

```
In [86]: import pandas as pd
```

```
In [87]: df = pd.read_csv('fitness_dataset.csv')
```

```
In [88]: df.head()
```

```
Out[88]:
```

	age	height_cm	weight_kg	heart_rate	blood_pressure	sleep_hours	nutrition_quality	activity_index
0	56	152	65	69.6	117.0	NaN	2.37	1.0
1	69	186	95	60.8	114.8	7.5	8.77	1.5
2	46	192	103	61.4	116.4	NaN	8.20	1.2
3	32	189	83	60.2	130.1	7.0	6.18	1.8
4	60	175	99	58.1	115.8	8.0	9.95	1.1

```
In [89]: len(df)
```

```
Out[89]: 2000
```

```
In [90]: df.isnull().sum()
```

```
Out[90]: age                0
height_cm              0
weight_kg              0
heart_rate             0
blood_pressure         0
sleep_hours           160
nutrition_quality      0
activity_index         0
smokes                0
gender                 0
is_fit                0
dtype: int64
```

```
In [91]: missing_percentage = df.isnull().mean() * 100
columns_with_missing_30 = missing_percentage[missing_percentage > 30]
print("Columns with more than 30% missing values:")
print(columns_with_missing_30)
```

```
Columns with more than 30% missing values:
Series([], dtype: float64)
```

```
In [92]: df.describe()
```

Out[92]:

	age	height_cm	weight_kg	heart_rate	blood_pressure	sleep_hours	nu
count	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	1840.000000	
mean	49.114000	174.53300	83.540500	70.288600	119.908850	7.513315	
std	17.926564	14.37175	25.852534	11.846339	14.578032	1.502031	
min	18.000000	150.00000	30.000000	45.000000	90.000000	4.000000	
25%	34.000000	162.00000	64.000000	62.100000	109.700000	6.500000	
50%	49.000000	174.00000	83.000000	70.250000	120.000000	7.500000	
75%	65.000000	187.00000	102.000000	78.425000	129.800000	8.600000	
max	79.000000	199.00000	250.000000	118.600000	171.200000	12.000000	

In [93]: `print(df['sleep_hours'].mean())`

7.5133152173913045

In [94]: `print(df['sleep_hours'].median())`

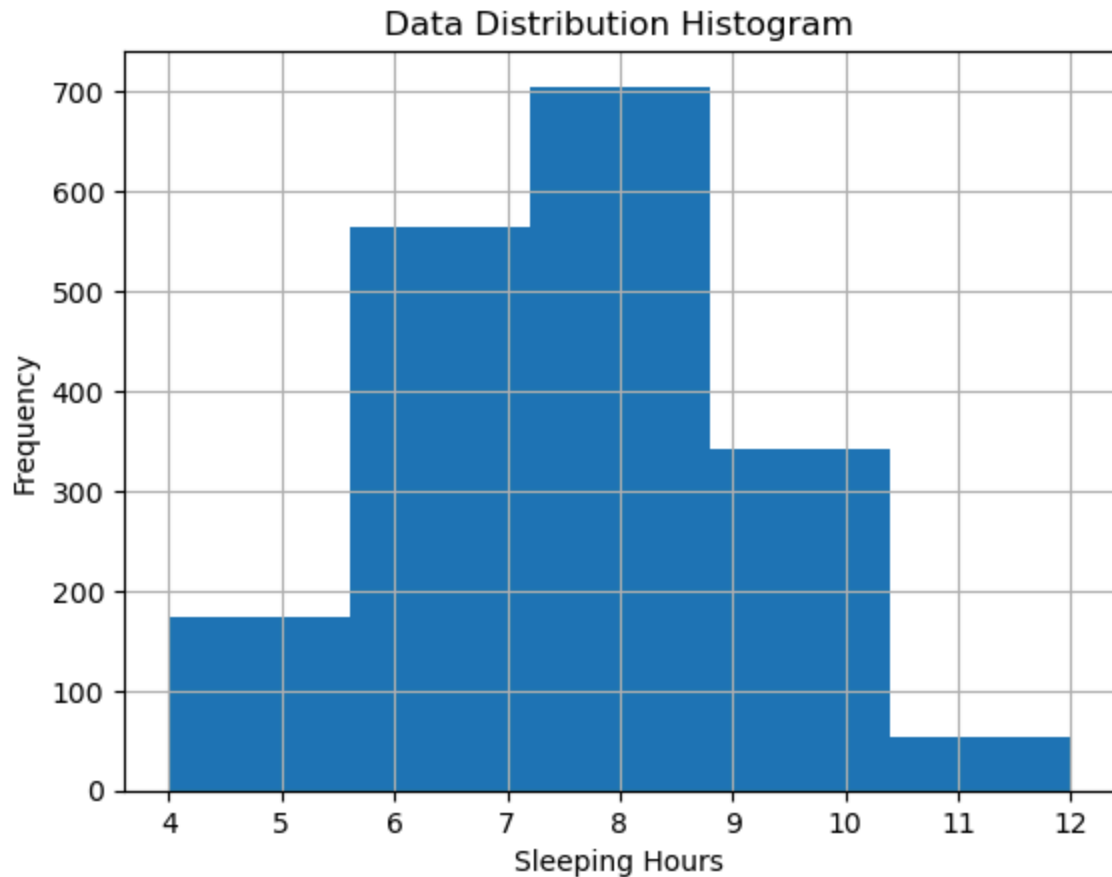
7.5

```
In [95]: import matplotlib.pyplot as plt

# Histogram for the column "sleep_hours"
df['sleep_hours'].hist(bins=5)

plt.title('Data Distribution Histogram')
plt.xlabel('Sleeping Hours')
plt.ylabel('Frequency')

plt.show()
```



```
In [96]: df['sleep_hours'] = df['sleep_hours'].fillna(df['sleep_hours'].mean())
```

```
In [97]: print(df['sleep_hours'].isna().sum())
```

0

```
In [98]: def detect_outliers_iqr(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = (column < lower_bound) | (column > upper_bound)
    return outliers

outlier_df = pd.DataFrame({
    'Column': data.select_dtypes(include='number').columns,
    'Has Outliers': ["Yes" if detect_outliers_iqr(data[col]).any() else "No"
                     for col in data.select_dtypes(include='number').columns]
})

print(outlier_df)
```

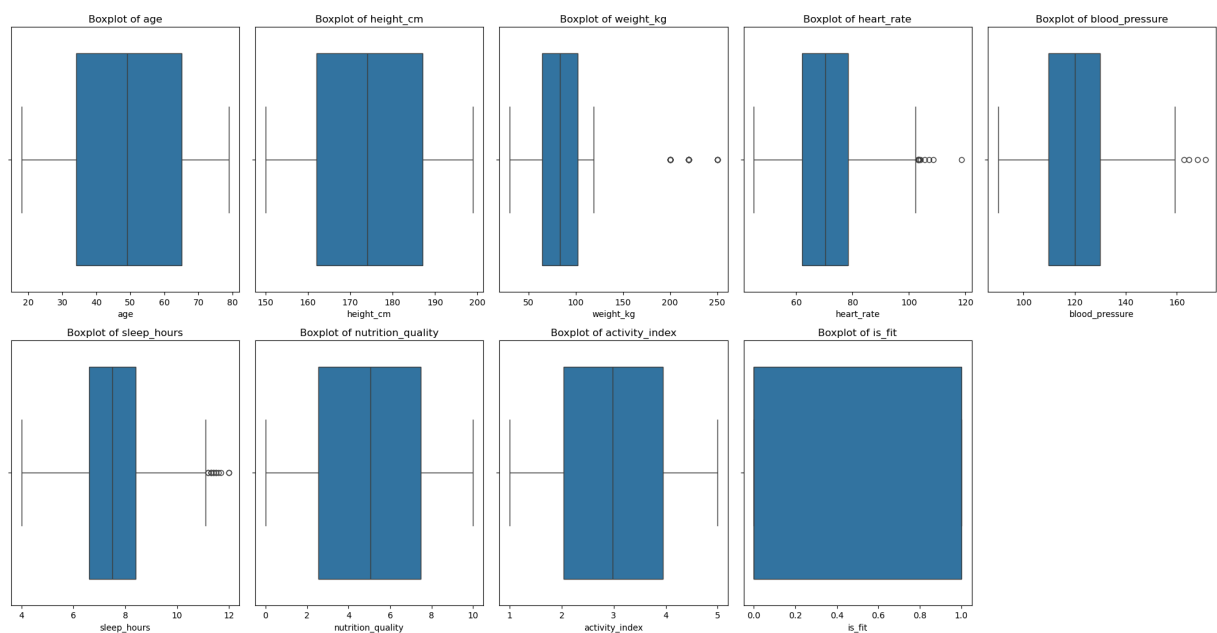
	Column	Has Outliers
0	age	No
1	height_cm	No
2	weight_kg	Yes
3	heart_rate	Yes
4	blood_pressure	Yes
5	sleep_hours	Yes
6	nutrition_quality	No
7	activity_index	No
8	is_fit	No

```
In [99]: numerical_df = df.select_dtypes(include=['number'])

plt.figure(figsize=(20, 5 * rows))

for i, col in enumerate(num_cols):
    plt.subplot(rows, 5, i + 1)    # plots per row
    sns.boxplot(x=numerical_df[col])
    plt.title(f"Boxplot of {col}")

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



```
In [100... data = df.copy()

def remove_outliers_iqr(data):

    # Select only numeric columns
    numeric_data = data.select_dtypes(include='number')

    # Calculate Q1 (25th percentile) and Q3 (75th percentile) for each feature
    Q1 = numeric_data.quantile(0.25)
    Q3 = numeric_data.quantile(0.75)
    IQR = Q3 - Q1

    # Define Lower and upper bounds
    lower_bound = Q1 - 1.5 * IQR
```

```

upper_bound = Q3 + 1.5 * IQR

# Remove rows with outliers in any numeric column
data_cleaned = numeric_data[~((numeric_data < lower_bound) | (numeric_data > up

return data_cleaned

cleaned_data = remove_outliers_iqr(data)

print("Original Data length:", len(data))
print("Cleaned Data length (without outliers):", len(cleaned_data))

```

Original Data length: 2000

Cleaned Data length (without outliers): 1953

In [101...

```

print("\n" + "="*60)
print("PART 2: VISUALIZATION")
print("="*60 + "\n")

```

```

=====
PART 2: VISUALIZATION
=====

```

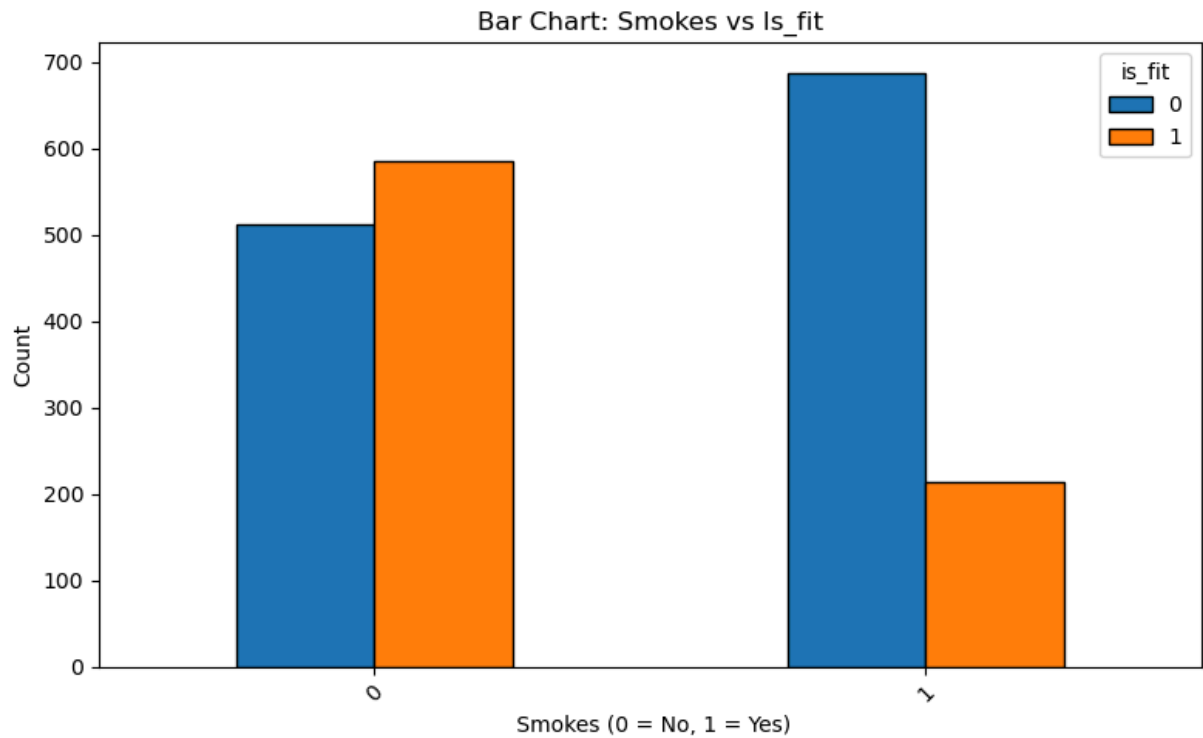
In [105...

```

# Count combinations of two columns
counts = df.groupby(['smokes', 'is_fit']).size().unstack(fill_value=0)

# Plot grouped bar chart
counts.plot(kind='bar', figsize=(8,5), edgecolor='black')
plt.title('Bar Chart: Smokes vs Is_fit')
plt.xlabel(' Smokes (0 = No, 1 = Yes)')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

```

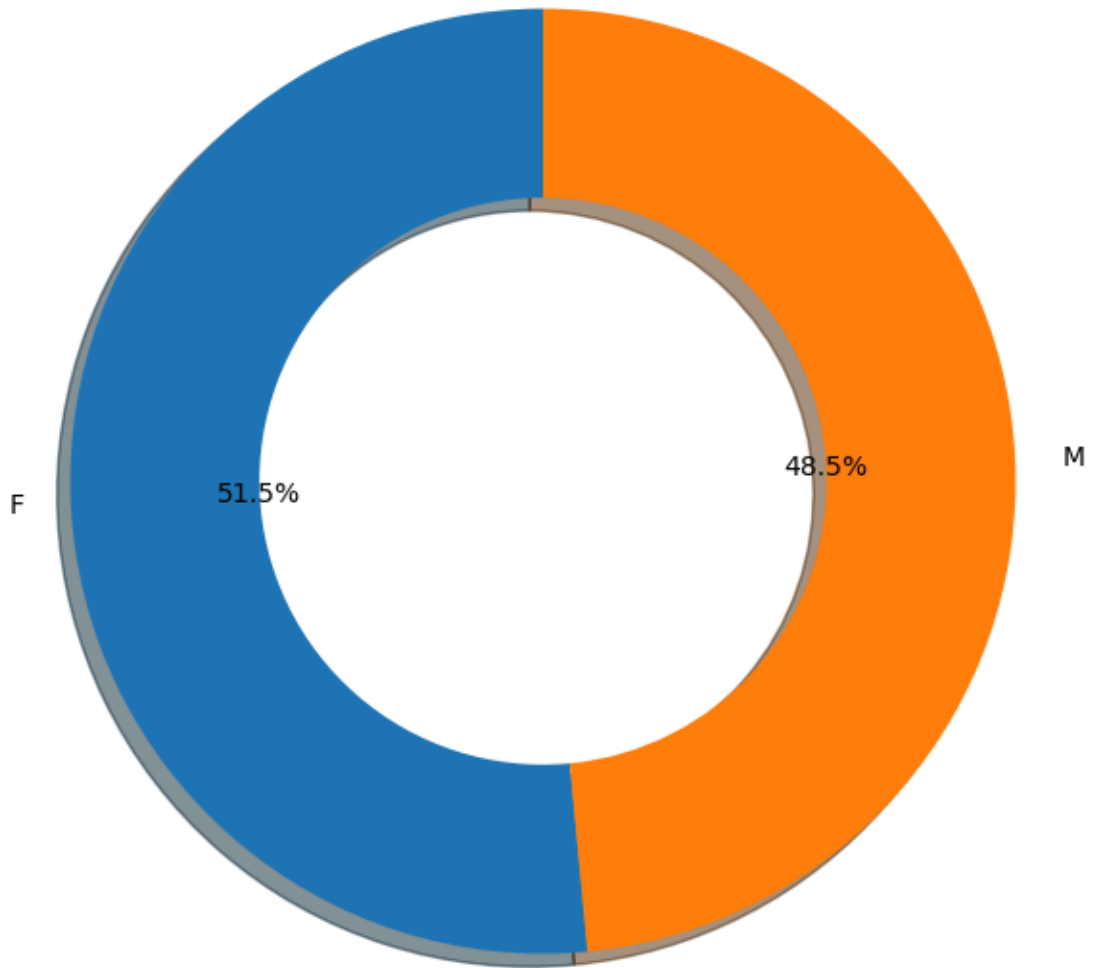


```
In [115... category_counts = df['gender'].value_counts()

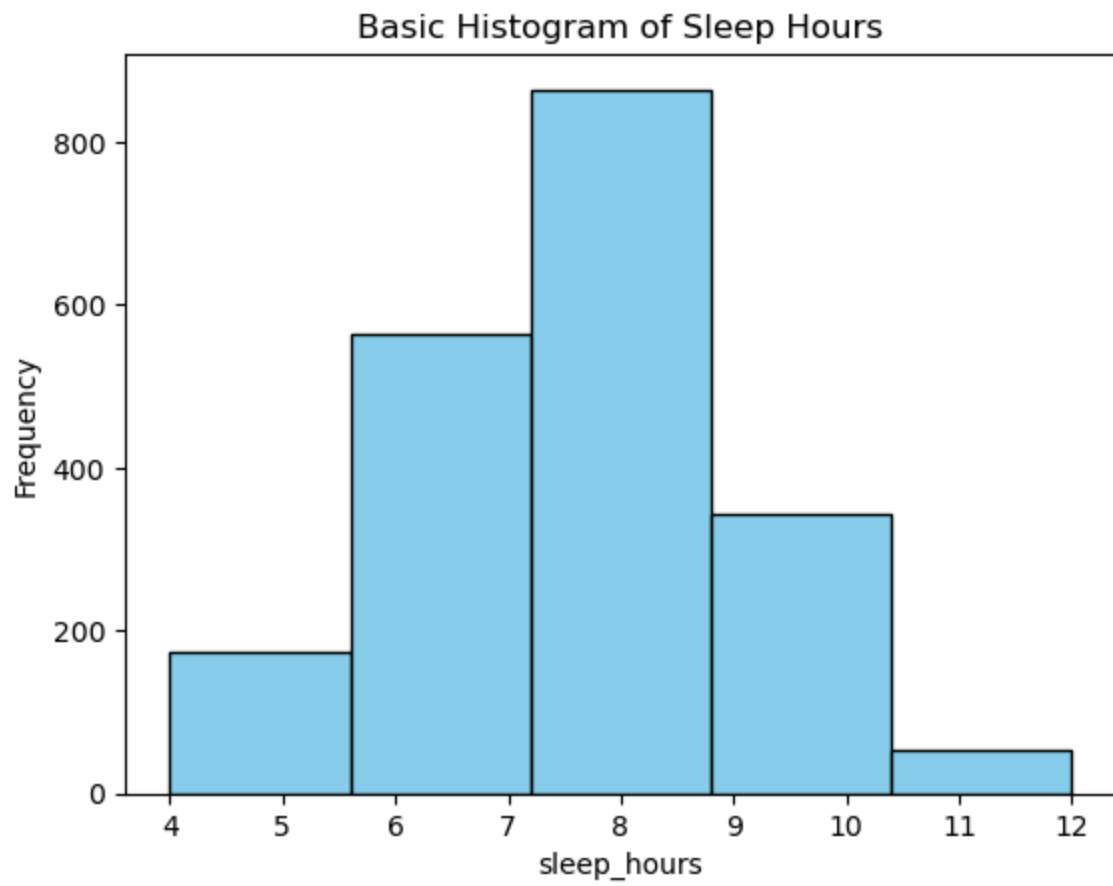
plt.figure(figsize=(8, 8)) # Optional: Adjust figure size

plt.pie(category_counts,
        labels=category_counts.index,
        autopct='%1.1f%%',
        startangle=90,
        shadow=True,
        wedgeprops={'width': 0.4}) # create the donut hole
plt.title('Distribution of Gender (Donut Chart)')
plt.show()
```

Distribution of Gender (Donut Chart)



```
In [126... plt.hist(data['sleep_hours'], bins=5, color='skyblue', edgecolor='black')
plt.title('Basic Histogram of Sleep Hours')
plt.xlabel('sleep_hours')
plt.ylabel('Frequency')
plt.show()
```



In []: