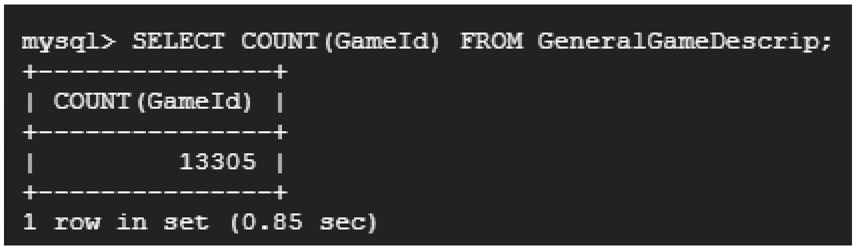
Stage 3: Database Implementation and Indexing (FINAL SUBMISSION)

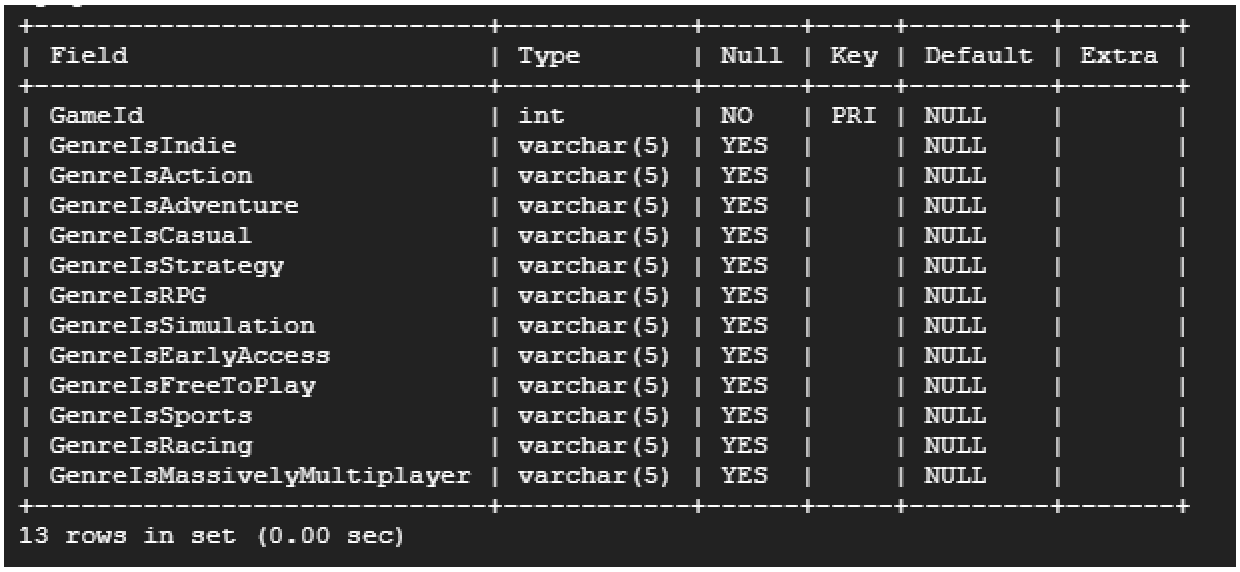
Relational Database - DDL Commands

Proof of at least 100 rows:



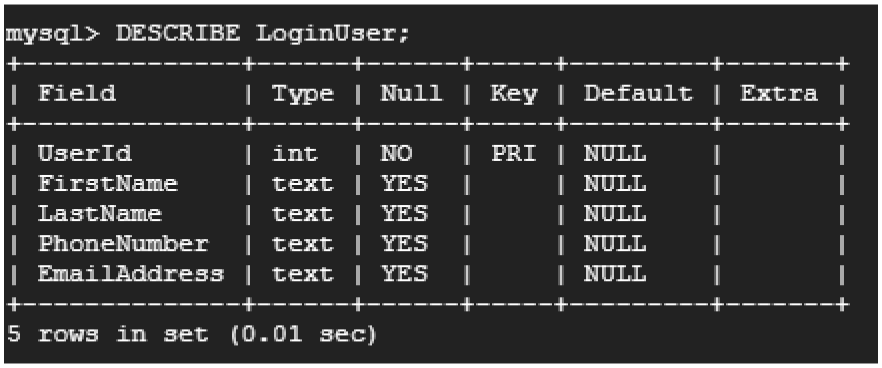
```
mysql> SELECT COUNT(GameId) FROM GeneralGameDescrip;
+---------------+
| COUNT(GameId) |
+---------------+
|         13305 |
+---------------+
1 row in set (0.85 sec)
```

```
CREATE TABLE Genres (
GameId INT NOT NULL,
GenreIsIndie VARCHAR(5),
GenreIsAction VARCHAR(5),
GenreIsAdventure VARCHAR(5),
GenreIsCasual VARCHAR(5),
GenreIsStrategy VARCHAR(5),
GenreIsRPG VARCHAR(5),
GenreIsSimulation VARCHAR(5),
GenreIsEarlyAccess VARCHAR(5),
GenreIsFreeToPlay VARCHAR(5),
GenreIsSports VARCHAR(5),
GenreIsRacing VARCHAR(5),
GenreIsMassivelyMultiplayer VARCHAR(5),
PRIMARY KEY (GameId)
);
```



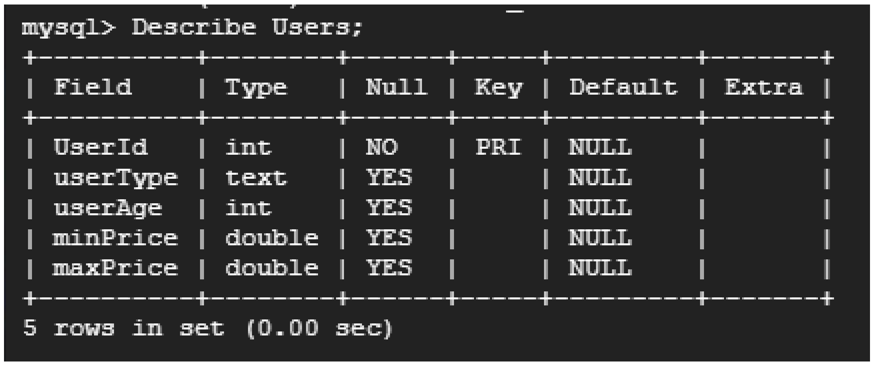
```
+-----------------------------+------------+------+-----+---------+-------+
| Field                       | Type       | Null | Key | Default | Extra |
+-----------------------------+------------+------+-----+---------+-------+
| GameId                      | int        | NO   | PRI | NULL    |       |
| GenreIsIndie                | varchar(5) | YES  |     | NULL    |       |
| GenreIsAction               | varchar(5) | YES  |     | NULL    |       |
| GenreIsAdventure            | varchar(5) | YES  |     | NULL    |       |
| GenreIsCasual               | varchar(5) | YES  |     | NULL    |       |
| GenreIsStrategy             | varchar(5) | YES  |     | NULL    |       |
| GenreIsRPG                  | varchar(5) | YES  |     | NULL    |       |
| GenreIsSimulation           | varchar(5) | YES  |     | NULL    |       |
| GenreIsEarlyAccess          | varchar(5) | YES  |     | NULL    |       |
| GenreIsFreeToPlay           | varchar(5) | YES  |     | NULL    |       |
| GenreIsSports               | varchar(5) | YES  |     | NULL    |       |
| GenreIsRacing               | varchar(5) | YES  |     | NULL    |       |
| GenreIsMassivelyMultiplayer | varchar(5) | YES  |     | NULL    |       |
+-----------------------------+------------+------+-----+---------+-------+
13 rows in set (0.00 sec)
```

```
CREATE TABLE LoginUser (
UserId INT NOT NULL,
FirstName Text,
LastName Text,
PhoneNumber Text,
EmailAddress Text,
PRIMARY KEY (UserId)
);
```



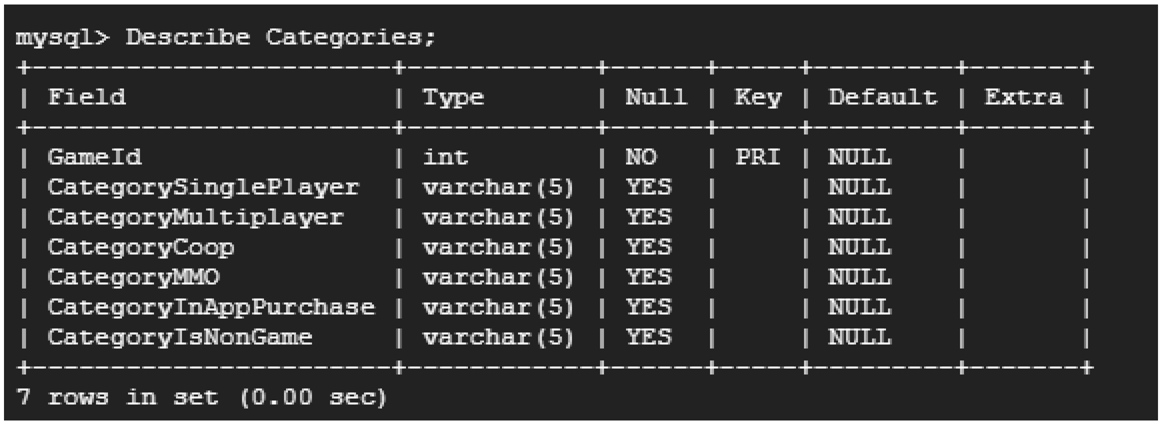
```
mysql> DESCRIBE LoginUser;
+--------------+------+------+-----+---------+-------+
| Field        | Type | Null | Key | Default | Extra |
+--------------+------+------+-----+---------+-------+
| UserId       | int  | NO   | PRI | NULL    |       |
| FirstName    | text | YES  |     | NULL    |       |
| LastName     | text | YES  |     | NULL    |       |
| PhoneNumber  | text | YES  |     | NULL    |       |
| EmailAddress | text | YES  |     | NULL    |       |
+--------------+------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

```
CREATE TABLE Users (
UserId INT NOT NULL,
userType Text,
userAge INT,
minPrice REAL,
maxPrice REAL,
PRIMARY KEY (UserId)
);
```



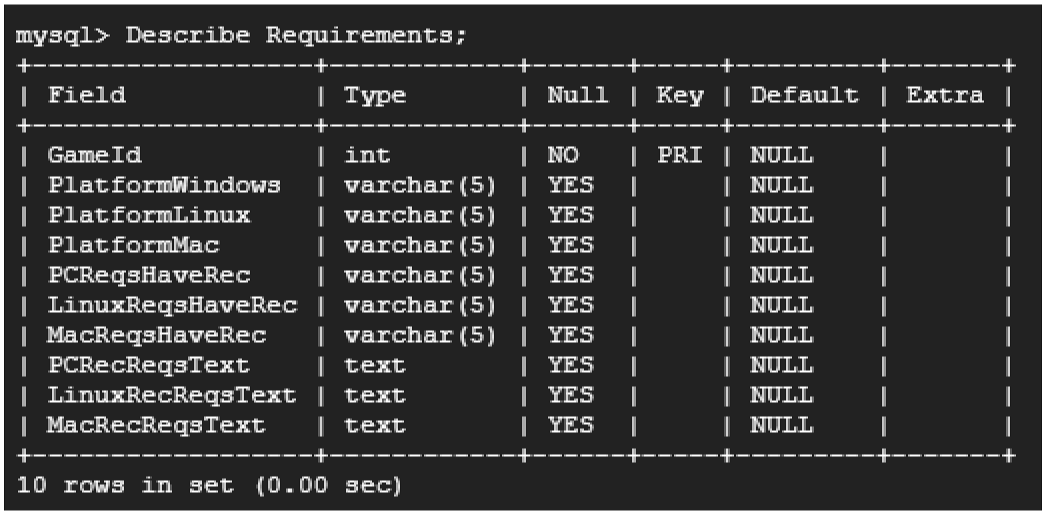
```
mysql> Describe Users;
+----------+--------+------+-----+---------+-------+
| Field    | Type   | Null | Key | Default | Extra |
+----------+--------+------+-----+---------+-------+
| UserId   | int    | NO   | PRI | NULL    |       |
| userType | text   | YES  |     | NULL    |       |
| userAge  | int    | YES  |     | NULL    |       |
| minPrice | double | YES  |     | NULL    |       |
| maxPrice | double | YES  |     | NULL    |       |
+----------+--------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

```
CREATE TABLE Categories (
GameId INT,
CategorySinglePlayer VARCHAR(5),
CategoryMultiplayer VARCHAR(5),
CategoryCoop VARCHAR(5),
CategoryMMO VARCHAR(5),
CategoryInAppPurchase VARCHAR(5),
CategoryIsNonGame VARCHAR(5),
PRIMARY KEY(GameId)
);
```



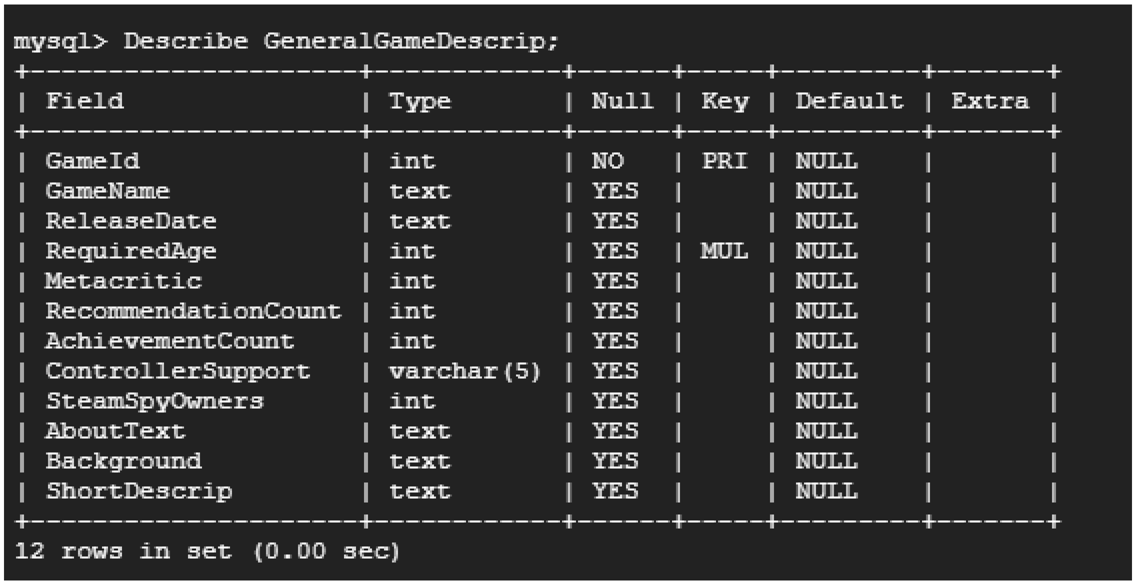
```
mysql> Describe Categories;
+-----------------------+------------+------+-----+---------+-------+
| Field                 | Type       | Null | Key | Default | Extra |
+-----------------------+------------+------+-----+---------+-------+
| GameId                | int        | NO   | PRI | NULL    |       |
| CategorySinglePlayer  | varchar(5) | YES  |     | NULL    |       |
| CategoryMultiplayer   | varchar(5) | YES  |     | NULL    |       |
| CategoryCoop          | varchar(5) | YES  |     | NULL    |       |
| CategoryMMO           | varchar(5) | YES  |     | NULL    |       |
| CategoryInAppPurchase | varchar(5) | YES  |     | NULL    |       |
| CategoryIsNonGame     | varchar(5) | YES  |     | NULL    |       |
+-----------------------+------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

```
CREATE TABLE Requirements (
GameId INT,
PlatformWindows VARCHAR(5),
PlatformLinux VARCHAR(5),
PlatformMac VARCHAR(5),
PCReqsHaveRec VARCHAR(5),
LinuxReqsHaveRec VARCHAR(5),
MacReqsHaveRec VARCHAR(5),
PCRecReqsText Text,
LinuxRecReqsText Text,
MacRecReqsText Text,
PRIMARY KEY(GameId)
);
```



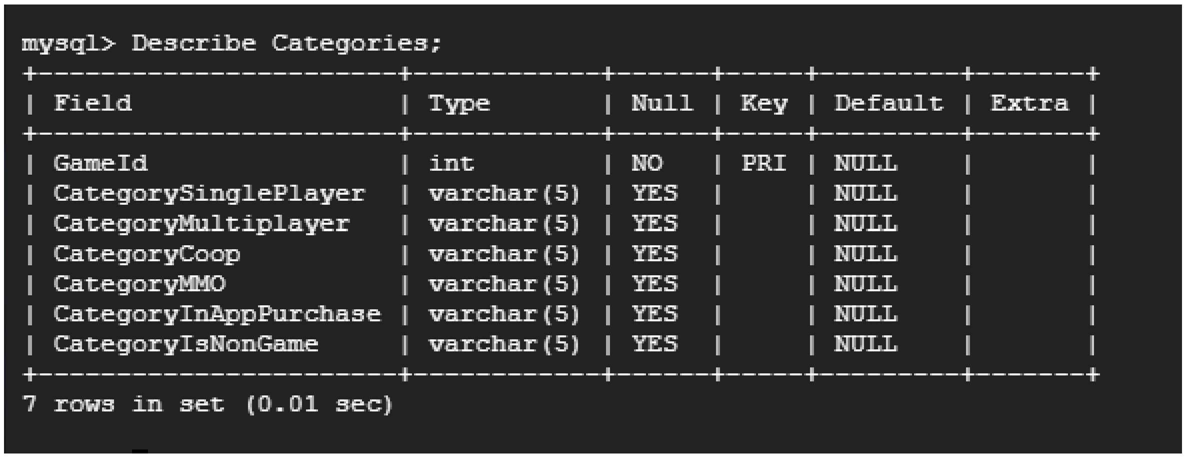
```
mysql> Describe Requirements;
+-------------------+------------+------+-----+---------+-------+
| Field             | Type       | Null | Key | Default | Extra |
+-------------------+------------+------+-----+---------+-------+
| GameId            | int        | NO   | PRI | NULL    |       |
| PlatformWindows   | varchar(5) | YES  |     | NULL    |       |
| PlatformLinux     | varchar(5) | YES  |     | NULL    |       |
| PlatformMac       | varchar(5) | YES  |     | NULL    |       |
| PCReqsHaveRec     | varchar(5) | YES  |     | NULL    |       |
| LinuxReqsHaveRec  | varchar(5) | YES  |     | NULL    |       |
| MacReqsHaveRec    | varchar(5) | YES  |     | NULL    |       |
| PCRecReqsText     | text       | YES  |     | NULL    |       |
| LinuxRecReqsText  | text       | YES  |     | NULL    |       |
| MacRecReqsText    | text       | YES  |     | NULL    |       |
+-------------------+------------+------+-----+---------+-------+
10 rows in set (0.00 sec)
```

```
CREATE TABLE GeneralGameDescrip (
GameId INT,
GameName Text,
ReleaseDate TEXT,
RequiredAge INT,
Metacritic INT,
RecommendationCount INT,
AchievementCount INT,
ControllerSupport VARCHAR(5),
SteamSpyOwners INT,
AboutText Text,
Background Text,
ShortDescrip Text,
PRIMARY KEY (GameId)
);
CREATE TABLE GamePurchasing (
GameId INT,
PriceCurrency Text,
PriceFinal REAL,
isFree VARCHAR(5),
DLCcount Text,
SubscriptionAvail VARCHAR(5),
PRIMARY KEY (GameId)
);
```



```
mysql> Describe GeneralGameDescrip;
+---------------------+------------+------+-----+---------+-------+
| Field               | Type       | Null | Key | Default | Extra |
+---------------------+------------+------+-----+---------+-------+
| GameId              | int        | NO   | PRI | NULL    |       |
| GameName            | text       | YES  |     | NULL    |       |
| ReleaseDate         | text       | YES  |     | NULL    |       |
| RequiredAge         | int        | YES  | MUL | NULL    |       |
| Metacritic          | int        | YES  |     | NULL    |       |
| RecommendationCount | int        | YES  |     | NULL    |       |
| AchievementCount    | int        | YES  |     | NULL    |       |
| ControllerSupport   | varchar(5) | YES  |     | NULL    |       |
| SteamSpyOwners      | int        | YES  |     | NULL    |       |
| AboutText           | text       | YES  |     | NULL    |       |
| Background          | text       | YES  |     | NULL    |       |
| ShortDescrip        | text       | YES  |     | NULL    |       |
+---------------------+------------+------+-----+---------+-------+
12 rows in set (0.00 sec)
```

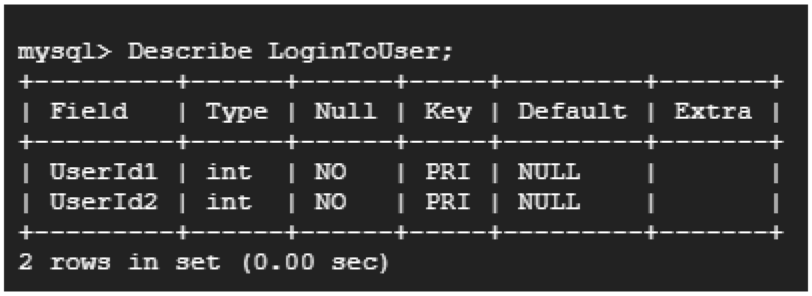
```
CREATE TABLE Categories (
GameId INT,
CategorySinglePlayer VARCHAR(5),
CategoryMultiplayer VARCHAR(5),
CategoryCoop VARCHAR(5),
CategoryMMO VARCHAR(5),
CategoryInAppPurchase VARCHAR(5),
CategoryIsNonGame VARCHAR(5),
PRIMARY KEY (GameId)
)
```



```
mysql> Describe Categories;
+-----------------------+------------+------+-----+---------+-------+
| Field                 | Type       | Null | Key | Default | Extra |
+-----------------------+------------+------+-----+---------+-------+
| GameId                | int        | NO   | PRI | NULL    |       |
| CategorySinglePlayer  | varchar(5) | YES  |     | NULL    |       |
| CategoryMultiplayer   | varchar(5) | YES  |     | NULL    |       |
| CategoryCoop          | varchar(5) | YES  |     | NULL    |       |
| CategoryMMO           | varchar(5) | YES  |     | NULL    |       |
| CategoryInAppPurchase | varchar(5) | YES  |     | NULL    |       |
| CategoryIsNonGame     | varchar(5) | YES  |     | NULL    |       |
+-----------------------+------------+------+-----+---------+-------+
7 rows in set (0.01 sec)
```

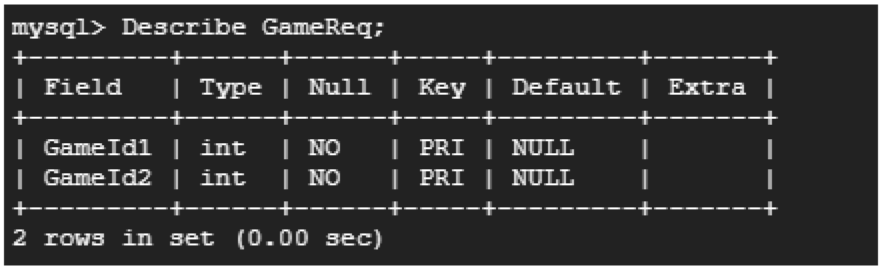
RELATIONSHIP TABLES - DDL Commands:

```
CREATE TABLE LoginToUser (
UserId1 INT,
UserId2 INT,
PRIMARY KEY (UserId1,UserId2),
FOREIGN KEY (UserId1) REFERENCES LoginUser(UserId) ON DELETE
CASCADE,
FOREIGN KEY (UserId2) REFERENCES Users(UserId) ON DELETE CASCADE
);
```



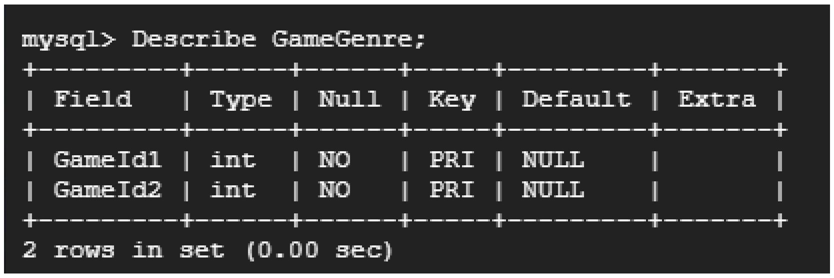
```
mysql> Describe LoginToUser;
+---------+------+------+-----+---------+-------+
| Field   | Type | Null | Key | Default | Extra |
+---------+------+------+-----+---------+-------+
| UserId1 | int  | NO   | PRI | NULL    |       |
| UserId2 | int  | NO   | PRI | NULL    |       |
+---------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

```
CREATE TABLE GameReq (
GameId1 INT,
GameId2 INT,
PRIMARY KEY (GameId1, GameId2),
FOREIGN KEY(GameId1) REFERENCES GeneralGameDescrip(GameId) ON
DELETE CASCADE,
FOREIGN KEY(GameId2) REFERENCES Requirements(GameId) ON DELETE
CASCADE
);
```



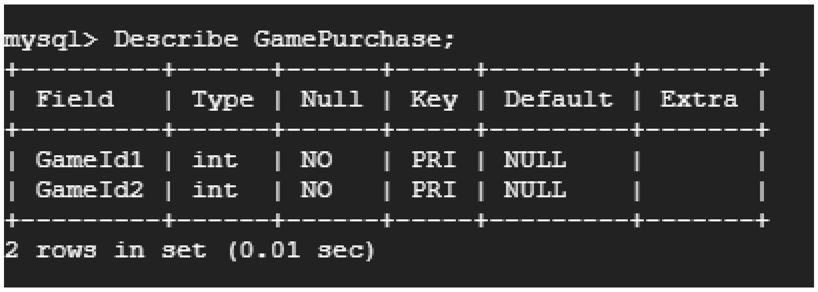
```
mysql> Describe GameReq;
+---------+------+------+-----+---------+-------+
| Field   | Type | Null | Key | Default | Extra |
+---------+------+------+-----+---------+-------+
| GameId1 | int  | NO   | PRI | NULL    |       |
| GameId2 | int  | NO   | PRI | NULL    |       |
+---------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

```
CREATE TABLE GameGenre (
GameId1 INT,
GameId2 INT,
PRIMARY KEY (GameId1, GameId2),
FOREIGN KEY (GameId1) REFERENCES GeneralGameDescrip (GameId) ON
DELETE CASCADE,
FOREIGN KEY (GameId2) REFERENCES Genres (GameId) ON DELETE
CASCADE
);
```



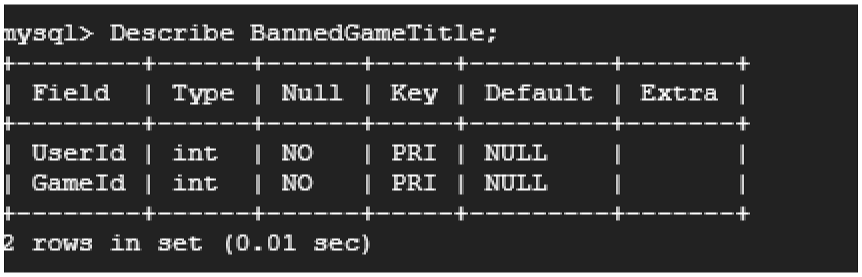
```
mysql> Describe GameGenre;
+---------+------+------+-----+---------+-------+
| Field   | Type | Null | Key | Default | Extra |
+---------+------+------+-----+---------+-------+
| GameId1 | int  | NO   | PRI | NULL    |       |
| GameId2 | int  | NO   | PRI | NULL    |       |
+---------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

```
CREATE TABLE GamePurchase (
GameId1 INT,
GameId2 INT,
PRIMARY KEY (GameId1, GameId2),
FOREIGN KEY (GameId1) REFERENCES GeneralGameDescrip (GameID) ON
DELETE CASCADE,
FOREIGN KEY (GameId2) REFERENCES GamePurchasing (GameId) ON
DELETE CASCADE
);
```



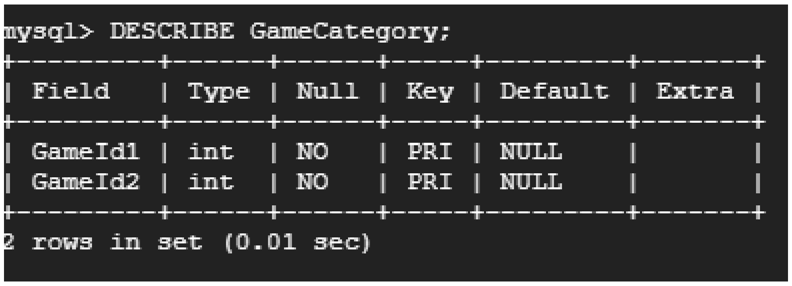
```
mysql> Describe GamePurchase;
+---------+------+------+-----+---------+-------+
| Field   | Type | Null | Key | Default | Extra |
+---------+------+------+-----+---------+-------+
| GameId1 | int  | NO   | PRI | NULL    |       |
| GameId2 | int  | NO   | PRI | NULL    |       |
+---------+------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

```
CREATE TABLE BannedGameTitle (
UserId INT,
GameId INT,
PRIMARY KEY (UserId, GameId),
FOREIGN KEY (UserId) REFERENCES LoginUser(UserId) ON DELETE
CASCADE,
FOREIGN KEY (GameId) REFERENCES GeneralGameDescrip(GameId) ON
DELETE CASCADE
);
```



```
mysql> Describe BannedGameTitle;
+--------+------+------+-----+---------+-------+
| Field  | Type | Null | Key | Default | Extra |
+--------+------+------+-----+---------+-------+
| UserId | int  | NO   | PRI | NULL    |       |
| GameId | int  | NO   | PRI | NULL    |       |
+--------+------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

```
CREATE TABLE GameCategory (
GameId1 INT,
GameId2 INT,
PRIMARY KEY (GameId1, GameId2),
FOREIGN KEY (GameId1) REFERENCES GeneralGameDescrip (GameId) ON
DELETE CASCADE,
FOREIGN KEY (GameId2) REFERENCES Categories (GameId) ON DELETE
CASCADE
);
```



```
mysql> DESCRIBE GameCategory;
+---------+------+------+-----+---------+-------+
| Field   | Type | Null | Key | Default | Extra |
+---------+------+------+-----+---------+-------+
| GameId1 | int  | NO   | PRI | NULL    |       |
| GameId2 | int  | NO   | PRI | NULL    |       |
+---------+------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

** Inserted our CSV file data into the below table and then split it into our main tables and then deleted this table **

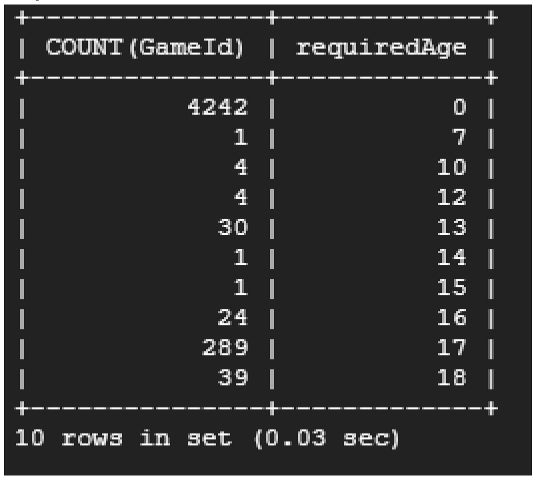
```
CREATE TABLE AllData (
QueryID INT,
ResponseID INT,
QueryName Text,
ResponseName Text,
ReleaseDate Text,
RequiredAge Int,
DemoCount      INT,
DeveloperCount      INT,
DLCCount INT,
Metacritic      Int,
MovieCount INT,
PackageCount INT,
RecommendationCount Int,
PublisherCount      INT,
ScreenshotCount INT,
SteamSpyOwners      Int,
SteamSpyOwnersVariance Int,
SteamSpyPlayersEstimate Int,
SteamSpyPlayersVariance Int,
AchievementCount Int,
AchievementHighlightedCount INT,
ControllerSupport VARCHAR(5),
IsFree VARCHAR(5),
FreeVerAvail VARCHAR(5),
PurchaseAvail VARCHAR(5),
SubscriptionAvail VARCHAR(5),
PlatformWindows VARCHAR(5),
PlatformLinux VARCHAR(5),
PlatformMac VARCHAR(5),
PCReqsHaveMin VARCHAR(5),
PCReqsHaveRec VARCHAR(5),
LinuxReqsHaveMin VARCHAR(5),
LinuxReqsHaveRec VARCHAR(5),
MacReqsHaveMin      VARCHAR(5),
MacReqsHaveRec      VARCHAR(5),
CategorySinglePlayer VARCHAR(5),
CategoryMultiplayer VARCHAR(5),
CategoryCoop VARCHAR(5),
CategoryMMO VARCHAR(5),
CategoryInAppPurchase VARCHAR(5),
CategoryIncludeSrcSDK VARCHAR(5),
CategoryIncludeLevelEditor VARCHAR(5),
CategoryVRSupport VARCHAR(5),
GenreIsNonGame VARCHAR(5),
GenreIsIndie VARCHAR(5),
GenreIsAction VARCHAR(5),
GenreIsAdventure VARCHAR(5),
GenreIsCasual VARCHAR(5),
GenreIsStrategy VARCHAR(5),
GenreIsRPG VARCHAR(5),
GenreIsSimulation VARCHAR(5),
GenreIsEarlyAccess VARCHAR(5),
GenreIsFreeToPlay VARCHAR(5),
GenreIsSports VARCHAR(5),
GenreIsRacing VARCHAR(5),
GenreIsMassivelyMultiplayer VARCHAR(5),
PriceCurrency TEXT,
PriceInitial FLOAT,
PriceFinal FLOAT,
SupportEmail Text,
SupportURL Text,
AboutText Text,
Background Text,
ShortDescrip Text,
DetailedDescrip Text,
DRMNotice Text,
ExtUserAcctNotice Text,
HeaderImage Text,
LegalNotice Text,
Reviews Text,
SupportedLanguages Text,
Website Text,
PCMinReqsText Text,
PCRecReqsText Text,
LinuxMinReqsText Text,
LinuxRecReqsText Text,
MacMinReqsText Text,
MacRecReqsText Text,
PRIMARY KEY (QueryID)
);
```

Advanced SQL Queries

<u>ADVANCED QUERY 1:</u>
Searches for number of non-free Action Games and Groups by Age:

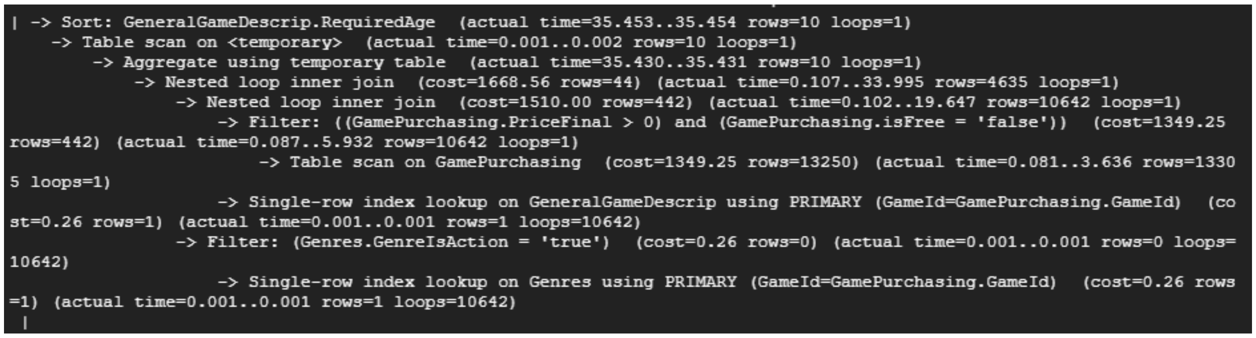
```
SELECT COUNT(GameId), requiredAge

FROM (SELECT * FROM GamePurchasing WHERE isFree = "false")
nonFree NATURAL JOIN Genres NATURAL JOIN GeneralGameDescrip

WHERE GenreIsAction = "true" AND PriceFinal > 0 GROUP BY
requiredAge ORDER BY requiredAge;
```

Top 15:



```
+---------------+-------------+
| COUNT(GameId) | requiredAge |
+---------------+-------------+
|          4242 |           0 |
|             1 |           7 |
|             4 |          10 |
|             4 |          12 |
|            30 |          13 |
|             1 |          14 |
|             1 |          15 |
|            24 |          16 |
|           289 |          17 |
|            39 |          18 |
+---------------+-------------+
10 rows in set (0.03 sec)
```

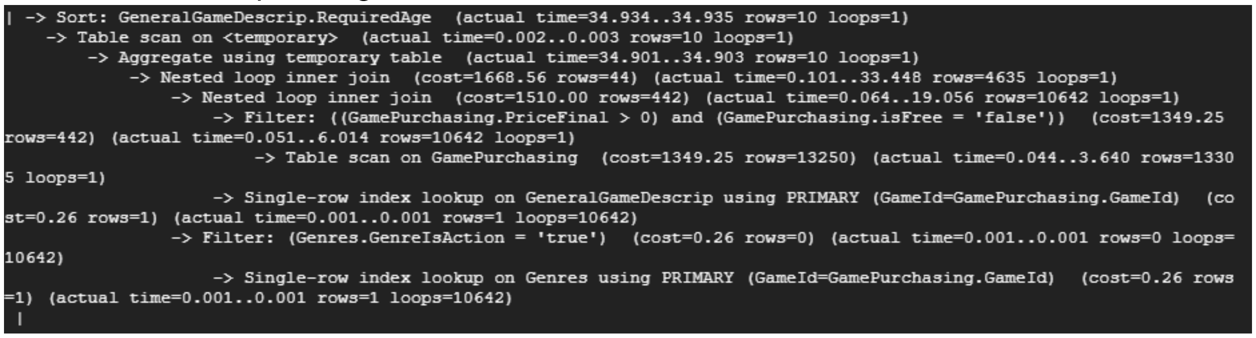
Without Index:



```
| -> Sort: GeneralGameDescrip.RequiredAge  (actual time=35.453..35.454 rows=10 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.002 rows=10 loops=1)
        -> Aggregate using temporary table  (actual time=35.430..35.431 rows=10 loops=1)
            -> Nested loop inner join  (cost=1668.56 rows=44) (actual time=0.107..33.995 rows=4635 loops=1)
                -> Nested loop inner join  (cost=1510.00 rows=442) (actual time=0.102..19.647 rows=10642 loops=1)
                    -> Filter: ((GamePurchasing.PriceFinal > 0) and (GamePurchasing.isFree = 'false'))  (cost=1349.25
rows=442) (actual time=0.087..5.932 rows=10642 loops=1)
                        -> Table scan on GamePurchasing  (cost=1349.25 rows=13250) (actual time=0.081..3.636 rows=1330
5 loops=1)
                    -> Single-row index lookup on GeneralGameDescrip using PRIMARY (GameId=GamePurchasing.GameId)  (co
st=0.26 rows=1) (actual time=0.001..0.001 rows=1 loops=10642)
                -> Filter: (Genres.GenreIsAction = 'true')  (cost=0.26 rows=0) (actual time=0.001..0.001 rows=0 loops=
10642)
                    -> Single-row index lookup on Genres using PRIMARY (GameId=GamePurchasing.GameId)  (cost=0.26 rows
=1) (actual time=0.001..0.001 rows=1 loops=10642)
 |
```

Index: `CREATE INDEX RegAge ON GeneralGameDescrip(requiredAge)`

With Index on RequiredAge:

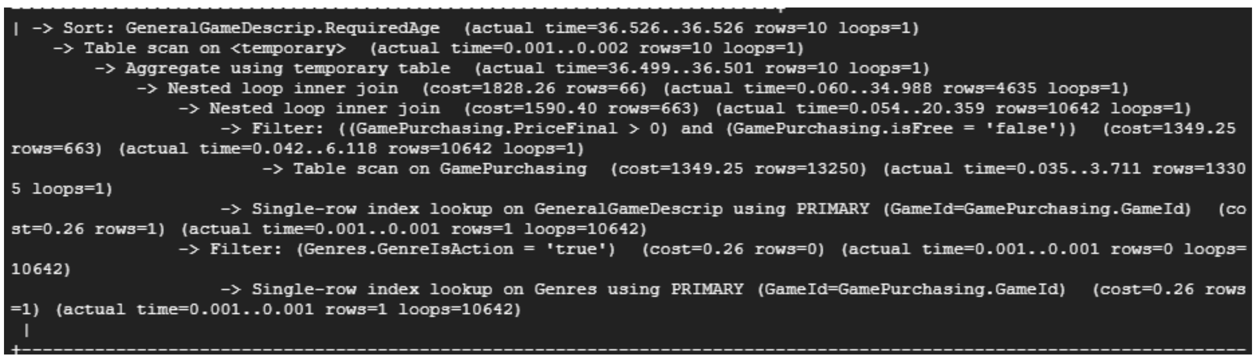


```
| -> Sort: GeneralGameDescrip.RequiredAge  (actual time=34.934..34.935 rows=10 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.003 rows=10 loops=1)
        -> Aggregate using temporary table  (actual time=34.901..34.903 rows=10 loops=1)
            -> Nested loop inner join  (cost=1668.56 rows=44) (actual time=0.101..33.448 rows=4635 loops=1)
                -> Nested loop inner join  (cost=1510.00 rows=442) (actual time=0.064..19.056 rows=10642 loops=1)
                    -> Filter: ((GamePurchasing.PriceFinal > 0) and (GamePurchasing.isFree = 'false'))  (cost=1349.25
rows=442) (actual time=0.051..6.014 rows=10642 loops=1)
                        -> Table scan on GamePurchasing  (cost=1349.25 rows=13250) (actual time=0.044..3.640 rows=1330
5 loops=1)
                    -> Single-row index lookup on GeneralGameDescrip using PRIMARY (GameId=GamePurchasing.GameId)  (co
st=0.26 rows=1) (actual time=0.001..0.001 rows=1 loops=10642)
                -> Filter: (Genres.GenreIsAction = 'true')  (cost=0.26 rows=0) (actual time=0.001..0.001 rows=0 loops=
10642)
                    -> Single-row index lookup on Genres using PRIMARY (GameId=GamePurchasing.GameId)  (cost=0.26 rows
=1) (actual time=0.001..0.001 rows=1 loops=10642)
 |
```

The cost of this query was the exact same with or without the index on RequiredAge. This is probably the case because we are simply taking all requiredAge's into account and we do not need to query for a specific age.

Index: `CREATE INDEX PriceFinal ON GamePurchasing(PriceFinal)`

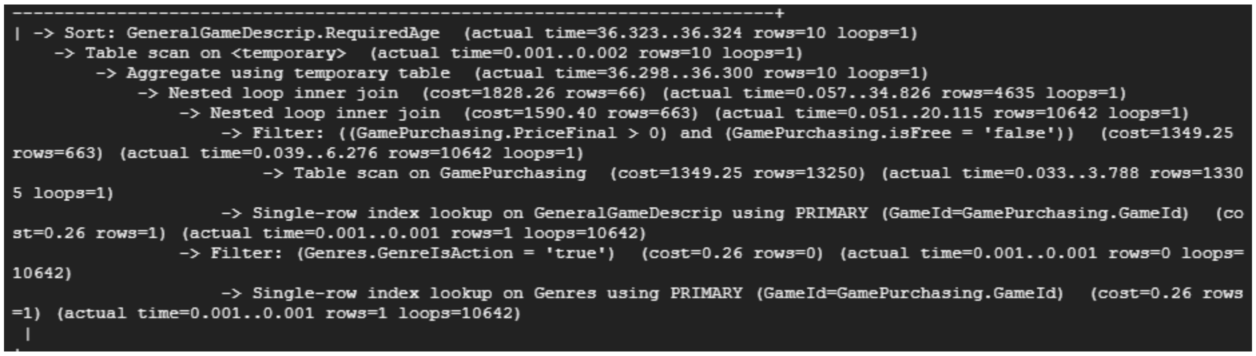
With Index on PriceFinal:



```
| -> Sort: GeneralGameDescrip.RequiredAge  (actual time=36.526..36.526 rows=10 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.002 rows=10 loops=1)
        -> Aggregate using temporary table  (actual time=36.499..36.501 rows=10 loops=1)
            -> Nested loop inner join  (cost=1828.26 rows=66) (actual time=0.060..34.988 rows=4635 loops=1)
                -> Nested loop inner join  (cost=1590.40 rows=663) (actual time=0.054..20.359 rows=10642 loops=1)
                    -> Filter: ((GamePurchasing.PriceFinal > 0) and (GamePurchasing.isFree = 'false'))  (cost=1349.25
rows=663) (actual time=0.042..6.118 rows=10642 loops=1)
                        -> Table scan on GamePurchasing  (cost=1349.25 rows=13250) (actual time=0.035..3.711 rows=1330
5 loops=1)
                    -> Single-row index lookup on GeneralGameDescrip using PRIMARY (GameId=GamePurchasing.GameId)  (co
st=0.26 rows=1) (actual time=0.001..0.001 rows=1 loops=10642)
                -> Filter: (Genres.GenreIsAction = 'true')  (cost=0.26 rows=0) (actual time=0.001..0.001 rows=0 loops=
10642)
                    -> Single-row index lookup on Genres using PRIMARY (GameId=GamePurchasing.GameId)  (cost=0.26 rows
=1) (actual time=0.001..0.001 rows=1 loops=10642)
 |
+------------------------------------------------------------------------------------------------------------------
```

We realized that for this query, creating indexes would either have no effect or an adverse effect on the overall cost of the query. Running the query with an index on required age showed no changes to the overall cost and running the query with an index on PriceFinal actually created more cost because we had to search our index which delayed our overall query speed.

With Index on PriceFinal and RequiredAge:



```
-----------------------------------------------------------------------+
| -> Sort: GeneralGameDescrip.RequiredAge  (actual time=36.323..36.324 rows=10 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.002 rows=10 loops=1)
        -> Aggregate using temporary table  (actual time=36.298..36.300 rows=10 loops=1)
            -> Nested loop inner join  (cost=1828.26 rows=66) (actual time=0.057..34.826 rows=4635 loops=1)
                -> Nested loop inner join  (cost=1590.40 rows=663) (actual time=0.051..20.115 rows=10642 loops=1)
                    -> Filter: ((GamePurchasing.PriceFinal > 0) and (GamePurchasing.isFree = 'false'))  (cost=1349.25
rows=663) (actual time=0.039..6.276 rows=10642 loops=1)
                        -> Table scan on GamePurchasing  (cost=1349.25 rows=13250) (actual time=0.033..3.788 rows=1330
5 loops=1)
                    -> Single-row index lookup on GeneralGameDescrip using PRIMARY (GameId=GamePurchasing.GameId)  (co
st=0.26 rows=1) (actual time=0.001..0.001 rows=1 loops=10642)
                -> Filter: (Genres.GenreIsAction = 'true')  (cost=0.26 rows=0) (actual time=0.001..0.001 rows=0 loops=
10642)
                    -> Single-row index lookup on Genres using PRIMARY (GameId=GamePurchasing.GameId)  (cost=0.26 rows
=1) (actual time=0.001..0.001 rows=1 loops=10642)
 |
```

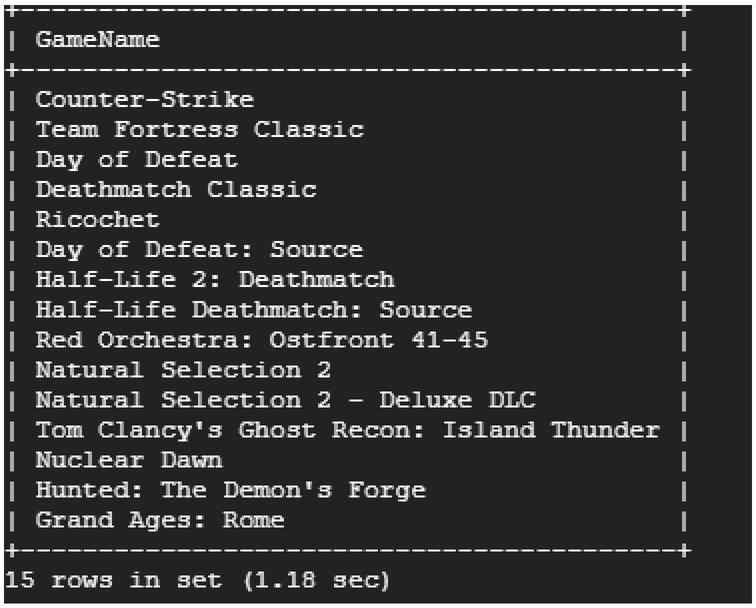
We realized that for this query we still have an adverse effect on the overall query time because we are still using the PriceFinal index which could either have no effect or an adverse effect on the query so when we combine that with required age we are guaranteed to get a worse query time.

ADVANCE QUERY 2:

Gets the game name for paid non-singleplayer games with prices less than or equal to the average paid non-singleplayer game.

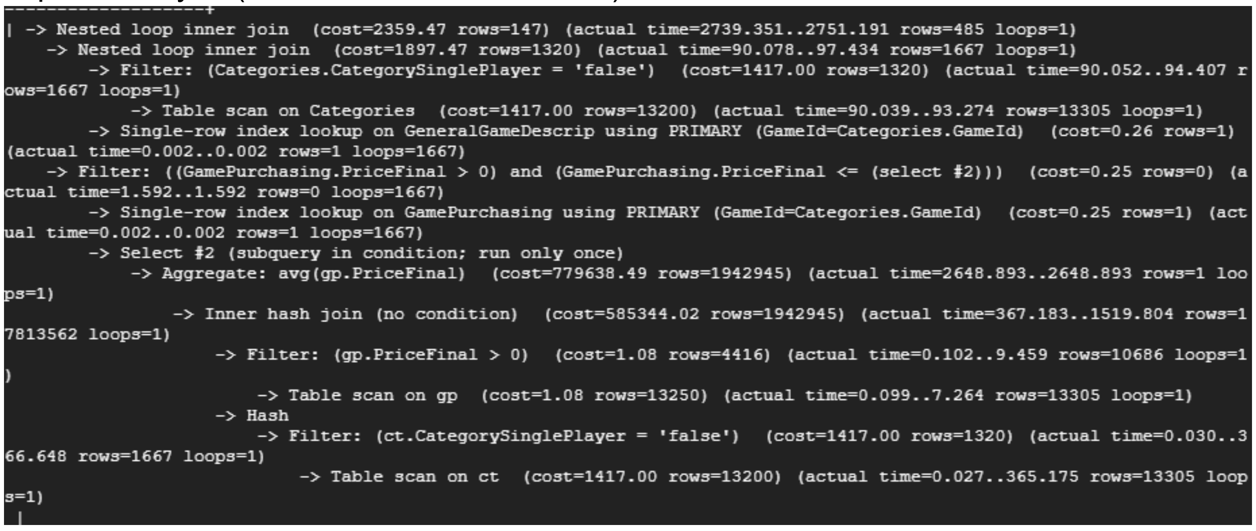
```
SELECT GameName

FROM GeneralGameDescrip NATURAL JOIN Categories NATURAL JOIN
GamePurchasing

WHERE CategorySinglePlayer="false" AND PriceFinal>0 AND
PriceFinal<=(SELECT AVG(gp.PriceFinal) FROM GamePurchasing gp
join Categories ct WHERE PriceFinal>0 AND
ct.CategorySinglePlayer="false");
```

First 15:



```
+------------------------------------------+
| GameName                                 |
+------------------------------------------+
| Counter-Strike                           |
| Team Fortress Classic                    |
| Day of Defeat                            |
| Deathmatch Classic                       |
| Ricochet                                 |
| Day of Defeat: Source                    |
| Half-Life 2: Deathmatch                  |
| Half-Life Deathmatch: Source             |
| Red Orchestra: Ostfront 41-45            |
| Natural Selection 2                      |
| Natural Selection 2 - Deluxe DLC         |
| Tom Clancy's Ghost Recon: Island Thunder |
| Nuclear Dawn                             |
| Hunted: The Demon's Forge                |
| Grand Ages: Rome                         |
+------------------------------------------+
15 rows in set (1.18 sec)
```

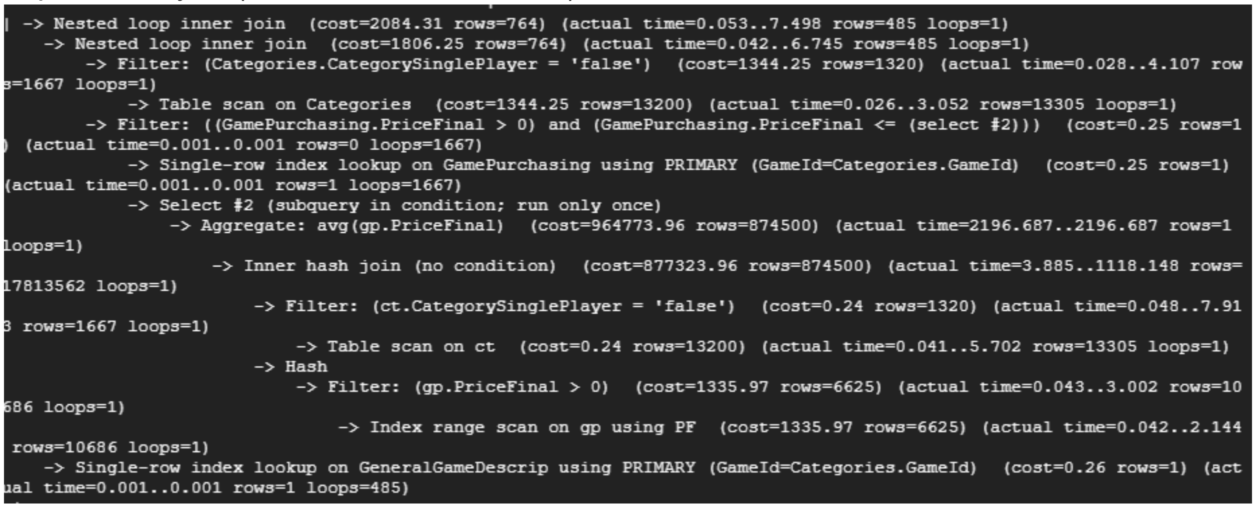
Explain Analyze (Without Index - PriceFinal):



```
------------------+
| -> Nested loop inner join  (cost=2359.47 rows=147) (actual time=2739.351..2751.191 rows=485 loops=1)
    -> Nested loop inner join  (cost=1897.47 rows=1320) (actual time=90.078..97.434 rows=1667 loops=1)
        -> Filter: (Categories.CategorySinglePlayer = 'false')  (cost=1417.00 rows=1320) (actual time=90.052..94.407 r
ows=1667 loops=1)
            -> Table scan on Categories  (cost=1417.00 rows=13200) (actual time=90.039..93.274 rows=13305 loops=1)
        -> Single-row index lookup on GeneralGameDescrip using PRIMARY (GameId=Categories.GameId)  (cost=0.26 rows=1)
(actual time=0.002..0.002 rows=1 loops=1667)
    -> Filter: ((GamePurchasing.PriceFinal > 0) and (GamePurchasing.PriceFinal <= (select #2)))  (cost=0.25 rows=0) (a
ctual time=1.592..1.592 rows=0 loops=1667)
        -> Single-row index lookup on GamePurchasing using PRIMARY (GameId=Categories.GameId)  (cost=0.25 rows=1) (act
ual time=0.002..0.002 rows=1 loops=1667)
        -> Select #2 (subquery in condition; run only once)
            -> Aggregate: avg(gp.PriceFinal)  (cost=779638.49 rows=1942945) (actual time=2648.893..2648.893 rows=1 loo
ps=1)
                -> Inner hash join (no condition)  (cost=585344.02 rows=1942945) (actual time=367.183..1519.804 rows=1
7813562 loops=1)
                    -> Filter: (gp.PriceFinal > 0)  (cost=1.08 rows=4416) (actual time=0.102..9.459 rows=10686 loops=1
)
                        -> Table scan on gp  (cost=1.08 rows=13250) (actual time=0.099..7.264 rows=13305 loops=1)
                    -> Hash
                        -> Filter: (ct.CategorySinglePlayer = 'false')  (cost=1417.00 rows=1320) (actual time=0.030..3
66.648 rows=1667 loops=1)
                            -> Table scan on ct  (cost=1417.00 rows=13200) (actual time=0.027..365.175 rows=13305 loop
s=1)
 |
```

Index command: `CREATE INDEX PF ON GamePurchasing(PriceFinal)`

Explain Analyze (With Index - PriceFinal):

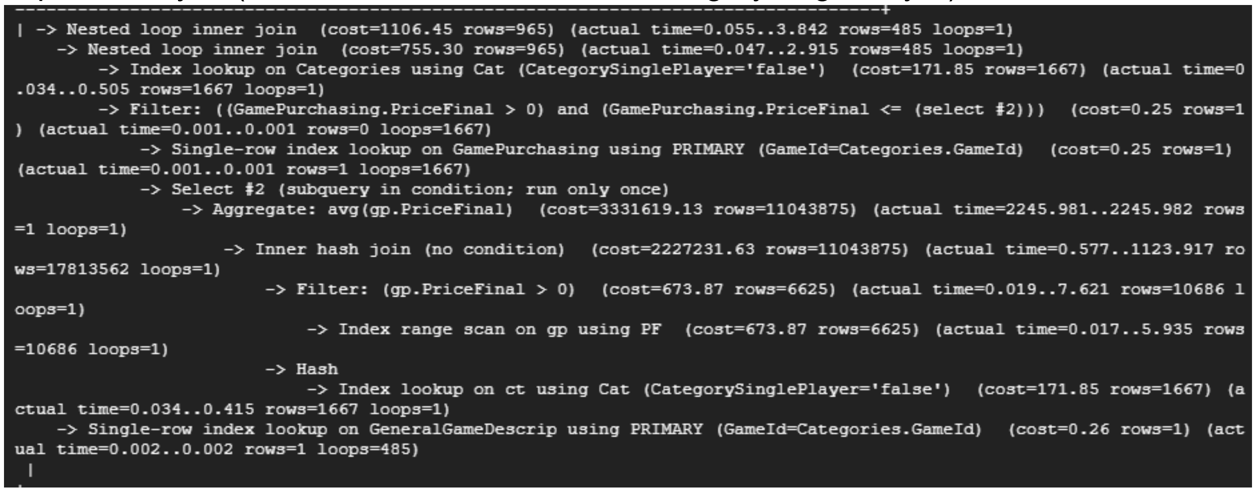


```
| -> Nested loop inner join  (cost=2084.31 rows=764) (actual time=0.053..7.498 rows=485 loops=1)
    -> Nested loop inner join  (cost=1806.25 rows=764) (actual time=0.042..6.745 rows=485 loops=1)
        -> Filter: (Categories.CategorySinglePlayer = 'false')  (cost=1344.25 rows=1320) (actual time=0.028..4.107 row
s=1667 loops=1)
            -> Table scan on Categories  (cost=1344.25 rows=13200) (actual time=0.026..3.052 rows=13305 loops=1)
        -> Filter: ((GamePurchasing.PriceFinal > 0) and (GamePurchasing.PriceFinal <= (select #2)))  (cost=0.25 rows=1
) (actual time=0.001..0.001 rows=0 loops=1667)
            -> Single-row index lookup on GamePurchasing using PRIMARY (GameId=Categories.GameId)  (cost=0.25 rows=1)
(actual time=0.001..0.001 rows=1 loops=1667)
            -> Select #2 (subquery in condition; run only once)
                -> Aggregate: avg(gp.PriceFinal)  (cost=964773.96 rows=874500) (actual time=2196.687..2196.687 rows=1
loops=1)
                    -> Inner hash join (no condition)  (cost=877323.96 rows=874500) (actual time=3.885..1118.148 rows=
17813562 loops=1)
                        -> Filter: (ct.CategorySinglePlayer = 'false')  (cost=0.24 rows=1320) (actual time=0.048..7.91
3 rows=1667 loops=1)
                            -> Table scan on ct  (cost=0.24 rows=13200) (actual time=0.041..5.702 rows=13305 loops=1)
                        -> Hash
                            -> Filter: (gp.PriceFinal > 0)  (cost=1335.97 rows=6625) (actual time=0.043..3.002 rows=10
686 loops=1)
                                -> Index range scan on gp using PF  (cost=1335.97 rows=6625) (actual time=0.042..2.144
 rows=10686 loops=1)
    -> Single-row index lookup on GeneralGameDescrip using PRIMARY (GameId=Categories.GameId)  (cost=0.26 rows=1) (act
ual time=0.001..0.001 rows=1 loops=485)
```

This Index showed improvement on the overall cost of the query. This makes sense because we indexed upon a data value that we were searching or scanning based on and there we could index to values that were all greater than 0 so we could skip scanning the price finals that would index to 0.

Index command: `CREATE INDEX PF ON GamePurchasing(PriceFinal)`
Index command: `CREATE INDEX Cat ON Categories(CategorySinglePlayer)`

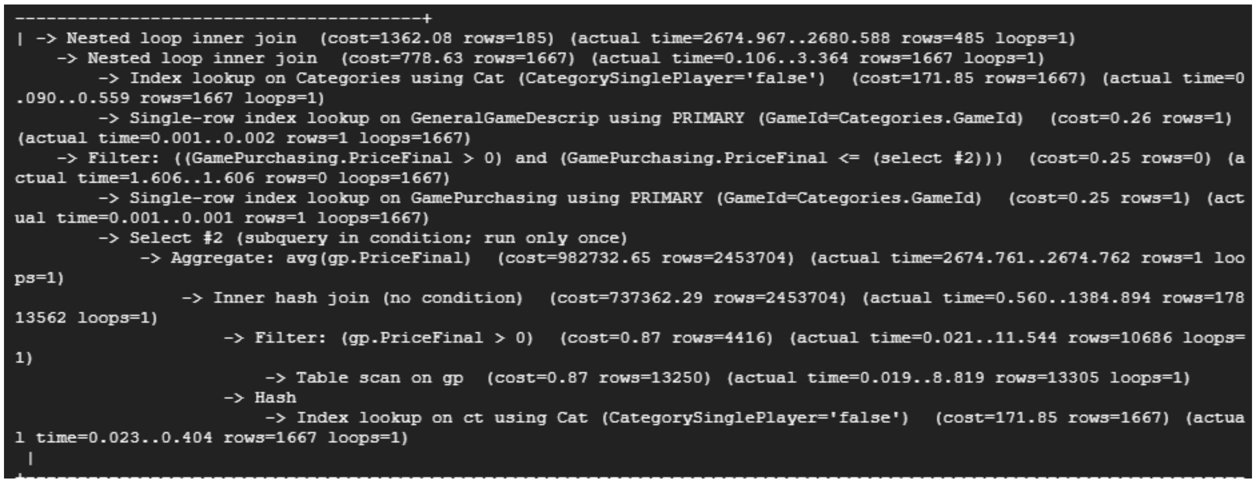
Explain Analyze (With Index - PriceFinal and CategorySinglePlayer)



```
-----------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=1106.45 rows=965) (actual time=0.055..3.842 rows=485 loops=1)
    -> Nested loop inner join  (cost=755.30 rows=965) (actual time=0.047..2.915 rows=485 loops=1)
        -> Index lookup on Categories using Cat (CategorySinglePlayer='false')  (cost=171.85 rows=1667) (actual time=0
.034..0.505 rows=1667 loops=1)
        -> Filter: ((GamePurchasing.PriceFinal > 0) and (GamePurchasing.PriceFinal <= (select #2)))  (cost=0.25 rows=1
) (actual time=0.001..0.001 rows=0 loops=1667)
            -> Single-row index lookup on GamePurchasing using PRIMARY (GameId=Categories.GameId)  (cost=0.25 rows=1)
(actual time=0.001..0.001 rows=1 loops=1667)
            -> Select #2 (subquery in condition; run only once)
                -> Aggregate: avg(gp.PriceFinal)  (cost=3331619.13 rows=11043875) (actual time=2245.981..2245.982 rows
=1 loops=1)
                    -> Inner hash join (no condition)  (cost=2227231.63 rows=11043875) (actual time=0.577..1123.917 ro
ws=17813562 loops=1)
                        -> Filter: (gp.PriceFinal > 0)  (cost=673.87 rows=6625) (actual time=0.019..7.621 rows=10686 l
oops=1)
                            -> Index range scan on gp using PF  (cost=673.87 rows=6625) (actual time=0.017..5.935 rows
=10686 loops=1)
                        -> Hash
                            -> Index lookup on ct using Cat (CategorySinglePlayer='false')  (cost=171.85 rows=1667) (a
ctual time=0.034..0.415 rows=1667 loops=1)
    -> Single-row index lookup on GeneralGameDescrip using PRIMARY (GameId=Categories.GameId)  (cost=0.26 rows=1) (act
ual time=0.002..0.002 rows=1 loops=485)
 |
```

We also added an index on our category for SinglePlayer which is a varchar value that can either be assigned to true, false, or NULL. We found that this index was efficient because the total cost of the query was lower. This is probably because we were only searching for false values so we could quickly index to all the false values.

Explain Analyze (With Index - CategorySinglePlayer):



```
-------------------------------------+
| -> Nested loop inner join  (cost=1362.08 rows=185) (actual time=2674.967..2680.588 rows=485 loops=1)
    -> Nested loop inner join  (cost=778.63 rows=1667) (actual time=0.106..3.364 rows=1667 loops=1)
        -> Index lookup on Categories using Cat (CategorySinglePlayer='false')  (cost=171.85 rows=1667) (actual time=0
.090..0.559 rows=1667 loops=1)
        -> Single-row index lookup on GeneralGameDescrip using PRIMARY (GameId=Categories.GameId)  (cost=0.26 rows=1)
 (actual time=0.001..0.002 rows=1 loops=1667)
    -> Filter: ((GamePurchasing.PriceFinal > 0) and (GamePurchasing.PriceFinal <= (select #2)))  (cost=0.25 rows=0) (a
ctual time=1.606..1.606 rows=0 loops=1667)
        -> Single-row index lookup on GamePurchasing using PRIMARY (GameId=Categories.GameId)  (cost=0.25 rows=1) (act
ual time=0.001..0.001 rows=1 loops=1667)
        -> Select #2 (subquery in condition; run only once)
            -> Aggregate: avg(gp.PriceFinal)  (cost=982732.65 rows=2453704) (actual time=2674.761..2674.762 rows=1 loo
ps=1)
                -> Inner hash join (no condition)  (cost=737362.29 rows=2453704) (actual time=0.560..1384.894 rows=178
13562 loops=1)
                    -> Filter: (gp.PriceFinal > 0)  (cost=0.87 rows=4416) (actual time=0.021..11.544 rows=10686 loops=
1)
                        -> Table scan on gp  (cost=0.87 rows=13250) (actual time=0.019..8.819 rows=13305 loops=1)
                    -> Hash
                        -> Index lookup on ct using Cat (CategorySinglePlayer='false')  (cost=171.85 rows=1667) (actua
l time=0.023..0.404 rows=1667 loops=1)
 |
+--------------------------------------------------------------------------------------------------------------------
```

We queried on just CategorySinglePlayer and we found that the cost had improved from the original query but the overall cost was still higher than the query where we also combined the Price Final index. This is probably because the Price Final index is very effective for this query.

When selecting data columns to index in the future, we take into account the following aspects:

1. Select columns that are commonly used for queries: Select columns that are frequently used for queries for indexing to improve the efficiency of queries. For example, users may often use price or required age to index individual games and query related data, so we consider creating indexes on data such as price and required age.

2. Select columns with wide data distribution: Select columns with wide data distribution for indexing, which can reduce the number of rows that need to be searched in the database and improve query efficiency. We create indexes on features that are available in every game because these columns have a wide data distribution.

3. Avoid selecting large data columns: Don't select columns that are too large to create indexes, as this increases the size of the index and can cause queries to slow down. For example, we did not create indexes on columns containing large images or binary data.

4. Avoid choosing duplicate data columns: If we choose duplicate data columns for indexing, it may increase the size of the index, waste storage space, and may cause queries to slow down. Therefore we choose columns such as price, such data is repeated very little.

5. Consider the usage of the table: If we frequently perform insert, update or delete operations, then creating indexes on the table may increase the maintenance cost. In this case, we consider using a small number of indexes and optimizing them when necessary.