The project 3: "HR Analytics"

In [1]:
```
# Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
%matplotlib inline
```

In [2]:
```
# Load data from the files diabetes.csv
df = pd.DataFrame()
df = pd.read_csv('HR-Employee-Attrition.csv')
```

In [3]:
```
df
```

Out[3]:

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

1470 rows × 35 columns

In [4]:
```
df.columns
```

Out[4]:
```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

In [5]:
```
df.sample(25)
```

Out[5]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Ed
228	35	No	Travel_Frequently	944	Sales	1	3	
765	38	No	Travel_Frequently	1186	Research & Development	3	4	
1383	36	No	Non-Travel	1351	Research & Development	9	4	
442	36	No	Non-Travel	635	Sales	10	4	
738	39	No	Travel_Rarely	466	Research & Development	1	1	
101	32	No	Travel_Rarely	827	Research & Development	1	1	
27	42	No	Travel_Rarely	691	Sales	8	4	
901	48	No	Travel_Rarely	969	Research & Development	2	2	
867	50	No	Travel_Frequently	1421	Research & Development	2	3	
1278	36	No	Travel_Rarely	1383	Research & Development	10	3	
1176	49	No	Travel_Rarely	301	Research & Development	22	4	
851	56	No	Travel_Rarely	718	Research & Development	4	4	
1262	43	Yes	Travel_Frequently	807	Research & Development	17	3	
619	33	No	Travel_Rarely	586	Sales	1	3	
0	41	Yes	Travel_Rarely	1102	Sales	1	2	
631	44	No	Travel_Rarely	986	Research & Development	8	4	
172	36	No	Travel_Frequently	1480	Research & Development	3	2	
777	21	Yes	Travel_Rarely	1334	Research & Development	10	3	
1311	18	No	Non-Travel	1431	Research & Development	14	3	
542	38	No	Travel_Rarely	168	Research & Development	1	3	
1333	46	Yes	Travel_Rarely	1254	Sales	10	3	
332	54	No	Travel_Frequently	928	Research & Development	20	4	
664	36	No	Travel_Rarely	1425	Research & Development	14	1	

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Ed
1388	32	No	Non-Travel	1146	Research & Development	15	4	
659	28	No	Travel_Rarely	821	Sales	5	4	

25 rows × 35 columns

In [6]:
```
# List the types of the columns
df.dtypes
```

Out[6]:
```
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
```

In [7]:
```
# check whether somewhere has no value in the data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          1470 non-null   int64
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

In [8]:
```
#Check Null values
df.isna().sum()
```

Out[8]:
```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

In [9]:
```
#Check Duplicate values
df.duplicated().sum()
```

Out[9]:
```
0
```

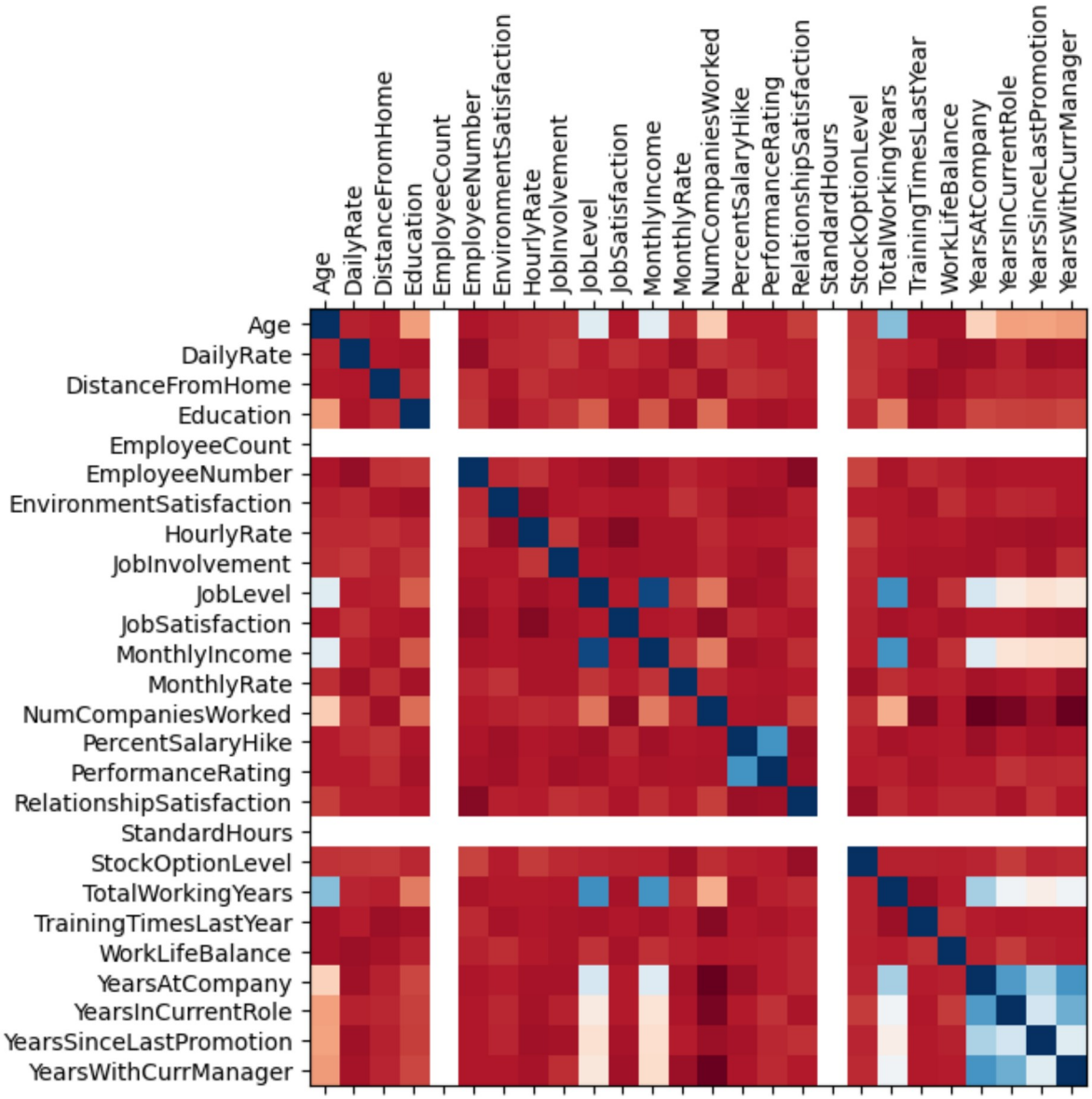
In [10]:
```
columns = ['Age', 'DailyRate',
       'DistanceFromHome', 'Education','EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobSatisfaction', 'MonthlyIncome', 'MonthlyRa
       'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager']
corre = df[columns].corr()
corre
```

Out[10]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	Em
Age	1.000000	0.010661	-0.001686	0.208034	NaN	
DailyRate	0.010661	1.000000	-0.004985	-0.016806	NaN	
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	NaN	
Education	0.208034	-0.016806	0.021042	1.000000	NaN	
EmployeeCount	NaN	NaN	NaN	NaN	NaN	
EmployeeNumber	-0.010145	-0.050990	0.032916	0.042070	NaN	
EnvironmentSatisfaction	0.010146	0.018355	-0.016075	-0.027128	NaN	
HourlyRate	0.024287	0.023381	0.031131	0.016775	NaN	
JobInvolvement	0.029820	0.046135	0.008783	0.042438	NaN	
JobLevel	0.509604	0.002966	0.005303	0.101589	NaN	
JobSatisfaction	-0.004892	0.030571	-0.003669	-0.011296	NaN	
MonthlyIncome	0.497855	0.007707	-0.017014	0.094961	NaN	
MonthlyRate	0.028051	-0.032182	0.027473	-0.026084	NaN	
NumCompaniesWorked	0.299635	0.038153	-0.029251	0.126317	NaN	
PercentSalaryHike	0.003634	0.022704	0.040235	-0.011111	NaN	
PerformanceRating	0.001904	0.000473	0.027110	-0.024539	NaN	
RelationshipSatisfaction	0.053535	0.007846	0.006557	-0.009118	NaN	
StandardHours	NaN	NaN	NaN	NaN	NaN	
StockOptionLevel	0.037510	0.042143	0.044872	0.018422	NaN	
TotalWorkingYears	0.680381	0.014515	0.004628	0.148280	NaN	
TrainingTimesLastYear	-0.019621	0.002453	-0.036942	-0.025100	NaN	
WorkLifeBalance	-0.021490	-0.037848	-0.026556	0.009819	NaN	
YearsAtCompany	0.311309	-0.034055	0.009508	0.069114	NaN	
YearsInCurrentRole	0.212901	0.009932	0.018845	0.060236	NaN	
YearsSinceLastPromotion	0.216513	-0.033229	0.010029	0.054254	NaN	
YearsWithCurrManager	0.202089	-0.026363	0.014406	0.069065	NaN	

26 rows × 26 columns

In [11]:
```
# Heatmap

#sns.heatmap(corre,annot = True, cmap = "Greens")
#plt.title("Correlation matric")

fig = plt.figure(figsize=(6,6))
plt.matshow(corre, cmap='RdBu', fignum=fig.number)
plt.xticks(range(len(corre.columns)), corre.columns, rotation='vertical');
plt.yticks(range(len(corre.columns)), corre.columns);
```

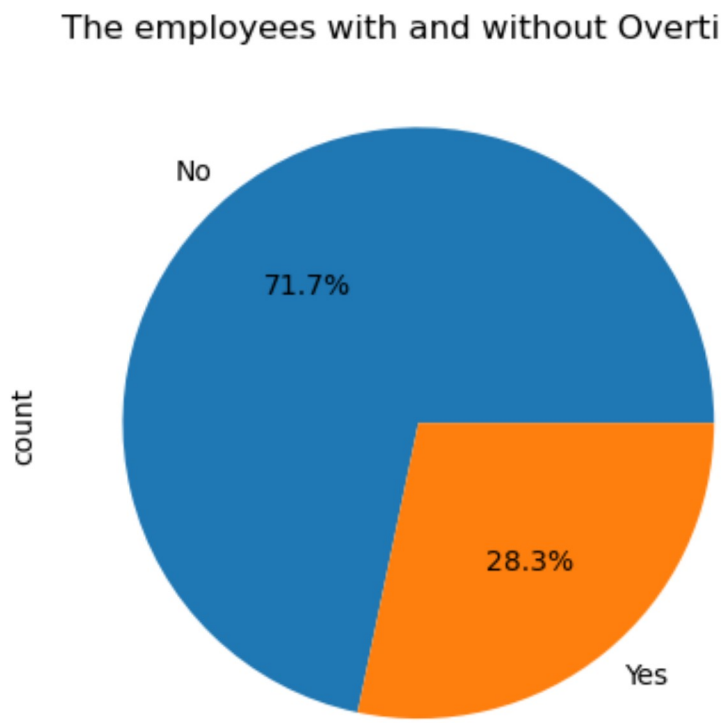


In [12]:
```
##Count Plot
# Overtime count plot
df['OverTime'].value_counts()
```

Out[12]:
```
OverTime
No     1054
Yes     416
Name: count, dtype: int64
```

In [13]:
```
#
ax = df['OverTime'].value_counts().plot.pie(autopct='%1.1f%%',shadow=False)
ax.set_title('The employees with and without Overtime')
```

Out[13]:
```
Text(0.5, 1.0, 'The employees with and without Overtime')
```



In [14]:
```
# statistic
df.describe()
```

Out[14]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumb
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000(
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865:
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024:
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000(
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250(
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500(
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750(
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000(

8 rows × 26 columns

In [15]:
```
#Category the Age
rep = []
for row in df['Age']:
    if row < 31:
        rep.append('Young_age')
    elif row > 50:
        rep.append('Old_age')
    else:
        rep.append('Middle_age')
```

In [16]:
```
# Add a new column 'Categories' with the values are the list 'rep'
df['Categories'] = rep
df
```

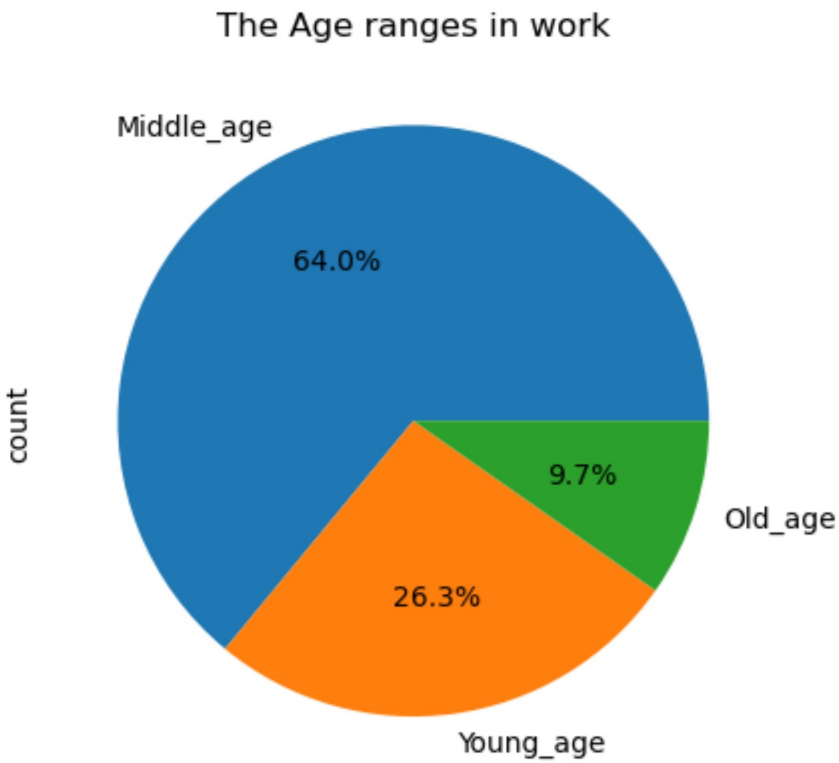
Out[16]:

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

1470 rows × 36 columns

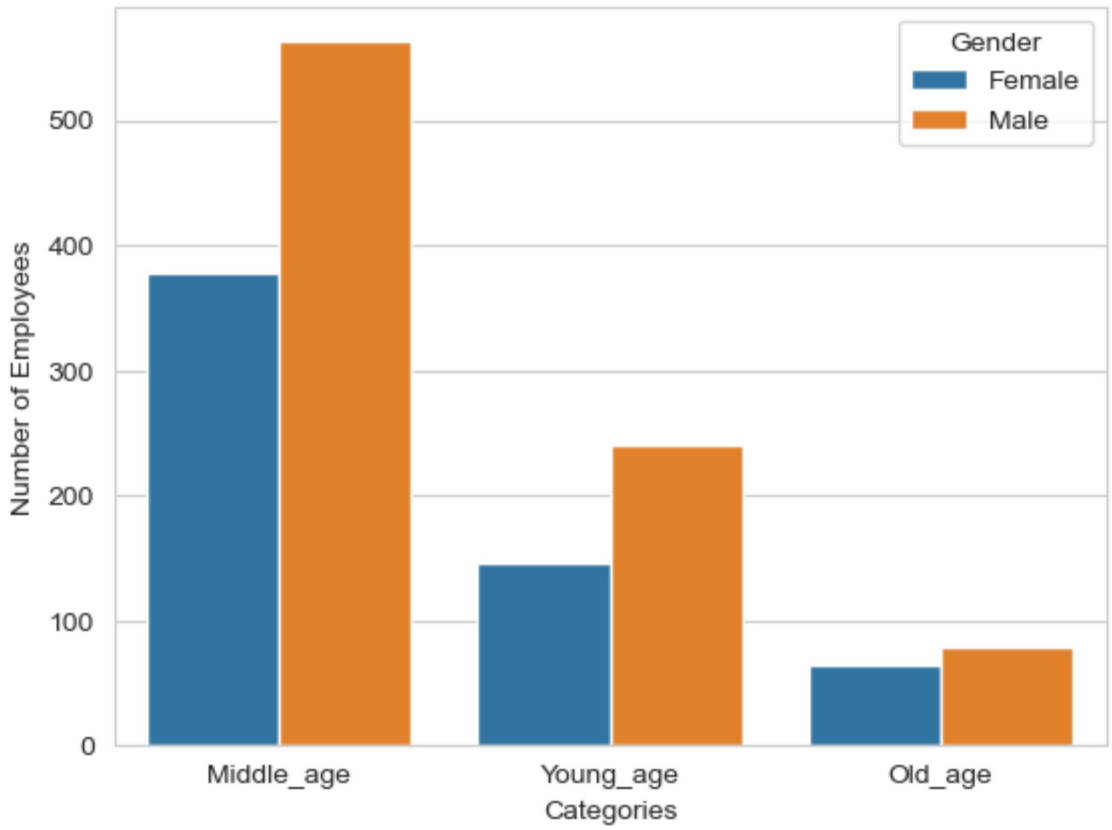
In [17]:
```
##Categories Plot
# Categories count plot

ax = df['Categories'].value_counts().plot.pie(autopct='%1.1f%%',shadow=False)
ax.set_title('The Age ranges in work')

y = df['Categories'].value_counts()['Young_age']
m = df['Categories'].value_counts()['Middle_age']
o = df['Categories'].value_counts()['Old_age']

print(f'>>> There are {y} numbers of employees in the young age.')
print(f'>>> There are {m} numbers of employees in the middle age.')
print(f'>>> There are {o} numbers of employees in the old age.')
```

```
>>> There are 386 numbers of employees in the young age.
>>> There are 941 numbers of employees in the middle age.
>>> There are 143 numbers of employees in the old age.
```



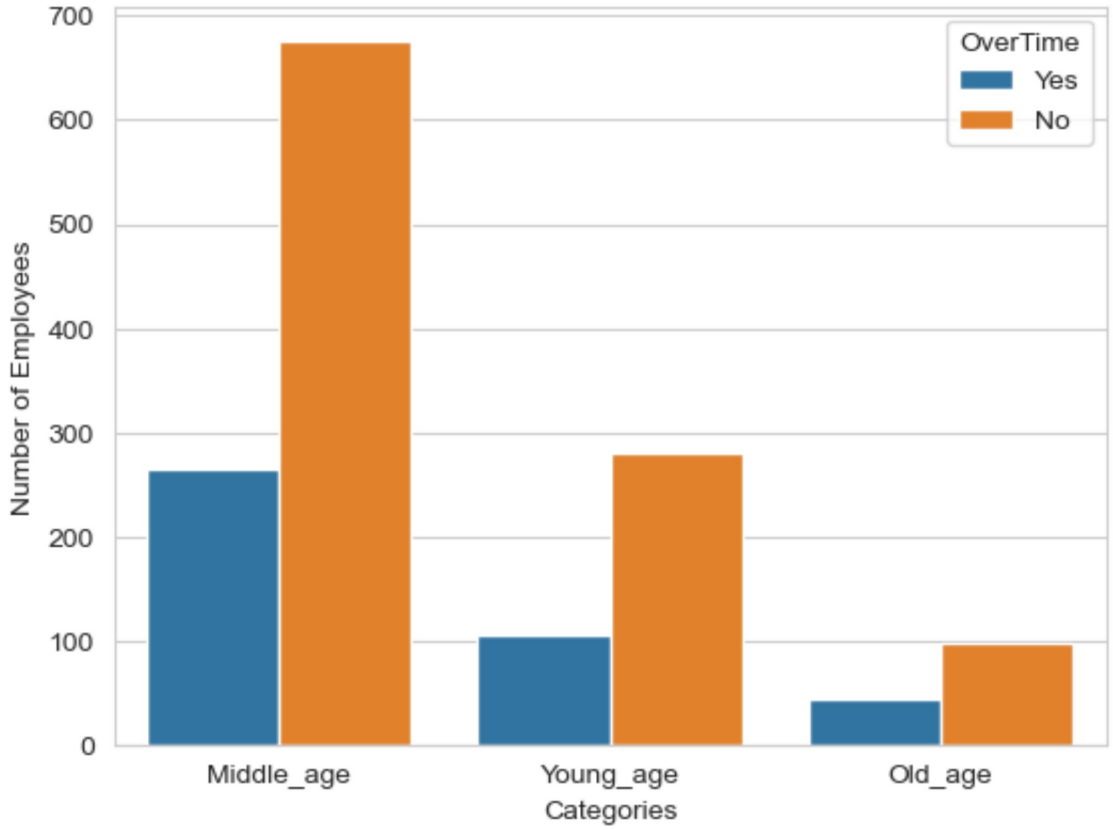
In [18]:
```
#split in group based on 'Gender'
sns.set_style('whitegrid')
ax = sns.barplot(x='Categories', y = 'Age', hue = 'Gender', data = df, estimator =
ax.set_ylabel('Number of Employees')
```

Out[18]:
```
Text(0, 0.5, 'Number of Employees')
```



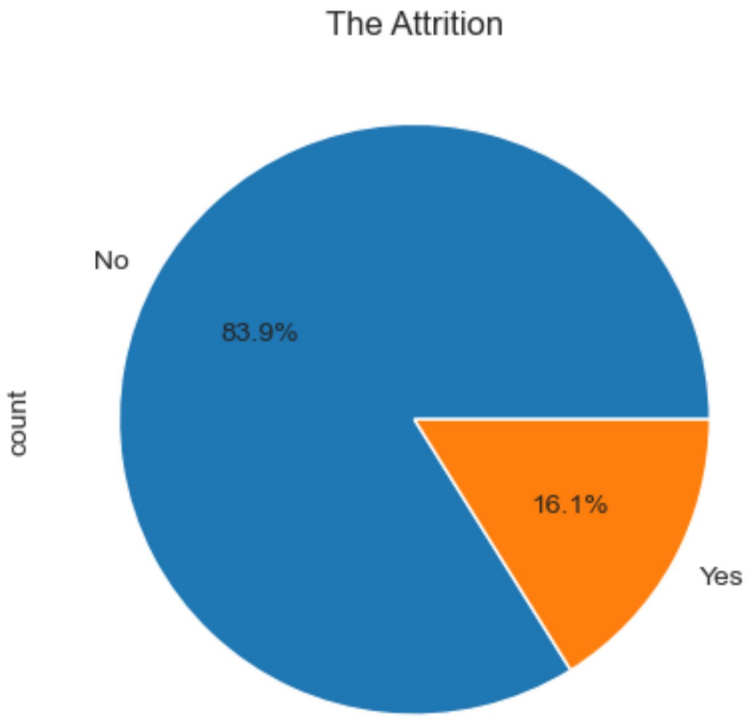
In [19]:
```
#split in group based on 'Overtime'
sns.set_style('whitegrid')
ax = sns.barplot(x='Categories', y = 'Age', hue = 'OverTime', data = df, estimator
ax.set_ylabel('Number of Employees')
```

Out[19]:
```
Text(0, 0.5, 'Number of Employees')
```



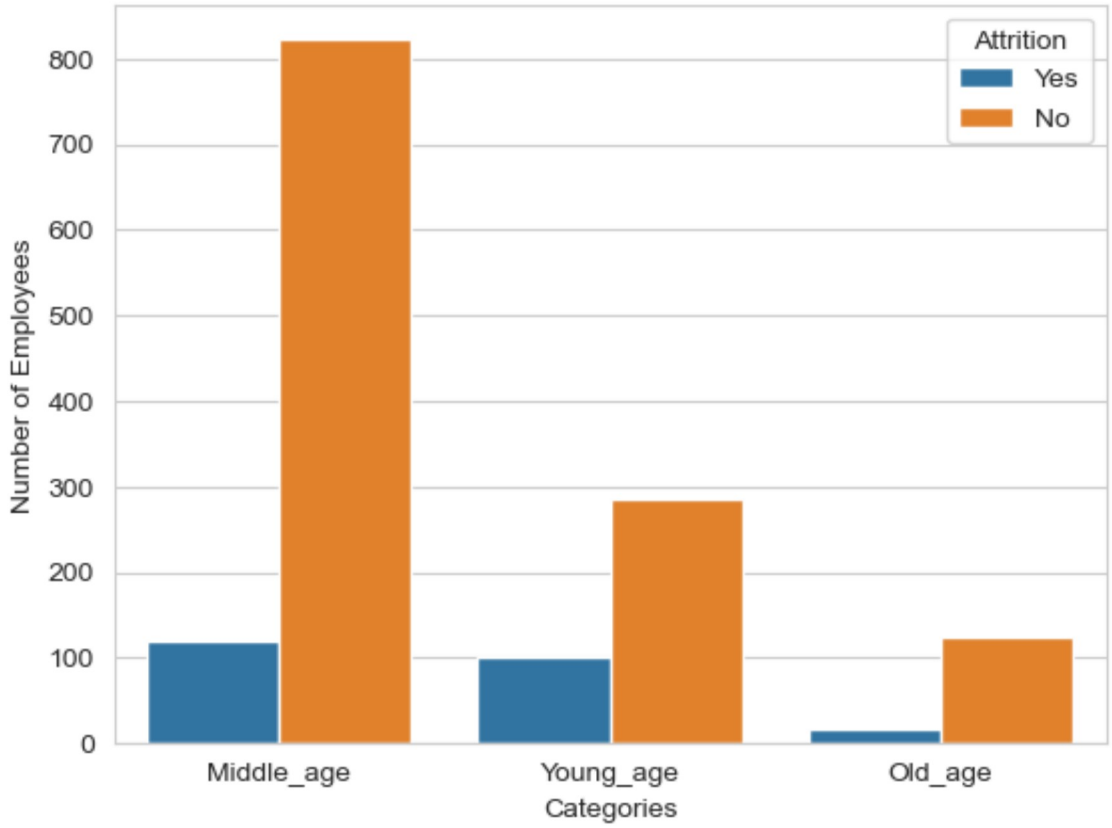
In [20]:
```
##Categories Plot
# Categories count plot

ax = df['Attrition'].value_counts().plot.pie(autopct='%1.1f%%',shadow=False)
ax.set_title('The Attrition')

y = df['Attrition'].value_counts()['Yes']
n = df['Attrition'].value_counts()['No']

print(f'>>> There are {n} numbers of employees in active.')
print(f'>>> There are {y} numbers of employees not in active.')
```

```
>>> There are 1233 numbers of employees in active.
>>> There are 237 numbers of employees not in active.
```



In [21]:
```
#split in group based on 'Attrition'
sns.set_style('whitegrid')
ax = sns.barplot(x='Categories', y = 'Age', hue = 'Attrition', data = df, estimator
ax.set_ylabel('Number of Employees')
```

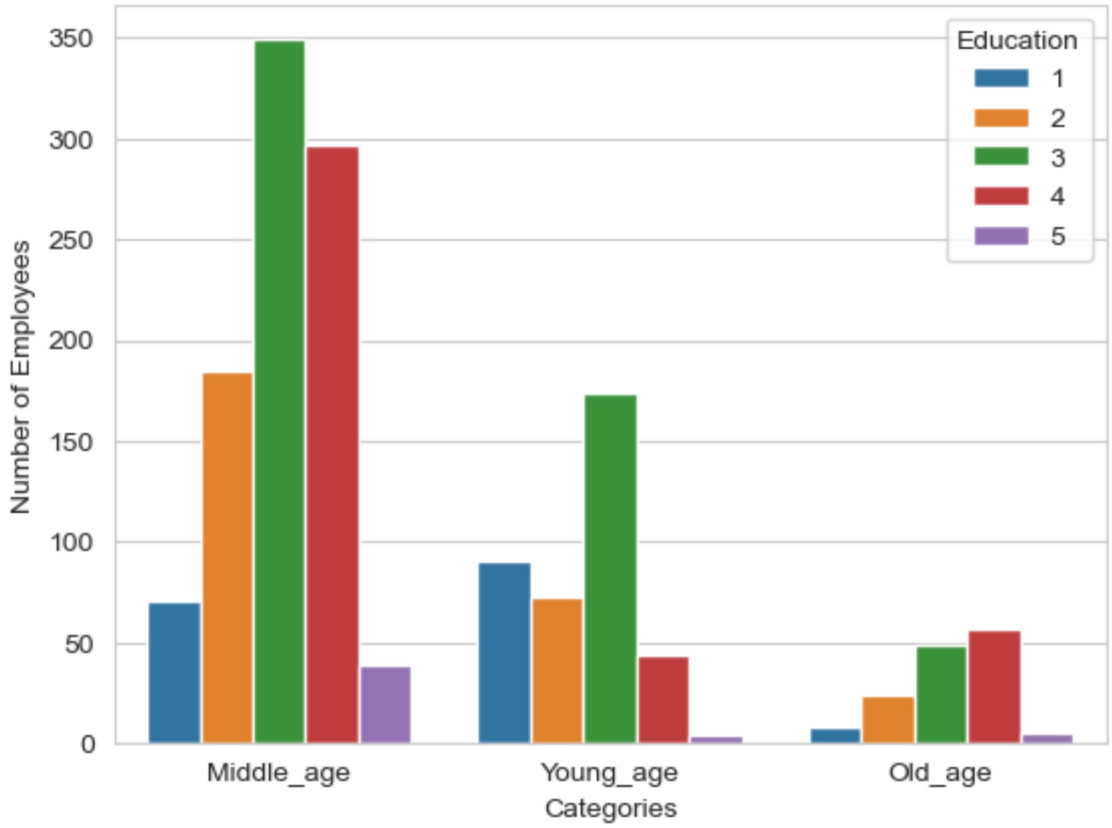
Out[21]:
```
Text(0, 0.5, 'Number of Employees')
```



In [22]:
```
# Histogram of each feature
df.hist(bins=10,figsize=(15,15),color='c')
plt.show()
```

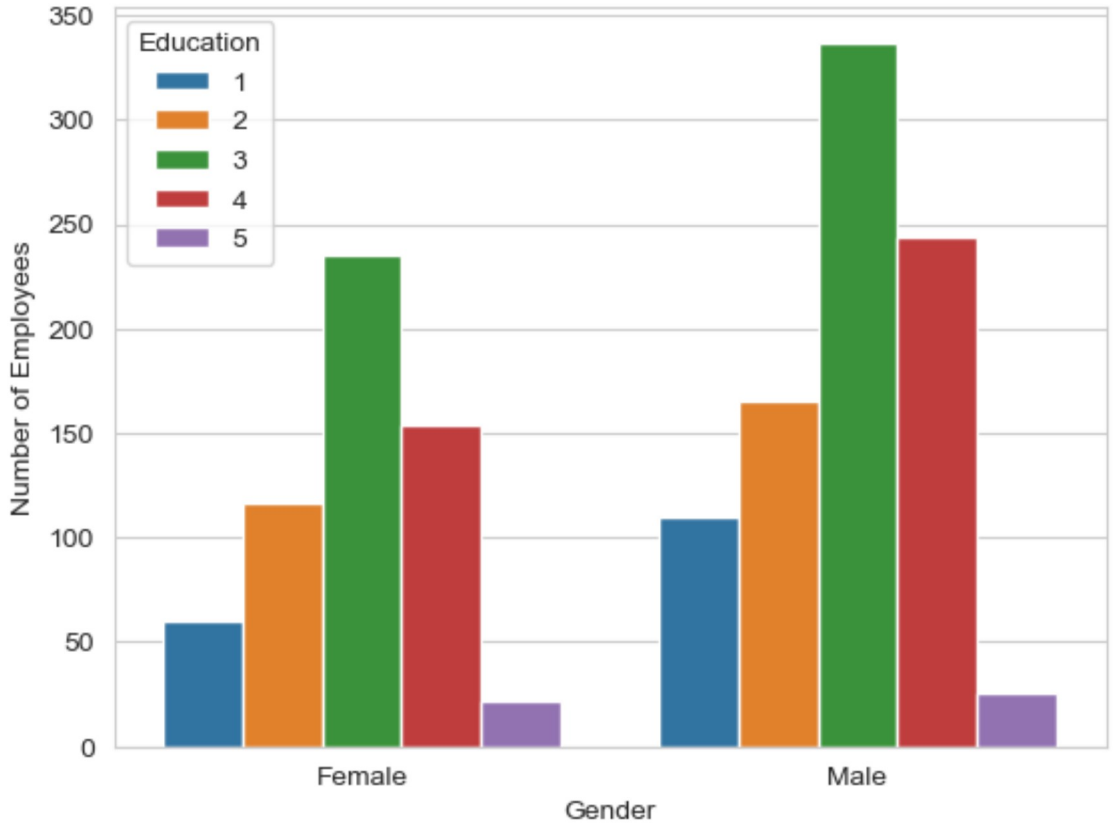
In [23]:
```
#split in group based on 'Education'
sns.set_style('whitegrid')
ax = sns.barplot(x='Categories', y = 'Age', hue = 'Education', data = df, estimator
ax.set_ylabel('Number of Employees')
```

Out[23]:
```
Text(0, 0.5, 'Number of Employees')
```



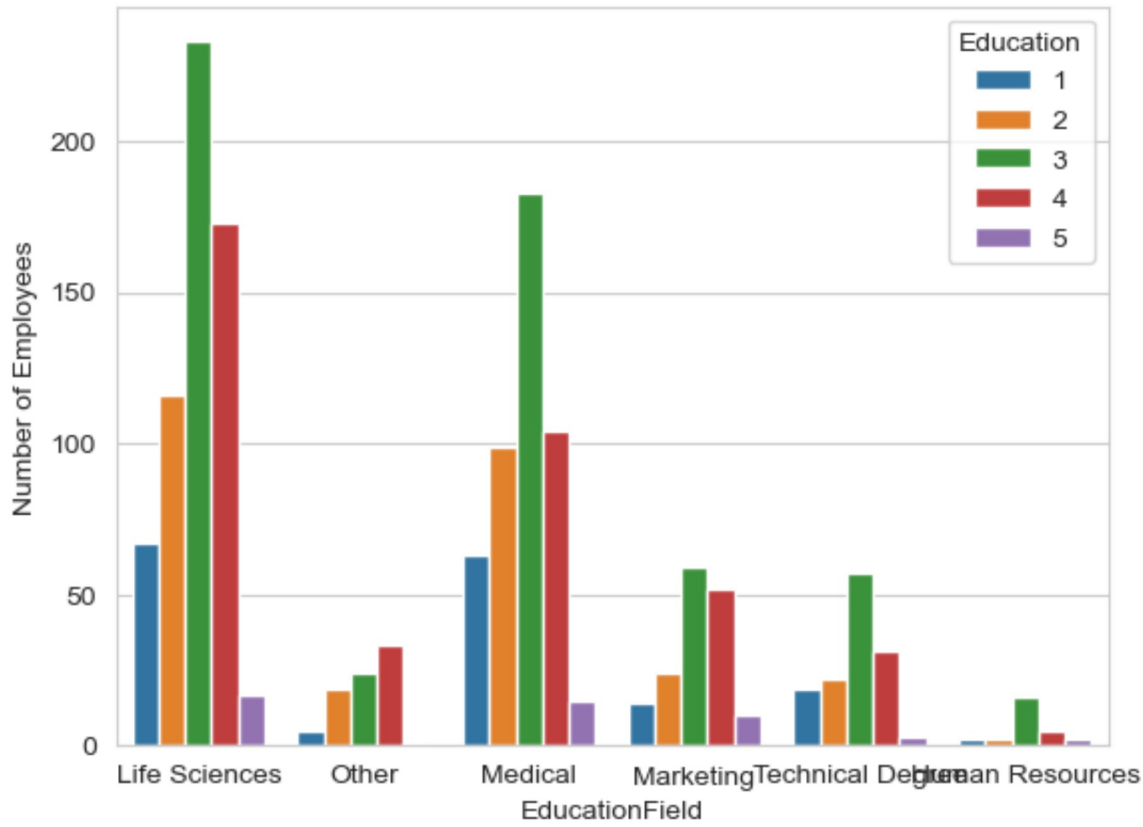
In [24]:
```
#split in group based on 'Education'
sns.set_style('whitegrid')
ax = sns.barplot(x='Gender', y = 'Age', hue = 'Education', data = df, estimator = l
ax.set_ylabel('Number of Employees')
```

Out[24]:
```
Text(0, 0.5, 'Number of Employees')
```



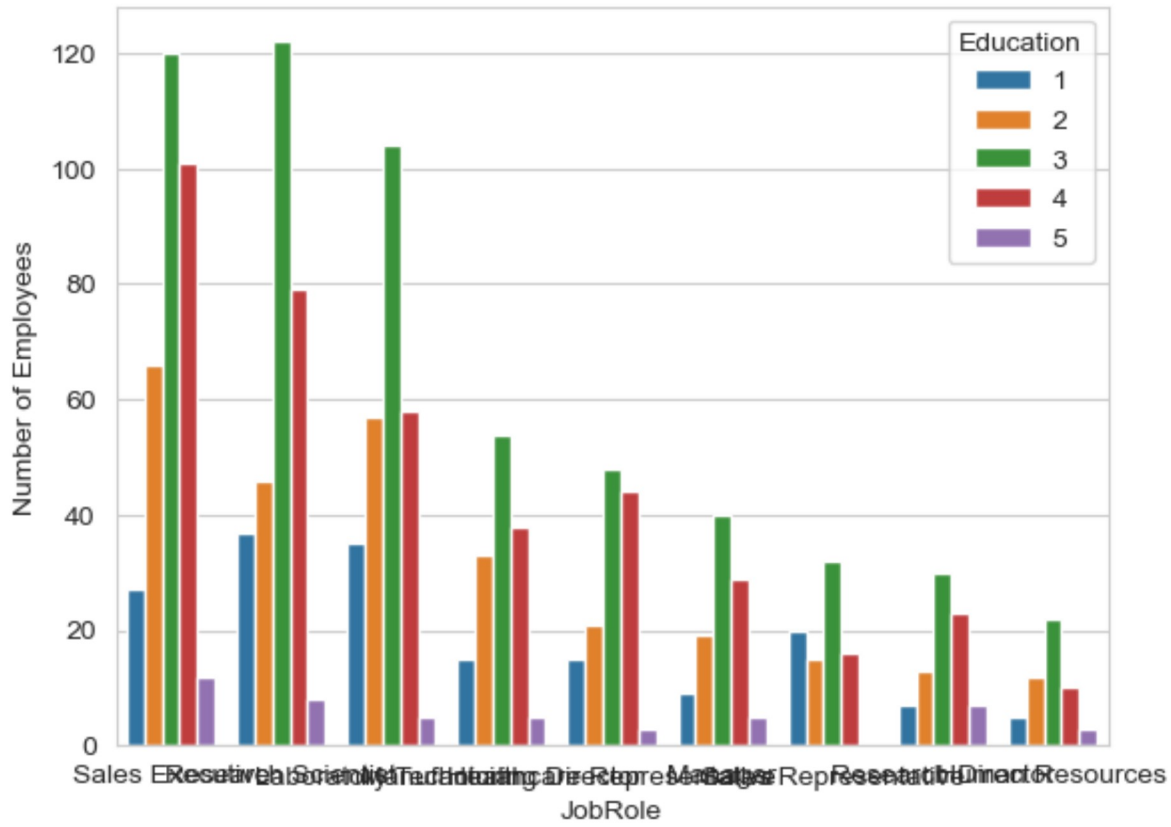
In [25]:
```
#split in group based on 'Education'
sns.set_style('whitegrid')
ax = sns.barplot(x='EducationField', y = 'Age', hue = 'Education', data = df, estim
ax.set_ylabel('Number of Employees')
```

Out[25]:
```
Text(0, 0.5, 'Number of Employees')
```



In [26]:
```
#split in group based on 'Education'
sns.set_style('whitegrid')
ax = sns.barplot(x='JobRole', y = 'Age', hue = 'Education', data = df, estimator =
ax.set_ylabel('Number of Employees')
```

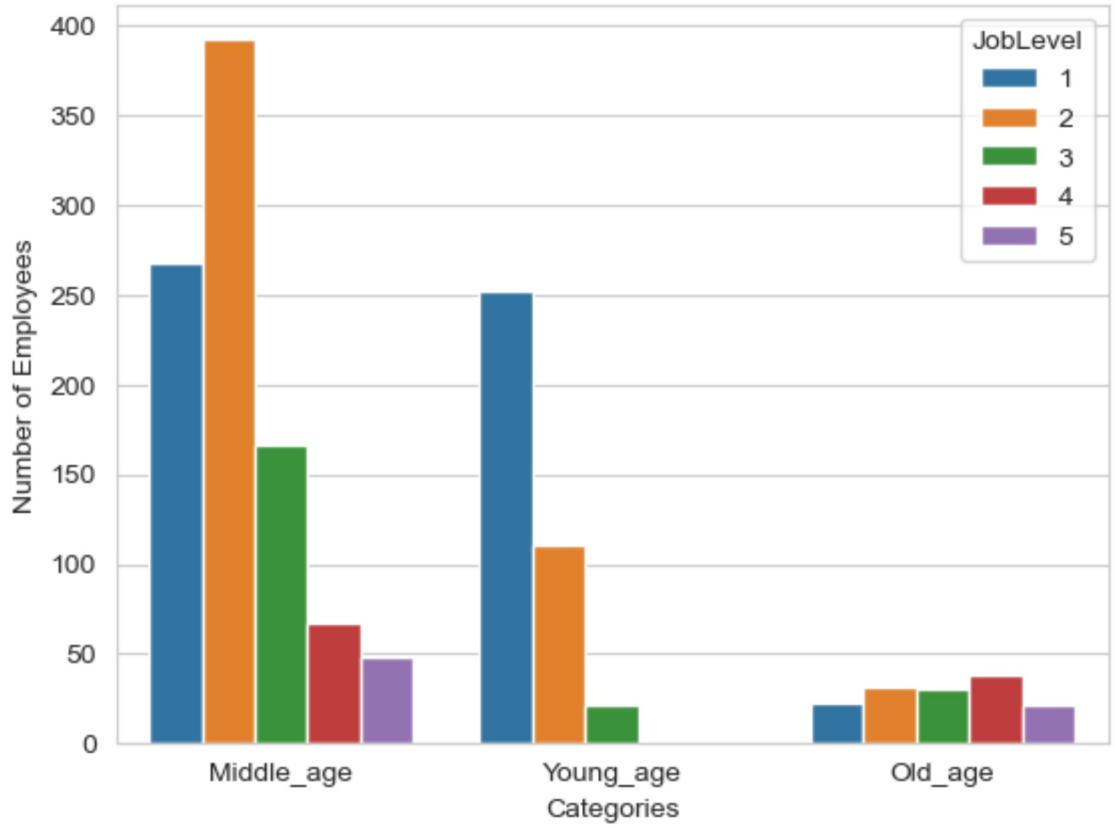
Out[26]:
```
Text(0, 0.5, 'Number of Employees')
```



In [ ]:

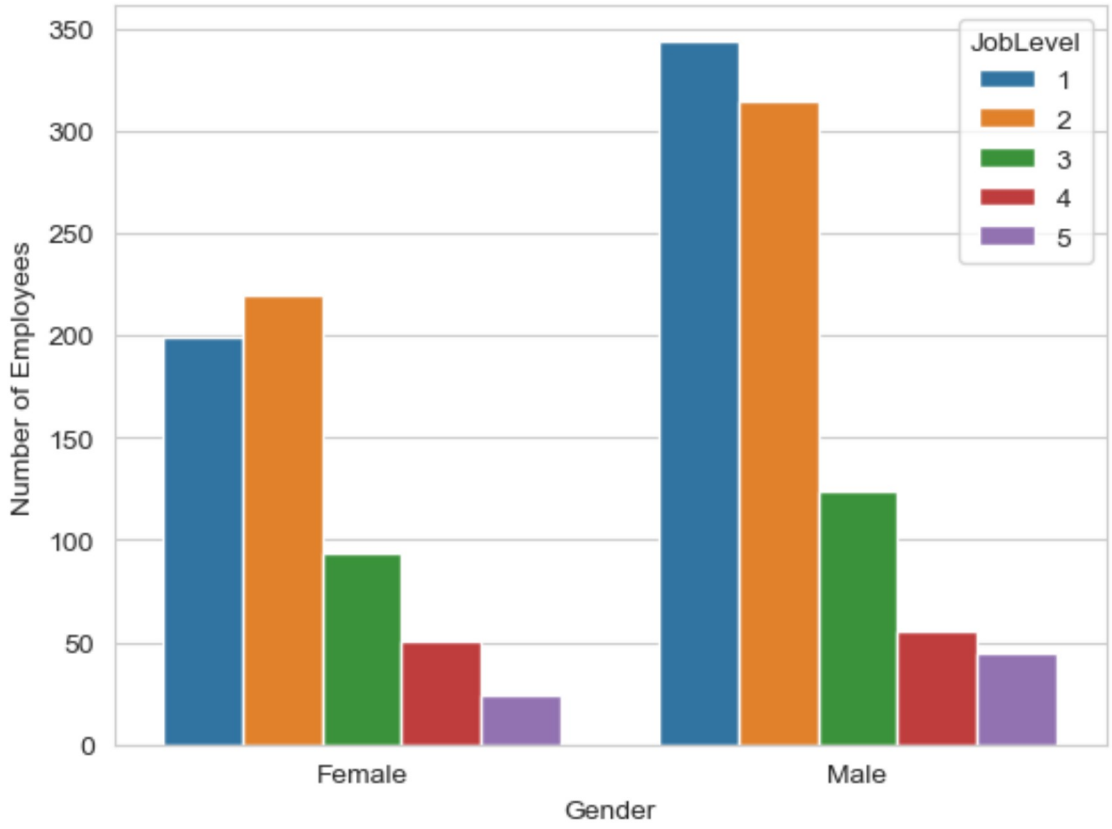
In [27]:
```
#split in group based on 'JobLevel'
sns.set_style('whitegrid')
ax = sns.barplot(x='Categories', y = 'Age', hue = 'JobLevel', data = df, estimator
ax.set_ylabel('Number of Employees')
```

Out[27]:
```
Text(0, 0.5, 'Number of Employees')
```

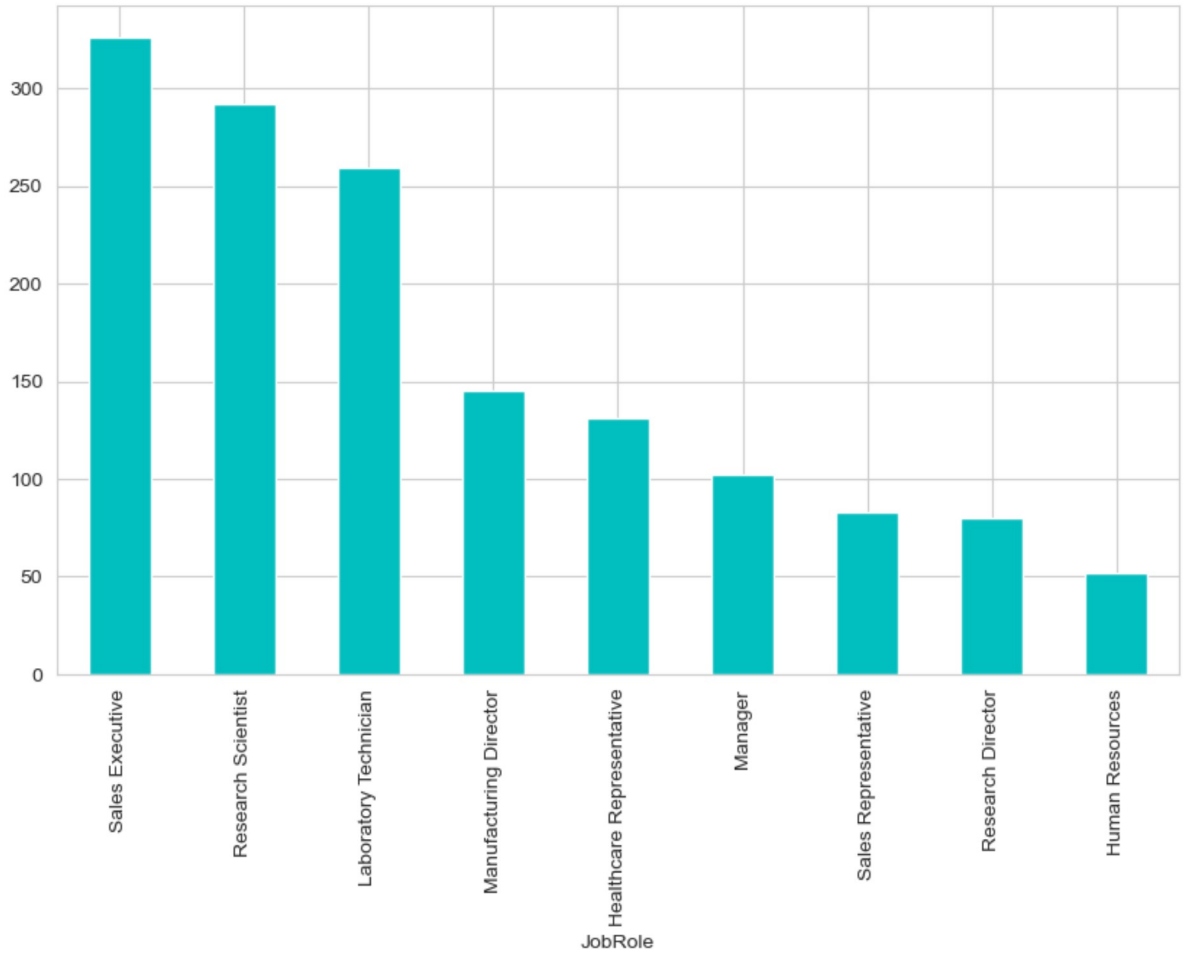


In [28]:
```
#split in group based on 'JobLevel'
sns.set_style('whitegrid')
ax = sns.barplot(x='Gender', y = 'Age', hue = 'JobLevel', data = df, estimator = le
ax.set_ylabel('Number of Employees')
```

Out[28]:
```
Text(0, 0.5, 'Number of Employees')
```

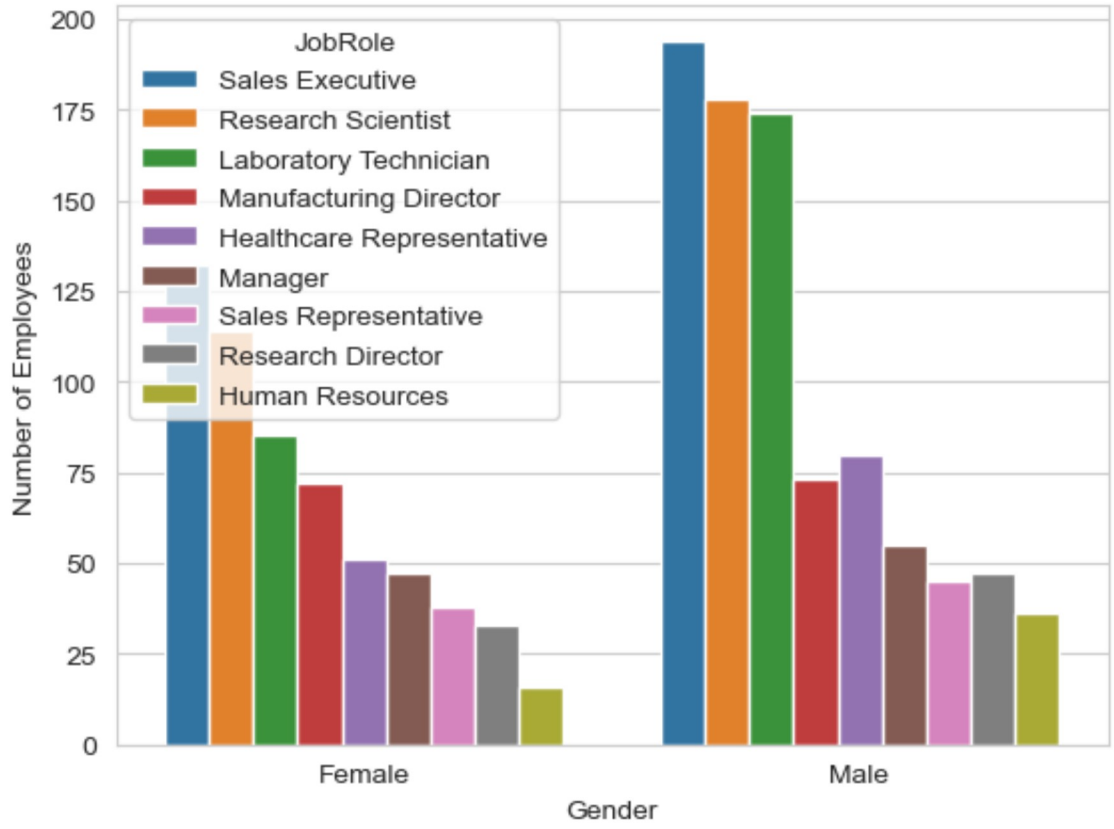


In [29]:
```
#
ax = df['JobRole'].value_counts().plot(kind='bar', figsize=(10,6), color ='c')
```



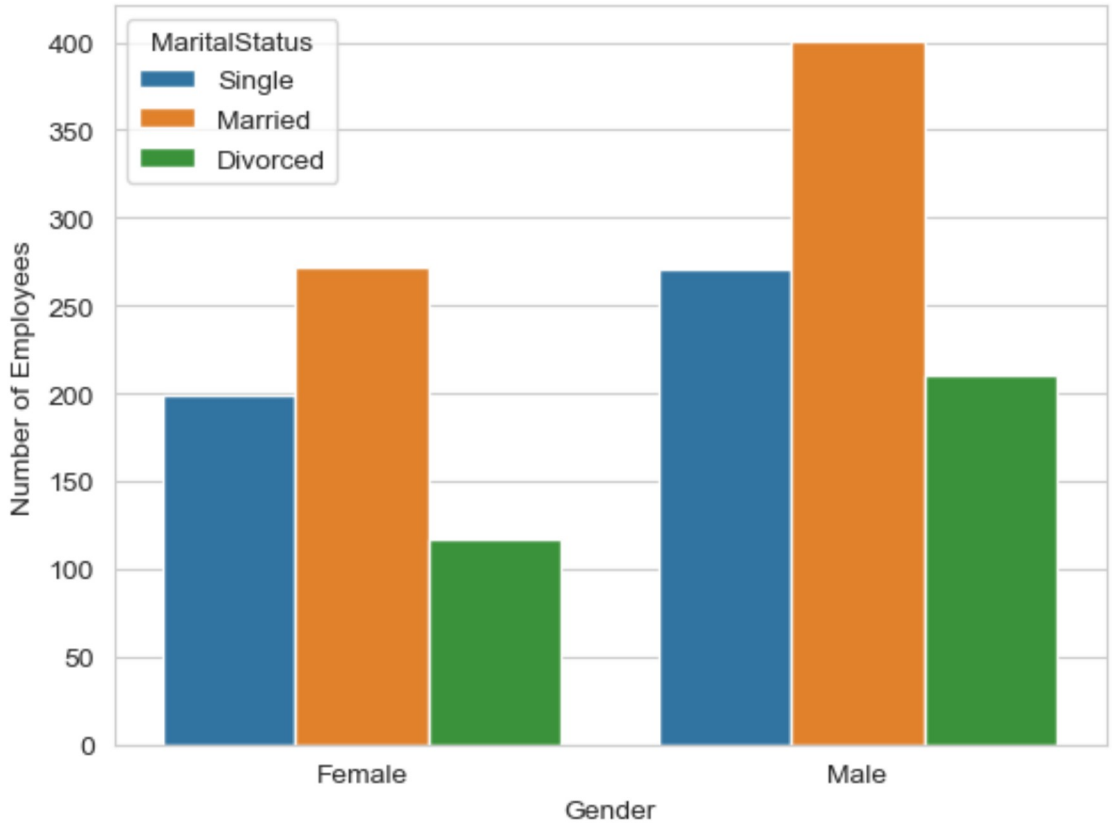
In [30]:
```
#split in group based on 'JobRole'
sns.set_style('whitegrid')
ax = sns.barplot(x='Gender', y = 'Age', hue = 'JobRole', data = df, estimator = len
ax.set_ylabel('Number of Employees')
```

Out[30]:
```
Text(0, 0.5, 'Number of Employees')
```



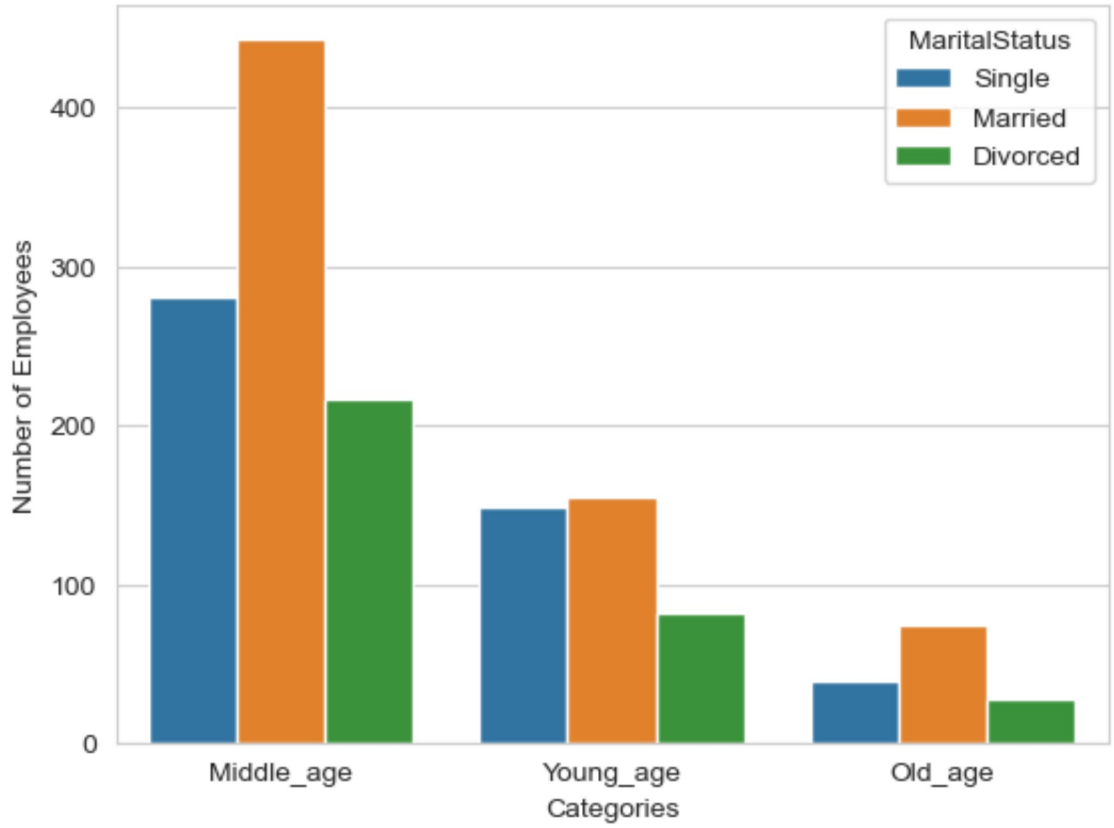
In [31]:
```
#split in group based on 'MaritalStatus'
sns.set_style('whitegrid')
ax = sns.barplot(x='Gender', y = 'Age', hue = 'MaritalStatus', data = df, estimator
ax.set_ylabel('Number of Employees')
```

Out[31]:
```
Text(0, 0.5, 'Number of Employees')
```



In [32]:
```
#split in group based on 'MaritalStatus'
sns.set_style('whitegrid')
ax = sns.barplot(x='Categories', y = 'Age', hue = 'MaritalStatus', data = df, estim
ax.set_ylabel('Number of Employees')
```

Out[32]:
```
Text(0, 0.5, 'Number of Employees')
```

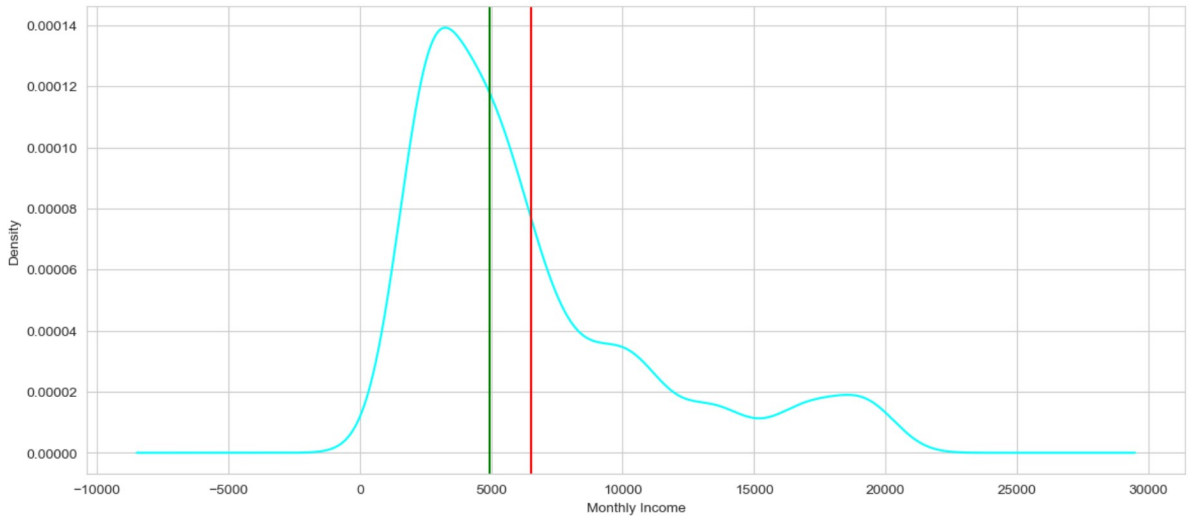


In [33]:
```
#
df['MonthlyIncome'].describe()
```

Out[33]:
```
count     1470.000000
mean      6502.931293
std       4707.956783
min       1009.000000
25%       2911.000000
50%       4919.000000
75%       8379.000000
max      19999.000000
Name: MonthlyIncome, dtype: float64
```

In [34]:
```
ax = df['MonthlyIncome'].plot(kind='density', figsize=(14,6), color ='cyan')
ax.axvline(df['MonthlyIncome'].mean(), color='red')
ax.axvline(df['MonthlyIncome'].median(), color='green')
ax.set_xlabel('Monthly Income')
```

Out[34]:
```
Text(0.5, 0, 'Monthly Income')
```

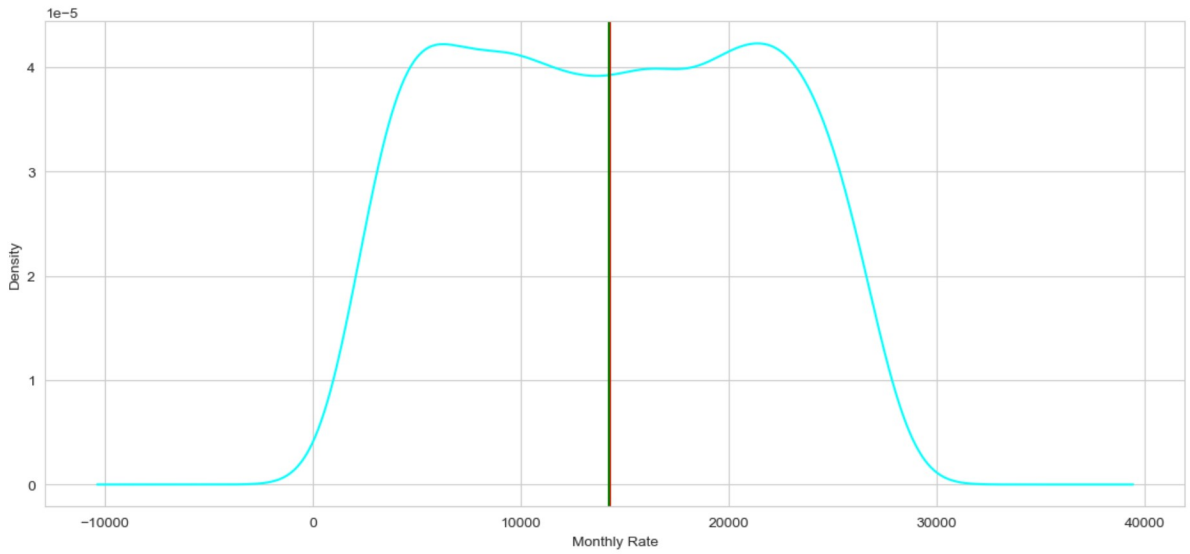


In [35]:
```
#
df['MonthlyRate'].describe()
```

Out[35]:
```
count     1470.000000
mean     14313.103401
std       7117.786044
min       2094.000000
25%       8047.000000
50%      14235.500000
75%      20461.500000
max      26999.000000
Name: MonthlyRate, dtype: float64
```

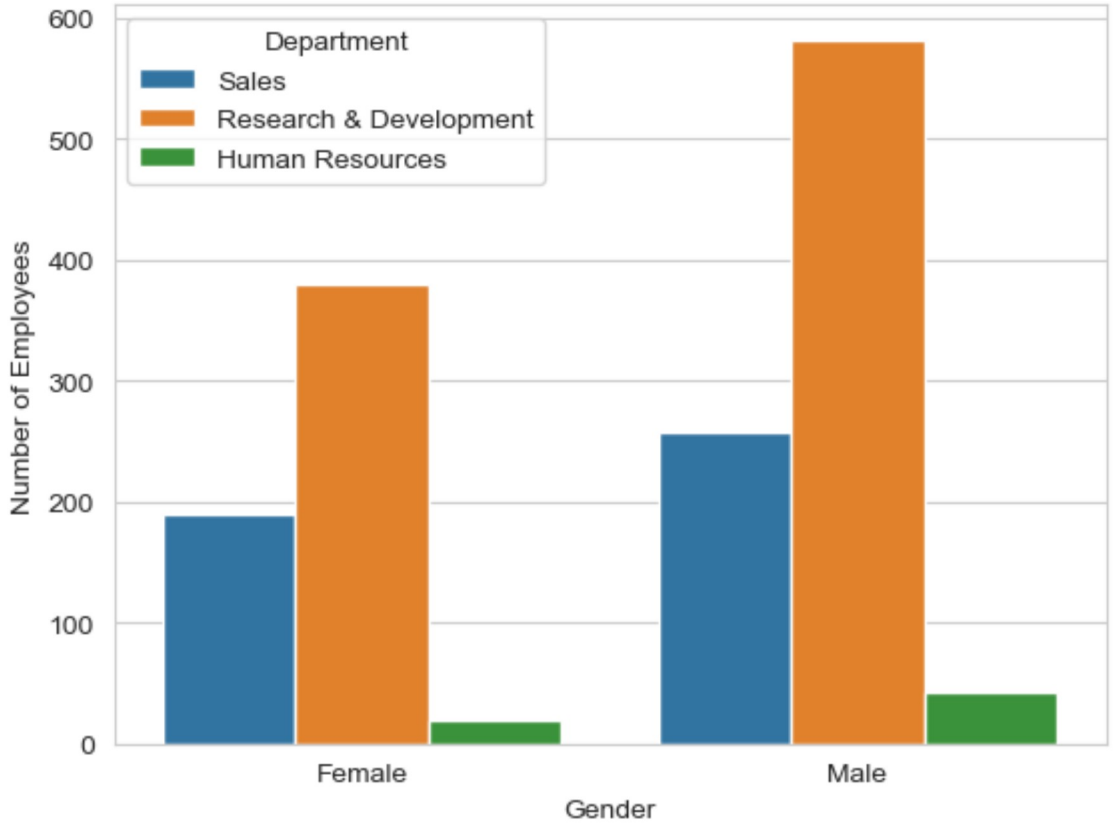
In [36]:
```
#
ax = df['MonthlyRate'].plot(kind='density', figsize=(14,6), color ='cyan')
ax.axvline(df['MonthlyRate'].mean(), color='red')
ax.axvline(df['MonthlyRate'].median(), color='green')
ax.set_xlabel('Monthly Rate')
```

Out[36]:
```
Text(0.5, 0, 'Monthly Rate')
```



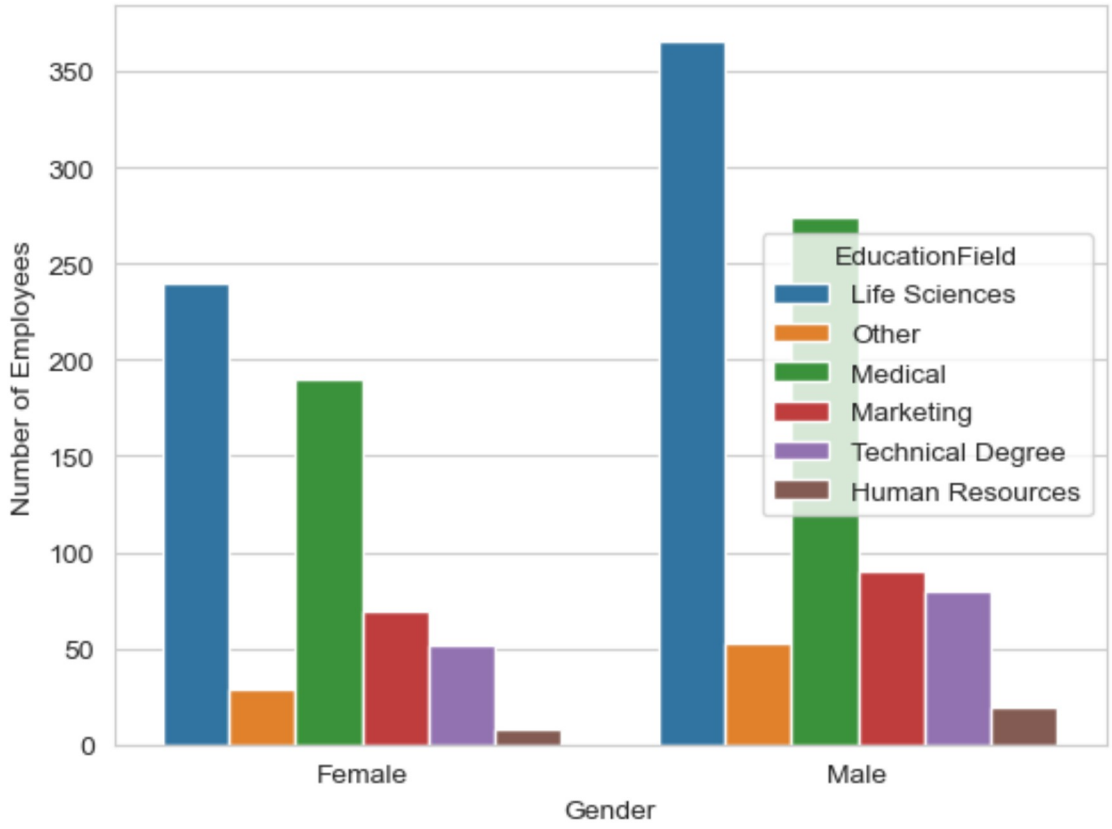
In [37]:
```
#split in group based on 'Department'
sns.set_style('whitegrid')
ax = sns.barplot(x='Gender', y = 'Age', hue = 'Department', data = df, estimator =
ax.set_ylabel('Number of Employees')
```

Out[37]:
```
Text(0, 0.5, 'Number of Employees')
```



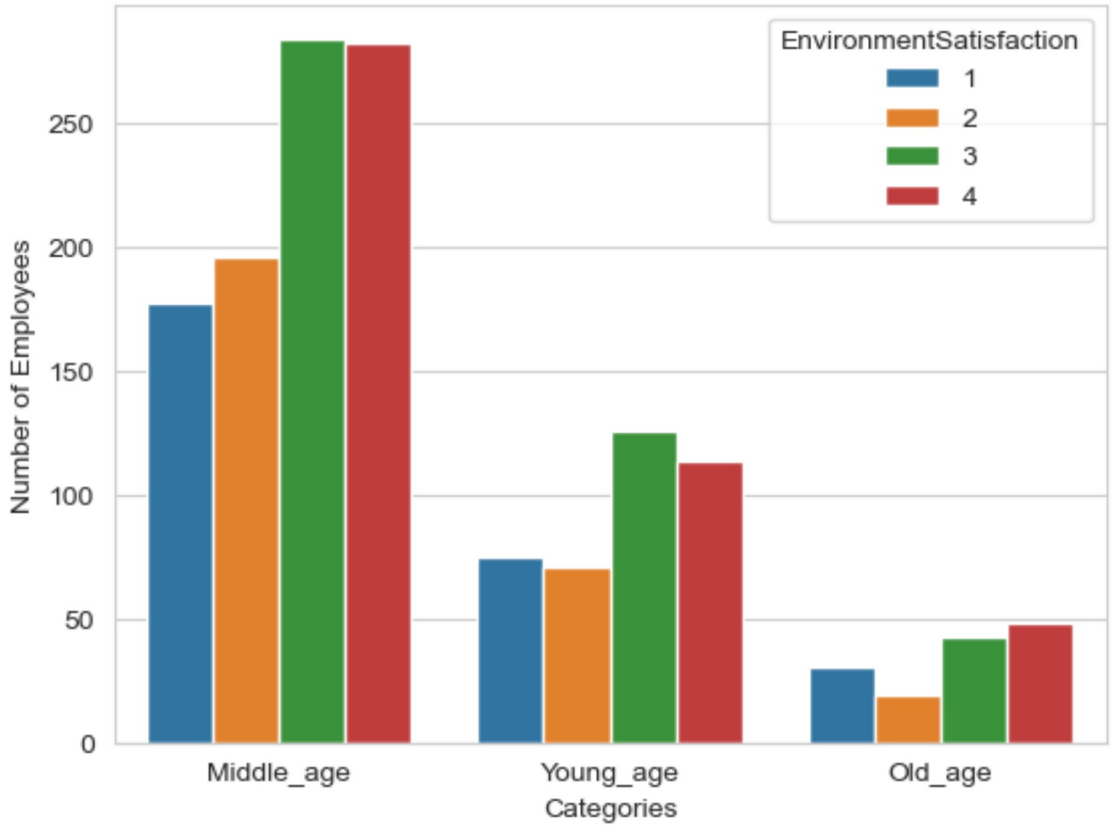
In [38]:
```
#split in group based on 'EducationField'
sns.set_style('whitegrid')
ax = sns.barplot(x='Gender', y = 'Age', hue = 'EducationField', data = df, estimato
ax.set_ylabel('Number of Employees')
```

Out[38]:
```
Text(0, 0.5, 'Number of Employees')
```



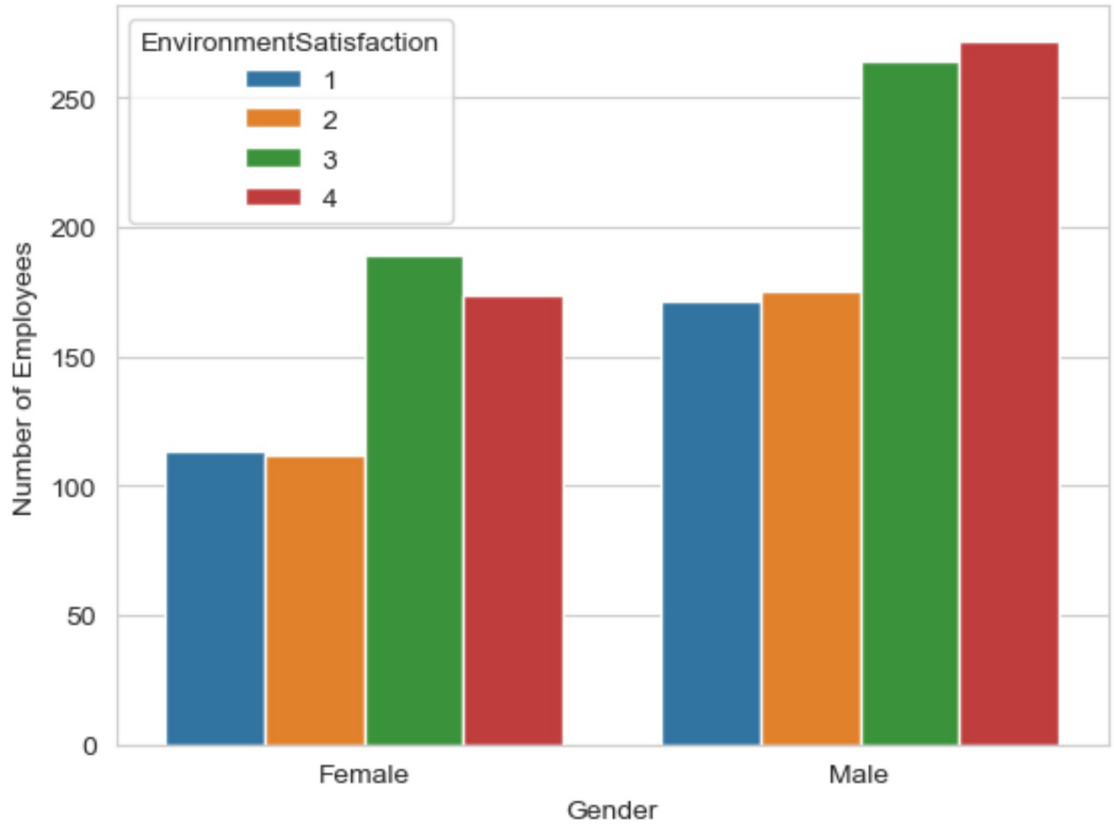
In [39]:
```
#split in group based on 'EnvironmentSatisfaction'
sns.set_style('whitegrid')
ax = sns.barplot(x='Categories', y = 'Age', hue = 'EnvironmentSatisfaction', data =
ax.set_ylabel('Number of Employees')
```

Out[39]:
```
Text(0, 0.5, 'Number of Employees')
```



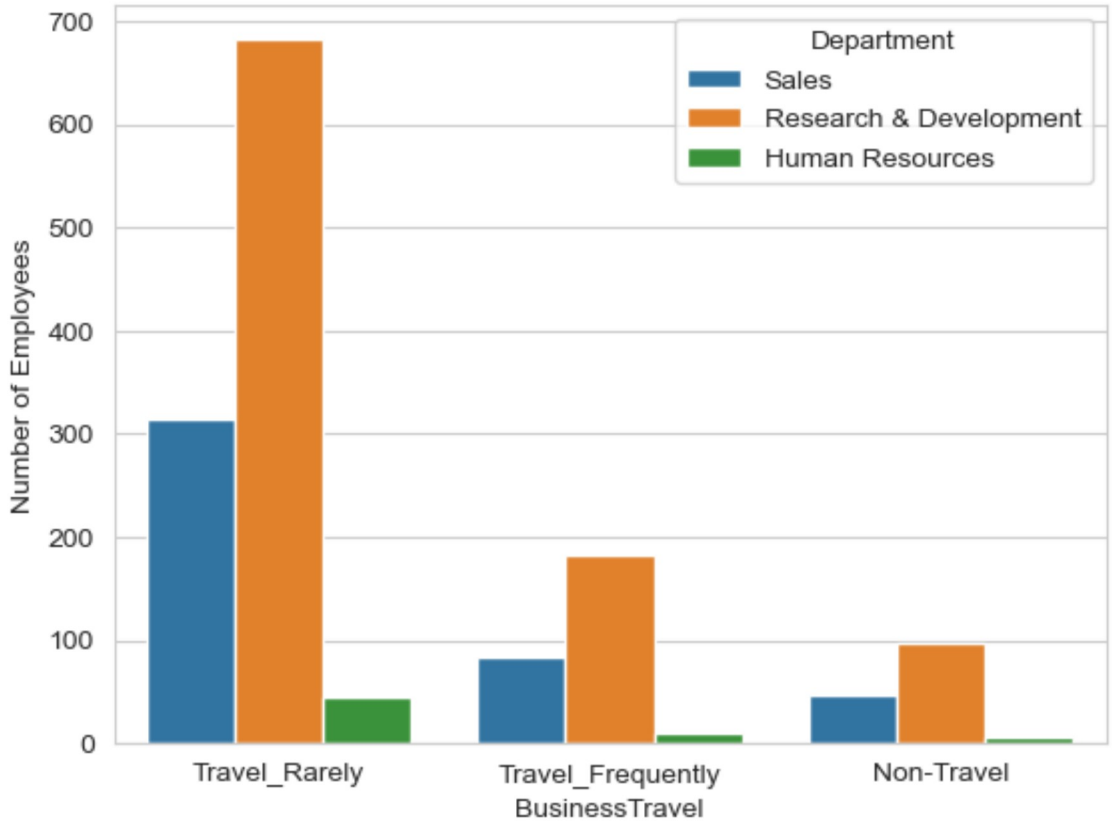
In [40]:
```
#split in group based on 'EnvironmentSatisfaction'
sns.set_style('whitegrid')
ax = sns.barplot(x='Gender', y = 'Age', hue = 'EnvironmentSatisfaction', data = df,
ax.set_ylabel('Number of Employees')
```

Out[40]:
```
Text(0, 0.5, 'Number of Employees')
```



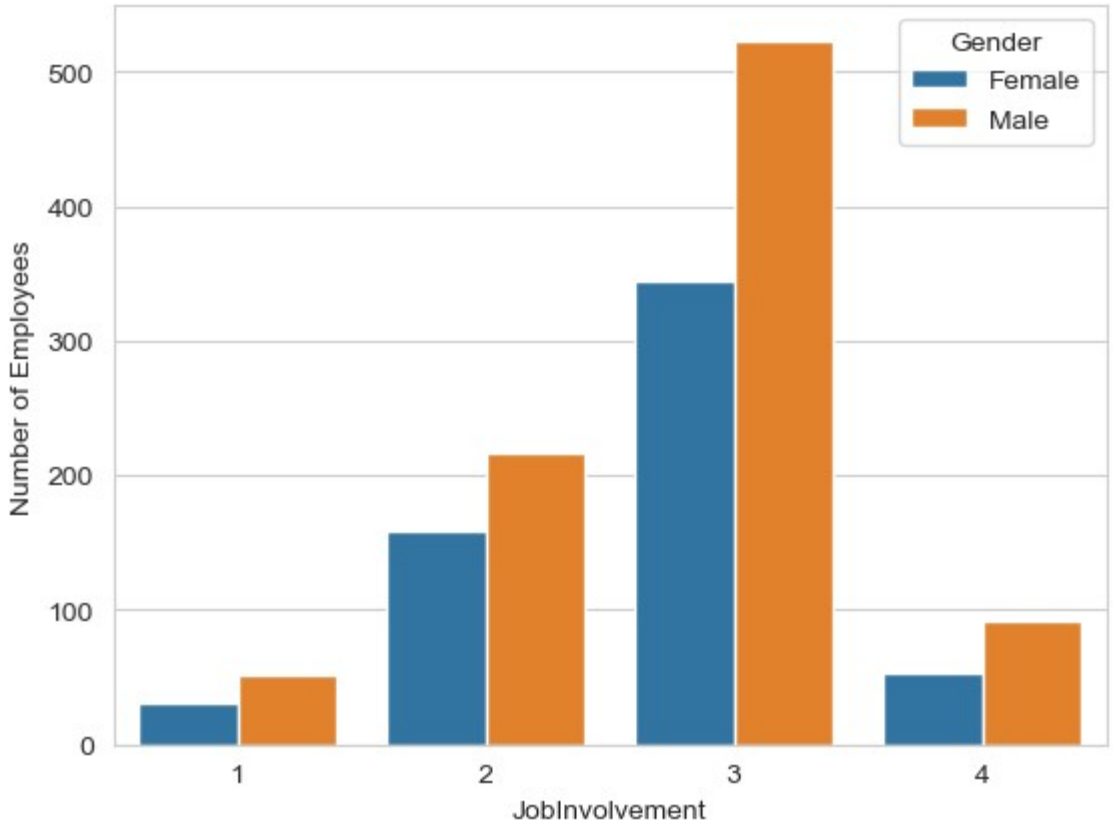
In [41]:
```
#split in group based on 'Department'
sns.set_style('whitegrid')
ax = sns.barplot(x='BusinessTravel', y = 'Age', hue = 'Department', data = df, esti
ax.set_ylabel('Number of Employees')
```

Out[41]:
```
Text(0, 0.5, 'Number of Employees')
```



In [42]:
```
#split in group based on 'Gender'
sns.set_style('whitegrid')
ax = sns.barplot(x='JobInvolvement', y = 'EmployeeCount', hue = 'Gender', data = df
ax.set_ylabel('Number of Employees')
```

Out[42]:
```
Text(0, 0.5, 'Number of Employees')
```


Gender
Female
Male
Number of Employees
0
100
200
300
400
500
1
2
3
4
JobInvolvement