Class Project: Should NFL Teams Invest in the Runningback Position?

Introduction

Research Question

Recently, in the NFL, it seems that the runningback position has lost some of its value. Compared to other positions like quarterback and receiver, runningbacks get paid significantly less. Furthermore, in recent history, a trend has been that Super Bowl teams have not invested much in the runningback position, thus questioning its worth compared to other positions. In this analysis, I hope to determine whether this devaluing of the runningback position has a statistical basis.

For this project, some of the variables that I use are rushing, receiving, and passing yards. Rushing, passing, and receiving yards are essentially how many yards a player moves the ball on offense through running the ball (rushing), throwing the ball (passing), and catching the ball (receiving). I decided to use yards for each position because I believe that yards are the most reliable and consistent statistic for determining a player's offensive production. For example, while touchdowns ultimately accumulate points for a team, a player can have a lot of short touchdowns and consequently not a lot of yards. Therefore, he may not actually be contributing much to his team's offensive production. Meanwhile, a player with a lot of yards but not a lot of touchdowns is still contributing to his team's offensive production but may not be scoring touchdowns due to situational factors. For example, the Philadelphia Eagles are a team that like to do quarterback runs when they are close to scoring a touchdown. Therefore, while the runningback for the Eagles may be accumulating rushing yards, he is not scoring many touchdowns. Nonetheless, he is still contributing to the Eagles offensive production, which can be seen through the rushing yards statistic.

Another variable that I use in this project is cap spent. In football, the salary cap is the maximum amount of money that I team can spend on all of its players' salaries. Therefore, the cap spent is essentially how much money a team spends on one player.

Data Sourcing: I obtained the data for this project from a number of sources.

I obtained passing, rushing, and receiving stats from Next Gen Stats. https://nextgenstats.nfl.com/

I obtained salary information on quarterbacks, runningbacks, and receivers from OverTheCap. https://overthecap.com/

Lastly, I obtained the wins for each NFL team for the 2021, 2022, and 2023 seasons from Pro Football Reference. https://www.pro-football-reference.com/

Other Analyses: One project that I took some inspiration from was "A Short NFL analytics project" by Chris Raddatz. Although his project analyzed defensive data, I liked one visual of his, which was comparing the top tacklers at a position vs the bottom tacklers at a position. In my analysis, I create a similar visual. Link: https://medium.com/@raddatzchris/a-short-nfl-analytics-project-b6cbde9d877c

Another project that I looked at was, "Where Should NFL Teams Invest Their Financial Resources?" by Peter Renkoski. His project looked at the positional salary distribution across the NFL. One interesting analysis he did was comparing the salary distribution of a very good team in the Patriots vs a bad team in the Browns. I thought this type of comparison would be useful in my project and I do a similar type of comparison. Link: https://www.samford.edu/sports-analytics/fans/2021/Where-Should-NFL-Teams-Invest-Their-Financial-Resources

Import Packages

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import statistics
import seaborn as sns
```

Part 1: Examining Next Gen Statistics

Data Cleaning: Next Gen Statistics

In this part, I import the Next Gen Statistics Data Frames for rushing, receiving, and passing. I then reduce the Data Frames to only look at statistics for the whole regular season, which is week 0.

In [2]:
```
#Next Gen DataFrames
ngs_2023_receiving = pd.read_csv('ngs_2023_receiving.csv')
#only stats for whole season regular season
ngs_2023_receiving_rs = ngs_2023_receiving[ngs_2023_receiving["week"] == 0]

ngs_2023_rushing = pd.read_csv('ngs_2023_rushing.csv')
#regular season
ngs_2023_rushing_rs = ngs_2023_rushing[ngs_2023_rushing["week"] == 0]

ngs_2023_passing = pd.read_csv('ngs_2023_passing.csv')
#regular season
ngs_2023_passing_rs = ngs_2023_passing[ngs_2023_passing["week"] == 0]
ngs_2023_passing_rs.head(1)
```

Out[2]:

	season	season_type	week	player_display_name	player_position	team_abbr	avg_time_to_throw	avg_completed_air_yards	avg_intended_air_yards	avg_a
0	2023	REG	0	Tua Tagovailoa	QB	MIA	2.363346	6.11299	7.814328	

1 rows × 29 columns

Analyzing the Skill Gap Within Various Positions

To analyze the skill gap within the quarterback, runningback, and receiver positions, I split each position into two groups, the top players at that position and the bottom players at the position. To determine a top and bottom players at a position, I first wrangled the data to get the yards for each starting player at a position for each team, i.e. the player who leads his respective team in yards. Therefore, at this point, the DataFrame for each position contained 32 players who led their team in yards. Next, I sorted these DataFrames in ascending order based on yards. Afterwards, I took the sum of the first 5 players to get the total yards for "bottom" players at the position and the sum of the last 5 players to get the total yards for the "top" players at the position. Finally, I calculated the percent difference between the top

and bottom players at a position and also created a bar chart displaying these differences between top and bottom players. An example of the data wrangling for the receiver position is displayed below, while the quarterback and runningback data wrangling is contained within the appendix.

In [4]:
```
team_best_receiving = ngs_2023_receiving_rs.groupby("team_abbr").agg(
    leading_receiver = ("yards", "max")
)
team_best_receiving = team_best_receiving.sort_values("leading_receiver")
#bottom 5 starting receivers
bottom_receivers = team_best_receiving[0:5]
#top 5 starting receivers
top_receivers = team_best_receiving[27:32]
#total yards of bottom receivers and total yards of top receivers
total_bottom_receivers = np.sum(bottom_receivers["leading_receiver"])
total_top_receivers = np.sum(top_receivers["leading_receiver"])

print(f"Total Bottom Receivers: {total_bottom_receivers}")
print(f"Total Top Receivers: {total_top_receivers}")

#Percent Difference
percent_diff_receivers = (abs(total_top_receivers - total_bottom_receivers)/
                          ((total_top_receivers + total_bottom_receivers)/2)) * 100
print(f"Percent Difference b/w Top and Bottom Receivers: {percent_diff_receivers}")
```

```
Total Bottom Receivers: 3721.0
Total Top Receivers: 8005.0
Percent Difference b/w Top and Bottom Receivers: 73.06839501961453
```

Figure 1a: Table Containing Yards of Top and Bottom Players at a Position

In [28]:
```
data = {
    "Category": ["Total Bottom Receivers", "Total Top Receivers", "Total Bottom Rushers",
                 "Total Top Rushers", "Total Bottom Passers", "Total Top Passers"],
    "Yards": [3721.0, 8005.0, 3231.0, 5941.0, 8634.0, 22301.0],}
df = pd.DataFrame(data)
df
```

Out[28]:

	Category	Yards
0	Total Bottom Receivers	3721.0
1	Total Top Receivers	8005.0
2	Total Bottom Rushers	3231.0
3	Total Top Rushers	5941.0
4	Total Bottom Passers	8634.0
5	Total Top Passers	22301.0

Figure 1b: Table Containing Percent Differences b/w Top and Bottom Players at a Position

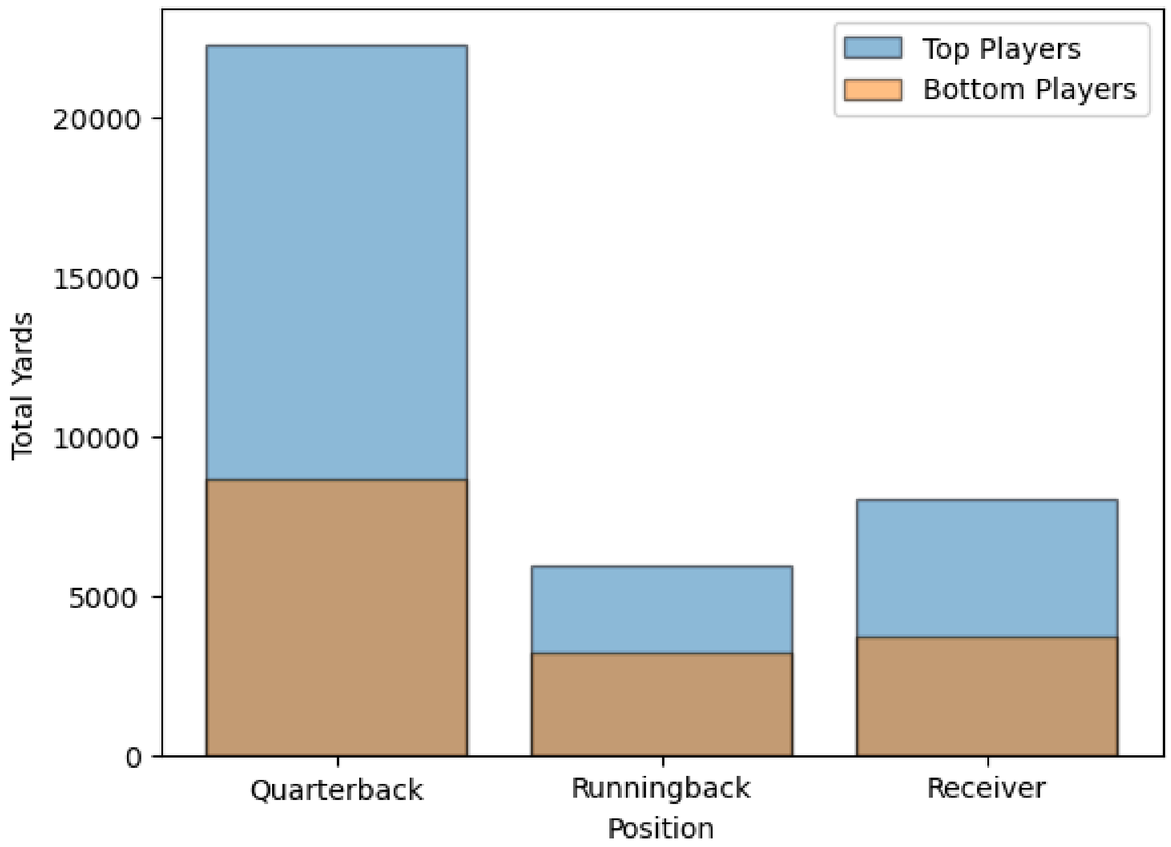
In [29]:
```
#Percent Differences
percent_diff_receivers = (abs(8005 - 3721)/((8005 + 3721)/2)) * 100
percent_diff_rushers = (abs(5941 - 3231)/((5941 + 3231)/2)) * 100
percent_diff_passers = (abs(22301 - 8634)/((22301 + 8634)/2)) * 100

data2 = {
    "Position": ["Receivers", "Runningbacks", "Quarterbacks"],
    "Percent Difference b/w Top and Bottom": [percent_diff_receivers, percent_diff_rushers,
                                              percent_diff_passers]}
df2 = pd.DataFrame(data2)
df2
```

Out[29]:

	Position	Percent Difference b/w Top and Bottom
0	Receivers	73.068395
1	Runningbacks	59.092891
2	Quarterbacks	88.359463

Figure 1c: Bar Chart Containing Total Yards of Top vs. Bottom Players at a Position

In [98]:
```
#Bar Chart
positions = ["Quarterback", "Runningback", "Receiver"]
bottom_statistics = [total_bottom_passers, total_bottom_rushers, total_bottom_receivers]
top_statistics = [total_top_passers, total_top_rushers, total_top_receivers]

fig, ax = plt.subplots()
bar1 = ax.bar(positions, top_statistics, alpha = 0.5, edgecolor = "k", label = "Top Players")
bar2 = ax.bar(positions, bottom_statistics, alpha = 0.5, edgecolor = "k", label = "Bottom Players")
plt.legend();
plt.ylabel("Total Yards");
plt.xlabel("Position");
```



Analyzing Figure 1

Based on the graph above and the percent differences for each position, it appears that there is the least skill gap between top and bottom players for runningbacks. The percent difference for runningbacks is 59%, where it is 73% for receivers and 88% for quarterbacks. What this tells me is that out of the three positions, having one of the worst runningbacks is less significant than having one of the worst quarterbacks or receivers. This is because there is less of a gap between stats of the top and bottom runningbacks than the stats of the top and bottom quarterbacks and receivers. Looking at the graph, this is seen because there is less of a distance between the top runningbacks and the bottom runningbacks, opposed to the large distance between the top and bottom quarterbacks and receivers. Ultimately, this information can be useful to an NFL team because out of the three positions analyzed, investing less in a runningback will likely not be as harmful as investing less in a receiver or quarterback. While there is a drop in production from a top runningback to a bottom runningback, this drop is not as drastic as the one from a top quarterback to bottom quarterback or from a top receiver to a bottom receiver as demonstrated by this figure.

Part 2: The Impact of Salaries

Import Contract DataFrames and Data Cleaning

Below is an example of importing the RB contracts for 2023 and getting the total cap each team spent on the runningback position for the 2023 season. To get this in a usable format, some data cleaning was done, inclluding removing the characters "$" and ",", as well as converting the cap column to numbers. This process was repeated for RB contracts from 2022 and 2021, as well as receiver and quarterback contracts from 2021, 2022, and 2023. This code can be seen in the appendix.

In [31]:
```
#Reading in Running Back Contracts DataFrame
RB_contracts_2023 = pd.read_csv("RB_contracts.csv")

#Remove Extra Empty Columns
RB_contracts_2023 = RB_contracts_2023[["player", "team", "cap"]]

#Remove $ and , from cap column
RB_contracts_2023["cap"] = RB_contracts_2023["cap"].str.replace("$", "")
RB_contracts_2023["cap"] = RB_contracts_2023["cap"].str.replace(",", "")

#convert column to numbers
RB_contracts_2023["cap"] = pd.to_numeric(RB_contracts_2023["cap"])

#get total rb cap for each team
RB_caps_2023 = RB_contracts_2023.groupby("team").agg(RB_cap = ("cap", "sum"))
RB_caps_2023 = RB_caps_2023.reset_index()
RB_caps_2023.head(2)
```

Out[31]:

	team	RB_cap
0	49ers	5395448
1	Bears	5775142

Import Regular Season Wins DataFrames and Data Cleaning

Below is an example of importing the DataFrame for regular season wins for each team from the 2023 season. For this part, not much data cleaning was done. This is because to get this data, I manually typed the team and wins into an excel spreadsheat, which meant it was already in a pretty usable format. However, upon reading in the data, there were some extra blank columns and rows, which I did remove, as seen below. I repeated this process of importing the DataFrames for the 2022 and 2021 seasons, which can be seen in the appendix.

In [33]:
```
team_wins_rs_2023 = pd.read_csv("team_wins_rs.csv")
#remove extra blank columns and rows
team_wins_rs_2023 = team_wins_rs_2023[["team", "wins"]]
team_wins_rs_2023.dropna(axis = 0, inplace = True)
team_wins_rs_2023.head(2)
```

Out[33]:

	team	wins
0	Ravens	13.0
1	49ers	12.0

Merge Position Cap DataFrames with Regular Season Wins DataFrames

For each season, the total cap spent on a position and the regular season wins DataFrames were merged. This resulted in 9 additional DataFrames. For example, below is an example of joining the RB caps for each team for the 2023 season and the regular season wins for each team for the 2023 season. Another detail is that the values for the RB_cap, QB_cap, and WR_cap were converted to millions of dollars so as to ensure that the graphs later on would be more visually appealing. The rest of the merging can be seen in the appendix.

In [38]:
```
#joining rb cap and wins for 2023 season
RB_cap_wins_2023 = RB_caps_2023.merge(team_wins_rs_2023, how = "left")
#convert to millions of dollars
RB_cap_wins_2023["RB_cap"] = RB_cap_wins_2023["RB_cap"]/1000000
RB_cap_wins_2023.head(2)
```

Out[38]:

	team	RB_cap	wins
0	49ers	5.395448	12.0
1	Bears	5.775142	7.0

Join Data From All Three Seasons

Next, I joined the data from all three seasons, thus creating 3 DataFrames that contained data from the last 3 seasons containing a team's cap spent on that position and their regular season wins.

In [40]:
```
#joining the data from the three seasons
RB_cap_wins_2021_2022 = pd.concat([RB_cap_wins_2021, RB_cap_wins_2022])
RB_cap_wins_total = pd.concat([RB_cap_wins_2021_2022, RB_cap_wins_2023])

QB_cap_wins_2021_2022 = pd.concat([QB_cap_wins_2021, QB_cap_wins_2022])
QB_cap_wins_total = pd.concat([QB_cap_wins_2021_2022, QB_cap_wins_2023])

WR_cap_wins_2021_2022 = pd.concat([WR_cap_wins_2021, WR_cap_wins_2022])
WR_cap_wins_total = pd.concat([WR_cap_wins_2021_2022, WR_cap_wins_2023])

RB_cap_wins_total.head(2)
```

Out[40]:

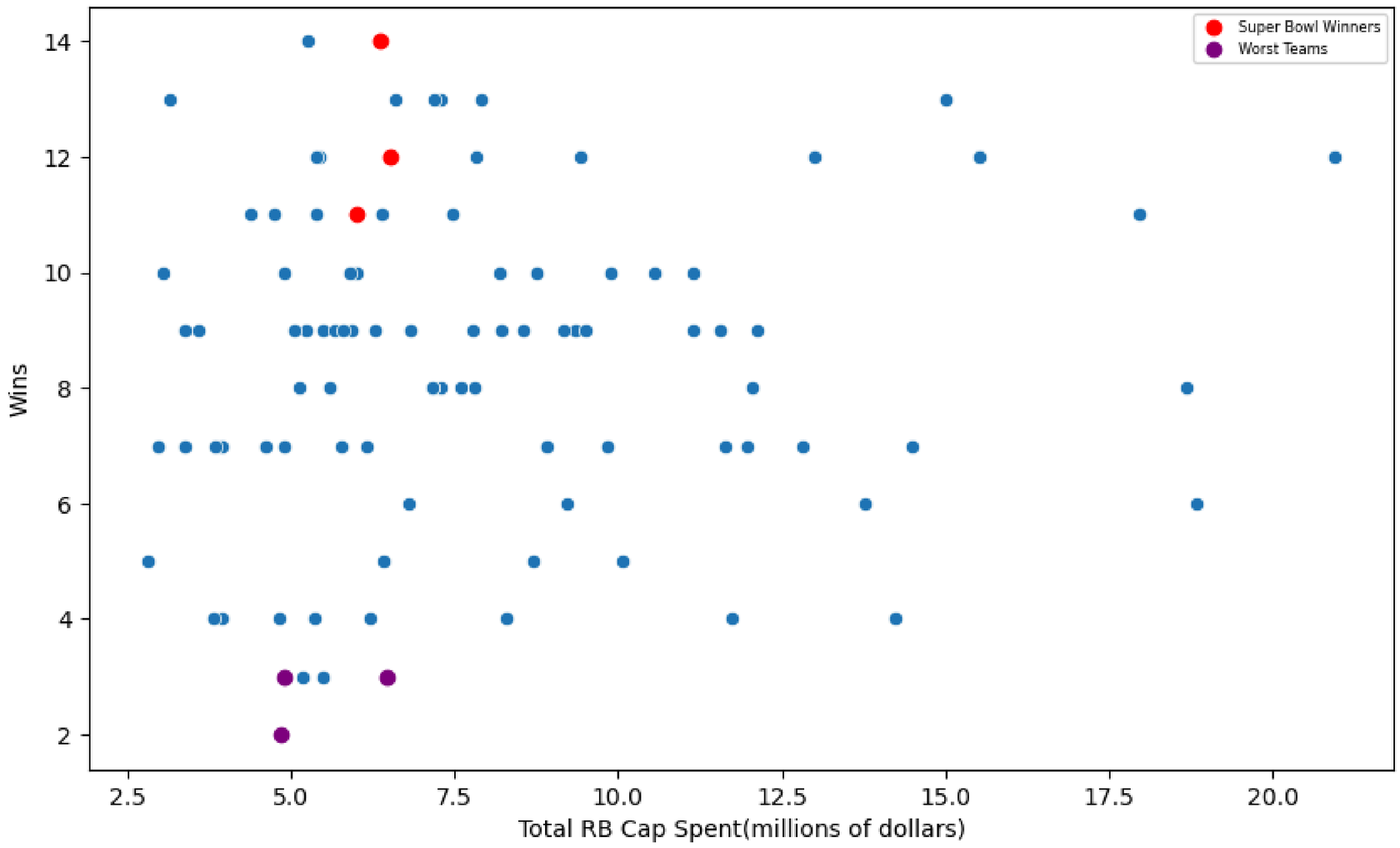
	team	RB_cap	wins
0	49ers	8.755882	10.0
1	Bears	6.793142	6.0

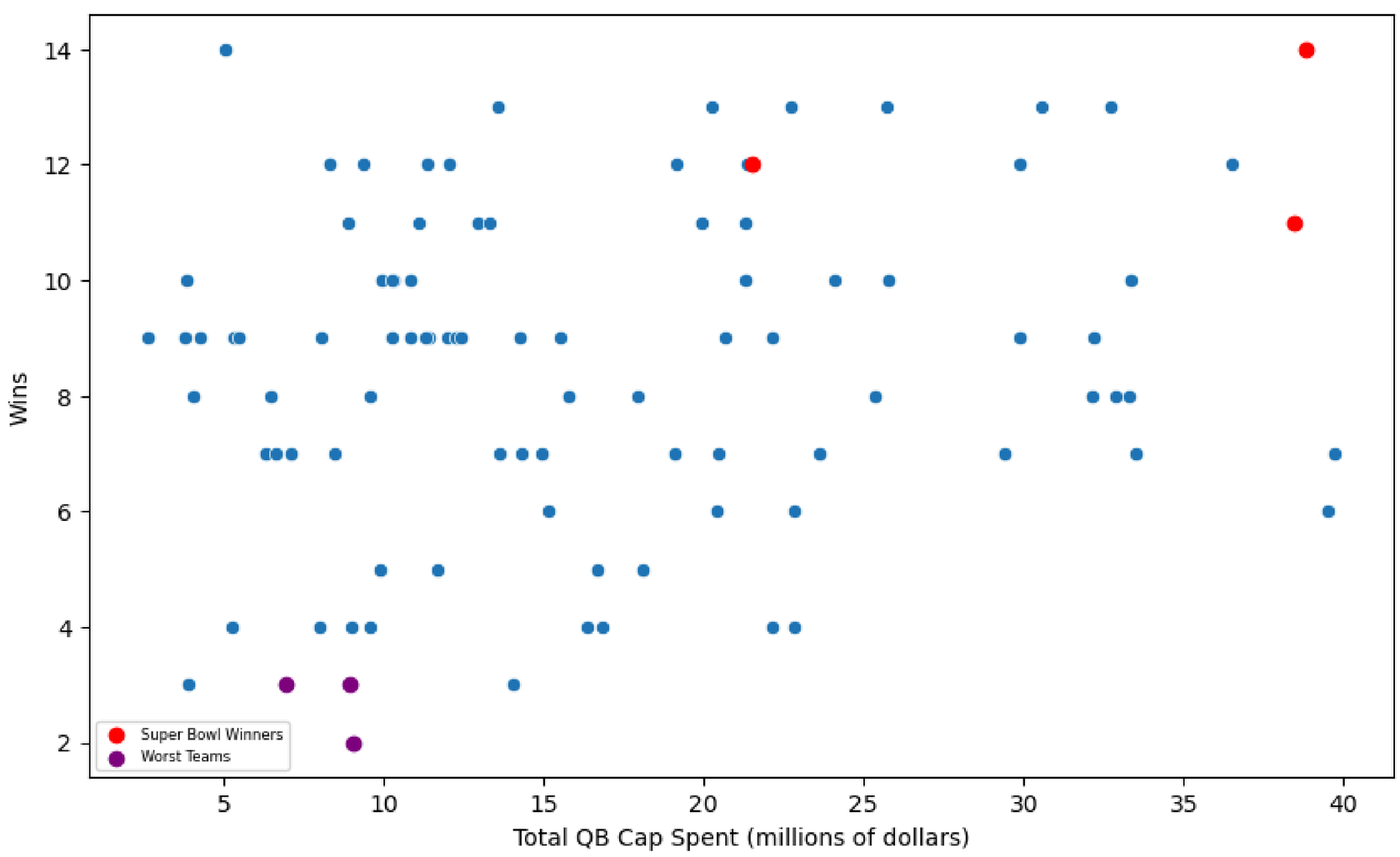
Figures 2a, 2b, and 2c: Scatterplots for Wins vs. Cap Spent on a Position

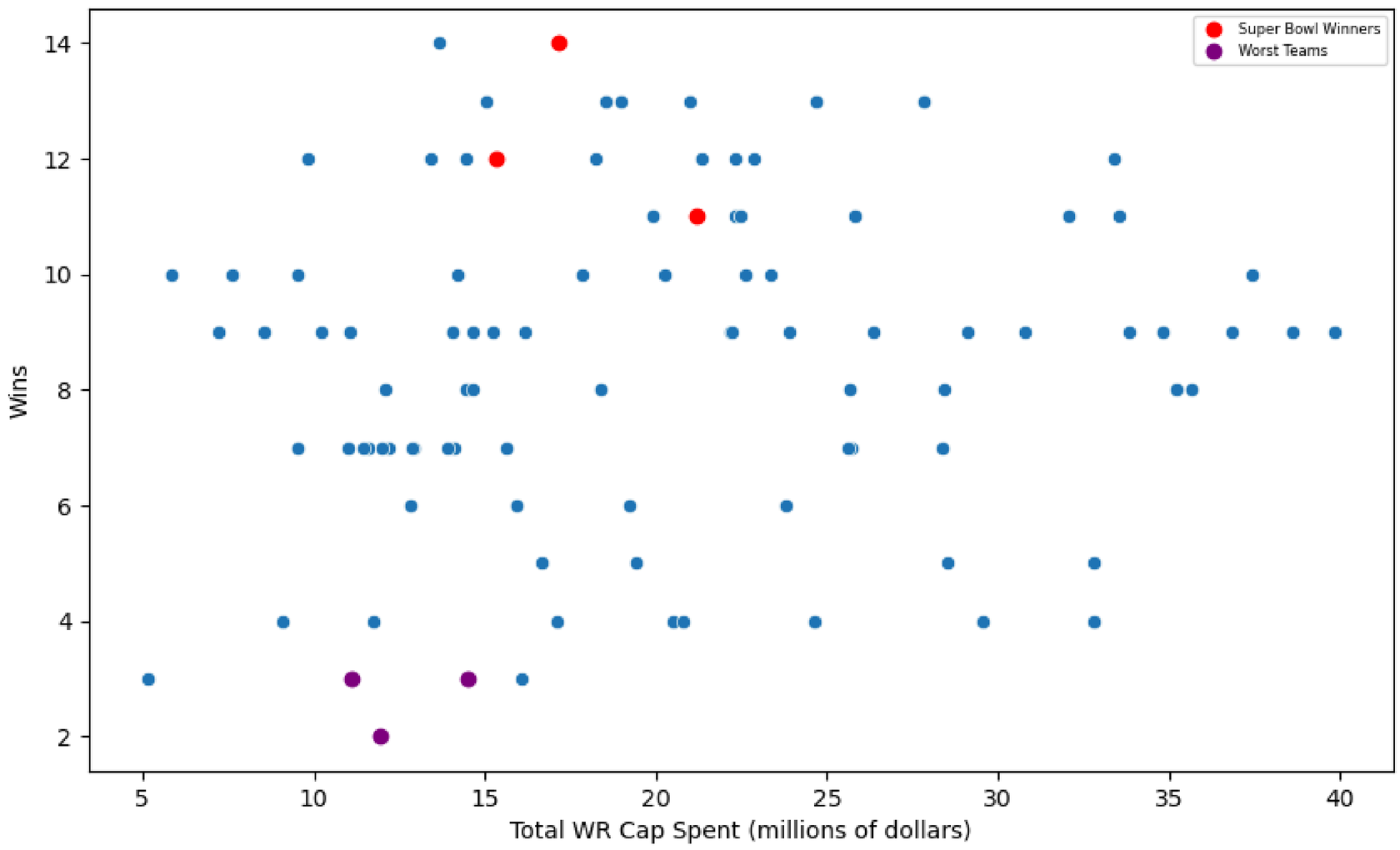
Next, I created 3 scatterplots. Wins vs. Total RB Cap Spent, Wins vs. Total QB Cap Spent, and Wins vs. Total WR Cap Spent. Also, on these plots I highlighted the 3 teams which won a superbowl and also the 3 teams that had the worst record for the year.

In [41]:
```
plt.figure(figsize = (10, 20));

plt.subplot(3, 1, 1);

#superbowl winners
sb_winner_2023_RB = RB_cap_wins_2023[RB_cap_wins_2023.team == "Chiefs"]
sb_winner_2022_RB = RB_cap_wins_2022[RB_cap_wins_2022.team == "Chiefs"]
sb_winner_2021_RB = RB_cap_wins_2021[RB_cap_wins_2021.team == "Rams"]

#worst teams (#1 draft pick)
worst_2023_RB = RB_cap_wins_2023[RB_cap_wins_2023.team == "Panthers"]
worst_2022_RB = RB_cap_wins_2022[RB_cap_wins_2022.team == "Bears"]
worst_2021_RB = RB_cap_wins_2021[RB_cap_wins_2021.team == "Jaguars"]

#scatter plot of wins vs QB_cap
sns.scatterplot(RB_cap_wins_total, x = "RB_cap", y = "wins");
plt.xlabel("Total RB Cap Spent(millions of dollars)");
plt.ylabel("Wins");

#highlighting super bowl champions
plt.scatter(sb_winner_2023_RB["RB_cap"].iloc[0], sb_winner_2023_RB["wins"].iloc[0], color = "red", label = "Super Bowl Winners");
plt.scatter(sb_winner_2022_RB["RB_cap"].iloc[0], sb_winner_2022_RB["wins"].iloc[0], color = "red");
plt.scatter(sb_winner_2021_RB["RB_cap"].iloc[0], sb_winner_2021_RB["wins"].iloc[0], color = "red");

#highlighting worst teams
plt.scatter(worst_2023_RB["RB_cap"].iloc[0], worst_2023_RB["wins"].iloc[0], color = "purple", label = "Worst Teams");
plt.scatter(worst_2022_RB["RB_cap"].iloc[0], worst_2022_RB["wins"].iloc[0], color = "purple");
plt.scatter(worst_2021_RB["RB_cap"].iloc[0], worst_2021_RB["wins"].iloc[0], color = "purple");

plt.legend(fontsize = 6);

plt.subplot(3, 1, 2);

#superbowl winners
sb_winner_2023_QB = QB_cap_wins_2023[QB_cap_wins_2023.team == "Chiefs"]
sb_winner_2022_QB = QB_cap_wins_2022[QB_cap_wins_2022.team == "Chiefs"]
sb_winner_2021_QB = QB_cap_wins_2021[QB_cap_wins_2021.team == "Rams"]

#worst teams (#1 draft pick)
worst_2023_QB = QB_cap_wins_2023[QB_cap_wins_2023.team == "Panthers"]
worst_2022_QB = QB_cap_wins_2022[QB_cap_wins_2022.team == "Bears"]
worst_2021_QB = QB_cap_wins_2021[QB_cap_wins_2021.team == "Jaguars"]

#scatter plot of wins vs QB_cap
sns.scatterplot(QB_cap_wins_total, x = "QB_cap", y = "wins");
plt.xlabel("Total QB Cap Spent (millions of dollars)");
plt.ylabel("Wins");

#highlighting super bowl champions
plt.scatter(sb_winner_2023_QB["QB_cap"].iloc[0], sb_winner_2023_QB["wins"].iloc[0], color = "red", label = "Super Bowl Winners");
plt.scatter(sb_winner_2022_QB["QB_cap"].iloc[0], sb_winner_2022_QB["wins"].iloc[0], color = "red");
plt.scatter(sb_winner_2021_QB["QB_cap"].iloc[0], sb_winner_2021_QB["wins"].iloc[0], color = "red");
```

```
#highlighting worst teams
plt.scatter(worst_2023_QB["QB_cap"].iloc[0], worst_2023_QB["wins"].iloc[0], color = "purple", label = "Worst Teams");
plt.scatter(worst_2022_QB["QB_cap"].iloc[0], worst_2022_QB["wins"].iloc[0], color = "purple");
plt.scatter(worst_2021_QB["QB_cap"].iloc[0], worst_2021_QB["wins"].iloc[0], color = "purple");

plt.legend(fontsize = 6);

plt.subplot(3, 1, 3);

#superbowl winners
sb_winner_2023_WR = WR_cap_wins_2023[WR_cap_wins_2023.team == "Chiefs"]
sb_winner_2022_WR = WR_cap_wins_2022[WR_cap_wins_2022.team == "Chiefs"]
sb_winner_2021_WR = WR_cap_wins_2021[WR_cap_wins_2021.team == "Rams"]

#worst teams (#1 draft pick)
worst_2023_WR = WR_cap_wins_2023[WR_cap_wins_2023.team == "Panthers"]
worst_2022_WR = WR_cap_wins_2022[WR_cap_wins_2022.team == "Bears"]
worst_2021_WR = WR_cap_wins_2021[WR_cap_wins_2021.team == "Jaguars"]

#scatter plot of wins vs WR_cap
sns.scatterplot(WR_cap_wins_total, x = "WR_cap", y = "wins");
plt.xlabel("Total WR Cap Spent (millions of dollars)");
plt.ylabel("Wins");

#highlighting super bowl champions
plt.scatter(sb_winner_2023_WR["WR_cap"].iloc[0], sb_winner_2023_WR["wins"].iloc[0], color = "red", label = "Super Bowl Winners");
plt.scatter(sb_winner_2022_WR["WR_cap"].iloc[0], sb_winner_2022_WR["wins"].iloc[0], color = "red");
plt.scatter(sb_winner_2021_WR["WR_cap"].iloc[0], sb_winner_2021_WR["wins"].iloc[0], color = "red");

#highlighting worst teams
plt.scatter(worst_2023_WR["WR_cap"].iloc[0], worst_2023_WR["wins"].iloc[0], color = "purple", label = "Worst Teams");
plt.scatter(worst_2022_WR["WR_cap"].iloc[0], worst_2022_WR["wins"].iloc[0], color = "purple");
plt.scatter(worst_2021_WR["WR_cap"].iloc[0], worst_2021_WR["wins"].iloc[0], color = "purple");

plt.legend(fontsize = 6);
```







Correlations

Below are the correlations for the three scatterplots displayed above.

In [78]:
```
#Correlations
print(f"Wins vs. RB Cap Spent Correlation: {statistics.correlation(RB_cap_wins_total.RB_cap, RB_cap_wins_total.wins)}")
```

```
print(f"Wins vs. QB Cap Spent Correlation: {statistics.correlation(QB_cap_wins_total.QB_cap, QB_cap_wins_total.wins)}")
print(f"Wins vs. WR Cap Spent Correlation: {statistics.correlation(WR_cap_wins_total.WR_cap, WR_cap_wins_total.wins)}")
```

```
Wins vs. RB Cap Spent Correlation: 0.12037918039648045
Wins vs. QB Cap Spent Correlation: 0.21731867163925164
Wins vs. WR Cap Spent Correlation: 0.11764651076260627
```

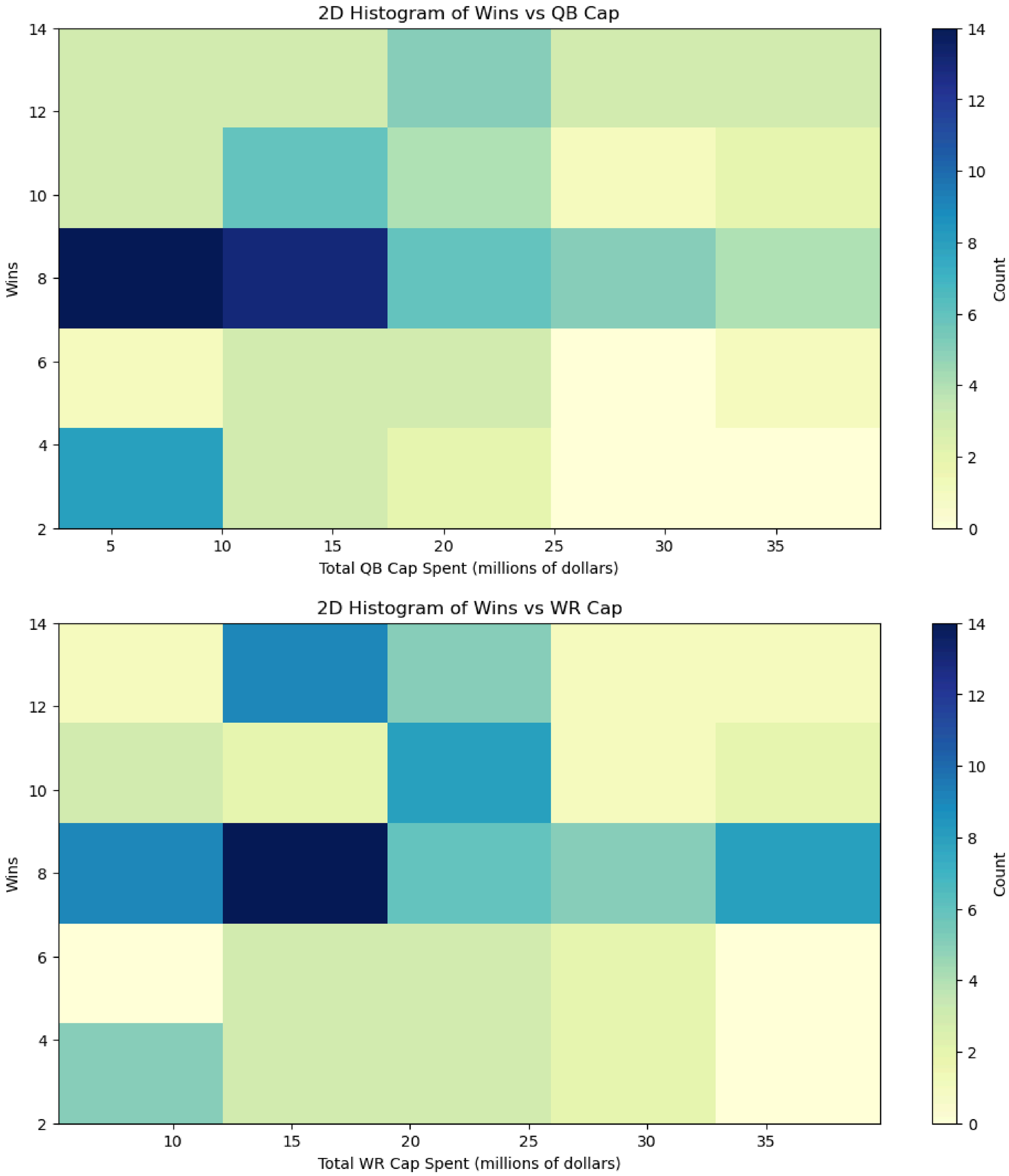
Analysis on Figures 2a, 2b, and 2c

Based on the scatterplots, one interesting trend that I found is that the worst teams invested very little in all three positions. As seen on the scatterplots, the purple points are all situated on the left side. As for the superbowl winning teams, they invested little in the runningback position, highly in the quarterback position, and in the middle for the receiver position. However, in looking at the correlations and the overall trend for the scatterplots, it seems like there is really no correlation between wins and the amount of cap spent on a position. The scatterplots appear very randomly distributed. Therefore, I did not think it was necessary to hypothesis test for a correlation.

On the other hand, in examining the scatterplots closer, I realized that a good amount of points for 10+ win teams fall on the left side of the Wins vs. RB Cap Spent scatterplot compared to the Wins vs. QB Cap Spent and Wins vs. WR Cap Spent Scatterplots. Therefore, I decided to examine the proportion of teams with 10+ wins who have a cap spent below the first quartile for each position. First, I decided to create 2d histograms of wins vs. cap spent at a position to see if this trend could be visualized better.

Figures 3a, 3b, 3c: 2D Histograms of Wins vs. Cap Spent at a Position

In [97]:
```
plt.figure(figsize = (12, 20));
plt.subplot(3, 1, 1)

hist1 = plt.hist2d(
    RB_cap_wins_total["RB_cap"],
    RB_cap_wins_total["wins"],
    bins=[5, 5],
    cmap="YlGnBu"
)
plt.colorbar(hist1[3], label="Count")
plt.xlabel("Total RB Cap Spent (millions of dollars)")
plt.ylabel("Wins")
plt.title("2D Histogram of Wins vs RB Cap")
plt.show()

plt.figure(figsize = (12, 20));
plt.subplot(3, 1, 2)

hist2 = plt.hist2d(
    QB_cap_wins_total["QB_cap"],
    QB_cap_wins_total["wins"],
    bins=[5, 5],
    cmap="YlGnBu"
)
plt.colorbar(hist2[3], label="Count")
plt.xlabel("Total QB Cap Spent (millions of dollars)")
plt.ylabel("Wins")
plt.title("2D Histogram of Wins vs QB Cap")
plt.show()

plt.figure(figsize = (12, 20));
plt.subplot(3, 1, 3)

hist3 = plt.hist2d(
    WR_cap_wins_total["WR_cap"],
    WR_cap_wins_total["wins"],
    bins=[5, 5],
    cmap="YlGnBu"
)
plt.colorbar(hist3[3], label="Count")
plt.xlabel("Total WR Cap Spent (millions of dollars)")
plt.ylabel("Wins")
plt.title("2D Histogram of Wins vs WR Cap")
plt.show()
```



Analysis of Figures 3a, 3b, and 3c

From looking at the 2d histograms, it appears as if are more teams who spent little on the runningback position and also had a higher amount of wins than those teams who spent little on the quarterback and receiver positons and also had a higher amount of wins. From looking at the 2d histograms, the top left corner of the 2d histogram of Wins vs. RB cap appear to be more blue than the other two histograms. Furthermore, each histogram is on a slightly different color scale, whereas for the runningback histogram, it is on a larger scale where there has to be a higher count to be more blue. Therefore, since this runningback histogram is on a different scale and it still appears more blue in the top left, it appears as if there is a higher proportion of 10+ win teams who spent below the first quartile for cap spent for runningbacks. To test this in a way to get more concrete evidence, I decided to run a statistical test.

Hypothesis Testing: Is the proportion of 10+ Win Teams Below the First Quartile Cap Spent for a position higher for runningbacks than quarterbacks and receivers?

To do these hypotheses test, I first need to calculate the first quartiles of cap spent on each position and the proportion of 10+ win teams below this first quartile for each position.

Calculate First Quartiles

In [99]:
```
RB_cap_first = np.percentile(RB_cap_wins_total["RB_cap"], 25)

QB_cap_first = np.percentile(QB_cap_wins_total["QB_cap"], 25)

WR_cap_first = np.percentile(WR_cap_wins_total["WR_cap"], 25)
```

Find Proportions of 10+ Win Teams Below First Quartile Cap Spent For Each Position

In [102…
```
RB_below_first = RB_cap_wins_total[RB_cap_wins_total.RB_cap < RB_cap_first]
RB_below_prop = np.mean(RB_below_first.wins >= 10)


QB_below_first = QB_cap_wins_total[QB_cap_wins_total.QB_cap < QB_cap_first]
QB_below_prop = np.mean(QB_below_first.wins >= 10)
```

```
WR_below_first = WR_cap_wins_total[WR_cap_wins_total.WR_cap < WR_cap_first]
WR_below_prop = np.mean(WR_below_first.wins >= 10)

print(f"Proportion of 10+ Win Teams Below First Quartile for RBs: {RB_below_prop}")
print(f"Proportion of 10+ Win Teams Below First Quartile for QBs: {QB_below_prop}")
print(f"Proportion of 10+ Win Teams Below First Quartile for WRs: {WR_below_prop}")
```

```
Proportion of 10+ Win Teams Below First Quartile for RBs: 0.25
Proportion of 10+ Win Teams Below First Quartile for QBs: 0.16666666666666666
Proportion of 10+ Win Teams Below First Quartile for WRs: 0.20833333333333334
```

Hypothesis Testing for a Difference in Proportions Between RBs and QBs

Null and Alternate Hypotheses

H_0: $\pi_{RB} = \pi_{QB}$ or H_0: $\pi_{RB} - \pi_{QB} = 0$

H_A: $\pi_{RB} > \pi_{QB}$ or H_0: $\pi_{RB} - \pi_{QB} > 0$

Observed Statistic

In [104…
```
obs_stat = RB_below_prop - QB_below_prop
obs_stat
```

Out[104… `0.08333333333333334`

Null Distribution

Create a DataFrame that can be used for creating null distribution

In [145…
```
RB_below_prop_series = RB_below_first.wins >= 10
RB_below_prop_df = RB_below_prop_series.to_frame()
RB_below_prop_df["position"] = "RB"

QB_below_prop_series = QB_below_first.wins >= 10
QB_below_prop_df = QB_below_prop_series.to_frame()
QB_below_prop_df["position"] = "QB"

QB_RB_below_prop_df = pd.concat([RB_below_prop_df, QB_below_prop_df])
```

Create a function to calculate difference in proportions

In [150…
```
def get_prop_diff_QB(df_QB_RB):
    QB_RB_means = df_QB_RB.groupby("position").mean()
    the_difference = QB_RB_means.loc["RB"] - QB_RB_means.loc["QB"]
    return the_difference.iloc[0]
```

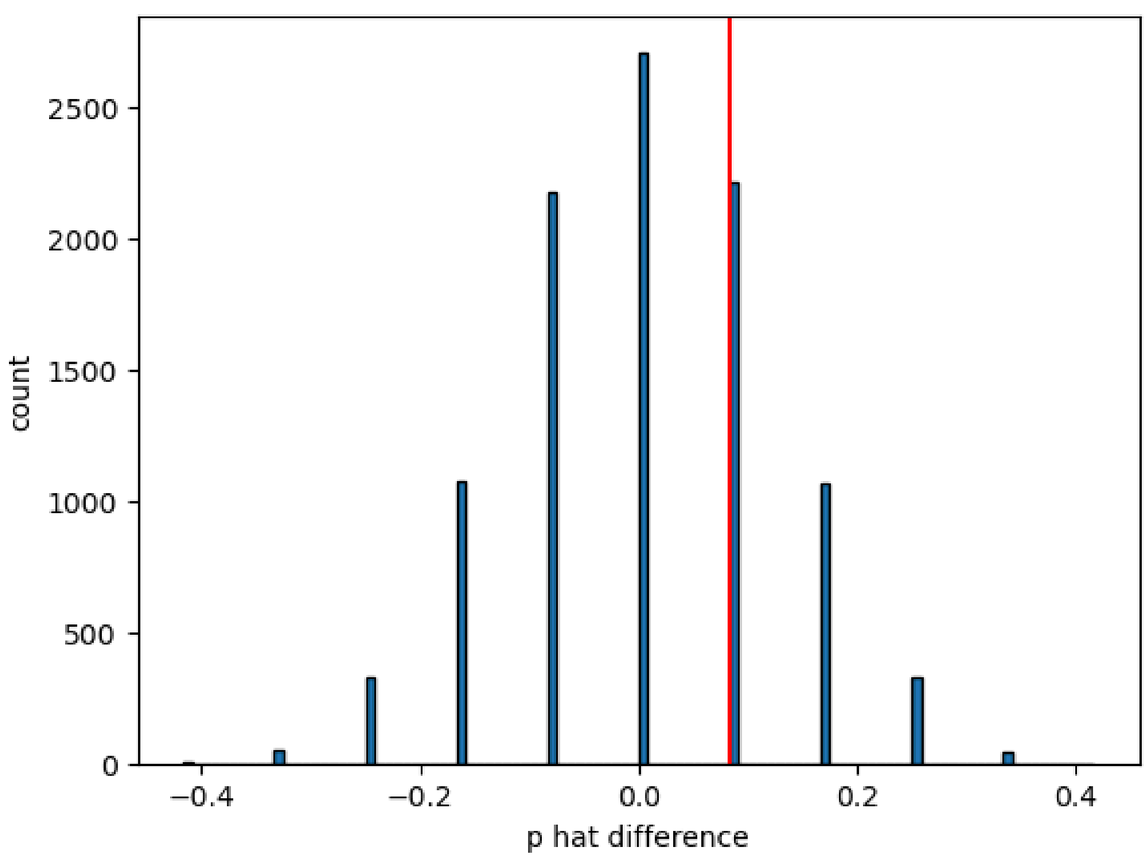
Create null

In [135…
```
%%time
shuff_QB_RB = QB_RB_below_prop_df.copy()
null_dist = []

for i in range(10000):
    shuff_QB_RB['position'] = np.random.permutation(QB_RB_below_prop_df.position)
    shuff_stat = get_prop_diff_QB(shuff_QB_RB)
    null_dist.append(shuff_stat)
```

```
CPU times: user 7.82 s, sys: 6.06 ms, total: 7.83 s
Wall time: 7.96 s
```

Visualize null

In [140…
```
plt.hist(null_dist, edgecolor = "black", bins = 100);
plt.axvline(obs_stat, color = "red");
plt.xlabel("p hat difference");
plt.ylabel("count");
```



Calculate p-value

In [142…
```
pval = np.mean(np.array(null_dist) >= obs_stat)
pval
```

Out[142… 0.3654

Conclusion of Hypothesis Test for A Difference in Proportions Between RBs and QBs

Since the p-value of 0.3654 is greater than the typical significance level of 0.05, I fail to reject that there is not a difference in the proportion of 10+ wins team who spend below the first quartile on runningbacks and the proportion of 10+ wins teams who spend below the first quartile for quarterbacks.

Hypothesis Testing for a Difference in Proportions Between RBs and WRs

Null and Alternate Hypotheses

H_0: $\pi_{RB} = \pi_{WR}$ or H_0: $\pi_{RB} - \pi_{WR} = 0$

H_A: $\pi_{RB} > \pi_{WR}$ or H_0: $\pi_{RB} - \pi_{WR} > 0$

Observed Statistic

In [156…
```
obs_stat_2 = RB_below_prop - WR_below_prop
obs_stat_2
```

Out[156… 0.04166666666666666

Null Distribution

Create a DataFrame that can be used for creating null distribution

In [148…
```
WR_below_prop_series = WR_below_first.wins >= 10
WR_below_prop_df = WR_below_prop_series.to_frame()
WR_below_prop_df["position"] = "WR"

RB_WR_below_prop_df = pd.concat([RB_below_prop_df, WR_below_prop_df])
```

Create a function to calculate difference in proportions

In [154…
```
def get_prop_diff_WR(df_RB_WR):
    RB_WR_means = df_RB_WR.groupby("position").mean()
    the_difference = RB_WR_means.loc["RB"] - RB_WR_means.loc["WR"]
    return the_difference.iloc[0]
```

Create Null

In [155…
```
%%time
shuff_RB_WR = RB_WR_below_prop_df.copy()
null_dist_2 = []

for i in range(10000):
    shuff_RB_WR['position'] = np.random.permutation(RB_WR_below_prop_df.position)
    shuff_stat = get_prop_diff_WR(shuff_RB_WR)
    null_dist_2.append(shuff_stat)
```

```
CPU times: user 7.9 s, sys: 12.9 ms, total: 7.92 s
Wall time: 8.05 s
```

Visualize Null

In [157…
```
plt.hist(null_dist_2, edgecolor = "black", bins = 100);
plt.axvline(obs_stat_2, color = "red");
plt.xlabel("p hat difference");
plt.ylabel("count");
```

Calculate p-value

In [158…
```
pval2 = np.mean(np.array(null_dist_2) >= obs_stat_2)
pval2
```

Out[158… 0.496

Conclusion of Hypothesis Test for A Difference in Proportions Between RBs and WRs

Since the p-value of 0.496 is greater than the typical significance level of 0.05, I fail to reject that there is not a difference in the proportion of 10+ wins team who spend below the first quartile on runningbacks and the proportion of 10+ wins teams who spend below the first quartile for receivers.

Final Conclusions

Ultimately, based off this project alone, I can not determine whether NFL teams should invest in the runningback position. However, I believe this project does provide some good direction for further analysis into this question.

For part 1, I do think that the difference in skill gap within quarterbacks, runningbacks, and receivers does provide some statistical basis for investing less in the runningback position. As demonstrated by the percent differences, for the 2023 season, there was less of a statistical difference between the production of top runningbacks vs the production of bottom runningbacks compared to the production of top and bottom quarterbacks and receivers. Therefore, having a bad runningback will likely provide more production for an NFL team compared to having a bad quarterback or receiver. However, there may be some limitations with how strong of evidence this data actually is. After all, this data only analyzes one season and one statistic, yards. Analyzing more than one season and more than one statistic would build stronger evidence for this data if the trend were to continue.

For part 2, one key takeaway was the difference in spending between superbowl winning teams and the worst teams. The worst teams spent very low for all three categories, whereas the super bowl winning team appeared to spend the most on quarterback and less on runningbacks and receivers. Although this trend is reserved to three seasons, it would be interesting to examine this trend over more seasons. Also, instead of just the best and worst team, maybe analyze the spending of the top 5 and bottom 5 teams.

For the hypothesis tests, although there was not enough statistical evidence to support a difference between the proportion of 10+ win teams who spend below the 1st quartile for runningbacks and the proportion of 10+ win teams who spend below the 1st quartile for quarterbacks or receivers, it is still possible that there could be a difference. For the three seasons, there was a higher proportion of 10+ win teams who spent below the 1st quartile for runningbacks compared to quarterbacks and receivers. Perhaps with more data from more seasons, there could be a more significant difference.

Nonetheless, based on this project, the spending of an NFL team does not seem to directly correlate to wins. Just by looking at the scatterplots of wins vs. spending for different positions, the scatterplots do not really show a pattern and appear randomly dispersed. Therefore, there appears to be a number of factors that affect how well a team does and cannot just simply be broken down into how much they spend on certain positions. For example, if a team spends a lot on a certain position, it does not guarantee production from that position, as that player could get hurt or simply fail to produce.

Reflection

I think the toughest part of this project was the data wrangling. I spent a very good chunk of time just getting the data into a form that I could do data analysis on. Something that I think went good on this project was that I was able to use a good variety of different data analysis techniques that we have done in class. Although, reflecting back on this project, there are numerous other analyses that I wish I could have done but due to a length constraint and the large amount of data that I would have had to gather, I decided to limit some of the analyses. For example, instead of analyzing the last 10 seasons, I only analyzed the last 3 seasons. Overall, I spent around 15-20 hours on this project.

Appendix

Next Gen Code

Below is the rest of the code for the data wrangling for part 1 to get the top and bottom rushers and passers.

In [30]:
```
#rushing
team_best_rushing = ngs_2023_rushing_rs.groupby("team_abbr").agg(
    leading_rusher = ("rush_yards", "max")
)
team_best_rushing = team_best_rushing.sort_values("leading_rusher")
#bottom 5 starting rushers
bottom_rushers = team_best_rushing[0:5]
#top 5 starting rushers
top_rushers = team_best_rushing[27:32]
#total yards of bottom rushers and total yards of top rushers
total_bottom_rushers = np.sum(bottom_rushers["leading_rusher"])
total_top_rushers = np.sum(top_rushers["leading_rusher"])

#passing
team_best_passers = ngs_2023_passing_rs.groupby("team_abbr").agg(
    leading_passer = ("pass_yards", "max")
)
team_best_passers = team_best_passers.sort_values("leading_passer")
team_best_passers
#bottom 5 starting rushers
bottom_passers = team_best_passers[0:5]
#top 5 starting rushers
top_passers = team_best_passers[27:32]
#total yards of bottom rushers and total yards of top rushers
total_bottom_passers = np.sum(bottom_passers["leading_passer"])
total_top_passers = np.sum(top_passers["leading_passer"])
```

Importing Contracts for Each Position

Below is the full code for preparing the DataFrames containing the cap spent on a position from each season

In [40]:
```
#Reading in Quarterback Contracts DataFrame
QB_contracts_2023 = pd.read_csv("QB_contracts.csv")

#Remove $ and , from cap column
QB_contracts_2023["cap"] = QB_contracts_2023["cap"].str.replace("$", "")
QB_contracts_2023["cap"] = QB_contracts_2023["cap"].str.replace(",", "")

#convert column to numbers
QB_contracts_2023["cap"] = pd.to_numeric(QB_contracts_2023["cap"])

#get total qb cap for each team
```

```
QB_caps_2023 = QB_contracts_2023.groupby("team").agg(QB_cap = ("cap", "sum"))
QB_caps_2023 = QB_caps_2023.reset_index()
QB_caps_2023.head(2)
```

Out[40]:

	team	QB_cap
0	49ers	8281753
1	Bears	6584118

In [14]:
```
#Reading in Receiver Contracts DataFrame
WR_contracts_2023 = pd.read_csv("WR_contracts.csv")

#Remove $ and , from cap column
WR_contracts_2023["cap"] = WR_contracts_2023["cap"].str.replace("$", "")
WR_contracts_2023["cap"] = WR_contracts_2023["cap"].str.replace(",", "")

#convert column to numbers
WR_contracts_2023["cap"] = pd.to_numeric(WR_contracts_2023["cap"])

#get total WR cap for each team
WR_caps_2023 = WR_contracts_2023.groupby("team").agg(WR_cap = ("cap", "sum"))
WR_caps_2023 = WR_caps_2023.reset_index()
WR_caps_2023.head(2)
```

Out[14]:

	team	WR_cap
0	49ers	18223006
1	Bears	28376015

In [15]:
```
#Reading in DataFrame
RB_contracts_2022 = pd.read_csv("RB_contracts_2022.csv")

#Remove Extra Empty Columns
RB_contracts_2022 = RB_contracts_2022[["player", "team", "cap"]]

#Remove $ and , from cap column
RB_contracts_2022["cap"] = RB_contracts_2022["cap"].str.replace("$", "")
RB_contracts_2022["cap"] = RB_contracts_2022["cap"].str.replace(",", "")

#convert column to numbers
RB_contracts_2022["cap"] = pd.to_numeric(RB_contracts_2022["cap"])

#get total rb cap for each team
RB_caps_2022 = RB_contracts_2022.groupby("team").agg(RB_cap = ("cap", "sum"))
RB_caps_2022 = RB_caps_2022.reset_index()
RB_caps_2022.head(2)
```

Out[15]:

	team	RB_cap
0	49ers	3134553
1	Bears	4900984

In [16]:
```
#Reading in Quarterback Contracts DataFrame
QB_contracts_2022 = pd.read_csv("QB_contracts_2022.csv")

#Remove $ and , from cap column
QB_contracts_2022["cap"] = QB_contracts_2022["cap"].str.replace("$", "")
QB_contracts_2022["cap"] = QB_contracts_2022["cap"].str.replace(",", "")

#convert column to numbers
QB_contracts_2022["cap"] = pd.to_numeric(QB_contracts_2022["cap"])

#get total qb cap for each team
QB_caps_2022 = QB_contracts_2022.groupby("team").agg(QB_cap = ("cap", "sum"))
QB_caps_2022 = QB_caps_2022.reset_index()
QB_caps_2022.head(2)
```

Out[16]:

	team	QB_cap
0	49ers	22715235
1	Bears	6917413

In [17]:
```
#Reading in Receiver Contracts DataFrame
WR_contracts_2022 = pd.read_csv("WR_contracts_2022.csv")

#Remove $ and , from cap column
WR_contracts_2022["cap"] = WR_contracts_2022["cap"].str.replace("$", "")
WR_contracts_2022["cap"] = WR_contracts_2022["cap"].str.replace(",", "")

#convert column to numbers
WR_contracts_2022["cap"] = pd.to_numeric(WR_contracts_2022["cap"])

#get total WR cap for each team
WR_caps_2022 = WR_contracts_2022.groupby("team").agg(WR_cap = ("cap", "sum"))
WR_caps_2022 = WR_caps_2022.reset_index()
WR_caps_2022.head(2)
```

Out[17]:

	team	WR_cap
0	49ers	15039393
1	Bears	11115639

In [18]:
```
#Reading in DataFrame
RB_contracts_2021 = pd.read_csv("RB_contracts_2021.csv")
```

```
#Remove Extra Empty Columns
RB_contracts_2021 = RB_contracts_2021[["player", "team", "cap"]]

#Remove $ and , from cap column
RB_contracts_2021["cap"] = RB_contracts_2021["cap"].str.replace("$", "")
RB_contracts_2021["cap"] = RB_contracts_2021["cap"].str.replace(",", "")

#convert column to numbers
RB_contracts_2021["cap"] = pd.to_numeric(RB_contracts_2021["cap"])

#get total rb cap for each team
RB_caps_2021 = RB_contracts_2021.groupby("team").agg(RB_cap = ("cap", "sum"))
RB_caps_2021 = RB_caps_2021.reset_index()
RB_caps_2021.head(2)
```

Out[18]:

	team	RB_cap
0	49ers	8755882
1	Bears	6793142

In [19]:
```
#Reading in Quarterback Contracts DataFrame
QB_contracts_2021 = pd.read_csv("QB_contracts_2021.csv")

#Remove $ and , from cap column
QB_contracts_2021["cap"] = QB_contracts_2021["cap"].str.replace("$", "")
QB_contracts_2021["cap"] = QB_contracts_2021["cap"].str.replace(",", "")

#convert column to numbers
QB_contracts_2021["cap"] = pd.to_numeric(QB_contracts_2021["cap"])

#get total qb cap for each team
QB_caps_2021 = QB_contracts_2021.groupby("team").agg(QB_cap = ("cap", "sum"))
QB_caps_2021 = QB_caps_2021.reset_index()
QB_caps_2021.head(2)
```

Out[19]:

	team	QB_cap
0	49ers	33349156
1	Bears	15162197

In [20]:
```
#Reading in Receiver Contracts DataFrame
WR_contracts_2021 = pd.read_csv("WR_contracts_2021.csv")

#Remove $ and , from cap column
WR_contracts_2021["cap"] = WR_contracts_2021["cap"].str.replace("$", "")
WR_contracts_2021["cap"] = WR_contracts_2021["cap"].str.replace(",", "")

#convert column to numbers
WR_contracts_2021["cap"] = pd.to_numeric(WR_contracts_2021["cap"])

#get total WR cap for each team
WR_caps_2021 = WR_contracts_2021.groupby("team").agg(WR_cap = ("cap", "sum"))
WR_caps_2021 = WR_caps_2021.reset_index()
WR_caps_2021.head(2)
```

Out[20]:

	team	WR_cap
0	49ers	9519367
1	Bears	23816178

Importing Regular Season Wins for Each Team

Below is the rest of the code for importing the regular season wins for each team from 2021 and 2022.

In [24]:
```
team_wins_rs_2022 = pd.read_csv("team_wins_rs_2022.csv")
#remove extra blank columns and rows
team_wins_rs_2022 = team_wins_rs_2022[["team", "wins"]]
team_wins_rs_2022.dropna(axis = 0, inplace = True)
team_wins_rs_2022.head(2)
```

Out[24]:

	team	wins
0	Bills	13
1	Dolphins	9

In [26]:
```
team_wins_rs_2021 = pd.read_csv("team_wins_rs_2021.csv")
#remove extra blank columns and rows
team_wins_rs_2021 = team_wins_rs_2021[["team", "wins"]]
team_wins_rs_2021.dropna(axis = 0, inplace = True)
team_wins_rs_2021.head(2)
```

Out[26]:

	team	wins
0	Bills	11
1	Patriots	10

Merging Positional Cap DataFrames with Regular Season Wins DataFrames

Below is the rest of the code for merging the positional cap DataFrames with the Regular Season Wins DataFrames

In [32]:
```
#joining qb and wins
QB_cap_wins_2023 = QB_caps_2023.merge(team_wins_rs_2023, how = "left")
QB_cap_wins_2023["QB_cap"] = QB_cap_wins_2023["QB_cap"]/1000000
QB_cap_wins_2023.head(2)
```

Out[32]:

	team	QB_cap	wins
0	49ers	8.281753	12.0
1	Bears	6.584118	7.0

In [33]:
```
#joining wr and wins
WR_cap_wins_2023 = WR_caps_2023.merge(team_wins_rs_2023, how = "left")
WR_cap_wins_2023["WR_cap"] = WR_cap_wins_2023["WR_cap"]/1000000
WR_cap_wins_2023.head(2)
```

Out[33]:

	team	WR_cap	wins
0	49ers	18.223006	12.0
1	Bears	28.376015	7.0

In [34]:
```
#joining rb and wins
RB_cap_wins_2022 = RB_caps_2022.merge(team_wins_rs_2022, how = "left")
RB_cap_wins_2022["RB_cap"] = RB_cap_wins_2022["RB_cap"]/1000000
RB_cap_wins_2022.head(2)
```

Out[34]:

	team	RB_cap	wins
0	49ers	3.134553	13
1	Bears	4.900984	3

In [35]:
```
#joining qb and wins
QB_cap_wins_2022 = QB_caps_2022.merge(team_wins_rs_2022, how = "left")
QB_cap_wins_2022["QB_cap"] = QB_cap_wins_2022["QB_cap"]/1000000
QB_cap_wins_2022.head(2)
```

Out[35]:

	team	QB_cap	wins
0	49ers	22.715235	13
1	Bears	6.917413	3

In [36]:
```
#joining wr and wins
WR_cap_wins_2022 = WR_caps_2022.merge(team_wins_rs_2022, how = "left")
WR_cap_wins_2022["WR_cap"] = WR_cap_wins_2022["WR_cap"]/1000000
WR_cap_wins_2022.head(2)
```

Out[36]:

	team	WR_cap	wins
0	49ers	15.039393	13
1	Bears	11.115639	3

In [37]:
```
#joining rb and wins
RB_cap_wins_2021 = RB_caps_2021.merge(team_wins_rs_2021, how = "left")
RB_cap_wins_2021["RB_cap"] = RB_cap_wins_2021["RB_cap"]/1000000
RB_cap_wins_2021.head(2)
```

Out[37]:

	team	RB_cap	wins
0	49ers	8.755882	10
1	Bears	6.793142	6

In [38]:
```
#joining qb and wins
QB_cap_wins_2021 = QB_caps_2021.merge(team_wins_rs_2021, how = "left")
QB_cap_wins_2021["QB_cap"] = QB_cap_wins_2021["QB_cap"]/1000000
QB_cap_wins_2021.head(2)
```

Out[38]:

	team	QB_cap	wins
0	49ers	33.349156	10
1	Bears	15.162197	6

In [39]:
```
#joining wr and wins
WR_cap_wins_2021 = WR_caps_2021.merge(team_wins_rs_2021, how = "left")
WR_cap_wins_2021["WR_cap"] = WR_cap_wins_2021["WR_cap"]/1000000
WR_cap_wins_2021.head(2)
```

Out[39]:

	team	WR_cap	wins
0	49ers	9.519367	10
1	Bears	23.816178	6

Project Reviews

Review 1

Project writer name: Kyle Levesque

Project writer email address: kyle.levesque@yale.edu

Reviewer name: Samuel Rosenberg
Reviwer email address: s.rosenberg@yale.edu

Summary

The project analyzes data from past NFL games to draw conclusions about the utility and value of the runningback position in American Football. The project analyzes the skill gap between the runningback and other positions, and compares the performance of football teams with the annual salaries of their runningbacks. The project concluded that while the runningback position is less important than other positions like the quarterback, the spending of an NFL team does not directly correlate to its success.

1. Overall strengths and weaknesses

The project clearly states the questions it is trying to answer. It explains the code cleary and succiently, and the graphs are nice to look at and easy to understand. However, the project can be somewhat unapproachable for someone not familiar with American Football.

2. Major Revisions

- N/A

3. Minor Revisions

- Add clearer explanations for certain terminology and roles specific to American Football.

3. Rubric Score

- Introduction: 15 points
- Data Cleaning: 12 points
- Data visualization: 23 points
- Analyses: 23 points
- Conclusions: 12 points
- Review & Reflection: 12 points

Rubric items where points would be taken off if not addressed:

- N/A

Total Score: 90/90

Review 2

Project writer name: Kyle Levesque
Project writer email address: kyle.levesque@yale.edu
Reviewer name: Jason Abohwo
Reviwer email address: Jason.Abohwo@yale.edu

Summary

This project aims to determine whether the devaluation of the running back position has a statistical basis (in terms of performance metrics). By comparing the skill gaps between player, this project finds that running backs exhibit the smallest disparity in performance (which indicates that having a less skilled running back is not a large detriment to overall team performance). Additionally, analyses revealed that there is no significant correlation between running back investment and team wins. On a whole, the findings in the study support the notion that the running back position is not of the same value as other positions (though, the study does have and acknowledge several limitations).

1. Overall strengths and weaknesses

With respect to strengths, I would say that the diversity of sources helps to create a well-rounded analysis (because it mitgates the bias the leaks into the analysis from a singular source). Personally, I found the statistical test and the visualizations that accompanied it to be the most convincing, as it was (1) easy to understand, and (2) accounted for things like statistical variance (as statistical tests do this innately). In terms of weaknesses, I would assert that the structure of the project is somewhat confusing. Having specific for Data Preprocessing, Visualizations, and Analyses may make the boundry lines between which is which more clear.

2. Major revisions

- Consider modifying figures 2a, 2b, and 2c to be of a different type (i.e. not scatterplot). I found it difficult to understand what trends were present in the data at first glance; changing the visualization type to a type better suited for the given data may lead to easier interpretability

3. Minor Revisions

- Given that you have 2 extra pages of space, I would suggest deepening your analyses and visualizations.
- Consider fixing grammatical and typographical errors (e.g. consistent when using runningback vs running back)
- Consider using different styles to improve visuals

3. Rubric score

- Introduction: 15 / 15

- Data Cleaning: 10 / 12
- Data Visualization: 20 / 23
- Analyses: 23 / 23
- Conclusion: 12 / 12
- Review and reflection: 12 / 12

Rubric items where points would be taken off if not addressed:

- Data Cleaning: Within the first 10 pages of the project, address the data cleaning (data wrangling, outlier removal, etc), that you did in order to make the data usable for the purposes of your visualizations and analyses
- Data Visualization: Improve visualization interpretability as well as visualization aesthetics

Total score: 92/97

Review 3

Never received a third review