# Project: Predict Sleep Health and Lifestyle

## Feature Descrition

- 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 sleep, ranging from 1 to 10.
- Physical Activity Level (minutes/day): The number of minutes the person engages in physical activity daily.
- Stress Level (scale: 1-10): A subjective rating of the stress level experienced by the person, ranging from 1 to 10.
- BMI Category: The BMI category of the person (e.g., Underweight, Normal, Overweight).
- Blood Pressure (systolic/diastolic): The blood pressure measurement of the person, indicated as systolic pressure over diastolic pressure.
- Heart Rate (bpm): The resting heart rate of the person in beats per minute.
- Daily Steps: The number of steps the person takes per day.
- Sleep Disorder: The presence or absence of a sleep disorder in the person (None, Insomnia, Sleep Apnea).

### Details about Sleep Disorder Column:

- None: The individual does not exhibit any specific sleep disorder.
- Insomnia: The individual experiences difficulty falling asleep or staying asleep, leading to inadequate or poor-quality sleep.
- Sleep Apnea: The individual suffers from pauses in breathing during sleep, resulting in disrupted sleep patterns and potential health risks.

## Import Libary

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from statistics import mean
sns.set_theme(color_codes=True)

from sklearn.model_selection import train_test_split, KFold
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import HistGradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from imblearn.over_sampling import SMOTE


from imblearn.over_sampling import RandomOverSampler

import warnings
warnings.filterwarnings("ignore")
```

## Function

In [3]:

```
def sexo(texto):
    if texto == 'Male':
        return(0)
    else:
        return(1)

def BMI_Cat(texto):
    if texto == 'Normal':
        return(0)
    elif texto == 'Overweight':
        return(1)
    elif texto == 'Normal Weight':
        return(2)
    else:
        return(3)
```

In [4]:

```
def target(texto):
    if texto == 'None':
        return(0)
    elif texto == 'Sleep Apnea':
        return(1)
    else:
        return(2)
```

## Import Dataset

In [5]:

```
df_health = pd.read_csv('Sleep_health_and_lifestyle_dataset.csv')
```

In [6]:

```
df_health.head()
```

Out[6]:

| | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | Male | 27 | Software Engineer | 6.1 | 6 | 42 | 6 | Overweight | 126/83 | 77 | 4200 | None |
| **1** | 2 | Male | 28 | Doctor | 6.2 | 6 | 60 | 8 | Normal | 125/80 | 75 | 10000 | None |
| **2** | 3 | Male | 28 | Doctor | 6.2 | 6 | 60 | 8 | Normal | 125/80 | 75 | 10000 | None |
| **3** | 4 | Male | 28 | Sales Representative | 5.9 | 4 | 30 | 8 | Obese | 140/90 | 85 | 3000 | Sleep Apnea |
| **4** | 5 | Male | 28 | Sales Representative | 5.9 | 4 | 30 | 8 | Obese | 140/90 | 85 | 3000 | Sleep Apnea |

## Data Preprocessing Part 1

In [7]:

```
# Tamanho do dataset
df_health.shape
```

Out[7]:

```
(374, 13)
```

In [8]:

```
# Conferindo os tipos dos dados
df_health.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 [9]:

```
# Validando a quantidade de valores unicos nas colunas categoricas
```

```
df_health.select_dtypes(include='object').nunique()
```

Out[9]:
```
Gender              2
Occupation         11
BMI Category        4
Blood Pressure     25
Sleep Disorder      3
dtype: int64
```

In [10]:
```
# Observando os valores categoricos
df_health['Occupation'].unique()
```

Out[10]:
```
array(['Software Engineer', 'Doctor', 'Sales Representative', 'Teacher',
       'Nurse', 'Engineer', 'Accountant', 'Scientist', 'Lawyer',
       'Salesperson', 'Manager'], dtype=object)
```

In [11]:
```
df_health['Blood Pressure'].unique()
```

Out[11]:
```
array(['126/83', '125/80', '140/90', '120/80', '132/87', '130/86',
       '117/76', '118/76', '128/85', '131/86', '128/84', '115/75',
       '135/88', '129/84', '130/85', '115/78', '119/77', '121/79',
       '125/82', '135/90', '122/80', '142/92', '140/95', '139/91',
       '118/75'], dtype=object)
```

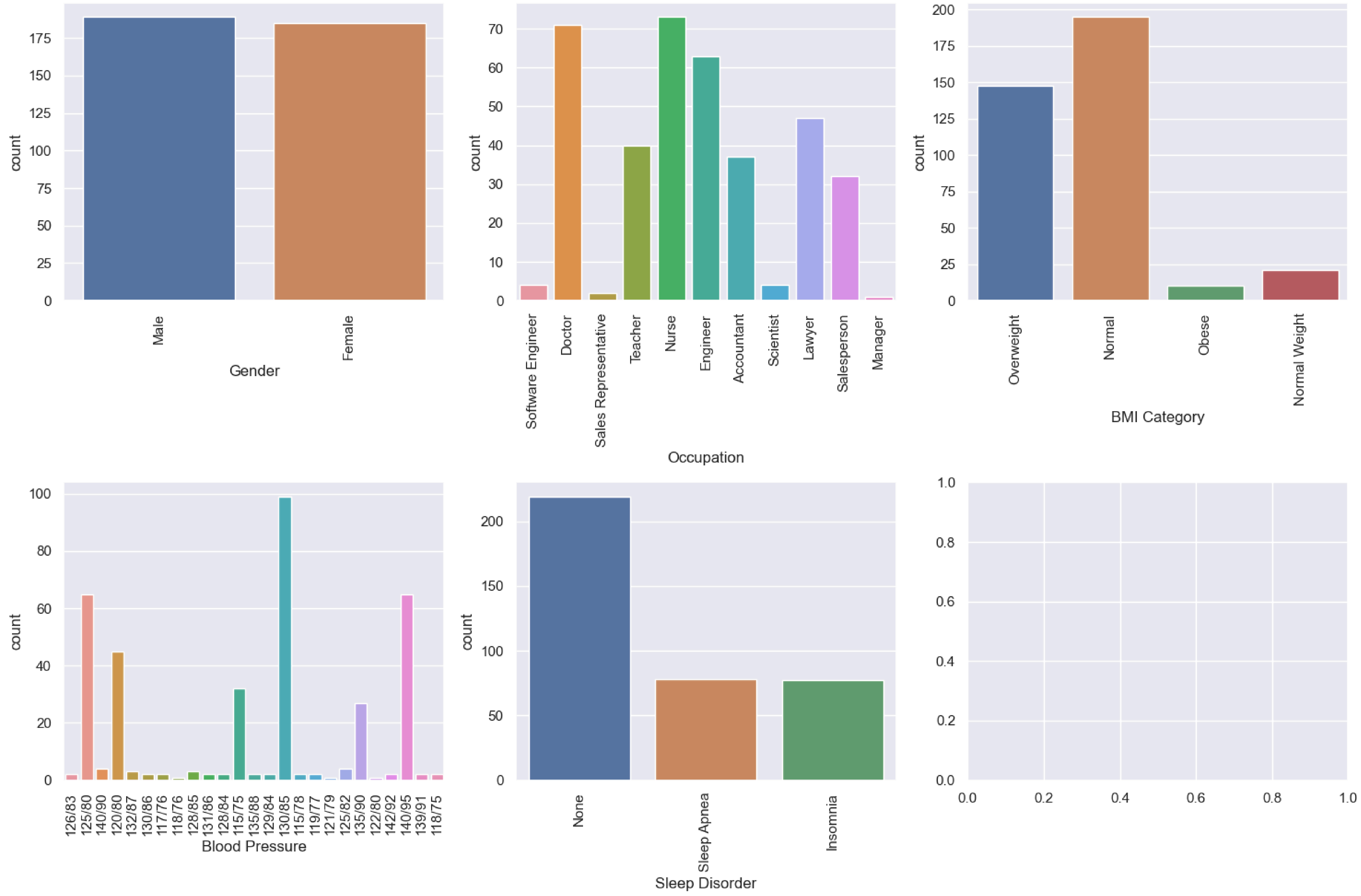
## Exploratory Data Analysis (EDA)

In [12]:
```
# Resumo estatistico
df_health.describe()
```

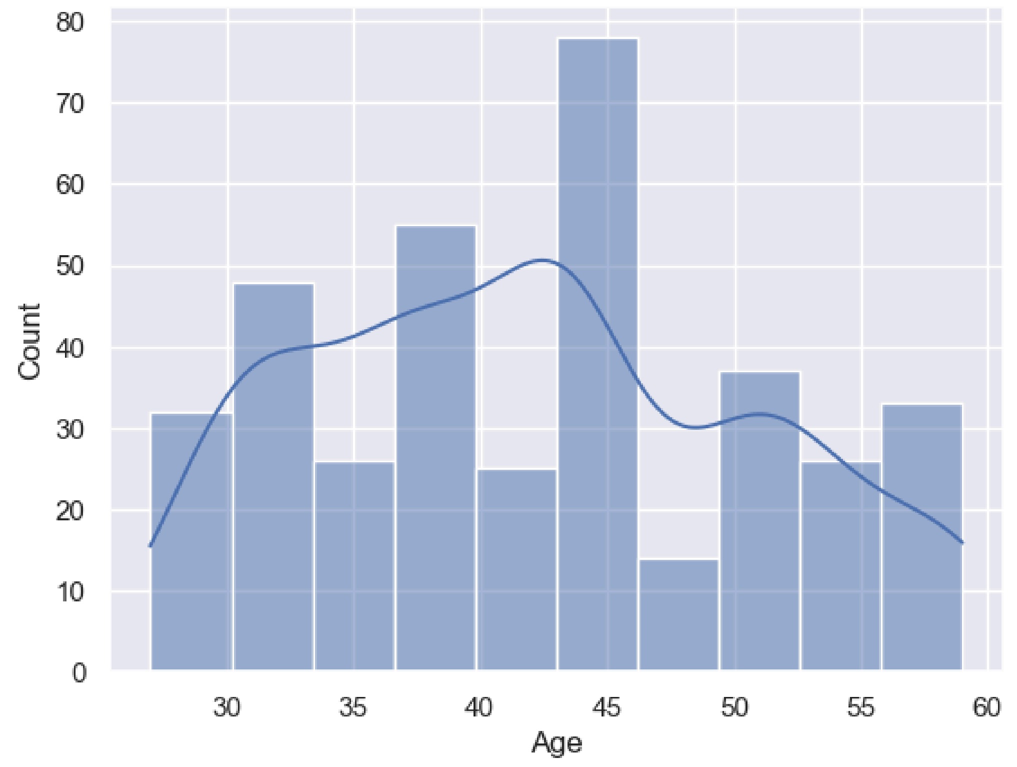
Out[12]:

| | Person ID | Age | Sleep Duration | Quality of Sleep | Physical Activity Level | Stress Level | Heart Rate | Daily Steps |
|---|---|---|---|---|---|---|---|---|
| **count** | 374.000000 | 374.000000 | 374.000000 | 374.000000 | 374.000000 | 374.000000 | 374.000000 | 374.000000 |
| **mean** | 187.500000 | 42.184492 | 7.132086 | 7.312834 | 59.171123 | 5.385027 | 70.165775 | 6816.844920 |
| **std** | 108.108742 | 8.673133 | 0.795657 | 1.196956 | 20.830804 | 1.774526 | 4.135676 | 1617.915679 |
| **min** | 1.000000 | 27.000000 | 5.800000 | 4.000000 | 30.000000 | 3.000000 | 65.000000 | 3000.000000 |
| **25%** | 94.250000 | 35.250000 | 6.400000 | 6.000000 | 45.000000 | 4.000000 | 68.000000 | 5600.000000 |
| **50%** | 187.500000 | 43.000000 | 7.200000 | 7.000000 | 60.000000 | 5.000000 | 70.000000 | 7000.000000 |
| **75%** | 280.750000 | 50.000000 | 7.800000 | 8.000000 | 75.000000 | 7.000000 | 72.000000 | 8000.000000 |
| **max** | 374.000000 | 59.000000 | 8.500000 | 9.000000 | 90.000000 | 8.000000 | 86.000000 | 10000.000000 |

In [13]:
```
# listar as categorias para plotar
col_categ = ['Gender', 'Occupation', 'BMI Category', 'Blood Pressure', 'Sleep Disorder']

fig, ax = plt.subplots(nrows=2, ncols=3, figsize=(15,10))
axs = ax.flatten()

for i, var in enumerate(col_categ):
    sns.countplot(x=var, data=df_health, ax=axs[i])
    axs[i].set_xticklabels(axs[i].get_xticklabels(), rotation=90)

fig.tight_layout()

plt.show()
```



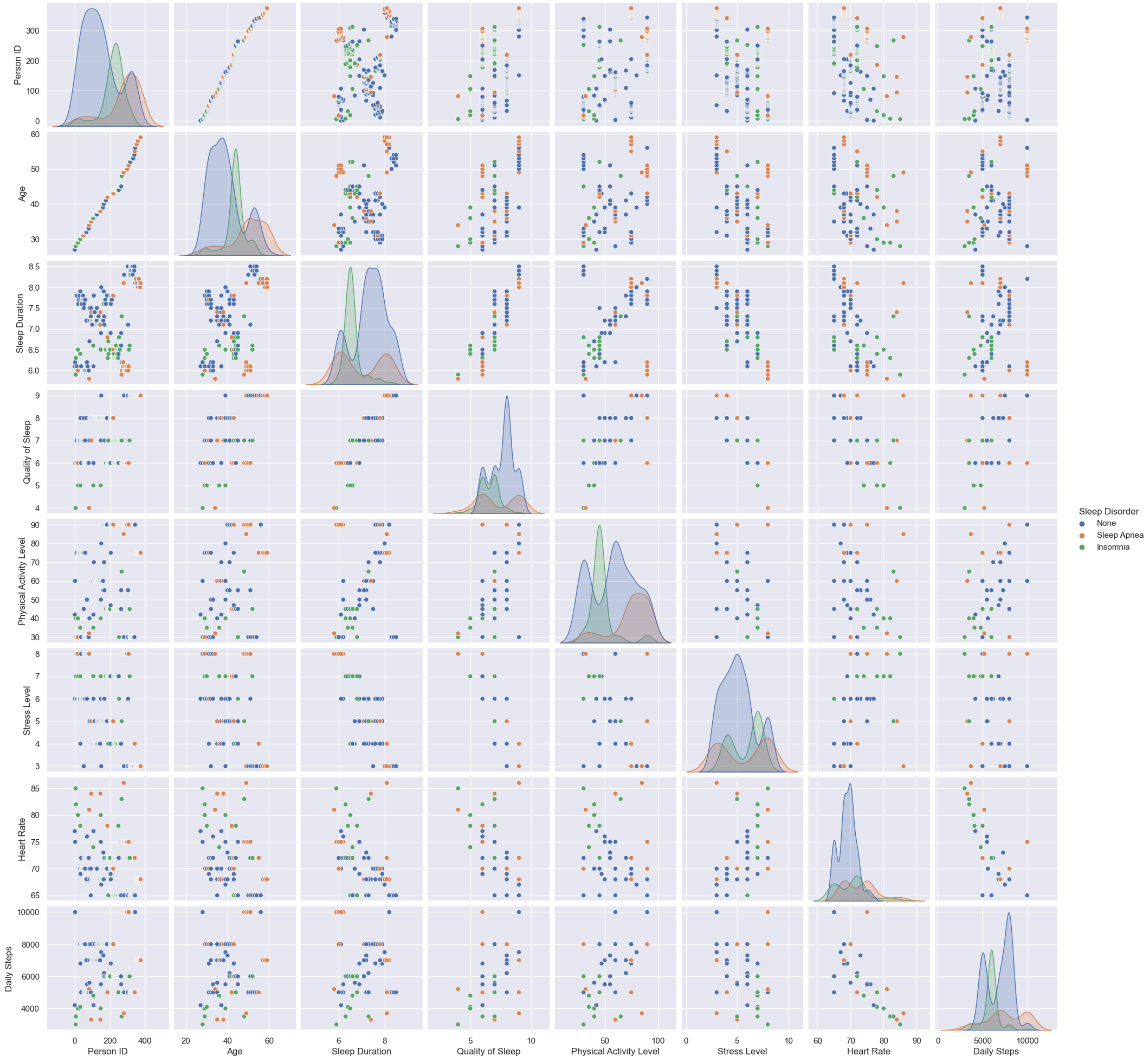
In [15]:
```
sns.histplot(data=df_health, x='Age', kde=True);
```



In [16]:

```
fig = plt.figure(figsize=(20,12))
sns.pairplot(data=df_health, hue='Sleep Disorder');
```

```
<Figure size 2000x1200 with 0 Axes>
```



## Data Preprocessing Part 2

In [17]:

```
df_health.head()
```

Out[17]:

|   | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | Male | 27 | Software Engineer | 6.1 | 6 | 42 | 6 | Overweight | 126/83 | 77 | 4200 | None |
| **1** | 2 | Male | 28 | Doctor | 6.2 | 6 | 60 | 8 | Normal | 125/80 | 75 | 10000 | None |
| **2** | 3 | Male | 28 | Doctor | 6.2 | 6 | 60 | 8 | Normal | 125/80 | 75 | 10000 | None |
| **3** | 4 | Male | 28 | Sales Representative | 5.9 | 4 | 30 | 8 | Obese | 140/90 | 85 | 3000 | Sleep Apnea |
| **4** | 5 | Male | 28 | Sales Representative | 5.9 | 4 | 30 | 8 | Obese | 140/90 | 85 | 3000 | Sleep Apnea |

In [18]:

```
df_health['Blood Pressure'] = df_health['Blood Pressure'].str.replace('/', '.')
```

In [19]:

```
## Alterar as categorias para numeros

df_health['Gender'] = df_health['Gender'].apply(sexo)
df_health['BMI Category'] = df_health['BMI Category'].apply(BMI_Cat)
df_health['Sleep Disorder'] = df_health['Sleep Disorder'].apply(target)
```

In [20]:

```
dummy_features = pd.get_dummies(df_health['Occupation'], drop_first=True)
```

In [21]:

```
df_final = pd.concat([df_health, dummy_features], axis=1)
```

In [22]:

```
df_final = df_final.drop('Occupation', axis=1)
```

In [23]:

```
df_final.shape
```

Out[23]:

```
(374, 22)
```

## Split Train / Test

In [24]:

```
X = df_final.drop('Sleep Disorder', axis=1)
y = df_final['Sleep Disorder']
```

In [25]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [26]:

```
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

Out[26]:

```
((261, 21), (261,), (113, 21), (113,))
```

## Classifier Models

In [27]:

```
# Logistic Regression
model_lr = LogisticRegression()
model_lr.fit(X_train, y_train)
```

Out[27]:

```
▾ LogisticRegression
LogisticRegression()
```

In [28]:

```
predict_lr = model_lr.predict(X_test)
```

In [29]:

```
print(classification_report(y_test,predict_lr))
```

```
              precision    recall  f1-score   support

           0       0.66      0.82      0.73        62
           1       1.00      0.63      0.77        27
           2       0.37      0.29      0.33        24

    accuracy                           0.66       113
   macro avg       0.68      0.58      0.61       113
weighted avg       0.68      0.66      0.66       113
```

In [30]:

```
print(confusion_matrix(y_test, predict_lr))
```

```
[[51  0 11]
 [ 9 17  1]
 [17  0  7]]
```

In [31]:

```
# Decision Tree Classifier
model_dtc = DecisionTreeClassifier()
model_dtc.fit(X_train, y_train)
```

Out[31]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [32]:

```
predict_dtc = model_dtc.predict(X_test)
```

In [33]:

```
print(classification_report(y_test, predict_dtc))
```

```
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        62
           1       0.95      0.74      0.83        27
           2       0.81      0.88      0.84        24

    accuracy                           0.91       113
   macro avg       0.90      0.87      0.88       113
weighted avg       0.91      0.91      0.91       113
```

In [146…

```
print(confusion_matrix(y_test, predict_dtc))
```

```
[[62  0  0]
 [ 2 20  5]
 [ 2  1 21]]
```

In [34]:

```
# Random Forest Classifier
model_rfc = RandomForestClassifier()
model_rfc.fit(X_train, y_train)
```

Out[34]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [35]:

```
predict_rfc = model_rfc.predict(X_test)
```

In [36]:

```
print(classification_report(y_test, predict_rfc))
```

```
              precision    recall  f1-score   support

           0       0.95      0.98      0.97        62
           1       0.91      0.74      0.82        27
           2       0.78      0.88      0.82        24

    accuracy                           0.90       113
   macro avg       0.88      0.87      0.87       113
weighted avg       0.91      0.90      0.90       113
```

In [37]:

```
print(confusion_matrix(y_test, predict_dtc))
```

```
[[62  0  0]
 [ 2 20  5]
 [ 2  1 21]]
```

In [38]:

```
# Gradient Boosting Classifier
model_hgb = HistGradientBoostingClassifier()
model_hgb.fit(X_train, y_train)
```

Out[38]:

```
▾ HistGradientBoostingClassifier
HistGradientBoostingClassifier()
```

In [39]:

```
predict_hgb = model_hgb.predict(X_test)
```

In [40]:

```
print(classification_report(y_test, predict_hgb))
```

```
              precision    recall  f1-score   support

           0       0.95      1.00      0.98        62
           1       0.91      0.74      0.82        27
           2       0.81      0.88      0.84        24

    accuracy                           0.91       113
   macro avg       0.89      0.87      0.88       113
weighted avg       0.91      0.91      0.91       113
```

# Comparação com a coluna Blood Pressure alterada

## Logistc Regression

precision recall f1-score support

```
           0       0.75      0.81      0.78        62
           1       0.95      0.74      0.83        27
           2       0.48      0.50      0.49        24

    accuracy                           0.73       113
```

macro avg 0.73 0.68 0.70 113 weighted avg 0.74 0.73 0.73 113

precision recall f1-score support

```
           0       0.66      0.82      0.73        62
           1       1.00      0.63      0.77        27
           2       0.37      0.29      0.33        24

    accuracy                           0.66       113
```

macro avg 0.68 0.58 0.61 113 weighted avg 0.68 0.66 0.66 113

# Decision Tree

precision recall f1-score support

```
           0       0.91      0.98      0.95        62
           1       0.91      0.74      0.82        27
           2       0.88      0.88      0.88        24

    accuracy                           0.90       113
```

macro avg 0.90 0.87 0.88 113 weighted avg 0.90 0.90 0.90 113

precision recall f1-score support

```
           0       0.94      1.00      0.97        62
           1       0.95      0.74      0.83        27
           2       0.81      0.88      0.84        24

    accuracy                           0.91       113
```

macro avg 0.90 0.87 0.88 113 weighted avg 0.91 0.91 0.91 113

# Random Forest

precision recall f1-score support

```
           0       0.93      1.00      0.96        62
           1       0.91      0.74      0.82        27
           2       0.88      0.88      0.88        24

    accuracy                           0.91       113
```

macro avg 0.90 0.87 0.88 113 weighted avg 0.91 0.91 0.91 113

precision recall f1-score support

```
           0       0.92      0.98      0.95        62
           1       0.87      0.74      0.80        27
           2       0.88      0.88      0.88        24

    accuracy                           0.90       113
```

macro avg 0.89 0.87 0.88 113 weighted avg 0.90 0.90 0.90 113

# Gradient Boosting Classifier

precision recall f1-score support

```
           0       0.95      1.00      0.98        62
           1       0.91      0.74      0.82        27
           2       0.81      0.88      0.84        24

    accuracy                           0.91       113
```

macro avg 0.89 0.87 0.88 113 weighted avg 0.91 0.91 0.91 113

```
In [ ]:
```