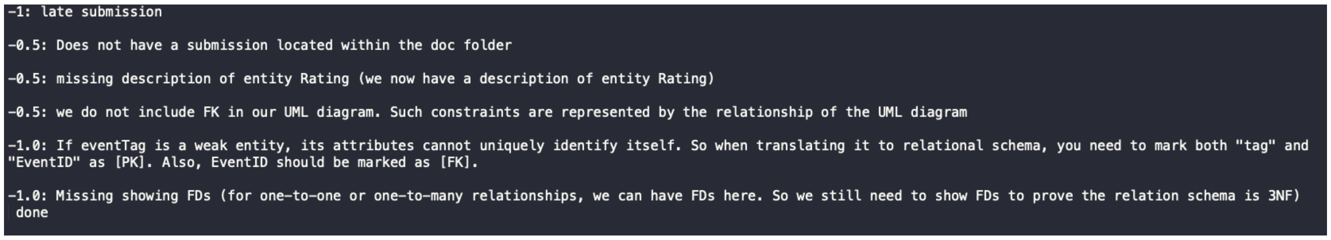
CS 411 PT1 Stage 3

Changes Made to PT2:

Comments made by Peiran:

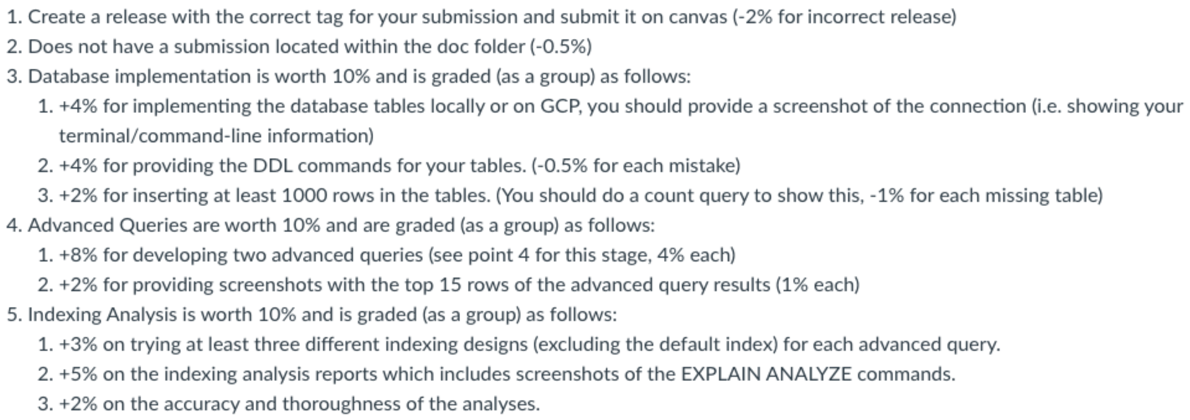


```
-1: late submission

-0.5: Does not have a submission located within the doc folder

-0.5: missing description of entity Rating (we now have a description of entity Rating)

-0.5: we do not include FK in our UML diagram. Such constraints are represented by the relationship of the UML diagram

-1.0: If eventTag is a weak entity, its attributes cannot uniquely identify itself. So when translating it to relational schema, you need to mark both "tag" and
"EventID" as [PK]. Also, EventID should be marked as [FK].

-1.0: Missing showing FDs (for one-to-one or one-to-many relationships, we can have FDs here. So we still need to show FDs to prove the relation schema is 3NF)
 done
```

Changes we made:

- Changed the UML diagram so that foreign keys weren't shown, modified the names for certain attributes in EventTags and Ratings
- Made Tags a primary key in EventTags Table
- Put the submission in the right folder
- Showed the FDs for proving that our relations were in 3NF

PT3 Rubric:

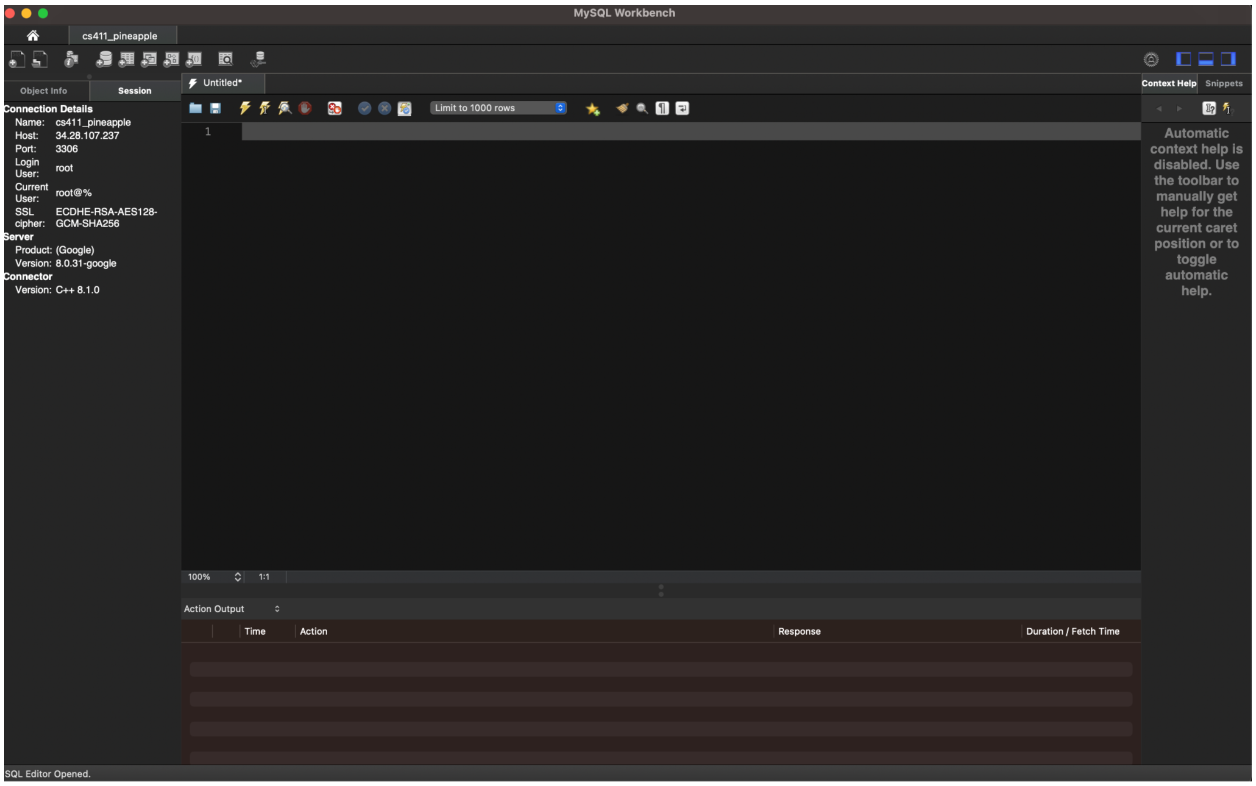


1. Create a release with the correct tag for your submission and submit it on canvas (-2% for incorrect release)
2. Does not have a submission located within the doc folder (-0.5%)
3. Database implementation is worth 10% and is graded (as a group) as follows:
    1. +4% for implementing the database tables locally or on GCP, you should provide a screenshot of the connection (i.e. showing your terminal/command-line information)
    2. +4% for providing the DDL commands for your tables. (-0.5% for each mistake)
    3. +2% for inserting at least 1000 rows in the tables. (You should do a count query to show this, -1% for each missing table)
4. Advanced Queries are worth 10% and are graded (as a group) as follows:
    1. +8% for developing two advanced queries (see point 4 for this stage, 4% each)
    2. +2% for providing screenshots with the top 15 rows of the advanced query results (1% each)
5. Indexing Analysis is worth 10% and is graded (as a group) as follows:
    1. +3% on trying at least three different indexing designs (excluding the default index) for each advanced query.
    2. +5% on the indexing analysis reports which includes screenshots of the EXPLAIN ANALYZE commands.
    3. +2% on the accuracy and thoroughness of the analyses.

Database Implementation:

CS 411 PT1 Stage 3

Connection of Database:



```
Connection Details
  Name:    cs411_pineapple
  Host:    34.28.107.237
  Port:    3306
  Login
  User:    root
  Current
  User:    root@%
  SSL      ECDHE-RSA-AES128-
  cipher:  GCM-SHA256
Server
  Product: (Google)
  Version: 8.0.31-google
Connector
  Version: C++ 8.1.0
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

SQL Editor Opened.

DDL Commands For Tables:

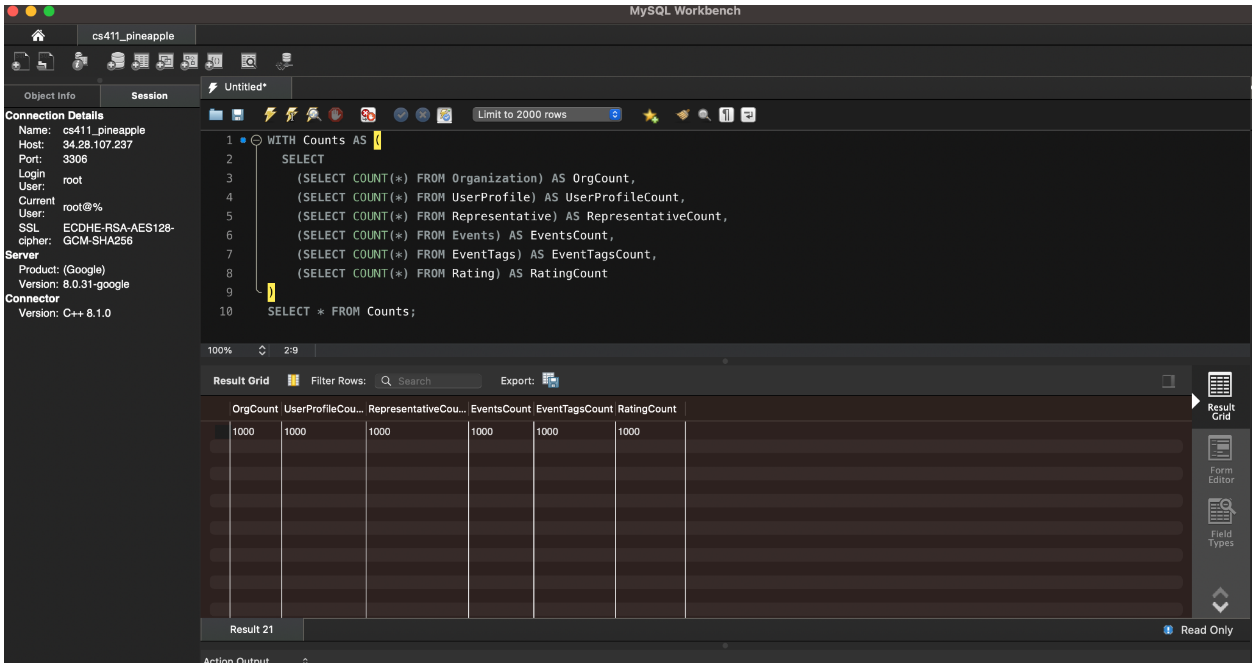
```
CREATE TABLE Organization (
   OrgID INT PRIMARY KEY AUTO_INCREMENT,
   OrgName VARCHAR(255),
   OrgType VARCHAR(55),
   Location VARCHAR(255),
   ContactInfo VARCHAR(100)
);


CREATE TABLE UserProfile (
   UserID INT PRIMARY KEY AUTO_INCREMENT,
   Name VARCHAR(100),
   Preferences VARCHAR(255),
   Contact VARCHAR(100)
);
```

CS 411 PT1 Stage 3

```
CREATE TABLE Events (
   EventID INT PRIMARY KEY AUTO_INCREMENT,
   OrgID INT,
   Location VARCHAR(100),
   FOREIGN KEY (OrgID) REFERENCES Organization(OrgID)
);


CREATE TABLE EventTags (
   EventNum INT,
   Tag VARCHAR(25),
   EventName VARCHAR(30),
   PRIMARY KEY (EventNum, Tag),
   FOREIGN KEY (EventNum) REFERENCES Events(EventID)
);


CREATE TABLE Representative (
   RepID INT PRIMARY KEY AUTO_INCREMENT,
   OrgID INT,
   Name VARCHAR(100),
   Contact VARCHAR(100),
   FOREIGN KEY (OrgID) REFERENCES Organization(OrgID)
);

CREATE TABLE Rating (
   RatingID INT AUTO_INCREMENT,
   EventIdentifier INT,
   UID INT,
   Rating INT,
   Comments VARCHAR(8192),
   PRIMARY KEY (RatingID, EventIdentifier, UID),
   FOREIGN KEY (EventIdentifier) REFERENCES Events(EventID),
   FOREIGN KEY (UID) REFERENCES UserProfile(UserID)
);
```

CS 411 PT1 Stage 3

Insert 1000 Rows in Tables



Connection Details
Name: cs411_pineapple
Host: 34.28.107.237
Port: 3306
Login User: root
Current User: root@%
SSL cipher: ECDHE-RSA-AES128-GCM-SHA256
Server
Product: (Google)
Version: 8.0.31-google
Connector
Version: C++ 8.1.0

```
WITH Counts AS (
    SELECT
        (SELECT COUNT(*) FROM Organization) AS OrgCount,
        (SELECT COUNT(*) FROM UserProfile) AS UserProfileCount,
        (SELECT COUNT(*) FROM Representative) AS RepresentativeCount,
        (SELECT COUNT(*) FROM Events) AS EventsCount,
        (SELECT COUNT(*) FROM EventTags) AS EventTagsCount,
        (SELECT COUNT(*) FROM Rating) AS RatingCount
)
SELECT * FROM Counts;
```

OrgCount	UserProfileCou...	RepresentativeCou...	EventsCount	EventTagsCount	RatingCount
1000	1000	1000	1000	1000	1000

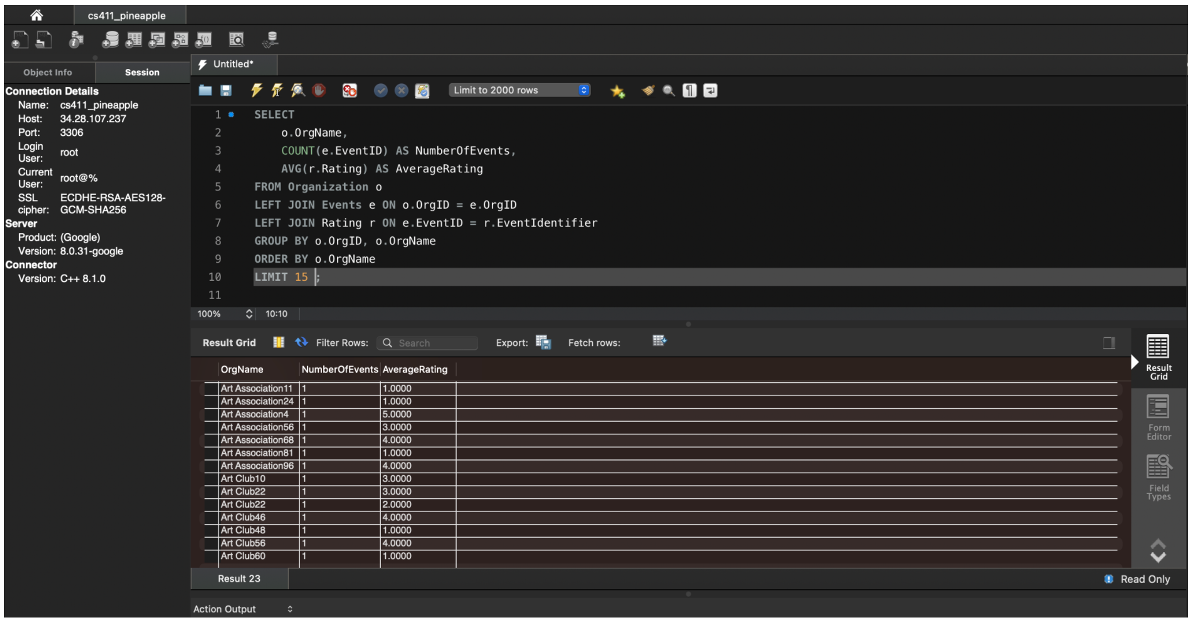
Result 21

Advanced Queries:

CS 411 PT1 Stage 3

1:Lists Organization Names with their highest rating for some event hosted.

```
EXPLAIN ANALYZE
SELECT o.OrgName, sub.MaxRating
FROM Organization o
JOIN (
   SELECT e.OrgID, r.EventIdentifier, MAX(r.Rating) AS MaxRating
   FROM Events e
   JOIN Rating r ON e.EventID = r.EventIdentifier
   GROUP BY e.OrgID, r.EventIdentifier
) AS sub ON o.OrgID = sub.OrgID
ORDER BY o.OrgName, sub.MaxRating DESC
LIMIT 15;
```



Connection Details
Name: cs411_pineapple
Host: 34.28.107.237
Port: 3306
Login User: root
Current User: root@%
SSL cipher: ECDHE-RSA-AES128-GCM-SHA256
Server
Product: (Google)
Version: 8.0.31-google
Connector
Version: C++ 8.1.0

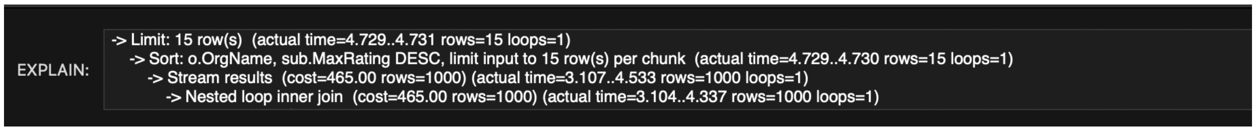
```
SELECT
    o.OrgName,
    COUNT(e.EventID) AS NumberOfEvents,
    AVG(r.Rating) AS AverageRating
FROM Organization o
LEFT JOIN Events e ON o.OrgID = e.OrgID
LEFT JOIN Rating r ON e.EventID = r.EventIdentifier
GROUP BY o.OrgID, o.OrgName
ORDER BY o.OrgName
LIMIT 15 ;
```

100% 10:10

OrgName	NumberOfEvents	AverageRating
Art Association11	1	1.0000
Art Association24	1	1.0000
Art Association4	1	5.0000
Art Association56	1	3.0000
Art Association68	1	4.0000
Art Association81	1	1.0000
Art Association96	1	4.0000
Art Club10	1	3.0000
Art Club22	1	3.0000
Art Club22	1	2.0000
Art Club46	1	4.0000
Art Club48	1	1.0000
Art Club56	1	4.0000
Art Club60	1	1.0000

Indexing:

Original Query Stats:

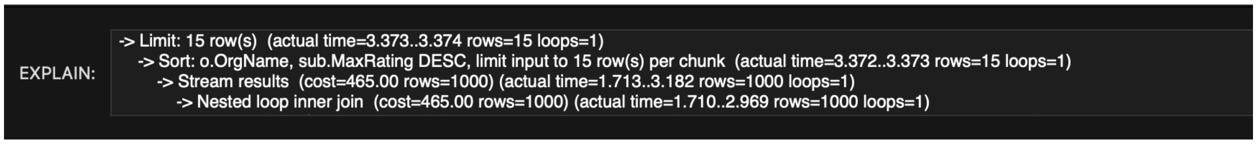


```
EXPLAIN:  -> Limit: 15 row(s)  (actual time=4.729..4.731 rows=15 loops=1)
              -> Sort: o.OrgName, sub.MaxRating DESC, limit input to 15 row(s) per chunk  (actual time=4.729..4.730 rows=15 loops=1)
                  -> Stream results  (cost=465.00 rows=1000) (actual time=3.107..4.533 rows=1000 loops=1)
                      -> Nested loop inner join  (cost=465.00 rows=1000) (actual time=3.104..4.337 rows=1000 loops=1)
```

Indexing Approaches:

1. Composite Index on Events(EventID, OrgID);

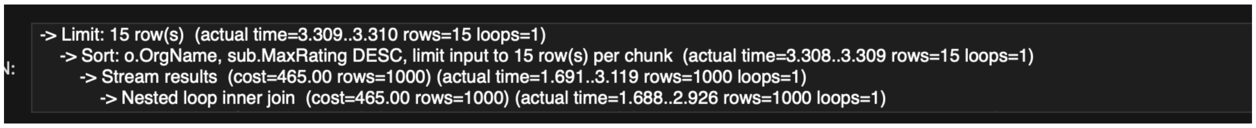
CREATE INDEX idx_events_eventID_orgID ON Events(EventID, OrgID);



```
EXPLAIN:  -> Limit: 15 row(s)  (actual time=3.373..3.374 rows=15 loops=1)
              -> Sort: o.OrgName, sub.MaxRating DESC, limit input to 15 row(s) per chunk  (actual time=3.372..3.373 rows=15 loops=1)
                  -> Stream results  (cost=465.00 rows=1000) (actual time=1.713..3.182 rows=1000 loops=1)
                      -> Nested loop inner join  (cost=465.00 rows=1000) (actual time=1.710..2.969 rows=1000 loops=1)
```

2. Composite Index on Rating(EventIdentifier, Rating);

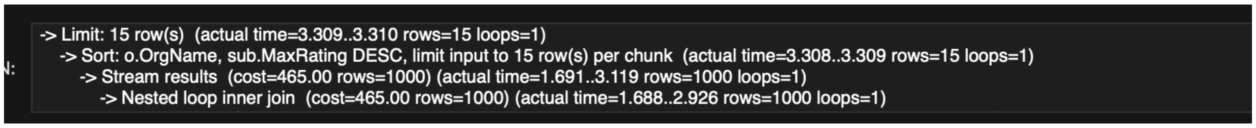
CREATE INDEX idx_rating_eventIdentifier_rating ON Rating(EventIdentifier, Rating);



```
N:  -> Limit: 15 row(s)  (actual time=3.309..3.310 rows=15 loops=1)
        -> Sort: o.OrgName, sub.MaxRating DESC, limit input to 15 row(s) per chunk  (actual time=3.308..3.309 rows=15 loops=1)
            -> Stream results  (cost=465.00 rows=1000) (actual time=1.691..3.119 rows=1000 loops=1)
                -> Nested loop inner join  (cost=465.00 rows=1000) (actual time=1.688..2.926 rows=1000 loops=1)
```

3. Composite Index on Organization(OrgName, OrgID);

CREATE INDEX idx_organization_orgname_orgID ON Organization(OrgName, OrgID);

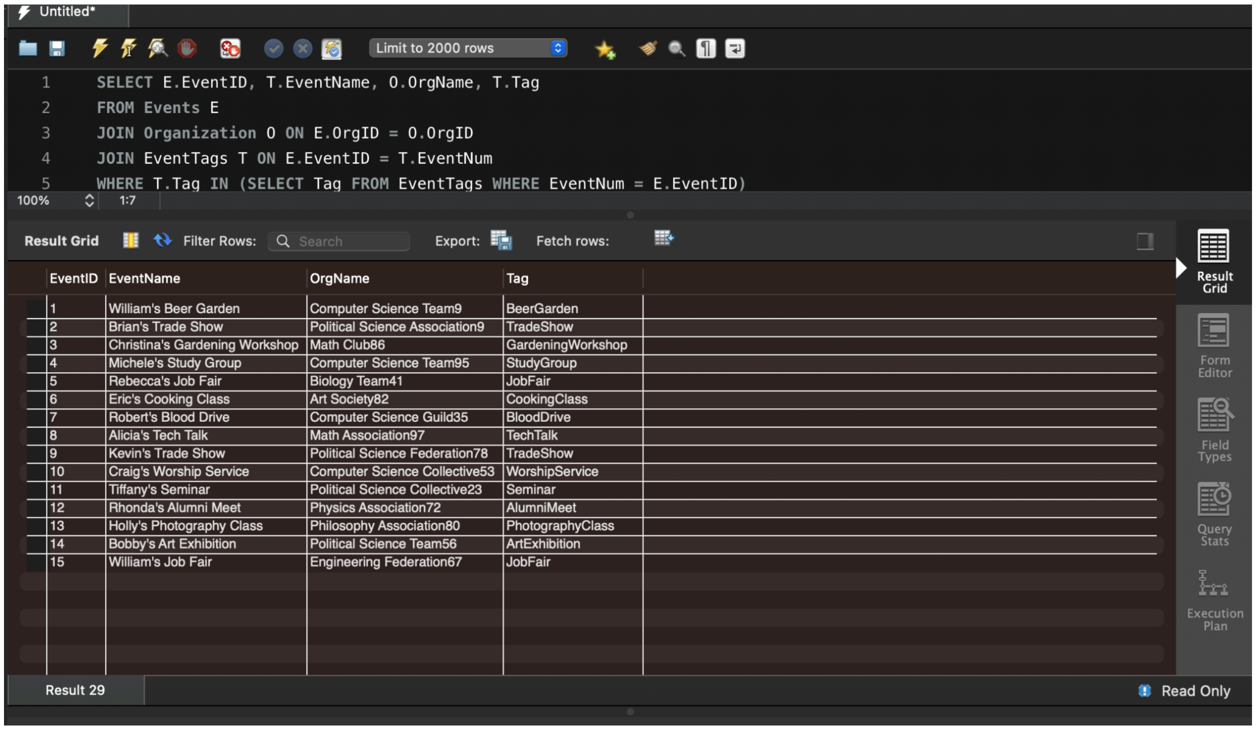


```
N:  -> Limit: 15 row(s)  (actual time=3.309..3.310 rows=15 loops=1)
        -> Sort: o.OrgName, sub.MaxRating DESC, limit input to 15 row(s) per chunk  (actual time=3.308..3.309 rows=15 loops=1)
            -> Stream results  (cost=465.00 rows=1000) (actual time=1.691..3.119 rows=1000 loops=1)
                -> Nested loop inner join  (cost=465.00 rows=1000) (actual time=1.688..2.926 rows=1000 loops=1)
```

In all the indexes we noticed no change/ improvement in the cost. This might be a result of having a small dataset or maybe the database might store it in RAM, making queries inherently fast, irrespective of the applied indexing. It could also be because the database may already have well-optimized indexes in place, which means the new indexes don't provide added benefits. It's also possible that the overhead of using the index, both in terms of computation and data storage, negates the performance improvements. Additionally, it could be that the database might decide not to use the newly created index, especially if it deems other existing indexes or strategies more efficient.

2: Retrieve information about events, their associated organization names, and event tags. (Join & Subquery)

```
SELECT E.EventID, T.EventName, O.OrgName, T.Tag
FROM Events E
JOIN Organization O ON E.OrgID = O.OrgID
JOIN EventTags T ON E.EventID = T.EventNum
WHERE T.Tag IN (SELECT Tag FROM EventTags WHERE EventNum = E.EventID)
LIMIT 15;
```



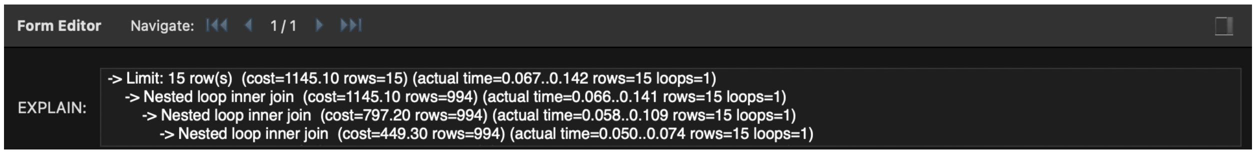
```
SELECT E.EventID, T.EventName, O.OrgName, T.Tag
FROM Events E
JOIN Organization O ON E.OrgID = O.OrgID
JOIN EventTags T ON E.EventID = T.EventNum
WHERE T.Tag IN (SELECT Tag FROM EventTags WHERE EventNum = E.EventID)
```

EventID	EventName	OrgName	Tag
1	William's Beer Garden	Computer Science Team9	BeerGarden
2	Brian's Trade Show	Political Science Association9	TradeShow
3	Christina's Gardening Workshop	Math Club86	GardeningWorkshop
4	Michele's Study Group	Computer Science Team95	StudyGroup
5	Rebecca's Job Fair	Biology Team41	JobFair
6	Eric's Cooking Class	Art Society82	CookingClass
7	Robert's Blood Drive	Computer Science Guild35	BloodDrive
8	Alicia's Tech Talk	Math Association97	TechTalk
9	Kevin's Trade Show	Political Science Federation78	TradeShow
10	Craig's Worship Service	Computer Science Collective53	WorshipService
11	Tiffany's Seminar	Political Science Collective23	Seminar
12	Rhonda's Alumni Meet	Physics Association72	AlumniMeet
13	Holly's Photography Class	Philosophy Association80	PhotographyClass
14	Bobby's Art Exhibition	Political Science Team56	ArtExhibition
15	William's Job Fair	Engineering Federation67	JobFair

Result 29

Indexing

Original Query Stats:

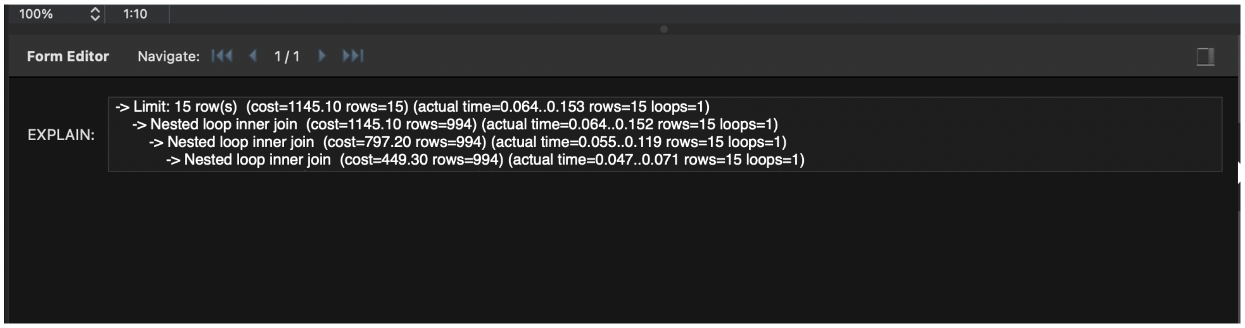


```
EXPLAIN: -> Limit: 15 row(s)  (cost=1145.10 rows=15) (actual time=0.067..0.142 rows=15 loops=1)
           -> Nested loop inner join  (cost=1145.10 rows=994) (actual time=0.066..0.141 rows=15 loops=1)
              -> Nested loop inner join  (cost=797.20 rows=994) (actual time=0.058..0.109 rows=15 loops=1)
                 -> Nested loop inner join  (cost=449.30 rows=994) (actual time=0.050..0.074 rows=15 loops=1)
```

Indexing Approaches:

4. Composite Index on (EventTags.EventNum, EventTags.Tag):

CREATE INDEX idx_EventTags_EventNum_Tag ON EventTags (EventNum, Tag);

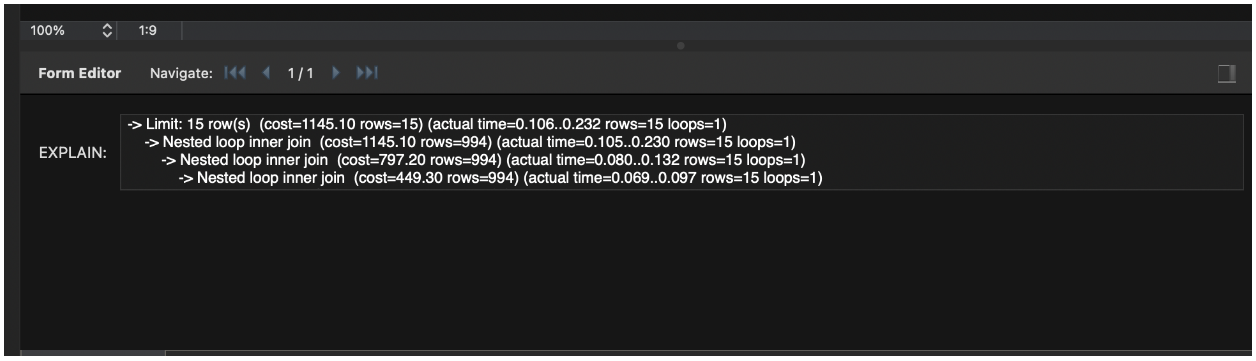


```
-> Limit: 15 row(s)  (cost=1145.10 rows=15) (actual time=0.064..0.153 rows=15 loops=1)
   -> Nested loop inner join  (cost=1145.10 rows=994) (actual time=0.064..0.152 rows=15 loops=1)
      -> Nested loop inner join  (cost=797.20 rows=994) (actual time=0.055..0.119 rows=15 loops=1)
         -> Nested loop inner join  (cost=449.30 rows=994) (actual time=0.047..0.071 rows=15 loops=1)
```

Unfortunately, there is no change in the cost. The index may not be selective due to skewed data distribution or limited dataset size (the former is more likely than the latter). The query optimizer's plan and other existing indexes can also influence index usage.

5. Index on EventTags.Tag:

CREATE INDEX idx_EventTags_Tag ON EventTags (Tag);



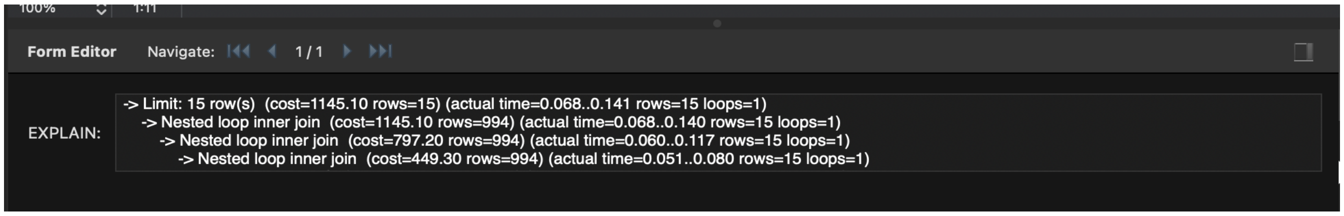
```
-> Limit: 15 row(s)  (cost=1145.10 rows=15) (actual time=0.106..0.232 rows=15 loops=1)
    -> Nested loop inner join  (cost=1145.10 rows=994) (actual time=0.105..0.230 rows=15 loops=1)
        -> Nested loop inner join  (cost=797.20 rows=994) (actual time=0.080..0.132 rows=15 loops=1)
            -> Nested loop inner join  (cost=449.30 rows=994) (actual time=0.069..0.097 rows=15 loops=1)
```

There is no improvement after creating an index on EventTags.Tag, which may occur due to factors such as low selectivity in the Tag column, the query's overall complexity, and the query optimizer's choice of execution plan. We think it's possible that the index is not being utilized by the optimizer.

6. Two separate indices on different attributes in 2 different tables

CREATE INDEX eventID_idx_1 ON Events (EventID);

CREATE INDEX eventTags_idx_1 ON EventTags (EventNum);



```
-> Limit: 15 row(s)  (cost=1145.10 rows=15) (actual time=0.068..0.141 rows=15 loops=1)
    -> Nested loop inner join  (cost=1145.10 rows=994) (actual time=0.068..0.140 rows=15 loops=1)
        -> Nested loop inner join  (cost=797.20 rows=994) (actual time=0.060..0.117 rows=15 loops=1)
            -> Nested loop inner join  (cost=449.30 rows=994) (actual time=0.051..0.080 rows=15 loops=1)
```

There was still no change in the performance .This may once again be due to the database optimizer's query execution plan. The optimizer may have chosen a different plan that doesn't effectively utilize the newly created index.