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

```
Mounted at /content/drive
```

```
%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 numpy as np
import pandas as pd
import plotly.express as px
```

```
gym_data = pd.read_csv('./megaGymDataset.csv')
gym_data.columns = gym_data.columns.str.replace('Unnamed: 0', 'index')
gym_data
```

	index	Title	Desc	Type	BodyPart	Equipment	Level	Rating
0	0	Partner plank band row	The partner plank band row is an abdominal exe...	Strength	Abdominals	Bands	Intermediate	0.0
1	1	Banded crunch isometric hold	The banded crunch isometric hold is an exercis...	Strength	Abdominals	Bands	Intermediate	NaN
2	2	FYR Banded Plank Jack	The banded plank jack is a variation on the pl...	Strength	Abdominals	Bands	Intermediate	NaN
			The					

```
missing_values_count = gym_data.isnull().sum()
missing_values_count
```

```
index            0
Title            0
Desc          1550
Type             0
BodyPart         0
Equipment        0
Level            0
Rating        1887
RatingDesc    2056
dtype: int64
```

```
gym_data.dtypes
```

```
index           int64
Title          object
Desc           object
Type           object
BodyPart       object
Equipment      object
Level          object
Rating        float64
RatingDesc     object
dtype: object
```

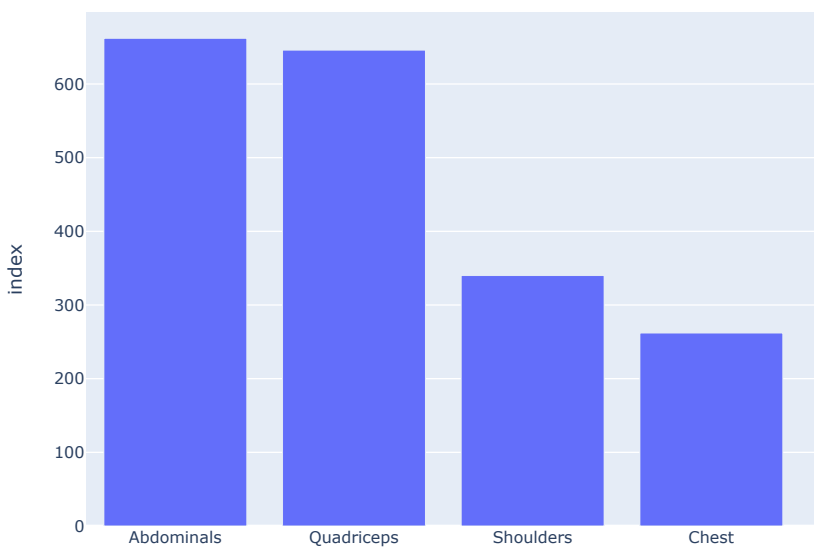
```
print("Row count:\t" + str(gym_data.shape[0]))
print("Col count:\t" + str(gym_data.shape[1]))
```

```
Row count:      2918
Col count:      9
```

```
count_exercises = gym_data.groupby(['BodyPart']).count()
count_exercises
```

BodyPart	index	Title	Desc	Type	Equipment	Level	Rating	RatingDesc
Abdominals	662	662	298	662	662	662	112	94
Abductors	21	21	8	21	21	21	10	9
Adductors	17	17	7	17	17	17	13	10
Biceps	168	168	101	168	168	168	55	53
Calves	47	47	26	47	47	47	26	26
Chest	262	262	149	262	262	262	113	90
Forearms	31	31	16	31	31	31	24	23
Glutes	81	81	29	81	81	81	25	22
Hamstrings	121	121	50	121	121	121	85	71
Lats	124	124	69	124	124	124	45	41
Lower Back	97	97	42	97	97	97	30	27
Middle Back	118	118	49	118	118	118	44	40
Neck	8	8	0	8	8	8	8	8
Quadriceps	646	646	245	646	646	646	218	155
Shoulders	340	340	174	340	340	340	141	118
Traps	24	24	17	24	24	24	16	16
Triceps	151	151	88	151	151	151	66	59

```
count_exercises= count_exercises.sort_values(by='index', ascending= False)
fig = px.bar(count_exercises, x=count_exercises.index, y='index')
fig.show()
```



```
beginner = gym_data[gym_data.Level == 'Beginner']
beginner
```

	index	Title	Desc	Type	BodyPart	Equipment	Level	Ratin
11	11	Bench barbell roll-out	The bench barbell roll-out is a challenging ex...	Strength	Abdominals	Barbell	Beginner	8.
12	12	Barbell Side Bend	NaN	Strength	Abdominals	Barbell	Beginner	7.
20	20	Advanced Kettlebell Windmill	NaN	Strength	Abdominals	Kettlebells	Beginner	8.
22	22	Kettlebell Pass Between The Legs	NaN	Strength	Abdominals	Kettlebells	Beginner	7.
		Dumbbell	The dumbbell spell caster					

```
body_only =beginner[beginner.Equipment == 'Body Only']
body_only
```

	index	Title	Desc	Type	BodyPart	Equipment	Level	Rating
194	194	Gorilla Chin/Crunch	NaN	Strength	Abdominals	Body Only	Beginner	9.0
202	202	Crunch - Hands Overhead	NaN	Strength	Abdominals	Body Only	Beginner	8.6
204	204	Stomach Vacuum	NaN	Stretching	Abdominals	Body Only	Beginner	8.5
208	208	Butt-Ups	NaN	Strength	Abdominals	Body Only	Beginner	8.3
217	217	Janda Sit-Up	NaN	Strength	Abdominals	Body Only	Beginner	7.4
...	...	...	...	...	...	...	...	...
2175	2175	Slow Jog	NaN	Cardio	Quadriceps	Body Only	Beginner	0.0
2178	2178	Square Hop	NaN	Plyometrics	Quadriceps	Body Only	Beginner	0.0
		Seated						

```
beginner_bodyOnly = body_only.groupby(['BodyPart']).count()
beginner_bodyOnly= beginner_bodyOnly.sort_values(by='index')
fig = px.bar(beginner_bodyOnly, x=beginner_bodyOnly.index, y='index', color ='index')
fig.show()
```

```
bodyPart_dist=gym_data.groupby(['Type','BodyPart']).count()
bodyPart_dist
```

		index	Title	Desc	Equipment	Level	Rating	RatingDesc
Type	**BodyPart**							
Cardio	**Abdominals**	2	2	0	2	2	0	0
	Calves	1	1	0	1	1	0	0
	Chest	1	1	0	1	1	0	0
	Hamstrings	1	1	1	1	1	1	1
	Middle Back	1	1	1	1	1	1	1
...	**...**	...	...	...	...	...	...	...
Strongman	**Forearms**	2	2	1	2	2	2	2
	Hamstrings	1	1	0	1	1	1	1
	Lower Back	4	4	0	4	4	4	4
	Quadriceps	10	10	3	10	10	10	8
	Shoulders	4	4	0	4	4	4	4

65 rows × 7 columns

```
bodyPart_dist=bodyPart_dist.groupby(level=[0,1]).sum()
bodyPart_dist
```

		index	Title	Desc	Equipment	Level	Rating	RatingDesc
Type	**BodyPart**							
Cardio	**Abdominals**	2	2	0	2	2	0	0
	Calves	1	1	0	1	1	0	0
	Chest	1	1	0	1	1	0	0
	Hamstrings	1	1	1	1	1	1	1
	Middle Back	1	1	1	1	1	1	1
...	**...**	...	...	...	...	...	...	...
Strongman	**Forearms**	2	2	1	2	2	2	2
	Hamstrings	1	1	0	1	1	1	1
	Lower Back	4	4	0	4	4	4	4
	Quadriceps	10	10	3	10	10	10	8
	Shoulders	4	4	0	4	4	4	4

65 rows × 7 columns

```
allTypes =(gym_data["Type"].unique())
len(allTypes)
```

```
7
```

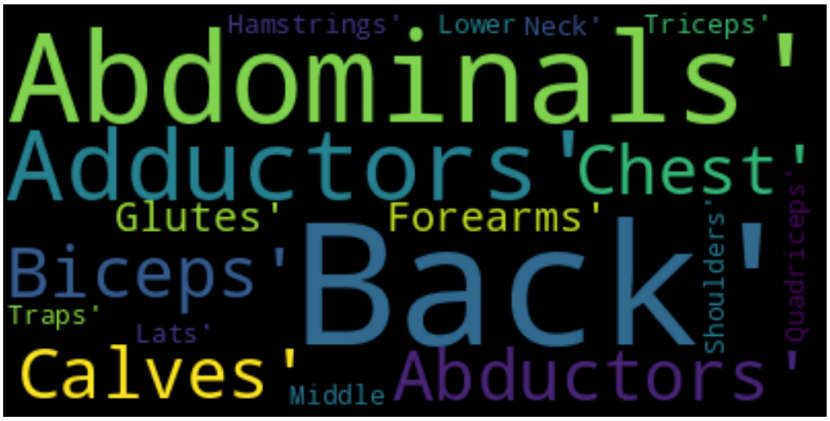
```
typeDfs = []
for i in range(7):
    typeDfs.append(bodyPart_dist.iloc[bodyPart_dist.index.get_level_values('Type') == allTypes[i]])
```

```
cardio =bodyPart_dist.iloc[bodyPart_dist.index.get_level_values('Type') == 'Cardio']
```

```
from plotly.subplots import make_subplots
import plotly.graph_objects as go

fig = make_subplots(
    rows=4, cols=2,
    specs=[[{"type": "domain"},{"type": "domain"}],
           [{"type": "domain"},{"type": "domain"}],
           [{"type": "domain"},{"type": "domain"}],
           [{"type": "domain"},{"type": "domain"}],
          ])

fig.add_trace(go.Pie(values=typeDfs[0]['index'].values, title=allTypes[0], labels=typeDfs[0].index,marker=dict(colors=['#100b','#f00560'], li
              row=1, col=1)

fig.add_trace(go.Pie(values=typeDfs[1]['index'].values, title=allTypes[1], labels=typeDfs[1].index,marker=dict(colors=['#100b','#f00560'], li
              row=1, col=2)
fig.add_trace(go.Pie(values=typeDfs[2]['index'].values, title=allTypes[2], labels=typeDfs[2].index,marker=dict(colors=['#100b','#f00560'], li
              row=2, col=1)
fig.add_trace(go.Pie(values=typeDfs[3]['index'].values, title=allTypes[3],labels=typeDfs[3].index,marker=dict(colors=['#100b','#f00560'], lin
              row=2, col=2)
fig.add_trace(go.Pie(values=typeDfs[4]['index'].values, title=allTypes[4], labels=typeDfs[4].index,marker=dict(colors=['#100b','#f00560'], li
              row=3, col=1)
fig.add_trace(go.Pie(values=typeDfs[5]['index'].values, title=allTypes[5], labels=typeDfs[5].index,marker=dict(colors=['#100b','#f00560'], li
              row=3, col=2)
fig.add_trace(go.Pie(values=typeDfs[6]['index'].values, title=allTypes[6], labels=typeDfs[6].index,marker=dict(colors=['#100b','#f00560'], li
              row=4, col=1)
fig.update_layout(height=900, showlegend=False)
fig.update_layout(height=1200, showlegend=False)
```

2
2.

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import seaborn as sns

text = gym_data["BodyPart"].unique()

wordcloud = WordCloud(max_words=1000000,background_color="black").generate(str(text))
plt.rcParams['figure.figsize'] = (13, 13)
plt.imshow(wordcloud,interpolation="bilinear")
plt.axis("off")
plt.show()
```

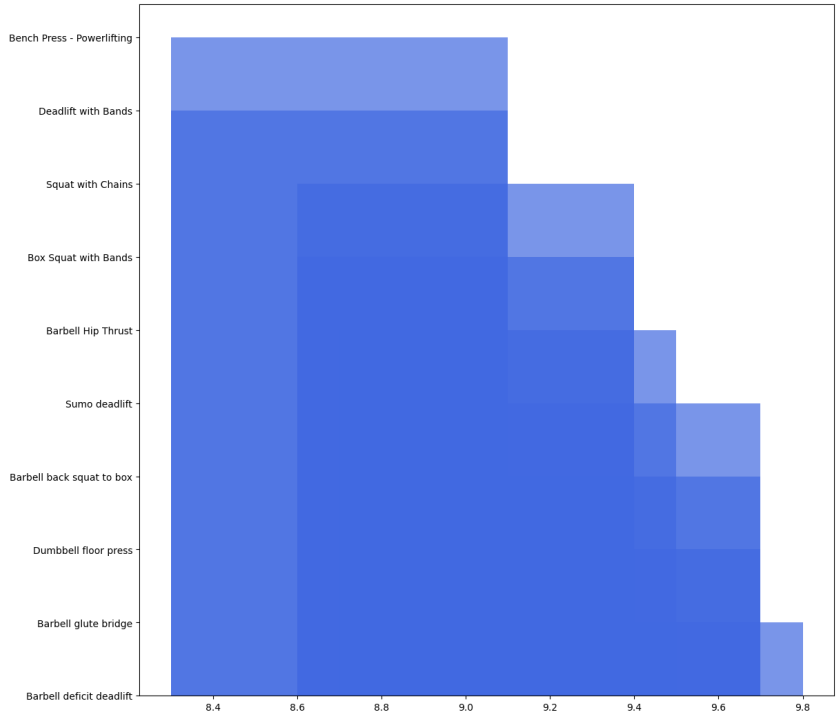



```
ratingSorted= gym_data.sort_values(by='Rating',ascending=False)
ratingSorted =ratingSorted.head(10)
ratingSorted
```

index	Title	Desc	Type	BodyPart	Equipment	Level	Ra
		The					

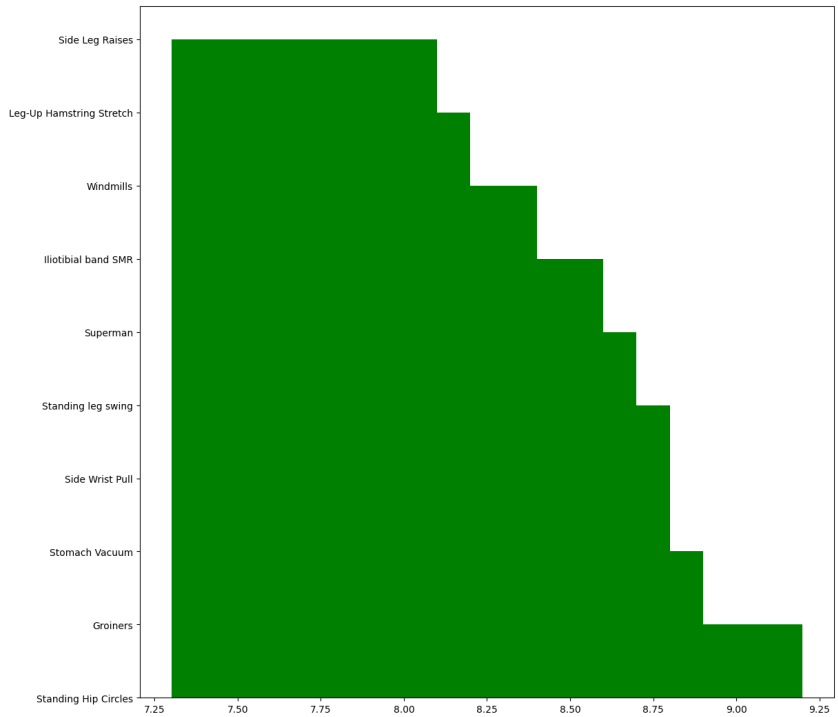
```
ratingSorted= gym_data[gym_data['Type']=='Powerlifting'].sort_values(by='Rating',ascending=False)
ratingSorted =ratingSorted.head(10)
plt.bar(ratingSorted.Rating, ratingSorted.Title,color='royalblue', alpha=0.7)
```

```
<BarContainer object of 10 artists>
```



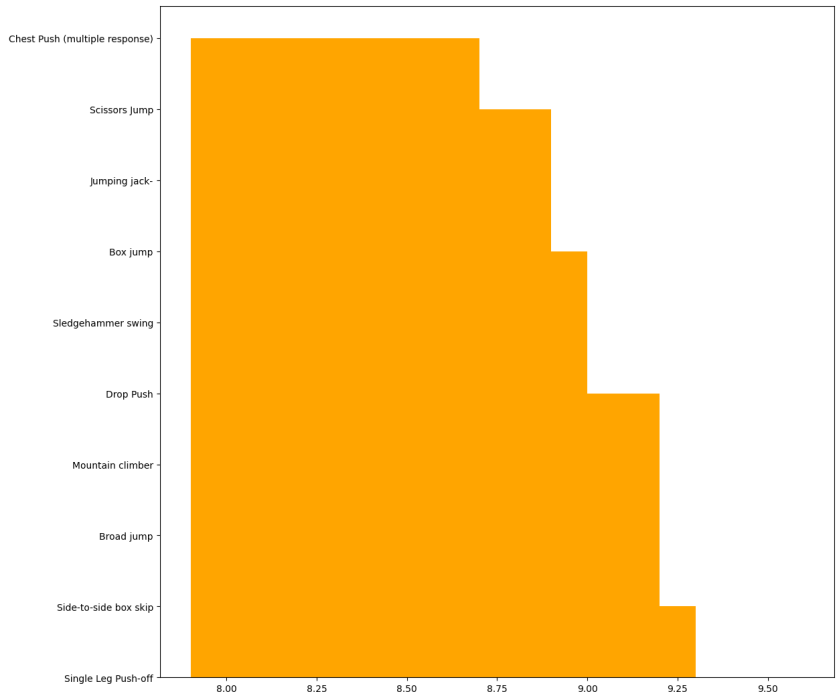

```
rating_stretch= gym_data[gym_data['Type']=='Stretching'].sort_values(by='Rating',ascending=False)
rating_stretch =rating_stretch.head(10)
plt.bar(rating_stretch.Rating, rating_stretch.Title,color='green')
```

```
<BarContainer object of 10 artists>
```




```
rating_plyo= gym_data[gym_data['Type']=='Plyometrics'].sort_values(by='Rating',ascending=False)
rating_plyo =rating_plyo.head(10)
plt.bar(rating_plyo.Rating, rating_plyo.Title,color='orange')
```

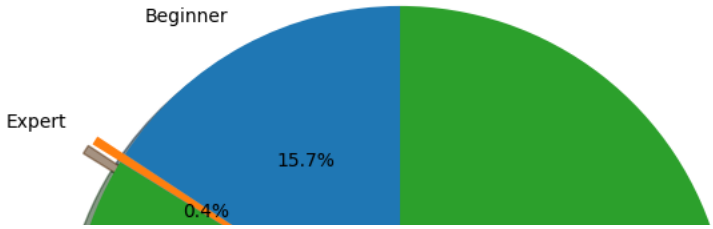
```
<BarContainer object of 10 artists>
```




```
level_counts=gym_data.groupby(['Level']).count()
level_counts
```

Level	index	Title	Desc	Type	BodyPart	Equipment	Rating	RatingDesc
Beginner	459	459	108	459	459	459	459	369
Expert	13	13	10	13	13	13	7	7
Intermediate	2446	2446	1250	2446	2446	2446	565	486

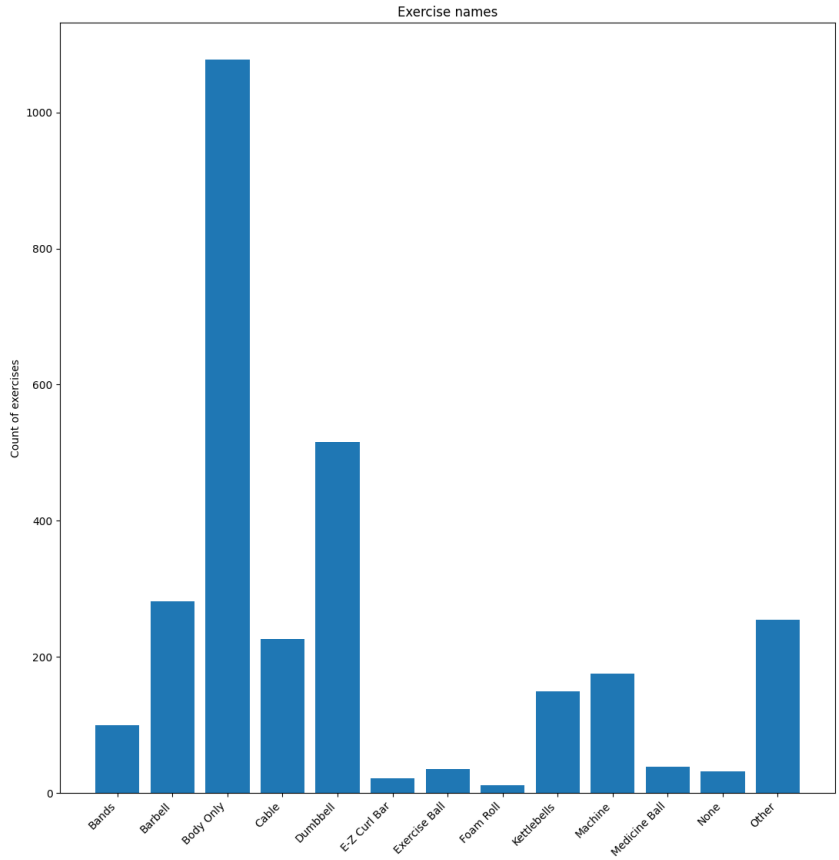
```
explode = (0, 0.1, 0)  # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots(figsize=(7, 7))
ax1.pie(level_counts.Title, explode=explode, labels=level_counts.index, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.figure(figsize=(1,1))

plt.show()
```




```
equip_data=gym_data.groupby(['Equipment']).count()
equip_data
```

Equipment	index	Title	Desc	Type	BodyPart	Level	Rating	RatingDesc
Bands	100	100	49	100	100	100	30	20
Barbell	282	282	161	282	282	282	180	168
Body Only	1078	1078	404	1078	1078	1078	269	196
Cable	226	226	149	226	226	226	77	77
Dumbbell	516	516	246	516	516	516	140	129
E-Z Curl Bar	22	22	14	22	22	22	11	11
Exercise Ball	35	35	28	35	35	35	12	11
Foam Roll	11	11	8	11	11	11	9	9
Kettlebells	149	149	53	149	149	149	57	39
Machine	175	175	121	175	175	175	67	65
Medicine Ball	38	38	25	38	38	38	21	11
None	32	32	9	32	32	32	29	28
Other	254	254	101	254	254	254	129	98

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

ax.bar(equip_data.index, equip_data.Title)

ax.set_ylabel('Count of exercises')
ax.set_title('Exercise names')
plt.xticks(rotation=45, ha='right')

plt.show()
```




```
leg_exer= gym_data[gym_data['BodyPart']=='Quadriceps'].sort_values(by='Rating',ascending=False)
leg_exer_best =leg_exer[leg_exer['Type']=='Strength'].head(5)
leg_exer_best
```

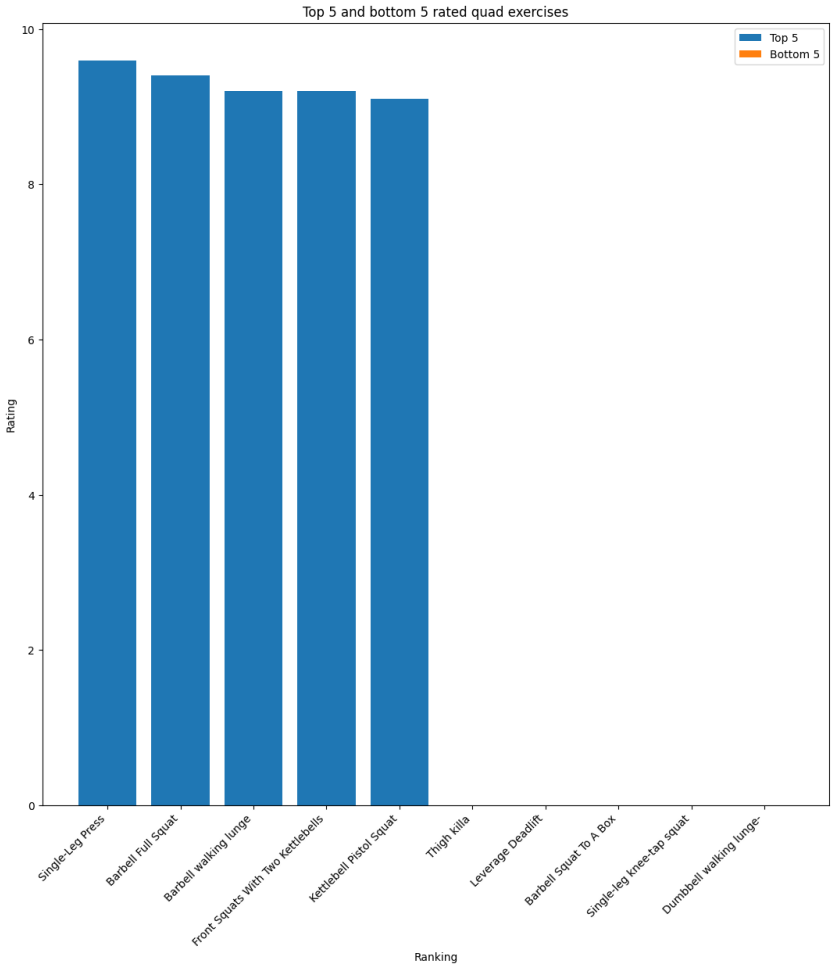
	index	Title	Desc	Type	BodyPart	Equipment	Level	Ratin
2065	2065	Single-Leg Press	The single-leg leg press is an exercise target...	Strength	Quadriceps	Machine	Intermediate	9.
			The					

```
leg_exer_worst= gym_data[gym_data['BodyPart']=='Quadriceps'].sort_values(by='Rating',ascending=True)
leg_exer_worst =leg_exer_worst[leg_exer_worst['Type']=='Strength'].head(5)
leg_exer_worst
```

	index	Title	Desc	Type	BodyPart	Equipment	Level	Ratin
2152	2152	Thigh killa	The thigh killa is a challenging lower-body bo...	Strength	Quadriceps	Body Only	Intermediate	0.
2083	2083	Leverage Deadlift	NaN	Strength	Quadriceps	Machine	Beginner	0.

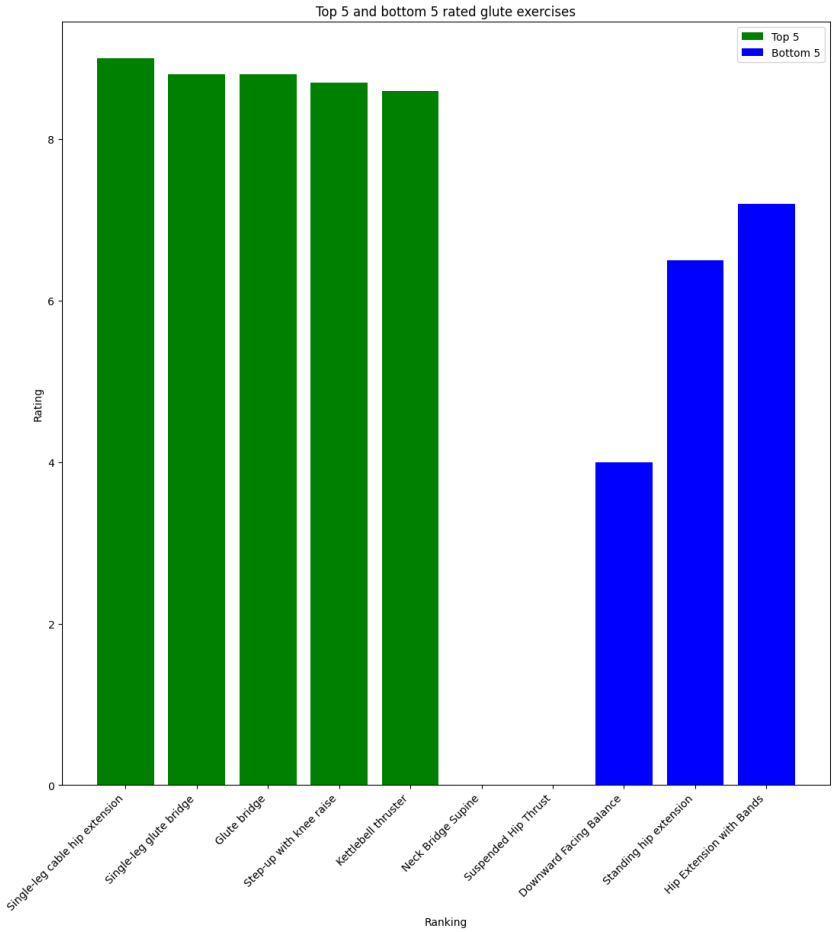
```
import numpy as np
import matplotlib.pyplot as plt

plt.bar(leg_exer_best.Title,leg_exer_best.Rating, label = 'Top 5')
plt.bar(leg_exer_worst.Title,leg_exer_worst.Rating, label = 'Bottom 5')
plt.xticks(rotation=45, ha='right')

plt.xlabel("Ranking")
plt.ylabel("Rating")
```

```
plt.title("Top 5 and bottom 5 rated quad exercises")
plt.legend()
plt.show()
```

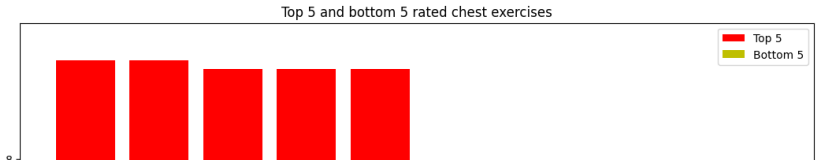



```
glute_exer= gym_data[gym_data['BodyPart']=='Glutes'].sort_values(by='Rating',ascending=False)
glute_exer_best =glute_exer[glute_exer['Type']=='Strength'].head(5)

glute_exer_worst= gym_data[gym_data['BodyPart']=='Glutes'].sort_values(by='Rating',ascending=True)
glute_exer_worst =glute_exer_worst[glute_exer_worst['Type']=='Strength'].head(5)
```

```
plt.bar(glute_exer_best.Title,glute_exer_best.Rating,color='g', label = 'Top 5')
plt.bar(glute_exer_worst.Title,glute_exer_worst.Rating,color='b', label = 'Bottom 5')
plt.xticks(rotation=45, ha='right')

plt.xlabel("Ranking")
plt.ylabel("Rating")
plt.title("Top 5 and bottom 5 rated glute exercises")
plt.legend()
plt.show()
```



```
chest_exer= gym_data[gym_data['BodyPart']=='Chest'].sort_values(by='Rating',ascending=False)
chest_exer_best =chest_exer[chest_exer['Type']=='Strength'].head(5)

chest_exer_worst= gym_data[gym_data['BodyPart']=='Chest'].sort_values(by='Rating',ascending=True)
chest_exer_worst =chest_exer_worst[chest_exer_worst['Type']=='Strength'].head(5)


plt.bar(chest_exer_best.Title,chest_exer_best.Rating,color='r', label = 'Top 5')
plt.bar(chest_exer_worst.Title,chest_exer_worst.Rating,color='y', label = 'Bottom 5')
plt.xticks(rotation=45, ha='right')

plt.xlabel("Ranking")
plt.ylabel("Rating")
plt.title("Top 5 and bottom 5 rated chest exercises")
plt.legend()
plt.show()
```




```
options = ['Lats','Lower Back','Middle Back']
back_exer= gym_data[gym_data['BodyPart'].isin(options)].sort_values(by='Rating',ascending=False)
back_exer_best =back_exer[back_exer['Type']=='Strength'].head(5)

back_exer_worst= gym_data[gym_data['BodyPart'].isin(options)].sort_values(by='Rating',ascending=True)
back_exer_worst =back_exer_worst[back_exer_worst['Type']=='Strength'].head(5)


plt.bar(back_exer_best.Title,back_exer_best.Rating,color='b', label = 'Top 5')
plt.bar(back_exer_worst.Title,back_exer_worst.Rating,color='m', label = 'Bottom 5')
plt.xticks(rotation=45, ha='right')

plt.xlabel("Ranking")
plt.ylabel("Rating")
plt.title("Top 5 and bottom 5 rated back exercises")
plt.legend()
plt.show()
```