Brain storming: topic - fantasy football projections

I want to find statistical indicators of good fantasy football performance

what are stats that correlate to good fantasy football production?

This is a broad question. To start we will focus on Wide Recievers and Tight Ends, broadly known as pass catchers.

The type of league can have a huge impact on the fantasy output as well. The main 3 types of scoring format are standard, half points per reception, and full points per reception (PPR).

For standard, a fantasy football score is determined by yards and TDs. For half-PPR and full PPR leagues, a fantasy football score is determined by yards, TDs, and receptions.

This is an example for a WR/TE ex:

- Standard - if a player has 3 receptions, 55 yards, and 1 touchdown they will have 5.5 (1 point per 10 yards) + 6 points (6 points per TD) = 11.5 points
- Half PPR - if a player has 3 receptions, 55 yards, and 1 touchdown they will have 5.5 (1 point per 10 yards) + 6 points (6 points per TD) + 1.5 (half a point per reception) = 13 points
- Full PPR - if a player has 3 receptions, 55 yards, and 1 touchdown they will have 5.5 (1 point per 10 yards) + 6 points (6 points per TD) + 3 (1 point per reception) = 14.5 points

To start lets gather the top scrimmage yard leaders in the NFL

In [55]:
```
#import libraries needed

from sklearn.linear_model import LinearRegression
import requests
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

We will start by pulling data from pro football reference. This website has up-to-date statistics on various aspects about the NFL.

In [56]:
```
#get the url
nfl_url = "https://www.pro-football-reference.com/years/2022/scrimmage.htm"

#ESPN data
#year = 2022
#league_id = '1571831857'
#swid = '{8E04BD2C-6729-4D7F-A097-9BEE26ADA920}'
#espn_s2 = 'AEAVoRBHHwBdTj4OFKC65JBxbepvMyHDOr%2F0MU%2BRYxSwW%2FNSuwiV5DmAS2zOzZ7zytB6jDU05zVM5Rj8cF0TW1JuCBMniCFZ8L%2F6Mzz8osxjuV63nAW1njsOkqVTN7h0RoYazk9irPVJjD6Kz5cVKecW8U4U6Zoerx4BqaHjA57j1h4nUefp13F9a8vjT4pY9px4kHHarNt7ZUX1MbgzMoAmmM7azCsFzfqnmyHvCVVFzxiaB3dMLru5zoeyWZx8eVIo4IzbGNG%2FGcs5E%2FYfyuzNnF6fUMmwHzUiVEkF%2BP%2BVAL6lpspQ8jUCdIDk6%2F%2FgynE%3D'
#fantasy_url = f"https://fantasy.espn.com/apis/v3/games/ffl/seasons/{year}/segments/0/leagues/{league_id}"
```

In [57]:
```
#request access to data
nfl_page = requests.get(nfl_url)
#fantasy_page = requests.get(fantasy_url, cookies = { "swid" : swid, "espn_s2": espn_s2 })
```

In [58]:
```
#parse  HTML for table we want
nfl_soup = BeautifulSoup(nfl_page.text, 'lxml')
#nfl_soup
nfl_table = nfl_soup.find('table', id='receiving_and_rushing')

#code obsolete
#fantasy_soup = BeautifulSoup(fantasy_page.text, 'lxml')
#fantasy_soup
#fantasy_table = fantasy_soup.find('table')
#fantasy_table
```

In [59]:
```
#nfl reference headers in table
nfl_headers = []
for i in nfl_table.find_all('th'):
  title = i.text
  nfl_headers.append(title)

#renaming headers to differentiate between recieving and rushing
nfl_headers[16] = 'RecYds'
nfl_headers[18] = 'RecTD'
nfl_headers[19] = 'Rec1D'
nfl_headers[20] = 'RecLng'
nfl_headers[22] = 'Rec_Y/G'
nfl_headers[26] = 'RushYds'
nfl_headers[27] = 'RushTD'
nfl_headers[28] = 'Rush1D'
nfl_headers[29] = 'RushLng'
nfl_headers[31] = 'Rush_Y/G'
```

In [60]:
```
#grab just the headers we want for table
nfl_data = pd.DataFrame(columns = nfl_headers[7:38])
```

In [61]:
```
#table for real nfl stats
for j in nfl_table.find_all('tr')[1:]:
  row_data_th = j.find_all('th')
  row_data_tr = j.find_all('td')
  row_data = row_data_th + row_data_tr
  row = [i.text for i in row_data]
  length = len(nfl_data)
  nfl_data.loc[length] = row
```

In [62]:
```
#helper method to turn stats of string type to float
def is_float(string):
    try:
        float(string)
        return True
    except ValueError:
        return False
```

Only skill players (RB, WR, TE) with over 10 yards will qualify for my table

In [63]:
```
#clean the data
for index, row in nfl_data.iterrows():
  count = row['Player'].count('*') + row['Player'].count('+')
  name = row['Player']
  row['Player'] = name.replace('*', '').replace('+', '')

for index, row in nfl_data.iterrows():

  if row['Rk'] == 'Rk':
    nfl_data.drop(index, inplace=True)

  elif row['YScm'].isdigit() and int(row['YScm']) < 10:
    nfl_data.drop(index, inplace=True)

  elif row['Pos'] == 'QB':
    nfl_data.drop(index, inplace=True)

  elif int(row['G']) < 3:
    nfl_data.drop(index, inplace=True)

  if len(row['Fmb']) == 0:
    nfl_data = nfl_data.replace(row['Fmb'], '0')
  if len(row['Rec']) == 0:
    nfl_data = nfl_data.replace(row['Rec'], '0')

  if row['RRTD'].isdigit():
    nfl_data = nfl_data.replace(row['RRTD'], int(row['RRTD']))

nfl_data = nfl_data.reset_index(drop=True)
#nfl_data.to_string()
pd.set_option('display.max_rows', None)

nfl_data[:10]
```

Out[63]:

	Rk	Player	Tm	Age	Pos	G	GS	Tgt	Rec	RecYds	Y/R	RecTD	Rec1D	RecLng	R/G	Rec_Y/G	Ctch%	Y/Tgt	Att	RushYds	RushTD	Rush1D	RushLng	Y/A	Rush_Y/G	A/G	Touch	Y/Tch	YScm	RRTD	Fmb
0	1	Josh Jacobs	LVR	24	RB	17	17	64	53	400	7.5	0	16	43	3.1	23.5	82.8%	6.3	340	1653	12	93	86	4.9	97.2	20.0	393	5.2	2053	12	3
1	2	Derrick Henry	TEN	28	RB	16	16	41	33	398	12.1	0	11	69	2.1	24.9	80.5%	9.7	349	1538	13	65	56	4.4	96.1	21.8	382	5.1	1936	13	6
2	3	Christian McCaffrey	2TM	26	RB	17	16	108	85	741	8.7	5	36	49	5.0	43.6	78.7%	6.9	244	1139	8	59	49	4.7	67.0	14.4	329	5.7	1880	13	1
3	4	Justin Jefferson	MIN	23	WR	17	17	184	128	1809	14.1	8	80	64	7.5	106.4	69.6%	9.8	4	24	1	2	10	6.0	1.4	0.2	132	13.9	1833	9	0
4	5	Nick Chubb	CLE	27	RB	17	17	37	27	239	8.9	1	12	26	1.6	14.1	73.0%	6.5	302	1525	12	69	41	5.0	89.7	17.8	329	5.4	1764	13	1
5	6	Tyreek Hill	MIA	28	WR	17	17	170	119	1710	14.4	7	77	64	7.0	100.6	70.0%	10.1	7	32	1	3	10	4.6	1.9	0.4	126	13.8	1742	8	1
6	7	Saquon Barkley	NYG	25	RB	16	16	76	57	338	5.9	0	10	41	3.6	21.1	75.0%	4.4	295	1312	10	62	68	4.4	82.0	18.4	352	4.7	1650	10	1
7	8	Austin Ekeler	LAC	27	RB	17	17	127	107	722	6.7	5	36	23	6.3	42.5	84.3%	5.7	204	915	13	46	72	4.5	53.8	12.0	311	5.3	1637	18	5
8	9	Aaron Jones	GNB	28	RB	17	17	72	59	395	6.7	5	19	30	3.5	23.2	81.9%	5.5	213	1121	2	54	36	5.3	65.9	12.5	272	5.6	1516	7	5
9	10	Davante Adams	LVR	30	WR	17	17	180	100	1516	15.2	14	65	60	5.9	89.2	55.6%	8.4	3	-1	0	0	4	-0.3	-0.1	0.2	103	14.7	1515	14	1

Since all 3 types of scoring are widely used. I will use the middle ground for our data (half PPR).

Calculating the data myself will have a few flaws. First off, any obsecure way a player gained fantasy points will not be included (for example, scoring on a 2pt conversion, or scoring on a punt/kickoff return). In addition, the 18 week of the regular season is usually a very inaccurate week to look at for stats as many teams are not playing their starters 100% as they tank for the draft or get ready for the playoffs.

It is all very subjective, but I think calculating myself to remove outliers is the best way to have fair data.

In [64]:
```
#calculate fantasy points per game and total fantasy points. Add both columns to table.
fppg = []
fp_total = []
for index, row in nfl_data.iterrows():
  fantasy_points = (int(row['RRTD']) * 6) + (int(row['YScm']) * .1) + (int(row['Rec']) * .5) + (int(row['Fmb']) * -2)
  avg = fantasy_points / int(row['G'])
  fp_total.append(fantasy_points)
  fppg.append(avg)

nfl_data['FPPG'] = fppg
nfl_data["FPT"] = fp_total

for index, row in nfl_data.iterrows():
  nfl_data = nfl_data.replace(row['FPPG'], np.round(row['FPPG'], 2))
  nfl_data = nfl_data.replace(row['FPT'], np.round(row['FPT'], 2))
  if row['FPPG'] < 4:
    nfl_data.drop(index, inplace=True)

#sort my fantasy table data by fantasy point per game
nfl_data = nfl_data.sort_values(by=['FPT'], ascending=False)
nfl_data = nfl_data.reset_index(drop=True)
nfl_data[:10]
```

Out[64]:

	Rk	Player	Tm	Age	Pos	G	GS	Tgt	Rec	RecYds	Y/R	RecTD	Rec1D	RecLng	R/G	Rec_Y/G	Ctch%	Y/Tgt	Att	RushYds	RushTD	Rush1D	RushLng	Y/A	Rush_Y/G	A/G	Touch	Y/Tch	YScm	RRTD	Fmb	FPPG	FPT
0	8.0	Austin Ekeler	LAC	27	RB	17	17	127	107	722	6.7	5	36	23	6.3	42.5	84.3%	5.7	204	915	13.0	46	72	4.5	53.8	12.0	311	5.3	1637	18.0	5	18.54	315.2
1	3.0	Christian McCaffrey	2TM	26	RB	17	16	108	85	741	8.7	5	36	49	5.0	43.6	78.7%	6.9	244	1139	8.0	59	49	4.7	67.0	14.4	329	5.7	1880	13.0	1	18.03	306.5
2	4.0	Justin Jefferson	MIN	23	WR	17	17	184	128	1809	14.1	8	80	64	7.5	106.4	69.6%	9.8	4.0	24	1.0	2.0	10.0	6.0	1.4	0.2	132	13.9	1833	9.0	0	17.72	301.3
3	1.0	Josh Jacobs	LVR	24	RB	17	17	64	53	400	7.5	0	16	43	3.1	23.5	82.8%	6.3	340	1653	12.0	93	86	4.9	97.2	20.0	393	5.2	2053	12.0	3	17.52	297.8
4	10.0	Davante Adams	LVR	30	WR	17	17	180	100	1516	15.2	14	65	60	5.9	89.2	55.6%	8.4	3.0	-1	0.0	0	4.0	-0.3	-0.1	0.2	103	14.7	1515	14.0	1	16.68	283.5
5	6.0	Tyreek Hill	MIA	28	WR	17	17	170	119	1710	14.4	7	77	64	7.0	100.6	70.0%	10.1	7.0	32	1.0	3.0	10.0	4.6	1.9	0.4	126	13.8	1742	8.0	1	16.45	279.7
6	2.0	Derrick Henry	TEN	28	RB	16	16	41	33	398	12.1	0	11	69	2.1	24.9	80.5%	9.7	349	1538	13.0	65	56	4.4	96.1	21.8	382	5.1	1936	13.0	6	17.26	276.1
7	5.0	Nick Chubb	CLE	27	RB	17	17	37	27	239	8.9	1	12	26	1.6	14.1	73.0%	6.5	302	1525	12.0	69	41	5.0	89.7	17.8	329	5.4	1764	13.0	1	15.64	265.9
8	15	Stefon Diggs	BUF	29	WR	16	16	154	108	1429	13.2	11	74	53	6.8	89.3	70.1%	9.3	1.0	-3	0.0	0	-3	-3.0	-0.2	0.1	109	13.1	1426	11.0	1	16.29	260.6
9	21	Travis Kelce	KAN	33	TE	17	17	152	110	1338	12.2	12	78	52	6.5	78.7	72.4%	8.8	2.0	5.0	0.0	1.0	4.0	2.5	0.3	0.1	112	12.0	1343	12.0	1	15.25	259.3

In [65]:
```
#create tables for Recievers and Runningbacks
rec_data = nfl_data.copy(deep=True)
run_data = nfl_data.copy(deep=True)

for index, row in rec_data.iterrows():
  if row['Pos'] == 'WR':
    if is_float(str(row['Y/R'])):
      rec_data = rec_data.replace(row['Y/R'], float(row['Y/R']))
    if int(row['Rec']) < 15:
      rec_data.drop(index, inplace=True)
  elif row['Pos'] == 'TE':
    if is_float(str(row['Y/R'])):
      rec_data = rec_data.replace(row['Y/R'], float(row['Y/R']))
    if int(row['Rec']) < 15:
      rec_data.drop(index, inplace=True)
  else:
    rec_data.drop(index, inplace=True)

  #[Rk, Player, Tm, Age, Pos, G, GS, Tgt, Rec, RecYds, Y/R, RecTD, Rec1D, RecLng, R/G, Rec_Y/G, Ctch%, Y/Tgt, Att, RushYds, RushTD, Rush1D, RushLng, Y/A, Rush_Y/G, A/G, Touch, Y/Tch, YScm, RRTD, Fmb]

rec_data = rec_data.drop(['Att', 'RushYds', 'RushTD', 'Rush1D', 'RushLng', 'Y/A', 'Rush_Y/G', 'A/G', 'Touch', 'Y/Tch', 'YScm', 'RRTD'], axis=1)

for index, row in run_data.iterrows():
  if row['Pos'] != 'RB':
    run_data.drop(index, inplace=True)

pd.set_option('display.max_columns', None)
rec_data[:10]
```

Out[65]:

	Rk	Player	Tm	Age	Pos	G	GS	Tgt	Rec	RecYds	Y/R	RecTD	Rec1D	RecLng	R/G	Rec_Y/G	Ctch%	Y/Tgt	Fmb	FPPG	FPT
2	4.0	Justin Jefferson	MIN	23	WR	17	17	184	128	1809	14.1	8	80	64	7.5	106.4	69.6%	9.8	0	17.72	301.3
4	10.0	Davante Adams	LVR	30	WR	17	17	180	100	1516	15.2	14	65	60	5.9	89.2	55.6%	8.4	1	16.68	283.5
5	6.0	Tyreek Hill	MIA	28	WR	17	17	170	119	1710	14.4	7	77	64	7.0	100.6	70.0%	10.1	1	16.45	279.7
8	15	Stefon Diggs	BUF	29	WR	16	16	154	108	1429	13.2	11	74	53	6.8	89.3	70.1%	9.3	1	16.29	260.6
9	21	Travis Kelce	KAN	33	TE	17	17	152	110	1338	12.2	12	78	52	6.5	78.7	72.4%	8.8	1	15.25	259.3
10	11	A.J. Brown	PHI	25	WR	17	16	145	88	1496	17.0	11	59	78	5.2	88.0	60.7%	10.3	2	15.04	255.6
12	16	CeeDee Lamb	DAL	23	WR	17	17	156	107	1359	12.7	9	67	39	6.3	79.9	68.6%	8.7	0	14.59	248.1
14	18.0	Jaylen Waddle	MIA	24	WR	17	17	117	75	1356	18.1	8	60	84	4.4	79.8	64.1%	11.6	1	13.04	221.7
17	23	Amon-Ra St. Brown	DET	23	WR	16	16	146	106	1161	11.0	6	68	49	6.6	72.6	72.6%	8.0	0	13.41	214.6
20	30	Amari Cooper	CLE	28	WR	17	17	132	78	1160	14.9	9	61	55	4.6	68.2	59.1%	8.8	0	12.29	209.0

Second, we will pull data from Pro Fantasy Football. This website focuses on the fantasy aspect of football. This allows us to get the total fantasy points of an individual player based on Half PPR scoring . This is the official number of points a player scored for half PPR.

In [66]:
```
#get pff football data to compare fantasy score outputs (response 200)
pff_url = "https://www.fantasypros.com/nfl/reports/leaders/half-ppr.php?year=2022"
pff_page = requests.get(pff_url)
```

In [67]:
```
#parse HTML for table we want
pff_soup = BeautifulSoup(pff_page.text, 'lxml')
pff_soup
pff_table = pff_soup.find('table', id='data')
```

In [68]:
```
#pff headers
pff_headers = []
for k in pff_table.find_all('th'):
  title = k.text
  pff_headers.append(title)
```

In [69]:
```
#grab headers we want for dataframe
pff_data = pd.DataFrame(columns = pff_headers)
```

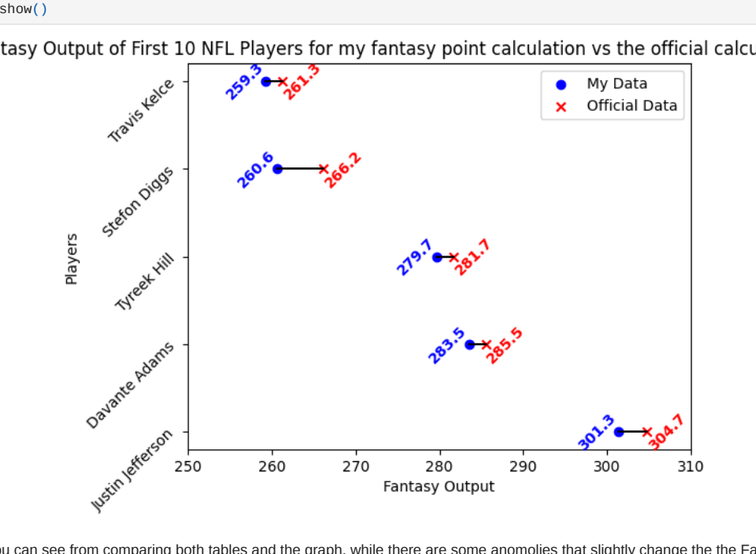
In [70]:
```
#create dataframe
for j in pff_table.find_all('tr')[1:]:
  row_data = j.find_all('td')
  row = [k.text for k in row_data]
  length = len(pff_data)
  pff_data.loc[length] = row
```

In [71]:
```
#clean data
for index, row in pff_data.iterrows():
  if row['Pos'] == 'QB' or row['Pos'] == 'DST' or row['Pos'] == 'K' or row['Pos'] == 'RB':
    pff_data.drop(index, inplace=True)

  elif float(row['TTL']) < 10:
    pff_data.drop(index, inplace=True)

  if is_float(str(row['TTL'])):
      pff_data = pff_data.replace(row['TTL'], np.round(float(row['TTL']), 2))

pff_data[:5]
```

Out[71]:

	#	Player	Pos	Team	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	AVG	TTL
9	10.0	Justin Jefferson	WR	MIN	34.9	7.8	2.9	26.0	24.3	13.7	BYE	12.8	22.0	30.3	4.8	24.8	15.1	27.8	24.3	25.3	2.0	5.8	17.9	304.7
16	17	Davante Adams	WR	LV	25.1	8.2	11.7	15.0	25.9	BYE	13.5	0.7	31.6	23.1	29.6	10.9	33.7	8.6	4.8	2.5	30.8	9.8	16.8	285.5
17	18.0	Tyreek Hill	WR	MIA	14.0	36.5	4.3	20.9	9.5	23.7	10.7	25.5	23.8	12.9	BYE	12	25.1	16.1	17.4	12.3	13.7	3.3	16.6	281.7
20	21	Stefon Diggs	WR	BUF	22.2	38.8	10.9	8.2	20.2	25.8	BYE	19.8	11.8	18.8	12.8	17.7	18.7	5.2	8.2	3.6	3.6	19.9	15.7	266.2
21	22.0	Travis Kelce	TE	KC	22.1	7.6	15.9	19.7	30	15.2	12.8	BYE	15.6	17.1	32.5	13.7	5.6	9.1	15.5	14.3	7.8	6.8	15.4	261.3

In [72]:
```
players = list(rec_data[:5]['Player'])
my_data = list(rec_data[:5]['FPT'])
official_data = list(pff_data[:5]['TTL'])

plt.scatter(my_data, players, marker='o', color='blue', label='My Data', zorder=2)
plt.scatter(official_data, players, marker='x', color='red', label='Official Data', zorder=2)
for i in range(5):
    plt.plot([my_data[i], official_data[i]], [players[i], players[i]], 'k-')
for x, y, my, official in zip(my_data, players, my_data, official_data):
    plt.text(x - 5, y, str(my), ha='left', va='center', color='blue', fontweight='bold', rotation = 45)
    plt.text(official + 5, y, str(official), ha='right', va='center', color='red', fontweight='bold', rotation = 45)

plt.ylabel('Players')
plt.xlabel('Fantasy Output')
plt.title('Fantasy Output of First 10 NFL Players for my fantasy point calculation vs the official calculations (half PPR)')
plt.xlim(250, 310)
plt.legend()
plt.yticks(rotation=45)
plt.tight_layout()
plt.show()
```



As you can see from comparing both tables and the graph, while there are some anomolies that slightly change the the Fantasy points per game / fantasy point total, it is relatively similar.

This is a relatively small example of just 5 players. If we look at other cases, some players came out with the exact same score, while others had a lower official score compared to my calculations by a small margin.

Now that we understand why we are using an objective calculation of fantasy output rather than the official numbers. Lets take a look at statistic that we expected to be heavily correlated with fantasy point output, total touchdowns (This will include all skill players).

In [73]:
```
#sort data by fantasy points per game. I can allows players who got injured for some part of the year to be included where they would be projecting a full season
nfl_data = nfl_data.sort_values(by=['FPPG'], ascending=False)
nfl_data = nfl_data.reset_index(drop=True)
nfl_data[:10]
```

Out[73]:

	Rk	Player	Tm	Age	Pos	G	GS	Tgt	Rec	RecYds	Y/R	RecTD	Rec1D	RecLng	R/G	Rec_Y/G	Ctch%	Y/Tgt	Att	RushYds	RushTD	Rush1D	RushLng	Y/A	Rush_Y/G	A/G	Touch	Y/Tch	YScm	RRTD	Fmb	FPPG	FPT
0	8.0	Austin Ekeler	LAC	27	RB	17	17	127	107	722	6.7	5	36	23	6.3	42.5	84.3%	5.7	204	915	13.0	46	72	4.5	53.8	12.0	311	5.3	1637	18.0	5	18.54	315.2
1	3.0	Christian McCaffrey	2TM	26	RB	17	16	108	85	741	8.7	5	36	49	5.0	43.6	78.7%	6.9	244	1139	8.0	59	49	4.7	67.0	14.4	329	5.7	1880	13.0	1	18.03	306.5
2	66	Cooper Kupp	LAR	29	WR	9.0	9.0	98	75	812	10.8	6	42	75	8.3	90.2	76.5%	8.3	9.0	52	1.0	4.0	20	5.8	5.8	1.0	84	10.3	864	7.0	2	17.99	161.9
3	4.0	Justin Jefferson	MIN	23	WR	17	17	184	128	1809	14.1	8	80	64	7.5	106.4	69.6%	9.8	4.0	24	1.0	2.0	10.0	6.0	1.4	0.2	132	13.9	1833	9.0	0	17.72	301.3
4	1.0	Josh Jacobs	LVR	24	RB	17	17	64	53	400	7.5	0	16	43	3.1	23.5	82.8%	6.3	340	1653	12.0	93	86	4.9	97.2	20.0	393	5.2	2053	12.0	3	17.52	297.8
5	2.0	Derrick Henry	TEN	28	RB	16	16	41	33	398	12.1	0	11	69	2.1	24.9	80.5%	9.7	349	1538	13.0	65	56	4.4	96.1	21.8	382	5.1	1936	13.0	6	17.26	276.1
6	10.0	Davante Adams	LVR	30	WR	17	17	180	100	1516	15.2	14	65	60	5.9	89.2	55.6%	8.4	3.0	-1	0.0	0	4.0	-0.3	-0.1	0.2	103	14.7	1515	14.0	1	16.68	283.5
7	41	Ja'Marr Chase	CIN	22	WR	12	12	134	87	1046	12.0	9	58	60	7.3	87.2	64.9%	7.8	5.0	8.0	0.0	0	6.0	1.6	0.7	0.4	92	11.5	1054	9.0	2	16.58	198.9
8	6.0	Tyreek Hill	MIA	28	WR	17	17	170	119	1710	14.4	7	77	64	7.0	100.6	70.0%	10.1	7.0	32	1.0	3.0	10.0	4.6	1.9	0.4	126	13.8	1742	8.0	1	16.45	279.7
9	15	Stefon Diggs	BUF	29	WR	16	16	154	108	1429	13.2	11	74	53	6.8	89.3	70.1%	9.3	1.0	-3	0.0	0	-3	-3.0	-0.2	0.1	109	13.1	1426	11.0	1	16.29	260.6

In [74]:
```
#sort data by fantasy points per game. I can allows players who got injured for some part of the year to be included where they would be projecting a full season
rec_data = rec_data.sort_values(by=['FPPG'], ascending=False)
rec_data = rec_data.reset_index(drop=True)
rec_data[:10]
```

Out[74]:

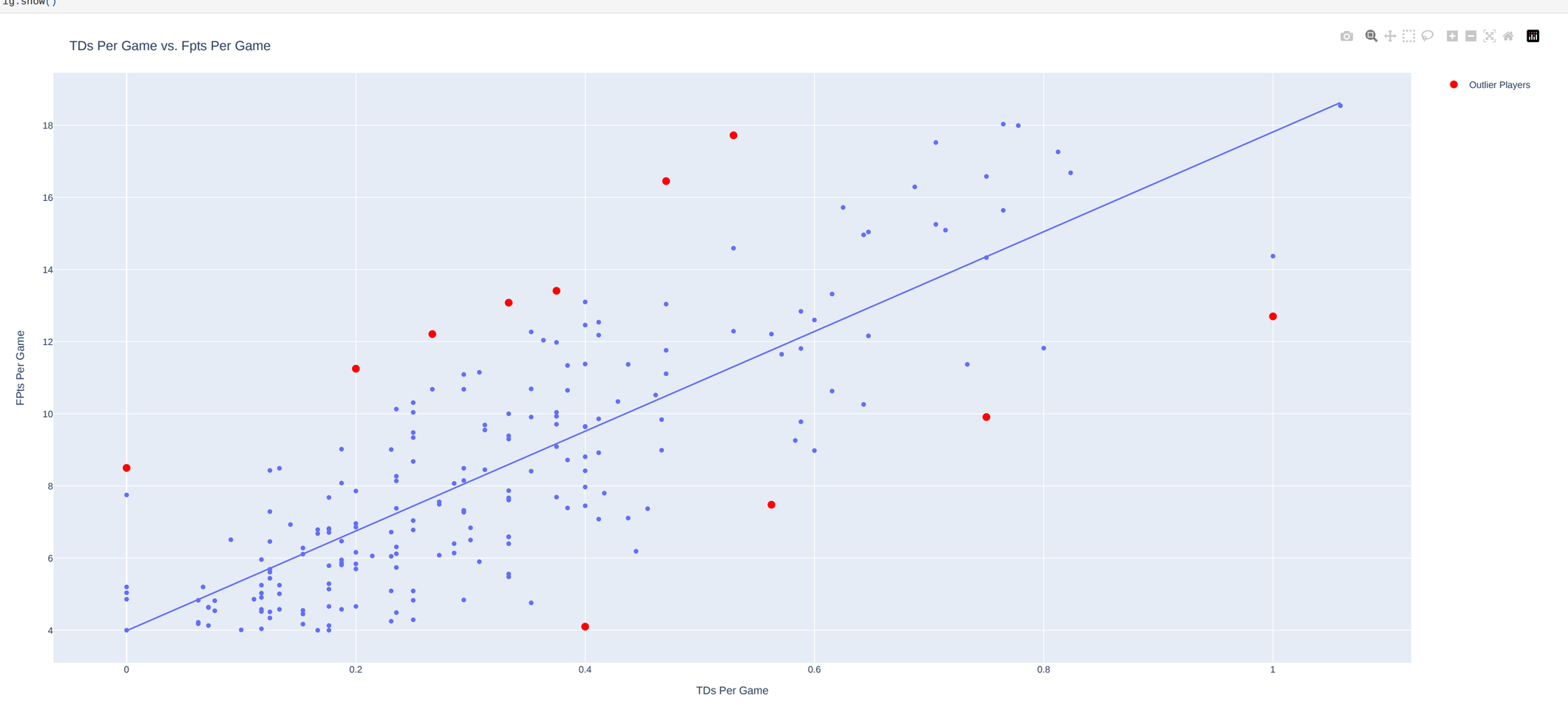
	Rk	Player	Tm	Age	Pos	G	GS	Tgt	Rec	RecYds	Y/R	RecTD	Rec1D	RecLng	R/G	Rec_Y/G	Ctch%	Y/Tgt	Fmb	FPPG	FPT
0	66	Cooper Kupp	LAR	29	WR	9.0	9.0	98	75	812	10.8	6	42	75	8.3	90.2	76.5%	8.3	2	17.99	161.9
1	4.0	Justin Jefferson	MIN	23	WR	17	17	184	128	1809	14.1	8	80	64	7.5	106.4	69.6%	9.8	0	17.72	301.3
2	10.0	Davante Adams	LVR	30	WR	17	17	180	100	1516	15.2	14	65	60	5.9	89.2	55.6%	8.4	1	16.68	283.5
3	41	Ja'Marr Chase	CIN	22	WR	12	12	134	87	1046	12.0	9	58	60	7.3	87.2	64.9%	7.8	2	16.58	198.9
4	6.0	Tyreek Hill	MIA	28	WR	17	17	170	119	1710	14.4	7	77	64	7.0	100.6	70.0%	10.1	1	16.45	279.7
5	15	Stefon Diggs	BUF	29	WR	16	16	154	108	1429	13.2	11	74	53	6.8	89.3	70.1%	9.3	1	16.29	260.6
6	21	Travis Kelce	KAN	33	TE	17	17	152	110	1338	12.2	12	78	52	6.5	78.7	72.4%	8.8	1	15.25	259.3
7	11	A.J. Brown	PHI	25	WR	17	16	145	88	1496	17.0	11	59	78	5.2	88.0	60.7%	10.3	2	15.04	255.6
8	16	CeeDee Lamb	DAL	23	WR	17	17	156	107	1359	12.7	9	67	39	6.3	79.9	68.6%	8.7	0	14.59	248.1
9	287	Michael Thomas	NOR	29	WR	3.0	3.0	22	16	171	10.7	3	13	21	5.3	57.0	72.7%	7.8	0	14.37	43.1

In [75]:
```
#do analysis -> linear regression between total tounchdowns scored and fantasy point output

nfl_data['TDPG'] = nfl_data.apply(lambda x: np.round(int(x['RRTD']) / int(x['G']), 4), axis = 1)
X = nfl_data['TDPG'].values.reshape(-1, 1)
y = nfl_data['FPPG'].values
np.set_printoptions(suppress=True)

model = LinearRegression()
model.fit(X, y)

coef = model.coef_[0]
intercept = model.intercept_

y_pred = model.predict(X)
```

In [76]:
```
''' #graphing but i wanted a different graph
plt.figure(figsize=(15,10))

plt.scatter(X.ravel(), y, color='red', label='Data points')
# Add player names next to data points
for i, player in enumerate(nfl_data['Player']):
    print(i)
      plt.text(X[i], y[i], player, ha='center', va='bottom')


plt.plot(X.ravel(), y_pred, color='blue', label='LinReg line: y = ' + str(np.round(coef, 3)) + 'x + ' + str(np.round(intercept, 3)))

plt.xlabel('TDs')
plt.ylabel('FPts')
plt.title('TDs vs. Fpts')
plt.legend()
plt.show()
'''
```

Out[76]:
```
" #graphing but i wanted a different graph\nplt.figure(figsize=(15,10))\n\nplt.scatter(X.ravel(), y, color='red', label='Data points')\n# Add player names next to data points\nfor i, player in enumerate(nfl_data['Player']):\n    print(i)\n      plt.text(X[i], y[i], player, ha='center', va='bottom')\n\n\nplt.plot(X.ravel(), y_pred, color='blue', label='LinReg line: y = ' + str(np.round(coef, 3)) + 'x + ' + str(np.round(intercept, 3)))\n\nplt.xlabel('TDs')\nplt.ylabel('FPts')\nplt.title('TDs vs. Fpts')\nplt.legend()\nplt.show()\n"
```

In [77]:
```
#graph it
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
#pio.renderers.default = 'colab'
pio.renderers.default = "notebook_connected"
import plotly
plotly.offline.init_notebook_mode(connected=True)

# Create a DataFrame with the player names, TDs, and FPts
df = pd.DataFrame({'Player': nfl_data['Player'], 'TDs': X.ravel(), 'FPts': y, 'GP' : nfl_data['G']})

# Create a scatter plot with hover labels
fig = px.scatter(df, x='TDs', y='FPts', hover_data=['Player', 'GP'], trendline='ols')



# Calculate the linear regression coefficients using NumPy polyfit
slope, intercept = np.polyfit(df['TDs'], df['FPts'], 1)

# y = mx + b
predicted_fpts = slope * df['TDs'] + intercept

# Calculate the residuals (differences between actual FPts and predicted FPts)
residuals = df['FPts'] - predicted_fpts

# Define a threshold to identify outliers (example: 2 standard deviations from the residuals mean)
threshold = 2
outliers = df[abs(residuals) > threshold * residuals.std()]

fig.add_trace(go.Scatter(x=outliers['TDs'], y=outliers['FPts'], mode='markers',
                         text=outliers['Player'] + ', GP: ' + outliers['GP'].astype(str) + '<br>TDPG: ' + outliers['TDs'].astype(str) + ', FPPG: ' + outliers['FPts'].astype(str),
                         marker=dict(color='red', size=10),
                         name='Outlier Players'))


# Update the layout to add axis labels and a title
fig.update_layout(
    xaxis_title='TDs Per Game',
    yaxis_title='FPts Per Game',
    title='TDs Per Game vs. Fpts Per Game',
    height = 900,
    width = 2000
)

# Show the interactive plot
fig.show()
```



It's very interesting to look at this data. In my opinion, TDs are very random in any given season, but heavily impact fantasy output. This is shown in the R^2 value of .65. It doesn't have an extremely strong correlation, although it does have some.

Overall, TDs should hold some weight for fantasy output, but it's really hard to know how much they can be used to predict future output.

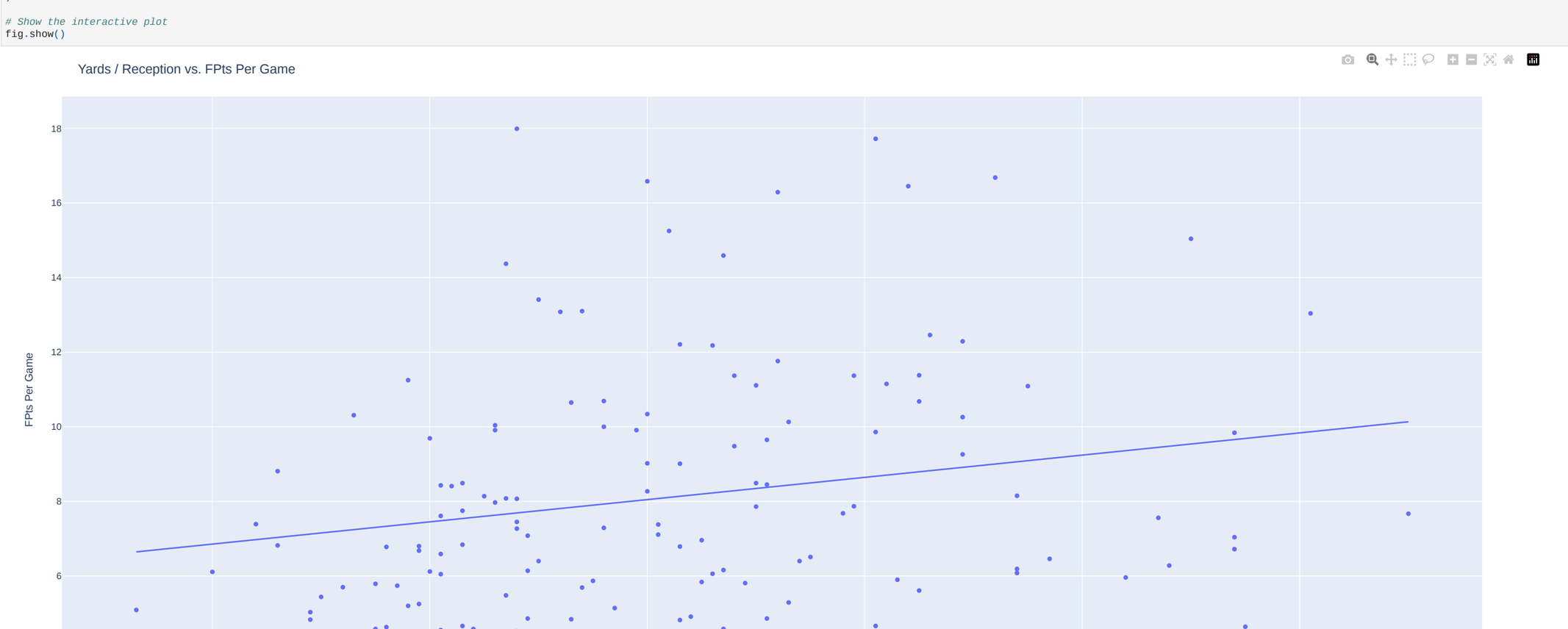
For example, looking at an outlier above the regression line, you can see Diontae Johnson scored 131.7 half PPR fantasy points without scoring a single touchdown. Compare this to someone who was an outlier below the regression line like Christiian Watson. Watson scored 143.6 half PPR fantasy points but has 9 TDs that boosted his output heavily.

Some of the TD scoring can be attributed to random luck, but not entirely. Countless factors go into the likelihood a player scores a TD. A few factors that can have a large impact is 'big play' ability (yards per reception), the skill level of the pass catchers Quarterback, and the amount of redzone targets (pass thrown to them inside the 20 yard line) can show how likely a player is to score a TD.

After going through the first question, it will be easier to look at recievers and running backs separately (at least at first) because the stats differ quite widely between the two. The downside of doing this is the question of is a WR/TE or RB better more valuable for fantasy production will not be answered.

Lets compare the Yards per Reception (Y/R) to fantasy point output for pass catchers.

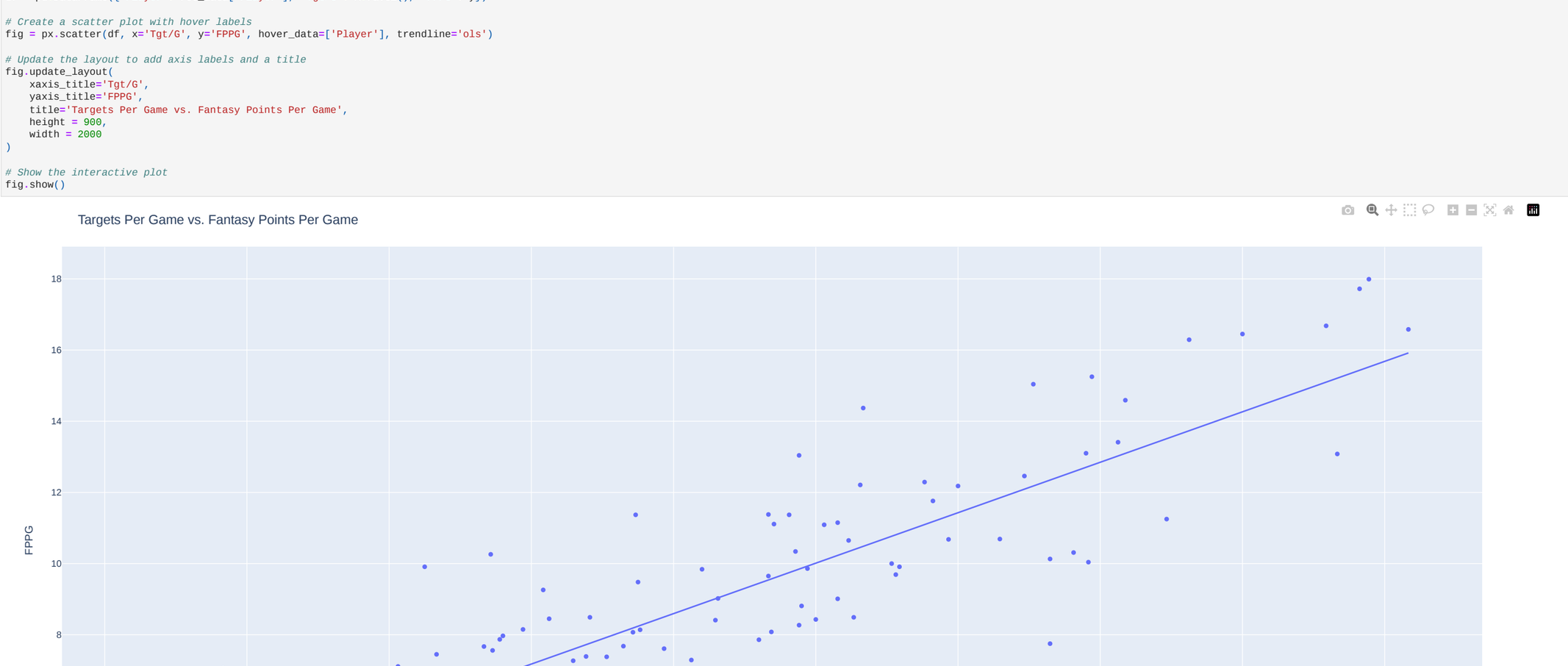
In [78]:
```
X = rec_data['Y/R'].values.reshape(-1, 1)
y = rec_data['FPPG'].values
np.set_printoptions(suppress=True)

model = LinearRegression()
model.fit(X, y)

coef = model.coef_[0]
intercept = model.intercept_

y_pred = model.predict(X)

# Create a DataFrame with the player names, Yds/Rec, and FPts
df = pd.DataFrame({'Player': rec_data['Player'], 'Yds/Rec': X.ravel(), 'FPts': y})

# Create a scatter plot with hover labels
fig = px.scatter(df, x='Yds/Rec', y='FPts', hover_data=['Player'], trendline='ols')

# Update the layout to add axis labels and a title
fig.update_layout(
    xaxis_title='Yds/Rec',
    yaxis_title='FPts Per Game',
    title='Yards / Reception vs. FPts Per Game',
    height = 900,
    width = 2000
)

# Show the interactive plot
fig.show()
```



Interestingly, it seems Yards per Reception has very little correlation to fantasy production with an R^2 value of .04. Many of the top fantasy producers seem to be in the middle of yards per reception. At the same time there are plenty of middle of the pack recievers with lower fantasy production. This makes it seem like volume may be king for fantasy production.

Lets take a look at volume of targets.

In [79]:
```
rec_data['Tgt/G'] = rec_data.apply(lambda x: np.round(int(x['Tgt']) / int(x['G']), 4), axis = 1)

X = rec_data['Tgt/G'].values.reshape(-1, 1)
y = rec_data['FPPG'].values
np.set_printoptions(suppress=True)

model = LinearRegression()
model.fit(X, y)

coef = model.coef_[0]
intercept = model.intercept_

y_pred = model.predict(X)

# Create a DataFrame with the player names, Yds/Rec, and FPts
df = pd.DataFrame({'Player': rec_data['Player'], 'Tgt/G': X.ravel(), 'FPPG': y})

# Create a scatter plot with hover labels
fig = px.scatter(df, x='Tgt/G', y='FPPG', hover_data=['Player'], trendline='ols')

# Update the layout to add axis labels and a title
fig.update_layout(
    xaxis_title='Tgt/G',
    yaxis_title='FPPG',
    title='Targets Per Game vs. Fantasy Points Per Game',
    height = 900,
    width = 2000
)

# Show the interactive plot
fig.show()
```



As seen in the graph above, targets seem to have strong correlation with fantasy point production (r^2 value of .79). That is not a shock.

So does this mean if most recievers had a large amount of volume, they would produce in fantasy?

For the most part, I'd say yes (in the short term), but you have to ask why certain recievers get more volume than others. That is because they have proven to produce and be talented enough to warrant the amounts of targets given. This is why looking at volume alone without any context can be decieving.

Currently fleshing out the rest of the project. I plan to do k-means clustering next revolving around different pass catcher archetypes. This is just to show my progress.

In [80]:
```
!jupyter nbconvert --to html /content/drive/MyDrive/DS_Fantasy/Fantasy_Football_projections.ipynb
```

```
[NbConvertApp] Converting notebook /content/drive/MyDrive/DS_Fantasy/Fantasy_Football_projections.ipynb to html
[NbConvertApp] Writing 660780 bytes to /content/drive/MyDrive/DS_Fantasy/Fantasy_Football_projections.html
```

In [80]: