# HR ANALYTICS

# Importing Necessary Libraries

In [25]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from warnings import filterwarnings
filterwarnings ('ignore')
from sklearn.feature_selection import mutual_info_classif
from sklearn.preprocessing import LabelEncoder
import re
```

In [26]:
```
df=pd.read_csv(r"C:\Users\priya\Downloads\HR-Employee-Attrition.csv")
df.head()
```

Out[26]:

| | Age | Attrition | BusinessTravel | DailyRate | Department | DistanceFromHome | Education | I |
|---|---|---|---|---|---|---|---|---|
| **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.shape
```

Out[27]:
```
(1470, 35)
```

In [28]:
```
df.columns
```

Out[28]:
```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCoun
t',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRat
e',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWork
ed',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

In [29]:
```
df.describe()
```

Out[29]:

| | Age | DailyRate | DistanceFromHome | Education | EmployeeCount | Employe |
|---|---|---|---|---|---|---|
| **count** | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.0 | 147 |
| **mean** | 36.923810 | 802.485714 | 9.192517 | 2.912925 | 1.0 | 102 |
| **std** | 9.135373 | 403.509100 | 8.106864 | 1.024165 | 0.0 | 60 |
| **min** | 18.000000 | 102.000000 | 1.000000 | 1.000000 | 1.0 | |
| **25%** | 30.000000 | 465.000000 | 2.000000 | 2.000000 | 1.0 | 49 |
| **50%** | 36.000000 | 802.000000 | 7.000000 | 3.000000 | 1.0 | 102 |
| **75%** | 43.000000 | 1157.000000 | 14.000000 | 4.000000 | 1.0 | 155 |
| **max** | 60.000000 | 1499.000000 | 29.000000 | 5.000000 | 1.0 | 206 |

8 rows × 26 columns

In [30]:
```
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
```

# Data Cleaning

In [31]:
```
df.isnull().sum() #checking for null values
```

Out[31]:
```
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 [32]:
```
#no null values present
```

In [33]:
```
df.duplicated().sum()
```

Out[33]:
```
0
```

In [34]:
```
#no duplicate values present
```

# Dropping redundant columns

In [35]:
```
df=df.drop(['EmployeeCount'],axis=1) #unnecessary columns
```

# Identifying unnecessary columns

In [36]:
```
df1=df.copy()
```

In [37]:
```
le = LabelEncoder()
for col in df1.columns:
    df1[col] = le.fit_transform(df1[col])
```

In [38]:
```
df1.head()
```

Out[38]:

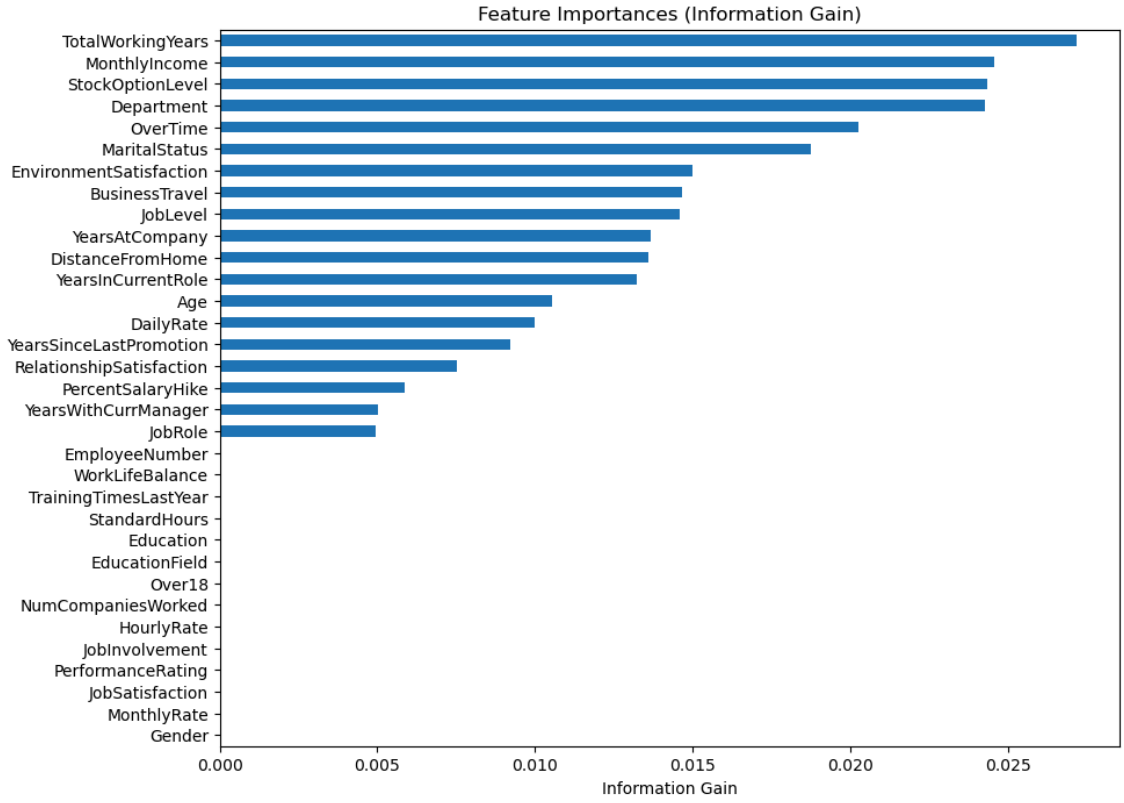
| | Age | Attrition | BusinessTravel | DailyRate | Department | DistanceFromHome | Education | Ed |
|---|---|---|---|---|---|---|---|---|
| **0** | 23 | 1 | 2 | 624 | 2 | 0 | 1 | |
| **1** | 31 | 0 | 1 | 113 | 1 | 7 | 0 | |
| **2** | 19 | 1 | 2 | 805 | 1 | 1 | 1 | |
| **3** | 15 | 0 | 1 | 820 | 1 | 2 | 3 | |
| **4** | 9 | 0 | 2 | 312 | 1 | 1 | 0 | |

5 rows × 34 columns

In [39]:
```
X=df1.drop('Attrition',axis=1)
```

In [40]:
```
y=df1['Attrition']
```

In [47]:
```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.feature_selection import mutual_info_classif

# Calculate feature importances
importances = mutual_info_classif(X, y)
feat_importances = pd.Series(importances, index=X.columns)
sorted_importances = feat_importances.sort_values()

# Plot horizontal bar plot for Information Gain
plt.figure(figsize=(10, 8))
sorted_importances.plot(kind='barh')
plt.xlabel('Information Gain')
plt.title('Feature Importances (Information Gain)')
plt.show()
```



In [48]:
```
imp=sorted_importances.sort_values(ascending = False)
imp
```

```
YearsSinceLastPromotion     0.009206
RelationshipSatisfaction    0.007520
PercentSalaryHike           0.005873
YearsWithCurrManager        0.005036
JobRole                     0.004959
MonthlyRate                 0.000000
EmployeeNumber              0.000000
WorkLifeBalance             0.000000
TrainingTimesLastYear       0.000000
StandardHours               0.000000
Education                   0.000000
EducationField              0.000000
Over18                      0.000000
NumCompaniesWorked          0.000000
HourlyRate                  0.000000
JobInvolvement              0.000000
PerformanceRating           0.000000
JobSatisfaction             0.000000
Gender                      0.000000
dtype: float64
```

In [49]:
```
imp_columns=imp[imp>0]
imp_columns
```

Out[49]:
```
TotalWorkingYears           0.027157
MonthlyIncome               0.024573
StockOptionLevel            0.024336
Department                  0.024261
OverTime                    0.020251
MaritalStatus               0.018758
EnvironmentSatisfaction     0.014987
BusinessTravel              0.014673
JobLevel                    0.014577
YearsAtCompany              0.013675
DistanceFromHome            0.013575
YearsInCurrentRole          0.013219
Age                         0.010545
DailyRate                   0.009968
YearsSinceLastPromotion     0.009206
RelationshipSatisfaction    0.007520
PercentSalaryHike           0.005873
YearsWithCurrManager        0.005036
JobRole                     0.004959
dtype: float64
```

In [50]:
```
non_imp_columns=imp[imp==0]
non_imp_columns
```

Out[50]:
```
MonthlyRate              0.0
EmployeeNumber           0.0
WorkLifeBalance          0.0
TrainingTimesLastYear    0.0
StandardHours            0.0
Education                0.0
EducationField           0.0
Over18                   0.0
NumCompaniesWorked       0.0
HourlyRate               0.0
JobInvolvement           0.0
PerformanceRating        0.0
JobSatisfaction          0.0
Gender                   0.0
dtype: float64
```

# Renaming the Columns

In [51]:
```
def format_column_name(column_name):
    return ' '.join(re.findall('[A-Z][a-z]*', column_name))
df.rename(columns=lambda x: format_column_name(x), inplace=True)
```

In [52]:
```
df.head()
```

Out[52]:

| | Age | Attrition | Business Travel | Daily Rate | Department | Distance From Home | Education | Education Field | Emp Nu |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 41 | Yes | Travel_Rarely | 1102 | Sales | 1 | 2 | Life Sciences | |
| **1** | 49 | No | Travel_Frequently | 279 | Research & Development | 8 | 1 | Life Sciences | |
| **2** | 37 | Yes | Travel_Rarely | 1373 | Research & Development | 2 | 2 | Other | |
| **3** | 33 | No | Travel_Frequently | 1392 | Research & Development | 3 | 4 | Life Sciences | |
| **4** | 27 | No | Travel_Rarely | 591 | Research & Development | 2 | 1 | Medical | |

5 rows × 34 columns

# Data Visualization

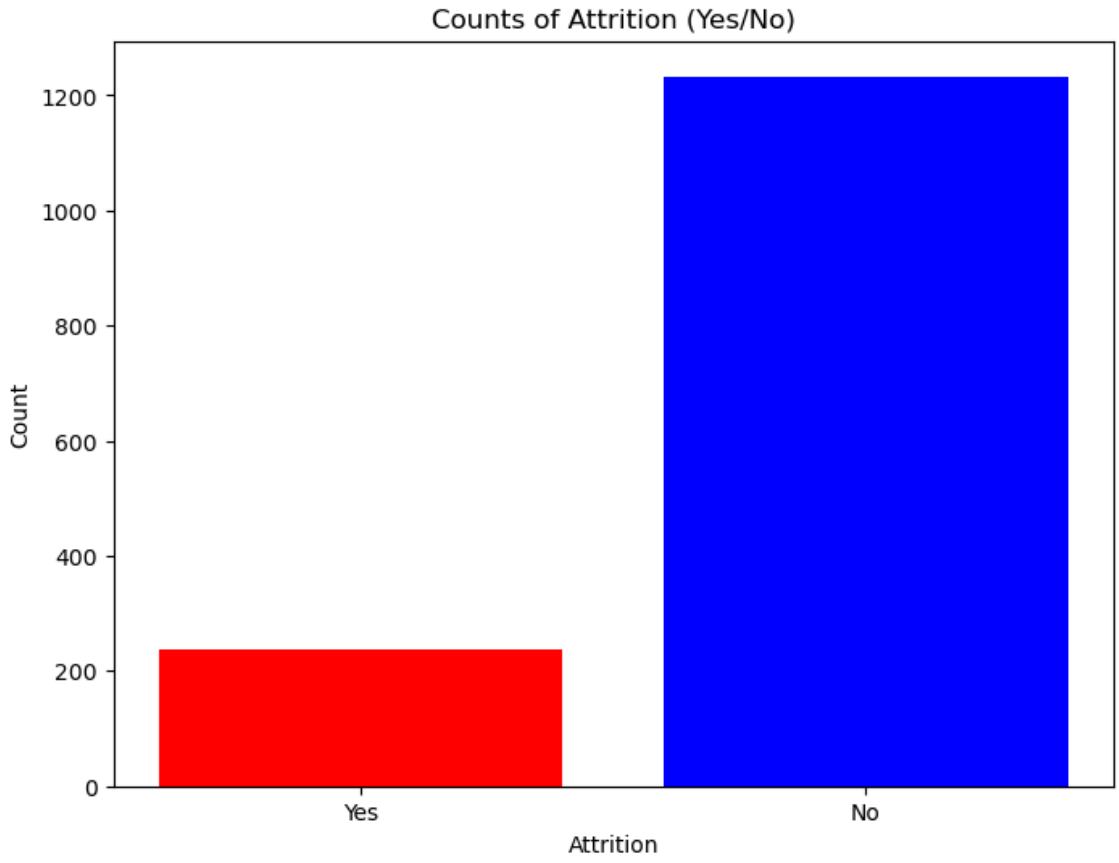
In [53]:
```
df.columns
```

Out[53]:
```
Index(['Age', 'Attrition', 'Business Travel', 'Daily Rate', 'Department',
       'Distance From Home', 'Education', 'Education Field', 'Employee Nu
mber',
       'Environment Satisfaction', 'Gender', 'Hourly Rate', 'Job Involvem
ent',
       'Job Level', 'Job Role', 'Job Satisfaction', 'Marital Status',
       'Monthly Income', 'Monthly Rate', 'Num Companies Worked', 'Over',
       'Over Time', 'Percent Salary Hike', 'Performance Rating',
       'Relationship Satisfaction', 'Standard Hours', 'Stock Option Leve
l',
       'Total Working Years', 'Training Times Last Year', 'Work Life Bala
nce',
       'Years At Company', 'Years In Current Role',
       'Years Since Last Promotion', 'Years With Curr Manager'],
      dtype='object')
```

In [54]:
```
df.Attrition.unique()
```

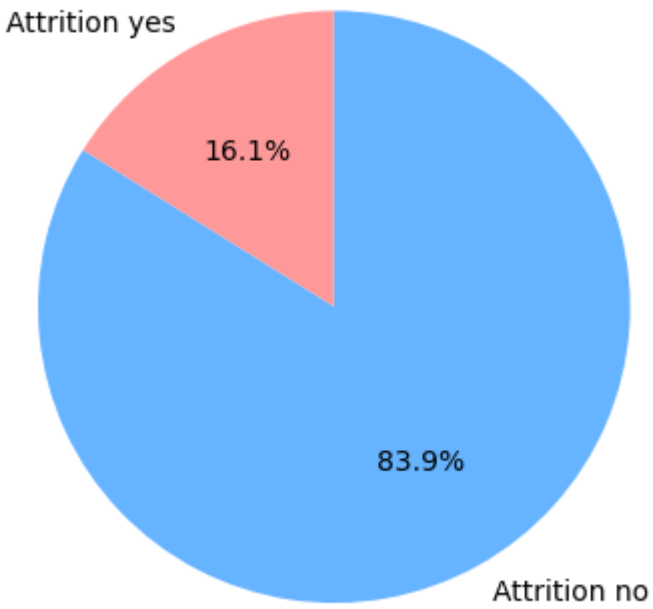
Out[54]:
```
array(['Yes', 'No'], dtype=object)
```

In [55]:
```
# Calculate counts of 'Yes' and 'No' in 'Attrition' column
count_yes = (df['Attrition'] == 'Yes').sum()
count_no = (df['Attrition'] == 'No').sum()

# Create a bar plot
plt.figure(figsize=(8, 6))
plt.bar(['Yes', 'No'], [count_yes, count_no], color=['red', 'blue'])
plt.xlabel('Attrition')
plt.ylabel('Count')
plt.title('Counts of Attrition (Yes/No)')
plt.show()
```



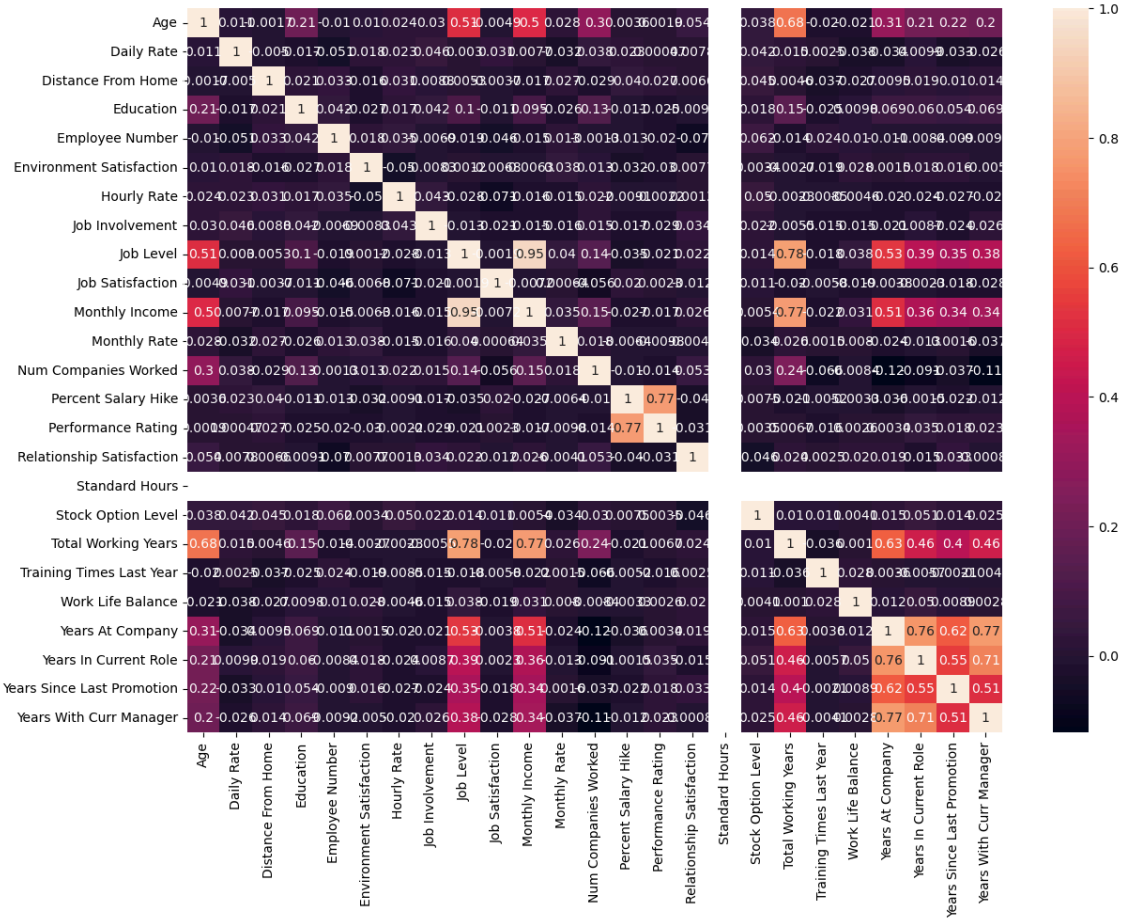
In [56]:
```
labels = ['Attrition yes', 'Attrition no']
sizes = [count_yes, count_no]
colors = ['#ff9999', '#66b3ff']
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=
```

Out[56]:
```
([<matplotlib.patches.Wedge at 0x159a24904f0>,
  <matplotlib.patches.Wedge at 0x159a2490430>],
 [Text(-0.5336332157899545, 0.9618916732177651, 'Attrition yes'),
  Text(0.5336332157899544, -0.9618916732177653, 'Attrition no')],
 [Text(-0.291072663158157, 0.5246681853915082, '16.1%'),
  Text(0.2910726631581569, -0.5246681853915083, '83.9%')])
```



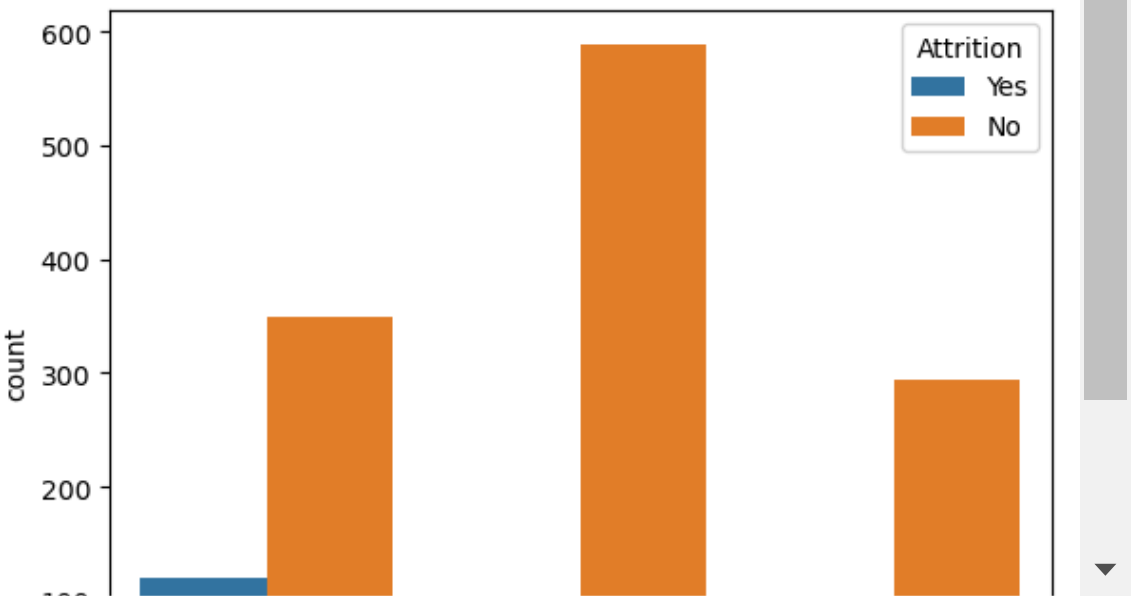
In [57]:
```
plt.figure(figsize=(14, 10))
sns.heatmap(df.corr(),annot=True)
```

Out[57]: <Axes: >

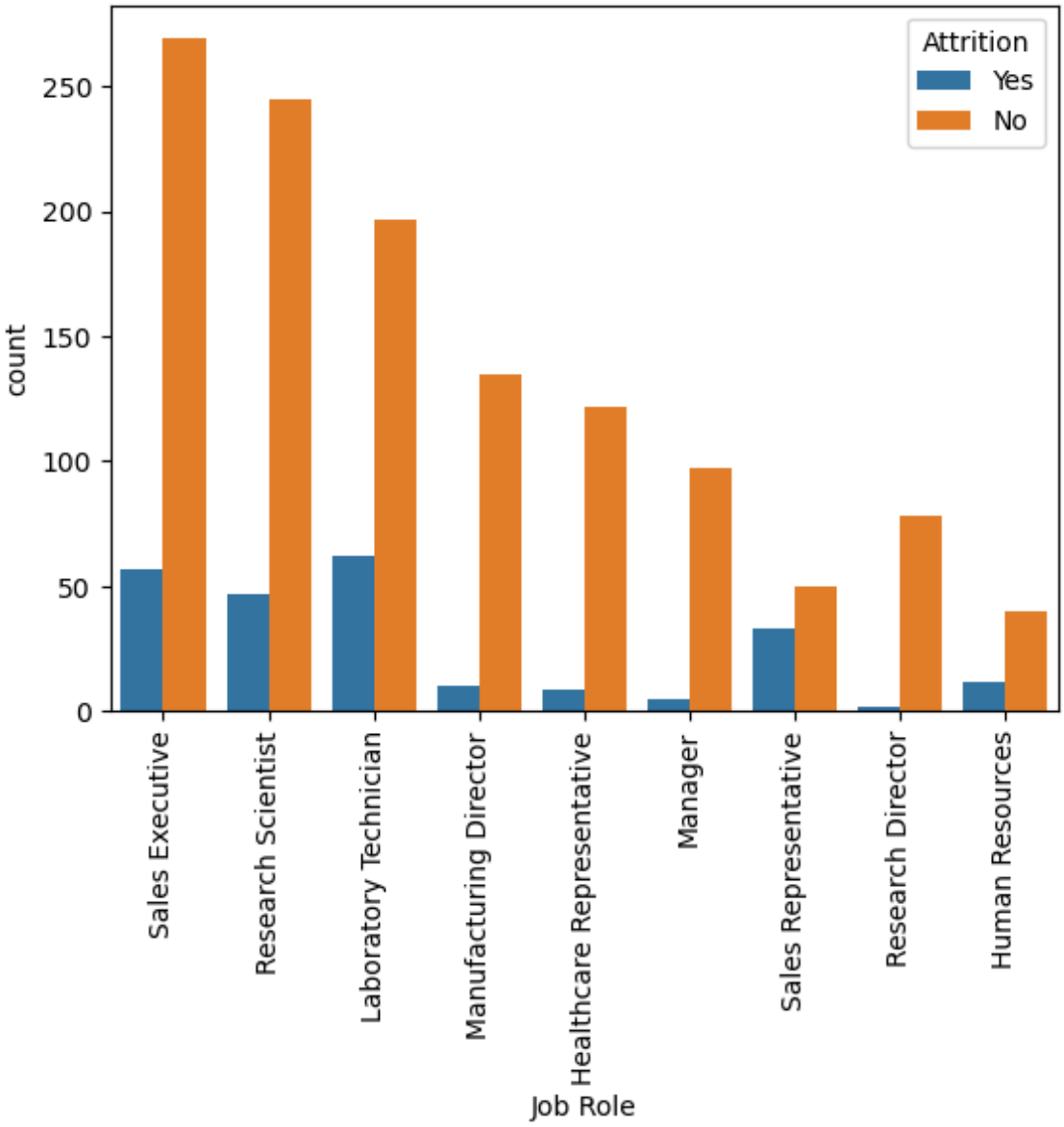



In [58]:
```
sns.countplot(data=df,x=df['Marital Status'],hue=df.Attrition)
```

Out[58]: <Axes: xlabel='Marital Status', ylabel='count'>

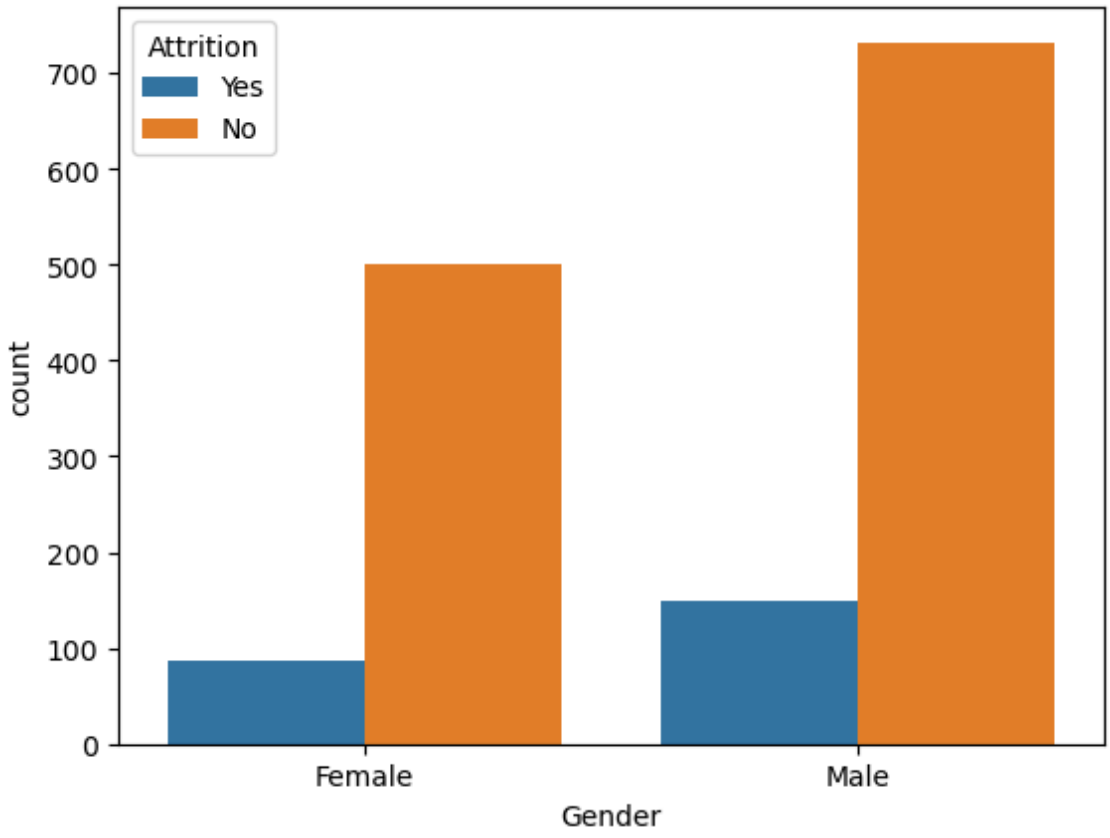


In [59]:
```
sns.countplot(data=df,x=df['Job Role'],hue=df.Attrition)
plt.xticks(rotation=90)
plt.show()
```



In [60]:
```
sns.countplot(data=df,x=df['Gender'],hue=df.Attrition)
```

Out[60]:
```
<Axes: xlabel='Gender', ylabel='count'>
```




In [61]:
```
df.groupby(['Gender','Attrition'])['Attrition'].size().unstack()
```

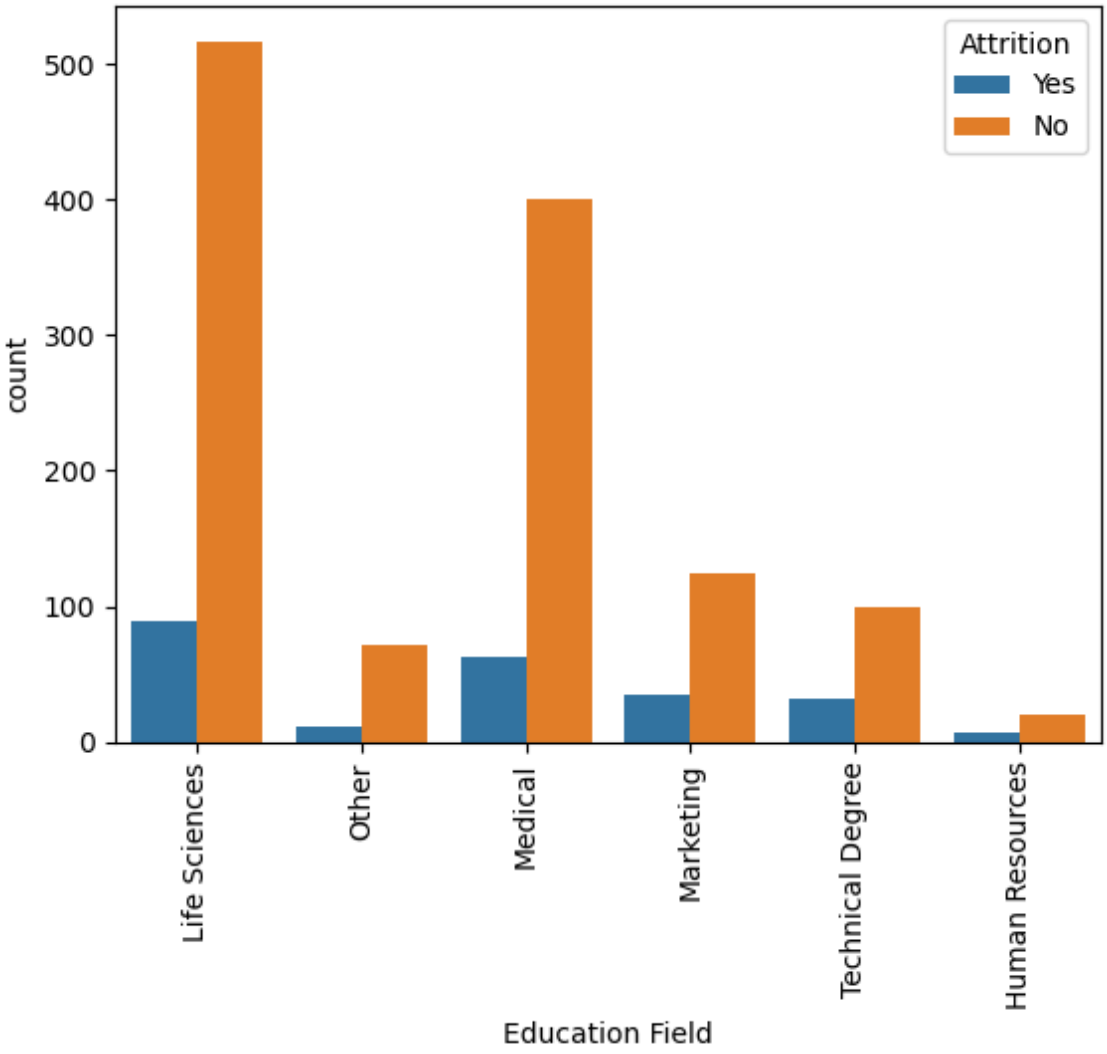
Out[61]:

| Attrition | No | Yes |
|---|---|---|
| **Gender** | | |
| **Female** | 501 | 87 |
| **Male** | 732 | 150 |

In [62]:
```
print('Male attrition rate= ',(150/(150+732))*100,'%')
print('Female attrition rate= ',(87/(87+501))*100,'%')
```

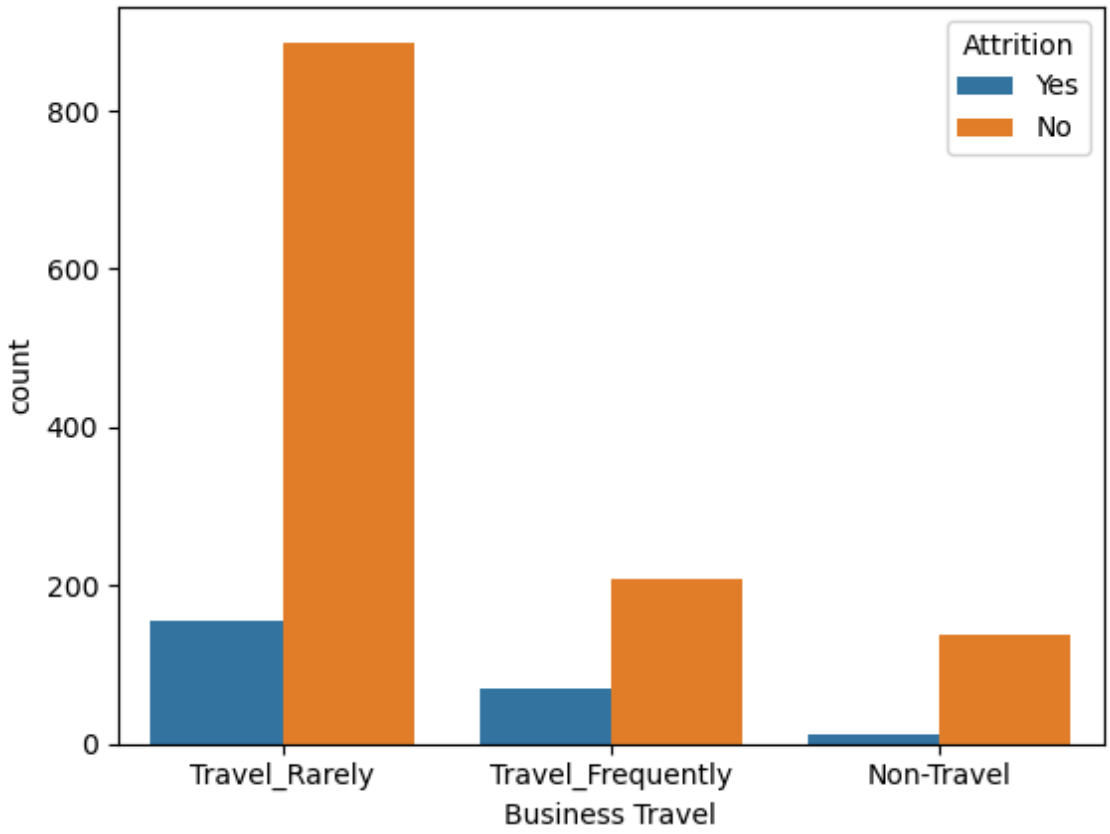
```
Male attrition rate=  17.006802721088434 %
Female attrition rate=  14.795918367346939 %
```

In [63]:
```
sns.countplot(data=df,x=df['Education Field'],hue=df.Attrition)
plt.xticks(rotation=90)
plt.show()
```



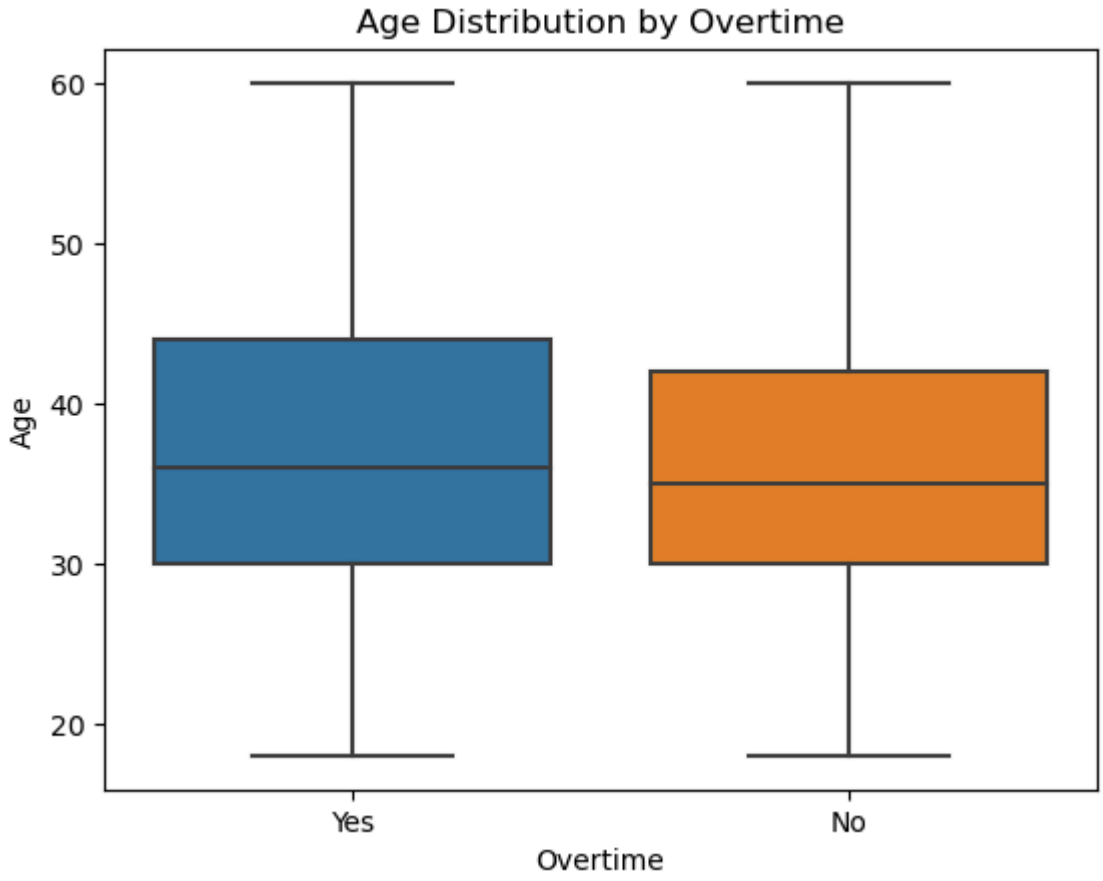
In [64]:
```
sns.countplot(data=df,x=df['Business Travel'],hue=df.Attrition)
```

Out[64]:
```
<Axes: xlabel='Business Travel', ylabel='count'>
```

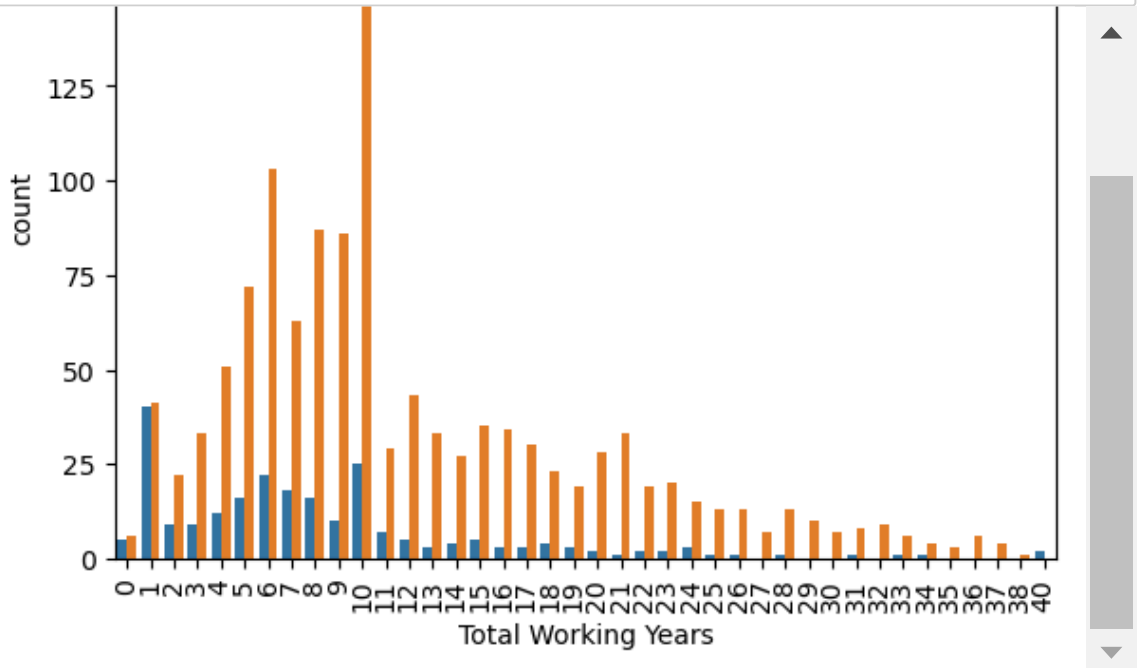



More attrition is seen by people who travel very rarely for the company business

In [65]:
```
sns.boxplot(x='Over Time', y='Age', data=df)
plt.xlabel("Overtime")
plt.ylabel("Age")
plt.title("Age Distribution by Overtime")
plt.show()
```

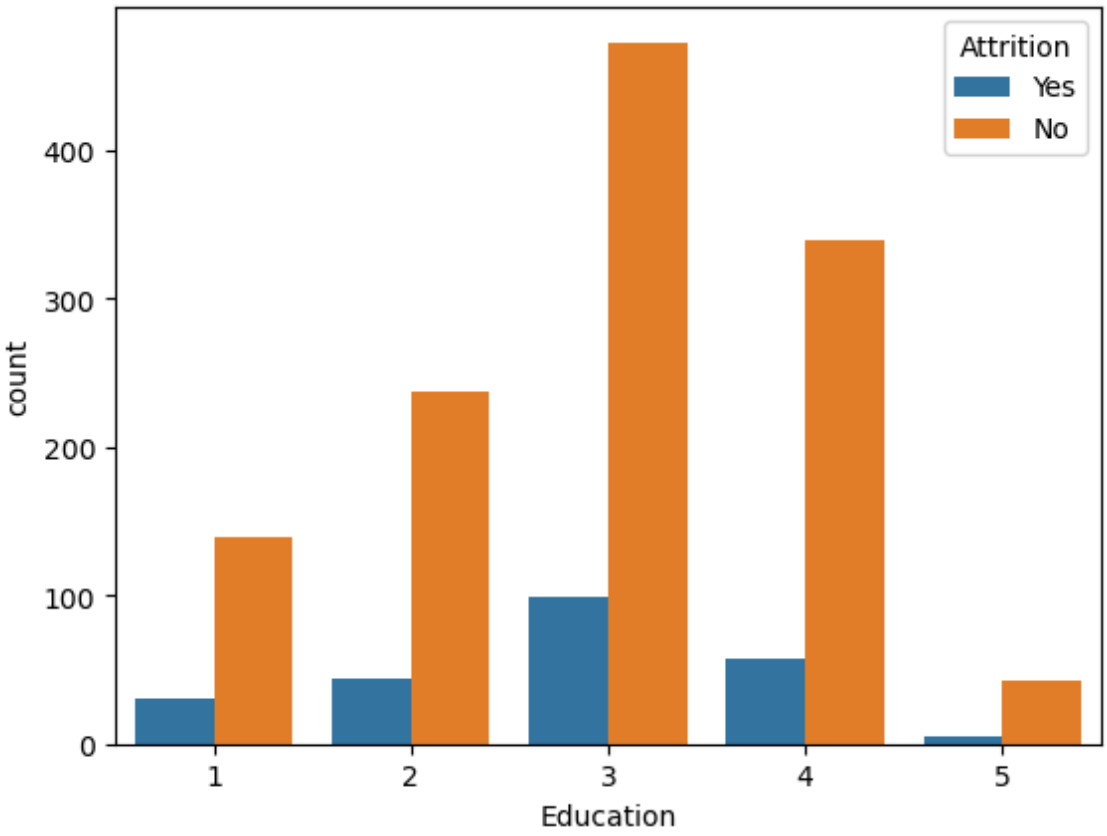



In [66]:
```
sns.countplot(data=df,x=df['Total Working Years'],hue=df.Attrition)
plt.xticks(rotation=90)
plt.show()
```

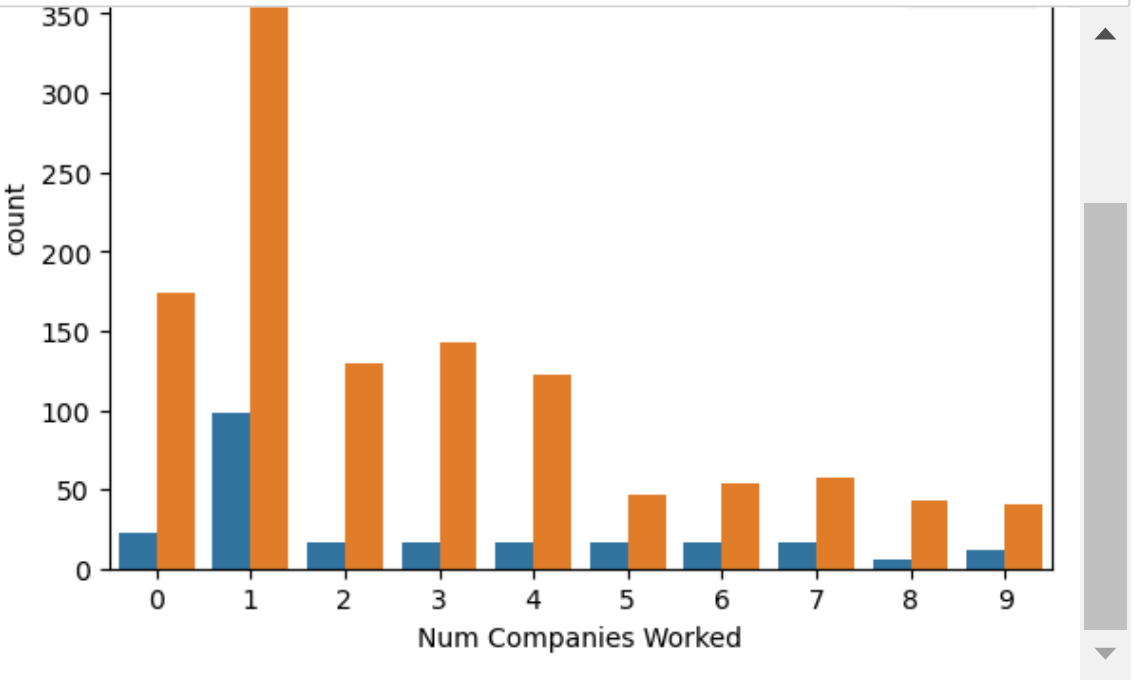


In [67]:
```
sns.countplot(data=df,x=df['Education'],hue=df.Attrition)
```

Out[67]:
```
<Axes: xlabel='Education', ylabel='count'>
```




In [68]:
```
sns.countplot(data=df,x=df['Num Companies Worked'],hue=df.Attrition)
```

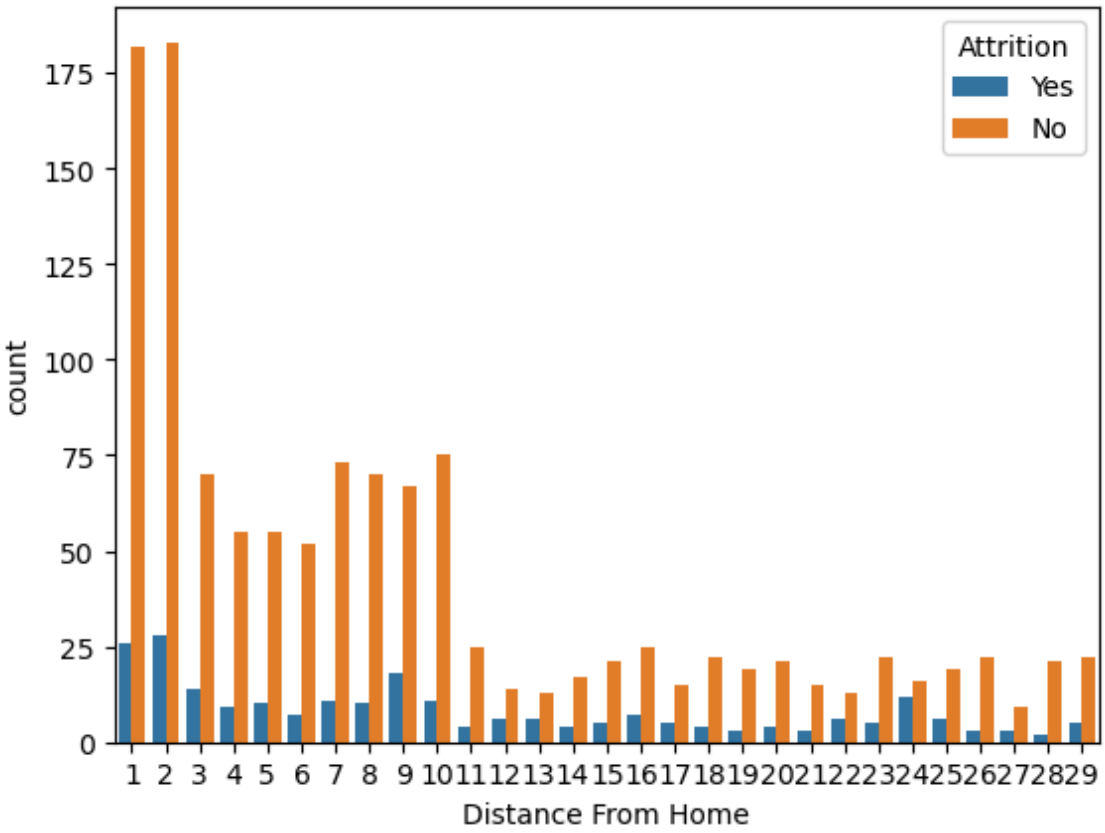



In [69]:
```
df['Distance From Home'].unique()
```

Out[69]:
```
array([ 1,  8,  2,  3, 24, 23, 27, 16, 15, 26, 19, 21,  5, 11,  9,  7,
6,
       10,  4, 25, 12, 18, 29, 22, 14, 20, 28, 17, 13], dtype=int64)
```

In [70]:
```
sns.countplot(data=df,x=df['Distance From Home'],hue=df.Attrition)
```

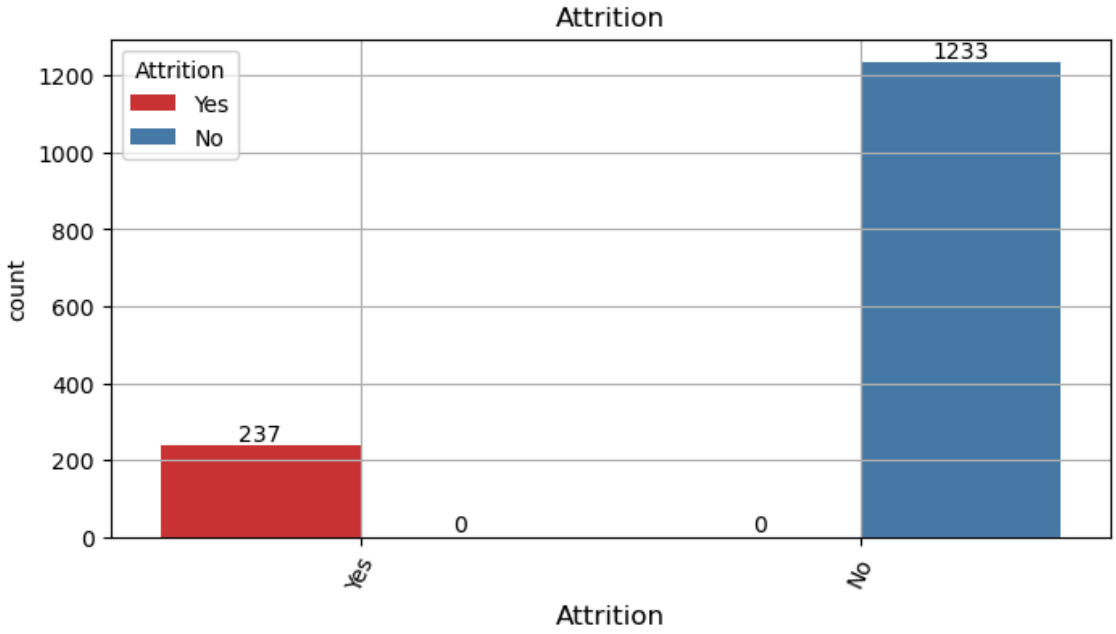
Out[70]:
```
<Axes: xlabel='Distance From Home', ylabel='count'>
```

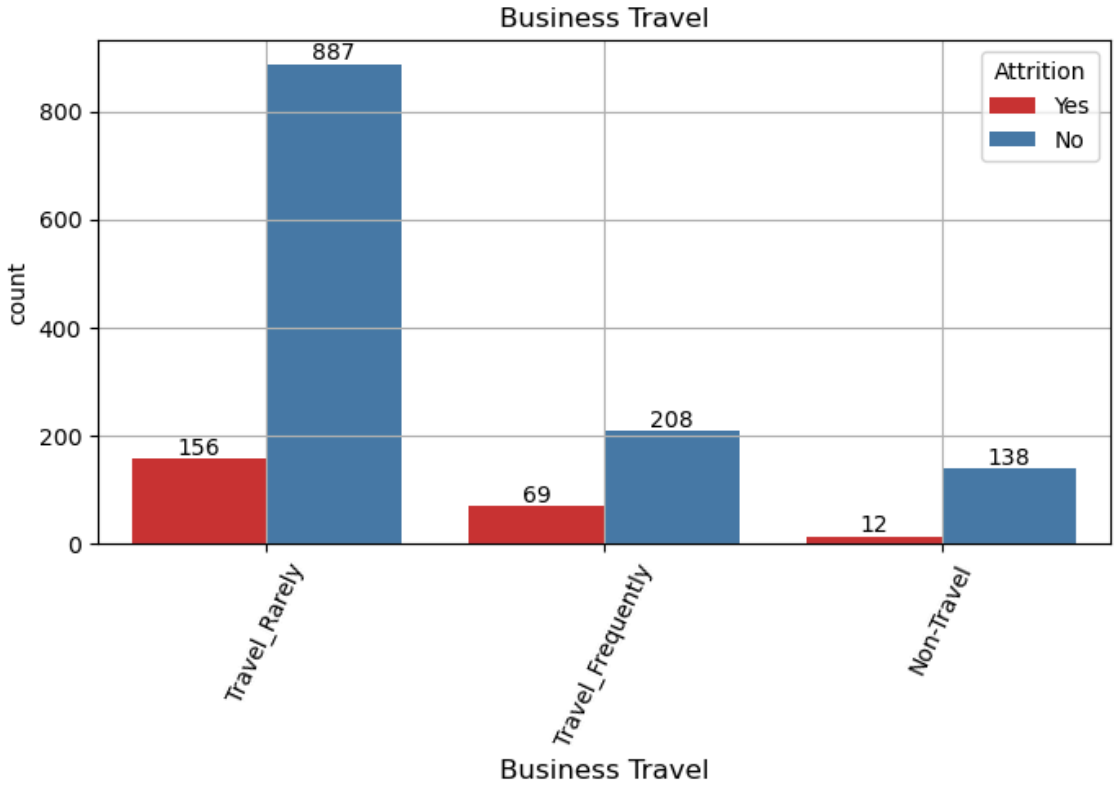


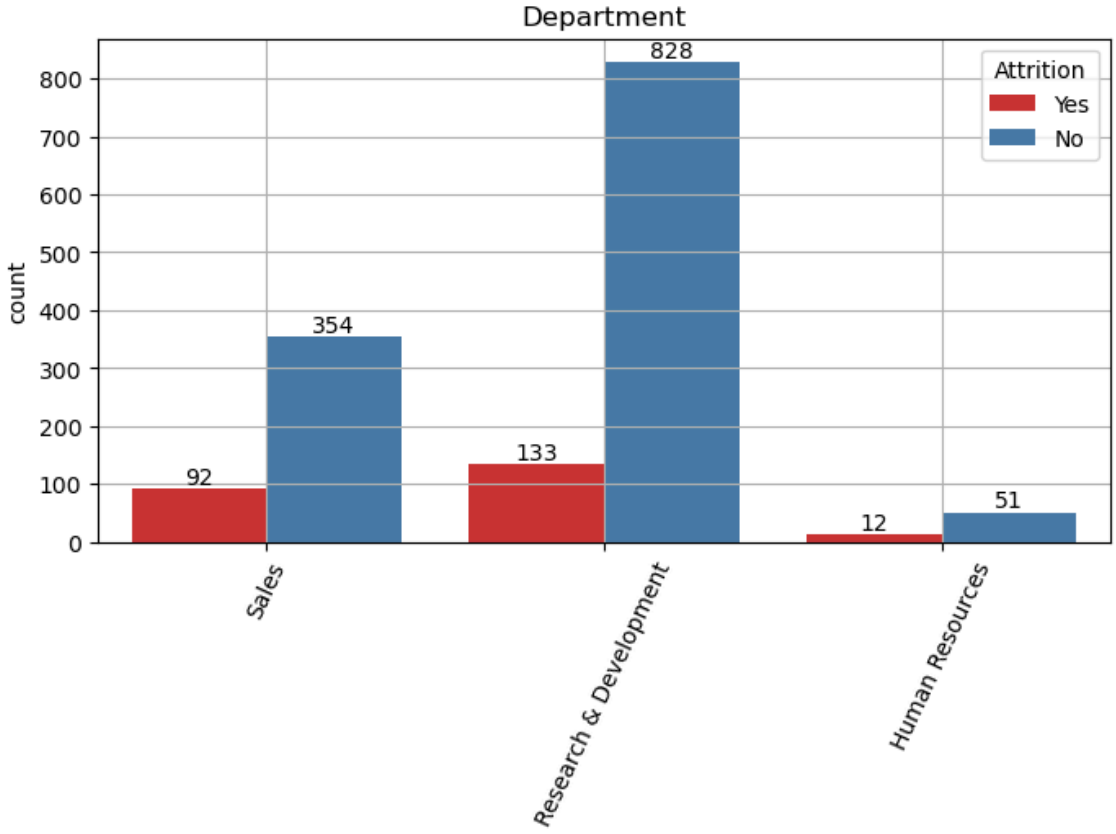

## Similar graphs are plotted along with the count of each bar in various categories as shown:

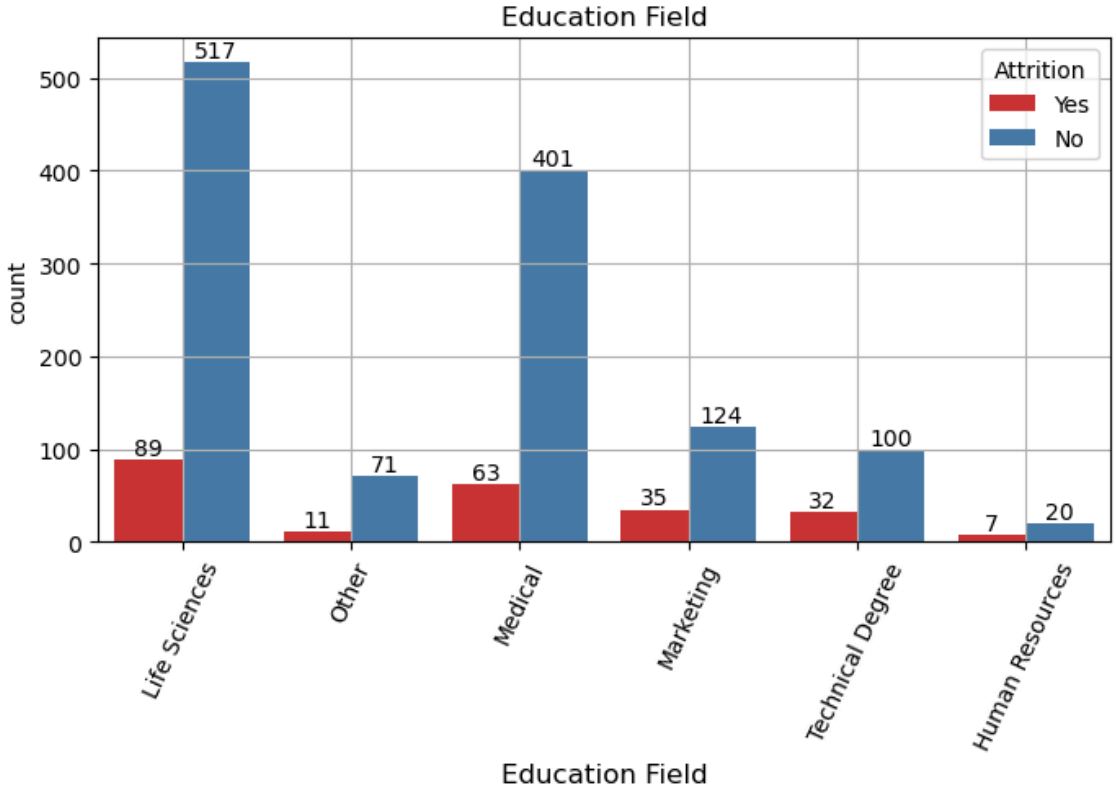
In [71]:
```
cat = df.select_dtypes(['object']).columns
```

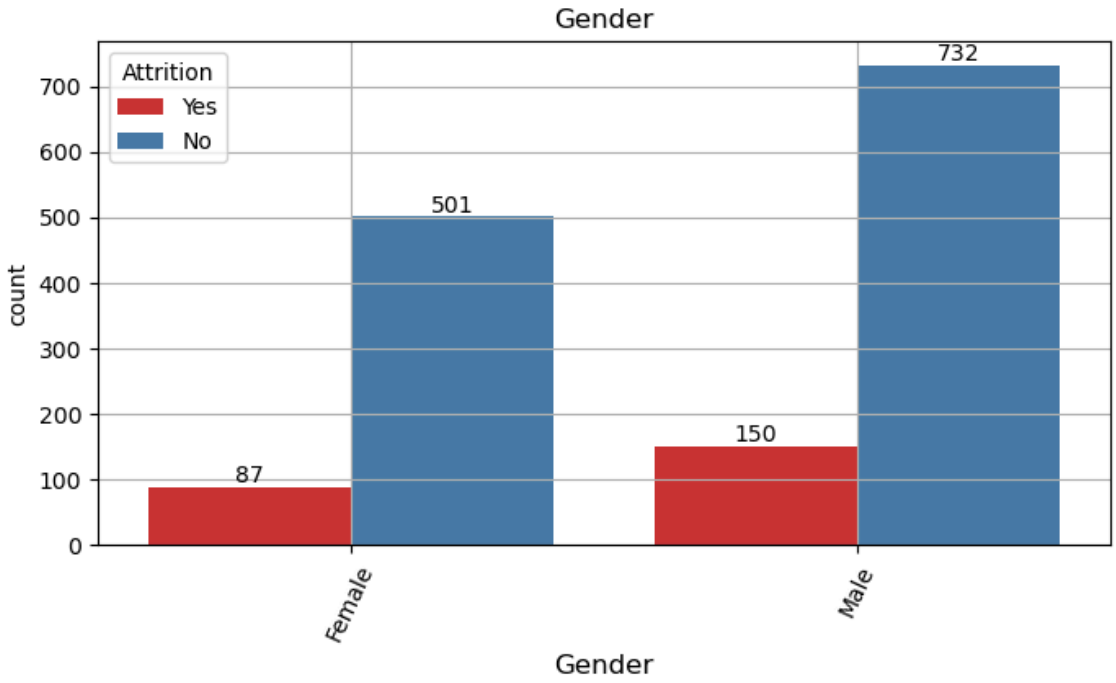
In [72]:
```
for column in cat :
    plt.figure(figsize=(8,4))

    ax=sns.countplot(x=df[column], data=df,hue="Attrition",palette='Set1')
    for container in ax.containers:
        ax.bar_label(container)
    plt.title(column)
    plt.xticks(rotation=65)
    plt.xlabel(column,fontsize=12)
    plt.grid()
    plt.show()
```

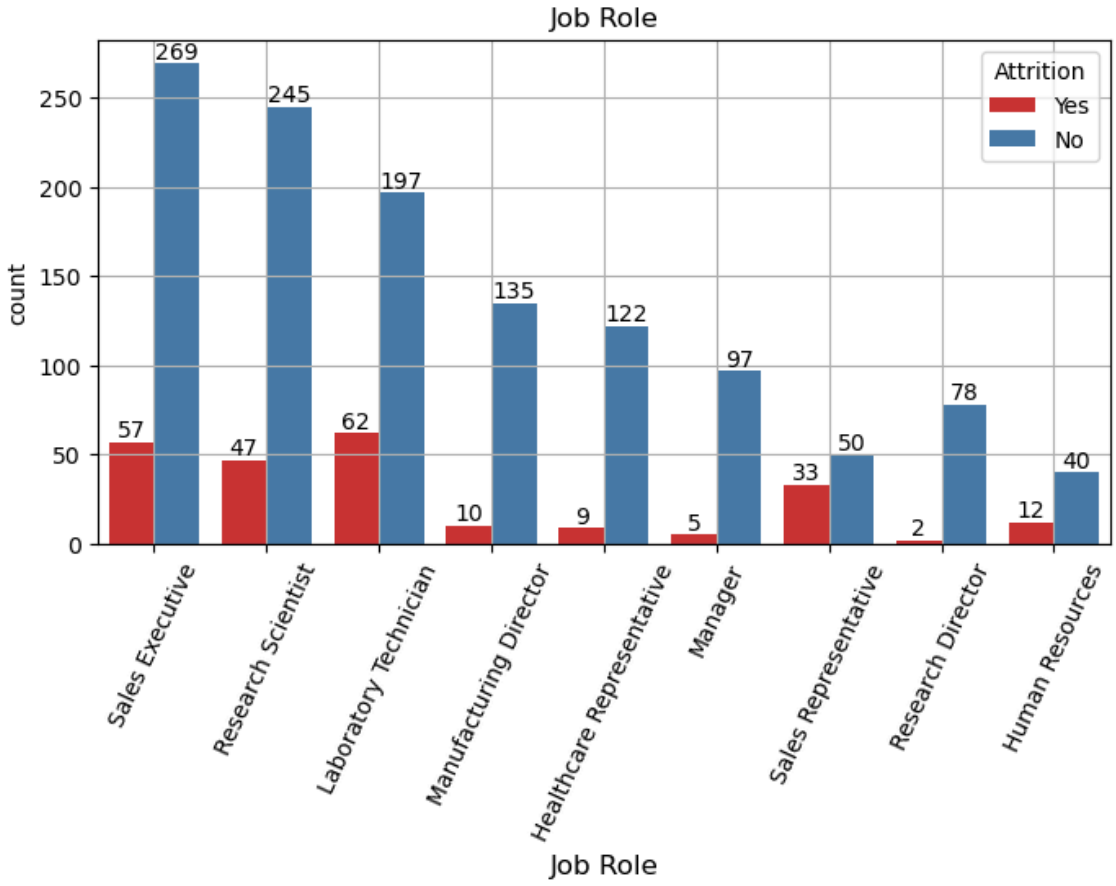


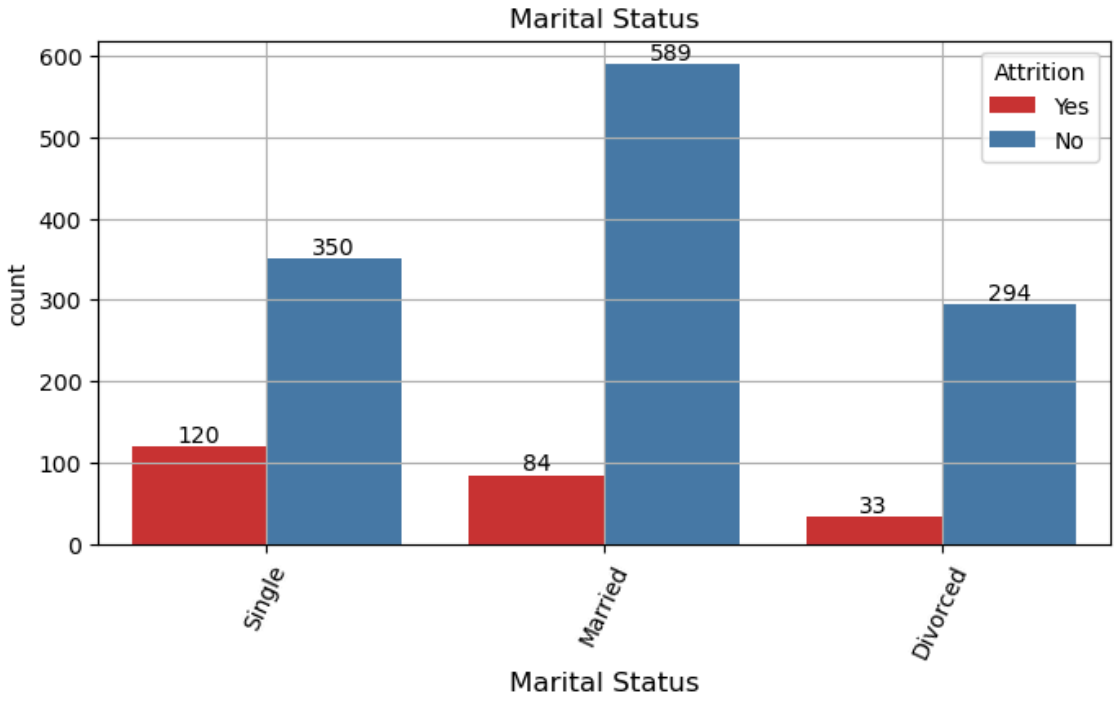



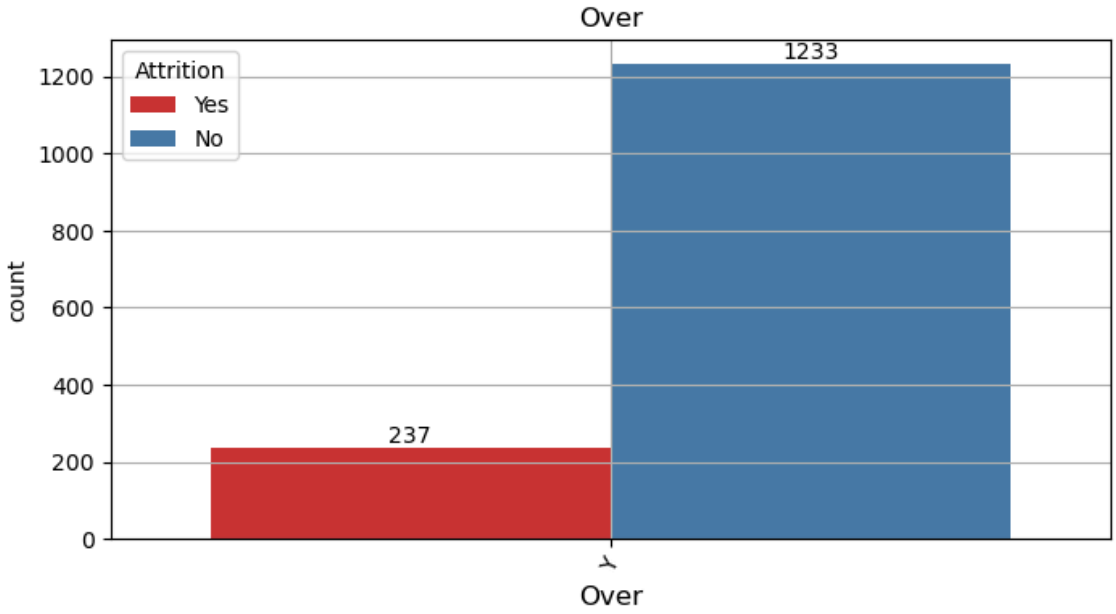

Department
Attrition
Yes
No
count
0
100
200
300
400
500
600
700
800
92
354
133
828
12
51
Sales
Research & Development
Human Resources
Department

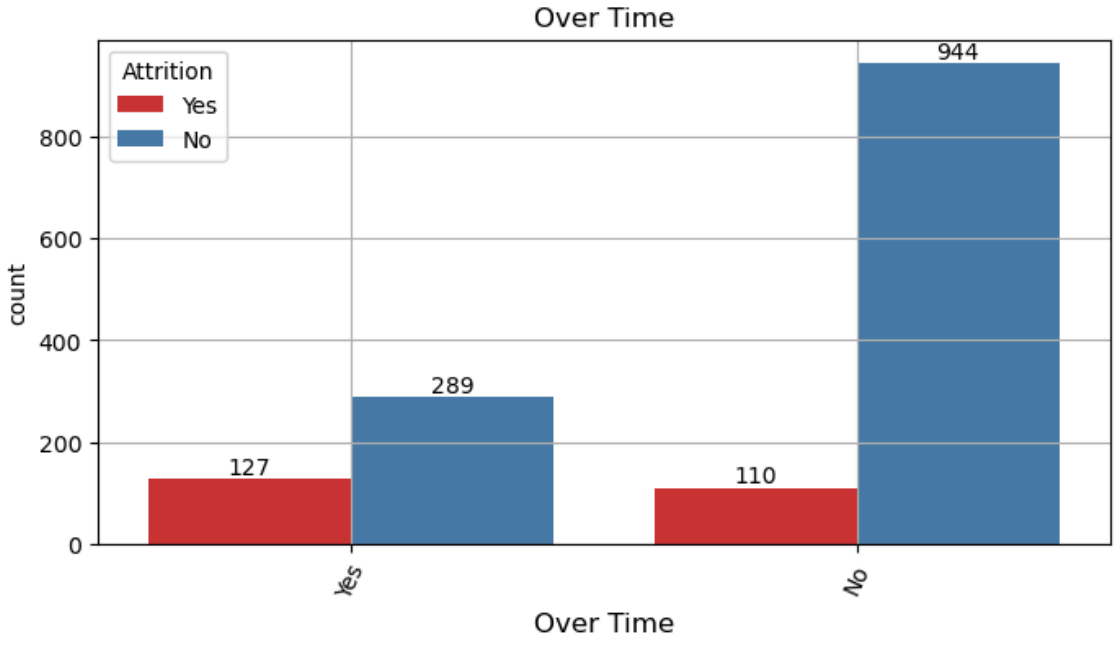

Education Field
Attrition
Yes
No
count
0
100
200
300
400
500
89
517
11
71
63
401
35
124
32
100
7
20
Life Sciences
Other
Medical
Marketing
Technical Degree
Human Resources
Education Field


Gender
Attrition
Yes
No
count
0
100
200
300
400
500
600
700
87
501
150
732
Female
Male
Gender


Job Role
Attrition
Yes
No
count
0
50
100
150
200
250
57
269
47
245
62
197
10
135
9
122
5
97
33
50
2
78
12
40
Sales Executive
Research Scientist
Laboratory Technician
Manufacturing Director
Healthcare Representative
Manager
Sales Representative
Research Director
Human Resources
Job Role


Marital Status
Attrition
Yes
No
count
600
500
400
300
200
100
0
120
350
84
589
33
294
Single
Married
Divorced
Marital Status


Over
Attrition
Yes
No
count
1200
1000
800
600
400
200
0
237
1233
Y
Over


Over Time
Attrition
Yes
No
count
800
600
400
200
0
127
289
110
944
Yes
No
Over Time