PROBLEM STATEMENT

*Take any dataset of you choice and perform EDA(Exploratory Data Analysis) and apply a suitable Classifier,Regressor or Clusterer and calculate accuracy of model**

```
Name : Shivam Bendre
Batch: AprilL -May
Domain: Data Science With Python
```

In [56]:

```
# Importing the necessary libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.ensemble import RandomForestClassifier
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

Library Import Done...

In [57]:

```
# Loading the csv file
data = pd.read_csv('Hr.csv')
```

CSV FiLe Loaded...

In [58]:

```
#Printing Rows,CoLumns Count
data.shape
```

Out[58]:

```
(1200, 28)
```

In [59]:

```
#Printing Index Range Of DaTa
data.index
```

Out[59]:

```
RangeIndex(start=0, stop=1200, step=1)
```

In [60]:

```
#Printing Index / Names Of  ALL CoLumnS
data.columns
```

Out[60]:

```
Index(['EmpNumber', 'Age', 'Gender', 'EducationBackground', 'MaritalStatu
s',
       'EmpDepartment', 'EmpJobRole', 'BusinessTravelFrequency',
       'DistanceFromHome', 'EmpEducationLevel', 'EmpEnvironmentSatisfact
ion',
       'EmpHourlyRate', 'EmpJobInvolvement', 'EmpJobLevel',
       'EmpJobSatisfaction', 'NumCompaniesWorked', 'OverTime',
       'EmpLastSalaryHikePercent', 'EmpRelationshipSatisfaction',
       'TotalWorkExperienceInYears', 'TrainingTimesLastYear',
       'EmpWorkLifeBalance', 'ExperienceYearsAtThisCompany',
       'ExperienceYearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager', 'Attrition', 'PerformanceRating'],
      dtype='object')
```

In [61]:

```
#Printing CoLumn Values Array
data.columns.values
```

Out[61]:

```
array(['EmpNumber', 'Age', 'Gender', 'EducationBackground',
       'MaritalStatus', 'EmpDepartment', 'EmpJobRole',
       'BusinessTravelFrequency', 'DistanceFromHome', 'EmpEducationLevel',
       'EmpEnvironmentSatisfaction', 'EmpHourlyRate', 'EmpJobInvolvemen
t',
       'EmpJobLevel', 'EmpJobSatisfaction', 'NumCompaniesWorked',
       'OverTime', 'EmpLastSalaryHikePercent',
       'EmpRelationshipSatisfaction', 'TotalWorkExperienceInYears',
       'TrainingTimesLastYear', 'EmpWorkLifeBalance',
       'ExperienceYearsAtThisCompany', 'ExperienceYearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrManager', 'Attrition',
       'PerformanceRating'], dtype=object)
```

In [62]:

```
#Printing data from CSV FiLe
data.head(4)
```

Out[62]:

	EmpNumber	Age	Gender	EducationBackground	MaritalStatus	EmpDepa
0	E1001000	32	Male	Marketing	Single	
1	E1001006	47	Male	Marketing	Single	
2	E1001007	40	Male	Life Sciences	Married	
3	E1001009	41	Male	Human Resources	Divorced	Re:

4 rows × 28 columns

In [63]:

```
# Looking for missing data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1200 entries, 0 to 1199
Data columns (total 28 columns):
 #   Column                        Non-Null Count  Dtype 
---  ------                        --------------  ----- 
 0   EmpNumber                     1200 non-null   object
 1   Age                           1200 non-null   int64 
 2   Gender                        1200 non-null   object
 3   EducationBackground           1200 non-null   object
 4   MaritalStatus                 1200 non-null   object
 5   EmpDepartment                 1200 non-null   object
 6   EmpJobRole                    1200 non-null   object
 7   BusinessTravelFrequency       1200 non-null   object
 8   DistanceFromHome              1200 non-null   int64 
 9   EmpEducationLevel             1200 non-null   int64 
 10  EmpEnvironmentSatisfaction    1200 non-null   int64 
 11  EmpHourlyRate                 1200 non-null   int64 
 12  EmpJobInvolvement             1200 non-null   int64 
 13  EmpJobLevel                   1200 non-null   int64 
 14  EmpJobSatisfaction            1200 non-null   int64 
 15  NumCompaniesWorked            1200 non-null   int64 
 16  OverTime                      1200 non-null   object
 17  EmpLastSalaryHikePercent      1200 non-null   int64 
 18  EmpRelationshipSatisfaction   1200 non-null   int64 
 19  TotalWorkExperienceInYears    1200 non-null   int64 
 20  TrainingTimesLastYear         1200 non-null   int64 
 21  EmpWorkLifeBalance            1200 non-null   int64 
 22  ExperienceYearsAtThisCompany  1200 non-null   int64 
 23  ExperienceYearsInCurrentRole  1200 non-null   int64 
 24  YearsSinceLastPromotion       1200 non-null   int64 
 25  YearsWithCurrManager          1200 non-null   int64 
 26  Attrition                     1200 non-null   object
 27  PerformanceRating             1200 non-null   int64 
dtypes: int64(19), object(9)
memory usage: 262.6+ KB
```

In [64]:

```
#Printing DaTaTypes Of ALL CoLumnS
data.dtypes
```

Out[64]:

```
EmpNumber                        object
Age                               int64
Gender                           object
EducationBackground              object
MaritalStatus                    object
EmpDepartment                    object
EmpJobRole                       object
BusinessTravelFrequency          object
DistanceFromHome                  int64
EmpEducationLevel                 int64
EmpEnvironmentSatisfaction        int64
EmpHourlyRate                     int64
EmpJobInvolvement                 int64
EmpJobLevel                       int64
EmpJobSatisfaction                int64
NumCompaniesWorked                int64
OverTime                         object
EmpLastSalaryHikePercent          int64
EmpRelationshipSatisfaction       int64
TotalWorkExperienceInYears        int64
TrainingTimesLastYear             int64
EmpWorkLifeBalance                int64
ExperienceYearsAtThisCompany      int64
ExperienceYearsInCurrentRole      int64
YearsSinceLastPromotion           int64
YearsWithCurrManager              int64
Attrition                        object
PerformanceRating                 int64
dtype: object
```

In [65]:

```
data.describe()
```

Out[65]:

iesWorked	EmpLastSalaryHikePercent	EmpRelationshipSatisfaction	TotalWor
200.000000	1200.000000	1200.000000	
2.665000	15.222500	2.725000	
2.469384	3.625918	1.075642	
0.000000	11.000000	1.000000	
1.000000	12.000000	2.000000	
2.000000	14.000000	3.000000	
4.000000	18.000000	4.000000	
9.000000	25.000000	4.000000	

DATA VISUALIZATION

In [66]:

```
# A new pandas Dataframe is created to analyze department wise performance as ask
dept = data.iloc[:,[5,27]].copy()
dept_per = dept.copy()
```

In [67]:

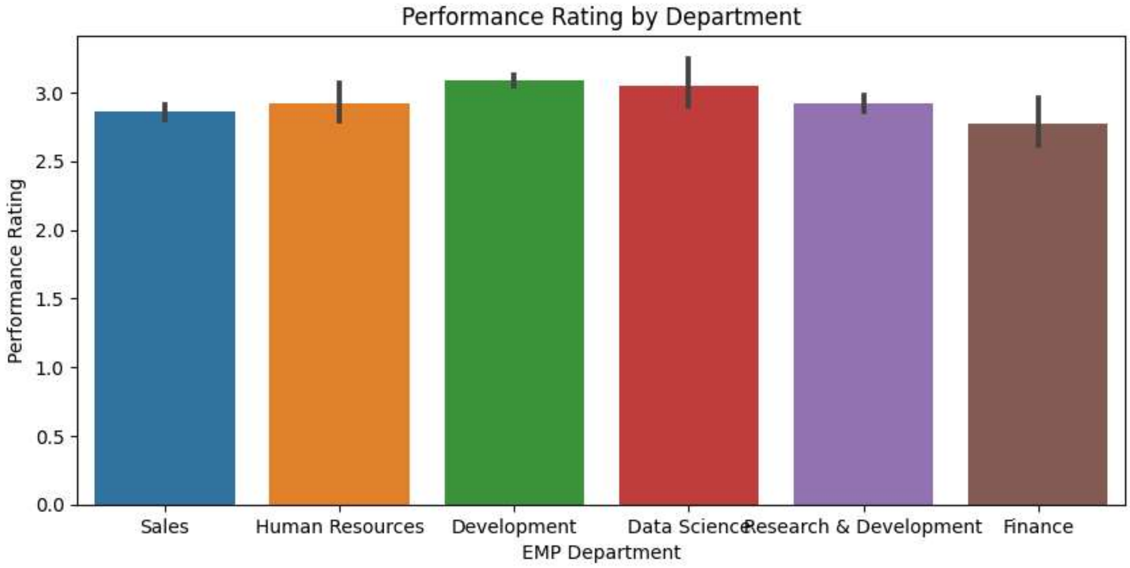
```
# Finding out the mean performance of all the departments and plotting its bar graph
dept_per.groupby(by='EmpDepartment')['PerformanceRating'].mean()
```

Out[67]:

```
EmpDepartment
Data Science              3.050000
Development               3.085873
Finance                   2.775510
Human Resources           2.925926
Research & Development    2.921283
Sales                     2.860590
Name: PerformanceRating, dtype: float64
```

In [68]:

```
plt.figure(figsize=(10,4.5)) # Set figure size
sns.barplot(x='EmpDepartment', y='PerformanceRating', data=dept_per)#color='blue
plt.title('Performance Rating by Department') # Add title
plt.xlabel('EMP Department') # Add x-axis label
plt.ylabel('Performance Rating') # Add y-axis label
plt.show() # Show the plot
```



In [69]:

```
# Analyze each department separately
dept_per.groupby(by='EmpDepartment')['PerformanceRating'].value_counts()
```

Out[69]:

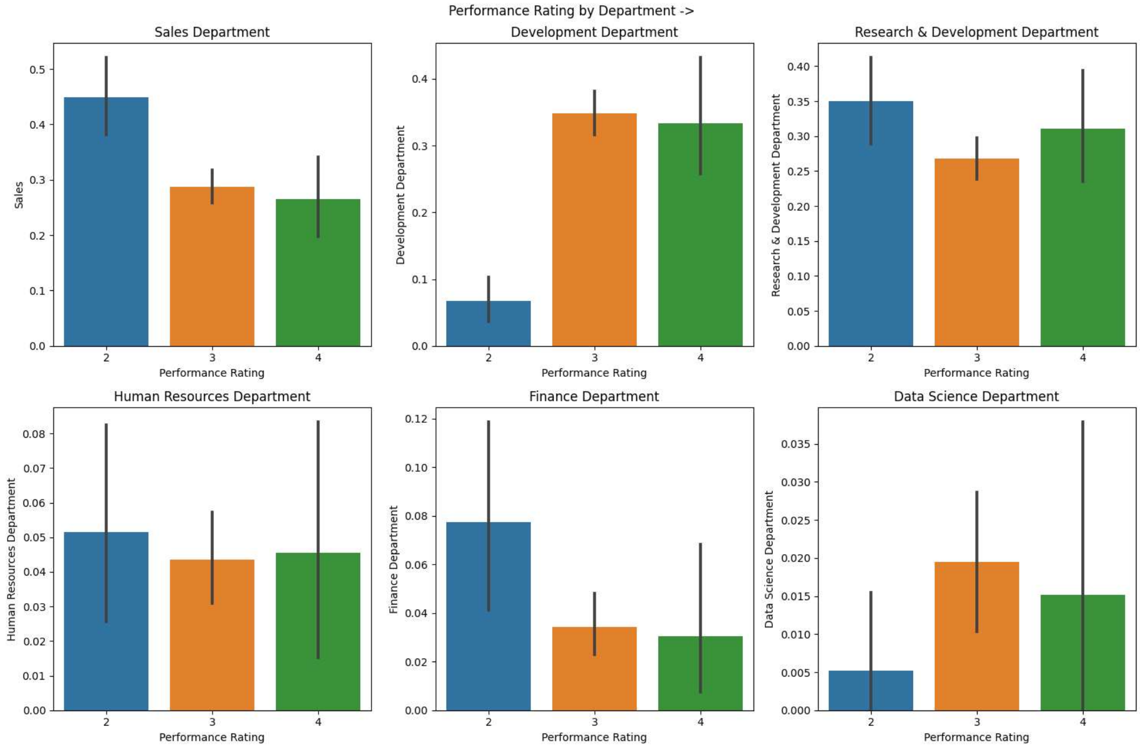
```
EmpDepartment           PerformanceRating
Data Science            3                     17
                        4                      2
                        2                      1
Development             3                    304
                        4                     44
                        2                     13
Finance                 3                     30
                        2                     15
                        4                      4
Human Resources         3                     38
                        2                     10
                        4                      6
Research & Development  3                    234
                        2                     68
                        4                     41
Sales                   3                    251
                        2                     87
                        4                     35
Name: count, dtype: int64
```

In [70]:

```
# Creating a new dataframe(dummy Values) to analyze each department separately
department = pd.get_dummies(dept_per['EmpDepartment'])
performance = pd.DataFrame(dept_per['PerformanceRating'])
dept_rating = pd.concat([department,performance],axis=1)
```

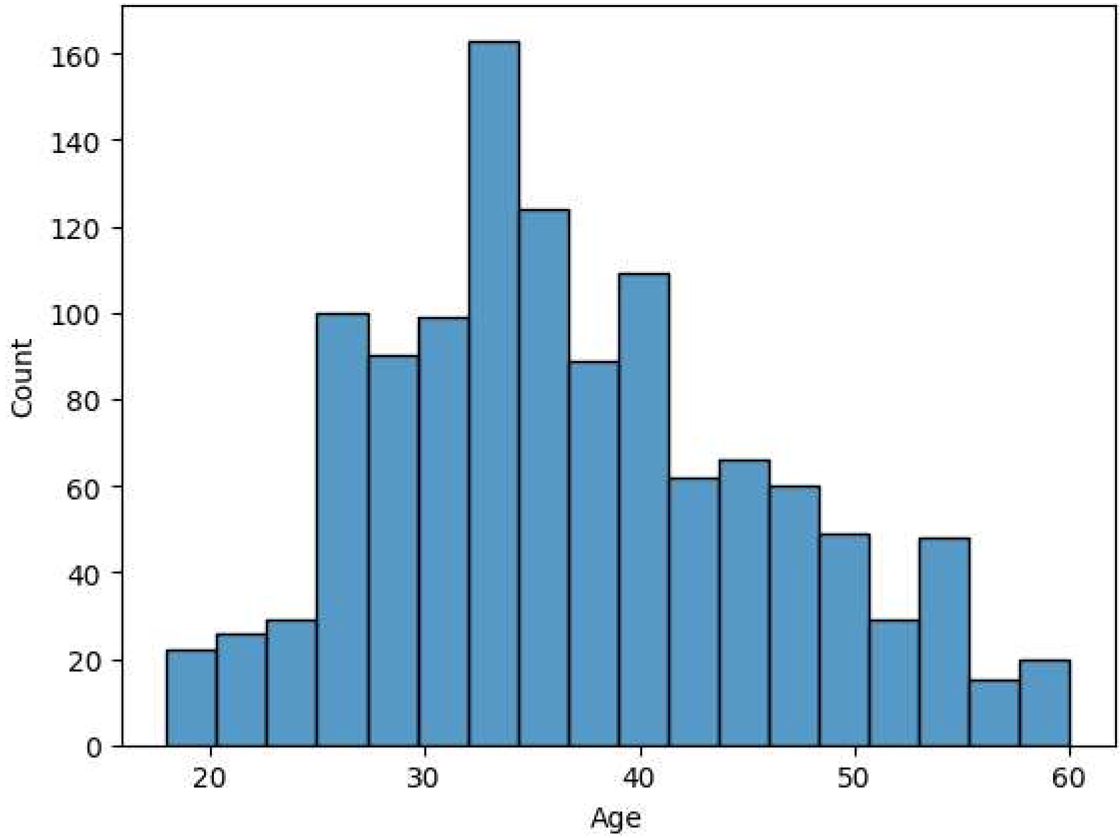
In [71]:

```
# Plotting a separate bar graph for performance of each department using seaborn

plt.figure(figsize=(15, 10)) # Set figure size
plt.suptitle('Performance Rating by Department ->')# Add overall title

plt.subplot(2, 3, 1)
sns.barplot(x='PerformanceRating', y='Sales', data=dept_rating)
plt.title('Sales Department')
plt.xlabel('Performance Rating')
plt.ylabel('Sales')

plt.subplot(2, 3, 2)
sns.barplot(x='PerformanceRating', y='Development', data=dept_rating)
plt.title('Development Department')
plt.xlabel('Performance Rating')
plt.ylabel('Development Department')

plt.subplot(2, 3, 3)
sns.barplot(x='PerformanceRating', y='Research & Development', data=dept_rating)
plt.title('Research & Development Department')
plt.xlabel('Performance Rating')
plt.ylabel('Research & Development Department')

plt.subplot(2, 3, 4)
sns.barplot(x='PerformanceRating', y='Human Resources', data=dept_rating)
plt.title('Human Resources Department')
plt.xlabel('Performance Rating')
plt.ylabel('Human Resources Department')

plt.subplot(2, 3, 5)
sns.barplot(x='PerformanceRating', y='Finance', data=dept_rating)
plt.title('Finance Department')
plt.xlabel('Performance Rating')
plt.ylabel('Finance Department')

plt.subplot(2, 3, 6)
sns.barplot(x='PerformanceRating', y='Data Science', data=dept_rating)
plt.title('Data Science Department')
plt.xlabel('Performance Rating')
plt.ylabel('Data Science Department')

plt.tight_layout() # Improve subplot spacing
plt.show() # Show the plot

```



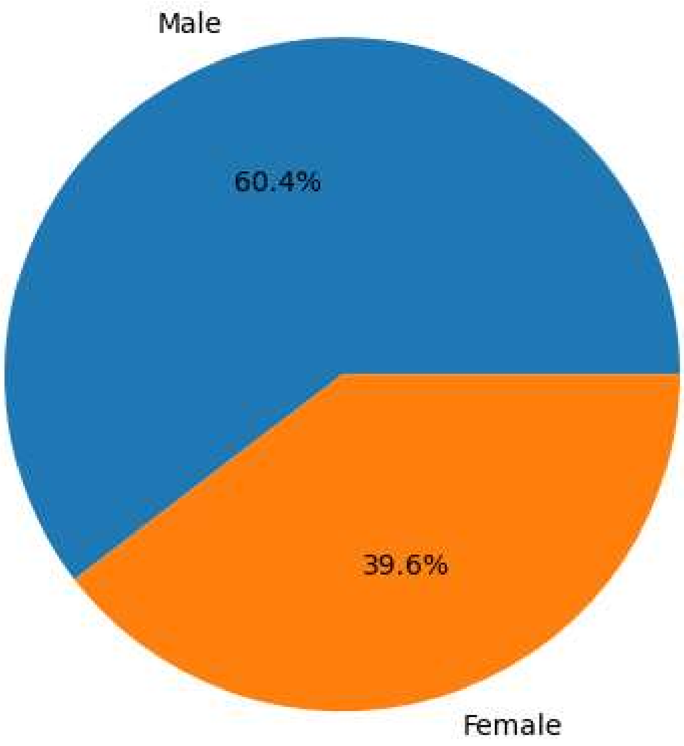
In [72]:

```
sns.histplot(data=data, x='Age')
plt.show()
```



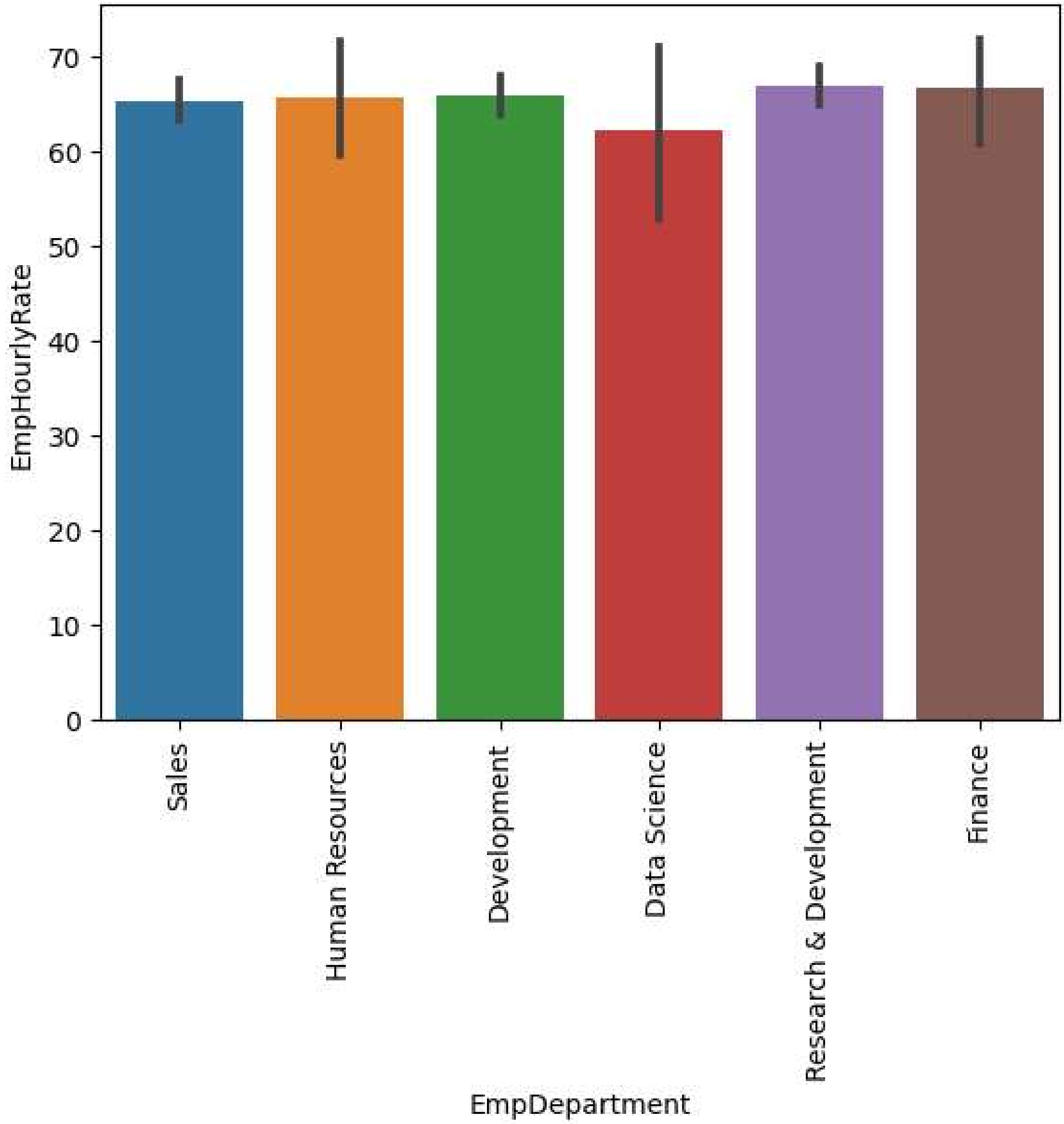
In [73]:

```
gender_counts = data['Gender'].value_counts()
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%')
plt.axis('equal')
plt.show()
```



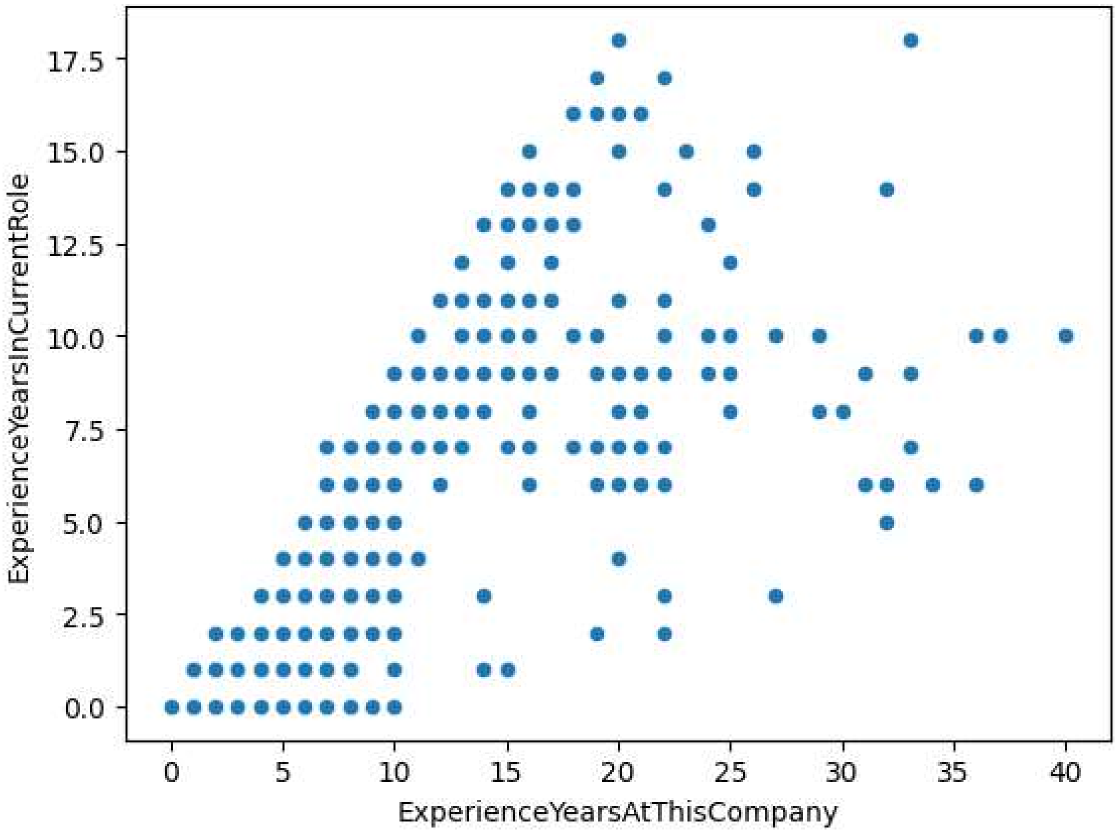
In [75]:

```
sns.barplot(data=data, x='EmpDepartment', y='EmpHourlyRate')
plt.xticks(rotation=90)
plt.show()
```



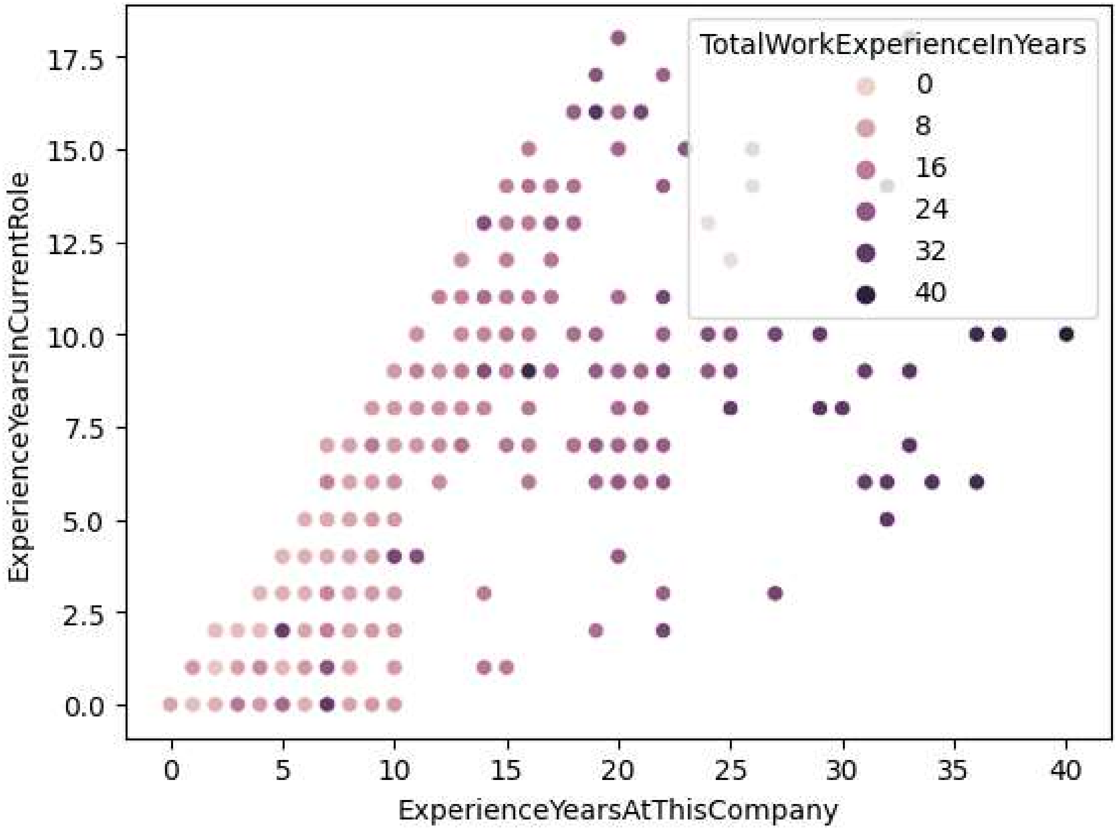
In [76]:

```
sns.scatterplot(data=data, x='ExperienceYearsAtThisCompany', y='ExperienceYearsIr
plt.show()
```



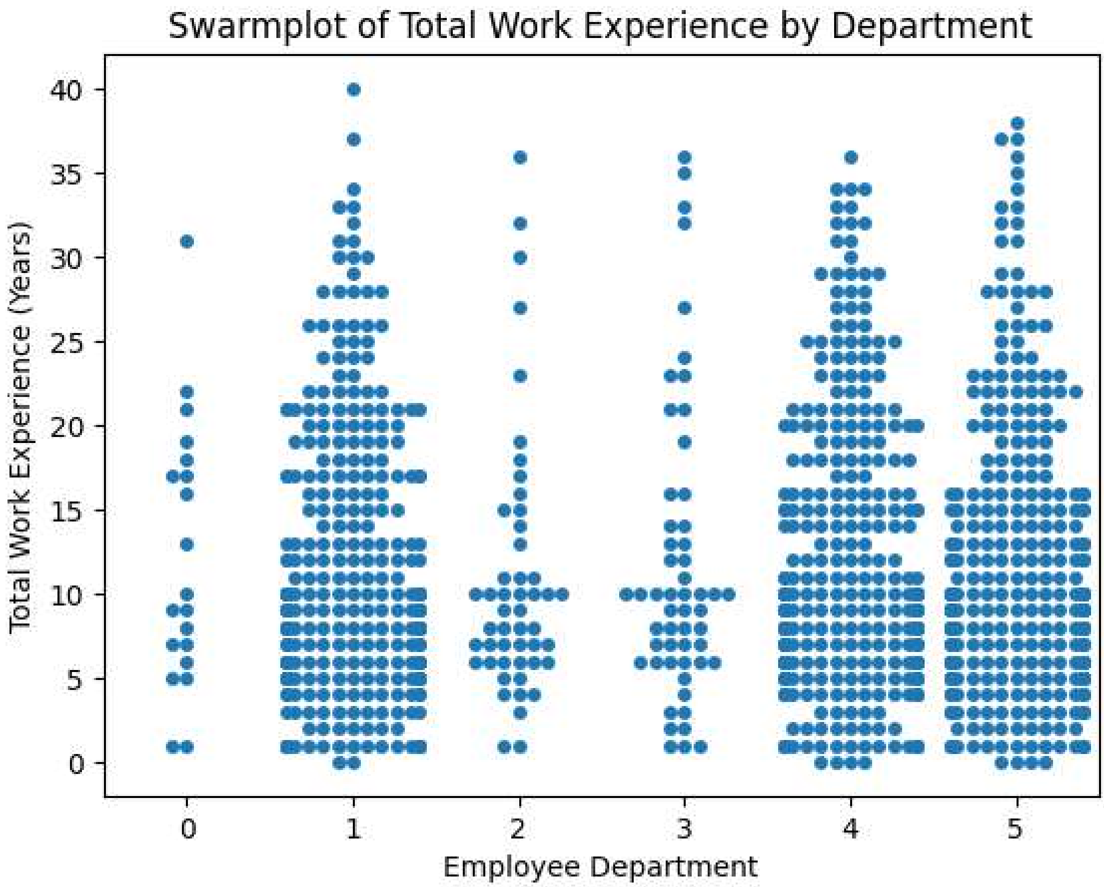
In [86]:

```
sns.scatterplot(data=data, x='ExperienceYearsAtThisCompany', y='ExperienceYearsIr
plt.show()
```



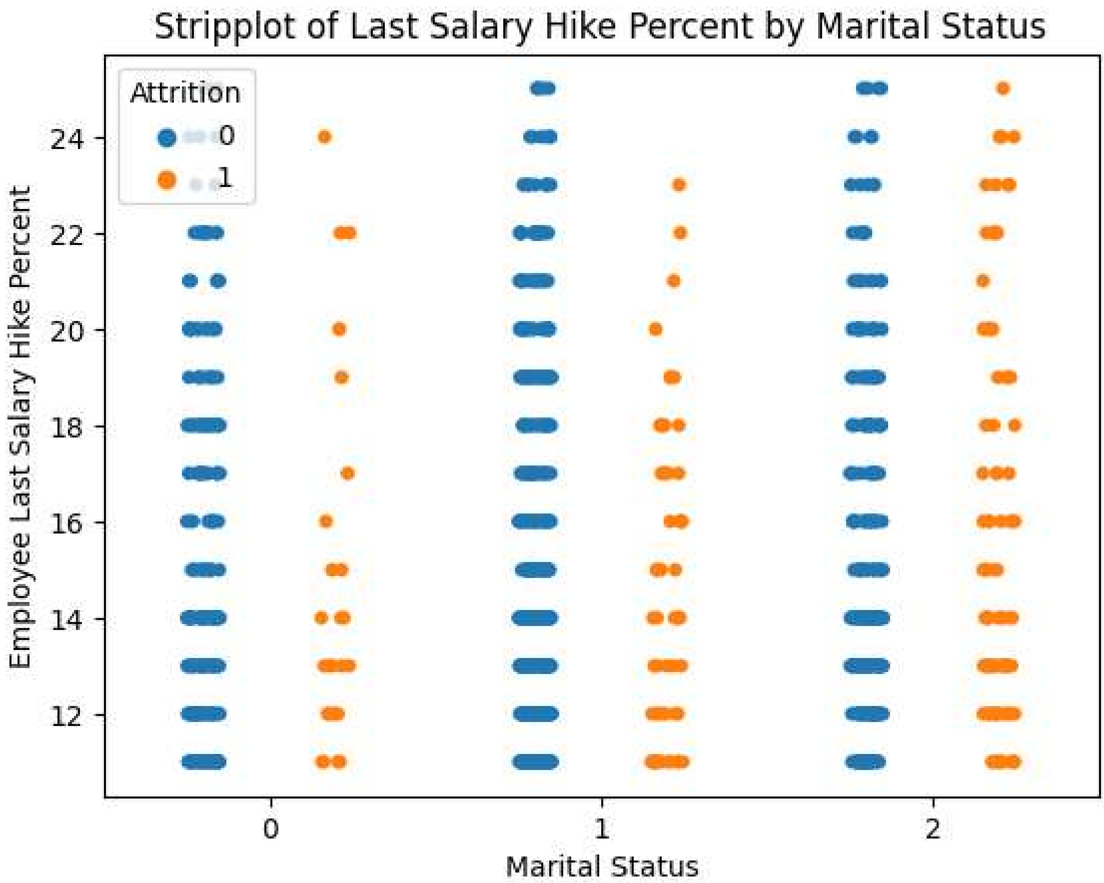
In [87]:

```
sns.swarmplot(data=data, x='EmpDepartment', y='TotalWorkExperienceInYears')
plt.xlabel('Employee Department')
plt.ylabel('Total Work Experience (Years)')
plt.title('Swarmplot of Total Work Experience by Department')
plt.show()
```



In [88]:

```
sns.stripplot(data=data, x='MaritalStatus', y='EmpLastSalaryHikePercent', hue='Attri
plt.xlabel('Marital Status')
plt.ylabel('Employee Last Salary Hike Percent')
plt.title('Stripplot of Last Salary Hike Percent by Marital Status')
plt.legend(title='Attrition')
plt.show()
```



In [89]:

```
sns.violinplot(data=data, x='EmpJobRole', y='EmpHourlyRate')
plt.xlabel('Employee Job Role')
plt.ylabel('Employee Hourly Rate')
plt.title('Violin Plot of Hourly Rate by Job Role')
plt.xticks(rotation=90)  # Rotate x-axis labels if needed
plt.show()
```

In [ ]:

```

```

DATA MODELLING

*Data modeling is the process of creating a mathematical or statistical representation of a real-world problem or system, while training involves the process of fitting a model to data by adjusting its parameters to minimize the difference between predicted and actual outcomes.**

In [37]:

```
#Printing Skewness & Kurtosis Of DaTa
print("Skewness: %f" %data['Age'].skew())
print("Kurtosis: %f" %data['Age'].kurt())
```

Skewness: 0.384145
Kurtosis: -0.431000

In [38]:

```
data.mode()
```

Out[38]:

	EmpNumber	Age	Gender	EducationBackground	MaritalStatus	EmpD
0	E1001000	34.0	Male	Life Sciences	Married	
1	E1001006	NaN	NaN	NaN	NaN	
2	E1001007	NaN	NaN	NaN	NaN	
3	E1001009	NaN	NaN	NaN	NaN	
4	E1001010	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	
1195	E100992	NaN	NaN	NaN	NaN	
1196	E100993	NaN	NaN	NaN	NaN	
1197	E100994	NaN	NaN	NaN	NaN	
1198	E100995	NaN	NaN	NaN	NaN	
1199	E100998	NaN	NaN	NaN	NaN	

1200 rows × 28 columns

In [77]:

```
# Encoding all the ordinal columns and creating a dummy variable for them to see if t
enc = LabelEncoder()
for i in (2,3,4,5,6,7,16,26):
    data.iloc[:,i] = enc.fit_transform(data.iloc[:,i])
data.head()
```

Out[77]:

	EmpNumber	Age	Gender	EducationBackground	MaritalStatus	EmpDepa
0	E1001000	32	1	2	2	
1	E1001006	47	1	2	2	
2	E1001007	40	1	1	1	
3	E1001009	41	1	0	0	
4	E1001010	60	1	2	2	

5 rows × 28 columns

In [78]:

```
# Dropping the first columns as it is of no use for analysis.
data.drop(['EmpNumber'],inplace=True,axis=1)
```

*CoLumn Dropped Done.**

In [79]:

```
#checking the droppped Colimn is present or not
data.head()
```

Out[79]:

	Age	Gender	EducationBackground	MaritalStatus	EmpDepartment	EmpJ
0	32	1	2	2	5	
1	47	1	2	2	5	
2	40	1	1	1	5	
3	41	1	0	0	3	
4	60	1	2	2	5	

5 rows × 27 columns

In [80]:

```
#Printing the Co-ReaLation Between Data
data.corr()
```

Out[80]:

	Age	Gender	EducationBackground
Age	1.000000	-0.040107	-0.055905
Gender	-0.040107	1.000000	0.009922
EducationBackground	-0.055905	0.009922	1.000000
MaritalStatus	-0.098368	-0.042169	-0.001097
EmpDepartment	-0.000104	-0.010925	-0.026874
EmpJobRole	-0.037665	0.011332	-0.012325
BusinessTravelFrequency	0.040579	-0.043608	0.012382
DistanceFromHome	0.020937	-0.001507	-0.013919
EmpEducationLevel	0.207313	-0.022960	-0.047978
EmpEnvironmentSatisfaction	0.013814	0.000033	0.045028
EmpHourlyRate	0.062867	0.002218	-0.030234
EmpJobInvolvement	0.027216	0.010949	-0.025505
EmpJobLevel	0.509139	-0.050685	-0.056338
EmpJobSatisfaction	-0.002436	0.024680	-0.030977
NumCompaniesWorked	0.284408	-0.036675	-0.032879
OverTime	0.051910	-0.038410	0.007046
EmpLastSalaryHikePercent	-0.006105	-0.005319	-0.009788
EmpRelationshipSatisfaction	0.049749	0.030707	0.005652
TotalWorkExperienceInYears	0.680886	-0.061055	-0.027929
TrainingTimesLastYear	-0.016053	-0.057654	0.051596
EmpWorkLifeBalance	-0.019563	0.015793	0.022890
ExperienceYearsAtThisCompany	0.318852	-0.030392	-0.009887
ExperienceYearsInCurrentRole	0.217163	-0.031823	-0.003215
YearsSinceLastPromotion	0.228199	-0.021575	0.014277
YearsWithCurrManager	0.205098	-0.036643	0.002767
Attrition	-0.189317	0.035758	0.027161
PerformanceRating	-0.040164	-0.001780	0.005607

27 rows × 27 columns

In [ ]:

```

```

In [43]:

```
# Here we have selected only the important columns
y = data.PerformanceRating
#X = data.iloc[:,0:-1] All predictors were selected it resulted in dropping of accuracy
X = data.iloc[:,[4,5,9,16,20,21,22,23,24]] # Taking only variables with correlation co
X.head()
```

Out[43]:

	EmpDepartment	EmpJobRole	EmpEnvironmentSatisfaction	EmpLastSala
0	5	13	4	
1	5	13	4	
2	5	13	4	
3	3	8	2	
4	5	13	1	

In [44]:

```
# Splitting into train and test for calculating the accuracy
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=1(
```

In [45]:

```
# Standardization technique is used
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [46]:

```
X_train.shape
```

Out[46]:

(840, 9)

In [47]:

```
X_test.shape
```

Out[47]:

(360, 9)

In [48]:

```
# Training the model
classifier_rfg=RandomForestClassifier(random_state=33,n_estimators=23)
parameters=[{'min_samples_split':[2,3,4,5],'criterion':['gini','entropy'],'min_samples
model_gridrf=GridSearchCV(estimator=classifier_rfg, param_grid=parameters, scorir
model_gridrf.fit(X_train,y_train)
```

Out[48]:

```
GridSearchCV(estimator=RandomForestClassifier(n_estimators=23, rando
m_state=33),
             param_grid=[{'criterion': ['gini', 'entropy'],
                          'min_samples_leaf': [1, 2, 3],
                          'min_samples_split': [2, 3, 4, 5]}],
             scoring='accuracy')
```

In [49]:

```
model_gridrf.best_params_
```

Out[49]:

```
{'criterion': 'entropy', 'min_samples_leaf': 1, 'min_samples_split': 4}
```

In [50]:

```
# Predicting the model
y_predict_rf = model_gridrf.predict(X_test)
```

In [51]:

```
# Finding accuracy, precision, recall and confusion matrix
print(accuracy_score(y_test,y_predict_rf))
print(classification_report(y_test,y_predict_rf))
```

```
0.9333333333333333
              precision    recall  f1-score   support

           2       0.90      0.89      0.90        63
           3       0.95      0.97      0.96       264
           4       0.83      0.76      0.79        33

    accuracy                           0.93       360
   macro avg       0.90      0.87      0.88       360
weighted avg       0.93      0.93      0.93       360
```

In [52]:

```
confusion_matrix(y_test,y_predict_rf)
```

Out[52]:

```
array([[ 56,   7,   0],
       [  4, 255,   5],
       [  2,   6,  25]], dtype=int64)
```

From the above calculation, it can be concluded that this model has 93.05% Accuracy.

The features that are positively correlated are:

- Environment Satisfaction
- Last Salary Hike Percent
- Worklife Balance

*This means that if these factors increases, Performance Rating will increase.**

On the other hand, the features that are negatively correlated are:

- Years Since Last Promotion
- Experience Years at this Company
- Experience years in Current Role
- Years with Current Manager.

*This means that if these factors increases, Performance Rating will go down.**

Conclusion: The company should provide a better environment as it increases the performance drastically.The company should increase the salary of the employee from time to time and help them maintain a worklife balance, shuffling the manager from time to time will also affect performance

In [ ]:

```

```