2025 The International Mathematical Modelling Challenge (IMMC)

Global Sports League

1. Summary

The International Multi-Continental Matchmaking Committee (IMMC) aims to develop a schedule for the Global Super League (GSL) and has tasked us to choose the sport and develop a mathematical model which will make the schedule. We chose *Cricket* as it is highly popular in areas such South-East Asia, Australia, and England but has yet to establish a foothold in areas such as North America and especially South America, which shows potential for global expansion.

We then choose twenty teams based on ICC T20 rankings to participate in the tournament, ensuring that each continent has representation in the tournament. We also found the factors that influence the making of a league schedule and divide them into venue related and tournament related.

Furthermore, we took a two-pronged approach to construct our model, first using piecewise functions and the Analytical Hierarchy Process (AHP) to decide the best venue which we found to be UAE and then we wrote code in python which allowed us to make groups with the most fair matchups by considering the variance in the ratings of the teams present in the group

We then expanded the GSL by adding four new teams: UAE, Spain, Uganda and Mexico based on their ratings and recent T20 forms. We found out that even with the addition teams, the league schedule was not impacted greatly on the fronts of fairness, sustainability and geographic representation but, the number of games changed.

Then, we made changes to some key constraints to see how the output of the model was impacted before we made a graphic to represent the schedule in a user-friendly way.

2. Introduction

2.1 Background

Scheduling. It is an aspect of sports that is often just a footnote for the spectators, often being cast aside by the anticipation of the thrilling clash of teams putting their skills on show in order to emerge triumphant.

Sports and scheduling have gone hand-in-hand from ancient times, with ancient Romans scheduling their annual *Roman Games* around religious festivals. [1] By the 19th century, structured scheduling became more prominent, with it being implemented in organised leagues such Major League Baseball. [2]

With the rise of the television (TV) in the mid-20th century, scheduling evening games such as NFL's "Monday Night Football," came to the fray. [3] Though at the time, this seemed like the apex of sports scheduling, that has yet to come as in the present, leagues such as the NFL have started using computer algorithms to determine the most suitable schedule. [4]

2.2 Problem restatement

The International Multi-Continental Matchmaking Committee (IMMC) aims to develop an effective schedule for a global sports event known as the *Global Sports League (GSL)*, akin to the *Cricket World Cups* hosted by the International Cricket Council (ICC).

They task us to firstly, choose a team sport which has potential to gain global recognition via GSL, expanding the sport's fanbase into previously untapped regions. We also have to select 20 teams in total to participate in this event and, to ensure that the tournament lives up to its name, choose at least 2 teams from every continent except Antarctica. Finally, we have to consider the factors that we need to consider in order to make a schedule that will be suitable and impartial to all teams so that it is a controlled variable.

Secondly, we have to forge a mathematical model that would create an efficient schedule, spanning over eight to nine months, for the GSL, ensuring that each team participates in a similar number of matches and that there are sufficient rest periods. Moreover, the durations and distances each team covers to arrive at the venue must be similar, so that all teams face similar logistical issues. The model must also guarantee that each match allows for a level playing field between teams by making teams face a mix of strong, moderate, and weak opponents to avoid unfairly easy or difficult matchups. To cap it off, the model must be built according to IMMC's principles of eco-consciousness, diversity, and inclusion while still ensuring that it's economically viable and can generate income.

Task three asks us to expand our model by incorporating four additional teams to the GSL. Does this still fulfil the IMMC's specific standards? By how much does this change the number of matches each team participates in? Does that still accurately address the limitations given? The model should be able to cater all these changes while still retaining all its credibility.

Next, we are to prove the model's flexibility by applying its methodology to any team sport. It should skilfully incorporate any team sport's specific regulations and a generate a suitable schedule.

Lastly, we must simplify and explain our entire methodology in a letter so that it's easily understandable to a reader who's unfamiliar with all the professional jargon. That is, to the International Multi-Continental Matchmaking Committee, so they can make an apt and well-informed decision regarding the schedule for the Global Super League. We are to, also, attach a illustration of the schedule so it's easy to catch on with.

3. Step One: Cricket meets the Global Super League

Cricket boasts an international fan-base of 2.5 billion people, which is approximately 31% of the world's population . [5] 90% of these fans reside in countries such as Pakistan, India, Sri Lanka and Bangladesh while regions such as the United Kingdom, Australia, New Zealand, the Caribbean Islands and South Africa also share a deep interest in cricket.

However, there is still room for global expansion as sports such as Football have a massive global fanbase of 5 billion people. [6] Attempts were made to expand cricket to North America, with the ICC World T20 2024 being hosted by the United States of America and saw some success as 2.7 million U.S.-based users visited the ICC website and app during the tournament, showing potential interest of new fans in the sport.

Cricket has three formats: test cricket which lasts five days, one-day-international (ODI) which as the name suggests, lasts approximately one day and finally, twenty-twenty (T20) cricket which goes on for three or four hours. For the purposes of GSL, we will choose the T20 format of cricket as it is fast-paced, ensuring new fans will not loose interest and does not last long, making scheduling easier while ensuring players do not get fatigued.

3.1 Choosing teams

We decided to choose teams using the ICC's Men's T20 Team Rankings, [7] which are calculated using the formula:

$$R = \frac{total\ points\ acquired\ by\ a\ team}{total\ number\ of\ matches\ played\ by\ the\ team}$$

where R is the overall rating of the team. [8] After a match, the winning team will gain points equivalent to the sum of the opposing team's ranking and fifty (50), while the losing team will gain points equivalent to the sum of the opposing team's ranking and minus fifty (-50).

This rating system is perfectly aligned by IMMC's goal of fairness as it is based on the match outcomes in the past four years, while also ensuring the current form or performance of a team is prioritised.

For example, if a team has played in the period 2012-2016, then the points earned in the first two years (2012-2014) will have 50% weightage and will be halved, while the points earned in the next two years (2014-2016) will have 100% weightage.

From Asia, we decided to choose the most number of teams (five) as it has a large fanbase in countries such as India and also has numerous experienced T20 teams which would make the tournament highly competitive.

Europe follows suit with four teams chosen due to having a strong domestic cricket circuit in countries such as the UK which would unearth mounds of cricket talent.

Oceania and North America would have three teams each and even though the latter does not have a strong fanbase, the rise in cricket interest in the USA due to the ICC World T20 2024 would likely trigger a domino effect in the continent.

Lastly, South America would have the least number of teams (two) as there are no full-member nations* in the continent and cricket does not have a foothold there.

Continent	Countries Chosen
Asia	Pakistan India Sri Lanka Bangladesh Afghanistan
Europe	England Ireland Scotland

	Netherlands
Africa	South Africa Zimbabwe Namibia
Oceania	NZ AUS Papua New Guinea
North America	Canada Usa West Indies
South America	Argentina Brazil

3.2 Aspects needed to be considered when constructing a cricket league schedule

Travel distance, number of games played and equitable matchups are some general factors that are considered when constructing the tournament schedule of any sport. However, when it comes to cricket, there are some specific factors that need to be considered which we divide into venue related and tournament related factors:

Venue related

- **Past experience:** the country hosting the tournament should ideally have hosted a major global tournament before to effectively cope with the logistical issues an event of this scale brings. This is illustrated by the fact that in 2010, when India for the first time hosted the Commonwealth Games, there was a degree of mismanagement as there were delays in construction of the main games' venues resulting in negative connotations being attached to the event by the public. [9]
- **Cricket fanbase:** there should be a large number of cricket enthusiasts near the vicinity of the venues to ensure a large attendance in stadiums on matchday, ensuring a large amount of profit is earned from ticket sales.
- **Number of stadiums:** a large number of stadiums ensures that multiple games can run simultaneously and ensures that teams get challenged by new conditions offered by a new venue, while also tapping into the fanbase of that venue, keeping the game thrilling.
- **Political tensions:** the country hosting the tournament should have minimal political tensions with the participating countries, if at all. Or else, some participating may refuse to travel to the host country, as was the case with India refusing to travel to Pakistan for the 2025 ICC Champions Trophy which led to a long deadlock between the ICC, Pakistan and India until finally it was settled that India could play its matches in a neutral venue (United Arab Emirates). [12]
- **Weather:** the venue should have suitable weather, to avoid rain delays during matches or complete washouts which can ruin fan experience while also preventing a fair display of each teams' skills.

Tournament related

- **Rest periods between games:** a sufficient break between consecutive games ensures that players do not get injured and fatigued. For example, the English Cricket Board (ECB) recommends fast bowlers to have at least one day rest between bowling sessions. [10] In addition, playing in the same venue reduces time for curators to prepare a suitable pitch, which may impact the result of the game significantly.

- **Broadcast and commercial considerations:** highly anticipated games are often scheduled in time slots in which both the team's supporters are available to spectate the game such as after work hours or on the weekends to maximise the reach of the game, increasing profits.

- **Clashes with official tournaments and franchise leagues:** a global tournament which lasts eight to nine months ensures that clashes with other cricket tournaments become inevitable. With the ICC and franchise cricket leagues such as the Indian Premier League (IPL) already having a firm foothold, teams and players may be occupied with these commitments. Franchise leagues especially offer players lucrative incentives, so much so that the ECB barred its players to participate in franchise leagues (except the IPL). [11]

4. Step Two: Constructing the mathematical model

4.1 Choosing the venue

4.1.1 Quantifying variables

Our mathematical model will take a two-pronged approach for deciding the schedule. Firstly, to settle on a venue, we would let *P, T,* and *D* denote past experience, political tensions and travel distance respectively. We also combine cricket fanbase and number of stadiums into a new variable: fan turnout, denoted by *F*. We do not consider weather and assume that it will suitable on math day as it is difficult to predict and quantify.

To determine if a venue is ready to host we will calculate the preparedness score, *PS*, of each venue, where *PS* is calculated by:

$$PS = \Sigma\ (W_i\ \times FS_i)$$

where W_i is the weight of the factor *i* and FS_i is the score obtained by the venue for factor *i.*

We consider *P* to be a *Boolean* variable which can be easily scored by the following piecewise function:

$$FS_P = \begin{cases} 1 & if\ yes \\ 0 & if\ no \end{cases}$$

To quantify the remaining variables, we would use linear piecewise functions as they require minimal computational power and are a simple but efficient way to quantify non-linear variables.

For *F*, we compute its score using:

$$FS_F = \begin{cases} 0 & if\ P = 0\ or\ S = 0 \\ 0.3\left(\frac{S}{P}\right) + 0.7A & if\ 0 < \frac{S}{P} \leq 1 \\ 0.3 + 0.7A & if\ \ \frac{S}{P} > 1 \end{cases}$$

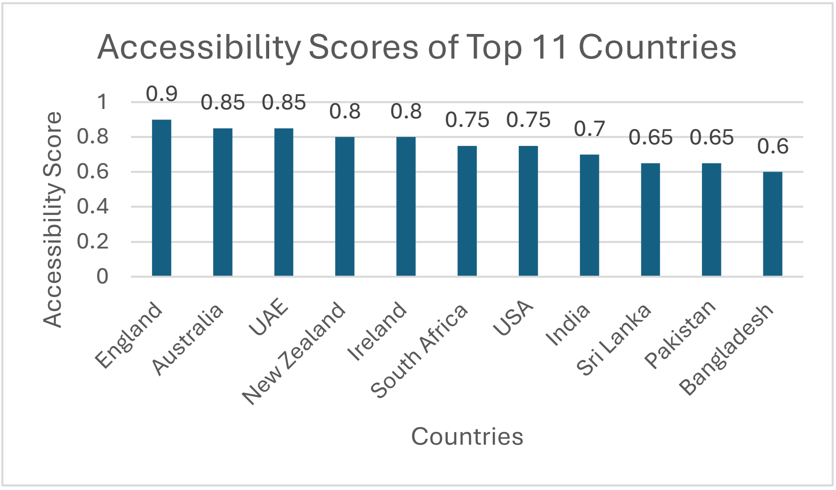
Where P is the population of cricket fans (in millions), S is the total stadium capacity across all cricket stadiums in the country (in thousands) and A is the accessibility factor between 0 and 1 where 1 means excellent accessibility and 0 means poor accessibility.

We assigned a higher weight (0.7) to accessibility than stadium capacity, $\frac{S}{P}$ (0.3) as stadiums which have poor accessibility due to inadequate transport links among other factors, struggle to seamlessly host events even though they might have a high capacity. For example, the Saitama Stadium in Japan has a high capacity of 63,700. However, its poor accessibility is highlighted by the fact that during the 2026 FIFA World Cup qualifiers, the Australian team took 2 hours to reach the stadium, even though it was estimated to be a 40 minute commute, hindering athlete performance. [13]

To deduce the accessibility factor, we used relevant sources [14] to judge each stadium based on public transport, traffic and parking, infrastructure and disabled access. We also preferred stadiums with more links to public transport especially as it aligns with IMMC goals of environmental sustainability by reducing greenhouse gas emissions into the atmosphere.

To make this process easier, we made the following assumptions:

Assumptions – Accessibility Score
• Stadiums located in urban centres are more accessible than those in rural areas as urban centres have relatively better transport links (roads, bridges, trams, busses etc.)
• Countries with well-developed public transport (e.g. England, Australia, New Zealand) score higher in accessibility as public transport is environmentally friendly and reduces reliance of private vehicles
• Stadiums with ample parking facilities score higher, but reliance on parking alone reduces the score as it suggests poor public transport links and more air pollution
• Countries with higher traffic congestion (e.g. Pakistan, India, Bangladesh) score lower in accessibility as travel times are increased



Top 11 highest accessibility scores

To score travel distance, D, we use the following piecewise function:

$$FS_D = \begin{cases} 1 & if\ \sigma^2 = \sigma^2 min \\ \dfrac{\sigma^2 max - \sigma^2}{\sigma^2 max - \sigma^2 min} & if\ \sigma^2 min < \sigma^2 < \sigma^2 max \\ 0 & if\ \sigma^2 = \sigma^2 max \end{cases}$$

Where σ^2 is the variance of travel distances for a given country, $\sigma^2 min$ is the minimum variance observed across all countries and $\sigma^2 max$ is the maximum variance observed across all countries. The lower the variance in travel distances to the venue country, the better as it shows equitable travel distances.

Assumption – Travel Distance
We assume that there is only variation in the travel distances from the country to the venue and the travel distances within the host countries are similar for all teams

If the host country has political tensions with other nations in the tournament, then the following function will apply:

$$FS_T = \begin{cases} 0 & if\ no\ political\ tensions \\ -\infty & if\ political\ tensions\ exist \end{cases}$$

This way we can eliminate those countries from hosting.

4.1.2 Determining the weights

We will use the Analytical Hierarchy Process (AHP) to determine the weights of each factor. AHP is a decision-making method based on pairwise comparisons. It is a type of multi-criteria decision analysis (MCDA).

Firstly, we determine the hierarchy structure with the top level having the goal (choosing the best venue) and the middle level having the criteria (the factors we considered above).

Secondly, we make the pairwise comparison matrix to compare each factor to the other factors based on relative importance. We do not include political tensions as it is only a means of elimination. Note that the columns are Past Experience (*P*), Fan Turnout (*F*) and Travel Distance (*D*).

$$\omega = \begin{vmatrix} 1 & 1/4 & 1/3 \\ 4 & 1 & 2 \\ 3 & 1/2 & 1 \end{vmatrix}$$

where ω is the pairwise comparison matrix.

We determined that fan turnout is more important that past experience (4 times as important) as the country may have built new stadiums since it last hosted a world T20 tournament. Secondly, travel distance important than past experience as the country may have faced logistical issues due to travel distance in the past. Finally, fanbase is slightly more important than travel distance (2 times as important) as logistical costs are offset by greater fan turnout.

Next, we find the principal eigenvalue and its corresponding eigenvector. The principal eigenvalue (λ_{max}) is 3.0183 while its associated eigenvector is [0.1862,0.8527,0.4881].

Then, we normalise the eigenvector to find the weights of each factor. After normalisation, the eigenvector became [0.1220,0.5584,0.3196]

To check if the decision-making of our matrix is valid, we calculate the consistency index, CI, where CI = $\frac{\lambda_{max}-n}{n-1}$ where n is the order of the matrix. This was found to be 0.0091. Then, we calculate the consistency ratio, CR where CR = $\frac{CI}{RI}$. RI is the random index and is predefined for every value of n.

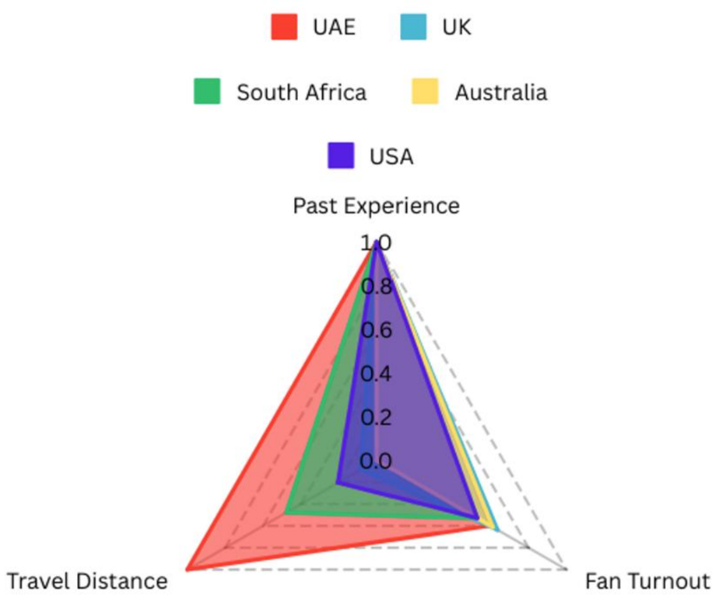
Size of matrix (n)	1	2	3	4	5	6	7	8
Random Index	0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41

Random Index values for n=1 up to n=8

For our matrix, CR comes to about 0.0158 which is less than 0.1 meaning the judgment of our matrix is acceptable.

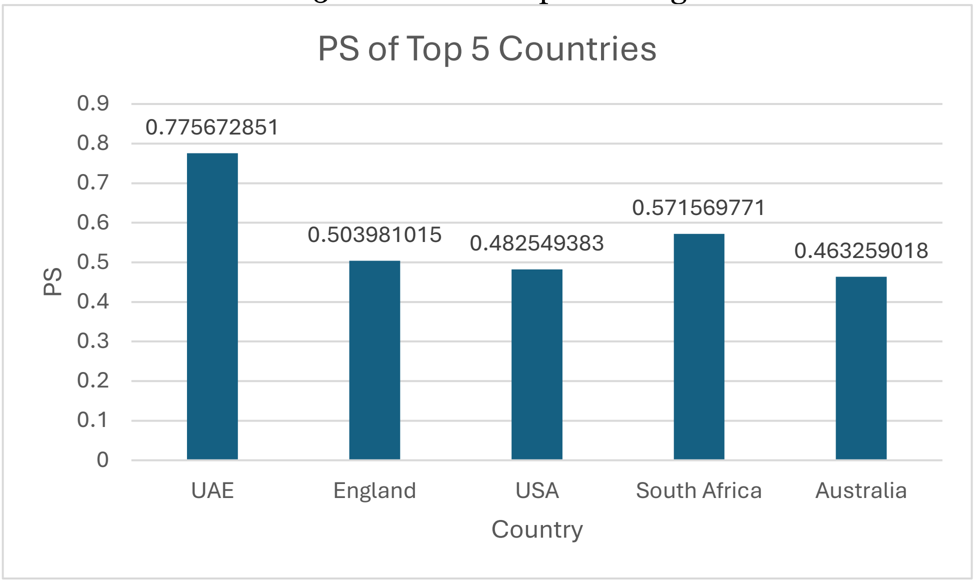
4.1.3 Results

Pakistan and India were eliminated due to political tensions highlighted by the Champions Trophy 2025. The following radar chart shows the scores achieved for each factor by the top five countries (in terms of PS):



Radar chart showing UAE, England, South Africa and Australia with their corresponding scores for travel distance, past experience and fan turnout

In terms of *PS* (preparedness score), UAE got the highest overall score of 0.776, showing that it is the best choice, considering all the factors, for the venue of a global cricket tournament. Australia has the lowest score of 0.463, mostly because it is not a suitable venue for keeping travel distances between participating teams similar.



Bar chart showing the preparedness score, PS of top five countries

4.2 Making the schedule

The code our methodology is based on is simple yet it gives fruitful outputs. We made a dictionary with all twenty of the selected teams in it. Based on the ratings allocated to each team due to their performances in previous T20 matches, they are then sorted into one of two lists: high rated teams and low rated teams, Teams with a higher rating than the mean rating are sorted into the former, while teams with a lower rating than the mean go into the latter list. This is a step towards ensuring balanced competitiveness between teams.

```
for team, score in teams.items():
if score > mean_score:
high_rated.append(team)
else:
low_rated.append(team)
```

Our idea was to divide the teams into groups of five, each group containing two teams from the previously made high rated list, and two from the low rated list. To further ensure that one group doesn't overpower the other, we introduced a new variable—variance. The code calculates the total score for each of the five groups, which is the sum of the ratings of each team in that group. The mean score of all five groups is then calculated. The variance, which is calculated using the above mentioned data, should be as low as possible to ensure that one group isn't significantly more skilled than the other. The code then continues making these groups—five groups of four teams—over a thousand iterations, to choose the best possible combination. That is, one which ensures fair matchups.

```
if variance < best_variance:
best_variance = variance
best_groups = groups
```

Now that the groups are made, the code moves on to making the schedule. Our design enables each team in a group to play all other teams in that group once, ensuring each participating team plays an equal number of matches. For this we simply used nested loops. It also assigns a date to each match. Our code is written in such a way that no more than one match can be played in a day, and there's a constant rest day between each game that is played. To best suit the duration of the GSL, we chose one day. We also defined a start and end date in accordance with the League duration.

Another factor we introduced into our code was preferred matchups. This was to increase the economic stability of the GSL. As in all other sports, cricket has certain team pairings that ensure high viewership, television ratings and attendance. Through our research and personal knowledge of the sport, we identified pairings which were likely to bring in more monetary benefits than others, and incorporated that into our selection of teams for the same group. India and Pakistan, for example, must be in the same group. This ensures that they play against each other early into the GSL, which creates hype for the League, and ensures longevity in viewership.

```
preferred = [ ("India", "Pakistan"), ("Australia", "New Zealand"),
("Sri Lanka", "Bangladesh")]
```

The next round of matches, the *Super Tens*, are matches played between the top ten teams from the previous round of matches. Since these ten teams are purely based on the points earned during the first round, we chose placeholder names instead of the country names moving forward. Using the same logic as before, each of the ten teams play the other team once.

Finally, the top four of the Super Tens progress into the playoffs. Here the code gets really simple. It outputs the top two teams as round one of the qualifier, and the bottom two as round one of the eliminator. Then, it outputs the winner of the eliminator and the loser of the qualifier together in a third match. Then it outputs the final match, which is between the winner of the first qualifier and the winner of the second eliminator.

5. Step Three: Expanding the GSL

5.1 Adding four new teams

To expand the tournament to newer frontiers and tap into newer audiences, we decided to add the following teams:

- UAE (Asia): the highest rated associate team in Asia, with a rating of 179 in T20s. We decided to add this team as in our initial 20-team schedule there were no associate nations from Asia and hence, adding UAE will give them representation. Also, UAE have been improving in T20 cricket, with them beating New Zealand, a full member by 7 wickets in August of 2023, which was their first victory against a full member in T20s. [15]
- Uganda (Africa): it is the second highest associate nation in Africa in terms of rating (136), only being behind Namibia in this regard. We have chosen them as they made history qualifying for the 2024 T20 World Cup and beat PNG in their group stage matches, even though they could not progress further into the tournament.
- Spain (Europe): Spain has a rating 111 in ICC T20 rankings, having an impressive win to lose ratio. In August 2024, Spain set a new record for consecutive T20I victories, achieving their 14th straight win and surpassing the previous record of 13 set by Malaysia in 2022, which shows how much they have progressed in T20 cricket. [16]
- Mexico (North America): having a relatively low rating of 13, Mexico is still struggling to establish a foothold in T20 cricket. However, Mexico secured a convincing 8-wicket win

over Suriname, successfully chasing down a modest target of 49 runs in December of 2024 which shows their potential. Including them will also further bring cricket to the forefront in the Americas hence aiding in global expansion.

5.2 How this expansion impacts the league schedule

In terms of fairness, it is not greatly impacted since the system still divides the 24 teams into 6 groups of 4 and choses the groups with the lowest variance while still ensuring there are two low rated and two high rated teams in a group so that the matchups are fair and there are no groups of death. However, since some teams, such as Mexico have been added which have not faced full members frequently in matches, there may be some difficulty for these teams in the group stage, but this is necessary for the global expansion of the tournament.

For sustainability, even with the addition of these teams, variance in the travel distances to UAE (the venue chosen) remains largely the same and since these countries have not hosted global T20 cricket tournaments before, they cannot be chosen as venues. Also, since UAE remains the venue of the tournament, environmental sustainability is ensured as UAE's stadiums have good public transport connections which lower the reliance on private vehicles, reducing greenhouse gas emissions.

Lastly, for geographic representation, with the addition of these four teams, South America once again has the least geographic representation with two teams. However, there are not a lot of cricket playing nations in South America which have established a foothold in T20 cricket and hence, even if we tried to increase their representation, it would come at an expense of equitable matchups. Furthermore, Asia will once again have the most geographic representation with six teams which is fair as it has the largest population of cricket fans and the most skilful cricket nation. Oceania also, did not get additional teams as there are not a lot of nations which already have a foundation of T20 cricket.

5.3 Impact on number of games

We extended the model such that it would accept an input of twenty four teams and divide them into six groups of four teams. Instead of a super tens phase, there is now a super twelves phase since the top two teams from each of the six groups qualify for this phase resulting in twelve teams. We also made it so that if the number of teams entered is not a multiple of four then it will require you to either remove a team or add a team based on how many are needed to add or subtract.

When we ran the updated model, we obtained 107 total matches played compared to 80 matches when there were 20 teams.

5.4 The impact of changing key constraints on the model

Venue:

- Past experience: if we increase the weight of past experience, countries which have hosted T20 world cups before (such as England, Australia, India) are favoured more. However, if we decrease the weight, associate nations such as USA would have a chance to host, leading to global expansion.
- Fan turnout: if we increase the weight, countries with high cricket fanbases such as India and Pakistan would be favoured boosting sales and engagement. However, if it decreases

emerging cricket markets such as Canada can be preferred which can lead to the expansion of the cricket fanbase.

- Travel distance: if we increase its weight, centrally located countries such as UAE would be favoured, however if we decrease its weight, countries such as Australia and New Zealand will have a chance to host.
- Political tensions: relaxing the political tensions piecewise function such that it does not eliminate the country but lowers its score would allow countries such as Pakistan and India to host.

Tournament:

- **Preferred matchups:** if the number of preferred matchups is increased, unbalanced groups may start forming, resulting in groups of death where low rated teams have little chance of progressing further in the tournament.
- **Number of gap days between matches:** 1-2 days between each match is ideal for player recovery while ensuring fairness. However, 3 days and above may increase tournament duration and make it longer than 8-9 months. No rest days can lead to player injuries and shorten the duration of the tournament so its less than 8-9 months.#
- **Number of groups:** increasing group size means there are less groups and there are more group stage matches. Decreasing group size results in more groups resulting in a shorter group stage meaning that some teams may qualify for the next stage while only playing a few matches.
- **Changing playoffs status:** if playoffs are set to true, the top two teams are given two chances to qualify for the final which is more fair as they showed greater skill than the other teams. However, if playoffs are set to false, the tournament follows the usual semifinals followed by finals schedule which results in a lower number of games played.

6. Criticism

Our model does not account for the time zones of the teams playing, thus hindering commercial considerations. Efficient coding techniques can be used and we have used minimal functions.

However, our venue choosing model has accurate decision making as supported by the consistency index calculations. It also requires minimal computing power.

7. Letter and graphics

Dear Directors,

It's safe to say that we've developed a scheduling model that fits your every requirement.

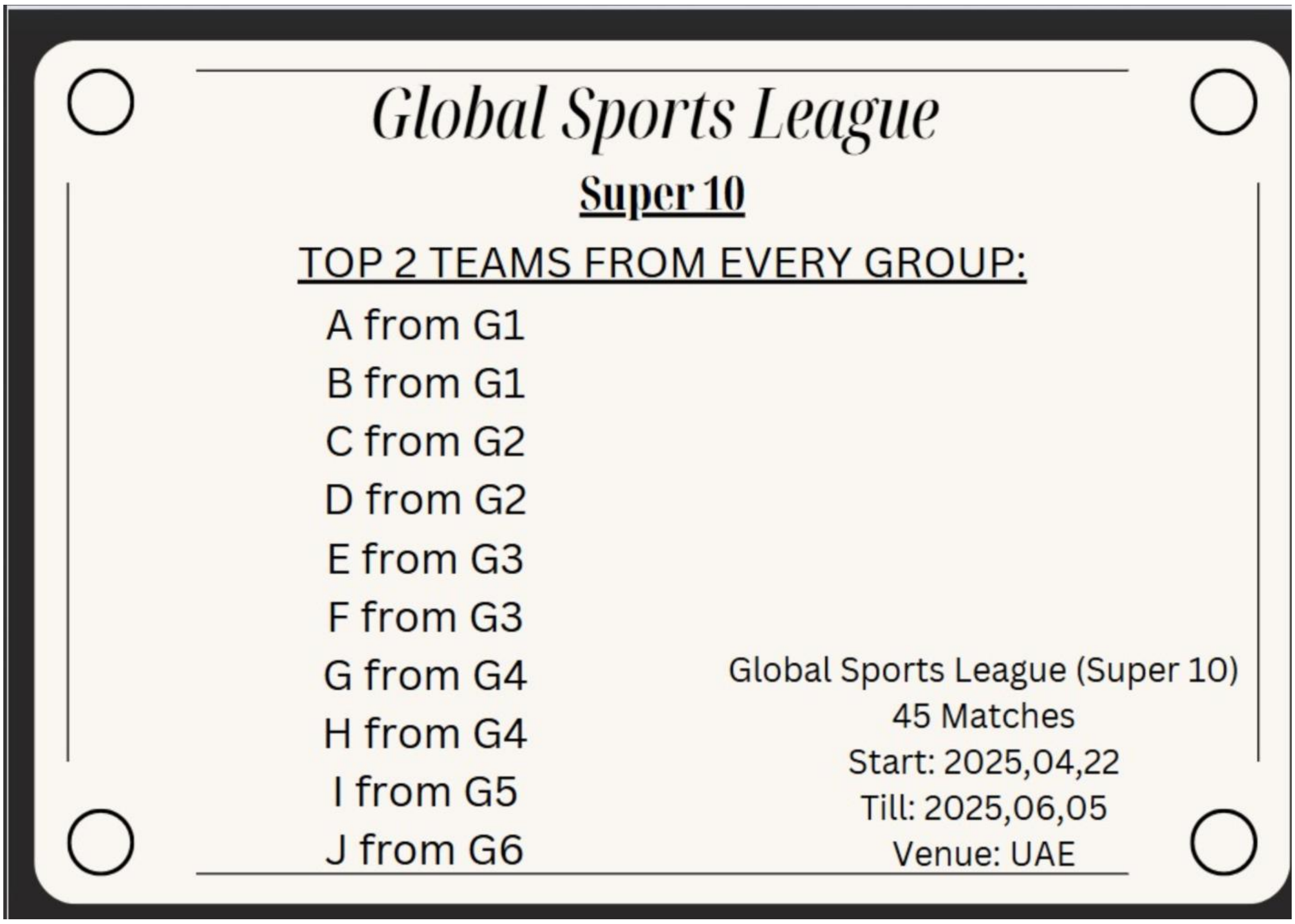
Don't believe us? Here's some key features our model considers which makes it your number one choice for not just the Global Super League's schedule, but for any and every team sport event you choose to hold in the future. Remember, we are with you every step of the way.

1. It will bring in all the money. Since our model takes into account highly anticipated matches, you are sure to start off the tournament well. After all, no one's going to miss a Pak vs. Ind match. People will fly over from all over the world. Which brings us to our next point.

2. Our model selects the most suitable venue for the League for you. Since people will already be at the chosen venue at the beginning of the GSL season, they will most likely stay to attend future matches, generating more revenue. This does not happen often since matches are usually all over the world. Which, once again, brings us to the next point.
3. Environmental sustainability! We love it. Our model evaluates data from a set of potential venues, and choses one which releases the least carbon emissions due to teams travelling from all over the world. Being eco-friendly isn't just a trend you'll be following, you'll be actively contributing towards bettering our planet.
4. Our model also ensures no team is left out. Since each team plays an equal number of matches in the group stage, you'll be sure to hear no complaints from any team representative about unfair advantages. Each team progresses through the tournament fair and square.
5. The model ensures cricket becomes a world famous sport. Since we've incorporated teams from all the world—from every continent— it is sure to attract a new audience to the sport. They will know cricket because they saw it in the Global Super League, making it a classic for the sport. It will be the name on every tongue.

We've also attached a generalized version for our cricket scheduling model. It keeps all your key concerns in mind, while still generating an accurate and appropriate schedule. Whether it's football or *cheese rolling*, our model will work for you.

Best



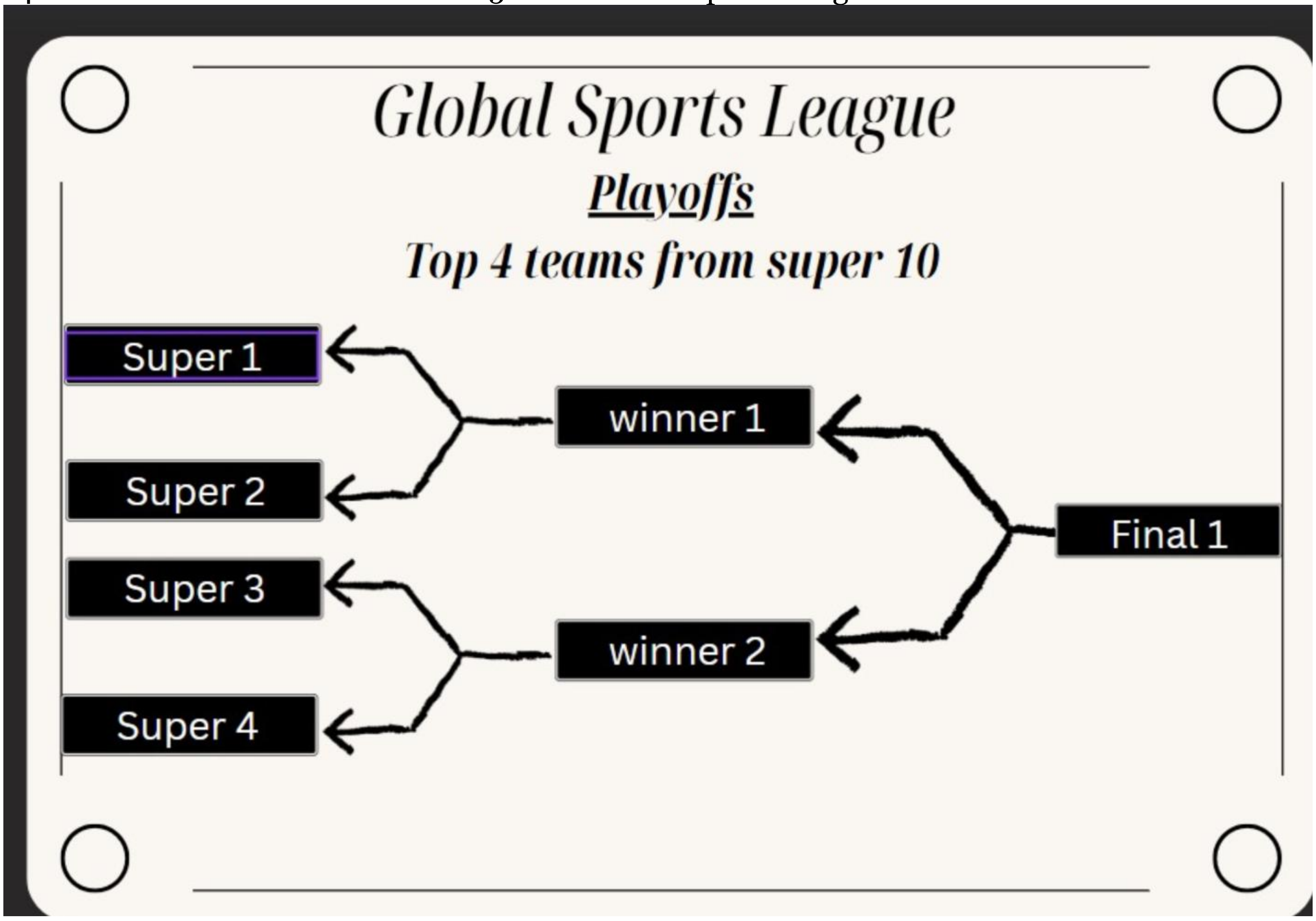
Global Sports League

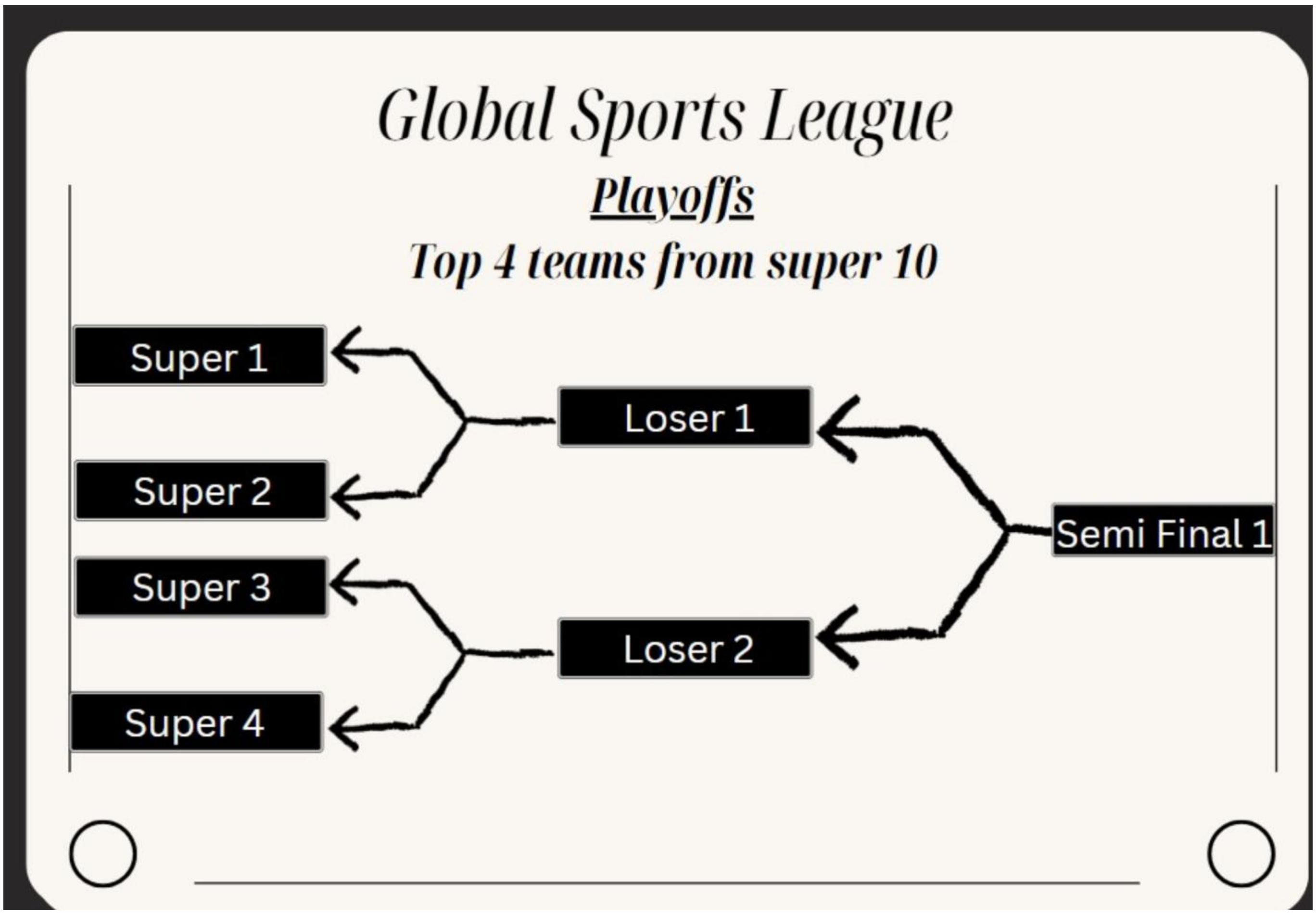
Super 10

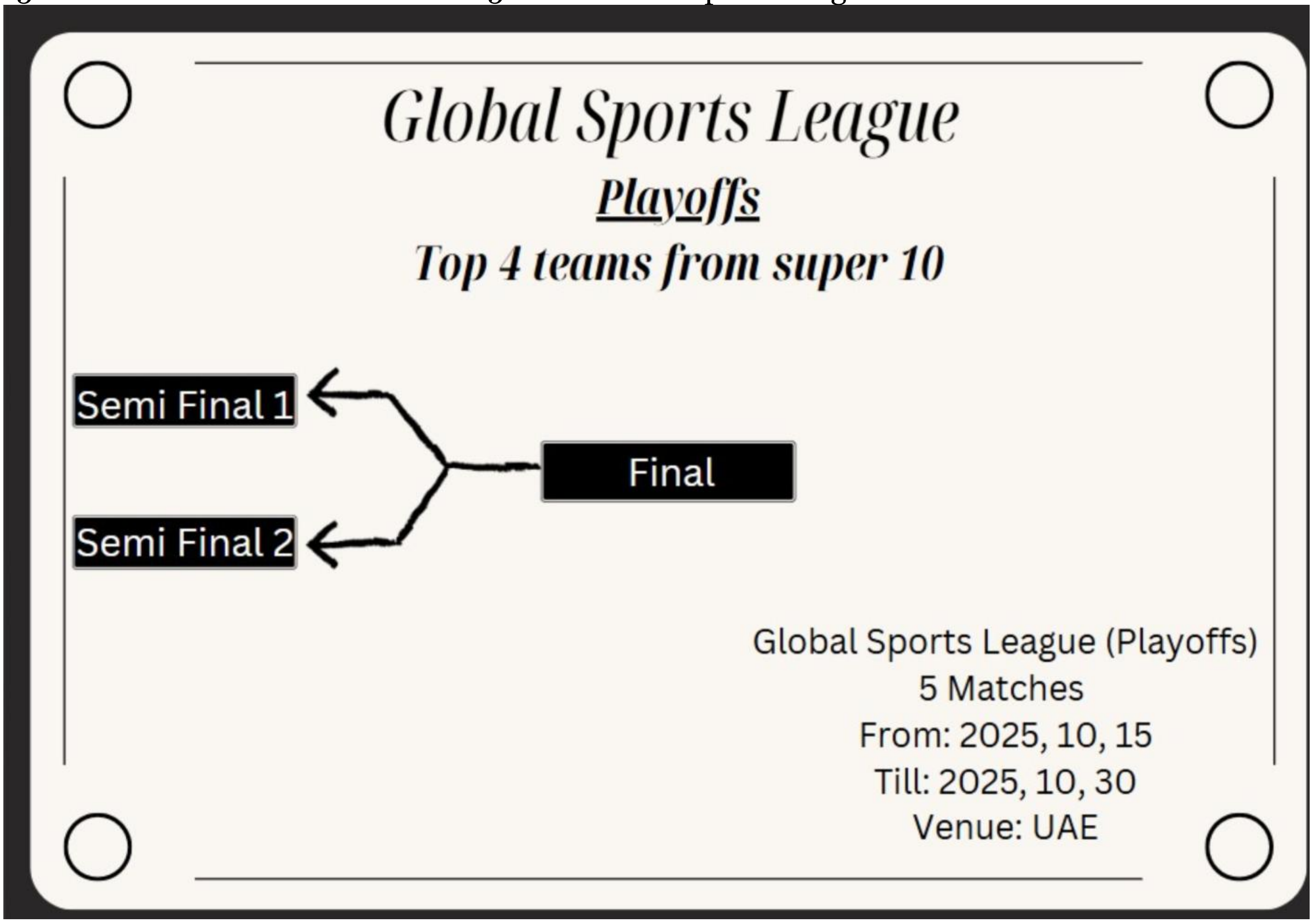
TOP 2 TEAMS FROM EVERY GROUP:

A from G1
B from G1
C from G2
D from G2
E from G3
F from G3
G from G4
H from G4
I from G5
J from G6

Global Sports League (Super 10)
45 Matches
Start: 2025,04,22
Till: 2025,06,05
Venue: UAE


Global Sports League
Playoffs
Top 4 teams from super 10
Super 1
Super 2
Super 3
Super 4
winner 1
winner 2
Final 1


Global Sports League
Playoffs
Top 4 teams from super 10
Super 1
Super 2
Super 3
Super 4
Loser 1
Loser 2
Semi Final 1

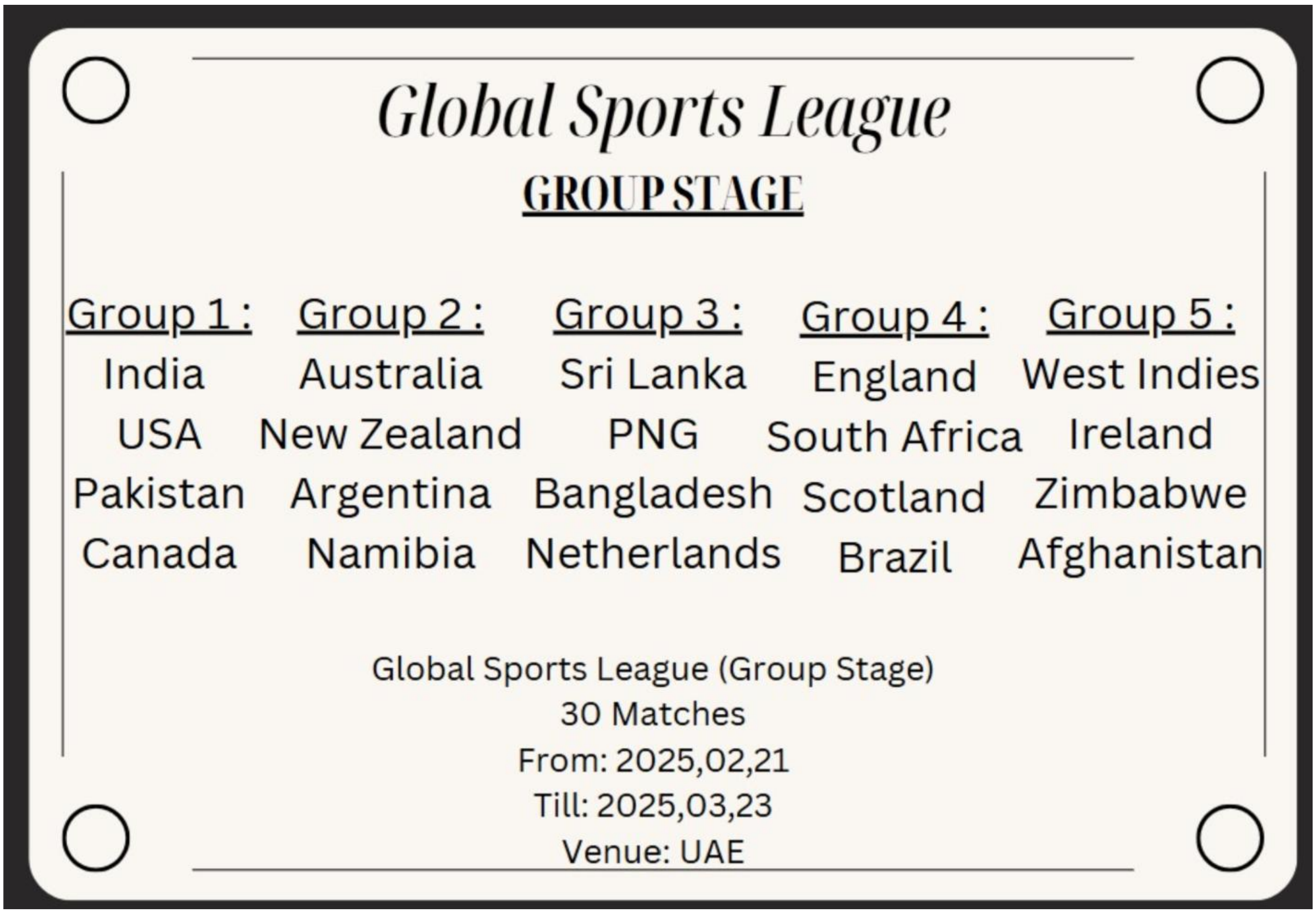


Global Sports League

Playoffs

Top 4 teams from super 10

Global Sports League (Playoffs)
5 Matches
From: 2025, 10, 15
Till: 2025, 10, 30
Venue: UAE



Global Sports League

GROUP STAGE

Group 1 :	Group 2 :	Group 3 :	Group 4 :	Group 5 :
India	Australia	Sri Lanka	England	West Indies
USA	New Zealand	PNG	South Africa	Ireland
Pakistan	Argentina	Bangladesh	Scotland	Zimbabwe
Canada	Namibia	Netherlands	Brazil	Afghanistan

Global Sports League (Group Stage)
30 Matches
From: 2025,02,21
Till: 2025,03,23
Venue: UAE

8. Appendix:

8.1 References:

[1]: https://en.wikipedia.org/wiki/Ludi_Romani

[2]: https://mat.tepper.cmu.edu/blog/index.php/2013/11/06/scheduling-major-league-baseball

[3]: https://en.wikipedia.org/wiki/Monday_Night_Football

[4]: https://operations.nfl.com/gameday/nfl-schedule/creating-the-nfl-schedule

[5]: https://en.wikipedia.org/wiki/Cricket_in_South_Asia

[6]: https://publications.fifa.com/en/vision-report-2021/the-football-landscape

[7]: https://www.icc-cricket.com/rankings/team-rankings/mens/t20i

[8]: https://youtu.be/2QndnA21zK0?si=rmNmXWRUJxtnszgM

[9]: https://en.wikipedia.org/wiki/2010_Commonwealth_Games

[10]: https://redballdata.blog/2019/09/17/recovery-time-in-one-day-cricket-quick-turnarounds-punish-bowlers

[11]: https://www.telegraph.co.uk/cricket/2024/10/01/english-cricket-block-players-franchise-league-pakistan-ipl

[12]: https://www.espncricinfo.com/story/india-will-not-travel-to-pakistan-for-2025-champions-trophy-1458978

[13]: https://en.wikipedia.org/wiki/Saitama_Stadium_2002

[14]: https://en.wikipedia.org/wiki/List_of_cricket_grounds_by_capacity

[15]: https://www.espncricinfo.com/series/new-zealand-in-united-arab-emirates-2023-1377725/united-arab-emirates-vs-new-zealand-2nd-t20i-1377728/full-scorecard

[16]: https://www.cricbuzz.com/cricket-team/spain/523

8.2 Code:

Disclaimer: no AI was used at all during the making of this report

20 teams code:

```
import random

import datetime

# These are constants that remain same or variable that we can use
again and again

tournament_start_date = datetime.date(2025, 2, 21)

preferred_time = "8pm to 10pm"

gap_day = 2

# This is basically a dictionary that has a score for these cricket
teams
```

```
teams = {

    "India": 268, "Pakistan": 236, "Sri Lanka": 232, "Bangladesh":
229,

    "Afghanistan": 222, "South Africa": 247, "Zimbabwe": 196,
"Namibia": 182,

    "England": 255, "Ireland": 197, "Scotland": 191, "Netherlands":
183,

    "New Zealand": 247, "Australia": 259, "PNG": 144, "Canada": 140,
"USA": 167,

    "West Indies": 247, "Argentina": 47, "Brazil": 15

}

# a pair has a prefered team to create that rivalry more is explained
in the report why we chose this

# lastly this is in a list

preferred = [

    ("India", "Pakistan"), ("Australia", "New Zealand"), ("Sri Lanka",
"Bangladesh")

]


sum_of_score = 0

count_of_teams = 0

# A very simple formula to add to count the total score of each team
in the dictionary of team's'

for team in teams:

    sum_of_score = sum_of_score + teams[team]

    count_of_teams = count_of_teams + 1


mean_score = sum_of_score / count_of_teams

# this is high rated empty list and simliarly low rated empty list

high_rated = []

low_rated = []

# this is a simple logic that state that teams of the index 'team'
without the 's' are sorted in high rated or low rated empty lists

for team, score in teams.items():

    if score > mean_score:

        high_rated.append(team)
    else:
        low_rated.append(team)

random.shuffle(high_rated)
random.shuffle(low_rated)

best_variance = 999999999
best_groups = None

num_groups = (count_of_teams + 3) // 4  # This ensures we round up if
there's a remainder

for _ in range(1000):
    random.shuffle(high_rated)
    random.shuffle(low_rated)

    used_teams = set()
    groups = []

    for i in range(num_groups):
        group = []

        # Add preferred pairs only if both teams are available
        for pair in preferred:
            if pair[0] in high_rated and pair[1] in high_rated:
                if pair[0] not in used_teams and pair[1] not in
used_teams:
                    group.append(pair[0])
                    group.append(pair[1])
                    used_teams.add(pair[0])
                    used_teams.add(pair[1])
                    break

        # Add remaining high-rated teams
        for team in high_rated:
            if team not in used_teams:
                group.append(team)
                used_teams.add(team)
            if len(group) == 2:
                break

        # Add low-rated teams
        for team in low_rated:
            if team not in used_teams:
                group.append(team)
                used_teams.add(team)
            if len(group) == 4:
                break

        groups.append(group)

    # So this part is to Ensure all teams are assigned to groups
    all_teams = set(teams.keys())
    assigned_teams = set().union(*groups)
    unassigned_teams = all_teams - assigned_teams

    if unassigned_teams:
        for team in unassigned_teams:
            for group in groups:
                if len(group) < 4:
                    group.append(team)
                    break

    # Calculate group scores
    group_scores = []
    for group in groups:
        total_score = 0
        for team in group:
            total_score = total_score + teams[team]
        group_scores.append(total_score)

    # Calculate mean group score
    sum_of_scores = 0
    count_of_groups = 0
    for score in group_scores:
        sum_of_scores = sum_of_scores + score
        count_of_groups = count_of_groups + 1
    mean_group_score = sum_of_scores / count_of_groups

    # Calculate variance
    sum_of_squared_diffs = 0
    for score in group_scores:
        diff = score - mean_group_score
        squared_diff = diff * diff
        sum_of_squared_diffs = sum_of_squared_diffs + squared_diff
    variance = sum_of_squared_diffs / count_of_groups

    # Update best groups if variance is lower
    if variance < best_variance:
        best_variance = variance
        best_groups = groups

# Shuffle the order of groups for randomization
random.shuffle(best_groups)


print("Best Balanced Groups with Minimum Variance and equal total
scores in each group:")
for i in range(len(best_groups)):
    print("Group", i + 1, ":", best_groups[i])
```

```
# this is a funciton to add the current index grouping to the current
index of matches

def generate_matches(group):

    matches = []

    for i in range(len(group)):

        for j in range(i + 1, len(group)):

            matches.append(group[i] + " vs " + group[j])

    return matches


# one list of all matches is a list that is going get ammended in the
loop below similarly we are making a dictionary of all the match dates


all_matches = []

match_dates = {}


for group in best_groups:

    group_matches = generate_matches(group)

    all_matches.extend(group_matches)


current_date = tournament_start_date

for match in all_matches:

    match_dates[match] = current_date

    current_date = current_date + datetime.timedelta(days=gap_day)


# so in simple terms we are concatinating(joining) the output for the
match details for each group

group_num = 1

for group in best_groups:

    print("\nGroup " + str(group_num) + " Teams: " + ", ".join(group))

    group_matches = generate_matches(group)

    start_date = match_dates[group_matches[0]]

    end_date = match_dates[group_matches[0]]


    for match in group_matches:
        if match_dates[match] < start_date:

            start_date = match_dates[match]

        if match_dates[match] > end_date:

            end_date = match_dates[match]


    print("Matches held from " + str(start_date) + " to " +
str(end_date) + ":")

    for match in group_matches:

        print("  " + match + " on " + str(match_dates[match]) + " " +
preferred_time)


    group_num = group_num + 1


# we created a blank list of the super matches

super = []


# Predefined input values

team_names = ["top1 of group1", "top2 of group1", "top1 of
group2","top2 of group2", "top1 of group3","top2 of group3","top1 of
group4","top2 of group4","top1 of group5","top2 of group5", "done"]

per_team_matches_input = 1

year_input = 2025

month_input = 6

day_input = 10

top1_input = "top 1st scorer of super matches"

top2_input = "top 2nd scorer of super matches"

top3_input = "top 3rd scorer of super matches"

top4_input = "top 4th scorer of super matches"

play_off_input = "True"

start_date_input = "10-06-2025"

match_time_input = "5: PM to 7:00 PM"

qualifier_winner_input = "top 1st scorer of super matches and top2nd
scorer of super matches"

eliminator1_winner_input = "top 3rd scorer of super matches and top
4th scorer of super matches"
```

```
eliminator2_winner_input = "Loser of Qualifier vs Winner of Eliminator
1"

final_winner_input = "Winner of Qualifier vs Winner of Eliminator 2"

semi1_winner_input = "top 1st scorer of super matches and top2nd
scorer of super matches"

semi2_winner_input = "top 3rd scorer of super matches and top 4th
scorer of super matches"


# Input team names

for ask in team_names:

    if ask == "done":

        break

    super.append(ask)


# So basically this Displays number of teams and their names

num_super = len(super)

print("Number of teams: %d" % num_super)

print("Teams: %s" % super)


# Input number of matches each team will play

per_team_matches = per_team_matches_input


# Generating matches and this is the empty list down below

matches = []

for q in range(per_team_matches):

    for i in range(num_super):

        for j in range(i + 1, num_super):

            matches.append((super[i], "vs", super[j]))


# Shuffle matches

random.shuffle(matches)


# Set the start date to today

print("Enter super matches start date")
```

```
year = year_input

month = month_input

day = day_input


# Use datetime.date to create the start date

supermatches_start_date = datetime.date(year, month, day)


print("Supermatches start date set to:", supermatches_start_date)


# Print matches with dates

no_matches = len(matches)

current_date = supermatches_start_date

preferred_time2 = " at 6pm"  # Example preferred time


for i in range(no_matches):

    print("Match on %s: %s" % (current_date, matches[i]) +
preferred_time2)

    current_date = current_date + datetime.timedelta(days=1)  #
Increment the date by 1 day


def increment_date(date_str, days):


    date_obj = datetime.datetime.strptime(date_str, "%d-%m-%Y")  #
Convert string to datetime object

    new_date = date_obj + datetime.timedelta(days=days)

    return new_date.strftime("%d-%m-%Y")  # Convert back to string


def tournament():

    # Input teams

    top1 = top1_input

    top2 = top2_input

    top3 = top3_input

    top4 = top4_input
    # Ask if playoffs are enabled

    play_off = play_off_input.strip().lower() == "true"


    # Input start date for the matches

    start_date = start_date_input


    # Input match time

    match_time = match_time_input


    if play_off:

        qualifier_date = start_date

        eliminator1_date = increment_date(qualifier_date, 1)

        eliminator2_date = increment_date(eliminator1_date, 1)

        final_date = increment_date(eliminator2_date, 1)


        print("\nPossible Matches (Playoffs Enabled):")

        print("Qualifier Match:", top1, "vs", top2, "on", 
qualifier_date, "at", match_time)

        print("Eliminator 1:", top3, "vs", top4, "on", 
eliminator1_date, "at", match_time)

        print("Eliminator 2: Loser of Qualifier vs Winner of 
Eliminator 1 on", eliminator2_date, "at", match_time)

        print("Final: Winner of Qualifier vs Winner of Eliminator 2 
on", final_date, "at", match_time)


        print("\nQualifier Match:", top1, "vs", top2, "on", 
qualifier_date, "at", match_time)

        qualifier_winner = qualifier_winner_input

        qualifier_loser = top1 if qualifier_winner == top2 else top2

        print(qualifier_winner, "wins and goes to Final")


        print("\nEliminator 1:", top3, "vs", top4, "on", 
eliminator1_date, "at", match_time)

        eliminator1_winner = eliminator1_winner_input
        print("\nEliminator 2:", qualifier_loser, "vs", 
eliminator1_winner, "on", eliminator2_date, "at", match_time)

        eliminator2_winner = eliminator2_winner_input


        print("\nFinal:", qualifier_winner, "vs", eliminator2_winner, 
"on", final_date, "at", match_time)

        final_winner = final_winner_input

        print(final_winner, "wins the Tournament!")


    else:

        semi1_date = start_date

        semi2_date = increment_date(semi1_date, 1)

        final_date = increment_date(semi2_date, 1)


        print("\nPossible Matches (Playoffs Disabled):")

        print("Semi-Final 1:", top1, "vs", top2, "on", semi1_date, 
"at", match_time)

        print("Semi-Final 2:", top3, "vs", top4, "on", semi2_date, 
"at", match_time)

        print("Final: Winner of Semi-Final 1 vs Winner of Semi-Final 2 
on", final_date, "at", match_time)


        print("\nSemi-Final 1:", top1, "vs", top2, "on", semi1_date, 
"at", match_time)

        semi1_winner = semi1_winner_input


        print("\nSemi-Final 2:", top3, "vs", top4, "on", semi2_date, 
"at", match_time)

        semi2_winner = semi2_winner_input


        print("\nFinal:", semi1_winner, "vs", semi2_winner, "on", 
final_date, "at", match_time)

        final_winner = final_winner_input

        print(final_winner, "wins the Tournament!")


# Run the tournament
```

```
tournament()
```

20 teams Output

```
Best Balanced Groups with Minimum Variance and equal total scores in each group:
Group 1 : ['India', 'Pakistan', 'Canada', 'Scotland']
Group 2 : ['England', 'South Africa', 'Brazil', 'Namibia']
Group 3 : ['Sri Lanka', 'Bangladesh', 'PNG', 'USA']
Group 4 : ['Zimbabwe', 'Afghanistan', 'West Indies', 'Ireland']
Group 5 : ['Australia', 'New Zealand', 'Argentina', 'Netherlands']

Group 1 Teams: India, Pakistan, Canada, Scotland
Matches held from 2025-02-21 to 2025-03-03:
  India vs Pakistan on 2025-02-21 8pm to 10pm
  India vs Canada on 2025-02-23 8pm to 10pm
  India vs Scotland on 2025-02-25 8pm to 10pm
  Pakistan vs Canada on 2025-02-27 8pm to 10pm
  Pakistan vs Scotland on 2025-03-01 8pm to 10pm
  Canada vs Scotland on 2025-03-03 8pm to 10pm

Group 2 Teams: England, South Africa, Brazil, Namibia
Matches held from 2025-03-05 to 2025-03-15:
  England vs South Africa on 2025-03-05 8pm to 10pm
  England vs Brazil on 2025-03-07 8pm to 10pm
  England vs Namibia on 2025-03-09 8pm to 10pm
  South Africa vs Brazil on 2025-03-11 8pm to 10pm
  South Africa vs Namibia on 2025-03-13 8pm to 10pm
  Brazil vs Namibia on 2025-03-15 8pm to 10pm

Group 3 Teams: Sri Lanka, Bangladesh, PNG, USA
Matches held from 2025-03-17 to 2025-03-27:
  Sri Lanka vs Bangladesh on 2025-03-17 8pm to 10pm
  Sri Lanka vs PNG on 2025-03-19 8pm to 10pm
  Sri Lanka vs USA on 2025-03-21 8pm to 10pm
  Bangladesh vs PNG on 2025-03-23 8pm to 10pm
  Bangladesh vs USA on 2025-03-25 8pm to 10pm
  PNG vs USA on 2025-03-27 8pm to 10pm


Group 4 Teams: Zimbabwe, Afghanistan, West Indies, Ireland
Matches held from 2025-03-29 to 2025-04-08:
  Zimbabwe vs Afghanistan on 2025-03-29 8pm to 10pm
  Zimbabwe vs West Indies on 2025-03-31 8pm to 10pm
  Zimbabwe vs Ireland on 2025-04-02 8pm to 10pm
  Afghanistan vs West Indies on 2025-04-04 8pm to 10pm
  Afghanistan vs Ireland on 2025-04-06 8pm to 10pm
  West Indies vs Ireland on 2025-04-08 8pm to 10pm


Group 5 Teams: Australia, New Zealand, Argentina, Netherlands
Matches held from 2025-04-10 to 2025-04-20:
  Australia vs New Zealand on 2025-04-10 8pm to 10pm
  Australia vs Argentina on 2025-04-12 8pm to 10pm
  Australia vs Netherlands on 2025-04-14 8pm to 10pm
  New Zealand vs Argentina on 2025-04-16 8pm to 10pm
  New Zealand vs Netherlands on 2025-04-18 8pm to 10pm
  Argentina vs Netherlands on 2025-04-20 8pm to 10pm
Number of teams: 10
Teams: ['top1 of group1', 'top2 of group1', 'top1 of group2', 'top2 of group2', 'top1 of group3',
'top2 of group3', 'top1 of group4', 'top2 of group4', 'top1 of group5', 'top2 of group5']
Enter super matches start date
Supermatches start date set to: 2025-06-10
Match on 2025-06-10: ('top2 of group2', 'vs', 'top1 of group4') at 6pm
Match on 2025-06-11: ('top1 of group3', 'vs', 'top1 of group5') at 6pm
Match on 2025-06-12: ('top2 of group2', 'vs', 'top1 of group3') at 6pm
Match on 2025-06-13: ('top2 of group1', 'vs', 'top1 of group3') at 6pm
Match on 2025-06-14: ('top1 of group4', 'vs', 'top2 of group4') at 6pm
Match on 2025-06-15: ('top1 of group3', 'vs', 'top2 of group4') at 6pm
Match on 2025-06-16: ('top2 of group1', 'vs', 'top2 of group4') at 6pm
Match on 2025-06-17: ('top1 of group2', 'vs', 'top2 of group5') at 6pm
```

Match on 2025-06-18: ('top2 of group1', 'vs', 'top2 of group3') at 6pm

Match on 2025-06-19: ('top1 of group2', 'vs', 'top2 of group2') at 6pm

Match on 2025-06-20: ('top1 of group2', 'vs', 'top2 of group3') at 6pm

Match on 2025-06-21: ('top2 of group2', 'vs', 'top2 of group5') at 6pm

Match on 2025-06-22: ('top1 of group2', 'vs', 'top1 of group3') at 6pm

Match on 2025-06-23: ('top1 of group1', 'vs', 'top1 of group5') at 6pm

Match on 2025-06-24: ('top1 of group1', 'vs', 'top1 of group4') at 6pm

Match on 2025-06-25: ('top2 of group1', 'vs', 'top2 of group5') at 6pm

Match on 2025-06-26: ('top1 of group4', 'vs', 'top1 of group5') at 6pm

Match on 2025-06-27: ('top2 of group1', 'vs', 'top1 of group5') at 6pm

Match on 2025-06-28: ('top2 of group2', 'vs', 'top1 of group5') at 6pm

Match on 2025-06-29: ('top2 of group3', 'vs', 'top2 of group4') at 6pm

Match on 2025-06-30: ('top1 of group2', 'vs', 'top1 of group4') at 6pm

Match on 2025-07-01: ('top1 of group3', 'vs', 'top2 of group3') at 6pm

Match on 2025-07-02: ('top1 of group1', 'vs', 'top2 of group2') at 6pm

Match on 2025-07-03: ('top1 of group1', 'vs', 'top1 of group3') at 6pm

Match on 2025-07-04: ('top1 of group1', 'vs', 'top2 of group4') at 6pm

Match on 2025-07-05: ('top2 of group3', 'vs', 'top1 of group4') at 6pm

Match on 2025-07-06: ('top1 of group1', 'vs', 'top2 of group3') at 6pm

Match on 2025-07-07: ('top2 of group2', 'vs', 'top2 of group3') at 6pm

Match on 2025-07-08: ('top1 of group2', 'vs', 'top1 of group5') at 6pm

Match on 2025-07-09: ('top1 of group1', 'vs', 'top2 of group1') at 6pm

Match on 2025-07-10: ('top2 of group1', 'vs', 'top2 of group2') at 6pm

Match on 2025-07-11: ('top1 of group1', 'vs', 'top1 of group2') at 6pm

Match on 2025-07-12: ('top1 of group5', 'vs', 'top2 of group5') at 6pm

Match on 2025-07-13: ('top1 of group1', 'vs', 'top2 of group5') at 6pm

Match on 2025-07-14: ('top2 of group4', 'vs', 'top1 of group5') at 6pm

Match on 2025-07-15: ('top1 of group2', 'vs', 'top2 of group4') at 6pm

Match on 2025-07-16: ('top2 of group2', 'vs', 'top2 of group4') at 6pm

Match on 2025-07-17: ('top2 of group3', 'vs', 'top1 of group5') at 6pm

Match on 2025-07-18: ('top2 of group4', 'vs', 'top2 of group5') at 6pm

Match on 2025-07-19: ('top1 of group3', 'vs', 'top2 of group5') at 6pm

Match on 2025-07-20: ('top2 of group1', 'vs', 'top1 of group2') at 6pm

Match on 2025-07-21: ('top2 of group3', 'vs', 'top2 of group5') at 6pm

Match on 2025-07-22: ('top2 of group1', 'vs', 'top1 of group4') at 6pm

Match on 2025-07-23: ('top1 of group3', 'vs', 'top1 of group4') at 6pm

Match on 2025-07-24: ('top1 of group4', 'vs', 'top2 of group5') at 6pm

Possible Matches (Playoffs Enabled):

Qualifier Match: top 1st scorer of super matches vs top 2nd scorer of super matches on 10-06-2025 at 5: PM to 7:00 PM

Eliminator 1: top 3rd scorer of super matches vs top 4th scorer of super matches on 11-06-2025 at 5: PM to 7:00 PM

Eliminator 2: Loser of Qualifier vs Winner of Eliminator 1 on 12-06-2025 at 5: PM to 7:00 PM

Final: Winner of Qualifier vs Winner of Eliminator 2 on 13-06-2025 at 5: PM to 7:00 PM

Qualifier Match: top 1st scorer of super matches vs top 2nd scorer of super matches on 10-06-2025 at 5: PM to 7:00 PM

top 1st scorer of super matches and top2nd scorer of super matches wins and goes to Final

Eliminator 1: top 3rd scorer of super matches vs top 4th scorer of super matches on 11-06-2025 at 5: PM to 7:00 PM

Eliminator 2: top 2nd scorer of super matches vs top 3rd scorer of super matches and top 4th scorer of super matches on 12-06-2025 at 5: PM to 7:00 PM

Final: top 1st scorer of super matches and top2nd scorer of super matches vs Loser of Qualifier vs Winner of Eliminator 1 on 13-06-2025 at 5: PM to 7:00 PM

Winner of Qualifier vs Winner of Eliminator 2 wins the Tournament!

24 teams code:

```
import random

import datetime


year = int(input("Enter the year of the tournament start date: "))

month = int(input("Enter the month of the tournament start date: "))

day = int(input("Enter the day of the tournament start date: "))
```

```
tournament_start_date = datetime.date(year, month, day)


preferred_time = input("Enter your preferred time (e.g., '8pm to 
10pm'): ")


gap_day = int(input("Enter the gap day (e.g., 1): "))


def populate_teams():
    teams = {}
    while True:
        team_name = input("Enter the name of the team (or type 'done' 
to finish): ")
        if team_name.lower() == 'done':
            break
        team_score = int(input("Enter the score for " + team_name + ": 
"))
        teams[team_name] = team_score


    return teams


def check_min_teams(teams):
    count = len(teams)
    while count < 8:
        print("The current number of teams is " + str(count) + ", 
which is less than the minimum requirement of 8.")
        print("You need to add " + str(8 - count) + " more teams.")
        for _ in range(8 - count):
            team_name = input("Enter the name of the team to add: ")
            team_score = int(input("Enter the score for " + team_name 
+ ": "))
            teams[team_name] = team_score
        count = len(teams)
    return teams
```

```
def check_team_count(teams):

    count = len(teams)

    remainder = count % 4

    while remainder != 0:

        print("The current number of teams is " + str(count) + ",
which is not divisible by 4.")

        print("You need to add " + str(4 - remainder) + " more teams
or remove " + str(remainder) + " teams.")

        choice = input("Do you want to add or remove teams? (Type
'add' or 'remove'): ").lower()


        if choice == 'add':

            for _ in range(4 - remainder):

                team_name = input("Enter the name of the team to add:
")

                team_score = int(input("Enter the score for " +
team_name + ": "))

                teams[team_name] = team_score

        elif choice == 'remove':

            for _ in range(remainder):

                team_name = input("Enter the name of the team to
remove: ")

                if team_name in teams:

                    del teams[team_name]

                    print("Removed " + team_name + ".")

                else:

                    print("Team " + team_name + " not found in the
dictionary.")

        else:

            print("Invalid choice. Please type 'add' or 'remove'.")



        count = len(teams)

        remainder = count % 4
    return teams



def populate_preferred(teams):
    preferred = []
    num_preferred = int(input("Enter the number of preferred matchups: 
"))

    for i in range(num_preferred):
        team1 = input("Enter the first team for preferred matchup " + 
str(i+1) + ": ")
        team2 = input("Enter the second team for preferred matchup " + 
str(i+1) + ": ")

        if team1 in teams and team2 in teams:
            preferred.append((team1, team2))
        else:
            print("One or both teams (" + team1 + ", " + team2 + ") 
are not in the teams dictionary. Skipping this matchup.")

    return preferred

teams = populate_teams()
teams = check_min_teams(teams)
teams = check_team_count(teams)
preferred = populate_preferred(teams)



print("\nTeams and their scores:")
for team, score in teams.items():
    print(team + ": " + str(score))

print("\nPreferred matchups:")
for matchup in preferred:
    print(matchup[0] + " vs " + matchup[1])


sum_of_score = 0

count_of_teams = 0


for team in teams:

    sum_of_score = sum_of_score + teams[team]

    count_of_teams = count_of_teams + 1



mean_score = sum_of_score / count_of_teams


high_rated = []

low_rated = []


for team, score in teams.items():

    if score > mean_score:

        high_rated.append(team)

    else:

        low_rated.append(team)


random.shuffle(high_rated)

random.shuffle(low_rated)


best_variance = 999999999

best_groups = None


for _ in range(1000):

    random.shuffle(high_rated)

    random.shuffle(low_rated)


    used_teams = set()

    groups = []
    for i in range(5):
        group = []



        for pair in preferred:
            if pair[0] in high_rated and pair[1] in high_rated:
                if pair[0] not in used_teams and pair[1] not in
used_teams:
                    group.append(pair[0])
                    group.append(pair[1])
                    used_teams.add(pair[0])
                    used_teams.add(pair[1])
                    break



        for team in high_rated:
            if team not in used_teams:
                group.append(team)
                used_teams.add(team)
            if len(group) == 2:
                break



        for team in low_rated:
            if team not in used_teams:
                group.append(team)
                used_teams.add(team)
            if len(group) == 4:
                break

        groups.append(group)
    group_scores = []
    for group in groups:
        total_score = 0
        for team in group:
            total_score = total_score + teams[team]
        group_scores.append(total_score)

    sum_of_scores = 0
    count_of_groups = 0
    for score in group_scores:
        sum_of_scores = sum_of_scores + score
        count_of_groups = count_of_groups + 1
    mean_group_score = sum_of_scores / count_of_groups

    sum_of_squared_diffs = 0
    for score in group_scores:
        diff = score - mean_group_score
        squared_diff = diff * diff
        sum_of_squared_diffs = sum_of_squared_diffs + squared_diff
    variance = sum_of_squared_diffs / count_of_groups

    if variance < best_variance:
        best_variance = variance
        best_groups = groups

print("Best Balanced Groups with Minimum Variance and equal total
scores in each group:")
for i in range(len(best_groups)):
    print("Group", i + 1, ":", best_groups[i])




def generate_matches(group):
    matches = []

    for i in range(len(group)):

        for j in range(i + 1 , len(group)):

            matches.append(group[i] + " vs " + group[j])

    return matches


all_matches = []

match_dates = {}


for group in best_groups:

    group_matches = generate_matches(group)

    all_matches.extend(group_matches)


for i in range(len(all_matches)):

    match_date = tournament_start_date + datetime.timedelta(days=i *
gap_day)

    match_dates[all_matches[i]] = match_date



group_num = 1

for group in best_groups:

    print("\nGroup " + str(group_num) + " Teams: " + ", ".join(group))

    group_matches = generate_matches(group)

    start_date = match_dates[group_matches[0]]

    end_date = match_dates[group_matches[0]]


    for match in group_matches:

        if match_dates[match] < start_date:

            start_date = match_dates[match]

        if match_dates[match] > end_date:

            end_date = match_dates[match]


    print("Matches held from " + str(start_date) + " to " +
str(end_date) + ":")
```

```
    for match in group_matches:
        print("  " + match + " on " + str(match_dates[match]) + " " +
preferred_time)

    group_num = group_num + 1




super = []


while True:
    ask = input("Enter the top teams that qualified to super matches
or enter 'done' to finish: ")
    if ask == "done":
        break
    super.append(ask)


num_super = len(super)
print("Number of teams: %d" % num_super)
print("Teams: %s" % super)


per_team_matches = int(input("Enter the number of matches each team
will play: "))


matches = []
for q in range(per_team_matches):
    for i in range(num_super):
        for j in range(i + 1, num_super):
            matches.append((super[i], "vs", super[j]))
```

```
random.shuffle(matches)



print("Enter super matches start date")
year = int(input("Enter the year: "))
month = int(input("Enter the month: "))
day = int(input("Enter the day: "))



supermatches_start_date = datetime.date(year, month, day)

print("Supermatches start date set to:", supermatches_start_date)



no_matches = len(matches)
current_date = supermatches_start_date
daycount = datetime.timedelta(days=gap_day)
preferred_time2 = " at 10:00 AM"

for i in range(no_matches):
    print("Match on %s: %s" % (current_date, matches[i]) +
preferred_time2)
    current_date = daycount + current_date




def increment_date(date_str, days):
    """Increment date manually using datetime module."""
    date_obj = datetime.datetime.strptime(date_str, "%d-%m-%Y")
    new_date = date_obj + datetime.timedelta(days=days)
    return new_date.strftime("%d-%m-%Y")
```

```
def tournament():


    top1 = input("Enter Top 1 team: ")

    top2 = input("Enter Top 2 team: ")

    top3 = input("Enter Top 3 team: ")

    top4 = input("Enter Top 4 team: ")




    play_off = input("Enable playoffs? (True/False): 
").strip().lower() == "true"




    start_date = input("Enter the start date for the matches (DD-MM-
YYYY): ")




    match_time = input("Enter the match time (e.g., 7:00 PM): ")


    if play_off:

        qualifier_date = start_date

        eliminator1_date = increment_date(qualifier_date, gap_day)

        eliminator2_date = increment_date(eliminator1_date, gap_day)

        final_date = increment_date(eliminator2_date, gap_day)


        print("\nPossible Matches (Playoffs Enabled):")

        print("Qualifier Match:", top1, "vs", top2, "on", 
qualifier_date, "at", match_time)

        print("Eliminator 1:", top3, "vs", top4, "on", 
eliminator1_date, "at", match_time)

        print("Eliminator 2: Loser of Qualifier vs Winner of 
Eliminator 1 on", eliminator2_date, "at", match_time)

        print("Final: Winner of Qualifier vs Winner of Eliminator 2 
on", final_date, "at", match_time)
```

```
        print("\nQualifier Match:", top1, "vs", top2, "on",
qualifier_date, "at", match_time)

        qualifier_winner = input("Enter winner of Qualifier Match: ")

        qualifier_loser = top1 if qualifier_winner == top2 else top2

        print(qualifier_winner, "wins and goes to Final")


        print("\nEliminator 1:", top3, "vs", top4, "on",
eliminator1_date, "at", match_time)

        eliminator1_winner = input("Enter winner of Eliminator 1: ")


        print("\nEliminator 2:", qualifier_loser, "vs",
eliminator1_winner, "on", eliminator2_date, "at", match_time)

        eliminator2_winner = input("Enter winner of Eliminator 2: ")


        print("\nFinal:", qualifier_winner, "vs", eliminator2_winner,
"on", final_date, "at", match_time)

        final_winner = input("Enter winner of Final: ")

        print(final_winner, "wins the Tournament!")


    else:

        semi1_date = start_date

        semi2_date = increment_date(semi1_date, gap_day)

        final_date = increment_date(semi2_date, gap_day)


        print("\nPossible Matches (Playoffs Disabled):")

        print("Semi-Final 1:", top1, "vs", top2, "on", semi1_date,
"at", match_time)

        print("Semi-Final 2:", top3, "vs", top4, "on", semi2_date,
"at", match_time)

        print("Final: Winner of Semi-Final 1 vs Winner of Semi-Final 2
on", final_date, "at", match_time)


        print("\nSemi-Final 1:", top1, "vs", top2, "on", semi1_date,
"at", match_time)

        semi1_winner = input("Enter winner of Semi-Final 1: ")
        print("\nSemi-Final 2:", top3, "vs", top4, "on", semi2_date,
"at", match_time)

        semi2_winner = input("Enter winner of Semi-Final 2: ")


        print("\nFinal:", semi1_winner, "vs", semi2_winner, "on",
final_date, "at", match_time)

        final_winner = input("Enter winner of Final: ")

        print(final_winner, "wins the Tournament!")



tournament()


24 team output:
```