In [1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [41]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

#Importing the total season fantasy data for each position
totalRB = pd.read_csv("/content/drive/MyDrive/Fantasy Football Analysis 2023 Season/To
totalRB = totalRB.drop(["ATT","Rank","YDS", "Y/A","LG","20+" ,"TD", "REC", "TGT", "YDS
totalRB['Position']='RB'
totalRB['Team'] = totalRB['Player'].str.split('(').str[1]
totalRB['Team']= totalRB['Team'].str.replace(")", "")
totalRB['Player'] = totalRB['Player'].str.split('(').str[0]
totalRB = totalRB.sort_values('Player')


totalQB = pd.read_csv("/content/drive/MyDrive/Fantasy Football Analysis 2023 Season/To
totalQB=totalQB.reset_index()
totalQB = totalQB.drop(["CMP", "ATT", "PCT", "YDS", "Y/A", "TD", "INT", "SACKS", "ATT.
totalQB['Position']='QB'
totalQB['Team'] = totalQB['Player'].str.split('(').str[1]
totalQB['Team']= totalQB['Team'].str.replace(")", "")
totalQB['Player'] = totalQB['Player'].str.split('(').str[0]
totalQB = totalQB.sort_values('Player')
totalQB.rename(columns = {'index':'Rank'}, inplace = True)


totalWR = pd.read_csv("/content/drive/MyDrive/Fantasy Football Analysis 2023 Season/To
totalWR = totalWR.drop(["REC","TGT","YDS", "Y/R","LG","20+" ,"TD", "ATT","YDS","YDS.1'
totalWR['Position']='WR'
totalWR['Team'] = totalWR['Player'].str.split('(').str[1]
totalWR['Team']= totalWR['Team'].str.replace(")", "")
totalWR['Player'] = totalWR['Player'].str.split('(').str[0]
totalWR = totalWR.sort_values('Player')

totalTE = pd.read_csv("/content/drive/MyDrive/Fantasy Football Analysis 2023 Season/To
totalTE = totalTE.drop(["REC","TGT","YDS", "Y/R","LG","20+" ,"TD", "ATT","YDS","YDS.1'
totalTE['Position']='TE'
totalTE['Team'] = totalTE['Player'].str.split('(').str[1]
totalTE['Team']= totalWR['Team'].str.replace(")", "")
totalTE['Player'] = totalTE['Player'].str.split('(').str[0]
totalTE = totalTE.sort_values('Player')

totalswithgamesplayed=pd.concat([totalQB, totalRB, totalWR, totalTE], axis=0)
totalswithgamesplayed = totalswithgamesplayed.drop(columns=["FPTS", "Team"])


Schedule2023 = pd.read_csv("/content/drive/MyDrive/Fantasy Football Analysis 2023 Seas
Schedule2023= Schedule2023.drop(["W17", "W18"], axis = 1)

Defense = pd.read_csv("/content/drive/MyDrive/Fantasy Football Analysis 2023 Season/De
twx = totalswithgamesplayed.pop('Rank')
totalswithgamesplayed.insert(3, 'Rank', twx+1)
totalswithgamesplayed.rename(columns={'Position': 'POS'}, inplace=True)
```

```
totalswithgamesplayed = totalswithgamesplayed.sort_values(by='Rank')
print(totalswithgamesplayed[totalswithgamesplayed['Player']=='Keenan Allen '])
```

```
           Player     G  FPTS/G  Rank POS
10  Keenan Allen   13.0    13.1  12.0  WR
```

In [42]:
```
#establishing some data frames.  Some used and some not used.  Alot changed as we got
qblist=list(totalQB['Player'])
completedataqb=pd.DataFrame()
weeklydata_QB=pd.DataFrame()
wdata=pd.DataFrame()
totalqbdata = pd.DataFrame()
tdata=pd.DataFrame()
completedatarb=pd.DataFrame()
weeklydata_RB=pd.DataFrame()
wdatar=pd.DataFrame()
print(qblist)
rblist=list(totalRB['Player'])
wdatawr=pd.DataFrame()
```

```
['AJ McCarron ', 'Aaron Rodgers ', 'Adam Froman ', "Aidan O'Connell ", 'Alex McGough
', 'Andy Dalton ', 'Anthony Brown Jr. ', 'Anthony Richardson ', 'Bailey Zappe ', 'Bak
er Mayfield ', 'Ben Chappell ', 'Ben DiNucci ', 'Blaine Gabbert ', 'Bo Nix ', 'Brando
n Allen ', 'Brett Rypien ', 'Brett Smith ', 'Brian Hoyer ', 'Brock Purdy ', 'Bryce Yo
ung ', 'C.J. Beathard ', 'C.J. Stroud ', 'Caleb Williams ', 'Carson Wentz ', 'Carter
Bradley ', 'Case Keenum ', 'Casey Bauman ', 'Cephus Johnson III ', 'Chris Oladokun ',
'Clayton Tune ', 'Cooper Rush ', 'Dak Prescott ', 'Daniel Jones ', 'Davis Mills ', 'D
erek Carr ', 'Deshaun Watson ', 'Desmond Ridder ', 'Devin Leary ', 'Dorian Thompson-R
obinson ', 'Drake Maye ', 'Dresser Winn ', 'Drew Lock ', 'Drew Plitt ', 'Easton Stick
', 'Erik Ainge ', 'Feleipe Franks ', 'Gardner Minshew II ', 'Geno Smith ', 'Hendon Ho
oker ', 'Hunter Cantwell ', 'Ian Book ', 'J.J. McCarthy ', 'Jacob Eason ', 'Jacoby Br
issett ', 'Jake Browning ', 'Jake Fromm ', 'Jake Haener ', 'Jalen Hurts ', 'Jameis Wi
nston ', 'Jared Goff ', 'Jaren Hall ', 'Jarrett Stidham ', 'Jayden Daniels ', 'Jeff D
riskel ', 'Jimmy Garoppolo ', 'Joe Burrow ', 'Joe Flacco ', 'Joe Milton III ', 'John
Rhys Plumlee ', 'John Wolford ', 'Jordan Love ', 'Jordan Travis ', 'Josh Allen ', 'Jo
sh Johnson ', 'Joshua Dobbs ', 'Justin Fields ', 'Justin Herbert ', 'Kellen Mond ',
'Kenny Pickett ', 'Kirk Cousins ', 'Kyle Allen ', 'Kyle Trask ', 'Kyler Murray ', 'La
mar Jackson ', 'Logan Woodside ', 'Mac Jones ', 'Malik Cunningham ', 'Malik Willis ',
'Marcus Mariota ', 'Mason Rudolph ', 'Matt Barkley ', 'Matthew Stafford ', 'Max Dugga
n ', 'Michael Penix Jr. ', 'Michael Penix Jr. ', 'Michael Pratt ', 'Mike White ', 'Mi
tchell Trubisky ', 'Nate Sudfeld ', 'Nathan Peterman ', 'Nathan Rourke ', 'Nick Mulle
ns ', 'P.J. Walker ', 'Patrick Mahomes II ', 'Russell Wilson ', 'Ryan Tannehill ', 'S
am Darnold ', 'Sam Ehlinger ', 'Sam Howell ', 'Sean Clifford ', 'Shane Buechele ', 'S
kylar Thompson ', 'Spencer Rattler ', 'Stetson Bennett ', 'Tanner McKee ', 'Taylor He
inicke ', 'Teddy Bridgewater ', 'Tim Boyle ', 'Tommy DeVito ', 'Trevor Lawrence ', 'T
revor Siemian ', 'Trey Lance ', 'Tua Tagovailoa ', 'Tyler Huntley ', 'Tyrod Taylor ',
'Tyson Bagent ', 'Will Grier ', 'Will Levis ', 'Zach Wilson ']
```

The following several coding frames are importing the week by week data for each player by position. I couldn't find a file with every player week by week in 1 file, so I had to do it position by position and will merge it all together later on down the line

In [43]:
```
import pandas as pd

# Initialize an empty DataFrame for the total QB data
totalqbdata = pd.DataFrame()

# Iterate through each week (1 to 16)
for i in range(1, 17):
```

```
    # Read the weekly QB data
    weeklydata_QB = pd.read_csv(f"/content/drive/MyDrive/Fantasy Football Analysis 202

    # Drop the unnecessary columns
    columns_to_drop = ["Rank", "CMP", "ATT", "PCT", "YDS", "Y/A", "TD", "INT", "SACKS'
    weeklydata_QB = weeklydata_QB.drop(columns=columns_to_drop)

    # Rename the FPTS column to indicate the week number
    weeklydata_QB = weeklydata_QB.rename(columns={"FPTS": f"FPTS_{i}"})

    # Extract the team name and clean the player names
    weeklydata_QB['Team'] = weeklydata_QB['Player'].str.split('(').str[1].str.replace(
    weeklydata_QB['Player'] = weeklydata_QB['Player'].str.split('(').str[0]

    # Sort the data by player name
    weeklydata_QB = weeklydata_QB.sort_values('Player')

    # If it's the first week, initialize the totalqbdata DataFrame
    if i == 1:
        totalqbdata = weeklydata_QB
    else:
        # For subsequent weeks, merge the new data with the totalqbdata DataFrame
        totalqbdata = pd.merge(totalqbdata, weeklydata_QB, on=['Player', 'Team'], how=

# After processing all weeks, add the 'Pos' column
totalqbdata['Pos'] = 'QB'



print(totalqbdata[totalqbdata["Player"]== 'Tua Tagovailoa '])
```

```
                Player  FPTS_1 Team  FPTS_2  FPTS_3  FPTS_4  FPTS_5  FPTS_6  \
65654  Tua Tagovailoa     28.1  MIA    13.3    28.4    15.0    18.9    22.5

       FPTS_7  FPTS_8  FPTS_9  FPTS_10  FPTS_11  FPTS_12  FPTS_13  FPTS_14  \
65654    11.3    22.8    12.4      0.0     18.6      9.8     19.2      9.1

       FPTS_15  FPTS_16 Pos
65654     13.0     15.5  QB
```

In [44]:
```
totalqbdata['AvgFantasyPoints']=((totalqbdata['FPTS_1']+totalqbdata['FPTS_2']+totalqbd
print(totalqbdata[totalqbdata["Player"]== 'Tua Tagovailoa '])
```

```
                Player  FPTS_1 Team  FPTS_2  FPTS_3  FPTS_4  FPTS_5  FPTS_6  \
65654  Tua Tagovailoa     28.1  MIA    13.3    28.4    15.0    18.9    22.5

       FPTS_7  FPTS_8  FPTS_9  FPTS_10  FPTS_11  FPTS_12  FPTS_13  FPTS_14  \
65654    11.3    22.8    12.4      0.0     18.6      9.8     19.2      9.1

       FPTS_15  FPTS_16 Pos  AvgFantasyPoints
65654     13.0     15.5  QB         17.193333
```

In [45]:
```
# Initialize an empty DataFrame for the total RB data
totalrbdata = pd.DataFrame()

# Iterate through each week (1 to 17)
for i in range(1, 17):
    # Read the weekly RB data
    weeklydata_RB = pd.read_csv(f"/content/drive/MyDrive/Fantasy Football Analysis 202

    # Drop the unnecessary columns
```

```
    columns_to_drop = ["ATT", "Rank", "YDS", "Y/A", "LG", "20+", "TD", "REC", "TGT", '
    weeklydata_RB = weeklydata_RB.drop(columns=columns_to_drop)

    # Rename the FPTS column to indicate the week number
    weeklydata_RB = weeklydata_RB.rename(columns={"FPTS": f"FPTS_{i}"})

    # Extract the team name and clean the player names
    weeklydata_RB['Team'] = weeklydata_RB['Player'].str.split('(').str[1].str.replace(
    weeklydata_RB['Player'] = weeklydata_RB['Player'].str.split('(').str[0]

    # Sort the data by player name
    weeklydata_RB = weeklydata_RB.sort_values('Player')

    # If it's the first week, initialize the totalrbdata DataFrame
    if i == 1:
        totalrbdata = weeklydata_RB
    else:
        # For subsequent weeks, merge the new data with the totalrbdata DataFrame
        totalrbdata = pd.merge(totalrbdata, weeklydata_RB, on=['Player', 'Team'], how=



totalrbdata['AvgFantasyPoints']=((totalrbdata['FPTS_1']+totalrbdata['FPTS_2']+totalrbd


totalrbdata=totalrbdata.sort_values('AvgFantasyPoints', ascending=False)
totalrbdata=totalrbdata.reset_index()
totalrbdata.rename(columns = {'index':'Rank'}, inplace=True)
twr = totalrbdata.pop('Rank')
totalrbdata.insert(19, 'Rank', twr)
toralrbdata=totalrbdata.sort_values('Player', ascending=True)
totalrbdata['Pos']='RB'
print(totalrbdata[totalrbdata['Player']=='Aaron Jones '])
print(totalswithgamesplayed.head())
```

```
         Player  FPTS_1 Team  FPTS_2  FPTS_3  FPTS_4  FPTS_5  FPTS_6  FPTS_7  \
0  Aaron Jones     24.7  MIN    24.7    24.7    24.7    24.7    24.7    24.7

   FPTS_8  ...  FPTS_10  FPTS_11  FPTS_12  FPTS_13  FPTS_14  FPTS_15  FPTS_16  \
0    24.7  ...     24.7     24.7     24.7     24.7     24.7     24.7     24.7

   AvgFantasyPoints  Rank  Pos
0         26.346667     1   RB

[1 rows x 21 columns]
         Player     G  FPTS/G  Rank POS
0   Josh Allen   17.0    24.2   1.0  QB
0  CeeDee Lamb   17.0    15.8   2.0  WR
1  Jalen Hurts   17.0    21.9   2.0  QB
0  Sam LaPorta   17.0     9.0   2.0  TE
1  Tyreek Hill   16.0    16.1   3.0  WR
```

In [46]:

```
# Initialize an empty DataFrame for the total WR data
totalwrdata = pd.DataFrame()

# Iterate through each week (1 to 17)
for i in range(1, 17):
    # Read the weekly WR data
    weeklydata_WR = pd.read_csv(f"/content/drive/MyDrive/Fantasy Football Analysis 202
```

```
    # Drop the unnecessary columns
    columns_to_drop = ["ATT", "Rank", "YDS", "Y/R", "LG", "20+", "TD", "REC", "TGT", '
    weeklydata_WR = weeklydata_WR.drop(columns=columns_to_drop)

    # Rename the FPTS column to indicate the week number
    weeklydata_WR = weeklydata_WR.rename(columns={"FPTS": f"FPTS_{i}"})

    # Extract the team name and clean the player names
    weeklydata_WR['Team'] = weeklydata_WR['Player'].str.split('(').str[1].str.replace(
    weeklydata_WR['Player'] = weeklydata_WR['Player'].str.split('(').str[0]

    # Sort the data by player name
    weeklydata_WR = weeklydata_WR.sort_values('Player')

    # If it's the first week, initialize the totalwrdata DataFrame
    if i == 1:
        totalwrdata = weeklydata_WR
    else:
        # For subsequent weeks, merge the new data with the totalwrdata DataFrame
        totalwrdata = pd.merge(totalwrdata, weeklydata_WR, on=['Player', 'Team'], how=




totalwrdata['AvgFantasyPoints']=((totalwrdata['FPTS_1']+totalwrdata['FPTS_2']+totalwrd
totalwrdata['Pos']='WR'
print(totalwrdata[totalwrdata['Player']== 'A.J. Brown '])
```

```
        Player  FPTS_1 Team  FPTS_2  FPTS_3  FPTS_4  FPTS_5  FPTS_6  FPTS_7  \
0  A.J. Brown     -1.1  PHI     5.3     8.0     5.6     7.4    11.4     9.7

   FPTS_8  FPTS_9  FPTS_10  FPTS_11  FPTS_12  FPTS_13  FPTS_14  FPTS_15  \
0     0.8     0.0     12.6     25.0     19.7     13.1     12.7     29.5

   FPTS_16  AvgFantasyPoints Pos
0     13.1             11.52  WR
```

In [47]:
```
totaltedata = pd.DataFrame()
# Iterate through each week (1 to 16)
for i in range(1, 17):
    # Read the weekly TE data
    weeklydata_TE = pd.read_csv(f"/content/drive/MyDrive/Fantasy Football Analysis 202

    # Drop the unnecessary columns
    columns_to_drop = ["ATT", "Rank", "YDS", "Y/R", "LG", "20+", "TD", "REC", "TGT", '
    weeklydata_TE = weeklydata_TE.drop(columns=columns_to_drop)

    # Rename the FPTS column to indicate the week number
    weeklydata_TE = weeklydata_TE.rename(columns={"FPTS": f"FPTS_{i}"})

    # Extract the team name and clean the player names
    weeklydata_TE['Team'] = weeklydata_TE['Player'].str.split('(').str[1].str.replace(
    weeklydata_TE['Player'] = weeklydata_TE['Player'].str.split('(').str[0]

    # Sort the data by player name
    weeklydata_TE = weeklydata_TE.sort_values('Player')

    # If it's the first week, initialize the totaltedata DataFrame
    if i == 1:
```

```
        totaltedata = weeklydata_TE
    else:
        # For subsequent weeks, merge the new data with the totaltedata DataFrame
        totaltedata = pd.merge(totaltedata, weeklydata_TE, on=['Player', 'Team'], how=



totaltedata['AvgFantasyPoints']=((totaltedata['FPTS_1']+totaltedata['FPTS_2']+totaltec
totaltedata['Pos']='TE'
print(totaltedata.head())
```

```
                  Player  FPTS_1 Team  FPTS_2  FPTS_3  FPTS_4  FPTS_5  \
0              AJ Barner      0.0  SEA     0.0     0.0     0.0     0.0
1           Adam Trautman      3.4  DEN     0.0     0.0     0.0     8.6
2  Albert Okwuegbunam Jr.      0.0  PHI     0.0     0.0     0.0     0.0
3            Andrew Beck      0.0  HOU     0.2     6.0     0.0     0.0
4          Andrew DePaola      0.0  MIN     0.0     0.0     0.0     0.0

   FPTS_6  FPTS_7  FPTS_8  FPTS_9  FPTS_10  FPTS_11  FPTS_12  FPTS_13  \
0     0.0     0.0     0.0     0.0      0.0      0.0      0.0      0.0
1     0.4     0.5     0.0     0.0      1.5      3.3      6.8      0.6
2     0.0     0.0     0.0     0.0      0.0      0.0      0.0      0.0
3     0.0     0.0     4.9     0.5      0.0      0.0      0.1      0.5
4     0.0     0.0     0.0     0.0      0.0      0.0      0.0      0.0

   FPTS_14  FPTS_15  FPTS_16  AvgFantasyPoints Pos
0      0.0      0.0      0.0          0.000000  TE
1      7.9      2.4      0.0          2.360000  TE
2      0.0      0.0      0.0          0.000000  TE
3      2.6     -0.5      6.6          1.393333  TE
4      0.0      0.0      0.0          0.000000  TE
```

The next coding string takes a while so be patient

In [49]:
```
statmashup = pd.concat([totalqbdata, totalrbdata, totalwrdata, totaltedata], axis=0, s
#removeAway goes through the team names in the Schedule and the Defense data frames an
def removeAway(x):
    x = x.replace("@", "")
    x = x.replace("JAX", "JAC")
    x = x.replace("SFO", "SF")
    x = x.replace("GNB", "GB")
    x = x.replace("OAK", "LV")
    x = x.replace("NEW", "NE")
    x = x.replace("NOR", "NO")
    x = x.replace("TAM", "TB")
    x = x.replace("KAN", "KC")
    return x

Schedule2023["Team"] = Schedule2023["Team"].apply(removeAway)
Defense["Team"] = Defense["Team"].apply(removeAway)


for i in range(1, 17):
    Schedule2023["W" + str(i)] = Schedule2023["W" + str(i)].apply(removeAway)




#Stars to merge some dataframes together on the Players names and eliminates the dupli
```

```
statmashup = pd.merge(statmashup, totalswithgamesplayed, on='Player')
statmashup=statmashup.drop('Rank_x', axis=1)
statmashup.rename(columns={'Rank_y': 'Rank'}, inplace=True)


sorted_df = statmashup.sort_values(by='AvgFantasyPoints', ascending=False)
sorted_df = sorted_df.drop(columns=['FPTS/G'])

merged_df = pd.merge(sorted_df, Schedule2023, left_on='Team', right_on='Team', how='le


#Attaches the ranks of the defenses played each week to the rows for each player
rank_lookup = {}
for _, row in Defense.iterrows():
    rank_lookup[row['Team']] = {
        'QB': row['QB Rank'],
        'RB': row['RB Rank'],
        'WR': row['WR Rank'],
        'TE': row['TE Rank']
}

for week in ['W1', 'W2', 'W3', 'W4', 'W5', 'W6', 'W7', 'W8', 'W9', 'W10', 'W11', 'W12'
    merged_df[f'{week}_Opponent_Rank'] = merged_df.apply(
        lambda row: rank_lookup[row[week]].get(row['Pos'], None) if row[week] in rank_
            week] != 'BYE' else None,
        axis=1
    )

statmashup2=merged_df

#Just a check to allow me to visually check the team names in the Schedule for correct

print(Schedule2023)
```

	Team	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	\
0	ARI	WAS	NYG	DAL	SF	CIN	LAR	SEA	BAL	CLE	ATL	HOU	LAR	PIT	BYE	
1	ATL	CAR	GB	DET	JAC	HOU	WAS	TB	TEN	MIN	ARI	BYE	NO	NYJ	TB	
2	BAL	HOU	CIN	IND	CLE	PIT	TEN	DET	ARI	SEA	CLE	CIN	LAC	BYE	LAR	
3	BUF	NYJ	LV	WAS	MIA	JAC	NYG	NE	TB	CIN	DEN	NYJ	PHI	BYE	KC	
4	CAR	ATL	NO	SEA	MIN	DET	MIA	BYE	HOU	IND	CHI	DAL	TEN	TB	NO	
5	CHI	GB	TB	KC	DEN	WAS	MIN	LV	LAC	NO	CAR	DET	MIN	BYE	DET	
6	CIN	CLE	BAL	LAR	TEN	ARI	SEA	BYE	SF	BUF	HOU	BAL	PIT	JAC	IND	
7	CLE	CIN	PIT	TEN	BAL	BYE	SF	IND	SEA	ARI	BAL	PIT	DEN	LAR	JAC	
8	DAL	NYG	NYJ	ARI	NE	SF	LAC	BYE	LAR	PHI	NYG	CAR	WAS	SEA	PHI	
9	DEN	LV	WAS	MIA	CHI	NYJ	KC	GB	KC	BYE	BUF	MIN	CLE	HOU	LAC	
10	DET	KC	SEA	ATL	GB	CAR	TB	BAL	LV	BYE	LAC	CHI	GB	NO	CHI	
11	GB	CHI	ATL	NO	DET	LV	BYE	DEN	MIN	LAR	PIT	LAC	DET	KC	NYG	
12	HOU	BAL	IND	JAC	PIT	ATL	NO	BYE	CAR	TB	CIN	ARI	JAC	DEN	NYJ	
13	IND	JAC	HOU	BAL	LAR	TEN	JAC	CLE	NO	CAR	NE	BYE	TB	TEN	CIN	
14	JAC	IND	KC	HOU	ATL	BUF	IND	NO	PIT	BYE	SF	TEN	HOU	CIN	CLE	
15	KC	DET	JAC	CHI	NYJ	MIN	DEN	LAC	DEN	MIA	BYE	PHI	LV	GB	BUF	
16	LAC	MIA	TEN	MIN	LV	BYE	DAL	KC	CHI	NYJ	DET	GB	BAL	NE	DEN	
17	LAR	SEA	SF	CIN	IND	PHI	ARI	PIT	DAL	GB	BYE	SEA	ARI	CLE	BAL	
18	LV	DEN	BUF	PIT	LAC	GB	NE	CHI	DET	NYG	NYJ	MIA	KC	BYE	MIN	
19	MIA	LAC	NE	DEN	BUF	NYG	CAR	PHI	NE	KC	BYE	LV	NYJ	WAS	TEN	
20	MIN	TB	PHI	LAC	CAR	KC	CHI	SF	GB	ATL	NO	DEN	CHI	BYE	LV	
21	NE	PHI	MIA	NYJ	DAL	NO	LV	BUF	MIA	WAS	IND	BYE	NYG	LAC	PIT	
22	NO	TEN	CAR	GB	TB	NE	HOU	JAC	IND	CHI	MIN	BYE	ATL	DET	CAR	
23	NYG	DAL	ARI	SF	SEA	MIA	BUF	WAS	NYJ	LV	DAL	WAS	NE	BYE	GB	
24	NYJ	BUF	DAL	NE	KC	DEN	PHI	BYE	NYG	LAC	LV	BUF	MIA	ATL	HOU	
25	PHI	NE	MIN	TB	WAS	LAR	NYJ	MIA	WAS	DAL	BYE	KC	BUF	SF	DAL	
26	PIT	SF	CLE	LV	HOU	BAL	BYE	LAR	JAC	TEN	GB	CLE	CIN	ARI	NE	
27	SEA	LAR	DET	CAR	NYG	BYE	CIN	ARI	CLE	BAL	WAS	LAR	SF	DAL	SF	
28	SF	PIT	LAR	NYG	ARI	DAL	CLE	MIN	CIN	BYE	JAC	TB	SEA	PHI	SEA	
29	TB	MIN	CHI	PHI	NO	BYE	DET	ATL	BUF	HOU	TEN	SF	IND	CAR	ATL	
30	TEN	NO	LAC	CLE	CIN	IND	BAL	BYE	ATL	PIT	TB	JAC	CAR	IND	MIA	
31	WAS	ARI	DEN	BUF	PHI	CHI	ATL	NYG	PHI	NE	SEA	NYG	DAL	MIA	BYE	

	W15	W16
0	SF	CHI
1	CAR	IND
2	JAC	SF
3	DAL	LAC
4	ATL	GB
5	CLE	ARI
6	MIN	PIT
7	CHI	HOU
8	BUF	MIA
9	DET	NE
10	DEN	MIN
11	TB	CAR
12	TEN	CLE
13	PIT	ATL
14	BAL	TB
15	NE	LV
16	LV	BUF
17	WAS	NO
18	LAC	KC
19	NYJ	DAL
20	CIN	DET
21	KC	DEN
22	NYG	LAR
23	NO	PHI
24	MIA	WAS

```
25  SEA  NYG
26  IND  CIN
27  PHI  TEN
28  ARI  BAL
29   GB  JAC
30  HOU  SEA
31  LAR  NYJ
```

In [50]:
```
print(totalswithgamesplayed.head())
```

```
         Player     G  FPTS/G  Rank POS
0   Josh Allen   17.0    24.2   1.0  QB
0  CeeDee Lamb   17.0    15.8   2.0  WR
1  Jalen Hurts   17.0    21.9   2.0  QB
0  Sam LaPorta   17.0     9.0   2.0  TE
1  Tyreek Hill   16.0    16.1   3.0  WR
```

In [51]:
```
# Ensure all team columns are strings and strip whitespace
Schedule2023['Team'] = Schedule2023['Team'].astype(str).str.strip()
Defense['Team'] = Defense['Team'].astype(str).str.strip()
statmashup2['Team'] = statmashup2['Team'].astype(str).str.strip()
statmashup2['Pos'] = statmashup2['Pos'].astype(str).str.strip()
statmashup2['Player'] = statmashup2['Player'].astype(str).str.strip()
# Also ensure the index of Schedule2023 and Defense are strings and stripped
Schedule2023.index = Schedule2023.index.astype(str).str.strip()
Defense.index = Defense.index.astype(str).str.strip()
totalswithgamesplayed['Player']=totalswithgamesplayed['Player'].astype(str).str.strip(
# Convert 'Team' columns to strings
Schedule2023['Team'] = Schedule2023['Team'].astype(str)
Defense['Team'] = Defense['Team'].astype(str)
statmashup2['Team'] = statmashup2['Team'].astype(str)

# Set 'Team' as the index for relevant DataFrames
Schedule2023.set_index('Team', inplace=True)
Defense.set_index('Team', inplace=True)
statmashup2.set_index('Player', inplace=True)
```

In [60]:
```
totalswithgamesplayed.set_index('Player', inplace=True)
```

In [63]:
```
print(totalswithgamesplayed.head())
```

```
                G  FPTS/G  Rank POS
Player
Josh Allen   17.0    24.2   1.0  QB
CeeDee Lamb  17.0    15.8   2.0  WR
Jalen Hurts  17.0    21.9   2.0  QB
Sam LaPorta  17.0     9.0   2.0  TE
Tyreek Hill  16.0    16.1   3.0  WR
```

In [57]:
```
def combine_duplicates(df):
    # Assuming numerical columns should be summed and non-numerical columns should be
    return df.groupby(df.index).agg(lambda x: x.sum() if pd.api.types.is_numeric_dtype

# Combine duplicates in the statmashup2 DataFrame
statmashup2 = combine_duplicates(statmashup2)
```

Handling for duplicate data points. Eliminates duplicate data within teh database to prevent duplication errors

In [68]:

```
def correlation(player):
    try:
        pointdiffs = []
        rankings = []

        # Check if player exists in the DataFrame index
        if player not in statmashup2.index:
            print(f"Player {player} not found in statmashup2")
            return 0

        # Fetch the team and position for the player
        player_data = statmashup2.loc[player]

        team = player_data.get("Team")
        pos = player_data.get("Pos")

        # Check if team exists in Schedule2023
        if team not in Schedule2023.index:
            print(f"Team {team} not found in Schedule2023")
            return 0

        for i in range(1, 16):
            week_key = "Week " + str(i)
            points = player_data.get(week_key)
            avg_fantasy_points = player_data.get("AvgFantasyPoints")



            opponent = Schedule2023.loc[team].get("W" + str(i))

            if opponent == "BYE" or pd.isna(opponent):

                continue

            if opponent not in Defense.index:
                print(f"Opponent {opponent} not found in Defense")
                continue

            if points is None or avg_fantasy_points is None:

                continue

            pointdiff = points - avg_fantasy_points
            ranking = Defense.loc[opponent].get(pos + " Rank")

            if ranking is None:
                print(f"Missing defense ranking for opponent {opponent} in {week_key}"
                continue

            pointdiffs.append(round(pointdiff, 2))
            rankings.append(ranking)

        if not pointdiffs or not rankings:

            return 0

        x = pd.Series(rankings)
        y = pd.Series(pointdiffs)
```

```
        correlation_value = round(x.corr(y), 2)
        return correlation_value
    except Exception as e:
        print(f"Exception occurred for player {player}: {e}")
        return 0
```

Lots of check point prints built into here. If this were ever released to the public. I'd eliminate all of the print statements to check for potential errors.

In [59]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler


def graph(player):
    pointdiffs = []
    rankings = []
    team_names = []

    if player not in statmashup2.index:
        print(f"Player {player} not found in statmashup2")
        return

    team = statmashup2.loc[player, 'Team']
    pos = statmashup2.loc[player, 'Pos']

    for i in range(1, 17):
        points = statmashup2.loc[player].get("FPTS_" + str(i))
        opponent = Schedule2023.loc[team].get("W" + str(i))

        if opponent == "BYE" or points is None:
            continue

        pointdiff = points - statmashup2.loc[player].get("AvgFantasyPoints")
        ranking = Defense.loc[opponent].get(pos + " Rank")

        if ranking is not None:
            pointdiffs.append(round(pointdiff, 2))
            rankings.append(ranking)
            team_names.append(f"{opponent} ({ranking})")



    x = pd.Series(rankings)
    y = pd.Series(pointdiffs)

    plt.figure(figsize=(16, 8))  # Increase figure size for better readability
    plt.scatter(x, y)

    # Set the x-ticks with team names and rankings
    unique_rankings = np.unique(rankings)
    plt.xticks(ticks=unique_rankings, labels=[team_names[rankings.index(r)] for r in u

    plt.title(f"Effect of Defense Strength on {player} in 2023")
    plt.xlabel("Defense Ranking (1-32) with Team")
    plt.ylabel("Fantasy Production Above/Below Yearly Mean")
```

```
    xdata = np.array(x)
    ydata = np.array(y)
    m, b = np.polyfit(xdata, ydata, 1)
    plt.plot(x, m * x + b, color='red')  # Add line of best fit in red
    plt.show()
```

In [85]:
```
print(totalswithgamesplayed.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 974 entries, Josh Allen to nan
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   G       968 non-null    float64
 1   FPTS/G  968 non-null    float64
 2   Rank    725 non-null    float64
 3   POS     974 non-null    object 
dtypes: float64(3), object(1)
memory usage: 70.3+ KB
None
```

In [95]:
```
def perform_clustering(player):
    if player not in totalswithgamesplayed.index:
        print(f"Player {player} not found in the dataset.")
        return
    totalswithgamesplayed1 = totalswithgamesplayed.dropna()
    position = totalswithgamesplayed1.loc[player, 'POS']

    # Filter dataset to include only players of the same position
    position_players = totalswithgamesplayed1[totalswithgamesplayed1['POS'] == positic
    features = ['FPTS/G']
    data_for_clustering = position_players[features]

    if data_for_clustering.empty:
        print(f"No data available for players of position {position}.")
        return

    # Standardize the data
    scaler = StandardScaler()
    scaled_features = scaler.fit_transform(data_for_clustering)

    # Perform K-Means Clustering
    kmeans = KMeans(n_clusters=5, random_state=42, n_init=10)  # Explicitly set n_init
    clusters = kmeans.fit_predict(scaled_features)

    # Add cluster labels to the DataFrame using .loc to avoid SettingWithCopyWarning
    position_players.loc[:, 'Cluster'] = clusters

    # Visualize Clusters
    plt.figure(figsize=(14, 8))
    for cluster in range(5):
        cluster_data = position_players[position_players['Cluster'] == cluster]
        plt.scatter(cluster_data['FPTS/G'], cluster_data['FPTS/G'], label=f'Cluster {c

    plt.scatter(totalswithgamesplayed1.loc[player, 'FPTS/G'], totalswithgamesplayed1.l

    plt.title('Player Clustering Based on Fantasy Points per Game')
    plt.xlabel('Fantasy Points per Game')
    plt.ylabel('Fantasy Points per Game')
    plt.legend()
```

```
    plt.show()

    # Print similar players
    player_cluster = position_players.loc[player, 'Cluster']
    similar_players = position_players[position_players['Cluster'] == player_cluster].
    similar_players.remove(player)  # Remove the player themselves
    print(f"Players similar to {player}: {', '.join(similar_players)}")
```

In [65]:
```
statmashup2 = statmashup2[statmashup2['Team'] != 'FA']

statmashup2.drop(index='nan', inplace=True)
```

```
<ipython-input-65-c19ab4c93594>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  statmashup2.drop(index='nan', inplace=True)
```

In [66]:
```
statmashup2 = combine_duplicates(statmashup2)
```

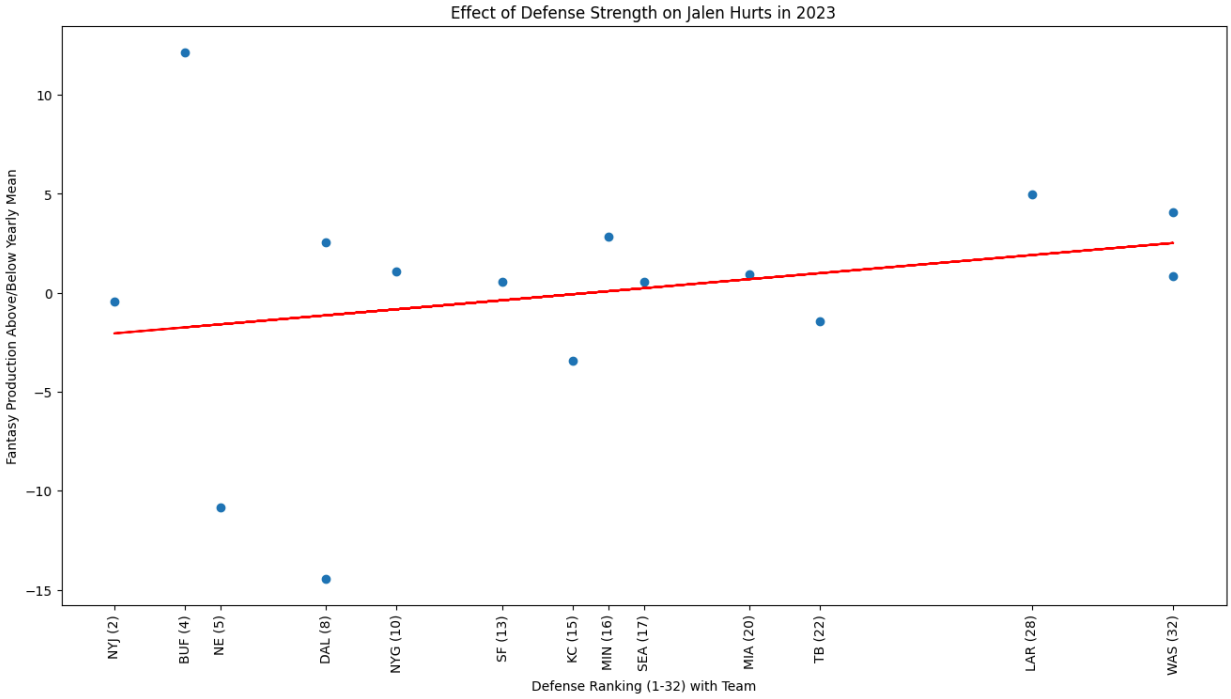
In [69]:
```
statmashup2 = statmashup2.assign(Correlation=statmashup2.index.map(correlation))
statmashup2 = statmashup2.assign(AbsCorrelation=statmashup2.get("Correlation").apply(a
statmashup2['Correlation'] = statmashup2['Correlation'].astype(float)
statmashup2['AbsCorrelation'] = statmashup2['AbsCorrelation'].astype(float)
sigdata = statmashup2[statmashup2.get("AvgFantasyPoints") > 6].sort_values("AbsCorrela
sigdata = sigdata[sigdata.get("AbsCorrelation") >= 0]
sigdata = sigdata[sigdata.get("G") > 6]
```

In [97]:
```
# Assuming statmashup2, Schedule2023, and Defense are already defined DataFrames
def get_top_players_by_position(position, top_n=60):
    filtered_df = statmashup2[statmashup2['Pos'] == position]
    sorted_df = filtered_df.sort_values(by='AvgFantasyPoints', ascending=False)
    return sorted_df.head(top_n)

def main():
    # Prompt user for a position
    position = input("Enter the position you wish to analyze (e.g., QB, RB, WR, TE): "

    # Get top 60 players for the entered position
    top_players = get_top_players_by_position(position, 60)
    if top_players.empty:
        print(f"No players found for position {position}.")
        return

    # Display the top 60 players
    print("\nTop 60 players for position", position)
    print(top_players[['AvgFantasyPoints']])

    # Prompt user to select a player
    player = input("\nEnter the name of the player you wish to analyze (exactly as dis

    if player not in top_players.index:
        print(f"Player {player} not found in the top 60 players for position {positior
        return

    # Execute the graph function on the selected player
```

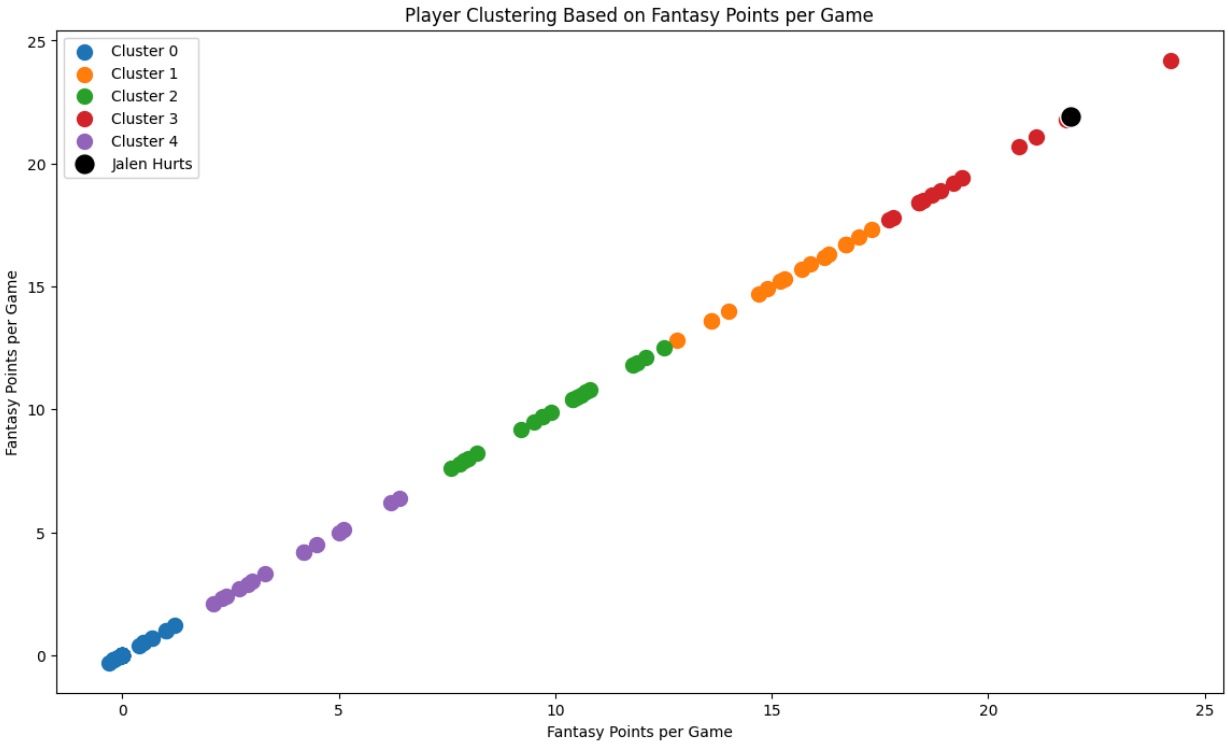
```
    graph(player)
    perform_clustering(player)

if __name__ == "__main__":
    main()
```

```
Enter the position you wish to analyze (e.g., QB, RB, WR, TE): QB

Top 60 players for position QB
                         AvgFantasyPoints
Player
Josh Allen                       24.240000
Jalen Hurts                      23.340000
Dak Prescott                     20.300000
Lamar Jackson                    20.126667
Brock Purdy                      19.266667
Jordan Love                      18.880000
Patrick Mahomes II               18.806667
Jared Goff                       17.953333
Russell Wilson                   17.660000
Sam Howell                       17.366667
Baker Mayfield                   17.300000
Trevor Lawrence                  17.226667
Tua Tagovailoa                   17.193333
C.J. Stroud                      16.440000
Justin Herbert                   16.013333
Matthew Stafford                 15.980000
Joshua Dobbs                     14.033333
Derek Carr                       13.753333
Justin Fields                    13.686667
Geno Smith                       13.206667
Gardner Minshew II               12.226667
Desmond Ridder                   11.453333
Bryce Young                      10.573333
Kirk Cousins                     10.326667
Joe Burrow                       10.213333
Zach Wilson                       8.453333
Mac Jones                         7.886667
Kenny Pickett                     7.613333
Jake Browning                     7.360000
Kyler Murray                      7.260000
Will Levis                        7.166667
Tommy DeVito                      6.393333
Aidan O'Connell                   6.306667
Deshaun Watson                    6.060000
Joe Flacco                        5.680000
Anthony Richardson                4.906667
Jimmy Garoppolo                   4.873333
Ryan Tannehill                    4.460000
Daniel Jones                      4.200000
Bailey Zappe                      4.020000
Tyson Bagent                      3.953333
Mitchell Trubisky                 3.580000
Taylor Heinicke                   3.460000
Tyrod Taylor                      3.453333
Nick Mullens                      2.860000
Easton Stick                      2.680000
Drew Lock                         2.140000
Dorian Thompson-Robinson          1.740000
P.J. Walker                       1.660000
Andy Dalton                       1.573333
Jacoby Brissett                   1.526667
Mason Rudolph                     1.340000
Davis Mills                       1.193333
Tim Boyle                         0.993333
Jameis Winston                    0.993333
```

```
Trevor Siemian                  0.886667
Case Keenum                     0.853333
C.J. Beathard                   0.800000
Clayton Tune                    0.626667
Tyler Huntley                   0.473333

Enter the name of the player you wish to analyze (exactly as displayed or you can cop
y and paste): Jalen Hurts
```




```
<ipython-input-95-51f2ddc1953d>:26: 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/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  position_players.loc[:, 'Cluster'] = clusters
```




```
Players similar to Jalen Hurts: Josh Allen, Dak Prescott, Lamar Jackson, Jordan Love,
Brock Purdy, Jared Goff, Patrick Mahomes II, C.J. Stroud, Russell Wilson, Justin Herb
ert, Justin Fields, Kirk Cousins, Kyler Murray, Joe Flacco, Anthony Richardson
```

you'll notice that some teams show two dots. This reflects the fact that within the division, teams play each other twice, so they will have two points plotted per in team division. If there are not plot points for a division game, it indicates that for whatever reason, the playe did not play, or they scored 0 points. You may want to do further research on what happened during that week of play.

Also, fantasy seasons are short compared to regular season, so we only looked at 16 weeks which in almost every instance included a bye week. This should result in 15 data points per player and anything less than that indicates either injury, or did not play for whatever reason.

Also observe in the graphs that the teams on the bottom go from best to worst defensive ranking vs to position. If there is a positive slope in the mean of the graph, that means the player overall tends to perform better as the defensive ranking worsens, and the player performs worse as the defensive ranking improves. This is the exact opposite if the slope of the mean line is negative.

https://github.com/anishkasam/fantasy-football/blob/master/2020.py

Got the initial idea for the build of this from this project. Used some base coding build, but I wasn't able to find databases for the 2023 stats built the same as his. This resulted in alot of data cleaning of the .cvs files I was able to find, and then merging the data together in a way that was consistent with what we wanted to analyze.

I added on some additional functionality to be able to specify in on individual players to analyze, and a cluster with a list of players of the same position who would perform similarly.

Definitely got some assistance from Chat GPT towards the end as I wanted to get the project finished.

https://chatgpt.com/

Going into the 2024 season, as written, I'd say the best use of the code would be to determine how a player might perform against an upcoming team. Some things to consider are: Did the defensive coordinator stay the same, were there major changes in defensive personnel between seasons, and was there a change in QB, head coach, or offensive coordinator. These are all things to consider in making choices as well, that are not taken into account in the information provided because we are looking at historical data that only takes into acccount players and teams. Coaches are not taken into account here.

Lastly, code was added in to provide a clustering chart and use that information to provide a list is similar players to the player being looked at. This information can assist with choosing players of similar performance, who may be easier to attain in a draft format, or cheaper in a salary restricted format. It also provides a black dot, so that it can be seen where the player performs within the clustering set up.