In [123...
```python
import pandas as pd
import numpy as np
import warnings; warnings.simplefilter('ignore')
```

In [124...
```python
df = pd.read_csv("employee.csv")
df.head()
```

Out[124...

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	Em
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	

5 rows × 35 columns

In [125...
```python
df.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 [126...
```python
df.describe()
```

Out[126...

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	Environ
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	

8 rows × 26 columns

In [127...
```python
df.isnull().sum()
```

Out[127...
```
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
```

In [128...
```python
df.Education.value_counts()
```

Out[128...
```
3    572
4    398
2    282
1    170
5     48
Name: Education, dtype: int64
```

In [129...
```python
df.EnvironmentSatisfaction.value_counts()
```

Out[129...
```
3    453
4    446
2    287
1    284
Name: EnvironmentSatisfaction, dtype: int64
```

In [130...
```python
df.JobInvolvement.value_counts()
```

Out[130...
```
3    868
2    375
4    144
1     83
Name: JobInvolvement, dtype: int64
```

In [131...
```python
df.JobSatisfaction.value_counts()
```

Out[131...
```
4    459
3    442
1    289
2    280
Name: JobSatisfaction, dtype: int64
```

In [132...
```python
df.PerformanceRating.value_counts()
```

Out[132...
```
3    1244
4     226
Name: PerformanceRating, dtype: int64
```

In [133...
```python
df.RelationshipSatisfaction.value_counts()
```

Out[133...
```
3    459
4    432
2    303
1    276
Name: RelationshipSatisfaction, dtype: int64
```

In [134...
```python
df.WorkLifeBalance.value_counts()
```

Out[134...
```
3    893
2    344
4    153
1     80
Name: WorkLifeBalance, dtype: int64
```

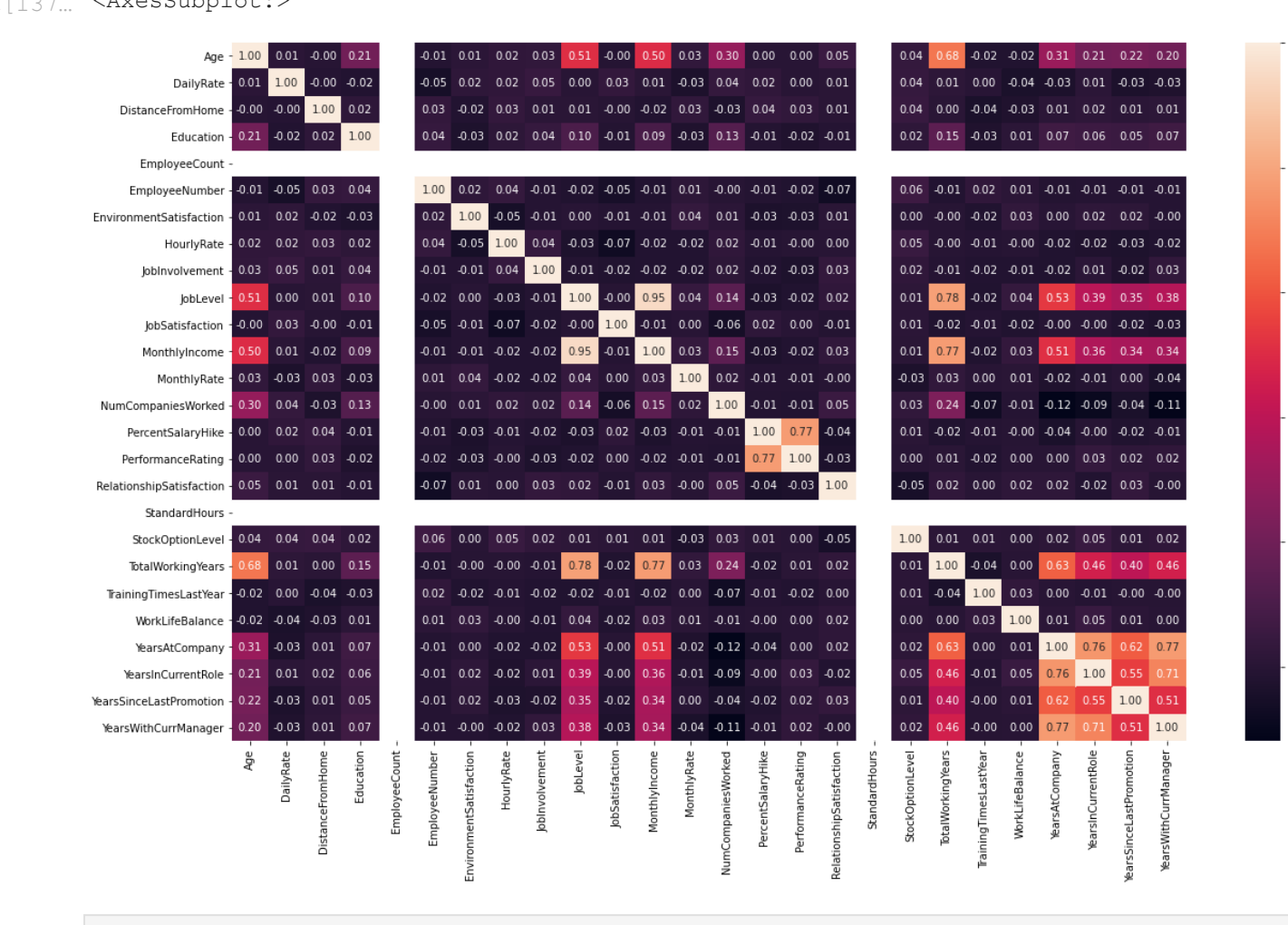
In [135...
```python
df.Attrition.value_counts()
```

Out[135...
```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

In [136...
```python
import seaborn as sns
import matplotlib.pyplot as plt
```

In [137...
```python
plt.figure(figsize=(20,12))
sns.heatmap(df.corr(), annot= True, fmt='.2f')
```

Out[137... <AxesSubplot:>

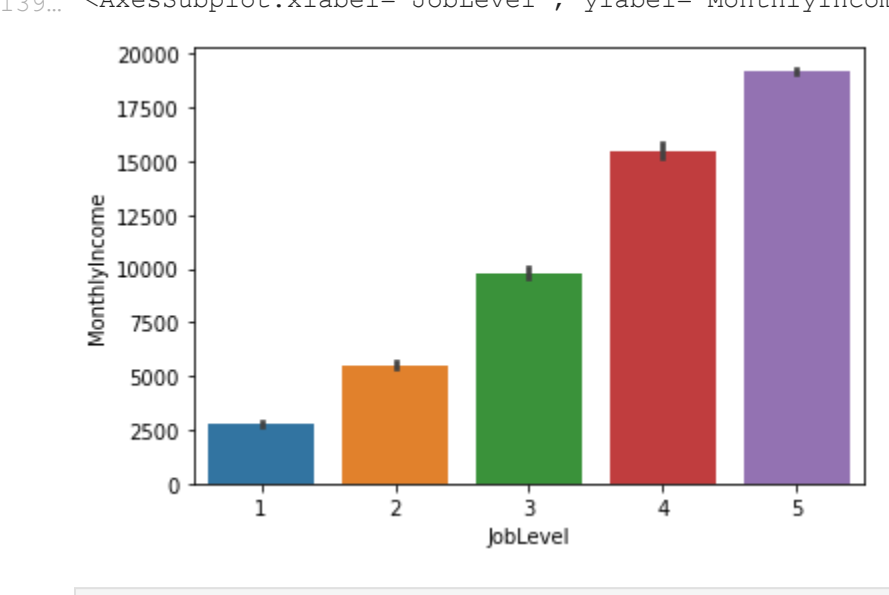


In [138...
```python
sns.scatterplot(x=df['MonthlyIncome'], y=df['TotalWorkingYears'], hue=df['Age'])
```

Out[138... <AxesSubplot:xlabel='MonthlyIncome', ylabel='TotalWorkingYears'>

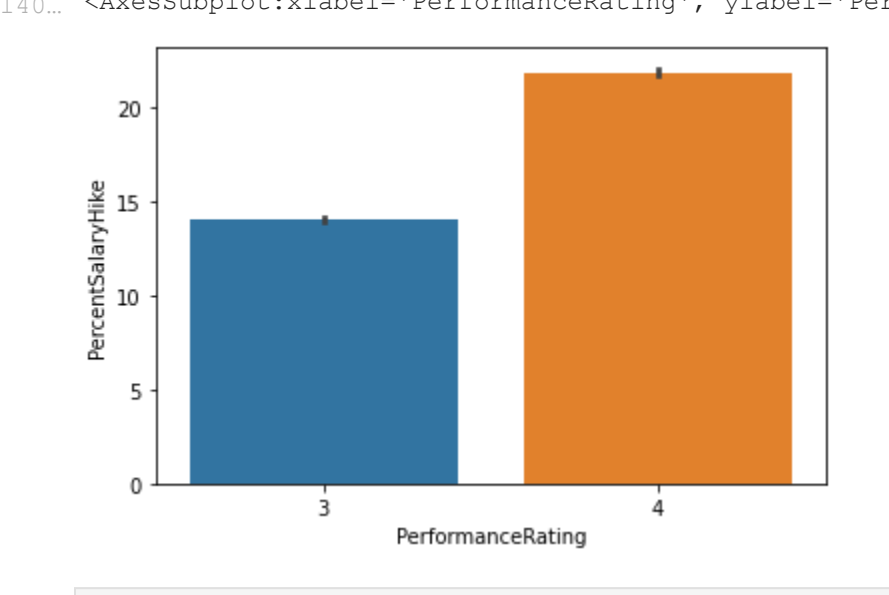
In [139...
```python
sns.barplot(y=df['MonthlyIncome'], x=df['JobLevel'])
```

Out[139... <AxesSubplot:xlabel='JobLevel', ylabel='MonthlyIncome'>



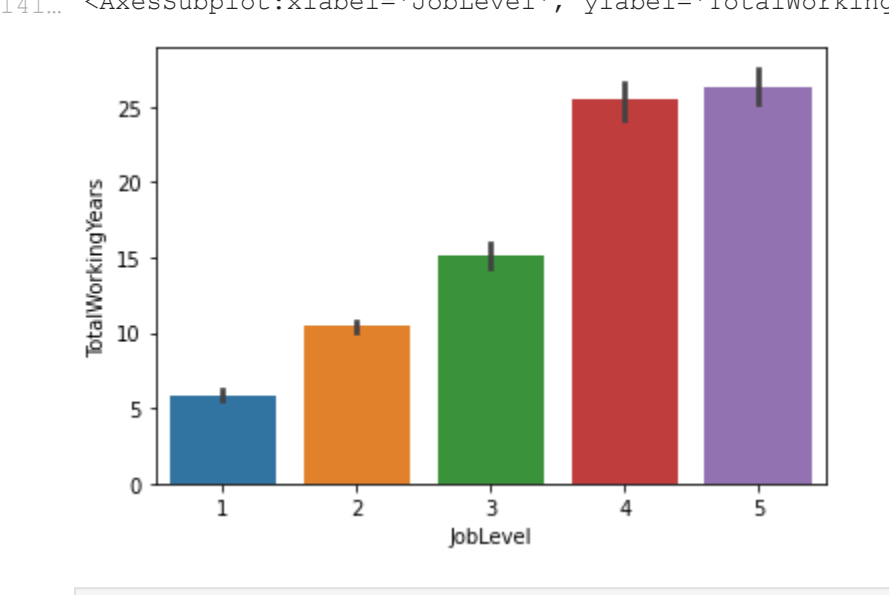
In [140...
```python
sns.barplot(y=df['PercentSalaryHike'], x=df['PerformanceRating'])
```

Out[140... <AxesSubplot:xlabel='PerformanceRating', ylabel='PercentSalaryHike'>



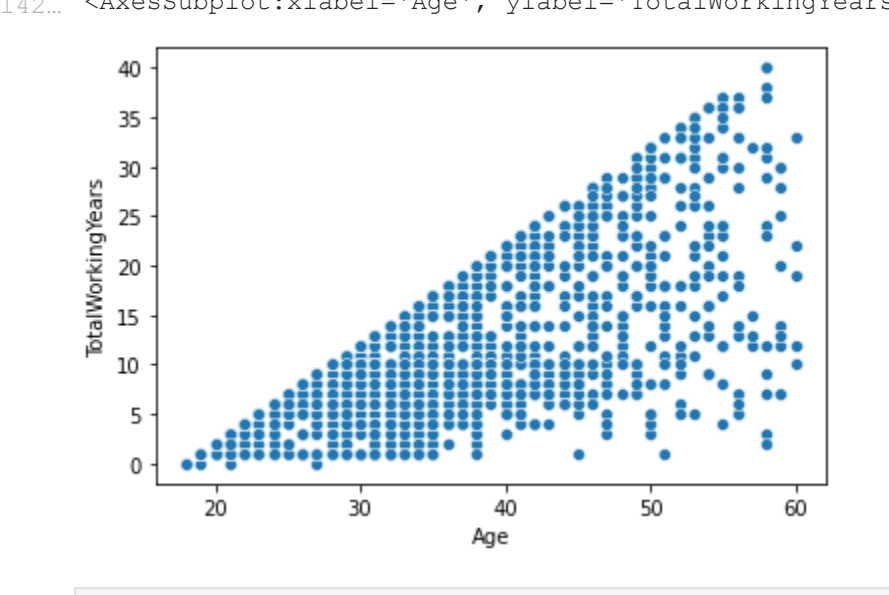
In [141...
```python
sns.barplot(y=df['TotalWorkingYears'], x=df['JobLevel'])
```

Out[141... <AxesSubplot:xlabel='JobLevel', ylabel='TotalWorkingYears'>



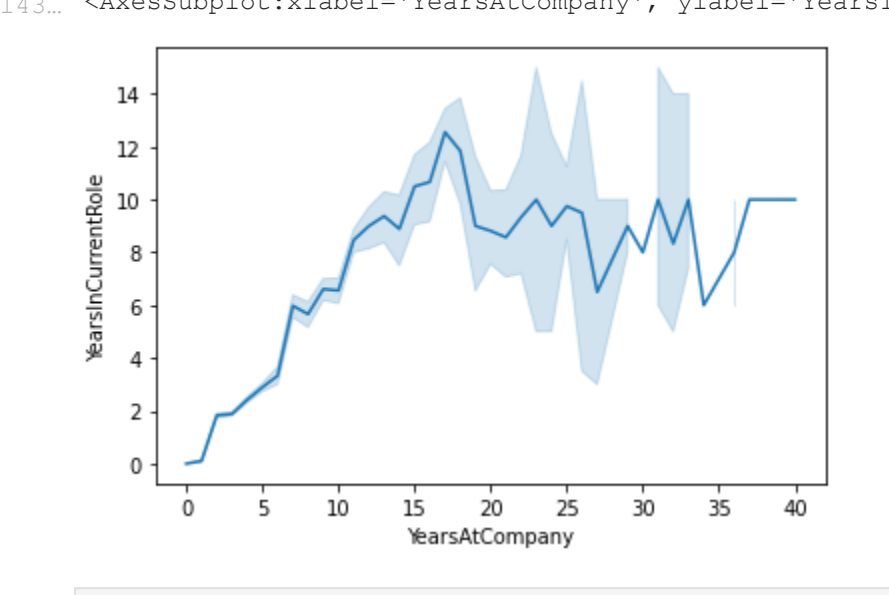
In [142...
```python
sns.scatterplot(x=df['Age'], y=df['TotalWorkingYears'])
```

Out[142... <AxesSubplot:xlabel='Age', ylabel='TotalWorkingYears'>



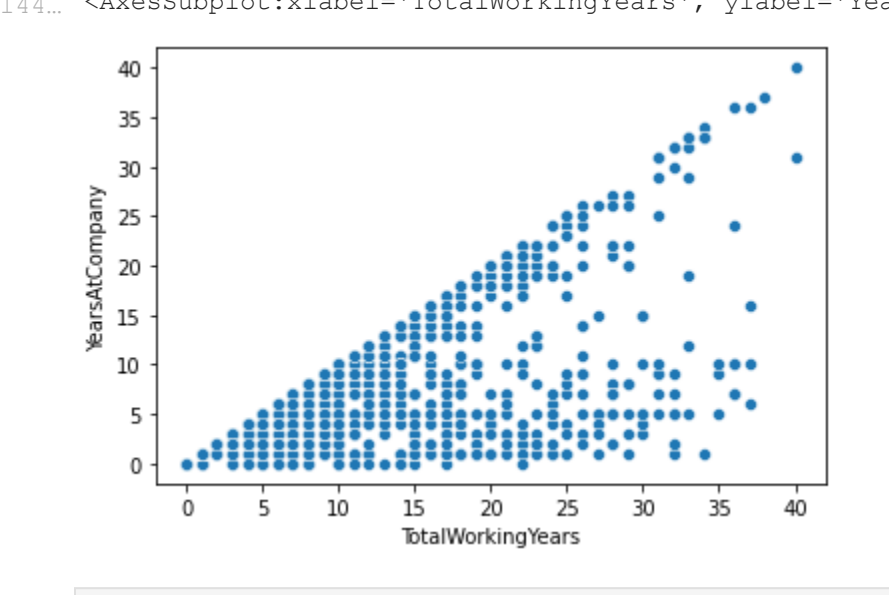
In [143...
```python
sns.lineplot(y=df['YearsInCurrentRole'], x=df['YearsAtCompany'])
```

Out[143... <AxesSubplot:xlabel='YearsAtCompany', ylabel='YearsInCurrentRole'>



In [144...
```python
sns.scatterplot(x=df['TotalWorkingYears'], y=df['YearsAtCompany'])
```

Out[144... <AxesSubplot:xlabel='TotalWorkingYears', ylabel='YearsAtCompany'>



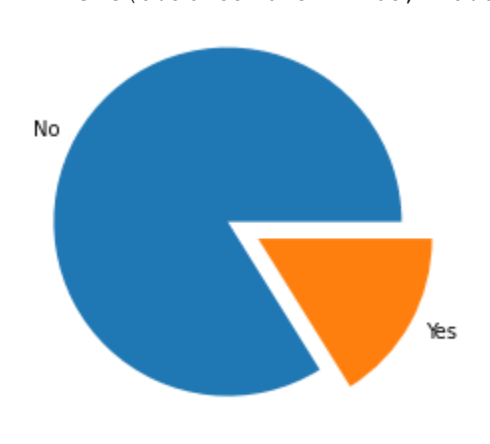
In [145...
```python
attrition_count = pd.DataFrame(df['Attrition'].value_counts())
attrition_count
```

Out[145...

	Attrition
No	1233
Yes	237

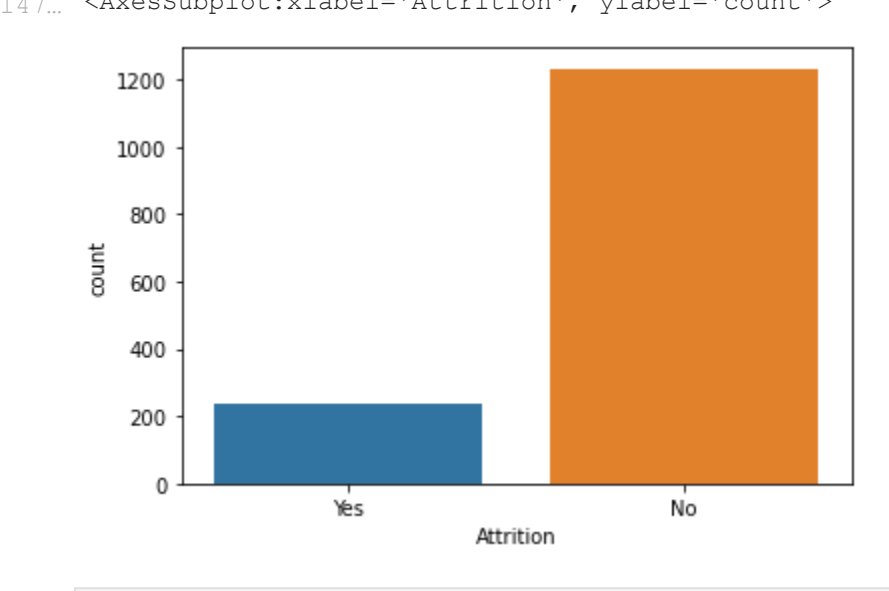
In [146...
```python
plt.pie(attrition_count['Attrition'], labels = ['No', 'Yes'], explode = (0.2,0))
```

Out[146...
```
([<matplotlib.patches.Wedge at 0x17d06ded070>,
  <matplotlib.patches.Wedge at 0x17d06ded580>],
 [Text(-1.136781068348268, 0.6306574368426737, 'No'),
  Text(0.961891673217765, -0.5336332157899547, 'Yes')])
```



In [147...
```python
sns.countplot(data=df, x='Attrition')
```

Out[147... <AxesSubplot:xlabel='Attrition', ylabel='count'>



In [148...
```python
df.drop(['EmployeeCount','EmployeeNumber'], axis=1, inplace=True)
```

In [149...
```python
attrition_dummies = pd.get_dummies(df['Attrition'])
attrition_dummies.head()
```

Out[149...

	No	Yes
0	0	1
1	1	0
2	0	1
3	1	0
4	1	0

In [150...
```python
df = pd.concat([df, attrition_dummies], axis=1)
df.head()
```

Out[150...

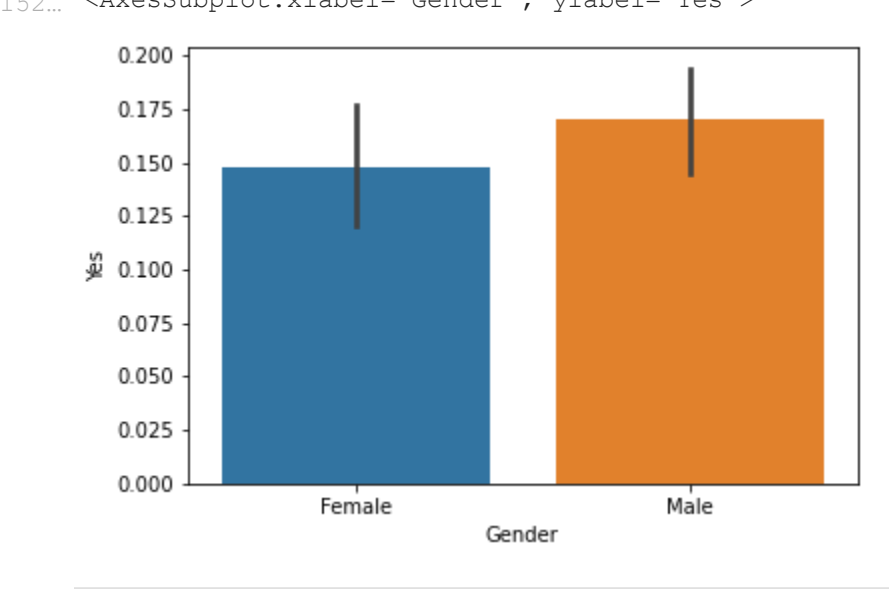
	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	En
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	

5 rows × 35 columns

In [151...
```python
df.drop(['Attrition','No'],axis=1, inplace=True)
```

In [152...
```python
sns.barplot(x=df['Gender'], y=df['Yes'])
```

Out[152... <AxesSubplot:xlabel='Gender', ylabel='Yes'>



In [153...
```python
df.drop (['Age', 'JobLevel'], axis=1, inplace=True)
```

In [154...
```python
from sklearn.preprocessing import LabelEncoder
for column in df.columns:
    if df[column].dtype==np.number:
        continue
    else:
        df[column]=LabelEncoder().fit_transform(df[column])
```

In [155...
```python
df.head()
```

Out[155...

	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfacti
0	2	624	2	0	1	1	
1	1	113	1	7	0	1	
2	2	805	1	1	1	4	
3	1	820	1	2	3	1	
4	2	312	1	1	0	3	

5 rows × 31 columns

In [156...
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 31 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   BusinessTravel            1470 non-null   int32
 1   DailyRate                 1470 non-null   int64
 2   Department                1470 non-null   int32
 3   DistanceFromHome          1470 non-null   int64
 4   Education                 1470 non-null   int64
 5   EducationField            1470 non-null   int32
 6   EnvironmentSatisfaction   1470 non-null   int64
 7   Gender                    1470 non-null   int32
 8   HourlyRate                1470 non-null   int64
 9   JobInvolvement            1470 non-null   int64
 10  JobRole                   1470 non-null   int32
 11  JobSatisfaction           1470 non-null   int64
 12  MaritalStatus             1470 non-null   int32
 13  MonthlyIncome             1470 non-null   int64
 14  MonthlyRate               1470 non-null   int64
 15  NumCompaniesWorked        1470 non-null   int64
 16  Over18                    1470 non-null   int32
 17  OverTime                  1470 non-null   int32
 18  PercentSalaryHike         1470 non-null   int64
 19  PerformanceRating         1470 non-null   int64
 20  RelationshipSatisfaction  1470 non-null   int64
 21  StandardHours             1470 non-null   int64
 22  StockOptionLevel          1470 non-null   int64
 23  TotalWorkingYears         1470 non-null   int64
 24  TrainingTimesLastYear     1470 non-null   int64
 25  WorkLifeBalance           1470 non-null   int64
 26  YearsAtCompany            1470 non-null   int64
 27  YearsInCurrentRole        1470 non-null   int64
 28  YearsSinceLastPromotion   1470 non-null   int64
 29  YearsWithCurrManager      1470 non-null   int64
 30  Yes                       1470 non-null   int64
dtypes: int32(8), int64(23)
memory usage: 310.2 KB
```

In [157...
```python
from sklearn.model_selection import train_test_split
X = df.drop(['Yes'],axis=1)
y = df['Yes']
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=
```

In [158...
```python
X_train.shape
```

Out[158... (1176, 30)

In [159...
```python
X_test.shape
```

Out[159... (294, 30)

In [160...
```python
y_train.shape
```

Out[160... (1176,)

In [161...
```python
y_test.shape
```

Out[161... (294,)

In [162...
```python
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=10, criterion='entropy', random_state=0)
```

In [163...
```python
rf.fit(X_train, y_train)
```

Out[163... RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)

In [164...
```python
y_pred = rf.predict(X_test)
```

In [165...
```python
y_pred
```

Out[165...
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [166...
```python
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

Out[166... 0.8741496598639455

In [167...
```python
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
```

In [111...
```python
lr.fit(X_train, y_train)
```

Out[111... LogisticRegression()

In [112...
```python
lr_pred = lr.predict(X_test)
```

In [113...
```python
lr_pred
```

Out[113...
```
array([0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [114...
```python
accuracy_score(y_test, lr_pred)
```

Out[114... 0.8571428571428571

In [117...
```python
from sklearn.tree import DecisionTreeClassifier as DTC
```

In [118...
```python
model = DTC()
model.fit(X_train, y_train)
```

Out[118... DecisionTreeClassifier()

In [119...
```python
model_pred = model.predict(X_test)
```

In [120...
```python
model_pred
```

Out[120...
```
array([0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [121...
```python
accuracy_score(y_test, model_pred)
```

Out[121... 0.7687074829931972

In [ ]:
```python
'''RANDOM FOREST CLASSIFIER IS THE BEST MODEL AMONG THREE.'''
```

In [ ]: