League of Legends Pro-Players performance trends compared in off-seasons and tournaments.

Project Details

Purpose

This project aims to **analyze and compare the performance of professional League of Legends players** in two distinct competitive environments: **off-season games** and **official tournament matches**. By evaluating differences in measurable performance metrics across both contexts, the goal is to identify **whether players exhibit statistically significant variations** in gameplay behavior depending on match type.

Performance Metrics

The following metrics are used to assess and compare the individual performance of pro players:

1. **Kill/Death/Assist (KDA) Ratio** – Represents a player's combat efficiency, calculated as *(Kills + Assists) / Deaths*. A higher KDA generally indicates more impactful play and fewer mistakes.

2. **Damage Dealt** – Measures the total damage a player inflicts on enemy champions, providing insight into their offensive contribution.

3. **Gold Earned per Minute (GPM)** – Captures how efficiently a player generates gold throughout a match, often through farming and objective control.

4. **Creep Score (CS)** – Tracks the number of minions or neutral monsters a player kills. It's a core indicator of farming ability and overall resource acquisition.

5. **Custom Performance Score** – A weighted metric created for this project to combine multiple in-game statistics (KDA, CS/min, damage/min, gold/min, and win outcome) into a single performance indicator.

Commonly Used Terms

- **Summoner** – Refers to the player.
- **Champion** – Refers to the character the player controls in-game.
- **Farming** – The act of killing minions or neutral monsters to gain gold and CS.

Role Context

In League of Legends, each team consists of five unique roles:

- **Top Laner**
- **Jungler**
- **Mid Laner**
- **AD Carry (ADC)**
- **Support**

For the purpose of this analysis, the **Support role has been excluded**. Unlike other roles, support performance relies more on **team coordination, vision control, and utility**, rather than individually quantifiable metrics like KDA, CS, or gold. As such, this project focuses on roles with **clear, performance-driven statistics**.

Selected Pro Players

The following players were selected based on their prominence, availability of data, and diversity across **roles, regions, and leagues**:

Player	Role	Region	League	Team	Summoner / Account Name
Faker	Mid Laner	South Korea	LCK	T1	Hide on bush
Chovy	Mid Laner	South Korea	LCK	Gen.G	허거덩
Ruler	ADC	South Korea	LPL	JD Gaming	귀찮게하지마
Caps	Mid Laner	Denmark	LEC	G2 Esports	G2 Caps
Blaber	Jungler	United States	LCS	Cloud9	blaberfish2
Zeus	Top Laner	South Korea	LCK	T1	Zeus10
Hans Sama	ADC	France	LEC	G2 Esports	G2 Hans Sama

These players represent a broad range of styles and strengths, providing a rich dataset for **cross-role**, **cross-league**, and **cross-contextual** performance comparison.

1. Import Libraries

We'll begin by importing the libraries needed for data analysis and visualization.

In [70]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Gathering and Organizing the Scraped Data

The data was collected from the following sources:

1. **Riot Games League of Legends Client**
2. **OP.GG** — primarily used for collecting off-season match data
3. **Riot Games Developer API** — used to retrieve match histories and performance statistics
4. **LoL Fandom Wiki** — primarily used for gathering tournament match data

Each player's match history was compiled into two CSV files, one for **off-season** and one for **tournament** games. These files are stored in the `data/` directory of the project.

- A total of **14 CSV files** were created (2 per player × 7 players).
- Each file contains data from **10 matches**, ensuring consistency and comparability.
- For **tournament games**, only the **first match of each series** (e.g., best-of-3 or best-of-5) was considered in order to avoid introducing skill bias from repetitive matchups against the same team.

3. Process the data in the CSV files for each player

The following code is responsible for importing, separating, enriching, and combining all CSV files. This processing step prepares the dataset for subsequent Exploratory Data Analysis (EDA) and hypothesis testing.

In [69]:
```
def enrich_player_data(filepath, player_name, match_type):
    df = pd.read_csv(filepath)

    # function to convert game time from its current format to something that ca
    def parse_game_time(time_str):
        min_part, sec_part = time_str.strip().upper().split("M")
        minutes = int(min_part.strip())
        seconds = int(sec_part.replace("S", "").strip())
        return minutes + seconds / 60

    # Overall enrichment of the data using gametime, and making a personal perfo
    df['GameMinutes'] = df['Game Time'].apply(parse_game_time)
    df['KDA'] = (df['Kills'] + df['Assists']) / df['Deaths'].replace(0, 1) # Cal
    df['CS_per_min'] = df['CS'] / df['GameMinutes'] # Enrichment by converting C
    df['Damage_per_min'] = df['Damage'] / df['GameMinutes'] # Enrichment by conv
    df['Gold_per_min'] = df['Gold'] / df['GameMinutes'] # Enrichment by converti
    df['Win'] = df['Win/Loss'].apply(lambda x: 1 if x.lower() == 'win' else 0) #

    # Safely ignore missing columns like Performance(OP.GG) because tournament C
    if 'Performance(OP.GG)' in df.columns:
        df.drop(columns=['Performance(OP.GG)'], inplace=True)

    # Creating a personal performance indicator giving weightage to different me
```

```
    df['PerformanceScore'] = (
        df['KDA'] * 0.3 +
        df['CS_per_min'] * 0.2 +
        df['Damage_per_min'] * 0.2 +
        df['Gold_per_min'] * 0.2 +
        df['Win'] * 0.1
    )
    df['Player'] = player_name
    df['MatchType'] = match_type
    return df

# Calling above function to enrich each player's DataFrame
faker_off = enrich_player_data("data/faker_offseason_games.csv", "Faker", "Offse
faker_tourn = enrich_player_data("data/faker_tournament_games.csv", "Faker", "To

chovy_off = enrich_player_data("data/chovy_offseason_games.csv", "Chovy", "Offse
chovy_tourn = enrich_player_data("data/chovy_tournament_games.csv", "Chovy", "To

ruler_off = enrich_player_data("data/ruler_offseason_games.csv", "Ruler", "Offse
ruler_tourn = enrich_player_data("data/ruler_tournament_games.csv", "Ruler", "To

caps_off = enrich_player_data("data/caps_offseason_games.csv", "Caps", "Offseaso
caps_tourn = enrich_player_data("data/caps_tournament_games.csv", "Caps", "Tourn

blaber_off = enrich_player_data("data/blaber_offseason_games.csv", "Blaber", "Of
blaber_tourn = enrich_player_data("data/blaber_tournament_games.csv", "Blaber",

zeus_off = enrich_player_data("data/zeus_offseason_games.csv", "Zeus", "Offseaso
zeus_tourn = enrich_player_data("data/zeus_tournament_games.csv", "Zeus", "Tourn

hanssama_off = enrich_player_data("data/hanssama_offseason_games.csv", "Hans Sam
hanssama_tourn = enrich_player_data("data/hanssama_tournament_games.csv", "Hans

# Combine into grouped dataframes
offseason_df = pd.concat([
    faker_off, chovy_off, ruler_off, caps_off, blaber_off, zeus_off, hanssama_of
], ignore_index=True)

tournament_df = pd.concat([
    faker_tourn, chovy_tourn, ruler_tourn, caps_tourn, blaber_tourn, zeus_tourn,
], ignore_index=True)

# final dataframe containing all 140 datasets
all_games_df = pd.concat([offseason_df, tournament_df], ignore_index=True)

# Checking if everything is loaded correctly
print("All games loaded:", all_games_df.shape[0])
print(all_games_df['MatchType'].value_counts())

# Displaying 20 games for each player in a table format
for player in all_games_df['Player'].unique():
    print(f"\n=== {player} ===")
    display(all_games_df[all_games_df['Player'] == player])
```

```
All games loaded: 140
MatchType
Offseason     70
Tournament    70
Name: count, dtype: int64

=== Faker ===
```

	Game	Champion	Kills	Deaths	Assists	CS	Damage	Gold	Win/Loss	Game Time	Gar
0	1	Jayce	12	2	7	327	70810	21258	Win	40M 09S	
1	2	Viktor	12	4	10	214	37472	13272	Win	26M 41S	
2	3	Orianna	5	4	4	220	23306	10333	Loss	23M 48S	
3	4	Yone	4	7	16	234	29644	13445	Win	29M 41S	
4	5	Yone	5	4	4	266	24291	13105	Loss	28M 28S	
5	6	Sylas	3	4	11	180	22045	9443	Loss	26M 04S	
6	7	Sylas	3	6	8	290	22854	13932	Loss	38M 33S	
7	8	Viktor	7	3	10	207	25315	11464	Win	23M 59S	
8	9	Sylas	3	5	5	236	20173	11499	Loss	28M 26S	
9	10	Galio	2	5	11	206	16990	10926	Loss	26M 52S	
70	1	Viktor	5	3	7	377	19000	33500	Win	44M 36S	
71	2	Viktor	1	2	1	249	26000	14300	Loss	28M 32S	
72	3	Viktor	2	1	14	304	30500	14100	Win	35M 21S	
73	4	Viktor	0	7	1	241	24300	8900	Loss	27M 54S	
74	5	Viktor	2	2	5	270	16800	11700	Win	30M 11S	
75	6	Taliyah	3	4	5	445	42600	19500	Loss	50M 39S	
76	7	Sylas	3	3	7	284	28600	13300	Win	34M 05S	
77	8	Zoe	3	2	6	356	24900	16100	Loss	43M 08S	
78	9	Corki	10	4	8	275	40300	16600	Win	30M 59S	
79	10	Galio	2	1	11	296	11700	14700	Win	40M 41S	

```
=== Chovy ===
```

	Game	Champion	Kills	Deaths	Assists	CS	Damage	Gold	Win/Loss	Game Time	Gar
10	1	Azir	6	8	6	209	23492	9576	Loss	25M 32S	
11	2	Zoe	12	6	12	201	37663	12593	Win	30M 45S	
12	3	Corki	9	6	8	266	35799	14562	Win	29M 46S	
13	4	Ahri	6	2	11	246	25873	12507	Win	25M 07S	
14	5	Zoe	3	1	6	133	13753	7121	Win	15M 37S	
15	6	Corki	3	10	5	335	43207	15366	Loss	36M 20S	
16	7	Xin Zhao	6	3	3	132	10453	7814	Win	19M 33S	
17	8	Viktor	6	5	10	215	29290	12959	Win	27M 33S	
18	9	Jax	10	9	15	271	46416	17726	Loss	45M 20S	
19	10	Sylas	18	10	11	225	59942	16088	Win	33M 04S	
80	1	Sylas	13	0	6	246	29200	15200	Win	28M 05S	
81	2	Viktor	6	0	12	264	30500	14300	Win	28M 48S	
82	3	Ryze	6	0	4	262	22800	12900	Win	24M 34S	
83	4	Sylas	2	3	6	285	18300	13500	Win	32M 24S	
84	5	Viktor	5	1	3	294	23000	13700	Win	28M 07S	
85	6	Viktor	7	0	7	458	54000	22600	Win	50M 39S	
86	7	Azir	3	2	15	310	23800	14600	Win	29M 10S	
87	8	Aurelion	2	1	0	382	29500	15300	Loss	38M 59S	
88	9	Ahri	4	3	9	247	16100	12300	Win	27M 16S	
89	10	Azir	3	2	6	341	24400	15500	Win	32M 35S	

```
=== Ruler ===
```

	Game	Champion	Kills	Deaths	Assists	CS	Damage	Gold	Win/Loss	Game Time	Gar
20	1	Kaisa	9	7	10	296	34003	17579	Win	32M 43S	
21	2	Kalista	8	3	8	142	19623	9798	Win	18M 25S	
22	3	Jhin	1	3	3	95	7800	5146	Loss	15M 31S	
23	4	Xayah	16	6	12	308	50511	16470	Win	29M 41S	
24	5	Ezreal	8	1	4	173	21703	10586	Win	18M 32S	
25	6	Varus	4	1	3	170	10001	9075	Win	17M 36S	
26	7	Ezreal	4	3	6	181	15155	8573	Win	20M 31S	
27	8	Varus	16	10	7	288	50033	18988	Loss	38M 40S	
28	9	Kaisa	13	9	10	321	50482	17760	Win	30M 49S	
29	10	Kaisa	1	4	2	180	8255	8002	Loss	20M 36S	
90	1	Xayah	7	0	12	275	27600	13700	Win	28M 05S	
91	2	Miss Fortune	7	1	5	304	18100	14600	Win	28M 48S	
92	3	Jhin	11	0	7	330	29700	17500	Win	33M 53S	
93	4	Jhin	6	0	5	229	21900	11700	Win	24M 34S	
94	5	Kalista	2	1	7	337	11700	14400	Win	32M 24S	
95	6	Ezreal	7	0	7	280	27000	13600	Win	28M 07S	
96	7	Xayah	5	2	8	539	34800	25200	Win	50M 39S	
97	8	Corki	11	3	8	255	33100	14500	Win	29M 10S	
98	9	Ashe	11	2	5	241	19000	14600	Win	27M 16S	
99	10	Zeri	8	1	4	380	33700	19000	Win	32M 35S	

```
=== Caps ===
```

	Game	Champion	Kills	Deaths	Assists	CS	Damage	Gold	Win/Loss	Game Time	Ga
30	1	Ryze	3	2	12	205	25098	11244	Win	28M 05S	
31	2	Ryze	7	5	7	142	12229	8980	Win	19M 35S	
32	3	Syndra	4	1	10	233	24047	12243	Win	24M 25S	
33	4	Azir	8	10	11	289	33254	16789	Loss	43M 54S	
34	5	Taliyah	2	6	4	217	13974	9733	Loss	27M 12S	
35	6	Aurora	14	6	8	225	45407	16223	Win	32M 05S	
36	7	Zed	7	9	4	172	15768	10854	Loss	28M 32S	
37	8	Taliyah	10	7	12	267	25916	16051	Win	34M 34S	
38	9	Viktor	1	6	0	107	4641	4603	Loss	15M 12S	
39	10	Gragas	6	8	2	179	17997	10269	Loss	24M 54S	
100	1	Hwei	4	2	5	399	39800	19200	Loss	45M 35S	
101	2	Akali	5	0	3	311	23500	15700	Win	36M 08S	
102	3	Hwei	3	0	10	287	15600	13500	Win	35M 38S	
103	4	Viktor	1	1	8	255	19400	11600	Win	29M 02S	
104	5	Ryze	0	6	5	181	11400	8500	Loss	24M 37S	
105	6	Anivia	2	0	5	206	13800	9500	Win	26M 12S	
106	7	Azir	5	2	9	282	27500	13700	Win	29M 10S	
107	8	Aurora	4	1	5	316	16200	13900	Win	30M 29S	
108	9	Ahri	7	2	8	241	28100	12700	Win	27M 57S	
109	10	Aurora	1	1	10	266	13900	12000	Win	29M 28S	

```
=== Blaber ===
```

	Game	Champion	Kills	Deaths	Assists	CS	Damage	Gold	Win/Loss	Game Time	Ga
40	1	Khazix	6	3	10	194	16528	11014	Win	24M 21S	
41	2	Viego	10	9	5	193	20011	13504	Loss	29M 28S	
42	3	Vi	8	3	10	229	19827	13843	Win	28M 57S	
43	4	Lilliah	11	5	12	290	39073	17055	Win	32M 29S	
44	5	Pantheon	9	3	5	195	21320	11977	Win	26M 20S	
45	6	Nidalee	9	4	17	186	28925	12391	Loss	29M 28S	
46	7	Viego	5	9	2	192	16734	12247	Loss	27M 59S	
47	8	Naafiri	5	4	7	198	15622	12066	Loss	28M 05S	
48	9	Lilliah	10	2	1	122	11684	7908	Win	15M 16S	
49	10	Zyra	3	1	11	155	10730	8335	Win	18M 04S	
110	1	Sejuani	3	0	12	196	10600	10700	Win	26M 29S	
111	2	Vi	2	3	7	209	6200	10700	Win	28M 27S	
112	3	Vi	4	1	4	224	10700	11900	Win	27M 45S	
113	4	Vi	3	2	15	280	16700	14600	Win	34M 40S	
114	5	Poppy	2	1	6	278	8000	13600	Win	34M 46S	
115	6	Vi	1	4	5	222	9800	11200	Win	35M 04S	
116	7	LeeSin	4	6	5	249	15900	13400	Loss	38M 25S	
117	8	Wukong	2	5	3	202	9000	10600	Loss	32M 57S	
118	9	Vi	5	2	10	216	12100	12100	Win	29M 34S	
119	10	Wukong	9	2	7	228	12800	13500	Win	30M 58S	

```
=== Zeus ===
```

	Game	Champion	Kills	Deaths	Assists	CS	Damage	Gold	Win/Loss	Game Time	C
50	1	Mordekaiser	8	9	15	277	47959	16946	Win	43M 03S	
51	2	Renekton	2	5	2	175	18145	7392	Loss	29M 52S	
52	3	Sion	5	8	6	210	20303	11788	Win	28M 53S	
53	4	Sion	1	7	5	216	20846	10833	Loss	27M 42S	
54	5	Renekton	3	9	3	207	20647	11264	Loss	28M 29S	
55	6	Akali	6	8	9	190	31174	11613	Win	29M 41S	
56	7	Akali	13	3	10	239	36963	15587	Win	34M 01S	
57	8	Viktor	4	5	2	181	21054	9380	Loss	21M 43S	
58	9	Ambessa	9	5	6	347	44664	18630	Win	36M 09S	
59	10	Ambessa	7	3	7	267	36538	14569	Win	30M 49S	
120	1	Renekton	2	6	10	292	24800	15100	Win	35M 24S	
121	2	Jax	9	1	8	178	24300	12100	Win	27M 32S	
122	3	Nidalee	1	4	3	296	29800	15100	Win	32M 00S	
123	4	Jax	5	2	6	236	22900	12100	Win	27M 54S	
124	5	Aurora	1	1	11	268	16600	12600	Win	29M 31S	
125	6	Jax	4	2	10	240	17700	13100	Win	30M 31S	
126	7	Aatrox	2	6	2	233	14700	10300	Loss	29M 10S	
127	8	Camille	5	2	5	244	21900	14600	Win	35M 53S	
128	9	Jax	3	5	7	284	17700	13700	Win	33M 56S	
129	10	Ambessa	5	1	11	212	23000	11100	Win	27M 37S	

```
=== Hans Sama ===
```

	Game	Champion	Kills	Deaths	Assists	CS	Damage	Gold	Win/Loss	Game Time	Ga
60	1	Ezreal	9	9	8	292	56969	16288	Loss	39M 10S	
61	2	Galio	8	5	18	286	34633	15337	Win	34M 57S	
62	3	Varus	6	11	5	146	16763	9621	Loss	24M 54S	
63	4	Tristana	1	6	10	356	20843	17031	Loss	43M 54S	
64	5	Tristana	5	4	2	147	13329	9033	Win	15M 33S	
65	6	Tristana	3	3	2	176	13172	8336	Loss	20M 34S	
66	7	Zeri	6	7	14	338	34938	17965	Loss	37M 24S	
67	8	Tristana	8	2	8	152	15935	9116	Win	18M 45S	
68	9	Ezreal	5	6	3	144	18020	7859	Loss	21M 03S	
69	10	Varus	8	4	6	290	28834	15246	Loss	33M 05S	
130	1	Ezreal	2	2	5	466	38900	19300	Loss	45M 35S	
131	2	Varus	5	1	4	365	33700	16100	Win	36M 08S	
132	3	Ashe	12	1	7	327	34900	17700	Win	35M 38S	
133	4	Caitilin	4	1	1	271	10800	12900	Win	29M 02S	
134	5	Caitilin	1	3	1	223	10400	9500	Loss	24M 37S	
135	6	Caitilin	0	0	4	256	12500	10400	Win	26M 12S	
136	7	Sivir	5	0	9	279	21300	13400	Win	29M 10S	
137	8	Sivir	2	0	6	325	19600	13300	Win	30M 29S	
138	9	Ezreal	3	2	10	243	26700	11700	Win	27M 57S	
139	10	Ezreal	7	3	11	240	29600	13800	Win	29M 28S	

4. Exploratory Data Analysis (EDA)

In this step, we analyze trends and patterns within the combined dataset using summary statistics and visualizations.

In [64]:
```
#describe the overall data in textual format so we can notice patterns and decid

display(all_games_df.describe())
all_games_df.groupby('MatchType')[['PerformanceScore', 'KDA', 'CS_per_min', 'Dam
```

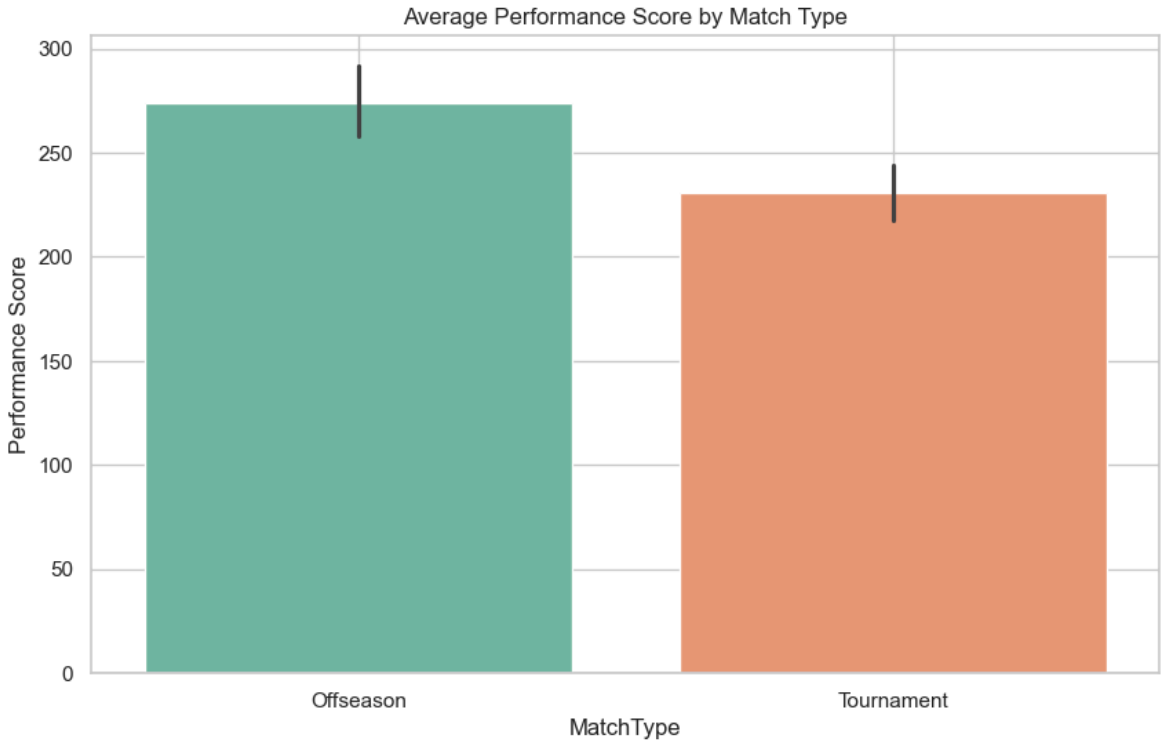
	Game	Kills	Deaths	Assists	CS	Damage	
count	140.000000	140.000000	140.000000	140.000000	140.000000	140.000000	140.0(
mean	5.500000	5.514286	3.628571	7.164286	251.964286	24446.914286	13285.9
std	2.882595	3.678610	2.829299	3.779176	72.575380	11954.958391	3773.0
min	1.000000	0.000000	0.000000	0.000000	95.000000	4641.000000	4603.0(
25%	3.000000	3.000000	1.000000	5.000000	205.750000	16058.750000	10908.0(
50%	5.500000	5.000000	3.000000	7.000000	245.000000	22422.500000	13286.0(
75%	8.000000	8.000000	6.000000	10.000000	290.000000	29725.000000	15211.5(
max	10.000000	18.000000	11.000000	18.000000	539.000000	70810.000000	33500.0(

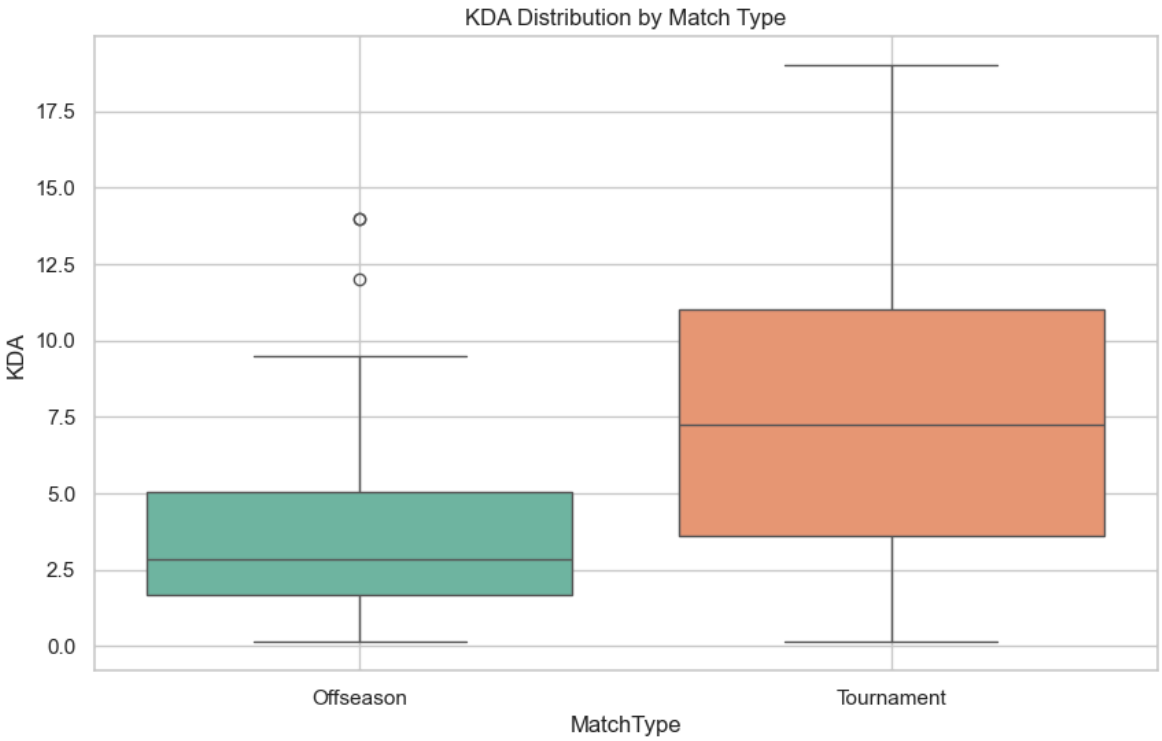
Out[64]:

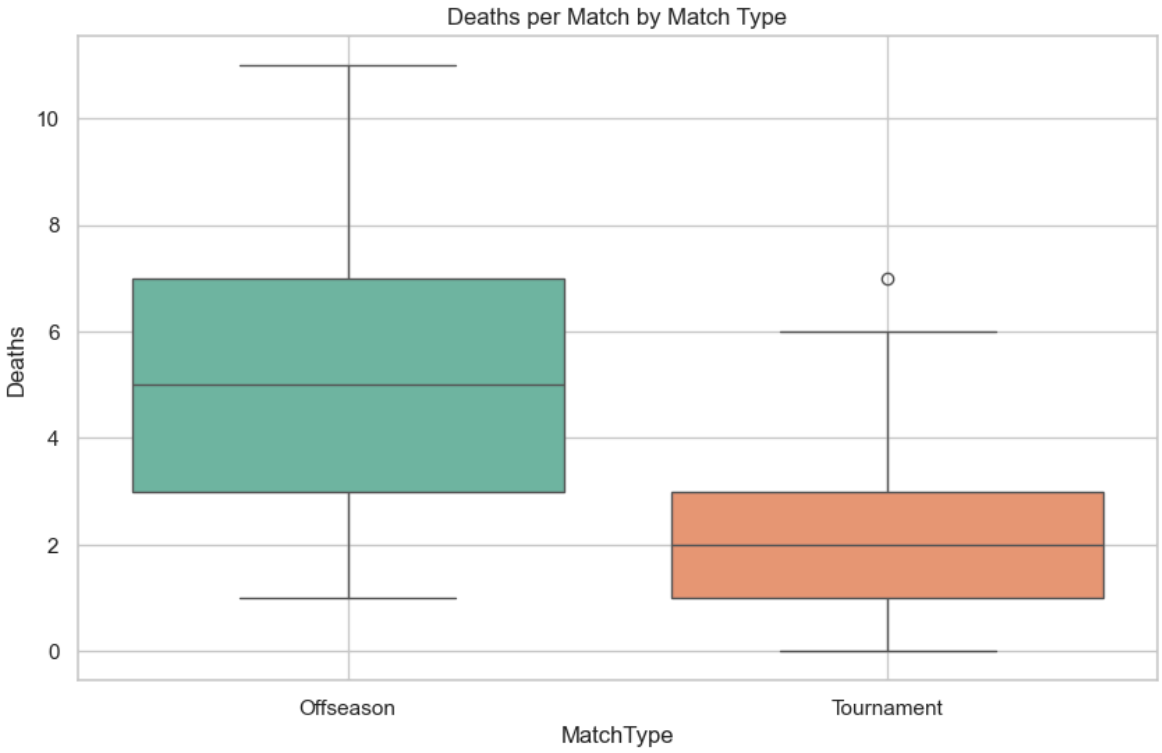
MatchType	PerformanceScore	KDA	CS_per_min	Damage_per_min	Deaths
Offseason	274.073921	3.716995	7.887053	913.422025	5.328571
Tournament	230.528830	7.749660	8.798727	692.952675	1.928571

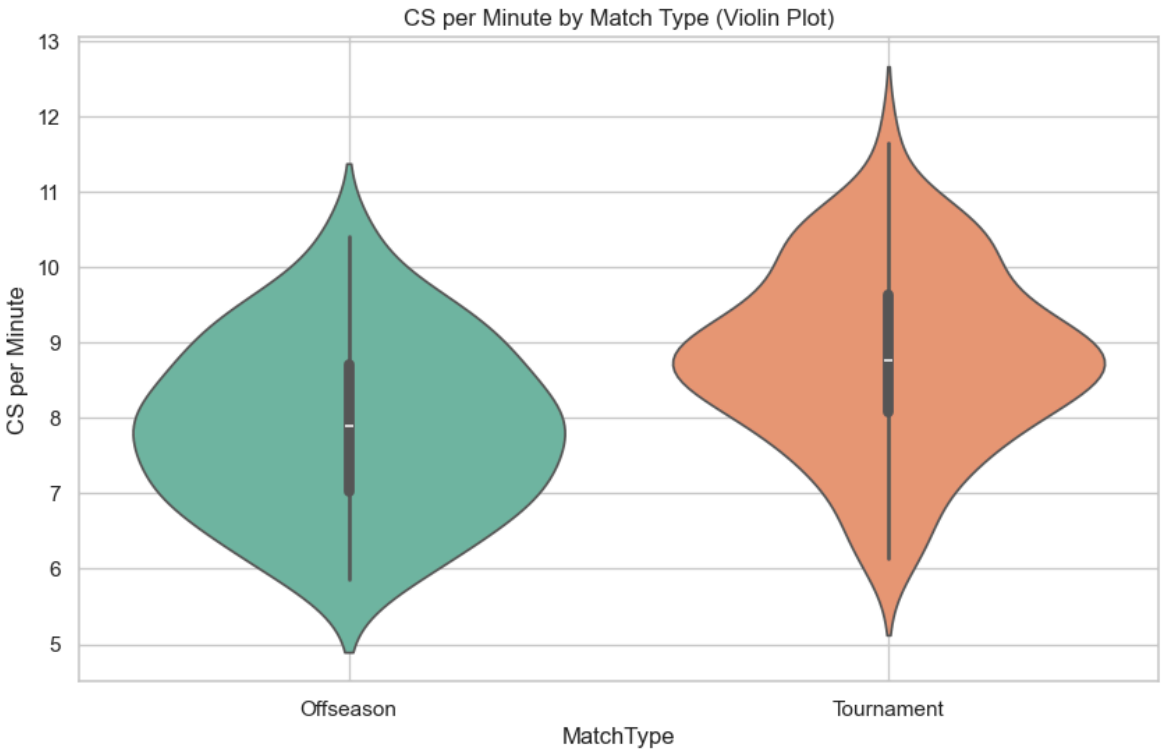
In [65]:
```
# setting the style
sns.set(style="whitegrid")
plt.rcParams["figure.figsize"] = (10, 6)

# Barplot - Average Performance Score by Match Type
plt.figure()
sns.barplot(data=all_games_df, x="MatchType", y="PerformanceScore", hue="MatchTy
plt.title("Average Performance Score by Match Type")
plt.ylabel("Performance Score")
plt.grid(True)
plt.show()

# Boxplot - KDA Distribution by Match Type
plt.figure()
sns.boxplot(data=all_games_df, x="MatchType", y="KDA", hue="MatchType", legend=F
plt.title("KDA Distribution by Match Type")
plt.grid(True)
plt.show()

# Boxplot - Deaths Distribution by Match Type
plt.figure()
```

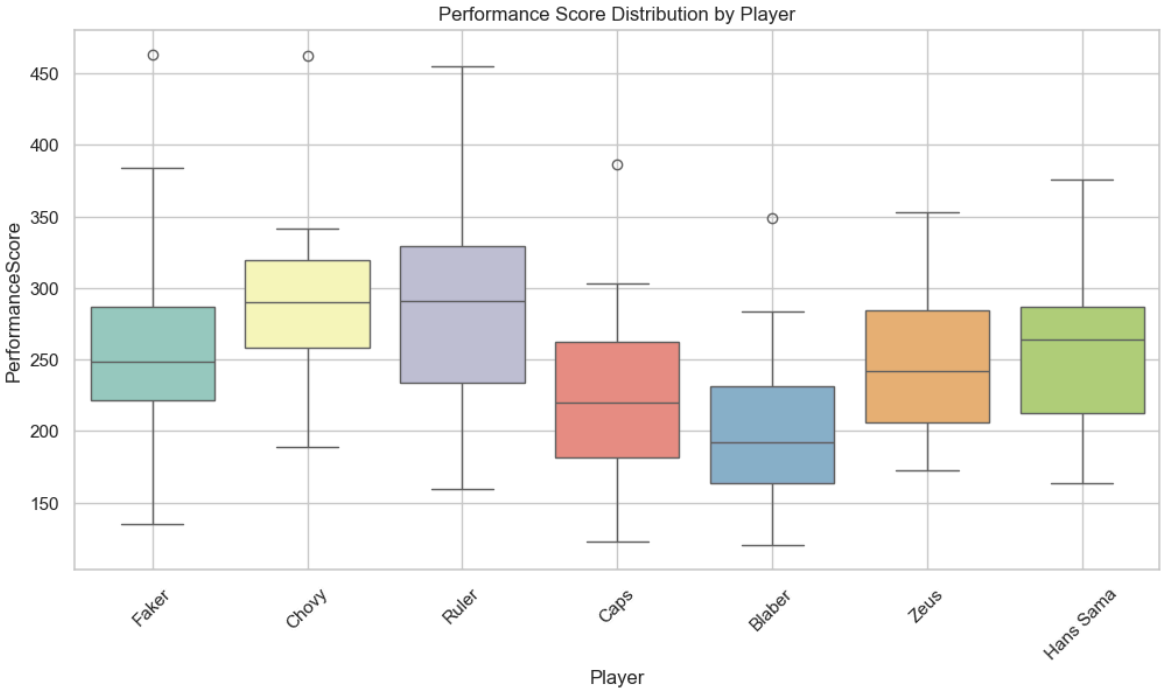
```
sns.boxplot(data=all_games_df, x="MatchType", y="Deaths", hue="MatchType", legen
plt.title("Deaths per Match by Match Type")
plt.grid(True)
plt.show()

# Violinplot - CSPM by Match Type
plt.figure()
sns.violinplot(data=all_games_df, x="MatchType", y="CS_per_min", hue="MatchType"
plt.title("CS per Minute by Match Type (Violin Plot)")
plt.ylabel("CS per Minute")
plt.grid(True)
plt.show()

# Boxplot - Performance Score by Player (combined)
plt.figure(figsize=(12,6))
sns.boxplot(data=all_games_df, x="Player", y="PerformanceScore", hue="Player", l
plt.title("Performance Score Distribution by Player")
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```




KDA Distribution by Match Type
KDA
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Offseason
Tournament
MatchType


Deaths per Match by Match Type
Deaths
10
8
6
4
2
0
Offseason
Tournament
MatchType





EDA Summary

The visualizations above highlight key differences between off-season and tournament matches, as well as variability across individual players:

- Players tend to have **higher performance scores** during off-season games, suggesting more aggressive or less risk-averse play.
- **KDA is higher** during tournaments, likely due to more careful and coordinated strategies.
- The number of **deaths per match is significantly lower** during tournaments, supporting the hypothesis that players adopt safer playstyles under competitive pressure.

- The **CS per minute violin plot** shows that players generally farm slightly more efficiently during tournament matches.
- The **performance score distribution by player** reveals clear variability in performance styles and consistency. Players like **Ruler** and **Chovy** exhibit higher median performance scores compared to others like **Blaber** and **Caps**.

These observed patterns set the stage for formal hypothesis testing in the next section.

5. Hypothesis Testing

Hypothesis 1 – Main Hypothesis: Performance Score Comparison

We aim to test whether there is a significant difference in the average **performance score** between off-season and tournament matches.

- **Null Hypothesis:** There is no significant difference in the average performance score between off-season and tournament games.
- **Alternative Hypothesis:** There is a significant difference in the average performance score between off-season and tournament games.

This is a **two-tailed hypothesis test**, and we will use an **independent samples t-test** to evaluate it, assuming the two groups are independent and drawn from approximately normal distributions.

In [66]:
```
from scipy.stats import ttest_ind

# Separating the performance scores based on match type
offseason_scores = all_games_df[all_games_df['MatchType'] == "Offseason"]['Perfo
tournament_scores = all_games_df[all_games_df['MatchType'] == "Tournament"]['Per

# Perform independent two-sample t-test, no assumption of equal variance
t_stat, p_val = ttest_ind(offseason_scores, tournament_scores, equal_var=False)

# Print the relevant results
print("Mean Performance Score (Offseason):", offseason_scores.mean())
print("Mean Performance Score (Tournament):", tournament_scores.mean())
print("T-statistic:", round(t_stat, 3))
print("P-value:", round(p_val, 5))

# Reject/Accept according to the results
alpha = 0.05  # standard level of significance
if p_val < alpha:
    print("\nResult: Reject the null hypothesis (significant difference exists).
else:
    print("\nResult: Fail to reject the null hypothesis (no significant differen
```

```
Mean Performance Score (Offseason): 274.07392076111137
Mean Performance Score (Tournament): 230.52882990975857
T-statistic: 3.872
P-value: 0.00017

Result: Reject the null hypothesis (significant difference exists).
```

Hypothesis 1 Result: Performance Score Comparison

An independent two-sample t-test was conducted to compare the average **performance scores** of players in **off-season** and **tournament** matches.

- **T-statistic**: 3.872
- **P-value**: 0.00017

Since the p-value is well below the standard significance level of 0.05, we **reject the null hypothesis**. This indicates that there is a **statistically significant difference** in average performance scores between off-season and tournament games.

Interpretation:
Players tend to perform **differently** in off-season matches compared to tournament matches when evaluated using our custom performance metric. This supports the hypothesis that match context (casual vs. competitive) influences overall gameplay behavior.

Hypothesis 2 – Additional Hypothesis: Death Rate Comparison

We aim to test whether there is a significant difference in the average **number of deaths** between off-season and tournament matches.

- **Null Hypothesis:** There is no significant difference in the average number of deaths per game between off-season and tournament matches.
- **Alternative Hypothesis:** There is a significant difference in the average number of deaths per game between off-season and tournament matches.

This is a **two-tailed hypothesis test**, using an **independent samples t-test** to compare the mean number of deaths between the two match types.

In [67]:
```
from scipy.stats import ttest_ind

#  Separating the death values by match type
offseason_deaths = all_games_df[all_games_df['MatchType'] == "Offseason"]['Death
tournament_deaths = all_games_df[all_games_df['MatchType'] == "Tournament"]['Dea

# Perform independent two-sample t-test
t_stat2, p_val2 = ttest_ind(offseason_deaths, tournament_deaths, equal_var=False

# Print the re;evant results
print("Average Deaths (Offseason):", round(offseason_deaths.mean(), 3))
print("Average Deaths (Tournament):", round(tournament_deaths.mean(), 3))
print("T-statistic:", round(t_stat2, 3))
print("P-value:", round(p_val2, 5))


# Reject/Accept according to the results
alpha = 0.05
if p_val2 < alpha:
    print("\nResult: Reject the null hypothesis (significant difference exists i
```

```
else:
    print("\nResult: Fail to reject the null hypothesis (no significant differen
```

```
Average Deaths (Offseason): 5.329
Average Deaths (Tournament): 1.929
T-statistic: 8.88
P-value: 0.0

Result: Reject the null hypothesis (significant difference exists in death rate).
```

Hypothesis 2 Result: Death Rate Comparison

To determine whether players die significantly less during tournament matches, we performed an independent two-sample t-test comparing the average number of deaths per game in off-season and tournament matches.

- **Average Deaths (Offseason):** 5.329
- **Average Deaths (Tournament):** 1.929
- **T-statistic:** 8.88
- **P-value:** < 0.00001

Since the p-value is well below the 0.05 significance threshold, we **reject the null hypothesis**. This confirms that there is a **statistically significant difference in death rates** between match types.

Interpretation:
Players die significantly less often in **tournament matches** compared to **off-season games**. This supports the idea that in competitive settings, players adopt a more careful and strategic playstyle to reduce risks.

Hypothesis 3 – Additional Hypothesis: CS per Minute Comparison

We aim to test whether there is a significant difference in the average **CS per minute (Creep Score per Minute)** between off-season and tournament matches.

- **Null Hypothesis:** There is no significant difference in the average CS per minute between off-season and tournament matches.
- **Alternative Hypothesis:** There is a significant difference in the average CS per minute between off-season and tournament matches.

This is a **two-tailed hypothesis test**, using an independent samples t-test to compare average CS/min between the two match types.

In [68]:
```
from scipy.stats import ttest_ind

# Separating the CS/min values by match type
offseason_cs = all_games_df[all_games_df['MatchType'] == "Offseason"]['CS_per_mi
tournament_cs = all_games_df[all_games_df['MatchType'] == "Tournament"]['CS_per_

# Run the t-test
t_stat3, p_val3 = ttest_ind(offseason_cs, tournament_cs, equal_var=False)
```

```
#print the mean of both offseason and tournament cs value
print("Average CS/min (Offseason):", round(offseason_cs.mean(), 3))
print("Average CS/min (Tournament):", round(tournament_cs.mean(), 3))
# Output the t-test results
print("\nT-statistic:", round(t_stat3, 3))
print("P-value:", round(p_val3, 5))

# Interpret the result
alpha = 0.05
if p_val3 < alpha:
    print("\nResult: Reject the null hypothesis (significant difference exists i
else:
    print("\nResult: Fail to reject the null hypothesis (no significant differen
```

```
Average CS/min (Offseason): 7.887
Average CS/min (Tournament): 8.799

T-statistic: -4.679
P-value: 1e-05

Result: Reject the null hypothesis (significant difference exists in CS per minut
e).
```

Hypothesis 3 Result: CS per Minute Comparison

An independent two-sample t-test was conducted to compare the average **CS per minute** between off-season and tournament matches.

- **Average CS/min (Offseason):** 7.887
- **Average CS/min (Tournament):** 8.799
- **T-statistic:** -4.679
- **P-value:** 1e-05

Since the p-value is far below the 0.05 significance threshold, we **reject the null hypothesis**. This indicates that there is a **statistically significant difference in CS per minute** between off-season and tournament games.

Interpretation:
Players farm **more efficiently during tournament matches**. This may reflect more focused laning strategies, better coordination with team objectives, or simply higher discipline and prioritization of gold income during official competitive play.

Conclusion

This project analyzed the in-game performance of seven professional League of Legends players across 140 matches, split evenly between off-season and tournament games. Using a custom-built performance score and other key gameplay metrics such as KDA, deaths, and CS per minute, the goal was to uncover statistical differences in player behavior across different match contexts.

Three hypotheses were tested:

- **Hypothesis 1:** Players have significantly different overall performance scores in off-season vs tournament matches. *Supported*
- **Hypothesis 2:** Players die significantly less in tournament matches. *Strongly supported*
- **Hypothesis 3:** Players have significantly different CS per minute between match types. *Supported*

These results highlight a measurable shift in playstyle between casual and competitive environments. Tournament matches are characterized by safer, more efficient gameplay — with higher KDA, fewer deaths, and better farming — suggesting that pro players adapt their strategies significantly in high-stakes settings.

Overall, the project demonstrates how structured data analysis can reveal deeper trends in esports performance, and serves as a model for applying hypothesis-driven data science in competitive gaming contexts.