In [1]:

```python
# Import the necessary libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
```

In [2]:

```python
# load dataset
hr = pd.read_csv('C:\\Users\\Sakawat Siyam\\Downloads\\Data P3 MeriSKILL\\HR-Employee-Attrition.csv')
```

In [3]:

```python
hr.head(5)
```

Out[3]:

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

5 rows × 35 columns

In [4]:

```python
hr.describe()
```

Out[4]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	1470.000000
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156	2.729932
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428	0.711561
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000	1.000000
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000	2.000000
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000	3.000000
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000	3.000000
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000	4.000000

8 rows × 26 columns

In [5]:

```python
 #check the duplicate value
 hr.duplicated().sum()
```

Out[5]:

```
0
```

In [6]:

```python
#check the null value
hr.isnull().sum()
```

Out[6]:

```
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 [7]:

```python
# Select columns with data type 'object' from the 'hr' DataFrame
object_columns = hr.select_dtypes(include='object').columns

# Initialize a dictionary to store unique values for each object column
unique_values = {}

# Iterate through the selected object columns
for column in object_columns:
    # Store the unique values of the current column in the dictionary
    unique_values[column] = hr[column].unique()

# Create a Pandas Series to display the unique values
unique_values_series = pd.Series(unique_values)

# Print or return the unique values for object columns
print(unique_values_series)
```

```
Attrition                                                [Yes, No]
BusinessTravel       [Travel_Rarely, Travel_Frequently, Non-Travel]
Department          [Sales, Research & Development, Human Resources]
EducationField     [Life Sciences, Other, Medical, Marketing, Tec...
Gender                                              [Female, Male]
JobRole            [Sales Executive, Research Scientist, Laborato...
MaritalStatus                          [Single, Married, Divorced]
Over18                                                         [Y]
OverTime                                                 [Yes, No]
dtype: object
```

In [8]:

```python
# Delete specific columns from the 'hr' DataFrame if they exist
# 'EmployeeCount', 'Over18', and 'StandardHours' are being removed
hr.drop(['EmployeeCount', 'Over18', 'StandardHours'], axis=1, inplace=True, errors='ignore')
```

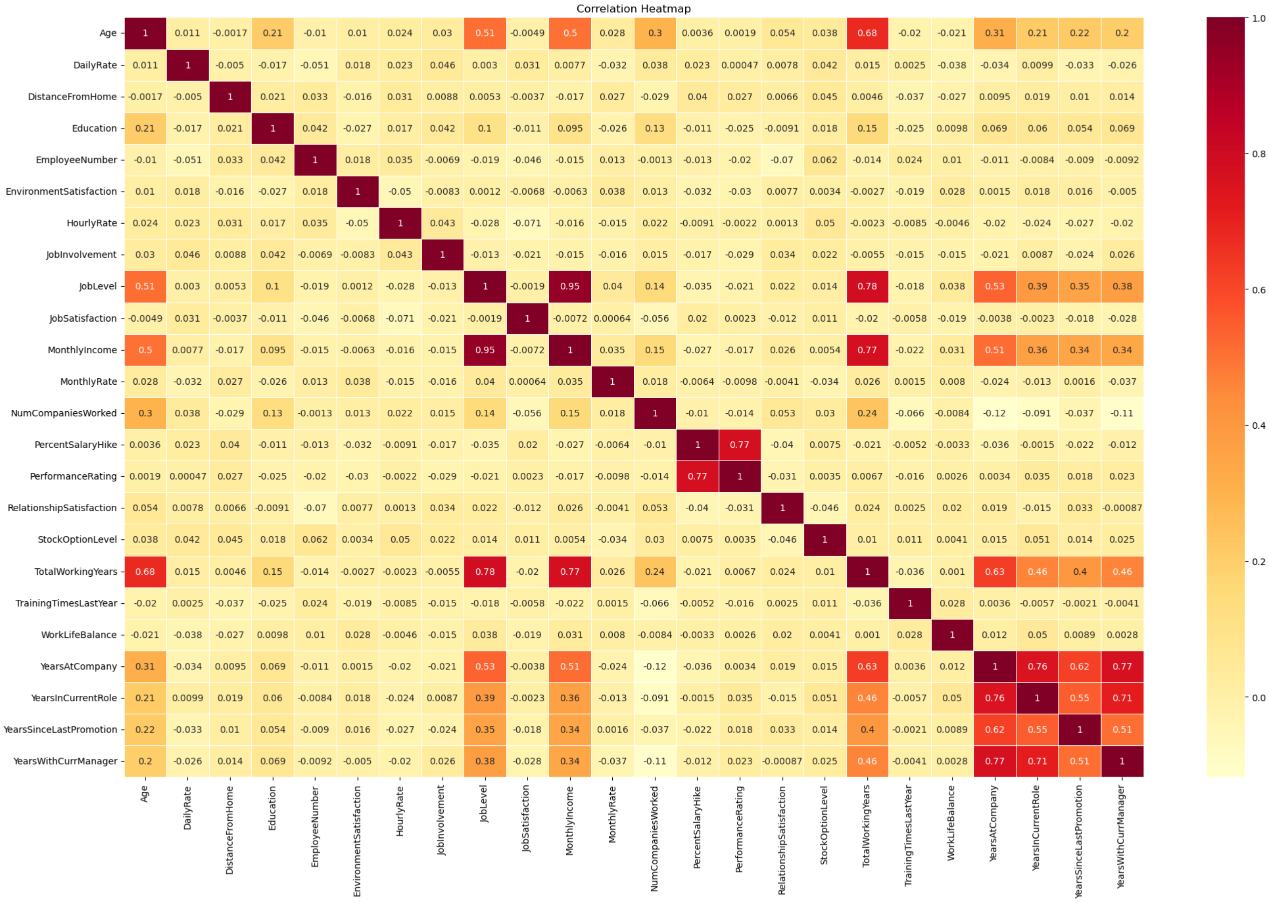
In [9]:

```python
# Compute the correlation matrix of numeric variables in the 'hr' DataFrame
correlation_matrix = hr.corr()
```

```
C:\Users\Sakawat Siyam\AppData\Local\Temp\ipykernel_14952\3214836715.py:2: FutureWarning: The default value of numeric_only in DataFr
ame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  correlation_matrix = hr.corr()
```

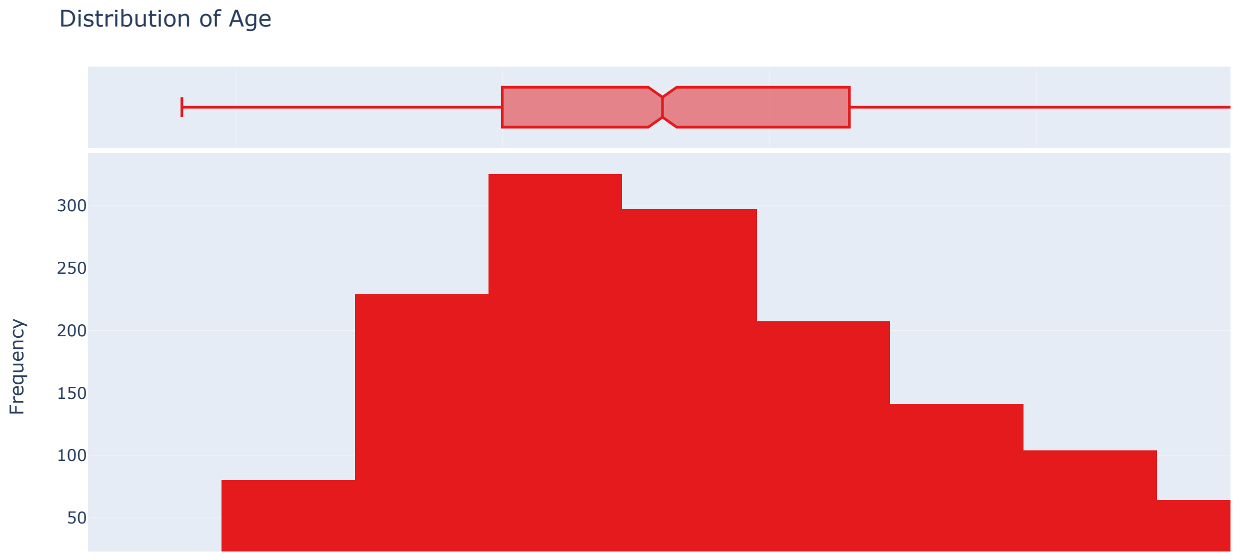
In [10]:

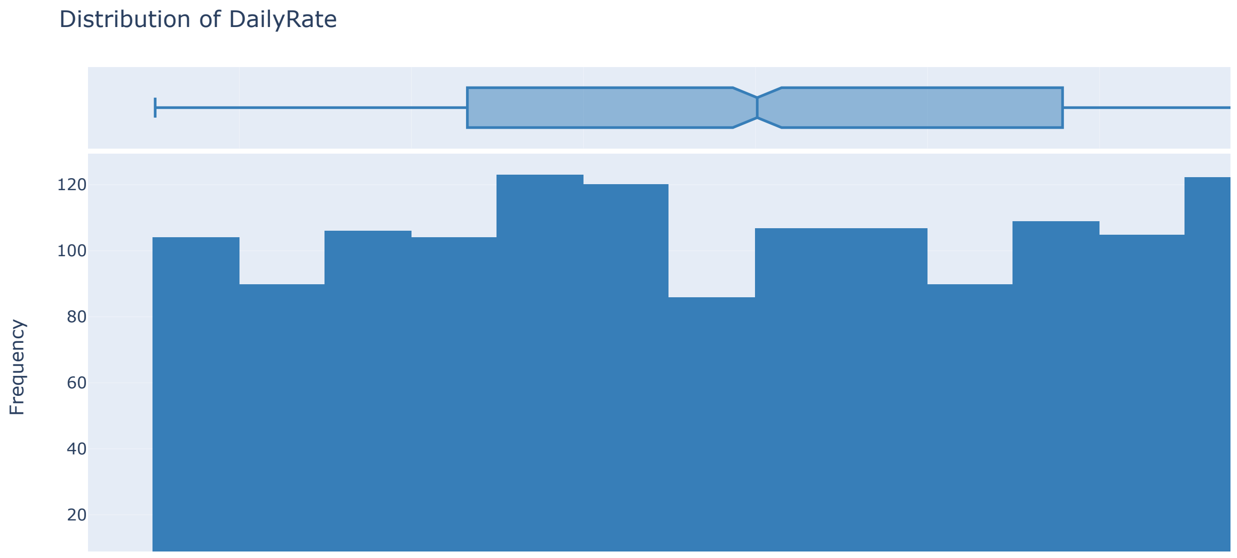
```python
# Select only the numeric columns from the 'hr' DataFrame
numeric_columns = hr.select_dtypes(include='number')

# Compute the correlation matrix
correlation_matrix = numeric_columns.corr()

# Set the size of the heatmap
plt.figure(figsize=(25, 15))

# Create a correlation heatmap
sns.heatmap(correlation_matrix, annot=True, cmap='YlOrRd', linewidths=.5)

# Set the title of the heatmap
plt.title("Correlation Heatmap")

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



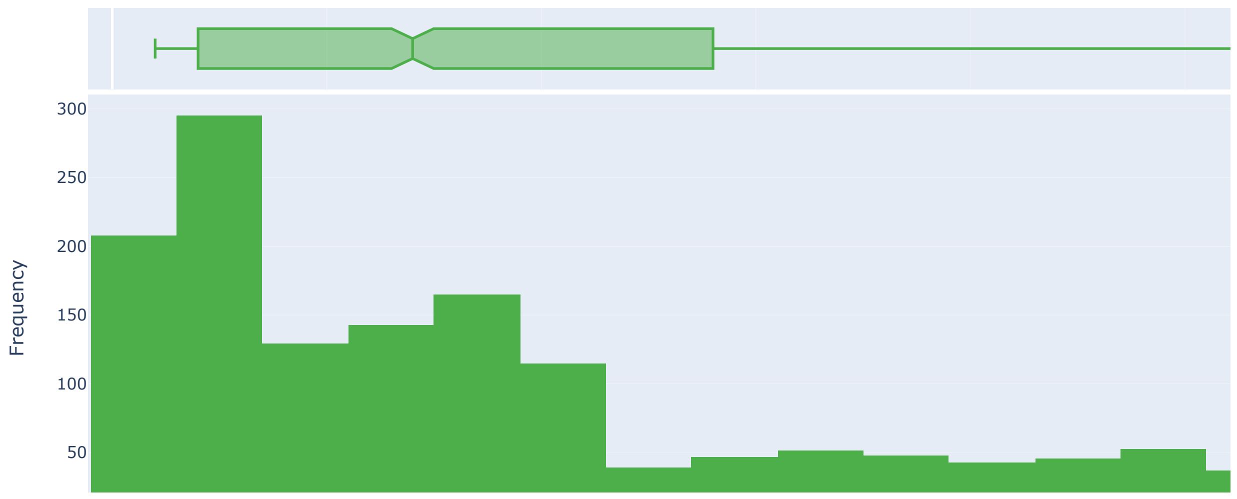
In [11]:

```
# Select only the numeric columns from the 'hr' DataFrame
numeric_columns = hr.select_dtypes(include='number')

# Define a list of colors for the histograms (use a cyclic palette to repeat colors)
colors = px.colors.qualitative.Set1 * (len(numeric_columns) // len(px.colors.qualitative.Set1) + 1)

# Iterate through each numeric column and create a histogram with different colors
for i, column in enumerate(numeric_columns.columns):
    # Create a histogram using Plotly Express with a specific color
    fig = px.histogram(numeric_columns, x=column, nbins=20, marginal="box", color_discrete_sequence=[colors[i]])

    # Update the layout for the individual histogram
    fig.update_layout(
        title=f"Distribution of {column}",
        xaxis_title=column,
        yaxis_title="Frequency",
        # You can uncomment and modify the width and height settings here
        # width=600,
        # height=400,
        showlegend=False
    )

    # Show the histogram
    fig.show()
```

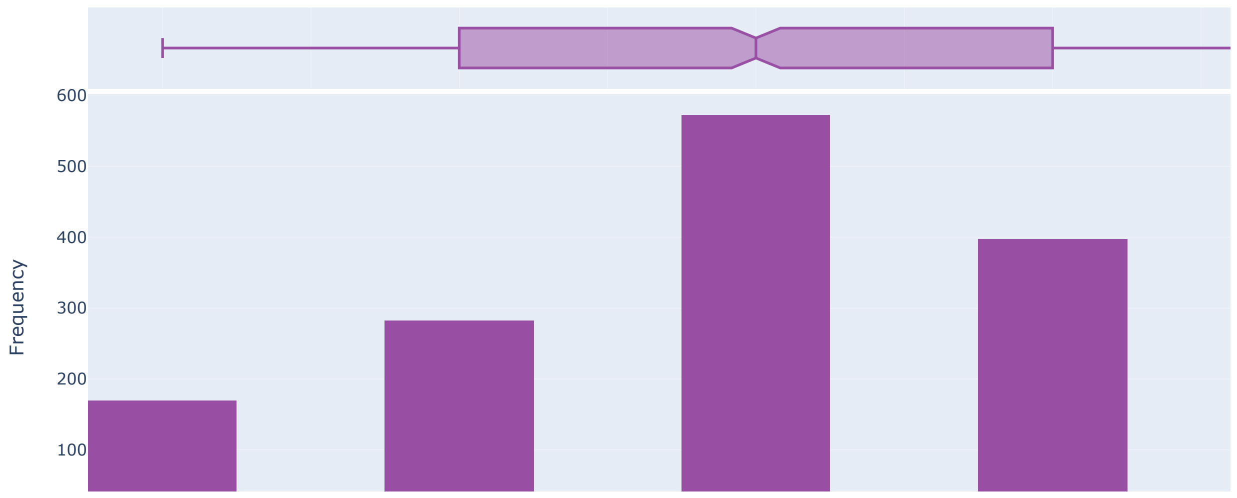

Distribution of Age
Frequency
300
250
200
150
100
50


Distribution of DailyRate
Frequency
120
100
80
60
40
20

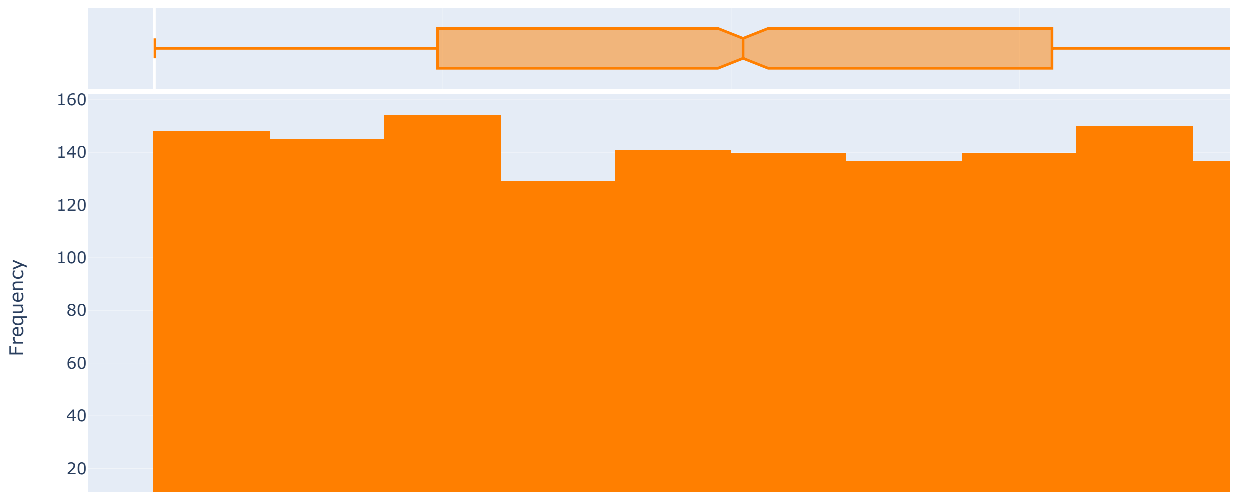
Distribution of DistanceFromHome



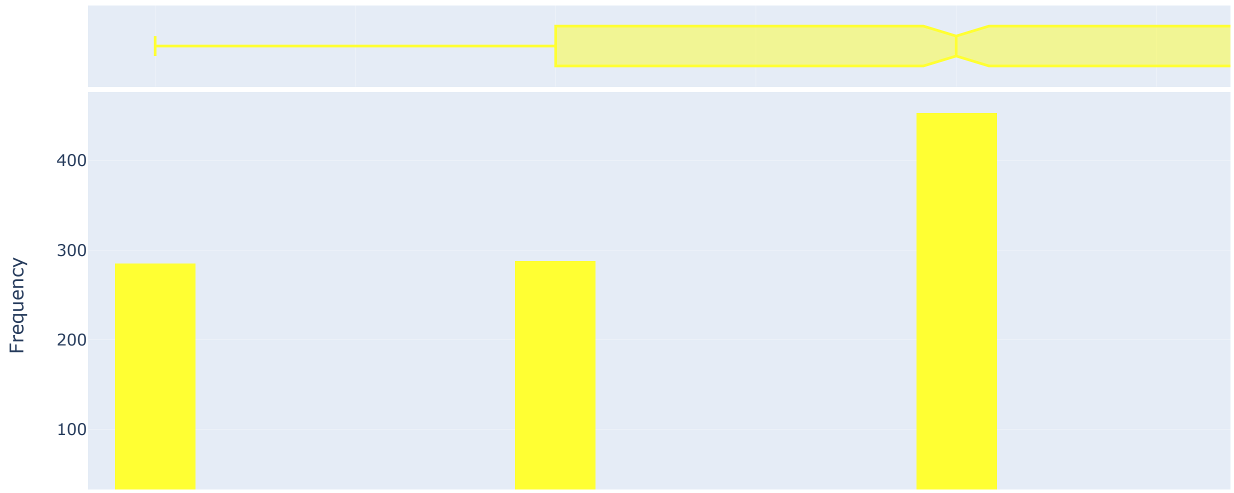
Distribution of Education



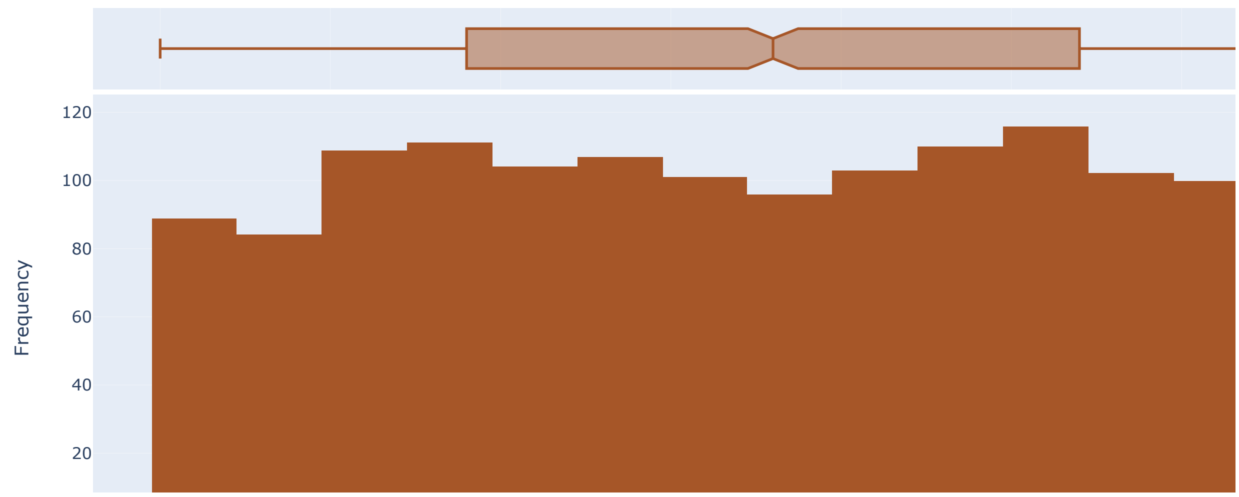
Distribution of EmployeeNumber



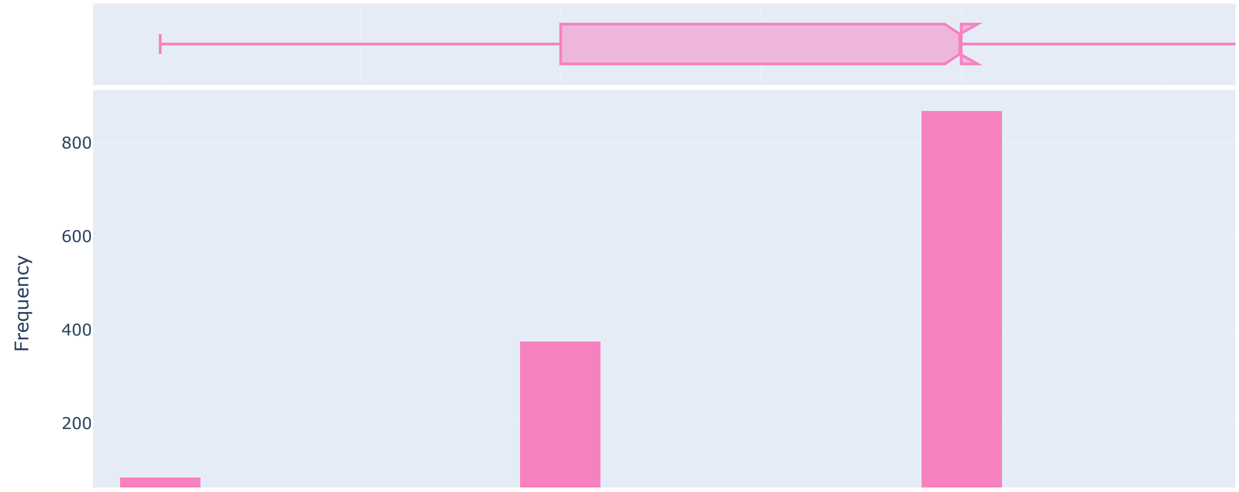
Distribution of EnvironmentSatisfaction

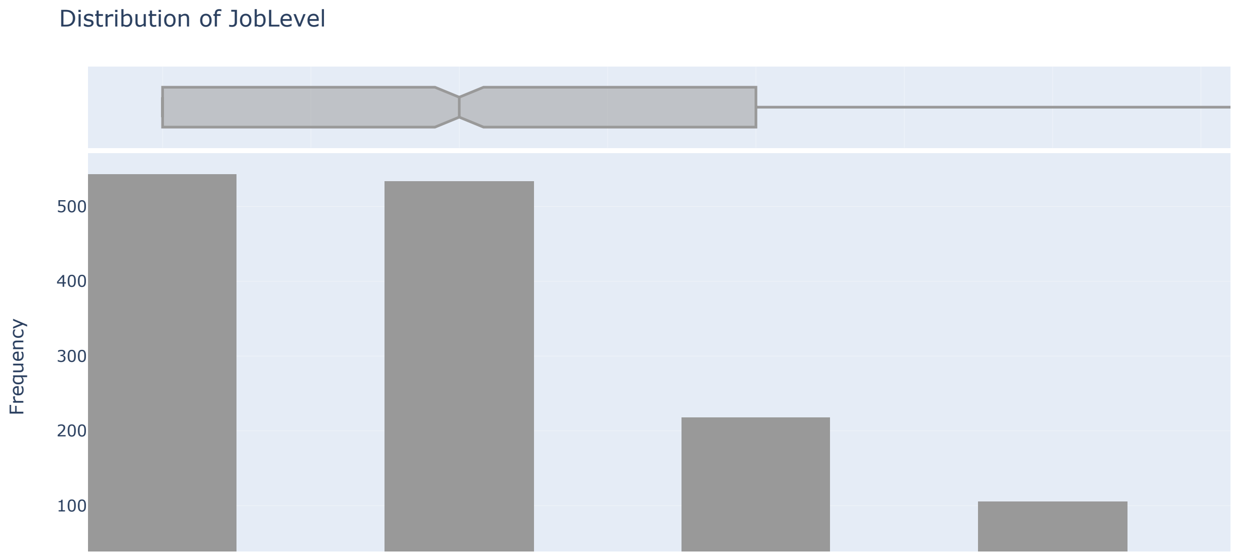


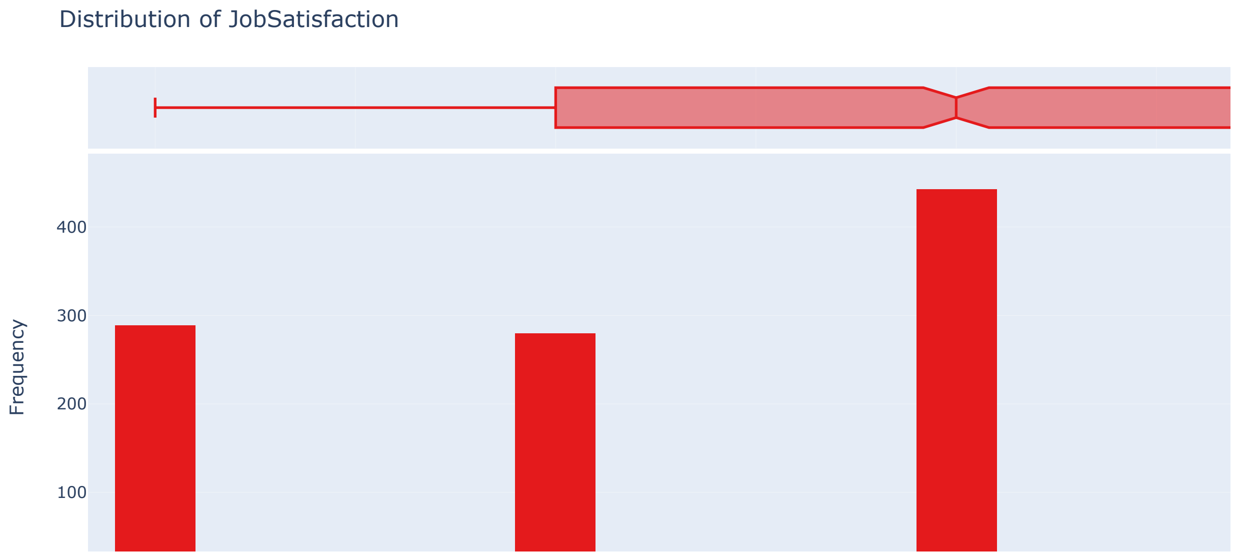
Distribution of HourlyRate



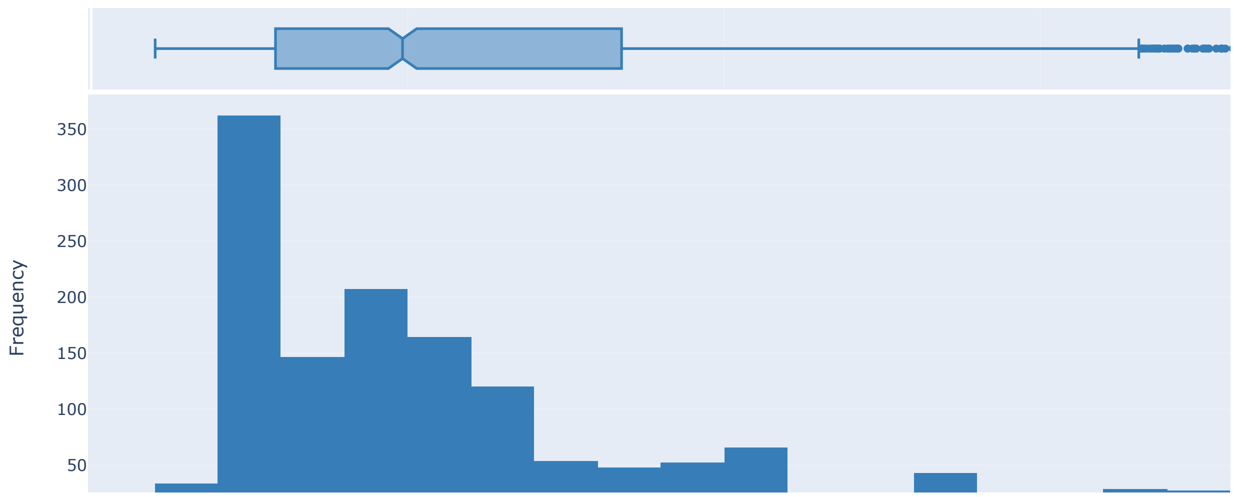
Distribution of JobInvolvement



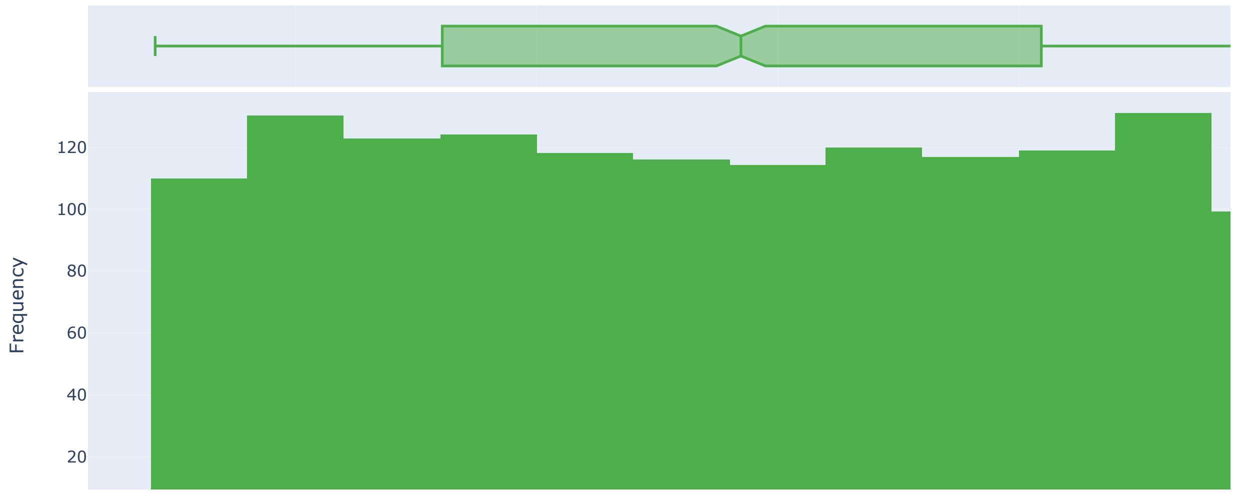

Distribution of JobLevel
Frequency
500
400
300
200
100


Distribution of JobSatisfaction
Frequency
400
300
200
100

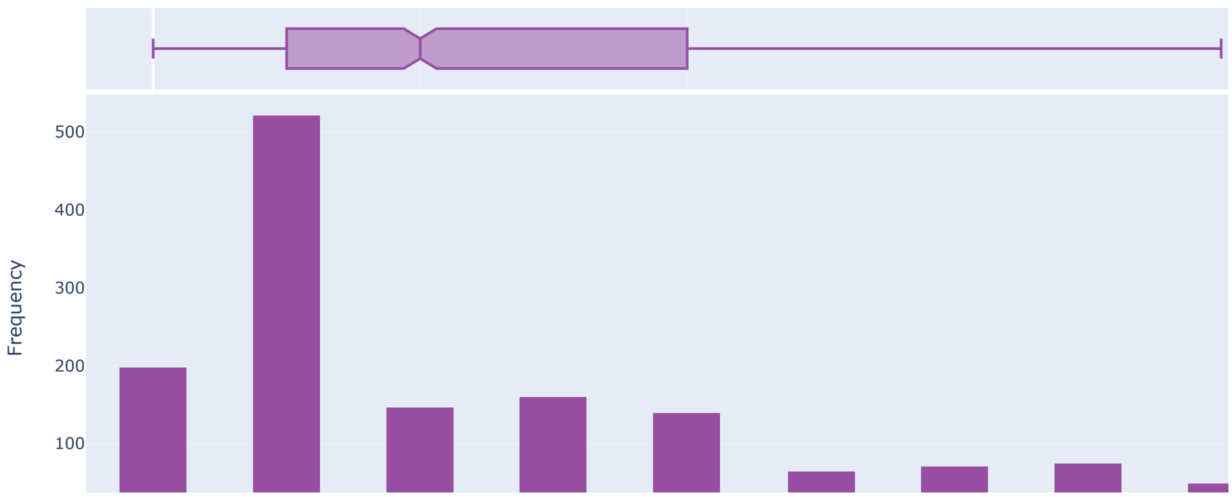
Distribution of MonthlyIncome



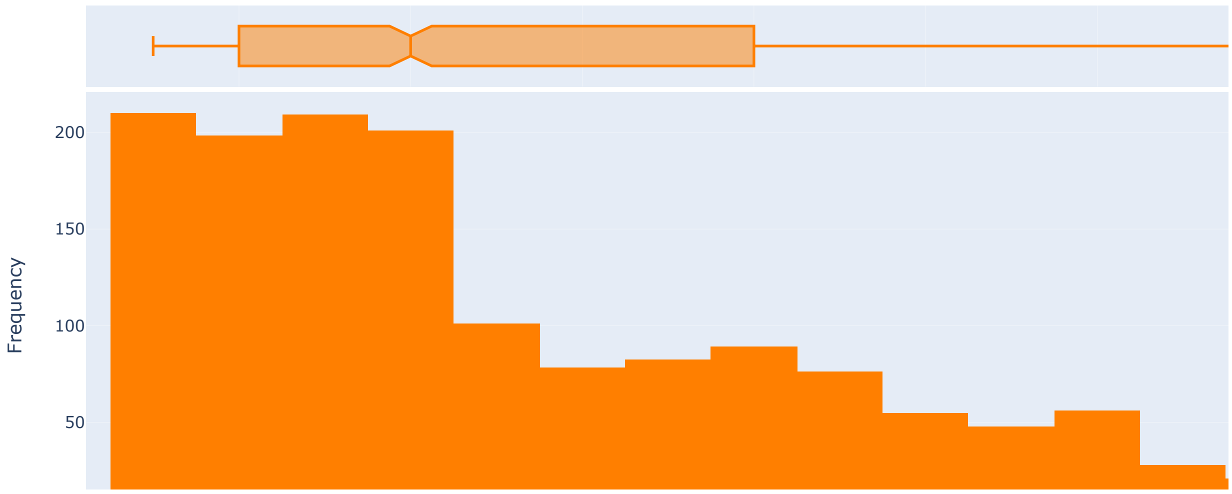
Distribution of MonthlyRate



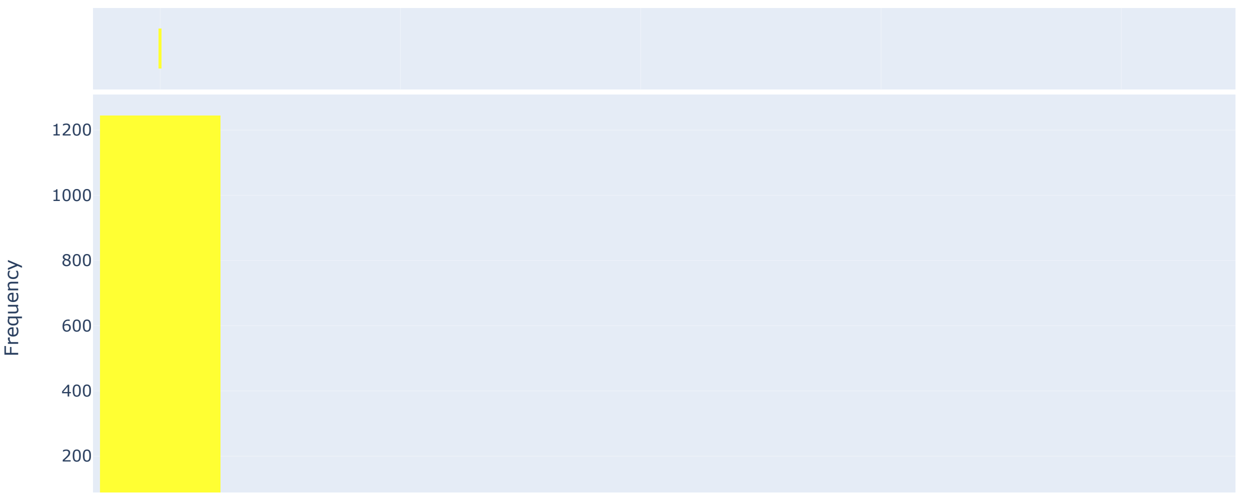
Distribution of NumCompaniesWorked



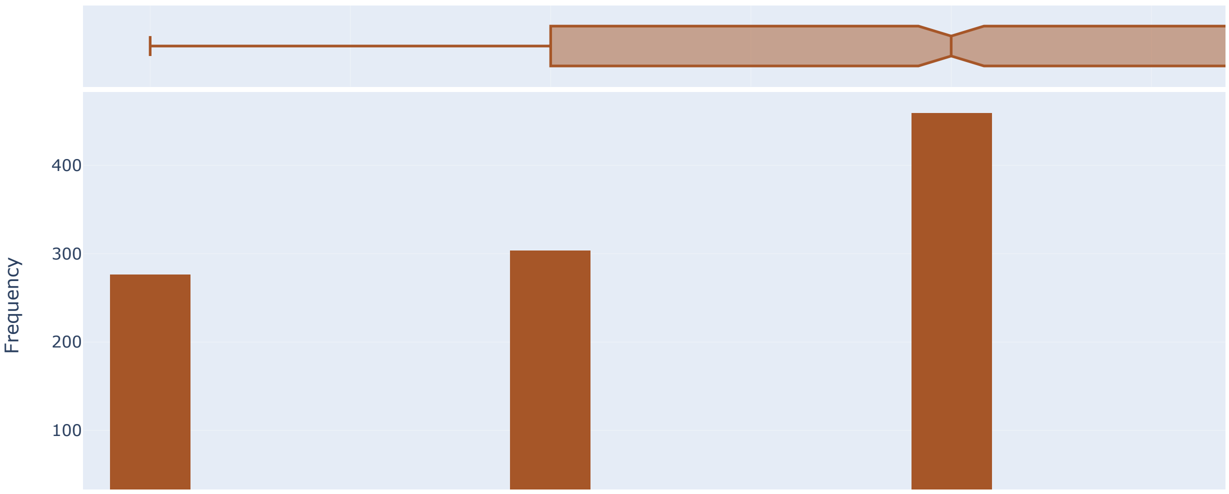
Distribution of PercentSalaryHike



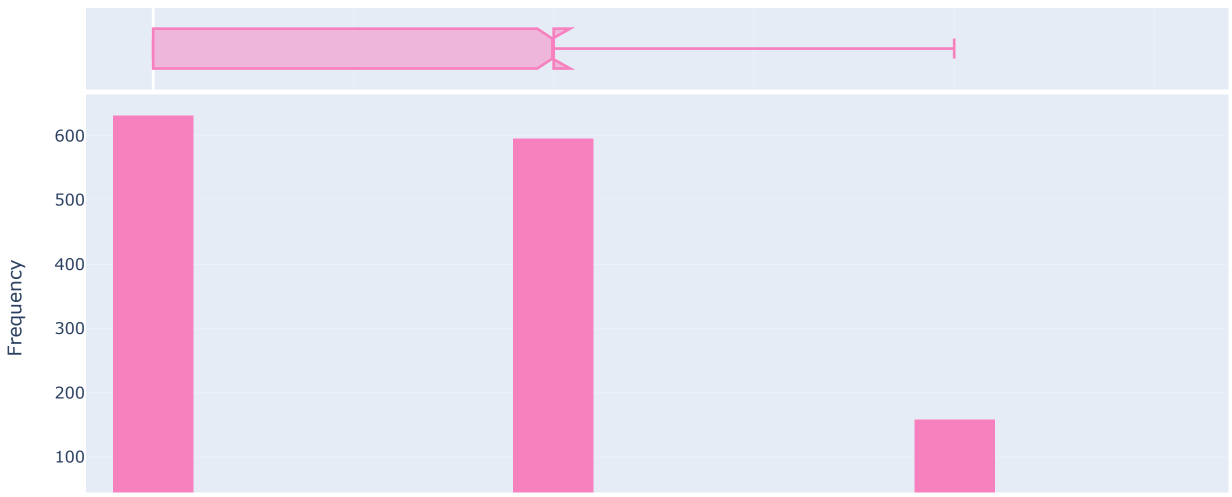
Distribution of PerformanceRating



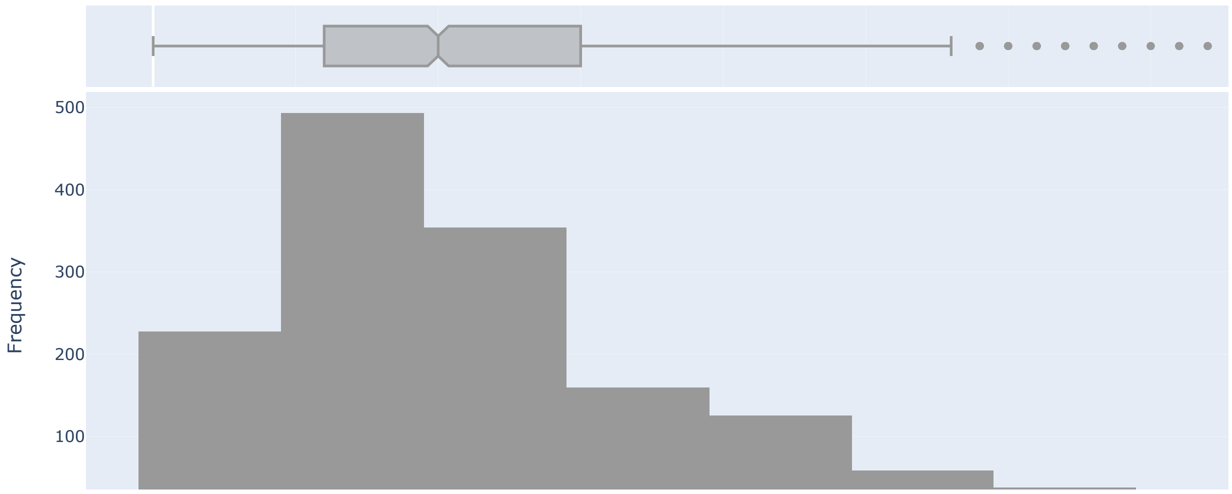
Distribution of RelationshipSatisfaction



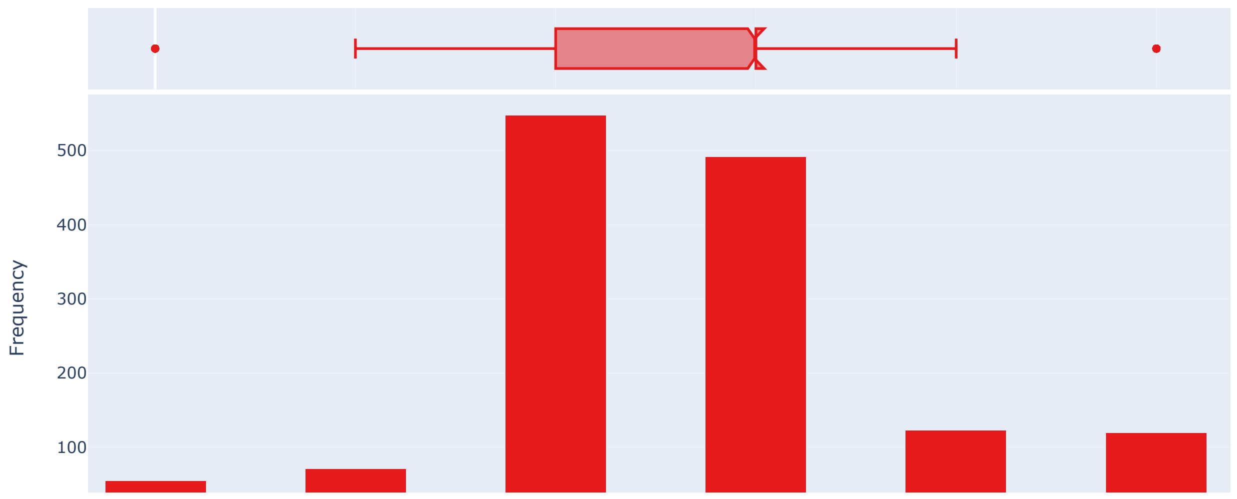
Distribution of StockOptionLevel



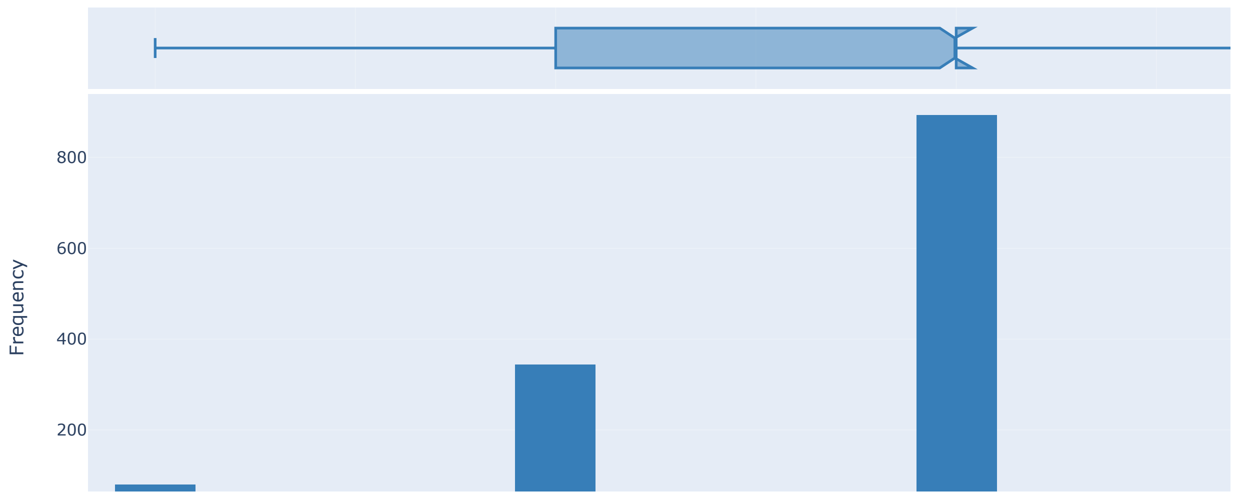
Distribution of TotalWorkingYears



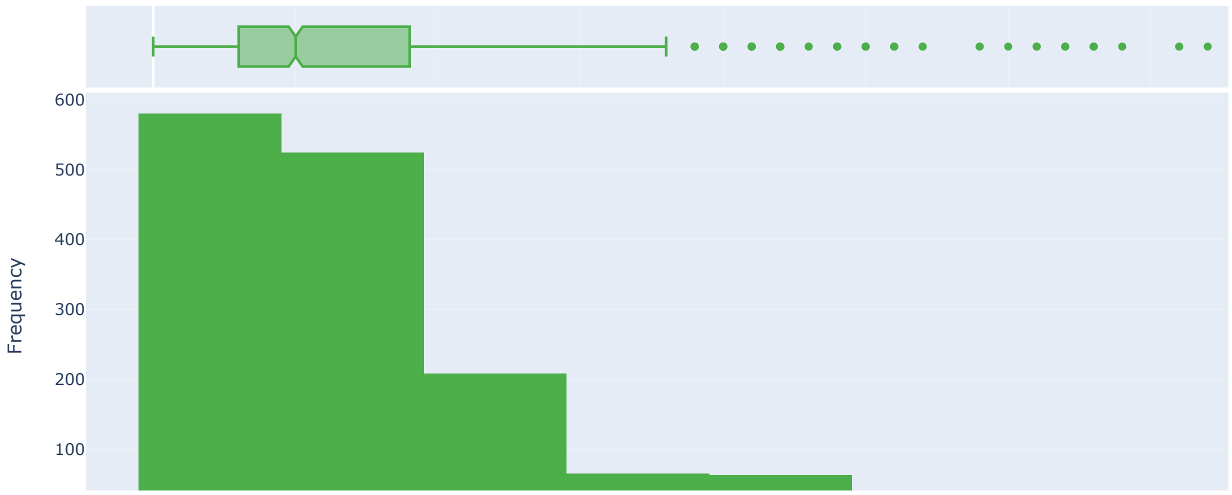
Distribution of TrainingTimesLastYear



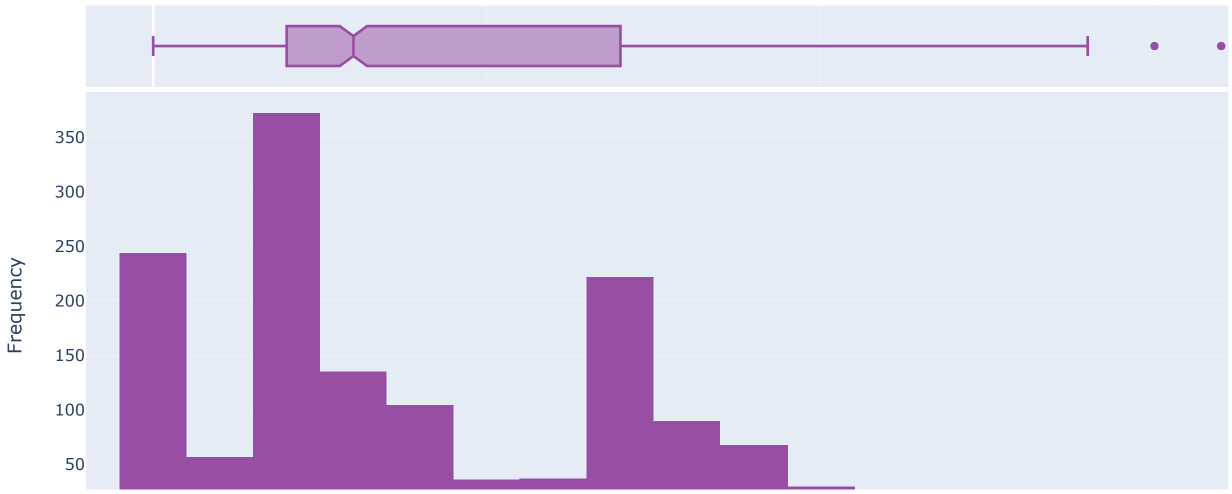
Distribution of WorkLifeBalance

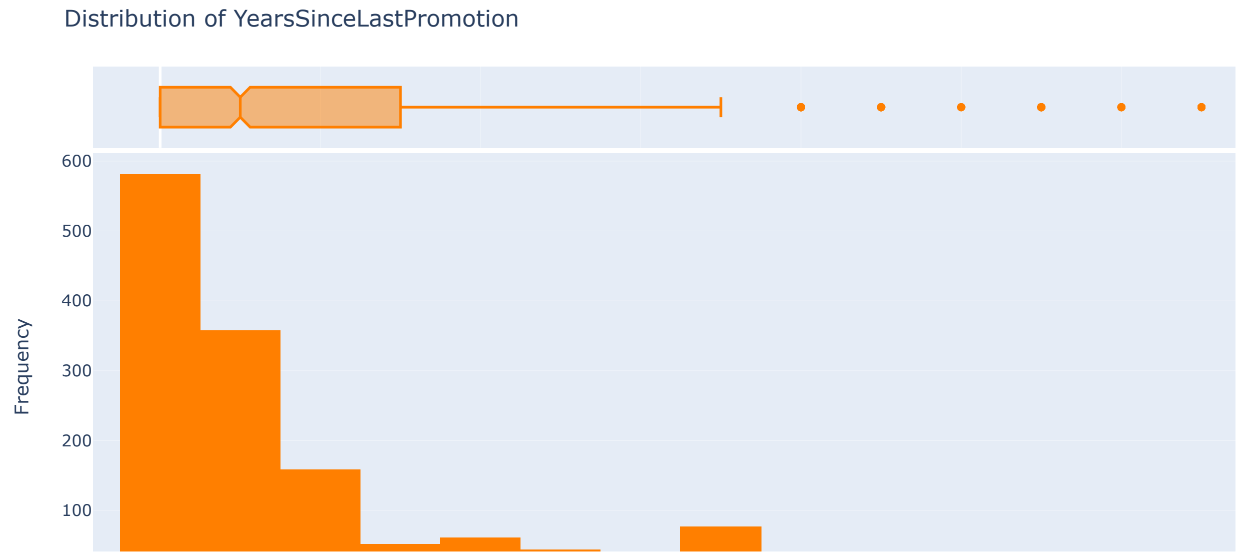


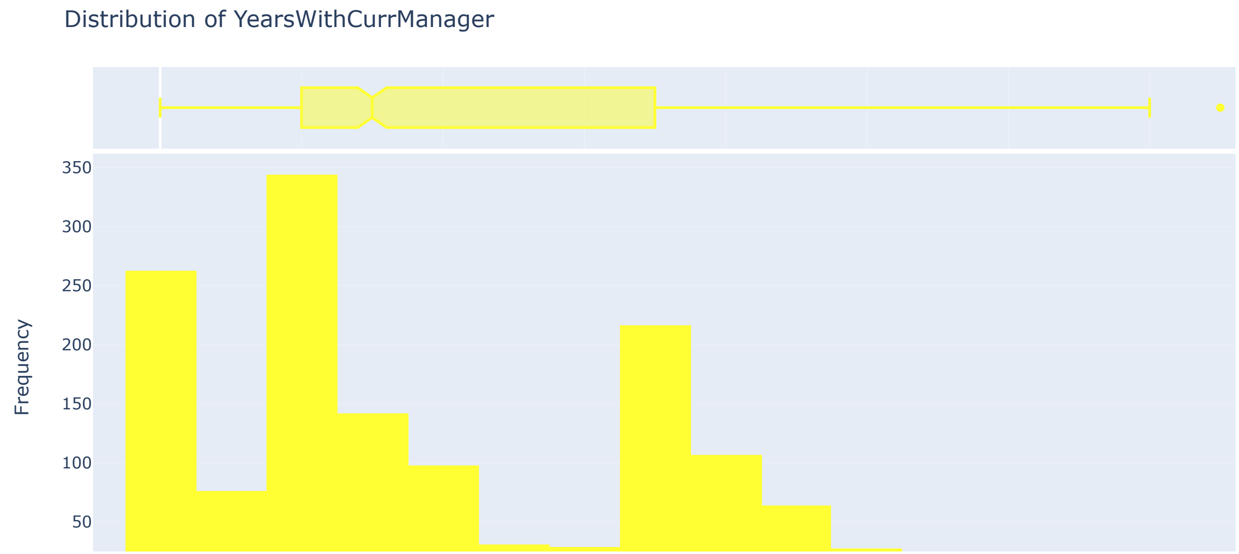
Distribution of YearsAtCompany



Distribution of YearsInCurrentRole




Distribution of YearsSinceLastPromotion

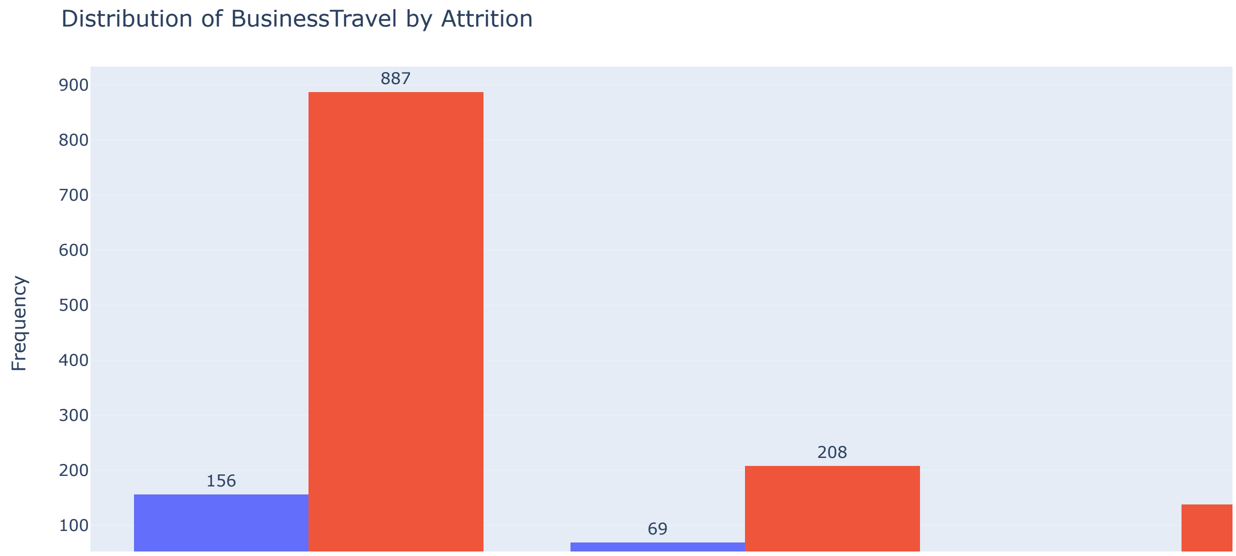

Distribution of YearsWithCurrManager

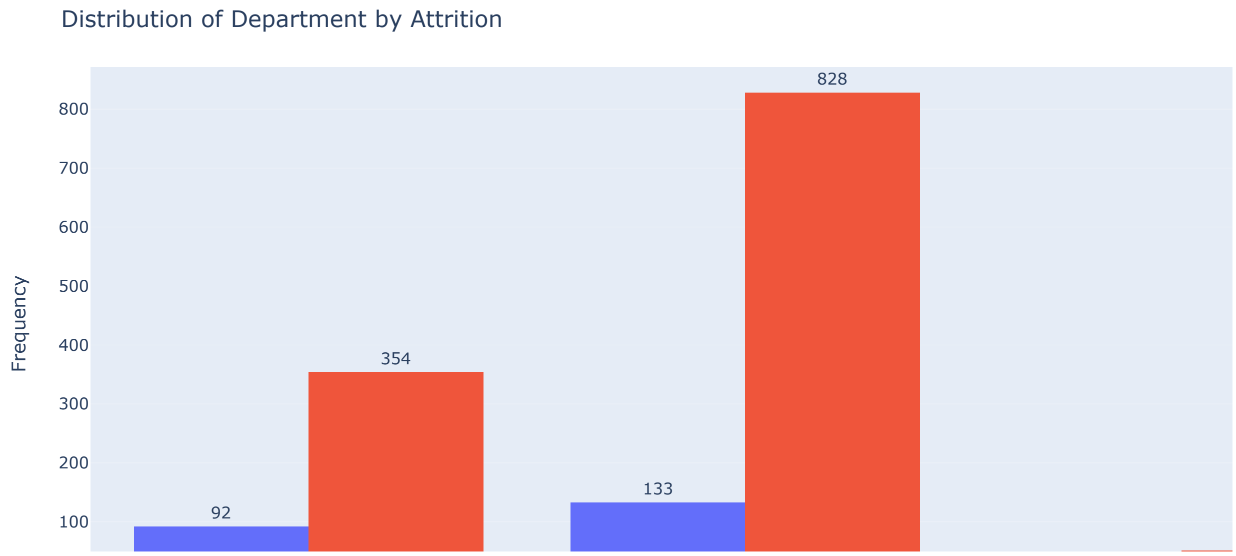
In [12]:

```
# Select columns with data type 'object' and exclude the 'Attrition' column
obj_columns = hr.select_dtypes(include='object').drop(['Attrition'], axis=1)
```

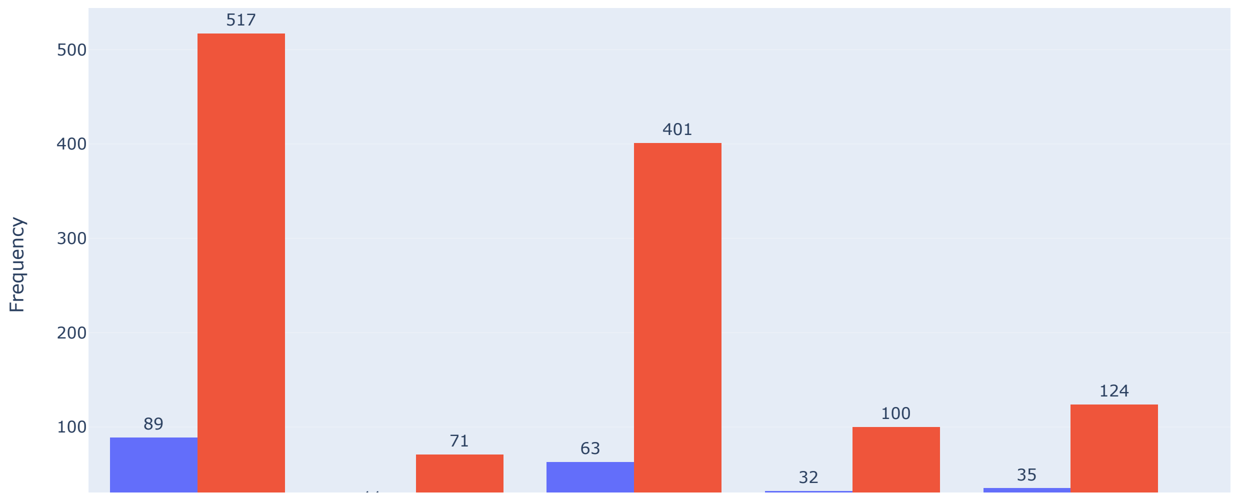
In [13]:

```
# Iterate through each object column in obj_columns
for column in obj_columns.columns:
    # Create a histogram using Plotly Express, grouped by 'Attrition'
    fig = px.histogram(hr, x=column, color='Attrition', barmode='group', text_auto=True, color_discrete_sequence=px.colors.qualitativ

    # Update the layout for the individual histogram
    fig.update_layout(
        title=f"Distribution of {column} by Attrition",
        xaxis_title=column,
        yaxis_title="Frequency",
        # You can uncomment and modify the width and height settings here
        # width=600,
        # height=400,
        showlegend=True
    )

    # Place the text labels outside the bars for clarity
    fig.update_traces(textposition='outside')

    # Show the histogram
    fig.show()
```

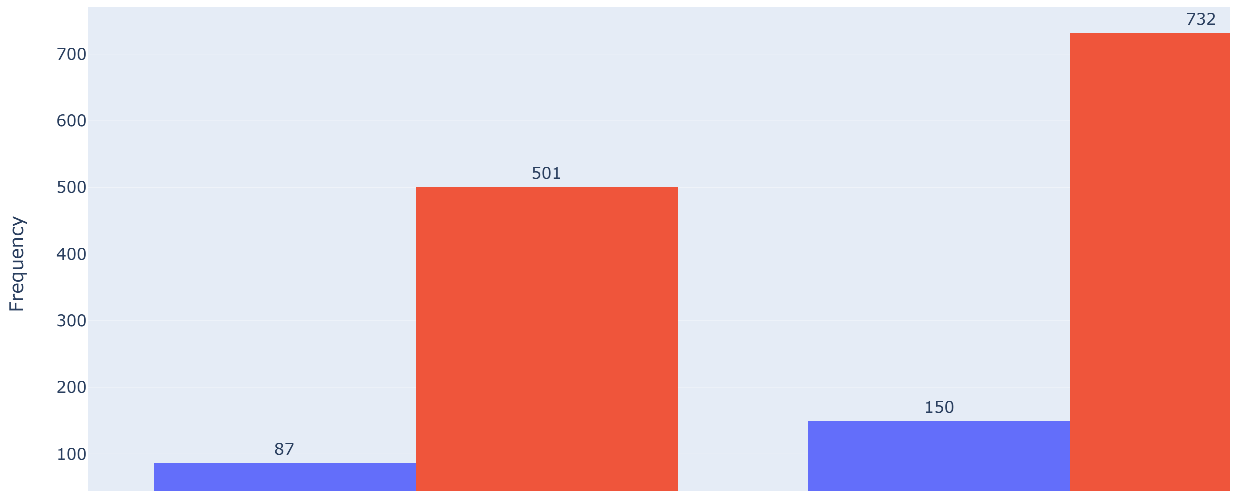

Distribution of BusinessTravel by Attrition
Frequency
900
800
700
600
500
400
300
200
100
887
156
208
69


Distribution of Department by Attrition
Frequency
800
700
600
500
400
300
200
100
828
354
92
133

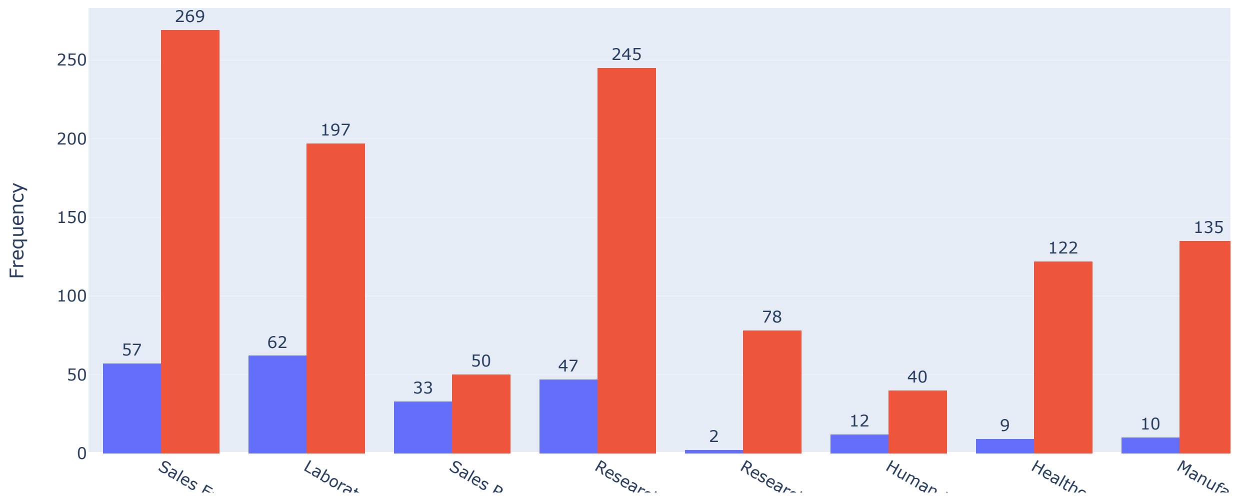
Distribution of EducationField by Attrition



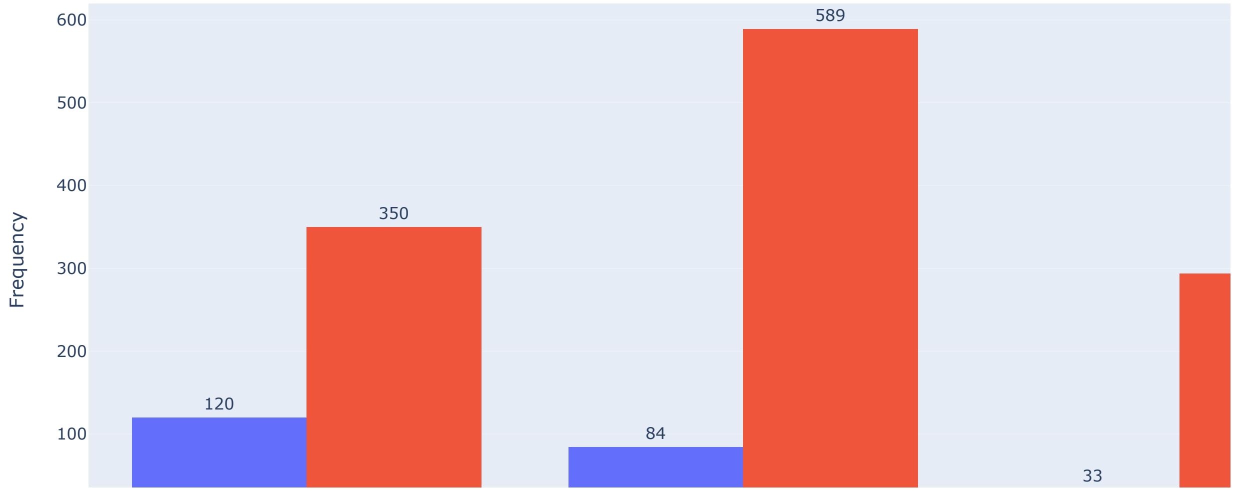
Distribution of Gender by Attrition

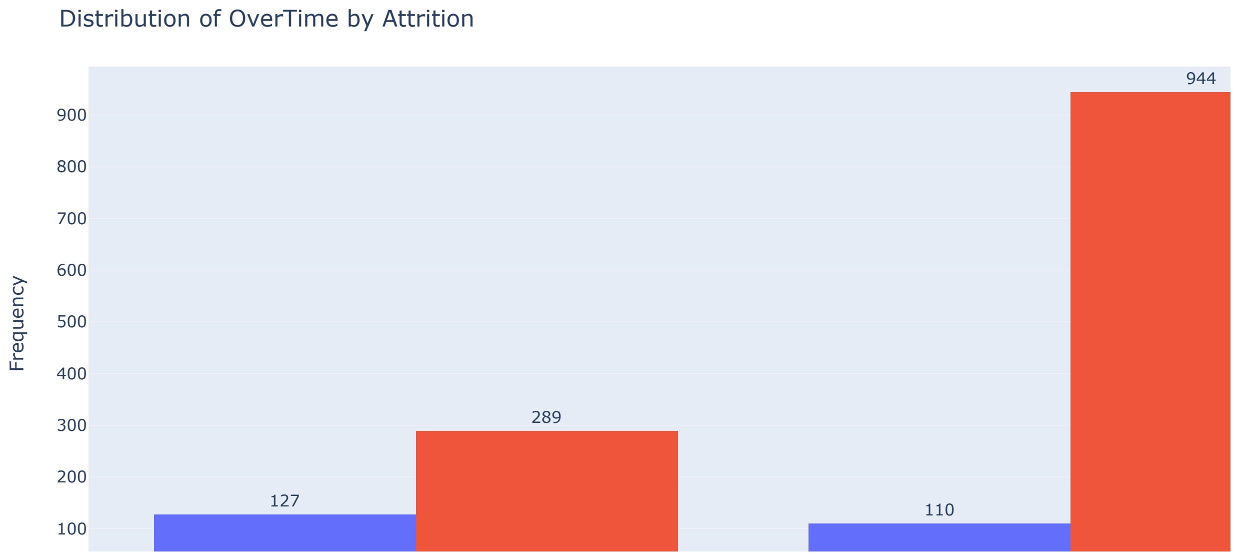


Distribution of JobRole by Attrition

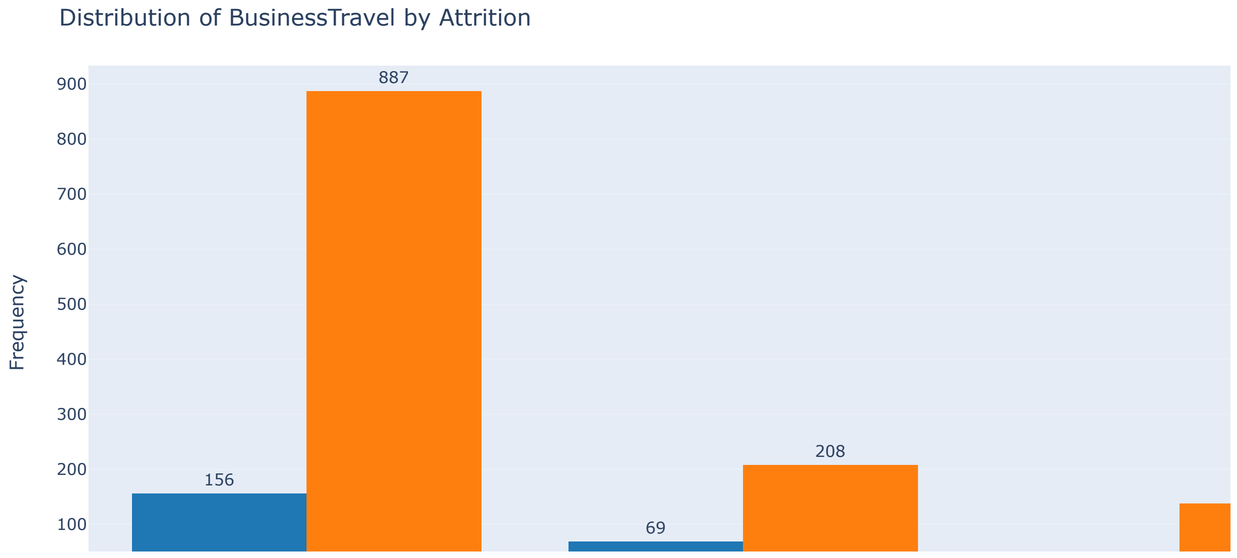


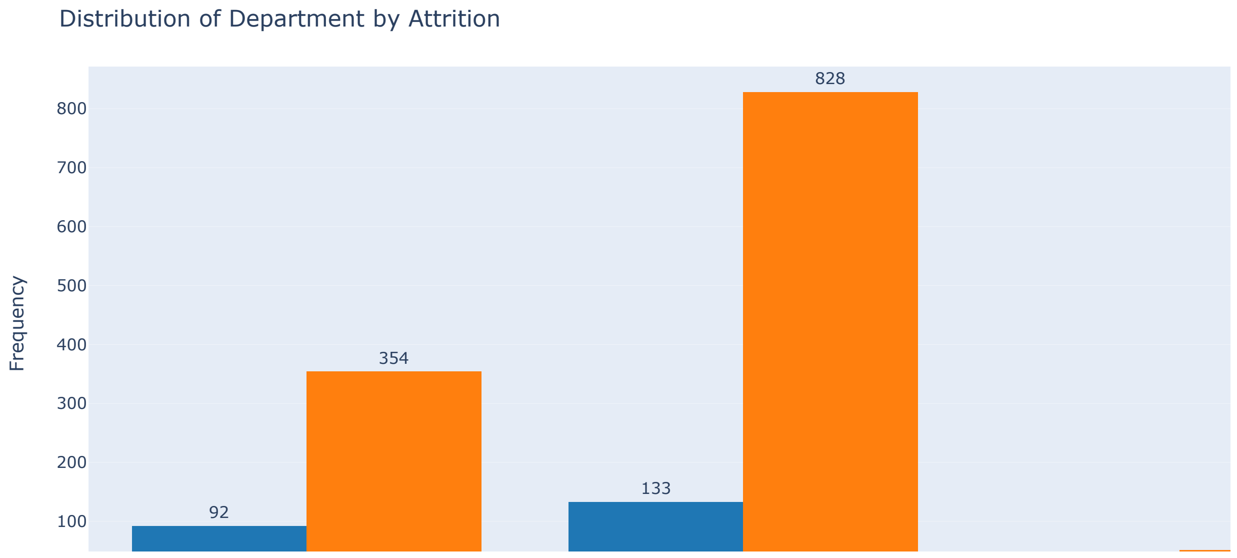
Distribution of MaritalStatus by Attrition

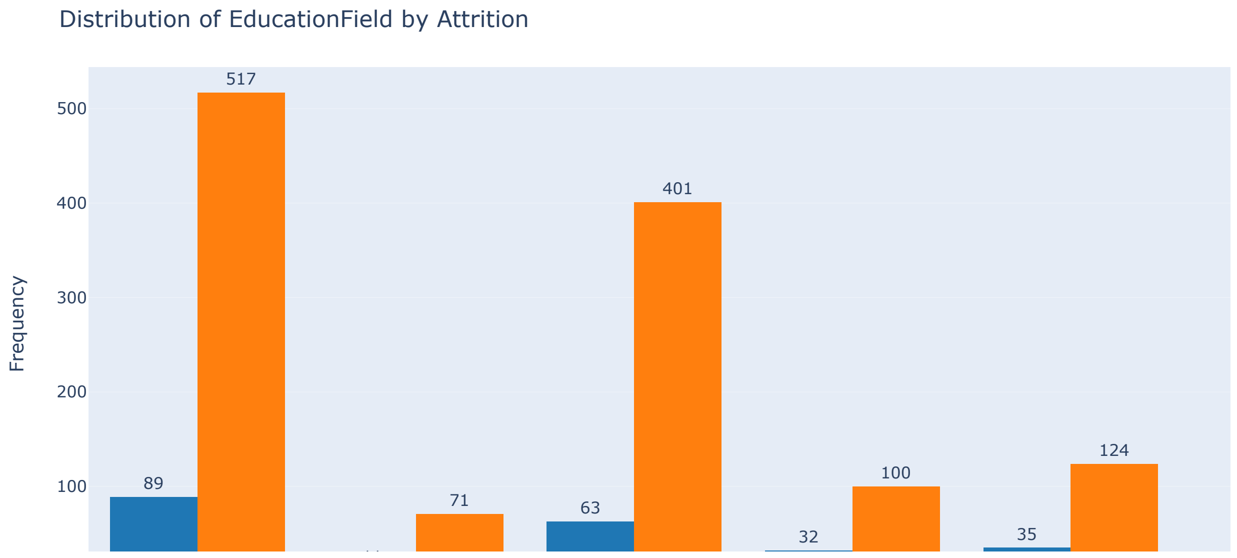




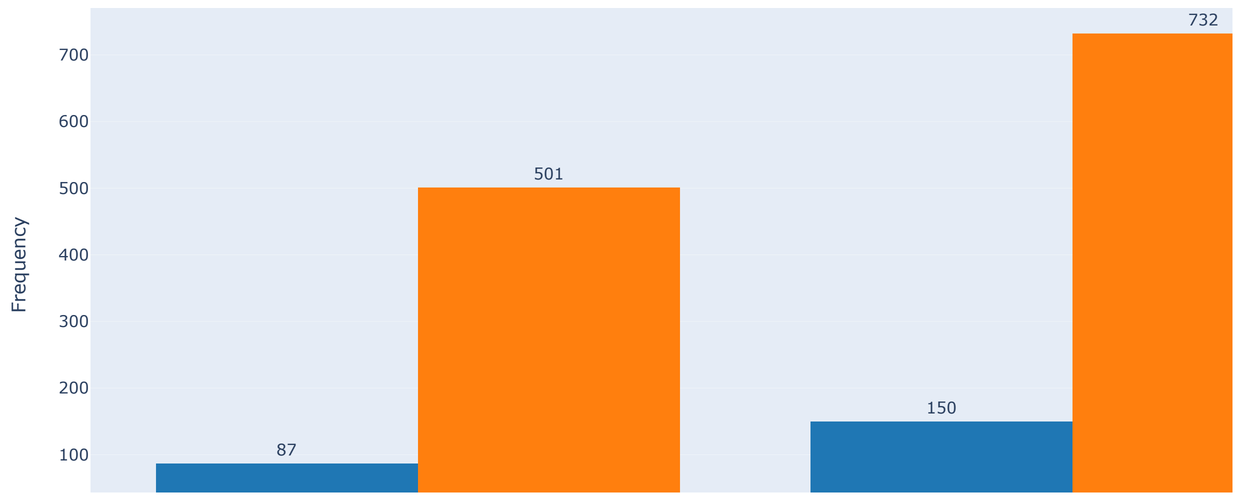
In [14]:
```
# Define a custom color sequence for the histograms
custom_colors = ['#1f77b4', '#ff7f0e']

# Iterate through each object column in obj_columns
for column in obj_columns.columns:
    # Create a histogram using Plotly Express, grouped by 'Attrition', with custom colors
    fig = px.histogram(hr, x=column, color='Attrition', barmode='group', text_auto=True, color_discrete_sequence=custom_colors)

    # Update the layout for the individual histogram
    fig.update_layout(
        title=f"Distribution of {column} by Attrition",
        xaxis_title=column,
        yaxis_title="Frequency",
        # You can uncomment and modify the width and height settings here
        # width=600,
        # height=400,
        showlegend=True
    )

    # Place the text labels outside the bars for clarity
    fig.update_traces(textposition='outside')

    # Show the histogram
    fig.show()
```



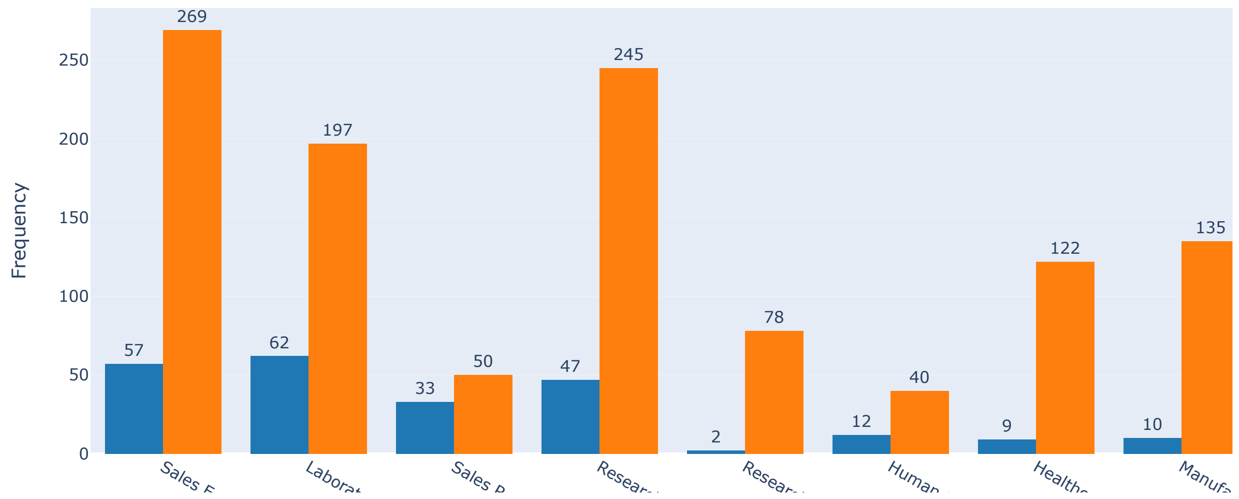

Distribution of Department by Attrition

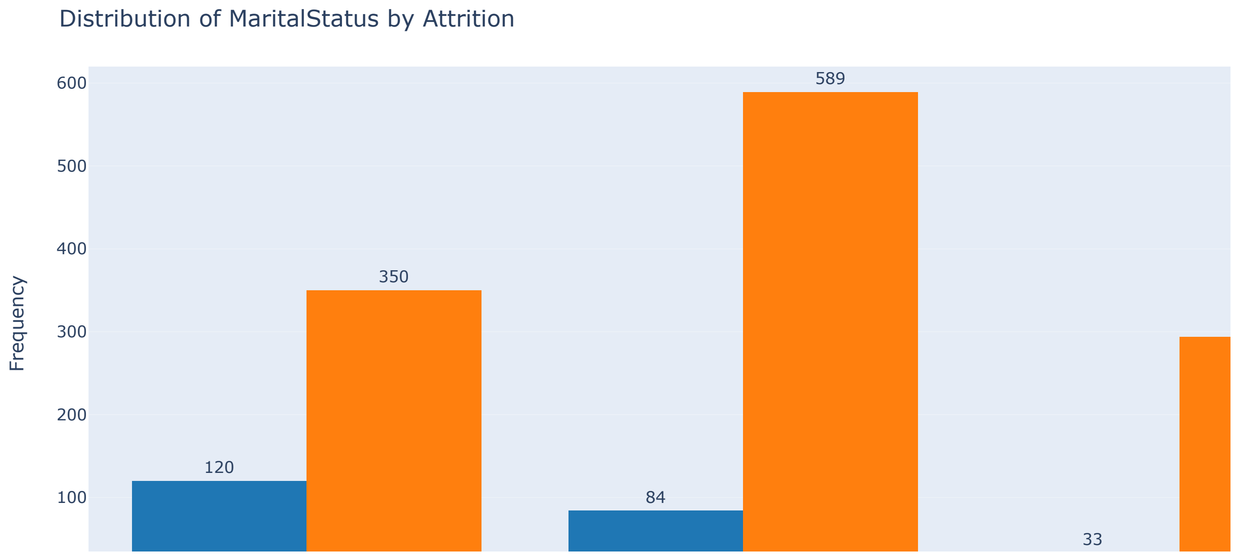

Distribution of EducationField by Attrition

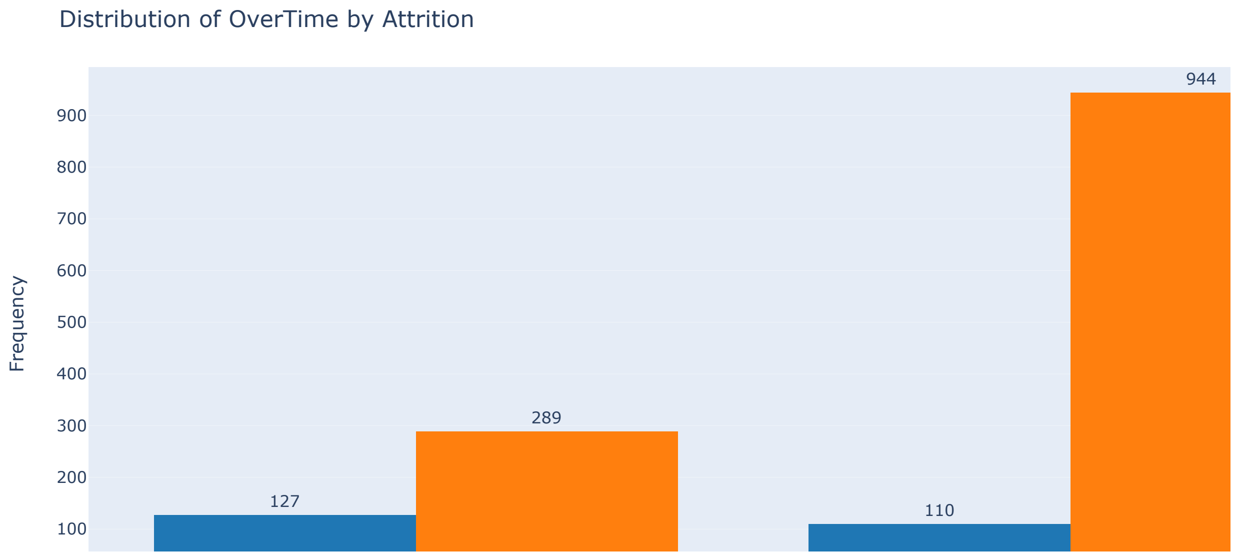
Distribution of Gender by Attrition



Distribution of JobRole by Attrition

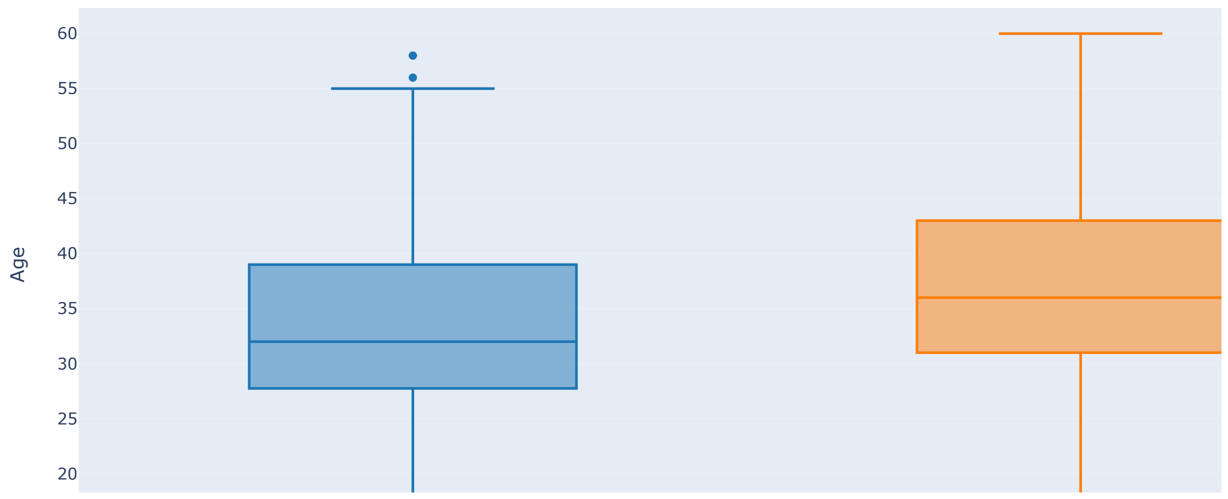






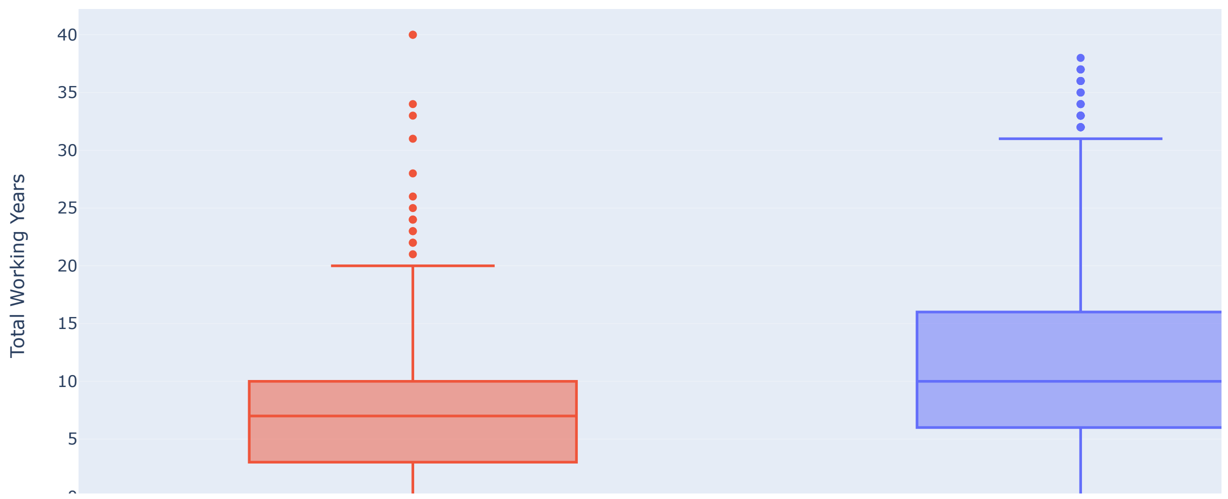
In [15]:
```
# Define a custom color sequence for the box plot
custom_colors = ['#1f77b4', '#ff7f0e']

# Create a box plot to show the distribution of 'Age' by 'Attrition' with custom colors
fig = px.box(hr, x="Attrition", y="Age", color="Attrition", color_discrete_sequence=custom_colors)

# Update the layout for the box plot
fig.update_layout(
    title="Attrition by Age",
    xaxis_title="Attrition",
    yaxis_title="Age",
    showlegend=False
)

# Show the box plot
fig.show()
```

Attrition by Age

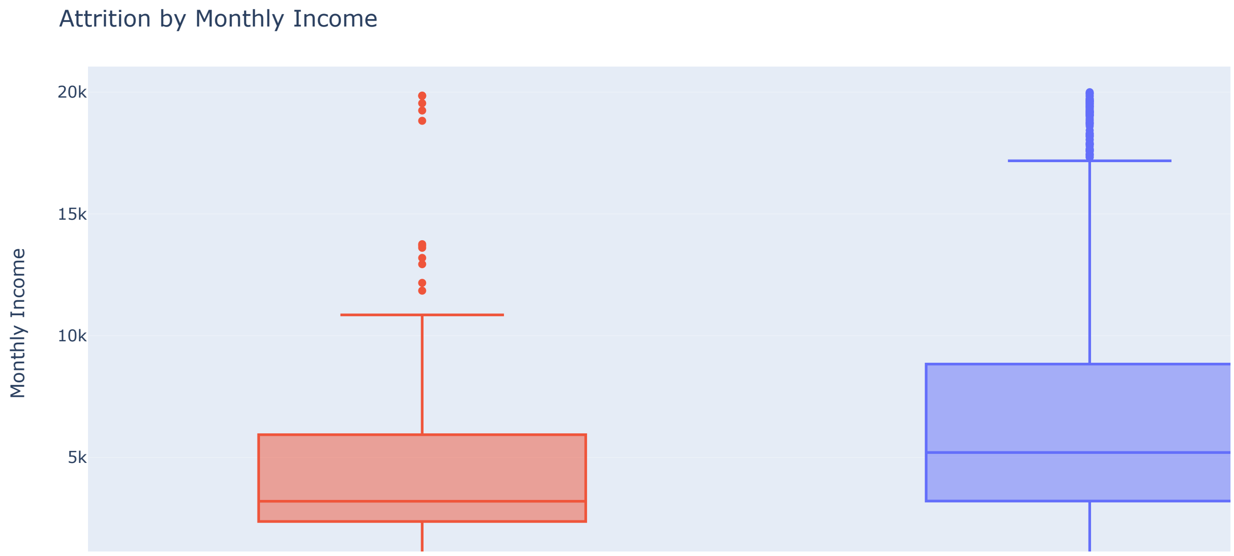


```
In [16]: # Create a box plot to show the distribution of 'TotalWorkingYears' by 'Attrition'
         fig = px.box(hr, x="Attrition", y="TotalWorkingYears", color="Attrition", color_discrete_sequence=[px.colors.qualitative.Plotly[1], p

         # Update the layout for the box plot
         fig.update_layout(
             title="Attrition by Total Working Years",
             xaxis_title="Attrition",
             yaxis_title="Total Working Years",
             showlegend=False
         )

         # Show the box plot
         fig.show()
```

Attrition by Total Working Years

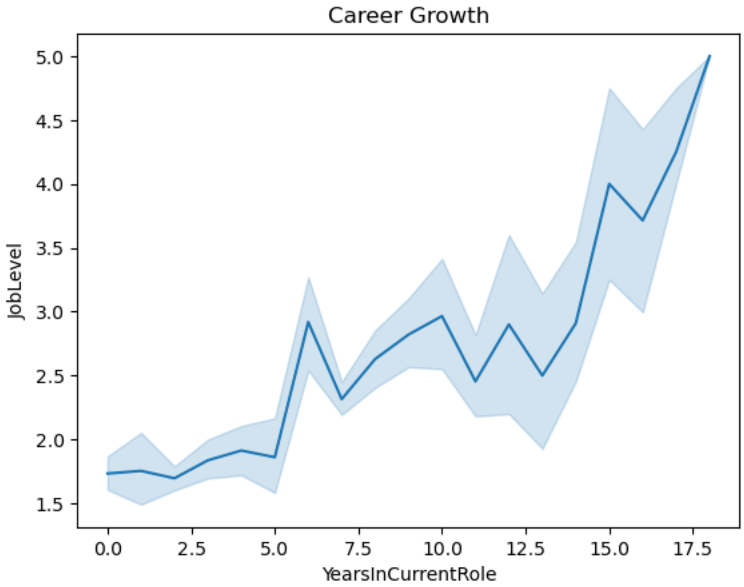


```
In [17]: # Create a box plot to show the distribution of 'MonthlyIncome' by 'Attrition'
         fig = px.box(hr, x="Attrition", y="MonthlyIncome", color="Attrition", color_discrete_sequence=[px.colors.qualitative.Plotly[1], px.cc

         # Update the layout for the box plot
         fig.update_layout(
             title="Attrition by Monthly Income",
             xaxis_title="Attrition",
             yaxis_title="Monthly Income",
             showlegend=False
         )

         # Show the box plot
         fig.show()
```



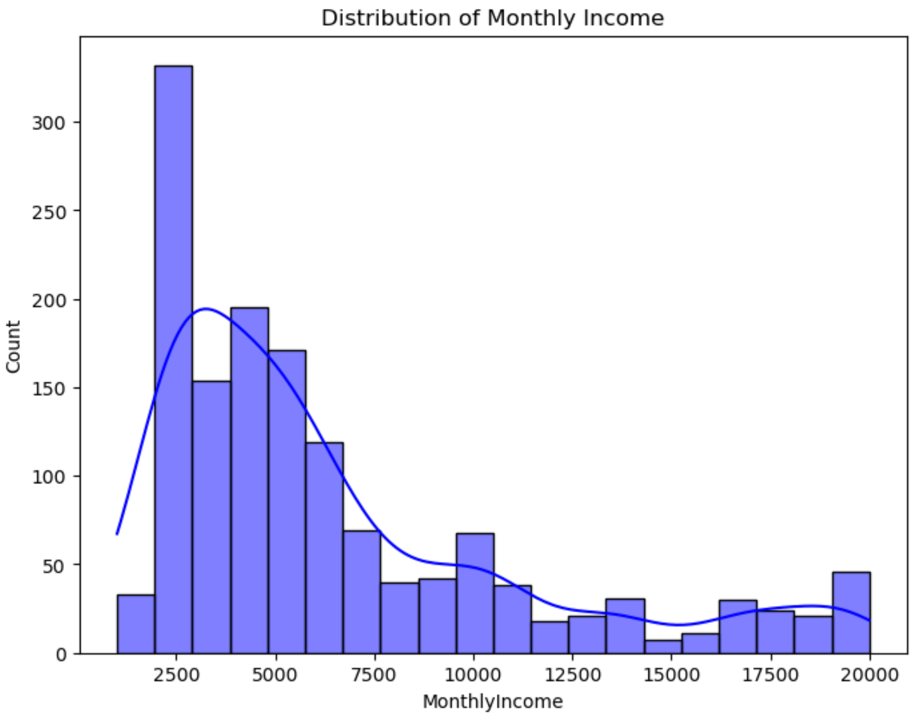
In [18]:

```python
# Create a line plot to show the relationship between 'YearsInCurrentRole' and 'JobLevel'
sns.lineplot(data=hr, y='JobLevel', x='YearsInCurrentRole')

# Set the title for the plot
plt.title('Career Growth')

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



In [19]:

```python
# Plot a histogram for MonthlyIncome
plt.figure(figsize=(8, 6))
sns.histplot(data=hr, x='MonthlyIncome', kde=True, color='blue')
plt.title('Distribution of Monthly Income')
plt.show()
```



In [20]:

```
# Assuming 'hr' contains your HR data with categorical variables
# Perform one-hot encoding on the categorical columns
hr_encoded = pd.get_dummies(hr, columns=['BusinessTravel', 'Department', 'EducationField', 'Gender', 'JobRole', 'MaritalStatus', 'Ove

# Split the data into training and testing sets
X = hr_encoded.drop(['Attrition'], axis=1)
y = hr_encoded['Attrition']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Fit a decision tree classifier
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

# Make predictions
y_pred = clf.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print(f"Accuracy: {accuracy}")
print(report)
```

```
Accuracy: 0.7789115646258503
              precision    recall  f1-score   support

          No       0.88      0.87      0.87       255
         Yes       0.19      0.21      0.20        39

    accuracy                           0.78       294
   macro avg       0.53      0.54      0.53       294
weighted avg       0.79      0.78      0.78       294
```

In [ ]: