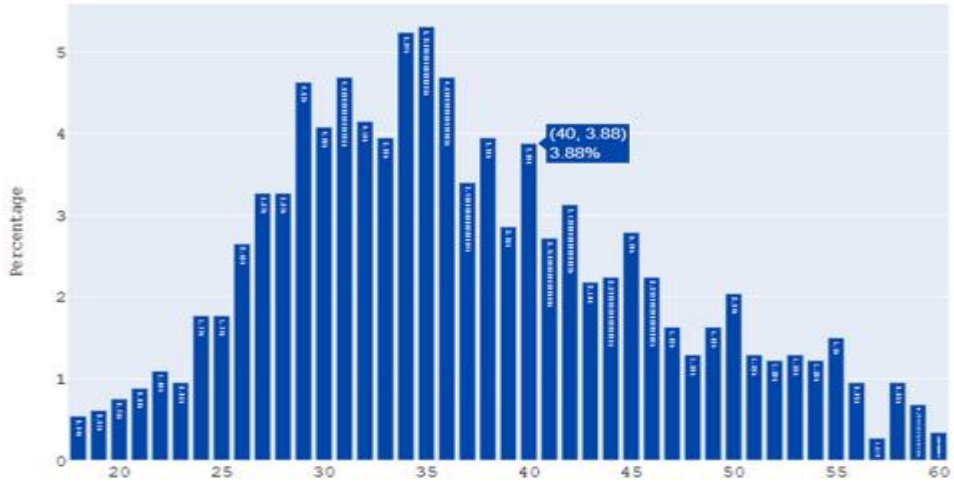
Data Visualization using plotly, matplotlib, seaborn and squarify | Data Science

Data Visualization is one of the important activities we perform when doing Exploratory Data Analysis. It helps in preparing business reports, visual dashboards, storytelling etc important tasks. In this post I have explained how to ask questions from the data and in return get the self-explanatory graphs. In this You will learn the use of various python libraries like plotly, matplotlib, seaborn, squarify etc to plot those graphs.

Key takeaways from this post are:

- Asking questions from data set
- Univariate Analysis
- Bivariate Analysis
- Analysis of more than 3 variables
- 3D Visualization
- Case Study on employee Attrition Rate using HR Data Set




```
import warnings
warnings.filterwarnings('ignore')
!pip install plotly
!pip install squarify
```

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
import plotly
import plotly.offline as pyoff
import plotly.figure_factory as ff
from plotly.offline import init_notebook_mode, iplot, plot
import plotly.graph_objs as go
import squarify # for tree maps
%matplotlib inline
```

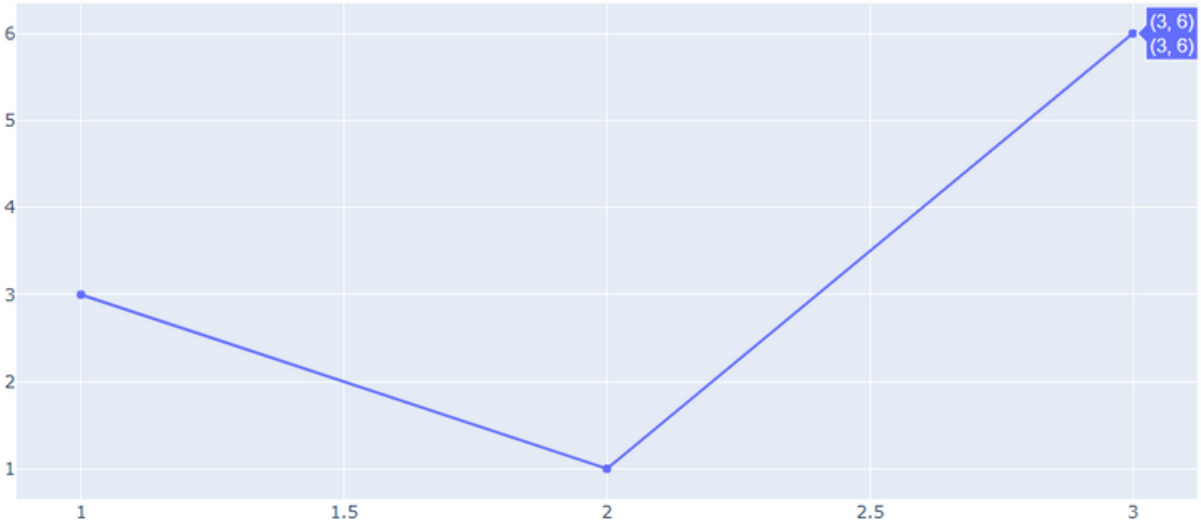
plotly

- Visualization library for the data Era

Line Chart in plotly

- 2 numeric variables with 1-1 mapping, i.e in situations where we have 1 y value corresponding to 1 x value

```
x=[1, 2, 3]
y=[3, 1, 6]
iplot([go.Scatter(x=x,
y=y,
text = [str(i) for i in (zip(x,y))],
textposition = 'top center')])
```



You can export images to html file only with offline mode

- https://plot.ly/python/static-image-export/
- https://plot.ly/python/privacy/

```
from plotly.offline import plot
plot([go.Scatter(x=x,
y=y,
text = [str(i) for i in (zip(x,y))],
textposition = 'top center')],
output_type='file' ,
filename='temp-histogram.jpeg',image='jpeg',auto_open=False)
```

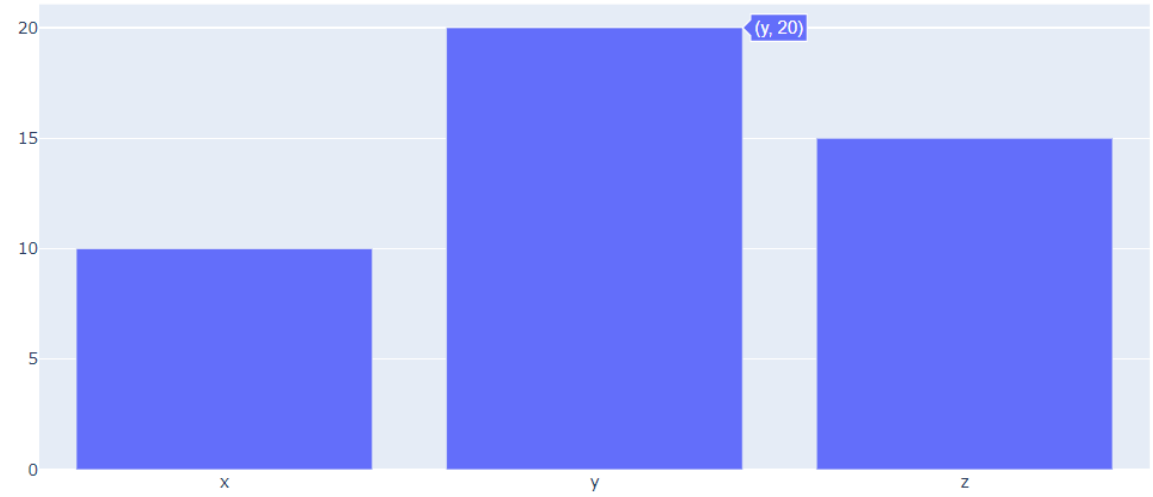
```
output -> 'temp-histogram.jpeg.html'
```

Note that this is a bare chart with no information, later in the activity we will add title, x labels and y labels.

Basic Bar chart in plotly

- 1 Categorical variable

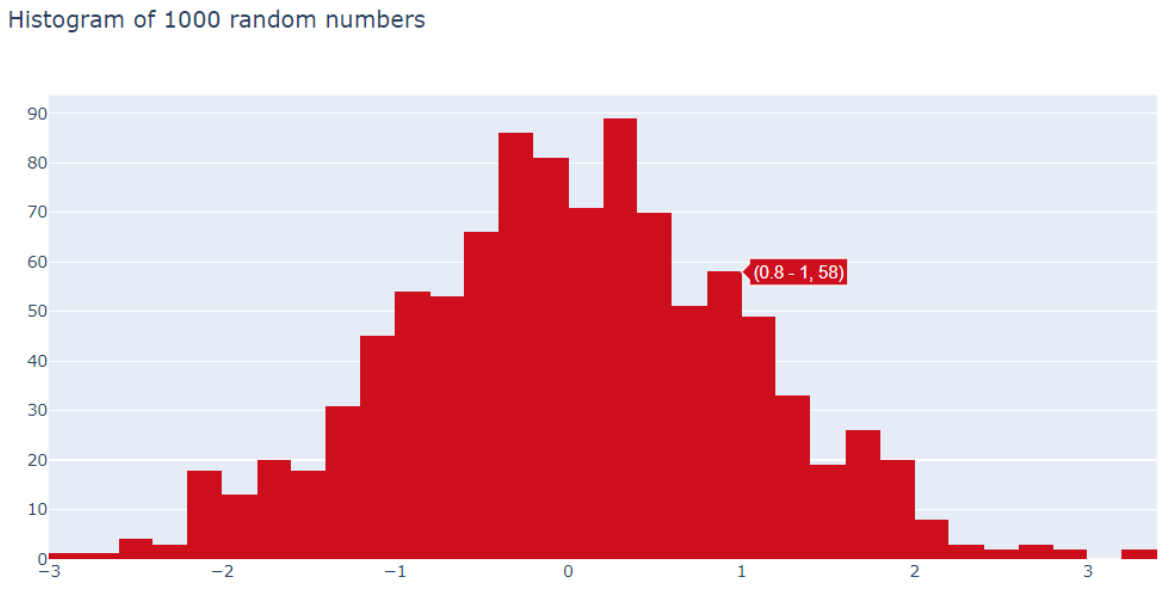
```
data = [go.Bar(
x=['x', 'y', 'z'],
y=[10, 20, 15])]
iplot(data)
```



Histogram in plotly

- 1 numeric variable

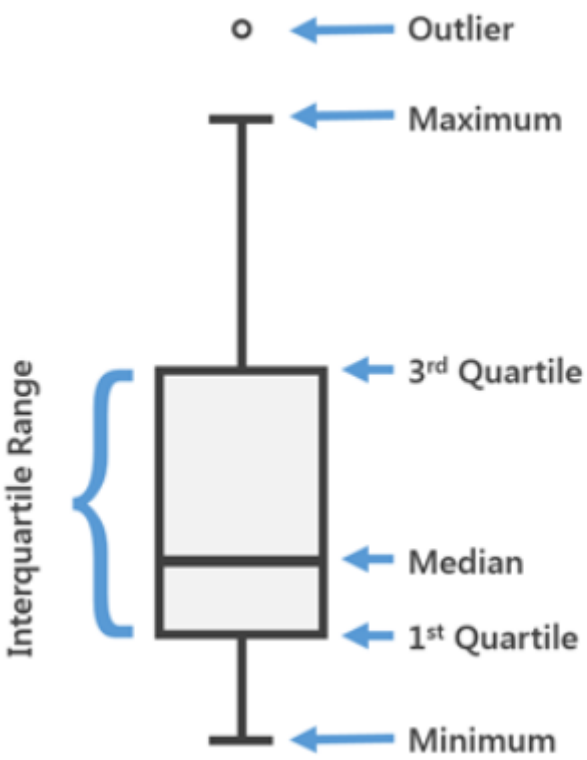
```
n = 1000
x = np.random.randn(n)
data = [go.Histogram(x=x,
marker=dict(
color='#CC0E1D',# Lava (#CC0E1D)
color = 'rgb(200,0,0)' # you can provide color in HEX format or rgb format, genrally
<code>))]</code>
layout = go.Layout(title = "Histogram of {} random numbers".format(n))
fig = go.Figure(data= data, layout=layout)
iplot(fig)
```



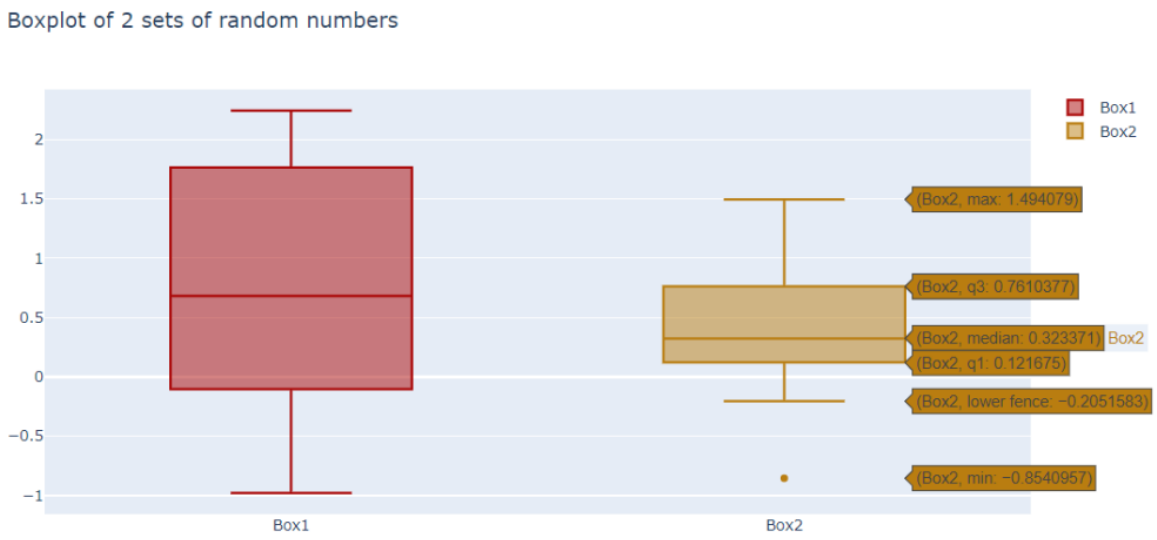
Boxplot in plotly

- 1 Numeric variable

```
from IPython.display import Image
Image("img/boxplot.png")
```



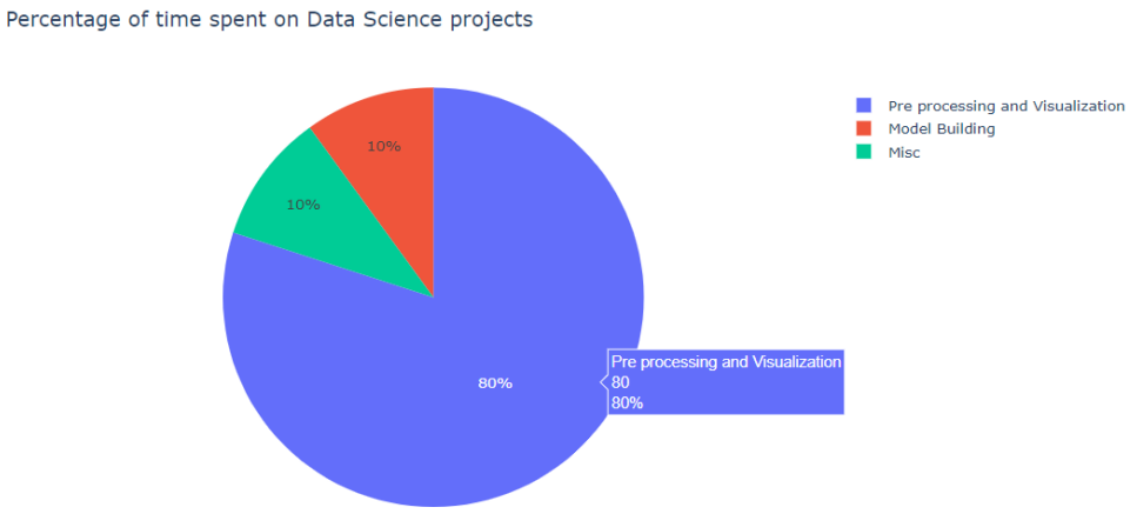
```
np.random.seed(0) # Set seed for reproducibility
n = 10
r1 = np.random.randn(n)
r2 = np.random.randn(n)
trace0 = go.Box(
y=r1,
name = 'Box1',
marker = dict(
color = '#AA0505',
)
)
trace1 = go.Box(
y=r2,
name = 'Box2',
marker = dict(
color = '#B97D10',
)
)
data = [trace0, trace1]
layout = go.Layout(title = "Boxplot of 2 sets of random numbers")
fig = go.Figure(data= data, layout=layout)
iplot(fig)
```



Pie chart in plotly

- 1 Categorical variable

```
labels = ["Pre processing and Visualization", "Model Building", "Misc"]
values = [80,10,10]
trace = go.Pie(labels=labels, values=values)
layout = go.Layout(title = 'Percentage of time spent on Data Science projects')
data = [trace]
fig = go.Figure(data= data,layout=layout)
iplot(fig)
```

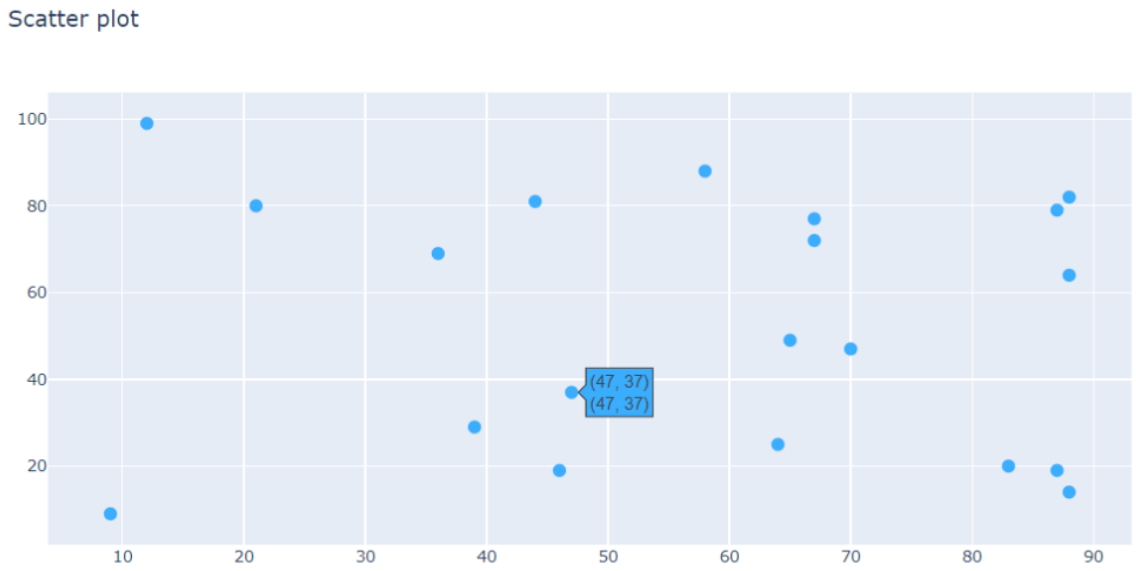


Note: We do not suggest you use pie chart, one reason being the total is not always obvious and second, having many levels will make the chart cluttered.

Scatter plot in plotly

- 2 numeric variables
- One x might have multiple corresponding y values

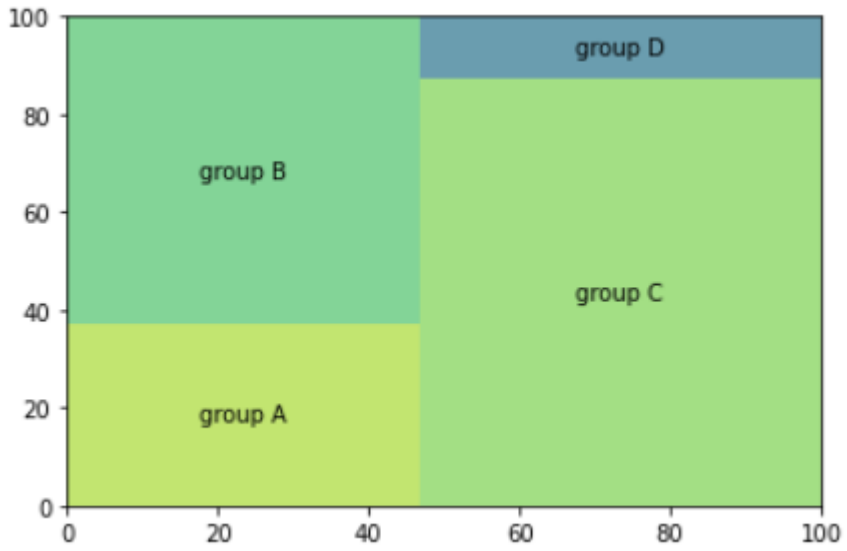
```
np.random.seed(0)
n = 20
x=np.random.randint(0,100,n)
y=np.random.randint(0,100,n)
data = [go.Scatter(x=x,y=y,
text = [str(i) for i in (zip(x,y))],
textposition = 'top center',
marker = dict(color = 'rgba(17, 157, 255, 0.8)',
size = 10), mode = 'markers')]
layout = go.Layout(title = 'Scatter plot')
fig = go.Figure(data= data,layout=layout)
iplot(fig)
```



Tree map

https://plot.ly/python/treemaps/

```
squarify.plot(sizes=[13,22,35,5], label=["group A", "group B", "group C",
"group D"], alpha=.7 )
plt.show()
```

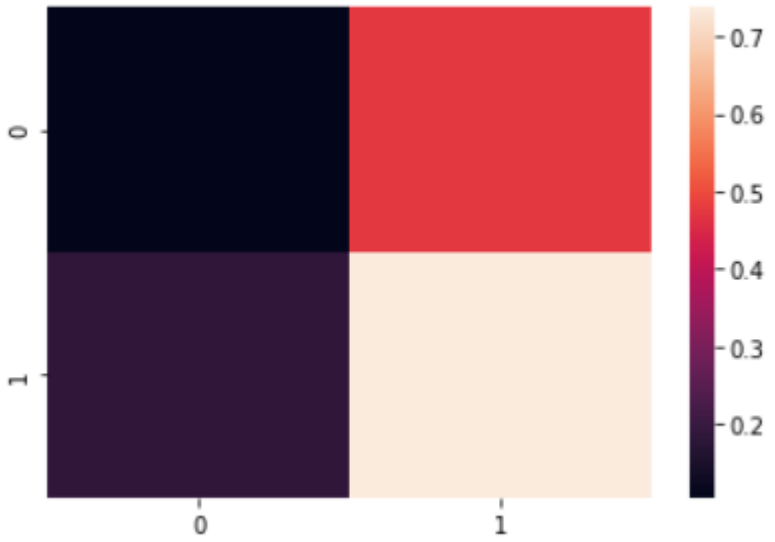


```
np.random.rand(2, 2)
```

```
array([[0.72063265, 0.58201979],
       [0.53737323, 0.75861562]])
```

```
# trace = go.Heatmap(z=[[1, 20], [22, 1]], x=['Monday', 'Tuesday'],y=['Morning', 'Af
# data=[trace]
# iplot(data)
sns.heatmap(np.random.rand(2, 2))
```

```
<AxesSubplot:>
```



Case Study

Now let us use our new found skill to extract insights from a dataset

hr_data Description

Education 1 ‘Below College’ 2 ‘College’ 3 ‘Bachelor’ 4 ‘Master’ 5 ‘Doctor’

EnvironmentSatisfaction 1 ‘Low’ 2 ‘Medium’ 3 ‘High’ 4 ‘Very High’

JobInvolvement 1 ‘Low’ 2 ‘Medium’ 3 ‘High’ 4 ‘Very High’

JobSatisfaction 1 ‘Low’ 2 ‘Medium’ 3 ‘High’ 4 ‘Very High’

PerformanceRating 1 ‘Low’ 2 ‘Good’ 3 ‘Excellent’ 4 ‘Outstanding’

RelationshipSatisfaction 1 ‘Low’ 2 ‘Medium’ 3 ‘High’ 4 ‘Very High’

WorkLifeBalance 1 ‘Bad’ 2 ‘Good’ 3 ‘Better’ 4 ‘Best’

```
1 hr_data = pd.read_csv("HR_Attrition.csv")
```

```
1 hr_data.head()
```

In [22]:
```
1 hr_data.head()
```

Out[22]:

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeNumber	EnvironmentSatisfaction	Gender	...	Relatic
0	41	Yes	Travel_Rarely	Sales	1	2	Life Sciences	1	2	Female	...	
1	49	No	Travel_Frequently	Research & Development	8	1	Life Sciences	2	3	Male	...	
2	37	Yes	Travel_Rarely	Research & Development	2	2	Other	4	4	Male	...	
3	33	No	Travel_Frequently	Research & Development	3	4	Life Sciences	5	4	Female	...	
4	27	No	Travel_Rarely	Research & Development	2	1	Medical	7	1	Male	...	

5 rows × 31 columns

Checking the datatypes

In [23]:
```
1 hr_data.dtypes
```

Out[23]:
```
Age                         int64
Attrition                  object
BusinessTravel             object
Department                 object
DistanceFromHome            int64
Education                   int64
EducationField             object
EmployeeNumber              int64
EnvironmentSatisfaction     int64
Gender                     object
JobInvolvement              int64
JobLevel                    int64
JobRole                    object
JobSatisfaction             int64
MaritalStatus              object
MonthlyIncome               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 [24]:
```
1 hr_data.describe(include='all')
```

Out[24]:

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeNumber	EnvironmentSatisfaction	Gend
count	1470.000000	1470	1470	1470	1470.000000	1470.000000	1470	1470.000000	1470.000000	147
unique	NaN	2	3	3	NaN	NaN	6	NaN	NaN	
top	NaN	No	Travel_Rarely	Research & Development	NaN	NaN	Life Sciences	NaN	NaN	Ma
freq	NaN	1233	1043	961	NaN	NaN	606	NaN	NaN	88
mean	36.923810	NaN	NaN	NaN	9.192517	2.912925	NaN	1024.865306	2.721769	Na
std	9.135373	NaN	NaN	NaN	8.106864	1.024165	NaN	602.024335	1.093082	Na
min	18.000000	NaN	NaN	NaN	1.000000	1.000000	NaN	1.000000	1.000000	Na
25%	30.000000	NaN	NaN	NaN	2.000000	2.000000	NaN	491.250000	2.000000	Na
50%	36.000000	NaN	NaN	NaN	7.000000	3.000000	NaN	1020.500000	3.000000	Na
75%	43.000000	NaN	NaN	NaN	14.000000	4.000000	NaN	1555.750000	4.000000	Na
max	60.000000	NaN	NaN	NaN	29.000000	5.000000	NaN	2068.000000	4.000000	Na

11 rows × 31 columns

```
In [25]: 1 hr_data.isnull().sum()
```

```
Out[25]: Age                        0
         Attrition                  0
         BusinessTravel             0
         Department                 0
         DistanceFromHome           0
         Education                  0
         EducationField             0
         EmployeeNumber             0
         EnvironmentSatisfaction    0
         Gender                     0
         JobInvolvement             0
         JobLevel                   0
         JobRole                    0
         JobSatisfaction            0
         MaritalStatus              0
         MonthlyIncome              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
```

Checking the number of unique values in each column

```
1 for i in hr_data.columns:
2     print ("Number of unique values in {} column are {} \n
3 The unique values are {}".format(i, len(hr_data[i].unique()),hr_data[i].unique()))
4     print ("---------------------- \n")
```

```
Number of unique values in Age column are 43
 The unique values are [41 49 37 33 27 32 59 30 38 36 35 29 31 34 28 22 53 24 21 42 44 46 39 43
 50 26 48 55 45 56 23 51 40 54 58 20 25 19 57 52 47 18 60]
----------------------

Number of unique values in Attrition column are 2
 The unique values are ['Yes' 'No']
----------------------

Number of unique values in BusinessTravel column are 3
 The unique values are ['Travel_Rarely' 'Travel_Frequently' 'Non-Travel']
----------------------

Number of unique values in Department column are 3
 The unique values are ['Sales' 'Research & Development' 'Human Resources']
----------------------

Number of unique values in DistanceFromHome column are 29
 The unique values are [ 1  8  2  3 24 23 27 16 15 26 19 21  5 11  9  7  6 10  4 25 12 18 29 22
 14 20 28 17 13]
----------------------

Number of unique values in Education column are 5
 The unique values are [2 1 4 3 5]
----------------------

Number of unique values in EducationField column are 6
 The unique values are ['Life Sciences' 'Other' 'Medical' 'Marketing' 'Technical Degree'
 'Human Resources']
----------------------

Number of unique values in EmployeeNumber column are 1470
 The unique values are [   1    2    4 ... 2064 2065 2068]
----------------------

Number of unique values in EnvironmentSatisfaction column are 4
 The unique values are [2 3 4 1]
----------------------

Number of unique values in Gender column are 2
 The unique values are ['Female' 'Male']
----------------------
```

```
Number of unique values in JobInvolvement column are 4
 The unique values are [3 2 4 1]
----------------------

Number of unique values in JobLevel column are 5
 The unique values are [2 1 3 4 5]
----------------------

Number of unique values in JobRole column are 9
 The unique values are ['Sales Executive' 'Research Scientist' 'Laboratory Technician'
 'Manufacturing Director' 'Healthcare Representative' 'Manager'
 'Sales Representative' 'Research Director' 'Human Resources']
----------------------

Number of unique values in JobSatisfaction column are 4
 The unique values are [4 2 3 1]
----------------------

Number of unique values in MaritalStatus column are 3
 The unique values are ['Single' 'Married' 'Divorced']
----------------------

Number of unique values in MonthlyIncome column are 1349
 The unique values are [5993 5130 2090 ... 9991 5390 4404]
----------------------

Number of unique values in NumCompaniesWorked column are 10
 The unique values are [8 1 6 9 0 4 5 2 7 3]
----------------------

Number of unique values in Over18 column are 1
 The unique values are ['Y']
----------------------

Number of unique values in OverTime column are 2
 The unique values are ['Yes' 'No']
----------------------

Number of unique values in PercentSalaryHike column are 15
 The unique values are [11 23 15 12 13 20 22 21 17 14 16 18 19 24 25]
----------------------


  Number of unique values in PerformanceRating column are 2
   The unique values are [3 4]
  ----------------------

  Number of unique values in RelationshipSatisfaction column are 4
   The unique values are [1 4 2 3]
  ----------------------

  Number of unique values in StandardHours column are 1
   The unique values are [80]
  ----------------------

  Number of unique values in StockOptionLevel column are 4
   The unique values are [0 1 3 2]
  ----------------------

  Number of unique values in TotalWorkingYears column are 40
   The unique values are [ 8 10  7  6 12  1 17  5  3 31 13  0 26 24 22  9 19  2 23 14 15  4 29 28
   21 25 20 11 16 37 38 30 40 18 36 34 32 33 35 27]
  ----------------------

  Number of unique values in TrainingTimesLastYear column are 7
   The unique values are [0 3 2 5 1 4 6]
  ----------------------

  Number of unique values in WorkLifeBalance column are 4
   The unique values are [1 3 2 4]
  ----------------------

  Number of unique values in YearsAtCompany column are 37
   The unique values are [ 6 10  0  8  2  7  1  9  5  4 25  3 12 14 22 15 27 21 17 11 13 37 16 20
   40 24 33 19 36 18 29 31 32 34 26 30 23]
  ----------------------

  Number of unique values in YearsInCurrentRole column are 19
   The unique values are [ 4  7  0  2  5  9  8  3  6 13  1 15 14 16 11 10 12 18 17]
  ----------------------


 Number of unique values in YearsSinceLastPromotion column are 16
  The unique values are [ 0  1  3  2  7  4  8  6  5 15  9 13 12 10 11 14]
 ----------------------

 Number of unique values in YearsWithCurrManager column are 18
  The unique values are [ 5  7  0  2  6  8  3 11 17  1  4 12  9 10 15 13 16 14]
 ----------------------
```

Observations:

Most columns have fewer than 4 unique levels

NumCompaniesWorked and PercentSalaryHike have less than 15 values and we can convert these into categorical values for analysis purposes,

this is fairly subjective. You can also continue with these as integer values.

Replacing the integers with above values with the values in the description

- hr_data.Education = hr_data.Education.replace(to_replace=[1,2,3,4,5],value=['Below College', 'College', 'Bachelor', 'Master', 'Doctor'])
- hr_data.EnvironmentSatisfaction = hr_data.EnvironmentSatisfaction.replace(to_replace=[1,2,3,4],value=['Low', 'Medium', 'High', 'Very High'])
- hr_data.JobInvolvement = hr_data.JobInvolvement.replace(to_replace=[1,2,3,4],value=['Low', 'Medium', 'High', 'Very High'])
- hr_data.JobSatisfaction = hr_data.JobSatisfaction.replace(to_replace=[1,2,3,4],value=['Low', 'Medium', 'High', 'Very High'])
- hr_data.PerformanceRating = hr_data.PerformanceRating.replace(to_replace=[1,2,3,4],value=['Low', 'Good', 'Excellent', 'Outstanding'])
- hr_data.RelationshipSatisfaction = hr_data.RelationshipSatisfaction.replace(to_replace=[1,2,3,4],value=['Low', 'Medium', 'High', 'Very High'])
- hr_data.WorkLifeBalance = hr_data.WorkLifeBalance.replace(to_replace=[1,2,3,4],value=['Bad', 'Good', 'Better', 'Best'])

```
Education_dict = {1:'Below College',
2:'College',
3:'Bachelor',
4:'Master',
5:'Doctor',
}
EnvironmentSatisfaction_dict = {1:'Low',
2:'Medium',
3:'High',
4:'Very High',
}
JobInvolvement_dict = {1:'Low',
2:'Medium',
3:'High',
4:'Very High',
}
JobSatisfaction_dict = {1:'Low',
2:'Medium',
3:'High',
4:'Very High',
}
PerformanceRating_dict = {1:'Low',
2:'Good',
3:'Excellent',
4:'Outstanding',
}
RelationshipSatisfaction_dict = {1:'Low',
2:'Medium',
3:'High',
4:'Very High',
}
WorkLifeBalance_dict = {1:'Bad',
2:'Good',
3:'Better',
4:'Best',
}
```

```
hr_data = hr_data.replace({
"Education":Education_dict,
"EnvironmentSatisfaction":EnvironmentSatisfaction_dict,
"JobInvolvement":JobInvolvement_dict,
"JobSatisfaction":JobSatisfaction_dict,
"PerformanceRating":PerformanceRating_dict,
"RelationshipSatisfaction":RelationshipSatisfaction_dict,
"WorkLifeBalance":WorkLifeBalance_dict
})
```

Extract categorical columns

Columns with 15 or less levels are considered as categorical columns for the purpose of this analysis

We have decided to treat all the columns with 15 or less levels as categorical columns, the following few lines of code extract all the columns which satisfy the condition.

```
cat_cols = []
for i in hr_data.columns:
if hr_data[i].dtype =='object' or len(np.unique(hr_data[i]))<=15 :
# if the number of levels is less that 15 considering the column
as categorial
cat_cols.append(i)
print("{} : {} : {} ".format(i,len(np.unique(hr_data[i])),np.unique(hr_data[i])))
```

```
Attrition : 2 : ['No' 'Yes']
BusinessTravel : 3 : ['Non-Travel' 'Travel_Frequently' 'Travel_Rarely']
Department : 3 : ['Human Resources' 'Research & Development' 'Sales']
Education : 5 : ['Bachelor' 'Below College' 'College' 'Doctor' 'Master']
EducationField : 6 : ['Human Resources' 'Life Sciences' 'Marketing' 'Medical' 'Other'
 'Technical Degree']
EnvironmentSatisfaction : 4 : ['High' 'Low' 'Medium' 'Very High']
Gender : 2 : ['Female' 'Male']
JobInvolvement : 4 : ['High' 'Low' 'Medium' 'Very High']
JobLevel : 5 : [1 2 3 4 5]
JobRole : 9 : ['Healthcare Representative' 'Human Resources' 'Laboratory Technician'
 'Manager' 'Manufacturing Director' 'Research Director'
 'Research Scientist' 'Sales Executive' 'Sales Representative']
JobSatisfaction : 4 : ['High' 'Low' 'Medium' 'Very High']
MaritalStatus : 3 : ['Divorced' 'Married' 'Single']
NumCompaniesWorked : 10 : [0 1 2 3 4 5 6 7 8 9]
Over18 : 1 : ['Y']
OverTime : 2 : ['No' 'Yes']
PercentSalaryHike : 15 : [11 12 13 14 15 16 17 18 19 20 21 22 23 24 25]
PerformanceRating : 2 : ['Excellent' 'Outstanding']
RelationshipSatisfaction : 4 : ['High' 'Low' 'Medium' 'Very High']
StandardHours : 1 : [80]
StockOptionLevel : 4 : [0 1 2 3]
TrainingTimesLastYear : 7 : [0 1 2 3 4 5 6]
WorkLifeBalance : 4 : ['Bad' 'Best' 'Better' 'Good']
```

Print the categorical column names

```
In [30]: 1 cat_cols
```

```
Out[30]: ['Attrition',
          'BusinessTravel',
          'Department',
          'Education',
          'EducationField',
          'EnvironmentSatisfaction',
          'Gender',
          'JobInvolvement',
          'JobLevel',
          'JobRole',
          'JobSatisfaction',
          'MaritalStatus',
          'NumCompaniesWorked',
          'Over18',
          'OverTime',
          'PercentSalaryHike',
          'PerformanceRating',
          'RelationshipSatisfaction',
          'StandardHours',
          'StockOptionLevel',
          'TrainingTimesLastYear',
          'WorkLifeBalance']
```

Check if the above columns are categorical in the data set

```
In [31]: 1 hr_data[cat_cols].dtypes
```

```
Out[31]: Attrition                 object
         BusinessTravel            object
         Department                object
         Education                 object
         EducationField            object
         EnvironmentSatisfaction   object
         Gender                    object
         JobInvolvement            object
         JobLevel                   int64
         JobRole                   object
         JobSatisfaction           object
         MaritalStatus             object
         NumCompaniesWorked         int64
         Over18                    object
         OverTime                  object
         PercentSalaryHike          int64
         PerformanceRating         object
         RelationshipSatisfaction  object
         StandardHours              int64
         StockOptionLevel           int64
         TrainingTimesLastYear      int64
         WorkLifeBalance           object
         dtype: object
```

Type Conversion

- n dimensional type conversion to ‘category’ is not implemented yet

```
for i in cat_cols:
hr_data[i] = hr_data[i].astype('category')
```

Categorical attributes summary

In [33]:
```
hr_data.describe(include='all').transpose()
```

Out[33]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Age	1470	NaN	NaN	NaN	36.9238	9.13537	18	30	36	43	60
Attrition	1470	2	No	1233	NaN	NaN	NaN	NaN	NaN	NaN	NaN
BusinessTravel	1470	3	Travel_Rarely	1043	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Department	1470	3	Research & Development	961	NaN	NaN	NaN	NaN	NaN	NaN	NaN
DistanceFromHome	1470	NaN	NaN	NaN	9.19252	8.10686	1	2	7	14	29
Education	1470	5	Bachelor	572	NaN	NaN	NaN	NaN	NaN	NaN	NaN
EducationField	1470	6	Life Sciences	606	NaN	NaN	NaN	NaN	NaN	NaN	NaN
EmployeeNumber	1470	NaN	NaN	NaN	1024.87	602.024	1	491.25	1020.5	1555.75	2068
EnvironmentSatisfaction	1470	4	High	453	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Gender	1470	2	Male	882	NaN	NaN	NaN	NaN	NaN	NaN	NaN
JobInvolvement	1470	4	High	868	NaN	NaN	NaN	NaN	NaN	NaN	NaN
JobLevel	1470	5	1	543	NaN	NaN	NaN	NaN	NaN	NaN	NaN
JobRole	1470	9	Sales Executive	326	NaN	NaN	NaN	NaN	NaN	NaN	NaN
JobSatisfaction	1470	4	Very High	459	NaN	NaN	NaN	NaN	NaN	NaN	NaN
MaritalStatus	1470	3	Married	673	NaN	NaN	NaN	NaN	NaN	NaN	NaN
MonthlyIncome	1470	NaN	NaN	NaN	6502.93	4707.96	1009	2911	4919	8379	19999
NumCompaniesWorked	1470	10	1	521	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Over18	1470	1	Y	1470	NaN	NaN	NaN	NaN	NaN	NaN	NaN
OverTime	1470	2	No	1054	NaN	NaN	NaN	NaN	NaN	NaN	NaN
PercentSalaryHike	1470	15	11	210	NaN	NaN	NaN	NaN	NaN	NaN	NaN
PerformanceRating	1470	2	Excellent	1244	NaN	NaN	NaN	NaN	NaN	NaN	NaN
RelationshipSatisfaction	1470	4	High	459	NaN	NaN	NaN	NaN	NaN	NaN	NaN
StandardHours	1470	1	80	1470	NaN	NaN	NaN	NaN	NaN	NaN	NaN
StockOptionLevel	1470	4	0	631	NaN	NaN	NaN	NaN	NaN	NaN	NaN
TotalWorkingYears	1470	NaN	NaN	NaN	11.2796	7.78078	0	6	10	15	40
TrainingTimesLastYear	1470	7	2	547	NaN	NaN	NaN	NaN	NaN	NaN	NaN
WorkLifeBalance	1470	4	Better	893	NaN	NaN	NaN	NaN	NaN	NaN	NaN
YearsAtCompany	1470	NaN	NaN	NaN	7.00816	6.12653	0	3	5	9	40
YearsInCurrentRole	1470	NaN	NaN	NaN	4.22925	3.62314	0	2	3	7	18
YearsSinceLastPromotion	1470	NaN	NaN	NaN	2.18776	3.22243	0	0	1	3	15
YearsWithCurrManager	1470	NaN	NaN	NaN	4.12313	3.56814	0	2	3	7	17

Extracting Numeric Columns

```
num_cols = [i for i in hr_data.columns if i not in cat_cols]
```

```
hr_data[cat_cols].dtypes
```

```
Attrition                 category
BusinessTravel            category
Department                category
Education                 category
EducationField            category
EnvironmentSatisfaction   category
Gender                    category
JobInvolvement            category
JobLevel                  category
JobRole                   category
JobSatisfaction           category
MaritalStatus             category
NumCompaniesWorked        category
Over18                    category
OverTime                  category
PercentSalaryHike         category
PerformanceRating         category
RelationshipSatisfaction  category
StandardHours             category
StockOptionLevel          category
TrainingTimesLastYear     category
WorkLifeBalance           category
dtype: object
```

```
hr_data.head()
```

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeNumber	EnvironmentSatisfaction	Gender	...	Relatic
0	41	Yes	Travel_Rarely	Sales	1	College	Life Sciences	1	Medium	Female	...	
1	49	No	Travel_Frequently	Research & Development	8	Below College	Life Sciences	2	High	Male	...	
2	37	Yes	Travel_Rarely	Research & Development	2	College	Other	4	Very High	Male	...	
3	33	No	Travel_Frequently	Research & Development	3	Master	Life Sciences	5	Very High	Female	...	
4	27	No	Travel_Rarely	Research & Development	2	Below College	Medical	7	Low	Male	...	

5 rows × 31 columns

Exploratory Data Analysis

Univariate Analysis

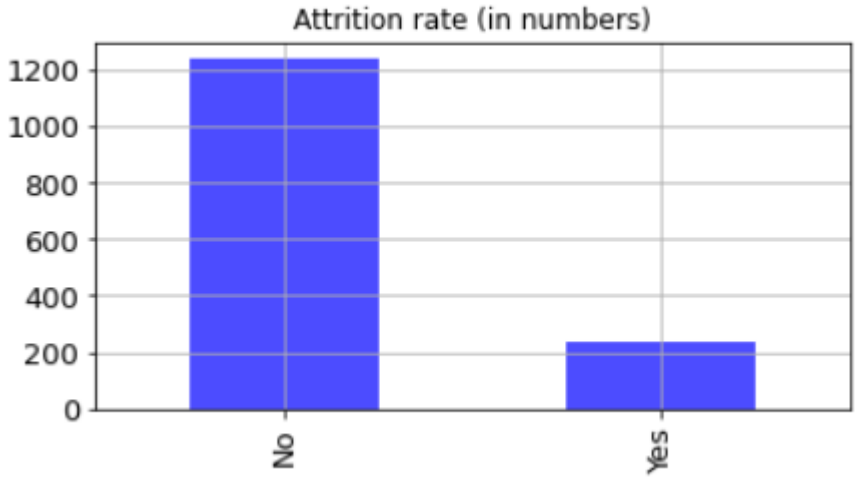
1. What is the attrition rate in the company?

Attrition in numbers (pandas)

```
hr_data.Attrition.value_counts()
```

```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

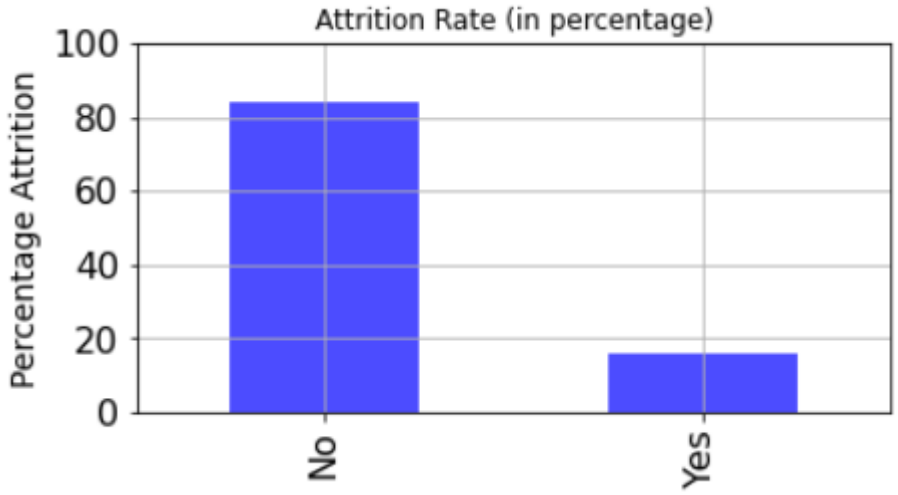
```
plt.figure()
hr_data.Attrition.value_counts().plot(kind='bar',
figsize=(6,3), color="blue", alpha = 0.7, fontsize=13)
plt.title('Attrition rate (in numbers)')
plt.grid()
plt.show()
```



This is one way to tell matplotlib to plot the graphs in the notebook

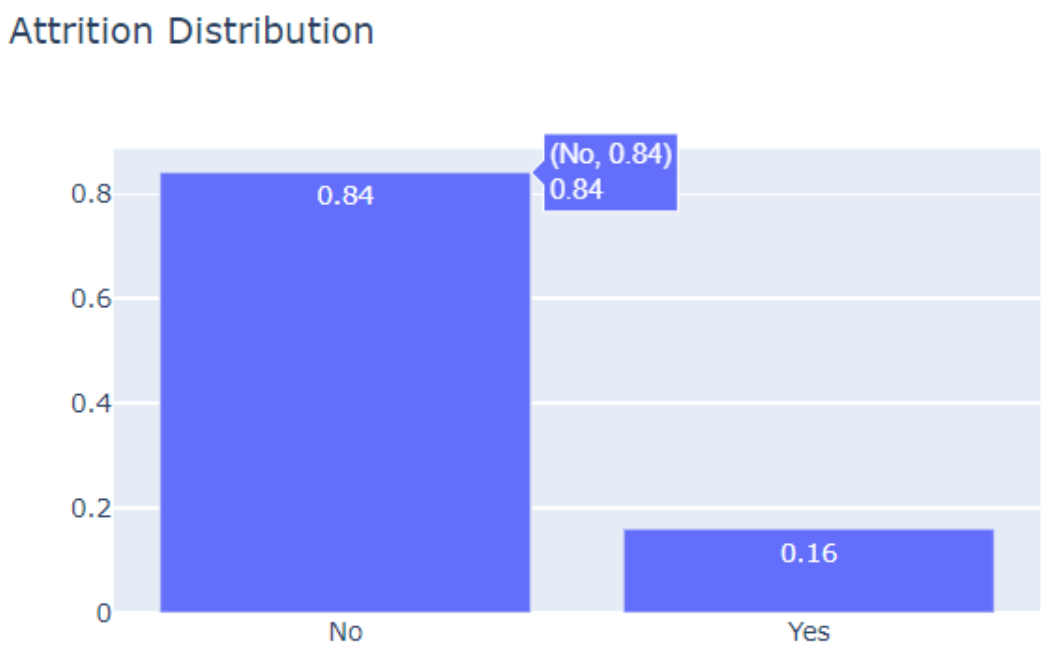
Attrition rate in percentage (pandas)

```
((hr_data.Attrition.value_counts()/sum(hr_data.Attrition.value_counts()))*100).plot(
kind='bar', figsize=(6,3), color=["blue"], alpha = 0.7, fontsize=16)
plt.ylim([0,100])
plt.title('Attrition Rate (in percentage)')
plt.ylabel('Percentage Attrition',fontsize = 14)
plt.grid(True)
plt.show()
```



plotly In percentages

```
temp = hr_data.Attrition.value_counts()
trace = go.Bar(x=temp.index,
y= np.round(temp.astype(float)/temp.values.sum(),2),
text = np.round(temp.astype(float)/temp.values.sum(),2),
textposition = 'auto',
name = 'Attrition')
data = [trace]
layout = go.Layout(autosize=False, width=600, height=400,title =
"Attrition Distribution"
)
fig = go.Figure(data=data, layout=layout)
iplot(fig)
del temp
```

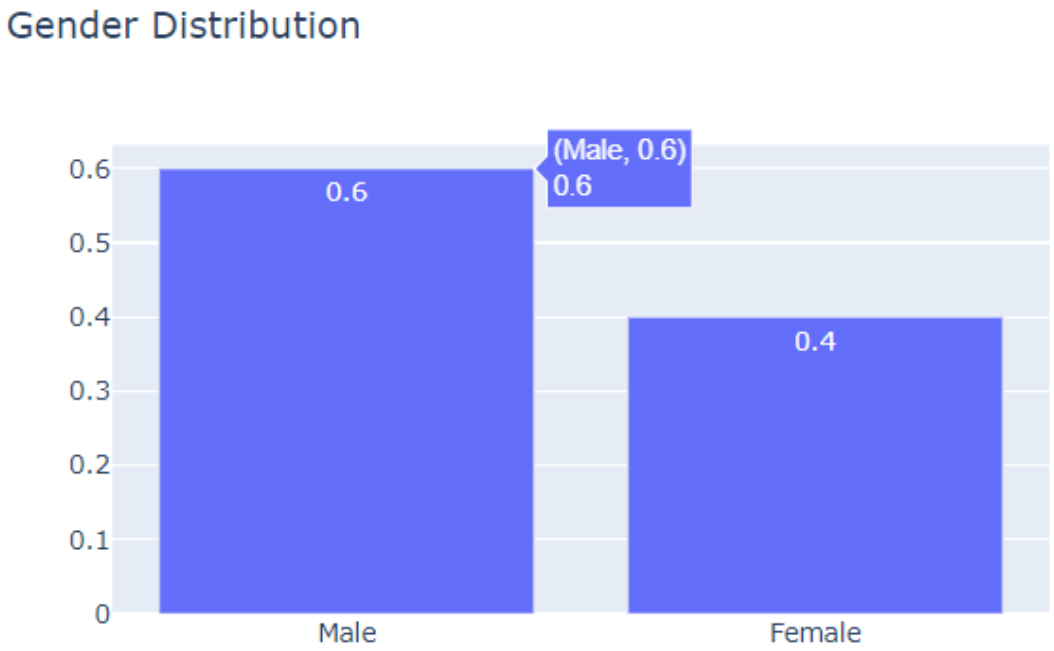


2. What is the Gender Distribution in the company?

```
temp = hr_data.Gender.value_counts()
temp
```

```
Male      882
Female    588
Name: Gender, dtype: int64
```

```
data = [go.Bar(
x=temp.index,
y= np.round(temp.astype(float)/temp.values.sum(),2),
text = np.round(temp.astype(float)/temp.values.sum(),2),
textposition = 'auto',
)]
layout = go.Layout(
autosize=False,
width=600,
height=400,title = "Gender Distribution",
)
fig = go.Figure(data=data, layout=layout)
iplot(fig)
del temp
```




```
temp = hr_data.Gender.value_counts()
temp
```

```
Male      882
Female    588
Name: Gender, dtype: int64
```

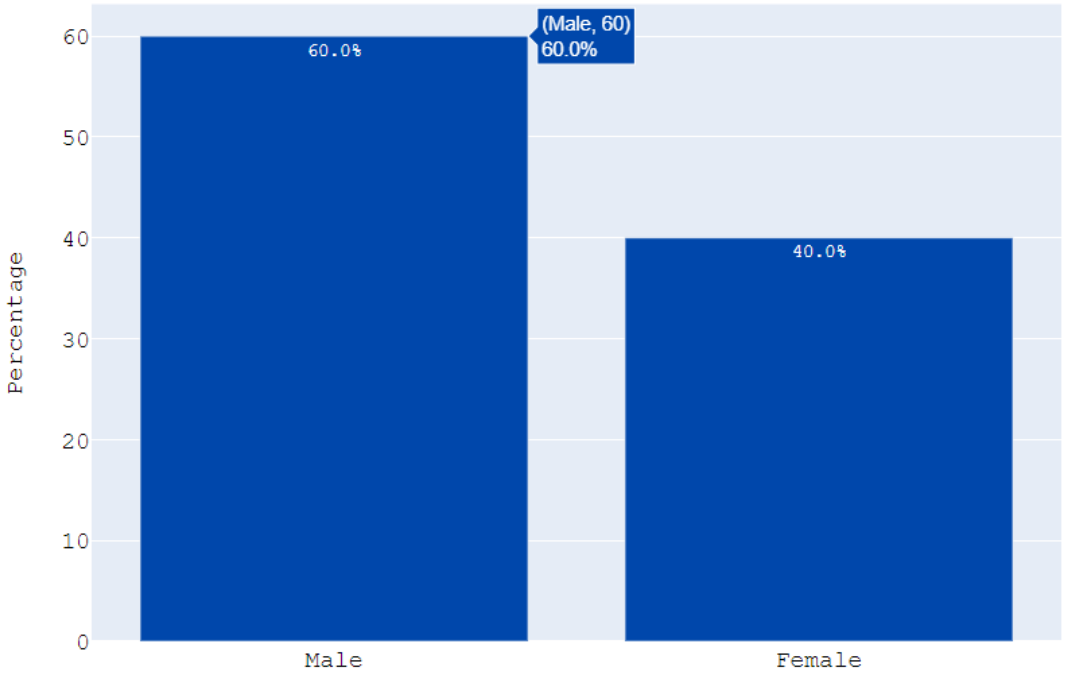
Steps to create a bar chart with counts for a categorical variable in plotly

- Steps to create a bar chart with counts for a categorical variable
  - create an object and store the counts (optional)
  - create a bar object
    - pass the x values
    - pass the y values
    - optional :
      - text to be displayed
      - text position
      - color of the bar
      - name of the bar (trace in plotly terminology)
  - create a layout object
    - title – font and size of title
    - x axis – font and size of xaxis text
    - y axis – font and size of yaxis text
  - create a figure object:

- add data
- add layout

- plot the figure object

```
# create a table with value counts
temp = hr_data.Gender.value_counts()
# creating a Bar chart object of plotly
data = [go.Bar(
x=temp.index.astype(str), # x axis values
y=np.round(temp.values.astype(float)/temp.values.sum(),4)100,
text=['{}%'.format(i) for i in
np.round(temp.values.astype(float)/temp.values.sum(),4)100],
textposition = 'auto', # specify at which position on the bar the text should appea
marker = dict(color = '#0047AB'),)] # change color of the bar
# color used here Cobalt Blue
# these are used to define the layout options
layout = go.Layout(
autosize=False, # auto size the graph? use False if you are specifying the height a
width=800, # height of the figure in pixels
height=600, # height of the figure in pixels
title = "Distribution of {} column".format('Gender'), # title of the figure
# more granular control on the title font
titlefont=dict(
family='Courier New, monospace', # font family
size=16, # size of the font
color='black' # color of the font
),
# granular control on the axes objects
xaxis=dict(
tickfont=dict(
family='Courier New, monospace', # font family
size=16, # size of ticks displayed on the x axis
color='black' # color of the font
)
),
yaxis=dict(
title='Percentage',
titlefont=dict(
size=16,
color='black'
),
tickfont=dict(
family='Courier New, monospace', # font family
size=16, # size of ticks displayed on the y axis
color='black' # color of the font
)
),
font = dict(
family='Courier New, monospace', # font family
color = "white",# color of the font
size = 12 # size of the font displayed on the bar
)
)
fig = go.Figure(data=data, layout=layout)
iplot(fig)
del temp
```



We will save the above layout in an object and define a function for future use

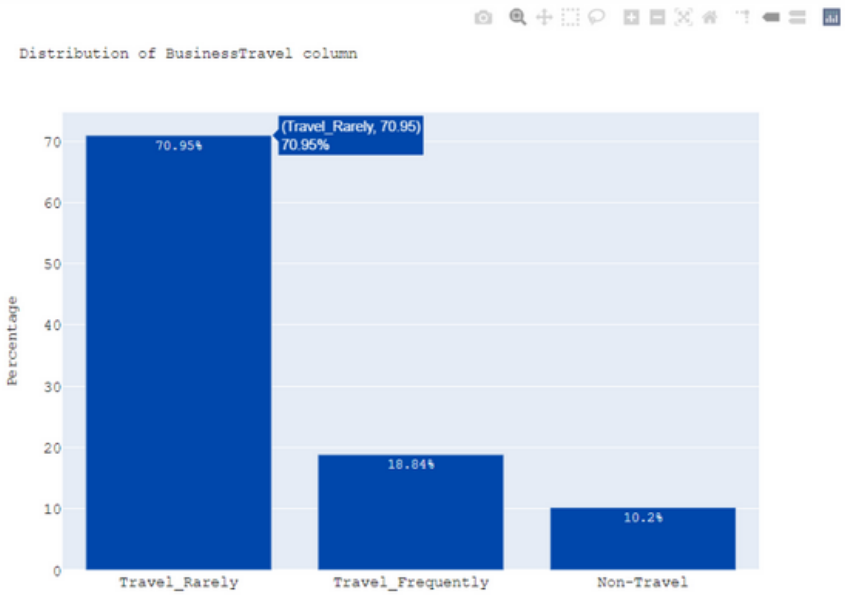
```
def generate_layout_bar(col_name):
    layout_bar = go.Layout(
    autosize=False,
# auto size the graph? use False if you are specifying the height and width
    width=800, # height of the figure in pixels
    height=600, # height of the figure in pixels
    title = "Distribution of {} column".format(col_name), # title of the figure
# more granular control on the title font
    titlefont=dict(
    family='Courier New, monospace', # font family
    size=14, # size of the font
    color='black' # color of the font
   ),
# granular control on the axes objects
    xaxis=dict(
    tickfont=dict(
    family='Courier New, monospace', # font family
    size=14, # size of ticks displayed on the x axis
    color='black' # color of the font
       )
    ),
    yaxis=dict(title='Percentage',titlefont=dict(size=14, color='black'),
    tickfont=dict(family='Courier New, monospace', size = 14, color='black')),
    font=dict(family='Courier New, monospace', color = "white", size = 12))

    return layout_bar
```

Defining a function to plot the bar charts

```
def plot_bar(col_name):
# create a table with value counts
    temp = hr_data[col_name].value_counts()
# creating a Bar chart object of plotly
    data = [go.Bar(
        x=temp.index.astype(str),
        y=np.round(temp.values.astype(float)/temp.values.sum(),4)100,
        text = ['{}%'.format(i) for i in
np.round(temp.values.astype(float)/temp.values.sum(),4)100],
        textposition = 'auto',
# specify at which position on the bar the text should appear
        marker = dict(color = '#0047AB'),)]
    layout_bar = generate_layout_bar(col-name=col_name)
    fig = go.Figure(data = data, layout=layout_bar)
    return iplot(fig)
```

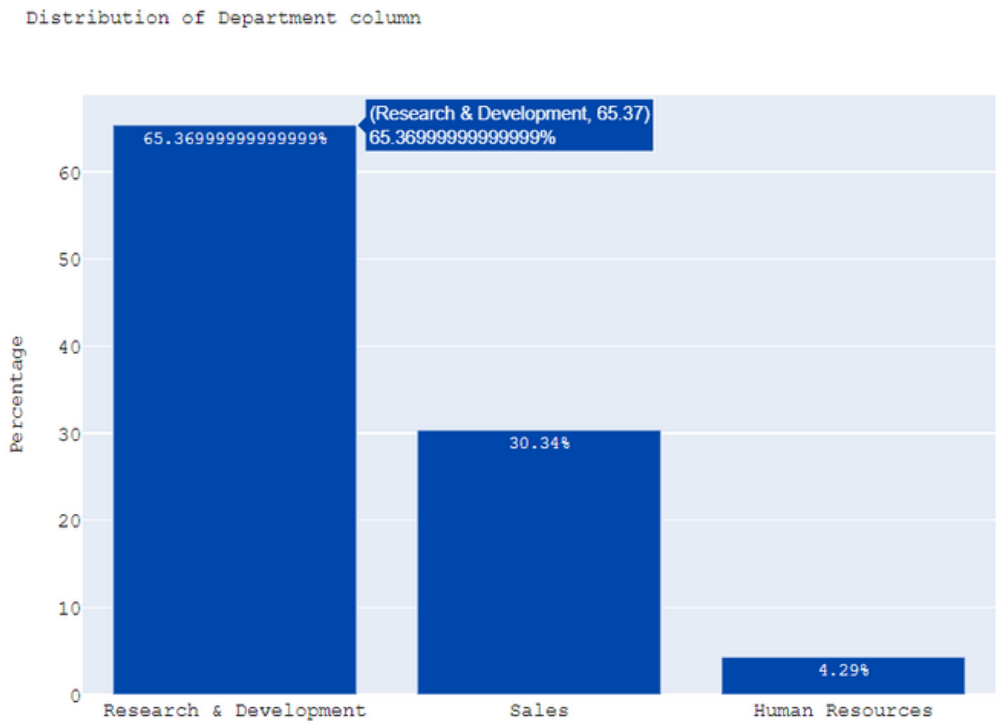
3.How many people travel? (Business Travel)

```
In [47]: plot_bar('BusinessTravel')
```



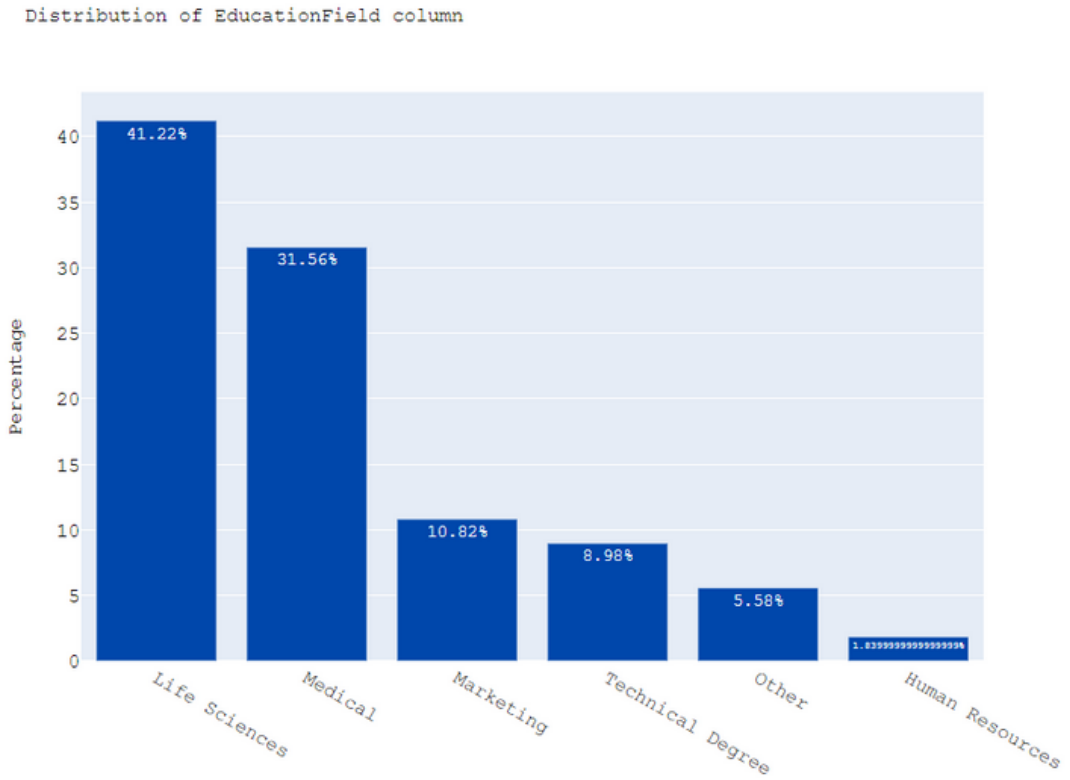
4. Which department has the highest number of employees? (Department)

```
1 plot_bar('Department')
```



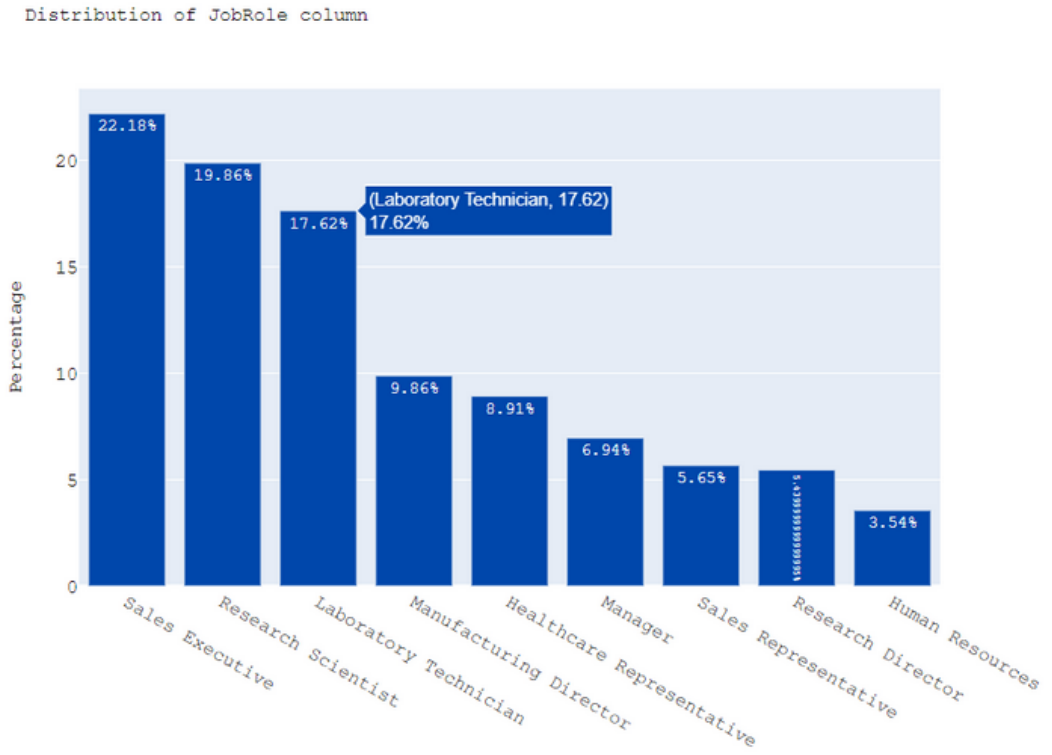
5. What is the most common educational background of the employees (Education Field)

```
1 plot_bar('EducationField')
```



6. In what roles are the employees working and what is the common job role? (Job Role)

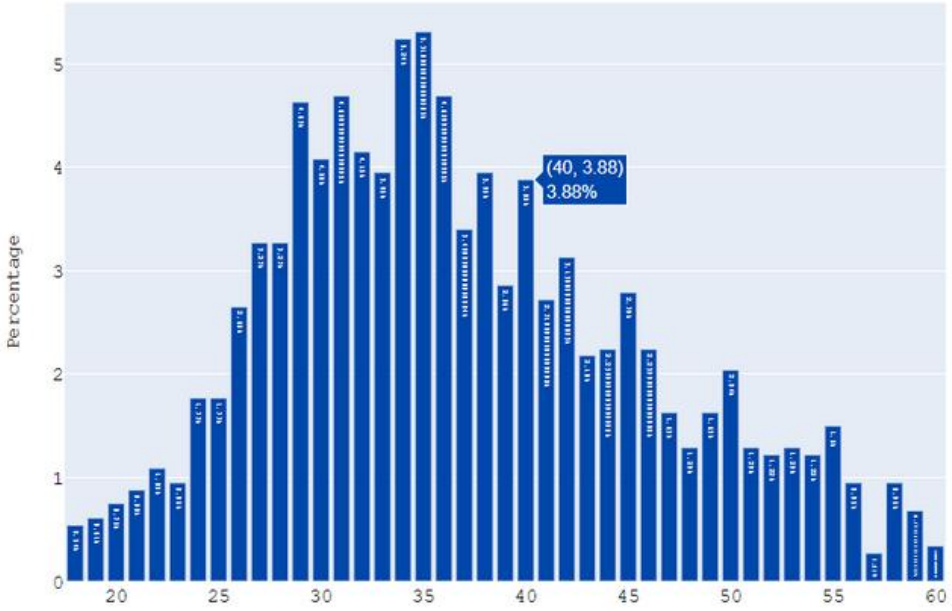
```
1 plot_bar('JobRole')
```



7. Is the workforce in the company young? (Age)

```
1 plot_bar('Age')
```

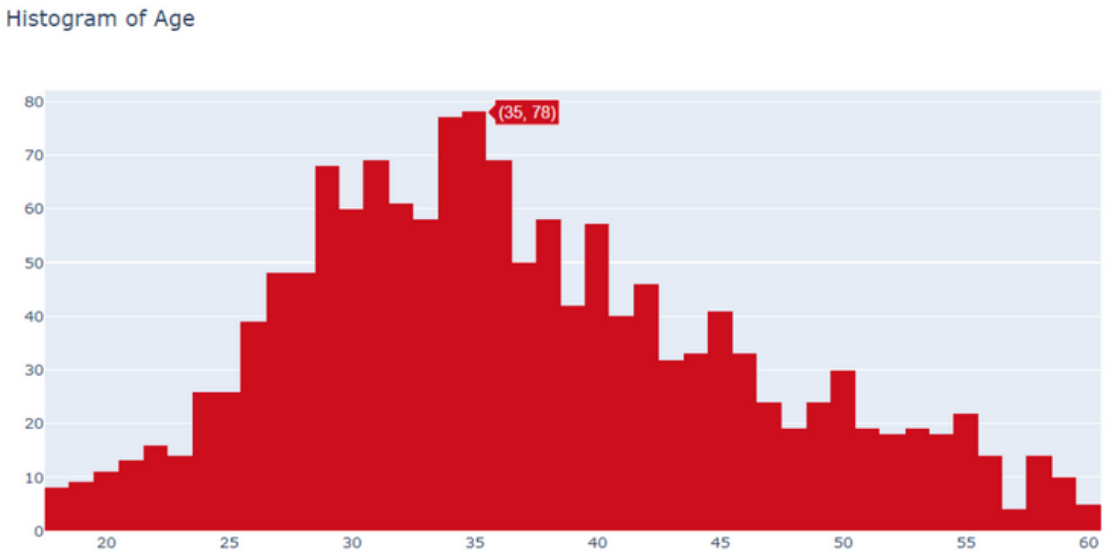
```
Distribution of Age column
```



Age is a continuous variable, it makes more sense to plot a histogram rather than a bar chart

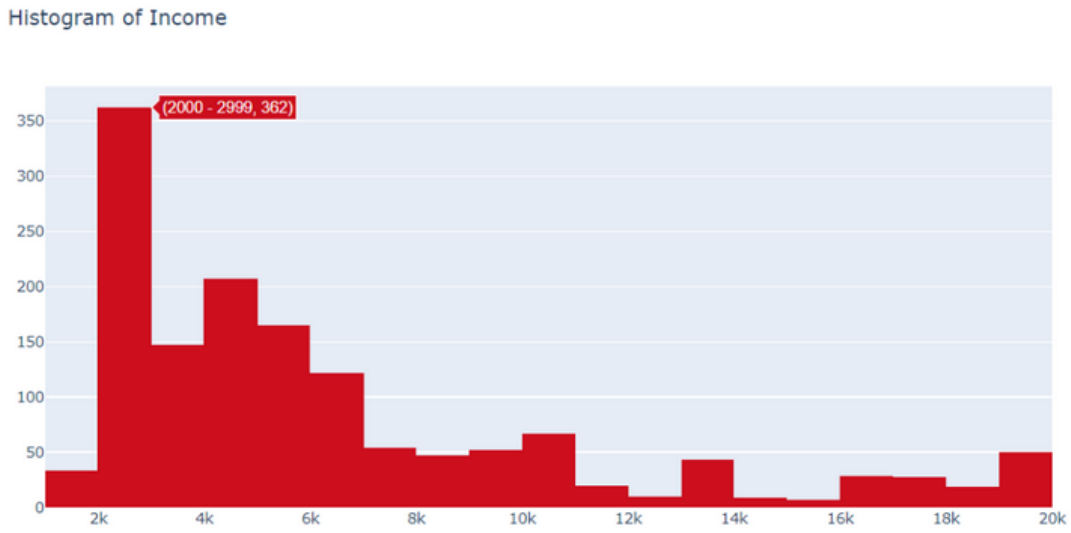
Histogram

```
data = [go.Histogram(x=hr_data.Age,marker=dict(color='#CC0E1D'<code>))]</code>
layout = go.Layout(title = "Histogram of Age".format(n))
fig = go.Figure(data= data, layout=layout)
iplot(fig)
```



8. What is the income distribution in the company? (Monthly Income)

```
data = [go.Histogram(x=hr_data.MonthlyIncome,
marker=dict(
color='#CC0E1D'))]
layout = go.Layout(title = "Histogram of Income".format(n))
fig = go.Figure(data= data, layout=layout)
iplot(fig
```



Observations:

```
- We see that the income column has a long tailed distribution
- Binning might give better insights into the distribution
```

Let us bin the Income column

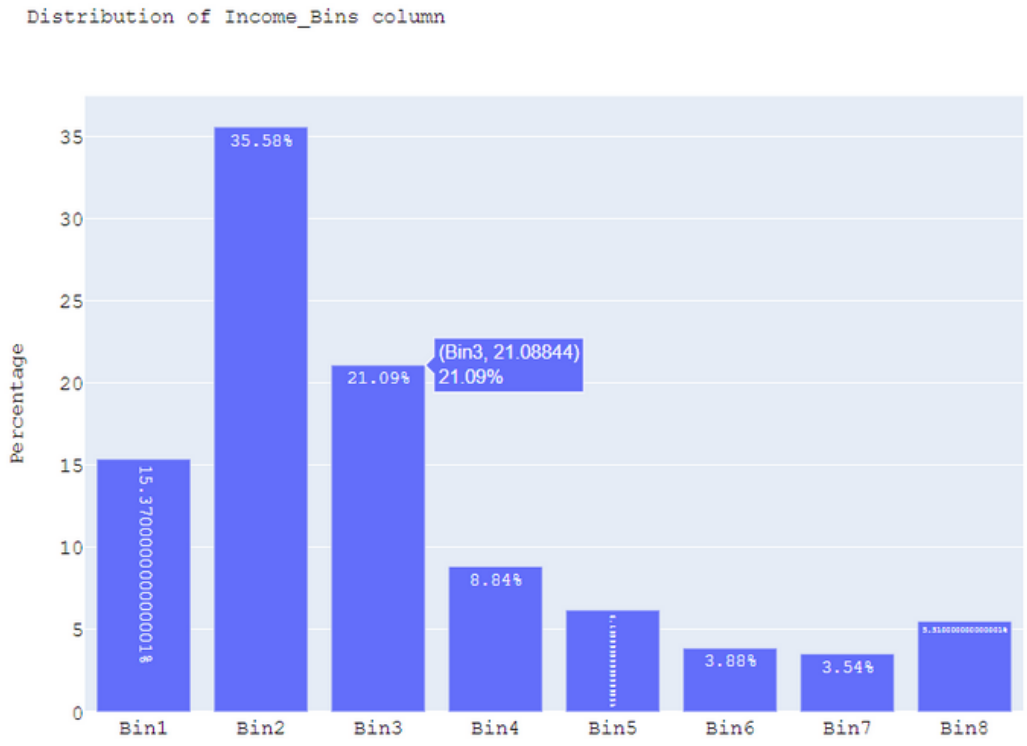
```
hr_data['Income_Bins'] = np.digitize(hr_data.MonthlyIncome,
list(range(0,hr_data.MonthlyIncome.max()+10,2500)),right=True)

list(range(0,hr_data.MonthlyIncome.max()+10,2500))

hr_data['Income_Bins'].value_counts()
```

```
2    523
3    310
1    226
4    130
5     91
8     81
6     57
7     52
Name: Income_Bins, dtype: int64
```

```
hr_data['Income_Bins'] = hr_data['Income_Bins'].replace(to_replace=[1,2,3,4,5,6,7,8],
value=['Bin1','Bin2','Bin3',
'Bin4','Bin5','Bin6','Bin7','Bin8'])
```

```
temp = hr_data['Income_Bins'].value_counts()
temp=temp.sort_index()
```

```
trace1 = go.Bar(x = temp.index,
y=(temp.values.astype(float)/sum(temp.values))100,
text=['{}%'.format(i) for i in
np.round(temp.values.astype(float)/temp.values.sum(),4)100],
textposition = 'auto',
name = 'Income_Bins')
data = [trace1]
# these are used to define the layout options
layout = generate_layout_bar('Income_Bins')
fig = go.Figure(data=data, layout=layout)
iplot(fig)
print(list(range(0,hr_data.MonthlyIncome.max()+10,2500)))
```

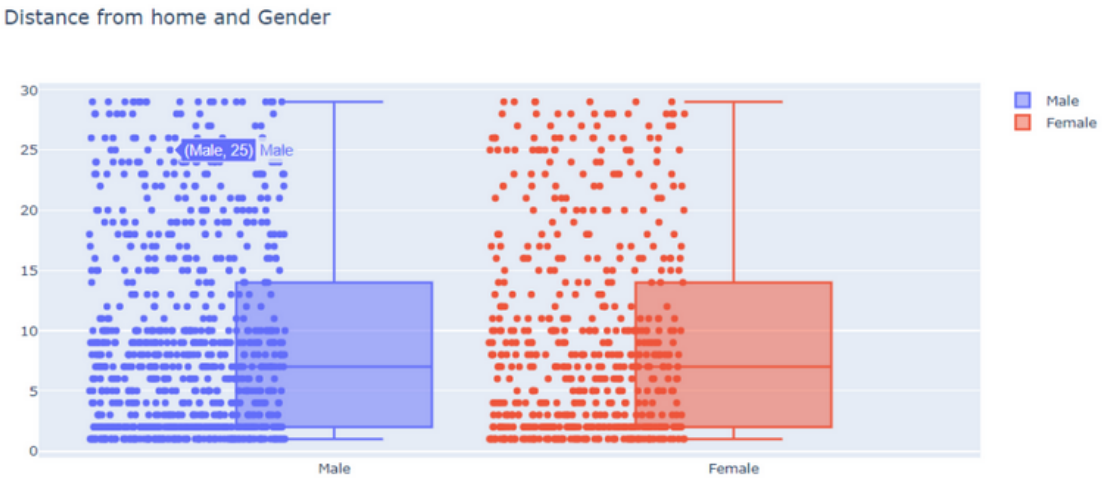



```
[0, 2500, 5000, 7500, 10000, 12500, 15000, 17500, 20000]
```

Bivariate Analysis

1. Is a particular gender travelling more distance than other?(Gender and Distance from home)

```
trace1 = go.Box(y = hr_data.DistanceFromHome[hr_data.Gender=='Male'],name = 'Male',
boxpoints = 'all',jitter = 1
)
# boxpoints is used to specify the points to plot
# jitter is used to specify how far from each should the points be
trace2 = go.Box(y = hr_data.DistanceFromHome[hr_data.Gender=='Female'],name= 'Female
boxpoints = 'all',jitter = 1
)
data = [trace1,trace2]
layout = go.Layout(width = 1000,
height = 500,title = 'Distance from home and Gender')
fig = go.Figure(data=data,layout = layout)
iplot(fig)
```



Distance Bins and Gender

```
hr_data['Distance_Bins']=(np.digitize(
hr_data.DistanceFromHome,[0,5,15,np.max(hr_data.DistanceFromHome)],right=True))

temp = hr_data.groupby(['Distance_Bins','Gender']).size().to_frame()
temp = temp.reset_index()
temp.columns = ['Distance_Bins','Gender','Count']
temp
```

	Distance_Bins	Gender	Count
0	1	Female	257
1	1	Male	375
2	2	Female	200
3	2	Male	309
4	3	Female	131
5	3	Male	198

```
trace1 = go.Bar(x = temp.Distance_Bins[temp.Gender=='Male'],
y = temp.Count[temp.Gender=='Male'],
text = temp.Count[temp.Gender=='Male'],
textposition = 'auto',
name = 'Male')
trace2 = go.Bar(x = temp.Distance_Bins[temp.Gender=='Female'],
y = temp.Count[temp.Gender=='Female'],
text = temp.Count[temp.Gender=='Female'],
textposition = 'auto',
name = 'Female')
data = [trace1,trace2]
layout = go.Layout(width = 700,
height = 500,title = 'Gender and Distance bins',
yaxis = dict(title='Count'))
fig = go.Figure(data=data, layout=layout)
iplot(fig)
```

Observations:

Irrespective of the distance bin, there is a global pattern i.e every bin has more male employees

2. Are employees working overtime getting better ratings? (Over Time and Performance Rating.)

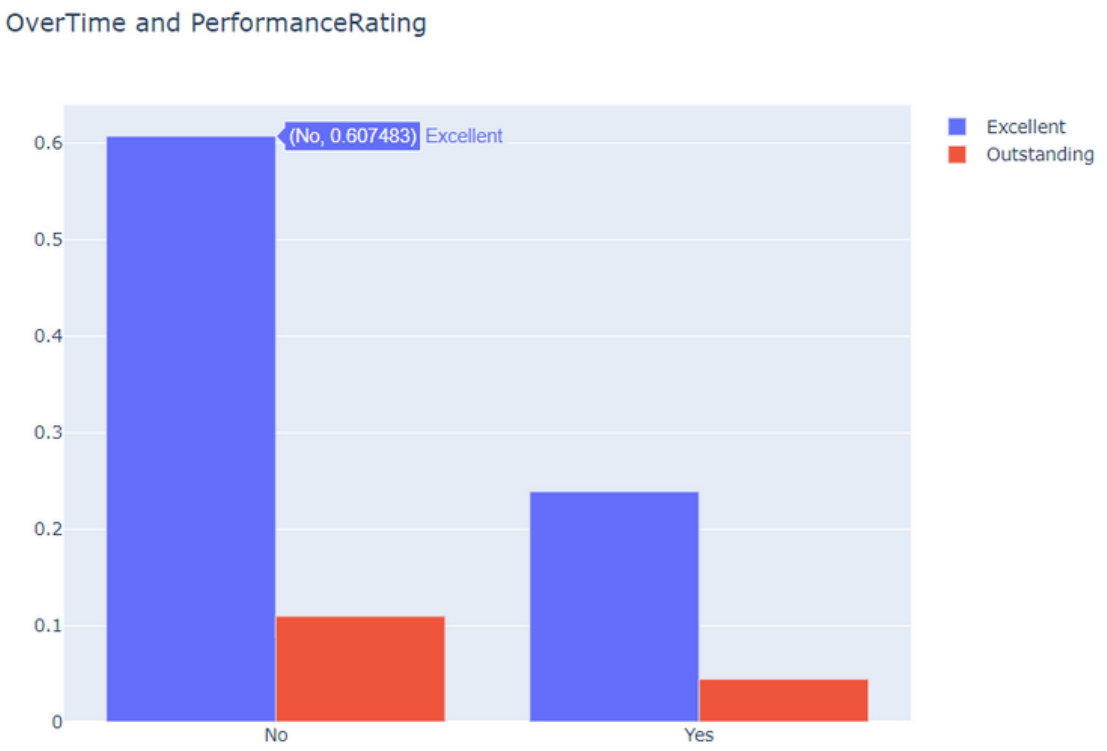
```
temp =
hr_data.groupby(['OverTime','PerformanceRating']).size().to_frame()
temp = temp.reset_index()
temp.columns = ['OverTime','PerformanceRating','Count']
temp
```

	OverTime	PerformanceRating	Count
0	No	Excellent	893
1	No	Outstanding	161
2	Yes	Excellent	351
3	Yes	Outstanding	65

```
hr_data.OverTime.value_counts()
```

```
No     1054
Yes     416
Name: OverTime, dtype: int64
```

```
trace1 = go.Bar(x = temp.OverTime[temp.PerformanceRating=='Excellent'],
y = temp.Count[temp.PerformanceRating=='Excellent']/temp.Count.sum(),
name = 'Excellent')
trace2 = go.Bar(x = temp.OverTime[temp.PerformanceRating=='Outstanding'],
y = temp.Count[temp.PerformanceRating=='Outstanding']/temp.Count.sum(),
name = 'Outstanding')
data = [trace1,trace2]
layout = go.Layout(width = 800,
height = 600,title = 'OverTime and PerformanceRating')
fig = go.Figure(data=data, layout=layout)
iplot(fig)
fig
```



All the percentages add up to one, so we can compare the numbers globally

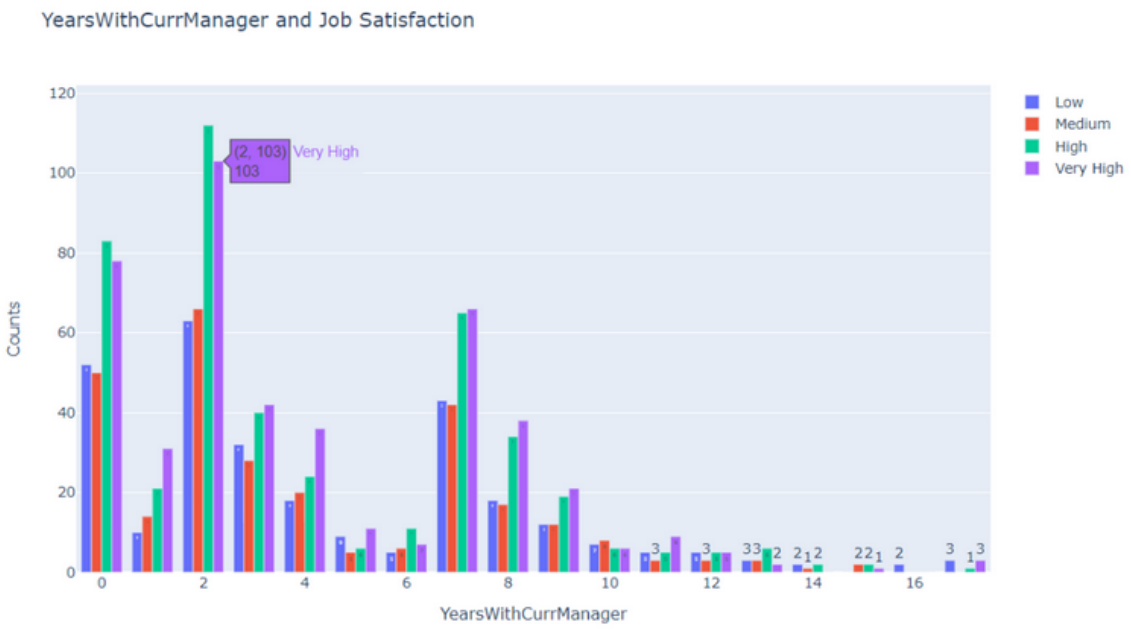
3. Does working longer with a manager have any relationship with Job satisfaction? (Years With Current Manager, Job Satisfaction)

```
yearscurrman_jobsat = hr_data.groupby(['YearsWithCurrManager','JobSatisfaction']).si
yearscurrman_jobsat = yearscurrman_jobsat.reset_index()
yearscurrman_jobsat.columns = ['YearsWithCurrManager','JobSatisfaction','Counts']

np.random.seed(0)
yearscurrman_jobsat.sample(frac =0.1)
```

Out[69]:

	YearsWithCurrManager	JobSatisfaction	Counts
26	6	Medium	6
27	6	Very High	7
48	12	High	5
22	5	Medium	5
30	7	Medium	42
51	12	Very High	5
7	1	Very High	31

```
tracelow = go.Bar(
x=yearscurrman_jobsat.YearsWithCurrManager[
yearscurrman_jobsat.JobSatisfaction=='Low'],
y = yearscurrman_jobsat.Counts[
yearscurrman_jobsat.JobSatisfaction=='Low'],
text = yearscurrman_jobsat.Counts[
yearscurrman_jobsat.JobSatisfaction=='Low'],
textposition = 'auto',
name = 'Low')
tracemedium = go.Bar(
x = yearscurrman_jobsat.YearsWithCurrManager[
yearscurrman_jobsat.JobSatisfaction=='Medium'],
y = yearscurrman_jobsat.Counts[
yearscurrman_jobsat.JobSatisfaction=='Medium'],
text = yearscurrman_jobsat.Counts[
yearscurrman_jobsat.JobSatisfaction=='Medium'],
textposition = 'auto',
name = 'Medium')
traceHigh = go.Bar(x = yearscurrman_jobsat.YearsWithCurrManager[
yearscurrman_jobsat.JobSatisfaction=='High'],
y = yearscurrman_jobsat.Counts[yearscurrman_jobsat.JobSatisfaction=='High'],
text = yearscurrman_jobsat.Counts[
yearscurrman_jobsat.JobSatisfaction=='High'],
textposition = 'auto',
name = 'High')
traceVHigh = go.Bar(x = yearscurrman_jobsat.YearsWithCurrManager[
yearscurrman_jobsat.JobSatisfaction=='Very High'],
y = yearscurrman_jobsat.Counts[yearscurrman_jobsat.JobSatisfaction=='Very High'],
text = yearscurrman_jobsat.Counts[yearscurrman_jobsat.JobSatisfaction=='Very High'],
textposition = 'auto',
name = 'Very High')
data = [tracelow, tracemedium, traceHigh, traceVHigh]
layout = go.Layout(width = 1000,
barmode='stack',height=600,title='YearsWithCurrManager and Job Satisfaction',
xaxis = dict(title='YearsWithCurrManager'),yaxis=dict(title='Counts',
range=[0,yearscurrman_jobsat.Counts.max()+10]))
fig = go.Figure(data=data, layout=layout)
iplot(fig)
```



We observe that the red bars are higher than the green bars only after 2 years , we can infer that employees generally tend to be comfortable working with the manager after 2 years.

4. Are married employees staying far from the office? (Marital status and Distance from home)

```
1 hr_data.MaritalStatus.unique()
```

```
Out[71]: ['Single', 'Married', 'Divorced']
         Categories (3, object): ['Single', 'Married', 'Divorced']
```

```
1 hr_data.DistanceFromHome[hr_data.MaritalStatus=='Divorced'].describe()
```

```
Out[72]: count    327.000000
         mean       9.110092
         std        7.728133
         min        1.000000
         25%        2.000000
         50%        7.000000
         75%       13.000000
         max       29.000000
         Name: DistanceFromHome, dtype: float64
```

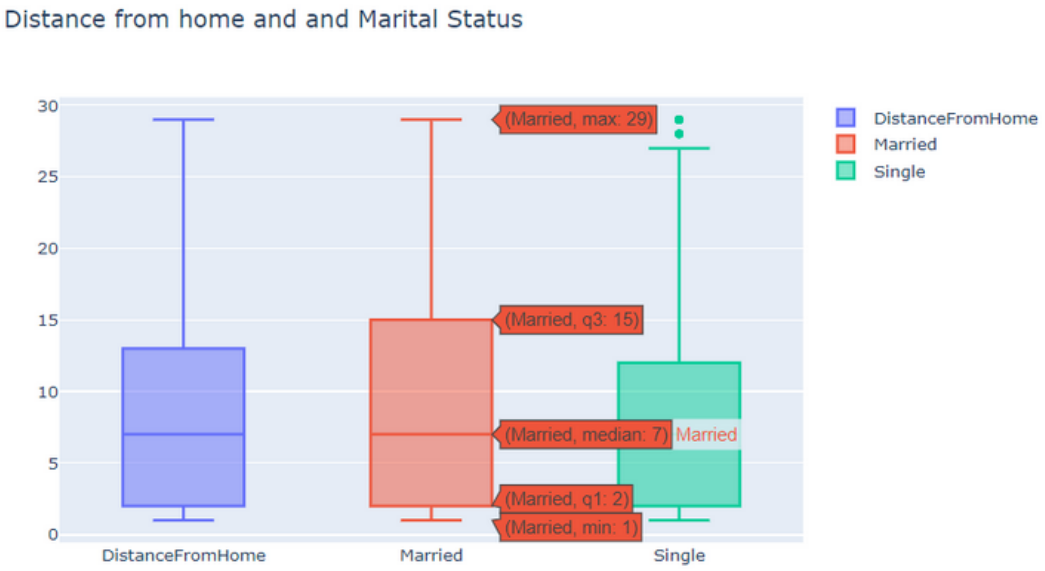
```
1 hr_data.DistanceFromHome[hr_data.MaritalStatus=='Married'].describe()
```

Out[73]:
```
count    673.000000
mean       9.459138
std        8.348717
min        1.000000
25%        2.000000
50%        7.000000
75%       15.000000
max       29.000000
Name: DistanceFromHome, dtype: float64
```

```
1 hr_data.DistanceFromHome[hr_data.MaritalStatus=='Single'].describe()
```

Out[74]:
```
count    470.000000
mean       8.868085
std        8.015952
min        1.000000
25%        2.000000
50%        7.000000
75%       12.000000
max       29.000000
Name: DistanceFromHome, dtype: float64
```

```
tracediv = go.Box(y = hr_data.DistanceFromHome[hr_data.MaritalStatus=='Divorced'],
name = 'DistanceFromHome')
tracemarried = go.Box(y = hr_data.DistanceFromHome[hr_data.MaritalStatus=='Married'],
name= 'Married')
tracesin = go.Box(y = hr_data.DistanceFromHome[hr_data.MaritalStatus=='Single'],
name= 'Single')
data = [tracediv,tracemarried,tracesin]
layout = go.Layout(width = 800,
height = 500,title = 'Distance from home and and Marital Status')
fig = go.Figure(data=data,layout = layout)
iplot(fig)
```



5. Is there any relationship between Attrition and Gender?

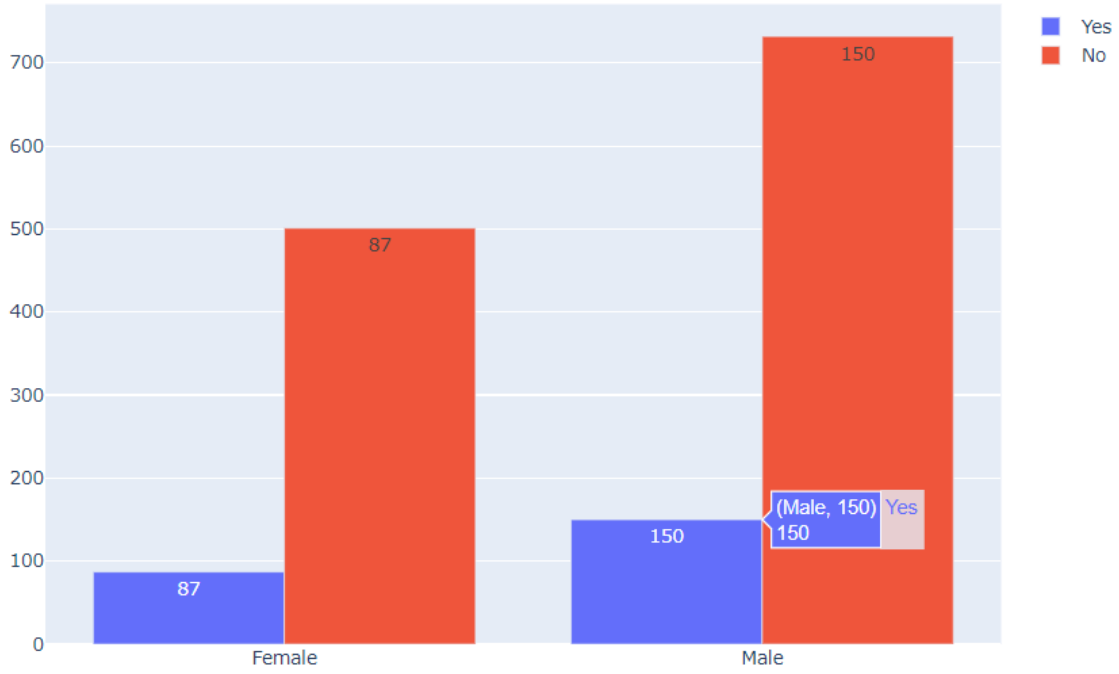
```
Gender_Attrition = hr_data.groupby(['Gender','Attrition']).size().to_frame()
Gender_Attrition = Gender_Attrition.reset_index()
Gender_Attrition.columns = ['Gender','Attrition','Count']
Gender_Attrition
```

Out[76]:

	Gender	Attrition	Count
0	Female	No	501
1	Female	Yes	87
2	Male	No	732
3	Male	Yes	150

```
trace1 = go.Bar(x = Gender_Attrition.Gender[Gender_Attrition.Attrition=='Yes'],
y = Gender_Attrition.Count[Gender_Attrition.Attrition=='Yes'],
text = Gender_Attrition.Count[Gender_Attrition.Attrition=='Yes'],
textposition = 'auto',
name = 'Yes')
trace2 = go.Bar(x = Gender_Attrition.Gender[Gender_Attrition.Attrition=='No'],
y = Gender_Attrition.Count[Gender_Attrition.Attrition=='No'],
text = Gender_Attrition.Count[Gender_Attrition.Attrition=='Yes'],
textposition = 'auto',
name = 'No')
data = [trace1,trace2]
layout = go.Layout(width = 800,
height = 600,title = 'Gender and Attrition')
fig = go.Figure(data=data, layout=layout)
iplot(fig)
```

Gender and Attrition



6. Employees who spend more years in the company tend to leave. Verify if this is true.(Years at company and Attrition)

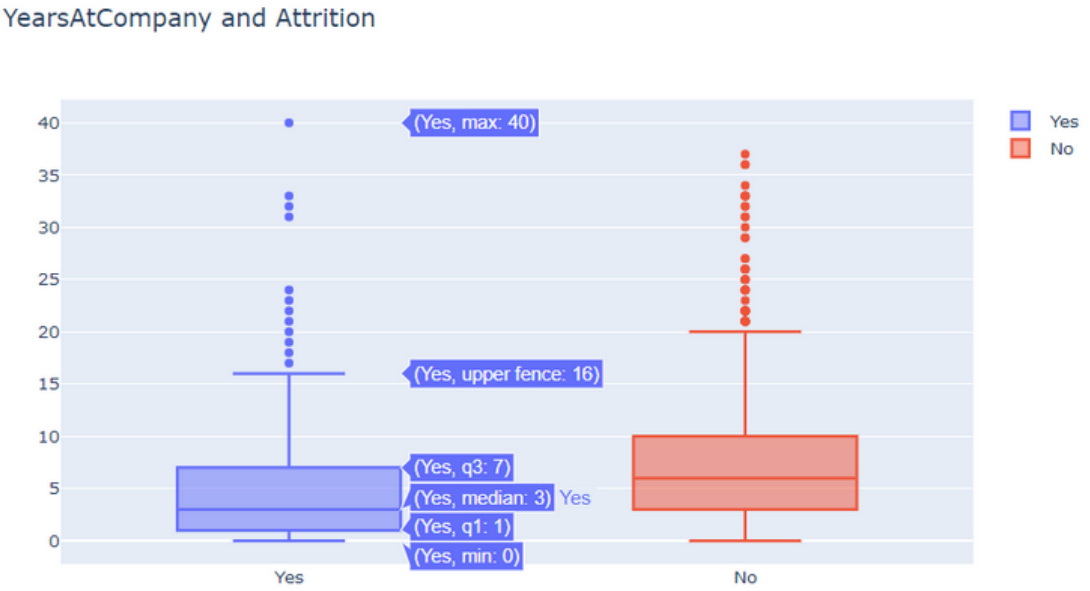
```
1 hr_data.YearsAtCompany[hr_data.Attrition=='Yes'].describe()
```

```
Out[79]: count    1233.000000
         mean        7.369019
         std         6.096298
         min         0.000000
         25%         3.000000
         50%         6.000000
         75%        10.000000
         max        37.000000
         Name: YearsAtCompany, dtype: float64
```

```
1 hr_data.YearsAtCompany[hr_data.Attrition=='No'].describe()
```

```
Out[79]: count    1233.000000
         mean        7.369019
         std         6.096298
         min         0.000000
         25%         3.000000
         50%         6.000000
         75%        10.000000
         max        37.000000
         Name: YearsAtCompany, dtype: float64
```

```
trace1 = go.Box(y = hr_data.YearsAtCompany[hr_data.Attrition=='Yes'],name = 'Yes',
boxpoints = 'all',jitter = 1
<code>)</code>
# boxpoints is used to specify the points to plot
# jitter is used to specify how far from each should the points be
trace2 = go.Box(y = hr_data.YearsAtCompany[hr_data.Attrition=='No'],name= 'No',
boxpoints = 'all',jitter = 1
<code>)</code>
data = [trace1,trace2]
layout = go.Layout(width = 800,
height = 500,title = 'YearsAtCompany and Attrition')
fig = go.Figure(data=data,layout = layout)
iplot(fig)
```



7 . Is a particular age group more prone to leaving the company? (Age and Attrition)

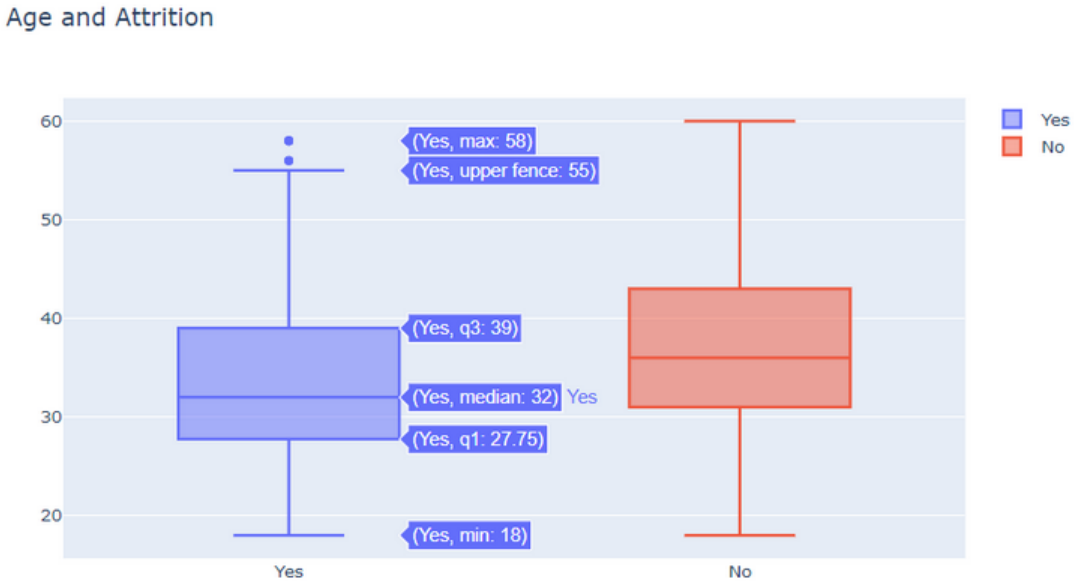
```
1 hr_data.Age[hr_data.Attrition=='Yes'].describe()
```

Out[81]:
```
count    237.000000
mean      33.607595
std        9.689350
min       18.000000
25%       28.000000
50%       32.000000
75%       39.000000
max       58.000000
Name: Age, dtype: float64
```

```
1 hr_data.Age[hr_data.Attrition=='No'].describe()
```

Out[82]:
```
count    1233.000000
mean       37.561233
std         8.888360
min        18.000000
25%        31.000000
50%        36.000000
75%        43.000000
max        60.000000
Name: Age, dtype: float64
```

```
trace1 = go.Box(y = hr_data.Age[hr_data.Attrition=='Yes'],name = 'Yes')
trace2 = go.Box(y = hr_data.Age[hr_data.Attrition=='No'],name= 'No')
data = [trace1,trace2]
layout = go.Layout(width = 800,
height = 500,title = 'Age and Attrition')
fig = go.Figure(data=data,layout = layout)
iplot(fig)
```



You can also bin the age column and do the same.

8. Employees earning less tend to leave the company. Verify if this is true. (Monthly Income vs Attrition)

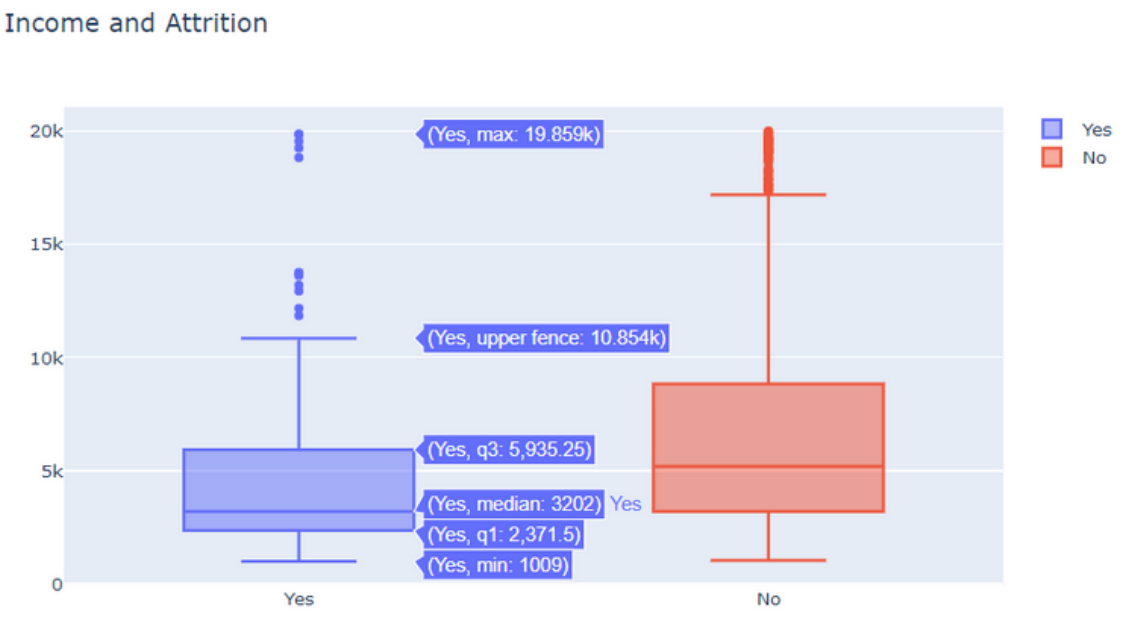
In [84]:
```
1 hr_data.MonthlyIncome[hr_data.Attrition=='Yes'].describe()
```

Out[84]:
```
count      237.000000
mean      4787.092827
std       3640.210367
min       1009.000000
25%       2373.000000
50%       3202.000000
75%       5916.000000
max      19859.000000
Name: MonthlyIncome, dtype: float64
```

In [85]:
```
1 hr_data.MonthlyIncome[hr_data.Attrition=='No'].describe()
```

Out[85]:
```
count     1233.000000
mean      6832.739659
std       4818.208001
min       1051.000000
25%       3211.000000
50%       5204.000000
75%       8834.000000
max      19999.000000
Name: MonthlyIncome, dtype: float64
```

```
1 trace1 = go.Box(y = hr_data.MonthlyIncome[hr_data.Attrition=='Yes'],name = 'Yes')
2 trace2 = go.Box(y = hr_data.MonthlyIncome[hr_data.Attrition=='No'],name= 'No')
3 data = [trace1,trace2]
4 layout = go.Layout(width = 800,
5 height = 500,title = 'Income and Attrition')
6 fig = go.Figure(data=data,layout = layout)
7 iplot(fig)
```



In [87]:
```
1 num_cols
```

Out[87]:
```
['Age',
 'DistanceFromHome',
 'EmployeeNumber',
 'MonthlyIncome',
 'TotalWorkingYears',
 'YearsAtCompany',
 'YearsInCurrentRole',
 'YearsSinceLastPromotion',
 'YearsWithCurrManager']
```

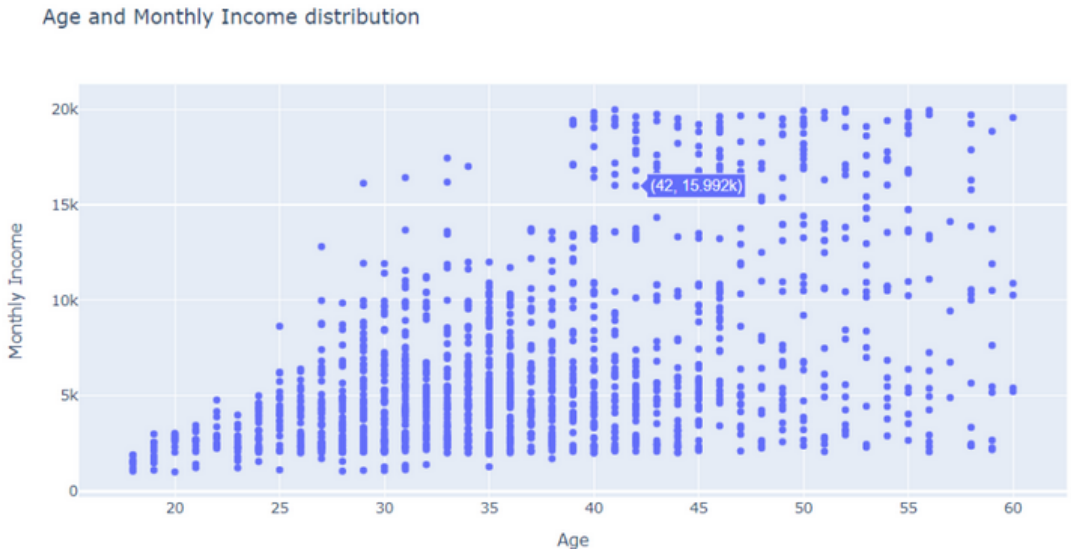
9. How do Age and Monthly Income vary?

In [88]:
```
1 hr_data[['Age','MonthlyIncome']].corr()
```

Out[88]:

	Age	MonthlyIncome
Age	1.000000	0.497855
MonthlyIncome	0.497855	1.000000

```
trace = go.Scatter(x=hr_data.Age ,
y= hr_data.MonthlyIncome,
name = 'Age and MonthlyIncome',
mode= 'markers')
data = [trace]
layout = go.Layout(title = ' Age and Monthly Income distribution',
xaxis = dict(title = 'Age'),
yaxis = dict(title = 'Monthly Income'))
fig = go.Figure(data=data,layout=layout)
iplot(fig)
```



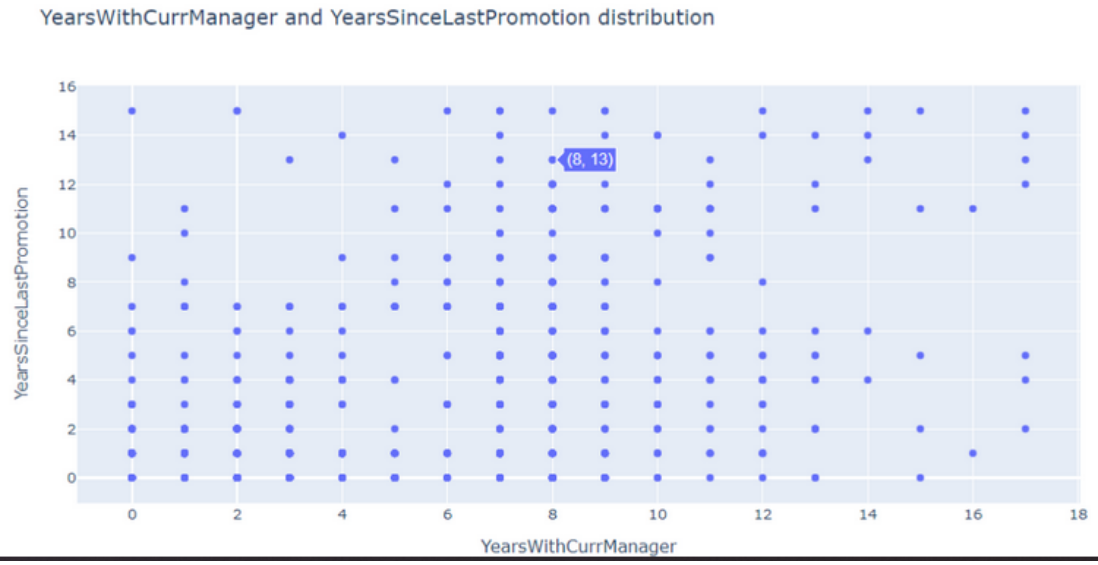
10. Does Years With Curr Manager have to do anything with Years Since Last Promotion?

In [90]:
```
1 hr_data[['YearsWithCurrManager','YearsSinceLastPromotion']].corr()
```

Out[90]:

	YearsWithCurrManager	YearsSinceLastPromotion
YearsWithCurrManager	1.000000	0.510224
YearsSinceLastPromotion	0.510224	1.000000

```
trace = go.Scatter(x=hr_data.YearsWithCurrManager ,
y= hr_data.YearsSinceLastPromotion,
name = 'YearsWithCurrManager and YearsSinceLastPromotion',
mode= 'markers')
data = [trace]
layout = go.Layout(title = ' YearsWithCurrManager and YearsSinceLastPromotion distr
xaxis = dict(title = 'YearsWithCurrManager'),
yaxis = dict(title = 'YearsSinceLastPromotion'))
fig = go.Figure(data=data,layout=layout)
iplot(fig)
```



3 variables

1. What is the relationship between number of companies worked , age and attrition.(Number of companies worked, Age, Attrition.)

In [92]:
```
hr_data.NumCompaniesWorked.value_counts()
```

Out[92]:
```
1    521
0    197
3    159
2    146
4    139
7     74
6     70
5     63
9     52
8     49
Name: NumCompaniesWorked, dtype: int64
```

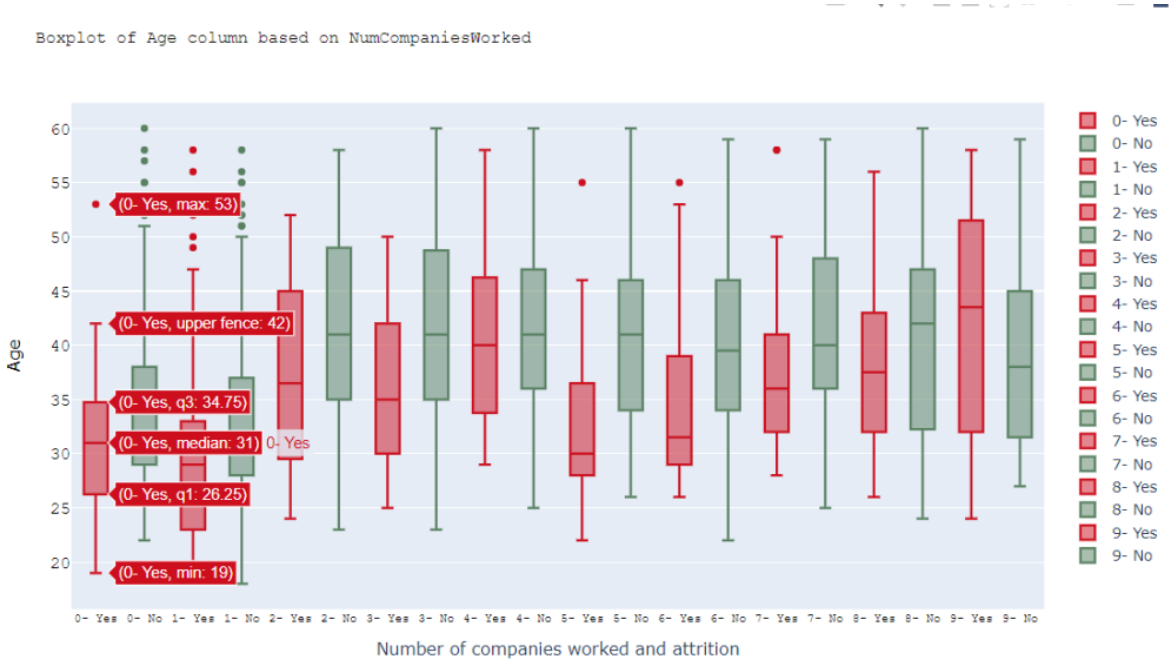
Is a particular gender of a particular age group prone to leave the company?

In [93]:
```
data = []
for i in hr_data.NumCompaniesWorked.unique():
    data.append(i)
```

In [94]:
```
data
```

Out[94]:
```
[8, 1, 6, 9, 0, 4, 5, 2, 7, 3]
```

```
data = []
for i in np.sort(hr_data.NumCompaniesWorked.unique()):
data.append(go.Box(y = hr_data.Age[
hr_data.NumCompaniesWorked==i][hr_data.Attrition=='Yes'],
marker = dict(
color = '#CC0E1D',
),
name = "{}- Yes".format(str(i))))
data.append(go.Box(y = hr_data.Age[
hr_data.NumCompaniesWorked==i][hr_data.Attrition=='No'],
marker = dict(
color = '#588061',
),
name = "{}- No".format(str(i))))
layout = go.Layout(
autosize=False, # auto size the graph? use False if you are specifying the height a
width=1000, # height of the figure in pixels
height=600, # height of the figure in pixels
title = "Boxplot of {} column based on {} ".format('Age','NumCompaniesWorked'), # t
# more granular control on the title font
<code>titlefont=dict( family='Courier New, monospace', # font family </code>
<code>size=14, # size of the font </code>
<code>color='black' # color of the font ), </code>
<code># granular control on the axes objects</code>
xaxis=dict(title='Number of Companies worked and attrition',
tickfont=dict(family = 'Courier New, monospace',
size=10,
color='black',
),
yaxis=dict(
# range=[0,100],title='Age',
titlefont=dict(size=14,color='black'),
tickfont=dict(family='Courier New, monospace', size=14,color='black')),)
fig = go.Figure(data=data, layout=layout)
iplot(fig)
```



Observe the last two plots, how are they different from others?

Creating new features and plotting

2. What is the relationship between total working years, number of companies worked and attrition (Total Working Years , Number of companies and Attrition.)

Generate a new feature using Total Working Years and Number of companies worked

```
hr_data['TotalWorkingYears_NumCompWorked'] = np.round(
hr_data.TotalWorkingYears / (hr_data.NumCompaniesWorked.astype(int)+1))
# adding 1 to avoid dividng by 0

hr_data.TotalWorkingYears_NumCompWorked.head()
```

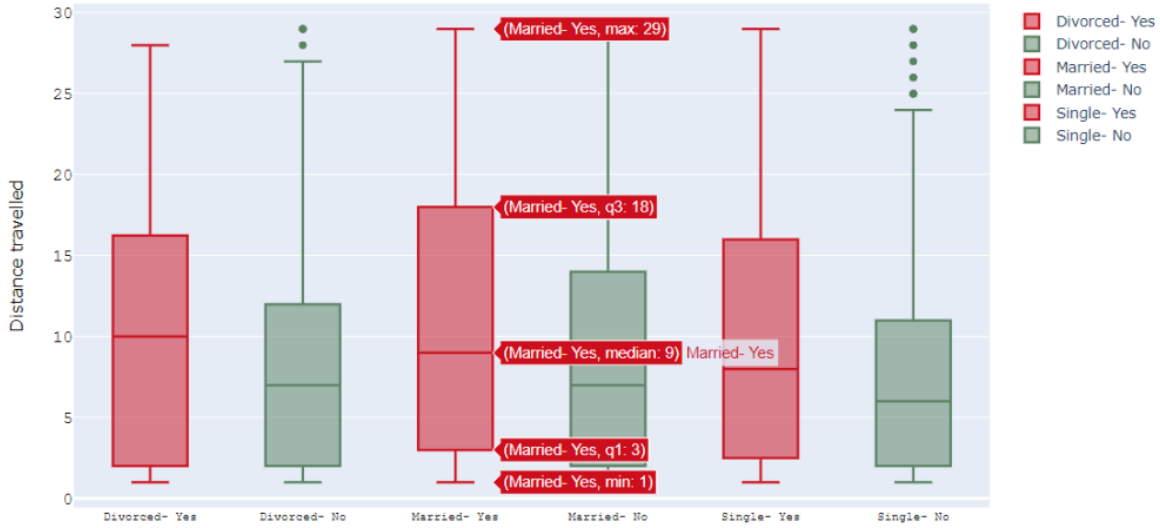
```
Out[97]: 0    1.0
         1    5.0
         2    1.0
         3    4.0
         4    1.0
         Name: TotalWorkingYears_NumCompWorked, dtype: float64
```

```
trace0 = go.Box(y= hr_data.TotalWorkingYears_NumCompWorked[hr_data.Attrition=='Yes'],
name = 'Yes')
trace1 = go.Box(y = hr_data.TotalWorkingYears_NumCompWorked[hr_data.Attrition=='No'],
name = 'No')
data =[trace0,trace1]
layout = go.Layout(width = 900,
height = 600,
title = 'Ratio of Age and Number of Companies worked vs Attrition',
titlefont=dict(
family='Courier New, monospace', # font family
size=14, # size of the font
color='black' # color of the font
),
# granular control on the axes objects
xaxis=dict(
tickfont=dict(
family='Courier New, monospace',
size=10,
color='black'
)
),
yaxis=dict(
# range=[0,100],
<code>title='(TotalWorkingYears/NumCompWorked)', </code>
<code>titlefont=dict( size=14, color='black' ), </code>
<code>tickfont=dict( family='Courier New, monospace', </code>
<code>size=14,</code>
<code>color='black' # color of the font )</code>
),
)
fig = go.Figure(data=data, layout=layout)
iplot(fig)
```

3. Do marital status and distance from home affect attrition? (Marital Status, Distance From Home and Attrition)

```
data = []
for i in np.sort(hr_data.MaritalStatus.unique()):
data.append(go.Box(y = hr_data.DistanceFromHome[
hr_data.MaritalStatus==i][hr_data.Attrition=='Yes'],
marker = dict(color = '#CC0E1D', # red),
),
name = "{}- Yes".format(str(i)))
)
<code>data.append(go.Box(y = hr_data.DistanceFromHome[</code>
<code>hr_data.MaritalStatus==i][hr_data.Attrition=='No'], </code>
<code>marker = dict(color = '#588061', # green), ), </code>
<code>name = "{}- No".format(str(i))) )</code>
layout = go.Layout(
autosize=False,
width=1000, # height of the figure in pixels
height=600, # height of the figure in pixels
title = "Boxplot of {} column based on {} ".format('DistanceFromHome','MaritalStatus'),
titlefont=dict(
family='Courier New, monospace', # font family
size=14, # size of the font
color='black' # color of the font
),
# granular control on the axes objects
xaxis=dict(
tickfont=dict(
family='Courier New, monospace', # font family
size=10, # size of ticks displayed on the x axis
color='black' # color of the font
)
),
yaxis=dict(
# range=[0,100],
<code>title='Distance travelled', titlefont=dict( size=14, color='black' ), </code>
<code>tickfont=dict( family='Courier New, monospace', </code>
<code>size=14,  color='black' # color of the font ) ), )</code>
fig = go.Figure(data=data, layout=layout)
iplot(fig)
```

Boxplot of DistanceFromHome column based on MaritalStatus



Extras

>3 variables, 3D plots.

```
n = 1500
Extracting th x, y ,z values
temp = hr_data.iloc[0:n,]
temp.shape
```

Out[115]: (1470, 35)

x-axis > PercentSalaryHike

x-axis > YearsAtCompany

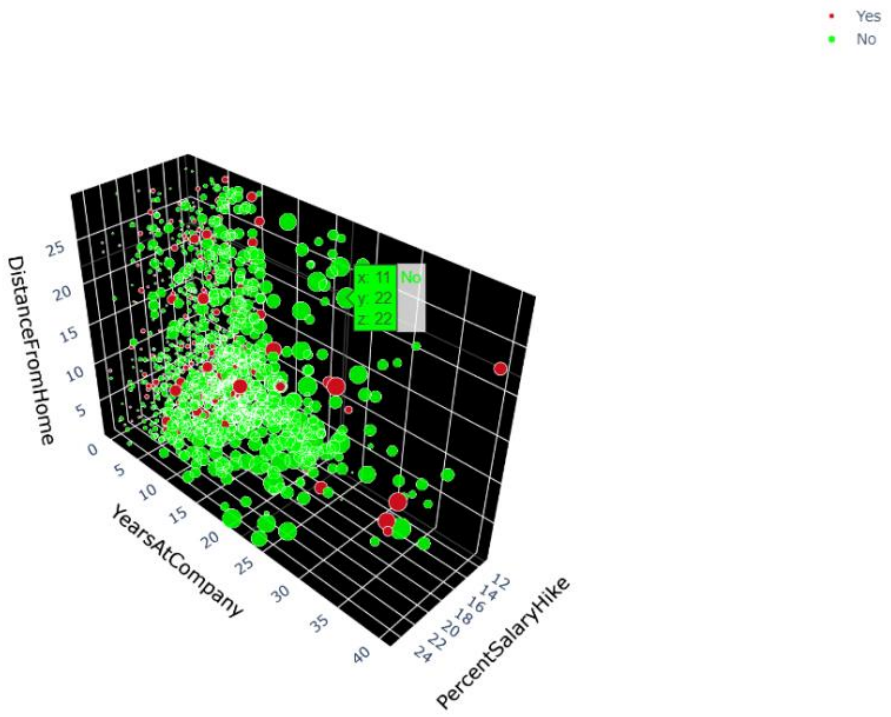
x-axis > DistanceFromHome

Color > Attrition

Size > DistanceFromHome

```
trace1 = go.Scatter3d(
x=temp.PercentSalaryHike[temp.Attrition=='Yes'],
y=temp.YearsAtCompany[temp.Attrition=='Yes'],
z=temp.DistanceFromHome[temp.Attrition=='Yes'],
mode='markers',name ='Yes',
marker=dict(
size=temp.YearsInCurrentRole[temp.Attrition=='Yes']+2,
color='#CC0E1D', # ferarri red
# colorscale='Viridis', # choose a colorscale
opacity=1
)
)
trace2 = go.Scatter3d(
x=temp.PercentSalaryHike[temp.Attrition=='No'],
y=temp.YearsAtCompany[temp.Attrition=='No'],
z=temp.DistanceFromHome[temp.Attrition=='No'],
mode='markers',name ='No',
marker=dict(
size=temp.YearsInCurrentRole[temp.Attrition=='No']+2,
color='rgb(0,255,0)', #green
# colorscale='Viridis', # choose a colorscale
opacity=0.9,
)
)
data = [trace1,trace2]
layout = go.Layout(
scene = dict(
xaxis = dict(
title='PercentSalaryHike',
backgroundcolor="black",
showbackground=True,
titlefont=dict(
size=16,
color='black'
)
),
yaxis = dict(
title='YearsAtCompany',
showbackground=True,
backgroundcolor="black",
titlefont=dict(
size=16,
color='black'
)
),
zaxis = dict(
title='DistanceFromHome',
backgroundcolor="black",
showbackground=True,
titlefont=dict(
size=16,
color='black'
)
)
),
width=1000, # height of the figure in pixels
height=800, # height of the figure in pixels
)
fig = go.Figure(data=data, layout=layout)
fig['layout'].update(title= "PercentSalaryHike,
YearsAtCompany, DistanceFromHome, YearsInCurrentRole and Attrition")
iplot(fig, filename='3d-scatter-colorscale')
```

PercentSalaryHike, YearsAtCompany, DistanceFromHome, YearsInCurrentRole and Attrition



Clustering

In [102]:
```
from sklearn.preprocessing import scale
# import KMeans
from sklearn.cluster import KMeans
```

In [103]:
```
data_clustering = hr_data[['Age','MonthlyIncome','DistanceFromHome']]
```

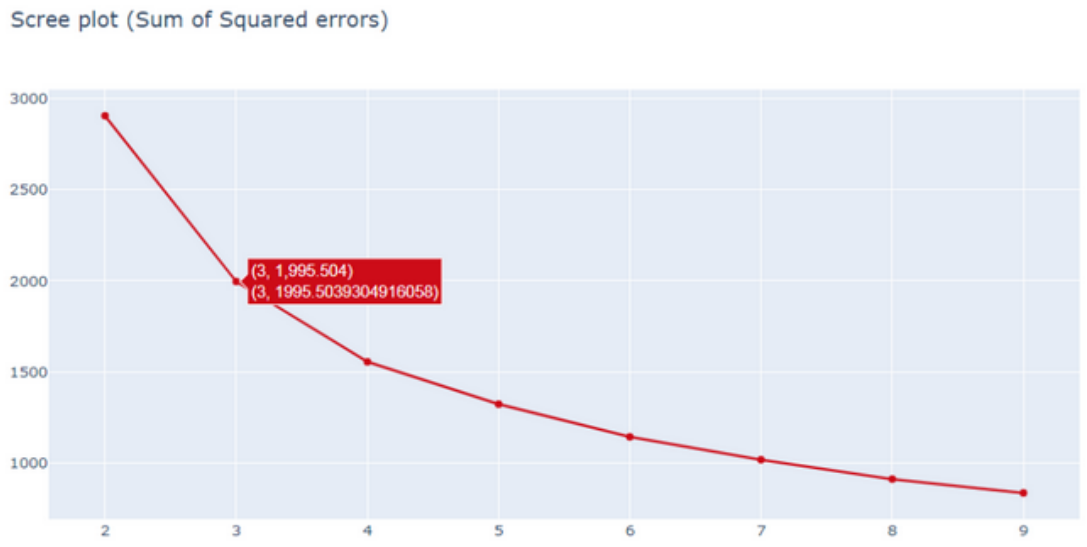
Standardizing the data

In [104]:
```
data_clustering = scale(data_clustering)
```

In [105]:
```
sse = []
for i in range(2,10):
    # create kmeans object
    kmeans = kmeans = KMeans(init='k-means++', n_clusters=i, n_init=10)
    # fit kmeans object to data
    kmeans.fit(data_clustering)
    # print location of clusters learned by kmeans object
#     print(kmeans.cluster_centers_)
    # save new clusters for chart
    sse.append(kmeans.inertia_)
```

Scree Plot

```
x=list(range(2,10))
y=sse
data = [go.Scatter(x=x, # number of clusters
y=y, # sum of squared errors
text = [str(i) for i in (zip(x,y))], # text to display on hover
textposition = 'top center',
line = dict(color = ('rgb(205, 12, 24)')) # line color
)]
layout = go.Layout(title ='Scree plot (Sum of Squared errors)')
fig = go.Figure(data=data,layout=layout)
iplot(fig)
```



From the above scree plot, 7 or 8 clusters seem to be a good start

for plotting and simplicity I will take 5 clusters

In [107]:
```
kmeans = KMeans(n_clusters=5)
# fit kmeans object to data
kmeans.fit(data_clustering)
# print location of clusters learned by kmeans object
print(kmeans.cluster_centers_)
# save new clusters for chart
y_km = kmeans.predict(data_clustering) # read what fit_predict does
```

```
[[-0.79175583 -0.48014936 -0.51464506]
 [ 1.13924347  2.07625635 -0.49711528]
 [-0.48317099 -0.42882527  1.47006256]
 [ 1.19305984  0.88284983  1.73789408]
 [ 0.65109605 -0.189357   -0.46848887]]
```

In [108]:
```
np.unique(y_km)
```

Out[108]:
```
array([0, 1, 2, 3, 4])
```

Adding the cluster centers to the data

In [109]:
```
hr_data['Cluster_Centers'] = y_km
```

In [110]:
```
colors_clusters = hr_data.Cluster_Centers
```

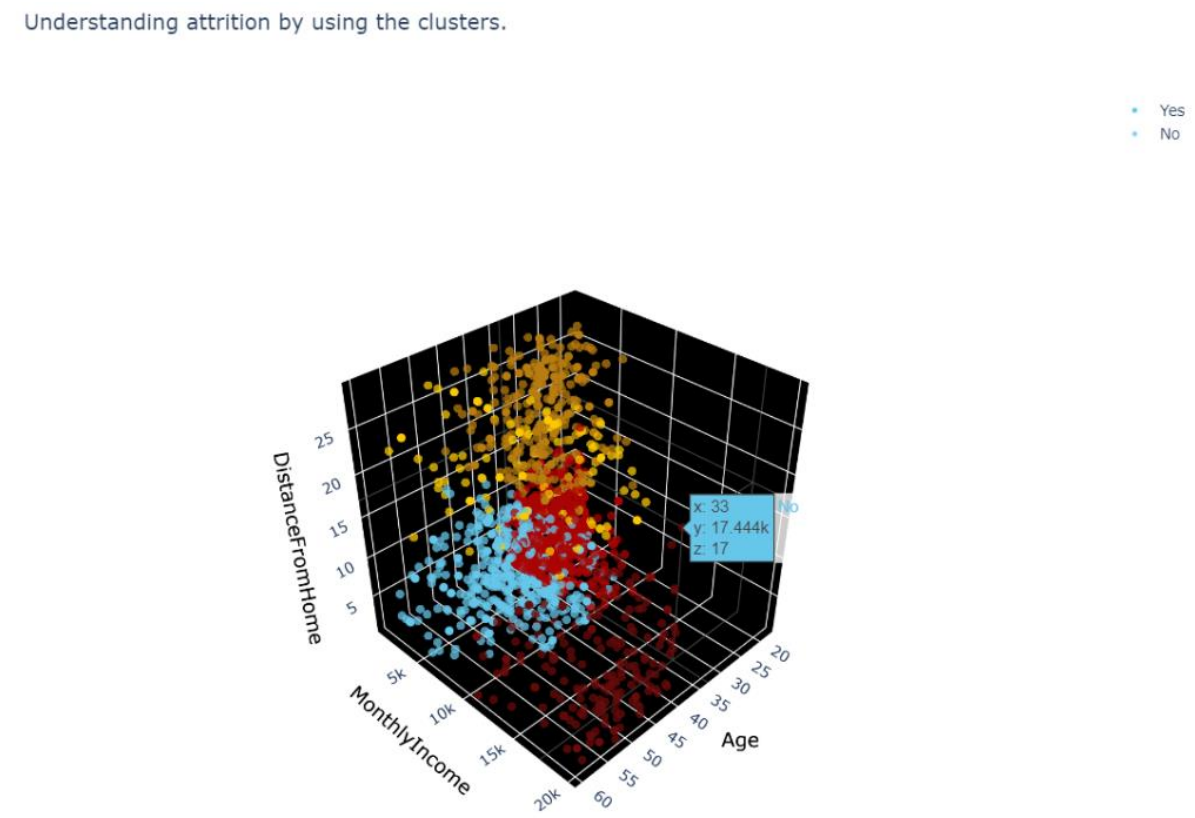
In [111]:
```
colors_clusters = colors_clusters.replace(to_replace=[0,1,2,3,4],value = ['rgb(170, 5, 5)',
                                  'rgb(106, 12, 11)',
                                  'rgb(185, 125, 16)',
                                  'rgb (251, 202, 3)',
                                  'rgb(103, 199, 235)'])
```

Adding a column with the cluster colors

In [112]:
```
hr_data.colors_clusters = colors_clusters
```

In [113]:
```
hr_data.to_csv("data_with_clusters.csv",index=False)
```

```
trace0 = go.Scatter3d(
x=hr_data.Age[hr_data.Attrition=='Yes'],
y=hr_data.MonthlyIncome[hr_data.Attrition=='Yes'],
z=hr_data.DistanceFromHome[hr_data.Attrition=='Yes'],
mode='markers',name ='Yes',
marker=dict(
size=4,
color=hr_data.colors_clusters[hr_data.Attrition=='Yes'],
# colorscale='Viridis', # choose a colorscale
opacity=1
)
)
trace1 = go.Scatter3d(
x=hr_data.Age[hr_data.Attrition=='No'],
y=hr_data.MonthlyIncome[hr_data.Attrition=='No'],
z=hr_data.DistanceFromHome[hr_data.Attrition=='No'],
mode='markers',name ='No',
marker=dict(
size=4,
color=hr_data.colors_clusters[hr_data.Attrition=='No'],
# colorscale='Viridis', # choose a colorscale
opacity=0.75
)
)
data = [trace0,trace1]
layout = go.Layout(
scene = dict(
xaxis = dict(
title='Age',
backgroundcolor="black",
showbackground=True,
titlefont=dict(
size=16,
color='black'
)
<code>), </code>
<code>yaxis = dict( title='MonthlyIncome', showbackground=True, backgroundcolor="black",
<code>titlefont=dict( size=16, color='black' ) ), zaxis = dict( title='DistanceFromHome',
<code>backgroundcolor="black", showbackground=True, </code>
<code>titlefont=dict( size=16, color='black' ) ) ), </code>
<code>width=1000, # height of the figure in pixels </code>
<code>height=800, </code>
<code>margin = dict( b =15),)</code>
fig = go.Figure(data=data, layout=layout)
fig['layout'].update(
title= "Understanding attrition by using the clusters.")
iplot(fig)
```



One of the metric to find out if you have chosen the correct number of clusters is to see if you can give a name to all your clusters in terms of business.

This is all for now. I have also created a report on Employee Attrition Rate Analysis. you may like to check it as well. Please read it using the below link.

[Report on Employee Attrition Rate Analysis]

Thank you for reading. Your comments, thoughts on this post are most welcome.