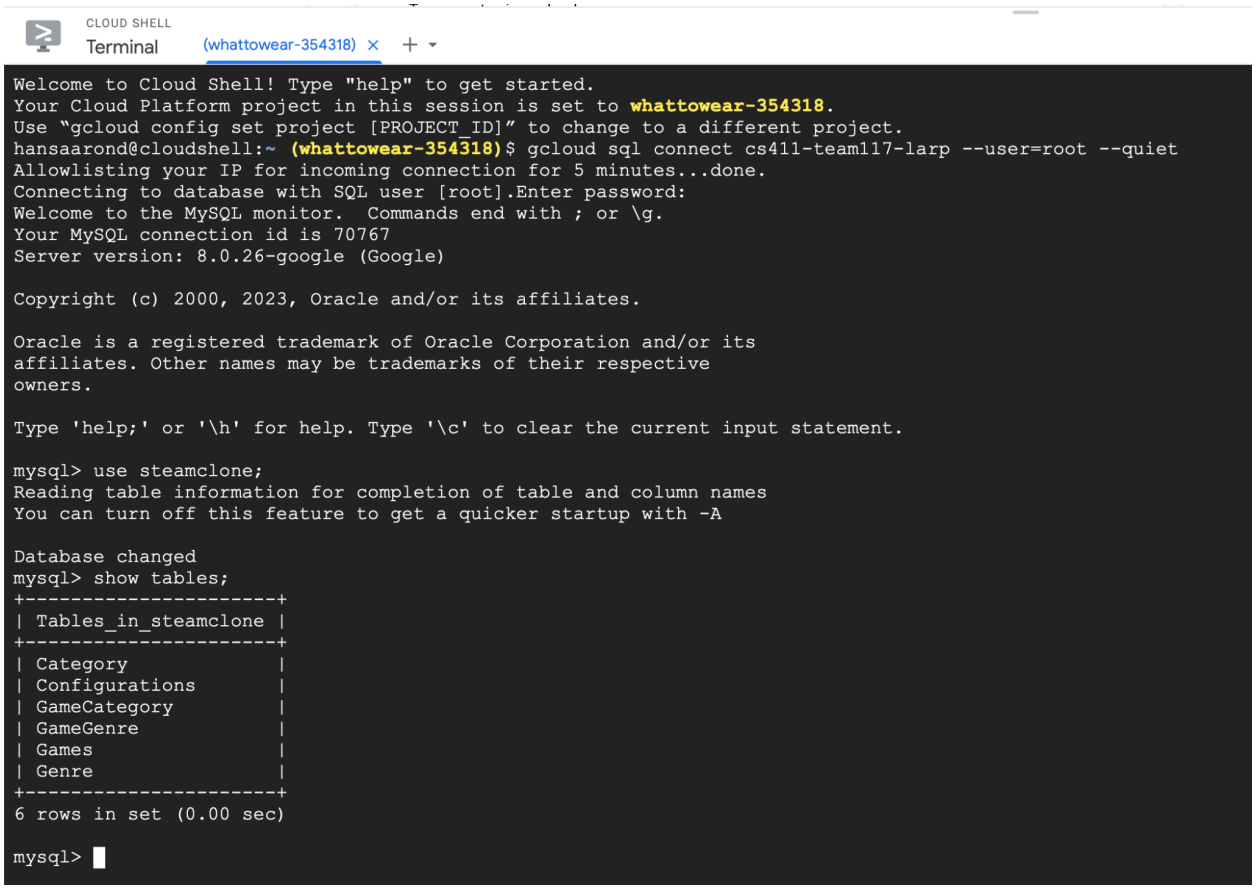
CS 411 Stage 3

Screenshot of the connection:

Database is implemented on GCP.



DDL commands:

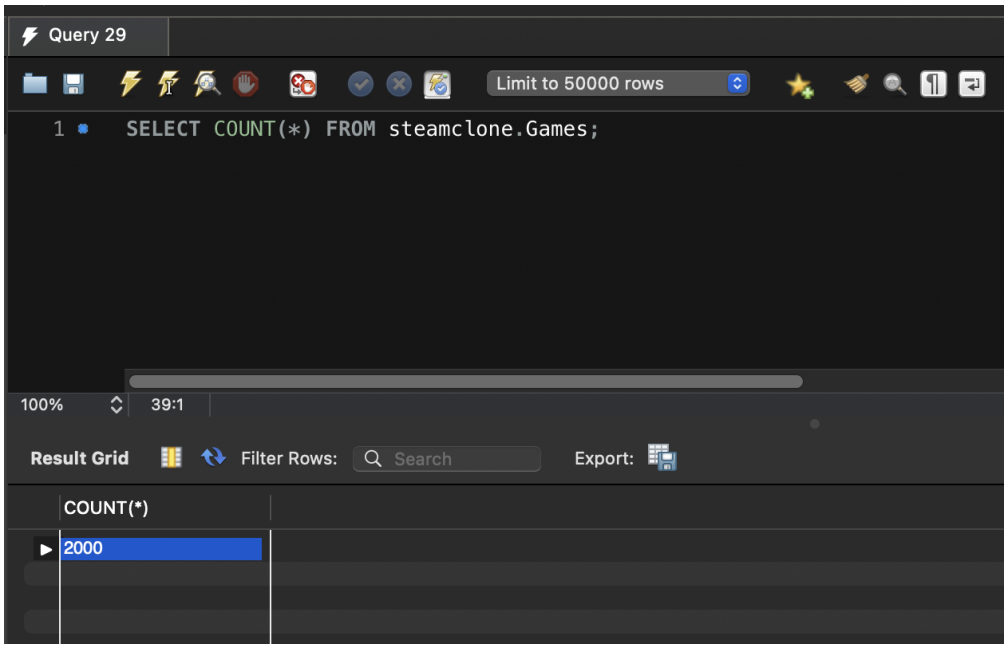
```
CREATE DATABASE steamclone;

CREATE TABLE Genre (
    type VARCHAR(255) NOT NULL,
```

```
    description VARCHAR(255) NOT NULL,
    PRIMARY KEY(type)
);

CREATE TABLE Category (
    type VARCHAR(255) NOT NULL PRIMARY KEY,
    description VARCHAR(255) NOT NULL
);

CREATE TABLE Configurations (
     configurationID INT NOT NULL,
     minRequirementPC VARCHAR(2545),
     recRequirementPC VARCHAR(2545),
     minRequirementMac VARCHAR(2545),
     recRequirementMac VARCHAR(2545),
     minRequirementLinux VARCHAR(2545),
     recRequirementLinux VARCHAR(2545),
     PRIMARY KEY(configurationID)
);

CREATE TABLE Games (
    gameName VARCHAR(255) NOT NULL PRIMARY KEY,
    releaseDate VARCHAR(255) NOT NULL,
    requiredAge INT NOT NULL,
    metacriticScore DECIMAL(10,4) NOT NULL,
    steamSpyNumOwners INT NOT NULL,
    steamSpyNumPlayers INT NOT NULL,
    website VARCHAR(255) NOT NULL,
    shortDescription VARCHAR(2051) NOT NULL,
    detailedDescription TEXT NOT NULL,
    configurationID INT NOT NULL,
    supportEmail VARCHAR(255) NOT NULL,
    supportURL VARCHAR(255) NOT NULL,
    currencyPrice VARCHAR(20) NOT NULL,
    initialPrice DECIMAL(10,4) NOT NULL,
```

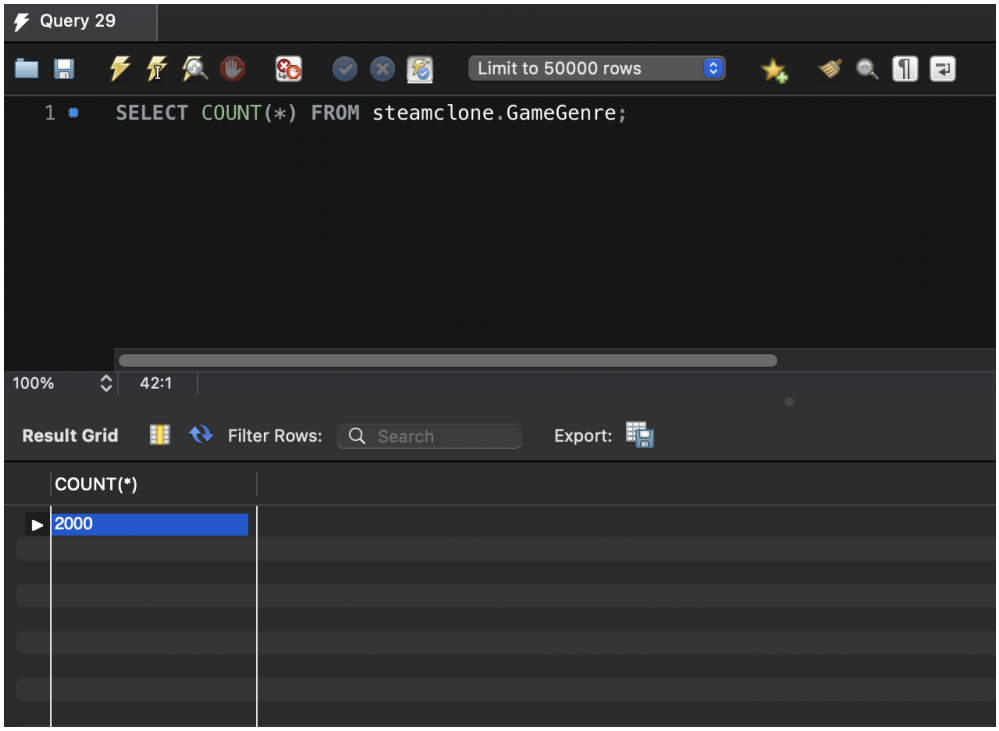
```
    finalPrice DECIMAL(10,4) NOT NULL,
    supportedLanguages VARCHAR(307) NOT NULL,
    image VARCHAR(255) NOT NULL,
    FOREIGN KEY(configurationID) REFERENCES
Configurations(configurationID)
    ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE GameGenre (
    gameName VARCHAR(255),
    gameGenre VARCHAR(255),
    FOREIGN KEY (gameName) REFERENCES Games(gameName)
      ON UPDATE CASCADE
      ON DELETE CASCADE,
    FOREIGN KEY (gameGenre) REFERENCES Genre(type)
      ON UPDATE CASCADE
      ON DELETE CASCADE,
    PRIMARY KEY(gameName, gameGenre)
);

CREATE TABLE GameCategory (
    gameName VARCHAR(255) NOT NULL,
      gameCategory VARCHAR(255) NOT NULL,
    PRIMARY KEY(gameName, gameCategory),
    FOREIGN KEY (gameName) REFERENCES Games(gameName)
           ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (gameCategory) REFERENCES Category(type)
           ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

Insert 1000+ Rows in At Least Three Tables:

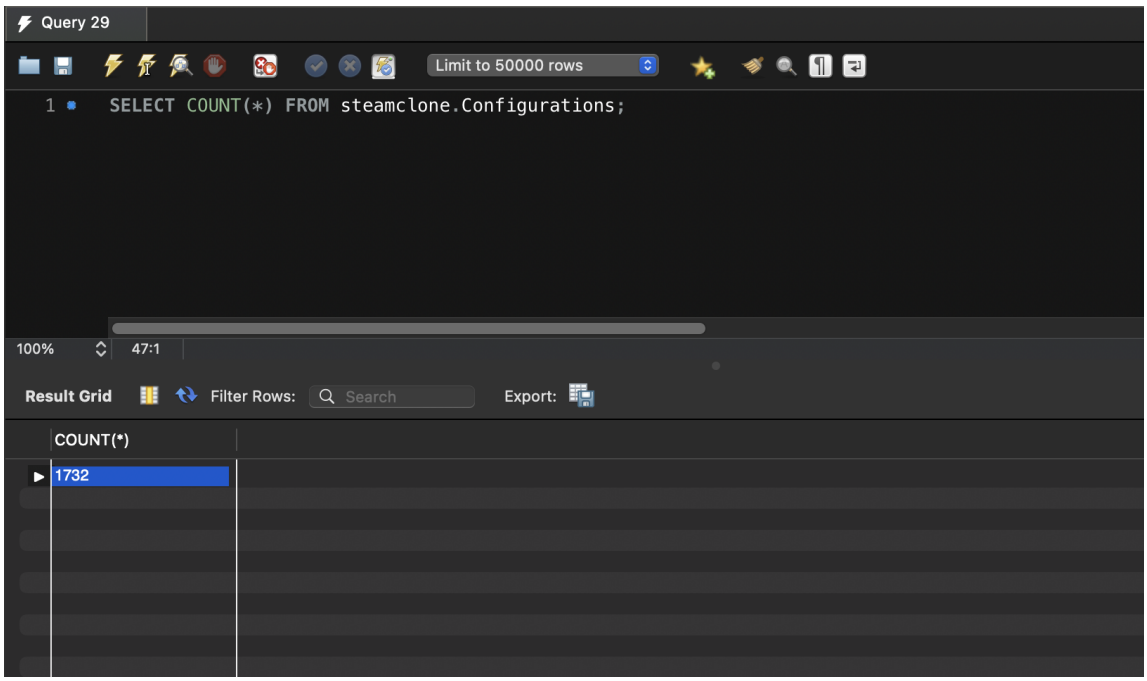
Games table (2000 rows)



GameGenre table (2000 rows)



Configuration table (1732 rows)



2 advanced queries

First query

Return the game name, website, genre, and initial price of games that share the same website as the game Bonanza Bros, have steam spy number of players greater than 10000 (inclusive), and have initial price less than 10 (not inclusive):

```
SELECT g.gameName, g.website, ge.gameGenre, g.initialPrice
FROM Games g JOIN GameGenre ge ON g.gameName = ge.gameName
WHERE website = (SELECT website
                 FROM Games
                 WHERE gameName = "Bonanza Bros")
      AND g.steamSpyNumPlayers >= 10000
  AND g.initialPrice < 10
```

Result only has 4 rows:

```
mysql> SELECT g.gameName, g.website, ge.gameGenre, g.initialPrice FROM Games g JOIN GameGenre g
e ON g.gameName = ge.gameName WHERE website = (SELECT website FROM Games WHERE gameName = "Bona
nza Bros") AND g.steamSpyNumPlayers >= 10000     AND g.initialPrice < 10;
+------------------------+--------------------+------------+--------------+
| gameName               | website            | gameGenre  | initialPrice |
+------------------------+--------------------+------------+--------------+
| Crazy Taxi             | http://www.sega.com | Action     |       7.9900 |
| SEGA Bass Fishing      | http://www.sega.com | Simulation |       7.9900 |
| Sonic Adventure DX     | http://www.sega.com | Action     |       7.9900 |
| Space Channel 5: Part 2 | http://www.sega.com | Action     |       7.9900 |
+------------------------+--------------------+------------+--------------+
4 rows in set (0.01 sec)
```

Second query

Return the game name and PC minimum and recommended requirements of all games that have the same PC requirements as Counter-Strike, ordered by game name in ascending order:

```
SELECT g.gameName, c.minRequirementPC, c.recRequirementPC
FROM Games g JOIN Configurations c ON g.configurationID = c.configurationID
WHERE c.minRequirementPC = (SELECT c2.minRequirementPC
                                                    FROM Games g2 JOIN Configurations c2
ON g2.configurationID = c2.configurationID
```

WHERE g2.gameName = "Counter-Strike")
ORDER BY g.gameName

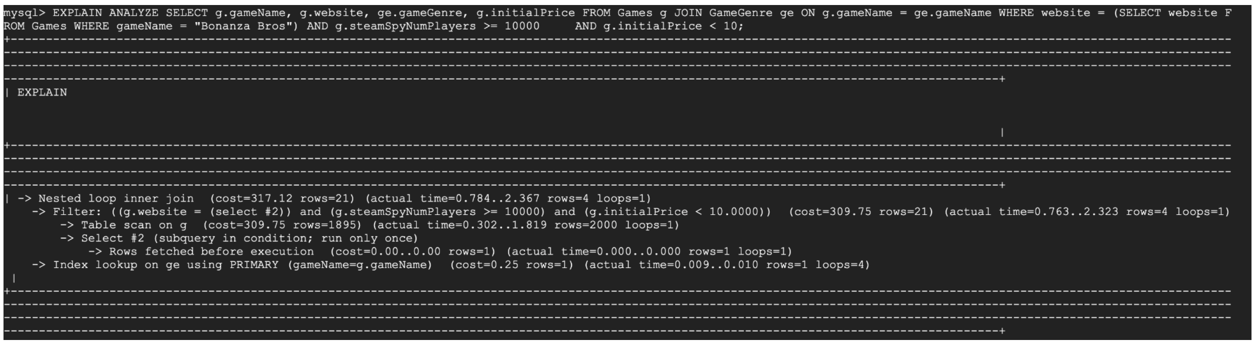
Result only has 10 rows:

```
mysql> SELECT g.gameName, c.minRequirementPC, c.recRequirementPC FROM Games g JOIN Configurations c ON g.configurati
onID = c.configurationID WHERE c.minRequirementPC = (SELECT c2.minRequirementPC  FROM Games g2 JOIN Configurations c
2 ON g2.configurationID = c2.configurationID                          WHERE g2.gameName = "Counter-Strike") ORDER
 BY g.gameName;
+-----------------------------------------+-----------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------
------------------+------------------+
| gameName                                | minRequirementPC  | recRequirementPC |
+-----------------------------------------+-----------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------
------------------+------------------+
| Counter-Strike                          | Minimum: 500 mhz processor 96mb ram 16mb video card Windows XP Mou
se Keyboard Internet ConnectionRecommended: 800 mhz processor 128mb ram 32mb+ video card Windows XP Mouse Keyboard I
nternet Connection |                  |
| Counter-Strike: Condition Zero          | Minimum: 500 mhz processor 96mb ram 16mb video card Windows XP Mou
se Keyboard Internet ConnectionRecommended: 800 mhz processor 128mb ram 32mb+ video card Windows XP Mouse Keyboard I
nternet Connection |                  |
| Counter-Strike: Condition Zero Deleted Scenes | Minimum: 500 mhz processor 96mb ram 16mb video card Windows XP Mou
se Keyboard Internet ConnectionRecommended: 800 mhz processor 128mb ram 32mb+ video card Windows XP Mouse Keyboard I
nternet Connection |                  |
| Day of Defeat                           | Minimum: 500 mhz processor 96mb ram 16mb video card Windows XP Mou
se Keyboard Internet ConnectionRecommended: 800 mhz processor 128mb ram 32mb+ video card Windows XP Mouse Keyboard I
nternet Connection |                  |
| Deathmatch Classic                      | Minimum: 500 mhz processor 96mb ram 16mb video card Windows XP Mou
se Keyboard Internet ConnectionRecommended: 800 mhz processor 128mb ram 32mb+ video card Windows XP Mouse Keyboard I
nternet Connection |                  |
| Half-Life                               | Minimum: 500 mhz processor 96mb ram 16mb video card Windows XP Mou
se Keyboard Internet ConnectionRecommended: 800 mhz processor 128mb ram 32mb+ video card Windows XP Mouse Keyboard I
nternet Connection |                  |
| Half-Life: Blue Shift                   | Minimum: 500 mhz processor 96mb ram 16mb video card Windows XP Mou
se Keyboard Internet ConnectionRecommended: 800 mhz processor 128mb ram 32mb+ video card Windows XP Mouse Keyboard I
nternet Connection |                  |
| Half-Life: Opposing Force               | Minimum: 500 mhz processor 96mb ram 16mb video card Windows XP Mou
se Keyboard Internet ConnectionRecommended: 800 mhz processor 128mb ram 32mb+ video card Windows XP Mouse Keyboard I
nternet Connection |                  |
| Ricochet                                | Minimum: 500 mhz processor 96mb ram 16mb video card Windows XP Mou
se Keyboard Internet ConnectionRecommended: 800 mhz processor 128mb ram 32mb+ video card Windows XP Mouse Keyboard I
nternet Connection |                  |
| Team Fortress Classic                   | Minimum: 500 mhz processor 96mb ram 16mb video card Windows XP Mou
se Keyboard Internet ConnectionRecommended: 800 mhz processor 128mb ram 32mb+ video card Windows XP Mouse Keyboard I
nternet Connection |                  |
+-----------------------------------------+-----------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------
------------------+------------------+
10 rows in set (0.01 sec)
```

Indexing analysis:

First query

1. Before adding indices:



```
mysql> EXPLAIN ANALYZE SELECT g.gameName, g.website, ge.gameGenre, g.initialPrice FROM Games g JOIN GameGenre ge ON g.gameName = ge.gameName WHERE website = (SELECT website F
ROM Games WHERE gameName = "Bonanza Bros") AND g.steamSpyNumPlayers >= 10000     AND g.initialPrice < 10;
+-----------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                       |
+-----------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=317.12 rows=21) (actual time=0.784..2.367 rows=4 loops=1)
    -> Filter: ((g.website = (select #2)) and (g.steamSpyNumPlayers >= 10000) and (g.initialPrice < 10.0000))  (cost=309.75 rows=21) (actual time=0.763..2.323 rows=4 loops=1)
        -> Table scan on g  (cost=309.75 rows=1895) (actual time=0.302..1.819 rows=2000 loops=1)
        -> Select #2 (subquery in condition; run only once)
            -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)
    -> Index lookup on ge using PRIMARY (gameName=g.gameName)  (cost=0.25 rows=1) (actual time=0.009..0.010 rows=1 loops=4)
 |
+-----------------------------------------------------------------------------------------------+
```

| -> Nested loop inner join (cost=317.12 rows=21) (actual time=0.452..1.929 rows=4 loops=1)
    -> Filter: ((g.website = (select #2)) and (g.steamSpyNumPlayers >= 10000) and (g.initialPrice < 10.0000)) (cost=309.75 rows=21) (actual time=0.438..1.895 rows=4 loops=1)
        -> Table scan on g (cost=309.75 rows=1895) (actual time=0.023..1.371 rows=2000 loops=1)
        -> Select #2 (subquery in condition; run only once)
            -> Rows fetched before execution (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)
    -> Index lookup on ge using PRIMARY (gameName=g.gameName) (cost=0.25 rows=1) (actual time=0.007..0.008 rows=1 loops=4) |

The total cost of the query is 317.12. Notice that most of this comes from the filter on the website, steamSpyNumPlayers, and initialPrice, which is done by a table scan on Games and costs 309.75.

2. After adding 3 separate different indices:
    a. CREATE INDEX idx_website on Games(website(20));

```
mysql> EXPLAIN ANALYZE SELECT g.gameName, g.website, ge.gameGenre, g.initialPrice FROM Games g JOIN GameGenre ge ON g.gameName = ge.gameName WHERE website = (SELECT website F
ROM Games WHERE gameName = "Bonanza Bros") AND g.steamSpyNumPlayers >= 10000     AND g.initialPrice < 10;
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
+
| EXPLAIN



|
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
+
| -> Nested loop inner join  (cost=15.60 rows=6) (actual time=0.208..0.382 rows=4 loops=1)
    -> Filter: ((g.website = (select #2)) and (g.steamSpyNumPlayers >= 10000) and (g.initialPrice < 10.0000))  (cost=13.58 rows=6) (actual time=0.199..0.357 rows=4 loops=1)
        -> Index lookup on g using idx_website (website=(select #2))  (cost=13.58 rows=52) (actual time=0.134..0.331 rows=52 loops=1)
        -> Select #2 (subquery in condition; run only once)
            -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)
    -> Index lookup on ge using PRIMARY (gameName=g.gameName)  (cost=0.27 rows=1) (actual time=0.005..0.005 rows=1 loops=4)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

| -> Nested loop inner join (cost=15.60 rows=6) (actual time=0.208..0.382 rows=4 loops=1)
-> Filter: ((g.website = (select #2)) and (g.steamSpyNumPlayers >= 10000) and (g.initialPrice < 10.0000)) (cost=13.58 rows=6) (actual time=0.199..0.357 rows=4 loops=1)
-> Index lookup on g using idx_website (website=(select #2)) (cost=13.58 rows=52) (actual time=0.134..0.331 rows=52 loops=1)
-> Select #2 (subquery in condition; run only once)
-> Rows fetched before execution (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)
-> Index lookup on ge using PRIMARY (gameName=g.gameName) (cost=0.27 rows=1) (actual time=0.005..0.005 rows=1 loops=4) |

With an index on website(20) within Games, the total cost of the query drops significantly from 317.12 to 15.6. Notice that the previous table scan on Games that costs 309.75 was improved by an index lookup on Games by the defined index on website(20). This index lookup costs 13.58, significantly better than 309.75 for the table scan.

b. CREATE INDEX idx_NumPlayers on Games(steamSpyNumPlayers);

```
mysql> EXPLAIN ANALYZE SELECT g.gameName, g.website, ge.gameGenre, g.initialPrice FROM Games g JOIN GameGenre ge ON g.gameName = ge.gameName WHERE website = (SELECT website F
ROM Games WHERE gameName = "Bonanza Bros") AND g.steamSpyNumPlayers >= 10000     AND g.initialPrice < 10;
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN


                                                                                                                                                                      |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=327.95 rows=52) (actual time=0.462..1.902 rows=4 loops=1)
    -> Filter: ((g.website = (select #2)) and (g.steamSpyNumPlayers >= 10000) and (g.initialPrice < 10.0000))  (cost=309.75 rows=52) (actual time=0.448..1.867 rows=4 loops=1)
        -> Table scan on g  (cost=309.75 rows=1895) (actual time=0.021..1.370 rows=2000 loops=1)
        -> Select #2 (subquery in condition; run only once)
            -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)
    -> Index lookup on ge using PRIMARY (gameName=g.gameName)  (cost=0.25 rows=1) (actual time=0.007..0.008 rows=1 loops=4)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
```

| -> Nested loop inner join (cost=327.95 rows=52) (actual time=0.462..1.902 rows=4 loops=1)

-> Filter: ((g.website = (select #2)) and (g.steamSpyNumPlayers >= 10000) and (g.initialPrice < 10.0000)) (cost=309.75 rows=52) (actual time=0.448..1.867 rows=4 loops=1)
-> Table scan on g (cost=309.75 rows=1895) (actual time=0.021..1.370 rows=2000 loops=1)
-> Select #2 (subquery in condition; run only once)
-> Rows fetched before execution (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)
-> Index lookup on ge using PRIMARY (gameName=g.gameName) (cost=0.25 rows=1) (actual time=0.007..0.008 rows=1 loops=4)
|

With an index on steamSpyNumPlayers within Games, the total cost of the query becomes slightly worse from 317.12 without indexing to 327.95 with indexing. Notice that the previous table scan on Games that costs 309.75 is still a table scan. This shows that the index on steamSpyNumPlayers is not used, despite the query having a WHERE clause on the number of players, and so the index on steamSpyNumPlayers is not useful, and may even hamper performance due to the memory and performance hit of an index.

c. CREATE INDEX idx_initialPrice on Games(initialPrice);

```
mysql> EXPLAIN ANALYZE SELECT g.gameName, g.website, ge.gameGenre, g.initialPrice FROM Games g JOIN GameGenre ge ON g.gameName = ge.gameName WHERE website = (SELECT website F
ROM Games WHERE gameName = "Bonanza Bros") AND g.steamSpyNumPlayers >= 10000     AND g.initialPrice < 10;
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                                                                                                     |
                                                                                                                                                                              |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=326.44 rows=48) (actual time=0.492..2.167 rows=4 loops=1)
    -> Filter: ((g.website = (select #2)) and (g.steamSpyNumPlayers >= 10000) and (g.initialPrice < 10.0000))  (cost=309.75 rows=48) (actual time=0.479..2.130 rows=4 loops=1)
        -> Table scan on g  (cost=309.75 rows=1895) (actual time=0.027..1.631 rows=2000 loops=1)
        -> Select #2 (subquery in condition; run only once)
            -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)
    -> Index lookup on ge using PRIMARY (gameName=g.gameName)  (cost=0.25 rows=1) (actual time=0.007..0.008 rows=1 loops=4)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
```

| -> Nested loop inner join (cost=326.44 rows=48) (actual time=0.492..2.167 rows=4 loops=1)
-> Filter: ((g.website = (select #2)) and (g.steamSpyNumPlayers >= 10000) and (g.initialPrice < 10.0000)) (cost=309.75 rows=48) (actual time=0.479..2.130 rows=4 loops=1)
-> Table scan on g (cost=309.75 rows=1895) (actual time=0.027..1.631 rows=2000 loops=1)
-> Select #2 (subquery in condition; run only once)
-> Rows fetched before execution (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)

-> Index lookup on ge using PRIMARY (gameName=g.gameName) (cost=0.25 rows=1) (actual time=0.007..0.008 rows=1 loops=4)
|

With an index on initialPrice within Games, the total cost of the query becomes slightly worse from 317.12 without indexing to 326.44 with indexing. Again notice that the previous table scan on Games that costs 309.75 is still a table scan despite the index. This shows that the index on initialPrice is not used, even though the query has a WHERE clause on initialPrice. This suggests the index on initialPrice is not useful and may even hamper performance due to the memory and performance hit of an index.

3. We chose to use the indexing on website(20) within the Games table as this results in the best possible performance gain out of the 3 indexing choices tested. Indexing on website(20) results in the nested inner loop join cost decreasing significantly. Without indexing, this costs 317.12. After indexing on website(20), the cost significantly decreases to 15.6. The other indexing options of indexing on steamSpyNumPlayers and initialPrice resulted in the nested inner loop join cost slightly increasing to 327.95 and 326.44, respectively. This is because neither indexes on steamSpyNumPlayers and initialPrice are used by the query, as seen in the explain analyze result showing a table scan on the Games table. In comparison, the index on website(20) allows this table scan to be optimized by an index lookup on Games. So, we decided to index on Games on website(20) as it provides the best performance for this query out of the index choices tested.

Second query

1. EXPLAIN ANALYZE before adding indices:

```
mysql> EXPLAIN ANALYZE SELECT g.gameName, c.minRequirementPC, c.recRequirementPC FROM Games g JOIN Configurations c ON g.configurationID = c.configurationID WHERE c.minRequirem
entPC = (SELECT c2.minRequirementPC         FROM Games g2 JOIN Configurations c2 ON g2.configurationID = c2.configurationID                              WHERE g2.gameName = "Cou
nter-Strike") ORDER BY g.gameName;
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                                                                                   |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------+
| -> Sort: g.gameName  (actual time=1.140..1.141 rows=10 loops=1)
    -> Stream results  (cost=238.86 rows=176) (actual time=0.045..1.113 rows=10 loops=1)
        -> Nested loop inner join  (cost=238.86 rows=176) (actual time=0.042..1.104 rows=10 loops=1)
            -> Filter: (c.minRequirementPC = (select #2))  (cost=181.35 rows=157) (actual time=0.030..1.083 rows=2 loops=1)
                -> Table scan on c  (cost=181.35 rows=1571) (actual time=0.022..0.772 rows=1732 loops=1)
                -> Select #2 (subquery in condition; run only once)
                    -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)
            -> Index lookup on g using configurationID (configurationID=c.configurationID)  (cost=0.25 rows=1) (actual time=0.007..0.009 rows=5 loops=2)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

| -> Sort: g.gameName (actual time=1.140..1.141 rows=10 loops=1)
    -> Stream results (cost=238.86 rows=176) (actual time=0.045..1.113 rows=10 loops=1)
        -> Nested loop inner join (cost=238.86 rows=176) (actual time=0.042..1.104 rows=10 loops=1)
            -> Filter: (c.minRequirementPC = (select #2)) (cost=181.35 rows=157) (actual time=0.030..1.083 rows=2 loops=1)
                -> Table scan on c (cost=181.35 rows=1571) (actual time=0.022..0.772 rows=1732 loops=1)
                -> Select #2 (subquery in condition; run only once)
                    -> Rows fetched before execution (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)
            -> Index lookup on g using configurationID (configurationID=c.configurationID) (cost=0.25 rows=1) (actual time=0.007..0.009 rows=5 loops=2)
 |

Notice that most of the total cost of 238.86 comes from the filter for minRequirementPC within the Configurations table being equal to the minRequirementPC for Counter-Strike, as this has a cost of 181.35 for a table scan on the Configurations table. Notice also there is a default index within the Games table for configurationID, as configurationID is a foreign key. This results in the cost for the lookup on Games for a specific configurationID being relatively low at 0.25 using the index.

2. After adding 3 separate different indices:
    a. CREATE INDEX pcRecReq ON Configurations(recRequirementPC(20));

```
mysql> EXPLAIN ANALYZE SELECT g.gameName, c.minRequirementPC, c.recRequirementPC FROM Games g JOIN Configurations c ON g.configurationID = c.configurationID WHERE c.minRequirem
entPC = (SELECT c2.minRequirementPC        FROM Games g2 JOIN Configurations c2 ON g2.configurationID = c2.configurationID                          WHERE g2.gameName = "Cou
nter-Strike") ORDER BY g.gameName;
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                                                                                                      |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Sort: g.gameName  (actual time=1.307..1.308 rows=10 loops=1)
    -> Stream results  (cost=238.86 rows=176) (actual time=0.066..1.287 rows=10 loops=1)
        -> Nested loop inner join  (cost=238.86 rows=176) (actual time=0.056..1.271 rows=10 loops=1)
            -> Filter: (c.minRequirementPC = (select #2))  (cost=181.35 rows=157) (actual time=0.042..1.248 rows=2 loops=1)
                -> Table scan on c  (cost=181.35 rows=1571) (actual time=0.033..0.922 rows=1732 loops=1)
                -> Select #2 (subquery in condition; run only once)
                    -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)
            -> Index lookup on g using configurationID (configurationID=c.configurationID)  (cost=0.25 rows=1) (actual time=0.007..0.010 rows=5 loops=2)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.01 sec)
```

| -> Sort: g.gameName (actual time=1.307..1.308 rows=10 loops=1)
    -> Stream results (cost=238.86 rows=176) (actual time=0.066..1.287 rows=10 loops=1)

-> Nested loop inner join (cost=238.86 rows=176) (actual time=0.056..1.271 rows=10 loops=1)
-> Filter: (c.minRequirementPC = (select #2)) (cost=181.35 rows=157) (actual time=0.042..1.248 rows=2 loops=1)
-> Table scan on c (cost=181.35 rows=1571) (actual time=0.033..0.922 rows=1732 loops=1)
-> Select #2 (subquery in condition; run only once)
-> Rows fetched before execution (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)
-> Index lookup on g using configurationID (configurationID=c.configurationID) (cost=0.25 rows=1) (actual time=0.007..0.010 rows=5 loops=2)
|

Here we see that the index on recRequirementPC is not actually used. This is because recRequirementPC is only used within the SELECT clause. The earlier EXPLAIN ANALYZE within indices also suggested that there are no scans based on recRequirementPC, and so we expect the cost to remain the same with or without an index on recRequirementPC. This is indeed the case, with the total cost of the query remaining at 238.86.

b. CREATE INDEX pcMinReq ON Configurations(minRequirementPC(20));

```
mysql> EXPLAIN ANALYZE SELECT g.gameName, c.minRequirementPC, c.recRequirementPC FROM Games g JOIN Configurations c ON g.configurationID = c.configurationID WHERE c.minRequirem
entPC = (SELECT c2.minRequirementPC         FROM Games g2 JOIN Configurations c2 ON g2.configurationID = c2.configurationID                                WHERE g2.gameName = "Cou
nter-Strike") ORDER BY g.gameName;
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                                                                                                     |
                                                                                                                                                                              |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Sort: g.gameName  (actual time=0.064..0.065 rows=10 loops=1)
    -> Stream results  (cost=1.43 rows=2) (actual time=0.033..0.051 rows=10 loops=1)
        -> Nested loop inner join  (cost=1.43 rows=2) (actual time=0.031..0.042 rows=10 loops=1)
            -> Filter: (c.minRequirementPC = (select #2))  (cost=0.70 rows=2) (actual time=0.019..0.023 rows=2 loops=1)
                -> Index lookup on c using pcMinReq (minRequirementPC=(select #2))  (cost=0.70 rows=2) (actual time=0.016..0.017 rows=2 loops=1)
                -> Select #2 (subquery in condition; run only once)
                    -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)
            -> Index lookup on g using configurationID (configurationID=c.configurationID)  (cost=0.31 rows=1) (actual time=0.006..0.008 rows=5 loops=2)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

| -> Sort: g.gameName (actual time=0.064..0.065 rows=10 loops=1)
-> Stream results (cost=1.43 rows=2) (actual time=0.033..0.051 rows=10 loops=1)
-> Nested loop inner join (cost=1.43 rows=2) (actual time=0.031..0.042 rows=10 loops=1)
-> Filter: (c.minRequirementPC = (select #2)) (cost=0.70 rows=2) (actual time=0.019..0.023 rows=2 loops=1)

-> Index lookup on c using pcMinReq (minRequirementPC=(select #2)) (cost=0.70 rows=2) (actual time=0.016..0.017 rows=2 loops=1)
-> Select #2 (subquery in condition; run only once)
-> Rows fetched before execution (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)
-> Index lookup on g using configurationID (configurationID=c.configurationID) (cost=0.31 rows=1) (actual time=0.006..0.008 rows=5 loops=2)
|

Here we see that the cost of the total query significantly goes down from 238.86 without indexing, to 1.43 with indexing on minRequirementPC within the Configurations table. This is because now, the filter for minRequirementPC within the Configurations table being equal to the minRequirementPC for Counter-Strike no longer needs a table scan, but now uses an index lookup using the index on minRequirementPC. This results in the actual filter having a cost of 0.70 using the index, compared to 181.35 without it.

c. CREATE INDEX pcMinRecReq ON Configurations(minRequirementPC(20), recRequirementPC(20));

```
mysql> EXPLAIN ANALYZE SELECT g.gameName, c.minRequirementPC, c.recRequirementPC FROM Games g JOIN Configurations c ON g.configurationID = c.configurationID WHERE c.minRequirem
entPC = (SELECT c2.minRequirementPC         FROM Games g2 JOIN Configurations c2 ON g2.configurationID = c2.configurationID                         WHERE g2.gameName = "Cou
nter-Strike") ORDER BY g.gameName;
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN



                                                                                                                                             |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------+
| -> Sort: g.gameName  (actual time=0.081..0.082 rows=10 loops=1)
    -> Stream results  (cost=1.43 rows=2) (actual time=0.046..0.063 rows=10 loops=1)
        -> Nested loop inner join  (cost=1.43 rows=2) (actual time=0.042..0.054 rows=10 loops=1)
            -> Filter: (c.minRequirementPC = (select #2))  (cost=0.70 rows=2) (actual time=0.026..0.030 rows=2 loops=1)
                -> Index lookup on c using pcMinRecReq (minRequirementPC=(select #2))  (cost=0.70 rows=2) (actual time=0.021..0.022 rows=2 loops=1)
                -> Select #2 (subquery in condition; run only once)
                    -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)
            -> Index lookup on g using configurationID (configurationID=c.configurationID)  (cost=0.31 rows=1) (actual time=0.008..0.011 rows=5 loops=2)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

| -> Sort: g.gameName (actual time=0.081..0.082 rows=10 loops=1)
-> Stream results (cost=1.43 rows=2) (actual time=0.046..0.063 rows=10 loops=1)
-> Nested loop inner join (cost=1.43 rows=2) (actual time=0.042..0.054 rows=10 loops=1)
-> Filter: (c.minRequirementPC = (select #2)) (cost=0.70 rows=2) (actual time=0.026..0.030 rows=2 loops=1)
-> Index lookup on c using pcMinRecReq (minRequirementPC=(select #2)) (cost=0.70 rows=2) (actual time=0.021..0.022 rows=2 loops=1)
-> Select #2 (subquery in condition; run only once)

-> Rows fetched before execution (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)
    -> Index lookup on g using configurationID (configurationID=c.configurationID) (cost=0.31 rows=1) (actual time=0.008..0.011 rows=5 loops=2)
|

Here we see that the total cost is about the same as a single index on minRequirementPC. Here with indexing on both minRequirementPC, then recRequirementPC, the total cost is 1.43, the same as indexing on just minRequirementPC. This is again expected, as similar to the indexing approach on part (a) done on just recRequirementPC, there are no filters based on recRequirementPC, as it is just used in the SELECT clause, and so we expect no performance gain.

3. We chose to use the indexing on Configurations, on minRequirementPC(20). This is because indexing on minRequirementPC(20) compared to indexing on recRequirementPC(20) results in the nested loop inner join cost decreasing significantly. Without any indexing, the cost of the nested loop inner join is 238.86. With indexing on recRequirementPC(20), the cost is, as expected, the same at 238.86. The reason for both being slow is seen in the explain analyze command, which shows that most of the cost (181.35) comes from a table scan on configurations for PC minimum requirements. With indexing on recRequirementPC(20), the cost of the nested loop inner join drops significantly to 1.43 from 238.86 earlier. This is because now, we can do an index lookup on the configurations table for PC recommended requirements using the index, rather than a table scan. The third option is a composite index on both minRequirementPC(20) and recRequirementPC(20), in that order. This does not present a significant speedup compared with just an index on minRequirementPC(20), since there is no opportunity to use the indexing on recRequirementPC(20), as for example, there is no table scan on configurations for PC recommended requirements. So, we decided to index on Configurations on minRequirementPC(20) as it provides the best performance for this query with just a single index.

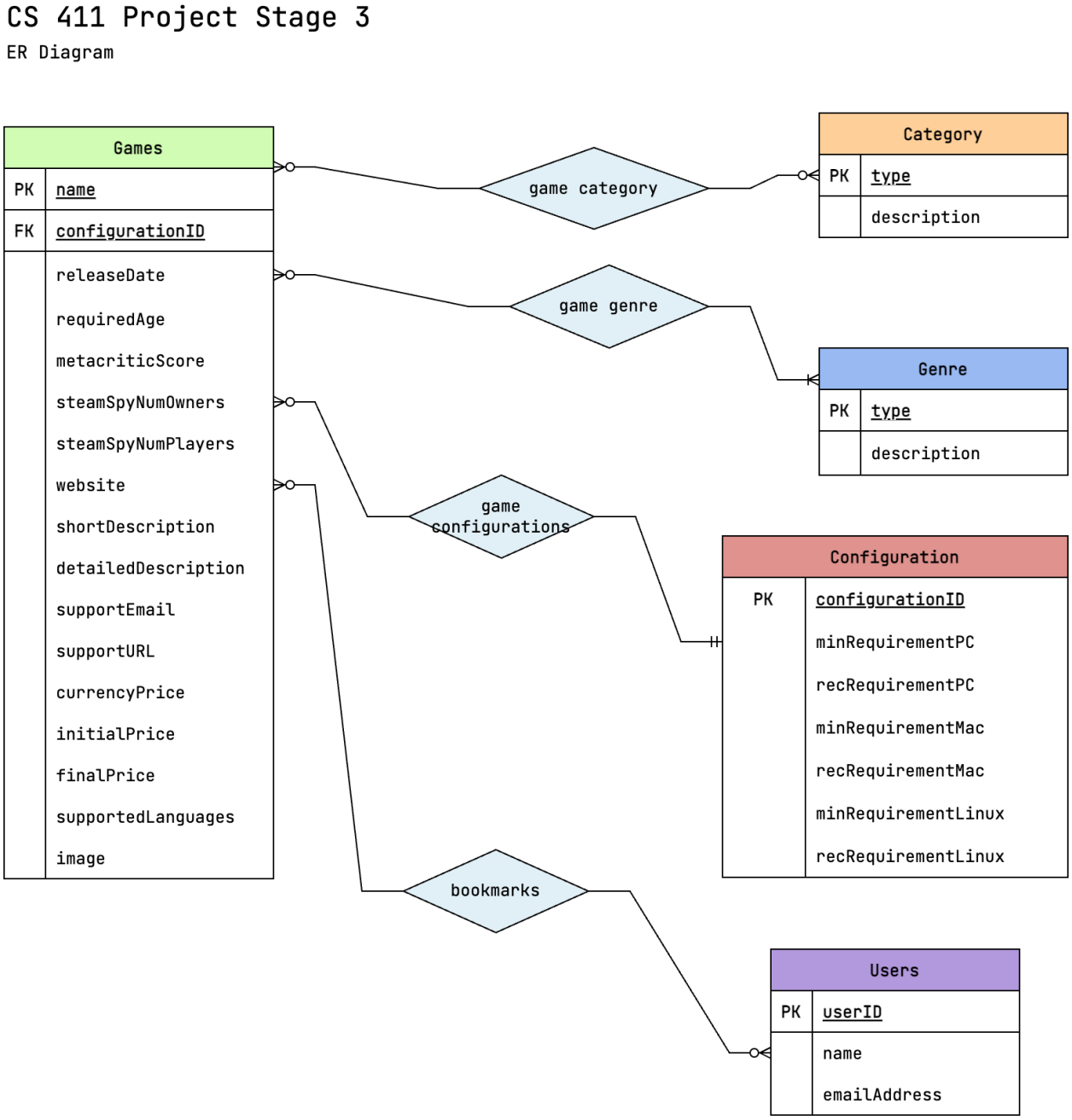
|

NOT TO SUBMIT:

DEADLINES Everything done by 3/8

- Inserting data and finalized schema done by Monday (3/6)
    - Leon
- Execute 2 advanced queries by Tuesday (3/7)
    - Pinakin
- Indexing analysis by Wednesday (3/8)
    - 3 different indexing designs for each query
    - Raymond
    - Aaron
- Buffer (3/9)

Reference diagram:



Further notes:

Original First query (works if we have bookmarks, perhaps use in next stage once users is populated)

- Display average metacritic score for bookmarked games
    - Join games to bookmarks to users
    - Group by user, then take AVG(metacritic score)
    - Index on: user

```
SELECT AVG(metacriticScore)
FROM Games JOIN Bookmarks ON Games.name = Bookmarks.gameName JOIN Users ON Bookmarks.userID = Users.userID
GROUP BY Games.name
```

Return the game name, game genre, and metacritic rating of distinct games that have a higher rating than the average rating for genre, ordered by genre in ascending order, and then rating in descending order

```
SELECT DISTINCT g.gameName, ge.gameGenre, g.metacriticScore
FROM Games g JOIN GameGenre ge ON g.gameName = ge.gameName
WHERE g.metacriticScore > (          SELECT AVG(g2.metacriticScore)
                                                FROM Games g2 JOIN GameGenre ge2
ON g2.gameName = ge2.gameName
                    WHERE ge.gameGenre = ge2.gameGenre)
ORDER BY ge.gameGenre, g.metacriticScore DESC
```