

## Command Line Connection:

```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to game-finder-379921.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
rbhatt06@cloudshell:~ (game-finder-379921)$ gcloud sql connect mygamefinder --user=root
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1471
Server version: 8.0.26-google (Google)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
```

## DDL Command for Tables:

Reviews Table:

```
CREATE TABLE Reviews (
  app_id INT,
  app_name TEXT,
  review_text TEXT,
  review_score INT,
  review_votes INT
);
```

Purchases Table:

```
CREATE TABLE Purchases (
  User_ID TEXT,
  Name TEXT,
  Behavior INT,
  Hours INT
);
```

Games Table:

```
CREATE TABLE Games (  
    QueryID INT,  
    ResponseID INT,  
    QueryName VARCHAR(255),  
    ResponseName VARCHAR(255),  
    ReleaseDate DATE,  
    RequiredAge INT,  
    DemoCount INT,  
    DeveloperCount INT,  
    DLCCount INT,  
    Metacritic INT,  
    MovieCount INT,  
    PackageCount INT,  
    RecommendationCount INT,  
    PublisherCount INT,  
    ScreenshotCount INT,  
    SteamSpyOwners BIGINT,  
    SteamSpyOwnersVariance BIGINT,  
    SteamSpyPlayersEstimate BIGINT,  
    SteamSpyPlayersVariance BIGINT,  
    AchievementCount INT,  
    AchievementHighlightedCount INT,  
    PriceCurrency VARCHAR(10),  
    PriceInitial FLOAT,  
    PriceFinal FLOAT,  
    SupportEmail VARCHAR(255),  
    SupportURL VARCHAR(255),  
    AboutText TEXT,  
    Background VARCHAR(255),  
    ShortDescrip TEXT,  
    DetailedDescrip TEXT,  
    DRMNotice TEXT,  
    ExtUserAcctNotice TEXT,  
    HeaderImage VARCHAR(255),  
    LegalNotice TEXT,  
    Reviews TEXT,  
    SupportedLanguages VARCHAR(255),  
    Website VARCHAR(255)  
);
```

PC table:

```
CREATE TABLE PC (  
  QueryName VARCHAR(255),  
  ControllerSupport BOOLEAN,  
  IsFree BOOLEAN,  
  FreeVerAvail BOOLEAN,  
  PurchaseAvail BOOLEAN,  
  SubscriptionAvail BOOLEAN,  
  PlatformWindows BOOLEAN,  
  PlatformLinux BOOLEAN,  
  PlatformMac BOOLEAN,  
  PCReqsHaveMin BOOLEAN,  
  PCReqsHaveRec BOOLEAN,  
  LinuxReqsHaveMin BOOLEAN,  
  LinuxReqsHaveRec BOOLEAN,  
  MacReqsHaveMin BOOLEAN,  
  MacReqsHaveRec BOOLEAN,  
  CategorySinglePlayer BOOLEAN,  
  CategoryMultiplayer BOOLEAN,  
  CategoryCoop BOOLEAN,  
  CategoryMMO BOOLEAN,  
  CategoryInAppPurchase BOOLEAN,  
  CategoryIncludeSrcSDK BOOLEAN,  
  CategoryIncludeLevelEditor BOOLEAN,  
  CategoryVRSupport BOOLEAN,  
  GenreIsNonGame BOOLEAN,  
  GenreIsIndie BOOLEAN,  
  GenreIsAction BOOLEAN,  
  GenreIsAdventure BOOLEAN,  
  GenreIsCasual BOOLEAN,  
  GenreIsStrategy BOOLEAN,  
  GenreIsRPG BOOLEAN,  
  GenreIsSimulation BOOLEAN,  
  GenreIsEarlyAccess BOOLEAN,  
  GenreIsFreeToPlay BOOLEAN,  
  GenreIsSports BOOLEAN,  
  GenreIsRacing BOOLEAN,  
  GenreIsMassivelyMultiplayer BOOLEAN,  
  SupportedLanguages VARCHAR(255),  
  Website VARCHAR(255),  
  PCMinReqsText TEXT,
```

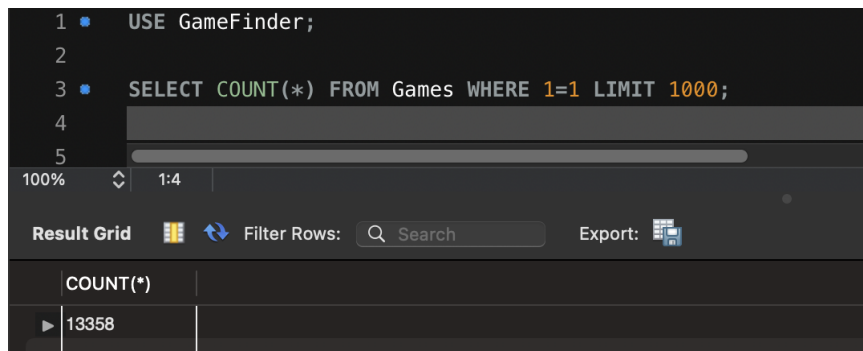
```
PCRecReqstext TEXT,  
LinuxMinReqstext TEXT,  
LinuxRecReqstext TEXT,  
MacMinReqstext TEXT,  
MacRecReqstext TEXT  
);
```

## Inserting at least 1000 rows:

Number of Rows in each Table:

Games: 13,358

Proof:



The screenshot shows a database query tool interface. The SQL query entered is: `USE GameFinder;` followed by `SELECT COUNT(*) FROM Games WHERE 1=1 LIMIT 1000;`. The result grid below the query shows a single row with the value 13358 under the column header COUNT(\*).

| COUNT(*) |
|----------|
| 13358    |

PC: 13,358

Proof:

```
1 • USE GameFinder;  
2  
3 • SELECT COUNT(*) FROM PC;  
4  
5
```

100% 24:3

Result Grid Filter Rows: Search Exp

| COUNT(*) |
|----------|
| ▶ 13358  |

Purchases: 200,000

Proof:

```
1 • USE GameFinder;  
2  
3 • SELECT COUNT(*) FROM Purchases;  
4  
5
```

100% 31:3

Result Grid Filter Rows: Search Exp

| COUNT(*) |
|----------|
| ▶ 200000 |

Reviews: 6,417,107

Proof:

```
1  USE GameFinder;
2
3  SELECT COUNT(*) FROM Reviews;
4
5
```

100% 29:3

Result Grid Filter Rows: Search Export:

| COUNT(*) |
|----------|
| 6417107  |

## Advanced SQL Queries:

SQL Query 1:

```
USE GameFinder;

-- Select * From PC;
SELECT g.QueryName, AVG(r.review_score) AS avg_review_score
FROM Games g JOIN Reviews r ON g.QueryName = r.app_name
GROUP BY g.QueryID, g.QueryName
HAVING AVG(r.review_score) >= 0.75
LIMIT 15;
```

output:

```
1 • USE GameFinder;
2
3 -- Select * From PC;
4 • SELECT g.QueryName, AVG(r.review_score) AS avg_review_score
5 FROM Games g JOIN Reviews r ON g.QueryName = r.app_name
6 GROUP BY g.QueryID, g.QueryName
7 HAVING AVG(r.review_score) >= 0.75
8 LIMIT 15;
9
```

100% 1:9

Result Grid Filter Rows: Search Export: Fetch rows:

| QueryName                      | avg_review_score |
|--------------------------------|------------------|
| Counter-Strike                 | 0.9372           |
| Call of Duty: World at War     | 0.8657           |
| Kingdoms of Amalur: Reckoning™ | 0.7680           |
| Orcs Must Die!                 | 0.9423           |
| Terraria                       | 0.9416           |
| Bastion                        | 0.9540           |
| Space Pirates and Zombies      | 0.8667           |
| Arma 3                         | 0.7771           |
| Waves                          | 0.9046           |
| Ticket to Ride                 | 0.7574           |
| Vessel                         | 0.7855           |
| Project Zomboid                | 0.7849           |
| Alan Wake                      | 0.7898           |
| Penguins Arena: Sedna's World  | 0.7561           |
| Monaco                         | 0.8292           |

## SQL Query 2:

```
use GameFinder;

SELECT DISTINCT pu.Name, AVG(pu.Hours), pc.PlatformWindows
FROM Purchases pu JOIN PC pc ON pu.Name = pc.QueryName
WHERE pc.PlatformWindows = 1
GROUP BY pu.Name
ORDER BY AVG(pu.Hours) DESC
LIMIT 15;
```

Output:

```
1 • USE GameFinder;
2
3 -- SELECT COUNT(*) FROM Reviews;
4
5 • SELECT DISTINCT pu.Name, AVG(pu.Hours), pc.PlatformWindows
6 FROM Purchases pu JOIN PC pc ON pu.Name = pc.QueryName
7 WHERE pc.PlatformWindows = 1
8 GROUP BY pu.Name
9 ORDER BY AVG(pu.Hours) DESC
10 LIMIT 15;
```

Result Grid

| Name                                     | AVG(pu.Hours) | PlatformWindo... |
|--|---------------|------------------|
| Eastside Hockey Manager                  | 648.0000      | 1                |
| Perpetuum                                | 201.0000      | 1                |
| X-Plane 10 Global - 64 Bit               | 134.5000      | 1                |
| Dota 2                                   | 101.9042      | 1                |
| Cultures - Northland                     | 97.5000       | 1                |
| Europa Universalis IV                    | 96.2157       | 1                |
| Counter-Strike                           | 94.8926       | 1                |
| Pro Cycling Manager 2014                 | 89.2500       | 1                |
| Sid Meier's Civilization V               | 87.3243       | 1                |
| Movie Studio 13 Platinum - Steam Powered | 83.5000       | 1                |
| Empires                                  | 78.5000       | 1                |
| Arma 3                                   | 76.1509       | 1                |
| The Treasures of Montezuma 4             | 69.8750       | 1                |
| Freaking Meatbags                        | 67.0000       | 1                |
| Supreme Ruler Cold War                   | 64.8571       | 1                |

## Explain Analysis:

SQL Query 1: Explain Analyze (Default)

```
USE GameFinder;

EXPLAIN ANALYZE
SELECT g.QueryName, AVG(r.review_score) AS avg_review_score
FROM Games g JOIN Reviews r ON g.QueryName = r.app_name
GROUP BY g.QueryID, g.QueryName
HAVING AVG(r.review_score) >= 0.75
LIMIT 15;
```

Output:

```
'-> Limit: 15 row(s) (actual time=46685.864..46685.923 rows=15 loops=1)\n
```



```
-> Filter: (avg(r.review_score) >= 0.75) (actual
time=46685.863..46685.921 rows=15 loops=1)\n
-> Table scan on <temporary> (actual time=0.004..0.041 rows=81 loops=1)\n
-> Aggregate using temporary table (actual time=46685.855..46685.897
rows=81 loops=1)\n
-> Filter: (g.QueryName = r.app_name)
(cost=6856972331.35 rows=6851028832) (actual time=1327.588..34089.638
rows=6240381 loops=1)\n
-> Inner hash join (<hash>(g.QueryName)=<hash>(r.app_name))
(cost=6856972331.35 rows=6851028832) (actual time=1327.581..31565.167
rows=6240381 loops=1)\n
-> Table scan on r
(cost=632.76 rows=7208574) (actual time=77.414..22220.783 rows=6417107
loops=1)\n
-> Hash\n
-> Table scan on g (cost=1951.90 rows=9504) (actual time=42.023..1226.656
rows=13358 loops=1)\n'
```

#### SQL Query 1: Explain Analyze (Default) index on Games

```
USE GameFinder;

EXPLAIN ANALYZE
SELECT g.QueryName, AVG(r.review_score) AS avg_review_score
FROM Games g JOIN Reviews r ON g.QueryName = r.app_name
GROUP BY g.QueryID, g.QueryName
HAVING AVG(r.review_score) >= 0.75
LIMIT 15;

CREATE INDEX games ON Games(QueryName);
```

#### Output:

```
'-> Limit: 15 row(s) (actual time=66388.390..66388.466 rows=15 loops=1)\n
-> Filter: (avg(r.review_score) >= 0.75) (actual time=66388.388..66388.463
rows=15 loops=1)\n
-> Table scan on <temporary> (actual time=0.003..0.042 rows=81 loops=1)\n
-> Aggregate using temporary table (actual time=66388.324..66388.370
rows=81 loops=1)\n
-> Nested loop inner join (cost=3398674.30 rows=7208574) (actual
time=228.348..51932.477 rows=6240381 loops=1)\n
-> Filter: (r.app_name is not null) (cost=875673.40 rows=7208574) (actual
time=85.468..16735.284 rows=6417107 loops=1)\n
-> Table scan on r (cost=875673.40 rows=7208574) (actual
```

```
time=85.464..16090.710 rows=6417107 loops=1)\n-> Index lookup on g using idx (QueryName=r.app_name), with index  
condition: (g.QueryName = r.app_name) (cost=0.25 rows=1) (actual  
time=0.004..0.005 rows=1 loops=6417107)\n'
```

### SQL Query 1: Explain Analyze on Reviews

```
EXPLAIN ANALYZE  
SELECT g.QueryName, AVG(r.review_score) AS avg_review_score  
FROM Games g JOIN Reviews r ON g.QueryName = r.app_name  
GROUP BY g.QueryID, g.QueryName  
HAVING AVG(r.review_score) >= 0.75  
LIMIT 15;  
  
CREATE INDEX reviews ON Reviews(QueryName, review_score);
```

### Output:

```
'-> Limit: 15 row(s) (actual time=64108.762..64108.837 rows=15 loops=1)\n-> Filter: (avg(r.review_score) >= 0.75) (actual time=64108.759..64108.833  
rows=15 loops=1)\n-> Table scan on <temporary> (actual time=0.002..0.041 rows=81 loops=1)\n-> Aggregate using temporary table (actual time=64108.753..64108.799  
rows=81 loops=1)\n-> Nested loop inner join (cost=3398674.30 rows=7208574) (actual  
time=82.962..49337.779 rows=6240381 loops=1)\n-> Filter: (r.app_name is not null) (cost=875673.40 rows=7208574) (actual  
time=82.867..18079.131 rows=6417107 loops=1)\n-> Table scan on r (cost=875673.40 rows=7208574) (actual  
time=82.863..17443.687 rows=6417107 loops=1)\n-> Index lookup on g using idx (QueryName=r.app_name), with index  
condition: (g.QueryName = r.app_name) (cost=0.25 rows=1) (actual  
time=0.004..0.005 rows=1 loops=6417107)\n'
```

### SQL Query 1: Final Analysis

The three Explain Analysis commands attached to the above sql query to change the overall efficiency of the query.

- 1) In the default state, no index was created. In doing so, the query creates a new temp table to join and search for the proceeding outputs. This in turn would have the slowest run time. The final line of output is: `Table scan on g (cost=1951.90 rows=9504) (actual time=42.023..1226.656 rows=13358 loops=1)\n'`. This shows that the total cost of running the query would be 1951 and the total time would be approx 42.023.
- 2) In the second Explain Analyze command, we create an index on the Games table. This helps to build more efficiency by having a search up on values in the Games table. Thus, there is no need for a table scan because the database engine looks up the corresponding rows in the Games dataset using the index. The query still necessitates a database scan on the Reviews table, though. The final output line is: `(cost=0.25 rows=1) (actual time=0.004..0.005)`. This shows that the cost has gone down a lot from the default. The difference is that it only indexes the Games table.
- 3) In the final analysis, the Reviews table's QueryName and review\_score columns are both given indexes in the final output. This index enables the database engine to quickly search the Reviews table using an index query rather than a table scan. As a consequence, compared to the second output, the query execution time becomes more enhanced.

2nd Advanced Query:

Initial explain analyze:

```
-> Limit: 15 row(s) (actual time=588.968..588.972 rows=15 loops=1)
    -> Sort: AVG(pu.Hours) DESC, limit input to 15 row(s) per chunk
    (actual time=588.967..588.970 rows=15 loops=1)
        -> Table scan on <temporary> (actual time=0.002..0.803 rows=2951
        loops=1)
            -> Aggregate using temporary table (actual
            time=586.866..587.847 rows=2951 loops=1)
                -> Filter: (pu.`Name` = pc.QueryName) (cost=28475006.24
                rows=28472028) (actual time=16.014..432.604 rows=126514 loops=1)
                    -> Inner hash join
                    (<hash>(pu.`Name`)=<hash>(pc.QueryName)) (cost=28475006.24 rows=28472028)
                    (actual time=16.011..385.651 rows=126514 loops=1)
                        -> Table scan on pu (cost=2.41 rows=200479)
                        (actual time=0.026..114.561 rows=200000 loops=1)
                            -> Hash
                                -> Filter: (pc.PlatformWindows = 1)
                                (cost=1556.45 rows=1420) (actual time=0.037..12.206 rows=13355 loops=1)
                                    -> Table scan on pc (cost=1556.45
                                    rows=1420) (actual time=0.031..11.058 rows=13358 loops=1)
```

```
1 use GameFinder;
2 EXPLAIN ANALYZE
3
4 SELECT DISTINCT pu.Name, AVG(pu.Hours), pc.PlatformWindows
5 FROM Purchases pu JOIN PC pc ON pu.Name = pc.QueryName
6 WHERE pc.PlatformWindows = 1
7 GROUP BY pu.Name
8 ORDER BY AVG(pu.Hours) DESC
9 LIMIT 15;
```

100% 16:2

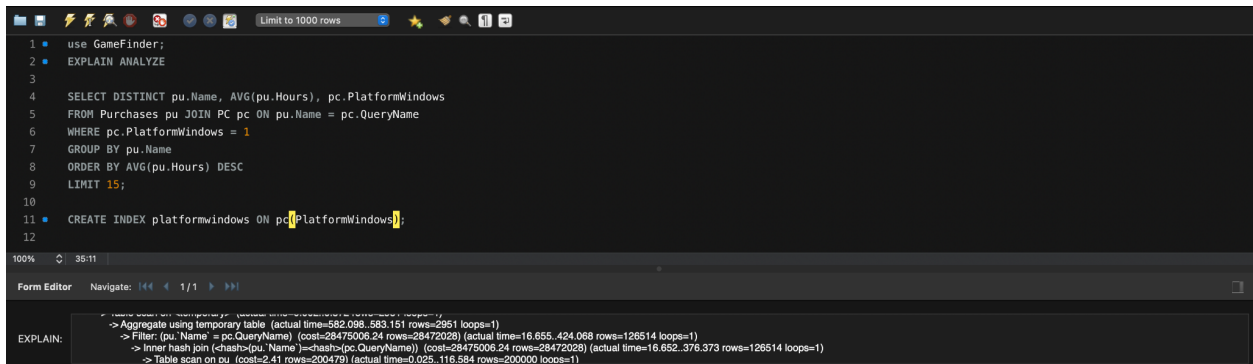
Form Editor Navigate: 1/1

EXPLAIN:

```
-> Limit: 15 row(s) (actual time=588.968..588.972 rows=15 loops=1)
    -> Sort: AVG(pu.Hours) DESC, limit input to 15 row(s) per chunk (actual time=588.967..588.970 rows=15 loops=1)
        -> Table scan on <temporary> (actual time=0.002..0.803 rows=2951 loops=1)
            -> Aggregate using temporary table (actual time=586.866..587.847 rows=2951 loops=1)
```

Explain Analyze for CREATE INDEX platformwindows ON pc (PlatformWindows):

```
-> Limit: 15 row(s) (actual time=583.194..583.199 rows=15 loops=1)
  -> Sort: AVG(pu.Hours) DESC, limit input to 15 row(s) per chunk
(actual time=583.192..583.196 rows=15 loops=1)
  -> Table scan on <temporary> (actual time=0.002..0.817 rows=2951
loops=1)
    -> Aggregate using temporary table (actual
time=581.105..582.108 rows=2951 loops=1)
      -> Filter: (pu.`Name` = pc.QueryName) (cost=28475006.24
rows=28472028) (actual time=15.376..422.494 rows=126514 loops=1)
        -> Inner hash join
(<hash>(pu.`Name`)=<hash>(pc.QueryName)) (cost=28475006.24 rows=28472028)
(actual time=15.373..375.322 rows=126514 loops=1)
          -> Table scan on pu (cost=2.41 rows=200479)
(actual time=0.021..114.809 rows=200000 loops=1)
            -> Hash
              -> Filter: (pc.PlatformWindows = 1)
(cost=1556.45 rows=1420) (actual time=0.032..11.671 rows=13355 loops=1)
                -> Table scan on pc (cost=1556.45
rows=14202) (actual time=0.027..10.495 rows=13358 loops=1)
```



The screenshot shows a SQL IDE with a query editor and an execution plan window. The query is as follows:

```
1 use GameFinder;
2 EXPLAIN ANALYZE
3
4 SELECT DISTINCT pu.Name, AVG(pu.Hours), pc.PlatformWindows
5 FROM Purchases pu JOIN PC pc ON pu.Name = pc.QueryName
6 WHERE pc.PlatformWindows = 1
7 GROUP BY pu.Name
8 ORDER BY AVG(pu.Hours) DESC
9 LIMIT 15;
10
11 CREATE INDEX platformwindows ON pc(PlatformWindows);
12
```

The execution plan window shows the following details:

```
EXPLAIN:
  -> Aggregate using temporary table (actual time=582.098..583.151 rows=2951 loops=1)
    -> Filter: (pu.`Name` = pc.QueryName) (cost=28475006.24 rows=28472028) (actual time=16.655..424.068 rows=126514 loops=1)
      -> Inner hash join (<hash>(pu.`Name`)=<hash>(pc.QueryName)) (cost=28475006.24 rows=28472028) (actual time=16.652..376.373 rows=126514 loops=1)
        -> Table scan on pu (cost=2.41 rows=200479) (actual time=0.025..116.584 rows=200000 loops=1)
```

Explain Analyze for CREATE INDEX queryname ON pc (QueryName):

```
-> Limit: 15 row(s) (actual time=583.194..583.199 rows=15 loops=1)
    -> Sort: AVG(pu.Hours) DESC, limit input to 15 row(s) per chunk
    (actual time=583.192..583.196 rows=15 loops=1)
        -> Table scan on <temporary> (actual time=0.002..0.817 rows=2951
        loops=1)
            -> Aggregate using temporary table (actual
            time=581.105..582.108 rows=2951 loops=1)
                -> Filter: (pu.`Name` = pc.QueryName) (cost=28475006.24
                rows=28472028) (actual time=15.376..422.494 rows=126514 loops=1)
                    -> Inner hash join
                    (<hash>(pu.`Name`)=<hash>(pc.QueryName)) (cost=28475006.24 rows=28472028)
                    (actual time=15.373..375.322 rows=126514 loops=1)
                        -> Table scan on pu (cost=2.41 rows=200479)
                        (actual time=0.021..114.809 rows=200000 loops=1)
                            -> Hash
                            -> Filter: (pc.PlatformWindows = 1)
                            (cost=1556.45 rows=1420) (actual time=0.032..11.671 rows=13355 loops=1)
                                -> Table scan on pc (cost=1556.45
                                rows=14202) (actual time=0.027..10.495 rows=13358 loops=1)
```

```
1  use GameFinder;
2  EXPLAIN ANALYZE
3
4  SELECT DISTINCT pu.Name, AVG(pu.Hours), pc.PlatformWindows
5  FROM Purchases pu JOIN PC pc ON pu.Name = pc.QueryName
6  WHERE pc.PlatformWindows = 1
7  GROUP BY pu.Name
8  ORDER BY AVG(pu.Hours) DESC
9  LIMIT 15;
10
11 CREATE INDEX queryname ON pc(QueryName);
12
```

100% 41:11

Form Editor Navigate: << 1/1 >>

EXPLAIN: -> Limit: 15 row(s) (actual time=607.300..607.307 rows=15 loops=1)  
-> Sort: AVG(pu.Hours) DESC, limit input to 15 row(s) per chunk (actual time=607.299..607.305 rows=15 loops=1)  
-> Table scan on <temporary> (actual time=0.003..0.871 rows=2951 loops=1)  
-> Aggregate using temporary table (actual time=605.059..606.135 rows=2951 loops=1)

Explain Analyze for CREATE INDEX hours ON pu (Hours):

```
-> Limit: 15 row(s) (actual time=601.609..601.613 rows=15 loops=1)
```

```
-> Sort: AVG(pu.Hours) DESC, limit input to 15 row(s) per chunk
(actual time=601.607..601.611 rows=15 loops=1)
  -> Table scan on <temporary> (actual time=0.002..0.835 rows=2951
loops=1)
    -> Aggregate using temporary table (actual
time=599.361..600.416 rows=2951 loops=1)
      -> Filter: (pu.`Name` = pc.QueryName) (cost=28475006.24
rows=28472028) (actual time=16.675..434.637 rows=126514 loops=1)
        -> Inner hash join
        (<hash>(pu.`Name`)=<hash>(pc.QueryName)) (cost=28475006.24 rows=28472028)
        (actual time=16.672..385.550 rows=126514 loops=1)
          -> Table scan on pu (cost=2.41 rows=200479)
          (actual time=0.023..119.901 rows=200000 loops=1)
            -> Hash
              -> Filter: (pc.PlatformWindows = 1)
              (cost=1556.45 rows=1420) (actual time=0.045..12.820 rows=13355 loops=1)
                -> Table scan on pc (cost=1556.45
rows=1420) (actual time=0.040..11.693 rows=13358 loops=1)
```

```
1 use GameFinder;
2 EXPLAIN ANALYZE
3
4 SELECT DISTINCT pu.Name, AVG(pu.Hours), pc.PlatformWindows
5 FROM Purchases pu JOIN PC pc ON pu.Name = pc.QueryName
6 WHERE pc.PlatformWindows = 1
7 GROUP BY pu.Name
8 ORDER BY AVG(pu.Hours) DESC
9 LIMIT 15;
10
11 CREATE INDEX hours ON pu(Hours);
12
```

100% 10:9

Form Editor Navigate: 1/1

EXPLAIN:

```
-> Limit: 15 row(s) (actual time=581.078..581.085 rows=15 loops=1)
-> Sort: AVG(pu.Hours) DESC, limit input to 15 row(s) per chunk (actual time=581.077..581.082 rows=15 loops=1)
-> Table scan on <temporary> (actual time=0.003..0.849 rows=2951 loops=1)
-> Aggregate using temporary table (actual time=578.899..579.936 rows=2951 loops=1)
```

## SQL Query 2: Final Analysis

The three Explain Analysis commands attached to the above sql query to change the overall efficiency of the query.

- 1) In the default state, no index was created. In doing so, the query creates a new temp table to join and search for the proceeding outputs. This in turn would have the slowest run time.
- 2) In the second Explain Analyze command, we create an index on the PC table. This helps to build more efficiency by having a search up on values in the PC table. Thus, there is no need for a table scan because the database engine looks up the corresponding rows in the PC dataset using the index. The query still necessitates a database scan on the Purchases table, though. As we can see from the explain analyze, the cost has gone down because of this, making it more efficient.
- 3) In the third analysis, the Reviews table's QueryName is given an index in the final output. This index enables the database engine to quickly search the Reviews table using an index query rather than a table scan. This helps reduce the run time of the query yet again, making it more efficient then the default query.
- 4) In the final analysis, we give the Hours column from the Purchases table an index. This index, however, actually increased our run time and cost, as the database engine still needed to go through and calculate the hours after scanning through the hours with the index query.