1. Importing Packages & loading the dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.rcParams.update({'figure.figsize': (12, 8)})
plt.rcParams.update({'font.size': 14})
```

```
data = pd.read_excel('INX_Future_Inc_Employee_Performance_CDS_Project2_Data_V1.8.xls') #loa
```

```
data.head()
```

	EmpNumber	Age	Gender	EducationBackground	MaritalStatus	EmpDepartment	EmpJobRol
0	E1001000	32	Male	Marketing	Single	Sales	Sale Executiv
1	E1001006	47	Male	Marketing	Single	Sales	Sale Executiv
2	E1001007	40	Male	Life Sciences	Married	Sales	Sale Executiv
3	E1001009	41	Male	Human Resources	Divorced	Human Resources	Manage
4	E1001010	60	Male	Marketing	Single	Sales	Sale Executiv

```
data.shape
```

```
(1200, 28)
```

```
data.columns
```

```
Index(['EmpNumber', 'Age', 'Gender', 'EducationBackground', 'MaritalStatus',
       'EmpDepartment', 'EmpJobRole', 'BusinessTravelFrequency',
       'DistanceFromHome', 'EmpEducationLevel', 'EmpEnvironmentSatisfaction',
       'EmpHourlyRate', 'EmpJobInvolvement', 'EmpJobLevel',
       'EmpJobSatisfaction', 'NumCompaniesWorked', 'OverTime',
       'EmpLastSalaryHikePercent', 'EmpRelationshipSatisfaction',
       'TotalWorkExperienceInYears', 'TrainingTimesLastYear',
       'EmpWorkLifeBalance', 'ExperienceYearsAtThisCompany',
       'ExperienceYearsInCurrentRole', 'YearsSinceLastPromotion',
```

```
       'YearsWithCurrManager', 'Attrition', 'PerformanceRating'],
      dtype='object')
```

2. EDA

```
data.isnull().sum() #checking for nan values
```

```
EmpNumber                     0
Age                           0
Gender                        0
EducationBackground           0
MaritalStatus                 0
EmpDepartment                 0
EmpJobRole                    0
BusinessTravelFrequency       0
DistanceFromHome              0
EmpEducationLevel             0
EmpEnvironmentSatisfaction    0
EmpHourlyRate                 0
EmpJobInvolvement             0
EmpJobLevel                   0
EmpJobSatisfaction            0
NumCompaniesWorked            0
OverTime                      0
EmpLastSalaryHikePercent      0
EmpRelationshipSatisfaction   0
TotalWorkExperienceInYears    0
TrainingTimesLastYear         0
EmpWorkLifeBalance            0
ExperienceYearsAtThisCompany  0
ExperienceYearsInCurrentRole  0
YearsSinceLastPromotion       0
YearsWithCurrManager          0
Attrition                     0
PerformanceRating             0
dtype: int64
```

```
data.dtypes
```

```
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
```

```
data.describe()
```

	Age	DistanceFromHome	EmpEducationLevel	EmpEnvironmentSatisfaction	EmpHou
count	1200.00	1200.00	1200.00	1200.00	
mean	36.92	9.17	2.89	2.72	
std	9.09	8.18	1.04	1.09	
min	18.00	1.00	1.00	1.00	
25%	30.00	2.00	2.00	2.00	
50%	36.00	7.00	3.00	3.00	
75%	43.00	14.00	4.00	4.00	
max	60.00	29.00	5.00	4.00	

```
data.nunique(axis=0)
```

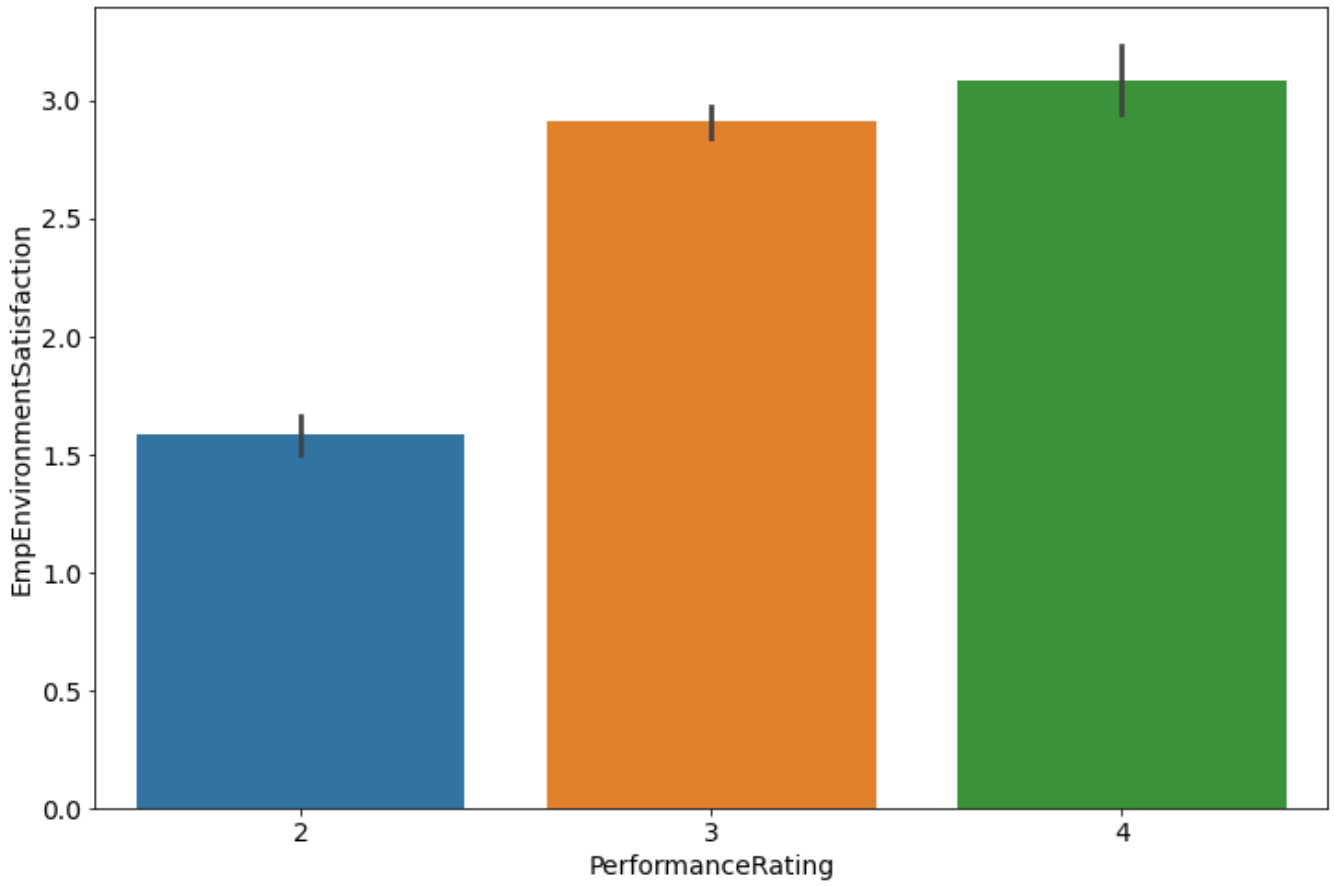
```
EmpNumber                    1200
Age                            43
Gender                          2
EducationBackground             6
MaritalStatus                   3
EmpDepartment                   6
EmpJobRole                     19
BusinessTravelFrequency         3
DistanceFromHome               29
EmpEducationLevel               5
EmpEnvironmentSatisfaction      4
EmpHourlyRate                  71
EmpJobInvolvement               4
EmpJobLevel                     5
EmpJobSatisfaction              4
NumCompaniesWorked             10
```

```
OverTime                          2
EmpLastSalaryHikePercent         15
EmpRelationshipSatisfaction       4
TotalWorkExperienceInYears       40
TrainingTimesLastYear             7
EmpWorkLifeBalance                4
ExperienceYearsAtThisCompany     37
ExperienceYearsInCurrentRole     19
YearsSinceLastPromotion          16
YearsWithCurrManager             18
Attrition                         2
PerformanceRating                 3
dtype: int64
```

3. Relationship between performance rating & other variables

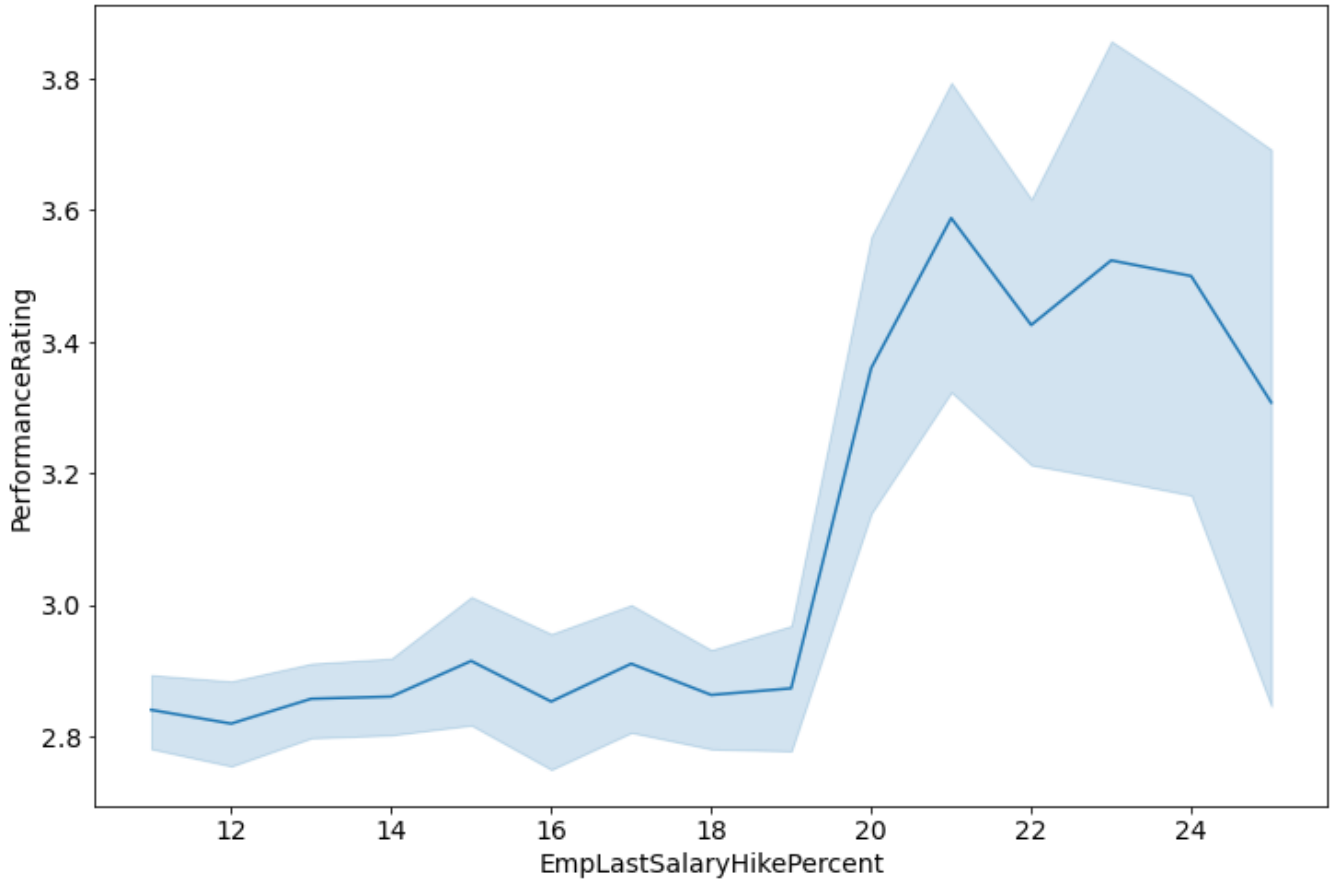
```
sns.barplot(y='EmpEnvironmentSatisfaction',x='PerformanceRating',data=data,estimator=np.mea
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed92595a90>
```




```
sns.lineplot(y='PerformanceRating',x='EmpLastSalaryHikePercent',data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed92f7ea58>
```

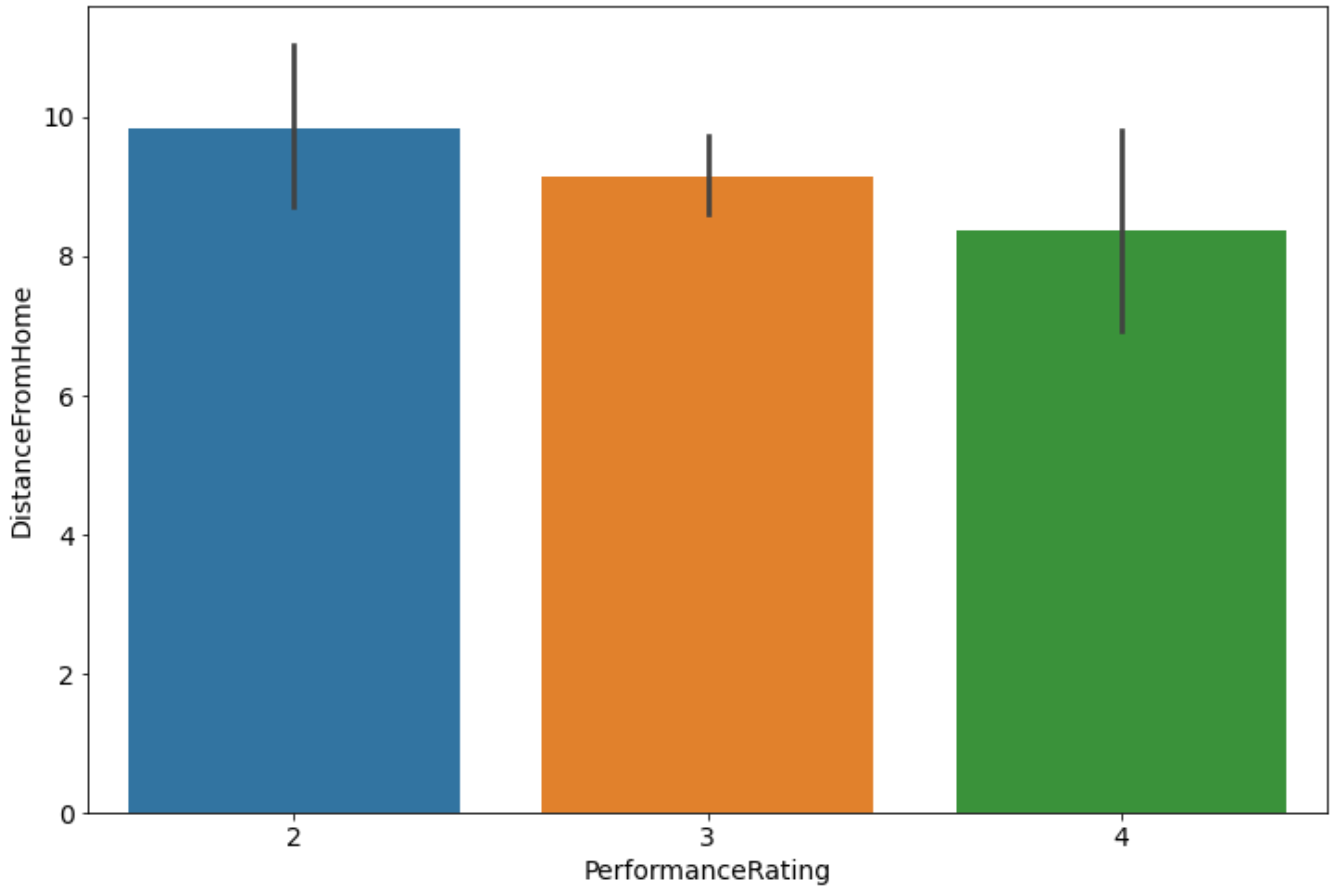



```
sns.barplot(y='YearsSinceLastPromotion',x='PerformanceRating',data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed92f7efd0>
```

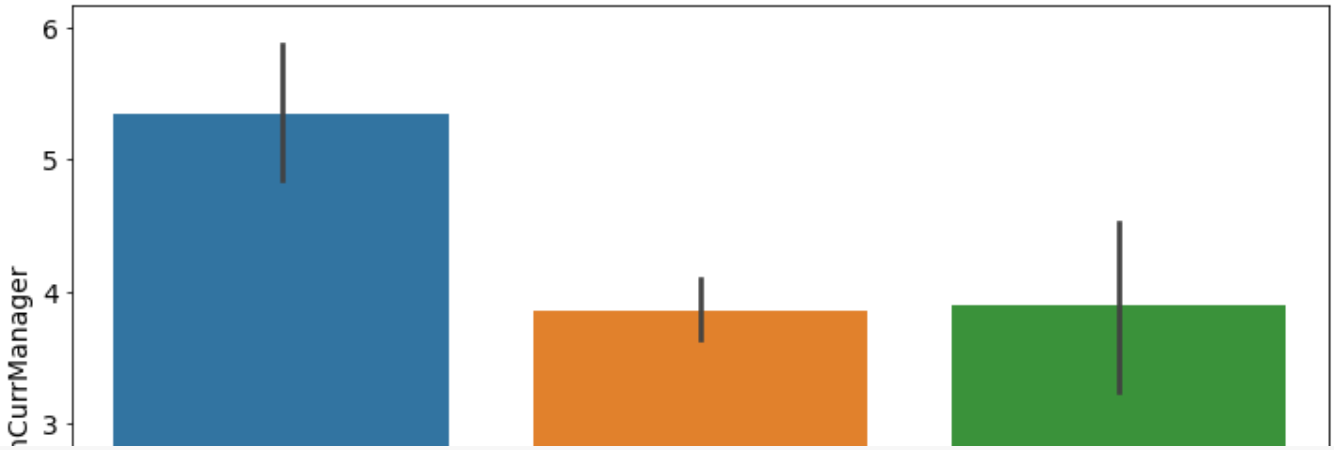

```
sns.barplot(y='DistanceFromHome',x='PerformanceRating',data=data,estimator=np.mean)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed92981d68>
```



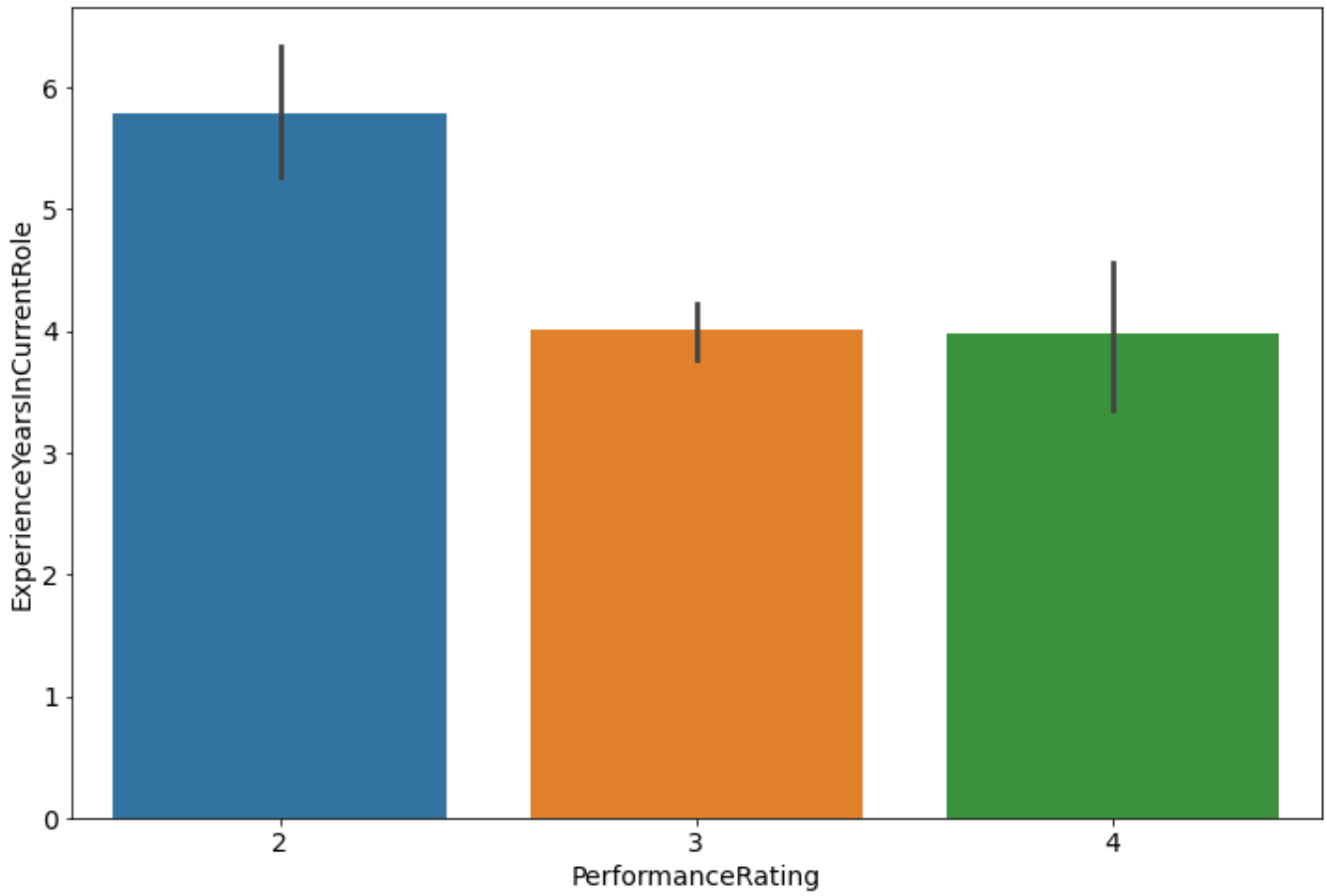

```
sns.barplot(y='YearsWithCurrManager',x='PerformanceRating',data=data,estimator=np.mean)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed929d6630>
```



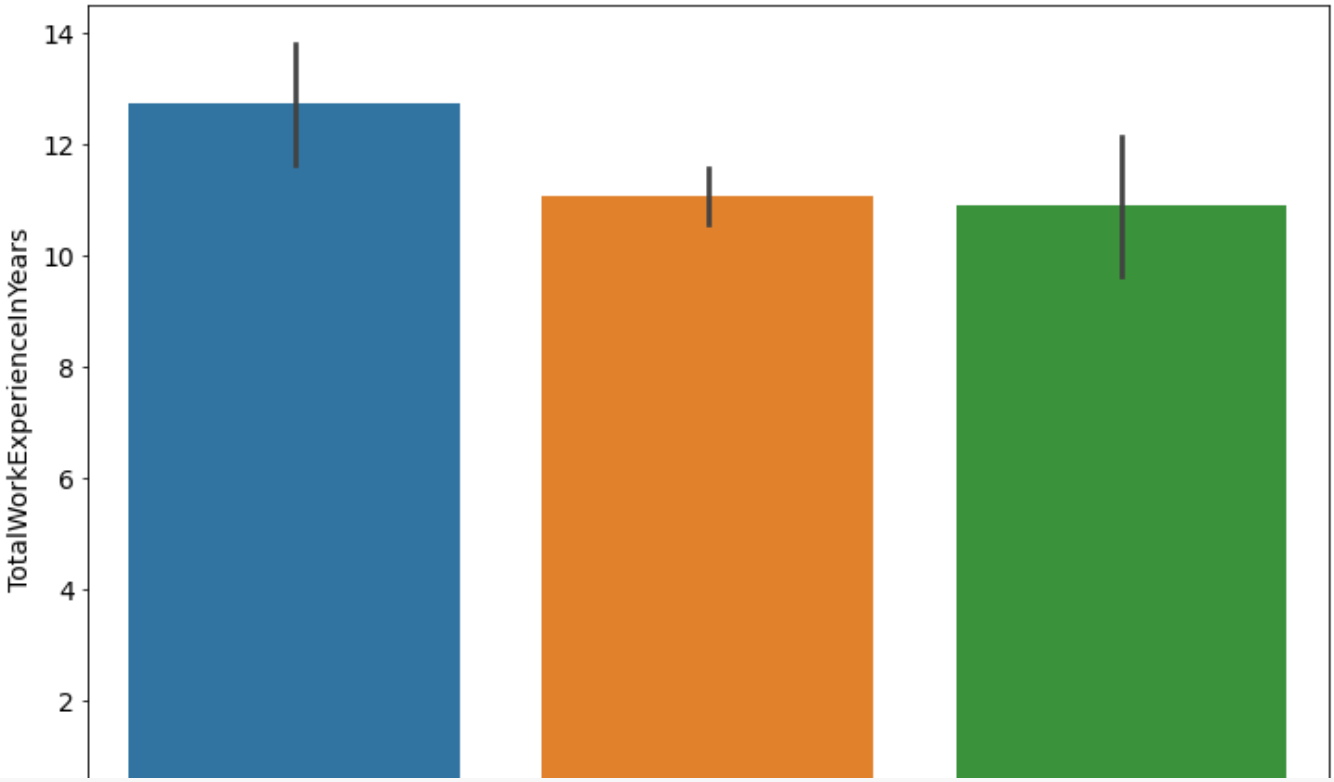

```
sns.barplot(y='ExperienceYearsInCurrentRole',x='PerformanceRating',data=data,estimator=np.m
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed9b0456d8>
```



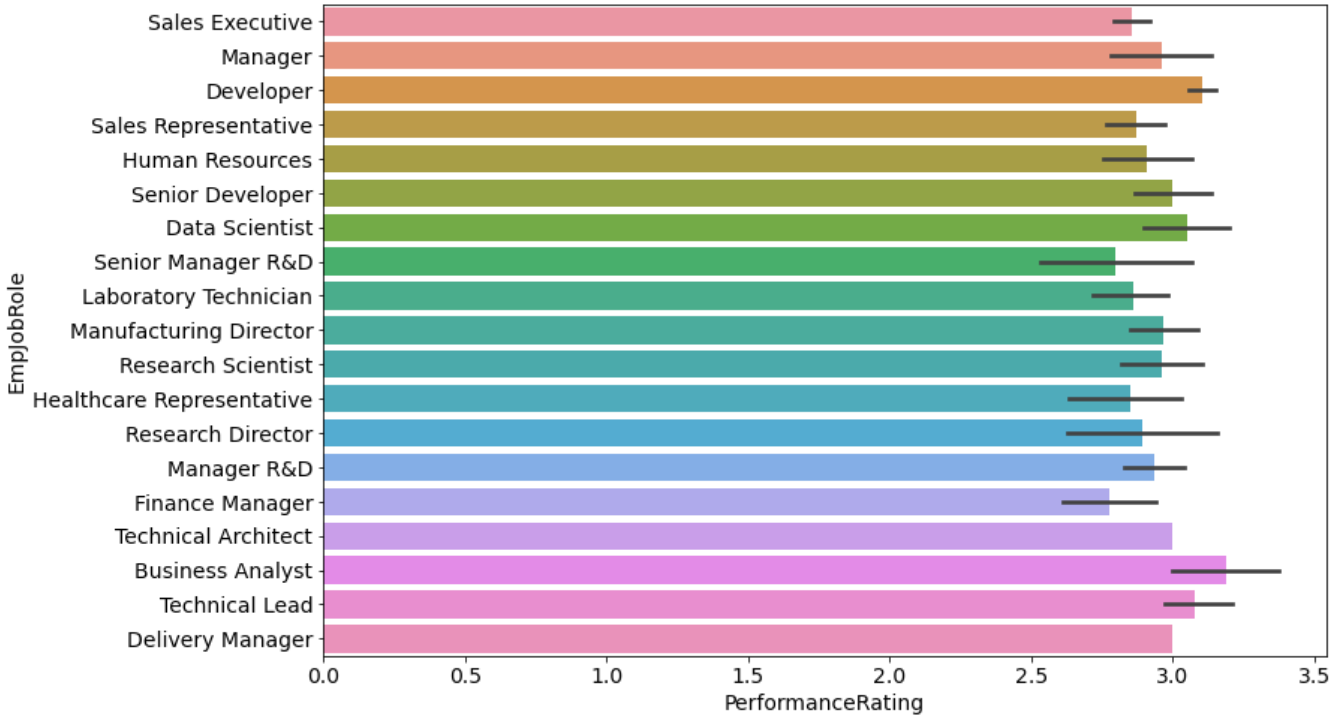

```
sns.barplot(y='TotalWorkExperienceInYears',x='PerformanceRating',data=data,estimator=np.mea
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed9b0f3a90>
```



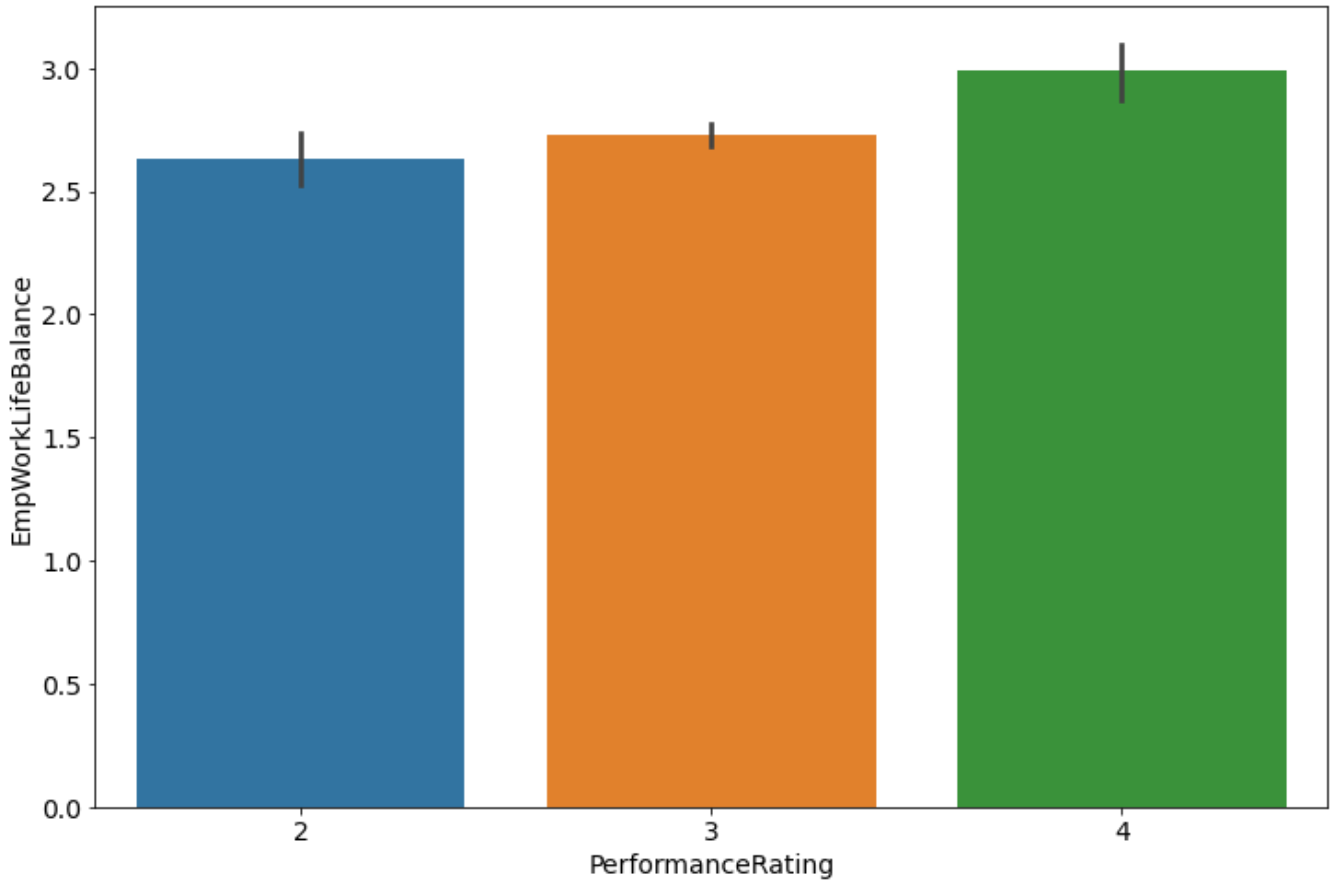

```
sns.barplot(y='EmpJobRole',x='PerformanceRating',data=data,estimator=np.mean)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed9324dd68>
```



```
sns.barplot(y='EmpWorkLifeBalance',x='PerformanceRating',data=data,estimator=np.mean)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed9b0ecc88>
```



3. feature Engineerring

```
idx = features = data.columns.values[0:28]
for df in [data]:
    df['sum'] = df[idx].sum(axis=1)
    df['min'] = df[idx].min(axis=1)
    df['max'] = df[idx].max(axis=1)
    df['mean'] = df[idx].mean(axis=1)
    df['std'] = df[idx].std(axis=1)
    df['skew'] = df[idx].skew(axis=1)
    df['kurt'] = df[idx].kurtosis(axis=1)
    df['med'] = df[idx].median(axis=1)
```

```
data.head(5)
```

	EmpNumber	Age	Gender	EducationBackground	MaritalStatus	EmpDepartment	EmpJobRol
0	E1001000	32	Male	Marketing	Single	Sales	Sale Executiv
1	E1001006	47	Male	Marketing	Single	Sales	Sale Executiv
2	E1001007	40	Male	Life Sciences	Married	Sales	Sale Executiv
3	E1001009	41	Male	Human Resources	Divorced	Human Resources	Manage
4	E1001010	60	Male	Marketing	Single	Sales	Sale

4. Modeling

```
# importing packages
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, r2_sco
```

```
# allocating x * y
X = data.loc[:, : 'Attrition']
Y = data.PerformanceRating
X.head()
```

	EmpNumber	Age	Gender	EducationBackground	MaritalStatus	EmpDepartment	EmpJobRol
0	E1001000	32	Male	Marketing	Single	Sales	Sale Executiv
1	E1001006	47	Male	Marketing	Single	Sales	Sale Executiv
2	E1001007	40	Male	Life Sciences	Married	Sales	Sale Executiv
3	E1001009	41	Male	Human Resources	Divorced	Human Resources	Manage
4	E1001010	60	Male	Marketing	Single	Sales	Sale Executiv

```
# Label Encoding
enc= LabelEncoder()
for i in (0,2,3,4,5,6,7,16,26):
    X.iloc[:,i] = enc.fit_transform(X.iloc[:,i])
```

```
X.head()
```

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

```
X.dtypes
```

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

```
# Splitting the data into test and train for calculating accuracy
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.25,random_state=10)
```

```
X_train_shape = print(X_train.shape)
X_test_shape = print(X_test.shape)
Y_train_shape = print(Y_train.shape)
Y_test_shape = print(Y_test.shape)
```

```
(900, 27)
```

```
(300, 27)
(900,)
(300,)
```

1. Decision Tree model

```
# Training the model
from sklearn.tree import DecisionTreeClassifier
model_dtree=DecisionTreeClassifier()
model_dtree.fit(X_train,Y_train)
y_predict_dtree = model_dtree.predict(X_test)
print(accuracy_score(Y_test,y_predict_dtree))
print(classification_report(Y_test,y_predict_dtree))
```

```
0.8933333333333333
              precision    recall  f1-score   support

           2       0.81      0.85      0.83        54
           3       0.94      0.92      0.93       219
           4       0.74      0.74      0.74        27

    accuracy                           0.89       300
   macro avg       0.83      0.84      0.83       300
weighted avg       0.89      0.89      0.89       300
```

```
confusion_matrix(Y_test,y_predict_dtree)
```

```
array([[ 46,   8,   0],
       [ 10, 202,   7],
       [  1,   6,  20]])
```

```
Y_test[0:5]
```

```
123     3
1036    3
1034    3
426     2
726     3
Name: PerformanceRating, dtype: int64
```

```
y_predict_dtree[0:5]
```

```
array([3, 3, 3, 3, 3])
```

2.Random forest classifier

```
#trainig the model
from sklearn.ensemble import RandomForestClassifier
model_rf=RandomForestClassifier()
model_rf.fit(X_train,Y_train)
y_predict_rf = model_rf.predict(X_test)
print(accuracy_score(Y_test,y_predict_rf))
print(classification_report(Y_test,y_predict_rf))
```

```
0.9433333333333334
              precision    recall  f1-score   support

           2       0.96      0.91      0.93        54
           3       0.95      0.97      0.96       219
           4       0.84      0.78      0.81        27

    accuracy                           0.94       300
   macro avg       0.92      0.89      0.90       300
weighted avg       0.94      0.94      0.94       300
```

```
confusion_matrix(Y_test,y_predict_rf)
```

```
array([[ 49,   5,   0],
       [  2, 213,   4],
       [  0,   6,  21]])
```

```
Y_test[0:5] # checking the tested & predicted value
```

```
123     3
1036    3
1034    3
426     2
726     3
Name: PerformanceRating, dtype: int64
```

```
y_predict_rf[0:5] # predicted values
```

```
array([3, 3, 3, 2, 3])
```

3. XGBOOST

```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in /usr/local/lib/python3.6/dist-packages (0.90)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from xgb
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from xgb
```

```
from xgboost import XGBClassifier
```

```
#modeling the data
model_X = XGBClassifier()
model_X.fit(X_train,Y_train)
y_predict_X = model_X.predict(X_test)
print(accuracy_score(Y_test,y_predict_X))
print(classification_report(Y_test,y_predict_X))
```

```
0.9366666666666666
              precision    recall  f1-score   support

           2       0.94      0.91      0.92        54
           3       0.95      0.96      0.96       219
           4       0.81      0.78      0.79        27

    accuracy                           0.94       300
   macro avg       0.90      0.88      0.89       300
weighted avg       0.94      0.94      0.94       300
```

```
confusion_matrix(Y_test,y_predict_X)
```

```
array([[ 49,   5,   0],
       [  3, 211,   5],
       [  0,   6,  21]])
```

```
Y_test[0:11] # checking the tested & predicted value
```

```
123     3
1036    3
1034    3
426     2
726     3
508     3
483     3
67      3
598     3
840     3
87      3
Name: PerformanceRating, dtype: int64
```

```
y_predict_X[0:11]
```

```
array([3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3])
```

5. Top 3 Important Factors effecting employee performance

```
!pip install shap
```

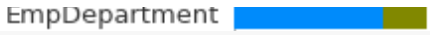
```
Requirement already satisfied: shap in /usr/local/lib/python3.6/dist-packages (0.37.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from sha
Requirement already satisfied: slicer==0.0.3 in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from sha
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packages (from sh
Requirement already satisfied: numba in /usr/local/lib/python3.6/dist-packages (from sha
Requirement already satisfied: tqdm>4.25.0 in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.6/dist-p
Requirement already satisfied: llvmlite<0.32.0,>=0.31.0dev0 in /usr/local/lib/python3.6/
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from
```

```
import shap
```

```
#modeling
explainer = shap.TreeExplainer(model_X)
shap_values = explainer.shap_values(X_test)
```

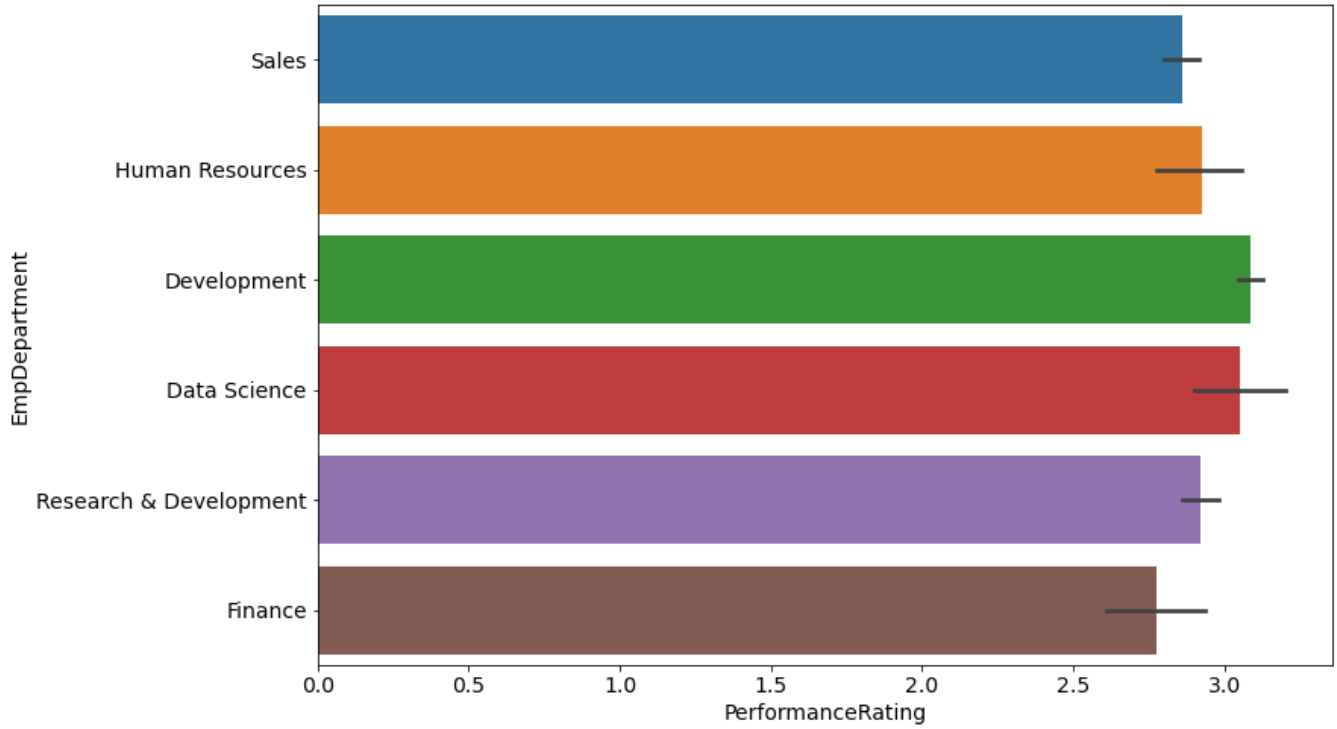
```
shap.summary_plot(shap_values, X_test, plot_type="bar")
```

6. Departmeant Wise Performance



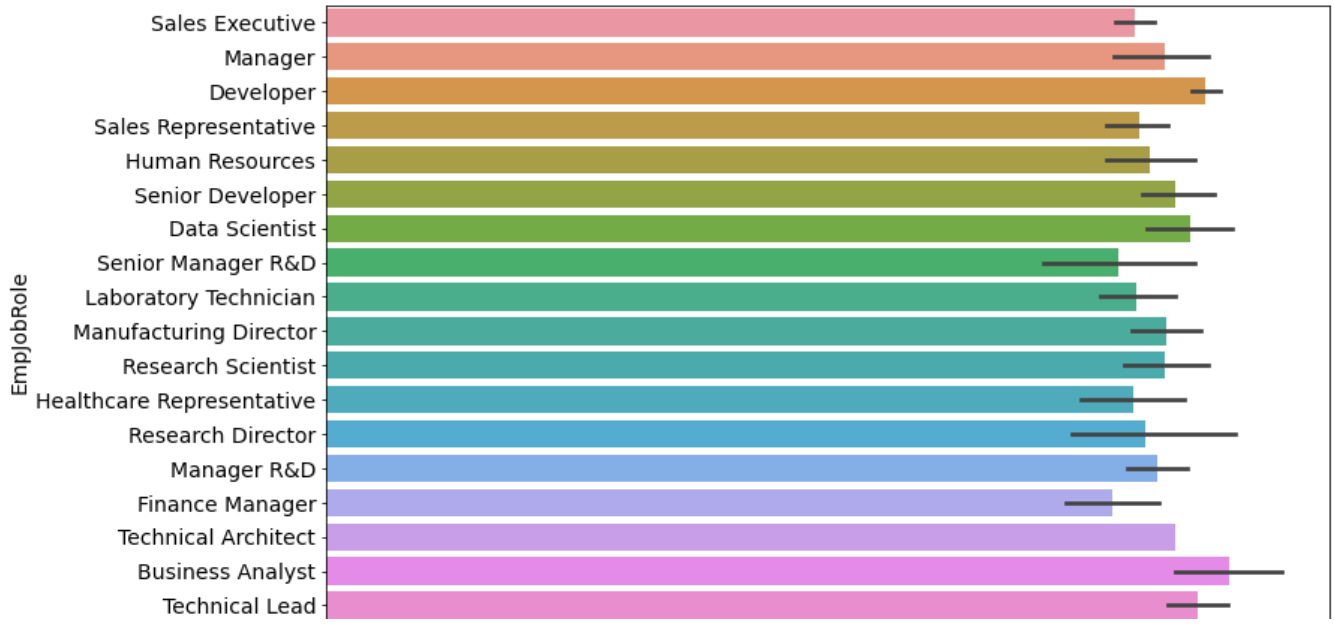

```
sns.barplot(y='EmpDepartment',x='PerformanceRating',data=data,estimator=np.mean)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed9266dcf8>
```




```
sns.barplot(y='EmpJobRole',x='PerformanceRating',data=data,estimator=np.mean)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed9253e3c8>
```



Recommendations to improve the employee performance

1. Empployee environment satisfaction should be improve to achieve high performance.

2. Have 20% or more hike in salary to get high performance of employee.

3. Promotion should be done for every 2 years to get better performance.(for intermediate level employee)

4. Work life balance should be considered too.

5. Sales & finance department should be on more focus to improve employee performance.