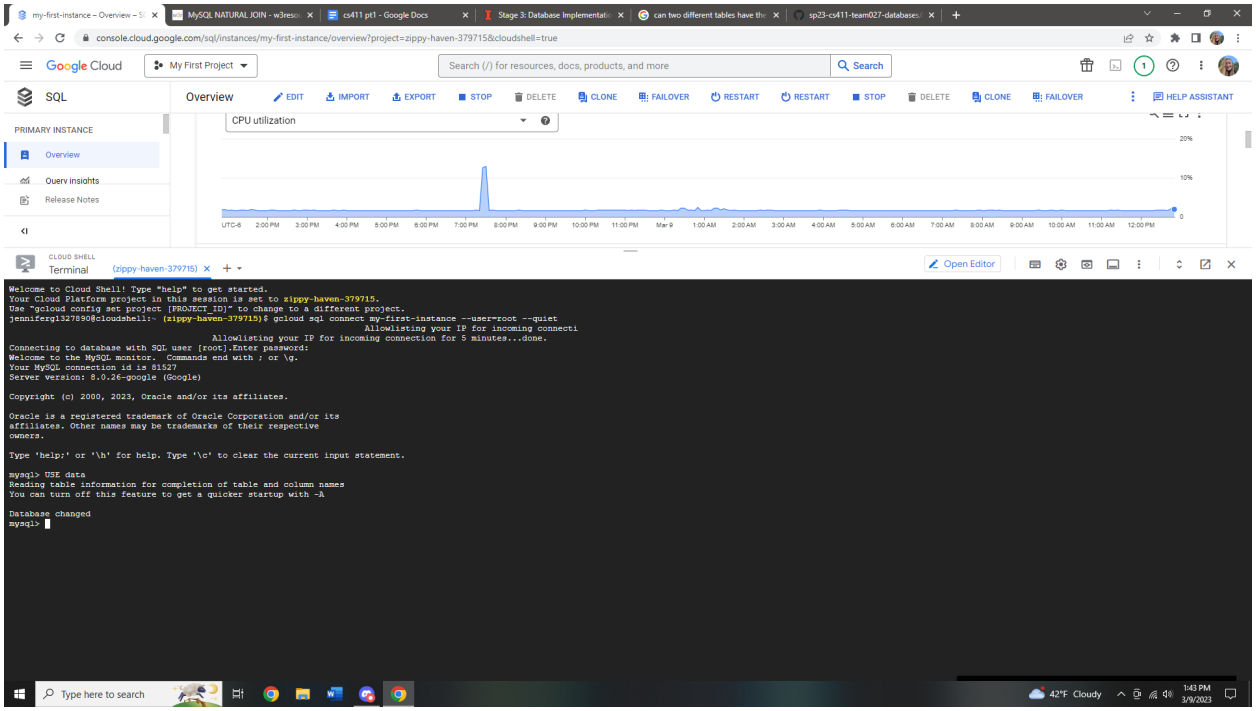
Divya Machineni
Michael Ma
Jennifer Gaw
Kaan Yigit

# CONNECTION:



## DDL COMMANDS:

```
CREATE TABLE GameData (
      GameID int,
      GameName varchar(255),
      Description varchar(1000),
      Rating int,
      Price REAL,
      Reviews varchar(255),
      PCreq varchar(1000),
      Developer int,
      RequiredAge int,
      Language varchar(255),
      PRIMARY KEY (GameID)
);

CREATE TABLE GenreData (
      GameID int,
      Singleplayer varchar(10),
      Multiplayer varchar(10),
      Coop varchar(10),
      MMO varchar(10),
      InAppPurchase varchar(10),
      VRSupport varchar(10),
      NonGame varchar(10),
      Indie varchar(10),
      Action varchar(10),
      Adventure varchar(10),
      Casual varchar(10),
      Strategy varchar(10),
      RPG varchar(10),
      Simulation varchar(10),
      EarlyAccess varchar(10),
      FreeToPlay varchar(10),
      Sports varchar(10),
      Racing varchar(10),
      MassivelyMultiplayer varchar(10),
      PRIMARY KEY (GameID)
      FOREIGN KEY (GameID) REFERENCES Game(GameID)
```

```
);



CREATE TABLE Region (
     RegionName varchar(255),
     PRIMARY KEY (RegionName)
);

CREATE TABLE PriceRange (
     Range int,
     PriceRangeMin int,
     PriceRangeMax int,
     PRIMARY KEY (Range)
);
```

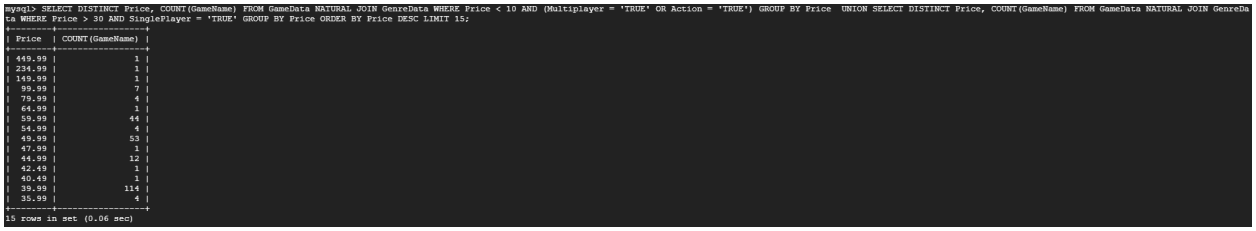
## TABLE COUNTS:

```
mysql> SELECT COUNT(*) FROM GameData
    -> ;
+----------+
| COUNT(*) |
+----------+
|    13358 |
+----------+
1 row in set (0.01 sec)

mysql> SELECT COUNT(*) FROM GenreData;
+----------+
| COUNT(*) |
+----------+
|    13358 |
+----------+
1 row in set (0.01 sec)

mysql> SELECT COUNT(*) FROM Region;
+----------+
| COUNT(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM PriceRange;
+----------+
| COUNT(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.00 sec)

mysql>
```

```
SELECT COUNT(*) FROM GameData;
SELECT COUNT(*) FROM GenreData;
SELECT COUNT(*) FROM Region;
SELECT COUNT(*) FROM PriceRange;
```

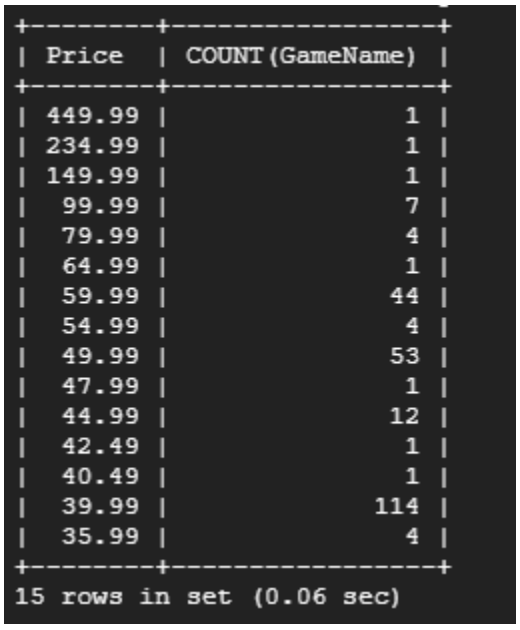
## ADVANCED QUERY 1:

```
SELECT DISTINCT Price, COUNT(GameName)
FROM GameData NATURAL JOIN GenreData
WHERE Price < 10 AND (Multiplayer = ‘TRUE’ OR Action = ‘TRUE’)
GROUP BY Price
UNION
SELECT DISTINCT Price, COUNT(GameName)
FROM GameData NATURAL JOIN GenreData
WHERE Price > 30 AND SinglePlayer = ‘TRUE’
GROUP BY Price
ORDER BY Price DESC
LIMIT 15;
```



| Price | COUNT(GameName) |
|---|---|
| 449.99 | 1 |
| 234.99 | 1 |
| 149.99 | 1 |
| 99.99 | 7 |
| 79.99 | 4 |
| 64.99 | 1 |
| 59.99 | 44 |
| 54.99 | 4 |
| 49.99 | 53 |
| 47.99 | 1 |
| 44.99 | 12 |
| 42.49 | 1 |
| 40.49 | 1 |
| 39.99 | 114 |
| 35.99 | 4 |

15 rows in set (0.06 sec)

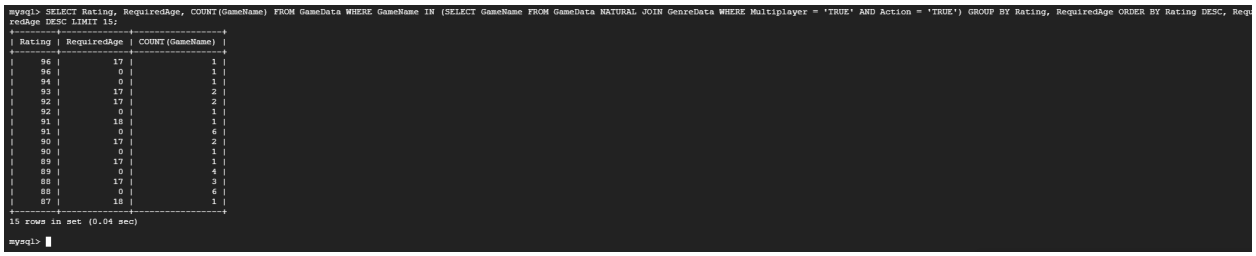


| Price | COUNT(GameName) |
|---|---|
| 449.99 | 1 |
| 234.99 | 1 |
| 149.99 | 1 |
| 99.99 | 7 |
| 79.99 | 4 |
| 64.99 | 1 |
| 59.99 | 44 |
| 54.99 | 4 |
| 49.99 | 53 |
| 47.99 | 1 |
| 44.99 | 12 |
| 42.49 | 1 |
| 40.49 | 1 |
| 39.99 | 114 |
| 35.99 | 4 |

15 rows in set (0.06 sec)

## ADVANCED QUERY 2:

```
SELECT Rating, RequiredAge, COUNT(GameName)
FROM  GameData
WHERE GameName IN (SELECT GameName
                   FROM GameData NATURAL JOIN GenreData
                   WHERE Multiplayer = 'TRUE' AND Action = 'TRUE')
GROUP BY Rating, RequiredAge
ORDER BY Rating DESC, RequiredAge DESC
LIMIT 15;
```




| Rating | RequiredAge | COUNT(GameName) |
|---|---|---|
| 96 | 17 | 1 |
| 96 | 0 | 1 |
| 94 | 0 | 1 |
| 93 | 17 | 2 |
| 92 | 17 | 2 |
| 92 | 0 | 1 |
| 91 | 18 | 1 |
| 91 | 0 | 6 |
| 90 | 17 | 2 |
| 90 | 0 | 1 |
| 89 | 17 | 1 |
| 89 | 0 | 4 |
| 88 | 17 | 3 |
| 88 | 0 | 6 |
| 87 | 18 | 1 |

15 rows in set (0.04 sec)

# INDEXING:

# Advanced Query 1:

### Original:

```
mysql> EXPLAIN ANALYZE SELECT DISTINCT Price, COUNT(GameName) FROM GameData NATURAL JOIN GenreData WHERE Price < 10 AND (Multiplayer = 'TRUE' OR Action = 'TRUE') GROUP BY Price UNION SELECT DI
STINCT Price, COUNT(GameName) FROM GameData NATURAL JOIN GenreData WHERE Price > 30 AND SinglePlayer = 'TRUE' GROUP BY Price ORDER BY Price DESC LIMIT 15;
```

```
| -> Limit: 15 row(s)  (cost=2.50 rows=0) (actual time=0.030..0.031 rows=15 loops=1)
    -> Sort: Price DESC, limit input to 15 row(s) per chunk  (cost=2.50 rows=0) (actual time=0.029..0.030 rows=15 loops=1)
        -> Table scan on <union temporary>  (cost=2.50 rows=0) (actual time=0.000..0.005 rows=99 loops=1)
            -> Union materialize with deduplication  (cost=2.50..2.50 rows=0) (actual time=56.520..56.523 rows=99 loops=1)
                -> Table scan on <temporary>  (actual time=0.001..0.005 rows=80 loops=1)
                    -> Aggregate using temporary table  (actual time=31.509..31.517 rows=80 loops=1)
                        -> Inner hash join (GameData.GameID = GenreData.GameID)  (cost=96587.25 rows=31103) (actual time=13.217..29.979 rows=5241 loops=1)
                            -> Filter: (GameData.Price < 10)  (cost=0.76 rows=368) (actual time=0.025..11.661 rows=10305 loops=1)
                                -> Table scan on GameData  (cost=0.76 rows=11030) (actual time=0.023..10.460 rows=13358 loops=1)
                            -> Hash
                                -> Filter: ((GenreData.Multiplayer = 'TRUE') or (GenreData.`Action` = 'TRUE'))  (cost=1376.25 rows=2538) (actual time=0.038..11.104 rows=7008 loops=1)
                                    -> Table scan on GenreData  (cost=1376.25 rows=13360) (actual time=0.027..8.258 rows=13358 loops=1)
                -> Table scan on <temporary>  (actual time=0.001..0.002 rows=19 loops=1)
                    -> Aggregate using temporary table  (actual time=24.937..24.939 rows=19 loops=1)
                        -> Inner hash join (GameData.GameID = GenreData.GameID)  (cost=51756.30 rows=16370) (actual time=12.636..24.813 rows=288 loops=1)
                            -> Filter: (GameData.Price > 30)  (cost=0.97 rows=368) (actual time=0.150..10.729 rows=396 loops=1)
                                -> Table scan on GameData  (cost=0.97 rows=11030) (actual time=0.020..9.928 rows=13358 loops=1)
                            -> Hash
                                -> Filter: (GenreData.Singleplayer = 'TRUE')  (cost=1376.25 rows=1336) (actual time=0.029..10.362 rows=11687 loops=1)
                                    -> Table scan on GenreData  (cost=1376.25 rows=13360) (actual time=0.018..7.991 rows=13358 loops=1)
 |
```

### Index 1:

```
mysql> CREATE INDEX Singleplayer_idx on GenreData(Singleplayer);
Query OK, 0 rows affected (0.15 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Results:

```
| -> Limit: 15 row(s)  (cost=2.50 rows=0) (actual time=0.029..0.031 rows=15 loops=1)
    -> Sort: Price DESC, limit input to 15 row(s) per chunk  (cost=2.50 rows=0) (actual time=0.028..0.029 rows=15 loops=1)
        -> Table scan on <union temporary>  (cost=2.50 rows=0) (actual time=0.000..0.005 rows=99 loops=1)
            -> Union materialize with deduplication  (cost=2.50..2.50 rows=0) (actual time=129.487..129.490 rows=99 loops=1)
                -> Table scan on <temporary>  (actual time=0.001..0.005 rows=80 loops=1)
                    -> Aggregate using temporary table  (actual time=32.317..32.326 rows=80 loops=1)
                        -> Inner hash join (GameData.GameID = GenreData.GameID)  (cost=96587.25 rows=31103) (actual time=13.408..30.835 rows=5241 loops=1)
                            -> Filter: (GameData.Price < 10)  (cost=0.76 rows=368) (actual time=0.028..12.238 rows=10305 loops=1)
                                -> Table scan on GameData  (cost=0.76 rows=11030) (actual time=0.026..11.035 rows=13358 loops=1)
                            -> Hash
                                -> Filter: ((GenreData.Multiplayer = 'TRUE') or (GenreData.`Action` = 'TRUE'))  (cost=1376.25 rows=2538) (actual time=0.033..11.423 rows=7008 loops=1)
                                    -> Table scan on GenreData  (cost=1376.25 rows=13360) (actual time=0.025..8.443 rows=13358 loops=1)
                -> Table scan on <temporary>  (actual time=0.001..0.002 rows=19 loops=1)
                    -> Aggregate using temporary table  (actual time=97.094..97.096 rows=19 loops=1)
                        -> Inner hash join (GameData.GameID = GenreData.GameID)  (cost=248624.51 rows=81851) (actual time=84.648..96.955 rows=288 loops=1)
                            -> Filter: (GameData.Price > 30)  (cost=0.34 rows=368) (actual time=0.208..10.925 rows=396 loops=1)
                                -> Table scan on GameData  (cost=0.34 rows=11030) (actual time=0.024..10.052 rows=13358 loops=1)
                            -> Hash
                                -> Index lookup on GenreData using Singleplayer_idx (Singleplayer='TRUE')  (cost=788.75 rows=6680) (actual time=0.064..81.502 rows=11687 loops=1)
 |
```

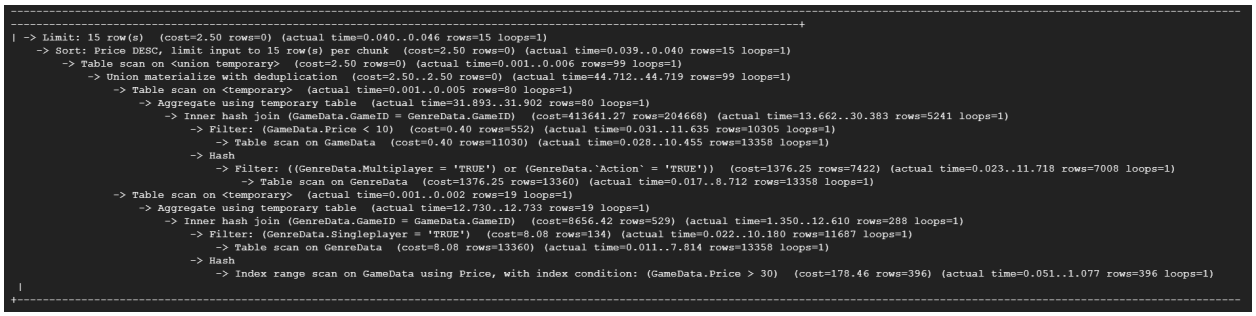
Explanation:

When we only index GenreData.Singleplayer, we can see that the cost decreases from 1376.25 to 788.75. This is because the data is stored in Main memory so it can be accessed a lot faster than without indexing Singleplayer.

### Index 2:

```
mysql> CREATE INDEX Price ON GameData(price);
```

Results:




Explanation:

When we only index Price, we can see that it effects both parts of the union query.
For the query before the Union:

We can see that the cost relating to GameData.Price decreases from 0.76 to 0.4. This could be because the indexing of the Price would help access the data faster since it is stored in the main memory.

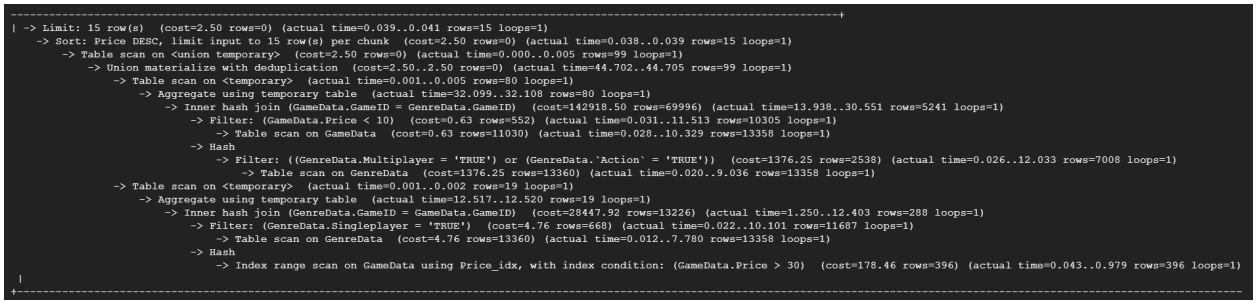
For the query after the Union:

We can see that cost relating to GameData.Price increased from 0.97 to 178.46. However, the cost relating to GenreData.SinglePlayer decreased from 1376.25 to 8.08. These two variables are linked via this SQL statement: `WHERE Price > 30 AND SinglePlayer = 'TRUE'`. Therefore, the changes in the costs could be due to the Price condition being executed first after indexing it, so this condition would filter out more rows than without the indexing. Thus, for the SinglePlayer condition, there are less rows for it to filter out and that's why the cost lowers for SinglePlayer instead.

**Index 3:**

```
mysql> CREATE INDEX Singleplayer_idx on GenreData(Singleplayer);
ERROR 1061 (42000): Duplicate key name 'Singleplayer_idx'
mysql> CREATE INDEX Price_idx on GameData(Price);
Query OK, 0 rows affected (0.12 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Results:



```
-> Limit: 15 row(s)  (cost=2.50 rows=0) (actual time=0.039..0.041 rows=15 loops=1)
    -> Sort: Price DESC, limit input to 15 row(s) per chunk  (cost=2.50 rows=0) (actual time=0.038..0.039 rows=15 loops=1)
        -> Table scan on <union temporary>  (cost=2.50 rows=0) (actual time=0.000..0.005 rows=99 loops=1)
            -> Union materialize with deduplication  (cost=2.50..2.50 rows=0) (actual time=44.702..44.705 rows=99 loops=1)
                -> Table scan on <temporary>  (actual time=0.001..0.005 rows=80 loops=1)
                    -> Aggregate using temporary table  (actual time=32.099..32.108 rows=80 loops=1)
                        -> Inner hash join (GameData.GameID = GenreData.GameID)  (cost=142918.50 rows=69996) (actual time=13.938..30.551 rows=5241 loops=1)
                            -> Filter: (GameData.Price < 10)  (cost=0.63 rows=552) (actual time=0.031..11.513 rows=10305 loops=1)
                                -> Table scan on GameData  (cost=0.63 rows=11030) (actual time=0.028..10.329 rows=13358 loops=1)
                            -> Hash
                                -> Filter: ((GenreData.Multiplayer = 'TRUE') or (GenreData.`Action` = 'TRUE'))  (cost=1376.25 rows=2538) (actual time=0.026..12.033 rows=7008 loops=1)
                                    -> Table scan on GenreData  (cost=1376.25 rows=13360) (actual time=0.020..9.036 rows=13358 loops=1)
                -> Table scan on <temporary>  (actual time=0.001..0.002 rows=19 loops=1)
                    -> Aggregate using temporary table  (actual time=12.517..12.520 rows=19 loops=1)
                        -> Inner hash join (GenreData.GameID = GameData.GameID)  (cost=28447.92 rows=13226) (actual time=1.250..12.403 rows=288 loops=1)
                            -> Filter: (GenreData.Singleplayer = 'TRUE')  (cost=4.76 rows=668) (actual time=0.022..10.101 rows=11687 loops=1)
                                -> Table scan on GenreData  (cost=4.76 rows=13360) (actual time=0.012..7.780 rows=13358 loops=1)
                            -> Hash
                                -> Index range scan on GameData using Price_idx, with index condition: (GameData.Price > 30)  (cost=178.46 rows=396) (actual time=0.043..0.979 rows=396 loops=1)
 |
```

Explanation:
When we index both Singleplayer and Price, we can see that the effects from index 2 still holds (explained above in the index 2 section). However, the key difference is that the Singleplayer cost decreased even more than the cost seen in index 2. For reference, with no indexing, the cost was 1376.25. With the conditions listed in index 2, the cost was 8.08. Yet with the conditions listed in index 3, the cost was 4.76. This is because compared to index 2, we have also indexed SinglePlayer so it will be a lot faster to access the data.

## Advanced Query 2: