```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
```

```
# Load the dataset
data = pd.read_csv('BodyPerformance.csv')

# Display the first few rows to understand the structure
data.head()
```

	age	gender	height_cm	weight_kg	body fat_%	diastolic	systolic	gripForce	sit and bend forward_cm	sit-ups counts	broad jump_cm	class
0	27.0	M	172.3	75.24	21.3	80.0	130.0	54.9	18.4	60.0	217.0	C
1	25.0	M	165.0	55.80	15.7	77.0	126.0	36.4	16.3	53.0	229.0	A
2	31.0	M	179.6	78.00	20.1	92.0	152.0	44.8	12.0	49.0	181.0	C
3	32.0	M	174.5	71.10	18.4	76.0	147.0	41.4	15.2	53.0	219.0	B
4	28.0	M	173.8	67.70	17.1	70.0	127.0	43.5	27.1	45.0	217.0	B

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13393 entries, 0 to 13392
Data columns (total 12 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   age                      13393 non-null  float64
 1   gender                   13393 non-null  object 
 2   height_cm                13393 non-null  float64
 3   weight_kg                13393 non-null  float64
 4   body fat_%               13393 non-null  float64
 5   diastolic                13393 non-null  float64
 6   systolic                 13393 non-null  float64
 7   gripForce                13393 non-null  float64
 8   sit and bend forward_cm  13393 non-null  float64
 9   sit-ups counts           13393 non-null  float64
 10  broad jump_cm            13393 non-null  float64
 11  class                    13393 non-null  object 
dtypes: float64(10), object(2)
memory usage: 1.2+ MB
```

```
data['class'].unique()
```

```
array(['C', 'A', 'B', 'D'], dtype=object)
```

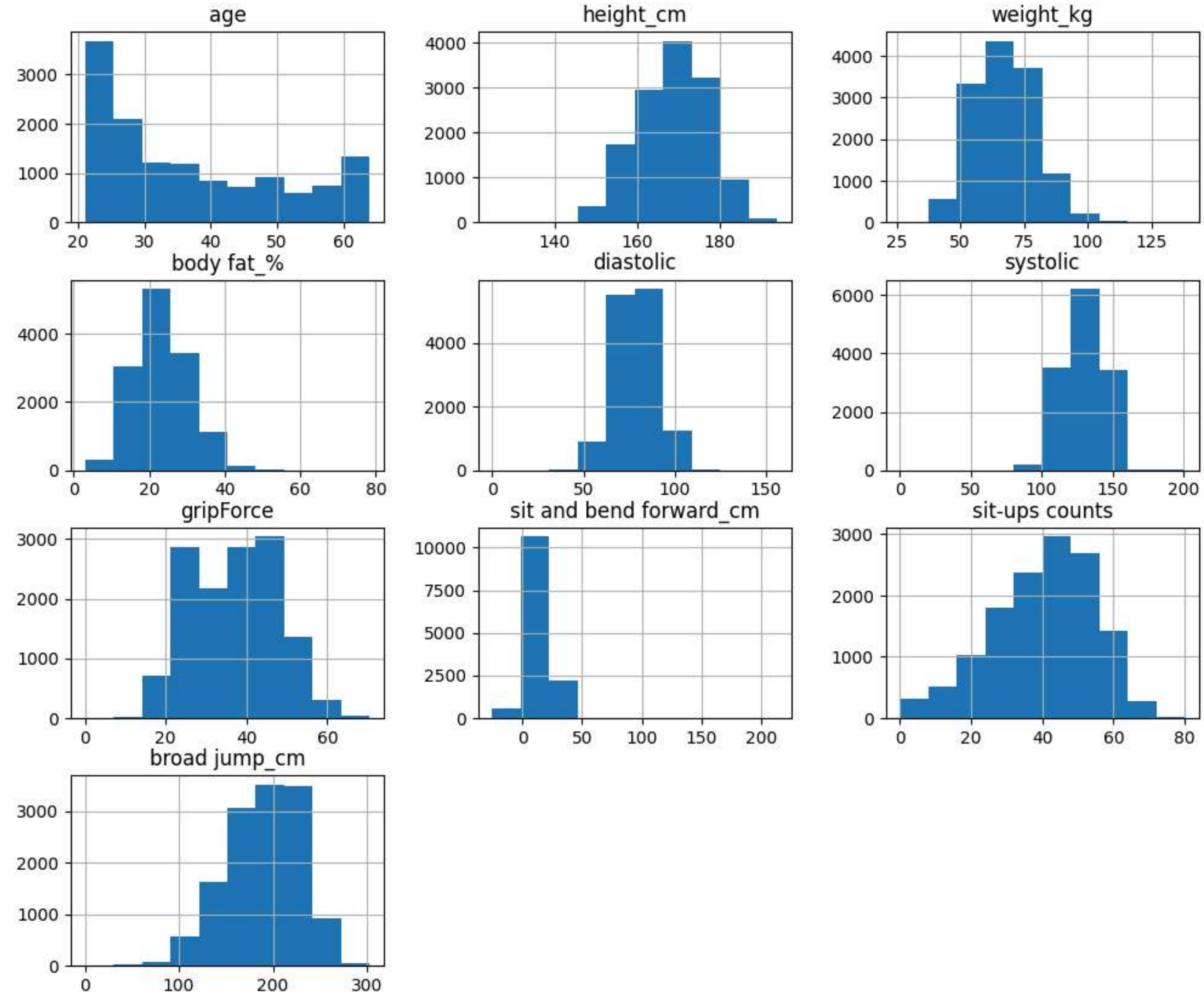
```
# Descriptive statistics for numerical features
data.describe()
```

	age	height_cm	weight_kg	body fat_%	diastolic	systolic	gripForce	sit and bend forward_cm	sit-ups counts	ju
count	13393.000000	13393.000000	13393.000000	13393.000000	13393.000000	13393.000000	13393.000000	13393.000000	13393.000000	13393.0
mean	36.775106	168.559807	67.447316	23.240165	78.796842	130.234817	36.963877	15.209268	39.771224	190.1
std	13.625639	8.426583	11.949666	7.256844	10.742033	14.713954	10.624864	8.456677	14.276698	39.8
min	21.000000	125.000000	26.300000	3.000000	0.000000	0.000000	0.000000	-25.000000	0.000000	0.0
25%	25.000000	162.400000	58.200000	18.000000	71.000000	120.000000	27.500000	10.900000	30.000000	162.0
50%	32.000000	169.200000	67.400000	22.800000	79.000000	130.000000	37.900000	16.200000	41.000000	193.0
75%	48.000000	174.800000	75.300000	28.000000	86.000000	141.000000	45.200000	20.700000	50.000000	221.0

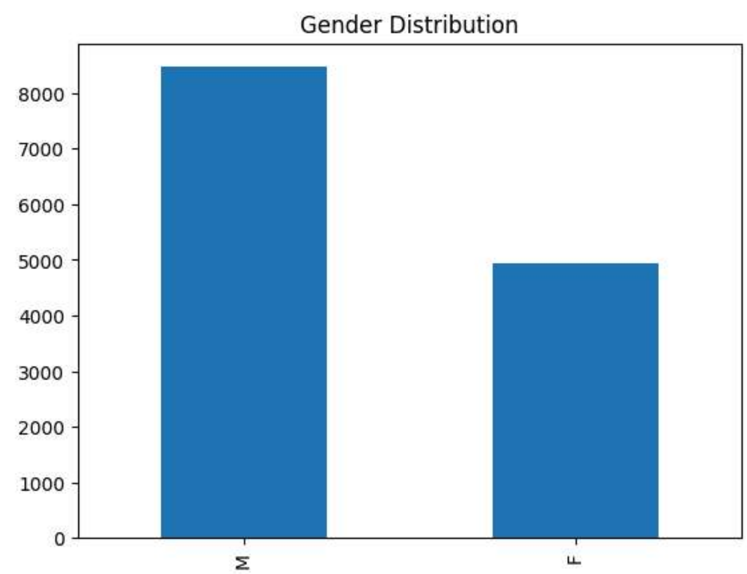
```
# Central Tendency: mean, median
central_tendency = data.mean()
print("Central Tendency (Mean):\n", central_tendency)

# Measure of Variance/Standard Deviation
print()
print()
variance = data.var()
std_deviation = data.std()
print("Variance:\n", variance)
print()
print()
print("Standard Deviation:\n", std_deviation)
```

```
Central Tendency (Mean):
 age                        36.775106
height_cm                 168.559807
weight_kg                  67.447316
body fat_%                 23.240165
diastolic                  78.796842
systolic                  130.234817
gripForce                  36.963877
sit and bend forward_cm    15.209268
sit-ups counts             39.771224
broad jump_cm             190.129627
dtype: float64


Variance:
 age                        185.658051
height_cm                   71.007293
weight_kg                  142.794526
body fat_%                  52.661786
diastolic                  115.391275
systolic                   216.500428
gripForce                  112.887736
sit and bend forward_cm     71.515386
sit-ups counts             203.824115
broad jump_cm             1589.457435
dtype: float64


Standard Deviation:
 age                        13.625639
height_cm                  8.426583
weight_kg                 11.949666
body fat_%                 7.256844
diastolic                 10.742033
systolic                  14.713954
gripForce                 10.624864
sit and bend forward_cm    8.456677
sit-ups counts            14.276698
broad jump_cm             39.868000
dtype: float64
C:\Users\Aumba\AppData\Local\Temp\ipykernel_4864\1681146331.py:2: FutureWarning: The default value of numeric_only in DataFrame.mean is
  central_tendency = data.mean()
C:\Users\Aumba\AppData\Local\Temp\ipykernel_4864\1681146331.py:8: FutureWarning: The default value of numeric_only in DataFrame.var is d
  variance = data.var()
C:\Users\Aumba\AppData\Local\Temp\ipykernel_4864\1681146331.py:9: FutureWarning: The default value of numeric_only in DataFrame.std is d
  std_deviation = data.std()
```

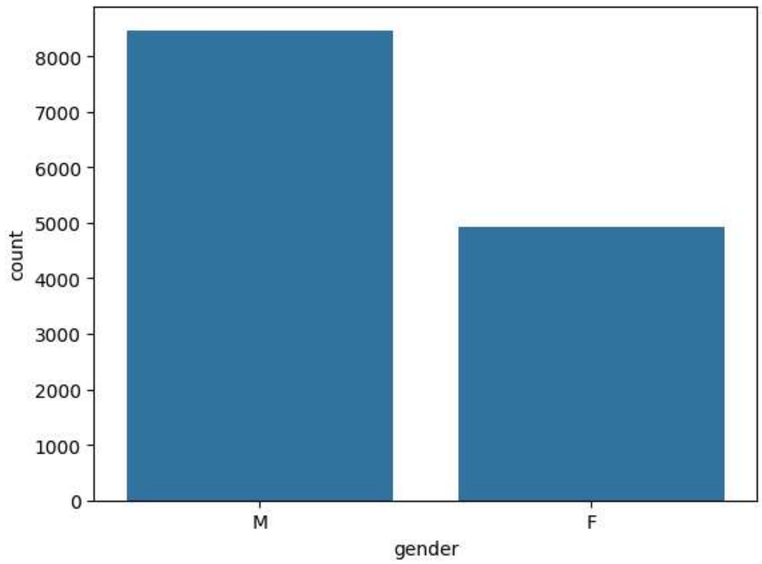
```
# Histogram for numerical data distribution
data.hist(figsize=(12, 10))
plt.show()
```

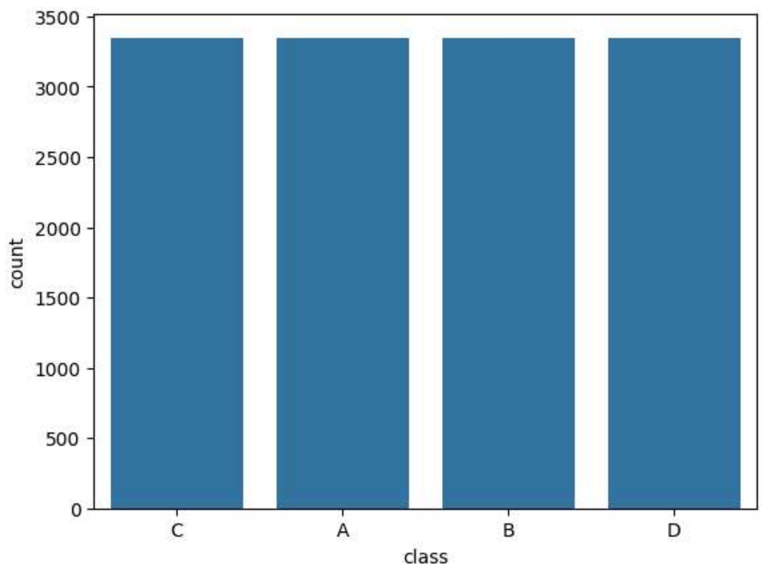



```
# Bar plot for categorical data
data['gender'].value_counts().plot(kind='bar')
plt.title('Gender Distribution')
plt.show()
```



```
# Categorical variables
sns.countplot(data=data, x='gender')
plt.show()
sns.countplot(data=data, x='class')
plt.show()
```





Confidence Interval of weight

```
confidence_level = 0.95
degrees_freedom = len(data['weight_kg']) - 1
sample_mean = data['weight_kg'].mean()
sample_standard_error = stats.sem(data['weight_kg'])

confidence_interval = stats.t.interval(confidence_level, degrees_freedom, sample_mean, sample_standard_error)
print("Confidence Interval for weight:", confidence_interval)
```

```
Confidence Interval for weight: (67.2449187070222, 67.6497128169082)
```

Confidence Interval for weight: (67.2449187070222, 67.6497128169082) meaning that there is 95% confidence that the true mean weight of the population is between 67.24 and 67.65 kilograms.

t_statistic for weight column

Null hypothesis: mean weight is 70 kg

alternative hypothesis: mean weight is not 70 kg

```
t_statistic, p_value = stats.ttest_1samp(data['weight_kg'], 70)
print("t-statistic:", t_statistic, "p-value:", p_value)
```

```
t-statistic: -24.721809301407955 p-value: 5.541607489679008e-132
```

t-statistic: -24.721809301407955, p-value: 5.541607489679008e-132, the mean weight is significantly different from 70 kg. The very low p-value provides strong evidence against the null hypothesis, supporting the alternative hypothesis that the true mean weight is different from 70 kg.

ANOVA test between different groups, e.g., weight across classs

null hypothesis: the weight is the same across the classes

alternative hypothesis: the weight is different across the classes

```
anova_results = stats.f_oneway(
    data[data["class"] == "A"]["weight_kg"],
    data[data["class"] == "B"]["weight_kg"],
    data[data["class"] == "C"]["weight_kg"],
    data[data["class"] == "D"]["weight_kg"],
)
print("ANOVA test results:", anova_results)
```

```
ANOVA test results: F_onewayResult(statistic=256.5613769545435, pvalue=6.70227429053531e-162)
```

THE ANOVA test results has an p-value of 6.70227429053531e-162 which is very low p-value, that means that the weight is significantly different across the classes and that is very low p-value.

Chi-Square Test for independence between two categorical variables

```
contingency_table = pd.crosstab(data['gender'], data['class'])
chi2_stat, chi2_p, chi2_dof, _ = stats.chi2_contingency(contingency_table)
print("Chi-Square Test results - Statistic:", chi2_stat, "P-value:", chi2_p)
```

```
Chi-Square Test results - Statistic: 112.77302615919672 P-value: 2.7764655088894975e-24
```

Chi-Square Test results - Statistic: 112.77302615919672 P-value: 2.7764655088894975e-24, given this values we can conclude there is a significant association between gender and class. The very low p-value strongly rejects the null hypothesis of independence, suggesting that gender and class are not independent and that there is a