2020 CrossFit Games Open

Author: Scott Graham

Data link - https://www.kaggle.com/datasets/jeanmidev/crossfit-games (https://www.kaggle.com/datasets/jeanmidev/crossfit-games)

Overview

The CrossFit Open is the largest single sporting competition held worldwide. The competition is held over a five week period with one workout per week to be submitted by participants.

The goal of this analysis is to view the data retrieved from the 2020 CrossFit Open and the athletes that have competed and determine if there are any standout features that make the top atheletes great including weight and height. The impact of age on performance.

In [1]:

```
# Import standard packages
import numpy as np
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
import scipy.stats as stats
from scipy import stats
from statsmodels.formula.api import ols
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from mpl_toolkits import mplot3d
import sklearn.metrics as metrics
import random
from math import sqrt
import seaborn as sns
plt.style.use('seaborn')
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
athlete_20 = pd.read_csv ('data/2020_opens_athletes.csv')
score_20 = pd.read_csv ('data/2020_opens_scores.csv')
```

Data Understanding

In [3]:

```
athlete_20.sort_values('competitorid').head()
```

Out[3]:

	competitorid	competitorname	firstname	lastname	postcompstatus	gender	profilep
274127	86	Justin Bergh	Justin	Bergh	NaN	M	3466c
361680	86	Justin Bergh	Justin	Bergh	NaN	M	3466c
203450	88	Cary Hair	Cary	Hair	NaN	M	c77af
346964	88	Cary Hair	Cary	Hair	NaN	M	c77af
369715	93	Leif Edmundson	Leif	Edmundson	NaN	M	13417

We will now drop unnessary columns.

In [4]:

```
athlete_20.drop(columns = ['firstname', 'lastname', 'profilepics3key', 'countryoforigincode
athlete_20.head()
```

Out[4]:

	competitorid	competitorname	postcompstatus	gender	countryoforiginname	divisionid	affil
0	9184	Janet Black	NaN	F	United States	4	
1	12348	Karen McCadam	NaN	F	United States	4	
2	288661	Christine Tensley	NaN	F	United States	4	F
3	37629	Heather Wood	NaN	F	Canada	4	Q
4	50423	Jennifer Dieter	NaN	F	United States	4	

In [5]:

```
athlete_20.isna().sum()
```

Out[5]:

```
competitorid               0
competitorname             0
postcompstatus        393222
gender                     0
countryoforiginname        0
divisionid                 0
affiliatename          21994
age                        0
height                181473
weight                171993
overallrank                0
overallscore               0
is_scaled                  0
division                   0
dtype: int64
```

Check how many athletes have their score registered in multiple divisions. Open division covers all athletes up to 54 years old. Plus there are sub categories from 35 years old up. Hence the duplicates. I will remove all sub category duplicates that are also in the open category.

In [6]:

```
# Checking how many athletes are competing in multiple divisions, as atheletes 35-54 will b
# the individual age category
athlete_20.duplicated('competitorname').value_counts()
```

Out[6]:

```
False    229095
True     164440
dtype: int64
```

In [7]:

```
athlete_20.duplicated('competitorid').value_counts()
```

Out[7]:

```
False    238829
True     154706
dtype: int64
```

In [8]:

```
athlete_20.sort_values('competitorname')
```

Out[8]:

	competitorid	competitorname	postcompstatus	gender	countryoforiginname	divisionid
184384	1775438	/Sarah Hashiguchi/	NaN	F	United States	2
153995	1775438	/Sarah Hashiguchi/	NaN	F	United States	2
168726	1116887	A C Davis	NaN	F	United States	2
222580	870593	A J LeRoy	NaN	M	United States	1
81714	300323	A J Palmisano	NaN	M	United States	5
...	...	...	...	...	...	...
151597	1786311	승원 Kim	NaN	F	Korea, Republic of	2
216020	1806706	원 유석	NaN	M	Korea, Republic of	1
89684	839362	유정 Jeong	NaN	F	Korea, Republic of	2
205490	63051	이 용훈	NaN	M	Korea, Republic of	1
280245	1826757	재석 박	NaN	M	Korea, Republic of	1

393535 rows × 14 columns

In [9]:

```
athlete_20['divisionid'].value_counts()
```

Out[9]:

```
1     74430
2     60747
18    59504
19    42130
12    41850
13    28752
3     26372
4     17700
5     14926
6     10496
7      4270
8      3351
9      3009
10     2442
16     1958
17     1598
Name: divisionid, dtype: int64
```

In [10]:

```
athlete_20.sort_values(['division', 'is_scaled'], ascending=False)
```

Out[10]:

	competitorid	competitorname	postcompstatus	gender	countryoforiginname	divisionid
75621	1641808	Susan Barton	NaN	F	United States	10
75622	715607	Lea Kroll	NaN	F	United States	10
75623	1817586	Carrie Runge	NaN	F	United States	10
75624	1810878	Mary Magistrale-Allan	NaN	F	United States	10
75625	967888	France Roy	NaN	F	Canada	10
...	...	...	...	...	...	...
332482	701430	Anthony Gallego	NaN	M	United Kingdom	3
332483	941007	Thomas Woodward	NaN	M	United Kingdom	1
332484	464456	Stephen Hipskind	NaN	M	United States	1
332485	81415	Shane Lemon	NaN	M	Canada	3
332486	28739	Scott Amory	NaN	M	United States	12

393535 rows × 14 columns

Data Cleaning

We want to drop duplicates of athletes as these duplicates occur when athletes in age 35-54 division are also represented in the open male or female division. We want to keep all of these athletes in the open division for simplicity.

Some athletes were scaled AND not-scaled for some reason also, we removed the scaled version if they had done both.

In [11]:

```
athlete_20 = athlete_20.sort_values(['division', 'is_scaled'], ascending=False).drop_duplic
```

In [12]:

```
athlete_20['division'].unique()
```

Out[12]:

```
array(['Women (60+)', 'Women (55-59)', 'Women', 'Men (60+)',
       'Men (55-59)', 'Men'], dtype=object)
```

In [13]:

```
athlete_20.sort_values('competitorname').head()
```

Out[13]:

	competitorid	competitorname	postcompstatus	gender	countryoforiginname	divisionid
153995	1775438	/Sarah Hashiguchi/	NaN	F	United States	2
168726	1116887	A C Davis	NaN	F	United States	2
222580	870593	A J LeRoy	NaN	M	United States	1
278040	300323	A J Palmisano	NaN	M	United States	5
135548	1821317	A Young Song	NaN	F	Korea, Republic of	2

In [14]:

```
athlete_20.rename(columns={'countryoforiginname':'country', 'height':'height_m', 'weight':'
```

Checking for athletes that have no input for their height and weight, we will remove these for now under seperate dataframes so that we can use these body stats to determine any significance to results.

In [15]:

```
height_20 = athlete_20.dropna(subset = ['height_m'])
height_20
```

Out[15]:

	competitorid	competitorname	postcompstatus	gender	country	divisionid	affiliatenar
73815	330392	Patricia McGill	NaN	F	Canada	10	Cross Linds
73816	145039	Pauline Sciascia	NaN	F	New Zealand	10	Cross Cent Wellingt
73817	24334	Lynne Knapman	NaN	F	Australia	10	Cross Acti
73818	177229	Patricia Failla	NaN	F	United States	10	CrossFit Fa
73819	130909	Nancy Bodet	NaN	F	United States	10	CrossFit B Fa
...	...	...	...	...	...	...	
332482	701430	Anthony Gallego	NaN	M	United Kingdom	3	Cc Performan Cross
332483	941007	Thomas Woodward	NaN	M	United Kingdom	1	Cross Salfc
332484	464456	Stephen Hipskind	NaN	M	United States	1	CrossFit H
332485	81415	Shane Lemon	NaN	M	Canada	3	Physi Cross
332486	28739	Scott Amory	NaN	M	United States	12	Na

132340 rows × 14 columns

In [16]:

```
weight_20 = athlete_20.dropna(subset = ['weight_kg'])
weight_20
```

Out[16]:

	competitorid	competitorname	postcompstatus	gender	country	divisionid	affiliatenar
73815	330392	Patricia McGill	NaN	F	Canada	10	Cross Linds
73816	145039	Pauline Sciascia	NaN	F	New Zealand	10	Cross Cent Wellingt
73817	24334	Lynne Knapman	NaN	F	Australia	10	Cross Acti
73818	177229	Patricia Failla	NaN	F	United States	10	CrossFit Fa
73819	130909	Nancy Bodet	NaN	F	United States	10	CrossFit B Fa
...	...	...	...	...	...	...	
332481	4520	Michael Huynh	NaN	M	United States	12	Cross Rising Wa
332483	941007	Thomas Woodward	NaN	M	United Kingdom	1	Cross Salfc
332484	464456	Stephen Hipskind	NaN	M	United States	1	CrossFit H
332485	81415	Shane Lemon	NaN	M	Canada	3	Physi Cross
332486	28739	Scott Amory	NaN	M	United States	12	Na

138261 rows × 14 columns

Compare the impact of removal of previous duplicates on height and weight values

In [17]:

```
height_20.isna().sum()
```

Out[17]:

```
competitorid           0
competitorname         0
postcompstatus    132060
gender                 0
country                0
divisionid             0
affiliatename       8232
age                    0
height_m               0
weight_kg           5112
overallrank            0
overallscore           0
is_scaled              0
division               0
dtype: int64
```

In [18]:

```
weight_20.isna().sum()
```

Out[18]:

```
competitorid           0
competitorname         0
postcompstatus    137981
gender                 0
country                0
divisionid             0
affiliatename       8599
age                    0
height_m           11033
weight_kg              0
overallrank            0
overallscore           0
is_scaled              0
division               0
dtype: int64
```

In [19]:

```
athlete_20 = athlete_20.dropna(subset = ['height_m', 'weight_kg'])
athlete_20.isna().sum()
```

Out[19]:

```
competitorid           0
competitorname         0
postcompstatus    126948
gender                 0
country                0
divisionid             0
affiliatename       8035
age                    0
height_m               0
weight_kg              0
overallrank            0
overallscore           0
is_scaled              0
division               0
dtype: int64
```

In [20]:

```
athlete_20.isna().sum()
```

Out[20]:

```
competitorid           0
competitorname         0
postcompstatus    126948
gender                 0
country                0
divisionid             0
affiliatename       8035
age                    0
height_m               0
weight_kg              0
overallrank            0
overallscore           0
is_scaled              0
division               0
dtype: int64
```

In [21]:

```
athlete_20['is_scaled'].describe()
```

Out[21]:

```
count    127228.0
mean          0.0
std           0.0
min           0.0
25%           0.0
50%           0.0
75%           0.0
max           0.0
Name: is_scaled, dtype: float64
```

Now that these scaled athletes have been resolved, we'll now remove the columns.

In [22]:

```
athlete_20.drop(columns = ['is_scaled', 'divisionid'], inplace=True)
athlete_20.head()
```

Out[22]:

	competitorid	competitorname	postcompstatus	gender	country	affiliatename	age	hei
73815	330392	Patricia McGill	NaN	F	Canada	CrossFit Lindsay	61	
73816	145039	Pauline Sciascia	NaN	F	New Zealand	CrossFit Central Wellington	62	
73817	24334	Lynne Knapman	NaN	F	Australia	CrossFit Active	60	
73818	177229	Patricia Failla	NaN	F	United States	CrossFit Fit Farm	63	
73819	130909	Nancy Bodet	NaN	F	United States	CrossFit Bull Falls	60	

In [23]:

```
athlete_20.sort_values('weight_kg').head()
```

Out[23]:

	competitorid	competitorname	postcompstatus	gender	country	affiliatename	age	he
309863	1306831	Caleb Tripp	NaN	M	United States	CrossFit Red Hook	34	
130091	279478	Janelle Lavallee	NaN	F	Canada	Norak CrossFit	25	
268717	1092806	Harsit Patel	NaN	M	United States	Iron Battalion CrossFit	42	
142317	495767	Louise Garcia	NaN	F	United States	CrossFit Arioch	50	
99165	1163442	Margaret Forbes	NaN	F	United States	CrossFit Colfax	36	

In [24]:

```
athlete_20.sort_values('height_m').head()
```

Out[24]:

	competitorid	competitorname	postcompstatus	gender	country	affiliatename	age
306582	1758585	Marc Tavakolian	NaN	M	Iran	CrossFit ASP	46
104944	1305703	Iarinjatovo Santatra	accepted	F	Madagascar	CrossFit Lemurs	32
235710	1601593	Vince in het Panhuis	NaN	M	Netherlands	CrossFit Etten-Leur	27
70714	926621	Caroline Adcock	NaN	F	France	CrossFit W2S	56
296201	461076	Alberto Cusi	NaN	M	Spain	CrossFit Betulo	39

We can see from above that heights and weights are unrealistic for some athletes, we will later remove these and then remove any further outliers from the dataset.

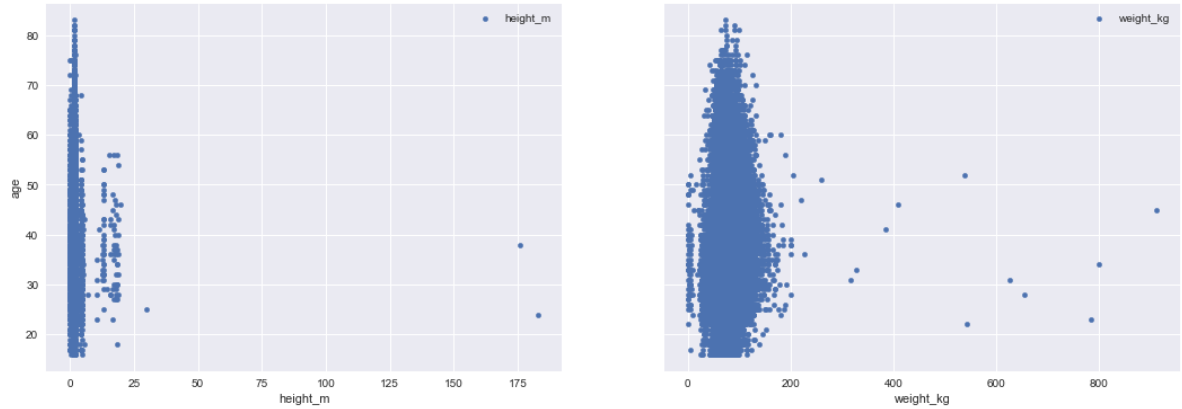
In [25]:

```
athlete_20.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 127228 entries, 73815 to 332486
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   competitorid    127228 non-null  int64  
 1   competitorname  127228 non-null  object 
 2   postcompstatus  280 non-null     object 
 3   gender          127228 non-null  object 
 4   country         127228 non-null  object 
 5   affiliatename   119193 non-null  object 
 6   age             127228 non-null  int64  
 7   height_m        127228 non-null  float64
 8   weight_kg       127228 non-null  float64
 9   overallrank     127228 non-null  int64  
 10  overallscore    127228 non-null  int64  
 11  division        127228 non-null  object 
dtypes: float64(2), int64(4), object(6)
memory usage: 12.6+ MB
```

In [26]:

```
fig, axs = plt.subplots(1, 2, sharey=True, figsize=(18, 6))
for idx, channel in enumerate(['height_m', 'weight_kg']):
 athlete_20.plot(kind='scatter', x=channel, y='age', ax=axs[idx], label=channel)
plt.legend()
plt.show()
```



In [27]:

```
# Remove obvious outliers based on poor input from competitor
athlete_20 = athlete_20.drop(athlete_20[athlete_20.height_m >= 2.2].index)
athlete_20 = athlete_20.drop(athlete_20[athlete_20.height_m <= 1.2].index)
athlete_20 = athlete_20.drop(athlete_20[athlete_20.weight_kg >= 200].index)
athlete_20 = athlete_20.drop(athlete_20[athlete_20.weight_kg <= 25].index)
```

In [28]:

```
athlete_20['height_m'].describe()
```

Out[28]:

```
count    125240.000000
mean          1.742415
std           0.096840
min           1.220000
25%           1.680000
50%           1.750000
75%           1.810000
max           2.150000
Name: height_m, dtype: float64
```

In [29]:

```
athlete_20['weight_kg'].describe()
```

Out[29]:

```
count    125240.000000
mean         78.006227
std          14.751000
min          25.400000
25%          67.590000
50%          78.930000
75%          87.540000
max         189.150000
Name: weight_kg, dtype: float64
```

View the scatter plot again after the removal of obvious outliers

In [30]:

```
fig, axs = plt.subplots(1, 2, sharey=True, figsize=(18, 6))
for idx, channel in enumerate(['height_m', 'weight_kg']):
 athlete_20.plot(kind='scatter', x=channel, y='age', ax=axs[idx], label=channel)
plt.legend()
plt.show()
```

In [31]:

```
athlete_20.isna().sum()
```

Out[31]:

```
competitorid           0
competitorname         0
postcompstatus    124968
gender                 0
country                0
affiliatename       7861
age                    0
height_m               0
weight_kg              0
overallrank            0
overallscore           0
division               0
dtype: int64
```

In [32]:

```
df = athlete_20.iloc[:,[7,8]]
# z = np.abs(stats.zscore(df))
threshold = 3
df.shape
```

Out[32]:

```
(125240, 2)
```

In [33]:

```
df.isna().sum()
```

Out[33]:

```
height_m     0
weight_kg    0
dtype: int64
```

In [34]:

```
df
```

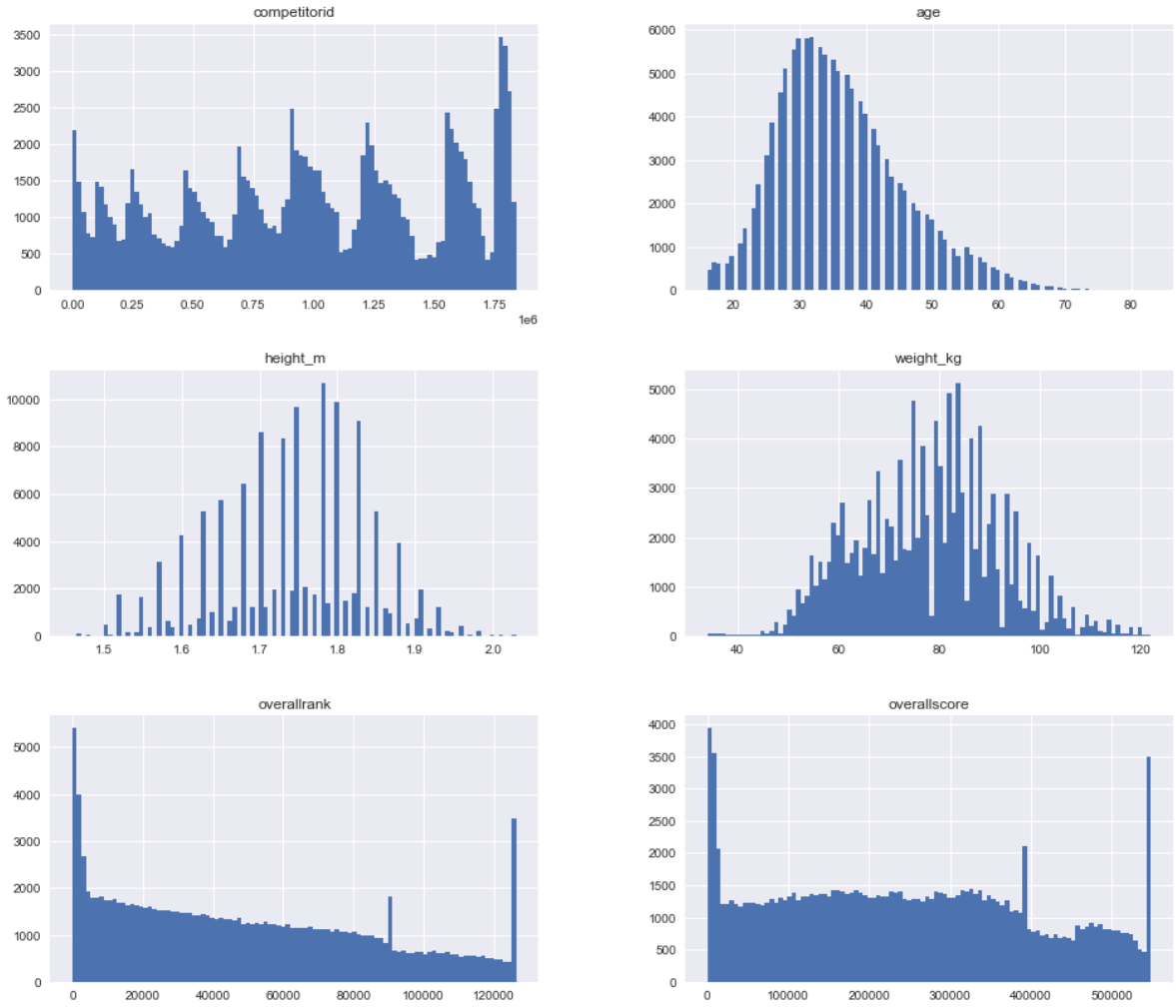
Out[34]:

	height_m	weight_kg
73815	1.73	67.59
73816	1.60	58.00
73817	1.57	56.70
73818	1.57	53.52
73819	1.68	52.16
...	...	...
332481	1.78	61.23
332483	1.68	55.79
332484	1.80	70.31
332485	1.73	83.91
332486	1.96	142.88

125240 rows × 2 columns

In [159]:

```
fig = plt.figure(figsize = (16,14))
ax = fig.gca()
athlete_20.hist(ax = ax, bins=100);
```



In [36]:

```
# data = df[(z < 3).all(axis=1)]
athlete_20 = athlete_20[(np.abs(stats.zscore(df)) < 3).all(axis=1)]
# athlete_201.drop(columns = ['height_m', 'weight_kg'], inplace=True)
# df = pd.concat([athlete_201, data], axis = 1)
# athlete_201 = df
```

In [37]:

```
#obtain stats details
pd.options.display.float_format = lambda x : '{:.0f}'.format(x) if int(x) == x else '{:,.2f
athlete_20.describe()
```

Out[37]:

	competitorid	age	height_m	weight_kg	overallrank	overallscore
count	124190	124190	124190	124190	124190	124190
mean	983,642.85	36.09	1.74	77.78	49,363.75	244,413.15
std	546,032.25	9.51	0.10	14.02	36,364.77	157,574.23
min	86	16	1.46	34.02	1	10
25%	527380	29	1.68	67.59	17,511.25	111,197.50
50%	987711	35	1.75	78.47	43988	236508
75%	1,483,789.25	42	1.80	87	76,709.25	364,359.25
max	1838935	83	2.03	122.02	126461	548388

In [38]:

```
athlete_20['weight_kg'].describe()
```

Out[38]:

```
count   124190
mean     77.78
std      14.02
min      34.02
25%      67.59
50%      78.47
75%         87
max     122.02
Name: weight_kg, dtype: float64
```

In [39]:

```
athlete_20['height_m'].describe()
```

Out[39]:

```
count   124190
mean      1.74
std       0.10
min       1.46
25%       1.68
50%       1.75
75%       1.80
max       2.03
Name: height_m, dtype: float64
```

In [40]:

```
athlete_20.isna().sum()
```

Out[40]:

```
competitorid           0
competitorname         0
postcompstatus    123919
gender                 0
country                0
affiliatename       7798
age                    0
height_m               0
weight_kg              0
overallrank            0
overallscore           0
division               0
dtype: int64
```

Isolating Peak Performers

Isolate those that have been offered a spot at the Crossfit Games and are the top athletes in division.

In [41]:

```
athlete_20['postcompstatus'] = athlete_20['postcompstatus'].replace(np.nan, 0)
```

In [42]:

```
athlete_20['postcompstatus'].unique()
```

Out[42]:

```
array([0, 'accepted', 'declined', 'invited'], dtype=object)
```

In [43]:

```
offered = ['accepted', 'invited', 'declined']
games_athletes = athlete_20[athlete_20['postcompstatus'].isin(offered)]
games_athletes
```

Out[43]:

	competitorid	competitorname	postcompstatus	gender	country	affiliatename	age
86729	8859	Ragnheiður Sara Sigmundsdottir	accepted	F	Iceland	Simmagym CrossFit	27
86730	18588	Annie Thorisdottir	accepted	F	Iceland	CrossFit Reykjavík	30
86731	120480	Kristin Holte	accepted	F	Norway	CrossFit Oslo	34
86732	163097	Tia-Clair Toomey	accepted	F	Australia	CrossFit Mayhem	26
86733	264512	Jamie Greene	accepted	F	New Zealand	CrossFit Yas	29
...	...	...	...	...	...	...	...
221655	1577944	Isaac Muhima	accepted	M	Uganda	CrossFit BECC	29
221934	1522068	Thabo Mosimane	accepted	M	Botswana	CrossFit Gaborone	29
225129	1550856	David Alpoim de Sa	accepted	M	Mozambique	Reebok CrossFit Reading	32
237037	994360	Utsab Karki	accepted	M	Nepal	CrossFit Hyperion	30
288007	1806128	Jack Karo Jr.	accepted	M	Papua New Guinea	CrossFit Port Moresby	33

271 rows × 12 columns

Remove postcompstatus from dataframe to simplify things later on with NaN cells and we have saved a seperate dataframe with that column in it.

In [44]:

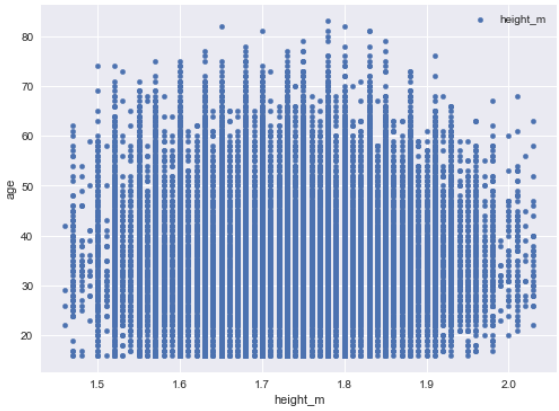
```
athlete_20.drop(columns = ['postcompstatus', 'affiliatename'], inplace=True)
athlete_20.head()
```

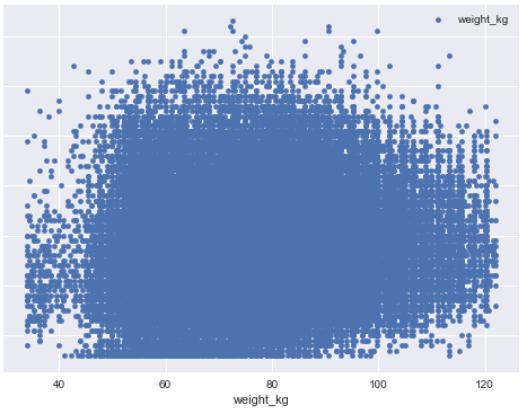
Out[44]:

	competitorid	competitorname	gender	country	age	height_m	weight_kg	overallrank
73815	330392	Patricia McGill	F	Canada	61	1.73	67.59	1
73816	145039	Pauline Sciascia	F	New Zealand	62	1.60	58	2
73817	24334	Lynne Knapman	F	Australia	60	1.57	56.70	3
73818	177229	Patricia Failla	F	United States	63	1.57	53.52	4
73819	130909	Nancy Bodet	F	United States	60	1.68	52.16	5

In [45]:

```
fig, axs = plt.subplots(1, 2, sharey=True, figsize=(18, 6))
for idx, channel in enumerate(['height_m', 'weight_kg']):
    athlete_20.plot(kind='scatter', x=channel, y='age', ax=axs[idx], label=channel)
plt.legend()
plt.show()
```





In [46]:

```
athlete_20.corr()
```

Out[46]:

	competitorid	age	height_m	weight_kg	overallrank	overallscore
competitorid	1	-0.23	0.02	-0.04	0.34	0.34
age	-0.23	1	0.03	0.09	-0.00	-0.02
height_m	0.02	0.03	1	0.74	0.21	0.23
weight_kg	-0.04	0.09	0.74	1	0.22	0.24
overallrank	0.34	-0.00	0.21	0.22	1	0.99
overallscore	0.34	-0.02	0.23	0.24	0.99	1

In [47]:

```
athlete_20.sort_values('weight_kg')
```

Out[47]:

	competitorid	competitorname	gender	country	age	height_m	weight_kg	overallrank
253163	1396803	Jean Charles Bouchard	M	Canada	30	1.47	34.02	54551
239429	1721299	Tomas Konir	M	Czech Republic	22	1.71	34.02	40817
221975	1424959	Magnar Fuglevig	M	Norway	34	1.74	34.02	23363
233294	1037638	Franck Thommas	M	France	40	1.74	34.02	34682
131467	1790702	Camille Henri	F	France	27	1.69	34.02	44739
...	...	...	...	...	...	...	...	...
323933	829240	Gary Carruthers	M	United States	39	1.93	122.02	125311
281057	1608947	Darlyn Scott	M	United States	36	1.80	122.02	82445
135792	1577559	Lindsae Kish	F	United States	34	1.57	122.02	49064
319477	318654	Timothy Kinsworthy	M	United States	38	1.78	122.02	120865
280316	106103	Kaleo Cornwell	M	United States	43	1.85	122.02	81704

124190 rows × 10 columns

In [48]:

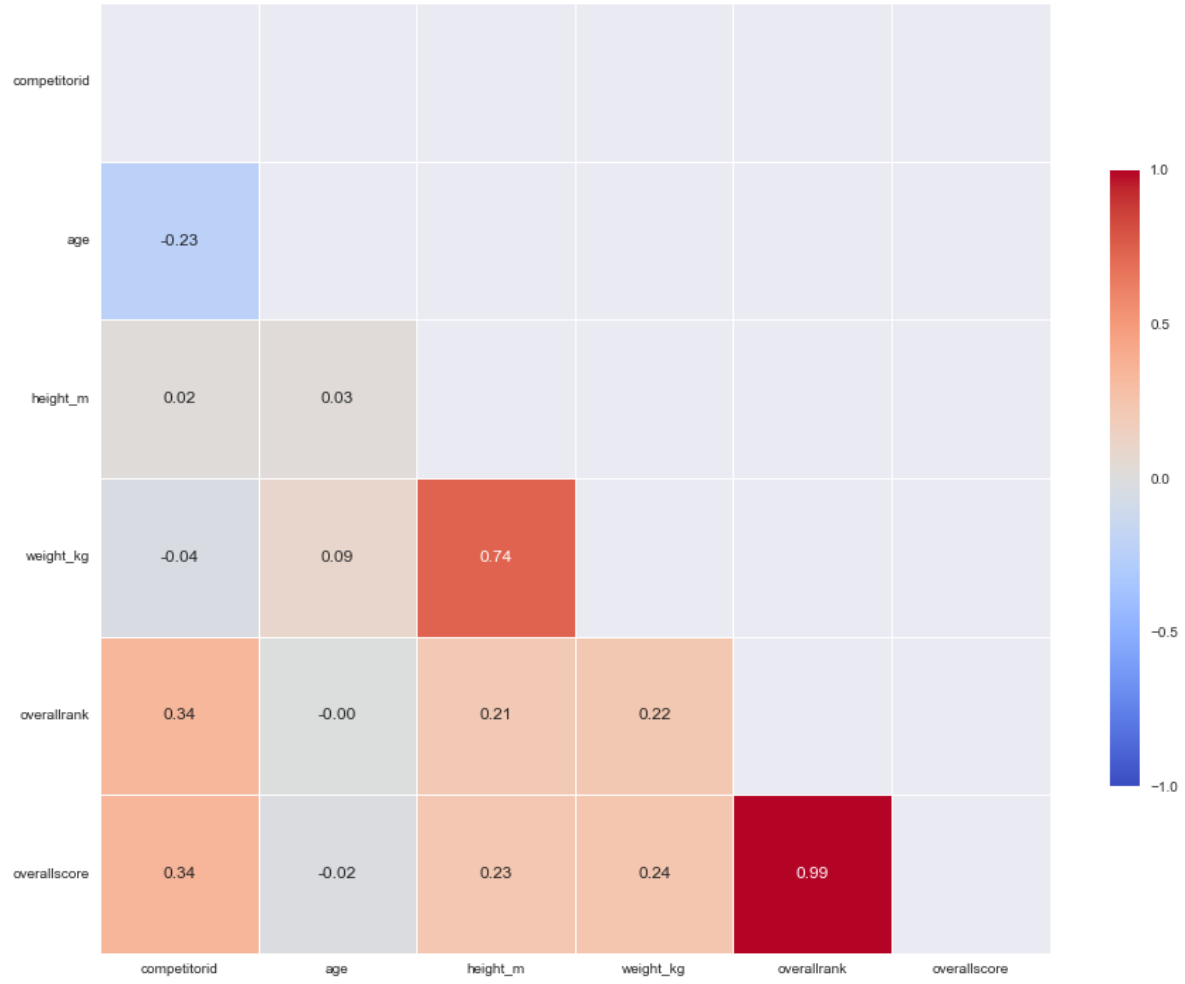
```
athlete_20.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 124190 entries, 73815 to 332485
Data columns (total 10 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   competitorid    124190 non-null  int64  
 1   competitorname  124190 non-null  object 
 2   gender          124190 non-null  object 
 3   country         124190 non-null  object 
 4   age             124190 non-null  int64  
 5   height_m        124190 non-null  float64
 6   weight_kg       124190 non-null  float64
 7   overallrank     124190 non-null  int64  
 8   overallscore    124190 non-null  int64  
 9   division        124190 non-null  object 
dtypes: float64(2), int64(4), object(4)
memory usage: 10.4+ MB
```

Correlation Visualisation

In [49]:

```
# Visulaise with a correlation matrix using heatmap
corr = athlete_20.corr()
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
f, ax = plt.subplots(figsize=(15, 20))
sns.heatmap(corr,
 mask=mask,
 square = True,
 linewidths = .5,
 cmap = "coolwarm",
 cbar_kws = {'shrink': .4,
 "ticks" : [-1, -.5, 0, 0.5, 1]},
 vmin = -1,
 vmax = 1,
 annot = True,
 annot_kws = {"size": 12},
 fmt = ".2f")
ax.set_yticklabels(corr.columns, rotation = 0)
ax.set_xticklabels(corr.columns)
plt.show()
```

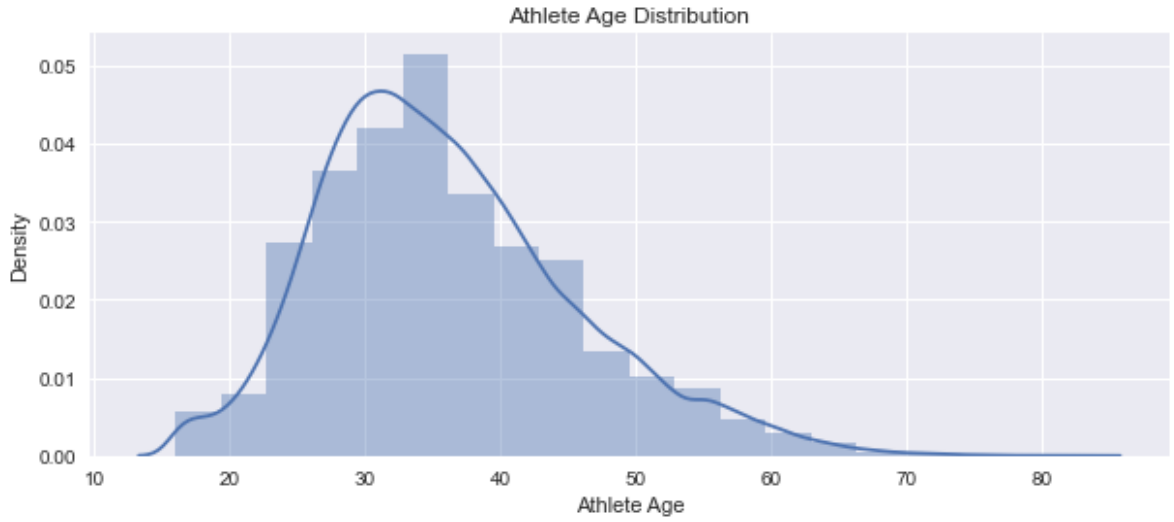


Dispertion of athletes

Checking for the distribution of athlete participants

In [50]:

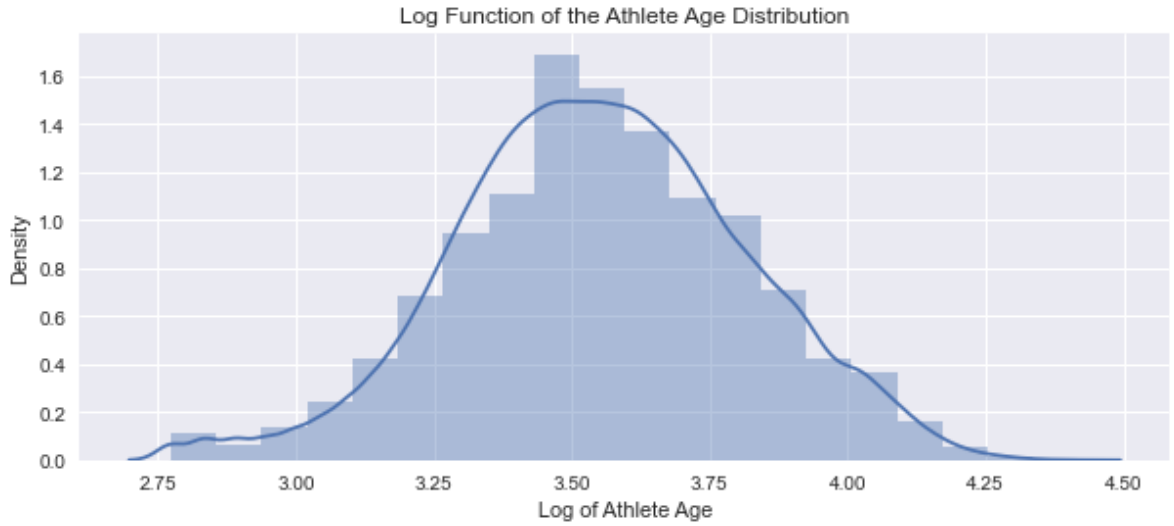
```
plt.figure(figsize=(10,4))

price_dist = sns.distplot(athlete_20["age"], bins=20)
price_dist.set(xlabel="Athlete Age", title="Athlete Age Distribution")
plt.show()
```



The distribution is right skewed so we can try to make this more nomralised by creating a log distribution

In [160]:

```
plt.figure(figsize=(10,4))

logged_price_dist = sns.distplot(np.log(athlete_20["age"]), bins=20)
logged_price_dist.set(xlabel="Log of Athlete Age", title="Log Function of the Athlete Age D
plt.show()
```



In [52]:

```
#OLS squared data representative of the variables to predict
outcome = 'overallrank'
x_cols = ['age', 'height_m', 'weight_kg']
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model = ols(formula=formula, data=athlete_20).fit()
model.summary()
```

Out[52]:

OLS Regression Results

Dep. Variable:	overallrank	**R-squared:**	0.054
Model:	OLS	**Adj. R-squared:**	0.054
Method:	Least Squares	**F-statistic:**	2363.
Date:	Fri, 24 Jun 2022	**Prob (F-statistic):**	0.00
Time:	13:31:46	**Log-Likelihood:**	-1.4769e+06
No. Observations:	124190	**AIC:**	2.954e+06
Df Residuals:	124186	**BIC:**	2.954e+06
Df Model:	3		
Covariance Type:	nonrobust		

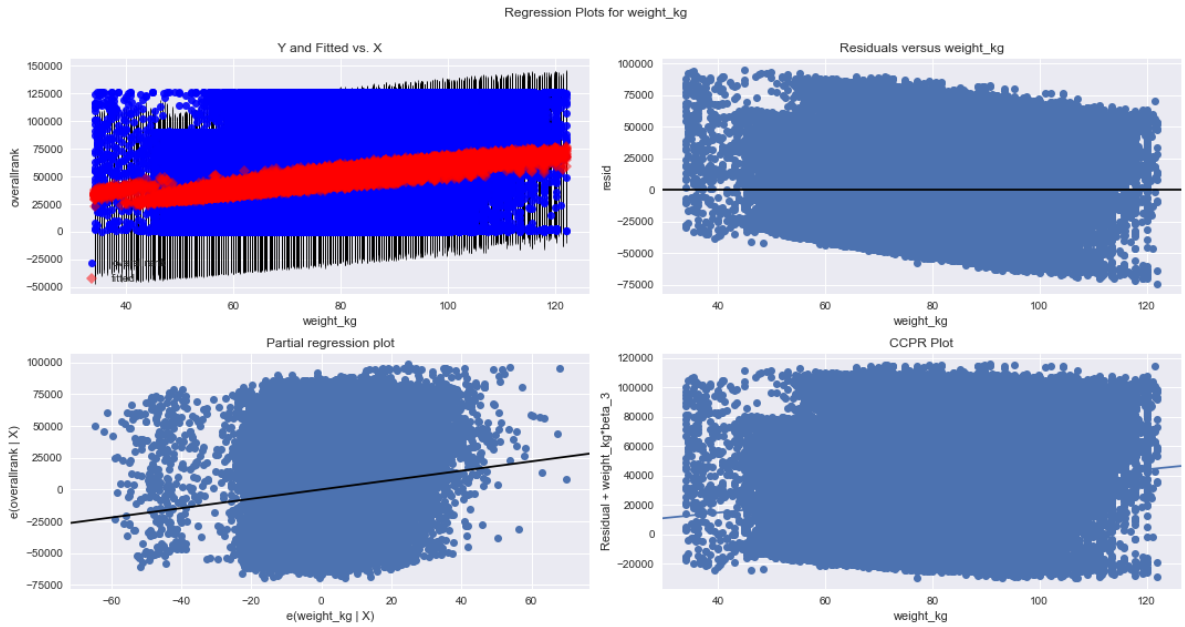
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-4.786e+04	2221.091	-21.549	0.000	-5.22e+04	-4.35e+04
age	-67.4278	10.613	-6.353	0.000	-88.229	-46.627
height_m	4.08e+04	1560.874	26.139	0.000	3.77e+04	4.39e+04
weight_kg	367.4660	10.721	34.274	0.000	346.452	388.480

Omnibus:	15928.927	**Durbin-Watson:**	0.054
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	6623.791
Skew:	0.375	**Prob(JB):**	0.00
Kurtosis:	2.153	**Cond. No.**	2.33e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.33e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

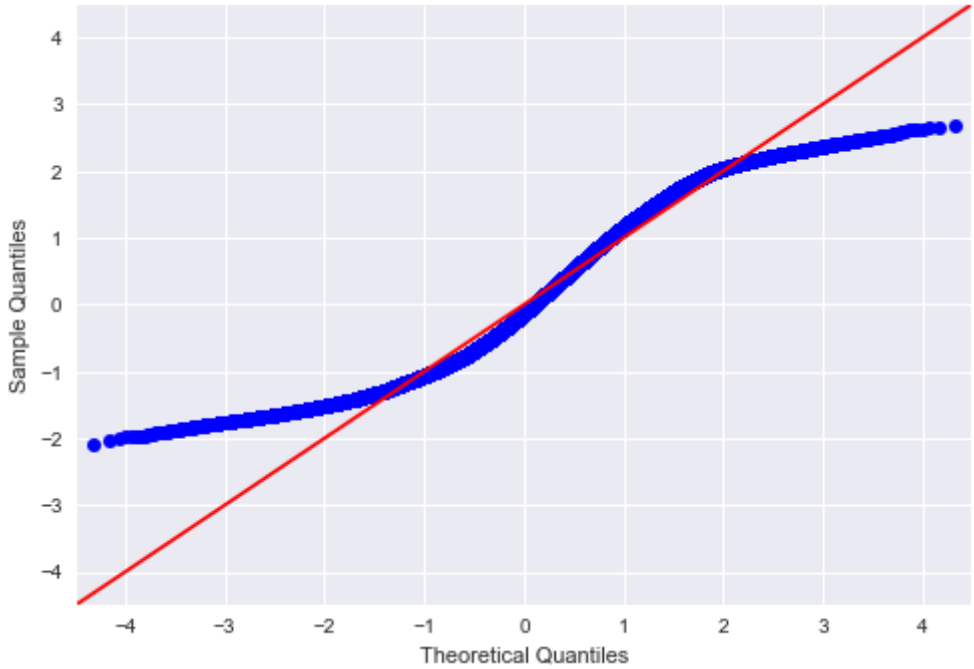
In [53]:

```
fig = plt.figure(figsize=(15,8))
fig = sm.graphics.plot_regress_exog(model, "weight_kg", fig=fig)
plt.show()
```



In [54]:

```
import scipy.stats as stats
residuals = model.resid
sm.graphics.qqplot(residuals, dist=stats.norm, line='45', fit=True)
plt.show()
```



Score Dataset Data Preparation

In [55]:

```
score_20
```

Out[55]:

	affiliate	breakdown	competitorid	division	is_scaled	judge	ordinal	rank	scal
0	CrossFit Sunninghill	10 rounds	120333	Men (35-39)	0	Richard Smith	1	12	
1	CrossFit Sunninghill	27 rounds +\n3 thrusters\n	120333	Men (35-39)	0	Richard Smith	2	2	
2	CrossFit Sunninghill	165 reps	120333	Men (35-39)	0	Richard Smith	3	9	
3	CrossFit Sunninghill	240 reps	120333	Men (35-39)	0	Richard Smith	4	1	
4	CrossFit Sunninghill	240 reps	120333	Men (35-39)	0	Richard Smith	5	1	
...	...	...	...	...	...	...	...	...	
1967670	CrossFit Gympie	1 ground-to-overhead\n	1755911	Women	1	Jemma Cowper	1	15874	
1967671	NaN	NaN	1755911	Women	1	NaN	2	15488	
1967672	NaN	NaN	1755911	Women	1	NaN	3	13971	
1967673	NaN	NaN	1755911	Women	1	NaN	4	12903	
1967674	NaN	NaN	1755911	Women	1	NaN	5	11606	

1967675 rows × 13 columns

In [56]:

```
score_20.rename(columns={'ordinal':'workout', 'score':'delete', 'scoredisplay':'score', 'ra
score_20.head(10)
```

Out[56]:

	affiliate	breakdown	competitorid	division	is_scaled	judge	workout	workoutrank	sc
0	CrossFit Sunninghill	10 rounds	120333	Men (35-39)	0	Richard Smith	1	12	
1	CrossFit Sunninghill	27 rounds +\n3 thrusters\n	120333	Men (35-39)	0	Richard Smith	2	2	
2	CrossFit Sunninghill	165 reps	120333	Men (35-39)	0	Richard Smith	3	9	
3	CrossFit Sunninghill	240 reps	120333	Men (35-39)	0	Richard Smith	4	1	
4	CrossFit Sunninghill	240 reps	120333	Men (35-39)	0	Richard Smith	5	1	
5	CrossFit 061	10 rounds	5312	Men (35-39)	0	Harold Johnson	1	6	
6	CrossFit 061	25 rounds +\n4 thrusters\n4 toes-to-bars\n	5312	Men (35-39)	0	Dave Ruiz	2	27	
7	CrossFit 061	165 reps	5312	Men (35-39)	0	Carole Turnbo	3	11	
8	CrossFit 061	240 reps	5312	Men (35-39)	0	Kevin Jones	4	3	
9	CrossFit 061	240 reps	5312	Men (35-39)	0	Kevin Jones	5	17	

In [57]:

```
score_20 = score_20.sort_values(['competitorid', 'workout'], ascending=True)
```

In [58]:

```
score_20 = score_20.drop_duplicates(subset=['competitorid','workout'], keep='last')
score_20.head(10)
```

Out[58]:

	affiliate	breakdown	competitorid	division	is_scaled	judge	workout	workoutrank
526205	CrossFit HQ	5 rounds\n	86	Men	0	Lucas Zepeda	1	103375
526206	CrossFit HQ	11 rounds\n	86	Men	0	Katie Hogan	2	62328
526207	CrossFit HQ	21 deadlifts\n21 hand-release push-ups\n15 dea...	86	Men	0	Louis Lee	3	104034
526208	CrossFit HQ	30 box jumps\n15 clean and jerks\n30 box jumps...	86	Men	0	Leah Polaski	4	81206
526209	CrossFit HQ	36 muscle-ups\n80-cal. row\n120 wall-ball shot...	86	Men	0	Hans Roos	5	18095
172820	CrossFit HQ	10 rounds	88	Men	0	Jason Aucoin	1	9437
172821	CrossFit Santa Cruz	20 rounds +\n1 thruster\n	88	Men	0	Roy Feague	2	7594
172822	CrossFit Santa Cruz	21 deadlifts\n21 handstand push-ups\n15 deadli...	88	Men	0	Lucas Zepeda	3	6936
172823	CrossFit Santa Cruz	30 box jumps\n15 clean and jerks\n30 box jumps...	88	Men	0	Brad hoffeld	4	5694
172824	CrossFit Santa Cruz	240 reps	88	Men	0	Leah Polaski	5	6156

In [59]:

```
score_20.sort_values(['is_scaled'], ascending=False).head(10)
```

Out[59]:

	affiliate	breakdown	competitorid	division	is_scaled	judge	workout	workoutı
1896483	CrossFit Campbell	30 box jumps\n15 clean and jerks\n30 box jumps...	500000	Women	1	Jennifer saban	4	4
846067	CrossFit Strength Haven	21 deadlifts\n21 hand-release push-ups\n15 dea...	1791070	Men	1	Ruel	3	7
1060044	CrossFit Richards Bay	240 reps	1723275	Men (40-44)	1	Paul Botha	5	
1061839	Tornado Alley CrossFit	8 chin-over-bar pull-ups\n80-cal. row\n74 wall...	1791075	Men (40-44)	1	Eric Sandoval	5	1
1061838	Tornado Alley CrossFit	30 box jumps\n15 clean and jerks\n30 box jumps...	1791075	Men (40-44)	1	Eric Sandoval	4	1
1061837	Tornado Alley CrossFit	21 deadlifts\n21 hand-release push-ups\n15 dea...	1791075	Men (40-44)	1	Eric Montoya	3	1
1061836	Tornado Alley CrossFit	13 rounds +\n2 thrusters\n	1791075	Men (40-44)	1	Eric Sandoval	2	1
1061835	Tornado Alley CrossFit	6 rounds +\n8 ground-to-overheads\n4 step over...	1791075	Men (40-44)	1	Eric Montoya	1	1
857770	CrossFit Little Ships	7 rounds\n	1723309	Men	1	Ternoy Quentin	1	9
857771	CrossFit Little Ships	19 rounds +\n4 thrusters\n	1723309	Men	1	Pierre Ponchaux	2	6

In [60]:

```
score_20 = score_20.loc[score_20['is_scaled'] == 0]
score_20.head()
```

Out[60]:

	affiliate	breakdown	competitorid	division	is_scaled	judge	workout	workoutrank
526205	CrossFit HQ	5 rounds\n	86	Men	0	Lucas Zepeda	1	103375
526206	CrossFit HQ	11 rounds\n	86	Men	0	Katie Hogan	2	62328
526207	CrossFit HQ	21 deadlifts\n21 hand-release push-ups\n15 dea...	86	Men	0	Louis Lee	3	104034
526208	CrossFit HQ	30 box jumps\n15 clean and jerks\n30 box jumps...	86	Men	0	Leah Polaski	4	81206
526209	CrossFit HQ	36 muscle-ups\n80-cal. row\n120 wall-ball shot...	86	Men	0	Hans Roos	5	18095

In [61]:

```
score_20.drop(columns = ['affiliate', 'breakdown', 'judge', 'scoreidentifier', 'delete', 'i
score_20
```

Out[61]:

	competitorid	division	workout	workoutrank	scaled	score
526205	86	Men	1	103375	0	90 reps
526206	86	Men	2	62328	0	374 reps
526207	86	Men	3	104034	1	121 reps - s
526208	86	Men	4	81206	0	120 reps
526209	86	Men	5	18095	0	236 reps
...	...	...	...	...	...	...
780940	1840200	Men	1	120482	0	NaN
780941	1840200	Men	2	114611	0	NaN
780942	1840200	Men	3	107942	0	NaN
780943	1840200	Men	4	105229	0	NaN
780944	1840200	Men	5	100124	0	NaN

1024815 rows × 6 columns

In [62]:

```
score_20.sort_values(['competitorid', 'workout']).head(12)
```

Out[62]:

	competitorid	division	workout	workoutrank	scaled	score
526205	86	Men	1	103375	0	90 reps
526206	86	Men	2	62328	0	374 reps
526207	86	Men	3	104034	1	121 reps - s
526208	86	Men	4	81206	0	120 reps
526209	86	Men	5	18095	0	236 reps
172820	88	Men	1	9437	0	13:46
172821	88	Men	2	7594	0	681 reps
172822	88	Men	3	6936	0	118 reps
172823	88	Men	4	5694	0	201 reps
172824	88	Men	5	6156	0	16:26
725960	93	Men	1	120482	0	NaN
725961	93	Men	2	114611	0	NaN

In [63]:

```
score_20['score'].isnull().sum()
```

Out[63]:

169613

In [64]:

```
score_20.dropna(subset=['score'], inplace = True)
```

In [65]:

```
score_20
```

Out[65]:

	competitorid	division	workout	workoutrank	scaled	score
526205	86	Men	1	103375	0	90 reps
526206	86	Men	2	62328	0	374 reps
526207	86	Men	3	104034	1	121 reps - s
526208	86	Men	4	81206	0	120 reps
526209	86	Men	5	18095	0	236 reps
...	...	...	...	...	...	...
755804	1837207	Men	5	80796	0	200 reps
650058	1838606	Men	4	51187	0	160 reps
650059	1838606	Men	5	70081	0	200 reps
758573	1839158	Men	4	87681	0	84 reps
671544	1839242	Men	5	29722	0	223 reps

855202 rows × 6 columns

In [66]:

```
score_20['compid'] = score_20['competitorid'][score_20['workout'] ==1]
score_20['workout1'] = score_20['score'][score_20['workout'] == 1]
score_20['rank1'] = score_20['workoutrank'][score_20['workout'] == 1]
score_20['workout2'] = score_20['score'][score_20['workout'] == 2]
score_20['rank2'] = score_20['workoutrank'][score_20['workout'] == 2]
score_20['workout3'] = score_20['score'][score_20['workout'] == 3]
score_20['rank3'] = score_20['workoutrank'][score_20['workout'] == 3]
score_20['workout4'] = score_20['score'][score_20['workout'] == 4]
score_20['rank4'] = score_20['workoutrank'][score_20['workout'] == 4]
score_20['workout5'] = score_20['score'][score_20['workout'] == 5]
score_20['rank5'] = score_20['workoutrank'][score_20['workout'] == 5]
score_20.head(20)
```

Out[66]:

	competitorid	division	workout	workoutrank	scaled	score	compid	workout1	rank
526205	86	Men	1	103375	0	90 reps	86	90 reps	1033
526206	86	Men	2	62328	0	374 reps	NaN	NaN	Na
526207	86	Men	3	104034	1	121 reps - s	NaN	NaN	Na
526208	86	Men	4	81206	0	120 reps	NaN	NaN	Na
526209	86	Men	5	18095	0	236 reps	NaN	NaN	Na
172820	88	Men	1	9437	0	13:46	88	13:46	94
172821	88	Men	2	7594	0	681 reps	NaN	NaN	Na
172822	88	Men	3	6936	0	118 reps	NaN	NaN	Na
172823	88	Men	4	5694	0	201 reps	NaN	NaN	Na
172824	88	Men	5	6156	0	16:26	NaN	NaN	Na
725964	93	Men	5	62444	0	200 reps	NaN	NaN	Na
1327190	1617	Men (50-54)	1	223	0	175 reps	1617	175 reps	2
1327191	1617	Men (50-54)	2	159	0	554 reps	NaN	NaN	Na
1327192	1617	Men (50-54)	3	484	0	92 reps	NaN	NaN	Na
1327193	1617	Men (50-54)	4	152	0	167 reps	NaN	NaN	Na
1327194	1617	Men (50-54)	5	13	0	16:42	NaN	NaN	Na
252445	1618	Men	1	25081	0	171 reps	1618	171 reps	250
252446	1618	Men	2	43456	0	455 reps	NaN	NaN	Na

	competitorid	division	workout	workoutrank	scaled	score	compid	workout1	ranl
252447	1618	Men	3	18001	0	102 reps	NaN	NaN	Na
252448	1618	Men	4	16004	0	186 reps	NaN	NaN	Na

In [67]:

```
score_20.isnull().sum()
```

Out[67]:

```
competitorid         0
division             0
workout              0
workoutrank          0
scaled               0
score                0
compid          668869
workout1        668869
rank1           668869
workout2        677860
rank2           677860
workout3        686651
rank3           686651
workout4        689243
rank4           689243
workout5        698185
rank5           698185
dtype: int64
```

In [68]:

```
athlete_20 = athlete_20.sort_values('competitorid')
athlete_20
```

Out[68]:

	competitorid	competitorname	gender	country	age	height_m	weight_kg	overallrar
274127	86	Justin Bergh	M	United States	38	1.96	102.06	755
203450	88	Cary Hair	M	United States	35	1.83	86.64	48
314078	93	Leif Edmundson	M	United States	38	1.83	92.99	1154
213866	1617	John Mclaughlin	M	United States	52	1.78	84.82	152
219375	1618	Daniel Campbell	M	United States	36	1.73	78.93	207
...	...	...	...	...	...	...	...	
174346	1837223	Shaymaa Ismail	F	Saudi Arabia	22	1.61	63	876
325087	1837961	Maksim Kuzin	M	Russian Federation	28	1.78	80	1264
170526	1838184	Charlene Fielding	F	United States	34	1.65	60.33	837
298897	1838606	Tadas Vismantas	M	Lithuania	25	1.81	86	1002
325079	1838935	David Riesing	M	United States	28	1.80	86.18	1264

124190 rows × 10 columns

In [69]:

```
pd.set_option('display.max_columns', None)
athlete_score = athlete_20.merge(score_20, on = 'competitorid')
athlete_score = athlete_score.sort_values(['competitorid', 'workout'])
athlete_score
```

Out[69]:

	competitorid	competitorname	gender	country	age	height_m	weight_kg	overallrank
0	86	Justin Bergh	M	United States	38	1.96	102.06	75515
1	86	Justin Bergh	M	United States	38	1.96	102.06	75515
2	86	Justin Bergh	M	United States	38	1.96	102.06	75515
3	86	Justin Bergh	M	United States	38	1.96	102.06	75515
4	86	Justin Bergh	M	United States	38	1.96	102.06	75515
...	...	...	...	...	...	...	...	...
492037	1837174	Albert Daigle	M	United States	57	1.83	99.79	2338
492038	1837174	Albert Daigle	M	United States	57	1.83	99.79	2338
492039	1837174	Albert Daigle	M	United States	57	1.83	99.79	2338
492040	1838606	Tadas Vismantas	M	Lithuania	25	1.81	86	100285
492041	1838606	Tadas Vismantas	M	Lithuania	25	1.81	86	100285

492042 rows × 26 columns

Adding a column for the number of rows to help keep track when breaking apart and join df's below

In [70]:

```
def justify(a, invalid_val=0, axis=0, side='up'):
    """
    Justifies a 2D array

    Parameters
    ----------
    A : ndarray
        Input array to be justified
    axis : int
        Axis along which justification is to be made
    side : str
        Direction of justification. It could be 'left', 'right', 'up', 'down'
        It should be 'left' or 'right' for axis=1 and 'up' or 'down' for axis=0.

    """

    if invalid_val is np.nan:
        #change to notnull
        mask = pd.notnull(a)
    else:
        mask = a!=invalid_val
    justified_mask = np.sort(mask,axis=axis)
    if (side=='up'):
        justified_mask = np.flip(justified_mask,axis=axis)
    #change dtype to object
    out = np.full(a.shape, invalid_val, dtype=object)
    if axis==1:
        out[justified_mask] = a[mask]
    else:
        out.T[justified_mask.T] = a.T[mask.T]
    return out
```

In [71]:

```
athlete_score = pd.DataFrame(justify(athlete_score.values, invalid_val=np.nan, side='up', a
athlete_score
```

Out[71]:

	competitorid	competitorname	gender	country	age	height_m	weight_kg	overallrank
0	86	Justin Bergh	M	United States	38	1.96	102.06	75515
1	86	Justin Bergh	M	United States	38	1.96	102.06	75515
2	86	Justin Bergh	M	United States	38	1.96	102.06	75515
3	86	Justin Bergh	M	United States	38	1.96	102.06	75515
4	86	Justin Bergh	M	United States	38	1.96	102.06	75515
...	...	...	...	...	...	...	...	...
492037	1837174	Albert Daigle	M	United States	57	1.83	99.79	2338
492038	1837174	Albert Daigle	M	United States	57	1.83	99.79	2338
492039	1837174	Albert Daigle	M	United States	57	1.83	99.79	2338
492040	1838606	Tadas Vismantas	M	Lithuania	25	1.81	86	100285
492041	1838606	Tadas Vismantas	M	Lithuania	25	1.81	86	100285

492042 rows × 26 columns

In [72]:

```
dummy_df = athlete_score.iloc[:,15:26]
dummy_df
```

Out[72]:

	compid	workout1	rank1	workout2	rank2	workout3	rank3	workout4	rank4	wor
0	86	90 reps	103375	374 reps	62328	121 reps - s	104034	120 reps	81206	23(
1	88	13:46	9437	681 reps	7594	118 reps	6936	201 reps	5694	
2	1617	175 reps	223	554 reps	159	92 reps	484	167 reps	152	20(
3	1618	171 reps	25081	455 reps	43456	102 reps	18001	186 reps	16004	
4	1619	167 reps	982	476 reps	1586	86 reps	1816	162 reps	990	22:
...	...	...	...	...	...	...	...	...	...	
492037	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
492038	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
492039	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
492040	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
492041	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

492042 rows × 11 columns

Drop all athletes that didn't complete all 5 workouts

In [73]:

```
dummy_df.dropna(subset=['workout5'], inplace=True)
dummy_df
```

Out[73]:

	compid	workout1	rank1	workout2	rank2	workout3	rank3	workout4	rank4	worl
0	86	90 reps	103375	374 reps	62328	121 reps - s	104034	120 reps	81206	236
1	88	13:46	9437	681 reps	7594	118 reps	6936	201 reps	5694	
2	1617	175 reps	223	554 reps	159	92 reps	484	167 reps	152	200
3	1618	171 reps	25081	455 reps	43456	102 reps	18001	186 reps	16004	
4	1619	167 reps	982	476 reps	1586	86 reps	1816	162 reps	990	223
...	...	...	...	...	...	...	...	...	...	
91645	1626446	166 reps	29680	309 reps	8996	8:09 - s	90438	121 reps	75621	200
91646	1626448	160 reps	37760	201 reps	85595	136 reps - s	62761	160 reps	47630	213
91647	1626452	162 reps	35184	729 reps - s	64571	126 reps - s	66731	120 reps	47097	200
91648	1626453	145 reps	56939	511 reps	12372	65 reps	51500	160 reps	39355	203
91649	1626490	147 reps	53558	294 reps	36057	106 reps	14681	165 reps - s	67811	200

91650 rows × 11 columns

In [74]:

```
athlete_score.drop(athlete_score.iloc[:,15:26],axis=1, inplace=True)
athlete_score
```

Out[74]:

	competitorid	competitorname	gender	country	age	height_m	weight_kg	overallrank
0	86	Justin Bergh	M	United States	38	1.96	102.06	75515
1	86	Justin Bergh	M	United States	38	1.96	102.06	75515
2	86	Justin Bergh	M	United States	38	1.96	102.06	75515
3	86	Justin Bergh	M	United States	38	1.96	102.06	75515
4	86	Justin Bergh	M	United States	38	1.96	102.06	75515
...	...	...	...	...	...	...	...	...
492037	1837174	Albert Daigle	M	United States	57	1.83	99.79	2338
492038	1837174	Albert Daigle	M	United States	57	1.83	99.79	2338
492039	1837174	Albert Daigle	M	United States	57	1.83	99.79	2338
492040	1838606	Tadas Vismantas	M	Lithuania	25	1.81	86	100285
492041	1838606	Tadas Vismantas	M	Lithuania	25	1.81	86	100285

492042 rows × 15 columns

In [75]:

```
athlete_score['competitorid'].nunique()
```

Out[75]:

```
108760
```

In [76]:

```
athlete_score.drop(columns = ['division_y'], inplace=True)
athlete_score.rename(columns = {'division_x':'division'}, inplace=True)
```

In [77]:

```
athlete_score.drop_duplicates(subset='competitorid', inplace=True)
athlete_score
```

Out[77]:

	competitorid	competitorname	gender	country	age	height_m	weight_kg	overallrank
0	86	Justin Bergh	M	United States	38	1.96	102.06	75515
5	88	Cary Hair	M	United States	35	1.83	86.64	4838
10	93	Leif Edmundson	M	United States	38	1.83	92.99	115466
11	1617	John Mclaughlin	M	United States	52	1.78	84.82	15254
16	1618	Daniel Campbell	M	United States	36	1.73	78.93	20763
...	...	...	...	...	...	...	...	...
492029	1835478	Jean Sebastien Vaillancourt	M	Canada	37	1.83	80.74	76612
492032	1835964	Wisit Kaewsanit	M	Thailand	40	1.67	63	106996
492034	1836343	Andy Cook	M	United States	30	1.65	88.45	101703
492037	1837174	Albert Daigle	M	United States	57	1.83	99.79	2338
492040	1838606	Tadas Vismantas	M	Lithuania	25	1.81	86	100285

108760 rows × 14 columns

In [78]:

```
dummy_df.rename(columns = {'compid':'competitorid'}, inplace=True)
dummy_df
```

Out[78]:

	competitorid	workout1	rank1	workout2	rank2	workout3	rank3	workout4	rank4
0	86	90 reps	103375	374 reps	62328	121 reps - s	104034	120 reps	81206
1	88	13:46	9437	681 reps	7594	118 reps	6936	201 reps	5694
2	1617	175 reps	223	554 reps	159	92 reps	484	167 reps	152
3	1618	171 reps	25081	455 reps	43456	102 reps	18001	186 reps	16004
4	1619	167 reps	982	476 reps	1586	86 reps	1816	162 reps	990
...	...	...	...	...	...	...	...	...	...
91645	1626446	166 reps	29680	309 reps	8996	8:09 - s	90438	121 reps	75621
91646	1626448	160 reps	37760	201 reps	85595	136 reps - s	62761	160 reps	47630
91647	1626452	162 reps	35184	729 reps - s	64571	126 reps - s	66731	120 reps	47097
91648	1626453	145 reps	56939	511 reps	12372	65 reps	51500	160 reps	39355
91649	1626490	147 reps	53558	294 reps	36057	106 reps	14681	165 reps - s	67811

91650 rows × 11 columns

In [79]:

```
athlete_score = athlete_score.merge(dummy_df, on = 'competitorid')
athlete_score.sort_values(['competitorid'])
```

Out[79]:

	competitorid	competitorname	gender	country	age	height_m	weight_kg	overallrank
0	86	Justin Bergh	M	United States	38	1.96	102.06	75515
1	88	Cary Hair	M	United States	35	1.83	86.64	4838
2	1617	John Mclaughlin	M	United States	52	1.78	84.82	15254
3	1618	Daniel Campbell	M	United States	36	1.73	78.93	20763
4	1619	Victor Nicholas	M	United States	46	1.85	85.28	28105
...	...	...	...	...	...	...	...	...
91645	1626446	Tyler Brooker	M	Canada	33	1.80	83.91	18034
91646	1626448	Andrew Smith	M	Australia	34	1.78	75	27569
91647	1626452	Max Moxey	M	United States	17	1.57	52.16	94445
91648	1626453	Eric Williams	M	United States	37	1.83	83.91	62655
91649	1626490	Greg Duval	M	United States	26	1.68	74.84	53231

91650 rows × 24 columns

In [80]:

```
athlete_score.drop(columns = ['workout', 'scaled', 'workoutrank', 'score'], inplace=True)
athlete_score
```

Out[80]:

	competitorid	competitorname	gender	country	age	height_m	weight_kg	overallrank
0	86	Justin Bergh	M	United States	38	1.96	102.06	75515
1	88	Cary Hair	M	United States	35	1.83	86.64	4838
2	1617	John Mclaughlin	M	United States	52	1.78	84.82	15254
3	1618	Daniel Campbell	M	United States	36	1.73	78.93	20763
4	1619	Victor Nicholas	M	United States	46	1.85	85.28	28105
...	...	...	...	...	...	...	...	...
91645	1626446	Tyler Brooker	M	Canada	33	1.80	83.91	18034
91646	1626448	Andrew Smith	M	Australia	34	1.78	75	27569
91647	1626452	Max Moxey	M	United States	17	1.57	52.16	94445
91648	1626453	Eric Williams	M	United States	37	1.83	83.91	62655
91649	1626490	Greg Duval	M	United States	26	1.68	74.84	53231

91650 rows × 20 columns

In [81]:

```
athlete_score.isna().sum()
```

Out[81]:

```
competitorid      0
competitorname    0
gender            0
country           0
age               0
height_m          0
weight_kg         0
overallrank       0
overallscore      0
division          0
workout1          0
rank1             0
workout2          0
rank2             0
workout3          0
rank3             0
workout4          0
rank4             0
workout5          0
rank5             0
dtype: int64
```

In [82]:

```
athlete_score.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 91650 entries, 0 to 91649
Data columns (total 20 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   competitorid    91650 non-null  object
 1   competitorname  91650 non-null  object
 2   gender          91650 non-null  object
 3   country         91650 non-null  object
 4   age             91650 non-null  object
 5   height_m        91650 non-null  object
 6   weight_kg       91650 non-null  object
 7   overallrank     91650 non-null  object
 8   overallscore    91650 non-null  object
 9   division        91650 non-null  object
 10  workout1        91650 non-null  object
 11  rank1           91650 non-null  object
 12  workout2        91650 non-null  object
 13  rank2           91650 non-null  object
 14  workout3        91650 non-null  object
 15  rank3           91650 non-null  object
 16  workout4        91650 non-null  object
 17  rank4           91650 non-null  object
 18  workout5        91650 non-null  object
 19  rank5           91650 non-null  object
dtypes: object(20)
memory usage: 14.7+ MB
```

Converting data types to usable values

In [83]:

```
integers = ['competitorid', 'age', 'overallrank', 'overallscore', 'rank1', 'rank2', 'rank3'
decimals = ['height_m', 'weight_kg']
athlete_score[integers] = athlete_score[integers].astype('int')
athlete_score[decimals] = athlete_score[decimals].astype('float')
athlete_score.dtypes
```

Out[83]:

```
competitorid        int32
competitorname     object
gender             object
country            object
age                 int32
height_m          float64
weight_kg         float64
overallrank         int32
overallscore        int32
division           object
workout1           object
rank1               int32
workout2           object
rank2               int32
workout3           object
rank3               int32
workout4           object
rank4               int32
workout5           object
rank5               int32
dtype: object
```

Create Dataframe for Masters (55+) athletes

Create the dataframe with masters athletes and then remove them from the main dataframe.

In [84]:

```
athlete_score['division'].unique()
```

Out[84]:

```
array(['Men', 'Women (55-59)', 'Women', 'Men (55-59)', 'Men (60+)',
       'Women (60+)'], dtype=object)
```

In [85]:

```
options = ['Women (55-59)', 'Men (55-59)', 'Men (60+)', 'Women (60+)']
masters = athlete_score[athlete_score['division'].isin(options)]
masters
```

Out[85]:

	competitorid	competitorname	gender	country	age	height_m	weight_kg	overallrar
10	1660	Angela Pause	F	Canada	56	1.70	65.32	76
25	1746	Bruce Tappen	M	United States	56	1.75	81.65	308
42	1909	Robert Schubring	M	United States	60	1.75	77.11	10
46	1938	Tori Gray	F	United States	59	1.60	62.14	22
49	1962	Brett Wilson	M	United States	66	1.80	83.91	56
...	...	...	...	...	...	...	...	
91351	1622660	Rich Joline	M	United States	59	1.75	81.65	240
91357	1622705	Jan Raven	F	United States	55	1.70	71.21	63
91484	1624537	Kiera Werner	F	United States	61	1.70	61.69	14
91507	1624906	Edgar Secca	M	Portugal	65	1.71	66	128
91555	1625488	Martin Doorn	M	Netherlands	55	1.87	81	123

4477 rows × 20 columns

In [86]:

```
options = ['Men', 'Women']
athlete_score = athlete_score[athlete_score['division'].isin(options)]
athlete_score = athlete_score.drop('division', axis=1)
athlete_score
```

Out[86]:

	competitorid	competitorname	gender	country	age	height_m	weight_kg	overallrank
0	86	Justin Bergh	M	United States	38	1.96	102.06	75515
1	88	Cary Hair	M	United States	35	1.83	86.64	4838
2	1617	John Mclaughlin	M	United States	52	1.78	84.82	15254
3	1618	Daniel Campbell	M	United States	36	1.73	78.93	20763
4	1619	Victor Nicholas	M	United States	46	1.85	85.28	28105
...	...	...	...	...	...	...	...	...
91645	1626446	Tyler Brooker	M	Canada	33	1.80	83.91	18034
91646	1626448	Andrew Smith	M	Australia	34	1.78	75	27569
91647	1626452	Max Moxey	M	United States	17	1.57	52.16	94445
91648	1626453	Eric Williams	M	United States	37	1.83	83.91	62655
91649	1626490	Greg Duval	M	United States	26	1.68	74.84	53231

87173 rows × 19 columns

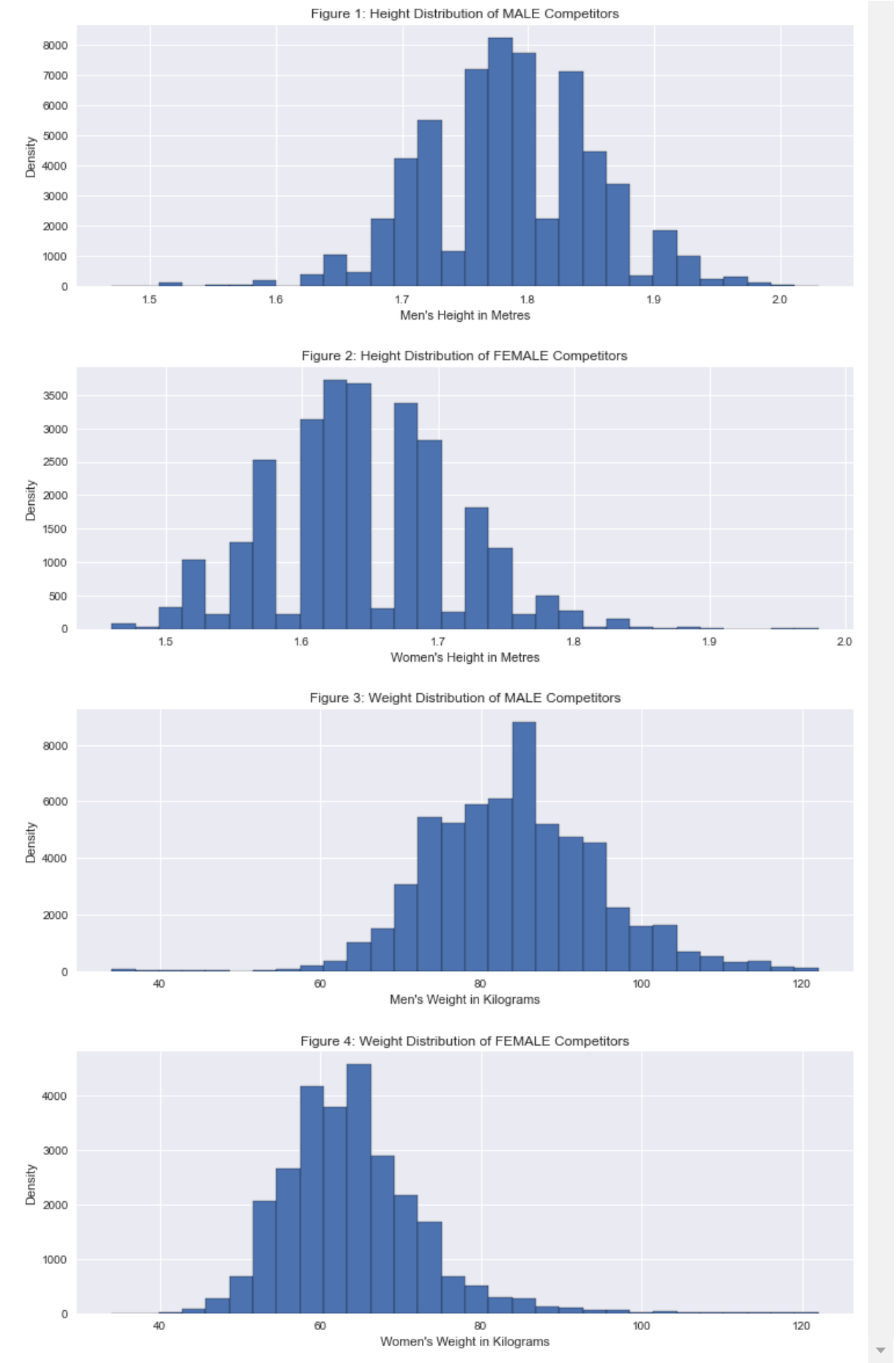
Split into Male and Female

In [87]:

```
# Split to M and F
athlete_m = athlete_score[athlete_score["gender"] == "M"]
athlete_f = athlete_score[athlete_score["gender"] == "F"]
```

In [88]:

```
fig, ax = plt.subplots(4, 1, figsize=(10,16))
fig.tight_layout(h_pad=5)
grid = plt.GridSpec(4, 1, hspace=10)
ax[0].hist(athlete_m["height_m"], bins=30, edgecolor = 'black')
ax[0].set(xlabel = "Men's Height in Metres", ylabel = "Density")
ax[0].set_title("Figure 1: Height Distribution of MALE Competitors")
ax[1].hist(athlete_f["height_m"], bins=30, edgecolor = 'black')
ax[1].set(xlabel = "Women's Height in Metres", ylabel = "Density")
ax[1].set_title("Figure 2: Height Distribution of FEMALE Competitors")
ax[2].hist(athlete_m["weight_kg"], bins=30, edgecolor = 'black')
ax[2].set(xlabel = "Men's Weight in Kilograms", ylabel = "Density")
ax[2].set_title("Figure 3: Weight Distribution of MALE Competitors")
ax[3].hist(athlete_f["weight_kg"], bins=30, edgecolor = 'black')
ax[3].set(xlabel = "Women's Weight in Kilograms", ylabel = "Density")
ax[3].set_title("Figure 4: Weight Distribution of FEMALE Competitors")
plt.show()
```

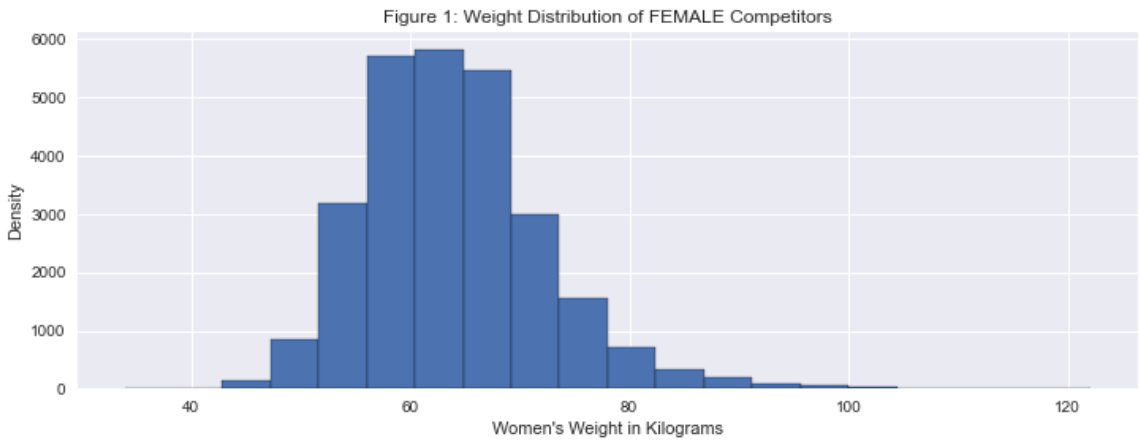


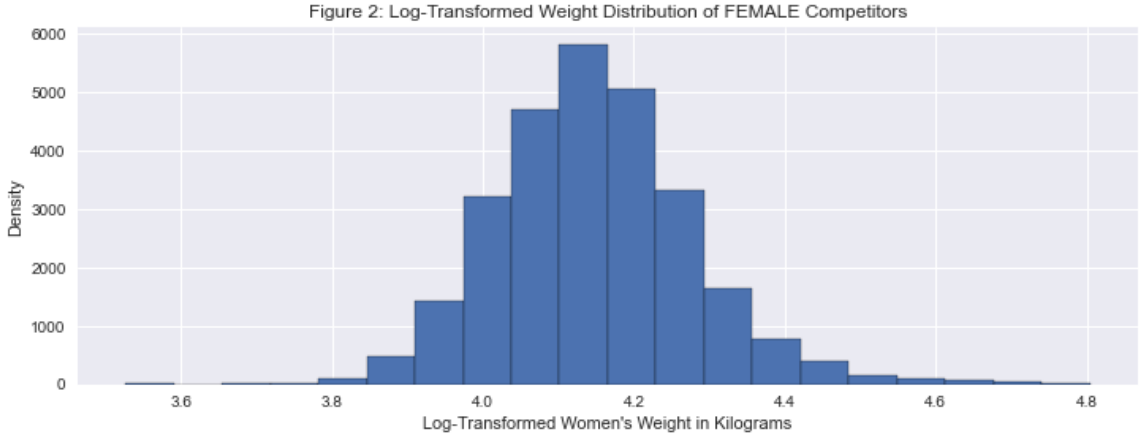
We can see that the height data is dispersed quite odd. Upon further investigation the jumps up and down in height represents standardised markers in feet. Eg. 5ft 7in converts to 1.70m and 5ft 8in is 1.73m etc. So we adjusted the bins to be smaller to eliminate this as a visual above.

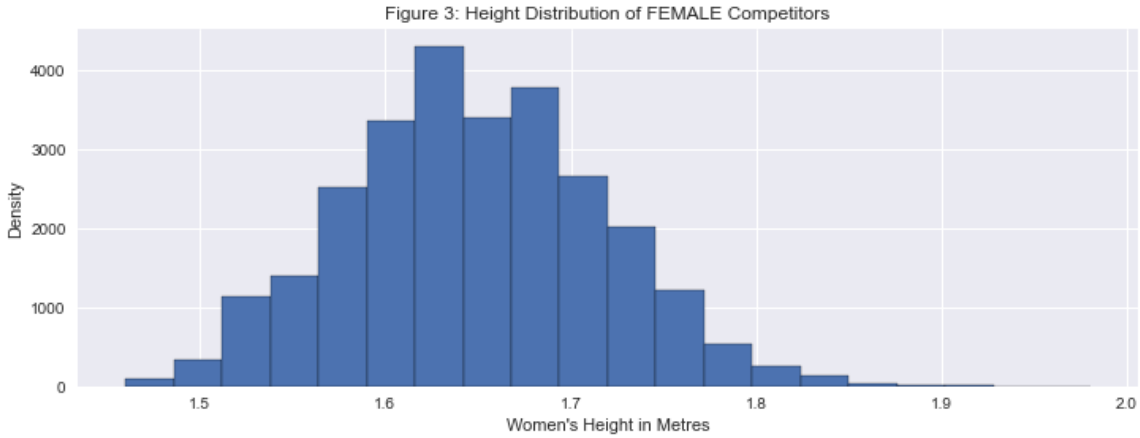
From the histograms above we can see that most are actually normalised already. We will see if we ccan get the women's kilogram graph to be more normalised by taking the log transformation below:

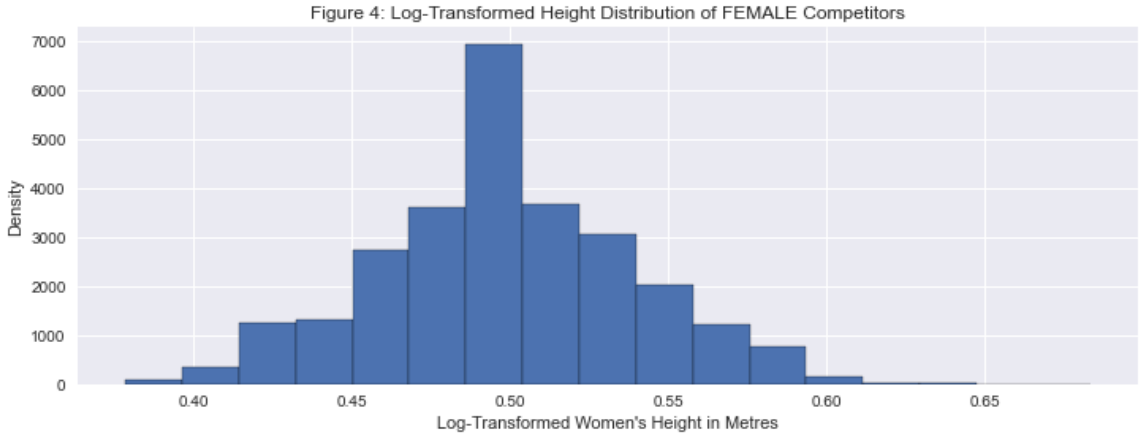
In [89]:

```
X_log = np.log(athlete_f['weight_kg'])
X_1 = athlete_f['weight_kg']
X2_log = np.log(athlete_f['height_m'])
X_2 = athlete_f['height_m']

fig, ax = plt.subplots(4, 1, figsize=(10,16))
fig.tight_layout(h_pad=5)
grid = plt.GridSpec(4, 1, hspace=10)

ax[0].hist(X_1, bins=20, edgecolor = 'black')
ax[0].set(xlabel = "Women's Weight in Kilograms", ylabel = "Density")
ax[0].set_title("Figure 1: Weight Distribution of FEMALE Competitors")
ax[1].hist(X_log, bins=20, edgecolor = 'black')
ax[1].set(xlabel = "Log-Transformed Women's Weight in Kilograms", ylabel = "Density")
ax[1].set_title("Figure 2: Log-Transformed Weight Distribution of FEMALE Competitors")
ax[2].hist(X_2, bins=20, edgecolor = 'black')
ax[2].set(xlabel = "Women's Height in Metres", ylabel = "Density")
ax[2].set_title("Figure 3: Height Distribution of FEMALE Competitors")
ax[3].hist(X2_log, bins=17, edgecolor = 'black')
ax[3].set(xlabel = "Log-Transformed Women's Height in Metres", ylabel = "Density")
ax[3].set_title("Figure 4: Log-Transformed Height Distribution of FEMALE Competitors")

plt.show()
```









Correlation Visualisation

In [90]:

```
athlete_m.corr()
```

Out[90]:

	competitorid	age	height_m	weight_kg	overallrank	overallscore	rank1	rank2
competitorid	1	-0.20	0.02	-0.07	0.28	0.29	0.28	0.36
age	-0.20	1	0.03	0.13	0.30	0.31	-0.38	-0.08
height_m	0.02	0.03	1	0.56	0.08	0.08	0.05	0.01
weight_kg	-0.07	0.13	0.56	1	0.10	0.09	-0.03	-0.03
overallrank	0.28	0.30	0.08	0.10	1	1.00	0.42	0.10
overallscore	0.29	0.31	0.08	0.09	1.00	1	0.43	0.11
rank1	0.28	-0.38	0.05	-0.03	0.42	0.43	1	0.11
rank2	0.36	-0.08	0.01	-0.03	0.10	0.11	0.11	1
rank3	0.38	-0.08	0.01	-0.03	0.11	0.11	0.11	0.14
rank4	0.39	-0.08	0.01	-0.02	0.11	0.11	0.12	0.15
rank5	0.39	-0.08	0.00	-0.03	0.11	0.11	0.11	0.14

In [91]:

```
athlete_f.corr()
```

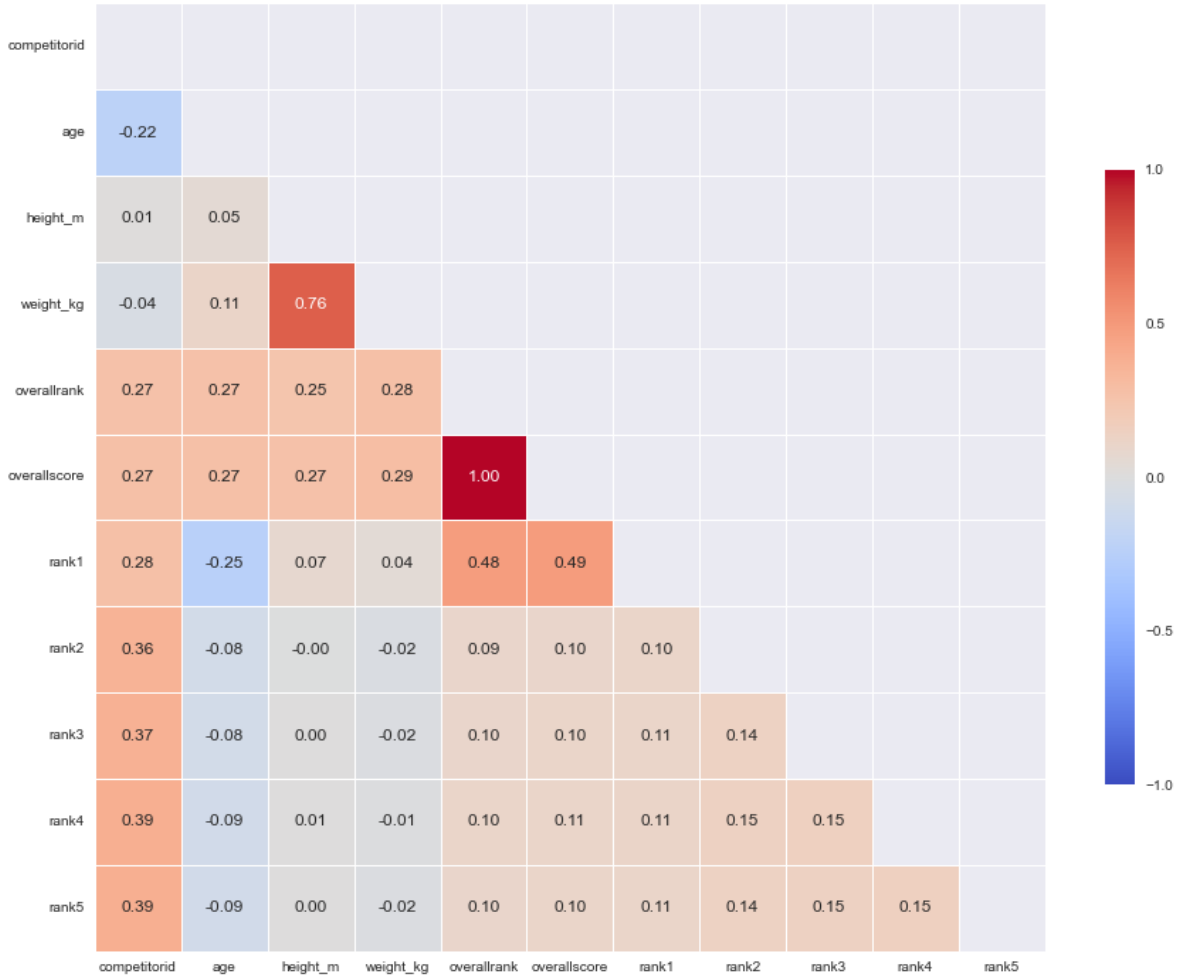
Out[91]:

	competitorid	age	height_m	weight_kg	overallrank	overallscore	rank1	rank2
competitorid	1	-0.27	0.02	-0.01	0.28	0.30	0.29	0.35
age	-0.27	1	0.00	0.02	0.16	0.16	0.18	-0.09
height_m	0.02	0.00	1	0.49	0.04	0.04	0.06	0.00
weight_kg	-0.01	0.02	0.49	1	0.15	0.15	0.15	-0.01
overallrank	0.28	0.16	0.04	0.15	1	1.00	0.81	0.10
overallscore	0.30	0.16	0.04	0.15	1.00	1	0.82	0.11
rank1	0.29	0.18	0.06	0.15	0.81	0.82	1	0.11
rank2	0.35	-0.09	0.00	-0.01	0.10	0.11	0.11	1
rank3	0.37	-0.09	0.00	-0.00	0.11	0.11	0.11	0.14
rank4	0.38	-0.10	0.01	0.00	0.11	0.12	0.12	0.14
rank5	0.38	-0.12	0.01	-0.00	0.11	0.11	0.11	0.13

Correlation Matrix using male and female combined

In [92]:

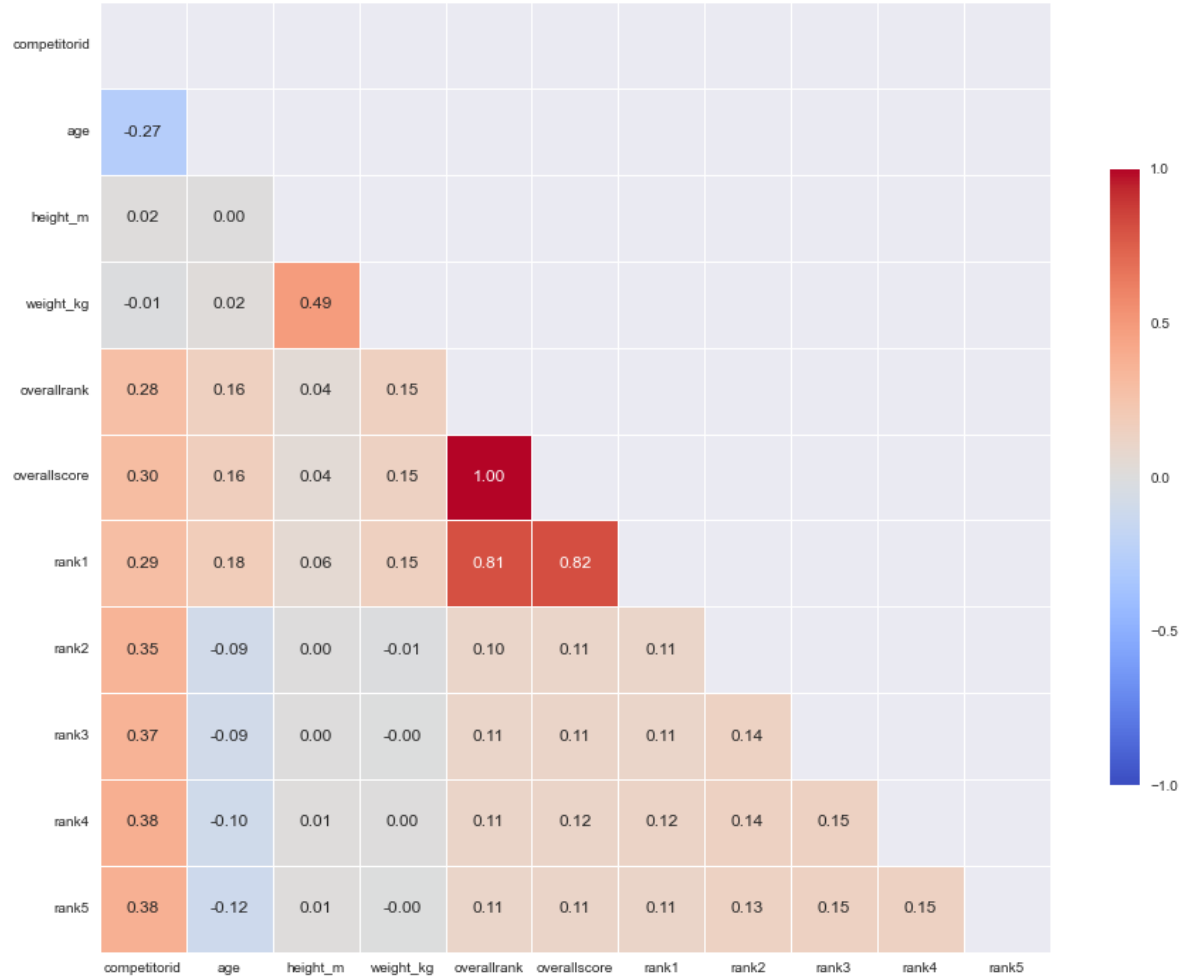
```
# Visulaise with a correlation matrix using heatmap
corr = athlete_score.corr()
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
f, ax = plt.subplots(figsize=(15, 20))
sns.heatmap(corr,
 mask=mask,
 square = True,
 linewidths = .5,
 cmap = "coolwarm",
 cbar_kws = {'shrink': .4,
 "ticks" : [-1, -.5, 0, 0.5, 1]},
 vmin = -1,
 vmax = 1,
 annot = True,
 annot_kws = {"size": 12},
 fmt = ".2f")
ax.set_yticklabels(corr.columns, rotation = 0)
ax.set_xticklabels(corr.columns)
plt.show()
```



Correlation Matrix using female only

In [93]:

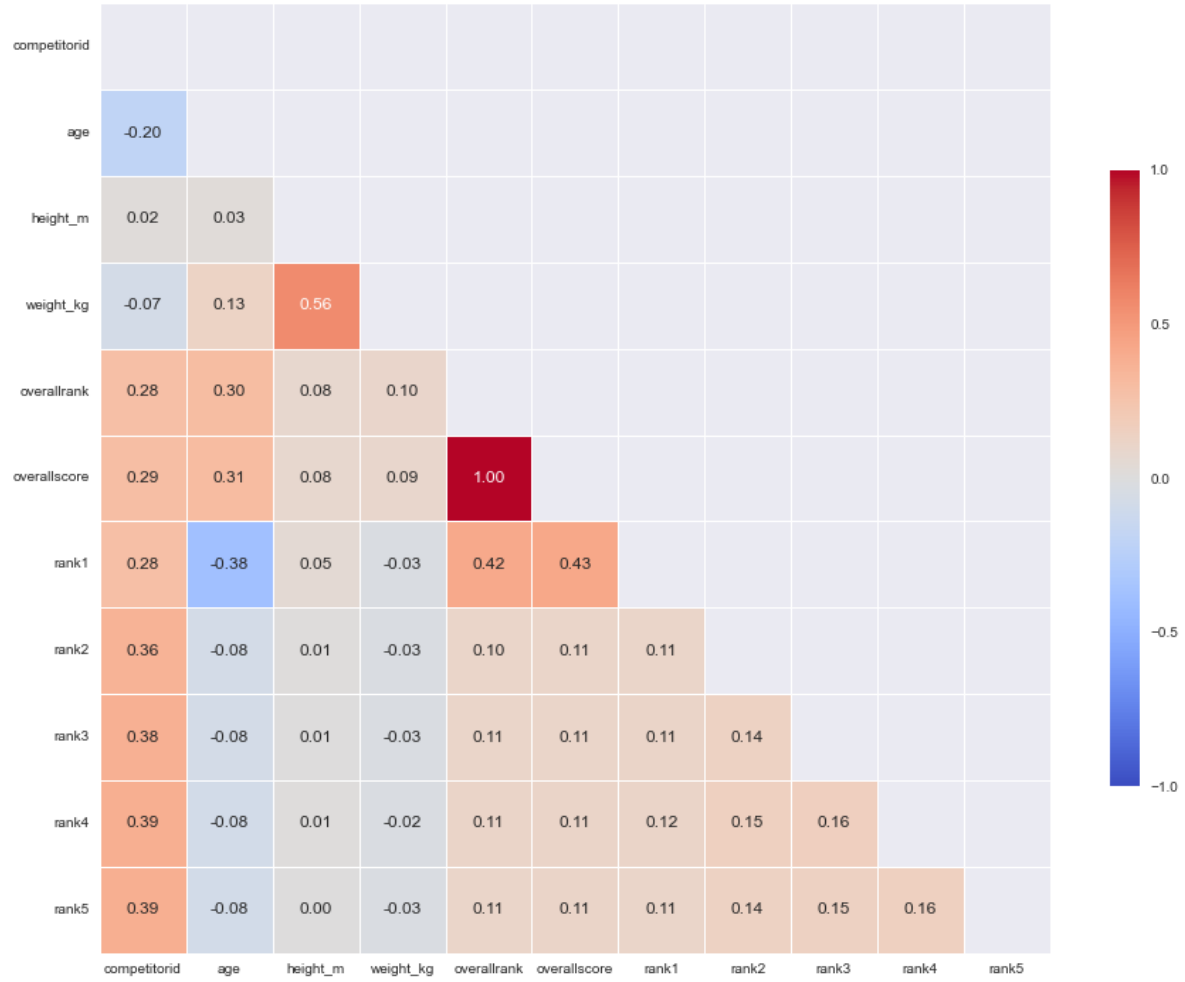
```
# Visulaise with a correlation matrix using heatmap
corr = athlete_f.corr()
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
f, ax = plt.subplots(figsize=(15, 20))
sns.heatmap(corr,
 mask=mask,
 square = True,
 linewidths = .5,
 cmap = "coolwarm",
 cbar_kws = {'shrink': .4,
 "ticks" : [-1, -.5, 0, 0.5, 1]},
 vmin = -1,
 vmax = 1,
 annot = True,
 annot_kws = {"size": 12},
 fmt = ".2f")
ax.set_yticklabels(corr.columns, rotation = 0)
ax.set_xticklabels(corr.columns)
plt.show()
```



Correlation Matrix using male only

In [94]:

```
# Visulaise with a correlation matrix using heatmap
corr = athlete_m.corr()
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
f, ax = plt.subplots(figsize=(15, 20))
sns.heatmap(corr,
 mask=mask,
 square = True,
 linewidths = .5,
 cmap = "coolwarm",
 cbar_kws = {'shrink': .4,
 "ticks" : [-1, -.5, 0, 0.5, 1]},
 vmin = -1,
 vmax = 1,
 annot = True,
 annot_kws = {"size": 12},
 fmt = ".2f")
ax.set_yticklabels(corr.columns, rotation = 0)
ax.set_xticklabels(corr.columns)
plt.show()
```



Reviewing the 3 correlation matices above, there is a standout with female athletes and their performance in workout 1 having a very strong correlation to their overall performance at the end of the 5 week competition, especially in comparison to male athletes. However, the other 4 workouts provide little correlation to their

athletes final overall results and score. It will be interesting to work out why this is.

There is obvioulsy a strong correlatation between height and weight of athletes.

It is interesting that age has very little to do with the overall score and rank. I would have thought this would be much more of a distinguishing point, especially as the age categories are removed and up to the age of 54 are competing against athletes as young as 16.

OLS Model Check

In [95]:

```
#OLS squared data representative of the variables to predict
outcome = 'overallrank'
x_cols = ['age', 'height_m', 'weight_kg', 'rank1', 'rank2', 'rank3', 'rank4', 'rank5']
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model = ols(formula=formula, data=athlete_f).fit()
```

In [96]:

```
model.summary()
```

Out[96]:

OLS Regression Results

Dep. Variable:	overallrank	**R-squared:**	0.653
Model:	OLS	**Adj. R-squared:**	0.653
Method:	Least Squares	**F-statistic:**	6397.
Date:	Fri, 24 Jun 2022	**Prob (F-statistic):**	0.00
Time:	13:31:59	**Log-Likelihood:**	-2.9527e+05
No. Observations:	27247	**AIC:**	5.906e+05
Df Residuals:	27238	**BIC:**	5.906e+05
Df Model:	8		
Covariance Type:	nonrobust		

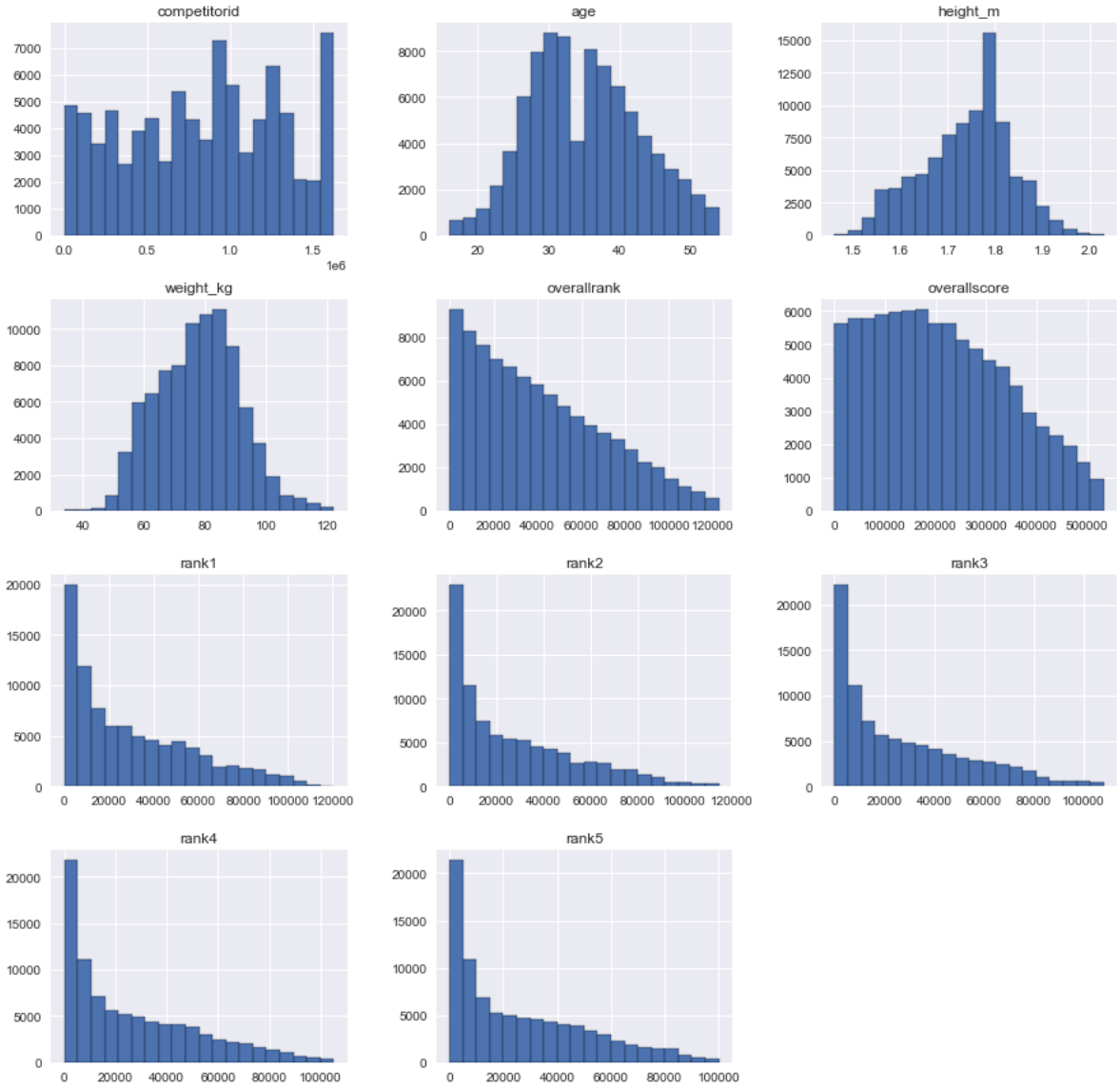
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	8182.3374	1873.913	4.366	0.000	4509.373	1.19e+04
age	43.2793	9.894	4.374	0.000	23.887	62.671
height_m	-8096.7275	1252.689	-6.463	0.000	-1.06e+04	-5641.394
weight_kg	107.1042	10.006	10.704	0.000	87.493	126.716
rank1	0.8415	0.004	209.671	0.000	0.834	0.849
rank2	0.0115	0.003	3.918	0.000	0.006	0.017
rank3	0.0168	0.003	5.631	0.000	0.011	0.023
rank4	0.0100	0.003	3.260	0.001	0.004	0.016
rank5	0.0122	0.003	3.863	0.000	0.006	0.018

Omnibus:	9340.046	**Durbin-Watson:**	1.982
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	33377.794
Skew:	1.724	**Prob(JB):**	0.00
Kurtosis:	7.184	**Cond. No.**	2.02e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.02e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

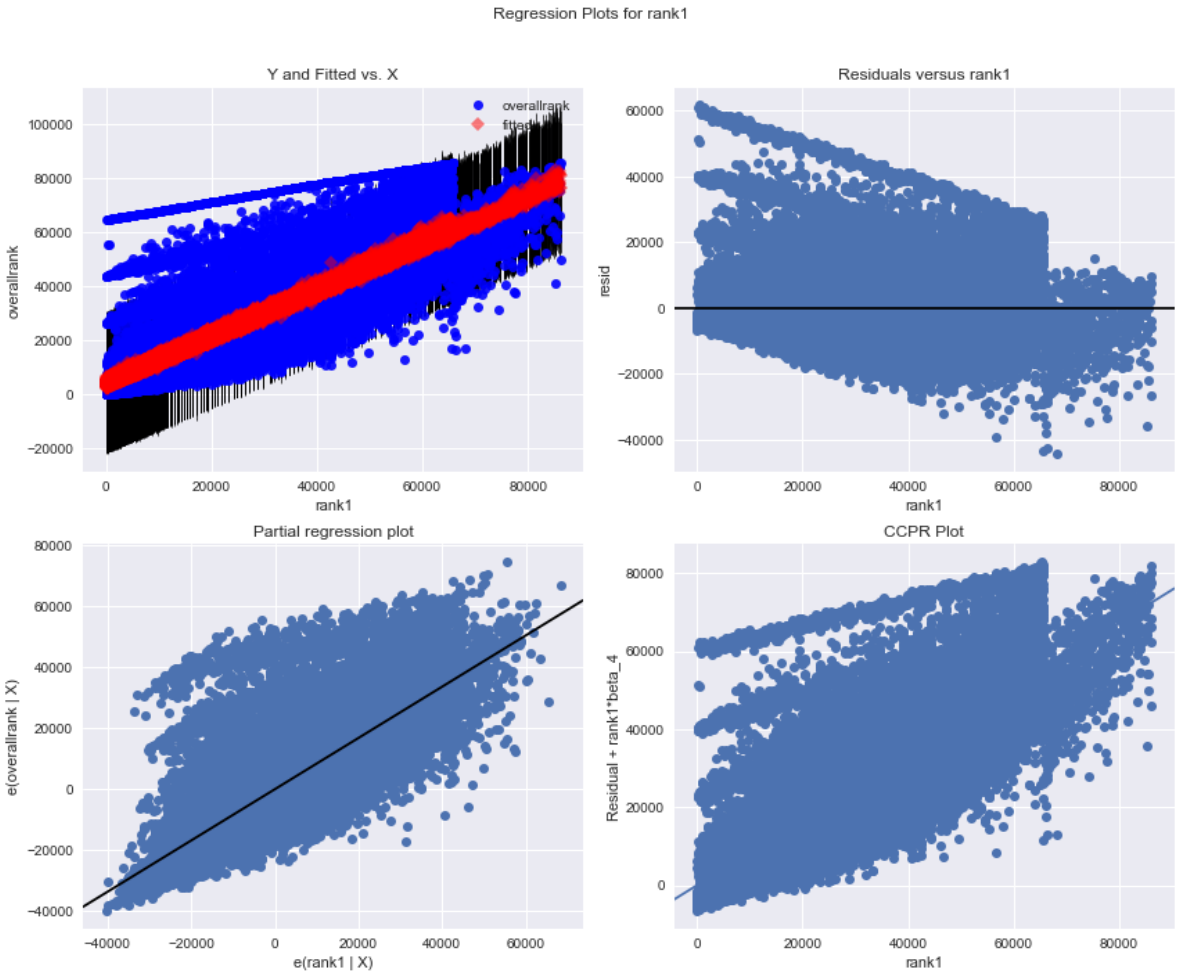
In [97]:

```
athlete_score.hist(figsize=(15,15), bins=20, edgecolor = 'black');
```



In [98]:

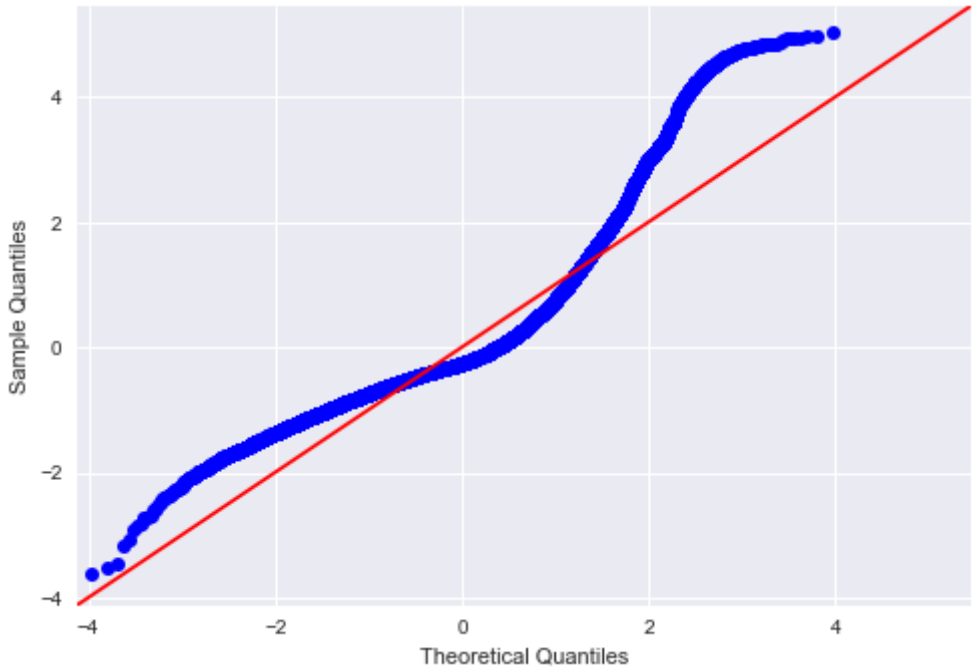
```
fig = plt.figure(figsize=(12,10))
fig = sm.graphics.plot_regress_exog(model, "rank1", fig=fig)
plt.show()
```



Linear Regression Checks

In [99]:

```
# Code for QQ-plot here
import scipy.stats as stats
residuals = model.resid
sm.graphics.qqplot(residuals, dist=stats.norm, line='45', fit=True)
plt.show()
```



Regression Analysis

The data is now clean and in one data frame and now that the baseline model is done we can move on to regression analysis.

In [100]:

```
athlete_score.sort_values('overallrank')
```

Out[100]:

	competitorid	competitorname	gender	country	age	height_m	weight_kg	overallrank
9971	158264	Patrick Vellner	M	Canada	30	1.80	88.45	1
963	8859	Ragnheiður Sara Sigmundsdottir	F	Iceland	27	1.73	69	1
1980	18588	Annie Thorisdottir	F	Iceland	30	1.70	68.95	2
9706	153604	Mathew Fraser	M	United States	30	1.70	88.45	2
7726	120480	Kristin Holte	F	Norway	34	1.62	59	3
...	...	...	...	...	...	...	...	...
1197	10797	Scott Tudge	M	United Kingdom	39	1.74	75.75	122694
11154	185159	Dirk Peters	M	El Salvador	41	1.96	108.86	122694
21822	413917	Christian Orskov	M	Denmark	29	1.85	95	122694
84233	1549443	Mads Nielsen	M	Denmark	30	1.82	82	122694
7817	121772	Heewoon Yun	M	Korea, Republic of	34	1.70	74	122694

87173 rows × 19 columns

In [101]:

```
athlete_score.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 87173 entries, 0 to 91649
Data columns (total 19 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   competitorid    87173 non-null  int32  
 1   competitorname  87173 non-null  object 
 2   gender          87173 non-null  object 
 3   country         87173 non-null  object 
 4   age             87173 non-null  int32  
 5   height_m        87173 non-null  float64
 6   weight_kg       87173 non-null  float64
 7   overallrank     87173 non-null  int32  
 8   overallscore    87173 non-null  int32  
 9   workout1        87173 non-null  object 
 10  rank1           87173 non-null  int32  
 11  workout2        87173 non-null  object 
 12  rank2           87173 non-null  int32  
 13  workout3        87173 non-null  object 
 14  rank3           87173 non-null  int32  
 15  workout4        87173 non-null  object 
 16  rank4           87173 non-null  int32  
 17  workout5        87173 non-null  object 
 18  rank5           87173 non-null  int32  
dtypes: float64(2), int32(9), object(8)
memory usage: 10.3+ MB
```

In [102]:

```
athlete_score.duplicated('overallrank').value_counts()
```

Out[102]:

```
False    64352
True     22821
dtype: int64
```

In [103]:

```
athlete_score['overallrank'].unique()
```

Out[103]:

```
array([75515,  4838, 15254, ..., 94445, 62655, 53231])
```

In [104]:

```
athlete_score.sort_values(['overallrank', 'overallscore'])
```

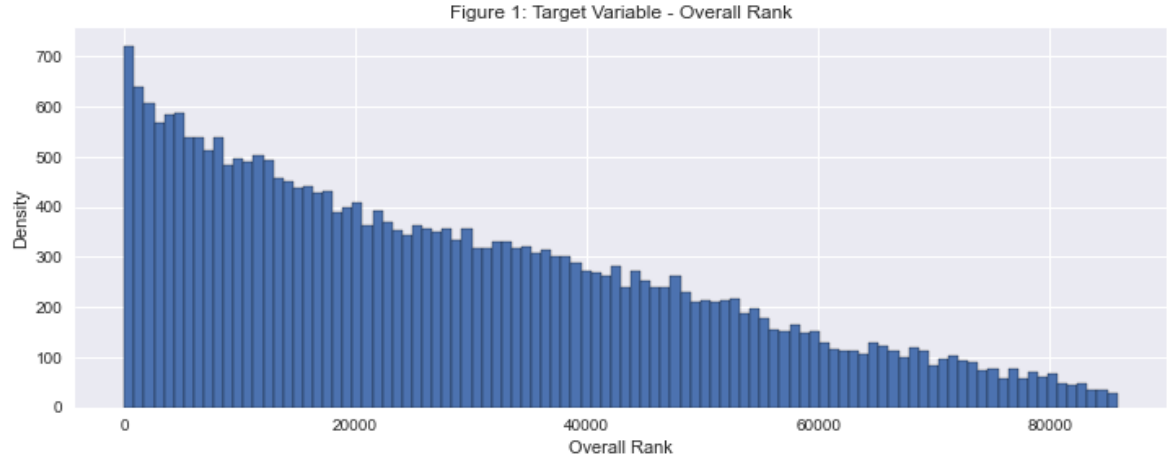
Out[104]:

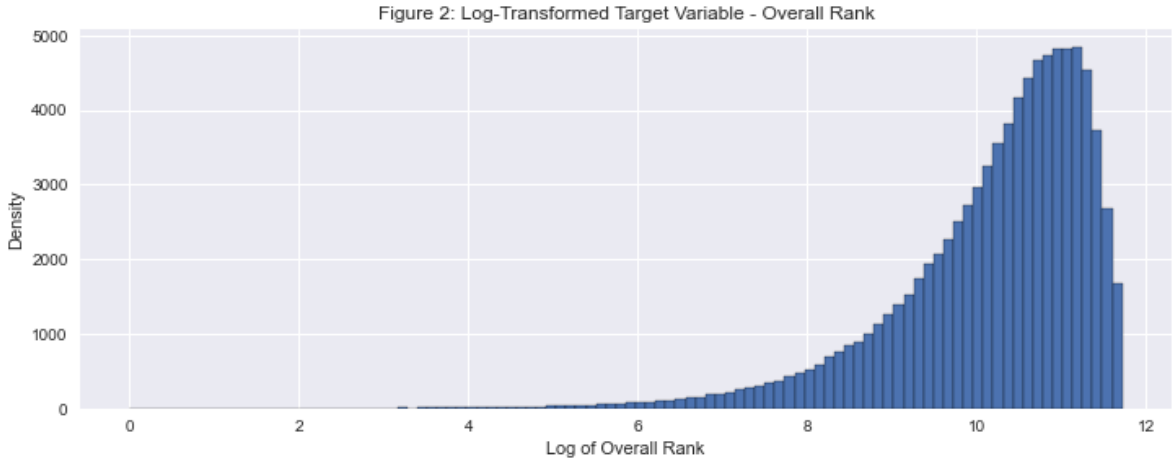
	competitorid	competitorname	gender	country	age	height_m	weight_kg	overallrank
963	8859	Ragnheiður Sara Sigmundsdottir	F	Iceland	27	1.73	69	1
9971	158264	Patrick Vellner	M	Canada	30	1.80	88.45	1
1980	18588	Annie Thorisdottir	F	Iceland	30	1.70	68.95	2
9706	153604	Mathew Fraser	M	United States	30	1.70	88.45	2
7726	120480	Kristin Holte	F	Norway	34	1.62	59	3
...	...	...	...	...	...	...	...	...
14035	246392	JP du Plessis	M	United States	29	1.70	74.39	122694
16343	278829	Dennis Schimanski	M	United States	41	1.70	83.91	122694
21822	413917	Christian Orskov	M	Denmark	29	1.85	95	122694
37102	707203	Tobias Havaas	M	Sweden	41	1.77	69	122694
84233	1549443	Mads Nielsen	M	Denmark	30	1.82	82	122694

87173 rows × 19 columns

In [105]:

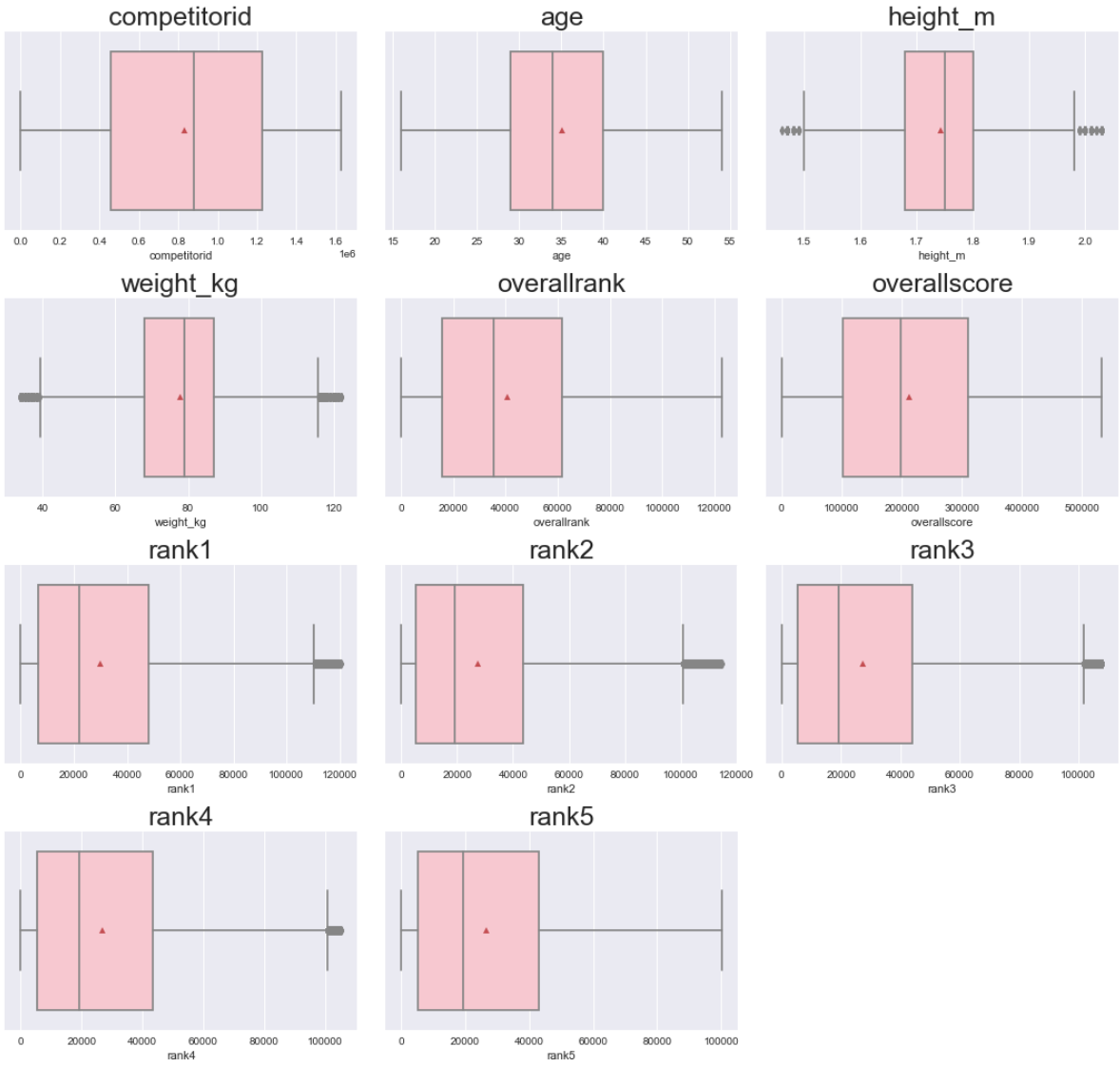
```
X_log = np.log(athlete_score['overallrank'])
X_1 = athlete_f['overallrank']

fig, ax = plt.subplots(2, 1, figsize=(10,8))
fig.tight_layout(h_pad=5)
grid = plt.GridSpec(2, 1, hspace=10)

ax[0].hist(X_1, bins=100, edgecolor = 'black')
ax[0].set(xlabel = "Overall Rank", ylabel = "Density")
ax[0].set_title("Figure 1: Target Variable - Overall Rank")
ax[1].hist(X_log, bins=100, edgecolor = 'black')
ax[1].set(xlabel = "Log of Overall Rank", ylabel = "Density")
ax[1].set_title("Figure 2: Log-Transformed Target Variable - Overall Rank")
plt.show()
```





In [106]:

```
#Plotting a box plot to confirm assumptions
Uni_num = athlete_score.select_dtypes(include=np.number).columns.tolist()
plt.figure(figsize=(15,35))
for i in range(len(Uni_num)):
    plt.subplot(10,3,i+1)
    sns.boxplot(athlete_score[Uni_num[i]],showmeans=True, color='pink')
    plt.tight_layout()
    plt.title(Uni_num[i],fontsize=25)
plt.show()
```



Test the model to predict overall rank

In [107]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# , 'rank1', 'rank2', 'rank3', 'rank4', 'rank5'
continuous = ['height_m','weight_kg', 'age', 'overallrank']
categoricals = ['gender']
# , 'division'

data_cont = athlete_score[continuous]

log_names = [f'{column}_log' for column in data_cont.columns]

data_log = np.log(data_cont)
data_log.columns = log_names

# normalize (subract mean and divide by std)

def normalize(feature):
    return (feature - feature.mean()) / feature.std()

data_log_norm = data_log.apply(normalize)

# one hot encode categoricals
data_ohe = pd.get_dummies(athlete_score[categoricals], drop_first=True)

preprocessed = pd.concat([data_log_norm, data_ohe], axis=1)

X = preprocessed.drop('overallrank_log', axis=1)
y = preprocessed['overallrank_log']

preprocessed.head()
```

Out[107]:

	height_m_log	weight_kg_log	age_log	overallrank_log	gender_M
0	2.17	1.58	0.46	0.90	1
1	0.92	0.68	0.10	-1.46	1
2	0.42	0.56	1.82	-0.47	1
3	-0.10	0.16	0.22	-0.21	1
4	1.12	0.59	1.29	0.05	1

In [108]:

```
# Train-test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
print(len(X_train), len(X_test), len(y_train), len(y_test))
```

```
61021 26152 61021 26152
```

In [109]:

```
# Apply and model the train-test set
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(X_train, y_train)
y_hat_train = linreg.predict(X_train)
y_hat_test = linreg.predict(X_test)

# Calculate training and test residuals
train_residuals = y_hat_train - y_train
test_residuals = y_hat_test - y_test

# Calculate Mean Square Error (MSE)
from sklearn.metrics import mean_squared_error
train_mse = mean_squared_error(y_train, y_hat_train)
test_mse = mean_squared_error(y_test, y_hat_test)
print('Train Mean Squarred Error:', train_mse)
print('Test Mean Squarred Error:', test_mse)
```

```
Train Mean Squarred Error: 0.8786019484413052
Test Mean Squarred Error: 0.8900844629714001
```

In [110]:

```
from sklearn.metrics import mean_squared_error, make_scorer
from sklearn.model_selection import cross_val_score
mse = make_scorer(mean_squared_error)
cv_20_results = cross_val_score(linreg, X, y, cv=20, scoring=mse)
cv_20_results.mean()
```

Out[110]:

```
0.8985511895722302
```

In [111]:

```
#OLS squared data representative of the variables to predict
outcome = 'overallrank'
# 'rank1', 'rank2', 'rank3', 'rank4', 'rank5',
x_cols = ['height_m', 'weight_kg', 'age', 'gender']
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model = ols(formula=formula, data=athlete_score).fit()
model.summary()
```

Out[111]:

OLS Regression Results

Dep. Variable:	overallrank	**R-squared:**	0.155
Model:	OLS	**Adj. R-squared:**	0.155
Method:	Least Squares	**F-statistic:**	3991.
Date:	Fri, 24 Jun 2022	**Prob (F-statistic):**	0.00
Time:	13:32:04	**Log-Likelihood:**	-1.0143e+06
No. Observations:	87173	**AIC:**	2.029e+06
Df Residuals:	87168	**BIC:**	2.029e+06
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-3.688e+04	2365.760	-15.588	0.000	-4.15e+04	-3.22e+04
gender[T.M]	1.211e+04	293.211	41.286	0.000	1.15e+04	1.27e+04
height_m	1.238e+04	1596.173	7.756	0.000	9251.518	1.55e+04
weight_kg	194.6443	11.134	17.482	0.000	172.821	216.467
age	929.9582	11.897	78.168	0.000	906.640	953.276

Omnibus:	4274.705	**Durbin-Watson:**	1.738
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	4326.303
Skew:	0.507	**Prob(JB):**	0.00
Kurtosis:	2.598	**Cond. No.**	2.64e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.64e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [112]:

```
remove=['competitorid','competitorname','overallscore','gender','country','workout1','worko
data = athlete_score.drop(remove, axis=1)
data
```

Out[112]:

	age	height_m	weight_kg	overallrank
0	38	1.96	102.06	75515
1	35	1.83	86.64	4838
2	52	1.78	84.82	15254
3	36	1.73	78.93	20763
4	46	1.85	85.28	28105
...	...	...	...	...
91645	33	1.80	83.91	18034
91646	34	1.78	75	27569
91647	17	1.57	52.16	94445
91648	37	1.83	83.91	62655
91649	26	1.68	74.84	53231

87173 rows × 4 columns

In [113]:

```
data = pd.get_dummies(data)
```

In [114]:

```
#Predicting data values
data_preds = data.drop('overallrank', axis=1)
data_target = data['overallrank']
data_preds.head()
```

Out[114]:

	age	height_m	weight_kg
0	38	1.96	102.06
1	35	1.83	86.64
2	52	1.78	84.82
3	36	1.73	78.93
4	46	1.85	85.28

In [115]:

```
predictors = sm.add_constant(data_preds)
predictors
```

Out[115]:

	const	age	height_m	weight_kg
0	1	38	1.96	102.06
1	1	35	1.83	86.64
2	1	52	1.78	84.82
3	1	36	1.73	78.93
4	1	46	1.85	85.28
...	...	...	...	...
91645	1	33	1.80	83.91
91646	1	34	1.78	75
91647	1	17	1.57	52.16
91648	1	37	1.83	83.91
91649	1	26	1.68	74.84

87173 rows × 4 columns

In [116]:

```
model = sm.OLS(data_target, predictors).fit()
pd.options.display.float_format = lambda x : '{:.0f}'.format(x) if int(x) == x else '{:,.2f
model.summary()
```

Out[116]:

OLS Regression Results

Dep. Variable:	overallrank	**R-squared:**	0.138
Model:	OLS	**Adj. R-squared:**	0.138
Method:	Least Squares	**F-statistic:**	4662.
Date:	Fri, 24 Jun 2022	**Prob (F-statistic):**	0.00
Time:	13:32:04	**Log-Likelihood:**	-1.0151e+06
No. Observations:	87173	**AIC:**	2.030e+06
Df Residuals:	87169	**BIC:**	2.030e+06
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-8.095e+04	2131.824	-37.970	0.000	-8.51e+04	-7.68e+04
age	923.3235	12.011	76.870	0.000	899.781	946.866
height_m	3.54e+04	1510.203	23.438	0.000	3.24e+04	3.84e+04
weight_kg	355.3347	10.533	33.735	0.000	334.690	375.980

Omnibus:	4570.773	**Durbin-Watson:**	1.743
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	5089.023
Skew:	0.574	**Prob(JB):**	0.00
Kurtosis:	2.713	**Cond. No.**	2.39e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.39e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [117]:

```
data_preds_scaled = (data_preds - np.mean(data_preds))/np.std(data_preds)
```

In [118]:

```
data_preds_scaled.describe()
```

Out[118]:

	age	height_m	weight_kg
count	87173	87173	87173
mean	-0.00	-0.00	-0.00
std	1.00	1.00	1.00
min	-2.43	-2.97	-3.21
25%	-0.78	-0.66	-0.73
50%	-0.14	0.08	0.08
75%	0.63	0.61	0.67
max	2.41	3.03	3.22

In [119]:

```
predictors = sm.add_constant(data_preds_scaled)
model = sm.OLS(data_target, predictors).fit()
model.summary()
```

Out[119]:

OLS Regression Results

Dep. Variable:	overallrank	**R-squared:**	0.138
Model:	OLS	**Adj. R-squared:**	0.138
Method:	Least Squares	**F-statistic:**	4662.
Date:	Fri, 24 Jun 2022	**Prob (F-statistic):**	0.00
Time:	13:32:04	**Log-Likelihood:**	-1.0151e+06
No. Observations:	87173	**AIC:**	2.030e+06
Df Residuals:	87169	**BIC:**	2.030e+06
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.081e+04	93.547	436.229	0.000	4.06e+04	4.1e+04
age	7243.3051	94.228	76.870	0.000	7058.620	7427.990
height_m	3361.6514	143.430	23.438	0.000	3080.530	3642.773
weight_kg	4861.5040	144.110	33.735	0.000	4579.049	5143.959

Omnibus:	4570.773	**Durbin-Watson:**	1.743
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	5089.023
Skew:	0.574	**Prob(JB):**	0.00
Kurtosis:	2.713	**Cond. No.**	2.72

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [120]:

```
#Transforming dataset
ss = StandardScaler()
ss.fit(data_preds)
data_preds_st_scaled = ss.transform(data_preds_scaled)
data_preds_scaled.head()
```

Out[120]:

	age	height_m	weight_kg
0	0.37	2.29	1.76
1	-0.01	0.92	0.64
2	2.16	0.40	0.50
3	0.12	-0.13	0.07
4	1.39	1.13	0.54

In [121]:

```
#boolean value declared
np.allclose(data_preds_st_scaled, data_preds_scaled)
```

Out[121]:

```
False
```

In [122]:

```
#obtaining statistic values of target mean
data_target.mean()
```

Out[122]:

```
40808.040803918644
```

In [123]:

```
#created array
data_preds_st_scaled[:5, :]
```

Out[123]:

```
array([[ -4.42489423,   5.79914436,  -5.56730869],
       [ -4.47364209,  -8.61315822,  -5.64968805],
       [ -4.19740424, -14.15635152,  -5.65941117],
       [ -4.4573928 , -19.69954482,  -5.69087773],
       [ -4.29489995,  -6.3958809 ,  -5.65695367]])
```

Model fit in Linear Regression

In [124]:

```
#confirm mlr to run
mlr = LinearRegression()
mlr.fit(data_preds_st_scaled, data_target)
```

Out[124]:

```
LinearRegression()
```

In [125]:

```
#setting co-efficient array
mlr.coef_
```

Out[125]:

```
array([56822.40999889,   319.26966913, 66512.55563351])
```

In [126]:

```
#calculating intercept data value
mlr.intercept_
```

Out[126]:

```
679659.4565217833
```

In [127]:

```
#calculating linear regression score
mlr.score(data_preds_st_scaled, data_target)
```

Out[127]:

```
0.13826195483301362
```

In [128]:

```
#validating though y^test
y_hat = mlr.predict(data_preds_st_scaled)
y_hat
```

Out[128]:

```
array([59781.86405158, 46931.22081429, 60211.23834685, ...,
        8856.63340725, 47807.80408827, 29119.03864396])
```

In [129]:

```
#find values for array
data_preds_st_scaled.shape
```

Out[129]:

```
(87173, 3)
```

In [130]:

```
#staged array
base_pred = np.zeros(3).reshape(1, -1)
base_pred
```

Out[130]:

```
array([[0., 0., 0.]])
```

In [131]:

```
#predicted array
mlr.predict(base_pred)
```

Out[131]:

```
array([679659.45652178])
```

In [132]:

```
#overall metrics predicted data target metric
metrics.r2_score(data_target, mlr.predict(data_preds_st_scaled))
```

Out[132]:

```
0.13826195483301362
```

In [133]:

```
#setting predictors location
data_pred = data.iloc[:,0:12]
data_pred.head()
```

Out[133]:

	age	height_m	weight_kg	overallrank
0	38	1.96	102.06	75515
1	35	1.83	86.64	4838
2	52	1.78	84.82	15254
3	36	1.73	78.93	20763
4	46	1.85	85.28	28105

In [134]:

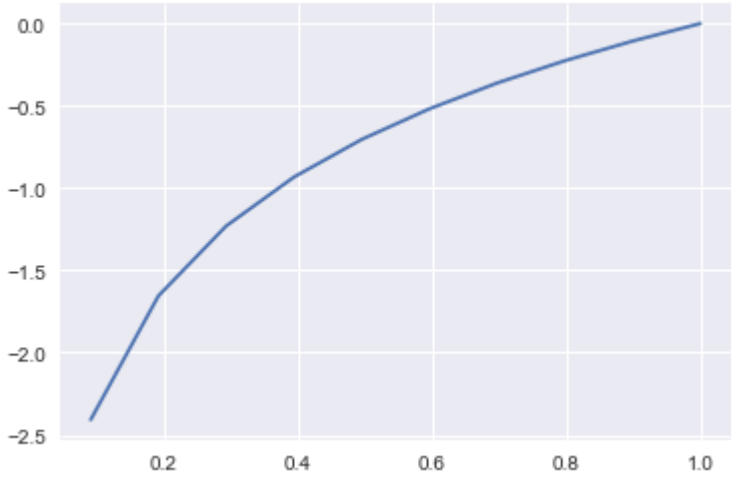
```
#visualse value of correlation percentage
data_pred.corr()
```

Out[134]:

	age	height_m	weight_kg	overallrank
age	1	0.05	0.11	0.27
height_m	0.05	1	0.76	0.25
weight_kg	0.11	0.76	1	0.28
overallrank	0.27	0.25	0.28	1

In [135]:

```
#Log trandsformation visual representation
x = np.linspace(start=-100, stop=1, num=10**3)
y = np.log(x)
plt.plot(x, y);
```



Further Analysis

It seems we can't predict a score based on someones age specifically, but can we find the mean of the different quartiles of scores and then compare them to Games athletes?

In [136]:

```
data = data.sort_values('overallrank', ascending=True)
data
```

Out[136]:

	age	height_m	weight_kg	overallrank
9971	30	1.80	88.45	1
963	27	1.73	69	1
1980	30	1.70	68.95	2
9706	30	1.70	88.45	2
7726	34	1.62	59	3
...	...	...	...	...
1197	39	1.74	75.75	122694
11154	41	1.96	108.86	122694
21822	29	1.85	95	122694
84233	30	1.82	82	122694
7817	34	1.70	74	122694

87173 rows × 4 columns

In [137]:

```
data['overallrank'].quantile([0.25,0.5,0.75,1])
```

Out[137]:

```
0.25    15605
0.50    35375
0.75    61486
1      122694
Name: overallrank, dtype: float64
```

In [138]:

```
q1 = data.iloc[:15605]
q2 = data.iloc[15606:35375]
q3 = data.iloc[35376:61486]
q4 = data.iloc[61487:122694]
```

In [139]:

```
print("Top Quartile\n", q1.mean())
print("\nSecond Quartile\n", q2.mean())
print("\nThird Quartile\n", q3.mean())
print("\nBottom Quartile\n", q4.mean())
```

```
Top Quartile
 age              31.67
height_m           1.72
weight_kg         74.51
overallrank    5,182.39
dtype: float64

Second Quartile
 age               33.74
height_m           1.73
weight_kg         75.04
overallrank   18,799.09
dtype: float64

Third Quartile
 age               35.67
height_m           1.74
weight_kg         76.92
overallrank   40,798.99
dtype: float64

Bottom Quartile
 age               37.60
height_m           1.77
weight_kg         83.28
overallrank   79,400.97
dtype: float64
```

In [140]:

```
games_athletes = games_athletes.sort_values('overallrank', ascending=True)
```

In [141]:

```
games_athletes = games_athletes.iloc[:,3:10]
```

In [142]:

```
games_athletes = games_athletes.drop(games_athletes[['country','affiliatename']], axis=1)
```

In [143]:

```
games_females = games_athletes[games_athletes['gender'] == 'F']
games_males = games_athletes[games_athletes['gender'] == 'M']
```

In [144]:

```
games_athletes
```

Out[144]:

	gender	age	height_m	weight_kg	overallrank
86729	F	27	1.73	69	1
198613	M	30	1.80	88.45	1
86730	F	30	1.70	68.95	2
198614	M	30	1.70	88.45	2
86731	F	34	1.62	59	3
...	...	...	...	...	...
113400	F	27	1.52	57	26672
118541	F	25	1.48	52	31813
123373	F	23	1.70	58.97	36644
237037	M	30	1.75	80.29	38424
288007	M	33	1.80	89	89395

271 rows × 5 columns

In [145]:

```
games_athletes = games_athletes.iloc[:,1:5]
games_athletes.sort_values('overallrank', ascending=True)
```

Out[145]:

	age	height_m	weight_kg	overallrank
86729	27	1.73	69	1
198613	30	1.80	88.45	1
86730	30	1.70	68.95	2
198614	30	1.70	88.45	2
86731	34	1.62	59	3
...	...	...	...	...
113400	27	1.52	57	26672
118541	25	1.48	52	31813
123373	23	1.70	58.97	36644
237037	30	1.75	80.29	38424
288007	33	1.80	89	89395

271 rows × 4 columns

In [146]:

```
print("Games Athletes\n", games_athletes.mean())
```

```
Games Athletes
 age              29.16
height_m          1.71
weight_kg        75.93
overallrank   3,501.61
dtype: float64
```

Comparing the above we can see that the higher the athlete place the lower all of their key features were, age, height and weight were all lower for higher performance.

These averages were combined as Male and Female. It would be worth comparing these results as seperate genders too.

Female Athletes

In [147]:

```
athlete_f = athlete_f.iloc[:,4:8]
athlete_f = athlete_f.sort_values('overallrank', ascending=True)
athlete_f
```

Out[147]:

	age	height_m	weight_kg	overallrank
963	27	1.73	69	1
1980	30	1.70	68.95	2
7726	34	1.62	59	3
10202	26	1.63	58	4
15444	29	1.63	61.23	5
...	...	...	...	...
35270	39	1.68	77.11	85349
35332	42	1.57	52.16	85355
2032	35	1.68	62.60	85355
62485	32	1.65	70.31	85392
85373	35	1.52	80	85808

27247 rows × 4 columns

In [148]:

```
athlete_f['overallrank'].quantile([0.25,0.5,0.75,1])
```

Out[148]:

```
0.25   10,293.50
0.50       24033
0.75       42360
1          85808
Name: overallrank, dtype: float64
```

In [149]:

```
q1 = athlete_f.iloc[:6812]
q2 = athlete_f.iloc[6813:13624]
q3 = athlete_f.iloc[13625:20435]
q4 = athlete_f.iloc[20436:27247]
```

In [150]:

```
print("Female Top Quartile\n", q1.mean())
print("\nFemale Second Quartile\n", q2.mean())
print("\nFemale Third Quartile\n", q3.mean())
print("\nFemale Bottom Quartile\n", q4.mean())
```

```
Female Top Quartile
 age              32.39
height_m           1.64
weight_kg         62.95
overallrank    4,837.77
dtype: float64

Female Second Quartile
 age              34.17
height_m           1.65
weight_kg         63.09
overallrank   16,763.24
dtype: float64

Female Third Quartile
 age              35.43
height_m           1.65
weight_kg         63.76
overallrank   32,737.57
dtype: float64

Female Bottom Quartile
 age              35.84
height_m           1.65
weight_kg         66.27
overallrank   57,667.13
dtype: float64
```

In [151]:

```
print("Games Female Athletes\n", games_females.mean())
```

```
Games Female Athletes
 age              29.35
height_m           1.64
weight_kg         64.21
overallrank    2,933.21
dtype: float64
```

Male Athletes

In [152]:

```
athlete_m = athlete_m.iloc[:,4:8]
athlete_m = athlete_m.sort_values('overallrank', ascending=True)
athlete_m
```

Out[152]:

	age	height_m	weight_kg	overallrank
9971	30	1.80	88.45	1
9706	30	1.70	88.45	2
27423	30	1.71	81	3
5303	27	1.78	83.91	4
24238	26	1.75	89.36	5
...	...	...	...	...
84233	30	1.82	82	122694
10557	39	1.78	81.65	122694
5682	27	1.80	91.17	122694
1197	39	1.74	75.75	122694
16343	41	1.70	83.91	122694

59926 rows × 4 columns

In [153]:

```
athlete_m['overallrank'].quantile([0.25,0.5,0.75,1])
```

Out[153]:

```
0.25   19,655.25
0.50       42584
0.75   70,618.50
1         122694
Name: overallrank, dtype: float64
```

In [154]:

```
q1 = athlete_m.iloc[:14981]
q2 = athlete_m.iloc[14982:29963]
q3 = athlete_m.iloc[29964:44944]
q4 = athlete_m.iloc[44945:59926]
```

In [155]:

```
print("Male Top Quartile\n", q1.mean())
print("\nMale Second Quartile\n", q2.mean())
print("\nMale Third Quartile\n", q3.mean())
print("\nMale Bottom Quartile\n", q4.mean())
```

```
Male Top Quartile
 age              31.86
height_m           1.78
weight_kg         83.64
overallrank   9,554.44
dtype: float64

Male Second Quartile
 age               34.63
height_m           1.78
weight_kg         83.02
overallrank   30,798.78
dtype: float64

Male Third Quartile
 age               36.74
height_m           1.79
weight_kg         84.20
overallrank   56,091.27
dtype: float64

Male Bottom Quartile
 age               38.25
height_m           1.79
weight_kg         86.17
overallrank   90,082.63
dtype: float64
```

In [156]:

```
print("Games Male Athletes\n", games_males.mean())
```

```
Games Male Athletes
 age              28.99
height_m           1.77
weight_kg         86.43
overallrank    4,010.39
dtype: float64
```

Conclusion

I was unable to produce a predictive model based on my original parameters of using height, weight and age as predictors of results. This is obviously due to many other parameters around fitness. I have broken the results into male and female components and compared the mean of the different quartiles and used them to compare against one another and the top athletes in the world (Games Athletes).

As expected the better the result the younger the athlete. Interestingly the weight didnt fluctuate between the quartiles as much as I would have expected. It was more significant in a combined male and female comparison.

Future Work

Better use of predictive modelling making better use of workout information would be a great way to advance this analysis.