MINOR PROJECT

TASK 1 - Exploratory Data Analysis

Loading and Reading Data

In [11]:
```
import pandas as pd
data = pd.read_csv(r"C:\Users\manoj\AppData\Local\Temp\Temp1_archive (2).zip\W
print(data)
```

```
      Age Attrition     BusinessTravel  DailyRate              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 [6]:
```
(data.columns)
```

Out[6]:
```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

In [14]:
```
data.head(5)
```

Out[14]:

	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 [7]:
```
missing_values = data.isnull().sum()
print("Missing values:\n", missing_values)
```

```
Missing values:
 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
```

QUESTION 1

In [9]:
```
# Now checking for duplicate records
duplicate_records = data.duplicated().sum()
print("Duplicate records:", duplicate_records)
```

```
Duplicate records: 0
```

In [10]:
```
# Remove duplicate records
data.drop_duplicates(inplace=True)
```

In [11]:
```
# Handle missing values
data.dropna(inplace=True)
```

In [13]:
```
print("Cleaned dataset:\n",data)
```

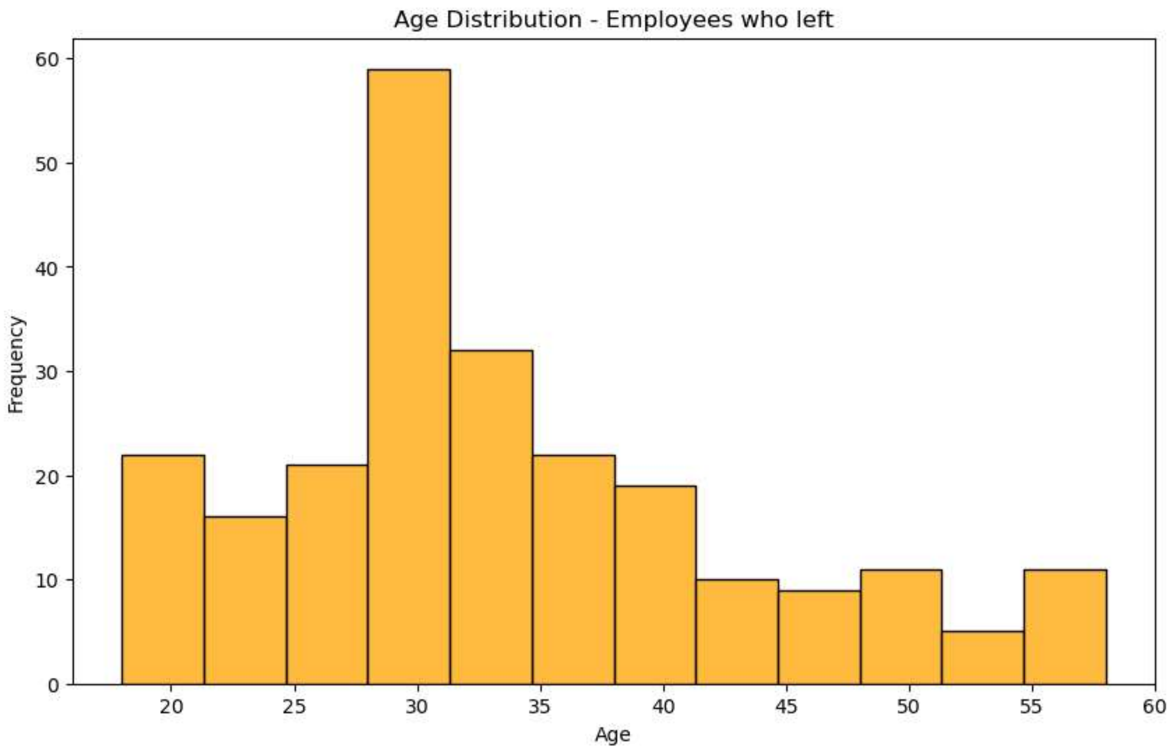
```
Cleaned dataset:
      Age Attrition     BusinessTravel  DailyRate              Department  \
0      41       Yes      Travel_Rarely       1102                   Sales
1      49        No  Travel_Frequently        279  Research & Development
2      37       Yes      Travel_Rarely       1373  Research & 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]
```

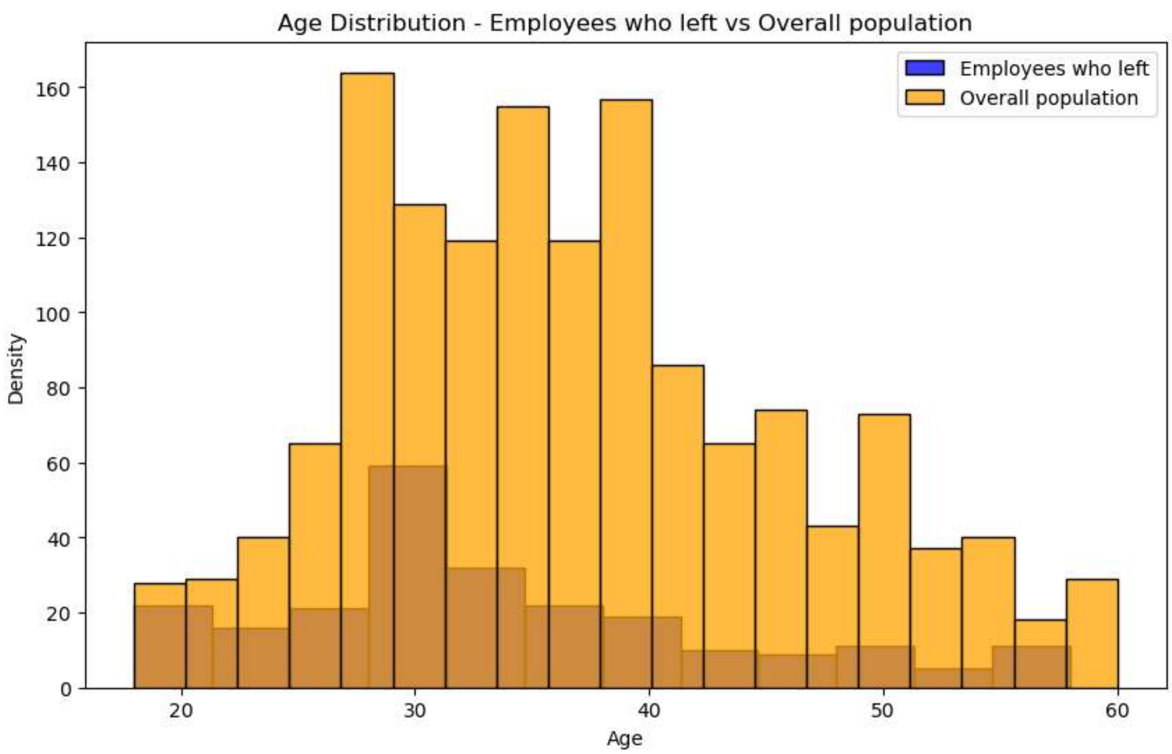
QUESTION 2

In [12]:
```
employees_that_left = data[data['Attrition'] == 'Yes']
```

In [30]:
```
import seaborn as sns
import matplotlib.pyplot as plt

employees_that_left = data[data['Attrition'] == 'Yes']

plt.figure(figsize=(10, 6))

sns.histplot(employees_that_left['Age'],color='orange')

plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Age Distribution - Employees who left')
plt.show()
```



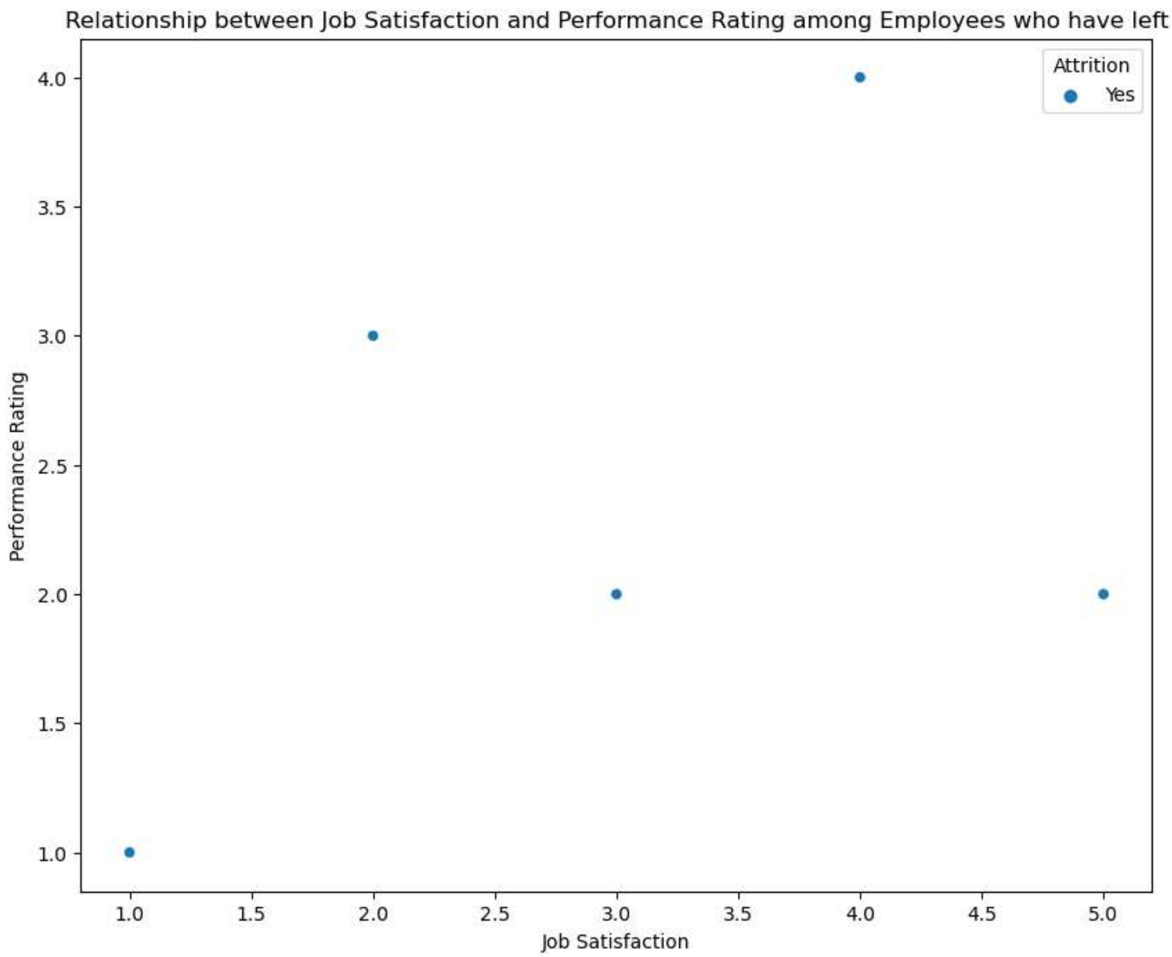
In [13]:
```
import seaborn as sns
import matplotlib.pyplot as plt

employees_that_left = data[data['Attrition'] == 'Yes']

plt.figure(figsize=(10, 6))

sns.histplot(employees_that_left['Age'],color='blue', label='Employees who lef

sns.histplot(data['Age'],color='orange', label='Overall population')

plt.xlabel('Age')
plt.ylabel('Density')
plt.title('Age Distribution - Employees who left vs Overall population')
plt.legend()
plt.show()
```



In [14]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.scatter(employees_that_left['TotalWorkingYears'], employees_that_left['Mon
plt.xlabel('Years of Experience')
plt.ylabel('Monthly Income')
plt.title('Salary vs Years of Experience (For Employees who left)')
plt.show()
```

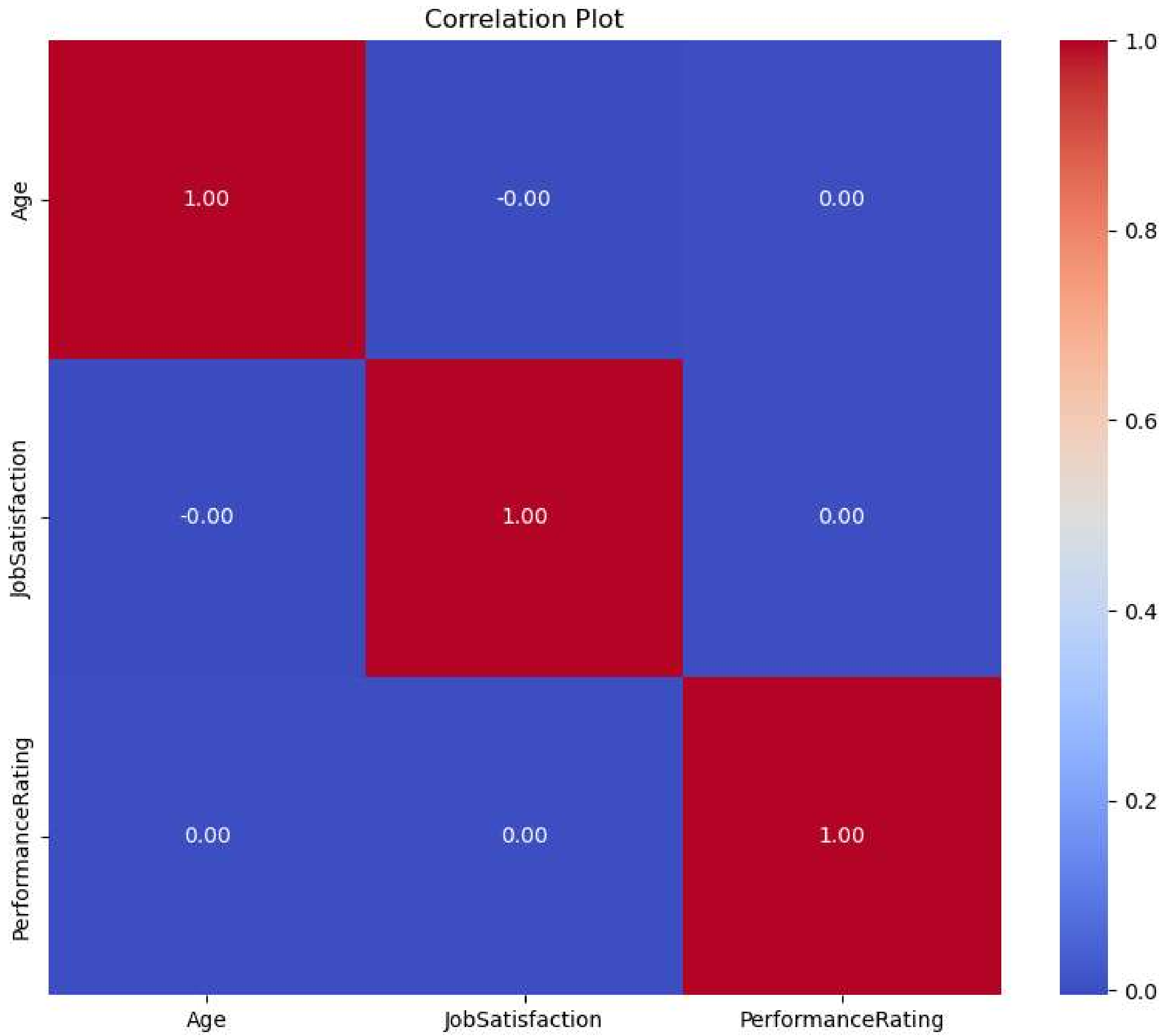
QUESTION 3

In [5]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 8))
sns.scatterplot(x='JobSatisfaction', y='PerformanceRating', data=employees_tha

plt.xlabel('Job Satisfaction')
plt.ylabel('Performance Rating')
plt.title('Relationship between Job Satisfaction and Performance Rating among

plt.show()
```



QUESTION 4

In [20]:
```
import seaborn as sns

relevant_columns = ['Age', 'JobSatisfaction', 'PerformanceRating', 'Attrition'
relevant_data = data[relevant_columns]

correlation_matrix = relevant_data.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", square

plt.title('Correlation Plot')

plt.show()
```



In [23]:
```
employees_that_left = len(data[data['Attrition'] == 'Yes'])
total_employees = len(data)

employees_that_left = (employees_that_left / total_employees) * 100

print("Overall Attrition Rate: {:.2f}%".format(employees_that_left))
```

```
Overall Attrition Rate: 16.12%
```

"In this example, we start by selecting specific columns of interest, namely 'Age', 'JobSatisfaction', 'PerformanceRating', and 'Attrition', from the dataset. Next, we calculate the correlation matrix using the corr() function, which measures the relationships between these variables.

QUESTION 5

In [29]:
```
data['DistanceFromHome'] = pd.to_numeric(data['DistanceFromHome'], errors='coe

data['Attrition_numeric'] = (data['Attrition'] == 'Yes').astype(int)

correlation_dis_attr = data['DistanceFromHome'].corr(data['Attrition_numeric']

print("Correlation coefficient between Distance From Home and Attrition:", cor
```

```
Correlation coefficient between Distance From Home and Attrition: 0.077923582
95570351
```

In [33]:
```
import pandas as pd
distance_ranges = [(0, 5), (6, 10), (11, 15), (16, 20), (21, 25), (26, 30), (3

attrition_rates = []
for distance_range in distance_ranges:
    distance_from = distance_range[0]
    distance_to = distance_range[1]

    filtered_data = data[(data['DistanceFromHome'] >= distance_from) & (data['

    attrition_count = filtered_data['Attrition'].value_counts().get('Yes', 0)
    total_count = filtered_data.shape[0]

    if total_count != 0:
        attrition_rate = (attrition_count / total_count) * 100
    else:
        attrition_rate = 0

    attrition_rates.append(attrition_rate)

results_df = pd.DataFrame({'Distance Range': distance_ranges, 'Attrition Rate'

print(results_df)
```

```
  Distance Range  Attrition Rate
0         (0, 5)       13.765823
1        (6, 10)       14.467005
2       (11, 15)       21.739130
3       (16, 20)       18.400000
4       (21, 25)       27.350427
5       (26, 30)       14.942529
6      (31, inf)        0.000000
```

TASK 2 - Classification/Regression

Data Preprocessing (as per requirement), Feature Engineering, Split dataset in train-test (80:20 ratio), Model selection, Model training, Model evaluation, Fine-tune the Model, Make predictions, Summarize your model's performance by evaluation metrices.

DATA PREPROCESSING AND FEATURE ENGINEERING

In [37]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

data = pd.read_csv(r"C:\Users\manoj\AppData\Local\Temp\Temp1_archive (2).zip\W

data.dropna(inplace=True)  # Dropping rows with missing values

# 2. Encoding Categorical Variables
categorical_columns = ['BusinessTravel', 'Department', 'EducationField', 'Genc
data_encoded = pd.get_dummies(data, columns=categorical_columns)

# 3. Feature Scaling
numerical_columns = ['Age', 'DailyRate', 'DistanceFromHome', 'HourlyRate', 'Mc
                     'PercentSalaryHike', 'TotalWorkingYears', 'TrainingTimesL
                     'YearsSinceLastPromotion', 'YearsWithCurrManager']
scaler = StandardScaler()
data_encoded[numerical_columns] = scaler.fit_transform(data_encoded[numerical_

# 4. Train-Test Split
X = data_encoded.drop('Attrition', axis=1)
y = data_encoded['Attrition']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc

print(data)
```

```
      Age Attrition     BusinessTravel  DailyRate              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]
```

Age Binning: Instead of using the continuous "Age" feature, you can create age groups or bins. This can capture non-linear relationships between age and the target variable.

In [38]:
```
data['AgeBin'] = pd.cut(data['Age'], bins=[18, 30, 40, 50, 60, 70], labels=['1
```

Employee Experience: You can create a new feature that represents the total experience of an employee by combining the "TotalWorkingYears" and "YearsAtCompany" features.

In [39]:
```
data['Experience'] = data['TotalWorkingYears'] - data['YearsAtCompany']
```

Interaction Features: Create interaction features by combining two or more existing features. For example, you can create a feature that represents the interaction between job satisfaction and work-life balance.

In [41]:
```
data['JobSatisfaction_WorkLifeBalance'] = data['JobSatisfaction'] * data['Work
```

Target Encoding: If you have categorical features with high cardinality, you can encode them using the mean target value of each category. This can capture the relationship between the categorical feature and the target variable. python.

In [46]:
```
data['Attrition'] = data['Attrition'].map({'Yes': 1, 'No': 0})
# As "Attrition" column used contains non-numeric values. Target encoding requ
mean_target_encoding = data.groupby('EducationField')['Attrition'].mean()
data['EducationField_Encoded'] = data['EducationField'].map(mean_target_encodi
```

SPLITTING DATASET

In [50]:
```
from sklearn.model_selection import train_test_split
X = data.drop('Attrition', axis=1)
y = data['Attrition']

# Splitting the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
```

MODEL SELECTION AND TRAINING

MODEL USED - LOGISTIC REGRESSION

In [77]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Load the dataset
data = pd.read_csv(r"C:\Users\manoj\AppData\Local\Temp\Temp1_archive (2).zip\W

# Select the features and target variable
features = ['Age', 'DistanceFromHome', 'Education', 'JobLevel']
target = 'Attrition'

# List of categorical columns for one-hot encoding
categorical_cols = ['BusinessTravel', 'Department', 'Gender']

# Convert categorical variables to numerical using one-hot encoding
data = pd.get_dummies(data, columns=categorical_cols)
```

In [78]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data[features], data[targe
```

In [84]:
```
param_grid = {
    'C': [0.1, 1, 10],
    'penalty': ['l1', 'l2']
}
```

In [105]:
```
# Initialize the logistic regression model
model = LogisticRegression(solver='liblinear')
```

In [106]:
```
from sklearn.model_selection import GridSearchCV

# Perform grid search cross-validation
grid_search = GridSearchCV(model, param_grid, cv=5)
grid_search.fit(X_train, y_train)

# Get the best hyperparameter values
best_params = grid_search.best_params_
print("Best Hyperparameters:", best_params)

# Train the model with the best hyperparameter values
best_model = LogisticRegression(solver='liblinear', **best_params)
```

```
Best Hyperparameters: {'C': 0.1, 'penalty': 'l1'}
```

MODEL TRAINING

In [107]:
```
# Train the model
model.fit(X_train, y_train)
# Make predictions on the test set
y_pred = model.predict(X_test)
```

MODEL EVALUATION AND CLASSIFICATION REPORT

In [108]:
```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8673469387755102
```

In [109]:
```
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred, zero_division=1)
print("Classification Report:\n", report)
```

```
Classification Report:
               precision    recall  f1-score   support

          No       0.87      1.00      0.93       255
         Yes       1.00      0.00      0.00        39

    accuracy                           0.87       294
   macro avg       0.93      0.50      0.46       294
weighted avg       0.88      0.87      0.81       294
```

Overall, this trains a logistic regression model on the provided dataset and evaluates its accuracy and performance using metrics from the classification report.

Name - APOORVA

Branch - AI-ML

Roll NO. - 23076