Data cleaning project - FIFA 2021 Database

1. Import Libraries

In [1]:

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

2. Read the dataset

In [2]:

```
fifa21_rawdata = pd.read_csv('fifa21_raw_data_v2.csv')
fifa21_rawdata
# fifa21_rawdata.shape
```

```
C:\Users\imgal\AppData\Local\Temp\ipykernel_11392\2118387280.py:1: DtypeWa
rning: Columns (76) have mixed types. Specify dtype option on import or se
t low_memory=False.
  fifa21_rawdata = pd.read_csv('fifa21_raw_data_v2.csv')
```

Out[2]:

	ID	Name	LongName	photoUrl	
0	158023	L. Messi	Lionel Messi	https://cdn.sofifa.com/players/158/023/21_60.png	http://sofifa
1	20801	Cristiano Ronaldo	C. Ronaldo dos Santos Aveiro	https://cdn.sofifa.com/players/020/801/21_60.png	http:/
2	200389	J. Oblak	Jan Oblak	https://cdn.sofifa.com/players/200/389/21_60.png	http://so
3	192985	K. De Bruyne	Kevin De Bruyne	https://cdn.sofifa.com/players/192/985/21_60.png	http://sofifa
4	190871	Neymar Jr	Neymar da Silva Santos Jr.	https://cdn.sofifa.com/players/190/871/21_60.png	http://sofifa.c
...	...	...	...	...	
18974	247223	Xia Ao	Ao Xia	https://cdn.sofifa.com/players/247/223/21_60.png	http://so
18975	258760	B. Hough	Ben Hough	https://cdn.sofifa.com/players/258/760/21_60.png	http://sof
18976	252757	R. McKinley	Ronan McKinley	https://cdn.sofifa.com/players/252/757/21_60.png	http://sofifa
18977	243790	Wang Zhen'ao	Zhen'ao Wang	https://cdn.sofifa.com/players/243/790/21_60.png	http://sofifa.c
18978	252520	Zhou Xiao	Xiao Zhou	https://cdn.sofifa.com/players/252/520/21_60.png	http://sofi

18979 rows × 77 columns

In [3]:

```
# View the data
fifa21_rawdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18979 entries, 0 to 18978
Data columns (total 77 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   photoUrl          18979 non-null  object
 4   playerUrl         18979 non-null  object
 5   Nationality       18979 non-null  object
 6   Age               18979 non-null  int64
 7   ↓OVA              18979 non-null  int64
 8   POT               18979 non-null  int64
 9   Club              18979 non-null  object
 10  Contract          18979 non-null  object
 11  Positions         18979 non-null  object
 12  Height            18979 non-null  object
 13  Weight            18979 non-null  object
 14  Preferred Foot    18979 non-null  object
 15  BOV               18979 non-null  int64
 16  Best Position     18979 non-null  object
 17  Joined            18979 non-null  object
 18  Loan Date End     1013 non-null   object
 19  Value             18979 non-null  object
 20  Wage              18979 non-null  object
 21  Release Clause    18979 non-null  object
 22  Attacking         18979 non-null  int64
 23  Crossing          18979 non-null  int64
 24  Finishing         18979 non-null  int64
 25  Heading Accuracy  18979 non-null  int64
 26  Short Passing     18979 non-null  int64
 27  Volleys           18979 non-null  int64
 28  Skill             18979 non-null  int64
 29  Dribbling         18979 non-null  int64
 30  Curve             18979 non-null  int64
 31  FK Accuracy       18979 non-null  int64
 32  Long Passing      18979 non-null  int64
 33  Ball Control      18979 non-null  int64
 34  Movement          18979 non-null  int64
 35  Acceleration      18979 non-null  int64
 36  Sprint Speed      18979 non-null  int64
 37  Agility           18979 non-null  int64
 38  Reactions         18979 non-null  int64
 39  Balance           18979 non-null  int64
 40  Power             18979 non-null  int64
 41  Shot Power        18979 non-null  int64
 42  Jumping           18979 non-null  int64
 43  Stamina           18979 non-null  int64
 44  Strength          18979 non-null  int64
 45  Long Shots        18979 non-null  int64
 46  Mentality         18979 non-null  int64
 47  Aggression        18979 non-null  int64
 48  Interceptions     18979 non-null  int64
 49  Positioning       18979 non-null  int64
 50  Vision            18979 non-null  int64
 51  Penalties         18979 non-null  int64
 52  Composure         18979 non-null  int64
 53  Defending         18979 non-null  int64
 54  Marking           18979 non-null  int64
 55  Standing Tackle   18979 non-null  int64
 56  Sliding Tackle    18979 non-null  int64
 57  Goalkeeping       18979 non-null  int64
 58  GK Diving         18979 non-null  int64
 59  GK Handling       18979 non-null  int64
 60  GK Kicking        18979 non-null  int64
 61  GK Positioning    18979 non-null  int64
 62  GK Reflexes       18979 non-null  int64
 63  Total Stats       18979 non-null  int64
 64  Base Stats        18979 non-null  int64
 65  W/F               18979 non-null  object
 66  SM                18979 non-null  object
 67  A/W               18979 non-null  object
 68  D/W               18979 non-null  object
 69  IR                18979 non-null  object
 70  PAC               18979 non-null  int64
 71  SHO               18979 non-null  int64
 72  PAS               18979 non-null  int64
 73  DRI               18979 non-null  int64
 74  DEF               18979 non-null  int64
 75  PHY               18979 non-null  int64
 76  Hits              16384 non-null  object
dtypes: int64(54), object(23)
memory usage: 11.1+ MB
```

3. Data cleaning

3.1 Copy our data

I made a copy of the database in case I need to go back to the original data

In [4]:

```
# f21_copy = fifa21_rawdata.copy()
fdc = fifa21_rawdata
```

3.2 Name and Lastname

Let´s replace the columns 'Name' and 'LongName', with two other columns with the 'Name' and 'Surname'

In [5]:

```
fdc['LongName'].dtype
```

Out[5]:

```
dtype('O')
```

In [6]:

```python
# The name of the column was changed to better identify the values in i
fdc = fdc.rename(columns={'LongName':'Surname'})

def split_names(df_3):
    # Split the 'LongName' column into two columns using the space char
    df_3[['TempName', 'Surname']] = df_3['Surname'].str.split(n=1, expa

    # For the 'Name' column, keeping only the first part after splittir
    df_3['Name'] = df_3['Name'].str.split().str[0]

    # The pattern is used to identify the names similar to L. or K.
    pattern = r'^[A-Z]\.'

    # Checking the pattern of the names to modify them if necessary
    condition = (df_3['Name'].str.contains(pattern)) | (df_3['Name'] ==

    # Use loc to update values based on the condition
    df_3.loc[condition, 'Name'] = df_3['TempName']

    # Drop the temporary column when is no more needed
    df_3 = df_3.drop(columns=['TempName'])

    return df_3

fdc = split_names(fdc)
fdc[['Name', 'Surname']].head(10)
```

Out[6]:

	Name	Surname
0	Lionel	Messi
1	Cristiano	Ronaldo dos Santos Aveiro
2	Jan	Oblak
3	Kevin	De Bruyne
4	Neymar	da Silva Santos Jr.
5	Robert	Lewandowski
6	Mohamed	Salah
7	Alisson	Ramses Becker
8	Kylian	Mbappé
9	Marc-André	ter Stegen

3.3 Photo and Player

They are not going to be used in the analysis, so I decided to delete them

In [7]:

```python
fdc = fdc.drop(['photoUrl','playerUrl'], axis=1)
```

3.4 Nationality

Let´s check if the values are ok

In [8]:
```
fdc['Nationality'].dtype
```

Out[8]:
```
dtype('O')
```

In [9]:
```
fdc['Nationality'].unique()
```

Out[9]:
```
array(['Argentina', 'Portugal', 'Slovenia', 'Belgium', 'Brazil', 'Poland',
       'Egypt', 'France', 'Germany', 'Netherlands', 'Senegal', 'Spain',
       'England', 'Scotland', 'Korea Republic', 'Costa Rica', 'Italy',
       'Gabon', 'Croatia', 'Uruguay', 'Switzerland', 'Serbia', 'Slovakia',
       'Morocco', 'Algeria', 'Denmark', 'Hungary', 'Bosnia Herzegovina',
       'Nigeria', 'Cameroon', 'Norway', 'Ghana', 'Mexico', 'Austria',
       'Albania', 'Colombia', 'Chile', 'Ivory Coast', 'Greece', 'Finland',
       'Wales', 'Sweden', 'Togo', 'Czech Republic', 'Russia', 'Venezuela',
       'Canada', 'United States', 'Guinea', 'Montenegro', 'Israel',
       'Republic of Ireland', 'Ukraine', 'Turkey', 'Ecuador', 'Jamaica',
       'DR Congo', 'Australia', 'China PR', 'Armenia', 'Northern Ireland',
       'North Macedonia', 'Kosovo', 'Mali', 'Peru',
       'Central African Republic', 'Iceland', 'Burkina Faso', 'Paraguay',
       'Japan', 'Romania', 'New Zealand', 'Iran', 'Angola', 'Tunisia',
       'Syria', 'Dominican Republic', 'Cape Verde', 'Equatorial Guinea',
       'Kenya', 'Georgia', 'Panama', 'Zambia', 'Tanzania', 'Zimbabwe',
       'Congo', 'South Africa', 'Moldova', 'Mozambique', 'Iraq',
       'Guinea Bissau', 'Honduras', 'Cuba', 'Cyprus', 'Lithuania',
       'Estonia', 'Madagascar', 'Benin', 'Curacao', 'Saudi Arabia',
       'Gambia', 'Uzbekistan', 'Chad', 'United Arab Emirates',
       'Saint Kitts and Nevis', 'Libya', 'Sierra Leone', 'Philippines',
       'Liberia', 'Bulgaria', 'Comoros', 'Namibia', 'Luxembourg',
       'Trinidad & Tobago', 'Bermuda', 'Thailand', 'Burundi',
       'New Caledonia', 'Puerto Rico', 'Bolivia', 'Kazakhstan',
       'Antigua & Barbuda', 'Latvia', 'Malawi', 'Montserrat',
       'São Tomé & Príncipe', 'El Salvador', 'Mauritania', 'Jordan',
       'Eritrea', 'Aruba', 'Uganda', 'Chinese Taipei', 'Azerbaijan',
       'Afghanistan', 'Faroe Islands', 'Haiti', 'Sudan', 'Grenada',
       'Lebanon', 'Guam', 'Palestine', 'Belarus', 'Guyana', 'Rwanda',
       'Liechtenstein', 'Saint Lucia', 'Papua New Guinea', 'India',
       'Ethiopia', 'Belize', 'Andorra', 'Guatemala', 'Malta', 'Niger',
       'Korea DPR', 'Barbados', 'Macau', 'South Sudan', 'Singapore',
       'Hong Kong', 'Nicaragua', 'Malaysia', 'Indonesia'], dtype=object)
```

3.5 Age

Verifying if the values are ok for the ages.

In [10]:
```
fdc['Age'].dtype
```

Out[10]:
```
dtype('int64')
```

In [11]:
```
fdc['Age'].unique()
```

Out[11]:
```
array([33, 35, 27, 29, 28, 31, 21, 34, 32, 25, 26, 30, 20, 24, 22, 23, 19,
       38, 42, 36, 37, 18, 17, 39, 40, 41, 16, 43, 53], dtype=int64)
```

3.6 OVA and POT

The columns OVA and POT refer to OVERALL and POTENTIAL. Let´s change the name of the columns to make it clearer. It also important that our data type is int64 for future operations if is necessary

3.6.1 OVA

```
In [12]:  fdc['↓OVA'].dtype
```

```
Out[12]: dtype('int64')
```

```
In [13]:  fdc['↓OVA'].unique()
```

```
Out[13]: array([93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77,
                76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60,
                59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47], dtype=int64)
```

3.6.2 POT

```
In [14]:  fdc['POT'].dtype
```

```
Out[14]: dtype('int64')
```

```
In [15]:  fdc['POT'].unique()
```

```
Out[15]: array([93, 92, 91, 90, 95, 89, 88, 87, 86, 85, 84, 83, 82, 81, 80, 79, 78,
                77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62, 61,
                60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47],
               dtype=int64)
```

3.6.3 Changing the column's names

In [16]:

```python
# Now let´s change the name of the columns
fdc = fdc.rename(columns={'↓OVA':'Overall','POT':'Potential'})
fdc
```

Out[16]:

	ID	Name	Surname	Nationality	Age	Overall	Potential	Club	Cc
0	158023	Lionel	Messi	Argentina	33	93	93	\n\n\n\nFC Barcelona	
1	20801	Cristiano	Ronaldo dos Santos Aveiro	Portugal	35	92	92	\n\n\n\nJuventus	
2	200389	Jan	Oblak	Slovenia	27	91	93	\n\n\n\nAtlético Madrid	
3	192985	Kevin	De Bruyne	Belgium	29	91	91	\n\n\n\nManchester City	
4	190871	Neymar	da Silva Santos Jr.	Brazil	28	91	91	\n\n\n\nParis Saint-Germain	
...	...	...	...	...	...	...	...	...	
18974	247223	Ao	Xia	China PR	21	47	55	\n\n\n\nWuhan Zall	
18975	258760	Ben	Hough	England	17	47	67	\n\n\n\nOldham Athletic	
18976	252757	Ronan	McKinley	England	18	47	65	\n\n\n\nDerry City	
18977	243790	Zhen'ao	Wang	China PR	20	47	57	\n\n\n\nDalian YiFang FC	
18978	252520	Xiao	Zhou	China PR	21	47	57	\n\n\n\nDalian YiFang FC	

18979 rows × 75 columns

3.7 Club

Checking the names of the clubs, and transforming them into strings data type

In [17]:

```python
fdc['Club'].dtype
```

Out[17]:

```
dtype('O')
```

In [18]:
```
fdc['Club'].unique()
```

Out[18]:
```
array(['\n\n\n\nFC Barcelona', '\n\n\n\nJuventus',
       '\n\n\n\nAtlético Madrid', '\n\n\n\nManchester City',
       '\n\n\n\nParis Saint-Germain', '\n\n\n\nFC Bayern München',
       '\n\n\n\nLiverpool', '\n\n\n\nReal Madrid', '\n\n\n\nChelsea',
       '\n\n\n\nTottenham Hotspur', '\n\n\n\nInter', '\n\n\n\nNapoli',
       '\n\n\n\nBorussia Dortmund', '\n\n\n\nManchester United',
       '\n\n\n\nArsenal', '\n\n\n\nLazio', '\n\n\n\nLeicester City',
       '\n\n\n\nBorussia Mönchengladbach', '\n\n\n\nReal Sociedad',
       '\n\n\n\nAtalanta', '\n\n\n\nOlympique Lyonnais', '\n\n\n\nMila
n',
       '\n\n\n\nVillarreal CF', '\n\n\n\nRB Leipzig', '\n\n\n\nCagliar
i',
       '\n\n\n\nAjax', '\n\n\n\nSL Benfica', '\n\n\n\nAS Monaco',
       '\n\n\n\nWolverhampton Wanderers', '\n\n\n\nEverton',
       '\n\n\n\nFiorentina', '\n\n\n\nFC Porto', '\n\n\n\nRC Celta',
       '\n\n\n\nTorino', '\n\n\n\nSevilla FC', '\n\n\n\nGrêmio',
       '\n\n\n\nReal Betis', '\n\n\n\nRoma', '\n\n\n\nNewcastle Unite
d',
       '\n\n\n\nEintracht Frankfurt', '\n\n\n\nValencia CF',
```

In [19]:
```
fdc['Club'] = fdc['Club'].str.strip()
fdc['Club'].unique()
```

Out[19]:
```
array(['FC Barcelona', 'Juventus', 'Atlético Madrid', 'Manchester Cit
y',
       'Paris Saint-Germain', 'FC Bayern München', 'Liverpool',
       'Real Madrid', 'Chelsea', 'Tottenham Hotspur', 'Inter', 'Napol
i',
       'Borussia Dortmund', 'Manchester United', 'Arsenal', 'Lazio',
       'Leicester City', 'Borussia Mönchengladbach', 'Real Sociedad',
       'Atalanta', 'Olympique Lyonnais', 'Milan', 'Villarreal CF',
       'RB Leipzig', 'Cagliari', 'Ajax', 'SL Benfica', 'AS Monaco',
       'Wolverhampton Wanderers', 'Everton', 'Fiorentina', 'FC Porto',
       'RC Celta', 'Torino', 'Sevilla FC', 'Grêmio', 'Real Betis', 'Rom
a',
       'Newcastle United', 'Eintracht Frankfurt', 'Valencia CF',
       'Medipol Başakşehir FK', 'Inter Miami', 'Bayer 04 Leverkusen',
       'Levante UD', 'Crystal Palace', 'Athletic Club de Bilbao',
       'Shanghai SIPG FC', 'VfL Wolfsburg',
       'Guangzhou Evergrande Taobao FC', 'Al Shabab',
       'Olympique de Marseille', 'Los Angeles FC',
       'Beijing Sinobo Guoan FC', 'Getafe CF', 'SV Werder Bremen',
```

3.8 Contract

With this column, the data is going to be split and differentiated between a player under contract, on loan, or free.

In [20]:
```
fdc['Contract'].dtype
```

Out[20]:
```
dtype('O')
```

In [21]:

```
fdc['Contract'].unique()
```

Out[21]:

```
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)
```

3.8.1 Searching for the values 'On Loan' and 'Free'

In [22]:

```python
# Let´s search for the values 'On Loan' and 'Free' in 'Contract'
for index, row in fdc.iterrows():
    if 'On Loan' in row['Contract'] or 'Free' in row['Contract']:
        print(row['Contract'])
```

```
Jun 30, 2021 On Loan
Jun 30, 2021 On Loan
Jun 30, 2021 On Loan
Free
Free
Jun 30, 2021 On Loan
Jun 30, 2021 On Loan
Jun 30, 2021 On Loan
Jun 30, 2021 On Loan
Free
Free
Free
Free
Free
Free
Free
Free
Free
Jun 30, 2021 On Loan
```

3.8.2 Extracting the values into new columns

Let´s extract the values and put them into different columns. Identifying when the contract starts or ends, with those values, we want to know the contract length. For contracts 'Free' or 'On Loan' we use NaN values.

In [23]:

```python
# Let´s use a function to extract the values into new columns
def extract_contract_info(contract):
    if contract == 'Free' or 'On Loan' in contract:
        start_date = np.nan
        end_date = np.nan
        contract_length = 0
    else:
        start_date, end_date = contract.split(' ~ ')
        start_year = int(start_date[:4])
        end_year = int(end_date[:4])
        contract_length = end_year - start_year
    return start_date, end_date, contract_length

# We apply the function to the 'Contract' column. This will create 3 ne
new_cols = ['Contract Start', 'Contract End', 'Contract Length(years)']
new_data = fdc['Contract'].apply(lambda x: pd.Series(extract_contract_i

# Now we need a loop to go through all the values of the column
for i in range(len(new_cols)):
    fdc.insert(loc=fdc.columns.get_loc('Contract')+1+i, column=new_cols
```

In [24]:

```python
# Run to see the changes
fdc_new_col = fdc[['Contract', 'Contract Start', 'Contract End', 'Contr
fdc_new_col.head()
```

Out[24]:

	Contract	Contract Start	Contract End	Contract Length(years)
0	2004 ~ 2021	2004	2021	17.0
1	2018 ~ 2022	2018	2022	4.0
2	2014 ~ 2023	2014	2023	9.0
3	2015 ~ 2023	2015	2023	8.0
4	2017 ~ 2022	2017	2022	5.0

It´s time to determine the contract status of the players by using a function. Reflecting the results in a new column

In [25]:

```python
# Let´s define the contract categories
def categorize_contract_status(contract):
    if contract == 'Free':
        return 'Free'
    elif 'On Loan' in contract:
        return 'On Loan'
    else:
        return 'Contract'

# Add the new column 'Contract Status'
fdc.insert(fdc.columns.get_loc('Contract Length(years)')+1, 'Contract S
fdc_new_col = fdc[['Contract', 'Contract Start', 'Contract End', 'Contr
fdc_new_col.sample(5)
```

Out[25]:

	Contract	Contract Start	Contract End	Contract Length(years)	Contract Status
8515	Jun 30, 2021 On Loan	NaN	NaN	0.0	On Loan
7857	2020 ~ 2024	2020	2024	4.0	Contract
18063	2019 ~ 2021	2019	2021	2.0	Contract
17428	2018 ~ 2020	2018	2020	2.0	Contract
2662	2020 ~ 2021	2020	2021	1.0	Contract

3.9 Positions

Let´s check the values from the column Positions

In [26]:

```python
fdc['Positions'].dtype
```

Out[26]:

```
dtype('O')
```

In [27]:
```
fdc['Positions'].unique()
```

Out[27]:
```
array(['RW, ST, CF', 'ST, LW', 'GK', 'CAM, CM', 'LW, CAM', 'ST', 'RW',
       'ST, LW, RW', 'CB', 'LW', 'CDM', 'CF, ST', 'LW, RW', 'CDM, CM',
       'CDM, RB', 'CF, CAM', 'LW, ST', 'CM', 'ST, CF, LW', 'RM, LM, CA
M',
       'RB', 'RW, CAM, CM', 'LB', 'LM, CF', 'CF', 'RW, LW', 'CAM, RM, R
W',
       'CM, CDM', 'CAM, CF, ST', 'CM, CDM, CAM', 'CF, LW, CAM',
       'CAM, RM, CF', 'LM, ST', 'RM, LM, RW', 'LM', 'CAM, RW', 'CB, CD
M',
       'RW, RM', 'LW, CF', 'CM, RM, LM', 'LB, LM', 'CAM, CM, RM',
       'CAM, CM, CF', 'CAM, CF', 'LM, RM, LW', 'LM, LB, CM', 'CM, LM, L
B',
       'RM, RW', 'RM, CM', 'CAM, CM, LW', 'CB, LB', 'RM, RB', 'ST, RW',
       'LM, RW, LW', 'RB, LB', 'RB, RM', 'RM', 'LM, RM, CF', 'CAM, RM',
       'RB, RWB', 'CDM, CB, CM', 'CAM, RM, ST', 'LM, LW, RM', 'CM, CA
M',
       'ST, RM, CF', 'LM, RM', 'RM, CF', 'LM, LWB', 'RW, RM, CF',
       'RB, CM', 'LW, CAM, RW', 'CAM, LW, CM', 'CM, CAM, CDM',
       'RW, LW, CAM', 'CM, CAM, LM', 'CM, RM, ST', 'CDM, CM, RB',
```

In [28]:
```
missing_values = fdc['Positions'].isnull().sum()
print("Number of missing values: ", missing_values)
```

```
Number of missing values:  0
```

3.10 Height

Player's height in cm. Transform all the data to be 'int' data type. Also, transform the values expressed as feet and inches to the same values as the ones expressed in cm.

In [29]:
```
# Let´s check the data type of the values
fdc['Height'].dtype
```

Out[29]:
```
dtype('O')
```

In [30]:
```
fdc['Height'].unique()
```

Out[30]:
```
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=objec
t)
```

In [31]:

```
# Let´s create a function to clean all the numb and transform the other
def convert_height(height):
    if 'cm' in height:
        return int(height.strip('cm'))
    else:
        feet, inches = height.split("'")
        total_inches = int(feet) * 12 + int(inches.strip('"'))
        return round(total_inches * 2.54)

# Apply the function to the 'Height' column
fdc['Height'] = fdc['Height'].apply(convert_height)
fdc['Height'].unique()
```

Out[31]:

```
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, 201,
       158, 162, 161, 160, 203, 157, 156, 202, 159, 206, 155], dtype=int6
4)
```

3.10.1 Rename Column

Let´s change the name of the column for one more representative

In [32]:

```
# Rename the 'Height' column to express that the values are in cm
fdc = fdc.rename(columns={'Height':'Height(cm)'})
fdc['Height(cm)'].sample(3)
```

Out[32]:

```
14801    174
9579     180
17348    180
Name: Height(cm), dtype: int64
```

3.11 Weight

The player's weight is in kilograms. We do the same as done with the height. Removing the 'kg' and 'lbs', and also transforming the 'lbs' into kg.

In [33]:

```
# Let´s check the data type
fdc['Weight'].dtype
```

Out[33]:

```
dtype('O')
```

In [34]:

```python
# How the data is composed
fdc['Weight'].unique()
```

Out[34]:

```
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 [35]:

```python
# Let´s create a function to create our data in kg and int
def convert_weight(weight):
    if "kg" in weight:
        return int(weight.strip('kg'))
    else:
        pounds = int(weight.strip('lbs'))
        return round(pounds/2.205)

# Apply the function to the weight column
fdc['Weight'] = fdc['Weight'].apply(convert_weight)
fdc['Weight'].unique()
```

Out[35]:

```
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)
```

3.11.1 Rename Column

Let´s change the name of the column for one more representative

In [36]:

```python
# Rename the 'Weight' column
fdc = fdc.rename(columns={'Weight':'Weight(kg)'})
fdc['Weight(kg)'].sample(3)
```

Out[36]:

```
5504     68
18272    70
7078     81
Name: Weight(kg), dtype: int64
```

3.12 Preferred Foot

Checking the data from the 'Preferred Foot' column.

In [37]:

```python
fdc['Preferred Foot'].dtype
```

Out[37]:

```
dtype('O')
```

In [38]:

```
fdc['Preferred Foot'].unique()
```

Out[38]:

```
array(['Left', 'Right'], dtype=object)
```

In [39]:

```
fdc[['Preferred Foot']].head(10)
```

Out[39]:

	Preferred Foot
0	Left
1	Right
2	Right
3	Right
4	Right
5	Right
6	Left
7	Right
8	Right
9	Right

3.13 BOV

In [40]:

```
fdc['BOV'].dtype
```

Out[40]:

```
dtype('int64')
```

In [41]:

```
fdc['BOV'].unique()
```

Out[41]:

```
array([93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77,
       76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60,
       59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48], dtype=int64)
```

In [42]:

```
missing_values = fdc['BOV'].isnull().sum()
print("Number of missing values: ", missing_values)
```

```
Number of missing values:  0
```

3.14 Best Position

Check the values from the column

In [43]:

```
fdc['Best Position'].unique()
```

Out[43]:

```
array(['RW', 'ST', 'GK', 'CAM', 'LW', 'CB', 'CDM', 'CF', 'CM', 'RB', 'LB',
       'LM', 'RM', 'LWB', 'RWB'], dtype=object)
```

In [44]:

```
missing_values = fdc['Best Position'].isnull().sum()
print("Number of missing values: ", missing_values)
```

```
Number of missing values:  0
```

3.16 Joined

It´s time to verify and clean the values from the column

In [45]:

```
fdc['Joined'].dtype
```

Out[45]:

```
dtype('O')
```

In [46]:

```
fdc['Joined'].unique()
```

Out[46]:

```
array(['01-Jul-04', '10-Jul-18', '16-Jul-14', ..., '22-Sep-18',
       '28-Feb-15', '06-Mar-18'], dtype=object)
```

In [47]:

```
missing_values = fdc['Joined'].isnull().sum()
print("Number of missing values: ", missing_values)
```

```
Number of missing values:  0
```

Now that we know we don´t have any NULL values, let´s change the data type to datetime

In [48]:

```
fdc['Joined'] = pd.to_datetime(fdc['Joined'], dayfirst=True, format='%d
fdc['Joined'].sample(15)
```

Out[48]:

```
7652    2019-07-01
5197    2019-07-07
6777    2020-09-14
11794   2017-12-08
6256    2020-08-25
16396   2017-09-01
12845   2018-01-07
15884   2019-07-03
2631    2017-07-01
6790    2019-01-12
12664   2016-07-01
9871    2016-08-21
11394   2020-09-04
4683    2017-07-04
3250    2020-08-11
Name: Joined, dtype: datetime64[ns]
```

In [49]:

```python
print(fdc['Joined'].dt.strftime('%d-%b-%y'))
```

```
0        01-Jul-04
1        10-Jul-18
2        16-Jul-14
3        30-Aug-15
4        03-Aug-17
           ...
18974    13-Jul-18
18975    01-Aug-20
18976    08-Mar-19
18977    22-Sep-20
18978    29-Jul-19
Name: Joined, Length: 18979, dtype: object
```

3.17 Loan Date End

Date when the Loan ends (if the player is on loan)

In [50]:

```python
# Check the data type
fdc['Loan Date End'].dtype
```

Out[50]:

```
dtype('O')
```

In [51]:

```python
fdc['Loan Date End'].unique()
```

Out[51]:

```
array([nan, '30-Jun-21', '31-Dec-20', '30-Jan-21', '30-Jun-22',
       '31-May-21', '05-Jul-21', '31-Dec-21', '01-Jul-21', '01-Jan-21',
       '31-Aug-21', '31-Jan-21', '30-Dec-21', '23-Jun-21', '03-Jan-21',
       '27-Nov-21', '17-Jan-21', '30-Jun-23', '31-Jul-21', '22-Nov-20',
       '31-May-22', '30-Dec-20', '04-Jan-21', '30-Nov-20', '01-Aug-21'],
      dtype=object)
```

In [52]:

```
# Let´s compare the data with the one stored in column 'Contract Status
on_loan = fdc[fdc['Contract Status'] == 'On Loan']
on_loan[['Contract', 'Contract Status', 'Loan Date End']]
```

Out[52]:

	Contract	Contract Status	Loan Date End
205	Jun 30, 2021 On Loan	On Loan	30-Jun-21
248	Jun 30, 2021 On Loan	On Loan	30-Jun-21
254	Jun 30, 2021 On Loan	On Loan	30-Jun-21
302	Jun 30, 2021 On Loan	On Loan	30-Jun-21
306	Jun 30, 2021 On Loan	On Loan	30-Jun-21
...	...	...	...
18472	Aug 31, 2021 On Loan	On Loan	31-Aug-21
18571	Jun 30, 2021 On Loan	On Loan	30-Jun-21
18600	Dec 31, 2020 On Loan	On Loan	31-Dec-20
18622	Dec 31, 2020 On Loan	On Loan	31-Dec-20
18680	Dec 31, 2020 On Loan	On Loan	31-Dec-20

1013 rows × 3 columns

3.18 Value

Now is time to verify is the values are correct

In [53]:

```
fdc['Value'].dtype
```

Out[53]:

```
dtype('O')
```

In [54]:

```
fdc['Value'].unique()
```

Out[54]:

```
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)
```

3.18.1 Transform 'Value' Column

For the values, I am going to transform them into numbers, some are expressed in millions, and others in thousands. First, is necessary to extract the €, M, and K from the values, replace the dot with a comma, and change the values to millions. For future operations is necessary to have values type number.

In [55]:

```
# Convert the values to strings
fdc['Value'] = fdc['Value'].astype(str)

# The values are strings, so we replace the € currency
fdc['Value'] = fdc['Value'].str.replace('€','')
fdc['Value'].dtype
```

Out[55]:

```
dtype('O')
```

In [56]:

```python
fdc['Value'].unique()
```

Out[56]:

```
array(['103.5M', '63M', '120M', '129M', '132M', '111M', '120.5M', '102
M',
       '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', '77.5
M',
       '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', '47.5
M',
       '52.5M', '54.5M', '34.5M', '57.5M', '51.5M', '44.5M', '55M', '48
M',
       '60.5M', '63.5M', '61M', '29M', '58.5M', '55.5M', '42M', '40.5
M',
       '43M', '45.5M', '34M', '26.5M', '42.5M', '35.5M', '45M', '41.5
M',
       '40M', '11M', '13.5M', '29.5M', '27M', '15.5M', '38.5M', '52M',
```

Let´s remember that some contract values were defined as 'Free' or 'On Loan', which can give us an error when converting the values into data type 'int'. Let´s try to identify them.

In [57]:

```python
# Check for missing values or null
fdc['Value'].isna()
```

Out[57]:

```
0        False
1        False
2        False
3        False
4        False
         ...  
18974    False
18975    False
18976    False
18977    False
18978    False
Name: Value, Length: 18979, dtype: bool
```

In [58]:

```python
missing_values = fdc['Value'].isnull().sum()
print("Number of missing values: ", missing_values)
```

```
Number of missing values:  0
```

In [59]:

```
print(fdc[['Value','Surname']].head(15))
```

```
     Value                     Surname
0   103.5M                       Messi
1      63M   Ronaldo dos Santos Aveiro
2     120M                       Oblak
3     129M                   De Bruyne
4     132M         da Silva Santos Jr.
5     111M                 Lewandowski
6   120.5M                       Salah
7     102M               Ramses Becker
8   185.5M                      Mbappé
9     110M                  ter Stegen
10    113M                    van Dijk
11  120.5M                        Mané
12   90.5M  Henrique Venancio Casimiro
13     82M                    Courtois
14   17.5M                       Neuer
```

Now is the time to remove the 'M' and 'K' from the values and convert them into 'int' data type.

In [60]:

```
# Use a lambda function to represent the values as they are
fdc['Value'] = fdc['Value'].apply(lambda x: float(x[:-1]) * 1e6
                                  if x[-1] == 'M' and x[:-1] else float
                                  if x[-1] == 'K' and x[:-1] else float

# # Because our values are expressed as float numbers, let´s convert th
fdc['Value'] = fdc['Value'].astype(int)
fdc['Value'].head(10)
```

Out[60]:

```
0    103500000
1     63000000
2    120000000
3    129000000
4    132000000
5    111000000
6    120500000
7    102000000
8    185500000
9    110000000
Name: Value, dtype: int32
```

3.18.2 Change column name

It´s time to change the column name for one more representative

In [61]:

```
fdc = fdc.rename(columns={'Value':'Market Price(€)'})
fdc['Market Price(€)'].sample(3)
```

Out[61]:

```
1302     11500000
255      35500000
17039      375000
Name: Market Price(€), dtype: int32
```

3.19 Wage

Let's verify the player's salaries

In [62]:
```
fdc['Wage'].dtype
```

Out[62]:
```
dtype('O')
```

In [63]:
```
fdc['Wage'].unique()
```

Out[63]:
```
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 [64]:
```
missing_values = fdc['Wage'].isnull().sum()
print("Number of missing values: ", missing_values)
```

```
Number of missing values:  0
```

Time to correct the salaries values by removing the '€' from the values

In [65]:
```
# Replace the '€'
fdc['Wage'] = fdc['Wage'].str.replace('€', '')
print(fdc['Wage'])
```

```
0        560K
1        220K
2        125K
3        370K
4        270K
         ...
18974      1K
18975     500
18976     500
18977      2K
18978      1K
Name: Wage, Length: 18979, dtype: object
```

In [66]:

```
# Convert the values
fdc['Wage'] = fdc['Wage'].replace({'K': '*1e3'}, regex=True).map(pd.eva
print(fdc['Wage'])
```

```
0        560000
1        220000
2        125000
3        370000
4        270000
          ...
18974      1000
18975       500
18976       500
18977      2000
18978      1000
Name: Wage, Length: 18979, dtype: int32
```

3.20 Release Clause

Now it´s time to analyze the values from the column 'Release Clause'

In [67]:

```
fdc['Release Clause'].dtype
```

Out[67]:

```
dtype('O')
```

In [68]:

```
missing_values = fdc['Release Clause'].isnull().sum()
print("Number of missing values: ", missing_values)
```

```
Number of missing values:  0
```

In [69]:

```
fdc['Release Clause'].unique()
```

Out[69]:

```
array(['€138.4M', '€75.9M', '€159.4M', ..., '€59K', '€35K', '€64K'],
      dtype=object)
```

3.20.1 Transform the values

We are going to replace the '€' and transform the values with the 'M' or 'K'

In [70]:

```
# Convert the values to strings
fdc['Release Clause'] = fdc['Release Clause'].astype(str)

# The values are strings, so we replace the € currency
fdc['Release Clause'] = fdc['Release Clause'].str.replace('€','')

# Use a lambda function to represent the values as they are
fdc['Release Clause'] = fdc['Release Clause'].apply(lambda x: float(x[:
                                  if x[-1] == 'M' and x[:-1] else float
                                  if x[-1] == 'K' and x[:-1] else float

# # Because our values are expressed as float numbers, let´s convert th
fdc['Release Clause'] = fdc['Release Clause'].astype(int)
fdc['Release Clause'].head(10)
```

Out[70]:

```
0    138400000
1     75900000
2    159400000
3    161000000
4    166500000
5    132000000
6    144300000
7    120300000
8    203100000
9    147700000
Name: Release Clause, dtype: int32
```

In [71]:

```
# Change the name of the column
fdc = fdc.rename(columns={'Release Clause': 'Release Clause(€)'})
fdc['Release Clause(€)'].head()
```

Out[71]:

```
0    138400000
1     75900000
2    159400000
3    161000000
4    166500000
Name: Release Clause(€), dtype: int32
```

3.21 Stats

The following values in the columns all belong to stats; let's check their data types and make sure there are no missing values. The columns we are working with are:

'Attacking','Crossing', 'Finishing', 'Heading Accuracy', 'Short Passing', 'Volleys','Skill', 'Dribbling', 'Curve', 'FK Accuracy', 'Long Passing', 'Ball Control', 'Movement', 'Acceleration', 'Sprint Speed', 'Agility', 'Reactions', 'Balance', 'Power', 'Shot Power', 'Jumping', 'Stamina', 'Strength', 'Long Shots', 'Mentality', 'Aggression', 'Interceptions', 'Positioning', 'Vision', 'Penalties', 'Composure', 'Defending', 'Marking', 'Standing Tackle', 'Sliding Tackle', 'Goalkeeping', 'GK Diving', 'GK Handling', 'GK Kicking', 'GK Positioning','GK Reflexes', 'Total Stats', 'Base Stats'

3.21.1 Attacking

In [72]:

```
fdc['Attacking'].dtype
```

Out[72]:

```
dtype('int64')
```

In [73]:

```
missing_values = fdc['Attacking'].isna().sum()
print("Number of missing values: ", missing_values)
```

```
Number of missing values:  0
```

In [74]:

```
fdc['Attacking'].unique()
```

Out[74]:

```
array([429, 437,  95, 407, 408, 423, 392, 114, 118, 316, 410, 349,  86,
       119, 426, 374, 411, 360, 328, 383, 405, 123, 420, 224, 388, 397,
       425, 373, 365, 371, 311, 396, 345, 399, 400,  78, 280, 330, 403,
       379, 380,  94, 394, 419, 339, 293, 344, 390,  84, 359, 372, 377,
       346, 389, 386, 308, 277, 382, 368, 402, 292, 298, 366, 352, 363,
       322, 361,  91, 364, 341, 385, 355, 305, 321, 262,  93, 375, 387,
       356, 253, 285, 391, 353, 367,  90, 295, 378, 256, 338, 331,  69,
       105,  85, 358, 343, 319, 271, 113, 350, 406, 340, 393, 247, 334,
       351, 342, 302, 329, 354,  98, 301, 115, 384, 208,  72, 376,  92,
       258, 362,  74, 417,  99, 263,  88, 279, 101, 395, 100,  81,  87,
        55, 310,  82, 117, 409, 318, 323, 248, 315, 381, 348, 327, 309,
       130, 283, 336, 369, 106, 252, 320, 290, 370, 126, 251, 108, 335,
       297, 284,  80,  75, 357, 270,  97, 306, 337,  73, 286, 325, 326,
       324, 333, 103, 259, 273, 313, 296,  61, 312, 347, 401, 304, 278,
        83,  43, 314, 291, 264, 272, 317, 231, 250, 268,  54, 261, 255,
        70, 281, 265, 299, 287,  68, 294,  77, 219, 300, 269, 332, 289,
       288, 107, 282, 122, 244,  89, 112, 274, 276, 307, 229,  96, 109,
        76, 125, 102, 239, 227, 241, 257, 254, 228, 233, 124, 215, 246,
       110, 245, 214, 242, 266, 104,  66, 303, 260,  63, 230, 275,  50,
       238, 249, 111,  67, 240, 221, 237,  56, 235, 234, 243, 267, 232,
       203, 223,  64, 213, 222, 226, 225, 211, 207,  52, 173,  57, 217,
       236,  71, 204, 216, 199,  59, 189,  60, 194, 116, 205, 201, 193,
        65, 192, 209, 218, 128, 210,  79,  45, 206, 162, 220,  49, 197,
       202, 212,  58, 190, 181,  51,  62, 200, 198, 195, 191, 131, 185,
        42, 180, 182, 196, 188, 169, 187, 178,  53, 183, 184, 186, 165,
       172,  47, 171, 176, 159,  46, 179, 175, 167, 174, 161, 170, 177,
       164, 134, 168, 163, 166, 158, 150, 143,  48, 152, 160, 148, 151,
       157, 154, 141, 146, 147, 149, 156, 153, 138, 145, 142, 139, 155,
       144, 136, 137], dtype=int64)
```

3.21.2 Crossing

In [75]:

```
fdc['Crossing'].dtype
```

Out[75]:

```
dtype('int64')
```

In [76]:

```
missing_values = fdc['Crossing'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [77]:

```
fdc['Crossing'].unique()
```

Out[77]:

```
array([85, 84, 13, 94, 71, 79, 17, 78, 18, 53, 76, 58, 14, 15, 75, 66, 70,
       68, 91, 82, 20, 12, 30, 77, 88, 83, 93, 90, 87, 81, 73, 11, 54, 62,
       86, 80, 55, 42, 57, 65, 63, 64, 52, 40, 69, 47, 60,  9, 16, 44, 72,
       50, 56, 46, 89, 34, 45, 74, 49, 67, 24, 35, 36, 61, 19, 27, 25, 10,
       51, 38, 43, 59, 39, 48, 23,  8, 28, 92, 41, 29, 32, 22, 26, 37, 33,
       31, 21,  7,  6], dtype=int64)
```

3.21.3 Finishing

In [78]:

```
fdc['Finishing'].dtype
```

Out[78]:

```
dtype('int64')
```

In [79]:

```
missing_values = fdc['Finishing'].isnull().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [80]:

```
fdc['Finishing'].unique()
```

Out[80]:

```
array([95, 11, 82, 87, 94, 91, 13, 14, 52, 90, 64, 88, 65, 85, 66, 84, 10,
       22, 76, 81, 56, 79, 57, 45, 77, 63, 86, 80, 15, 33, 67, 12, 72, 92,
       93, 51, 46, 60, 75, 55, 73, 83, 50, 42, 39, 40,  9, 68, 48, 37, 70,
       78, 69,  8, 53, 89, 25, 62, 71, 74, 44, 26, 19, 32, 18, 61, 58, 30,
       54, 36, 29, 16, 38, 59, 27, 34, 47, 20, 31, 49, 43, 41, 28,  5,  7,
        6, 21, 17, 35, 23, 24,  4,  3], dtype=int64)
```

3.21.4 Heading Accuracy

In [81]:

```
fdc['Heading Accuracy'].dtype
```

Out[81]:

```
dtype('int64')
```

In [82]:

```
missing_values = fdc['Heading Accuracy'].isnull().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [83]:

```
fdc['Heading Accuracy'].unique()
```

Out[83]:

```
array([70, 90, 15, 55, 62, 85, 59, 19, 73, 11, 87, 84, 80, 13, 25, 91, 92,
       78, 46, 54, 72, 64, 14, 10, 61, 58, 83, 38, 69, 51, 67, 86, 75, 68,
       16, 81, 21, 79, 53, 65, 82, 12, 42, 48, 88, 66, 76, 74, 52, 23, 40,
       49, 60, 44, 20, 37, 71, 17, 45, 77, 50, 63, 43, 39, 57, 56, 47, 24,
       18, 31, 28, 35, 34, 41, 36, 93,  7, 30, 89,  8, 26, 33, 27, 32, 22,
       29,  9,  5,  6], dtype=int64)
```

3.21.5 Short Passing

In [84]:
```
fdc['Short Passing'].dtype
```

Out[84]:
```
dtype('int64')
```

In [85]:
```
missing_values = fdc['Short Passing'].isnull().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [86]:
```
fdc['Short Passing'].unique()
```

Out[86]:
```
array([91, 82, 43, 94, 87, 84, 45, 83, 61, 79, 85, 33, 55, 86, 57, 81, 42,
       74, 93, 88, 30, 65, 89, 77, 32, 50, 80, 78, 90, 69, 40, 92, 75, 73,
       34, 76, 35, 70, 37, 23, 44, 38, 48, 26, 60, 25, 46, 28, 24, 36, 51,
       17, 18, 39, 71, 67, 27, 72, 66, 20, 31, 68, 29, 11, 64, 62, 41, 63,
       19, 54, 16, 22, 49, 59, 56, 14, 58, 15, 21, 52, 53, 12, 47, 13,  8,
        7], dtype=int64)
```

3.21.6 Volleys

In [87]:
```
fdc['Volleys'].dtype
```

Out[87]:
```
dtype('int64')
```

In [88]:
```
missing_values = fdc['Volleys'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [89]:
```
fdc['Volleys'].unique()
```

Out[89]:
```
array([88, 86, 13, 82, 87, 89, 79, 20, 83, 14, 45, 75, 63, 12, 11, 69, 67,
       56, 18, 85, 62, 70, 32, 40, 47, 81, 44, 84, 78, 76, 90, 49, 42, 64,
       57, 60,  8, 72, 71, 59, 74, 80, 73, 37, 31, 38, 61, 10, 77, 68, 58,
       66, 30, 33, 65, 27, 51, 15, 16, 50, 43, 35, 24, 17, 34, 28,  9, 39,
       52, 46, 22, 19, 53, 55, 48, 54, 23,  5, 41, 25, 21, 36, 26, 29,  6,
        7,  4,  3], dtype=int64)
```

3.21.7 Skill

In [90]:
```
fdc['Skill'].dtype
```

Out[90]:
```
dtype('int64')
```

In [91]:
```
missing_values = fdc['Skill'].isnull().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [92]:
```
fdc['Skill'].unique()
```

Out[92]:
```
array([470, 414, 109, 441, 448, 407, 406, 138, 394, 144, 363, 391, 369,
       110, 160, 404, 381, 397, 387, 336, 400, 436, 157, 395, 100, 262,
       427, 432, 429, 380, 426, 411, 358, 351, 433, 365, 403,  98, 276,
       386, 383,  99, 413, 115, 341, 375, 143, 359, 309, 435, 330, 325,
       355,  96, 420, 412, 388, 319, 269, 399, 106, 402, 425, 297, 312,
       418, 372, 352, 439, 409, 349, 116, 371, 428, 104, 345, 430, 295,
       405, 440, 422, 252, 401, 417, 396, 233, 377, 251, 382, 368,  84,
       356, 342, 410, 271, 350,  83, 126, 103, 370, 362, 343, 328, 344,
       415, 378, 275, 416, 119, 127, 373, 384,  77, 393, 348, 317, 408,
       376, 300, 220,  89, 107, 334,  72, 390, 419, 305, 289, 398, 281,
       354, 102, 339, 385, 139, 292,  97, 421,  91, 105,  73, 335, 101,
       340, 337, 306, 113, 122, 123, 302, 364, 250, 347, 333, 323, 389,
       361, 322,  86, 367, 258, 392,  92,  90, 310, 331, 338, 121, 260,
        82, 245, 324, 346, 379, 299, 284, 320, 283, 108, 278, 286, 296,
       315, 274,  88, 114, 264, 288,  94, 326, 366, 117, 360, 424,  93,
       318, 124, 125, 327, 249,  75, 332, 303, 374, 239, 272, 357, 353,
       266, 321, 277, 268, 314, 294, 240,  95, 227, 112, 118, 263, 280,
       140, 282,  81, 329, 201,  87, 221, 257, 285, 316, 287, 307, 270,
       256, 313, 311, 228, 247, 254, 130,  80,  85, 232, 293, 298, 301,
       213, 168, 291, 216, 290, 308, 261, 171, 267, 242, 219, 248, 237,
       243, 279, 246, 273,  78, 255, 253, 230,  74, 210, 235, 231, 208,
       259, 304, 241, 199, 224, 206,  61, 129, 222, 223, 141, 149, 131,
       225,  71, 189, 265, 226,  70, 179, 192, 134, 209, 173, 234,  76,
       236, 212,  69, 218, 120, 177, 238, 204, 229, 215, 165, 211, 195,
        64, 202, 194, 190, 193, 203,  67, 214,  79, 205, 244, 196, 111,
       187,  65, 200,  63, 198, 217, 135,  68, 184, 167, 148, 207, 142,
       185, 133, 191, 181, 197,  43,  66, 175, 182,  51, 180, 169, 186,
       137, 188, 176, 132,  60, 178, 147, 163, 183, 162, 152, 170, 172,
       174, 159, 161, 154, 153, 128,  62, 166,  53, 155,  56, 151, 164,
       158,  46, 150,  59,  55,  58, 156, 146,  52, 136,  54,  47,  48,
       145,  40,  57], dtype=int64)
```

3.21.8 Dribbling

In [93]:
```
fdc['Dribbling'].dtype
```

Out[93]:
```
dtype('int64')
```

In [94]:
```
missing_values = fdc['Dribbling'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [95]:
```
fdc['Dribbling'].unique()
```

Out[95]:
```
array([96, 88, 12, 95, 85, 90, 27, 92, 21, 70, 91, 69, 13, 30, 87, 65, 79,
       83, 23, 80, 18, 93, 77, 63, 76, 16, 59, 81, 11, 84, 10, 75, 78, 55,
       15, 86, 66, 67, 28, 57, 64, 82, 62, 19, 53, 72, 50, 26, 43, 89, 73,
       20, 14, 68, 71, 74, 22, 54, 56, 61,  9, 24, 60, 25,  8, 17, 47, 58,
       46, 42, 51, 52, 49, 44, 35, 48, 39, 29, 40, 45, 34, 31, 33, 38, 41,
       32,  7, 37, 36,  5,  6], dtype=int64)
```

3.21.9 Curve

```
In [96]: fdc['Curve'].dtype

Out[96]: dtype('int64')
```

```
In [97]: missing_values = fdc['Curve'].isna().sum()
         print('Number of missing values: ', missing_values)

Number of missing values:  0
```

```
In [98]: fdc['Curve'].unique()

Out[98]: array([93, 81, 13, 85, 88, 79, 83, 19, 18, 60, 76, 63, 14, 74, 77, 49, 15,
                80, 12, 28, 86, 84, 82, 61, 71, 11, 66, 16, 89, 70, 21, 46, 78, 67,
                58, 65, 48, 34, 90, 59, 55, 87, 62,  9, 56, 36, 30, 32, 73, 69, 68,
                75, 45, 10, 72, 64, 41, 23, 47, 20, 51, 25, 44, 17, 54, 57, 53, 33,
                40, 50, 39, 35, 52, 42, 37, 43, 26, 31, 92, 91, 29, 94, 27, 38, 22,
                24,  8,  6,  7,  5,  4], dtype=int64)
```

3.21.10 FK Accuracy

```
In [99]: fdc['FK Accuracy'].dtype

Out[99]: dtype('int64')
```

```
In [100]: missing_values = fdc['FK Accuracy'].isna().sum()
          print('Number of missing values: ', missing_values)

Number of missing values:  0
```

```
In [101]: fdc['FK Accuracy'].unique()

Out[101]: array([94, 76, 14, 83, 89, 85, 69, 18, 63, 12, 70, 64, 74, 20, 11, 73, 49,
                 61, 88, 68, 28, 79, 84, 48, 67, 38, 87, 53, 65, 15, 31, 78, 82, 10,
                 51, 59, 19, 47, 52, 57, 43, 13, 77, 54, 75, 86, 55, 30, 62, 32, 58,
                 93,  8, 66, 71, 81, 92, 44, 17, 60, 40, 16, 72, 46, 35, 45, 29, 21,
                 56, 80, 24, 22, 39, 42, 26, 41,  9, 37, 27, 50, 33, 25, 36, 91, 34,
                 23,  7,  6, 90,  5], dtype=int64)
```

3.21.11Long Passing

```
In [102]: fdc['Long Passing'].dtype

Out[102]: dtype('int64')
```

```
In [103]: missing_values = fdc['Long Passing'].isna().sum()
          print('Number of missing values: ', missing_values)

Number of missing values:  0
```

In [104]:

```
fdc['Long Passing'].unique()
```

Out[104]:

```
array([91, 77, 40, 93, 81, 70, 75, 44, 63, 86, 71, 84, 35, 59, 73, 83, 64,
       69, 79, 82, 68, 89, 76, 80, 87, 37, 65, 36, 50, 53, 78, 47, 74, 48,
       31, 85, 24, 55, 90, 54, 62, 32, 49, 66, 67, 51, 28, 46, 52, 72, 56,
       41, 45, 22, 88, 61, 33, 12, 60, 17, 27, 29, 23, 38, 16, 58, 34, 25,
       39, 21, 30, 42, 43, 57, 20, 26, 18, 19, 13, 15, 11, 14,  9, 10,  5,
        8], dtype=int64)
```

3.21.12 Ball Control

In [105]:

```
fdc['Ball Control'].dtype
```

Out[105]:

```
dtype('int64')
```

In [106]:

```
missing_values = fdc['Ball Control'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [107]:

```
fdc['Ball Control'].unique()
```

Out[107]:

```
array([96, 92, 30, 95, 88, 89, 90, 77, 79, 23, 46, 83, 80, 85, 94, 40, 84,
       16, 74, 91, 87, 82, 78, 19, 61, 22, 34, 38, 81, 25, 86, 76, 69, 28,
       93, 75, 35, 60, 63, 73, 18, 71, 15, 21, 72, 14, 65, 20, 24, 27, 70,
       33, 17, 62, 64,  9, 68, 67, 32, 26, 66, 52, 11, 57, 58, 29, 12, 37,
       10, 36, 13, 31, 55, 59, 39, 54, 56, 48, 44, 51, 50, 47, 49, 53,  5,
       42,  8, 45, 43, 41,  7], dtype=int64)
```

3.21.13 Movement

In [108]:

```
fdc['Movement'].dtype
```

Out[108]:

```
dtype('int64')
```

In [109]:

```
missing_values = fdc['Movement'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [110]:
```
fdc['Movement'].unique()
```

Out[110]:
```
array([451, 431, 307, 398, 453, 407, 460, 268, 458, 254, 354, 343, 284,
       286, 388, 378, 424, 464, 420, 399, 437, 322, 367, 272, 328, 448,
       332, 425, 435, 391, 434, 400, 331, 349, 429, 416, 312, 326, 418,
       419, 417, 386, 321, 409, 374, 304, 403, 351, 401, 365, 414, 292,
       323, 299, 433, 350, 348, 413, 320, 281, 427, 353, 364, 410, 428,
       316, 381, 442, 375, 288, 395, 385, 251, 319, 444, 383, 298, 411,
       412, 415, 393, 397, 443, 423, 387, 422, 327, 390, 362, 352, 406,
       277, 361, 421, 396, 384, 450, 338, 363, 359, 287, 297, 430, 382,
       377, 380, 438, 449, 257, 371, 339, 341, 404, 345, 394, 295, 246,
       265, 258, 366, 294, 314, 266, 405, 218, 337, 267, 220, 376, 309,
       283, 426, 347, 244, 240, 291, 340, 250, 305, 290, 317, 334, 355,
       333, 389, 330, 318, 441, 402, 344, 335, 219, 264, 408, 274, 373,
       379, 256, 229, 392, 372, 360, 262, 346, 278, 248, 368, 279, 269,
       336, 342, 236, 370, 243, 315, 249, 227, 329, 239, 369, 223, 282,
       358, 271, 313, 270, 356, 263, 184, 311, 436, 432, 221, 301, 190,
       259, 308, 235, 260, 217, 275, 285, 210, 234, 276, 310, 447, 180,
       446, 300, 303, 209, 247, 252, 231, 357, 226, 238, 280, 440, 237,
       245, 296, 325, 273, 306, 196, 242, 199, 178, 222, 445, 324, 293,
       302, 289, 214, 192, 206, 225, 197, 241, 230, 188, 202, 208, 203,
       216, 213, 224, 439, 212, 232, 253, 228, 189, 204, 205, 207, 198,
       168, 255, 215, 194, 191, 185, 145, 261, 156, 201, 193, 181, 233,
       195, 183, 152, 211, 160, 173, 170, 176, 147, 143, 159, 187, 169,
       200, 165, 163, 177, 179, 167, 139, 162, 175, 155, 166, 172, 174,
       154, 164, 182, 150, 186, 146, 138, 157, 137, 135, 171, 158, 161,
       149, 124, 144, 151, 148, 141, 134, 153, 126, 142, 125, 132, 127,
       140, 133, 130, 131, 136, 122], dtype=int64)
```

3.21.14 Acceleration

In [111]:
```
fdc['Acceleration'].dtype
```

Out[111]:
```
dtype('int64')
```

In [112]:
```
missing_values = fdc['Acceleration'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [113]:
```
fdc['Acceleration'].unique()
```

Out[113]:
```
array([91, 87, 43, 77, 94, 56, 96, 38, 72, 95, 60, 42, 54, 79, 89, 64, 66,
       51, 73, 57, 80, 86, 85, 78, 40, 82, 76, 65, 68, 90, 48, 46, 88, 70,
       83, 84, 93, 52, 74, 92, 55, 58, 59, 67, 81, 62, 44, 71, 69, 50, 53,
       45, 49, 75, 41, 61, 63, 35, 47, 34, 36, 37, 39, 30, 97, 31, 33, 32,
       27, 28, 26, 29, 25, 17, 19, 24, 15, 23, 21, 20, 22, 16, 18, 13, 1
4],
      dtype=int64)
```

3.21.15 Sprint Speed

In [114]:
```
fdc['Sprint Speed'].dtype
```

Out[114]:
```
dtype('int64')
```

In [115]:

```
missing_values = fdc['Sprint Speed'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [116]:

```
fdc['Sprint Speed'].unique()
```

Out[116]:

```
array([80, 91, 60, 76, 89, 78, 92, 47, 96, 50, 79, 93, 69, 52, 72, 70, 90,
       66, 82, 63, 55, 77, 86, 81, 83, 85, 65, 68, 53, 43, 94, 62, 58, 61,
       87, 64, 67, 54, 88, 75, 95, 73, 49, 84, 56, 44, 74, 51, 57, 46, 59,
       71, 37, 34, 33, 42, 30, 35, 48, 39, 45, 40, 18, 38, 41, 27, 32, 29,
       28, 36, 26, 31, 22, 25, 23, 15, 20, 17, 16, 24, 19, 21, 12, 14],
      dtype=int64)
```

3.21.16 Agility

In [117]:

```
fdc['Agility'].dtype
```

Out[117]:

```
dtype('int64')
```

In [118]:

```
missing_values = fdc['Agility'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [119]:

```
fdc['Agility'].unique()
```

Out[119]:

```
array([91, 87, 67, 78, 96, 77, 40, 92, 37, 61, 93, 51, 79, 84, 94, 82, 60,
       69, 47, 52, 63, 74, 59, 66, 86, 85, 57, 55, 76, 75, 73, 62, 72, 90,
       68, 64, 80, 56, 48, 83, 41, 81, 54, 88, 33, 65, 49, 71, 89, 45, 70,
       43, 50, 32, 42, 39, 58, 36, 34, 53, 46, 95, 44, 38, 21, 29, 35, 31,
       19, 26, 30, 22, 28, 24, 25, 23, 27, 14, 18, 15, 20], dtype=int64)
```

3.21.17 Reactions

In [120]:

```
fdc['Reactions'].dtype
```

Out[120]:

```
dtype('int64')
```

In [121]:

```
missing_values = fdc['Reactions'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [122]:

```
fdc['Reactions'].unique()
```

Out[122]:

```
array([94, 95, 88, 91, 93, 92, 86, 89, 87, 84, 90, 83, 85, 82, 81, 79, 80,
       74, 75, 78, 77, 73, 76, 71, 70, 68, 72, 66, 69, 65, 67, 64, 59, 60,
       62, 63, 61, 58, 57, 56, 50, 54, 53, 55, 52, 32, 49, 48, 45, 51, 46,
       47, 37, 34, 44, 40, 38, 43, 41, 35, 42, 33, 39, 31, 36, 30, 24, 29,
       28], dtype=int64)
```

3.21.18 Balance

In [123]:

```
fdc['Balance'].dtype
```

Out[123]:

```
dtype('int64')
```

In [124]:

```
missing_values = fdc['Balance'].isnull().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [125]:

```
fdc['Balance'].unique()
```

Out[125]:

```
array([95, 71, 49, 76, 83, 82, 91, 37, 43, 53, 86, 66, 45, 35, 69, 94, 92,
       84, 90, 48, 73, 36, 41, 93, 74, 60, 79, 65, 78, 61, 57, 50, 68, 51,
       54, 77, 81, 39, 75, 58, 87, 85, 63, 38, 88, 67, 72, 62, 80, 44, 46,
       42, 55, 40, 70, 32, 89, 52, 59, 47, 64, 27, 56, 30, 31, 25, 34, 29,
       24, 96, 33, 28, 20, 23, 22, 26, 21, 17, 97, 19, 12, 18],
      dtype=int64)
```

3.21.19 Power

In [126]:

```
fdc['Power'].dtype
```

Out[126]:

```
dtype('int64')
```

In [127]:

```
missing_values = fdc['Power'].isnull().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [128]:

```
fdc['Power'].unique()
```

Out[128]:

```
array([389, 444, 268, 408, 357, 420, 393, 240, 404, 402, 406, 437, 249,
       284, 400, 403, 358, 381, 382, 273, 424, 264, 316, 361, 355, 328,
       370, 350, 365, 348, 411, 395, 385, 257, 337, 250, 379, 371, 409,
       223, 398, 388, 241, 347, 308, 426, 378, 343, 341, 262, 325, 345,
       359, 399, 421, 396, 315, 253, 368, 336, 340, 366, 387, 369, 375,
       260, 326, 346, 373, 412, 364, 279, 376, 372, 415, 356, 333, 338,
       342, 410, 407, 430, 394, 354, 331, 239, 234, 392, 270, 422, 374,
       360, 391, 300, 335, 242, 327, 215, 397, 321, 390, 339, 383, 265,
       288, 224, 351, 252, 429, 416, 380, 413, 377, 405, 349, 232, 386,
       362, 192, 320, 251, 329, 271, 237, 427, 259, 255, 266, 227, 353,
       258, 243, 263, 291, 302, 306, 332, 363, 256, 247, 301, 287, 322,
       419, 312, 245, 297, 401, 344, 235, 289, 233, 317, 334, 216, 367,
       352, 318, 226, 324, 219, 319, 292, 244, 423, 323, 304, 208, 314,
       313, 193, 299, 303, 311, 229, 211, 225, 309, 330, 238, 305, 220,
       296, 212, 231, 283, 207, 198, 281, 384, 307, 272, 298, 248, 310,
       267, 214, 282, 274, 280, 230, 228, 221, 277, 276, 285, 290, 269,
       246, 294, 293, 195, 236, 295, 217, 189, 275, 201, 278, 194, 206,
       218, 176, 205, 185, 196, 222, 204, 188, 197, 209, 286, 168, 254,
       200, 183, 179, 159, 180, 187, 164, 178, 190, 213, 202, 186, 191,
       261, 210, 203, 173, 199, 169, 152, 181, 175, 184, 182, 170, 160,
       162, 167, 177, 139, 161, 172, 165, 171, 128, 174, 158, 153, 166,
       155, 163, 151, 122, 142, 143, 156, 149, 144, 157, 147, 154, 150,
       134, 140], dtype=int64)
```

3.21.20 Shot Power

```
In [129]: fdc['Shot Power'].dtype
```

```
Out[129]: dtype('int64')
```

```
In [130]: missing_values = fdc['Shot Power'].isnull().sum()
          print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

```
In [131]: fdc['Shot Power'].unique()
```

```
Out[131]: array([86, 94, 59, 91, 80, 89, 64, 66, 81, 84, 88, 56, 68, 79, 78, 71, 82,
                 70, 55, 76, 61, 83, 51, 52, 90, 87, 62, 72, 77, 74, 50, 57, 58, 85,
                 60, 75, 67, 65, 93, 46, 54, 69, 41, 73, 40, 53, 95, 43, 63, 42, 48,
                 31, 44, 37, 49, 39, 45, 38, 47, 30, 33, 25, 34, 36, 28, 27, 32, 26,
                 35, 23, 22, 29, 20, 24, 21, 18], dtype=int64)
```

3.21.21 Jumping

```
In [132]: fdc['Jumping'].dtype
```

```
Out[132]: dtype('int64')
```

```
In [133]: missing_values = fdc['Jumping'].isnull().sum()
          print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

```
In [134]: fdc['Jumping'].unique()
```

```
Out[134]: array([68, 95, 78, 63, 62, 84, 69, 52, 77, 79, 90, 86, 87, 93, 57, 75, 66,
                 82, 56, 32, 51, 76, 72, 81, 74, 71, 67, 65, 73, 64, 70, 80, 85, 37,
                 89, 60, 49, 50, 83, 58, 53, 59, 88, 38, 92, 34, 61, 46, 43, 36, 91,
                 39, 45, 42, 40, 54, 33, 55, 31, 44, 35, 47, 48, 30, 41, 94, 28, 29,
                 27, 24, 19, 26, 17, 15, 22], dtype=int64)
```

3.21.22 Stamina

```
In [135]: fdc['Stamina'].dtype
```

```
Out[135]: dtype('int64')
```

```
In [136]: missing_values = fdc['Stamina'].isnull().sum()
          print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

```
In [137]: fdc['Stamina'].unique()
```

```
Out[137]: array([72, 84, 41, 89, 81, 76, 85, 32, 86, 35, 75, 88, 90, 38, 43, 78, 79,
                 96, 95, 70, 82, 77, 93, 94, 87, 39, 54, 80, 45, 83, 69, 65, 73, 91,
                 34, 66, 71, 92, 62, 67, 64, 63, 68, 36, 61, 74, 42, 40, 23, 44, 31,
                 57, 20, 37, 29, 30, 56, 60, 52, 48, 58, 25, 51, 26, 27, 59, 28, 53,
                 33, 49, 97, 55, 50, 46, 24, 21, 22, 15, 47, 17, 19, 16, 18, 14, 1
          2],
                dtype=int64)
```

3.21.23 Strength

```
In [138]: fdc['Strength'].dtype
```

```
Out[138]: dtype('int64')
```

```
In [139]: missing_values = fdc['Strength'].isnull().sum()
          print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

```
In [140]: fdc['Strength'].unique()
```

```
Out[140]: array([69, 78, 74, 50, 86, 75, 76, 92, 70, 91, 80, 85, 65, 72, 67, 60, 84,
                 71, 94, 63, 73, 62, 54, 81, 64, 87, 58, 43, 77, 66, 53, 89, 68, 46,
                 44, 61, 79, 88, 59, 83, 55, 34, 82, 95, 56, 37, 90, 57, 93, 49, 39,
                 51, 52, 40, 48, 41, 47, 35, 42, 33, 45, 32, 38, 30, 31, 36, 29, 27,
                 24, 28, 16, 97, 96, 20, 25, 26, 23], dtype=int64)
```

3.21.24 Long Shots

```
In [141]: fdc['Long Shots'].dtype
```

```
Out[141]: dtype('int64')
```

```
In [142]: missing_values = fdc['Long Shots'].isnull().sum()
          print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

```
In [143]: fdc['Long Shots'].unique()
```

```
Out[143]: array([94, 93, 12, 91, 84, 85, 14, 79, 10, 64, 78, 81, 17, 16, 65, 87, 18,
                 86, 19, 15, 82, 63, 74, 76, 47, 89, 70, 90, 77, 13, 49, 54, 88, 80,
                 53, 58, 51, 73, 66, 75, 83, 30, 46, 35, 71, 61, 72, 69, 43, 48, 62,
                 41, 60, 11, 26, 57, 59, 68, 67,  7, 27, 56, 20, 52, 92, 50, 22, 40,
                 39, 44, 31, 42,  9,  6, 55, 28, 23, 38, 24, 25, 34, 36, 29,  4,  8,
                 45, 33, 37, 21, 32,  5], dtype=int64)
```

3.21.25 Mentality

In [144]:

```
fdc['Mentality'].dtype
```

Out[144]:

```
dtype('int64')
```

In [145]:

```
missing_values = fdc['Mentality'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [146]:

```
fdc['Mentality'].unique()
```

Out[146]:

```
array([347, 353, 140, 408, 356, 391, 376, 341, 171, 358, 396, 122, 188,
       363, 414, 332, 386, 379, 348, 172, 382, 123, 294, 378, 313, 371,
       331, 412, 345, 377, 161, 306, 387, 339, 135, 360, 138, 369, 359,
       170, 361, 321, 397, 394, 385, 366, 162, 337, 362, 344, 319, 315,
       144, 336, 340, 373, 398, 324, 300, 338, 384, 139, 364, 372, 134,
       354, 342, 308, 322, 383, 263, 149, 304, 367, 357, 390, 291, 279,
       310, 388, 375, 349, 351, 365, 133, 334, 303, 380, 153, 392, 169,
       318, 350, 352, 401, 302, 325, 346, 132, 399, 281, 335, 403, 307,
       368, 141, 126, 328, 245, 131, 320, 127, 421, 400, 137, 374, 305,
        92, 316, 311, 120, 389, 145, 355, 148, 343, 142, 130, 121, 157,
       329, 323, 115, 150, 298, 154, 317, 295, 100, 301, 326, 327, 197,
       273, 287, 370, 290, 103, 393, 312, 297,  89, 271, 299, 124, 333,
       258, 309, 158, 272, 118, 314, 330, 292, 404, 101, 280, 277, 296,
       248, 285, 278, 109,  93, 146, 286, 284, 288, 105, 152, 111, 160,
       119, 156,  95,  99, 238, 104, 266, 276, 275, 265, 106, 254, 293,
       282, 168, 260, 136, 102, 267, 113, 289,  96, 270, 176, 164, 128,
       268, 283, 244, 182, 243, 240, 116, 264, 112, 274, 261, 114, 269,
       110, 257, 179, 155, 252, 262, 151, 247, 108, 256, 117, 249, 253,
       231, 159, 163,  84, 251,  97,  91,  75, 147, 129, 230, 242, 250,
       259, 125, 381,  77, 175,  82,  88,  90, 165,  83, 195,  87, 246,
       255,  85,  94, 226, 216, 236, 220, 107, 241, 228, 198, 239, 225,
       181, 233, 219, 166, 183,  98, 237, 235,  86, 229, 217, 143, 232,
       209, 234, 224, 206, 227, 222,  80,  78, 186, 221, 173, 214, 187,
        79,  68, 167,  81, 218, 212, 199, 210,  74, 223, 208, 213, 201,
        72, 215, 202, 205, 203, 204, 190,  76, 211, 207, 192,  70, 194,
       196, 189,  66, 193, 200,  67, 191, 184,  71,  64,  65,  69, 177,
        63,  73,  51,  58, 180, 185, 174,  60,  55, 178,  62,  50,  59],
      dtype=int64)
```

3.21.26 Aggression

In [147]:

```
fdc['Aggression'].dtype
```

Out[147]:

```
dtype('int64')
```

In [148]:

```
missing_values = fdc['Aggression'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [149]:

```
fdc['Aggression'].unique()
```

Out[149]:

```
array([44, 63, 34, 76, 51, 81, 27, 62, 43, 83, 75, 91, 23, 29, 90, 65, 59,
       89, 48, 38, 25, 87, 54, 60, 73, 74, 69, 85, 70, 86, 32, 40, 31, 77,
       84, 80, 78, 79, 71, 56, 42, 30, 61, 58, 28, 82, 46, 52, 36, 92, 55,
       35, 67, 37, 72, 57, 50, 64, 39, 47, 20, 68, 15, 66, 33, 93, 88, 22,
       24, 45, 17, 18, 26, 21, 11, 41, 53, 19, 12, 49, 94, 16, 95, 13, 14,
       96, 10,  9], dtype=int64)
```

3.21.27 Interceptions

In [150]:

```
fdc['Interceptions'].dtype
```

Out[150]:

```
dtype('int64')
```

In [151]:

```
missing_values = fdc['Interceptions'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [152]:

```
fdc['Interceptions'].unique()
```

Out[152]:

```
array([40, 29, 19, 66, 36, 49, 55, 11, 38, 22, 90, 35, 87, 15, 30, 39, 88,
       24, 91, 82, 42, 27, 41, 79, 74, 58, 20, 85, 48, 83, 64, 21, 50, 81,
       78, 28, 86, 26, 34, 52, 37, 80, 25, 56, 23, 47, 45, 77, 84, 44, 53,
       18, 46, 72, 61, 89, 54, 63, 65, 73, 16, 32, 76, 59, 13, 70, 31, 69,
       33, 17, 75, 68, 60, 51, 71, 12, 57, 10, 43, 67, 14,  9, 62,  8,  7,
        6,  4,  5,  3], dtype=int64)
```

3.21.28 Positioning

In [153]:

```
fdc['Positioning'].dtype
```

Out[153]:

```
dtype('int64')
```

In [154]:

```
missing_values = fdc['Positioning'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [155]:

```
fdc['Positioning'].unique()
```

Out[155]:

```
array([93, 95, 11, 88, 87, 94, 91, 13, 47, 92, 72, 12, 90, 73, 80, 85, 20,
       35, 76, 89, 83, 77, 54, 70, 86, 16, 28, 14, 84, 78, 10, 75, 52, 71,
       81, 64, 56, 15, 82, 79, 44, 30, 59,  7, 68, 38, 48, 67, 24, 26, 34,
       69, 74, 32, 66, 62, 65, 51, 18, 31,  9, 25, 49, 55, 63, 27, 61, 17,
       39, 58, 29, 50, 40, 19,  8, 42, 60, 57, 37, 45, 43, 53,  5,  4, 36,
        6, 46, 41, 23, 22, 33, 21,  3,  2], dtype=int64)
```

3.21.29 Vision

In [156]:
```
fdc['Vision'].dtype
```

Out[156]:
```
dtype('int64')
```

In [157]:
```
missing_values = fdc['Vision'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [158]:
```
fdc['Vision'].unique()
```

Out[158]:
```
array([95, 82, 65, 94, 90, 79, 84, 66, 80, 70, 85, 44, 87, 71, 83, 41, 52,
       86, 68, 50, 77, 48, 88, 30, 61, 74, 59, 73, 72, 64, 91, 78, 63, 57,
       89, 62, 56, 69, 42, 67, 27, 76, 81, 55, 75, 60, 49, 45, 58, 22, 53,
       46, 25, 43, 51, 40, 93, 33, 31, 34, 35, 39, 47, 21, 32, 28, 37, 36,
       38, 54, 24, 23, 14, 11, 15, 26, 19, 18, 12, 20, 17, 10, 29, 13, 16,
        9], dtype=int64)
```

3.21.30 Penalties

In [159]:
```
fdc['Penalties'].dtype
```

Out[159]:
```
dtype('int64')
```

In [160]:
```
missing_values = fdc['Penalties'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [161]:
```
fdc['Penalties'].unique()
```

Out[161]:
```
array([75, 84, 11, 92, 88, 83, 23, 70, 25, 62, 71, 66, 27, 47, 69, 54, 44,
       86, 17, 90, 33, 87, 73, 60, 55, 68, 91, 72, 50, 78, 18, 82, 40, 29,
       45, 43, 64, 24, 59, 46, 56, 81, 67, 49, 61, 74, 58, 63, 79, 38, 80,
       32, 20, 76, 77, 41, 19, 26, 85, 21, 52, 34, 53, 65, 57, 16, 42, 89,
       15, 13, 14, 22, 51, 37,  9, 48, 12, 31, 36, 39, 10, 30, 35, 28,  8,
        7,  6], dtype=int64)
```

3.21.31 Composure

In [162]:
```
fdc['Composure'].dtype
```

Out[162]:
```
dtype('int64')
```

In [163]:
```
missing_values = fdc['Composure'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [164]:
```
fdc['Composure'].unique()
```

Out[164]:
```
array([96, 95, 68, 91, 93, 88, 90, 65, 84, 70, 66, 80, 85, 69, 82, 89, 81,
       87, 83, 86, 67, 92, 94, 57, 78, 79, 75, 45, 61, 76, 58, 62, 77, 74,
       59, 55, 48, 40, 64, 73, 39, 71, 72, 63, 60, 52, 53, 56, 44, 54, 41,
       32, 49, 46, 31, 51, 50, 25, 18, 38, 30, 24, 21, 36, 33, 26, 23, 47,
       22, 28, 34, 35, 37, 43, 27, 12, 42, 17, 29, 13, 19, 14, 16, 20, 1
5],
      dtype=int64)
```

3.21.32 Defending

In [165]:
```
fdc['Defending'].dtype
```

Out[165]:
```
dtype('int64')
```

In [166]:
```
missing_values = fdc['Defending'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [167]:
```
fdc['Defending'].unique()
```

Out[167]:
```
array([ 91,  84,  57, 186,  94,  96, 122,  50, 100,  48, 272, 259,  54,
        38,  89, 263,  83, 147, 264, 245, 120,  52, 130, 267, 205, 162,
       105, 241, 148, 248, 266, 194, 258, 117, 166,  56, 249,  92,  45,
       214, 140,  99, 150,  59, 251, 262, 243, 195, 160,  40, 114, 236,
       244, 231,  80, 123, 253, 132, 103, 257, 261,  98,  78, 209, 229,
       230,  60, 101, 206, 242, 138,  61, 256, 171, 260, 226, 224,  44,
       131, 113, 240,  77, 232, 225, 109, 228, 247,  93, 121, 238, 111,
       128, 188, 173, 250, 255,  41, 144, 239, 217, 106, 165, 246, 235,
       126, 118, 203, 234, 135, 215, 175, 192, 108,  39,  33, 151, 156,
       174,  47, 216, 237, 102, 227, 161, 233,  67, 213,  75, 212,  36,
       254, 196,  88,  81, 134,  53, 155, 223,  43, 125,  46,  51, 137,
        71,  95,  35, 208, 110, 170,  87, 107,  55, 204, 177,  69, 152,
       163,  37, 181, 252, 159, 133, 124, 207,  82,  97,  65,  42,  79,
       104, 211, 129,  49, 157, 153, 185, 189, 146,  86, 112,  73, 127,
        31, 220, 164, 191, 219, 139,  64, 183,  66, 197,  90, 218,  34,
        72, 221, 222, 142,  63, 136, 179,  85, 169, 180,  74, 210,  62,
       187, 145, 198, 184, 199,  32,  30,  58, 172, 178, 116, 176,  70,
       202, 141, 115, 193, 149,  29, 201, 167, 168, 182, 119, 190, 200,
        76, 143, 158, 154,  68,  28,  27,  25,  24,  26,  23,  20,  21],
      dtype=int64)
```

3.21.33 Marking

In [168]:
```
fdc['Marking'].dtype
```

Out[168]:
```
dtype('int64')
```

In [169]:
```
missing_values = fdc['Marking'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

```
In [170]: fdc['Marking'].unique()
```

```
Out[170]: array([32, 28, 27, 68, 35, 38, 15, 34, 25, 93, 42, 84, 20, 17, 47, 85, 30,
                 89, 82, 29, 56, 91, 72, 59, 79, 49, 83, 86, 50, 60, 94, 41, 57, 78,
                 63, 88, 90,  9, 58, 74, 39, 92, 45, 36, 44, 87, 70, 76, 53, 80, 67,
                 77, 12, 48, 55, 75, 81, 11, 64, 69, 14, 24, 52, 65, 19, 31, 13, 10,
                 66, 71, 54, 46, 22, 40, 18, 51, 37, 43, 61, 26, 73, 21,  7, 33, 62,
                 16, 23,  8,  6,  5,  4,  3], dtype=int64)
```

3.21.34 Standing Tackle

```
In [171]: fdc['Standing Tackle'].dtype
```

```
Out[171]: dtype('int64')
```

```
In [172]: missing_values = fdc['Standing Tackle'].isnull().sum()
          print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

```
In [173]: fdc['Standing Tackle'].unique()
```

```
Out[173]: array([35, 32, 12, 65, 30, 42, 43, 19, 34, 13, 93, 88, 18, 10, 24, 29, 53,
                 90, 84, 48, 15, 36, 89, 27, 73, 54, 41, 83, 59, 67, 87, 64, 14, 55,
                 75, 45, 33, 57, 21, 82, 50, 86, 80, 79, 31, 46, 85, 40, 44, 56, 20,
                 70, 76, 81, 71, 16, 68, 37, 38, 78, 39, 77, 11, 74, 28, 49, 47, 72,
                 61, 51, 22, 17, 52, 63, 23, 60, 25, 26,  9, 62, 58, 66, 69,  7,  8,
                  6,  5], dtype=int64)
```

3.21.35 Sliding Tackle

```
In [174]: fdc['Sliding Tackle'].dtype
```

```
Out[174]: dtype('int64')
```

```
In [175]: missing_values = fdc['Sliding Tackle'].isnull().sum()
          print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

```
In [176]: fdc['Sliding Tackle'].unique()
```

```
Out[176]: array([24, 18, 53, 29, 19, 41, 16, 32, 10, 86, 38, 87, 11, 90, 47, 85, 79,
                 40,  8, 13, 22, 60, 49, 81, 88, 55, 33, 42, 14, 80, 36, 12, 52, 71,
                 46, 83, 65, 84, 34, 82, 77, 78, 74, 20, 43, 35, 69, 70, 30, 68, 45,
                 57, 44, 21, 75, 26, 51, 76, 39, 48, 28, 63, 59, 66, 72, 17, 67, 64,
                 31, 25, 15, 54, 58, 62, 56, 23, 37, 73, 50, 27,  9, 61,  7,  6,
4],
      dtype=int64)
```

Goalkeeper Stats

3.21.36 Goalkeeping

In [177]:
```
fdc['Goalkeeping'].dtype
```

Out[177]:
```
dtype('int64')
```

In [178]:
```
missing_values = fdc['Goalkeeping'].isnull().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [179]:
```
fdc['Goalkeeping'].unique()
```

Out[179]:
```
array([ 54,  58, 437,  56,  59,  51,  62, 439,  42,  67, 420, 440,  41,
        46,  63,  60,  26, 435, 424,  43,  45,  52,  50,  47,  53,  44,
       418,  15,  48, 416, 153, 413,  65,  64,  20, 423,  49,  55,  66,
        40,  57, 421,  13,  39,  61, 419,  21, 409,  37, 406, 410,  36,
       408,  34,  29, 405, 403, 402, 407,  16,  69, 391, 401, 398, 400,
        22,  68, 396,  38,  78,  73, 399, 390, 393, 395, 397,  80,  70,
       389, 394, 388,  27,  30,  75,  71, 386,  74, 378, 385, 384, 380,
       392, 381,  10, 387, 383, 375, 382,  19, 379,  24, 369, 356, 368,
       373, 370, 372,  72, 374, 376, 364,  25, 367,  17, 377, 371, 365,
       352, 362, 359, 363, 366,  82,  35, 361, 358,  76, 294,  83, 357,
       360, 355, 354,  77, 229, 350, 353, 347, 351,  32, 349, 169, 346,
       348, 343, 345, 339, 342,  33, 341,  28, 119, 337, 338, 340, 344,
       335,  98, 324, 248, 334, 298, 336, 328, 331, 321, 332,  81,  79,
       333, 278, 329, 261, 325,  31, 327, 330, 322, 305, 326, 283, 320,
       323, 318,  18, 319, 316, 317, 272, 315,  88, 311, 310, 314, 313,
       307, 312, 309, 308, 301, 304, 292, 303, 306, 296, 289, 300, 302,
       297, 290, 299, 293, 295, 291, 288,  93, 284, 287, 286, 285, 273,
       282, 279, 281, 280, 277, 275, 276, 274, 270, 271, 268, 269, 267,
       260, 265, 262, 266, 263, 264, 251, 259, 254, 257, 252, 255, 256,
       258, 247, 250, 243, 253, 249, 245, 236, 246, 234, 241, 231],
      dtype=int64)
```

3.21.37 GK Diving

In [180]:
```
fdc['GK Diving'].dtype
```

Out[180]:
```
dtype('int64')
```

In [181]:
```
missing_values = fdc['GK Diving'].isnull().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [182]:
```
fdc['GK Diving'].unique()
```

Out[182]:
```
array([ 6,  7, 87, 15,  9, 14, 86, 13, 88, 10, 84, 11,  8,  5, 12, 90,  3,
       27, 89, 80, 16, 85,  2, 82, 79, 83,  4, 81, 77, 18, 78, 17, 75, 74,
       76, 73, 71, 72, 52, 68, 70, 54, 69, 32, 66, 65, 67, 61, 22, 64, 23,
       40, 63, 55, 19, 50, 62, 58, 60, 59, 56, 57, 53, 51, 49, 46, 48, 47,
       45], dtype=int64)
```

3.21.38 GK Handling

In [183]:

```
fdc['GK Handling'].dtype
```

Out[183]:

```
dtype('int64')
```

In [184]:

```
missing_values = fdc['GK Handling'].isnull().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [185]:

```
fdc['GK Handling'].unique()
```

Out[185]:

```
array([11, 92, 13,  9,  6, 14, 88,  5, 85, 10, 89, 87,  8, 15, 12,  4, 82,
       81,  3,  7, 25, 86, 83,  2, 80, 16, 77, 79, 78, 76, 84, 75, 72, 74,
       71, 69, 73, 70, 67, 68, 65, 61, 62, 64, 41, 63, 66, 33, 22, 17, 57,
       18, 54, 55, 59, 49, 19, 40, 60, 58, 43, 45, 53, 47, 56, 51, 52, 50,
       48, 46], dtype=int64)
```

3.21.39 GK Kicking

In [186]:

```
fdc['GK Kicking'].dtype
```

Out[186]:

```
dtype('int64')
```

In [187]:

```
missing_values = fdc['GK Kicking'].isnull().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [188]:

```
fdc['GK Kicking'].unique()
```

Out[188]:

```
array([15, 78,  5, 12,  9, 85,  7, 88, 13, 16, 74, 91,  6, 10,  4, 93, 11,
       73, 14, 75,  2, 31, 68, 76,  8, 80, 82,  3, 87, 72, 83, 77, 79, 81,
       69, 71, 20, 67, 70, 64, 65, 63, 44, 60, 84, 54, 48, 61, 18, 66, 17,
       59, 62, 90, 43, 38, 58, 57, 28, 40, 53, 23, 47, 46, 19, 51, 55, 52,
       56, 22, 30, 25, 42, 35, 21, 49, 50, 36, 45], dtype=int64)
```

3.21.40 GK Positioning

In [189]:

```
fdc['GK Positioning'].dtype
```

Out[189]:

```
dtype('int64')
```

In [190]:

```
missing_values = fdc['GK Positioning'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

```
In [191]: fdc['GK Positioning'].unique()
```

```
Out[191]: array([14, 90, 10, 15,  8, 11, 91, 88,  7, 12, 85, 86,  5, 89, 13,  6, 82,
                  4,  9, 87, 33, 84, 16, 83,  2,  3, 79, 81, 80, 76, 78, 19, 77, 17,
                 75, 74, 73, 71, 18, 72, 70, 69, 66, 68, 40, 64, 20, 32, 67, 62, 65,
                 63, 24, 23, 50, 55, 58, 51, 59, 56, 61, 57, 60, 46, 54, 53, 52, 47,
                 49, 48, 43, 45, 42, 38, 44, 41], dtype=int64)
```

3.21.41 GK Reflexes

```
In [192]: fdc['GK Reflexes'].dtype
```

```
Out[192]: dtype('int64')
```

```
In [193]: missing_values = fdc['GK Reflexes'].isna().sum()
          print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

```
In [194]: fdc['GK Reflexes'].unique()
```

```
Out[194]: array([ 8, 11, 90, 13, 10, 14, 89,  6, 12, 88,  7,  9, 15,  5,  3, 37, 85,
                 86,  4, 16, 82, 83, 84, 87, 78, 80, 20, 18, 79, 81, 19, 77, 17,  2,
                 74, 71, 76, 73, 75, 72, 69, 46, 66, 51, 70, 34, 67, 23, 68, 45, 65,
                 21, 59, 54, 47, 61, 64, 63, 62, 60, 58, 56, 57, 55, 53, 50, 52, 49,
                 48, 44], dtype=int64)
```

Player´s Total Stats

3.21.42 Total Stats

```
In [195]: fdc['Total Stats'].dtype
```

```
Out[195]: dtype('int64')
```

```
In [196]: missing_values = fdc['Total Stats'].isna().sum()
          print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

```
In [197]: fdc['Total Stats'].unique()
```

```
Out[197]: array([2231, 2221, 1413, ...,  757,  747,  956], dtype=int64)
```

3.21.43 Base Stats

```
In [198]: fdc['Base Stats'].dtype
```

```
Out[198]: dtype('int64')
```

In [199]:

```
missing_values = fdc['Base Stats'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [200]:

```
fdc['Base Stats'].unique()
```

Out[200]:

```
array([466, 464, 489, 485, 451, 457, 470, 490, 484, 455, 469, 463, 468,
       497, 442, 439, 473, 452, 498, 449, 477, 401, 446, 447, 465, 430,
       461, 422, 476, 460, 453, 467, 471, 399, 424, 441, 459, 438, 437,
       454, 428, 445, 431, 474, 421, 435, 448, 475, 403, 444, 443, 419,
       405, 420, 423, 396, 388, 482, 478, 385, 394, 480, 433, 450, 462,
       456, 436, 434, 429, 400, 440, 425, 410, 458, 398, 413, 373, 406,
       408, 472, 426, 407, 432, 427, 415, 481, 417, 372, 380, 418, 383,
       414, 409, 412, 411, 386, 362, 402, 390, 404, 391, 416, 375, 389,
       361, 397, 366, 392, 393, 382, 368, 387, 352, 376, 384, 378, 379,
       341, 354, 369, 395, 357, 381, 377, 344, 360, 370, 338, 333, 367,
       363, 349, 355, 345, 358, 348, 374, 351, 343, 342, 353, 321, 350,
       365, 364, 371, 327, 331, 359, 347, 356, 339, 319, 317, 335, 346,
       329, 315, 324, 322, 325, 332, 336, 337, 330, 316, 313, 306, 307,
       328, 310, 340, 308, 318, 334, 301, 289, 302, 320, 323, 326, 311,
       297, 314, 304, 292, 305, 312, 294, 287, 300, 299, 285, 303, 288,
       278, 296, 277, 309, 291, 283, 286, 293, 295, 298, 276, 282, 272,
       284, 290, 271, 275, 279, 281, 262, 263, 280, 268, 270, 269, 264,
       273, 265, 252, 267, 257, 274, 266, 259, 247, 261, 251, 233, 239,
       253, 258, 254, 260, 244, 240, 255, 256, 250, 238, 243, 249, 248,
       245, 241, 232], dtype=int64)
```

3.22 W/F (Weak foot)

Player´s Weak foot rating (above 5). The rating is expressed in stars, for future operations we remove the stars and keep the numbers only.

In [201]:

```
fdc['W/F'].dtype
```

Out[201]:

```
dtype('O')
```

In [202]:

```
fdc['W/F'].unique()
```

Out[202]:

```
array(['4 ★', '3 ★', '5 ★', '2 ★', '1 ★'], dtype=object)
```

In [203]:

```
# If we want to do some calculations we need to remove the stars
fdc['W/F'] = fdc['W/F'].str.replace('★', '')
fdc['W/F'].unique()
```

Out[203]:

```
array(['4 ', '3 ', '5 ', '2 ', '1 '], dtype=object)
```

In [204]:

```
# Remove the whitespace
fdc['W/F'] = fdc['W/F'].str.strip()
fdc['W/F'].unique()
```

Out[204]:

```
array(['4', '3', '5', '2', '1'], dtype=object)
```

3.23 SM (Skill Move)

SM refers to 'Skill Move'. It´s a rating based on stars. The more starts better the skill move of the player.

In [205]:

```
fdc['SM'].dtype
```

Out[205]:

```
dtype('O')
```

In [206]:

```
fdc['SM'].unique()
```

Out[206]:

```
array(['4★', '5★', '1★', '2★', '3★'], dtype=object)
```

For this type of values, I am going to replace the numbers for the stars

In [207]:

```
# Replace the stars
fdc['SM'] = fdc['SM'].str.replace('★','').astype(int)
fdc['SM'].dtype
```

Out[207]:

```
dtype('int32')
```

In [208]:

```
# Using a function to iterate the rows and replace the numbers with sta
def replace_num(value):
    if value == 5:
        return '★★★★★'
    elif value == 4:
        return '★★★★'
    elif value == 3:
        return '★★★'
    elif value == 2:
        return'★★'
    else:
        return '★'

fdc['SM'] = fdc['SM'].apply(replace_num)
fdc['SM'] = fdc['SM'].str.strip()
fdc['SM'].head(10)
```

Out[208]:

```
0     ★★★★
1    ★★★★★
2        ★
3     ★★★★
4    ★★★★★
5     ★★★★
6     ★★★★
7        ★
8    ★★★★★
9        ★
Name: SM, dtype: object
```

3.24 A/W (Attacking work rate)

Refers to the player's attacking work rate.

In [209]:

```
fdc['A/W'].dtype
```

Out[209]:

```
dtype('O')
```

In [210]:

```
fdc['A/W'].unique()
```

Out[210]:

```
array(['Medium', 'High', 'Low'], dtype=object)
```

In [211]:

```
fdc['A/W'] = fdc['A/W'].str.strip()
```

In [212]:

```
missing_values = fdc['A/W'].isnull().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

3.25 D/W (Defensive work rate)

Refers to the player's defensive work rate.

In [213]:

```
fdc['D/W'].dtype
```

Out[213]:

```
dtype('O')
```

In [214]:

```
fdc['D/W'].unique()
```

Out[214]:

```
array(['Low', 'Medium', 'High'], dtype=object)
```

In [215]:

```
fdc['D/W'] = fdc['D/W'].str.strip()
fdc['D/W'].head(10)
```

Out[215]:

```
0       Low
1       Low
2    Medium
3      High
4    Medium
5    Medium
6    Medium
7    Medium
8       Low
9    Medium
Name: D/W, dtype: object
```

In [216]:

```
missing_values = fdc['D/W'].isnull().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

3.26 IR (International reputation rating)

This value refers to the player's international reputation rating

In [217]:

```
fdc['IR'].dtype
```

Out[217]:

```
dtype('O')
```

In [218]:

```
fdc['IR'].unique()
```

Out[218]:

```
array(['5 ★', '3 ★', '4 ★', '2 ★', '1 ★'], dtype=object)
```

For these values, I am going to use only the numerical ones.

In [219]:

```
fdc['IR'] = fdc['IR'].str.replace('★', '')
fdc['IR'] = fdc['IR'].str.strip()
fdc['IR'].unique()
```

Out[219]:

```
array(['5', '3', '4', '2', '1'], dtype=object)
```

3.27 PLAYER´S RATINGS

All the values expressed in these columns are integers. They are ratings from 100 to 0. I am going to check for null values and check the type.

3.27.1 PAC (Player´s Pace rating)

In [220]:

```
missing_values = fdc['PAC'].isnull().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [221]:

```
fdc['PAC'].unique()
```

Out[221]:

```
array([85, 89, 87, 76, 91, 78, 93, 86, 96, 88, 94, 65, 84, 74, 71, 77, 68,
       75, 54, 79, 83, 80, 81, 82, 63, 67, 90, 66, 42, 73, 70, 64, 57, 58,
       69, 72, 50, 59, 92, 60, 62, 55, 52, 56, 61, 53, 45, 37, 95, 43, 44,
       46, 48, 49, 47, 34, 39, 40, 51, 41, 36, 32, 33, 30, 31, 38, 35, 28,
       29, 25], dtype=int64)
```

3.27.2 SHO (Player´s Shooting rating)

In [222]:

```
missing_values = fdc['SHO'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

In [223]:

```
fdc['SHO'].unique()
```

Out[223]:

```
array([92, 93, 86, 85, 91, 88, 60, 73, 89, 87, 70, 90, 81, 66, 72, 82, 28,
       74, 77, 62, 50, 83, 69, 80, 46, 76, 54, 49, 61, 58, 79, 68, 59, 41,
       45, 64, 78, 55, 75, 65, 63, 48, 42, 56, 51, 30, 47, 84, 40, 57, 25,
       71, 37, 43, 53, 67, 38, 52, 39, 35, 36, 44, 32, 34, 33, 31, 27, 22,
       29, 26, 23, 18, 24, 20, 16, 21, 19, 17], dtype=int64)
```

3.27.3 PAS (Player´s Passing rating)

In [224]:
```
fdc['PAS'].unique()
```

Out[224]:
```
array([91, 81, 78, 93, 86, 85, 88, 71, 80, 76, 74, 77, 79, 84, 73, 55, 83,
       87, 72, 75, 58, 89, 82, 68, 67, 64, 66, 59, 69, 90, 65, 53, 63, 62,
       70, 56, 42, 54, 61, 57, 60, 48, 52, 47, 46, 44, 45, 50, 51, 49, 43,
       36, 38, 40, 41, 35, 39, 34, 33, 37, 30, 32, 29, 31, 26, 28, 25, 2
7],
      dtype=int64)
```

In [225]:
```
missing_values = fdc['PAS'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

3.27.4 DRI (Player´s Dribbling rating)

In [226]:
```
fdc['DRI'].unique()
```

Out[226]:
```
array([95, 89, 90, 88, 94, 85, 91, 71, 72, 86, 73, 81, 84, 92, 80, 68, 77,
       87, 60, 83, 78, 64, 67, 79, 69, 66, 65, 70, 82, 75, 61, 74, 54, 76,
       49, 63, 59, 62, 56, 55, 50, 57, 58, 52, 53, 51, 48, 47, 46, 39, 44,
       43, 36, 40, 45, 41, 37, 34, 35, 42, 32, 38, 31, 33, 30, 29, 28, 25,
       27], dtype=int64)
```

In [227]:
```
missing_values = fdc['DRI'].isnull().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

3.27.5 DEF (Player´s Defensive rating)

In [228]:
```
fdc['DEF'].unique()
```

Out[228]:
```
array([38, 35, 52, 64, 36, 43, 45, 51, 39, 91, 44, 86, 48, 57, 40, 88, 33,
       81, 63, 47, 53, 89, 71, 37, 80, 68, 85, 61, 90, 83, 49, 56, 58, 82,
       87, 79, 66, 55, 78, 32, 50, 76, 77, 70, 75, 41, 29, 73, 65, 59, 84,
       54, 72, 46, 42, 69, 34, 31, 30, 74, 24, 62, 25, 20, 26, 60, 27, 23,
       28, 67, 22, 19, 18, 21, 17, 15, 16, 12], dtype=int64)
```

In [229]:
```
missing_values = fdc['DEF'].isnull().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

3.27.6 PHY (Player´s Physical rating)

In [230]:
```
fdc['PHY'].unique()
```

Out[230]:
```
array([65, 77, 90, 78, 59, 82, 75, 91, 76, 88, 86, 85, 73, 67, 79, 63, 83,
       89, 66, 69, 72, 64, 71, 81, 87, 68, 84, 80, 55, 70, 44, 62, 51, 57,
       60, 58, 56, 74, 52, 61, 53, 45, 50, 54, 47, 48, 49, 42, 37, 40, 39,
       43, 38, 46, 41, 34, 35, 36, 31, 32, 33, 29, 28], dtype=int64)
```

In [231]:
```
missing_values = fdc['PHY'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  0
```

3.27.8 SHO - PAS - DRI - DEF - PHY ratings

Let's experiment and do the same for the 5 columns using a function

In [232]:
```
cols_to_check = ['SHO', 'PAS', 'DRI', 'DEF', 'PHY']

unique_values = fdc[cols_to_check].astype(str).apply(lambda x: ' '.joir

unique_values
```

Out[232]:
```
array(['92 91 95 38 65', '93 81 89 35 77', '92 78 90 52 90', ...,
       '39 44 46 40 53', '49 41 49 30 44', '22 39 42 45 55'], dtype=objec
t)
```

In [233]:
```
cols_to_check = ['SHO', 'PAS', 'DRI', 'DEF', 'PHY']

null_values = fdc[cols_to_check].astype(str).apply(lambda x: ' '.join(x

print('Number of null values are: ', null_values)
```

```
Number of null values are:  0
```

3.28 Hits

This column expresses the number of times a player was searched in FIFA´s database. For future operations, we are going to remove the 'K' and correct the values.

In [234]:
```
fdc['Hits'].dtype
```

Out[234]:
```
dtype('O')
```

In [235]:

```
fdc['Hits'].unique()
```

Out[235]:

```
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', '12
6',
       '313', '124', '145', '538', '182', '101', '45', '377', '99', '19
4',
       '403', '414', '593', '374', '245', '3.2K', '266', '299', '309',
       '215', '265', '211', '112', '337', '70', '159', '688', '116', '6
3',
       '144', '123', '71', '224', '113', '168', '61', '89', '137', '27
8',
       '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', '16
6',
       '121', '109', '32', '46', '122', '48', '527', '199', '282', '5
1',
```

In [236]:

```
missing_values = fdc['Hits'].isna().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  2595
```

In [237]:

```
fdc['Hits'].fillna(0, inplace=True)
fdc['Hits'].sample(5)
```

Out[237]:

```
11750    2
14087    2
17601    0
16785    0
13875    1
Name: Hits, dtype: object
```

In [238]:
```
fdc['Hits'].unique()
```

Out[238]:
```
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)
```

3.16.a Transform values

For this column, I am going to replace NaN values for 0 and express the K values with their full number

In [239]:

```
# Convert the values
fdc['Hits'] = fdc['Hits'].replace({'K': '*1e3'}, regex=True).map(pd.eva
fdc['Hits'].head(10)
```

Out[239]:

```
0     771
1     562
2     150
3     207
4     595
5     248
6     246
7     120
8    1600
9     130
Name: Hits, dtype: int32
```

In [ ]:

```

```

4. Data Cleaning Result

It´s time to check our database how it ended after our data-cleaning process

In [240]:

```
fdc.columns
```

Out[240]:

```
Index(['ID', 'Name', 'Surname', 'Nationality', 'Age', 'Overall', 'Potentia
l',
       'Club', 'Contract', 'Contract Start', 'Contract End',
       'Contract Length(years)', 'Contract Status', 'Positions', 'Height(c
m)',
       'Weight(kg)', 'Preferred Foot', 'BOV', 'Best Position', 'Joined',
       'Loan Date End', 'Market Price(€)', 'Wage', 'Release Clause(€)',
       'Attacking', 'Crossing', 'Finishing', 'Heading Accuracy',
       'Short Passing', 'Volleys', 'Skill', 'Dribbling', 'Curve',
       'FK Accuracy', 'Long Passing', 'Ball Control', 'Movement',
       'Acceleration', 'Sprint Speed', 'Agility', 'Reactions', 'Balance',
       'Power', 'Shot Power', 'Jumping', 'Stamina', 'Strength', 'Long Shot
s',
       'Mentality', 'Aggression', 'Interceptions', 'Positioning', 'Visio
n',
       'Penalties', 'Composure', 'Defending', 'Marking', 'Standing Tackl
e',
       'Sliding Tackle', 'Goalkeeping', 'GK Diving', 'GK Handling',
       'GK Kicking', 'GK Positioning', 'GK Reflexes', 'Total Stats',
       'Base Stats', 'W/F', 'SM', 'A/W', 'D/W', 'IR', 'PAC', 'SHO', 'PAS',
       'DRI', 'DEF', 'PHY', 'Hits'],
      dtype='object')
```

4.1 Export data frame

Now the data cleaned, it´s time to export the data for future analysis

In [241]:

```
# Export the data to an Excel file
fdc.to_excel('fifa21_clean_data.xlsx', sheet_name="fifa21_data_analysis
```

In [242]:

```
# Export the data to a .csv file
fdc.to_csv('fifa21_clean_data.csv', index=False)
```

5. Data Visualization

Now that the data is cleaned, it´s time to work with it. I am going to use Numpy for the calculations needed and Matplotlib to visualize the data.

5.1 Deleting some columns

I am not going to use all the columns, so I'm going to delete some of them. First let's make a copy of our data.

In [243]:

```
f21 = fdc.copy()
f21.sample(5)
```

Out[243]:

	ID	Name	Surname	Nationality	Age	Overall	Potential	Club	Contract	C
4487	245992	Billy	Gilmour	Scotland	19	71	86	Chelsea	2018 ~ 2023	
18752	258681	Ryan	Hillier	Wales	17	50	65	Newport County	2020 ~ 2021	
1560	216335	Yuriy	Gazinskiy	Russia	30	75	75	No Club	Free	
15929	256145	Felipe	Zenobio	Argentina	20	59	72	Club Atlético Tigre	2019 ~ 2024	
276	239231	Cucurella	Cucurella Saseta	Spain	21	81	89	Getafe CF	2020 ~ 2023	

5 rows × 79 columns

In [244]:

```
# Deleting the columns
f21 = f21.drop(['Potential','Contract','BOV','Best Position'], axis=1)
f21.columns
```

Out[244]:

```
Index(['ID', 'Name', 'Surname', 'Nationality', 'Age', 'Overall', 'Club',
       'Contract Start', 'Contract End', 'Contract Length(years)',
       'Contract Status', 'Positions', 'Height(cm)', 'Weight(kg)',
       'Preferred Foot', 'Joined', 'Loan Date End', 'Market Price(€)', 'Wa
ge',
       'Release Clause(€)', 'Attacking', 'Crossing', 'Finishing',
       'Heading Accuracy', 'Short Passing', 'Volleys', 'Skill', 'Dribblin
g',
       'Curve', 'FK Accuracy', 'Long Passing', 'Ball Control', 'Movement',
       'Acceleration', 'Sprint Speed', 'Agility', 'Reactions', 'Balance',
       'Power', 'Shot Power', 'Jumping', 'Stamina', 'Strength', 'Long Shot
s',
       'Mentality', 'Aggression', 'Interceptions', 'Positioning', 'Visio
n',
       'Penalties', 'Composure', 'Defending', 'Marking', 'Standing Tackl
e',
       'Sliding Tackle', 'Goalkeeping', 'GK Diving', 'GK Handling',
       'GK Kicking', 'GK Positioning', 'GK Reflexes', 'Total Stats',
       'Base Stats', 'W/F', 'SM', 'A/W', 'D/W', 'IR', 'PAC', 'SHO', 'PAS',
       'DRI', 'DEF', 'PHY', 'Hits'],
      dtype='object')
```

5.2 Data Describe

Let's see some descriptive statistics from our data

In [245]:

```
# Me are going to use the .describe() method
f21.describe()
```

Out[245]:

	ID	Age	Overall	Contract Length(years)	Height(cm)	Weight(kg
count	18979.000000	18979.000000	18979.000000	18979.000000	18979.000000	18979.00000
mean	226403.384794	25.194109	65.718636	3.491965	181.200221	75.019021
std	27141.054157	4.710520	6.968999	2.401495	6.840054	7.07354
min	41.000000	16.000000	47.000000	0.000000	155.000000	50.00000
25%	210135.000000	21.000000	61.000000	2.000000	176.000000	70.00000
50%	232418.000000	25.000000	66.000000	3.000000	181.000000	75.00000
75%	246922.500000	29.000000	70.000000	5.000000	186.000000	80.00000
max	259216.000000	53.000000	93.000000	23.000000	206.000000	110.00000

8 rows × 59 columns

5.3 Data Analysis and Visualizations

Now that our data is cleaned, we can start to use the data to answer the questions we need

5.3.1 Which is the Player's preferred foot?

Let's analyze the data from the 'Preferred Foot' column to answer this question and discover the player's preferred foot. To visualize the data, I am going to use a pie chart.

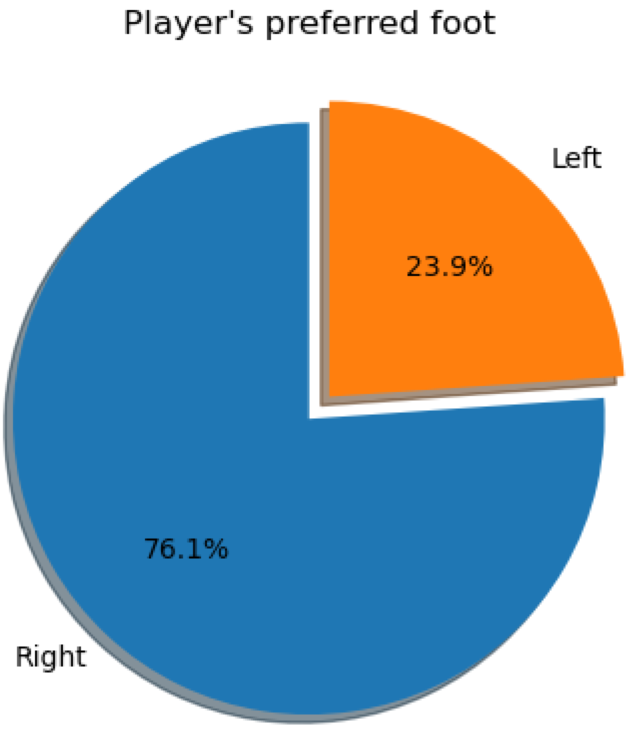
In [246]:

```python
# Count the values from the column
f21_pf = f21['Preferred Foot'].value_counts()
print(f21_pf)

# Creating the pie chart with the data
explode = [0.0, 0.1]
f21['Preferred Foot'].value_counts().plot(
    kind='pie',
    explode=explode,
    title="Player's preferred foot",
    autopct='%1.1f%%',
    shadow=True,
    startangle=90,
    ylabel="")
```

```
Right    14445
Left      4534
Name: Preferred Foot, dtype: int64
```

Out[246]:

```
<Axes: title={'center': "Player's preferred foot"}>
```



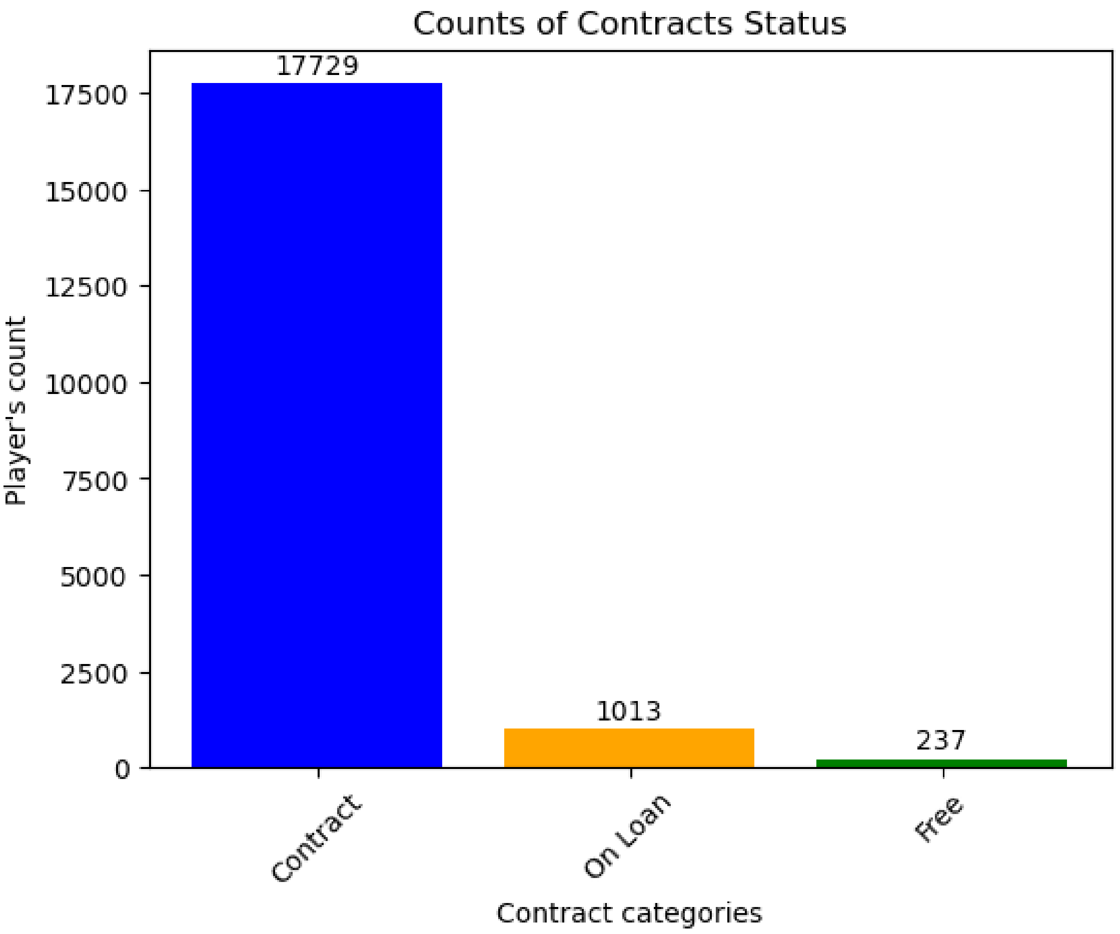
5.3.2 How is the contract status?

Let's analyze the number of players under contract, on loan or free

In [247]:

```
# Count the values of the column
f21_cs = f21['Contract Status'].value_counts()
print(f21_cs)

# Creating the bar plot
bars = plt.bar(f21_cs.index, f21_cs.values, color=['blue', 'orange', 'g

# Adding total values at the top of each bar
for i, v in enumerate(f21_cs):
    plt.text(i, v + 100, str(v), ha='center', va='bottom', fontsize=10)

# Customize labels and title
plt.xlabel("Contract categories")
plt.ylabel("Player's count")
plt.title("Counts of Contracts Status")
plt.xticks(rotation=45)

plt.show()
```

```
Contract    17729
On Loan      1013
Free          237
Name: Contract Status, dtype: int64
```



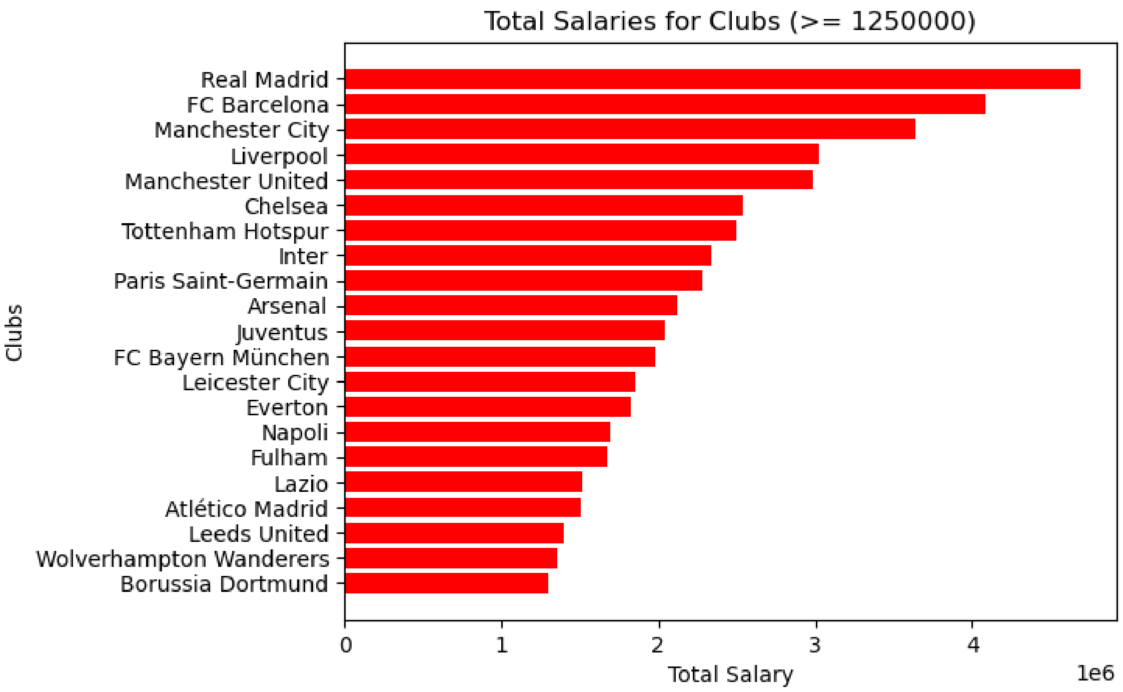
5.3.3 Clubs that pay more in salaries

Now we can also analyze the clubs that pay more in salaries to their player's

In [248]:

```python
# The data is group by clubs and sum their player's salaries
f21_csum = f21.groupby('Club')['Wage'].sum().sort_values()
print(f21_csum)

# Because there are a lot of clubs, I decided to apply a filter
ylab = f21_csum[f21_csum >= 1250000]

# Now let's create a barh chart to visualize our data
plt.barh(ylab.index, ylab.values, color="red")
plt.xlabel('Total Salary')
plt.ylabel('Clubs')
plt.title("Total Salaries for Clubs (>= 1250000)")

plt.show()
```

```
Club
No Club                        0
Llaneros de Guanare        10000
Central Coast Mariners     10350
Aragua FC                  10500
Waterford FC               10650
                          ...  
Manchester United        2986000
Liverpool                3028500
Manchester City          3639000
FC Barcelona             4083000
Real Madrid              4687000
Name: Wage, Length: 682, dtype: int32
```



5.3.4 Player's with the most 'Overall'

Let's say we want to figure out whether the player's 'Overall' is equal to or greater than 80

In [249]:

```
f21_ov = f21[(f21['Overall'] >= 85)]

list_ov = f21_ov[['Name', 'Surname', 'Overall']].sort_values(by='Overal
print(list_ov)
```

```
          Name                    Surname  Overall
0       Lionel                      Messi       93
1    Cristiano  Ronaldo dos Santos Aveiro       92
2          Jan                      Oblak       91
3        Kevin                  De Bruyne       91
4       Neymar        da Silva Santos Jr.       91
..         ...                        ...      ...
73     Clément                    Lenglet       85
74  Marquinhos               Aoás Corrêa       85
75       Riyad                     Mahrez       85
76     Ricardo            Barbosa Pereira       85
98       Marco                       Reus       85

[99 rows x 3 columns]
```

5.3.5 Player's by Nationality

Now it's time to know the player's by their Nationality.

In [250]:

```
f21_count = f21['Nationality'].value_counts()
f21_count
```

Out[250]:

```
England                 1705
Germany                 1195
Spain                   1065
France                  1003
Argentina                943
                        ... 
Malawi                     1
Rwanda                     1
São Tomé & Príncipe        1
Aruba                      1
Indonesia                  1
Name: Nationality, Length: 164, dtype: int64
```

Because there are too many countries with only one player, let's group them, and the condition will be, ***countries with 100 players or fewer***.

In [251]:

```python
# Let's see the list of countries with 100 players or fewer
f21_less_than_100 = f21_count[f21_count <= 100].index.tolist()
print(f21_less_than_100)
```

```
['Greece', 'Northern Ireland', 'Cameroon', 'Morocco', 'Russia', 'Canada',
'South Africa', 'Bosnia Herzegovina', 'Ukraine', 'Slovakia', 'DR Congo',
'Finland', 'Mali', 'Iceland', 'Slovenia', 'Algeria', 'Albania', 'Kosovo',
'New Zealand', 'Hungary', 'Bulgaria', 'Tunisia', 'Egypt', 'India', 'Costa
Rica', 'Montenegro', 'Guinea', 'Cape Verde', 'United Arab Emirates', 'Jama
ica', 'North Macedonia', 'Gambia', 'Georgia', 'Burkina Faso', 'Israel', 'I
ran', 'Guinea Bissau', 'Angola', 'Gabon', 'Honduras', 'Congo', 'Togo', 'Zi
mbabwe', 'Comoros', 'Panama', 'Moldova', 'Luxembourg', 'Benin', 'Haiti',
'Curacao', 'Zambia', 'Kenya', 'Lithuania', 'Sierra Leone', 'Madagascar',
'Uganda', 'Cyprus', 'Guyana', 'Uzbekistan', 'Mauritania', 'Latvia', 'Burun
di', 'Kazakhstan', 'Azerbaijan', 'Equatorial Guinea', 'Dominican Republi
c', 'Trinidad & Tobago', 'Faroe Islands', 'Cuba', 'Estonia', 'Mozambique',
'Liechtenstein', 'Libya', 'Iraq', 'El Salvador', 'Niger', 'Antigua & Barbu
da', 'Syria', 'Grenada', 'Liberia', 'Armenia', 'Thailand', 'Sudan', 'Monts
errat', 'Jordan', 'Belarus', 'Lebanon', 'Philippines', 'Central African Re
public', 'Namibia', 'Belize', 'Ethiopia', 'South Sudan', 'Palestine', 'Hon
g Kong', 'Eritrea', 'Afghanistan', 'Chinese Taipei', 'Saint Kitts and Nevi
s', 'Guatemala', 'Malaysia', 'Nicaragua', 'Chad', 'Singapore', 'Tanzania',
'Macau', 'Barbados', 'Korea DPR', 'Malta', 'Andorra', 'Guam', 'Bermuda',
'New Caledonia', 'Puerto Rico', 'Papua New Guinea', 'Saint Lucia', 'Malaw
i', 'Rwanda', 'São Tomé & Príncipe', 'Aruba', 'Indonesia']
```

In [252]:

```python
# Make a copy of the column
new_f21_count = f21['Nationality'].copy()
print(new_f21_count)
```

```
0        Argentina
1         Portugal
2         Slovenia
3          Belgium
4           Brazil
           ...
18974     China PR
18975      England
18976      England
18977     China PR
18978     China PR
Name: Nationality, Length: 18979, dtype: object
```

In [253]:

```python
# Replace the countries with 100 or fewer players
new_f21_count.loc[new_f21_count.isin(f21_less_than_100)] = 'Others'
print(new_f21_count)
```

```
0        Argentina
1         Portugal
2           Others
3          Belgium
4           Brazil
           ...
18974     China PR
18975      England
18976      England
18977     China PR
18978     China PR
Name: Nationality, Length: 18979, dtype: object
```

In [254]:

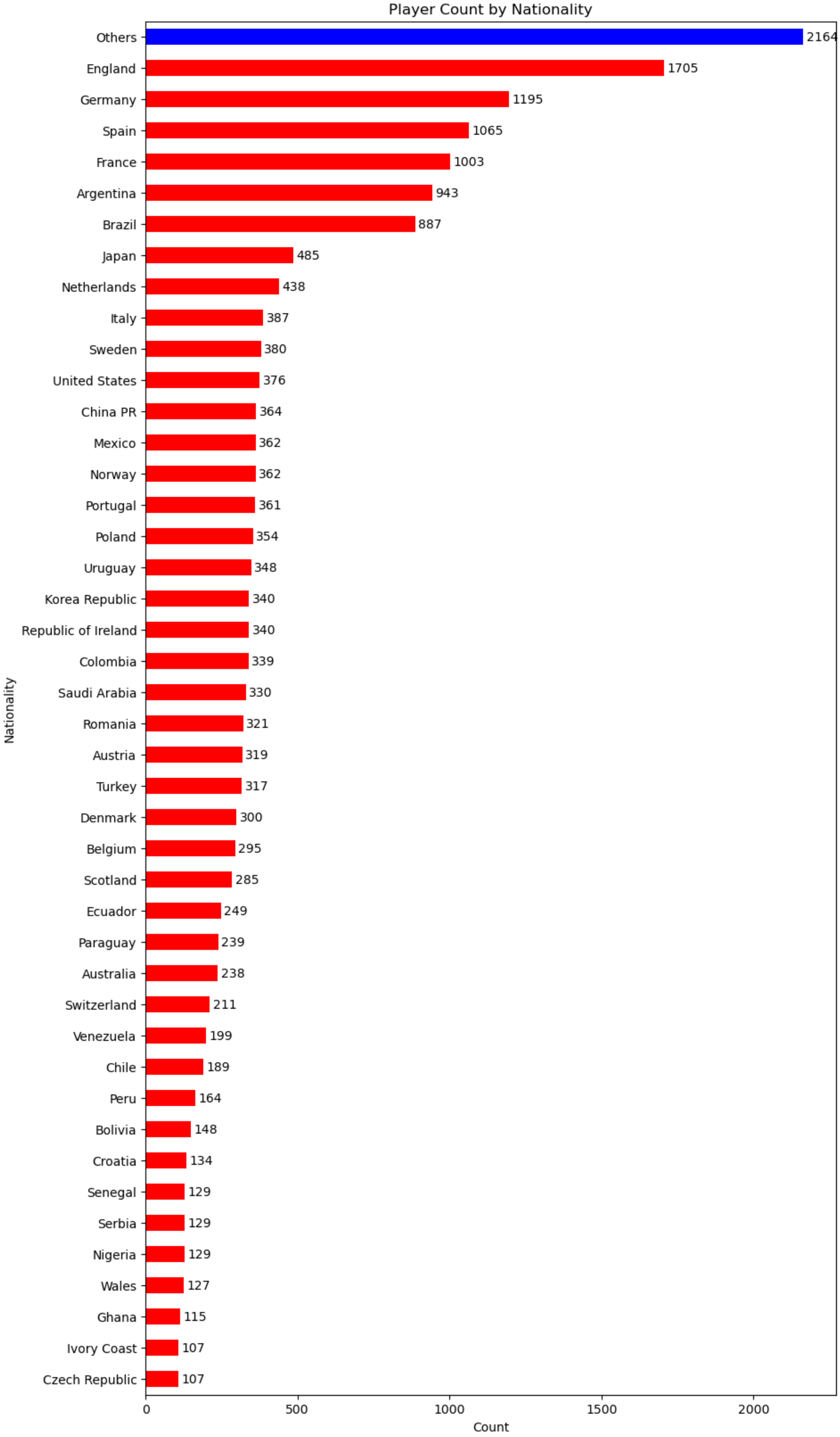
```
# Count the new values
new_count = new_f21_count.value_counts(ascending=True)
new_count
```

Out[254]:

```
Czech Republic          107
Ivory Coast             107
Ghana                   115
Wales                   127
Nigeria                 129
Serbia                  129
Senegal                 129
Croatia                 134
Bolivia                 148
Peru                    164
Chile                   189
Venezuela               199
Switzerland             211
Australia               238
Paraguay                239
Ecuador                 249
Scotland                285
Belgium                 295
Denmark                 300
Turkey                  317
Austria                 319
Romania                 321
Saudi Arabia            330
Colombia                339
Republic of Ireland     340
Korea Republic          340
Uruguay                 348
Poland                  354
Portugal                361
Norway                  362
Mexico                  362
China PR                364
United States           376
Sweden                  380
Italy                   387
Netherlands             438
Japan                   485
Brazil                  887
Argentina               943
France                 1003
Spain                  1065
Germany                1195
England                1705
Others                 2164
Name: Nationality, dtype: int64
```

In [255]:

```
# Creating our chart barh
colors = ['red' if index != 'Others' else 'blue' for index in new_count
ax = new_count.plot(kind='barh', figsize=(10,20), color=colors)
plt.xlabel('Count')
plt.ylabel('Nationality')
plt.title('Player Count by Nationality')
for i, v in enumerate(new_count):
    ax.text(v +10, i, str(v), color='black', va='center')
```


Player Count by Nationality
Nationality
Count
0
500
1000
1500
2000
Others 2164
England 1705
Germany 1195
Spain 1065
France 1003
Argentina 943
Brazil 887
Japan 485
Netherlands 438
Italy 387
Sweden 380
United States 376
China PR 364
Mexico 362
Norway 362
Portugal 361
Poland 354
Uruguay 348
Korea Republic 340
Republic of Ireland 340
Colombia 339
Saudi Arabia 330
Romania 321
Austria 319
Turkey 317
Denmark 300
Belgium 295
Scotland 285
Ecuador 249
Paraguay 239
Australia 238
Switzerland 211
Venezuela 199
Chile 189
Peru 164
Bolivia 148
Croatia 134
Senegal 129
Serbia 129
Nigeria 129
Wales 127
Ghana 115
Ivory Coast 107
Czech Republic 107

In [ ]: