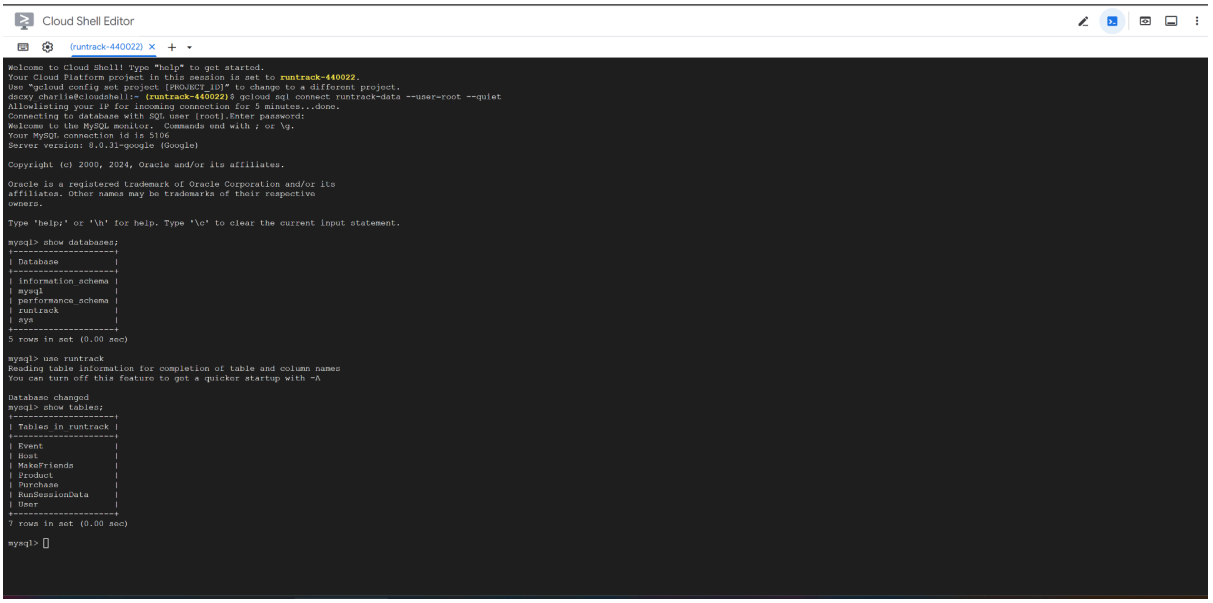
Stage 3

Rubric Part 3

1. Screenshot of the connection to our database



2. DDL commands for tables. (also on github under doc)

```
-- Create User table with UUID as primary key
CREATE TABLE User (
    UserId CHAR(36) PRIMARY KEY,
    FirstName VARCHAR(100),
    LastName VARCHAR(100),
    Email VARCHAR(255) UNIQUE,
    PhoneNumber VARCHAR(20)
);

-- Create MakeFriends table with UUIDs
CREATE TABLE MakeFriends (
    FriendshipId CHAR(36) PRIMARY KEY,
    UserId CHAR(36),
    FriendUserId CHAR(36),
    StartDate DATE,
    FriendshipLevel VARCHAR(50),
    FOREIGN KEY (UserId) REFERENCES User(UserId),
    FOREIGN KEY (FriendUserId) REFERENCES User(UserId)
);

-- Create Event table with UUIDs
```

```sql
CREATE TABLE Event (
    EventID CHAR(36) PRIMARY KEY,
    Date DATE,
    Location VARCHAR(255)
);

-- Create Host table with UUIDs
CREATE TABLE Host (
    UserId CHAR(36),
    EventId CHAR(36),
    PRIMARY KEY (UserId, EventId),
    FOREIGN KEY (UserId) REFERENCES User(UserId),
    FOREIGN KEY (EventId) REFERENCES Event(EventID)
);

-- Create RunSessionData table with UUIDs
CREATE TABLE RunSessionData (
    RunSessionId CHAR(36) PRIMARY KEY,
    UserId CHAR(36),
    SessionDistance FLOAT,
    StartTime DATETIME,
    EndTime DATETIME,
    EventId CHAR(36),
    FOREIGN KEY (UserId) REFERENCES User(UserId),
    FOREIGN KEY (EventId) REFERENCES Event(EventID)
);

-- Create Product table with UUID as primary key
CREATE TABLE Product (
    ProductId CHAR(36) PRIMARY KEY,
    ProductName VARCHAR(255),
    ProductPrice DECIMAL(10, 2)
);

-- Create Purchase table with UUIDs
CREATE TABLE Purchase (
    PurchaseId CHAR(36) PRIMARY KEY,
    PurchasePrice DECIMAL(10, 2),
    Quantity INT,
    UserId CHAR(36),
    ProductId CHAR(36),
    FOREIGN KEY (UserId) REFERENCES User(UserId),
    FOREIGN KEY (ProductId) REFERENCES Product(ProductId)
);
```

In stage 2, we use INT data types for all IDs. But now we want to use uuid instead. So the data type for all ids becomes CHAR(36).

3. Inserting at least 1000 rows in the tables

```
mysql> select count(*) from Event;
+----------+
| count(*) |
+----------+
|      100 |
+----------+
1 row in set (0.02 sec)

mysql> select count(*) from Host;
+----------+
| count(*) |
+----------+
|       98 |
+----------+
1 row in set (0.01 sec)

mysql> select count(*) from MakeFriends;
+----------+
| count(*) |
+----------+
|    11684 |
+----------+
1 row in set (0.01 sec)

mysql> select count(*) from Product;
+----------+
| count(*) |
+----------+
|      101 |
+----------+
1 row in set (0.01 sec)

mysql> select count(*) from Purchase;
+----------+
| count(*) |
+----------+
|      989 |
+----------+
1 row in set (0.00 sec)
```

```
mysql> select count(*) from RunSessionData;
+----------+
| count(*) |
+----------+
|    11850 |
+----------+
1 row in set (0.44 sec)

mysql> select count(*) from User;
+----------+
| count(*) |
+----------+
|     4936 |
+----------+
1 row in set (0.01 sec)
```

We have more than 1000 rows in MakeFriends, RunSessionData and User tables.

Advanced queries with the top 15 rows of query results

1. -- List each user's total spending on products

SELECT u.UserId, u.FirstName, u.LastName, SUM(p.PurchasePrice * p.Quantity) AS TotalSpent
FROM User u
JOIN Purchase p ON u.UserId = p.UserId
GROUP BY u.UserId, u.FirstName, u.LastName
ORDER BY TotalSpent DESC
LIMIT 15;

```
mysql> SELECT u.UserId, u.FirstName, u.LastName, SUM(p.PurchasePrice * p.Quantity) AS TotalSpent
    -> FROM User u
    -> JOIN Purchase p ON u.UserId = p.UserId
    -> GROUP BY u.UserId, u.FirstName, u.LastName
    -> ORDER BY TotalSpent DESC
    -> LIMIT 15;
+--------------------------------------+-----------+----------+------------+
| UserId                               | FirstName | LastName | TotalSpent |
+--------------------------------------+-----------+----------+------------+
| 8aec6cf6-d5ef-4dec-8523-30d27fcd3bbd | Susan     | Thomas   |    7052.00 |
| 8f6ea352-cad9-40af-ae3e-369ac2a9143f | Barbara   | Thompson |    6608.00 |
| 5e33414c-2bce-4003-a3d1-52ae092c9b0a | Jessica   | Thompson |    5572.00 |
| eed7c7b6-fa2a-4863-949e-8eaded5fda24 | James     | Brown    |    5193.00 |
| 661490d3-dfcb-4b67-a38b-dfdc16d59c60 | William   | Thompson |    5037.00 |
| e393ac2e-b9a4-4d16-a987-6a2555582304 | Thomas    | Wilson   |    4890.00 |
| afea9a99-6945-4963-a666-2057019ee5d4 | Nancy     | Harris   |    4870.00 |
| d3768dd0-9b6a-4d4e-bf14-c380a4f90443 | Robert    | White    |    4843.00 |
| 96722595-ee1d-4dda-9eef-42547c61898b | Barbara   | Wilson   |    4692.00 |
| 90550b72-bda5-488d-9458-8c1f2477cca3 | Joseph    | Thomas   |    4620.00 |
| b8d32496-e28c-4860-9156-c30f1d135054 | Patricia  | Jackson  |    4620.00 |
| 2d3b1652-1dc3-4a8e-9e0e-9dc1bbb676d4 | Barbara   | Williams |    4620.00 |
| 3281317f-6e96-42bc-befe-2d9a90592b51 | Patricia  | Anderson |    4420.00 |
| b8583bf7-a895-406f-8cfc-897e2ddeb54e | Nancy     | Garcia   |    4401.00 |
| 714b6270-22c0-490d-866b-d14eab033f7c | Karen     | Thomas   |    4400.00 |
+--------------------------------------+-----------+----------+------------+
15 rows in set (0.01 sec)
```

2. -- Find the distinct genre of products a user buy

SELECT u.UserId, u.FirstName, u.LastName, COUNT(DISTINCT p.ProductId) AS UniqueProductsPurchased
FROM User u
JOIN Purchase p ON u.UserId = p.UserId
GROUP BY u.UserId, u.FirstName, u.LastName
ORDER BY UniqueProductsPurchased DESC
LIMIT 15;

```
mysql> SELECT u.UserId, u.FirstName, u.LastName, COUNT(DISTINCT p.ProductId) AS UniqueProductsPurchased
    -> FROM User u
    -> JOIN Purchase p ON u.UserId = p.UserId
    -> GROUP BY u.UserId, u.FirstName, u.LastName
    -> ORDER BY UniqueProductsPurchased DESC
    -> LIMIT 15;
+--------------------------------------+-----------+----------+------------------------+
| UserId                               | FirstName | LastName | UniqueProductsPurchased |
+--------------------------------------+-----------+----------+------------------------+
| 8aec6cf6-d5ef-4dec-8523-30d27fcd3bbd | Susan     | Thomas   |                      4 |
| 2ca8ef44-cfd8-49ee-aabe-965034682227 | Barbara   | Thompson |                      3 |
| cbf4a765-e186-46d1-8791-a56676b1ca9e | Lisa      | Garcia   |                      3 |
| 5e33414c-2bce-4003-a3d1-52ae092c9b0a | Jessica   | Thompson |                      3 |
| d51f8418-9591-470d-b8cf-6dfd7d6142f9 | Charles   | Brown    |                      3 |
| d90a974b-2030-4d95-8011-b2edaba9c749 | David     | Smith    |                      3 |
| 23b9b047-2bf5-4f19-a9b5-379b1c5346af | Barbara   | Anderson |                      3 |
| eed7c7b6-fa2a-4863-949e-8eaded5fda24 | James     | Brown    |                      3 |
| 93252c2e-a3c9-4f6f-bafe-95b6e3739323 | Nancy     | Smith    |                      3 |
| 7ed859fe-b286-4f60-a25d-c41add6126df | William   | Thompson |                      3 |
| 17cf67e1-589e-426e-bd45-d28d922566b4 | Richard   | Davis    |                      2 |
| 1e911397-a520-488c-a7b3-7b91c237501f | Richard   | Thomas   |                      2 |
| 1ecee8e8-c2bf-4e7e-9650-72e232aaed2f | Robert    | Moore    |                      2 |
| 01951ab4-fe52-4fd9-874b-884c1861649a | Barbara   | White    |                      2 |
| 18ef53c9-248f-4e3f-ab08-b235cd8d5279 | Jessica   | Brown    |                      2 |
+--------------------------------------+-----------+----------+------------------------+
15 rows in set (0.01 sec)
```

3. -- Find top 15 users with the most friends

SELECT u.UserId, u.FirstName, u.LastName, COUNT(mf.FriendUserId) AS FriendCount
FROM User u
JOIN MakeFriends mf ON u.UserId = mf.UserId
GROUP BY u.UserId, u.FirstName, u.LastName
ORDER BY FriendCount DESC
LIMIT 15;

```
mysql> SELECT u.UserId, u.FirstName, u.LastName, COUNT(mf.FriendUserId) AS FriendCount
    -> FROM User u
    -> JOIN MakeFriends mf ON u.UserId = mf.UserId
    -> GROUP BY u.UserId, u.FirstName, u.LastName
    -> ORDER BY FriendCount DESC
    -> LIMIT 15;
+--------------------------------------+-----------+----------+-------------+
| UserId                               | FirstName | LastName | FriendCount |
+--------------------------------------+-----------+----------+-------------+
| 64a1dd7f-a0fc-49be-bea2-04aa0706fc1b | Richard   | Williams |          11 |
| f494c09a-c448-4a95-a769-c43fd10d908f | Richard   | Miller   |           9 |
| 15b3e572-4d43-4bb6-8223-7574535c9a97 | Richard   | Martin   |           9 |
| 033be9a8-50ed-4aa6-b6ff-f227bb89c6b0 | Sarah     | Miller   |           9 |
| 016650ec-5578-49a4-a719-385a21429506 | Sarah     | Harris   |           9 |
| 45e5a445-8861-4c81-ba4f-2c461aa3a6a9 | Joseph    | Brown    |           9 |
| b5e3ebb0-4972-4ee7-9c09-6c3398f2a43f | Thomas    | Robinson |           9 |
| 3ecc36f7-1baa-432f-83ea-99b5655aa623 | David     | Brown    |           8 |
| 79b63201-16e5-482d-a745-91645c701030 | Richard   | Thomas   |           8 |
| 14d34c31-6187-459f-8377-7ff69f3fad5f | Patricia  | Wilson   |           8 |
| 6ca83a07-5164-4633-b138-bb16bb24479a | Robert    | Williams |           8 |
| a65fbd27-09ca-4748-a9f2-65fd221657fe | Sarah     | Williams |           8 |
| 60664d08-265e-40b1-986d-9260cd2c2540 | Elizabeth | Anderson |           8 |
| b55b8919-e147-4af6-91ea-219e800ec84b | Richard   | Jackson  |           8 |
| 5b24e653-ebd0-40e4-bd82-2406ea181722 | John      | Robinson |           8 |
+--------------------------------------+-----------+----------+-------------+
15 rows in set (0.09 sec)
```

4. -- Find the Most Popular Events Based on User Attendance

```
WITH EventAttendance AS (
   SELECT e.EventID, e.Date, e.Location, COUNT(h.UserId) AS AttendanceCount
   FROM Event e
   JOIN Host h ON e.EventID = h.EventId
   GROUP BY e.EventID, e.Date, e.Location
)
SELECT EventID, Date, Location, AttendanceCount
FROM EventAttendance
ORDER BY AttendanceCount DESC
LIMIT 15;
```

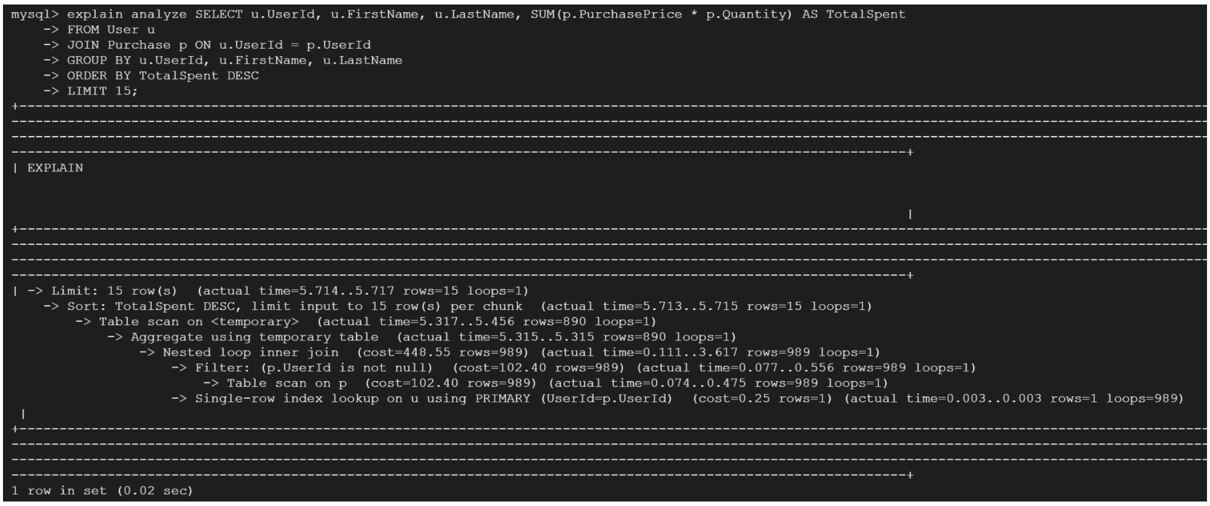
```
mysql> WITH EventAttendance AS (
    ->     SELECT e.EventID, e.Date, e.Location, COUNT(h.UserId) AS AttendanceCount
    ->     FROM Event e
    ->     JOIN Host h ON e.EventID = h.EventId
    ->     GROUP BY e.EventID, e.Date, e.Location
    -> )
    -> SELECT EventID, Date, Location, AttendanceCount
    -> FROM EventAttendance
    -> ORDER BY AttendanceCount DESC
    -> LIMIT 15;
+--------------------------------------+------------+-------------+-----------------+
| EventID                              | Date       | Location    | AttendanceCount |
+--------------------------------------+------------+-------------+-----------------+
| 03a4c737-0867-4dc7-9162-e87c0f281681 | 2024-05-13 | Chicago     |               1 |
| 05b2977c-b520-416f-b9c1-5dd39c999bdd | 2024-01-19 | Chicago     |               1 |
| 092313e9-8917-4a9c-9a23-fe04b62e3b5d | 2024-07-21 | Los Angeles |               1 |
| 0b8c5233-5990-4221-ac7a-8f61d99b2b5c | 2024-09-25 | Chicago     |               1 |
| 0fe6e082-966c-42b6-a421-1f8d07796489 | 2024-10-20 | Los Angeles |               1 |
| 1061ac6a-3b8e-4eae-b655-9193d7fc867f | 2024-05-02 | Los Angeles |               1 |
| 1148f6c8-ffed-4b7d-aa26-4a870f371694 | 2024-02-25 | Boston      |               1 |
| 11ce0304-9cf5-4b21-b706-c5d580ec4805 | 2024-02-15 | Chicago     |               1 |
| 134994e7-be37-4b72-b65b-2bd0dd8f1fb3 | 2024-03-07 | Los Angeles |               1 |
| 16a94d1f-e014-4e8d-882a-ee56f3097828 | 2023-12-28 | Seattle     |               1 |
| 1746663f-8526-4460-b597-d8f007864810 | 2024-09-14 | Urbana      |               1 |
| 19a94af1-abf8-4721-9ea2-a5a0fa79fd70 | 2024-02-22 | Boston      |               1 |
| 1d8c6aa1-758e-47e1-947a-6afa85abdb08 | 2024-02-07 | New York    |               1 |
| 22d07b91-f8a1-4fe2-ae41-06dea870eb07 | 2023-12-24 | New York    |               1 |
| 27906cc4-2760-429b-bf0d-7993f5741d17 | 2024-05-28 | Los Angeles |               1 |
+--------------------------------------+------------+-------------+-----------------+
15 rows in set (0.00 sec)
```

Rubric Part 5

Indexing:

Query 1:

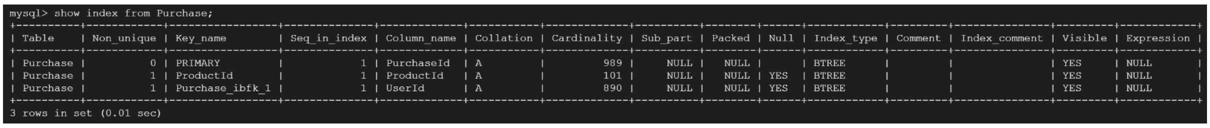
This is the initial performance analyze for query 1:



```
mysql> explain analyze SELECT u.UserId, u.FirstName, u.LastName, SUM(p.PurchasePrice * p.Quantity) AS TotalSpent
    -> FROM User u
    -> JOIN Purchase p ON u.UserId = p.UserId
    -> GROUP BY u.UserId, u.FirstName, u.LastName
    -> ORDER BY TotalSpent DESC
    -> LIMIT 15;
+-----------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------+
| EXPLAIN


                                                                                                                  |
+-----------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=5.714..5.717 rows=15 loops=1)
    -> Sort: TotalSpent DESC, limit input to 15 row(s) per chunk  (actual time=5.713..5.715 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=5.317..5.456 rows=890 loops=1)
            -> Aggregate using temporary table  (actual time=5.315..5.315 rows=890 loops=1)
                -> Nested loop inner join  (cost=448.55 rows=989) (actual time=0.111..3.617 rows=989 loops=1)
                    -> Filter: (p.UserId is not null)  (cost=102.40 rows=989) (actual time=0.077..0.556 rows=989 loops=1)
                        -> Table scan on p  (cost=102.40 rows=989) (actual time=0.074..0.475 rows=989 loops=1)
                    -> Single-row index lookup on u using PRIMARY (UserId=p.UserId)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=989)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------+
1 row in set (0.02 sec)
```

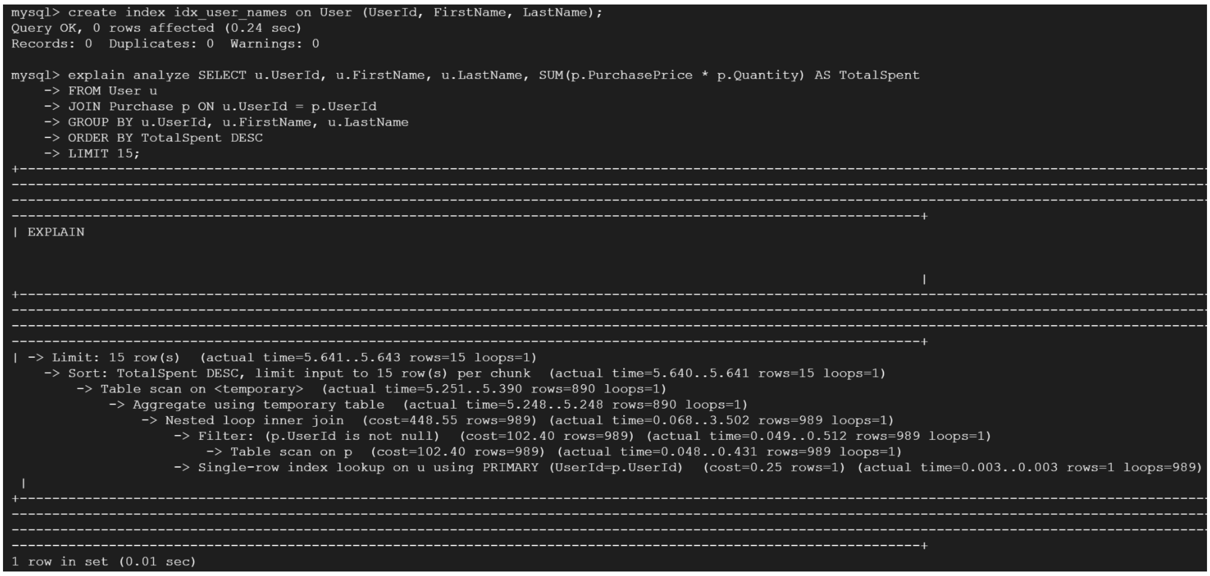
We first want to optimize the cost for join operation. So we add an index for UserId in the Purchase table. However, the cost does not change at all. This might be because UserId is a foreign key to the Purchase table. Therefore it’s already become part of the key, so extra index does not improve the performance.

The following screenshot shows the existing indices before we add an index for UserId in Purchase.



```
mysql> show index from Purchase;
+----------+------------+-----------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table    | Non_unique | Key_name        | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+----------+------------+-----------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Purchase |          0 | PRIMARY         |            1 | PurchaseId  | A         |         989 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Purchase |          1 | ProductId       |            1 | ProductId   | A         |         101 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| Purchase |          1 | Purchase_ibfk_1 |            1 | UserId      | A         |         890 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
+----------+------------+-----------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
3 rows in set (0.01 sec)
```

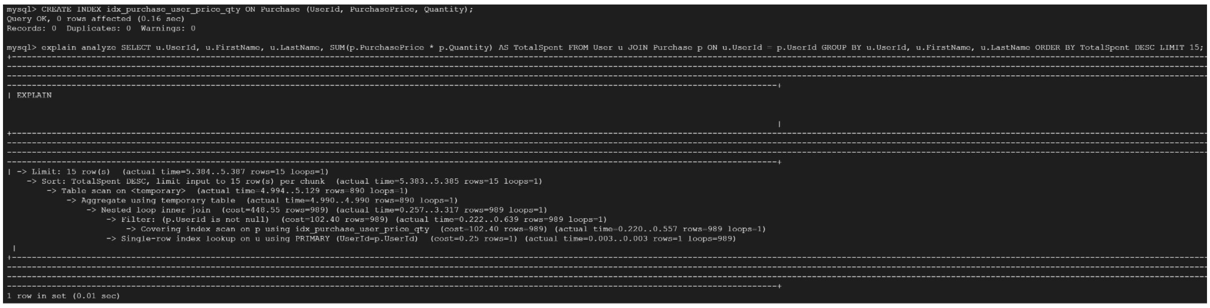
Then we try to optimize the group by. We add an index for UserId, FirstName and LastName in the User table. However, the cost does not change at all. This might be because UserId is the unique identifier for User Table. Therefore adding FirstName and LastName to an index does not offer any additional performance benefit.



```
mysql> create index idx_user_names on User (UserId, FirstName, LastName);
Query OK, 0 rows affected (0.24 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze SELECT u.UserId, u.FirstName, u.LastName, SUM(p.PurchasePrice * p.Quantity) AS TotalSpent
    -> FROM User u
    -> JOIN Purchase p ON u.UserId = p.UserId
    -> GROUP BY u.UserId, u.FirstName, u.LastName
    -> ORDER BY TotalSpent DESC
    -> LIMIT 15;
+-----------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------+
| EXPLAIN


                                                                                                                    |
+-----------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=5.641..5.643 rows=15 loops=1)
    -> Sort: TotalSpent DESC, limit input to 15 row(s) per chunk  (actual time=5.640..5.641 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=5.251..5.390 rows=890 loops=1)
            -> Aggregate using temporary table  (actual time=5.248..5.248 rows=890 loops=1)
                -> Nested loop inner join  (cost=448.55 rows=989) (actual time=0.068..3.502 rows=989 loops=1)
                    -> Filter: (p.UserId is not null)  (cost=102.40 rows=989) (actual time=0.049..0.512 rows=989 loops=1)
                        -> Table scan on p  (cost=102.40 rows=989) (actual time=0.048..0.431 rows=989 loops=1)
                    -> Single-row index lookup on u using PRIMARY (UserId=p.UserId)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=989)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------+
1 row in set (0.01 sec)
```

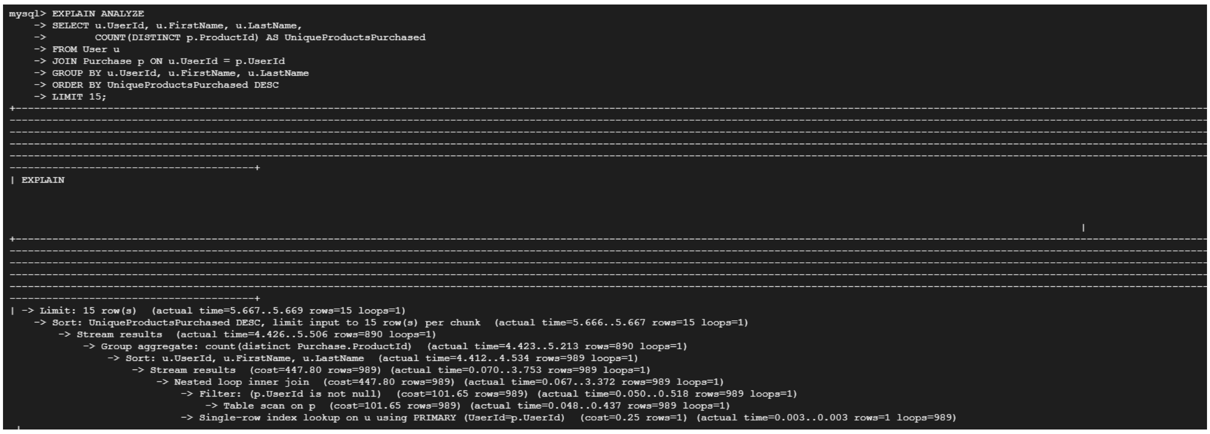
Finally we try to optimize the last aggregation sum(). We add an index for UserId, PurchasePrice and Quantity in the Purchase table. However, the cost does not change at all. This might be because of the same reason as the first attempt. Since

UserId is a foreign key of Purchase and has already become part of the key, the extra index does not improve the performance.

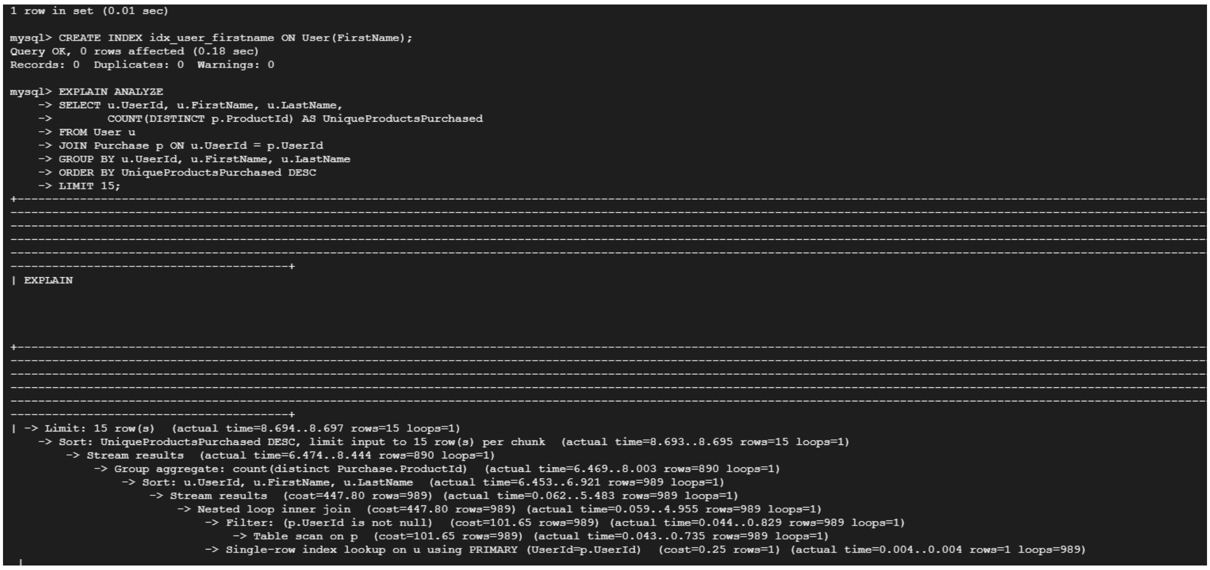


```
mysql> CREATE INDEX idx_purchase_user_price_qty ON Purchase (UserId, PurchasePrice, Quantity);
Query OK, 0 rows affected (0.16 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze SELECT u.UserId, u.FirstName, u.LastName, SUM(p.PurchasePrice * p.Quantity) AS TotalSpent FROM User u JOIN Purchase p ON u.UserId = p.UserId GROUP BY u.UserId, u.FirstName, u.LastName ORDER BY TotalSpent DESC LIMIT 15;
+---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------+
| EXPLAIN

                                                                                                   |
+---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=5.384..5.387 rows=15 loops=1)
    -> Sort: TotalSpent DESC, limit input to 15 row(s) per chunk  (actual time=5.383..5.385 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=4.994..5.129 rows=890 loops=1)
            -> Aggregate using temporary table  (actual time=4.990..4.990 rows=890 loops=1)
                -> Nested loop inner join  (cost=448.55 rows=989) (actual time=0.257..3.317 rows=989 loops=1)
                    -> Filter: (p.UserId is not null)  (cost=102.40 rows=989) (actual time=0.222..0.639 rows=989 loops=1)
                        -> Covering index scan on p using idx_purchase_user_price_qty  (cost=102.40 rows=989) (actual time=0.220..0.557 rows=989 loops=1)
                    -> Single-row index lookup on u using PRIMARY (UserId=p.UserId)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=989)
 |
+---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------+
1 row in set (0.01 sec)
```

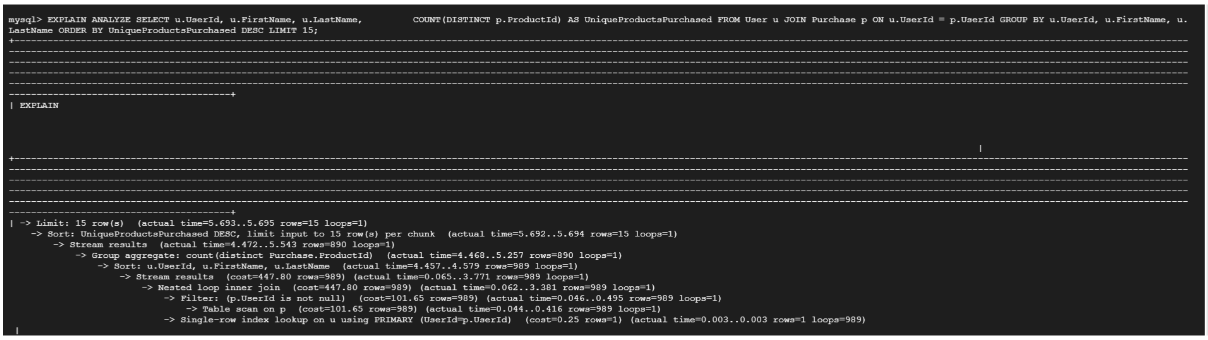
Query2:



```
mysql> EXPLAIN ANALYZE
    -> SELECT u.UserId, u.FirstName, u.LastName,
    ->        COUNT(DISTINCT p.ProductId) AS UniqueProductsPurchased
    -> FROM User u
    -> JOIN Purchase p ON u.UserId = p.UserId
    -> GROUP BY u.UserId, u.FirstName, u.LastName
    -> ORDER BY UniqueProductsPurchased DESC
    -> LIMIT 15;
+---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
--------------------------------------+
| EXPLAIN

                                                                                                          |
+---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
--------------------------------------+
| -> Limit: 15 row(s)  (actual time=5.667..5.669 rows=15 loops=1)
    -> Sort: UniqueProductsPurchased DESC, limit input to 15 row(s) per chunk  (actual time=5.666..5.667 rows=15 loops=1)
        -> Stream results  (actual time=4.426..5.506 rows=890 loops=1)
            -> Group aggregate: count(distinct Purchase.ProductId)  (actual time=4.423..5.213 rows=890 loops=1)
                -> Sort: u.UserId, u.FirstName, u.LastName  (actual time=4.412..4.534 rows=989 loops=1)
                    -> Stream results  (cost=447.80 rows=989) (actual time=0.070..3.753 rows=989 loops=1)
                        -> Nested loop inner join  (cost=447.80 rows=989) (actual time=0.067..3.372 rows=989 loops=1)
                            -> Filter: (p.UserId is not null)  (cost=101.65 rows=989) (actual time=0.050..0.518 rows=989 loops=1)
                                -> Table scan on p  (cost=101.65 rows=989) (actual time=0.048..0.437 rows=989 loops=1)
                            -> Single-row index lookup on u using PRIMARY (UserId=p.UserId)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=989)
```

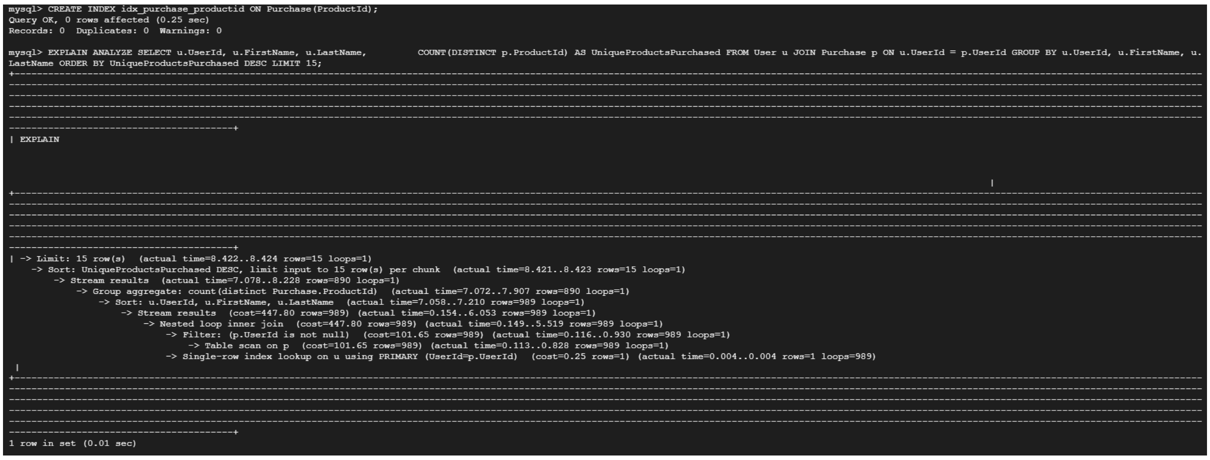


```
1 row in set (0.01 sec)

mysql> CREATE INDEX idx_user_firstname ON User(FirstName);
Query OK, 0 rows affected (0.18 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE
    -> SELECT u.UserId, u.FirstName, u.LastName,
    ->        COUNT(DISTINCT p.ProductId) AS UniqueProductsPurchased
    -> FROM User u
    -> JOIN Purchase p ON u.UserId = p.UserId
    -> GROUP BY u.UserId, u.FirstName, u.LastName
    -> ORDER BY UniqueProductsPurchased DESC
    -> LIMIT 15;
+---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
--------------------------------------+
| EXPLAIN




+---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
--------------------------------------+
| -> Limit: 15 row(s)  (actual time=8.694..8.697 rows=15 loops=1)
    -> Sort: UniqueProductsPurchased DESC, limit input to 15 row(s) per chunk  (actual time=8.693..8.695 rows=15 loops=1)
        -> Stream results  (actual time=6.474..8.444 rows=890 loops=1)
            -> Group aggregate: count(distinct Purchase.ProductId)  (actual time=6.469..8.003 rows=890 loops=1)
                -> Sort: u.UserId, u.FirstName, u.LastName  (actual time=6.453..6.921 rows=989 loops=1)
                    -> Stream results  (cost=447.80 rows=989) (actual time=0.062..5.483 rows=989 loops=1)
                        -> Nested loop inner join  (cost=447.80 rows=989) (actual time=0.059..4.955 rows=989 loops=1)
                            -> Filter: (p.UserId is not null)  (cost=101.65 rows=989) (actual time=0.044..0.829 rows=989 loops=1)
                                -> Table scan on p  (cost=101.65 rows=989) (actual time=0.043..0.735 rows=989 loops=1)
                            -> Single-row index lookup on u using PRIMARY (UserId=p.UserId)  (cost=0.25 rows=1) (actual time=0.004..0.004 rows=1 loops=989)
```



```
mysql> EXPLAIN ANALYZE SELECT u.UserId, u.FirstName, u.LastName,         COUNT(DISTINCT p.ProductId) AS UniqueProductsPurchased FROM User u JOIN Purchase p ON u.UserId = p.UserId GROUP BY u.UserId, u.FirstName, u.
LastName ORDER BY UniqueProductsPurchased DESC LIMIT 15;
+--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
| EXPLAIN



                                                                                                                                          |
+--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
| -> Limit: 15 row(s)  (actual time=5.693..5.695 rows=15 loops=1)
    -> Sort: UniqueProductsPurchased DESC, limit input to 15 row(s) per chunk  (actual time=5.692..5.694 rows=15 loops=1)
        -> Stream results  (actual time=4.472..5.543 rows=890 loops=1)
            -> Group aggregate: count(distinct Purchase.ProductId)  (actual time=4.468..5.257 rows=890 loops=1)
                -> Sort: u.UserId, u.FirstName, u.LastName  (actual time=4.457..4.579 rows=989 loops=1)
                    -> Stream results  (cost=447.80 rows=989) (actual time=0.065..3.771 rows=989 loops=1)
                        -> Nested loop inner join  (cost=447.80 rows=989) (actual time=0.062..3.381 rows=989 loops=1)
                            -> Filter: (p.UserId is not null)  (cost=101.65 rows=989) (actual time=0.046..0.495 rows=989 loops=1)
                                -> Table scan on p  (cost=101.65 rows=989) (actual time=0.044..0.416 rows=989 loops=1)
                            -> Single-row index lookup on u using PRIMARY (UserId=p.UserId)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=989)
 |
```

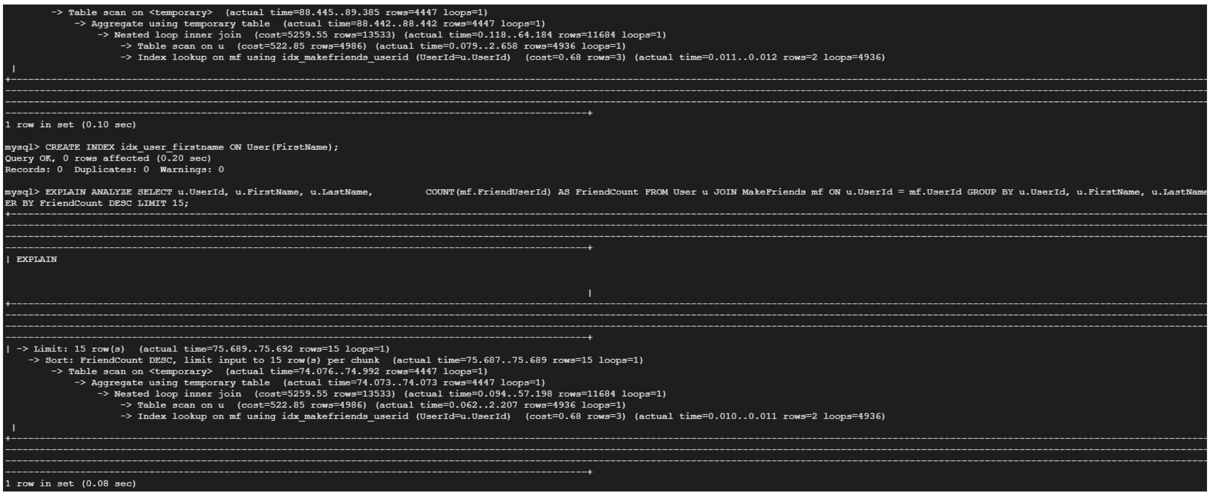
First, we add an index for UserId, but this didn’t change at all. We try to optimize the group by. We add an index for UserId, FirstName and LastName in the User table. However, the cost does not change at all. This might be because UserId is the unique identifier for User Table. Therefore adding FirstName and LastName to an index does not offer any additional performance benefit.



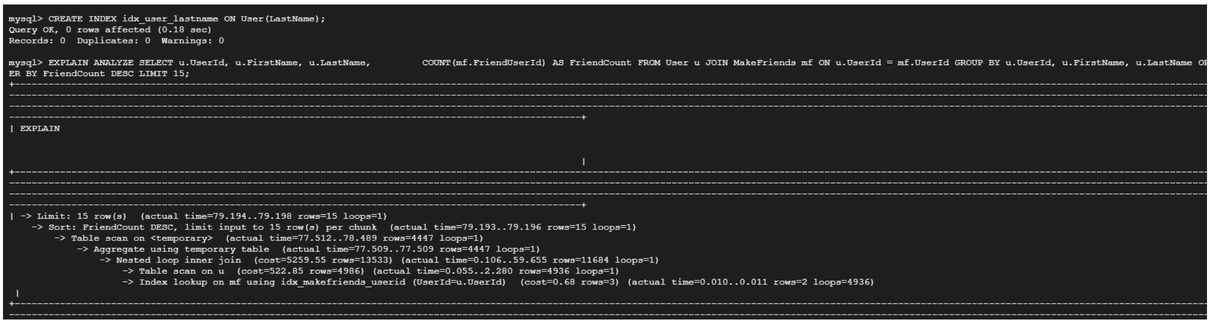
```
mysql> CREATE INDEX idx_purchase_productid ON Purchase(ProductId);
Query OK, 0 rows affected (0.25 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT u.UserId, u.FirstName, u.LastName,         COUNT(DISTINCT p.ProductId) AS UniqueProductsPurchased FROM User u JOIN Purchase p ON u.UserId = p.UserId GROUP BY u.UserId, u.FirstName, u.
LastName ORDER BY UniqueProductsPurchased DESC LIMIT 15;
+--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
| EXPLAIN



                                                                                                                                          |
+--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
| -> Limit: 15 row(s)  (actual time=8.422..8.424 rows=15 loops=1)
    -> Sort: UniqueProductsPurchased DESC, limit input to 15 row(s) per chunk  (actual time=8.421..8.423 rows=15 loops=1)
        -> Stream results  (actual time=7.078..8.228 rows=890 loops=1)
            -> Group aggregate: count(distinct Purchase.ProductId)  (actual time=7.072..7.907 rows=890 loops=1)
                -> Sort: u.UserId, u.FirstName, u.LastName  (actual time=7.058..7.210 rows=989 loops=1)
                    -> Stream results  (cost=447.80 rows=989) (actual time=0.154..6.053 rows=989 loops=1)
                        -> Nested loop inner join  (cost=447.80 rows=989) (actual time=0.149..5.519 rows=989 loops=1)
                            -> Filter: (p.UserId is not null)  (cost=101.65 rows=989) (actual time=0.116..0.930 rows=989 loops=1)
                                -> Table scan on p  (cost=101.65 rows=989) (actual time=0.113..0.828 rows=989 loops=1)
                            -> Single-row index lookup on u using PRIMARY (UserId=p.UserId)  (cost=0.25 rows=1) (actual time=0.004..0.004 rows=1 loops=989)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
1 row in set (0.01 sec)
```

Finally, tried to add index on productId, but still no effect.

Query 3:



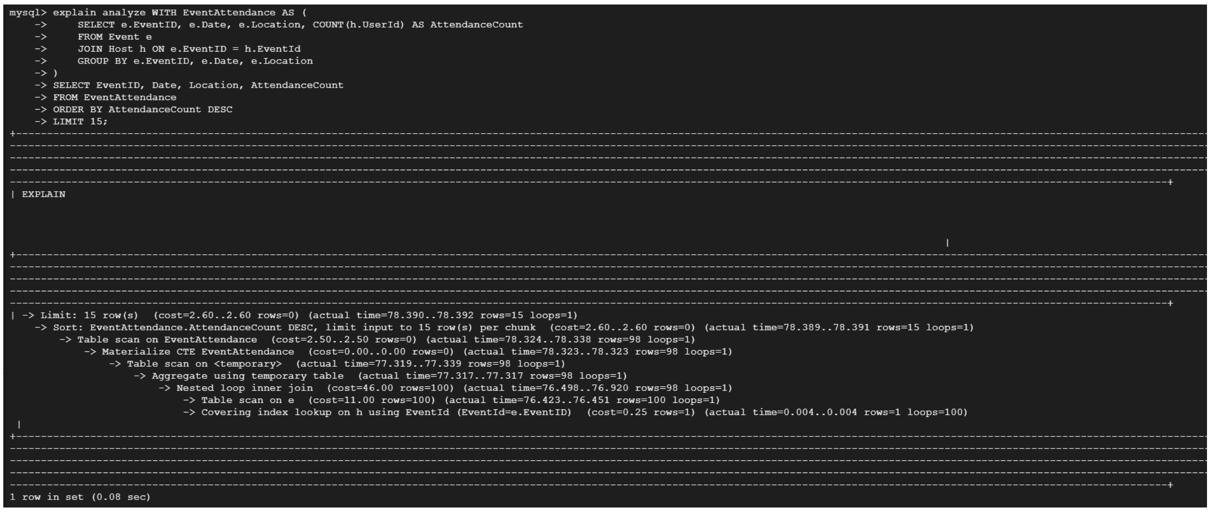
```
        -> Table scan on <temporary>  (actual time=88.445..89.385 rows=4447 loops=1)
            -> Aggregate using temporary table  (actual time=88.442..88.442 rows=4447 loops=1)
                -> Nested loop inner join  (cost=5259.55 rows=13533) (actual time=0.118..64.184 rows=11684 loops=1)
                    -> Table scan on u  (cost=522.85 rows=4986) (actual time=0.079..2.658 rows=4936 loops=1)
                    -> Index lookup on mf using idx_makefriends_userid (UserId=u.UserId)  (cost=0.68 rows=3) (actual time=0.011..0.012 rows=2 loops=4936)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------+
1 row in set (0.10 sec)

mysql> CREATE INDEX idx_user_firstname ON User(FirstName);
Query OK, 0 rows affected (0.20 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT u.UserId, u.FirstName, u.LastName,         COUNT(mf.FriendUserId) AS FriendCount FROM User u JOIN MakeFriends mf ON u.UserId = mf.UserId GROUP BY u.UserId, u.FirstName, u.LastName
ER BY FriendCount DESC LIMIT 15;
+------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------+
| EXPLAIN



                                                                                                     |
+------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=75.689..75.692 rows=15 loops=1)
    -> Sort: FriendCount DESC, limit input to 15 row(s) per chunk  (actual time=75.687..75.689 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=74.076..74.992 rows=4447 loops=1)
            -> Aggregate using temporary table  (actual time=74.073..74.073 rows=4447 loops=1)
                -> Nested loop inner join  (cost=5259.55 rows=13533) (actual time=0.094..57.198 rows=11684 loops=1)
                    -> Table scan on u  (cost=522.85 rows=4986) (actual time=0.062..2.207 rows=4936 loops=1)
                    -> Index lookup on mf using idx_makefriends_userid (UserId=u.UserId)  (cost=0.68 rows=3) (actual time=0.010..0.011 rows=2 loops=4936)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------+
1 row in set (0.08 sec)
```



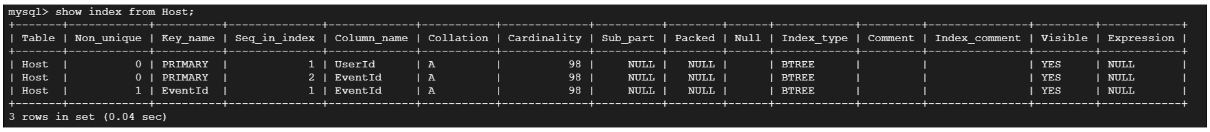
```
mysql> CREATE INDEX idx_user_lastname ON User(LastName);
Query OK, 0 rows affected (0.18 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT u.UserId, u.FirstName, u.LastName,         COUNT(mf.FriendUserId) AS FriendCount FROM User u JOIN MakeFriends mf ON u.UserId = mf.UserId GROUP BY u.UserId, u.FirstName, u.LastName OF
ER BY FriendCount DESC LIMIT 15;
+------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------+
| EXPLAIN



                                                                                                    |
+------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=79.194..79.198 rows=15 loops=1)
    -> Sort: FriendCount DESC, limit input to 15 row(s) per chunk  (actual time=79.193..79.196 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=77.512..78.489 rows=4447 loops=1)
            -> Aggregate using temporary table  (actual time=77.509..77.509 rows=4447 loops=1)
                -> Nested loop inner join  (cost=5259.55 rows=13533) (actual time=0.106..59.655 rows=11684 loops=1)
                    -> Table scan on u  (cost=522.85 rows=4986) (actual time=0.055..2.280 rows=4936 loops=1)
                    -> Index lookup on mf using idx_makefriends_userid (UserId=u.UserId)  (cost=0.68 rows=3) (actual time=0.010..0.011 rows=2 loops=4936)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
```

First, we add an index for UserId, but this didn't change at all. This might be because UserId is the unique identifier for User Table. Therefore adding FirstName and LastName to an index does not offer any additional performance benefit.

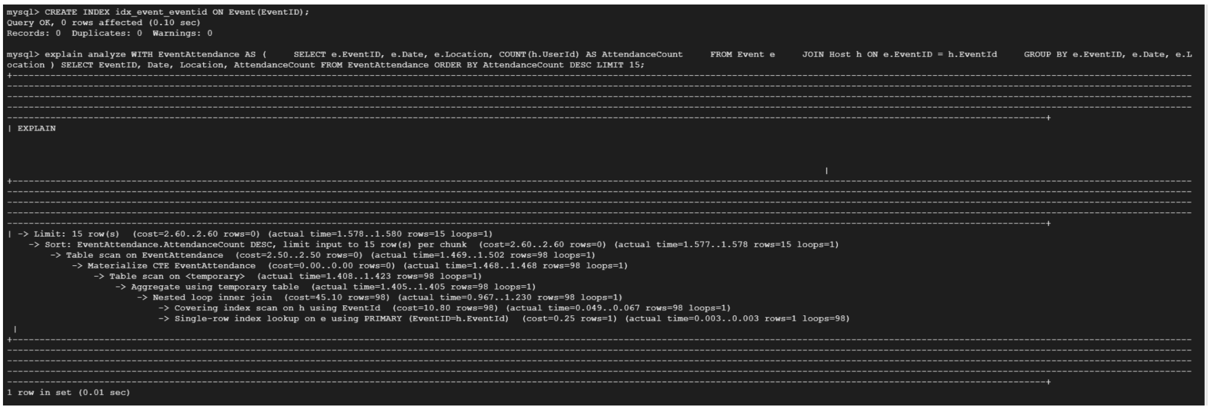
Query 4:



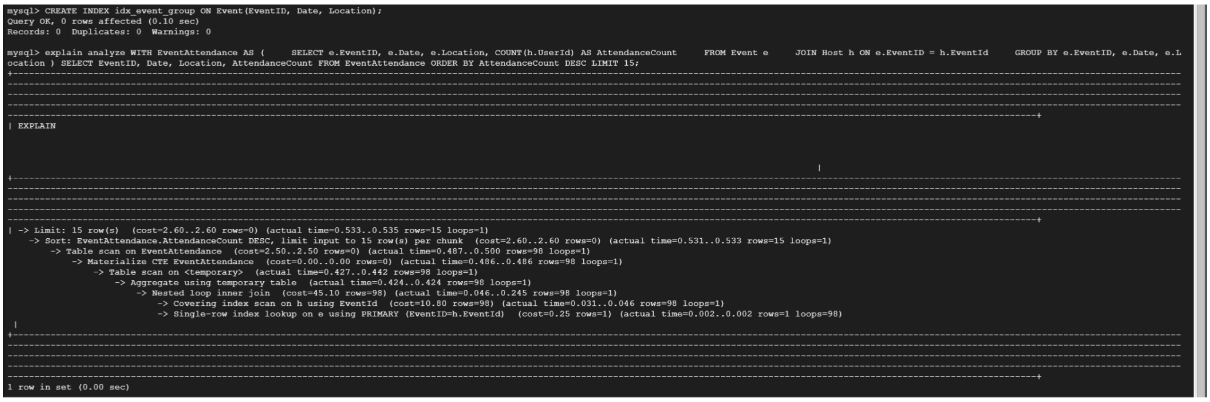
```
mysql> explain analyze WITH EventAttendance AS (
    ->     SELECT e.EventID, e.Date, e.Location, COUNT(h.UserId) AS AttendanceCount
    ->     FROM Event e
    ->     JOIN Host h ON e.EventID = h.EventId
    ->     GROUP BY e.EventID, e.Date, e.Location
    -> )
    -> SELECT EventID, Date, Location, AttendanceCount
    -> FROM EventAttendance
    -> ORDER BY AttendanceCount DESC
    -> LIMIT 15;
+------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN




                                                                                                                                                     |
+------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (cost=2.60..2.60 rows=0) (actual time=78.390..78.392 rows=15 loops=1)
    -> Sort: EventAttendance.AttendanceCount DESC, limit input to 15 row(s) per chunk  (cost=2.60..2.60 rows=0) (actual time=78.389..78.391 rows=15 loops=1)
        -> Table scan on EventAttendance  (cost=2.50..2.50 rows=0) (actual time=78.324..78.338 rows=98 loops=1)
            -> Materialize CTE EventAttendance  (cost=0.00..0.00 rows=0) (actual time=78.323..78.323 rows=98 loops=1)
                -> Table scan on <temporary>  (actual time=77.319..77.339 rows=98 loops=1)
                    -> Aggregate using temporary table  (actual time=77.317..77.317 rows=98 loops=1)
                        -> Nested loop inner join  (cost=46.00 rows=100) (actual time=76.498..76.920 rows=98 loops=1)
                            -> Table scan on e  (cost=11.00 rows=100) (actual time=76.423..76.451 rows=100 loops=1)
                            -> Covering index lookup on h using EventId (EventId=e.EventID)  (cost=0.25 rows=1) (actual time=0.004..0.004 rows=1 loops=100)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.08 sec)
```



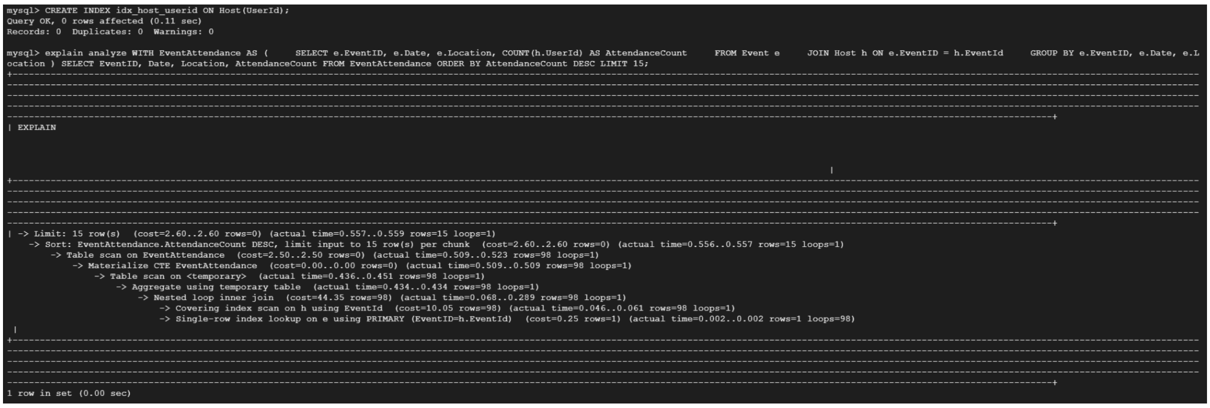
```
mysql> show index from Host;
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Host  |          0 | PRIMARY  |            1 | UserId      | A         |          98 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Host  |          0 | PRIMARY  |            2 | EventId     | A         |          98 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Host  |          1 | EventId  |            1 | EventId     | A         |          98 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
3 rows in set (0.04 sec)
```



```
mysql> CREATE INDEX idx_event_eventid ON Event(EventID);
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze WITH EventAttendance AS (     SELECT e.EventID, e.Date, e.Location, COUNT(h.UserId) AS AttendanceCount     FROM Event e     JOIN Host h ON e.EventID = h.EventId     GROUP BY e.EventID, e.Date, e.L
ocation ) SELECT EventID, Date, Location, AttendanceCount FROM EventAttendance ORDER BY AttendanceCount DESC LIMIT 15;
+------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------+
| EXPLAIN




                                                                                                                    |
+------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (cost=2.60..2.60 rows=0) (actual time=1.578..1.580 rows=15 loops=1)
    -> Sort: EventAttendance.AttendanceCount DESC, limit input to 15 row(s) per chunk  (cost=2.60..2.60 rows=0) (actual time=1.577..1.578 rows=15 loops=1)
        -> Table scan on EventAttendance  (cost=2.50..2.50 rows=0) (actual time=1.469..1.502 rows=98 loops=1)
            -> Materialize CTE EventAttendance  (cost=0.00..0.00 rows=0) (actual time=1.468..1.468 rows=98 loops=1)
                -> Table scan on <temporary>  (actual time=1.408..1.423 rows=98 loops=1)
                    -> Aggregate using temporary table  (actual time=1.405..1.405 rows=98 loops=1)
                        -> Nested loop inner join  (cost=45.10 rows=98) (actual time=0.967..1.230 rows=98 loops=1)
                            -> Covering index scan on h using EventId  (cost=10.80 rows=98) (actual time=0.049..0.067 rows=98 loops=1)
                            -> Single-row index lookup on e using PRIMARY (EventID=h.EventId)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=98)
 |
+------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------+
1 row in set (0.01 sec)
```



```
mysql> CREATE INDEX idx_event_group ON Event(EventID, Date, Location);
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze WITH EventAttendance AS (     SELECT e.EventID, e.Date, e.Location, COUNT(h.UserId) AS AttendanceCount     FROM Event e     JOIN Host h ON e.EventID = h.EventId     GROUP BY e.EventID, e.Date, e.L
ocation ) SELECT EventID, Date, Location, AttendanceCount FROM EventAttendance ORDER BY AttendanceCount DESC LIMIT 15;
+------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------+
| EXPLAIN




                                                                                                                    |
+------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (cost=2.60..2.60 rows=0) (actual time=0.533..0.535 rows=15 loops=1)
    -> Sort: EventAttendance.AttendanceCount DESC, limit input to 15 row(s) per chunk  (cost=2.60..2.60 rows=0) (actual time=0.531..0.533 rows=15 loops=1)
        -> Table scan on EventAttendance  (cost=2.50..2.50 rows=0) (actual time=0.487..0.500 rows=98 loops=1)
            -> Materialize CTE EventAttendance  (cost=0.00..0.00 rows=0) (actual time=0.486..0.486 rows=98 loops=1)
                -> Table scan on <temporary>  (actual time=0.427..0.442 rows=98 loops=1)
                    -> Aggregate using temporary table  (actual time=0.424..0.424 rows=98 loops=1)
                        -> Nested loop inner join  (cost=45.10 rows=98) (actual time=0.046..0.245 rows=98 loops=1)
                            -> Covering index scan on h using EventId  (cost=10.80 rows=98) (actual time=0.031..0.046 rows=98 loops=1)
                            -> Single-row index lookup on e using PRIMARY (EventID=h.EventId)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=98)
 |
+------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```



```
mysql> CREATE INDEX idx_host_userid ON Host(UserId);
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze WITH EventAttendance AS (     SELECT e.EventID, e.Date, e.Location, COUNT(h.UserId) AS AttendanceCount     FROM Event e     JOIN Host h ON e.EventID = h.EventId     GROUP BY e.EventID, e.Date, e.L
ocation ) SELECT EventID, Date, Location, AttendanceCount FROM EventAttendance ORDER BY AttendanceCount DESC LIMIT 15;
+------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------+
| EXPLAIN




                                                                                                                    |
+------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (cost=2.60..2.60 rows=0) (actual time=0.557..0.559 rows=15 loops=1)
    -> Sort: EventAttendance.AttendanceCount DESC, limit input to 15 row(s) per chunk  (cost=2.60..2.60 rows=0) (actual time=0.556..0.557 rows=15 loops=1)
        -> Table scan on EventAttendance  (cost=2.50..2.50 rows=0) (actual time=0.509..0.523 rows=98 loops=1)
            -> Materialize CTE EventAttendance  (cost=0.00..0.00 rows=0) (actual time=0.509..0.509 rows=98 loops=1)
                -> Table scan on <temporary>  (actual time=0.436..0.451 rows=98 loops=1)
                    -> Aggregate using temporary table  (actual time=0.434..0.434 rows=98 loops=1)
                        -> Nested loop inner join  (cost=44.35 rows=98) (actual time=0.068..0.289 rows=98 loops=1)
                            -> Covering index scan on h using EventId  (cost=10.05 rows=98) (actual time=0.046..0.061 rows=98 loops=1)
                            -> Single-row index lookup on e using PRIMARY (EventID=h.EventId)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=98)
 |
+------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

We try to index on GROUP BY, JOIN, and COUNT. The situation is similar to what we have above. First, we added an index on UserId. Next, we tried to optimize the GROUP BY operation by adding a composite index on UserId, FirstName, and LastName in the User table. Finally, we added an index on ProductId, aiming to speed up the JOIN operation between tables. The slight reduction in the table scan cost on table e from 11 to 10.05 indicates that the indexing may have a minor impact on query performance. It suggests that the database optimizer may be slightly benefiting from the new indexes.