I. Group Members

Izzy Kim, kimizzy@seas.upenn.edu, kimizzy1130
Joseph Cho, hwisang@seas.upenn.edu, hwisangcho00
Hyun Cho, hyunkic@seas.upenn.edu, saesak
Jin-seo Kim, jins0904@sas.upenn.edu, jinseo0904

II. Description of Web Application

This application allows users to simulate a League of Legend match in an interactive manner. This application will allow users to choose team compositions, decide whether they get objectives/first blood, etc, and then predict their chances of winning. Also, it will provide an interface for users to query the datasets to search for certain players, certain champion win rates, and other trivia/interesting facts that they can get from querying the dataset.

III. Dataset

A. League of Legends Ranked Games Dataset (3 csv files)
Link to dataset:
https://www.kaggle.com/datasets/gyejr95/league-of-legendslol-ranked-games-2020-ver1?select=match_loser_data_version1.csv
This dataset contains 3 large csv files, ***each*** of 108K rows: overall match data, winner team data, and loser team data. The data is based on the match results and in-game data of 108K games played in the Korean server in 2020. In-game data includes: first dragon, first blood, dragons count, rift herald count, etc.
The dataset was found on Kaggle. This dataset contains 108K rows and 18 attributes.

B. League of Legends Champion Dataset (2 csv files)
Link to dataset:
https://www.kaggle.com/datasets/gyejr95/league-of-legendslol-champion-and-item-2020
The dataset was found on Kaggle. This database contains ~ 400 rows and 40 attributes containing information about champions such as name, background, playing difficulty, stats such as health and mana, and item information such as name, price, and effects.
Overall, we are querying 5 different csv files.

IV. Summary Statistics of Several Attributes

- **First Blood:** First kill of the game. Represented as a boolean telling whether each team (winner / loser) scored a first blood or not.
    - Winner Team: True 60% (65.5K rows), False 40% (43.3K)
- **First Turret:** Turrets, also called towers, are fortifications that attack enemy units on sight. Represented as a boolean telling whether each team destroyed the first enemy turret faster than the opponent team
    - Winner Team: True 73% (78.9K rows), False 27% (29.9K)
- **First Dragon:** Dragons are critical neutral objectives which offer golds and other advantageous effects to the team which slayed them. The team which killed four dragons earns the dragon's soul, which greatly boosts the chance of winning the game. Represented as a boolean telling whether each team slayed the first dragon of the game.
    - Winner Team: 56% (60.9K rows), False 44% (47.9K)
- **Game Duration:** Represented in seconds. Mean 1420 seconds (23.7K), Standard Deviation 436 seconds

V. Sample Queries

Because the first dataset is split into 3 separate datasets that each contain a lot of disparate data, we will need a lot of subqueries and joins in the first place to query data from the datasets.

1. **Join** the champion dataset with the games dataset. Then given a team composition for each team with other attributes, etc specified, search the dataset for matches that are identical, and generate a probability that one team will win (probability = count of wins / count of total games). (Joins, Aggregations, Subqueries)

2. Search the database for how likely it is for your team to win the match based on game status (ex. Take dragon by 10 minutes) → (probability = count of wins / count of total games). (Joins, Aggregations, Subqueries)
3. **Join** the champions dataset with the game dataset. Then list out the champions with the best win rates (**count** of games won with champion / **count** of total games with champion) in descending order. (Joins, Aggregations, Subqueries, Order By)
4. Search for a certain player within the dataset, and then construct a profile for them, getting their champions with win rates (similar to query 3), how many games they have played (**count** on games played), etc. (Joins, Aggregations, Subqueries)
5. Show the best compositions/combinations of champions in positions. For example, the bot lane composition with the highest probability of winning (**count** of games won / count of total games) (Joins, Aggregations, Subqueries)