```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df=pd.read_csv('/content/set2data.csv')
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField
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

```
df
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationFie
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Scienc
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Scienc
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Oth
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Scienc
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medic
...	...	...	...	...	...	...	...	
1465	36	No	Travel_Frequently	884	Research & Development	23	2	Medic
1466	39	No	Travel_Rarely	613	Research & Development	6	1	Medic
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Life Scienc
1468	49	No	Travel_Frequently	1023	Sales	2	3	Medic
1469	34	No	Travel_Rarely	628	Research & Development	8	3	Medic

1470 rows × 35 columns

```
df.isnull()
```

```
df.isnull()
```

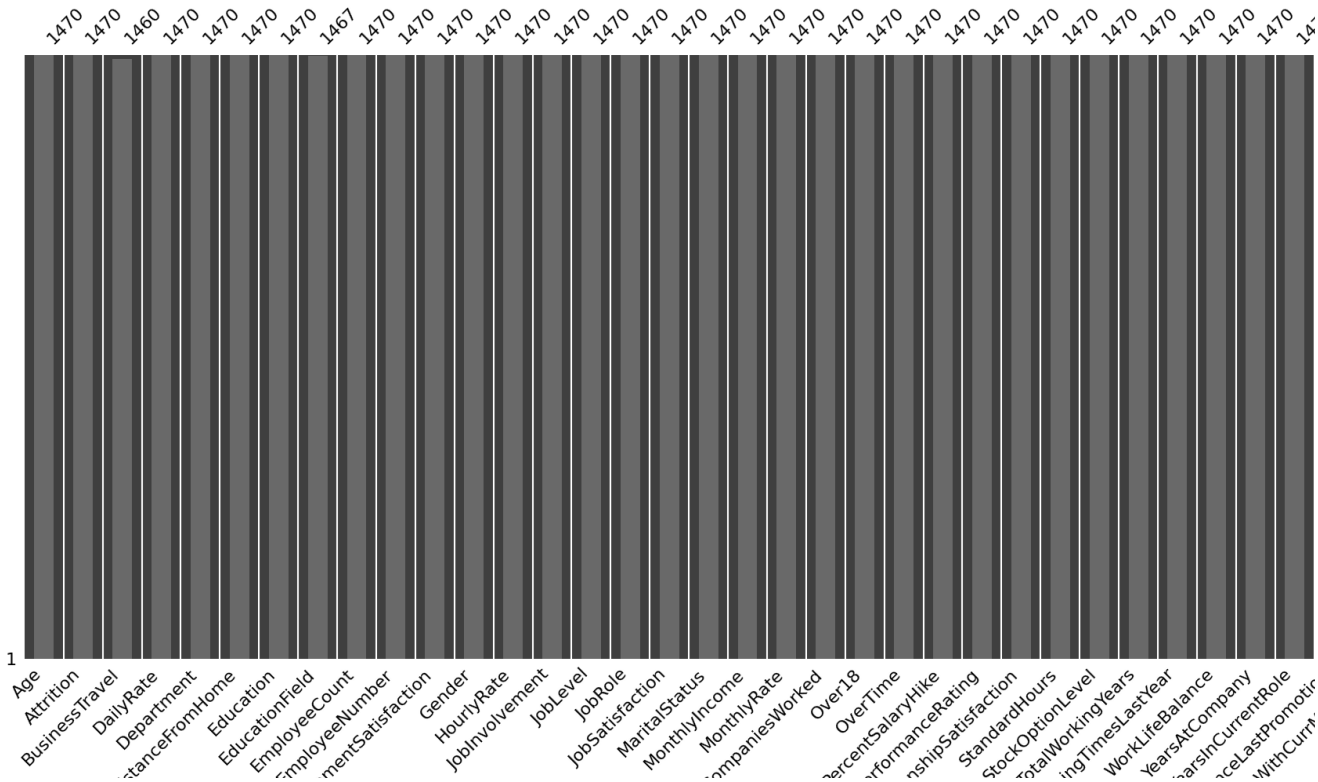
	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationFie
0	False	False	False	False	False	False	False	Fal
1	False	False	False	False	False	False	False	Fal
2	False	False	False	False	False	False	False	Fal
3	False	False	False	False	False	False	False	Fal
4	False	False	False	False	False	False	False	Fal
...	...	...	...	...	...	...	...	
1465	False	False	False	False	False	False	False	Fal
1466	False	False	False	False	False	False	False	Fal
1467	False	False	False	False	False	False	False	Fal
1468	False	False	False	False	False	False	False	Fal
1469	False	False	False	False	False	False	False	Fal

1470 rows × 35 columns

Since all the rows show false, we can conclude no NUll values are present in the given dataset.

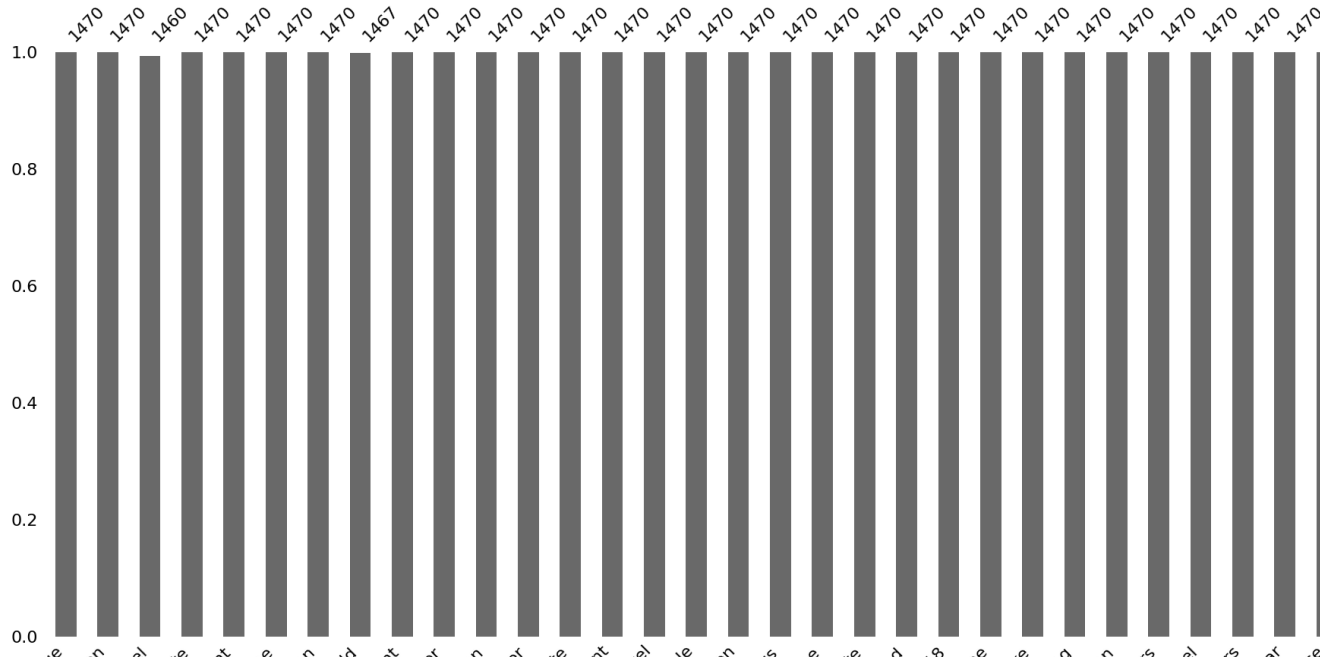
```
import missingno as msno
msno.matrix(df)
msno.bar(df)
```

```
<Axes: >
```




```
msno.bar(df)
```

```
<Axes: >
```




```
df.dtypes
```

```
Age                         int64
Attrition                  object
BusinessTravel             object
DailyRate                   int64
Department                 object
DistanceFromHome            int64
Education                   int64
EducationField             object
EmployeeCount               int64
EmployeeNumber              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
```

```
df['EducationField']
```

```
0       Life Sciences
1       Life Sciences
2               Other
3       Life Sciences
4             Medical
```

```
                ...
1465          Medical
1466          Medical
1467    Life Sciences
1468          Medical
1469          Medical
Name: EducationField, Length: 1470, dtype: object
```

```
df['EducationField'].replace('Other','Business Travel-Non-Travel')
```

```
0                    Life Sciences
1                    Life Sciences
2       Business Travel-Non-Travel
3                    Life Sciences
4                          Medical
                    ...
1465                       Medical
1466                       Medical
1467                 Life Sciences
1468                       Medical
1469                       Medical
Name: EducationField, Length: 1470, dtype: object
```

```
df['Gender']
```

```
0       Female
1         Male
2         Male
3       Female
4         Male
          ...
1465      Male
1466      Male
1467      Male
1468      Male
1469      Male
Name: Gender, Length: 1470, dtype: object
```

```
df['JobSatisfaction']
```

```
0       4
1       2
2       3
3       3
4       2
       ..
1465    4
1466    1
1467    2
1468    2
1469    3
Name: JobSatisfaction, Length: 1470, dtype: int64
```

```
data= df[df['Gender'].str.contains('Female')]
print(data)
```

```
1457            2051  ...                        2            80
1458            2052  ...                        4            80
1460            2054  ...                        2            80
1462            2056  ...                        1            80
1464            2060  ...                        4            80

      StockOptionLevel  TotalWorkingYears  TrainingTimesLastYear  \
0                    0                  8                      0
3                    0                  8                      3
6                    3                 12                      3
11                   0                 10                      3
15                   1                 10                      1
...                ...                ...                    ...
1457                 3                 20                      2
1458                 1                  4                      5
1460                 0                  5                      3
1462                 1                 21                      2
1464                 0                  5                      2

     WorkLifeBalance  YearsAtCompany YearsInCurrentRole  \
0                  1               6                  4
3                  3               8                  7
6                  2               1                  0
11                 3               9                  5
15                 3              10                  9
...              ...             ...                ...
1457               3               5                  3
1458               3               4                  3
1460               1               5                  4
1462               2              20                  9
1464               3               4                  2

      YearsSinceLastPromotion  YearsWithCurrManager
0                           0                     5
3                           3                     0
6                           0                     0
11                          0                     8
15                          8                     8
...                       ...                   ...
1457                        0                     2
1458                        1                     1
1460                        0                     4
1462                        9                     6
1464                        0                     0

[588 rows x 35 columns]
```

```
d=pd.DataFrame(df,columns=["Gender","JobSatisfaction"])
```

```
d.plot(kind="bar")
```

```
<Axes: >
```


```
b=pd.DataFrame(df,columns=["MonthlyIncome","RelationshipSatisfaction"])
b
```

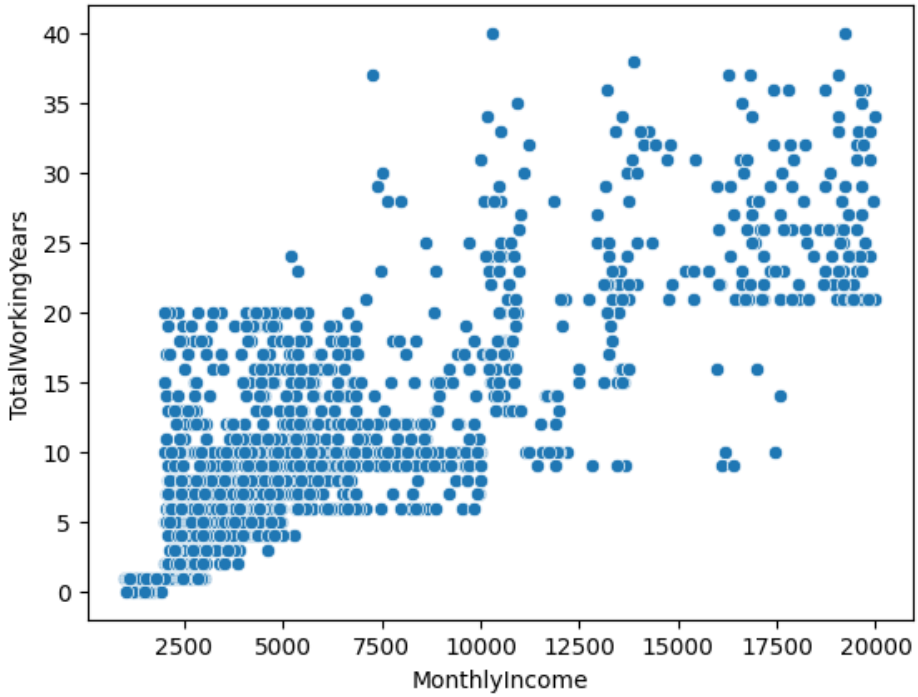
```
sns.scatterplot(data=b,columns=["MonthlyIncome","RelationshipSatisfaction"])
```


```
a=pd.DataFrame(df,columns=["MonthlyIncome","TotalWorkingYears"])
a
```



```
sns.scatterplot(data=a,x="MonthlyIncome",y="TotalWorkingYears")
```

```
<Axes: xlabel='MonthlyIncome', ylabel='TotalWorkingYears'>
```



Formulated a problem statement: given a person's monthly income we can determine for how many years the person has been working based on Linear Regression.

```
a.corr()
```

	MonthlyIncome	TotalWorkingYears
MonthlyIncome	1.000000	0.772893
TotalWorkingYears	0.772893	1.000000

```
sns.heatmap(a.corr(),annot=True)
```

```
<Axes: >
```


```
x=a.iloc[:,:-1]
y=a.iloc[:,-1]
```

```
x.dtypes
```

```
MonthlyIncome    int64
dtype: object
```

```
y.dtypes
```

```
dtype('int64')
```

```
x.shape
```

```
(1470, 1)
```

```
y.shape
```

```
(1470,)
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

```
x_train
```

MonthlyIncome

```
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(x_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
regressor.intercept_
```

```
2.9398158385624154
```

1294 6870

```
regressor.coef_
```

```
array([0.0012815])
```

```
years=regressor.predict([[4000]])
print(years)
```

```
[8.06583308]
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature
  warnings.warn(
```

```
y_pred=regressor.predict(x_test)
print(y_pred)
```

```
[13.78518683  8.64251002  4.93255504 15.40116376 10.51863234  6.24481546
  8.36057908  5.80013346 14.30035156  5.84754912  7.16109104 16.05088645
 10.54298092  5.77578488  7.26489289  9.30632926 11.00560398  6.45498217
 18.12436043 24.73435967  7.39816934  9.4370427  14.73478152 10.1226475
 11.1311914  28.03935928  5.91034283  8.11324874  7.32256058 10.57373702
  9.72409966  9.95220743  8.37852014 12.53956464 10.80056329 12.96502407
 20.56947065  5.63994542  7.34947217  7.54169782 14.07096229  9.95348894
  8.17732396  6.5549395   6.59338463  9.66643197 14.59381605  9.76254479
 11.64891914  5.32213235 25.82876435  9.11794812 27.97656557 10.78902975
  6.75100966  9.34477439  6.45241916 10.58783357 10.81081532  9.40116058
 15.4421719   4.75442594 12.48189694 25.54427039  9.06156193 10.52119535
 10.90308363  8.42978031  7.18159511 24.00390221  9.99706008 27.22047803
 15.24482024  8.46566243 11.42721889 21.84969346  8.99364221 11.26959386
 16.07779804 18.21022122 26.78348506 11.12734689 25.84670541  5.62584888
 12.34093147  6.99193247 12.01542937 10.60961914  9.57160065  8.07992963
 10.61987118  4.51862915 20.03636486  6.11282051 12.45882986 12.09360114
 13.986383    8.64122852  8.82320213 14.56434145 27.34606545 12.74973134
 12.74844984  6.18714776 17.03508176 14.14016352  9.67796551  8.89624788
 20.45029075 27.54726163  9.91119929 10.33281421 11.82320373 15.69975427
  5.12093618  7.36356872  9.95348894 15.72922887 22.64807065  7.85822939
  5.58740375 13.83260249  6.57672507 19.55451924 10.60321162  6.22559289
  9.28326218 18.53444181  8.77450497 11.81935921  7.82491027  4.88001337
 16.0957391   8.71683728  8.6040649   9.89710275  7.07907476 15.5293142
  6.39475146 20.84627558  5.70017613 12.43063677  8.5451157  14.12222246
 20.83730505  6.42294456  9.17946033  6.30376466  6.72794258 10.3007766
  9.96758548 15.87788337  8.39517969  7.71982692 10.78262223  8.10555972
 13.63525082  6.17305122 28.09190096  5.97698106  9.16151927  9.77920435
  9.54468906  6.25506749  8.04789202 11.75272099  5.76681435  7.94024566
 24.78433833 13.31359324 10.5955226  24.49856287  6.34861731  7.54938685
  9.41782013  5.89752779  9.96758548 15.03593504 16.32769138  7.75827205
 20.11453662  9.67155799 27.58570676 13.67369595 13.21748042  5.12606219
 13.12136759 16.39048509 14.37852332  7.63012162 11.28625342 12.72666427
 13.17647228  9.87275416 11.33238757  5.87446071  6.21918537 26.5579403
  9.81252346 11.33751359  7.69163383  9.30632926  8.7552824  27.53060207
 21.92402071  6.1384506  14.56177844  6.23328192  7.64678117 28.0137292
 15.91376549  5.95263248  7.64165516 12.36015403 15.67284268  6.22559289
 11.46566402 11.11196883 11.15425848 16.35203996  5.55664764  6.63823728
 20.37211899 10.32768819  9.75870028  5.80397797  9.06412494 10.47249818
```

```
 16.23798608  7.36356872  9.4677988   8.36442359 26.08506521 15.61133047
  8.41440226  6.87275257 11.02098203  7.66984825 20.52974402 27.94580947
  6.40884801  8.6937702   9.7651078  27.53188358 15.02952751  8.62969498
  6.48701977 10.06754282  9.18843086  9.84712408 11.63097808 27.46908987
 27.37554005 10.51991384 11.29009793 10.34050324  5.76040683  6.11666503
  7.57501693 25.49813623 11.05686415  7.52631977  4.57245233  9.42807217
 15.75229595 12.42679226 27.95093549 13.71983011  8.61047242  9.3191443
  6.60363667 27.49471995  9.18843086 10.72367303 12.08975662  9.95733345
  7.3264051   9.04618388 11.23371174 14.78219718  8.3964612   9.00902026
  5.97313654 15.38963023 20.83474205  6.75485417  9.21662396  5.92828389
  8.4579734  10.97997389  9.50239942 24.57160862  8.93725602  8.7540009 ]
```

```
df_preds=pd.DataFrame({'Actual': y_test,'Predicted':y_pred})
df_preds
```

	Actual	Predicted
1041	6	13.785187
184	5	8.642510
1222	1	4.932555
67	25	15.401164
220	16	10.518632
...	...	...
567	6	10.979974
560	7	9.502399
945	25	24.571609
522	4	8.937256
651	8	8.754001

294 rows × 2 columns

To understand the difference between the actual and predicted values of our Linear Regression model.

```
from sklearn.metrics import mean_absolute_error,mean_squared_error
mean=mean_absolute_error(y_test,y_pred);
msq=mean_squared_error(y_test,y_pred);
rmse=np.sqrt(msq)
print(rmse)
```

```
4.818871376953875
```

```
prediction=regressor.predict(x_test)
prediction.shape
```

```
(294,)
```

```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
```

6m 46s completed at 4:57 PM