In [1]:

```
import pybaseball as pyb
from pybaseball import statcast, pitching_stats, playerid_lookup, stat
import numpy as np
import math
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import glob
import os
import re
import ast
import unicodedata
from datetime import datetime
from itertools import groupby
from operator import itemgetter
from ast import literal_eval
```

```
C:\Users\johns\anaconda3\Lib\site-packages\pandas\core\arrays\masked.py:6
0: UserWarning: Pandas requires version '1.3.6' or newer of 'bottleneck'
(version '1.3.5' currently installed).
  from pandas.core import (
```

First, organize the TJ DF

In [2]:

```
tj_list_df = pd.read_csv('~/Documents/Flatiron/Project_5/tj_surgery_li
tj_list_df.columns = tj_list_df.iloc[0]

tj_list_df = tj_list_df.drop(tj_list_df.index[0])

tj_list_df.reset_index(drop=True, inplace=True)

tj_list_df.head()
```

Out[2]:

	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	S
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 [3]:

```
new_column_names = [
    "Player", "TJ Surgery Date", "Team", "Level", "Position", "Throws"
    "High School", "College(s)", "Age", "Return Date (same level)", "R
    "mlbamid", "fgid", "Surgeon(s)", "Post-TJ MLB G", "Post-TJ MLB IP/
    "Year", "Month", "Day", "Started\nThrowing", "Mound", "Bullpen", "
    "Game", "Setback", "Setback Type", "Setback", "Setback Type", "G",
    "K-BB%", "ERA-", "FIP-", "G", "GS", "IP", "K-BB%", "ERA-", "FIP-"
]

tj_list_df.columns = new_column_names

tj_list_df.head()
```

Out[3]:

	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 [4]:

```
tj_list_df = tj_list_df.drop(tj_list_df.index[0])
tj_list_df.reset_index(drop=True, inplace=True)
tj_list_df.head()
```

Out[4]:

	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 [5]:
```
tj_list_df_filtered = tj_list_df[tj_list_df['Position'] == 'P']
tj_list_df_filtered.reset_index(drop=True, inplace=True)
tj_list_df_filtered
```

Out[5]:

	Player	TJ Surgery Date	Team	Level	Position	Throws	Country	High School	College
0	Johan Oviedo	12/1/2023	PIT	MLB	P	R	Cuba	NaN	Na
1	Jovani Moran	11/1/2023	MIN	AAA	P	L	Puerto Rico	NaN	Na
2	Taylor Broadway	10/18/2023	BOS	AA	P	R	United States	Texas	Mississip
3	Felix Bautista	10/9/2023	BAL	MLB	P	R	Dominican	NaN	Na
4	Sandy Alcantara	10/6/2023	MIA	MLB	P	R	Dominican	NaN	Na
...	...	...	...	...	...	...	...	...	
2186	Tom Candiotti	10/13/1981	MIL	AA	P	R	United States	California	St. Mary

In [6]:
```
tj_list_df_filtered['Surgeon(s)'].value_counts()
```

Out[6]:
```
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
```

Frank Jobe did the first TJ surgery.

May have useful data.

In [7]:
```
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
```

In [8]:
```
tj_list_df_filtered.drop(['High School', 'College(s)', 'Position', 'FI
tj_list_df_filtered
```

```
C:\Users\johns\AppData\Local\Temp\ipykernel_30356\2220704903.py:1: Sett
ingWithCopyWarning:
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 (ht
tps://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy)
  tj_list_df_filtered.drop(['High School', 'College(s)', 'Position', 'F
IP-', 'ERA-', 'K-BB%'], axis=1, inplace=True)
```

Out[8]:

	Player	TJ Surgery Date	Team	Level	Throws	Country	Age	Return Date (same level)	Recovery Time (months)
0	Johan Oviedo	12/1/2023	PIT	MLB	R	Cuba	25	NaN	NaN
1	Jovani	11/1/2023	MIN	AAA	L	Puerto	26	NaN	NaN

In [9]:
```
tj_list_df_filtered['Player'].value_counts()
```

Out[9]:
```
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 [10]:
```
isringhausen_df = tj_list_df_filtered[tj_list_df_filtered['Player'] ==
isringhausen_df
```

Out[10]:

	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 [11]:
```
mask = (tj_list_df_filtered['Player'] == 'Jason Isringhausen') & (tj_l

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

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_30356\2318486528.py:3: Settin
gWithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  tj_list_df_filtered.loc[mask, 'TJ Surgery Date'] = '9/1/2008'
```

In [12]:
```
tj_list_df_filtered['Game'].value_counts()
```

Out[12]:
```
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 [13]:
```
tj_list_df_filtered['Throws'].unique()
```

Out[13]:
```
array(['R', 'L', 'P', 'L/R'], dtype=object)
```

In [14]:
```
tj_list_df_filtered['Throws'].value_counts()
```

Out[14]:
```
Throws
R      1647
L       542
P         1
L/R       1
Name: count, dtype: int64
```

In [15]:
```
lr_tj = tj_list_df_filtered[tj_list_df_filtered['Throws'] == 'L/R']
lr_tj
```

Out[15]:

	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 [16]:
```
p_tj = tj_list_df_filtered[tj_list_df_filtered['Throws'] == 'P']
p_tj
```

Out[16]:

	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 [17]:

```
indices_to_drop = tj_list_df_filtered[
    (tj_list_df_filtered['Player'] == 'Ryan Perez') |
    (tj_list_df_filtered['Player'] == 'Michael Mercado')
    ].index
tj_list_df_filtered = tj_list_df_filtered.drop(indices_to_drop)
tj_list_df_filtered
```

Out[17]:

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

Make 'Throws' a binary value.

L = 0, R = 1

In [18]:

```
tj_list_df_filtered['Throws'] = tj_list_df_filtered['Throws'].replace(
tj_list_df_filtered['Throws'] = tj_list_df_filtered['Throws'].astype(i
tj_list_df_filtered.head()
```

```
C:\Users\johns\AppData\Local\Temp\ipykernel_30356\918451578.py:1: FutureW
arning: Downcasting behavior in `replace` is deprecated and will be remov
ed in a future version. To retain the old behavior, explicitly call `resu
lt.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[18]:

	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 [19]:
```
tj_list_df_filtered = tj_list_df_filtered.rename(columns={'Player': 'N
tj_list_df_filtered
```

Out[19]:

	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	6
1	Jovani Moran	11/1/2023	MIN	AAA	0	Puerto Rico	26	NaN	NaN	6
2	Taylor Broadway	10/18/2023	BOS	AA	1	United States	26	NaN	NaN	6
3	Felix Bautista	10/9/2023	BAL	MLB	1	Dominican	28	NaN	NaN	6
4	Sandy Alcantara	10/6/2023	MIA	MLB	1	Dominican	27	NaN	NaN	6
...	...	...	...	...	...	...	...	...	...	
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	1
2190	Tommy John	9/25/1974	LAD	MLB	0	United States	31	4/16/1976	19	1

2189 rows × 33 columns

In [20]:
```
tj_list_df_filtered.columns
```

Out[20]:
```
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', 'Yea
r',
       '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 [21]:

```
def remove_accents(input_str):
    nfkd_form = unicodedata.normalize('NFKD', input_str)
    return "".join([c for c in nfkd_form if not unicodedata.combining(

def clean_name(name):
    name = name.lower()
    name = remove_accents(name)
    name = re.sub(r'[-.]', '', name)
    name = re.sub(r'\s+', ' ', name).strip()
    return name

tj_list_df_filtered['Name'] = tj_list_df_filtered['Name'].apply(clean_
```

In [22]:

```
tj_list_df_filtered['Surgery'] = tj_list_df_filtered.groupby('Name')['
tj_list_df_filtered[['Name', 'TJ Surgery Date', 'Surgery']]
```

Out[22]:

	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 [23]:

```
isringhausen_df = tj_list_df_filtered[tj_list_df_filtered['Name'] == '
isringhausen_df
```

Out[23]:

	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 [24]:
```
tj_list_df_filtered
```

Out[24]:

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

In [25]:

```
# Identify duplicate columns by name
columns_seen = {}
duplicates = []

for col in tj_list_df_filtered.columns:
    if col in columns_seen:
        duplicates.append(col)
    else:
        columns_seen[col] = True

# Remove duplicates - keeping the first occurrence
tj_list_df_filtered = tj_list_df_filtered.loc[:,~tj_list_df_filtered.c

# Now, reorder the columns as intended
new_column_order = ['Name', 'Surgery'] + [col for col in tj_list_df_fi
tj_list_df_filtered = tj_list_df_filtered[new_column_order]

# Check the result
tj_list_df_filtered.head()
```

Out[25]:

	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 × 29 columns

In [26]:

```
tj_list_df_filtered['Date'] = pd.to_datetime(tj_list_df_filtered[['Yea
tj_list_df_filtered.head()
```

Out[26]:

	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 [27]:

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

Out[27]:

	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 [28]:

```
position = tj_list_df_filtered.columns.get_loc('Active') + 1
columns = list(tj_list_df_filtered.columns)
columns.remove('Date')
columns.insert(position, 'Date')

tj_list_df_filtered = tj_list_df_filtered[columns]

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	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 [29]:

```
tj_list_df_filtered['TJ Surgery Date'] = pd.to_datetime(tj_list_df_fil
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	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 [30]:
```
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
```

In [31]:
```
tj_list_df_filtered.columns
```

Out[31]:
```
Index(['Name', 'Surgery', 'TJ Surgery Date', 'Team', 'Level', 'Throws',
       'Country', 'Age', 'Return Date (same level)', 'Recovery Time (mont
hs)',
       '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', 'I
P'],
      dtype='object')
```

In [32]:

```
tj_list_df_filtered['Throws'] = tj_list_df_filtered['Throws'].astype('
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
```

In [33]:

```
tj_list_df_filtered['Age'] = tj_list_df_filtered['Age'].astype('int64'
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
```

In [34]:

```
# Group by 'Name' and aggregate 'TJ Surgery Date' into a list
surgery_dates_by_player = tj_list_df_filtered.groupby('Name')['TJ Surg

# Filter players with multiple surgeries
multiple_surgeries = surgery_dates_by_player[surgery_dates_by_player['

# Display the DataFrame with players who have had multiple surgeries
print(multiple_surgeries)

```

```
                  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 [35]:

```
# Create a DataFrame with lists of surgery dates for each player
surgery_dates_by_player = tj_list_df_filtered.groupby('Name')['TJ Surg

# Merge this information back to the original DataFrame
tj_list_df_filtered = pd.merge(tj_list_df_filtered, surgery_dates_by_p
```

In [36]:
```
tj_list_df_filtered
```

Out[36]:

	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	Na
1	jovani moran	1	2023-11-01	MIN	AAA	0	Puerto Rico	26	NaN	Na
2	taylor broadway	1	2023-10-18	BOS	AA	1	United States	26	NaN	Na
3	felix bautista	1	2023-10-09	BAL	MLB	1	Dominican	28	NaN	Na
4	sandy alcantara	1	2023-10-06	MIA	MLB	1	Dominican	27	NaN	Na
...	...	...	...	...	...	...	...	...	...	
2184	tom candiotti	1	1981-10-13	MIL	AA	1	United States	23	1/1/1983	Na
2185	bill bordley	1	1981-04-13	SF	MLB	0	United States	23	NaN	Na
2186	joe hesketh	1	1981-01-01	WAS	AA	0	United States	22	1/1/1983	Na
2187	brent strom	1	1978-01-01	SD	MLB	0	United States	28	NaN	Na
2188	tommy john	1	1974-09-25	LAD	MLB	0	United States	31	4/16/1976	

2189 rows × 28 columns

In [37]:
```
tj_list_df_filtered.drop(columns=['TJ Surgery Date_x'], inplace=True)
tj_list_df_filtered.rename(columns={'TJ Surgery Date_y': 'TJ Surgery D
```

In [38]:
```
isringhausen_df = tj_list_df_filtered[tj_list_df_filtered['Name'] == '
isringhausen_df
```

Out[38]:

	Name	Surgery	Team	Level	Throws	Country	Age	Return Date (same level)	Recovery Time (months)	mlba
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 [39]:
```
clean_tj_df = tj_list_df_filtered
```

In [40]:
```
clean_tj_df['Surgery'].value_counts()
```

Out[40]:
```
Surgery
1    1901
2     276
3      12
Name: count, dtype: int64
```

In [41]:
```
instances_with_surgery_3 = clean_tj_df[clean_tj_df['Surgery'] == 3]['N
instances_with_surgery_3
```

Out[41]:
```
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 [42]:
```
clean_tj_df.to_csv('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 [43]:

```
directory_path = os.path.expanduser('~/Documents/Flatiron/Project_5')
file_pattern = os.path.join(directory_path, '*_pitch_stats.csv')

pitch_csv_files = glob.glob(file_pattern)

dataframes = []
for file in pitch_csv_files:
    year = os.path.basename(file).split('_')[0]
    df = pd.read_csv(file)
    df['Year'] = year
    dataframes.append(df)

if dataframes:
    combined_df = pd.concat(dataframes, ignore_index=True)
    print(combined_df.head())
else:
    print("No CSV files were found.")
```

```
   Rk             Name  Age   Tm     IP   G  GS  Wgs  Lgs  ND  ...  RS/IP
\
0   1         Ed Acosta   28  SDP   89.0  46   2    1    1   0  ...    5.4
1   2  Doyle Alexander   21  BAL  106.1  35   9    2    4   3  ...    2.8
2   3       Lloyd Allen   22  CAL   85.1  42   6    0    4   2  ...    0.9
3   4       Steve Arlin   26  SDP  250.0  38  37    9   21   7  ...    2.9
4   5      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
0    7.0     NaN  NaN    NaN      NaN   NaN  NaN        acosted01  1972
1    6.8     NaN  NaN    NaN      NaN   NaN  NaN        alexado01  1972
2    4.8     NaN  NaN    NaN      NaN   NaN  NaN        allenll01  1972
3    6.7     NaN  NaN    NaN      NaN   NaN  NaN        arlinst01  1972
4    6.1     NaN  NaN    NaN      NaN   NaN  NaN        bahnsst01  1972

[5 rows x 39 columns]
```

In [44]:

```
combined_df.head()
```

Out[44]:

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

5 rows × 39 columns

In [45]:

```
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
```

In [46]:

```
pitch_combined_df = combined_df.drop(['Rk', 'Wgs', 'Lgs', 'ND', 'Wchp'
pitch_combined_df
```

Out[46]:

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

16573 rows × 18 columns

In [47]:
```
pitch_combined_df['Name'].value_counts()
```

Out[47]:
```
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 [48]:
```
pitch_combined_df['Throws'] = np.where(pitch_combined_df['Name'].str.e
pitch_combined_df
```

Out[48]:

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

16573 rows × 19 columns

In [49]:

```
new_column_order = [
    'Name', 'Age', 'Year', 'Throws', 'Tm', 'IP', 'G', 'GS', 'CG', 'SHO
    'IP/GS', 'Pit/GS', '<80', '80-99', '100-119', '≥120', 'Max'
]

pitch_combined_df = pitch_combined_df[new_column_order]

pitch_combined_df.head()
```

Out[49]:

	Name	Age	Year	Throws	Tm	IP	G	GS	CG	SHO	sDR	lDR	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 [50]:

```
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
```

In [51]:
```
pitch_combined_df['Tm'].value_counts()
```

Out[51]:
```
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
```

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 [52]:

```
pitch_combined_df['Tm'] = pitch_combined_df['Tm'].replace(['LAA', 'CAL
pitch_combined_df['Tm'] = pitch_combined_df['Tm'].replace(['TBD'], 'TB
pitch_combined_df['Tm'] = pitch_combined_df['Tm'].replace(['FLA'], 'MI
pitch_combined_df['Tm'] = pitch_combined_df['Tm'].replace(['MON'], 'WS

pitch_combined_df
```

Out[52]:

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

16573 rows × 19 columns

In [53]:

```
pitch_combined_df['Tm'].value_counts()
```

Out[53]:

```
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
```

In [54]:

```
tot_df = pitch_combined_df[pitch_combined_df['Tm'] == 'TOT']
tot_df
```

Out[54]:

	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	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	18	
16540	Ryan Weathers*	23	2023	0	TOT	57.2	15	12	0	0	0	10	
16543	Luke Weaver	29	2023	1	TOT	123.2	29	25	0	0	1	13	
16544	Luke Weaver	29	2023	1	TOT	26.2	8	4	0	0	1	3	
16568	Ryan Yarbrough*	31	2023	0	TOT	89.2	25	9	0	0	0	5	

1214 rows × 19 columns

In [55]:

```
verlander_df = pitch_combined_df[pitch_combined_df['Name'].str.contain
verlander_df
```

Out[55]:

	Name	Age	Year	Throws	Tm	IP	G	GS	CG	SHO	sDR	IDR	IP
10017	Justin Verlander	22	2005	1	DET	11.1	2	2	0	0	0	2	
10367	Justin Verlander	23	2006	1	DET	186.0	30	30	1	1	0	14	
10712	Justin Verlander	24	2007	1	DET	201.2	32	32	1	1	0	18	
11060	Justin Verlander	25	2008	1	DET	201.0	33	33	1	0	0	17	
11407	Justin Verlander	26	2009	1	DET	240.0	35	35	3	1	0	12	
11715	Justin Verlander	27	2010	1	DET	224.1	33	33	4	0	0	16	
12011	Justin Verlander	28	2011	1	DET	251.0	34	34	4	2	0	13	
12336	Justin Verlander	29	2012	1	DET	238.1	33	33	6	1	0	13	
12665	Justin Verlander	30	2013	1	DET	218.1	34	34	0	0	0	16	
13005	Justin Verlander	31	2014	1	DET	206.0	32	32	0	0	0	14	
13350	Justin Verlander	32	2015	1	DET	133.1	20	20	1	1	0	9	

In [56]:

```
# Define the function to replace 'TOT' with actual teams
def replace_tot_with_teams(row, df):
    if row['Tm'] != 'TOT':
        return row['Tm']

    player_teams = df[(df['Name'] == row['Name']) & (df['Year'] == row
    return '/'.join(player_teams)

# Apply the function to each row
pitch_combined_df['Tm'] = pitch_combined_df.apply(lambda row: replace_

# Display the updated DataFrame for Justin Verlander
pitch_combined_df[pitch_combined_df['Name'] == 'Justin Verlander'][['N

```

Out[56]:

Name	Year	Tm

In [57]:

```
pitch_combined_df['Tm'].unique()
```

Out[57]:

```
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/CL
E',
       'SFG/NYY', 'CIN/CHW', 'SDP/CIN', 'TEX/CHC', 'DET/CLE', 'SDP/CL
E',
       'MIN/TEX', 'ANA/NYY', 'HOU/STL', 'STL/ATL', 'CLE/NYY', 'MIN/NY
Y',
       'CHW/OAK', 'CLE/OAK', 'KCR/ATL', 'OAK/CHW', 'ATL/CLE', 'CHC/LA
D',
       '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/TE
X',
       'TOR/ATL', 'OAK/SEA/NYM', 'OAK/SEA', 'CIN/ANA', 'CHC/CHW',
       'NYM/CIN', 'OAK/NYY', 'PHI/WSN', 'OAK/TEX', 'CIN/NYM', 'KCR/SE
A',
```

In [58]:

```
pitch_combined_df
```

Out[58]:

	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 [59]:

```
combined_teams = pitch_combined_df[pitch_combined_df['Tm'].str.contain

# Step 2: Find and remove rows with individual teams for the same playe
for index, row in combined_teams.iterrows():
    names = row['Name']
    year = row['Year']
    # Extract the individual teams from the 'Tm' column
    teams = row['Tm'].split('/')
    # Remove rows where the team is one of the individual teams and the
    pitch_combined_df = pitch_combined_df[~((pitch_combined_df['Name']
                                            (pitch_combined_df['Year']
                                            (pitch_combined_df['Tm'].i

# Resulting DataFrame
pitch_combined_df
```

Out[59]:

	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 [60]:

```
pitch_combined_df['Name'] = pitch_combined_df['Name'].str.lower().str.
pitch_combined_df
```

Out[60]:

	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 [61]:

```
print(pitch_combined_df['Name'].unique())
```

```
['ed\xa0acosta' 'doyle\xa0alexander' 'lloyd\xa0allen' ... 'keaton\xa0win
n'
 'jackson\xa0wolf' 'bryan\xa0woo']
```

This is odd. Need to clean up.

In [62]:

```
pitch_combined_df['Name'] = pitch_combined_df['Name'].str.replace('\xa
pitch_combined_df['Name'] = pitch_combined_df['Name'].str.strip()
print(pitch_combined_df['Name'].unique())
```

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

In [63]:

```
def remove_accents(input_str):
    nfkd_form = unicodedata.normalize('NFKD', input_str)
    return "".join([c for c in nfkd_form if not unicodedata.combining(

def clean_name(name):
    name = name.lower()
    name = remove_accents(name)
    name = re.sub(r'[-.]', '', name)
    name = re.sub(r'\s+', ' ', name).strip()
    return name

pitch_combined_df['Name'] = pitch_combined_df['Name'].apply(clean_name
```

In [64]:

```
filtered_df = pitch_combined_df
```

In [65]:

```
filtered_df['Team'] = filtered_df['Tm']
filtered_df.drop(columns=['Tm'], inplace=True)
filtered_df.head()
```

Out[65]:

	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	NaN
1	doyle alexander	21	1972	1	106.1	35	9	2	2	3	4	6.8	NaN	NaN
2	lloyd allen	22	1972	1	85.1	42	6	0	0	5	0	4.8	NaN	NaN
3	steve arlin	26	1972	1	250.0	38	37	12	3	22	7	6.7	NaN	NaN
4	stan bahnsen	27	1972	1	252.1	43	41	5	1	32	4	6.1	NaN	NaN

In [66]:

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

Out[66]:

	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
1	doyle alexander	21	1972	1	106.1	35	9	2	2	3	...	6.8	NaN
2	lloyd allen	22	1972	1	85.1	42	6	0	0	5	...	4.8	NaN
3	steve arlin	26	1972	1	250.0	38	37	12	3	22	...	6.7	NaN
4	stan bahnsen	27	1972	1	252.1	43	41	5	1	32	...	6.1	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
16566	brandon woodruff	30	2023	1	67.0	11	11	1	1	0	...	6.1	95.0
16567	kyle wright	27	2023	1	31.0	9	7	0	0	0	...	3.7	72.0

In [67]:

```
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
```

In [68]:

```
tj_df = filtered_df[filtered_df['Name'] == 'tommy john']
tj_df
```

Out[68]:

	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

In [69]:

```
# Convert 'Career Start', 'Career End', and 'Year' to integers if not
filtered_df['Career Start'] = filtered_df['Career Start'].astype(int)
filtered_df['Career End'] = filtered_df['Career End'].astype(int)
filtered_df['Year'] = filtered_df['Year'].astype(int)

# Define a function to calculate inactive years for each player
def calculate_inactive_years(row):
    active_years = set(filtered_df[filtered_df['Name'] == row['Name']]
    all_years = set(range(row['Career Start'], row['Career End'] + 1))
    inactive_years = all_years - active_years
    return sorted(list(inactive_years))

# Apply the function to each row to calculate 'Inactive Years'
filtered_df['Inactive Years'] = filtered_df.apply(calculate_inactive_y

# Now, let's ensure there's only one 'Inactive Years' column and clean
columns_to_keep = ['Name', 'Age', 'Year', 'Throws', 'IP', 'G', 'GS', '
filtered_df = filtered_df[columns_to_keep]

# Display a snippet of the DataFrame to verify the results
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 [70]:

```
tj_df = filtered_df[filtered_df['Name'] == 'tommy john']
tj_df
```

Out[70]:

	Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	Inactiv Year
109	tommy john	29	1972	0	186.2	29	29	4	1	5	1972	1989	[197
348	tommy john	30	1973	0	218.0	36	31	4	2	5	1972	1989	[197
599	tommy john	31	1974	0	153.0	22	22	5	3	5	1972	1989	[197
1124	tommy john	33	1976	0	207.0	31	31	6	2	1	1972	1989	[197
1384	tommy john	34	1977	0	220.1	31	31	11	3	3	1972	1989	[197
1669	tommy john	35	1978	0	213.0	33	30	7	0	4	1972	1989	[197
1940	tommy john	36	1979	0	276.1	37	36	17	3	11	1972	1989	[197

In [71]:

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

Out[71]:

	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	0	2016	2023	[2017 2018 2019 2020 2021 2022
16571	rob zastryzny	31	2023	0	20.2	21	1	0	0	0	2016	2023	[2017 2018 2019 2020 2021 2022

This is wrong.

Zastryzny played 2016-2023.

Missing all years inbetween.

In [72]:

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

Out[72]:

	Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	Inac 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 [73]:

```
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 [74]:
```
filtered_df.to_csv('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 [75]:
```
historic_tj_df = filtered_df.merge(
    tj_list_df_filtered[['Name', 'Surgery', 'TJ Surgery Date', 'Surgeo
    on='Name',
    how='left'
)

historic_tj_df['Surgery'] = historic_tj_df['Surgery'].fillna(0)
historic_tj_df
```

Out[75]:

	Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	In
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	27	2023	1	31.0	9	7	0	0	0	2019	2023	

In [76]:
```
historic_tj_df['TJ Surgery Date'] = historic_tj_df['TJ Surgery Date'].
```

In [77]:
```
historic_tj_df.head()
```

Out[77]:

	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	[]
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	[1979, 1980]

In [78]:
```
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 [79]:
```
historic_tj_df['Surgery'].value_counts()
```

Out[79]:
```
Surgery
0.0    11946
1.0     2828
2.0      558
3.0       21
Name: count, dtype: int64
```

In [80]:
```
surgery_3 = historic_tj_df[historic_tj_df['Surgery'] == 3]['Name']
surgery_3
```

Out[80]:
```
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 [81]:
```
# Calculating the sum total of 'IP' for each player in the merged data
ip_sum = historic_tj_df.groupby('Name')['IP'].sum().sort_values(ascend
ip_sum.head(1200)
```

Out[81]:
```
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 [82]:

```
doyle_df = historic_tj_df[historic_tj_df['Name'] == 'doyle alexander']
doyle_df
```

Out[82]:

	Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	Ina Y
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	27	1978	1	191.0	31	28	7	1	2	1972	1989	

In [83]:

```
# Clear 'Total_IP' column first
historic_tj_df['Total_IP'] = None

# Calculate Total_IP for each player
player_total_ip = historic_tj_df.groupby('Name')['IP'].sum().reset_ind

# Merge info back to original df, only for the last year of play for e
historic_tj_df = historic_tj_df.merge(player_total_ip, on='Name', how=

# Identify the last year of play for each player
historic_tj_df['Last_Year_Flag'] = historic_tj_df.groupby('Name')['Yea

# Update 'Total_IP' only for the last year of play
historic_tj_df.loc[historic_tj_df['Year'] == historic_tj_df['Last_Year_

# Clean up df, remove unnecessary columns
historic_tj_df.drop(columns=['Total_IP_y', 'Last_Year_Flag'], inplace=
historic_tj_df.rename(columns={'Total_IP_x': 'Total_IP'}, inplace=True

historic_tj_df[historic_tj_df['Year'] == historic_tj_df['Year'].max()]
```

Out[83]:

	Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	Inacti Yea
14940	andrew abbott	24	2023	0	109.1	21	21	0	0	0	2023	2023	
14941	joan adon	24	2023	1	51.2	12	10	0	0	0	2021	2023	
14942	keegan akin	28	2023	0	23.2	24	1	0	0	1	2020	2023	
14943	sandy alcantara	27	2023	1	184.2	28	28	3	1	0	2018	2023	
14944	scott alexander	33	2023	0	48.1	55	8	0	0	7	2018	2023	[20 20 20

In [84]:

```
players_below_320_ip_recheck = historic_tj_df[historic_tj_df['Total_IP
historic_tj_df_temp_check = historic_tj_df[~historic_tj_df['Name'].isi
historic_tj_df_temp_check
```

Out[84]:

	Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	In
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	

In [85]:

```
historic_tj_df_temp_check['Name'].value_counts()
```

Out[85]:

```
Name
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 [86]:
```
min_ip_df = historic_tj_df[historic_tj_df['Total_IP'] >= 320.0][['Name
min_ip_df
```

Out[86]:

	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 [87]:
```
min_ip_df['Surgery'].value_counts()
```

Out[87]:
```
Surgery
0.0    1067
1.0     239
2.0      78
3.0       6
Name: count, dtype: int64
```

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

Out[88]:
```
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 [89]:

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

Out[89]:

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

In [90]:

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

Out[90]:

	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 [91]:

```
jason_df = new_min_ip_df[new_min_ip_df['Name'] == 'jason isringhausen'
jason_df
```

Out[91]:

	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 [92]:

```
new_min_ip_df = new_min_ip_df.drop_duplicates(subset=['Name'])
```

In [93]:
```
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	
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	[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	

1244 rows × 19 columns

In [94]:
```
new_min_ip_df['Surgery'].value_counts()
```

Out[94]:
```
Surgery
0.0    976
1.0    230
2.0     37
3.0      1
Name: count, dtype: int64
```

In [95]:

```
new_merge_df = new_min_ip_df
new_merge_df
```

Out[95]:

	Name	Age	Year	Throws	IP	G	GS	CG	SHO	sDR	Career Start	Career End	Inact Ye
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	[20 20
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 [96]:

```
def convert_surgery_date_to_year(item):
    # Handle cases where the item is 0, indicating no surgery
    if item == 0:
        return 0
    # Handle non-list cases, directly converting to year if it's a val
    elif not isinstance(item, list):
        return item.year if pd.notnull(item) else 0
    # Handle list of dates, extracting the year from each valid date
    else:
        return [date.year for date in item if pd.notnull(date)]

# Apply the function to the column
new_merge_df['TJ Surgery Year'] = new_merge_df['TJ Surgery Date'].appl

```

```
C:\Users\johns\AppData\Local\Temp\ipykernel_30356\209469094.py:13: Settin
gWithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  new_merge_df['TJ Surgery Year'] = new_merge_df['TJ Surgery Date'].apply
(convert_surgery_date_to_year)
```

In [97]:

```
new_merge_df.head()
```

Out[97]:

	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 [98]:
```
new_merge_df['TJ Surgery Year'].value_counts()
```

Out[98]:
```
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 [99]:
```
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 [100]:
```
new_merge_df['Total_IP'].sort_values()
```

Out[100]:
```
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 [101]:

```
new_merge_df['TJ Surgery Year'] = new_merge_df['TJ Surgery Year'].asty

# Convert string representation of list to actual list and then to int
# Using ast.literal_eval to safely evaluate the string as a Python exp
new_merge_df['TJ Surgery Year'] = new_merge_df['TJ Surgery Year'].appl

new_merge_df['TJ Surgery Year']
```

```
C:\Users\johns\AppData\Local\Temp\ipykernel_30356\3841373166.py:1: Settin
gWithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  new_merge_df['TJ Surgery Year'] = new_merge_df['TJ Surgery Year'].astyp
e(str)
C:\Users\johns\AppData\Local\Temp\ipykernel_30356\3841373166.py:5: Settin
gWithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-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[101]:

```
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 [102]:

```
john_df = new_merge_df[new_merge_df['Name'] == 'tommy john']
john_df
```

Out[102]:

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

In [103]:

```
def safe_convert_tj_year(entry):
    if isinstance(entry, str):  # If the entry is a string, attempt to
        try:
            # Attempt to evaluate the string to a Python object (list)
            evaluated = literal_eval(entry)
            if isinstance(evaluated, list):
                return [int(i) for i in evaluated if i != 0]
            else:
                return []
        except (ValueError, SyntaxError):
            # In case of evaluation error, return an empty list
            return []
    elif isinstance(entry, list):  # If the entry is already a list, p
        return [int(i) for i in entry if i != 0]
    else:
        # For any other data type, return an empty list
        return []

# Example usage on a simulated column (simulating 'TJ Surgery Year' wi
simulated_tj_year = ['[2014, 2016]', '[0]', '[]', [2018, 2019], 0, 'no

# Apply the safe conversion function
converted_tj_year = [safe_convert_tj_year(item) for item in simulated_

# Show the result of the conversion
converted_tj_year
```

Out[103]:

```
[[2014, 2016], [], [], [2018, 2019], [], []]
```

In [104]:

```
# Apply the safe conversion function to the 'TJ Surgery Year' column i
new_merge_df['TJ Surgery Year'] = new_merge_df['TJ Surgery Year'].appl
```

```
C:\Users\johns\AppData\Local\Temp\ipykernel_30356\1494688879.py:2: Settin
gWithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  new_merge_df['TJ Surgery Year'] = new_merge_df['TJ Surgery Year'].apply
(safe_convert_tj_year)
```

In [105]:
```
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 [106]:
```
new_merge_df['TJ Surgery Year'].value_counts()
```

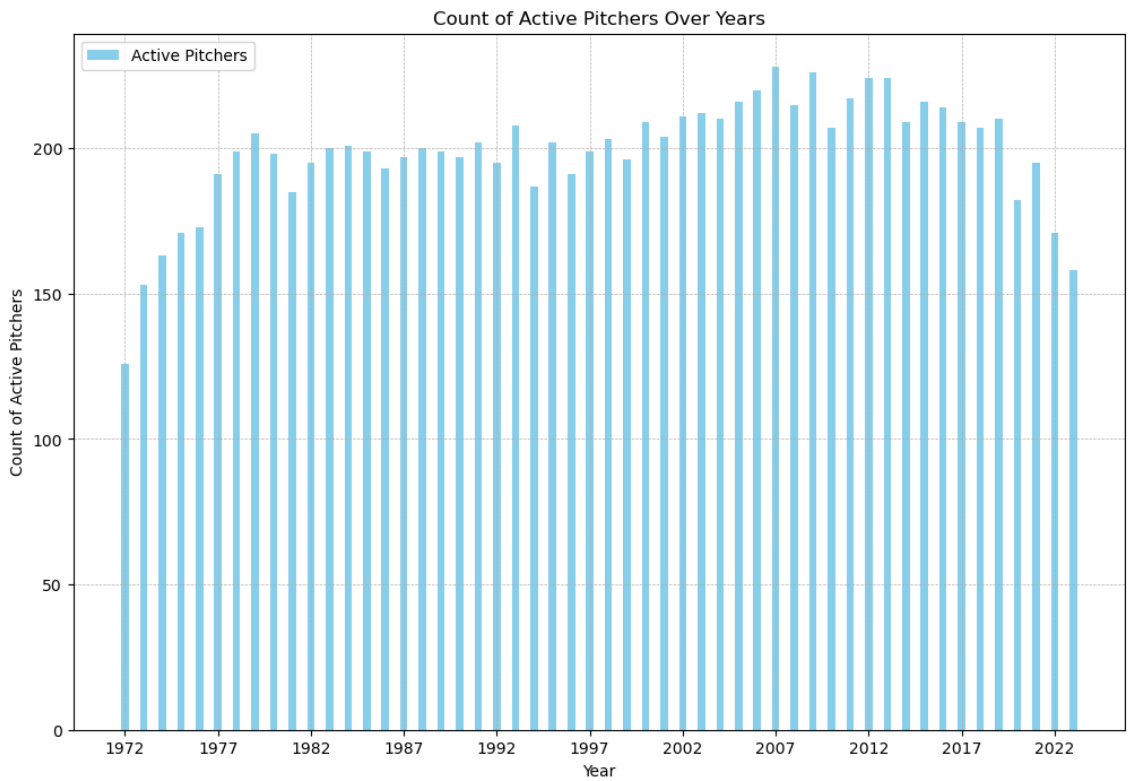
Out[106]:
```
TJ Surgery Year
[]              976
[2014]           16
[2003]           15
[2018]           12
[2016]           12
                ...
[2014, 2009]      1
[1991, 1990]      1
[2008, 2002]      1
[2012, 2005]      1
[2019, 2014]      1
Name: count, Length: 79, dtype: int64
```

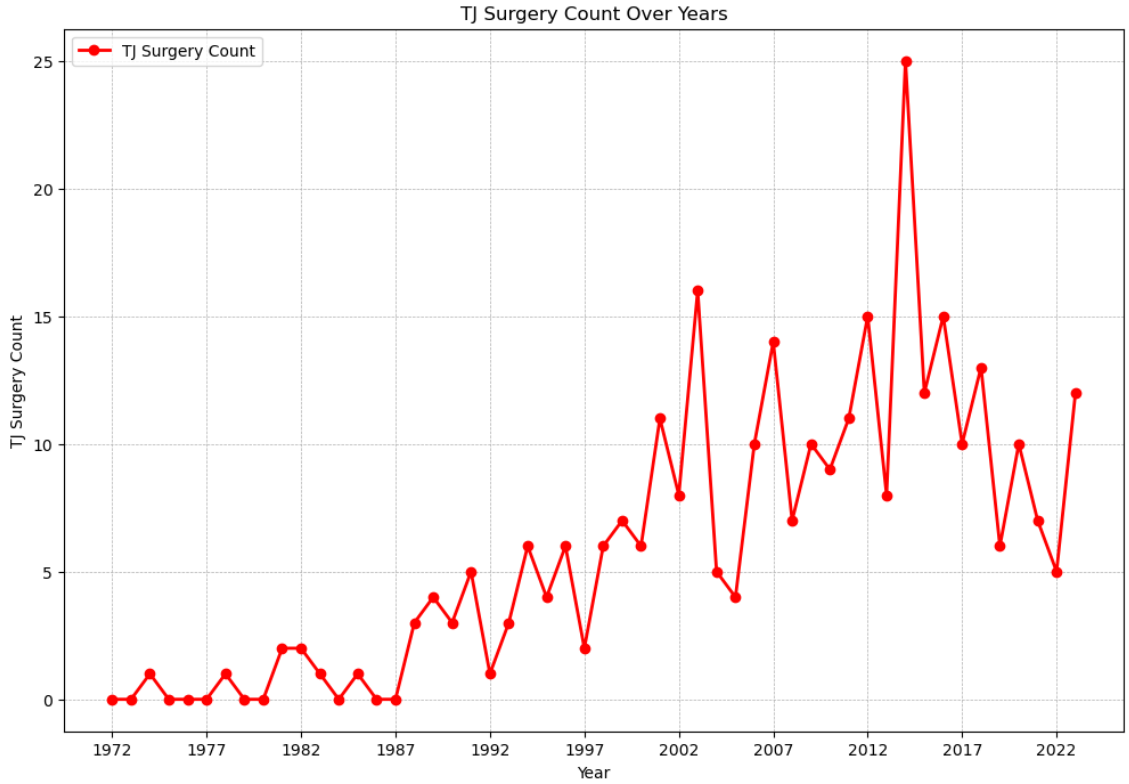
In [107]:

```
# Step 1: Determine the range of years
min_year = new_merge_df['Career Start'].min()
max_year = new_merge_df['Career End'].max()
years_range = np.arange(min_year, max_year + 1)

# Step 2: Initialize count series
active_pitchers_count = pd.Series(0, index=years_range)
tj_surgery_count = pd.Series(0, index=years_range)

# Step 3: Calculate active pitchers count
for index, row in new_merge_df.iterrows():
    active_years = set(range(row['Career Start'], row['Career End'] +
    for year in active_years:
        if year in active_pitchers_count.index:
            active_pitchers_count.loc[year] += 1

# Step 4: Calculate TJ surgery count
for index, row in new_merge_df.iterrows():
    for surgery_year in row['TJ Surgery Year']:
        if surgery_year in tj_surgery_count.index:
            tj_surgery_count.loc[surgery_year] += 1

```

In [108]:

```
min_year = active_pitchers_count.index.min()
max_year = active_pitchers_count.index.max()

fig, ax1 = plt.subplots(figsize=(12, 8))
ax1.bar(active_pitchers_count.index, active_pitchers_count.values, col
ax1.set_xlabel('Year')
ax1.set_ylabel('Count of Active Pitchers')
ax1.set_title('Count of Active Pitchers Over Years')
ax1.legend(loc='upper left')
ax1.grid(True, which='both', linestyle='--', linewidth=0.5)
ax1.set_xticks(np.arange(min_year, max_year + 1, 5))  # Adjusting x-ax
plt.show()

fig, ax2 = plt.subplots(figsize=(12, 8))
ax2.plot(tj_surgery_count.index, tj_surgery_count.values, color='red',
ax2.set_xlabel('Year')
ax2.set_ylabel('TJ Surgery Count')
ax2.set_title('TJ Surgery Count Over Years')
ax2.legend(loc='upper left')
ax2.grid(True, which='both', linestyle='--', linewidth=0.5)
ax2.set_xticks(np.arange(min_year, max_year + 1, 5))  # Adjusting x-ax
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 [109]:
```
new_merge_df.to_csv('hist_tj.csv')
```

In [ ]:
```

```