NBA Seasonal Analysis

Introduction

The NBA player dataset that is being investigated contains NBA player information for accumalated total stats per regular season and playoffs from 2012-2021 season. Our dataset consists of 29 columns such and 7293 rows.

Questions for our Analysis

Since we have data from 10 seasons it would be interesting to explore the following questions

- What players stats are correlated with each other?
- Distribution of Minutes compared to regular season and playoffs?
- Which players have the most Points and Assists?
- Which players have the most Rebounds and Blocks?
- Which players have the best Shooting percentages?
- How has the game changed in the past 10 years?

In [ ]:

```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
import numpy as np
import seaborn as sns
import pylab
from scipy import stats
import statsmodels.api as sm
import pylab
pio.renderers.default='plotly_mimetype+notebook'
#Ali
```

In [ ]:

```
data=pd.read_excel('/Users/araza/Downloads/nba_player_data.xlsx')
```

In [ ]:

```
data.columns
#data.head(10)
```

Out[ ]:

```
Index(['Year', 'Season_type', 'PLAYER_ID', 'RANK', 'PLAYER', 'TEAM', 'GP',
       'MIN', 'FGM', 'FGA', 'FG_PCT', 'FG3M', 'FG3A', 'FG3_PCT', 'FTM', 'FTA',
       'FT_PCT', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'TOV', 'PF',
       'PTS', 'EFF', 'AST_TOV', 'STL_TOV'],
      dtype='object')
```

In [ ]:

```
data.tail(10)
```

Out[ ]:

	Year	Season_type	PLAYER_ID	RANK	PLAYER	TEAM	GP	MIN	FGM	FGA	...	REB	AST	STL	BLK	TOV	PF	PTS	EFF	AST_TOV	STL_TOV
7283	2021-22	Playoffs	1630549	206	Day'Ron Sharpe	BKN	1	0	0	0	...	0	0	0	0	0	0	0	0	0.0	0.0
7284	2021-22	Playoffs	1630267	206	Facundo Campazzo	DEN	4	13	0	2	...	3	2	0	0	2	3	0	1	1.0	0.0
7285	2021-22	Playoffs	202066	206	Garrett Temple	NOP	1	2	0	0	...	1	0	0	0	0	0	0	1	0.0	0.0
7286	2021-22	Playoffs	1630552	206	Jalen Johnson	ATL	2	9	0	3	...	0	0	0	0	1	0	0	-4	0.0	0.0
7287	2021-22	Playoffs	1630215	206	Jared Butler	UTA	1	5	0	2	...	1	0	0	0	0	0	0	-1	0.0	0.0
7288	2021-22	Playoffs	1629006	206	Josh Okogie	MIN	1	2	0	0	...	0	0	0	0	0	0	0	0	0.0	0.0
7289	2021-22	Playoffs	1630556	206	Kessler Edwards	BKN	2	7	0	0	...	0	1	1	0	1	3	0	1	1.0	1.0
7290	2021-22	Playoffs	1630201	206	Malachi Flynn	TOR	6	36	0	7	...	3	3	1	0	1	6	0	-1	3.0	1.0
7291	2021-22	Playoffs	202693	206	Markieff Morris	MIA	2	3	0	1	...	1	0	0	0	1	2	0	-1	0.0	0.0
7292	2021-22	Playoffs	200794	206	Paul Millsap	PHI	1	6	0	0	...	1	1	0	0	0	1	0	2	0.0	0.0

10 rows × 29 columns

In [ ]:

```
data.shape # size of dataframe
```

Loading [MathJax]/extensions/Safe.js

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7293 entries, 0 to 7292
Data columns (total 29 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Year         7293 non-null   object 
 1   Season_type  7293 non-null   object 
 2   PLAYER_ID    7293 non-null   int64  
 3   RANK         7293 non-null   int64  
 4   PLAYER       7293 non-null   object 
 5   TEAM         7293 non-null   object 
 6   GP           7293 non-null   int64  
 7   MIN          7293 non-null   int64  
 8   FGM          7293 non-null   int64  
 9   FGA          7293 non-null   int64  
 10  FG_PCT       7293 non-null   float64
 11  FG3M         7293 non-null   int64  
 12  FG3A         7293 non-null   int64  
 13  FG3_PCT      7293 non-null   float64
 14  FTM          7293 non-null   int64  
 15  FTA          7293 non-null   int64  
 16  FT_PCT       7293 non-null   float64
 17  OREB         7293 non-null   int64  
 18  DREB         7293 non-null   int64  
 19  REB          7293 non-null   int64  
 20  AST          7293 non-null   int64  
 21  STL          7293 non-null   int64  
 22  BLK          7293 non-null   int64  
 23  TOV          7293 non-null   int64  
 24  PF           7293 non-null   int64  
 25  PTS          7293 non-null   int64  
 26  EFF          7293 non-null   int64  
 27  AST_TOV      7293 non-null   float64
 28  STL_TOV      7293 non-null   float64
dtypes: float64(5), int64(20), object(4)
memory usage: 1.6+ MB
```

Data Cleaning for analysis

In this section, we would try to point out instances and occurences in the dataset that might need some cleaning and wrangling. I will first check for null values and any duplicate values that we might need to take note of from our dataset.

In [ ]:

```
data.isna().sum() #checking for nulls
```

Out[ ]:

```
Year           0
Season_type    0
PLAYER_ID      0
RANK           0
PLAYER         0
TEAM           0
GP             0
MIN            0
FGM            0
FGA            0
FG_PCT         0
FG3M           0
FG3A           0
FG3_PCT        0
FTM            0
FTA            0
FT_PCT         0
OREB           0
DREB           0
REB            0
AST            0
STL            0
BLK            0
TOV            0
PF             0
PTS            0
EFF            0
AST_TOV        0
STL_TOV        0
dtype: int64
```

In [ ]:

```
boolean = data.duplicated(subset=['PLAYER_ID']).any() #looking for any duplicate player IDs that we need to be made aware of
boolean
```

Out[ ]:

```
True
```

In [ ]:

```
data.pivot_table(index=['PLAYER_ID'], aggfunc='size') # locating any repeating playerIds
```

Out[ ]:

```
PLAYER_ID
255        2
436        1
467        2
703        1
708        6
          ..
```

```
1630792    1
1630846    1
1630994    1
Length: 1386, dtype: int64
```

In [ ]:

```
data.loc[data['PLAYER_ID']==708]  # we can see that a playerID will be the same for players even if they switch teams
```

Out[ ]:

	Year	Season_type	PLAYER_ID	RANK	PLAYER	TEAM	GP	MIN	FGM	FGA	...	REB	AST	STL	BLK	TOV	PF	PTS	EFF	AST_TOV	STL
74	2012-13	Regular%20Season	708	75	Kevin Garnett	BOS	68	2022	422	850	...	530	159	78	62	110	154	1004	1252	1.45	
536	2012-13	Playoffs	708	69	Kevin Garnett	BOS	6	212	30	60	...	82	21	5	6	19	23	76	140	1.10	
927	2013-14	Regular%20Season	708	250	Kevin Garnett	BKN	54	1109	157	356	...	358	82	43	40	69	123	352	598	1.19	
1231	2013-14	Playoffs	708	72	Kevin Garnett	BKN	12	250	33	63	...	76	16	9	5	13	25	83	140	1.23	
1643	2014-15	Regular%20Season	708	281	Kevin Garnett	MIN	47	952	143	306	...	311	77	46	17	46	109	323	556	1.67	
2434	2015-16	Regular%20Season	708	373	Kevin Garnett	MIN	38	556	54	115	...	150	62	28	10	16	70	122	288	3.88	

6 rows × 29 columns

Looks like we had no nulls to be worried about and the duplicates that we noted in the dataset are players that played in playoffs and the regular season and their Player_ID will repeat for however many seasons they were actively playing.

In [ ]:

```
# getting rid of columns that we dont want to look at
data.drop(columns=["RANK"], inplace=True)
```

In [ ]:

```
#Create a new column to distinguish for season or playoffs instead of how Season_type column is provided
data["Playoffs_Or_Season"]=data.Season_type.apply(lambda x: "Regular Season" if len(x)>8 else "Playoffs")
```

In [ ]:

```
# Dropping Season_type column now
data.drop(columns=["Season_type"], inplace=True)
```

In [ ]:

```
data.tail(10)
```

Out[ ]:

	Year	PLAYER_ID	PLAYER	TEAM	GP	MIN	FGM	FGA	FG_PCT	FG3M	...	AST	STL	BLK	TOV	PF	PTS	EFF	AST_TOV	STL_TOV	Playoffs_C
7283	2021-22	1630549	Day'Ron Sharpe	BKN	1	0	0	0	0.0	0	...	0	0	0	0	0	0	0	0.0	0.0	
7284	2021-22	1630267	Facundo Campazzo	DEN	4	13	0	2	0.0	0	...	2	0	0	2	3	0	1	1.0	0.0	
7285	2021-22	202066	Garrett Temple	NOP	1	2	0	0	0.0	0	...	0	0	0	0	0	0	1	0.0	0.0	
7286	2021-22	1630552	Jalen Johnson	ATL	2	9	0	3	0.0	0	...	0	0	0	1	0	0	-4	0.0	0.0	
7287	2021-22	1630215	Jared Butler	UTA	1	5	0	2	0.0	0	...	0	0	0	0	0	0	-1	0.0	0.0	
7288	2021-22	1629006	Josh Okogie	MIN	1	2	0	0	0.0	0	...	0	0	0	0	0	0	0	0.0	0.0	
7289	2021-22	1630556	Kessler Edwards	BKN	2	7	0	0	0.0	0	...	1	1	0	1	3	0	1	1.0	1.0	
7290	2021-22	1630201	Malachi Flynn	TOR	6	36	0	7	0.0	0	...	3	1	0	1	6	0	-1	3.0	1.0	
7291	2021-22	202693	Markieff Morris	MIA	2	3	0	1	0.0	0	...	0	0	0	1	2	0	-1	0.0	0.0	
7292	2021-22	200794	Paul Millsap	PHI	1	6	0	0	0.0	0	...	1	0	0	0	1	0	2	0.0	0.0	

10 rows × 28 columns

In [ ]:

```
# clean up the year column
data["season_start_year"]=data["Year"].str[:4].astype(int)
```

In [ ]:

```
data.head(10)
```

Out[ ]:

	Year	PLAYER_ID	PLAYER	TEAM	GP	MIN	FGM	FGA	FG_PCT	FG3M	...	STL	BLK	TOV	PF	PTS	EFF	AST_TOV	STL_TOV	Playoffs_Or_Se
[illegible]	[illegible]	[illegible]	Kevin Durant	OKC	81	3119	731	1433	0.510	139	...	116	105	280	143	2280	2462	1.34	0.41	Regular Se

Loading [MathJax]/extensions/Safe.js

	Year	PLAYER_ID	PLAYER	TEAM	GP	MIN	FGM	FGA	FG_PCT	FG3M	...	STL	BLK	TOV	PF	PTS	EFF	AST_TOV	STL_TOV	Playoffs_Or_Se
1	2012-13	977	Kobe Bryant	LAL	78	3013	738	1595	0.463	132	...	106	25	287	173	2133	1921	1.63	0.37	Regular S
2	2012-13	2544	LeBron James	MIA	76	2877	765	1354	0.565	103	...	129	67	226	110	2036	2446	2.44	0.57	Regular S
3	2012-13	201935	James Harden	HOU	78	2985	585	1337	0.438	179	...	142	38	295	178	2023	1872	1.54	0.48	Regular S
4	2012-13	2546	Carmelo Anthony	NYK	67	2482	669	1489	0.449	157	...	52	32	175	205	1920	1553	0.98	0.30	Regular S
5	2012-13	201566	Russell Westbrook	OKC	82	2861	673	1535	0.438	97	...	145	24	273	189	1903	1857	2.22	0.53	Regular S
6	2012-13	201939	Stephen Curry	GSW	78	2983	626	1388	0.451	272	...	126	12	240	198	1786	1746	2.25	0.53	Regular S
7	2012-13	101145	Monta Ellis	MIL	82	3076	597	1436	0.416	94	...	169	36	254	164	1577	1416	1.95	0.67	Regular S
8	2012-13	203081	Damian Lillard	POR	82	3167	553	1288	0.429	185	...	74	19	243	172	1562	1415	2.19	0.30	Regular S
9	2012-13	200746	LaMarcus Aldridge	POR	74	2790	638	1318	0.484	2	...	62	91	143	187	1560	1686	1.34	0.43	Regular S

10 rows × 29 columns

In [ ]:
```
data.TEAM.nunique() #we have 31 teams in the NBA we only have 30
```
Out[ ]:
```
31
```

In [ ]:
```
data.TEAM.unique()  # We have have to NOH anf NOP which are the same team but they channged their name
```
Out[ ]:
```
array(['OKC', 'LAL', 'MIA', 'HOU', 'NYK', 'GSW', 'MIL', 'POR', 'TOR',
       'BKN', 'CHA', 'LAC', 'BOS', 'UTA', 'PHI', 'IND', 'SAS', 'ATL',
       'CLE', 'NOH', 'DET', 'CHI', 'SAC', 'DAL', 'DEN', 'MEM', 'PHX',
       'ORL', 'MIN', 'WAS', 'NOP'], dtype=object)
```

In [ ]:
```
data["TEAM"].replace(to_replace=['NOP','NOH'],value='NO', inplace=True)
```

In [ ]:
```
data.TEAM.nunique() # we have 30 teams now
```
Out[ ]:
```
30
```

In [ ]:
```
# creating seprate df for playoffs and regular season
rs_df=data[data["Playoffs_Or_Season"]=="Regular Season"]
playoff_df=data[data["Playoffs_Or_Season"]=="Playoffs"]
rs_df.columns
```
Out[ ]:
```
Index(['Year', 'PLAYER_ID', 'PLAYER', 'TEAM', 'GP', 'MIN', 'FGM', 'FGA',
       'FG_PCT', 'FG3M', 'FG3A', 'FG3_PCT', 'FTM', 'FTA', 'FT_PCT', 'OREB',
       'DREB', 'REB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS', 'EFF',
       'AST_TOV', 'STL_TOV', 'Playoffs_Or_Season', 'season_start_year'],
      dtype='object')
```

In [ ]:
```
# we need to look at a per minute played basis in order to fairly justify any correl
```

Below I will creat my own lists of columns that I want to keep from the datasets for the analysis in the next steps

In [ ]:
```
data.columns
```
Out[ ]:
```
Index(['Year', 'PLAYER_ID', 'PLAYER', 'TEAM', 'GP', 'MIN', 'FGM', 'FGA',
       'FG_PCT', 'FG3M', 'FG3A', 'FG3_PCT', 'FTM', 'FTA', 'FT_PCT', 'OREB',
       'DREB', 'REB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS', 'EFF',
       'AST_TOV', 'STL_TOV', 'Playoffs_Or_Season', 'season_start_year'],
      dtype='object')
```

In [ ]:
```
total_cols=['MIN','FGM','FGA','FG3M','FG3A','FTM'
            ,'FTA','OREB','DREB','REB','AST','STL','BLK'
            ,'TOV','PF','PTS','GP']
```

What players stats are correlated with each other

We will divide our stats by minutes played because our data are totals and to give a fair representations we will like to look at the total stats at a per minute played basis, in order to do this we will group by player,player_id and season to find the correlation.

In [ ]:
```
per_min_data=data.groupby(['PLAYER','PLAYER_ID','season_start_year'])[total_cols].sum().reset_index()
```
Loading [MathJax]/extensions/Safe.js

In [ ]:
```
for col in per_min_data.columns[4:]:
    per_min_data[col]=per_min_data[col]/per_min_data['MIN']


per_min_data
```

Out[ ]:

	PLAYER	PLAYER_ID	season_start_year	MIN	FGM	FGA	FG3M	FG3A	FTM	FTA	OREB	DREB	REB	
0	AJ Hammons	1627773	2016	163	0.104294	0.257669	0.030675	0.061350	0.055215	0.122699	0.049080	0.171779	0.220859	0.02
1	AJ Price	201985	2012	1278	0.125978	0.323161	0.054773	0.156495	0.038341	0.048513	0.015649	0.073552	0.089202	0.16
2	AJ Price	201985	2013	99	0.191919	0.464646	0.060606	0.222222	0.000000	0.020202	0.010101	0.090909	0.101010	0.13
3	AJ Price	201985	2014	324	0.157407	0.422840	0.046296	0.175926	0.049383	0.074074	0.018519	0.080247	0.098765	0.14
4	Aaron Brooks	201166	2012	1064	0.146617	0.328947	0.047932	0.134398	0.040414	0.053571	0.015038	0.068609	0.083647	0.11
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
5150	Zion Williamson	1629627	2019	668	0.314371	0.538922	0.008982	0.020958	0.170659	0.266467	0.095808	0.128743	0.224551	0.07
5151	Zion Williamson	1629627	2020	2026	0.312932	0.511846	0.004936	0.016782	0.182132	0.261106	0.082428	0.135242	0.217670	0.11
5152	Zoran Dragic	204054	2014	75	0.146667	0.400000	0.040000	0.186667	0.040000	0.066667	0.066667	0.040000	0.106667	0.06
5153	Zylan Cheatham	1629597	2019	51	0.117647	0.176471	0.000000	0.019608	0.000000	0.000000	0.058824	0.117647	0.176471	0.05
5154	Zylan Cheatham	1629597	2021	5	0.000000	0.600000	0.000000	0.400000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00

5155 rows × 20 columns

In [ ]:
```
per_min_data['FG%']=per_min_data['FGM']/per_min_data['FGA']
per_min_data['3PT%']=per_min_data['FG3M']/per_min_data['FG3A']
per_min_data['FT%']=per_min_data['FTM']/per_min_data['FTA']
per_min_data['FG3A%']=per_min_data['FG3A']/per_min_data['FGA']
per_min_data['FTA/FGA']=per_min_data['FTA']/per_min_data['FGA']
per_min_data['FG3M/FGM']=per_min_data['FG3M']/per_min_data['FGM']
per_min_data['AST_to_TOV']=per_min_data['AST']/per_min_data['TOV']
per_min_data['TRU%']=0.5*per_min_data['PTS']/(per_min_data['FGA']+0.475*per_min_data['FTA'])
per_min_data['Avg_ORB']=per_min_data['OREB']/per_min_data['REB']
per_min_data['Avg_DRB']=per_min_data['DREB']/per_min_data['REB']

per_min_data
```

Out[ ]:

	PLAYER	PLAYER_ID	season_start_year	MIN	FGM	FGA	FG3M	FG3A	FTM	FTA	...	FG%	3PT%	FT%
0	AJ Hammons	1627773	2016	163	0.104294	0.257669	0.030675	0.061350	0.055215	0.122699	...	0.404762	0.500000	0.450000
1	AJ Price	201985	2012	1278	0.125978	0.323161	0.054773	0.156495	0.038341	0.048513	...	0.389831	0.350000	0.790323
2	AJ Price	201985	2013	99	0.191919	0.464646	0.060606	0.222222	0.000000	0.020202	...	0.413043	0.272727	0.000000
3	AJ Price	201985	2014	324	0.157407	0.422840	0.046296	0.175926	0.049383	0.074074	...	0.372263	0.263158	0.666667
4	Aaron Brooks	201166	2012	1064	0.146617	0.328947	0.047932	0.134398	0.040414	0.053571	...	0.445714	0.356643	0.754386
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5150	Zion Williamson	1629627	2019	668	0.314371	0.538922	0.008982	0.020958	0.170659	0.266467	...	0.583333	0.428571	0.640449
5151	Zion Williamson	1629627	2020	2026	0.312932	0.511846	0.004936	0.016782	0.182132	0.261106	...	0.611379	0.294118	0.697543
5152	Zoran Dragic	204054	2014	75	0.146667	0.400000	0.040000	0.186667	0.040000	0.066667	...	0.366667	0.214286	0.600000
5153	Zylan Cheatham	1629597	2019	51	0.117647	0.176471	0.000000	0.019608	0.000000	0.000000	...	0.666667	0.000000	NaN
5154	Zylan Cheatham	1629597	2021	5	0.000000	0.600000	0.000000	0.400000	0.000000	0.000000	...	0.000000	0.000000	NaN

5155 rows × 30 columns

Now we want to only see players who played atleast a minimum of 50 mins total per season.

In [ ]:
```
(per_min_data['MIN']>=50).mean()# looking to make 50 mins played in the entire season as our minimum since close to 92% of our d
```

Out[ ]: 0.9212415130940834

Loading [MathJax]/extensions/Safe.js

In [ ]:
```
per_min_data = per_min_data[per_min_data['MIN']>=50]
per_min_data
```

Out[ ]:

	PLAYER	PLAYER_ID	season_start_year	MIN	FGM	FGA	FG3M	FG3A	FTM	FTA	...	FG%	3PT%	FT%
0	AJ Hammons	1627773	2016	163	0.104294	0.257669	0.030675	0.061350	0.055215	0.122699	...	0.404762	0.500000	0.450000
1	AJ Price	201985	2012	1278	0.125978	0.323161	0.054773	0.156495	0.038341	0.048513	...	0.389831	0.350000	0.790323
2	AJ Price	201985	2013	99	0.191919	0.464646	0.060606	0.222222	0.000000	0.020202	...	0.413043	0.272727	0.000000
3	AJ Price	201985	2014	324	0.157407	0.422840	0.046296	0.175926	0.049383	0.074074	...	0.372263	0.263158	0.666667
4	Aaron Brooks	201166	2012	1064	0.146617	0.328947	0.047932	0.134398	0.040414	0.053571	...	0.445714	0.356643	0.754386
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5149	Ziaire Williams	1630533	2021	1514	0.141347	0.314399	0.057464	0.183620	0.036328	0.044914	...	0.449580	0.312950	0.808824
5150	Zion Williamson	1629627	2019	668	0.314371	0.538922	0.008982	0.020958	0.170659	0.266467	...	0.583333	0.428571	0.640449
5151	Zion Williamson	1629627	2020	2026	0.312932	0.511846	0.004936	0.016782	0.182132	0.261106	...	0.611379	0.294118	0.697543
5152	Zoran Dragic	204054	2014	75	0.146667	0.400000	0.040000	0.186667	0.040000	0.066667	...	0.366667	0.214286	0.600000
5153	Zylan Cheatham	1629597	2019	51	0.117647	0.176471	0.000000	0.019608	0.000000	0.000000	...	0.666667	0.000000	NaN

4749 rows × 30 columns

In [ ]:
```
per_min_data.drop(columns='PLAYER_ID',inplace=True)
```

```
/Users/araza/opt/anaconda3/lib/python3.8/site-packages/pandas/core/frame.py:4308: 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-ver
sus-a-copy
```

In [ ]:
```
per_min_data.corr()
```

Out[ ]:

	season_start_year	MIN	FGM	FGA	FG3M	FG3A	FTM	FTA	OREB	DREB	...	FG%
season_start_year	1.000000	-0.111548	0.107807	0.081735	0.246270	0.269883	-0.003356	-0.026292	-0.027885	0.080990	...	0.066756
MIN	-0.111548	1.000000	0.433333	0.349036	0.178089	0.096533	0.341287	0.270328	-0.107239	0.043372	...	0.213041
FGM	0.107807	0.433333	1.000000	0.873006	0.180439	0.128051	0.606417	0.560888	0.083727	0.224036	...	0.406107
FGA	0.081735	0.349036	0.873006	1.000000	0.409439	0.427795	0.549125	0.457985	-0.208121	-0.017605	...	-0.065531
FG3M	0.246270	0.178089	0.180439	0.409439	1.000000	0.958148	-0.026343	-0.169342	-0.620638	-0.398729	...	-0.368385
FG3A	0.269883	0.096533	0.128051	0.427795	0.958148	1.000000	-0.040917	-0.181771	-0.654959	-0.426937	...	-0.499887
FTM	-0.003356	0.341287	0.606417	0.549125	-0.026343	-0.040917	1.000000	0.951315	0.101340	0.194898	...	0.205978
FTA	-0.026292	0.270328	0.560888	0.457985	-0.169342	-0.181771	0.951315	1.000000	0.252285	0.296213	...	0.287117
OREB	-0.027885	-0.107239	0.083727	-0.208121	-0.620638	-0.654959	0.101340	0.252285	1.000000	0.681661	...	0.560845
DREB	0.080990	0.043372	0.224036	-0.017605	-0.398729	-0.426937	0.194898	0.296213	0.681661	1.000000	...	0.475532
REB	0.041634	-0.016957	0.183577	-0.100131	-0.527283	-0.560509	0.171822	0.302929	0.875480	0.950363	...	0.552582
AST	0.072351	0.244129	0.219162	0.329416	0.159518	0.180099	0.246573	0.169153	-0.390062	-0.246848	...	-0.170991
STL	0.007010	0.068363	-0.018783	0.029896	0.002777	0.030084	0.040755	0.041324	-0.134719	-0.132451	...	-0.098598
BLK	-0.001369	-0.052983	0.074562	-0.167451	-0.439309	-0.466425	0.073147	0.187604	0.618420	0.551969	...	0.473743
TOV	-0.068441	0.139789	0.394146	0.419311	-0.072549	-0.049827	0.454391	0.457685	-0.008295	0.105683	...	0.027284
PF	-0.043430	-0.385106	-0.195538	-0.322980	-0.400384	-0.395855	-0.085690	0.020942	0.491974	0.329409	...	0.213223
PTS	0.133439	0.459900	0.956170	0.896883	0.348590	0.296345	0.736594	0.656697	-0.047077	0.133531	...	0.277666
GP	0.004242	-0.685000	-0.346481	-0.276294	-0.175722	-0.110710	-0.222722	-0.142996	0.128554	-0.010128	...	-0.172159
FG%	0.066756	0.213041	0.406107	-0.065531	-0.368385	-0.499887	0.205978	0.287117	0.560845	0.475532	...	1.000000
3PT%	0.107843	0.168443	0.116770	0.169510	0.577978	0.465828	-0.004717	-0.094310	-0.414312	-0.256467	...	-0.079229
FT%	0.076496	0.255956	0.225496	0.326590	0.403877	0.383952	0.283721	0.029268	-0.375695	-0.248103	...	-0.159242
FG3A%	0.253652	-0.024285	-0.226480	0.049382	0.845528	0.886661	-0.286759	-0.410173	-0.663681	-0.475308	...	-0.566746
FTA/FGA	-0.072840	0.019731	0.030657	-0.138717	-0.399516	-0.424219	0.590675	0.708676	0.397886	0.307806	...	0.334414
FG3M/FGM	0.209035	0.004088	-0.214274	0.040825	0.868526	0.854612	-0.282579	-0.401815	-0.632058	-0.458497	...	-0.521511

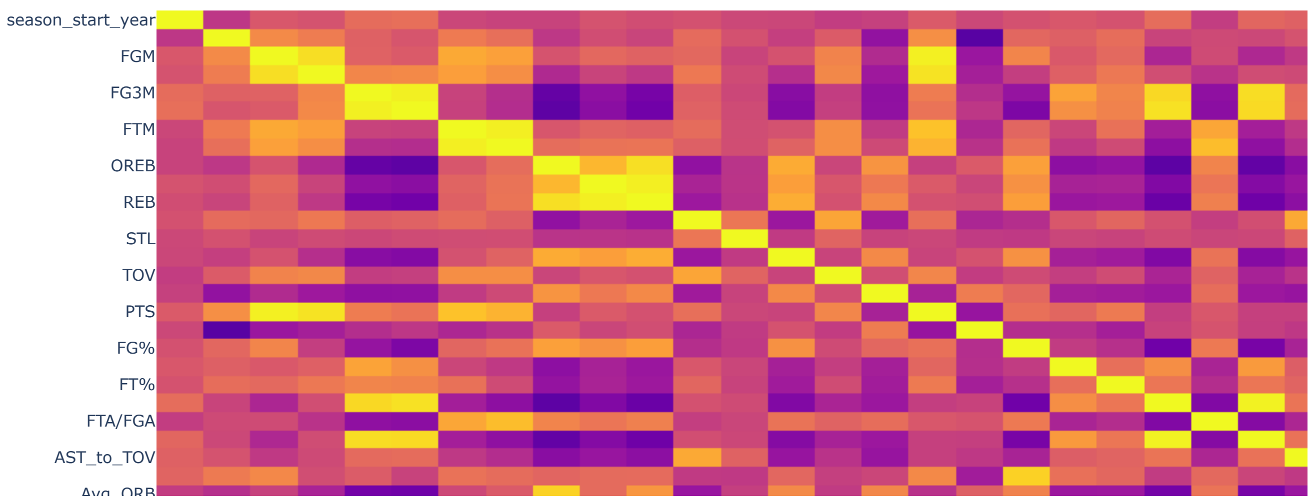
Loading [MathJax]/extensions/Safe.js

	season_start_year	MIN	FGM	FGA	FG3M	FG3A	FTM	FTA	OREB	DREB	...	FG%
AST_to_TOV	0.174095	0.081535	-0.095543	0.020900	0.234463	0.250394	-0.099926	-0.180862	-0.432835	-0.356301	...	-0.252204
TRU%	0.196349	0.338836	0.422043	0.045582	0.140892	-0.030552	0.287822	0.250926	0.202090	0.239733	...	0.812254
Avg_ORB	-0.073953	-0.162415	-0.036846	-0.248415	-0.547197	-0.564546	0.009490	0.131519	0.813880	0.240485	...	0.387304
Avg_DRB	0.073953	0.162415	0.036846	0.248415	0.547197	0.564546	-0.009490	-0.131519	-0.813880	-0.240485	...	-0.387304

28 rows × 28 columns

Now we have a correlation heatmap below between stats

In [ ]:
```
fig = px.imshow(per_min_data.corr(),aspect="auto")
fig.show(renderer='notebook')
```



What players stats are correlated with each other?

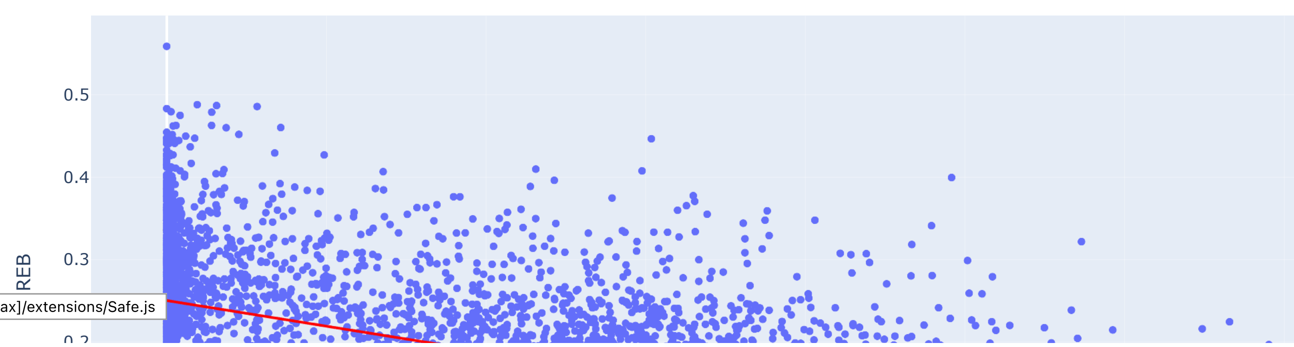
From the heatmap that w have above we can see the correlations between player stats.

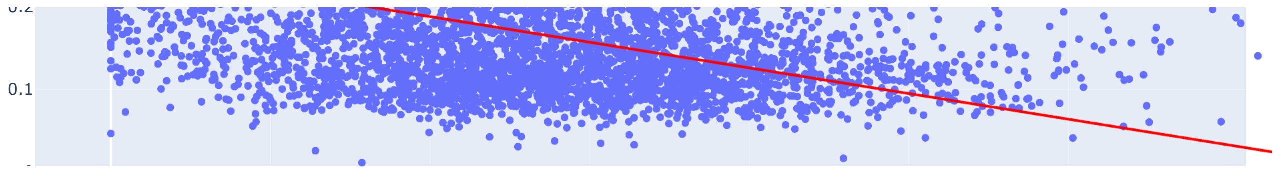
One that stands out the most is that REB and FG3A are strongly negatively correlated (-0.527), This does tend to make sense as a player who tends to shoot alot of threes will not usually be a great rebounder. This can be due to the fact that most players who shoot alot of threes are not in a great position to rebound.

When looking at rebounding and blocks we can see they are strongly positively correlated(0.627). This also makes sense as a players who is near the rim playing a more defensive role in protecting the basketball tends to go for more blocks and in better rebounding position.

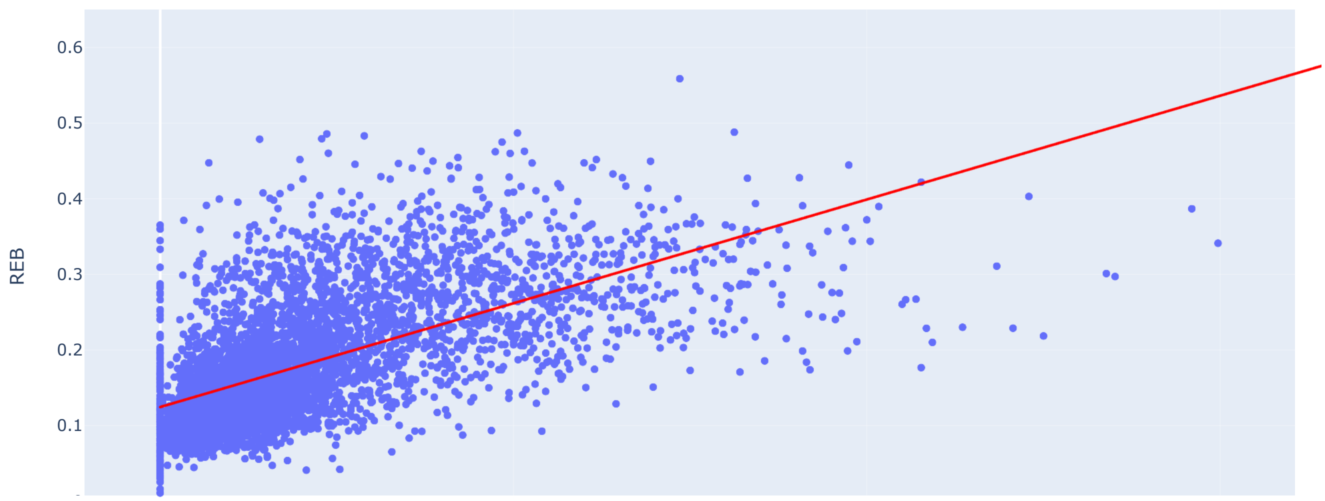
Below are the scatterplot distribution for those 2 specific correlations, to furhter justify our findings.

In [ ]:
```
fig = px.scatter(per_min_data, x=per_min_data['FG3A'], y=per_min_data['REB'], trendline="ols",trendline_color_override="red")
fig.show(renderer='notebook')
```





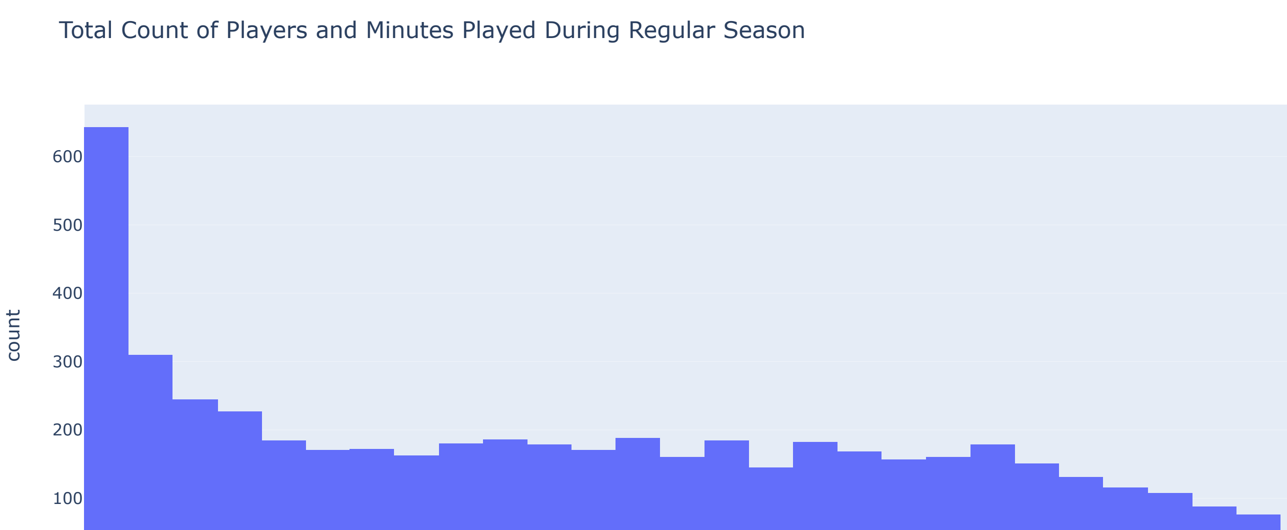
In [ ]:
```
fig = px.scatter(per_min_data, x=per_min_data['BLK'], y=per_min_data['REB'], trendline="ols",trendline_color_override="red")
fig.show()
```



Distribution of Minutes

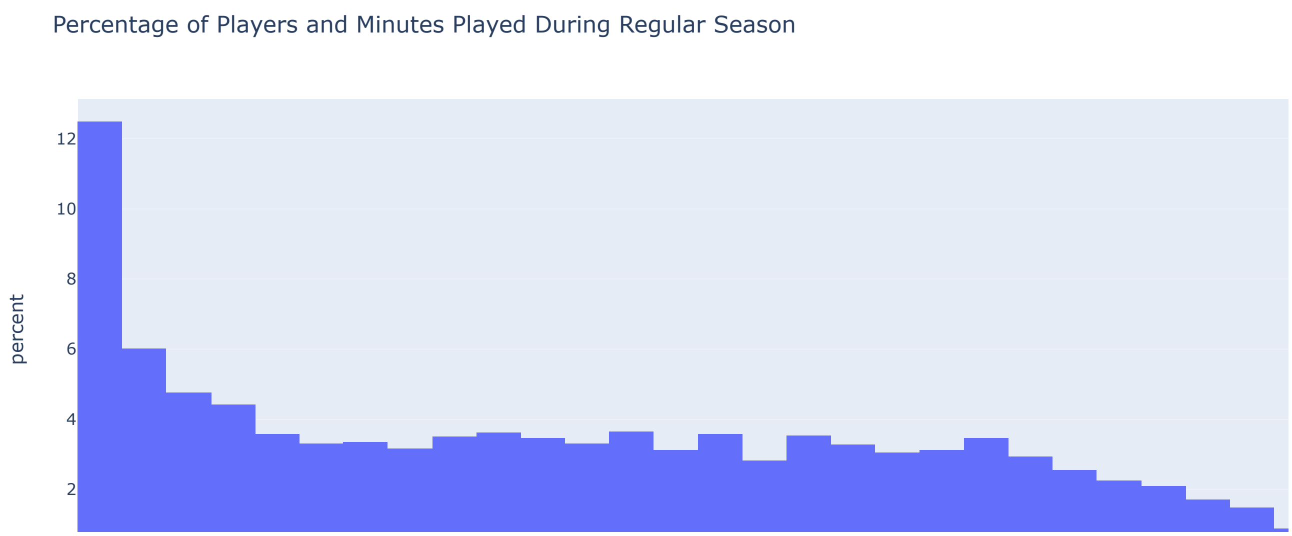
We will look at our minutes distribution in our dataset for regualar season and also playoffs and see if we notice any differences in the way the minutes our distributed between those two different time periods of the season.

In [ ]:
```
fig=px.histogram(rs_df,x='MIN', title='Total Count of Players and Minutes Played During Regular Season')
fig.show()
```



We can from the histogram above that 642 players play less than 100 mins season, and as we increase the minutes the less count of total players we get, showing that alot of NBA players actually dont get to play too many minutes

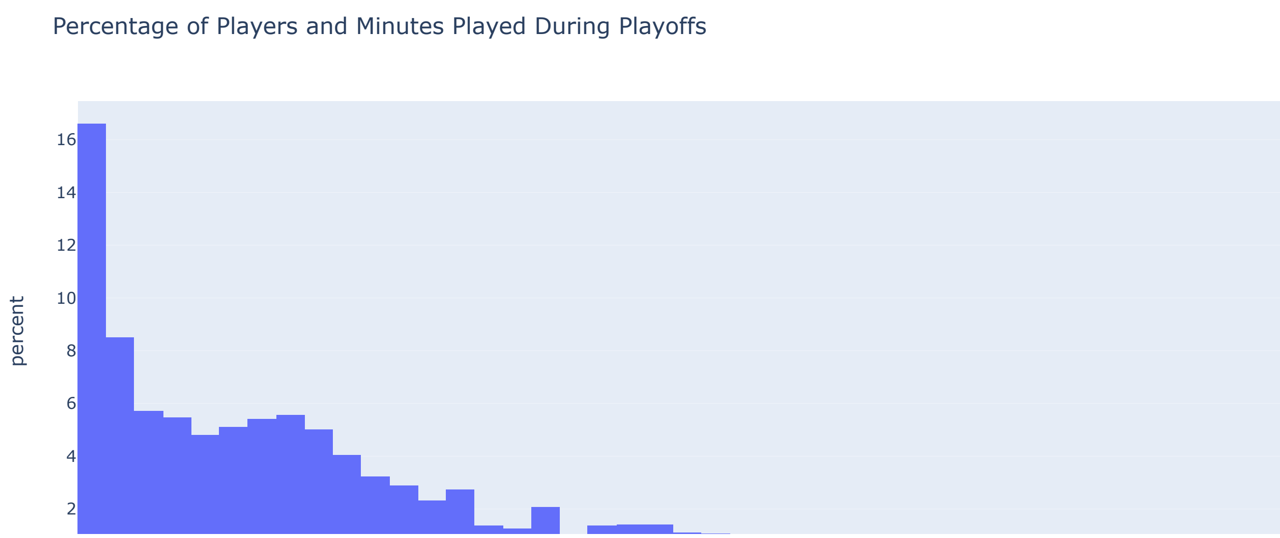
In [ ]:
```
fig=px.histogram(rs_df,x='MIN',histnorm='percent',title='Percentage of Players and Minutes Played During Regular Season')
fig.show()
```



Above histogram gives us the percentages instead of count for the minutes distribution for regular season.

Below we will graph the same histogram but look at playoffs dataframe

In [ ]:
```
fig=px.histogram(playoff_df,x='MIN',histnorm='percent',title='Percentage of Players and Minutes Played During Playoffs')
fig.show()
```

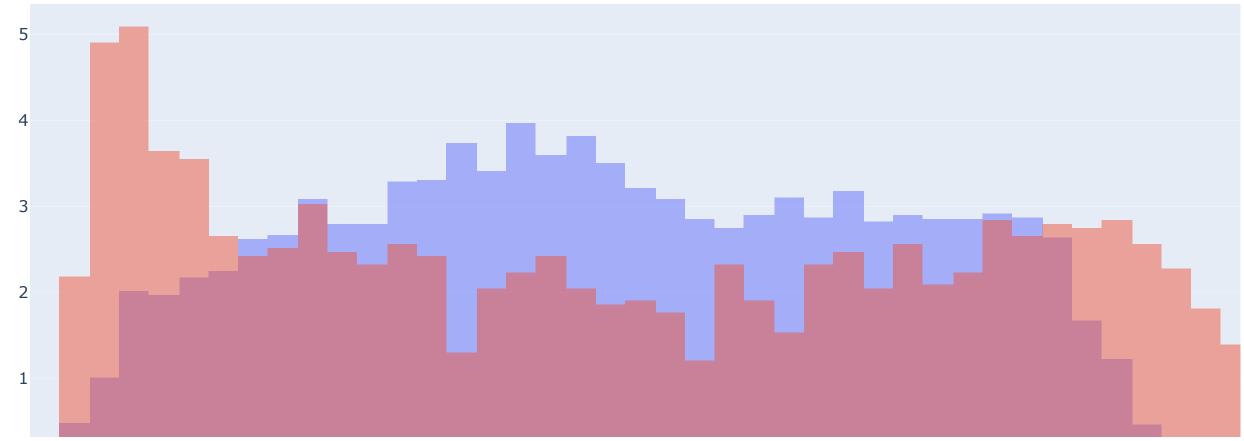


Distribution of Minutes compared to regular season and playoffs?

In [ ]:
```
fig=go.Figure()
fig.add_trace(go.Histogram(x=rs_df['MIN']/rs_df['GP'],histnorm='percent',name='RS',
                           xbins={'start':0,'end':46,'size':1}))

fig.add_trace(go.Histogram(x=playoff_df['MIN']/playoff_df['GP'],histnorm='percent',name='PLAYOFFS',
                           xbins={'start':0,'end':46,'size':1}))


fig.update_layout(barmode='overlay')
fig.update_traces(opacity=0.5)
fig.show()
```



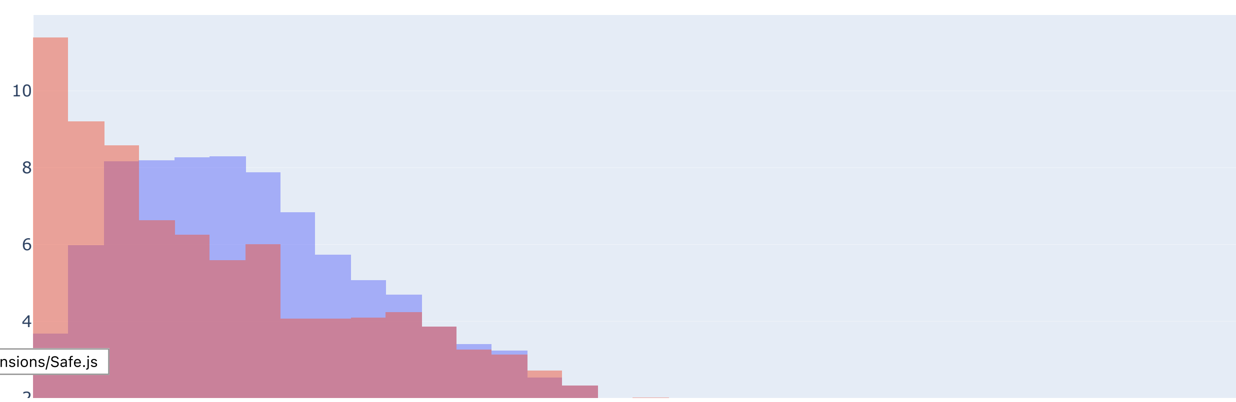
From our histogram above we can see that in the playoffs the minutes are not as evenly distributed compared to when players are playing in the regular season. This makes sense as most players in the playoffs will play less minutes and essentially the players who are playing close to 40 plus minutes a game are only less than around 1.5% of players.

Distribution of Points compared to regular season and playoffs?

In [ ]:
```
fig=go.Figure()
fig.add_trace(go.Histogram(x=rs_df['PTS']/rs_df['GP'],histnorm='percent',name='RS',
                           xbins={'start':0,'end':38,'size':1}))

fig.add_trace(go.Histogram(x=playoff_df['PTS']/playoff_df['GP'],histnorm='percent',name='PLAYOFFS',
                           xbins={'start':0,'end':38,'size':1}))


fig.update_layout(barmode='overlay')
fig.update_traces(opacity=0.5)
fig.show()
```



From our historgram above we can infer that our data seems to be right skewed and that players dont score as much in the playoffs when compared to the regular season. This comes to no surprise as players due seem to play less minutes and also your stars will play more in the playoffs. It is still interesting to examine the distribution because youi can see how much of a small percentage of the players in the NBA actually score more than 20 points per game in the playoffs which seems to be just around 1%.

Which players have the most Points and Assists?

I want to look at accumulated stats for each player summed up within the last 10 seasons. In order to do this I will create a new dataframe where I will group each players regular season stats and get their totals.

From created 5 new columns FG%, 3PT%, Avg_ORB, Avg_DRB, FT% from our accumalated stats from 10 seasons and select the top 10 players with the most Points.

In [ ]:

```
per_min_data.season_start_year = per_min_data.season_start_year.astype(str)
```

```
/Users/araza/opt/anaconda3/lib/python3.8/site-packages/pandas/core/generic.py:5494: 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-ver
sus-a-copy
```

In [ ]:

```
top10_SCORING=rs_df.groupby('PLAYER')[total_cols].sum().reset_index()
top10_SCORING['FG%']=top10_SCORING['FGM']/top10_SCORING['FGA']
top10_SCORING['3PT%']=top10_SCORING['FG3M']/top10_SCORING['FG3A']
top10_SCORING['Avg_ORB']=top10_SCORING['OREB']/top10_SCORING['REB']
top10_SCORING['Avg_DRB']=top10_SCORING['DREB']/top10_SCORING['REB']
top10_SCORING['FT%']=top10_SCORING['FTM']/top10_SCORING['FTA']
top10_score=top10_SCORING.nlargest(10, ['PTS'])
top10_SCORE = pd.DataFrame(data=top10_score)
top10_SCORE
```

Out[ ]:

	PLAYER	MIN	FGM	FGA	FG3M	FG3A	FTM	FTA	OREB	DREB	...	BLK	TOV	PF	PTS	GP	FG%	3PT%	Avg_ORB	Av
568	James Harden	26745	6080	13760	2273	6312	6249	7243	623	3920	...	453	3190	1899	20682	722	0.441860	0.360108	0.137134	0.8
844	LeBron James	24643	6749	12768	1223	3414	3296	4591	749	4518	...	459	2489	1187	18017	677	0.528587	0.358231	0.142206	0.8
255	Damian Lillard	25834	5725	13097	2143	5752	3917	4388	425	2551	...	221	1995	1368	17510	711	0.437123	0.372566	0.142809	0.
1141	Russell Westbrook	24766	6209	14094	988	3214	3963	5144	1152	4909	...	212	3091	1937	17369	709	0.440542	0.307405	0.190068	0.8
1207	Stephen Curry	22245	5697	12028	2745	6447	2770	3046	438	2670	...	140	2064	1434	16909	646	0.473645	0.425779	0.140927	0.8
305	DeMar DeRozan	26069	5958	12780	377	1264	4450	5275	527	2951	...	214	1657	1607	16743	735	0.466197	0.298259	0.151524	0.8
782	Kevin Durant	20033	5320	10318	1208	3067	3700	4165	310	3823	...	673	1770	1082	15548	559	0.515604	0.393870	0.075006	0.9
72	Anthony Davis	20805	5375	10434	292	964	3348	4217	1518	4644	...	1413	1179	1419	14390	604	0.515143	0.302905	0.246349	0.7
454	Giannis Antetokounmpo	21354	5188	9702	447	1550	3498	4873	1119	5030	...	856	1884	1969	14321	656	0.534735	0.288387	0.181981	0.8
119	Bradley Beal	22418	5180	11357	1434	3851	2437	2968	560	2081	...	251	1563	1435	14231	645	0.456106	0.372371	0.212041	0.7

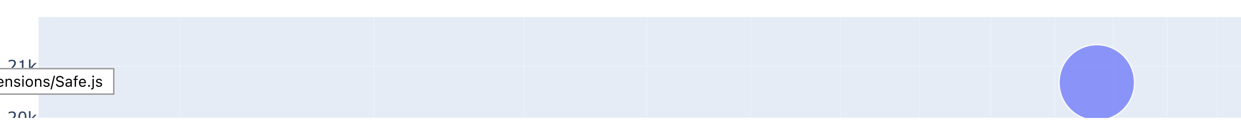
10 rows × 23 columns

In [ ]:

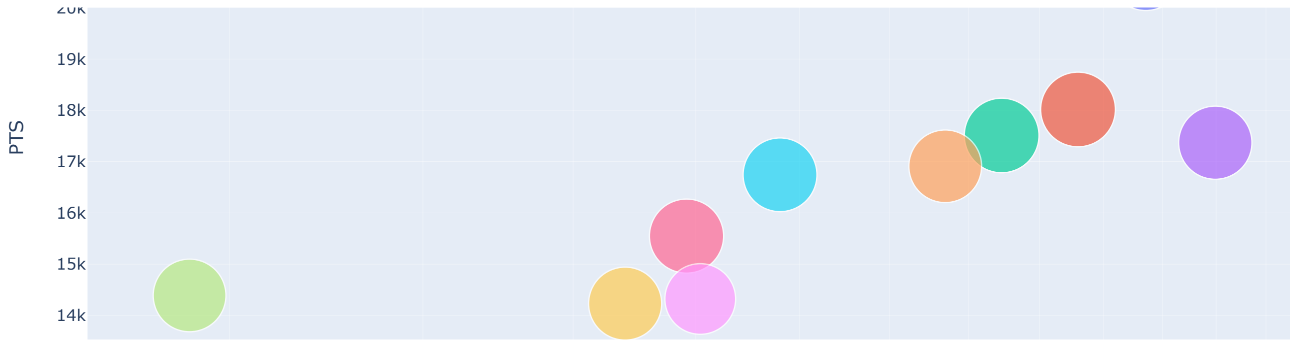
```
fig = px.scatter( top10_SCORE, x="AST", y="PTS",
                 size=top10_SCORE['MIN']/top10_SCORE['GP'], color="PLAYER",
                 log_x=True, size_max=40,title='Top 10 Players With The Most Points and Assists from past 10 seasons')

fig.show()
```

Top 10 Players With The Most Points and Assists from past 10 seasons



Loading [MathJax]/extensions/Safe.js



Who were the most effective shooters?

In order to see who was an actual effective shooter I added TSA which is true shooting percentage and true shooting attempts.

In [ ]:
```
top10_SCORING
#lets add true shooting attempt when looking at our best shooters (FGA + 0.44 * FTA.)
top10_SCORING['TSA']=top10_SCORING['FGA']+0.44*top10_SCORING['FTA']
(top10_SCORING['FGA']>=20).mean()
top10_SCORING = top10_SCORING[top10_SCORING['FGA']>=20]
#True Shooting Percentage; the formula is PTS / (2 * TSA)
top10_SCORING['TS%']=(top10_SCORING['PTS'])/(2*(top10_SCORING['TSA']))
```

In [ ]:
```
top10_SCORING = top10_SCORING[top10_SCORING['FG3A']>=100]
top10_shooter=top10_SCORING.nlargest(10, ['TSA'])
top10_Shooting = pd.DataFrame(data=top10_shooter)
top10_Shooting
```

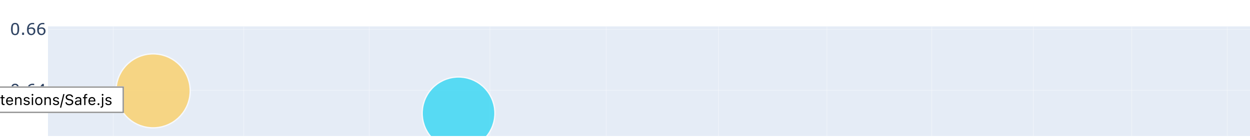
Out[ ]:

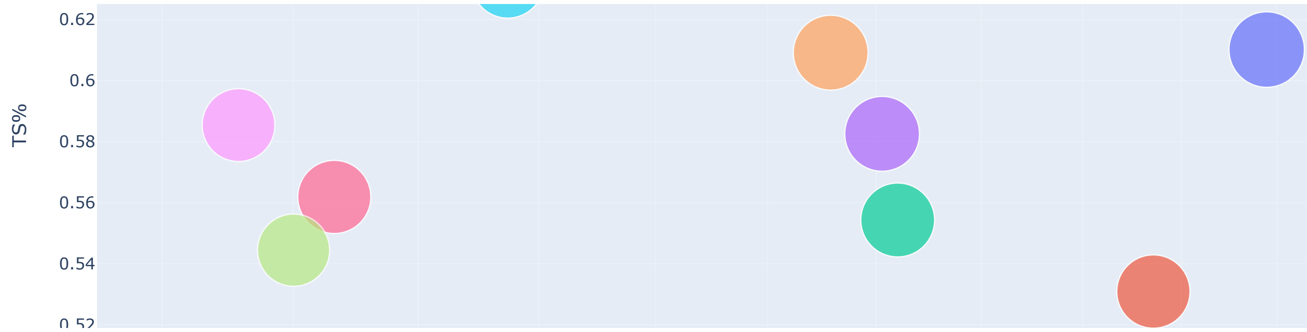
	PLAYER	MIN	FGM	FGA	FG3M	FG3A	FTM	FTA	OREB	DREB	...	PF	PTS	GP	FG%	3PT%	Avg_ORB	Avg_DRB	FT%
568	James Harden	26745	6080	13760	2273	6312	6249	7243	623	3920	...	1899	20682	722	0.441860	0.360108	0.137134	0.862866	0.862764
1141	Russell Westbrook	24766	6209	14094	988	3214	3963	5144	1152	4909	...	1937	17369	709	0.440542	0.307405	0.190068	0.809932	0.770412
305	DeMar DeRozan	26069	5958	12780	377	1264	4450	5275	527	2951	...	1607	16743	735	0.466197	0.298259	0.151524	0.848476	0.843602
255	Damian Lillard	25834	5725	13097	2143	5752	3917	4388	425	2551	...	1368	17510	711	0.437123	0.372566	0.142809	0.857191	0.892662
844	LeBron James	24643	6749	12768	1223	3414	3296	4591	749	4518	...	1187	18017	677	0.528587	0.358231	0.142206	0.857794	0.717926
1207	Stephen Curry	22245	5697	12028	2745	6447	2770	3046	438	2670	...	1434	16909	646	0.473645	0.425779	0.140927	0.859073	0.909389
119	Bradley Beal	22418	5180	11357	1434	3851	2437	2968	560	2081	...	1435	14231	645	0.456106	0.372371	0.212041	0.787959	0.821092
770	Kemba Walker	22874	4694	11130	1594	4388	2633	3120	385	2196	...	1021	13615	675	0.421743	0.363263	0.149167	0.850833	0.843910
72	Anthony Davis	20805	5375	10434	292	964	3348	4217	1518	4644	...	1419	14390	604	0.515143	0.302905	0.246349	0.753651	0.793929
782	Kevin Durant	20033	5320	10318	1208	3067	3700	4165	310	3823	...	1082	15548	559	0.515604	0.393870	0.075006	0.924994	0.888355

10 rows × 25 columns

In [ ]:
```
fig = px.scatter( top10_Shooting, x="TSA", y="TS%",
                 size=top10_Shooting['MIN']/top10_Shooting['GP'], color="PLAYER",
                 log_x=True, size_max=40,title='Top 10 players TS% and TSA from past 10 seasons')
fig.show()
```

Top 10 players TS% and TSA from past 10 seasons



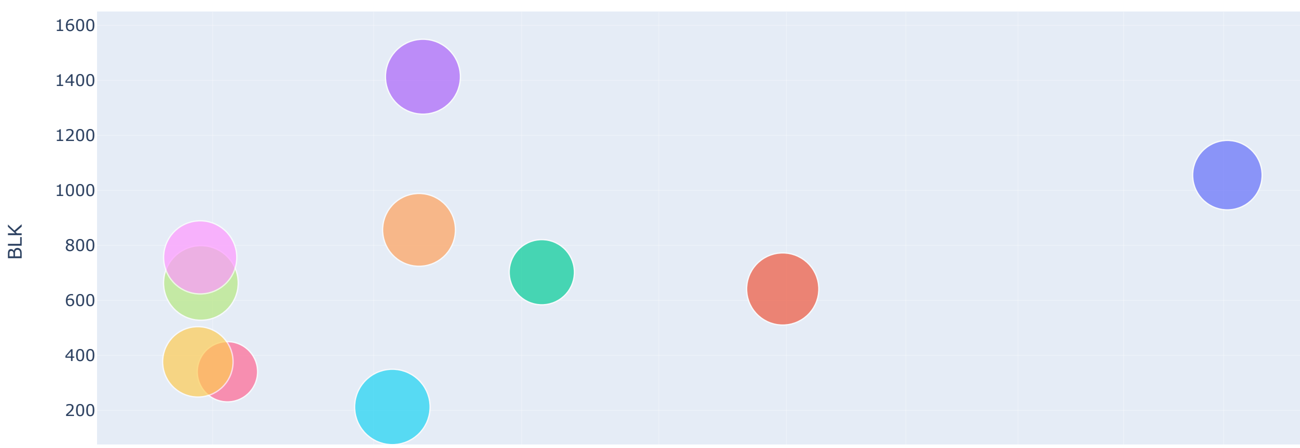


Who were the best players defensively, in terms of rebounds and Blocks over the past 10 seasons?

In [ ]:
```
top10_d = top10_SCORING.nlargest(10, ['REB'])
top10_defense = pd.DataFrame(data=top10_d)
top10_defense

fig = px.scatter( top10_defense, x="REB", y="BLK",
                 size=top10_defense['MIN']/top10_defense['GP'], color="PLAYER",
                 log_x=True, size_max=40,title='Top 10 players with most Blocks and Rebounds from the past 10 seasons')
fig.show()
```

Top 10 players with most Blocks and Rebounds from the past 10 seasons



How has the game changing over the past 10 years?

To make this analysis I want to group all the stats on the column 'season_start_year' and then add columns that we had on the dataframe earlier which I will do for our new dataframe change_df below.

In [ ]:
```
change_df=data.groupby(['season_start_year'])[total_cols].sum().reset_index()
change_df['Poss_est']=change_df['FGA']-change_df['OREB']+change_df['TOV']+0.44*change_df['FTA']
change_df = change_df[list(change_df.columns[0:2])+['Poss_est']+list(change_df.columns[2:-1])]

change_df['FG%']=change_df['FGM']/change_df['FGA']
change_df['3PT%']=change_df['FG3M']/change_df['FG3A']
change_df['FT%']=change_df['FTM']/change_df['FTA']
change_df['FG3A%']=change_df['FG3A']/change_df['FGA']
change_df['FTA/FGA']=change_df['FTA']/change_df['FGA']
change_df['FG3M/FGM']=change_df['FG3M']/change_df['FGM']
change_df['AST_to_TOV']=change_df['AST']/change_df['TOV']
change_df['TRU%']=0.5*change_df['PTS']/(change_df['FGA']+0.475*change_df['FTA'])
change_df['Avg_ORB']=change_df['OREB']/change_df['REB']
Loading [MathJax]/extensions/Safe.js RB']=change_df['DREB']/change_df['REB']
```

Out[ ]:

	season_start_year	MIN	Poss_est	FGM	FGA	FG3M	FG3A	FTM	FTA	OREB	...	FG%	3PT%	FT%	FG3A%	FTA/FGA
0	2012	635884	248201.92	97235	215105	18808	52569	44125	58618	29237	...	0.452035	0.357777	0.752755	0.244388	0.272509
1	2013	638373	254032.80	99251	218411	20480	56952	47219	62420	28669	...	0.454423	0.359601	0.756472	0.260756	0.285791
2	2014	634546	253004.12	98251	219265	20724	59276	45098	60248	28566	...	0.448092	0.349619	0.748539	0.270340	0.274773
3	2015	636391	258064.80	100351	222344	22524	63673	46516	61520	27426	...	0.451332	0.353745	0.756112	0.286372	0.276688
4	2016	632482	258443.80	102147	223333	25408	71018	46806	60620	26470	...	0.457375	0.357768	0.772121	0.317992	0.271433
5	2017	633425	260904.52	103729	225523	27530	76245	43721	57008	25397	...	0.459949	0.361073	0.766927	0.338081	0.252781
6	2018	634231	268739.84	107374	233717	29817	84143	46671	60811	27128	...	0.459419	0.354361	0.767476	0.360021	0.260191
7	2019	552262	234384.64	92997	202223	28032	78279	40949	52906	22802	...	0.459874	0.358104	0.773995	0.387092	0.261622
8	2020	562518	235759.48	95849	205754	29549	80653	39624	50917	22918	...	0.465843	0.366372	0.778208	0.391988	0.247465
9	2021	635572	264004.96	106569	231293	32733	92552	44740	57709	27052	...	0.460753	0.353671	0.775269	0.400150	0.249506

10 rows × 29 columns

In [ ]:

```python
per48_df=change_df.copy()

for col in per48_df.columns[2:18]:
    per48_df[col]=(per48_df[col]/per48_df['MIN'])*48*5

per48_df.drop(columns='MIN',inplace=True)
```

In [ ]:

```python
per48_df
```

Out[ ]:

	season_start_year	Poss_est	FGM	FGA	FG3M	FG3A	FTM	FTA	OREB	DREB	...	FG%	3PT%	
0	2012	93.678188	36.699146	81.186506	7.098653	19.840977	16.653981	22.124035	11.034843	30.708242	...	0.452035	0.357777	0.75
1	2013	95.505092	37.313984	82.112871	7.699574	21.411432	17.752255	23.467158	10.778275	31.509603	...	0.454423	0.359601	0.75
2	2014	95.692020	37.160805	82.931103	7.838297	22.419557	17.057109	22.787190	10.804323	32.236339	...	0.448092	0.349619	0.74
3	2015	97.323111	37.845036	83.851846	8.494400	24.012785	17.542423	23.200831	10.343075	33.040442	...	0.451332	0.353745	0.7
4	2016	98.068423	38.760439	84.745368	9.641255	26.948308	17.760885	23.002710	10.044238	33.078443	...	0.457375	0.357768	0.7
5	2017	98.854773	39.302143	85.448980	10.430911	28.888661	16.565560	21.599905	9.622734	33.599432	...	0.459949	0.361073	0.76
6	2018	101.694117	40.631505	88.441088	11.283081	31.840639	17.660821	23.011553	10.265534	34.571631	...	0.459419	0.354361	0.76
7	2019	101.858020	40.414296	87.881332	12.182044	34.018202	17.795467	22.991696	9.909210	34.469726	...	0.459874	0.358104	0.77
8	2020	100.587493	40.894265	87.785564	12.607170	34.410846	16.905699	21.723892	9.778034	34.196666	...	0.465843	0.366372	0.77
9	2021	99.691601	40.241798	87.339153	12.360393	34.948802	16.894388	21.791646	10.215176	33.834845	...	0.460753	0.353671	0.77

10 rows × 28 columns

How has the game changed in the past 10 years?

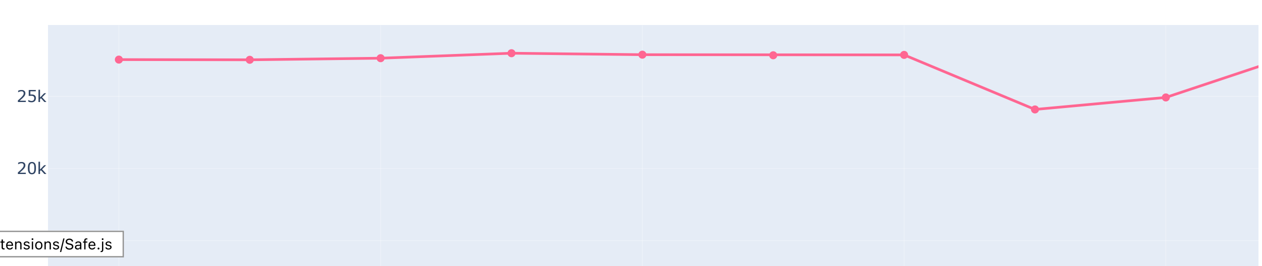
From our visual above when we look at the specific statistics (Double Tap on the Legend) over the past 10 years we can notice changes with the statistics of the game.

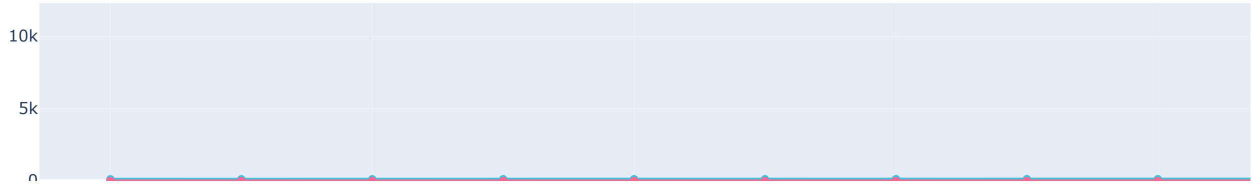
In [ ]:

```python
fig=go.Figure()

for col in per48_df.columns[1:]:
    fig.add_trace(go.Scatter(x=per48_df['season_start_year'],
                             y=per48_df[col],name=col))

fig.show()
```





Some notable changes over the past 10 seasons.

- FG3A attempted have increased which is not surprising as the game has definitely been more 3pt dependent
  going from 20 3 point shots attempted per 48 min to greater than 34 3 point shots per 48 min.
- Points per game has increased from under 98 points per game to around 110 points per game over the past 10 seasons.
- FTA have decreased surprisingly as the general conses is that players go to the free throw line alot more, from our data is seems like this does not seem to be the case.

Playoff Comparison

In [ ]:

```
#rs_df=data[data["Playoffs_Or_Season"]=="Regular Season"]
#playoff_df=data[data["Playoffs_Or_Season"]=="Playoffs"]
```

In [ ]:

```
rs_accum_df=rs_df.groupby(['season_start_year'])[total_cols].sum().reset_index()
playoff_accum_df=playoff_df.groupby(['season_start_year'])[total_cols].sum().reset_index()
```

In [ ]:

```
for i in [rs_accum_df,playoff_accum_df]:
    i['Poss_est']=i['FGA']-i['OREB']+i['TOV']+0.44*i['FTA']
    i['Poss_per48']= ( i['Poss_est']/i['MIN'])*48
    #i['Poss_est']

    i['FG%']=i['FGM']/i['FGA']
    i['3PT%']=i['FG3M']/i['FG3A']
    i['FT%']=i['FTM']/i['FTA']
    i['FG3A%']=i['FG3A']/i['FGA']
    i['FTA/FGA']=i['FTA']/i['FGA']
    i['FG3M/FGM']=i['FG3M']/i['FGM']
    i['AST_to_TOV']=i['AST']/i['TOV']
    i['TRU%']=0.5*i['PTS']/(i['FGA']+0.475*i['FTA'])
    i['Avg_ORB']=i['OREB']/i['REB']
    i['Avg_DRB']=i['DREB']/i['REB']
```

In [ ]:

```
rs_accum_df
```

Out[ ]:

	season_start_year	MIN	FGM	FGA	FG3M	FG3A	FTM	FTA	OREB	DREB	...	FG%	3PT%	FT%	FG3A%	FTA/FGA	FG3
0	2012	594486	91282	201609	17603	49067	41056	54533	27456	76118	...	0.452767	0.358754	0.752865	0.243377	0.270489	0.
1	2013	595202	92779	204172	19054	52974	43870	58029	26846	78315	...	0.454416	0.359686	0.756001	0.259458	0.284216	0.
2	2014	595214	92287	205570	19300	55137	42161	56198	26781	79723	...	0.448932	0.350037	0.750222	0.268215	0.273376	0.
3	2015	594864	94065	208049	20953	59241	43489	57469	25624	82021	...	0.452129	0.353691	0.756738	0.284745	0.276228	0.
4	2016	594409	96061	210114	23748	66421	43883	56855	24936	82109	...	0.457185	0.357538	0.771841	0.316119	0.270591	0.
5	2017	593865	97435	211707	25807	71339	40903	53325	23890	83159	...	0.460235	0.361752	0.767051	0.336970	0.251881	0.2
6	2018	594465	101062	219458	27955	78742	43494	56758	25454	85653	...	0.460507	0.355020	0.766306	0.358802	0.258628	0.
7	2019	512068	86550	188116	25862	72252	37826	48943	21340	73617	...	0.460088	0.357942	0.772858	0.384082	0.260175	0.
8	2020	521512	89020	190983	27427	74822	36650	47135	21232	74454	...	0.466115	0.366563	0.777554	0.391773	0.246802	0.3
9	2021	593758	99930	216722	30598	86535	41657	53781	25422	83925	...	0.461098	0.353591	0.774567	0.399290	0.248157	0.

10 rows × 30 columns

In [ ]:

```
change_tenyear_df=round(100*(playoff_accum_df-rs_accum_df)/rs_accum_df,3)
change_tenyear_df['season_start_year']=list(range(2012,2022))
change_tenyear_df
```

Out[ ]:

	season_start_year	MIN	FGM	FGA	FG3M	FG3A	FTM	FTA	OREB	DREB	...	FG%	3PT%	FT%	FG3A%	FTA/FGA	FG3M
	2	-93.036	-93.478	-93.306	-93.155	-92.863	-92.525	-92.509	-93.513	-93.111	...	-2.578	-4.088	-0.210	6.618	11.902	

Loading [MathJax]/extensions/Safe.js

	season_start_year	MIN	FGM	FGA	FG3M	FG3A	FTM	FTA	OREB	DREB	...	FG%	3PT%	FT%	FG3A%	FTA/FGA	FG3M
1	2013	-92.747	-93.024	-93.026	-92.516	-92.491	-92.366	-92.433	-93.209	-92.981	...	0.024	-0.338	0.886	7.676	8.501	
2	2014	-93.392	-93.538	-93.338	-92.622	-92.493	-93.034	-92.793	-93.335	-93.091	...	-2.995	-1.712	-3.337	12.681	8.176	
3	2015	-93.019	-93.317	-93.129	-92.502	-92.519	-93.040	-92.951	-92.968	-93.185	...	-2.741	0.220	-1.257	8.883	2.591	[illegible]
4	2016	-93.595	-93.664	-93.709	-93.010	-93.079	-93.339	-93.378	-93.848	-93.833	...	0.703	0.998	0.586	10.008	5.257	[illegible]
5	2017	-93.339	-93.540	-93.474	-93.324	-93.123	-93.111	-93.093	-93.692	-93.363	...	-1.016	-2.916	-0.250	5.379	5.834	
6	2018	-93.311	-93.754	-93.503	-93.339	-93.141	-92.696	-92.859	-93.423	-93.337	...	-3.874	-2.893	2.291	5.567	9.904	
7	2019	-92.151	-92.551	-92.501	-91.609	-91.658	-91.744	-91.903	-93.149	-92.256	...	-0.670	0.588	1.964	11.235	7.975	1
8	2020	-92.137	-92.329	-92.266	-92.263	-92.207	-91.885	-91.976	-92.059	-92.348	...	-0.813	-0.722	1.132	0.762	3.744	
9	2021	-92.958	-93.356	-93.277	-93.022	-93.047	-92.599	-92.696	-93.588	-93.236	...	-1.186	0.350	1.331	3.419	8.632	

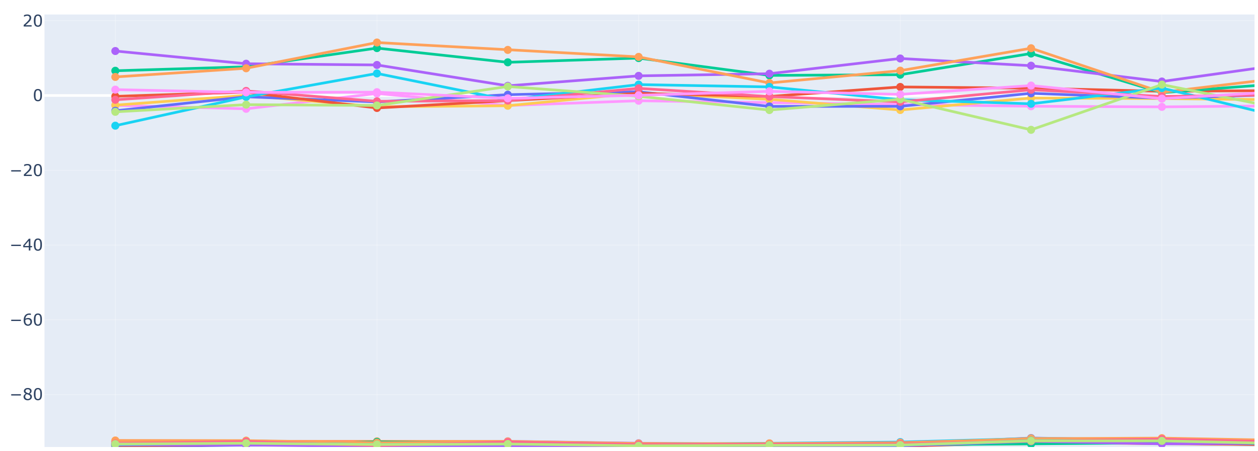
10 rows × 30 columns

In [ ]:

```python
fig=go.Figure()

for col in change_tenyear_df.columns[1:]:
    fig.add_trace(go.Scatter(x=change_tenyear_df['season_start_year'],
                             y=change_tenyear_df[col], name=col))

fig.show()
```



Conclusion

From our analysis we were able to answer most of these questions

What players stats are correlated with each other? Distribution of Minutes compared to regular season and playoffs? Which players have the most Points and Assists? Which players have the most Rebounds and Blocks? Which players have the best Shooting percentages? How has the game changed in the past 10 years?

In [ ]:

Loading [MathJax]/extensions/Safe.js