```
# Import the google drive folders that contain the data
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
%cd /content/drive/MyDrive/DSC680/Weeks5-8/Week8/datasets/
```

```
/content/drive/MyDrive/DSC680/Weeks5-8/Week8/datasets
```

```
%ls
```

```
meets.csv             openpowerlifting_full-cleaned.csv  pml-training_full.csv
megaGymDataset.csv    openpowerlifting_short.csv
openpowerlifting.csv  pml-testing.csv
```

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

df = pd.read_csv('./openpowerlifting.csv', engine='python')
```

```
df.head()
```

	Name	Sex	Event	Equipment	Age	AgeClass	Division	BodyweightKg	Weigl
0	Abbie Murphy	F	SBD	Wraps	29.0	24-34	F-OR	59.8	
1	Abbie Tuong	F	SBD	Wraps	29.0	24-34	F-OR	58.5	
2	Ainslee Hooper	F	B	Raw	40.0	40-44	F-OR	55.4	
3	Amy Moldenhauer	F	SBD	Wraps	23.0	20-23	F-OR	60.0	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1423354 entries, 0 to 1423353
Data columns (total 37 columns):
 #   Column           Non-Null Count    Dtype
---  ------           --------------    -----
 0   Name             1423354 non-null  object
 1   Sex              1423354 non-null  object
 2   Event            1423354 non-null  object
 3   Equipment        1423354 non-null  object
 4   Age              757527 non-null   float64
 5   AgeClass         786800 non-null   object
 6   Division         1415176 non-null  object
 7   BodyweightKg     1406622 non-null  float64
 8   WeightClassKg    1410042 non-null  object
 9   Squat1Kg         337580 non-null   float64
 10  Squat2Kg         333349 non-null   float64
 11  Squat3Kg         323842 non-null   float64
 12  Squat4Kg         3696 non-null     float64
 13  Best3SquatKg     1031450 non-null  float64
 14  Bench1Kg         499779 non-null   float64
 15  Bench2Kg         493486 non-null   float64
 16  Bench3Kg         478485 non-null   float64
```



```
 20  Deadlift2Kg      356023 non-null   float64
 21  Deadlift3Kg      339947 non-null   float64
 22  Deadlift4Kg      9246 non-null     float64
 23  Best3DeadliftKg  1081808 non-null  float64
 24  TotalKg          1313184 non-null  float64
 25  Place            1423354 non-null  object
 26  Wilks            1304407 non-null  float64
 27  McCulloch        1304254 non-null  float64
 28  Glossbrenner     1304407 non-null  float64
 29  IPFPoints        1273286 non-null  float64
```

```
 30  Tested           1093892 non-null  object
 31  Country          388884 non-null   object
 32  Federation       1423354 non-null  object
 33  Date             1423354 non-null  object
 34  MeetCountry      1423354 non-null  object
 35  MeetState        941545 non-null   object
 36  MeetName         1423354 non-null  object
dtypes: float64(22), object(15)
memory usage: 401.8+ MB
```

```
cleaned_columns_df = df[['Name','Best3SquatKg','Best3BenchKg','Best3DeadliftKg',
                         'TotalKg','Sex','Equipment','BodyweightKg','Tested']]
```

```
cleaned_columns_df
```

	Name	Best3SquatKg	Best3BenchKg	Best3DeadliftKg	TotalKg	Sex	Equipme
0	Abbie Murphy	105.0	55.0	130.0	290.0	F	Wra
1	Abbie Tuong	120.0	67.5	145.0	332.5	F	Wra
2	Ainslee Hooper	NaN	32.5	NaN	32.5	F	Ra
3	Amy Moldenhauer	105.0	72.5	132.5	310.0	F	Wra
4	Andrea Rowan	140.0	80.0	170.0	390.0	F	Wra
...	...	...	...	...	...	...	
1423349	Marian Cafalik	175.0	87.5	190.0	452.5	M	Ra
1423350	Marian Piwowarczyk	110.0	95.0	170.0	375.0	M	Ra

```
cleaned_columns_df['Equipment'].value_counts()
```

```
Single-ply    787141
Raw           467421
Wraps         103739
Multi-ply      65035
Straps            18
Name: Equipment, dtype: int64
```

```
filt = cleaned_columns_df['Equipment'] == 'Single-ply'
single_ply_df = cleaned_columns_df[filt].drop('Equipment', axis=1).reset_index(drop = True)
single_ply_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 787141 entries, 0 to 787140
Data columns (total 8 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   Name             787141 non-null  object
 1   Best3SquatKg     623177 non-null  float64
 2   Best3BenchKg     703565 non-null  float64
 3   Best3DeadliftKg  615585 non-null  float64
 4   TotalKg          705450 non-null  float64
 5   Sex              787141 non-null  object
 6   BodyweightKg     773417 non-null  float64
 7   Tested           728794 non-null  object
dtypes: float64(5), object(3)
memory usage: 48.0+ MB
```



```
hancing Drugs)
single_ply_df['Tested'].fillna('Enhanced', inplace = True)
single_ply_df['Tested'].replace('Yes', 'Natural', inplace = True)
single_ply_df['Tested'].value_counts()
```

```
Natural     728794
Enhanced     58347
Name: Tested, dtype: int64
```

```
compete_once_df = single_ply_df.groupby(['Name','Sex',
                  'Tested'])[['Best3SquatKg','Best3BenchKg'
```

```
                      'Best3DeadliftKg','TotalKg','BodyweightKg']].mean().reset_index()
compete_once_df = compete_once_df.dropna()
compete_once_df
```

	Name	Sex	Tested	Best3SquatKg	Best3BenchKg	Best3DeadliftKg	TotalK
0	A Abduzhabarov	M	Natural	155.000000	110.000000	170.000000	435.0000
1	A Akins	M	Natural	115.670000	90.720000	129.270000	335.6600
2	A Allmehat	M	Natural	165.000000	120.000000	170.000000	455.0000
3	A Almeida	F	Natural	45.000000	25.000000	75.000000	145.0000
4	A Ashwin	M	Natural	180.000000	95.000000	210.000000	485.0000
...	...	...	...	...	...	...	
221582	齊藤 蒼斗	M	Natural	220.000000	130.000000	260.000000	610.0000
221584	齋藤 怜馬	M	Natural	206.666667	118.333333	183.333333	508.3333
221585	齋藤 恵太	M	Natural	215.000000	185.000000	210.000000	610.0000
221586	齋藤 誠一郎	M	Natural	231.250000	140.000000	222.500000	593.7500
221587	齋藤誠一郎	M	Natural	225.000000	145.000000	215.000000	585.0000

```
compete_once_df['WeightClass'] = compete_once_df['BodyweightKg'].apply(float)

def weight_class(x):
    for i in range(10, 140, 10):
        if(x < i):
            return f"{str(i-10).zfill(3)} - {i} kg"
    return "130+ kg"

compete_once_df['WeightClass'] = compete_once_df['BodyweightKg'].apply( lambda x: weight_class(x))
```

```
<ipython-input-12-f25817810d92>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  compete_once_df['WeightClass'] = compete_once_df['BodyweightKg'].apply(float)
```

```
compete_once_df
```

	Name	Sex	Tested	Best3SquatKg	Best3BenchKg	Best3DeadliftKg	
0	A Abduzhabarov	M	Natural	155.000000	110.000000	170.000000	43
1	A Akins	M	Natural	115.670000	90.720000	129.270000	33
2	A Allmehat	M	Natural	165.000000	120.000000	170.000000	45
3	A Almeida	F	Natural	45.000000	25.000000	75.000000	14
4	A Ashwin	M	Natural	180.000000	95.000000	210.000000	48
...	...	...	...	...	...	...	
221582	齊藤 蒼斗	M	Natural	220.000000	130.000000	260.000000	61
221584	齋藤 怜馬	M	Natural	206.666667	118.333333	183.333333	50
221585	齋藤 恵太	M	Natural	215.000000	185.000000	210.000000	61
						.500000	59
221587	齋藤誠一郎	M	Natural	225.000000	145.000000	215.000000	58

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
pivot_df = compete_once_df.groupby(['WeightClass','Tested','Sex'])['TotalKg'].count().reset_index()
pivot_df.pivot_table(columns=['Tested', 'Sex'], index=['WeightClass'], values='TotalKg')
```

Tested	Enhanced		Natural	
Sex	F	M	F	M
WeightClass				
020 - 30 kg	NaN	NaN	7.0	1.0
030 - 40 kg	1.0	NaN	137.0	68.0
040 - 50 kg	376.0	15.0	4935.0	1117.0
050 - 60 kg	1008.0	1182.0	12400.0	11476.0
060 - 70 kg	851.0	2406.0	9335.0	16923.0
070 - 80 kg	373.0	2640.0	5690.0	20839.0
080 - 90 kg	199.0	3247.0	3481.0	21479.0
090 - 100 kg	64.0	2888.0	1694.0	15617.0
100 - 110 kg	20.0	2504.0	887.0	11705.0
110 - 120 kg	11.0	1584.0	466.0	7247.0

```
filt = compete_once_df['WeightClass'].isin(['020 - 30 kg','030 - 40 kg', '120 - 130 kg', '130+ kg'])
compete_once_df.drop(index = compete_once_df[filt].index, inplace = True)

pivot_df = compete_once_df.groupby(['WeightClass','Tested','Sex'])['TotalKg'].count().reset_index()
pivot_df.pivot_table(columns=['Tested', 'Sex'], index=['WeightClass'], values='TotalKg')
```

Tested	Enhanced		Natural	
Sex	F	M	F	M
WeightClass				
040 - 50 kg	376	15	4935	1117
050 - 60 kg	1008	1182	12400	11476
060 - 70 kg	851	2406	9335	16923
070 - 80 kg	373	2640	5690	20839
080 - 90 kg	199	3247	3481	21479
090 - 100 kg	64	2888	1694	15617
100 - 110 kg	20	2504	887	11705
110 - 120 kg	11	1584	466	7247

```
clean_df = compete_once_df[compete_once_df['Tested'] == 'Natural'].drop('Tested', axis=1)
enhanced_df = compete_once_df[compete_once_df['Tested'] == 'Enhanced'].drop('Tested', axis=1)

lred, dred, lblue, dblue = ["#fb9a99", "#e31a1c", "#a6cee3", "#1f78b4"]

clean_df.groupby('Sex')['TotalKg'].count()
```

```
Sex
F     38888
M    106403
Name: TotalKg, dtype: int64
```

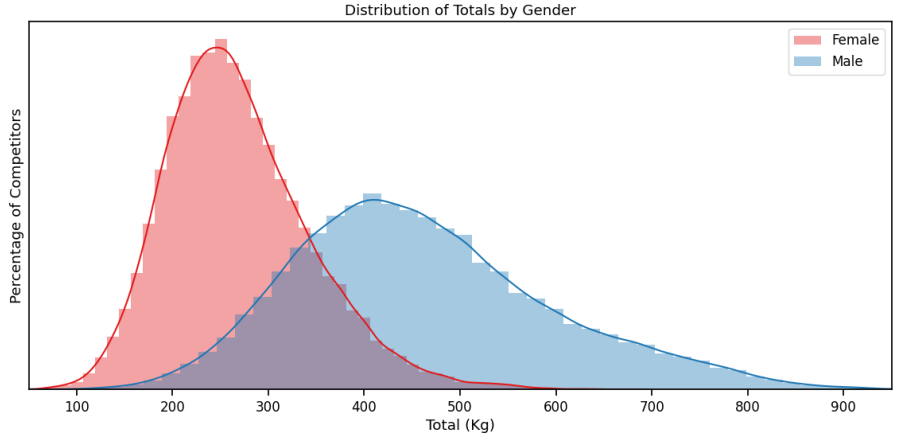
```
Male_series = clean_df[clean_df['Sex'] == 'M']['TotalKg']
Female_series = clean_df[clean_df['Sex'] == 'F']['TotalKg']

plt.close('all')
```



```
sns.distplot(Female_series, label ='Female', color = dred)
sns.distplot(Male_series, label ='Male', color = dblue)
plt.legend()

plt.title('Distribution of Totals by Gender')
plt.yticks([])
plt.xticks([0, 100, 200, 300, 400, 500, 600, 700, 800, 900])
plt.xlim(50, 950)
plt.xlabel('Total (Kg)')
```

```
plt.ylabel('Percentage of Competitors')
plt.show()
```

```
<ipython-input-17-0a4608c7d880>:8: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(Female_series, label ='Female', color = dred)
<ipython-input-17-0a4608c7d880>:9: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(Male_series, label ='Male', color = dblue)
```




```
female_mean = Female_series.mean()
male_mean = Male_series.mean()

Mean_male_v_females = stats.percentileofscore(Female_series, male_mean)
Mean_female_v_males = stats.percentileofscore(Male_series, female_mean)
print(f'Average Female Total: {round(female_mean, 1)} kg')
print(f'Average Male Total: {round(male_mean, 1)} kg')
print()
print(f'Average male > {round(Mean_male_v_females, 1)}% of females')
print(f'Average female > {round(Mean_female_v_males, 1)}% of males')
```



```
Average male > 98.3% of females
Average female > 5.3% of males
```

```
D = male_mean - female_mean

female_sd = Female_series.std()
male_sd = Male_series.std()
D_sd = (female_sd**2 + male_sd**2)**0.5
```

```
print(f'Difference: {round(D,1)} kg')
print()
print(f'Female Standard Deviation: {round(female_sd, 1)} kg')
print(f'Male Standard Deviation: {round(male_sd, 1)} kg')
print(f"Difference's Standard Deviation: {round(D_sd, 1)} kg")
```

```
Difference: 192.4 kg

Female Standard Deviation: 75.8 kg
Male Standard Deviation: 135.2 kg
Difference's Standard Deviation: 155.0 kg
```

```
z_score = D/D_sd
print(f"Z-score: {round(z_score, 2)}")
print(f"Probability: {round(stats.norm.cdf(z_score)*100,2)}%")
```

```
Z-score: 1.24
Probability: 89.28%
```

```
E_Male_series = enhanced_df[enhanced_df['Sex'] == 'M']['TotalKg']
E_Female_series = enhanced_df[enhanced_df['Sex'] == 'F']['TotalKg']

E_D = E_Male_series.mean() - E_Female_series.mean()

E_female_sd = E_Female_series.std()
E_male_sd = E_Male_series.std()
E_D_sd = (E_female_sd**2 + E_male_sd**2)**0.5

print(f'Enhanced Difference: {round(E_D,1)} kg')
print(f"Enhanced Difference's Standard Deviation: {round(E_D_sd, 1)} kg")
```

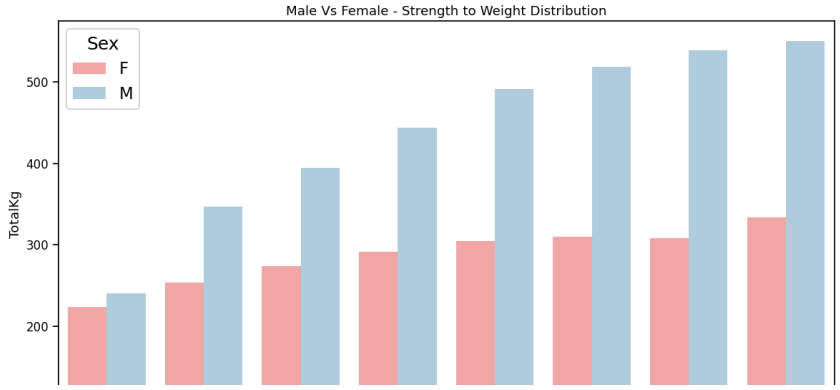
```
Enhanced Difference: 243.5 kg
Enhanced Difference's Standard Deviation: 172.4 kg
```

```
E_z_score = E_D/E_D_sd
print(f"Z-score: {round(E_z_score, 2)}")
print(f"Probability: {round(stats.norm.cdf(E_z_score)*100,2)}%")
```

```
Z-score: 1.41
Probability: 92.11%
```

```
plt.clf()
MvF_df = clean_df.groupby(['WeightClass', 'Sex']).TotalKg.mean().reset_index()
plt.figure(figsize = (14, 7))
sns.set_context("notebook", font_scale = 1.1)
sns.set_palette([lred, lblue])

plt.xticks(rotation =20)
plt.title('Male Vs Female - Strength to Weight Distribution')
sns.set_context("talk")

sns.barplot(data = MvF_df, x = 'WeightClass', y = 'TotalKg', hue = 'Sex')
plt.ylim(100, 575)
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```




```
s_score_df = clean_df.drop(['Best3SquatKg','Best3BenchKg','Best3DeadliftKg'], axis = 1)
s_score_df['T/BW'] = s_score_df['TotalKg']/s_score_df['BodyweightKg']

wc_s_score_df = s_score_df.groupby(['WeightClass'])['T/BW'].mean().reset_index()
wc_s_score_df.columns = ['WeightClass', 'WC-T/BW']

s_score_df = s_score_df.merge(wc_s_score_df, how='left')
s_score_df['S_score'] = s_score_df['T/BW']/s_score_df['WC-T/BW']
s_score_df.drop(columns = ['T/BW','WC-T/BW'], inplace = True)
s_score_df
```

	Name	Sex	TotalKg	BodyweightKg	WeightClass	S_score
0	A Abduzhabarov	M	435.000000	74.000000	070 - 80 kg	1.069188
1	A Akins	M	335.660000	107.050000	100 - 110 kg	0.625936
2	A Allmehat	M	455.000000	72.500000	070 - 80 kg	1.141484
3	A Almeida	F	145.000000	44.000000	040 - 50 kg	0.676234
4	A Ashwin	M	485.000000	81.700000	080 - 90 kg	1.078988
...	...	...	...	...	...	...
145286	齊藤 蒼斗	M	610.000000	73.700000	070 - 80 kg	1.505424
145287	齋藤 怜馬	M	508.333333	91.493333	090 - 100 kg	1.054109
145288	齋藤 恵太	M	610.000000	86.100000	080 - 90 kg	1.287726
145289	齋藤 誠一郎	M	593.750000	92.125000	090 - 100 kg	1.222792
145290	齋藤誠一郎	M	585.000000	103.000000	100 - 110 kg	1.133798

145291 rows × 6 columns

```
E_Male_series = s_score_df[s_score_df['Sex'] == 'M']['S_score']
E_Female_series = s_score_df[s_score_df['Sex'] == 'F']['S_score']

E_D = E_Male_series.mean() - E_Female_series.mean()

E_female_sd = E_Female_series.std()
E_male_sd = E_Male_series.std()
E_D_sd = (E_female_sd**2 + E_male_sd**2)**0.5

print(f'S_score Difference: {round(E_D,2)*100}%')
print(f"S_score Difference's SD: {round(E_D_sd, 2)*100}%")
```

```
S_score Difference: 27.0%
S_score Difference's SD: 36.0%
```



```
print(f"Z-score: {round(E_z_score, 2)}")
print(f"Probability: {round(stats.norm.cdf(E_z_score)*100,2)}%")
```

```
Z-score: 0.77
Probability: 77.88%
```

```
resp_vars = ['Best3SquatKg','Best3BenchKg','Best3DeadliftKg','TotalKg']
MG_averages_df = compete_once_df.groupby(['Tested','Sex'])[resp_vars].mean().reset_index()
MG_averages_df
```

	Tested	Sex	Best3SquatKg	Best3BenchKg	Best3DeadliftKg	TotalKg
0	Enhanced	F	116.269102	63.621701	133.124515	304.769231
1	Enhanced	M	207.970837	137.160266	220.532362	548.301604
2	Natural	F	106.814354	55.189425	115.055113	269.082502
3	Natural	M	177.627802	112.787898	187.061009	461.499752

```
print("Female average strength as a proportion of male's:")
print(f"Upper body(bench press): {round(55.2/112.8 *100, 1)}%")
print(f"Lower body(squat): {round(106.8/177.6 *100, 1)}%")
print(f"Back(deadlift): {round(115.1/187.1 *100, 1)}%")
```

```
Female average strength as a proportion of male's:
Upper body(bench press): 48.9%
Lower body(squat): 60.1%
Back(deadlift): 61.5%
```

```
MG_percents_df = MG_averages_df[['Tested','Sex']].copy()

for var in resp_vars:
    MG_percents_df[var] = round(MG_averages_df[var]/MG_averages_df['TotalKg']*100, 1)

MG_percents_df['Tested-Sex'] = MG_percents_df['Sex'] + ' - ' + MG_percents_df['Tested']
MG_percents_df.drop(['TotalKg', 'Sex', 'Tested'], axis = 1, inplace=True)
MG_percents_df.columns = ['Squat/legs', 'Bench/chest', 'Deadlift/back', 'Tested-Sex']
MG_percents_df
```

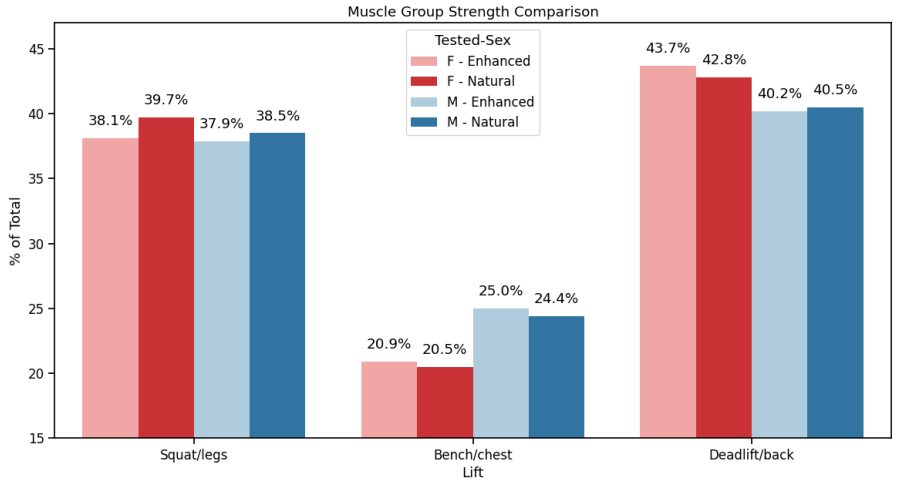
	Squat/legs	Bench/chest	Deadlift/back	Tested-Sex
0	38.1	20.9	43.7	F - Enhanced
1	37.9	25.0	40.2	M - Enhanced
2	39.7	20.5	42.8	F - Natural
3	38.5	24.4	40.5	M - Natural

```
MG_percents_df = pd.melt(frame=MG_percents_df, id_vars = ['Tested-Sex'], value_vars = ['Squat/legs',
    'Bench/chest', 'Deadlift/back'], value_name ="% of Total", var_name = 'Lift')
MG_percents_df = MG_percents_df.sort_values(['Tested-Sex'])
MG_percents_df
```

	Tested-Sex	Lift	% of Total
0	F - Enhanced	Squat/legs	38.1
4	F - Enhanced	Bench/chest	20.9
8	F - Enhanced	Deadlift/back	43.7
2	F - Natural	Squat/legs	39.7
6	F - Natural	Bench/chest	20.5
10	F - Natural	Deadlift/back	42.8
1	M - Enhanced	Squat/legs	37.9
5	M - Enhanced	Bench/chest	25.0
9	M - Enhanced	Deadlift/back	40.2
7	M - Natural	Bench/chest	24.4
11	M - Natural	Deadlift/back	40.5


```
plt.close('all')

def display_figures(ax,df):
    show=df['% of Total'].to_list()
    i=0
```

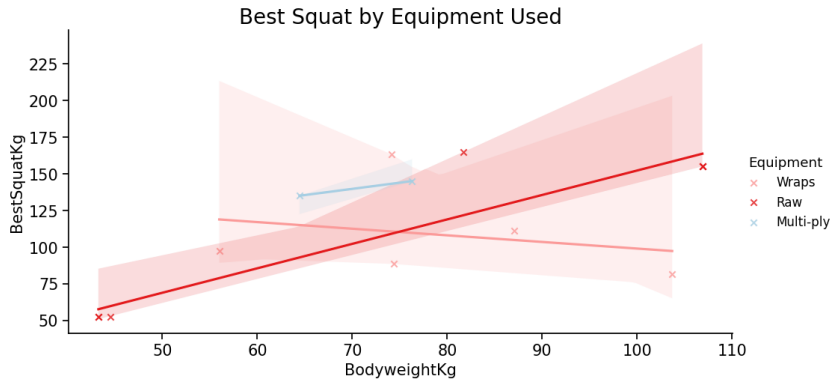
```
    for p in ax.patches:
        h=p.get_height()
        if (h>0):
            value= str(show[i])+"%"
            ax.text(p.get_x()+p.get_width()/2,h+1, value, ha='center')
            i=i+1

plt.figure(figsize = (14, 7))
sns.set_context("notebook", font_scale = 1.1)
sns.set_palette([lred, dred, lblue, dblue])

ax = sns.barplot(data = MG_percents_df, x = 'Lift', y = '% of Total', hue ='Tested-Sex')
plt.ylim(15, 47)
plt.title('Muscle Group Strength Comparison')
# plt.yticks([15,25,35,45])
display_figures(ax, MG_percents_df)

plt.show()
```




```
sns.lmplot(x='BodyweightKg',
           y='Best3SquatKg',
           data=df.dropna(),
           hue='Equipment',
           markers='x',
           aspect=2)
plt.title('Best Squat by Equipment Used',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.ylabel('BestSquatKg',fontsize=15)
plt.xlabel('BodyweightKg',fontsize=15)
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
print(df['Equipment'].dropna().value_counts())
```




```
Equipment Used by Lifters:

Single-ply    787141
Raw           467421
Wraps         103739
Multi-ply      65035
Straps            18
Name: Equipment, dtype: int64
```

✓ 8s completed at 4:25 PM

Automatic saving failed. This file was updated remotely or in another tab. Show diff