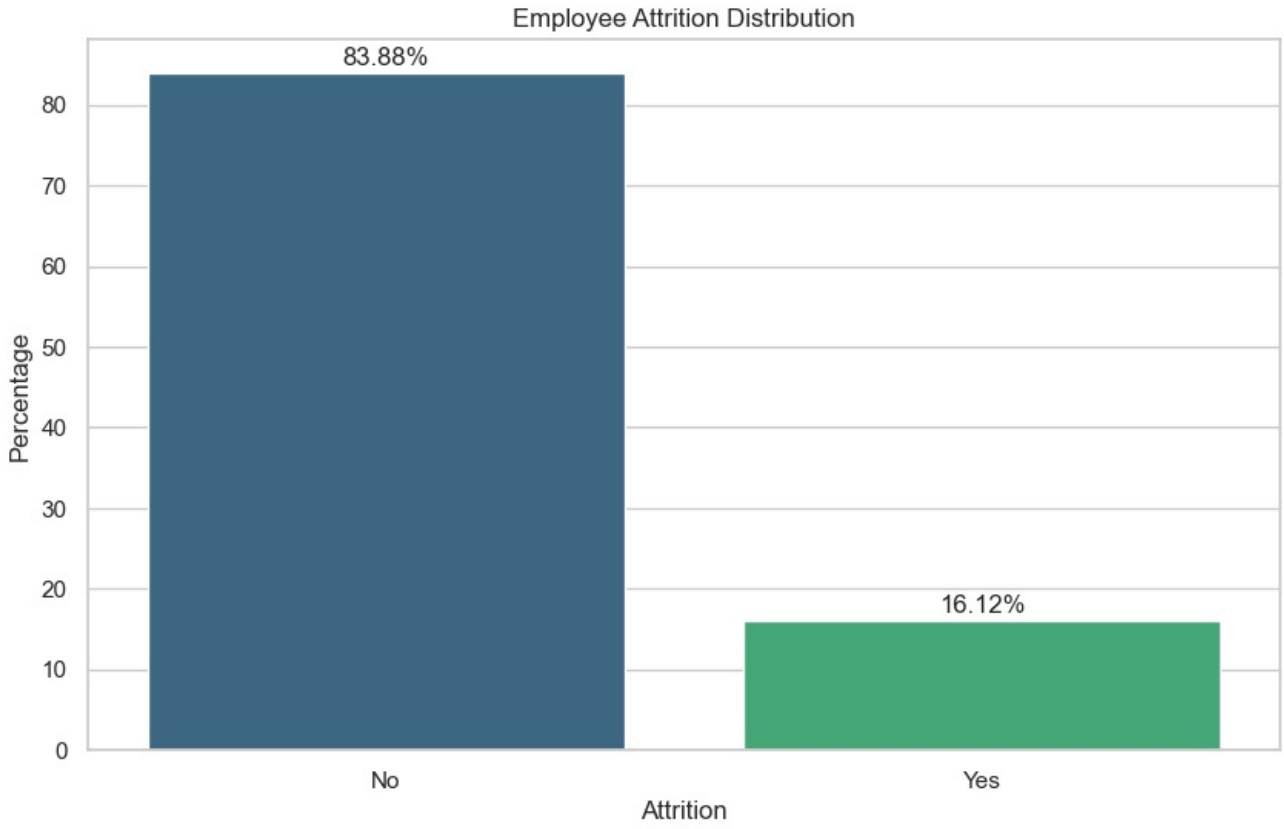
In [1]:
```
# IBM HR Analytics Employee Attrition & Performance Project

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
warnings.filterwarnings('ignore')
sns.set(style='whitegrid')
plt.rcParams['figure.figsize'] = (10, 6)
```

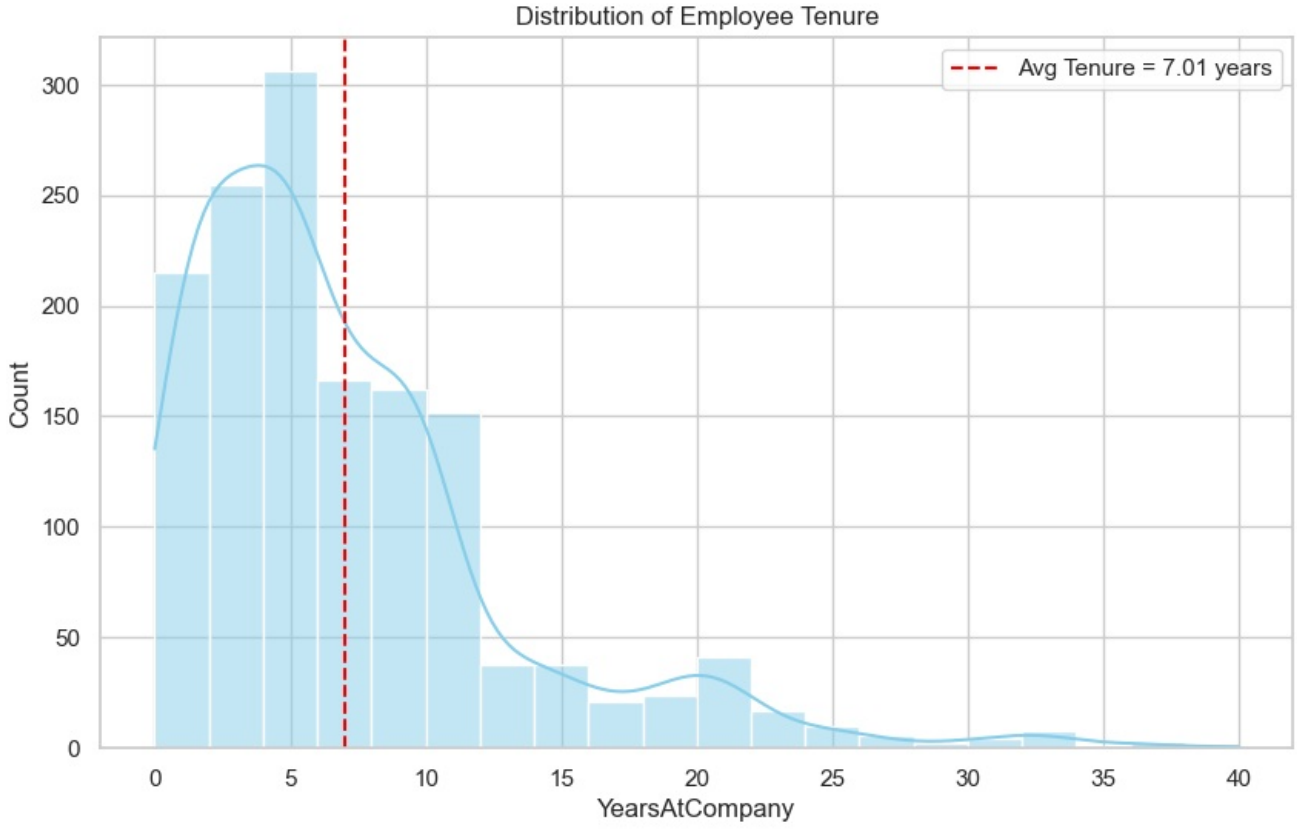
In [3]:
```
#Load Dataset
df = pd.read_csv("C:\\Users\\HP\\Downloads\\WA_Fn-UseC_-HR-Employee-Attrition.csv")

# Data Overview
df.drop(['EmployeeCount', 'EmployeeNumber', 'StandardHours', 'Over18'], axis=1, inplace=True)
print(f"Shape: {df.shape}")
print(f"Missing values: \n{df.isnull().sum().sum()}")
print(f"Duplicated rows: {df.duplicated().sum()}")
print(df.dtypes)
```

```
Shape: (1470, 31)
Missing values:
0
Duplicated rows: 0
Age                         int64
Attrition                  object
BusinessTravel             object
DailyRate                   int64
Department                 object
DistanceFromHome            int64
Education                   int64
EducationField             object
EnvironmentSatisfaction     int64
Gender                     object
HourlyRate                  int64
JobInvolvement              int64
JobLevel                    int64
JobRole                    object
JobSatisfaction             int64
MaritalStatus              object
MonthlyIncome               int64
MonthlyRate                 int64
NumCompaniesWorked          int64
OverTime                   object
PercentSalaryHike           int64
PerformanceRating           int64
RelationshipSatisfaction    int64
StockOptionLevel            int64
TotalWorkingYears           int64
TrainingTimesLastYear       int64
WorkLifeBalance             int64
YearsAtCompany              int64
YearsInCurrentRole          int64
YearsSinceLastPromotion     int64
YearsWithCurrManager        int64
dtype: object
```

In [4]:
```
# Attrition Rate
attrition_percentage = df['Attrition'].value_counts(normalize=True) * 100
sns.barplot(x=attrition_percentage.index, y=attrition_percentage.values, palette='viridis')
for index, value in enumerate(attrition_percentage.values):
    plt.text(index, value + 1, f"{value:.2f}%", ha='center')
plt.title("Employee Attrition Distribution")
plt.xlabel("Attrition")
plt.ylabel("Percentage")
plt.show()
```

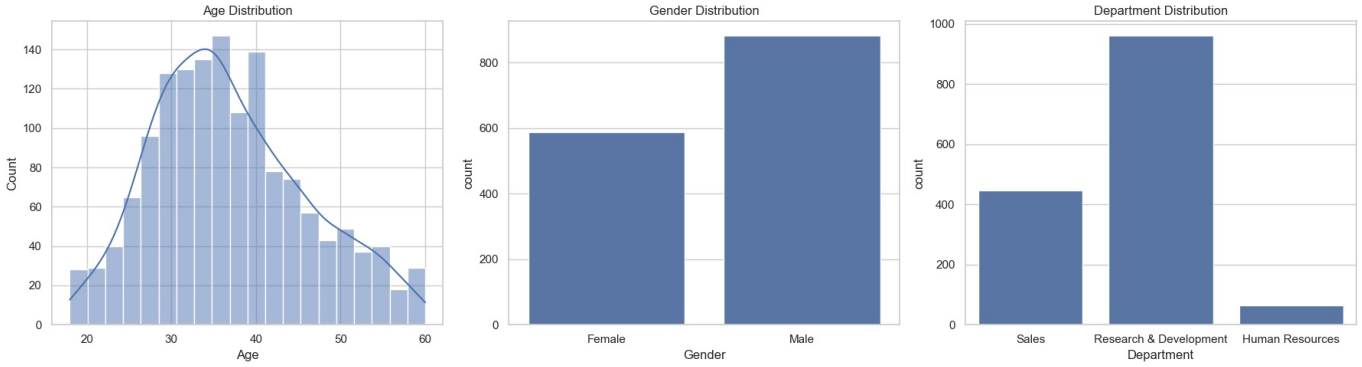


In [5]:
```
# Average Tenure
avg_tenure = df['YearsAtCompany'].mean()
sns.histplot(df['YearsAtCompany'], kde=True, bins=20, color='skyblue')
plt.axvline(avg_tenure, color='red', linestyle='--', label=f"Avg Tenure = {avg_tenure:.2f} years")
plt.title("Distribution of Employee Tenure")
plt.legend()
plt.show()
```

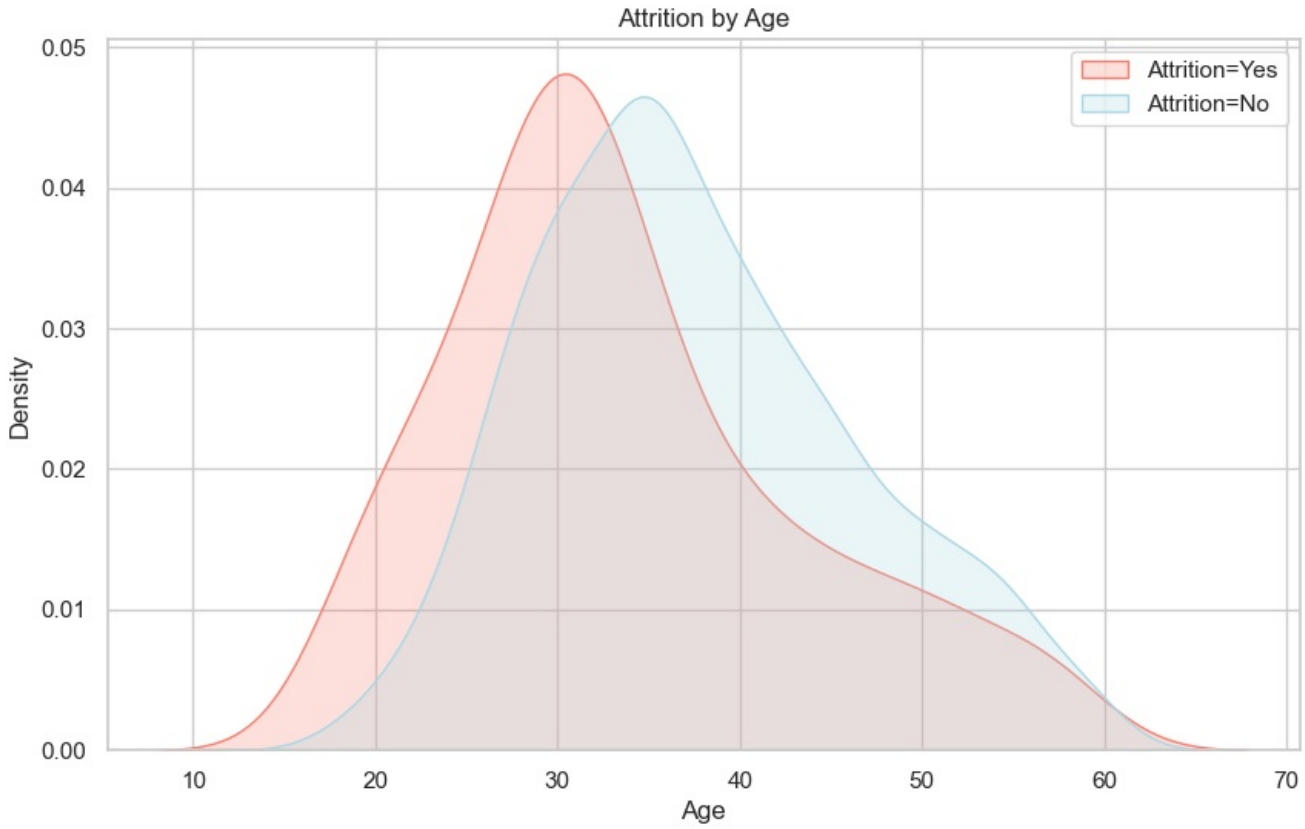


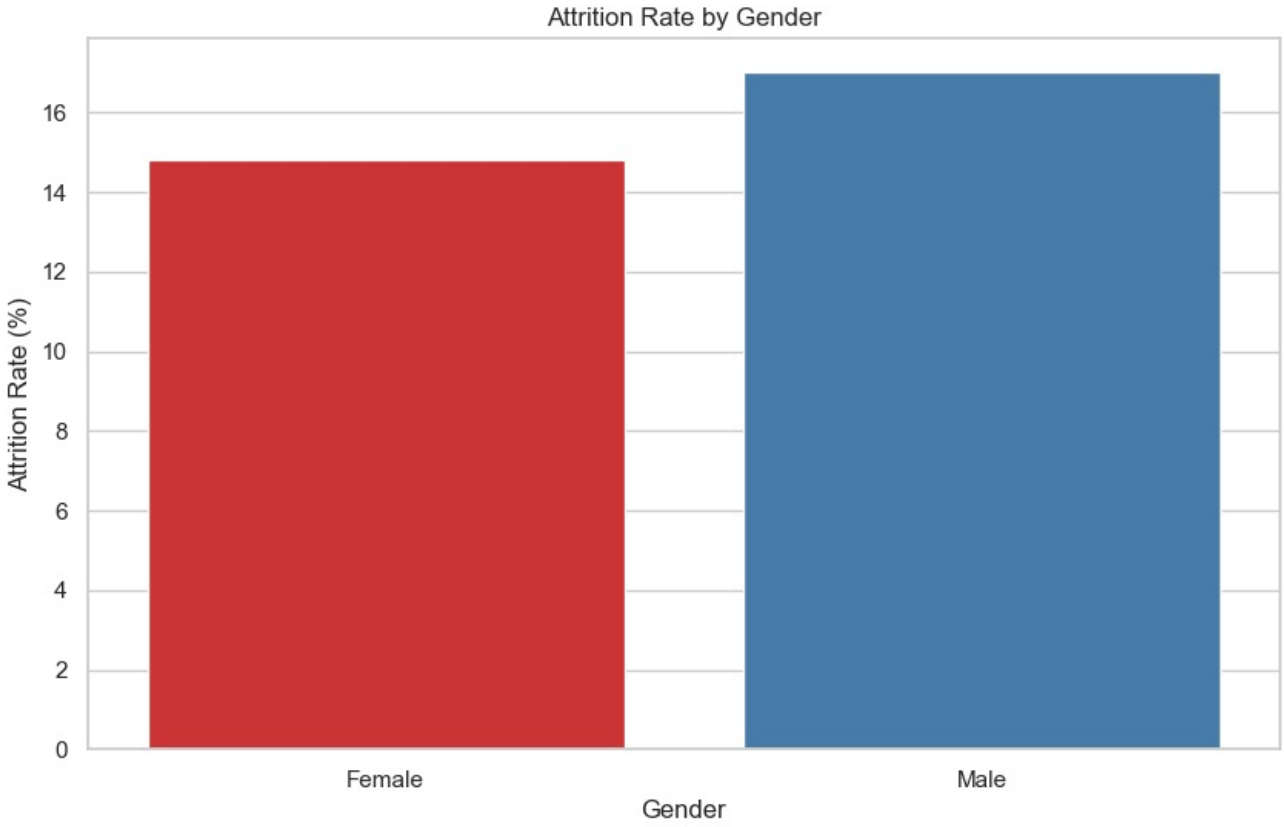
In [6]:
```
# Demographic Distribution
fig, axes = plt.subplots(1, 3, figsize=(18, 5))
sns.histplot(df['Age'], kde=True, bins=20, ax=axes[0])
sns.countplot(data=df, x='Gender', ax=axes[1])
sns.countplot(data=df, x='Department', ax=axes[2])
axes[0].set_title('Age Distribution')
axes[1].set_title('Gender Distribution')
axes[2].set_title('Department Distribution')
plt.tight_layout()
```

```
plt.show()
```

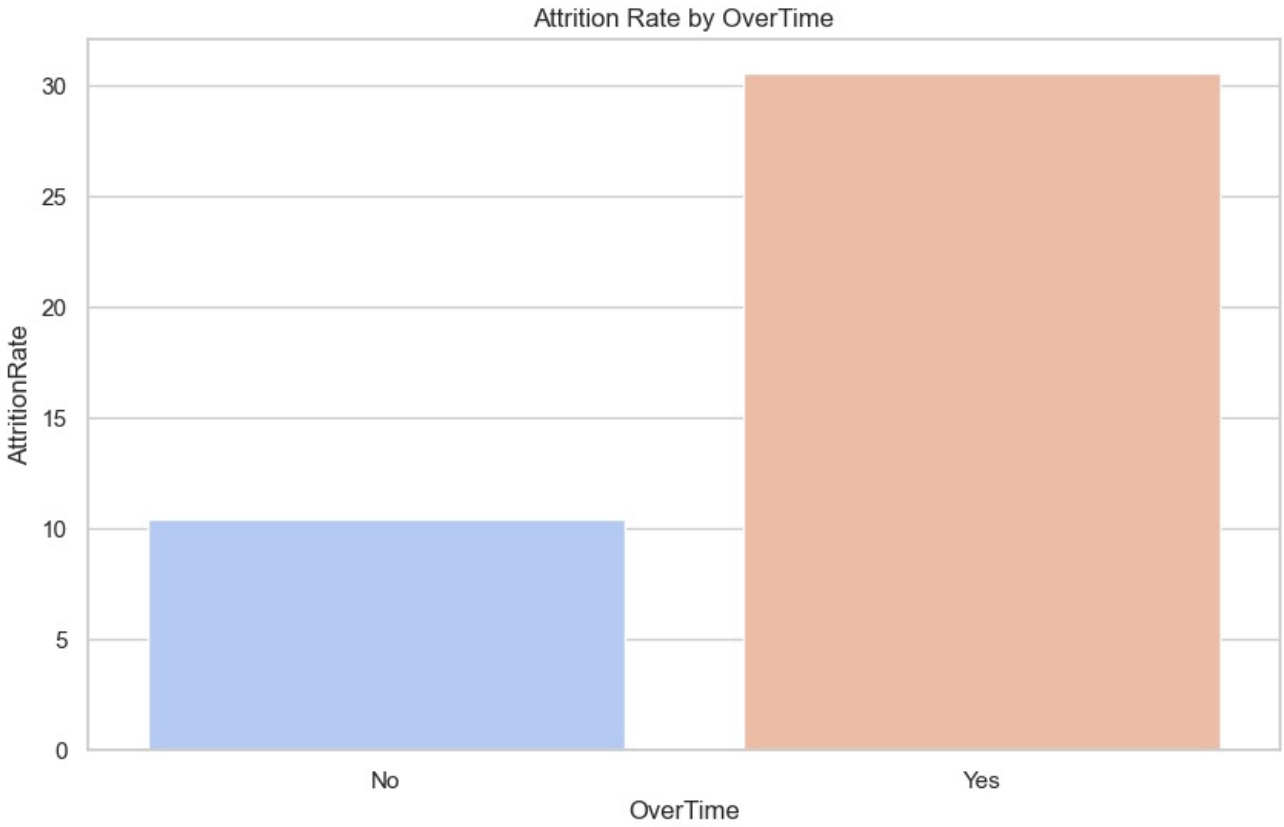


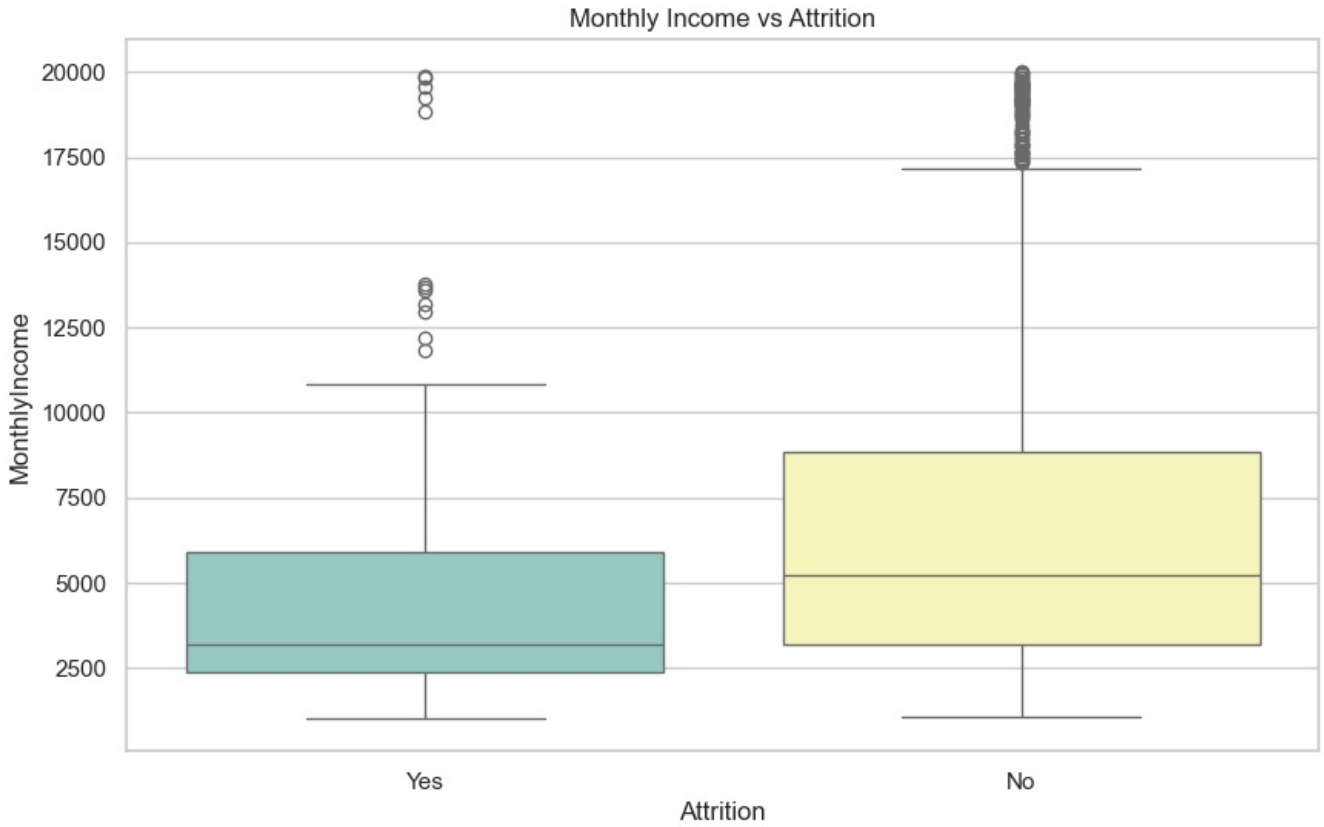
In [8]:
```
# Attrition by Age and Gender
df['Attrition_Flag'] = df['Attrition'].apply(lambda x: 1 if x == 'Yes' else 0)
sns.kdeplot(data=df[df['Attrition'] == 'Yes'], x='Age', fill=True, label='Attrition=Yes', color='salmon')
sns.kdeplot(data=df[df['Attrition'] == 'No'], x='Age', fill=True, label='Attrition=No', color='lightblue')
plt.title("Attrition by Age")
plt.legend()
plt.show()

def calculate_attrition_rate(df, column):
    attrition_counts = df.groupby([column, 'Attrition']).size().unstack(fill_value=0)
    attrition_rate = (attrition_counts['Yes'] / attrition_counts.sum(axis=1)) * 100
    return attrition_rate.reset_index().rename(columns={0: 'AttritionRate'})

gender_attr = calculate_attrition_rate(df, 'Gender')
sns.barplot(data=gender_attr, x='Gender', y='AttritionRate', palette='Set1')
plt.title("Attrition Rate by Gender")
plt.ylabel("Attrition Rate (%)")
plt.show()
```

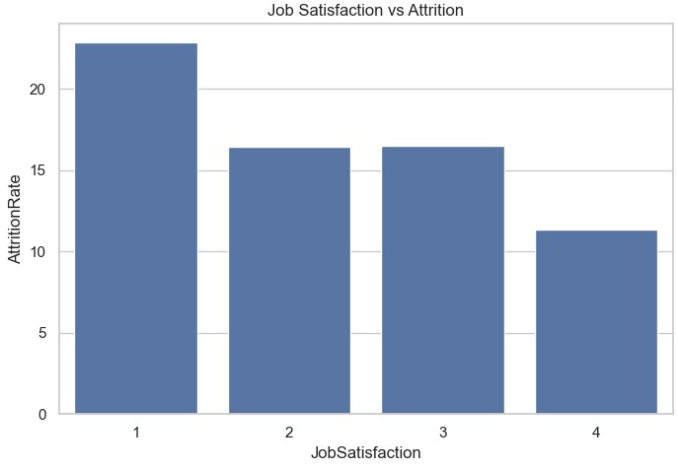


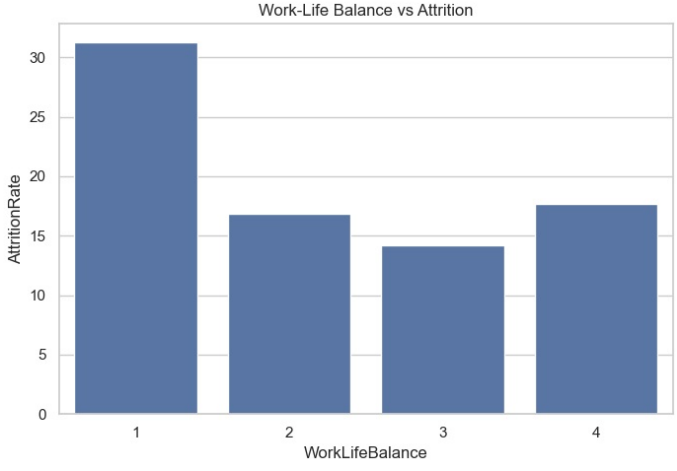


In [9]:
```
# Key Factors Analysis
overtime_attr = calculate_attrition_rate(df, 'OverTime')
sns.barplot(data=overtime_attr, x='OverTime', y='AttritionRate', palette='coolwarm')
plt.title("Attrition Rate by OverTime")
plt.show()

sns.boxplot(data=df, x='Attrition', y='MonthlyIncome', palette='Set3')
plt.title("Monthly Income vs Attrition")
plt.show()

satisfaction_attr = calculate_attrition_rate(df, 'JobSatisfaction')
wlb_attr = calculate_attrition_rate(df, 'WorkLifeBalance')

fig, axes = plt.subplots(1, 2, figsize=(14, 5))
sns.barplot(data=satisfaction_attr, x='JobSatisfaction', y='AttritionRate', ax=axes[0])
sns.barplot(data=wlb_attr, x='WorkLifeBalance', y='AttritionRate', ax=axes[1])
axes[0].set_title("Job Satisfaction vs Attrition")
axes[1].set_title("Work-Life Balance vs Attrition")
plt.tight_layout()
plt.show()
```

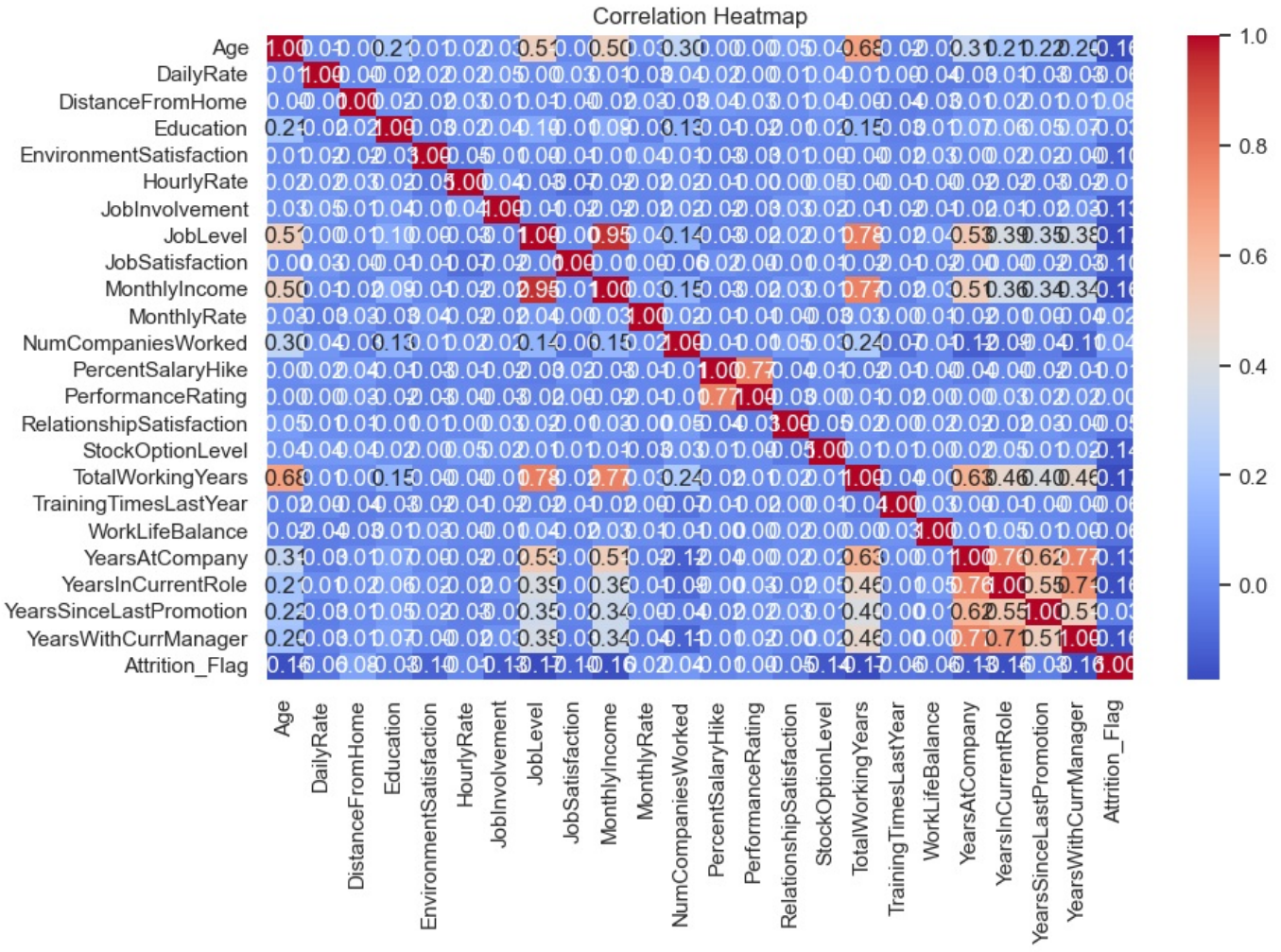

Attrition Rate by OverTime
AttritionRate
30
25
20
15
10
5
0
No
Yes
OverTime


Monthly Income vs Attrition
MonthlyIncome
20000
17500
15000
12500
10000
7500
5000
2500
Yes
No
Attrition


Job Satisfaction vs Attrition
AttritionRate
20
15
10
5
0
1
2
3
4
JobSatisfaction


Work-Life Balance vs Attrition
AttritionRate
30
25
20
15
10
5
0
1
2
3
4
WorkLifeBalance

In [10]:
```
# Correlation Heatmap
numeric_df = df.select_dtypes(include=['int64', 'float64'])
sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```



In [11]:
```
# Predictive Modeling
df_encoded = pd.get_dummies(df.drop('Attrition', axis=1), drop_first=True)
X = df_encoded.drop('Attrition_Flag', axis=1)
y = df_encoded['Attrition_Flag']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

In [12]:
```
# Evaluation
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

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

           0       0.88      0.97      0.92       255
           1       0.42      0.13      0.20        39

    accuracy                           0.86       294
   macro avg       0.65      0.55      0.56       294
weighted avg       0.82      0.86      0.83       294

Accuracy: 0.8605442176870748
Confusion Matrix:
 [[248   7]
 [ 34   5]]
```

In [ ]: