مدينة زويل للعلوم والتكنولوچيا
Zewail City of Science and Technology

Mini Project 2 - Apache Spark

CIE 427

Prepared For
Dr. Elsayed Hemayed

Prepared By

Ibrahim Hamada	201800739
Mazen Hassan	201801897

Table of content :

Data Collection

We started by collecting data from the Riot Games API. We collected a total of “140822” matches, then we filtered out all non-ranked matches to end up with “52941” ranked matches. We ran different iterations using different API keys in order to increase the obtained data given that each API key is valid for only 24 hours with RATE LIMITS of 20 requests every 1 seconds and 100 requests every 2 minutes. The data collection pipeline is as follows:

1) **<u>Get summoner’s names :</u>**

In each iteration, we collected 1000 different summoners from the 'euw1' server and 'europe' region. The summoners’ ranks are distributed based on the rank distribution of the
Game. We can see that each tier has a percentage of the game, so we took a sample of players according to the distribution of each rank.

2) **<u>Get matches IDs each summoner played:</u>**

After getting summoners names, we obtained their unique identifier (puuid) by calling the API responsible for getting (puuid) given a summoner name. Then, we extracted different matches that each summoner participated in. We got the matches IDs for each summoner by queuring the matches API.

3) **<u>Get matches information from match ids:</u>**

This step is considered as the bottleneck of the whole process since it took very much time. We extracted the information of each match given the IDs that were extracted in the previous step.

4) **<u>Get Champions information:</u>**

In this step, we retrieved the JSON file that contains all champions information to use it in the requirements such as: champion win rates, champion pick rates, champion ban rates, and champion synergies.

5) **<u>Get Items information:</u>**

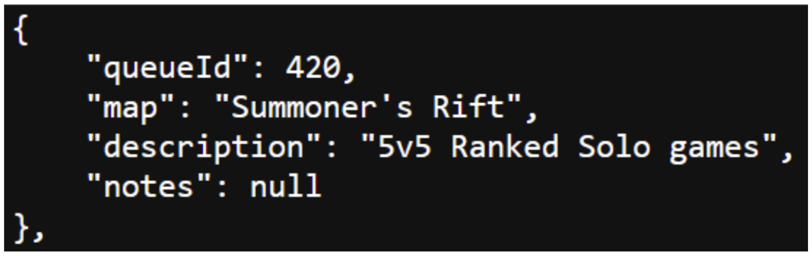
In this step, we retrieved the JSON file that contains all items information to use it in the requirements such as: Item win rates, item pick rates, item synergies, and item suggestion.

Data Analysis

In this stage, the obtained JSON files from the data collection process are preprocessed before starting analyzing the data.

1) **<u>Obtaining ranked matches only:</u>**

Since it is not necessary for all ranked summoners to play only ranked matches, we filtered out all non-ranked matches. We explored the documentation and found that "queueId" is responsible for determining the match type. Additionally, we found that matches with "queueId = 420" are the ranked matches, so we filtered them out of the total matches.



```
{
    "queueId": 420,
    "map": "Summoner's Rift",
    "description": "5v5 Ranked Solo games",
    "notes": null
},
```

2) **<u>Calculating some values:</u>**

We found the total number of matches using the count() function. We found teams count based on the fact that each match is played with 2 teams. We found the participants count given the fact that each team consists of 10 members.

After that, we started dealing with ranked_matches rdd to find the requirements as follows:

1) **<u>Requirement 1:</u>** (Champion pick rate, Champion win rate, and Champion ban rate)
    a) Champion pick rate:
        I. We firstly found the participants in all Ranked Matches
        II. We found the champions associated with the participants.
        III. We found the count of each champion.
        IV. Finally, we found the champion's pick rates.

    b) Champion win rate:
        I. We firstly found the participants who winning in all Ranked Matches
        II. We found the champions associated with the winning participants.
        III. We found the count of each winning champion.
        IV. Finally, we found the champions' win rates.

c) Champion ban rate:

    I. We firstly found the participated teams in all Ranked Matches
    II. We found the teams that were banned.
    III. We found the champions' IDs associated with the banned participants.
    IV. Finally, we found the champions' ban rates associated with each champion name by dividing the number of bans by ranked_matches_count.

2) **Requirement 2:** (Champion Synergies or duos)

    a) Champion Synergies: it is defined as the number of times 2 champions won together (in the same team) divided by the total number of times these 2 champions played together (same team).

        I. We firstly found the participants in all Ranked Matches.
        II. We found all possible duos in all ranked matches.
        III. We found the count of each possible duos.
        IV. We found the all winning duos.
        V. We found the count of winning duos.
        VI. Finally, we found the champions' synergies by dividing the count that a certain duo won by the number of times that this duo played together.

3) **Requirement 3:** (Item win rates, item pick rates)

    a) Item pick rate:

        I. We firstly found the participants in all Ranked Matches
        II. We found the items associated with the participants.
        III. We found the count of each item.
        IV. Finally, we found the items' pick rates.

    d) Item win rate:

        I. We firstly found the participants who winning in all Ranked Matches
        II. We found the items associated with the winning participants.
        III. We found the count of each winning item.
        IV. Finally, we found the items' win rates.

4) **Requirement 4:** (Item Synergies (item with champion, item with class))
   a) item with champion:
      V. We firstly found the participants in all Ranked Matches
      VI. We found the items associated with the participants.
      VII. We found the winning participants in all Ranked Matches.
      VIII. We found the items associated with the winning participants.
      IX. We joined the two values and divided them to determine the pairs of champions with items and the win rate.

   b) item with class:
      V. We firstly found the participants in all Ranked Matches
      VI. We found the lanes in all Ranked Matches
      VII. We found the items associated with the participants.
      VIII. We found the winning participants in all Ranked Matches.
      IX. We found the lanes and items associated with the winning participants.
      X. We joined the two values and divided them to determine the pairs of lanes with items and the win rate.

5) **Requirement 5:** (Item suggestion)

   Given the champion name, we get the obtained champion items duos that were found in requirement 4a and sort them based on the win rates.

6) **Extra Requirement:** (Best lane for champion / win rates of different lanes for champions)

   Following the same approach of finding synergies, we found the count of all matches that champions played in lanes.
   Then, we found the count of won matches in lanes.
   We joined them together and divided the two values.
   Finally, we sorted the obtained lanes for a given champion using filter and sort functions.

Data Problems

There were some problems related to the data such as:

1) While extracting the data from the API, we encountered some problems because the API fails to retrieve the data associated with certain champion names. We thought that their accounts might be deleted, so we ignored their names using the try-except approach.
2) There exist some old batches that go back to game versions in 2010. We thought that these batches are associated with participants who didn't play any match for a long period of time, but return to play in the current versions.
3) There exist some champion Ids of (-1) which led to an error while extracting champion names from the champion.json files.
4) Fiddlesticks champion was written in the champions.json file as "Fiddlesticks", however it was written in the matches information as "FiddleSticks". We solved this problem by renaming the champion in the champions.json file to "FiddleSticks".
5) It is known that each champion has 6 items numbered from 0 to 6. Each item has an ID, but there is a value of "0" that indicates "no item". The case of (item_id = 0) is handled while extracting the item name from the item Id.

Challenges:

We faced some challenges while collecting the data because as mentioned above: Riot Games API sets some constraints such as each API key is valid for only 24 hours with RATE LIMITS of 20 requests every 1 seconds and 100 requests every 2 minutes. Accordingly, we used more than one API key at the same time by creating different accounts so that they are used sequentially in a manner that allows requesting from a certain API, then keeping it and requesting from another one as not to exceed the rate limit. Then, data will be saved to the Google drive.

We faced another problem when dealing with very large JSON files that contain 60K or more matches since they consume a lot of time to be processed, so we divided these large files into small files. After that, we read all of the small files using the following command that gets the paths of all files within the directory. Therefore, all files are combined in the same RDD.

```
path = '/content/drive/MyDrive/BigData/Data2/*.json'
data_rdd = spark.read.option("multiline","true").json(path).rdd
```