Introduction

This is an end-of-phase project. In this phase, my cohort and I took a deep dive into inferential analysis. Using what I learned in this phase along with what I learned in previous phases, I did a decriptive and inferential project on NHL team stats over the past 3 seasons

Descriptive Analysis

In my descriptive Analysis section I plan to answer at least 5 descriptive question about the data

Inferential Analysis

In my inferential analysis ssection I plan to propose at least 3 hypothesis and find conclusions for each

Table of Contents

```
import nbformat

def generate_toc(notebook_path):
    with open(notebook_path) as f:
        nb = nbformat.read(f, as_version=4)

    toc = []
    for cell in nb.cells:
        if cell.cell_type == 'markdown':
            lines = cell.source.split('\n')
            for line in lines:
                if line.startswith('#'):
                    header_level = line.count('#')
                    header_text = line.replace('#', '').strip()
                    toc.append((header_level, header_text))

    toc_md = ['## Table of Contents']
    for level, text in toc:
        toc_md.append(f"{'    ' * (level - 1)}- [{text}](#{text.replace(' ', '-')})")

    return '\n'.join(toc_md)

notebook_path = 'Phase 2 project.ipynb'
toc_md = generate_toc(notebook_path)

# Print the generated TOC
print(toc_md)
```

```
## Table of Contents
- [Introduction](#Introduction)
    - [Descriptive Analysis](#Descriptive-Analysis)
    - [Inferential Analysis](#Inferential-Analysis)
    - [Table of Contents](#Table-of-Contents)
    - [Imports, Data Cleaning and Dataframe Prep](#Imports,-Data-Cleaning-and-Dataframe-Prep)
    - [Descriptive Questions](#Descriptive-Questions)
        - [Descriptive Question 1: Over the Past 3 Seasons, Which Division attributed for the most amount of goals?](#Descriptive-Questi
        - [Descriptive Question 2: Which Division attributed for the most amount of points in the past 3 seasons?](#Descriptive-Question
        - [Descriptive Question 3: Which Team accumulated the most amount of points over the past 3 season?](#Descriptive-Question-3:-Wh
        - [Descriptive Question 4: Which Team scored the most amount of Goals over the past 3 seasons?](#Descriptive-Question-4:-Which-T
        - [Descriptive Question 5: Which Team gave up the **MOST** amount of Goals over the past 3 seasons?](#Descriptive-Question-5:-Wh
        - [Descriptive Question 6: Which Team gave up the **LEAST** amount of Goals over the past 3 seasons?](#Descriptive-Question-6:-W
        - [Descriptive Question 7: What is the relation between Goals For and Goals Against for each team over the past 3 seasons?](#Des
        - [Descriptive Question 8+ : I wanted to see the relation between Goals For and Different Shot Types, so I plotted them in a sca
    - [Inferential Questions](#Inferential-Questions)
        - [Inferential Question 1: Does a change in Division have any relation to a change in average points at the end of the season?](
            - [Null Hypothesis ($H_0$): $\mu_{ATL}$ = $\mu_{MET}$ = $\mu_{CEN}$ = $\mu_{PAC}$](#Null-Hypothesis-($H_0$):-$\mu_{ATL}$-=-$
            - [Alternate Hypothesis ($H_A$): $\mu_{ATL}$ $\neq$ $\mu_{MET}$ $\neq$ $\mu_{CEN}$ $\neq$ $\mu_{PAC}$](#Alternate-Hypothesis
        - [Inferential Question 2: Does a change in Conference have any relation to a change in average points at the end of the season?
            - [Null Hypothesis ($H_0$): $\mu_{EAST}$ = $\mu_{WEST}$](#Null-Hypothesis-($H_0$):-$\mu_{EAST}$-=-$\mu_{WEST}$)
            - [Alternate Hypothesis ($H_A$): $\mu_{EAST}$ $\neq$ $\mu_{WEST}$](#Alternate-Hypothesis-($H_A$):-$\mu_{EAST}$-$\neq$-$\mu_{
        - [Inferential Question 3: Does a change in a Team's Power Play percentage have a realtion with a Team's place in the standings
            - [Null Hypothesis ($H_0$): $\mu_{PP\%>20}$ = $\mu_{PP\%<20}$](#Null-Hypothesis-($H_0$):-$\mu_{PP\%>20}$-=-$\mu_{PP\%<20}$)
            - [Alternate Hypothesis ($H_A$): $\mu_{PP\%>20}$ $\neq$ $\mu_{PP\%<20}$](#Alternate-Hypothesis-($H_A$):-$\mu_{PP\%>20}$-$\ne
        - [Inferential Question 4: Does a change in a Team's Penalty Kill percentage have a relation  a Team's place in the standings at
            - [Null Hypothesis ($H_0$): $\mu_{PK\%>80}$ = $\mu_{PK\%<80}$](#Null-Hypothesis-($H_0$):-$\mu_{PK\%>80}$-=-$\mu_{PK\%<80}$)
            - [Alternate Hypothesis ($H_A$): $\mu_{PK\%>80}$ $\neq$ $\mu_{PK\%<80}$](#Alternate-Hypothesis-($H_A$):-$\mu_{PK\%>80}$-$\ne
        - [Inferential Question 5: Does a change in a  Team's overall Special Teams quality have a relation with a Team's place in the s
            - [Null Hypothesis ($H_0$): $\mu_{SP>100}$ = $\mu_{SP<100}$](#Null-Hypothesis-($H_0$):-$\mu_{SP>100}$-=-$\mu_{SP<100}$)
            - [Alternate Hypothesis ($H_A$): $\mu_{SP>100}$ $\neq$ $\mu_{SP<100}$](#Alternate-Hypothesis-($H_A$):-$\mu_{SP>100}$-$\neq$-
        - [Inferential Question 6+ : There are a number of percentage stats that are calculated using (`Stat` For)/ (`Stat` For + `Stat`
            - [Null Hypothesis ($H_0$): $\mu_{`Stat`\%>50}$ = $\mu_{`Stat`\%<50}$](#Null-Hypothesis-($H_0$):-$\mu_{`Stat`\%>50}$-=-$\mu_
            - [Alternate Hypothesis ($H_A$): $\mu_{`Stat`\%>50}$ $\neq$ $\mu_{`Stat`\%<50}$](#Alternate-Hypothesis-($H_A$):-$\mu_{`Stat`
            - [Expected Goals](#Expected-Goals)
```

Imports, Data Cleaning and Dataframe Prep

Before getting started, I need to import various packages, upload data and clean my dataframe to maek it ready for my questions and hypotheses

```
#Imports
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from scipy import stats
from PIL import Image
import matplotlib.image as mpimg
from matplotlib.offsetbox import OffsetImage, AnnotationBbox
import os
sns.set_style('whitegrid')
```

```
# Uploading csvs into pandas dfs
mp2021 = pd.read_csv('teams2021-2022.csv')
nhl2021 = pd.read_csv('Summary2021-2022.csv', encoding='latin-1')
mp2022 = pd.read_csv('teams2022-2023.csv')
nhl2022 = pd.read_csv('Summary2022-2023.csv', encoding='latin-1')
mp2023 = pd.read_csv('teams2023-2024.csv')
nhl2023 = pd.read_csv('Summary2023-2024.csv', encoding='latin-1')
```

```
# Paths
path_ana = '3rd times the charm\\ANA.webp'
path_ari = '3rd times the charm\\ARI.webp'
path_bos = '3rd times the charm\\BOS.webp'
path_buf = '3rd times the charm\\BUF.webp'
path_car = '3rd times the charm\\CAR.webp'
path_cbj = '3rd times the charm\\CBJ.webp'
path_cgy = '3rd times the charm\\CGY.webp'
path_chi = '3rd times the charm\\CHI.webp'
path_col = '3rd times the charm\\COL.webp'
path_dal = '3rd times the charm\\DAL.webp'
path_det = '3rd times the charm\\DET.webp'
path_edm = '3rd times the charm\\EDM.webp'
path_fla = '3rd times the charm\\FLA.webp'
path_lak = '3rd times the charm\\LAK.webp'
path_min = '3rd times the charm\\MIN.webp'
path_mtl = '3rd times the charm\\MTL.webp'
path_njd = '3rd times the charm\\NJD.webp'
path_nsh = '3rd times the charm\\NSH.webp'
path_nyi = '3rd times the charm\\NYI.webp'
path_nyr = '3rd times the charm\\NYR.webp'
path_ott = '3rd times the charm\\OTT.webp'
path_phi = '3rd times the charm\\PHI.webp'
path_pit = '3rd times the charm\\PIT.webp'
path_sea = '3rd times the charm\\SEA.webp'
path_sjs = '3rd times the charm\\SJS.webp'
path_stl = '3rd times the charm\\STL.webp'
path_tbl = '3rd times the charm\\TBL.webp'
path_tor = '3rd times the charm\\TOR.webp'
path_van = '3rd times the charm\\VAN.webp'
path_vgk = '3rd times the charm\\VGK.webp'
path_wpg = '3rd times the charm\\WPG.webp'
path_wsh = '3rd times the charm\\WSH.webp'



logos = {
    'ANA': '3rd times the charm\\ANA.webp',
    'ARI': '3rd times the charm\\ARI.webp',
    'BOS': '3rd times the charm\\BOS.webp',
    'BUF': '3rd times the charm\\BUF.webp',
    'CAR': '3rd times the charm\\CAR.webp',
    'CBJ': '3rd times the charm\\CBJ.webp',
    'CGY': '3rd times the charm\\CGY.webp',
    'CHI': '3rd times the charm\\CHI.webp',
    'COL': '3rd times the charm\\COL.webp',
    'DAL': '3rd times the charm\\DAL.webp',
    'DET': '3rd times the charm\\DET.webp',
    'EDM': '3rd times the charm\\EDM.webp',
    'FLA': '3rd times the charm\\FLA.webp',
    'LAK': '3rd times the charm\\LAK.webp',
    'MIN': '3rd times the charm\\MIN.webp',
    'MTL': '3rd times the charm\\MTL.webp',
    'NSH': '3rd times the charm\\NSH.webp',
    'NJD': '3rd times the charm\\NJD.webp',
    'NYI': '3rd times the charm\\NYI.webp',
    'NYR': '3rd times the charm\\NYR.webp',
    'OTT': '3rd times the charm\\OTT.webp',
    'PHI': '3rd times the charm\\PHI.webp',
    'PIT': '3rd times the charm\\PIT.webp',
    'SEA': '3rd times the charm\\SEA.webp',
    'SJS': '3rd times the charm\\SJS.webp',
    'STL': '3rd times the charm\\STL.webp',
    'TBL': '3rd times the charm\\TBL.webp',
    'TOR': '3rd times the charm\\TOR.webp',
    'VAN': '3rd times the charm\\VAN.webp',
    'VGK': '3rd times the charm\\VGK.webp',
    'WSH': '3rd times the charm\\WSH.webp',
    'WPG': '3rd times the charm\\WPG.webp'
}
```

```
ABV_mapping = {
    'New York Rangers': 'NYR',
    'Dallas Stars': 'DAL',
    'Carolina Hurricanes': 'CAR',
    'Florida Panthers': 'FLA',
    'Winnipeg Jets': 'WPG',
    'Vancouver Canucks': 'VAN',
    'Boston Bruins': 'BOS',
    'Colorado Avalanche': 'COL',
    'Edmonton Oilers': 'EDM',
    'Toronto Maple Leafs': 'TOR',
    'Nashville Predators': 'NSH',
    'Los Angeles Kings': 'LAK',
    'Tampa Bay Lightning': 'TBL',
    'Vegas Golden Knights': 'VGK',
    'New York Islanders': 'NYI',
    'St. Louis Blues': 'STL',
    'Detroit Red Wings': 'DET',
    'Washington Capitals': 'WSH',
    'Pittsburgh Penguins': 'PIT',
    'Minnesota Wild': 'MIN',
    'Philadelphia Flyers': 'PHI',
    'Buffalo Sabres': 'BUF',
    'New Jersey Devils': 'NJD',
    'Calgary Flames': 'CGY',
    'Seattle Kraken': 'SEA',
    'Ottawa Senators': 'OTT',
    'Arizona Coyotes': 'ARI',
    'Montréal Canadiens': 'MTL',
    'Columbus Blue Jackets': 'CBJ',
    'Anaheim Ducks': 'ANA',
    'Chicago Blackhawks': 'CHI',
    'San Jose Sharks': 'SJS',
}


playoffs_2023= {
    'New York Rangers': 1,
    'Dallas Stars': 1,
    'Carolina Hurricanes': 1,
    'Florida Panthers': 1,
    'Winnipeg Jets': 1,
    'Vancouver Canucks': 1,
    'Boston Bruins': 1,
    'Colorado Avalanche': 1,
    'Edmonton Oilers': 1,
    'Toronto Maple Leafs': 1,
    'Nashville Predators': 1,
    'Los Angeles Kings': 1,
    'Tampa Bay Lightning': 1,
    'Vegas Golden Knights': 1,
    'New York Islanders': 1,
    'St. Louis Blues': 0,
    'Detroit Red Wings': 0,
    'Washington Capitals': 1,
    'Pittsburgh Penguins': 0,
    'Minnesota Wild': 0,
    'Philadelphia Flyers': 0,
    'Buffalo Sabres': 0,
    'New Jersey Devils': 0,
    'Calgary Flames': 0,
    'Seattle Kraken': 0,
    'Ottawa Senators': 0,
    'Arizona Coyotes': 0,
    'Montréal Canadiens': 0,
    'Columbus Blue Jackets': 0,
    'Anaheim Ducks': 0,
    'Chicago Blackhawks': 0,
    'San Jose Sharks': 0,
}
```

```
playoffs_2022= {
    'New York Rangers': 1,
    'Dallas Stars': 1,
    'Carolina Hurricanes': 1,
    'Florida Panthers': 1,
    'Winnipeg Jets': 1,
    'Vancouver Canucks': 0,
    'Boston Bruins': 1,
    'Colorado Avalanche': 1,
    'Edmonton Oilers': 1,
    'Toronto Maple Leafs': 1,
    'Nashville Predators': 0,
    'Los Angeles Kings': 1,
    'Tampa Bay Lightning': 1,
    'Vegas Golden Knights': 1,
    'New York Islanders': 1,
    'St. Louis Blues': 0,
    'Detroit Red Wings': 0,
    'Washington Capitals': 0,
    'Pittsburgh Penguins': 0,
    'Minnesota Wild': 1,
    'Philadelphia Flyers': 0,
    'Buffalo Sabres': 0,
    'New Jersey Devils': 1,
    'Calgary Flames': 0,
    'Seattle Kraken': 1,
    'Ottawa Senators': 0,
    'Arizona Coyotes': 0,
    'Montréal Canadiens': 0,
    'Columbus Blue Jackets': 0,
    'Anaheim Ducks': 0,
    'Chicago Blackhawks': 0,
    'San Jose Sharks': 0,
}


playoffs_2021= {
    'New York Rangers': 1,
    'Dallas Stars': 1,
    'Carolina Hurricanes': 1,
    'Florida Panthers': 1,
    'Winnipeg Jets': 0,
    'Vancouver Canucks': 0,
    'Boston Bruins': 1,
    'Colorado Avalanche': 1,
    'Edmonton Oilers': 1,
    'Toronto Maple Leafs': 1,
    'Nashville Predators': 1,
    'Los Angeles Kings': 1,
    'Tampa Bay Lightning': 1,
    'Vegas Golden Knights': 0,
    'New York Islanders': 0,
    'St. Louis Blues': 1,
    'Detroit Red Wings': 0,
    'Washington Capitals': 1,
    'Pittsburgh Penguins': 1,
    'Minnesota Wild': 1,
    'Philadelphia Flyers': 0,
    'Buffalo Sabres': 0,
    'New Jersey Devils': 0,
    'Calgary Flames': 1,
    'Seattle Kraken': 0,
    'Ottawa Senators': 0,
    'Arizona Coyotes': 0,
    'Montréal Canadiens': 0,
    'Columbus Blue Jackets': 0,
    'Anaheim Ducks': 0,
    'Chicago Blackhawks': 0,
    'San Jose Sharks': 0,
}


#apply mapping to the nhl datasets
nhl2021['ABV'] = nhl2021['Team'].map(ABV_mapping)
nhl2022['ABV'] = nhl2022['Team'].map(ABV_mapping)
nhl2023['ABV'] = nhl2023['Team'].map(ABV_mapping)
```

```
#apply mapping to the nhl dataset
nhl2021['logo'] = nhl2021['ABV'].map(logos)
nhl2022['logo'] = nhl2022['ABV'].map(logos)
nhl2023['logo'] = nhl2023['ABV'].map(logos)


#apply mapping to the nhl datasets
nhl2021['playoffs'] = nhl2021['Team'].map(playoffs_2021)
nhl2022['playoffs'] = nhl2022['Team'].map(playoffs_2022)
nhl2023['playoffs'] = nhl2023['Team'].map(playoffs_2023)


teams_2021 = pd.merge(mp2021, nhl2021, left_on='team', right_on='ABV')
teams_2022 = pd.merge(mp2022, nhl2022, left_on='team', right_on='ABV')
teams_2023 = pd.merge(mp2023, nhl2023, left_on='team', right_on='ABV')


teams = pd.concat([teams_2021, teams_2022, teams_2023])


division_map = {
    'TOR': 'Atlantic',
    'NSH': 'Central',
    'NJD': 'Metropolitan',
    'COL': 'Central',
    'VGK': 'Pacific',
    'PIT': 'Metropolitan',
    'CHI': 'Central',
    'CBJ': 'Metropolitan',
    'NYR': 'Metropolitan',
    'CGY': 'Pacific',
    'OTT': 'Atlantic',
    'LAK': 'Pacific',
    'SEA': 'Pacific',
    'EDM': 'Pacific',
    'TBL': 'Atlantic',
    'MTL': 'Atlantic',
    'DAL': 'Central',
    'FLA': 'Atlantic',
    'STL': 'Central',
    'CAR': 'Metropolitan',
    'VAN': 'Pacific',
    'NYI': 'Metropolitan',
    'WPG': 'Central',
    'MIN': 'Central',
    'ANA': 'Pacific',
    'WSH': 'Metropolitan',
    'BUF': 'Atlantic',
    'PHI': 'Metropolitan',
    'BOS': 'Atlantic',
    'DET': 'Atlantic',
    'ARI': 'Central',
    'SJS': 'Pacific'
}


conference_map = {
    'Atlantic': 'Eastern',
    'Metropolitan': 'Eastern',
    'Central': 'Western',
    'Pacific': 'Western'
}


teams['division'] = teams['ABV'].map(division_map)
teams['conference'] = teams['division'].map(conference_map)
```

```
columns = ['Team',
           'ABV',
           'Season',
           'division',
           'conference',
           'GP',
           'W',
           'L',
           'OT',
           'P',
           'P%',
           'RW',
           'ROW',
           'GF',
           'GA',
           'GF/GP',
           'GA/GP',
           'PP%',
           'PK%',
           'Shots/GP',
           'SA/GP',
           'FOW%',
           'situation',
           'iceTime',
           'xGoalsFor',
           'xGoalsAgainst',
           'xGoalsPercentage',
           'corsiPercentage',
           'fenwickPercentage',
           'hitsFor',
           'penaltiesFor',
           'takeawaysFor',
           'giveawaysFor',
           'lowDangerShotsFor',
           'mediumDangerShotsFor',
           'highDangerShotsFor',
           'penaltiesAgainst',
           'hitsAgainst',
           'takeawaysAgainst',
           'giveawaysAgainst',
           'lowDangerShotsAgainst',
           'mediumDangerShotsAgainst',
           'highDangerShotsAgainst',
           'logo',
           'playoffs'
]

teams = teams[columns]

teams.rename(columns={
    'Team': 'team',
    'ABV': 'abv',
    'Season': 'season',
    'iceTime' : 'icetime',
    'xGoalsFor' : 'xGF',
    'xGoalsAgainst': 'xGA',
    'xGoalsPercentage': 'xG%',
    'corsiPercentage': 'corsi',
    'fenwickPercentage': 'fenwick'
}, inplace=True)

teams.head()
```

	team	abv	season	division	conference	GP	W	L	OT	P	...	highDangerShotsFor	penaltiesAgainst	hitsAgainst	takeawaysA
0	Winnipeg Jets	WPG	20212022	Central	Western	82	39	32	11	89	...	46.0	6.0	40.0	
1	Winnipeg Jets	WPG	20212022	Central	Western	82	39	32	11	89	...	268.0	297.0	1658.0	
2	Winnipeg Jets	WPG	20212022	Central	Western	82	39	32	11	89	...	156.0	264.0	1527.0	
3	Winnipeg Jets	WPG	20212022	Central	Western	82	39	32	11	89	...	17.0	16.0	30.0	

```
teams['hit%'] = teams['hitsFor'] / ((teams['hitsFor']) + (teams['hitsAgainst']))
teams['takeaway%'] = teams['takeawaysFor'] / ((teams ['takeawaysFor']) + (teams['takeawaysAgainst']))
teams['giveaway%'] = teams['giveawaysFor'] / ((teams['giveawaysFor']) + (teams['giveawaysAgainst']))
teams['PIM%'] = teams['penaltiesFor'] / ((teams['penaltiesFor']) + (teams['penaltiesAgainst']))
teams['LDS%'] = teams['lowDangerShotsFor'] / ((teams['lowDangerShotsFor']) + (teams['lowDangerShotsAgainst']))
teams['MDS%'] = teams['mediumDangerShotsFor'] / ((teams['mediumDangerShotsFor']) + (teams['mediumDangerShotsAgainst']))
teams['HDS%'] = teams['highDangerShotsFor'] / ((teams['highDangerShotsFor']) + (teams['highDangerShotsAgainst']))
teams['special'] = teams['PP%'] + teams['PK%']
teams['G%'] = teams['GF'] / ((teams['GF']) + (teams['GA']))
teams['shotsFor'] = teams['Shots/GP'] * 82
teams['shotsAgainst'] = teams['SA/GP'] * 82
teams['shots%'] = teams['shotsFor'] / ((teams ['shotsFor']) + (teams['shotsAgainst']))
```

```
teams.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 480 entries, 0 to 159
Data columns (total 57 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   team                      480 non-null    object
 1   abv                       480 non-null    object
 2   season                    480 non-null    int64
 3   division                  480 non-null    object
 4   conference                480 non-null    object
 5   GP                        480 non-null    int64
 6   W                         480 non-null    int64
 7   L                         480 non-null    int64
 8   OT                        480 non-null    int64
 9   P                         480 non-null    int64
 10  P%                        480 non-null    float64
 11  RW                        480 non-null    int64
 12  ROW                       480 non-null    int64
 13  GF                        480 non-null    int64
 14  GA                        480 non-null    int64
 15  GF/GP                     480 non-null    float64
 16  GA/GP                     480 non-null    float64
 17  PP%                       480 non-null    float64
 18  PK%                       480 non-null    float64
 19  Shots/GP                  480 non-null    float64
 20  SA/GP                     480 non-null    float64
 21  FOW%                      480 non-null    float64
 22  situation                 480 non-null    object
 23  icetime                   480 non-null    float64
 24  xGF                       480 non-null    float64
 25  xGA                       480 non-null    float64
 26  xG%                       480 non-null    float64
 27  corsi                     480 non-null    float64
 28  fenwick                   480 non-null    float64
 29  hitsFor                   480 non-null    float64
 30  penaltiesFor              480 non-null    float64
 31  takeawaysFor              480 non-null    float64
 32  giveawaysFor              480 non-null    float64
 33  lowDangerShotsFor         480 non-null    float64
 34  mediumDangerShotsFor      480 non-null    float64
 35  highDangerShotsFor        480 non-null    float64
 36  penaltiesAgainst          480 non-null    float64
 37  hitsAgainst               480 non-null    float64
 38  takeawaysAgainst          480 non-null    float64
 39  giveawaysAgainst          480 non-null    float64
 40  lowDangerShotsAgainst     480 non-null    float64
 41  mediumDangerShotsAgainst  480 non-null    float64
 42  highDangerShotsAgainst    480 non-null    float64
 43  logo                      480 non-null    object
 44  playoffs                  480 non-null    int64
 45  hit%                      480 non-null    float64
 46  takeaway%                 480 non-null    float64
 47  giveaway%                 480 non-null    float64
 48  PIM%                      480 non-null    float64
 49  LDS%                      480 non-null    float64
 50  MDS%                      480 non-null    float64
 51  HDS%                      480 non-null    float64
 52  special                   480 non-null    float64
```

```
# Situational masks
MASK_5on5 = (teams['situation'] == '5on5')
MASK_PK = (teams['situation'] == '4on5')
MASK_PP = (teams['situation'] == '5on4')
MASK_ALL = (teams['situation'] == 'all')
```

```
# Division Masks
MASK_ATL = (teams['division'] == 'Atlantic')
MASK_MET = (teams['division'] == 'Metropolitan')
MASK_CEN = (teams['division'] == 'Central')
MASK_PAC = (teams['division'] == 'Pacific')


# Conference Masks
MASK_EAST = (teams['conference'] == 'Eastern')
MASK_WEST = (teams['conference'] == 'Western')


# year masks
MASK2021 = (teams['season'] == 20212022)
MASK2022 = (teams['season'] == 20222023)
MASK2023 = (teams['season'] == 20232024)
```

The following function creates a kdeplot for a two sample ttest when it comes to comparing mean points. Also inlcuded in this a comparison to the average amount of points it has taken to make the playoffs in the past 3 seasons (93.5) as a measure of if a team would make the playoffs if they had a certain level of points

```
# Creating a function to plot the KDE of two datasets
def two_sample_kdeplot(dataset1, dataset2, label1, label2, save=False):
    # Calculate the means of both datasets
    mean1 = np.mean(dataset1)
    mean2 = np.mean(dataset2)

    # Perform two sample t-test
    _, p_value = stats.ttest_ind(dataset1, dataset2)
    if p_value < 0.05:
        print('Reject the Null Hypothesis')
    else:
        print('Fail to reject the null hypothesis')

    # Use a style template
    sns.set_style('whitegrid')

    plt.figure(figsize=(12, 8))

    # Plot both datasets
    sns.kdeplot(dataset1, label=label1, color='red', alpha=1)
    sns.kdeplot(dataset2, label=label2, color='blue', alpha=1)

    # Compute KDE values
    kde_dataset1 = stats.gaussian_kde(dataset1)
    kde_dataset2 = stats.gaussian_kde(dataset2)

    threshold = 93.5

    # Calculate the probability for 'dataset1' group
    prob_dataset1 = 1 - kde_dataset1.integrate_box_1d(threshold, np.inf)

    # Calculate the probability for 'dataset2' group
    prob_dataset2 = 1 - kde_dataset2.integrate_box_1d(threshold, np.inf)

    print(f'Probability of making the playoffs for {label1}: {prob_dataset1:.4f}')
    print(f'Probability of making the playoffs for {label2}: {prob_dataset2:.4f}')

    # Generate x values for KDE
    x_vals = np.linspace(min(dataset1.min(), dataset2.min()), max(dataset1.max(), dataset2.max()), 1000)

    # Fill the entire area under the curves
    plt.fill_between(x_vals, kde_dataset1(x_vals), color='red', alpha=0.3)
    plt.fill_between(x_vals, kde_dataset2(x_vals), color='blue', alpha=0.3)

    # Get KDE values at means
    kde_dataset1_at_mean1 = kde_dataset1(mean1)[0]
    kde_dataset2_at_mean2 = kde_dataset2(mean2)[0]

    # Add vertical lines at the means, stopping at the KDE curves
    plt.plot([mean1, mean1], [0, kde_dataset1_at_mean1], 'r--', label=f'{label1} Mean: {mean1:.2f}')
    plt.plot([mean2, mean2], [0, kde_dataset2_at_mean2], 'b--', label=f'{label2} Mean: {mean2:.2f}')

    # Add a gold vertical line at 93.5
    plt.axvline(93.5, color='gold', linestyle='-', linewidth=2, label='Playoff Line')

    # Add labels and title
    plt.title('KDE Plots of Two Datasets', fontsize=20, fontweight='bold')
    plt.xlabel('Value', fontsize=16)
    plt.ylabel('')
    plt.yticks(visible=False)

    # Add annotations
    plt.annotate(f'{label1} | Mean: {mean1:.2f}', xy=(0.75, 0.85), xycoords='axes fraction', color='red', fontsize=12,
                 bbox=dict(facecolor='white', edgecolor='red', boxstyle='round,pad=0.5'))
    plt.annotate(f'{label2} | Mean: {mean2:.2f}', xy=(0.75, 0.75), xycoords='axes fraction', color='blue', fontsize=12,
                 bbox=dict(facecolor='white', edgecolor='blue', boxstyle='round,pad=0.5'))
    plt.annotate('Playoff Line', xy=(0.75, 0.65), xycoords='axes fraction', color='black', fontsize=12,
                 bbox=dict(facecolor='gold', edgecolor='gold', boxstyle='round,pad=0.5'))

    if save == True:
        plt.savefig(f'{label1} v {label2} TTest.png', transparent=True)
    plt.show()
```

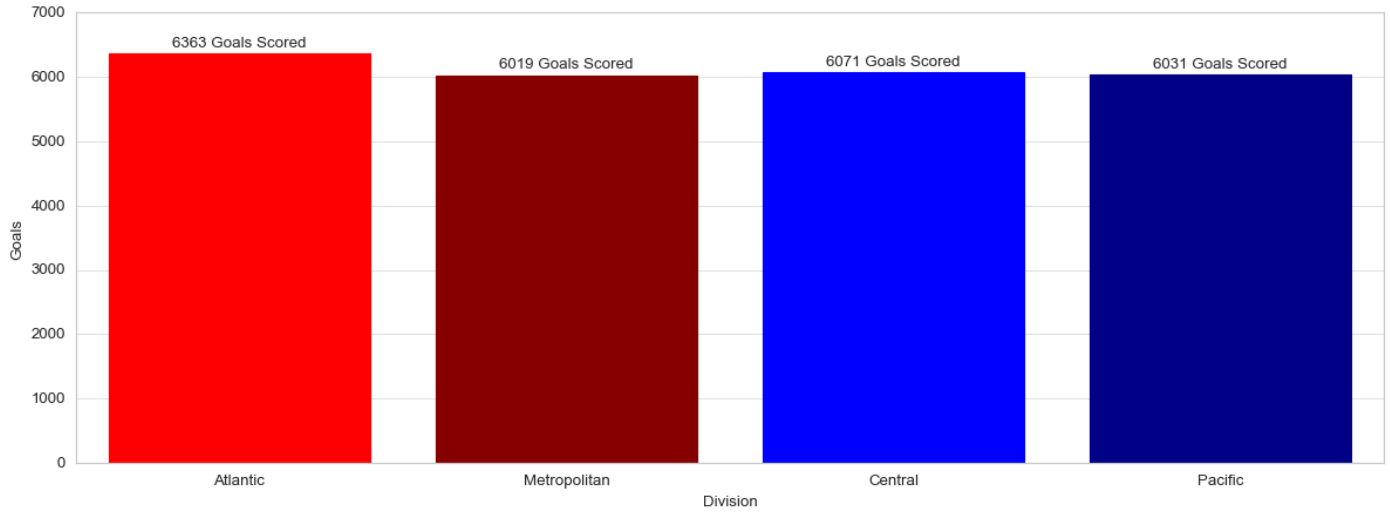
Descriptive Questions

Descriptive Question 1: Over the Past 3 Seasons, Which Division attributed for the most amount of goals?

For this, I will group the data by division and get the sum of `Goals For` and then graph the results in a bar plot

```
# Over the past 3 season, which division has the most amount of goals
Q1_ATL_GF = teams[MASK_ATL & MASK_ALL]['GF'].sum()
Q1_MET_GF = teams[MASK_MET & MASK_ALL]['GF'].sum()
Q1_CEN_GF = teams[MASK_CEN & MASK_ALL]['GF'].sum()
Q1_PAC_GF = teams[MASK_PAC & MASK_ALL]['GF'].sum()


#combining all the division into a single DataFrame

Q1 = pd.DataFrame({
    'Division': ['Atlantic', 'Metropolitan', 'Central', 'Pacific'],
    'Goals': [Q1_ATL_GF, Q1_MET_GF, Q1_CEN_GF, Q1_PAC_GF]
})


# Plot size
fig, ax = plt.subplots(figsize=(18, 8))
fig.subplots_adjust(bottom=0.3)

# Data
bars = sns.barplot(x=Q1['Division'], y=Q1['Goals'], palette='bwr_r')

# Customizing bar colors
colors = ['red', 'darkred', 'blue', 'darkblue']
for bar, color in zip(bars.patches, colors):
    bar.set_color(color)

# Adding goal totals over each bar
for bar in bars.patches:
    height = bar.get_height()
    ax.annotate(f'{height:.0f} Goals Scored',
                xy=(bar.get_x() + bar.get_width() / 2, height),
                xytext=(0, 3),  # 3 points vertical offset
                textcoords="offset points",
                ha='center', va='bottom')

# plt.savefig('GoalsByDivision.png')
plt.show()
```

```
C:\Users\Shank\AppData\Local\Temp\ipykernel_5504\1407060638.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  bars = sns.barplot(x=Q1['Division'], y=Q1['Goals'], palette='bwr_r')
```



Descriptive Question 2: Which Division attributed for the most amount of points in the past 3 seasons?

POINTS (`P`) - Points are awarded to team after each game. Teams receive **TWO** points for every win and **ONE** point for every overtime or shootout loss

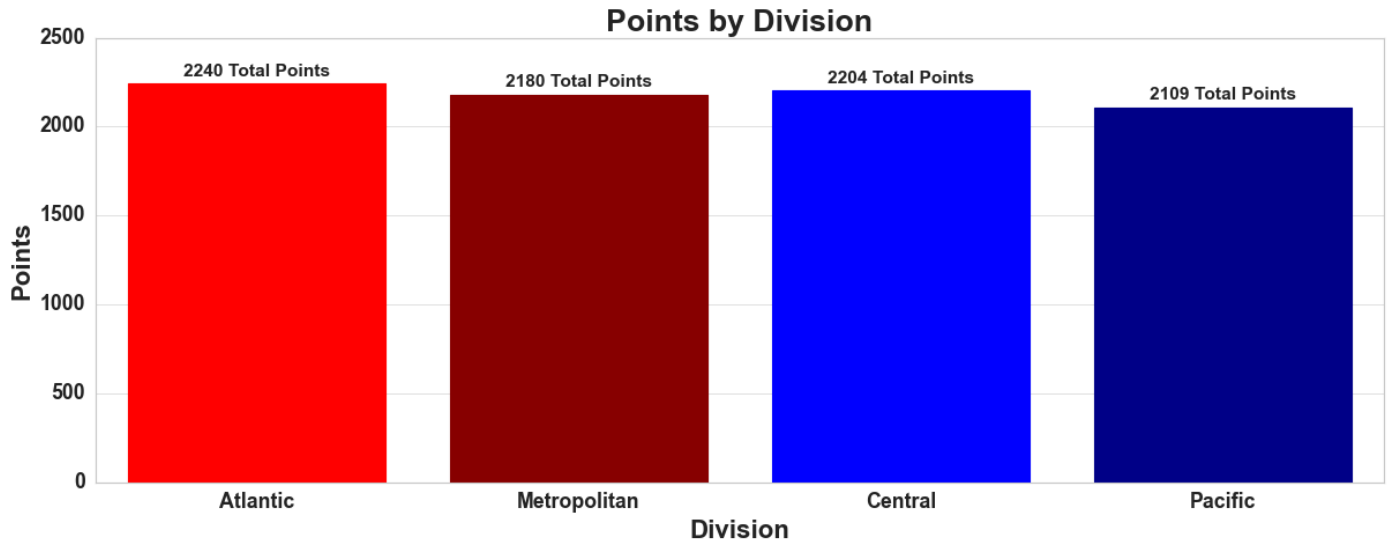
For this question, I grouped the data by division and calculated the sum of all points (`P`) from the past 3 seasons.

```
Q2_ATL_P = teams[MASK_ATL & MASK_ALL]['P'].sum()
Q2_MET_P = teams[MASK_MET & MASK_ALL]['P'].sum()
Q2_CEN_P = teams[MASK_CEN & MASK_ALL]['P'].sum()
Q2_PAC_P = teams[MASK_PAC & MASK_ALL]['P'].sum()
```

```
Q2 = pd.DataFrame({
    'Division': ['Atlantic', 'Metropolitan', 'Central', 'Pacific'],
    'Points': [Q2_ATL_P, Q2_MET_P, Q2_CEN_P, Q2_PAC_P]
})
```

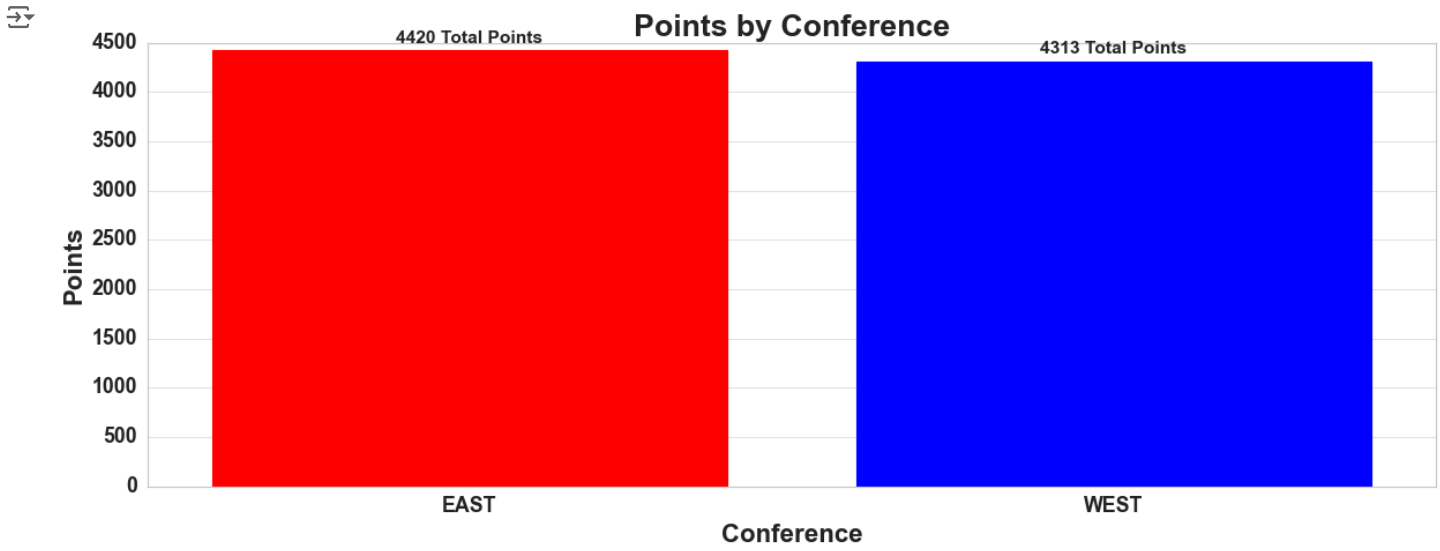
```
# Plot size
fig, ax = plt.subplots(figsize=(18, 8))
fig.subplots_adjust(bottom=0.3)

# Data
bars = sns.barplot(x=Q2['Division'], y=Q2['Points'], palette='bwr_r')

# Customizing bar colors
colors = ['red', 'darkred', 'blue', 'darkblue']
for bar, color in zip(bars.patches, colors):
    bar.set_color(color)

# Adding point totals over each bar
for bar in bars.patches:
    height = bar.get_height()
    ax.annotate(f'{height:.0f} Total Points',
                xy=(bar.get_x() + bar.get_width() / 2, height),
                xytext=(0, 3),  # 3 points vertical offset
                textcoords="offset points",
                ha='center', va='bottom',
                fontsize=14, fontweight='bold')  # Make annotation text bigger and bolder

# Customizing title and axis labels
ax.set_title('Points by Division', fontsize=24, fontweight='bold')
ax.set_xlabel('Division', fontsize=20, fontweight='bold')
ax.set_ylabel('Points', fontsize=20, fontweight='bold')

# Customizing tick labels
ax.tick_params(axis='both', which='major', labelsize=16, width=2)  # Increase tick label size
plt.xticks(fontsize=16, fontweight='bold')
plt.yticks(fontsize=16, fontweight='bold')

# Customizing legend (if applicable)
# If you have a legend, you can set its fontsize and fontweight as well
# ax.legend(fontsize=16, title_fontsize=18, title_fontweight='bold')

plt.savefig('PointsByDivision.png', transparent=True)
plt.show()
```

```
C:\Users\Shank\AppData\Local\Temp\ipykernel_5504\1446310048.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  bars = sns.barplot(x=Q2['Division'], y=Q2['Points'], palette='bwr_r')
```




```
Q2_EAST_P = teams[MASK_EAST & MASK_ALL]['P'].sum()
Q2_WEST_P = teams[MASK_WEST & MASK_ALL]['P'].sum()
```

```
Q8 = pd.DataFrame({
    'Division': ['EAST', 'WEST'],
    'Points': [Q2_EAST_P, Q2_WEST_P]
})


# Plot size
fig, ax = plt.subplots(figsize=(18, 8))
fig.subplots_adjust(bottom=0.3)

# Data
bars = sns.barplot(x=Q8['Division'], y=Q8['Points'])
sns.set_style('whitegrid')

# Customizing bar colors
colors = ['red', 'blue']
for bar, color in zip(bars.patches, colors):
    bar.set_color(color)

# Adding point totals over each bar
for bar in bars.patches:
    height = bar.get_height()
    ax.annotate(f'{height:.0f} Total Points',
                xy=(bar.get_x() + bar.get_width() / 2, height),
                xytext=(0, 3),  # 3 points vertical offset
                textcoords="offset points",
                ha='center', va='bottom',
                fontsize=14, fontweight='bold')  # Make annotation text bigger and bolder

# Customizing title and axis labels
ax.set_title('Points by Conference', fontsize=24, fontweight='bold')
ax.set_xlabel('Conference', fontsize=20, fontweight='bold')
ax.set_ylabel('Points', fontsize=20, fontweight='bold')

# Customizing tick labels
ax.tick_params(axis='both', which='major', labelsize=16, width=2)  # Increase tick label size
plt.xticks(fontsize=16, fontweight='bold')
plt.yticks(fontsize=16, fontweight='bold')

# Customizing legend (if applicable)
# If you have a legend, you can set its fontsize and fontweight as well
# ax.legend(fontsize=16, title_fontsize=18, title_fontweight='bold')

plt.savefig('PointsByConference.png', transparent=True)
plt.show()
```

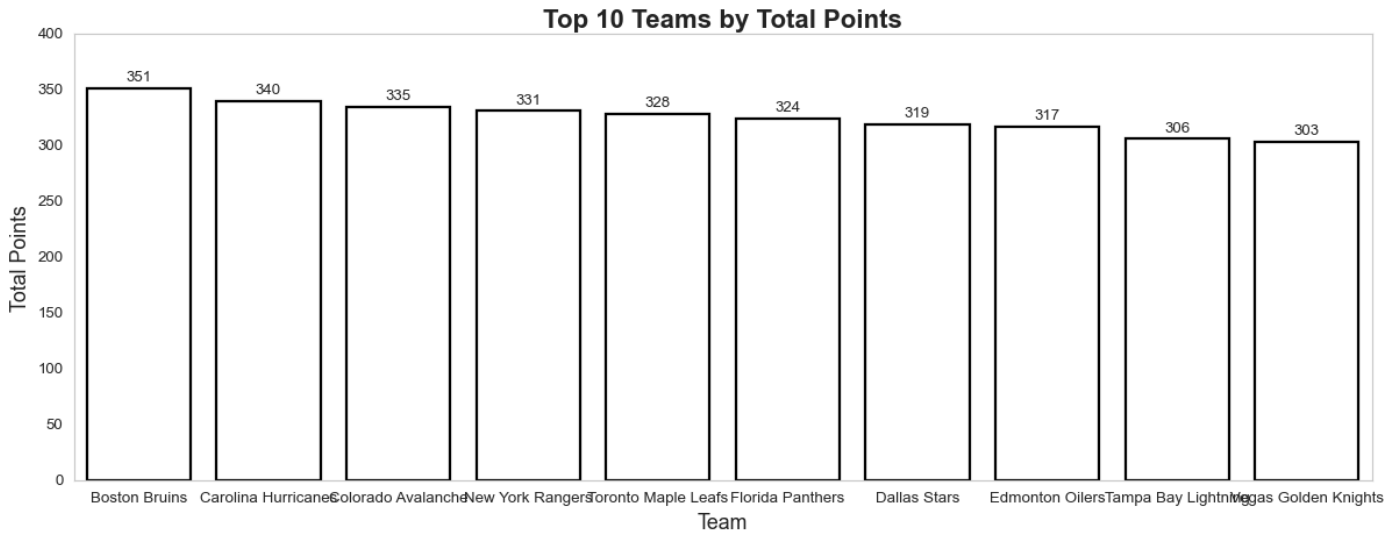


Descriptive Question 3: Which Team accumulated the most amount of points over the past 3 season?

For this question, I grouped the data by team and then calculated the point totals for each team

```
Q3_df = teams[MASK_ALL]
Q3 = Q3_df.groupby('team')['P'].sum().sort_values(ascending=False).head(10)


fig, ax = plt.subplots(figsize=(18, 8))
fig.subplots_adjust(bottom=0.3)

# Plot data
bars = sns.barplot(x=Q3.index, y=Q3.values, palette=['white']*10, edgecolor='black', linewidth=2)

# Add title and labels
ax.set_title('Top 10 Teams by Total Points', fontsize=20, fontweight='bold')
ax.set_xlabel('Team', fontsize=16)
ax.set_ylabel('Total Points', fontsize=16)

# Add data labels
for bar in bars.patches:
    ax.annotate(format(bar.get_height(), '.0f'),
                xy=(bar.get_x() + bar.get_width() / 2, bar.get_height()),
                xytext=(0, 3),  # 3 points vertical offset
                textcoords="offset points",
                ha='center', va='bottom', fontsize=12)

# Improve the overall appearance
sns.set_style('whitegrid')
ax.tick_params(axis='x', rotation=0)  # Straighten out the tick labels
ax.yaxis.grid(True, linestyle='--', which='both', color='grey', alpha=0)
ax.xaxis.grid(False)
# plt.xticks(visible=False)

# Show plot
plt.show()
```

```
C:\Users\Shank\AppData\Local\Temp\ipykernel_5504\2806150959.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  bars = sns.barplot(x=Q3.index, y=Q3.values, palette=['white']*10, edgecolor='black', linewidth=2)
```



Descriptive Question 4: Which Team scored the most amount of Goals over the past 3 seasons?

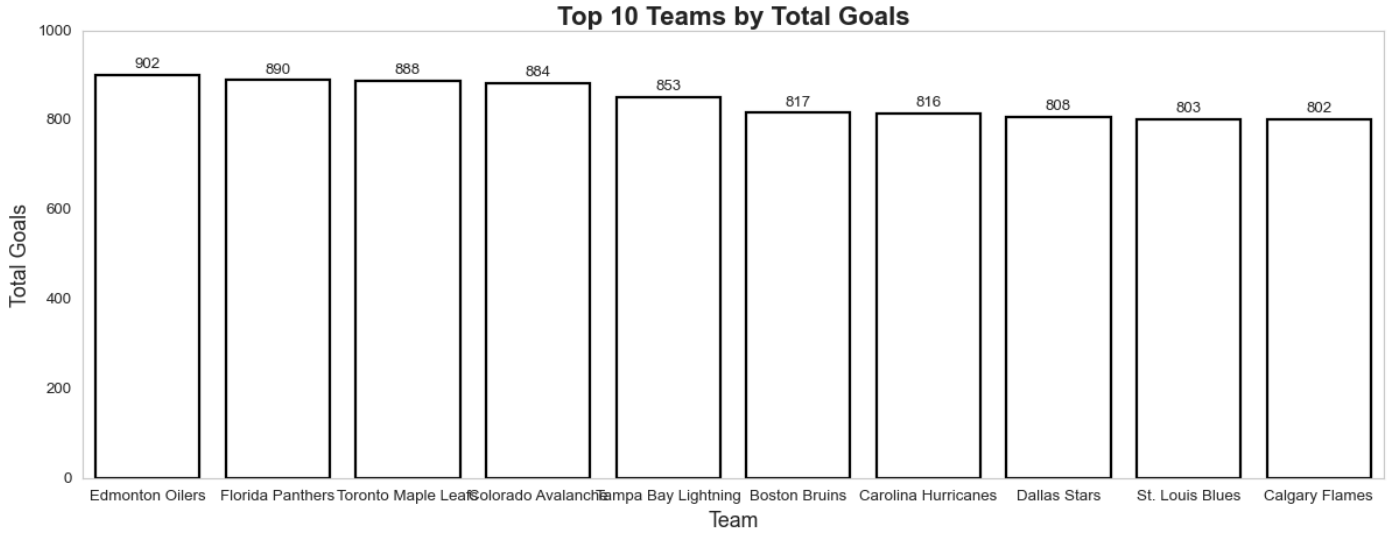
For this question, I grouped the data by team and then calculated the Goals For totals for each team.

`Goals For` - The amount of goals a team scores

```
Q4_df = teams[MASK_ALL]
Q4 = Q4_df.groupby('team')['GF'].sum().sort_values(ascending=False).head(10)
```

```
fig, ax = plt.subplots(figsize=(18, 8))
fig.subplots_adjust(bottom=0.3)

# Plot data
bars = sns.barplot(x=Q4.index, y=Q4.values, palette=['white']*10, edgecolor='black', linewidth=2)

# Add title and labels
ax.set_title('Top 10 Teams by Total Goals', fontsize=20, fontweight='bold')
ax.set_xlabel('Team', fontsize=16)
ax.set_ylabel('Total Goals', fontsize=16)

# Add data labels
for bar in bars.patches:
    ax.annotate(format(bar.get_height(), '.0f'),
                xy=(bar.get_x() + bar.get_width() / 2, bar.get_height()),
                xytext=(0, 3),  # 3 points vertical offset
                textcoords="offset points",
                ha='center', va='bottom', fontsize=12)

# Improve the overall appearance
sns.set_style('whitegrid')
ax.tick_params(axis='x', rotation=0)  # Straighten out the tick labels
ax.yaxis.grid(True, linestyle='--', which='both', color='grey', alpha=0)
ax.xaxis.grid(False)
# plt.xticks(visible=False)

# Show plot
# plt.savefig('TeamGoals.png', transparent=True)
plt.show()
```

```
C:\Users\Shank\AppData\Local\Temp\ipykernel_5504\1332555113.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  bars = sns.barplot(x=Q4.index, y=Q4.values, palette=['white']*10, edgecolor='black', linewidth=2)
```



Descriptive Question 5: Which Team gave up the **MOST** amount of Goals over the past 3 seasons?

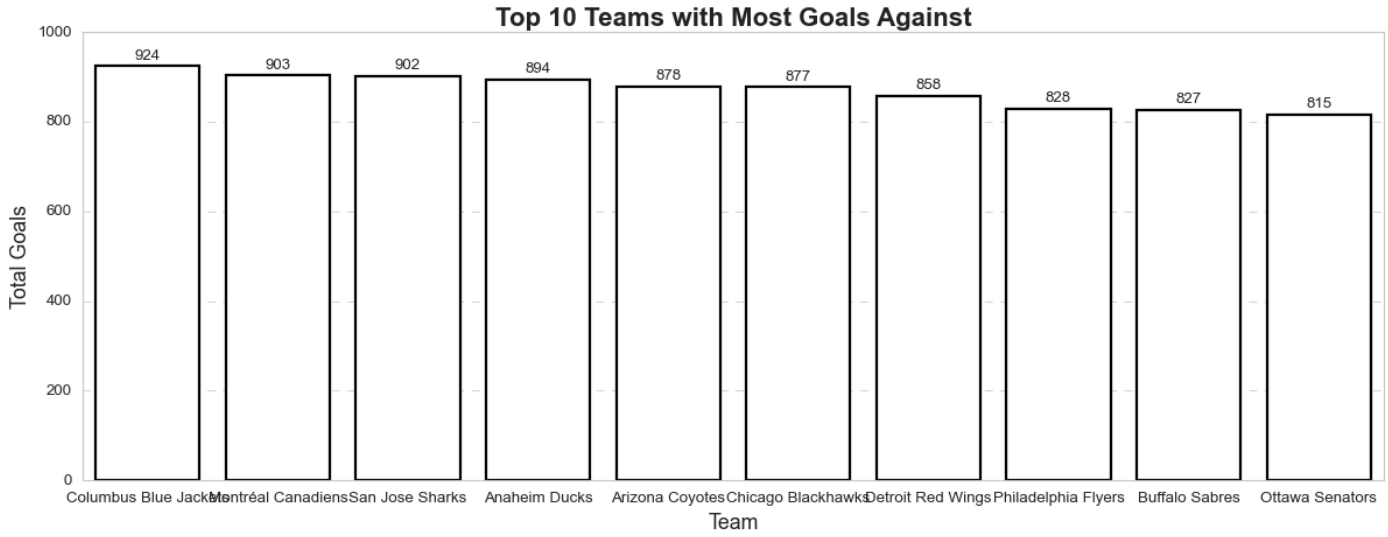
For this question, I grouped the data by team and then calculated the Goal Against total for each team.

`Goals Against` - The amount of goals a team gives up

```
Q5_df = teams[MASK_ALL]
Q5 = Q5_df.groupby('team')['GA'].sum().sort_values(ascending=False).head(10)
```

```
# Create the plot
fig, ax = plt.subplots(figsize=(18, 8))
fig.subplots_adjust(bottom=0.3)

# Plot data
bars = sns.barplot(x=Q5.index, y=Q5.values, palette=['white']*10, edgecolor='black', linewidth=2)

# Add title and labels
ax.set_title('Top 10 Teams with Most Goals Against', fontsize=20, fontweight='bold')
ax.set_xlabel('Team', fontsize=16)
ax.set_ylabel('Total Goals', fontsize=16)

# Add data labels
for bar in bars.patches:
    ax.annotate(format(bar.get_height(), '.0f'),
                xy=(bar.get_x() + bar.get_width() / 2, bar.get_height()),
                xytext=(0, 3),  # 3 points vertical offset
                textcoords="offset points",
                ha='center', va='bottom', fontsize=12)

# Improve the overall appearance
sns.set_style('whitegrid')
ax.tick_params(axis='x', rotation=0)  # Straighten out the tick labels
ax.yaxis.grid(True, linestyle='--', which='both', color='grey', alpha=0.6)
ax.xaxis.grid(False)

plt.show()
```

```
C:\Users\Shank\AppData\Local\Temp\ipykernel_5504\3423024388.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  bars = sns.barplot(x=Q5.index, y=Q5.values, palette=['white']*10, edgecolor='black', linewidth=2)
```



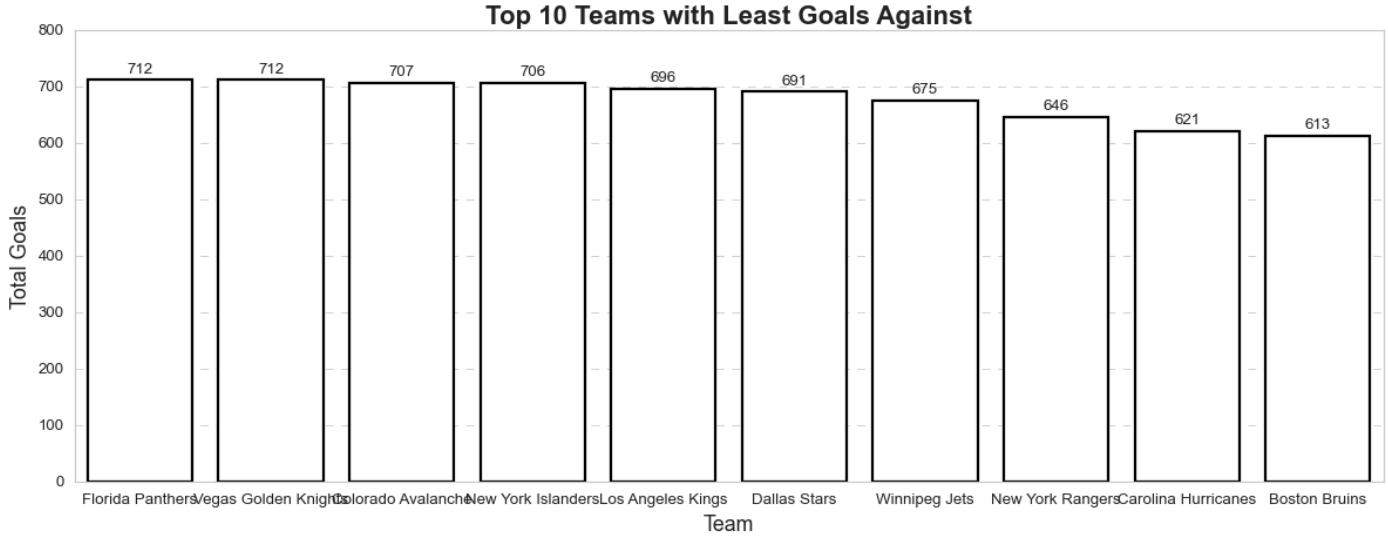
Descriptive Question 6: Which Team gave up the **LEAST** amount of Goals over the past 3 seasons?

For this question, I grouped the data by team and then calculated the Goal Against total for each team.

```
Q6_df = teams[MASK_ALL]
Q6 = Q6_df.groupby('team')['GA'].sum().sort_values(ascending=False).tail(10)
```

```
# Create the plot
fig, ax = plt.subplots(figsize=(18, 8))
fig.subplots_adjust(bottom=0.3)

# Plot data
bars = sns.barplot(x=Q6.index, y=Q6.values, palette=['white']*10, edgecolor='black', linewidth=2)

# Add title and labels
ax.set_title('Top 10 Teams with Least Goals Against', fontsize=20, fontweight='bold')
ax.set_xlabel('Team', fontsize=16)
ax.set_ylabel('Total Goals', fontsize=16)

# Add data labels
for bar in bars.patches:
    ax.annotate(format(bar.get_height(), '.0f'),
                xy=(bar.get_x() + bar.get_width() / 2, bar.get_height()),
                xytext=(0, 3),  # 3 points vertical offset
                textcoords="offset points",
                ha='center', va='bottom', fontsize=12)

# Improve the overall appearance
sns.set_style('whitegrid')
ax.tick_params(axis='x', rotation=0)  # Straighten out the tick labels
ax.yaxis.grid(True, linestyle='--', which='both', color='grey', alpha=0.6)
ax.xaxis.grid(False)

# plt.xticks(visible=False)
# Show plot
# plt.savefig('TeamGoalsAgainst.png', transparent=True)
plt.show()
```

```
C:\Users\Shank\AppData\Local\Temp\ipykernel_5504\1136071622.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  bars = sns.barplot(x=Q6.index, y=Q6.values, palette=['white']*10, edgecolor='black', linewidth=2)
```



Descriptive Question 7: What is the relation between Goals For and Goals Against for each team over the past 3 seasons?

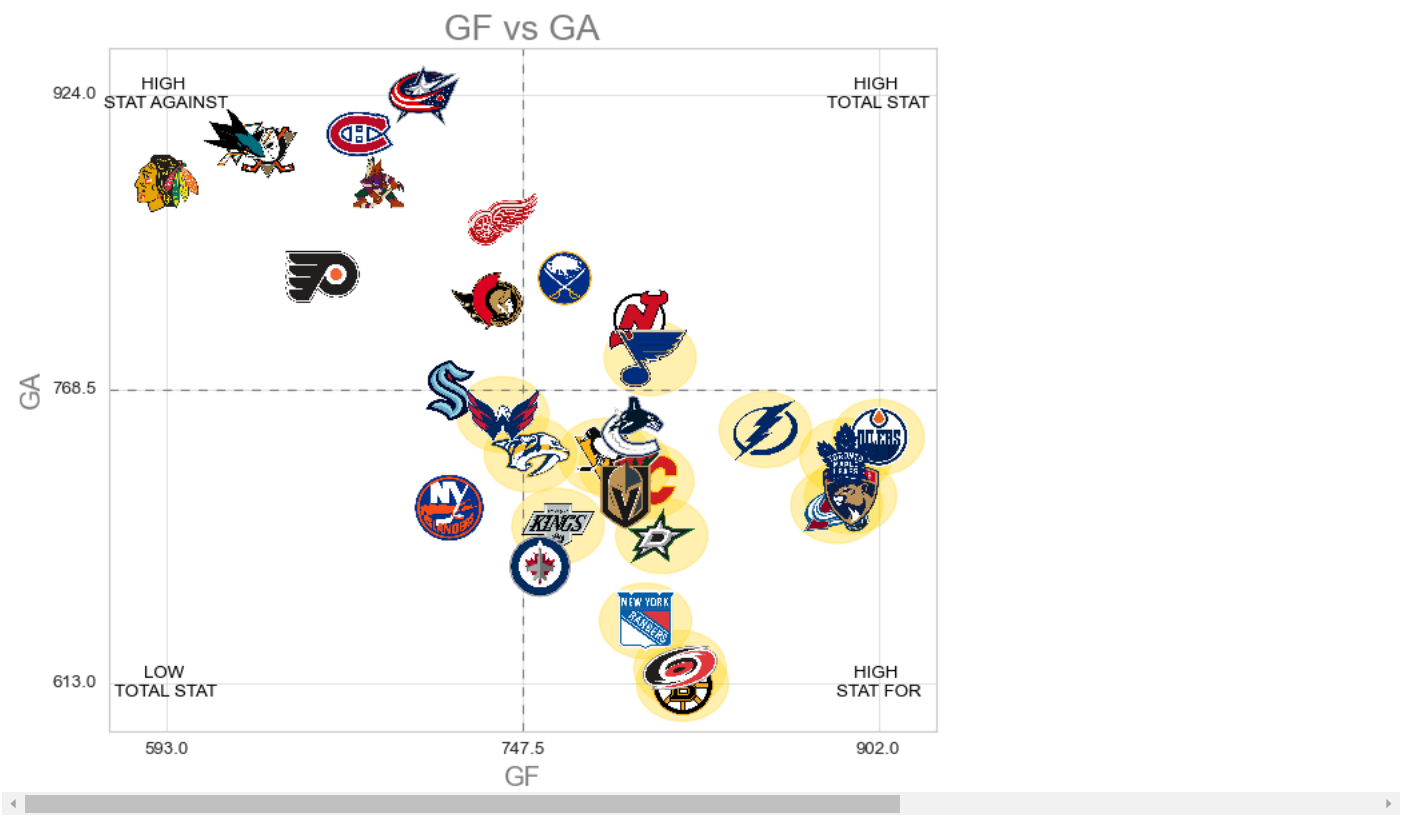
For this I got the totals from each of the previous 3 questions and graphed them against each other in a scatter plot

After creating the chart to be used to answer this question, I functionalized it for future use

```
def scatter(situation='all', sf='GF', sa='GA', logos=logos, year='all',save=False):
    if year == 'all':
        MASK_YEAR = (teams['season'] != 'all')
    else:
        MASK_YEAR = (teams['season'] == year)
    # Masking and creating data
    MASK_SIT = (teams['situation'] == situation)
    Q = teams[MASK_SIT & MASK_YEAR]
    SF = Q.groupby('team')[sf].sum()
    SA = Q.groupby('team')[sa].sum()
    abv = Q.groupby('team')['abv'].first()

    df = pd.DataFrame({
        'GF': SF,
        'GA': SA,
        'abv': abv
    }).reset_index()

    # Assuming Q7 DataFrame is already defined and has 'abv', 'GF', 'GA' columns
    df['logo'] = df['abv'].map(logos)

    # Convert all paths to strings and replace NaNs with a default or empty string
    df['logo'] = df['logo'].fillna('').astype(str)

    # Data from Q7 DataFrame
    abv = df['abv']
    goals_for = df['GF']
    goals_against = df['GA']

    min_goals_for = goals_for.min()
    max_goals_for = goals_for.max()
    min_goals_against = goals_against.min()
    max_goals_against = goals_against.max()
    mid_gf = (max_goals_for + min_goals_for) / 2
    mid_ga = (max_goals_against + min_goals_against) /2

    # Constants for image scaling
    OFFSET = 25
    IMAGE_SIZE = 0.075  # Adjust as needed

    width = max_goals_for - min_goals_for + 2 * OFFSET
    height = max_goals_against - min_goals_against + 2 * OFFSET

    fig, ax = plt.subplots(figsize=(10, 8))

    # Scatter plot
    ax.scatter(goals_for, goals_against)
    sns.set_style('whitegrid')

    # Style the plot
    ax.set_title(f'{sf} vs {sa}', fontsize=24, color='gray')
    ax.set_xlabel(f'{sf}', fontsize=18, color='gray')
    ax.set_ylabel(f'{sa}', fontsize=18, color='gray')
    ax.set_xticks([min_goals_for, mid_gf, max_goals_for])
    ax.set_yticks([min_goals_against, mid_ga, max_goals_against])
    ax.set_xlim(min_goals_for - OFFSET, max_goals_for + OFFSET)
    ax.set_ylim(min_goals_against - OFFSET, max_goals_against + OFFSET)
    ax.axvline(mid_gf, color='gray', linestyle='--')
    ax.axhline(mid_ga, color='gray', linestyle='--')

    # Adding labels for each quadrant
    ax.text(min_goals_for, max_goals_against, "HIGH \nSTAT AGAINST", color="black", va="center", ha="center")
    ax.text(min_goals_for, min_goals_against, "LOW \nTOTAL STAT", color="black", va="center", ha="center")
    ax.text(max_goals_for, max_goals_against, "HIGH \nTOTAL STAT", color="black", va="center", ha="center")
    ax.text(max_goals_for, min_goals_against, "HIGH \nSTAT FOR", color="black", va="center", ha="center")

    # Function to load images
    def get_image(path):
        if os.path.isfile(path):
            return OffsetImage(mpimg.imread(path), zoom=IMAGE_SIZE)
        else:
            print(f"Image path does not exist: {path}")
            return None

    # Add images to the plot
    for t, gf, ga, logo_path in zip(abv, goals_for, goals_against, df['logo']):
        img = get_image(logo_path)
```

```
        if img is not None:
            ab = AnnotationBbox(img, (gf, ga), frameon=False, pad=0.5)
            ax.add_artist(ab)
    if save == True:
        plt.savefig('scatterplot.png', transparent=True)
    plt.show()
```

```
scatter()
```

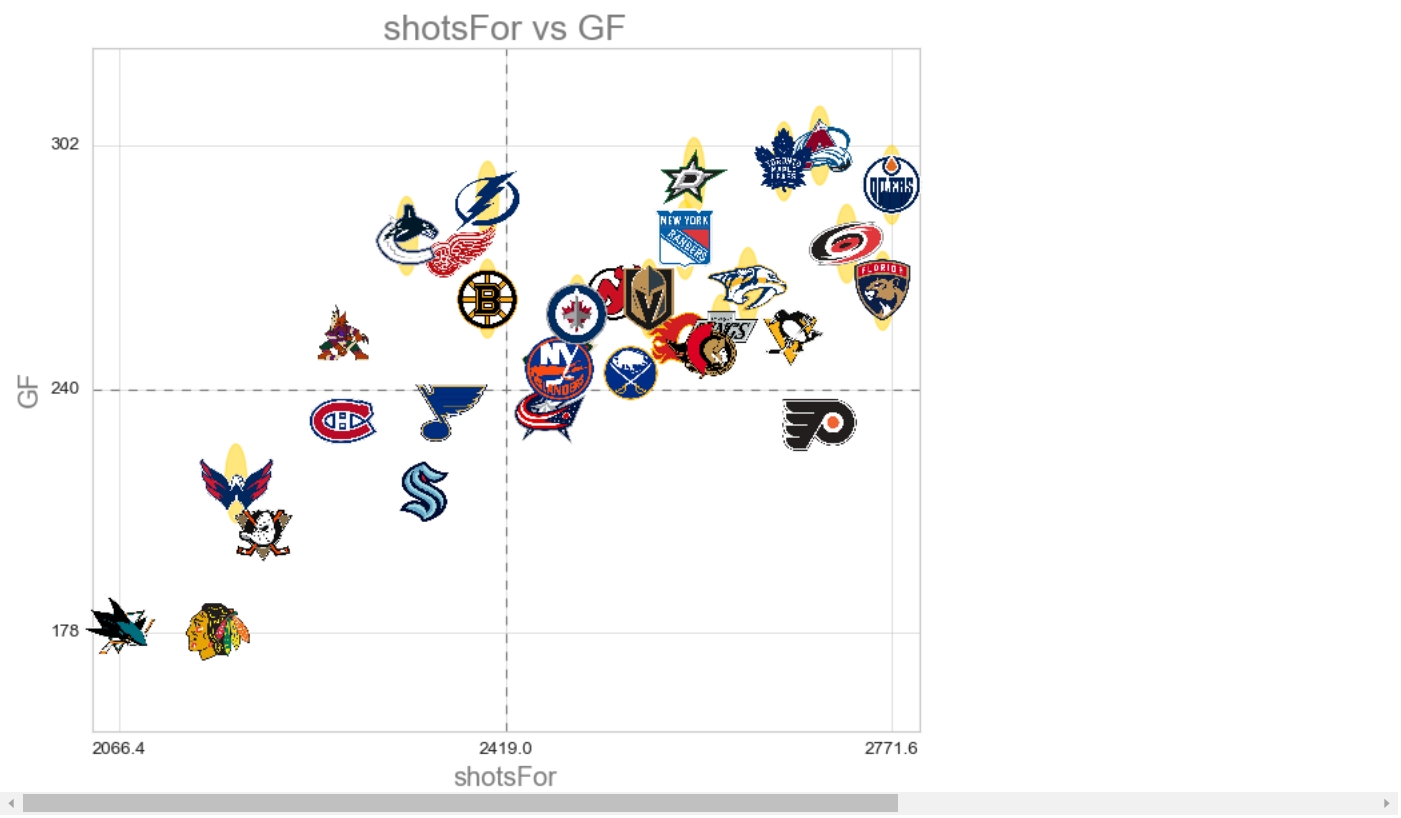


Descriptive Question 8+ : I wanted to see the relation between Goals For and Different Shot Types, so I plotted them in a scatter plot.

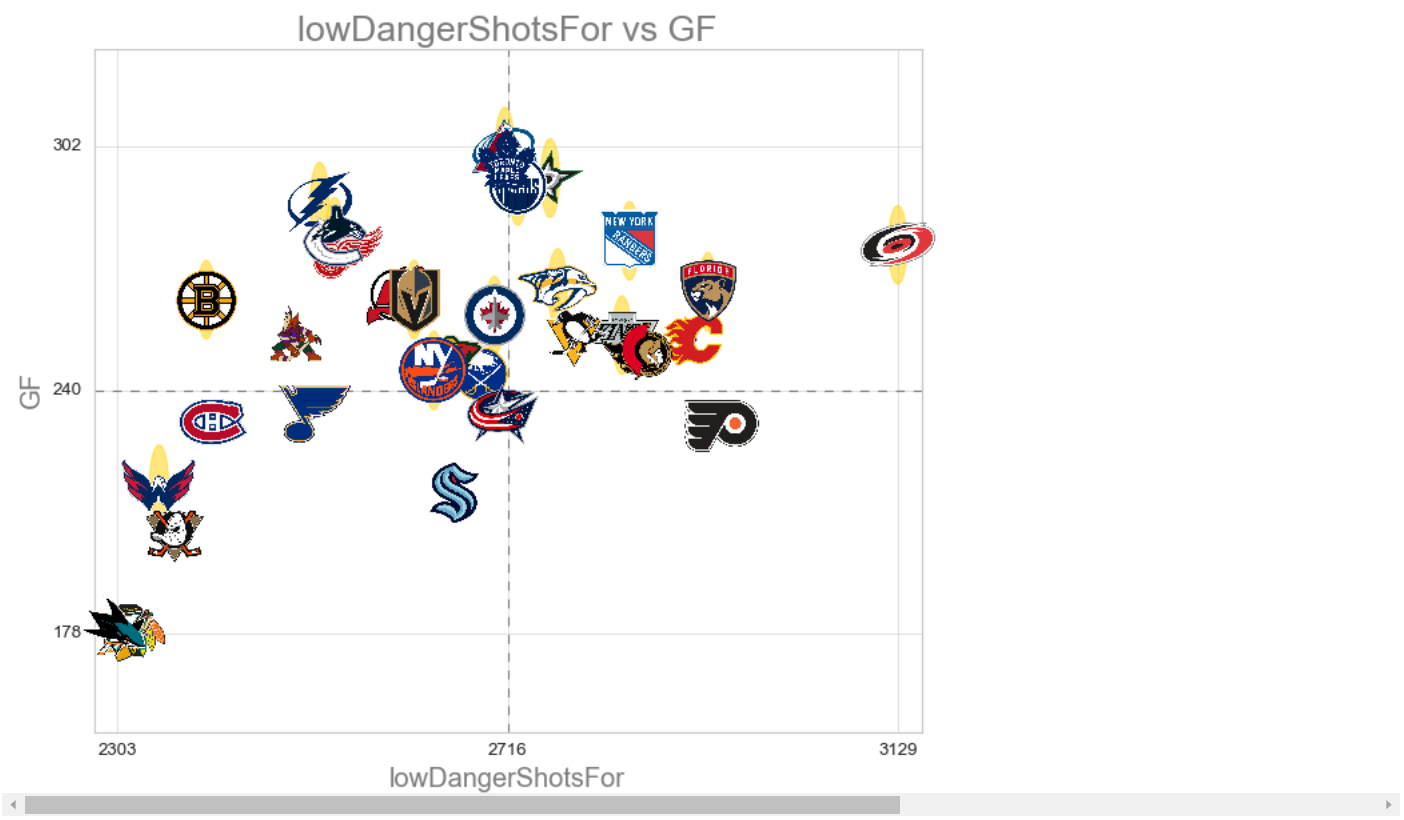
Here i recreated the previous function and added some specific features to highlight in specific graph, such as the playoff highlighter

```
def scatterspecific(situation='all', sf='GF', sa='GA', logos=logos, year='all', save=False):
    if year == 'all':
        MASK_YEAR = (teams['season'] != 'all')
    else:
        MASK_YEAR = (teams['season'] == year)

    # Masking and creating data
    MASK_SIT = (teams['situation'] == situation)
    Q = teams[MASK_SIT & MASK_YEAR]
    SF = Q.groupby('team')[sf].sum()
    SA = Q.groupby('team')[sa].sum()
    abv = Q.groupby('team')['abv'].first()
    playoff = Q.groupby('team')['playoffs'].first()

    df = pd.DataFrame({
        'GF': SF,
        'GA': SA,
        'abv': abv,
        'playoff': playoff
    }).reset_index()

    # Map logos to the teams
    df['logo'] = df['abv'].map(logos)
    df['logo'] = df['logo'].fillna('').astype(str)

    abv = df['abv']
    goals_for = df['GF']
    goals_against = df['GA']

    min_goals_for = goals_for.min()
    max_goals_for = goals_for.max()
    min_goals_against = goals_against.min()
    max_goals_against = goals_against.max()
    mid_gf = (max_goals_for + min_goals_for) / 2
    mid_ga = (max_goals_against + min_goals_against) /2

    # Constants for image scaling
    OFFSET = 25
    IMAGE_SIZE = 0.075  # Adjust as needed

    width = max_goals_for - min_goals_for + 2 * OFFSET
    height = max_goals_against - min_goals_against + 2 * OFFSET

    fig, ax = plt.subplots(figsize=(10, 8))

    # Scatter plot
    ax.scatter(goals_for, goals_against)
    sns.set_style('whitegrid')

    # Style the plot
    ax.set_title(f'{sf} vs {sa}', fontsize=24, color='gray')
    ax.set_xlabel(f'{sf}', fontsize=18, color='gray')
    ax.set_ylabel(f'{sa}', fontsize=18, color='gray')
    ax.set_xticks([min_goals_for, mid_gf, max_goals_for])
    ax.set_yticks([min_goals_against, mid_ga, max_goals_against])
    ax.set_xlim(min_goals_for - OFFSET, max_goals_for + OFFSET)
    ax.set_ylim(min_goals_against - OFFSET, max_goals_against + OFFSET)
    ax.axvline(mid_gf, color='gray', linestyle='--')
    ax.axhline(mid_ga, color='gray', linestyle='--')

    # # Adding labels for each quadrant
    # ax.text(min_goals_for, max_goals_against, "HIGH \nSTAT AGAINST", color="black", va="center", ha="center")
    # ax.text(min_goals_for, min_goals_against, "LOW \nTOTAL STAT", color="black", va="center", ha="center")
    # ax.text(max_goals_for, max_goals_against, "HIGH \nTOTAL STAT", color="black", va="center", ha="center")
    # ax.text(max_goals_for, min_goals_against, "HIGH \nSTAT FOR", color="black", va="center", ha="center")

    # Function to load images
    def get_image(path):
        if os.path.isfile(path):
            return OffsetImage(mpimg.imread(path), zoom=IMAGE_SIZE)
        else:
            print(f"Image path does not exist: {path}")
            return None

    # Highlight playoff teams with a gold circle
    for t, gf, ga, is_playoff in zip(abv, goals_for, goals_against, df['playoff']):
        if is_playoff == 1:
```

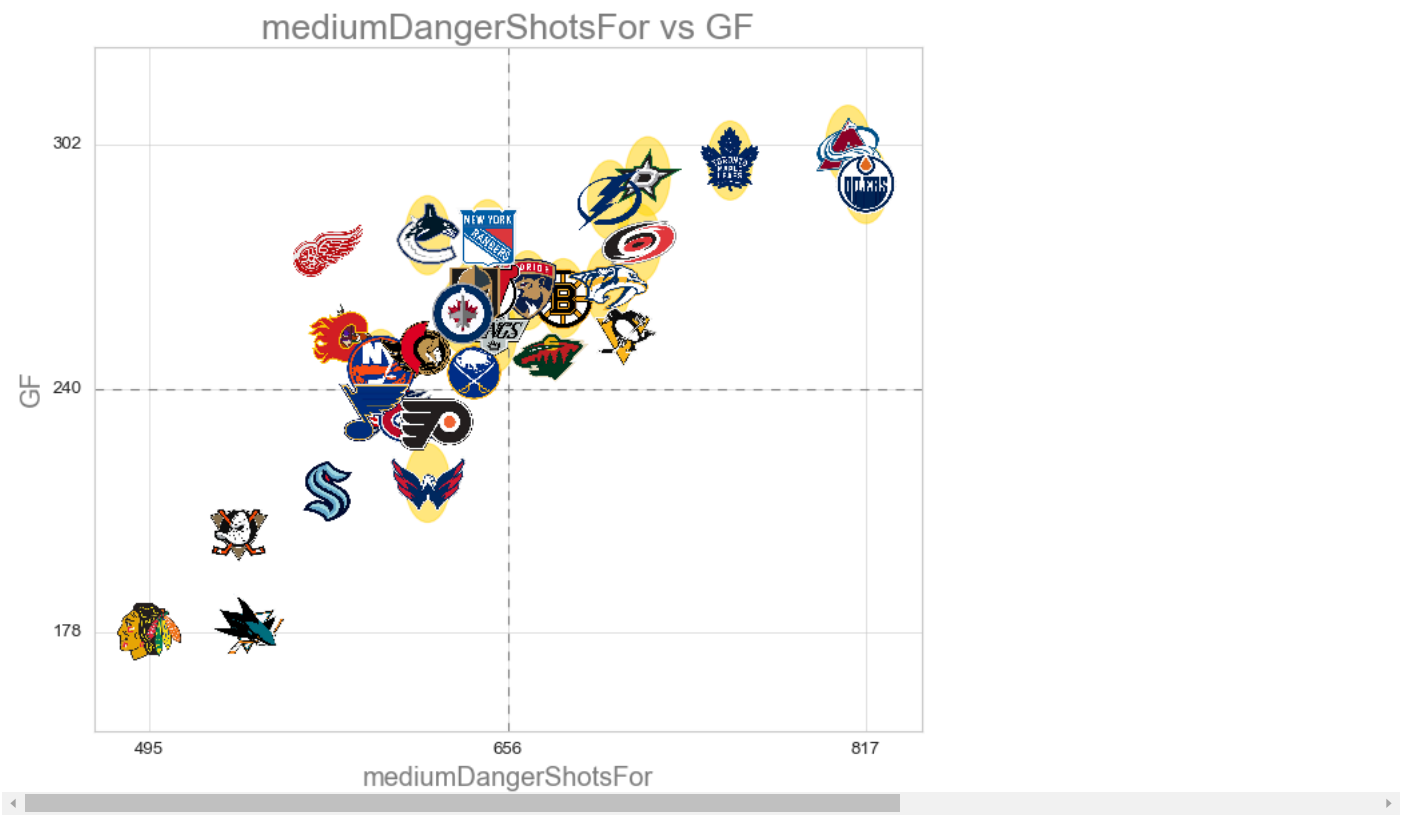
```
            ax.add_patch(plt.Circle((gf, ga), 10, color='gold', alpha=0.5))

    # Add images to the plot
    for t, gf, ga, logo_path in zip(abv, goals_for, goals_against, df['logo']):
        img = get_image(logo_path)
        if img is not None:
            ab = AnnotationBbox(img, (gf, ga), frameon=False, pad=0.5)
            ax.add_artist(ab)

    if save:
        plt.savefig(f'scatterplot{sf}.png', transparent=True)
    plt.show()


# scatter plot between shots for and goals for
scatterspecific(sf='shotsFor', sa='GF',year=20232024)
```



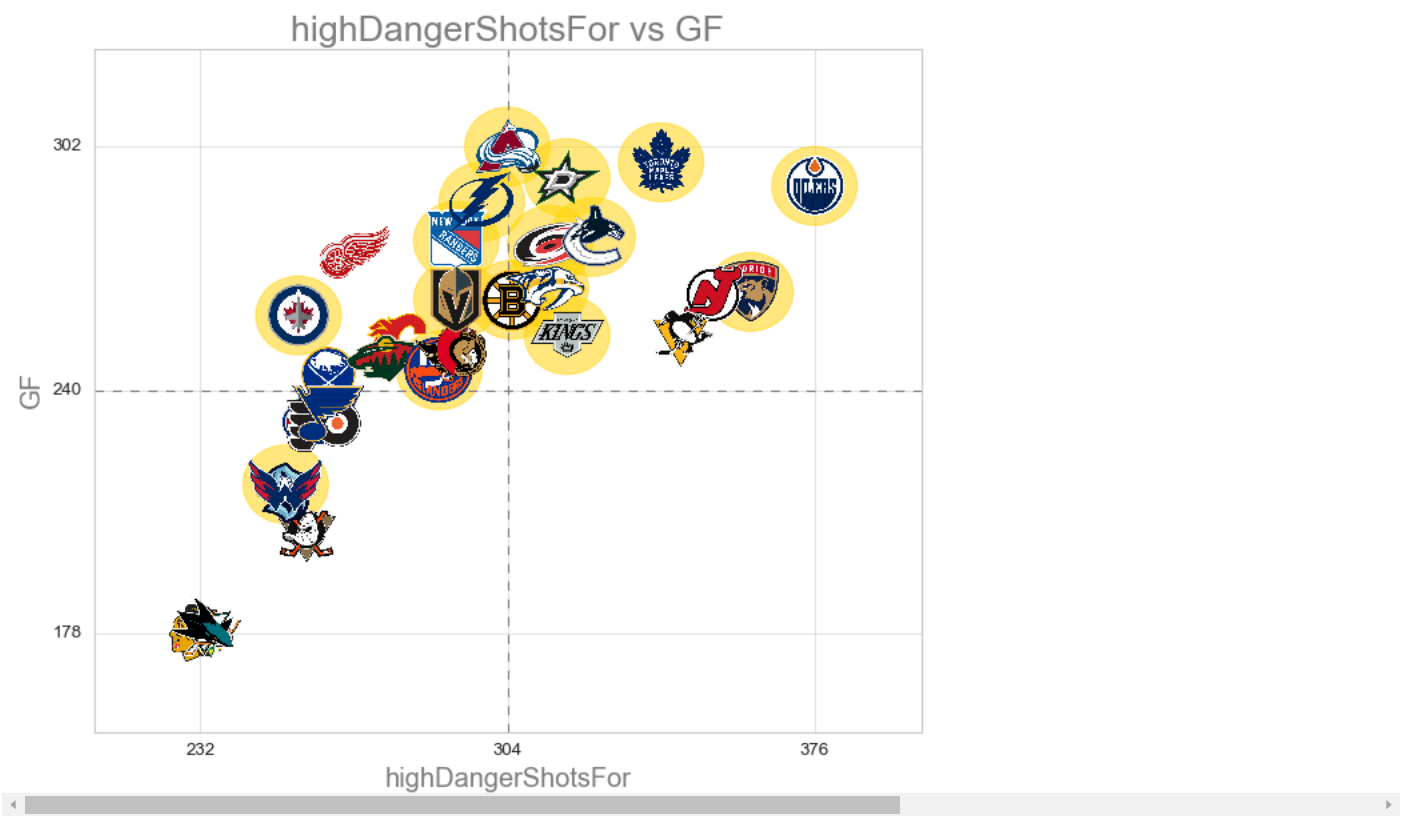

```
# scatter plot between low danger shots for and goals for
scatterspecific(sf='lowDangerShotsFor', sa='GF',year=20232024)
```



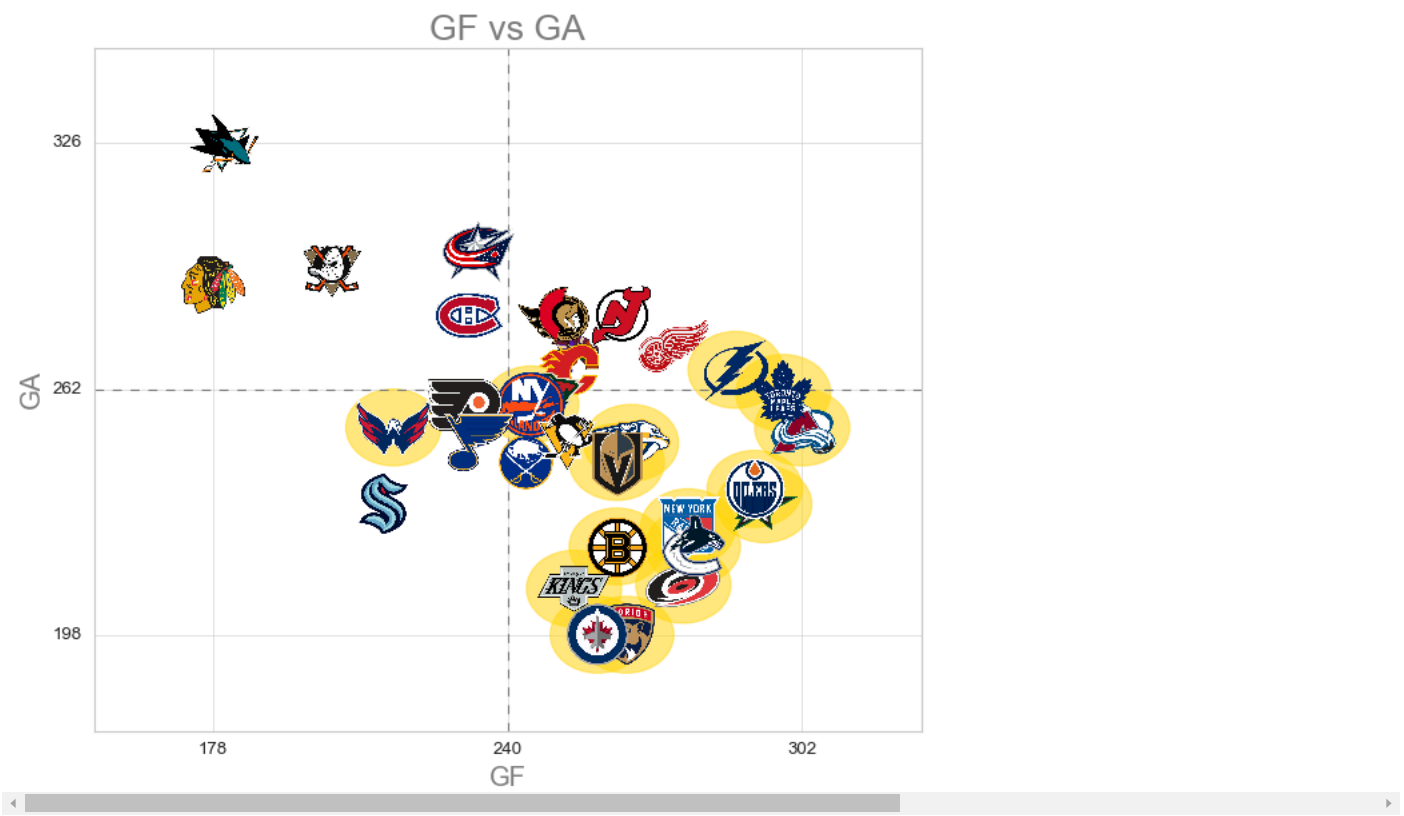

```
# scatter plot between medium danger shots for and goals for
scatterspecific(sf='mediumDangerShotsFor', sa='GF',year=20232024)
```




```
# scatter plot between high danger shots for and goals for
scatterspecific(sf='highDangerShotsFor', sa='GF',year=20232024)
```




```
# scatter plot between goals for and goals against in the 2023-24 season with the playoff teams highlighed in yellow
scatterspecific(sf='GF', sa='GA',year=20232024)
```



Inferential Questions

Inferential Question 1: Does a change in Division have any relation to a change in average points at the end of the season?

Null Hypothesis (H_0): μ_{ATL} = μ_{MET} = μ_{CEN} = μ_{PAC}

Alternate Hypothesis (H_A): $\mu_{ATL} \neq \mu_{MET} \neq \mu_{CEN} \neq \mu_{PAC}$

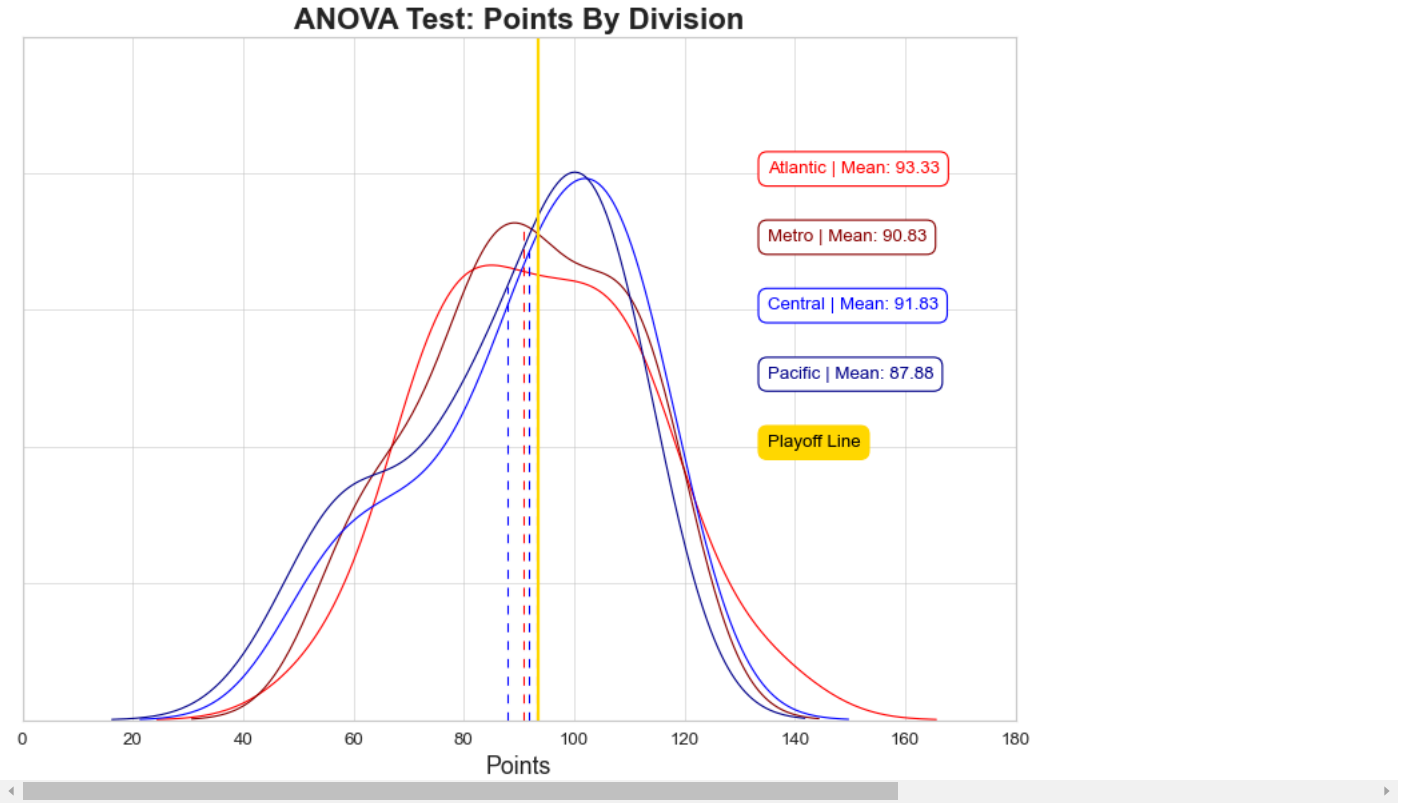
where μ is the mean points at the end of each season

```
H1_ATL = teams[MASK_ALL & MASK_ATL]['P']
H1_MET = teams[MASK_ALL & MASK_MET]['P']
H1_CEN = teams[MASK_ALL & MASK_CEN]['P']
H1_PAC = teams[MASK_ALL & MASK_PAC]['P']
```

```
# Calculate the means of all datasets
mean1 = np.mean(H1_ATL)
mean2 = np.mean(H1_MET)
mean3 = np.mean(H1_CEN)
mean4 = np.mean(H1_PAC)

# Perform one-way ANOVA test
f_stat, p_value = stats.f_oneway(H1_ATL, H1_MET, H1_CEN, H1_PAC)
if p_value < 0.05:
    print('Reject the Null Hypothesis')
else:
    print('Fail to reject the null hypothesis')

# Compute KDE values
kde_atl = stats.gaussian_kde(H1_ATL)
kde_met = stats.gaussian_kde(H1_MET)
kde_cen = stats.gaussian_kde(H1_CEN)
kde_pac = stats.gaussian_kde(H1_PAC)

threshold = 93.5

# Calculate the probability for each division
prob_atl = 1 - kde_atl.integrate_box_1d(threshold, np.inf)
prob_met = 1 - kde_met.integrate_box_1d(threshold, np.inf)
prob_cen = 1 - kde_cen.integrate_box_1d(threshold, np.inf)
prob_pac = 1 - kde_pac.integrate_box_1d(threshold, np.inf)

print(f'Probability of making the playoffs for Atlantic: {prob_atl:.4f}')
print(f'Probability of making the playoffs for Metro: {prob_met:.4f}')
print(f'Probability of making the playoffs for Central: {prob_cen:.4f}')
print(f'Probability of making the playoffs for Pacific: {prob_pac:.4f}')

# Use a style template
sns.set_style('whitegrid')

plt.figure(figsize=(12, 8))

# Plot all datasets
sns.kdeplot(H1_ATL, color='red', alpha=1)
sns.kdeplot(H1_MET, color='darkred', alpha=1)
sns.kdeplot(H1_CEN, color='blue', alpha=1)
sns.kdeplot(H1_PAC, color='darkblue', alpha=1)

# Generate x values for KDE
x_vals = np.linspace(min(min(H1_ATL), min(H1_MET), min(H1_CEN), min(H1_PAC)),
                     max(max(H1_ATL), max(H1_MET), max(H1_CEN), max(H1_PAC)), 1000)

# Fill the entire area under the curves
plt.fill_between(x_vals, kde_atl(x_vals), color='red', alpha=0)
plt.fill_between(x_vals, kde_met(x_vals), color='darkred', alpha=0)
plt.fill_between(x_vals, kde_cen(x_vals), color='blue', alpha=0)
plt.fill_between(x_vals, kde_pac(x_vals), color='darkblue', alpha=0)

# Get KDE values at means
kde_atl_at_mean1 = kde_atl(mean1)[0]
kde_met_at_mean2 = kde_met(mean2)[0]
kde_cen_at_mean3 = kde_cen(mean3)[0]
kde_pac_at_mean4 = kde_pac(mean4)[0]

# Add vertical lines at the means, stopping at the KDE curves
plt.plot([mean1, mean1], [0, kde_atl_at_mean1], 'r--', label=f'Atlantic Mean: {mean1:.2f}')
plt.plot([mean2, mean2], [0, kde_met_at_mean2], 'r--', label=f'Metro Mean: {mean2:.2f}')
plt.plot([mean3, mean3], [0, kde_cen_at_mean3], 'b--', label=f'Central Mean: {mean3:.2f}')
plt.plot([mean4, mean4], [0, kde_pac_at_mean4], 'b--', label=f'Pacific Mean: {mean4:.2f}')

# Add a gold vertical line at 93.5
plt.axvline(93.5, color='gold', linestyle='-', linewidth=2, label='Playoff Line')

# Add labels and title
plt.title('ANOVA Test: Points By Division', fontsize=20, fontweight='bold')
plt.xlabel('Points', fontsize=16)
plt.ylabel('')
plt.yticks(visible=False)

# Add annotations
plt.annotate(f'Atlantic | Mean: {mean1:.2f}', xy=(0.75, 0.8), xycoords='axes fraction', color='red', fontsize=12,
             bbox=dict(facecolor='white', edgecolor='red', boxstyle='round,pad=0.5'))
```

```
plt.annotate(f'Metro | Mean: {mean2:.2f}', xy=(0.75, 0.7), xycoords='axes fraction', color='darkred', fontsize=12,
             bbox=dict(facecolor='white', edgecolor='darkred', boxstyle='round,pad=0.5'))
plt.annotate(f"Central | Mean: {mean3:.2f}", xy=(0.75, 0.6), xycoords='axes fraction', color='blue', fontsize=12,
             bbox=dict(facecolor='white', edgecolor='blue', boxstyle='round,pad=0.5'))
plt.annotate(f'Pacific | Mean: {mean4:.2f}', xy=(0.75, 0.5), xycoords='axes fraction', color='darkblue', fontsize=12,
             bbox=dict(facecolor='white', edgecolor='darkblue', boxstyle='round,pad=0.5'))
plt.annotate('Playoff Line', xy=(0.75, 0.4), xycoords='axes fraction', color='black', fontsize=12,
             bbox=dict(facecolor='gold', edgecolor='gold', boxstyle='round,pad=0.5'))

# Save the plot
plt.savefig('ANOVAPointsByDivision.png', transparent=True)
plt.show()
```

```
Fail to reject the null hypothesis
Probability of making the playoffs for Atlantic: 0.5100
Probability of making the playoffs for Metro: 0.5356
Probability of making the playoffs for Central: 0.4737
Probability of making the playoffs for Pacific: 0.5395
```



Inferential Question 2: Does a change in Conference have any relation to a change in average points at the end of the season?

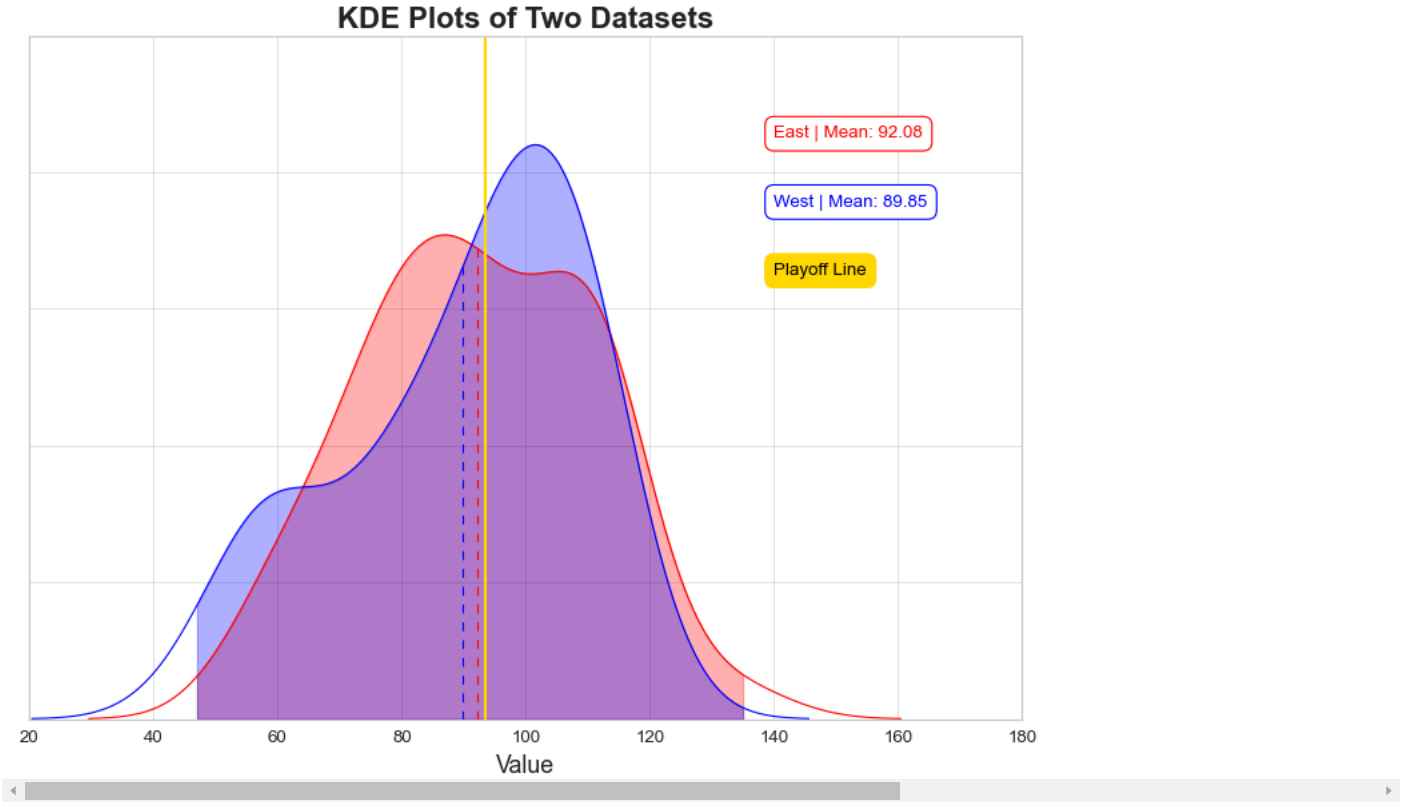
Null Hypothesis (H_0): $\mu_{EAST} = \mu_{WEST}$

Alternate Hypothesis (H_A): $\mu_{EAST} \neq \mu_{WEST}$

```
H1_EAST = teams[MASK_ALL & MASK_EAST]['P']
H1_WEST = teams[MASK_ALL & MASK_WEST]['P']
```

```
two_sample_kdeplot(H1_EAST, H1_WEST, label2='West', label1='East', save=True)
```

```
Fail to reject the null hypothesis
Probability of making the playoffs for East: 0.5241
Probability of making the playoffs for West: 0.5005
```



Inferential Question 3: Does a change in a Team's Power Play percentage have a realtion with a Team's place in the standings at the end of the season?

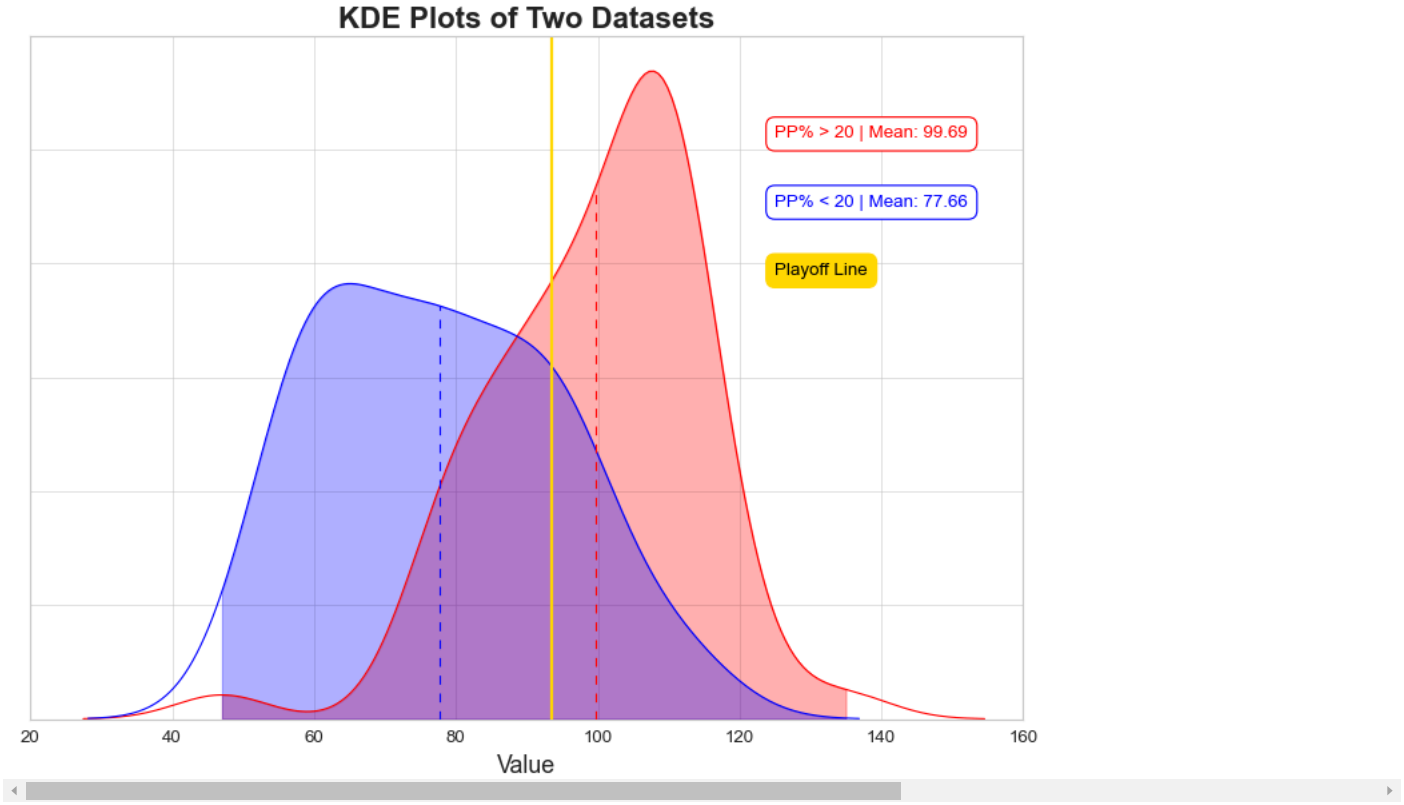
Null Hypothesis (H_0): $\mu_{PP\%>20} = \mu_{PP\%<20}$

Alternate Hypothesis (H_A): $\mu_{PP\%>20} \neq \mu_{PP\%<20}$

A Power Play is when your team has a man-advantage on the ice. This will happen when the other team takes a penalty. A Power Play percentage (PP%) is how often a team scores when they have the man-advantage

```
MASK_PPG = teams['PP%'] >= 20
MASK_PPB = teams['PP%'] < 20


H2_PPG = teams[MASK_ALL & MASK_PPG]['P']
H2_PPB = teams[MASK_ALL & MASK_PPB]['P']


two_sample_kdeplot(H2_PPG, H2_PPB, label1='PP% > 20' , label2= 'PP% < 20')
```

```
Reject the Null Hypothesis
Probability of making the playoffs for PP% > 20: 0.3227
Probability of making the playoffs for PP% < 20: 0.7884
```



Inferential Question 4: Does a change in a Team's Penalty Kill percentage have a relation a Team's place in the standings at the end of the season?

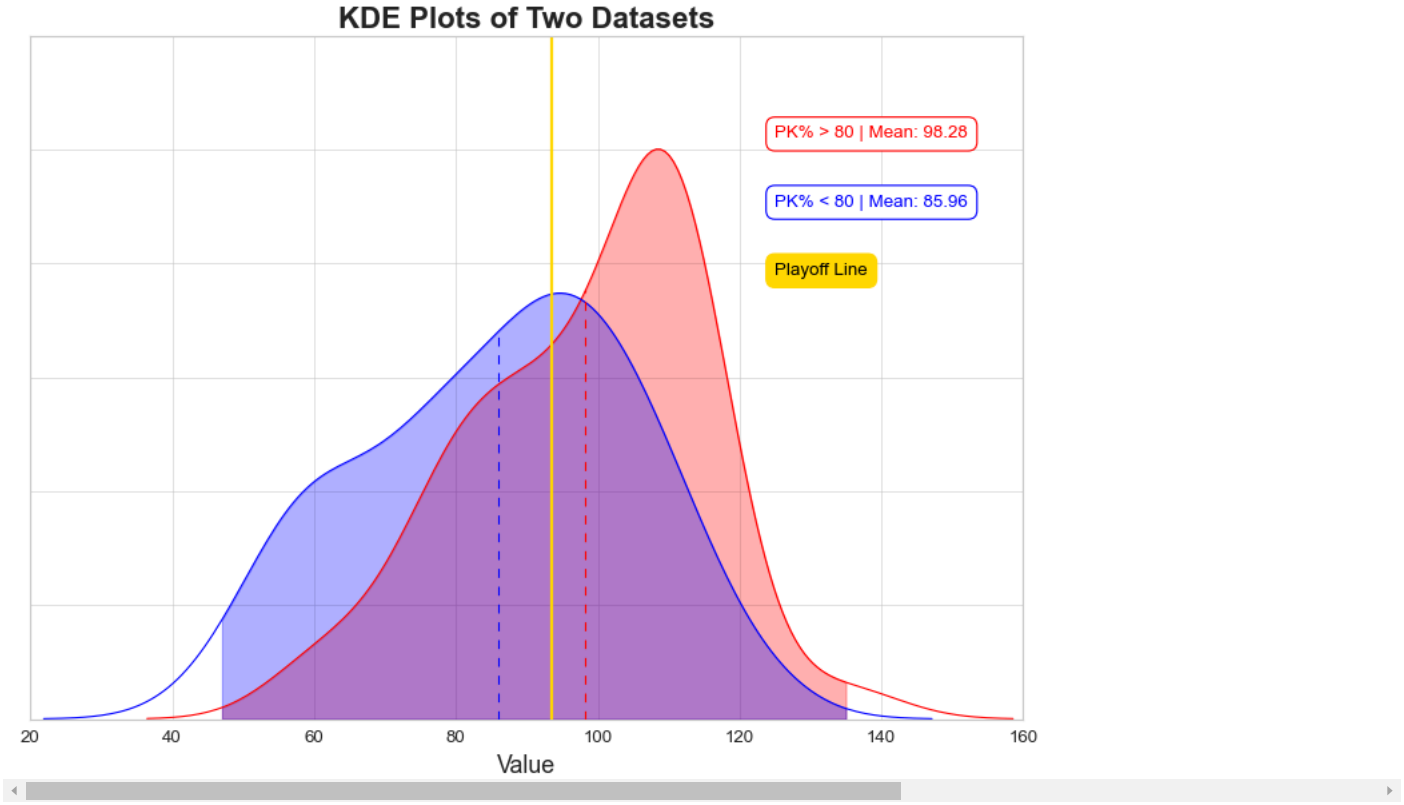
Null Hypothesis (H_0): $\mu_{PK\%>80} = \mu_{PK\%<80}$

Alternate Hypothesis (H_A): $\mu_{PK\%>80} \neq \mu_{PK\%<80}$

A Penalty Kill is when you successfull stop the opposing team from scoring when they have the man advantage (i.e. after your team took the penalty). A Penalty Kill percentage is the amount of time you finish the penalty without giving up a goal divided by the total amount of penalties taken.

```
MASK_PKG = teams['PK%'] >= 80
MASK_PKB = teams['PK%'] < 80


H2_PKG = teams[MASK_ALL & MASK_PKG]['P']
H2_PKB = teams[MASK_ALL & MASK_PKB]['P']


two_sample_kdeplot(H2_PKG, H2_PKB, label1='PK% > 80' , label2= 'PK% < 80')
```

```
Reject the Null Hypothesis
Probability of making the playoffs for PK% > 80: 0.3687
Probability of making the playoffs for PK% < 80: 0.6078
```



Inferential Question 5: Does a change in a Team's overall Special Teams quality have a relation with a Team's place in the standings at the end of the season?

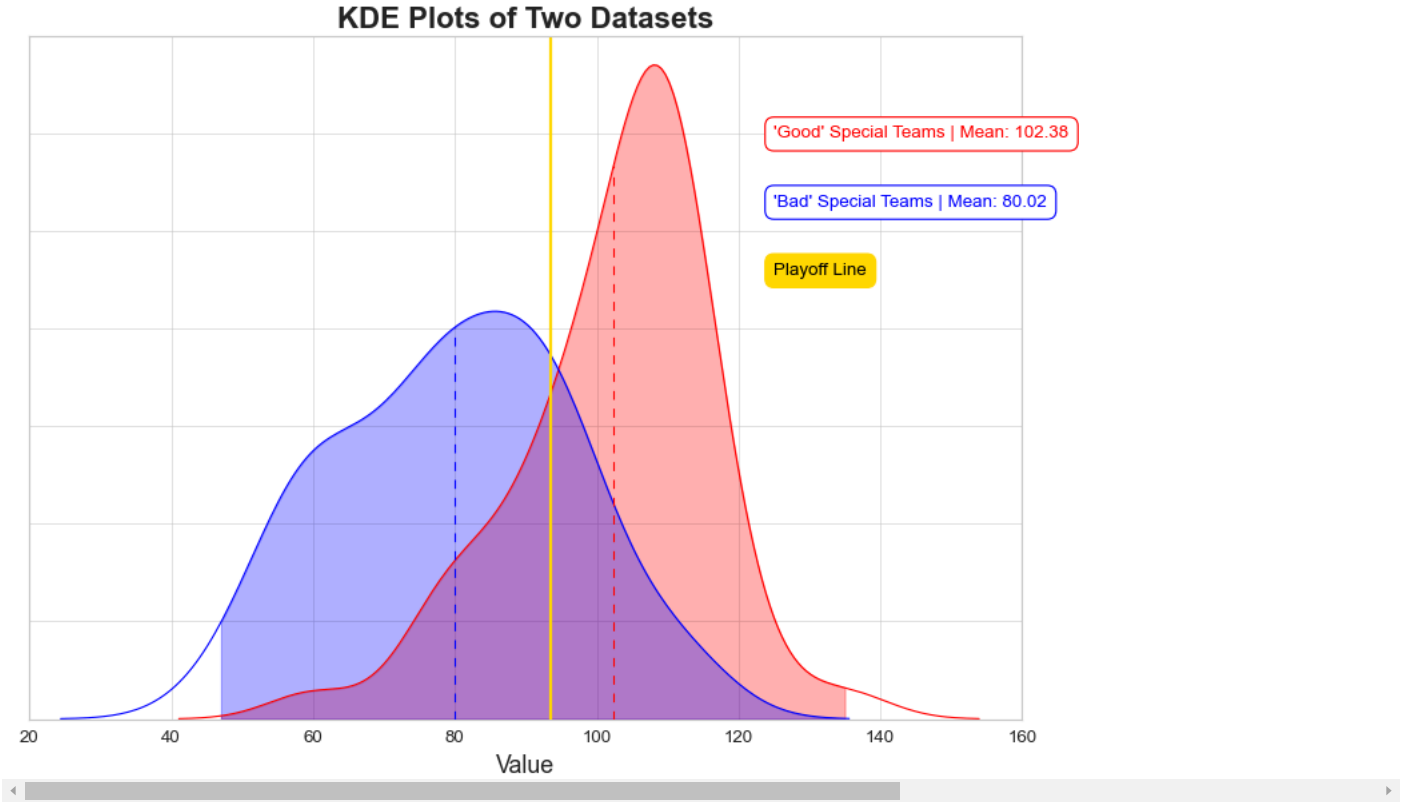
Null Hypothesis (H_0): $\mu_{SP>100} = \mu_{SP<100}$

Alternate Hypothesis (H_A): $\mu_{SP>100} \neq \mu_{SP<100}$

A team's overall special teams quality is determined by a combination of a team's PP% and PK%. When added together, a Team with a Special Teams quality of over 100 is considered good while under is considered bad

```
MASK_SPG = teams['special'] >= 100
MASK_SPB = teams['special'] < 100


H2_SPG = teams[MASK_ALL & MASK_SPG]['P']
H2_SPB = teams[MASK_ALL & MASK_SPB]['P']


two_sample_kdeplot(H2_SPG, H2_SPB, label1= "'Good' Special Teams", label2= "'Bad' Special Teams")
```

```
Reject the Null Hypothesis
Probability of making the playoffs for 'Good' Special Teams: 0.2402
Probability of making the playoffs for 'Bad' Special Teams: 0.7601
```



Inferential Question 6+ : There are a number of percentage stats that are calculated using (`Stat` For)/ (`Stat` For + `Stat` Against), For each of `Stat`, does having a positive share of that `Stat` have any relation to a Team's place in the standings at the end of the season?

Null Hypothesis (H_0): $\mu_{`Stat`\%>50}$ = $\mu_{`Stat`\%<50}$

Alternate Hypothesis (H_A): $\mu_{`Stat`\%>50} \neq \mu_{`Stat`\%<50}$

this function is very similar to the kdeplot two sample function but its key difference is that it automatically creates all masks necessary to perfom the hypothesis tests

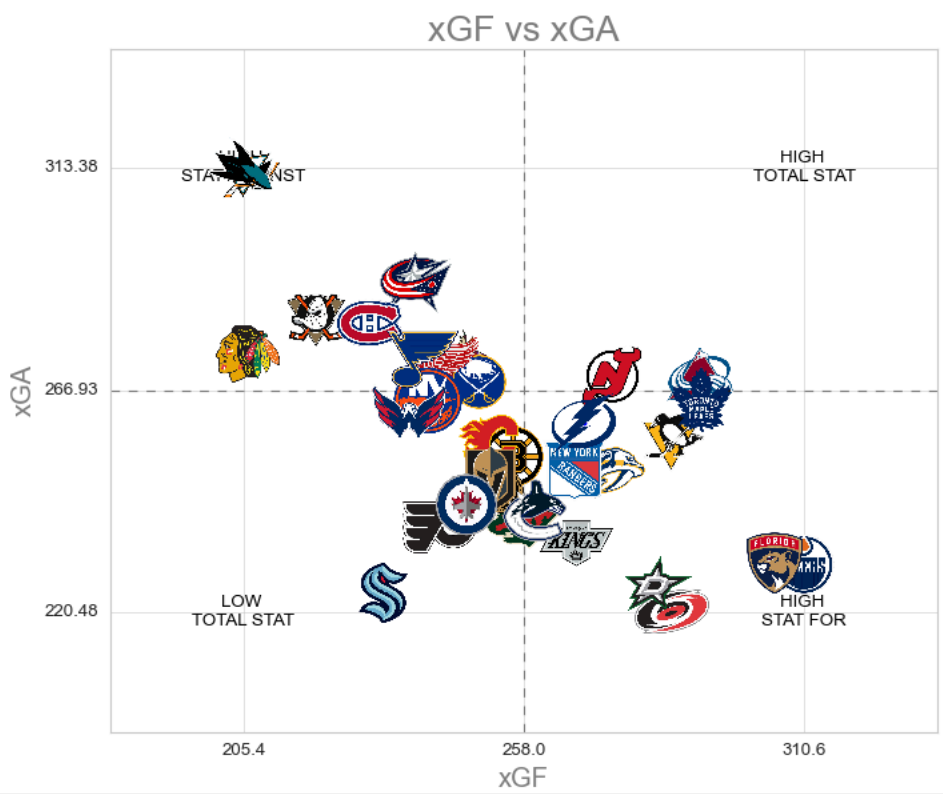
```
def two_sample_5050_test(df=teams, situation='all', stat='xG%', save=False):
    # Masking and creating data
    MASK_50P = df[stat] >= .5
    MASK_50M = df[stat] < .5
    MASK_SIT = (df['situation'] == situation)
    good = (df[MASK_SIT & MASK_50P]['P'])
    bad = (df[MASK_SIT & MASK_50M]['P'])

    if len(good) == 0 or len(bad) == 0:
        print('Check input')
        return

    # Running two sample t-test and printing result
    _, p_value = stats.ttest_ind(good, bad)
    if p_value < 0.05:
        print('Reject the Null Hypothesis')
    else:
        print('Fail to reject the null hypothesis')

    # Creating label for KDE Plot
    label1 = (f'{stat} > 50%')
    label2 = (f'{stat} < 50%')

    # Calculate the means of both datasets
    mean1 = np.mean(good)
    mean2 = np.mean(bad)
```

```
    mean2 = np.mean(bad)

    # Use a style template
    sns.set_style('whitegrid')

    plt.figure(figsize=(12, 8))

    # Plot both datasets
    sns.kdeplot(good, label=label1, color='red', alpha=1)
    sns.kdeplot(bad, label=label2, color='blue', alpha=1)

    # Compute KDE values
    kde_good = stats.gaussian_kde(good)
    kde_bad = stats.gaussian_kde(bad)

    threshold = 93.5

    # Calculate the probability for 'good' group
    prob_good = 1 - kde_good.integrate_box_1d(threshold, np.inf)

    # Calculate the probability for 'bad' group
    prob_bad = 1 - kde_bad.integrate_box_1d(threshold, np.inf)

    print(f'Probability of making the playoffs when {stat} > 50%: {prob_good:.4f}')
    print(f'Probability of making the playoffs when {stat} < 50%: {prob_bad:.4f}')

    # Generate x values for KDE
    x_vals = np.linspace(min(good.min(), bad.min()), max(good.max(), bad.max()), 1000)

    # Fill the entire area under the curves
    plt.fill_between(x_vals, kde_good(x_vals), color='red', alpha=0.3)
    plt.fill_between(x_vals, kde_bad(x_vals), color='blue', alpha=0.3)

    # Get KDE values at means
    kde_good_at_mean1 = kde_good(mean1)[0]
    kde_bad_at_mean2 = kde_bad(mean2)[0]

    # Add vertical lines at the means, stopping at the KDE curves
    plt.plot([mean1, mean1], [0, kde_good_at_mean1], 'r--', label=f'{label1} Mean: {mean1:.2f}')
    plt.plot([mean2, mean2], [0, kde_bad_at_mean2], 'b--', label=f'{label2} Mean: {mean2:.2f}')

    # Add a gold vertical line at 93.5
    plt.axvline(93.5, color='gold', linestyle='-', linewidth=2, label='Playoff Line')

    # Add annotations
    plt.annotate(f'{label1} | Mean: {mean1:.2f}', xy=(0.75, 0.85), xycoords='axes fraction', color='red', fontsize=12,
                 bbox=dict(facecolor='white', edgecolor='red', boxstyle='round,pad=0.5'))
    plt.annotate(f'{label2} | Mean: {mean2:.2f}', xy=(0.75, 0.75), xycoords='axes fraction', color='blue', fontsize=12,
                 bbox=dict(facecolor='white', edgecolor='blue', boxstyle='round,pad=0.5'))
    plt.annotate('Playoff Line', xy=(0.75, 0.65), xycoords='axes fraction', color='gold', fontsize=12,
                 bbox=dict(facecolor='white', edgecolor='gold', boxstyle='round,pad=0.5'))

    plt.title(f'Two Sample T-Test: ({stat} > 50%) vs ({stat} < 50%)', fontsize=20, fontweight='bold')
    plt.xlabel('Points', fontsize=16)
    plt.ylabel('')
    plt.yticks(visible=False)
    if save is True:
        plt.savefig(f'Two Sample T-Test {stat}.png')

    plt.show()
```

Expected Goals

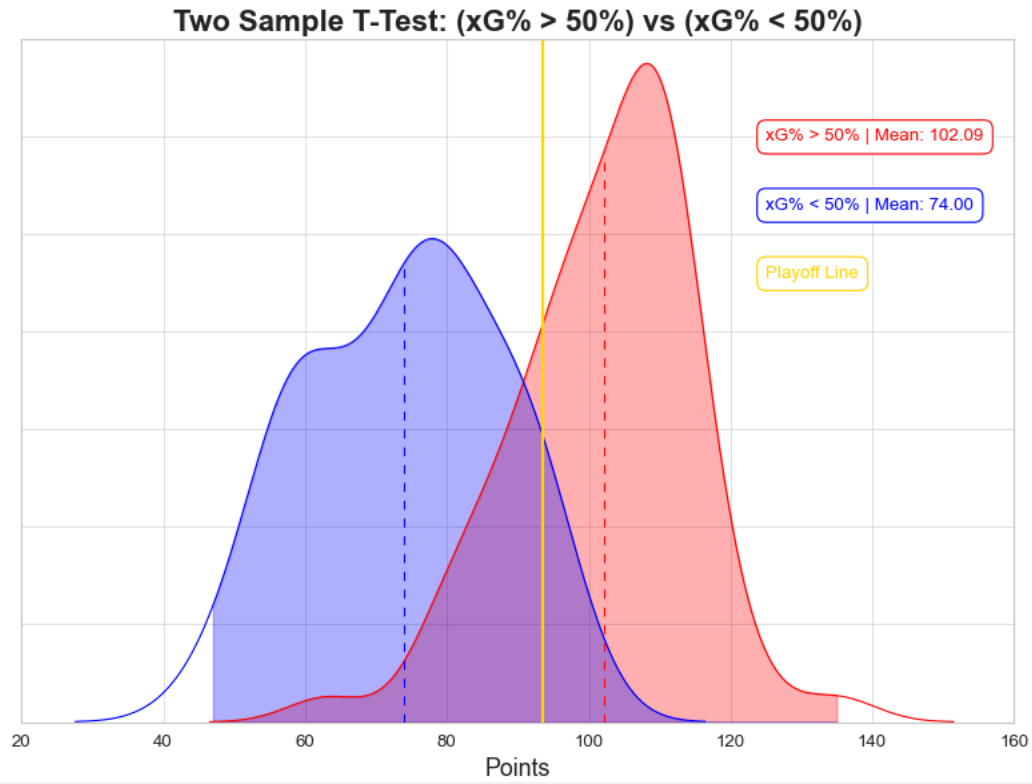
An Expected Goal (xGoals), is the percent of the time each shot or opportunity results in a goal.

```
# a scatter play of the following test for Expected Goals during this past season
scatter(sf='xGF', sa='xGA', year=20232024)
```




```
# Hypothesis test for Expected Goals Percentage (xG%)
two_sample_5050_test(save=True)
```

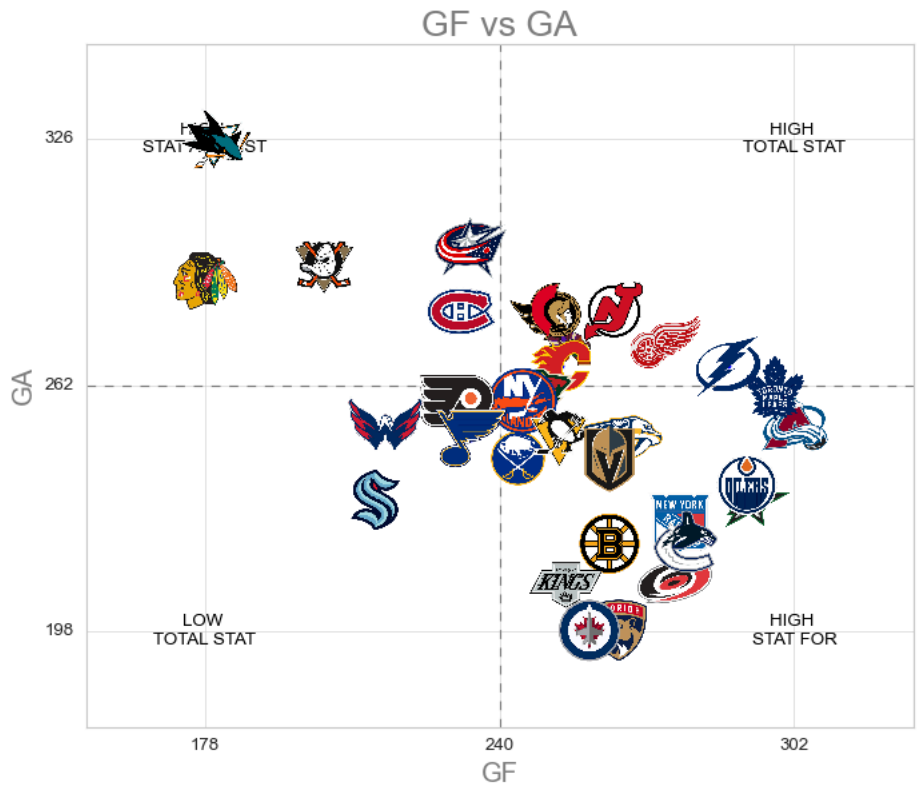
```
Reject the Null Hypothesis
Probability of making the playoffs when xG% > 50%: 0.2433
Probability of making the playoffs when xG% < 50%: 0.9038
```



Goals

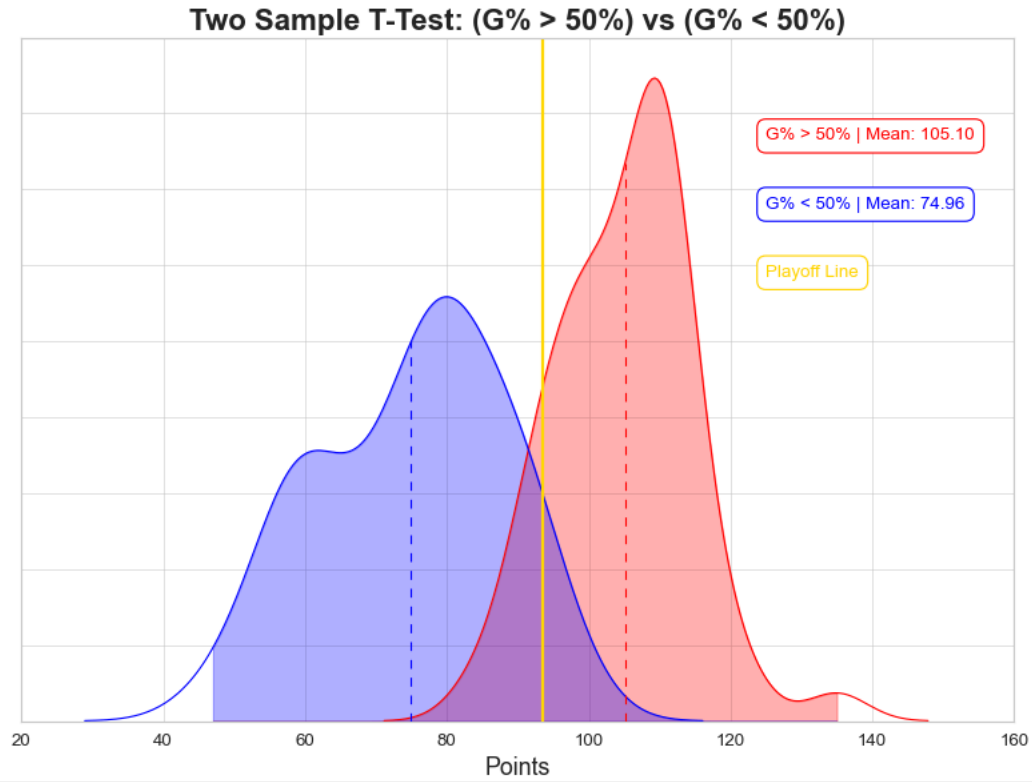
A Goal is scored when your team succefully gets the puck in the net in a legal fashion

```
# a scatter plot of the following test for Goals during this past season
scatter(sf='GF', sa='GA', year=20232024)
```



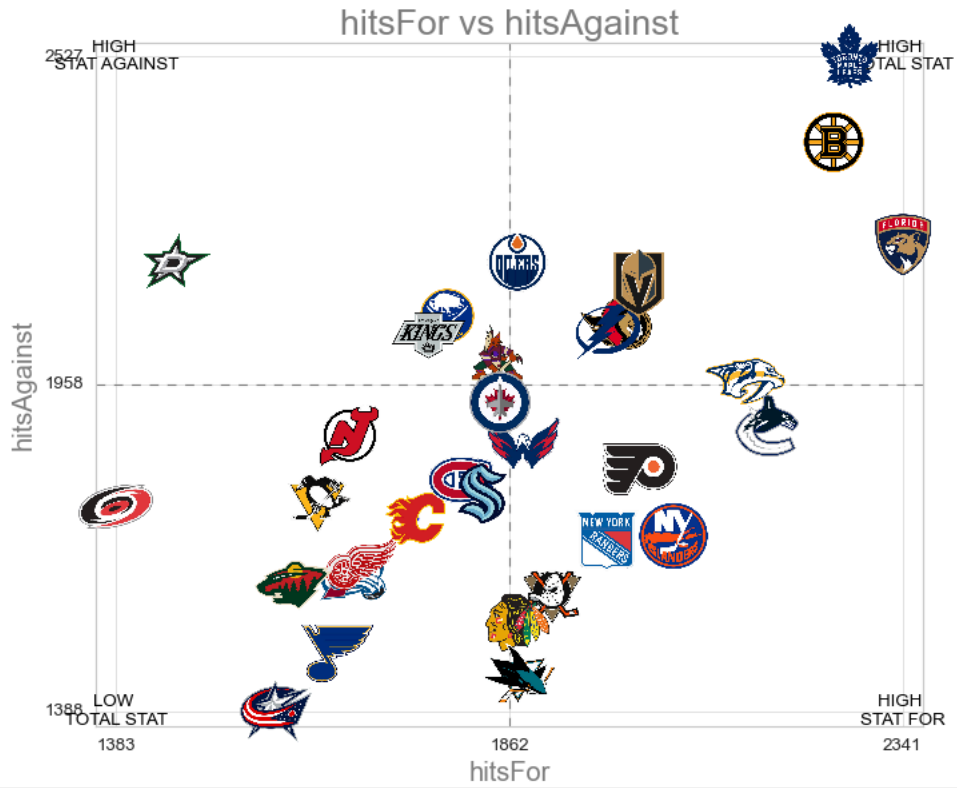

```
# Hypothesis test for Goals Percentage (G%)
two_sample_5050_test(stat='G%')
```

```
Reject the Null Hypothesis
Probability of making the playoffs when G% > 50%: 0.1362
Probability of making the playoffs when G% < 50%: 0.9116
```



Hits

```
# a scatter plot of the following test for hits during this past season
scatter(sf='hitsFor', sa='hitsAgainst', year=20232024)
```

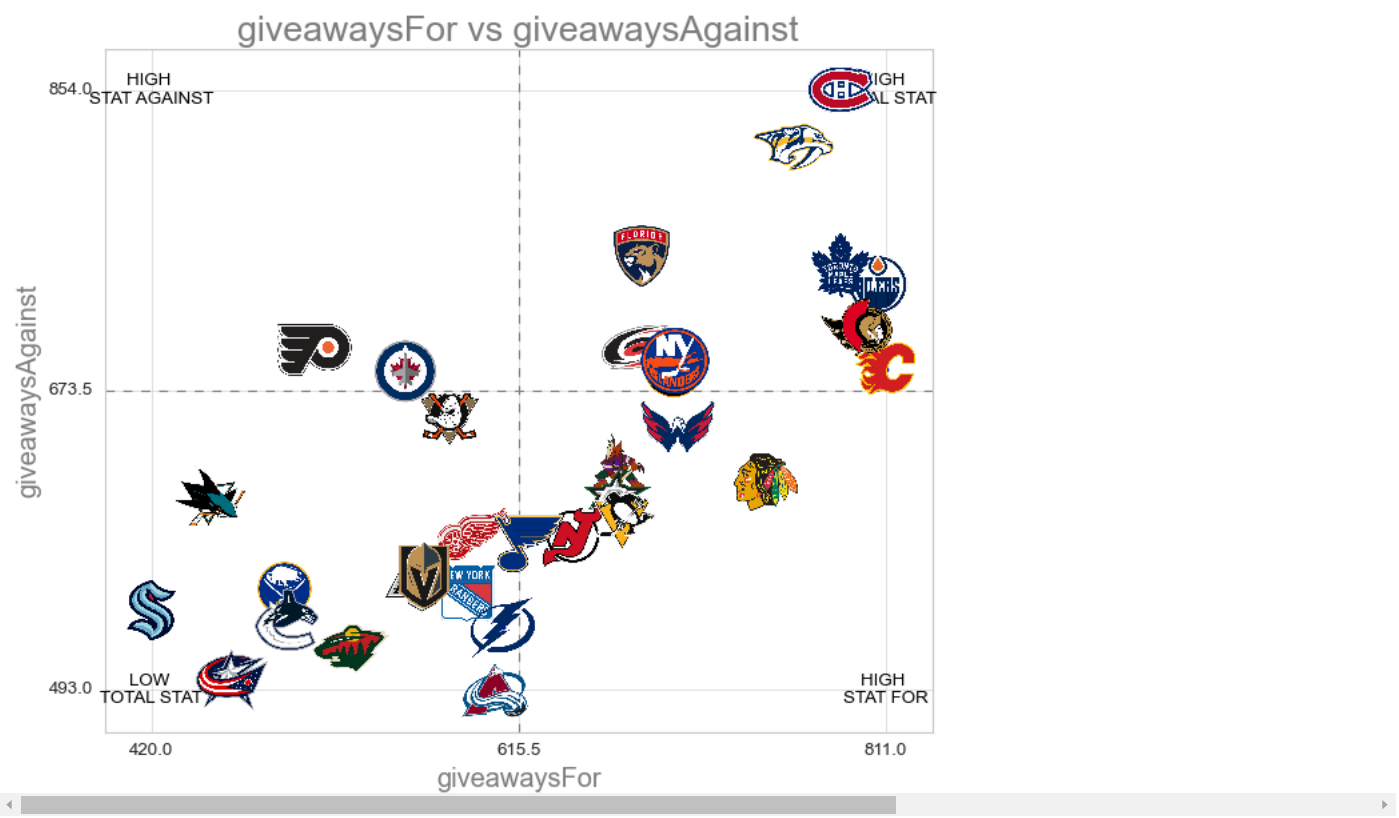



```
# Hypothesis test for hit Percentage (hit%)
two_sample_5050_test(stat='hit%')
```

```
Reject the Null Hypothesis
Probability of making the playoffs when hit% > 50%: 0.6164
Probability of making the playoffs when hit% < 50%: 0.3663
```

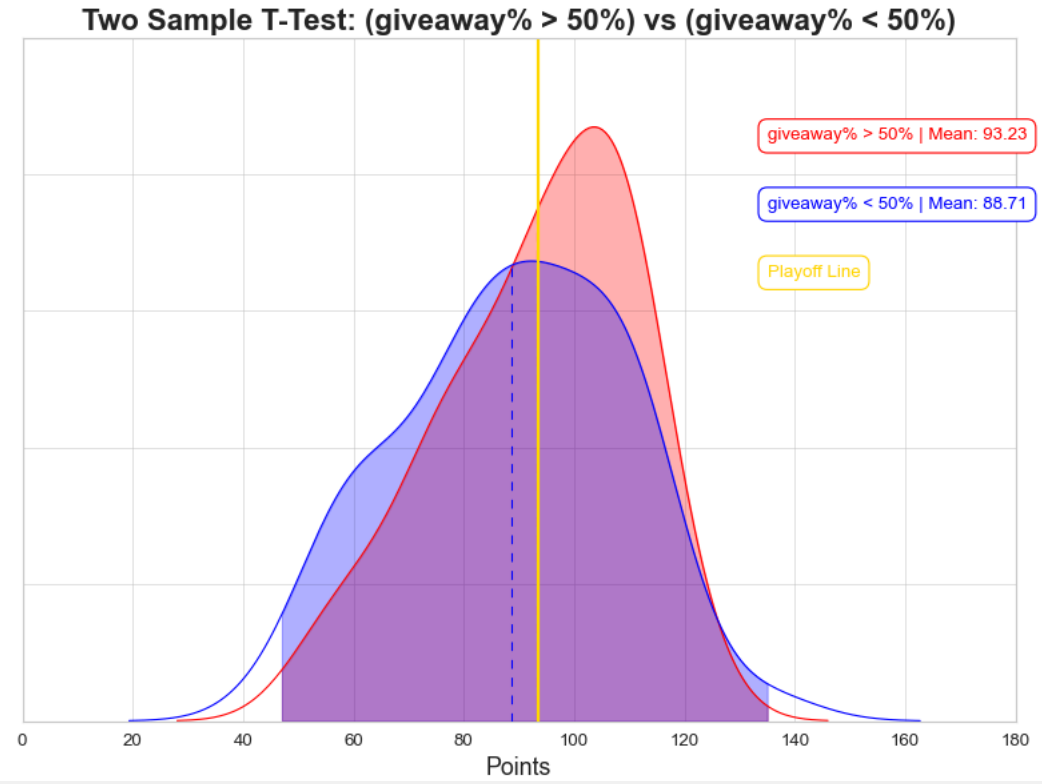
Giveaways

```
# a scatter plot of the following test for giveaways during this past season
scatter(sf='giveawaysFor', sa='giveawaysAgainst', year=20232024)
```



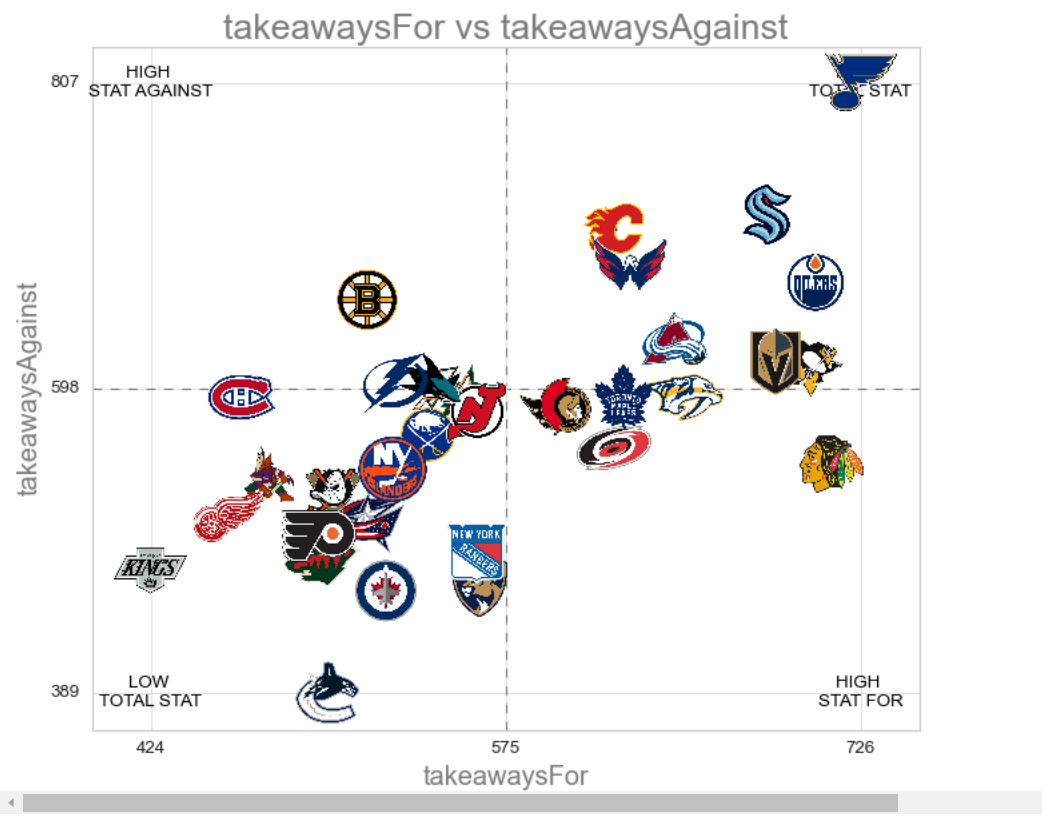

```
# Hypothesis test for Giveaway Percentage (giveaway%)
two_sample_5050_test(stat='giveaway%')
```

```
Fail to reject the null hypothesis
Probability of making the playoffs when giveaway% > 50%: 0.4583
Probability of making the playoffs when giveaway% < 50%: 0.5625
```



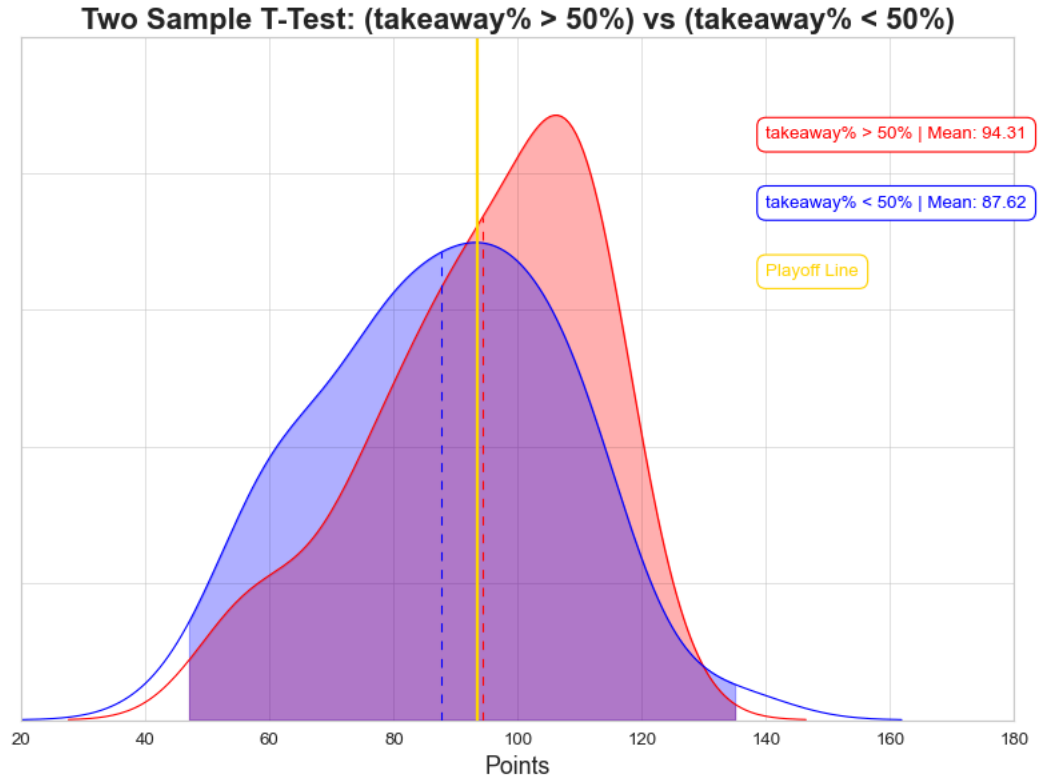
Takeaways

```
# a scatter plot of the following test for takeaways during this past season
scatter(sf='takeawaysFor', sa='takeawaysAgainst', year=20232024)
```



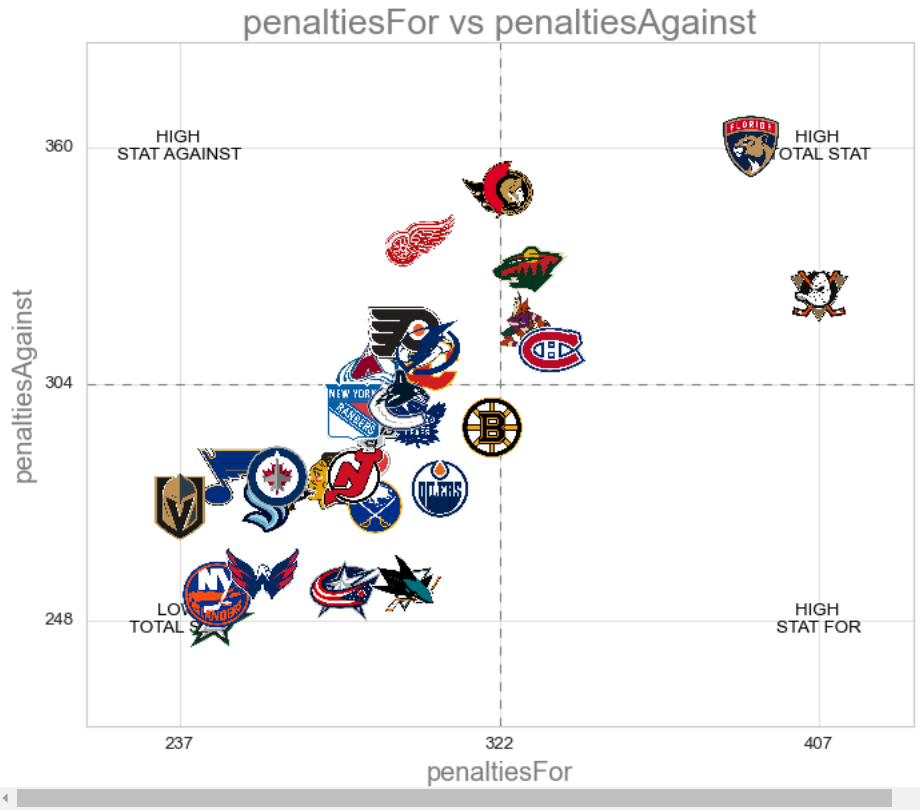
```
# Hypothesis test for Takeaway Percentage (takeaway%)
two_sample_5050_test(stat='takeaway%')
```

```
Fail to reject the null hypothesis
Probability of making the playoffs when takeaway% > 50%: 0.4320
Probability of making the playoffs when takeaway% < 50%: 0.5897
```



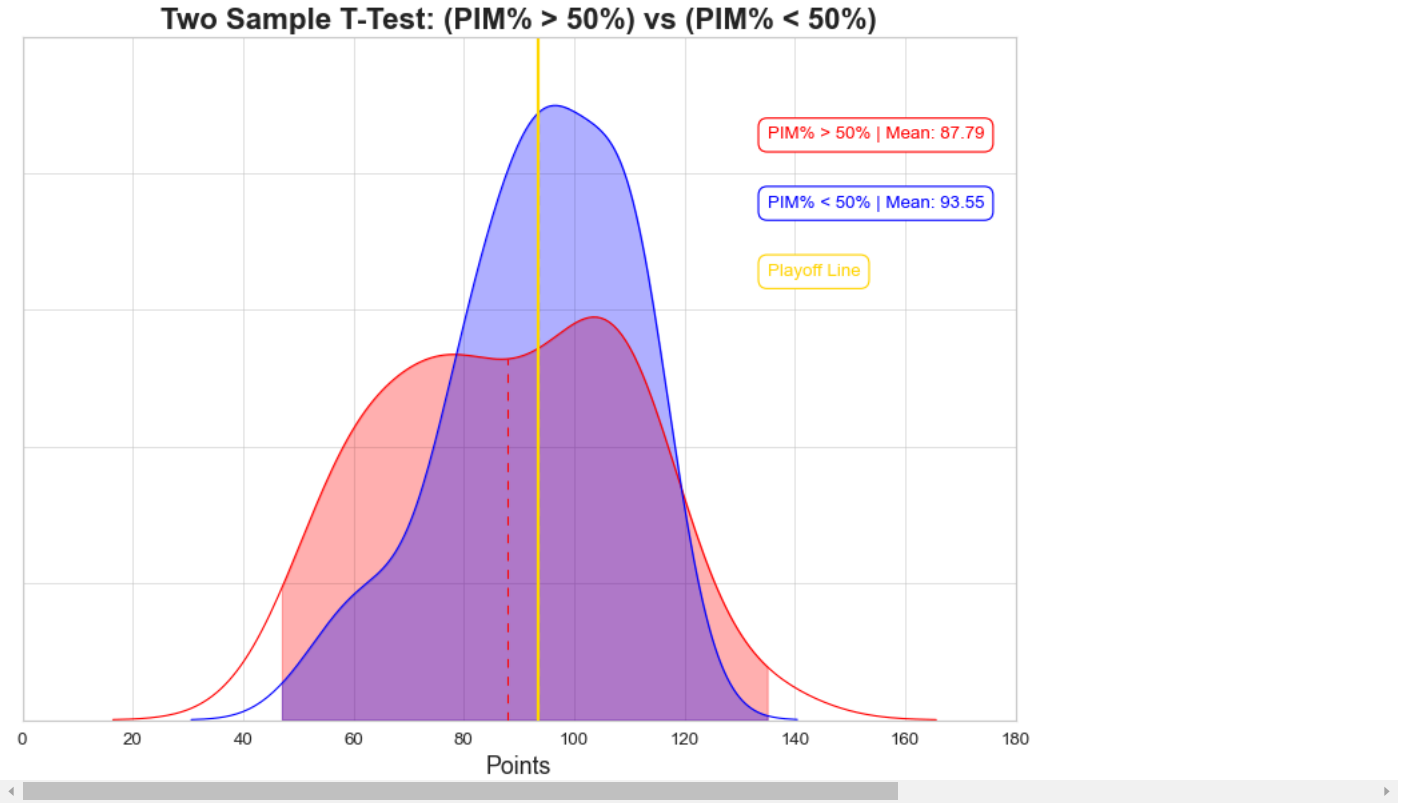
Penalties

```
# a scatter plot of the following test for Penalties in Minutes during this past season
scatter(sf='penaltiesFor', sa='penaltiesAgainst', year=20232024)
```



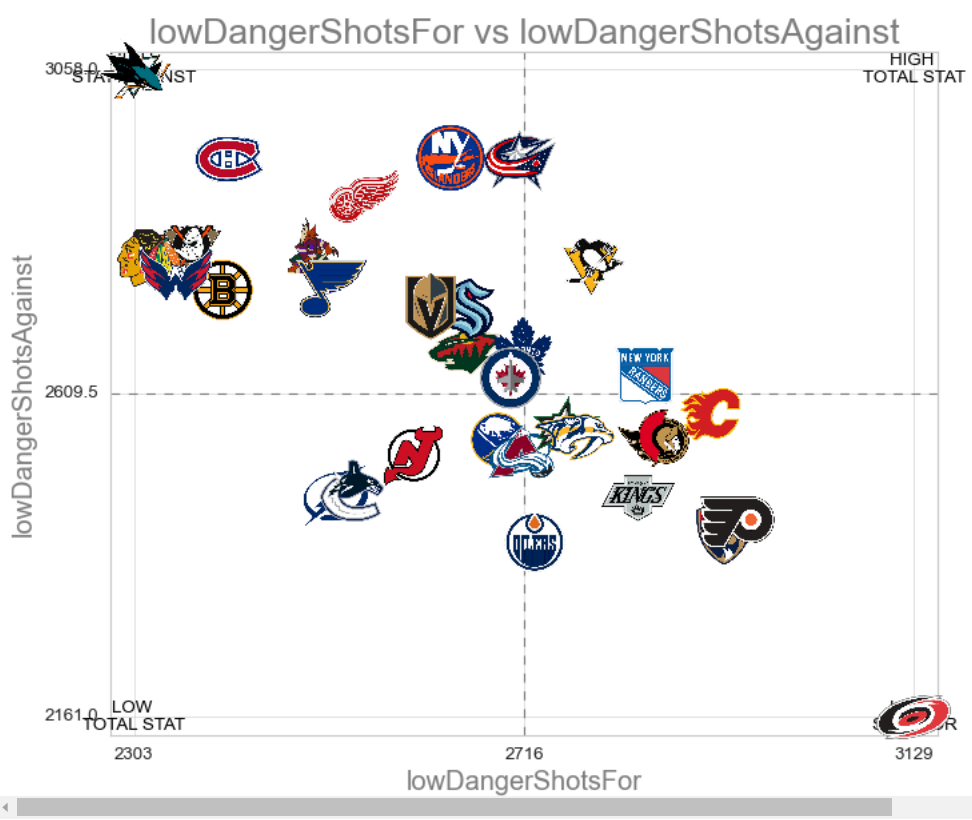
```
# Hypothesis test for Penalties In Minutes Percentage (PIM%)
two_sample_5050_test(stat='PIM%')
```

```
Fail to reject the null hypothesis
Probability of making the playoffs when PIM% > 50%: 0.5698
Probability of making the playoffs when PIM% < 50%: 0.4638
```



Low Danger Shots

```
# a scatter plot of the following test for Low Danger Shots during this past season
scatter(sf='lowDangerShotsFor', sa='lowDangerShotsAgainst', year=20232024)
```

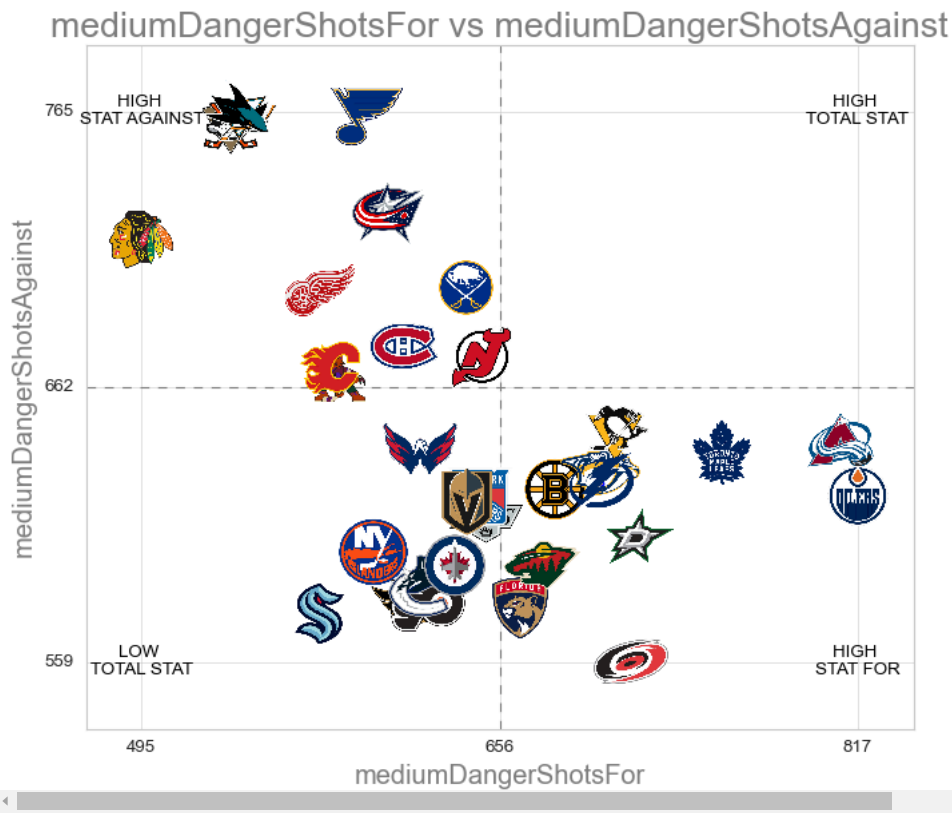



```
# Hypothesis test for Low Danger Shots Percentage (LDS%)
two_sample_5050_test(stat='LDS%')
```

```
Reject the Null Hypothesis
Probability of making the playoffs when LDS% > 50%: 0.2777
Probability of making the playoffs when LDS% < 50%: 0.7632
```

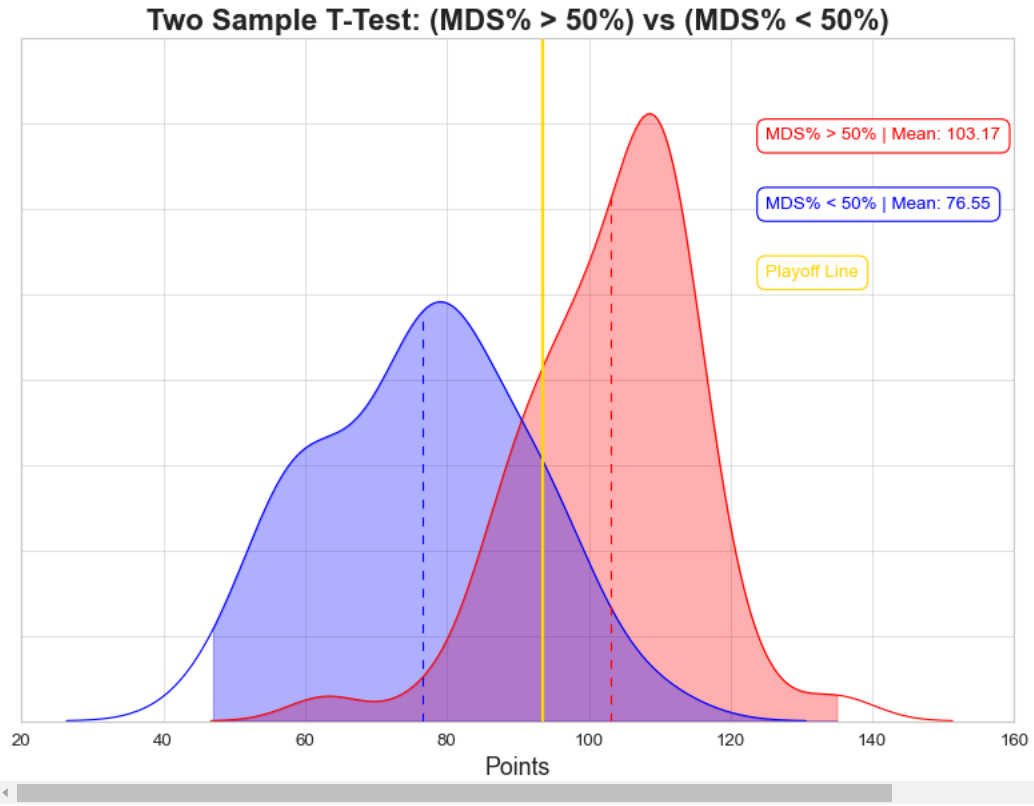
Medium Danger Shots

```
# a scatter plot of the following test for Medium Danger Shots during this past season
scatter(sf='mediumDangerShotsFor', sa='mediumDangerShotsAgainst', year=20232024)
```



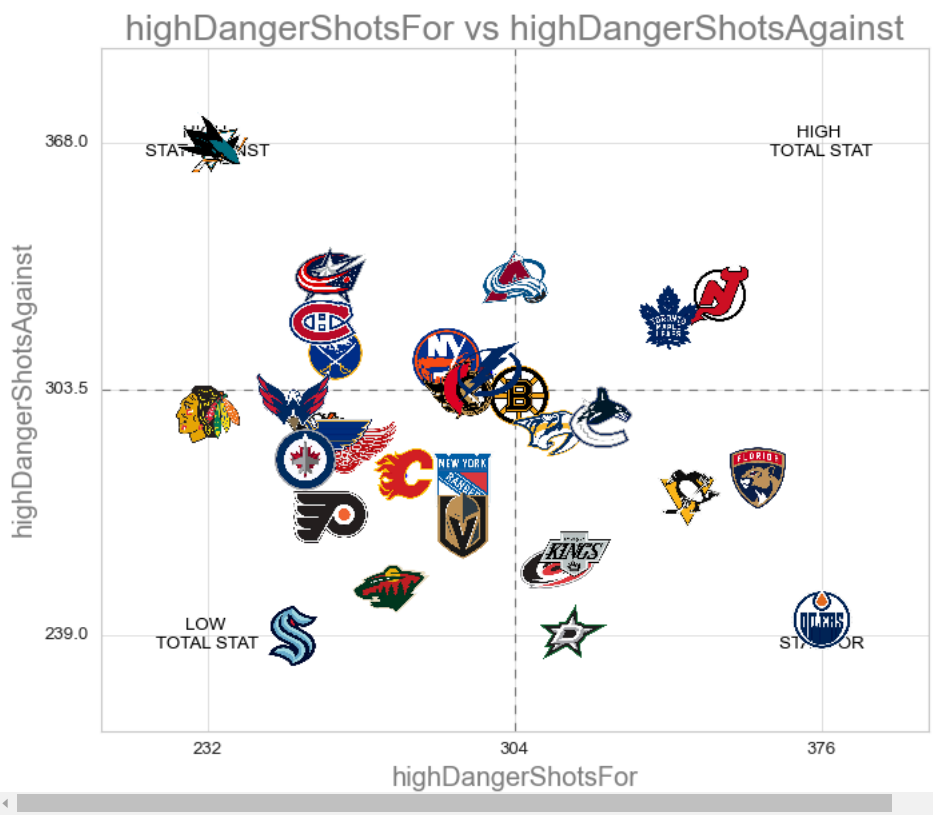

```
# Hypothesis test for Medium Danger Shots Percentage (MDS%)
two_sample_5050_test(stat='MDS%')
```

```
Reject the Null Hypothesis
Probability of making the playoffs when MDS% > 50%: 0.2131
Probability of making the playoffs when MDS% < 50%: 0.8482
```



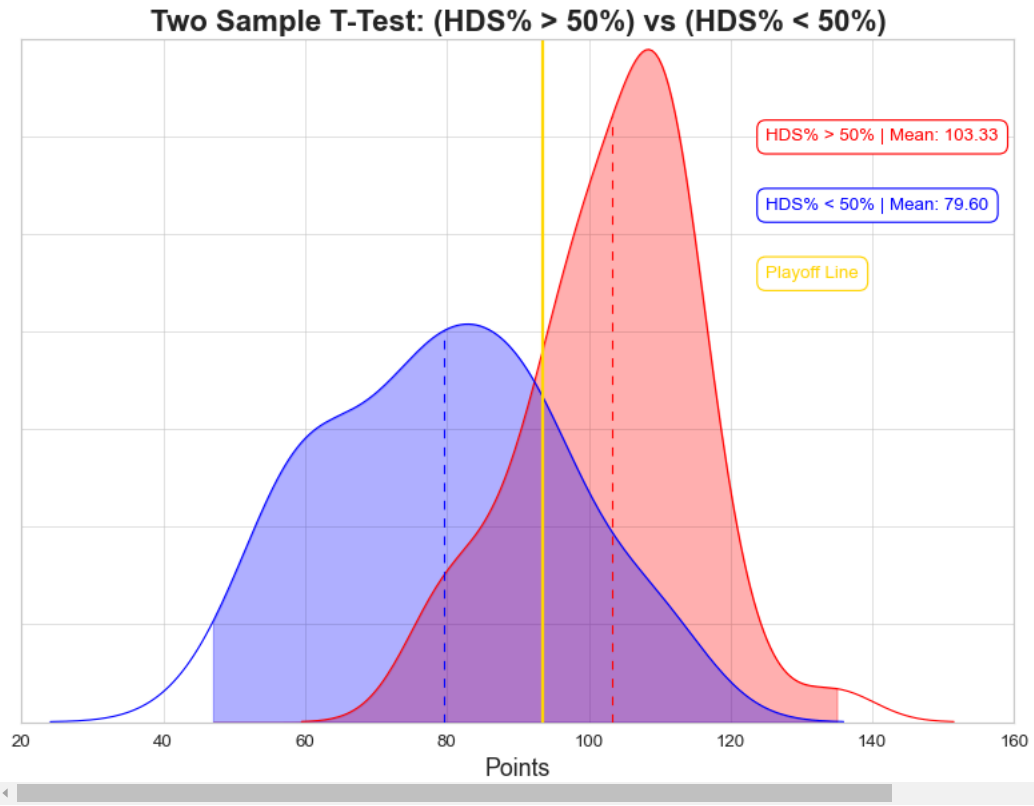
High Danger Shots

```
# a scatter plot of the following test for High Danger Shots during this past season
scatter(sf='highDangerShotsFor', sa='highDangerShotsAgainst', year=20232024)
```




```
# Hypothesis test for High Danger Shots Percentage (HDS%)
two_sample_5050_test(stat='HDS%')
```

```
Reject the Null Hypothesis
Probability of making the playoffs when HDS% > 50%: 0.2139
Probability of making the playoffs when HDS% < 50%: 0.7663
```

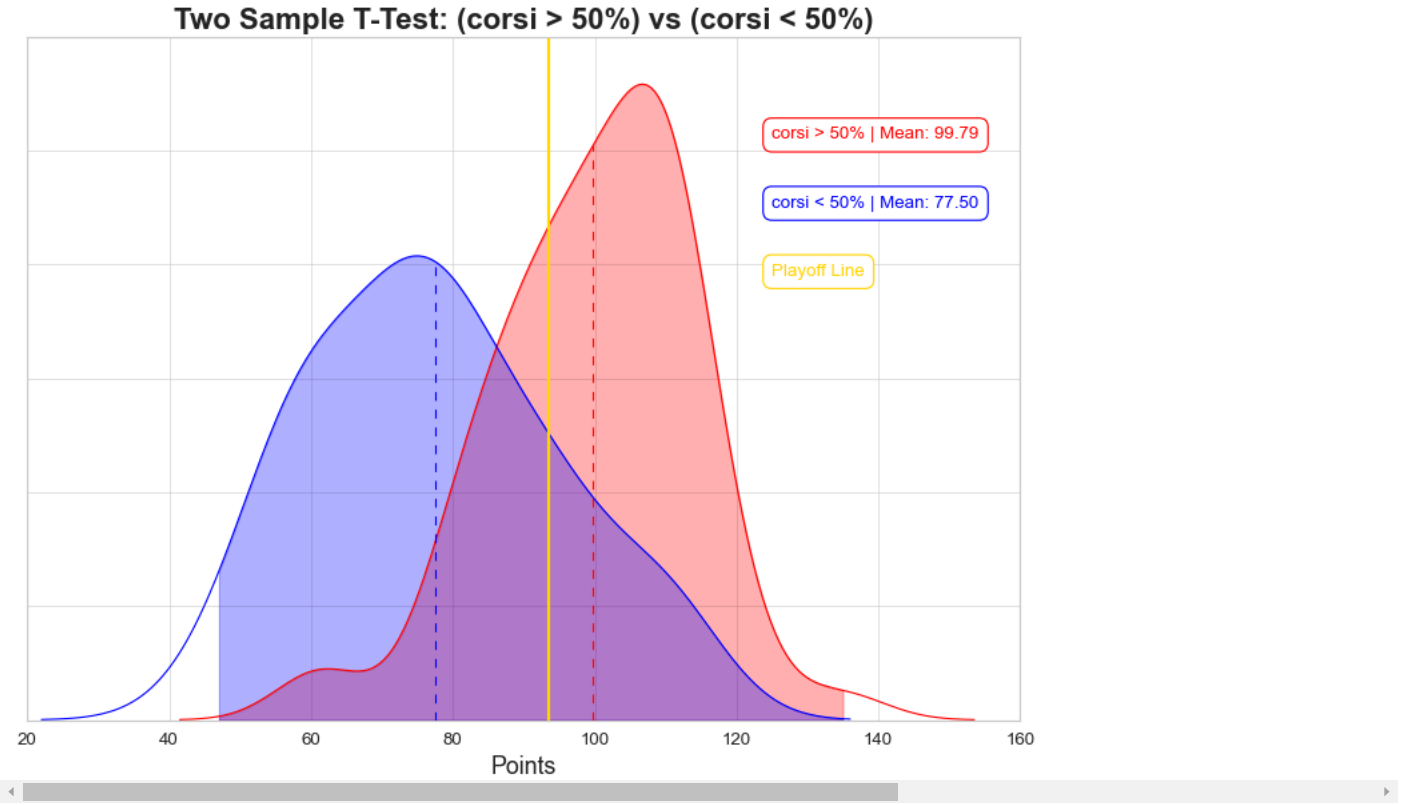


Corsi & Fenwick

Corsi and Fenwick are calulations to determine how much possession/momentum a team has in a game

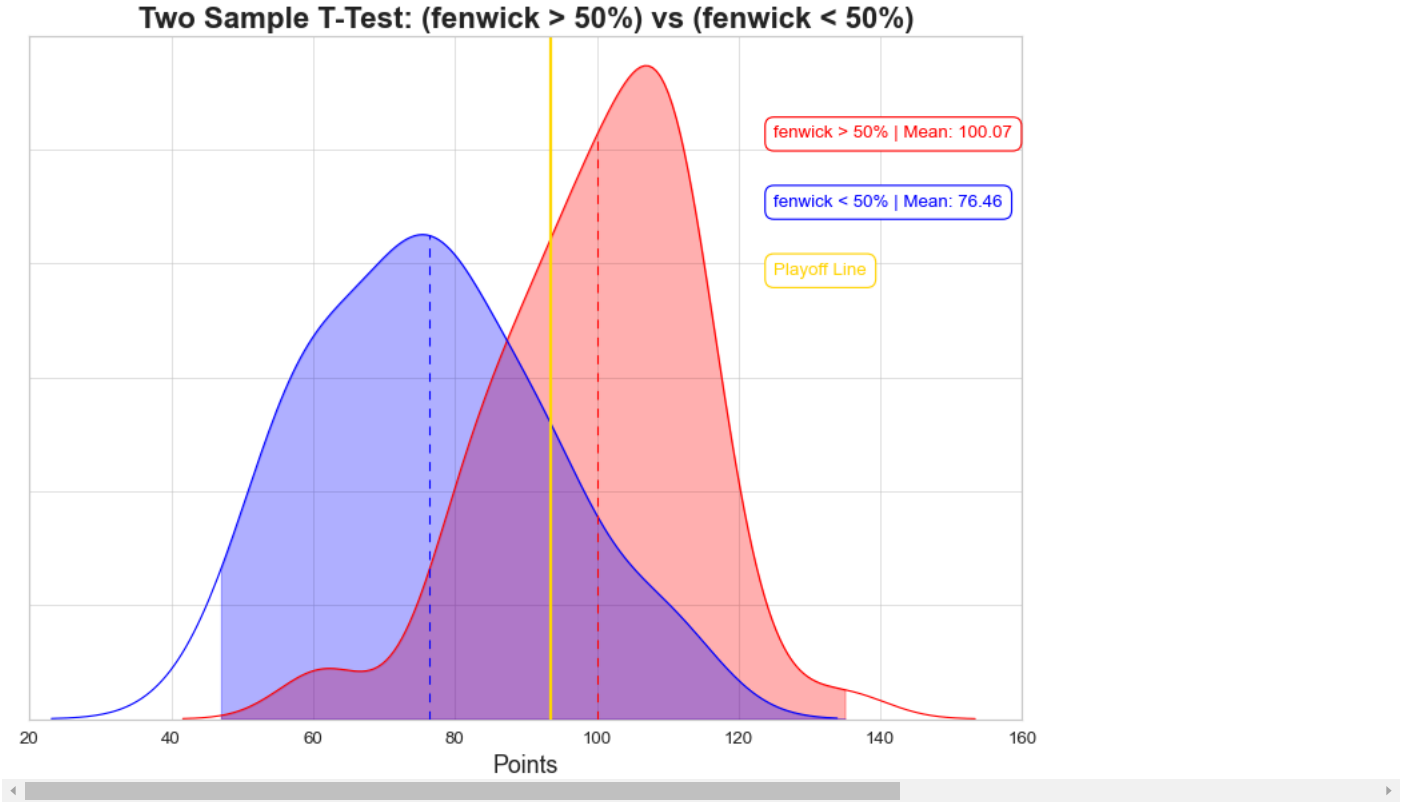
```
two_sample_5050_test(stat='corsi')
```

```
Reject the Null Hypothesis
Probability of making the playoffs when corsi > 50%: 0.3183
Probability of making the playoffs when corsi < 50%: 0.7905
```




```
two_sample_5050_test(stat='fenwick')
```

```
Reject the Null Hypothesis
Probability of making the playoffs when fenwick > 50%: 0.3076
Probability of making the playoffs when fenwick < 50%: 0.8195
```



Inferential Question 7+ : Of the Stats involving shots; Does being above average in Shots For have any relation to a Team's Goals For? Does being above average in Shots Against have any relation to a Team's Goals Against?

Null Hypothesis (H_0): $\mu_{AboveAVGShots} = \mu_{BelowAVGShots}$

Alternate Hypothesis (H_A): $\mu_{AboveAVGShots} \neq \mu_{BelowAVGShots}$

The following two functions are very similar to each other and the perviously mentioned functions, the main difference between the two of these function is what it prints as one is for "Goals For" and the other is for "Goals Against"

```
def two_sample_test_goals(df=teams, situation='all', stat='', statlabel='' ,save=False):
    MASK_SIT = (df['situation'] == situation)

    stat_mean = df[MASK_SIT][stat].mean()

    MASK_50P = df[stat] >= stat_mean
    MASK_50M = df[stat] < stat_mean

    good = (df[MASK_SIT & MASK_50P]['GF'])
    bad = (df[MASK_SIT & MASK_50M]['GF'])

    if len(good) == 0 or len(bad) == 0:
        print('Check input')
        return

    # Running two sample t-test and printing result
    _, p_value = stats.ttest_ind(good, bad)
    if p_value < 0.05:
        print('Reject the Null Hypothesis')
    else:
        print('Fail to reject the null hypothesis')

    # Creating label for KDE Plot
    label1 = (f'{statlabel} Above Average')
    label2 = (f'{statlabel} Below Average')

    # Calculate the means of both datasets
    mean1 = np.mean(good)
    mean2 = np.mean(bad)

    # Use a style template
    sns.set_style('whitegrid')

    plt.figure(figsize=(12, 8))

    # Plot both datasets
    sns.kdeplot(good, label=label1, color='red', alpha=1)
    sns.kdeplot(bad, label=label2, color='blue', alpha=1)

    # Compute KDE values
    kde_good = stats.gaussian_kde(good)
    kde_bad = stats.gaussian_kde(bad)

    threshold = 280

    # Calculate the probability for 'good' group
    prob_good = kde_good.integrate_box_1d(threshold, np.inf)

    # Calculate the probability for 'bad' group
    prob_bad = kde_bad.integrate_box_1d(threshold, np.inf)

    print(f'Probability of being Above Average in Goals For when {statlabel} is Above Average: {prob_good:.4f}')
    print(f'Probability of being Above Average in Goals For when {statlabel} is Below Average: {prob_bad:.4f}')

    # Generate x values for KDE
    x_vals = np.linspace(min(good.min(), bad.min()), max(good.max(), bad.max()), 1000)

    # Fill the entire area under the curves
    plt.fill_between(x_vals, kde_good(x_vals), color='red', alpha=0.3)
    plt.fill_between(x_vals, kde_bad(x_vals), color='blue', alpha=0.3)

    # Get KDE values at means
    kde_good_at_mean1 = kde_good(mean1)[0]
    kde_bad_at_mean2 = kde_bad(mean2)[0]

    # Add vertical lines at the means, stopping at the KDE curves
    plt.plot([mean1, mean1], [0, kde_good_at_mean1], 'r--', label=f'{label1} Mean: {mean1:.2f}')
    plt.plot([mean2, mean2], [0, kde_bad_at_mean2], 'b--', label=f'{label2} Mean: {mean2:.2f}')

    # Add a gold vertical line at 255
    plt.axvline(280, color='gold', linestyle='-', linewidth=2, label='Playoff Line')

    # Add annotations
    plt.annotate(f'{label1} \nMean: {mean1:.2f}', xy=(0.75, 0.85), xycoords='axes fraction', color='red', fontsize=12,
                 bbox=dict(facecolor='white', edgecolor='red', boxstyle='round,pad=0.5'))
    plt.annotate(f'{label2} \nMean: {mean2:.2f}', xy=(0.75, 0.75), xycoords='axes fraction', color='blue', fontsize=12,
                 bbox=dict(facecolor='white', edgecolor='blue', boxstyle='round,pad=0.5'))
```

```
        plt.annotate('Average Goals For Line', xy=(0.75, 0.65), xycoords='axes fraction', color='black', fontsize=12,
                     bbox=dict(facecolor='gold', edgecolor='gold', boxstyle='round,pad=0.5'))

        plt.title(f'{statlabel} \nAbove Average vs Below Average', fontsize=20, fontweight='bold')
        plt.xlabel('Points', fontsize=16)
        plt.ylabel('')
        plt.yticks(visible=False)

        if save:
            plt.savefig(f'gaa vs gba {stat}.png', transparent=True)

        plt.show()
```

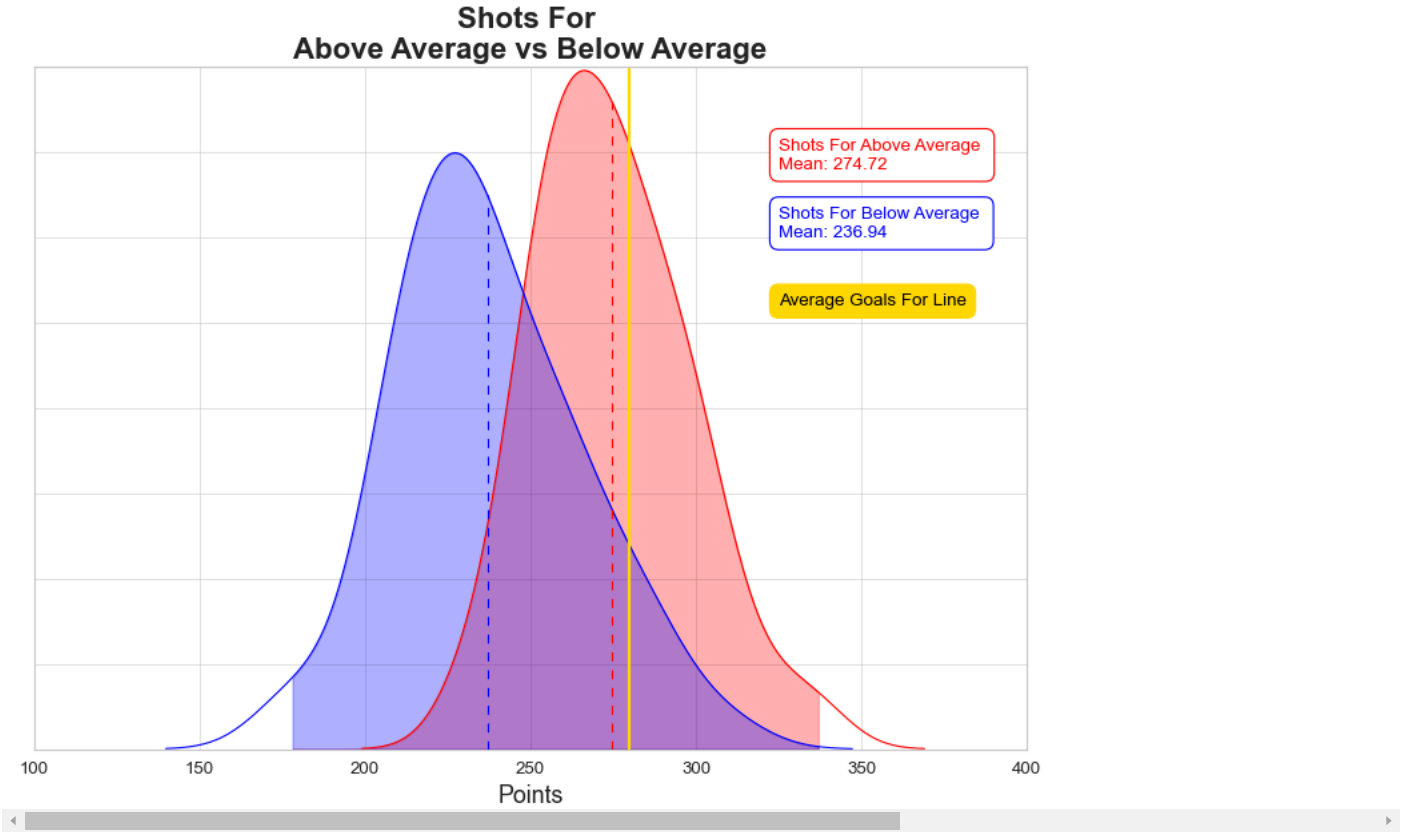
```
def two_sample_test_goalsa(df=teams, situation='all', stat='', statlabel='' ,save=False):
    MASK_SIT = (df['situation'] == situation)

    stat_mean = df[MASK_SIT][stat].mean()

    MASK_50P = df[stat] >= stat_mean
    MASK_50M = df[stat] < stat_mean

    good = (df[MASK_SIT & MASK_50P]['GA'])
    bad = (df[MASK_SIT & MASK_50M]['GA'])

    if len(good) == 0 or len(bad) == 0:
        print('Check input')
        return

    # Running two sample t-test and printing result
    _, p_value = stats.ttest_ind(good, bad)
    if p_value < 0.05:
        print('Reject the Null Hypothesis')
    else:
        print('Fail to reject the null hypothesis')

    # Creating label for KDE Plot
    label1 = (f'{statlabel} Above Average')
    label2 = (f'{statlabel} Below Average')

    # Calculate the means of both datasets
    mean1 = np.mean(good)
    mean2 = np.mean(bad)

    # Use a style template
    sns.set_style('whitegrid')

    plt.figure(figsize=(12, 8))

    # Plot both datasets
    sns.kdeplot(good, label=label1, color='red', alpha=1)
    sns.kdeplot(bad, label=label2, color='blue', alpha=1)

    # Compute KDE values
    kde_good = stats.gaussian_kde(good)
    kde_bad = stats.gaussian_kde(bad)

    threshold = 255

    # Calculate the probability for 'good' group
    prob_good = kde_good.integrate_box_1d(threshold, np.inf)

    # Calculate the probability for 'bad' group
    prob_bad = kde_bad.integrate_box_1d(threshold, np.inf)

    print(f'Probability of being Above Average in Goals Against when {stat} is Above Average: {prob_good:.4f}')
    print(f'Probability of being Above Average in Goals Against when {stat} is Below Average: {prob_bad:.4f}')

    # Generate x values for KDE
    x_vals = np.linspace(min(good.min(), bad.min()), max(good.max(), bad.max()), 1000)

    # Fill the entire area under the curves
    plt.fill_between(x_vals, kde_good(x_vals), color='red', alpha=0.3)
    plt.fill_between(x_vals, kde_bad(x_vals), color='blue', alpha=0.3)

    # Get KDE values at means
    kde_good_at_mean1 = kde_good(mean1)[0]
    kde_bad_at_mean2 = kde_bad(mean2)[0]

    # Add vertical lines at the means, stopping at the KDE curves
    plt.plot([mean1, mean1], [0, kde_good_at_mean1], 'r--', label=f'{label1} Mean: {mean1:.2f}')
    plt.plot([mean2, mean2], [0, kde_bad_at_mean2], 'b--', label=f'{label2} Mean: {mean2:.2f}')

    # Add a gold vertical line at 255
    plt.axvline(255, color='gold', linestyle='-', linewidth=2, label='Playoff Line')

    # Add annotations
    plt.annotate(f'{label1} \nMean: {mean1:.2f}', xy=(0.75, 0.85), xycoords='axes fraction', color='red', fontsize=12,
                 bbox=dict(facecolor='white', edgecolor='red', boxstyle='round,pad=0.5'))
    plt.annotate(f'{label2} \nMean: {mean2:.2f}', xy=(0.75, 0.75), xycoords='axes fraction', color='blue', fontsize=12,
                 bbox=dict(facecolor='white', edgecolor='blue', boxstyle='round,pad=0.5'))
```

```
    plt.annotate('Average Goals Against Line', xy=(0.75, 0.65), xycoords='axes fraction', color='black', fontsize=12,
                 bbox=dict(facecolor='gold', edgecolor='gold', boxstyle='round,pad=0.5'))

    plt.title(f'{statlabel} \nAbove Average vs Below Average', fontsize=20, fontweight='bold')
    plt.xlabel('Points', fontsize=16)
    plt.ylabel('')
    plt.yticks(visible=False)

    if save:
        plt.savefig(f'gaa vs gba {stat}.png', transparent=True)

    plt.show()
```

Shots For/Against

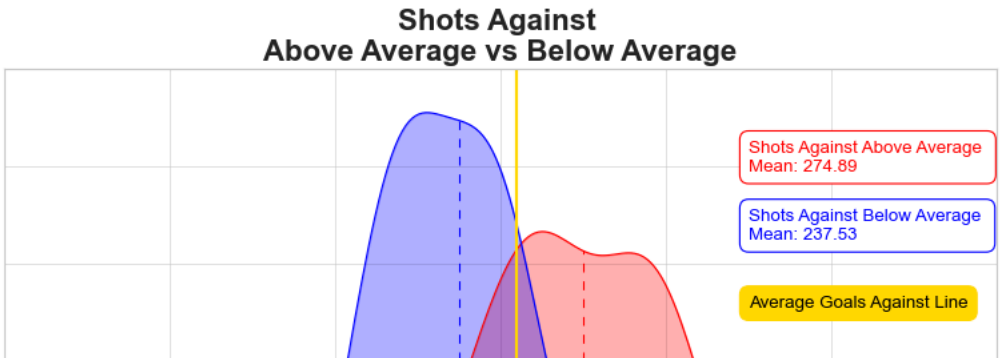
```
two_sample_test_goals(stat='shotsFor', statlabel='Shots For')
```

```
Reject the Null Hypothesis
Probability of being Above Average in Goals For when Shots For is Above Average: 0.3921
Probability of being Above Average in Goals For when Shots For is Below Average: 0.0929
```




```
two_sample_test_goalsa(stat='shotsAgainst', statlabel='Shots Against')
```

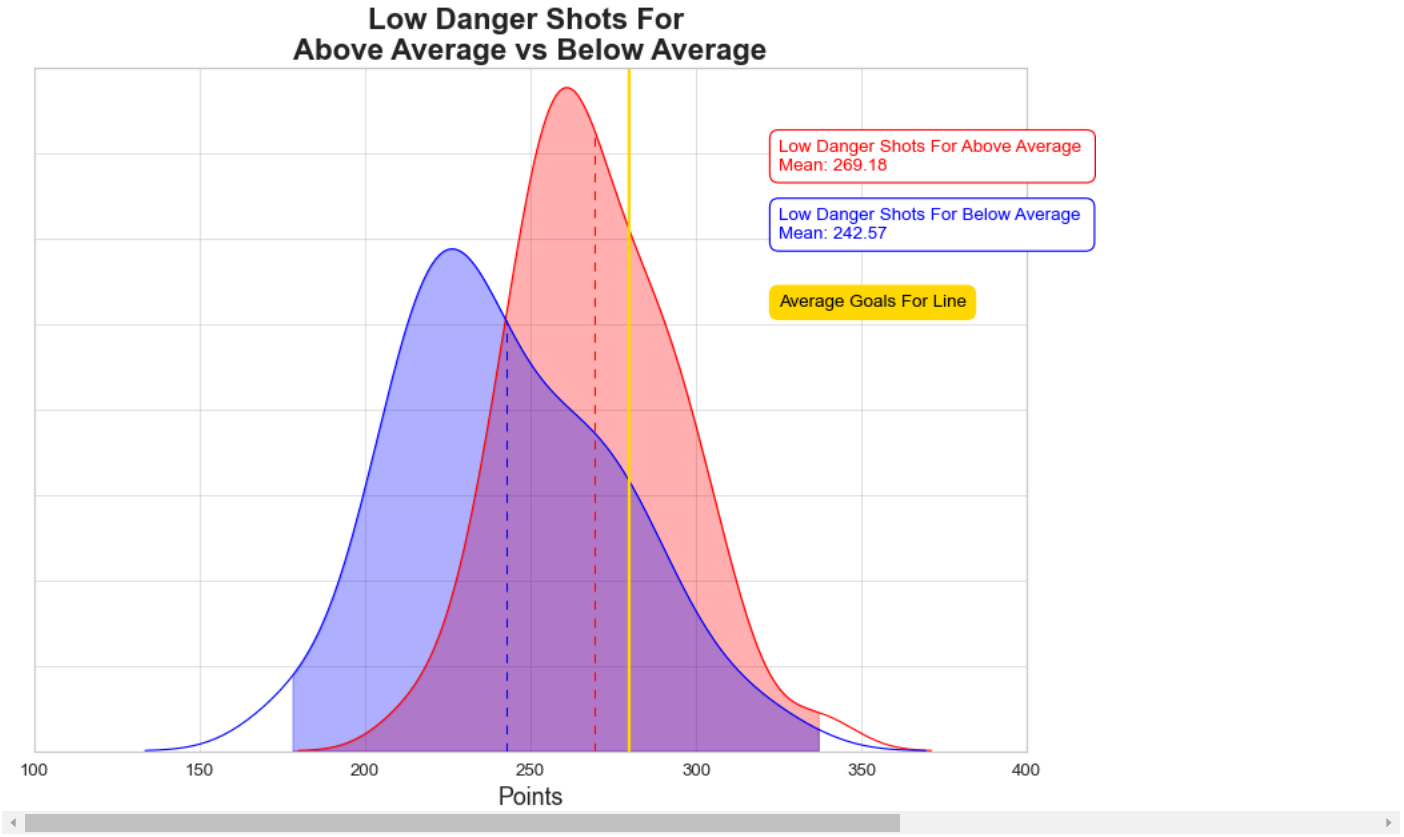
```
Reject the Null Hypothesis
Probability of being Above Average in Goals Against when shotsAgainst is Above Average: 0.7049
Probability of being Above Average in Goals Against when shotsAgainst is Below Average: 0.2671
```



Low Danger Shots For/Against

```
two_sample_test_goals(stat='lowDangerShotsFor', statlabel='Low Danger Shots For')
```

```
Reject the Null Hypothesis
Probability of being Above Average in Goals For when Low Danger Shots For is Above Average: 0.3290
Probability of being Above Average in Goals For when Low Danger Shots For is Below Average: 0.1599
```




```
two_sample_test_goalsa(stat='lowDangerShotsAgainst', statlabel='Low Danger Shots Against')
```

```
Reject the Null Hypothesis
Probability of being Above Average in Goals Against when lowDangerShotsAgainst is Above Average: 0.6174
Probability of being Above Average in Goals Against when lowDangerShotsAgainst is Below Average: 0.3437
```