+ Code + Text

```
import pandas as pd
```

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

```
df.head()
```

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

5 rows × 35 columns

```
df.describe
```

```
pandas.core.generic.NDFrame.describe
def describe(percentiles=None, include=None, exclude=None) -> Self

Generate descriptive statistics.

Descriptive statistics include those that summarize the central
tendency, dispersion and shape of a
dataset's distribution, excluding ``NaN`` values.
```

Data Cleaning

```
df.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
```

```
df = df.drop(['EmployeeCount', 'StandardHours', 'Over18', 'EmployeeNumber'], axis=1)
df['Attrition'] = df['Attrition'].map({'Yes': 1, 'No': 0})
```

EDA

```
df.shape
```

```
(1470, 31)
```

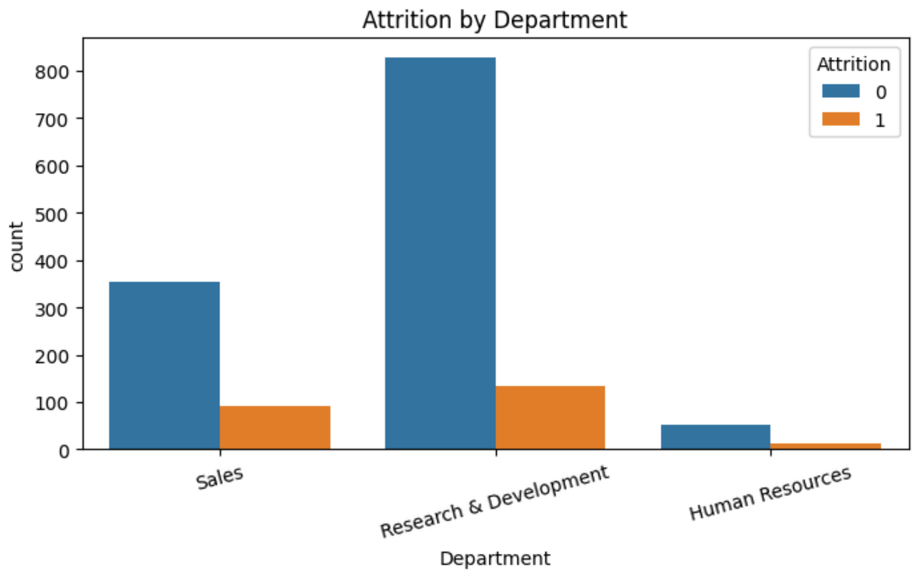
```
df['Attrition'].value_counts()
```

Attrition	count
0	1233
1	237

dtype: int64

Department-Wise Attrition

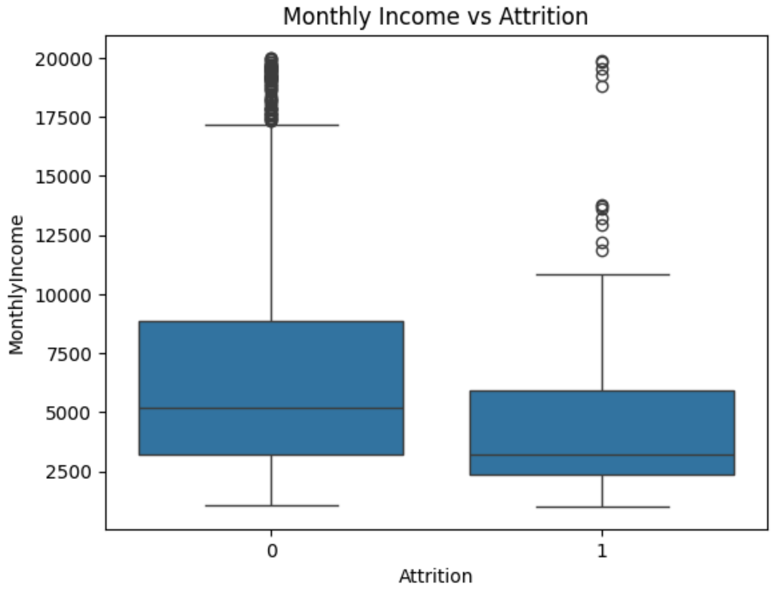
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8,4))
sns.countplot(data=df, x='Department', hue='Attrition')
plt.title("Attrition by Department")
plt.xticks(rotation=15)
plt.show()
```




```
#Sales and Research&Development departments have the highest attrition, with Sales having a higher relative attrition rate
```

Monthly Income vs Attrition

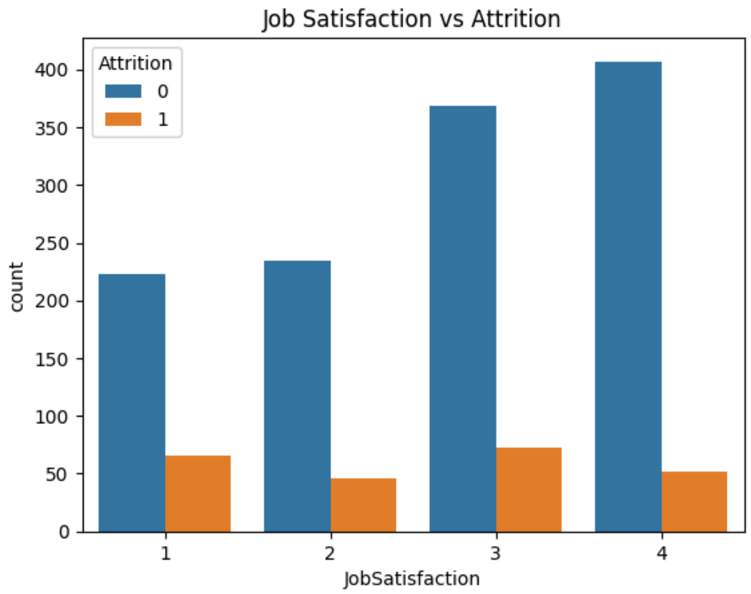
```
sns.boxplot(data=df, x='Attrition', y='MonthlyIncome')
plt.title("Monthly Income vs Attrition")
plt.show()
```




```
#People who earn less salary are more likely to leave the company
```

Job Satisfaction vs Attrition

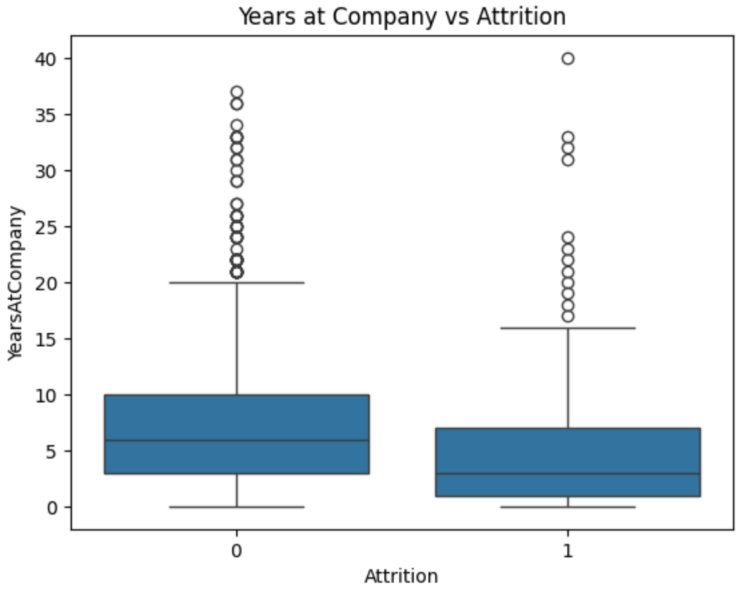
```
sns.countplot(data=df, x='JobSatisfaction', hue='Attrition')
plt.title("Job Satisfaction vs Attrition")
plt.show()
```




```
#Employees with low job satisfaction (levels 1 and 2) are more likely to resign.
#Employees with high job satisfaction (levels 3 and 4) usually stay.
```

Years at Company vs Attrition

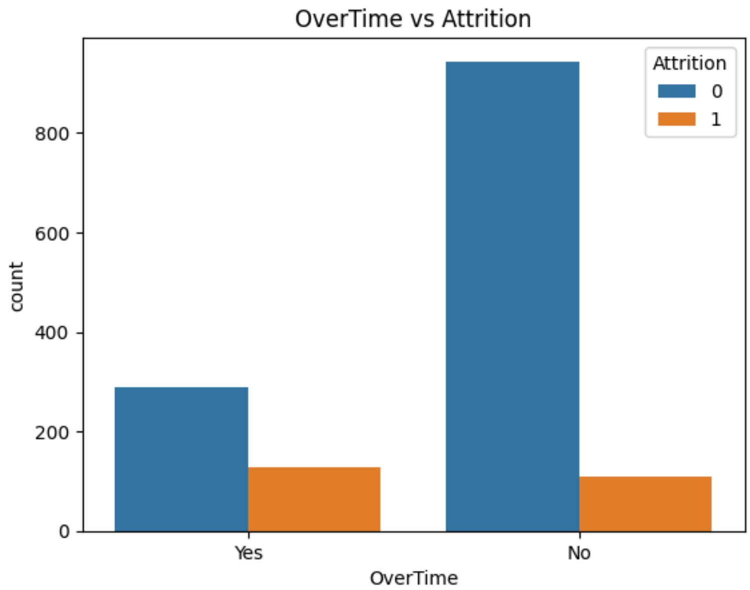
```
sns.boxplot(data=df, x='Attrition', y='YearsAtCompany')
plt.title("Years at Company vs Attrition")
plt.show()
```




```
#Employees are more likely to leave the company within their first few years.
```

OverTime vs Attrition

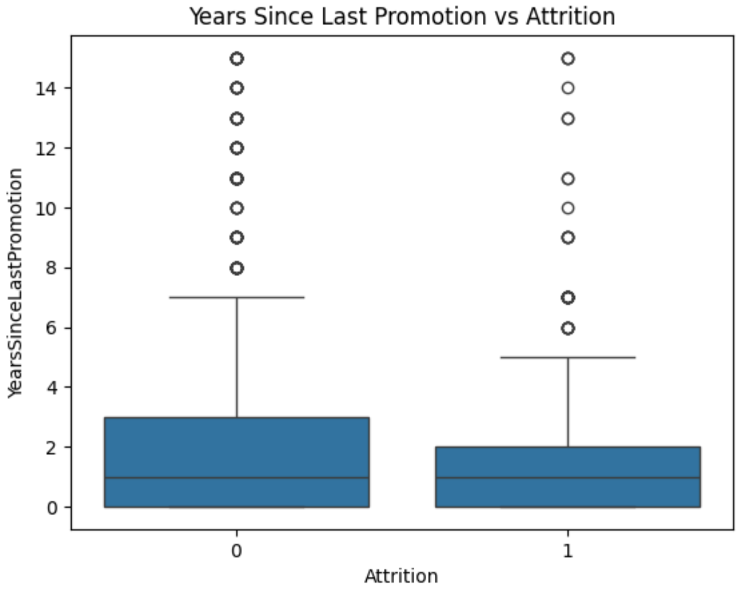
```
sns.countplot(data=df, x='OverTime', hue='Attrition')
plt.title("OverTime vs Attrition")
plt.show()
```




```
#Employees who work overtime are much more likely to resign and Employees who don't work overtime usually stay.
```

Promotions vs Attrition

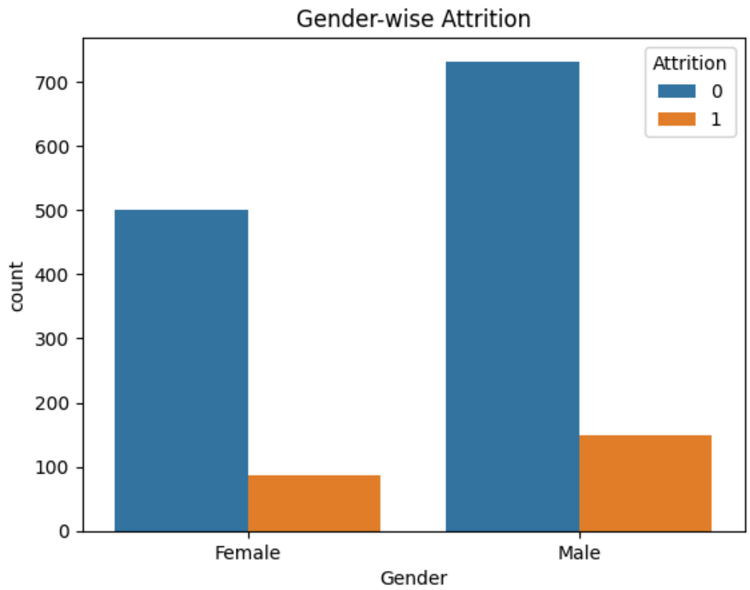
```
sns.boxplot(data=df, x='Attrition', y='YearsSinceLastPromotion')
plt.title("Years Since Last Promotion vs Attrition")
plt.show()
```




```
#The boxplot shows that many employees who left had not received a promotion in recent years. While the difference is not huge, timely p
```

Gender vs Attrition

```
sns.countplot(data=df, x='Gender', hue='Attrition')
plt.title("Gender-wise Attrition")
plt.show()
```




```
#More men left the company than women.
```

MODELLING

```
from sklearn.preprocessing import LabelEncoder

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

X = df.drop('Attrition', axis=1)
y = df['Attrition']
```

Logistic Regression Model

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

```
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)

print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.8775510204081632
Confusion Matrix:
 [[252   3]
 [ 33   6]]
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=
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(
```

** Use SHAP for Explainability**

```
import shap

explainer = shap.Explainer(model, X_test)
shap_values = explainer(X_test)

shap.summary_plot(shap_values, X_test)
```