Queries

1. Given a champion, choose five champions from each lane (TOP, MIDDLE, BOTTOM, JUNGLE) with the highest winning probability when played as a team

\*Champion\_id must be set to the champion picked by the user.

\*{Each Lane} could be either one of these four: TOP, JUNGLE, MIDDLE, or BOTTOM.

\* The final SELECT statement will have to be executed once for each lane.

with champ\_games as (select game\_id as gamenum, team\_id as teamnum

from Player

where champion\_id = {Your\_Champion\_ID}),

team\_champs as (select \*

from Player p join champ\_games c on p.team\_id = c.teamnum and p.game\_id = c.gamenum),

prob\_table as (SELECT champion\_id, *SUM*(win) as wins, *COUNT*(\*) as total\_games,

*SUM*(win) / *COUNT*(\*) AS win\_probability

FROM team\_champs

GROUP BY champion\_id),

probs as (

select p.\*, C.champion\_name

from prob\_table p join Champion C on p.champion\_id = C.champion\_id

order by win\_probability desc),

lanecount as (

select

pr.champion\_id,

pr.champion\_name,

pr.win\_probability,

lc.lane,

lc.count

from probs pr

join (

select

champion\_id,

timeline\_lane as lane,

*count*(\*) as count

from team\_champs

where timeline\_lane <> 'NONE'

group by champion\_id, timeline\_lane

order by champion\_id, *count*(\*) desc

) lc on pr.champion\_id = lc.champion\_id

group by pr.champion\_id)

select \*

from lanecount

where lane = '{Each Lane}' and count > 10 and champion\_id <> {Your\_Champion\_ID}

order by win\_probability desc

limit 5

1. Given a champion, choose five champions from each lane with the highest losing probability when played as an opponent team

with champ\_games as (select game\_id as gamenum, team\_id as teamnum

from Player

where champion\_id = {Your\_Champion\_ID}),

op\_champs as (select \*

from Player p join champ\_games c on p.team\_id <> c.teamnum and p.game\_id = c.gamenum),

prob\_table as (SELECT champion\_id, *SUM*(win) as wins, *COUNT*(\*) as total\_games,

*SUM*(win) / *COUNT*(\*) AS win\_probability

FROM op\_champs

GROUP BY champion\_id),

probs as (

select p.\*, C.champion\_name

from prob\_table p join Champion C on p.champion\_id = C.champion\_id

order by win\_probability desc),

lanecount as (

select

pr.champion\_id,

pr.champion\_name,

pr.win\_probability,

lc.lane,

lc.count

from probs pr

join (

select

champion\_id,

timeline\_lane as lane,

*count*(\*) as count

from op\_champs

where timeline\_lane <> 'NONE'

group by champion\_id, timeline\_lane

order by champion\_id, *count*(\*) desc

) lc on pr.champion\_id = lc.champion\_id

group by pr.champion\_id)

select \*

from lanecount

where lane = '{Each Lane}' and count > 10 and champion\_id <> {Your\_Champion\_ID}

order by win\_probability desc

limit 5;

1. Given a champion, recommend the list of items by presenting them in order of highest winning probability to lowest, when an item is purchased.

-- find the items, merge all columns

with champ\_games as (select game\_id as gamenum, team\_id as teamnum

from Player

where champion\_id = {Your\_Champion\_ID}),

team\_champs as (select \*

from champ\_games c join Player p on p.team\_id = c.teamnum and p.game\_id = c.gamenum),

champ\_items as (

select \*

from team\_champs

where champion\_id = {Your\_Champion\_ID}),

merged\_items as (SELECT win, item

FROM (SELECT win, stats\_item0 AS item

FROM champ\_items

UNION ALL

SELECT win, stats\_item1

FROM champ\_items

UNION ALL

SELECT win, stats\_item2

FROM champ\_items

UNION ALL

SELECT win, stats\_item3

FROM champ\_items

UNION ALL

SELECT win, stats\_item4

FROM champ\_items

UNION ALL

SELECT win, stats\_item5

FROM champ\_items

UNION ALL

SELECT win, stats\_item6

FROM champ\_items) as items\_single\_column),

item\_recs as (

select item, *SUM*(win) as wins, *COUNT*(\*) as total, *SUM*(win)/*COUNT*(\*) as prob\_wins

from merged\_items

where item <> 0

group by item

having *count*(\*) > 100)

select i.\*, it.item\_name as name

from item\_recs i join Item it on i.item = it.item\_id

order by prob\_wins desc

;

1. Given 5 champion\_ids, return the number of wins and losses for when those 5 champion\_ids are on the same team

SELECT

*COUNT*(\*) AS total\_games,

*SUM*(CASE WHEN p.win = 1 THEN 1 ELSE 0 END) AS wins,

*SUM*(CASE WHEN p.win = 0 THEN 1 ELSE 0 END) AS losses

FROM

Player p

WHERE

p.champion\_id IN (266, 84, 12, 523, 60)

GROUP BY

p.game\_id, p.team\_id

HAVING

*COUNT*(DISTINCT p.champion\_id) = 5;

1. Calculate winrate for teams that have 1, 2, 3, 4, or 5 champions that are ranged on the team (trying to see how ranged champions influence winrate)

WITH TeamAttackRange AS (

SELECT

t.win,

t.team\_id,

t.game\_id,

*COUNT*(DISTINCT CASE WHEN c.stats\_attackrange > 200 THEN p.player\_id END) AS champions\_above\_200

FROM

Team t

JOIN

Player p ON t.game\_id = p.game\_id AND t.team\_id = p.team\_id

JOIN

Champion c ON p.champion\_id = c.champion\_id

GROUP BY

t.team\_id, t.game\_id

) -- returns champions\_above\_200, a number that is how many champions have attack range over 200

SELECT

champions\_above\_200,

*COUNT*(\*) AS total\_games,

*SUM*(CASE WHEN win = 1 THEN 1 ELSE 0 END) AS wins,

*CAST*(*SUM*(CASE WHEN win = 1 THEN 1 ELSE 0 END) AS DECIMAL) / *NULLIF*(*COUNT*(\*), 0) AS winrate

FROM

TeamAttackRange

GROUP BY

champions\_above\_200

ORDER BY

winrate DESC;

1. Show data for a certain champion (champion\_id, name, average (kills, deaths, assists, largestKillingSpree, largestMultiKill, killingSprees, longestTimeSpentLiving))

WITH ChampionStats AS (

SELECT

p.champion\_id,

p.stats\_kills,

p.stats\_deaths,

p.stats\_assists,

p.stats\_largestKillingSpree,

p.stats\_largestMultiKill,

p.stats\_killingSprees,

p.stats\_longestTimeSpentLiving

FROM

Player p

WHERE

p.champion\_id = '84'

)

SELECT

*AVG*(p.stats\_kills) AS avg\_kills,

*AVG*(p.stats\_deaths) AS avg\_deaths,

*AVG*(p.stats\_assists) AS avg\_assists,

*AVG*(p.stats\_largestKillingSpree) AS avg\_largestKillingSpree,

*AVG*(p.stats\_largestMultiKill) AS avg\_largestMultiKill,

*AVG*(p.stats\_killingSprees) AS avg\_killingSprees,

*AVG*(p.stats\_longestTimeSpentLiving) AS avg\_longestTimeSpentLiving

FROM

ChampionStats as p

GROUP BY

champion\_id;

1. Calculate the win rate for each champion (each champion can be distinguished by the unique champion\_id) and return the list of champions and their respective win rates in descending order (the champion with the highest win rate appears first on the list).

SELECT champion\_id, *COUNT*(DISTINCT game\_id) AS total\_games\_played, *SUM*(win) AS total\_games\_won, *AVG*(win) \* 100 AS win\_rate

FROM Player

GROUP BY champion\_id

ORDER BY win\_rate DESC;

1. Calculate the pick rate for each champion (that is, how many times that champion has been chosen to play in a game) and return the list of champions and their respective pick rate in descending order (the champion with the highest pick rate appears first on the list).

WITH TotalGames AS (

SELECT *COUNT*(DISTINCT game\_id) AS total\_games

FROM Game

)

,

GamesPlayedByChampion AS (

SELECT champion\_id, *COUNT*(DISTINCT game\_id) AS total\_games\_played

FROM Player

GROUP BY champion\_id

)

SELECT GPC.champion\_id, (*CAST*(GPC.total\_games\_played AS DECIMAL) / TG.total\_games) \* 100 AS pick\_rate

FROM GamesPlayedByChampion GPC

CROSS JOIN TotalGames TG

ORDER BY pick\_rate DESC;

1. Calculate the win rate for each stat

SELECT 'firstBlood' AS stat, *SUM*(firstBlood = 1 AND win = 1) / *SUM*(firstBlood = 1) AS winrate

FROM Team

UNION

SELECT 'firstTower' AS stat, *SUM*(firstTower = 1 AND win = 1) / *SUM*(firstTower = 1) AS winrate

FROM Team

UNION

SELECT

'firstInhibitor' AS stat, *SUM*(firstInhibitor = 1 AND win = 1) / *SUM*(firstInhibitor = 1) AS winrate

FROM Team

UNION

SELECT

'firstBaron' AS stat, *SUM*(firstBaron = 1 AND win = 1) / *SUM*(firstBaron = 1) AS winrate

FROM Team

UNION

SELECT

'firstDragon' AS stat, *SUM*(firstDragon = 1 AND win = 1) / *SUM*(firstDragon = 1) AS winrate

FROM Team

UNION

SELECT

'firstRiftHerald' AS stat, *SUM*(firstRiftHerald = 1 AND win = 1) / *SUM*(firstRiftHerald = 1) AS winrate

FROM Team;

1. Calculate the winrates for each item, also returning some of the information about each item

WITH ItemWinRates AS (

SELECT

p.stats\_item0 AS item\_id,

*COUNT*(\*) AS total\_games,

*SUM*(CASE WHEN p.win = 'Win' THEN 1 ELSE 0 END) AS wins

FROM

Player p

WHERE

p.stats\_item0 IS NOT NULL -- Exclude games where the item is not specified

GROUP BY

p.stats\_item0

)

SELECT

iwr.item\_id,

i.item\_name,

i.item\_explain,

iwr.total\_games,

iwr.wins,

(iwr.wins \* 100.0 / *NULLIF*(iwr.total\_games, 0)) AS win\_rate

FROM

ItemWinRates iwr

JOIN

Item i ON iwr.item\_id = i.item\_id

ORDER BY

win\_rate DESC, total\_games DESC;

**3. Credentials**

AWS Credentials:

MySQL Data Source

User: admin

Password: qlalfqjsgh99

Host: database-1.cwmp2orqeyfp.us-east-1.rds.amazonaws.com

Port: 3306