Analyzing PUBG Data with Python

PlayerUnknown's Battlegrounds (PUBG) has taken the gaming world by storm, offering an immersive battle royale experience. Beyond its gaming aspect, PUBG also provides a wealth of data that can be mined and analyzed to gain insights into player behavior,strategies, and performance.

In this Jupyter Notebook project, we will delve into the exciting world of PUBG data analysis using Python. We will be working with a dataset containing a treasure trove of information,including player statistics, match details, and in-game events. Our goal is to harness the power of Python libraries such as Pandas, NumPy, and Matplotlib to extract meaningful insights from this dataset.

1.Import Libraries

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

2.Upload CSV File

In [2]:
```
df = pd.read_csv('Pubg_Stats.csv')
```

Data Preprocessing

a).head()

head is used show to the By default = 5 rows in the dataset

In [3]:
```
df.head()
```

Out[3]:

	Unnamed: 0	Player_Name	Matches_Played	Kills	Deaths	Assists	Damage_Dealt	Headshots	Wins	Top_10s	Revives	Distance_Traveled	Weapons_Used	Time_Survived	Rank
0	0	StealthMaster	250	587	143	98	15243	234	32	145	67	72560	12	28976	Gold
1	1	SniperLion	312	823	218	112	18975	312	42	189	95	89042	15	33652	Diamond
2	2	NinjaGamer	186	492	84	56	11786	156	28	97	48	60924	10	21764	Platinum
3	3	ThunderStrike	409	923	267	134	21037	288	55	258	128	98234	18	40128	Silver
4	4	SpeedDemon	143	368	68	42	9865	123	20	72	36	52072	8	16834	Gold

b).tail()

tail is used to show rows by Descending order

In [4]:
```
df.tail()
```

Out[4]:

	Unnamed: 0	Player_Name	Matches_Played	Kills	Deaths	Assists	Damage_Dealt	Headshots	Wins	Top_10s	Revives	Distance_Traveled	Weapons_Used	Time_Survived	Rank
216	216	CrimsonRider	294	743	187	132	17567	242	39	133	67	66987	13	28345	Diamond
217	217	BlazingSorcerer	203	521	109	72	13123	193	27	91	45	55487	10	23267	Gold
218	218	FrozenFlare	206	553	117	76	13756	196	29	96	49	58789	10	24579	Platinum
219	219	AbyssGuardian	220	597	144	98	14967	212	32	108	54	60978	11	25679	Platinum
220	220	SpectralPhantom	225	624	149	100	15345	219	33	110	55	61789	11	25967	Platinum

c).shape

It show the total no of rows & Column in the dataset

In [5]:
```
df.shape
```

Out[5]:
```
(221, 15)
```

d).Columns

It Shows the no of each column

In [6]:
```
df.columns
```

Out[6]:
```
Index(['Unnamed: 0', 'Player_Name', 'Matches_Played', 'Kills', 'Deaths',
       'Assists', 'Damage_Dealt', 'Headshots', 'Wins', 'Top_10s', 'Revives',
       'Distance_Traveled', 'Weapons_Used', 'Time_Survived', 'Rank'],
      dtype='object')
```

e).dtypes

This Attribute show the data type of each column

In [7]:
```
df.dtypes
```

Out[7]:
```
Unnamed: 0           int64
Player_Name         object
Matches_Played       int64
Kills                int64
Deaths               int64
Assists              int64
Damage_Dealt         int64
Headshots            int64
Wins                 int64
Top_10s              int64
Revives              int64
Distance_Traveled    int64
Weapons_Used         int64
Time_Survived        int64
Rank                object
dtype: object
```

f).unique()

In a column, It show the unique value of specific column.

In [8]:
```
df["Player_Name"].unique()
```

Out[8]:
```
array(['StealthMaster', 'SniperLion', 'NinjaGamer', 'ThunderStrike',
       'SpeedDemon', 'BlazeFury', 'RapidShadow', 'Frostbite',
       'SavageQueen', 'SwiftStriker', 'VenomousViper', 'PhoenixFury',
       'SteelStorm', 'BlazingBlade', 'StormChaser', 'Nightmare',
       'CrimsonTide', 'SilentShadow', 'VengefulViper', 'SolarFlare',
       'SkyDancer', 'RogueWraith', 'LethalLynx', 'FrostFang',
       'ScarletStrider', 'RagingRaptor', 'ShadowWisp', 'VenomStrike',
       'FireFury', 'BlazingSun', 'ShadowStrike', 'SteelGuardian',
       'WickedWitch', 'RuthlessRaptor', 'FrostyFox', 'ViperVenom',
       'CrimsonReaper', 'PhantomGhost', 'StormStrider', 'StormBreaker',
       'SapphireSword', 'ShadowReign', 'DragonSlayer', 'SilverShadow',
       'EagleEye', 'BlazingStorm', 'MidnightSage', 'RapidBlaze',
       'FrostFire', 'ScarletWitch', 'RagingTiger', 'SpectralRogue',
       'BlazingRaptor', 'EternalShadow', 'WickedStrider', 'CrimsonStorm',
       'RuthlessReaper', 'FrostFury', 'ShadowBlade', 'RapidPhantom',
       'ViperStrike', 'EternalBlaze', 'Vengeance', 'LunarShadow',
       'Deathstrike', 'AzureBlade', 'RavenHeart', 'SerpentFury',
       'CrimsonRogue', 'VoidSeeker', 'AstralSword', 'FrozenFlame',
       'TwilightWarden', 'ShadowPhoenix', 'PhantomStrider', 'EternalFire',
       'NebulaBlade', 'SilverHawk', 'SolarSword', 'EclipseShadow',
       'StarBlade', 'LethalWraith', 'RadiantBlaze', 'FrostGuardian',
       'MysticSerpent', 'InfernoStorm', 'BlazeRanger', 'RagingFire',
       'ShadowDancer', 'PhoenixWings', 'IceStorm', 'MoonlitSorcerer',
       'DarkReaper', 'CosmicGhost', 'StormRider', 'FlareRogue',
       'RadiantBlade', 'TempestPhantom', 'SapphireViper', 'EternalFlame',
       'StarlightBlade', 'CrimsonRider', 'BlazingSorcerer', 'FrozenFlare',
       'AbyssGuardian', 'SpectralPhantom'], dtype=object)
```

g).nuique()

It will show the total no of unque value from whole data frame

In [9]:
```
df.nunique()
```

Out[9]:
```
Unnamed: 0           221
Player_Name          106
Matches_Played        70
Kills                 90
Deaths                82
Assists               65
Damage_Dealt         102
Headshots             70
Wins                  28
Top_10s               59
Revives               39
Distance_Traveled    105
Weapons_Used           9
Time_Survived        107
Rank                   4
dtype: int64
```

h).describe()

It show the Count, mean , median etc

In [10]:
```
df.describe()
```

Out[10]:

	Unnamed: 0	Matches_Played	Kills	Deaths	Assists	Damage_Dealt	Headshots	Wins	Top_10s	Revives	Distance_Traveled	Weapons_Used	Time_Survived
count	221.000000	221.000000	221.000000	221.000000	221.000000	221.000000	221.000000	221.000000	221.000000	221.000000	221.000000	221.000000	221.000000
mean	110.000000	234.624434	612.674208	142.579186	92.615385	14801.004525	207.361991	31.895928	110.357466	54.330317	61449.316742	10.859729	25444.420814
std	63.941379	37.178429	89.311216	32.882564	21.423045	1902.947975	29.775909	5.806190	23.794648	10.436145	5775.529199	1.318835	2699.278875
min	0.000000	143.000000	368.000000	68.000000	42.000000	9865.000000	123.000000	18.000000	67.000000	32.000000	49785.000000	8.000000	16834.000000
25%	55.000000	206.000000	543.000000	117.000000	76.000000	13589.000000	193.000000	28.000000	97.000000	48.000000	57856.000000	10.000000	23879.000000
50%	110.000000	224.000000	604.000000	138.000000	92.000000	14894.000000	210.000000	32.000000	108.000000	54.000000	60986.000000	11.000000	25467.000000
75%	165.000000	257.000000	674.000000	167.000000	111.000000	15987.000000	226.000000	36.000000	125.000000	60.000000	64279.000000	12.000000	26987.000000
max	220.000000	409.000000	923.000000	267.000000	139.000000	21037.000000	312.000000	55.000000	258.000000	128.000000	98234.000000	18.000000	40128.000000

i).value_counts

It Shows all the unique values with their count

In [11]:
```
df["Player_Name"].value_counts()
```

Out[11]:
```
VengefulViper      7
Frostbite          5
VenomousViper      5
LethalLynx         4
Nightmare          4
                  ..
MidnightSage       1
BlazingStorm       1
EagleEye           1
SilverShadow       1
SpectralPhantom    1
Name: Player_Name, Length: 106, dtype: int64
```

j).isnull()

It shows the how many null values

In [12]:
```
df.isnull()
```

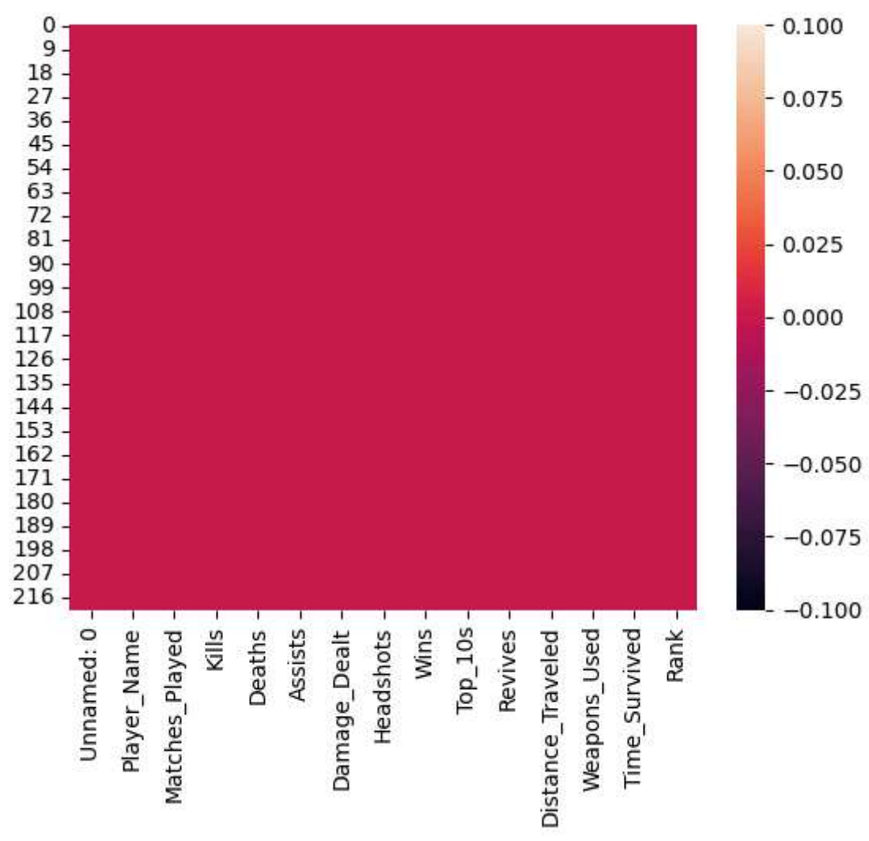
Out[12]:

	Unnamed: 0	Player_Name	Matches_Played	Kills	Deaths	Assists	Damage_Dealt	Headshots	Wins	Top_10s	Revives	Distance_Traveled	Weapons_Used	Time_Survived	Rank
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
216	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
217	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
218	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
219	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
220	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False

221 rows × 15 columns

In [13]:
```
# Various Plots
# 1.HeatMap
```

In [14]:
```
sns.heatmap(df.isnull())
plt.show()
```



In [15]:
```
df.isna().sum()
```

Out[15]:
```
Unnamed: 0          0
Player_Name         0
Matches_Played      0
Kills               0
Deaths              0
Assists             0
Damage_Dealt        0
Headshots           0
Wins                0
Top_10s             0
Revives             0
Distance_Traveled   0
Weapons_Used        0
Time_Survived       0
Rank                0
dtype: int64
```

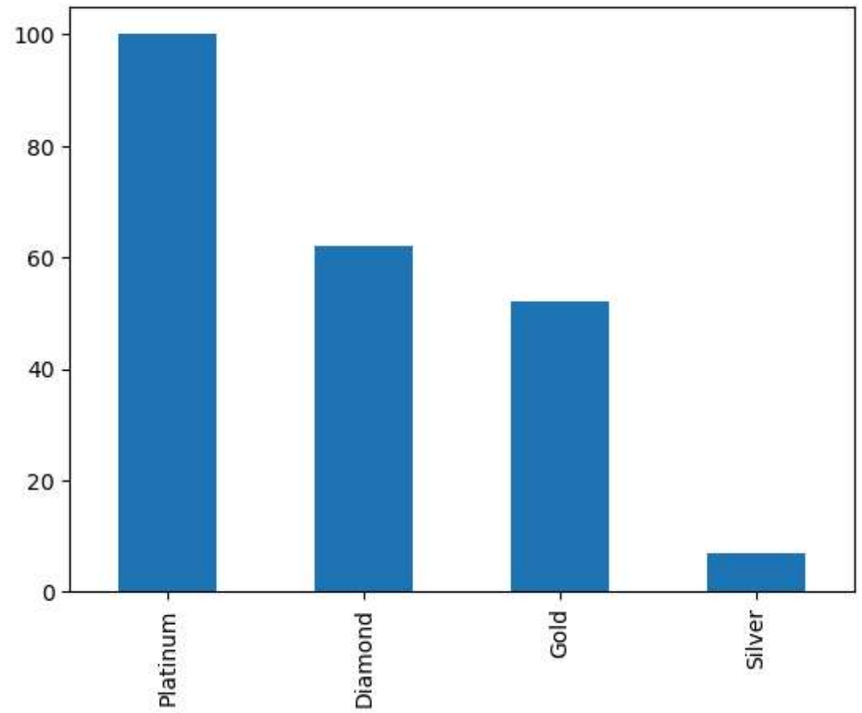
Drop the Unnamed Column

In [16]:
```
df.drop(['Unnamed: 0'], axis=1, inplace=True)
```

Show the Rank in Barplot

In [17]:
```
df.Rank.value_counts().plot(kind = "bar")
```

Out[17]: <Axes: >



Top 10 players By Matches Played

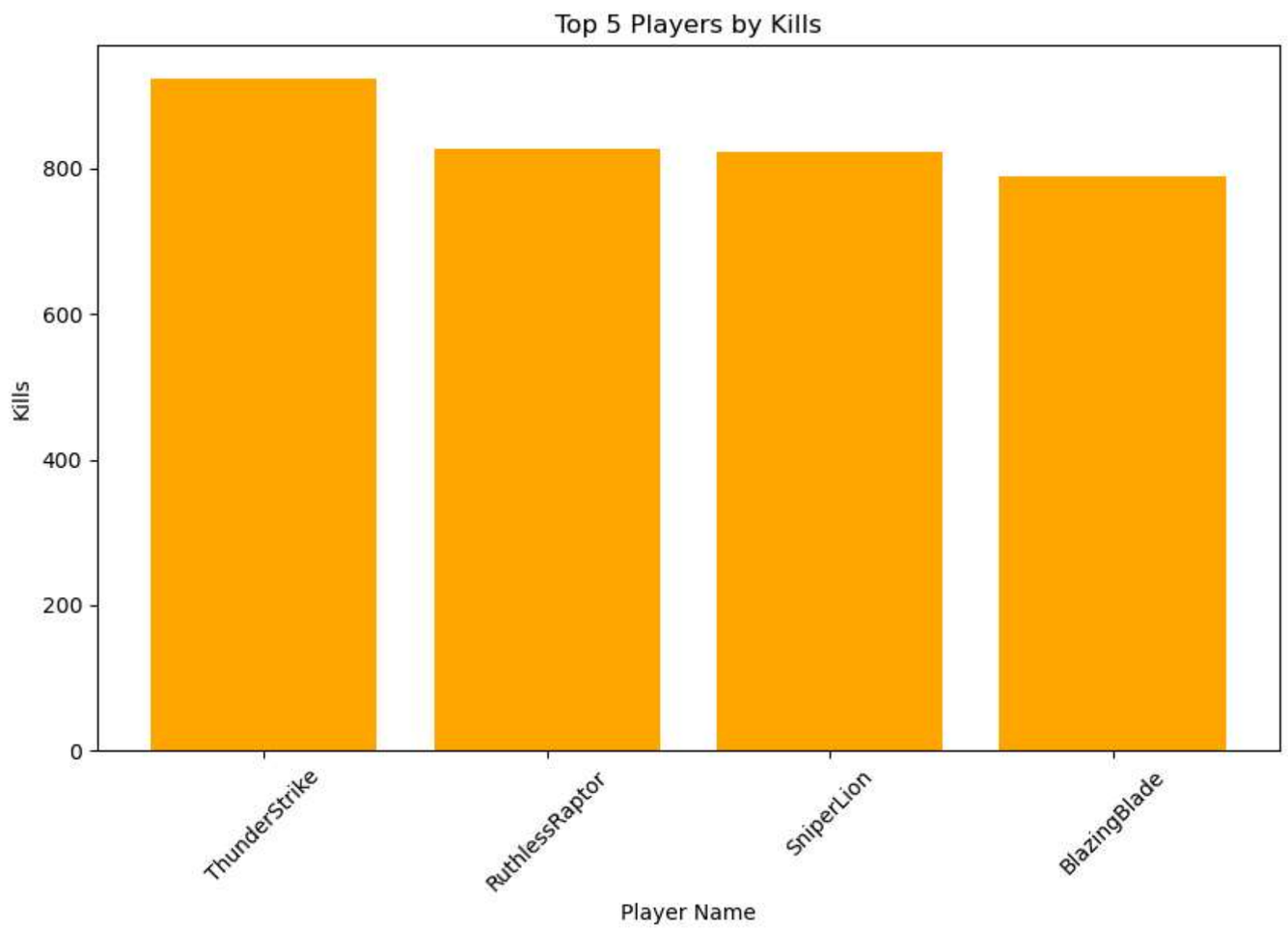
In [18]:
```
# Sort the DataFrame by 'Matches_Played' column in descending order and select the top 10 rows
top_10_players = df.sort_values(by='Matches_Played', ascending=False).head(10)
fig = px.bar(top_10_players, x="Player_Name", y="Matches_Played", title="Top 10 Players")
# Customizing colors
fig.update_traces(marker_color='skyblue')
# Show the plot
fig.show()
```

Top 5 players By Matches Played

In [19]:
```
# Sort the DataFrame by 'Matches_Played' column in descending order and select the top 5 rows
top_5_players = df.sort_values(by='Matches_Played', ascending=False).head(5)
fig = px.bar(top_5_players, x="Player_Name", y="Matches_Played", title="Top 5 Players")
# Customizing colors to orange
fig.update_traces(marker_color='orange')
# Show the plot
fig.show()
```

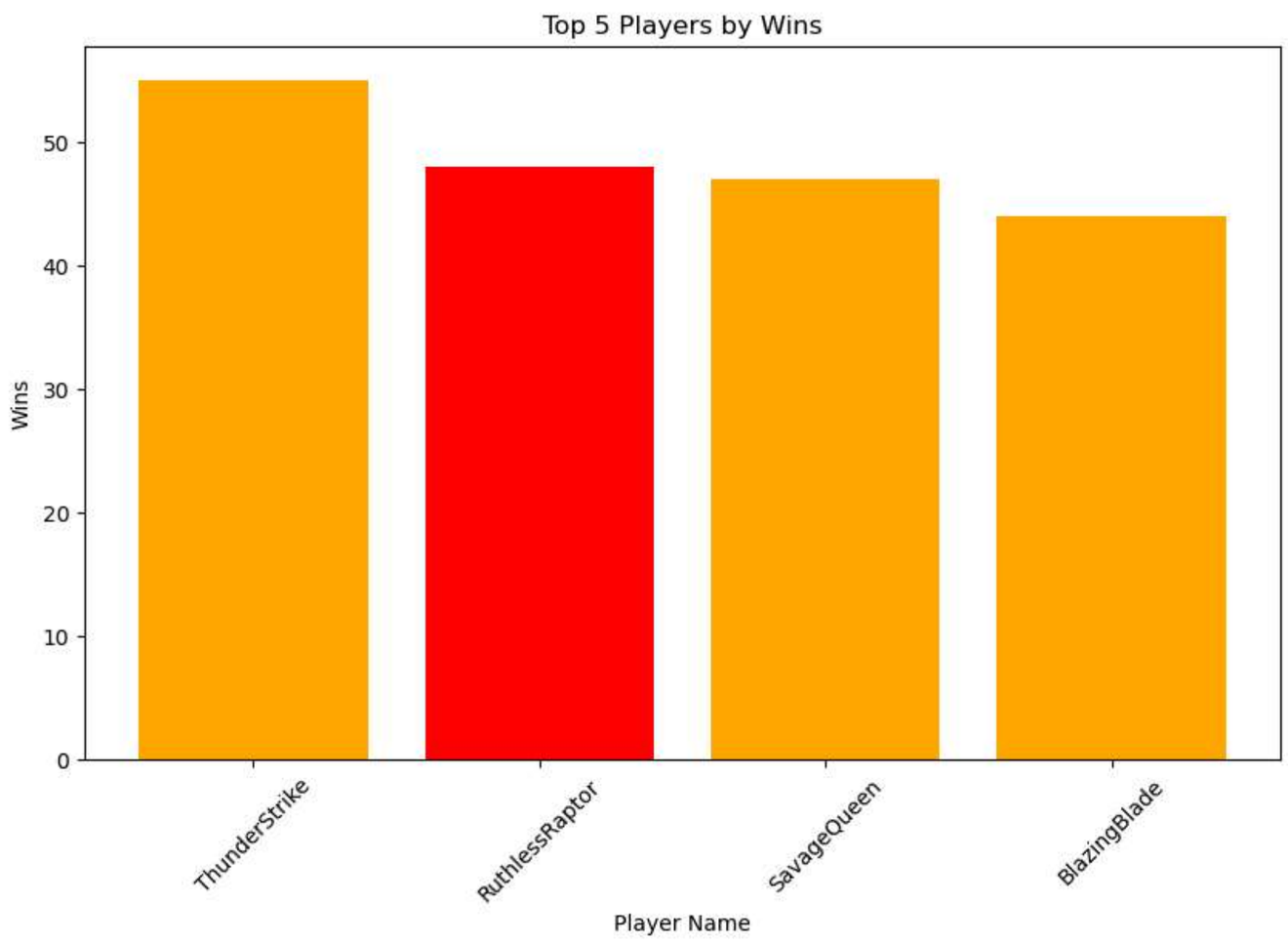
In [20]:
```
# Top 5 players By Kills
```

In [21]:
```
# Sort the DataFrame by "Kills" in descending order
df = df.sort_values(by="Kills", ascending=False)

# Select the top 5 players with the highest kills
top_5_players = df.head(5)

# Create a bar plot for the top 5 players
plt.figure(figsize=(10, 6))

plt.bar(top_5_players["Player_Name"], top_5_players["Kills"], color='orange')  # Color bars orange
plt.xlabel("Player Name")
plt.ylabel("Kills")
plt.title("Top 5 Players by Kills")
plt.xticks(rotation=45)
plt.show()
```

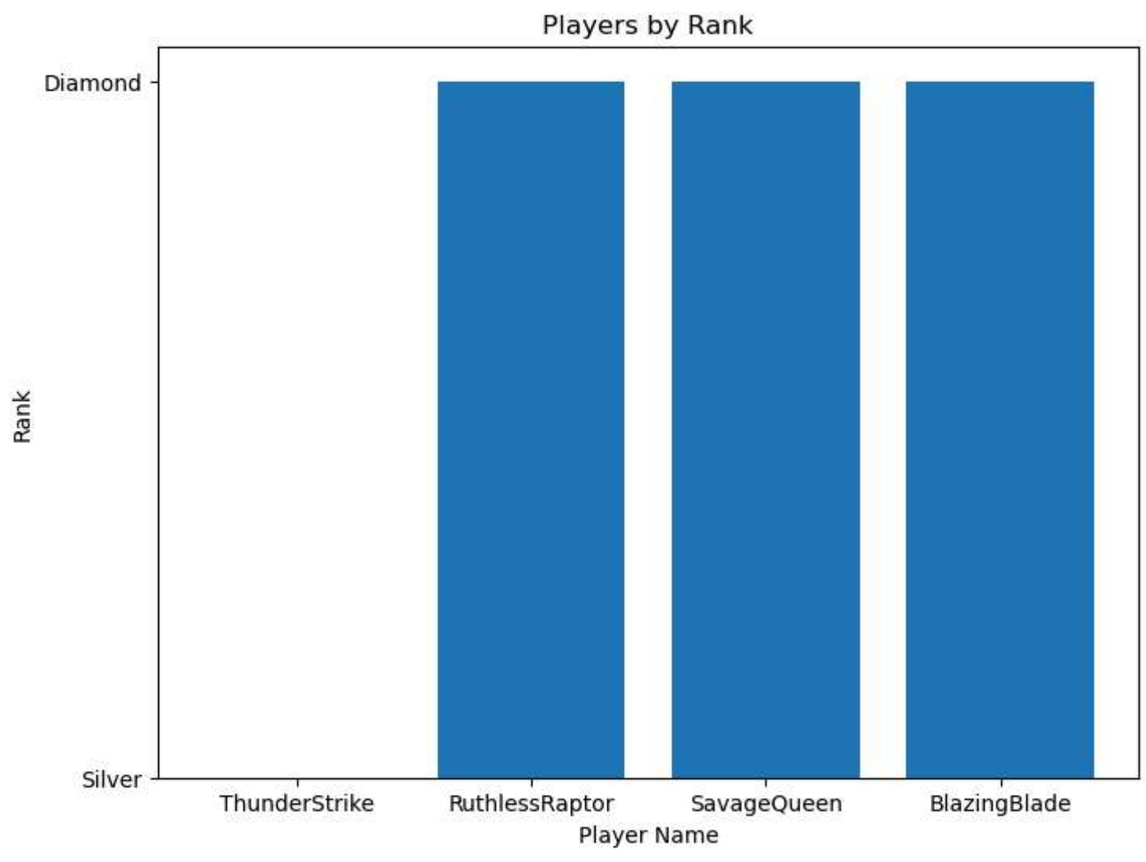


Top 5 players By Wins

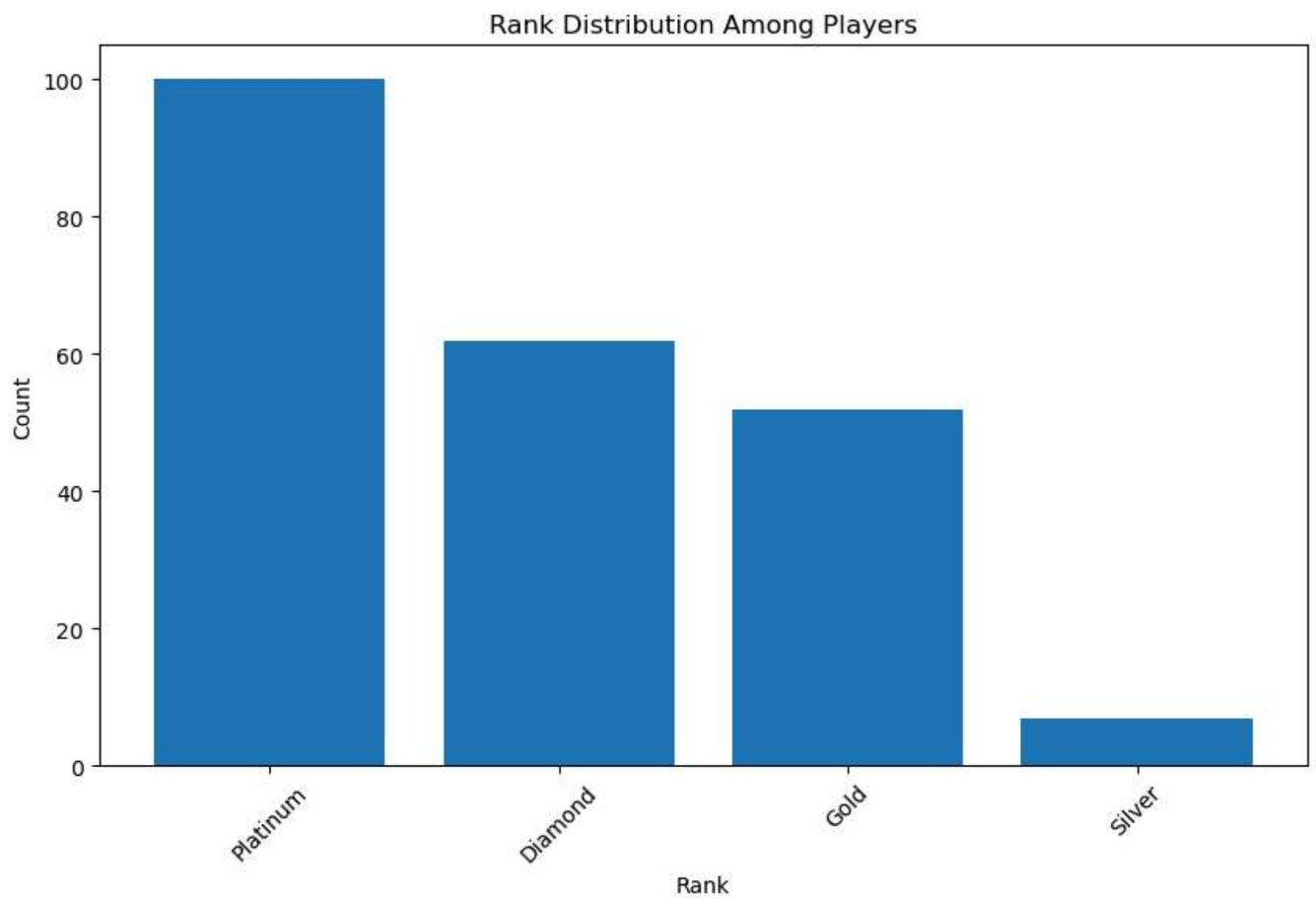
In [22]:
```python
# Sort the DataFrame by "Wins" in descending order
df = df.sort_values(by="Wins", ascending=False)
# Select the top 5 players with the highest wins
top_5_players = df.head(5)
# Create a color palette with a mix of orange and red
colors = ['orange', 'red', 'orange', 'red', 'orange']
# Create a bar plot for the top 5 players with custom colors
plt.figure(figsize=(10, 6))
plt.bar(top_5_players["Player_Name"], top_5_players["Wins"], color=colors)
plt.xlabel("Player Name")
plt.ylabel("Wins")
plt.title("Top 5 Players by Wins")
plt.xticks(rotation=45)
plt.show()
```



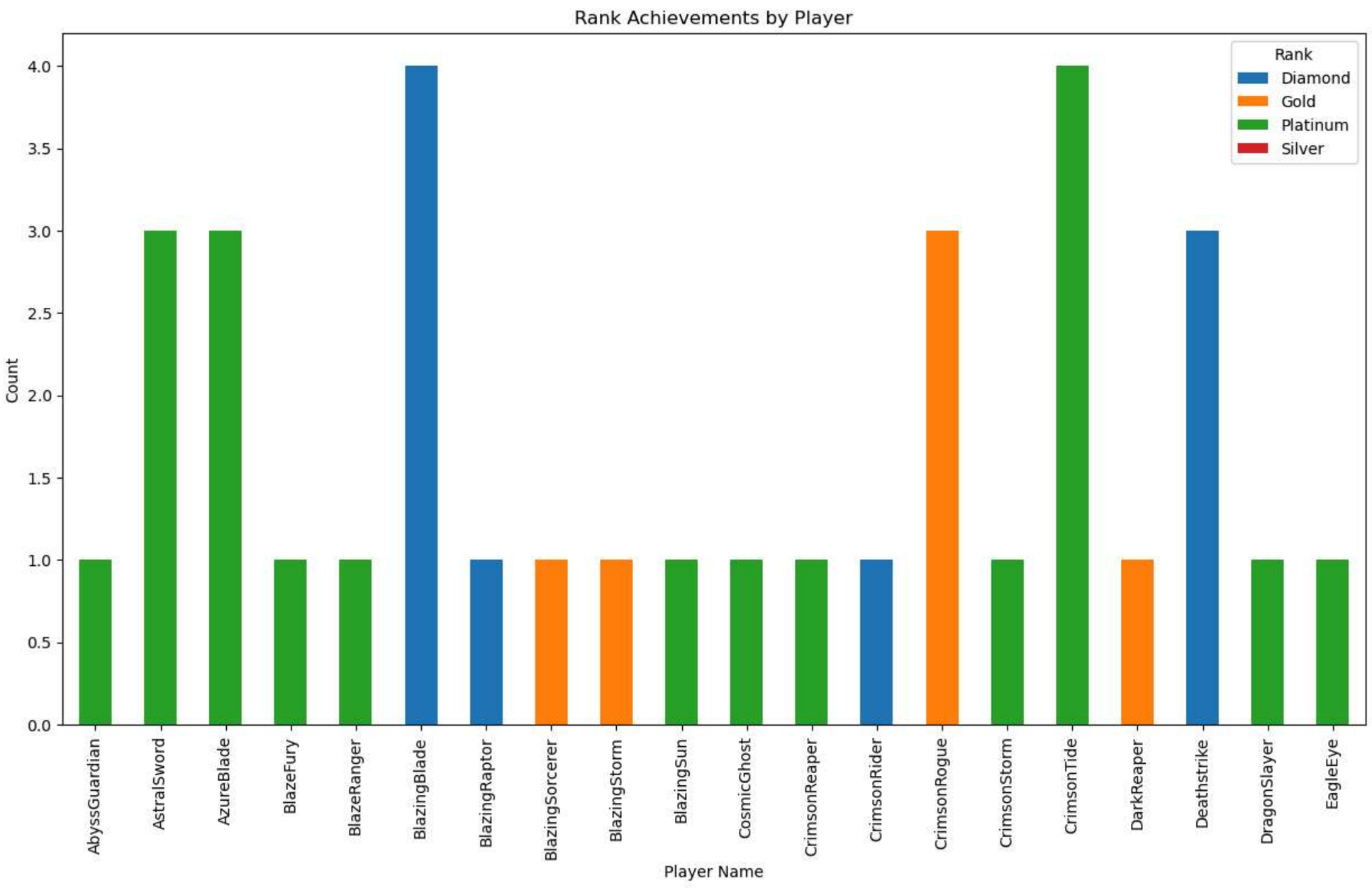
In [23]:
```python
# Player Name By Rank
# Sort the DataFrame by "Matches Played" in descending order
df = df.sort_values(by="Matches_Played", ascending=False)
# Select the top 5 players with the highest matches played
top_5_players = df.head(5)
plt.figure(figsize=(8, 6))
plt.bar(top_5_players["Player_Name"], top_5_players["Rank"])
plt.xlabel('Player Name')
plt.ylabel('Rank')
plt.title('Players by Rank')
plt.show()
```



In [24]:
```
# How many times Rank are Published
rank_counts = df["Rank"].value_counts().reset_index()
rank_counts.columns = ["Rank", "Count" ]
rank_counts = rank_counts.sort_values(by="Count", ascending=False)

plt.figure(figsize=(10, 6))
plt.bar(rank_counts["Rank"], rank_counts["Count"])
plt.xlabel("Rank")

plt.ylabel("Count")

plt.title("Rank Distribution Among Players")
plt.xticks(rotation=45)

plt.show()
```



In [25]:
```
# How many Times Rank Published by Player Name
# Create a cross-tabulation to count how many times each rank was achieved by ¢
cross_tab = pd.crosstab(df["Player_Name"], df["Rank"]).head(20)
# PLot the bar chart
cross_tab.plot(kind="bar", stacked=True, figsize=(15, 8))
# Customize the plot

plt.xlabel("Player Name")

plt.ylabel("Count")
plt.title("Rank Achievements by Player")

# Show the plot
plt.show()
```


Rank Achievements by Player
Rank
Diamond
Gold
Platinum
Silver
Count
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
AbyssGuardian
AstralSword
AzureBlade
BlazeFury
BlazeRanger
BlazingBlade
BlazingRaptor
BlazingSorcerer
BlazingStorm
BlazingSun
CosmicGhost
CrimsonReaper
CrimsonRider
CrimsonRogue
CrimsonStorm
CrimsonTide
DarkReaper
Deathstrike
DragonSlayer
EagleEye
Player Name

In [ ]: