FIFA 21 Data Cleaning Challenge Documented by Trevor Machimbidza

Here is a brief documentation for each column name in the given dataset:

- photoUrl: The URL of the player's photo.
- LongName: The full name of the player.
- playerUrl: The URL of the player's page on sofifa.com.
- Nationality: The nationality of the player.
- Positions: The positions the player can play.
- Name: The short name of the player.
- Age: The age of the player.
- OVA: The overall rating of the player in FIFA 21.
- POT: The potential rating of the player in FIFA 21.
- Team & Contract: The team the player is playing for in FIFA 21, along with their contract details.
- ID: The unique identifier for the player.
- Height: The height of the player in feet and inches.
- Weight: The weight of the player in pounds.
- foot: The preferred foot of the player.
- BOV: The best overall rating the player has achieved in their career.
- BP: The best position the player has played in their career.
- Growth: The difference between the potential rating and overall rating of the player.
- Joined: The date the player joined their current team in FIFA 21.
- Loan Date End: The date the player's loan contract ends.
- Value: The market value of the player in FIFA 21.
- Wage: The weekly wage of the player in FIFA 21.
- Release Clause: The release clause value of the player in FIFA 21.
- Attacking: The attacking attributes of the player.
- Crossing: The crossing attribute of the player.
- Finishing: The finishing attribute of the player.
- Heading Accuracy: The heading accuracy attribute of the player.
- Short Passing: The short passing attribute of the player.
- Volleys: The volleys attribute of the player.
- Skill: The skill attributes of the player.
- Dribbling: The dribbling attribute of the player.
- Curve: The curve attribute of the player.
- FK Accuracy: The free kick accuracy attribute of the player.
- Long Passing: The long passing attribute of the player.
- Ball Control: The ball control attribute of the player.
- Movement: The movement attributes of the player.
- Acceleration: The acceleration attribute of the player.
- Sprint Speed: The sprint speed attribute of the player.
- Agility: The agility attribute of the player.
- Reactions: The reactions attribute of the player.
- Balance: The balance attribute of the player.
- Power: The power attributes of the player.
- Shot Power: The shot power attribute of the player.
- Jumping: The jumping attribute of the player.
- Stamina: The stamina attribute of the player.
- Strength: The strength attribute of the player.
- Long Shots: The long shots attribute of the player.
- Mentality: The mentality attributes of the player.
- Aggression: The aggression attribute of the player.
- Interceptions: The interceptions attribute of the player.
- Positioning: The positioning attribute of the player.
- Vision: The vision attribute of the player.
- Penalties: The penalties attribute of the player.
- Composure: The composure attribute of the player.
- Defending: The defending attributes of the player.
- Marking: The marking attribute of the player.
- Standing Tackle: The standing tackle attribute of the player.
- Sliding Tackle: The sliding tackle attribute of the player.
- Goalkeeping: The goalkeeping attributes of the player.
- GK Diving: The goalkeeper diving attribute of the player.
- GK Handling: The goalkeeper handling attribute of the player.
- GK Kicking: The goalkeeper kicking attribute of the player.
- GK Positioning: The goalkeeper positioning attribute of the player.
- GK Reflexes: This refers to the goalkeeper's ability to react and make saves quickly.
- Total Stats: This refers to the overall rating of the player based on their performance in all areas of the game.
- Base Stats: This refers to the player's rating in the six main areas of the game: Pace, Shooting, Passing, Dribbling, - Defending, and Physicality.

- W/F: This refers to the player's weaker foot ability.
- SM: This refers to the player's skill moves ability.
- A/W: This refers to the player's attacking work rate. It measures how frequently the player participates in attacking actions, such as making runs or positioning themselves in the opponent's half.
- D/W: This refers to the player's defensive work rate. It measures how frequently the player participates in defensive actions, such as tracking back or making tackles.
- IR: This refers to the player's injury resistance. It measures the player's ability to avoid injuries and how quickly they recover from them.
- PAC: This refers to the player's pace or speed attribute. It measures how quickly the player can move with and without the ball.
- SHO: This refers to the player's shooting ability. It measures the player's accuracy and power when shooting the ball.
- PAS: This refers to the player's passing ability. It measures the player's accuracy and range when passing the ball.
- DRI: This refers to the player's dribbling ability. It measures the player's agility, balance, and ball control when dribbling the ball.
- DEF: This refers to the player's defensive ability. It measures the player's ability to tackle, intercept, and defend against opposing players.
- PHY: This refers to the player's physicality or strength. It measures the player's ability to win physical battles and maintain possession of the ball.
- Hits: This refers to the number of times the player's profile has been viewed on the website.

In [1]:
```
#Importing the relevant libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import re
```

In [2]:
```
df = pd.read_csv(r'C:\Users\Admin\Documents\PROJECTS\ML Projects\FIFA Data Cleaning\DS-main\fifa21 raw data v2.csv')
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_11616\3767539345.py:1: DtypeWarning: Columns (76) have mixed types. Specify dtype o
ption on import or set low_memory=False.
  df = pd.read_csv(r'C:\Users\Admin\Documents\PROJECTS\ML Projects\FIFA Data Cleaning\DS-main\fifa21 raw data v2.csv')
```

In [3]:
```
df
```

Out[3]:

	ID	Name	LongName	photoUrl	playerUrl	Nationality	Age	↓OVA	POT	
0	158023	L. Messi	Lionel Messi	https://cdn.sofifa.com/players/158/023/21_60.png	http://sofifa.com/player/158023/lionel-messi/2...	Argentina	33	93	93	\n\r Ba
1	20801	Cristiano Ronaldo	C. Ronaldo dos Santos Aveiro	https://cdn.sofifa.com/players/020/801/21_60.png	http://sofifa.com/player/20801/c-ronaldo-dos-s...	Portugal	35	92	92	\n\n\n\nJu
2	200389	J. Oblak	Jan Oblak	https://cdn.sofifa.com/players/200/389/21_60.png	http://sofifa.com/player/200389/jan-oblak/210006/	Slovenia	27	91	93	\n\n\n\n
3	192985	K. De Bruyne	Kevin De Bruyne	https://cdn.sofifa.com/players/192/985/21_60.png	http://sofifa.com/player/192985/kevin-de-bruyn...	Belgium	29	91	91	\n\n\n\nMan
4	190871	Neymar Jr	Neymar da Silva Santos Jr.	https://cdn.sofifa.com/players/190/871/21_60.png	http://sofifa.com/player/190871/neymar-da-silv...	Brazil	28	91	91	\n\n\n\nParis G
...	...	...	...	...	...	...	...	...	...	
18974	247223	Xia Ao	Ao Xia	https://cdn.sofifa.com/players/247/223/21_60.png	http://sofifa.com/player/247223/ao-xia/210006/	China PR	21	47	55	\n\n\n\nWuh
18975	258760	B. Hough	Ben Hough	https://cdn.sofifa.com/players/258/760/21_60.png	http://sofifa.com/player/258760/ben-hough/210006/	England	17	47	67	\n\n\n\n
18976	252757	R. McKinley	Ronan McKinley	https://cdn.sofifa.com/players/252/757/21_60.png	http://sofifa.com/player/252757/ronan-mckinley...	England	18	47	65	\n\n\n\nDe
18977	243790	Wang Zhen'ao	Zhen'ao Wang	https://cdn.sofifa.com/players/243/790/21_60.png	http://sofifa.com/player/243790/zhenao-wang/21...	China PR	20	47	57	\n\n\n\n YiF
18978	252520	Zhou Xiao	Xiao Zhou	https://cdn.sofifa.com/players/252/520/21_60.png	http://sofifa.com/player/252520/xiao-zhou/210006/	China PR	21	47	57	\n\n\n\n YiF

18979 rows × 77 columns

Data Statistics

In [4]:
```
df.head()
```

Out[4]:

	ID	Name	LongName	photoUrl	playerUrl	Nationality	Age	↓OVA	POT	Clu
0	158023	L. Messi	Lionel Messi	https://cdn.sofifa.com/players/158/023/21_60.png	http://sofifa.com/player/158023/lionel-messi/2...	Argentina	33	93	93	\n\n\n\nF Barcelor
1	20801	Cristiano Ronaldo	C. Ronaldo dos Santos Aveiro	https://cdn.sofifa.com/players/020/801/21_60.png	http://sofifa.com/player/20801/c-ronaldo-dos-s...	Portugal	35	92	92	\n\n\n\nJuventu
2	200389	J. Oblak	Jan Oblak	https://cdn.sofifa.com/players/200/389/21_60.png	http://sofifa.com/player/200389/jan-oblak/210006/	Slovenia	27	91	93	\n\n\n\nAtlétic Madr
3	192985	K. De Bruyne	Kevin De Bruyne	https://cdn.sofifa.com/players/192/985/21_60.png	http://sofifa.com/player/192985/kevin-de-bruyn...	Belgium	29	91	91	\n\n\n\nManchest Ci
4	190871	Neymar Jr	Neymar da Silva Santos Jr.	https://cdn.sofifa.com/players/190/871/21_60.png	http://sofifa.com/player/190871/neymar-da-silv...	Brazil	28	91	91	\n\n\n\nParis Sair Germa

5 rows × 77 columns

In [5]:
```
df.tail()
```

Out[5]:

	ID	Name	LongName	photoUrl	playerUrl	Nationality	Age	↓OVA	POT	C
18974	247223	Xia Ao	Ao Xia	https://cdn.sofifa.com/players/247/223/21_60.png	http://sofifa.com/player/247223/ao-xia/210006/	China PR	21	47	55	\n\n\n\nWul
18975	258760	B. Hough	Ben Hough	https://cdn.sofifa.com/players/258/760/21_60.png	http://sofifa.com/player/258760/ben-hough/210006/	England	17	47	67	\n\n\n\nOldh Athl
18976	252757	R. McKinley	Ronan McKinley	https://cdn.sofifa.com/players/252/757/21_60.png	http://sofifa.com/player/252757/ronan-mckinley...	England	18	47	65	\n\n\n\nDe
18977	243790	Wang Zhen'ao	Zhen'ao Wang	https://cdn.sofifa.com/players/243/790/21_60.png	http://sofifa.com/player/243790/zhenao-wang/21...	China PR	20	47	57	\n\n\n\nDal YiFang
18978	252520	Zhou Xiao	Xiao Zhou	https://cdn.sofifa.com/players/252/520/21_60.png	http://sofifa.com/player/252520/xiao-zhou/210006/	China PR	21	47	57	\n\n\n\nDal YiFang

5 rows × 77 columns

In [6]:
```
df.shape
```

Out[6]: (18979, 77)

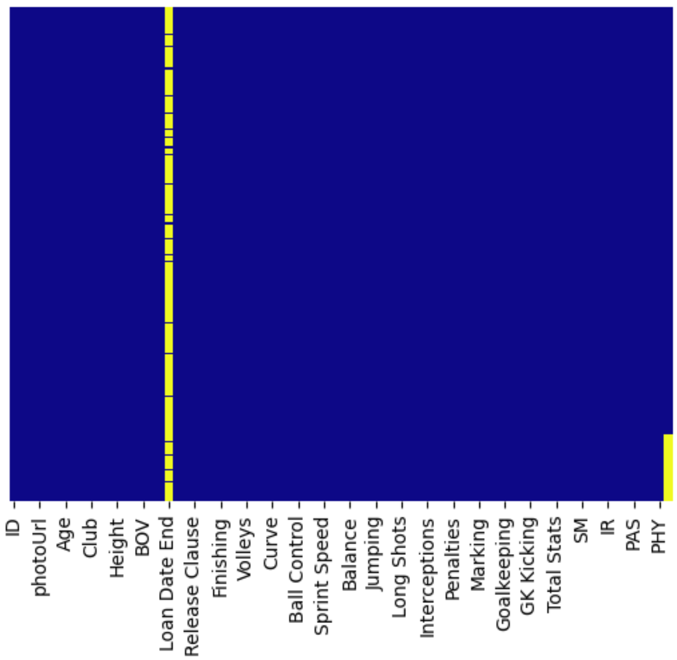
In [*]:
```
df.info()
```

In [*]:
```
df.describe()
```

Handling Null Values

In [7]:
```
sns.heatmap(df.isnull(), yticklabels = False, cbar = False, cmap ='plasma')
```

Out[7]: <Axes: >



In [8]:
```
#isna is used to detect missing values
df.columns[df.isna().any()].tolist()
```

Out[8]: ['Loan Date End', 'Hits']

In [9]:
```
df.loc[df['Loan Date End'].notnull()].head()
```

Out[9]:

	ID	Name	LongName	photoUrl	playerUrl	Nationality	Age	↓OVA	POT	C
205	173731	G. Bale	Gareth Bale	https://cdn.sofifa.com/players/173/731/21_60.png	http://sofifa.com/player/173731/gareth-bale/21...	Wales	30	83	83	\n\n\n\nTotten Hot
248	193105	A. Areola	Alphonse Areola	https://cdn.sofifa.com/players/193/105/21_60.png	http://sofifa.com/player/193105/alphonse-areol...	France	27	82	86	\n\n\n\nFul
254	200888	Danilo Pereira	Danilo Luís Hélio Pereira	https://cdn.sofifa.com/players/200/888/21_60.png	http://sofifa.com/player/200888/danilo-luis-he...	Portugal	28	82	82	\n\n\n\nF Saint-Ger
302	216409	M. Politano	Matteo Politano	https://cdn.sofifa.com/players/216/409/21_60.png	http://sofifa.com/player/216409/matteo-politan...	Italy	26	81	81	\n\n\n\nN
306	223959	L. Torreira	Lucas Torreira	https://cdn.sofifa.com/players/223/959/21_60.png	http://sofifa.com/player/223959/lucas-torreira...	Uruguay	24	81	85	\n\n\n\nAtl Ma

5 rows × 77 columns

In [*]:
```
#Replacing all Nan values with No
#Every players on loan is a No

df["Loan Date End"]=df["Loan Date End"].apply(lambda x: "Yes" if len(x)>3 else "No",)
```

In [10]:
```
df['Loan Date End'].unique()
```

Out[10]:
```
array([nan, 'Jun 30, 2021', 'Dec 31, 2020', 'Jan 30, 2021',
       'Jun 30, 2022', 'May 31, 2021', 'Jul 5, 2021', 'Dec 31, 2021',
       'Jul 1, 2021', 'Jan 1, 2021', 'Aug 31, 2021', 'Jan 31, 2021',
       'Dec 30, 2021', 'Jun 23, 2021', 'Jan 3, 2021', 'Nov 27, 2021',
       'Jan 17, 2021', 'Jun 30, 2023', 'Jul 31, 2021', 'Nov 22, 2020',
       'May 31, 2022', 'Dec 30, 2020', 'Jan 4, 2021', 'Nov 30, 2020',
       'Aug 1, 2021'], dtype=object)
```

In [11]:
```
# Change column name to 'loan'
df.rename(columns={"Loan Date End": "Loan"}, inplace=True)
```

Handling Missing values with hits columns

In [12]:
```
df['Hits'].unique()
```

Out[12]:
```
array(['771', '562', '150', '207', '595', '248', '246', '120', '1.6K',
       '130', '321', '189', '175', '96', '118', '216', '212', '154',
       '205', '202', '339', '408', '103', '332', '86', '173', '161',
       '396', '1.1K', '433', '242', '206', '177', '1.5K', '198', '459',
       '117', '119', '209', '84', '187', '165', '203', '65', '336', '126',
       '313', '124', '145', '538', '182', '101', '45', '377', '99', '194',
       '403', '414', '593', '374', '245', '3.2K', '266', '299', '309',
       '215', '265', '211', '112', '337', '70', '159', '688', '116', '63',
       '144', '123', '71', '224', '113', '168', '61', '89', '137', '278',
       '75', '148', '176', '197', '264', '214', '247', '402', '440',
       '1.7K', '2.3K', '171', '320', '657', '87', '259', '200', '255',
       '253', '196', '60', '97', '85', '169', '256', '132', '239', '166',
       '121', '109', '32', '46', '122', '48', '527', '199', '282', '51',
       '1.9K', '642', '155', '323', '288', '497', '509', '79', '49',
       '270', '511', '80', '128', '115', '156', '204', '143', '140',
       '152', '220', '134', '225', '94', '74', '135', '142', '50', '77',
       '40', '107', '193', '179', '34', '64', '453', '57', '81', '28',
       '78', '133', '43', '425', '88', '42', '36', '233', '376', '210',
       '444', '100', '263', '98', '29', '160', '39', '257', '6', '310',
       '138', '62', '293', '285', '362', '66', '69', '58', '21', '20',
       '131', '38', '406', '68', '108', '110', '93', '512', '443', '306',
       '352', '422', '585', '346', '178', '841', '76', '394', '72', '172',
       '44', '407', '230', '367', '295', '157', '243', '56', '111', '326',
       '679', '18', '92', '59', '25', '184', '53', '12', '90', '55', '73',
       '11', '566', '180', '83', '262', '17', '26', '31', '280', '359',
       '213', '297', '387', '480', '381', '677', '486', '8', '244', '129',
       '388', '275', '319', '2K', '52', '91', '421', '153', '27', '41',
       '222', '35', '102', '23', '30', '33', '146', '13', '19', '14',
       '106', '276', '568', '353', '47', '478', '249', '254', '369',
       '219', '565', '237', '227', '434', '375', '162', '605', '654', '3',
       '7', '9', '104', '114', '186', '446', '756', '22', '139', '500',
       '67', '147', '149', '16', '82', '54', '37', '15', '1.3K', '3K',
       '952', '5', '749', '541', '330', '393', '517', '770', '409', '170',
       '125', '283', '342', '363', '580', '105', '217', '24', '141', '10',
       '427', '158', '426', '4', '666', '181', '324', '979', '1.4K',
       '302', '751', '298', '411', '944', '2', '947', '292', '349', '621',
       '1', '2.8K', '338', '287', '261', '218', '1.8K', '240', '279',
       '229', '188', '315', '664', '613', '190', '706', '127', '462',
       '386', '695', '491', '167', '281', '250', '307', '95', '231',
       '174', '680', '633', '221', '348', '602', '183', '653', '195',
       '164', '151', '258', '8.4K', '343', '419', '655', '136', '399',
       '531', '357', '228', '385', '312', '340', '238', '487', '355',
       '499', '4.3K', '296', '515', '943', '1.2K', '903', '335', '191',
       '594', '267', '617', '516', '504', '331', '652', '410', '550',
       '473', '442', '344', '208', '1K', '2.5K', '273', '485', '826',
       '192', '405', '941', '477', '644', '303', '417', '6K', nan, 11.0,
       2.0, 1.0, 31.0, 3.0, 10.0, 9.0, 17.0, 7.0, 4.0, 6.0], dtype=object)
```

In [13]:
```
df['Hits'] = df['Hits'].fillna(0)
df['Hits'].isnull().sum()
```

Out[13]:
```
0
```

In [15]:
```
def hit_func(val):
    if "k" in str(val):
        val=val.replace("k","")
        return int(float(val)*1000)
    else:
        return int(val)
```

In [ ]:
```
df['Hits'] = df["Hits"].apply(hit_func)
```

In [16]:
```
df['Hits'].unique()
#You observe how values are numeric and alpha-numeric
```

Out[16]:
```
array(['771', '562', '150', '207', '595', '248', '246', '120', '1.6K',
       '130', '321', '189', '175', '96', '118', '216', '212', '154',
       '205', '202', '339', '408', '103', '332', '86', '173', '161',
       '396', '1.1K', '433', '242', '206', '177', '1.5K', '198', '459',
       '117', '119', '209', '84', '187', '165', '203', '65', '336', '126',
       '313', '124', '145', '538', '182', '101', '45', '377', '99', '194',
       '403', '414', '593', '374', '245', '3.2K', '266', '299', '309',
       '215', '265', '211', '112', '337', '70', '159', '688', '116', '63',
       '144', '123', '71', '224', '113', '168', '61', '89', '137', '278',
       '75', '148', '176', '197', '264', '214', '247', '402', '440',
       '1.7K', '2.3K', '171', '320', '657', '87', '259', '200', '255',
       '253', '196', '60', '97', '85', '169', '256', '132', '239', '166',
       '121', '109', '32', '46', '122', '48', '527', '199', '282', '51',
       '1.9K', '642', '155', '323', '288', '497', '509', '79', '49',
       '270', '511', '80', '128', '115', '156', '204', '143', '140',
       '152', '220', '134', '225', '94', '74', '135', '142', '50', '77',
       '40', '107', '193', '179', '34', '64', '453', '57', '81', '28',
       '78', '133', '43', '425', '88', '42', '36', '233', '376', '210',
       '444', '100', '263', '98', '29', '160', '39', '257', '6', '310',
       '138', '62', '293', '285', '362', '66', '69', '58', '21', '20',
       '131', '38', '406', '68', '108', '110', '93', '512', '443', '306',
       '352', '422', '585', '346', '178', '841', '76', '394', '72', '172',
       '44', '407', '230', '367', '295', '157', '243', '56', '111', '326',
       '679', '18', '92', '59', '25', '184', '53', '12', '90', '55', '73',
       '11', '566', '180', '83', '262', '17', '26', '31', '280', '359',
       '213', '297', '387', '480', '381', '677', '486', '8', '244', '129',
       '388', '275', '319', '2K', '52', '91', '421', '153', '27', '41',
       '222', '35', '102', '23', '30', '33', '146', '13', '19', '14',
       '106', '276', '568', '353', '47', '478', '249', '254', '369',
       '219', '565', '237', '227', '434', '375', '162', '605', '654', '3',
       '7', '9', '104', '114', '186', '446', '756', '22', '139', '500',
       '67', '147', '149', '16', '82', '54', '37', '15', '1.3K', '3K',
       '952', '5', '749', '541', '330', '393', '517', '770', '409', '170',
       '125', '283', '342', '363', '580', '105', '217', '24', '141', '10',
       '427', '158', '426', '4', '666', '181', '324', '979', '1.4K',
       '302', '751', '298', '411', '944', '2', '947', '292', '349', '621',
       '1', '2.8K', '338', '287', '261', '218', '1.8K', '240', '279',
       '229', '188', '315', '664', '613', '190', '706', '127', '462',
       '386', '695', '491', '167', '281', '250', '307', '95', '231',
       '174', '680', '633', '221', '348', '602', '183', '653', '195',
       '164', '151', '258', '8.4K', '343', '419', '655', '136', '399',
       '531', '357', '228', '385', '312', '340', '238', '487', '355',
       '499', '4.3K', '296', '515', '943', '1.2K', '903', '335', '191',
       '594', '267', '617', '516', '504', '331', '652', '410', '550',
       '473', '442', '344', '208', '1K', '2.5K', '273', '485', '826',
       '192', '405', '941', '477', '644', '303', '417', '6K', 0, 11.0,
       2.0, 1.0, 31.0, 3.0, 10.0, 9.0, 17.0, 7.0, 4.0, 6.0], dtype=object)
```

In [ ]:
```
#  We equate the 0 values with mean values of the column since it is Nan initially
df[df["Hits"]==0]=int(df["Hits"].mean())
```

Cleaning Club Column

In [18]:
```
df['Club']=df["Club"].replace(["\n\n\n\n"],'', regex=True)
```

Checking for Duplicates

In [19]:
```
duplicates = df.duplicated()
```

In [20]:
```
#Dropping duplicate columns
df.drop_duplicates(inplace=True)
```

In [21]:
```
duplicates = df.duplicated()
print(df[duplicates])
```

```
Empty DataFrame
Columns: [ID, Name, LongName, photoUrl, playerUrl, Nationality, Age, ↓OVA, POT, Club, Contract, Positions, Height, Weight, Pref
erred Foot, BOV, Best Position, Joined, Loan, Value, Wage, Release Clause, Attacking, Crossing, Finishing, Heading Accuracy, Sh
ort Passing, Volleys, Skill, Dribbling, Curve, FK Accuracy, Long Passing, Ball Control, Movement, Acceleration, Sprint Speed, A
gility, Reactions, Balance, Power, Shot Power, Jumping, Stamina, Strength, Long Shots, Mentality, Aggression, Interceptions, Po
sitioning, Vision, Penalties, Composure, Defending, Marking, Standing Tackle, Sliding Tackle, Goalkeeping, GK Diving, GK Handli
ng, GK Kicking, GK Positioning, GK Reflexes, Total Stats, Base Stats, W/F, SM, A/W, D/W, IR, PAC, SHO, PAS, DRI, DEF, PHY, Hit
s]
Index: []

[0 rows x 77 columns]
```

Cleaning Height column

In [22]:
```
df['Height'].unique()
```

Out[22]:
```
array(['170cm', '187cm', '188cm', '181cm', '175cm', '184cm', '191cm',
       '178cm', '193cm', '185cm', '199cm', '173cm', '168cm', '176cm',
       '177cm', '183cm', '180cm', '189cm', '179cm', '195cm', '172cm',
       '182cm', '186cm', '192cm', '165cm', '194cm', '167cm', '196cm',
       '163cm', '190cm', '174cm', '169cm', '171cm', '197cm', '200cm',
       '166cm', '6\'2"', '164cm', '198cm', '6\'3"', '6\'5"', '5\'11"',
       '6\'4"', '6\'1"', '6\'0"', '5\'10"', '5\'9"', '5\'6"', '5\'7"',
       '5\'4"', '201cm', '158cm', '162cm', '161cm', '160cm', '203cm',
       '157cm', '156cm', '202cm', '159cm', '206cm', '155cm'], dtype=object)
```

In [23]:
```
#Function will clean Column Height
def convert_height(val):
    if str(val).endswith("cm"):

        # Function will remove cm for series values
        s = [int(s) for s in re.findall(r'-?\d+\.?\d*', val)]

        return int(s[0])

    elif str(val).endswith("\""):

        # Since 1 foot=30.48cm and 1 inch= 2.54cm, we multiply first number instance by 30.48 and second instance by 2.54
        # We add up and we have series values in cm

        s = [int(s) for s in re.findall(r'-?\d+\.?\d*', val)]
        answer_cm= int(float((s[0]*30.48)+(s[1]*2.54)))
        return answer_cm
```

In [24]:
```
df['Height']=df['Height'].apply(convert_height)
```

In [25]:
```
df['Height'].unique()
```

Out[25]:
```
array([170, 187, 188, 181, 175, 184, 191, 178, 193, 185, 199, 173, 168,
       176, 177, 183, 180, 189, 179, 195, 172, 182, 186, 192, 165, 194,
       167, 196, 163, 190, 174, 169, 171, 197, 200, 166, 164, 198, 162,
       201, 158, 161, 160, 203, 157, 156, 202, 159, 206, 155], dtype=int64)
```

In [26]:
```
df['Height'].isna().sum()
```

Out[26]:
```
0
```

In [27]:
```
df['Weight'].unique()
```

Out[27]:
```
array(['72kg', '83kg', '87kg', '70kg', '68kg', '80kg', '71kg', '91kg',
       '73kg', '85kg', '92kg', '69kg', '84kg', '96kg', '81kg', '82kg',
       '75kg', '86kg', '89kg', '74kg', '76kg', '64kg', '78kg', '90kg',
       '66kg', '60kg', '94kg', '79kg', '67kg', '65kg', '59kg', '61kg',
       '93kg', '88kg', '97kg', '77kg', '62kg', '63kg', '95kg', '100kg',
       '58kg', '183lbs', '179lbs', '172lbs', '196lbs', '176lbs', '185lbs',
       '170lbs', '203lbs', '168lbs', '161lbs', '146lbs', '130lbs',
       '190lbs', '174lbs', '148lbs', '165lbs', '159lbs', '192lbs',
       '181lbs', '139lbs', '154lbs', '157lbs', '163lbs', '98kg', '103kg',
       '99kg', '102kg', '56kg', '101kg', '57kg', '55kg', '104kg', '107kg',
       '110kg', '53kg', '50kg', '54kg', '52kg'], dtype=object)
```

In [28]:
```
#We have two different weight SI units
# Function will clean Column Weight
def convert_weight(val):
    if val.endswith("kg"):
        # Function will remove cm for series values
        s = [int(s) for s in re.findall(r'-?\d+\.?\d*', val)]
        return int(s[0])

    elif val.endswith("lbs"):
        # We add up and we have series values in kg
        s = [int(s) for s in re.findall(r'-?\d+\.?\d*', val)]
        answer_cm= s[0]*0.453592
        return int(answer_cm)
```

In [29]:
```
df['Weight']=df['Weight'].apply(convert_weight)
```

In [30]:
```
df['Weight'].unique()
#We have two different weight SI units; kg and lbs
```

Out[30]:
```
array([ 72,  83,  87,  70,  68,  80,  71,  91,  73,  85,  92,  69,  84,
        96,  81,  82,  75,  86,  89,  74,  76,  64,  78,  90,  66,  60,
        94,  79,  67,  65,  59,  61,  93,  88,  97,  77,  62,  63,  95,
       100,  58,  98, 103,  99, 102,  56, 101,  57,  55, 104, 107, 110,
        53,  50,  54,  52], dtype=int64)
```

In [*]:
```
df['Weight'].isna().sum()
```

Cleaning Value Column

In [31]:
```
df['Value'].unique()
```

Out[31]:
```
array(['€103.5M', '€63M', '€120M', '€129M', '€132M', '€111M', '€120.5M',
       '€102M', '€185.5M', '€110M', '€113M', '€90.5M', '€82M', '€17.5M',
       '€83.5M', '€33.5M', '€114.5M', '€78M', '€103M', '€109M', '€92M',
       '€10M', '€76.5M', '€89.5M', '€87.5M', '€79.5M', '€124M', '€114M',
       '€95M', '€92.5M', '€105.5M', '€88.5M', '€85M', '€81.5M', '€26M',
       '€21M', '€56M', '€67.5M', '€53M', '€36.5M', '€51M', '€65.5M',
       '€46.5M', '€61.5M', '€72.5M', '€77.5M', '€43.5M', '€32.5M', '€36M',
       '€32M', '€54M', '€49.5M', '€57M', '€66.5M', '€74.5M', '€71.5M',
       '€121M', '€99M', '€67M', '€86.5M', '€93.5M', '€70M', '€62M',
       '€66M', '€58M', '€44M', '€81M', '€37M', '€14.5M', '€46M', '€47.5M',
       '€52.5M', '€54.5M', '€34.5M', '€57.5M', '€51.5M', '€44.5M', '€55M',
       '€48M', '€60.5M', '€63.5M', '€61M', '€29M', '€58.5M', '€55.5M',
       '€42M', '€40.5M', '€43M', '€45.5M', '€34M', '€26.5M', '€42.5M',
       '€35.5M', '€45M', '€41.5M', '€40M', '€11M', '€13.5M', '€29.5M',
       '€27M', '€15.5M', '€38.5M', '€52M', '€33M', '€19M', '€73.5M',
       '€38M', '€35M', '€47M', '€24M', '€30.5M', '€18M', '€28M', '€25.5M',
       '€25M', '€31M', '€23.5M', '€30M', '€31.5M', '€22.5M', '€28.5M',
       '€4M', '€12.5M', '€37.5M', '€27.5M', '€16M', '€15M', '€20.5M',
       '€22M', '€3.4M', '€5M', '€56.5M', '€62.5M', '€0', '€39M', '€24.5M',
       '€21.5M', '€13M', '€8M', '€20M', '€8.5M', '€2.9M', '€9M', '€4.6M',
       '€50M', '€23M', '€18.5M', '€7M', '€19.5M', '€5.5M', '€7.5M',
       '€3.8M', '€14M', '€10.5M', '€16.5M', '€3.6M', '€9.5M', '€39.5M',
       '€17M', '€12M', '€11.5M', '€4.9M', '€3M', '€1.9M', '€6.5M',
       '€1.7M', '€2.4M', '€3.1M', '€6M', '€3.7M', '€4.7M', '€4.3M',
       '€2.1M', '€1.2M', '€1.8M', '€4.8M', '€3.2M', '€1.3M', '€825K',
       '€2.3M', '€1.5M', '€3.9M', '€2.6M', '€3.5M', '€2.8M', '€2.7M',
       '€4.4M', '€4.1M', '€950K', '€1.6M', '€625K', '€1.1M', '€4.5M',
       '€4.2M', '€2.2M', '€3.3M', '€1.4M', '€2M', '€475K', '€925K',
       '€750K', '€725K', '€2.5M', '€1M', '€350K', '€525K', '€600K',
       '€850K', '€800K', '€550K', '€250K', '€400K', '€425K', '€575K',
       '€210K', '€325K', '€900K', '€875K', '€650K', '€700K', '€500K',
       '€975K', '€375K', '€775K', '€275K', '€180K', '€450K', '€675K',
       '€150K', '€240K', '€300K', '€130K', '€220K', '€200K', '€110K',
       '€170K', '€230K', '€90K', '€120K', '€80K', '€190K', '€140K',
       '€160K', '€100K', '€60K', '€50K', '€70K', '€45K', '€35K', '€40K',
       '€25K', '€20K', '€15K', '€30K', '€9K'], dtype=object)
```

In [32]:
```
df['Value'] = df['Value'].str.replace('€','')
```

In [33]:
```
def convert_value(val):

    if 'M' in val:
        # We split val by space and we get a list in return
        x=float(val.split('M')[0])
        # After extracting number fron val, we multiply by 1000000
        return int(x*1000000)

    elif 'K' in val:
        x=float(val.split('K')[0])
        # After extracting number fron val, we multiply by 1000000
        return int(x*1000)
    else:
        return int(val)
```

In [34]:
```
df['Value']=df['Value'].apply(convert_value)
```

In [36]:
```
df['Value'].unique()
```

Out[36]:
```
array([103500000,  63000000, 120000000, 129000000, 132000000, 111000000,
       120500000, 102000000, 185500000, 110000000, 113000000,  90500000,
        82000000,  17500000,  83500000,  33500000, 114500000,  78000000,
       103000000, 109000000,  92000000,  10000000,  76500000,  89500000,
        87500000,  79500000, 124000000, 114000000,  95000000,  92500000,
       105500000,  88500000,  85000000,  81500000,  26000000,  21000000,
        56000000,  67500000,  53000000,  36500000,  51000000,  65500000,
        46500000,  61500000,  72500000,  77500000,  43500000,  32500000,
        36000000,  32000000,  54000000,  49500000,  57000000,  66500000,
        74500000,  71500000, 121000000,  99000000,  67000000,  86500000,
        93500000,  70000000,  62000000,  66000000,  58000000,  44000000,
        81000000,  37000000,  14500000,  46000000,  47500000,  52500000,
        54500000,  34500000,  57500000,  51500000,  44500000,  55000000,
        48000000,  60500000,  63500000,  61000000,  29000000,  58500000,
        55500000,  42000000,  40500000,  43000000,  45500000,  34000000,
        26500000,  42500000,  35500000,  45000000,  41500000,  40000000,
        11000000,  13500000,  29500000,  27000000,  15500000,  38500000,
        52000000,  33000000,  19000000,  73500000,  38000000,  35000000,
        47000000,  24000000,  30500000,  18000000,  28000000,  25500000,
        25000000,  31000000,  23500000,  30000000,  31500000,  22500000,
        28500000,   4000000,  12500000,  37500000,  27500000,  16000000,
        15000000,  20500000,  22000000,   3400000,   5000000,  56500000,
        62500000,         0,  39000000,  24500000,  21500000,  13000000,
         8000000,  20000000,   8500000,   2900000,   9000000,   4600000,
        50000000,  23000000,  18500000,   7000000,  19500000,   5500000,
         7500000,   3800000,  14000000,  10500000,  16500000,   3600000,
         9500000,  39500000,  17000000,  12000000,  11500000,   4900000,
         3000000,   1900000,   6500000,   1700000,   2400000,   3100000,
         6000000,   3700000,   4700000,   4300000,   2100000,   1200000,
         1800000,   4800000,   3200000,   1300000,    825000,   2300000,
         1500000,   3900000,   2600000,   3500000,   2800000,   2700000,
         4400000,   4099999,    950000,   1600000,    625000,   1100000,
         4500000,   4200000,   2200000,   3300000,   1400000,   2000000,
          475000,    925000,    750000,    725000,   2500000,   1000000,
          350000,    525000,    600000,    850000,    800000,    550000,
          250000,    400000,    425000,    575000,    210000,    325000,
          900000,    875000,    650000,    700000,    500000,    975000,
          375000,    775000,    275000,    180000,    450000,    675000,
          150000,    240000,    300000,    130000,    220000,    200000,
          110000,    170000,    230000,     90000,    120000,     80000,
          190000,    140000,    160000,    100000,     60000,     50000,
           70000,     45000,     35000,     40000,     25000,     20000,
           15000,     30000,      9000], dtype=int64)
```

In [35]:
```
df['Value'].isna().sum()
```

Out[35]:
```
0
```

Cleaning wage column

In [37]:
```
df['Wage'].unique()
#If you observe closely, you will observe some players earn in thousands of euros and hundreds of eurosWe slice with caution here
```

Out[37]:
```
array(['€560K', '€220K', '€125K', '€370K', '€270K', '€240K', '€250K',
       '€160K', '€260K', '€210K', '€310K', '€130K', '€350K', '€300K',
       '€190K', '€145K', '€195K', '€100K', '€140K', '€290K', '€82K',
       '€110K', '€230K', '€155K', '€200K', '€165K', '€95K', '€170K',
       '€105K', '€115K', '€150K', '€135K', '€55K', '€58K', '€81K', '€34K',
       '€120K', '€59K', '€90K', '€65K', '€56K', '€71K', '€18K', '€75K',
       '€47K', '€20K', '€84K', '€86K', '€74K', '€78K', '€27K', '€68K',
       '€85K', '€25K', '€46K', '€83K', '€54K', '€79K', '€175K', '€43K',
       '€49K', '€45K', '€38K', '€41K', '€39K', '€23K', '€51K', '€50K',
       '€87K', '€30K', '€14K', '€69K', '€31K', '€64K', '€53K', '€35K',
       '€21K', '€28K', '€17K', '€33K', '€70K', '€32K', '€89K', '€26K',
       '€40K', '€76K', '€72K', '€48K', '€36K', '€29K', '€60K', '€16K',
       '€37K', '€24K', '€52K', '€0', '€62K', '€73K', '€63K', '€19K',
       '€1K', '€66K', '€80K', '€12K', '€2K', '€42K', '€13K', '€900',
       '€57K', '€77K', '€61K', '€22K', '€67K', '€44K', '€15K', '€11K',
       '€8K', '€850', '€10K', '€88K', '€500', '€7K', '€6K', '€9K', '€5K',
       '€700', '€950', '€750', '€3K', '€650', '€600', '€4K', '€800',
       '€550'], dtype=object)
```

In [38]:
```
df['Wage'] = df['Wage'].str.replace('€','')
```

In [39]:
```
# Function will clean Column Wage
def convert_wage(val):

    if "K" in val:
        value=val.split("K")[0]
        value= int(value)*1000
        return value
    else:
        value= int(val)

        return val
```

In [41]:
```
df['Wage']=df['Wage'].apply(convert_wage)
```

In [42]:
```
df['Wage'].isna().sum()
```

Out[42]:
```
0
```

Cleaning the column Release clause

In [43]:
```
df['Release Clause'] = df['Release Clause'].str.replace('€','')
```

In [44]:
```
def run(val):
    if 'M' in val:
        # We split val by space and we get a list in return
        x=float(val.split('M')[0])
        return int(x*1000000)

    elif 'K' in val:
       # We split val by space and we get a list in return
        x=float(val.split('K')[0])
        return int(x*1000)
    else:
        return int(val)
```

In [45]:
```
df["Release Clause"]=df['Release Clause'].apply(run)
```

In [46]:
```
df["Release Clause"].unique()
```

Out[46]:
```
array([138400000,  75900000, 159400000, ...,     59000,     35000,
           64000], dtype=int64)
```

In [47]:
```
df["Release Clause"].isna().sum()
```

Out[47]:
```
0
```

Handling column Contract

In [48]:
```
df['Contract'].unique()
```

Out[48]:
```
array(['2004 ~ 2021', '2018 ~ 2022', '2014 ~ 2023', '2015 ~ 2023',
       '2017 ~ 2022', '2017 ~ 2023', '2018 ~ 2024', '2014 ~ 2022',
       '2018 ~ 2023', '2016 ~ 2023', '2013 ~ 2023', '2011 ~ 2023',
       '2009 ~ 2022', '2005 ~ 2021', '2011 ~ 2021', '2015 ~ 2022',
       '2017 ~ 2024', '2010 ~ 2024', '2012 ~ 2021', '2019 ~ 2024',
       '2015 ~ 2024', '2017 ~ 2025', '2020 ~ 2025', '2019 ~ 2023',
       '2008 ~ 2023', '2015 ~ 2021', '2020 ~ 2022', '2012 ~ 2022',
       '2016 ~ 2025', '2013 ~ 2022', '2011 ~ 2022', '2012 ~ 2024',
       '2016 ~ 2021', '2012 ~ 2023', '2008 ~ 2022', '2019 ~ 2022',
       '2017 ~ 2021', '2013 ~ 2024', '2020 ~ 2024', '2010 ~ 2022',
       '2020 ~ 2021', '2011 ~ 2024', '2020 ~ 2023', '2014 ~ 2024',
       '2013 ~ 2026', '2016 ~ 2022', '2010 ~ 2021', '2013 ~ 2021',
       '2019 ~ 2025', '2018 ~ 2025', '2016 ~ 2024', '2018 ~ 2021',
       '2009 ~ 2024', '2007 ~ 2022', 'Jun 30, 2021 On Loan',
       '2009 ~ 2021', '2019 ~ 2021', '2019 ~ 2026', 'Free', '2012 ~ 2028',
       '2010 ~ 2023', '2014 ~ 2021', '2015 ~ 2025', '2014 ~ 2026',
       '2012 ~ 2025', '2017 ~ 2020', '2002 ~ 2022', '2020 ~ 2027',
       '2013 ~ 2025', 'Dec 31, 2020 On Loan', '2019 ~ 2020',
       '2011 ~ 2025', '2016 ~ 2020', '2007 ~ 2021', '2020 ~ 2026',
       '2010 ~ 2025', '2009 ~ 2023', '2008 ~ 2021', '2020 ~ 2020',
       '2016 ~ 2026', 'Jan 30, 2021 On Loan', '2012 ~ 2020',
       '2014 ~ 2025', 'Jun 30, 2022 On Loan', '2015 ~ 2020',
       'May 31, 2021 On Loan', '2018 ~ 2020', '2014 ~ 2020',
       '2013 ~ 2020', '2006 ~ 2024', 'Jul 5, 2021 On Loan',
       'Dec 31, 2021 On Loan', '2004 ~ 2025', '2011 ~ 2020',
       'Jul 1, 2021 On Loan', 'Jan 1, 2021 On Loan', '2006 ~ 2023',
       'Aug 31, 2021 On Loan', '2006 ~ 2021', '2005 ~ 2023',
       '2003 ~ 2020', '2009 ~ 2020', '2002 ~ 2020', '2005 ~ 2020',
       '2005 ~ 2022', 'Jan 31, 2021 On Loan', '2010 ~ 2020',
       'Dec 30, 2021 On Loan', '2008 ~ 2020', '2007 ~ 2020',
       '2003 ~ 2021', 'Jun 23, 2021 On Loan', 'Jan 3, 2021 On Loan',
       'Nov 27, 2021 On Loan', '2002 ~ 2021', 'Jan 17, 2021 On Loan',
       'Jun 30, 2023 On Loan', '1998 ~ 2021', '2003 ~ 2022',
       '2007 ~ 2023', 'Jul 31, 2021 On Loan', 'Nov 22, 2020 On Loan',
       'May 31, 2022 On Loan', '2006 ~ 2020', 'Dec 30, 2020 On Loan',
       '2007 ~ 2025', 'Jan 4, 2021 On Loan', 'Nov 30, 2020 On Loan',
       '2004 ~ 2020', '2009 ~ 2025', 'Aug 1, 2021 On Loan'], dtype=object)
```

In [49]:
```
# Function create new column "Contract_end" for every player
def convert_contract(val):
    # We split values into list by space
    val= val.split(" ")
    if 'Free' in val:
        return val[0]
    elif "Loan" in val:
        return val[2]
    elif "~" in val:
        return val[2]
```

Creating New Column "Contract_end"

-Contract players have contract with parent club. -Loan players have contract with parent team despite temporary borrowing. -Free Players hae no club, no contract.

In [50]:
```
df['Contract_end']=df["Contract"].apply(convert_contract)
```

In [51]:
```
df['Contract_end'].unique()
```

Out[51]:
```
array(['2021', '2022', '2023', '2024', '2025', '2026', 'Free', '2028',
       '2020', '2027'], dtype=object)
```

Creating new column "Contract_start"

In [52]:
```
df["Joined"].unique()
```

Out[52]:
```
array(['Jul 1, 2004', 'Jul 10, 2018', 'Jul 16, 2014', ..., 'Sep 22, 2018',
       'Feb 28, 2015', 'Mar 6, 2018'], dtype=object)
```

In [53]:
```
# Function create new column "Contract_start" for every player
def convert_contract(val):
    val= val.split(" ")
    return val[-1]
```

In [54]:
```
df["Contract_start"]= df["Joined"].apply(convert_contract)
```

In [55]:
```
df["Contract_start"].unique()
```

Out[55]:
```
array(['2004', '2018', '2014', '2015', '2017', '2016', '2013', '2011',
       '2009', '2005', '2010', '2012', '2019', '2020', '2008', '2007',
       '2002', '2006', '2003', '1998'], dtype=object)
```

Creating Column Player Status

In [56]:
```
# Function create new column "Player Status" for every player
def status(val):
    # We split values in list by space
    val= val.split(" ")
    if 'Free' in val:
        # We assign free to free players with no contract
        return "Free"
    elif "Loan" in val:
        # We assign Loan to Loan players with temporary club
        return "Loan"
    elif "~" in val:
        # We assign Contract to  players with  contract with parent club
        return "Contract"
```

In [57]:
```
df["Player Status"]=df["Contract"].apply(status)
```

In [58]:
```
df["Player Status"].unique()
```

Out[58]:
```
array(['Contract', 'Loan', 'Free'], dtype=object)
```

In [59]:
```
df["Player Status"].isna().sum()
```

Out[59]:
```
0
```

We have no null values, we are good here

#Removing star symbol from column W/F,SM and IP

In [62]:
```
df.loc[:,["SM","IR",'W/F']]
```

Out[62]:

	SM	IR	W/F
0	4★	5 ★	4 ★
1	5★	5 ★	4 ★
2	1★	3 ★	3 ★
3	4★	4 ★	5 ★
4	5★	5 ★	5 ★
...	...	...	...
18974	2★	1 ★	2 ★
18975	2★	1 ★	2 ★
18976	2★	1 ★	2 ★
18977	2★	1 ★	3 ★
18978	2★	1 ★	3 ★

18979 rows × 3 columns

In [63]:
```
def star_remove(val):
    s = [int(s) for s in re.findall(r'[0-9]+', val)]
    return s[0]
```

In [64]:
```
df['W/F']=df['W/F'].apply(star_remove)
```

In [65]:
```
df['IR']=df['IR'].apply(star_remove)
```

In [66]:
```
df['SM']=df['SM'].apply(star_remove)
```

In [67]:
```
df.head()
```

Out[67]:

	ID	Name	LongName	photoUrl	playerUrl	Nationality	Age	↓OVA	POT	Club	...
0	158023	L. Messi	Lionel Messi	https://cdn.sofifa.com/players/158/023/21_60.png	http://sofifa.com/player/158023/lionel-messi/2...	Argentina	33	93	93	FC Barcelona	...
1	20801	Cristiano Ronaldo	C. Ronaldo dos Santos Aveiro	https://cdn.sofifa.com/players/020/801/21_60.png	http://sofifa.com/player/20801/c-ronaldo-dos-s...	Portugal	35	92	92	Juventus	...
2	200389	J. Oblak	Jan Oblak	https://cdn.sofifa.com/players/200/389/21_60.png	http://sofifa.com/player/200389/jan-oblak/210006/	Slovenia	27	91	93	Atlético Madrid	...
3	192985	K. De Bruyne	Kevin De Bruyne	https://cdn.sofifa.com/players/192/985/21_60.png	http://sofifa.com/player/192985/kevin-de-bruyn...	Belgium	29	91	91	Manchester City	...
4	190871	Neymar Jr	Neymar da Silva Santos Jr.	https://cdn.sofifa.com/players/190/871/21_60.png	http://sofifa.com/player/190871/neymar-da-silv...	Brazil	28	91	91	Paris Saint-Germain	...

5 rows × 80 columns

Dropping columns

In [68]:
```
# We are dropping irrelevant columns
df.drop(["playerUrl","Contract"],axis=1, inplace=True)
```

In [69]:
```
df.head()
```

Out[69]:

	ID	Name	LongName	photoUrl	Nationality	Age	↓OVA	POT	Club	Positions	...	PAC	SHO	PAS	DRI	D
0	158023	L. Messi	Lionel Messi	https://cdn.sofifa.com/players/158/023/21_60.png	Argentina	33	93	93	FC Barcelona	RW, ST, CF	...	85	92	91	95	
1	20801	Cristiano Ronaldo	C. Ronaldo dos Santos Aveiro	https://cdn.sofifa.com/players/020/801/21_60.png	Portugal	35	92	92	Juventus	ST, LW	...	89	93	81	89	
2	200389	J. Oblak	Jan Oblak	https://cdn.sofifa.com/players/200/389/21_60.png	Slovenia	27	91	93	Atlético Madrid	GK	...	87	92	78	90	
3	192985	K. De Bruyne	Kevin De Bruyne	https://cdn.sofifa.com/players/192/985/21_60.png	Belgium	29	91	91	Manchester City	CAM, CM	...	76	86	93	88	
4	190871	Neymar Jr	Neymar da Silva Santos Jr.	https://cdn.sofifa.com/players/190/871/21_60.png	Brazil	28	91	91	Paris Saint-Germain	LW, CAM	...	91	85	86	94	

5 rows × 78 columns

In [70]:
```
# Renaming columns
df.rename(columns= {"photoUrl":"Photo URL","↓OVA": "Overall", "POT":"Potential","BOV":"Best Overall Rating",
                     "W/F":"Weak Foot Ability", "SM":"Skill Move","A/W":"Attack WR","D/W":"Defensive WR",
                    "IR":"Injury Resistance", "PAC":"Pace", "SHO":"Shooting",
                    "PAS":"Passing Ability", "DRI":"Dribbling Ability","PHY":"Physical Ability",
                     "Contract_end": "Contract Valid Till", "Contract_start":"Contract Start",
                     "DEF":"Defensive Ability"}, inplace=True)
```

In [71]:
```
df.head()
```

Out[71]:

	ID	Name	LongName	Photo URL	Nationality	Age	Overall	Potential	Club	Positions	...	Pace	Shooting	Pa A
0	158023	L. Messi	Lionel Messi	https://cdn.sofifa.com/players/158/023/21_60.png	Argentina	33	93	93	FC Barcelona	RW, ST, CF	...	85	92	
1	20801	Cristiano Ronaldo	C. Ronaldo dos Santos Aveiro	https://cdn.sofifa.com/players/020/801/21_60.png	Portugal	35	92	92	Juventus	ST, LW	...	89	93	
2	200389	J. Oblak	Jan Oblak	https://cdn.sofifa.com/players/200/389/21_60.png	Slovenia	27	91	93	Atlético Madrid	GK	...	87	92	
3	192985	K. De Bruyne	Kevin De Bruyne	https://cdn.sofifa.com/players/192/985/21_60.png	Belgium	29	91	91	Manchester City	CAM, CM	...	76	86	
4	190871	Neymar Jr	Neymar da Silva Santos Jr.	https://cdn.sofifa.com/players/190/871/21_60.png	Brazil	28	91	91	Paris Saint-Germain	LW, CAM	...	91	85	

5 rows × 78 columns

In [72]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 18979 entries, 0 to 18978
Data columns (total 78 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   ID                  18979 non-null  int64 
 1   Name                18979 non-null  object
 2   LongName            18979 non-null  object
 3   Photo URL           18979 non-null  object
 4   Nationality         18979 non-null  object
 5   Age                 18979 non-null  int64 
 6   Overall             18979 non-null  int64 
 7   Potential           18979 non-null  int64 
 8   Club                18979 non-null  object
 9   Positions           18979 non-null  object
 10  Height              18979 non-null  int64 
 11  Weight              18979 non-null  int64 
 12  Preferred Foot      18979 non-null  object
 13  Best Overall Rating  18979 non-null  int64 
 14  Best Position       18979 non-null  object
 15  Joined              18979 non-null  object
 16  Loan                1013 non-null   object
 17  Value               18979 non-null  int64 
 18  Wage                18979 non-null  object
 19  Release Clause      18979 non-null  int64 
 20  Attacking           18979 non-null  int64 
 21  Crossing            18979 non-null  int64 
 22  Finishing           18979 non-null  int64 
 23  Heading Accuracy    18979 non-null  int64 
 24  Short Passing       18979 non-null  int64 
 25  Volleys             18979 non-null  int64 
 26  Skill               18979 non-null  int64 
 27  Dribbling           18979 non-null  int64 
 28  Curve               18979 non-null  int64 
 29  FK Accuracy         18979 non-null  int64 
 30  Long Passing        18979 non-null  int64 
 31  Ball Control        18979 non-null  int64 
 32  Movement            18979 non-null  int64 
 33  Acceleration        18979 non-null  int64 
 34  Sprint Speed        18979 non-null  int64 
 35  Agility             18979 non-null  int64 
 36  Reactions           18979 non-null  int64 
 37  Balance             18979 non-null  int64 
 38  Power               18979 non-null  int64 
 39  Shot Power          18979 non-null  int64 
 40  Jumping             18979 non-null  int64 
 41  Stamina             18979 non-null  int64 
 42  Strength            18979 non-null  int64 
 43  Long Shots          18979 non-null  int64 
 44  Mentality           18979 non-null  int64 
 45  Aggression          18979 non-null  int64 
 46  Interceptions       18979 non-null  int64 
 47  Positioning         18979 non-null  int64 
 48  Vision              18979 non-null  int64 
 49  Penalties           18979 non-null  int64 
 50  Composure           18979 non-null  int64 
 51  Defending           18979 non-null  int64 
 52  Marking             18979 non-null  int64 
 53  Standing Tackle     18979 non-null  int64 
 54  Sliding Tackle      18979 non-null  int64 
 55  Goalkeeping         18979 non-null  int64 
 56  GK Diving           18979 non-null  int64 
 57  GK Handling         18979 non-null  int64 
 58  GK Kicking          18979 non-null  int64 
 59  GK Positioning      18979 non-null  int64 
 60  GK Reflexes         18979 non-null  int64 
 61  Total Stats         18979 non-null  int64 
 62  Base Stats          18979 non-null  int64 
 63  Weak Foot Ability   18979 non-null  int64 
 64  Skill Move          18979 non-null  int64 
 65  Attack WR           18979 non-null  object
 66  Defensive WR        18979 non-null  object
 67  Injury Resistance   18979 non-null  int64 
 68  Pace                18979 non-null  int64 
 69  Shooting            18979 non-null  int64 
 70  Passing Ability     18979 non-null  int64 
 71  Dribbling Ability   18979 non-null  int64 
 72  Defensive Ability   18979 non-null  int64 
 73  Physical Ability    18979 non-null  int64 
 74  Hits                18979 non-null  object
 75  Contract Valid Till  18979 non-null  object
 76  Contract Start      18979 non-null  object
 77  Player Status       18979 non-null  object
dtypes: int64(61), object(17)
memory usage: 11.4+ MB
```

In [73]:
```
df.to_csv("Clean FIFA21 data.csv")
```

From all indications from info above, we have cleaned the data properly.

In [74]:
```
df.describe()
```

Out[74]:

	ID	Age	Overall	Potential	Height	Weight	Best Overall Rating	Value	Release Clause	Attacking	...	E
count	18979.000000	18979.000000	18979.000000	18979.000000	18979.000000	18979.000000	18979.000000	1.897900e+04	1.897900e+04	18979.000000	...	189
mean	226403.384794	25.194109	65.718636	71.136414	181.199220	75.018494	66.751726	2.865063e+06	3.962951e+06	248.938142	...	3
std	27141.054157	4.710520	6.968999	6.114635	6.840033	7.073402	6.747193	7.685154e+06	9.772762e+06	74.299428	...	
min	41.000000	16.000000	47.000000	47.000000	155.000000	50.000000	48.000000	0.000000e+00	0.000000e+00	42.000000	...	2
25%	210135.000000	21.000000	61.000000	67.000000	176.000000	70.000000	62.000000	4.750000e+05	4.235000e+05	222.000000	...	3
50%	232418.000000	25.000000	66.000000	71.000000	181.000000	75.000000	67.000000	9.500000e+05	1.000000e+06	263.000000	...	3
75%	246922.500000	29.000000	70.000000	75.000000	186.000000	80.000000	71.000000	2.000000e+06	2.800000e+06	297.000000	...	3
max	259216.000000	53.000000	93.000000	95.000000	206.000000	110.000000	93.000000	1.855000e+08	2.031000e+08	437.000000	...	4

8 rows × 61 columns

We can check the correlation between columns

In [75]:
```
pd.set_option('display.max_rows', None)
df.corr()
```

```
ric_only to silence this warning.
  df.corr()
```

Out[75]:

	ID	Age	Overall	Potential	Height	Weight	Best Overall Rating	Value	Release Clause	Attacking	...	Base Stats	Weak Foot Ability	Skill Move
ID	1.000000	-0.753413	-0.486968	0.023736	-0.108101	-0.209691	-0.443686	-0.131001	-0.161860	-0.180955	...	-0.434793	-0.106433	-0.123692
Age	-0.753413	1.000000	0.466140	-0.269473	0.090020	0.241859	0.401796	0.040994	0.074079	0.146765	...	0.390236	0.071559	0.060805
Overall	-0.486968	0.466140	1.000000	0.632166	0.033110	0.147845	0.987149	0.552893	0.599142	0.446337	...	0.845894	0.222609	0.381024
Potential	0.023736	-0.269473	0.632166	1.000000	-0.009992	-0.024704	0.669677	0.528200	0.548897	0.284542	...	0.520473	0.163596	0.298001
Height	-0.108101	0.090020	0.033110	-0.009992	1.000000	0.772042	0.022208	0.004099	0.003841	-0.364827	...	-0.104219	-0.166392	-0.417961
Weight	-0.209691	0.241859	0.147845	-0.024704	0.772042	1.000000	0.128448	0.034004	0.039476	-0.275463	...	0.014983	-0.125865	-0.344635
Best Overall Rating	-0.443686	0.401796	0.987149	0.669677	0.022208	0.128448	1.000000	0.563253	0.608117	0.487301	...	0.841199	0.236708	0.415148
Value	-0.131001	0.040994	0.552893	0.528200	0.004099	0.034004	0.563253	1.000000	0.966440	0.259654	...	0.462458	0.144755	0.260043

When explore the correlation table, you will see some player attributes correlates with othe

We can explore some visualizations

Importing Python visualizations

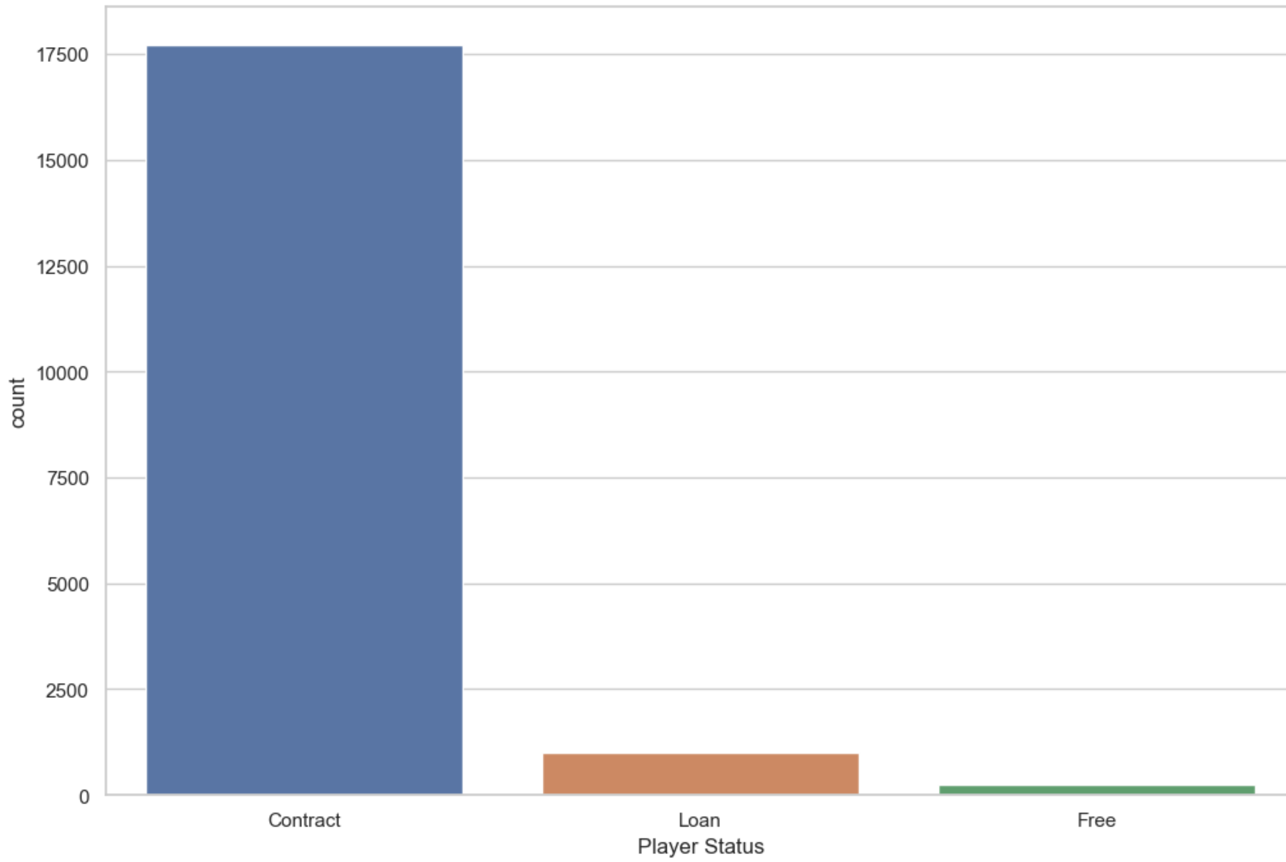
In [78]:
```
import chart_studio.plotly as py
import cufflinks as cf
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
%matplotlib inline
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
cf.go_offline()
```

We set seaborn theme to whitegrid for better visualization

In [79]:
```
sns.set_theme(style="whitegrid")
```

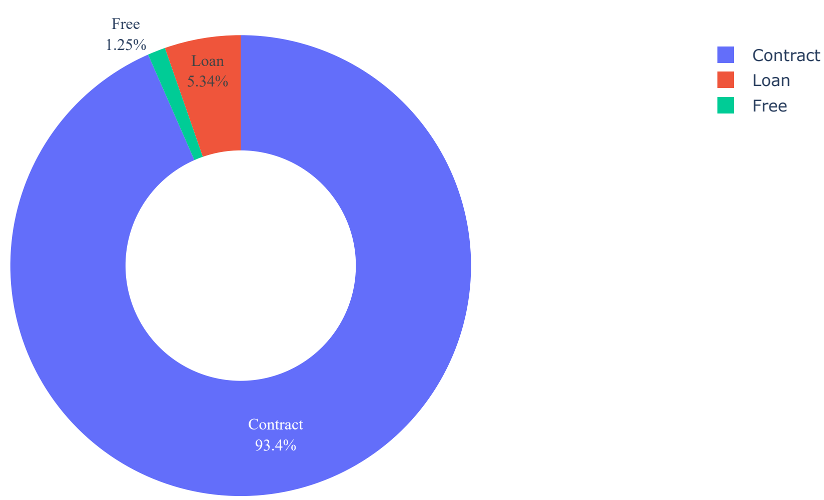
In [80]:
```
plt.figure(figsize=(12,8))
sns.countplot(data=df, x="Player Status")
```

Out[80]:
```
<Axes: xlabel='Player Status', ylabel='count'>
```



Creating donut to show the percentage of contract, loan and free players

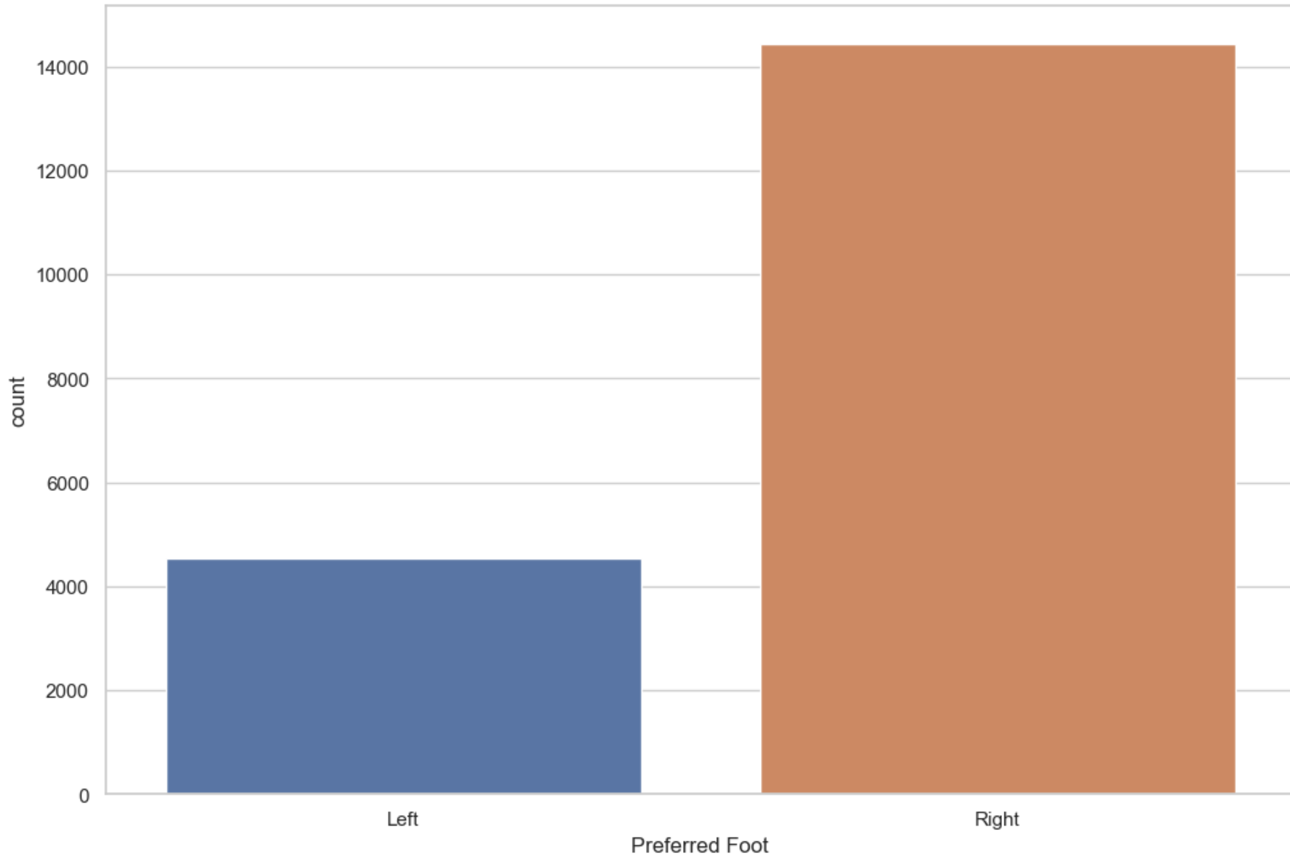
In [81]:
```
values= list(df["Player Status"].value_counts())
labels= list(df["Player Status"].unique())

fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.5)])
fig.update_traces(textposition='auto',textinfo='percent+label', textfont={"family":"Droid San"})
fig.show()
```



Countplot compares count of Left and Right footed players

In [82]:
```
plt.figure(figsize=(12,8))
sns.countplot(data=df, x="Preferred Foot",)
```

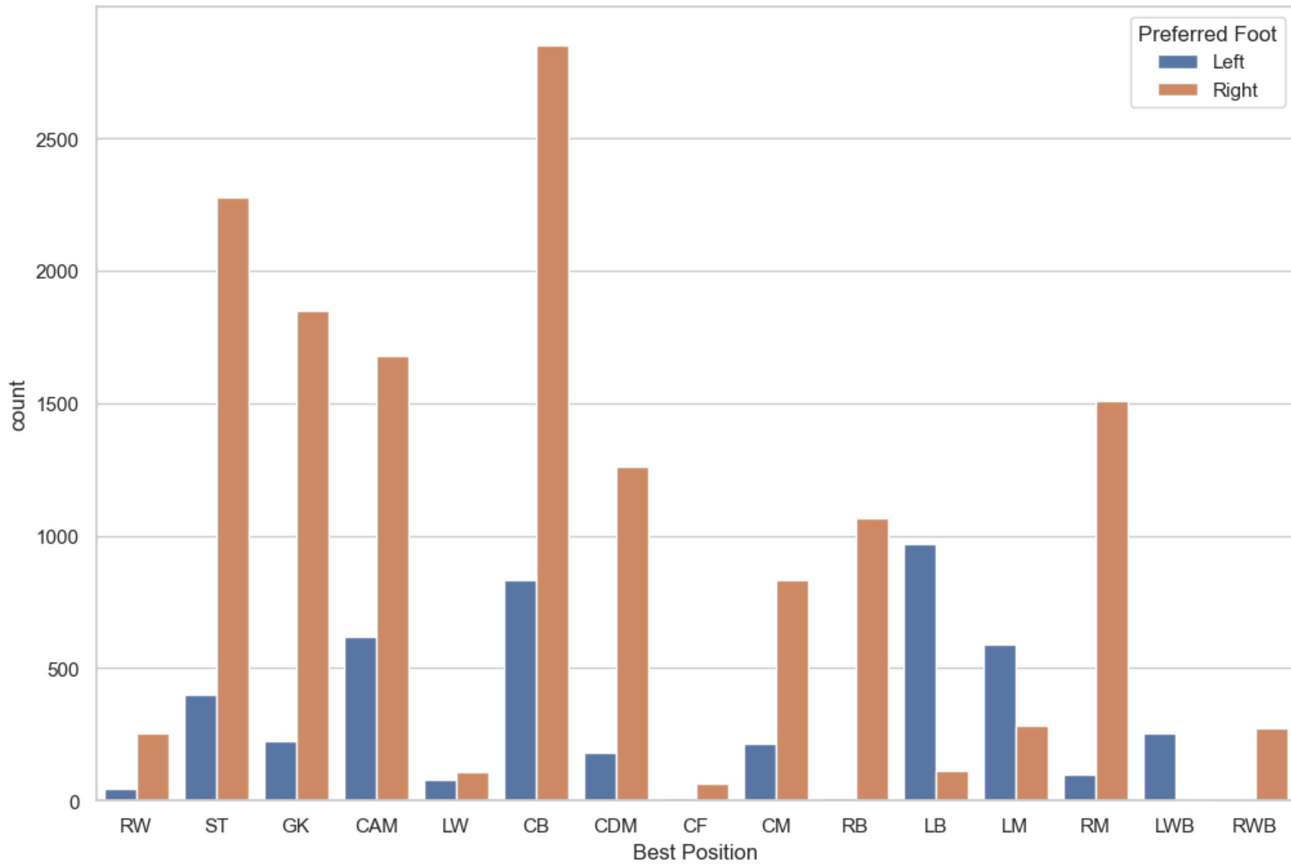
Out[82]: <Axes: xlabel='Preferred Foot', ylabel='count'>



Count Plot shows numbers of Right and Left footed players in each postions

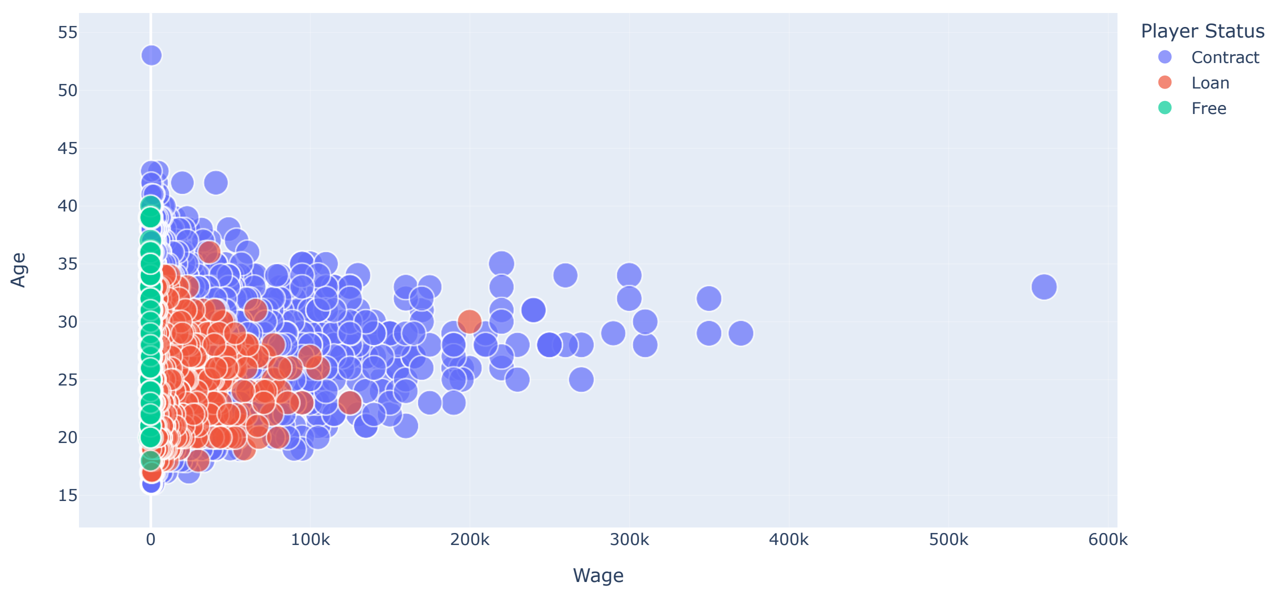
In [83]:
```
plt.figure(figsize=(12,8))
sns.countplot(data=df,x="Best Position" ,hue="Preferred Foot")
```

Out[83]: <Axes: xlabel='Best Position', ylabel='count'>



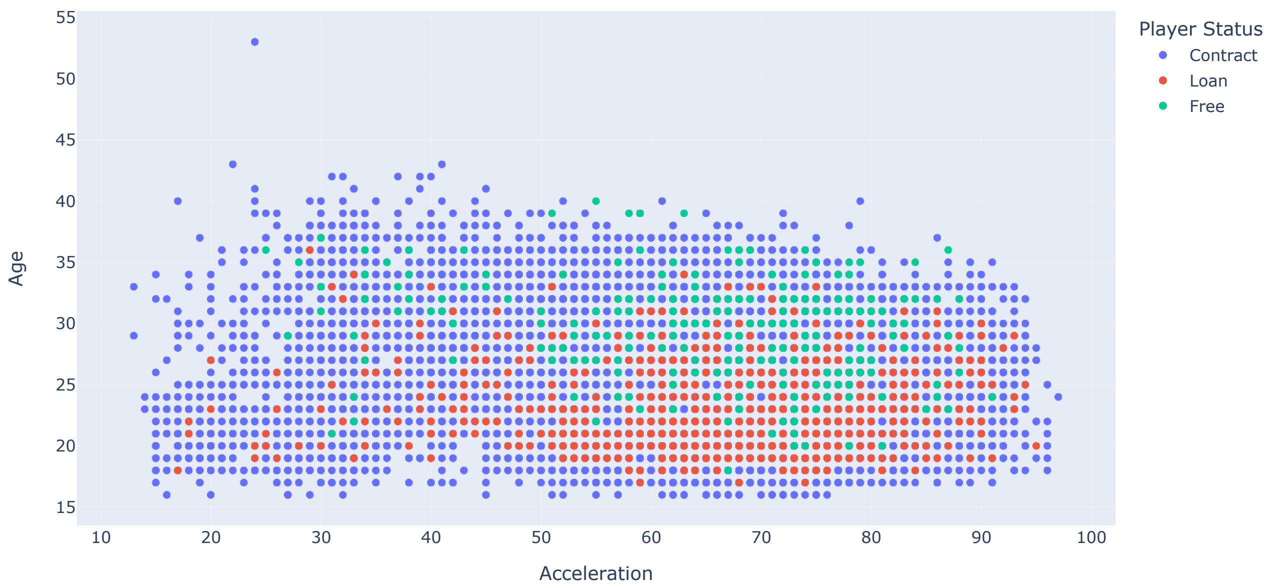
Scatter Plot showing relationship between Age and Wage

In [84]:
```
fig = px.scatter(df, x="Wage", y="Age", color="Player Status",
                 size='Overall', hover_data=["Name",'Club'])
fig.show()
```



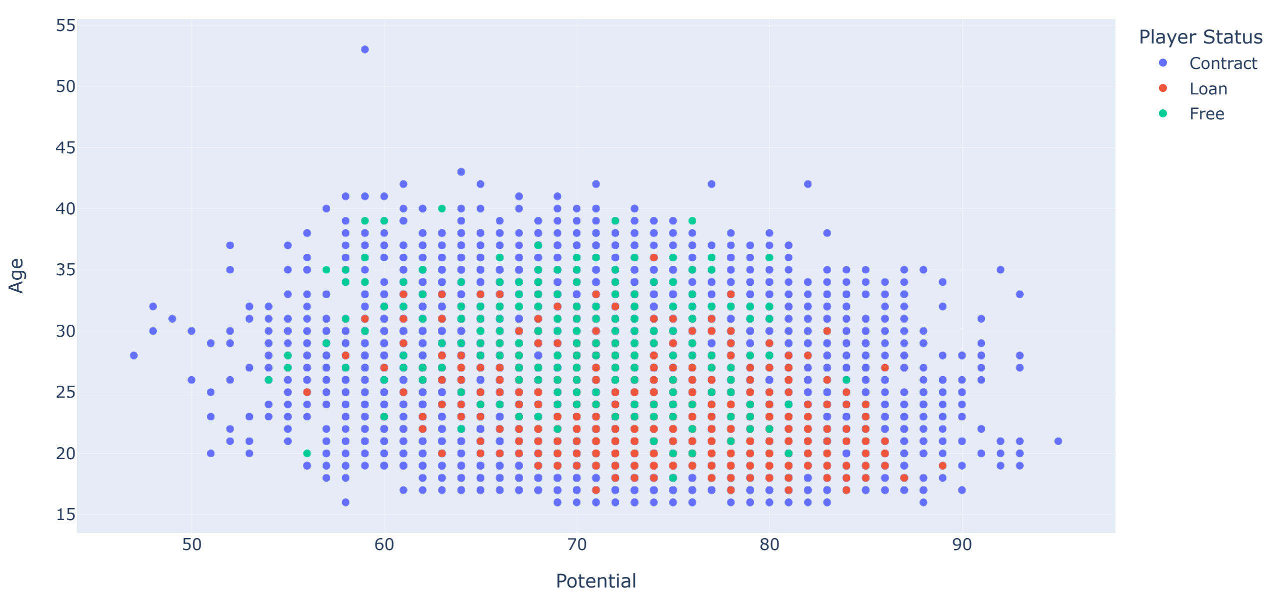
Scatter Plot showing relationship between Acceleration and Wage

In [86]:
```
fig = px.scatter(df, x="Acceleration", y="Age", color="Player Status", hover_data=["Name",'Club'])
fig.show()
```



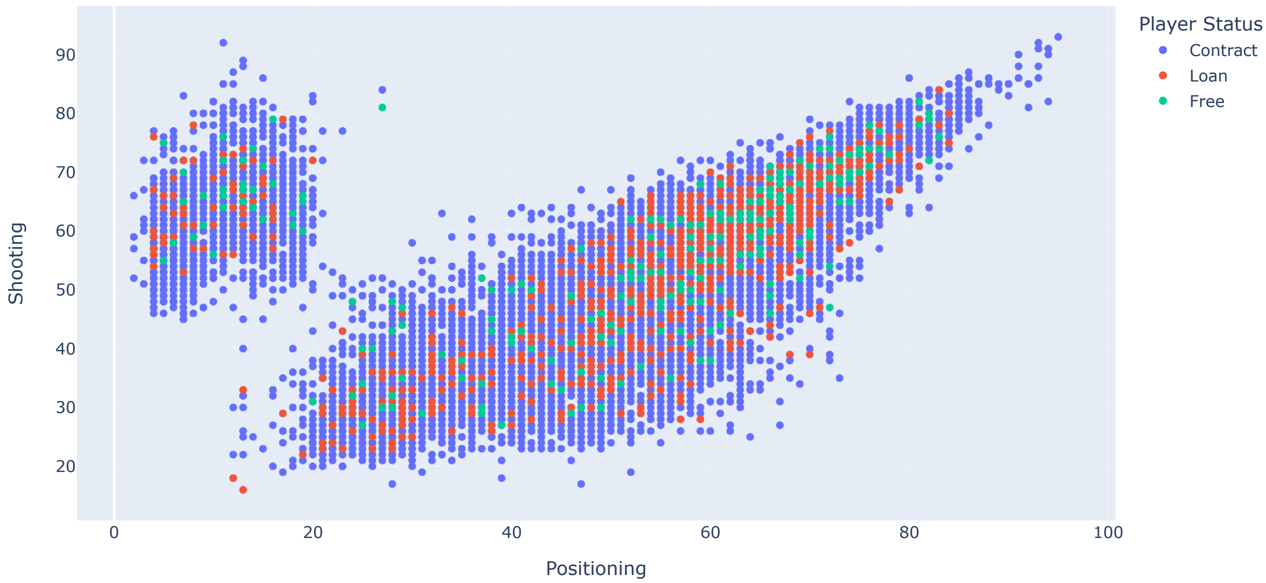
Scatter Plot showing relationship between Age and Potential

In [87]:
```
fig = px.scatter(df, x="Potential", y="Age", color="Player Status", hover_data=["Name",'Club'])
fig.show()
```



Scatter Plot showing relationship between Positioning and Shooting

In [88]:
```
fig = px.scatter(df, x="Positioning", y="Shooting", color="Player Status", hover_data=["Name",'Club'])
fig.show()
```



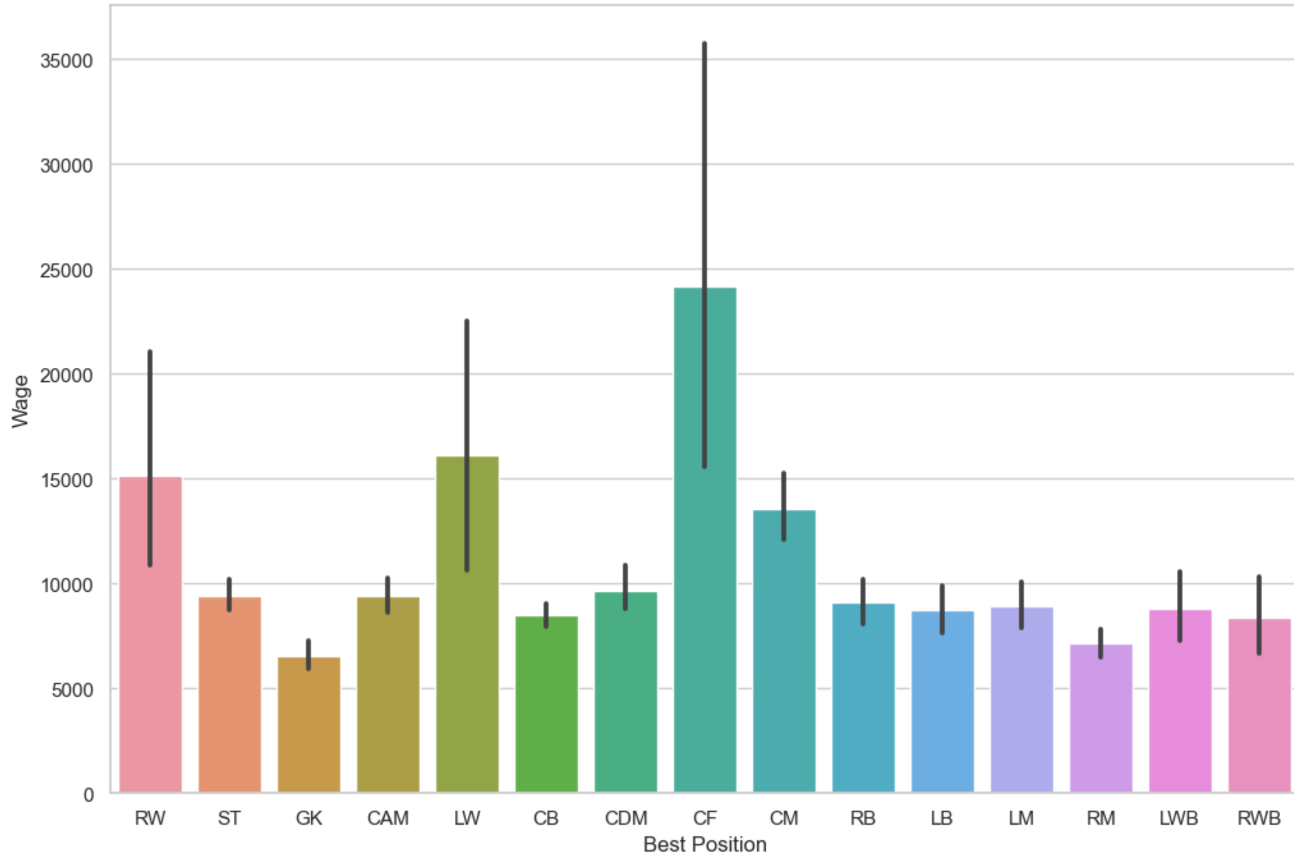
You observe the plot looks divided, small fraction are Goal Keepers and larger fraction are other positions aside GK If you hover over the plot, You see other player detail.

Bar plot show Position and Wages

In [89]:
```
df["Wage"]=df["Wage"].astype(int)
```

In [90]:
```
plt.figure(figsize=(12,8))

sns.barplot(data=df, x="Best Position", y="Wage")
```

Out[90]: <Axes: xlabel='Best Position', ylabel='Wage'>



In [ ]: