In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
#importing all libraries
```

In [2]:
```
df=pd.read_csv("Sleep_health_and_lifestyle_dataset.csv")
df
#reaing CSV(Common Seperate Values)
```

Out[2]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blo Pressu
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126
1	2	Male	28	Doctor	6.2	6	60	8	Normal	125
2	3	Male	28	Doctor	6.2	6	60	8	Normal	125
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	140
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	140
...	...	...	...	...	...	...	...	...	...	
369	370	Female	59	Nurse	8.1	9	75	3	Overweight	140
370	371	Female	59	Nurse	8.0	9	75	3	Overweight	140
371	372	Female	59	Nurse	8.1	9	75	3	Overweight	140
372	373	Female	59	Nurse	8.1	9	75	3	Overweight	140
373	374	Female	59	Nurse	8.1	9	75	3	Overweight	140

374 rows × 13 columns

In [3]:
```
-Person ID: An identifier for each individual.
-Gender: The gender of the person (Male/Female).
-Age: The age of the person in years.
-Occupation: The occupation or profession of the person.
-Sleep Duration (hours): The number of hours the person sleeps per day.
-Quality of Sleep (scale: 1-10): A subjective rating of the quality of sle
-Physical Activity Level (minutes/day): The number of minutes the person e
-Stress Level (scale: 1-10): A subjective rating of the stress level exper
-BMI Category: The BMI category of the person (e.g., Underweight, Normal,
-Blood Pressure (systolic/diastolic): The blood pressure measurement of th
    diastolic pressure.
-Heart Rate (bpm): The resting heart rate of the person in beats per minut
-Daily Steps: The number of steps the person takes per day.
-Sleep Disorder: The presence or absence of a sleep disorder in the person


-The Sleep Health and Lifestyle Dataset comprises 400 rows and 13 columns,
sleep and daily habits. It includes details such as gender, age, occupatio
activity level, stress levels, BMI category, blood pressure, heart rate, d
disorders.
```

```
  Cell In[3], line 1
    -Person ID: An identifier for each individual.
            ^
SyntaxError: invalid syntax
```

Total number of rows and columns

In [ ]:
```
print("Number of Row",df.shape[0])
print("Number of Column",df.shape[1])
#total number of rows and columns
```

EDA(Exploratory Data Analysis)

In [4]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 374 entries, 0 to 373
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Person ID                374 non-null    int64  
 1   Gender                   374 non-null    object 
 2   Age                      374 non-null    int64  
 3   Occupation               374 non-null    object 
 4   Sleep Duration           374 non-null    float64
 5   Quality of Sleep         374 non-null    int64  
 6   Physical Activity Level  374 non-null    int64  
 7   Stress Level             374 non-null    int64  
 8   BMI Category             374 non-null    object 
 9   Blood Pressure           374 non-null    object 
 10  Heart Rate               374 non-null    int64  
 11  Daily Steps              374 non-null    int64  
 12  Sleep Disorder           374 non-null    object 
dtypes: float64(1), int64(7), object(5)
memory usage: 38.1+ KB
```

In [5]:
```
1 df.isna().sum()
2 #Checj for null values
```

Out[5]:
```
Person ID                  0
Gender                     0
Age                        0
Occupation                 0
Sleep Duration             0
Quality of Sleep           0
Physical Activity Level    0
Stress Level               0
BMI Category               0
Blood Pressure             0
Heart Rate                 0
Daily Steps                0
Sleep Disorder             0
dtype: int64
```

In [6]:
```
df.isnull().sum()
#Showing null values
```

Out[6]:
```
Person ID                  0
Gender                     0
Age                        0
Occupation                 0
Sleep Duration             0
Quality of Sleep           0
Physical Activity Level    0
Stress Level               0
BMI Category               0
Blood Pressure             0
Heart Rate                 0
Daily Steps                0
Sleep Disorder             0
dtype: int64
```

In [7]:
```
df.columns
#name of all columns
```

Out[7]:
```
Index(['Person ID', 'Gender', 'Age', 'Occupation', 'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level',
       'BMI Category', 'Blood Pressure', 'Heart Rate', 'Daily Steps',
       'Sleep Disorder'],
      dtype='object')
```

In [8]:
```
df.rename(columns={"Occupation":"Profession"})
#Changing name of column/temporary
```

Out[8]:

	Person ID	Gender	Age	Profession	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blo Pressu
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126
1	2	Male	28	Doctor	6.2	6	60	8	Normal	125
2	3	Male	28	Doctor	6.2	6	60	8	Normal	125
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	140
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	140
...	...	...	...	...	...	...	...	...	...	
369	370	Female	59	Nurse	8.1	9	75	3	Overweight	140
370	371	Female	59	Nurse	8.0	9	75	3	Overweight	140
371	372	Female	59	Nurse	8.1	9	75	3	Overweight	140
372	373	Female	59	Nurse	8.1	9	75	3	Overweight	140
373	374	Female	59	Nurse	8.1	9	75	3	Overweight	140

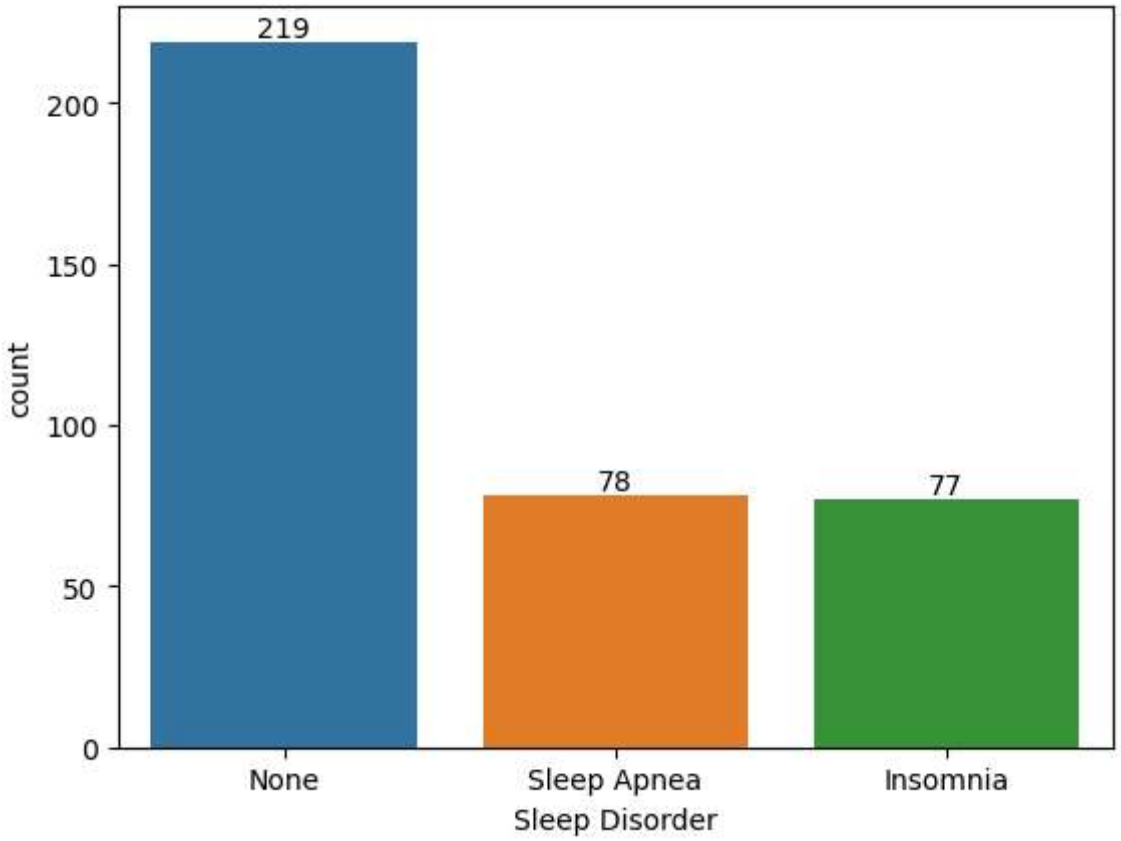
374 rows × 13 columns

In [9]:
```
df.describe()
#describe() method gives description of the dataframe
```

Out[9]:

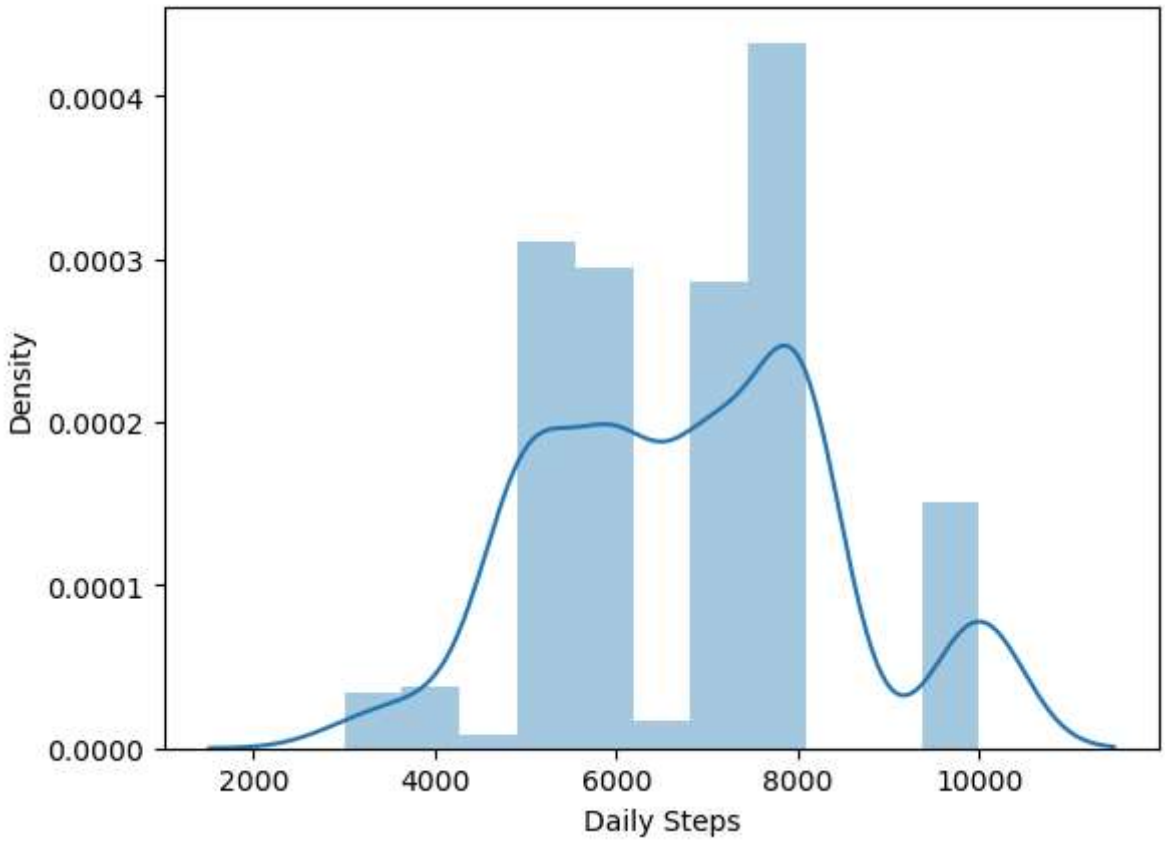
	Person ID	Age	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	Heart Rate	Da
count	374.000000	374.000000	374.000000	374.000000	374.000000	374.000000	374.000000	37
mean	187.500000	42.184492	7.132086	7.312834	59.171123	5.385027	70.165775	681
std	108.108742	8.673133	0.795657	1.196956	20.830804	1.774526	4.135676	161
min	1.000000	27.000000	5.800000	4.000000	30.000000	3.000000	65.000000	300
25%	94.250000	35.250000	6.400000	6.000000	45.000000	4.000000	68.000000	560
50%	187.500000	43.000000	7.200000	7.000000	60.000000	5.000000	70.000000	700
75%	280.750000	50.000000	7.800000	8.000000	75.000000	7.000000	72.000000	800
max	374.000000	59.000000	8.500000	9.000000	90.000000	8.000000	86.000000	1000

In [10]:
```
l=sns.countplot(x="Sleep Disorder",data=df)
for bars in l.containers:
    l.bar_label(bars)
#showing most of people are not having sleep disorder.
```



In [11]:
```
sns.distplot(df["Daily Steps"])
#It is right skewness.
#mean is greater than median.
```

Out[11]: <Axes: xlabel='Daily Steps', ylabel='Density'>



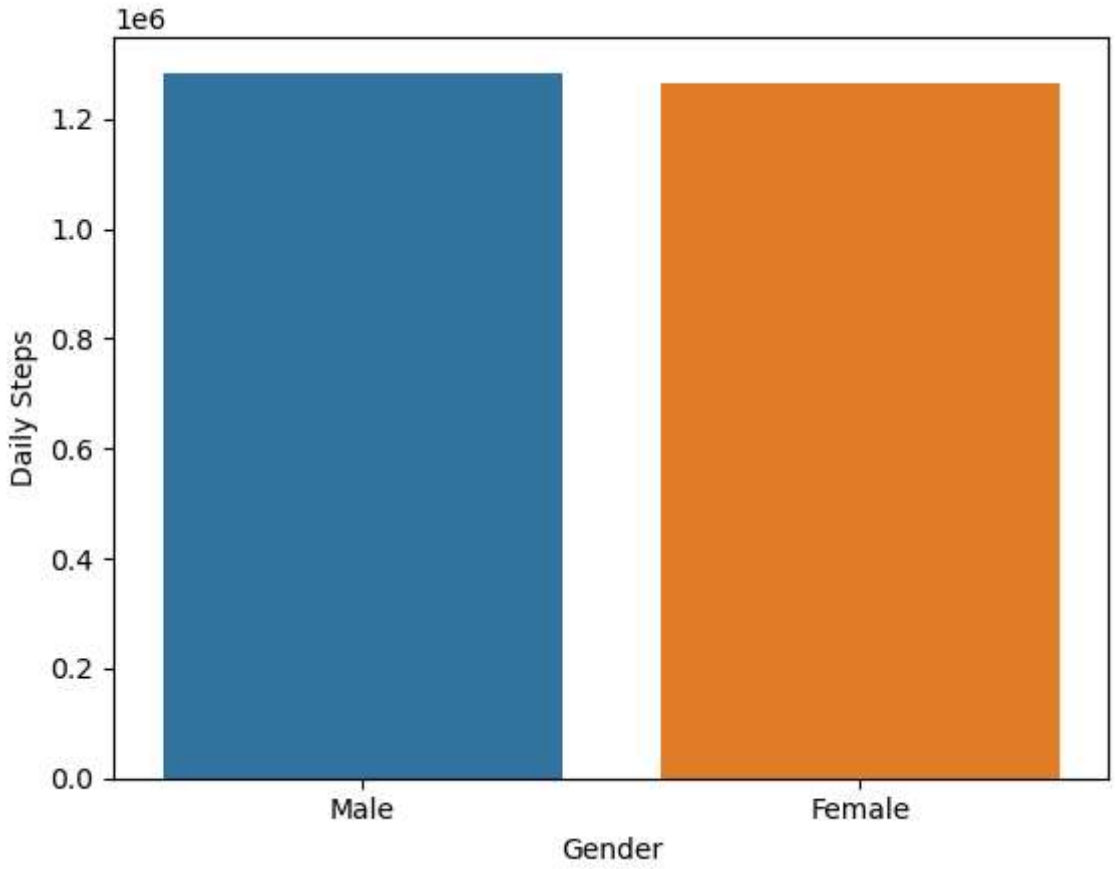
In [12]:
```
df.groupby(["Gender"], as_index=False)["Daily Steps"].sum().sort_values(by
#total steps spent by male and female.
```

Out[12]:

	Gender	Daily Steps
1	Male	1284000
0	Female	1265500

In [13]:
```
step=df.groupby(["Gender"], as_index=False)["Daily Steps"].sum().sort_valu
sns.barplot(x="Gender",y="Daily Steps",data=step)
#from the below graph we say that both take equal daily steps.
```

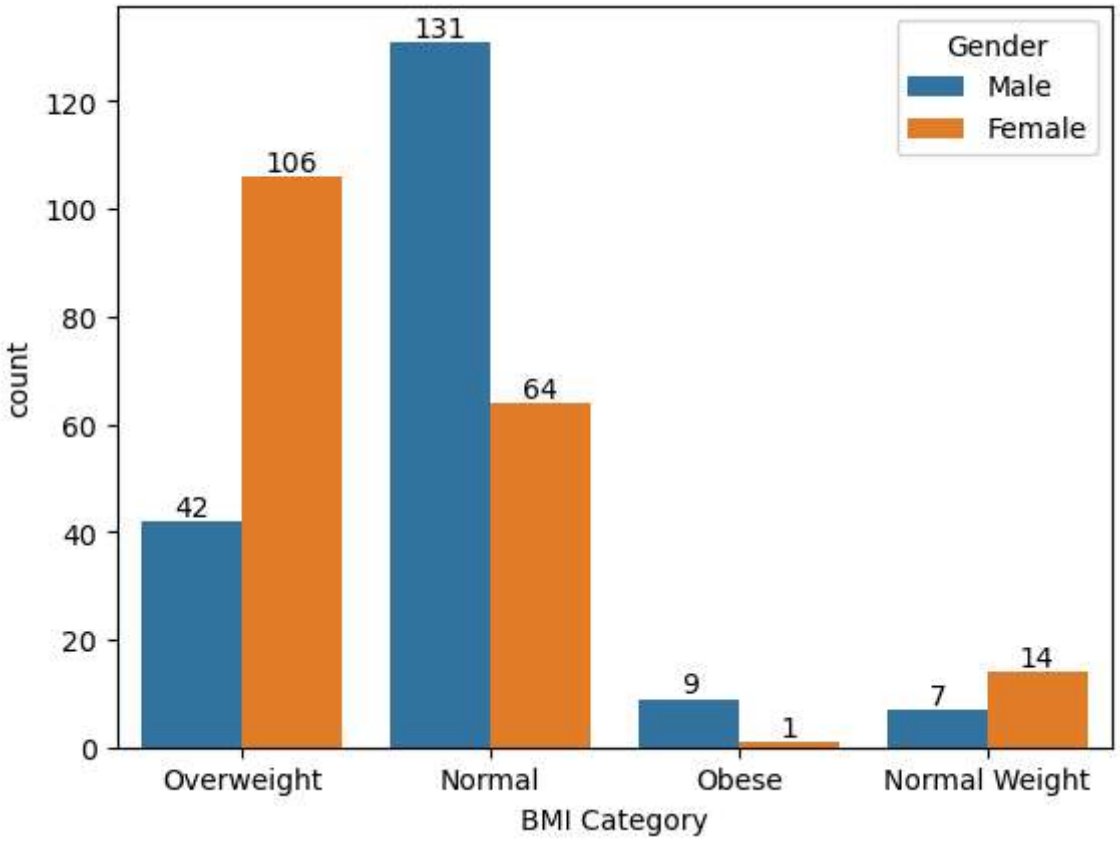
Out[13]: <Axes: xlabel='Gender', ylabel='Daily Steps'>



In [14]:
```
df["BMI Category"].value_counts()
#value counts of BMI Category
```

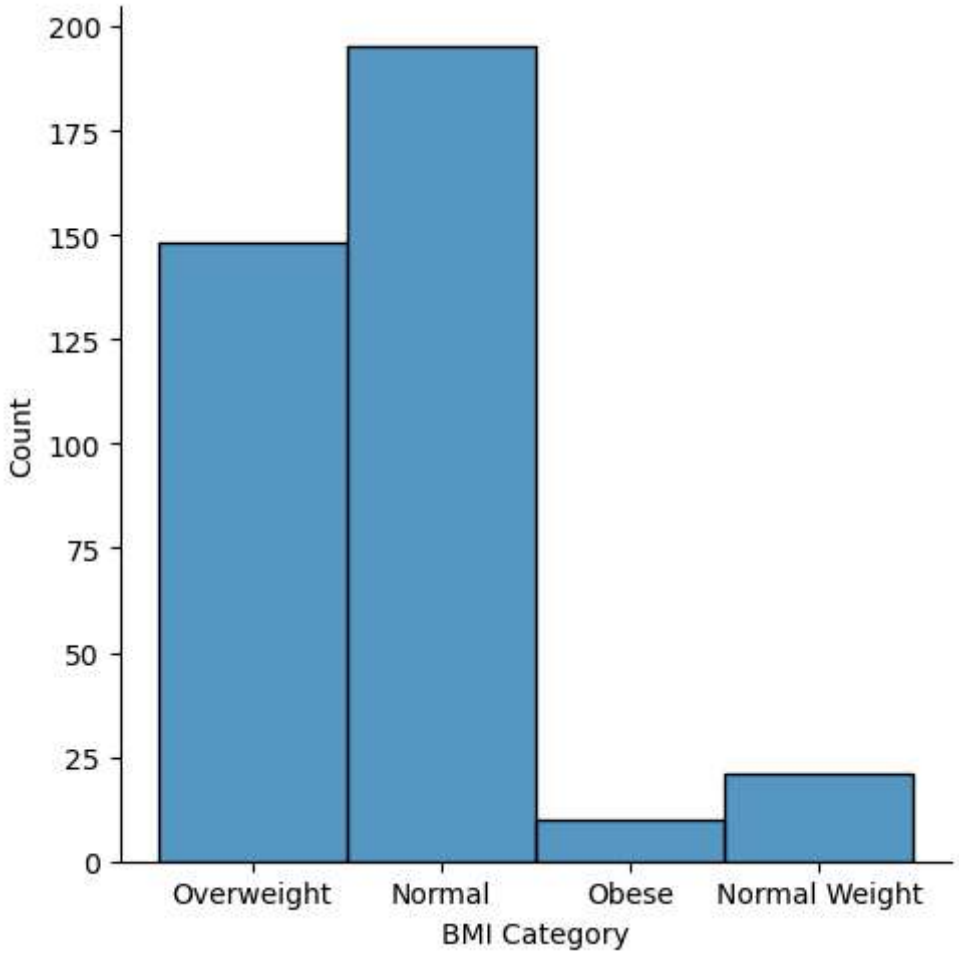
Out[14]:
```
Normal           195
Overweight       148
Normal Weight     21
Obese             10
Name: BMI Category, dtype: int64
```

In [15]:
```
a=sns.countplot(data=df,x="BMI Category",hue="Gender")
for bars in a.containers:
    a.bar_label(bars)
#from the below graph we can say that female are overweight and male are n
```



In [16]:
```
1 sns.displot(df["BMI Category"])
2 #Normal BMI Category count as more than other.
```

Out[16]: <seaborn.axisgrid.FacetGrid at 0x28a2c612050>



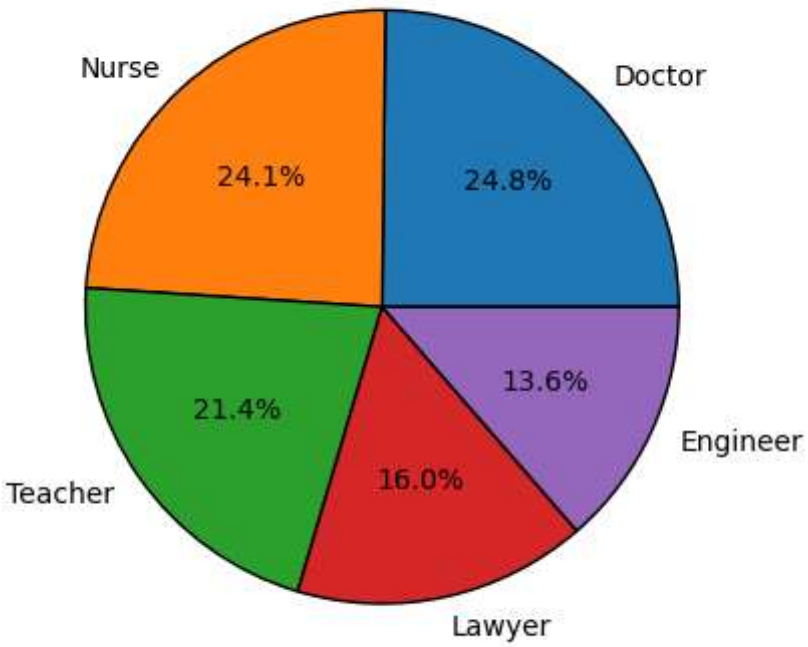
In [17]:
```
1 df["Occupation"].value_counts()
2 #value counts of Occupation
```

Out[17]:
```
Nurse                    73
Doctor                   71
Engineer                 63
Lawyer                   47
Teacher                  40
Accountant               37
Salesperson              32
Software Engineer         4
Scientist                 4
Sales Representative      2
Manager                   1
Name: Occupation, dtype: int64
```

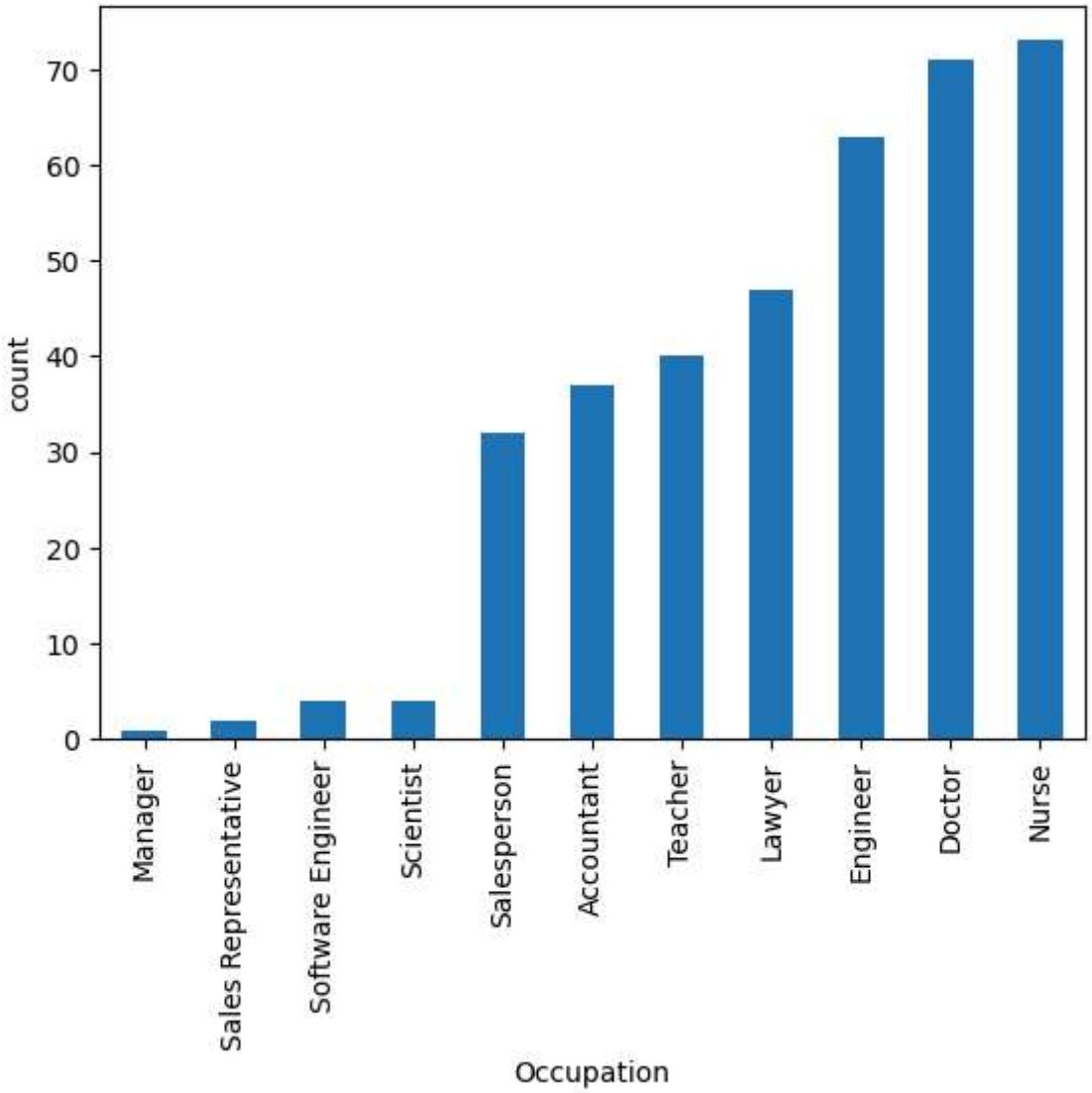
In [18]:
```
oc=df[df["Occupation"].isin(["Doctor","Nurse","Teacher","Lawyer","Engineer
occ_counts=oc["Occupation"].value_counts()
occ_counts

```

Out[18]:
```
Nurse       73
Doctor      71
Engineer    63
Lawyer      47
Teacher     40
Name: Occupation, dtype: int64
```

In [19]:
```
wedgeprops={
    "edgecolor":"k"
}
plt.pie(oc["Occupation"].value_counts(),labels=["Doctor","Nurse","Teacher"
       autopct="%1.1f%%",wedgeprops=wedgeprops)
plt.show()
#showing pie graph on some selectively Occupation.
```

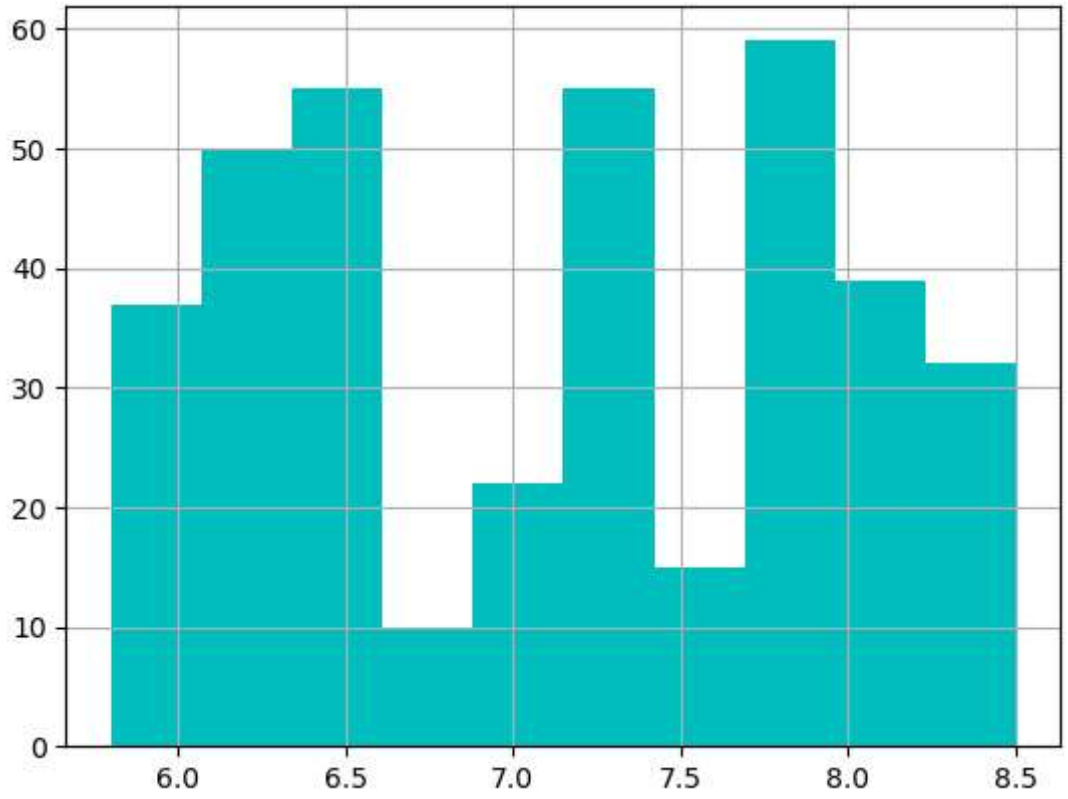


In [20]:
```
df["Occupation"].value_counts().sort_values(ascending=True).plot(kind="bar
plt.xlabel("Occupation")
plt.ylabel("count")
plt.show()
```

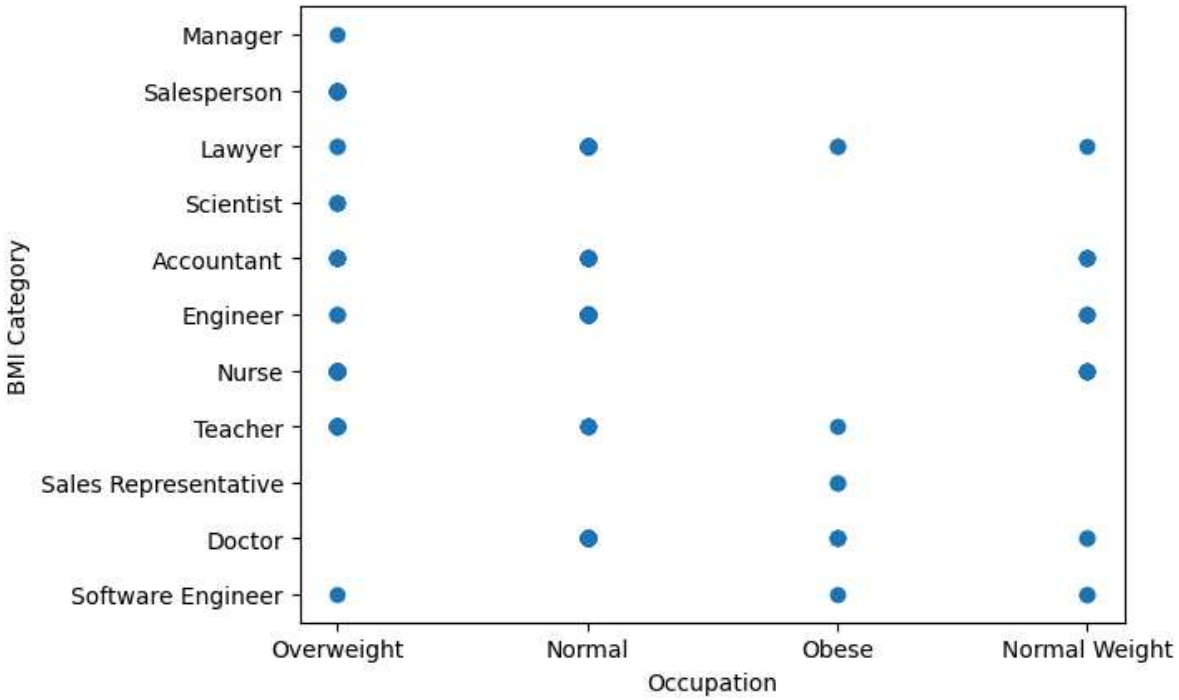


In [21]:
```
1 df["Sleep Duration"].hist(color="c")
```

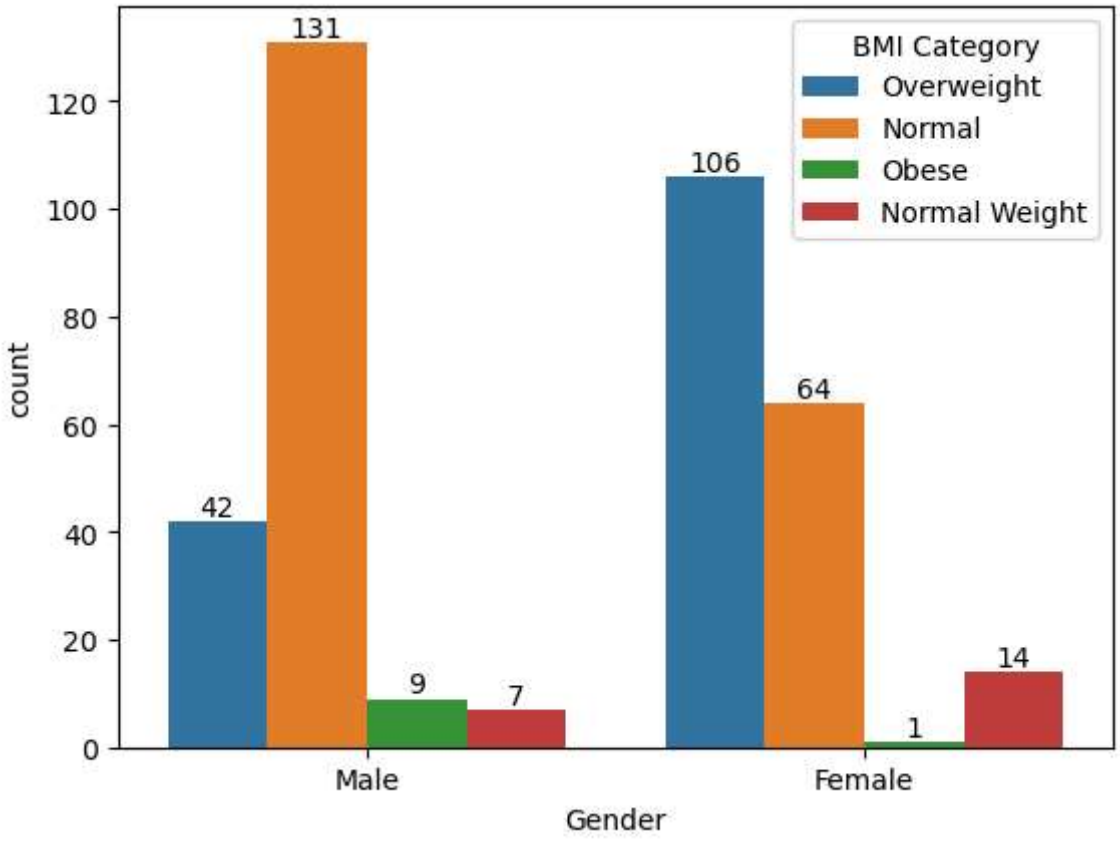
Out[21]: <Axes: >



In [22]:
```
1 plt.scatter(x=df["BMI Category"],y=df["Occupation"])
2 plt.xlabel("Occupation")
3 plt.ylabel("BMI Category")
4 plt.show()
```

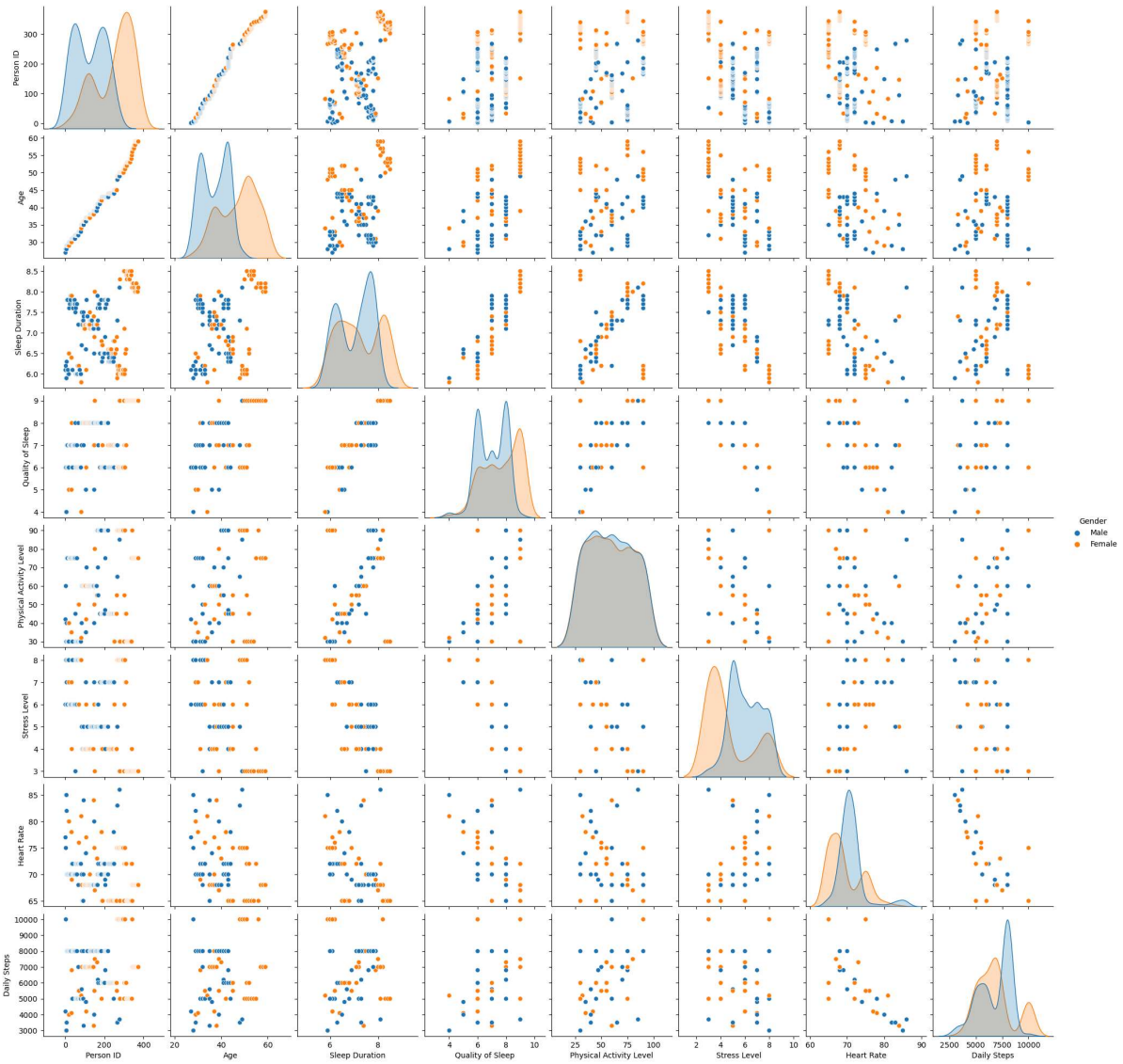


In [23]:
```
b=sns.countplot(data=df,x="Gender",hue="BMI Category")
for bars in b.containers:
    b.bar_label(bars)
```



In [25]:
```
1 sns.pairplot(df,hue="Gender")
```

Out[25]: <seaborn.axisgrid.PairGrid at 0x28a3941e770>



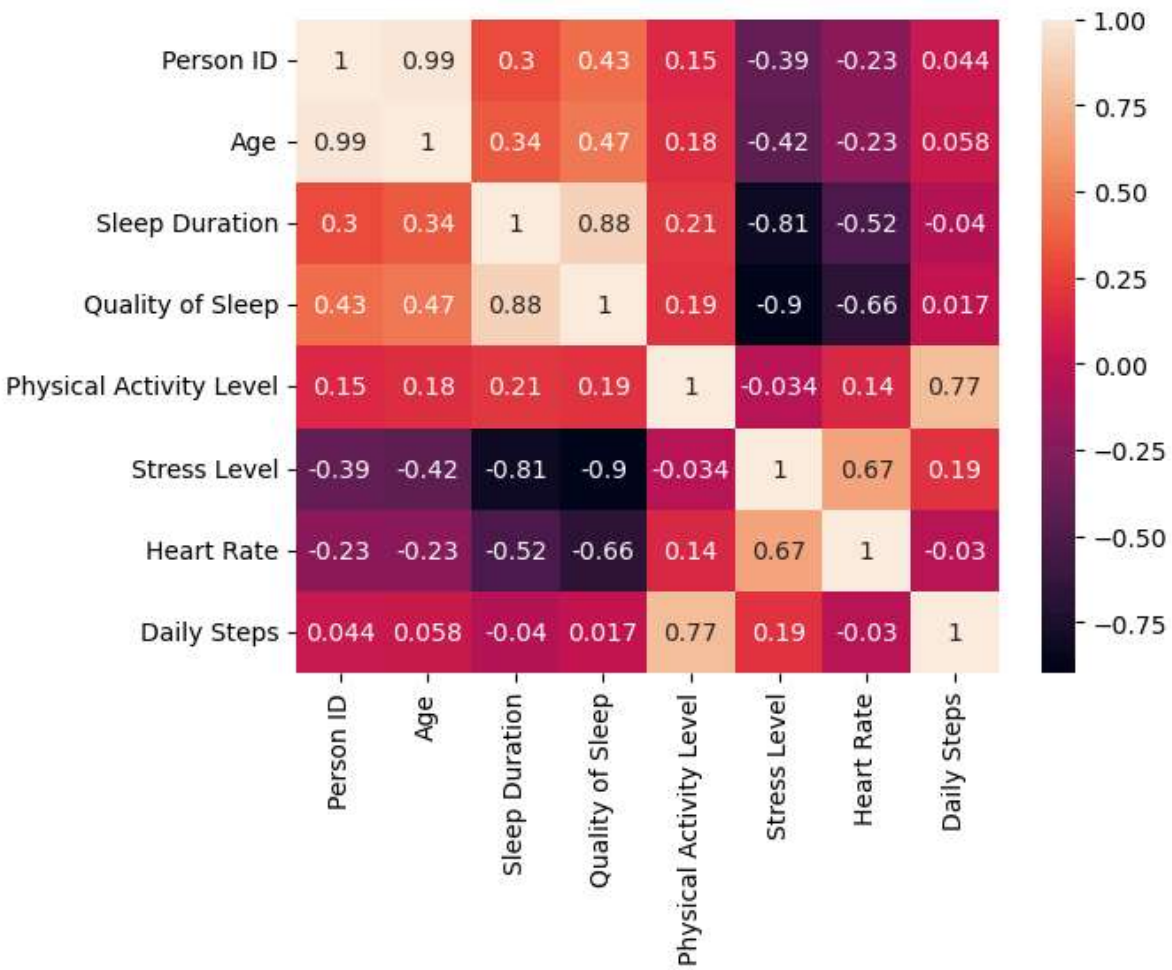
In [27]:
```
corr=df.corr()
corr
```

Out[27]:

	Person ID	Age	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	Heart Rate	Daily Steps
Person ID	1.000000	0.990516	0.296305	0.431612	0.149882	-0.394287	-0.225467	0.043844
Age	0.990516	1.000000	0.344709	0.473734	0.178993	-0.422344	-0.225606	0.057973
Sleep Duration	0.296305	0.344709	1.000000	0.883213	0.212360	-0.811023	-0.516455	-0.039533
Quality of Sleep	0.431612	0.473734	0.883213	1.000000	0.192896	-0.898752	-0.659865	0.016791
Physical Activity Level	0.149882	0.178993	0.212360	0.192896	1.000000	-0.034134	0.136971	0.772723
Stress Level	-0.394287	-0.422344	-0.811023	-0.898752	-0.034134	1.000000	0.670026	0.186829
Heart Rate	-0.225467	-0.225606	-0.516455	-0.659865	0.136971	0.670026	1.000000	-0.030309
Daily Steps	0.043844	0.057973	-0.039533	0.016791	0.772723	0.186829	-0.030309	1.000000

In [28]:
```
1 sns.heatmap(corr,annot=True)
```

Out[28]: <Axes: >



In [ ]:
```
1
```