

–TABLE DDL COMMANDS–

```
CREATE TABLE User(UserId INT Primary Key,  
Username VARCHAR(255) NOT NULL,  
Email VARCHAR(255) NOT NULL,  
Password VARCHAR(255) NOT NULL);
```

```
CREATE TABLE Developer (DeveloperID INT Primary Key, Name VARCHAR(255),  
Country VARCHAR(255));
```

```
CREATE TABLE Game(GameID INT Primary Key, Title VARCHAR(255) NOT NULL,  
ReleaseDate VARCHAR(255),  
Price FLOAT,  
DeveloperID INT,  
Foreign Key (DeveloperID) References Developer(DeveloperID));
```

```
CREATE TABLE Plays (UserID INT, GameID INT,  
Primary Key (UserID, GameID),  
Foreign Key (UserID) References User(UserID),  
Foreign Key (GameID) References Game(GameID) );
```

```
CREATE TABLE Tag(TagID INT Primary Key,  
TagName VARCHAR(255) NOT NULL);
```

```
CREATE TABLE Recommendation(UserID INT, GameID INT, Rating INT, RecommendDate  
VARCHAR(255),  
Primary Key (UserID, GameID),  
Foreign Key (UserID) References User(UserID), Foreign Key (GameID) References  
Game(GameID));
```

```
CREATE TABLE GameTags (GameID INT, TagID INT,  
Primary Key (GameID, TagID),  
Foreign Key (GameID) References Game(GameID),  
Foreign Key (TagID) References Tag(TagID));
```

```
mysql> show tables;  
+-----+  
| Tables_in_GameRecommender |  
+-----+  
| Developer                  |  
| Game                      |  
| GameTags                  |  
| Plays                    |  
| Recommendation            |  
| Tag                      |  
| User                     |  
| gamefeatures             |  
+-----+  
8 rows in set (0.02 sec)
```

GameCount
81521

DeveloperCount
81521

RecommendationCount
1001

TagCount
450

-QUERIES-

1) Finding Developer/Developer Teams that make above average games

```

SELECT d.Name, AVG(r.Rating) as AvgRating, COUNT(g.GameID) as GameCount
FROM Developer d
JOIN Game g ON d.DeveloperID = g.DeveloperID
JOIN Recommendation r ON g.GameID = r.GameID
GROUP BY d.DeveloperID
HAVING AVG(r.Rating) > (
    SELECT AVG(r2.Rating)
    FROM Recommendation r2
)
ORDER BY AvgRating DESC, GameCount DESC
LIMIT 15;

```

Name	AvgRating	GameCount
Turtle Rock Studios	6.7027	222
Team Meat	6.6085	212
Gearbox Software	6.5783	166
3 rows in set (0.26 sec)		

Before Indexing:

```

-> Limit: 15 row(s) (actual time=7.258..7.259 rows=2 loops=1)
-> Sort: AvgRating DESC, GameCount DESC (actual time=7.257..7.258 rows=2 loops=1)
-> Filter: (avg(r.Rating) > (select #2)) (actual time=7.230..7.233 rows=2 loops=1)
-> Table scan on <temporary> (actual time=6.612..6.614 rows=6 loops=1)
-> Aggregate using temporary table (actual time=6.609..6.609 rows=6 loops=1)
-> Nested loop inner join (cost=2725.88 rows=1801) (actual time=0.065..5.023 rows=1801 loops=1)
-> Nested loop inner join (cost=1869.80 rows=1801) (actual time=0.056..3.068 rows=1801 loops=1)
-> Table scan on r (cost=184.10 rows=1801) (actual time=0.040..0.638 rows=1801 loops=1)
-> Filter: (g.DeveloperID is not null) (cost=0.84 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
-> Single-row index lookup on g using PRIMARY (GameID=r.GameID) (cost=0.84 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
-> Single-row index lookup on d using PRIMARY (DeveloperID=g.DeveloperID) (cost=0.38 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
-> Select #2 (subquery in condition; run only once)
-> Aggregate: avg(r2.Rating) (cost=364.20 rows=1) (actual time=0.580..0.580 rows=1 loops=1)
-> Table scan on r2 (cost=184.10 rows=1801) (actual time=0.032..0.419 rows=1801 loops=1)

```

Cost – 2725

First Index:

CREATE INDEX recIndex ON Recommendation (Rating);

```

-> Limit: 15 row(s) (actual time=71.126..71.126 rows=2 loops=1)
-> Sort: AvgRating DESC, GameCount DESC (actual time=71.125..71.125 rows=2 loops=1)
-> Filter: (avg(r.Rating) > (select #2)) (actual time=70.997..71.001 rows=2 loops=1)
-> Table scan on <temporary> (actual time=68.229..68.238 rows=6 loops=1)
-> Aggregate using temporary table (actual time=68.226..68.226 rows=6 loops=1)
-> Nested loop inner join (cost=3414.12 rows=1801) (actual time=61.782..66.415 rows=1801 loops=1)
-> Nested loop inner join (cost=1433.02 rows=1801) (actual time=0.856..3.596 rows=1801 loops=1)
-> Covering index scan on r using recIndex (cost=184.10 rows=1801) (actual time=0.081..0.570 rows=1801 loops=1)
-> Filter: (g.DeveloperID is not null) (cost=0.59 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
-> Single-row index lookup on g using PRIMARY (GameID=r.GameID) (cost=0.59 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
-> Single-row index lookup on d using PRIMARY (DeveloperID=g.DeveloperID) (cost=1.00 rows=1) (actual time=0.035..0.035 rows=1 loops=1801)
-> Select #2 (subquery in condition; run only once)
-> Aggregate: avg(r2.Rating) (cost=364.20 rows=1) (actual time=0.544..0.545 rows=1 loops=1)
-> Covering index scan on r2 using recIndex (cost=184.10 rows=1801) (actual time=0.042..0.377 rows=1801 loops=1)

```

Cost – 3414

Second Index:

CREATE INDEX nameIndex ON Developer (Name);

```

-> Limit: 15 row(s) (actual time=270.138..270.138 rows=2 loops=1)
-> Sort: AvgRating DESC, GameCount DESC (actual time=270.137..270.137 rows=2 loops=1)
-> Filter: (avg(r.Rating) > (select #2)) (actual time=270.112..270.114 rows=2 loops=1)
-> Table scan on <temporary> (actual time=269.417..269.420 rows=6 loops=1)
-> Aggregate using temporary table (actual time=269.414..269.414 rows=6 loops=1)
-> Nested loop inner join (cost=2686.11 rows=1801) (actual time=98.926..267.670 rows=1801 loops=1)
-> Nested loop inner join (cost=1881.00 rows=1801) (actual time=75.440..241.973 rows=1801 loops=1)
-> Table scan on r (cost=195.30 rows=1801) (actual time=75.406..239.216 rows=1801 loops=1)
-> Filter: (g.DeveloperID is not null) (cost=0.84 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
-> Single-row index lookup on g using PRIMARY (GameID=r.GameID) (cost=0.84 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
-> Single-row index lookup on d using PRIMARY (DeveloperID=g.DeveloperID) (cost=0.35 rows=1) (actual time=0.014..0.014 rows=1 loops=1801)
-> Select #2 (subquery in condition; run only once)
-> Aggregate: avg(r2.Rating) (cost=375.40 rows=1) (actual time=0.657..0.657 rows=1 loops=1)
-> Table scan on r2 (cost=195.30 rows=1801) (actual time=0.039..0.471 rows=1801 loops=1)

```

Cost – 2686

Third Index:

CREATE INDEX recIndex ON Recommendation (Rating);

CREATE INDEX nameIndex ON Developer (Name);

```

-> Limit: 15 row(s) (actual time=6.735..6.735 rows=2 loops=1)
-> Sort: AvgRating DESC, GameCount DESC (actual time=6.734..6.734 rows=2 loops=1)
-> Filter: (avg(r.Rating) > (select #2)) (actual time=6.706..6.708 rows=2 loops=1)
-> Table scan on <temporary> (actual time=6.177..6.179 rows=6 loops=1)
-> Aggregate using temporary table (actual time=6.173..6.173 rows=6 loops=1)
-> Nested loop inner join (cost=2671.27 rows=1801) (actual time=0.100..4.585 rows=1801 loops=1)
-> Nested loop inner join (cost=1869.80 rows=1801) (actual time=0.090..2.821 rows=1801 loops=1)
-> Covering index scan on r using recIndex (cost=184.10 rows=1801) (actual time=0.064..0.609 rows=1801 loops=1)
-> Filter: (g.DeveloperID is not null) (cost=0.84 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
-> Single-row index lookup on g using PRIMARY (GameID=r.GameID) (cost=0.84 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
-> Single-row index lookup on d using PRIMARY (DeveloperID=g.DeveloperID) (cost=0.35 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
-> Select #2 (subquery in condition; run only once)
-> Aggregate: avg(r2.Rating) (cost=364.20 rows=1) (actual time=0.493..0.493 rows=1 loops=1)
-> Covering index scan on r2 using recIndex (cost=184.10 rows=1801) (actual time=0.026..0.336 rows=1801 loops=1)

```

Cost – 2671

Analysis: Our chosen design was the third index, which creates an index on both the Rating column in the Recommendation table, and the Name column in the developer table. Indexing the Name column seems to be key, as it targets a column that has a lot of operations on it. However, this optimization only improved performance a little more than the default setting.

2) Find top-rated games through average rating

```

SELECT g.GameID, g.Title, d.Name AS Developer, AVG(r.Rating) AS AvgRating
FROM Game g

```

JOIN Recommendation r ON g.GameID = r.GameID
 JOIN Developer d ON g.DeveloperID = d.DeveloperID
 GROUP BY g.GameID, g.Title, d.Name
 ORDER BY AvgRating DESC
 LIMIT 15;

GameID	Title	Developer	AvgRating
3	Evolve Stage 2	Turtle Rock Studios	6.7027
1	Super Meat Boy	Team Meat	6.6085
4	Borderlands 3	Gearbox Software	6.5783
2	DCS World Steam Edition	Eagle Dynamics SA	6.4398
5	BioShock Infinite	Irrational Games,Virtual Programming (Linux)	6.3110
0	Title	Name	0.0000

INDEX:

NO INDEXING:

```
-----+
-> Limit: 15 row(s) (actual time=7.639..7.640 rows=6 loops=1)
-> Sort: AvgRating DESC, limit input to 15 row(s) per chunk (actual time=7.638..7.639 rows=6 loops=1)
-> Table scan on <temporary> (actual time=7.609..7.610 rows=6 loops=1)
-> Aggregate using temporary table (actual time=7.602..7.602 rows=6 loops=1)
-> Nested loop inner join (cost=3165.57 rows=1801) (actual time=0.079..5.177 rows=1801 loops=1)
-> Nested loop inner join (cost=1188.12 rows=1801) (actual time=0.072..3.175 rows=1801 loops=1)
-> Table scan on r (cost=184.10 rows=1801) (actual time=0.055..0.603 rows=1801 loops=1)
-> Filter: (g.DeveloperID is not null) (cost=0.46 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
-> Single-row index lookup on g using PRIMARY (GameID=r.GameID) (cost=0.46 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
-> Single-row index lookup on d using PRIMARY (DeveloperID=g.DeveloperID) (cost=1.00 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
```

Cost = 3165.57

First Index:

CREATE INDEX idx_Title_Game ON Game (Title);

```
-----+
-> Limit: 15 row(s) (actual time=7.602..7.603 rows=6 loops=1)
-> Sort: AvgRating DESC, limit input to 15 row(s) per chunk (actual time=7.601..7.602 rows=6 loops=1)
-> Table scan on <temporary> (actual time=7.570..7.572 rows=6 loops=1)
-> Aggregate using temporary table (actual time=7.569..7.569 rows=6 loops=1)
-> Nested loop inner join (cost=3365.03 rows=1801) (actual time=0.051..5.129 rows=1801 loops=1)
-> Nested loop inner join (cost=1387.57 rows=1801) (actual time=0.046..3.165 rows=1801 loops=1)
-> Table scan on r (cost=184.10 rows=1801) (actual time=0.034..0.624 rows=1801 loops=1)
-> Filter: (g.DeveloperID is not null) (cost=0.57 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
-> Single-row index lookup on g using PRIMARY (GameID=r.GameID) (cost=0.57 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
-> Single-row index lookup on d using PRIMARY (DeveloperID=g.DeveloperID) (cost=1.00 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
```

Cost = 3365.03

Second Index:

CREATE INDEX idx_Name_Developer ON Developer (Name);

```
-----+
-> Limit: 15 row(s) (actual time=7.462..7.463 rows=6 loops=1)
-> Sort: AvgRating DESC, limit input to 15 row(s) per chunk (actual time=7.461..7.462 rows=6 loops=1)
-> Table scan on <temporary> (actual time=7.431..7.432 rows=6 loops=1)
-> Aggregate using temporary table (actual time=7.429..7.429 rows=6 loops=1)
-> Nested loop inner join (cost=2790.50 rows=1801) (actual time=0.048..5.037 rows=1801 loops=1)
-> Nested loop inner join (cost=2160.15 rows=1801) (actual time=0.042..3.091 rows=1801 loops=1)
-> Table scan on r (cost=184.10 rows=1801) (actual time=0.031..0.591 rows=1801 loops=1)
-> Filter: (g.DeveloperID is not null) (cost=1.00 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
-> Single-row index lookup on g using PRIMARY (GameID=r.GameID) (cost=1.00 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
-> Single-row index lookup on d using PRIMARY (DeveloperID=g.DeveloperID) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
```

Cost = 2790.50

Third Index:

CREATE INDEX idx_Title_Game ON Game (Title);

CREATE INDEX idx_Name_Developer ON Developer (Name);

```

-----+
| -> Limit: 15 row(s) (actual time=7.360..7.361 rows=6 loops=1)
|   -> Sort: AvgRating DESC, limit input to 15 row(s) per chunk (actual time=7.359..7.360 rows=6 loops=1)
|     -> Table scan on <temporary> (actual time=7.332..7.334 rows=6 loops=1)
|       -> Aggregate using temporary table (actual time=7.330..7.330 rows=6 loops=1)
|         -> Nested loop inner join (cost=3145.38 rows=1801) (actual time=0.048..5.002 rows=1801 loops=1)
|           -> Nested loop inner join (cost=1167.92 rows=1801) (actual time=0.042..3.031 rows=1801 loops=1)
|             -> Table scan on r (cost=184.10 rows=1801) (actual time=0.031..0.579 rows=1801 loops=1)
|               -> Filter: (g.DeveloperID is not null) (cost=0.45 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
|                 -> Single-row index lookup on g using PRIMARY (GameID=r.GameID) (cost=0.45 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
|               -> Single-row index lookup on d using PRIMARY (DeveloperID=g.DeveloperID) (cost=1.00 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
|             -> Single-row index lookup on d using PRIMARY (DeveloperID=g.DeveloperID) (cost=1.00 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
|           -> Single-row index lookup on d using PRIMARY (DeveloperID=g.DeveloperID) (cost=1.00 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
|         -> Single-row index lookup on d using PRIMARY (DeveloperID=g.DeveloperID) (cost=1.00 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
|       -> Single-row index lookup on d using PRIMARY (DeveloperID=g.DeveloperID) (cost=1.00 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
|     -> Single-row index lookup on d using PRIMARY (DeveloperID=g.DeveloperID) (cost=1.00 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
|   -> Single-row index lookup on d using PRIMARY (DeveloperID=g.DeveloperID) (cost=1.00 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
| -> Single-row index lookup on d using PRIMARY (DeveloperID=g.DeveloperID) (cost=1.00 rows=1) (actual time=0.001..0.001 rows=1 loops=1801)
-----+

```

Cost = 3145.38

ANALYSIS: The index design that we have decided to go for is the index for Developer Name. The reason why the `Developer.Name` column was effective is that it optimized the operations the query performs most intensively through its grouping. Specifically, by the developer name and potentially improving the efficiency of the join operation.

3) Find a specific game using a specific tag for both Adventure and Indie

```

SELECT g.GameID, g.Title, g.ReleaseDate, g.Price
FROM Game g JOIN GameTags m ON (g.GameID = m.GameID)
JOIN Tag t ON (m.TagID = t.TagID)
WHERE t.TagName IN ('Adventure', 'Indie')
GROUP BY g.GameID, g.Title, g.ReleaseDate, g.Price
HAVING COUNT(DISTINCT t.TagName) = 2
LIMIT 15;

```

GameID	Title	ReleaseDate	Price
11	Papers, Please	Aug 8, 2013	9.99
18	Garry's Mod	Nov 29, 2006	9.99
49	Robocraft	Aug 24, 2017	0
62	Warhammer: Vermintide 2	Mar 8, 2018	29.99
63	Kholat	Jun 9, 2015	3.99
68	LIMBO	Aug 2, 2011	9.99
69	Kathy Rain	May 5, 2016	14.99
74	SCUM	Aug 29, 2018	34.99
85	Realm of the Mad God Exalt	Feb 20, 2012	0
86	Trine Enchanted Edition	Jul 2, 2009	3.74
88	Guns of Icarus Online	Oct 29, 2012	4.99
89	Bloons TD 6	Dec 17, 2018	9.99
90	Clicker Heroes	May 13, 2015	0
93	Starbound	Jul 22, 2016	14.99
98	Spiral Knights	Jun 14, 2011	0

15 rows in set (0.45 sec)

```

-> Limit: 15 row(s) (actual time=2004.297..2004.377 rows=15 loops=1)
-> Filter: (count(distinct Tag.TagName) = 2) (actual time=2004.295..2004.374 rows=15 loops=1)
-> Group aggregate: count(distinct Tag.TagName) (actual time=2004.252..2004.362 rows=72 loops=1)
-> Sort: g.GameID, g.Title, g.ReleaseDate, g.Price (actual time=2004.221..2004.232 rows=98 loops=1)
-> Stream results (cost=177089.03 rows=163156) (actual time=138.750..1691.797 rows=68856 loops=1)
-> Nested loop inner join (cost=177089.03 rows=163156) (actual time=138.739..1654.299 rows=68856 loops=1)
-> Nested loop inner join (cost=107633.46 rows=163156) (actual time=138.707..833.265 rows=68856 loops=1)
-> Filter: (t.TagName in ('Adventure','Indie')) (cost=45.25 rows=90) (actual time=0.993..54.545 rows=2 loops=1)
-> Table scan on t (cost=45.25 rows=450) (actual time=0.072..53.543 rows=450 loops=1)
-> Filter: (m.GameID is not null) (cost=1016.15 rows=1813) (actual time=118.631..387.036 rows=34428 loops=2)
-> Index lookup on m using TagID (TagID=t.TagID) (cost=1016.15 rows=1813) (actual time=118.627..383.158 rows=34428 loops=2)
-> Single-row index lookup on g using PRIMARY (GameID=m.GameID) (cost=0.33 rows=1) (actual time=0.012..0.012 rows=1 loops=68856)

```

INDEX:

NO INDEXING:

```

| -> Limit: 15 row(s) (actual time=3326.876..3326.976 rows=15 loops=1)
-> Filter: (count(distinct Tag.TagName) = 2) (actual time=3326.874..3326.973 rows=15 loops=1)
-> Group aggregate: count(distinct Tag.TagName) (actual time=3326.861..3326.961 rows=72 loops=1)
-> Sort: g.GameID, g.Title, g.ReleaseDate, g.Price (actual time=3326.786..3326.797 rows=98 loops=1)
-> Stream results (cost=337324.70 rows=192005) (actual time=198.232..3189.650 rows=68856 loops=1)
-> Nested loop inner join (cost=337324.70 rows=192005) (actual time=198.220..3147.758 rows=68856 loops=1)
-> Nested loop inner join (cost=126657.80 rows=192005) (actual time=198.175..1013.657 rows=68856 loops=1)
-> Filter: (t.TagName in ('Adventure','Indie')) (cost=46.00 rows=90) (actual time=35.785..35.988 rows=2 loops=1)
-> Table scan on t (cost=46.00 rows=450) (actual time=35.768..35.906 rows=450 loops=1)
-> Filter: (m.GameID is not null) (cost=1195.83 rows=2133) (actual time=119.408..486.166 rows=34428 loops=2)
-> Index lookup on m using TagID (TagID=t.TagID) (cost=1195.83 rows=2133) (actual time=119.403..482.319 rows=34428 loops=2)
-> Single-row index lookup on g using PRIMARY (GameID=m.GameID) (cost=1.00 rows=1) (actual time=0.031..0.031 rows=1 loops=68856)
|

```

Cost = 337,324.70

First Index:

CREATE INDEX idx_title_game ON Game (Title);

```

| -> Limit: 15 row(s) (actual time=2092.744..2092.877 rows=15 loops=1)
-> Filter: (count(distinct Tag.TagName) = 2) (actual time=2092.743..2092.875 rows=15 loops=1)
-> Group aggregate: count(distinct Tag.TagName) (actual time=2092.727..2092.860 rows=72 loops=1)
-> Sort: g.GameID, g.Title, g.ReleaseDate, g.Price (actual time=2092.696..2092.714 rows=98 loops=1)
-> Stream results (cost=200049.52 rows=192005) (actual time=177.535..1934.397 rows=68856 loops=1)
-> Nested loop inner join (cost=200049.52 rows=192005) (actual time=177.520..1861.578 rows=68856 loops=1)
-> Nested loop inner join (cost=126657.05 rows=192005) (actual time=144.151..1051.588 rows=68856 loops=1)
-> Filter: (t.TagName in ('Adventure','Indie')) (cost=45.25 rows=90) (actual time=0.100..0.440 rows=2 loops=1)
-> Table scan on t (cost=45.25 rows=450) (actual time=0.092..0.329 rows=450 loops=1)
-> Filter: (m.GameID is not null) (cost=1195.83 rows=2133) (actual time=79.242..522.279 rows=34428 loops=2)
-> Index lookup on m using TagID (TagID=t.TagID) (cost=1195.83 rows=2133) (actual time=79.238..517.560 rows=34428 loops=2)
-> Single-row index lookup on g using PRIMARY (GameID=m.GameID) (cost=0.28 rows=1) (actual time=0.011..0.011 rows=1 loops=68856)
|

```

Cost = 200049.52

Second Index:

CREATE INDEX idx_tag_tagid_tagname ON Tag (TagName);

```

| -> Limit: 15 row(s) (actual time=1176.232..1176.318 rows=15 loops=1)
-> Filter: (count(distinct Tag.TagName) = 2) (actual time=1176.230..1176.314 rows=15 loops=1)
-> Group aggregate: count(distinct Tag.TagName) (actual time=1176.214..1176.301 rows=72 loops=1)
-> Sort: g.GameID, g.Title, g.ReleaseDate, g.Price (actual time=1176.188..1176.200 rows=98 loops=1)
-> Stream results (cost=4373.45 rows=4267) (actual time=88.250..1037.174 rows=68856 loops=1)
-> Nested loop inner join (cost=4373.45 rows=4267) (actual time=88.240..1000.437 rows=68856 loops=1)
-> Nested loop inner join (cost=2814.29 rows=4267) (actual time=88.204..841.174 rows=68856 loops=1)
-> Filter: (t.TagName in ('Adventure','Indie')) (cost=0.69 rows=2) (actual time=1.372..1.401 rows=2 loops=1)
-> Covering index range scan on t using idx_tag_tagid_tagname over (TagName = 'Adventure') OR (TagName = 'Indie') (cost=0.69 rows=2) (actual time=0.046..0.070 rows=2 loops=1)
-> Filter: (m.GameID is not null) (cost=1300.13 rows=2133) (actual time=58.772..417.624 rows=34428 loops=2)
-> Index lookup on m using TagID (TagID=t.TagID) (cost=1300.13 rows=2133) (actual time=58.768..413.996 rows=34428 loops=2)
-> Single-row index lookup on g using PRIMARY (GameID=m.GameID) (cost=0.27 rows=1) (actual time=0.002..0.002 rows=1 loops=68856)
|

```

Cost = 4373.45

Third Index:

CREATE INDEX idx_game_composite ON Game (Title, ReleaseDate, Price);

```

| -> Limit: 15 row(s) (actual time=3720.607..3720.733 rows=15 loops=1)
-> Filter: (count(distinct Tag.TagName) = 2) (actual time=3720.606..3720.730 rows=15 loops=1)
-> Group aggregate: count(distinct Tag.TagName) (actual time=3720.591..3720.716 rows=72 loops=1)
-> Sort: g.GameID, g.Title, g.ReleaseDate, g.Price (actual time=3720.555..3720.573 rows=98 loops=1)
-> Stream results (cost=256305.87 rows=192005) (actual time=251.492..3366.429 rows=68856 loops=1)
-> Nested loop inner join (cost=256305.87 rows=192005) (actual time=251.478..3300.064 rows=68856 loops=1)
-> Nested loop inner join (cost=126657.80 rows=192005) (actual time=229.511..1376.620 rows=68856 loops=1)
-> Filter: (t.TagName in ('Adventure','Indie')) (cost=46.00 rows=90) (actual time=52.119..97.241 rows=2 loops=1)
-> Table scan on t (cost=46.00 rows=450) (actual time=52.103..97.114 rows=450 loops=1)
-> Filter: (m.GameID is not null) (cost=1195.83 rows=2133) (actual time=154.308..636.239 rows=34428 loops=2)
-> Index lookup on m using TagID (TagID=t.TagID) (cost=1195.83 rows=2133) (actual time=154.304..630.957 rows=34428 loops=2)
-> Single-row index lookup on g using PRIMARY (GameID=m.GameID) (cost=0.58 rows=1) (actual time=0.028..0.028 rows=1 loops=68856)
|

```

COST = 256305.87

ANALYSIS: The index design that we have decided to go for is the index for Tag Name. We can say for a fact that using the TagName index has definitely improved the cost by a large margin.

The reason why the `TagName` column was super effective is that it solely optimized the operations that the query specifically demands for. The TagName column alone has a lot of values inside so to have an index that only takes a look at that column and its IDs has majorly saved both money and time.

4) Recommend Game based on games played

```
SELECT g.Title, g.Price, COUNT(p2.GameID) as PlayCount
FROM Game g
JOIN Plays p2 ON g.GameID = p2.GameID
JOIN GameTags gt ON g.GameID = gt.GameID
JOIN Tag t ON gt.TagID = t.TagID
WHERE t.TagName IN (
    SELECT DISTINCT t2.TagName
    FROM GameTags gt2
    JOIN Tag t2 ON gt2.TagID = t2.TagID
    JOIN Plays p ON gt2.GameID = p.GameID
)
AND g.GameID NOT IN (
    SELECT GameID
    FROM Plays
    WHERE UserID = @UserID
)
GROUP BY g.GameID
ORDER BY PlayCount DESC, g.Price ASC
LIMIT 15;
```

Title	Price	PlayCount
Evolve Stage 2	0	7800
Super Meat Boy	14.99	7340
BioShock Infinite	29.99	7320
DCS World Steam Edition	0	6880
Borderlands 3	59.99	6660
Title	0	1

6 rows in set (4.66 sec)

INDEX:

No index:

```

-> Limit: 15 row(s) (actual time=3077.613..3077.614 rows=6 loops=1)
-> Sort: PlayCount DESC, g.Price, limit input to 15 row(s) per chunk (actual time=3077.612..3077.612 rows=6 loops=1)
-> Table scan on <temporary> (actual time=3077.579..3077.581 rows=6 loops=1)
-> Aggregate using temporary table (actual time=3077.576..3077.576 rows=6 loops=1)
-> Nested loop antijoin (cost=1265327841.89 rows=114946638) (actual time=98.288..3045.691 rows=36001 loops=1)
-> Nested loop inner join (cost=1253832772.89 rows=63887610) (actual time=99.271..3028.659 rows=36001 loops=1)
-> Nested loop inner join (cost=1247444007.37 rows=63887610) (actual time=99.257..3012.377 rows=36001 loops=1)
-> Inner hash join (no condition) (cost=128502568.05 rows=127752136) (actual time=99.033..116.957 rows=122468 loops=1)
-> Covering index scan on p2 using GameID (cost=4.19 rows=1801) (actual time=0.024..1.087 rows=1801 loops=1)
-> Hash
-> Filter: (t.TagName = 'subquery2')>.TagName) (cost=727346.13 rows=709468) (actual time=98.789..98.981 rows=68 loops=1)
-> Inner hash join (<hash>(t.TagName)=<hash>(<subquery2>.TagName)) (cost=727346.13 rows=709468) (actual time=98.785..98.959 rows=68 loops=1)
-> Table scan on t (cost=5.63 rows=450) (actual time=0.045..0.151 rows=450 loops=1)
-> Hash
-> Table scan on <subquery2> (cost=17677.31..17876.86 rows=15766) (actual time=98.672..98.683 rows=68 loops=1)
-> Materialize with deduplication (cost=17677.30..17677.30 rows=15766) (actual time=98.668..98.668 rows=68 loops=1)
-> Nested loop inner join (cost=16100.70 rows=15766) (actual time=0.109..83.731 rows=36001 loops=1)
-> Nested loop inner join (cost=10582.61 rows=15766) (actual time=0.098..48.381 rows=36001 loops=1)
-> Covering index scan on p using GameID (cost=186.23 rows=1801) (actual time=0.056..0.567 rows=1801 loops=1)
-> Filter: (gt2.TagID is not null) (cost=4.90 rows=9) (actual time=0.002..0.025 rows=20 loops=1801)
-> Index lookup on gt2 using GameID (GameID=p.GameID) (cost=4.90 rows=9) (actual time=0.002..0.023 rows=20 loops=1801)
-> Single-row index lookup on t2 using PRIMARY (TagID=gt2.TagID) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=36001)
-> Filter: (gt.TagID = t.TagID) (cost=4.90 rows=0.05) (actual time=0.020..0.024 rows=0 loops=122468)
-> Index lookup on gt using GameID (GameID=p2.GameID) (cost=4.90 rows=9) (actual time=0.002..0.022 rows=20 loops=122468)
-> Single-row index lookup on g using PRIMARY (GameID=p2.GameID) (cost=0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=36001)
-> Single-row index lookup on <subquery3> using <auto distinct key> (GameID=p2.GameID) (actual time=0.000..0.000 rows=0 loops=36001)
-> Materialize with deduplication (cost=1.24..1.24 rows=2) (actual time=0.013..0.013 rows=0 loops=1)
-> Filter: (Plays.GameID is not null) (cost=1.06 rows=2) (actual time=0.011..0.011 rows=0 loops=1)
-> Filter: (Plays.UserID = <cache>{@UserID}) (cost=1.06 rows=2) (actual time=0.011..0.011 rows=0 loops=1)
-> Covering index lookup on Plays using PRIMARY (UserID=@UserID) (cost=1.06 rows=2) (actual time=0.010..0.010 rows=0 loops=1)

```

Cost: 1265327841

First Index:

CREATE INDEX tagnameIndex ON Tag (TagName);

```

-> Limit: 15 row(s) (actual time=3071.132..3071.133 rows=6 loops=1)
-> Sort: PlayCount DESC, g.Price, limit input to 15 row(s) per chunk (actual time=3071.131..3071.132 rows=6 loops=1)
-> Table scan on <temporary> (actual time=3071.031..3071.033 rows=6 loops=1)
-> Aggregate using temporary table (actual time=3071.027..3071.027 rows=6 loops=1)
-> Nested loop antijoin (cost=28121894.46 rows=2554370) (actual time=109.548..3040.294 rows=36001 loops=1)
-> Nested loop inner join (cost=27866448.49 rows=1419725) (actual time=109.529..3023.247 rows=36001 loops=1)
-> Nested loop inner join (cost=27724386.22 rows=1419725) (actual time=109.515..3006.964 rows=36001 loops=1)
-> Inner hash join (no condition) (cost=2859021.28 rows=28394491) (actual time=109.350..127.522 rows=122468 loops=1)
-> Covering index scan on p2 using GameID (cost=186.27 rows=1801) (actual time=0.027..1.164 rows=1801 loops=1)
-> Hash
-> Nested loop inner join (cost=19565.95 rows=15766) (actual time=109.046..109.273 rows=68 loops=1)
-> Table scan on <subquery2> (cost=17677.31..17876.86 rows=15766) (actual time=109.013..109.025 rows=68 loops=1)
-> Materialize with deduplication (cost=17677.30..17677.30 rows=15766) (actual time=109.009..109.009 rows=68 loops=1)
-> Nested loop inner join (cost=16100.70 rows=15766) (actual time=0.069..92.289 rows=36001 loops=1)
-> Nested loop inner join (cost=10582.61 rows=15766) (actual time=0.061..53.357 rows=36001 loops=1)
-> Covering index scan on p using GameID (cost=186.23 rows=1801) (actual time=0.036..0.606 rows=1801 loops=1)
-> Filter: (gt2.TagID is not null) (cost=4.90 rows=9) (actual time=0.002..0.028 rows=20 loops=1801)
-> Index lookup on gt2 using GameID (GameID=p.GameID) (cost=4.90 rows=9) (actual time=0.002..0.026 rows=20 loops=1801)
-> Single-row index lookup on t2 using PRIMARY (TagID=gt2.TagID) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=36001)
-> Covering index lookup on t using TagnameIndex (TagName=<subquery2>.TagName) (cost=112.60 rows=1) (actual time=0.003..0.003 rows=1 loops=68)
-> Index lookup on gt using GameID (GameID=p2.GameID) (cost=4.90 rows=9) (actual time=0.002..0.022 rows=20 loops=122468)
-> Single-row index lookup on g using PRIMARY (GameID=p2.GameID) (cost=1.00 rows=1) (actual time=0.000..0.000 rows=1 loops=36001)
-> Single-row index lookup on <subquery3> using <auto distinct key> (GameID=p2.GameID) (actual time=0.000..0.000 rows=0 loops=36001)
-> Materialize with deduplication (cost=1.24..1.24 rows=2) (actual time=0.015..0.015 rows=0 loops=1)
-> Filter: (Plays.GameID is not null) (cost=1.06 rows=2) (actual time=0.012..0.012 rows=0 loops=1)
-> Filter: (Plays.UserID = <cache>{@UserID}) (cost=1.06 rows=2) (actual time=0.012..0.012 rows=0 loops=1)
-> Covering index lookup on Plays using PRIMARY (UserID=@UserID) (cost=1.06 rows=2) (actual time=0.011..0.011 rows=0 loops=1)

```

Cost: 28121894

Second Index:

CREATE INDEX titleIndex ON Game(Title);

```

-> Limit: 15 row(s) (actual time=3039.075..3039.076 rows=6 loops=1)
-> Sort: PlayCount DESC, g.Price, limit input to 15 row(s) per chunk (actual time=3039.074..3039.075 rows=6 loops=1)
-> Table scan on <temporary> (actual time=3039.044..3039.045 rows=6 loops=1)
-> Aggregate using temporary table (actual time=3039.040..3039.040 rows=6 loops=1)
-> Nested loop antijoin (cost=1265327839.93 rows=114946638) (actual time=101.925..3001.484 rows=36001 loops=1)
-> Nested loop inner join (cost=1253832770.93 rows=63887610) (actual time=101.909..2984.321 rows=36001 loops=1)
-> Nested loop inner join (cost=1247444007.37 rows=63887610) (actual time=101.897..2968.377 rows=36001 loops=1)
-> Inner hash join (no condition) (cost=128502568.05 rows=127752136) (actual time=101.739..119.163 rows=122468 loops=1)
-> Covering index scan on p2 using GameID (cost=4.19 rows=1801) (actual time=0.024..1.087 rows=1801 loops=1)
-> Hash
-> Filter: (t.TagName = 'subquery2')>.TagName) (cost=727346.13 rows=709468) (actual time=101.485..101.682 rows=68 loops=1)
-> Inner hash join (<hash>(t.TagName)=<hash>(<subquery2>.TagName)) (cost=727346.13 rows=709468) (actual time=101.481..101.661 rows=68 loops=1)
-> Table scan on t (cost=5.63 rows=450) (actual time=0.052..0.154 rows=450 loops=1)
-> Hash
-> Table scan on <subquery2> (cost=17677.31..17876.86 rows=15766) (actual time=101.360..101.370 rows=68 loops=1)
-> Materialize with deduplication (cost=17677.30..17677.30 rows=15766) (actual time=101.356..101.356 rows=68 loops=1)
-> Nested loop inner join (cost=16100.70 rows=15766) (actual time=0.097..85.233 rows=36001 loops=1)
-> Nested loop inner join (cost=10582.61 rows=15766) (actual time=0.085..48.682 rows=36001 loops=1)
-> Covering index scan on p using GameID (cost=186.23 rows=1801) (actual time=0.054..0.554 rows=1801 loops=1)
-> Filter: (gt2.TagID is not null) (cost=4.90 rows=9) (actual time=0.002..0.025 rows=20 loops=1801)
-> Index lookup on gt2 using GameID (GameID=p.GameID) (cost=4.90 rows=9) (actual time=0.002..0.023 rows=20 loops=1801)
-> Single-row index lookup on t2 using PRIMARY (TagID=gt2.TagID) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=36001)
-> Filter: (gt.TagID = t.TagID) (cost=4.90 rows=0.05) (actual time=0.020..0.023 rows=0 loops=122468)
-> Index lookup on gt using GameID (GameID=p2.GameID) (cost=4.90 rows=9) (actual time=0.002..0.021 rows=20 loops=122468)
-> Single-row index lookup on g using PRIMARY (GameID=p2.GameID) (cost=0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=36001)
-> Single-row index lookup on <subquery3> using <auto distinct key> (GameID=p2.GameID) (actual time=0.000..0.000 rows=0 loops=36001)
-> Materialize with deduplication (cost=1.24..1.24 rows=2) (actual time=0.012..0.012 rows=0 loops=1)
-> Filter: (Plays.GameID is not null) (cost=1.06 rows=2) (actual time=0.010..0.010 rows=0 loops=1)
-> Filter: (Plays.UserID = <cache>{@UserID}) (cost=1.06 rows=2) (actual time=0.010..0.010 rows=0 loops=1)
-> Covering index lookup on Plays using PRIMARY (UserID=@UserID) (cost=1.06 rows=2) (actual time=0.009..0.009 rows=0 loops=1)

```

Cost: 1265327839

Third Index:

CREATE INDEX tagnameIndex ON Tag (TagName);

CREATE INDEX titleIndex ON Game(Title);


```

-> Limit: 15 row(s) (actual time=3056.267..3056.268 rows=6 loops=1)
-> Sort: PlayCount DESC, g.Price, limit input to 15 row(s) per chunk (actual time=3056.266..3056.266 rows=6 loops=1)
-> Table scan on <temporary> (actual time=3056.233..3056.234 rows=6 loops=1)
-> Aggregate using temporary table (actual time=3056.229..3056.229 rows=6 loops=1)
-> Nested loop antijoin (cost=20121864.67 rows=2554390) (actual time=103.535..3026.835 rows=36001 loops=1)
-> Nested loop inner join (cost=27866418.70 rows=1419725) (actual time=103.517..3009.802 rows=36001 loops=1)
-> Nested loop inner join (cost=27724386.22 rows=1419725) (actual time=103.504..2994.239 rows=36001 loops=1)
-> Inner hash join (no condition) (cost=2859021.28 rows=28394491) (actual time=103.249..123.278 rows=122468 loops=1)
-> Covering index scan on p2 using GameID (cost=186.27 rows=1801) (actual time=0.041..1.156 rows=1801 loops=1)
-> Hash
-> Nested loop inner join (cost=19565.95 rows=15766) (actual time=102.777..103.150 rows=68 loops=1)
-> Table scan on <subquery2> (cost=17677.31..17876.86 rows=15766) (actual time=102.748..102.766 rows=68 loops=1)
-> Materialize with deduplication (cost=17677.30..17677.30 rows=15766) (actual time=102.744..102.744 rows=68 loops=1)
-> Nested loop inner join (cost=16100.70 rows=15766) (actual time=0.061..0.874 rows=36001 loops=1)
-> Nested loop inner join (cost=10582.61 rows=15766) (actual time=0.055..51.698 rows=36001 loops=1)
-> Covering index scan on p using GameID (cost=186.23 rows=1801) (actual time=0.029..0.576 rows=1801 loops=1)
-> Filter: (gt2.TagID is not null) (cost=4.90 rows=9) (actual time=0.002..0.027 rows=20 loops=1801)
-> Index lookup on gt2 using GameID (GameID=p.GameID) (cost=4.90 rows=9) (actual time=0.002..0.025 rows=20 loops=1801)
-> Single-row index lookup on t2 using PRIMARY (TagID=gt2.TagID) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=36001)
-> Covering index lookup on t using tagnameIndex (TagName='<subquery2>'.TagName) (cost=112.60 rows=1) (actual time=0.005..0.005 rows=1 loops=68)
-> Filter: (gt.TagID = t.TagID) (cost=4.90 rows=0.05) (actual time=0.020..0.023 rows=0 loops=122468)
-> Index lookup on gt using GameID (GameID=p2.GameID) (cost=4.90 rows=9) (actual time=0.002..0.022 rows=20 loops=122468)
-> Single-row index lookup on g using PRIMARY (GameID=p2.GameID) (cost=0.67 rows=1) (actual time=0.000..0.000 rows=1 loops=36001)
-> Single-row index lookup on <subquery3> using <auto distinct key> (GameID=p2.GameID) (actual time=0.000..0.000 rows=0 loops=36001)
-> Materialize with deduplication (cost=1.24..1.24 rows=2) (actual time=0.015..0.015 rows=0 loops=1)
-> Filter: (Plays.GameID is not null) (cost=1.06 rows=2) (actual time=0.012..0.012 rows=0 loops=1)
-> Filter: (Plays.UserID = <cache>{@UserID})) (cost=1.06 rows=2) (actual time=0.011..0.011 rows=0 loops=1)
-> Covering index lookup on Plays using PRIMARY (UserID=@UserID) (cost=1.06 rows=2) (actual time=0.011..0.011 rows=0 loops=1)

```

Cost: 28121864

ANALYSIS: The index design we chose was the third configuration, although it is basically the same as the first. It looks like the important index was the TagName column in the Tag table. This column was a key part of the WHERE clause, so it makes sense that indexing it would greatly increase performance.