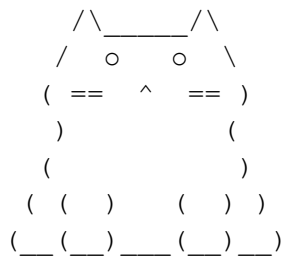
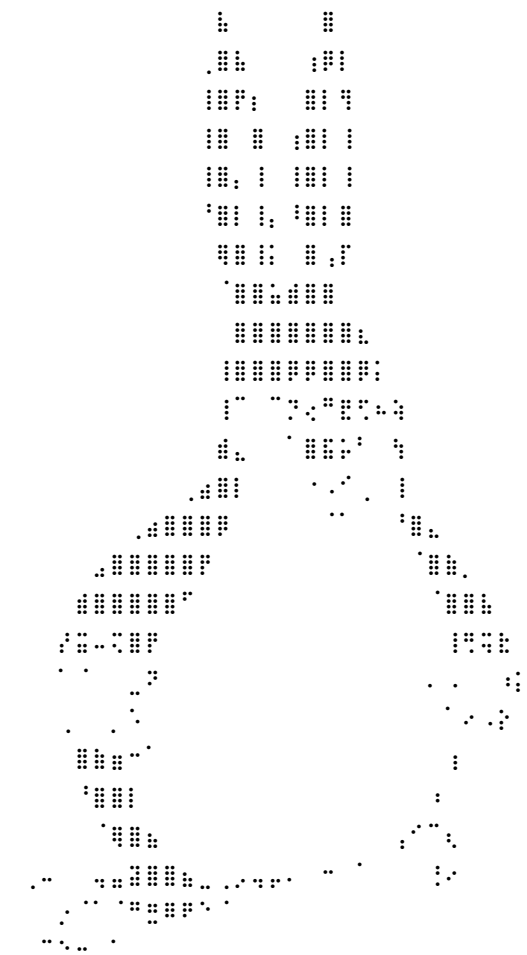
Team Chonk





BIG CHUNGUS

```
create table games (
    game_id varchar(20) ,
    map_name varchar(20),
    date_info varchar (20),
    primary key(game_id),
);
```

```
create table maps (
    map_name varchar(20),
    play_rate real,
    atk_win real,
    def_win real,
    num_matches int,
    tier int,
    primary key (map_name, tier)
);

create table player_stats (
    game_id int,
    team_name varchar(50),
    player_id varchar(50),
    agent_name varchar(20),
    average_combat_score int,
    kills int,
    deaths int,
    assists int,
    kill_assist_trade_survive_ratio real,
    average_damage_per_round int,
    headshot_ratio real,
    first_kills int,
    first_deaths int,
    team_side char(2),
    tier int,
    primary key (game_id, player_id, team_side)
);

create table weapons (
    weapon_name varchar(20),
    kills_per_match real,
    headshot real,
    bodyshot real,
    legshot real,
    damage_per_round int,
    map_name varchar(20),
    tier int,
    primary key (weapon_name, map_name, tier),
    foreign key (map_name, tier) references maps(map_name, tier)
);
```

```
create table agents (
    agent_name varchar(20),
    kd real,
    kills real,
    deaths real,
    assists real,
    win_rate real,
    pick_rate real,
    acs int,
    first_blood real,
    num_matches int,
    map_name varchar(20),
    tier int,
    ability_1 real,
    ability_2 real,
    ability_3 real,
    ultimate real,
    primary key (agent_name, map_name, tier),
    foreign key (map_name, tier) references maps(map_name, tier)
);
```

```
❯ mysql -u chonk -p -h 34.173.148.192
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 20789
Server version: 8.0.31-google (Google)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use chonk;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----------------+
| Tables_in_chonk |
+-----------------+
| agents          |
| games           |
| maps            |
| player_stats    |
| weapons         |
+-----------------+
5 rows in set (0.03 sec)
```

```
mysql> select * from player_stats limit 10; select count(*) from player_stats;
+--------+-------------+------------+--------+-----------------------+-------+--------+---------+--------------+-----------------------------------+-------------------------+-----------------+-------------+-------------+-----------+------+
| game_id | player_id     | agent_name | rating | average_combat_score | kills | deaths | assists | kills_deaths | kill_assist_trade_survive_ratio | average_damage_per_round | headshot_ratio | first_kills | first_deaths | team_side | tier |
+--------+-------------+------------+--------+-----------------------+-------+--------+---------+--------------+-----------------------------------+-------------------------+-----------------+-------------+-------------+-----------+------+
|  117185 | AsLanM4shadoW | Killjoy    |   1.17 |                   289 |     7 |      5 |       1 |            2 |                               100 |                     184 |              33 |           0 |           0 | ct        |   21 |
|  117185 | AsLanM4shadoW | Killjoy    |   1.32 |                   229 |    11 |      7 |       0 |            4 |                                83 |                     155 |              56 |           2 |           0 | t         |   21 |
|  117185 | Boaster       | Omen       |    0.6 |                   113 |     4 |     10 |       6 |           -6 |                                67 |                      77 |              32 |           0 |           3 | ct        |   21 |
|  117185 | Boaster       | Omen       |   0.62 |                    72 |     2 |      6 |       3 |           -4 |                                57 |                      39 |               0 |           0 |           2 | t         |   21 |
|  117185 | Brave         | Omen       |    0.5 |                    69 |     2 |      5 |       2 |           -3 |                                71 |                      62 |              67 |           0 |           1 | ct        |   21 |
|  117185 | Brave         | Omen       |   0.86 |                   193 |     9 |      8 |       1 |            1 |                                75 |                     111 |              30 |           2 |           1 | t         |   21 |
|  117185 | Chronicle     | Kayo       |   1.01 |                   216 |     9 |      8 |       6 |            1 |                                75 |                     127 |              22 |           0 |           0 | ct        |   21 |
|  117185 | Chronicle     | Kayo       |   0.97 |                   203 |     5 |      5 |       1 |            0 |                                71 |                     138 |              33 |           1 |           1 | t         |   21 |
|  117185 | Derke         | Jett       |   1.08 |                   275 |    11 |     12 |       1 |           -1 |                                67 |                     170 |              43 |           4 |           3 | ct        |   21 |
|  117185 | Derke         | Jett       |   1.54 |                   375 |    10 |      6 |       1 |            4 |                               100 |                     261 |              46 |           2 |           0 | t         |   21 |
+--------+-------------+------------+--------+-----------------------+-------+--------+---------+--------------+-----------------------------------+-------------------------+-----------------+-------------+-------------+-----------+------+
10 rows in set (0.04 sec)

+----------+
| count(*) |
+----------+
|   249170 |
+----------+
1 row in set (0.08 sec)
```

```
mysql> select * from weapons limit 10; select count(*) from weapons;
+--------------+-----------------+-----------+-----------+---------+-------------------+-----------+------+
| weapon_name | kills_per_match | headshot | bodyshot | legshot | damage_per_round | map_name | tier |
+--------------+-----------------+-----------+-----------+---------+-------------------+-----------+------+
| Ares         |             4.0 |      6.0 |     67.6 |    26.4 |                89 | ascent    |    3 |
| Ares         |             2.7 |      5.9 |     69.4 |    24.7 |                97 | ascent    |    4 |
| Ares         |             3.2 |      6.3 |     74.0 |    19.7 |                95 | ascent    |    5 |
| Ares         |             2.4 |      5.3 |     69.7 |    25.0 |                98 | ascent    |    6 |
| Ares         |             2.7 |      7.6 |     70.6 |    21.8 |                99 | ascent    |    7 |
| Ares         |             2.1 |      7.5 |     72.9 |    19.6 |                92 | ascent    |    8 |
| Ares         |             1.8 |      8.4 |     73.8 |    17.8 |                94 | ascent    |    9 |
| Ares         |             1.3 |      8.1 |     72.3 |    19.6 |                93 | ascent    |   10 |
| Ares         |             1.7 |     13.5 |     71.9 |    14.6 |               102 | ascent    |   11 |
| Ares         |             1.2 |      8.8 |     71.2 |    20.0 |                80 | ascent    |   12 |
+--------------+-----------------+-----------+-----------+---------+-------------------+-----------+------+
10 rows in set (0.03 sec)

+----------+
| count(*) |
+----------+
|     1495 |
+----------+
1 row in set (0.04 sec)
```

```
mysql> select * from agents limit 10; select count(*) from agents;
+------------+------+-------+--------+---------+----------+-----------+------+-------------+-------------+----------+------+-----------+-----------+-----------+----------+
| agent_name | kd   | kills | deaths | assists | win_rate | pick_rate | acs  | first_blood | num_matches | map_name | tier | ability_1 | ability_2 | ability_3 | ultimate |
+------------+------+-------+--------+---------+----------+-----------+------+-------------+-------------+----------+------+-----------+-----------+-----------+----------+
| Astra      | 1.13 |  17.0 |   15.0 |     6.0 |      0.0 |       0.8 |  252 |        11.8 |          51 | ascent   |    3 |       3.3 |      16.5 |       7.0 |      2.3 |
| Astra      | 0.76 |  13.0 |   17.0 |     7.0 |      0.0 |       1.1 |  184 |        23.1 |          51 | ascent   |    4 |       2.4 |      17.4 |       6.9 |      1.9 |
| Astra      | 0.69 |  10.0 |   14.5 |     6.5 |     50.0 |       0.6 |  176 |         5.0 |         102 | ascent   |    5 |       3.1 |      26.2 |       7.5 |      2.2 |
| Astra      | 1.11 |  20.0 |   18.0 |     5.0 |     50.0 |       0.7 |  225 |         2.5 |         102 | ascent   |    6 |       4.8 |      22.5 |       6.4 |      2.2 |
| Astra      | 0.99 |  14.9 |   15.1 |     4.9 |     40.0 |       1.4 |  211 |         8.7 |         510 | ascent   |    7 |       3.4 |      25.7 |       7.4 |      1.9 |
| Astra      | 0.87 |  13.0 |   15.0 |     6.5 |     62.5 |       1.1 |  175 |        14.4 |         408 | ascent   |    8 |       4.7 |      27.4 |       8.6 |      2.1 |
| Astra      | 1.08 |  15.7 |   14.5 |     6.4 |     61.5 |       1.0 |  197 |        14.2 |         663 | ascent   |    9 |       3.7 |      23.9 |       9.9 |      1.9 |
| Astra      | 0.92 |  12.6 |   13.7 |     4.3 |     41.7 |       1.6 |  205 |        11.9 |         612 | ascent   |   10 |       3.8 |      26.3 |      10.8 |      3.0 |
| Astra      | 0.80 |  13.2 |   16.4 |     4.6 |     40.0 |       1.0 |  183 |        10.6 |         255 | ascent   |   11 |       2.8 |      28.3 |       8.8 |      1.8 |
| Astra      | 1.06 |  16.5 |   15.5 |     5.7 |     50.0 |       1.8 |  211 |        11.1 |         306 | ascent   |   12 |       3.7 |      28.4 |      10.7 |      2.1 |
+------------+------+-------+--------+---------+----------+-----------+------+-------------+-------------+----------+------+-----------+-----------+-----------+----------+
10 rows in set (0.04 sec)

+----------+
| count(*) |
+----------+
|     1245 |
+----------+
1 row in set (0.03 sec)
```

```
mysql> select * from maps limit 10; select count(*) from maps;
+----------+-----------+---------+---------+-------------+------+
| map_name | play_rate | atk_win | def_win | num_matches | tier |
+----------+-----------+---------+---------+-------------+------+
| Ascent   |      13.6 |    48.2 |    51.8 |        3927 |    3 |
| Ascent   |      15.5 |    47.0 |    53.0 |        3621 |    4 |
| Ascent   |      15.7 |    46.5 |    53.5 |       11628 |    5 |
| Ascent   |      16.9 |    45.8 |    54.2 |       10149 |    6 |
| Ascent   |      17.0 |    47.8 |    52.2 |       21675 |    7 |
| Ascent   |      17.6 |    47.3 |    52.7 |       22542 |    8 |
| Ascent   |      17.7 |    47.2 |    52.8 |       32946 |    9 |
| Ascent   |      18.6 |    49.0 |    51.0 |       23715 |   10 |
| Ascent   |      17.7 |    48.8 |    51.2 |       16422 |   11 |
| Ascent   |      17.8 |    48.0 |    52.0 |       10557 |   12 |
+----------+-----------+---------+---------+-------------+------+
10 rows in set (0.03 sec)

+----------+
| count(*) |
+----------+
|      111 |
+----------+
1 row in set (0.03 sec)
```

```
mysql> mysql> select games limit 10; select count(*) from games;
+---------+----------+---------------------+
| game_id | map_name | date                |
+---------+----------+---------------------+
|  159497 | Sunset   | 2024-02-12 17:45:00 |
|  159498 | Icebox   | 2024-02-12 17:45:00 |
|  130890 | Lotus    | 2023-06-01 08:30:00 |
|  130891 | Haven    | 2023-06-01 08:30:00 |
|  130892 | Split    | 2023-06-01 08:30:00 |
|  149978 | Haven    | 2023-12-14 17:30:00 |
|  149979 | Lotus    | 2023-12-14 17:30:00 |
|  149980 | Ascent   | 2023-12-14 17:30:00 |
|  171314 | Lotus    | 2024-06-23 08:00:00 |
|  171315 | Bind     | 2024-06-23 08:00:00 |
+---------+----------+---------------------+
10 rows in set (0.04 sec)

+----------+
| count(*) |
+----------+
|    15548 |
+----------+
1 row in set (0.04 sec)
```

Advanced Queries:

1. Query for finding the most commonly played agents given a starting agent (“Reyna” in this case, which represents our user’s most played agent) and our player’s rank. This query would be useful for picking teams because certain agents have synergies with each other. There is currently an issue with this query because it uses “game_id” instead of “team_id” (which is not present in the dataset). This causes the repeat in the output here because two players on different teams can both play Reyna. We will be scraping vlr.gg ourselves to get the team_id and we’ll update the query so it’s correct.

```
mysql> select agent_name from player_stats where game_id in (select game_id from player_stats where agent_name = "Reyna") and tier = 21 group by agent_name order by count(agent_name) desc limit 15;
+------------+
| agent_name |
+------------+
| Viper      |
| Reyna      |
| Jett       |
| Killjoy    |
| Sova       |
| Raze       |
| Omen       |
| Skye       |
| Cypher     |
| Kayo       |
| Gekko      |
| Brimstone  |
| Harbor     |
| Breach     |
| Astra      |
+------------+
15 rows in set (0.29 sec)
```

```
mysql> explain analyze select agent_name from player_stats where game_id in (select game_id from player_stats where agent_name = "Reyna") group by agent_name order by count(agent_name) desc limit 15;
+--------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------+
| EXPLAIN


+--------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=664.360..664.362 rows=15 loops=1)
    -> Sort: count(player_stats.agent_name) DESC, limit input to 15 row(s) per chunk  (actual time=664.359..664.360 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=664.323..664.331 rows=24 loops=1)
            -> Aggregate using temporary table  (actual time=664.320..664.320 rows=24 loops=1)
                -> Nested loop inner join  (cost=576644389.40 rows=5765953624) (actual time=296.892..646.524 rows=16410 loops=1)
                    -> Table scan on player_stats  (cost=25014.65 rows=240124) (actual time=0.137..214.032 rows=249170 loops=1)
                    -> Single-row index lookup on <subquery2> using <auto_distinct_key> (game_id=player_stats.game_id)  (actual time=0.002..0.002 rows=0 loops=249170)
                        -> Materialize with deduplication  (cost=27415.89..27415.89 rows=24012) (actual time=296.267..296.267 rows=824 loops=1)
                            -> Filter: (player_stats.agent_name = 'Reyna')  (cost=25014.65 rows=24012) (actual time=0.408..294.978 rows=1808 loops=1)
                                -> Table scan on player_stats  (cost=25014.65 rows=240124) (actual time=0.079..247.177 rows=249170 loops=1)
 |
+--------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------+
1 row in set (0.77 sec)
```

Initial: Nested loop inner join (cost=57686952.98 rows=576595371)

```
mysql> create index idx_agent_name on player_stats(agent_name);
Query OK, 0 rows affected (2.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze select agent_name from player_stats where game_id in (select game_id from player_stats where agent_name = "Reyna") and tier = 21 group by agent_name order by count(agent_name) desc limit 15;
+--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
| EXPLAIN


+--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
| -> Limit: 15 row(s)  (actual time=27.947..27.948 rows=15 loops=1)
    -> Sort: count(player_stats.agent_name) DESC, limit input to 15 row(s) per chunk  (actual time=27.946..27.947 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=27.920..27.923 rows=24 loops=1)
            -> Aggregate using temporary table  (actual time=27.917..27.917 rows=24 loops=1)
                -> Nested loop inner join  (cost=4450.17 rows=3739) (actual time=0.084..17.495 rows=16410 loops=1)
                    -> Remove duplicates from input sorted on idx_agent_name  (cost=230.64 rows=1808) (actual time=0.058..1.350 rows=824 loops=1)
                        -> Covering index lookup on player_stats using idx_agent_name (agent_name='Reyna')  (cost=230.64 rows=1808) (actual time=0.057..0.992 rows=1808 loops=1)
                    -> Filter: (player_stats.tier = 21)  (cost=480.34 rows=2) (actual time=0.013..0.018 rows=20 loops=824)
                        -> Index lookup on player_stats using PRIMARY (game_id=player_stats.game_id)  (cost=480.34 rows=21) (actual time=0.013..0.016 rows=20 loops=824)
 |
+--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
1 row in set (0.07 sec)
```

index idx_agent_name on player_stats(agent_name); Nested loop inner join (cost=4450.17 rows=3739)

- Significantly reduced cost
- Before index query needed to scan player_stats fully for all instances of Reyna, very expensive due to size of table
- After index: database able to use index to quickly locate all rows with agent name reyna. No need to look through other rows
- Nested loop requiring scanning for agent_name multiple time makes the performance increase substantial

```
mysql> create index idx_tier on player_stats(tier);
Query OK, 0 rows affected (1.62 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze select agent_name from player_stats where game_id in (select game_id from player_stats where agent_name = "Reyna") and tier = 21 group by agent_name order by count(agent_name) desc limit 15;
+--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------+
| EXPLAIN


+--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=1001.951..1001.953 rows=15 loops=1)
    -> Sort: count(player_stats.agent_name) DESC, limit input to 15 row(s) per chunk  (actual time=1001.950..1001.951 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=1001.923..1001.929 rows=24 loops=1)
            -> Aggregate using temporary table  (actual time=1001.919..1001.919 rows=24 loops=1)
                -> Nested loop inner join  (cost=288324700.33 rows=2882976812) (actual time=119.930..989.916 rows=16410 loops=1)
                    -> Index lookup on player_stats using idx_tier (tier=21)  (cost=15012.95 rows=120062) (actual time=0.370..780.265 rows=249170 loops=1)
                    -> Single-row index lookup on <subquery2> using <auto_distinct_key> (game_id=player_stats.game_id)  (actual time=0.001..0.001 rows=0 loops=249170)
                        -> Materialize with deduplication  (cost=27415.89..27415.89 rows=24012) (actual time=118.713..118.713 rows=824 loops=1)
                            -> Filter: (player_stats.agent_name = 'Reyna')  (cost=25014.65 rows=24012) (actual time=0.233..118.151 rows=1808 loops=1)
                                -> Table scan on player_stats  (cost=25014.65 rows=240124) (actual time=0.061..99.705 rows=249170 loops=1)
 |
+--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------+
1 row in set (1.05 sec)
```

index idx_tier on player_stats(tier); Nested loop inner join (cost=288324700.33 rows=2882976812)

- Substantial increase in cost
- Currently all data in the player_stats table is at a single rank
- Indexing caused massive storage use but no real perks since it just indexed the whole table
- Query still checks index table each time due to checking for a single tier
    - Since all entries are at the same tier likely that one bucket is a very long linked list

```
mysql> create index idx_kills on player_stats(kills);
Query OK, 0 rows affected (1.69 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze select agent_name from player_stats where game_id in (select game_id from player_stats where agent_name = "Reyna") and tier = 21 group by agent_name order by count(agent_name) desc limit 15;
+----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
| EXPLAIN


+----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
| -> Limit: 15 row(s)  (actual time=353.714..353.716 rows=15 loops=1)
    -> Sort: count(player_stats.agent_name) DESC, limit input to 15 row(s) per chunk  (actual time=353.714..353.714 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=353.685..353.689 rows=24 loops=1)
            -> Aggregate using temporary table  (actual time=353.683..353.683 rows=24 loops=1)
                -> Nested loop inner join  (cost=57686952.98 rows=576595371) (actual time=120.277..344.612 rows=16410 loops=1)
                    -> Filter: (player_stats.tier = 21)  (cost=25014.65 rows=24012) (actual time=0.065..139.331 rows=249170 loops=1)
                        -> Table scan on player_stats  (cost=25014.65 rows=240124) (actual time=0.063..116.256 rows=249170 loops=1)
                    -> Single-row index lookup on <subquery2> using <auto_distinct_key> (game_id=player_stats.game_id)  (actual time=0.001..0.001 rows=0 loops=249170)
                        -> Materialize with deduplication  (cost=27415.89..27415.89 rows=24012) (actual time=119.776..119.776 rows=824 loops=1)
                            -> Filter: (player_stats.agent_name = 'Reyna')  (cost=25014.65 rows=24012) (actual time=0.241..119.234 rows=1808 loops=1)
                                -> Table scan on player_stats  (cost=25014.65 rows=240124) (actual time=0.035..99.737 rows=249170 loops=1)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
1 row in set (0.39 sec)
```

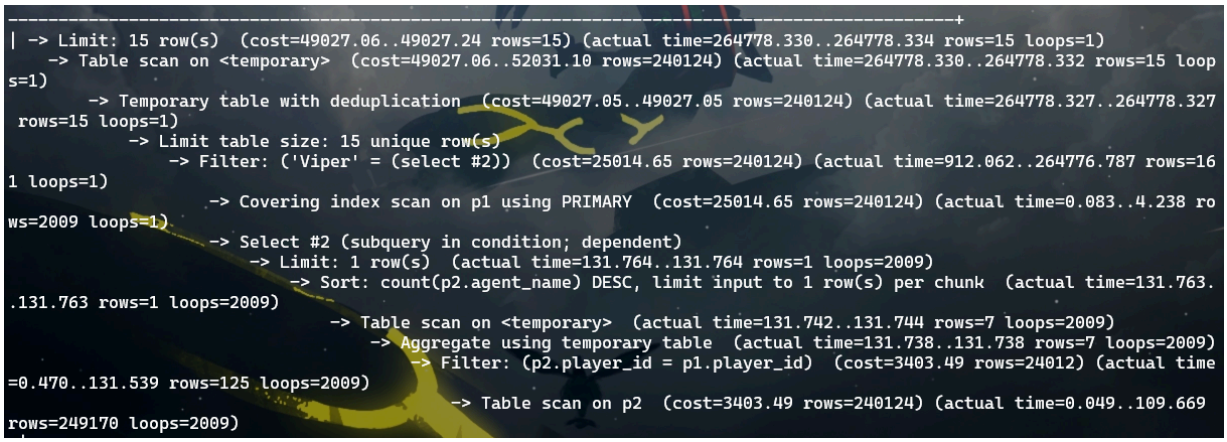
index idx_kills on player_stats(kills); Nested loop inner join (cost=57686952.98 rows=576595371)

- No change in cost
- Kills not used in the query

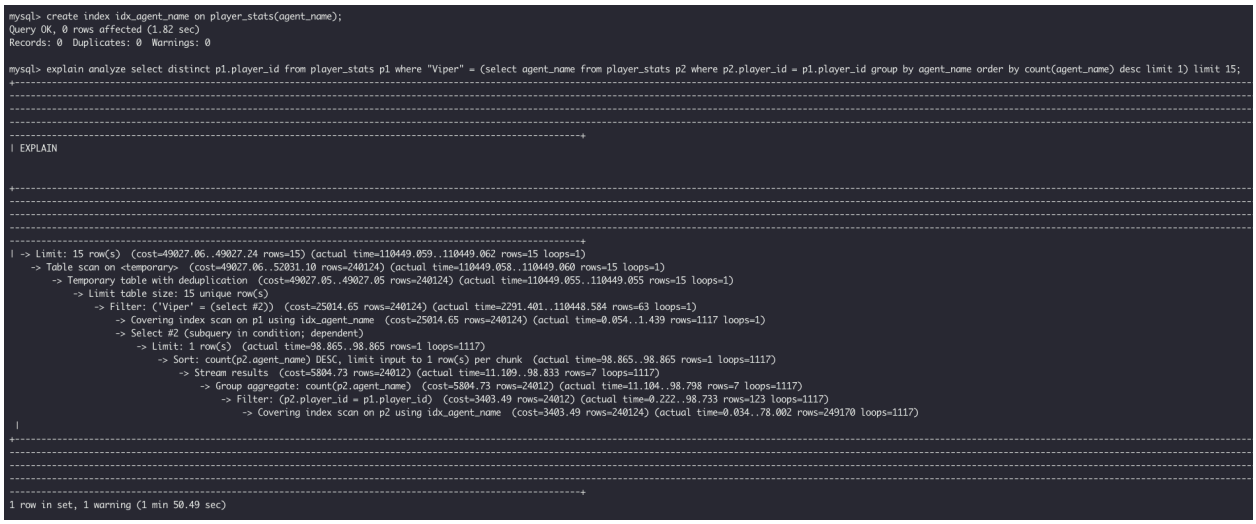
2.) Query can be used to find the most played agent of a particular player to analyze user data. It will also be used in determining a pro-player look alike, as we can match most played agents between a user and pro's and then look to the other stats to further narrow down the match.

```
mysql> select distinct p1.player_id from player_stats p1 where "Viper" = (select agent_name as games_played from player_
stats p2 where p2.player_id = p1.player_id group by agent_name order by count(agent_name) desc limit 1) limit 15;
+------------+
| player_id  |
+------------+
| Chronicle  |
| Mixwell    |
| Wild0reoo  |
| frantastic |
| jeyjey     |
| Karen      |
| mmonch     |
| LiNLiN     |
| bang       |
| mazin      |
| Less       |
| runi       |
| FiNESSE    |
| kiNgg      |
| Marved     |
+------------+
15 rows in set (3 min 52.04 sec)

mysql> █
```



Initial: (cost=49027.06 rows=15)



index idx_agent_name on player_stats(agent_name); (cost=49027.06 rows=15)

- Not sure why this doesn't help
  - Could be due to the query being more dependent on player_id
  - Agent_name field is involved in grouping and so the query will iterate over all of the rows regardless

```
mysql> create index idx_tier on player_stats(tier);
Query OK, 0 rows affected (1.62 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select distinct p1.player_id from player_stats p1 where "Viper" = (select agent_name from player_
    -> stats p2 where p2.player_id = p1.player_id group by agent_name order by count(agent_name) desc limit 1) limit 15;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'p2 w
mysql> explain analyze select distinct p1.player_id from player_stats p1 where "Viper" = (select agent_name from player_stats p2 where p2.player_id = p1.playe
+-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------+
| EXPLAIN


+-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------+
| -> Limit: 15 row(s)  (cost=49027.06..49027.24 rows=15) (actual time=247061.693..247061.699 rows=15 loops=1)
    -> Table scan on <temporary>  (cost=49027.06..52031.10 rows=240124) (actual time=247061.692..247061.696 rows=15 loops=1)
        -> Temporary table with deduplication  (cost=49027.05..49027.05 rows=240124) (actual time=247061.689..247061.689 rows=15 loops=1)
            -> Limit table size: 15 unique row(s)
                -> Filter: ('Viper' = (select #2))  (cost=25014.65 rows=240124) (actual time=886.604..247060.361 rows=161 loops=1)
                    -> Covering index scan on p1 using idx_tier  (cost=25014.65 rows=240124) (actual time=0.050..7.736 rows=2009 loops=1)
                    -> Select #2 (subquery in condition; dependent)
                        -> Limit: 1 row(s)  (actual time=122.949..122.949 rows=1 loops=2009)
                            -> Sort: count(p2.agent_name) DESC, limit input to 1 row(s) per chunk  (actual time=122.949..122.949 rows=1 loops=2009)
                                -> Table scan on <temporary>  (actual time=122.925..122.927 rows=7 loops=2009)
                                    -> Aggregate using temporary table  (actual time=122.922..122.922 rows=7 loops=2009)
                                        -> Filter: (p2.player_id = p1.player_id)  (cost=3403.49 rows=24012) (actual time=0.435..122.749 rows=125 loops=2009)
                                            -> Table scan on p2  (cost=3403.49 rows=240124) (actual time=0.045..102.113 rows=249170 loops=2009)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------+
1 row in set, 1 warning (4 min 7.12 sec)
```

index idx_tier on player_stats(tier); (cost = 49027 rows = 15)

- No notable change in performance:
    - idx_tier is not an attribute we are querying on, so it makes sense to have no effect

```
mysql> create index idx_rating on player_stats(rating);
Query OK, 0 rows affected (1.89 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze select distinct p1.player_id from player_stats p1 where "Viper" = (select agent_name from player_stats p2 where p2.player_id = p1.player
+----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------+
| EXPLAIN


+----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------+
| -> Limit: 15 row(s)  (cost=49027.06..49027.24 rows=15) (actual time=250656.350..250656.354 rows=15 loops=1)
    -> Table scan on <temporary>  (cost=49027.06..52031.10 rows=240124) (actual time=250656.349..250656.351 rows=15 loops=1)
        -> Temporary table with deduplication  (cost=49027.05..49027.05 rows=240124) (actual time=250656.347..250656.347 rows=15 loops=1)
            -> Limit table size: 15 unique row(s)
                -> Filter: ('Viper' = (select #2))  (cost=25014.65 rows=240124) (actual time=883.176..250654.956 rows=161 loops=1)
                    -> Covering index scan on p1 using idx_tier  (cost=25014.65 rows=240124) (actual time=0.048..4.031 rows=2009 loops=1)
                    -> Select #2 (subquery in condition; dependent)
                        -> Limit: 1 row(s)  (actual time=124.741..124.741 rows=1 loops=2009)
                            -> Sort: count(p2.agent_name) DESC, limit input to 1 row(s) per chunk  (actual time=124.740..124.740 rows=1 loops=2009)
                                -> Table scan on <temporary>  (actual time=124.716..124.718 rows=7 loops=2009)
                                    -> Aggregate using temporary table  (actual time=124.712..124.712 rows=7 loops=2009)
                                        -> Filter: (p2.player_id = p1.player_id)  (cost=3403.49 rows=24012) (actual time=0.443..124.537 rows=125 loops=2009)
                                            -> Table scan on p2  (cost=3403.49 rows=240124) (actual time=0.045..102.696 rows=249170 loops=2009)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------+
1 row in set, 1 warning (4 min 10.69 sec)
```

index idx_rating on player_stats(rating);

- Negligible difference in performance:
- Rating is not part of the query

Note: We know the last two attributes used in indexing are not part of the query, however due to the limitation of the query itself, we did this purely to fulfill the requirements.

3.) This query combines the information from Vlr.gg and Blitz.gg together to help make agent recommendations to users. First, we would get the top 5 agents played on a specific map by pros, and then find the win rates of those agents on the specific map for the player's rank. In the screenshot we limited it to 15 instead of 5 due to requirements, but wouldn't do this in practice. It is also worth noting that at some ranks not all of the agents have win rates in Blitz, and not every agent will be used on a particular map in pro play.

```
mysql> select Y.agent_name, Z.win_rate from   (select p.agent_name from player_stats p join
games g where p.game_id = g.game_id and g.map_name = "Split" group by agent_name order by count(agent_name) desc limit 1
5) as Y left join (select a.win_rate, a.agent_name from agents a where a.map_name = "Split" and a.tier = 10) as Z on Y.a
gent_name = Z.agent_name order by Z.
win_rate desc;
+------------+----------+
| agent_name | win_rate |
+------------+----------+
| Skye       |     60.0 |
| Cypher     |     54.3 |
| Raze       |     53.6 |
| Reyna      |     53.6 |
| Sage       |     52.9 |
| Killjoy    |     49.1 |
| Jett       |     48.4 |
| Omen       |     45.0 |
| Viper      |     39.4 |
| Breach     |     35.3 |
| Astra      |      0.0 |
| Fade       |     NULL |
| Kayo       |     NULL |
| Chamber    |     NULL |
| Gekko      |     NULL |
+------------+----------+
15 rows in set (0.08 sec)

mysql>
```

```
mysql> explain analyze select Y.agent_name, Z.win_rate from
    ->  (select p.agent_name from player_stats p join games g where p.game_id = g.game_id and g.map_name = "Split" group by agent_name order by count(agent_name) desc limit 15) as
    -> order by Z.win_rate desc;
+---------------------------------------------------------------------------------------------------------------------------
| EXPLAIN

+---------------------------------------------------------------------------------------------------------------------------
| -> Sort: win_rate DESC  (actual time=53.136..53.137 rows=15 loops=1)
    -> Stream results  (cost=6.25 rows=0) (actual time=53.060..53.122 rows=15 loops=1)
        -> Nested loop left join  (cost=6.25 rows=0) (actual time=53.057..53.114 rows=15 loops=1)
            -> Table scan on Y  (cost=2.50..2.50 rows=0) (actual time=53.035..53.038 rows=15 loops=1)
                -> Materialize  (cost=0.00..0.00 rows=0) (actual time=53.035..53.035 rows=15 loops=1)
                    -> Limit: 15 row(s)  (actual time=53.005..53.008 rows=15 loops=1)
                        -> Sort: count(p.agent_name) DESC, limit input to 15 row(s) per chunk  (actual time=53.005..53.006 rows=15 loops=1)
                            -> Table scan on <temporary>  (actual time=52.978..52.981 rows=24 loops=1)
                                -> Aggregate using temporary table  (actual time=52.976..52.976 rows=24 loops=1)
                                    -> Nested loop inner join  (cost=5217.33 rows=32217) (actual time=0.092..37.194 rows=31396 loops=1)
                                        -> Filter: ((g.map_name = 'Split') and (g.game_id is not null))  (cost=1581.95 rows=1558) (actual time=0.033..10.913 rows=2031 loops=1)
                                            -> Table scan on g  (cost=1581.95 rows=15577) (actual time=0.025..8.960 rows=15548 loops=1)
                                        -> Index lookup on p using PRIMARY (game_id=g.game_id)  (cost=0.27 rows=21) (actual time=0.009..0.011 rows=15 loops=2031)
            -> Single-row index lookup on a using PRIMARY (agent_name=Y.agent_name, map_name='Split', tier=20)  (cost=0.26 rows=1) (actual time=0.005..0.005 rows=1 loops=15)
 |
+---------------------------------------------------------------------------------------------------------------------------
1 row in set (0.09 sec)
```

Initial: Stream results (cost=6.25 rows=0)

```
mysql> create index idx_agent_name on player_stats(agent_name);
Query OK, 0 rows affected (1.81 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze select Y.agent_name, Z.win_rate from
    ->  (select p.agent_name from player_stats p join games g where p.game_id = g.game_id and g.map_name = "Split" group by agent_name order by count(agent_name) desc limit 15) as
    -> order by Z.win_rate desc;
+------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
| EXPLAIN


+------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
| -> Sort: win_rate DESC  (actual time=55.196..55.197 rows=15 loops=1)
    -> Stream results  (cost=6.25 rows=0) (actual time=55.121..55.182 rows=15 loops=1)
        -> Nested loop left join  (cost=6.25 rows=0) (actual time=55.117..55.173 rows=15 loops=1)
            -> Table scan on Y  (cost=2.50..2.50 rows=0) (actual time=55.095..55.098 rows=15 loops=1)
                -> Materialize  (cost=0.00..0.00 rows=0) (actual time=55.094..55.094 rows=15 loops=1)
                    -> Limit: 15 row(s)  (actual time=55.067..55.069 rows=15 loops=1)
                        -> Sort: count(p.agent_name) DESC, limit input to 15 row(s) per chunk  (actual time=55.067..55.068 rows=15 loops=1)
                            -> Table scan on <temporary>  (actual time=55.042..55.045 rows=24 loops=1)
                                -> Aggregate using temporary table  (actual time=55.040..55.040 rows=24 loops=1)
                                    -> Nested loop inner join  (cost=5217.33 rows=32217) (actual time=0.095..38.435 rows=31396 loops=1)
                                        -> Filter: ((g.map_name = 'Split') and (g.game_id is not null))  (cost=1581.95 rows=1558) (actual time=0.053..11.409 rows=2031 loops=1)
                                            -> Table scan on g  (cost=1581.95 rows=15577) (actual time=0.045..9.355 rows=15548 loops=1)
                                        -> Index lookup on p using PRIMARY (game_id=g.game_id)  (cost=0.27 rows=21) (actual time=0.010..0.012 rows=15 loops=2031)
            -> Single-row index lookup on a using PRIMARY (agent_name=Y.agent_name, map_name='Split', tier=20)  (cost=0.26 rows=1) (actual time=0.005..0.005 rows=1 loops=15)
 |
+------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
1 row in set (0.09 sec)
```

index idx_agent_name on player_stats; (cost = 6.25 rows = 0)

- Negligible change in performance
    - Agent_name used in group by and Count(agent_name)
    - This index would not affect the rows traversed

```
mysql> explain analyze select Y.agent_name, Z.win_rate from   (select p.agent_name from player_stats p join games g where p.game_id = g.game_id and g.map_name = "Split" group by agent_name
der by Z.win_rate desc;
+------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN


+------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------+
| -> Sort: win_rate DESC  (actual time=52.247..52.249 rows=15 loops=1)
    -> Stream results  (cost=6.25 rows=0) (actual time=52.172..52.234 rows=15 loops=1)
        -> Nested loop left join  (cost=6.25 rows=0) (actual time=52.169..52.226 rows=15 loops=1)
            -> Table scan on Y  (cost=2.50..2.50 rows=0) (actual time=52.131..52.134 rows=15 loops=1)
                -> Materialize  (cost=0.00..0.00 rows=0) (actual time=52.130..52.130 rows=15 loops=1)
                    -> Limit: 15 row(s)  (actual time=52.100..52.102 rows=15 loops=1)
                        -> Sort: count(p.agent_name) DESC, limit input to 15 row(s) per chunk  (actual time=52.099..52.100 rows=15 loops=1)
                            -> Table scan on <temporary>  (actual time=52.073..52.077 rows=24 loops=1)
                                -> Aggregate using temporary table  (actual time=52.071..52.071 rows=24 loops=1)
                                    -> Nested loop inner join  (cost=5217.33 rows=32217) (actual time=0.077..36.436 rows=31396 loops=1)
                                        -> Filter: ((g.map_name = 'Split') and (g.game_id is not null))  (cost=1581.95 rows=1558) (actual time=0.034..10.863 rows=2031 loops=1)
                                            -> Table scan on g  (cost=1581.95 rows=15577) (actual time=0.025..9.003 rows=15548 loops=1)
                                        -> Index lookup on p using PRIMARY (game_id=g.game_id)  (cost=0.27 rows=21) (actual time=0.009..0.011 rows=15 loops=2031)
            -> Single-row index lookup on a using PRIMARY (agent_name=Y.agent_name, map_name='Split', tier=20)  (cost=0.26 rows=1) (actual time=0.006..0.006 rows=1 loops=15)
 |
+------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.09 sec)
```

index idx_win_rate on agents; (cost = 6.25 rows= 0)

Negligible change in performance

- win_rate too relevant to the query, only being returned at the end and ordering
- Doesn’t effect interior rows

```
mysql> create index idx_map_name on games(map_name);
Query OK, 0 rows affected (0.17 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze select Y.agent_name, Z.win_rate from   (select p.agent_name from player_stats p join games g where p.game_id = g.game_id and g.map_name = "Split" group b
der by Z.win_rate desc;
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| EXPLAIN


+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| -> Sort: win_rate DESC  (actual time=47.852..47.853 rows=15 loops=1)
    -> Stream results  (cost=6.25 rows=0) (actual time=47.765..47.838 rows=15 loops=1)
        -> Nested loop left join  (cost=6.25 rows=0) (actual time=47.761..47.829 rows=15 loops=1)
            -> Table scan on Y  (cost=2.50..2.50 rows=0) (actual time=47.742..47.744 rows=15 loops=1)
                -> Materialize  (cost=0.00..0.00 rows=0) (actual time=47.741..47.741 rows=15 loops=1)
                    -> Limit: 15 row(s)  (actual time=47.713..47.715 rows=15 loops=1)
                        -> Sort: count(p.agent_name) DESC, limit input to 15 row(s) per chunk  (actual time=47.712..47.713 rows=15 loops=1)
                            -> Table scan on <temporary>  (actual time=47.689..47.692 rows=24 loops=1)
                                -> Aggregate using temporary table  (actual time=47.686..47.686 rows=24 loops=1)
                                    -> Nested loop inner join  (cost=5015.82 rows=42006) (actual time=0.063..31.872 rows=31396 loops=1)
                                        -> Filter: (g.game_id is not null)  (cost=275.85 rows=2031) (actual time=0.026..5.875 rows=2031 loops=1)
                                            -> Index lookup on g using idx_map_name (map_name='Split')  (cost=275.85 rows=2031) (actual time=0.025..5.678 rows=2031 loops=1)
                                        -> Index lookup on p using PRIMARY (game_id=g.game_id)  (cost=0.27 rows=21) (actual time=0.009..0.011 rows=15 loops=2031)
            -> Single-row index lookup on a using PRIMARY (agent_name=Y.agent_name, map_name='Split', tier=20)  (cost=0.26 rows=1) (actual time=0.005..0.005 rows=1 loops=15)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
1 row in set (0.09 sec)
```

index idx_map_name on games(map_name); (cost=6.25 rows=0)

- Negligible change in performance
    - Overall result doesn't use map_name
    - Slight increase in nested loop inner join (cost = 5015 decrease from cost= 5217)
    - Due to filter condition looking for map_name = 'Split'

4) This query also will be useful in giving recommendations to players on which agents to play on which maps, returning the top peaking agents in pro games for a particular map.

```
mysql> SELECT a.map_name, a.agent_name, MAX(a.acs) AS MaxACS from agents a JOIN maps m ON a.map_name = m.map_name AND a.
tier =m.tier group by a.map_name, a.agent_name order by a.map_name, MaxACS DESC limit 15;
+----------+------------+--------+
| map_name | agent_name | MaxACS |
+----------+------------+--------+
| ascent   | Raze       |    350 |
| ascent   | Reyna      |    335 |
| ascent   | Yoru       |    319 |
| ascent   | Jett       |    277 |
| ascent   | Viper      |    268 |
| ascent   | Astra      |    252 |
| ascent   | Phoenix    |    250 |
| ascent   | Brimstone  |    234 |
| ascent   | Breach     |    230 |
| ascent   | Killjoy    |    227 |
| ascent   | Omen       |    225 |
| ascent   | Sova       |    225 |
| ascent   | Sage       |    223 |
| ascent   | Skye       |    217 |
| ascent   | Cypher     |    216 |
+----------+------------+--------+
15 rows in set (0.04 sec)

mysql>
```

```
mysql> explain analyze SELECT a.map_name, a.agent_name, MAX(a.acs) AS MaxACS
    -> FROM
    -> agents a
    -> JOIN maps m ON a.map_name = m.map_name AND a.tier =m.tier
    -> GROUP BY
    -> a.map_name,
    -> a.agent_name
    -> ORDER BY
    -> a.map_name,
    -> MaxACS DESC;
+------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------
| EXPLAIN

+------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------
| -> Sort: a.map_name, MaxACS DESC  (actual time=4.072..4.077 rows=75 loops=1)
    -> Table scan on <temporary>  (actual time=4.024..4.034 rows=75 loops=1)
        -> Aggregate using temporary table  (actual time=4.021..4.021 rows=75 loops=1)
            -> Nested loop inner join  (cost=560.99 rows=1570) (actual time=0.128..2.967 rows=1245 loops=1)
                -> Covering index scan on m using PRIMARY  (cost=11.35 rows=111) (actual time=0.043..0.068 rows=111 loops=1)
                -> Index lookup on a using map_name (map_name=m.map_name, tier=m.tier)  (cost=3.55 rows=14) (actual time=0.023..0.025 rows=11 loops=111)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------
1 row in set (0.05 sec)
```

Initial: Nested loop inner join (cost=560.99 rows=1570)

```
mysql> create index idx_agent_name on player_stats(agent_name);
Query OK, 0 rows affected (1.91 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze SELECT a.map_name, a.agent_name, MAX(a.acs) AS MaxACS
    -> FROM
    -> agents a
    -> JOIN maps m ON a.map_name = m.map_name AND a.tier =m.tier
    -> GROUP BY
    -> a.map_name,
    -> a.agent_name
    -> ORDER BY
    -> a.map_name,
    -> MaxACS DESC;
+------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
| EXPLAIN

+------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
| -> Sort: a.map_name, MaxACS DESC  (actual time=4.420..4.425 rows=75 loops=1)
    -> Table scan on <temporary>  (actual time=4.362..4.376 rows=75 loops=1)
        -> Aggregate using temporary table  (actual time=4.360..4.360 rows=75 loops=1)
            -> Nested loop inner join  (cost=560.99 rows=1570) (actual time=0.139..3.086 rows=1245 loops=1)
                -> Covering index scan on m using PRIMARY  (cost=11.35 rows=111) (actual time=0.067..0.095 rows=111 loops=1)
                -> Index lookup on a using map_name (map_name=m.map_name, tier=m.tier)  (cost=3.55 rows=14) (actual time=0.024..0.026 rows=11 loops=111)
 |
+------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
1 row in set (0.05 sec)
```

index idx_agent_name on player_stats(agent_name); Nested loop inner join (cost=560.99 rows=1570)

- Does not affect performance
    - Does not reduce the number of rows we iterate over since we are just aggregating data and not searching for a specific data type

```
mysql> create index idx_tier on maps(tier);
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze SELECT a.map_name, a.agent_name, MAX(a.acs) AS MaxACS  FROM agents a JOIN maps m ON a.map_name = m.map_name AND a.tier =m.tier GROUP BY a.map_name, a.agent_name ORDER BY a.map_name, MaxACS DESC;
+------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
| EXPLAIN

+------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
| -> Sort: a.map_name, MaxACS DESC  (actual time=4.114..4.119 rows=75 loops=1)
    -> Table scan on <temporary>  (actual time=4.065..4.075 rows=75 loops=1)
        -> Aggregate using temporary table  (actual time=4.052..4.052 rows=75 loops=1)
            -> Nested loop inner join  (cost=560.99 rows=1570) (actual time=0.103..2.925 rows=1245 loops=1)
                -> Covering index scan on m using idx_tier  (cost=11.35 rows=111) (actual time=0.024..0.046 rows=111 loops=1)
                -> Index lookup on a using map_name (map_name=m.map_name, tier=m.tier)  (cost=3.55 rows=14) (actual time=0.023..0.025 rows=11 loops=111)
 |
+------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
1 row in set (0.04 sec)
```

index idx_tier on maps(tier); Nested loop inner join (cost=560.99 rows=1570)

- Does not affect performance
    - Does not reduce the number of rows we iterate over since we are just aggregating data and not searching for a specific data type

```
mysql> create index idx_acs on agents(acs);
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze SELECT a.map_name, a.agent_name, MAX(a.acs) AS MaxACS  FROM agents a JOIN maps m ON a.map_name = m.map_name AND a.tier =m.tier GROUP BY a.map_name, a.agent_name ORDER BY a.map_name, MaxACS DESC;
+------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
| EXPLAIN

+------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
| -> Sort: a.map_name, MaxACS DESC  (actual time=4.224..4.229 rows=75 loops=1)
    -> Table scan on <temporary>  (actual time=4.170..4.184 rows=75 loops=1)
        -> Aggregate using temporary table  (actual time=4.167..4.167 rows=75 loops=1)
            -> Nested loop inner join  (cost=560.99 rows=1570) (actual time=0.126..3.001 rows=1245 loops=1)
                -> Covering index scan on m using idx_tier  (cost=11.35 rows=111) (actual time=0.036..0.062 rows=111 loops=1)
                -> Index lookup on a using map_name (map_name=m.map_name, tier=m.tier)  (cost=3.55 rows=14) (actual time=0.024..0.025 rows=11 loops=111)
 |
+------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
1 row in set (0.05 sec)
```

index idx_acs on agents(acs); Nested loop inner join (cost=560.99 rows=1570)

- Does not affect performance

    - Does not reduce the number of rows we iterate over since we are just aggregating data and not searching for a specific data type

Final Index design:

- Indexing on agent_name provides the best performance increase (going from cost=57686952.98 to 4450.17. This is because we are significantly reducing the total number of rows. Remove duplicates from input sorted on idx_agent_name (cost=230.64 rows=1808) (actual time=0.058..1.350 rows=824 loops=1).
- The index on tier is not helpful because our dataset only has tier = 21 right now, but we think it would help if tiers were evenly distributed. Indexing on player kills is not helpful at all and the cost stays the same because the column is not used in the query.