Importing the Needed Libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix, f1_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
```

Loading the Data and Performing Data Cleaning

In [2]:
```
df = pd.read_csv("C://Users//quays//Desktop//mental_health_dataset.csv")
```

In [3]:
```
# displaying the top 5 rows of the data
df.head()
```

Out[3]:

	age	gender	employment_status	work_environment	mental_health_history	seeks_treatment	stress_level	sleep_hours	physical_activity_days	dep
0	56	Male	Employed	On-site	Yes	Yes	6	6.2	wednesday	
1	46	Female	Student	On-site	No	Yes	10	9.0	thursday	
2	32	Female	Employed	On-site	Yes	No	7	7.7	tuesday	
3	60	Non-binary	Self-employed	On-site	No	No	4	4.5	thursday	
4	25	Female	Self-employed	On-site	Yes	Yes	3	5.4	0	

In [4]:
```
# displaying the last five rows of the data
df.tail()
```

Out[4]:

	age	gender	employment_status	work_environment	mental_health_history	seeks_treatment	stress_level	sleep_hours	physical_activity_days
9995	34	Female	Employed	On-site	Yes	Yes	5	6.1	wednesday
9996	47	Male	Employed	On-site	Yes	No	1	5.7	wednesday
9997	56	Female	Employed	On-site	Yes	No	1	8.3	0
9998	24	Male	Employed	On-site	Yes	Yes	9	6.1	0
9999	44	Male	Unemployed	Remote	No	Yes	5	6.4	sunday

In [5]:
```
# shape of the data
df.shape
```

Out[5]:
```
(10000, 14)
```

In [6]:
```
# checking for the missing values in the data
df.isnull().sum()
```

Out[6]:
```
age                       0
gender                    0
employment_status         0
work_environment          0
mental_health_history     0
seeks_treatment           0
stress_level              0
sleep_hours               0
physical_activity_days    0
depression_score          0
anxiety_score             0
social_support_score      0
productivity_score        0
mental_health_risk        0
dtype: int64
```

In [7]:
```
# checking the columns of the data
df.columns.tolist()
```

Out[7]:
```
['age',
 'gender',
 'employment_status',
 'work_environment',
 'mental_health_history',
 'seeks_treatment',
 'stress_level',
 'sleep_hours',
 'physical_activity_days',
 'depression_score',
 'anxiety_score',
 'social_support_score',
 'productivity_score',
 'mental_health_risk']
```

In [8]:
```
# checking for duplicated columns
df.duplicated().sum()
```

Out[8]:
```
0
```

In [9]:
```
# checking for data types of each column
df.dtypes
```

Out[9]:
```
age                        int64
gender                    object
employment_status         object
work_environment          object
mental_health_history     object
seeks_treatment           object
stress_level               int64
sleep_hours              float64
physical_activity_days    object
depression_score           int64
anxiety_score              int64
social_support_score       int64
productivity_score       float64
mental_health_risk        object
dtype: object
```

In [10]:
```
df
```

Out[10]:

	age	gender	employment_status	work_environment	mental_health_history	seeks_treatment	stress_level	sleep_hours	physical_activity_days
0	56	Male	Employed	On-site	Yes	Yes	6	6.2	wednesday
1	46	Female	Student	On-site	No	Yes	10	9.0	thursday
2	32	Female	Employed	On-site	Yes	No	7	7.7	tuesday
3	60	Non-binary	Self-employed	On-site	No	No	4	4.5	thursday
4	25	Female	Self-employed	On-site	Yes	Yes	3	5.4	0
...	...	...	...	...	...	...	...	...	...
9995	34	Female	Employed	On-site	Yes	Yes	5	6.1	wednesday
9996	47	Male	Employed	On-site	Yes	No	1	5.7	wednesday
9997	56	Female	Employed	On-site	Yes	No	1	8.3	0
9998	24	Male	Employed	On-site	Yes	Yes	9	6.1	0
9999	44	Male	Unemployed	Remote	No	Yes	5	6.4	sunday

10000 rows × 14 columns

In [11]:
```
# extracting data with physical activity day not equal 0
df = df[df['physical_activity_days'] != '0']
```

In [12]:
```
# reviewing the data
df.head()
```

Out[12]:

	age	gender	employment_status	work_environment	mental_health_history	seeks_treatment	stress_level	sleep_hours	physical_activity_days	dep
0	56	Male	Employed	On-site	Yes	Yes	6	6.2	wednesday	
1	46	Female	Student	On-site	No	Yes	10	9.0	thursday	
2	32	Female	Employed	On-site	Yes	No	7	7.7	tuesday	
3	60	Non-binary	Self-employed	On-site	No	No	4	4.5	thursday	
5	38	Female	Unemployed	On-site	Yes	Yes	3	9.9	wednesday	

In [13]:
```
df['physical_activity_days'].unique()
```

Out[13]:
```
array(['wednesday', 'thursday', 'tuesday', 'monday', 'friday', 'saturday',
       'sunday'], dtype=object)
```

In [14]:
```
# checking for the unique values of each column
df.nunique()
```

Out[14]:
```
age                        48
gender                      4
employment_status           4
work_environment            3
mental_health_history       2
seeks_treatment             2
stress_level               10
sleep_hours                71
physical_activity_days      7
depression_score           31
anxiety_score              22
social_support_score      101
productivity_score        543
mental_health_risk          3
dtype: int64
```

In [15]:
```
# making all the object column values lower
object_cols = df.select_dtypes(include='object').columns

df[object_cols] = df[object_cols].applymap(lambda x: x.lower() if isinstance(x, str) else x)
```

In [16]:
```
df.head()
```

Out[16]:

	age	gender	employment_status	work_environment	mental_health_history	seeks_treatment	stress_level	sleep_hours	physical_activity_days	dep
0	56	male	employed	on-site	yes	yes	6	6.2	wednesday	
1	46	female	student	on-site	no	yes	10	9.0	thursday	
2	32	female	employed	on-site	yes	no	7	7.7	tuesday	
3	60	non-binary	self-employed	on-site	no	no	4	4.5	thursday	
5	38	female	unemployed	on-site	yes	yes	3	9.9	wednesday	

In [17]:
```
# data info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8788 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   age                     8788 non-null   int64  
 1   gender                  8788 non-null   object 
 2   employment_status       8788 non-null   object 
 3   work_environment        8788 non-null   object 
 4   mental_health_history   8788 non-null   object 
 5   seeks_treatment         8788 non-null   object 
 6   stress_level            8788 non-null   int64  
 7   sleep_hours             8788 non-null   float64
 8   physical_activity_days  8788 non-null   object 
 9   depression_score        8788 non-null   int64  
 10  anxiety_score           8788 non-null   int64  
 11  social_support_score    8788 non-null   int64  
 12  productivity_score      8788 non-null   float64
 13  mental_health_risk      8788 non-null   object 
dtypes: float64(2), int64(5), object(7)
memory usage: 1.0+ MB
```

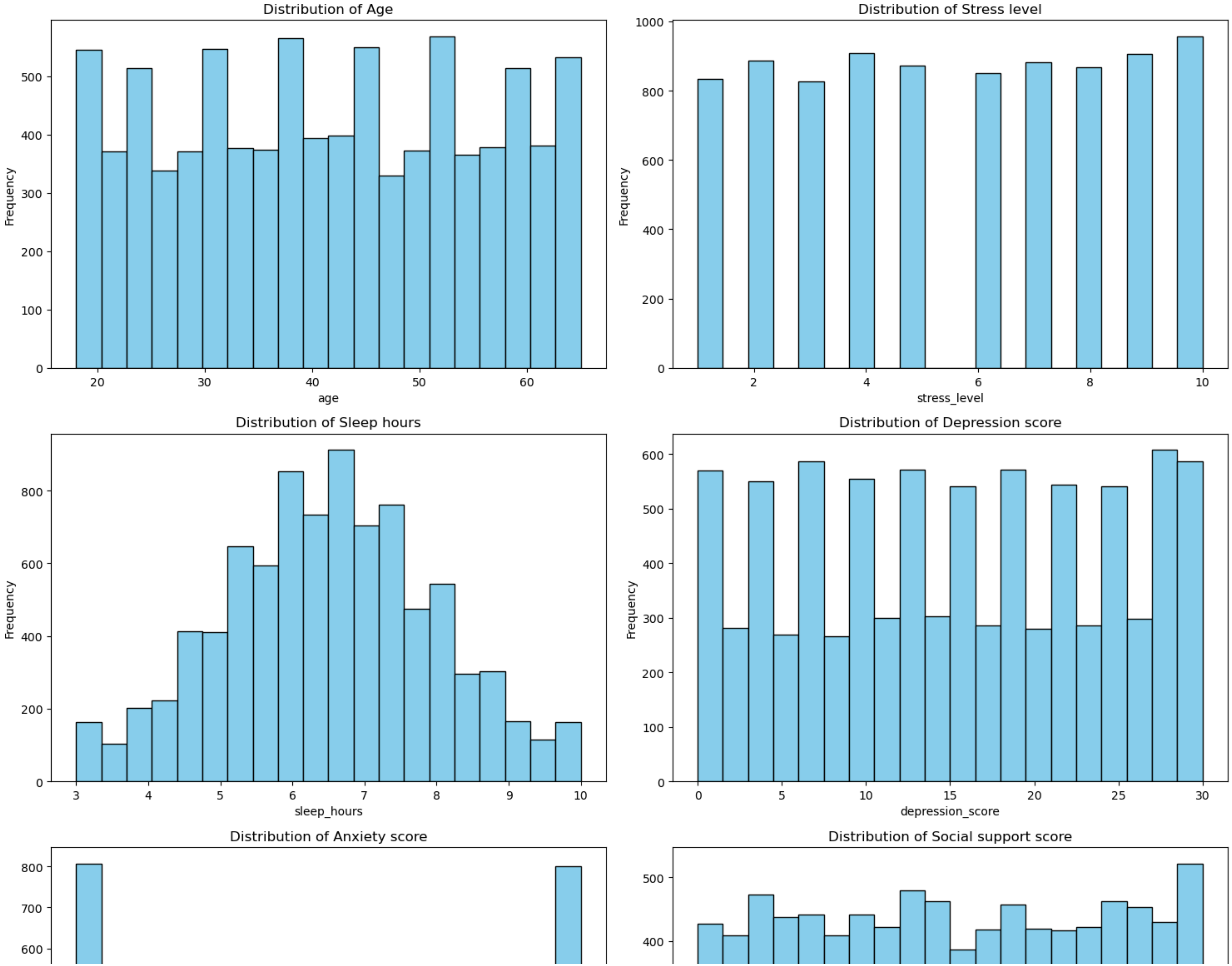
In [18]:
```
# data description
df.describe(include = 'all')
```

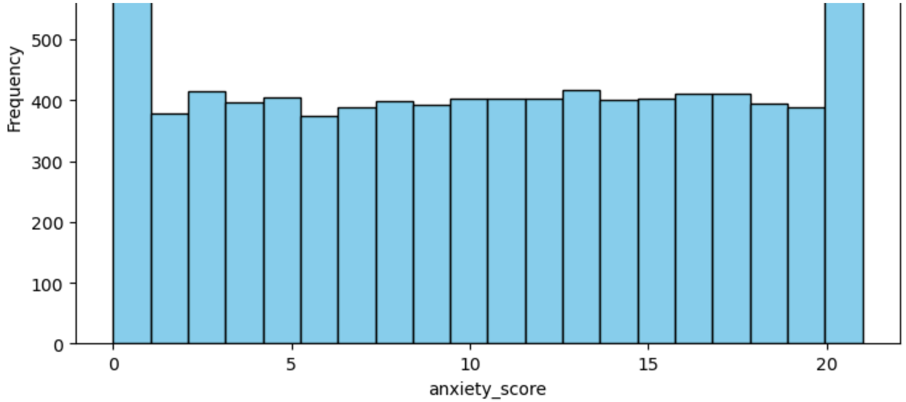
Out[18]:

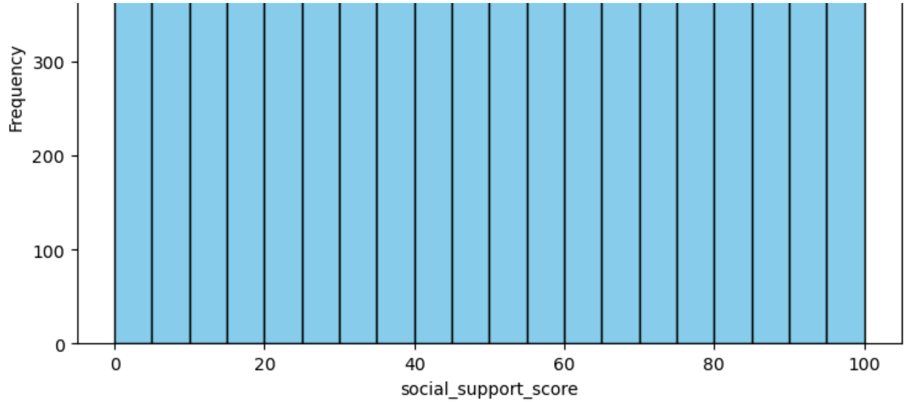
	age	gender	employment_status	work_environment	mental_health_history	seeks_treatment	stress_level	sleep_hours	physical_activ
count	8788.000000	8788	8788	8788	8788	8788	8788.000000	8788.000000	
unique	NaN	4	4	3	2	2	NaN	NaN	
top	NaN	male	employed	on-site	no	no	NaN	NaN	
freq	NaN	4000	5156	4469	6114	5275	NaN	NaN	
mean	41.536072	NaN	NaN	NaN	NaN	NaN	5.577378	6.471541	
std	13.755211	NaN	NaN	NaN	NaN	NaN	2.885911	1.477897	
min	18.000000	NaN	NaN	NaN	NaN	NaN	1.000000	3.000000	
25%	30.000000	NaN	NaN	NaN	NaN	NaN	3.000000	5.500000	
50%	41.000000	NaN	NaN	NaN	NaN	NaN	6.000000	6.500000	
75%	53.000000	NaN	NaN	NaN	NaN	NaN	8.000000	7.500000	
max	65.000000	NaN	NaN	NaN	NaN	NaN	10.000000	10.000000	

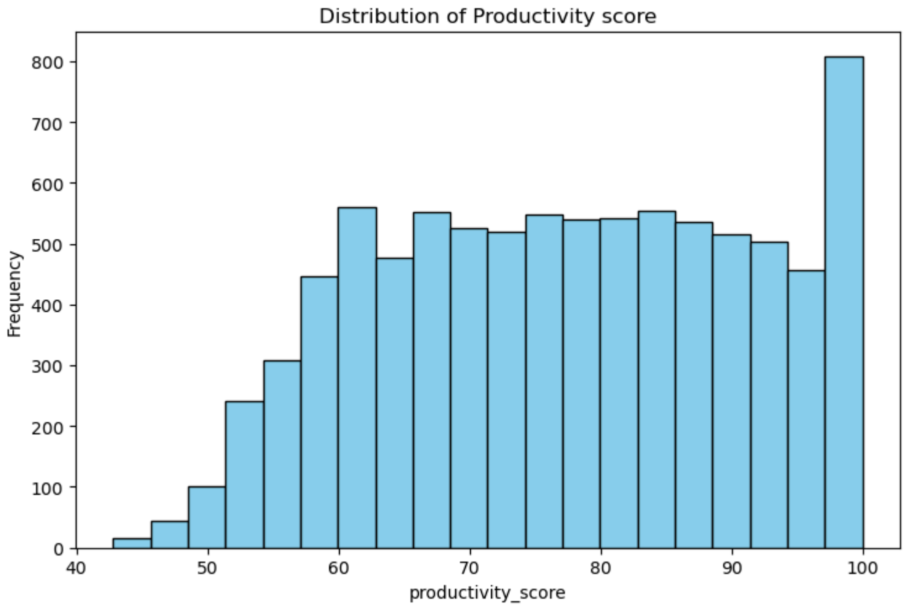
Exploratory Data Analysis

In [19]:
```
# histogram to show numerical columns values distribution
columns_to_plot = ["age","stress_level","sleep_hours","depression_score","anxiety_score","social_support_score","product

# Set up the plot grid
plt.figure(figsize=(15, 20))

for i, col in enumerate(columns_to_plot, 1):
    plt.subplot(4, 2, i)
    df[col].dropna().hist(bins=20, color='skyblue', edgecolor='black')
    plt.title(f'Distribution of {col.capitalize().replace("_", " ")}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.grid(False)

plt.tight_layout()
plt.show()
```

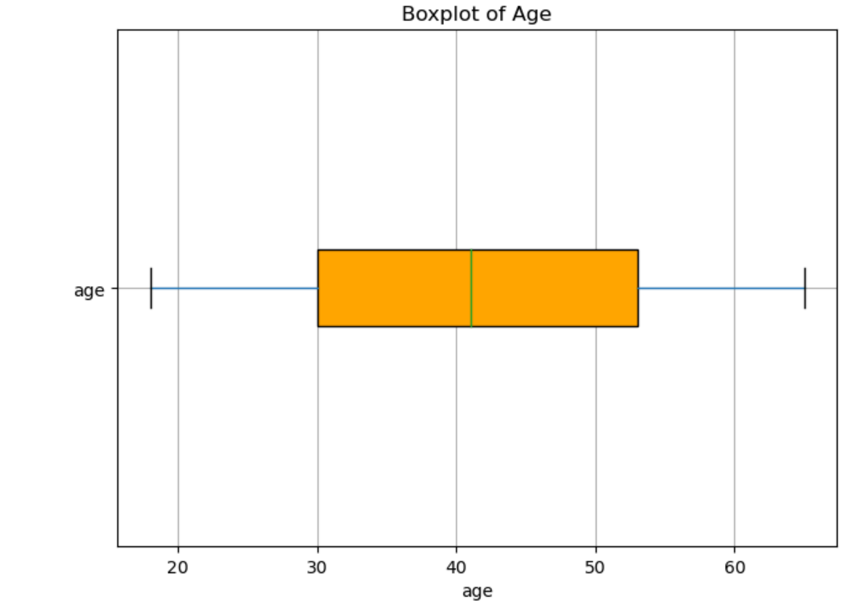

Distribution of Age
Frequency
0
100
200
300
400
500
20
30
40
50
60
age
Distribution of Stress level
Frequency
0
200
400
600
800
1000
2
4
6
8
10
stress_level
Distribution of Sleep hours
Frequency
0
200
400
600
800
3
4
5
6
7
8
9
10
sleep_hours
Distribution of Depression score
Frequency
0
100
200
300
400
500
600
0
5
10
15
20
25
30
depression_score
Distribution of Anxiety score
600
700
800
Distribution of Social support score
400
500

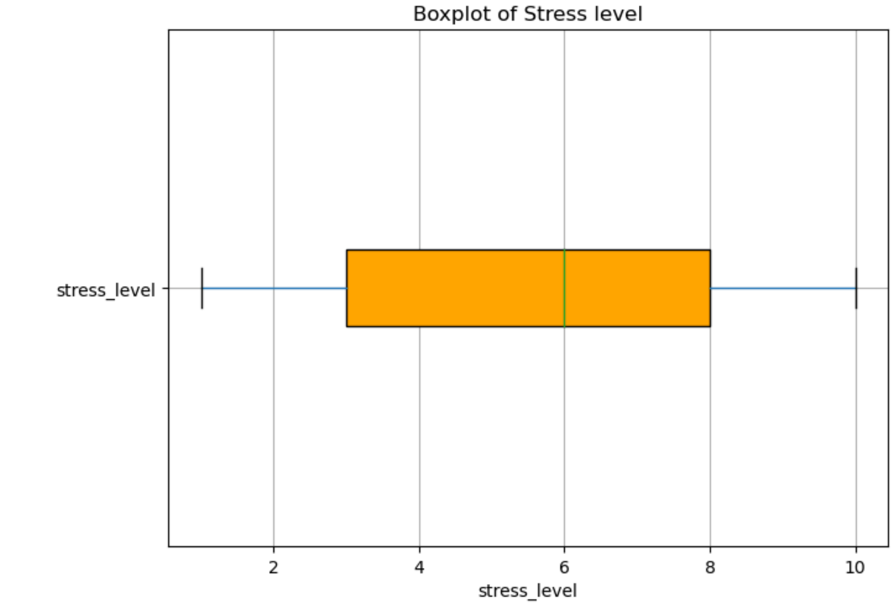

Frequency
0
100
200
300
400
500
0
5
10
15
20
anxiety_score

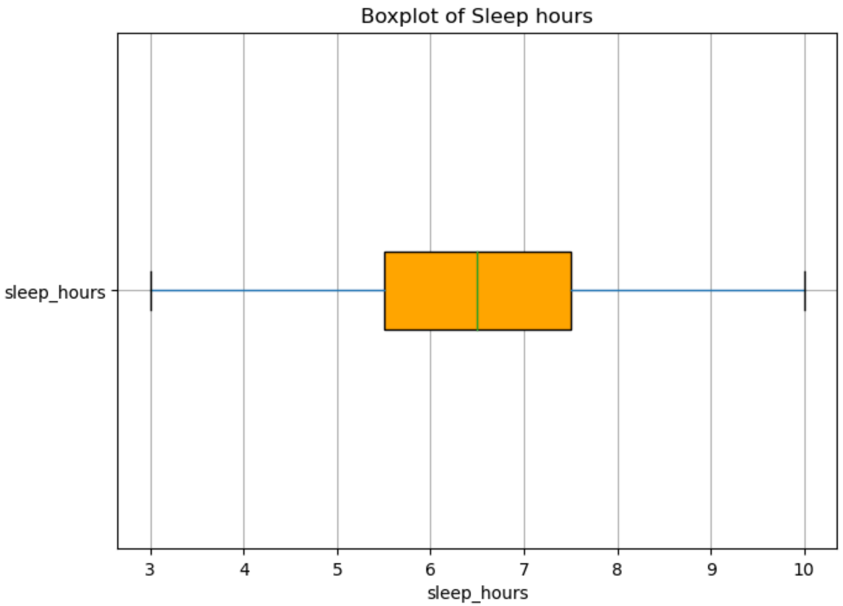

Frequency
0
100
200
300
0
20
40
60
80
100
social_support_score

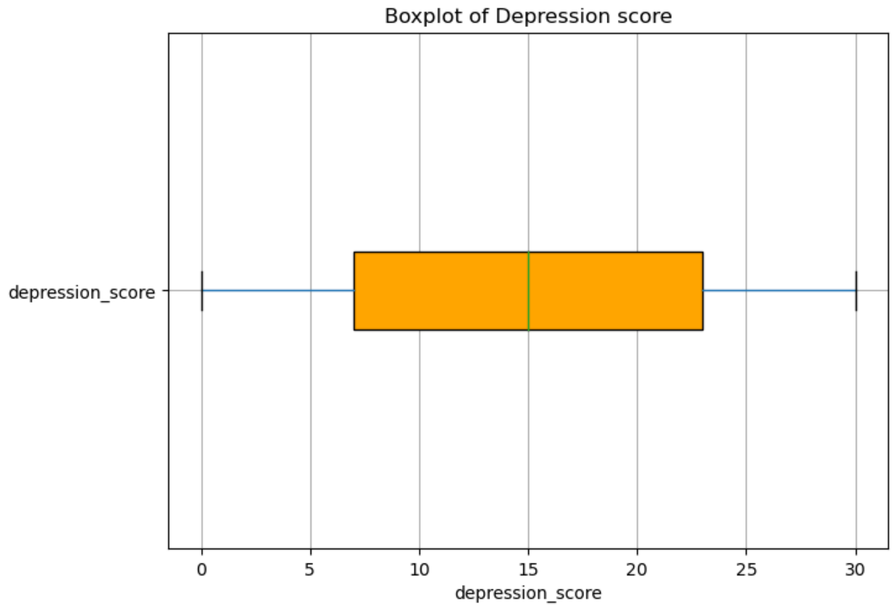

Distribution of Productivity score
Frequency
0
100
200
300
400
500
600
700
800
40
50
60
70
80
90
100
productivity_score

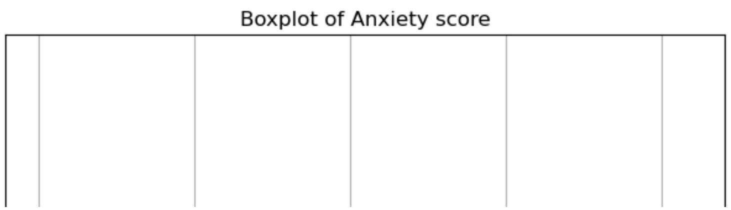
In [20]:
```
# boxplot to show numerical columns values distribution
columns_to_plot = ["age","stress_level","sleep_hours","depression_score","anxiety_score","social_support_score","product

# Set up the plot grid for boxplots
plt.figure(figsize=(15, 20))

for i, col in enumerate(columns_to_plot, 1):
    plt.subplot(4, 2, i)
    df.boxplot(column=col, vert=False, patch_artist=True, boxprops=dict(facecolor='orange'))
    plt.title(f'Boxplot of {col.capitalize().replace("_", " ")}')
    plt.xlabel(col)

plt.tight_layout()
plt.show()
```

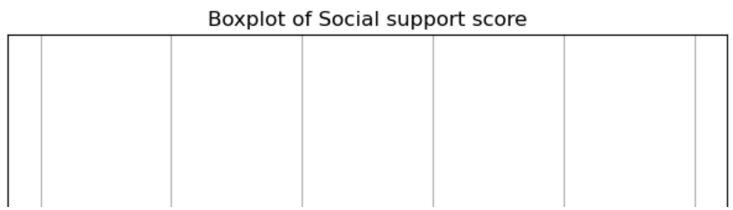

Boxplot of Age
age
20
30
40
50
60
age

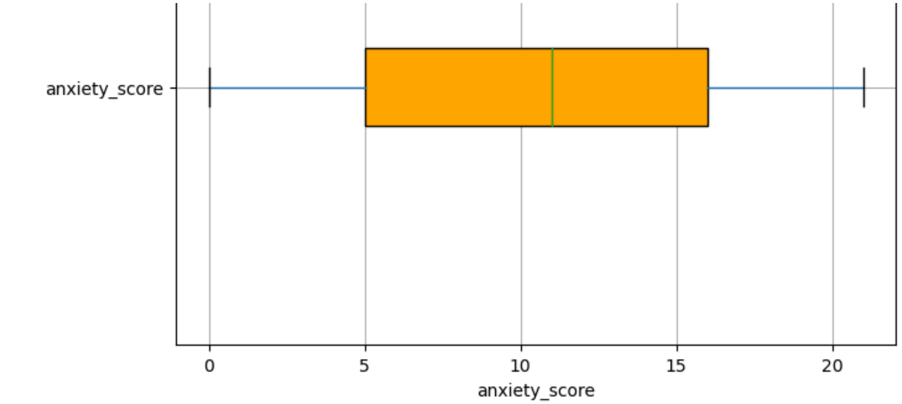

Boxplot of Stress level
stress_level
2
4
6
8
10
stress_level

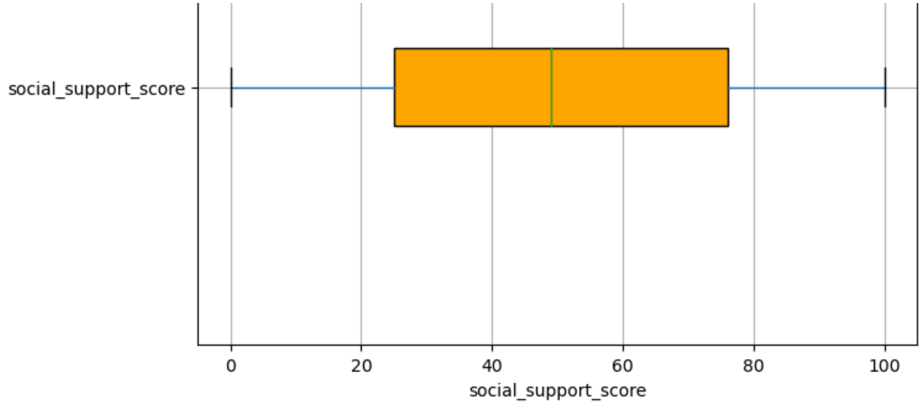

Boxplot of Sleep hours
sleep_hours
3
4
5
6
7
8
9
10
sleep_hours


Boxplot of Depression score
depression_score
0
5
10
15
20
25
30
depression_score

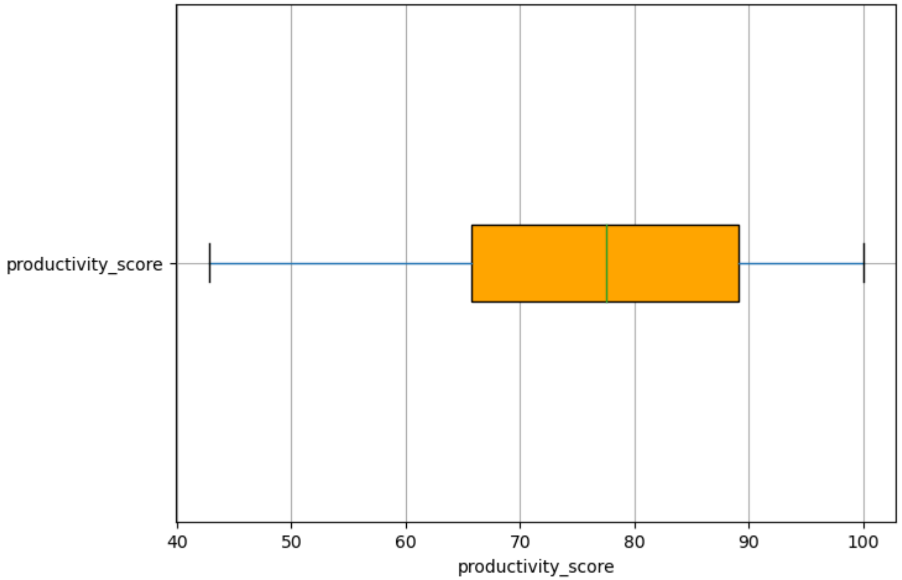

Boxplot of Anxiety score


Boxplot of Social support score



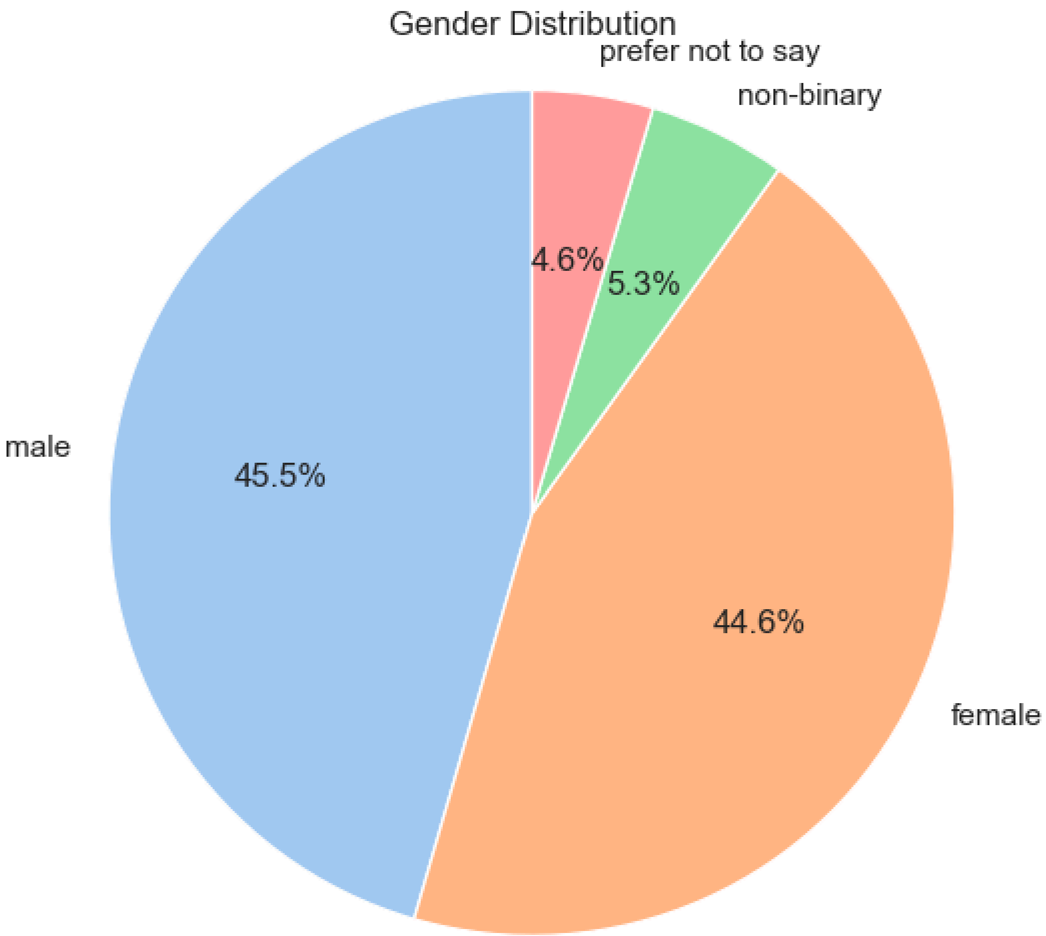


Boxplot of Productivity score

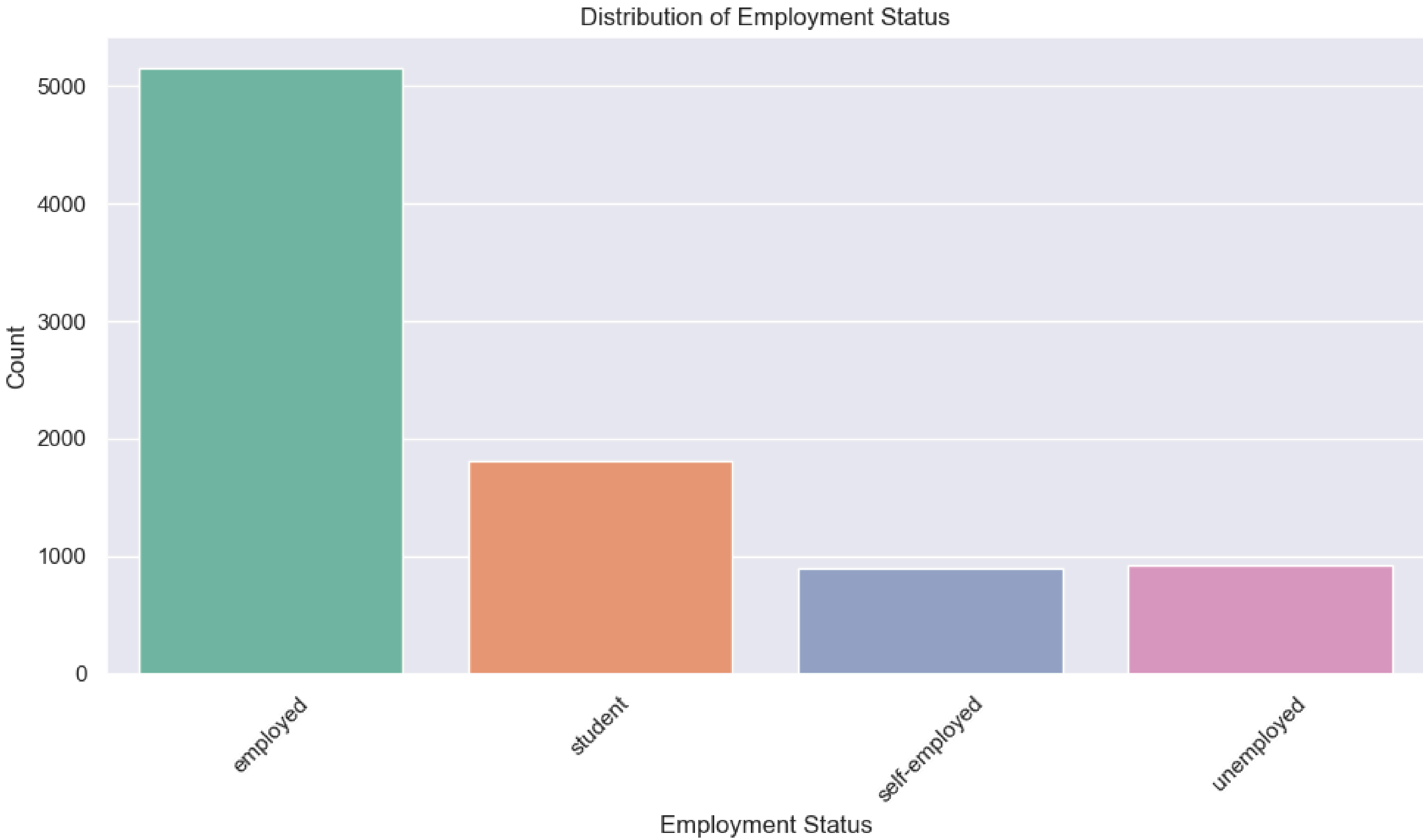


univariate analysis

In [21]:
```
# pie chart for the gender column to show distribution
sns.set_theme()
gender_counts = df['gender'].value_counts()

plt.figure(figsize=(6, 6))
plt.pie(gender_counts,labels=gender_counts.index, autopct='%1.1f%%', startangle=90, colors=sns.color_palette('pastel'))
plt.title('Gender Distribution')
plt.axis('equal')
plt.show()
```


Gender Distribution
prefer not to say
non-binary
male
female
4.6%
5.3%
45.5%
44.6%

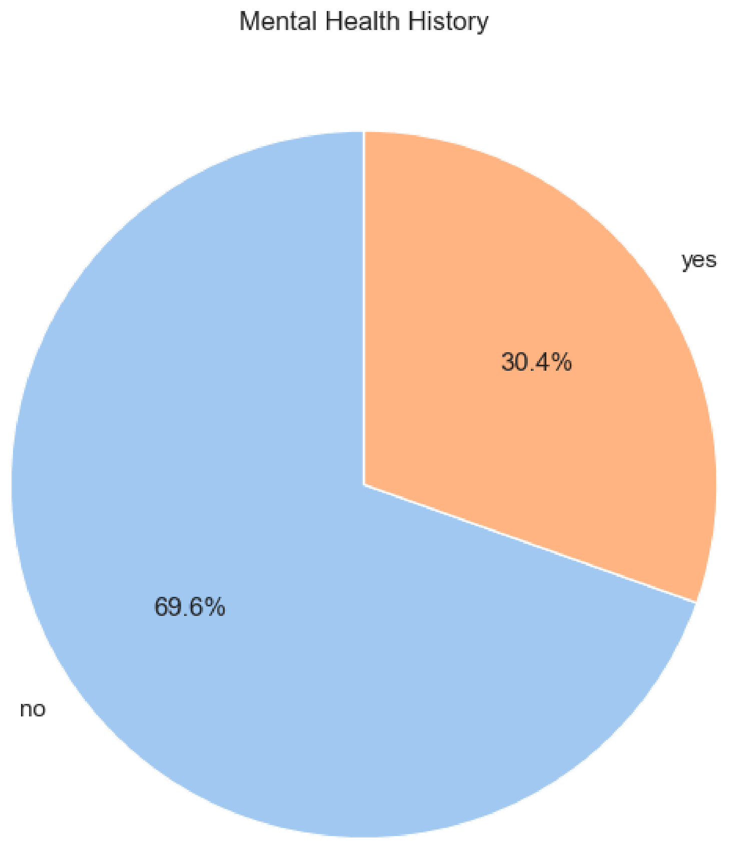
In [22]:
```
# Set Seaborn theme
sns.set_theme()

# Create the bar plot
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='employment_status', palette='Set2')

# Add labels and title
plt.title('Distribution of Employment Status')
plt.xlabel('Employment Status')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()

plt.show()
```

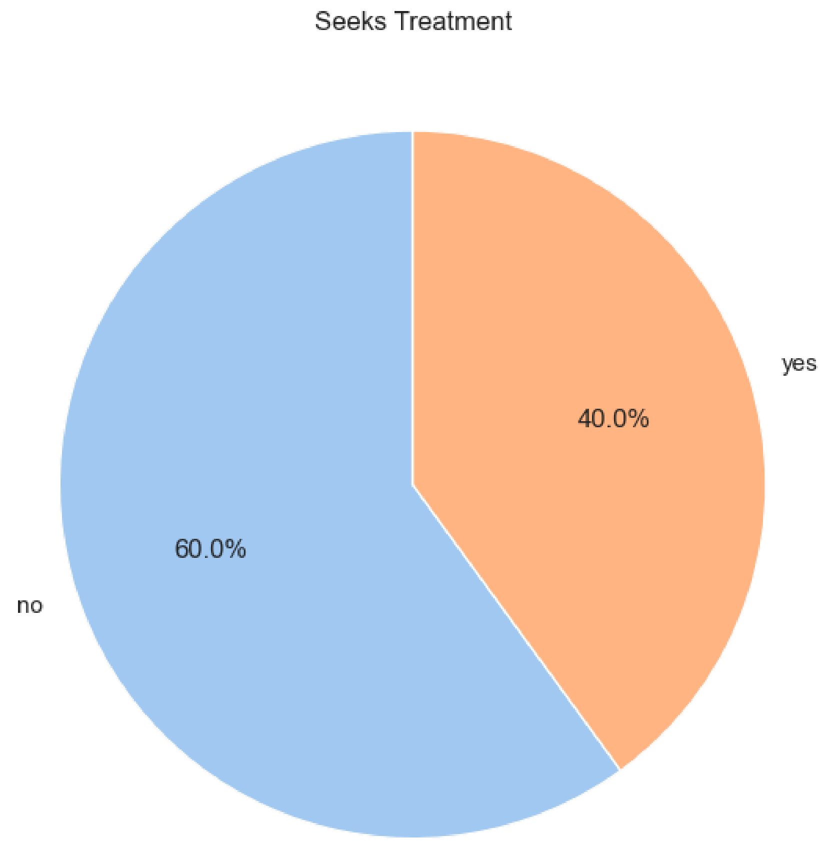

Distribution of Employment Status
Count
5000
4000
3000
2000
1000
0
employed
student
self-employed
unemployed
Employment Status

In [23]:
```
# distribution of the work environment column
# Set Seaborn theme
sns.set_theme()

# Create the bar plot
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='work_environment', palette='Set2')

# Add labels and title
plt.title('Distribution of Work_Environment')
plt.xlabel('Work_Environment')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()

plt.show()
```

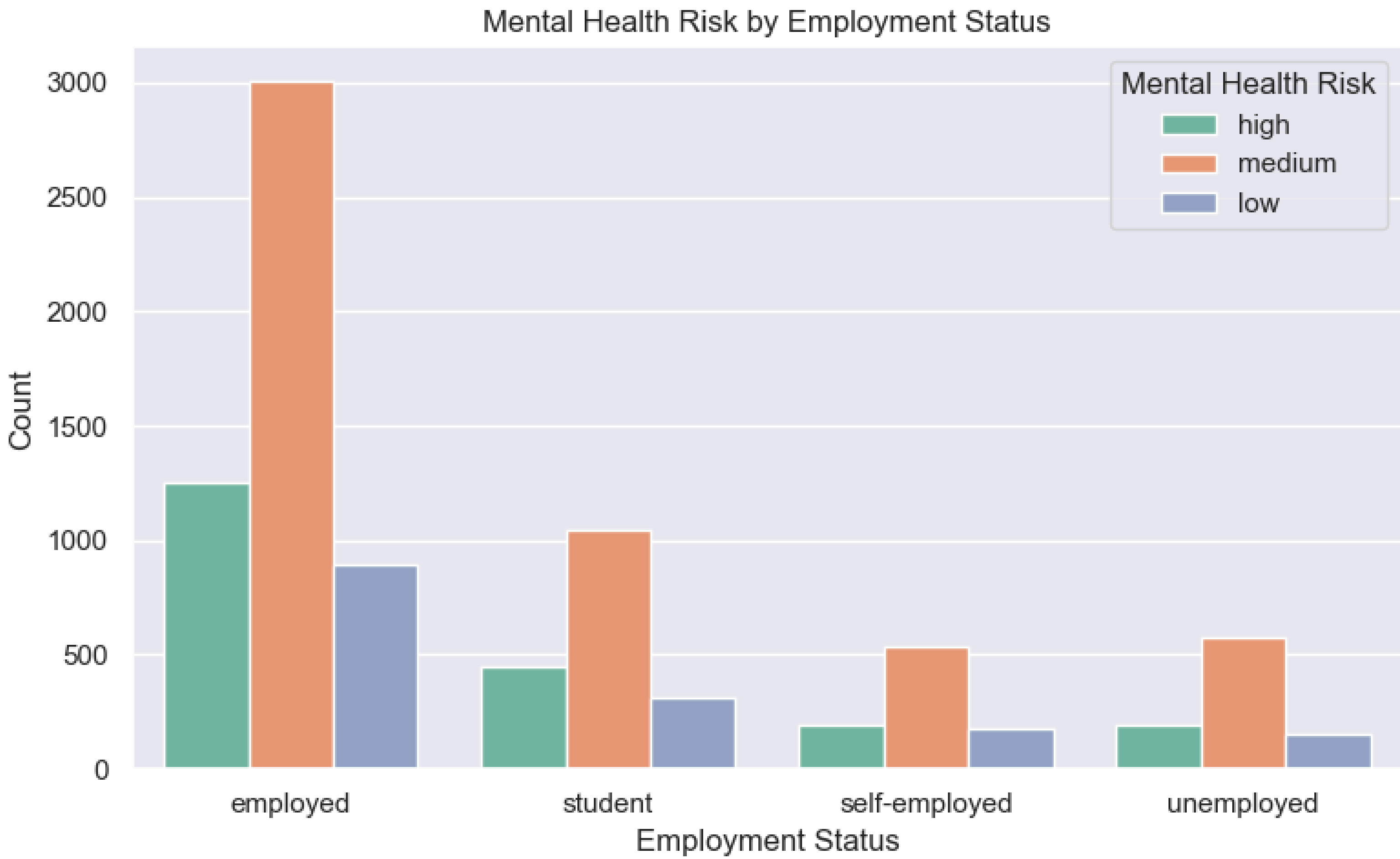

Distribution of Work_Environment
Count
0
1000
2000
3000
4000
on-site
remote
hybrid
Work_Environment

In [24]:
```
# distribution of the columns mental_health_history, seeks_treatment
sns.set_theme()

columns = ['mental_health_history', 'seeks_treatment']
plt.figure(figsize=(12, 6))

for i, col in enumerate(columns, 1):
    plt.subplot(1, 2, i)
    counts = df[col].value_counts()
    plt.pie(counts,labels=counts.index,autopct='%1.1f%%',startangle=90,colors=sns.color_palette('pastel'))
    plt.title(f'{col.replace("_", " ").title()}')

plt.tight_layout()
plt.show()
```

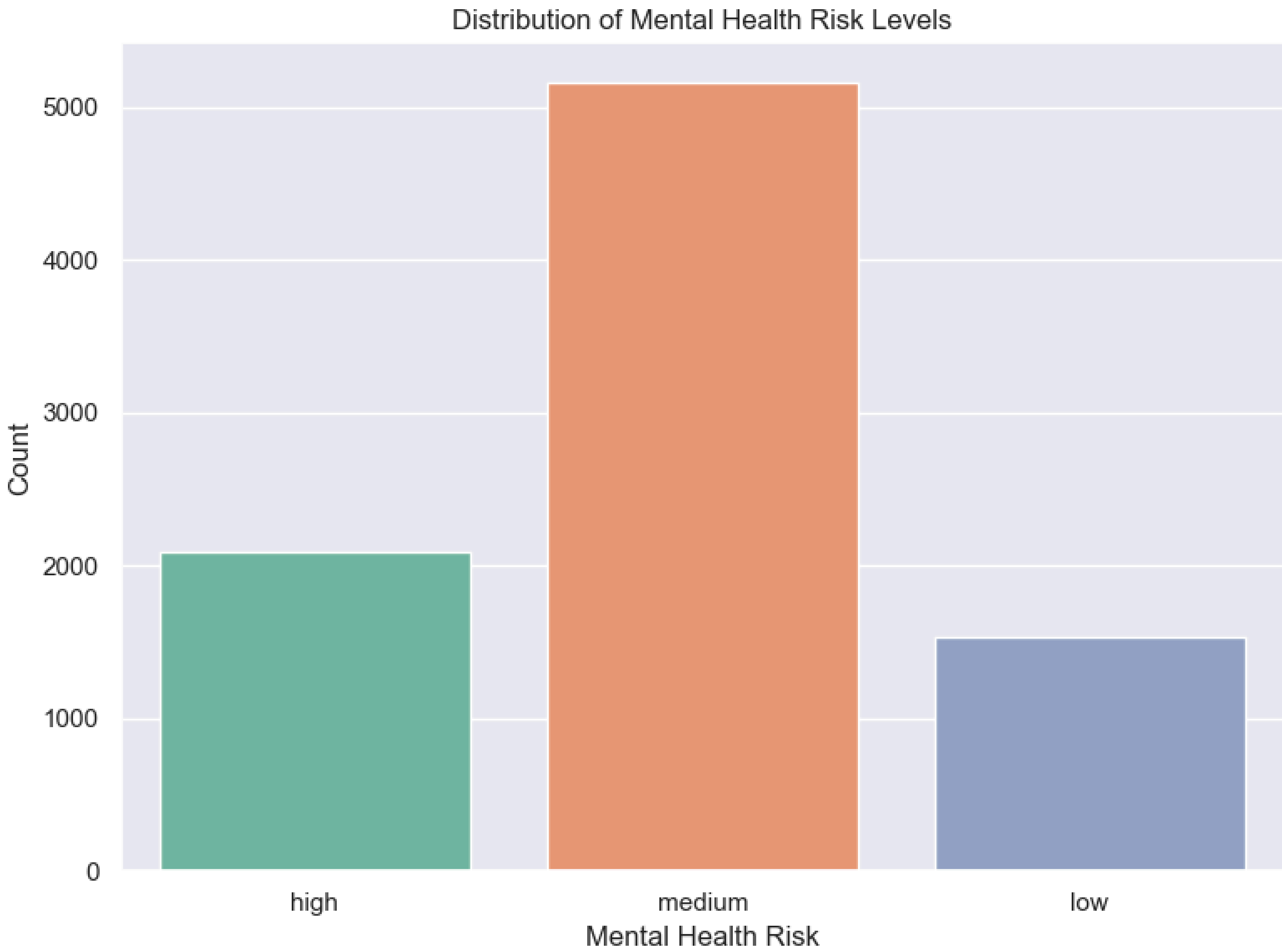

Mental Health History
yes
30.4%
69.6%
no


Seeks Treatment
yes
40.0%
60.0%
no

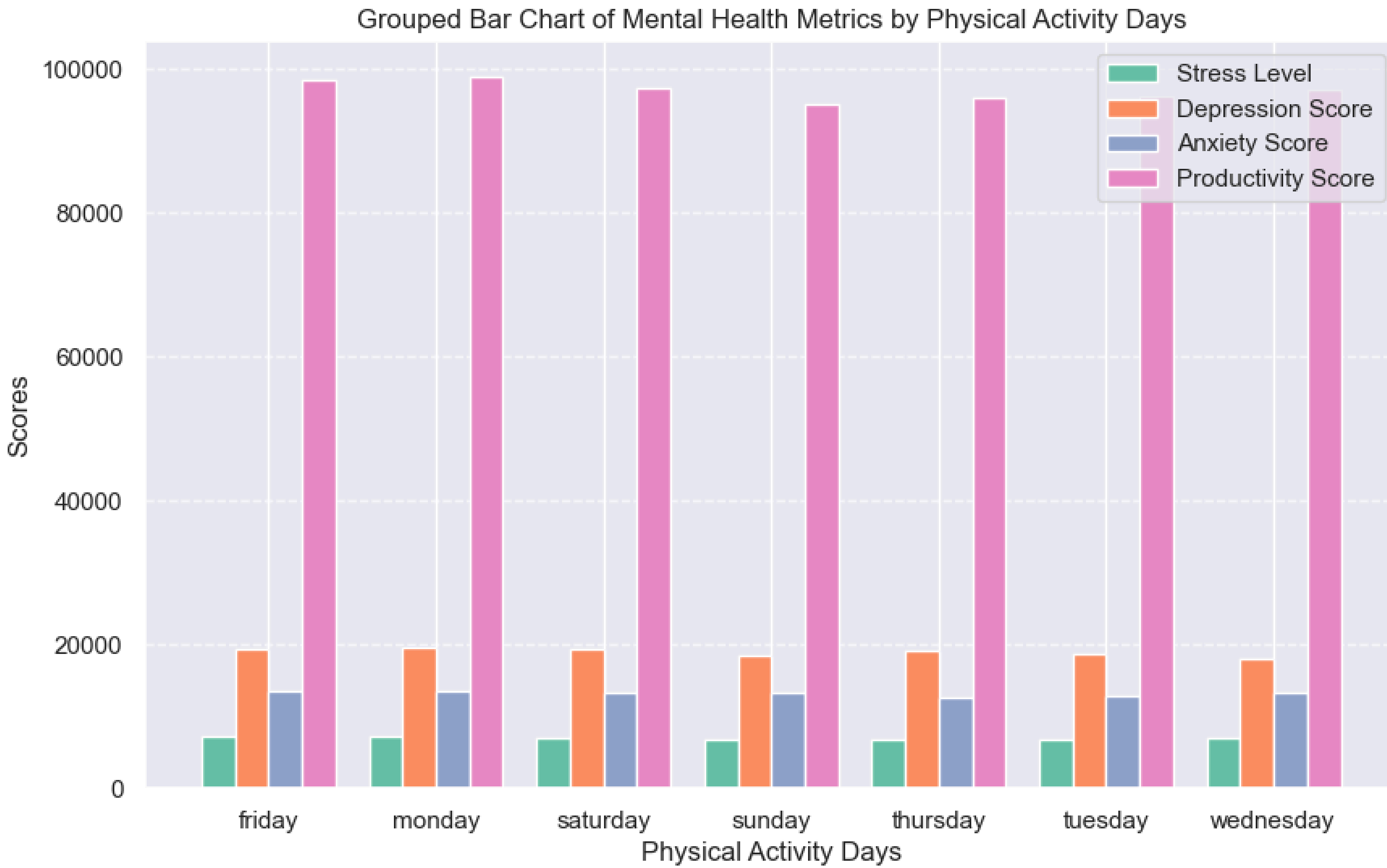
In [25]:
```
# high mental risk by work_status
# Set theme
sns.set_theme()

# Plotting mental health risk counts by employment status
plt.figure(figsize=(8, 5))
sns.countplot(data=df, x='employment_status', hue='mental_health_risk', palette='Set2')
plt.title('Mental Health Risk by Employment Status')
plt.xlabel('Employment Status')
plt.ylabel('Count')
plt.legend(title='Mental Health Risk')
plt.tight_layout()
plt.show()
```


Mental Health Risk by Employment Status
Mental Health Risk
high
medium
low
Count
0
500
1000
1500
2000
2500
3000
employed
student
self-employed
unemployed
Employment Status

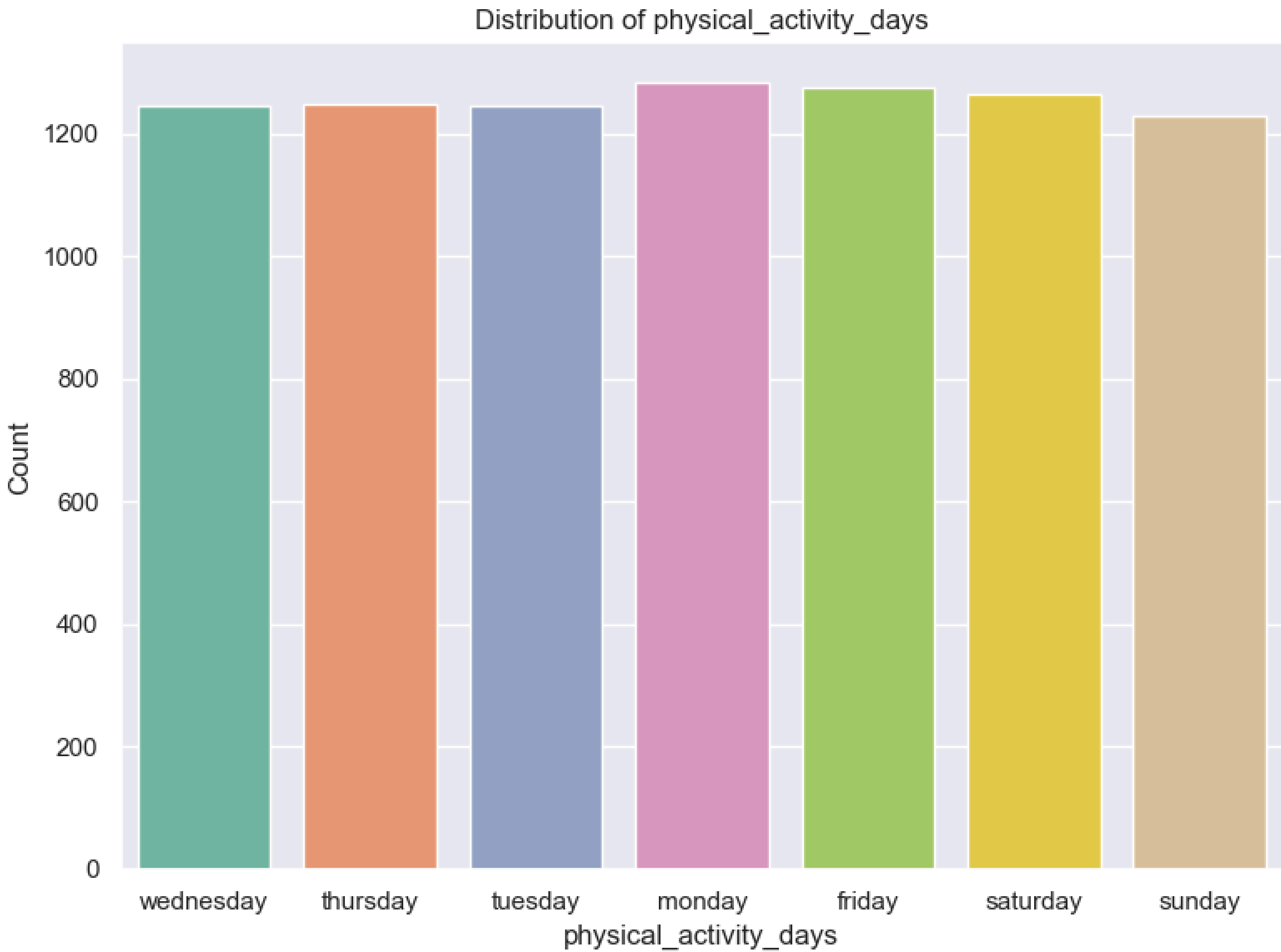
In [26]:
```
# Distribution of the mental health risk column
sns.set_theme()
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='mental_health_risk', palette='Set2')  # changed palette

plt.title('Distribution of Mental Health Risk Levels')
plt.xlabel('Mental Health Risk')
plt.ylabel('Count')
plt.tight_layout()

plt.show()
```


Distribution of Mental Health Risk Levels
Count
0
1000
2000
3000
4000
5000
high
medium
low
Mental Health Risk

In [27]:
```
# Group by 'physical_activity_days' and calculate the sum for each score
grouped_data = df.groupby('physical_activity_days').sum().reset_index()

bar_width = 0.2
index = np.arange(len(grouped_data))

plt.figure(figsize=(10, 6))

# Ploting each category side by side
plt.bar(index, grouped_data['stress_level'], width=bar_width, label='Stress Level', color='#66c2a5')
plt.bar(index + bar_width, grouped_data['depression_score'], width=bar_width, label='Depression Score', color='#fc8d62')
plt.bar(index + 2 * bar_width, grouped_data['anxiety_score'], width=bar_width, label='Anxiety Score', color='#8da0cb')
plt.bar(index + 3 * bar_width, grouped_data['productivity_score'], width=bar_width, label='Productivity Score', color='#

plt.title('Grouped Bar Chart of Mental Health Metrics by Physical Activity Days')
plt.xlabel('Physical Activity Days')
plt.ylabel('Scores')
plt.xticks(index + 1.5 * bar_width, grouped_data['physical_activity_days'])
plt.legend(loc='upper right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

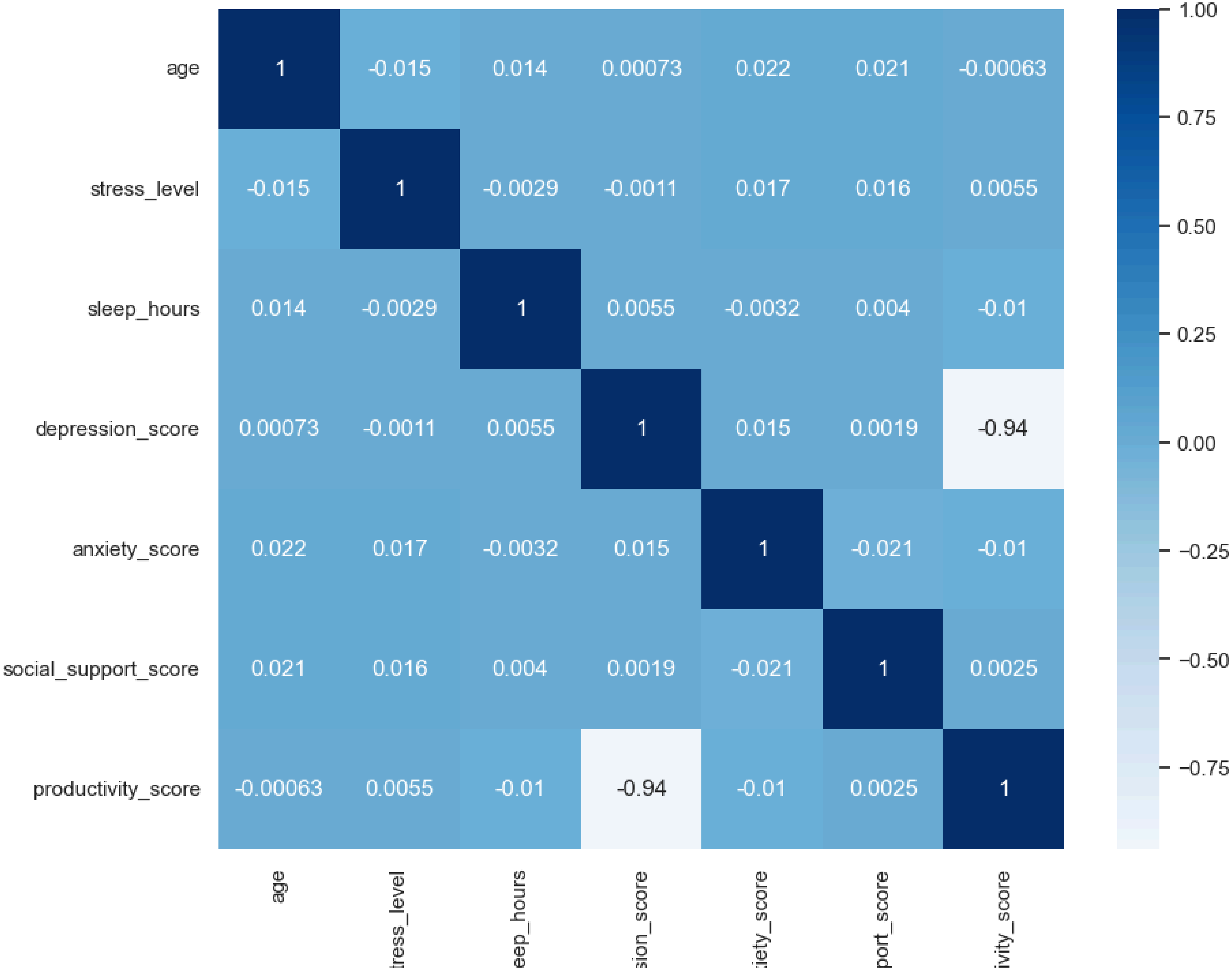

Grouped Bar Chart of Mental Health Metrics by Physical Activity Days
Stress Level
Depression Score
Anxiety Score
Productivity Score
Scores
100000
80000
60000
40000
20000
0
friday
monday
saturday
sunday
thursday
tuesday
wednesday
Physical Activity Days

In [28]:
```
# distribution of the physical_activity_days
sns.set_theme()
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='physical_activity_days', palette='Set2')
plt.title('Distribution of physical_activity_days')
plt.xlabel('physical_activity_days')
plt.ylabel('Count')
plt.tight_layout()

plt.show()
```


Distribution of physical_activity_days
Count
0
200
400
600
800
1000
1200
wednesday
thursday
tuesday
monday
friday
saturday
sunday
physical_activity_days

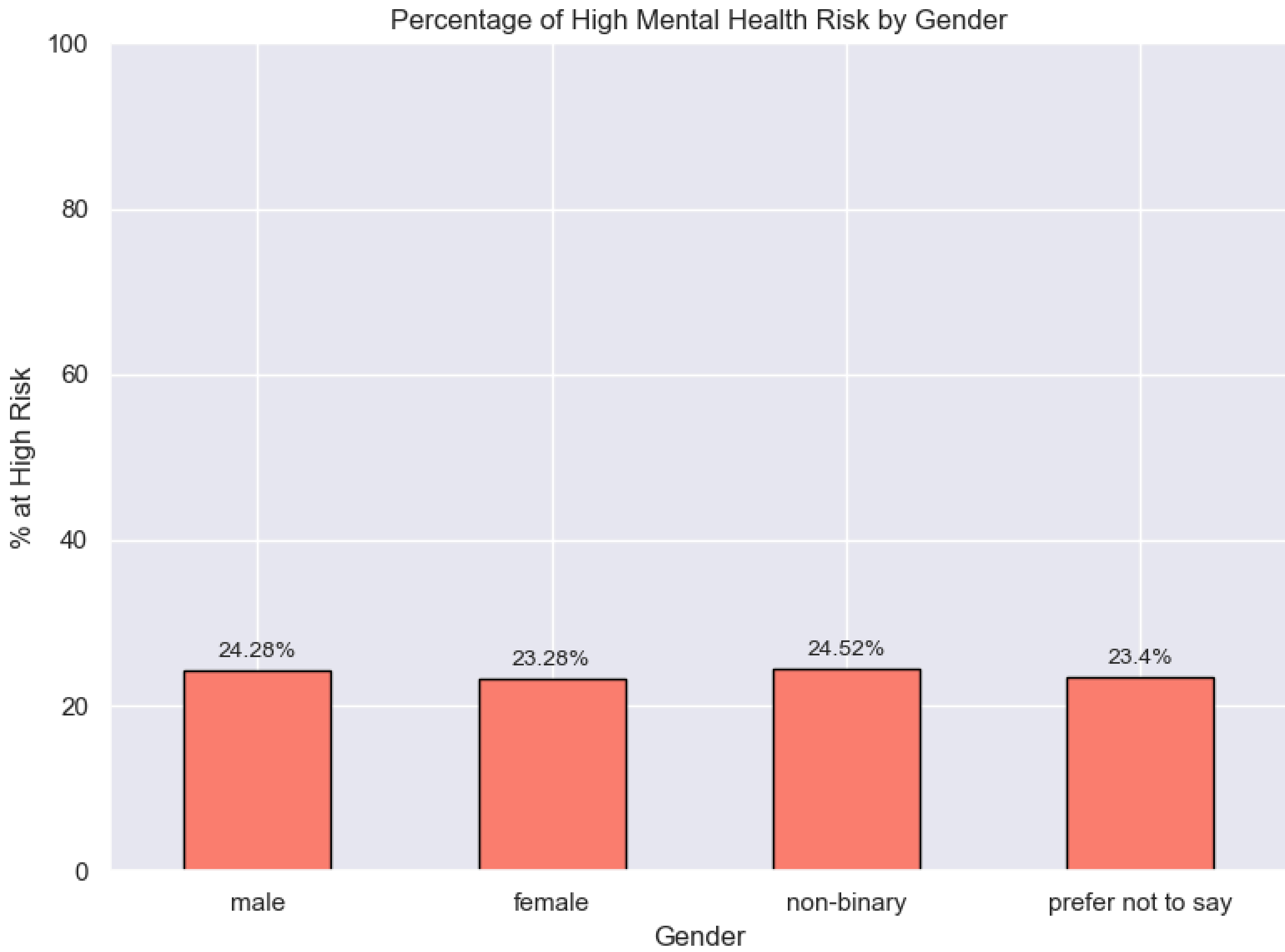
bivariate analysis

In [29]:
```
# heatmap to show numerical columns correlations
plt.figure(figsize = (10, 8))
sns.heatmap(df.corr(), annot = True, cmap = 'Blues', center = 0)
plt.show()
```

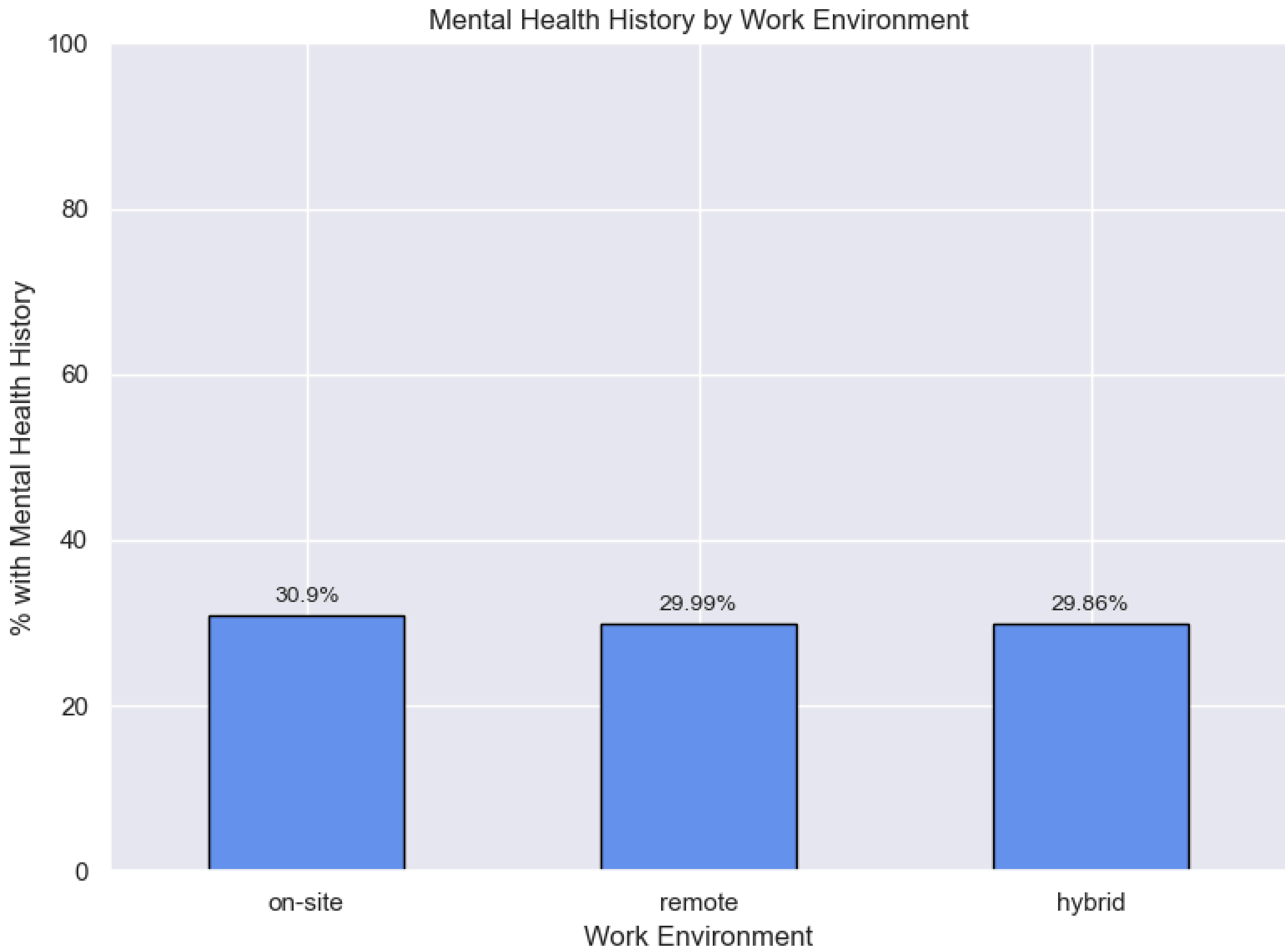

age
stress_level
sleep_hours
depression_score
anxiety_score
social_support_score
productivity_score
1 -0.015 0.014 0.00073 0.022 0.021 -0.00063
-0.015 1 -0.0029 -0.0011 0.017 0.016 0.0055
0.014 -0.0029 1 0.0055 -0.0032 0.004 -0.01
0.00073 -0.0011 0.0055 1 0.015 0.0019 -0.94
0.022 0.017 -0.0032 0.015 1 -0.021 -0.01
0.021 0.016 0.004 0.0019 -0.021 1 0.0025
-0.00063 0.0055 -0.01 -0.94 -0.01 0.0025 1
age
tress_level
eep_hours
sion_score
iety_score
port_score
ivity_score
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75


sl
depres
anx
social_sup
product

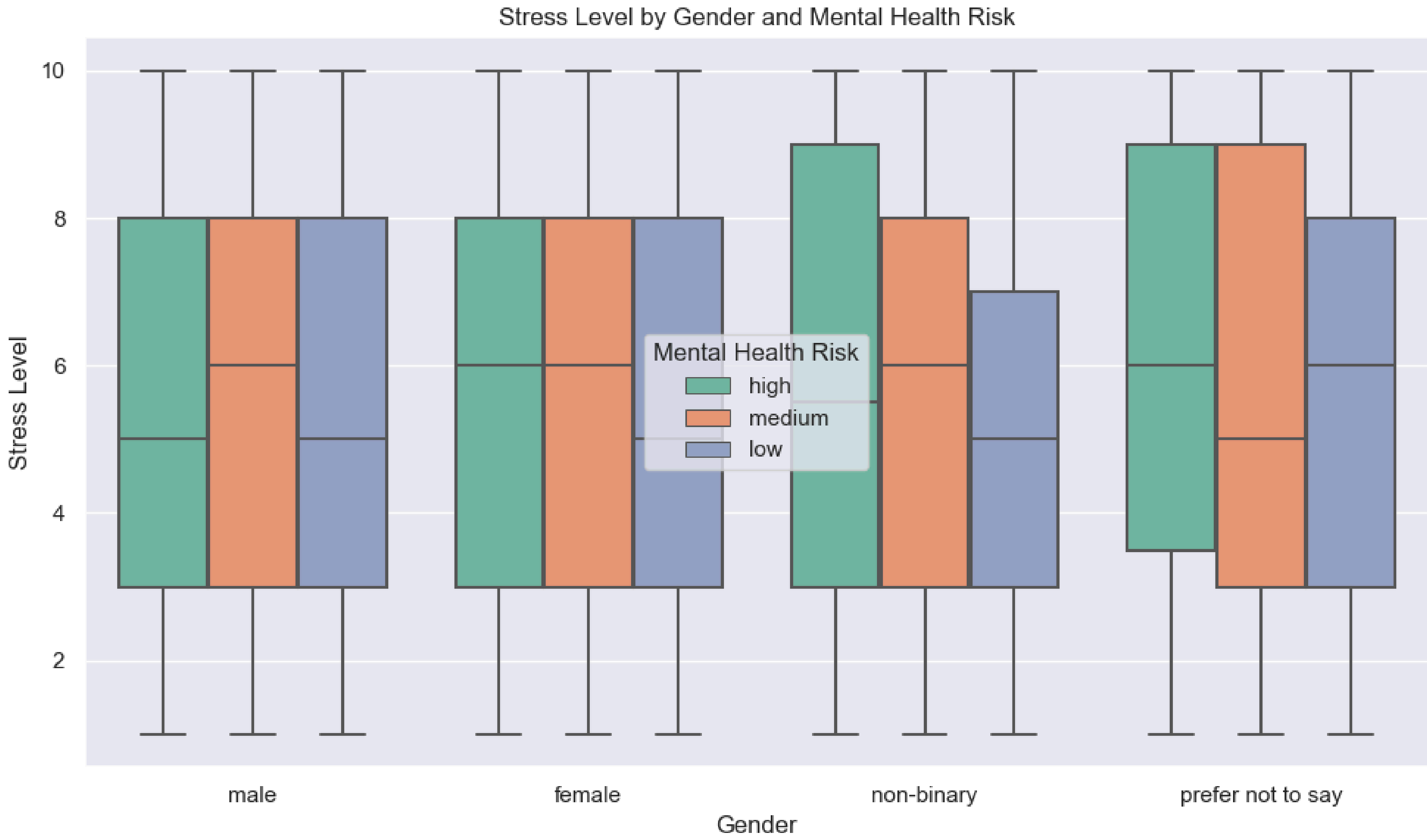
In [30]:
```
# Are certain genders more likely to be at high risk?

total_by_gender = df['gender'].value_counts()
high_risk_by_gender = df[df['mental_health_risk'] == 'high']['gender'].value_counts()
risk_percent = (high_risk_by_gender / total_by_gender * 100).round(2)

plt.figure(figsize=(8, 6))
bars = risk_percent.plot(kind='bar', color='salmon', edgecolor='black')

plt.title('Percentage of High Mental Health Risk by Gender')
plt.xlabel('Gender')
plt.ylabel('% at High Risk')
plt.xticks(rotation=0)
plt.ylim(0, 100)
plt.tight_layout()

for index, value in enumerate(risk_percent):
    plt.text(index, value + 1, f'{value}%', ha='center', va='bottom', fontsize=10)

plt.show()
```


Percentage of High Mental Health Risk by Gender
100
80
60
40
20
0
% at High Risk
24.28%
23.28%
24.52%
23.4%
male
female
non-binary
prefer not to say
Gender

In [31]:
```
# Do unemployed people seek treatment more or less?

df['work_environment'] = df['work_environment'].str.strip().str.lower()
df['mental_health_history'] = df['mental_health_history'].str.strip().str.lower()

history_counts = df[df['mental_health_history'] == 'yes']['work_environment'].value_counts()
total_counts = df['work_environment'].value_counts()
history_percent = (history_counts / total_counts * 100).round(2)

plt.figure(figsize=(8, 6))
bars = history_percent.plot(kind='bar', color='cornflowerblue', edgecolor='black')

plt.title('Mental Health History by Work Environment')
plt.xlabel('Work Environment')
plt.ylabel('% with Mental Health History')
plt.xticks(rotation=0)
plt.ylim(0, 100)

for i, value in enumerate(history_percent):
    plt.text(i, value + 1, f'{value}%', ha='center', va='bottom', fontsize=10)

plt.tight_layout()
plt.show()
```


Mental Health History by Work Environment
% with Mental Health History
100
80
60
40
20
0
30.9%
29.99%
29.86%
on-site
remote
hybrid
Work Environment

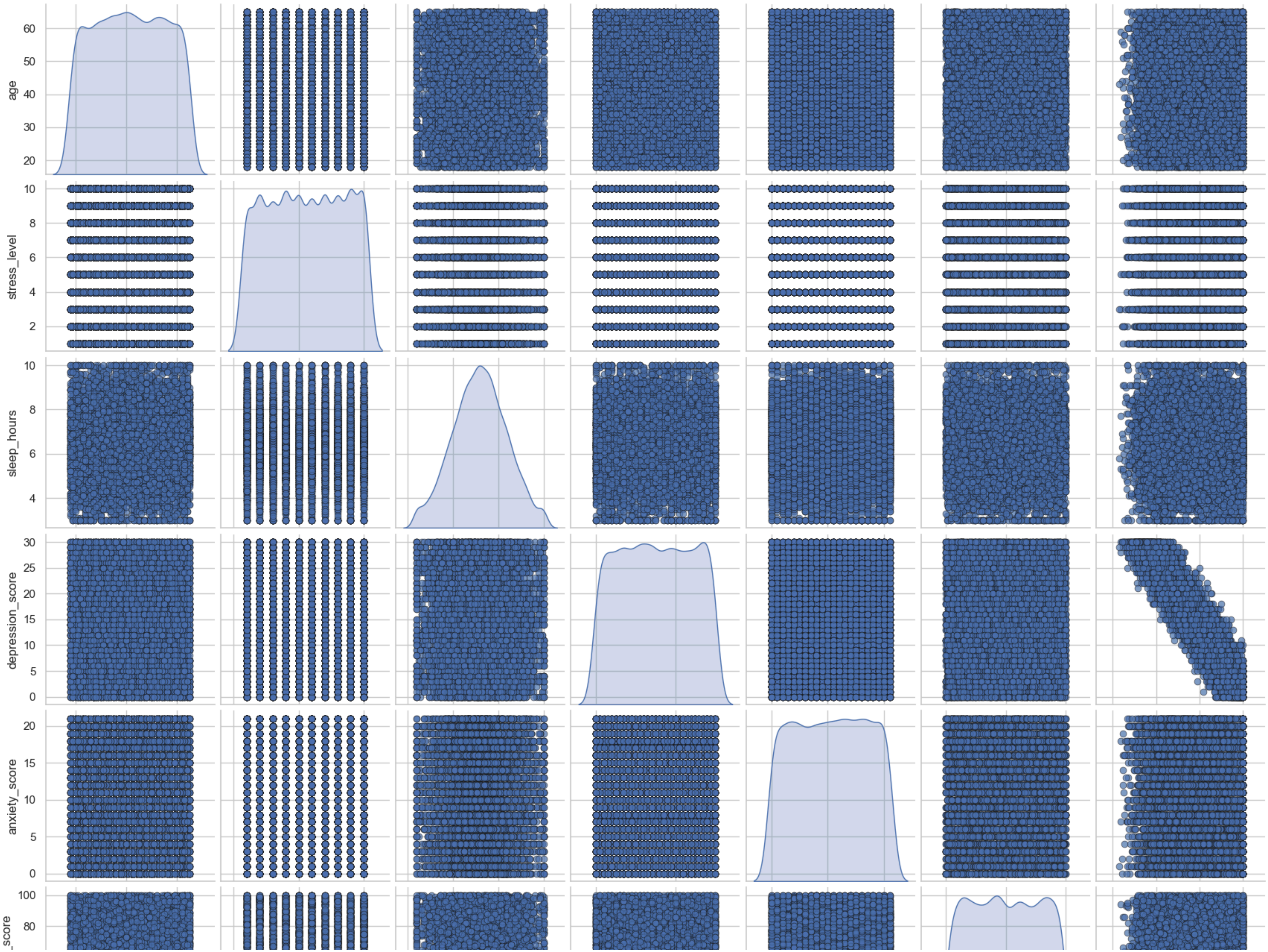
In [32]:
```
# Stress Level by Gender and Mental Health Risk
df['gender'] = df['gender'].str.strip().str.lower()
df['mental_health_risk'] = df['mental_health_risk'].str.strip().str.lower()

plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='gender', y='stress_level', hue='mental_health_risk', palette='Set2')

plt.title('Stress Level by Gender and Mental Health Risk')
plt.xlabel('Gender')
plt.ylabel('Stress Level')
plt.legend(title='Mental Health Risk')
plt.tight_layout()
plt.show()
```

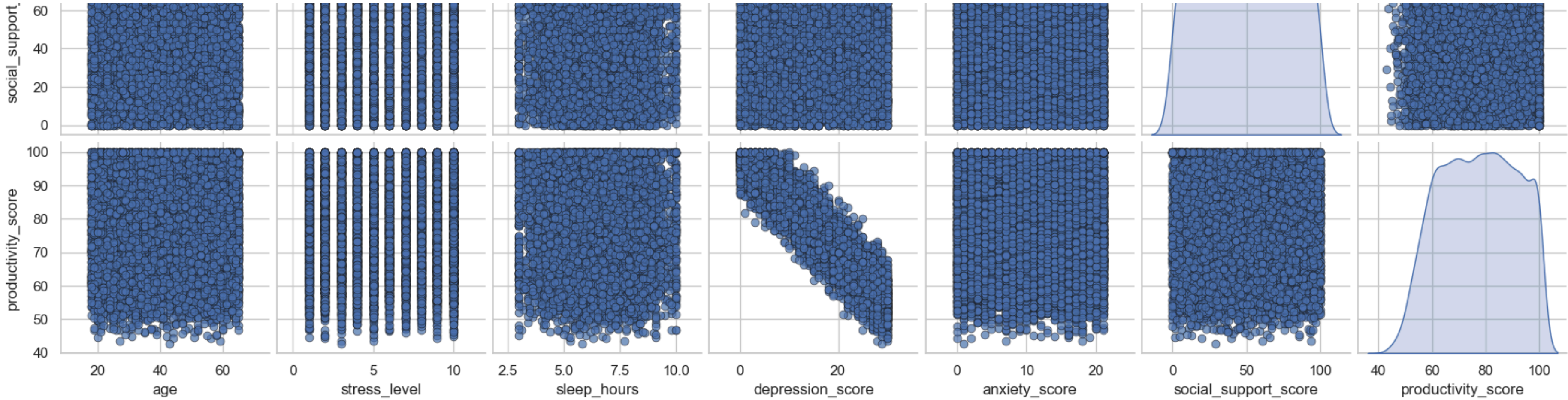


Stress Level by Gender and Mental Health Risk
Stress Level
10
8
6
4
2
Mental Health Risk
high
medium
low
male
female
non-binary
prefer not to say
Gender

In [33]:
```
# dataset pairplot
sns.set_style("whitegrid")

pairplot = sns.pairplot(df,diag_kind='kde',kind='scatter',plot_kws={'alpha':0.7, 's':40, 'edgecolor':'k'},  diag_kws={'sl
    palette='coolwarm',
)

plt.suptitle('Pairplot of Dataset Features', fontsize=16, y=1.02)
plt.show()
```

Pairplot of Dataset Features





Label Encoding Object Columns

In [34]:
```
encoder = LabelEncoder()
```

In [35]:
```
# encoding the columns
df['gender'] = encoder.fit_transform(df['gender'])
df['employment_status'] = encoder.fit_transform(df['employment_status'])
df['work_environment'] = encoder.fit_transform(df['work_environment'])
df['mental_health_history'] = encoder.fit_transform(df['mental_health_history'])
df['seeks_treatment'] = encoder.fit_transform(df['seeks_treatment'])
df['physical_activity_days'] = encoder.fit_transform(df['physical_activity_days'])
df['mental_health_risk'] = encoder.fit_transform(df['mental_health_risk'])
```

In [36]:
```
# reviewing the data
df
```

Out[36]:

	age	gender	employment_status	work_environment	mental_health_history	seeks_treatment	stress_level	sleep_hours	physical_activity_days
0	56	1	0	1	1	1	6	6.2	6
1	46	0	2	1	0	1	10	9.0	4
2	32	0	0	1	1	0	7	7.7	5
3	60	2	1	1	0	0	4	4.5	4
5	38	0	3	1	1	1	3	9.9	6
...	...	...	...	...	...	...	...	...	...
9993	59	0	0	1	0	1	9	6.5	0
9994	19	0	0	0	0	0	4	4.5	4
9995	34	0	0	1	1	1	5	6.1	6
9996	47	1	0	1	1	0	1	5.7	6
9999	44	1	3	2	0	1	5	6.4	3

8788 rows × 14 columns

In [37]:
```
# splitting the data into dependent and independent variables
X = df[['age', 'gender', 'employment_status', 'work_environment',
       'mental_health_history', 'seeks_treatment', 'stress_level',
       'sleep_hours', 'physical_activity_days', 'depression_score',
       'anxiety_score', 'social_support_score', 'productivity_score']]
y = df['mental_health_risk']
```

In [38]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=17)
```

In [39]:
```
X_train
```

Out[39]:

	age	gender	employment_status	work_environment	mental_health_history	seeks_treatment	stress_level	sleep_hours	physical_activity_days
9801	38	1	2	1	0	1	6	5.3	5
2316	21	0	0	1	0	0	1	9.2	5
3232	64	1	0	1	0	0	2	3.8	3
8622	48	0	2	1	0	0	10	6.7	4
3520	32	1	2	1	0	0	5	6.9	2
...	...	...	...	...	...	...	...	...	...
4136	30	0	0	1	0	1	9	4.1	5
9633	44	1	2	2	0	0	5	5.4	3
6914	57	1	0	1	0	0	7	5.2	3
462	46	1	1	0	0	0	1	7.6	2
2500	38	0	0	1	0	0	9	7.1	1

7030 rows × 13 columns

In [40]:
```
y_train
```

Out[40]:
```
9801    0
2316    2
3232    2
8622    2
3520    1
       ..
4136    2
9633    2
6914    2
462     2
2500    1
Name: mental_health_risk, Length: 7030, dtype: int32
```

Normalisation of The Independent Variables

In [41]:
```
scaler = MinMaxScaler(feature_range=(0, 1))
```

In [42]:
```
X_train = scaler.fit_transform(X_train)
```

In [43]:
```
X_test = scaler.transform(X_test)
```

In [44]:
```
X_test
```

Out[44]:
```
array([[0.46808511, 0.33333333, 0.        , ..., 0.33333333, 0.21      ,
        0.13461538],
       [0.25531915, 0.        , 1.        , ..., 0.0952381 , 0.48      ,
        0.40384615],
       [0.31914894, 0.33333333, 1.        , ..., 0.80952381, 0.        ,
        0.16608392],
       ...,
       [0.06382979, 0.33333333, 1.        , ..., 0.57142857, 0.07      ,
        0.61013986],
       [0.78723404, 0.        , 0.66666667, ..., 0.14285714, 0.45      ,
        0.75174825],
       [0.91489362, 0.        , 0.66666667, ..., 0.42857143, 0.04      ,
        0.61013986]])
```

In [45]:
```
X_train
```

Out[45]:
```
array([[0.42553191, 0.33333333, 0.66666667, ..., 0.61904762, 0.4       ,
        0.22552448],
       [0.06382979, 0.        , 0.        , ..., 0.57142857, 0.74      ,
        0.76048951],
       [0.9787234 , 0.33333333, 0.        , ..., 0.57142857, 0.91      ,
        0.79195804],
       ...,
       [0.82978723, 0.33333333, 0.        , ..., 0.61904762, 0.41      ,
        0.48076923],
       [0.59574468, 0.33333333, 0.33333333, ..., 0.19047619, 0.99      ,
        0.41783217],
       [0.42553191, 0.        , 0.        , ..., 0.23809524, 0.71      ,
        0.9020979 ]])
```

CLASSIFICATION MODELS

Logistic Regression

In [46]:
```
logistic = LogisticRegression()
```

In [47]:
```
logistic.fit(X_train, y_train)
```

Out[47]:
```
▾ LogisticRegression
LogisticRegression()
```

In [48]:
```
logistic_pred = logistic.predict(X_test)
logistic_pred
```

Out[48]:
```
array([0, 2, 0, ..., 2, 2, 2])
```

In [49]:
```
# logistic regression metrics
logistic_accuracy = accuracy_score(y_test, logistic_pred)
logistic_precision = precision_score(y_test, logistic_pred, average='weighted')
logistic_recall = recall_score(y_test, logistic_pred, average='weighted')
logistic_f1 = f1_score(y_test, logistic_pred, average='weighted')
logistic_confusion = confusion_matrix(y_test, logistic_pred)
```

In [64]:
```
# confusion matrix for logistic
print(logistic_confusion)
```

```
[[ 431    0   19]
 [   0  271   18]
 [   7    6 1006]]
```

Random Forest Classifier

In [51]:
```
randomforest = RandomForestClassifier()
```

In [52]:
```
randomforest.fit(X_train, y_train)
```

Out[52]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [53]:
```
randomforest_pred = randomforest.predict(X_test)
randomforest_pred
```

Out[53]:
```
array([2, 2, 0, ..., 2, 2, 2])
```

In [54]:
```
randomforest_accuracy = accuracy_score(y_test, randomforest_pred)
randomforest_precision = precision_score(y_test, randomforest_pred, average='weighted')
randomforest_recall = recall_score(y_test, randomforest_pred, average='weighted')
randomforest_f1 = f1_score(y_test, randomforest_pred, average='weighted')
randomforest_confusion = confusion_matrix(y_test, randomforest_pred)
```

In [63]:
```
print(randomforest_confusion)
```

```
[[ 417    0   33]
 [   0  277   12]
 [  10    2 1007]]
```

Decision Tress Classifier

In [56]:
```
decisiontree = DecisionTreeClassifier()
```

In [57]:
```
decisiontree.fit(X_train, y_train)
```

Out[57]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [58]:
```
decisiontree_pred = decisiontree.predict(X_test)
decisiontree_pred
```

Out[58]:
```
array([2, 2, 0, ..., 2, 2, 2])
```

In [59]:
```
decisiontree_accuracy = accuracy_score(y_test, decisiontree_pred)
decisiontree_precision = precision_score(y_test, decisiontree_pred, average='weighted')
decisiontree_recall = recall_score(y_test, decisiontree_pred, average='weighted')
decisiontree_f1 = f1_score(y_test, decisiontree_pred, average='weighted')
decisiontree_confusion = confusion_matrix(y_test, decisiontree_pred)
```

In [62]:
```
print(decisiontree_confusion)
```

```
[[ 450    0    0]
 [   0  289    0]
 [   0    0 1019]]
```

THE PERFORMANCE OF THE MODELS

In [61]:
```
model_performance = {
    "Model": ["Logistic Regression", "Random Forest", "Decision Tree"],
    "Accuracy": [logistic_accuracy, randomforest_accuracy, decisiontree_accuracy],
    "Precision": [logistic_precision, randomforest_precision, decisiontree_precision],
    "Recall": [logistic_recall, randomforest_recall, decisiontree_recall],
    "F1 Score": [logistic_f1, randomforest_f1, decisiontree_f1]
}

performance_df = pd.DataFrame(model_performance)

performance_df = performance_df.round(4)
performance_df
```

Out[61]:

	Model	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression	0.9716	0.9718	0.9716	0.9715
1	Random Forest	0.9676	0.9680	0.9676	0.9674
2	Decision Tree	1.0000	1.0000	1.0000	1.0000

Decison Tree CLassifier is the perfect model for the classification

In [ ]: