CS411 Project Track 1 Stage 3

Database Implementation:

- **Database Tables**

```
mysql> show tables;
+----------------+
| Tables_in_games |
+----------------+
| Game           |
| Game_info      |
| Personal_list  |
| Sales          |
+----------------+
4 rows in set (0.03 sec)
```

- **Number of rows** in tables Game, Game_info, Sales

```
mysql> select count(gameId) from Game
    -> ;
+---------------+
| count(gameId) |
+---------------+
|          5205 |
+---------------+
1 row in set (0.02 sec)

mysql> select count(gameId) from Game_info
    -> ;
+---------------+
| count(gameId) |
+---------------+
|          5205 |
+---------------+
1 row in set (0.02 sec)

mysql> select count(gameId) from Sales;
+---------------+
| count(gameId) |
+---------------+
|          5205 |
+---------------+
1 row in set (0.08 sec)
```

- **Data Definition Language Commands**

```
CREATE Table User (
userId VARCHAR(1000) Primary Key,
password VARCHAR(100),
listId INT,
FOREIGH KEY (listId) references Personal_list(listId)
ON DELETE CASCADE
);

CREATE Table Personal_list  (
listId INT Primary Key,
note VARCHAR(1000),
gameId INT,
FOREIGN KEY (gameId) references Game(gameId)
ON DELETE CASCADE
);

CREATE Table Sales (
gameId INT Primary Key,
Sale_NA FLOAT,
Sale_EU FLOAT,
Sale_JP FLOAT,
Sale_Others FLOAT,
Sale_Global FLOAT
);

CREATE Table Game (
gameId INT Primary Key,
name VARCHAR(100),
year INT,
genre VARCHAR(100)
);

CREATE Table Game_info (
gameId INT Primary Key,
review VARCHAR(1000),
summary VARCHAR(1000),
platform VARCHAR(30),
publisher VARCHAR(30)
);
```

Advanced Queries:

Advanced query 1

We want to obtain information about gameId, game name, average game review score of games that are published on PC platforms with game summary starting with ‘T’, and the number of sales in EU, or the year which the game was published was the same as the games which had the second series. Also, we needed the average review score to be above 4. ANd the final results were ordered by the output by gameId.

```
SELECT gameId, name, AVG(review) as average_review
FROM Game LEFT JOIN Game_info USING(gameId) LEFT JOIN Sales USING(gameId)
WHERE platform = 'PC' AND summary like 'T%' AND Sale_EU > 1
OR year IN (SELECT year FROM Game WHERE name LIKE '%2%')
GROUP BY gameId
HAVING average_review > 4
ORDER BY gameId
LIMIT 15
```

Screenshot of the top 15 rows for advanced query 1

gameId	name	average_review
1	The Legend of Zelda: Ocarina of Time	9.1
2	Tony Hawk's Pro Skater 2	7.4
3	Grand Theft Auto IV	7.7
4	Grand Theft Auto IV	7.9
5	Super Mario Galaxy	9.1
6	Super Mario Galaxy 2	9.1
7	Grand Theft Auto V	8.3
8	Grand Theft Auto V	8.3
9	Tony Hawk's Pro Skater 3	7.5
10	Perfect Dark	8.8
11	Grand Theft Auto V	8.4
12	Metroid Prime	8.6
13	Grand Theft Auto III	8.4
14	Halo: Combat Evolved	8.7
15	Half-Life 2	9.2

Advanced Query 2

We want info of the most popular games (games with sales in NA are over 2). Return the amount of sales, game name and review score for the most popular games sold on the PlayStation platform. Most popular games are defined as: in the NA market, its sales are over 2. Order output in inputOrder order by sales and game name.

```
SELECT Game.name, Game_info.review, popularGames.Sale_NA
FROM Game NATURAL JOIN Game_info JOIN (
                            SELECT gameId, Sale_NA
                            FROM Sales
                            WHERE Sale_NA > 2) AS popularGames
WHERE platform = “PlayStation 3”  AND Game.gameId = popularGames.gameId
ORDER BY popularGames.Sale_NA DESC, Game.name;
```

Screenshot of the top 15 rows for advanced query 2

name	review	Sale_NA
Grand Theft Auto V	8.3	7.01
Call of Duty: Modern Warfare 3	3.3	5.54
Call of Duty: Modern Warfare 2	6.6	4.99
Grand Theft Auto IV	7.7	4.76
Call of Duty: Ghosts	2.8	4.09
Uncharted 2: Among Thieves	8.8	3.27
Call of Duty 4: Modern Warfare	8.4	3.1
Gran Turismo 5	7.8	2.96
Battlefield 3	7.5	2.85
LittleBigPlanet	6.8	2.8
Red Dead Redemption	8.9	2.79
Uncharted 3: Drake's Deception	8.4	2.77
God of War III	8.8	2.74
Call of Duty: World at War	7.7	2.72
Batman: Arkham City	8.8	2.7

Indexing Analysis

Advanced query 1

Performance without indexing:

mysql> EXPLAIN ANALYZE (Advanced query 1)

```
 -> Limit: 15 row(s)  (cost=4802.50 rows=15) (actual time=3.257..3.343 rows=15 loops=1)
   -> Filter: (average_review > 4)  (cost=4802.50 rows=5330) (actual time=3.256..3.341 rows=15 loops=1)
       -> Group aggregate: avg(Game_info.review)  (cost=4802.50 rows=5330) (actual time=3.253..3.336 rows=15 loops=1)
           -> Filter: (((Game_info.platform = 'PC') and (Game_info.summary like 'T%') and (Sales.Sale_EU > 1)) or <in_optimizer>(Game.`year`,Game.`year` in (select #2)))  (cost=4269.50 rows=5330) (actual time=3.23
..3.317 rows=16 loops=1)
               -> Nested loop left join  (cost=4269.50 rows=5330) (actual time=3.228..3.309 rows=16 loops=1)
                   -> Filter: (((Game_info.platform = 'PC') and (Game_info.summary like 'T%')) or <in_optimizer>(Game.`year`,Game.`year` in (select #2)))  (cost=2404.00 rows=5330) (actual time=3.211..3.270 rows=16
loops=1)
                       -> Nested loop left join  (cost=2404.00 rows=5330) (actual time=0.090..0.132 rows=16 loops=1)
                           -> Index scan on Game using PRIMARY  (cost=538.50 rows=5330) (actual time=0.069..0.075 rows=16 loops=1)
                           -> Single-row index lookup on Game_info using PRIMARY (gameId=Game.gameId)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=16)
                       -> Select #2 (subquery in condition; run only once)
                           -> Filter: ((Game.`year` = `<materialized_subquery>`.`year`))  (actual time=0.001..0.001 rows=1 loops=14)
                               -> Limit: 1 row(s)  (actual time=0.001..0.001 rows=1 loops=14)
                                   -> Index lookup on <materialized_subquery> using <auto_distinct_key> (year=Game.`year`)  (actual time=0.000..0.000 rows=1 loops=14)
                                       -> Materialize with deduplication  (cost=597.72..597.72 rows=592) (actual time=3.117..3.117 rows=22 loops=1)
                                           -> Filter: (Game.`name` like '%2%')  (cost=538.50 rows=592) (actual time=0.034..2.947 rows=892 loops=1)
                                               -> Table scan on Game  (cost=538.50 rows=5330) (actual time=0.029..1.338 rows=5205 loops=1)
                   -> Single-row index lookup on Sales using PRIMARY (gameId=Game.gameId)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=16)
               -> Select #2 (subquery in condition; run only once)
                   -> Filter: ((Game.`year` = `<materialized_subquery>`.`year`))  (actual time=0.001..0.001 rows=1 loops=14)
                       -> Limit: 1 row(s)  (actual time=0.001..0.001 rows=1 loops=14)
                           -> Index lookup on <materialized_subquery> using <auto_distinct_key> (year=Game.`year`)  (actual time=0.000..0.000 rows=1 loops=14)
                               -> Materialize with deduplication  (cost=597.72..597.72 rows=592) (actual time=3.117..3.117 rows=22 loops=1)
                                   -> Filter: (Game.`name` like '%2%')  (cost=538.50 rows=592) (actual time=0.034..2.947 rows=892 loops=1)
                                       -> Table scan on Game  (cost=538.50 rows=5330) (actual time=0.029..1.338 rows=5205 loops=1)
 |
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-----------+
 row in set (0.16 sec)
```

Performance with indexing design 1:

mysql> CREATE INDEX **idx_game_name** ON **Game (name);**

mysql> EXPLAIN ANALYZE (Advanced query 1)

```
 -> Limit: 15 row(s)  (cost=4802.50 rows=15) (actual time=4.249..4.336 rows=15 loops=1)
   -> Filter: (average_review > 4)  (cost=4802.50 rows=5330) (actual time=4.248..4.334 rows=15 loops=1)
       -> Group aggregate: avg(Game_info.review)  (cost=4802.50 rows=5330) (actual time=4.244..4.328 rows=15 loops=1)
           -> Filter: (((Game_info.platform = 'PC') and (Game_info.summary like 'T%') and (Sales.Sale_EU > 1)) or <in_optimizer>(Game.`year`,Game.`year` in (select #2)))  (cost=4269.50 rows=5330) (actual time=4.21
..4.307 rows=16 loops=1)
               -> Nested loop left join  (cost=4269.50 rows=5330) (actual time=4.207..4.296 rows=16 loops=1)
                   -> Filter: (((Game_info.platform = 'PC') and (Game_info.summary like 'T%')) or <in_optimizer>(Game.`year`,Game.`year` in (select #2)))  (cost=2404.00 rows=5330) (actual time=4.179..4.243 rows=16
loops=1)
                       -> Nested loop left join  (cost=2404.00 rows=5330) (actual time=0.091..0.137 rows=16 loops=1)
                           -> Index scan on Game using PRIMARY  (cost=538.50 rows=5330) (actual time=0.072..0.077 rows=16 loops=1)
                           -> Single-row index lookup on Game_info using PRIMARY (gameId=Game.gameId)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=16)
                       -> Select #2 (subquery in condition; run only once)
                           -> Filter: ((Game.`year` = `<materialized_subquery>`.`year`))  (actual time=0.001..0.001 rows=1 loops=14)
                               -> Limit: 1 row(s)  (actual time=0.001..0.001 rows=1 loops=14)
                                   -> Index lookup on <materialized_subquery> using <auto_distinct_key> (year=Game.`year`)  (actual time=0.001..0.001 rows=1 loops=14)
                                       -> Materialize with deduplication  (cost=597.72..597.72 rows=592) (actual time=4.083..4.083 rows=22 loops=1)
                                           -> Filter: (Game.`name` like '%2%')  (cost=538.50 rows=592) (actual time=0.046..3.837 rows=892 loops=1)
                                               -> Table scan on Game  (cost=538.50 rows=5330) (actual time=0.040..2.032 rows=5205 loops=1)
                   -> Single-row index lookup on Sales using PRIMARY (gameId=Game.gameId)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=16)
               -> Select #2 (subquery in condition; run only once)
                   -> Filter: ((Game.`year` = `<materialized_subquery>`.`year`))  (actual time=0.001..0.001 rows=1 loops=14)
                       -> Limit: 1 row(s)  (actual time=0.001..0.001 rows=1 loops=14)
                           -> Index lookup on <materialized_subquery> using <auto_distinct_key> (year=Game.`year`)  (actual time=0.001..0.001 rows=1 loops=14)
                               -> Materialize with deduplication  (cost=597.72..597.72 rows=592) (actual time=4.083..4.083 rows=22 loops=1)
                                   -> Filter: (Game.`name` like '%2%')  (cost=538.50 rows=592) (actual time=0.046..3.837 rows=892 loops=1)
                                       -> Table scan on Game  (cost=538.50 rows=5330) (actual time=0.040..2.032 rows=5205 loops=1)
 |
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-----------+
 row in set (0.07 sec)
```

Performance with indexing design 2:

```
mysql> CREATE INDEX idx_game_year ON Game (year);

mysql> EXPLAIN ANALYZE (Advanced query 1)
-----------+
 -> Limit: 15 row(s)  (cost=4802.50 rows=15) (actual time=3.569..3.674 rows=15 loops=1)
   -> Filter: (average_review > 4)  (cost=4802.50 rows=5330) (actual time=3.568..3.671 rows=15 loops=1)
       -> Group aggregate: avg(Game_info.review)  (cost=4802.50 rows=5330) (actual time=3.539..3.640 rows=15 loops=1)
           -> Filter: (((Game_info.platform = 'PC') and (Game_info.summary like 'T%') and (Sales.Sale_EU > 1)) or <in_optimizer>(Game.`year`,Game.`year` in (select #2)))  (cost=4269.50 rows=5330) (actual time=3.4
..3.566 rows=16 loops=1)
               -> Nested loop left join  (cost=4269.50 rows=5330) (actual time=3.453..3.555 rows=16 loops=1)
                   -> Filter: (((Game_info.platform = 'PC') and (Game_info.summary like 'T%')) or <in_optimizer>(Game.`year`,Game.`year` in (select #2)))  (cost=2404.00 rows=5330) (actual time=3.363..3.438 rows=1
loops=1)
                       -> Nested loop left join  (cost=2404.00 rows=5330) (actual time=0.074..0.125 rows=16 loops=1)
                           -> Index scan on Game using PRIMARY  (cost=538.50 rows=5330) (actual time=0.058..0.064 rows=16 loops=1)
                           -> Single-row index lookup on Game_info using PRIMARY (gameId=Game.gameId)  (cost=0.25 rows=1) (actual time=0.004..0.004 rows=1 loops=16)
                       -> Select #2 (subquery in condition; run only once)
                           -> Filter: ((Game.`year` = `<materialized_subquery>`.`year`))  (actual time=0.001..0.001 rows=1 loops=14)
                               -> Limit: 1 row(s)  (actual time=0.001..0.001 rows=1 loops=14)
                                   -> Index lookup on <materialized_subquery> using <auto_distinct_key> (year=Game.`year`)  (actual time=0.001..0.001 rows=1 loops=14)
                                       -> Materialize with deduplication  (cost=597.72..597.72 rows=592) (actual time=3.286..3.286 rows=22 loops=1)
                                           -> Filter: (Game.`name` like '%2%')  (cost=538.50 rows=592) (actual time=0.037..3.092 rows=892 loops=1)
                                               -> Table scan on Game  (cost=538.50 rows=5330) (actual time=0.031..1.423 rows=5205 loops=1)
                   -> Single-row index lookup on Sales using PRIMARY (gameId=Game.gameId)  (cost=0.25 rows=1) (actual time=0.002..0.003 rows=1 loops=16)
               -> Select #2 (subquery in condition; run only once)
                   -> Filter: ((Game.`year` = `<materialized_subquery>`.`year`))  (actual time=0.001..0.001 rows=1 loops=14)
                       -> Limit: 1 row(s)  (actual time=0.001..0.001 rows=1 loops=14)
                           -> Index lookup on <materialized_subquery> using <auto_distinct_key> (year=Game.`year`)  (actual time=0.001..0.001 rows=1 loops=14)
                               -> Materialize with deduplication  (cost=597.72..597.72 rows=592) (actual time=3.286..3.286 rows=22 loops=1)
                                   -> Filter: (Game.`name` like '%2%')  (cost=538.50 rows=592) (actual time=0.037..3.092 rows=892 loops=1)
                                       -> Table scan on Game  (cost=538.50 rows=5330) (actual time=0.031..1.423 rows=5205 loops=1)
 |
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
-----------+
 row in set (0.05 sec)
```

Performance with indexing design 3:

```
mysql> CREATE INDEX idx_game_platform ON Game_info (platform);

mysql> EXPLAIN ANALYZE (Advanced query 1)
 -> Limit: 15 row(s)  (cost=4802.50 rows=15) (actual time=3.501..3.595 rows=15 loops=1)
   -> Filter: (average_review > 4)  (cost=4802.50 rows=5330) (actual time=3.500..3.593 rows=15 loops=1)
       -> Group aggregate: avg(Game_info.review)  (cost=4802.50 rows=5330) (actual time=3.495..3.586 rows=15 loops=1)
           -> Filter: (((Game_info.platform = 'PC') and (Game_info.summary like 'T%') and (Sales.Sale_EU > 1)) or <in_optimizer>(Game.`year`,Game.`year` in (select #2)))  (cost=4269.50 rows=5330) (actual time=3.4
..3.553 rows=16 loops=1)
               -> Nested loop left join  (cost=4269.50 rows=5330) (actual time=3.447..3.543 rows=16 loops=1)
                   -> Filter: (((Game_info.platform = 'PC') and (Game_info.summary like 'T%')) or <in_optimizer>(Game.`year`,Game.`year` in (select #2)))  (cost=2404.00 rows=5330) (actual time=3.426..3.496 rows=1
loops=1)
                       -> Nested loop left join  (cost=2404.00 rows=5330) (actual time=0.081..0.130 rows=16 loops=1)
                           -> Index scan on Game using PRIMARY  (cost=538.50 rows=5330) (actual time=0.063..0.069 rows=16 loops=1)
                           -> Single-row index lookup on Game_info using PRIMARY (gameId=Game.gameId)  (cost=0.25 rows=1) (actual time=0.003..0.004 rows=1 loops=16)
                       -> Select #2 (subquery in condition; run only once)
                           -> Filter: ((Game.`year` = `<materialized_subquery>`.`year`))  (actual time=0.001..0.001 rows=1 loops=14)
                               -> Limit: 1 row(s)  (actual time=0.001..0.001 rows=1 loops=14)
                                   -> Index lookup on <materialized_subquery> using <auto_distinct_key> (year=Game.`year`)  (actual time=0.001..0.001 rows=1 loops=14)
                                       -> Materialize with deduplication  (cost=597.72..597.72 rows=592) (actual time=3.340..3.340 rows=22 loops=1)
                                           -> Filter: (Game.`name` like '%2%')  (cost=538.50 rows=592) (actual time=0.038..3.143 rows=892 loops=1)
                                               -> Table scan on Game  (cost=538.50 rows=5330) (actual time=0.032..1.522 rows=5205 loops=1)
                   -> Single-row index lookup on Sales using PRIMARY (gameId=Game.gameId)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=16)
               -> Select #2 (subquery in condition; run only once)
                   -> Filter: ((Game.`year` = `<materialized_subquery>`.`year`))  (actual time=0.001..0.001 rows=1 loops=14)
                       -> Limit: 1 row(s)  (actual time=0.001..0.001 rows=1 loops=14)
                           -> Index lookup on <materialized_subquery> using <auto_distinct_key> (year=Game.`year`)  (actual time=0.001..0.001 rows=1 loops=14)
                               -> Materialize with deduplication  (cost=597.72..597.72 rows=592) (actual time=3.340..3.340 rows=22 loops=1)
                                   -> Filter: (Game.`name` like '%2%')  (cost=538.50 rows=592) (actual time=0.038..3.143 rows=892 loops=1)
                                       -> Table scan on Game  (cost=538.50 rows=5330) (actual time=0.032..1.522 rows=5205 loops=1)
 |
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
-----------+
 row in set (0.05 sec)
```

Indexing Analysis Report for Advanced Query 1:

For this query, we looked up information on PC games with a summary starting with ‘T’, and sales data in the EU market. Or PC games which have a second series published in the same year with the first edition, additionally we need the average review score of the games to be above 4. For the first indexing strategy, we used idx_game_name on the Game table. The running time result of EXPLAIN ANALYSIS shows that the processing time had a significant improvement compared to the default setting. Adding index on game name improved the processing time for ordering the query results. For the second indexing strategy, we used idx_game_year on the Game table. The performance of this indexing also improved the processing time as we also added constraints of game year in the original query and added a smaller domain on the game year. For the last indexing strategy, we used idx_game_platform on the Game_info table as this improved the execution time for querying game platforms for the WHERE clause in the original query.

Advanced Query 2

Performance without indexing:

```
mysql> EXPLAIN ANALYZE (Advanced query 2)
```

```
| -> Sort: popularGames.Sale_NA DESC, Game.`name`  (actual time=4.857..4.860 rows=28 loops=1)
    -> Stream results  (cost=788.27 rows=163) (actual time=0.121..4.817 rows=28 loops=1)
        -> Nested loop inner join  (cost=788.27 rows=163) (actual time=0.115..4.795 rows=28 loops=1)
            -> Nested loop inner join  (cost=731.32 rows=163) (actual time=0.098..4.706 rows=28 loops=1)
                -> Filter: (Game_info.platform = 'PlayStation 3')  (cost=560.45 rows=488) (actual time=0.077..3.681 rows=636 loops=1)
                    -> Table scan on Game_info  (cost=560.45 rows=4882) (actual time=0.067..2.677 rows=5205 loops=1)
                -> Filter: (Sales.Sale_NA > 2)  (cost=0.25 rows=0) (actual time=0.001..0.001 rows=0 loops=636)
                    -> Single-row index lookup on Sales using PRIMARY (gameId=Game_info.gameId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=636)
            -> Single-row index lookup on Game using PRIMARY (gameId=Game_info.gameId)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=28)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
1 row in set (0.20 sec)
```

Performance with indexing design 1:

```
mysql> CREATE INDEX idx_game_name ON Game (name);

mysql> EXPLAIN ANALYZE (Advanced query 2)
```

```
-------------------------------------------------------------------------------------------------------------------------------------------------------------
| -> Sort: popularGames.Sale_NA DESC, Game.`name`  (actual time=5.112..5.115 rows=28 loops=1)
    -> Stream results  (cost=788.27 rows=163) (actual time=0.098..5.076 rows=28 loops=1)
        -> Nested loop inner join  (cost=788.27 rows=163) (actual time=0.092..5.054 rows=28 loops=1)
            -> Nested loop inner join  (cost=731.32 rows=163) (actual time=0.077..4.807 rows=28 loops=1)
                -> Filter: (Game_info.platform = 'PlayStation 3')  (cost=560.45 rows=488) (actual time=0.062..3.718 rows=636 loops=1)
                    -> Table scan on Game_info  (cost=560.45 rows=4882) (actual time=0.054..2.735 rows=5205 loops=1)
                -> Filter: (Sales.Sale_NA > 2)  (cost=0.25 rows=0) (actual time=0.002..0.002 rows=0 loops=636)
                    -> Single-row index lookup on Sales using PRIMARY (gameId=Game_info.gameId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=636)
            -> Single-row index lookup on Game using PRIMARY (gameId=Game_info.gameId)  (cost=0.25 rows=1) (actual time=0.009..0.009 rows=1 loops=28)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
1 row in set (0.04 sec)
```

Performance with indexing design 2:

```
mysql> CREATE INDEX idx_sales_NA ON Sales (sale_NA);

mysql> EXPLAIN ANALYZE (Advanced query 2)
```

```
| -> Sort: popularGames.Sale_NA DESC, Game.`name`  (actual time=0.786..0.789 rows=28 loops=1)
    -> Stream results  (cost=118.05 rows=20) (actual time=0.101..0.758 rows=28 loops=1)
        -> Nested loop inner join  (cost=118.05 rows=20) (actual time=0.097..0.736 rows=28 loops=1)
            -> Nested loop inner join  (cost=111.05 rows=20) (actual time=0.086..0.669 rows=28 loops=1)
                -> Filter: (Sales.Sale_NA > 2)  (cost=41.05 rows=200) (actual time=0.055..0.112 rows=200 loops=1)
                    -> Index range scan on Sales using idx_sales_NA  (cost=41.05 rows=200) (actual time=0.037..0.075 rows=200 loops=1)
                -> Filter: (Game_info.platform = 'PlayStation 3')  (cost=0.25 rows=0) (actual time=0.003..0.003 rows=0 loops=200)
                    -> Single-row index lookup on Game_info using PRIMARY (gameId=Sales.gameId)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=200)
            -> Single-row index lookup on Game using PRIMARY (gameId=Sales.gameId)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=28)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--+
1 row in set (0.07 sec)
```

Performance with indexing design 3:

```
mysql> CREATE INDEX idx_game_platform ON Game_info (platform);

mysql> EXPLAIN ANALYZE (Advanced query 2)
```

```
------------------------------------------------------------------------------------------------------------------+
| -> Sort: popularGames.Sale_NA DESC, Game.`name`  (actual time=3.343..3.346 rows=28 loops=1)
    -> Stream results  (cost=482.44 rows=212) (actual time=0.107..3.312 rows=28 loops=1)
        -> Nested loop inner join  (cost=482.44 rows=212) (actual time=0.103..3.296 rows=28 loops=1)
            -> Nested loop inner join  (cost=408.25 rows=212) (actual time=0.090..3.218 rows=28 loops=1)
                -> Index lookup on Game_info using idx_game_platform (platform='PlayStation 3')  (cost=185.65 rows=636) (actual time=0.079..2.219 rows=636 loops=1)
                -> Filter: (Sales.Sale_NA > 2)  (cost=0.25 rows=0) (actual time=0.001..0.001 rows=0 loops=636)
                    -> Single-row index lookup on Sales using PRIMARY (gameId=Game_info.gameId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=636)
            -> Single-row index lookup on Game using PRIMARY (gameId=Game_info.gameId)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=28)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------+
1 row in set (0.06 sec)
```

Indexing Analysis Report for Advanced Query 2:

For the second advanced query, we tried to look for the most popular games (games with sales in NA are over 2), and return the amount of sales, game name and review score for the most popular games sold on the PlayStation platform. Thus, we decided to add an index on Game.name, Sales.sale_NA or Game_info.platform. The results showed these indexes all improve the performance of this query by a significant amount of time. The reason for us to choose Game.name was because we needed to return game.name and order the results based on it. And the index on Game.name was supposed to improve the performance for ordering. Also, Sales.sale_NA was selected as it was used for joining a table based on a limited condition. Therefore, the performance advanced significantly. Lastly, the index on Game_info.platform was used in the where clause, which resulted in the improvement of the query performance as platform was compared by many times in this query.