PES Master is a source for all Pro Evolution Soccer games, including all PES 2020 Full Game teams and players. It is possible to make in-depth searches of the Pro Evolution Soccer database using the PES 2020 Advanced Search feature. In the advanced search part search can be made with several way and results are shown as Player's name, club, nationality, age, height, position, overall power and specific powers. (pas, shoot, def and etc.) Just for PES 2020 there are 26039 players records and 21665 of them are unique.

Website build with JavaScript and it is using scripts to bring data from its database. In advanced search part at the beginning there are 30 player's records and every time we scroll down to the end of the page script makes a request to its server in order to show next 30 players. It is possible to add filters while using advanced search with drop-down menu, segmented controls and sliders.

Selenium

Selenium starts with main page and goes to the Advanced Search part by clicking "All PES 2020 Players". After that it clicks to remove filter "MyClub: yes" and then chooses from "Featured Players"[(1)] "no" to avoid having featured players in the list. Since I want to scrap top 100 players in every position it clicks in filter part to "Main Position" and choses one by one every position. For letting us know at which positions selenium is at the moment it prints the position. After it clicks to position it sends the position name and an empty list to the "scraper function".

Function starts with creating a data frame with 12 columns to store the data. And then creating values for loop like how many players should be gathered in every position(top_number) and loop iterators i (how many players scraped already) and k (number of row). Loop starts with while and in the beginning increases the number of row (k) by one. Next there is a if statement which works as in every 25 row clicks to the "END" key in order to go to the end of the page for retrieving further data. Every time it does that it prints the page number for letting us know at which page currently selenium is.

Before it starts to scrap everything from the current row it checks the name of the player and controls if there is already this player in the list. If it is, it does not scrap that row and continue to next one. If it is not on the list, first adds that name to the names list and then it scraps every data from that row and adds each value to the player list. At the end of every row it adds that player to the data frame(all_players) and increases i (number of players) by one. And this loop continues until number of players reaches top_number and when it does it exports data frame to csv with position name.

After it finishes scraping all positions it prints the time it took for selenium to scrap and then quits from browser.

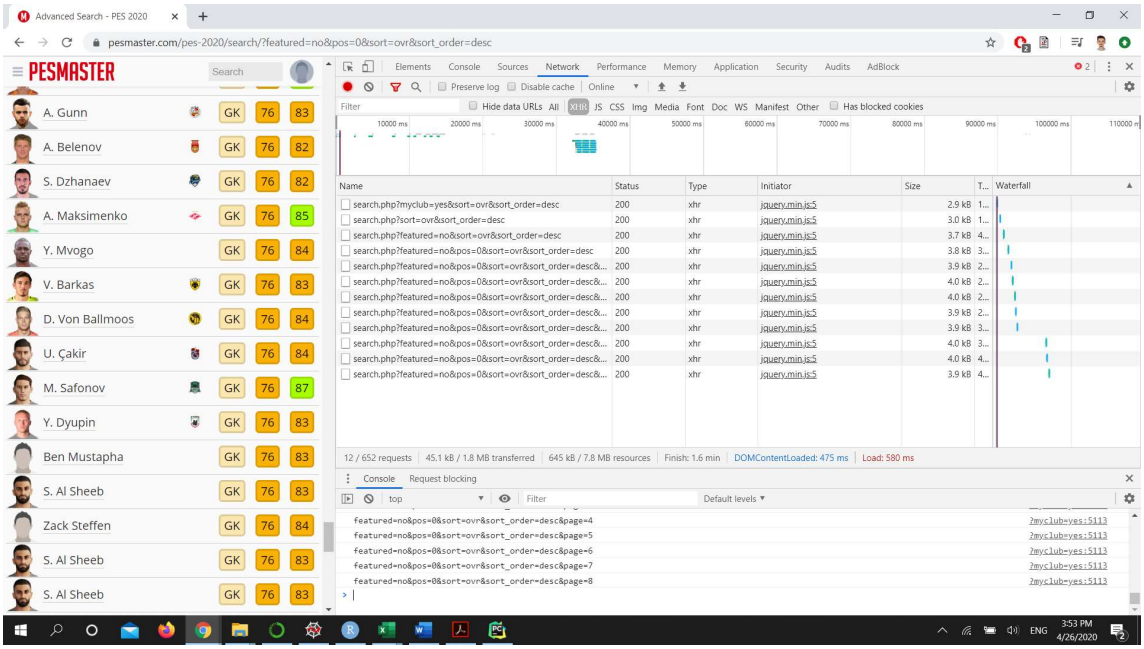
[(1)] Featured Players are available in PES 2020 myClub for one week each. They feature boosted stats compared to the regular 'cards' and can only be obtained using 'Special Agents'.

Scrapy

To run scrapy and write scraped data to csv file:

```
scrapy crawl pes_players -o pes_players.csv
```

I was trying to find the way to scrap pesmaster.com with scrapy, I realized that every time website request further data from its database it creates sublinks and takes data from there and shows in the browser.



So, I decided to reproduce this action in the scrapy with loop and add to the start_urls.

I created scrapy item in order to handle scrapped data from website with the column names from table in the website.

After that I created the spider with the settings can be seen below. Feed Export Fields specifies exported fields and order of them. Without this it was ordering alphabetical order

```
class PesmasterSpider(scrapy.Spider):
    name = 'pes_players'
    allowed_domains = ['pesmaster.com/']
    custom_settings = {
# specifies exported fields and order of them. without this settings columns were
in alphabetical order
        'FEED_EXPORT_FIELDS': ["name",
"nationality","age","height","pos","ovr","pot","shoot","str","defn","spd","dri"]
    }
```

Then I created a pipeline in order to drop duplicate players from the items. Which can be seen in pipelines.py

```
class DuplicatesPipeline:
    def __init__(self):
        self.ids_seen = set()
    def process_item(self, item, spider):
        if item['name'] in self.ids_seen:
            raise DropItem("Duplicate item found: %s" % item)
        else:
            self.ids_seen.add(item['name'])
            return item
```

Then I activate it from settings.py

```
ITEM_PIPELINES = {
    'pesmaster.pipelines.PesmasterPipeline': 300,
    'pesmaster.pipelines.DuplicatesPipeline': 300,
}
```

I also changed the settings of concurrent request in order to scrap players in order for not having mixed data. Without this settings it was going through 16 pages in the same time so the output was in random order.

```
# Configure maximum concurrent requests performed by Scrapy (default: 16)
CONCURRENT_REQUESTS = 1
```

At the end in the parse part I show xpaths of every value I want it to be scraped and add them to their places in scrapy item.

For collecting every player in the database, I created another spider with same way. Can be found in scrap_all.py and its name is scrap_all.

Output

The output I got is the best 100 players by their overall powers in PES 2020 game according to the main positions they are playing. In total 1300 rows and 12 columns.

name	nationality	age	height	pos	ovr	pot	shoot	str	defn	spd	dri
O. Kahn	Germany	34	188	GK	93	94	54	92	52	70	53
Casillas	Spain	26	185	GK	92	94	57	89	63	76	63
P. Čech	Czech Republic	29	196	GK	91	92	43	85	46	59	44
Kahn	Germany	34	188	GK	91	93	53	88	51	69	52
David de Gea	Spain	29	193	GK	90	93	43	82	43	68	43
J. Oblak	Slovenia	26	188	GK	90	94	40	87	54	65	54
Alisson	Brazil	27	191	GK	90	94	43	87	66	65	65
Schmeichel	Denmark	56	193	GK	90	92	47	91	51	72	59
M. Neuer	Germany	33	193	GK	89	93	41	88	60	61	57
Ederson	Brazil	26	188	GK	89	94	44	84	62	70	71

Exploratory Data Analysis

Mean of player's powers by positions:

pos	ovr	pot	shoot	str	defn	spd	dri
GK	82.89	87.97	44.91	82.09	56.52	64.01	52.43
CB	84.32	90.65	61.18	86.07	87.25	77.17	69.82
LB	79.78	87.95	63.54	71.44	72.31	81.51	76.66
RB	79.81	88.12	63.08	72.39	74.04	82.46	75.33
DMF	81.76	89.15	66.01	79.46	79.07	72.65	73.88
CMF	83.61	90.42	72.91	75.22	72.11	75.77	80.91
LMF	77.31	86.18	71.25	68.97	56.47	81.59	80.18
RMF	77.25	86.19	71.43	68.81	57.29	80.14	79.49
AMF	82.54	89.86	76.67	70.19	54.48	76.85	83.93
LWF	80.94	89.23	75.8	67.57	51.34	83.99	84.63
RWF	80.66	88.88	75.5	67.85	51.7	83.75	83.54
SS	77.35	86.83	75.16	70.24	48.03	78.91	80.15
CF	84.93	91.69	86.24	79.23	51.48	82.06	80.39

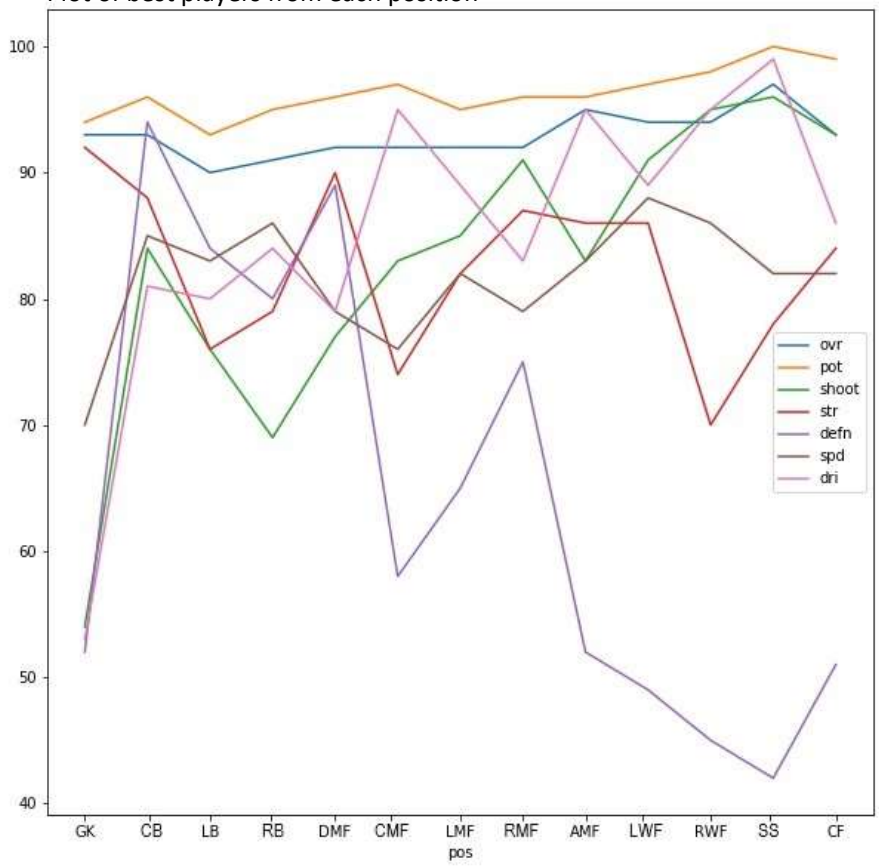
Description of data grouped by position:

	ovr	ovr	ovr	ovr	ovr	ovr	ovr	ovr
	count	mean	std	min	25%	50%	75%	max
pos								
GK	100	82.89	3.469623	79	80	82	84	93
CB	100	84.32	2.795668	81	82	84	86	93
LB	100	79.78	2.942479	77	78	79	81	90
RB	100	79.81	2.646496	77	78	79	81	91
DMF	100	81.76	3.43223	78	79	80.5	84	92
CMF	100	83.61	2.711349	80	81.75	83	85	92
LMF	100	77.31	3.732725	74	75	76	79	92
RMF	100	77.25	3.276408	74	75	77	78	92
AMF	100	82.54	3.775038	78	79	81.5	85	95
LWF	100	80.94	3.603646	77	78	79.5	83	94
RWF	100	80.66	3.432524	77	78	80	83	94
SS	100	77.35	6.05092	72	74	74	79	97
CF	100	84.93	3.337134	80	82	84	88	93

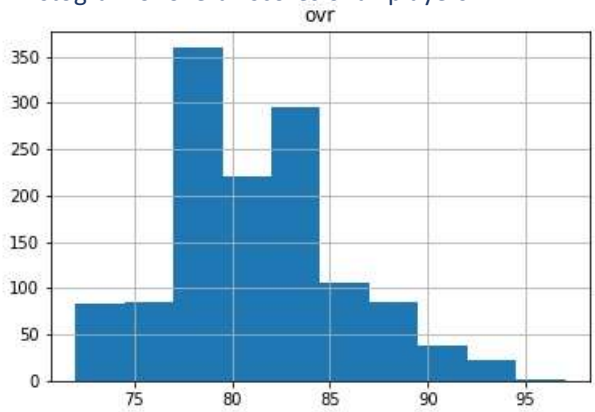
Description of overall power and means of specific power by age:

	ovr	ovr	ovr	pot	shoot	str	defn	spd	dri
	count	mean	std	mean	mean	mean	mean	mean	mean
age									
18	4	77	2.309401	91.5	71.5	67.25	53.75	78.75	77.75
19	10	78.8	3.457681	91.8	74.2	65.7	46.7	82.2	82.8
20	21	78.14286	4.186714	90.14286	66.85714	71.47619	55.38095	78.52381	76.28571
21	29	79.62069	4.678322	90.31034	70.96552	69.62069	57.55172	82.89655	81
22	75	79.26667	3.849652	89.54667	71.05333	70.49333	57.34667	81.94667	79.54667
23	83	80.3494	3.160093	89.45783	70.53012	73.18072	64.07229	81.21687	78.54217
24	81	81.16049	4.187651	89.7284	70.2716	74.23457	62.80247	80.40741	77.98765
25	110	81.44545	4.41554	89.70909	70.08182	72.63636	62.25455	79.84545	78.22727
26	113	81.14159	4.701571	89.02655	70.27434	73.60177	62.0177	79.69027	78.78761
27	123	80.85366	4.79187	88.47967	69.31707	71.91057	60.4065	79.38211	77.98374
28	125	81.272	4.456557	88.616	70.88	73.984	63.184	79.816	77.904
29	99	81.15152	4.479804	88.40404	68.70707	73.58586	63.44444	77.86869	75.90909
30	97	80.47423	3.897362	87.7732	68.1134	74.98969	65.17526	77.10309	74.73196
31	79	81.5443	4.651433	88.6962	69.07595	75.49367	65.07595	77.25316	76.77215
32	65	82.26154	4.697995	88.8	71.61538	75.73846	62.64615	76.55385	76.6
33	59	81.30508	3.962123	87.83051	66.0339	76.11864	65	72.71186	71.94915
34	37	81.18919	4.742467	86.91892	64.18919	75.81081	64.97297	73.86486	70.37838
35	24	80.95833	3.342469	86.5	64.08333	77.33333	63.875	71.70833	71.58333

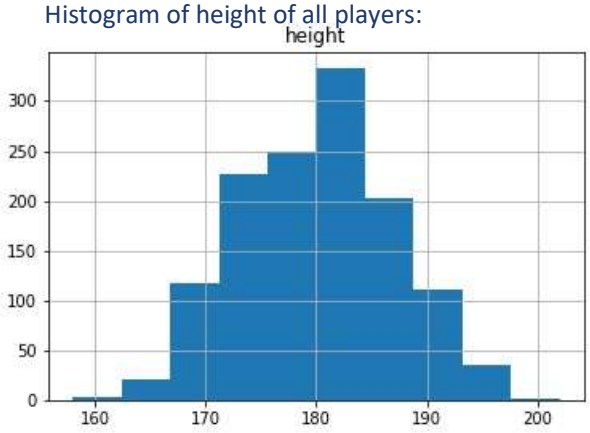
Plot of best players from each position



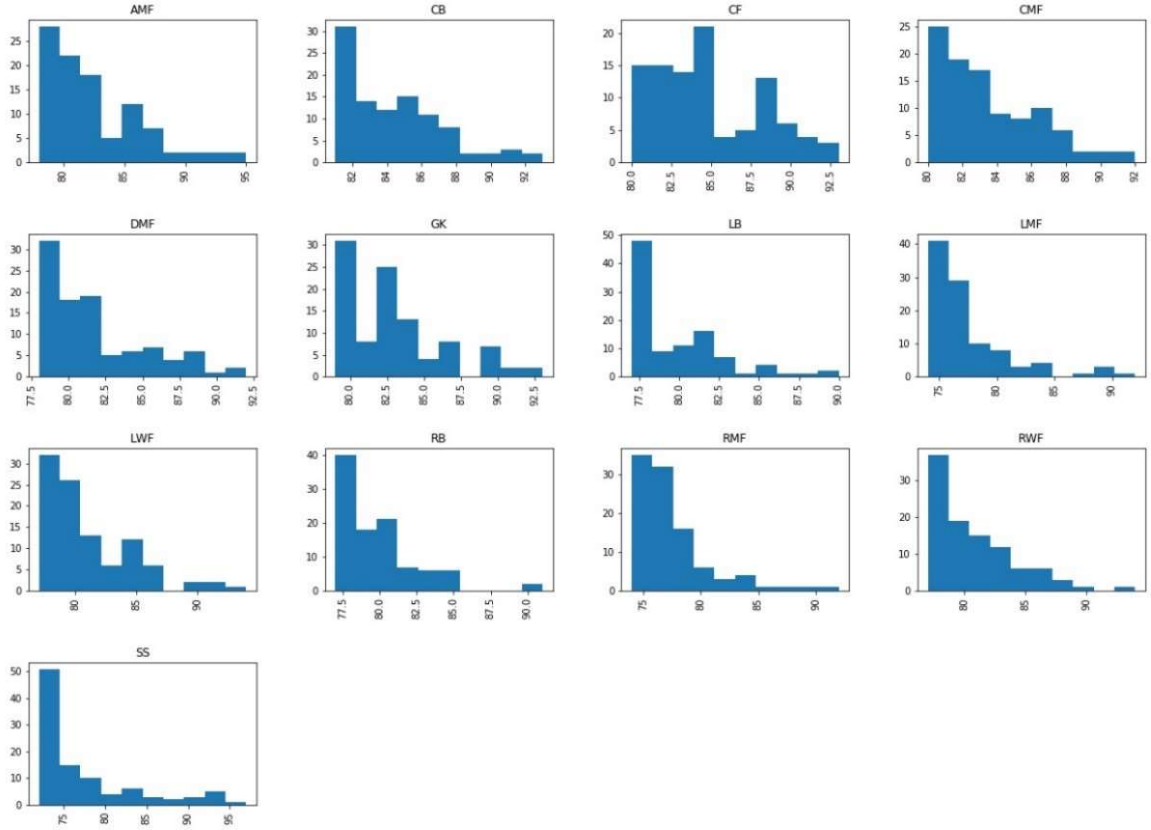
Histogram of overall scores of all players:



Histogram of height of all players:



Histogram of overall scores of players by position:



Histogram of height of player by position: