Fifa Data Cleaning Challenge

Data Cleaning is a very important part of data analysis. Clean Data ensures that your analysis is correct and your insights are reliable when used in decision making.

Objectives:

- Ensure that all columns are clearly named

- Ensure that columns have the correct datatypes

- Remove all unnecessary information from the dataset

Lets Get Started:

Import all the necessary libraries needed for this project

In [1]:

```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
```

import the csv data set from the folder where its saved

In [3]:

```
df = pd.read_csv(r'C:\Users\Admin\Desktop\csv files\fifa21 raw data v2.csv')
```

set your dataframe such that it displays all columns since the dataset has many columns. The benefit is that not only all colums are well displayed, but also the printed rows can be larger than the usual ~100 characters limit.

In [5]:

```
pd.set_option('display.max_rows', 1000)
pd.set_option('display.max_columns', 1000)
pd.set_option('display.width', 1000)
```

Display the first five rows

In [7]:

```
df.head()
```

Out[7]:

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

Check how many rows and columns make up your dafaframe

In [9]:

```
df.shape
```

Out[9]:

```
(18979, 77)
```

In [11]:

```
print('The number of rows are:', df.shape[0])
print('The number of columns are:', df.shape[1])
```

```
The number of rows are: 18979
The number of columns are: 77
```

Get summary information from your dataset.

In [13]:

```
df.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
```

1. From the above information the dataset has 18979 rows and 77 columns

2. Most columns dont have null values except 2 columns that is:

- Loan Date End

- Hits.

3. Also the columns are made up of two data types:

- object

- int64

Column Names:

Display a list of all column names and iterate through
to see columns of your data frame

In [17]:

```
for x in df.columns.tolist():
    print(x)
```

```
ID
Name
LongName
photoUrl
playerUrl
Nationality
Age
↓OVA
POT
Club
Contract
Positions
Height
Weight
Preferred Foot
BOV
Best Position
Joined
Loan Date End
Value
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 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
W/F
SM
A/W
D/W
IR
PAC
SHO
PAS
DRI
DEF
PHY
Hits
```

Make a copy of your data set so as to retain an original copy

```
In [20]: df1 = df.copy()
```

In [22]:

```
df1.head()
```

Out[22]:

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

Remove unnecessary columns

In [24]:

```
df1 = df1.drop(['Name', 'photoUrl', 'playerUrl'], axis = 1)
```

In [26]:

```
df1.head(1)
```

Out[26]:

	ID	LongName	Nationality	Age	↓OVA	POT	Club	Contract	Positions	Height	Weight	Preferred Foot	BOV	Best Position	Jo
0	158023	Lionel Messi	Argentina	33	93	93	\n\n\n\nFC Barcelona	2004 ~ 2021	RW, ST, CF	170cm	72kg	Left	93	RW	0

Rename columns

In [28]:

```
df1 = df1.rename(columns = {
                "LongName": "Name",
                "↓OVA":"Overall Rating(%)",
                 "POT": "Potential(%)",
                 "BOV": "Best Overall(%)",
                 "BP": "Best Position",
                 "W/F": "Weak Foot",
                 "SM": "Skill Moves",
                 "A/W": "Attacking Work Rate",
                 "D/W": "Defensive Work Rate",
                 "IR": "International Reputation",
                 "PAC": "Pace",
                 "SHO": "Shooting",
                 "PAS": "Passing",
                 "DRI": "Dribbling",
                 "DEF": "Defense",
                 "PHY": "Physicality"
})
```

Check for duplicates

In [33]:

```
df1['Name'].nunique()
```

Out[33]:

```
18852
```

Out of 18979 rows there are 18852 unique rows

In [35]:

```
duplicated_rows = df1[df1.duplicated(["Name"])]
duplicated_rows.head(2)
```

Out[35]:

	ID	Name	Nationality	Age	Overall Rating(%)	Potential(%)	Club	Contract	Positions	Height	Weight	Preferred Foot	O
1239	215051	Lisandro López	Argentina	30	76	76	\n\n\n\nBoca Juniors	2020 ~ 2023	CB	188cm	80kg	Right	
2511	213017	Ben Davies	England	24	73	79	\n\n\n\nPreston North End	2013 ~ 2021	CB	185cm	74kg	Left	

I investigated one of the values to see if truly they are duplicates but based on the findings below they are not duplicated values they only have similar names.

In [37]:

```
filt = df[df1.Name == "Ben Davies"]
filt
```

Out[37]:

	ID	Name	LongName	photoUrl	playerUrl	Nationality	Age
382	205923	B. Davies	Ben Davies	https://cdn.sofifa.com/players/205/923/21_60.png	http://sofifa.com/player/205923/ben-davies/210...	Wales	27
2511	213017	B. Davies	Ben Davies	https://cdn.sofifa.com/players/213/017/21_60.png	http://sofifa.com/player/213017/ben-davies/210...	England	24

Remove whitespaces in the club column

In [40]:

```
df1['Club'].head()
```

Out[40]:

```
0          \n\n\n\nFC Barcelona
1              \n\n\n\nJuventus
2       \n\n\n\nAtlético Madrid
3       \n\n\n\nManchester City
4    \n\n\n\nParis Saint-Germain
Name: Club, dtype: object
```

In [42]:

```
#Remove whitespaces from the club column
df1['Club'] = df1['Club'].str.lstrip()
```

In [44]:

```
df1.head(2)
```

Out[44]:

	ID	Name	Nationality	Age	Overall Rating(%)	Potential(%)	Club	Contract	Positions	Height	Weight	Preferred Foot	Best Overall(%)
0	158023	Lionel Messi	Argentina	33	93	93	FC Barcelona	2004 ~ 2021	RW, ST, CF	170cm	72kg	Left	93
1	20801	C. Ronaldo dos Santos Aveiro	Portugal	35	92	92	Juventus	2018 ~ 2022	ST, LW	187cm	83kg	Right	92

Check for unique values in the contract column

In [47]:

```
df1['Contract'].unique()
```

Out[47]:

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

In [49]:

```
#define a function to change contract column values
def contract_status(value):
    if 'On Loan' in value:
        value = 'On Loan'
        return value
    elif '~' in value:
        value = 'Active'
        return value
    else:
        value = 'Free'
        return value
```

In [51]:

```
#apply the function on contract column
df1['Contract'] = df1['Contract'].apply(contract_status).astype('category')
```

In [53]:

```
#Check for ujnique values in the contract column
df1['Contract'].unique()
```

Out[53]:

```
['Active', 'On Loan', 'Free']
Categories (3, object): ['Active', 'Free', 'On Loan']
```

In [55]:

```
#Rename the contract column
df1 = df1.rename(columns = {'Contract': 'Contract Status'})
df1.head(1)
```

Out[55]:

	ID	Name	Nationality	Age	Overall Rating(%)	Potential(%)	Club	Contract Status	Positions	Height	Weight	Preferred Foot	Best Overall(%)
0	158023	Lionel Messi	Argentina	33	93	93	FC Barcelona	Active	RW, ST, CF	170cm	72kg	Left	93

Process of cleaning contract column

A function was defined to change the row values from '2004 ~ 2021', 'On Loan' and 'Free' to 'Active', 'On Loan' and 'Free'
'Contract' column was renamed to 'Contract Status
. Also, the data type was changed to categoy.

Inspect Position and best position columns

In [57...

```
#Position and best position
df1[['Positions', 'Best Position']].head()
```

Out[57]:

	Positions	Best Position
0	RW, ST, CF	RW
1	ST, LW	ST
2	GK	GK
3	CAM, CM	CAM
4	LW, CAM	LW

From the above findings Positions holds the players best position and other positions held by the player so i dropped positions column.

In [60…

```
#drop the position column
df1 = df1.drop('Positions', axis = 1)
df1.head(2)
```

Out[60]:

	ID	Name	Nationality	Age	Overall Rating(%)	Potential(%)	Club	Contract Status	Height	Weight	Preferred Foot	Best Overall(%)	Bes Positio
0	158023	Lionel Messi	Argentina	33	93	93	FC Barcelona	Active	170cm	72kg	Left	93	RW
1	20801	C. Ronaldo dos Santos Aveiro	Portugal	35	92	92	Juventus	Active	187cm	83kg	Right	92	S

Check Height and Weight Columns

In [63…

```
#Check for unique values
for column in df[['Height', 'Weight']]:
    value = df1[column].unique()
    print(f'{column}\n{value}.\n')
```

```
Height
['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'].

Weight
['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'].
```

In [65…

```
#Function to convert height to cm
def convert_height(value):
    if 'cm' in value:
        value = int(value[:-2])
        return value
    else:
        feet, inches = value.split("'")
        total_inches = int(feet) * 12 + int(inches[:-1])
        height_cm = total_inches * 2.54
        return round(height_cm, 2)

df1['Height'] = df1['Height'].apply(convert_height).astype('int64')
df1['Height'].unique()
```

Out[65]:

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

In [67…

```
#function to convert weight to kg
def convert_weight(value):
    if 'kg' in value:
        value = value.strip('kg')
        return value
    else:
        value = value.strip('lbs')
        Weight = round((float(value) * 0.45359237), 2)
        return Weight

df1['Weight'] = df1['Weight'].apply(convert_weight).astype('int64')
df1['Weight'].unique()
```

Out[67]:

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

In [69…

```
#Rename Height and Weight columns
df1 = df1.rename(columns = {'Height' : 'Height(cm)', 'Weight' : 'Weight(kg)'})
df1.head()
```

Out[69]:

	ID	Name	Nationality	Age	Overall Rating(%)	Potential(%)	Club	Contract Status	Height(cm)	Weight(kg)	Preferred Foot	Best Overall(%)
0	158023	Lionel Messi	Argentina	33	93	93	FC Barcelona	Active	170	72	Left	93
1	20801	C. Ronaldo dos Santos Aveiro	Portugal	35	92	92	Juventus	Active	187	83	Right	92
2	200389	Jan Oblak	Slovenia	27	91	93	Atlético Madrid	Active	188	87	Right	91
3	192985	Kevin De Bruyne	Belgium	29	91	91	Manchester City	Active	181	70	Right	91
4	190871	Neymar da Silva Santos Jr.	Brazil	28	91	91	Paris Saint-Germain	Active	175	68	Right	91

Process of Cleaning 'Height' and 'Weight' columns

1. Create a function to convert Height
2. values with cm remain the same except that we extract only value without 'CM'
3. value in feet and inches we convert them to CM
4. Lastly, convert Height column to int

Process of cleaning Weight column

1. create a function to convert LBS to kgs
2. If value is in "KGS" it remains the same
3. If value is in "LBS" convert them to KGS by multiplying by 0.4535...

Convert date columns to date time using pandas

In [71…

```
for column in df[['Joined', 'Loan Date End']]:
    value = df1[column].unique()
    print(f'{column}\n{value}\n')
```

```
Joined
['01-Jul-04' '10-Jul-18' '16-Jul-14' ... '22-Sep-18' '28-Feb-15'
 '06-Mar-18']

Loan Date End
[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']
```

In [73…

```
#Covert date columns to datetime data types
df1['Joined'] = pd.to_datetime(df1['Joined'])
```

In [75…

```
df1['Loan Date End'] = pd.to_datetime(df1['Loan Date End'])
```

In [77…

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18979 entries, 0 to 18978
Data columns (total 73 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   ID                        18979 non-null  int64
 1   Name                      18979 non-null  object
 2   Nationality               18979 non-null  object
 3   Age                       18979 non-null  int64
 4   Overall Rating(%)         18979 non-null  int64
 5   Potential(%)              18979 non-null  int64
 6   Club                      18979 non-null  object
 7   Contract Status           18979 non-null  category
 8   Height(cm)                18979 non-null  int64
 9   Weight(kg)                18979 non-null  int64
 10  Preferred Foot            18979 non-null  object
 11  Best Overall(%)           18979 non-null  int64
 12  Best Position             18979 non-null  object
 13  Joined                    18979 non-null  datetime64[ns]
 14  Loan Date End             1013 non-null   datetime64[ns]
 15  Value                     18979 non-null  object
 16  Wage                      18979 non-null  object
 17  Release Clause            18979 non-null  object
 18  Attacking                 18979 non-null  int64
 19  Crossing                  18979 non-null  int64
 20  Finishing                 18979 non-null  int64
 21  Heading Accuracy          18979 non-null  int64
 22  Short Passing             18979 non-null  int64
 23  Volleys                   18979 non-null  int64
 24  Skill                     18979 non-null  int64
 25  Dribbling                 18979 non-null  int64
 26  Curve                     18979 non-null  int64
 27  FK Accuracy               18979 non-null  int64
 28  Long Passing              18979 non-null  int64
 29  Ball Control              18979 non-null  int64
 30  Movement                  18979 non-null  int64
 31  Acceleration              18979 non-null  int64
 32  Sprint Speed              18979 non-null  int64
 33  Agility                   18979 non-null  int64
 34  Reactions                 18979 non-null  int64
 35  Balance                   18979 non-null  int64
 36  Power                     18979 non-null  int64
 37  Shot Power                18979 non-null  int64
 38  Jumping                   18979 non-null  int64
 39  Stamina                   18979 non-null  int64
 40  Strength                  18979 non-null  int64
 41  Long Shots                18979 non-null  int64
 42  Mentality                 18979 non-null  int64
 43  Aggression                18979 non-null  int64
 44  Interceptions             18979 non-null  int64
 45  Positioning               18979 non-null  int64
 46  Vision                    18979 non-null  int64
 47  Penalties                 18979 non-null  int64
 48  Composure                 18979 non-null  int64
 49  Defending                 18979 non-null  int64
 50  Marking                   18979 non-null  int64
 51  Standing Tackle           18979 non-null  int64
 52  Sliding Tackle            18979 non-null  int64
 53  Goalkeeping               18979 non-null  int64
 54  GK Diving                 18979 non-null  int64
 55  GK Handling               18979 non-null  int64
 56  GK Kicking                18979 non-null  int64
 57  GK Positioning            18979 non-null  int64
 58  GK Reflexes               18979 non-null  int64
 59  Total Stats               18979 non-null  int64
 60  Base Stats                18979 non-null  int64
 61  Weak Foot                 18979 non-null  object
 62  Skill Moves               18979 non-null  object
 63  Attacking Work Rate       18979 non-null  object
 64  Defensive Work Rate       18979 non-null  object
 65  International Reputation  18979 non-null  object
 66  Pace                      18979 non-null  int64
 67  Shooting                  18979 non-null  int64
 68  Passing                   18979 non-null  int64
 69  Dribbling                 18979 non-null  int64
 70  Defense                   18979 non-null  int64
 71  Physicality               18979 non-null  int64
 72  Hits                      16384 non-null  object
dtypes: category(1), datetime64[ns](2), int64(56), object(14)
memory usage: 10.4+ MB
```

Look at "Value", "Wage", "Release Clause" columns

In [80…

```
df1[["Value", "Wage", "Release Clause"]].head(4)
```

Out[80]:

	Value	Wage	Release Clause
0	€103.5M	€560K	€138.4M
1	€63M	€220K	€75.9M
2	€120M	€125K	€159.4M
3	€129M	€370K	€161M

In [82…

```
# Creating function to convert string to numeric
def convert_to_numeric(value):
    #Remove currency symbol
    value_str = value.replace('€', '').replace(',', '')
    if 'K' in value_str:
        return float(value_str.replace('K', '')) * 1000
    elif 'M' in value_str:
        return float(value_str.replace('M', '')) * 1000000
    else:
        return float(value_str)
# Apply the conversion function to the "Wage" and "Value" columns
df1["Wage"] = df1["Wage"].apply(convert_to_numeric)
df1["Value"] = df1["Value"].apply(convert_to_numeric)
df1["Release Clause"] = df1["Release Clause"].apply(convert_to_numeric)
```

In [84…

```
#display the first five rows
df1[["Value", "Wage", "Release Clause"]].head()
```

Out[84]:

	Value	Wage	Release Clause
0	103500000.0	560000.0	138400000.0
1	63000000.0	220000.0	75900000.0
2	120000000.0	125000.0	159400000.0
3	129000000.0	370000.0	161000000.0
4	132000000.0	270000.0	166500000.0

Process

1. Create a function to convert values to numeric
2. Remove '€' and ',' from the columns
3. Check for 'K' in values and multiply by 1000
4. Check for 'M' in value and multiply by 1000000
5. Apply the function to those columns

Inspect 'Weak Foot', 'Skill Moves', 'International Reputation' Columns

In [86…

```
df1.head(1)
```

Out[86]:

	ID	Name	Nationality	Age	Overall Rating(%)	Potential(%)	Club	Contract Status	Height(cm)	Weight(kg)	Preferred Foot	Best Overall(%)	P
0	158023	Lionel Messi	Argentina	33	93	93	FC Barcelona	Active	170	72	Left	93	

In [88…

```
for column in df1[['Weak Foot', 'Skill Moves', 'International Reputation']]:
    value = df1[column].unique()
    print(f'{column}\n{value}\n')
```

```
Weak Foot
['4 ★' '3 ★' '5 ★' '2 ★' '1 ★']

Skill Moves
['4★' '5★' '1★' '2★' '3★']

International Reputation
['5 ★' '3 ★' '4 ★' '2 ★' '1 ★']
```

In [90…

```
def convert_to_num(value):
    value = int(value[:-1])
    return value

df1['Weak Foot'] = df1['Weak Foot'].apply(convert_to_num).astype('int')
```

In [92…

```
df1['Skill Moves'] = df1['Skill Moves'].apply(convert_to_num).astype('int')
df1['International Reputation'] = df1['International Reputation'].apply(convert_to_num).astype('int')
```

Process

Created a Function that only extracts value leaving out
the last character which is a special character '★'
After removing the special character i converted the column to int data type

Inspect Hits column

In [94…

```
df1["Hits"].unique()
```

Out[94]:

```
array(['771', '562', '150', '207', '595', '248', '246', '120', '1600',
       '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', '1.6K', '515', '943', '1.2K', '903', '335',
       '191', '594', '267', '617', '516', '504', '331', '652', '410',
       '550', '473', '442', '344', '208', '1K', '2.5K', '273', '485',
       '826', '192', '405', '941', '477', '644', '303', '417', '6K', nan,
       11.0, 2.0, 1.0, 31.0, 3.0, 10.0, 9.0, 17.0, 7.0, 4.0, 6.0],
      dtype=object)
```

In [96…

```
#convert string to numeric by extract method using regural expression
def convert_to_num(value):
    if isinstance(value, str):

        if "K" in value:
            value = value.strip('K')
            value = float(value) * 1000
            return value
        elif 'nan' in value:
            return np.nan
        else:
            return float(value)
    else:
        return value
```

```
df1['Hits'] = df1['Hits'].apply(convert_to_num)
df1['Hits'].unique()
```

Out[96]:

```
array([7.71e+02, 5.62e+02, 1.50e+02, 2.07e+02, 5.95e+02, 2.48e+02,
       2.46e+02, 1.20e+02, 1.60e+03, 1.30e+02, 3.21e+02, 1.89e+02,
       1.75e+02, 9.60e+01, 1.18e+02, 2.16e+02, 2.12e+02, 1.54e+02,
       2.05e+02, 2.02e+02, 3.39e+02, 4.08e+02, 1.03e+02, 3.32e+02,
       8.60e+01, 1.73e+02, 1.61e+02, 3.96e+02, 1.10e+03, 4.33e+02,
       2.42e+02, 2.06e+02, 1.77e+02, 1.50e+03, 1.98e+02, 4.59e+02,
       1.17e+02, 1.19e+02, 2.09e+02, 8.40e+01, 1.87e+02, 1.65e+02,
       2.03e+02, 6.50e+01, 3.36e+02, 1.26e+02, 3.13e+02, 1.24e+02,
       1.45e+02, 5.38e+02, 1.82e+02, 1.01e+02, 4.50e+01, 3.77e+02,
       9.90e+01, 1.94e+02, 4.03e+02, 4.14e+02, 5.93e+02, 3.74e+02,
       2.45e+02, 3.20e+03, 2.66e+02, 2.99e+02, 3.09e+02, 2.15e+02,
       2.65e+02, 2.11e+02, 1.12e+02, 3.37e+02, 7.00e+01, 1.59e+02,
       6.88e+02, 1.16e+02, 6.30e+01, 1.44e+02, 1.23e+02, 7.10e+01,
       2.24e+02, 1.13e+02, 1.68e+02, 6.10e+01, 8.90e+01, 1.37e+02,
       2.78e+02, 7.50e+01, 1.48e+02, 1.76e+02, 1.97e+02, 2.64e+02,
       2.14e+02, 2.47e+02, 4.02e+02, 4.40e+02, 1.70e+03, 2.30e+03,
       1.71e+02, 3.20e+02, 6.57e+02, 8.70e+01, 2.59e+02, 2.00e+02,
       2.55e+02, 2.53e+02, 1.96e+02, 6.00e+01, 9.70e+01, 8.50e+01,
       1.69e+02, 2.56e+02, 1.32e+02, 2.39e+02, 1.66e+02, 1.21e+02,
       1.09e+02, 3.20e+01, 4.60e+01, 1.22e+02, 4.80e+01, 5.27e+02,
       1.99e+02, 2.82e+02, 5.10e+01, 1.90e+03, 6.42e+02, 1.55e+02,
       3.23e+02, 2.88e+02, 4.97e+02, 5.09e+02, 7.90e+01, 4.90e+01,
       2.70e+02, 5.11e+02, 8.00e+01, 1.28e+02, 1.15e+02, 1.56e+02,
       2.04e+02, 1.43e+02, 1.40e+02, 1.52e+02, 2.20e+02, 1.34e+02,
       2.25e+02, 9.40e+01, 7.40e+01, 1.35e+02, 1.42e+02, 5.00e+01,
       7.70e+01, 4.00e+01, 1.07e+02, 1.93e+02, 1.79e+02, 3.40e+01,
       6.40e+01, 4.53e+02, 5.70e+01, 8.10e+01, 2.80e+01, 7.80e+01,
       1.33e+02, 4.30e+01, 4.25e+02, 8.80e+01, 4.20e+01, 3.60e+01,
       2.33e+02, 3.76e+02, 2.10e+02, 4.44e+02, 1.00e+02, 2.63e+02,
       9.80e+01, 2.90e+01, 1.60e+02, 3.90e+01, 2.57e+02, 6.00e+00,
       3.10e+02, 1.38e+02, 6.20e+01, 2.93e+02, 2.85e+02, 3.62e+02,
       6.60e+01, 6.90e+01, 5.80e+01, 2.10e+01, 2.00e+01, 1.31e+02,
       3.80e+01, 4.06e+02, 6.80e+01, 1.08e+02, 1.10e+02, 9.30e+01,
       5.12e+02, 4.43e+02, 3.06e+02, 3.52e+02, 4.22e+02, 5.85e+02,
       3.46e+02, 1.78e+02, 8.41e+02, 7.60e+01, 3.94e+02, 7.20e+01,
       1.72e+02, 4.40e+01, 4.07e+02, 2.30e+02, 3.67e+02, 2.95e+02,
       1.57e+02, 2.43e+02, 5.60e+01, 1.11e+02, 3.26e+02, 6.79e+02,
       1.80e+01, 9.20e+01, 5.90e+01, 2.50e+01, 1.84e+02, 5.30e+01,
       1.20e+01, 9.00e+01, 5.50e+01, 7.30e+01, 1.10e+01, 5.66e+02,
       1.80e+02, 8.30e+01, 2.62e+02, 1.70e+01, 2.60e+01, 3.10e+01,
       2.80e+02, 3.59e+02, 2.13e+02, 2.97e+02, 3.87e+02, 4.80e+02,
       3.81e+02, 6.77e+02, 4.86e+02, 8.00e+00, 2.44e+02, 1.29e+02,
       3.88e+02, 2.75e+02, 3.19e+02, 2.00e+03, 5.20e+01, 9.10e+01,
       4.21e+02, 1.53e+02, 2.70e+01, 4.10e+01, 2.22e+02, 3.50e+01,
       1.02e+02, 2.30e+01, 3.00e+01, 3.30e+01, 1.46e+02, 1.30e+01,
       1.90e+01, 1.40e+01, 1.06e+02, 2.76e+02, 5.68e+02, 3.53e+02,
       4.70e+01, 4.78e+02, 2.49e+02, 2.54e+02, 3.69e+02, 2.19e+02,
       5.65e+02, 2.37e+02, 2.27e+02, 4.34e+02, 3.75e+02, 1.62e+02,
       6.05e+02, 6.54e+02, 3.00e+00, 7.00e+00, 9.00e+00, 1.04e+02,
       1.14e+02, 1.86e+02, 4.46e+02, 7.56e+02, 2.20e+01, 1.39e+02,
       5.00e+02, 6.70e+01, 1.47e+02, 1.49e+02, 1.60e+01, 8.20e+01,
       5.40e+01, 3.70e+01, 1.50e+01, 1.30e+03, 3.00e+03, 9.52e+02,
       5.00e+00, 7.49e+02, 5.41e+02, 3.30e+02, 3.93e+02, 5.17e+02,
       7.70e+02, 4.09e+02, 1.70e+02, 1.25e+02, 2.83e+02, 3.42e+02,
       3.63e+02, 5.80e+02, 1.05e+02, 2.17e+02, 2.40e+01, 1.41e+02,
       1.00e+01, 4.27e+02, 1.58e+02, 4.26e+02, 4.00e+00, 6.66e+02,
       1.81e+02, 3.24e+02, 9.79e+02, 1.40e+03, 3.02e+02, 7.51e+02,
       2.98e+02, 4.11e+02, 9.44e+02, 2.00e+00, 9.47e+02, 2.92e+02,
       3.49e+02, 6.21e+02, 1.00e+00, 2.80e+03, 3.38e+02, 2.87e+02,
       2.61e+02, 2.18e+02, 1.80e+03, 2.40e+02, 2.79e+02, 2.29e+02,
       1.88e+02, 3.15e+02, 6.64e+02, 6.13e+02, 1.90e+02, 7.06e+02,
       1.27e+02, 4.62e+02, 3.86e+02, 6.95e+02, 4.91e+02, 1.67e+02,
       2.81e+02, 2.50e+02, 3.07e+02, 9.50e+01, 2.31e+02, 1.74e+02,
       6.80e+02, 6.33e+02, 2.21e+02, 3.48e+02, 6.02e+02, 1.83e+02,
       6.53e+02, 1.95e+02, 1.64e+02, 1.51e+02, 2.58e+02, 8.40e+03,
       3.43e+02, 4.19e+02, 6.55e+02, 1.36e+02, 3.99e+02, 5.31e+02,
       3.57e+02, 2.28e+02, 3.85e+02, 3.12e+02, 3.40e+02, 2.38e+02,
       4.87e+02, 3.55e+02, 4.99e+02, 4.30e+03, 2.96e+02, 5.15e+02,
       9.43e+02, 1.20e+03, 9.03e+02, 3.35e+02, 1.91e+02, 5.94e+02,
       2.67e+02, 6.17e+02, 5.16e+02, 5.04e+02, 3.31e+02, 6.52e+02,
       4.10e+02, 5.50e+02, 4.73e+02, 4.42e+02, 3.44e+02, 2.08e+02,
       1.00e+03, 2.50e+03, 2.73e+02, 4.85e+02, 8.26e+02, 1.92e+02,
       4.05e+02, 9.41e+02, 4.77e+02, 6.44e+02, 3.03e+02, 4.17e+02,
       6.00e+03,      nan])
```

In [98…

```
#remove Exponential and have the values as digits
df1['Hits'] = df1['Hits'].apply(lambda x: f"{x:.0f}")
df1['Hits'].unique()
```

```
Out[98]: array(['771', '562', '150', '207', '595', '248', '246', '120', '1600',
                '130', '321', '189', '175', '96', '118', '216', '212', '154',
                '205', '202', '339', '408', '103', '332', '86', '173', '161',
                '396', '1100', '433', '242', '206', '177', '1500', '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', '3200', '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',
                '1700', '2300', '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',
                '1900', '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', '2000', '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', '1300', '3000',
                '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', '1400',
                '302', '751', '298', '411', '944', '2', '947', '292', '349', '621',
                '1', '2800', '338', '287', '261', '218', '1800', '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', '8400', '343', '419', '655', '136', '399',
                '531', '357', '228', '385', '312', '340', '238', '487', '355',
                '499', '4300', '296', '515', '943', '1200', '903', '335', '191',
                '594', '267', '617', '516', '504', '331', '652', '410', '550',
                '473', '442', '344', '208', '1000', '2500', '273', '485', '826',
                '192', '405', '941', '477', '644', '303', '417', '6000', 'nan'],
               dtype=object)
```

```
In [10…  #convert the hits column to a float because of Nan
         df1['Hits'] = df1['Hits'].astype('float')
```

Process to covert hits to values

1. create a function to convert the values to a number
by removing character 'K' and multiply it by a 1000
2. replaced nan with numpy NANs
3. removed the exponential from the values

Save the cleaned data as a CSV File to the current working directorate

```
In [10…  #save clean data as a csv
         df1.to_csv("Fifa_cleaned_data.csv")
```

Conclusion

In this project we will walk through the data cleaning process.
The objective is to prepare the data for analysis by removing inconsistencies
and converting it into a more usable format. The following key tasks were undertaken:

1. Make a copy of your DataFrame

Make a copy of your data so as to retain an original copy

2. Remove Unnecessary Columns

Remove columns that are not significant to your analysis

3. Rename columns

Rename columns so as to have correct column names that can be understood

4. Check for duplicate values

Check for duplicated rows in your dataset

5. Convert Data types of Height and Weight columns

The data types of the "Height" and "Weight" columns were converted, and heights were transformed from feet (ft) to centimeters (cm), while weights were converted from pounds (lbs) to kilograms (kg).

6. Clean the Contract Column

Created a function to clean contract column

7. Converting Date Strings into Datetime Format

We converted date strings into the datetime data type, making it easier to work with time-related information.

8. Converting datatypes of Value, Wage, Release Clause columns

Transformed "Value", "Wage", "Release Clause" columns to int data types

9. Remove '★' from "Weak Foot", "Skill Move", "International Reputation"

Removed "★" character from the columns "Weak Foot", "Skill Move", "International Reputation"

10. Save the Data Frame

The cleaned data was saved as a CSV File

In this project, we embarked on a comprehensive data cleaning and transformation journey with the FIFA dataset. Our primary objective was to prepare the data for analysis by addressing inconsistencies, refining data types, and enhancing its usability.
In conclusion, data cleaning and transformation are foundational steps in any data analysis project.
By addressing inconsistencies, refining data types, and enhancing data quality, we have set the stage for more meaningful and insightful analyses. The clean and structured dataset is now well-equipped for advanced analytics, visualizations, and modeling.

In [ ]:

```

```