

```
In [4]: 1 import pybaseball as pb
2 from pybaseball import statcast, pitching_stats, playerid_lookup, stat
3 import numpy as np
4 import math
5 import pandas as pd
6 import seaborn as sns
7 import matplotlib.pyplot as plt
8 %matplotlib inline
9 import glob
10 import os
11 import re
12 import ast
13 import unicodedata
14 from datetime import datetime
15 from itertools import groupby
16 from operator import itemgetter
17 from ast import literal_eval
```

First, organize the TJ DF

```
In [5]: 1 tj_list_df = pd.read_csv('~/Documents/Flatiron/Project_5_/data/tj_surge
2 tj_list_df.columns = tj_list_df.iloc[0]
3
4 tj_list_df = tj_list_df.drop(tj_list_df.index[0])
5
6 tj_list_df.reset_index(drop=True, inplace=True)
7
8 tj_list_df.head()
```

Out[5]:

	Donations	https://paypal.me/mlbplayeranalys?country.x=CA&locale.x=en_US	NaN	NaN	NaN	NaN	NaN
0	Player	TJ Surgery Date	Team	Level	Position	Throws	Country
1	Endy Rodriguez	12/12/2023	PIT	MLB	C	R	Dominican
2	Johan Oviedo	12/1/2023	PIT	MLB	P	R	Cuba
3	Jovani Moran	11/1/2023	MIN	AAA	P	L	Puerto Rico
4	Taylor Broadway	10/18/2023	BOS	AA	P	R	United States

5 rows × 42 columns

```
In [6]: 1 new_column_names = [
2     "Player", "TJ Surgery Date", "Team", "Level", "Position", "Throws"
3     "High School", "College(s)", "Age", "Return Date (same level)", "R
4     "mlbamid", "fgid", "Surgeon(s)", "Post-TJ MLB G", "Post-TJ MLB IP/I
5     "Year", "Month", "Day", "Started\nThrowing", "Mound", "Bullpen", "
6     "Game", "Setback", "Setback Type", "Setback", "Setback Type", "G",
7     "K-BB%", "ERA-", "FIP-", "G", "GS", "IP", "K-BB%", "ERA-", "FIP-"
8 ]
9
10 tj_list_df.columns = new_column_names
11
12 tj_list_df.head()
```

Out[6]:

	Player	TJ Surgery Date	Team	Level	Position	Throws	Country	High School	College(s)	Age
0	Player	TJ Surgery Date	Team	Level	Position	Throws	Country	High School	College(s)	Age
1	Endy Rodriguez	12/12/2023	PIT	MLB	C	R	Dominican	NaN	NaN	23
2	Johan Oviedo	12/1/2023	PIT	MLB	P	R	Cuba	NaN	NaN	25
3	Jovani Moran	11/1/2023	MIN	AAA	P	L	Puerto Rico	NaN	NaN	26
4	Taylor Broadway	10/18/2023	BOS	AA	P	R	United States	Texas	Mississippi	26

5 rows × 42 columns

```
In [7]: 1 tj_list_df = tj_list_df.drop(tj_list_df.index[0])
2 tj_list_df.reset_index(drop=True, inplace=True)
3 tj_list_df.head()
```

Out[7]:

	Player	TJ Surgery Date	Team	Level	Position	Throws	Country	High School	College(s)	Age
0	Endy Rodriguez	12/12/2023	PIT	MLB	C	R	Dominican	NaN	NaN	23
1	Johan Oviedo	12/1/2023	PIT	MLB	P	R	Cuba	NaN	NaN	25
2	Jovani Moran	11/1/2023	MIN	AAA	P	L	Puerto Rico	NaN	NaN	26
3	Taylor Broadway	10/18/2023	BOS	AA	P	R	United States	Texas	Mississippi	26
4	Felix Bautista	10/9/2023	BAL	MLB	P	R	Dominican	NaN	NaN	28

5 rows × 42 columns

```
In [8]: ⏎
1 tj_list_df_filtered = tj_list_df[tj_list_df['Position'] == 'P']
2 tj_list_df_filtered.reset_index(drop=True, inplace=True)
3 tj_list_df_filtered
```

Out[8]:

	Player	TJ Surgery Date	Team	Level	Position	Throws	Country	High School	College(s)
0	Johan Oviedo	12/1/2023	PIT	MLB	P	R	Cuba	NaN	NaN
1	Jovani Moran	11/1/2023	MIN	AAA	P	L	Puerto Rico	NaN	NaN
2	Taylor Broadway	10/18/2023	BOS	AA	P	R	United States	Texas	Mississippi
3	Felix Bautista	10/9/2023	BAL	MLB	P	R	Dominican	NaN	NaN
4	Sandy Alcantara	10/6/2023	MIA	MLB	P	R	Dominican	NaN	NaN
...
2186	Tom Candiotti	10/13/1981	MIL	AA	P	R	United States	California	St. Mary's of California
2187	Bill Bordley	4/13/1981	SF	MLB	P	L	United States	California	El Camino, USC
2188	Joe Hesketh	1/1/1981	WAS	AA	P	L	United States	New York	State of New York
2189	Brent Strom	1/1/1978	SD	MLB	P	L	United States	California	San Diego City, USC
2190	Tommy John	9/25/1974	LAD	MLB	P	L	United States	Indiana	Indiana State

2191 rows × 42 columns



```
In [9]: 1 tj_list_df_filtered['Surgeon(s)'].value_counts()
```

```
Out[9]: Surgeon(s)
Dr. James Andrews           231
Dr. Keith Meister          112
Dr. Neal ElAttrache        82
Dr. Lewis Yocum            79
Dr. David Altchek           53
Dr. Timothy Kremchek       41
Dr. Frank Jobe              32
Dr. George Paletta          21
Dr. William Raasch          14
Dr. Christopher Ahmad       13
Dr. Thomas Mehlhoff         11
Dr. Thomas Noonan           5
Dr. Koco Eaton              4
Dr. John Uribe              3
Dr. John Steubs, Dr. Pearce McCarty 3
Dr. Vincent Key             3
Dr. Michael Ciccotti         3
Dr. Dan Buss, Dr. John Steubs 2
Dr. Ken Akizuki              2
Dr. Stephen Gryzlo           2
Dr. Wiemi Douoguih          2
Dr. Gary Waslewski           2
Dr. Christopher Camp          2
Dr. Bernard Morrey           1
Dr. David Lintner            1
Dr. Stephen Lemos            1
Dr. Damon Petty              1
Dr. Darryl Kan                1
Dr. Michael Lee, Dr. Don Sheridan 1
Dr. George Caldwell            1
Dr. Jan Fronek, Dr. Heinz Hoenecke, Dr. Daniel Keefe 1
Dr. Gregory Nicholson         1
Dr. Xavier Duralde            1
Dr. Derrick Papp              1
Dr. Patrick DeMeo, Dr. Darren Frank 1
Dr. Steven Hagan              1
Dr. Michael Shepard            1
Dr. Luis Vela                  1
Dr. Donald D'Alessandro        1
Dr. Tim Griffith              1
Dr. Jeffrey Dugas              1
Dr. Michael Schafer            1
Name: count, dtype: int64
```

Frank Jobe did the first TJ surgery.

May have useful data.

In [10]: 1 tj_list_df_filtered.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2191 entries, 0 to 2190
Data columns (total 42 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Player            2191 non-null    object  
 1   TJ Surgery Date  2191 non-null    object  
 2   Team              2191 non-null    object  
 3   Level             2191 non-null    object  
 4   Position          2191 non-null    object  
 5   Throws            2191 non-null    object  
 6   Country           2191 non-null    object  
 7   High School       1741 non-null    object  
 8   College(s)        1351 non-null    object  
 9   Age               2191 non-null    object  
 10  Return Date (same level) 1746 non-null    object  
 11  Recovery Time (months) 1045 non-null    object  
 12  mlbamid          2165 non-null    object  
 13  fgid              2128 non-null    object  
 14  Surgeon(s)        741 non-null    object  
 15  Post-TJ MLB G    2191 non-null    object  
 16  Post-TJ MLB IP/PA 2191 non-null    object  
 17  Active            2191 non-null    object  
 18  Year              2191 non-null    object  
 19  Month             2191 non-null    object  
 20  Day               2191 non-null    object  
 21  Started           77 non-null     object  
 22  Mound              37 non-null     object  
 23  Bullpen            37 non-null     object  
 24  Live               53 non-null     object  
 25  Game               607 non-null    object  
 26  Setback            52 non-null    object  
 27  Setback Type      52 non-null    object  
 28  Setback            7 non-null     object  
 29  Setback Type      7 non-null     object  
 30  G                  59 non-null    object  
 31  GS                 59 non-null    object  
 32  IP                 59 non-null    object  
 33  K-BB%              59 non-null    object  
 34  ERA-               59 non-null    object  
 35  FIP-               59 non-null    object  
 36  G                  59 non-null    object  
 37  GS                 59 non-null    object  
 38  IP                 59 non-null    object  
 39  K-BB%              58 non-null    object  
 40  ERA-               58 non-null    object  
 41  FIP-               58 non-null    object  
dtypes: object(42)
memory usage: 719.1+ KB
```

```
In [11]: 1 tj_list_df_filtered.drop(['High School', 'College(s)', 'Position', 'FI
2 tj_list_df_filtered
```

C:\Users\johns\AppData\Local\Temp\ipykernel_8628\2220704903.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy ([http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy))

```
tj_list_df_filtered.drop(['High School', 'College(s)', 'Position', 'FIP
-', 'ERA-', 'K-BB%'], axis=1, inplace=True)
```

Out[11]:

	Player	TJ Surgery Date	Team	Level	Throws	Country	Age	Return Date (same level)	Recovery Time (months)	mlt
0	Johan Oviedo	12/1/2023	PIT	MLB	R	Cuba	25	NaN	NaN	6
1	Jovani Moran	11/1/2023	MIN	AAA	L	Puerto Rico	26	NaN	NaN	6
2	Taylor Broadway	10/18/2023	BOS	AA	R	United States	26	NaN	NaN	6
3	Felix Bautista	10/9/2023	BAL	MLB	R	Dominican	28	NaN	NaN	6
4	Sandy Alcantara	10/6/2023	MIA	MLB	R	Dominican	27	NaN	NaN	6
...
2186	Tom Candiotti	10/13/1981	MIL	AA	R	United States	23	1/1/1983	NaN	1
2187	Bill Bordley	4/13/1981	SF	MLB	L	United States	23	NaN	NaN	
2188	Joe Hesketh	1/1/1981	WAS	AA	L	United States	22	1/1/1983	NaN	1
2189	Brent Strom	1/1/1978	SD	MLB	L	United States	28	NaN	NaN	1
2190	Tommy John	9/25/1974	LAD	MLB	L	United States	31	4/16/1976	19	1

2191 rows × 33 columns

In [12]: 1 tj_list_df_filtered['Player'].value_counts()

Out[12]: Player

Jason Isringhausen	3
Jonny Venters	3
Ben Leeper	3
Corey Black	3
Jose Adames	2
..	
Clarke Schmidt	1
Chase Johnson	1
Shelby Miller	1
Evan Grills	1
Tommy John	1
Name: count, Length: 2045, dtype: int64	

In [13]: 1 isringhausen_df = tj_list_df_filtered[tj_list_df_filtered['Player'] ==
2 isringhausen_df

Out[13]:

	Player	TJ Surgery Date	Team	Level	Throws	Country	Age	Return Date (same level)	Recovery Time (months)	mlb
1614	Jason Isringhausen	9/1/2009	TB	MLB	R	United States	36	4/11/2011	19	11
1634	Jason Isringhausen	6/16/2009	TB	MLB	R	United States	36	4/11/2011	22	11
2087	Jason Isringhausen	1/13/1998	NYM	MLB	R	United States	25	5/24/1999	16	11

3 rows × 33 columns

In [14]: 1 mask = (tj_list_df_filtered['Player'] == 'Jason Isringhausen') & (tj_list_df_filtered['TJ Surgery Date'] == '9/1/2008')
2
3 tj_list_df_filtered.loc[mask, 'TJ Surgery Date'] = '9/1/2008'
4
5 print(isringhausen_df[['Player', 'TJ Surgery Date']])

	Player	TJ Surgery Date
1614	Jason Isringhausen	9/1/2009
1634	Jason Isringhausen	6/16/2009
2087	Jason Isringhausen	1/13/1998

C:\Users\johns\AppData\Local\Temp\ipykernel_8628\2318486528.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

tj_list_df_filtered.loc[mask, 'TJ Surgery Date'] = '9/1/2008'

In [15]: 1 tj_list_df_filtered['Game'].value_counts()

Out[15]: Game

5/5/2021	15
5/6/2021	14
5/4/2021	9
8/16/2022	6
6/9/2022	5
..	
6/2/2021	1
3/4/2021	1
6/28/2022	1
4/24/2022	1
7/2/2010	1

Name: count, Length: 401, dtype: int64

In [16]: 1 tj_list_df_filtered['Throws'].unique()

Out[16]: array(['R', 'L', 'P', 'L/R'], dtype=object)

In [17]: 1 tj_list_df_filtered['Throws'].value_counts()

Out[17]: Throws

R	1647
L	542
P	1
L/R	1

Name: count, dtype: int64

In [18]: 1 lr_tj = tj_list_df_filtered[tj_list_df_filtered['Throws'] == 'L/R']
2 lr_tj

Out[18]:

Player	TJ Surgery Date	Team	Level	Throws	Country	Age	Return Date (same level)	Recovery Time (months)	mlbamid
1464 Ryan Perez	11/27/2011	CLE	HS	L/R	United States	17	6/22/2015	NaN	664147

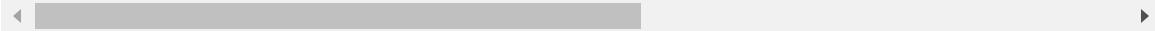
1 rows × 33 columns

```
In [19]: 1 p_tj = tj_list_df_filtered[tj_list_df_filtered['Throws'] == 'P']
2 p_tj
```

Out[19]:

	Player	TJ Surgery Date	Team	Level	Throws	Country	Age	Return Date (same level)	Recovery Time (months)	mlbamid
478	Michael Mercado	5/1/2019	TB	A-	P	United States	20	5/8/2021	24	675650

1 rows × 33 columns



P & L/R minor leagues, irrelevant. drop.

```
In [20]: 1 indices_to_drop = tj_list_df_filtered[
2     (tj_list_df_filtered['Player'] == 'Ryan Perez') |
3     (tj_list_df_filtered['Player'] == 'Michael Mercado')
4 ].index
5 tj_list_df_filtered = tj_list_df_filtered.drop(indices_to_drop)
6 tj_list_df_filtered
```

Out[20]:

	Player	TJ Surgery Date	Team	Level	Throws	Country	Age	Return Date (same level)	Recovery Time (months)	mkt
0	Johan Oviedo	12/1/2023	PIT	MLB	R	Cuba	25	NaN	NaN	6
1	Jovani Moran	11/1/2023	MIN	AAA	L	Puerto Rico	26	NaN	NaN	6
2	Taylor Broadway	10/18/2023	BOS	AA	R	United States	26	NaN	NaN	6
3	Felix Bautista	10/9/2023	BAL	MLB	R	Dominican	28	NaN	NaN	6
4	Sandy Alcantara	10/6/2023	MIA	MLB	R	Dominican	27	NaN	NaN	6
...
2186	Tom Candiotti	10/13/1981	MIL	AA	R	United States	23	1/1/1983	NaN	1
2187	Bill Bordley	4/13/1981	SF	MLB	L	United States	23	NaN	NaN	1
2188	Joe Hesketh	1/1/1981	WAS	AA	L	United States	22	1/1/1983	NaN	1
2189	Brent Strom	1/1/1978	SD	MLB	L	United States	28	NaN	NaN	1
2190	Tommy John	9/25/1974	LAD	MLB	L	United States	31	4/16/1976	19	1

2189 rows × 33 columns



Make 'Throws' a binary value.

L = 0, R = 1

```
In [21]: 1 tj_list_df_filtered['Throws'] = tj_list_df_filtered['Throws'].replace(
2 tj_list_df_filtered['Throws'] = tj_list_df_filtered['Throws'].astype(i)
3 tj_list_df_filtered.head()
```

C:\Users\johns\AppData\Local\Temp\ipykernel_8628\918451578.py:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
 tj_list_df_filtered['Throws'] = tj_list_df_filtered['Throws'].replace({'L': 0, 'R': 1})

Out[21]:

	Player	TJ Surgery Date	Team	Level	Throws	Country	Age	Return Date (same level)	Recovery Time (months)	mlbamid
0	Johan Oviedo	12/1/2023	PIT	MLB	1	Cuba	25	NaN	NaN	670912
1	Jovani Moran	11/1/2023	MIN	AAA	0	Puerto Rico	26	NaN	NaN	663558
2	Taylor Broadway	10/18/2023	BOS	AA	1	United States	26	NaN	NaN	699479
3	Felix Bautista	10/9/2023	BAL	MLB	1	Dominican	28	NaN	NaN	642585
4	Sandy Alcantara	10/6/2023	MIA	MLB	1	Dominican	27	NaN	NaN	645261

5 rows × 33 columns



In [22]:

```
1 tj_list_df_filtered = tj_list_df_filtered.rename(columns={'Player': 'Name'})
2 tj_list_df_filtered
```

Out[22]:

	Name	TJ Surgery Date	Team	Level	Throws	Country	Age	Return Date (same level)	Recovery Time (months)	mlbamid
0	Johan Oviedo	12/1/2023	PIT	MLB	1	Cuba	25	NaN	NaN	6
1	Jovani Moran	11/1/2023	MIN	AAA	0	Puerto Rico	26	NaN	NaN	60
2	Taylor Broadway	10/18/2023	BOS	AA	1	United States	26	NaN	NaN	69
3	Felix Bautista	10/9/2023	BAL	MLB	1	Dominican	28	NaN	NaN	64
4	Sandy Alcantara	10/6/2023	MIA	MLB	1	Dominican	27	NaN	NaN	64
...
2186	Tom Candiotti	10/13/1981	MIL	AA	1	United States	23	1/1/1983	NaN	1
2187	Bill Bordley	4/13/1981	SF	MLB	0	United States	23	NaN	NaN	
2188	Joe Hesketh	1/1/1981	WAS	AA	0	United States	22	1/1/1983	NaN	1
2189	Brent Strom	1/1/1978	SD	MLB	0	United States	28	NaN	NaN	12
2190	Tommy John	9/25/1974	LAD	MLB	0	United States	31	4/16/1976	19	1

2189 rows × 33 columns



In [23]:

```
1 tj_list_df_filtered.columns
```

Out[23]:

```
Index(['Name', 'TJ Surgery Date', 'Team', 'Level', 'Throws', 'Country',
       'Age',
       'Return Date (same level)', 'Recovery Time (months)', 'mlbamid',
       'fgid',
       'Surgeon(s)', 'Post-TJ MLB G', 'Post-TJ MLB IP/PA', 'Active', 'Year',
       'Month', 'Day', 'Started\nThrowing', 'Mound', 'Bullpen',
       'Live\nHitters', 'Game', 'Setback', 'Setback Type', 'Setback',
       'Setback Type', 'G', 'GS', 'IP', 'G', 'GS', 'IP'],
      dtype='object')
```

Clean names by making lowercase, removing special characters, excess whitespace, -. etc.

```
In [24]: ❶ 1 def remove_accents(input_str):
2     nfkd_form = unicodedata.normalize('NFKD', input_str)
3     return "".join([c for c in nfkd_form if not unicodedata.combining(
4
5 def clean_name(name):
6     name = name.lower()
7     name = remove_accents(name)
8     name = re.sub(r'[-.]', ' ', name)
9     name = re.sub(r'\s+', ' ', name).strip()
10    return name
11
12 tj_list_df_filtered['Name'] = tj_list_df_filtered['Name'].apply(clean_

```

```
In [25]: ❶ 1 tj_list_df_filtered['Surgery'] = tj_list_df_filtered.groupby('Name')[['
2 tj_list_df_filtered[['Name', 'TJ Surgery Date', 'Surgery']]
```

Out[25]:

	Name	TJ Surgery Date	Surgery
0	johan oviedo	12/1/2023	1
1	jovani moran	11/1/2023	1
2	taylor broadway	10/18/2023	1
3	felix bautista	10/9/2023	1
4	sandy alcantara	10/6/2023	1
...
2186	tom candiotti	10/13/1981	1
2187	bill bordley	4/13/1981	1
2188	joe hesketh	1/1/1981	1
2189	brent strom	1/1/1978	1
2190	tommy john	9/25/1974	1

2189 rows × 3 columns

In [26]:

```
1 isringhausen_df = tj_list_df_filtered[tj_list_df_filtered['Name'] == 'isringhausen']
2 isringhausen_df
```

Out[26]:

	Name	TJ Surgery Date	Team	Level	Throws	Country	Age	Return Date (same level)	Recovery Time (months)	mlb
1614	jason isringhausen	9/1/2008	TB	MLB	1	United States	36	4/11/2011	19	11
1634	jason isringhausen	6/16/2009	TB	MLB	1	United States	36	4/11/2011	22	11
2087	jason isringhausen	1/13/1998	NYM	MLB	1	United States	25	5/24/1999	16	11

3 rows × 34 columns



In [27]:

```
1 tj_list_df_filtered
```

Out[27]:

	Name	TJ Surgery Date	Team	Level	Throws	Country	Age	Return Date (same level)	Recovery Time (months)	mlb
0	johan oviedo	12/1/2023	PIT	MLB	1	Cuba	25	NaN	NaN	67
1	jovani moran	11/1/2023	MIN	AAA	0	Puerto Rico	26	NaN	NaN	66
2	taylor broadway	10/18/2023	BOS	AA	1	United States	26	NaN	NaN	69
3	felix bautista	10/9/2023	BAL	MLB	1	Dominican	28	NaN	NaN	64
4	sandy alcantara	10/6/2023	MIA	MLB	1	Dominican	27	NaN	NaN	64
...
2186	tom candiotti	10/13/1981	MIL	AA	1	United States	23	1/1/1983	NaN	1
2187	bill bordley	4/13/1981	SF	MLB	0	United States	23	NaN	NaN	
2188	joe hesketh	1/1/1981	WAS	AA	0	United States	22	1/1/1983	NaN	1
2189	brent strom	1/1/1978	SD	MLB	0	United States	28	NaN	NaN	12
2190	tommy john	9/25/1974	LAD	MLB	0	United States	31	4/16/1976	19	1

2189 rows × 34 columns



In [28]:

```

1 # Identify duplicate columns by name
2 columns_seen = {}
3 duplicates = []
4
5 for col in tj_list_df_filtered.columns:
6     if col in columns_seen:
7         duplicates.append(col)
8     else:
9         columns_seen[col] = True
10
11 # Remove duplicates - keeping the first occurrence
12 tj_list_df_filtered = tj_list_df_filtered.loc[:,~tj_list_df_filtered.columns.duplicated()]
13
14 # Now, reorder the columns as intended
15 new_column_order = ['Name', 'Surgery'] + [col for col in tj_list_df_filtered.columns if col not in new_column_order]
16 tj_list_df_filtered = tj_list_df_filtered[new_column_order]
17
18 # Check the result
19 tj_list_df_filtered.head()

```

Out[28]:

	Name	Surgery	TJ Surgery Date	Team	Level	Throws	Country	Age	Return Date (same level)	Recovery Time (months)
0	johan oviedo	1	12/1/2023	PIT	MLB	1	Cuba	25	NaN	NaN
1	iovani moran	1	11/1/2023	MIN	AAA	0	Puerto Rico	26	NaN	NaN
2	taylor broadway	1	10/18/2023	BOS	AA	1	United States	26	NaN	NaN
3	felix bautista	1	10/9/2023	BAL	MLB	1	Dominican	28	NaN	NaN
4	sandy alcantara	1	10/6/2023	MIA	MLB	1	Dominican	27	NaN	NaN

5 rows × 29 columns

In [29]:

```
1 tj_list_df_filtered['Date'] = pd.to_datetime(tj_list_df_filtered[['Year', 'Month']])
2 tj_list_df_filtered.head()
```

Out[29]:

	Name	Surgery	TJ Surgery Date	Team	Level	Throws	Country	Age	Return Date (same level)	Recovery Time (months)
0	johan oviedo	1	12/1/2023	PIT	MLB	1	Cuba	25	NaN	NaN
1	jovani moran	1	11/1/2023	MIN	AAA	0	Puerto Rico	26	NaN	NaN
2	taylor broadway	1	10/18/2023	BOS	AA	1	United States	26	NaN	NaN
3	felix bautista	1	10/9/2023	BAL	MLB	1	Dominican	28	NaN	NaN
4	sandy alcantara	1	10/6/2023	MIA	MLB	1	Dominican	27	NaN	NaN

5 rows × 30 columns



In [30]:

```
1 tj_list_df_filtered = tj_list_df_filtered.drop(columns=['Year', 'Month'])
2 tj_list_df_filtered.head()
```

Out[30]:

	Name	Surgery	TJ Surgery Date	Team	Level	Throws	Country	Age	Return Date (same level)	Recovery Time (months)
0	johan oviedo	1	12/1/2023	PIT	MLB	1	Cuba	25	NaN	NaN
1	jovani moran	1	11/1/2023	MIN	AAA	0	Puerto Rico	26	NaN	NaN
2	taylor broadway	1	10/18/2023	BOS	AA	1	United States	26	NaN	NaN
3	felix bautista	1	10/9/2023	BAL	MLB	1	Dominican	28	NaN	NaN
4	sandy alcantara	1	10/6/2023	MIA	MLB	1	Dominican	27	NaN	NaN

5 rows × 27 columns



```
In [31]: 1 position = tj_list_df_filtered.columns.get_loc('Active') + 1
2 columns = list(tj_list_df_filtered.columns)
3 columns.remove('Date')
4 columns.insert(position, 'Date')
5
6 tj_list_df_filtered = tj_list_df_filtered[columns]
7
8 tj_list_df_filtered.head()
```

Out[31]:

	Name	Surgery	TJ Surgery Date	Team	Level	Throws	Country	Age	Return Date (same level)	Recovery Time (months)
0	johan oviedo	1	12/1/2023	PIT	MLB	1	Cuba	25	NaN	NaN
1	jovani moran	1	11/1/2023	MIN	AAA	0	Puerto Rico	26	NaN	NaN
2	taylor broadway	1	10/18/2023	BOS	AA	1	United States	26	NaN	NaN
3	felix bautista	1	10/9/2023	BAL	MLB	1	Dominican	28	NaN	NaN
4	sandy alcantara	1	10/6/2023	MIA	MLB	1	Dominican	27	NaN	NaN

5 rows × 27 columns

```
In [32]: 1 tj_list_df_filtered['TJ Surgery Date'] = pd.to_datetime(tj_list_df_fil...
2 tj_list_df_filtered.head()
```

Out[32]:

	Name	Surgery	TJ Surgery Date	Team	Level	Throws	Country	Age	Return Date (same level)	Recovery Time (months)	...
0	johan oviedo	1	2023-12-01	PIT	MLB	1	Cuba	25	NaN	NaN	...
1	jovani moran	1	2023-11-01	MIN	AAA	0	Puerto Rico	26	NaN	NaN	...
2	taylor broadway	1	2023-10-18	BOS	AA	1	United States	26	NaN	NaN	...
3	felix bautista	1	2023-10-09	BAL	MLB	1	Dominican	28	NaN	NaN	...
4	sandy alcantara	1	2023-10-06	MIA	MLB	1	Dominican	27	NaN	NaN	...

5 rows × 27 columns

In [33]: 1 tj_list_df_filtered.info()

```
<class 'pandas.core.frame.DataFrame'>
Index: 2189 entries, 0 to 2190
Data columns (total 27 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Name              2189 non-null    object  
 1   Surgery           2189 non-null    int64  
 2   TJ Surgery Date  2189 non-null    datetime64[ns]
 3   Team              2189 non-null    object  
 4   Level             2189 non-null    object  
 5   Throws            2189 non-null    int32  
 6   Country           2189 non-null    object  
 7   Age               2189 non-null    object  
 8   Return Date (same level) 1744 non-null    object  
 9   Recovery Time (months) 1044 non-null    object  
 10  mlbamid          2163 non-null    object  
 11  fgid              2126 non-null    object  
 12  Surgeon(s)       741 non-null    object  
 13  Post-TJ MLB G   2189 non-null    object  
 14  Post-TJ MLB IP/PA 2189 non-null    object  
 15  Active            2189 non-null    object  
 16  Date              2189 non-null    datetime64[ns]
 17  Started           77 non-null     object  
 18  Mound             37 non-null     object  
 19  Bullpen           37 non-null     object  
 20  Live               Live Hitters  53 non-null    object  
 21  Game              606 non-null   object  
 22  Setback           52 non-null    object  
 23  Setback Type     52 non-null    object  
 24  G                 59 non-null    object  
 25  GS                59 non-null    object  
 26  IP                59 non-null    object  
dtypes: datetime64[ns](2), int32(1), int64(1), object(23)
memory usage: 470.3+ KB
```

In [34]: 1 tj_list_df_filtered.columns

```
Out[34]: Index(['Name', 'Surgery', 'TJ Surgery Date', 'Team', 'Level', 'Throws',
   'Country', 'Age', 'Return Date (same level)', 'Recovery Time (months)',
   'mlbamid', 'fgid', 'Surgeon(s)', 'Post-TJ MLB G', 'Post-TJ MLB IP/PA',
   'Active', 'Date', 'Started\nThrowing', 'Mound', 'Bullpen',
   'Live\nHitters', 'Game', 'Setback', 'Setback Type', 'G', 'GS', 'IP'],
  dtype='object')
```

```
In [35]: 1 tj_list_df_filtered['Throws'] = tj_list_df_filtered['Throws'].astype('int64')
2 tj_list_df_filtered.info()

<class 'pandas.core.frame.DataFrame'>
Index: 2189 entries, 0 to 2190
Data columns (total 27 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Name              2189 non-null    object  
 1   Surgery           2189 non-null    int64  
 2   TJ Surgery Date  2189 non-null    datetime64[ns]
 3   Team              2189 non-null    object  
 4   Level             2189 non-null    object  
 5   Throws            2189 non-null    int64  
 6   Country           2189 non-null    object  
 7   Age               2189 non-null    object  
 8   Return Date (same level) 1744 non-null    object  
 9   Recovery Time (months) 1044 non-null    object  
 10  mlbamid          2163 non-null    object  
 11  fgid              2126 non-null    object  
 12  Surgeon(s)        741 non-null    object  
 13  Post-TJ MLB G    2189 non-null    object  
 14  Post-TJ MLB IP/PA 2189 non-null    object  
 15  Active            2189 non-null    object  
 16  Date              2189 non-null    datetime64[ns]
 17  Started           77 non-null     object  
 18  Mound              37 non-null     object  
 19  Bullpen           37 non-null     object  
 20  Live               53 non-null     object  
 21  Game              606 non-null    object  
 22  Setback            52 non-null     object  
 23  Setback Type      52 non-null     object  
 24  G                 59 non-null     object  
 25  GS                59 non-null     object  
 26  IP                59 non-null     object  
dtypes: datetime64[ns](2), int64(2), object(23)
memory usage: 478.8+ KB
```

```
In [36]: 1 tj_list_df_filtered['Age'] = tj_list_df_filtered['Age'].astype('int64')
2 tj_list_df_filtered.info()

<class 'pandas.core.frame.DataFrame'>
Index: 2189 entries, 0 to 2190
Data columns (total 27 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Name             2189 non-null    object  
 1   Surgery          2189 non-null    int64  
 2   TJ Surgery Date 2189 non-null    datetime64[ns]
 3   Team             2189 non-null    object  
 4   Level            2189 non-null    object  
 5   Throws           2189 non-null    int64  
 6   Country          2189 non-null    object  
 7   Age              2189 non-null    int64  
 8   Return Date (same level) 1744 non-null    object  
 9   Recovery Time (months) 1044 non-null    object  
 10  mlbamid         2163 non-null    object  
 11  fgid            2126 non-null    object  
 12  Surgeon(s)      741 non-null    object  
 13  Post-TJ MLB G  2189 non-null    object  
 14  Post-TJ MLB IP/PA 2189 non-null    object  
 15  Active           2189 non-null    object  
 16  Date             2189 non-null    datetime64[ns]
 17  Started          77 non-null     object  
 18  Mound            37 non-null     object  
 19  Bullpen          37 non-null     object  
 20  Live             53 non-null     object  
 21  Game             606 non-null   object  
 22  Setback          52 non-null    object  
 23  Setback Type    52 non-null    object  
 24  G                59 non-null    object  
 25  GS               59 non-null    object  
 26  IP               59 non-null    object  
dtypes: datetime64[ns](2), int64(3), object(22)
memory usage: 478.8+ KB
```

```
In [37]: # Group by 'Name' and aggregate 'TJ Surgery Date' into a list
          surgery_dates_by_player = tj_list_df_filtered.groupby('Name')[['TJ Surgery Date']].agg(list).reset_index()
          # Filter players with multiple surgeries
          multiple_surgeries = surgery_dates_by_player[surgery_dates_by_player['TJ Surgery Date'].apply(len) > 1]
          # Display the DataFrame with players who have had multiple surgeries
          print(multiple_surgeries)
          1   Name
          2   TJ Surgery Date
          3
          4
          5
          6
          7
          8
          9
```

	Name	TJ Surgery Date
16	adam choplick	[2011-01-01 00:00:00, 2010-04-12 00:00:00]
22	adam parks	[2016-06-23 00:00:00, 2012-01-01 00:00:00]
44	al reyes	[2005-10-18 00:00:00, 1995-08-01 00:00:00]
73	anderson espinoza	[2019-04-22 00:00:00, 2017-07-31 00:00:00]
93	andrew lee	[2016-07-29 00:00:00, 2012-01-01 00:00:00]
...
1969	victor zambrano	[2006-05-15 00:00:00, 1996-01-01 00:00:00]
1975	walker buehler	[2022-08-23 00:00:00, 2015-08-05 00:00:00]
1976	walker powell	[2016-07-01 00:00:00, 2014-01-01 00:00:00]
1986	will childers	[2022-01-01 00:00:00, 2020-01-01 00:00:00]
2015	yosmer leal	[2017-06-06 00:00:00, 2015-05-28 00:00:00]

[142 rows x 2 columns]

```
In [38]: # Create a DataFrame with lists of surgery dates for each player
          surgery_dates_by_player = tj_list_df_filtered.groupby('Name')[['TJ Surgery Date']].agg(list).reset_index()
          # Merge this information back to the original DataFrame
          tj_list_df_filtered = pd.merge(tj_list_df_filtered, surgery_dates_by_p
          1   # Create a DataFrame with lists of surgery dates for each player
          2   surgery_dates_by_player = tj_list_df_filtered.groupby('Name')[['TJ Surgery Date']].agg(list).reset_index()
          3
          4   # Merge this information back to the original DataFrame
          5   tj_list_df_filtered = pd.merge(tj_list_df_filtered, surgery_dates_by_p
          6
          7
          8
          9
```

In [39]: 1 tj_list_df_filtered

Out[39]:

	Name	Surgery	TJ Surgery Date_x	Team	Level	Throws	Country	Age	Return Date (same level)	Recove Tin (month)
0	johan oviedo	1	2023-12-01	PIT	MLB	1	Cuba	25	NaN	NaN
1	jovani moran	1	2023-11-01	MIN	AAA	0	Puerto Rico	26	NaN	NaN
2	taylor broadway	1	2023-10-18	BOS	AA	1	United States	26	NaN	NaN
3	felix bautista	1	2023-10-09	BAL	MLB	1	Dominican	28	NaN	NaN
4	sandy alcantara	1	2023-10-06	MIA	MLB	1	Dominican	27	NaN	NaN
...
2184	tom candiotti	1	1981-10-13	MIL	AA	1	United States	23	1/1/1983	NaN
2185	bill bordley	1	1981-04-13	SF	MLB	0	United States	23	NaN	NaN
2186	joe hesketh	1	1981-01-01	WAS	AA	0	United States	22	1/1/1983	NaN
2187	brent strom	1	1978-01-01	SD	MLB	0	United States	28	NaN	NaN
2188	tommy john	1	1974-09-25	LAD	MLB	0	United States	31	4/16/1976	...

2189 rows × 28 columns



In [40]: 1 tj_list_df_filtered.drop(columns=['TJ_Surgery Date_x'], inplace=True)
2 tj_list_df_filtered.rename(columns={'TJ_Surgery Date_y': 'TJ_Surgery D

```
In [41]: 1 isringhausen_df = tj_list_df_filtered[tj_list_df_filtered['Name'] == 'isringhausen']
2 isringhausen_df
```

Out[41]:

		Name	Surgery	Team	Level	Throws	Country	Age	Return Date (same level)	Recovery Time	mlba (months)
1612	jason isringhausen		3	TB	MLB	1	United States	36	4/11/2011	19	116
1632	jason isringhausen		3	TB	MLB	1	United States	36	4/11/2011	22	116
2085	jason isringhausen		3	NYM	MLB	1	United States	25	5/24/1999	16	116

3 rows × 27 columns

```
In [42]: 1 clean_tj_df = tj_list_df_filtered
```

```
In [43]: 1 clean_tj_df['Surgery'].value_counts()
```

Out[43]: Surgery

1	1901
2	276
3	12

Name: count, dtype: int64

```
In [44]: 1 instances_with_surgery_3 = clean_tj_df[clean_tj_df['Surgery'] == 3]['Name']
          2 instances_with_surgery_3
```

```
Out[44]: 61           ben leeper
        482          corey black
        775          corey black
        933          ben leeper
       1126         jonny venters
       1149          ben leeper
       1308         jonny venters
       1523          corey black
       1612      jason isringhausen
       1632      jason isringhausen
       1862         jonny venters
       2085      jason isringhausen
Name: Name, dtype: object
```

```
In [113]: 1 clean_tj_df.to_csv('data/clean_tj_df.csv')
```

Will need to drop more eventually.

Now, download csv of each season from 1972-2023 from baseball reference, drop irrelevant columns, and join

```
In [46]: 1  directory_path = os.path.expanduser('~/Documents/Flatiron/Project_5')
2  file_pattern = os.path.join(directory_path, '*_pitch_stats.csv')
3
4  pitch_csv_files = glob.glob(file_pattern)
5
6  dataframes = []
7  for file in pitch_csv_files:
8      year = os.path.basename(file).split('_')[0]
9      df = pd.read_csv(file)
10     df['Year'] = year
11     dataframes.append(df)
12
13 if dataframes:
14     combined_df = pd.concat(dataframes, ignore_index=True)
15     print(combined_df.head())
16 else:
17     print("No CSV files were found.")
```

Rk	Name	Age	Tm	IP	G	GS	Wgs	Lgs	ND	...	RS/IP
0	Ed Acosta	28	SDP	89.0	46	2	1	1	0	...	5.4
1	Doyle Alexander	21	BAL	106.1	35	9	2	4	3	...	2.8
2	Lloyd Allen	22	CAL	85.1	42	6	0	4	2	...	0.9
3	Steve Arlin	26	SDP	250.0	38	37	9	21	7	...	2.9
4	Stan Bahnsen	27	CHW	252.1	43	41	21	15	5	...	3.6

IP/GS	Pit/GS	<80	80-99	100-119	≥120	Max	Name-additional	Year
7.0	NaN	NaN	NaN	NaN	NaN	NaN	acosted01	1972
6.8	NaN	NaN	NaN	NaN	NaN	NaN	alexado01	1972
4.8	NaN	NaN	NaN	NaN	NaN	NaN	allenll01	1972
6.7	NaN	NaN	NaN	NaN	NaN	NaN	arlinst01	1972
6.1	NaN	NaN	NaN	NaN	NaN	NaN	bahnsst01	1972

[5 rows x 39 columns]

```
In [47]: 1  combined_df.head()
```

Out[47]:

Rk	Name	Age	Tm	IP	G	GS	Wgs	Lgs	ND	...	RS/IP	IP/GS	Pit/GS
0	Ed Acosta	28	SDP	89.0	46	2	1	1	0	...	5.4	7.0	NaN
1	Doyle Alexander	21	BAL	106.1	35	9	2	4	3	...	2.8	6.8	NaN
2	Lloyd Allen	22	CAL	85.1	42	6	0	4	2	...	0.9	4.8	NaN
3	Steve Arlin	26	SDP	250.0	38	37	9	21	7	...	2.9	6.7	NaN
4	Stan Bahnsen	27	CHW	252.1	43	41	21	15	5	...	3.6	6.1	NaN

5 rows × 39 columns

In [48]: 1 combined_df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16573 entries, 0 to 16572
Data columns (total 39 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Rk               16573 non-null   int64  
 1   Name              16573 non-null   object  
 2   Age               16573 non-null   int64  
 3   Tm                16573 non-null   object  
 4   IP                16573 non-null   float64 
 5   G                 16573 non-null   int64  
 6   GS                16573 non-null   int64  
 7   Wgs               16573 non-null   int64  
 8   Lgs               16573 non-null   int64  
 9   ND                16573 non-null   int64  
 10  Wchp              16573 non-null   int64  
 11  Ltuf              16573 non-null   int64  
 12  Wtm               16566 non-null   float64 
 13  Ltm               16566 non-null   float64 
 14  tmW-L%            16564 non-null   float64 
 15  Wlst              16573 non-null   int64  
 16  Lsv               16573 non-null   int64  
 17  CG                16573 non-null   int64  
 18  SHO               16573 non-null   int64  
 19  QS                16573 non-null   int64  
 20  QS%               16573 non-null   object  
 21  GmScA              16112 non-null   float64 
 22  Best              16573 non-null   int64  
 23  Wrst              16573 non-null   int64  
 24  BQR               16573 non-null   int64  
 25  BQS               16573 non-null   int64  
 26  sDR               16573 non-null   int64  
 27  lDR               16573 non-null   int64  
 28  RS/GS              16566 non-null   float64 
 29  RS/IP              16557 non-null   float64 
 30  IP/GS              16573 non-null   float64  
 31  Pit/GS             12223 non-null   float64 
 32  <80                12235 non-null   float64 
 33  80-99               12235 non-null   float64 
 34  100-119              12235 non-null   float64 
 35  ≥120                12235 non-null   float64 
 36  Max                12235 non-null   float64 
 37  Name-additional      16573 non-null   object  
 38  Year                16573 non-null   object  
dtypes: float64(14), int64(20), object(5)
memory usage: 4.9+ MB
```

In [49]:

```
1 pitch_combined_df = combined_df.drop(['RK', 'Wgs', 'Lgs', 'ND', 'Wchp'])
2 pitch_combined_df
```

Out[49]:

		Name	Age	Tm	IP	G	GS	CG	SHO	sDR	IDR	IP/GS	Pit/GS	<8I
0		Ed Acosta	28	SDP	89.0	46	2	0	0	1	1	7.0	NaN	NaN
1		Doyle Alexander	21	BAL	106.1	35	9	2	2	3	4	6.8	NaN	NaN
2		Lloyd Allen	22	CAL	85.1	42	6	0	0	5	0	4.8	NaN	NaN
3		Steve Arlin	26	SDP	250.0	38	37	12	3	22	7	6.7	NaN	NaN
4		Stan Bahnsen	27	CHW	252.1	43	41	5	1	32	4	6.1	NaN	NaN
...	
16568		Ryan Yarbrough*	31	TOT	89.2	25	9	0	0	0	5	5.1	79.0	6.0
16569		Ryan Yarbrough*	31	KCR	51.0	14	7	0	0	0	4	5.4	80.0	4.0
16570		Ryan Yarbrough*	31	LAD	38.2	11	2	0	0	0	1	4.0	75.0	2.0
16571		Rob Zastryzny*	31	PIT	20.2	21	1	0	0	0	1	1.0	25.0	1.0
16572		Angel Zerpa*	23	KCR	42.2	15	3	0	0	1	1	4.2	77.0	1.0

16573 rows × 18 columns

In [50]:

```
1 pitch_combined_df['Name'].value_counts()
```

Out[50]:

Name	
Terry Mulholland*	29
Jamie Moyer*	29
Danny Darwin	28
Mike Morgan	27
Greg Maddux	27
..	
Jason Phillips	1
Dave Pember	1
Josh Pearce	1
Jimmy Osting*	1
Bryan Woo	1
Name: count, Length: 3451, dtype: int64	

Add a 'Throws' column to match TJ DF

```
In [51]: 1 pitch_combined_df['Throws'] = np.where(pitch_combined_df['Name'].str.e
2 pitch_combined_df
```

Out[51]:

	Name	Age	Tm	IP	G	GS	CG	SHO	sDR	IDR	IP/GS	Pit/GS	<80
0	Ed Acosta	28	SDP	89.0	46	2	0	0	1	1	7.0	NaN	NaN
1	Doyle Alexander	21	BAL	106.1	35	9	2	2	3	4	6.8	NaN	NaN
2	Lloyd Allen	22	CAL	85.1	42	6	0	0	5	0	4.8	NaN	NaN
3	Steve Arlin	26	SDP	250.0	38	37	12	3	22	7	6.7	NaN	NaN
4	Stan Bahnsen	27	CHW	252.1	43	41	5	1	32	4	6.1	NaN	NaN
...
16568	Ryan Yarbrough*	31	TOT	89.2	25	9	0	0	0	5	5.1	79.0	6.0
16569	Ryan Yarbrough*	31	KCR	51.0	14	7	0	0	0	4	5.4	80.0	4.0
16570	Ryan Yarbrough*	31	LAD	38.2	11	2	0	0	0	1	4.0	75.0	2.0
16571	Rob Zastryzny*	31	PIT	20.2	21	1	0	0	0	1	1.0	25.0	1.0
16572	Angel Zerpa*	23	KCR	42.2	15	3	0	0	1	1	4.2	77.0	1.0

16573 rows × 19 columns

```
In [52]: 1 new_column_order = [
2     'Name', 'Age', 'Year', 'Throws', 'Tm', 'IP', 'G', 'GS', 'CG', 'SHO',
3     'IP/GS', 'Pit/GS', '<80', '80-99', '100-119', '≥120', 'Max'
4 ]
5
6 pitch_combined_df = pitch_combined_df[new_column_order]
7
8 pitch_combined_df.head()
```

Out[52]:

	Name	Age	Year	Throws	Tm	IP	G	GS	CG	SHO	sDR	IDR	IP/GS	F
0	Ed Acosta	28	1972		1	SDP	89.0	46	2	0	0	1	1	7.0
1	Doyle Alexander	21	1972		1	BAL	106.1	35	9	2	2	3	4	6.8
2	Lloyd Allen	22	1972		1	CAL	85.1	42	6	0	0	5	0	4.8
3	Steve Arlin	26	1972		1	SDP	250.0	38	37	12	3	22	7	6.7
4	Stan Bahnsen	27	1972		1	CHW	252.1	43	41	5	1	32	4	6.1

In [53]: 1 pitch_combined_df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16573 entries, 0 to 16572
Data columns (total 19 columns):
 #   Column    Non-Null Count  Dtype  
--- 
 0   Name       16573 non-null   object  
 1   Age        16573 non-null   int64  
 2   Year       16573 non-null   object  
 3   Throws     16573 non-null   int32  
 4   Tm         16573 non-null   object  
 5   IP          16573 non-null   float64 
 6   G           16573 non-null   int64  
 7   GS          16573 non-null   int64  
 8   CG          16573 non-null   int64  
 9   SHO         16573 non-null   int64  
 10  SDR         16573 non-null   int64  
 11  LDR         16573 non-null   int64  
 12  IP/GS       16573 non-null   float64 
 13  Pit/GS      12223 non-null   float64 
 14  <80         12235 non-null   float64 
 15  80-99        12235 non-null   float64 
 16  100-119      12235 non-null   float64 
 17  ≥120         12235 non-null   float64 
 18  Max          12235 non-null   float64 
dtypes: float64(8), int32(1), int64(7), object(3)
memory usage: 2.3+ MB
```

```
In [54]: 1 pitch_combined_df['Tm'].value_counts()
```

```
Out[54]: Tm
TOT      1214
TEX       610
SDP       572
NYY       572
DET       570
KCR       568
CLE       556
OAK       553
BOS       551
NYM       550
CIN       549
PHI       541
PIT       540
CHC       536
BAL       535
MIL       531
MIN       528
CHW       528
STL       524
SEA       522
SFG       518
ATL       518
LAD       509
TOR       501
HOU       500
COL       347
MON       337
ARI       272
CAL       257
LAA       223
WSN       211
FLA       196
TBR       179
MIA       155
TBD       119
ANA       81
Name: count, dtype: int64
```

LAA, CAL, ANA - Angels. Combine as ANA

TBR & TBD - Rays. Combine as TBR

FLA & MIA - Marlins. Combine as MIA

Combine MON and WSN as WSN

TOT is total for one year (multiple teams). Drop all other instances in same year, replace TOT with split of all teams played.

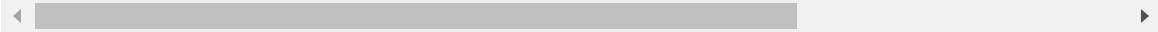
In [55]:

```
1 pitch_combined_df['Tm'] = pitch_combined_df['Tm'].replace(['LAA', 'CAL'])
2 pitch_combined_df['Tm'] = pitch_combined_df['Tm'].replace(['TBD'], 'TB')
3 pitch_combined_df['Tm'] = pitch_combined_df['Tm'].replace(['FLA'], 'MI')
4 pitch_combined_df['Tm'] = pitch_combined_df['Tm'].replace(['MON'], 'WS')
5
6 pitch_combined_df
```

Out[55]:

	Name	Age	Year	Throws	Tm	IP	G	GS	CG	SHO	sDR	IDR	IP/C
0	Ed Acosta	28	1972	1	SDP	89.0	46	2	0	0	1	1	-
1	Doyle Alexander	21	1972	1	BAL	106.1	35	9	2	2	3	4	€
2	Lloyd Allen	22	1972	1	ANA	85.1	42	6	0	0	5	0	€
3	Steve Arlin	26	1972	1	SDP	250.0	38	37	12	3	22	7	€
4	Stan Bahnsen	27	1972	1	CHW	252.1	43	41	5	1	32	4	€
...
16568	Ryan Yarbrough*	31	2023	0	TOT	89.2	25	9	0	0	0	5	€
16569	Ryan Yarbrough*	31	2023	0	KCR	51.0	14	7	0	0	0	4	€
16570	Ryan Yarbrough*	31	2023	0	LAD	38.2	11	2	0	0	0	1	€
16571	Rob Zastryzny*	31	2023	0	PIT	20.2	21	1	0	0	0	1	-
16572	Angel Zerpa*	23	2023	0	KCR	42.2	15	3	0	0	1	1	€

16573 rows × 19 columns



```
In [56]: 1 pitch_combined_df['Tm'].value_counts()
```

Out[56]:

Tm	
TOT	1214
TEX	610
SDP	572
NYY	572
DET	570
KCR	568
ANA	561
CLE	556
OAK	553
BOS	551
NYM	550
CIN	549
WSN	548
PHI	541
PIT	540
CHC	536
BAL	535
MIL	531
CHW	528
MIN	528
STL	524
SEA	522
SFG	518
ATL	518
LAD	509
TOR	501
HOU	500
MIA	351
COL	347
TBR	298
ARI	272

Name: count, dtype: int64

```
In [57]: 1 tot_df = pitch_combined_df[pitch_combined_df['Tm'] == 'TOT']
2 tot_df
```

Out[57]:

	Name	Age	Year	Throws	Tm	IP	G	GS	CG	SHO	sDR	IDR	IP
5	Steve Barber*	34	1972	0	TOT	73.2	39	3	0	0	1	0	
12	Wade Blasingame*	28	1972	0	TOT	25.1	22	1	0	0	0	0	1
43	Casey Cox	30	1972	1	TOT	77.0	40	5	0	0	3	1	
48	John Cumberland*	25	1972	0	TOT	46.2	23	7	0	0	3	4	
63	Eddie Fisher	35	1972	1	TOT	103.2	49	5	0	0	3	2	
...
16526	Justin Verlander	40	2023	1	TOT	162.1	27	27	0	0	0	0	18
16540	Ryan Weathers*	23	2023	0	TOT	57.2	15	12	0	0	0	0	10
16543	Luke Weaver	29	2023	1	TOT	123.2	29	25	0	0	1	1	13
16544	Luke Weaver	29	2023	1	TOT	26.2	8	4	0	0	1	1	3
16568	Ryan Yarbrough*	31	2023	0	TOT	89.2	25	9	0	0	0	0	5

1214 rows × 19 columns



In [58]:

```
1 verlander_df = pitch_combined_df[pitch_combined_df['Name'].str.contains('Verlander')]
2 verlander_df
```

Out[58]:

		Name	Age	Year	Throws	Tm	IP	G	GS	CG	SHO	sDR	IDR	IP/G
10017		Justin Verlander	22	2005	1	DET	11.1	2	2	0	0	0	2	5.
10367		Justin Verlander	23	2006	1	DET	186.0	30	30	1	1	0	14	6.
10712		Justin Verlander	24	2007	1	DET	201.2	32	32	1	1	0	18	6.
11060		Justin Verlander	25	2008	1	DET	201.0	33	33	1	0	0	17	6.
11407		Justin Verlander	26	2009	1	DET	240.0	35	35	3	1	0	12	6.
11715		Justin Verlander	27	2010	1	DET	224.1	33	33	4	0	0	16	6.
12011		Justin Verlander	28	2011	1	DET	251.0	34	34	4	2	0	13	7.
12336		Justin Verlander	29	2012	1	DET	238.1	33	33	6	1	0	13	7.
12665		Justin Verlander	30	2013	1	DET	218.1	34	34	0	0	0	16	6.
13005		Justin Verlander	31	2014	1	DET	206.0	32	32	0	0	0	14	6.
13350		Justin Verlander	32	2015	1	DET	133.1	20	20	1	1	0	9	6.
13723		Justin Verlander	33	2016	1	DET	227.2	34	34	2	0	0	14	6.
14090		Justin Verlander	34	2017	1	TOT	206.0	33	33	0	0	0	15	6.
14091		Justin Verlander	34	2017	1	DET	172.0	28	28	0	0	0	13	6.
14092		Justin Verlander	34	2017	1	HOU	34.0	5	5	0	0	0	2	6.
14486		Justin Verlander	35	2018	1	HOU	214.0	34	34	1	1	0	17	6.
14907		Justin Verlander	36	2019	1	HOU	223.0	34	34	2	1	0	16	6.
15227		Justin Verlander	37	2020	1	HOU	6.0	1	1	0	0	0	1	6.
16087		Justin Verlander	39	2022	1	HOU	175.0	28	28	0	0	0	23	6.
16526		Justin Verlander	40	2023	1	TOT	162.1	27	27	0	0	0	18	6.
16527		Justin Verlander	40	2023	1	NYM	94.1	16	16	0	0	0	12	5.
16528		Justin Verlander	40	2023	1	HOU	68.0	11	11	0	0	0	6	6.

In [59]:

```
1 # Define the function to replace 'TOT' with actual teams
2 def replace_tot_with_teams(row, df):
3     if row['Tm'] != 'TOT':
4         return row['Tm']
5
6     player_teams = df[(df['Name'] == row['Name']) & (df['Year'] == row['Year'])]
7     return '/'.join(player_teams)
8
9 # Apply the function to each row
10 pitch_combined_df['Tm'] = pitch_combined_df.apply(lambda row: replace_
11
12 # Display the updated DataFrame for Justin Verlander
13 pitch_combined_df[pitch_combined_df['Name'] == 'Justin Verlander'][['N
14
```

Out[59]:

Name	Year	Tm
------	------	----

In [60]: 1 pitch_combined_df['Tm'].unique()

```
Out[60]: array(['SDP', 'BAL', 'ANA', 'CHW', 'SFG', 'NYY', 'MIL', 'STL', 'CIN',
   'PIT', 'OAK', 'MIN', 'CHC', 'TEX', 'PHI', 'KCR', 'CLE', 'DET',
   'NYM', 'HOU', 'TEX/NYY', 'BOS', 'SFG/STL', 'LAD', 'ANA/CHW', 'AT
   L',
   'PHI/DET', 'WSN', 'OAK/ATL', 'ATL/PHI', 'SDP/CHC', 'STL/TEX',
   'TEX/CLE', 'ATL/NYY', 'CLE/TEX', 'TEX/ANA', 'MIN/CHW', 'NYY/CLE',
   'SFG/NYY', 'CIN/CHW', 'SDP/CIN', 'TEX/CHC', 'DET/CLE', 'SDP/CLE',
   'MIN/TEX', 'ANA/NYY', 'HOU/STL', 'STL/ATL', 'CLE/NYY', 'MIN/NYY',
   'CHW/OAK', 'CLE/OAK', 'KCR/ATL', 'OAK/CHW', 'ATL/CLE', 'CHC/LAD',
   'OAK/CLE/ATL', 'ATL/STL', 'SDP/OAK', 'BAL/NYY', 'DET/CHC',
   'HOU/SDP', 'ANA/KCR', 'NYY/BAL', 'WSN/CHC', 'TEX/MIN', 'SEA',
   'CIN/WSN', 'OAK/WSN', 'CHW/ANA', 'TOR', 'STL/SDP', 'NYY/OAK/TEX',
   'TOR/ATL', 'OAK/SEA/NYM', 'OAK/SEA', 'CIN/ANA', 'CHC/CHW',
   'NYM/CIN', 'OAK/NYY', 'PHI/WSN', 'OAK/TEX', 'CIN/NYM', 'KCR/SEA',
   'CHC/WSN', 'KCR/BOS', 'NYY/CHC', 'SFG/CHC', 'TEX/NYM/PIT',
   'TEX/CHW', 'CHW/SEA', 'PHI/NYY', 'SEA/MIL', 'SFG/PIT', 'NYM/SEA',
   'PIT/SFG', 'NYY/TEX', 'WSN/CLE', 'SFG/ANA', 'LAD/TEX', 'TOR/PIT',
   'TOR/ANA', 'KCR/TOR', 'CHC/NYY', 'NYY/OAK', 'CLE/PHI', 'LAD/ANA',
   'ATL/OAK', 'NYY/ANA', 'MIL/WSN', 'ATL/PIT', 'TEX/MIL', 'NYY/MIN',
   'SFG/WSN', 'HOU/MIL', 'SEA/DET', 'NYY/TOR', 'NYM/STL', 'CLE/ATL',
   'PHI/OAK', 'TEX/LAD', 'SDP/NYY', 'SEA/KCR', 'PHI/CHC', 'SDP/WSN',
   'BOS/CHC', 'WSN/CIN', 'CHC/TEX', 'NYY/PHI', 'CHC/CLE', 'CLE/CHC',
   'CLE/MIN', 'ANA/OAK', 'ANA/PIT', 'CHW/BOS', 'WSN/DET', 'HOU/NYY',
   'OAK/ANA', 'TEX/DET', 'BOS/SEA', 'MIN/CLE', 'PHI/SFG/CHW',
   'MIL/HOU', 'PIT/CHW', 'ATL/TOR', 'CHC/HOU', 'DET/SDP', 'SDP/DET',
   'NYY/SDP', 'ATL/DET', 'CHW/NYY', 'ANA/NYM', 'SDP/SFG', 'BAL/TOR',
   'SFG/SDP', 'CIN/NYY', 'LAD/OAK', 'STL/CHW', 'CLE/TOR/ATL',
   'NYY/CIN', 'ATL/CHC', 'BAL/BOS', 'STL/HOU', 'LAD/CHW', 'CHW/PIT',
   'KCR/DET', 'CIN/SDP', 'STL/LAD', 'SFG/PHI', 'NYY/CHW', 'PIT/SEA',
   'WSN/SEA', 'HOU/SFG', 'SEA/WSN', 'LAD/CIN', 'PHI/KCR', 'SFG/BOS',
   'CHW/MIL', 'MIN/NYM', 'NYM/MIN', 'CLE/TOR', 'MIN/TOR', 'PHI/LAD',
   'PIT/CHC', 'ATL/SDP', 'NYY/DET', 'CIN/MIL', 'WSN/PIT', 'SFG/SEA',
   'PIT/DET', 'TOR/CLE', 'WSN/TEX', 'NYM/WSN/OAK', 'PIT/TEX',
   'WSN/ATL', 'BOS/CIN', 'NYM/TOR', 'STL/PHI', 'SFG/OAK', 'WSN/KCR',
   'CHC/PIT', 'MIN/WSN', 'NYY/SEA', 'SDP/BAL', 'BAL/CLE', 'CHC/KCR',
   'ANA/PHI', 'CHC/TEX/PIT', 'TEX/OAK', 'KCR/NYY', 'MIA', 'COL/SDP',
   'COL', 'WSN/COL', 'SFG/MIN', 'MIN/SFG', 'SDP/COL', 'COL/WSN',
   'STL/ANA', 'TEX/MIA', 'ANA/SFG', 'NYM/NYY', 'PHI/SEA', 'CLE/BOS',
   'LAD/MIA', 'SDP/SEA', 'TOR/NYY', 'TOR/TEX', 'MIN/BAL', 'BAL/PHI',
   'STL/CLE', 'NYM/KCR', 'BAL/MIN', 'CHC/STL', 'ATL/BOS', 'CIN/DET',
   'SFG/CIN', 'BOS/MIN', 'NYM/COL', 'MIN/LAD', 'DET/CIN', 'MIA/TEX',
   'DET/MIN', 'CHW/TOR', 'MIL/NYY', 'PIT/HOU', 'COL/CHW', 'NYY/MIA',
   'STL/CIN', 'PIT/ATL', 'WSN/SFG', 'HOU/CLE', 'STL/DET', 'OAK/DET',
   'MIA/NYY', 'CHW/SFG', 'LAD/COL', 'CIN/KCR', 'CHC/COL', 'NYM/CHC',
   'NYM/MIL', 'MIN/DET', 'BAL/WSN', 'DET/SEA', 'CHC/SFG', 'MIA/SFG',
   'CIN/CLE', 'SDP/STL', 'ARI', 'TBR', 'CHW/CIN', 'ARI/NYM',
   'NYM/LAD', 'TOR/BAL', 'SEA/HOU', 'MIL/ANA', 'CLE/CIN', 'MIN/CHC',
   'LAD/NYM', 'TEX/STL', 'WSN/LAD', 'HOU/BOS', 'ARI/KCR', 'TEX/PIT',
   'KCR/OAK', 'OAK/CLE', 'SEA/TEX', 'DET/BOS', 'BAL/CIN', 'STL/BOS',
   'CHC/ATL', 'OAK/NYM', 'PHI/TOR', 'OAK/KCR', 'COL/BOS', 'PHI/ATL',
   'MIL/CLE', 'ARI/PHI', 'HOU/TBR/NYY', 'COL/ANA', 'TEX/TOR',
   'SDP/KCR', 'CLE/MIL', 'BOS/COL', 'PHI/ARI', 'CHW/TBR', 'TBR/TOR',
   'KCR/SDP', 'COL/HOU', 'CHW/LAD', 'CIN/TEX', 'ARI/TBR', 'PHI/NYM',
   'HOU/COL', 'TOR/CIN', 'TBR/TEX', 'HOU/DET', 'TBR/ARI', 'HOU/PIT',
   'DET/HOU', 'PIT/LAD', 'BOS/WSN', 'KCR/COL/MIL', 'CLE/WSN',
   'MIA/CIN', 'CLE/STL', 'SDP/NYM', 'WSN/MIA', 'COL/NYM', 'TEX/SEA',
   'DET/NYY', 'MIL/STL', 'CLE/KCR', 'ANA/TEX', 'TEX/TOR/MIL'],
  dtype='|S128')
```

```
'OAK/CIN', 'NYY/STL', 'ARI/BOS', 'SFG/BAL', 'SDP/PIT', 'BAL/SFG',  
'PHI/SDP', 'PIT/BOS', 'TEX/CIN', 'TBR/PHI', 'PIT/NYM', 'ARI/LAD',  
'COL/CLE', 'NYM/TEX', 'SEA/CHW', 'PHI/CIN', 'CIN/PHI', 'MIA/HOU',  
'MIA/LAD', 'CHW/WSN', 'SDP/MIA', 'TBR/NYM', 'TEX/SDP', 'COL/NYY',  
'TEX/WSN', 'COL/OAK', 'WSN/MIL', 'WSN/ARI', 'KCR/COL', 'NYY/PIT',  
'MIL/KCR', 'TBR/LAD', 'MIN/CIN', 'KCR/CIN', 'SEA/PHI', 'ARI/BAL',  
'LAD/KCR', 'STL/NYY', 'LAD/TBR', 'ANA/STL', 'BOS/SDP', 'ATL/KCR',  
'BOS/TEX', 'COL/MIA/ARI', 'OAK/LAD', 'DET/STL', 'ATL/COL',  
'LAD/SDP', 'BAL/CHC', 'SDP/LAD', 'SEA/SDP', 'OAK/PHI', 'CHC/OAK',  
'OAK/CHC', 'MIN/COL', 'SDP/HOU', 'CHW/COL', 'NYM/WSN', 'TBR/ANA',  
'BOS/CLE', 'SDP/CHW', 'BOS/SFG', 'CHW/SDP', 'BOS/STL', 'PHI/HOU',  
'ARI/ANA', 'CHW/ARI', 'ARI/CHW', 'LAD/PIT', 'HOU/PHI', 'ANA/ARI',  
'DET/ARI', 'STL/NYM', 'SEA/BOS', 'CHW/STL', 'TOR/CHW', 'CLE/COL',  
'BOS/LAD', 'COL/KCR', 'HOU/TOR', 'MIN/SDP', 'MIA/DET', 'DET/MIA',  
'MIL/BAL', 'CHC/BAL', 'BAL/ATL', 'SEA/NYM', 'ARI/SDP', 'HOU/BAL',  
'HOU/MIA', 'MIN/HOU', 'TOR/KCR', 'BOS/OAK', 'ARI/NYY', 'OAK/MIN',  
'NYY/ARI', 'TBR/DET', 'DET/TBR', 'MIA/CHC', 'TEX/PHI', 'TOR/DET',  
'TOR/OAK', 'PHI/TEX', 'SEA/PIT', 'OAK/HOU', 'CIN/SFG', 'DET/TOR',  
'ATL/LAD', 'ATL/ANA', 'MIA/SDP', 'ATL/TEX', 'PIT/TOR', 'MIN/ANA',  
'SEA/BAL', 'SEA/CHC', 'TBR/SFG', 'SDP/BOS', 'ANA/MIN', 'CHC/TOR',  
'ATL/MIN', 'ATL/MIN/NYY', 'STL/SEA', 'CHW/TEX', 'PHI/BAL',  
'MIA/TOR', 'MIL/NYM', 'CHW/CHC', 'TBR/SEA', 'OAK/TOR', 'TBR/PIT',  
'MIA/ANA', 'TBR/BOS', 'DET/OAK', 'TOR/CHC', 'OAK/BAL', 'MIL/BOS',  
'MIA/ARI', 'ATL/CIN', 'ARI/HOU', 'SEA/ARI', 'PIT/MIL', 'SFG/MIL',  
'MIA/TBR', 'CIN/OAK', 'TOR/HOU', 'TOR/NYM', 'NYM/PHI', 'CLE/SDP',  
'ARI/TOR', 'LAD/TOR', 'SEA/TOR', 'CHC/SDP', 'STL/MIN', 'LAD/WSN',  
'MIN/STL', 'BAL/STL', 'WSN/STL', 'OAK/MIA', 'CHC/NYM', 'CIN/SEA',  
'MIN/MIL/NYY', 'CHW/ARI/TEX', 'CIN/MIN', 'CHC/DET', 'HOU/ATL',  
'STL/PIT', 'PIT/STL', 'TBR/WSN', 'WSN/MIN', 'TBR/ATL', 'CLE/TBR',  
'KCR/CHC', 'PHI/PIT', 'STL/BAL', 'SEA/COL', 'CHW/ANA/CLE',  
'PIT/SDP', 'DET/PHI', 'LAD/CLE', 'CLE/CHW', 'NYM/HOU',  
'CIN/SEA/NYY', 'KCR/LAD'], dtype=object)
```

In [61]: 1 pitch_combined_df

Out[61]:

	Name	Age	Year	Throws	Tm	IP	G	GS	CG	SHO	sDR	IDR
0	Ed Acosta	28	1972	1	SDP	89.0	46	2	0	0	1	1
1	Doyle Alexander	21	1972	1	BAL	106.1	35	9	2	2	3	4
2	Lloyd Allen	22	1972	1	ANA	85.1	42	6	0	0	5	0
3	Steve Arlin	26	1972	1	SDP	250.0	38	37	12	3	22	7
4	Stan Bahnsen	27	1972	1	CHW	252.1	43	41	5	1	32	4
...
16568	Ryan Yarbrough*	31	2023	0	KCR/LAD	89.2	25	9	0	0	0	5
16569	Ryan Yarbrough*	31	2023	0	KCR	51.0	14	7	0	0	0	4
16570	Ryan Yarbrough*	31	2023	0	LAD	38.2	11	2	0	0	0	1
16571	Rob Zastryzny*	31	2023	0	PIT	20.2	21	1	0	0	0	1
16572	Angel Zerpa*	23	2023	0	KCR	42.2	15	3	0	0	1	1

16573 rows × 19 columns



```
In [62]: 1 combined_teams = pitch_combined_df[pitch_combined_df['Tm'].str.contains('
2
3 # Step 2: Find and remove rows with individual teams for the same player
4 for index, row in combined_teams.iterrows():
5     names = row['Name']
6     year = row['Year']
7     # Extract the individual teams from the 'Tm' column
8     teams = row['Tm'].split('/')
9     # Remove rows where the team is one of the individual teams and the
10    pitch_combined_df = pitch_combined_df[~((pitch_combined_df['Name'] == names) & (pitch_combined_df['Year'] == year) & (pitch_combined_df['Tm'].str.contains('/') | pitch_combined_df['Tm'].str.contains('
11
12
13
14 # Resulting DataFrame
15 pitch_combined_df
```

Out[62]:

	Name	Age	Year	Throws	Tm	IP	G	GS	CG	SHO	sDR	IDR
0	Ed Acosta	28	1972	1	SDP	89.0	46	2	0	0	1	1
1	Doyle Alexander	21	1972	1	BAL	106.1	35	9	2	2	3	4
2	Lloyd Allen	22	1972	1	ANA	85.1	42	6	0	0	5	0
3	Steve Arlin	26	1972	1	SDP	250.0	38	37	12	3	22	7
4	Stan Bahnsen	27	1972	1	CHW	252.1	43	41	5	1	32	4
...
16566	Brandon Woodruff	30	2023	1	MIL	67.0	11	11	1	1	0	10
16567	Kyle Wright	27	2023	1	ATL	31.0	9	7	0	0	0	5
16568	Ryan Yarbrough*	31	2023	0	KCR/LAD	89.2	25	9	0	0	0	5
16571	Rob Zastryzny*	31	2023	0	PIT	20.2	21	1	0	0	0	1
16572	Angel Zerpa*	23	2023	0	KCR	42.2	15	3	0	0	1	1

15060 rows × 19 columns



```
In [63]: 1 pitch_combined_df['Name'] = pitch_combined_df['Name'].str.lower().str.
2 pitch_combined_df
```

Out[63]:

	Name	Age	Year	Throws	Tm	IP	G	GS	CG	SHO	sDR	IDR
0	ed acosta	28	1972	1	SDP	89.0	46	2	0	0	1	1
1	doyle alexander	21	1972	1	BAL	106.1	35	9	2	2	3	4
2	lloyd allen	22	1972	1	ANA	85.1	42	6	0	0	5	0
3	steve arlin	26	1972	1	SDP	250.0	38	37	12	3	22	7
4	stan bahnsen	27	1972	1	CHW	252.1	43	41	5	1	32	4
...
16566	brandon woodruff	30	2023	1	MIL	67.0	11	11	1	1	0	10
16567	kyle wright	27	2023	1	ATL	31.0	9	7	0	0	0	5
16568	ryan yarbrough	31	2023	0	KCR/LAD	89.2	25	9	0	0	0	5
16571	rob zastryzny	31	2023	0	PIT	20.2	21	1	0	0	0	1
16572	angel zerpa	23	2023	0	KCR	42.2	15	3	0	0	1	1

15060 rows × 19 columns

```
In [64]: 1 print(pitch_combined_df['Name'].unique())
```

```
['ed\x00acosta' 'doyle\x00alexander' 'lloyd\x00allen' ... 'keaton\x00winn'
 'jackson\x00wolf' 'bryan\x00woo']
```

This is odd. Need to clean up.

```
In [65]: 1 pitch_combined_df['Name'] = pitch_combined_df['Name'].str.replace('\xa0')
2 pitch_combined_df['Name'] = pitch_combined_df['Name'].str.strip()
3 print(pitch_combined_df['Name'].unique())
```

```
['ed acosta' 'doyle alexander' 'lloyd allen' ... 'keaton winn'
 'jackson wolf' 'bryan woo']
```

```
In [66]: ❶ 1 def remove_accents(input_str):
2     nfkd_form = unicodedata.normalize('NFKD', input_str)
3     return "".join([c for c in nfkd_form if not unicodedata.combining(
4
5     def clean_name(name):
6         name = name.lower()
7         name = remove_accents(name)
8         name = re.sub(r'[-.]', ' ', name)
9         name = re.sub(r'\s+', ' ', name).strip()
10    return name
11
12 pitch_combined_df['Name'] = pitch_combined_df['Name'].apply(clean_name)
```

```
In [67]: ❷ 1 filtered_df = pitch_combined_df
```

```
In [68]: ❸ 1 filtered_df['Team'] = filtered_df['Tm']
2 filtered_df.drop(columns=['Tm'], inplace=True)
3 filtered_df.head()
```

Out[68]:

	Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	IDR	IP/GS	Pit/GS	<80
0	ed acosta	28	1972		1	89.0	46	2	0	0	1	1	7.0	NaN
1	doyle alexander	21	1972		1	106.1	35	9	2	2	3	4	6.8	NaN
2	lloyd allen	22	1972		1	85.1	42	6	0	0	5	0	4.8	NaN
3	steve arlin	26	1972		1	250.0	38	37	12	3	22	7	6.7	NaN
4	stan bahnsen	27	1972		1	252.1	43	41	5	1	32	4	6.1	NaN

In [69]:

```

1 filtered_df['Career Start'] = filtered_df.groupby('Name')['Year'].tran
2 filtered_df['Career End'] = filtered_df.groupby('Name')['Year'].transf
3 filtered_df

```

Out[69]:

		Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	...	IP/GS	Pit/GS	<=
0	ed acosta	28	1972		1	89.0	46	2	0	0	1	...	7.0	NaN	Na
1	doyle alexander	21	1972		1	106.1	35	9	2	2	3	...	6.8	NaN	Na
2	lloyd allen	22	1972		1	85.1	42	6	0	0	5	...	4.8	NaN	Na
3	steve arlin	26	1972		1	250.0	38	37	12	3	22	...	6.7	NaN	Na
4	stan bahnsen	27	1972		1	252.1	43	41	5	1	32	...	6.1	NaN	Na
...
16566	brandon woodruff	30	2023		1	67.0	11	11	1	1	0	...	6.1	95.0	0
16567	kyle wright	27	2023		1	31.0	9	7	0	0	0	...	3.7	72.0	5
16568	ryan yarbrough	31	2023		0	89.2	25	9	0	0	0	...	5.1	79.0	6
16571	rob zastryzny	31	2023		0	20.2	21	1	0	0	0	...	1.0	25.0	1
16572	angel zerpa	23	2023		0	42.2	15	3	0	0	1	...	4.2	77.0	1

15060 rows × 21 columns



In [70]: 1 filtered_df.info()

```
<class 'pandas.core.frame.DataFrame'>
Index: 15060 entries, 0 to 16572
Data columns (total 21 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Name        15060 non-null   object 
 1   Age         15060 non-null   int64  
 2   Year        15060 non-null   object 
 3   Throws      15060 non-null   int32  
 4   IP          15060 non-null   float64
 5   G           15060 non-null   int64  
 6   GS          15060 non-null   int64  
 7   CG          15060 non-null   int64  
 8   SHO         15060 non-null   int64  
 9   sDR         15060 non-null   int64  
 10  lDR         15060 non-null   int64  
 11  IP/GS       15060 non-null   float64
 12  Pit/GS      11067 non-null   float64
 13  <80          11078 non-null   float64
 14  80-99        11078 non-null   float64
 15  100-119      11078 non-null   float64
 16  ≥120         11078 non-null   float64
 17  Max          11078 non-null   float64
 18  Team         15060 non-null   object 
 19  Career Start 15060 non-null   object 
 20  Career End    15060 non-null   object 
dtypes: float64(8), int32(1), int64(7), object(5)
memory usage: 2.5+ MB
```

```
In [71]: 1 tj_df = filtered_df[filtered_df['Name'] == 'tommy john']
2 tj_df
```

Out[71]:

		Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	...	IP/GS	Pit/GS	<80		
109		tommy john	29	1972		0	186.2	29	29	4	1	5	...	6.4	NaN	NaN	↑
348		tommy john	30	1973		0	218.0	36	31	4	2	5	...	6.8	NaN	NaN	↑
599		tommy john	31	1974		0	153.0	22	22	5	3	5	...	7.0	NaN	NaN	↑
1124		tommy john	33	1976		0	207.0	31	31	6	2	1	...	6.7	NaN	NaN	↑
1384		tommy john	34	1977		0	220.1	31	31	11	3	3	...	7.1	NaN	NaN	↑
1669		tommy john	35	1978		0	213.0	33	30	7	0	4	...	6.8	NaN	NaN	↑
1940		tommy john	36	1979		0	276.1	37	36	17	3	11	...	7.6	NaN	NaN	↑
2219		tommy john	37	1980		0	265.1	36	36	16	6	8	...	7.4	NaN	NaN	↑
2473		tommy john	38	1981		0	140.1	20	20	7	0	4	...	7.0	NaN	NaN	↑
2732		tommy john	39	1982		0	221.2	37	33	10	2	8	...	6.6	NaN	NaN	↑
3017		tommy john	40	1983		0	234.2	34	34	9	0	2	...	6.9	NaN	NaN	↑
3296		tommy john	41	1984		0	181.1	32	29	4	1	4	...	6.1	NaN	NaN	↑
3564		tommy john	42	1985		0	86.1	23	17	0	0	2	...	4.7	NaN	NaN	↑
3861		tommy john	43	1986		0	70.2	13	10	1	0	0	...	6.4	NaN	NaN	↑
4164		tommy john	44	1987		0	187.2	33	33	3	1	3	...	5.7	NaN	NaN	↑
4461		tommy john	45	1988		0	176.1	35	32	0	0	3	...	5.3	84.0	11.0	↑
4736		tommy john	46	1989		0	63.2	10	10	0	0	0	...	6.4	101.0	1.0	↑

17 rows × 21 columns



In [72]:

```
1 # Convert 'Career Start', 'Career End', and 'Year' to integers if not
2 filtered_df['Career Start'] = filtered_df['Career Start'].astype(int)
3 filtered_df['Career End'] = filtered_df['Career End'].astype(int)
4 filtered_df['Year'] = filtered_df['Year'].astype(int)
5
6 # Define a function to calculate inactive years for each player
7 def calculate_inactive_years(row):
8     active_years = set(filtered_df[filtered_df['Name'] == row['Name']])
9     all_years = set(range(row['Career Start'], row['Career End'] + 1))
10    inactive_years = all_years - active_years
11    return sorted(list(inactive_years))
12
13 # Apply the function to each row to calculate 'Inactive Years'
14 filtered_df['Inactive Years'] = filtered_df.apply(calculate_inactive_ye
15
16 # Now, let's ensure there's only one 'Inactive Years' column and clean
17 columns_to_keep = ['Name', 'Age', 'Year', 'Throws', 'IP', 'G', 'GS', 'I
18 filtered_df = filtered_df[columns_to_keep]
19
20 # Display a snippet of the DataFrame to verify the results
21 print(filtered_df[['Name', 'Career Start', 'Career End', 'Inactive Yea
```

	Name	Career Start	Career End	Inactive Years
0	ed acosta	1972	1972	[]
1	doyle alexander	1972	1989	[]
2	lloyd allen	1972	1975	[]
3	steve arlin	1972	1974	[]
4	stan bahnsen	1972	1981	[1979, 1980]

In [73]:

```

1 tj_df = filtered_df[filtered_df['Name'] == 'tommy john']
2 tj_df

```

Out[73]:

		Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	Inactive Years
	109	tommy john	29	1972	0	186.2	29	29	4	1	5	1972	1989	[1975]
	348	tommy john	30	1973	0	218.0	36	31	4	2	5	1972	1989	[1975]
	599	tommy john	31	1974	0	153.0	22	22	5	3	5	1972	1989	[1975]
	1124	tommy john	33	1976	0	207.0	31	31	6	2	1	1972	1989	[1975]
	1384	tommy john	34	1977	0	220.1	31	31	11	3	3	1972	1989	[1975]
	1669	tommy john	35	1978	0	213.0	33	30	7	0	4	1972	1989	[1975]
	1940	tommy john	36	1979	0	276.1	37	36	17	3	11	1972	1989	[1975]
	2219	tommy john	37	1980	0	265.1	36	36	16	6	8	1972	1989	[1975]
	2473	tommy john	38	1981	0	140.1	20	20	7	0	4	1972	1989	[1975]
	2732	tommy john	39	1982	0	221.2	37	33	10	2	8	1972	1989	[1975]
	3017	tommy john	40	1983	0	234.2	34	34	9	0	2	1972	1989	[1975]
	3296	tommy john	41	1984	0	181.1	32	29	4	1	4	1972	1989	[1975]
	3564	tommy john	42	1985	0	86.1	23	17	0	0	2	1972	1989	[1975]
	3861	tommy john	43	1986	0	70.2	13	10	1	0	0	1972	1989	[1975]
	4164	tommy john	44	1987	0	187.2	33	33	3	1	3	1972	1989	[1975]
	4461	tommy john	45	1988	0	176.1	35	32	0	0	3	1972	1989	[1975]
	4736	tommy john	46	1989	0	63.2	10	10	0	0	0	1972	1989	[1975]

In [74]:

```
1 rob_zas_df = filtered_df[filtered_df['Name'] == 'rob zastryzny']
2 rob_zas_df
```

Out[74]:

	Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	Inactiv Year
13750	rob zastryzny	24	2016		0	16.0	8	1	0	0	2016	2023	[2017 2018 2019 2020 2021 2022]
16571	rob zastryzny	31	2023		0	20.2	21	1	0	0	2016	2023	[2017 2018 2019 2020 2021 2022]
◀ ⏴ ⏵ ⏹ ▶													

This is wrong.

Zastryzny played 2016-2023.

Missing all years inbetween.

In [75]:

```
1 jasoni_df = filtered_df[filtered_df['Name'] == 'jason isringhausen']
2 jasoni_df
```

Out[75]:

	Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	Inact Y
6510	jason isringhausen	22	1995		1	93.0	14	14	1	0	0	1995	1999 [1]
6829	jason isringhausen	23	1996		1	171.2	27	27	2	1	0	1995	1999 [1]
7165	jason isringhausen	24	1997		1	29.2	6	6	0	0	0	1995	1999 [1]
7827	jason isringhausen	26	1999		1	64.2	33	5	0	0	0	1995	1999 [1]
7828	jason isringhausen	26	1999		1	39.1	13	5	0	0	0	1995	1999 [1]
◀ ⏴ ⏵ ⏹ ▶													

Isringhausen played 1995-2012. Missed 1998.

Data came from "Starting Pitching Baseball Reference".

Pitchers may not be labeled SP for those years.

How accurate to make innactivity?

Isringhausen had TJ surgery 3x, only spent 1 calendar year out of baseball.

In [76]: 1 filtered_df.info()

```
<class 'pandas.core.frame.DataFrame'>
Index: 15060 entries, 0 to 16572
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Name              15060 non-null   object  
 1   Age               15060 non-null   int64  
 2   Year              15060 non-null   int32  
 3   Throws             15060 non-null   int32  
 4   IP                15060 non-null   float64 
 5   G                 15060 non-null   int64  
 6   GS                15060 non-null   int64  
 7   CG                15060 non-null   int64  
 8   SHO               15060 non-null   int64  
 9   sDR               15060 non-null   int64  
 10  Career Start      15060 non-null   int32  
 11  Career End        15060 non-null   int32  
 12  Inactive Years    15060 non-null   object  
dtypes: float64(1), int32(4), int64(6), object(2)
memory usage: 1.4+ MB
```

Save as CSV just in case

In [77]: 1 filtered_df.to_csv('data/cleaning_filtered_df.csv')

This df looks good.

Time to compare it to the TJ df.

Need to merge tj_list_df_filtered and filtered_df

```
In [78]: ┌─┐
1 historic_tj_df = filtered_df.merge(
2     tj_list_df_filtered[['Name', 'Surgery', 'TJ Surgery Date', 'Surgeon'],
3     on='Name',
4     how='left'
5 )
6
7 historic_tj_df['Surgery'] = historic_tj_df['Surgery'].fillna(0)
8 historic_tj_df
```

Out[78]:

	Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	Inact Year
0	ed acosta	28	1972		1	89.0	46	2	0	0	1	1972	1972
1	doyle alexander	21	1972		1	106.1	35	9	2	2	3	1972	1989
2	lloyd allen	22	1972		1	85.1	42	6	0	0	5	1972	1975
3	steve arlin	26	1972		1	250.0	38	37	12	3	22	1972	1974
4	stan bahnsen	27	1972		1	252.1	43	41	5	1	32	1972	1981
...
15348	brandon woodruff	30	2023		1	67.0	11	11	1	1	0	2017	2023
15349	kyle wright	27	2023		1	31.0	9	7	0	0	0	2019	2023
15350	ryan yarbrough	31	2023		0	89.2	25	9	0	0	0	2018	2023
15351	rob zastryzny	31	2023		0	20.2	21	1	0	0	0	2016	2023
15352	angel zerpa	23	2023		0	42.2	15	3	0	0	1	2021	2023

15353 rows × 18 columns

```
In [79]: ┌─┐
1 historic_tj_df['TJ Surgery Date'] = historic_tj_df['TJ Surgery Date'].
```

In [80]: 1 historic_tj_df.head()

Out[80]:

	Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	Inactive Years
#													
0	ed acosta	28	1972	1	89.0	46	2	0	0	1	1972	1972	0
1	doyle alexander	21	1972	1	106.1	35	9	2	2	3	1972	1989	0
2	lloyd allen	22	1972	1	85.1	42	6	0	0	5	1972	1975	0
3	steve arlin	26	1972	1	250.0	38	37	12	3	22	1972	1974	0
4	stan bahnsen	27	1972	1	252.1	43	41	5	1	32	1972	1981	[1979, 1980]

In [81]: 1 historic_tj_df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15353 entries, 0 to 15352
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   Name            15353 non-null   object 
 1   Age             15353 non-null   int64  
 2   Year            15353 non-null   int32  
 3   Throws          15353 non-null   int32  
 4   IP              15353 non-null   float64
 5   G               15353 non-null   int64  
 6   GS              15353 non-null   int64  
 7   CG              15353 non-null   int64  
 8   SHO             15353 non-null   int64  
 9   sDR             15353 non-null   int64  
 10  Career Start    15353 non-null   int32  
 11  Career End      15353 non-null   int32  
 12  Inactive Years  15353 non-null   object 
 13  Surgery         15353 non-null   float64
 14  TJ Surgery Date 15353 non-null   object 
 15  Surgeon(s)     2405  non-null    object 
 16  Country         3407  non-null    object 
 17  Level           3407  non-null    object 
dtypes: float64(2), int32(4), int64(6), object(6)
memory usage: 1.9+ MB
```

In [82]: 1 historic_tj_df['Surgery'].value_counts()

Out[82]: Surgery

0.0	11946
1.0	2828
2.0	558
3.0	21

Name: count, dtype: int64

```
In [83]: 1 surgery_3 = historic_tj_df[historic_tj_df['Surgery'] == 3]['Name']
          2 surgery_3
```

```
Out[83]: 6003    jason isringhausen
          6004    jason isringhausen
          6005    jason isringhausen
          6289    jason isringhausen
          6290    jason isringhausen
          6291    jason isringhausen
          6588    jason isringhausen
          6589    jason isringhausen
          6590    jason isringhausen
          7181    jason isringhausen
          7182    jason isringhausen
          7183    jason isringhausen
          7184    jason isringhausen
          7185    jason isringhausen
          7186    jason isringhausen
          13363   jonny venters
          13364   jonny venters
          13365   jonny venters
          13366   jonny venters
          13367   jonny venters
          13368   jonny venters
Name: Name, dtype: object
```

Need to filter down.

Minimum 320 IP (assumed 2 seasons played)

```
In [84]: 1 # Calculating the sum total of 'IP' for each player in the merged data;
          2 ip_sum = historic_tj_df.groupby('Name')['IP'].sum().sort_values(ascending=True)
          3 ip_sum.head(1200)
```

```
Out[84]: Name
          greg maddux      5005.3
          roger clemens     4910.6
          nolan ryan        4871.8
          randy wolf         4646.4
          bert blyleven     4523.0
          ...
          jason bergmann    337.7
          jimmy gobble       337.4
          steve blass        337.4
          don schulze        336.8
          lance painter      336.8
Name: IP, Length: 1200, dtype: float64
```

In [85]: ⏎

```

1 doyle_df = historic_tj_df[historic_tj_df['Name'] == 'doyle alexander']
2 doyle_df

```

Out[85]:

		Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	Inactiv Year
1		doyle alexander	21	1972		1	106.1	35	9	2	2	3	1972	1989
227		doyle alexander	22	1973		1	174.2	29	26	10	0	8	1972	1989
459		doyle alexander	23	1974		1	114.1	30	12	2	0	5	1972	1989
685		doyle alexander	24	1975		1	133.1	32	11	3	1	4	1972	1989
925		doyle alexander	25	1976		1	201.0	30	25	7	3	6	1972	1989
1151		doyle alexander	26	1977		1	237.0	34	34	12	1	6	1972	1989
1422		doyle alexander	27	1978		1	191.0	31	28	7	1	3	1972	1989
1673		doyle alexander	28	1979		1	113.1	23	18	0	0	7	1972	1989
1940		doyle alexander	29	1980		1	231.2	35	35	7	1	9	1972	1989
2191		doyle alexander	30	1981		1	152.1	24	24	1	1	4	1972	1989
2414		doyle alexander	31	1982		1	66.2	16	11	0	0	0	1972	1989
2675		doyle alexander	32	1983		1	145.0	25	20	5	0	1	1972	1989
2931		doyle alexander	33	1984		1	261.2	36	35	11	2	8	1972	1989
3190		doyle alexander	34	1985		1	260.2	36	36	6	1	5	1972	1989
3456		doyle alexander	35	1986		1	228.1	34	34	5	0	2	1972	1989
3718		doyle alexander	36	1987		1	206.0	27	27	6	3	3	1972	1989
3983		doyle alexander	37	1988		1	229.0	34	34	5	1	4	1972	1989
4241		doyle alexander	38	1989		1	223.0	33	33	5	1	2	1972	1989

In [86]:

```

1 # Clear 'Total_IP' column first
2 historic_tj_df['Total_IP'] = None
3
4 # Calculate Total_IP for each player
5 player_total_ip = historic_tj_df.groupby('Name')['IP'].sum().reset_index()
6
7 # Merge info back to original df, only for the last year of play for each player
8 historic_tj_df = historic_tj_df.merge(player_total_ip, on='Name', how='left')
9
10 # Identify the last year of play for each player
11 historic_tj_df['Last_Year_Flag'] = historic_tj_df.groupby('Name')['Year'].transform(lambda x: x == x.max())
12
13 # Update 'Total_IP' only for the last year of play
14 historic_tj_df.loc[historic_tj_df['Year'] == historic_tj_df['Last_Year_Flag'], 'Total_IP'] = player_total_ip['Total_IP']
15
16 # Clean up df, remove unnecessary columns
17 historic_tj_df.drop(columns=['Total_IP_y', 'Last_Year_Flag'], inplace=True)
18 historic_tj_df.rename(columns={'Total_IP_x': 'Total_IP'}, inplace=True)
19
20 historic_tj_df[historic_tj_df['Year'] == historic_tj_df['Year'].max()]

```

Out[86]:

		Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	Inact Year
14940		andrew abbott	24	2023		0	109.1	21	21	0	0	2023	2023	
14941		joan adon	24	2023		1	51.2	12	10	0	0	2021	2023	
14942		keegan akin	28	2023		0	23.2	24	1	0	0	2020	2023	
14943		sandy alcantara	27	2023		1	184.2	28	28	3	1	2018	2023	
14944		scott alexander	33	2023		0	48.1	55	8	0	0	2018	2023	[2018, 2023)

```
In [87]: 1 players_below_320_ip_recheck = historic_tj_df[historic_tj_df['Total_IP']
2 historic_tj_df_temp_check = historic_tj_df[~historic_tj_df['Name'].isna()
3 historic_tj_df_temp_check
```

Out[87]:

		Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	Inact Year
1		doyle alexander	21	1972		1	106.1	35	9	2	2	3	1972	1989
3		steve arlin	26	1972		1	250.0	38	37	12	3	22	1972	1974
4		stan bahnsen	27	1972		1	252.1	43	41	5	1	32	1972	1981
7		jim barr	24	1972		1	179.0	44	18	8	2	3	1972	1982
10		jim bibby	27	1972		1	40.1	6	6	0	0	1	1972	1983
...	
15334		zack wheeler	33	2023		1	192.0	32	32	0	0	0	2013	2023
15339		trevor williams	31	2023		1	144.1	30	30	0	0	0	2016	2023
15346		alex wood	32	2023		0	97.2	29	12	0	0	1	2013	2023
15348		brandon woodruff	30	2023		1	67.0	11	11	1	1	0	2017	2023
15350		ryan yarbrough	31	2023		0	89.2	25	9	0	0	0	2018	2023

10846 rows × 19 columns

```
In [88]: 1 historic_tj_df_temp_check['Name'].value_counts()
```

Out[88]:

Name	count
randy wolf	32
edinson volquez	28
brian anderson	26
jamie moyer	25
nathan eovaldi	24
..	
manny sarmiento	3
milt pappas	2
steve blass	2
rick matula	2
george kirby	2

Name: count, Length: 1244, dtype: int64

```
In [89]: 1 min_ip_df = historic_tj_df[historic_tj_df['Total_IP'] >= 320.0][['Name', 'Throws', 'Surgery', 'Total_IP']]
2 min_ip_df
```

Out[89]:

	Name	Throws	Surgery	Total_IP
240	steve blass	1	0.0	337.4
285	eddie fisher	1	0.0	330.6
286	eddie fisher	1	0.0	330.6
385	milt pappas	1	0.0	357.0
463	steve arlin	1	0.0	537.2
...
15334	zack wheeler	1	1.0	1377.5
15339	trevor williams	1	0.0	857.1
15346	alex wood	0	1.0	1214.0
15348	brandon woodruff	1	0.0	678.7
15350	ryan yarbrough	0	0.0	667.7

1390 rows × 4 columns

```
In [90]: 1 min_ip_df['Surgery'].value_counts()
```

Out[90]: Surgery

0.0	1067
1.0	239
2.0	78
3.0	6

Name: count, dtype: int64

```
In [91]: 1 min_ip_df['Name'].value_counts()
```

Out[91]: Name

jason isringhausen	6
cole ragans	4
daniel hudson	4
kameron loe	3
dan petry	3
..	
mike harkey	1
dave fleming	1
john doherty	1
jim deshaies	1
ryan yarbrough	1

Name: count, Length: 1244, dtype: int64

In [92]:

```
1 eddie = min_ip_df[min_ip_df['Name'] == 'eddie fisher']
2 eddie
```

Out[92]:

	Name	Throws	Surgery	Total_IP
285	eddie fisher	1	0.0	330.6
286	eddie fisher	1	0.0	330.6

In [93]:

```
1 new_min_ip_df = historic_tj_df[historic_tj_df['Total_IP'] >= 320.0]
2 new_min_ip_df
```

Out[93]:

	Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	Inact Yea
240	steve blass	31	1973		1	88.2	23	18	1	0	1	1972	1973
285	eddie fisher	36	1973		1	117.2	32	16	2	0	9	1972	1973
286	eddie fisher	36	1973		1	110.2	26	16	2	0	9	1972	1973
385	milt pappas	34	1973		1	162.0	30	29	1	1	8	1972	1973
463	steve arlin	28	1974		1	107.2	27	22	2	0	9	1972	1974
...
15334	zack wheeler	33	2023		1	192.0	32	32	0	0	0	2013	2023 [2020]
15339	trevor williams	31	2023		1	144.1	30	30	0	0	0	2016	2023
15346	alex wood	32	2023		0	97.2	29	12	0	0	1	2013	2023
15348	brandon woodruff	30	2023		1	67.0	11	11	1	1	0	2017	2023
15350	ryan yarbrough	31	2023		0	89.2	25	9	0	0	0	2018	2023

1390 rows × 19 columns



```
In [94]: 1 jason_df = new_min_ip_df[new_min_ip_df['Name'] == 'jason isringhausen']
2 jason_df
```

Out[94]:

		Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	Inact	Ye
7181		jason isringhausen	26	1999		1	64.2	33	5	0	0	0	1995	1999	[19]
7182		jason isringhausen	26	1999		1	64.2	33	5	0	0	0	1995	1999	[19]
7183		jason isringhausen	26	1999		1	64.2	33	5	0	0	0	1995	1999	[19]
7184		jason isringhausen	26	1999		1	39.1	13	5	0	0	0	1995	1999	[19]
7185		jason isringhausen	26	1999		1	39.1	13	5	0	0	0	1995	1999	[19]
7186		jason isringhausen	26	1999		1	39.1	13	5	0	0	0	1995	1999	[19]

```
In [95]: 1 new_min_ip_df = new_min_ip_df.drop_duplicates(subset=['Name'])
```

In [96]: 1 new_min_ip_df

Out[96]:

		Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	Inact Year
240		steve blass	31	1973		1	88.2	23	18	1	0	1	1972	1973
285		eddie fisher	36	1973		1	117.2	32	16	2	0	9	1972	1973
385		milt pappas	34	1973		1	162.0	30	29	1	1	8	1972	1973
463		steve arlin	28	1974		1	107.2	27	22	2	0	9	1972	1974
589		ernie mcanally	27	1974		1	128.2	25	21	5	2	6	1972	1974
...	
15334		zack wheeler	33	2023		1	192.0	32	32	0	0	0	2013	2023
15339		trevor williams	31	2023		1	144.1	30	30	0	0	0	2016	2023
15346		alex wood	32	2023		0	97.2	29	12	0	0	1	2013	2023
15348		brandon woodruff	30	2023		1	67.0	11	11	1	1	0	2017	2023
15350		ryan yarbrough	31	2023		0	89.2	25	9	0	0	0	2018	2023

1244 rows × 19 columns

In [97]: 1 new_min_ip_df['Surgery'].value_counts()

Out[97]: Surgery

0.0	976
1.0	230
2.0	37
3.0	1

Name: count, dtype: int64

```
In [98]: 1 new_merge_df = new_min_ip_df
          2 new_merge_df
```

Out[98]:

		Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	Inact Year
240		steve blass	31	1973		1	88.2	23	18	1	0	1	1972	1973
285		eddie fisher	36	1973		1	117.2	32	16	2	0	9	1972	1973
385		milt pappas	34	1973		1	162.0	30	29	1	1	8	1972	1973
463		steve arlin	28	1974		1	107.2	27	22	2	0	9	1972	1974
589		ernie mcanally	27	1974		1	128.2	25	21	5	2	6	1972	1974
...	
15334		zack wheeler	33	2023		1	192.0	32	32	0	0	0	2013	2023
15339		trevor williams	31	2023		1	144.1	30	30	0	0	0	2016	2023
15346		alex wood	32	2023		0	97.2	29	12	0	0	1	2013	2023
15348		brandon woodruff	30	2023		1	67.0	11	11	1	1	0	2017	2023
15350		ryan yarbrough	31	2023		0	89.2	25	9	0	0	0	2018	2023

1244 rows × 19 columns



In [99]:

```

1 def convert_surgery_date_to_year(item):
2     # Handle cases where the item is 0, indicating no surgery
3     if item == 0:
4         return 0
5     # Handle non-list cases, directly converting to year if it's a val-
6     elif not isinstance(item, list):
7         return item.year if pd.notnull(item) else 0
8     # Handle list of dates, extracting the year from each valid date
9     else:
10        return [date.year for date in item if pd.notnull(date)]
11
12 # Apply the function to the column
13 new_merge_df['TJ Surgery Year'] = new_merge_df['TJ Surgery Date'].apply(
14

```

C:\Users\johns\AppData\Local\Temp\ipykernel_8628\209469094.py:13: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
new_merge_df['TJ Surgery Year'] = new_merge_df['TJ Surgery Date'].apply(
    convert_surgery_date_to_year)
```

In [100]:

```
1 new_merge_df.head()
```

Out[100]:

		Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	Inactive Years
240		steve blass	31	1973		1	88.2	23	18	1	0	1	1972	1973
285		eddie fisher	36	1973		1	117.2	32	16	2	0	9	1972	1973
385		milt pappas	34	1973		1	162.0	30	29	1	1	8	1972	1973
463		steve arlin	28	1974		1	107.2	27	22	2	0	9	1972	1974
589		ernie mcanally	27	1974		1	128.2	25	21	5	2	6	1972	1974

```
In [101]: 1 new_merge_df['TJ Surgery Year'].value_counts()
```

```
Out[101]: TJ Surgery Year
0                976
[2014]            16
[2003]            15
[2018]            12
[2016]            12
...
[2014, 2009]      1
[1991, 1990]      1
[2008, 2002]      1
[2012, 2005]      1
[2019, 2014]      1
Name: count, Length: 79, dtype: int64
```

```
In [102]: 1 new_merge_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1244 entries, 240 to 15350
Data columns (total 20 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Name             1244 non-null    object 
 1   Age              1244 non-null    int64  
 2   Year             1244 non-null    int32  
 3   Throws           1244 non-null    int32  
 4   IP               1244 non-null    float64
 5   G                1244 non-null    int64  
 6   GS               1244 non-null    int64  
 7   CG               1244 non-null    int64  
 8   SHO              1244 non-null    int64  
 9   sDR              1244 non-null    int64  
 10  Career Start     1244 non-null    int32  
 11  Career End       1244 non-null    int32  
 12  Inactive Years   1244 non-null    object 
 13  Surgery          1244 non-null    float64
 14  TJ Surgery Date 1244 non-null    object 
 15  Surgeon(s)       197 non-null     object 
 16  Country          268 non-null     object 
 17  Level             268 non-null     object 
 18  Total_IP          1244 non-null    object 
 19  TJ Surgery Year  1244 non-null    object 
dtypes: float64(2), int32(4), int64(6), object(8)
memory usage: 184.7+ KB
```

```
In [103]: 1 new_merge_df['Total_IP'].sort_values()
```

```
Out[103]: 15125    320.2
9038     320.4
2542     320.7
1198     321.3
5387     321.7
...
5105     4523.0
12338    4646.4
5570     4871.8
9594     4910.6
10038    5005.3
Name: Total_IP, Length: 1244, dtype: object
```

```
In [104]: 1 new_merge_df['TJ Surgery Year'] = new_merge_df['TJ Surgery Year'].astype(str)
2
3 # Convert string representation of list to actual list and then to integer
4 # Using ast.literal_eval to safely evaluate the string as a Python expression
5 new_merge_df['TJ Surgery Year'] = new_merge_df['TJ Surgery Year'].apply(lambda x: ast.literal_eval(x) if x.startswith('[') else [int(x)])
6
7 new_merge_df['TJ Surgery Year']
```

C:\Users\johns\AppData\Local\Temp\ipykernel_8628\3841373166.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
new_merge_df['TJ Surgery Year'] = new_merge_df['TJ Surgery Year'].astype(str)

C:\Users\johns\AppData\Local\Temp\ipykernel_8628\3841373166.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
new_merge_df['TJ Surgery Year'] = new_merge_df['TJ Surgery Year'].apply(lambda x: ast.literal_eval(x) if x.startswith('[') else [int(x)])

```
Out[104]: 240      [0]
285      [0]
385      [0]
463      [0]
589      [0]
...
15334    [2015]
15339    [0]
15346    [2009]
15348    [0]
15350    [0]
Name: TJ Surgery Year, Length: 1244, dtype: object
```

```
In [105]: 1 john_df = new_merge_df[new_merge_df['Name'] == 'tommy john']
2 john_df
```

Out[105]:

	Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	Inactive Years	
4363	tommy john	46	1989		0	63.2	10	10	0	0	1972	1989	[1975]	

```
In [106]: ❶ def safe_convert_tj_year(entry):
2     if isinstance(entry, str): # If the entry is a string, attempt to
3         try:
4             # Attempt to evaluate the string to a Python object (list)
5             evaluated = literal_eval(entry)
6             if isinstance(evaluated, list):
7                 return [int(i) for i in evaluated if i != 0]
8             else:
9                 return []
10        except (ValueError, SyntaxError):
11            # In case of evaluation error, return an empty list
12            return []
13    elif isinstance(entry, list): # If the entry is already a list, p
14        return [int(i) for i in entry if i != 0]
15    else:
16        # For any other data type, return an empty list
17        return []
18
19 # Example usage on a simulated column (simulating 'TJ Surgery Year' wi
20 simulated_tj_year = ['[2014, 2016]', '[0]', '[]', [2018, 2019], 0, 'no
21
22 # Apply the safe conversion function
23 converted_tj_year = [safe_convert_tj_year(item) for item in simulated_
24
25 # Show the result of the conversion
26 converted_tj_year
```

Out[106]: [[2014, 2016], [], [], [2018, 2019], [], []]

```
In [107]: ❶ # Apply the safe conversion function to the 'TJ Surgery Year' column i
2 new_merge_df['TJ Surgery Year'] = new_merge_df['TJ Surgery Year'].apply(safe_c
```

C:\Users\johns\AppData\Local\Temp\ipykernel_8628\1494688879.py:2: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
new_merge_df['TJ Surgery Year'] = new_merge_df['TJ Surgery Year'].apply(safe_c
```

In [108]: 1 new_merge_df.info()

```
<class 'pandas.core.frame.DataFrame'>
Index: 1244 entries, 240 to 15350
Data columns (total 20 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Name              1244 non-null    object  
 1   Age               1244 non-null    int64  
 2   Year              1244 non-null    int32  
 3   Throws             1244 non-null    int32  
 4   IP                1244 non-null    float64 
 5   G                 1244 non-null    int64  
 6   GS                1244 non-null    int64  
 7   CG                1244 non-null    int64  
 8   SHO               1244 non-null    int64  
 9   sDR               1244 non-null    int64  
 10  Career Start      1244 non-null    int32  
 11  Career End        1244 non-null    int32  
 12  Inactive Years    1244 non-null    object  
 13  Surgery            1244 non-null    float64 
 14  TJ Surgery Date   1244 non-null    object  
 15  Surgeon(s)         197 non-null    object  
 16  Country            268 non-null    object  
 17  Level              268 non-null    object  
 18  Total_IP           1244 non-null    object  
 19  TJ Surgery Year   1244 non-null    object  
dtypes: float64(2), int32(4), int64(6), object(8)
memory usage: 184.7+ KB
```

In [109]: 1 new_merge_df['TJ Surgery Year'].value_counts()

Out[109]: TJ Surgery Year

[]	976
[2014]	16
[2003]	15
[2018]	12
[2016]	12
...	
[2014, 2009]	1
[1991, 1990]	1
[2008, 2002]	1
[2012, 2005]	1
[2019, 2014]	1

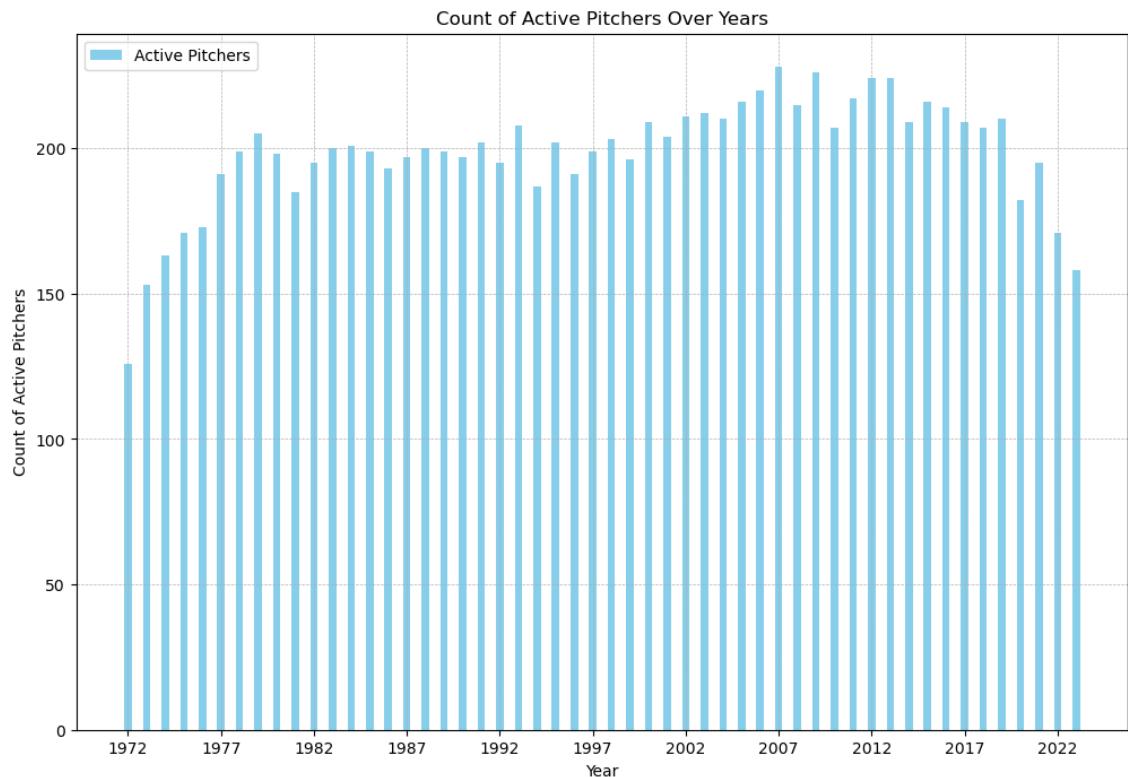
Name: count, Length: 79, dtype: int64

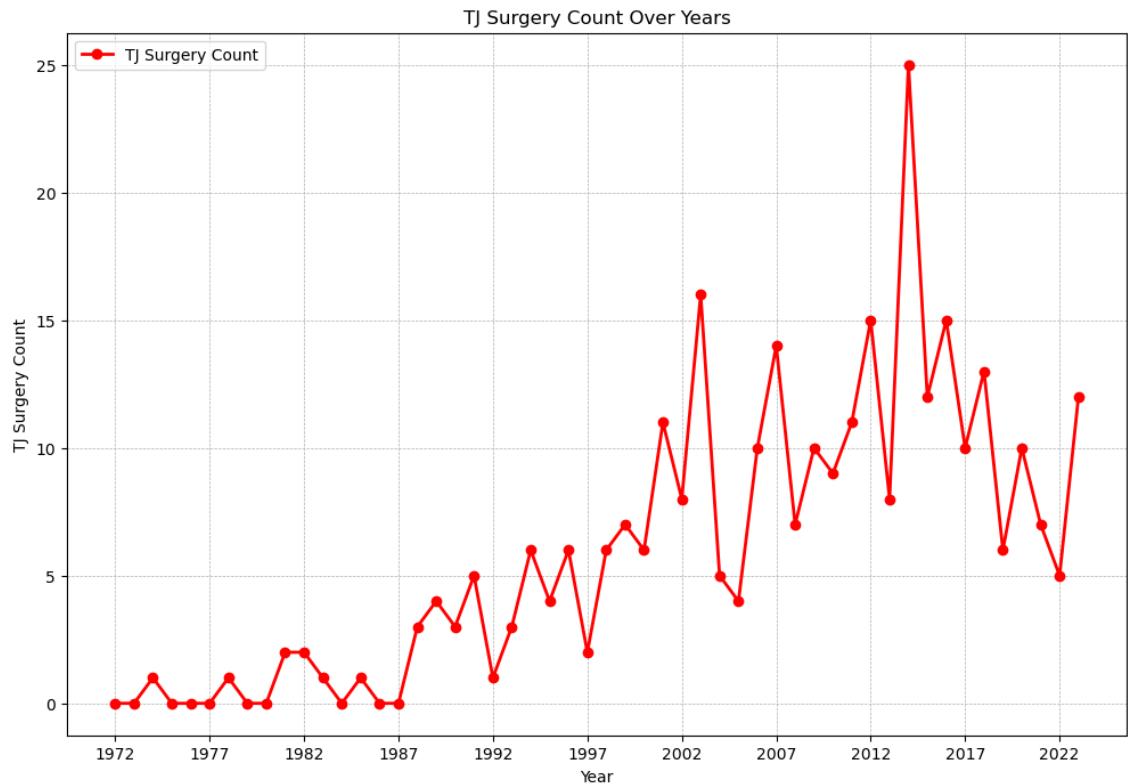
In [110]:

```
1 # Step 1: Determine the range of years
2 min_year = new_merge_df['Career Start'].min()
3 max_year = new_merge_df['Career End'].max()
4 years_range = np.arange(min_year, max_year + 1)
5
6 # Step 2: Initialize count series
7 active_pitchers_count = pd.Series(0, index=years_range)
8 tj_surgery_count = pd.Series(0, index=years_range)
9
10 # Step 3: Calculate active pitchers count
11 for index, row in new_merge_df.iterrows():
12     active_years = set(range(row['Career Start'], row['Career End'] + 1))
13     for year in active_years:
14         if year in active_pitchers_count.index:
15             active_pitchers_count.loc[year] += 1
16
17 # Step 4: Calculate TJ surgery count
18 for index, row in new_merge_df.iterrows():
19     for surgery_year in row['TJ Surgery Year']:
20         if surgery_year in tj_surgery_count.index:
21             tj_surgery_count.loc[surgery_year] += 1
22
```

In [111]:

```
1 min_year = active_pitchers_count.index.min()
2 max_year = active_pitchers_count.index.max()
3
4 fig, ax1 = plt.subplots(figsize=(12, 8))
5 ax1.bar(active_pitchers_count.index, active_pitchers_count.values, color='blue')
6 ax1.set_xlabel('Year')
7 ax1.set_ylabel('Count of Active Pitchers')
8 ax1.set_title('Count of Active Pitchers Over Years')
9 ax1.legend(loc='upper left')
10 ax1.grid(True, which='both', linestyle='--', linewidth=0.5)
11 ax1.set_xticks(np.arange(min_year, max_year + 1, 5)) # Adjusting x-axis ticks
12 plt.show()
13
14 fig, ax2 = plt.subplots(figsize=(12, 8))
15 ax2.plot(tj_surgery_count.index, tj_surgery_count.values, color='red', marker='o')
16 ax2.set_xlabel('Year')
17 ax2.set_ylabel('TJ Surgery Count')
18 ax2.set_title('TJ Surgery Count Over Years')
19 ax2.legend(loc='upper left')
20 ax2.grid(True, which='both', linestyle='--', linewidth=0.5)
21 ax2.set_xticks(np.arange(min_year, max_year + 1, 5)) # Adjusting x-axis ticks
22 plt.show()
```





NOTE:

1972 - 24 teams, 2 week players strike in April

1977 - 26 teams

1993 - 28 teams

1994 - players strike (50 reg season games & post cancelled. Many players careers ended)

1998 - 30 teams

2020 - shortened season (covid), 60 games

```
In [112]: ⏎ 1 new_merge_df.to_csv('data/hist_tj.csv')
```

```
In [ ]: ⏎ 1
```