In [4]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, C


df = pd.read_csv("C:\\Users\hp\\Downloads\\WA_Fn-UseC_-HR-Employee-Attrition.csv")


df.drop(['EmployeeNumber', 'Over18', 'StandardHours', 'EmployeeCount'], axis=1, inplac

# Encode categorical variables
le = LabelEncoder()
for column in df.select_dtypes(include='object').columns:
    df[column] = le.fit_transform(df[column])

# Separate features and target
X = df.drop('Attrition', axis=1)
y = df['Attrition']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=

# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Logistic Regression
log_model = LogisticRegression()
log_model.fit(X_train, y_train)
log_pred = log_model.predict(X_test)

# Decision Tree
tree_model = DecisionTreeClassifier()
tree_model.fit(X_train, y_train)
tree_pred = tree_model.predict(X_test)

# Random Forest
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)


# Evaluation
print("Logistic Regression:\n", classification_report(y_test, log_pred))
print("Decision Tree:\n", classification_report(y_test, tree_pred))
print("Random Forest:\n", classification_report(y_test, rf_pred))
```

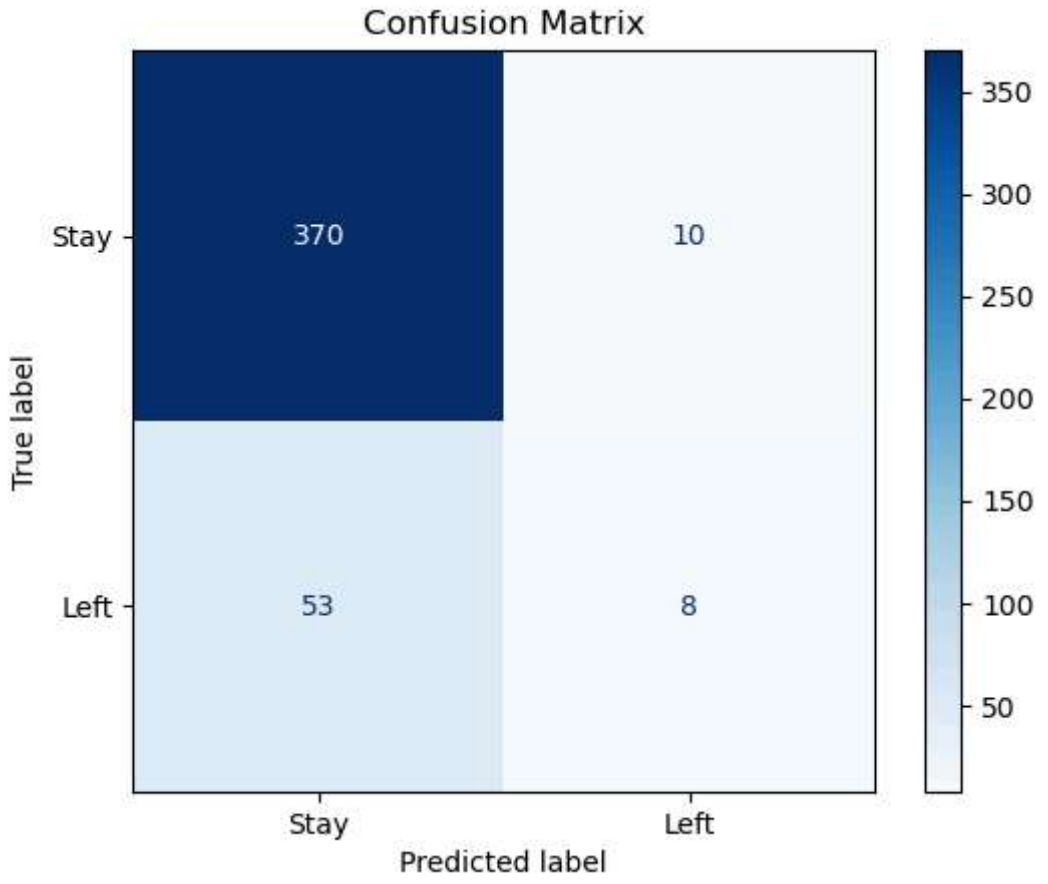
```
Logistic Regression:
              precision    recall  f1-score   support

           0       0.91      0.98      0.94       255
           1       0.70      0.36      0.47        39

    accuracy                           0.89       294
   macro avg       0.80      0.67      0.71       294
weighted avg       0.88      0.89      0.88       294

Decision Tree:
              precision    recall  f1-score   support

           0       0.87      0.89      0.88       255
           1       0.15      0.13      0.14        39

    accuracy                           0.79       294
   macro avg       0.51      0.51      0.51       294
weighted avg       0.77      0.79      0.78       294

Random Forest:
              precision    recall  f1-score   support

           0       0.88      1.00      0.93       255
           1       0.80      0.10      0.18        39

    accuracy                           0.88       294
   macro avg       0.84      0.55      0.56       294
weighted avg       0.87      0.88      0.83       294
```

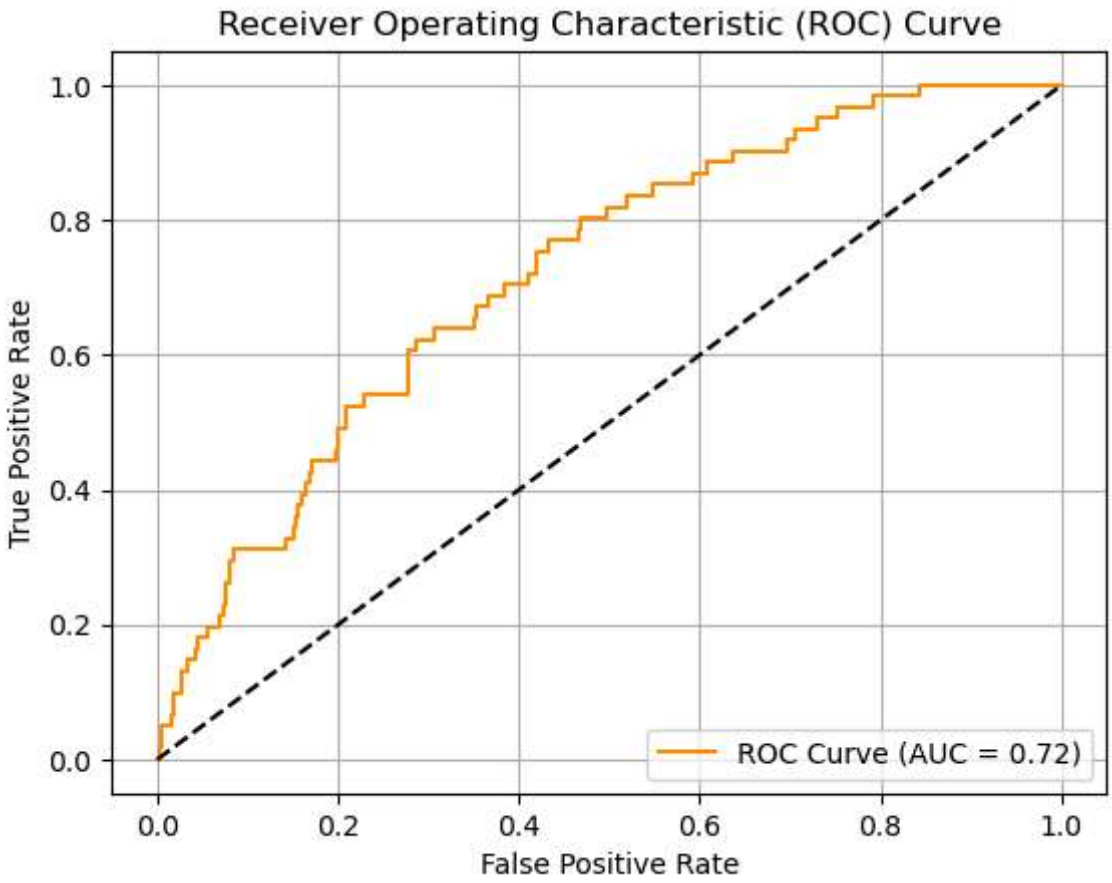
In [4]:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, roc_curve, auc
import matplotlib.pyplot as plt


df = pd.read_csv("C:\\Users\hp\\Downloads\\WA_Fn-UseC_-HR-Employee-Attrition.csv")


#  Encode Target Variable (Attrition)
df['Attrition'] = df['Attrition'].apply(lambda x: 1 if x == 'Yes' else 0)

#  Encode Categorical Columns
le = LabelEncoder()
for col in df.select_dtypes(include='object').columns:
    if col != 'Attrition':
        df[col] = le.fit_transform(df[col])

#  Split Features and Target
X = df.drop('Attrition', axis=1)
y = df['Attrition']

#  Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=

#  Train Logistic Regression Model
model = LogisticRegression(max_iter=1000)
```

```
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
y_proba = model.predict_proba(X_test)[:, 1]

#  Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Stay", "Left"])
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.show()

#  ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_proba)
roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label=f"ROC Curve (AUC = {roc_auc:.2f})", color="darkorange")
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operating Characteristic (ROC) Curve")
plt.legend(loc="lower right")
plt.grid(True)
plt.show()
```

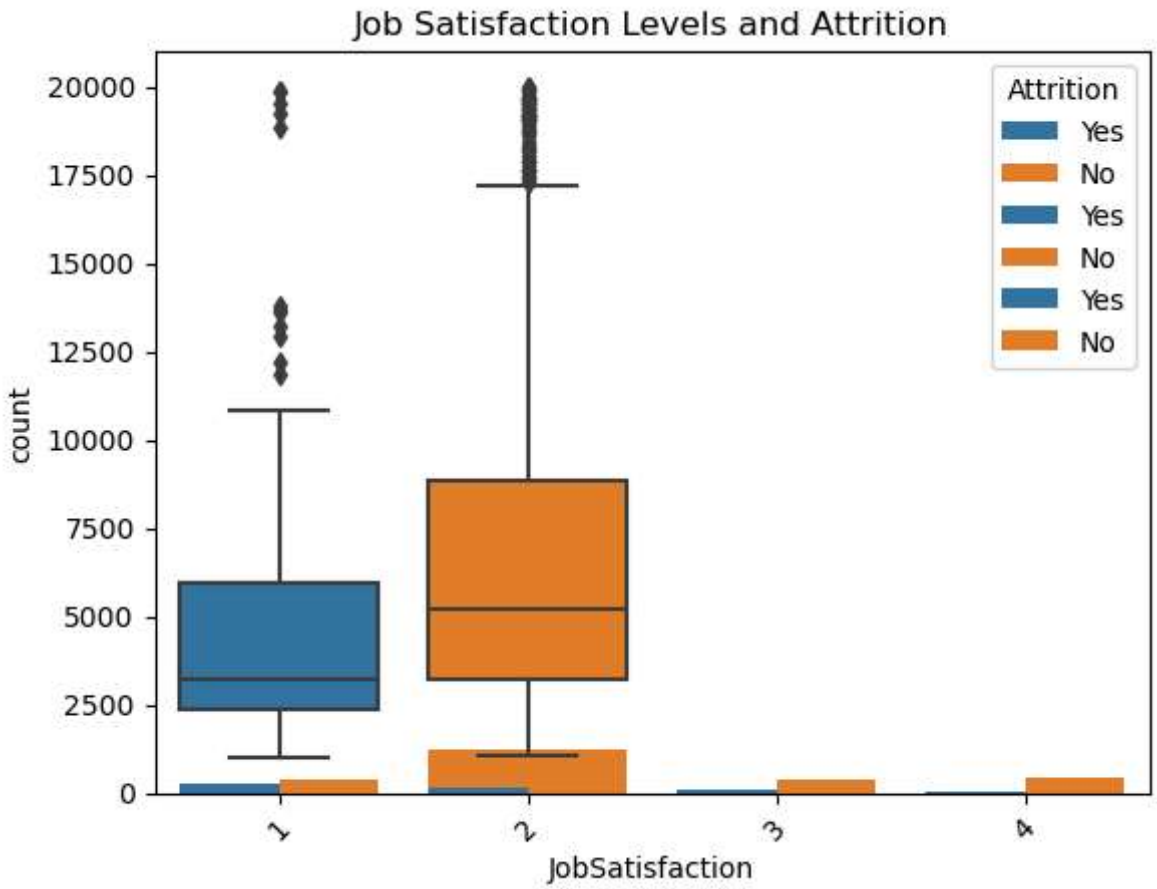
```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:460: Con
vergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```





In [3]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = df = pd.read_csv("C:\\Users\hp\\Downloads\\WA_Fn-UseC_-HR-Employee-Attrition.csv"

sns.countplot(data=df, x='Attrition')
plt.title("Distribution of Employee Attrition")
plt.xlabel("Attrition (1 = Yes, 0 = No)")
plt.ylabel("Number of Employees")
sns.countplot(data=df, x='Department', hue='Attrition')
plt.title("Attrition Count by Department")
plt.xticks(rotation=45)
sns.boxplot(x='Attrition', y='MonthlyIncome', data=df)
plt.title("Monthly Income and Attrition")
sns.countplot(data=df, x='OverTime', hue='Attrition')
plt.title("Attrition Based on Overtime")
sns.histplot(df[df['Attrition'] == 1]['Age'], kde=True, bins=10)
plt.title("Age Distribution of Employees Who Left")
sns.countplot(data=df, x='JobSatisfaction', hue='Attrition')
plt.title("Job Satisfaction Levels and Attrition")
```

Out[3]:
```
Text(0.5, 1.0, 'Job Satisfaction Levels and Attrition')
```


Job Satisfaction Levels and Attrition
Attrition
Yes
No
Yes
No
Yes
No
count
20000
17500
15000
12500
10000
7500
5000
2500
0
1
2
3
4
JobSatisfaction

In [ ]: