CS 411 Project Track 1 Stage 3

Group Number: 096

GroupName: OpenSource

Each of our 5 entity tables have at least 1000 rows:

```
mysql> show tables;
+-----------------------+
| Tables_in_opensource |
+-----------------------+
| author                |
| music                 |
| topic                 |
| topic_work            |
| university            |
| work                  |
| work_author           |
+-----------------------+
7 rows in set (0.01 sec)
```

```
mysql> select count(*) from author;
+----------+
| count(*) |
+----------+
|    36996 |
+----------+
1 row in set (0.46 sec)

mysql> select count(*) from topic;
+----------+
| count(*) |
+----------+
|     2892 |
+----------+
1 row in set (0.24 sec)

mysql> select count(*) from university;
+----------+
| count(*) |
+----------+
|    21607 |
+----------+
1 row in set (0.22 sec)
```

```
mysql> select count(*) from work;
+----------+
| count(*) |
+----------+
|     9969 |
+----------+
1 row in set (0.03 sec)

mysql> select count(*) from music;
+----------+
| count(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.05 sec)
```

```
mysql> select count(*) from topic_work;
+----------+
| count(*) |
+----------+
|    24830 |
+----------+
1 row in set (0.49 sec)
```

```
mysql> select count(*) from work author;
+----------+
| count(*) |
+----------+
|     9969 |
+----------+
1 row in set (0.01 sec)
```

For these queries, the hard coded values would be input from the user (or parsed data from the user).

Query 1: Shows how many works are released each year for medicine.

```
SELECT work.publication_year,
       Count(work.work_id)
FROM   work
       JOIN topic_work
         ON topic_work.work_id = work.work_id
       JOIN topic
         ON topic_work.topic_id = topic.topic_id
WHERE  topic.category = 'Medicine'
GROUP  BY work.publication_year
ORDER  BY work.publication_year ASC;
```

publication_year	COUNT(work.work_id)
1889	1
1907	1
1909	1
1938	1
1949	5
1950	1
1952	1
1954	1
1955	1
1956	5
1957	4
1958	2
1959	6
1960	1
1961	4

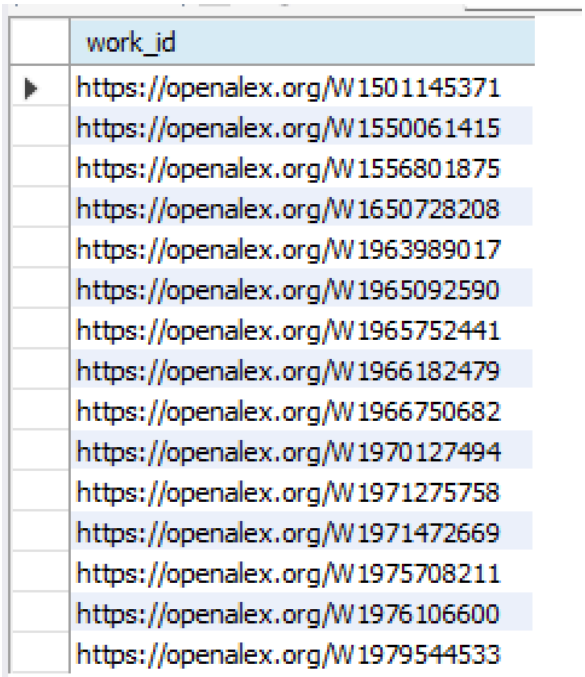
Query 2: Shows works published by universities other than the one selected

```
SELECT DISTINCT work.work_id,
                author.university_name
FROM   work
       JOIN work_author
         ON work.work_id = work_author.work_id
       JOIN author
         ON author.author_id = work_author.author_id
       JOIN university
         ON university.institution_name = author.university_name
WHERE  university.country = 'United States'
EXCEPT
SELECT DISTINCT work.work_id,
                author.university_name
FROM   work
       JOIN work_author
         ON work.work_id = work_author.work_id
       JOIN author
         ON author.author_id = work_author.author_id
WHERE  author.university_name = 'Cornell University';
```

work_id	university_name
https://openalex.org/W1518101824	Harvard University
https://openalex.org/W1513425511	Harvard University
https://openalex.org/W1487557448	Harvard University
https://openalex.org/W1004584821	Harvard University
https://openalex.org/W1514621373	Harvard University Press
https://openalex.org/W1509292812	Harvard University Press
https://openalex.org/W1255676896	Harvard University Press
https://openalex.org/W1533942137	Broad Institute
https://openalex.org/W1255676896	Broad Institute
https://openalex.org/W1533942137	Massachusetts General Hospital
https://openalex.org/W1532325895	Stanford University
https://openalex.org/W1527927437	Stanford University
https://openalex.org/W1509562192	Stanford University
https://openalex.org/W1504303645	Johns Hopkins University
https://openalex.org/W1494198834	Johns Hopkins University

Query 3: This will get all works related to the given paper.

```
SELECT work_outer.work_id
FROM   work work_outer
JOIN   topic topic_outer
JOIN   topic_work tw_outer
ON     work_outer.work_id = tw_outer.work_id
AND    topic_outer.topic_id = tw_outer.topic_id
WHERE  work_outer.work_id <> 'https://openalex.org/works/W157808733'
AND    topic_outer.category IN
       (
              SELECT topic.category
              FROM   work
              JOIN   topic_work
              ON     topic_work.work_id = work.work_id
              JOIN   topic
              ON     topic_work.topic_id = topic.topic_id
              WHERE  work.work_id LIKE '%W950821216%' );
```



work_id
https://openalex.org/W1501145371
https://openalex.org/W1550061415
https://openalex.org/W1556801875
https://openalex.org/W1650728208
https://openalex.org/W1963989017
https://openalex.org/W1965092590
https://openalex.org/W1965752441
https://openalex.org/W1966182479
https://openalex.org/W1966750682
https://openalex.org/W1970127494
https://openalex.org/W1971275758
https://openalex.org/W1971472669
https://openalex.org/W1975708211
https://openalex.org/W1976106600
https://openalex.org/W1979544533

Query 4: This returns how many publications each university has made

```
SELECT   Count(work.work_id),
         author.university_name
FROM     work
JOIN     author
JOIN     work_author
ON       work_author.work_id = work.work_id
AND      work_author.author_id = author.author_id
WHERE    author.university_name <> 'Unknown'
GROUP BY author.university_name
ORDER BY Count(work.work_id) DESC;
```

count(work.work_id)	university_name
494	Stanford University
468	Harvard University
411	Broad Institute
384	Massachusetts Institute of Technology
342	University of Oxford
339	Washington University in St. Louis
338	University of Washington
326	Institute for Health Metrics and Evaluation
313	Wellcome Sanger Institute
309	University of California, Berkeley
282	National Institutes of Health
272	Google (United States)
252	University of Michigan–Ann Arbor
250	University of Cambridge
221	Johns Hopkins University

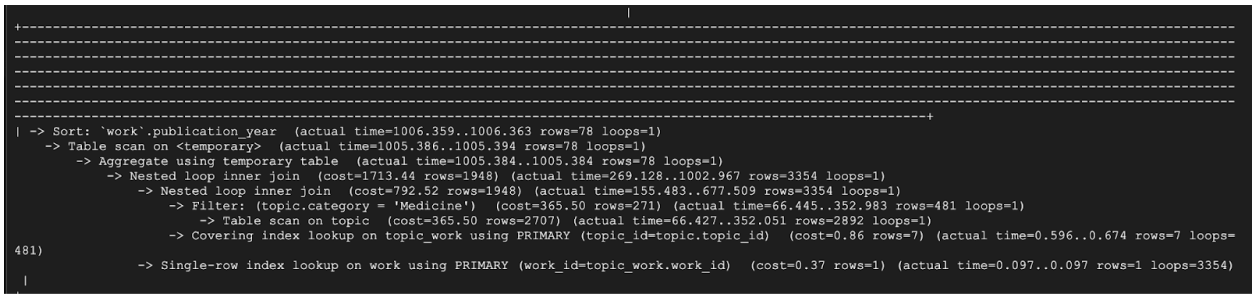
DDL Commands

```
create table author (
    author_id varchar(255),
    author_name varchar(255),
    university_name varchar(255),
    PRIMARY KEY (author_id),
    FOREIGN KEY (university_name)
                references university(InstitutionId));

create table university (
    InstitutionId int,
    Institution_Name varchar(255),
    City varchar(255),
    Country varchar(255),
    PRIMARY KEY (InstitutionId));

create table work (
    work_id varchar(255),
    title varchar(1000),
    publication_year int,
    primary key (work_id));

create table topic (
    topic_id varchar(255),
    topic_name varchar(255),
    category varchar(255),
    primary key (topic_id));

create table music (
    MusicID int,
    filename varchar(255),
    genre  varchar(255),
    Primary key (MusicID));

create table topic_work (
    topic_id varchar(255),
    work_id varchar(255),
    PRIMARY KEY (topic_id, work_id),
    FOREIGN KEY (topic_id) REFERENCES topic(topic_id)
    FOREIGN KEY (work_id) REFERENCES work(work_id));
```

```
create table work_author (
    work_id varchar(255),
    author_id varchar(255),
    PRIMARY KEY (work_id, author_id),
    FOREIGN KEY (work_id) REFERENCES work(work_id)
    FOREIGN KEY (author_id) REFERENCES authors(author_id));

create table topic_music (
    category varchar(255),
    genre varchar(255),
    PRIMARY KEY (category),
    FOREIGN KEY (category) REFERENCES topic(category)
    FOREIGN KEY (genre) REFERENCES music(genre));
```

Indexing

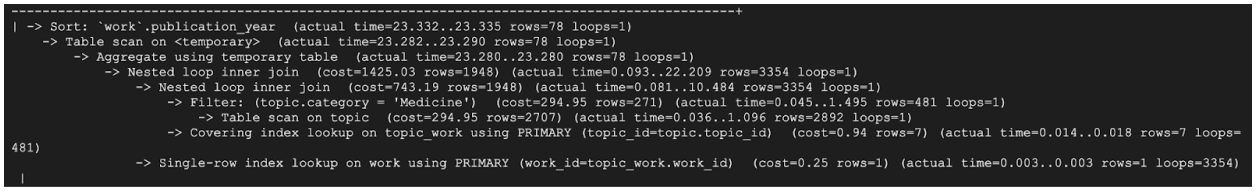
Query 1 without any indexing: Cost:



```
| -> Sort: `work`.publication_year  (actual time=1006.359..1006.363 rows=78 loops=1)
    -> Table scan on <temporary>  (actual time=1005.386..1005.394 rows=78 loops=1)
        -> Aggregate using temporary table  (actual time=1005.384..1005.384 rows=78 loops=1)
            -> Nested loop inner join  (cost=1713.44 rows=1948) (actual time=269.128..1002.967 rows=3354 loops=1)
                -> Nested loop inner join  (cost=792.52 rows=1948) (actual time=155.483..677.509 rows=3354 loops=1)
                    -> Filter: (topic.category = 'Medicine')  (cost=365.50 rows=271) (actual time=66.445..352.983 rows=481 loops=1)
                        -> Table scan on topic  (cost=365.50 rows=2707) (actual time=66.427..352.051 rows=2892 loops=1)
                    -> Covering index lookup on topic_work using PRIMARY (topic_id=topic.topic_id)  (cost=0.86 rows=7) (actual time=0.596..0.674 rows=7 loops=
481)
                -> Single-row index lookup on work using PRIMARY (work_id=topic_work.work_id)  (cost=0.37 rows=1) (actual time=0.097..0.097 rows=1 loops=3354)
```

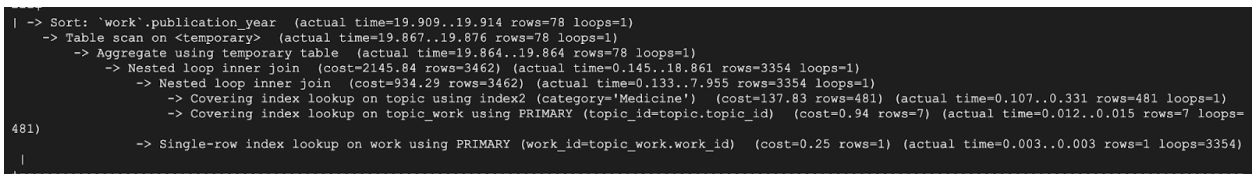
`Config 1`

With index on work.publication_year:



```
| -> Sort: `work`.publication_year  (actual time=23.332..23.335 rows=78 loops=1)
    -> Table scan on <temporary>  (actual time=23.282..23.290 rows=78 loops=1)
        -> Aggregate using temporary table  (actual time=23.280..23.280 rows=78 loops=1)
            -> Nested loop inner join  (cost=1425.03 rows=1948) (actual time=0.093..22.209 rows=3354 loops=1)
                -> Nested loop inner join  (cost=743.19 rows=1948) (actual time=0.081..10.484 rows=3354 loops=1)
                    -> Filter: (topic.category = 'Medicine')  (cost=294.95 rows=271) (actual time=0.045..1.495 rows=481 loops=1)
                        -> Table scan on topic  (cost=294.95 rows=2707) (actual time=0.036..1.096 rows=2892 loops=1)
                    -> Covering index lookup on topic_work using PRIMARY (topic_id=topic.topic_id)  (cost=0.94 rows=7) (actual time=0.014..0.018 rows=7 loops=
481)
                -> Single-row index lookup on work using PRIMARY (work_id=topic_work.work_id)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=3354)
```

With index on topic.category and work.publication_year



```
| -> Sort: `work`.publication_year  (actual time=19.909..19.914 rows=78 loops=1)
    -> Table scan on <temporary>  (actual time=19.867..19.876 rows=78 loops=1)
        -> Aggregate using temporary table  (actual time=19.864..19.864 rows=78 loops=1)
            -> Nested loop inner join  (cost=2145.84 rows=3462) (actual time=0.145..18.861 rows=3354 loops=1)
                -> Nested loop inner join  (cost=934.29 rows=3462) (actual time=0.133..7.955 rows=3354 loops=1)
                    -> Covering index lookup on topic using index2 (category='Medicine')  (cost=137.83 rows=481) (actual time=0.107..0.331 rows=481 loops=1)
                    -> Covering index lookup on topic_work using PRIMARY (topic_id=topic.topic_id)  (cost=0.94 rows=7) (actual time=0.012..0.015 rows=7 loops=
481)
                -> Single-row index lookup on work using PRIMARY (work_id=topic_work.work_id)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=3354)
```

With index on topic.category

```
| -> Sort: `work`.publication_year  (actual time=21.315..21.319 rows=78 loops=1)
    -> Table scan on <temporary>  (actual time=21.267..21.276 rows=78 loops=1)
        -> Aggregate using temporary table  (actual time=21.265..21.265 rows=78 loops=1)
            -> Nested loop inner join  (cost=2145.84 rows=3462) (actual time=0.076..20.042 rows=3354 loops=1)
                -> Nested loop inner join  (cost=934.29 rows=3462) (actual time=0.063..7.806 rows=3354 loops=1)
                    -> Covering index lookup on topic using index2 (category='Medicine')  (cost=137.83 rows=481) (actual time=0.035..0.300 rows=481 loops=1)
                    -> Covering index lookup on topic_work using PRIMARY (topic_id=topic.topic_id)  (cost=0.94 rows=7) (actual time=0.011..0.015 rows=7 loops=
481)
                -> Single-row index lookup on work using PRIMARY (work_id=topic_work.work_id)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=3354)
 |
```

We will only keep work.publication_year indexing.
Work.publication_year was kept and very helpful. It is because there are many years in the table, and the year shows up in intensive clauses like GROUP BY and ORDER BY. Both of these take up resources and seem to requery the fields, making indexing worthwhile.
topics.category seems to help as it is used in some intensive SQL clauses, particularly the where clause that checks if it is equal to a given value.

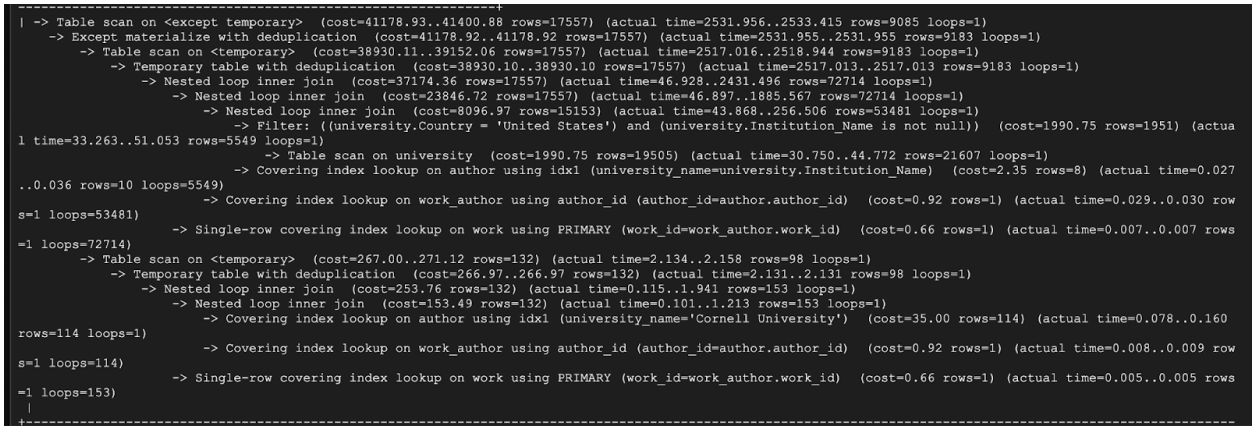
Query 2 without any indexing:

```
-------------------------------------------------------------------+
| -> Table scan on <except temporary>  (cost=32128.02..32349.97 rows=17557) (actual time=2819.528..2820.778 rows=9085 loops=1)
    -> Except materialize with deduplication  (cost=32128.01..32128.01 rows=17557) (actual time=2819.526..2819.526 rows=9183 loops=1)
        -> Table scan on <temporary>  (cost=29947.68..30169.63 rows=17557) (actual time=2798.573..2800.263 rows=9183 loops=1)
            -> Temporary table with deduplication  (cost=29947.67..29947.67 rows=17557) (actual time=2798.568..2798.568 rows=9183 loops=1)
                -> Nested loop inner join  (cost=28191.93 rows=17557) (actual time=135.920..2713.103 rows=72714 loops=1)
                    -> Nested loop inner join  (cost=22046.85 rows=17557) (actual time=135.889..2442.864 rows=72714 loops=1)
                        -> Nested loop inner join  (cost=8216.51 rows=15153) (actual time=62.095..842.304 rows=53481 loops=1)
                            -> Filter: ((university.Country = 'United States') and (university.Institution_Name is not null))  (cost=2110.29 rows=1951) (actua
l time=15.099..369.694 rows=5549 loops=1)
                                -> Table scan on university  (cost=2110.29 rows=19505) (actual time=13.881..362.640 rows=21607 loops=1)
                            -> Covering index lookup on author using idx1 (university_name=university.Institution_Name)  (cost=2.35 rows=8) (actual time=0.071
..0.084 rows=10 loops=5549)
                        -> Covering index lookup on work_author using author_id (author_id=author.author_id)  (cost=0.80 rows=1) (actual time=0.028..0.030 row
s=1 loops=53481)
                    -> Single-row covering index lookup on work using PRIMARY (work_id=work_author.work_id)  (cost=0.25 rows=1) (actual time=0.003..0.004 rows
=1 loops=72714)
        -> Table scan on <temporary>  (cost=198.52..202.64 rows=132) (actual time=2.629..2.646 rows=98 loops=1)
            -> Temporary table with deduplication  (cost=198.49..198.49 rows=132) (actual time=2.625..2.625 rows=98 loops=1)
                -> Nested loop inner join  (cost=185.28 rows=132) (actual time=0.636..2.223 rows=153 loops=1)
                    -> Nested loop inner join  (cost=139.05 rows=132) (actual time=0.619..1.510 rows=153 loops=1)
                        -> Covering index lookup on author using idx1 (university_name='Cornell University')  (cost=35.00 rows=114) (actual time=0.593..0.656
rows=114 loops=1)
                        -> Covering index lookup on work_author using author_id (author_id=author.author_id)  (cost=0.80 rows=1) (actual time=0.006..0.007 row
s=1 loops=114)
                    -> Single-row covering index lookup on work using PRIMARY (work_id=work_author.work_id)  (cost=0.25 rows=1) (actual time=0.004..0.004 rows
=1 loops=153)
 |
+---------------------------------------------------------------------------------------------------------------------------------------------------
```

With index on university.country

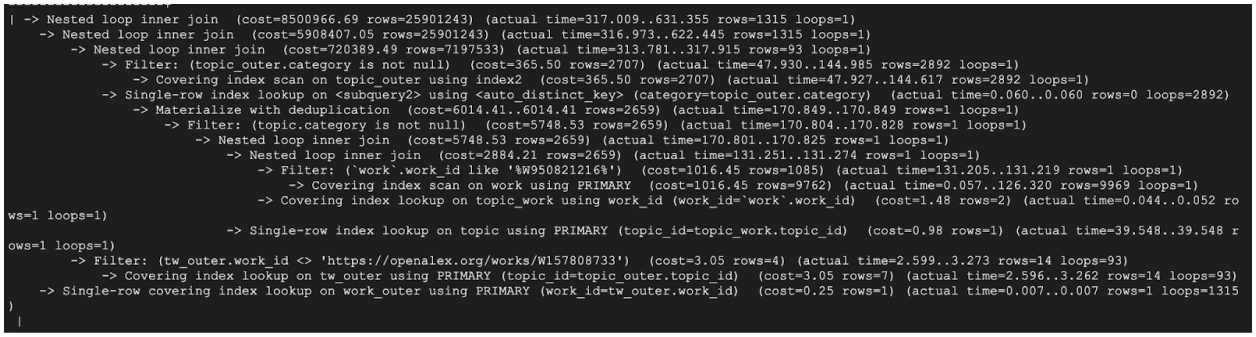
```
 -----------------------------------------------+
| -> Table scan on <except temporary>  (cost=92337.35..92964.20 rows=49949) (actual time=1202.613..1203.825 rows=9085 loops=1)
    -> Except materialize with deduplication  (cost=92337.34..92337.34 rows=49949) (actual time=1202.612..1202.612 rows=9183 loops=1)
        -> Table scan on <temporary>  (cost=86495.41..87122.26 rows=49949) (actual time=1188.854..1190.486 rows=9183 loops=1)
            -> Temporary table with deduplication  (cost=86495.40..86495.40 rows=49949) (actual time=1188.850..1188.850 rows=9183 loops=1)
                -> Nested loop inner join  (cost=81500.48 rows=49949) (actual time=20.162..1113.603 rows=72714 loops=1)
                    -> Nested loop inner join  (cost=34389.33 rows=49949) (actual time=0.896..579.814 rows=72714 loops=1)
                        -> Nested loop inner join  (cost=18047.29 rows=43108) (actual time=0.881..246.538 rows=53481 loops=1)
                            -> Filter: (university.Institution_Name is not null)  (cost=675.65 rows=5549) (actual time=0.851..19.163 rows=5549 loops=1)
                                -> Index lookup on university using index3 (Country='United States')  (cost=675.65 rows=5549) (actual time=0.848..18.343 rows=
5549 loops=1)
                            -> Covering index lookup on author using idx1 (university_name=university.Institution_Name)  (cost=2.35 rows=8) (actual time=0.033
..0.040 rows=10 loops=5549)
                        -> Covering index lookup on work_author using author_id (author_id=author.author_id)  (cost=0.26 rows=1) (actual time=0.005..0.006 row
s=1 loops=53481)
                    -> Single-row covering index lookup on work using PRIMARY (work_id=work_author.work_id)  (cost=0.84 rows=1) (actual time=0.007..0.007 rows
=1 loops=72714)
        -> Table scan on <temporary>  (cost=216.04..220.16 rows=132) (actual time=3.194..3.210 rows=98 loops=1)
            -> Temporary table with deduplication  (cost=216.01..216.01 rows=132) (actual time=3.190..3.190 rows=98 loops=1)
                -> Nested loop inner join  (cost=202.80 rows=132) (actual time=1.521..3.026 rows=153 loops=1)
                    -> Nested loop inner join  (cost=78.22 rows=132) (actual time=1.508..2.425 rows=153 loops=1)
                        -> Covering index lookup on author using idx1 (university_name='Cornell University')  (cost=35.00 rows=114) (actual time=1.475..1.539
rows=114 loops=1)
                        -> Covering index lookup on work_author using author_id (author_id=author.author_id)  (cost=0.26 rows=1) (actual time=0.006..0.008 row
s=1 loops=114)
                    -> Single-row covering index lookup on work using PRIMARY (work_id=work_author.work_id)  (cost=0.84 rows=1) (actual time=0.004..0.004 rows
=1 loops=153)
 |
+---------------------------------------------------------------------------------------------------------------------------------------------------
```

With index on author.university_name

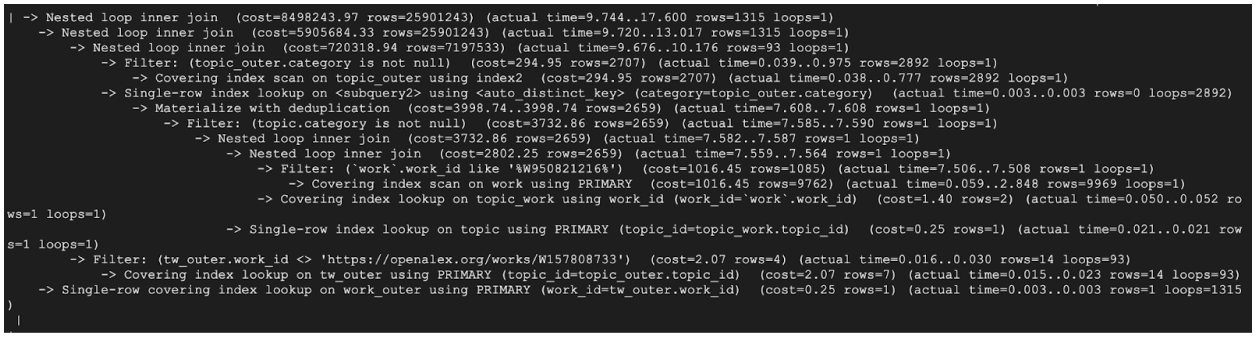


We did not keep university.country because it made performance worse. It may be because there are very few countries relative to the number of entries in the university table. The overhead of maintaining indexing is not worth it for such few performance gains.
This didn't get us much. This may be because many author.university_name entries are equal to Unknown so we don't get as much benefit. This field also isn't used in any intensive SQL clause so its overhead isn't worth it.

Query 3 without any indexing:

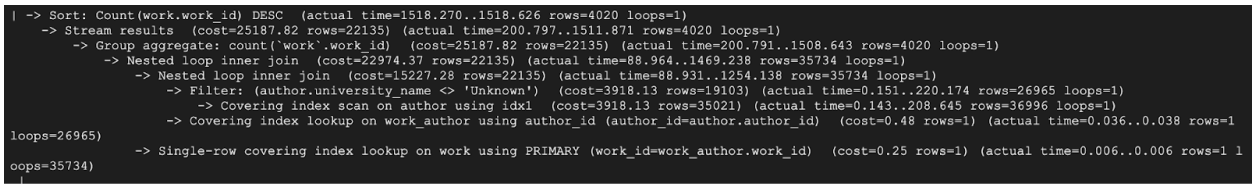


With index on topic.category



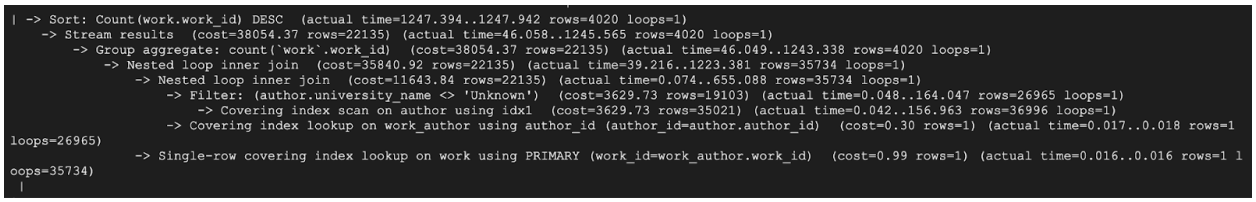
topic.category is the only field in this query that we should manually apply an index to as it is the only non-primary key. However, it does not seem to have any effect on the cost. This may be because there are relatively few categories compared to topics so there aren't many benefits to indexing this.

Query 4 without any indexing



```
| -> Sort: Count(work.work_id) DESC  (actual time=1518.270..1518.626 rows=4020 loops=1)
    -> Stream results  (cost=25187.82 rows=22135) (actual time=200.797..1511.871 rows=4020 loops=1)
        -> Group aggregate: count(`work`.work_id)  (cost=25187.82 rows=22135) (actual time=200.791..1508.643 rows=4020 loops=1)
            -> Nested loop inner join  (cost=22974.37 rows=22135) (actual time=88.964..1469.238 rows=35734 loops=1)
                -> Nested loop inner join  (cost=15227.28 rows=22135) (actual time=88.931..1254.138 rows=35734 loops=1)
                    -> Filter: (author.university_name <> 'Unknown')  (cost=3918.13 rows=19103) (actual time=0.151..220.174 rows=26965 loops=1)
                        -> Covering index scan on author using idx1  (cost=3918.13 rows=35021) (actual time=0.143..208.645 rows=36996 loops=1)
                    -> Covering index lookup on work_author using author_id (author_id=author.author_id)  (cost=0.48 rows=1) (actual time=0.036..0.038 rows=1
loops=26965)
                -> Single-row covering index lookup on work using PRIMARY (work_id=work_author.work_id)  (cost=0.25 rows=1) (actual time=0.006..0.006 rows=1 l
oops=35734)
 |
```

With index of author.university_name



```
| -> Sort: Count(work.work_id) DESC  (actual time=1247.394..1247.942 rows=4020 loops=1)
    -> Stream results  (cost=38054.37 rows=22135) (actual time=46.058..1245.565 rows=4020 loops=1)
        -> Group aggregate: count(`work`.work_id)  (cost=38054.37 rows=22135) (actual time=46.049..1243.338 rows=4020 loops=1)
            -> Nested loop inner join  (cost=35840.92 rows=22135) (actual time=39.216..1223.381 rows=35734 loops=1)
                -> Nested loop inner join  (cost=11643.84 rows=22135) (actual time=0.074..655.088 rows=35734 loops=1)
                    -> Filter: (author.university_name <> 'Unknown')  (cost=3629.73 rows=19103) (actual time=0.048..164.047 rows=26965 loops=1)
                        -> Covering index scan on author using idx1  (cost=3629.73 rows=35021) (actual time=0.042..156.963 rows=36996 loops=1)
                    -> Covering index lookup on work_author using author_id (author_id=author.author_id)  (cost=0.30 rows=1) (actual time=0.017..0.018 rows=1
loops=26965)
                -> Single-row covering index lookup on work using PRIMARY (work_id=work_author.work_id)  (cost=0.99 rows=1) (actual time=0.016..0.016 rows=1 l
oops=35734)
 |
```

This didn’t get us much. This may be because many author.university_name entries are equal to Unknown so we don’t get as much benefit. This field also isn’t used in any intensive SQL clause so its overhead isn’t worth it.

We will not be keeping this indexing.

For queries 3 and 4, we only have one field that is not a primary key. Therefore, we can only try one configuration (other than the default).