

```
In [1]: import pandas as pd
import numpy as np
import seaborn as sns
```

```
In [2]: data = pd.read_csv("HR Data.csv")
```

```
In [3]: data.head()
```

Out[3]:

	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	

5 rows × 35 columns

Getting general information about data

```
In [4]: 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
```

Checking missing or N/A values

In [5]: data.isnull().any()

Out[5]:

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

Printing name of every column

```
In [7]: for col in data.columns:  
        print(col)
```

```
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
```

Renaming necessary columns

```
In [8]: data = data.rename(columns={"YearsWithCurrManager": "YearsWithCurrentManager"})
```

```
In [9]: for col in data.columns:  
        print(col)
```

```
Age  
Attrition  
Bus.Travel  
DailyRate  
Department  
DisFromHome  
Ed.  
Ed.Field  
EmployeeCount  
EmployeeNumber  
Env.Satisfaction  
Gender  
HourlyRate  
JobInvolvement  
JobLevel  
JobRole  
JobSatisfaction  
MaritalStatus  
MonthlyIncome  
MonthlyRate  
NumCompaniesWorked  
Over18  
OverTime  
PercentSalaryHike  
PerformanceRating  
RelationshipSatisfaction  
StandardHours  
StockOptionLevel  
TotalWorkingYears  
TrainingTimesLastYear  
WorkLifeBalance  
YearsAtCompany  
YearsInCurrentRole  
YearsSinceLastPromotion  
YearsWithCurrentManager
```

Checking Duplicated Values

```
In [10]: data.duplicated()
```

```
Out[10]: 0      False  
1      False  
2      False  
3      False  
4      False  
...  
1465   False  
1466   False  
1467   False  
1468   False  
1469   False  
Length: 1470, dtype: bool
```

```
In [11]: z_scores = (data['DailyRate'] - data['DailyRate'].mean()) / data['DailyRate']
data = data[(z_scores < 3)]
```

```
In [27]: # 5. Correcting Data Types
# Convert 'Education' column to categorical
data['Ed.'] = data['Ed.'].astype('category')
data['JobRole'] = data['JobRole'].astype('category')
```

Feature Engineering

```
In [28]: # Create a new feature 'TotalExperience' by adding 'YearsAtCompany' and 'YearsSinceLastPromotion'
data['TotalExperience'] = data['YearsAtCompany'] + data['YearsSinceLastPromotion']
```

```
In [29]: # Creating age groups
bins = [18, 30, 40, 50, 60, 70]
labels = ['18-29', '30-39', '40-49', '50-59', '60-69']
data['AgeGroup'] = pd.cut(data['Age'], bins=bins, labels=labels, right=False)
```

```
In [19]: # Create monthly rate
data['MonthlyRate'] = data['DailyRate'] * 30
```

```
In [20]: # creating Tenure
data['Tenure'] = data['YearsAtCompany'] + data['YearsInCurrentRole']
```

```
In [21]: # Promotion ratio
data['PromotionRatio'] = data['YearsSinceLastPromotion'] / data['TotalWorkingHours']
```

```
In [33]: data['TotalWorkingHours'] = data['DailyRate'] * 30 # Assuming 30 working days per month
```

```
In [38]: # Convert 'OverTime' column to numeric
data['OverTime'] = data['OverTime'].map({'Yes': 1, 'No': 0})

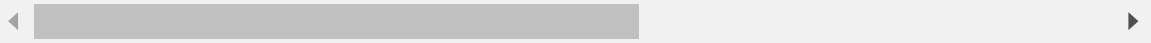
# Calculate 'OvertimeRate' by dividing 'OverTime' by 'TotalWorkingHours'
data['OvertimeRate'] = data['OverTime'] / data['TotalWorkingHours']
```

```
In [39]: # Overtime
data['OvertimeRate'] = data['OverTime'] / data['TotalWorkingHours']
```

```
In [41]: #Drop 'EmployeeCount' and 'StandardHours' columns as they have the same val
data.drop(['EmployeeCount', 'StandardHours'], axis=1, inplace=True)
```

Checking data integrity

```
In [44]: assert (data['YearsAtCompany'] <= data['TotalWorkingYears']).all(), "YearsA
```

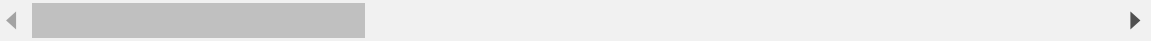


```
In [45]: data.head()
```

```
Out[45]:
```

	Age	Attrition	Bus.Travel	DailyRate	Department	DisFromHome	Ed.	Ed.Field	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 × 39 columns



```
In [ ]:
```