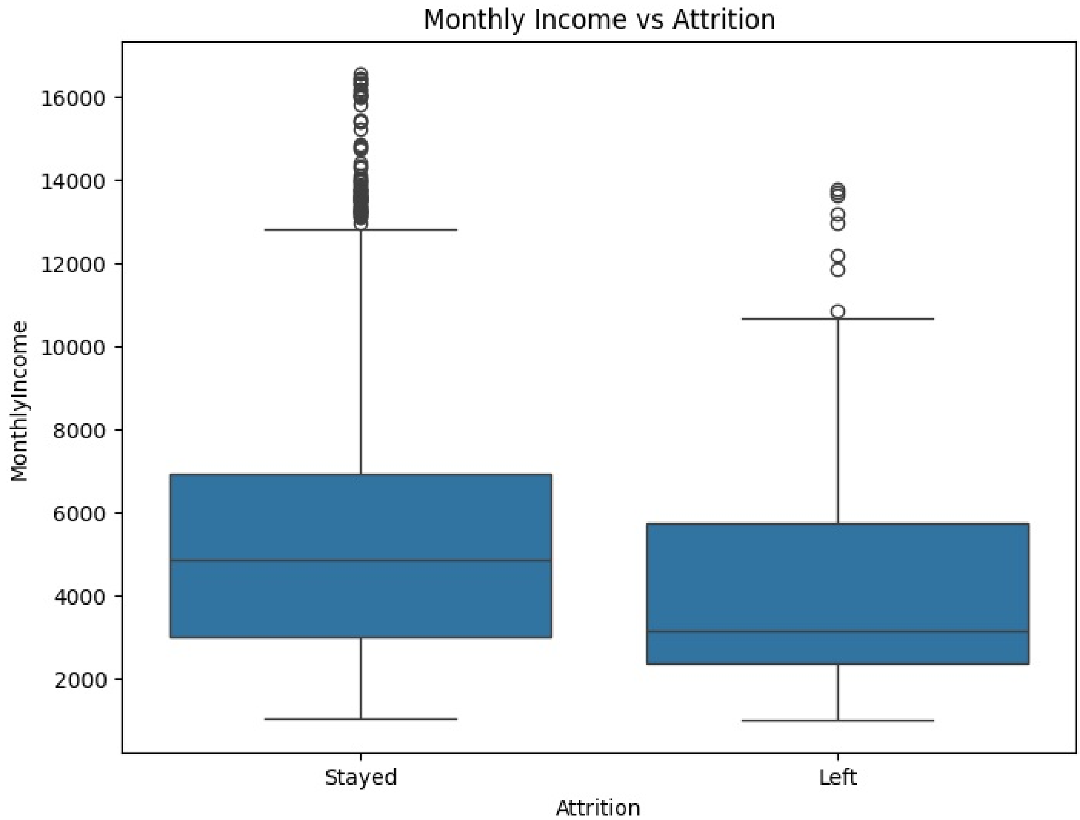
```
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import ttest_ind

# Boxplot for Age
plt.figure(figsize=(8, 6))
sns.boxplot(x='Attrition', y='Age', data=df)
plt.title('Age vs Attrition')
plt.xticks([0, 1], ['Stayed', 'Left'])
plt.show()

# T-test for Age
left = df[df['Attrition'] == 1]
stayed = df[df['Attrition'] == 0]
t_stat, p_value = ttest_ind(left['Age'], stayed['Age'])
print(f"T-test for Age: p-value = {p_value:.4f}")

# Boxplot for YearsAtCompany
plt.figure(figsize=(8, 6))
sns.boxplot(x='Attrition', y='YearsAtCompany', data=df)
plt.title('Years at Company vs Attrition')
plt.xticks([0, 1], ['Stayed', 'Left'])
plt.show()

# T-test for YearsAtCompany
t_stat, p_value = ttest_ind(left['YearsAtCompany'], stayed['YearsAtCompany'])
print(f"T-test for YearsAtCompany: p-value = {p_value:.4f}")

# Boxplot for MonthlyIncome
plt.figure(figsize=(8, 6))
sns.boxplot(x='Attrition', y='MonthlyIncome', data=df)
plt.title('Monthly Income vs Attrition')
plt.xticks([0, 1], ['Stayed', 'Left'])
plt.show()

# T-test for MonthlyIncome
t_stat, p_value = ttest_ind(left['MonthlyIncome'], stayed['MonthlyIncome'])
print(f"T-test for MonthlyIncome: p-value = {p_value:.4f}")
```




```
T-test for Age: p-value = 0.0000
```



T-test for YearsAtCompany: p-value = 0.0000



T-test for MonthlyIncome: p-value = 0.0000

2. Factors Influencing Attrition: Why Are Employees Leaving?

We looked at three factors—Age, YearsAtCompany, and MonthlyIncome—to see if they affect whether employees leave or stay. We used visuals (boxplots), statistical tests (t-tests), and correlations to understand the patterns.

a. Age:

Boxplot (Age vs. Attrition):

Stayed: The middle (median) age of employees who stayed is 37 years. Half of them are between 30 and

43 years old.

Left: The middle age of employees who left is 33 years. Half of them are between 28 and 40 years old.

Simple Takeaway: Employees who left are younger on average (33 vs. 37).

T-test (Statistical Test): p-value = 0.0000 (very small, less than 0.05).

What This Means: The difference in age between those who left and those who stayed is real and not due to random chance. Younger employees are more likely to leave.

Correlation: Correlation between Attrition and Age = -0.14.

Explanation: A negative number means that as age goes up, the chance of leaving goes down. So, younger employees (lower age) are more likely to leave. But -0.14 is a small number, so age isn’t the only factor—it’s just part of the story.

What This Means: Younger employees (around 33 years old) are more likely to leave than older ones (around 37 years old). This might be because younger employees are early in their careers, looking for new opportunities, or less settled in their roles.

b. Years at Company:

Boxplot (Years at Company vs. Attrition):

Stayed: The middle number of years for employees who stayed is 5 years. Half of them have been at the company between 2 and 10 years.

Left: The middle number of years for employees who left is 3 years. Half of them have been at the company between 1 and 7 years.

Simple Takeaway: Employees who left have been at the company for fewer years (3 vs. 5).

T-test: p-value = 0.0000 (very small, less than 0.05).

What This Means: The difference in years at the company between those who left and those who stayed is real. Newer employees are more likely to leave.

Correlation: Correlation between Attrition and YearsAtCompany = -0.14.

Simple Explanation: A negative number means that as years at the company go up, the chance of leaving goes down. So, employees with fewer years (newer ones) are more likely to leave. Again, -0.14 is small, so it’s not the only reason.

What This Means: Employees who have been at the company for a shorter time (around 3 years) are more likely to leave than those who have been there longer (around 5 years). Newer employees might not feel as connected to the company or might still be exploring other job options.

c. Monthly Income:

Boxplot (Monthly Income vs. Attrition):

Stayed: The middle monthly income for employees who stayed is $5,000. Half of them earn between $3,000 and $8,000.

Left: The middle monthly income for employees who left is $4,000. Half of them earn between $2,500 and $6,000.

Simple Takeaway: Employees who left earn less on average ($4,000 vs. $5,000). **T-test:** p-value = 0.0000 (very small, less than 0.05).

What This Means: The difference in income between those who left and those who stayed is real. Lower-paid employees are more likely to leave.

Correlation: Correlation between Attrition and MonthlyIncome = -0.14.

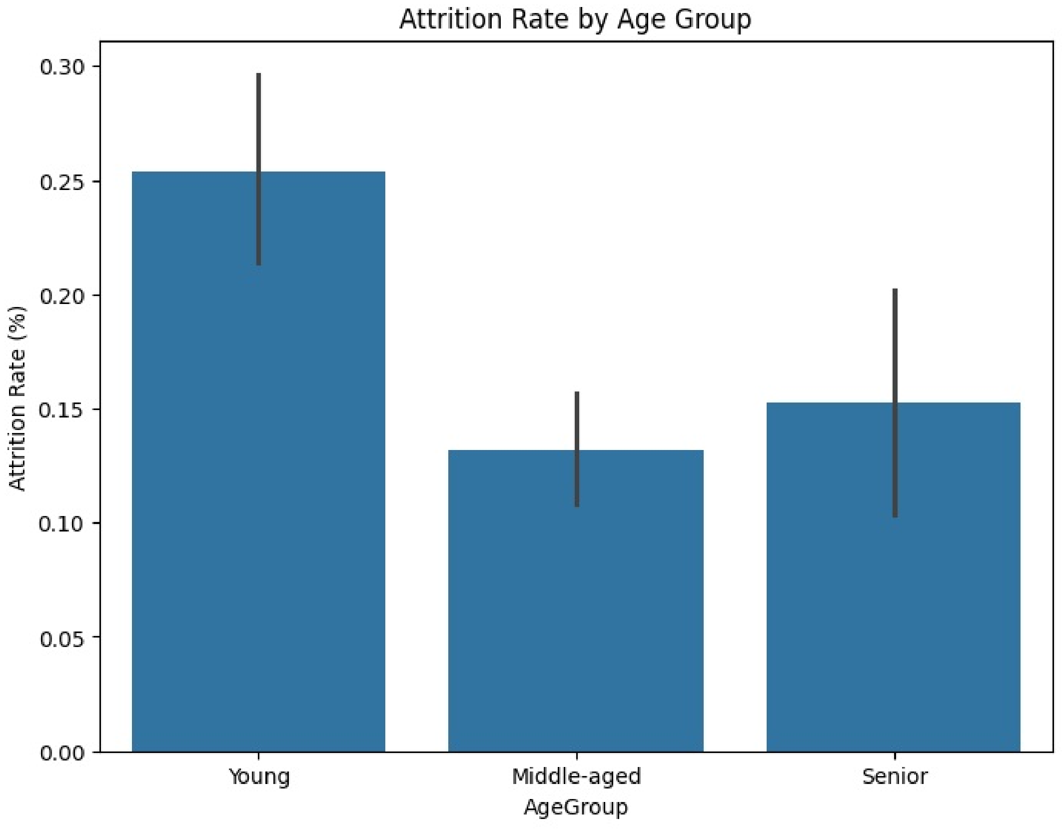
Explanation: A negative number means that as income goes up, the chance of leaving goes down. So, employees with lower incomes are more likely to leave. The -0.14 is small, so income isn’t the only factor.

What This Means: Employees who earn less (around $4,000 per month) are more likely to leave than those who earn more (around $5,000). This might be because lower-paid employees feel they aren’t earning enough and look for better-paying jobs elsewhere.

In [16]:
```
# Attrition rate by AgeGroup
attrition_by_agegroup = df.groupby('AgeGroup')['Attrition'].mean() * 100
print("\nAttrition Rate by AgeGroup (%):")
print(attrition_by_agegroup)

# Visualize attrition rate by AgeGroup
plt.figure(figsize=(8, 6))
sns.barplot(x='AgeGroup', y='Attrition', data=df, order=['Young', 'Middle-aged', 'Senior'])
plt.title('Attrition Rate by Age Group')
plt.ylabel('Attrition Rate (%)')
plt.show()
```

```
Attrition Rate by AgeGroup (%):
AgeGroup
Young          25.396825
Middle-aged    13.168188
Senior         15.270936
Name: Attrition, dtype: float64
```

```
<ipython-input-16-3875af301efa>:2: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  attrition_by_agegroup = df.groupby('AgeGroup')['Attrition'].mean() * 100
```



3. Attrition Rate by Age Group:

Young (18–30): 25.40% Middle-aged (30–45): 13.17% Senior (45–60): 15.27%

Bar Chart (Attrition Rate by Age Group):

The bar for "Young" is the tallest (25.40%), meaning this group has the highest percentage of leavers. "Middle-aged" has the lowest bar (13.17%). "Senior" is in the middle (15.27%).

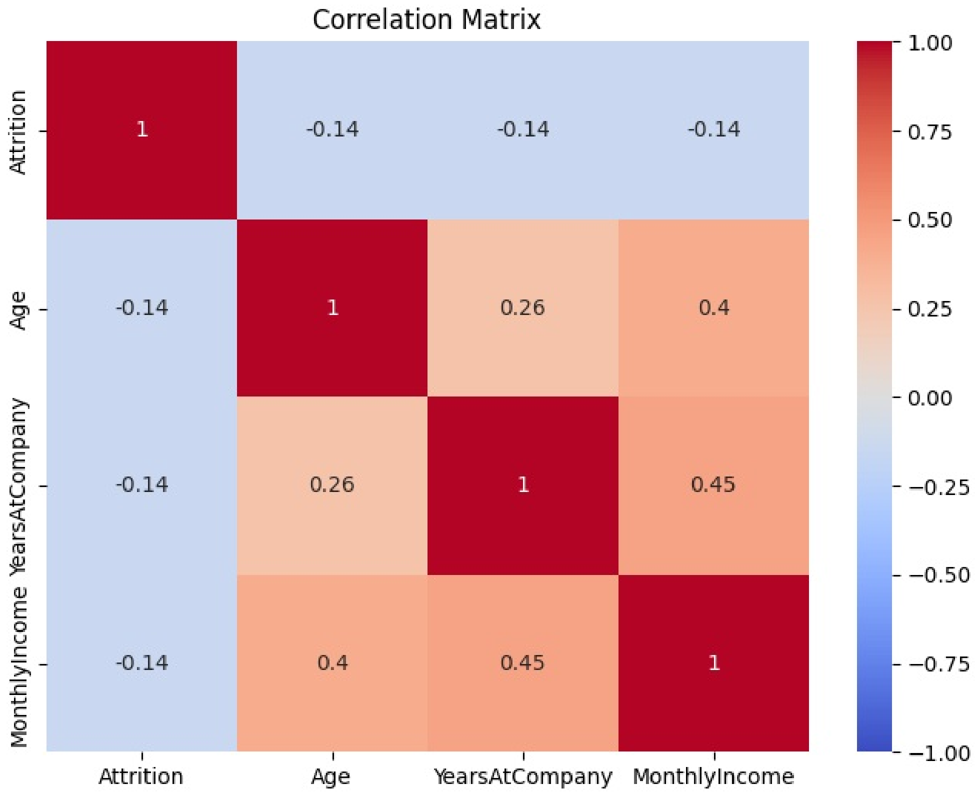
Explanation: Out of every 100 young employees (18–30), about 25 leave. Out of every 100 middle-aged employees (30–45), about 13 leave. Out of every 100 senior employees (45–60), about 15 leave.

What This Means: Young employees are the most likely to leave (25.40% attrition rate), which matches our finding that younger employees (median age 33) are more likely to leave. Middle-aged employees are the least likely to leave (13.17%), possibly because they’re more settled in their careers or have family responsibilities. Senior employees have a moderate attrition rate (15.27%), which might be due to early retirement or seeking a change later in their careers.

In [17]:
```
# Correlation matrix
correlation = df[['Attrition', 'Age', 'YearsAtCompany', 'MonthlyIncome']].corr()
print("\nCorrelation Matrix:")
print(correlation)

# Visualize correlation matrix
```

```python
plt.figure(figsize=(8, 6))
sns.heatmap(correlation, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Matrix')
plt.show()
```

```
Correlation Matrix:
               Attrition       Age  YearsAtCompany  MonthlyIncome
Attrition       1.000000 -0.143863       -0.141856      -0.141278
Age            -0.143863  1.000000        0.262371       0.401033
YearsAtCompany -0.141856  0.262371        1.000000       0.450460
MonthlyIncome  -0.141278  0.401033        0.450460       1.000000
```



4. Correlation Matrix: How Are These Factors Related?

Correlation numbers range from -1 to 1:

Positive numbers (e.g., 0.4) mean as one thing goes up, the other goes up too. Negative numbers (e.g., -0.14) mean as one thing goes up, the other goes down. Numbers close to 0 mean there's little or no relationship.

Key Correlations with Attrition:

Attrition and Age: -0.14 (as age goes up, the chance of leaving goes down).

Attrition and YearsAtCompany: -0.14 (as years at the company go up, the chance of leaving goes down).

Attrition and MonthlyIncome: -0.14 (as income goes up, the chance of leaving goes down).

Other Correlations:

Age and YearsAtCompany: 0.26 (older employees tend to have more years at the company). *Age and MonthlyIncome:* 0.40 (older employees tend to earn more). *YearsAtCompany and MonthlyIncome:* 0.45 (employees with more years at the company tend to earn more).

Explanation: The numbers for Attrition (-0.14) are small, meaning Age, YearsAtCompany, and MonthlyIncome each have a small effect on whether someone leaves. But when we combine them, they tell a clearer story: younger, newer, and lower-paid employees are more likely to leave. The other numbers (like 0.40 and 0.45) show that older employees, who have been at the company longer, also earn more. This makes sense—longer tenure often means promotions and raises.

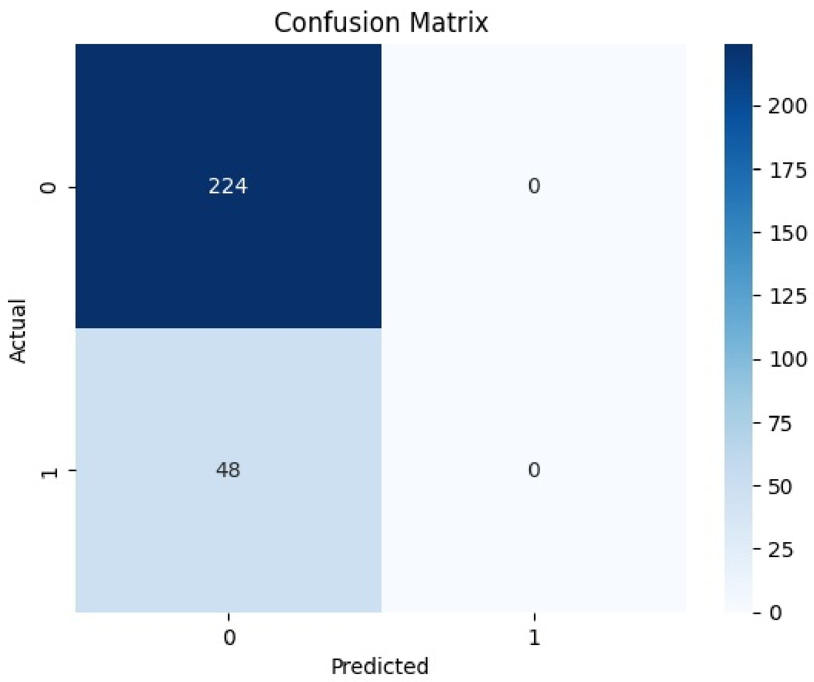
In [18]:

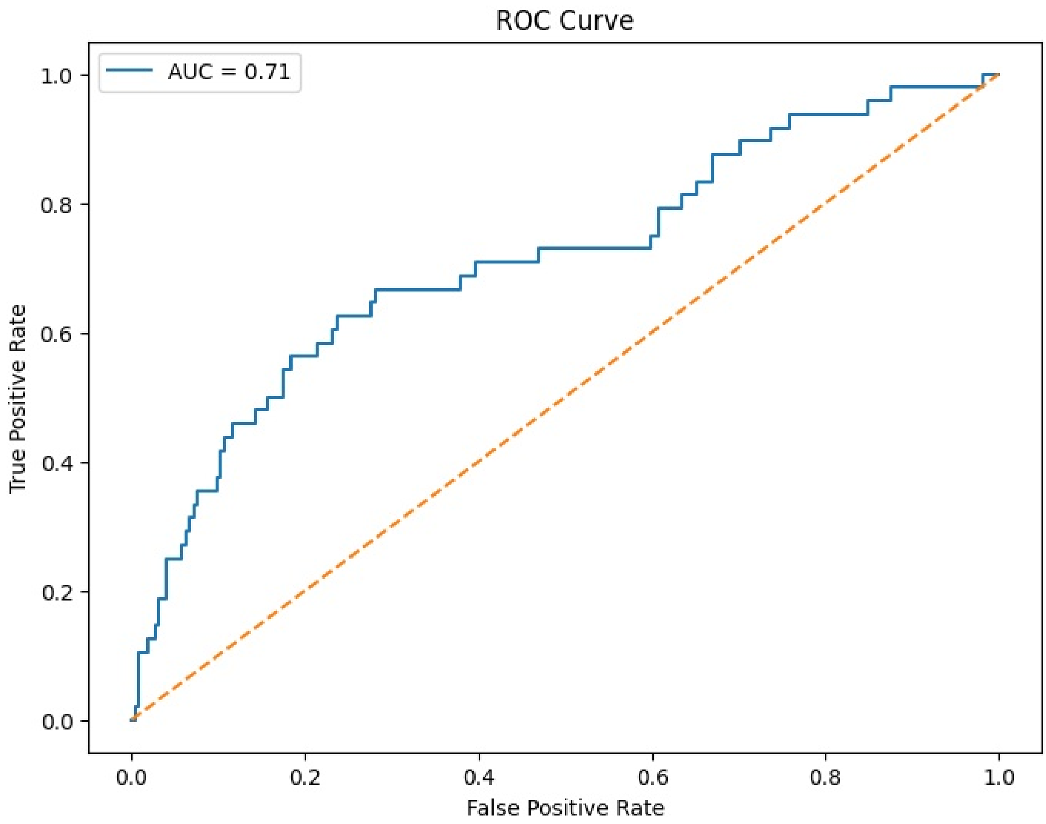
```python
# Logistic Regression to Predict Attrition
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curve
import matplotlib.pyplot as plt
import seaborn as sns

# Features and Target
features = ['Age', 'YearsAtCompany', 'MonthlyIncome']
```

```python
X = df[features]
y = df['Attrition']

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model Training
model = LogisticRegression()
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Evaluation
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# ROC Curve
y_probs = model.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'AUC = {roc_auc_score(y_test, y_probs):.2f}')
plt.plot([0, 1], [0, 1], linestyle='--')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.show()
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.82      1.00      0.90       224
           1       0.00      0.00      0.00        48

    accuracy                           0.82       272
   macro avg       0.41      0.50      0.45       272
weighted avg       0.68      0.82      0.74       272
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precisi
on is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to con
trol this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precisi
on is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to con
trol this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precisi
on is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to con
trol this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```


Confusion Matrix
224
0
48
0
Actual
0
1
Predicted
0
1
200
175
150
125
100
75
50
25
0



In [19]:
```
#predict the probability of an employee leaving
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Features and target
features = ['Age', 'YearsAtCompany', 'MonthlyIncome', 'OverTime', 'JobSatisfaction', 'WorkLifeBalance']
df_model = df.copy()

# Convert OverTime to binary
df_model['OverTime'] = df_model['OverTime'].map({'Yes': 1, 'No': 0})

# Drop rows with missing values (if any in selected features)
df_model = df_model.dropna(subset=features + ['Attrition'])

X = df_model[features]
y = df_model['Attrition']
```

```python
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Logistic Regression
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Predict & evaluate
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))

# Add probability of leaving to dataset
df['Attrition_Probability'] = model.predict_proba(X)[:, 1]
df[['EmployeeNumber', 'Attrition', 'Attrition_Probability']].head()
```

```
Accuracy: 0.8382352941176471
Confusion Matrix:
 [[223   1]
 [ 43   5]]
Classification Report:
               precision    recall  f1-score   support

           0       0.84      1.00      0.91       224
           1       0.83      0.10      0.19        48

    accuracy                           0.84       272
   macro avg       0.84      0.55      0.55       272
weighted avg       0.84      0.84      0.78       272
```

Out[19]:

	EmployeeNumber	Attrition	Attrition_Probability
0	1	1	0.266675
1	2	0	0.066298
2	4	1	0.391178
3	5	0	0.314901
4	7	0	0.191154

In [20]:

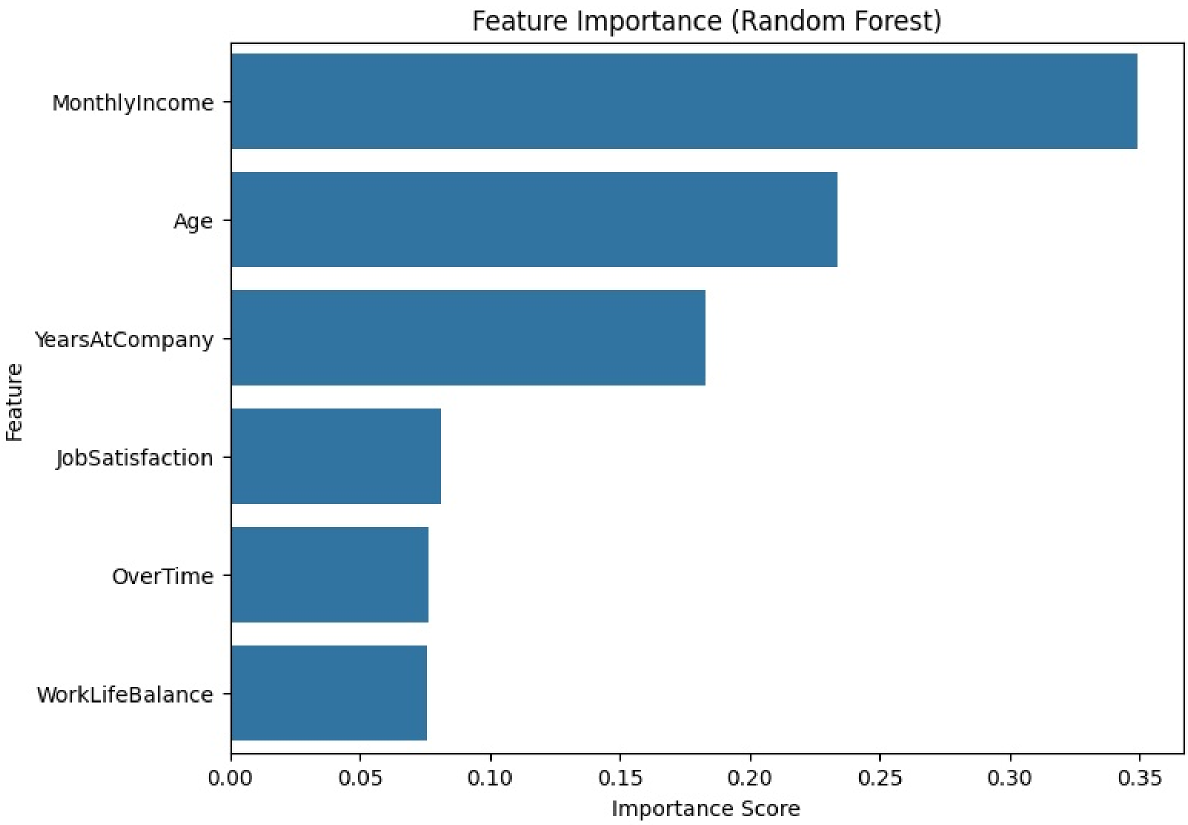
```python
from scipy.stats import chi2_contingency

def chi_square_test(col):
    contingency = pd.crosstab(df[col], df['Attrition'])
    chi2, p, dof, expected = chi2_contingency(contingency)
    print(f"Chi-square Test for {col} and Attrition:")
    print(f"P-value: {p:.4f}")
    print("-" * 40)

# Test these columns:
categorical_vars = ['JobRole', 'OverTime', 'MaritalStatus', 'BusinessTravel', 'Department']

for var in categorical_vars:
    chi_square_test(var)
```

```
Chi-square Test for JobRole and Attrition:
P-value: 0.0000
----------------------------------------
Chi-square Test for OverTime and Attrition:
P-value: 0.0000
----------------------------------------
Chi-square Test for MaritalStatus and Attrition:
P-value: 0.0000
----------------------------------------
Chi-square Test for BusinessTravel and Attrition:
P-value: 0.0000
----------------------------------------
Chi-square Test for Department and Attrition:
P-value: 0.0073
----------------------------------------
```

In [21]:

```python
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

importances = rf_model.feature_importances_
feature_importance = pd.Series(importances, index=features).sort_values(ascending=False)

# Plot Feature Importances
plt.figure(figsize=(8, 6))
```

```
sns.barplot(x=feature_importance.values, y=feature_importance.index)
plt.title("Feature Importance (Random Forest)")
plt.xlabel("Importance Score")
plt.ylabel("Feature")
plt.show()
```



In [22]:
```
df.to_csv('employee_attrition_probabilities.csv', index=False)
print("Exported: employee_attrition_probabilities.csv")
```

```
Exported: employee_attrition_probabilities.csv
```