ROCKET LEAGUE

WORLD

CHAMPIONSHIP

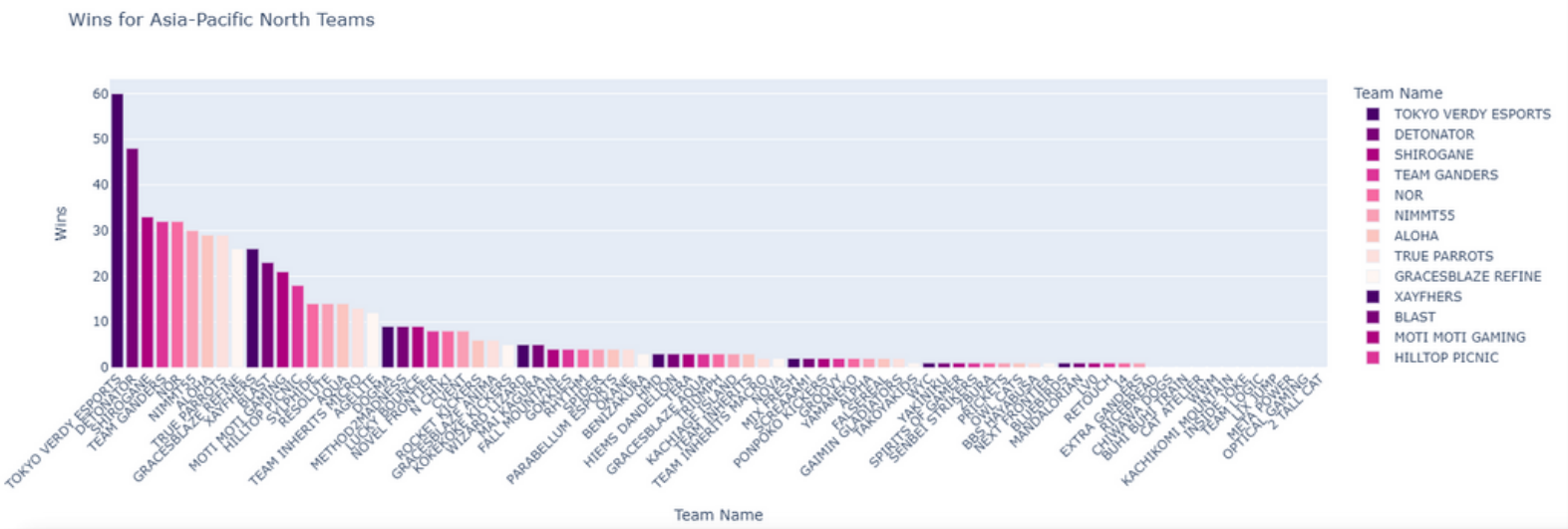
2021 - 2022

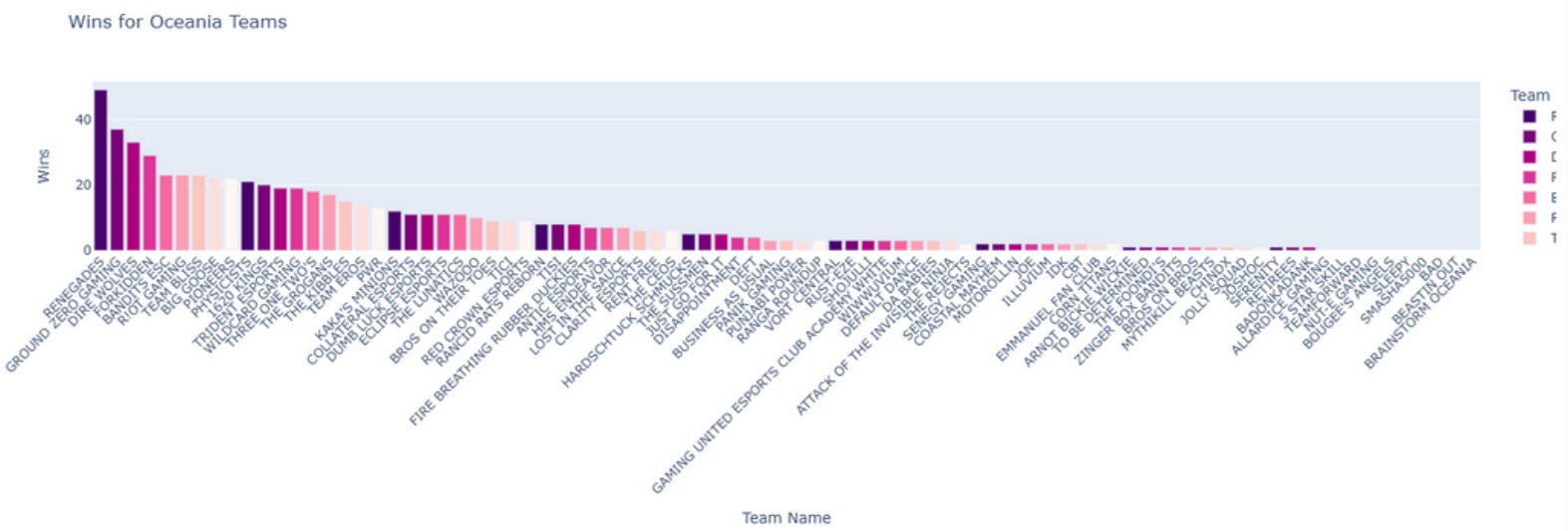
DATA ANALYTICS

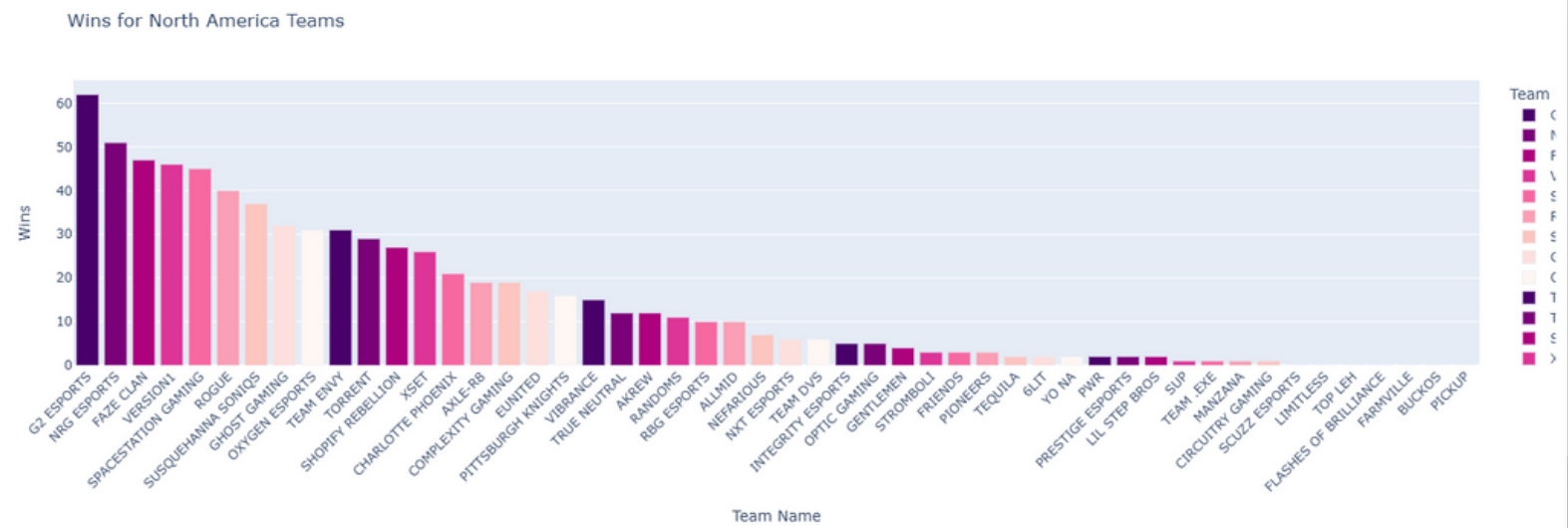
20BCM016
Dhiraj Makhijani
Assignment 2

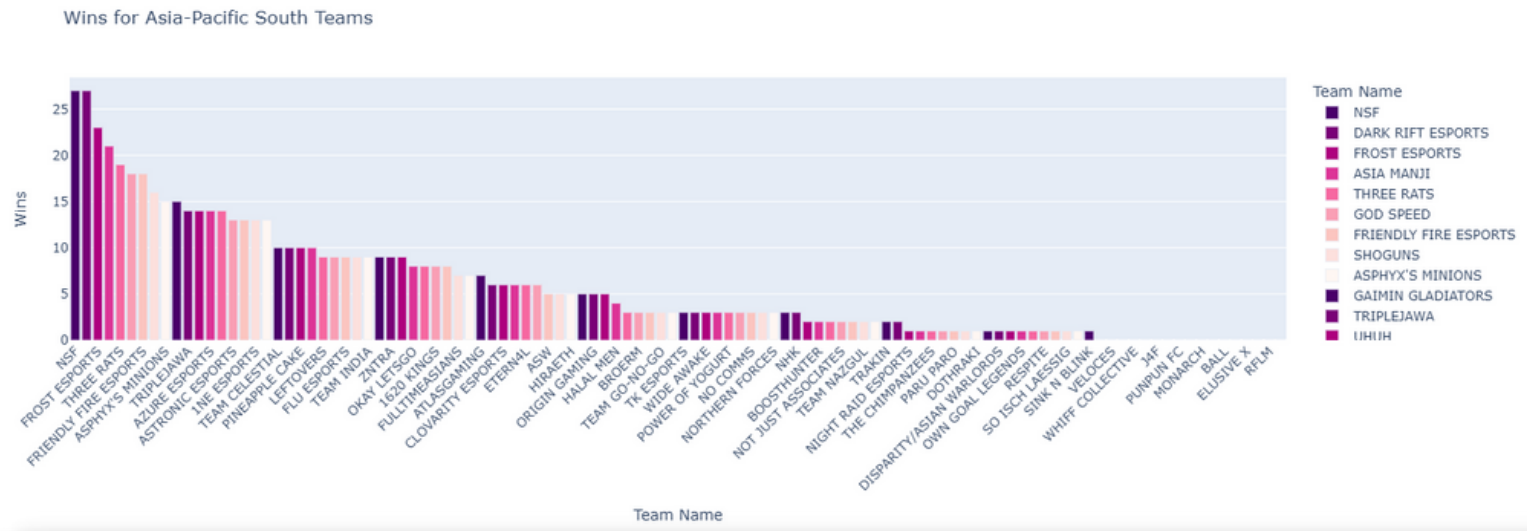
Overview

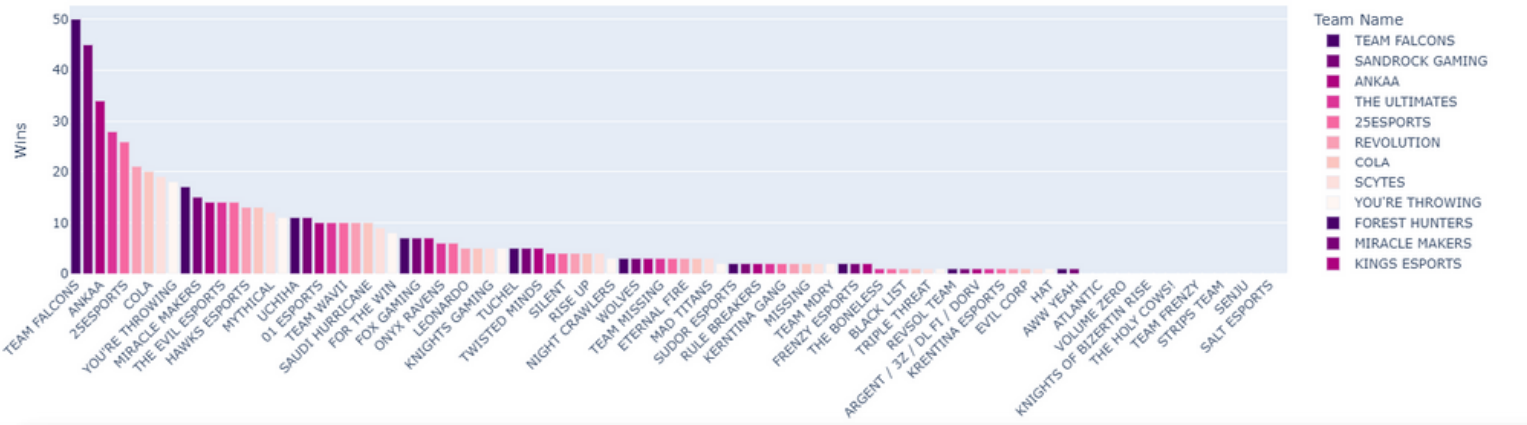
The given bar graph displays all teams in each region based on their wins in the year 2021-2022. It provides a clear understanding of the performance of teams per region.

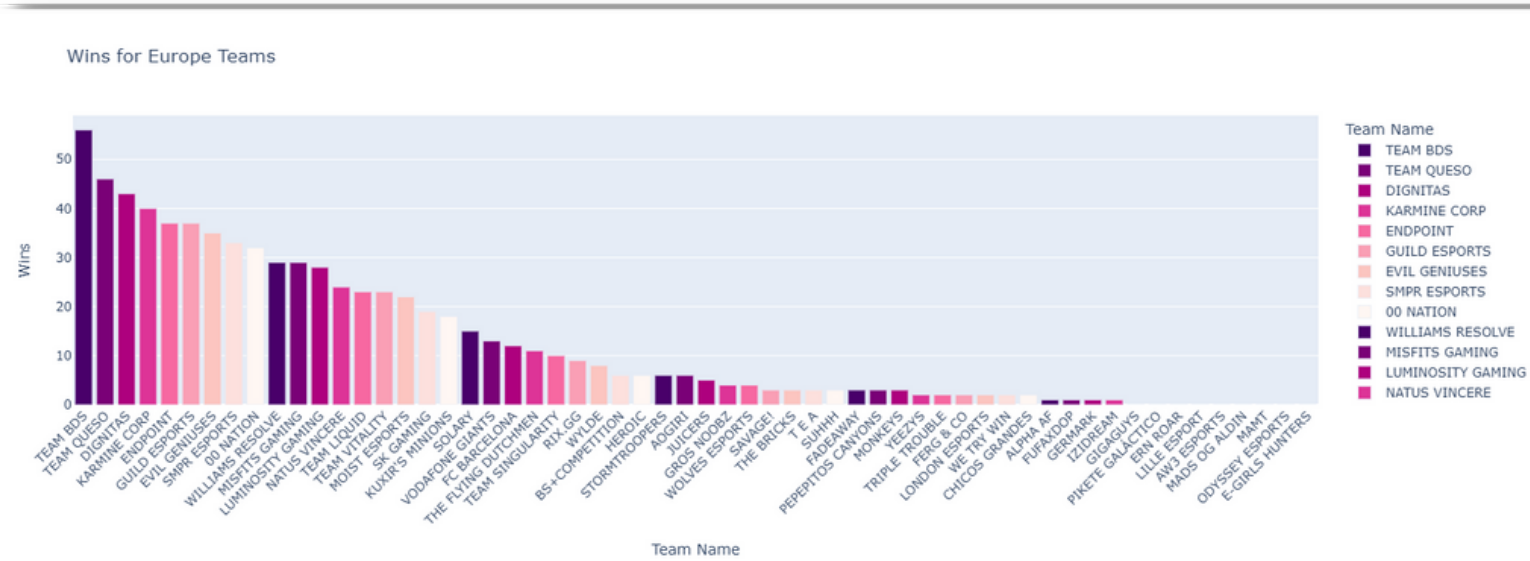


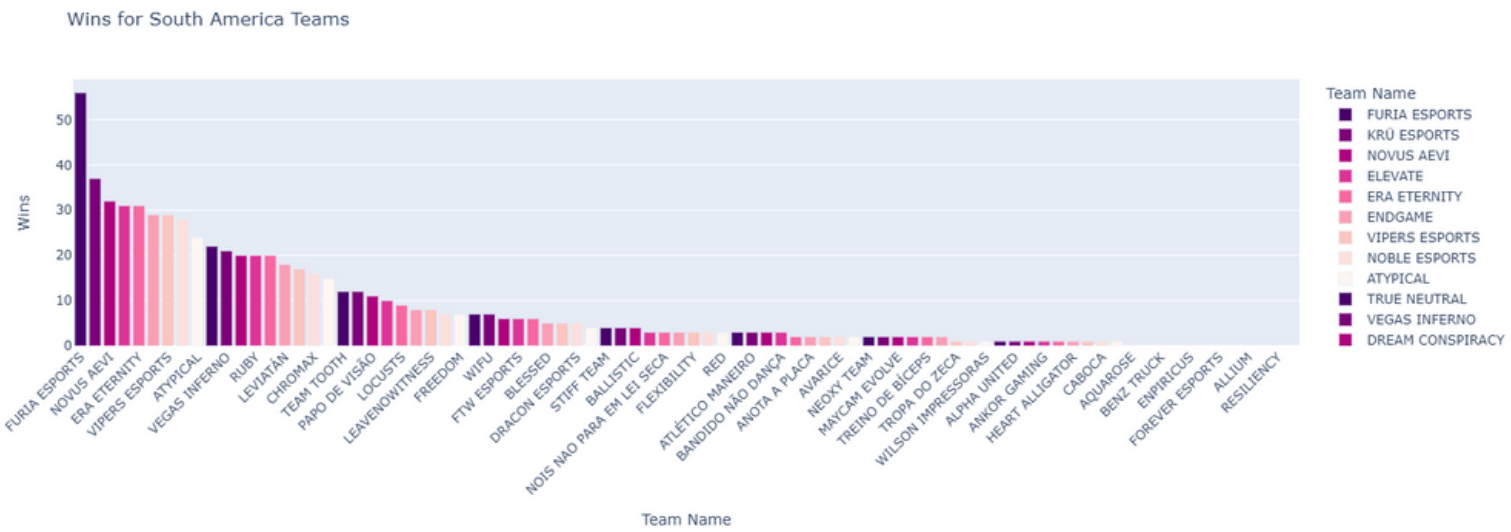




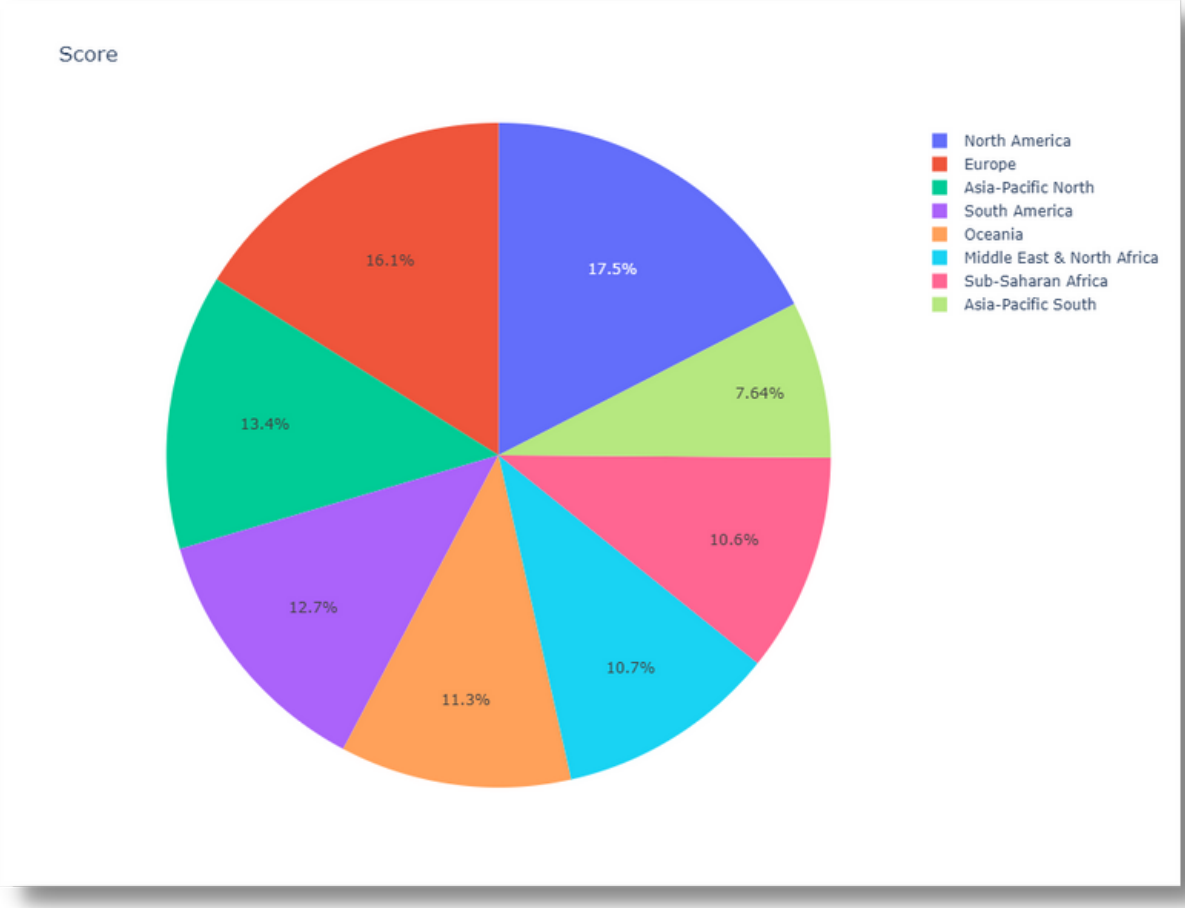

Wins for Asia-Pacific South Teams
Wins
Team Name
NSF
FROST ESPORTS
THREE RATS
FRIENDLY FIRE ESPORTS
ASPHYX'S MINIONS
TRIPLEJAWA
AZURE ESPORTS
ASTRONIC ESPORTS
1NE ESPORTS
TEAM CELESTIAL
PINEAPPLE CAKE
LEFTOVERS
FLU ESPORTS
TEAM INDIA
ZNTRA
OKAY LETSGO
1620 KINGS
FULLTIMEASIANS
ATLASGAMING
CLOVARITY ESPORTS
ETERNAL
ASW
HIRAETH
ORIGIN GAMING
HALAL MEN
BROERM
TEAM GO-NO-GO
TK ESPORTS
WIDE AWAKE
POWER OF YOGURT
NO COMMS
NORTHERN FORCES
NHK
BOOSTHUNTER
NOT JUST ASSOCIATES
TEAM NAZGUL
TRAKIN
NIGHT RAID ESPORTS
THE CHIMPANZEES
PARU PARO
DISPARITY/ASIAN WARLORDS
DOTHRAKI
OWN GOAL LEGENDS
RESPITE
SO ISCH LAESSIG
SINK N BLINK
VELOCES
WHIFF COLLECTIVE
J4F
PUNPUN FC
MONARCH
BALL
ELUSIVE X
RFLM
Team Name
NSF
DARK RIFT ESPORTS
FROST ESPORTS
ASIA MANJI
THREE RATS
GOD SPEED
FRIENDLY FIRE ESPORTS
SHOGUNS
ASPHYX'S MINIONS
GAIMIN GLADIATORS
TRIPLEJAWA
UHUH


Wins for Middle East & North Africa Teams
Wins
Team Name
TEAM FALCONS
ANKAA
25ESPORTS
COLA
YOU'RE THROWING
MIRACLE MAKERS
THE EVIL ESPORTS
HAWKS ESPORTS
MYTHICAL
UCHIHA
01 ESPORTS
TEAM WAVII
SAUDI HURRICANE
FOR THE WIN
FOX GAMING
ONYX RAVENS
LEONARDO
KNIGHTS GAMING
TUCHEL
TWISTED MINDS
SILENT
RISE UP
NIGHT CRAWLERS
WOLVES
TEAM MISSING
ETERNAL FIRE
MAD TITANS
SUDOR ESPORTS
RULE BREAKERS
KERNTINA GANG
MISSING
TEAM MORY
FRENZY ESPORTS
THE BONELESS
BLACK LIST
TRIPLE THREAT
REVSOL TEAM
ARGENT / 3Z / DL F1 / DORV
KRENTINA ESPORTS
EVIL CORP
HAT
AWW YEAH
ATLANTIC
VOLUME ZERO
KNIGHTS OF BIZERTIN RISE
THE HOLY COWS!
TEAM FRENZY
STRIPS TEAM
SENJU
SALT ESPORTS
Team Name
TEAM FALCONS
SANDROCK GAMING
ANKAA
THE ULTIMATES
25ESPORTS
REVOLUTION
COLA
SCYTES
YOU'RE THROWING
FOREST HUNTERS
MIRACLE MAKERS
KINGS ESPORTS


Wins for Europe Teams
Wins
Team Name
TEAM BDS
TEAM QUESO
DIGNITAS
KARMINE CORP
ENDPOINT
GUILD ESPORTS
EVIL GENIUSES
SMPR ESPORTS
00 NATION
WILLIAMS RESOLVE
MISFITS GAMING
LUMINOSITY GAMING
NATUS VINCERE
TEAM LIQUID
TEAM VITALITY
MOIST ESPORTS
SK GAMING
KUKIR'S MINIONS
SOLARY
VODAFONE GIANTS
FC BARCELONA
THE FLYING DUTCHMEN
TEAM SINGULARITY
RIX.GG
WYLDE
BS+COMPETITION
HEROIC
STORMTROOPERS
ADGIRI
JUICERS
GROS NOOBZ
WOLVES ESPORTS
SAVAGE!
THE BRICKS
T E A
SUHHH
FADEAWAY
PEPEPITOS CANYONS
MONKEYS
YEEZYS
TRIPLE TROUBLE
FERG & CO
LONDON ESPORTS
WE TRY WIN
CHICOS GRANDES
ALPHA AF
FUFAXDOP
GERMARK
IZIDREAM
GIGAGUYS
PIKETE GALACTICO
ERN ROAR
LILLE ESPORT
AW3 ESPORTS
MADS OG ALDIN
MAMT
ODYSSEY ESPORTS
E-GIRLS HUNTERS
Team Name
TEAM BDS
TEAM QUESO
DIGNITAS
KARMINE CORP
ENDPOINT
GUILD ESPORTS
EVIL GENIUSES
SMPR ESPORTS
00 NATION
WILLIAMS RESOLVE
MISFITS GAMING
LUMINOSITY GAMING
NATUS VINCERE



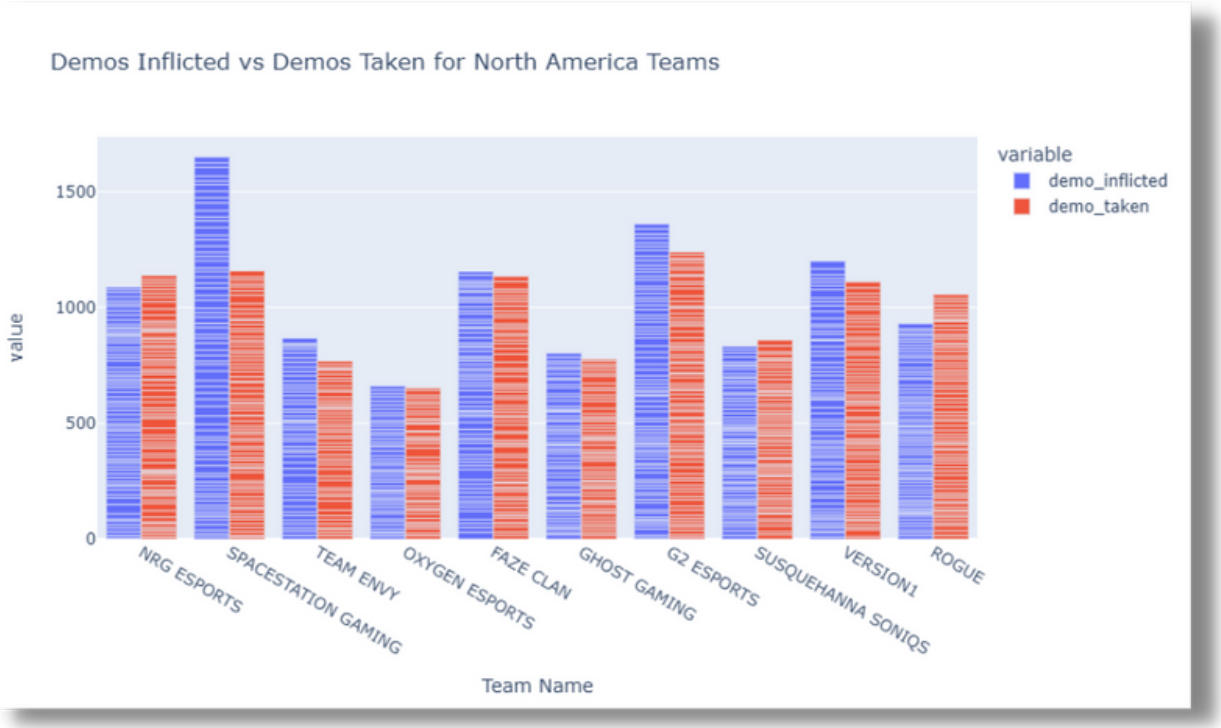
The following pie-chart showcases every goal scored by every region.

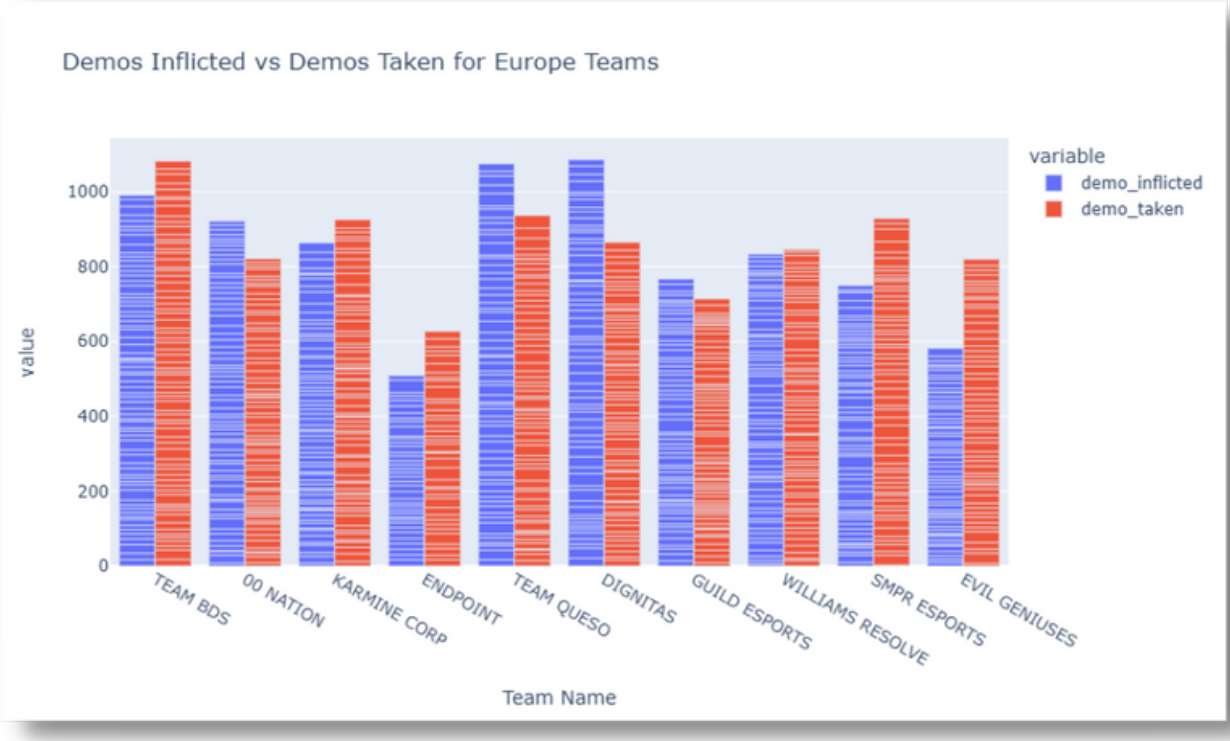


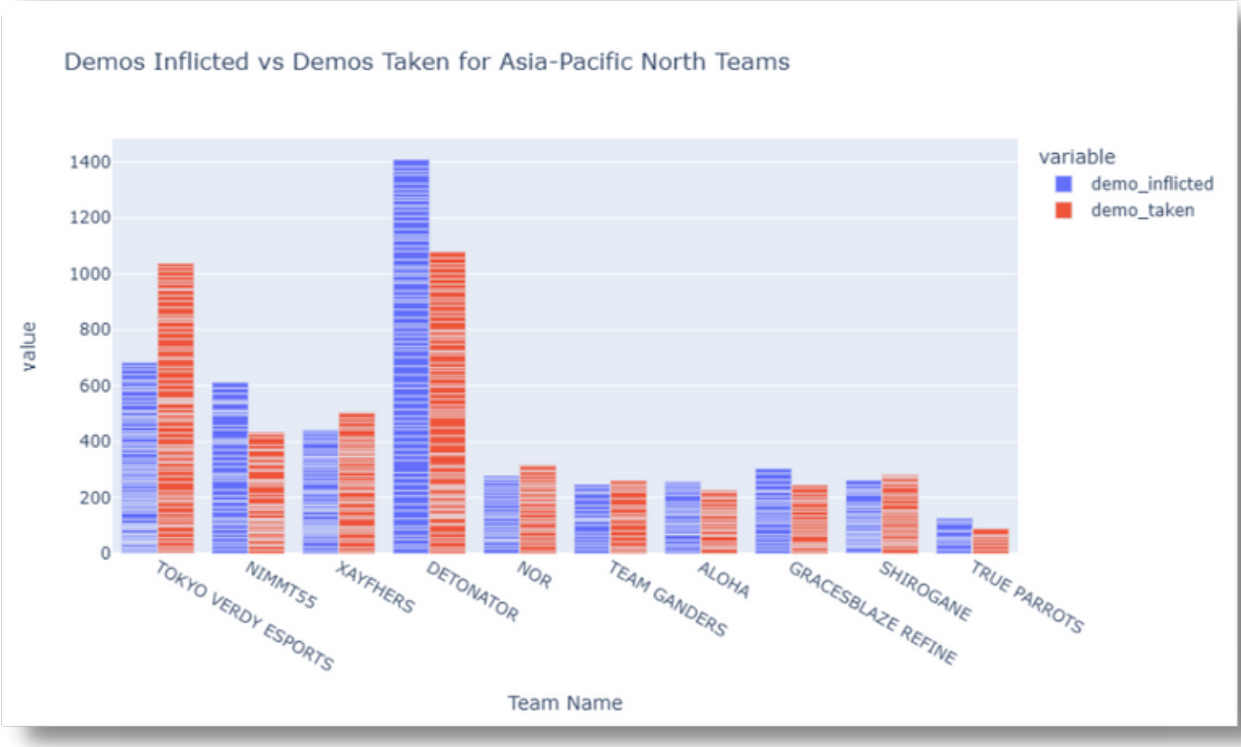
It is interesting to see Asia North being 3rd despite it having poor performance in LAN-based events where it faces off against other regions.

Out of 600 teams, we are going to focus on the **top 10** teams from every region from now on.

Let's discuss the concept of demos game mechanic here. When one player demos another player, the latter is removed from the game for approximately 5 seconds, creating opportunities to alter the game's outcome. Now, let's examine the number of demos taken and inflicted by each team across three significant regions.

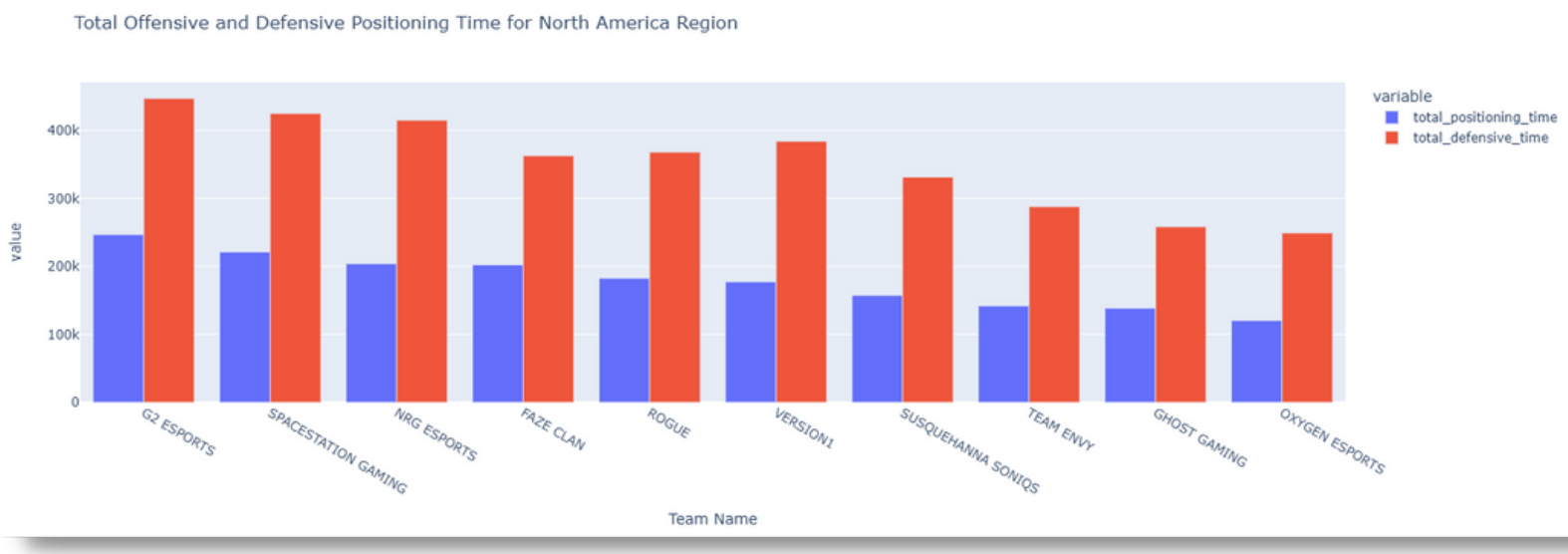






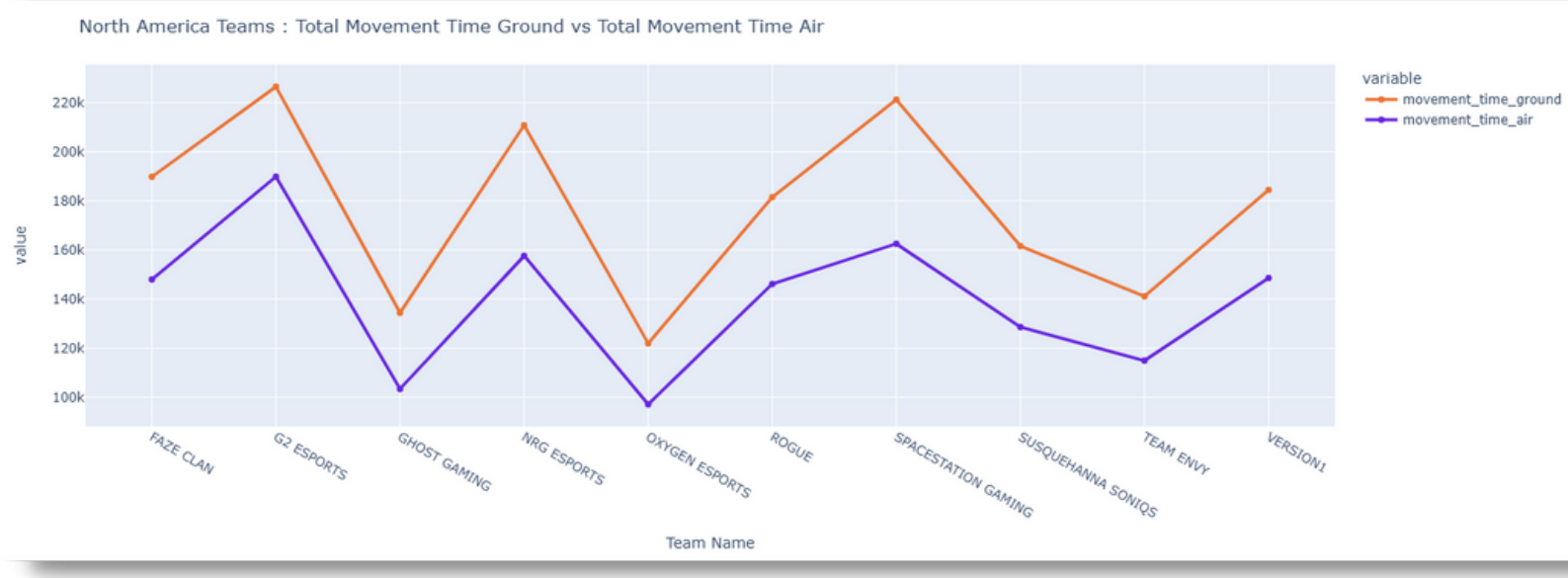
We can clearly see the demo meta is huge in North America followed by Europe and the least in Asia. North America and Europe are the top 2 regions in the world and we can say that demos is an important part of their playstyle.

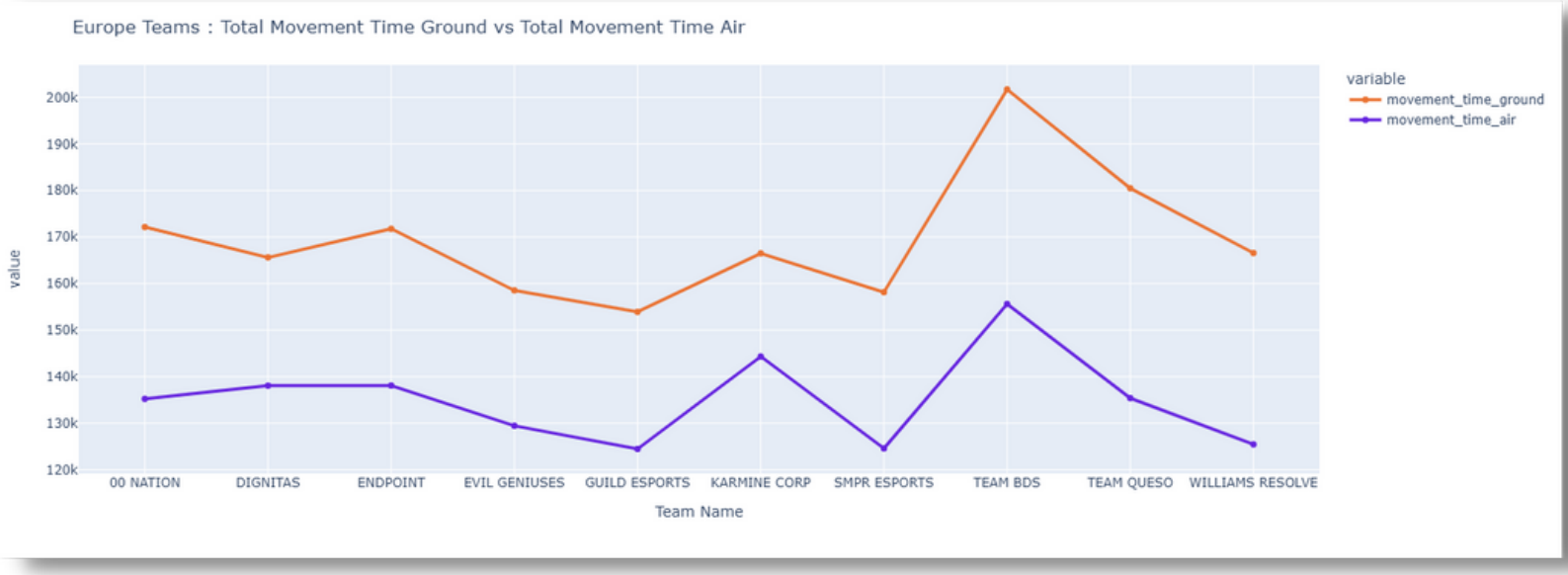
Let's see the total times each team spent on the offensive side and defensive side of the field respectively.

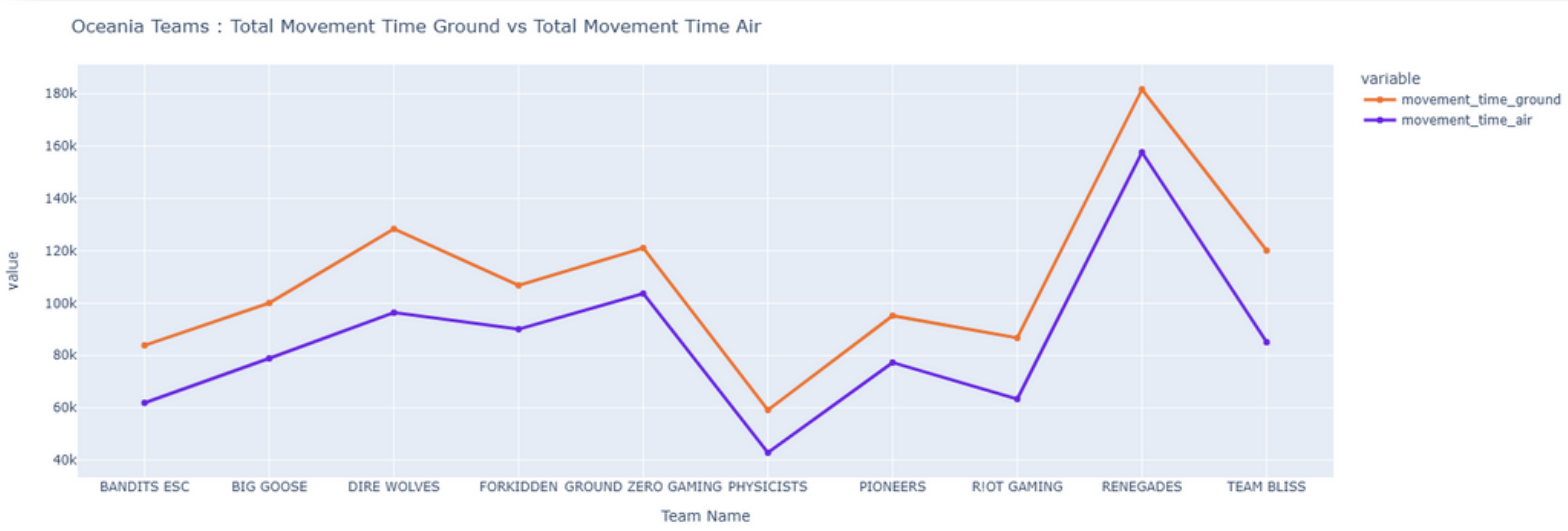


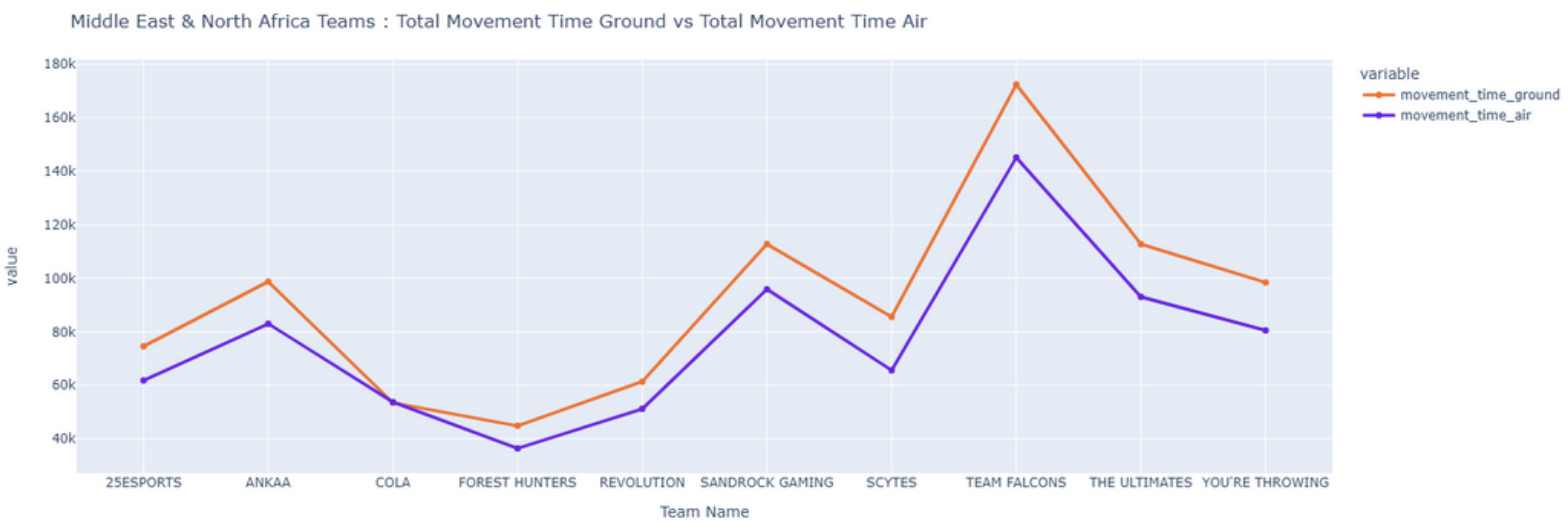
Every single region follows this pattern. It seems the counter-attacking strategy is more prominent. More time playing the defensive side, profiting off a strong clear and quickly scoring seems the most used strategy.

Next up is a team's ground play and airplay. Rocket League enables players to have control of the ball even in the air based on how a player manages his boost. Let's check ground vs air time of such teams.





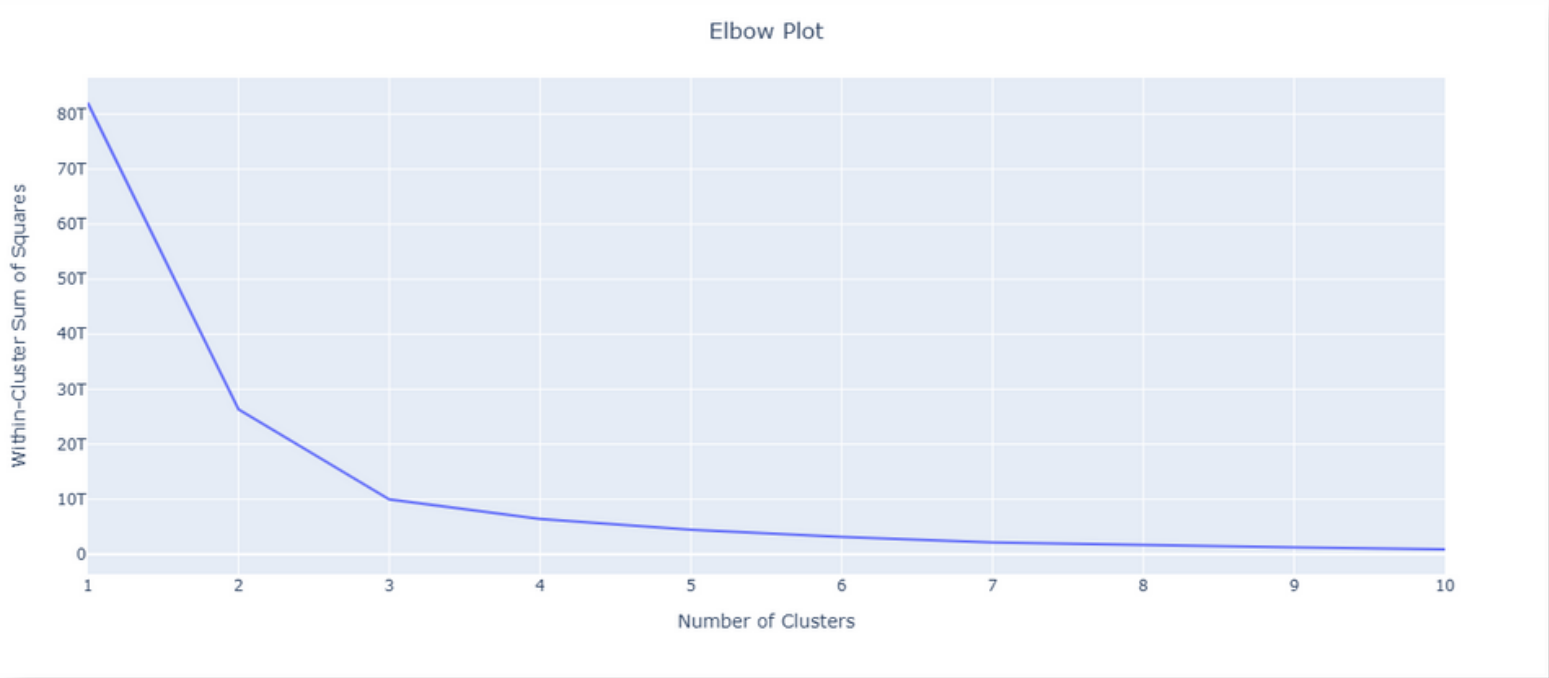


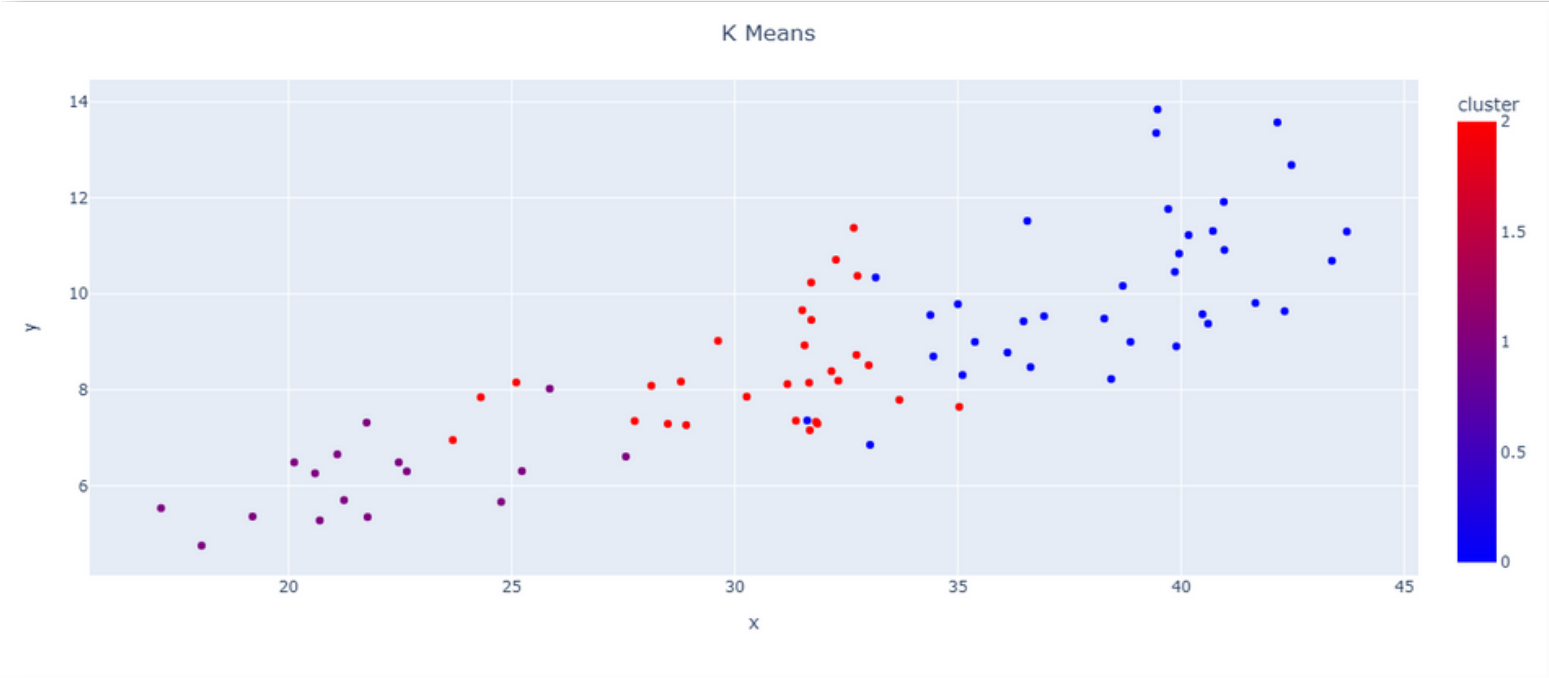


Every single region has more time on the ground as compared to time on air. However, it seems the middle eastern region spends a comparatively equal amount of time in both air and ground as compared with teams like North America and Europe where Europe spends way less time in air and is focused more on ground gameplay.

Clustering

We are now going to apply the **K Means** algorithm to our dataset. We are taking gameplay values which include goals, assists, saves, scores, boost time, position, etc. about 45 columns as our datapoints for 80 teams.



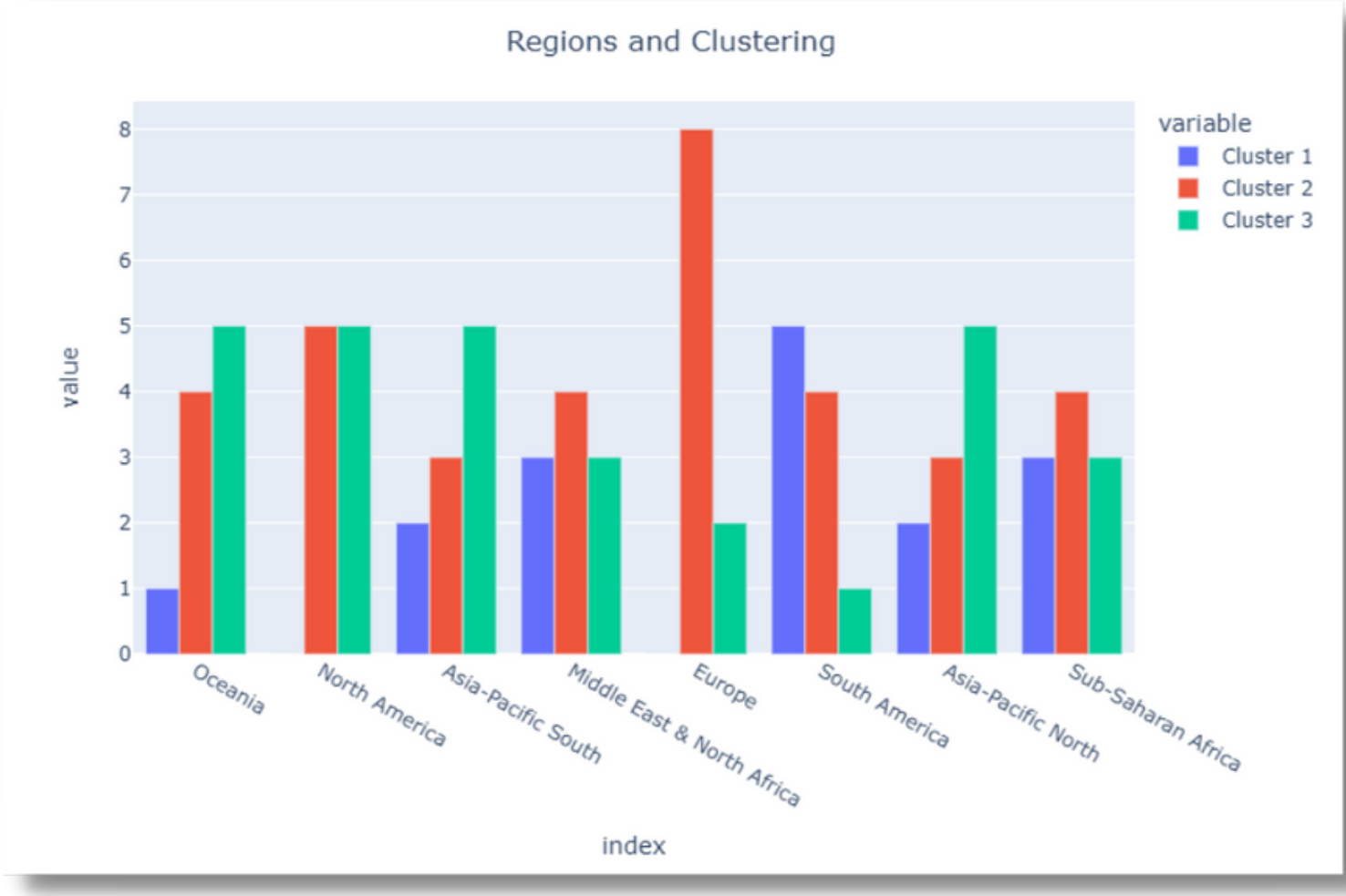


Upon hovering on our K mean plot, we are able to see the team name and its region. In our plot, we can see verticality with our points. This verticality adheres to a team's skills because it can be observed that all the top tier teams like Team BDS, G2, and Team Falcons are shown in the blue region, the middle tier teams are in the red region and the lowest tier teams are purple ones.

	Cluster 1	Cluster 2	Cluster 3
Oceania	1	4	5
North America	0	5	5
Asia-Pacific South	2	3	5
Middle East & North Africa	3	4	3
Europe	0	8	2
South America	5	4	1
Asia-Pacific North	2	3	5
Sub-Saharan Africa	3	4	3

To check the familiarity among regions, we have assigned each region its respective clusters by creating a dataframe with regions as rows and 3 clusters as columns.

We then plot the following as a bar graph.

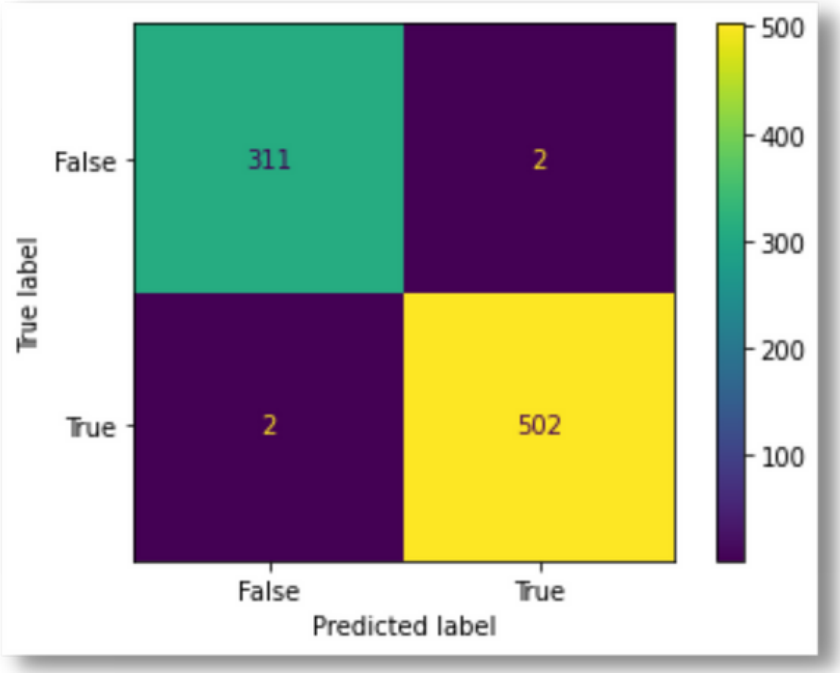


This gives an idea that our first cluster is based around South America, the second cluster is based around Europe and the third cluster is based on a number of regions which include Oceania, North America, and Asia.

Modelling

Using a random forest classifier and taking those gameplay-related data as our features we are able to build a model that will predict the winner between two teams.

This model is about 98% accurate with the confusion matrix given below.




```
[141] team1 = 'TEAM BDS'
      team2 = 'TEAM FALCONS'
      winner = predict_winner(team1, team2)
      print(f"The predicted winner between {team1} and {team2} is {winner}.")

      The predicted winner between TEAM BDS and TEAM FALCONS is TEAM BDS.
```

```
[142] team1 = 'G2 ESPORTS'
      team2 = 'KUXIR\'S MINIONS'
      winner = predict_winner(team1, team2)
      print(f"The predicted winner between {team1} and {team2} is {winner}.")

      The predicted winner between G2 ESPORTS and KUXIR'S MINIONS is G2 ESPORTS.
```

```
[143] team1 = 'FAZE CLAN'
      team2 = 'MOIST ESPORTS'
      winner = predict_winner(team1, team2)
      print(f"The predicted winner between {team1} and {team2} is {winner}.")

      The predicted winner between FAZE CLAN and MOIST ESPORTS is FAZE CLAN.
```

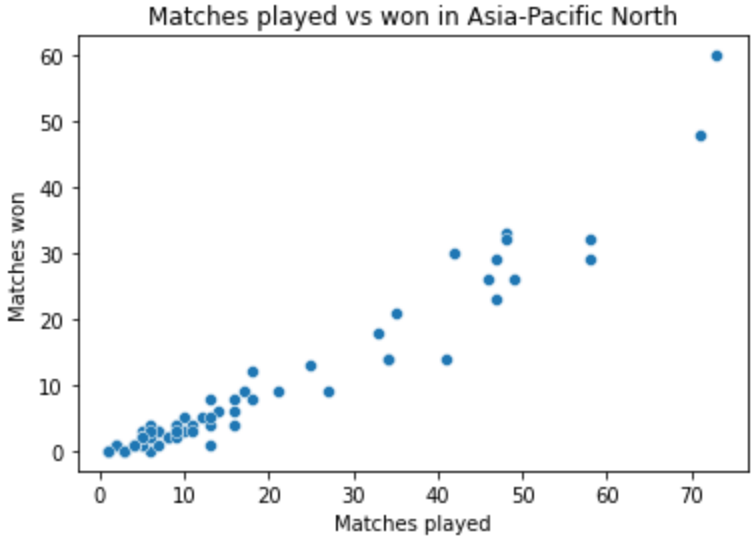
```
[144] team1 = 'JUICERS'
      team2 = 'PULSE CLAN'
      winner = predict_winner(team1, team2)
      print(f"The predicted winner between {team1} and {team2} is {winner}.")

      The predicted winner between JUICERS and PULSE CLAN is PULSE CLAN.
```

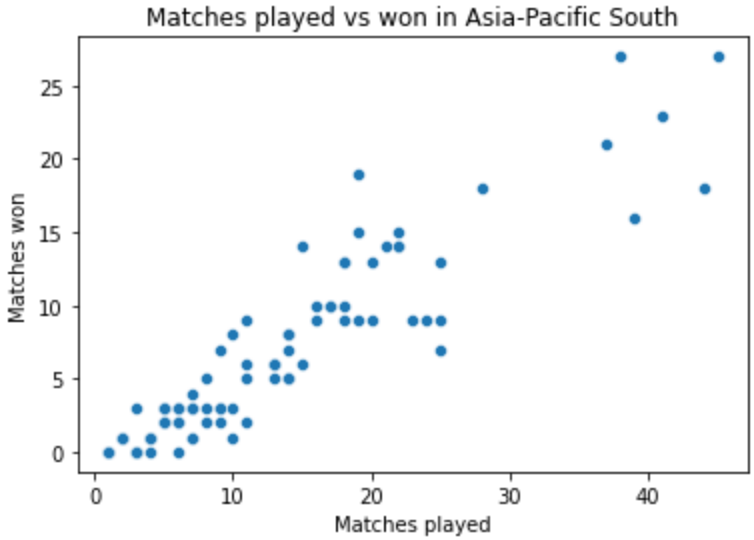
This models are indeed very accurate as per 2021-2022 matches and can also be used to predict the ongoing 2022-2023 match ups.

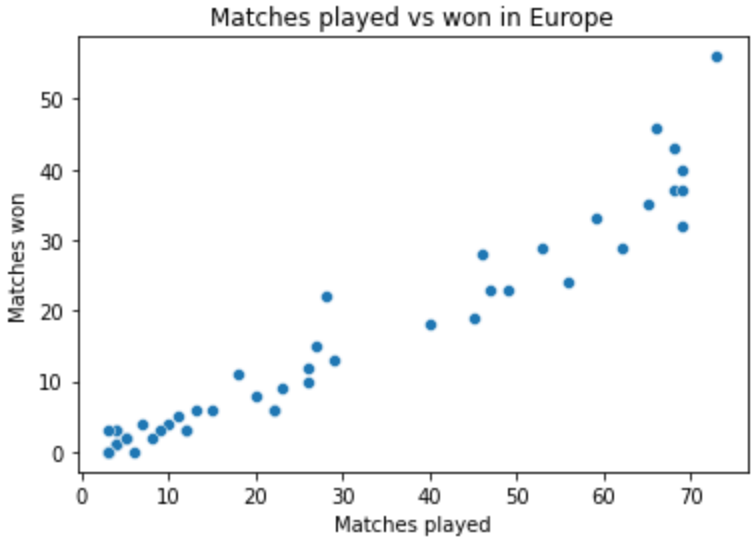
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns


df=pd.read_csv('/content/preprocessed_matches_by_teams.csv')
```

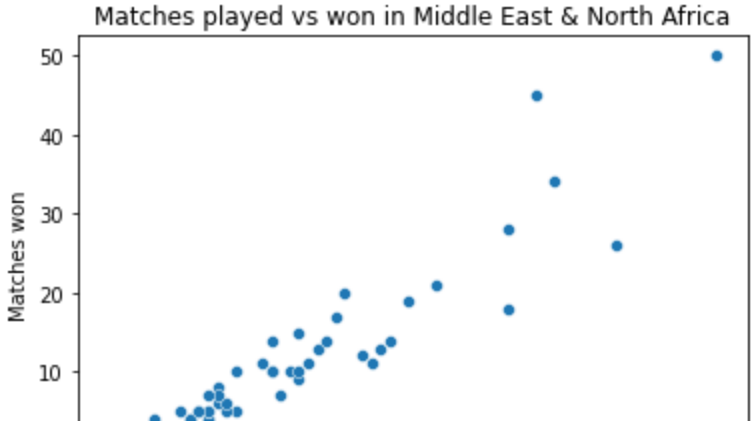
Overview

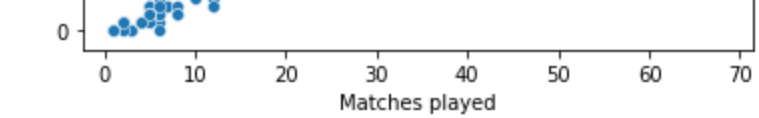
```
# Group data by region and team
grouped = df.groupby(['team_region', 'team_name'])

# Count number of matches played and matches won for each team in each region
played = grouped['match_id'].count().reset_index(name='played')
won = grouped.apply(lambda x: x[x['winner'] == True]['match_id'].count()).reset_index(name
result = played.merge(won, on=['team_region', 'team_name'])

regions = result['team_region'].unique()
for region in regions:
    region_data = result[result['team_region'] == region]
    plt.figure()
    sns.scatterplot(x='played', y='won', data=region_data)
    plt.title(f'Matches played vs won in {region}')
    plt.xlabel('Matches played')
    plt.ylabel('Matches won')

plt.show()
```

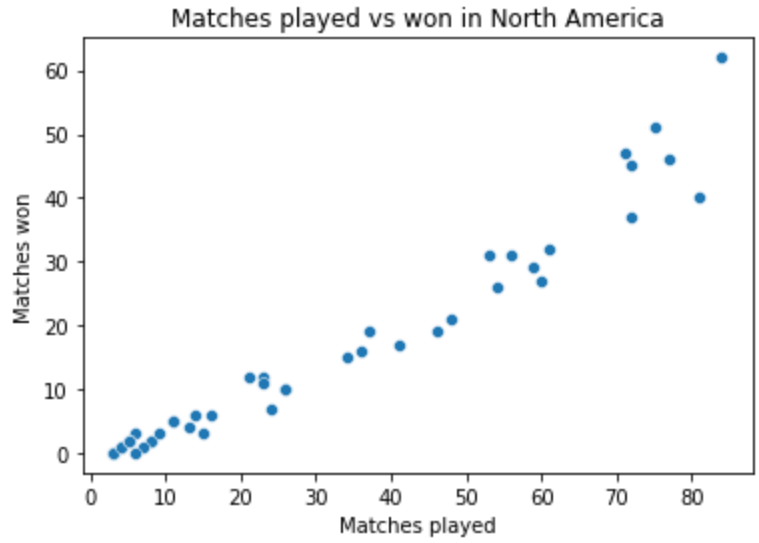

Matches played vs won in Asia-Pacific North
Matches won
Matches played

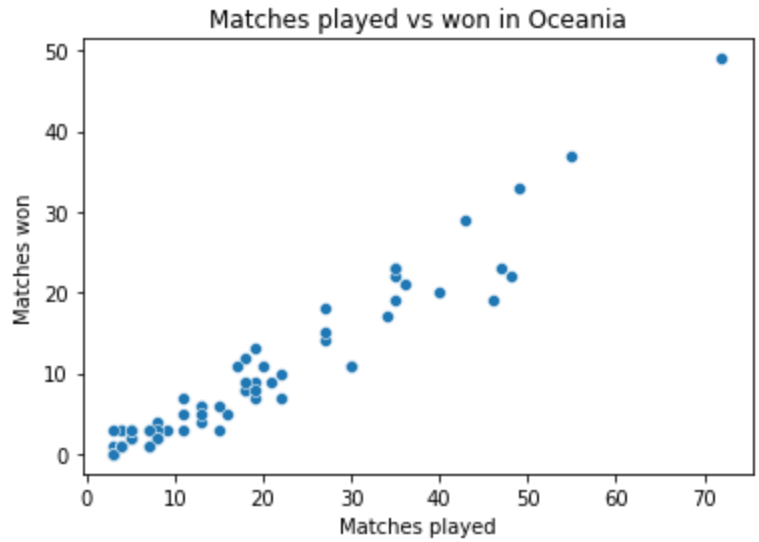

Matches played vs won in Asia-Pacific South
Matches won
Matches played

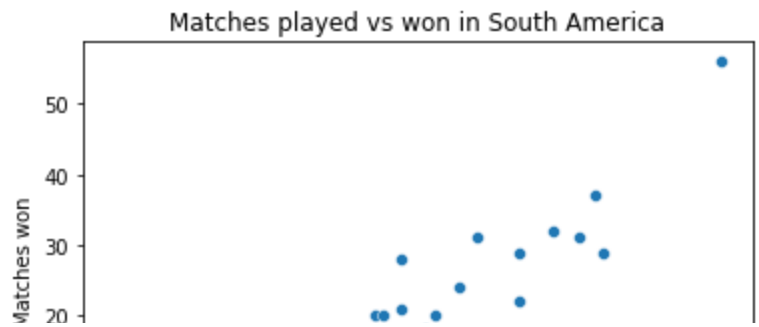

Matches played vs won in Europe
Matches won
Matches played


Matches played vs won in Middle East & North Africa
Matches won







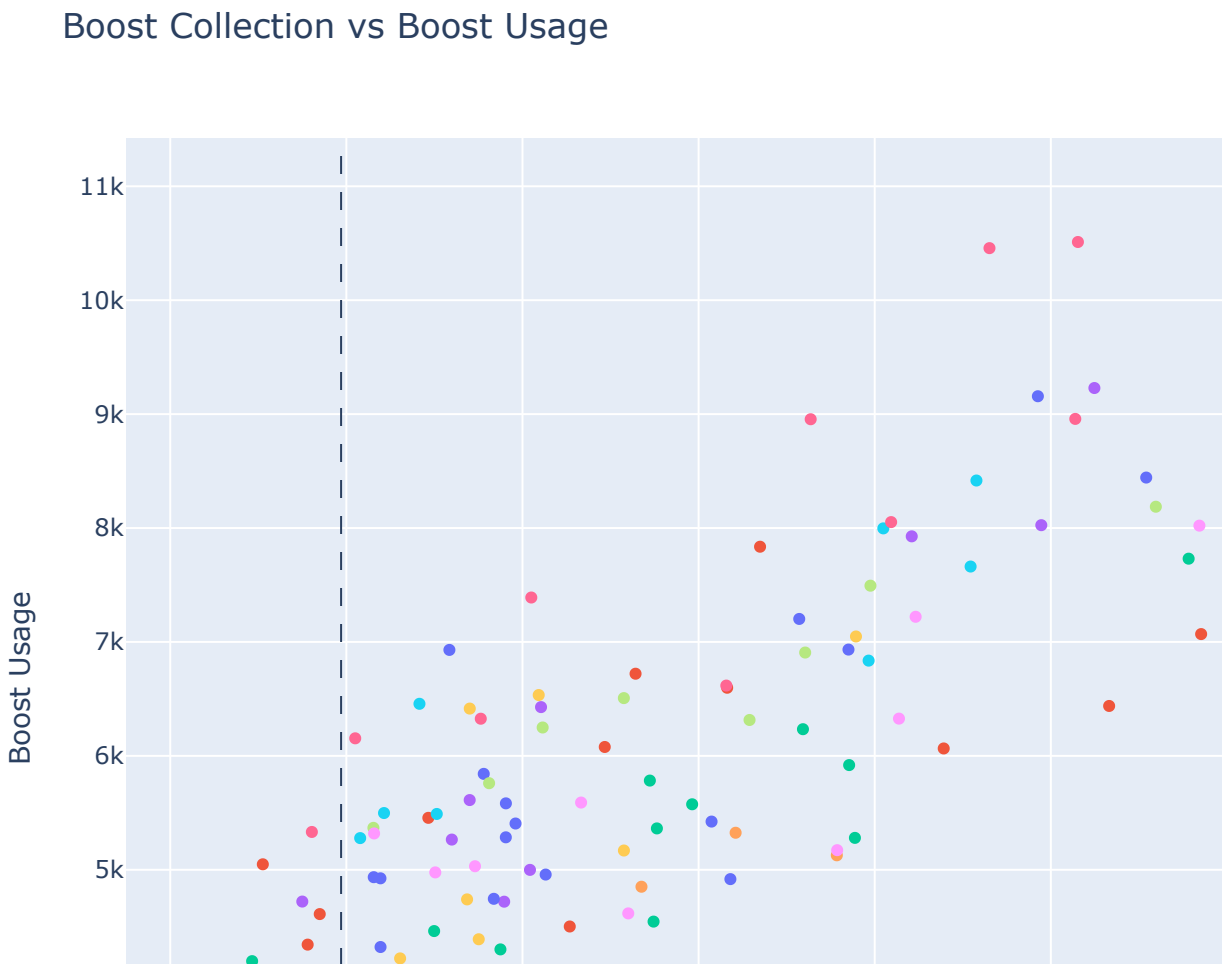



```
import plotly.express as px
```

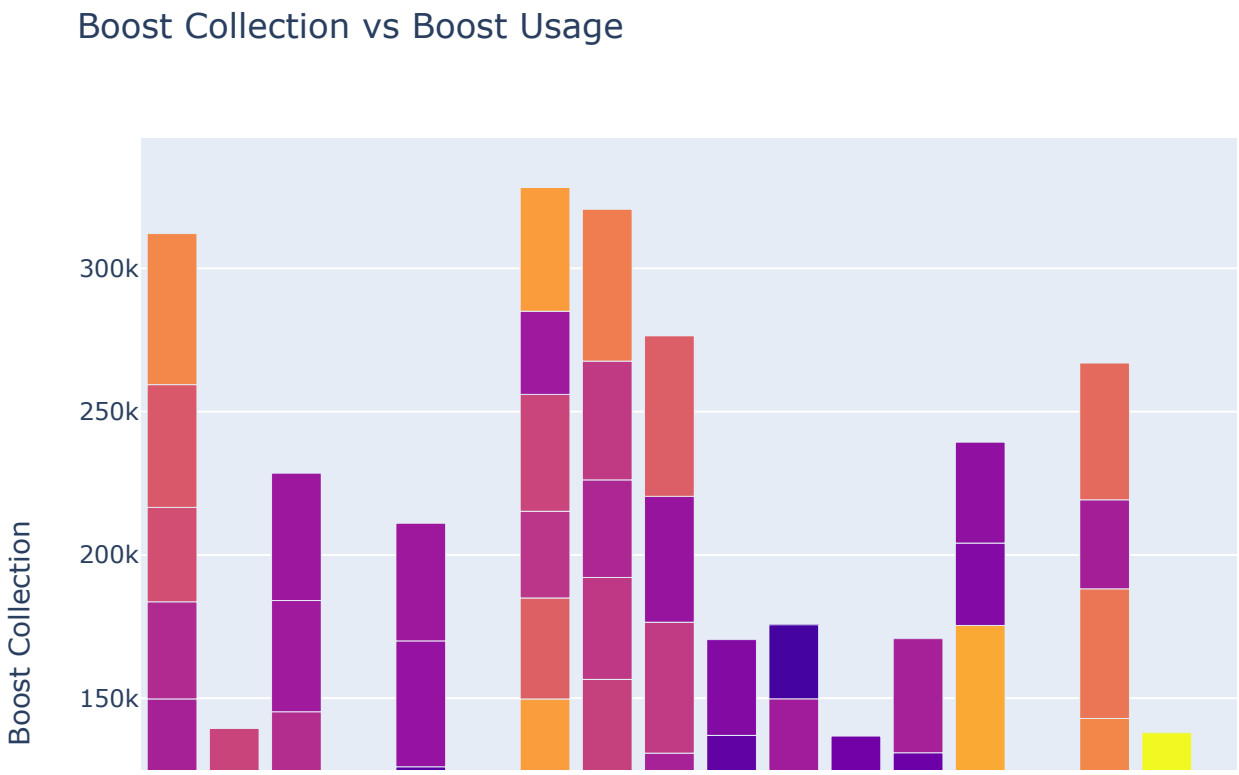
```
worlds=df.loc[df['Mode'] == 'World Championship']
worlds
```

	Unnamed: 0	match_id	color	team_id	
10474	10474	62e022abc437fde7e02da876	blue	6020c1cdf1e4807cc7025d03	https:
10475	10475	62e022abc437fde7e02da876	orange	60bb483e6d9c1362119ca7cd	https
10476	10476	62e022abc437fde7e02da877	blue	6020bc70f1e4807cc70023fb	https
10477	10477	62e022abc437fde7e02da877	orange	61845f24f8090ec74528af10	https
10478	10478	62e022abc437fde7e02da878	blue	6020bc70f1e4807cc70023e9	https
...	...	...	...	...	
10589	10589	62e02371da9d7ca1c7bb2f61	orange	6020bd98f1e4807cc700dc74	https
10590	10590	62e02371da9d7ca1c7bb2f62	blue	6020bc70f1e4807cc70023a5	https:
10591	10591	62e02371da9d7ca1c7bb2f62	orange	605aca6853a71a78eacbc155	https
10592	10592	62e02371da9d7ca1c7bb2f63	blue	6020bd98f1e4807cc700dc74	https

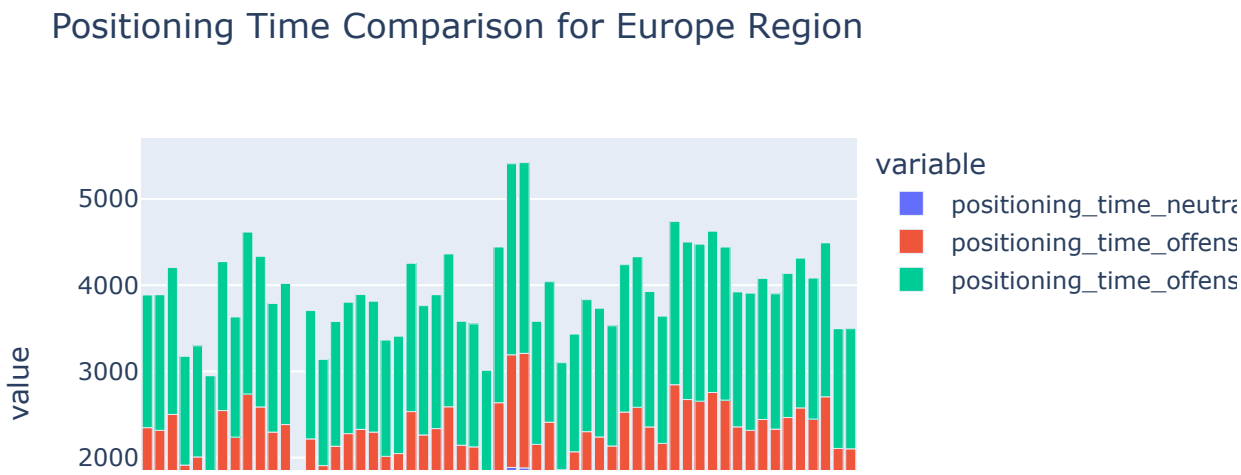
```
fig = px.scatter(worlds, x='boost_amount_collected', y='boost_amount_used_while_supersonic
                 color='team_name', hover_data=['team_name'],
                 labels={'boost_amount_collected':'Boost Collection',
                         'boost_amount_used_while_supersonic':'Boost Usage',
                         'team_name': 'Team Name'},
                 title='Boost Collection vs Boost Usage',
                 width=960, height=720)

fig.add_vline(x=df['boost_amount_collected'].mean(), line_width=1, line_dash="dash")
fig.add_hline(y=df['boost_amount_used_while_supersonic'].mean(), line_width=1, line_dash='

fig.show()
```




```
fig = px.bar(worlds, x='team_name', y='boost_amount_collected', color='boost_amount_used_w
             labels={'boost_amount_collected':'Boost Collection',
                     'boost_amount_used_while_supersonic':'Boost Usage',
                     'team_name': 'Team Name'},
             title='Boost Collection vs Boost Usage',
             width=960, height=720)

fig.show()
```

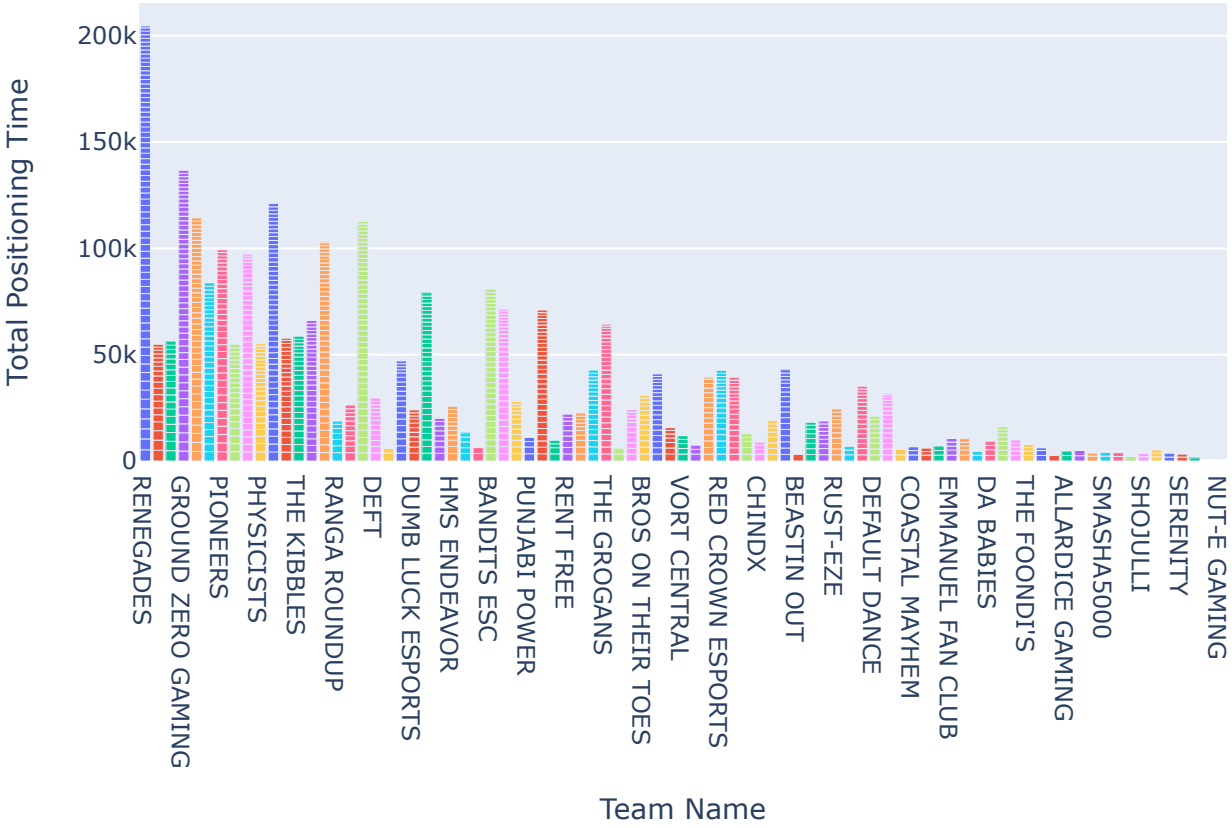



```
# filter data for Europe region
europe = df[df['team_region'] == 'Europe']

# pivot the data to have team_name as columns and positioning times as values
df_pivot = pd.pivot_table(europe, values=['positioning_time_neutral_third', 'positioning_t
index='team_name', aggfunc='mean')

# plot the bar graph
fig = px.bar(df_pivot, x=df_pivot.index, y=['positioning_time_neutral_third', 'positioning
             labels={'positioning_time_neutral_third':'Positioning Time Neutral Third',
                     'positioning_time_offensive_third':'Positioning Time Offensive Third'
                     'positioning_time_offensive_half': 'Positioning Time Offensive Half',
                     'team_name':'Team Name'},width=700, height=520,
             title='Positioning Time Comparison for Europe Region')

fig.show()
```

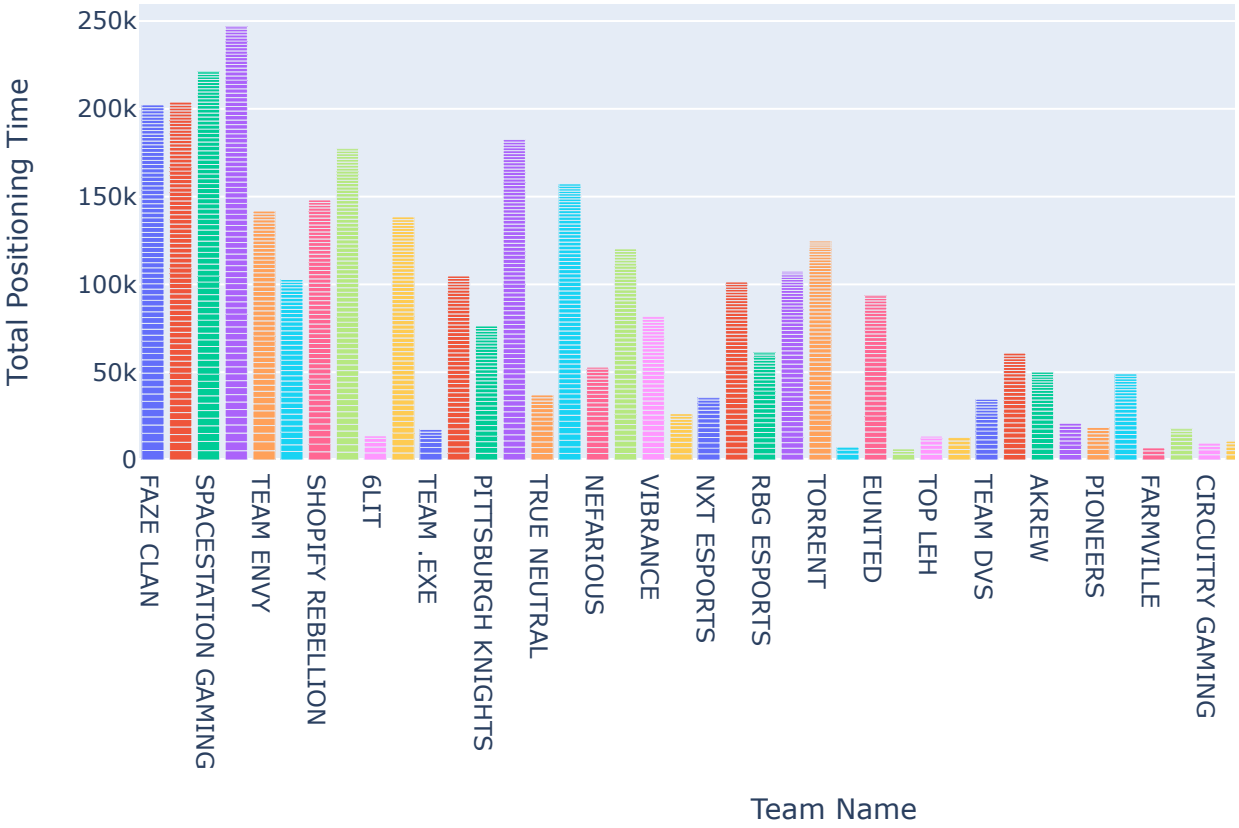



```
regions = ['Oceania', 'North America', 'Asia-Pacific South','Middle East & North Africa',

# Create a new column for total positioning time for each region
df['total_positioning_time'] = df['positioning_time_offensive_third'] + df['positioning_ti

for region in regions:
    region_df = df[df['team_region'] == region]
    region_df=region_df.sort_values(by='total_positioning_time', ascending=False)
    fig = px.bar(region_df, x='team_name', y='total_positioning_time',
                 color='team_name', hover_data=['team_name'],
                 labels={'total_positioning_time':'Total Positioning Time',
                         'team_name': 'Team Name'},width=1000, height=520,
                 title=f'Total Positioning Time for {region} Region')
    fig.show()
```

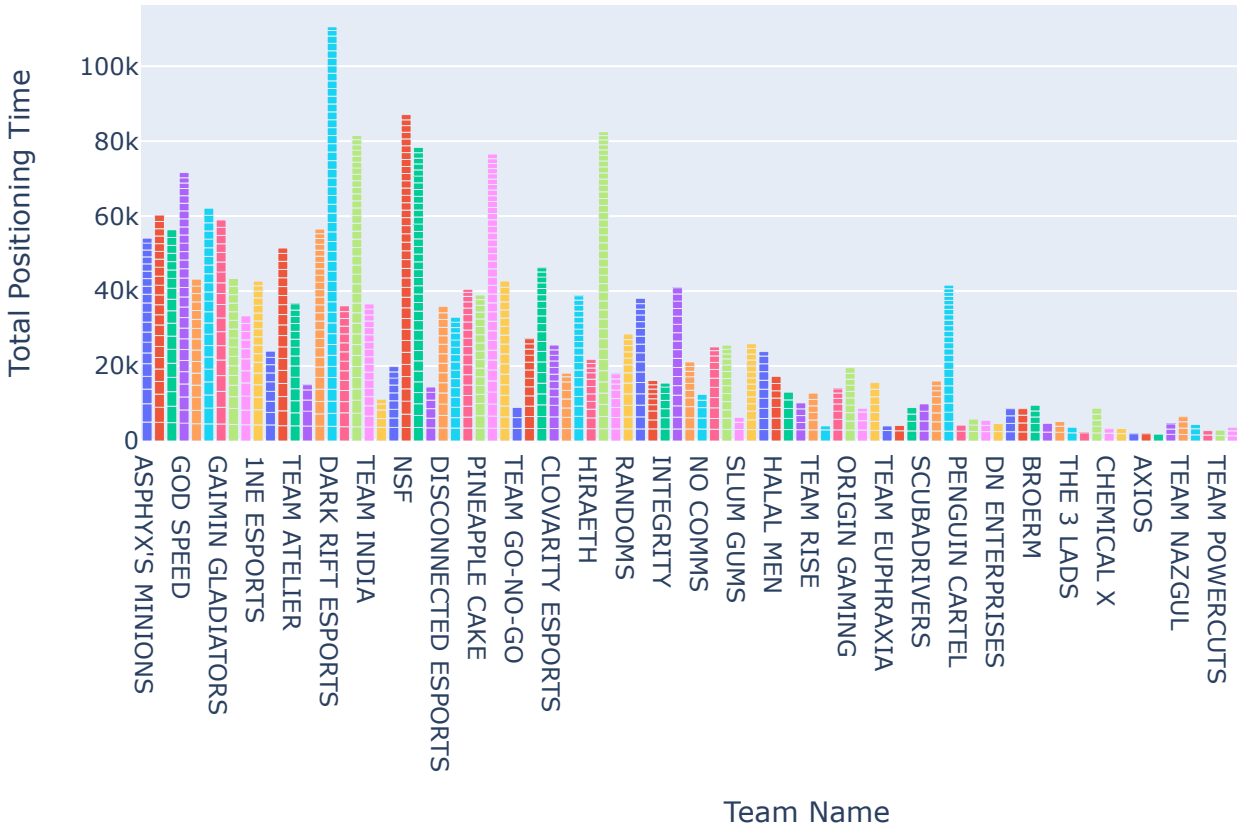
Total Positioning Time for Oceania Region



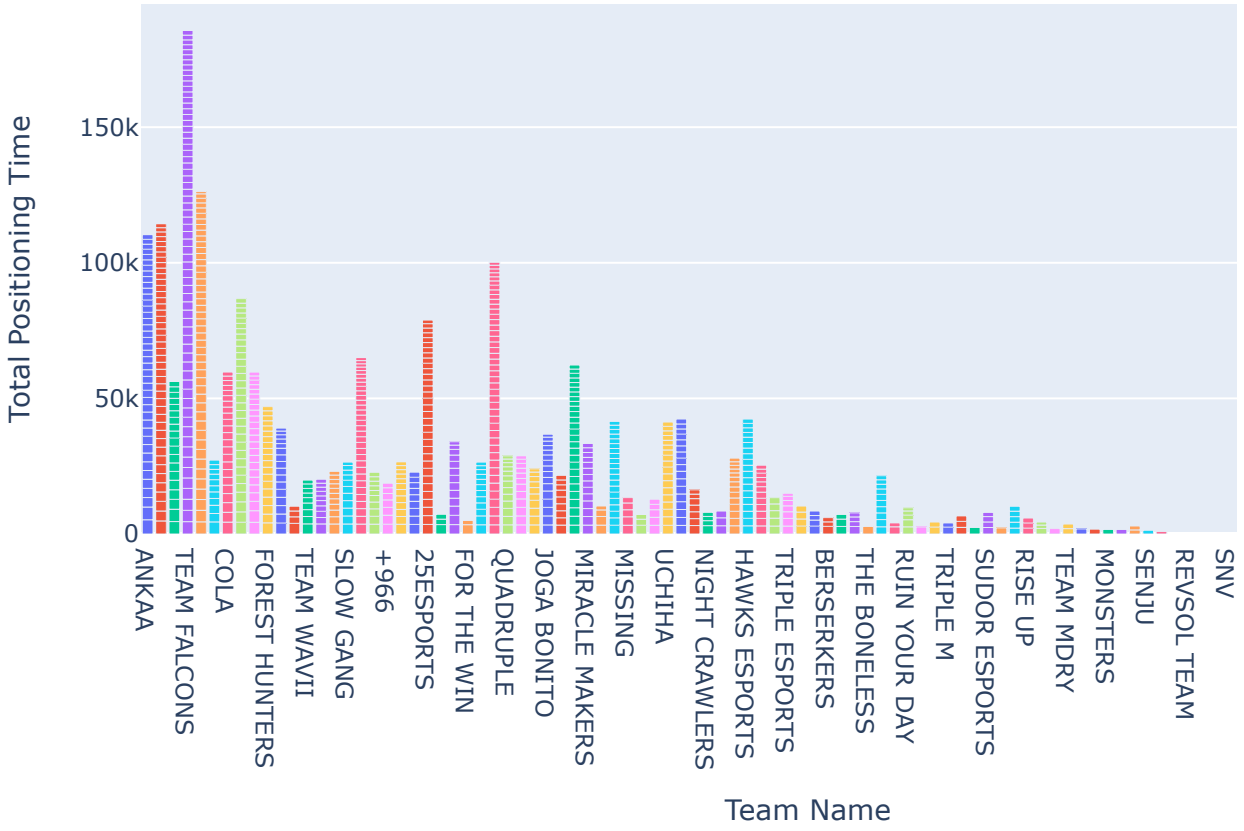
Total Positioning Time for North America Region



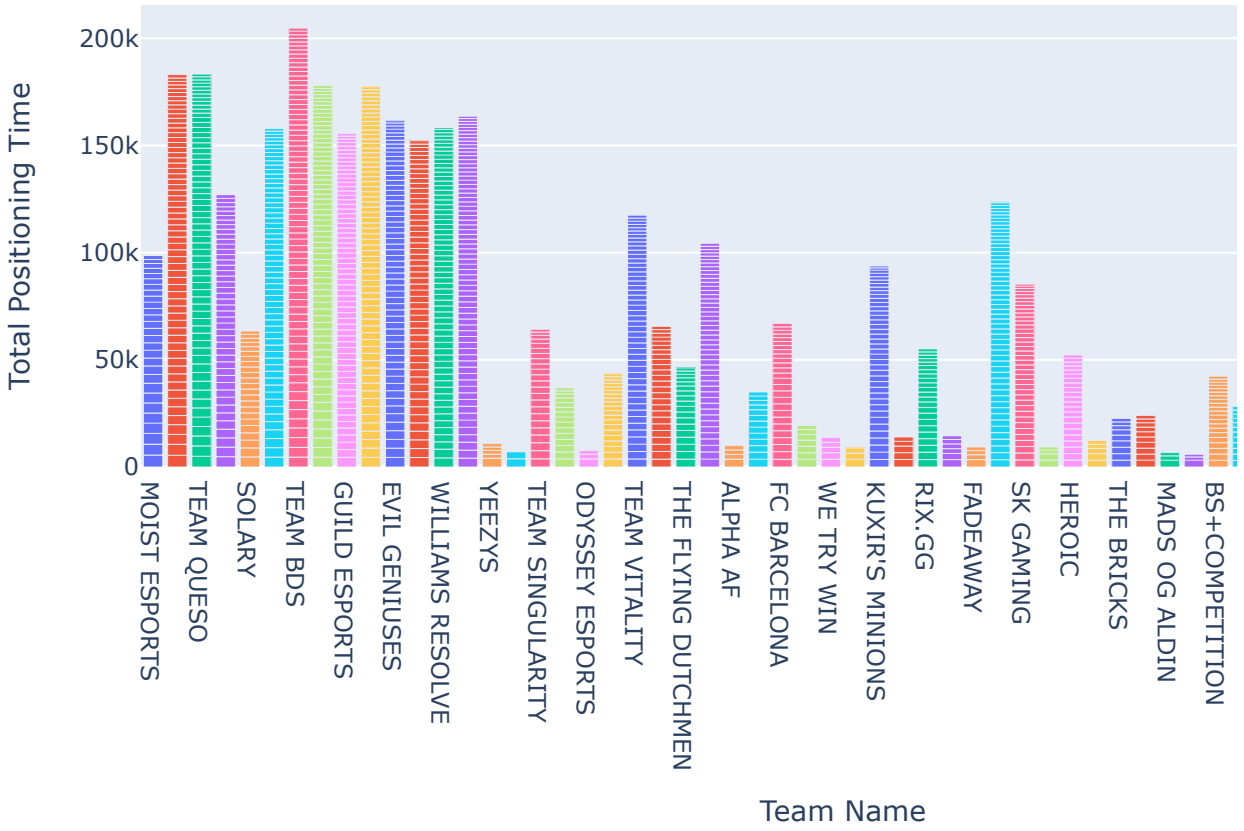
Total Positioning Time for Asia-Pacific South Region



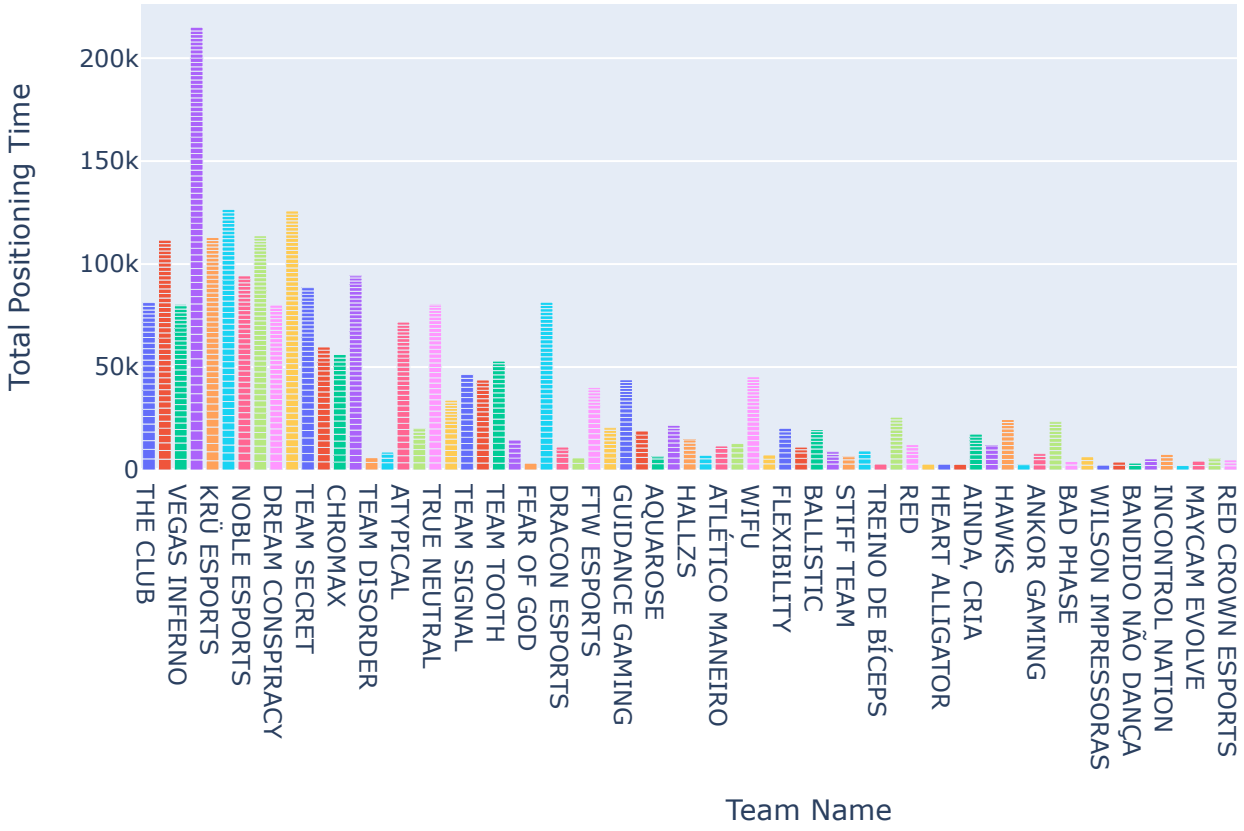
Total Positioning Time for Middle East & North Africa Region



Total Positioning Time for Europe Region

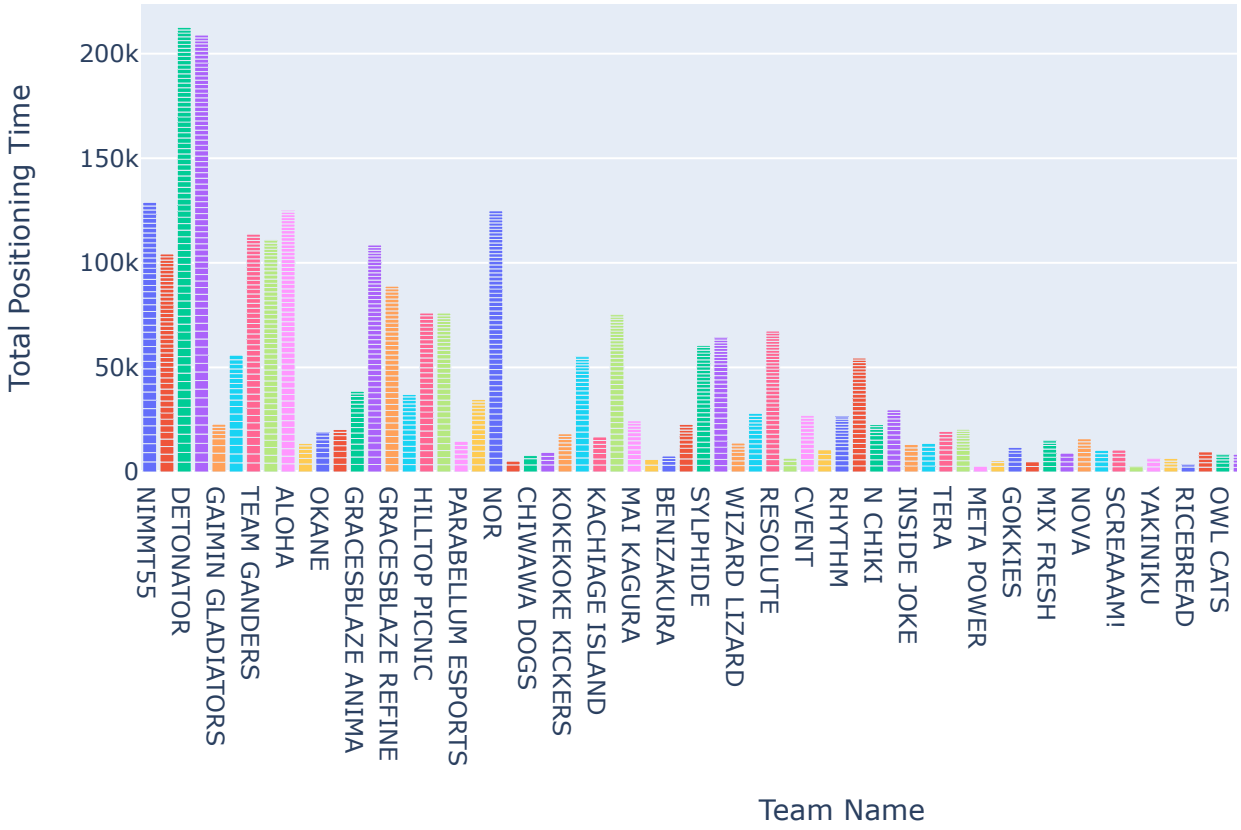


Total Positioning Time for South America Region



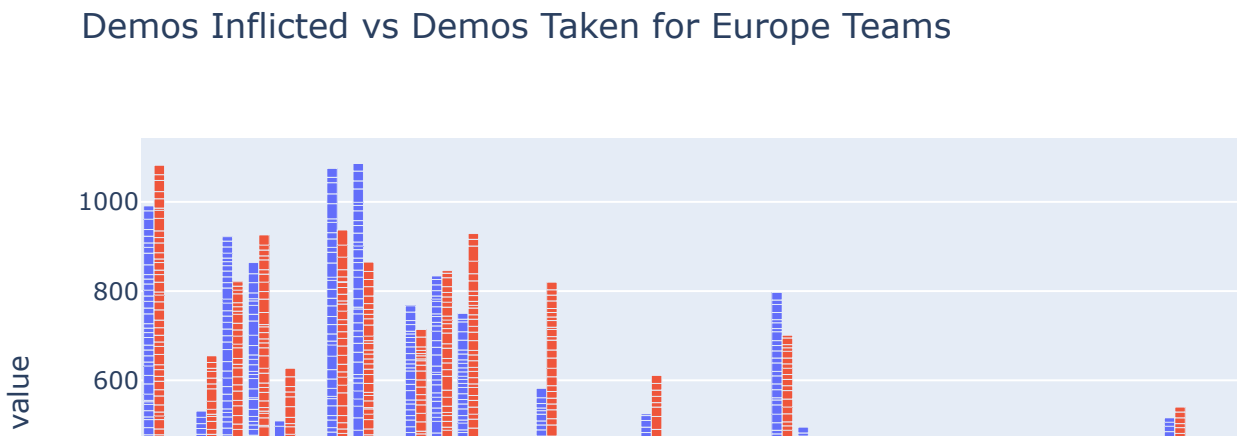
Total Positioning Time for Asia-Pacific North Region

Total Positioning Time for Asia-Pacific North Region



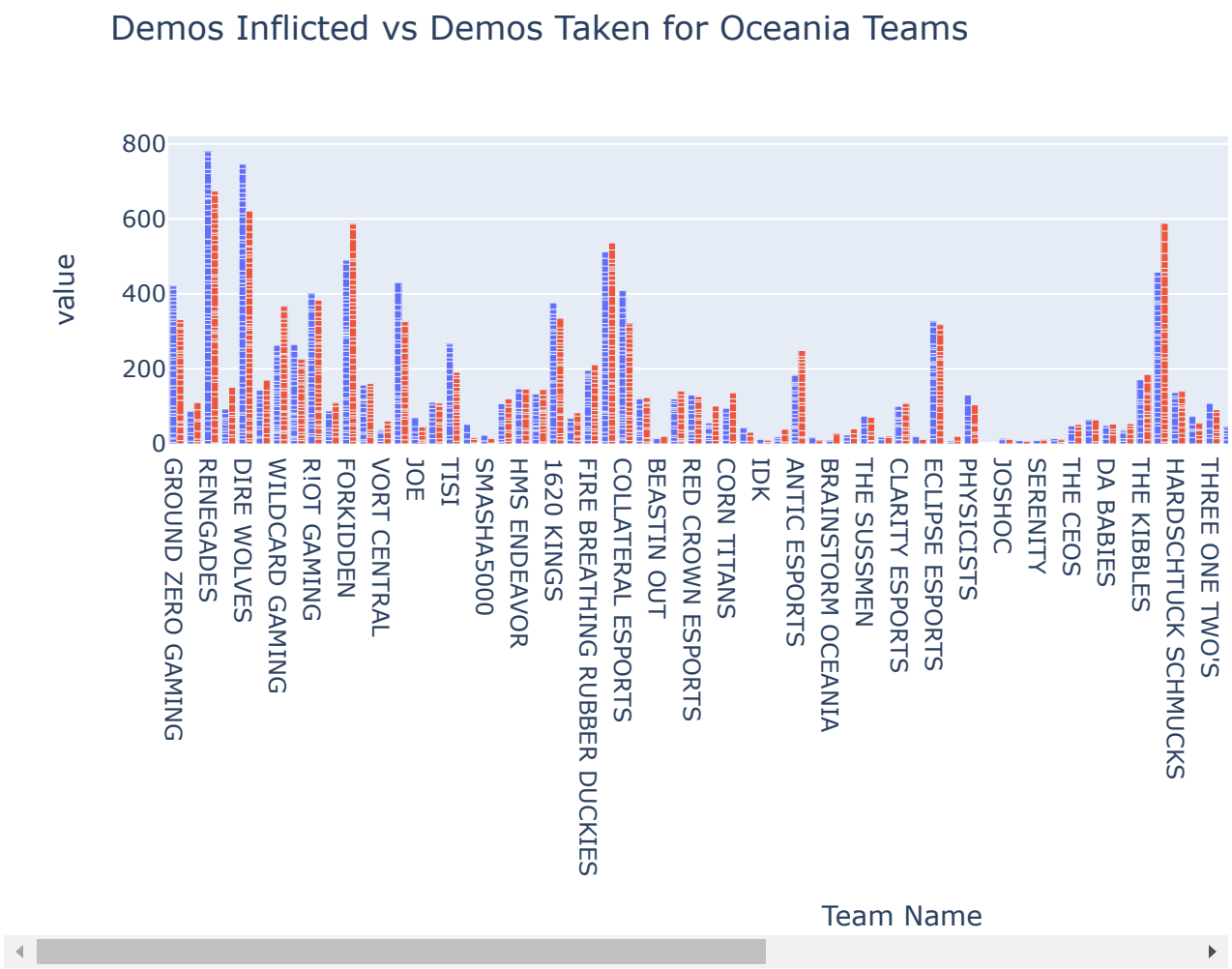
Total Positioning Time for Sub-Saharan Africa Region

```
europe_teams = df[df['team_region'] == 'Europe']
fig = px.bar(europe_teams, x='team_name', y=['demo_inflicted', 'demo_taken'],
barmode='group',
labels={'demo_inflicted':'Demos Inflicted',
'demo_taken':'Demos Taken',
'team_name': 'Team Name'},width=1000, height=520,
title='Demos Inflicted vs Demos Taken for Europe Teams')
fig.show()
```



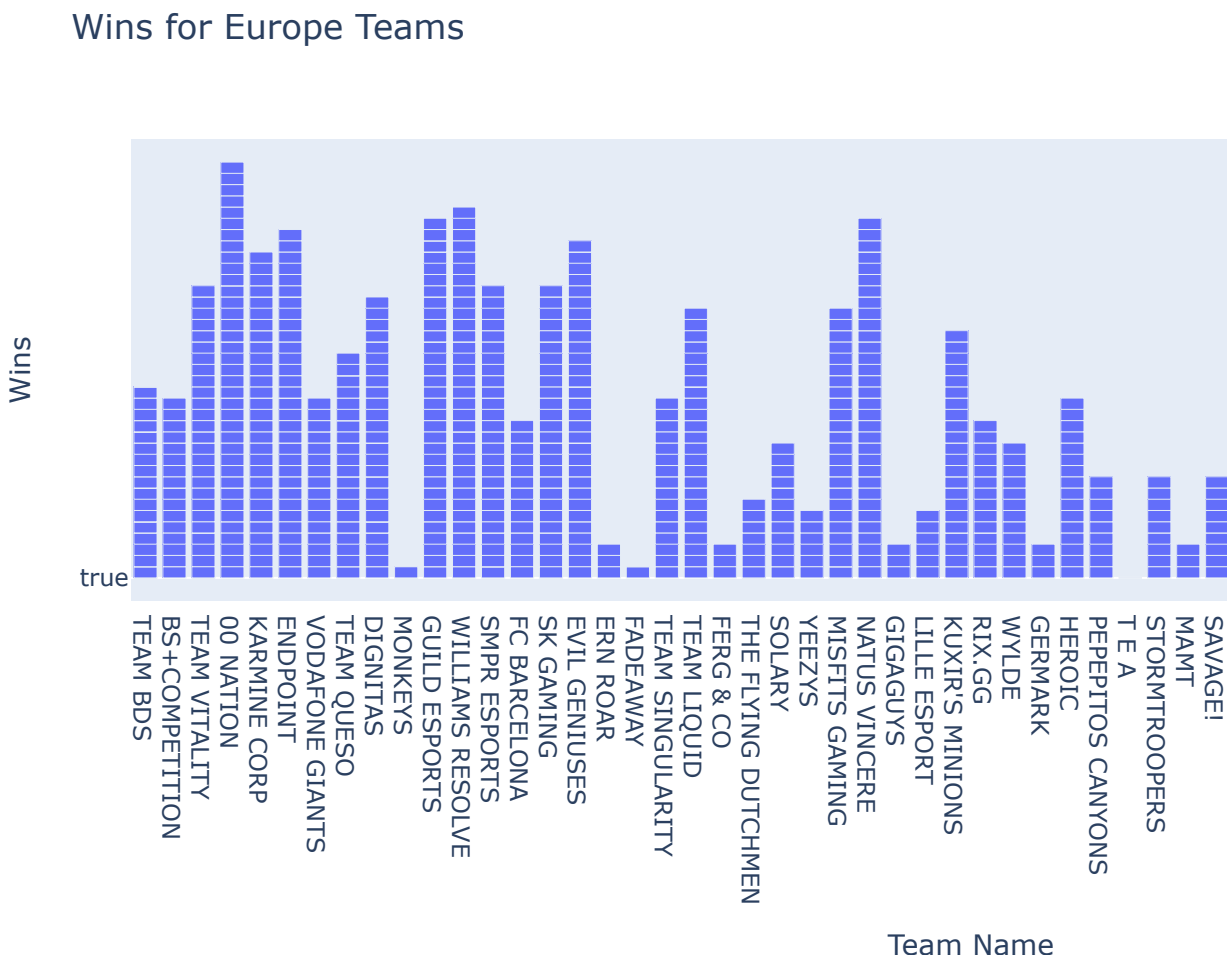

```
region = 'Europe'
region_df = worlds[worlds['team_region'] == region]
region_df['total_demo'] = region_df['demo_inflicted'] + region_df['demo_taken']
region_df = region_df.sort_values(by='total_demo', ascending=False)

fig = px.bar(region_df, x='team_name', y=['demo_inflicted', 'demo_taken'], color='team_nam
             labels={'demo_inflicted':'Demos Inflicted',
                     'demo_taken':'Demos Taken',
                     'team_name': 'Team Name'},width=700, height=520,
             title=f'Demos Inflicted and Taken by Teams in {region} Region')

fig.show()
```

```
regions = ['Oceania', 'North America', 'Asia-Pacific South',
'Middle East & North Africa', 'Europe', 'South America',
'Asia-Pacific North', 'Sub-Saharan Africa']

for region in regions:
  region_teams = df[df['team_region'] == region]
  fig = px.bar(region_teams, x='team_name', y=['demo_inflicted', 'demo_taken'],
  barmode='group',width=1000, height=520,
  labels={'demo_inflicted':'Demos Inflicted',
  'demo_taken':'Demos Taken',
  'team_name': 'Team Name'},
  title=f'Demos Inflicted vs Demos Taken for {region} Teams')
  fig.show()
  break
```

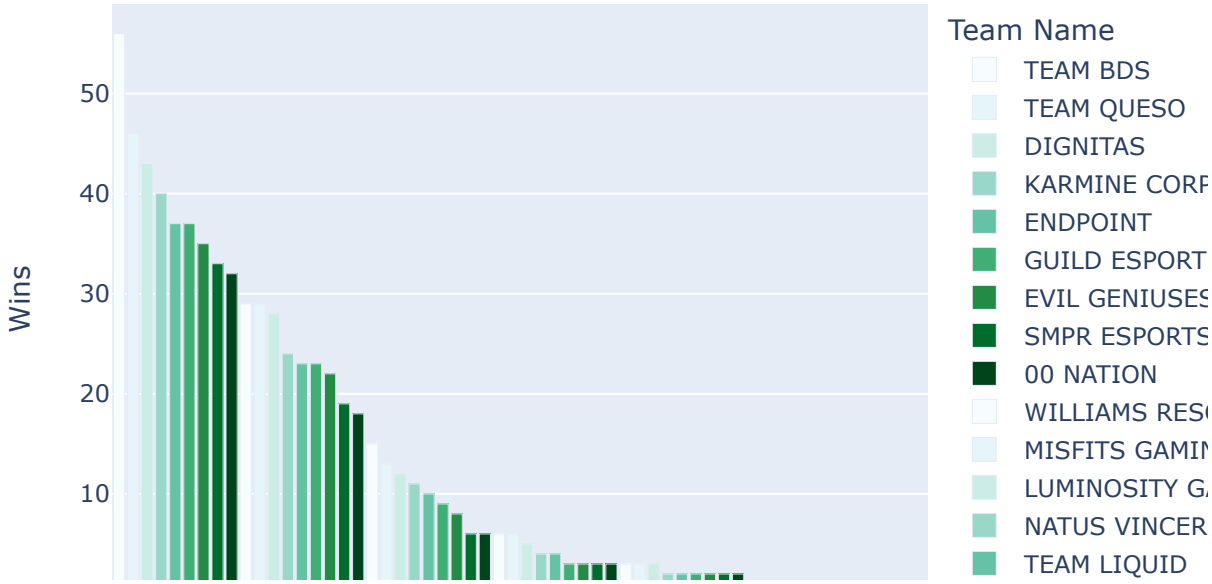



```
europe_teams = df[df['team_region'] == 'Europe']
fig = px.bar(europe_teams, x='team_name', y='winner',
             labels={'winner':'Wins',
                     'team_name': 'Team Name'},width=1000, height=520,
             title='Wins for Europe Teams')
```

```
fig.show()
```

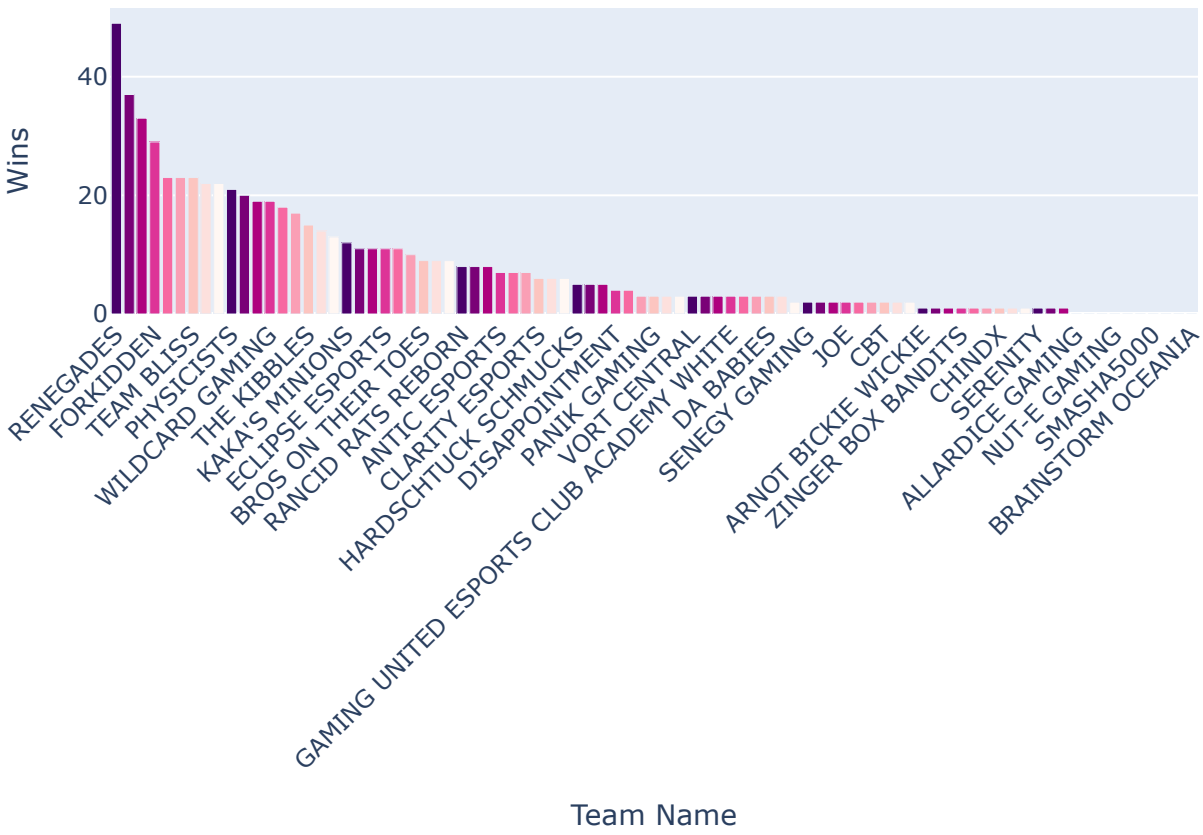



```
euro_teams = df[df['team_region'] == 'Europe']
# Count the number of wins for each team
team_wins = euro_teams.groupby('team_name')['winner'].sum().reset_index()

# Sort the teams by number of wins
team_wins = team_wins.sort_values(by='winner', ascending=False)

# Plot the results using plotly.express
fig = px.bar(team_wins, x='team_name', y='winner',
             labels={'team_name':'Team Name',
                         'winner': 'Wins'},width=700, height=520,
            hover_data=['team_name'], color='team_name',color_discrete_sequence=px.colors.
fig.show()
```

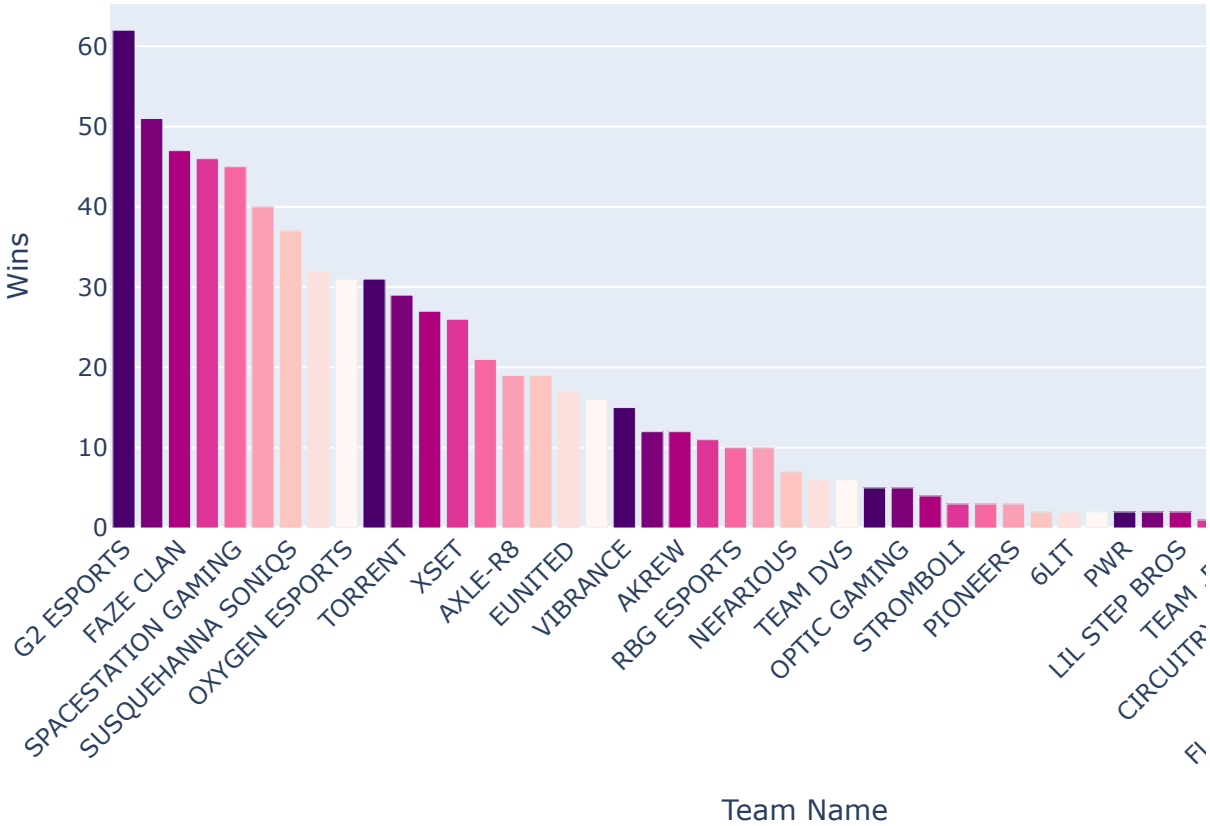



```
regions_wins=df['team_region'].unique()
top_teams = pd.DataFrame()
for i in regions_wins:
  team = df[df['team_region'] == i]

  # Count the number of wins for each team
  team_wins = team.groupby('team_name')['winner'].sum().reset_index()
  # Sort the teams by number of wins
  team_wins = team_wins.sort_values(by='winner', ascending=False)
  top_10 = team_wins.head(10)
  top_teams = pd.concat([top_teams, team[team['team_name'].isin(top_10['team_name'])]])
  # Plot the results using plotly.express
  fig = px.bar(team_wins, x='team_name', y='winner',
              labels={'team_name':'Team Name',
                          'winner': 'Wins'},width=1000, height=520,
              hover_data=['team_name'], color='team_name',color_discrete_sequence=px.color
               title=f'Wins for {i} Teams')
  fig.update_layout(xaxis_tickangle=-45)
  fig.show()
```

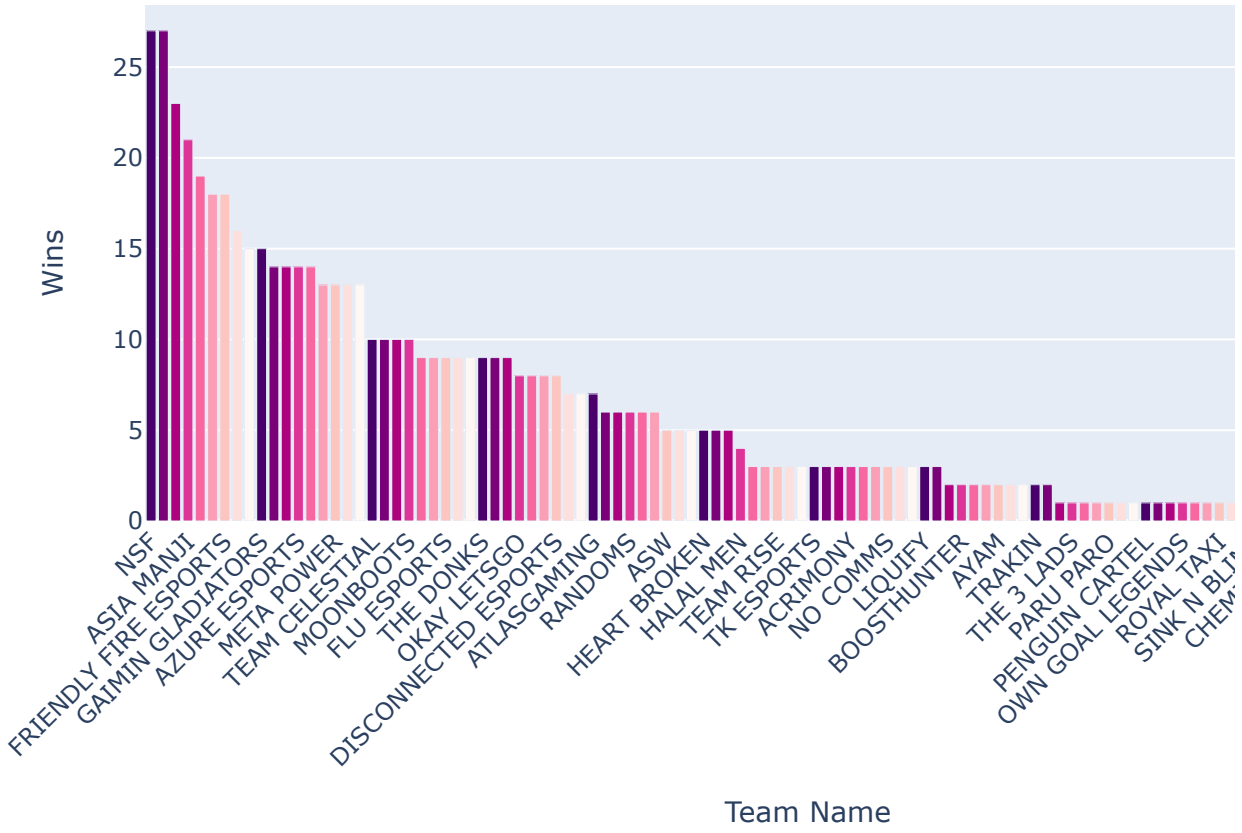
Wins for Oceania Teams



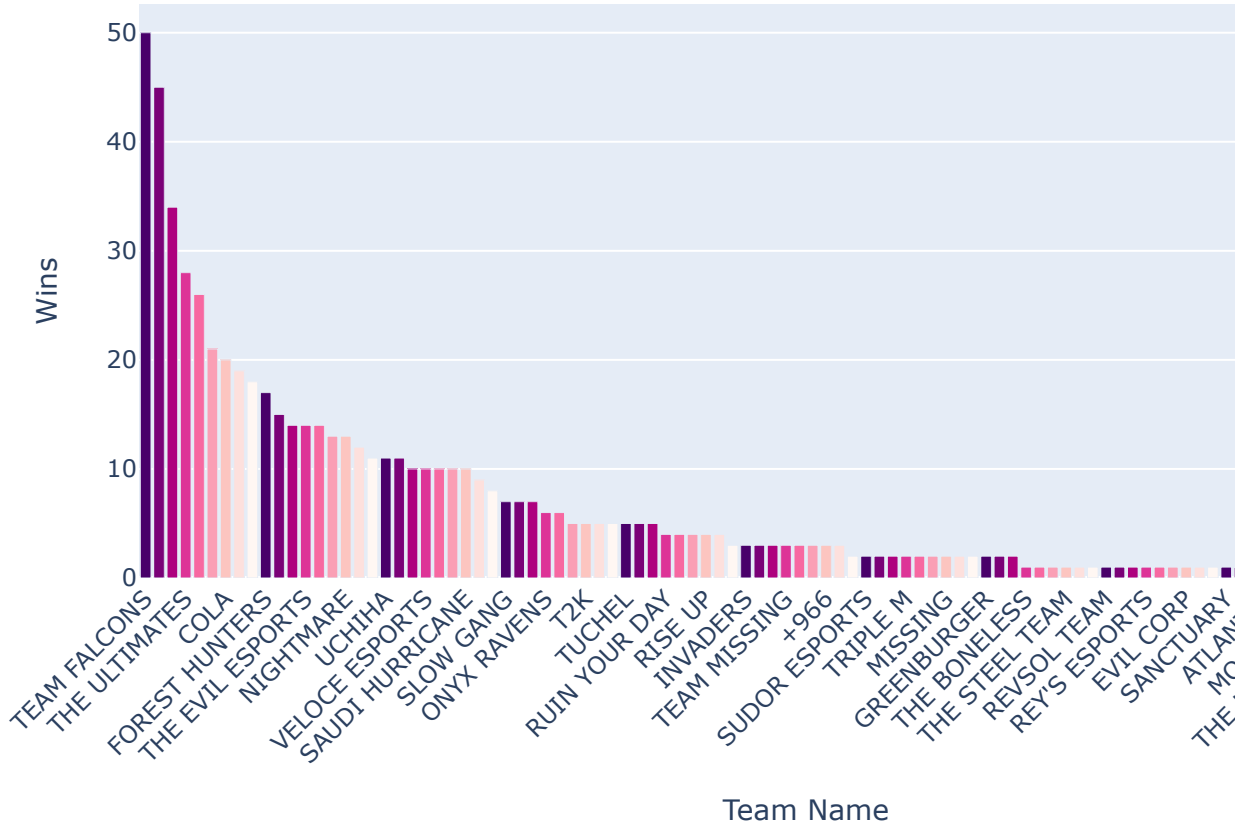
Wins for North America Teams



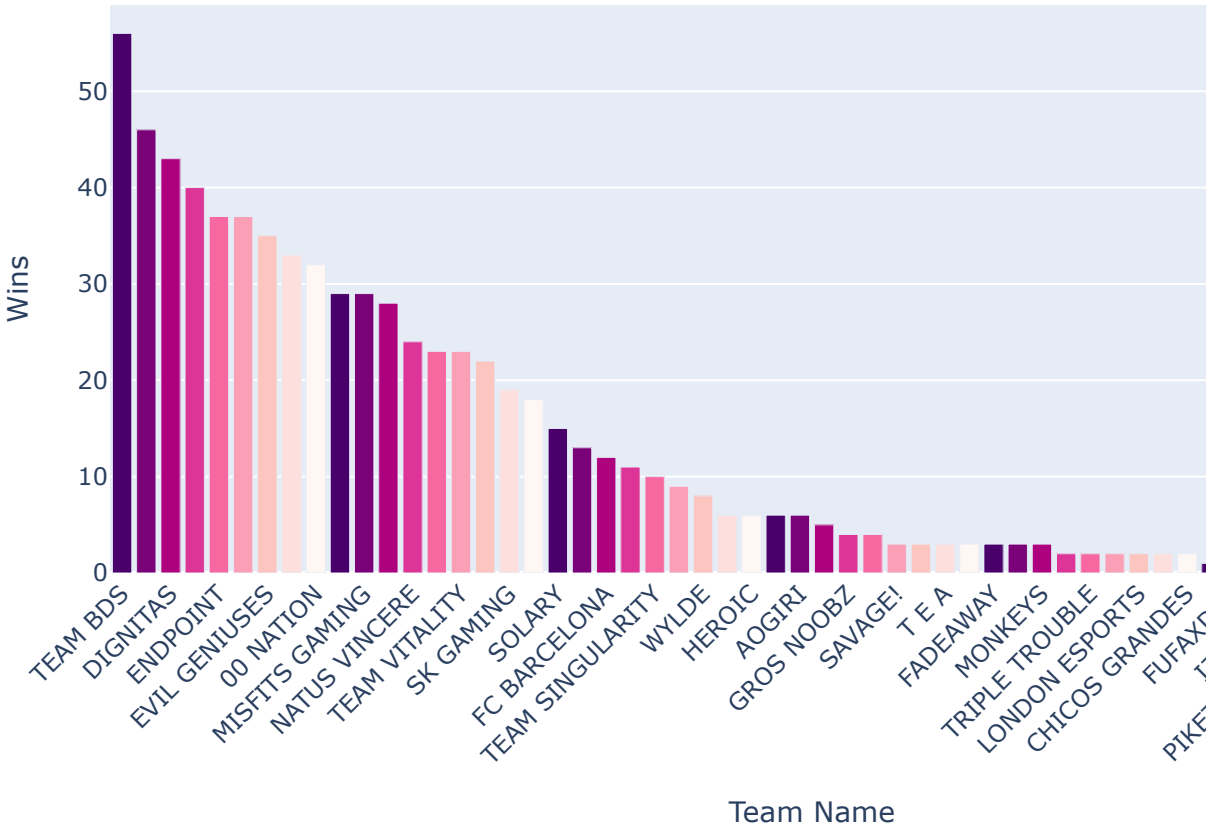
Wins for Asia-Pacific South Teams



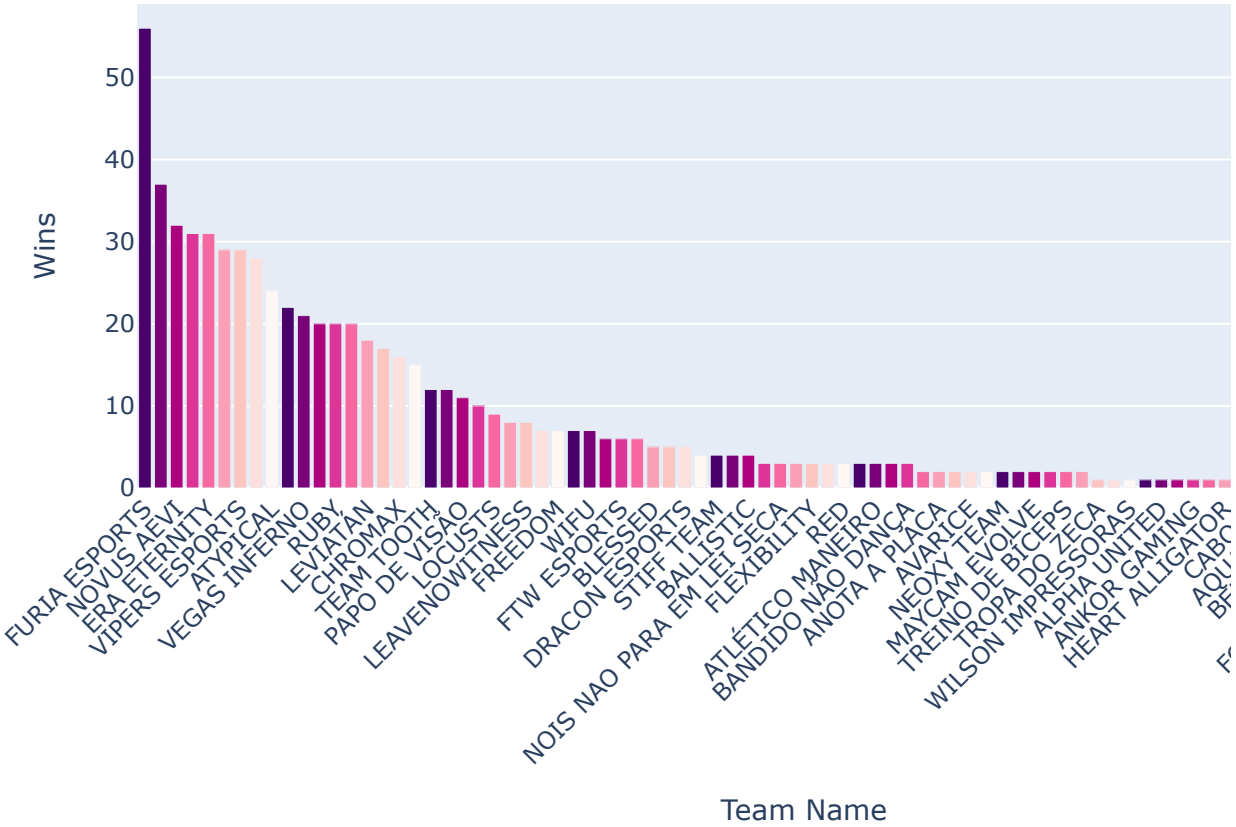
Wins for Middle East & North Africa Teams



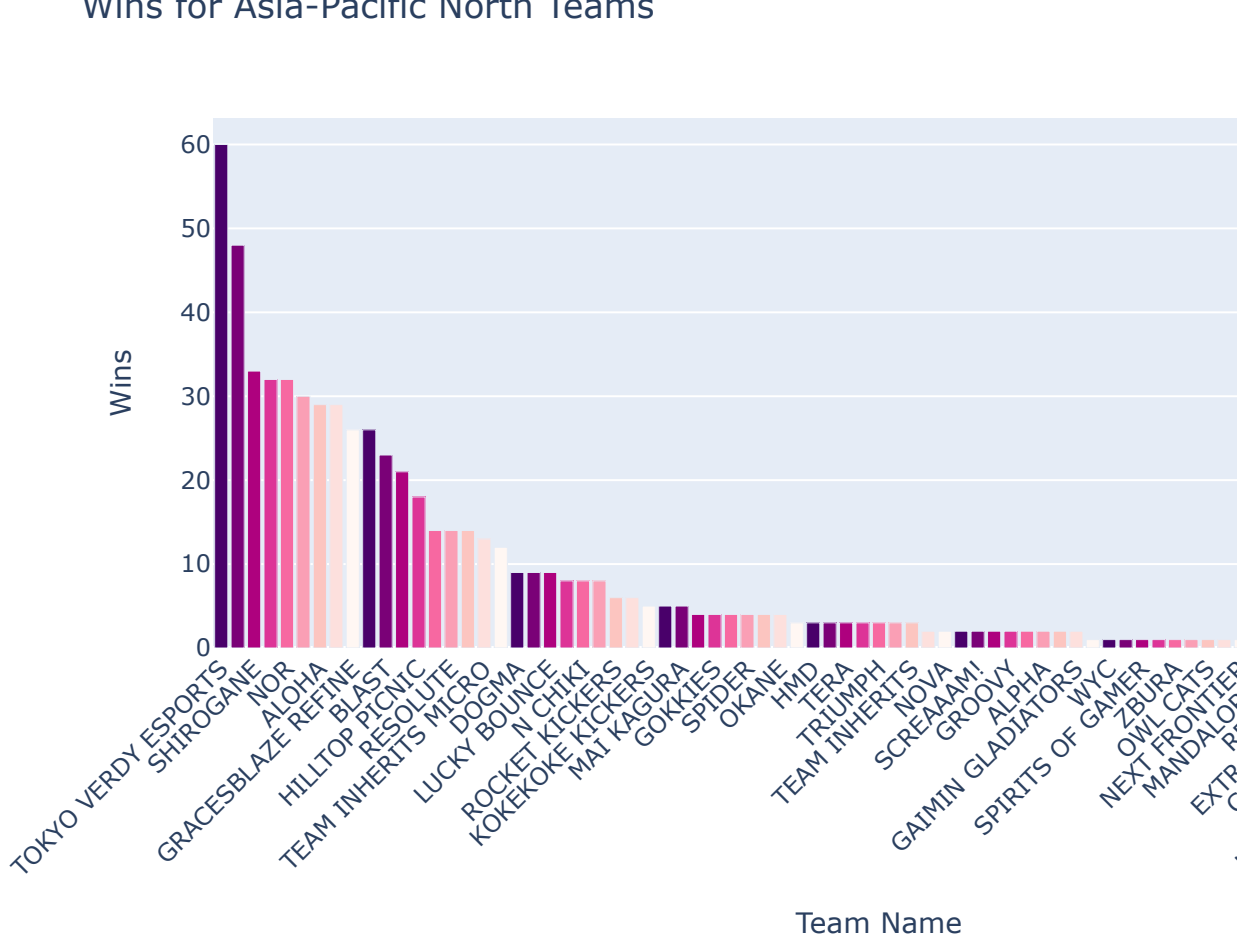
Wins for Europe Teams

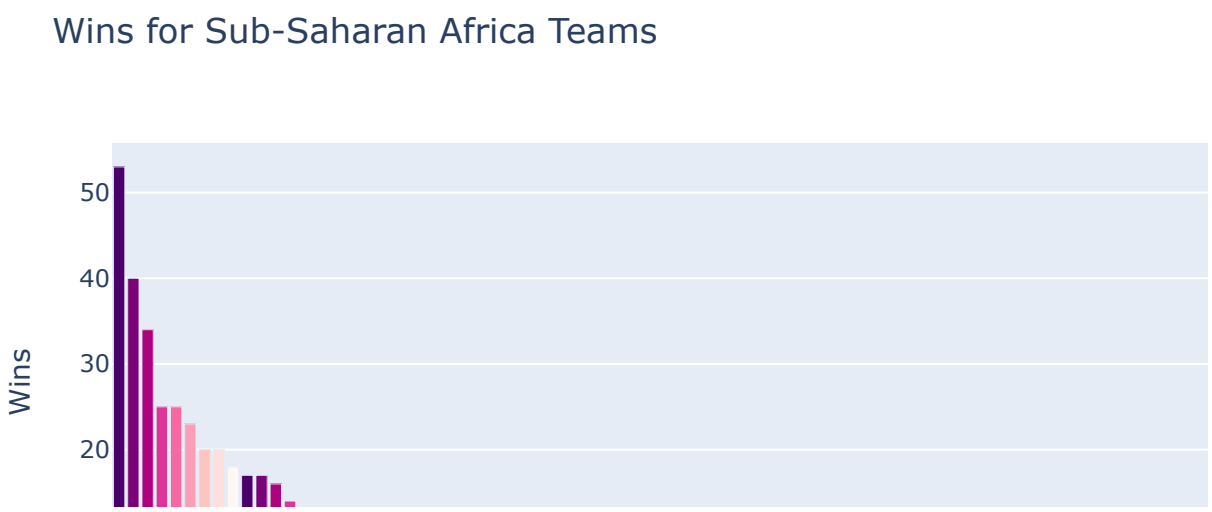


Wins for South America Teams

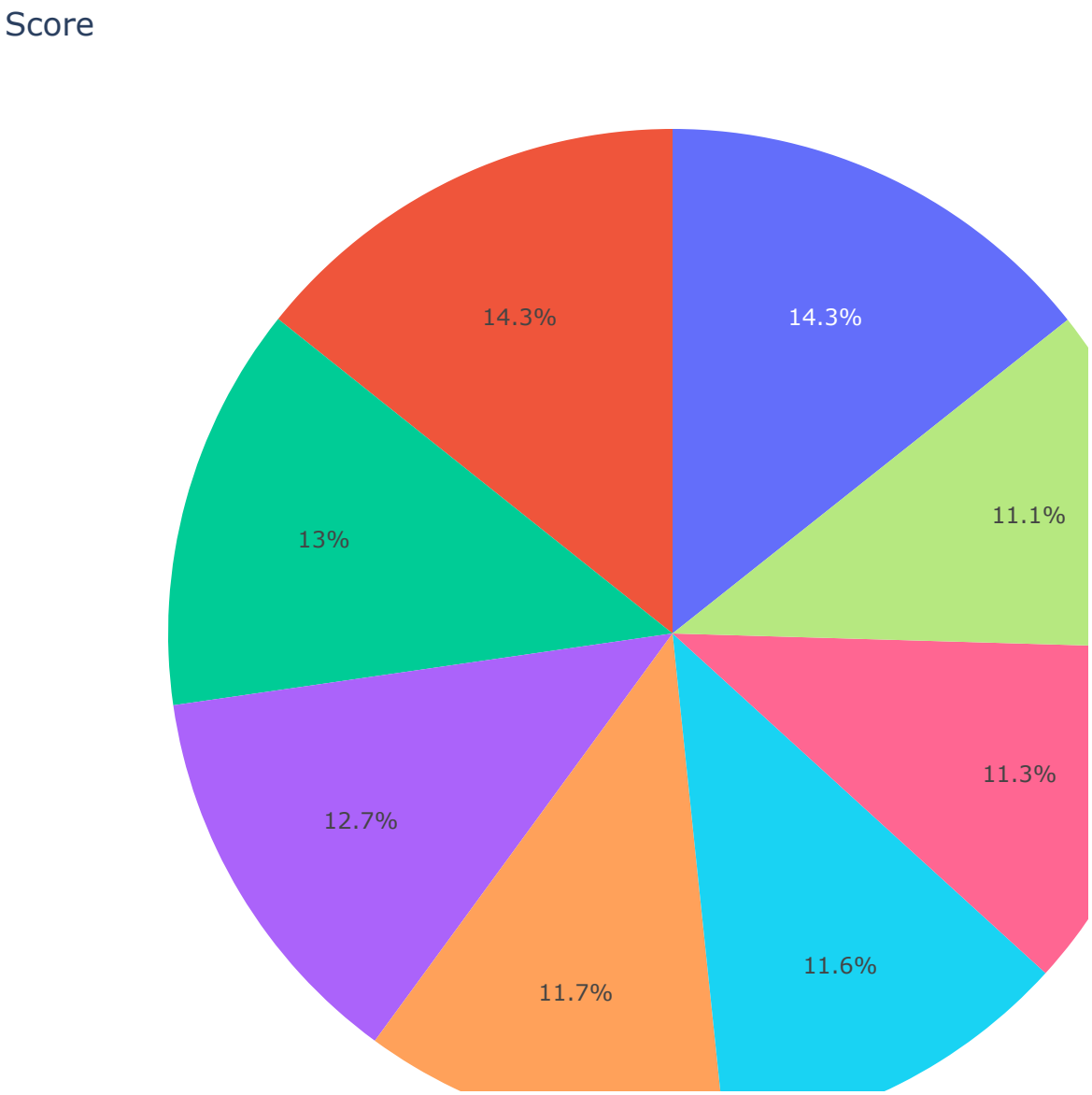


Wins for Asia-Pacific North Teams

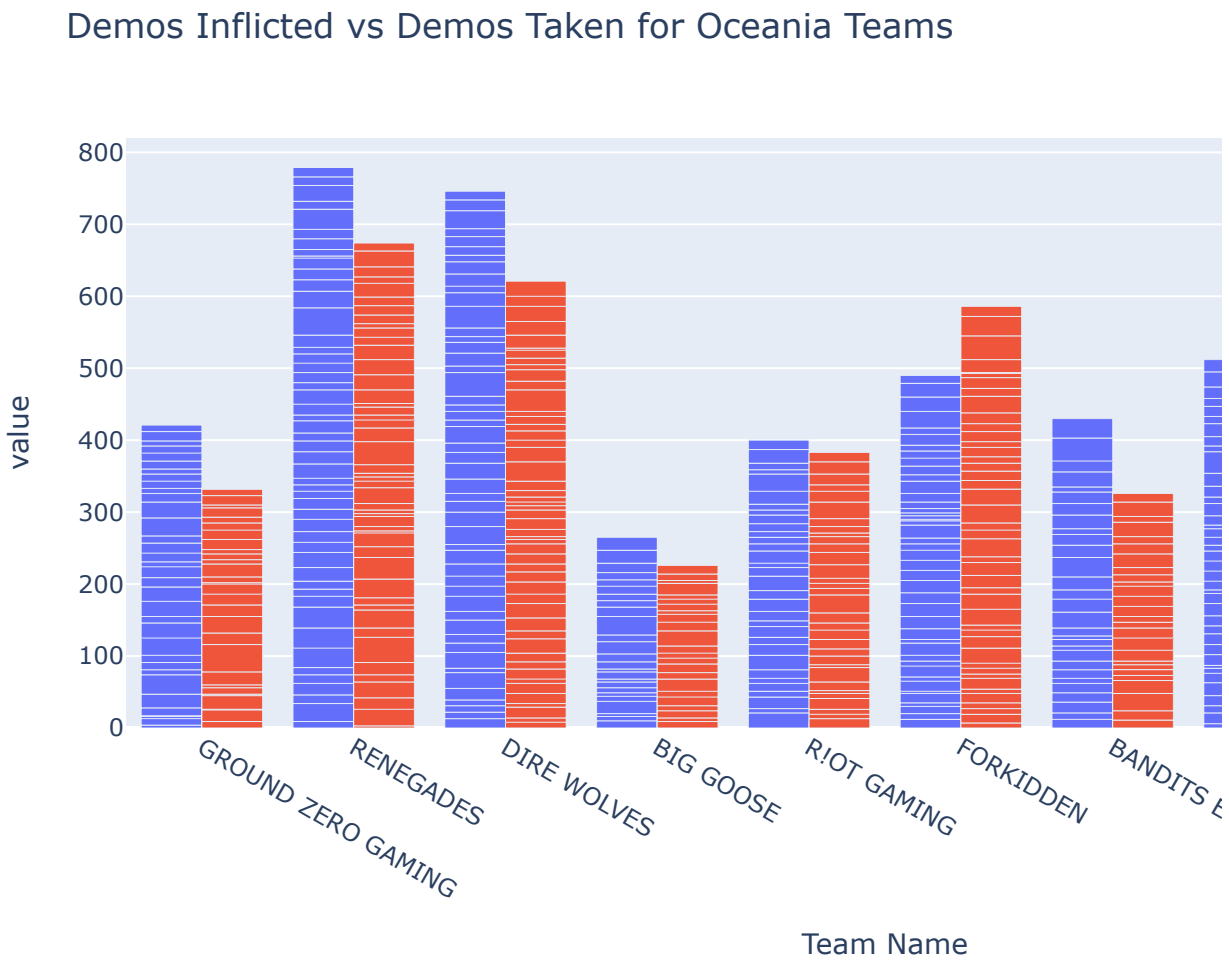


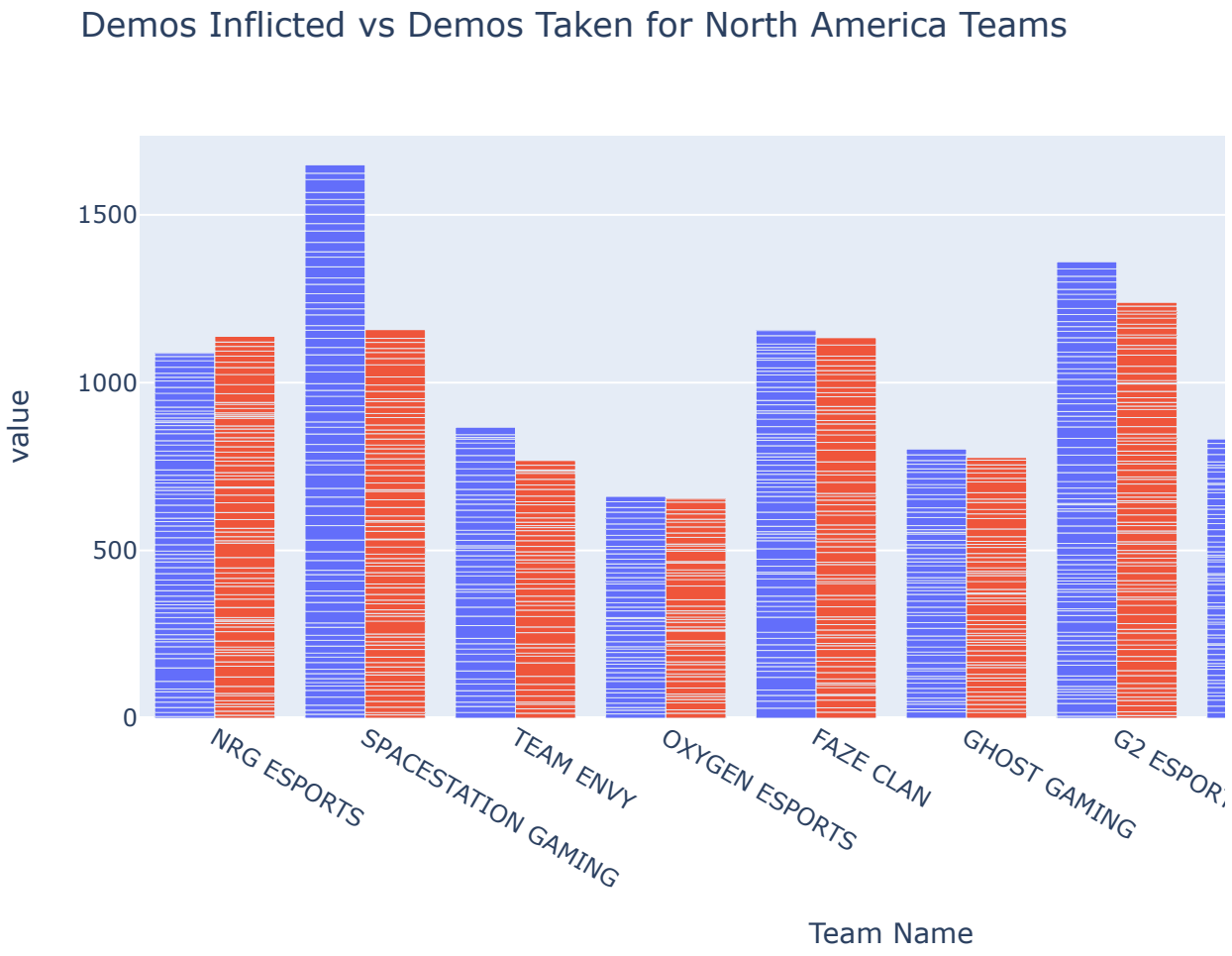


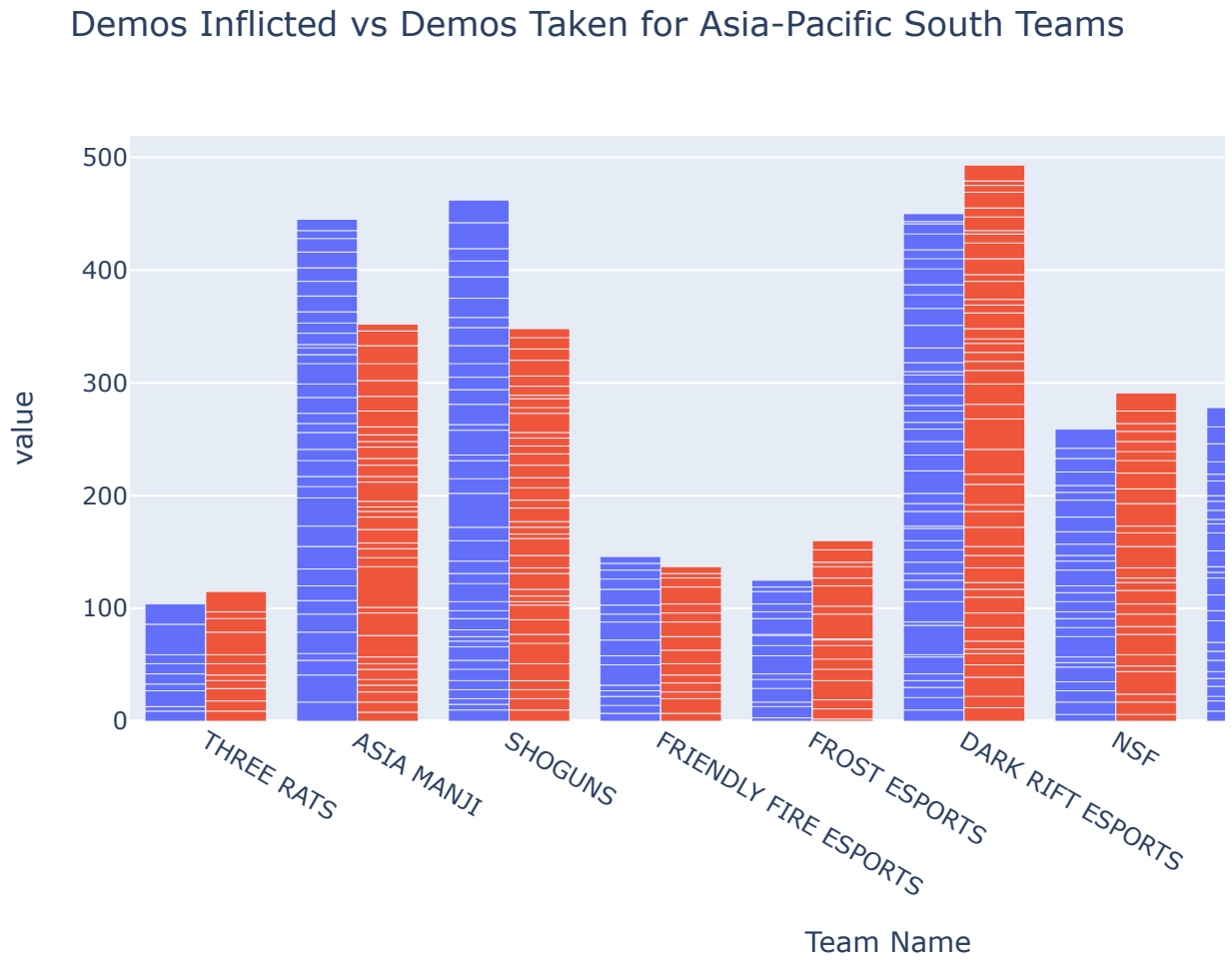

```
# group the data by region and sum up the scores
df_agg = df.groupby("team_region").agg({"score": "sum"}).reset_index()

# create the pie chart
fig = px.pie(df_agg, values='score', names='team_region',title=f'Score',width=960, height=
fig.show()
```

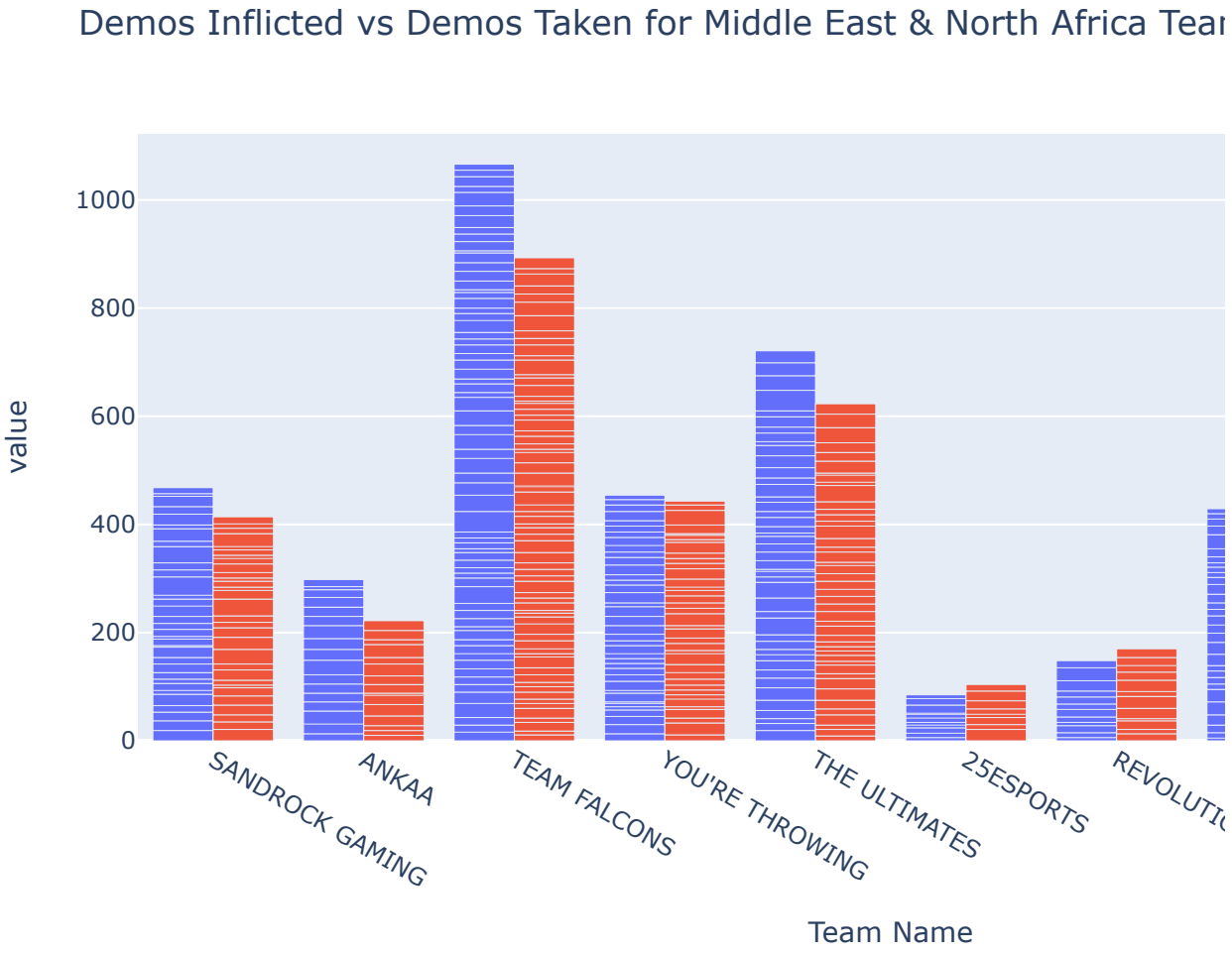



```
for region in regions:
  region_teams = top_teams[top_teams['team_region'] == region]
  fig = px.bar(region_teams, x='team_name', y=['demo_inflicted', 'demo_taken'],
  barmode='group',
  labels={'demo_inflicted':'Demos Inflicted',
  'demo_taken':'Demos Taken',
  'team_name': 'Team Name'},width=1000, height=520,
  title=f'Demos Inflicted vs Demos Taken for {region} Teams')
  fig.show()
```

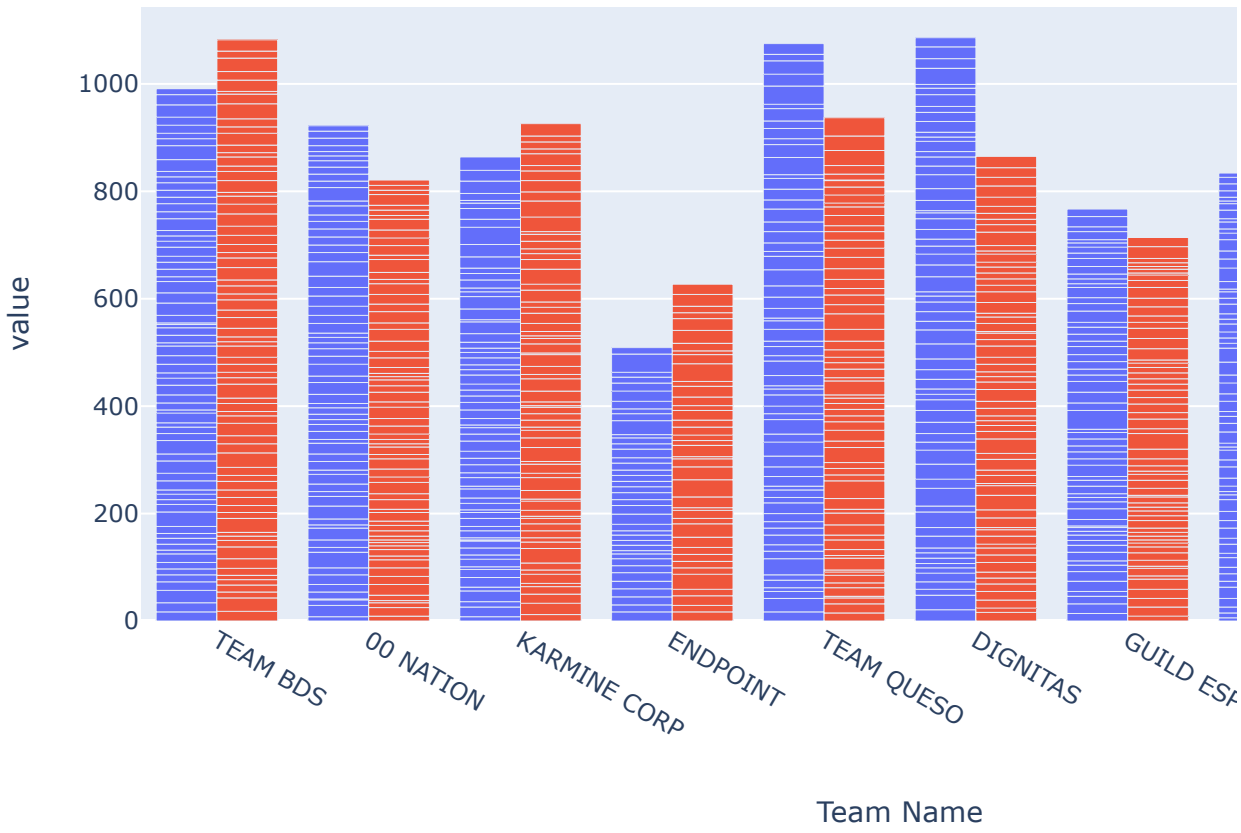

Demos Inflicted vs Demos Taken for Oceania Teams
800
700
600
500
400
300
200
100
0
value
GROUND ZERO GAMING
RENEGADES
DIRE WOLVES
BIG GOOSE
R!OT GAMING
FORKIDDEN
BANDITS E
Team Name


Demos Inflicted vs Demos Taken for North America Teams
1500
1000
500
0
value
NRG ESPORTS
SPACESTATION GAMING
TEAM ENVY
OXYGEN ESPORTS
FAZE CLAN
GHOST GAMING
G2 ESPOR
Team Name


Demos Inflicted vs Demos Taken for Asia-Pacific South Teams
value
0
100
200
300
400
500
THREE RATS
ASIA MANJI
SHOGUNS
FRIENDLY FIRE ESPORTS
FROST ESPORTS
DARK RIFT ESPORTS
NSF
Team Name

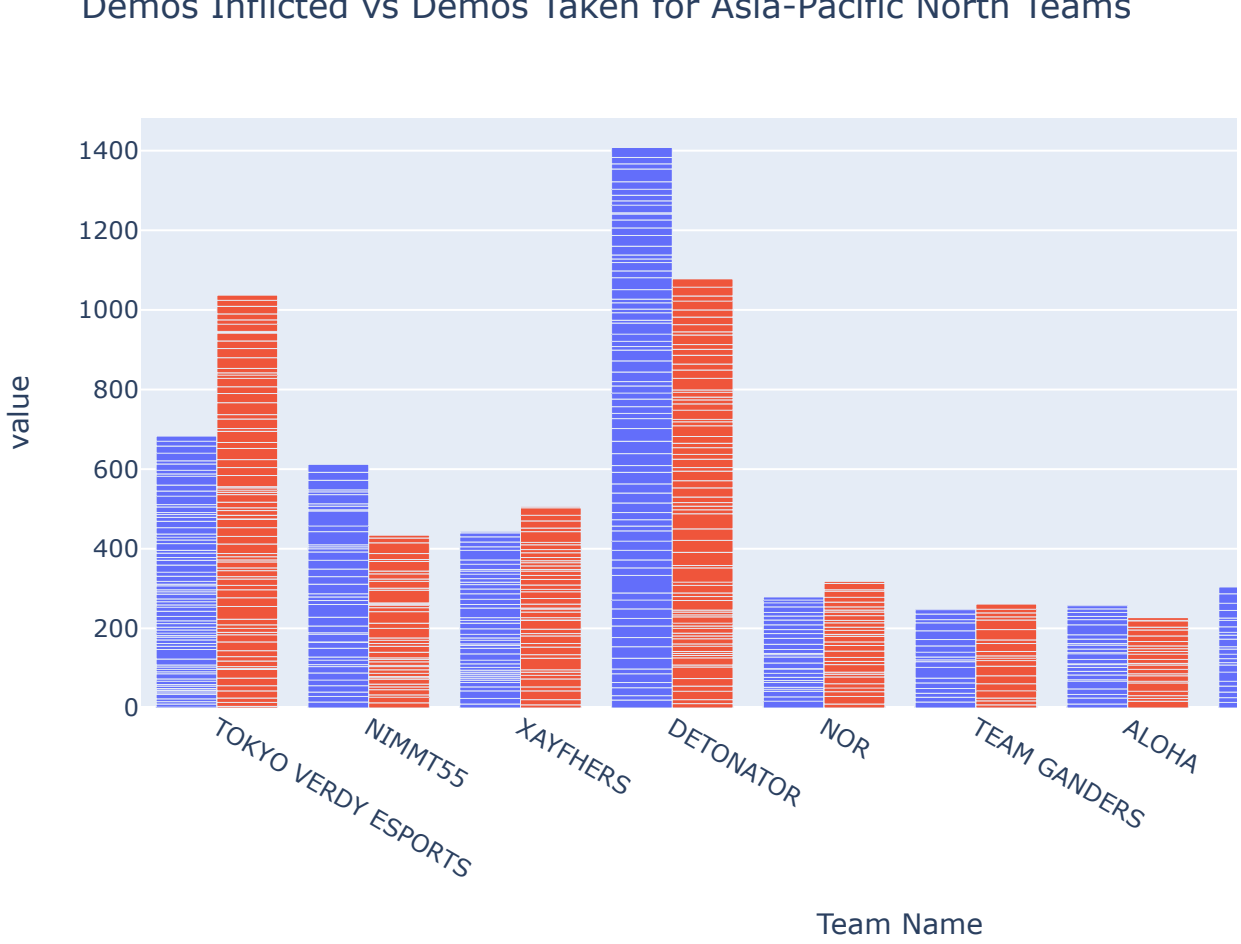

Demos Inflicted vs Demos Taken for Middle East & North Africa Tear
value
0
200
400
600
800
1000
SANDROCK GAMING
ANKAA
TEAM FALCONS
YOU'RE THROWING
THE ULTIMATES
25ESPORTS
REVOLUTIO
Team Name

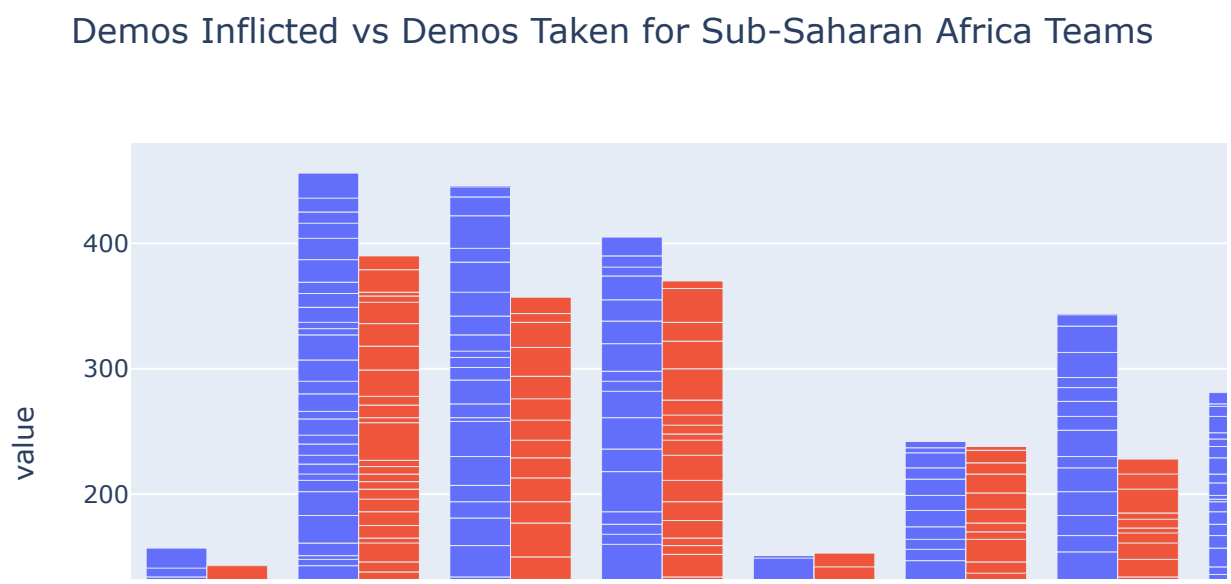
Demos Inflicted vs Demos Taken for Europe Teams



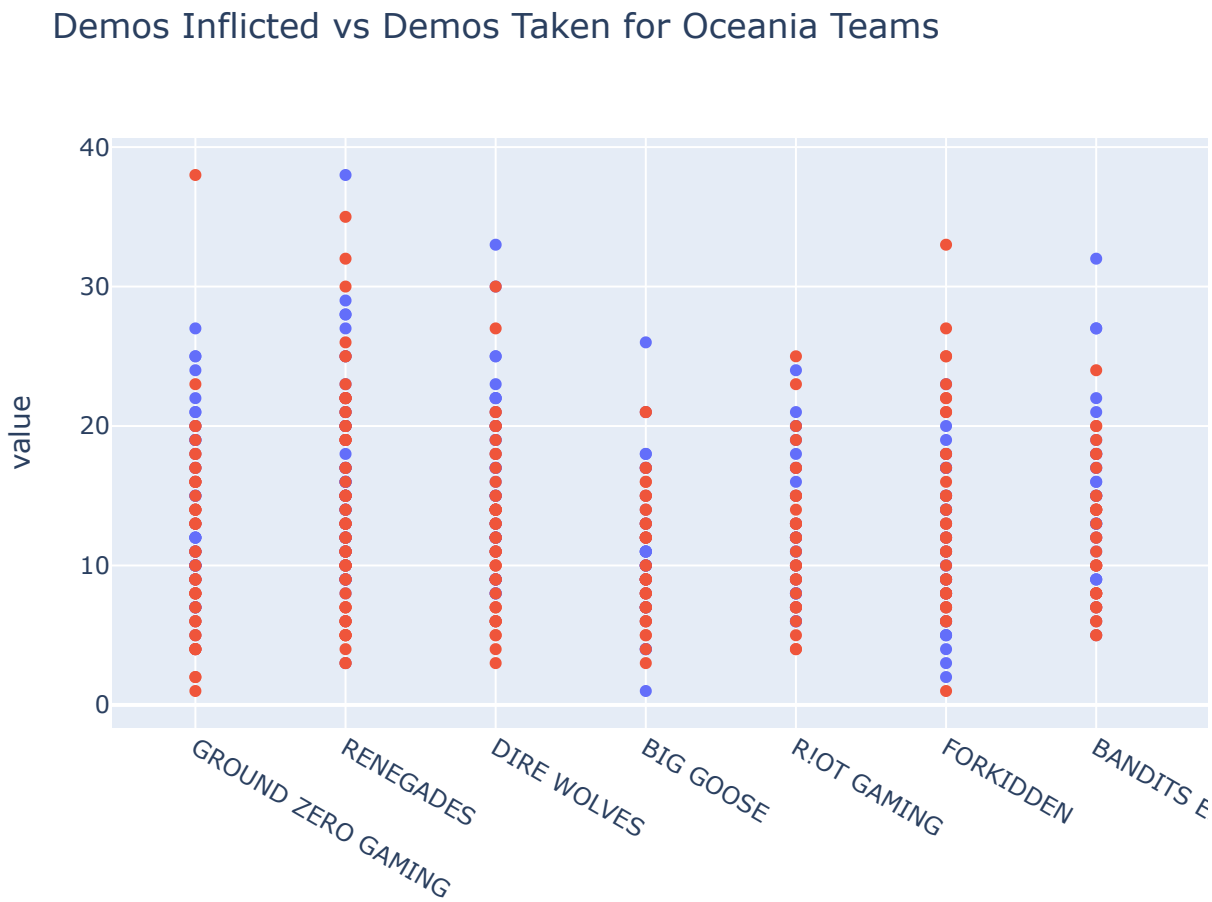
Demos Inflicted vs Demos Taken for South America Teams

Demos Inflicted vs Demos Taken for Asia-Pacific North Teams



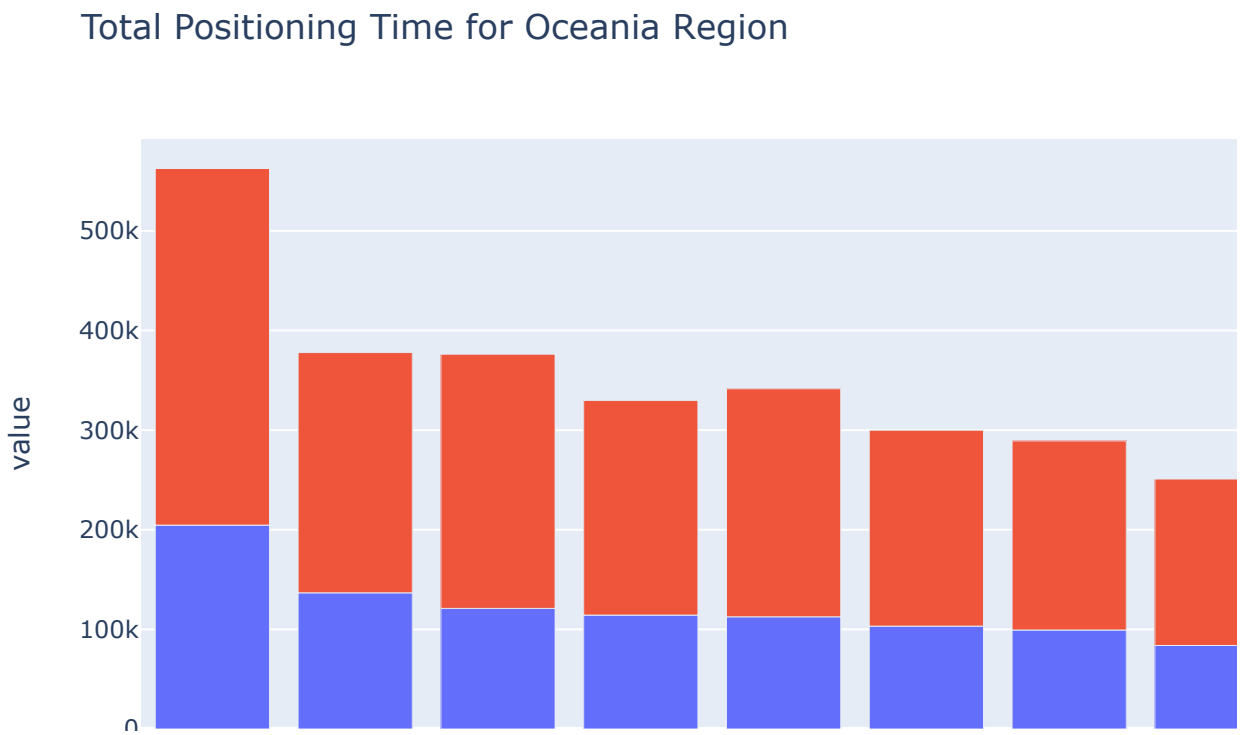



```
for region in regions:
  region_teams = top_teams[top_teams['team_region'] == region]
  fig = px.scatter(region_teams, x='team_name', y=['demo_inflicted', 'demo_taken'],
  labels={'demo_inflicted':'Demos Inflicted',
  'demo_taken':'Demos Taken',
  'team_name': 'Team Name'},width=1000, height=520,
  title=f'Demos Inflicted vs Demos Taken for {region} Teams')
  fig.show()
  break
```

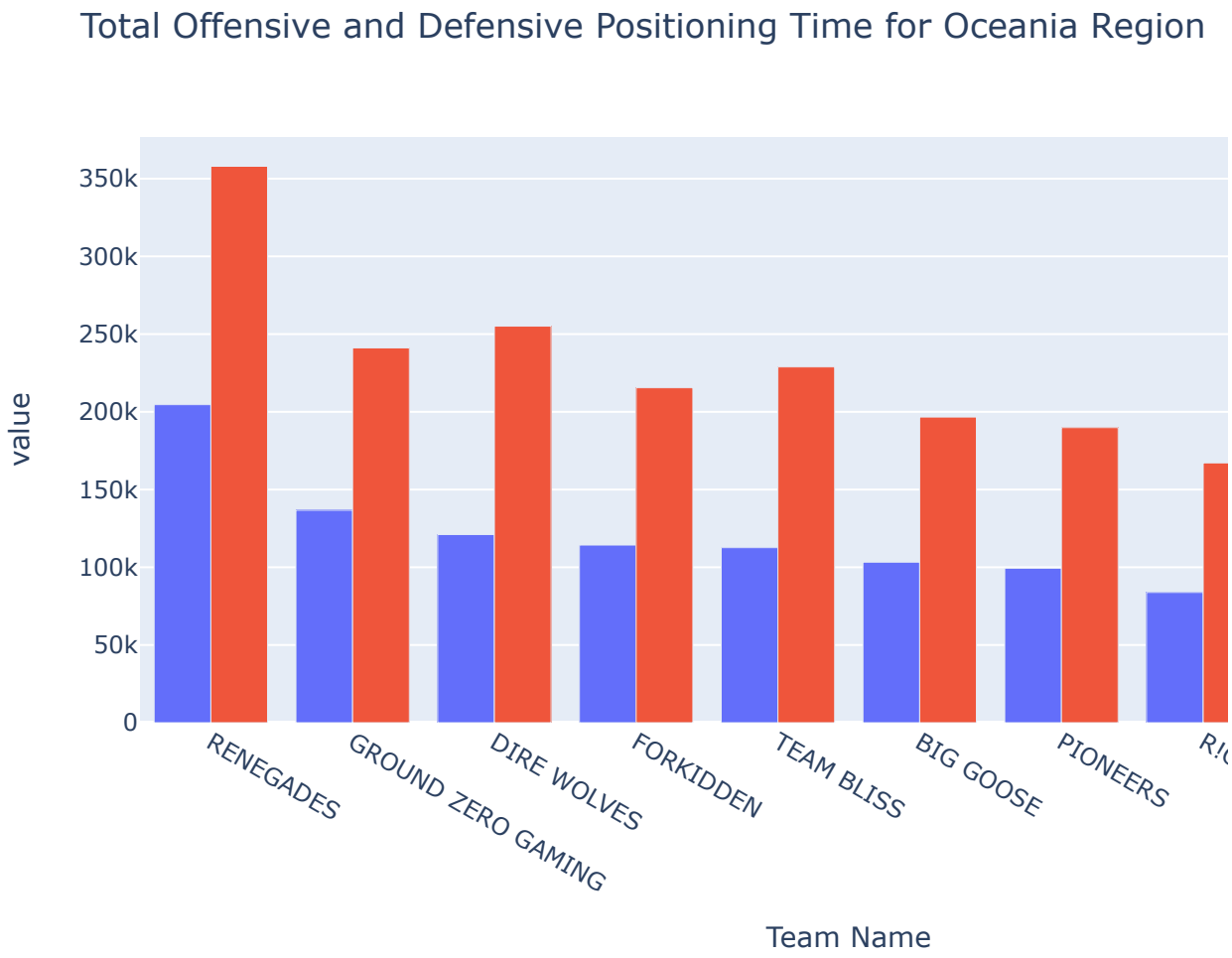


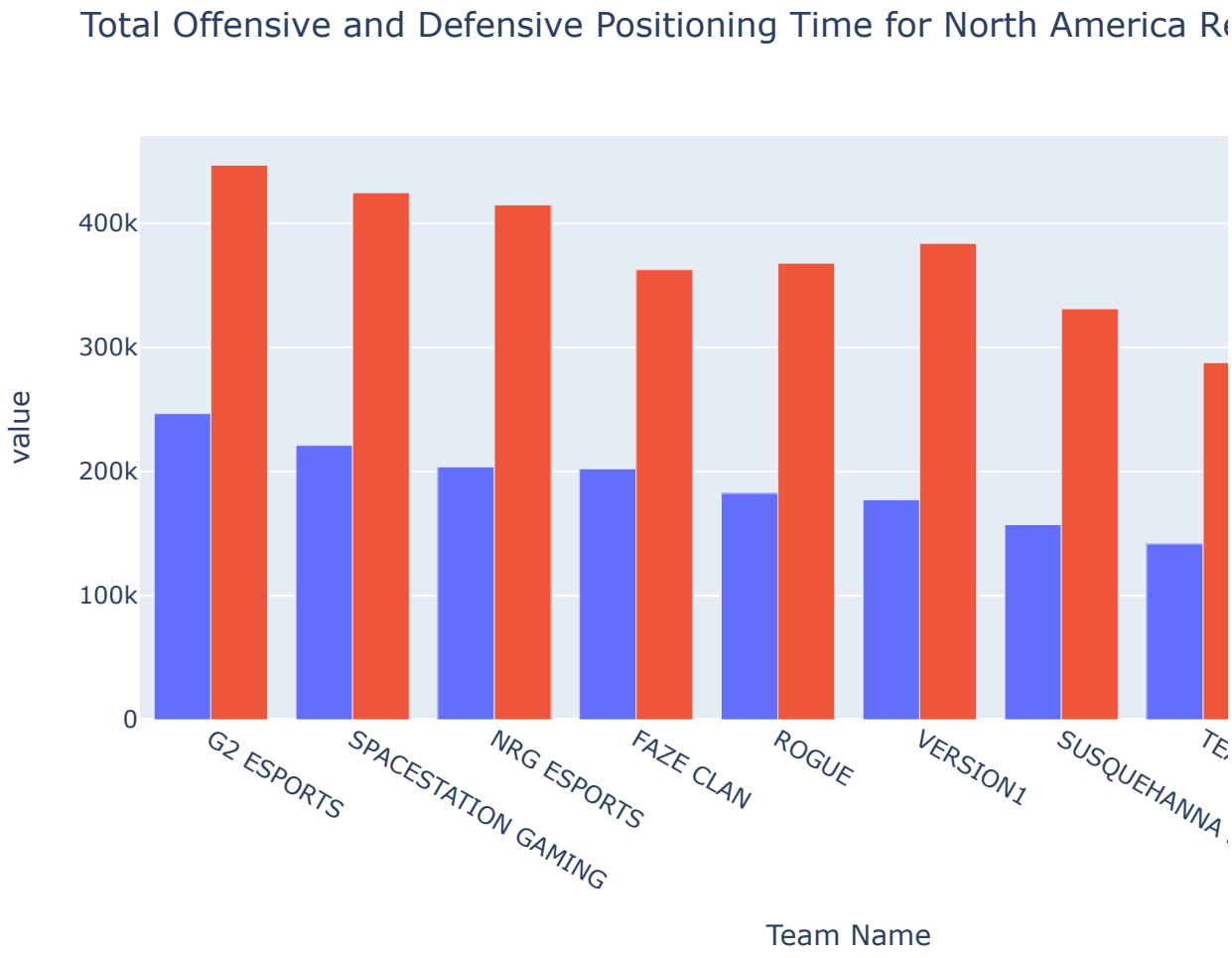

```
top_teams['total_defensive_time'] = top_teams['positioning_time_defensive_third'] + top_te


for region in regions:
    region_df = top_teams[top_teams['team_region'] == region]
    # region_df=region_df.sort_values(by='total_positioning_time', ascending=False)
    grouped = region_df.groupby('team_name').agg({'total_positioning_time': 'sum',
    'total_defensive_time': 'sum'})
    grouped = grouped.reset_index()
    grouped = grouped.sort_values(by='total_positioning_time', ascending=False)
    print(grouped.head())
    fig = px.bar(grouped, x='team_name', y=['total_positioning_time','total_defensive_time
                  hover_data=['team_name'],width=1000, height=520,
                 labels={'total_positioning_time':'Total Positioning Time',
                         'team_name': 'Team Name'},
                 title=f'Total Positioning Time for {region} Region')
    fig.show()
    break
```

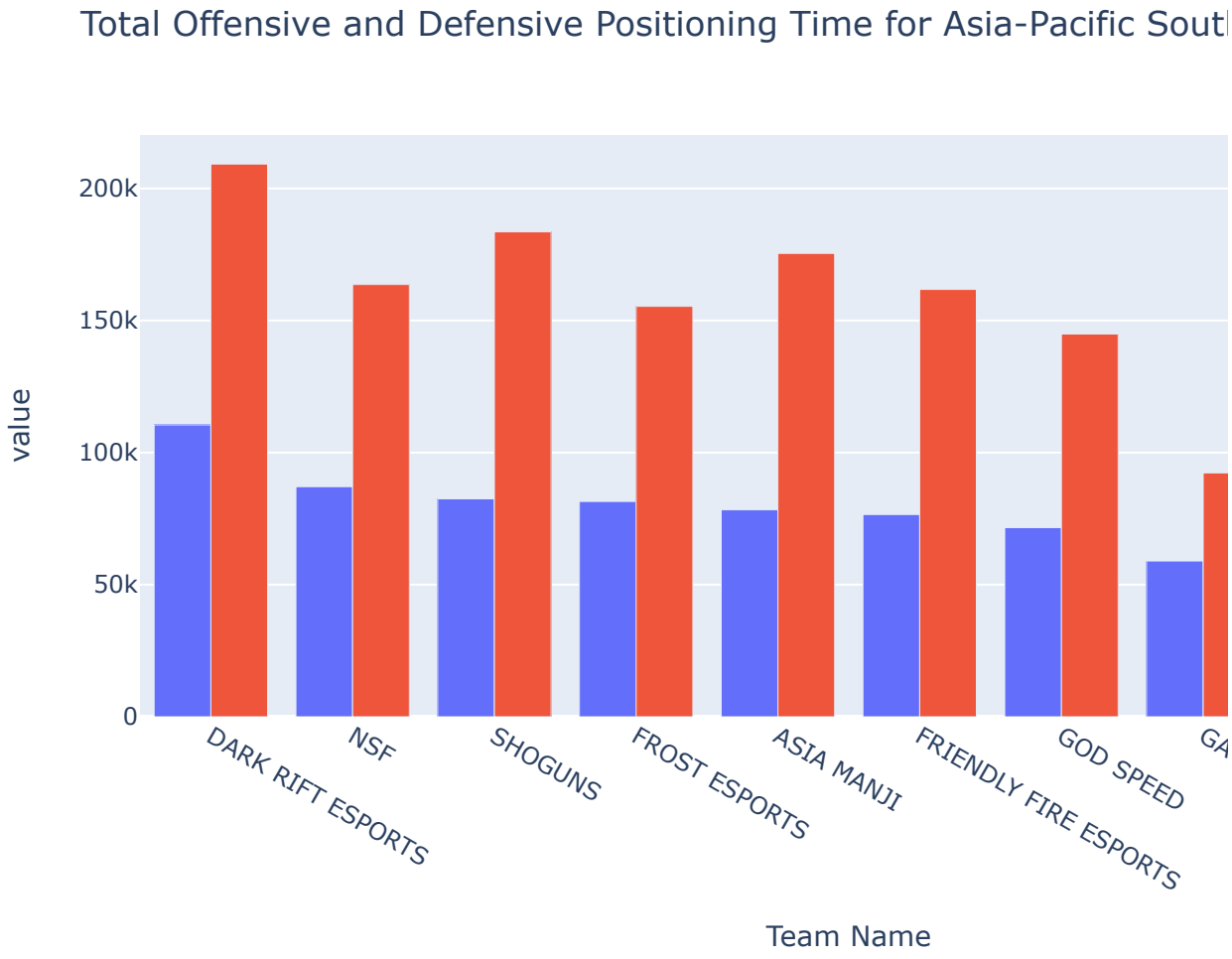
```
            team_name  total_positioning_time  total_defensive_time
8           RENEGADES               204617.79             357966.77
4  GROUND ZERO GAMING               136694.72             241125.60
2         DIRE WOLVES               121055.86             255116.78
3           FORKIDDEN               114312.75             215488.63
9          TEAM BLISS               112623.26             228883.91
```

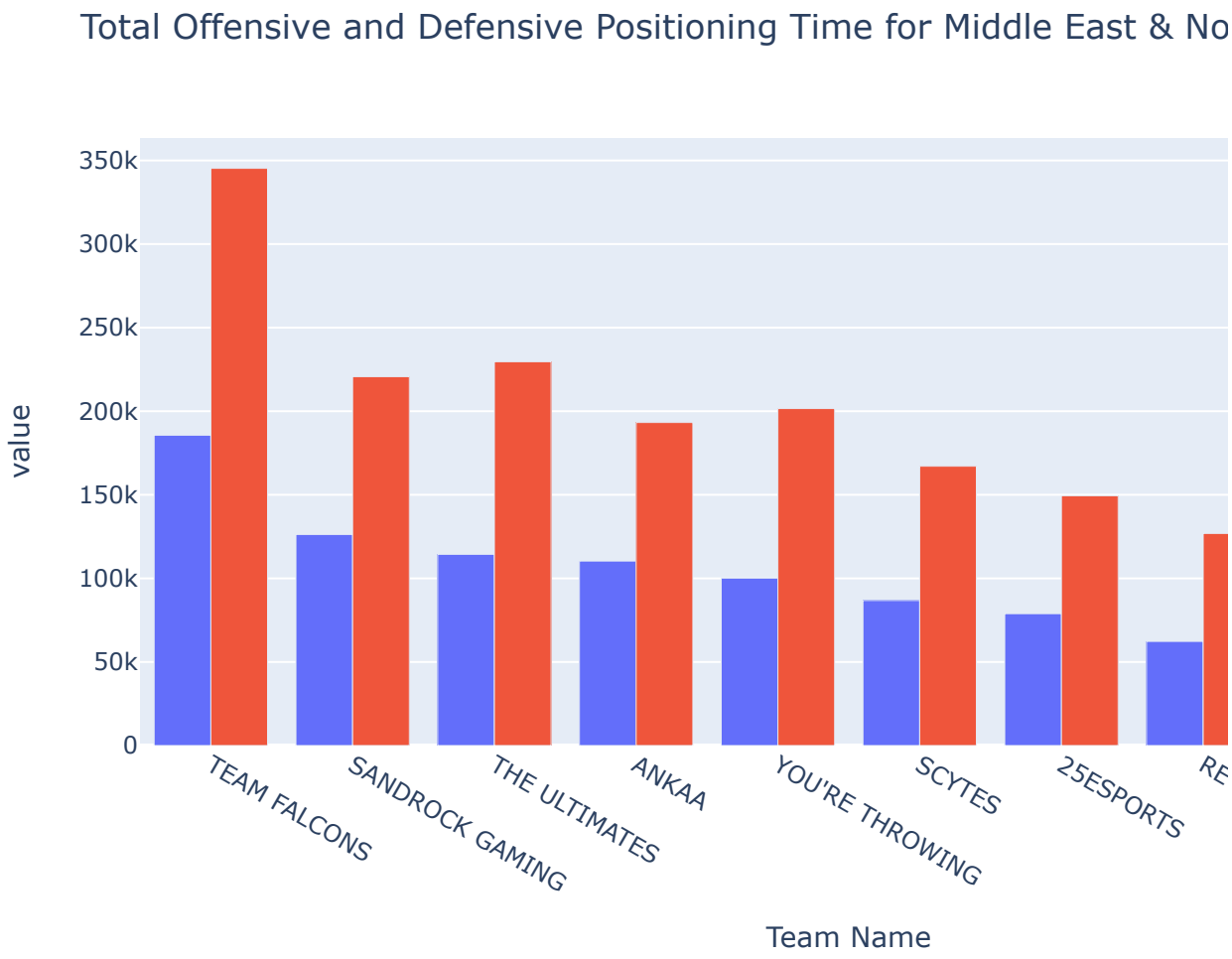


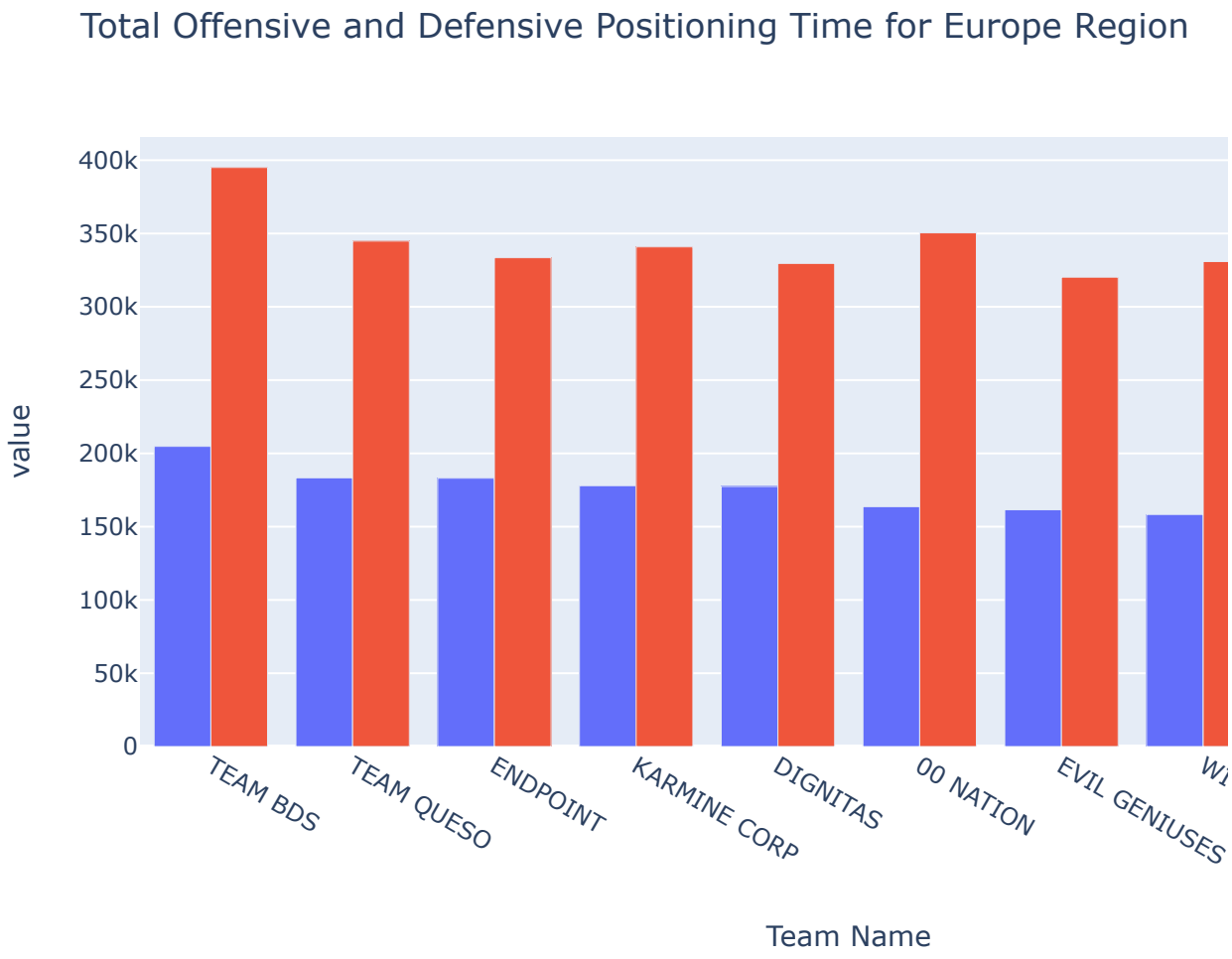

```
for region in regions:
    region_df = top_teams[top_teams['team_region'] == region]
    grouped = region_df.groupby('team_name').agg({'total_positioning_time': 'sum',
    'total_defensive_time': 'sum'})
    grouped = grouped.reset_index()
    grouped = grouped.sort_values(by='total_positioning_time', ascending=False)
    fig = px.bar(grouped, x='team_name', y=['total_positioning_time','total_defensive_time
                barmode='group',
                labels={'total_positioning_time':'Total Offensive Time',
                        'total_defensive_time': 'Total Offensive Time',
                        'team_name': 'Team Name'},width=1000, height=520,
                title=f'Total Offensive and Defensive Positioning Time for {region} Regio
    fig.show()
```

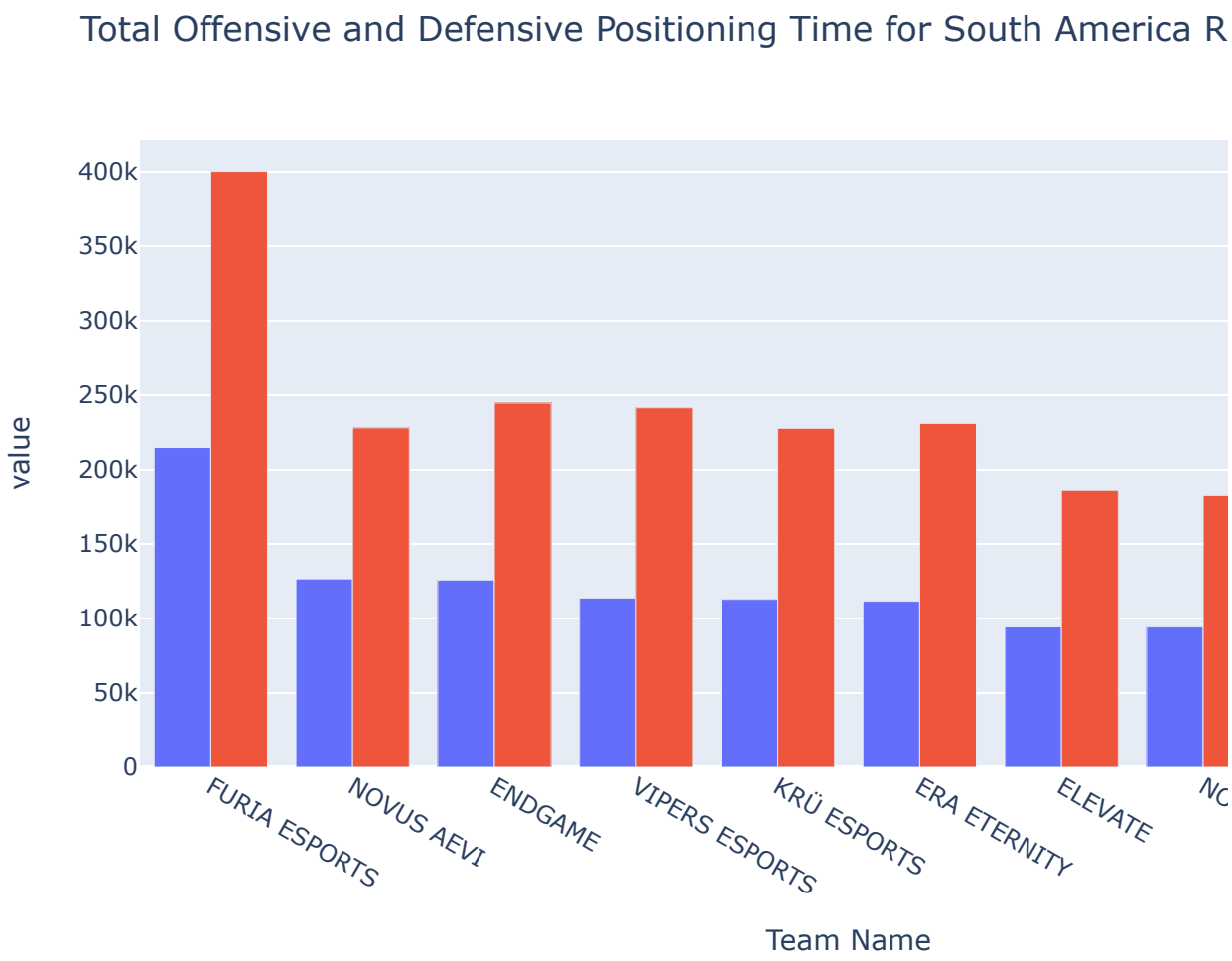

Total Offensive and Defensive Positioning Time for Oceania Region
value
350k
300k
250k
200k
150k
100k
50k
0
RENEGADES
GROUND ZERO GAMING
DIRE WOLVES
FORKIDDEN
TEAM BLISS
BIG GOOSE
PIONEERS
Team Name

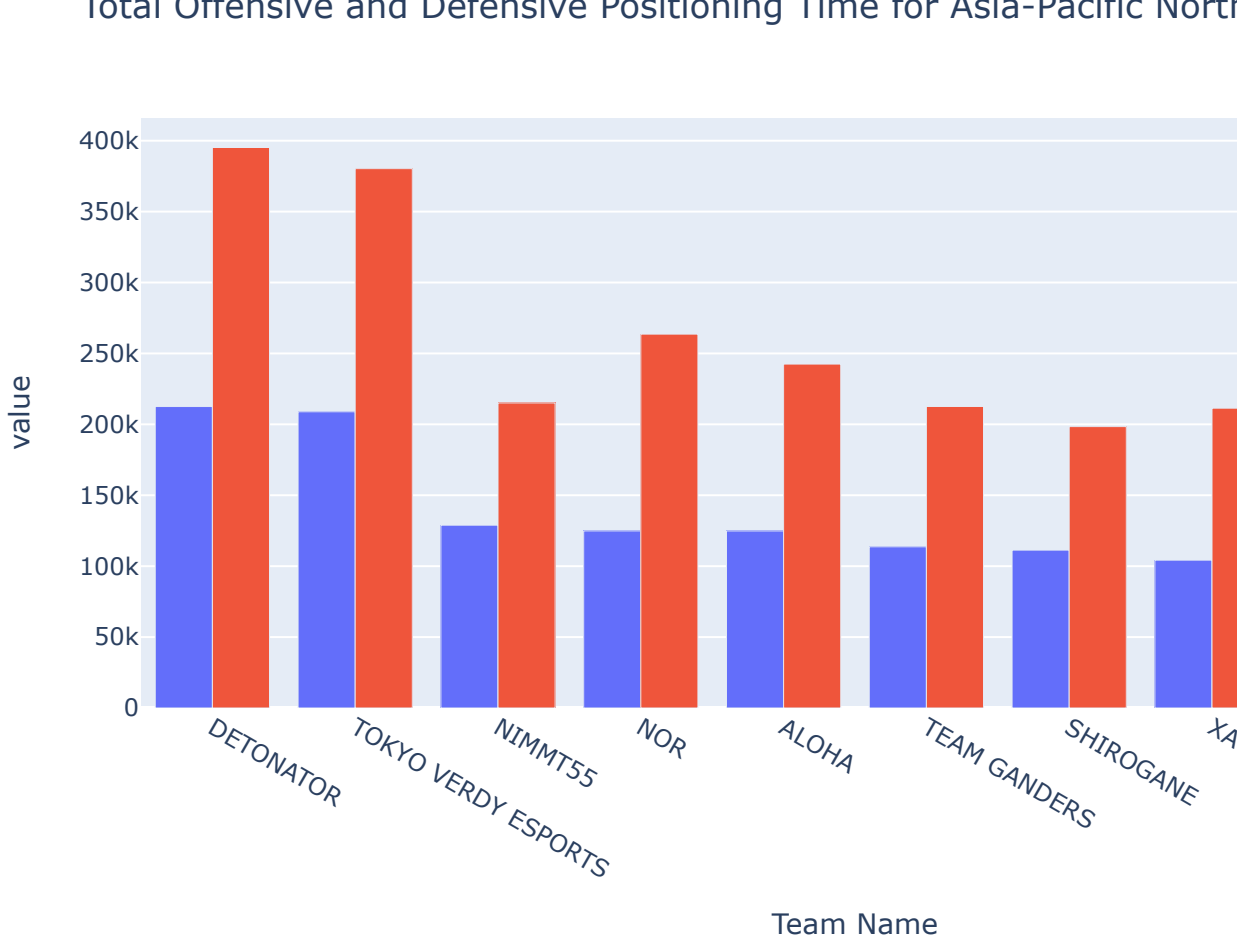

Total Offensive and Defensive Positioning Time for North America R
value
400k
300k
200k
100k
0
G2 ESPORTS
SPACESTATION GAMING
NRG ESPORTS
FAZE CLAN
ROGUE
VERSION1
SUSQUEHANNA
Team Name

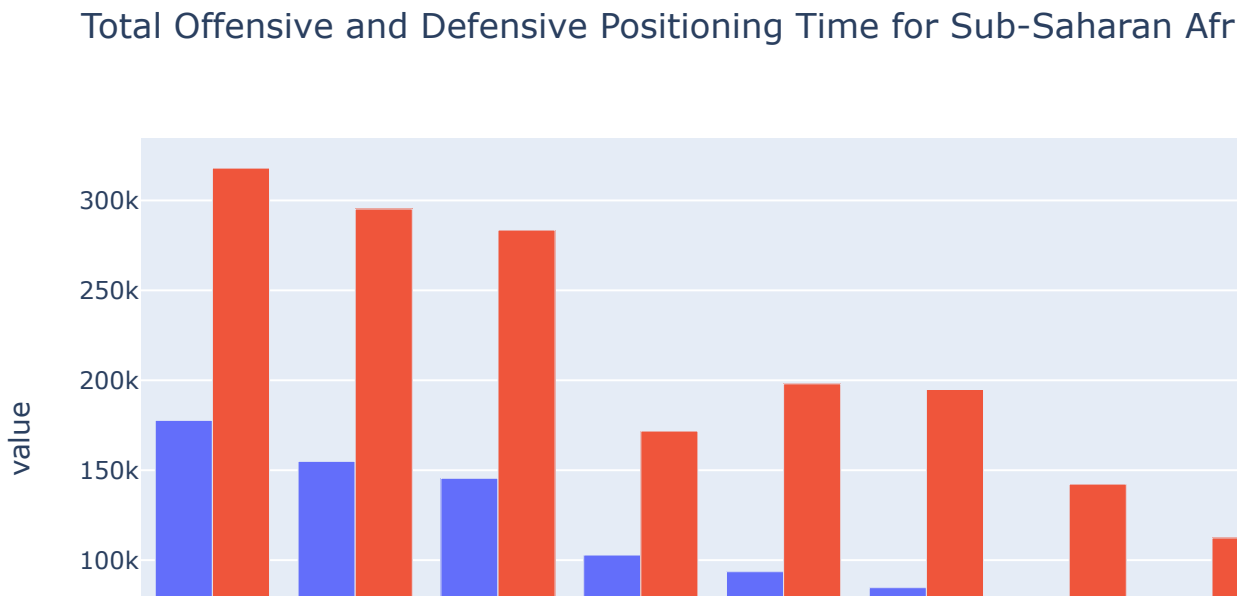

Total Offensive and Defensive Positioning Time for Asia-Pacific Soutl
value
200k
150k
100k
50k
0
DARK RIFT ESPORTS
NSF
SHOGUNS
FROST ESPORTS
ASIA MANJI
FRIENDLY FIRE ESPORTS
GOD SPEED
GA
Team Name


Total Offensive and Defensive Positioning Time for Middle East & No
value
350k
300k
250k
200k
150k
100k
50k
0
TEAM FALCONS
SANDROCK GAMING
THE ULTIMATES
ANKAA
YOU'RE THROWING
SCYTES
25ESPORTS
RE
Team Name

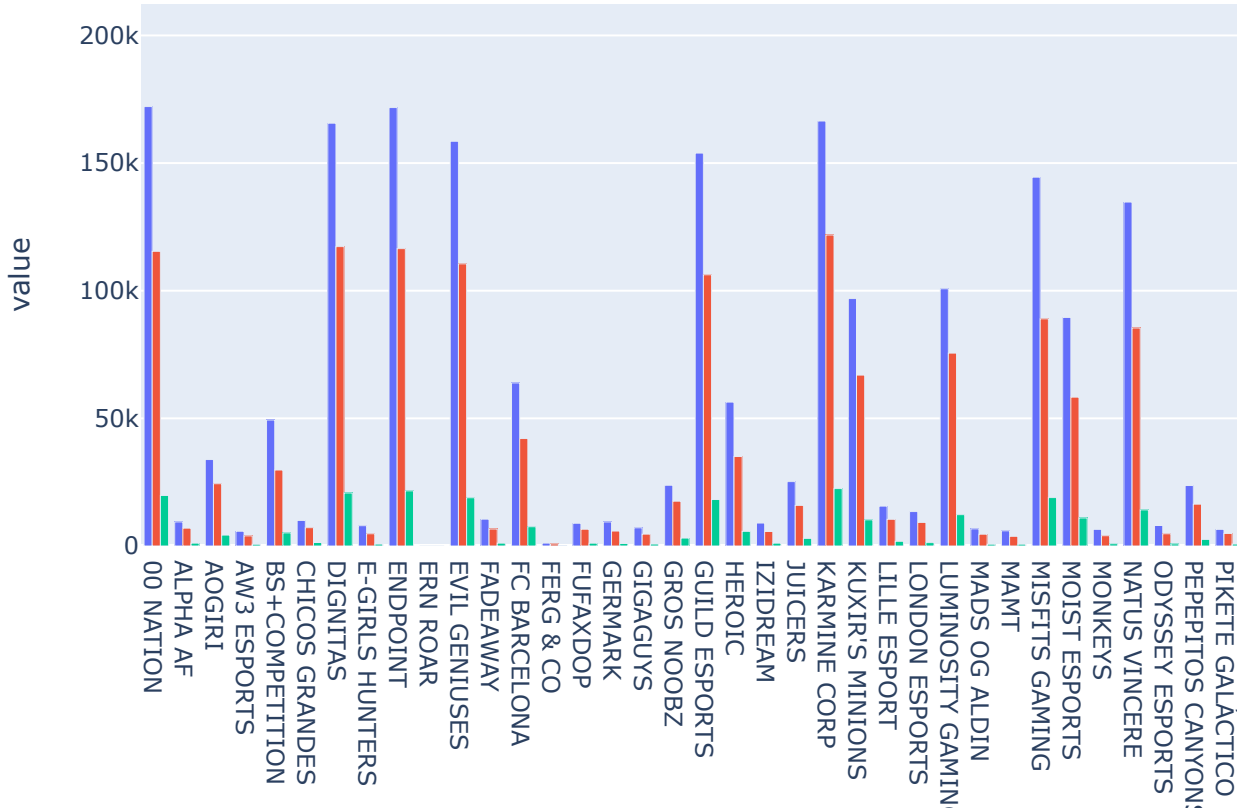

Total Offensive and Defensive Positioning Time for Europe Region
value
400k
350k
300k
250k
200k
150k
100k
50k
0
TEAM BDS
TEAM QUESO
ENDPOINT
KARMINE CORP
DIGNITAS
00 NATION
EVIL GENIUSES
Team Name


Total Offensive and Defensive Positioning Time for South America R
value
400k
350k
300k
250k
200k
150k
100k
50k
0
FURIA ESPORTS
NOVUS AEVI
ENDGAME
VIPERS ESPORTS
KRÜ ESPORTS
ERA ETERNITY
ELEVATE
Team Name

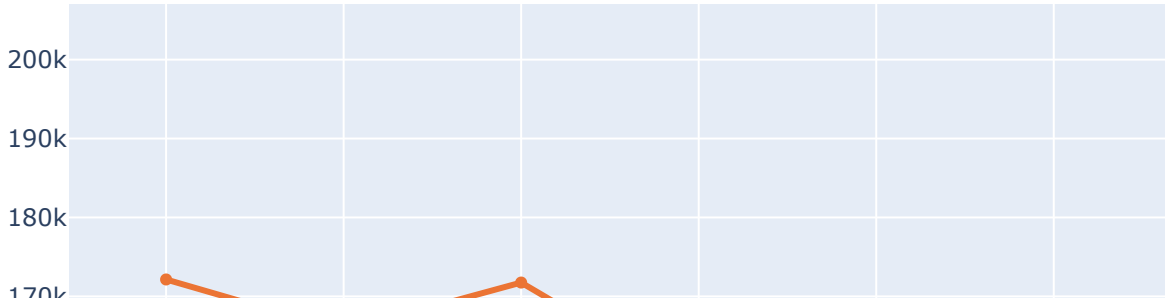




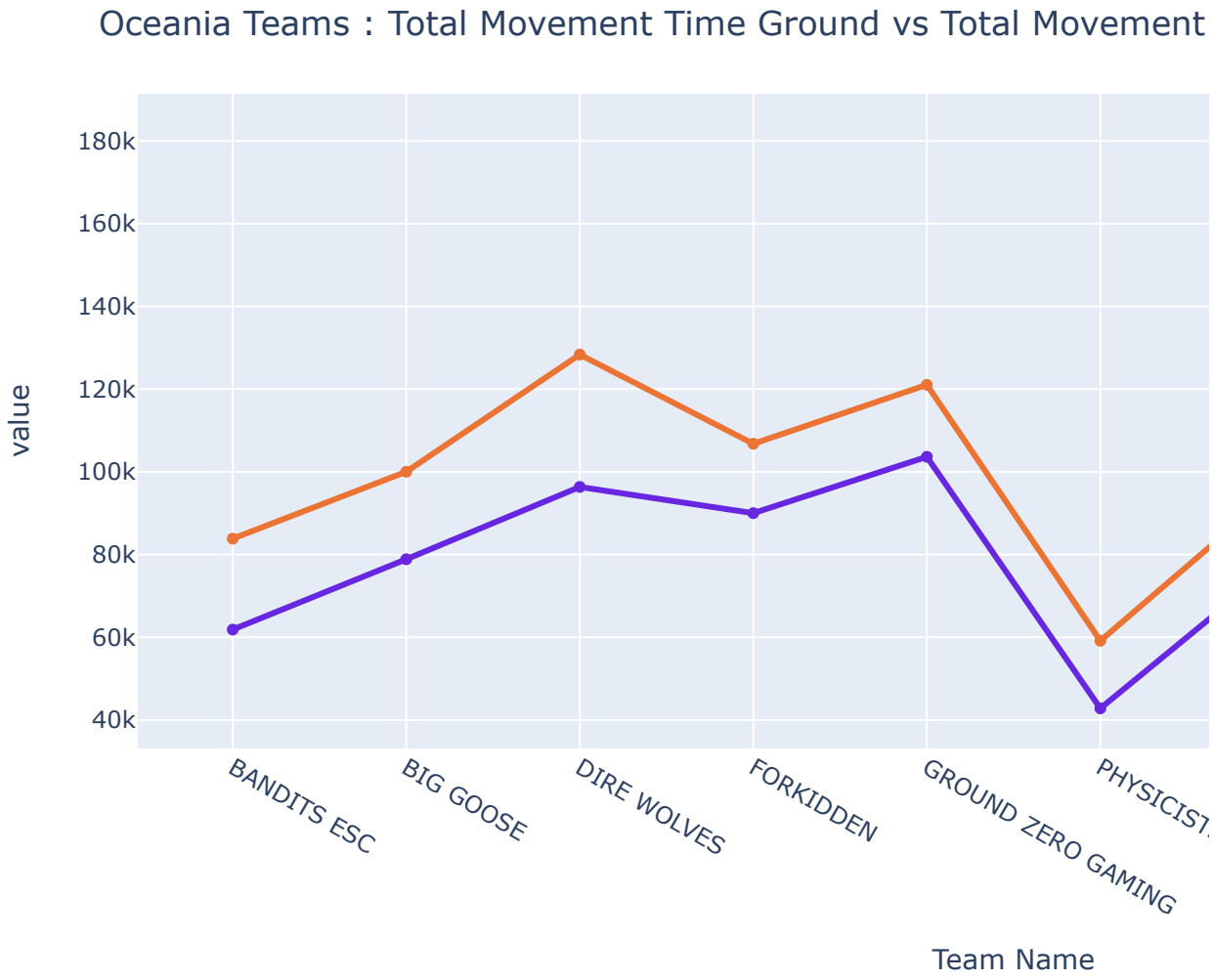

```
df_europe = df[df["team_region"] == "Europe"]
df_europe_agg = df_europe.groupby("team_name").agg({"movement_time_ground": "sum", "moveme

fig = px.bar(df_europe_agg, x="team_name", y=["movement_time_ground", "movement_time_low_a
fig.show()
```

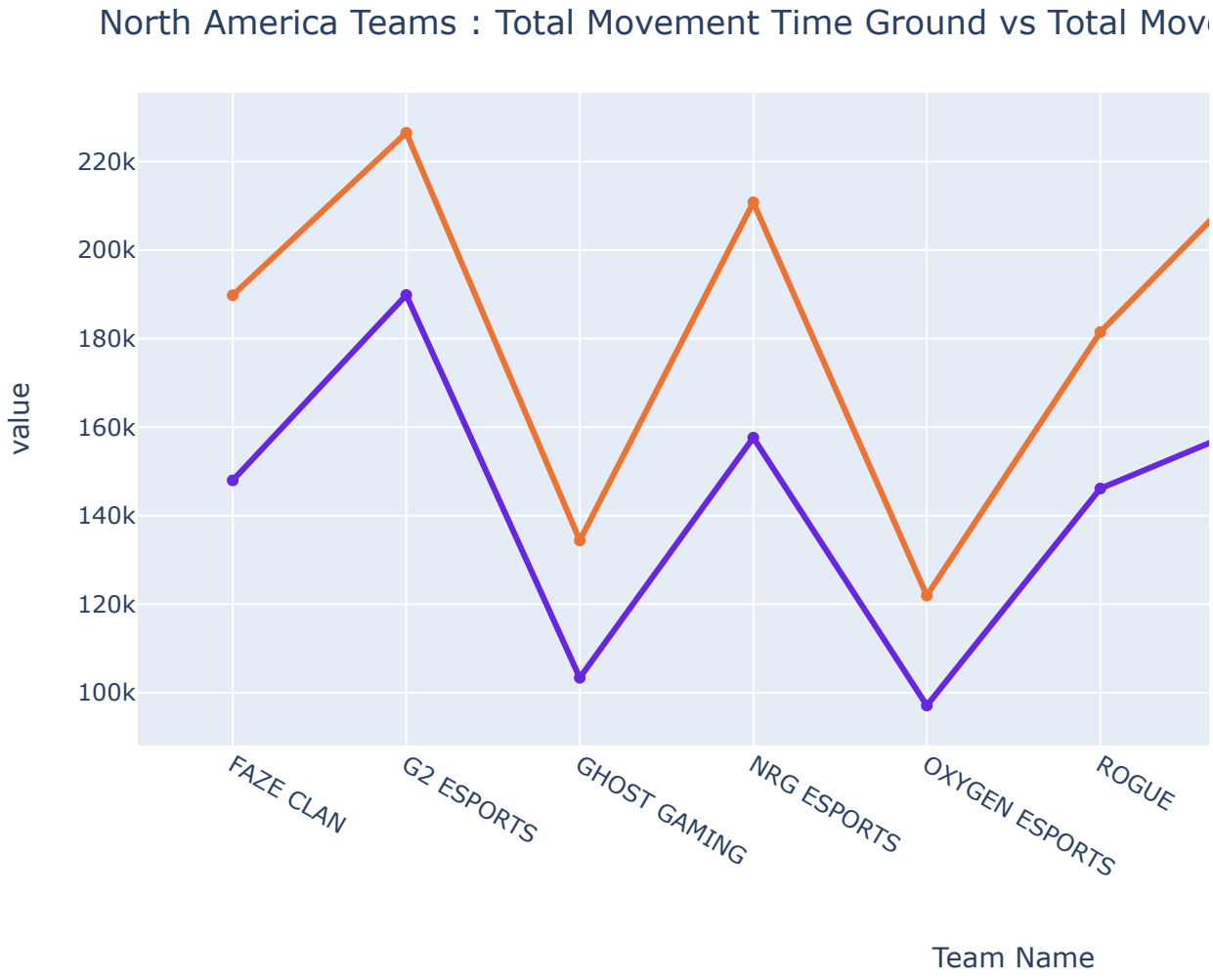


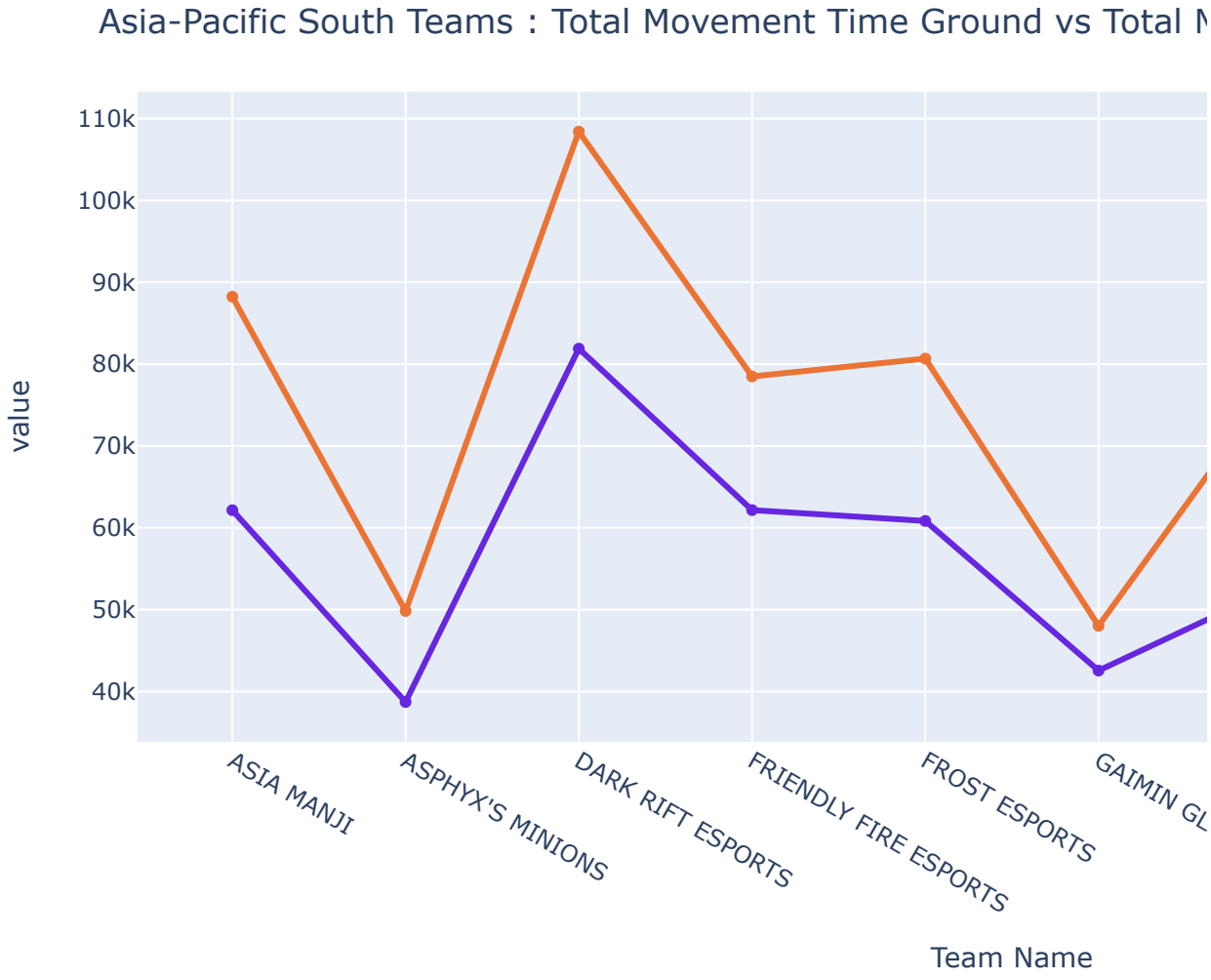

```
df_europe = top_teams[top_teams["team_region"] == "Europe"]
df_europe['movement_time'] = df_europe['movement_time_low_air'] + df_europe['movement_time

df_europe_agg = df_europe.groupby("team_name").agg({"movement_time_ground": "sum", "moveme

fig = px.line(df_europe_agg, x="team_name", y=["movement_time_ground", "movement_time"],ma
fig['data'][0]['line']['color']='rgb(235, 116, 52)'
fig['data'][0]['line']['width']=3
fig['data'][1]['line']['color']='rgb(103, 38, 224)'
fig['data'][1]['line']['width']=3
fig.show()
```

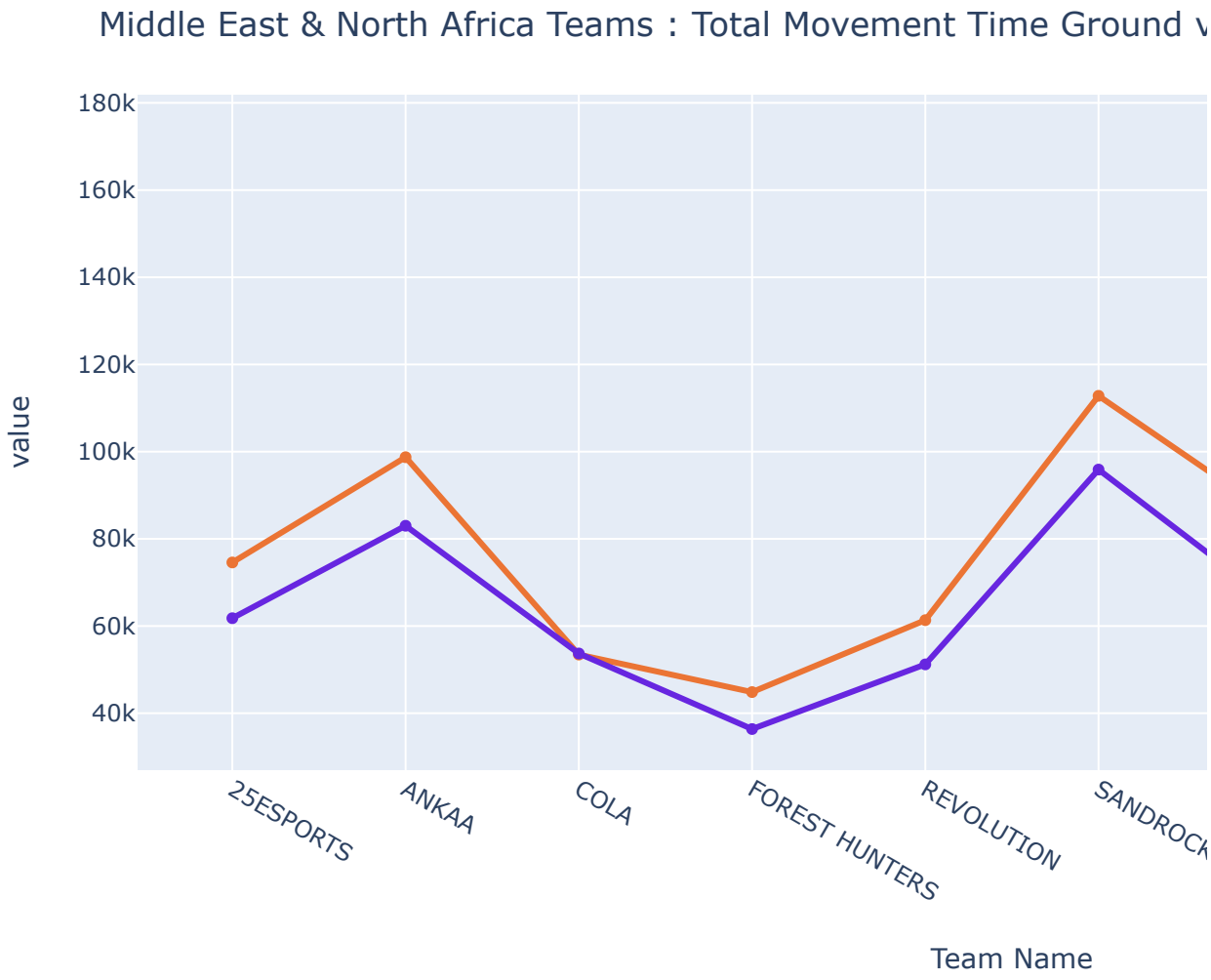



```
for region in regions:
    df_region = top_teams[top_teams["team_region"] == region]
    df_region['movement_time_air'] = df_region['movement_time_low_air'] + df_region['movem

    df_region_agg = df_region.groupby("team_name").agg({"movement_time_ground": "sum", "mo

    fig = px.line(df_region_agg, x="team_name", y=["movement_time_ground", "movement_time_
                  labels={'movement_time_ground':'Total Ground Time',
                          'movement_time_air': 'Total Air Time',
                          'team_name': 'Team Name'},width=1200, height=520)
    fig['data'][0]['line']['color'] = 'rgb(235, 116, 52)'
    fig['data'][0]['line']['width'] = 3
    fig['data'][1]['line']['color'] = 'rgb(103, 38, 224)'
    fig['data'][1]['line']['width'] = 3
    fig.update_layout(title=f"{region} Teams : Total Movement Time Ground vs Total Movemen
    fig.show()
```


Oceania Teams : Total Movement Time Ground vs Total Movement
value
180k
160k
140k
120k
100k
80k
60k
40k
BANDITS ESC
BIG GOOSE
DIRE WOLVES
FORKIDDEN
GROUND ZERO GAMING
PHYSICIST
Team Name

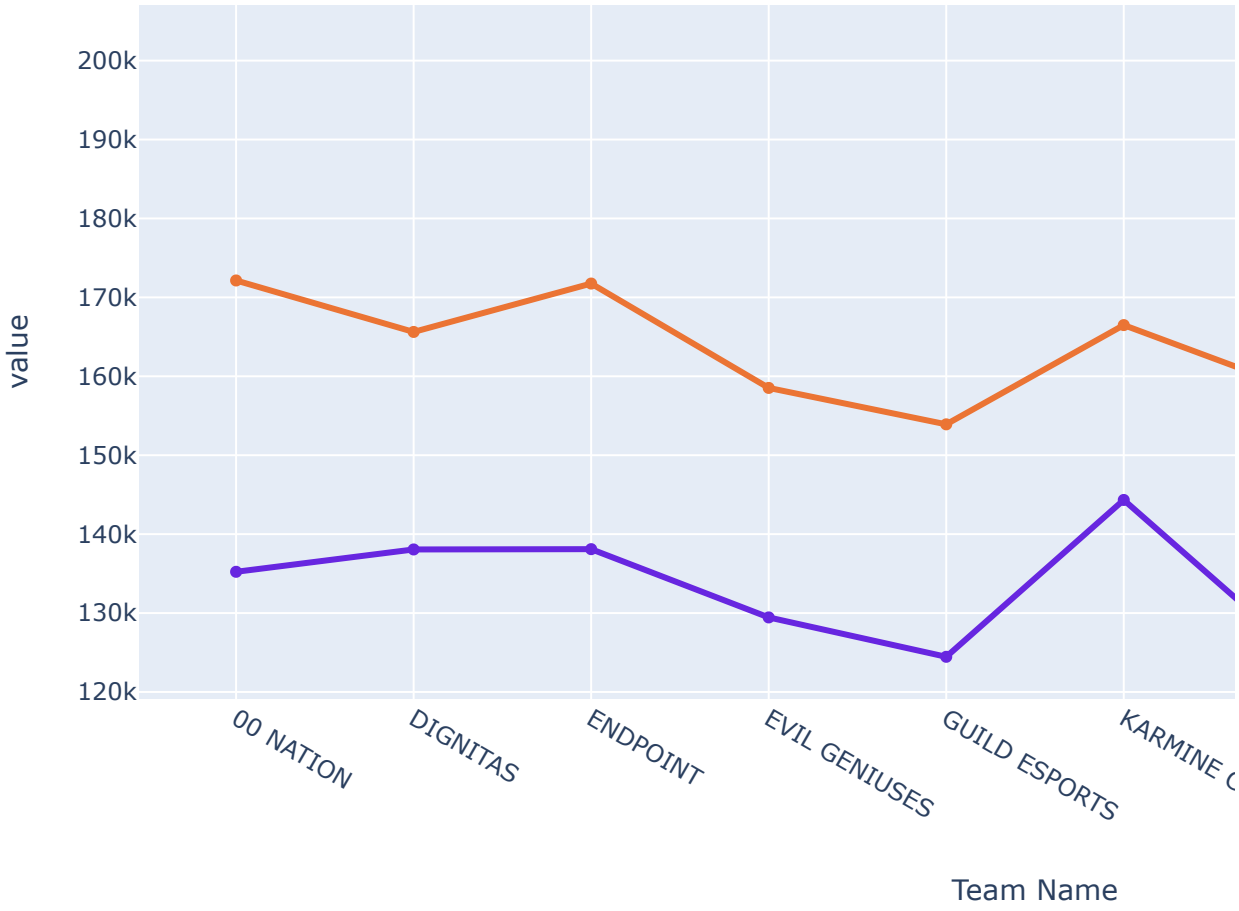

North America Teams : Total Movement Time Ground vs Total Mov
value
220k
200k
180k
160k
140k
120k
100k
FAZE CLAN
G2 ESPORTS
GHOST GAMING
NRG ESPORTS
OXYGEN ESPORTS
ROGUE
Team Name



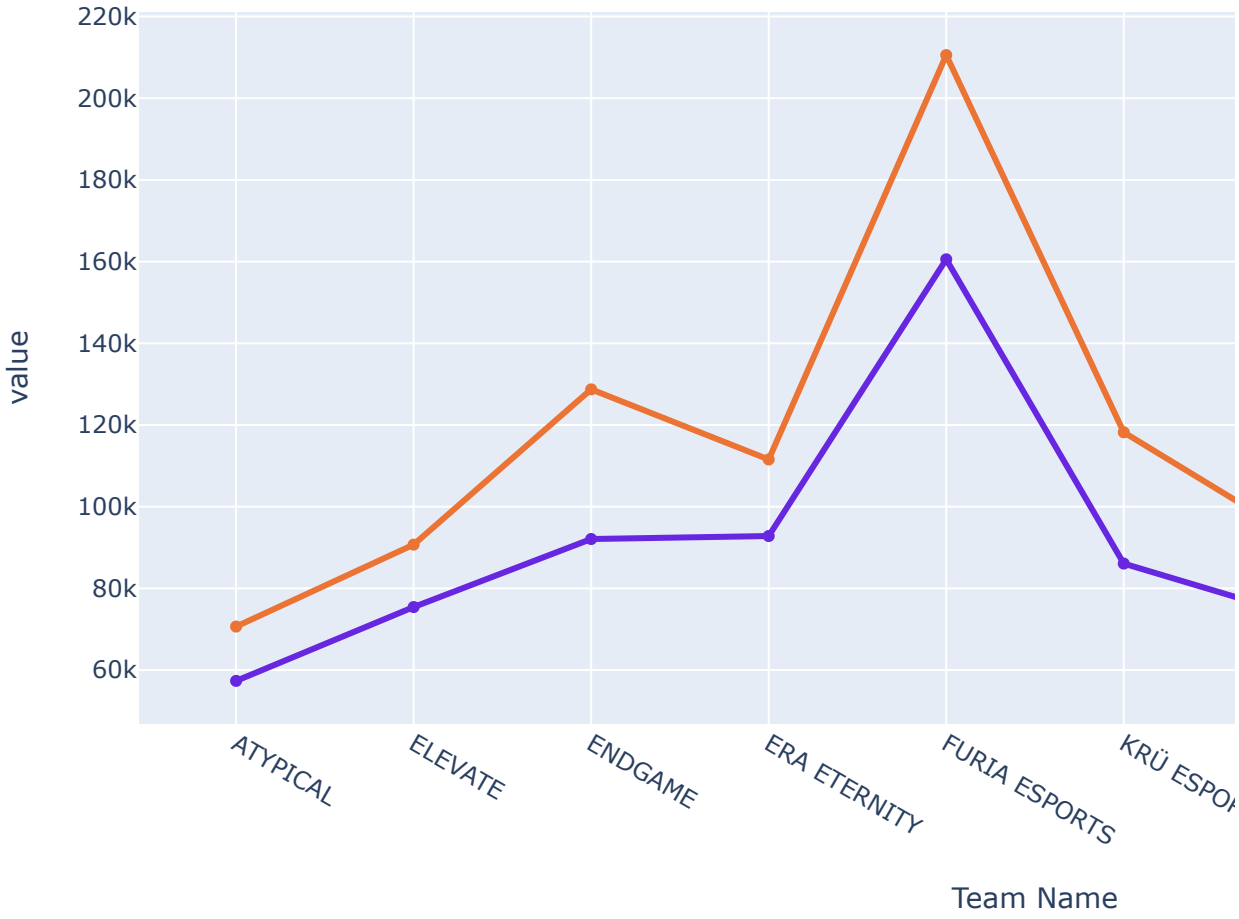


Europe Teams : Total Movement Time Ground vs Total Movement T

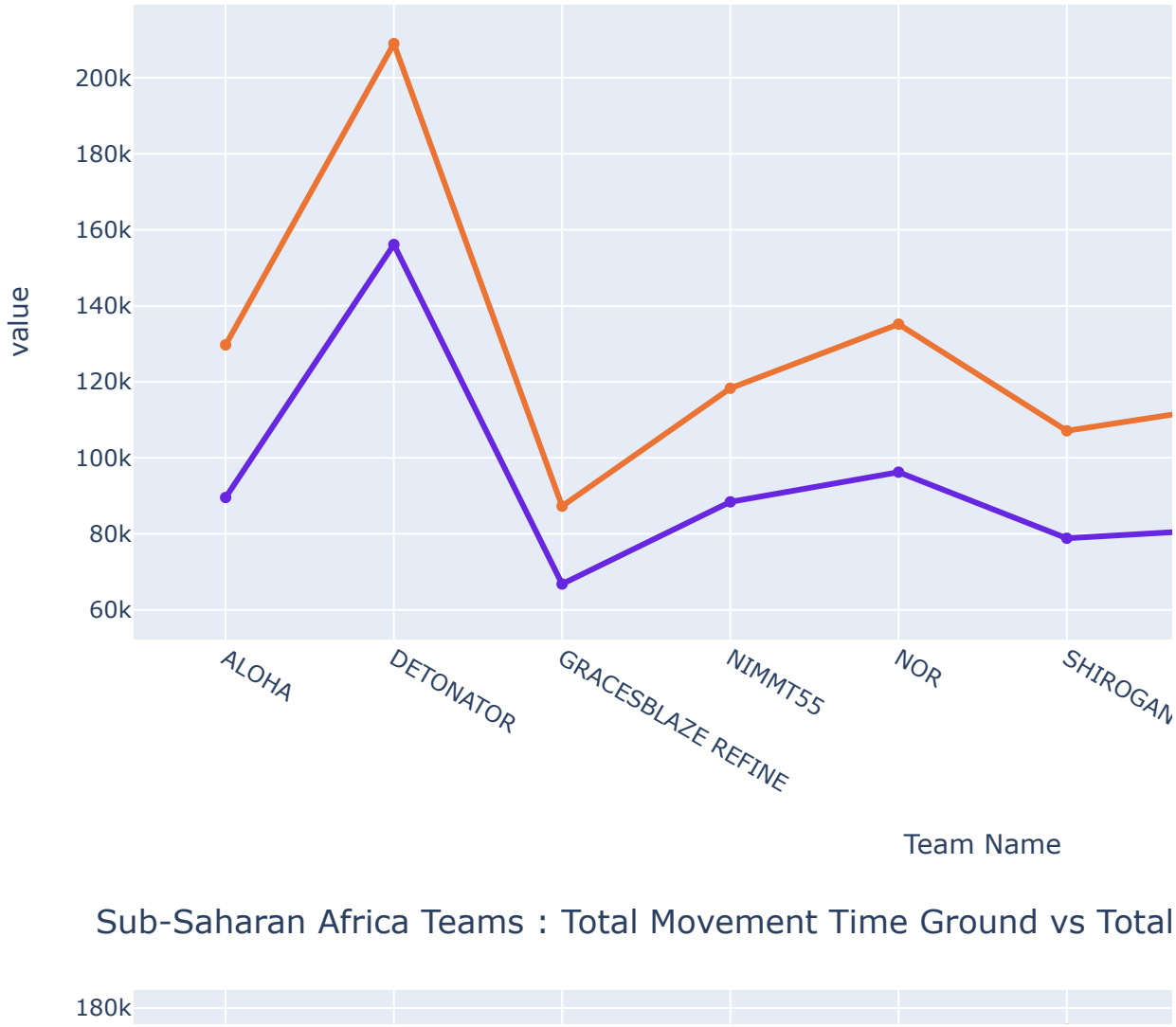
Europe Teams : Total Movement Time Ground vs Total Movement



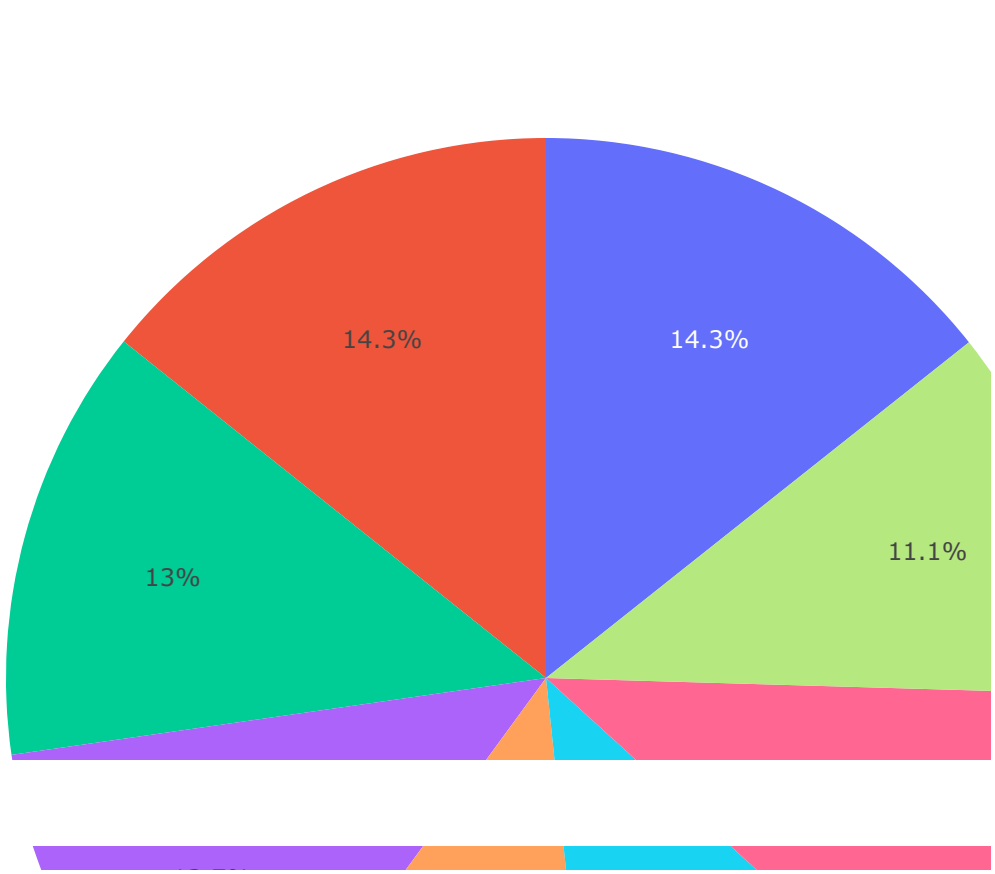
South America Teams : Total Movement Time Ground vs Total Mov



Asia-Pacific North Teams : Total Movement Time Ground vs Total M

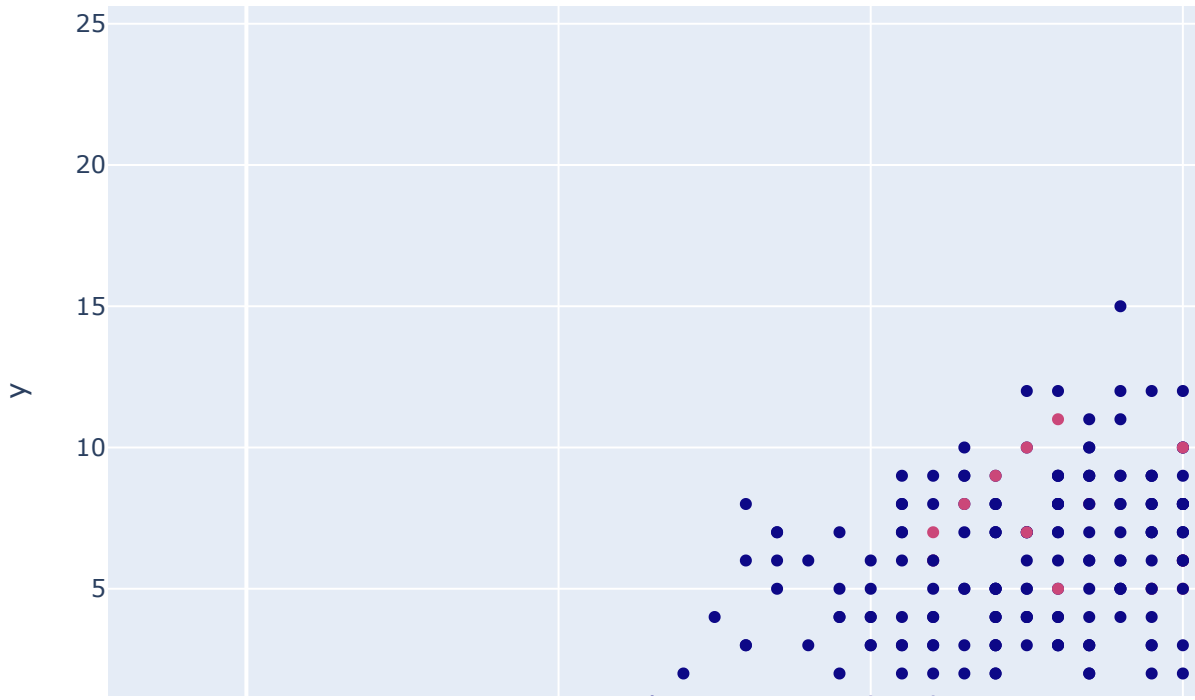



```
# group the data by region and sum up the scores
df_agg = df.groupby("team_region").agg({"score": "sum"}).reset_index()

# create the pie chart
fig = px.pie(df_agg, values='score', names='team_region',title=f'Score',width=960, height=
fig.show()
```

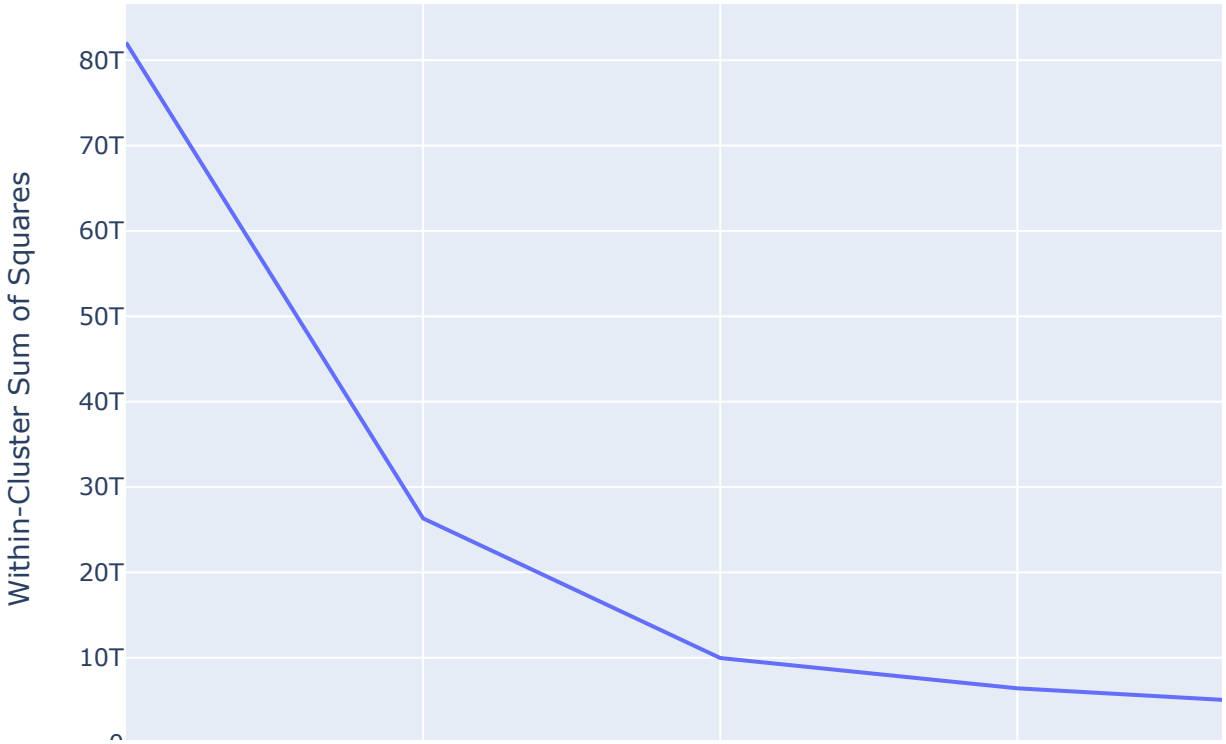


Clustering

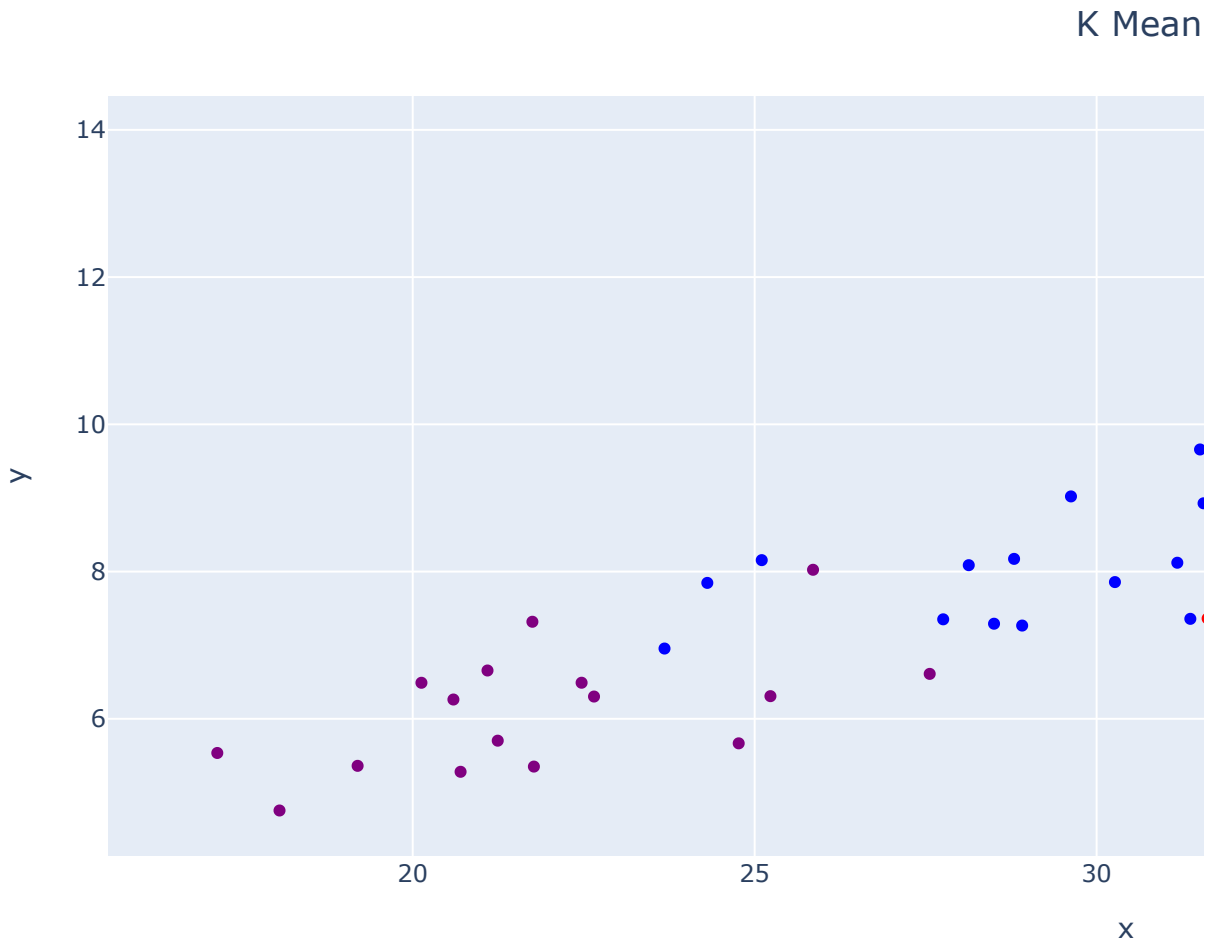
```
df = top_teams[top_teams["team_region"] == "Europe"]

# Select columns 5 to 53
df = df.iloc[:, 7:54]

# Fill missing values with 0
df.fillna(0, inplace=True)

# Perform KMeans clustering
kmeans = KMeans(n_clusters=3)
kmeans.fit(df)

# Add the cluster labels to the dataframe
df["cluster"] = kmeans.labels_
df["team_name"] = top_teams["team_name"]
df["Region"] = top_teams["team_region"]

# Plot the results using Plotly
fig = px.scatter(df, x=df.iloc[:, 0], y=df.iloc[:, 1], color="cluster",hover_data=['team_r
fig.show()
```




```
df = top_teams

# Select columns 5 to 53
df = df.iloc[:, 7:54]

# Fill missing values with 0
df.fillna(0, inplace=True)

# Group the data by team name
grouped = df.groupby(top_teams['team_name']).mean()

# Create an empty list to store the within-cluster sum of squares
wcss = []

# Iterate over a range of cluster numbers and fit KMeans to the grouped data
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(grouped)
    wcss.append(kmeans.inertia_)

fig = px.line(x=(range(1, 11)), y=wcss, labels={'x': 'Number of Clusters', 'y': 'Within-Cl
fig.show()
```




```
df = top_teams

# Select columns 5 to 53
df = df.iloc[:, 7:54]

# Fill missing values with 0
df.fillna(0, inplace=True)

# Group the data by team name
grouped = df.groupby(top_teams['team_name']).mean()
# Perform KMeans clustering on the grouped data
kmeans = KMeans(n_clusters=3)
kmeans.fit(grouped)

team_region_dict = dict(zip(top_teams["team_name"], top_teams["team_region"]))
# Add the cluster labels to the grouped data
grouped["cluster"] = kmeans.labels_
grouped["team_name"] = grouped.index
grouped["Region"] = grouped["team_name"].map(team_region_dict)

# Plot the results using Plotly
fig = px.scatter(grouped, x=grouped.iloc[:, 0], y=grouped.iloc[:, 1], color="cluster", wid
                 hover_data=['team_name', 'Region'],color_continuous_scale=px.colors.seque
fig.update_layout(title_text='K Means',title_x=0.5)
fig.show()
```



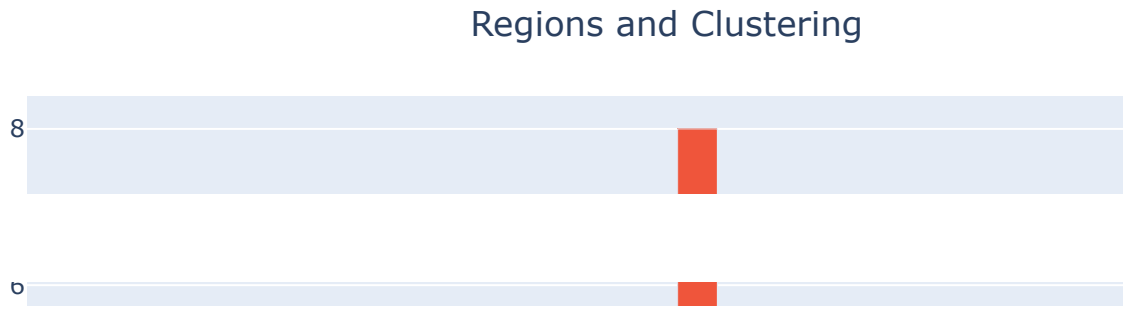

```
import scipy.cluster.hierarchy as shc
import plotly.figure_factory as ff

grouped_D = df.groupby(top_teams['team_name']).mean()
grouped_D["cluster"] = kmeans.labels_
linkage_matrix = shc.linkage(grouped_D, method='ward')
fig = ff.create_dendrogram(linkage_matrix, color_threshold=0)
fig.show()
```

```
''''''
```

```
  list=[0,0,0]
  me=grouped[grouped['Region']==region]
  # print(me.head())
  for value in me['cluster']:
    list[value-1]+=1
  tot.append(list)
res = pd.DataFrame(tot, columns=['Cluster 1', 'Cluster 2', 'Cluster 3'])
res.index = regions
res
```

	Cluster 1	Cluster 2	Cluster 3
Oceania	1	4	5
North America	0	5	5
Asia-Pacific South	2	3	5
Middle East & North Africa	3	4	3
Europe	0	8	2
South America	5	4	1
Asia-Pacific North	2	3	5
Sub-Saharan Africa	3	4	3

```
fig = px.bar(res, x=res.index, y=['Cluster 1', 'Cluster 2', 'Cluster 3'], barmode='group',
fig.update_layout(title_text='Regions and Clustering',title_x=0.5)
fig.show()
```



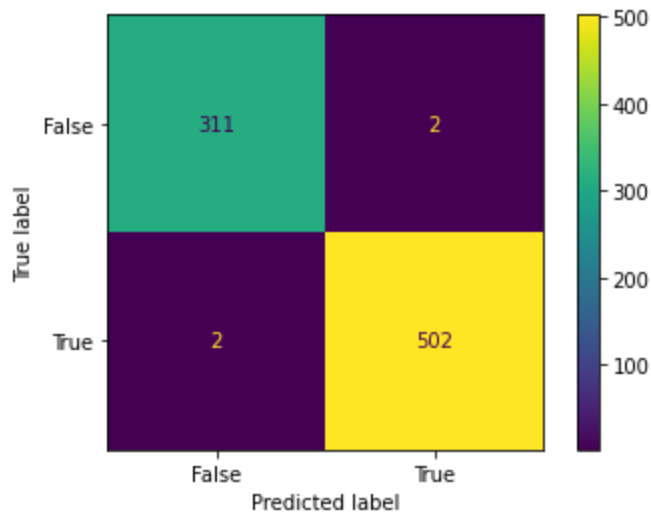
MODEL

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load the data
df = pd.read_csv("/content/preprocessed_matches_by_teams.csv")

# Convert team_name and team_region to one-hot encoded columns
df = pd.get_dummies(df, columns=["team_name", "team_region"])
# Remove columns not useful for prediction
df.drop(["match_id", "color", "team_id", "team_slug", "Mode"], axis=1, inplace=True)
# Replace missing values with the mean of each column
df.fillna(df.mean(), inplace=True)

# Split the data into training and testing sets
X = df.drop("winner", axis=1)
y = df["winner"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Train a Random Forest classifier
clf = RandomForestClassifier(n_estimators=100, random_state=0)
clf.fit(X_train, y_train)

# Predict on the test set
y_pred = clf.predict(X_test)

# Evaluate the model's accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

def predict_winner(team1, team2):
    # Get the data for the two teams
    team1_data = df[df['team_name_' + team1] == 1]
    team2_data = df[df['team_name_' + team2] == 1]
    X1 = team1_data.drop(columns=['winner'])
    X2 = team2_data.drop(columns=['winner'])
    X = pd.concat([X1, X2], axis=0)

    # Predict the winner
    y_pred = clf.predict(X)
```

```
    # Return the predicted winner
    if y_pred[0] == 1:
        return team1
    else:
        return team2
```

```
Accuracy: 0.9891458235016517
```

```
from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred, labels=clf.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=clf.classes_)
disp.plot()
plt.show()
```




```
team1 = 'JUICERS'
team2 = 'PULSE CLAN'
winner = predict_winner(team1, team2)
print(f"The predicted winner between {team1} and {team2} is {winner}.")
```

```
The predicted winner between JUICERS and PULSE CLAN is PULSE CLAN.
```

Colab paid products - Cancel contracts here

✓ 0s completed at 00:15