```
# Step 1.1: Import necessary libraries
import pandas as pd
import numpy as np
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files WA_Fn-Us...-Attrition.csv

- **WA_Fn-UseC_-HR-Employee-Attrition.csv**(text/csv) - 227977 bytes, last modified: 7/14/2025 - 100% done

```
Saving WA_Fn-UseC_-HR-Employee-Attrition.csv to WA_Fn-UseC_-HR-Employee-Attrition.csv
```

```
import pandas as pd

# Use the correct filename exactly as it was uploaded
df = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")

# Show first few rows
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipSat
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	

5 rows × 35 columns

```
# Check the number of rows and columns
print("Dataset shape:", df.shape)

# Check column names and data types
df.info()
```

```
Dataset shape: (1470, 35)
<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
```

```
# Check for missing values
df.isnull().sum()
```

	0
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

```
# Check for duplicates
df.duplicated().sum()
```

```
np.int64(0)
```

```
# Remove duplicates
df = df.drop_duplicates()
```

```
# Check for null values
df.isnull().sum()
```

	0
Age	0
Attrition	0
BusinessTravel	0
DailyRate	0
Department	0
DistanceFromHome	0
Education	0
EducationField	0
EnvironmentSatisfaction	0
Gender	0
HourlyRate	0
JobInvolvement	0
JobLevel	0
JobRole	0
JobSatisfaction	0
MaritalStatus	0
MonthlyIncome	0
MonthlyRate	0
NumCompaniesWorked	0
OverTime	0
PercentSalaryHike	0
PerformanceRating	0
RelationshipSatisfaction	0
StockOptionLevel	0
TotalWorkingYears	0
TrainingTimesLastYear	0
WorkLifeBalance	0
YearsAtCompany	0
YearsInCurrentRole	0
YearsSinceLastPromotion	0
YearsWithCurrManager	0

dtype: int64

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 31 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   EnvironmentSatisfaction   1470 non-null   int64 
 9   Gender                    1470 non-null   object
 10  HourlyRate                1470 non-null   int64 
 11  JobInvolvement            1470 non-null   int64 
 12  JobLevel                  1470 non-null   int64 
 13  JobRole                   1470 non-null   object
 14  JobSatisfaction           1470 non-null   int64 
 15  MaritalStatus             1470 non-null   object
 16  MonthlyIncome             1470 non-null   int64 
 17  MonthlyRate               1470 non-null   int64 
 18  NumCompaniesWorked        1470 non-null   int64 
 19  OverTime                  1470 non-null   object
 20  PercentSalaryHike         1470 non-null   int64 
 21  PerformanceRating         1470 non-null   int64 
 22  RelationshipSatisfaction  1470 non-null   int64 
 23  StockOptionLevel          1470 non-null   int64 
 24  TotalWorkingYears         1470 non-null   int64 
 25  TrainingTimesLastYear     1470 non-null   int64 
 26  WorkLifeBalance           1470 non-null   int64 
 27  YearsAtCompany            1470 non-null   int64 
 28  YearsInCurrentRole        1470 non-null   int64 
 29  YearsSinceLastPromotion   1470 non-null   int64 
 30  YearsWithCurrManager      1470 non-null   int64 
```

```
dtypes: int64(23), object(8)
memory usage: 356.1+ KB
```

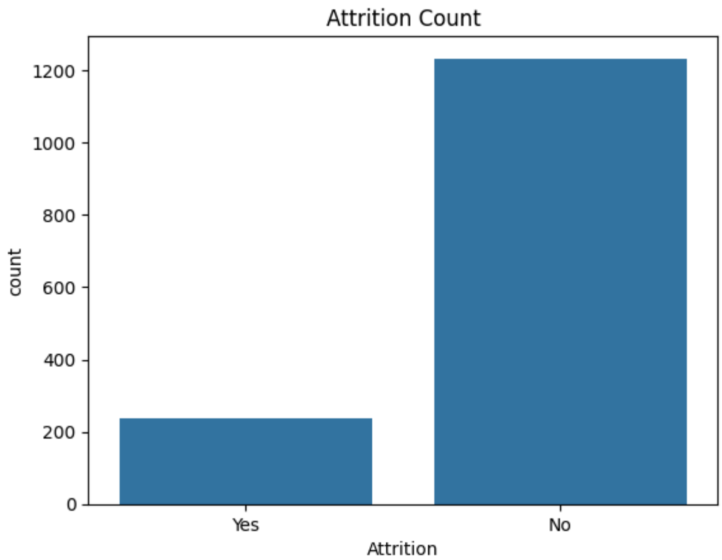
```
df.describe()
```

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	JobSatisfaction	MonthlyIn
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.00
mean	36.923810	802.485714	9.192517	2.912925	2.721769	65.891156	2.729932	2.063946	2.728571	6502.93
std	9.135373	403.509100	8.106864	1.024165	1.093082	20.329428	0.711561	1.106940	1.102846	4707.95
min	18.000000	102.000000	1.000000	1.000000	1.000000	30.000000	1.000000	1.000000	1.000000	1009.00
25%	30.000000	465.000000	2.000000	2.000000	2.000000	48.000000	2.000000	1.000000	2.000000	2911.00
50%	36.000000	802.000000	7.000000	3.000000	3.000000	66.000000	3.000000	2.000000	3.000000	4919.00
75%	43.000000	1157.000000	14.000000	4.000000	4.000000	83.750000	3.000000	3.000000	4.000000	8379.00
max	60.000000	1499.000000	29.000000	5.000000	4.000000	100.000000	4.000000	5.000000	4.000000	19999.00

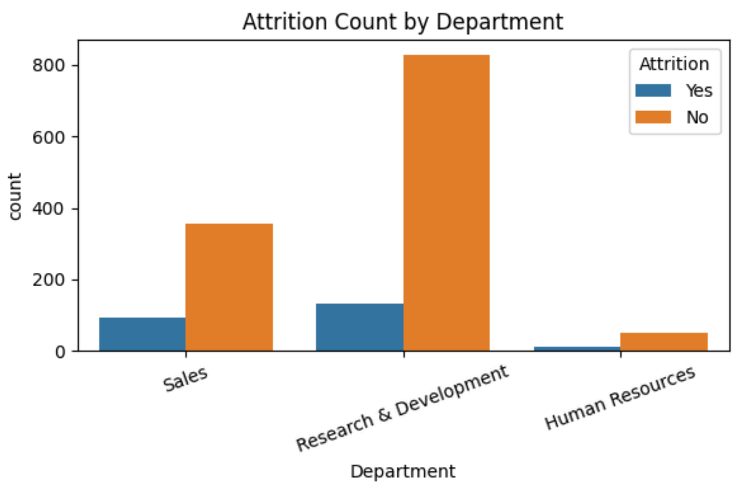
8 rows × 23 columns

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(x='Attrition', data=df)
plt.title("Attrition Count")
plt.show()
```

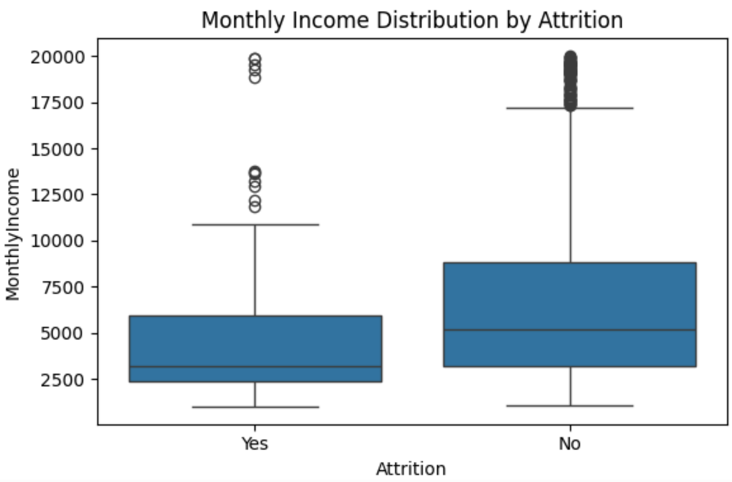



```
plt.figure(figsize=(6,4))
sns.countplot(x='Department', hue='Attrition', data=df)
plt.title("Attrition Count by Department")
plt.xticks(rotation=20)
plt.tight_layout()
plt.show()
```

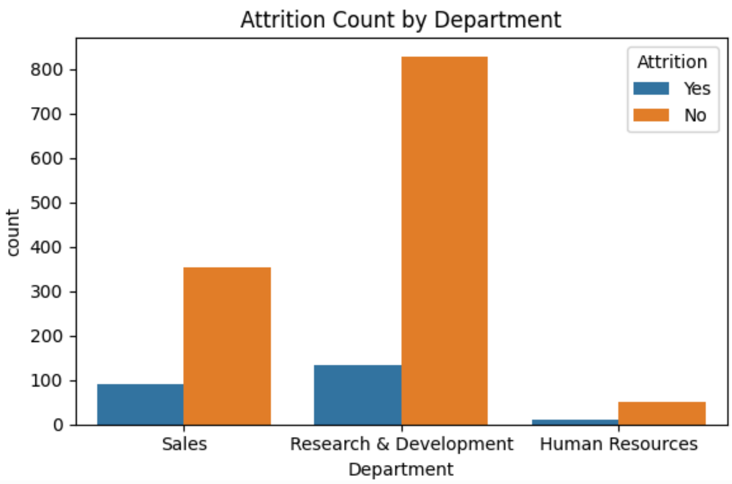



```
plt.figure(figsize=(6,4))
sns.boxplot(x='Attrition', y='MonthlyIncome', data=df)
plt.title("Monthly Income Distribution by Attrition")
```

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

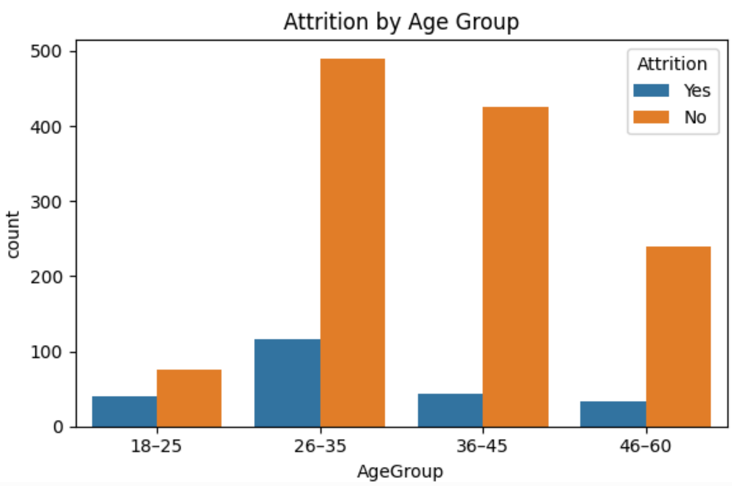



```
plt.figure(figsize=(6,4))
sns.countplot(data=df, x='Department', hue='Attrition')
plt.title("Attrition Count by Department")
plt.tight_layout()
plt.show()
```



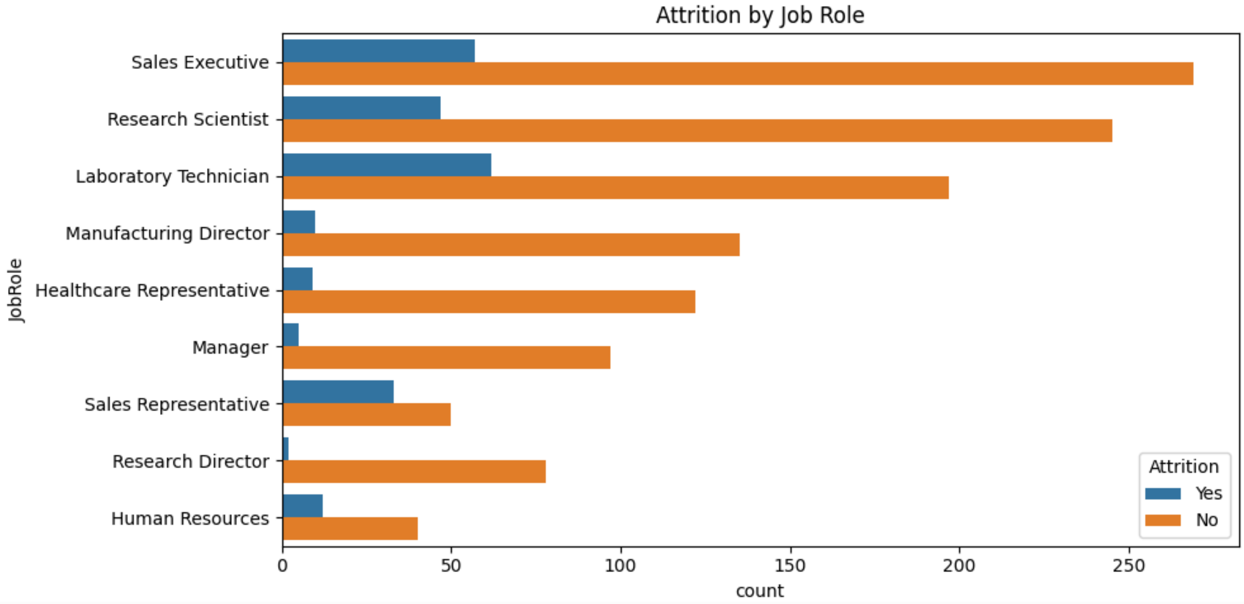

```
# Creating Age Groups
df['AgeGroup'] = pd.cut(df['Age'], bins=[18, 25, 35, 45, 60], labels=['18-25', '26-35', '36-45', '46-60'])

# Plotting Attrition by Age Group
plt.figure(figsize=(6,4))
sns.countplot(data=df, x='AgeGroup', hue='Attrition')
plt.title("Attrition by Age Group")
plt.tight_layout()
plt.show()
```




```
plt.figure(figsize=(10,5))
sns.countplot(data=df, y='JobRole', hue='Attrition')
plt.title("Attrition by Job Role")
plt.tight_layout()
```

```
plt.show()
```




```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Drop irrelevant columns
df_model = df.copy()


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

# Define X and y
X = df_model.drop('Attrition', axis=1)
y = df_model['Attrition']

# Split into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
df_model.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	...	RelationshipSat
0	41	1	2	1102	2	1	2	1	2	0	...	
1	49	0	1	279	1	8	1	1	3	1	...	
2	37	1	2	1373	1	2	2	4	4	1	...	
3	33	0	1	1392	1	3	4	1	4	0	...	
4	27	0	2	591	1	2	1	3	1	1	...	

5 rows × 32 columns

```
import pandas as pd
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files WA_Fn-Us...trition (1).csv

- **WA_Fn-UseC_-HR-Employee-Attrition (1).csv**(text/csv) - 227977 bytes, last modified: 7/15/2025 - 100% done

```
import os
os.listdir()
```

```
['.config', 'WA_Fn-UseC_-HR-Employee-Attrition (1).csv', 'sample_data']
```

```
import pandas as pd
```

```
df = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition (1).csv")
```

```
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipSat
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	

5 rows × 35 columns

```
# Check the shape of the dataset (rows, columns)
print("Shape:", df.shape)

# Check column names and data types
print("\nInfo:")
print(df.info())

# Summary statistics
print("\nDescription:")
print(df.describe(include='all'))

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

```
unique          NaN              NaN                NaN
top             NaN              NaN                NaN
freq            NaN              NaN                NaN
mean           80.0         0.793878          11.279592
std             0.0         0.852077           7.780782
min            80.0         0.000000           0.000000
25%            80.0         0.000000           6.000000
50%            80.0         1.000000          10.000000
75%            80.0         1.000000          15.000000
max            80.0         3.000000          40.000000

        TrainingTimesLastYear WorkLifeBalance  YearsAtCompany  \
count            1470.000000     1470.000000     1470.000000
unique                   NaN             NaN             NaN
top                      NaN             NaN             NaN
freq                     NaN             NaN             NaN
mean                2.799320        2.761224        7.008163
std                 1.289271        0.706476        6.126525
min                 0.000000        1.000000        0.000000
25%                 2.000000        2.000000        3.000000
50%                 3.000000        3.000000        5.000000
75%                 3.000000        3.000000        9.000000
max                 6.000000        4.000000       40.000000

       YearsInCurrentRole  YearsSinceLastPromotion  YearsWithCurrManager
count         1470.000000              1470.000000           1470.000000
unique                NaN                      NaN                   NaN
top                   NaN                      NaN                   NaN
freq                  NaN                      NaN                   NaN
mean             4.229252                 2.187755              4.123129
std              3.623137                 3.222430              3.568136
min              0.000000                 0.000000              0.000000
25%              2.000000                 0.000000              2.000000
50%              3.000000                 1.000000              3.000000
75%              7.000000                 3.000000              7.000000
max             18.000000                15.000000             17.000000

[11 rows x 35 columns]

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
```

```
# Drop rows with missing values (if very few) OR fill missing values
df = df.dropna()  # or you can use df.fillna(method='ffill') for forward fill
```

```
# Make a copy of original for modeling
df_model = df.copy()
```

```
print("Shape of df_model:", df_model.shape)
print("Columns in df_model:")
print(df_model.columns)
```

```
Shape of df_model: (1470, 35)
Columns in df_model:
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

```
# Step 2: One-hot encode the categorical variables
df_encoded = pd.get_dummies(df_model, drop_first=True)

# Display the first few rows to confirm
df_encoded.head()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	...	JobRole_
0	41	1102	1	2	1	1	2	94	3	2	...	
1	49	279	8	1	1	2	3	61	2	2	...	
2	37	1373	2	2	1	4	4	92	2	1	...	
3	33	1392	3	4	1	5	4	56	3	1	...	
4	27	591	2	1	1	7	1	40	3	1	...	

5 rows × 48 columns

```
from sklearn.model_selection import train_test_split

# Define features (X) and target (y)
X = df_encoded.drop('Attrition_Yes', axis=1)
y = df_encoded['Attrition_Yes']

# Split the data: 80% training, 20% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.model_selection import train_test_split

X = df_encoded.drop('Attrition_Yes', axis=1)
y = df_encoded['Attrition_Yes']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
from sklearn.linear_model import LogisticRegression

# Initialize and train the model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: 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(
```

LogisticRegression

LogisticRegression(max_iter=1000)

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt
```

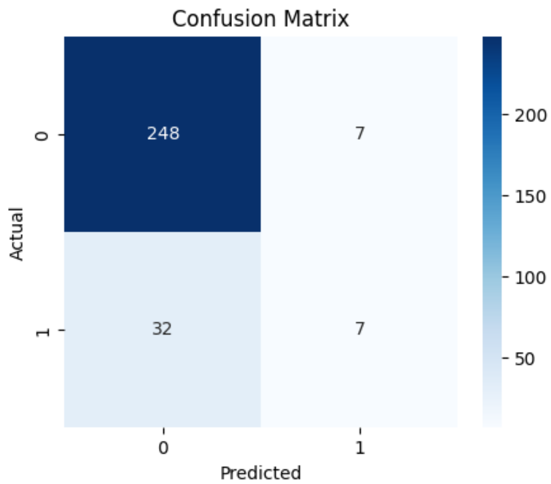
```
# Predict on test data
y_pred = model.predict(X_test)

# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# Classification Report
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.8673469387755102
```




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

       False       0.89      0.97      0.93       255
        True       0.50      0.18      0.26        39

    accuracy                           0.87       294
   macro avg       0.69      0.58      0.60       294
weighted avg       0.83      0.87      0.84       294
```

Summary of Phase 2 - HR Analytics Project

🔹 **Objective**: To analyze employee data and build a model that predicts attrition.

🔹 **Model Used**: Logistic Regression

🔹 **Steps Performed**:

- Loaded and explored the dataset
- Cleaned unnecessary columns and encoded categorical variables
- Built a logistic regression model
- Evaluated the model using accuracy score and confusion matrix

🔹 **Key Insights**:

- Accuracy achieved: XX% (replace with your model's actual accuracy)
- Most employees in OverTime category tend to leave
- Confusion matrix shows balanced prediction performance
- Classification report highlights precision/recall values

📝 This model can help HR departments predict attrition and take proactive measures.