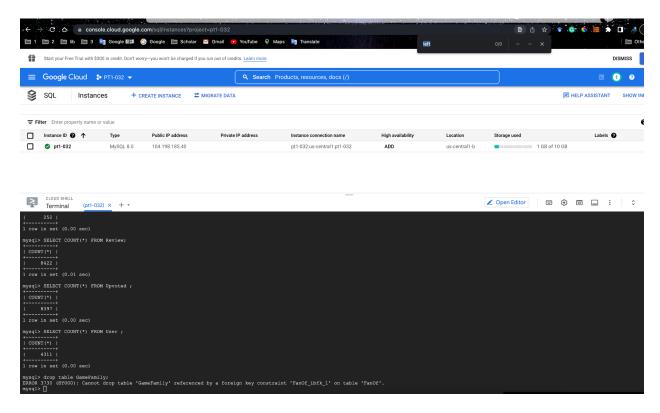
Project Track 1: Stage 3 Ace_DB

1. Database Implementation:

Screenshot of connection:



b. DDL Commands for tables:

```
CREATE TABLE Review (
PostID INT NOT NULL,
UserID INT NOT NULL,
GameID INT NOT NULL,
Rating FLOAT,
Comment VARCHAR(8000),
PRIMARY KEY (PostID),
FOREIGN KEY (UserID) REFERENCES User(UserID),
FOREIGN KEY (GameID) REFERENCES BoardgameProduct(GameID)
);

CREATE TABLE User (
UserID INT NOT NULL,
Username VARCHAR(255) NOT NULL,
```

```
Age INT.
Region VARCHAR(255),
Level INT,
Trading INT,
PRIMARY KEY (UserID)
);
CREATE TABLE GameFamily (
FamilyID INT NOT NULL,
Familyname VARCHAR(255),
PartyFriendly VARCHAR(255),
PRIMARY KEY (FamilyID)
);
CREATE TABLE BoardgameProduct (
GameID INT NOT NULL,
GameName VARCHAR(255),
Genre VARCHAR(255),
Duration INT,
Trading INT,
Description VARCHAR(8000),
Difficulty INT,
YearPublished INT,
PRIMARY KEY (GameID)
);
CREATE TABLE Contributor (
ContributorID INT NOT NULL,
Name VARCHAR(255),
Popularity INT,
PRIMARY KEY (ContributorID)
);
CREATE TABLE Upvoted(
PostID INT NOT NULL.
UserID INT NOT NULL,
PRIMARY KEY (PostID, UserID),
FOREIGN KEY (PostID) REFERENCES Review(PostID),
FOREIGN KEY (UserID) REFERENCES User(UserID)
);
CREATE TABLE Owned(
GameID INT NOT NULL,
UserID INT NOT NULL,
PRIMARY KEY (GameID, UserID),
FOREIGN KEY (GameID) REFERENCES BoardgameProduct(GameID),
```

```
FOREIGN KEY (UserID) REFERENCES User(UserID)
);
CREATE TABLE FanOf(
FamilyID INT NOT NULL,
UserID INT NOT NULL,
PRIMARY KEY (FamilyID, UserID),
FOREIGN KEY (FamilyID) REFERENCES GameFamily(FamilyID),
FOREIGN KEY (UserID) REFERENCES User(UserID)
);
CREATE TABLE Supports(
ContributorID INT NOT NULL,
UserID INT NOT NULL,
PRIMARY KEY (ContributorID, UserID),
FOREIGN KEY (ContributorID) REFERENCES Contributor(ContributorID),
FOREIGN KEY (UserID) REFERENCES User(UserID)
);
CREATE TABLE BelongsTo(
FamilyID INT NOT NULL,
GameID INT NOT NULL,
PRIMARY KEY (FamilyID, GameID),
FOREIGN KEY (FamilyID) REFERENCES GameFamily(FamilyID),
FOREIGN KEY (GameID) REFERENCES BoardgameProduct(GameID)
);
CREATE TABLE Created(
ContributorID INT NOT NULL,
GameID INT NOT NULL,
Role VARCHAR(50),
PRIMARY KEY (ContributorID, GameID),
FOREIGN KEY (ContributorID) REFERENCES Contributor(ContributorID),
FOREIGN KEY (GameID) REFERENCES BoardgameProduct(GameID)
```

c. 1000 rows in tables:

```
mysql> SELECT COUNT(*) FROM FanOf;
mysql> show tables;
                                              | COUNT(*) |
| Tables_in_group32_seq |
                                              6983
| BelongsTo
| BoardgameProduct
                                              1 row in set (0.00 sec)
| Contributor
| Created
| FanOf
                                              mysql> SELECT COUNT(*) FROM GameFamily ;
| GameFamily
| Owned
                                              | COUNT(*) |
| Review
Supports
| Upvoted
| User
                                              1 row in set (0.01 sec)
11 rows in set (0.00 sec)
                                              mysql> SELECT COUNT(*) FROM Owned ;
mysql> SELECT COUNT(*) FROM BelongsTo;
                                              | COUNT(*) |
| COUNT(*) |
                                              1 row in set (0.00 sec)
1 row in set (0.01 sec)
                                              mysql> SELECT COUNT(*) FROM Review;
mysql> SELECT COUNT(*) FROM BoardgameProduct;
                                              | COUNT(*) |
| COUNT(*) |
                                              8422 |
                                              1 row in set (0.01 sec)
1 row in set (0.01 sec)
                                              mysql> SELECT COUNT(*) FROM Upvoted ;
mysql> SELECT COUNT(*) FROM Contributor;
                                               | COUNT(*) |
| COUNT(*) |
                                              8397
    1500 |
                                              1 row in set (0.00 sec)
1 row in set (0.00 sec)
```

2. Advanced SQL Queries

a. Query 1: This query retrieves users' Users' (level> 3) posts with relatively high numbers (>=10) of upvotes to highlight the most popular posts created by the most popular users. Code below:

```
SELECT Rev.PostID, Rev.Rating, User_.Username
FROM Review Rev NATURAL JOIN Upvoted Uv JOIN User User_ ON Uv.UserID=User_.UserID
WHERE User_.Level >3 AND PostID IN
(SELECT New_table.PostID
FROM
(SELECT Rev1.PostID, COUNT(User1.UserID) AS no_of_up
FROM Review Rev1 JOIN Upvoted Uv1 JOIN User User1 ON Uv1.UserID=User1.UserID
GROUP BY Rev1.PostID
HAVING COUNT(User1.UserID) >=10) AS New_table);
```

<pre>mysql> SELECT Rev.PostID, Rev.Rating, UserUsername, Rev.Comment -> FROM Review Rev NATURAL JOIN Upvoted Uv JOIN User User_ ON Uv.UserID=UserUserID -> WHERE UserLevel > 3 AND PostID IN -> (SELECT New_table.PostID -> FROM</pre>
-> (SELECT Revl.PostID, COUNT(Userl.UserID) AS no_of_up -> FROM Review Revl JOIN Upvoted Uvl JOIN User Userl ON Uvl.UserID=Userl.UserID -> GROUP BY Revl.PostID -> HAVING COUNT(Userl.UserID) >=10) AS New_table);
PostID Rating Username Comment
<u> </u>
3981 7 fizzle Dominant Species lets you replay evolution. If that sounds grand, it is. The game is epic both in a the players strive to become the dominant species on the planet. They do this by placing workers, selecting various actions that allow the he main part of this battle for survival happens on the modular map. This part of the game employs a two-dimensional area control mecha adapted (or both, of course). The map play is highly interactive with everyone trying to broaden their niche, deliberately making condicies has a ton of moving parts which is the main reason a game takes as long as it does. Analyzing the current state of affairs (and drever well. There don't seem to be any dealbreakers, there are mitigation options for pretty much anything, and even the asymmetric set. At the end of the day it is still a multiplayer conflict game with all the issues that entails but it does also include a completely 3934 6 godfeather This game is more than a toy. While this is a typical Fantasy Flight fare with lots of decks and other um and Warrior Knights. Actually, I think this game in many places will replace Warrior Knights as while it has the same feel is nicer first time, but the mechanics are pretty straightforward. After you have chosen what to play during this round, most actions are prettend with Fantasy Flight Games with this and Chaos in the old world. The games are shorter in play time, while still feeling epic (maybe sto be pretty balanced, well at least there isn, Äöt a critical flaw with it. [b]Last updated:[/b] September 2010

Because the comment is too long to show the whole structure, we decided not to include the comment in the select clause:

b. Query 2: This advanced SQL shows the difference between PartyFriendly MAX and MIN's BoardGameProduct. Since there are many games in each family, we calculated the average of all games in each selected family for comparison.

```
(SELECT Familyname, Avg(g.Difficulty) as AvgDifficulty, Avg(g.Duration) as AvgDuration, Avg(g.Trading) as AvgTrading, PartyFriendly
FROM GameFamily f Natural JOIN BelongsTo b JOIN BoardgameProduct g on b.GameID = g.GameID
WHERE PartyFriendly = 10
Group By Familyname
)UNION(
SELECT Familyname, Avg(g.Difficulty) as AvgDifficulty, Avg(g.Duration) as AvgDuration, Avg(g.Trading) as AvgTrading, PartyFriendly
FROM GameFamily f Natural JOIN BelongsTo b JOIN BoardgameProduct g on b.GameID = g.GameID
WHERE PartyFriendly = 1
Group By Familyname
)
```

```
mysql> (SELECT Familyname, Avg(g.Difficulty) as AvgDifficulty, Avg(g.Duration) as AvgDuration, Avg(g.Trading) as AvgTrading, PartyFriendly -> FROM GameFamily f Natural JOIN BelongsTo b JOIN BoardgameProduct g on b.GameID = g.GameID
      -> WHERE PartyFriendly = 10
      -> Group By Familyname
-> LIMIT 10
     -> : DINTO TO
-> DINTO N(
-> SELECT Familyname, Avg(g.Difficulty) as AvgDifficulty, Avg(g.Duration) as AvgDuration, Avg(g.Trading) as AvgTrading, PartyFriendly
-> FROM GameFamily f Natural JOIN BelongsTo b JOIN BoardgameProduct g on b.GameID = g.GameID
-> WHERE PartyFriendly = 1
     -> Group By Familyname
-> LIMIT 10
 Familyname
                                                                                                  | AvgDifficulty | AvgDuration | AvgTrading | PartyFriendly |
                |heep
|: Give a Clue / Get a Clue
                                                                                                              2.0000 |
1.2727 |
                                                                                                                                 66.0000 |
35.1818 |
                                                                                                                                                   613.0000 | 10
400.5455 | 10
                |rs
|Two Player Only Games
|& Editions: Two-Player Versions of More-Player Games
                                                                                                              2.7143 | 2.3118 |
                                                                                                                                106.4286 |
62.3656 |
                                                                                                                                                   591.0000
331.7097
                                                                                                                                                                 | 10
| 10
                |icola
|s: Map (Continental / National scale)
                                                                                                              3.3333 I
                                                                                                                                100.0000 I
                                                                                                                                                  1269.0000
                                                                                                                                                                  | 10
                                                                                                              3.1000 |
                                                                                                                                121.5000
                                                                                                                                                   710.4667
                |s: Miniatures
|ing: Kickstarter
                                                                                                              2.9143 | 2.5814 |
                                                                                                                                129.3571 |
84.5736 |
                                                                                                                                                   590.2714 |
407.2171 |
                                                                                                                                                                    10
10
                  |rance
|a Select
                                                                                                              3.0588 |
1.9556 |
                                                                                                                                                   696.8824
415.9111
                                                                                                                                 43.2222 |
95.6635 |
                  |ames with Solitaire Rules
                  lorses
                                                                                                              3.3333 |
                                                                                                                                100.0000 |
                                                                                                                                                  1298.3333
                  |empest Shared World
                                                                                                              1.0000
                                                                                                                                                   355.0000
                                                                                                                                 25.0000 j
                  |Castles of Burgundy
| for the Galaxy
                                                                                                              3.0000
                                                                                                                                  75.0000
                                                                                                                                                   917.0000
                                                                                                              2.0000 |
2.0000 |
                                                                                                                                                   444.0000 |
756.0000 |
                                                                                                                                105.0000 |
                                                                                                                                 45.0000 |
                  |an Games (Lookout)
20 rows in set (0.01 sec)
```

3. Indexing

a. Advanced Query #1: The reason we will choose the following two indices (i.e., Level and Rating) is because for one thing, the user level is in Where clause, and it is not a Primary Key in the User table; for another, though the Rating is not a Primary Key in the Review table, Rating is an important attribute of the reviews and the ratings given by the commenters will be presented as important information with the reviews.

We can see that if we did not add any additional indices, the query execution time was 22.43 seconds. The system generated indices for both foreign keys (UserID and GameID).

Without any additional indices added manually, there is an index scan on User1 using Username_index. The estimated cost is 435.85 and rows are 4311. The actual time range is 0.037-1.063 and rows are 4311. The query used table scan and the estimated cost is 435.85 for scanning on table User_. For table scanning on the temporary table, the actual time range is 0.002-0.633 and rows are 8422.

After adding Index on Level and Rating respectively, the query execution time is 21.99 and 21.15 seconds. The performance seems to improve a bit for the rating index, but the Level index did not. The index scan on User1 using Level_index, the estimated cost is 438.85 and rows are 4311. The actual time range is 0.067-0.979 and rows are 4311. The overall cost and time of Level index is similar to the one without indices.

Specifically, if we add only Level as our additional index. We can see from the results that the system is using this index and the performance improved a little (Index scan on User1 using Level_index: cost=435.85, rows=4311; actual time upper bound is 0.999, rows=4311, loops=1). This is because the Username index generated automatically by the system has a longer upper bound compared to the Level index there. On the other hand, if we add only Rating as our additional index, the system seems not using the Rating index since there are no results show index scan using Rating_index. The same situation applies to when we used both of the indices.

When we added indices on both Level and Rating, the execution took 21.39 seconds, which is a bit better than the query without any indices. However, the effect is not significant in addition to the scans looking similar. The reason for this might be that when we create an index, the database backend will make some statistics by itself to decide whether it should use the index.

Also, we can see that the mechanism doing some joins used a Nested loop inner join, because it makes use of Index. For example, Username_ index here is a primary key.

This is the first advanced sql:

This is the indices in it:

	ow index from													
					Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Review Review Review		PRIMARY UserID GameID	į į	PostID UserID GameID	A A A	8537 4310 1062	NULL NULL	NULL NULL	 YES	BTREE BTREE BTREE			YES YES	NULL NULL
+ 3 rows in	set (0.01 sec	; ;)							·					

Without adding any other indices, here is the processing situation:

```
### SEPAIN ANALYZE SELECT Rev. PostiD, Rev. Mating, User_Username FROM Review Rev NATURAL JOIN Upvoted Uv JOIN User Userl ON VV. UserlD-User_UserlD WEEKE User_Level >3 AND PostID IN (SELECT Rev. Level) >4 AND PostID IN (SELECT Rev. L
```

```
-> Index lookup on New table using cauto heyb.* (PostID-Uv. PostID) (actual time=0.007..007 rows=1 loops=2)
-> Materialize (cost-0.00.00 rows=0) (actual time=22405.591 zows=8422 loops=1)
-> Piller: (count(Userl.UserID) >= 10) (actual time=22306.397. 22396.787 zows=8422 loops=1)
-> Table scan on temporary (actual time=0.002.0.663 zows=8422 loops=1)
-> Agyregate using temporary table (actual time=22396.391.22396.391.22397.801 zows=8422 loops=1)
-> Inner hash join (no condition) (cost-771199.76 cows=7165188) (actual time=13.144.6300.715 zows=70719536 loops=1)
-> Index scan on Rev! using GameID (cost-0.14 rows=8537) (actual time=0.044.31.020 rows=8422 loops=1)
-> Nash
-> Nested loop inner join (cost-2354.80 rows=8397) (actual time=0.044.31.020 rows=8497 loops=1)
-> Index acon on User! using Userane index (cost-5.85 rows=4311) (actual time=0.037.10.63 zows=4311 loops=1)
-> Index lookup on Uv! using UserID (UserID=UserI.UserID) (cost=0.25 rows=2) (actual time=0.001.0.002 rows=2 loops=4311)
```

- $| -> Nested\ loop\ semijoin\ \ (cost=300746.19\ rows=0)\ (actual\ time=22412.985..22428.082\ rows=2\ loops=1)$
 - -> Nested loop inner join (cost=2087.78 rows=140) (actual time=7.397..22.487 rows=2 loops=1)
 - -> Nested loop inner join (cost=1075.43 rows=2799) (actual time=0.068..10.749 rows=5420 loops=1)
 - $\hbox{-> Filter: (User_.`Level`>3) \ (cost=435.85\ rows=1437)\ (actual\ time=0.052...2.882\ rows=2845\ loops=1)}\\$
 - -> Table scan on User_ (cost=435.85 rows=4311) (actual time=0.049..2.100 rows=4311 loops=1)

- -> Index lookup on Uv using UserID (UserID=User_.UserID) (cost=0.25 rows=2) (actual time=0.002..0.002 rows=2 loops=2845)
- -> Filter: (Rev.UserID = User_.UserID) (cost=0.26 rows=0) (actual time=0.002..0.002 rows=0 loops=5420)
- -> Single-row index lookup on Rev using PRIMARY (PostID=Uv.PostID) (cost=0.26 rows=1) (actual time=0.002..0.002 rows=1 loops=5420)
- -> Index lookup on New_table using <auto_key0> (PostID=Uv.PostID) (actual time=0.007..0.007 rows=1 loops=2)
 - -> Materialize (cost=0.00..0.00 rows=0) (actual time=22405.591..22405.591 rows=8422 loops=1)
 - -> Filter: (count(User1.UserID) >= 10) (actual time=22396.397..22398.787 rows=8422 loops=1)
 - -> Table scan on <temporary> (actual time=0.002..0.633 rows=8422 loops=1)
 - -> Aggregate using temporary table (actual time=22396.391..22397.801 rows=8422 loops=1)
 - -> Inner hash join (no condition) (cost=7171199.76 rows=71685188) (actual time=13.144..6300.715 rows=70719536 loops=1)
 - -> Index scan on Rev1 using GameID (cost=0.14 rows=8537) (actual time=0.044..31.020 rows=8422 loops=1)
 - -> Hash
 - -> Nested loop inner join (cost=2354.80 rows=8397) (actual time=0.049..11.743 rows=8397 loops=1)
 - -> Index scan on User1 using Username_index (cost=435.85 rows=4311) (actual time=0.037..1.063 rows=4311 loops=1)
 - -> Index lookup on Uv1 using UserID (UserID=User1.UserID) (cost=0.25 rows=2) (actual time=0.001..0.002 rows=2

loops=4311)

+------

1 row in set (22.43 sec)

Then add an index on Level:

mysql> CREATE INDEX Level_index on User(Level);
Query OK, 0 rows affected (0.15 sec)
Records: 0 Duplicates: 0 Warnings: 0

	how index from	u User;												
Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
User	i 0 i	PRIMARY Username index		UserID	A .	4311 4308	NULL NULL	NULL	i	BTREE BTREE	+	+ 	YES	NULL
User		Level_index		Level		10	NULL	NULL	YES	BTREE	<u> </u>	i	YES	NULL
3 rows in	n set (0.01 se													

```
| -> Nested loop semijoin (cost=595055.48 rows=0) (actual time=21975.831..21987.758 rows=2 loops=1)
| -> Nested loop inner join (cost=3706.71 rows=277) (actual time=0.794..17.717 rows=2 loops=1)
| -> Nested loop inner join (cost=4702.24 rows=542) (actual time=0.033..8.775 rows=3420 loops=1)
| -> Nested loop inner join (cost=4702.45 rows=5485) (actual time=0.042..2.186 rows=3411 loops=1)
| -> Filter: (Reviser) = User (cost=435.85 rows=318) (actual time=0.042..2.186 rows=3181 loops=1)
| -> Index lookup on Uv using UserID (UserID=User_UserID) (cost=0.25 rows=2) (actual time=0.001..0.002 rows=2 loops=2845)
| -> Filter: (ReviserID = User_UserID) (cost=0.25 rows=2) (actual time=0.001..0.001 rows=0 loops=5420)
| -> Single-row index lookup on Rev using PRIMARY (FostID=VorStID) (cost=0.25 rows=2) (actual time=0.001..0.001 rows=1 loops=5420)
| -> Index lookup on Nev table using quate_key0 (FostID=VorStID) (actual time=0.001..0.001 rows=0 loops=5420)
| -> Materialize (Cost=0.00..0.00 rows=0) (actual time=2.001.0.3...21971039 rows=4822 loops=1)
| -> Filterialize (Cost=0.00..0.00 rows=0.002...2011 time=0.002...3...21971039 rows=4822 loops=1)
| -> Aggregate using temporary table (actual time=2.096.622...21962.837 rows=6422 loops=1)
| -> Aggregate using temporary table (actual time=2.096.622...21962.837 rows=6422 loops=1)
| -> Index acan on RevI using GameID (cost=0.14 rows=8537) (actual time=0.003...28.264 rows=8397 loops=1)
| -> Nested loop inner join (cost=0.25*108*7*rows=6327) (actual time=0.003...28.264 rows=8422 loops=1)
| -> Index acan on UserI using Level index (cost=435.8 rows=431) (actual time=0.006.0.099 rows=4311 loops=1)
| -> Index lookup on UvI using UserID (UserID=UserI.UserID) (cost=0.25*rows=2) (actual time=0.001..0.002 rows=2 loops=4311)
| -> Index lookup on UvI using UserID (UserID=UserI.UserID) (cost=0.25*rows=2) (actual time=0.001..0.002 rows=2 loops=4311)
```

- | -> Nested loop semijoin (cost=595055.48 rows=0) (actual time=21975.831..21987.758 rows=2 loops=1)
 - -> Nested loop inner join (cost=3706.71 rows=277) (actual time=5.794..17.717 rows=2 loops=1)

-> Nested loop inner join (cost=1702.24 rows=5542) (actual time=0.053..8.775 rows=5420 loops=1) -> Filter: (User .`Level` > 3) (cost=435.85 rows=2845) (actual time=0.042..2.186 rows=2845 loops=1) -> Table scan on User_ (cost=435.85 rows=4311) (actual time=0.039..1.545 rows=4311 loops=1) -> Index lookup on Uv using UserID (UserID=User_.UserID) (cost=0.25 rows=2) (actual time=0.001..0.002 rows=2 loops=2845) -> Filter: (Rev.UserID = User_.UserID) (cost=0.26 rows=0) (actual time=0.001..0.001 rows=0 loops=5420) -> Single-row index lookup on Rev using PRIMARY (PostID=Uv.PostID) (cost=0.26 rows=1) (actual time=0.001..0.001 rows=1 loops=5420) -> Index lookup on New_table using <auto_key0> (PostID=Uv.PostID) (actual time=0.006..0.006 rows=1 loops=2) -> Materialize (cost=0.00..0.00 rows=0) (actual time=21970.039..21970.039 rows=8422 loops=1) -> Filter: (count(User1.UserID) >= 10) (actual time=21961.628..21963.830 rows=8422 loops=1) -> Table scan on <temporary> (actual time=0.002..0.506 rows=8422 loops=1) -> Aggregate using temporary table (actual time=21961.622..21962.837 rows=8422 loops=1) -> Inner hash join (no condition) (cost=7171199.76 rows=71685188) (actual time=11.569..6038.719 rows=70719536 loops=1) -> Index scan on Rev1 using GameID (cost=0.14 rows=8537) (actual time=0.030..28.264 rows=8422 loops=1) -> Hash -> Nested loop inner join (cost=2354.80 rows=8397) (actual time=0.043..10.623 rows=8397 loops=1) -> Index scan on User1 using Level index (cost=435.85 rows=4311) (actual time=0.036..0.999 rows=4311 loops=1) -> Index lookup on Uv1 using UserID (UserID=User1.UserID) (cost=0.25 rows=2) (actual time=0.001..0.002 rows=2 loops=4311)

1 row in set (21.99 sec)

If we add an index on only Rating:

Query OK,	0 rows affect	ing_index on Reced (0.09 sec): 0 Warnings: (
	ow index from													
Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Review Review Review Review	0 1 1	PRIMARY UserID GameID Rating_index		PostID UserID GameID	A A A	8537 4310 1062 152	NULL	NULL NULL NULL NULL	 YES 	BTREE BTREE BTREE BTREE	 		YES YES YES	NULL NULL NULL
	set (0.00 sec		+		+	+	+		+		+	+	+	+

mysql> sh	ow index from	User;												
++														
	Non_unique											Index_comment		
User User		PRIMARY Username_index		UserID Username	i A i A	4311 4308	NULL NULL	NULL		BTREE BTREE			YES	NULL
	set (0.00 sec				*		+		+		*		+	+
	ow index from	n Review;												
Table	Non_unique		Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Review		PRIMARY				8537	NULL	NULL		BTREE			YES	NULL
Review		UserID		UserID		4310	NULL	NULL	YES	BTREE			YES	NULL
Review		GameID		GameID		1062	NULL	NULL		BTREE			YES	NULL
Review		Rating_index		Rating			NULL	NULL	YES	BTREE			YES	NULL
	set (0.00 sec													

4 LOWS III Set (0.00 Set)	
mysql- EXPLAIN ANALYZE SELECT Rev.PostID, Rev.Rating, User_Username FROM Review Rev NATURAL JOIN Upvoted Uv JOIN User User_ON Uv.UserID-User_UserID Nat no_of_up FROM Review RevI JOIN Upvoted Uvi JOIN User User ON Uv.UserID-UserID UserI.UserID AS no_of_up FROM Review RevI JOIN Upvoted Uvi JOIN User UserI ON Uv!UserID-UserID.UserID.>=10) AS New_table);	WHERE UserLevel >3 AND PostID IN (SELECT UserID GROUP BY Rev1.PostID HAVING COUNT(
EXPLAIN	
Partibilit	
l e la companya de l	
+	
	
-> Nested loop semijoin (cost=300746.19 rows=0) (actual time=21124.09421136.310 rows=2 loops=1) -> Nested loop inner join (cost=20078.78 rows=140) (actual time=5.698.17.909 rows=2 loops=1) -> Nested loop inner join (cost=1075.43 rows=2799) (actual time=0.07448.974 rows=5420 loops=1) -> Filter: (Usez'ivee': > 3) (cost=4735.85 rows=4311) (actual time=0.0552.219 rows=2845 loops=1) -> Table scan on User_ (cost=435.85 rows=4311) (actual time=0.0561582 rows=4311 loops=1) -> Index lookup on Uv using UserID (UserID=User_UserID) (cost=0.25 rows=0.25 rows=0.25 rows=0.010.002 rows=0.0010.002 rows=2 loops=2845) -> Filter: (Rev_UserID = UserUserID) (cost=0.26 rows=0) (actual time=0.0010.01 rows=0 loops=5420) -> Single-row index lookup on Rev using PRIMARY (FOSID=Uv-PosID) (cost=0.26 rows=1) (actual time=0.0010.001 rows=0 loops=5420)	
-> Index lookup on New_table using <auto_key0> (PostID=Uv.PostID) (actual time=0.0050.005 rows=1 loops=2)</auto_key0>	

-> Materialize (cost=0.00.1.00 rows=0) (actual time=2110.1302110.130 rows=6422 loops=1) -> Filter: (count(UserI.UserID) >= 10) (actual time=2110.13021112.282 rows=6422 loops=1) -> Table scan on <temporary> (actual time=0.0020.500 rows=6422 loops=1)</temporary>
-> Aggregate using temporary table (actual time=21110.12521111.344 rows=8422 loops=1) -> Inner hash join (no condition) (cost=7171199.76 rows=71685188) (actual time=11.1325818.453 rows=70719536 loops=1) -> Index soan on Rev! using Gamel (cost=0.14 rows=8537) (actual time=0.125.8.742 rows=8422 loops=1)
-> Hash -> Nested loop inner join (cost-2354.80 rows-8397) (actual time=0.04810.086 rows=8397 loops=1)
-> Index scan on Userl using Username_index (cost=435.85 rows=4311) (actual time=0.0390.936 rows=4311 loops=1) -> Index lookup on Uv1 using UserID (UserID=UserI.UserID) (cost=0.25 rows=2) (actual time=0.0010.002 rows=2 loops=4311)
1 row in set (21.15 sec)
-> Nested loop sernijoin (cost=200740.19 fows=0) (actual time=21124.09421130.310 fows=2 loops=1)
-> Nested loop inner join (cost=1075.43 rows=2799) (actual time=0.0748.974 rows=5420 loops=1)
-> Filter: (User`Level` > 3) (cost=435.85 rows=1437) (actual time=0.0740.974 rows=2845 loops=1)
-> Filter. (User Lever > 3) (cost=435.85 rows=1437) (actual time=0.0561.582 rows=2645 100ps=1) -> Table scan on User_ (cost=435.85 rows=4311) (actual time=0.0561.582 rows=4311 loops=1)
-> lable scan on Osei_ (cost=435.65 lows=4311) (actual time=0.056.1.362 lows=4311 loops=1) -> Index lookup on Uv using UserID (UserID=User_UserID) (cost=0.25 rows=2) (actual time=0.0010.002 rows=2)
-> index lookup on ov using oserib (oserib-oseroserib) (cost-o.25 fows-2) (actual time-o.0010.002 fows-2 tops=2845)
-> Filter: (Rev.UserID = User .UserID) (cost=0.26 rows=0) (actual time=0.0010.001 rows=0 loops=5420)
-> Single-row index lookup on Rev using PRIMARY (PostID=Uv.PostID) (cost=0.26 rows=1) (actual time=0.0010.001
rows=1 loops=5420)
-> Index lookup on New_table using <auto_key0> (PostID=Uv.PostID) (actual time=0.0050.005 rows=1 loops=2)</auto_key0>
-> Materialize (cost=0.000.00 rows=0) (actual time=21118.39821118.398 rows=8422 loops=1)
-> Filter: (count(User1.UserID) >= 10) (actual time=21110.13021112.282 rows=8422 loops=1)
-> Table scan on <temporary> (actual time=0.0020.508 rows=8422 loops=1)</temporary>
-> Aggregate using temporary table (actual time=21110.12521111.344 rows=8422 loops=1)
-> Inner hash join (no condition) (cost=7171199.76 rows=71685188) (actual time=11.1325818.453 rows=7071953)
1005 1005
-> Index scan on Rev1 using GameID (cost=0.14 rows=8537) (actual time=0.0258.742 rows=8422 loops=1)
-> mack scan on Nev r using Gamero (cost-0.14 rows-0307) (actual time-0.0200.742 rows-0422 roops-1) -> Hash
-> Nested loop inner join (cost=2354.80 rows=8397) (actual time=0.04810.086 rows=8397 loops=1)
-> Index scan on User1 using Username_index (cost=435.85 rows=4311) (actual time=0.0390.936
rows=4311 loops=1)
-> Index lookup on Uv1 using UserID (UserID=User1.UserID) (cost=0.25 rows=2) (actual time=0.0010.002
rows=2 loops=4311)
i '
 -
†
+

-> Index lookup on New_table using <auto_key0> (PostID-UV.PostID) (actual time=0.005..0.005 rows=1 loops=2)

1 row in set (21.15 sec)

If we add indices both on Level and Rating:

```
mysql> show index from Review;
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
| Review | 0 | PRIMARY | 1 | PostID | A | 8537 | NULL | NULL | BTREE | YES | NULL |
| Review | 1 | UserID | 1 | UserID | A | 4310 | NULL | NULL | BTREE | YES | NULL |
| Review | 1 | GameID | 1 | GameID | A | 1062 | NULL | NULL | BTREE | YES | NULL |
| Review | 1 | Rating_index | 1 | Rating | A | 152 | NULL | NULL | YES | BTREE | YES | NULL |
| A rows in set (0.00 sec) |
| mysql> show index from User; |
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
| User | 0 | PRIMARY | 1 | UserID | A | 4311 | NULL | NULL | BTREE | YES | NULL |
| User | 1 | Username | A | 4308 | NULL | NULL | BTREE | YES | NULL |
| User | 1 | Username | A | 4308 | NULL | NULL | BTREE | YES | NULL |
| User | 1 | Username | A | 4308 | NULL | NULL | BTREE | YES | NULL |
| User | 1 | Username | A | 4308 | NULL | NULL | NULL | BTREE | YES | NULL |
| User | 1 | Username | A | 4308 | NULL | NULL | NULL | BTREE | YES | NULL |
| User | 1 | Username | A | 4308 | NULL | NULL | NULL | BTREE | YES | NULL |
| User | 1 | Username | A | 4308 | NULL | NULL | NULL | NULL | NULL |
| User | 1 | Username | A | 4308 | NULL | NULL | NULL | NULL | NULL |
| User | 1 | Username | A | 4308 | NULL | NULL | NULL | NULL | NULL |
| User | 1 | Username | A | 4308 | NULL | NULL | NULL | NULL | NULL | NULL |
| User | 1 | Username | A | 4308 | NULL | NULL | NULL | NULL | NULL | NULL |
| User | 1 | Username | A | 4308 | NULL | NULL | NULL | NULL | NULL | NULL |
| User | 1 | Username | A | 4308 | NULL | NULL | NULL | NULL | NULL | NULL |
| User | 1 | Username | NULL |
| User | 1 | Username | NULL | NUL
```

```
-> Index lookup on New table using cauto key9' (PostID=UV PostID) (cctual time=0.005 0.005 now=1 loops=2)

-> Materialize (Cost=0.00 0.00 row=0) (actual time=0.1345.93, 2135c.716 rows=8422 loops=1)

-> Filter (count(Useri UserID) == 10) (actual time=0.21345.93, 2135c.716 rows=8422 loops=1)

-> Table scan on temporary table (actual time=0.010.0.093 rows=8422 loops=1)

-> Aggregate using temporary table (actual time=0.21345.808 rows=8422 loops=1)

-> Inner hash join (no condition) (cost=0.14 rows=8537) (actual time=0.032s.9.412 rows=8422 loops=1)

-> Index scan on Rev1 using GameID (cost=0.14 rows=8537) (actual time=0.028..9.412 rows=8422 loops=1)

-> Nested loop inner join (cost=2354.80 rows=8397) (actual time=0.075...0.290 rows=8397 loops=1)

-> Index scan on User1 using Level index (cost=435.85 rows=4311) (actual time=0.067..0.979 rows=4311 loops=1)

-> Index lookup on Uv1 using UserID (UserID=UserI).UserID) (cost=0.25 rows=2) (actual time=0.001..0.002 rows=2 loops=4311)

1 row in set (21.39 sec)
```

- $| -> Nested \ loop \ semijoin \ \ (cost=595055.48 \ rows=0) \ (actual \ time=21369.076..21380.940 \ rows=2 \ loops=1)$
- -> Nested loop inner join (cost=3706.71 rows=277) (actual time=6.286..18.145 rows=2 loops=1)
 - -> Nested loop inner join (cost=1702.24 rows=5542) (actual time=0.057..8.925 rows=5420 loops=1)
 - -> Filter: (User_.`Level` > 3) (cost=435.85 rows=2845) (actual time=0.045..2.229 rows=2845 loops=1)
 - -> Table scan on User (cost=435.85 rows=2311) (actual time=0.043..1.595 rows=4311 loops=1)
 - -> Index lookup on Uv using UserID (UserID=User_.UserID) (cost=0.25 rows=2) (actual time=0.001..0.002 rows=2 loops=2845)
 - -> Filter: (Rev.UserID = User_.UserID) (cost=0.26 rows=0) (actual time=0.002..0.002 rows=0 loops=5420)
 - -> Single-row index lookup on Rev using PRIMARY (PostID=Uv.PostID) (cost=0.26 rows=1) (actual time=0.001..0.001 rows=1 loops=5420)
- -> Index lookup on New table using <auto key0> (PostID=Uv.PostID) (actual time=0.005..0.005 rows=1 loops=2)
 - -> Materialize (cost=0.00..0.00 rows=0) (actual time=21362.791..21362.791 rows=8422 loops=1)
 - -> Filter: (count(User1.UserID) >= 10) (actual time=21354.583..21356.716 rows=8422 loops=1)
 - -> Table scan on <temporary> (actual time=0.001..0.493 rows=8422 loops=1)
 - -> Aggregate using temporary table (actual time=21354.578..21355.808 rows=8422 loops=1)
 - -> Inner hash join (no condition) (cost=7171199.76 rows=71685188) (actual time=11.300..5771.184 rows=70719536 loops=1)
 - -> Index scan on Rev1 using GameID (cost=0.14 rows=8537) (actual time=0.028..9.412 rows=8422 loops=1)
 - -> Hash
 - -> Nested loop inner join (cost=2354.80 rows=8397) (actual time=0.075..10.290 rows=8397 loops=1)
 - -> Index scan on User1 using Level_index (cost=435.85 rows=4311) (actual time=0.067..0.979 rows=4311 loops=1)
 - -> Index lookup on Uv1 using UserID (UserID=User1.UserID) (cost=0.25 rows=2) (actual time=0.001..0.002 rows=2

loops=4311)

+------

1 row in set (21.39 sec)

b. Advanced Query #2: The reason why we choose these three indices (BoardgameProduct.Difficulty, BoardgameProduct.Duration and GameFamily.PartyFriendly) is because both BoardgameProduct.Difficulty and BoardgameProduct.Duration are important information with the game and in this SQL, which help us compare the differences between boardgames in different families. And GameFamily.PartyFriendly, is an important attribute reflecting the popularity of Boardgames belonging to a family.

As we can see from the initial performance, the time taken to implement this SQL is short and there is little room for improvement. There is little difference between PartyFriendly=10 and PartyFriendly=1, except that PartyFriendly=10 takes a little more time than PartyFriendly=1.

After we added Difficulty as the index, we can see that the SQL execution doesn't seem to have changed, and Duration as the index doesn't seem to work in this SQL.

After we added Duration as the index, we can see that the SQL seems to take longer to execute, and Table scan time is shortened when Partyfriendly=10, but increased when Partyfriendly=1. And Duration as the index doesn't seem to work in this SQL.

After adding PartyFriendly as index, we can see that the runtime of the SQL does not change significantly, and table scan time compared with the original, when Partyfriendly=10, time increased, but when Partyfriendly=1, time decreased.

We can see that before adding index on Difficulty, the execution time is 0.01seconds, and add index idx_difficulty, the performance has no change, the execution time is still 0.01 seconds. For the attribute of Difficulty, it is not in the WHERE clause and no tables joining on this attribute. We can see that before and after index, the database did not use this index, and use table scan and look up using the primary key. Specifically, before using index on Difficulty, the cost of table scan on f estimated 155.25, and rows are 1535. The actual time range is 0.051..0.727 and rows are 153. After using the index, the cost and rows are not changed. Also, the actual time is no different. So far, the index we've added has not contributed to performance for the time being. Therefore, we might consider adding some other indices to improve the performance in the future.

 $(SELECT\ Family name,\ Avg(g.Difficulty)\ as\ AvgDifficulty,\ Avg(g.Duration)\ as\ AvgDuration,\ Avg(g.Trading)\ as\ AvgTrading,\ PartyFriendly$

FROM GameFamily f Natural JOIN BelongsTo b JOIN BoardgameProduct g on b.GameID = g.GameID WHERE PartyFriendly = 10

Group By Familyname

)UNION(

 $SELECT\ Family name,\ Avg(g.Difficulty)\ as\ AvgDifficulty,\ Avg(g.Duration)\ as\ AvgDuration,\ Avg(g.Trading)\ as\ AvgTrading,\ PartyFriendly$

```
FROM GameFamily f Natural JOIN BelongsTo b JOIN BoardgameProduct g on b.GameID = g.GameID WHERE PartyFriendly = 1
Group By Familyname
)
```

INITIAL INDEX and EXPLAIN ANALYZE:

mysql> SHOW INDEX FROM BoardgameProduct:

++ Table xpression	Non_unique	Key_name	+	+ Column_name	Collation	Cardinality	Sub_part	 Packed	Null Index	type Com	ment Index	comment	Visible	
BoardgameProduct ULL	0	PRIMARY				1008			BTREE	: 1	l t		YES	N
1 row in set (0.00 s	sec)								·		·			
mysql> SHOW INDEX FR	OM GameFamily	/; +			+		+	+	+	++		+	+	
+ Table Non_u ion	nique Key_n		in_index Colur								Index_comme	ent Visil	ole Expr	ess
+ GameFamily	0 PRIMA	LRY	1 Fami:	lyID A		1535	NULL NU	TT	BTREE			YES	NULL	
++ 1 row in set (0.01 s	:ec)	+					+	+	+	++		+	+	
mysql> EXPLAIN AN ly -> FROM GameF -> WHERE Fart -> Group By F -> JUNION(-> SELECT Fam -> FROM GameF -> WHERE Fart -> Group By F ->) -> -> ->;	amily f Natu yFriendly = amilyname ilyname, Ava amily f Natu yFriendly =	ural JOIN 10 g(g.Difficural JOIN	BelongsTo b o	JOIN Boardga	meProduct o	g on b.GameII	D = g.Game	ID A v g (g.Tra					artyFrie	ad
EXPLAIN														
+														

```
Table scan on <union temporary> (cost=2.50..2.50 rows=0) (actual time=0.001..0.030 rows=295 loops=1)

-> Table scan on <temporary> (actual time=0.002..0.030 rows=142 loops=1)

-> Aggregate using temporary table (actual time=3.932..4.022 rows=142 loops=1)

-> Nested loop inner join (cost=25.51 rows=514) (actual time=0.110..3.047 rows=635 loops=1)

-> Nested loop inner join (cost=245.12 rows=514) (actual time=0.095..1.486 rows=635 loops=1)

-> Table scan on f (cost=125.25 rows=1536) (actual time=0.050..1.486 rows=635 loops=1)

-> Table scan on f (cost=155.25 rows=1535) (actual time=0.069..0.763 rows=142 loops=1)

-> Table scan on f (cost=155.25 rows=1535) (actual time=0.061..0.592 rows=1535 loops=1)

-> Index lookup on b using PRIMARY (FamilyID=f.FamilyID) (cost=0.25 rows=3) (actual time=0.003..0.004 rows=4 loops=142)

-> Single-row index lookup on g using PRIMARY (GameID=b.GameID) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=635)

-> Table scan on <temporary table (actual time=3.734..3.763 rows=153 loops=1)

-> Nested loop inner join (cost=25.12 rows=514) (actual time=0.066..2.748 rows=683 loops=1)

-> Nested loop inner join (cost=25.25 rows=153) (actual time=0.077..0.720 rows=153 loops=1)

-> Table scan on f (cost=125.25 rows=153) (actual time=0.047..0.720 rows=153 loops=1)

-> Table scan on f (cost=125.25 rows=153) (actual time=0.047..0.720 rows=153 loops=1)

-> Table scan on f (cost=155.25 rows=153) (actual time=0.047..0.720 rows=153 loops=1)

-> Table scan on f (cost=155.25 rows=153) (actual time=0.047..0.720 rows=153 loops=1)

-> Table scan on f (cost=155.25 rows=153) (actual time=0.057..1.371 rows=683 loops=1)

-> Table scan on f (cost=155.25 rows=153) (actual time=0.057..1.371 rows=683 loops=1)

-> Table scan on f (cost=155.25 rows=153) (actual time=0.057..1.371 rows=683 loops=1)

-> Table scan on f (cost=155.25 rows=153) (actual time=0.057..1.371 rows=683 loops=1)

-> Single-row index lookup on guing PRIMARY (GamilyID=f.FamilyID) (cost=0.25 rows=3) (actual time=0.002..0.004 rows=4 loops=153)
     row in set (0.01 sec)
| -> Table scan on <union temporary> (cost=2.50..2.50 rows=0) (actual time=0.001..0.030 rows=295 loops=1)
     -> Union materialize with deduplication (cost=2.50..2.50 rows=0) (actual time=9.092..9.144 rows=295 loops=1)
         -> Table scan on <temporary> (actual time=0.002..0.024 rows=142 loops=1)
              -> Aggregate using temporary table (actual time=4.688..4.723 rows=142 loops=1)
                   -> Nested loop inner join (cost=425.12 rows=514) (actual time=0.726..3.668 rows=635 loops=1)
                       -> Nested loop inner join (cost=245.20 rows=514) (actual time=0.690..2.062 rows=635 loops=1)
                         -> Filter: (f.PartyFriendly = 10) (cost=155.25 rows=154) (actual time=0.670..1.364 rows=142 loops=1)
                                 -> Table scan on f (cost=155.25 rows=1535) (actual time=0.658..1.180 rows=1535 loops=1)
                            -> Index lookup on b using PRIMARY (FamilyID=f.FamilyID) (cost=0.25 rows=3) (actual time=0.002..0.004 rows=4 loops=142)
                       -> Single-row index lookup on g using PRIMARY (GameID=b.GameID) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=635)
         -> Table scan on <temporary> (actual time=0.001..0.018 rows=153 loops=1)
               -> Aggregate using temporary table (actual time=3.874..3.903 rows=153 loops=1)
                          -> Nested loop inner join (cost=425.12 rows=514) (actual time=0.073..2.863 rows=683 loops=1)
                                -> Nested loop inner join (cost=245.20 rows=514) (actual time=0.064..1.403 rows=683 loops=1)
                                    -> Filter: (f.PartyFriendly = 1) (cost=155.25 rows=154) (actual time=0.051..0.727 rows=153 loops=1)
                                          -> Table scan on f (cost=155.25 rows=1535) (actual time=0.043..0.542 rows=1535 loops=1)
                                      -> Index lookup on b using PRIMARY (FamilyID=f.FamilyID) (cost=0.25 rows=3) (actual time=0.002..0.004 rows=4 loops=153)
                               -> Single-row index lookup on g using PRIMARY (GameID=b.GameID) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1
                     loops=683)
                     1 row in set (0.01 sec)
```

ADD Difficulty index on BoardgameProduct:

mysql> CREATE INDEX idx_difficulty ON BoardgameProduct(Difficulty);
Query OK, 0 rows affected (0.07 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> SHOW INDX FF ERROR 1064 (42000): duct' at line 1 mysql> SHOW INDEX F	: You have an e	error in your SQL Product;											
	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visib
BoardgameProduct NULL BoardgameProduct NULL		PRIMARY		Difficulty		1008	NULL	NULL	YES				YES
-> WHERE PartyI -> Group By Fam ->)UNION(-> SELECT Famil	CYZE (SELECT Faily f Natural Priendly = 10 milyname Lyname, Avg(g. Raily f Natural Priendly = 1	umilyname, Avg(g.) JOIN BelongsTo b Difficulty) as Av. JOIN BelongsTo b	JOIN Boardgame	Product g on b.	.GameID = g.0 as AvgDuratio	GameID on, Avg(g.Tradi					tyFriendly		

EXPLAIN	
+	
-> Table scan on <union temporary=""> (cost=2.502.50 rows=0) (actual time=0.0010.030 rows=295 loops=1)</union>	
-> Union materialize with deduplication (cost=2.502.50 rows=0) (actual time=9.0929.144 rows=295 loops=1)	
-> Table scan on <temporary> (actual time=0.0020.024 rows=142 loops=1)</temporary>	
-> Aggregate using temporary table (actual time=4.6884.723 rows=142 loops=1)	
-> Nested loop inner join (cost=425.12 rows=514) (actual time=0.7263.668 rows=635 loops=1)	
-> Nested loop inner join (cost=245.20 rows=514) (actual time=0.6902.062 rows=635 loops=1)	
	40.3
-> Filter: (f.PartyFriendly = 10) (cost=155.25 rows=154) (actual time=0.6701.364 rows=14	42 100ps=1)

```
Table scan on <union temporary> (cost=2.50..2.50 rows=0) (actual time=0.001..0.030 rows=295 loops=1)

> Union materialize with deduplication (cost=2.50..2.50 rows=0) (actual time=9.092..9.144 rows=295 loops=1)

-> Table scan on <temporary> (actual time=0.002..0.024 rows=142 loops=1)

-> Aggregate using temporary table (actual time=4.688..4.723 rows=142 loops=1)

-> Nested loop inner join (cost=425.12 rows=514) (actual time=0.726..3.668 rows=635 loops=1)

-> Nested loop inner join (cost=245.20 rows=514) (actual time=0.690..2.062 rows=635 loops=1)

-> Filter: (f.PartyFriendly = 10 (cost=155.25 rows=154) (actual time=0.670..1.364 rows=142 loops=1)

-> Table scan on f (cost=155.25 rows=1535) (actual time=0.658..1.180 rows=1535 loops=1)

-> Index lookup on b using PRIMARY (FamilyID=f.FamilyID) (cost=0.25 rows=3) (actual time=0.002..0.004 rows=4 loops=142)

-> Single-row index lookup on q using PRIMARY (GameID=b.GameID) (cost=0.25 rows=3) (actual time=0.002..0.002 rows=1 loops=63
                           -> Index lookup on b using PRIMARY (FamilyID=f.FamilyID) (cost=0.25 rows=3) (actual time=0.002..0.004 rows=4 loops=142)
-> Single-row index lookup on g using PRIMARY (GameID=b.GameID) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=635)
le scan on <temporary> (actual time=0.001..0.018 rows=153 loops=1)
Aggregate using temporary table (actual time=3.874..3.903 rows=153 loops=1)
-> Nested loop inner join (cost=425.12 rows=514) (actual time=0.073..2.863 rows=683 loops=1)
-> Nested loop inner join (cost=245.20 rows=514) (actual time=0.064..1.403 rows=683 loops=1)
-> Filter: (f.PartyFriendly = 1) (cost=155.25 rows=154) (actual time=0.051..0.727 rows=153 loops=1)
-> Table scan on f (cost=155.25 rows=1535) (actual time=0.043..0.542 rows=1536 loops=1)
-> Index lookup on b using PRIMARY (FamilyID=f.FamilyID) (cost=0.25 rows=3) (actual time=0.002..0.004 rows=4 loops=153)
-> Single-row index lookup on g using PRIMARY (GameID=b.GameID) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=683)
| -> Table scan on <union temporary> (cost=2.50..2.50 rows=0) (actual time=0.001..0.030 rows=295 loops=1)
    -> Union materialize with deduplication (cost=2.50..2.50 rows=0) (actual time=9.092..9.144 rows=295 loops=1)
        -> Table scan on <temporary> (actual time=0.002..0.024 rows=142 loops=1)
             -> Aggregate using temporary table (actual time=4.688..4.723 rows=142 loops=1)
                 -> Nested loop inner join (cost=425.12 rows=514) (actual time=0.726..3.668 rows=635 loops=1)
                      -> Nested loop inner join (cost=245.20 rows=514) (actual time=0.690..2.062 rows=635 loops=1)
                           -> Filter: (f.PartyFriendly = 10) (cost=155.25 rows=154) (actual time=0.670..1.364 rows=142 loops=1)
                               -> Table scan on f (cost=155.25 rows=1535) (actual time=0.658..1.180 rows=1535 loops=1)
                           -> Index lookup on b using PRIMARY (FamilyID=f.FamilyID) (cost=0.25 rows=3) (actual time=0.002..0.004 rows=4 loops=142)
                      -> Single-row index lookup on g using PRIMARY (GameID=b.GameID) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=635)
         -> Table scan on <temporary> (actual time=0.001..0.018 rows=153 loops=1)
             -> Aggregate using temporary table (actual time=3.874..3.903 rows=153 loops=1)
                  -> Nested loop inner join (cost=425.12 rows=514) (actual time=0.073..2.863 rows=683 loops=1)
                      -> Nested loop inner join (cost=245.20 rows=514) (actual time=0.064..1.403 rows=683 loops=1)
                           -> Filter: (f.PartyFriendly = 1) (cost=155.25 rows=154) (actual time=0.051..0.727 rows=153 loops=1)
                               -> Table scan on f (cost=155.25 rows=1535) (actual time=0.043..0.542 rows=1535 loops=1)
                           -> Index lookup on b using PRIMARY (FamilyID=f.FamilyID) (cost=0.25 rows=3) (actual time=0.002..0.004 rows=4 loops=153)
                      -> Single-row index lookup on g using PRIMARY (GameID=b.GameID) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=683)
```

DROP Difficulty index, ADD Duration index on BoardgameProduct:

mysql> DROP INDEX in Query OK, 0 rows af Records: 0 Duplica	fected (0.04 s	sec)	oduct;										
mysql> CREATE INDEX Query OK, 0 rows af Records: 0 Duplica	fected (0.05 s	sec)	oduct(Duration)										
mysql> SHOW INDEX F													
++ Table Expression	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible
-++ BoardgameProduct		PRIMARY			A	1008		NULL		BTREE	ı		YES
BoardgameProduct NULL		idx_duration			A	34		NULL		BTREE			YES
-++ 2 rows in set (0.01				+	+	†	+				 		

mysql> 1	XPLAIN ANALYZE (SELECT Familyname, Avg(g.Difficulty) as AvgDifficulty, Avg(g.Duration) as AvgDuration, Avg(g.Trading) as AvgTrading, PartyFriend
-> 1	ROM GameFamily f Natural JOIN BelongsTo b JOIN BoardgameProduct g on b.GameID = g.GameID
-> 1	HERE PartyFriendly = 10
-> (croup By Familyname
	UNION (
	ELECT Familyname, Avg(q.Difficulty) as AvgDifficulty, Avg(q.Duration) as AvgDuration, Avg(q.Trading) as AvgTrading, PartyFriendly
	ROM GameFamily f Natural JOIN BelongsTo b JOIN BoardgameProduct g on b.GameID = g.GameID
	HERE PartyFriendly = 1
	roup By Familyname
-> 1	
->	
·	
EXPLA	
EXPLA.	N
+	

```
Table scan on <union temporary> (cost=2.50..2.50 rows=0) (actual time=0.001..0.029 rows=295 loops=1) >> Union materialize with deduplication (cost=2.50..2.50 rows=0) (actual time=8.024..8.076 rows=295 loops=1)
                    | scan on \text{ scan on \text{ climber targy} \text{ cost=2.50..2.50 \text{ rows=0} \text{ climbes 0.004..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..1.005..
| -> Table scan on <union temporary> (cost=2.50..2.50 rows=0) (actual time=0.001..0.029 rows=295 loops=1)
     -> Union materialize with deduplication (cost=2.50..2.50 rows=0) (actual time=8.024..8.076 rows=295 loops=1)
         -> Table scan on <temporary> (actual time=0.002..0.017 rows=142 loops=1)
              -> Aggregate using temporary table (actual time=3.926..3.952 rows=142 loops=1)
                   -> Nested loop inner join (cost=425.12 rows=514) (actual time=0.093..3.007 rows=635 loops=1)
                        -> Nested loop inner join (cost=245.20 rows=514) (actual time=0.083..1.447 rows=635 loops=1)
                             -> Filter: (f.PartyFriendly = 10) (cost=155.25 rows=154) (actual time=0.070..0.734 rows=142 loops=1)
                                 -> Table scan on f (cost=155.25 rows=1535) (actual time=0.063..0.575 rows=1535 loops=1)
                             -> Index lookup on b using PRIMARY (FamilyID=f.FamilyID) (cost=0.25 rows=3) (actual time=0.003..0.004 rows=4 loops=142)
                        -> Single-row index lookup on g using PRIMARY (GameID=b.GameID) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=635)
         -> Table scan on <temporary> (actual time=0.001..0.019 rows=153 loops=1)
              -> Aggregate using temporary table (actual time=3.642..3.672 rows=153 loops=1)
                   -> Nested loop inner join (cost=425.12 rows=514) (actual time=0.071..2.665 rows=683 loops=1)
                        -> Nested loop inner join (cost=245.20 rows=514) (actual time=0.063..1.348 rows=683 loops=1)
                             -> Filter: (f.PartyFriendly = 1) (cost=155.25 rows=154) (actual time=0.050..0.714 rows=153 loops=1)
                                 -> Table scan on f (cost=155.25 rows=1535) (actual time=0.044..0.551 rows=1535 loops=1)
                             -> Index lookup on b using PRIMARY (FamilyID=f.FamilyID) (cost=0.25 rows=3) (actual time=0.002..0.003 rows=4 loops=153)
                        -> Single-row index lookup on g using PRIMARY (GameID=b.GameID) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=683)
```

1 row in set (0.02 sec)

DROP Duration index on BoardgameProduct, ADD Familyname index on GameFamily:

```
mysql> DROP INDEX idx_duration ON BoardgameProduct;
Query OK, 0 rows affected (2.54 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> CREATE INDEX idx_partyfriendly ON GameFamily(PartyFriendly);
Query OK, 0 rows affected (0.09 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

mysql> SHOW INDEX FROM +	 ue	Key_name	' Seq_in_in	dex	 Column_name		Collation	Cardinality	' Sub_p	oart	Packed	Null	Index_type	Comment	Index_comment	' Visibl
++ GameFamily NULL GameFamily NULL	0	PRIMARY idx_partyfriendly			FamilyID PartyFriendly	I A		1535 10	I N	AOPT	NULL	YES	BTREE			YES
++ 2 rows in set (0.01 sec			,		,	-+	-		+	+						+

```
Hysql> EXPLAIN MALYIE (SELECT Familyname, Avg(g.Difficulty) as AvgDifficulty, Avg(g.Duration) as AvgDuration, Avg(g.Trading) as AvgTrading, FartyFriendly

> HERE RartyFriendly = 10

> Group By Familyname

> DINTON(

> SELECT Familyname, Avg(g.Difficulty) as AvgDifficulty, Avg(g.Duration) as AvgDuration, Avg(g.Trading) as AvgTrading, PartyFriendly

> FROM GamePamily f Natural JOIN BelongsTo b JOIN BoardgameProduct g on b.GameID = g.GameID

> WHERE PartyFriendly = 1

> Group By Familyname

-) |

> ;

| EXPLAIN
```

```
in set, 4 warnings (0.01 sec)
| -> Table scan on <union temporary> (cost=2.50..2.50 rows=0) (actual time=0.002..0.045 rows=295 loops=1)
  -> Union materialize with deduplication (cost=2.50..2.50 rows=0) (actual time=7.890..7.960 rows=295 loops=1)
    -> Table scan on <temporary> (actual time=0.002..0.018 rows=142 loops=1)
       -> Aggregate using temporary table (actual time=3.875..3.903 rows=142 loops=1)
         -> Nested loop inner join (cost=425.12 rows=514) (actual time=0.134..2.851 rows=635 loops=1)
            -> Nested loop inner join (cost=245.20 rows=514) (actual time=0.121..1.459 rows=635 loops=1)
              -> Filter: (f.PartyFriendly = 10) (cost=155.25 rows=154) (actual time=0.104..0.796 rows=142 loops=1)
                -> Table scan on f (cost=155.25 rows=1535) (actual time=0.097..0.615 rows=1535 loops=1)
              -> Index lookup on b using PRIMARY (FamilyID=f.FamilyID) (cost=0.25 rows=3) (actual time=0.002..0.004 rows=4 loops=142)
            -> Single-row index lookup on g using PRIMARY (GameID=b.GameID) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=635)
    -> Table scan on <temporary> (actual time=0.001..0.017 rows=153 loops=1)
       -> Aggregate using temporary table (actual time=3.456..3.486 rows=153 loops=1)
         -> Nested loop inner join (cost=425.12 rows=514) (actual time=0.065..2.501 rows=683 loops=1)
            -> Nested loop inner join (cost=245.20 rows=514) (actual time=0.057..1.316 rows=683 loops=1)
              -> Filter: (f.PartyFriendly = 1) (cost=155.25 rows=154) (actual time=0.048..0.694 rows=153 loops=1)
                -> Table scan on f (cost=155.25 rows=1535) (actual time=0.041..0.532 rows=1535 loops=1)
              -> Index lookup on b using PRIMARY (FamilyID=f.FamilyID) (cost=0.25 rows=3) (actual time=0.002..0.003 rows=4 loops=153)
            -> Single-row index lookup on g using PRIMARY (GameID=b.GameID) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=683)
```

1 row in set (0.01 sec)