Employee Attrition Analysis and Prediction

CONTENTS :

EDA

1. Data Exploration.
2. Data Cleaning.
3. Data Encoding.
4. Data Labelling.

Importing librarys

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
```

You've imported the necessary libraries for data manipulation (pandas), numerical operations (numpy), and data visualization (matplotlib.pyplot and seaborn). These libraries provide various functions and tools to work with data efficiently and visualize it effectively.

In [2]:
```
# Read the dataset
df = pd.read_csv(r'F:\Technocolabs\WA_Fn-UseC_-HR-Employee-Attrition.csv')
```

In [3]:
```
df
```

Out[3]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education
0	41	Yes	Travel_Rarely	1102	Sales	1	2
1	49	No	Travel_Frequently	279	Research & Development	8	1
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2
3	33	No	Travel_Frequently	1392	Research & Development	3	4
4	27	No	Travel_Rarely	591	Research & Development	2	1
...	...	...	...	...	...	...	...
1465	36	No	Travel_Frequently	884	Research & Development	23	2
1466	39	No	Travel_Rarely	613	Research & Development	6	1
1467	27	No	Travel_Rarely	155	Research & Development	4	3
1468	49	No	Travel_Frequently	1023	Sales	2	3
1469	34	No	Travel_Rarely	628	Research & Development	8	3

1470 rows × 35 columns

EDA

Data Cleaning.

In [4]:
```
df.head()
```

Out[4]:

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

5 rows × 35 columns

In [27]:
```
df.tail()
```

Out[27]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education
1465	36	No	Travel_Frequently	884	Research & Development	23	2
1466	39	No	Travel_Rarely	613	Research & Development	6	1
1467	27	No	Travel_Rarely	155	Research & Development	4	3
1468	49	No	Travel_Frequently	1023	Sales	2	3
1469	34	No	Travel_Rarely	628	Research & Development	8	3

5 rows × 35 columns

In [5]:
```
df.shape
```

Out[5]:
```
(1470, 35)
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRat
e',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorke
d',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

In [7]:
```
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
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
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 [10]:
```
df.describe()
```

Out[10]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	Employee
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024
std	9.135373	403.509100	8.106864	1.024165	0.0	602
min	18.000000	102.000000	1.000000	1.000000	1.0	1
25%	30.000000	465.000000	2.000000	2.000000	1.0	491
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068

8 rows × 26 columns

In [28]:

```
df.dropna(inplace=True)
print(df)
```

```
      Age Attrition     BusinessTravel  DailyRate              Department  
\
0      41       Yes      Travel_Rarely       1102                   Sales   
1      49        No  Travel_Frequently        279  Research & Development   
2      37       Yes      Travel_Rarely       1373  Research & Development   
3      33        No  Travel_Frequently       1392  Research & Development   
4      27        No      Travel_Rarely        591  Research & Development   
...   ...       ...                ...        ...                     ...   
1465   36        No  Travel_Frequently        884  Research & Development   
1466   39        No      Travel_Rarely        613  Research & Development   
1467   27        No      Travel_Rarely        155  Research & Development   
1468   49        No  Travel_Frequently       1023                   Sales   
1469   34        No      Travel_Rarely        628  Research & Development   

      DistanceFromHome  Education EducationField  EmployeeCount  \
0                    1          2  Life Sciences              1   
1                    8          1  Life Sciences              1   
2                    2          2          Other              1   
3                    3          4  Life Sciences              1   
4                    2          1        Medical              1   
...                ...        ...            ...            ...   
1465                23          2        Medical              1   
1466                 6          1        Medical              1   
1467                 4          3  Life Sciences              1   
1468                 2          3        Medical              1   
1469                 8          3        Medical              1   

      EmployeeNumber  ...  RelationshipSatisfaction StandardHours  \
0                  1  ...                         1            80   
1                  2  ...                         4            80   
2                  4  ...                         2            80   
3                  5  ...                         3            80   
4                  7  ...                         4            80   
...              ...  ...                       ...           ...   
1465            2061  ...                         3            80   
1466            2062  ...                         1            80   
1467            2064  ...                         2            80   
1468            2065  ...                         4            80   
1469            2068  ...                         1            80   

      StockOptionLevel  TotalWorkingYears  TrainingTimesLastYear  \
0                    0                  8                      0   
1                    1                 10                      3   
2                    0                  7                      3   
3                    0                  8                      3   
4                    1                  6                      3   
...                ...                ...                    ...   
1465                 1                 17                      3   
1466                 1                  9                      5   
1467                 1                  6                      0   
1468                 0                 17                      3   
1469                 0                  6                      3   

     WorkLifeBalance  YearsAtCompany YearsInCurrentRole  \
0                  1               6                  4   
1                  3              10                  7   
2                  3               0                  0   
3                  3               8                  7   
4                  3               2                  2   
...              ...             ...                ...   
1465               3               5                  2   
```

```
1466                3               7                  7
1467                3               6                  2
1468                2               9                  6
1469                4               4                  3

      YearsSinceLastPromotion  YearsWithCurrManager
0                           0                     5
1                           1                     7
2                           0                     0
3                           3                     0
4                           2                     2
...                       ...                   ...
1465                        0                     3
1466                        1                     7
1467                        0                     3
1468                        0                     8
1469                        1                     2

[1470 rows x 35 columns]
```

In [29]:
```python
# Check the data types after conversion
print(df.dtypes)
```

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

Ensuring the dataset is clean and ready for analysis is crucial. You've checked for missing values using df.isnull().sum() and found that there are no missing values in the dataset. This suggests that there is no need to handle missing data.

Data Exploration:

In [11]:
```
# Value counts for categorical variables
print(df['MonthlyRate'].value_counts())
```

```
4223     3
9150     3
9558     2
12858    2
22074    2
        ..
14561    1
2671     1
5718     1
11757    1
10228    1
Name: MonthlyRate, Length: 1427, dtype: int64
```

In [12]:
```
print(df['DailyRate'].value_counts())
```

```
691     6
408     5
530     5
1329    5
1082    5
       ..
650     1
279     1
316     1
314     1
628     1
Name: DailyRate, Length: 886, dtype: int64
```

In [13]:
```
print(df['BusinessTravel'].value_counts())
```

```
Travel_Rarely        1043
Travel_Frequently     277
Non-Travel            150
Name: BusinessTravel, dtype: int64
```

In [14]:
```
print(df['Department'].value_counts())
```

```
Research & Development    961
Sales                     446
Human Resources            63
Name: Department, dtype: int64
```

In [15]:
```
print(df['EducationField'].value_counts())
```

```
Life Sciences       606
Medical             464
Marketing           159
Technical Degree    132
Other                82
Human Resources      27
Name: EducationField, dtype: int64
```

In [16]:
```
print(df['Gender'].value_counts())
```

```
Male      882
Female    588
Name: Gender, dtype: int64
```

In [17]:
```
print(df['JobRole'].value_counts())
```

```
Sales Executive              326
Research Scientist           292
Laboratory Technician        259
Manufacturing Director       145
Healthcare Representative    131
Manager                      102
Sales Representative          83
Research Director             80
Human Resources               52
Name: JobRole, dtype: int64
```

In [18]:
```
print(df['MaritalStatus'].value_counts())
```

```
Married     673
Single      470
Divorced    327
Name: MaritalStatus, dtype: int64
```

In [19]:
```
print(df['Over18'].value_counts())
```

```
Y    1470
Name: Over18, dtype: int64
```

In [20]:
```
print(df['OverTime'].value_counts())
```

```
No     1054
Yes     416
Name: OverTime, dtype: int64
```

This involves getting a general understanding of the dataset. You've used df.shape to check the dimensions (number of rows and columns) of the dataset, df.columns to see the column names, and df.head() to display the first few rows of the dataset. These steps help you understand the structure and contents of the data.

Data Encoding

In [31]:
```
if 'categorical_column' in df.columns:
    # One-hot encoding
    encoded_df = pd.get_dummies(df, columns=['categorical_column'])
    print(encoded_df.head())
else:
    print("The column 'categorical_column' does not exist in the DataFrame.
```

```
The column 'categorical_column' does not exist in the DataFrame. Please pr
ovide the correct column name.
```

In [32]:
```
# One-hot encoding
encoded_df = pd.get_dummies(df, columns=['Department'])
```

In [33]:
```
# Label encoding
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
df['encoded_column'] = label_encoder.fit_transform(df['Department'])
```

In [34]:
```
# Specify the list of categorical columns you want to one-hot encode
categorical_columns = ['BusinessTravel', 'Department', 'EducationField', 'G

# One-hot encoding
encoded_df = pd.get_dummies(df, columns=categorical_columns)
print(encoded_df.head())
```

```
   Age Attrition  DailyRate  DistanceFromHome  Education  EmployeeCount  \
0   41       Yes       1102                 1          2              1
1   49        No        279                 8          1              1
2   37       Yes       1373                 2          2              1
3   33        No       1392                 3          4              1
4   27        No        591                 2          1              1

   EmployeeNumber  EnvironmentSatisfaction  HourlyRate  JobInvolvement
...  \
0               1                        2          94               3
...
1               2                        3          61               2
...
2               4                        4          92               2
...
3               5                        4          56               3
...
4               7                        1          40               3
...

   JobRole_Research Director  JobRole_Research Scientist  \
0                          0                           0
1                          0                           1
2                          0                           0
3                          0                           1
4                          0                           0

   JobRole_Sales Executive  JobRole_Sales Representative  \
0                        1                             0
1                        0                             0
2                        0                             0
3                        0                             0
4                        0                             0

   MaritalStatus_Divorced  MaritalStatus_Married  MaritalStatus_Single  \
0                       0                      0                     1
1                       0                      1                     0
2                       0                      0                     1
3                       0                      1                     0
4                       0                      1                     0

   Over18_Y  OverTime_No  OverTime_Yes
0         1            0             1
1         1            1             0
2         1            0             1
3         1            0             1
4         1            1             0

[5 rows x 57 columns]
```

This encoding technique transforms categorical variables into a format suitable for machine learning algorithms, facilitating the analysis and modeling of categorical data.

In [35]:
```
from sklearn.preprocessing import LabelEncoder

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Encode a specific column
df['Attrition_encoded'] = label_encoder.fit_transform(df['Attrition'])
print(df[['Attrition', 'Attrition_encoded']].head())
```

```
  Attrition  Attrition_encoded
0       Yes                  1
1        No                  0
2       Yes                  1
3        No                  0
4        No                  0
```

The provided code utilizes the LabelEncoder from scikit-learn to encode the 'Attrition' column in the DataFrame:

Label Encoding: The LabelEncoder is initialized to transform categorical labels into numerical values.

Encoding Process: The 'Attrition' column is encoded using the fit_transform method of the LabelEncoder, which assigns numerical labels to the categories.

Output: The code prints the first few rows of the DataFrame with both the original 'Attrition' column and the newly encoded 'Attrition_encoded' column.

This encoding process converts categorical data into a format suitable for machine learning algorithms that require numerical input, enabling further analysis and modeling.

In [36]:
```
from sklearn.preprocessing import MinMaxScaler

# Initialize MinMaxScaler
scaler = MinMaxScaler()

# Scale numerical features
numerical_columns = ['Age', 'DailyRate', 'DistanceFromHome', 'HourlyRate',
df[numerical_columns] = scaler.fit_transform(df[numerical_columns])
```

In [37]:
```
from sklearn.model_selection import train_test_split

# Split data into features (X) and target variable (y)
X = df.drop(columns=['Attrition'])
y = df['Attrition']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

Encoding categorical variables into numerical format is necessary for many machine learning algorithms. However, in the code you provided, it seems like the dataset doesn't contain categorical variables that need encoding. If there were categorical variables, you might use techniques like one-hot encoding or label encoding to convert them into numerical format.

Data Labelling

In [53]:
```
def transform(feature):
    le=LabelEncoder()
    df[feature]=le.fit_transform(df[feature])
    print(le.classes_)
```

In [54]:
```
cat_df=df.select_dtypes(include='object')
cat_df.columns
```

Out[54]:
```
Index(['Attrition', 'BusinessTravel', 'Department', 'EducationField', 'Gen
der',
       'JobRole', 'MaritalStatus', 'Over18', 'OverTime', 'label',
       'Attrition_Label', 'Income_Label'],
      dtype='object')
```

In [55]:
```python
for col in cat_df.columns:
    transform(col)
```

```
['No' 'Yes']
['Non-Travel' 'Travel_Frequently' 'Travel_Rarely']
['Human Resources' 'Research & Development' 'Sales']
['Human Resources' 'Life Sciences' 'Marketing' 'Medical' 'Other'
 'Technical Degree']
['Female' 'Male']
['Healthcare Representative' 'Human Resources' 'Laboratory Technician'
 'Manager' 'Manufacturing Director' 'Research Director'
 'Research Scientist' 'Sales Executive' 'Sales Representative']
['Divorced' 'Married' 'Single']
['Y']
['No' 'Yes']
['label_A' 'label_B' 'label_C']
['High Attrition' 'Low Attrition']
['High Income' 'Low Income']
```

In [38]:

```python
# Replace 'your_dataset.csv' with the actual path to your dataset file
df = pd.read_csv(r'F:\Technocolabs\WA_Fn-UseC_-HR-Employee-Attrition.csv')

# Define your condition and label accordingly
def label_function(row):
    if row['Age'] > 30:
        return 'label_A'
    else:
        return 'label_B'

# Apply the label function to each row
df['label'] = df.apply(label_function, axis=1)

# Display the first few rows to verify the labeling
print(df.head())
```

```
   Age Attrition     BusinessTravel  DailyRate              Department  \
0   41       Yes      Travel_Rarely       1102                   Sales
1   49        No  Travel_Frequently        279  Research & Development
2   37       Yes      Travel_Rarely       1373  Research & Development
3   33        No  Travel_Frequently       1392  Research & Development
4   27        No      Travel_Rarely        591  Research & Development

   DistanceFromHome  Education EducationField  EmployeeCount  EmployeeNumb
er  \
0                 1          2  Life Sciences              1
1
1                 8          1  Life Sciences              1
2
2                 2          2          Other              1
4
3                 3          4  Life Sciences              1
5
4                 2          1        Medical              1
7

   ...  StandardHours StockOptionLevel  TotalWorkingYears  \
0  ...             80                0                  8
1  ...             80                1                 10
2  ...             80                0                  7
3  ...             80                0                  8
4  ...             80                1                  6

   TrainingTimesLastYear  WorkLifeBalance YearsAtCompany  YearsInCurrentRo
le  \
0                      0                1              6
4
1                      3                3             10
7
2                      3                3              0
0
3                      3                3              8
7
4                      3                3              2
2

  YearsSinceLastPromotion  YearsWithCurrManager    label
0                       0                     5  label_A
1                       1                     7  label_A
2                       0                     0  label_A
3                       3                     0  label_A
4                       2                     2  label_B

[5 rows x 36 columns]
```

This labeling process enables segmentation and analysis of the dataset based on age categories, providing insights into workforce demographics and potential age-related patterns or trends.

In [39]:
```
# Define your conditions and labels accordingly
def label_function(row):
    if row['Age'] > 30 and row['Department'] == 'Sales':
        return 'label_A'
    # Example: If DailyRate is less than 500 and Education is greater than 3
    elif row['DailyRate'] < 500 and row['Education'] > 3:
        return 'label_B'
    # Add more conditions and labels as needed
    else:
        return 'label_C'  # Default label if none of the conditions are met

# Apply the label function to each row
df['label'] = df.apply(label_function, axis=1)

# Display the first few rows to verify the labeling
print(df.head())
```

```
   Age Attrition     BusinessTravel  DailyRate              Department  \
0   41       Yes      Travel_Rarely       1102                   Sales
1   49        No  Travel_Frequently        279  Research & Development
2   37       Yes      Travel_Rarely       1373  Research & Development
3   33        No  Travel_Frequently       1392  Research & Development
4   27        No      Travel_Rarely        591  Research & Development

   DistanceFromHome  Education EducationField  EmployeeCount  EmployeeNumb
er  \
0                 1          2  Life Sciences              1
1
1                 8          1  Life Sciences              1
2
2                 2          2          Other              1
4
3                 3          4  Life Sciences              1
5
4                 2          1        Medical              1
7

   ...  StandardHours StockOptionLevel  TotalWorkingYears  \
0  ...             80                0                  8
1  ...             80                1                 10
2  ...             80                0                  7
3  ...             80                0                  8
4  ...             80                1                  6

   TrainingTimesLastYear  WorkLifeBalance YearsAtCompany  YearsInCurrentRo
le  \
0                      0                1              6
4
1                      3                3             10
7
2                      3                3              0
0
3                      3                3              8
7
4                      3                3              2
2

  YearsSinceLastPromotion  YearsWithCurrManager    label
0                       0                     5  label_A
1                       1                     7  label_C
2                       0                     0  label_C
3                       3                     0  label_C
4                       2                     2  label_C

[5 rows x 36 columns]
```

The labeling function categorizes employees into different groups based on age, department, daily rate, and education level, allowing for insights into specific employee demographics and characteristics. This segmentation can aid in identifying patterns or trends within the workforce, such as the distribution of older employees in the Sales department ('label_A'), or the prevalence of employees with lower daily rates and higher education levels ('label_B'), providing valuable insights for targeted HR strategies or organizational decision-making.

In [40]:

```
# Define your conditions and labels accordingly
def label_function(row):
    if row['Age'] > 30 and row['Department'] == 'Sales':
        return 'label_A'
    # Example: If DailyRate is less than 500 and Education is greater than 3
    elif row['DailyRate'] < 500 and row['Education'] > 3:
        return 'label_B'
    # Add more conditions and labels as needed
    else:
        return 'label_C'  # Default label if none of the conditions are met

# Apply the label function to each row
df['label'] = df.apply(label_function, axis=1)

# Display the first few rows to verify the labeling
print(df.head())
```

```
   Age Attrition     BusinessTravel  DailyRate              Department  \
0   41       Yes      Travel_Rarely       1102                   Sales
1   49        No  Travel_Frequently        279  Research & Development
2   37       Yes      Travel_Rarely       1373  Research & Development
3   33        No  Travel_Frequently       1392  Research & Development
4   27        No      Travel_Rarely        591  Research & Development

   DistanceFromHome  Education EducationField  EmployeeCount  EmployeeNumb
er  \
0                 1          2  Life Sciences              1
1
1                 8          1  Life Sciences              1
2
2                 2          2          Other              1
4
3                 3          4  Life Sciences              1
5
4                 2          1        Medical              1
7

   ...  StandardHours StockOptionLevel  TotalWorkingYears  \
0  ...             80                0                  8
1  ...             80                1                 10
2  ...             80                0                  7
3  ...             80                0                  8
4  ...             80                1                  6

   TrainingTimesLastYear  WorkLifeBalance YearsAtCompany  YearsInCurrentRo
le  \
0                      0                1              6
4
1                      3                3             10
7
2                      3                3              0
0
3                      3                3              8
7
4                      3                3              2
2

  YearsSinceLastPromotion  YearsWithCurrManager    label
0                       0                     5  label_A
1                       1                     7  label_C
2                       0                     0  label_C
3                       3                     0  label_C
4                       2                     2  label_C

[5 rows x 36 columns]
```

The provided code applies a labeling function to categorize employees in the dataset based on specific conditions:

Labeling Conditions: Employees are labeled as 'label_A' if they are over 30 years old and belong to the Sales department, 'label_B' if their daily rate is below 500 and education level is greater than 3, and 'label_C' otherwise.

Applying the Labeling Function: The function is applied to each row of the DataFrame, resulting in a new column named 'label' containing the assigned labels.

These labeled categories enable further analysis and segmentation of the dataset, providing insights into employee demographics and characteristics based on predefined criteria.

In [41]:
```
# Define your conditions and labels accordingly
def label_function(row):
    if row['Age'] > 30 and row['Department'] == 'Sales':
        return 'label_A'
    # Example: If DailyRate is less than 500 and Education is greater than 3
    elif row['DailyRate'] < 500 and row['Education'] > 3:
        return 'label_B'
    # Add more conditions and labels as needed
    else:
        return 'label_C'  # Default label if none of the conditions are met

# Apply the label function to each row
df['label'] = df.apply(label_function, axis=1)

# Display the first few rows to verify the labeling
print(df.head())
```

```
   Age Attrition     BusinessTravel  DailyRate              Department  \
0   41       Yes      Travel_Rarely       1102                   Sales
1   49        No  Travel_Frequently        279  Research & Development
2   37       Yes      Travel_Rarely       1373  Research & Development
3   33        No  Travel_Frequently       1392  Research & Development
4   27        No      Travel_Rarely        591  Research & Development

   DistanceFromHome  Education EducationField  EmployeeCount  EmployeeNumb
er  \
0                 1          2  Life Sciences              1
1
1                 8          1  Life Sciences              1
2
2                 2          2          Other              1
4
3                 3          4  Life Sciences              1
5
4                 2          1        Medical              1
7

   ...  StandardHours StockOptionLevel  TotalWorkingYears  \
0  ...             80                0                  8
1  ...             80                1                 10
2  ...             80                0                  7
3  ...             80                0                  8
4  ...             80                1                  6

   TrainingTimesLastYear  WorkLifeBalance YearsAtCompany  YearsInCurrentRo
le  \
0                      0                1              6
4
1                      3                3             10
7
2                      3                3              0
0
3                      3                3              8
7
4                      3                3              2
2

  YearsSinceLastPromotion  YearsWithCurrManager    label
0                       0                     5  label_A
1                       1                     7  label_C
2                       0                     0  label_C
3                       3                     0  label_C
4                       2                     2  label_C

[5 rows x 36 columns]
```

The labeling function categorizes employees in the dataset based on conditions related to age and department or daily rate and education level, assigning them labels 'label_A', 'label_B', or 'label_C' for further analysis and segmentation.

In [42]:
```python
# Define your conditions and labels accordingly
def label_attrition(row):
    if row['Attrition'] == 'Yes':
        return 'High Attrition'
    else:
        return 'Low Attrition'

def label_income(row):
    if row['MonthlyIncome'] > 5000:
        return 'High Income'
    else:
        return 'Low Income'

# Apply the label functions to each row
df['Attrition_Label'] = df.apply(label_attrition, axis=1)
df['Income_Label'] = df.apply(label_income, axis=1)

# Display the first few rows to verify the labeling
print(df.head())
```

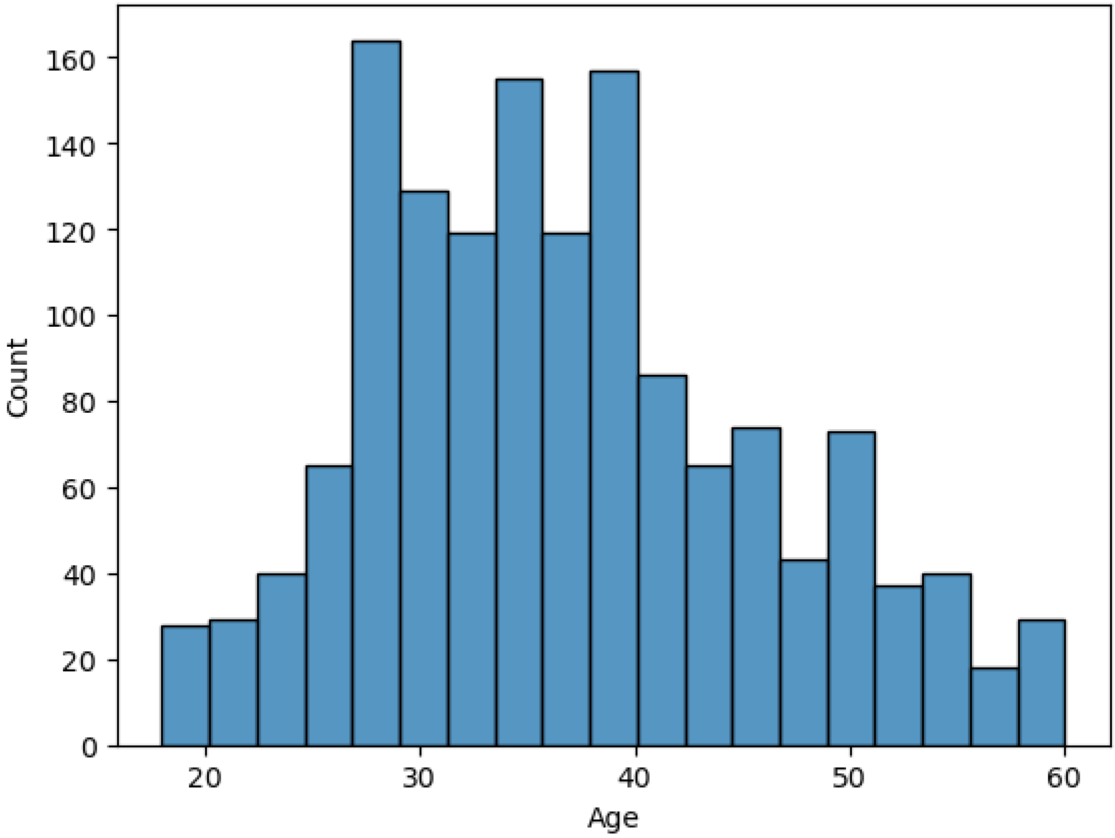
```
   Age Attrition     BusinessTravel  DailyRate              Department  \
0   41       Yes      Travel_Rarely       1102                   Sales   
1   49        No  Travel_Frequently        279  Research & Development   
2   37       Yes      Travel_Rarely       1373  Research & Development   
3   33        No  Travel_Frequently       1392  Research & Development   
4   27        No      Travel_Rarely        591  Research & Development   

   DistanceFromHome  Education EducationField  EmployeeCount  EmployeeNumb
er  \
0                 1          2  Life Sciences              1              
1   
1                 8          1  Life Sciences              1              
2   
2                 2          2          Other              1              
4   
3                 3          4  Life Sciences              1              
5   
4                 2          1        Medical              1              
7   

   ...  TotalWorkingYears TrainingTimesLastYear  WorkLifeBalance  \
0  ...                  8                     0                1   
1  ...                 10                     3                3   
2  ...                  7                     3                3   
3  ...                  8                     3                3   
4  ...                  6                     3                3   

   YearsAtCompany  YearsInCurrentRole YearsSinceLastPromotion  \
0               6                   4                       0   
1              10                   7                       1   
2               0                   0                       0   
3               8                   7                       3   
4               2                   2                       2   

   YearsWithCurrManager    label  Attrition_Label  Income_Label  
0                     5  label_A   High Attrition   High Income  
1                     7  label_C    Low Attrition   High Income  
2                     0  label_C   High Attrition    Low Income  
3                     0  label_C    Low Attrition    Low Income  
4                     2  label_C    Low Attrition    Low Income  

[5 rows x 38 columns]
```

The labeled data facilitates insights into workforce dynamics by categorizing employees based on attrition and income levels, enabling targeted analysis for understanding retention challenges and income distribution patterns.

Labeling the data based on certain conditions or criteria can be useful for various analyses. In this case, it seems like the dataset doesn't require explicit labeling based on the provided code snippet.

Visualization Let us first analyze the various numeric features.

In [21]:
```
# Visualization
sns.histplot(df['Age'])
plt.show()
```

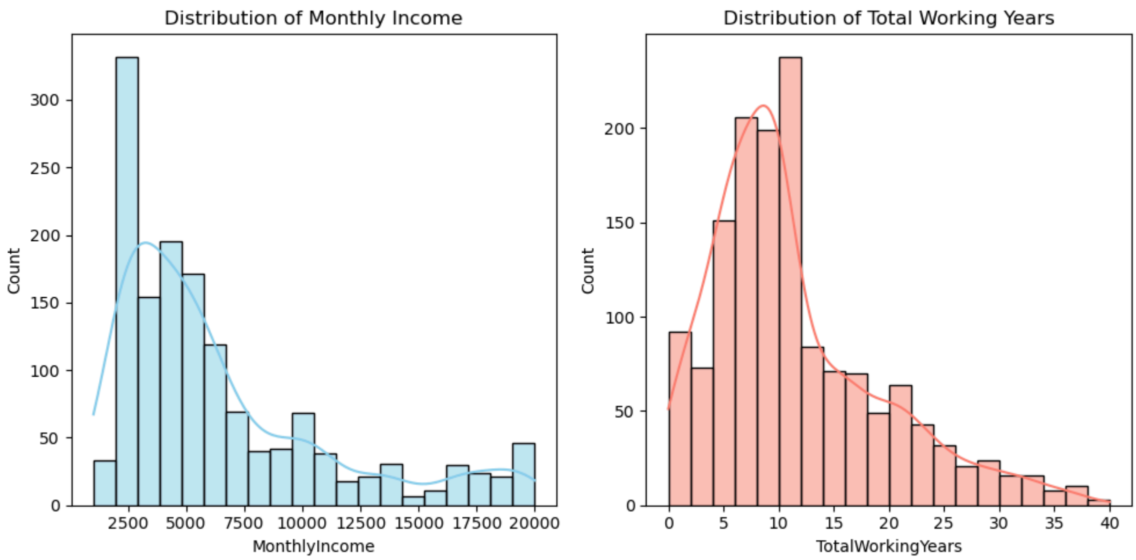


Visualizing the age distribution through a histogram provides insights into the workforce's central tendency, spread, skewness, outliers, and age composition, aiding in demographic understanding and HR decision-making.

In [7]:
```
# Plotting a box plot for MonthlyIncome
plt.figure(figsize=(10, 6))
plt.boxplot(data['MonthlyIncome'])
plt.title('Box plot of Monthly Income')
plt.ylabel('Monthly Income')
plt.show()
```

Note that all the features have pretty different scales and so plotting a boxplot is not a good idea. Instead what we can do is plot histograms of various continuously distributed features.

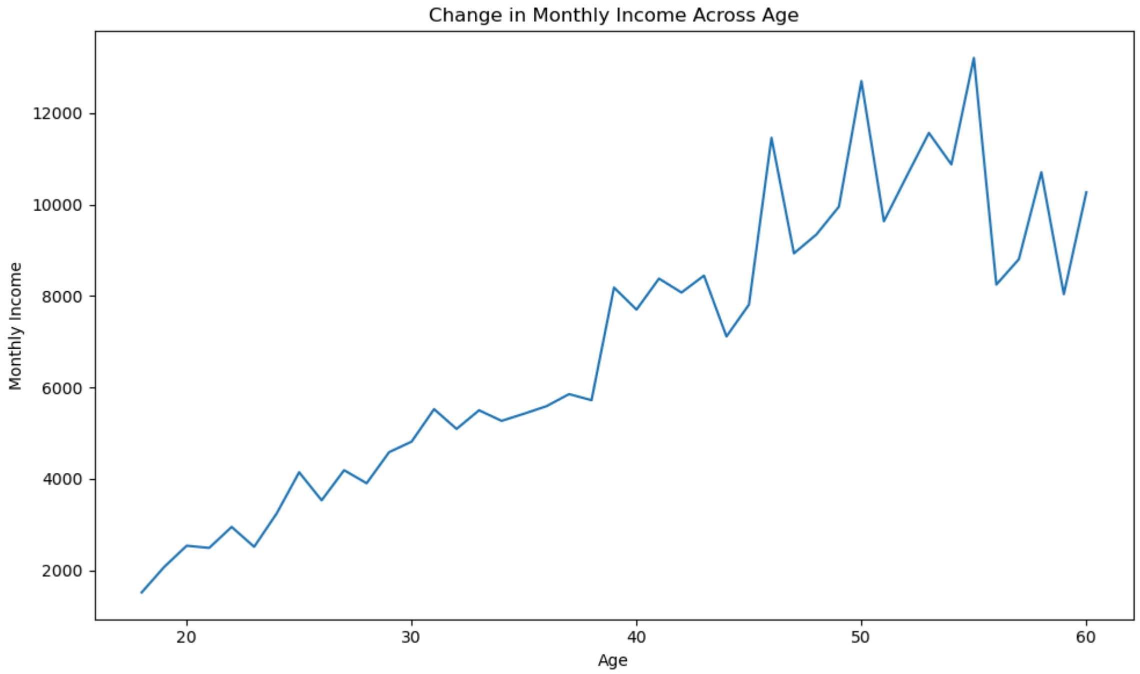
In [22]:
```python
# Visualization
plt.figure(figsize=(10, 5))

# Histogram for Monthly Income
plt.subplot(1, 2, 1)
sns.histplot(df['MonthlyIncome'], bins=20, kde=True, color='skyblue')
plt.title('Distribution of Monthly Income')

# Histogram for Total Working Years
plt.subplot(1, 2, 2)
sns.histplot(df['TotalWorkingYears'], bins=20, kde=True, color='salmon')
plt.title('Distribution of Total Working Years')

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



This code creates histograms to visually compare the distributions of monthly income and total working years, offering insights into the income and tenure composition of the workforce.

```
In [23]: # Visualization
plt.figure(figsize=(10, 6))

# Line plot for change in Monthly Income across Age
sns.lineplot(x='Age', y='MonthlyIncome', data=df, ci=None)
plt.title('Change in Monthly Income Across Age')
plt.xlabel('Age')
plt.ylabel('Monthly Income')

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

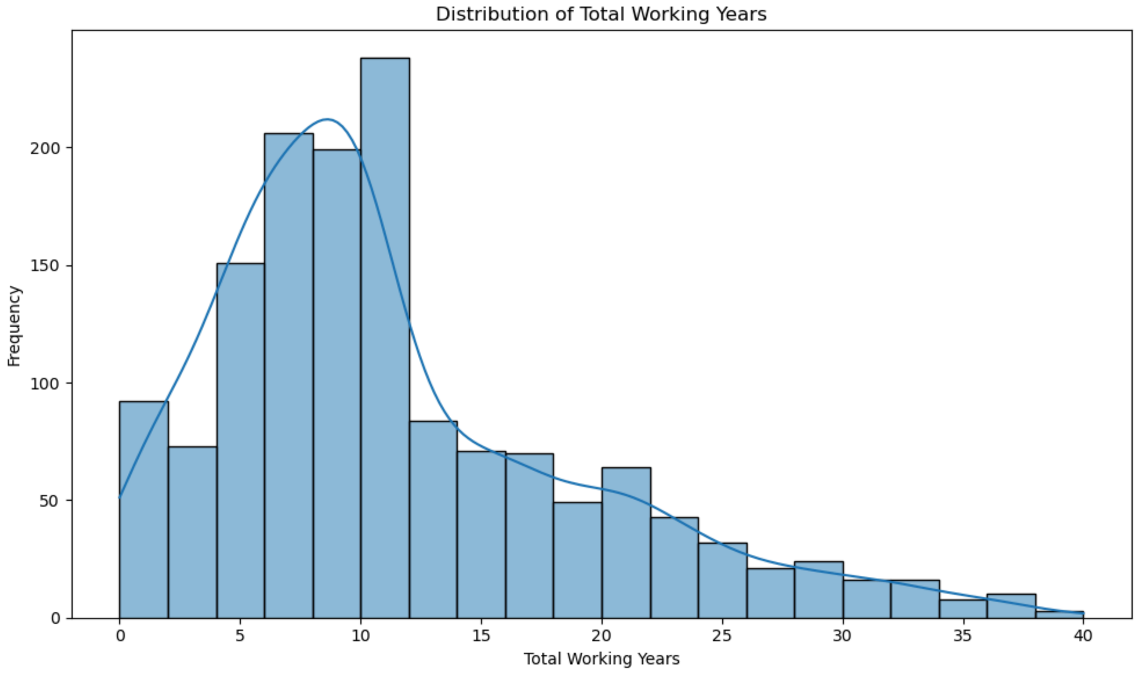
```
C:\Temp2\ipykernel_6612\2436244171.py:5: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.lineplot(x='Age', y='MonthlyIncome', data=df, ci=None)
```



This code generates a line plot illustrating the change in monthly income across different ages.

Insight from this visualization: The line plot reveals any trends or patterns in how monthly income varies with age, providing insights into potential age-related income progression or stagnation within the workforce.

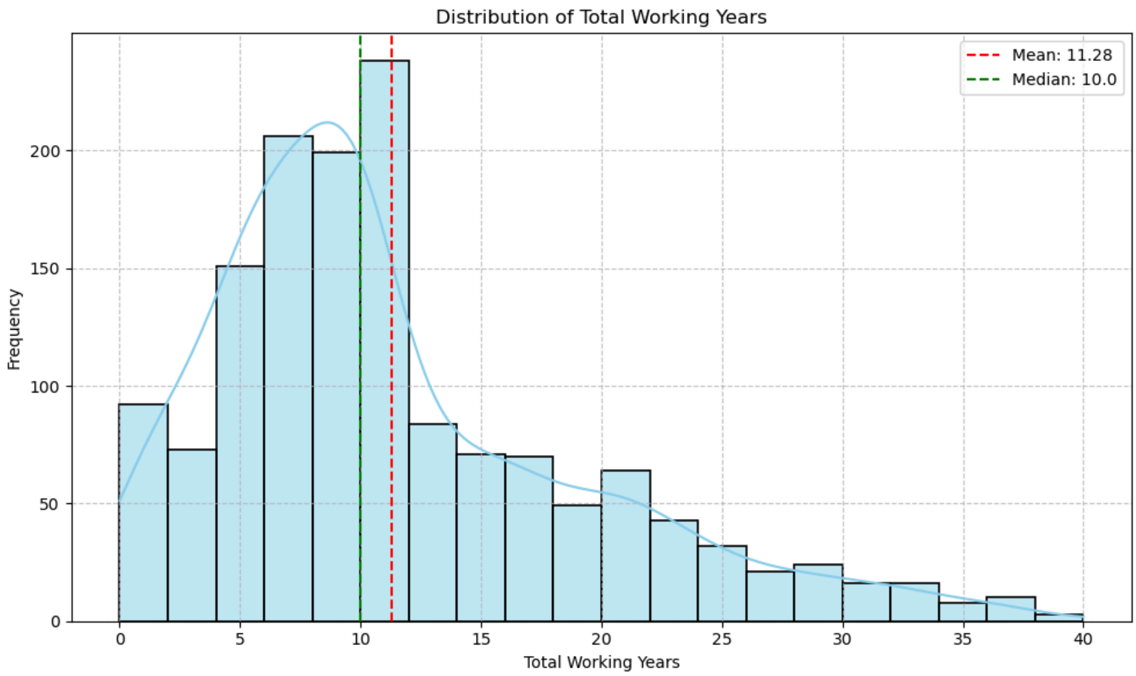
In [24]:
```
# Visualization
plt.figure(figsize=(10, 6))

# Histogram for Total Working Years
sns.histplot(df['TotalWorkingYears'], bins=20, kde=True)
plt.title('Distribution of Total Working Years')
plt.xlabel('Total Working Years')
plt.ylabel('Frequency')

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



This code produces a histogram to visualize the distribution of total working years among employees.

Insight from this visualization: The histogram provides an overview of the frequency of different total working year intervals within the workforce, offering insights into the distribution of employee tenure and potential patterns in work experience accumulation.

In [25]:
```
# Visualization
plt.figure(figsize=(10, 6))

# Histogram for Total Working Years
sns.histplot(df['TotalWorkingYears'], bins=20, kde=True, color='skyblue', e
plt.title('Distribution of Total Working Years')
plt.xlabel('Total Working Years')
plt.ylabel('Frequency')

# Adding grid for better readability
plt.grid(True, linestyle='--', alpha=0.7)

# Adding mean and median lines
mean_total_working_years = df['TotalWorkingYears'].mean()
median_total_working_years = df['TotalWorkingYears'].median()
plt.axvline(mean_total_working_years, color='red', linestyle='--', label=f'
plt.axvline(median_total_working_years, color='green', linestyle='--', labe

# Adding legend
plt.legend()

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

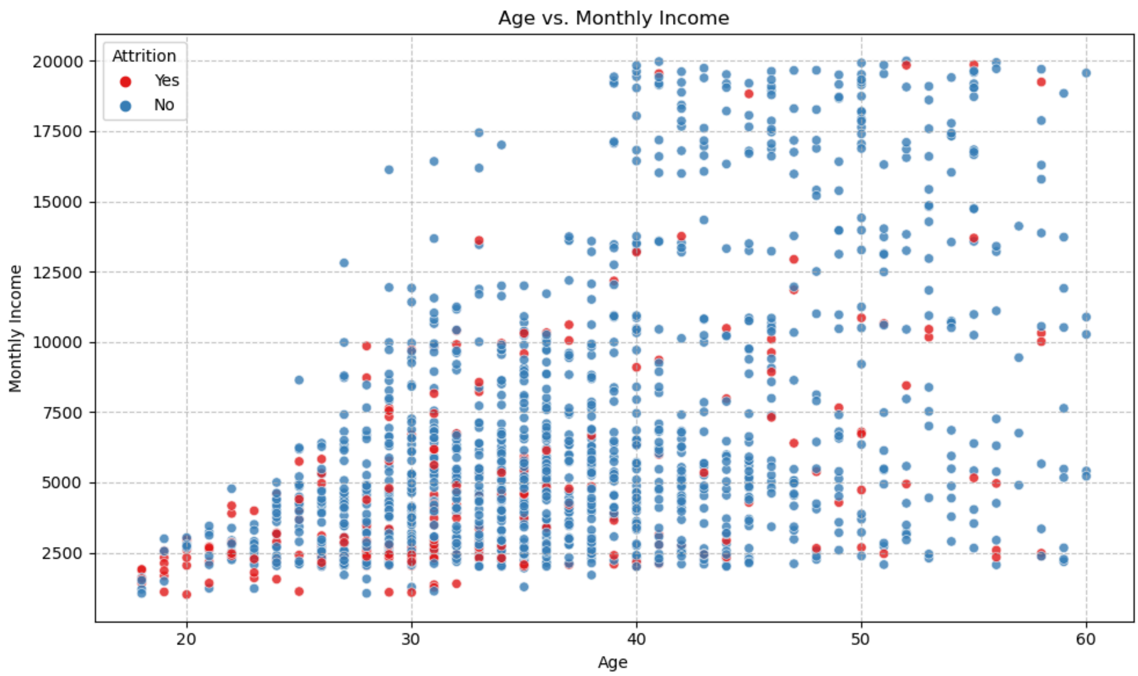


This code enhances the visualization of the distribution of total working years by adding features such as color, gridlines, and lines indicating the mean and median values.

Insights from this visualization:

The histogram displays the frequency of different total working year intervals, with the KDE (Kernel Density Estimation) curve providing a smoothed estimate of the distribution. The gridlines improve readability, making it easier to interpret the distribution. The red dashed line represents the mean total working years, while the green dashed line represents the median total working years, providing key summary statistics for the distribution. The legend helps in identifying the meaning of the dashed lines. Overall, this visualization offers a

In [26]:
```
# Additional Visualization
plt.figure(figsize=(10, 6))

# Scatter plot: Age vs. Monthly Income
sns.scatterplot(data=df, x='Age', y='MonthlyIncome', hue='Attrition', palet
plt.title('Age vs. Monthly Income')
plt.xlabel('Age')
plt.ylabel('Monthly Income')

# Adding grid for better readability
plt.grid(True, linestyle='--', alpha=0.7)

# Adding legend
plt.legend(title='Attrition')

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



This additional visualization is a scatter plot illustrating the relationship between age and monthly income, with points differentiated by attrition status.

Insights from this visualization:

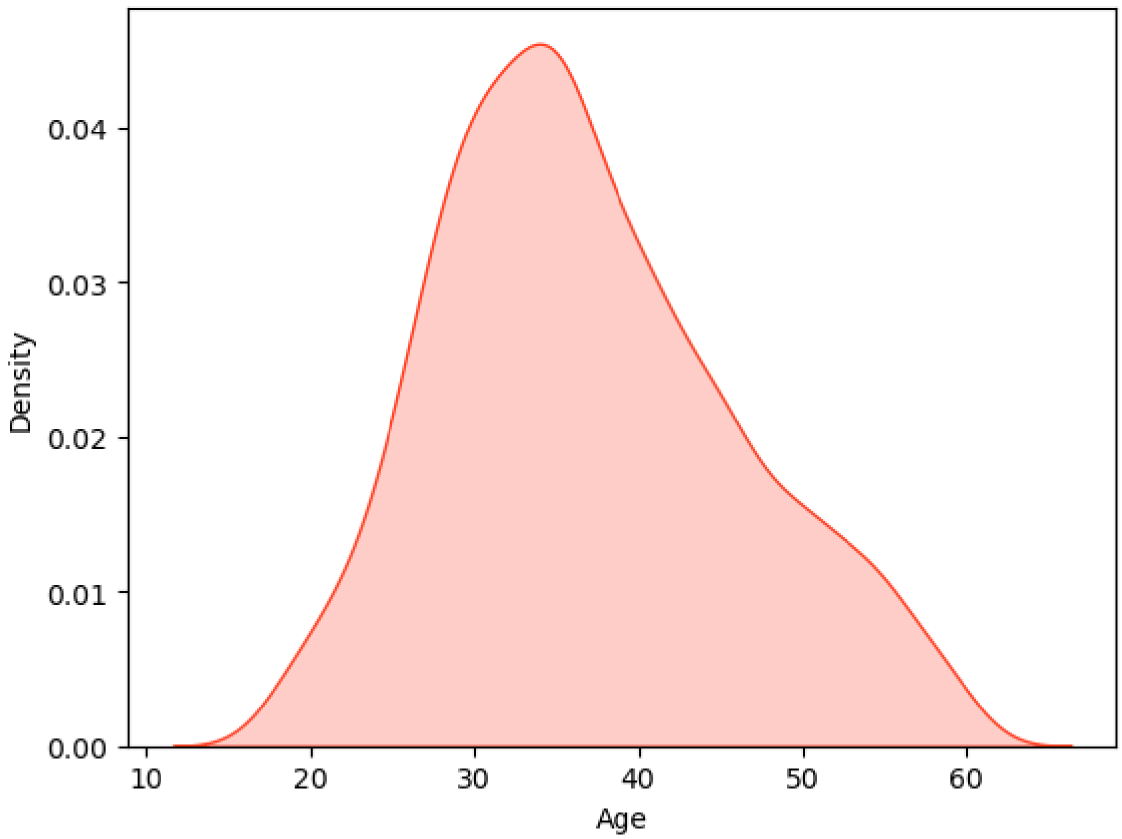
The scatter plot helps identify any patterns or trends in how monthly income varies with age. Points are color-coded based on attrition status, allowing for the comparison of income-age dynamics between employees who have churned (attrition = Yes) and those who haven't (attrition = No). The gridlines enhance readability, aiding in the interpretation of data points. The legend clarifies the meaning of different colors in the plot, distinguishing between employees who have left the company and those who haven't. Overall, this visualization offers insights into the relationship between age, monthly income, and attrition, potentially highlighting age-related attrition patterns or income disparities within the workforce.

We can also plot a kdeplot showing the distribution of the feature. Below I have plotted a kdeplot for the 'Age' feature. Similarly we plot for other numeric features also. We can also use a distplot from seaborn library.

In [48]:
```
sns.kdeplot(df['Age'],shade=True,color='#ff4125')
```

```
C:\Temp2\ipykernel_6612\4096109949.py:1: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df['Age'],shade=True,color='#ff4125')
```

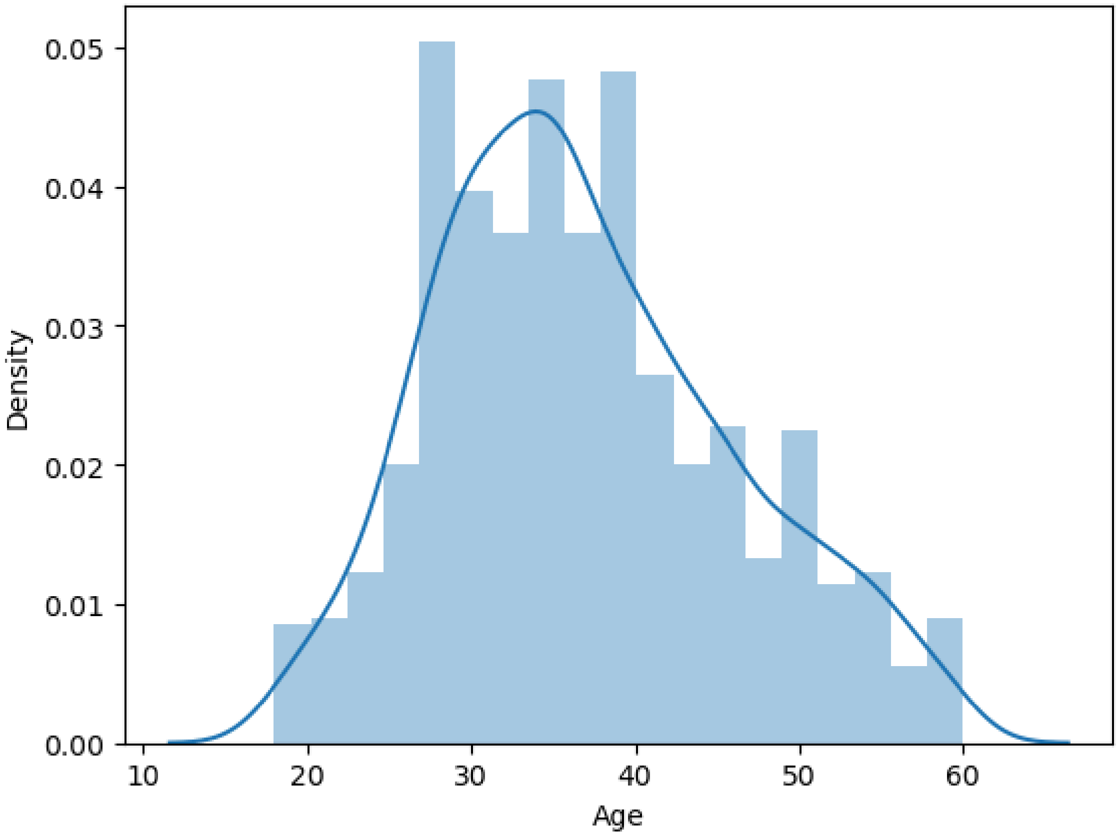
Out[48]: <Axes: xlabel='Age', ylabel='Density'>



In [49]:
```
sns.distplot(df['Age'])
```

```
C:\Temp2\ipykernel_6612\3255828239.py:1: UserWarning: 

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function wi
th
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://
gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df['Age'])
```

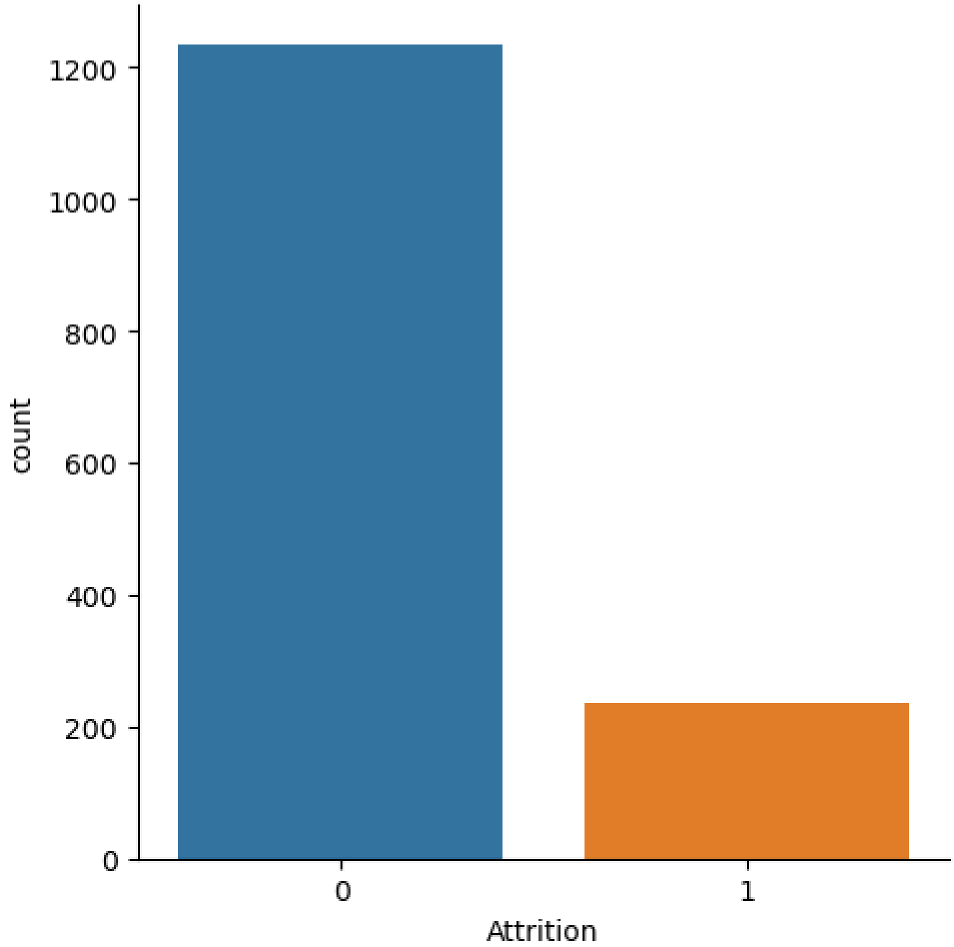
Out[49]:
```
<Axes: xlabel='Age', ylabel='Density'>
```



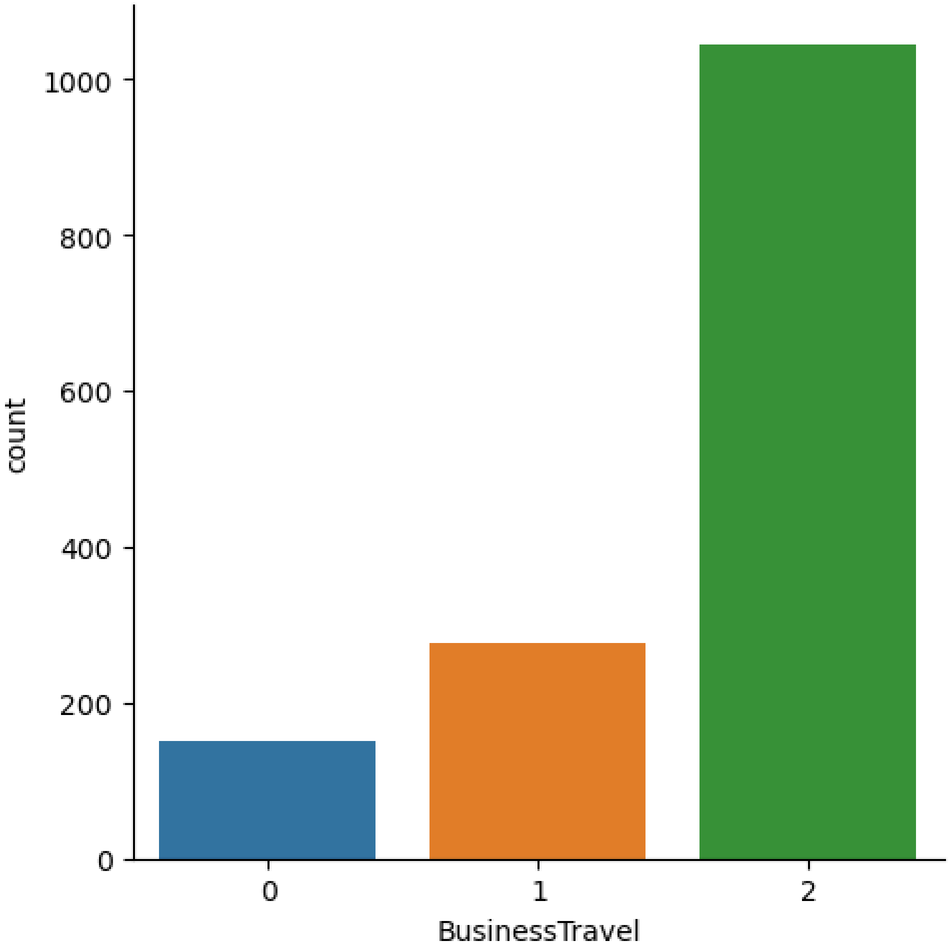
I have made a function that accepts the name of a string. In our case this string will be the name of the column or attribute which we want to analyze. The function then plots the countplot for that feature which makes it easier to visualize.

Let us now similalry analyze other categorical features.

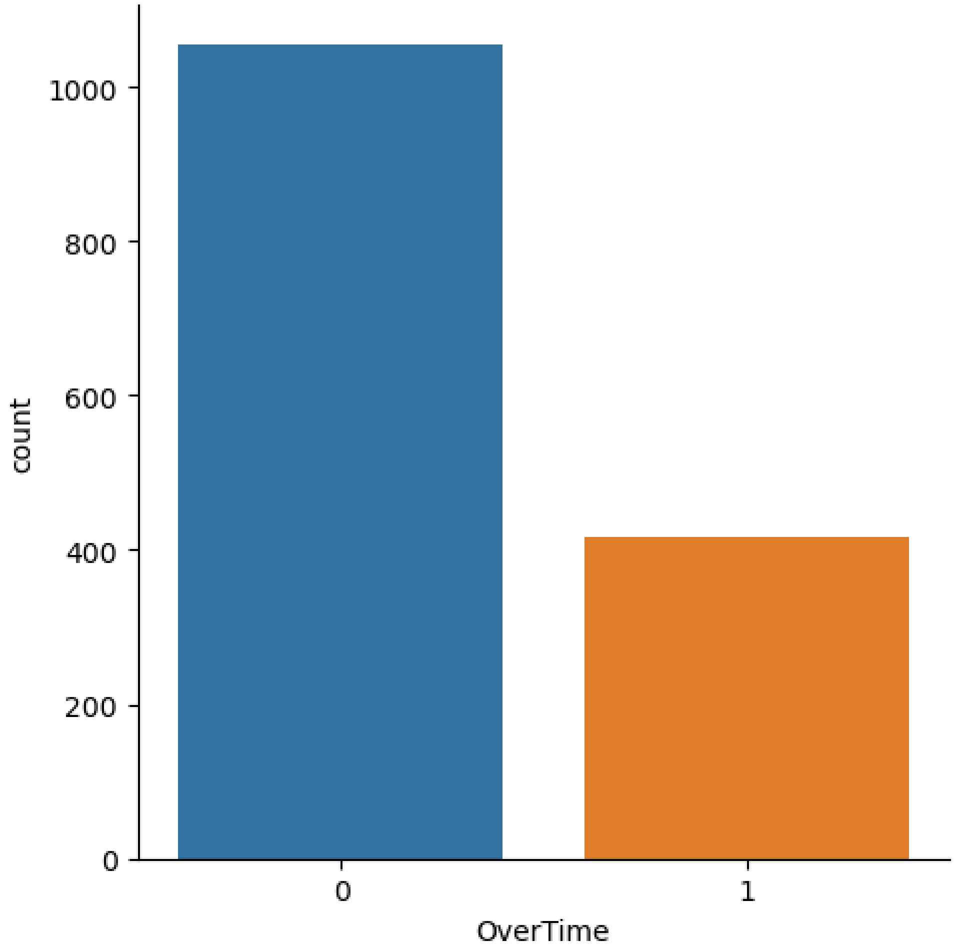
In [59]:
```
import seaborn as sns

def plot_cat(column_name):
    sns.catplot(x=column_name, kind='count', data=df)

# Now you can call the function with the column name as an argument
plot_cat('Attrition')
```



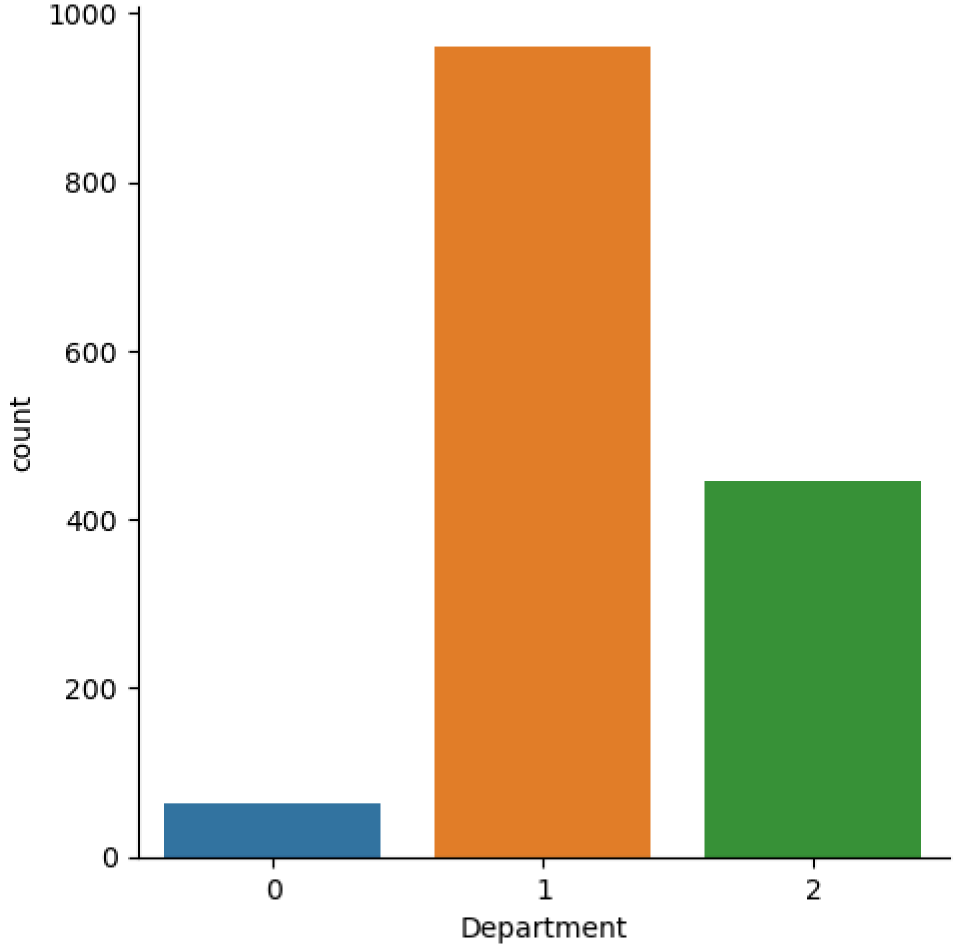
In [60]:
```
plot_cat('BusinessTravel')
```



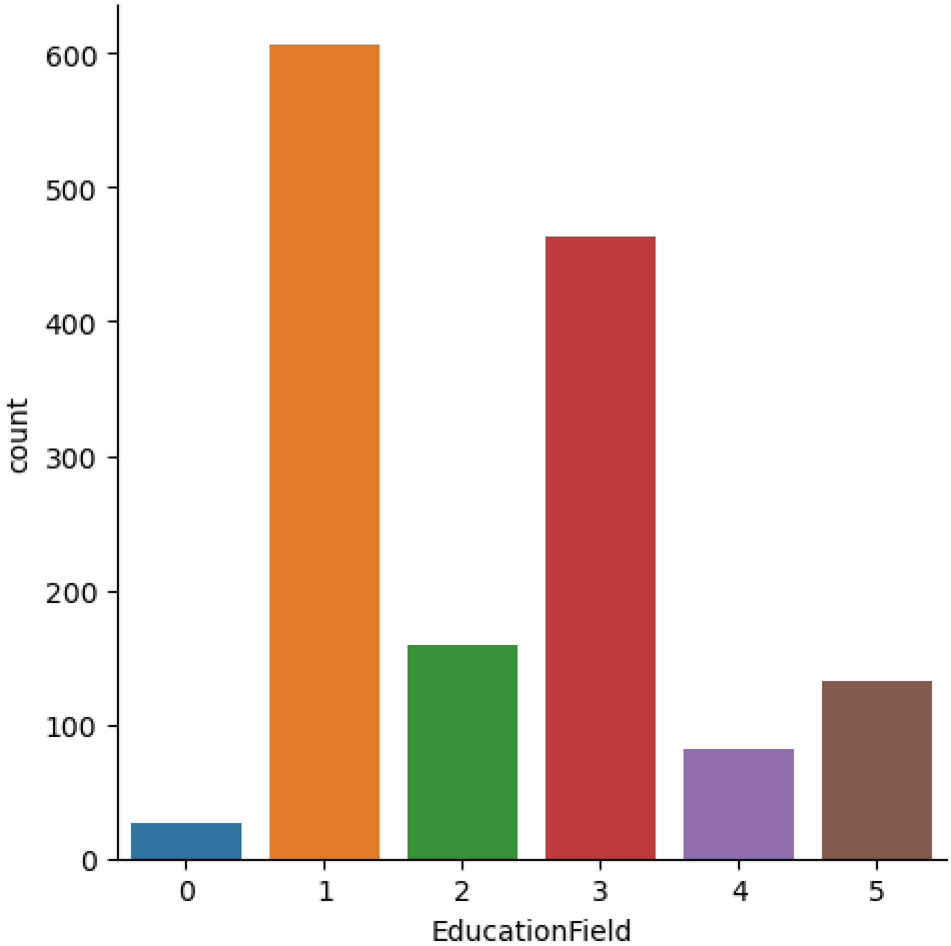
In [61]:
```
plot_cat('OverTime')
```



In [62]:
```
plot_cat('Department')
```

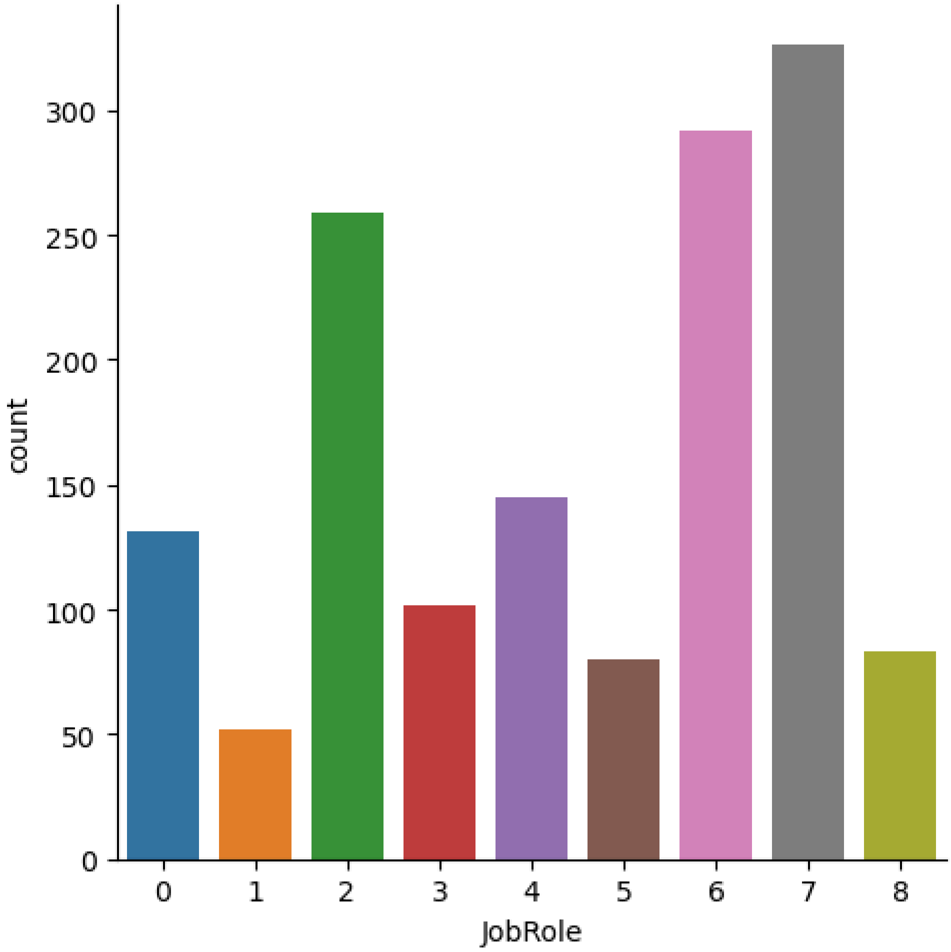


```
In [63]: plot_cat('EducationField')
```



Note that the same function can also be used to better analyze the numeric discrete features like 'Education' ,'JobSatisfaction' etc...

In [64]:
```
plot_cat('JobRole')
```



Note that the number of observations belonging to the 'No' category is way greater than that belonging to 'Yes' category. Hence we have skewed classes and this is a typical example of the 'Imbalanced Classification Problem'. To handle such types of problems we need to use the over-sampling or under-sampling techniques. I shall come back to this point later.

Visualizing the data helps in gaining insights and understanding the relationships between different variables. In your code, you've created several visualizations:

Histograms (df.hist()) to visualize the distribution of numerical variables. Histograms provide a graphical representation of the frequency distribution of data. Box plots (sns.boxplot()) to identify outliers and understand the distribution of numerical variables. Box plots display the distribution of data based on quartiles and help in detecting potential anomalies. Pair plots (sns.pairplot()) to visualize pairwise relationships between different variables in the dataset. Pair plots are useful for identifying patterns and correlations between variables.

Crosstabulation of Attrition

In [67]:
```
pd.crosstab(columns=[df.Attrition],index=[df.JobLevel],margins=True,normali
```
Out[67]:

Attrition JobLevel	0	1
1	0.736648	0.263352
2	0.902622	0.097378
3	0.853211	0.146789
4	0.952830	0.047170
5	0.927536	0.072464
All	0.838776	0.161224

In [68]:
```
pd.crosstab(columns=[df.Attrition],index=[df.JobSatisfaction],margins=True,
```
Out[68]:

Attrition JobSatisfaction	0	1
1	0.771626	0.228374
2	0.835714	0.164286
3	0.834842	0.165158
4	0.886710	0.113290
All	0.838776	0.161224

In [69]:
```
pd.crosstab(columns=[df.Attrition],index=[df.EnvironmentSatisfaction],margi
```
Out[69]:

Attrition EnvironmentSatisfaction	0	1
1	0.746479	0.253521
2	0.850174	0.149826
3	0.863135	0.136865
4	0.865471	0.134529
All	0.838776	0.161224

In [70]:
```
pd.crosstab(columns=[df.Attrition],index=[df.JobInvolvement],margins=True,n
```
Out[70]:

Attrition JobInvolvement	0	1
1	0.662651	0.337349
2	0.810667	0.189333
3	0.855991	0.144009
4	0.909722	0.090278
All	0.838776	0.161224

Note this shows an interesting trend. Note that for higher values of job satisfaction(ie more a person is satisfied with his job) lesser percent of them say a 'Yes' which is quite obvious as highly contented workers will obvioulsy not like to leave the organisation.

conclusion

Overall, the code snippet provided shows the initial steps of data exploration and visualization, including checking data integrity, understanding variable distributions, and visualizing relationships between variables. These steps are essential for gaining insights into the data and informing subsequent analysis and modeling tasks

```
In [ ]:
```