Employee Attrition Case Study

Section 01

Overall Business Objective:

The objective is to investigate and understand the factors that lead to employee attrition in the company. By analyzing the data, the goal is to identify patterns, trends, and insights that can help in reducing attrition and improving employee retention.

Understanding of the Problem:

The business analyst will align their understanding of the client's requirements with the problem of employee attrition. This ensures that the analysis is focused on addressing the specific needs of the client.

Approach:

The business analyst will follow a well-defined plan to address the problem. The high-level approach may include data health review, exploratory data analysis (EDA), and KPI/metric-based questions to generate insights and provide actionable recommendations.

In [1]:
```python
# importing all important package....
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from scipy.stats import norm, skew
from scipy import stats
import statsmodels.api as sm
import plotly.express as px
```

In [2]:
```python
#load data into pandas dataframe..
df = pd.read_excel('Employee Attrition DataSet.xlsx','Data')
df.head(5)
```

Out[2]:

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeID	Ger
0	51	No	Travel_Rarely	Sales	6	2	Life Sciences	1	Fer
1	31	Yes	Travel_Frequently	Research & Development	10	1	Life Sciences	2	Fer
2	32	No	Travel_Frequently	Research & Development	17	4	Other	3	M
3	38	No	Non-Travel	Research & Development	2	5	Life Sciences	4	M
4	32	No	Travel_Rarely	Research & Development	10	1	Medical	5	M

5 rows × 21 columns

Section 02: Data Health Review

In [3]:
```
# information of the dataset for checking whether the columns are in the right format is
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4410 entries, 0 to 4409
Data columns (total 21 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Age                      4410 non-null   int64  
 1   Attrition                4410 non-null   object 
 2   BusinessTravel           4410 non-null   object 
 3   Department               4410 non-null   object 
 4   DistanceFromHome         4410 non-null   int64  
 5   Education                4410 non-null   int64  
 6   EducationField           4410 non-null   object 
 7   EmployeeID               4410 non-null   int64  
 8   Gender                   4410 non-null   object 
 9   JobLevel                 4410 non-null   int64  
 10  JobRole                  4410 non-null   object 
 11  MaritalStatus            4410 non-null   object 
 12  MonthlyIncome            4410 non-null   int64  
 13  NumCompaniesWorked       4391 non-null   float64
 14  PercentSalaryHike        4410 non-null   int64  
 15  StockOptionLevel         4410 non-null   int64  
 16  TotalWorkingYears        4401 non-null   float64
 17  TrainingTimesLastYear    4410 non-null   int64  
 18  YearsAtCompany           4410 non-null   int64  
 19  YearsSinceLastPromotion  4410 non-null   int64  
 20  YearsWithCurrManager     4410 non-null   int64  
dtypes: float64(2), int64(12), object(7)
memory usage: 723.6+ KB
```

Dealing with null values

In [4]:
```
#  checking the sum of null values in every column...
totalnull_val = df.isnull().sum()
totalnull_val
```

Out[4]:
```
Age                         0
Attrition                   0
BusinessTravel              0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeID                  0
Gender                      0
JobLevel                    0
JobRole                     0
MaritalStatus               0
MonthlyIncome               0
NumCompaniesWorked         19
PercentSalaryHike           0
StockOptionLevel            0
TotalWorkingYears           9
TrainingTimesLastYear       0
YearsAtCompany              0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

In [5]:
```
# calculating the percentage of null values for every individual column….
percentnull_val = (totalnull_val/df.shape[0])*100
percentnull_val
```

Out[5]:
```
Age                        0.000000
Attrition                  0.000000
BusinessTravel             0.000000
Department                 0.000000
DistanceFromHome           0.000000
Education                  0.000000
EducationField             0.000000
EmployeeID                 0.000000
Gender                     0.000000
JobLevel                   0.000000
JobRole                    0.000000
MaritalStatus              0.000000
MonthlyIncome              0.000000
NumCompaniesWorked         0.430839
PercentSalaryHike          0.000000
StockOptionLevel           0.000000
TotalWorkingYears          0.204082
TrainingTimesLastYear      0.000000
YearsAtCompany             0.000000
YearsSinceLastPromotion    0.000000
YearsWithCurrManager       0.000000
dtype: float64
```

We can observe that NumCompaniesWorked and TotalWorkingYears have 0.43% & 0.20% percent null values respectively Hence our decision of either drop the Null values or imputing them. As the percent null values were very low So we can go ahead and drop null values.

In [6]:
```
# calculating the total no of null values for complete table ….
totalnull_val = df.isnull().sum().sum()
totalnull_val
```

Out[6]:
```
28
```

In [7]:
```
# calculating the percent of total no of null values for complete table ….
percentnull_val = (totalnull_val/df.shape[0])*100
percentnull_val
```

Out[7]:
```
0.6349206349206349
```

Hence 0.63% of values were only affected. So, we can go ahead and drop null values.

In [8]:
```
df.shape # With null values
```

Out[8]:
```
(4410, 21)
```

In [9]:
```
# droping null values
df = df.dropna()
```

In [10]:
```
df.shape #After removing null values
```

Out[10]:
```
(4382, 21)
```

Dealing with datatypes....

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4382 entries, 0 to 4408
Data columns (total 21 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Age                      4382 non-null   int64  
 1   Attrition                4382 non-null   object 
 2   BusinessTravel           4382 non-null   object 
 3   Department               4382 non-null   object 
 4   DistanceFromHome         4382 non-null   int64  
 5   Education                4382 non-null   int64  
 6   EducationField           4382 non-null   object 
 7   EmployeeID               4382 non-null   int64  
 8   Gender                   4382 non-null   object 
 9   JobLevel                 4382 non-null   int64  
 10  JobRole                  4382 non-null   object 
 11  MaritalStatus            4382 non-null   object 
 12  MonthlyIncome            4382 non-null   int64  
 13  NumCompaniesWorked       4382 non-null   float64
 14  PercentSalaryHike        4382 non-null   int64  
 15  StockOptionLevel         4382 non-null   int64  
 16  TotalWorkingYears        4382 non-null   float64
 17  TrainingTimesLastYear    4382 non-null   int64  
 18  YearsAtCompany           4382 non-null   int64  
 19  YearsSinceLastPromotion  4382 non-null   int64  
 20  YearsWithCurrManager     4382 non-null   int64  
dtypes: float64(2), int64(12), object(7)
memory usage: 753.2+ KB
```

As we can see the EmployeeID, Education, JobLevel, NumCompaniesWorked, TotalWorkingYears columns are not in the correct format changing them in to required datatype....

In [12]:
```
# convert datatypes of specific columns
df['EmployeeID'] = df['EmployeeID'].astype(object)
df['Education'] = df['Education'].astype(object)
df['JobLevel'] = df['JobLevel'].astype(object)
df['NumCompaniesWorked'] = df['NumCompaniesWorked'].astype(int)
df['TotalWorkingYears'] = df['TotalWorkingYears'].astype(int)
```

In [13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4382 entries, 0 to 4408
Data columns (total 21 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Age                      4382 non-null   int64  
 1   Attrition                4382 non-null   object 
 2   BusinessTravel           4382 non-null   object 
 3   Department               4382 non-null   object 
 4   DistanceFromHome         4382 non-null   int64  
 5   Education                4382 non-null   object 
 6   EducationField           4382 non-null   object 
 7   EmployeeID               4382 non-null   object 
 8   Gender                   4382 non-null   object 
 9   JobLevel                 4382 non-null   object 
 10  JobRole                  4382 non-null   object 
 11  MaritalStatus            4382 non-null   object 
```

```
 12  MonthlyIncome            4382 non-null   int64 
 13  NumCompaniesWorked       4382 non-null   int32 
 14  PercentSalaryHike        4382 non-null   int64 
 15  StockOptionLevel         4382 non-null   int64 
 16  TotalWorkingYears        4382 non-null   int32 
 17  TrainingTimesLastYear    4382 non-null   int64 
 18  YearsAtCompany           4382 non-null   int64 
 19  YearsSinceLastPromotion  4382 non-null   int64 
 20  YearsWithCurrManager     4382 non-null   int64 
dtypes: int32(2), int64(9), object(10)
memory usage: 718.9+ KB
```

In [14]:

```python
# check the no of unique values in every column for better data understanding
df.nunique()
```

Out[14]:

```
Age                          43
Attrition                     2
BusinessTravel                3
Department                    3
DistanceFromHome             29
Education                     5
EducationField                6
EmployeeID                 4382
Gender                        2
JobLevel                      5
JobRole                       9
MaritalStatus                 3
MonthlyIncome              1349
NumCompaniesWorked           10
PercentSalaryHike            15
StockOptionLevel              4
TotalWorkingYears            40
TrainingTimesLastYear         7
YearsAtCompany               37
YearsSinceLastPromotion      16
YearsWithCurrManager         18
dtype: int64
```

In [15]:

```python
# Looking at all the unique values in features

for i in df.columns:
    print("Unique values in",'',i)
    unique_vals = df[i].unique()
    print(unique_vals)
    print(50*'*')
```

```
Unique values in  Age
[51 31 32 38 46 28 29 25 45 36 55 47 37 21 35 26 50 53 44 49 42 18 41 39
 58 33 43 52 27 30 54 40 23 48 57 34 24 22 56 60 19 20 59]
**************************************************
Unique values in  Attrition
['No' 'Yes']
**************************************************
Unique values in  BusinessTravel
['Travel_Rarely' 'Travel_Frequently' 'Non-Travel']
**************************************************
Unique values in  Department
['Sales' 'Research & Development' 'Human Resources']
**************************************************
Unique values in  DistanceFromHome
[ 6 10 17  2  8 11 18  1  7 28 14  3 16  9  5  4 20 29 15 13 24 19 22 25
 21 26 27 12 23]
**************************************************
Unique values in  Education
[2 1 4 5 3]
**************************************************
```

```
Unique values in  EducationField
['Life Sciences' 'Other' 'Medical' 'Marketing' 'Technical Degree'
 'Human Resources']
**************************************************
Unique values in  EmployeeID
[1 2 3 ... 4407 4408 4409]
**************************************************
Unique values in  Gender
['Female' 'Male']
**************************************************
Unique values in  JobLevel
[1 4 3 2 5]
**************************************************
Unique values in  JobRole
['Healthcare Representative' 'Research Scientist' 'Sales Executive'
 'Human Resources' 'Research Director' 'Laboratory Technician'
 'Manufacturing Director' 'Sales Representative' 'Manager']
**************************************************
Unique values in  MaritalStatus
['Married' 'Single' 'Divorced']
**************************************************
Unique values in  MonthlyIncome
[131160  41890 193280 ...  48050  44340  83800]
**************************************************
Unique values in  NumCompaniesWorked
[1 0 3 4 2 7 9 5 6 8]
**************************************************
Unique values in  PercentSalaryHike
[11 23 15 12 13 20 22 21 17 14 16 18 19 24 25]
**************************************************
Unique values in  StockOptionLevel
[0 1 3 2]
**************************************************
Unique values in  TotalWorkingYears
[ 1  6  5 13  9 28 10 21 16 37  7  3 15  8 12 17 19 22  2  4 23  0 11 24
 25 20 14 26 18 30 36 31 33 32 34 40 29 35 27 38]
**************************************************
Unique values in  TrainingTimesLastYear
[6 3 2 5 4 0 1]
**************************************************
Unique values in  YearsAtCompany
[ 1  5  8  6  7  0  9 20 15 36 10  3 17  2  4 11 22 18 13 24 21 16 25 29
 27 14 31 32 34 26 12 19 33 30 23 37 40]
**************************************************
Unique values in  YearsSinceLastPromotion
[ 0  1  7  4 10  9  6  3  5  2  8 11 13 12 15 14]
**************************************************
Unique values in  YearsWithCurrManager
[ 0  4  3  5  7  8 10 11 13  9  1  2  6 12 17 16 15 14]
**************************************************
```

Dealing with Out layers....

In [16]:

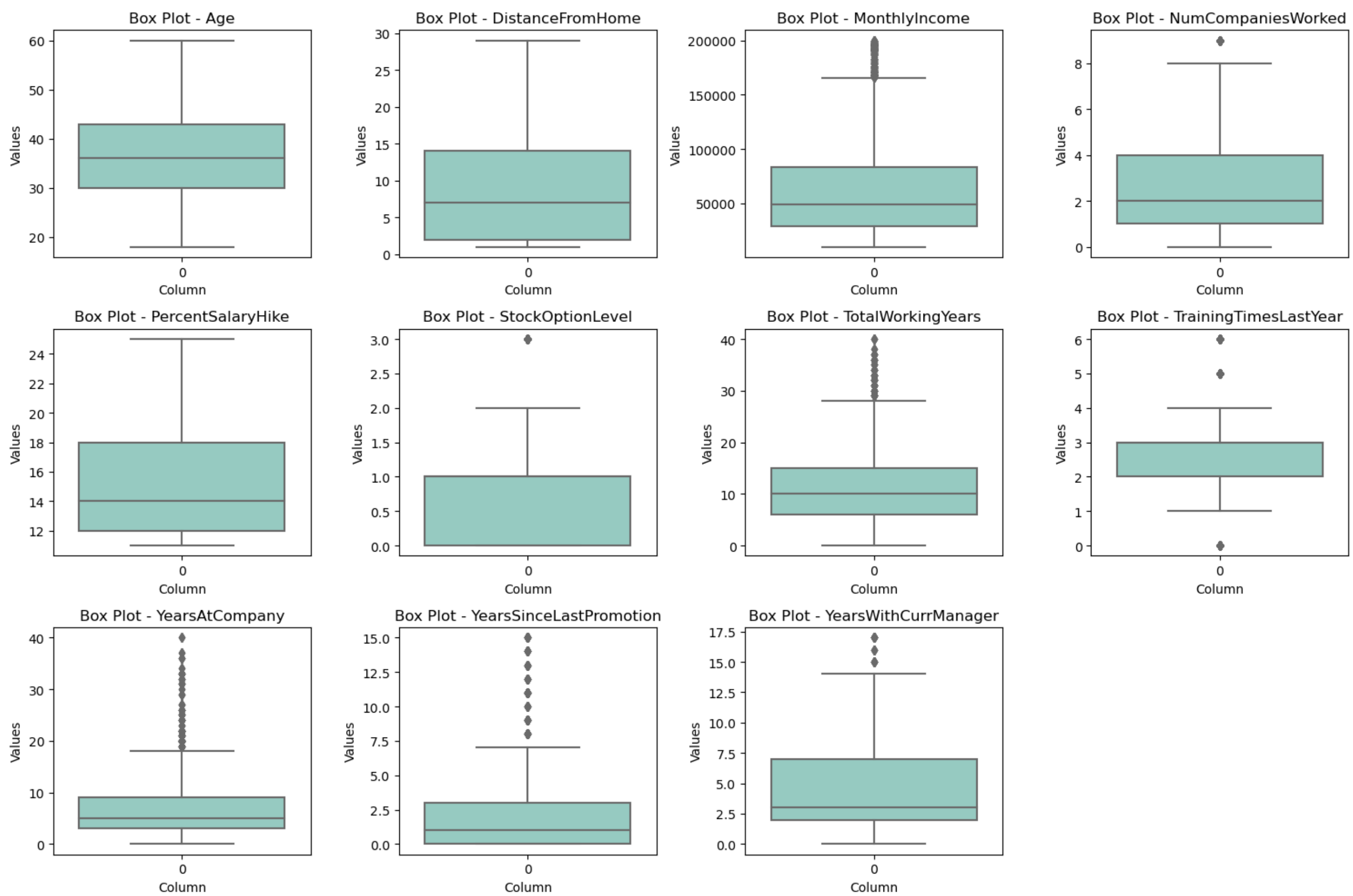
```
numerical_columns = df.select_dtypes(include='number')
numerical_columns.head()
```

Out[16]:

	Age	DistanceFromHome	MonthlyIncome	NumCompaniesWorked	PercentSalaryHike	StockOptionLevel	TotalW
0	51	6	131160	1	11	0	
1	31	10	41890	0	23	1	
2	32	17	193280	1	15	3	
3	38	2	83210	3	11	3	

4	32	10	23420	4	12	2

In [17]:
```python
# Create box plots for all columns one after the other
plt.figure(figsize=(15, 10))
for i, col in enumerate(numerical_columns.columns, 1):
    plt.subplot(3, 4, i)
    sns.boxplot(data=numerical_columns[col], palette='Set3')
    plt.title(f"Box Plot - {col}")
    plt.xlabel("Column")
    plt.ylabel("Values")
    plt.tight_layout()

plt.show()
```



In this code, we use sns.stripplot to create interactive strip plots for each column. The data=df parameter specifies the DataFrame, y=col sets the column on the y-axis, and color='blue' defines the color of the data points in the plot.

This is because swarmplots are slower to render on Jupyter notebooks.

Dealing with duplicate values ....

In [18]:
```python
df.duplicated().sum()
```

Out[18]: 0

there are no duplicate values in the column

Dealing with data cleaning ....

In [19]:
```
df.columns
```

Out[19]:
```
Index(['Age', 'Attrition', 'BusinessTravel', 'Department', 'DistanceFromHome',
       'Education', 'EducationField', 'EmployeeID', 'Gender', 'JobLevel',
       'JobRole', 'MaritalStatus', 'MonthlyIncome', 'NumCompaniesWorked',
       'PercentSalaryHike', 'StockOptionLevel', 'TotalWorkingYears',
       'TrainingTimesLastYear', 'YearsAtCompany', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

the column names should be changed unable to understand

In [20]:
```
#columns names should be change
new_column_names = {
    'BusinessTravel': 'Business_Travel',
    'DistanceFromHome': 'Distance_From_Home',
    'EducationField': 'Education_Field',
    'EmployeeID': 'Employee_ID',
    'JobLevel': 'Job_Level',
    'JobRole': 'Job_Role',
    'MaritalStatus': 'Marital_Status',
    'MonthlyIncome': 'Monthly_Income',
    'NumCompaniesWorked': 'Num_Companies_Worked',
    'PercentSalaryHike': 'Percent_Salary_Hike',
    'StockOptionLevel': 'Stock_Option_Level',
    'TotalWorkingYears': 'Total_Working_Years',
    'TrainingTimesLastYear': 'Training_Times_Last_Year',
    'YearsAtCompany': 'Years_At_Company',
    'YearsSinceLastPromotion': 'Years_Since_Last_Promotion',
    'YearsWithCurrManager': 'Years_With_Curr_Manager'
}
df = df.rename(columns=new_column_names)
df.head(10)
```

Out[20]:

	Age	Attrition	Business_Travel	Department	Distance_From_Home	Education	Education_Field	Employee_ID
0	51	No	Travel_Rarely	Sales	6	2	Life Sciences	1
1	31	Yes	Travel_Frequently	Research & Development	10	1	Life Sciences	2
2	32	No	Travel_Frequently	Research & Development	17	4	Other	3
3	38	No	Non-Travel	Research & Development	2	5	Life Sciences	4
4	32	No	Travel_Rarely	Research & Development	10	1	Medical	5
5	46	No	Travel_Rarely	Research & Development	8	3	Life Sciences	6
6	28	Yes	Travel_Rarely	Research & Development	11	2	Medical	7
7	29	No	Travel_Rarely	Research & Development	18	3	Life Sciences	8
8	31	No	Travel_Rarely	Research & Development	1	3	Life Sciences	9
9	25	No	Non-Travel	Research & Development	7	4	Medical	10

10 rows × 21 columns

Generate Extended Data Dictionary (EDD) of the provided dataset to help comment on data quality

In [21]:

```
summary_df = pd.DataFrame({
    'min': df.min(),
    'max': df.max(),
    'median': df.median(),
    'mean': df.mean(),
    'std': df.std(),
    '5%ile': df.quantile(0.05),
    '10%ile': df.quantile(0.10),
    '25%ile': df.quantile(0.25),
    '50%ile': df.quantile(0.50),
    '75%ile': df.quantile(0.75),
    '90%ile': df.quantile(0.90),
    '95%ile': df.quantile(0.95),
    'count': df.count(),
    'na': df.isnull().sum()})

summary_df.head(21)
```

Out[21]:

	min	max	median	mean	std	5%ile	10%ile
Age	18	60	36.0	36.933364	9.137272	24.0	26.0
Attrition	No	Yes	NaN	NaN	NaN	NaN	NaN
Business_Travel	Non-Travel	Travel_Rarely	NaN	NaN	NaN	NaN	NaN
Department	Human Resources	Sales	NaN	NaN	NaN	NaN	NaN
Distance_From_Home	1	29	7.0	9.198996	8.105396	1.0	1.0
Education	1	5	3.0	2.912369	1.024728	NaN	NaN
Education_Field	Human Resources	Technical Degree	NaN	NaN	NaN	NaN	NaN
Employee_ID	1	4409	2208.5	2207.804884	1271.688783	NaN	NaN
Gender	Female	Male	NaN	NaN	NaN	NaN	NaN
Job_Level	1	5	2.0	2.063898	1.106115	NaN	NaN
Job_Role	Healthcare Representative	Sales Representative	NaN	NaN	NaN	NaN	NaN
Marital_Status	Divorced	Single	NaN	NaN	NaN	NaN	NaN
Monthly_Income	10090	199990	49190.0	65061.702419	47142.310175	20970.0	23140.0
Num_Companies_Worked	0	9	2.0	2.693291	2.497832	0.0	0.0
Percent_Salary_Hike	11	25	14.0	15.210634	3.663007	11.0	11.0
Stock_Option_Level	0	3	1.0	0.794614	0.852397	0.0	0.0
Total_Working_Years	0	40	10.0	11.290278	7.785717	1.0	3.0
Training_Times_Last_Year	0	6	3.0	2.798266	1.289402	1.0	2.0
Years_At_Company	0	40	5.0	7.010497	6.129351	1.0	1.0
Years_Since_Last_Promotion	0	15	1.0	2.191693	3.224994	0.0	0.0

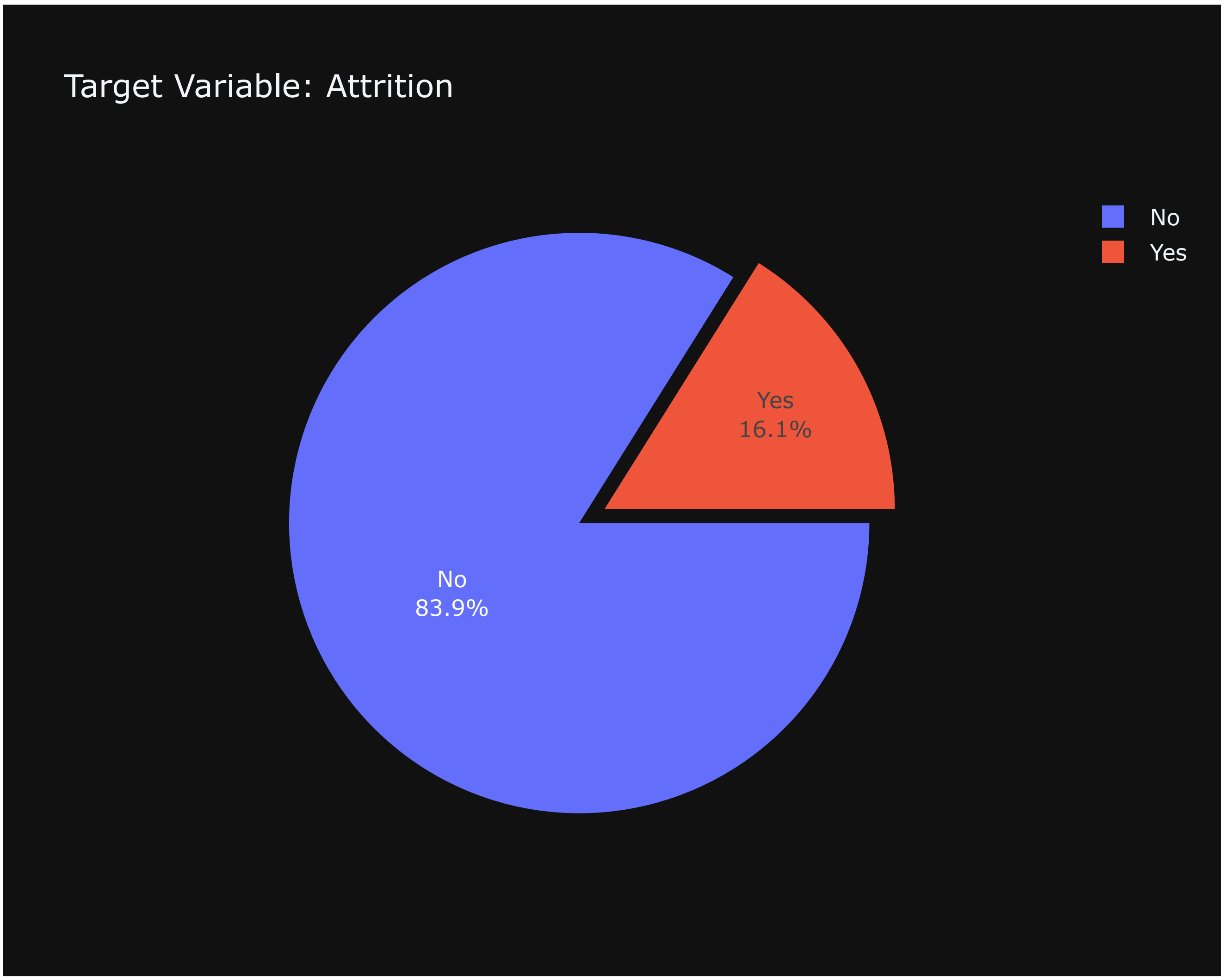
Years_With_Curr_Manager	0	17	3.0	4.126198	3.569674	0.0	0.0

From the above table, we can observe the Extended Data Dictionary of every column and get insights for every column.....

Section 03: Exploratory Data Analysis

Attrition Happened

In [22]:
```
# Visualizing target variable classes and its distribution among the dataset
fig = px.pie(df, names = 'Attrition', title = 'Target Variable: Attrition', template = '
fig.update_traces(rotation=90, pull = [0.1], textinfo = "percent+label")
fig.show()
```



Univariate Analysis

Numerical Features

In [23]:
```
Attrition = df.query("Attrition == 'Yes' ")
Attrition
```

Out[23]:

	Age	Attrition	Business_Travel	Department	Distance_From_Home	Education	Education_Field	Employee_I

1	31	Yes	Travel_Frequently	Research & Development	10	1	Life Sciences	
6	28	Yes	Travel_Rarely	Research & Development	11	2	Medical	
13	47	Yes	Non-Travel	Research & Development	1	1	Medical	1
28	44	Yes	Travel_Frequently	Research & Development	1	2	Medical	2
30	26	Yes	Travel_Rarely	Research & Development	4	3	Medical	3
...	...	...	...	...	...	...	...	
4381	29	Yes	Travel_Rarely	Research & Development	7	1	Life Sciences	438
4386	33	Yes	Travel_Rarely	Sales	11	4	Marketing	438
4388	33	Yes	Travel_Rarely	Sales	1	3	Life Sciences	438
4391	32	Yes	Travel_Rarely	Sales	23	1	Life Sciences	439
4402	37	Yes	Travel_Frequently	Sales	2	3	Marketing	440

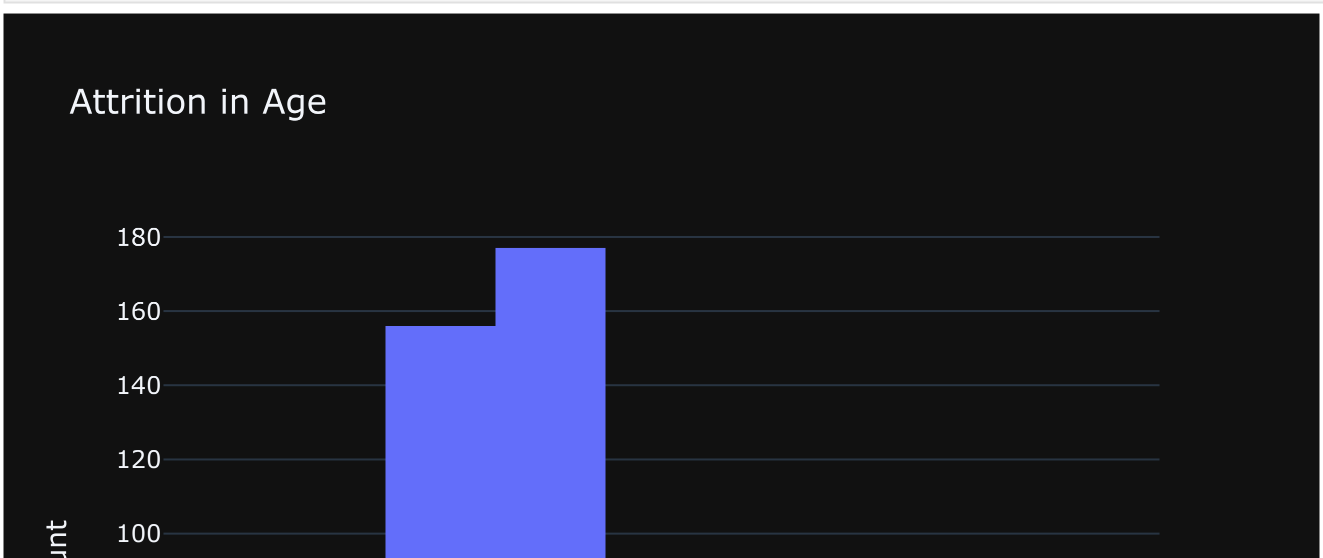
705 rows × 21 columns

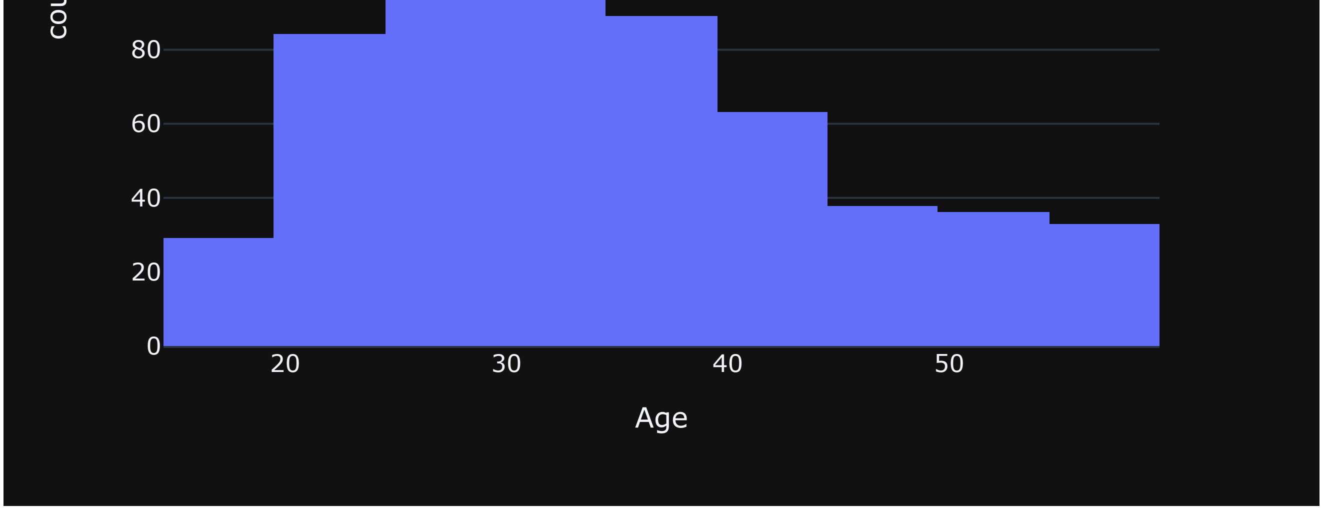
In [24]:

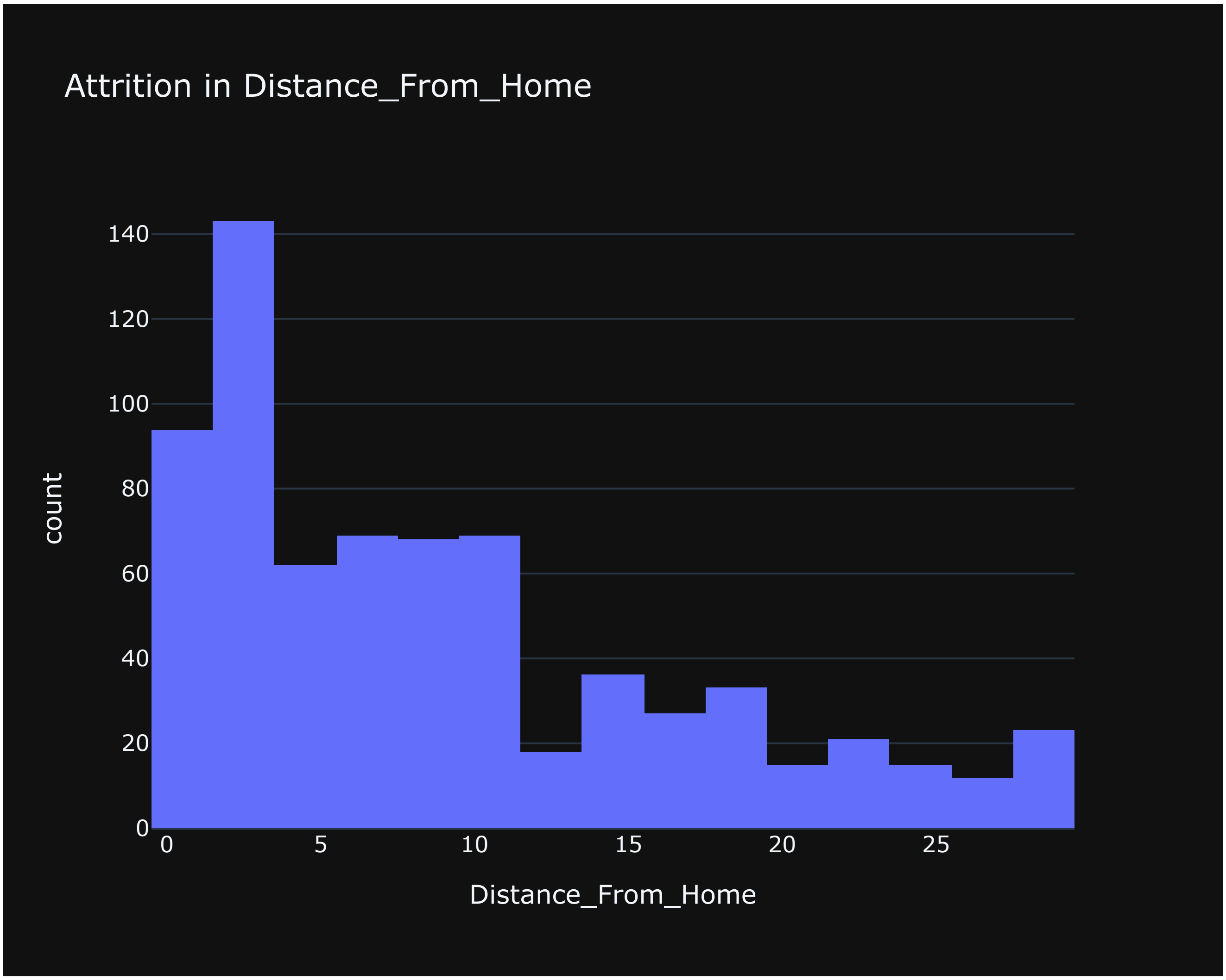
```
# Creating a function to plot histograms on numeric features
def numeric_plot(i):
    fig = px.histogram(Attrition, x = Attrition[i], nbins  = 20, template = 'plotly_dark
                       title = f'Attrition in {i}')
    fig.show()
```

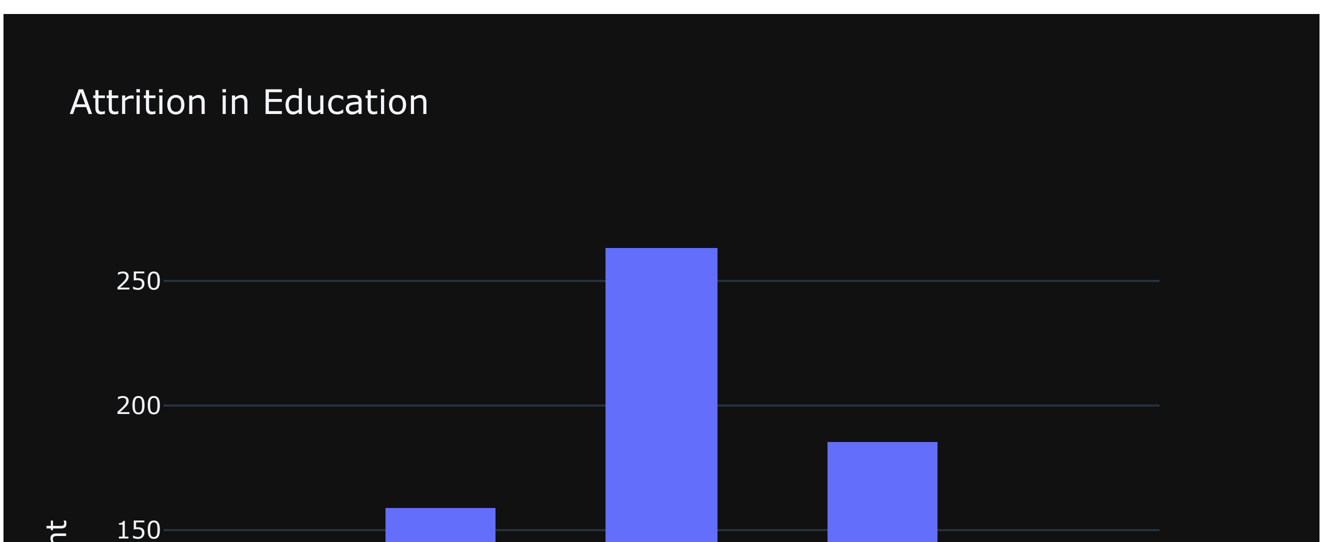
In [25]:

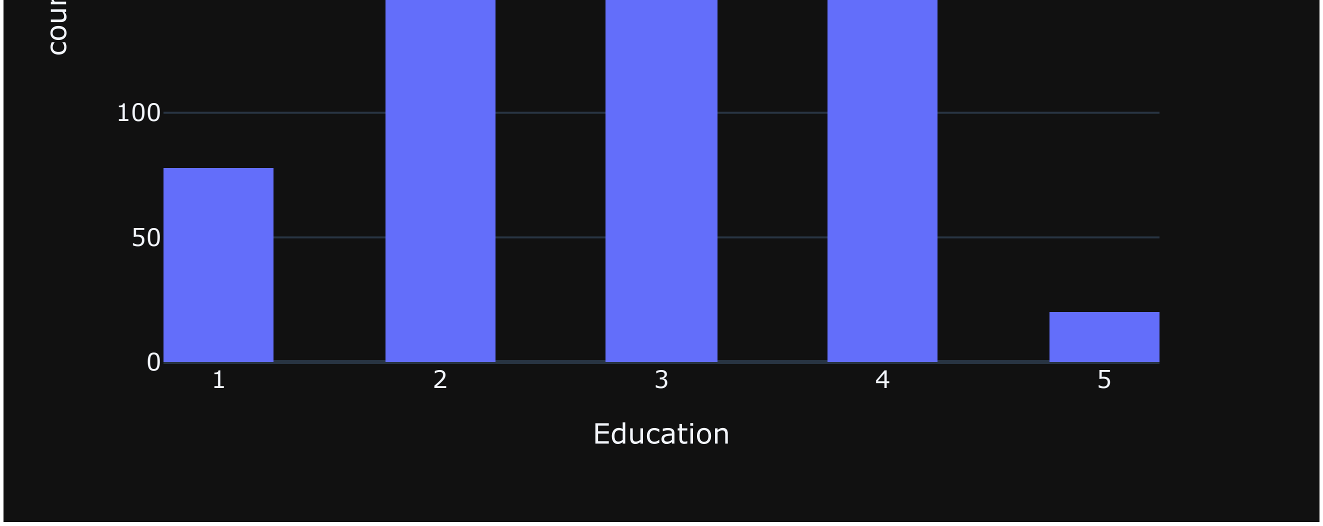
```
numeric_plot('Age')
numeric_plot('Distance_From_Home')
numeric_plot('Education')
numeric_plot('Job_Level')
numeric_plot('Monthly_Income')
numeric_plot('Num_Companies_Worked')
numeric_plot('Percent_Salary_Hike')
numeric_plot('Stock_Option_Level')
numeric_plot('Total_Working_Years')
numeric_plot('Training_Times_Last_Year')
numeric_plot('Years_At_Company')
numeric_plot('Years_Since_Last_Promotion')
numeric_plot('Years_With_Curr_Manager')
```



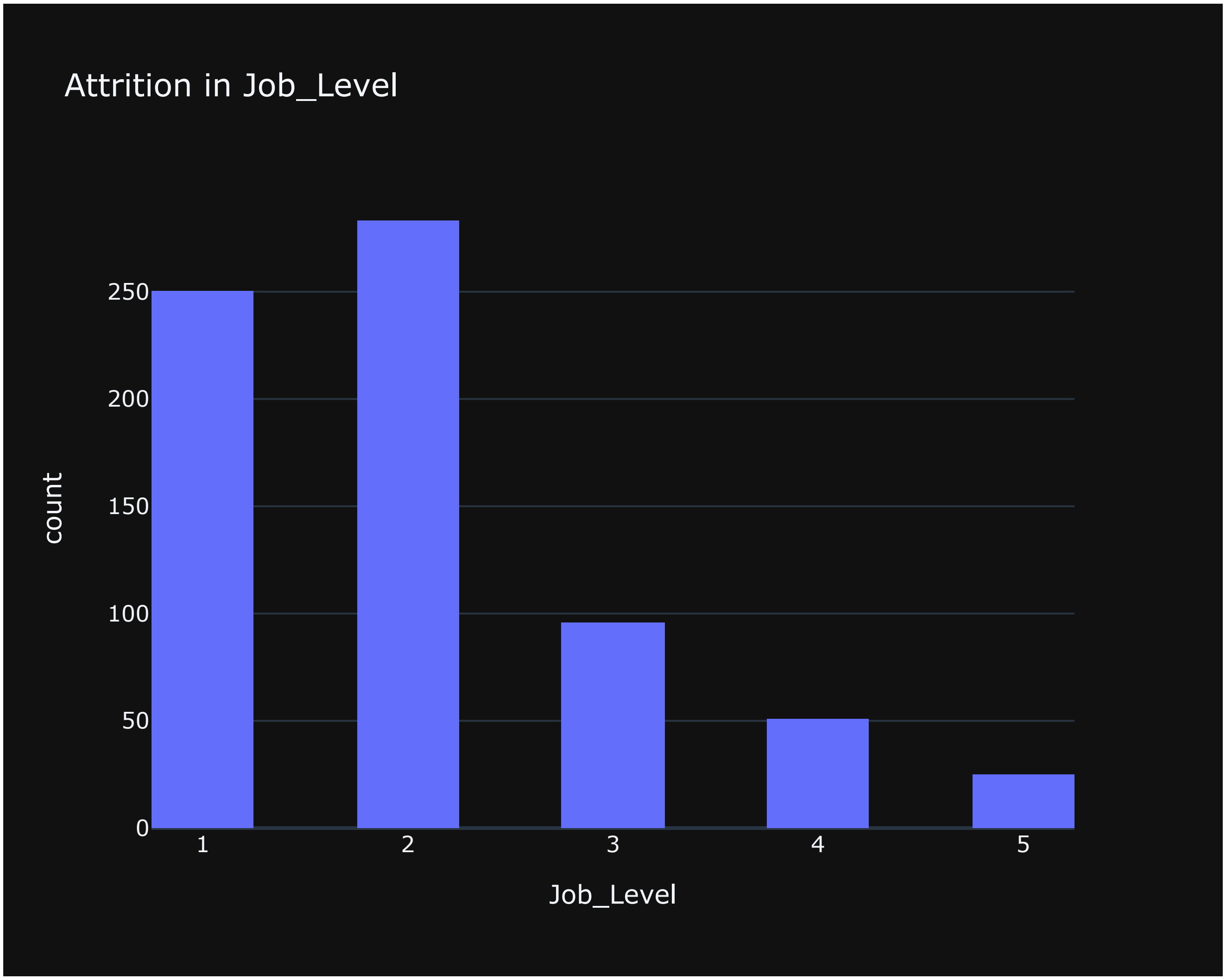

count
80
60
40
20
0
20
30
40
50
Age


Attrition in Distance_From_Home
140
120
100
80
60
40
20
0
count
0
5
10
15
20
25
Distance_From_Home

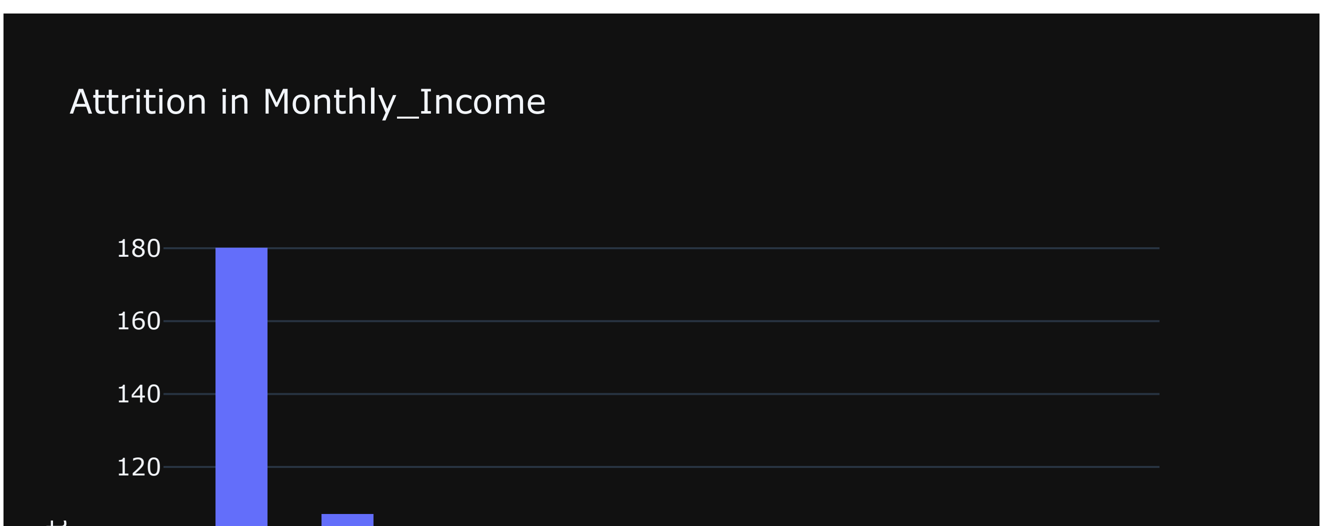

Attrition in Education
250
200
150



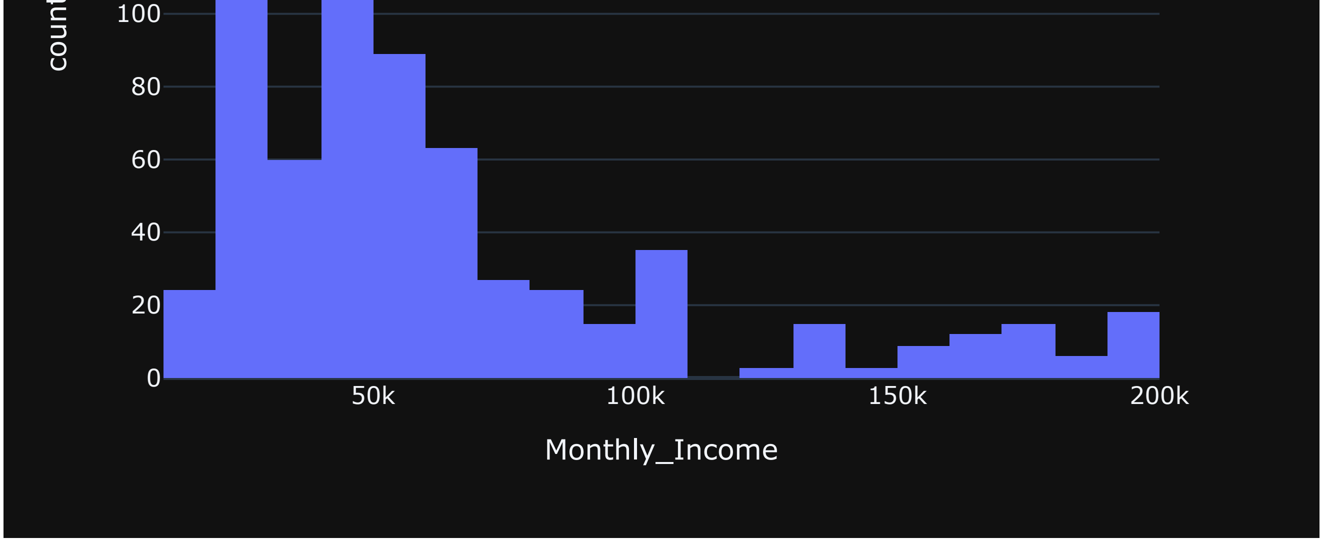
count
100
50
0
1
2
3
4
5
Education

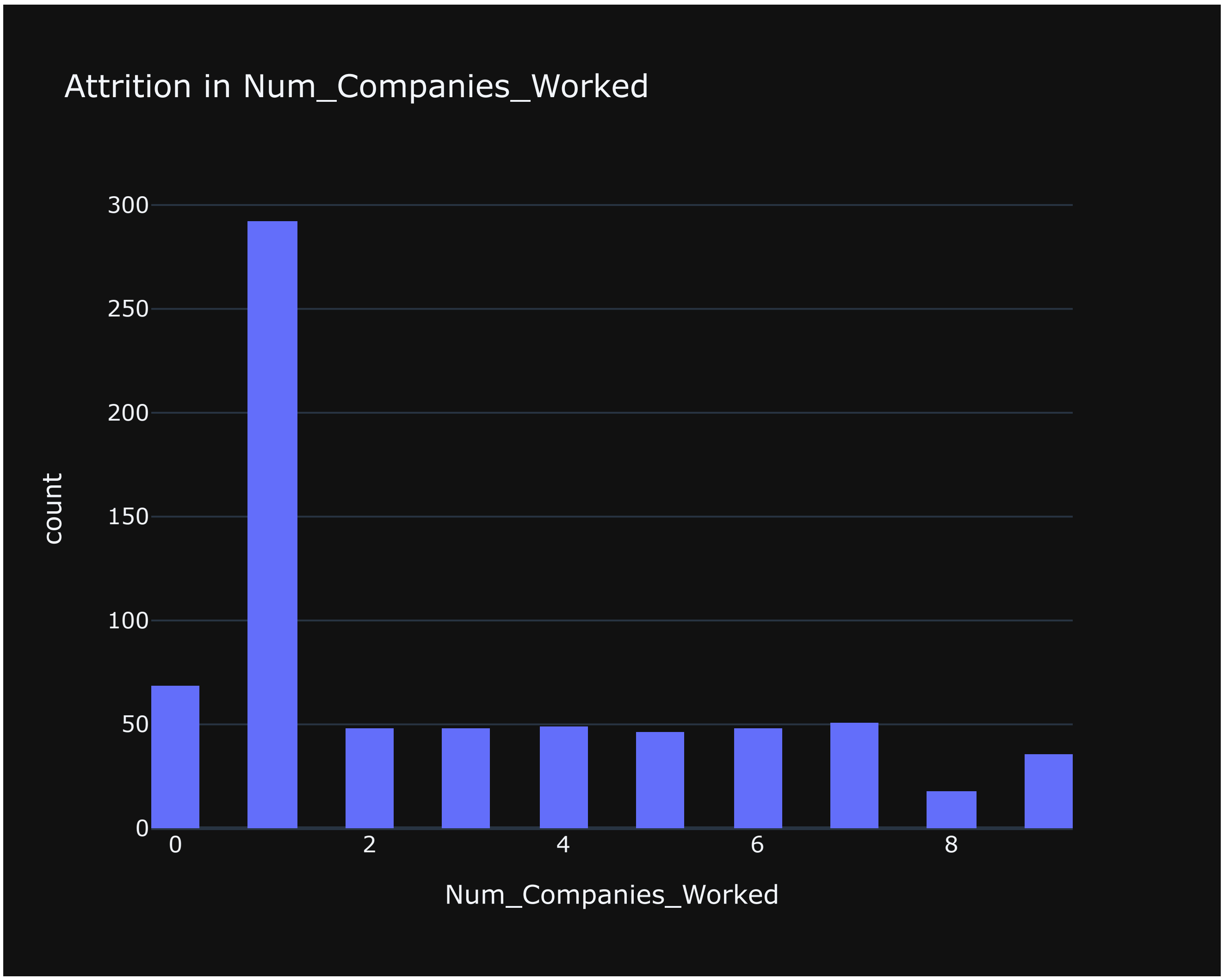


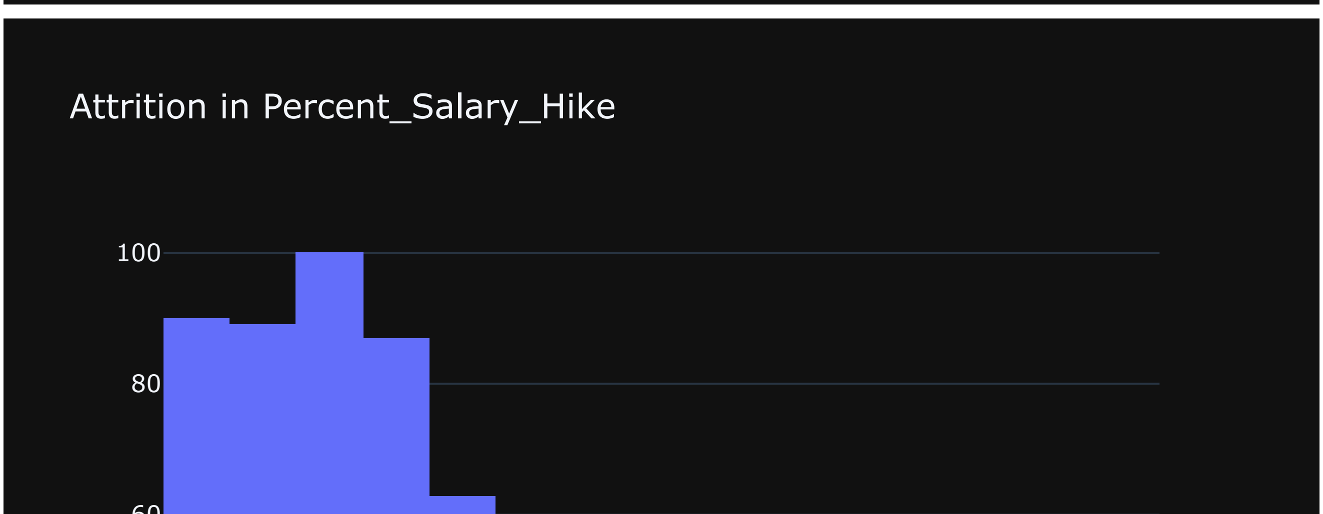
Attrition in Job_Level
count
250
200
150
100
50
0
1
2
3
4
5
Job_Level

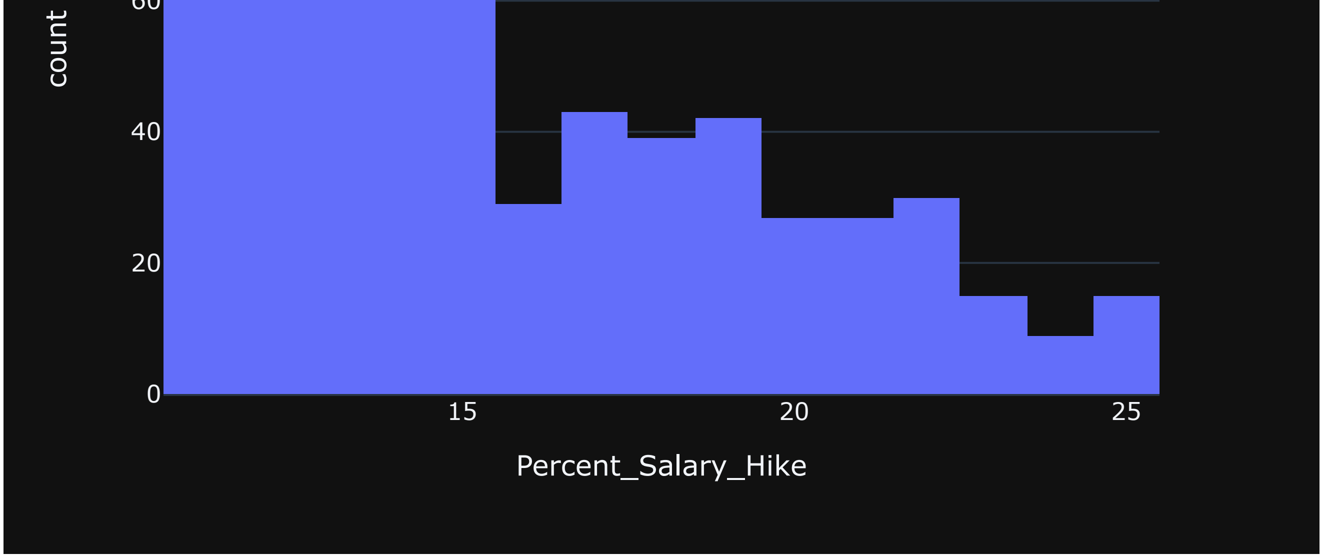


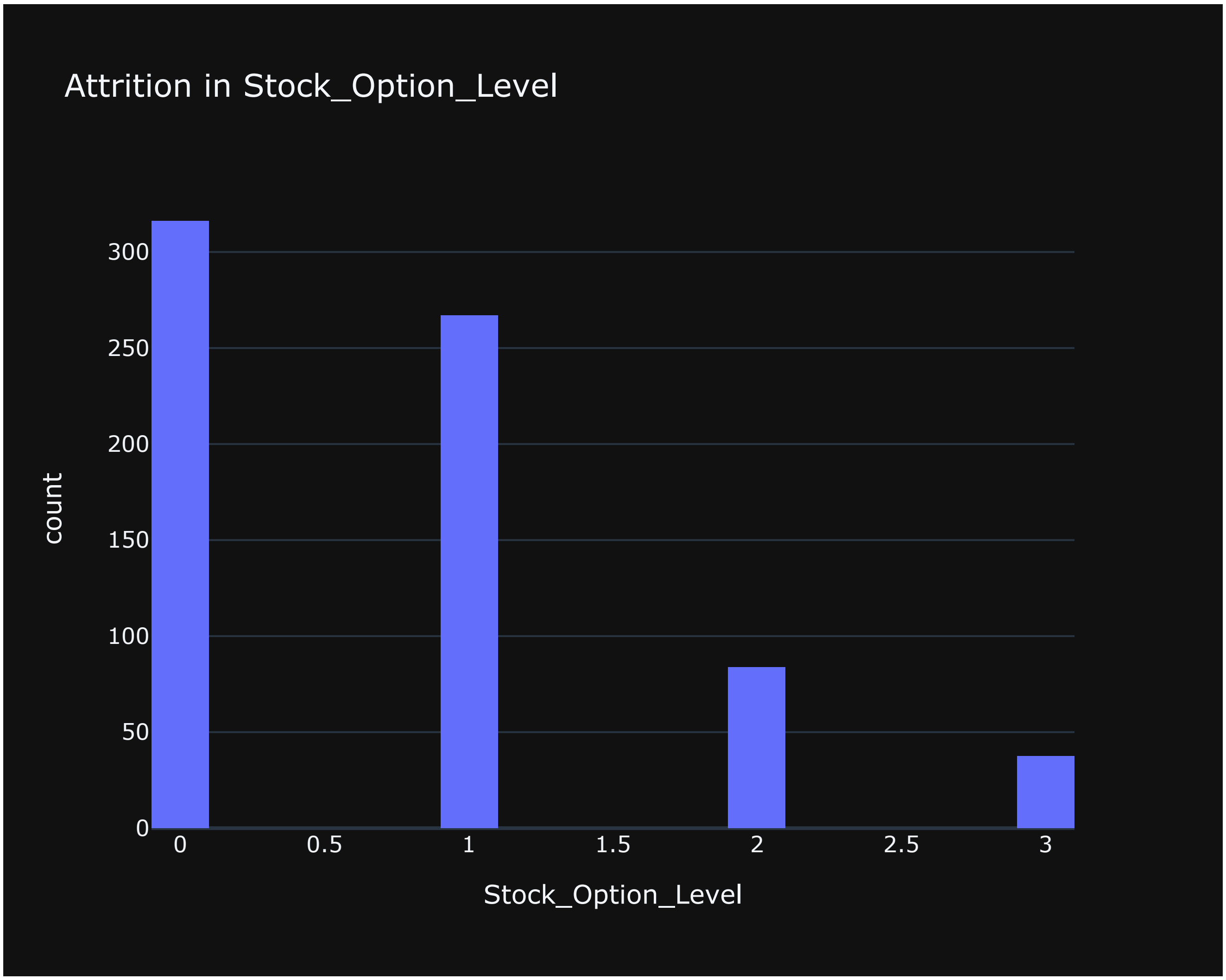
Attrition in Monthly_Income
180
160
140
120

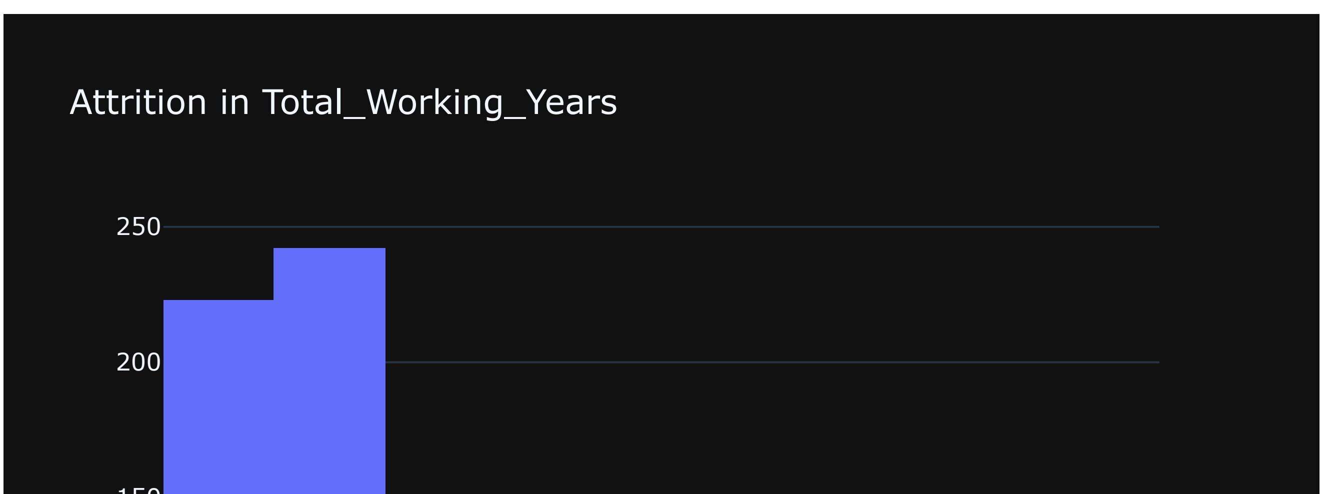

count
100
80
60
40
20
0
50k
100k
150k
200k
Monthly_Income

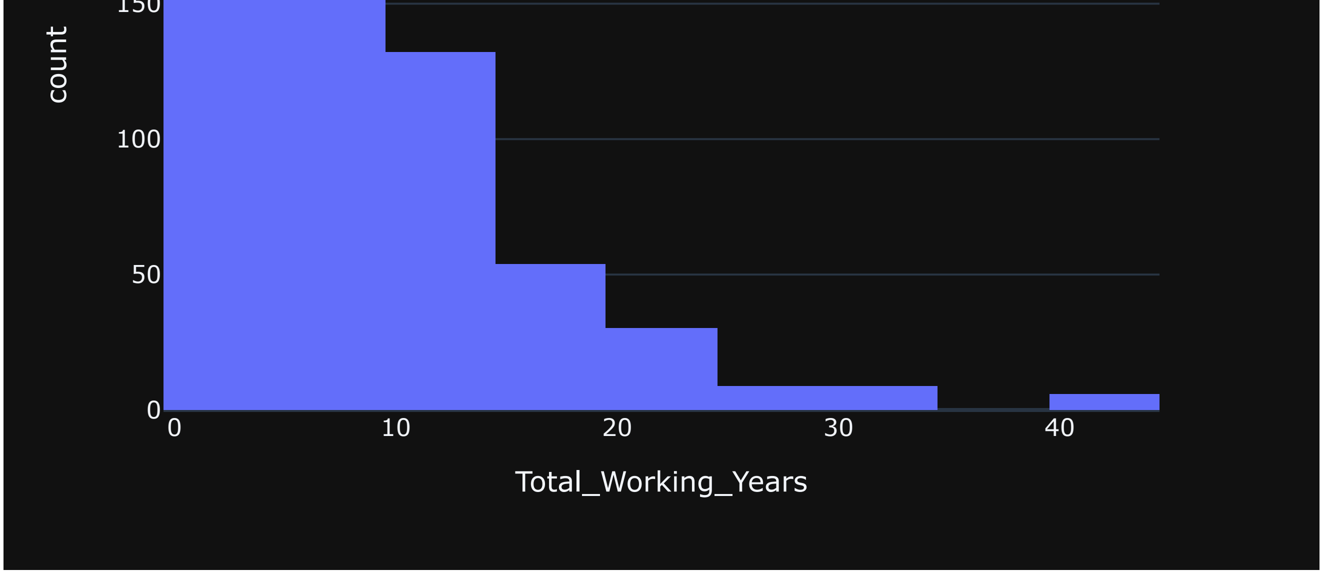

Attrition in Num_Companies_Worked
300
250
200
150
100
50
0
count
0
2
4
6
8
Num_Companies_Worked

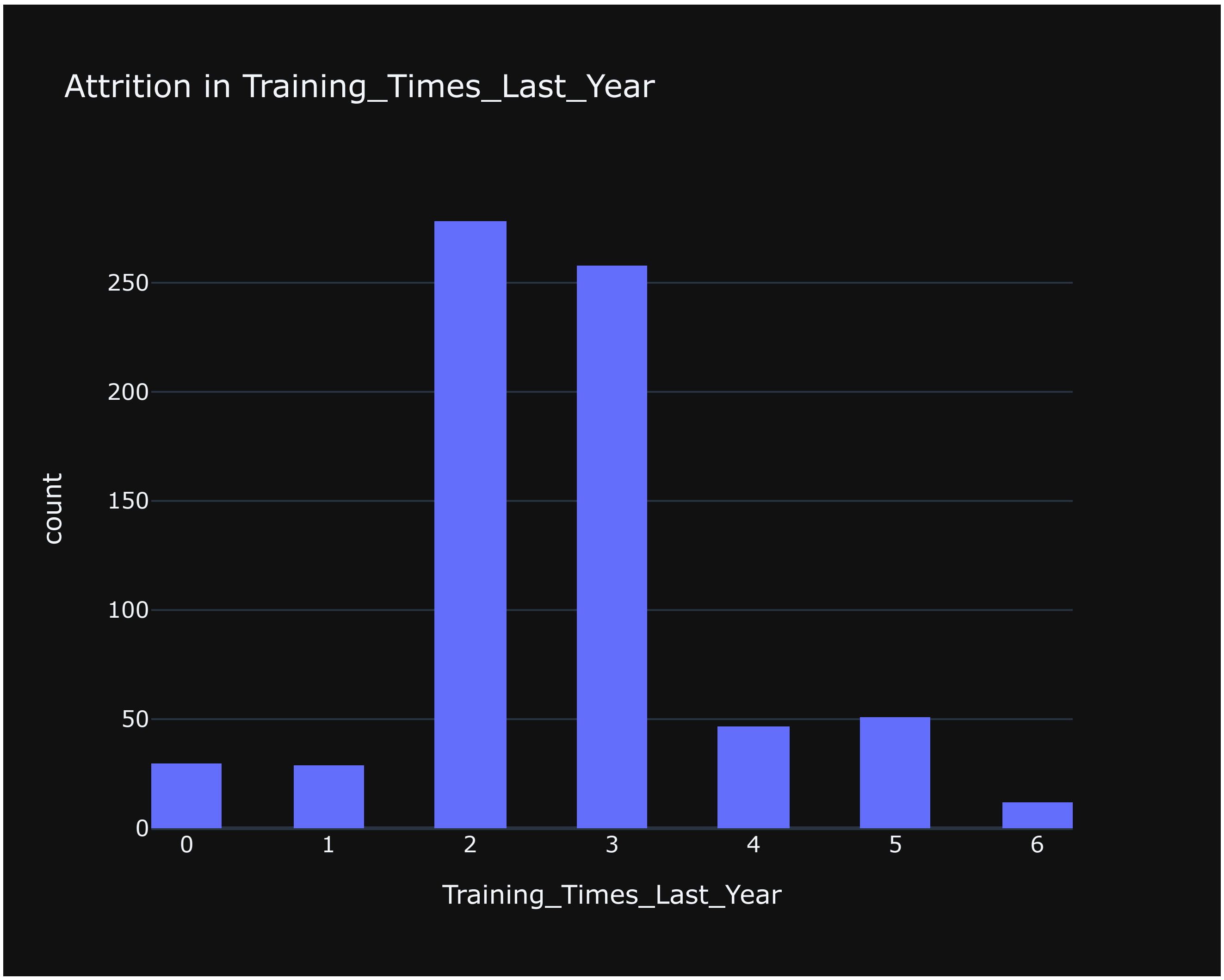

Attrition in Percent_Salary_Hike
100
80
60

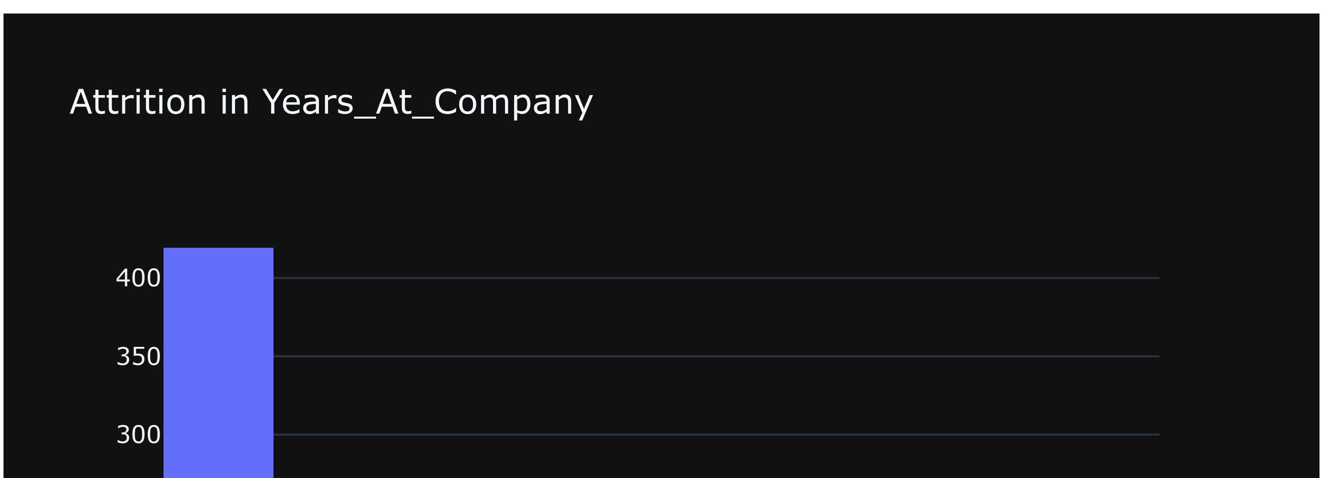

count
0
20
40
60
15
20
25
Percent_Salary_Hike

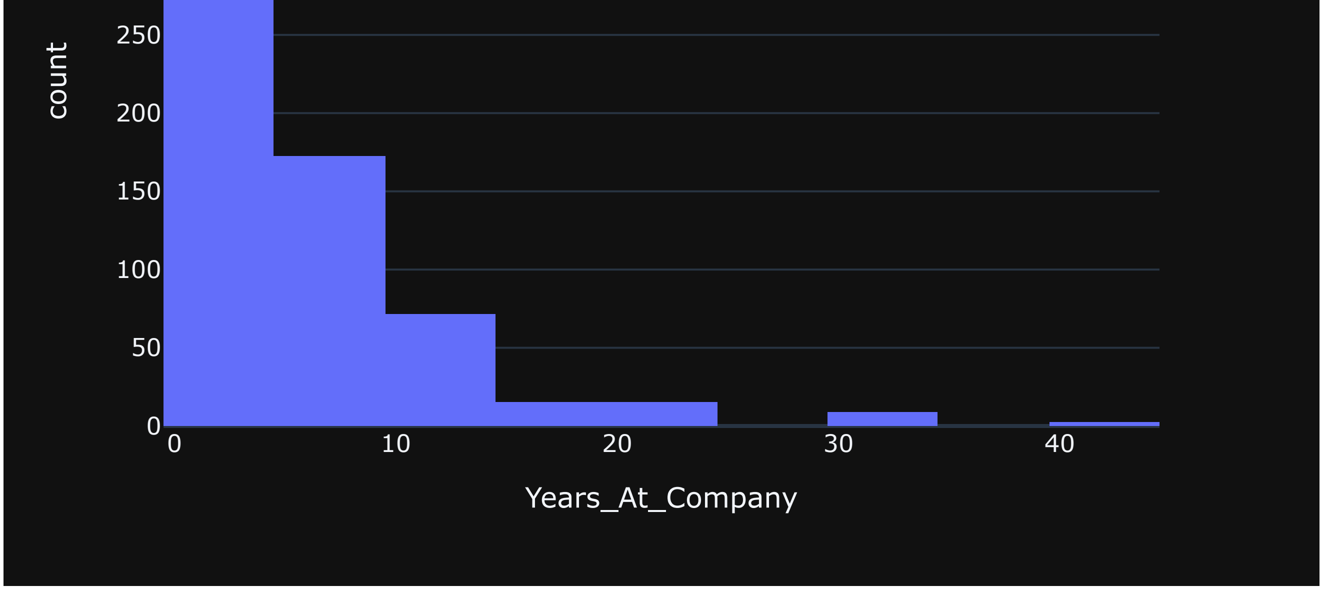

Attrition in Stock_Option_Level
count
0
50
100
150
200
250
300
0
0.5
1
1.5
2
2.5
3
Stock_Option_Level

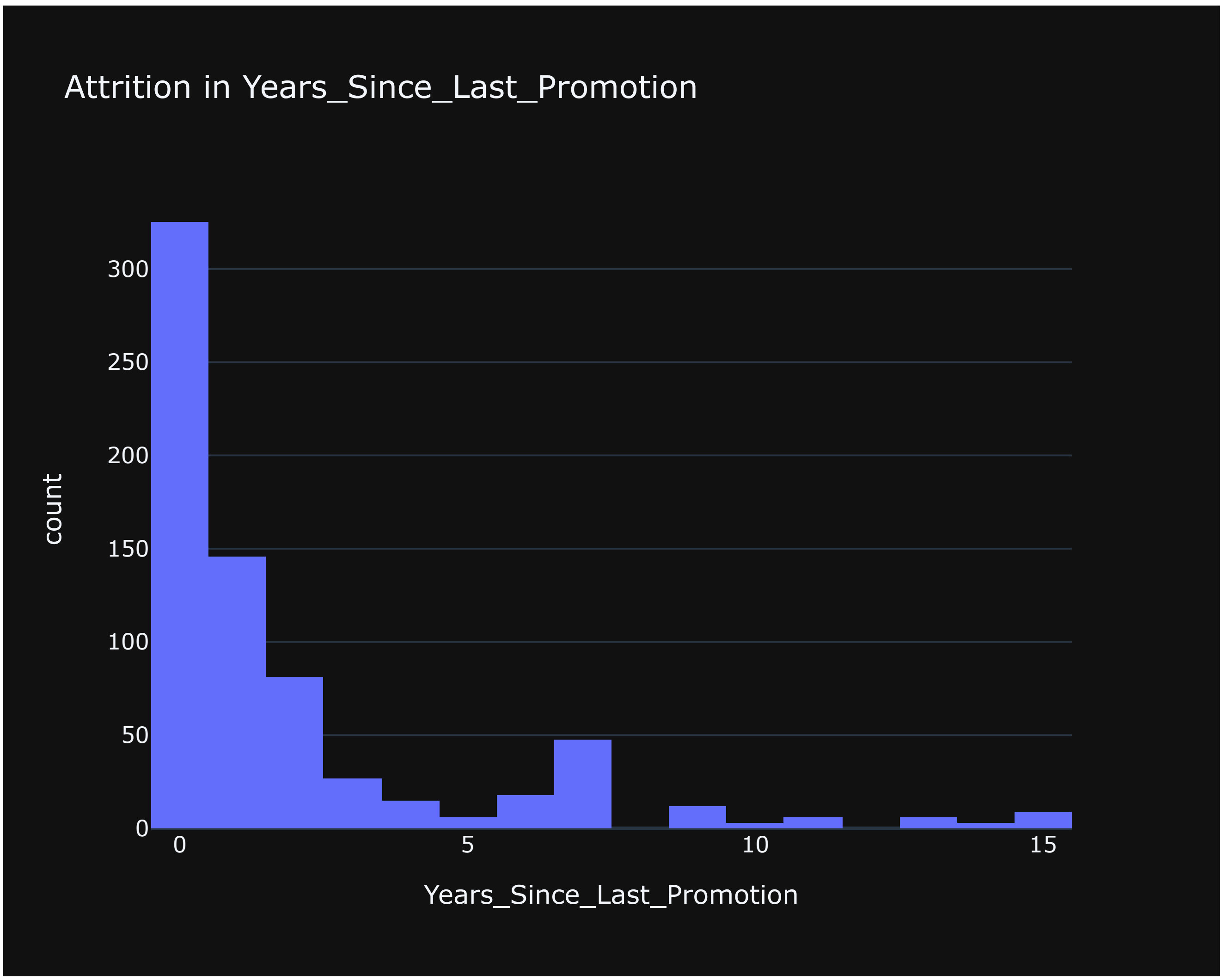

Attrition in Total_Working_Years
200
250

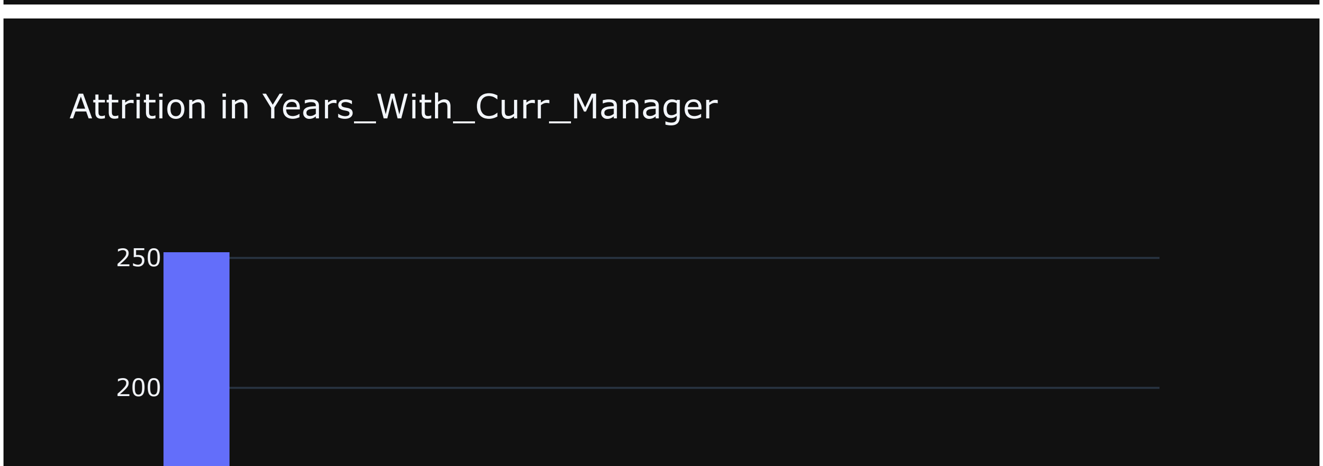

count
150
100
50
0
0
10
20
30
40
Total_Working_Years

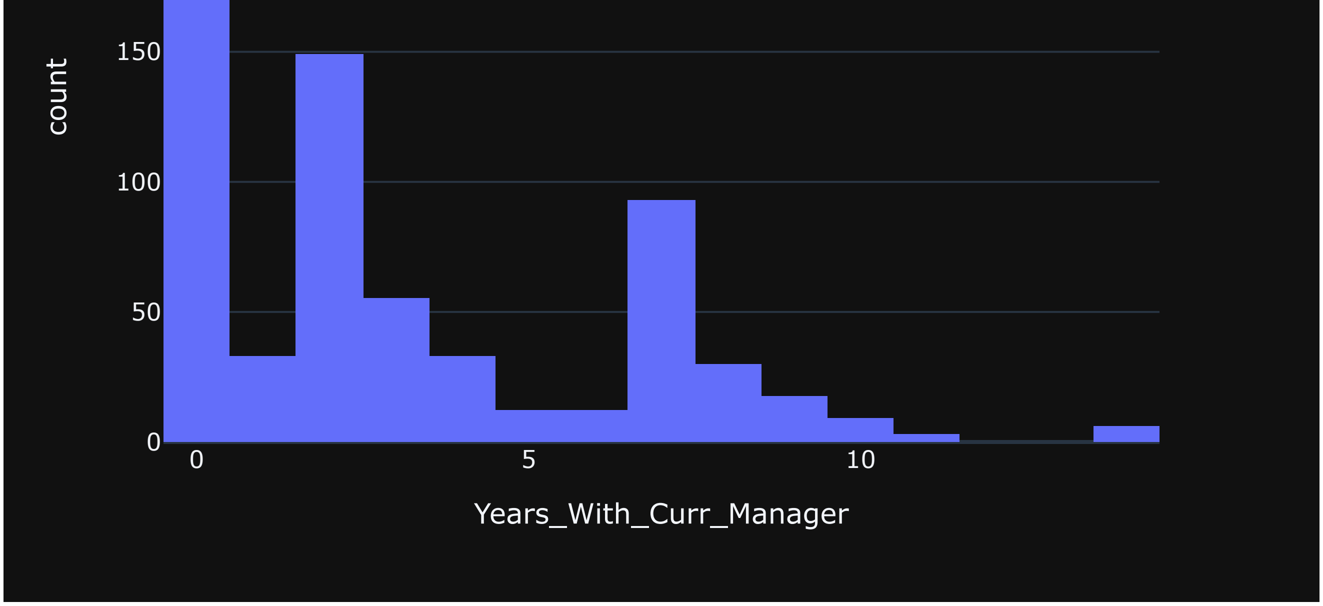

Attrition in Training_Times_Last_Year
count
250
200
150
100
50
0
0
1
2
3
4
5
6
Training_Times_Last_Year


Attrition in Years_At_Company
400
350
300


count
250
200
150
100
50
0
0
10
20
30
40
Years_At_Company


Attrition in Years_Since_Last_Promotion
count
300
250
200
150
100
50
0
0
5
10
15
Years_Since_Last_Promotion


Attrition in Years_With_Curr_Manager
250
200



A few observations can be made based on the information and histograms for numerical features:

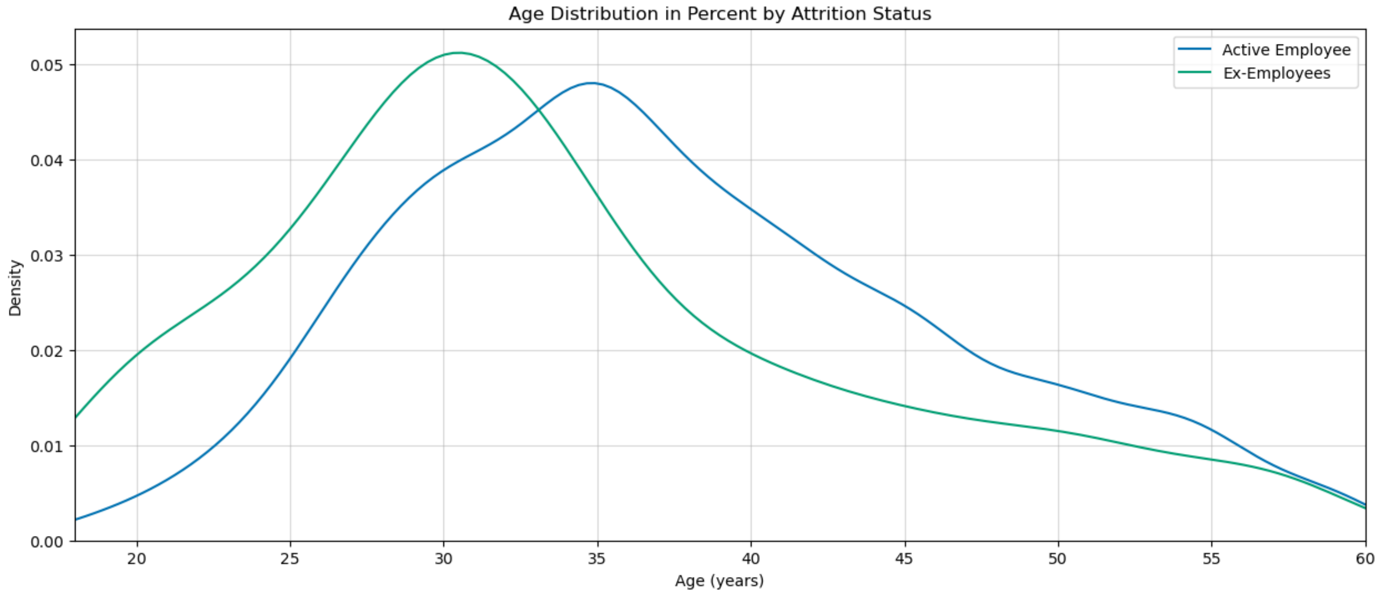
1. Many histograms are tail-heavy; indeed several distributions are right-skewed (e.g. MonthlyIncome DistanceFromHome, YearsAtCompany). Data transformation methods may be required to approach a normal distribution prior to fitting a model to the data.

2. Age distribution is a slightly right-skewed normal distribution with the bulk of the staff between 25 and 34 years old.

Feature distribution by target attribute

Age

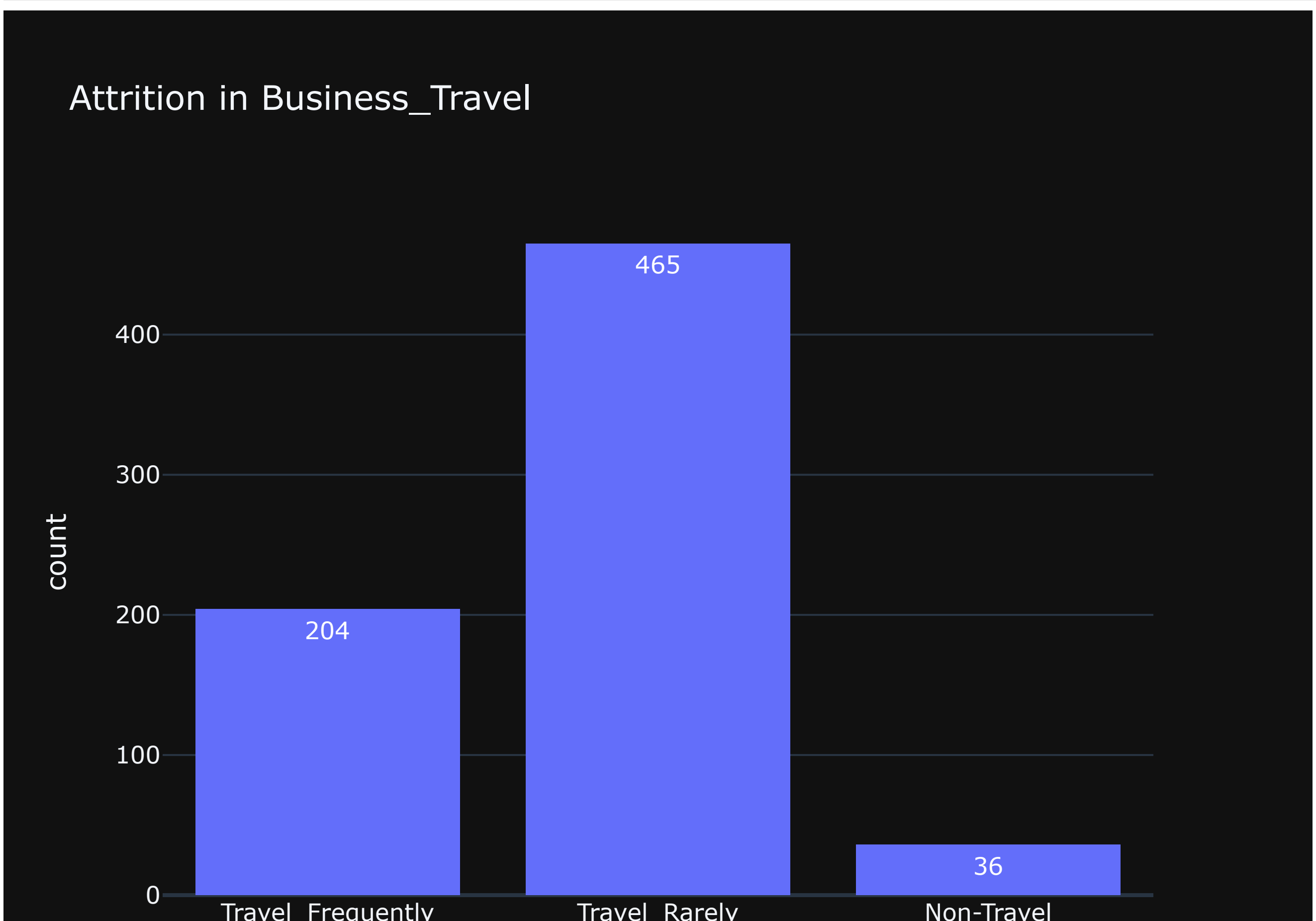
In [26]:
```
plt.figure(figsize=(15,6))
plt.style.use('seaborn-colorblind')
plt.grid(True, alpha=0.5)
sns.kdeplot(df.loc[df['Attrition'] == 'No', 'Age'], label = 'Active Employee')
sns.kdeplot(df.loc[df['Attrition'] == 'Yes', 'Age'], label = 'Ex-Employees')
plt.xlim(left=18, right=60)
plt.xlabel('Age (years)')
plt.ylabel('Density')
plt.legend()
plt.title('Age Distribution in Percent by Attrition Status');
```

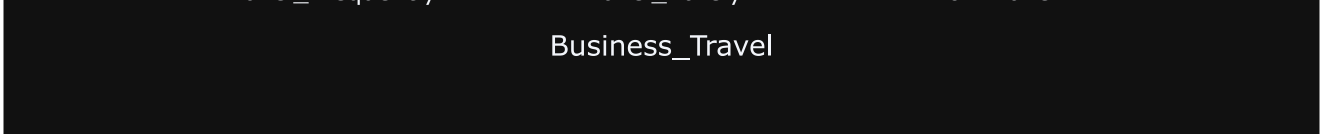


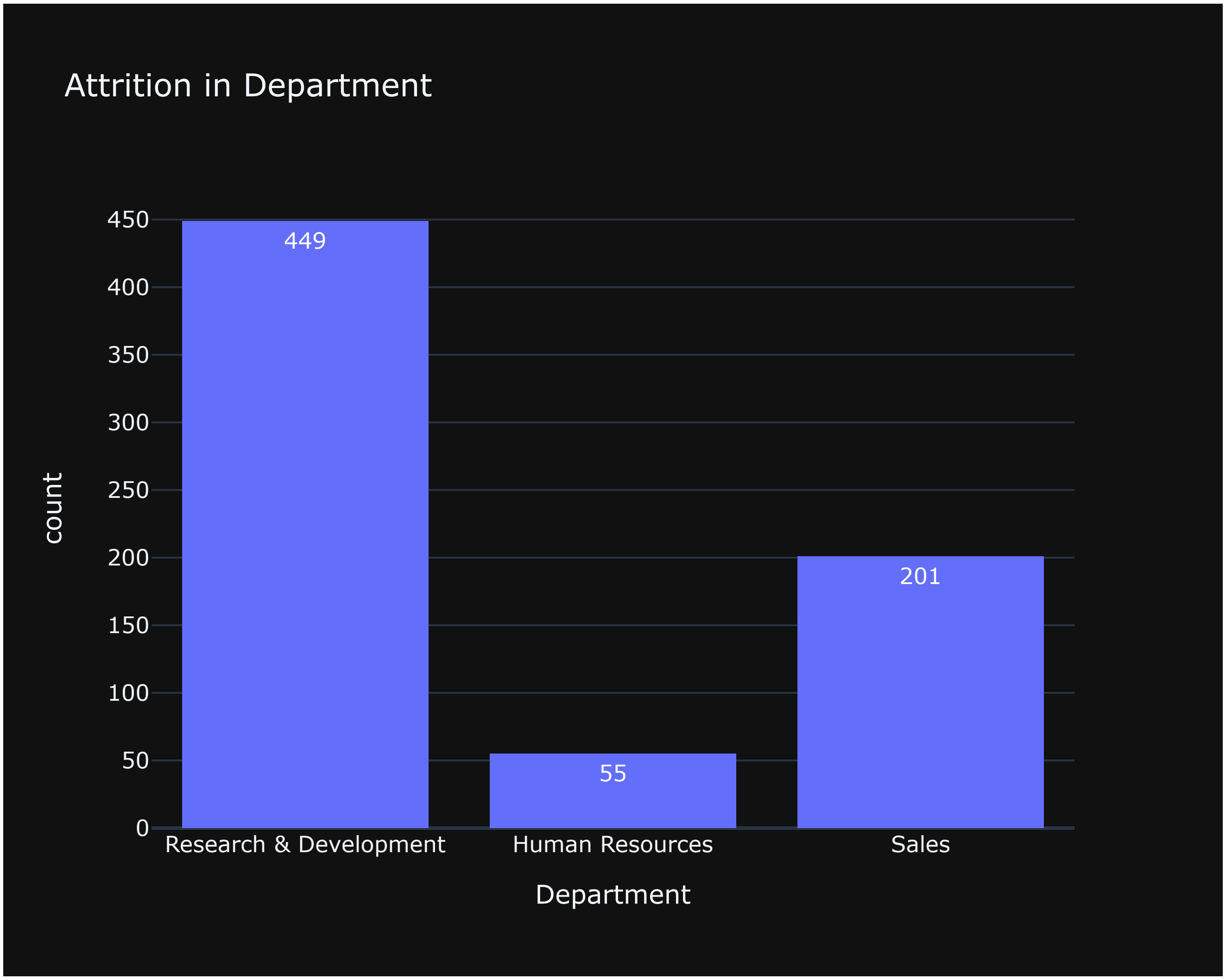
Categorical Features

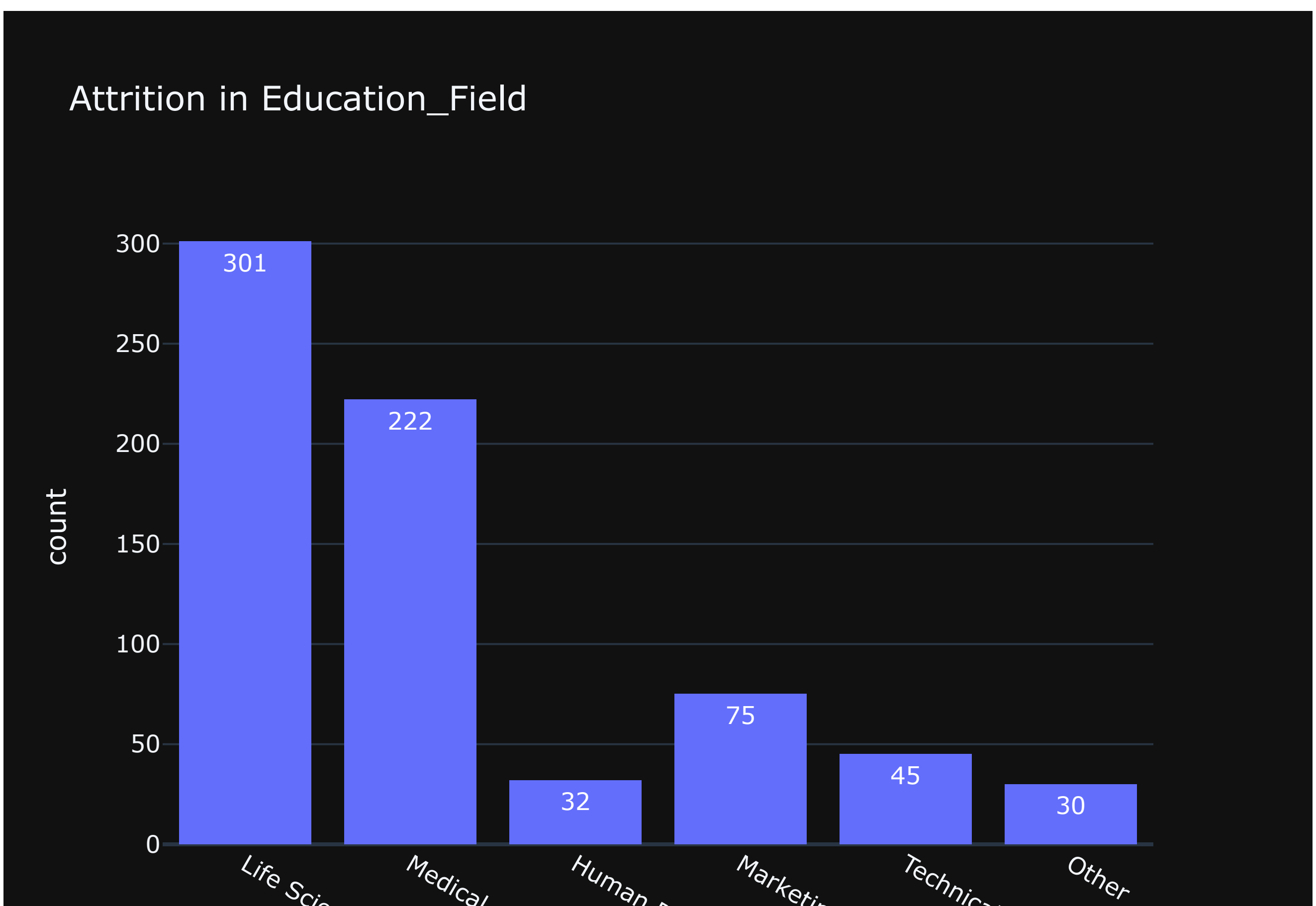
In [27]:
```
# Creating a function to plot histograms
def barplot(i):
    fig = px.histogram(Attrition, x = Attrition[i], template = 'plotly_dark',
                      title = f'Attrition in {i}', text_auto = 'd3-format')
    fig.show()
```

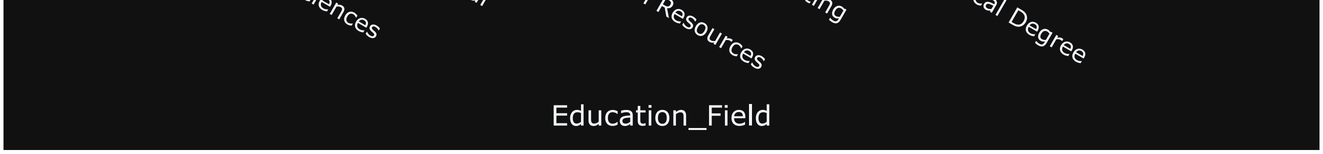
In [28]:
```
# Creating visualizations for categorical values
barplot('Business_Travel')
barplot('Department')
barplot('Education_Field')
barplot('Gender')
barplot('Job_Role')
barplot('Marital_Status')
```

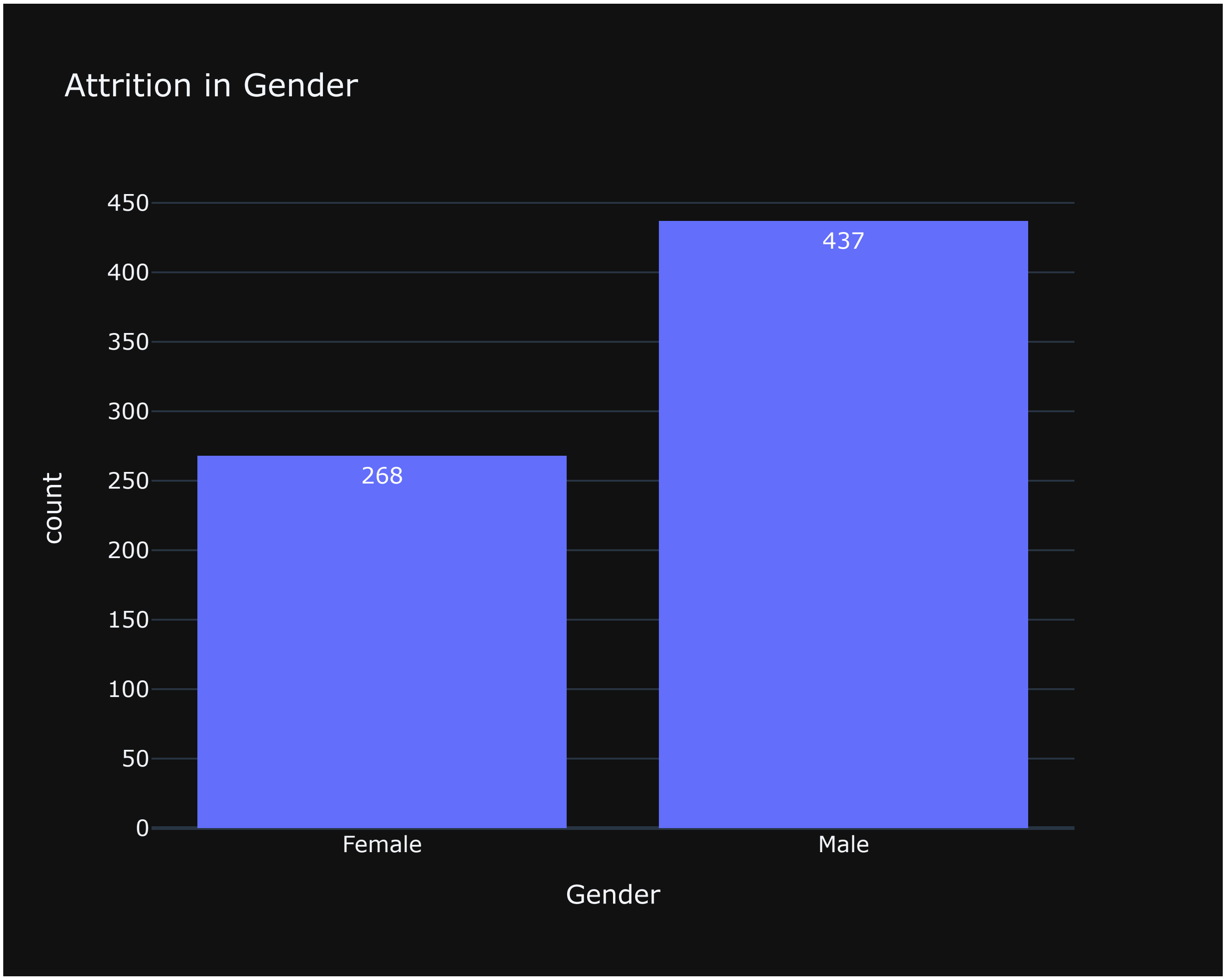


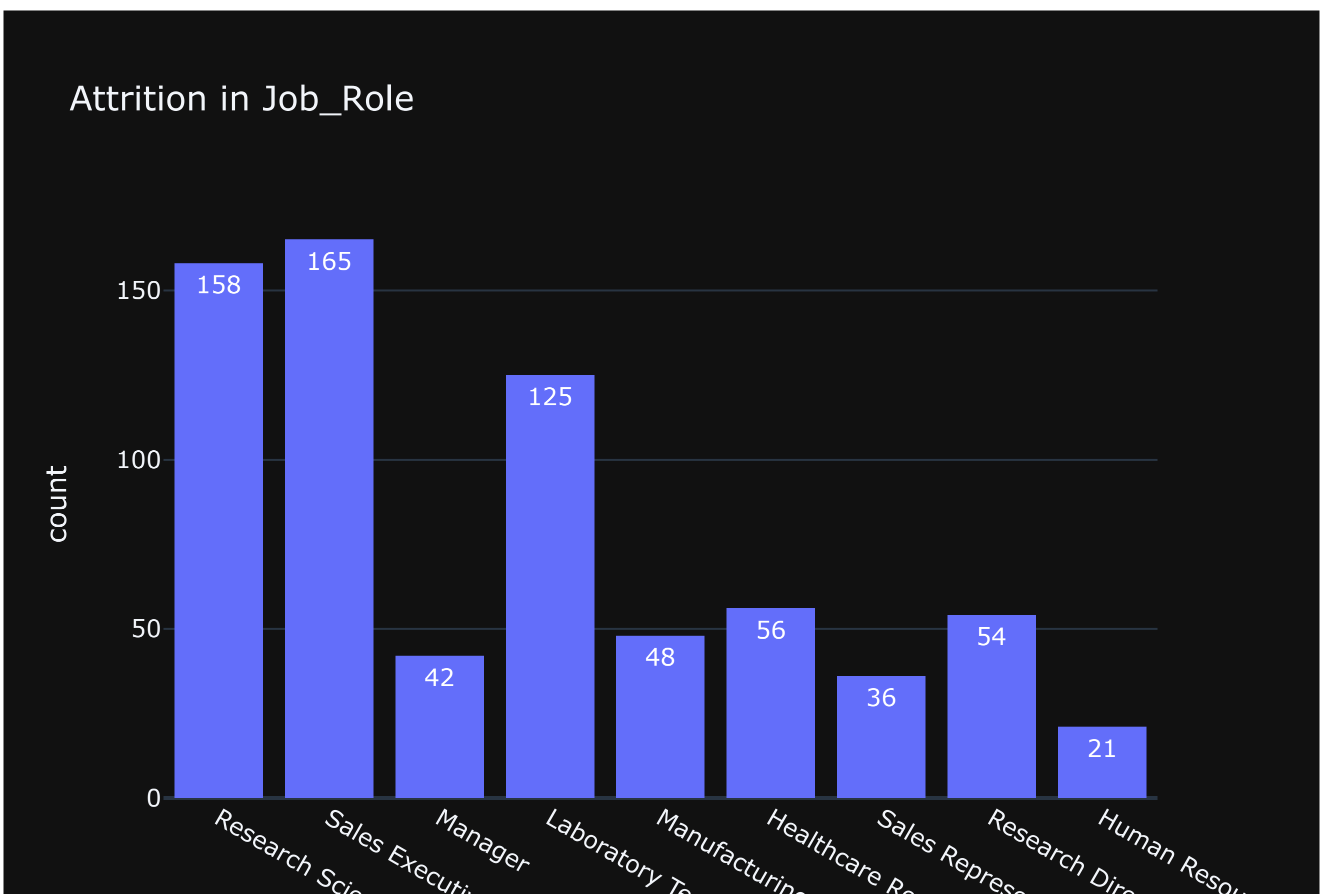

Business_Travel

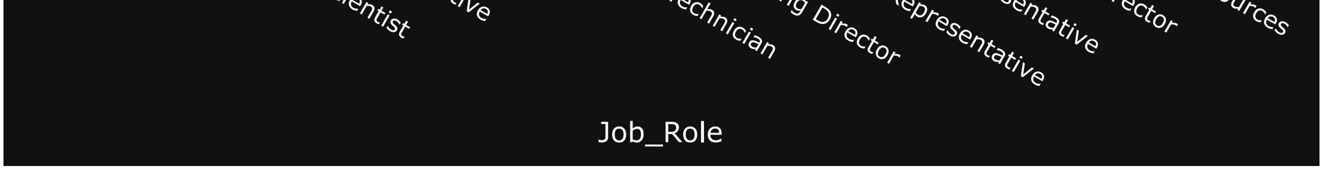

Attrition in Department
count
0
50
100
150
200
250
300
350
400
450
449
55
201
Research & Development
Human Resources
Sales
Department

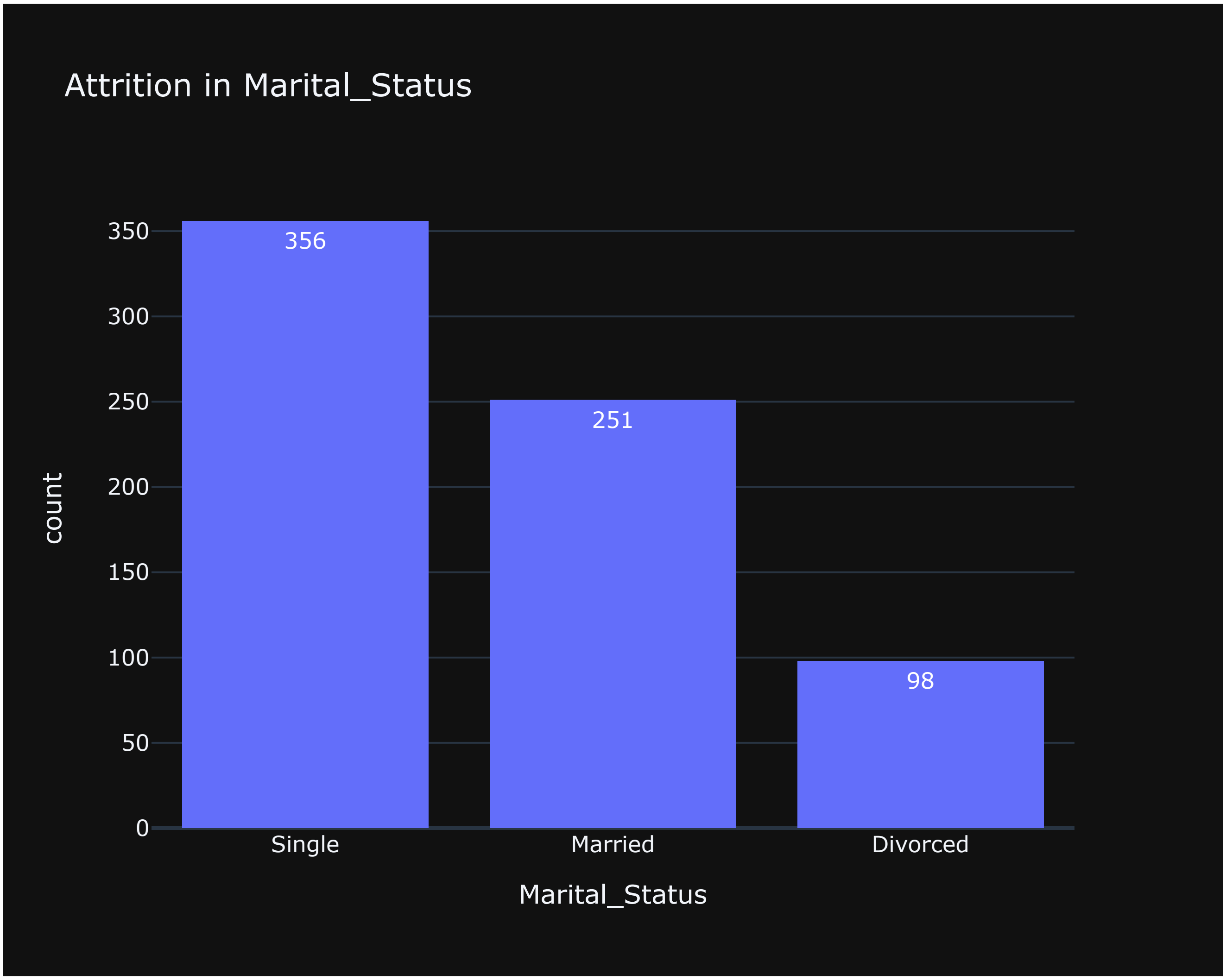

Attrition in Education_Field
count
0
50
100
150
200
250
300
301
222
32
75
45
30
Other


Resources
Education_Field


Attrition in Gender
count
450
400
350
300
250
200
150
100
50
0
268
437
Female
Male
Gender


Attrition in Job_Role
count
150
100
50
0
158
165
42
125
48
56
36
54
21
Manager





EDA Conclusions

Categorical Variables

1. When analyzing categorical variables, we can see that most employees who have left worked for the Research & Development department, with most of them being laboratory technicians, sales executives or research scientists.

2. Most of them had a Bachelor's degree and their education field was mostly either Life Sciences, Medical and Marketing.

3. How can we make the work environment better? What kind of changes must be done, especially for the research and development personel? These are important questions to be asked.

Numerical Variables

1. Looking at the attrition per age histogram, It's noticeable that as employees grow old, the less they tend to leave, and most of our employee attrition is made of employees ranging from 25 to 35 years old. The data also shows that the more working years, more years at the company, and more years in current role employees accumulate, the less likely they are to leave.

2. Those who've less percent salary hike tend to leave more than those with a higher percentual hike in salary.

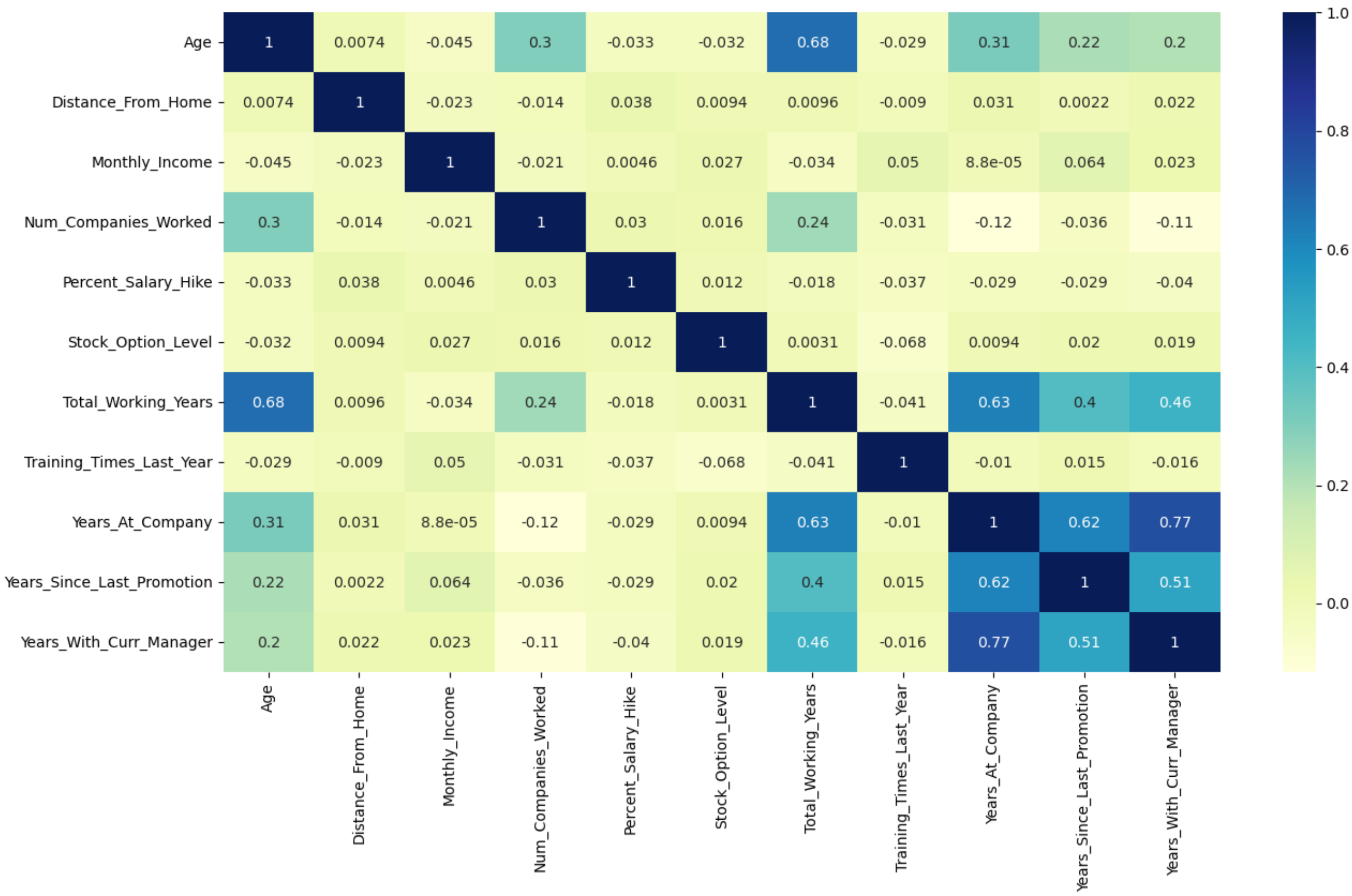
3. So mostly, employees who leave tend to be young, with less time working in the company and at the beginning of their career in general, since most of these employees were working for less than 10 years in total.

4. It's also curious to see that a lot of these employees had less than 2 years working with their last manager. Could we be having issues with some managers? How well are they trained to deal with their teams and the people they led? Are we, as a company doing the best we can to assure a good relationship between managers and teams?

MultiVariate Analysis

In [29]:
```
ax = plt.subplots(figsize=(15, 8))
sns.heatmap(data= df.corr(),annot=True,cmap="YlGnBu")
```

Out[29]:
```
<AxesSubplot:>
```



As shown above, "YearsWithCurrentManager", "YearsSinceLastPromotion" and "YearsAtCompany" are positively correlated to Attrition;

while "Total Working Years", "Job Level", and "PercentSalaryHike" are negatively correlated to Attrition.

Section 04: KPI/ Metric based questions

In [30]:
```
#creating 10 different buckets for Monthly_Income (Deciles)
df['salary_bins'] = pd.qcut(df['Monthly_Income'],14)
```

```
df.head()
```

Out[30]:

	Age	Attrition	Business_Travel	Department	Distance_From_Home	Education	Education_Field	Employee_ID
0	51	No	Travel_Rarely	Sales	6	2	Life Sciences	1
1	31	Yes	Travel_Frequently	Research & Development	10	1	Life Sciences	2
2	32	No	Travel_Frequently	Research & Development	17	4	Other	3
3	38	No	Non-Travel	Research & Development	2	5	Life Sciences	4
4	32	No	Travel_Rarely	Research & Development	10	1	Medical	5

5 rows × 22 columns

In [31]:

```
#create empty data frame
KPI = pd.DataFrame()
print(KPI)
```

```
Empty DataFrame
Columns: []
Index: []
```

In [32]:

```
# calculating Average of Monthly_Income for every bin
KPI['Average'] = df.groupby("salary_bins").Monthly_Income.mean().round()
KPI.head(14)
```

Out[32]:

	Average
salary_bins	
(10089.999, 21940.0]	18828.0
(21940.0, 24390.0]	23297.0
(24390.0, 27430.0]	26058.0
(27430.0, 31720.0]	29243.0
(31720.0, 39040.0]	35265.0
(39040.0, 44021.429]	41643.0
(44021.429, 49190.0]	46589.0
(49190.0, 54670.0]	52117.0
(54670.0, 62727.143]	58396.0
(62727.143, 73858.571]	66917.0
(73858.571, 95820.0]	83667.0
(95820.0, 114294.286]	103484.0
(114294.286, 167990.0]	140179.0
(167990.0, 199990.0]	185642.0

In [33]:

```
# calculating minimum value of Monthly_Income for every bin
KPI['Minimum_Value'] = df.groupby("salary_bins").Monthly_Income.min()
KPI.head(14)
```

Out[33]:

salary_bins	Average	Minimum_Value
(10089.999, 21940.0]	18828.0	10090
(21940.0, 24390.0]	23297.0	22010
(24390.0, 27430.0]	26058.0	24400
(27430.0, 31720.0]	29243.0	27560
(31720.0, 39040.0]	35265.0	31800
(39040.0, 44021.429]	41643.0	39070
(44021.429, 49190.0]	46589.0	44030
(49190.0, 54670.0]	52117.0	49300
(54670.0, 62727.143]	58396.0	54680
(62727.143, 73858.571]	66917.0	62740
(73858.571, 95820.0]	83667.0	74030
(95820.0, 114294.286]	103484.0	96020
(114294.286, 167990.0]	140179.0	115100
(167990.0, 199990.0]	185642.0	168230

In [34]:

```
# calculating maximum value of Monthly_Income for every bin
KPI['Maximum_Value'] = df.groupby("salary_bins").Monthly_Income.max()
KPI.head(14)
```

Out[34]:

salary_bins	Average	Minimum_Value	Maximum_Value
(10089.999, 21940.0]	18828.0	10090	21940
(21940.0, 24390.0]	23297.0	22010	24390
(24390.0, 27430.0]	26058.0	24400	27430
(27430.0, 31720.0]	29243.0	27560	31720
(31720.0, 39040.0]	35265.0	31800	39040
(39040.0, 44021.429]	41643.0	39070	44010
(44021.429, 49190.0]	46589.0	44030	49080
(49190.0, 54670.0]	52117.0	49300	54670
(54670.0, 62727.143]	58396.0	54680	62720
(62727.143, 73858.571]	66917.0	62740	73790
(73858.571, 95820.0]	83667.0	74030	95820
(95820.0, 114294.286]	103484.0	96020	114160
(114294.286, 167990.0]	140179.0	115100	167990
(167990.0, 199990.0]	185642.0	168230	199990

In [35]:

```
# calculating no of employees for every bin
KPI['No_Of_Emp'] = df.groupby("salary_bins").size()
```

```
KPI.head(14)
```

Out[35]:

salary_bins	Average	Minimum_Value	Maximum_Value	No_Of_Emp
(10089.999, 21940.0]	18828.0	10090	21940	315
(21940.0, 24390.0]	23297.0	22010	24390	312
(24390.0, 27430.0]	26058.0	24400	27430	313
(27430.0, 31720.0]	29243.0	27560	31720	313
(31720.0, 39040.0]	35265.0	31800	39040	317
(39040.0, 44021.429]	41643.0	39070	44010	308
(44021.429, 49190.0]	46589.0	44030	49080	313
(49190.0, 54670.0]	52117.0	49300	54670	314
(54670.0, 62727.143]	58396.0	54680	62720	312
(62727.143, 73858.571]	66917.0	62740	73790	313
(73858.571, 95820.0]	83667.0	74030	95820	315
(95820.0, 114294.286]	103484.0	96020	114160	311
(114294.286, 167990.0]	140179.0	115100	167990	314
(167990.0, 199990.0]	185642.0	168230	199990	312

In [36]:

```
# calculating no of employees attrition for every bin
KPI['No_of_people_Attrition'] = df.loc[df['Attrition'] == 'Yes'].groupby("salary_bins").
KPI.head(14)
```

Out[36]:

salary_bins	Average	Minimum_Value	Maximum_Value	No_Of_Emp	No_of_people_Attrition
(10089.999, 21940.0]	18828.0	10090	21940	315	57
(21940.0, 24390.0]	23297.0	22010	24390	312	41
(24390.0, 27430.0]	26058.0	24400	27430	313	65
(27430.0, 31720.0]	29243.0	27560	31720	313	47
(31720.0, 39040.0]	35265.0	31800	39040	317	51
(39040.0, 44021.429]	41643.0	39070	44010	308	48
(44021.429, 49190.0]	46589.0	44030	49080	313	44
(49190.0, 54670.0]	52117.0	49300	54670	314	48
(54670.0, 62727.143]	58396.0	54680	62720	312	77
(62727.143, 73858.571]	66917.0	62740	73790	313	54
(73858.571, 95820.0]	83667.0	74030	95820	315	45
(95820.0, 114294.286]	103484.0	96020	114160	311	47
(114294.286, 167990.0]	140179.0	115100	167990	314	39
(167990.0, 199990.0]	185642.0	168230	199990	312	42

In [37]:
```
# calculating percent of employees were attrition for every bin
KPI['Attrition rate'] = KPI['No_of_people_Attrition']/KPI['No_Of_Emp'] *100
KPI.head(14)
```

Out[37]:

salary_bins	Average	Minimum_Value	Maximum_Value	No_Of_Emp	No_of_people_Attrition	Attrition rate
(10089.999, 21940.0]	18828.0	10090	21940	315	57	18.095238
(21940.0, 24390.0]	23297.0	22010	24390	312	41	13.141026
(24390.0, 27430.0]	26058.0	24400	27430	313	65	20.766773
(27430.0, 31720.0]	29243.0	27560	31720	313	47	15.015974
(31720.0, 39040.0]	35265.0	31800	39040	317	51	16.088328
(39040.0, 44021.429]	41643.0	39070	44010	308	48	15.584416
(44021.429, 49190.0]	46589.0	44030	49080	313	44	14.057508
(49190.0, 54670.0]	52117.0	49300	54670	314	48	15.286624
(54670.0, 62727.143]	58396.0	54680	62720	312	77	24.679487
(62727.143, 73858.571]	66917.0	62740	73790	313	54	17.252396
(73858.571, 95820.0]	83667.0	74030	95820	315	45	14.285714
(95820.0, 114294.286]	103484.0	96020	114160	311	47	15.112540
(114294.286, 167990.0]	140179.0	115100	167990	314	39	12.420382
(167990.0, 199990.0]	185642.0	168230	199990	312	42	13.461538

In [38]:
```
# rounding attrition rate by 2 decimals
KPI['Attrition rate'] = KPI['Attrition rate'].round(2)
KPI.head(14)
```

Out[38]:

salary_bins	Average	Minimum_Value	Maximum_Value	No_Of_Emp	No_of_people_Attrition	Attrition rate
(10089.999, 21940.0]	18828.0	10090	21940	315	57	18.10
(21940.0, 24390.0]	23297.0	22010	24390	312	41	13.14
(24390.0, 27430.0]	26058.0	24400	27430	313	65	20.77
(27430.0, 31720.0]	29243.0	27560	31720	313	47	15.02
(31720.0, 39040.0]	35265.0	31800	39040	317	51	16.09
(39040.0, 44021.429]	41643.0	39070	44010	308	48	15.58
(44021.429,	46589.0	44030	49080	313	44	14.06

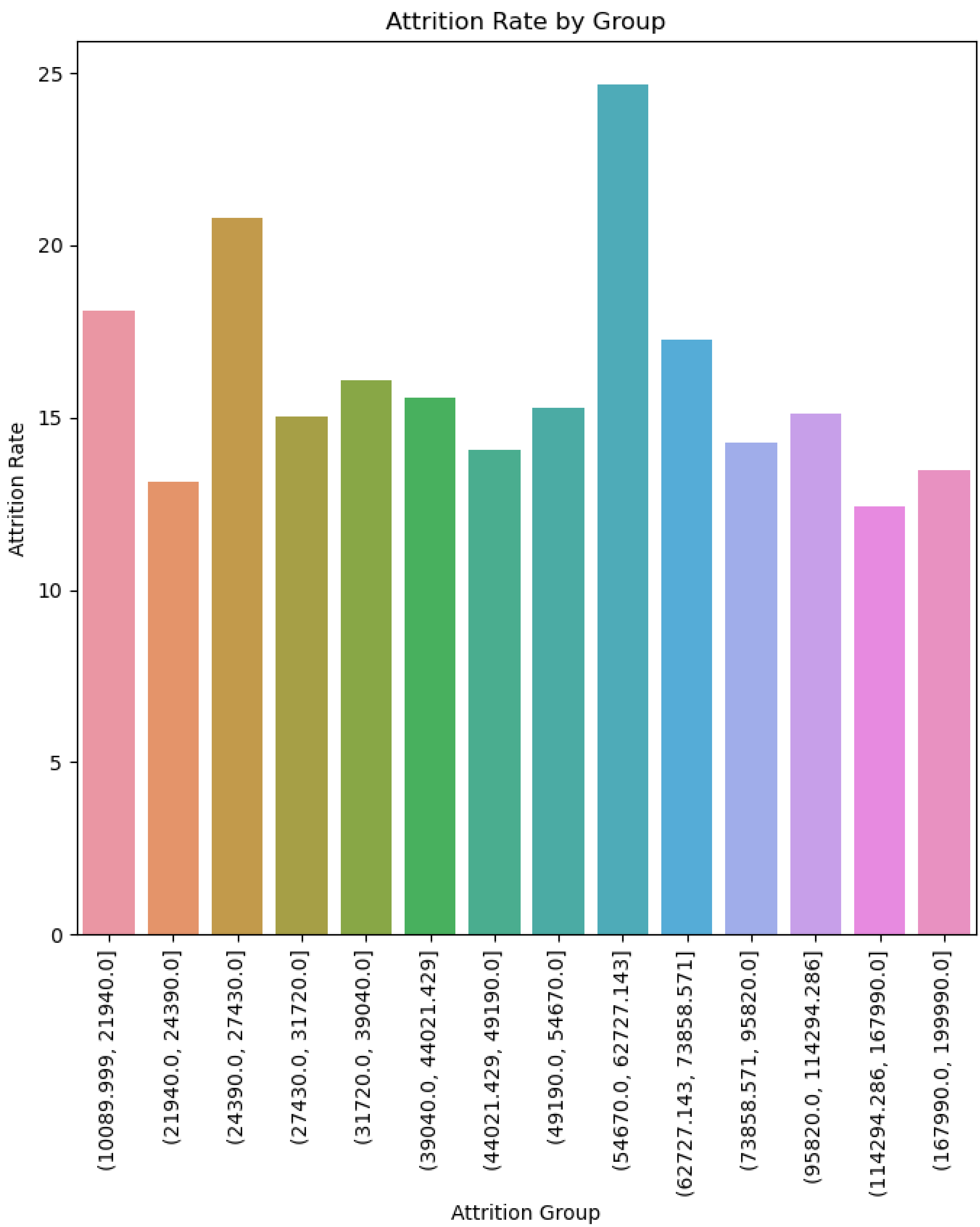
49190.0]						
(49190.0, 54670.0]	52117.0	49300	54670	314	48	15.29
(54670.0, 62727.143]	58396.0	54680	62720	312	77	24.68
(62727.143, 73858.571]	66917.0	62740	73790	313	54	17.25
(73858.571, 95820.0]	83667.0	74030	95820	315	45	14.29
(95820.0, 114294.286]	103484.0	96020	114160	311	47	15.11
(114294.286, 167990.0]	140179.0	115100	167990	314	39	12.42
(167990.0, 199990.0]	185642.0	168230	199990	312	42	13.46

In [39]:
```
KPI = KPI.reset_index()
KPI.head(14)
```

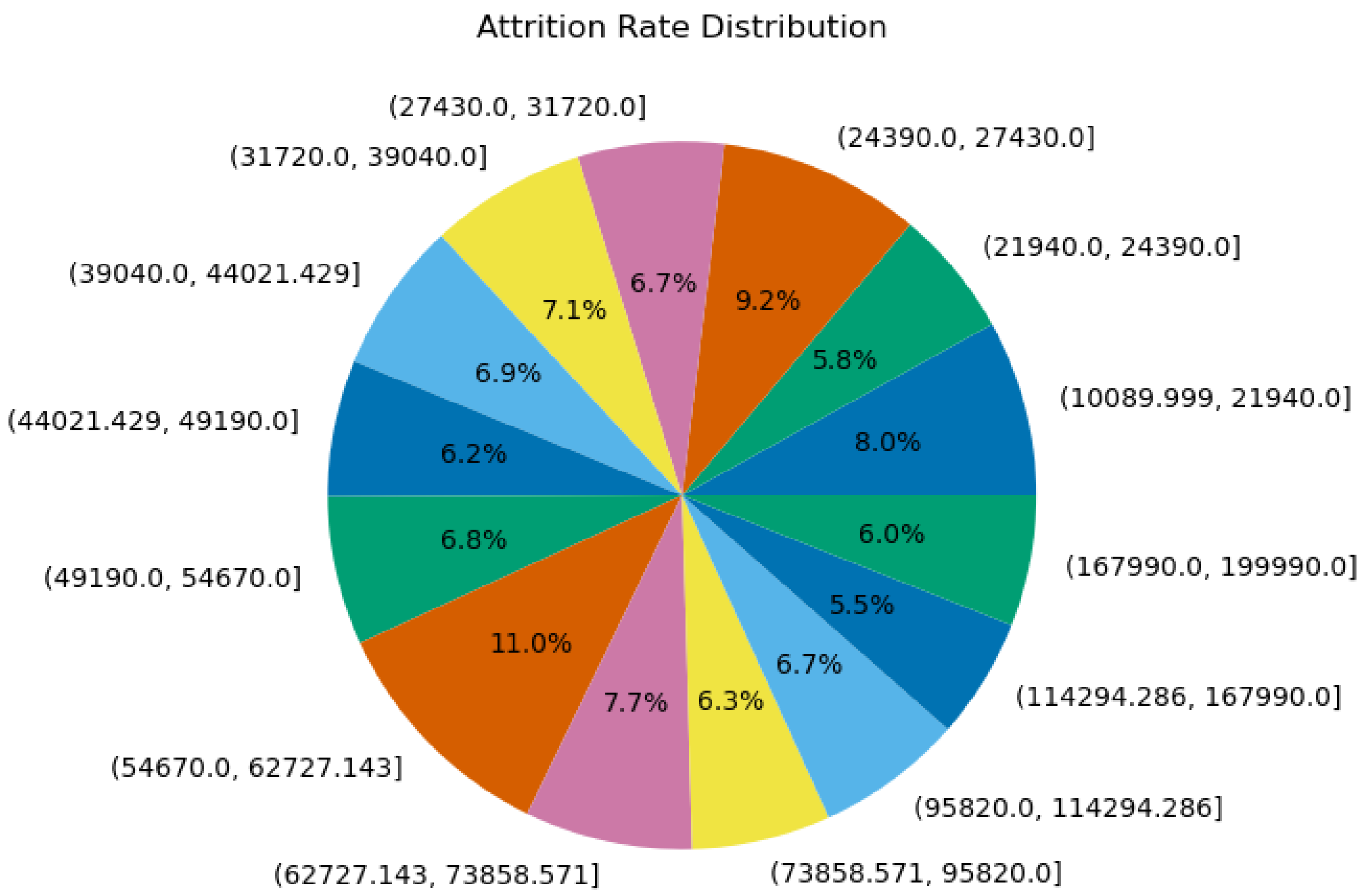
Out[39]:

	salary_bins	Average	Minimum_Value	Maximum_Value	No_Of_Emp	No_of_people_Attrition	Attrition rate
0	(10089.999, 21940.0]	18828.0	10090	21940	315	57	18.10
1	(21940.0, 24390.0]	23297.0	22010	24390	312	41	13.14
2	(24390.0, 27430.0]	26058.0	24400	27430	313	65	20.77
3	(27430.0, 31720.0]	29243.0	27560	31720	313	47	15.02
4	(31720.0, 39040.0]	35265.0	31800	39040	317	51	16.09
5	(39040.0, 44021.429]	41643.0	39070	44010	308	48	15.58
6	(44021.429, 49190.0]	46589.0	44030	49080	313	44	14.06
7	(49190.0, 54670.0]	52117.0	49300	54670	314	48	15.29
8	(54670.0, 62727.143]	58396.0	54680	62720	312	77	24.68
9	(62727.143, 73858.571]	66917.0	62740	73790	313	54	17.25
10	(73858.571, 95820.0]	83667.0	74030	95820	315	45	14.29
11	(95820.0, 114294.286]	103484.0	96020	114160	311	47	15.11
12	(114294.286, 167990.0]	140179.0	115100	167990	314	39	12.42
13	(167990.0, 199990.0]	185642.0	168230	199990	312	42	13.46

In [40]:
```
plt.figure(figsize=(8, 8))
ax = sns.barplot(x='salary_bins', y='Attrition rate', data=KPI)
plt.title('Attrition Rate by Group')
plt.xlabel('Attrition Group')
plt.ylabel('Attrition Rate')
plt.xticks(rotation=90)


plt.show()
```



In [41]:
```
# Pie Chart
plt.figure(figsize=(6, 6))
plt.pie(KPI['Attrition rate'], labels=KPI['salary_bins'], autopct='%1.1f%%')
plt.title('Attrition Rate Distribution')
plt.show()
```



A few observations can be made based on the Bar chat and pie for numerical features:

-> As we can observe that the for bin (54670.0, 62727.143) attrition rate was more i.e., 11.0 % compared to other bins. follows attrition rate more for the bins (21940.0, 24390.0) = 9.2%, (10089.999, 21940.0] = 8.0%, (62727.143, 73858.571) = 7.7 and as follows

-> As we can observe that we cannot get a conclusion on binning we need further investigation.

In [42]:
```
df.head()
```

Out[42]:

	Age	Attrition	Business_Travel	Department	Distance_From_Home	Education	Education_Field	Employee_ID
0	51	No	Travel_Rarely	Sales	6	2	Life Sciences	1
1	31	Yes	Travel_Frequently	Research & Development	10	1	Life Sciences	2
2	32	No	Travel_Frequently	Research & Development	17	4	Other	3
3	38	No	Non-Travel	Research & Development	2	5	Life Sciences	4
4	32	No	Travel_Rarely	Research & Development	10	1	Medical	5

5 rows × 22 columns

analyzing Attrition rate for a combination of Two variables

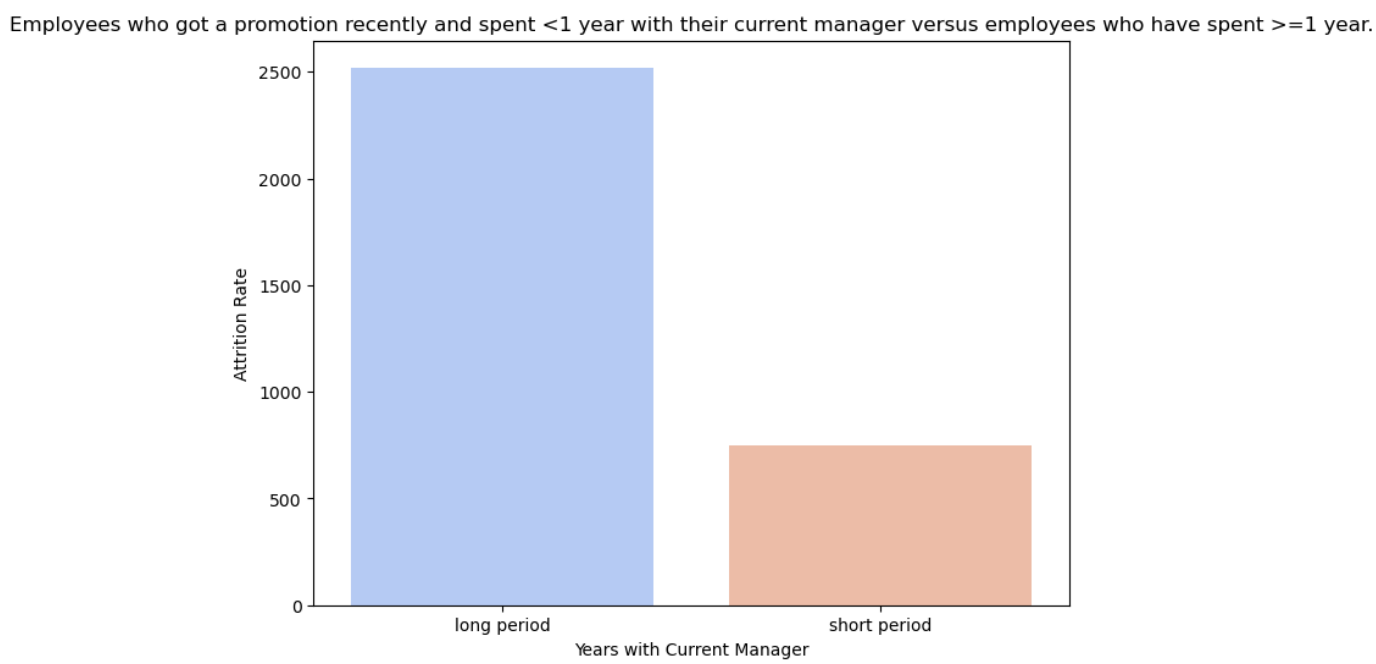
In [43]:

```
## Encoding the Attrition Column for Modelling
df['Attrition'] = [1 if i == 'Yes' else 0 for i in df['Attrition']]
```

Employees who got a promotion recently and spent <1 year with their current manager versus employees who have spent >=1 year.

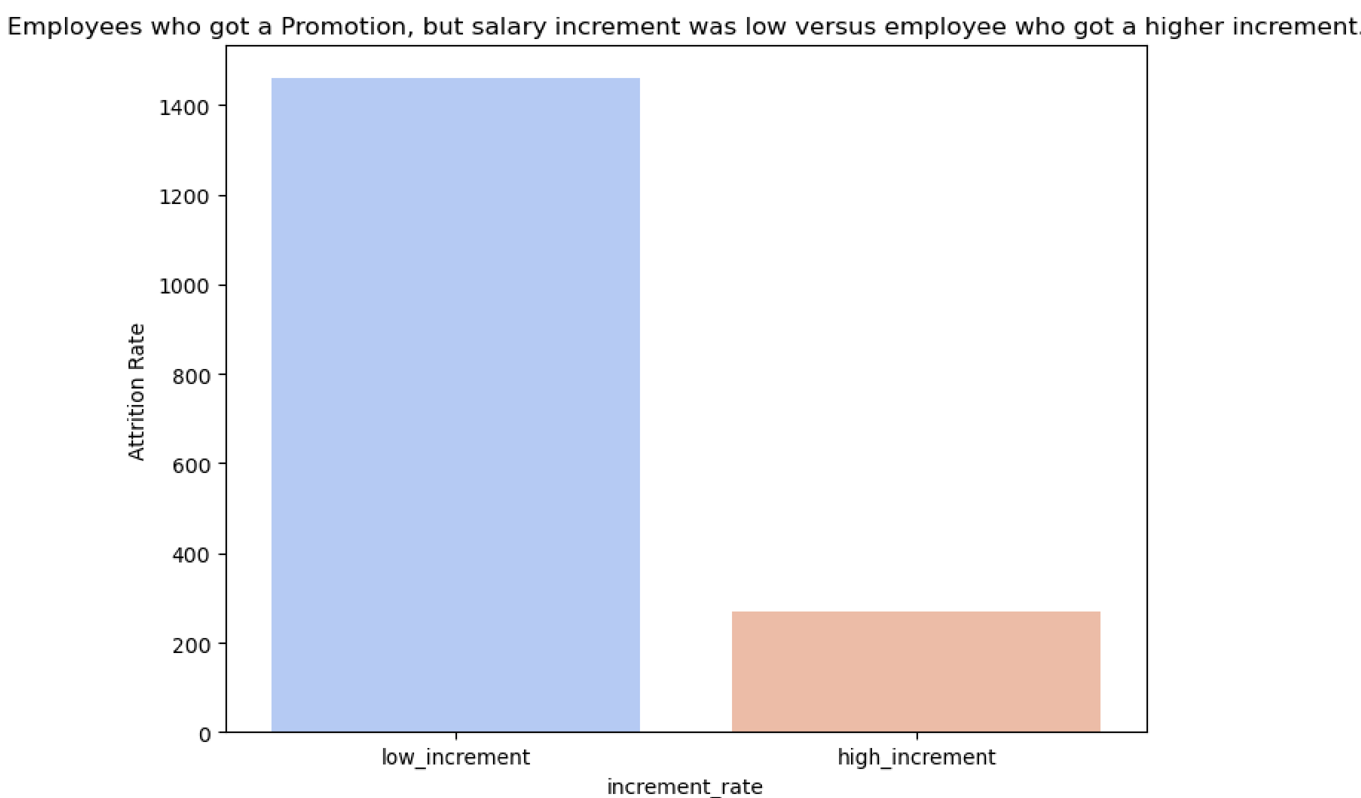
In [44]:

```
# creating column Emp_promot_rectly,Years_With_Curr_Manager_cat as per the requirement

df['Emp_promot_rectly'] = df['Years_Since_Last_Promotion'].apply(lambda x: 'prom _recent

df['Years_With_Curr_Manager_cat'] = df['Years_With_Curr_Manager'].apply(lambda x: 'short

# Filter the data to focus on employees who got a promotion recently
filtered_data = df[df['Emp_promot_rectly'] == 'prom _recently']

# Group by Manager_Years_Category and calculate the count of employees in each group
promotion_counts =filtered_data['Years_With_Curr_Manager_cat'].value_counts().reset_inde

# Rename the columns for the plot
promotion_counts.columns = ['Manager_Years_Category', 'Count']

# Plot the bar chart
plt.figure(figsize=(8, 6))
sns.barplot(x='Manager_Years_Category', y='Count', data=promotion_counts, palette='coolw

# Customize the plot
plt.title("Employees who got a promotion recently and spent <1 year with their current m
plt.xlabel("Years with Current Manager")
plt.ylabel("Attrition Rate")


# Show the plot
plt.show()
```



Employees who got a Promotion, but salary increment was low versus employee who got a higher increment.

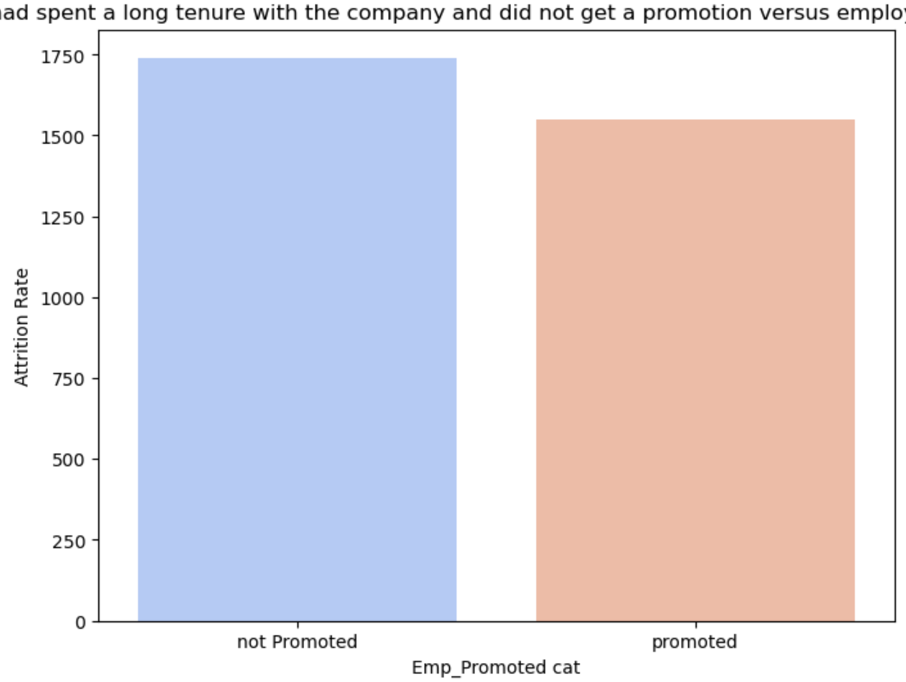
In [45]:
```
# creating column Emp_Promoted,salary_increment as per the requirement

df['Emp_Promoted'] = df['Years_Since_Last_Promotion'].apply(lambda x: 'promoted' if x <

df['salary_increment'] = df['Percent_Salary_Hike'].apply(lambda x: 'low_increment' if x


# Filter the data to focus on employees who got a promoted
filtered_data_1 = df[df['Emp_Promoted'] == 'promoted']


# Group by salary_increment and calculate the count of employees in each group
promotion_counts_1 =filtered_data_1['salary_increment'].value_counts().reset_index()

# Rename the columns for the plot
promotion_counts_1.columns = ['salary increment cat', 'Count']

# Plot the bar chart
plt.figure(figsize=(8, 6))
sns.barplot(x='salary increment cat', y='Count', data=promotion_counts_1, palette='coolw


# Customize the plot
plt.title("Employees who got a Promotion, but salary increment was low versus employee w
plt.xlabel("increment_rate")
plt.ylabel("Attrition Rate")
# Show the plot
plt.show()
```



Employees who had spent a long tenure with the company and did not get a promotion versus employee who got a promotion.

In [46]:
```
# creating column pent a long tenure as per the requirement

df['spent a long tenure'] = df['Years_At_Company'].apply(lambda x: 'spent_long' if x < 1

# Filter the data to focus on employees who got a spent a long tenure
filtered_data_2 = df[df['spent a long tenure'] == 'spent_long']


# Group by Emp_Promoted and calculate the count of employees in each group
promotion_counts_2 = filtered_data_2['Emp_Promoted'].value_counts().reset_index()

# Rename the columns for the plot
promotion_counts_2.columns = ['Emp_Promoted cat', 'Count']

# Plot the bar chart
plt.figure(figsize=(8, 6))
sns.barplot(x='Emp_Promoted cat', y='Count', data=promotion_counts_2, palette='coolwarm'


# Customize the plot
plt.title("Employees who had spent a long tenure with the company and did not get a prom
plt.xlabel("Emp_Promoted cat")
plt.ylabel("Attrition Rate")
# Show the plot
plt.show()
```

Employees who had spent a long tenure with the company and did not get a promotion versus employee who got a promotion.



Employees who had spent a long tenure with the company and got a low salary increment versus employee who got a decent hike.

In [47]:
```
# Group by salary_increment and calculate the count of employees in each group
promotion_counts_3 = filtered_data_2['salary_increment'].value_counts().reset_index()

# Rename the columns for the plot
promotion_counts_3.columns = ['salary_increment', 'Count']


# Plot the bar chart
plt.figure(figsize=(8, 6))
sns.barplot(x='salary_increment', y='Count', data=promotion_counts_3, palette='coolwarm'
```

```
# Customize the plot
plt.title("Employees who had spent a long tenure with the company and got a low salary i
plt.xlabel("salary_increment")
plt.ylabel("Attrition Rate")
# Show the plot
plt.show()
```

Employees who had spent a long tenure with the company and got a low salary increment versus employee who got a decent hike.

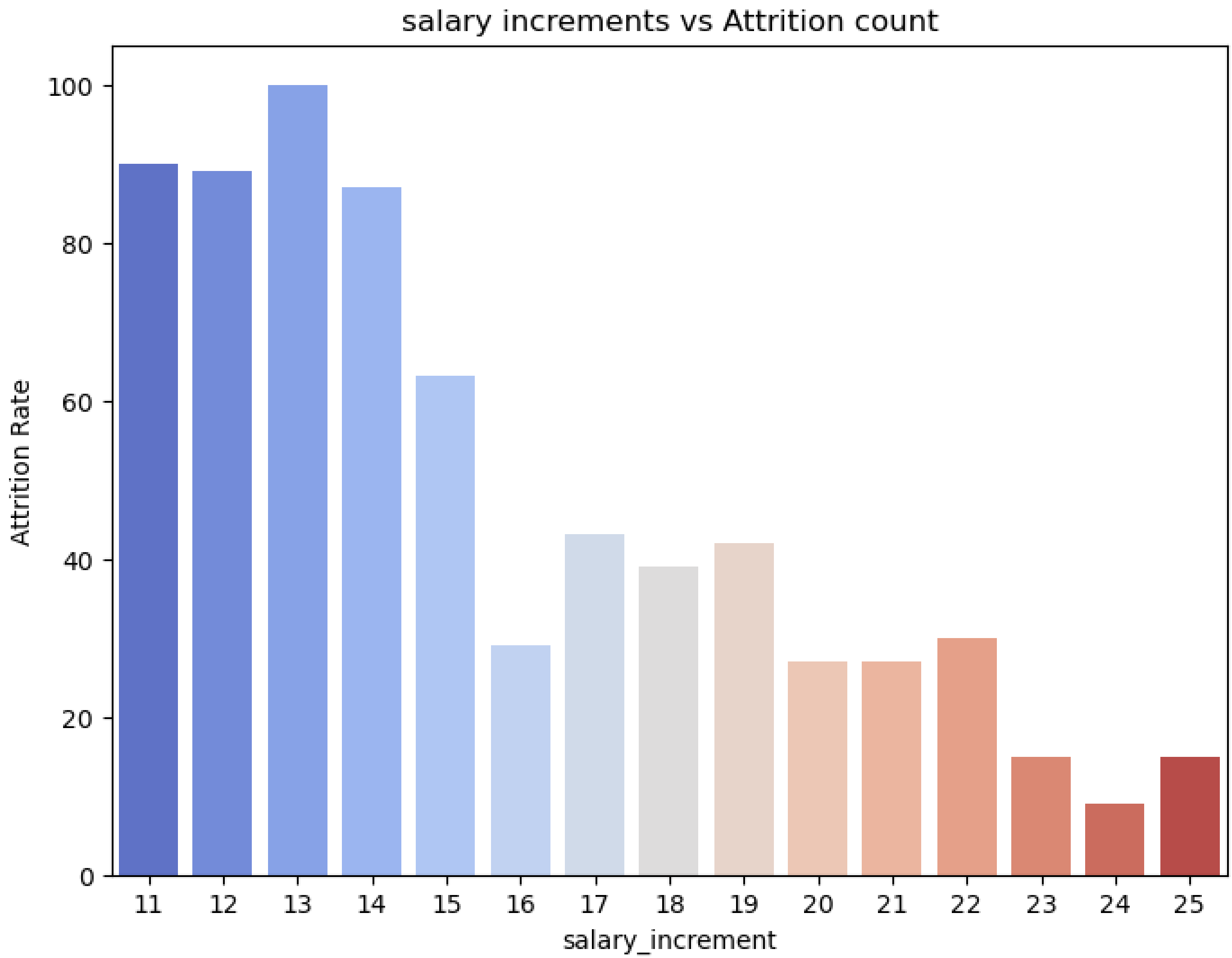
Section 05: Open-ended questions and recommendations

Which of the below are strong drivers of Attrition:

1) Low salary increments

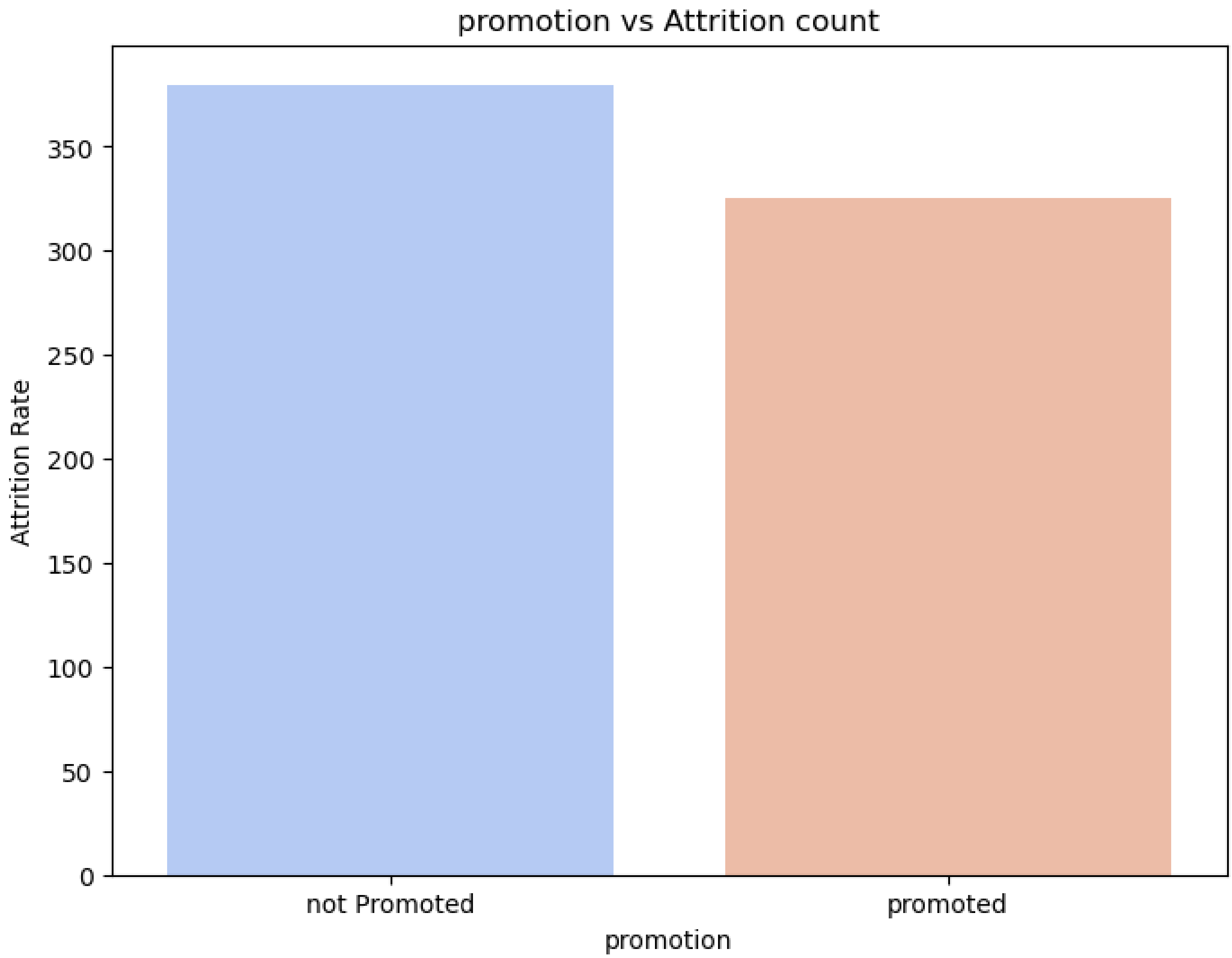
In [48]:

```
# Filter the data to focus on employees who left the company
filtered_data_5 = df[df['Attrition'] == 1]


# Group by Percent_Salary_Hike and calculate the count of employees who where left compa
promotion_counts_5 = filtered_data_5['Percent_Salary_Hike'].value_counts().reset_index()

# Rename the columns for the plot
promotion_counts_5.columns = ['Percent_Salary_Hike', 'Count']

# Plot the bar chart
plt.figure(figsize=(8, 6))
sns.barplot(x='Percent_Salary_Hike', y='Count', data=promotion_counts_5, palette='coolwa


# Customize the plot
plt.title("salary increments vs Attrition count")
plt.xlabel("salary_increment")
plt.ylabel("Attrition Rate")
# Show the plot
plt.show()
```



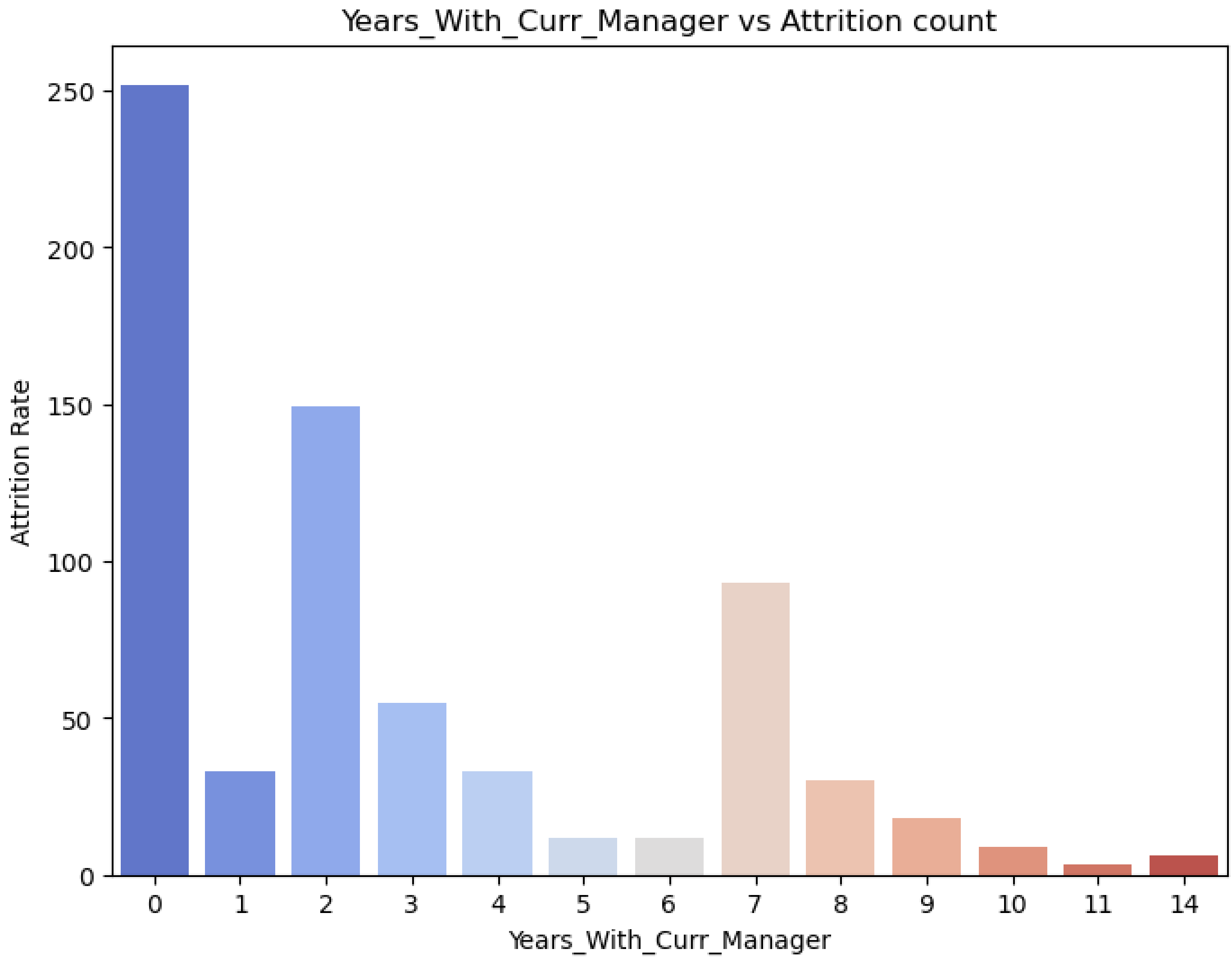
2) No promotion

In [49]:
```
# Group by Emp_Promoted and calculate the count of employees who where left company in e
promotion_counts_6 = filtered_data_5['Emp_Promoted'].value_counts().reset_index()

# Rename the columns for the plot
promotion_counts_6.columns = ['Emp_Promoted', 'Count']

# Plot the bar chart
plt.figure(figsize=(8, 6))
sns.barplot(x='Emp_Promoted', y='Count', data=promotion_counts_6, palette='coolwarm')


# Customize the plot
plt.title("promotion vs Attrition count")
plt.xlabel("promotion")
plt.ylabel("Attrition Rate")
# Show the plot
plt.show()
```



3) Current manager

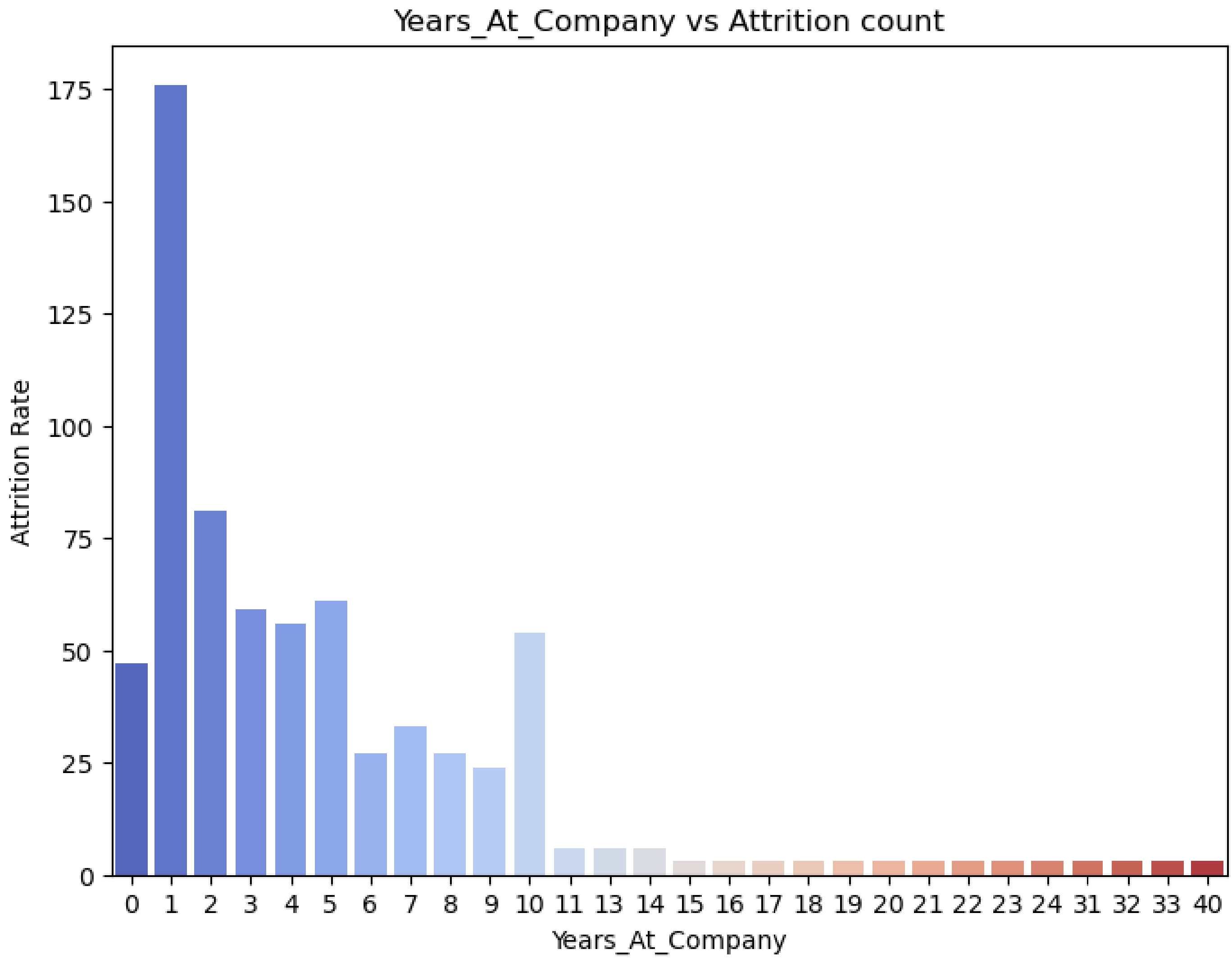
In [50]:
```
# Group by Years_With_Curr_Manager and calculate the count of employees who where left c
promotion_counts_7 = filtered_data_5['Years_With_Curr_Manager'].value_counts().reset_ind

# Rename the columns for the plot
promotion_counts_7.columns = ['Years_With_Curr_Manager', 'Count']

# Plot the bar chart
plt.figure(figsize=(8, 6))
sns.barplot(x='Years_With_Curr_Manager', y='Count', data=promotion_counts_7, palette='co


# Customize the plot
plt.title("Years_With_Curr_Manager vs Attrition count")
plt.xlabel("Years_With_Curr_Manager")
plt.ylabel("Attrition Rate")
# Show the plot
plt.show()
```



4) Need for a change (after having served the company for long)

In [51]:

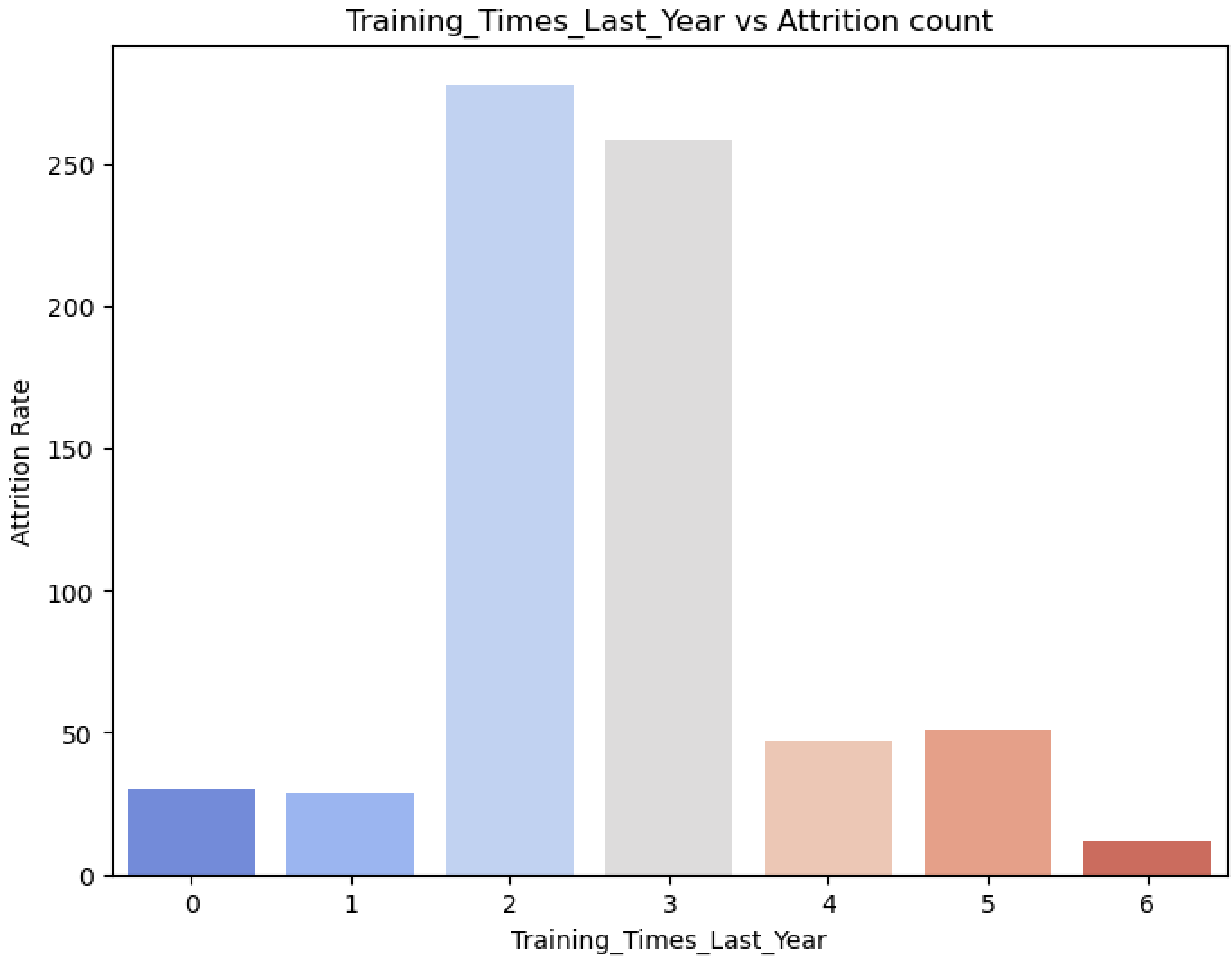
```
# Group by Years_At_Company and calculate the count of employees who where left company
promotion_counts_8 = filtered_data_5['Years_At_Company'].value_counts().reset_index()

# Rename the columns for the plot
promotion_counts_8.columns = ['Years_At_Company', 'Count']

# Plot the bar chart
plt.figure(figsize=(8, 6))
sns.barplot(x='Years_At_Company', y='Count', data=promotion_counts_8, palette='coolwarm'


# Customize the plot
plt.title("Years_At_Company vs Attrition count")
plt.xlabel("Years_At_Company")
plt.ylabel("Attrition Rate")
# Show the plot
plt.show()
```



5) Lesser training time spent in last year

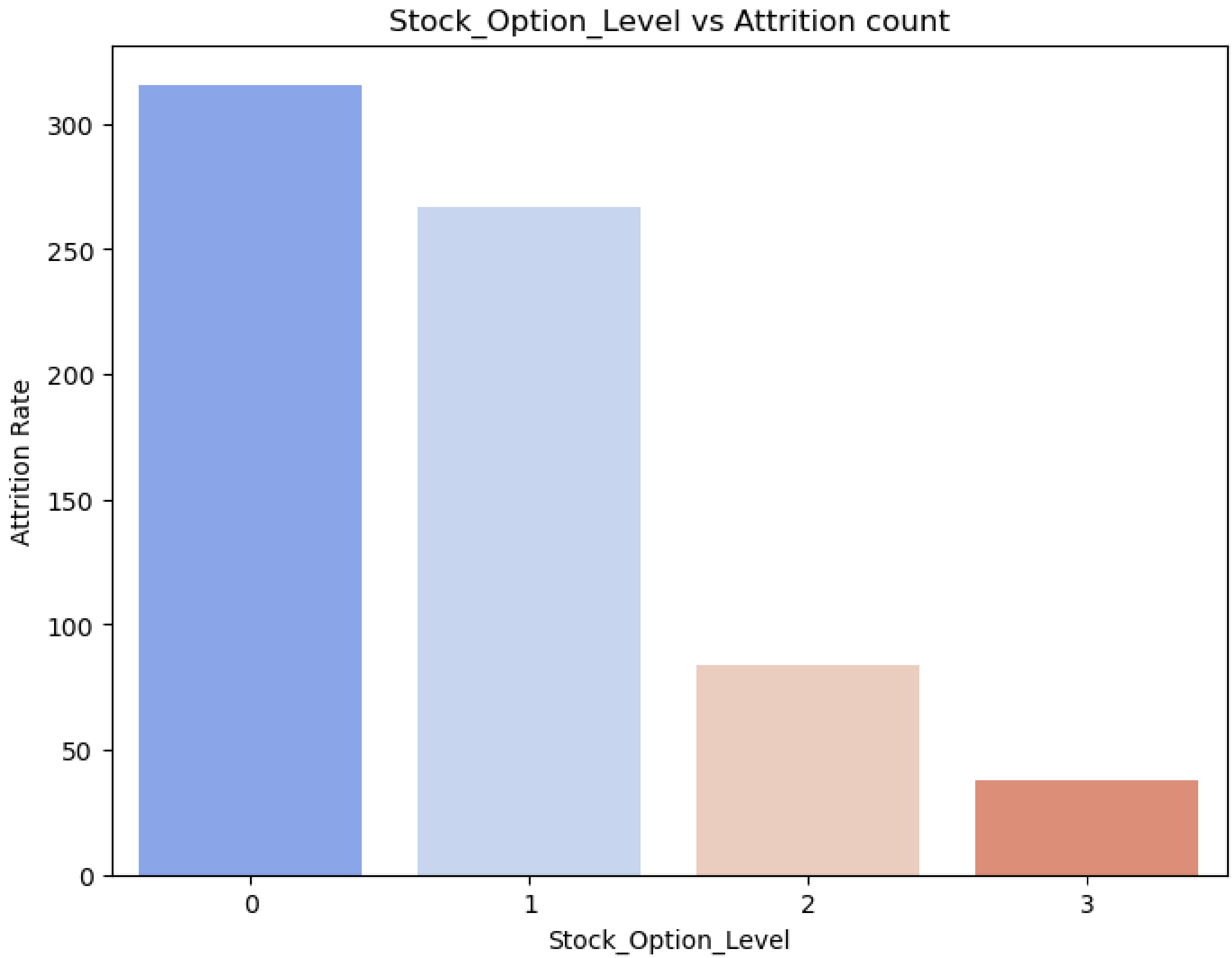
In [52]:

```
# Group by Training_Times_Last_Year and calculate the count of employees who where left
promotion_counts_9 = filtered_data_5['Training_Times_Last_Year'].value_counts().reset_in

# Rename the columns for the plot
promotion_counts_9.columns = ['Training_Times_Last_Year', 'Count']

# Plot the bar chart
plt.figure(figsize=(8, 6))
sns.barplot(x='Training_Times_Last_Year', y='Count', data=promotion_counts_9, palette='c


# Customize the plot
plt.title("Training_Times_Last_Year vs Attrition count")
plt.xlabel("Training_Times_Last_Year")
plt.ylabel("Attrition Rate")
# Show the plot
plt.show()
```



6)Stock Options not being given

In [53]:
```
# Group by Training_Times_Last_Year and calculate the count of employees who where left
promotion_counts_9 = filtered_data_5['Stock_Option_Level'].value_counts().reset_index()

# Rename the columns for the plot
promotion_counts_9.columns = ['Stock_Option_Level', 'Count']

# Plot the bar chart
plt.figure(figsize=(8, 6))
sns.barplot(x='Stock_Option_Level', y='Count', data=promotion_counts_9, palette='coolwar


# Customize the plot
plt.title("Stock_Option_Level vs Attrition count")
plt.xlabel("Stock_Option_Level")
plt.ylabel("Attrition Rate")
# Show the plot
plt.show()
```



Always open for feedback and suggestions.If it helps Thumbs Up !!!