In [1]:
```
ead the muskets data with pandas
port os
port pandas as pd

w_Muskets_data =pd.read_csv('Muskets_data.csv')

leaning up the raw Muskets_data by filling in the NaN values with 0 using fillna() and saving the cleaned data to a new csv file
eaned_Muskets_data = raw_Muskets_data.fillna(0)

rrentpath= os.getcwd()
w_csv_file = 'Muskets.csv'
Combining the current path and new CSV file name
w_csv_path = os.path.join(currentpath, new_csv_file)
eaned_Muskets_data.to_csv(new_csv_path, index=False)
skets_data=pd.read_csv('Muskets.csv')
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_17280\1475768515.py:5: DtypeWarning: Columns (26,29,76) have mixed types. Specify dtyp
e option on import or set low_memory=False.
  raw_Muskets_data =pd.read_csv('Muskets_data.csv')
C:\Users\HP\AppData\Local\Temp\ipykernel_17280\1475768515.py:15: DtypeWarning: Columns (26,29,76) have mixed types. Specify dty
pe option on import or set low_memory=False.
  Muskets_data=pd.read_csv('Muskets.csv')
```

In [2]:
```
#Extract the player names from the PlayerUrl column and create a new column name Player Name from the extracts
#Using regex, I created a pattern to match the names in the PlayerUrl and update the new column with the extracted values.
import re

playerUrl= list(Muskets_data["playerUrl"])
pattern = r'/player/\d+/([^/]+)/\d+/'
PlayerName=[]
for name in playerUrl:
    match = re.search(pattern, name)
    Names= match.group(1)
    UpdatedNames=Names.replace('-',' ')
    PlayerName.append(UpdatedNames)
Muskets_data["Player Name"]=PlayerName
```

In [3]:
```
#Create a new column titled Player Status from the CONTRACT column with 3 labels ; 'Active', 'Free', 'On Loan
#I converted the contract column to a list to enable me loop through it
#created a for loop to check if Free or On Loan is attached to the item, else, It should assign it as Active and append the value
#I then created a new column called Player Status with the PlayerStatuslist as values

contract= list(Muskets_data["Contract"])
PlayerStatus=[]
for item in contract:
    if item =="Free":
        PlayerStatus.append("Free")
    elif "On Loan" in item:
        PlayerStatus.append("On Loan")
    else:
        PlayerStatus.append("Active")

Muskets_data["Player Status"]=PlayerStatus
```

In [4]:
```
#Unpack the POSITIONS column into as many columns as there are positions and assign Boolean values in the columns for each player

positions= list(Muskets_data["Positions"])
allpositions=set()

#Looping through the list to create a set of positions
for items in positions:
    allpositions.update(items.split(','))
print(allpositions)

#some positions contain space in the string, so cleaning it up with strip()
cleanedPositionsSet = {position.strip() for position in allpositions}
print(cleanedPositionsSet)

#creating new columns with the cleanedPositionSet and setting the default to false
for position in cleanedPositionsSet:
    Muskets_data[position]=False

#Looping through the Muskets_data and filling in the new columns with the boolean values based on the Positions column
for index, row in Muskets_data.iterrows():
    for position in row['Positions'].split(','):
        Muskets_data.at[index,position.strip()]=True
print(Muskets_data.columns)
print(Muskets_data['RW'][:5])
```

```
{'LB', 'RW', 'CAM', 'CM', 'CB', 'LW', 'RWB', ' RB', 'GK', 'ST', ' CF', 'RM', ' CAM', ' LM', ' CB', 'CDM', 'CF', ' LW', ' LWB',
' RM', 'LWB', ' ST', ' RWB', 'LM', ' CM', ' CDM', ' LB', ' RW', 'RB'}
{'LM', 'LB', 'RW', 'CAM', 'CF', 'CDM', 'CM', 'ST', 'RM', 'CB', 'LW', 'RWB', 'GK', 'LWB', 'RB'}
Index(['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', 'Player Name', 'Player Status', 'LM', 'LB',
       'RW', 'CAM', 'CF', 'CDM', 'CM', 'ST', 'RM', 'CB', 'LW', 'RWB', 'GK',
       'LWB', 'RB'],
      dtype='object')
0     True
1    False
```

```
2    False
3    False
4    False
Name: RW, dtype: bool
```

In [5]:
```
#Weight and Height, W/F, SM and IR Columns: convert to integers

# Since the Height column also contains inches, and Weight column lbs, Create a function to extract numeric values from the strin
def extract_numbers(value):
    numeric = ''.join(filter(str.isdigit, value))
    return int(numeric) if numeric else None

# Convert the Height column using the extract_numeric function
for i, value in enumerate(Muskets_data['Height']):
    if 'cm' in value:
        Muskets_data.loc[i,'Height'] = extract_numbers(value)
    else:
        feet, inches = value.split("'")
        feet = int(feet)
        inches = int(inches[:-1]) # Remove the trailing double quotation mark
        total_cm = feet * 30.48 + inches * 2.54  # Convert feet and inches to cm
        Muskets_data.loc[i,'Height'] = round(total_cm, 2)  # Round to 2 decimal places


# Convert the Weight column using the extract_numeric function
for i, value in enumerate(Muskets_data['Weight']):
    if 'kg' in value:
        Muskets_data.loc[i,'Weight'] = extract_numbers(value)
    else:
        lbs = extract_numbers(value)
        in_kg = lbs*0.45 # Convert lbs to kg
        int_kg = int(in_kg) # Convert to integer
        Muskets_data.loc[i,'Weight'] = round(int_kg, 2)  # Round to 2 decimal places

# Muskets_data['Weight'] = Muskets_data['Weight'].str.replace('kg', '').str.replace('lbs', '').astype(int)
#For Colums W/F, SM and IR, I replaced star symbol with an empty string and converted to integer.
Muskets_data["W/F"] = Muskets_data["W/F"].str.replace('★', '').astype(int)
Muskets_data["SM"] = Muskets_data["SM"].str.replace('★', '').astype(int)
Muskets_data["IR"] = Muskets_data["IR"].str.replace('★', '').astype(int)
```

In [6]:
```
#Value, Wage and Release Clause columns: convert to Float
#Create a function to check each value, those with K are multiplied by 1000 which is e3, M are multiplied by 1000000 which is e6
# and then converted to float

# Function to convert values with '€', 'M', or 'K' to float
def convert_to_float(value):
    if value[-1] == 'M':     #check if the last vaue is M
        return float(value[1:-1]) * 1000000
    elif value[-1] == 'K':       #check if the last vaue is K
        return float(value[1:-1]) * 1000
    elif value[0] != '€' and (value[-1] != 'K'or value[-1] != 'M'):      #check for 0 or any other number without € sign
        return float(value)
    else:
        return float(value[1:])  #If it contains neither M nor K, return the value after the € sign

# Convert the Wage, value and Release Clause columns using the convert_to_float function

for i, wage in enumerate(Muskets_data['Wage']):
    Muskets_data.loc[i,'Wage'] = convert_to_float(wage)

for i, value in enumerate(Muskets_data['Value']):
    Muskets_data.loc[i,'Value'] = convert_to_float(value)

for i, release in enumerate(Muskets_data['Release Clause']):
    Muskets_data.loc[i,'Release Clause'] = convert_to_float(release)
```

In [7]:
```
#Inspect the HITS column and ensure its float
# The Hits column contains the letter K as in thousand, which is replaced with e3 as in multiplied by 1000 when converted to floa

print('inspecting the Hits column:')
print(Muskets_data["Hits"].dtype)
Muskets_data["Hits"] = Muskets_data["Hits"].str.replace('K', 'e3').astype(float)
print(Muskets_data["Hits"].dtype)
```

```
inspecting the Hits column:
object
float64
```

In [8]:
```
5 new categorical columns for the Height, Weight, Release Clause, Value and Wage into
ou convert the respective values into clusters/labels as follows
ht: Bucket intervals of 10 years
ht: Bucket intervals of 10 kg
: bucket intervals of 50K
e: bucket intervals of 50M
ase Clause: bucket intervals of 50M

mpy as np

 a list from the respective columns using range function, including the lower number in the list by subtracting the step/interval
tervals = range((min(Muskets_data['Height'])-10), max(Muskets_data['Height']) + 11, 10)
tervals = range((min(Muskets_data['Weight'])-10), max(Muskets_data['Weight']) + 11, 10)
rvals = np.arange((min(Muskets_data['Wage'])-50), max(Muskets_data['Wage']) + 51, 50)
ervals = np.arange((min(Muskets_data['Value'])-50), max(Muskets_data['Value']) + float(51.0), float(50.0))
intervals = np.arange((min(Muskets_data['Release Clause'])-50), max(Muskets_data['Release Clause']) + float(51.0), float(50.0))

ns = list(height_intervals) # Bucket intervals of 10 cm
ns = list(weight_intervals)  # Bucket intervals of 10 kg
 =  wage_intervals # Bucket intervals of 50K
s =  value_intervals # Bucket intervals of 50M
ins = realease_intervals  # Bucket intervals of 50M

 labels for each column
bel = [f"{start}-{start+10}cm" for start in height_bins[:-1]]
bel = [f"{start}-{start+10}kg" for start in weight_bins[:-1]]
l = [f"{start}-{start+50}K" for start in wage_bins[:-1]]
el = [f"{start}-{start+50}M" for start in value_bins[:-1]]
abel = [f"{start}-{start+50}M" for start in release_bins[:-1]]

categorical columns using pd.cut()
ata['HeightCategory'] = pd.cut(Muskets_data['Height'], bins=height_bins, labels=height_label)
ata['WeightCategory'] = pd.cut(Muskets_data['Weight'], bins=weight_bins, labels=weight_label)
ata['WageCategory'] = pd.cut(Muskets_data['Wage'], bins=wage_bins, labels=wage_label)
ata['ValueCategory'] = pd.cut(Muskets_data['Value'], bins=value_bins, labels=value_label)
```

```
ata['ReleaseCategory'] = pd.cut(Muskets_data['Release Clause'], bins=release_bins, labels=release_label)

kets_data["Value"][:10],Muskets_data["Weight"][:10],Muskets_data["ValueCategory"][:10],Muskets_data["WeightCategory"][:10])
```

```
0    103500000.0
1     63000000.0
2    120000000.0
3    129000000.0
4    132000000.0
5    111000000.0
6    120500000.0
7    102000000.0
8    185500000.0
9    110000000.0
Name: Value, dtype: object 0    72
1    83
2    87
3    70
4    68
5    80
6    71
7    91
8    73
9    85
Name: Weight, dtype: object 0    103499950.0-103500000.0M
1      62999950.0-63000000.0M
2    119999950.0-120000000.0M
3    128999950.0-129000000.0M
4    131999950.0-132000000.0M
5    110999950.0-111000000.0M
6    120499950.0-120500000.0M
7    101999950.0-102000000.0M
8    185499950.0-185500000.0M
9    109999950.0-110000000.0M
Name: ValueCategory, dtype: category
Categories (3710002, object): ['-50.0-0.0M' < '0.0-50.0M' < '50.0-100.0M' < '100.0-150.0M' ... '185499850.0-185499900.0M' < '18
5499900.0-185499950.0M' < '185499950.0-185500000.0M' < '185500000.0-185500050.0M'] 0     70-80kg
1     80-90kg
2     80-90kg
3     60-70kg
4     60-70kg
5     70-80kg
6     70-80kg
7    90-100kg
8     70-80kg
9     80-90kg
Name: WeightCategory, dtype: category
Categories (13, object): ['-10-0kg' < '0-10kg' < '10-20kg' < '20-30kg' ... '80-90kg' < '90-100kg' < '100-110kg' < '110-120kg']
```

In [ ]: