Introduction:

This task involves predicting employee turnover using a given dataset. The dataset contains various features related to employees, such as age, job level, job satisfaction, salary, and more. The goal is to build predictive models that can accurately classify whether an employee is likely to leave the company (attrition) or not. The task is important for HR departments to identify potential turnover risks and take preventive measures.

Business Questions:

Can we predict whether an employee is likely to leave the company based on various features? Which features are most strongly correlated with employee attrition? What are the key factors contributing to employee turnover?

Importing Libraries

In [61]:
```
# Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import hvplot.pandas
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_

import warnings
warnings.filterwarnings('ignore', category=FutureWarning)
```

Loading Dataset

In [62]:
```
data = pd.read_csv(f"C:/Users/zaid arman/Desktop/New folder/TechnoHackes Inter
```

In [63]:
```
data
```

Out[63]:

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

1470 rows × 35 columns

Data Exploration

In [64]:
```
data.head()
```

Out[64]:

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

5 rows × 35 columns

In [65]:
```
data.tail()
```

Out[65]:

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

5 rows × 35 columns

In [66]:
```
data.shape
```

Out[66]:
```
(1470, 35)
```

In [67]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          1470 non-null   int64
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

In [68]:
```
data.describe()
```

Out[68]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNu
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.0(
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.8(
std	9.135373	403.509100	8.106864	1.024165	0.0	602.0(
min	18.000000	102.000000	1.000000	1.000000	1.0	1.0(
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.2(
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.5(
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.7(
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.0(

8 rows × 26 columns

In [69]:
```
data.describe(include = 'all')
```

Out[69]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	
count	1470.000000	1470	1470	1470.000000	1470	1470.000000	1(
unique	NaN	2	3	NaN	3	NaN	
top	NaN	No	Travel_Rarely	NaN	Research & Development	NaN	
freq	NaN	1233	1043	NaN	961	NaN	
mean	36.923810	NaN	NaN	802.485714	NaN	9.192517	
std	9.135373	NaN	NaN	403.509100	NaN	8.106864	
min	18.000000	NaN	NaN	102.000000	NaN	1.000000	
25%	30.000000	NaN	NaN	465.000000	NaN	2.000000	
50%	36.000000	NaN	NaN	802.000000	NaN	7.000000	
75%	43.000000	NaN	NaN	1157.000000	NaN	14.000000	
max	60.000000	NaN	NaN	1499.000000	NaN	29.000000	

11 rows × 35 columns

In [70]:
```
data.describe
```

Out[70]:
```
<bound method NDFrame.describe of       Age Attrition     BusinessTravel  Da
ilyRate              Department  \
0      41       Yes      Travel_Rarely       1102                   Sales
1      49        No  Travel_Frequently        279  Research & Development
2      37       Yes      Travel_Rarely       1373  Research & Development
3      33        No  Travel_Frequently       1392  Research & Development
4      27        No      Travel_Rarely        591  Research & Development
...   ...       ...                 ...        ...                     ...
1465   36        No  Travel_Frequently        884  Research & Development
1466   39        No      Travel_Rarely        613  Research & Development
1467   27        No      Travel_Rarely        155  Research & Development
1468   49        No  Travel_Frequently       1023                   Sales
1469   34        No      Travel_Rarely        628  Research & Development

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

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

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

     WorkLifeBalance  YearsAtCompany YearsInCurrentRole  \
0                  1               6                  4
1                  3              10                  7
2                  3               0                  0
```

```
3                    3                8                   7
4                    3                2                   2
...                ...              ...                 ...
1465                 3                5                   2
1466                 3                7                   7
1467                 3                6                   2
1468                 2                9                   6
1469                 4                4                   3

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

[1470 rows x 35 columns]>
```

In [71]:
```
data.dtypes
```

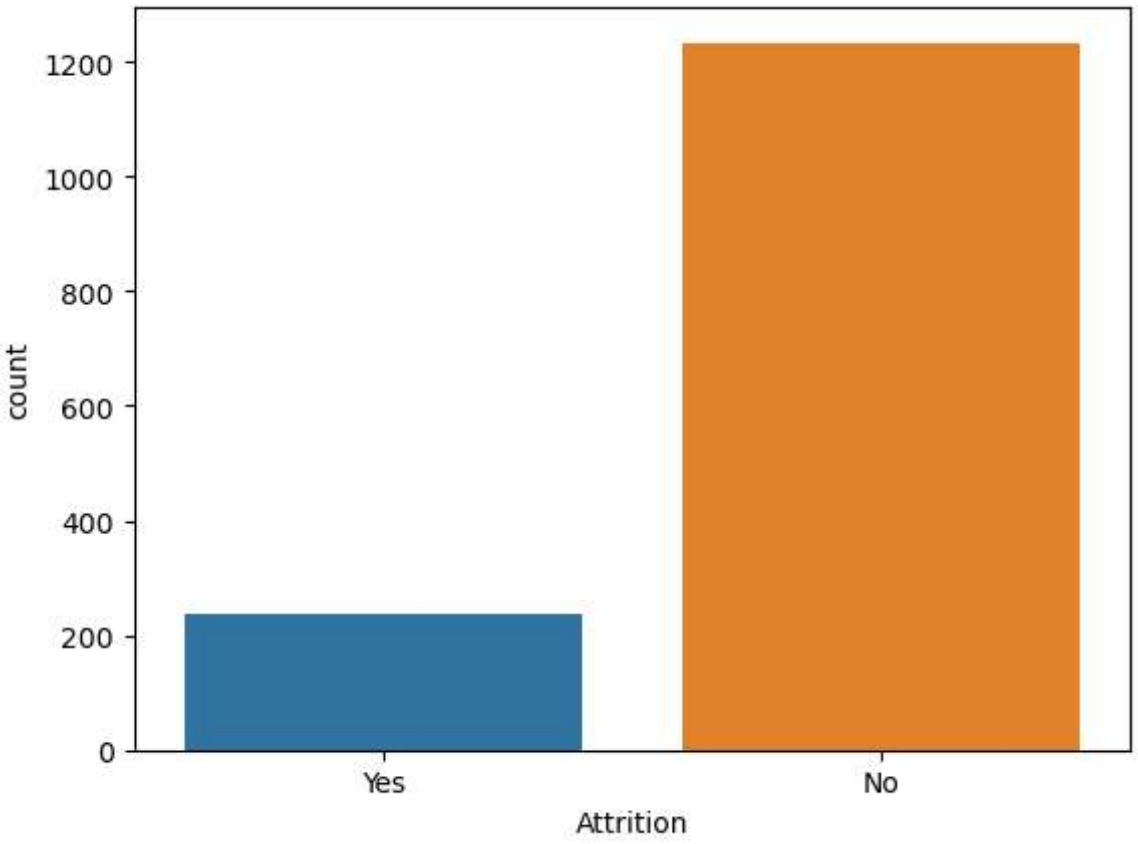
Out[71]:
```
Age                         int64
Attrition                  object
BusinessTravel             object
DailyRate                   int64
Department                 object
DistanceFromHome            int64
Education                   int64
EducationField             object
EmployeeCount               int64
EmployeeNumber              int64
EnvironmentSatisfaction     int64
Gender                     object
HourlyRate                  int64
JobInvolvement              int64
JobLevel                    int64
JobRole                    object
JobSatisfaction             int64
MaritalStatus              object
MonthlyIncome               int64
MonthlyRate                 int64
NumCompaniesWorked          int64
Over18                     object
OverTime                   object
PercentSalaryHike           int64
PerformanceRating           int64
RelationshipSatisfaction    int64
StandardHours               int64
StockOptionLevel            int64
TotalWorkingYears           int64
TrainingTimesLastYear       int64
WorkLifeBalance             int64
YearsAtCompany              int64
YearsInCurrentRole          int64
YearsSinceLastPromotion     int64
YearsWithCurrManager        int64
dtype: object
```

In [72]:
```
data['Attrition'].value_counts()
```

Out[72]:
```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

In [73]:
```
sns.countplot(data=data, x='Attrition')
```

Out[73]: <Axes: xlabel='Attrition', ylabel='count'>



Here, the distribution is important to consider when building a predictive model. Imbalanced class distributions, where one class significantly outweighs the other, this can impact the model's performance. Here, the "Yes" class (employees who have left) is less common than the "No" class (employees who have not left).

In [74]:
```
for column in data.columns:
    print(f"{column}: Number of unique values {data[column].nunique()}")
    print("=============================================================")
```

```
Age: Number of unique values 43
============================================================
Attrition: Number of unique values 2
============================================================
BusinessTravel: Number of unique values 3
============================================================
DailyRate: Number of unique values 886
============================================================
Department: Number of unique values 3
============================================================
DistanceFromHome: Number of unique values 29
============================================================
Education: Number of unique values 5
============================================================
EducationField: Number of unique values 6
============================================================
EmployeeCount: Number of unique values 1
============================================================
EmployeeNumber: Number of unique values 1470
============================================================
EnvironmentSatisfaction: Number of unique values 4
============================================================
Gender: Number of unique values 2
============================================================
HourlyRate: Number of unique values 71
============================================================
JobInvolvement: Number of unique values 4
============================================================
JobLevel: Number of unique values 5
============================================================
JobRole: Number of unique values 9
============================================================
JobSatisfaction: Number of unique values 4
============================================================
MaritalStatus: Number of unique values 3
============================================================
MonthlyIncome: Number of unique values 1349
============================================================
MonthlyRate: Number of unique values 1427
============================================================
NumCompaniesWorked: Number of unique values 10
============================================================
Over18: Number of unique values 1
============================================================
OverTime: Number of unique values 2
============================================================
PercentSalaryHike: Number of unique values 15
============================================================
PerformanceRating: Number of unique values 2
============================================================
RelationshipSatisfaction: Number of unique values 4
============================================================
StandardHours: Number of unique values 1
============================================================
StockOptionLevel: Number of unique values 4
============================================================
TotalWorkingYears: Number of unique values 40
```

```
============================================================
TrainingTimesLastYear: Number of unique values 7
============================================================
WorkLifeBalance: Number of unique values 4
============================================================
YearsAtCompany: Number of unique values 37
============================================================
YearsInCurrentRole: Number of unique values 19
============================================================
YearsSinceLastPromotion: Number of unique values 16
============================================================
YearsWithCurrManager: Number of unique values 18
============================================================
```

We notice that 'EmployeeCount', 'Over18', 'StandardHours' have only one unique values and 'EmployeeNumber' has 1470 unique values. This features aren't useful for us, So we are going to drop those columns.

In [75]:
```
data.drop(['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours'], axis
```

Categorical Feature

In [76]:
```
object_col = []
for column in data.columns:
    if data[column].dtype == object and len(data[column].unique()) <= 30:
        object_col.append(column)
        print(f"{column} : {data[column].unique()}")
        print(data[column].value_counts())
        print("----------------------------------------------------------")
object_col.remove('Attrition')
```

```
Attrition : ['Yes' 'No']
No     1233
Yes     237
Name: Attrition, dtype: int64
------------------------------------------------------------
BusinessTravel : ['Travel_Rarely' 'Travel_Frequently' 'Non-Travel']
Travel_Rarely        1043
Travel_Frequently     277
Non-Travel            150
Name: BusinessTravel, dtype: int64
------------------------------------------------------------
Department : ['Sales' 'Research & Development' 'Human Resources']
Research & Development    961
Sales                     446
Human Resources            63
Name: Department, dtype: int64
------------------------------------------------------------
EducationField : ['Life Sciences' 'Other' 'Medical' 'Marketing' 'Technical D
egree'
 'Human Resources']
Life Sciences       606
Medical             464
Marketing           159
Technical Degree    132
Other                82
Human Resources      27
Name: EducationField, dtype: int64
------------------------------------------------------------
Gender : ['Female' 'Male']
Male      882
Female    588
Name: Gender, dtype: int64
------------------------------------------------------------
JobRole : ['Sales Executive' 'Research Scientist' 'Laboratory Technician'
 'Manufacturing Director' 'Healthcare Representative' 'Manager'
 'Sales Representative' 'Research Director' 'Human Resources']
Sales Executive              326
Research Scientist           292
Laboratory Technician        259
Manufacturing Director       145
Healthcare Representative    131
Manager                      102
Sales Representative          83
Research Director             80
Human Resources               52
Name: JobRole, dtype: int64
------------------------------------------------------------
MaritalStatus : ['Single' 'Married' 'Divorced']
Married     673
Single      470
Divorced    327
Name: MaritalStatus, dtype: int64
------------------------------------------------------------
OverTime : ['Yes' 'No']
No     1054
Yes     416
```

```
Name: OverTime, dtype: int64
------------------------------------------------------------
```

In [77]:
```
len(object_col)
```

Out[77]:
```
7
```

In [78]:
```
label = LabelEncoder()
data["Attrition"] = label.fit_transform(data.Attrition)
```

Numerical Featurs

In [79]:
```
disc_col = []
for column in data.columns:
    if data[column].dtypes != object and data[column].nunique() < 30:
        print(f"{column} : {data[column].unique()}")
        disc_col.append(column)
        print("----------------------------------------------------")
disc_col.remove('Attrition')
```

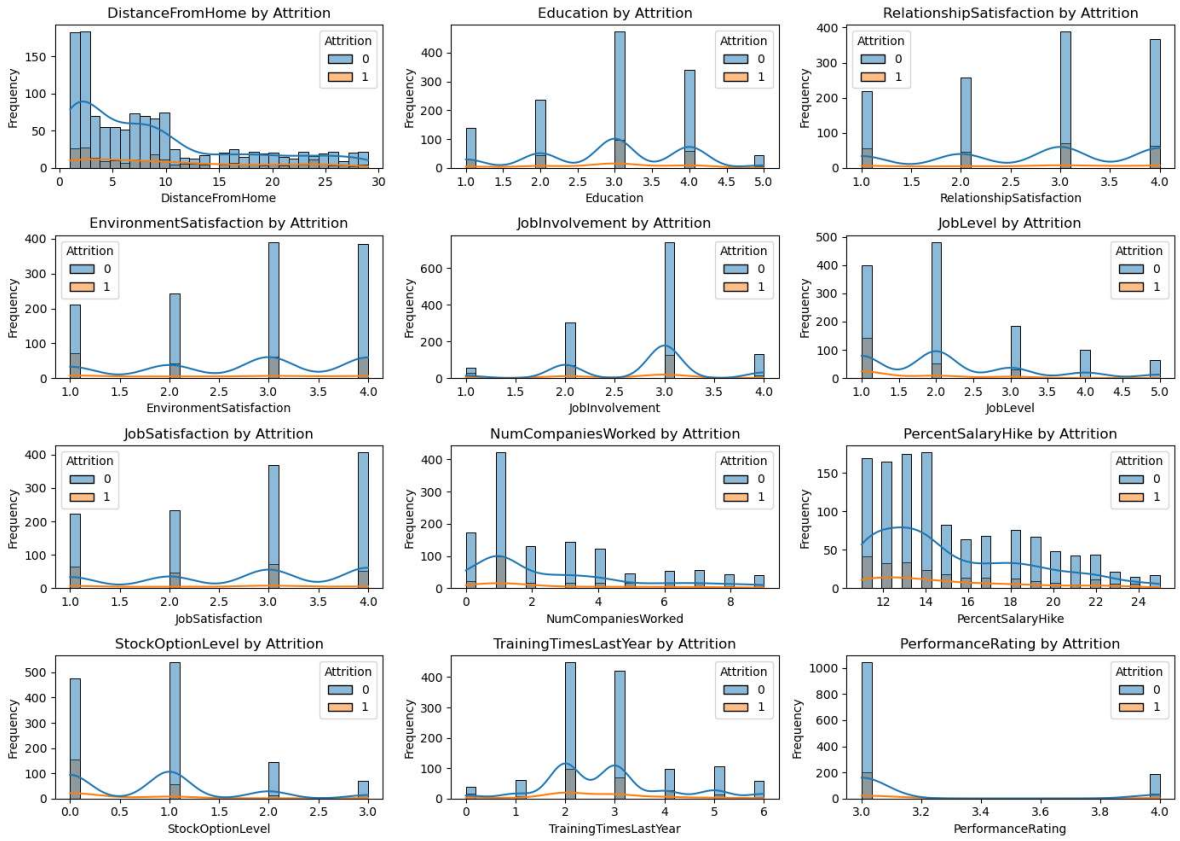
```
Attrition : [1 0]
----------------------------------------------------
DistanceFromHome : [ 1  8  2  3 24 23 27 16 15 26 19 21  5 11  9  7  6 10  4
25 12 18 29 22
 14 20 28 17 13]
----------------------------------------------------
Education : [2 1 4 3 5]
----------------------------------------------------
EnvironmentSatisfaction : [2 3 4 1]
----------------------------------------------------
JobInvolvement : [3 2 4 1]
----------------------------------------------------
JobLevel : [2 1 3 4 5]
----------------------------------------------------
JobSatisfaction : [4 2 3 1]
----------------------------------------------------
NumCompaniesWorked : [8 1 6 9 0 4 5 2 7 3]
----------------------------------------------------
PercentSalaryHike : [11 23 15 12 13 20 22 21 17 14 16 18 19 24 25]
----------------------------------------------------
PerformanceRating : [3 4]
----------------------------------------------------
RelationshipSatisfaction : [1 4 2 3]
----------------------------------------------------
StockOptionLevel : [0 1 3 2]
----------------------------------------------------
TrainingTimesLastYear : [0 3 2 5 1 4 6]
----------------------------------------------------
WorkLifeBalance : [1 3 2 4]
----------------------------------------------------
YearsInCurrentRole : [ 4  7  0  2  5  9  8  3  6 13  1 15 14 16 11 10 12 18
17]
----------------------------------------------------
YearsSinceLastPromotion : [ 0  1  3  2  7  4  8  6  5 15  9 13 12 10 11 14]
----------------------------------------------------
YearsWithCurrManager : [ 5  7  0  2  6  8  3 11 17  1  4 12  9 10 15 13 16 1
4]
----------------------------------------------------
```

In [80]:
```
cont_col = []
for column in data.columns:
    if data[column].dtypes != object and data[column].nunique() > 30:
        print(f"{column} : Minimum: {data[column].min()}, Maximum: {data[colu
        cont_col.append(column)
        print("----------------------------------------")
```

```
Age : Minimum: 18, Maximum: 60
----------------------------------------
DailyRate : Minimum: 102, Maximum: 1499
----------------------------------------
HourlyRate : Minimum: 30, Maximum: 100
----------------------------------------
MonthlyIncome : Minimum: 1009, Maximum: 19999
----------------------------------------
MonthlyRate : Minimum: 2094, Maximum: 26999
----------------------------------------
TotalWorkingYears : Minimum: 0, Maximum: 40
----------------------------------------
YearsAtCompany : Minimum: 0, Maximum: 40
----------------------------------------
```

Visualization

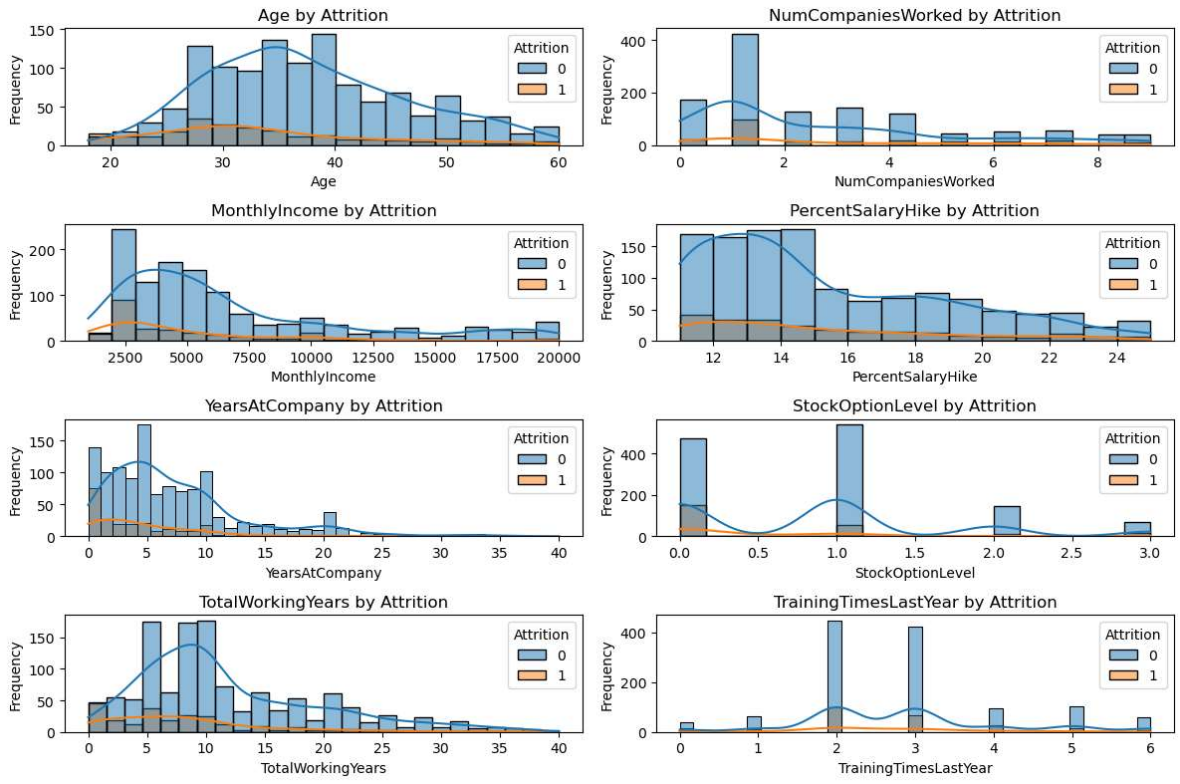
In [86]:
```
# Columns list for Ploting
columns_to_plot = [
    'DistanceFromHome', 'Education', 'RelationshipSatisfaction',
    'EnvironmentSatisfaction', 'JobInvolvement', 'JobLevel',
    'JobSatisfaction', 'NumCompaniesWorked', 'PercentSalaryHike',
    'StockOptionLevel', 'TrainingTimesLastYear', 'PerformanceRating'
]

# Set the figure size
plt.figure(figsize=(14, 10))

# Loop through the columns and create subplots
for i, column in enumerate(columns_to_plot, 1):
    plt.subplot(4, 3, i)
    sns.histplot(data=data, x=column, hue='Attrition', bins=30, kde=True)
    plt.title(f'{column} by Attrition')
    plt.xlabel(column)
    plt.ylabel('Frequency')

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



Note: 📌

It seems that EnvironmentSatisfaction, JobSatisfaction, PerformanceRating, and RelationshipSatisfaction features don't have big impact on the detrmination of Attrition of employees

In [87]:
```
# list of columns to plot
columns_to_plot = [
    ('Age', 35), ('NumCompaniesWorked', None),
    ('MonthlyIncome', 50), ('PercentSalaryHike', None),
    ('YearsAtCompany', 35), ('StockOptionLevel', None),
    ('TotalWorkingYears', 35), ('TrainingTimesLastYear', None)
]

# Set the figure size
plt.figure(figsize=(12, 8))

# Loop through the columns and create subplots
for i, (column, bins) in enumerate(columns_to_plot, 1):
    plt.subplot(4, 2, i)
    sns.histplot(data=data, x=column, hue='Attrition', kde=True)
    plt.title(f'{column} by Attrition')
    plt.xlabel(column)
    plt.ylabel('Frequency')

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



Discussions:

The workers with low JobLevel, MonthlyIncome, YearAtCompany, and TotalWorkingYears are more likely to quit there jobs. BusinessTravel : The workers who travel alot are more likely to quit then other employees.

Department: The worker in Research & Development are more likely to stay then the workers on other departement.

EducationFieId: The workers with Human Resources and Technical Degree are more likely to quit then employees from other fields of educations.

Gender: The Male are more likely to quit.

JobRole: The workers in Laboratory Technician, Sales Representative, and Human Resources are more likely to quit the workers in other positions.

MaritalStatus: The workers who have Single marital status are more likely to quit the Married, and Divorced.

OverTime: The workers who work more hours are likely to quit then others.

In [88]:
```
data.corr()
```

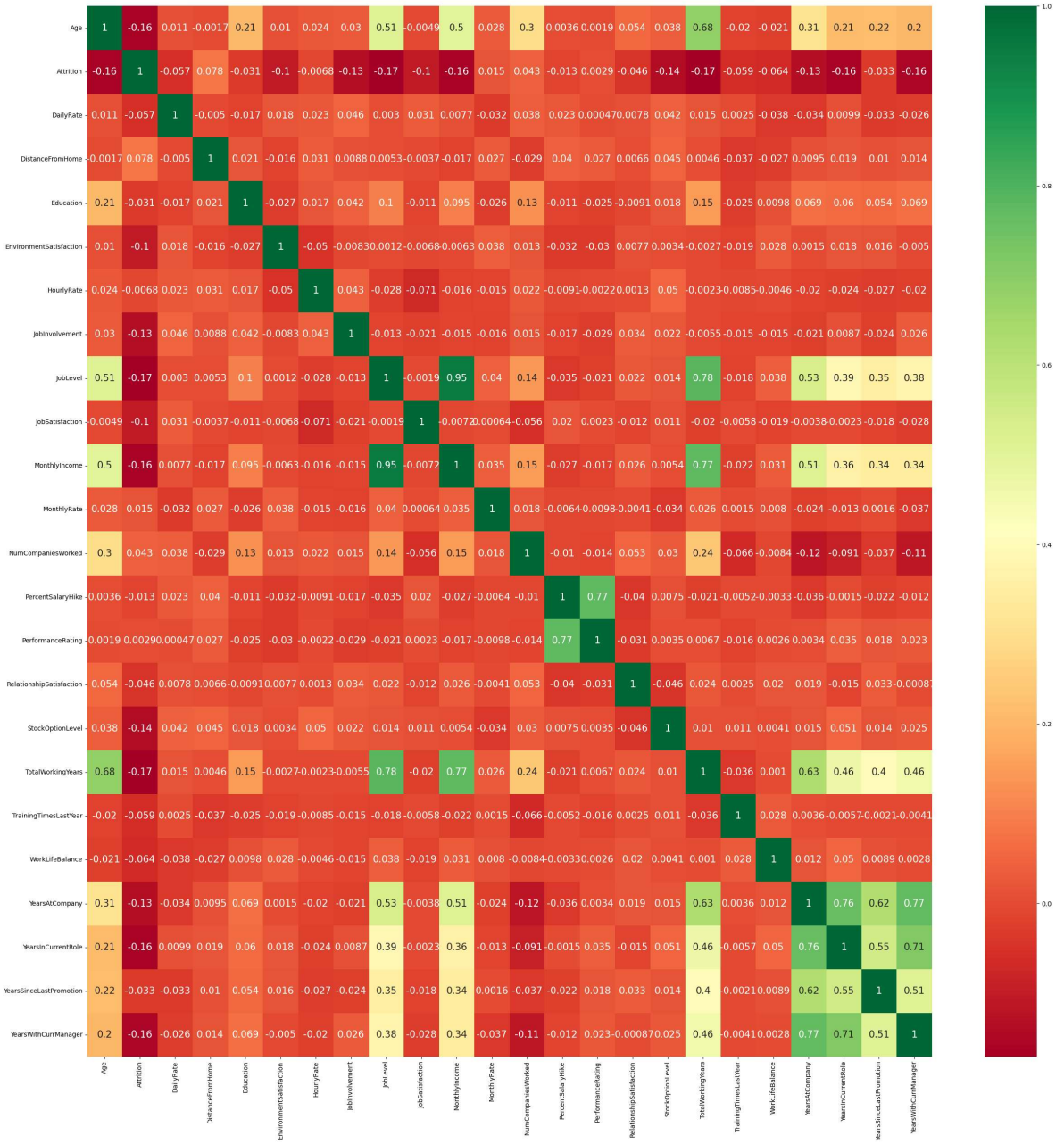
Out[88]:

	Age	Attrition	DailyRate	DistanceFromHome	Education	Enviro
Age	1.000000	-0.159205	0.010661	-0.001686	0.208034	
Attrition	-0.159205	1.000000	-0.056652	0.077924	-0.031373	
DailyRate	0.010661	-0.056652	1.000000	-0.004985	-0.016806	
DistanceFromHome	-0.001686	0.077924	-0.004985	1.000000	0.021042	
Education	0.208034	-0.031373	-0.016806	0.021042	1.000000	
EnvironmentSatisfaction	0.010146	-0.103369	0.018355	-0.016075	-0.027128	
HourlyRate	0.024287	-0.006846	0.023381	0.031131	0.016775	
JobInvolvement	0.029820	-0.130016	0.046135	0.008783	0.042438	
JobLevel	0.509604	-0.169105	0.002966	0.005303	0.101589	
JobSatisfaction	-0.004892	-0.103481	0.030571	-0.003669	-0.011296	
MonthlyIncome	0.497855	-0.159840	0.007707	-0.017014	0.094961	
MonthlyRate	0.028051	0.015170	-0.032182	0.027473	-0.026084	
NumCompaniesWorked	0.299635	0.043494	0.038153	-0.029251	0.126317	
PercentSalaryHike	0.003634	-0.013478	0.022704	0.040235	-0.011111	
PerformanceRating	0.001904	0.002889	0.000473	0.027110	-0.024539	
RelationshipSatisfaction	0.053535	-0.045872	0.007846	0.006557	-0.009118	
StockOptionLevel	0.037510	-0.137145	0.042143	0.044872	0.018422	
TotalWorkingYears	0.680381	-0.171063	0.014515	0.004628	0.148280	
TrainingTimesLastYear	-0.019621	-0.059478	0.002453	-0.036942	-0.025100	
WorkLifeBalance	-0.021490	-0.063939	-0.037848	-0.026556	0.009819	
YearsAtCompany	0.311309	-0.134392	-0.034055	0.009508	0.069114	
YearsInCurrentRole	0.212901	-0.160545	0.009932	0.018845	0.060236	
YearsSinceLastPromotion	0.216513	-0.033019	-0.033229	0.010029	0.054254	
YearsWithCurrManager	0.202089	-0.156199	-0.026363	0.014406	0.069065	

24 rows × 24 columns

In [89]:
```
plt.figure(figsize=(30, 30))
sns.heatmap(data.corr(), annot=True, cmap="RdYlGn", annot_kws={"size":15})
```

Out[89]: <Axes: >

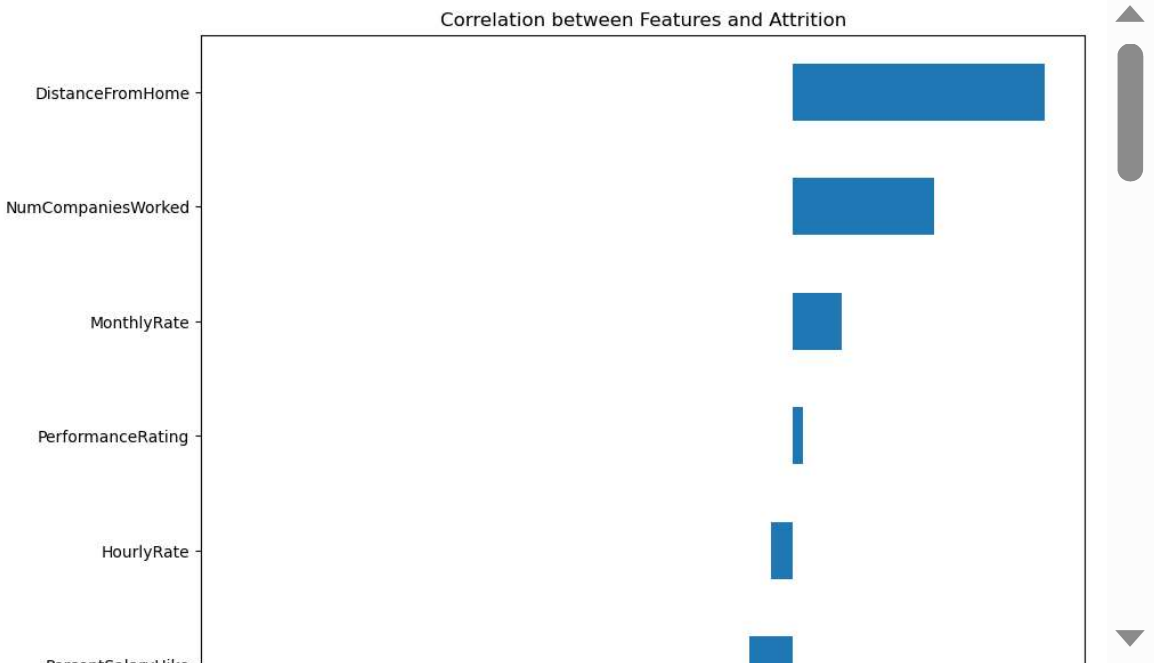


Now, we are going to calculate and display the correlation between each feature in a dataset and target variable. Correlation analysis helps us understand the relationships between variables and can provide insights into how different features are related to the target variable.

In [90]:
```
feature_correlation = data.corr()['Attrition'].sort_values()
feature_correlation
```

Out[90]:
```
TotalWorkingYears          -0.171063
JobLevel                   -0.169105
YearsInCurrentRole         -0.160545
MonthlyIncome              -0.159840
Age                        -0.159205
YearsWithCurrManager       -0.156199
StockOptionLevel           -0.137145
YearsAtCompany             -0.134392
JobInvolvement             -0.130016
JobSatisfaction            -0.103481
EnvironmentSatisfaction    -0.103369
WorkLifeBalance            -0.063939
TrainingTimesLastYear      -0.059478
DailyRate                  -0.056652
RelationshipSatisfaction   -0.045872
YearsSinceLastPromotion    -0.033019
Education                  -0.031373
PercentSalaryHike          -0.013478
HourlyRate                 -0.006846
PerformanceRating           0.002889
MonthlyRate                 0.015170
NumCompaniesWorked          0.043494
DistanceFromHome            0.077924
Attrition                   1.000000
Name: Attrition, dtype: float64
```

In [91]:
```
# Calculate the correlation between features and Attrition
feature_correlation = data.drop('Attrition', axis=1).corrwith(data.Attrition).

# Plot the correlation as a horizontal bar plot
plt.figure(figsize=(10, 30))
feature_correlation.plot(kind='barh')
plt.title('Correlation between Features and Attrition')
plt.xlabel('Correlation')
plt.ylabel('Features')
plt.show()
```



Analysis of correlation results:

- Monthly income is highly correlated with Job level.
- Job level is highly correlated with total working hours.
- Monthly income is highly correlated with total working hours.
- Age is also positively correlated with the Total working hours.
- Marital status and stock option level are negatively correlated

Handling Missing Values

In [92]:
```
# Transform categorical data into dummies
categorical_columns = []
for column in data.columns:
    if data[column].dtype == 'object' and data[column].nunique() < 20:
        categorical_columns.append(column)

data = pd.get_dummies(data, columns=categorical_columns, drop_first=True, dtyp
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 45 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Age                                   1470 non-null   int64
 1   Attrition                             1470 non-null   int32
 2   DailyRate                             1470 non-null   int64
 3   DistanceFromHome                      1470 non-null   int64
 4   Education                             1470 non-null   int64
 5   EnvironmentSatisfaction               1470 non-null   int64
 6   HourlyRate                            1470 non-null   int64
 7   JobInvolvement                        1470 non-null   int64
 8   JobLevel                              1470 non-null   int64
 9   JobSatisfaction                       1470 non-null   int64
 10  MonthlyIncome                         1470 non-null   int64
 11  MonthlyRate                           1470 non-null   int64
 12  NumCompaniesWorked                    1470 non-null   int64
 13  PercentSalaryHike                     1470 non-null   int64
 14  PerformanceRating                     1470 non-null   int64
 15  RelationshipSatisfaction              1470 non-null   int64
 16  StockOptionLevel                      1470 non-null   int64
 17  TotalWorkingYears                     1470 non-null   int64
 18  TrainingTimesLastYear                 1470 non-null   int64
 19  WorkLifeBalance                       1470 non-null   int64
 20  YearsAtCompany                        1470 non-null   int64
 21  YearsInCurrentRole                    1470 non-null   int64
 22  YearsSinceLastPromotion               1470 non-null   int64
 23  YearsWithCurrManager                  1470 non-null   int64
 24  BusinessTravel_Travel_Frequently      1470 non-null   uint8
 25  BusinessTravel_Travel_Rarely          1470 non-null   uint8
 26  Department_Research & Development     1470 non-null   uint8
 27  Department_Sales                      1470 non-null   uint8
 28  EducationField_Life Sciences          1470 non-null   uint8
 29  EducationField_Marketing              1470 non-null   uint8
 30  EducationField_Medical                1470 non-null   uint8
 31  EducationField_Other                  1470 non-null   uint8
 32  EducationField_Technical Degree       1470 non-null   uint8
 33  Gender_Male                           1470 non-null   uint8
 34  JobRole_Human Resources               1470 non-null   uint8
 35  JobRole_Laboratory Technician         1470 non-null   uint8
 36  JobRole_Manager                       1470 non-null   uint8
 37  JobRole_Manufacturing Director        1470 non-null   uint8
 38  JobRole_Research Director             1470 non-null   uint8
 39  JobRole_Research Scientist            1470 non-null   uint8
 40  JobRole_Sales Executive               1470 non-null   uint8
 41  JobRole_Sales Representative          1470 non-null   uint8
 42  MaritalStatus_Married                 1470 non-null   uint8
 43  MaritalStatus_Single                  1470 non-null   uint8
 44  OverTime_Yes                          1470 non-null   uint8
dtypes: int32(1), int64(23), uint8(21)
memory usage: 300.2 KB
```

In [93]:
```
print(data.shape)

# Remove duplicate Features
data = data.T.drop_duplicates()
data = data.T

# Remove Duplicate Rows
data.drop_duplicates(inplace=True)

print(data.shape)
```

```
(1470, 45)
(1470, 45)
```

In [94]:
```
data.isnull().sum()
```

Out[94]:
```
Age                                  0
Attrition                            0
DailyRate                            0
DistanceFromHome                     0
Education                            0
EnvironmentSatisfaction              0
HourlyRate                           0
JobInvolvement                       0
JobLevel                             0
JobSatisfaction                      0
MonthlyIncome                        0
MonthlyRate                          0
NumCompaniesWorked                   0
PercentSalaryHike                    0
PerformanceRating                    0
RelationshipSatisfaction             0
StockOptionLevel                     0
TotalWorkingYears                    0
TrainingTimesLastYear                0
WorkLifeBalance                      0
YearsAtCompany                       0
YearsInCurrentRole                   0
YearsSinceLastPromotion              0
YearsWithCurrManager                 0
BusinessTravel_Travel_Frequently     0
BusinessTravel_Travel_Rarely         0
Department_Research & Development    0
Department_Sales                     0
EducationField_Life Sciences         0
EducationField_Marketing             0
EducationField_Medical               0
EducationField_Other                 0
EducationField_Technical Degree      0
Gender_Male                          0
JobRole_Human Resources              0
JobRole_Laboratory Technician        0
JobRole_Manager                      0
JobRole_Manufacturing Director       0
JobRole_Research Director            0
JobRole_Research Scientist           0
JobRole_Sales Executive              0
JobRole_Sales Representative         0
MaritalStatus_Married                0
MaritalStatus_Single                 0
OverTime_Yes                         0
dtype: int64
```

Splitting Dataset

In [95]:
```
X = data.drop('Attrition', axis=1)
y = data.Attrition

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, randc
```

Logistic Regression

In [96]:
```
from sklearn.linear_model import LogisticRegression

Logistic_Regression_Model = LogisticRegression(solver='liblinear', penalty='l1
Logistic_Regression_Model.fit(X_train, y_train)
```

Out[96]:
```
LogisticRegression

LogisticRegression(penalty='l1', solver='liblinear')
```

Prediction

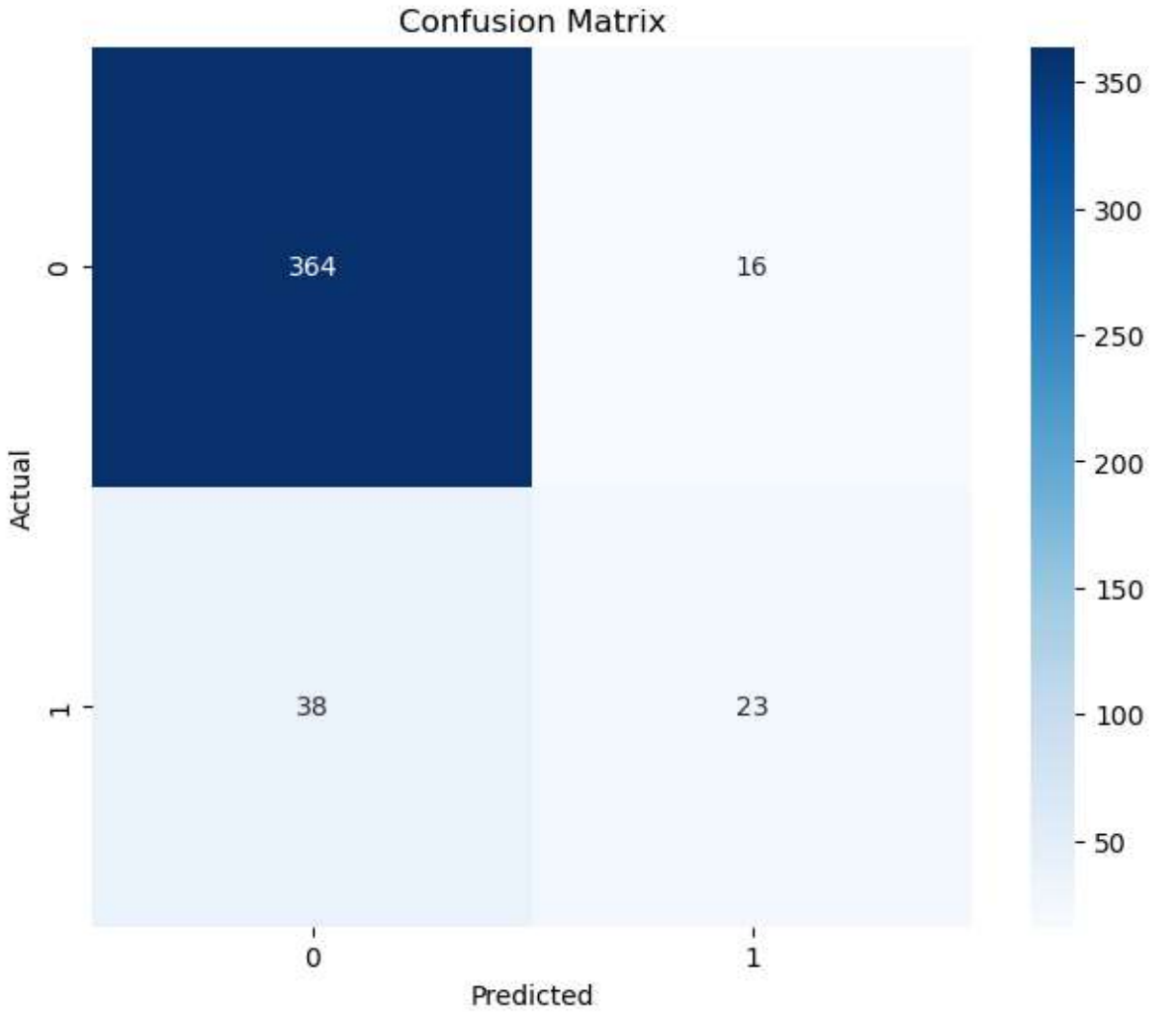
In [97]:
```
y_pred = Logistic_Regression_Model.predict(X_test)
```

Evaluation

In [98]:
```
Logistic_Regression_Model_Accuracy = accuracy_score(y_test, y_pred)
print("Test Accuracy:", Logistic_Regression_Model_Accuracy)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Test Accuracy: 0.8775510204081632

Classification Report:
               precision    recall  f1-score   support

           0       0.91      0.96      0.93       380
           1       0.59      0.38      0.46        61

    accuracy                           0.88       441
   macro avg       0.75      0.67      0.70       441
weighted avg       0.86      0.88      0.87       441
```

In [109]:
```
# Calculate the confusion matrix
Logistic_Regression_Confusion_Matrix = confusion_matrix(y_test, y_pred)

# Create a heatmap for the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(Logistic_Regression_Confusion_Matrix, annot=True, fmt='d', cmap='B
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



In [ ]:
```
```

Random Forest Model

In [117]:
```
Random_Forest_Model = RandomForestClassifier(random_state=42)
Random_Forest_Model.fit(X_train, y_train)
```

Out[117]:
```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

Prediction

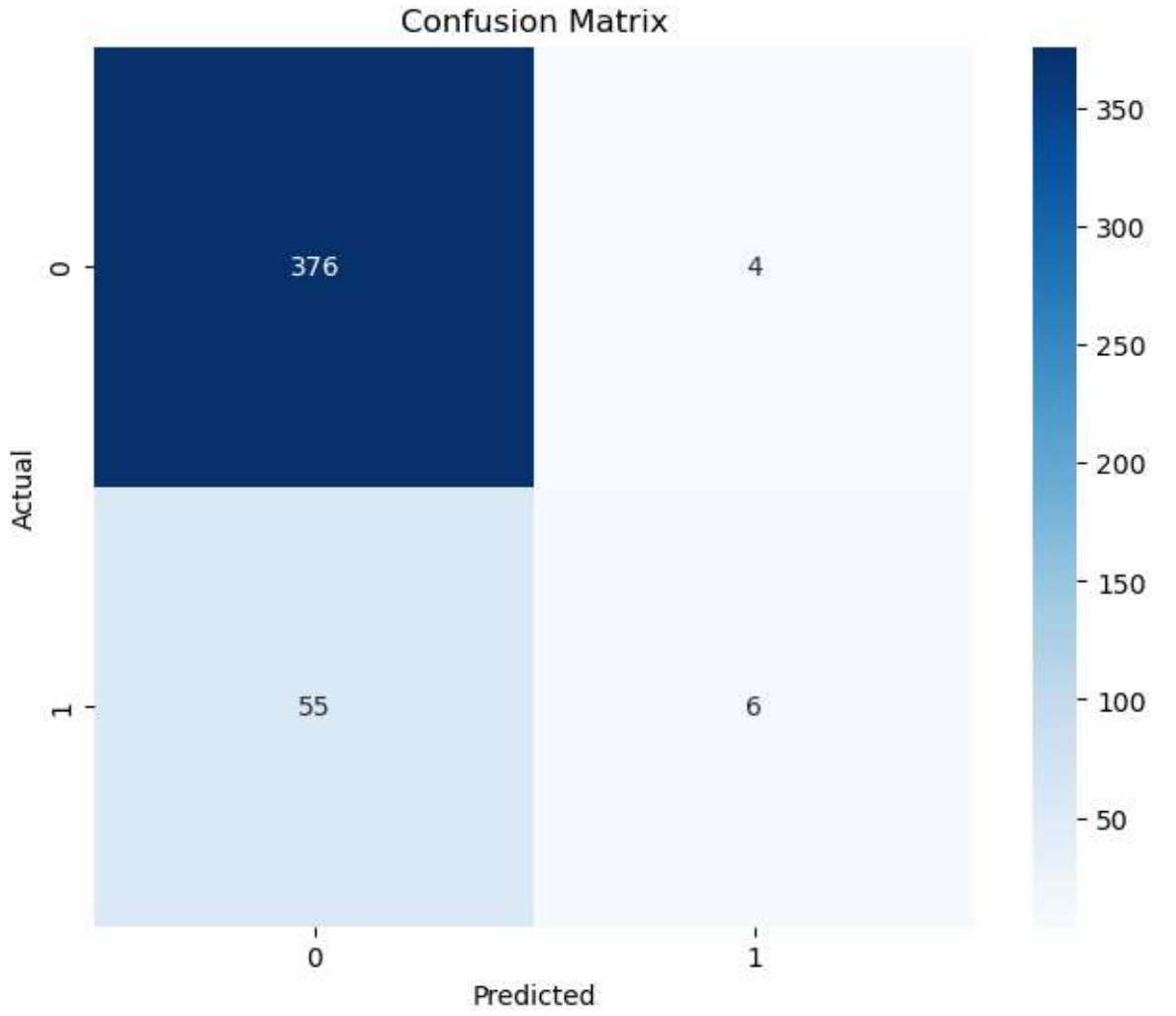
In [118]:
```
y_pred = Random_Forest_Model.predict(X_test)
```

Evaluation

In [119]:
```
Random_Forest_Model_Accuracy = accuracy_score(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print("Test Accuracy:", Random_Forest_Model_Accuracy)
print("\nClassification Report:\n", class_report)
```

```
Test Accuracy: 0.8662131519274376

Classification Report:
               precision    recall  f1-score   support

           0       0.87      0.99      0.93       380
           1       0.60      0.10      0.17        61

    accuracy                           0.87       441
   macro avg       0.74      0.54      0.55       441
weighted avg       0.83      0.87      0.82       441
```

In [120]:
```
# Calculate the confusion matrix
Random_Forest_Confusion_Matrix = confusion_matrix(y_test, y_pred)

# Create a heatmap for the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(Random_Forest_Confusion_Matrix, annot=True, fmt='d', cmap='Blues',
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



In [ ]:

Support Victor Machine

In [121]:
```
SVM_Model = SVC(kernel='linear', random_state=42)
SVM_Model.fit(X_train, y_train)
```

Out[121]:
```
SVC(kernel='linear', random_state=42)
```

Prediction

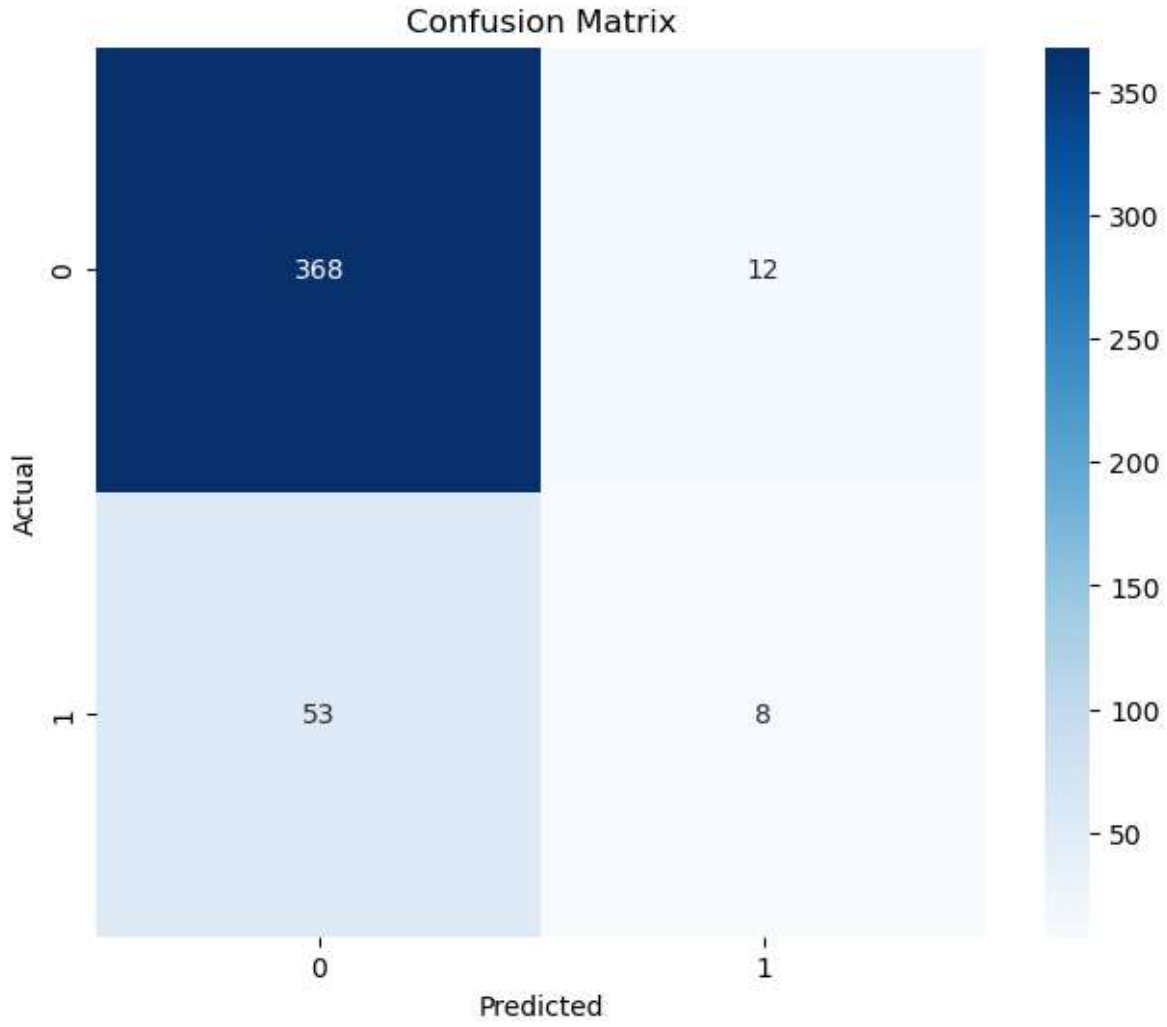
In [122]:
```
y_pred = SVM_Model.predict(X_test)
```

Evaluation

In [123]:
```
SVM_Model_Accuracy = accuracy_score(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print("Test Accuracy:", SVM_Model_Accuracy)
print("\nClassification Report:\n", class_report)
```

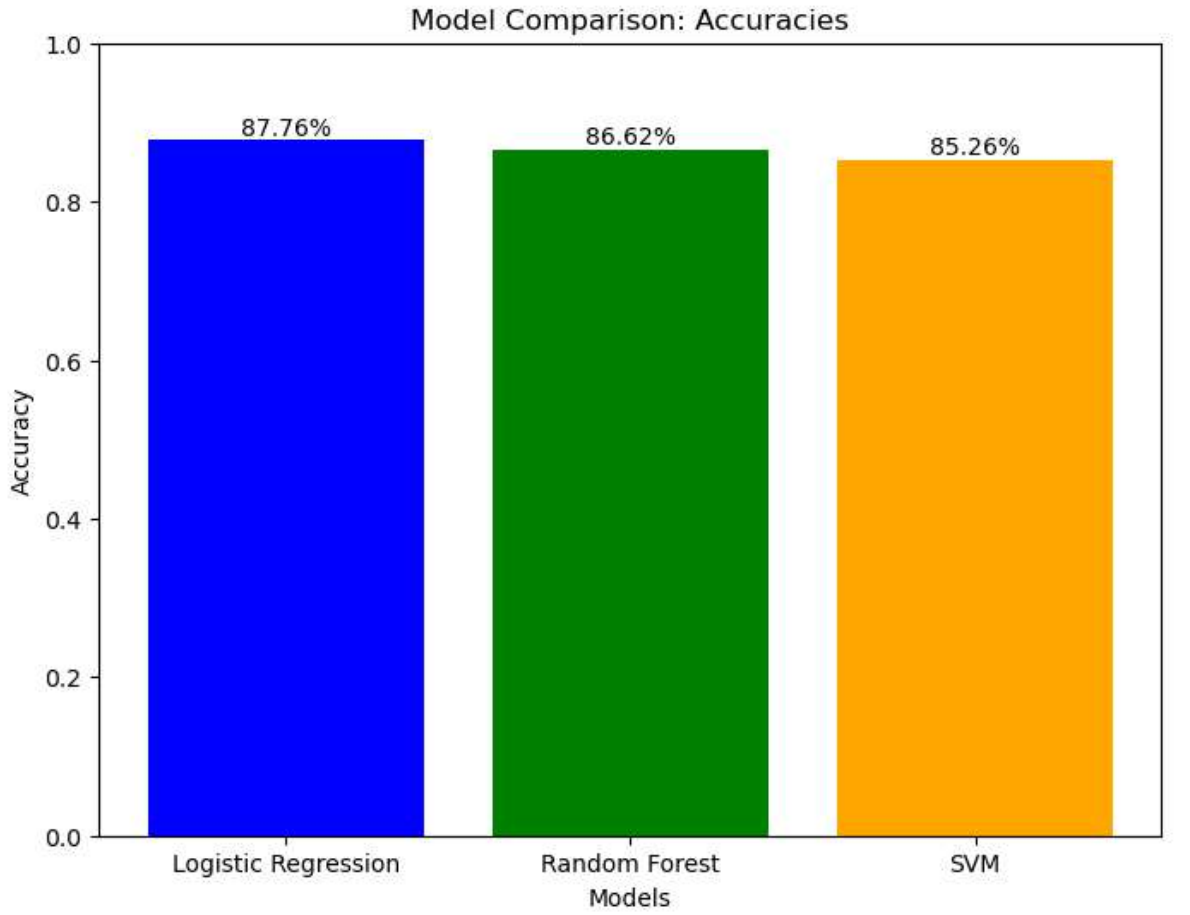
```
Test Accuracy: 0.8526077097505669

Classification Report:
               precision    recall  f1-score   support

           0       0.87      0.97      0.92       380
           1       0.40      0.13      0.20        61

    accuracy                           0.85       441
   macro avg       0.64      0.55      0.56       441
weighted avg       0.81      0.85      0.82       441
```

In [124]:
```
# Calculate the confusion matrix
SVM_Confusion_Matrix = confusion_matrix(y_test, y_pred)

# Create a heatmap for the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(SVM_Confusion_Matrix, annot=True, fmt='d', cmap='Blues', square=Tr
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



Comparing Logistic Regression VS Random Forest VS SVM

In [125]:
```
models = ['Logistic Regression', 'Random Forest', 'SVM']
accuracies = [Logistic_Regression_Model_Accuracy, Random_Forest_Model_Accuracy

# Create a bar plot with percentage labels
plt.figure(figsize=(8, 6))
bars = plt.bar(models, accuracies, color=['blue', 'green', 'orange'])
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Model Comparison: Accuracies')
plt.ylim([0, 1])  # Set y-axis limit to 0-1

# Add percentage labels to the bars
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, height, f'{height:.2%}', ha='

plt.show()
```



Logistic Regression achieved the highest accuracy of 87.76%, making it a reliable choice for this task. It provides a good balance between accuracy and simplicity, making it suitable for quick and interpretable predictions.

Random Forest followed closely with an accuracy of 86.62%. It offers a more complex approach by combining multiple decision trees, potentially capturing intricate patterns in the data. While it sacrifices some interpretability, it provides competitive accuracy.

Support Vector Machine (SVM) attained an accuracy of 85.26%. While slightly behind the other two models in accuracy, SVM's strength lies in handling complex decision boundaries. It could be considered if the data has non-linear relationships that require more advanced modeling.

Conclusion

In this employee turnover prediction task, we employed three machine learning models: Logistic Regression, Random Forest, and SVM. After thorough data exploration and model evaluation, we found that Logistic Regression achieved the highest accuracy (87.76%), making it a suitable choice for making quick and interpretable predictions. Random Forest closely followed with an accuracy of 86.62%, providing a more complex model that captures intricate patterns. SVM demonstrated an accuracy of 85.26%, showcasing its ability to handle complex decision boundaries. The choice of the best model depends on the specific needs of the HR department and the desired balance between accuracy and interpretability.