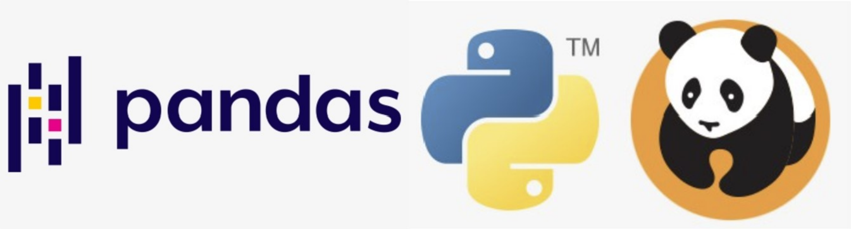
PANDAS

DATA ANALYSIS ASSIGNMENT #1

ANALYZING EMPLOYEE ATTRITION



[Link: Pandas Documentation](https://pandas.pydata.org/docs/)

Files needed for this assignment:

Employee_Attrition.csv

In [2]:
```
# set up notebook to display multiple output in one cell

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

print('The notebook is set up to display multiple output in one cell.')
```

```
The notebook is set up to display multiple output in one cell.
```

In [3]:
```
import pandas as pd
import numpy as np
```

Note: Throughout this assignment, add cells as needed.

Task #1: Reading the Dataset 1. Read in the [**Employee_Attrition.csv**](https://drive.google.com/file/d/1E-XLIZdyUYGf-cCvjx6KaNNICX1ATsVo/view?usp=share_link) dataset and store the results in a DataFrame.

2. Repeat Step #1, but this time set one of the columns as the index.

In [4]:

```
employee = pd.read_csv('Employee_Attrition.csv',sep=';',index_col='EmployeeNumber',skip
employee
```

Out[4]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educa
EmployeeNumber							
1	41	Yes	Travel_Rarely	1102	Sales	1	
2	49	No	Travel_Frequently	279	Research & Development	8	
4	37	Yes	Travel_Rarely	1373	Research & Development	2	
5	33	No	Travel_Frequently	1392	Research & Development	3	
7	27	No	Travel_Rarely	591	Research & Development	2	
...	...	...	...	...	...	...	
2061	36	No	Travel_Frequently	884	Research & Development	23	
2062	39	No	Travel_Rarely	613	Research & Development	6	
2064	27	No	Travel_Rarely	155	Research & Development	4	
2065	49	No	Travel_Frequently	1023	Sales	2	
2068	34	No	Travel_Rarely	628	Research & Development	8	

1470 rows × 34 columns

Task #2: Use DataFrame Attributes to Inspect the Anatomy of the DataFrame 1. Write code to access the DataFrame's index (i.e., to access the row labels).

2. Write code to access the DataFrame's column names/columns (column index).

3. Write code to determine the data type of each variable in your DataFrame.

4. Write code to access the DataFrame's data (i.e., just the data without the index or column names).

5. Write code to determine the shape (i.e., the dimensions) of the DataFrame.

In [5]:

```
employee.index
```

Out[5]:

```
Int64Index([   1,    2,    4,    5,    7,    8,   10,   11,   12,   13,
            ...
            2054, 2055, 2056, 2057, 2060, 2061, 2062, 2064, 2065, 2068],
           dtype='int64', name='EmployeeNumber', length=1470)
```

In [6]:
```
employee.columns
```

Out[6]:
```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       '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 [7]:
```
employee.dtypes
```

Out[7]:
```
Age                          int64
Attrition                   object
BusinessTravel              object
DailyRate                    int64
Department                  object
DistanceFromHome             int64
Education                    int64
EducationField              object
EmployeeCount                int64
EnvironmentSatisfaction      int64
Gender                      object
HourlyRate                   int64
JobInvolvement               int64
JobLevel                     int64
JobRole                     object
JobSatisfaction              int64
MaritalStatus               object
MonthlyIncome                int64
MonthlyRate                  int64
NumCompaniesWorked           int64
Over18                      object
OverTime                    object
PercentSalaryHike            int64
PerformanceRating            int64
RelationshipSatisfaction     int64
StandardHours                int64
StockOptionLevel             int64
TotalWorkingYears            int64
TrainingTimesLastYear        int64
WorkLifeBalance              int64
YearsAtCompany               int64
YearsInCurrentRole           int64
YearsSinceLastPromotion      int64
YearsWithCurrManager         int64
dtype: object
```

In [8]:
```
employee.values
```

Out[8]:

```
array([[41, 'Yes', 'Travel_Rarely', ..., 4, 0, 5],
       [49, 'No', 'Travel_Frequently', ..., 7, 1, 7],
       [37, 'Yes', 'Travel_Rarely', ..., 0, 0, 0],
       ...,
       [27, 'No', 'Travel_Rarely', ..., 2, 0, 3],
       [49, 'No', 'Travel_Frequently', ..., 6, 0, 8],
       [34, 'No', 'Travel_Rarely', ..., 3, 1, 2]], dtype=object)
```

In [9]:

```
employee.shape
```

Out[9]:

```
(1470, 34)
```

Task #3: Use DataFrame Methods to Inspect Your Data 1. Write code to access the first 5 rows of the data.

2. Write code to access the first 8 rows of the data.
3. Write code to access the last 5 rows of the data.
4. Write code to asccess the last 7 rows of the data.
5. Write code to get detailed information about your DataFrame.

In [10]:

```
employee.head()
```

Out[10]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educa
EmployeeNumber							
1	41	Yes	Travel_Rarely	1102	Sales	1	
2	49	No	Travel_Frequently	279	Research & Development	8	
4	37	Yes	Travel_Rarely	1373	Research & Development	2	
5	33	No	Travel_Frequently	1392	Research & Development	3	
7	27	No	Travel_Rarely	591	Research & Development	2	

5 rows × 34 columns

In [11]:

```
employee.head(8)
```

Out[11]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educa
EmployeeNumber							
1	41	Yes	Travel_Rarely	1102	Sales	1	
2	49	No	Travel_Frequently	279	Research & Development	8	
4	37	Yes	Travel_Rarely	1373	Research & Development	2	
5	33	No	Travel_Frequently	1392	Research & Development	3	
7	27	No	Travel_Rarely	591	Research & Development	2	
8	32	No	Travel_Frequently	1005	Research & Development	2	
10	59	No	Travel_Rarely	1324	Research & Development	3	
11	30	No	Travel_Rarely	1358	Research & Development	24	

8 rows × 34 columns

In [12]:

```
employee.tail()
```

Out[12]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educa
EmployeeNumber							
2061	36	No	Travel_Frequently	884	Research & Development	23	
2062	39	No	Travel_Rarely	613	Research & Development	6	
2064	27	No	Travel_Rarely	155	Research & Development	4	
2065	49	No	Travel_Frequently	1023	Sales	2	
2068	34	No	Travel_Rarely	628	Research & Development	8	

5 rows × 34 columns

In [13]:

```
employee.tail(7)
```

Out[13]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educa
EmployeeNumber							
2057	31	No	Non-Travel	325	Research & Development	5	
2060	26	No	Travel_Rarely	1167	Sales	5	
2061	36	No	Travel_Frequently	884	Research & Development	23	
2062	39	No	Travel_Rarely	613	Research & Development	6	
2064	27	No	Travel_Rarely	155	Research & Development	4	
2065	49	No	Travel_Frequently	1023	Sales	2	
2068	34	No	Travel_Rarely	628	Research & Development	8	

7 rows × 34 columns

In [14]:

```
employee.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1470 entries, 1 to 2068
Data columns (total 34 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   EnvironmentSatisfaction   1470 non-null   int64 
 10  Gender                    1470 non-null   object
 11  HourlyRate                1470 non-null   int64 
 12  JobInvolvement            1470 non-null   int64 
 13  JobLevel                  1470 non-null   int64 
 14  JobRole                   1470 non-null   object
 15  JobSatisfaction           1470 non-null   int64 
 16  MaritalStatus             1470 non-null   object
 17  MonthlyIncome             1470 non-null   int64 
 18  MonthlyRate               1470 non-null   int64 
 19  NumCompaniesWorked        1470 non-null   int64 
 20  Over18                    1470 non-null   object
 21  OverTime                  1470 non-null   object
 22  PercentSalaryHike         1470 non-null   int64 
 23  PerformanceRating         1470 non-null   int64 
 24  RelationshipSatisfaction  1470 non-null   int64 
 25  StandardHours             1470 non-null   int64 
 26  StockOptionLevel          1470 non-null   int64 
 27  TotalWorkingYears         1470 non-null   int64 
 28  TrainingTimesLastYear     1470 non-null   int64 
 29  WorkLifeBalance           1470 non-null   int64 
 30  YearsAtCompany            1470 non-null   int64 
 31  YearsInCurrentRole        1470 non-null   int64 
 32  YearsSinceLastPromotion   1470 non-null   int64 
 33  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(25), object(9)
memory usage: 402.0+ KB
```

Task #4: **Calculate Summary Statistics for the DataFrame's Columns** 1. Write code to compute summary statistics for the numeric variables.

2. Write code to compute summary statistics for just the Age column.

3. Repeat Step #1, but also include the 10th and 90th percentiles in the summary statistics that you compute.

4. Write code to compute summary statistics for the string variables.

5. Write code to compute summary statistics for just the MaritalStatus column.

6. Pick out 3 numeric variables that you think will be important in analyzing employee attrition and compute summary statistics for just those numeric variables.

In [15]:

```
employee.describe()
```

Out[15]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EnvironmentSatisfacti
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000
mean	36.923810	802.485714	9.192517	2.912925	1.0	2.721
std	9.135373	403.509100	8.106864	1.024165	0.0	1.093
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000
25%	30.000000	465.000000	2.000000	2.000000	1.0	2.000
50%	36.000000	802.000000	7.000000	3.000000	1.0	3.000
75%	43.000000	1157.000000	14.000000	4.000000	1.0	4.000
max	60.000000	1499.000000	29.000000	5.000000	1.0	4.000

8 rows × 25 columns

In [16]:

```
employee.Age.describe()
```

Out[16]:

```
count    1470.000000
mean       36.923810
std         9.135373
min        18.000000
25%        30.000000
50%        36.000000
75%        43.000000
max        60.000000
Name: Age, dtype: float64
```

In [17]:

```
employee.Age.describe(percentiles=[.1, .25,.50,.75, .9])
```

Out[17]:

```
count    1470.000000
mean       36.923810
std         9.135373
min        18.000000
10%        26.000000
25%        30.000000
50%        36.000000
75%        43.000000
90%        50.000000
max        60.000000
Name: Age, dtype: float64
```

In [18]:

```
employee.describe(include=[object])
```

Out[18]:

	Attrition	BusinessTravel	Department	EducationField	Gender	JobRole	MaritalStatus	Over1
count	1470	1470	1470	1470	1470	1470	1470	14
unique	2	3	3	6	2	9	3	
top	No	Travel_Rarely	Research & Development	Life Sciences	Male	Sales Executive	Married	
freq	1233	1043	961	606	882	326	673	14

Task #5: Answer Simple Questions about the Dataset
The HR director asks you to answer a few descripive questions about employees, you use this dataset to answer them: - How many employees are there by department in the dataset? - What is the overall attrition rate? - What is the average hourly rate and average yearly income? - What is the average number of years at the company? - Who are the 5 employees with the most number of years at the company? - How satisfied are employees overall?

Question 1: How many employees are there by department in the dataset?

In [19]:
```
employee['Department'].value_counts()
```

Out[19]:
```
Research & Development    961
Sales                     446
Human Resources            63
Name: Department, dtype: int64
```

Question 2: What is the overall attrition rate?

In [20]:
```
employee['Attrition'].value_counts(normalize=True)['Yes']*100
```

Out[20]:
```
16.122448979591837
```

Question 3: What is the average hourly rate and average yearly income?

In [21]:
```
print(f"Average Hourly Rate(Mean): {employee.HourlyRate.mean()}\n")
print(f"Average Hourly Rate(Median): {employee.HourlyRate.median()}\n")
print(f"Average Yearly Income(Mean): {employee.MonthlyIncome.sum() * 12 / len(employee.
print(f"Average Yearly Income(Median): {employee.MonthlyIncome.median() * 12}\n")
```

```
Average Hourly Rate(Mean): 65.89115646258503

Average Hourly Rate(Median): 66.0

Average Yearly Income(Mean): 78035.17551020408

Average Yearly Income(Median): 59028.0
```

Question 4: What is the average number of years at the company?

In [22]:
```
employee.YearsAtCompany.mean()
```

Out[22]:
```
7.0081632653061225
```

Question 5: Who are the 5 employees with the most number of years at the company?

```
In [23]: employee.YearsAtCompany.sort_values(ascending=False).head()
```

```
Out[23]: EmployeeNumber
165     40
131     37
1578    36
374     36
776     34
Name: YearsAtCompany, dtype: int64
```

```
In [24]: employee.YearsAtCompany.sort_values(ascending=False).value_counts()
```

```
Out[24]: 5     196
1     171
3     128
2     127
10    120
4     110
7      90
9      82
8      80
6      76
0      44
11     32
20     27
13     24
15     20
14     18
22     15
12     14
21     14
18     13
16     12
19     11
17      9
24      6
33      5
25      4
26      4
31      3
32      3
36      2
27      2
29      2
23      2
30      1
34      1
37      1
40      1
Name: YearsAtCompany, dtype: int64
```

Question 6: How satisfied are employees overall?

```
In [25]: employee.JobSatisfaction.value_counts()
```

```
Out[25]: 4    459
         3    442
         1    289
         2    280
         Name: JobSatisfaction, dtype: int64
```

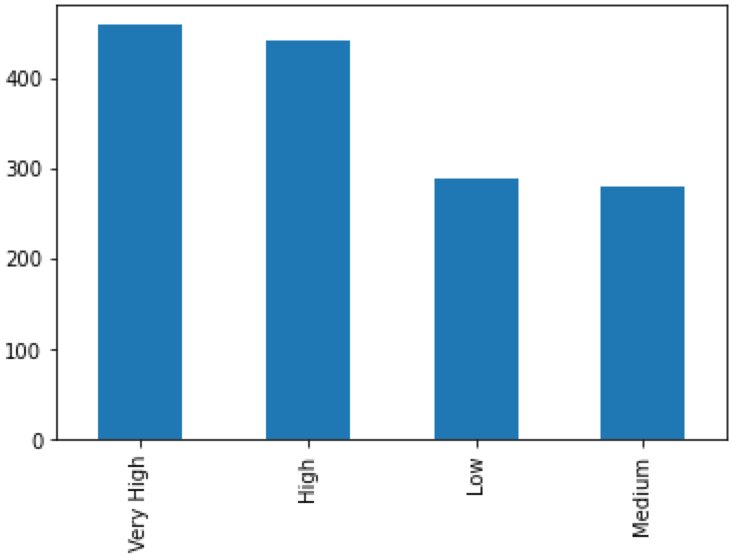
```
In [26]: JobSatisfaction_cat = {
             1: 'Low',
             2: 'Medium',
             3: 'High',
             4: 'Very High'
         }
         employee.JobSatisfaction = employee.JobSatisfaction.map(JobSatisfaction_cat)
```

```
In [27]: 100 * employee.JobSatisfaction.value_counts(normalize=True)
```

```
Out[27]: Very High    31.224490
         High         30.068027
         Low          19.659864
         Medium       19.047619
         Name: JobSatisfaction, dtype: float64
```

```
In [28]: employee.JobSatisfaction.value_counts().plot(kind = 'bar')
```

```
Out[28]: <AxesSubplot:>
```



Task #6: Answer Additional Questions about the Dataset

After taking a look at your answers the HR director, asks you more questions: - Give me the list of the employees with Low level of JobSatisfaction - Give me the list of the employees with Low level of both JobSatisfaction and PerformanceRating - Compare the employees with Low and Very High JobSatisfaction across the following variables: Age, Department, - DistanceFromHome, HourlyRate, MonthlyIncome and YearsAtCompany.

Question 7: Give me the list of the employees with Low level of JobSatisfaction

In [29]:
```
employee.loc[employee.JobSatisfaction == 'Low'].index
```

Out[29]:
```
Int64Index([  10,   20,   27,   31,   33,   38,   51,   52,   54,   68,
            ...
            1975, 1980, 1998, 2021, 2023, 2038, 2054, 2055, 2057, 2062],
           dtype='int64', name='EmployeeNumber', length=289)
```

In [30]:
```
employee[employee.JobSatisfaction == 'Low'].JobSatisfaction
```

Out[30]:
```
EmployeeNumber
10      Low
20      Low
27      Low
31      Low
33      Low
       ...
2038    Low
2054    Low
2055    Low
2057    Low
2062    Low
Name: JobSatisfaction, Length: 289, dtype: object
```

Question 8: Give me the list of the employees with Low level of both JobSatisfaction and JobInvolment

In [31]:
```
employee.JobInvolvement.value_counts()
```

Out[31]:
```
3    868
2    375
4    144
1     83
Name: JobInvolvement, dtype: int64
```

In [32]:
```
employee.JobInvolvement = employee.JobInvolvement.map(JobSatisfaction_cat)
```

In [33]:
```
employee.loc[(employee.JobSatisfaction == 'Low') & (employee.JobInvolvement == 'Low')].i
```

Out[33]:
```
Int64Index([33, 235, 454, 615, 1019, 1037, 1237, 1460, 1478, 1544, 1611, 1622,
            1905, 1956],
           dtype='int64', name='EmployeeNumber')
```

Question 9: Compare the employees with Low and Very High JobSatisfaction across the following variables: Age, Department, DistanceFromHome, HourlyRate, MonthlyIncome and YearsAtCompany.

In [38]:
```
subset_of_interest = employee.loc[(employee.JobSatisfaction == 'Low') | (employee.JobSa
```

In [39]:
```
subset_of_interest.shape
```

Out[39]:
```
(748, 34)
```

In [41]:
```
subset_of_interest.JobSatisfaction.value_counts()
```

Out[41]:

```
Very High    459
Low          289
Name: JobSatisfaction, dtype: int64
```

In [42]:

```
grouped = subset_of_interest.groupby('JobSatisfaction')
```

In [50]:

```
grouped.groups
```

Out[50]:

```
{'Low': [10, 20, 27, 31, 33, 38, 51, 52, 54, 68, 70, 74, 75, 81, 86, 88, 100, 101, 113,
124, 133, 134, 145, 153, 170, 190, 197, 199, 200, 235, 239, 240, 241, 244, 250, 267, 27
4, 282, 288, 297, 299, 303, 328, 334, 339, 340, 347, 351, 362, 369, 374, 382, 390, 396,
412, 424, 425, 429, 451, 454, 474, 486, 510, 515, 517, 522, 524, 530, 532, 534, 536, 53
8, 549, 567, 573, 590, 605, 615, 625, 630, 648, 650, 662, 664, 667, 682, 684, 702, 705,
725, 728, 729, 732, 733, 742, 758, 764, 771, 775, 776, ...], 'Very High': [1, 8, 18, 2
2, 23, 24, 30, 36, 39, 40, 42, 45, 49, 53, 57, 62, 63, 72, 73, 76, 78, 79, 97, 98, 104,
106, 107, 112, 116, 117, 118, 120, 137, 139, 140, 143, 144, 148, 152, 154, 155, 158, 16
5, 169, 174, 179, 184, 192, 195, 198, 207, 215, 217, 221, 223, 228, 230, 242, 243, 245,
246, 262, 264, 273, 275, 281, 283, 286, 287, 291, 298, 302, 306, 309, 311, 312, 315, 31
6, 319, 323, 325, 327, 333, 335, 336, 338, 346, 349, 353, 361, 367, 372, 373, 377, 378,
380, 388, 389, 391, 393, ...]}
```

In [52]:

```
grouped.get_group('Low').head()
```

Out[52]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educa
EmployeeNumber							
10	59	No	Travel_Rarely	1324	Research & Development	3	
20	29	No	Travel_Rarely	1389	Research & Development	21	
27	36	Yes	Travel_Rarely	1218	Sales	9	
31	34	Yes	Travel_Rarely	699	Research & Development	6	
33	32	Yes	Travel_Frequently	1125	Research & Development	16	

5 rows × 34 columns

In [54]:

```
grouped.get_group('Very High').head()
```

Out[54]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educa
EmployeeNumber							
1	41	Yes	Travel_Rarely	1102	Sales	1	
8	32	No	Travel_Frequently	1005	Research & Development	2	
18	34	No	Travel_Rarely	1346	Research & Development	19	
22	22	No	Non-Travel	1123	Research & Development	16	
23	53	No	Travel_Rarely	1219	Sales	2	

5 rows × 34 columns

Age

In [56]:

```
grouped['Age'].mean()
```

Out[56]:

```
JobSatisfaction
Low          36.916955
Very High    36.795207
Name: Age, dtype: float64
```

In [58]:

```
grouped['Age'].describe().unstack()
```

Out[58]:

```
       JobSatisfaction
count  Low                289.000000
       Very High          459.000000
mean   Low                 36.916955
       Very High           36.795207
std    Low                  9.245496
       Very High            9.125609
min    Low                 19.000000
       Very High           18.000000
25%    Low                 30.000000
       Very High           30.000000
50%    Low                 36.000000
       Very High           35.000000
75%    Low                 42.000000
       Very High           43.000000
max    Low                 60.000000
       Very High           60.000000
dtype: float64
```

Department

In [59]:

```
grouped['Department'].describe()
```

```
Out[59]:                 count  unique                      top  freq
         JobSatisfaction
                     Low   289       3   Research & Development   192
               Very High   459       3   Research & Development   295
```

DistanceFromHome

```
In [ ]:
```

HourlyRate

```
In [ ]:
```

MonthlyIncome

```
In [ ]:
```

YearsAtCompany

```
In [ ]:
```

Task #7: Create a DataFrame to Compare the Means Across All Numerical Variables

Comparing the means across all numerical variables

Although we we asked for just some specific columns, to give the HR director a better picture of how these groups compare across different variables, let's create a DataFrame that contains the mean for every numeric variable in our dataset.

```
In [ ]:
```

Task #8: Create Some Additional Questions That You Could Ask About the Dataset

```
In [ ]:
```

Final Note:

Issues to keep in mind about this dataset:

- There are many variables that are detected as numerical but are actually categorical (like Education).
- Since this is a simulated dataset, it is hard to find interesting patterns.

```
In [ ]:
```