adityaacharyaibmproject

August 3, 2023

Data Analysis Project on IBM Employee Dataset

[1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('ibmdataset1.csv', encoding= 'unicode_escape')
display(df)
```

```
      Age            State             Travel  DailyRate  \
0    34.0     UttarPradesh      Travel_Rarely        790
1    35.0      Maharashtra      Travel_Rarely        660
2    24.0         Karnatka  Travel_Frequently        381
3    24.0            Delhi         Non-Travel        830
4    44.0   Madhya Pradesh  Travel_Frequently       1193
..    …                …                  …        …
444  36.0 Himachal Pradesh  Travel_Frequently        884
445  39.0           Kerala      Travel_Rarely        613
446  27.0          Haryana      Travel_Rarely        155
447  49.0            Bihar  Travel_Frequently       1023
448  34.0           Gujrat      Travel_Rarely        628

                 Department  DistanceHome  Education EducationField  \
0                     Sales          24.0          4        Medical
1                     Sales           7.0          1  Life Sciences
2    Research & Development           9.0          3        Medical
3                     Sales          13.0          2  Life Sciences
4    Research & Development           2.0          1        Medical
..                      …            …         …             …
444  Research & Development          23.0          2        Medical
445  Research & Development           6.0          1        Medical
446  Research & Development           4.0          3  Life Sciences
447                   Sales           2.0          3        Medical
448  Research & Development           8.0          3        Medical

     EmployeeCount  EmployeeNumber  …  Income NumCompaniesWorked  Over18  \
0                1            1489  …    4599                  0       Y
1                1            1492  …    2404                  1       Y
```

```
2                    1              1494  …    3172                   2         Y
3                    1              1495  …    2033                   1         Y
4                    1              1496  …   10209                   5         Y
..                 …              …   …      …                 …      …
444                  1              2087  …    2571                   4         Y
445                  1              2088  …    9991                   4         Y
446                  1              2089  …    6142                   1         Y
447                  1              2090  …    5390                   2         Y
448                  1              2091  …    4404                   2         Y

     OverTime  PercentSalaryHike PerformanceRating  TotalWorkingYears  \
0         Yes                 23                 4                 16
1          No                 13                 3                  1
2         Yes                 11                 3                  4
3          No                 13                 3                  1
4         Yes                 18                 3                 16
..        …                 …                 …                 …
444        No                 17                 3                 17
445        No                 15                 3                  9
446       Yes                 20                 4                  6
447        No                 14                 3                 17
448        No                 12                 3                  6

    YearsAtCompany  unnamed1  unnamed2
0               15       NaN       NaN
1                1       NaN       NaN
2                0       NaN       NaN
3                1       NaN       NaN
4                2       NaN       NaN
..             …       …       …
444              5       NaN       NaN
445              7       NaN       NaN
446              6       NaN       NaN
447              9       NaN       NaN
448              4       NaN       NaN

[449 rows x 28 columns]
```

checking the shape of the dataset

[2]:
```
df.shape
```

[2]:
```
(449, 28)
```

printing the first few rows of the dataset

[3]:
```
df.head()
```

[3]:
```
    Age           State             Travel  DailyRate              Department  \
0  34.0    UttarPradesh      Travel_Rarely        790                   Sales
1  35.0     Maharashtra      Travel_Rarely        660                   Sales
2  24.0        Karnatka  Travel_Frequently        381  Research & Development
3  24.0           Delhi         Non-Travel        830                   Sales
4  44.0  Madhya Pradesh  Travel_Frequently       1193  Research & Development

   DistanceHome  Education EducationField  EmployeeCount  EmployeeNumber  …  \
0          24.0          4        Medical              1            1489  …
1           7.0          1  Life Sciences              1            1492  …
2           9.0          3        Medical              1            1494  …
3          13.0          2  Life Sciences              1            1495  …
4           2.0          1        Medical              1            1496  …

   Income NumCompaniesWorked  Over18  OverTime  PercentSalaryHike  \
0    4599                  0       Y       Yes                 23
1    2404                  1       Y        No                 13
2    3172                  2       Y       Yes                 11
3    2033                  1       Y        No                 13
4   10209                  5       Y       Yes                 18

  PerformanceRating  TotalWorkingYears YearsAtCompany  unnamed1  unnamed2
0                 4                 16             15       NaN       NaN
1                 3                  1              1       NaN       NaN
2                 3                  4              0       NaN       NaN
3                 3                  1              1       NaN       NaN
4                 3                 16              2       NaN       NaN

[5 rows x 28 columns]
```

checking the shape of the dataset

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 449 entries, 0 to 448
Data columns (total 28 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       449 non-null    float64
 1   State                     449 non-null    object
 2   Travel                    449 non-null    object
 3   DailyRate                 449 non-null    int64
 4   Department                449 non-null    object
 5   DistanceHome              446 non-null    float64
 6   Education                 449 non-null    int64
 7   EducationField            448 non-null    object
 8   EmployeeCount             449 non-null    int64
```

```
 9   EmployeeNumber           449 non-null    int64
 10  EnvironmentSatisfaction  448 non-null    float64
 11  Gender                   449 non-null    object
 12  HourlyRate               449 non-null    int64
 13  JobLevel                 449 non-null    int64
 14  error0##                 0 non-null      float64
 15  JobRole                  449 non-null    object
 16  Job satisfaction         0 non-null      float64
 17  Married                  449 non-null    object
 18  Income                   449 non-null    int64
 19  NumCompaniesWorked       449 non-null    int64
 20  Over18                   446 non-null    object
 21  OverTime                 449 non-null    object
 22  PercentSalaryHike        449 non-null    int64
 23  PerformanceRating        449 non-null    int64
 24  TotalWorkingYears        449 non-null    int64
 25  YearsAtCompany           449 non-null    int64
 26  unnamed1                 0 non-null      float64
 27  unnamed2                 0 non-null      float64
dtypes: float64(7), int64(12), object(9)
memory usage: 82.5+ KB
```

checking the columns of the dataset

[5]:
```
df.columns
```

[5]:
```
Index(['Age', 'State', 'Travel', 'DailyRate', 'Department', 'DistanceHome',
       'Education', 'EducationField', 'EmployeeCount', 'EmployeeNumber',
       'EnvironmentSatisfaction', 'Gender', 'HourlyRate', 'JobLevel',
       'error0##', 'JobRole', 'Job satisfaction', 'Married', 'Income',
       'NumCompaniesWorked', 'Over18', 'OverTime', 'PercentSalaryHike',
       'PerformanceRating', 'TotalWorkingYears', 'YearsAtCompany', 'unnamed1',
       'unnamed2'],
      dtype='object')
```

Data Cleaning

as the data above has several empty column so we have to drop or delete them

[6]:
```
df.drop(['unnamed1', 'unnamed2','error0##','Job satisfaction'], axis=1,␣
 ↪inplace=True)
```

now as the columns has been deleted we will now check for any null values in the rows

[7]:
```
pd.isnull(df).sum()
```

[7]:
```
Age                        0
State                      0
Travel                     0
```

```
DailyRate                   0
Department                  0
DistanceHome                3
Education                   0
EducationField              1
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     1
Gender                      0
HourlyRate                  0
JobLevel                    0
JobRole                     0
Married                     0
Income                      0
NumCompaniesWorked          0
Over18                      3
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
TotalWorkingYears           0
YearsAtCompany              0
dtype: int64
```

there are some of the values with the sum null so we have to drop or delete them using dropna function

[8]:
```
df.dropna(inplace=True)
```

checking that the values have been deleted

[9]:
```
pd.isnull(df).sum()
```

[9]:
```
Age                         0
State                       0
Travel                      0
DailyRate                   0
Department                  0
DistanceHome                0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobLevel                    0
JobRole                     0
Married                     0
```

```
Income                      0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
TotalWorkingYears           0
YearsAtCompany              0
dtype: int64
```

checking for any data type errors

[10]:
```
df.dtypes
```

[10]:
```
Age                         float64
State                        object
Travel                       object
DailyRate                     int64
Department                   object
DistanceHome                float64
Education                     int64
EducationField               object
EmployeeCount                 int64
EmployeeNumber                int64
EnvironmentSatisfaction     float64
Gender                       object
HourlyRate                    int64
JobLevel                      int64
JobRole                      object
Married                      object
Income                        int64
NumCompaniesWorked            int64
Over18                       object
OverTime                     object
PercentSalaryHike             int64
PerformanceRating             int64
TotalWorkingYears             int64
YearsAtCompany                int64
dtype: object
```

as age cannot be a float value so we have to change its data type to integer value

[11]:
```
df['Age'] = df['Age'].astype('int')
```

checking for the changed data type

[12]:
```
df['Age'].dtypes
```

[12]: dtype('int32')

checking for any columns name errors which can be provided more appropriate names

[13]:
```
df.columns
```

[13]:
```
Index(['Age', 'State', 'Travel', 'DailyRate', 'Department', 'DistanceHome',
       'Education', 'EducationField', 'EmployeeCount', 'EmployeeNumber',
       'EnvironmentSatisfaction', 'Gender', 'HourlyRate', 'JobLevel',
       'JobRole', 'Married', 'Income', 'NumCompaniesWorked', 'Over18',
       'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'TotalWorkingYears', 'YearsAtCompany'],
      dtype='object')
```

renaming them one by one

[14]:
```
df.rename(columns= {'DistanceHome':'DistanceFromHome'},inplace=True)
```

[15]:
```
df.rename(columns= {'Married':'MaritalStatus'},inplace=True)
```

[16]:
```
df.rename(columns= {'Travel':'BusinessTravel'},inplace=True)
```

Performing Descriptive Satistics

[17]:
```
df.describe()
```

[17]:
```
              Age    DailyRate  DistanceFromHome   Education  EmployeeCount  \
count  441.000000   441.000000        441.000000  441.000000          441.0
mean    36.696145   776.482993          9.845805    3.004535            1.0
std      8.415797   388.428279          8.412317    0.997714            0.0
min     18.000000   104.000000          1.000000    1.000000            1.0
25%     31.000000   461.000000          2.000000    2.000000            1.0
50%     36.000000   728.000000          8.000000    3.000000            1.0
75%     42.000000  1142.000000         15.000000    4.000000            1.0
max     60.000000  1495.000000         29.000000    5.000000            1.0

       EmployeeNumber  EnvironmentSatisfaction  HourlyRate    JobLevel  \
count      441.000000               441.000000  441.000000  441.000000
mean      1803.147392                 2.746032   66.643991    2.004535
std        176.348347                 1.105313   20.515576    1.031317
min       1489.000000                 1.000000   30.000000    1.000000
25%       1651.000000                 2.000000   48.000000    1.000000
50%       1799.000000                 3.000000   67.000000    2.000000
75%       1966.000000                 4.000000   85.000000    2.000000
max       2091.000000                 4.000000  100.000000    5.000000

           Income  NumCompaniesWorked  PercentSalaryHike  PerformanceRating  \
count  441.000000          441.000000         441.000000         441.000000
```

```
mean     6227.272109              2.657596          15.335601              3.163265
std      4361.727195              2.453799           3.719827              0.370027
min      1081.000000              0.000000          11.000000              3.000000
25%      2966.000000              1.000000          12.000000              3.000000
50%      5033.000000              2.000000          14.000000              3.000000
75%      7644.000000              4.000000          18.000000              3.000000
max     19833.000000              9.000000          25.000000              4.000000

       TotalWorkingYears  YearsAtCompany
count         441.000000      441.000000
mean           10.965986        6.938776
std             7.087199        5.730569
min             0.000000        0.000000
25%             6.000000        3.000000
50%            10.000000        5.000000
75%            14.000000       10.000000
max            37.000000       36.000000
```

[18]:

```python
df['Age'].describe()
```

[18]:

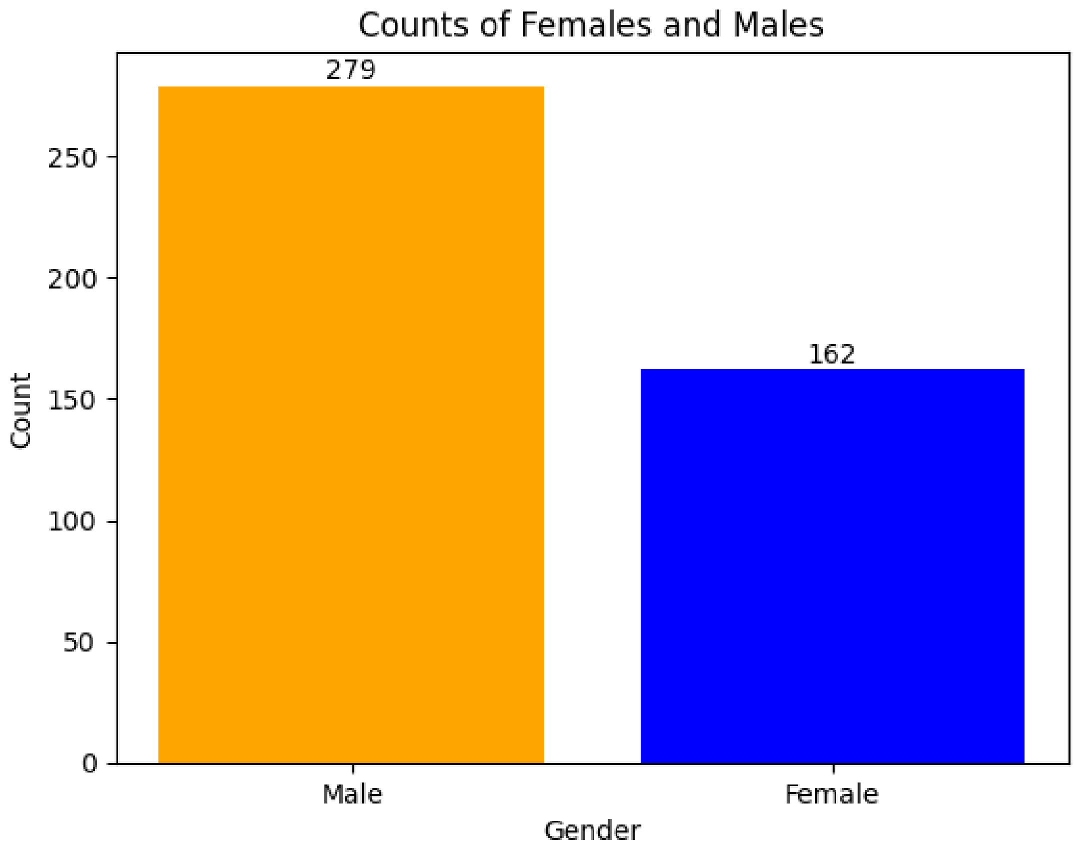
```
count    441.000000
mean      36.696145
std        8.415797
min       18.000000
25%       31.000000
50%       36.000000
75%       42.000000
max       60.000000
Name: Age, dtype: float64
```

Data Visualization

Visualizing Counts of Gender: With A Bar Graph

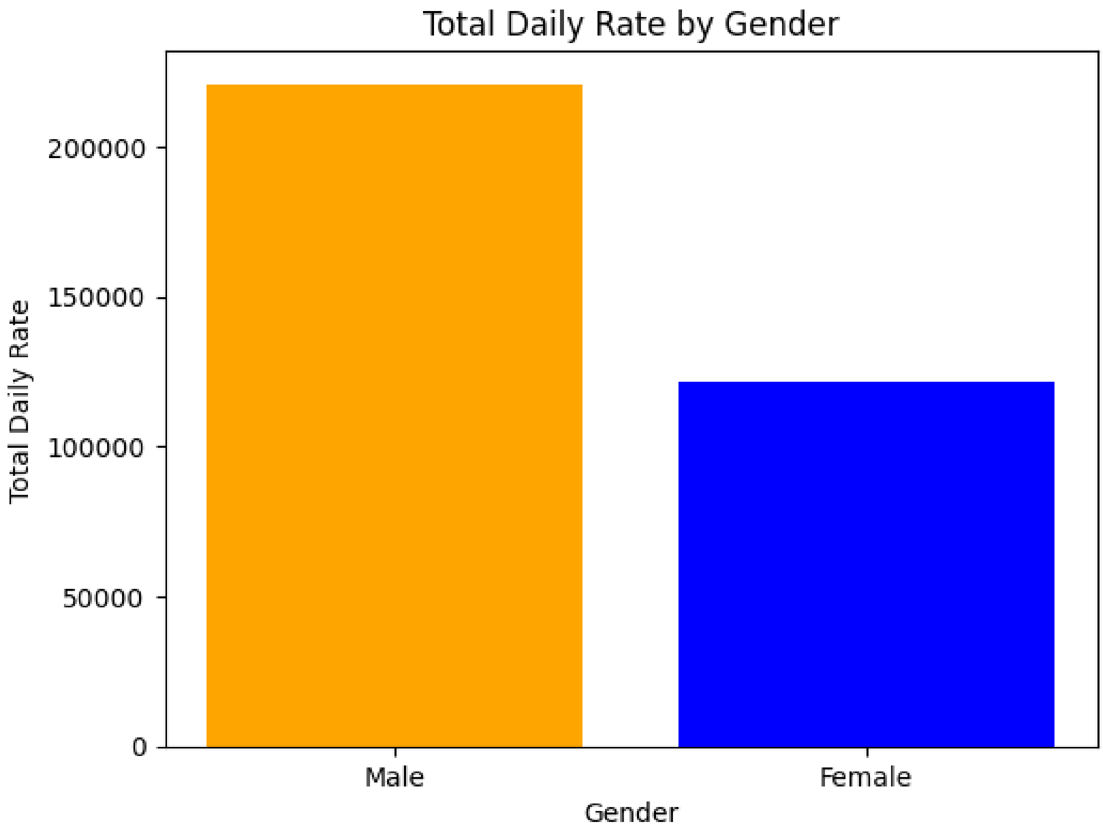
[44]:

```python
gender_counts = df['Gender'].value_counts()
num_females = gender_counts['Female']
num_males = gender_counts['Male']
colors = ['orange', 'blue']
plt.bar(['Male', 'Female'], [num_males,num_females], color=colors)
values = [num_males, num_females]

for index, v in enumerate(values):
    plt.text(index, v + 0.5, str(v), ha='center', va='bottom')

plt.xlabel('Gender')
plt.ylabel('Count')
plt.title('Counts of Females and Males')
plt.show()
```



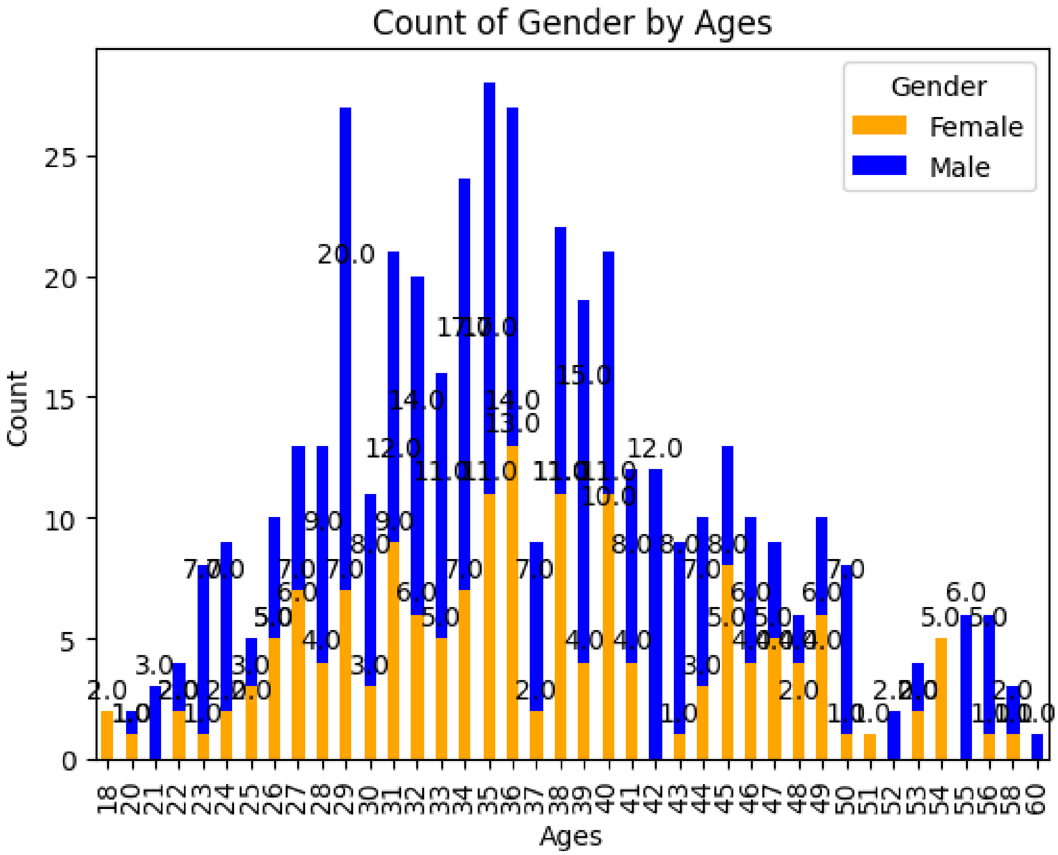
Visualizing Gender and Daily Rate: With A Bar Graph

[20]:
```
sales_gen = df.groupby(['Gender'], as_index=False)['DailyRate'].sum().
  ↪sort_values(by='DailyRate', ascending=False)
plt.bar(sales_gen['Gender'], sales_gen['DailyRate'], color=['orange', 'blue'])
plt.xlabel('Gender')
plt.ylabel('Total Daily Rate')
plt.title('Total Daily Rate by Gender')
plt.show()
```



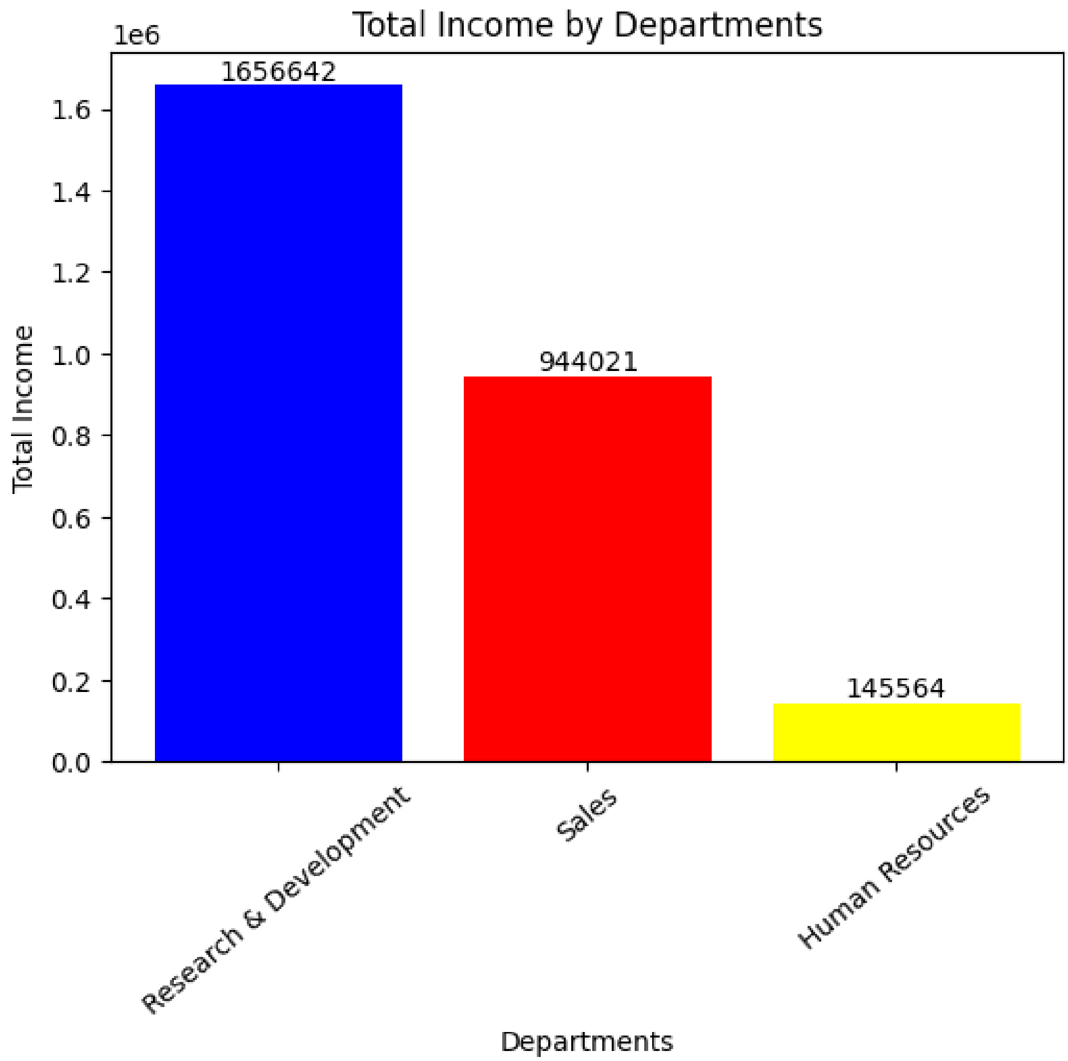
Visualizing Gender and Age Group: With A Bar Graph

[21]:
```
grouped_data = df.groupby(['Age', 'Gender']).size().unstack()
ax = grouped_data.plot(kind='bar', stacked=True, color=['orange', 'blue'])

for container in ax.containers:
    for bar in container:
        height = bar.get_height()
        if height > 0:
            ax.annotate('{}'.format(height),
                        xy=(bar.get_x() + bar.get_width() / 2, height),
                        xytext=(0, 3),  # 3 points vertical offset
                        textcoords="offset points",
                        ha='center', va='bottom')

plt.xlabel('Ages')
plt.ylabel('Count')
plt.title('Count of Gender by Ages')
plt.show()
```



Visualizing Department and Income: With A Bar Graph

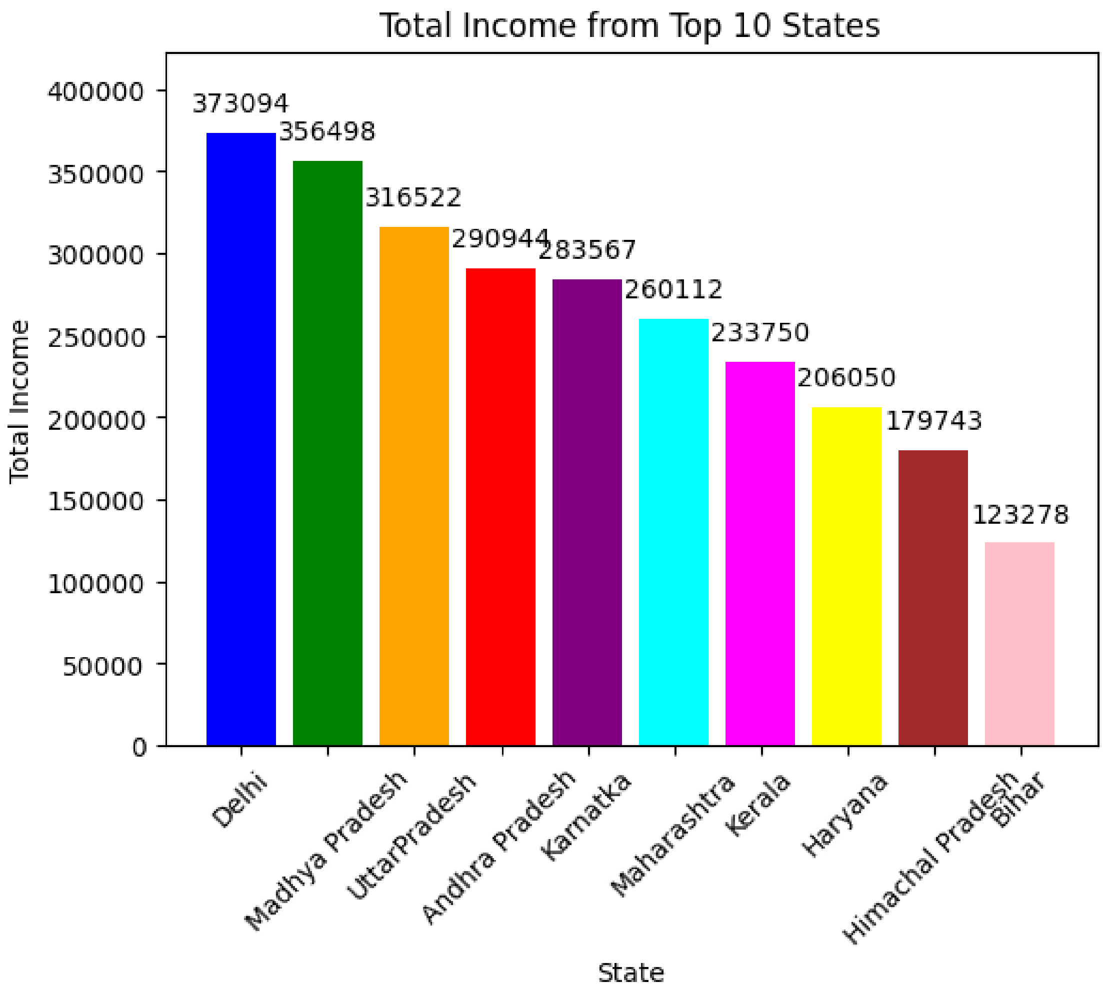
[22]:
```
sales_age = df.groupby(['Department'], as_index=False)['Income'].sum().
 ↪sort_values(by='Income', ascending=False)
plt.bar(sales_age['Department'], sales_age['Income'],␣
 ↪color=['blue','red','yellow'])

for i, v in enumerate(sales_age['Income']):
    plt.text(i, v + 1000, str(v), ha='center', va='bottom')

plt.xlabel('Departments')
plt.ylabel('Total Income')
plt.title('Total Income by Departments')
plt.xticks(rotation=40)
plt.show()
```



Visualizing States and Income: With A Bar Graph

[23]:
```
sales_state = df.groupby(['State'], as_index=False)['Income'].sum().
 ↪sort_values(by='Income', ascending=False).head(10)
colors = ['blue', 'green', 'orange', 'red', 'purple', 'cyan', 'magenta',␣
 ↪'yellow', 'brown', 'pink']
plt.bar(sales_state['State'], sales_state['Income'], color=colors)

for i, v in enumerate(sales_state['Income']):
    plt.text(i, v + 10000, str(v), ha='center', va='bottom')

plt.xlabel('State')
plt.ylabel('Total Income')
plt.title('Total Income from Top 10 States')
```

```
plt.xticks(rotation=45)
plt.ylim(0, max(sales_state['Income']) + 50000)
plt.figure(figsize=(15, 5))
plt.show()
```

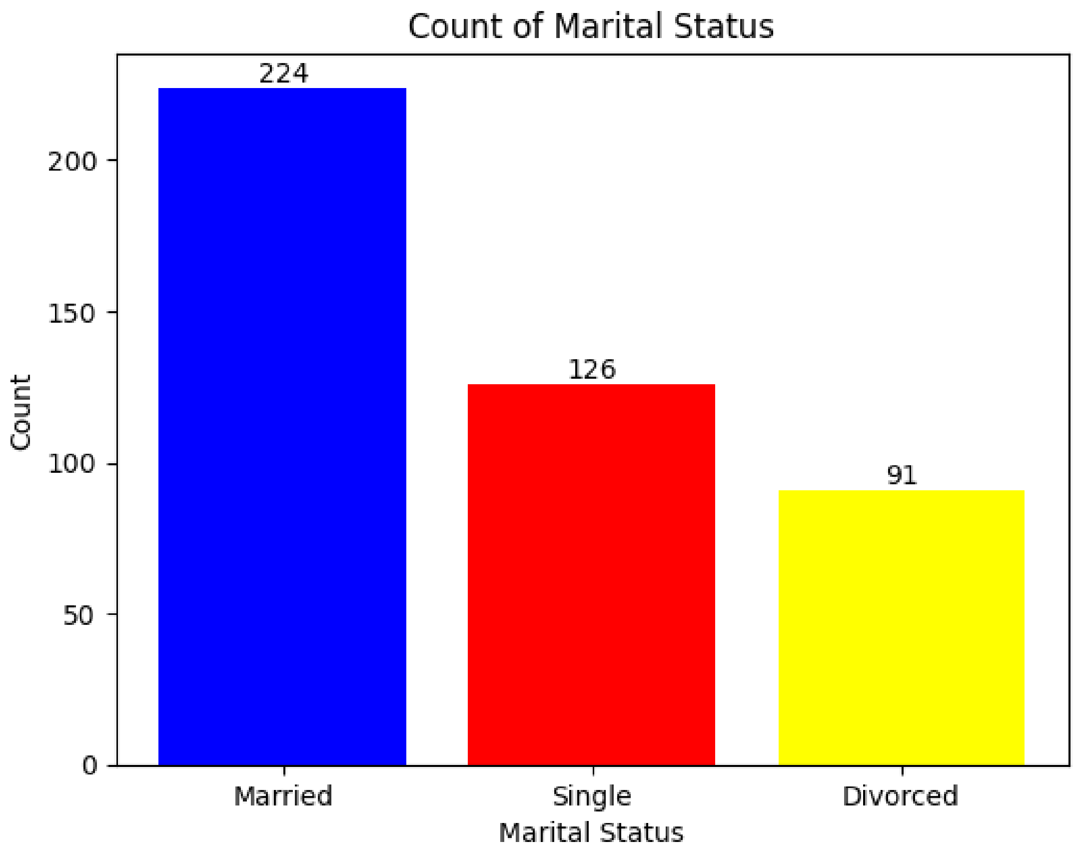



```
<Figure size 1500x500 with 0 Axes>
```

Visualizing Marital Status: With A Bar Graph

[24]:
```
marital_counts = df['MaritalStatus'].value_counts()
plt.bar(marital_counts.index, marital_counts.values,␣
 ↪color=['blue','red','yellow'])

for i, v in enumerate(marital_counts.values):
    plt.text(i, v, str(v), ha='center', va='bottom')

plt.xlabel('Marital Status')
```

```
plt.ylabel('Count')
plt.title('Count of Marital Status')
plt.figure(figsize=(7, 5))
plt.show()
```




```
<Figure size 700x500 with 0 Axes>
```

Visualizing Gender, Marital Status and Income: With A Bar Graph

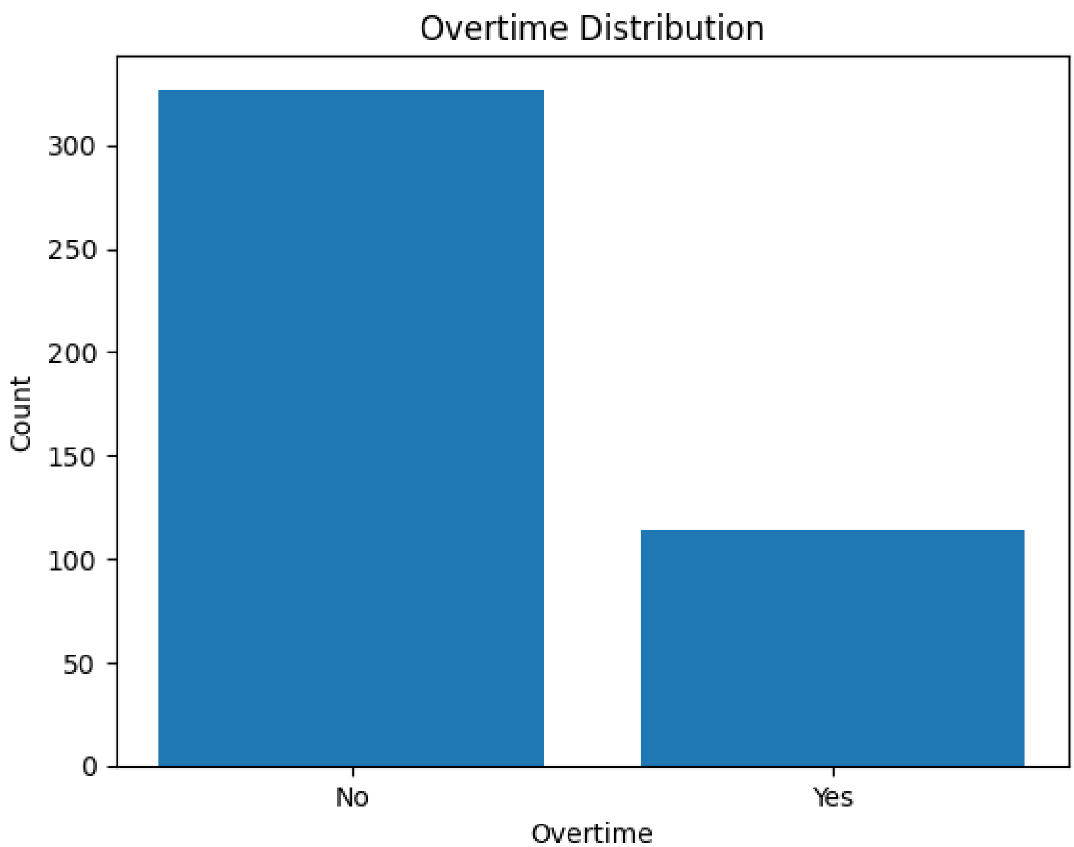
[25]:
```
income_data = df.groupby(['MaritalStatus', 'Gender'])['Income'].sum().
↪reset_index()

marital_status_categories = df['MaritalStatus'].unique()
x_positions = range(len(marital_status_categories))

gender_categories = df['Gender'].unique()
colors = ['orange', 'blue']

bar_width = 0.4
```

```
for i, gender in enumerate(gender_categories):
    gender_data = income_data[income_data['Gender'] == gender]
    plt.bar([pos + i * bar_width for pos in x_positions],␣
 ↪gender_data['Income'], width=bar_width, label=gender, color=colors[i])

plt.xlabel('Marital Status')
plt.ylabel('Total Income')
plt.title('Total Income by Marital Status and Gender')
plt.xticks([pos + bar_width / 2 for pos in x_positions],␣
 ↪marital_status_categories)
plt.legend(title='Gender', title_fontsize=10)
plt.tight_layout()
plt.show()
```

Visualizing Overtiming of employees: With A Bar Graph

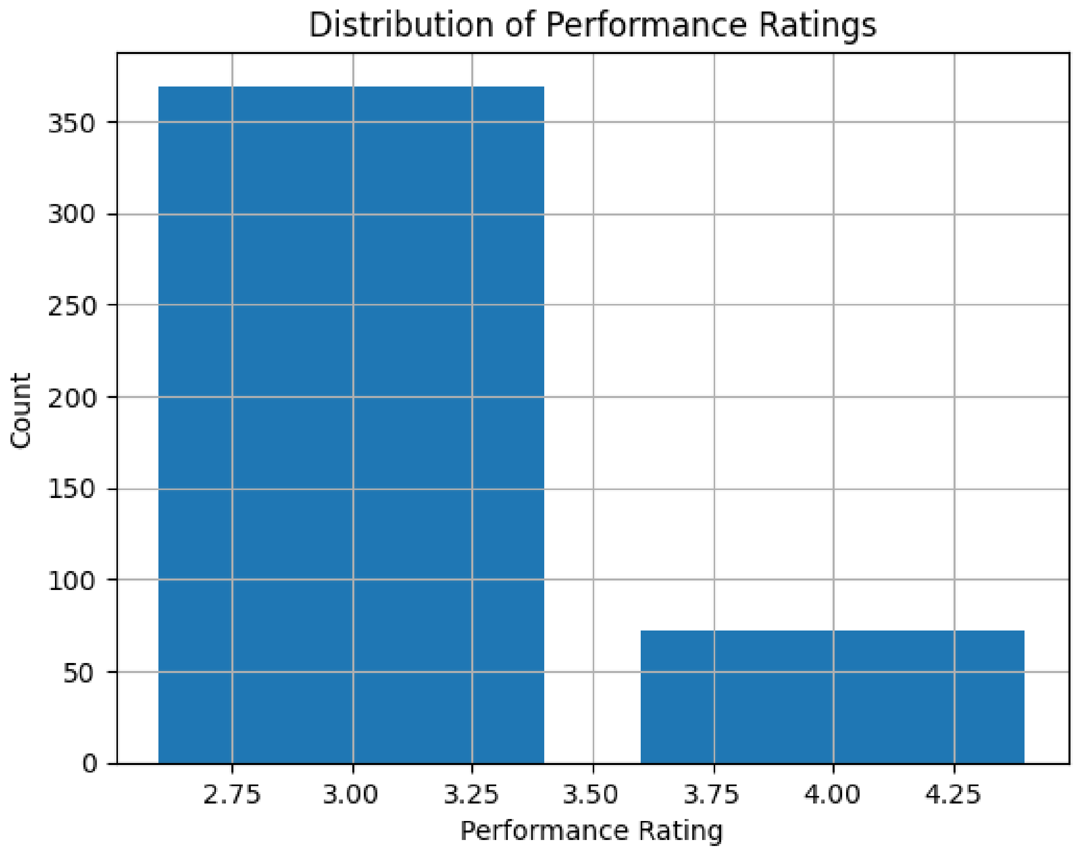
[26]:
```
overtime_counts = df['OverTime'].value_counts()
categories = overtime_counts.index
counts = overtime_counts.values
plt.bar(categories, counts)
plt.xlabel('Overtime')
```

```
plt.ylabel('Count')
plt.title('Overtime Distribution')
plt.show()
```



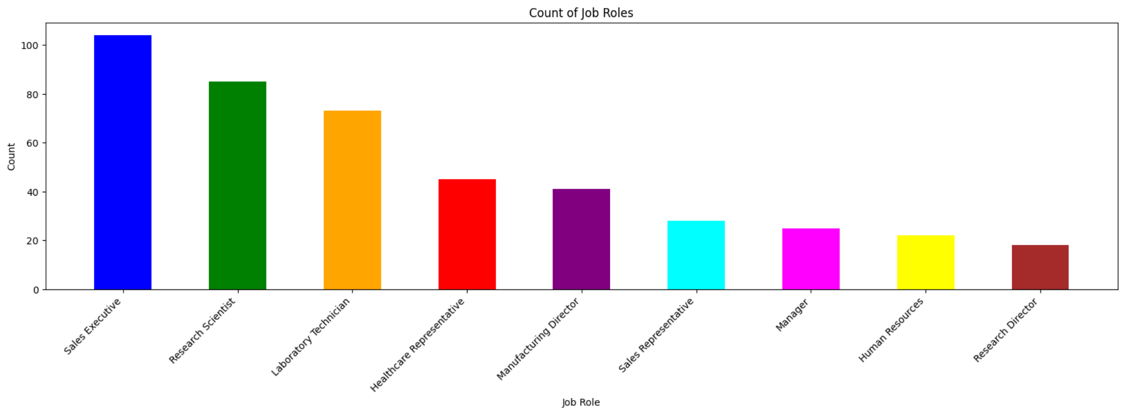
Visualizing Performance Rating: With A Bar Graph

[27]:
```
performance_ratings = df['PerformanceRating']
rating_counts = {}
for rating in performance_ratings:
    rating_counts[rating] = rating_counts.get(rating, 0) + 1

ratings = list(rating_counts.keys())
counts = list(rating_counts.values())
plt.bar(ratings, counts)
plt.xlabel('Performance Rating')
plt.ylabel('Count')
plt.title('Distribution of Performance Ratings')
plt.grid(True)
plt.show()
```



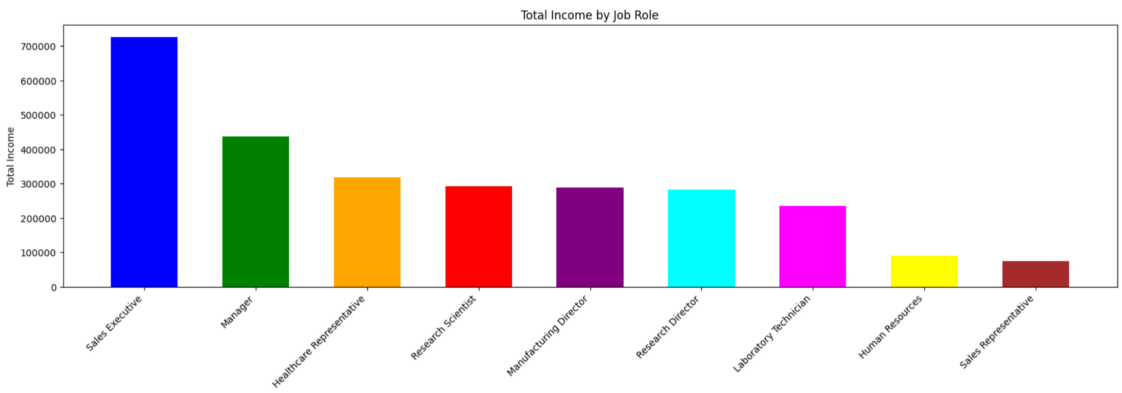
Visualizing Job Role's: With A Bar Graph

[28]:
```
job_role_counts = df['JobRole'].value_counts()
plt.figure(figsize=(20, 5))
colors = ['blue', 'green', 'orange', 'red', 'purple', 'cyan', 'magenta',␣
 ↪'yellow', 'brown', 'pink']
bar_width = 0.5
bar_positions = range(len(job_role_counts))
plt.bar(bar_positions, job_role_counts, width=bar_width, color=colors)
plt.xlabel('Job Role')
plt.ylabel('Count')
plt.title('Count of Job Roles')
plt.xticks(bar_positions, job_role_counts.index, rotation=45, ha='right')
plt.show()
```



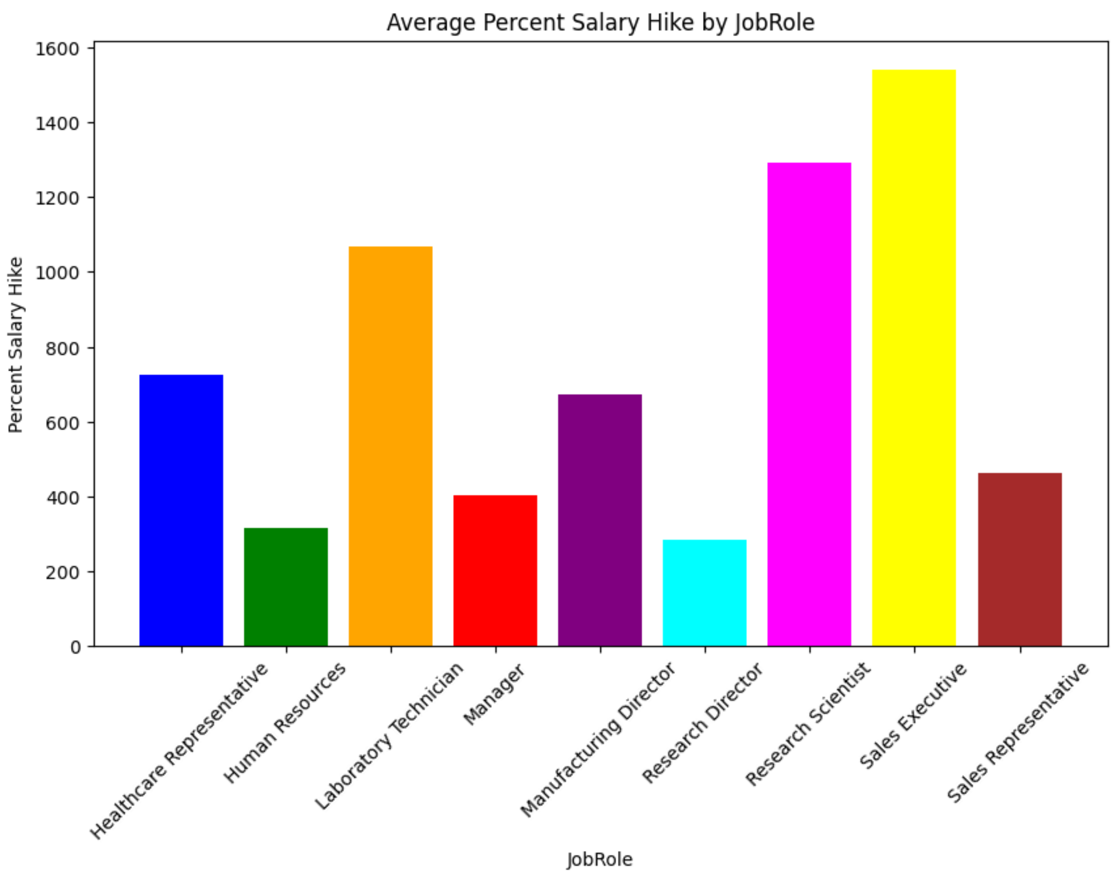
Visualizing Income and Job Role's: With A Bar Graph

[29]:
```
sales_state = df.groupby(['JobRole'], as_index=False)['Income'].sum().
    ↪sort_values(by='Income', ascending=False)
plt.figure(figsize=(20, 5))
colors = ['blue', 'green', 'orange', 'red', 'purple', 'cyan', 'magenta',␣
    ↪'yellow', 'brown', 'pink']
bar_width = 0.6
bar_positions = range(len(sales_state))
plt.bar(bar_positions, sales_state['Income'], width=bar_width, color=colors)
plt.xlabel('Job Role')
plt.ylabel('Total Income')
plt.title('Total Income by Job Role')
plt.xticks(bar_positions, sales_state['JobRole'], rotation=45, ha='right')
plt.show()
```



Visualizing Salary Hike Percentage and Job Role's: With A Bar Graph

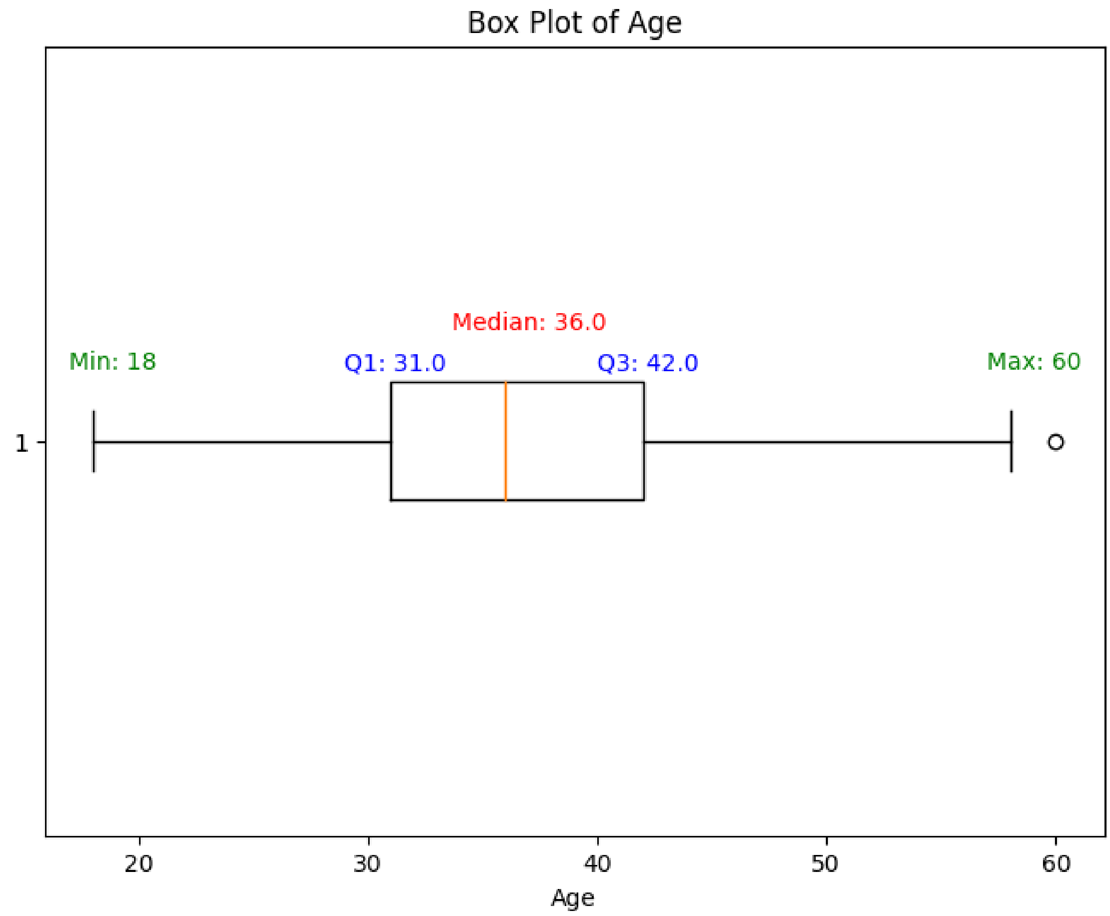
[30]:
```
department_percent_salary_hike = df.groupby(['JobRole'],␣
↪as_index=False)['PercentSalaryHike'].sum()
plt.figure(figsize=(10, 6))
colors = ['blue', 'green', 'orange', 'red', 'purple', 'cyan', 'magenta',␣
↪'yellow', 'brown', 'pink']
plt.bar(department_percent_salary_hike['JobRole'],␣
↪department_percent_salary_hike['PercentSalaryHike'], color=colors)
plt.xlabel('JobRole')
plt.ylabel('Percent Salary Hike')
plt.title('Average Percent Salary Hike by JobRole')
plt.xticks(rotation=45)
plt.show()
```



Visualizing Ages: With A Box Plot

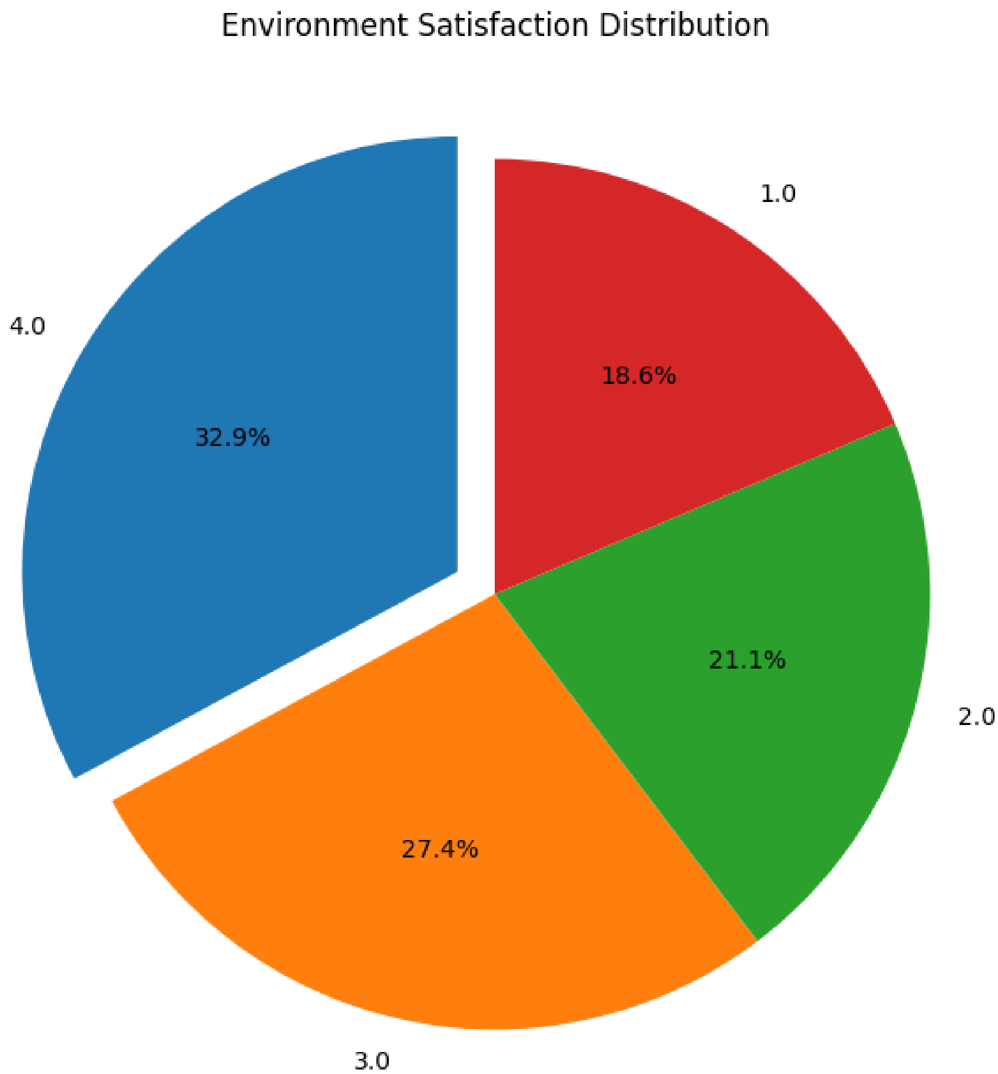
[31]:
```
q1 = df['Age'].quantile(0.25)
q3 = df['Age'].quantile(0.75)
median = df['Age'].median()
minimum = df['Age'].min()
maximum = df['Age'].max()
```

```python
plt.figure(figsize=(8, 6))
plt.boxplot(df['Age'], vert=False)
plt.xlabel('Age')
plt.title('Box Plot of Age')
plt.text(q1 -2, 1.1, f'Q1: {q1}', verticalalignment='center', color='blue')
plt.text(q3 -2, 1.1, f'Q3: {q3}', verticalalignment='center', color='blue')
plt.text(median - 2.3, 1.15, f'Median: {median}', verticalalignment='center',␣
    ↪color='red')
plt.text(maximum -3, 1.1, f'Max: {maximum}', verticalalignment='center',␣
    ↪color='green')
plt.text(minimum -1 , 1.1, f'Min: {minimum}', verticalalignment='center',␣
    ↪color='green')
plt.show()
```



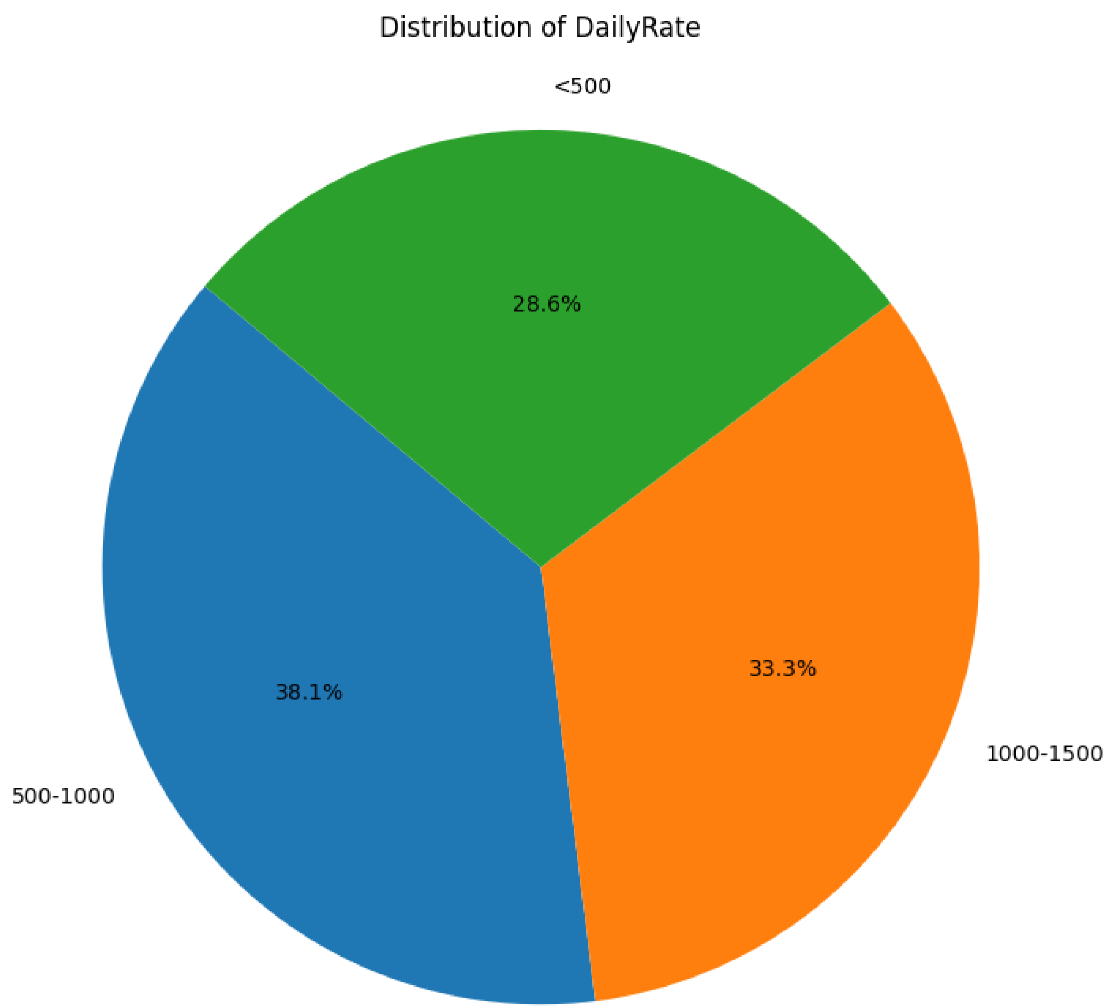
Visualizing Environment Satisfaction: With A Pie Chart

[37]:
```
EnvironmentSatisfaction = df['EnvironmentSatisfaction'].value_counts()
age_counts = df['Age'].value_counts()
myexp = [0.1, 0, 0, 0]
plt.figure(figsize=(8, 8))
plt.pie(EnvironmentSatisfaction.values, labels=EnvironmentSatisfaction.index,␣
 ↪autopct='%1.1f%%', startangle=90, explode=myexp)
plt.title('Environment Satisfaction Distribution')
plt.show()
```



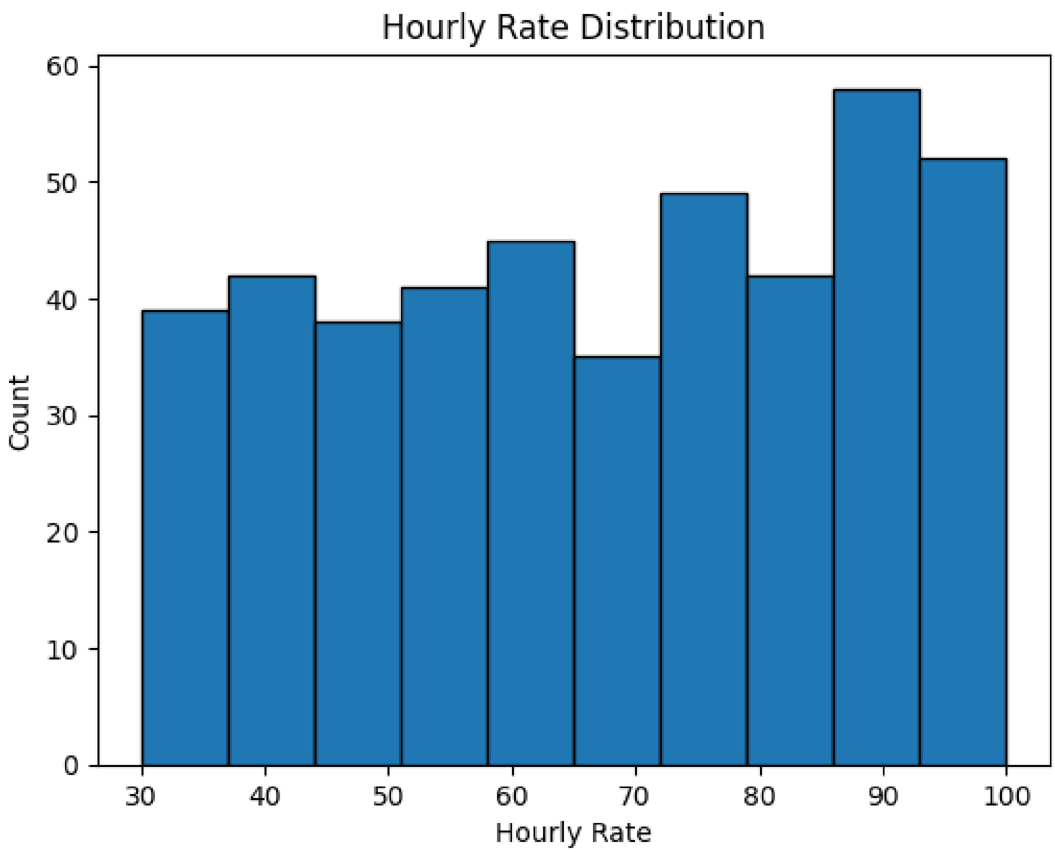
Visualizing Daily Rate: With A Pie Chart

[38]:
```
daily_rate_data = df['DailyRate']
bins = [100, 500, 1000, 1500]
daily_rate_categories = pd.cut(daily_rate_data, bins=bins, labels=['<500',␣
↪'500-1000', '1000-1500'])
category_counts = daily_rate_categories.value_counts()
plt.figure(figsize=(8, 8))
plt.pie(category_counts, labels=category_counts.index, autopct='%1.1f%%',␣
↪startangle=140)
plt.title('Distribution of DailyRate',pad=22)
plt.axis('equal')
plt.show()
```



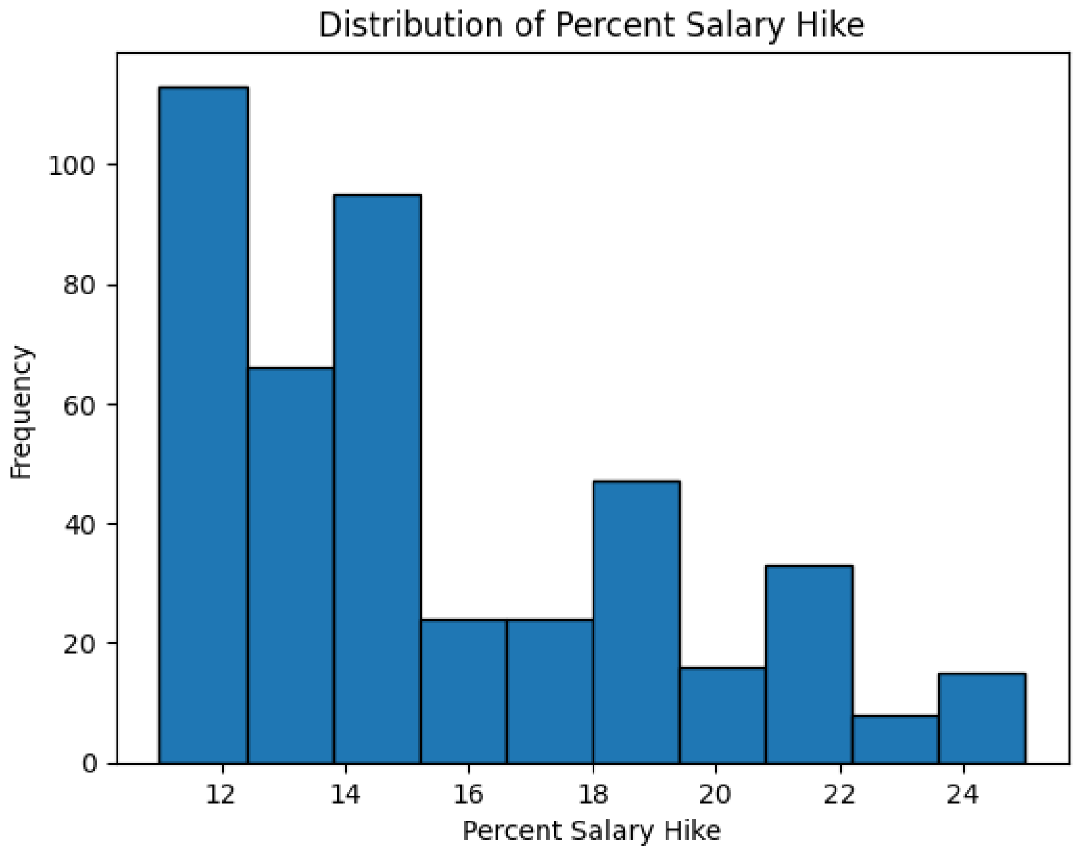
Visualizing Hourly Rate: With A Histogram

[39]:
```
hourly_rate_data = df['HourlyRate']
num_bins = 10
plt.hist(hourly_rate_data, bins=num_bins, histtype='bar', edgecolor='black')
plt.xlabel('Hourly Rate')
plt.ylabel('Count')
plt.title('Hourly Rate Distribution')
plt.show()
```



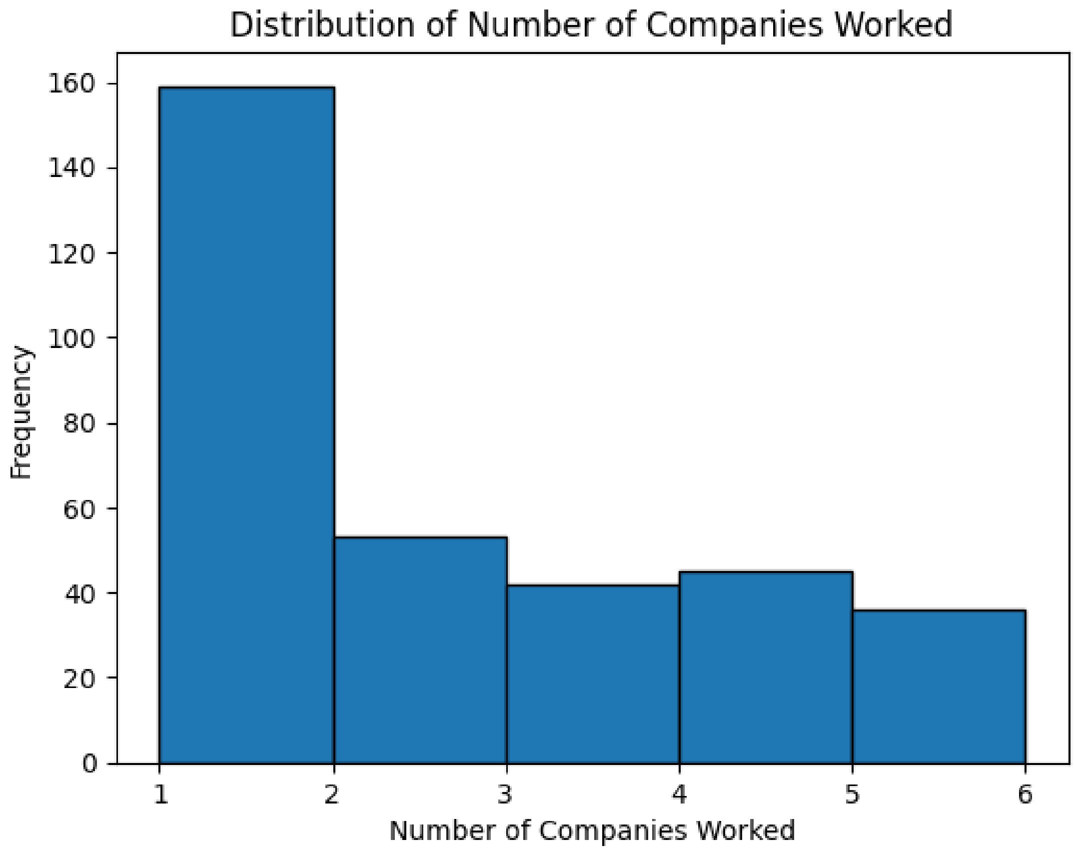
Visualizing Percent of Salary Hike: With A Histogram

[40]:
```
percent_salary_hike_data = df['PercentSalaryHike']
plt.hist(percent_salary_hike_data, bins=10, edgecolor='black')
plt.xlabel('Percent Salary Hike')
plt.ylabel('Frequency')
plt.title('Distribution of Percent Salary Hike')
plt.show()
```



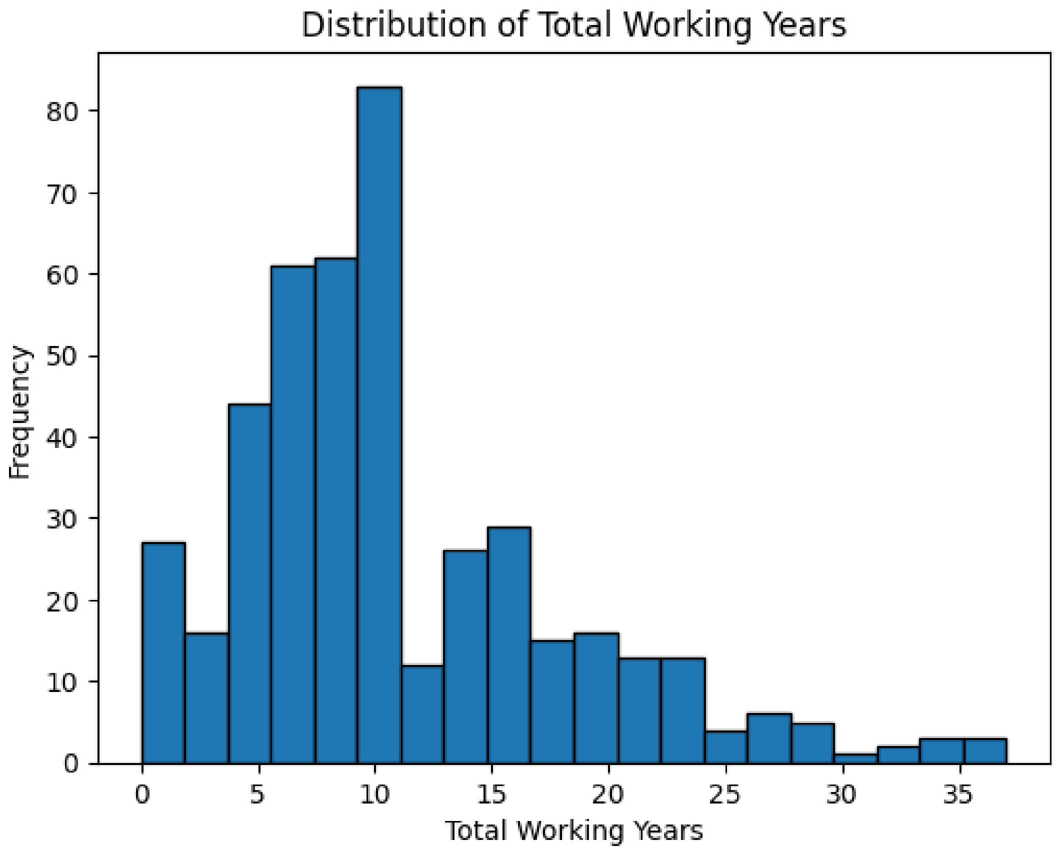
Visualizing Number of companies worked in: With A Histogram

[41]:
```
num_companies_worked = df['NumCompaniesWorked']
plt.hist(num_companies_worked, bins=range(1, 7), edgecolor='black')
plt.xlabel('Number of Companies Worked')
plt.ylabel('Frequency')
plt.title('Distribution of Number of Companies Worked')
plt.show()
```



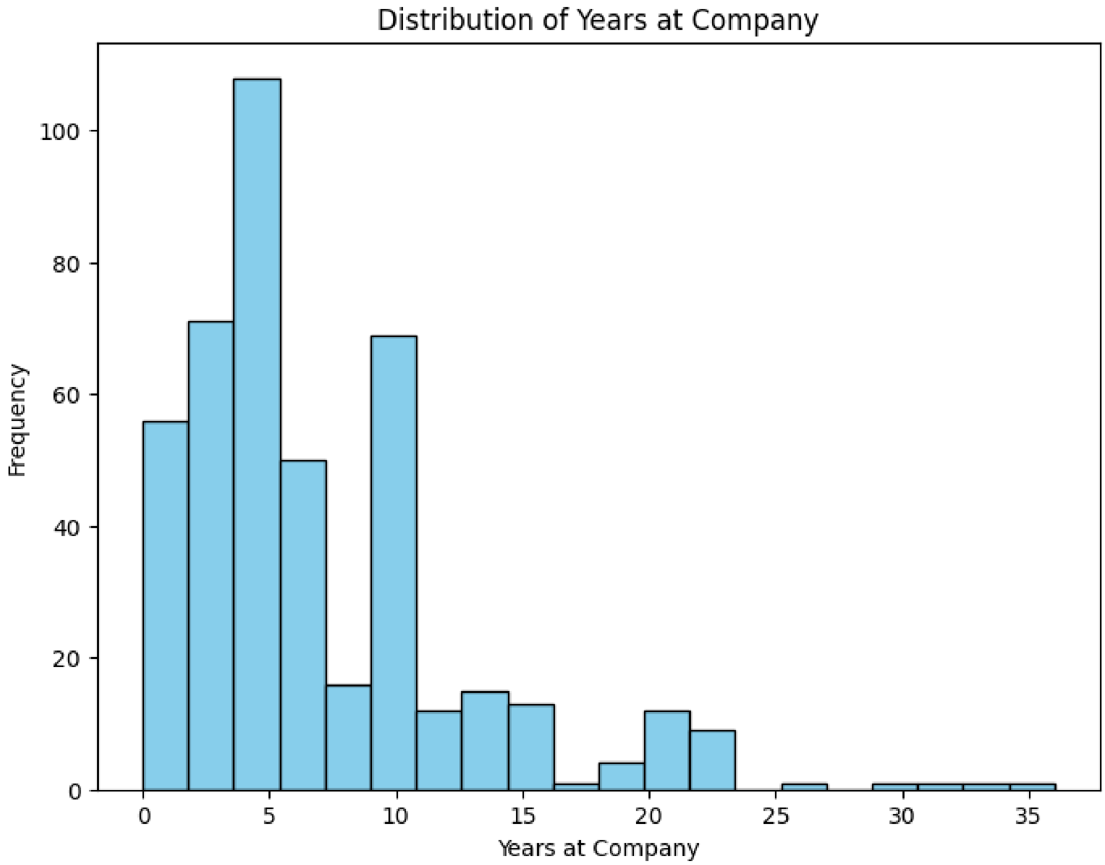
Visualizing Total Working Years: With A Histogram

[42]:
```python
total_working_years = df['TotalWorkingYears']
plt.hist(total_working_years, bins=20, edgecolor='black')
plt.xlabel('Total Working Years')
plt.ylabel('Frequency')
plt.title('Distribution of Total Working Years')
plt.show()
```



Visualizing Years At Company: With A Histogram

[43]:
```
years_at_company = df['YearsAtCompany']
plt.figure(figsize=(8, 6))
plt.hist(years_at_company, bins=20, color='skyblue', edgecolor='black')
plt.xlabel('Years at Company')
plt.ylabel('Frequency')
plt.title('Distribution of Years at Company')
plt.show()
```



Conclusion

In conclusion, this project explored a dataset containing various employee-related attributes, including age, job role, satisfaction levels, and more. Through data analysis and visualization, several valuable insights were uncovered. Factors such as job satisfaction, overtime hours, and distance from home emerged as critical contributors to employee attrition. Additionally, employee satisfaction seemed to be influenced significantly by environmental factors. These findings provide valuable guidance and offers actionable recommendations to create a more conducive and fulfilling work environment for employees.

by:- Aditya Acharya

```
[ ]:
```