DBMS

Project Milestone - 2

BY

Ruptirumal Sai Bodavula 2022MCS2064

Raajita Bhamidipaty 2022MCS2053

Siddharth S 2022MCS2061

18th April 2023

Database Design and Size

Size of Database: 9.8 MB

Relation Name	Tuple Count
Badges	11188
Comments	3577
Post_History	5646
Post_Links	268
Posts	1990
Tags	89
Users	6309
Votes	9325

Table 1.1: Size of relations in the database

As we can see above we have 8 relations with varying numbers of tuples. We had previously cleaned the data and uploaded the data on the remote server. In this milestone, we wrote down the queries that are necessary and relevant for our project to function as expected.
The queries are written in the **queries.sql** file which has been submitted with this report. We connected to the remote server and tried to run each of the queries that we have mentioned in the .sql file. While doing so, we ran the **Explain Analyze** command for each of the queries mentioned below and noted down the running time for both the query plan and the execution.

Queries to drive the application:

Following are the queries required to drive the application which are required to be executed based on users' interaction with the application.

- **New user account creation:**

  All not null fields must necessarily be populated, rest may or may not be
  Example of user creation:
  INSERT INTO users (reputation, display_name, creation_date,about_me,last_access_date) VALUES (0, 'some_user', CURRENT_TIMESTAMP, '<p>Likes: helping people, software and technology, personal finance, consumer protection..</p>', CURRENT_TIMESTAMP);

- **Deleting a user with given id**
  Following queries constituting a transaction must be executed in order

  1.Posts table: all posts owned by the user will have owner_user_id updated to NULL and owner_display_name populated

  UPDATE posts SET owner_user_id= NULL, owner_display_name=(select display_name from users where id='user_id_to_be_deleted' where owner_user_id='user_id_to_be_deleted');

  2.Comments table: all comments written by the user will have user_id updated to NULL and user_display_name populated

  UPDATE comments SET user_id=NULL,user_display_name= select display_name from users where id='user_id_to_be_deleted' where user_id='user_id_to_be_deleted';

  3.Deleting all votes by the user:
  Delete all entries of user from votes table
  delete from votes where user_id='deleted_user_id';

  4.Deleting all badges of the user
  Delete from badges where user_id='deleted_user_id';

  5.Finally, deleting user from users table
  Delete from users where id='deleted_user_id';

- **Authenticating a user during login**
  1.Checking if user_id is present in users table
  Select case when exists(select * from users where user_id=’entered_id’) then True when not exists (select * from users where user_id=’entered_id’) then False end as valid_user;

  2.Retrieving password
  Select password from users where user_id=’entered_user_id’;

- **Displaying users’ profile info**

  fetch user info given id
  Ex-
  select * from users where id=’828’;
  fetch badges earned given id
  Ex-
  Number of gold badges
  Select count(*) from badges where class=1 and user_id='828';
  Number of silver badges
  Select count(*) from badges where class=2 and user_id='828';
  Number of bronze badges
  Select count(*) from badges where class=3 and user_id='828';

  fetch top-k posts of user (top k with highest upvotes)
  **With t1 as (select id from posts where owner_user_id=’828’),
  t2 as (select post_id,count(*) as cnt from votes where post_id in (select id from t1) and vote_type_id=2 group by post_id) select post_id from t2 order by cnt desc limit 5;**

- **Updating user profile info**
  Ex-
  UPDATE users set display_name='new_name',about_me='<p>Im a stock market investor</p>' Where id='828';

- **Finding upvotes given post_id**
  Select count(*) from VOTES where VOTES.vote_type_id=2 and post_id=2;

- **Finding downvotes given post_id**

Select count(*) from VOTES where VOTES.vote_type_id=3 and post_id=2

- **Fetch top-k question posts based on upvotes for home page**
  With t as (select post_id from votes,posts where vote_type_id=2 and votes.post_id=posts.id and posts.post_type_id=1 group by post_id order by count(*) desc limit 5) select * from posts where id in (select post_id from t);

- **Fetch top-K question posts after offset**
  with t as (select post_id from votes,posts where vote_type_id=2 and votes.post_id=posts.id and posts.post_type_id=1 group by post_id order by count(*) desc limit 5 offset 5) select * from posts where id in (select post_id from t);

- **Fetch post info given post id**
  Select * from POSTS where POSTS.id=2;

- **Queries to implement search bar (Search by tag or title keywords)**
- Retrieve top-k posts having a certain tag
  With t as (select id from posts where tags like “%discussion%”), t1 as (select post_id,count(*) as cnt from votes,posts where vote_type_id=2 and post_id in (select id from t) and votes.post_id=posts.id and posts.post_type_id=1 group by post_id order by cnt desc limit 5) select * from posts where id in (select post_id from t1);

- Retrieve top-k question posts having with certain keyword
  with t as (select id from posts where title like “%links%”), t1 as (select post_id,count(*) as cnt from votes,posts where vote_type_id=2 and post_id in (select id from t) and votes.post_id=posts.id and posts.post_type_id=1 group by post_id order by cnt desc limit 5) select * from posts where id in (select post_id from t1);

- **Comments pertaining to a post:**
  Comments must be retrieved when a post is opened
  select * from COMMENTS where COMMENTS.POST_id=2;

- **Retrieving answer posts given question post id:**
  select * from posts where parent_id='2' and post_type_id=2;

- **Casting upvote/downvote**
  Update user table with cast vote. Following 2 queries must be executed in a single transaction.
  Ex- **1.**Increment up_vote count in users table
  Update users set up_votes=up_votes+1 where id=’828’;

2.update votes table

Insert into votes(id,user_id,post_id,vote_type_id,creation_date) values (DEFAULT,828,2,2,CURRENT_TIMESTAMP)

- **Updating comment count in posts table after inserting comment**
  update posts set comment_count=comment_count+1 where id='parent_post_id';

- **Creating a post (if it is answer post, parent_id can't be null)**
  INSERT INTO posts (owner_user_id, post_type_id, score, title, body, creation_date)
  VALUES (-1, 1, 0, 'Some post', 'Some text', CURRENT_TIMESTAMP);

- **Updating posts table**

  Only title, body and tags of a post can be edited
  INSERT INTO posts (title, tags, body, last_edit_date,last_activity_date, post_type_id, score, creation_date)
  VALUES ('New Title', 'New Tags', 'Edited body',CURRENT_TIMESTAMP,CURRENT_TIMESTAMP,1, 0, CURRENT_TIMESTAMP);

- **Marking answer as accepted**
  Ex-
  1.Retrieving user_id of question owner (to make sure he is authorized to accept answer)
  Select owner_user_id from posts where id=(Select parent_id from posts where id='2')
  2.Updating accepted_answer_id in question post
  update posts set accepted_answer_id= 'answer_id' where id=(select parent_id from posts where id='answer_id')

- **Editing a post**
  1.Checking user only one who the post belongs to should be allowed for editing
  Select case when exists (Select owner_user_id from posts where id=2 and owner_user_id is NULL) then False when exists (Select owner_user_id from posts where id=2 and owner_user_id is not NULL and owner_user_id=828) then True end as allowed_to_edit;

  2.Updating posts table
  Only title, body and tags of a post can be edited
  INSERT INTO posts (title, tags, body, last_edit_date,last_activity_date, post_type_id, score, creation_date)

VALUES ('New Title', 'New Tags', 'Edited body',CURRENT_TIMESTAMP,CURRENT_TIMESTAMP,1, 0, CURRENT_TIMESTAMP);

- **Deleting a post**
  1. Deleting from posts links
  delete from post_links where post_id=2 or related_post_id=2;

  2.deleting all comments
  delete from comments where post_id=2;

  3.Deleting from votes
  delete from votes where post_id=2;

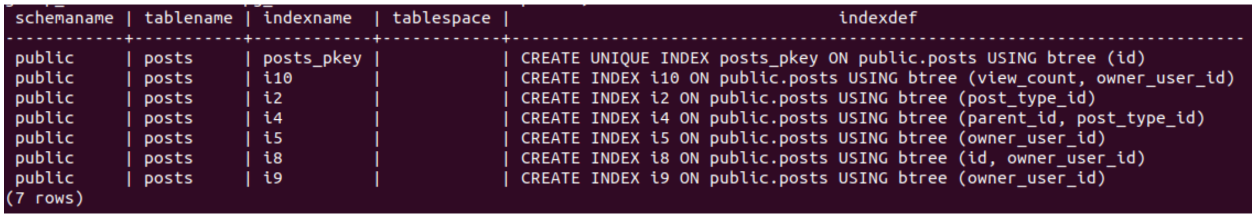
  4.Finally, delete from posts table
  delete from posts where id='post_to_be_deleted';

- **Remove comment to a post**
  Following 2 queries must be executed in a single transaction
  1.Decrementing comment count
  Update posts set comment_count=comment_count-1 where id=(select post_id from comments where id=7);
  2.Deleting from comments table
  delete from COMMENTS where id='7';

Choice of Indexes for optimization:

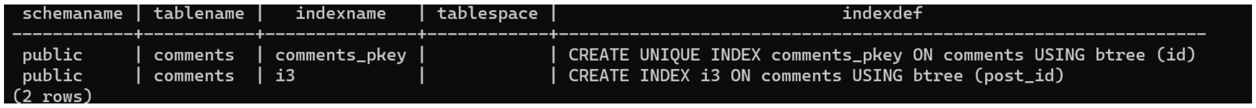
All tables have a serial primary key. There are default indexes built on these serial primary keys. Other than those we created other indexes that helped us optimize query execution. These choices were made after experimenting and observing the query plans generated. We experimented with various indexes that were created based on the predicates in the where clause. Some of these are indexes on composite keys (2 or more columns) that were observed to speed up queries with many predicates separated by 'and' involving multiple columns.

Posts table:



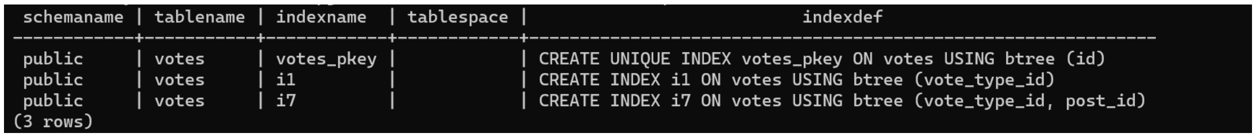
```
 schemaname | tablename | indexname  | tablespace |                                  indexdef
------------+-----------+------------+------------+-------------------------------------------------------------------------------
 public     | posts     | posts_pkey |            | CREATE UNIQUE INDEX posts_pkey ON public.posts USING btree (id)
 public     | posts     | i10        |            | CREATE INDEX i10 ON public.posts USING btree (view_count, owner_user_id)
 public     | posts     | i2         |            | CREATE INDEX i2 ON public.posts USING btree (post_type_id)
 public     | posts     | i4         |            | CREATE INDEX i4 ON public.posts USING btree (parent_id, post_type_id)
 public     | posts     | i5         |            | CREATE INDEX i5 ON public.posts USING btree (owner_user_id)
 public     | posts     | i8         |            | CREATE INDEX i8 ON public.posts USING btree (id, owner_user_id)
 public     | posts     | i9         |            | CREATE INDEX i9 ON public.posts USING btree (owner_user_id)
(7 rows)
```

Comments table:



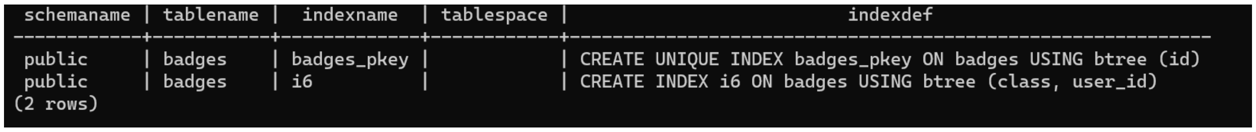
```
 schemaname | tablename |   indexname    | tablespace |                              indexdef
------------+-----------+----------------+------------+--------------------------------------------------------------------
 public     | comments  | comments_pkey  |            | CREATE UNIQUE INDEX comments_pkey ON comments USING btree (id)
 public     | comments  | i3             |            | CREATE INDEX i3 ON comments USING btree (post_id)
(2 rows)
```

Votes table:



```
 schemaname | tablename | indexname  | tablespace |                           indexdef
------------+-----------+------------+------------+--------------------------------------------------------------
 public     | votes     | votes_pkey |            | CREATE UNIQUE INDEX votes_pkey ON votes USING btree (id)
 public     | votes     | i1         |            | CREATE INDEX i1 ON votes USING btree (vote_type_id)
 public     | votes     | i7         |            | CREATE INDEX i7 ON votes USING btree (vote_type_id, post_id)
(3 rows)
```

Badges table:



```
 schemaname | tablename |  indexname  | tablespace |                            indexdef
------------+-----------+-------------+------------+--------------------------------------------------------------
 public     | badges    | badges_pkey |            | CREATE UNIQUE INDEX badges_pkey ON badges USING btree (id)
 public     | badges    | i6          |            | CREATE INDEX i6 ON badges USING btree (class, user_id)
(2 rows)
```

Create Index Commands

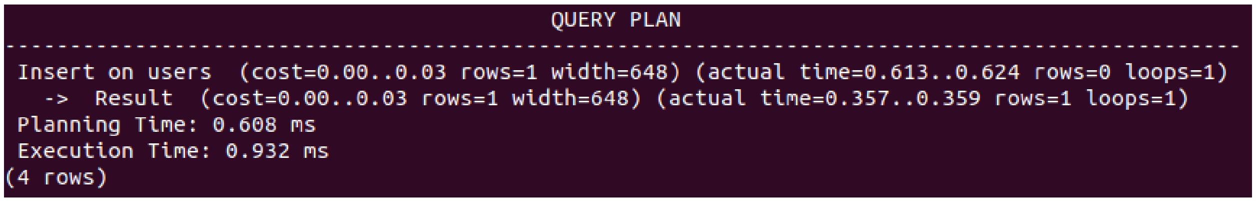
create index I1 on votes(vote_type_id);
create index I2 on posts(post_type_id);
create index I3 on comments(post_id);
create index I4 on posts(parent_id,post_type_id);
create index I5 on posts(owner_user_id);
create index I6 on badges(class,user_id);
create index I7 on votes(vote_type_id,post_id);
create index I8 on posts(id,owner_user_id);
create index I9 on posts(owner_user_id);
create index I10 on posts(view_count,owner_user_id);

Performance Metrics

Query 1:

Adding new user (user account creation)

```
INSERT INTO users (reputation, display_name,
creation_date,about_me,last_access_date)
VALUES (0, 'some_user', CURRENT_TIMESTAMP, '<p>Likes: helping people,
software and technology, personal finance, consumer protection..</p>',
CURRENT_TIMESTAMP);
```



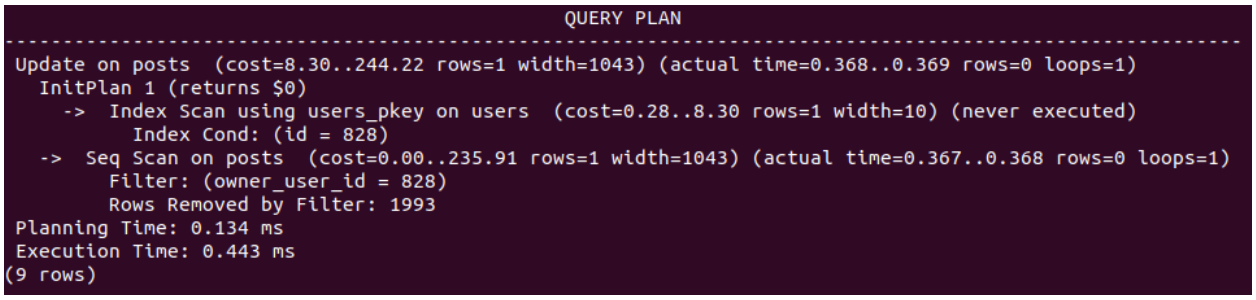
```
                                                QUERY PLAN
------------------------------------------------------------------------------------------------
 Insert on users  (cost=0.00..0.03 rows=1 width=648) (actual time=0.613..0.624 rows=0 loops=1)
   ->  Result  (cost=0.00..0.03 rows=1 width=648) (actual time=0.357..0.359 rows=1 loops=1)
 Planning Time: 0.608 ms
 Execution Time: 0.932 ms
(4 rows)
```

Time:1.540 ms

Query 2:

Deleting a user with given ID

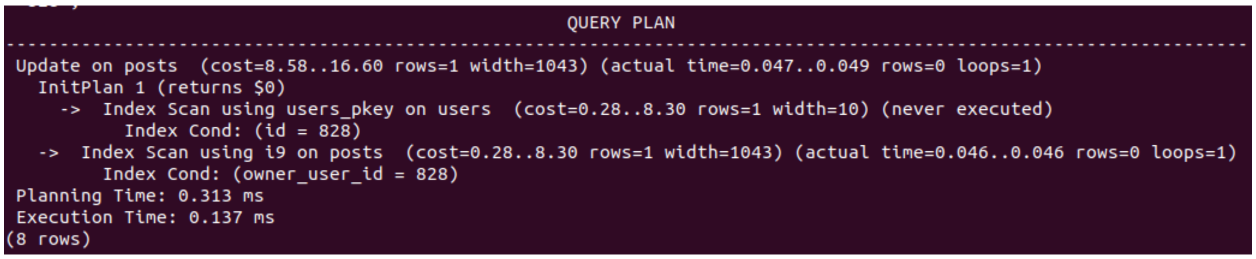
```
UPDATE posts SET owner_user_id= NULL, owner_display_name=(select
display_name from users where id='828') where owner_user_id='828';
```



```
                                                 QUERY PLAN
--------------------------------------------------------------------------------------------------------
 Update on posts  (cost=8.30..244.22 rows=1 width=1043) (actual time=0.368..0.369 rows=0 loops=1)
   InitPlan 1 (returns $0)
     ->  Index Scan using users_pkey on users  (cost=0.28..8.30 rows=1 width=10) (never executed)
           Index Cond: (id = 828)
   ->  Seq Scan on posts  (cost=0.00..235.91 rows=1 width=1043) (actual time=0.367..0.368 rows=0 loops=1)
         Filter: (owner_user_id = 828)
         Rows Removed by Filter: 1993
 Planning Time: 0.134 ms
 Execution Time: 0.443 ms
(9 rows)
```

Time:0.577 ms

AFTER INDEXING



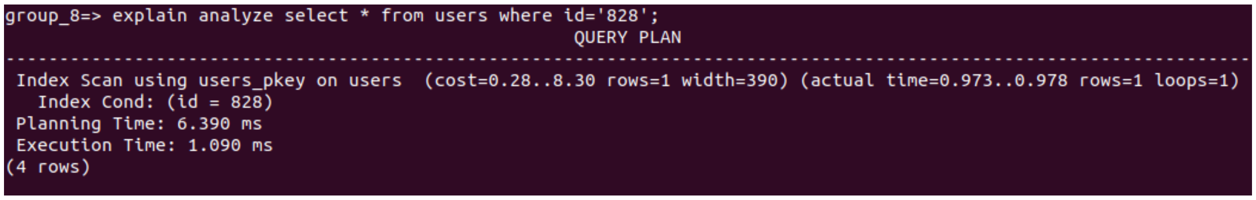
```
                                                 QUERY PLAN
--------------------------------------------------------------------------------------------------------
 Update on posts  (cost=8.58..16.60 rows=1 width=1043) (actual time=0.047..0.049 rows=0 loops=1)
   InitPlan 1 (returns $0)
     ->  Index Scan using users_pkey on users  (cost=0.28..8.30 rows=1 width=10) (never executed)
           Index Cond: (id = 828)
   ->  Index Scan using i9 on posts  (cost=0.28..8.30 rows=1 width=1043) (actual time=0.046..0.046 rows=0 loops=1)
         Index Cond: (owner_user_id = 828)
 Planning Time: 0.313 ms
 Execution Time: 0.137 ms
(8 rows)
```

Time: 0.45 ms

As can be seen after indexing, sequential scan to filter posts owned by a user was replaced by an index scan

Query 3:
Fetch user info given id

```
select * from users where id='828';
```

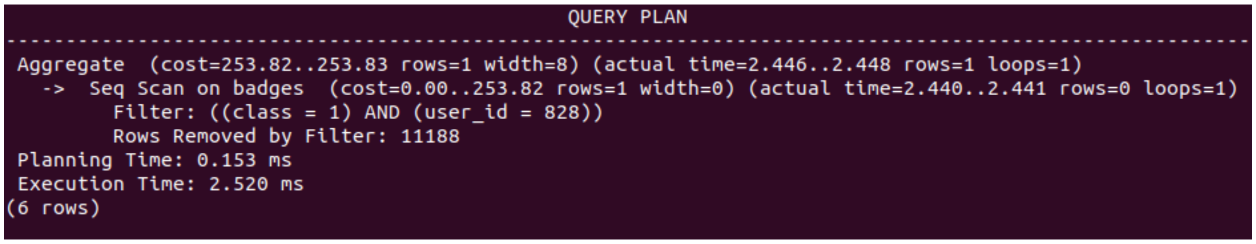


```
group_8=> explain analyze select * from users where id='828';
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Index Scan using users_pkey on users  (cost=0.28..8.30 rows=1 width=390) (actual time=0.973..0.978 rows=1 loops=1)
   Index Cond: (id = 828)
 Planning Time: 6.390 ms
 Execution Time: 1.090 ms
(4 rows)
```

Time: 7.4 ms

Query 4:
Fetch badges earned given ID

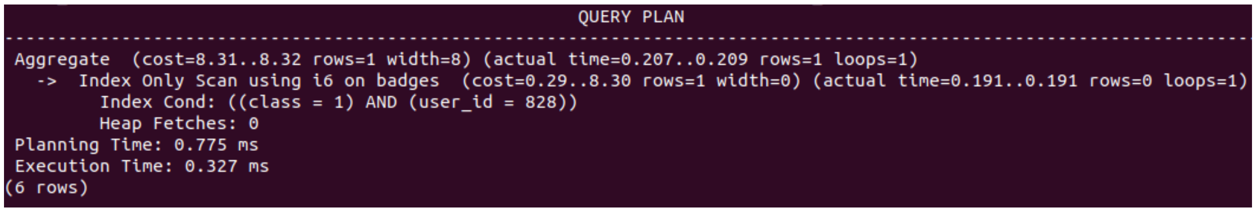
```
select count(*) from badges where class=1 and user_id='828';
```



```
                                              QUERY PLAN
--------------------------------------------------------------------------------------------------------
 Aggregate  (cost=253.82..253.83 rows=1 width=8) (actual time=2.446..2.448 rows=1 loops=1)
   ->  Seq Scan on badges  (cost=0.00..253.82 rows=1 width=0) (actual time=2.440..2.441 rows=0 loops=1)
         Filter: ((class = 1) AND (user_id = 828))
         Rows Removed by Filter: 11188
 Planning Time: 0.153 ms
 Execution Time: 2.520 ms
(6 rows)
```

Time: 3.7 ms

AFTER INDEXING



```
                                                      QUERY PLAN
------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=8.31..8.32 rows=1 width=8) (actual time=0.207..0.209 rows=1 loops=1)
   ->  Index Only Scan using i6 on badges  (cost=0.29..8.30 rows=1 width=0) (actual time=0.191..0.191 rows=0 loops=1)
         Index Cond: ((class = 1) AND (user_id = 828))
         Heap Fetches: 0
 Planning Time: 0.775 ms
 Execution Time: 0.327 ms
(6 rows)
```

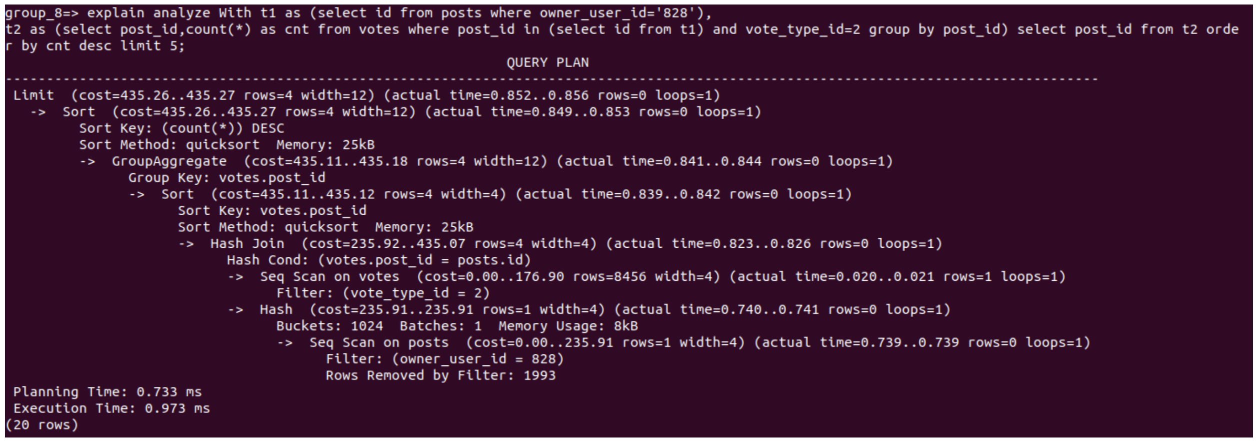
Time: 1.09 ms

Composite key index was created on keys class and user_id on badges tables to speed up query.

Query 5:

Fetch top-k posts of user (top k with highest upvotes)

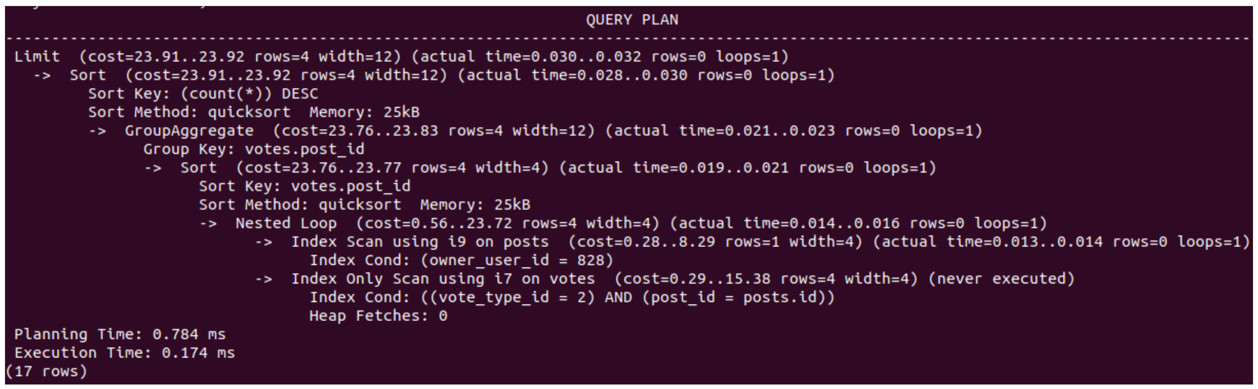
```sql
With t1 as (select id from posts where owner_user_id='828'),
t2 as (select post_id,count(*) as cnt from votes where post_id in (select
id from t1) and vote_type_id=2 group by post_id) select post_id from t2
order by cnt desc limit 5;
```



```
group_8=> explain analyze With t1 as (select id from posts where owner_user_id='828'),
t2 as (select post_id,count(*) as cnt from votes where post_id in (select id from t1) and vote_type_id=2 group by post_id) select post_id from t2 orde
r by cnt desc limit 5;
                                                              QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------
 Limit  (cost=435.26..435.27 rows=4 width=12) (actual time=0.852..0.856 rows=0 loops=1)
   ->  Sort  (cost=435.26..435.27 rows=4 width=12) (actual time=0.849..0.853 rows=0 loops=1)
         Sort Key: (count(*)) DESC
         Sort Method: quicksort  Memory: 25kB
         ->  GroupAggregate  (cost=435.11..435.18 rows=4 width=12) (actual time=0.841..0.844 rows=0 loops=1)
               Group Key: votes.post_id
               ->  Sort  (cost=435.11..435.12 rows=4 width=4) (actual time=0.839..0.842 rows=0 loops=1)
                     Sort Key: votes.post_id
                     Sort Method: quicksort  Memory: 25kB
                     ->  Hash Join  (cost=235.92..435.07 rows=4 width=4) (actual time=0.823..0.826 rows=0 loops=1)
                           Hash Cond: (votes.post_id = posts.id)
                           ->  Seq Scan on votes  (cost=0.00..176.90 rows=8456 width=4) (actual time=0.020..0.021 rows=1 loops=1)
                                 Filter: (vote_type_id = 2)
                           ->  Hash  (cost=235.91..235.91 rows=1 width=4) (actual time=0.740..0.741 rows=0 loops=1)
                                 Buckets: 1024  Batches: 1  Memory Usage: 8kB
                                 ->  Seq Scan on posts  (cost=0.00..235.91 rows=1 width=4) (actual time=0.739..0.739 rows=0 loops=1)
                                       Filter: (owner_user_id = 828)
                                       Rows Removed by Filter: 1993
 Planning Time: 0.733 ms
 Execution Time: 0.973 ms
(20 rows)
```

Time: 1.6 ms

AFTER INDEXING



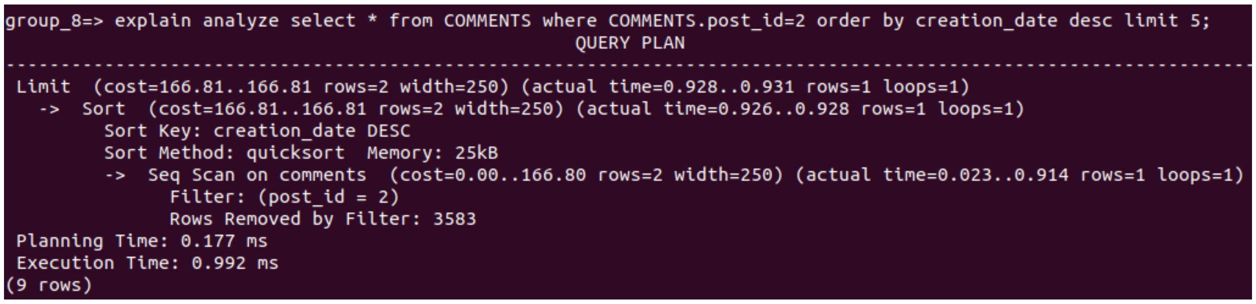
```
                                                              QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------
 Limit  (cost=23.91..23.92 rows=4 width=12) (actual time=0.030..0.032 rows=0 loops=1)
   ->  Sort  (cost=23.91..23.92 rows=4 width=12) (actual time=0.028..0.030 rows=0 loops=1)
         Sort Key: (count(*)) DESC
         Sort Method: quicksort  Memory: 25kB
         ->  GroupAggregate  (cost=23.76..23.83 rows=4 width=12) (actual time=0.021..0.023 rows=0 loops=1)
               Group Key: votes.post_id
               ->  Sort  (cost=23.76..23.77 rows=4 width=4) (actual time=0.019..0.021 rows=0 loops=1)
                     Sort Key: votes.post_id
                     Sort Method: quicksort  Memory: 25kB
                     ->  Nested Loop  (cost=0.56..23.72 rows=4 width=4) (actual time=0.014..0.016 rows=0 loops=1)
                           ->  Index Scan using i9 on posts  (cost=0.28..8.29 rows=1 width=4) (actual time=0.013..0.014 rows=0 loops=1)
                                 Index Cond: (owner_user_id = 828)
                           ->  Index Only Scan using i7 on votes  (cost=0.29..15.38 rows=4 width=4) (never executed)
                                 Index Cond: ((vote_type_id = 2) AND (post_id = posts.id))
                                 Heap Fetches: 0
 Planning Time: 0.784 ms
 Execution Time: 0.174 ms
(17 rows)
```

Time: 0.9ms

Query 6:

Once a post is clicked, we can load all the comments

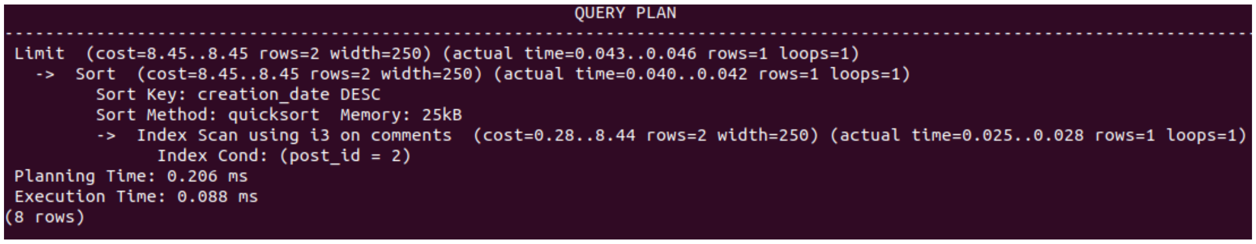
```
select * from COMMENTS where COMMENTS.post_id=2 order by creation_date
desc limit 5;
```



```
group_8=> explain analyze select * from COMMENTS where COMMENTS.post_id=2 order by creation_date desc limit 5;
                                                    QUERY PLAN
---------------------------------------------------------------------------------------------------------------------
 Limit  (cost=166.81..166.81 rows=2 width=250) (actual time=0.928..0.931 rows=1 loops=1)
   ->  Sort  (cost=166.81..166.81 rows=2 width=250) (actual time=0.926..0.928 rows=1 loops=1)
         Sort Key: creation_date DESC
         Sort Method: quicksort  Memory: 25kB
         ->  Seq Scan on comments  (cost=0.00..166.80 rows=2 width=250) (actual time=0.023..0.914 rows=1 loops=1)
               Filter: (post_id = 2)
               Rows Removed by Filter: 3583
 Planning Time: 0.177 ms
 Execution Time: 0.992 ms
(9 rows)
```

Time: 1.2 ms

AFTER INDEXING



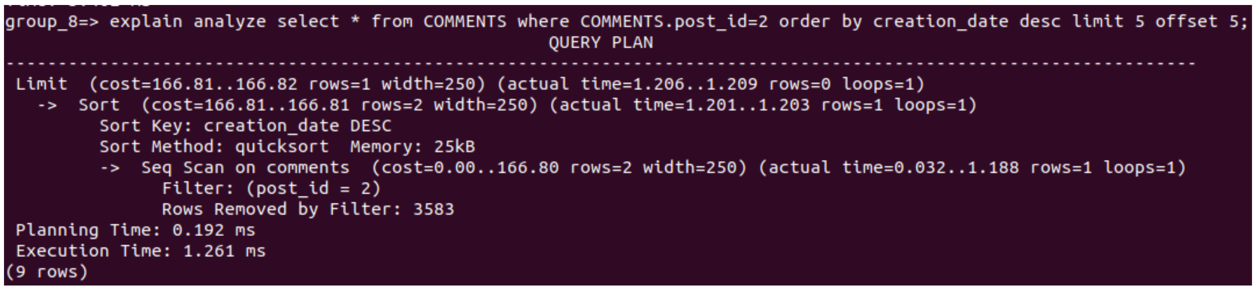
```
                                                    QUERY PLAN
---------------------------------------------------------------------------------------------------------------------
 Limit  (cost=8.45..8.45 rows=2 width=250) (actual time=0.043..0.046 rows=1 loops=1)
   ->  Sort  (cost=8.45..8.45 rows=2 width=250) (actual time=0.040..0.042 rows=1 loops=1)
         Sort Key: creation_date DESC
         Sort Method: quicksort  Memory: 25kB
         ->  Index Scan using i3 on comments  (cost=0.28..8.44 rows=2 width=250) (actual time=0.025..0.028 rows=1 loops=1)
               Index Cond: (post_id = 2)
 Planning Time: 0.206 ms
 Execution Time: 0.088 ms
(8 rows)
```

Time: 0.92ms

Query 7:

Next K=5 comments ordered by creation date

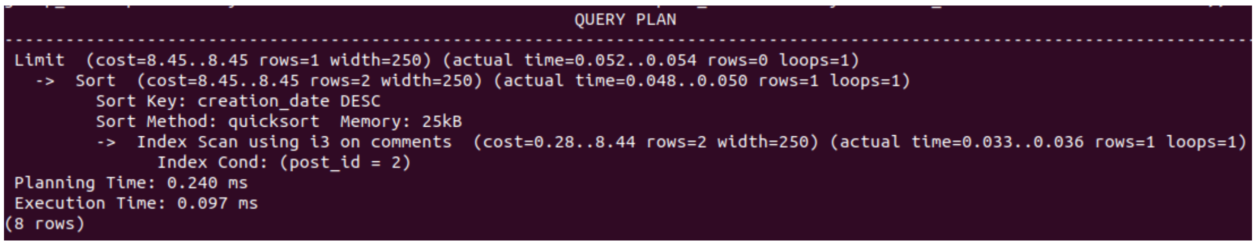
```
select * from COMMENTS where COMMENTS.post_id=2 order by creation_date
desc limit 5 offset 5;
```



```
group_8=> explain analyze select * from COMMENTS where COMMENTS.post_id=2 order by creation_date desc limit 5 offset 5;
                                                  QUERY PLAN
------------------------------------------------------------------------------------------------------------
 Limit  (cost=166.81..166.82 rows=1 width=250) (actual time=1.206..1.209 rows=0 loops=1)
   ->  Sort  (cost=166.81..166.81 rows=2 width=250) (actual time=1.201..1.203 rows=1 loops=1)
         Sort Key: creation_date DESC
         Sort Method: quicksort  Memory: 25kB
         ->  Seq Scan on comments  (cost=0.00..166.80 rows=2 width=250) (actual time=0.032..1.188 rows=1 loops=1)
               Filter: (post_id = 2)
               Rows Removed by Filter: 3583
 Planning Time: 0.192 ms
 Execution Time: 1.261 ms
(9 rows)
```

Time:1.540 ms

AFTER INDEXING

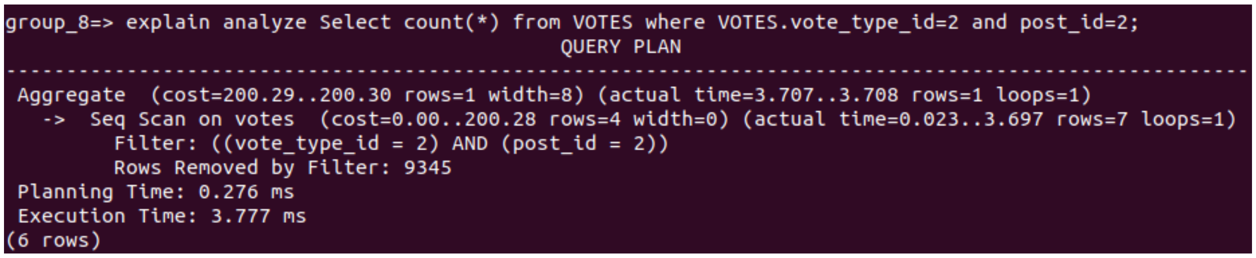


```
                                                   QUERY PLAN
--------------------------------------------------------------------------------------------------------------
 Limit  (cost=8.45..8.45 rows=1 width=250) (actual time=0.052..0.054 rows=0 loops=1)
   ->  Sort  (cost=8.45..8.45 rows=2 width=250) (actual time=0.048..0.050 rows=1 loops=1)
         Sort Key: creation_date DESC
         Sort Method: quicksort  Memory: 25kB
         ->  Index Scan using i3 on comments  (cost=0.28..8.44 rows=2 width=250) (actual time=0.033..0.036 rows=1 loops=1)
               Index Cond: (post_id = 2)
 Planning Time: 0.240 ms
 Execution Time: 0.097 ms
(8 rows)
```

Query 8:

Finding upvotes given post_id

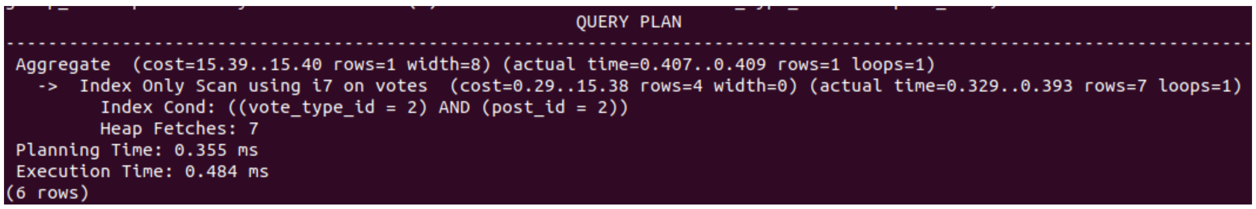
```sql
Select count(*) from VOTES where VOTES.vote_type_id=2 and post_id=2;
```



```
group_8=> explain analyze Select count(*) from VOTES where VOTES.vote_type_id=2 and post_id=2;
                                              QUERY PLAN
---------------------------------------------------------------------------------------------------------
 Aggregate  (cost=200.29..200.30 rows=1 width=8) (actual time=3.707..3.708 rows=1 loops=1)
   ->  Seq Scan on votes  (cost=0.00..200.28 rows=4 width=0) (actual time=0.023..3.697 rows=7 loops=1)
         Filter: ((vote_type_id = 2) AND (post_id = 2))
         Rows Removed by Filter: 9345
 Planning Time: 0.276 ms
 Execution Time: 3.777 ms
(6 rows)
```

Time:4.14 ms

AFTER INDEXING



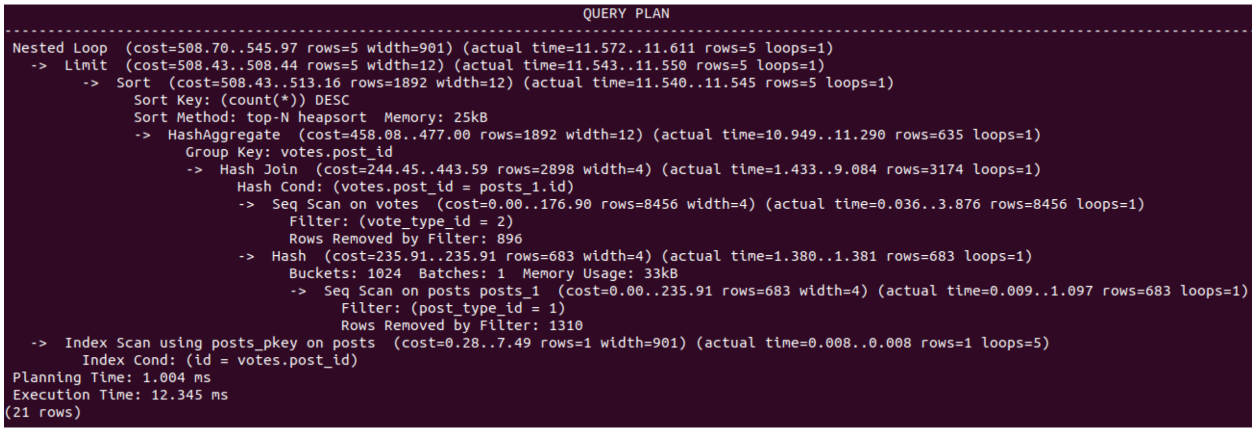
```
                                                         QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=15.39..15.40 rows=1 width=8) (actual time=0.407..0.409 rows=1 loops=1)
   ->  Index Only Scan using i7 on votes  (cost=0.29..15.38 rows=4 width=0) (actual time=0.329..0.393 rows=7 loops=1)
         Index Cond: ((vote_type_id = 2) AND (post_id = 2))
         Heap Fetches: 7
 Planning Time: 0.355 ms
 Execution Time: 0.484 ms
(6 rows)
```

Time: 0.835 ms

Query 9:

Fetch top-K=5 question posts based on upvotes for home page

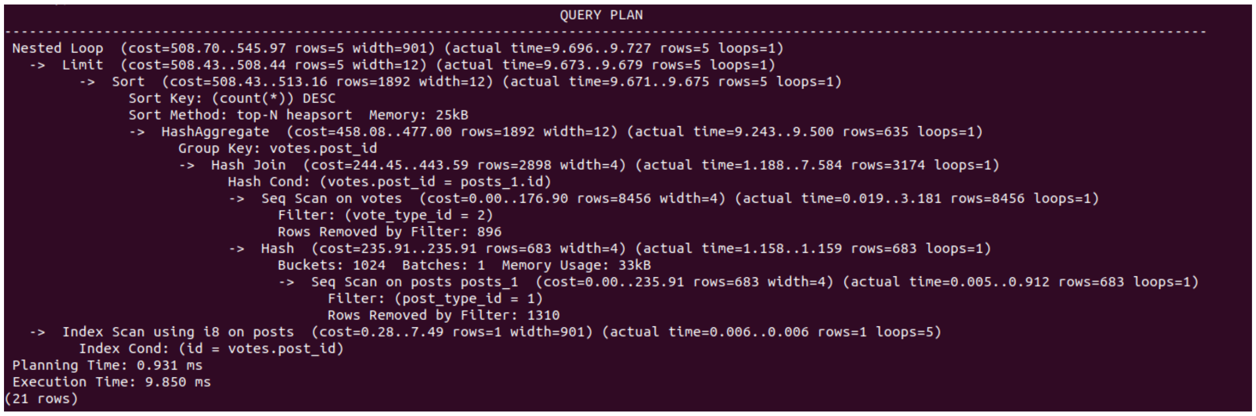
```
With t as (select post_id from votes,posts where vote_type_id=2 and
votes.post_id=posts.id and posts.post_type_id=1 group by post_id order by
count(*) desc limit 5) select * from posts where id in (select post_id
from t);
```



```
                                                            QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=508.70..545.97 rows=5 width=901) (actual time=11.572..11.611 rows=5 loops=1)
   ->  Limit  (cost=508.43..508.44 rows=5 width=12) (actual time=11.543..11.550 rows=5 loops=1)
         ->  Sort  (cost=508.43..513.16 rows=1892 width=12) (actual time=11.540..11.545 rows=5 loops=1)
               Sort Key: (count(*)) DESC
               Sort Method: top-N heapsort  Memory: 25kB
               ->  HashAggregate  (cost=458.08..477.00 rows=1892 width=12) (actual time=10.949..11.290 rows=635 loops=1)
                     Group Key: votes.post_id
                     ->  Hash Join  (cost=244.45..443.59 rows=2898 width=4) (actual time=1.433..9.084 rows=3174 loops=1)
                           Hash Cond: (votes.post_id = posts_1.id)
                           ->  Seq Scan on votes  (cost=0.00..176.90 rows=8456 width=4) (actual time=0.036..3.876 rows=8456 loops=1)
                                 Filter: (vote_type_id = 2)
                                 Rows Removed by Filter: 896
                           ->  Hash  (cost=235.91..235.91 rows=683 width=4) (actual time=1.380..1.381 rows=683 loops=1)
                                 Buckets: 1024  Batches: 1  Memory Usage: 33kB
                                 ->  Seq Scan on posts posts_1  (cost=0.00..235.91 rows=683 width=4) (actual time=0.009..1.097 rows=683 loops=1)
                                       Filter: (post_type_id = 1)
                                       Rows Removed by Filter: 1310
   ->  Index Scan using posts_pkey on posts  (cost=0.28..7.49 rows=1 width=901) (actual time=0.008..0.008 rows=1 loops=5)
         Index Cond: (id = votes.post_id)
 Planning Time: 1.004 ms
 Execution Time: 12.345 ms
(21 rows)
```

Time: 13.3 ms

AFTER INDEXING

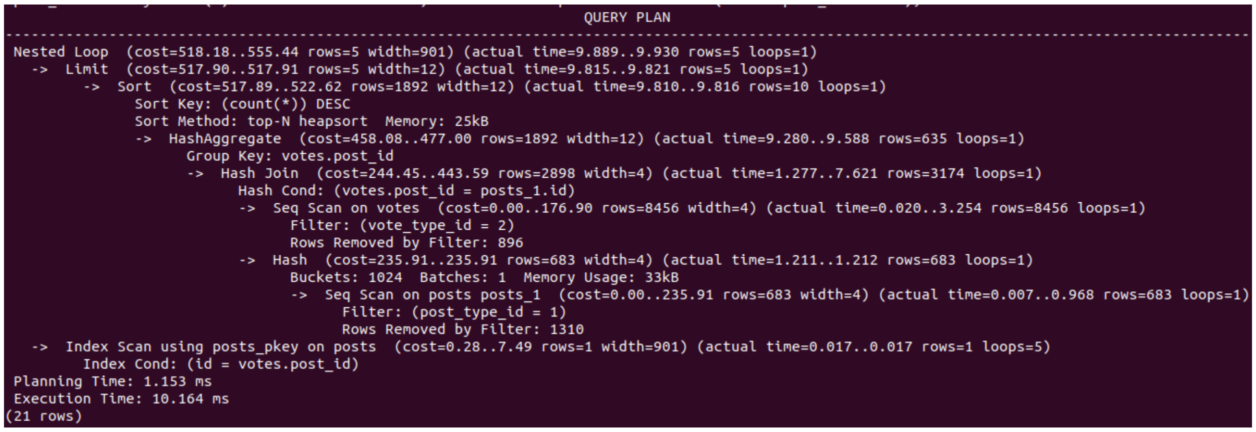


```
                                                            QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=508.70..545.97 rows=5 width=901) (actual time=9.696..9.727 rows=5 loops=1)
   ->  Limit  (cost=508.43..508.44 rows=5 width=12) (actual time=9.673..9.679 rows=5 loops=1)
         ->  Sort  (cost=508.43..513.16 rows=1892 width=12) (actual time=9.671..9.675 rows=5 loops=1)
               Sort Key: (count(*)) DESC
               Sort Method: top-N heapsort  Memory: 25kB
               ->  HashAggregate  (cost=458.08..477.00 rows=1892 width=12) (actual time=9.243..9.500 rows=635 loops=1)
                     Group Key: votes.post_id
                     ->  Hash Join  (cost=244.45..443.59 rows=2898 width=4) (actual time=1.188..7.584 rows=3174 loops=1)
                           Hash Cond: (votes.post_id = posts_1.id)
                           ->  Seq Scan on votes  (cost=0.00..176.90 rows=8456 width=4) (actual time=0.019..3.181 rows=8456 loops=1)
                                 Filter: (vote_type_id = 2)
                                 Rows Removed by Filter: 896
                           ->  Hash  (cost=235.91..235.91 rows=683 width=4) (actual time=1.158..1.159 rows=683 loops=1)
                                 Buckets: 1024  Batches: 1  Memory Usage: 33kB
                                 ->  Seq Scan on posts posts_1  (cost=0.00..235.91 rows=683 width=4) (actual time=0.005..0.912 rows=683 loops=1)
                                       Filter: (post_type_id = 1)
                                       Rows Removed by Filter: 1310
   ->  Index Scan using i8 on posts  (cost=0.28..7.49 rows=1 width=901) (actual time=0.006..0.006 rows=1 loops=5)
         Index Cond: (id = votes.post_id)
 Planning Time: 0.931 ms
 Execution Time: 9.850 ms
(21 rows)
```

Time: 10.1 ms

Query 10:
Fetch top-K=5 question posts after offset

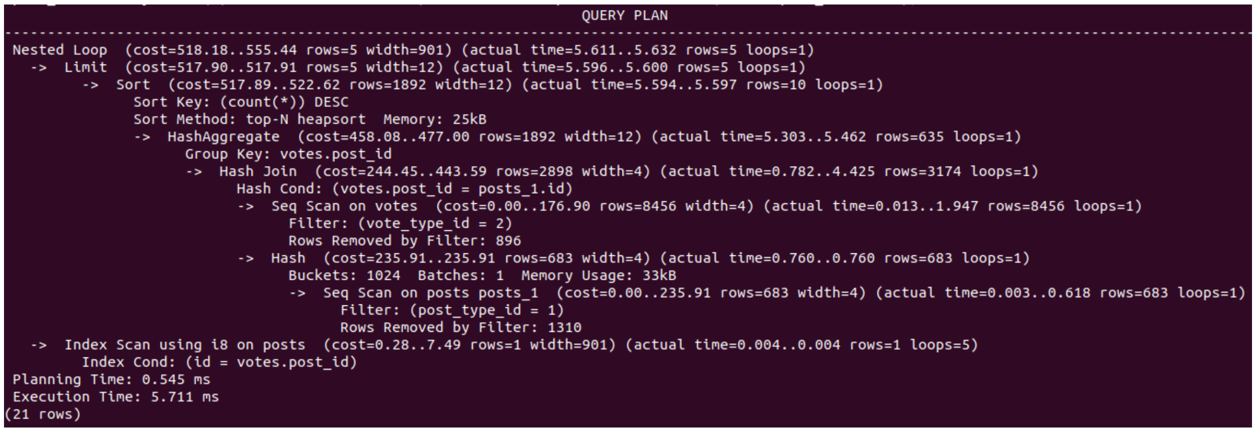
```
With t as (select post_id from votes,posts where vote_type_id=2 and
votes.post_id=posts.id and posts.post_type_id=1 group by post_id order by
count(*) desc limit 5 offset 5) select * from posts where id in (select
post_id from t);
```



```
                                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=518.18..555.44 rows=5 width=901) (actual time=9.889..9.930 rows=5 loops=1)
   ->  Limit  (cost=517.90..517.91 rows=5 width=12) (actual time=9.815..9.821 rows=5 loops=1)
         ->  Sort  (cost=517.89..522.62 rows=1892 width=12) (actual time=9.810..9.816 rows=10 loops=1)
               Sort Key: (count(*)) DESC
               Sort Method: top-N heapsort  Memory: 25kB
               ->  HashAggregate  (cost=458.08..477.00 rows=1892 width=12) (actual time=9.280..9.588 rows=635 loops=1)
                     Group Key: votes.post_id
                     ->  Hash Join  (cost=244.45..443.59 rows=2898 width=4) (actual time=1.277..7.621 rows=3174 loops=1)
                           Hash Cond: (votes.post_id = posts_1.id)
                           ->  Seq Scan on votes  (cost=0.00..176.90 rows=8456 width=4) (actual time=0.020..3.254 rows=8456 loops=1)
                                 Filter: (vote_type_id = 2)
                                 Rows Removed by Filter: 896
                           ->  Hash  (cost=235.91..235.91 rows=683 width=4) (actual time=1.211..1.212 rows=683 loops=1)
                                 Buckets: 1024  Batches: 1  Memory Usage: 33kB
                                 ->  Seq Scan on posts posts_1  (cost=0.00..235.91 rows=683 width=4) (actual time=0.007..0.968 rows=683 loops=1)
                                       Filter: (post_type_id = 1)
                                       Rows Removed by Filter: 1310
   ->  Index Scan using posts_pkey on posts  (cost=0.28..7.49 rows=1 width=901) (actual time=0.017..0.017 rows=1 loops=5)
         Index Cond: (id = votes.post_id)
 Planning Time: 1.153 ms
 Execution Time: 10.164 ms
(21 rows)
```

Time: 11.3 ms

AFTER INDEXING

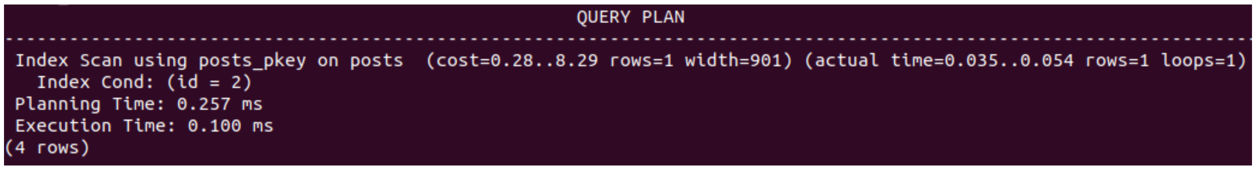


```
                                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=518.18..555.44 rows=5 width=901) (actual time=5.611..5.632 rows=5 loops=1)
   ->  Limit  (cost=517.90..517.91 rows=5 width=12) (actual time=5.596..5.600 rows=5 loops=1)
         ->  Sort  (cost=517.89..522.62 rows=1892 width=12) (actual time=5.594..5.597 rows=10 loops=1)
               Sort Key: (count(*)) DESC
               Sort Method: top-N heapsort  Memory: 25kB
               ->  HashAggregate  (cost=458.08..477.00 rows=1892 width=12) (actual time=5.303..5.462 rows=635 loops=1)
                     Group Key: votes.post_id
                     ->  Hash Join  (cost=244.45..443.59 rows=2898 width=4) (actual time=0.782..4.425 rows=3174 loops=1)
                           Hash Cond: (votes.post_id = posts_1.id)
                           ->  Seq Scan on votes  (cost=0.00..176.90 rows=8456 width=4) (actual time=0.013..1.947 rows=8456 loops=1)
                                 Filter: (vote_type_id = 2)
                                 Rows Removed by Filter: 896
                           ->  Hash  (cost=235.91..235.91 rows=683 width=4) (actual time=0.760..0.760 rows=683 loops=1)
                                 Buckets: 1024  Batches: 1  Memory Usage: 33kB
                                 ->  Seq Scan on posts posts_1  (cost=0.00..235.91 rows=683 width=4) (actual time=0.003..0.618 rows=683 loops=1)
                                       Filter: (post_type_id = 1)
                                       Rows Removed by Filter: 1310
   ->  Index Scan using i8 on posts  (cost=0.28..7.49 rows=1 width=901) (actual time=0.004..0.004 rows=1 loops=5)
         Index Cond: (id = votes.post_id)
 Planning Time: 0.545 ms
 Execution Time: 5.711 ms
(21 rows)
```

Time: 6.2 ms

Query 11:
Fetch post info given post id

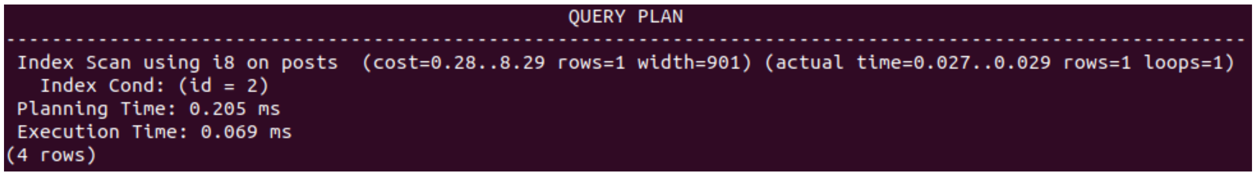
```sql
Select * from POSTS where POSTS.id=2;
```



```
                                                            QUERY PLAN
----------------------------------------------------------------------------------------------------------------------
 Index Scan using posts_pkey on posts  (cost=0.28..8.29 rows=1 width=901) (actual time=0.035..0.054 rows=1 loops=1)
   Index Cond: (id = 2)
 Planning Time: 0.257 ms
 Execution Time: 0.100 ms
(4 rows)
```

Time: 0.357 ms

AFTER INDEXING



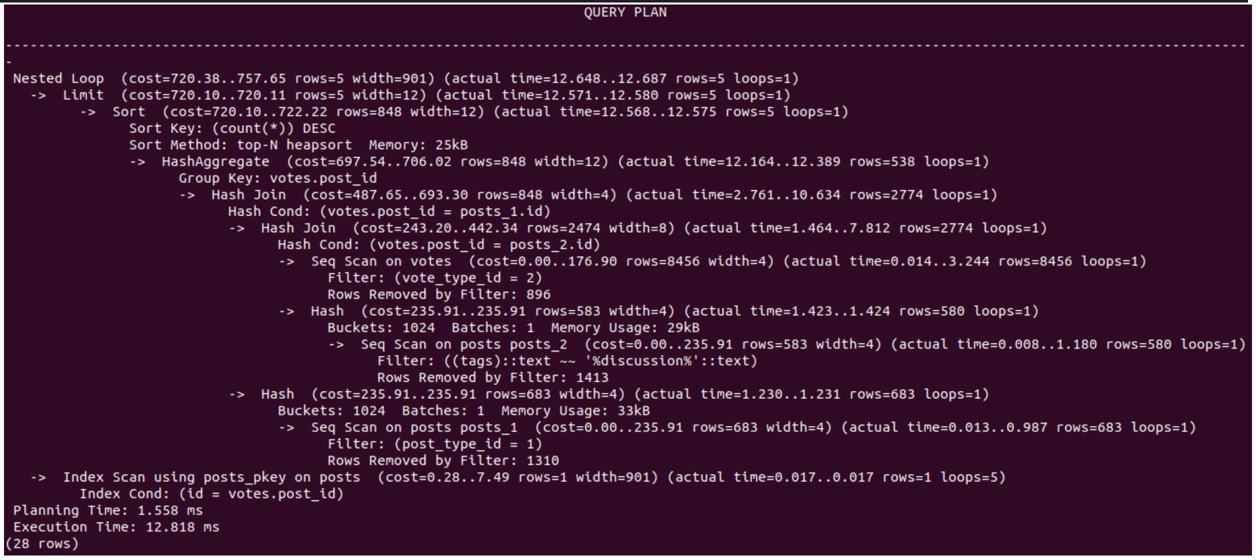
```
                                                       QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Index Scan using i8 on posts  (cost=0.28..8.29 rows=1 width=901) (actual time=0.027..0.029 rows=1 loops=1)
   Index Cond: (id = 2)
 Planning Time: 0.205 ms
 Execution Time: 0.069 ms
(4 rows)
```

Time: 0.27 ms

Query 12:

Retrieve top-k=5 posts having a certain tag

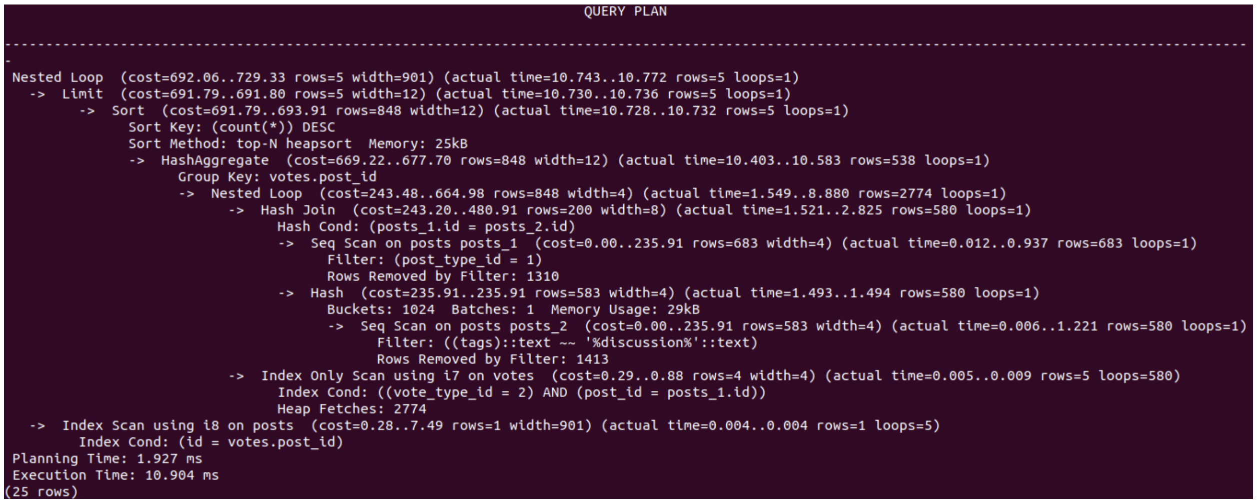
```
With t as (select id from posts where tags like '%discussion%'), t1 as
(select post_id,count(*)  as cnt from votes,posts where vote_type_id=2 and
post_id in (select id from t) and votes.post_id=posts.id and
posts.post_type_id=1 group by post_id order by cnt desc limit 5) select *
from posts where id in (select post_id from t1);
```



```
                                                                        QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------
-
 Nested Loop  (cost=720.38..757.65 rows=5 width=901) (actual time=12.648..12.687 rows=5 loops=1)
   ->  Limit  (cost=720.10..720.11 rows=5 width=12) (actual time=12.571..12.580 rows=5 loops=1)
         ->  Sort  (cost=720.10..722.22 rows=848 width=12) (actual time=12.568..12.575 rows=5 loops=1)
               Sort Key: (count(*)) DESC
               Sort Method: top-N heapsort  Memory: 25kB
               ->  HashAggregate  (cost=697.54..706.02 rows=848 width=12) (actual time=12.164..12.389 rows=538 loops=1)
                     Group Key: votes.post_id
                     ->  Hash Join  (cost=487.65..693.30 rows=848 width=4) (actual time=2.761..10.634 rows=2774 loops=1)
                           Hash Cond: (votes.post_id = posts_1.id)
                           ->  Hash Join  (cost=243.20..442.34 rows=2474 width=8) (actual time=1.464..7.812 rows=2774 loops=1)
                                 Hash Cond: (votes.post_id = posts_2.id)
                                 ->  Seq Scan on votes  (cost=0.00..176.90 rows=8456 width=4) (actual time=0.014..3.244 rows=8456 loops=1)
                                       Filter: (vote_type_id = 2)
                                       Rows Removed by Filter: 896
                                 ->  Hash  (cost=235.91..235.91 rows=583 width=4) (actual time=1.423..1.424 rows=580 loops=1)
                                       Buckets: 1024  Batches: 1  Memory Usage: 29kB
                                       ->  Seq Scan on posts posts_2  (cost=0.00..235.91 rows=583 width=4) (actual time=0.008..1.180 rows=580 loops=1)
                                             Filter: ((tags)::text ~~ '%discussion%'::text)
                                             Rows Removed by Filter: 1413
                           ->  Hash  (cost=235.91..235.91 rows=683 width=4) (actual time=1.230..1.231 rows=683 loops=1)
                                 Buckets: 1024  Batches: 1  Memory Usage: 33kB
                                 ->  Seq Scan on posts posts_1  (cost=0.00..235.91 rows=683 width=4) (actual time=0.013..0.987 rows=683 loops=1)
                                       Filter: (post_type_id = 1)
                                       Rows Removed by Filter: 1310
   ->  Index Scan using posts_pkey on posts  (cost=0.28..7.49 rows=1 width=901) (actual time=0.017..0.017 rows=1 loops=5)
         Index Cond: (id = votes.post_id)
 Planning Time: 1.558 ms
 Execution Time: 12.818 ms
(28 rows)
```

Time: 14.3 ms

AFTER INDEXING

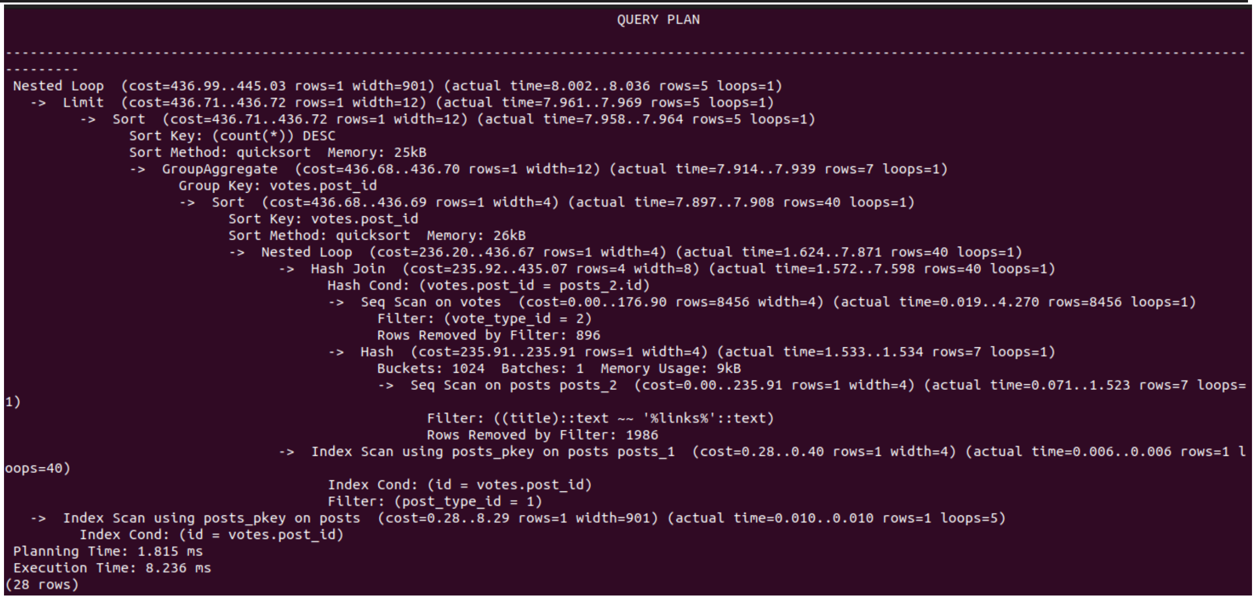


```
                                                                        QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------
-
 Nested Loop  (cost=692.06..729.33 rows=5 width=901) (actual time=10.743..10.772 rows=5 loops=1)
   ->  Limit  (cost=691.79..691.80 rows=5 width=12) (actual time=10.730..10.736 rows=5 loops=1)
         ->  Sort  (cost=691.79..693.91 rows=848 width=12) (actual time=10.728..10.732 rows=5 loops=1)
               Sort Key: (count(*)) DESC
               Sort Method: top-N heapsort  Memory: 25kB
               ->  HashAggregate  (cost=669.22..677.70 rows=848 width=12) (actual time=10.403..10.583 rows=538 loops=1)
                     Group Key: votes.post_id
                     ->  Nested Loop  (cost=243.48..664.98 rows=848 width=4) (actual time=1.549..8.880 rows=2774 loops=1)
                           ->  Hash Join  (cost=243.20..480.91 rows=200 width=8) (actual time=1.521..2.825 rows=580 loops=1)
                                 Hash Cond: (posts_1.id = posts_2.id)
                                 ->  Seq Scan on posts posts_1  (cost=0.00..235.91 rows=683 width=4) (actual time=0.012..0.937 rows=683 loops=1)
                                       Filter: (post_type_id = 1)
                                       Rows Removed by Filter: 1310
                                 ->  Hash  (cost=235.91..235.91 rows=583 width=4) (actual time=1.493..1.494 rows=580 loops=1)
                                       Buckets: 1024  Batches: 1  Memory Usage: 29kB
                                       ->  Seq Scan on posts posts_2  (cost=0.00..235.91 rows=583 width=4) (actual time=0.006..1.221 rows=580 loops=1)
                                             Filter: ((tags)::text ~~ '%discussion%'::text)
                                             Rows Removed by Filter: 1413
                           ->  Index Only Scan using i7 on votes  (cost=0.29..0.88 rows=4 width=4) (actual time=0.005..0.009 rows=5 loops=580)
                                 Index Cond: ((vote_type_id = 2) AND (post_id = posts_1.id))
                                 Heap Fetches: 2774
   ->  Index Scan using i8 on posts  (cost=0.28..7.49 rows=1 width=901) (actual time=0.004..0.004 rows=1 loops=5)
         Index Cond: (id = votes.post_id)
 Planning Time: 1.927 ms
 Execution Time: 10.904 ms
(25 rows)
```

Time: 12.8 ms

Query 13:
Retrieve top-K=5 question posts having with certain keyword

```
with t as (select id from posts where title like '%links%'), t1 as (select
post_id,count(*)  as cnt from votes,posts where vote_type_id=2 and post_id
in (select id from t) and votes.post_id=posts.id and posts.post_type_id=1
group by post_id order by cnt desc limit 5) select * from posts where id
in (select post_id from t1);
```

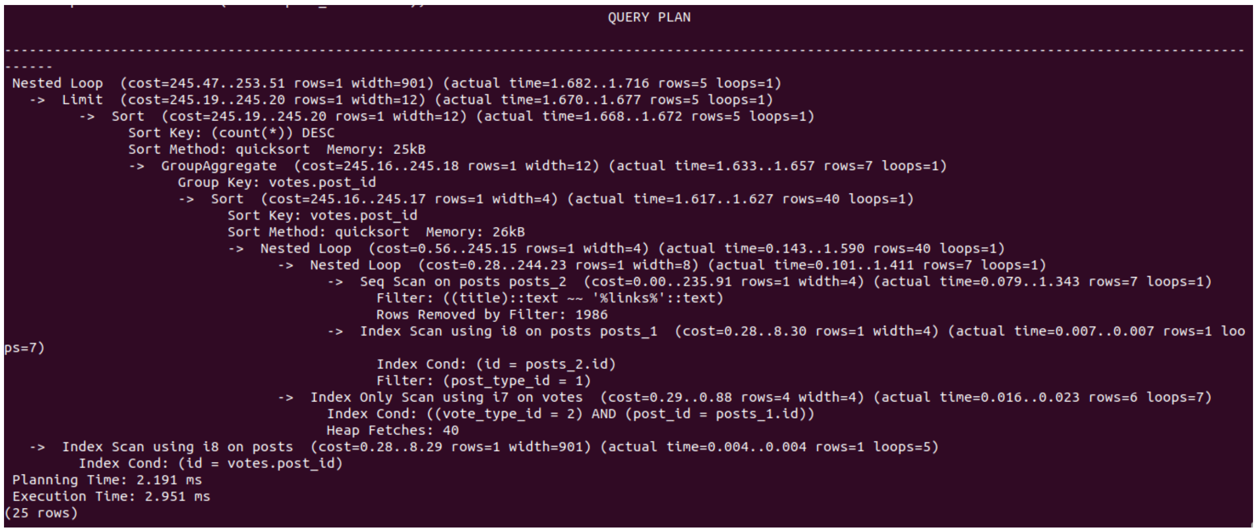


```
                                                                QUERY PLAN

------------------------------------------------------------------------------------------------------------------------------------------------------
---------
 Nested Loop  (cost=436.99..445.03 rows=1 width=901) (actual time=8.002..8.036 rows=5 loops=1)
   ->  Limit  (cost=436.71..436.72 rows=1 width=12) (actual time=7.961..7.969 rows=5 loops=1)
         ->  Sort  (cost=436.71..436.72 rows=1 width=12) (actual time=7.958..7.964 rows=5 loops=1)
               Sort Key: (count(*)) DESC
               Sort Method: quicksort  Memory: 25kB
               ->  GroupAggregate  (cost=436.68..436.70 rows=1 width=12) (actual time=7.914..7.939 rows=7 loops=1)
                     Group Key: votes.post_id
                     ->  Sort  (cost=436.68..436.69 rows=1 width=4) (actual time=7.897..7.908 rows=40 loops=1)
                           Sort Key: votes.post_id
                           Sort Method: quicksort  Memory: 26kB
                           ->  Nested Loop  (cost=236.20..436.67 rows=1 width=4) (actual time=1.624..7.871 rows=40 loops=1)
                                 ->  Hash Join  (cost=235.92..435.07 rows=4 width=8) (actual time=1.572..7.598 rows=40 loops=1)
                                       Hash Cond: (votes.post_id = posts_2.id)
                                       ->  Seq Scan on votes  (cost=0.00..176.90 rows=8456 width=4) (actual time=0.019..4.270 rows=8456 loops=1)
                                             Filter: (vote_type_id = 2)
                                             Rows Removed by Filter: 896
                                       ->  Hash  (cost=235.91..235.91 rows=1 width=4) (actual time=1.533..1.534 rows=7 loops=1)
                                             Buckets: 1024  Batches: 1  Memory Usage: 9kB
                                             ->  Seq Scan on posts posts_2  (cost=0.00..235.91 rows=1 width=4) (actual time=0.071..1.523 rows=7 loops=
1)
                                                   Filter: ((title)::text ~~ '%links%'::text)
                                                   Rows Removed by Filter: 1986
                                 ->  Index Scan using posts_pkey on posts posts_1  (cost=0.28..0.40 rows=1 width=4) (actual time=0.006..0.006 rows=1 l
oops=40)
                                       Index Cond: (id = votes.post_id)
                                       Filter: (post_type_id = 1)
   ->  Index Scan using posts_pkey on posts  (cost=0.28..8.29 rows=1 width=901) (actual time=0.010..0.010 rows=1 loops=5)
         Index Cond: (id = votes.post_id)
 Planning Time: 1.815 ms
 Execution Time: 8.236 ms
(28 rows)
```

Time: 10.1 ms

AFTER INDEXING



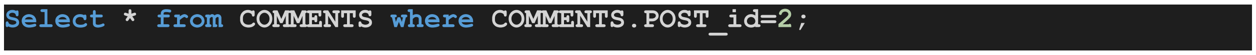
```
                                                                QUERY PLAN

------------------------------------------------------------------------------------------------------------------------------------------------------
------
 Nested Loop  (cost=245.47..253.51 rows=1 width=901) (actual time=1.682..1.716 rows=5 loops=1)
   ->  Limit  (cost=245.19..245.20 rows=1 width=12) (actual time=1.670..1.677 rows=5 loops=1)
         ->  Sort  (cost=245.19..245.20 rows=1 width=12) (actual time=1.668..1.672 rows=5 loops=1)
               Sort Key: (count(*)) DESC
               Sort Method: quicksort  Memory: 25kB
               ->  GroupAggregate  (cost=245.16..245.18 rows=1 width=12) (actual time=1.633..1.657 rows=7 loops=1)
                     Group Key: votes.post_id
                     ->  Sort  (cost=245.16..245.17 rows=1 width=4) (actual time=1.617..1.627 rows=40 loops=1)
                           Sort Key: votes.post_id
                           Sort Method: quicksort  Memory: 26kB
                           ->  Nested Loop  (cost=0.56..245.15 rows=1 width=4) (actual time=0.143..1.590 rows=40 loops=1)
                                 ->  Nested Loop  (cost=0.28..244.23 rows=1 width=8) (actual time=0.101..1.411 rows=7 loops=1)
                                       ->  Seq Scan on posts posts_2  (cost=0.00..235.91 rows=1 width=4) (actual time=0.079..1.343 rows=7 loops=1)
                                             Filter: ((title)::text ~~ '%links%'::text)
                                             Rows Removed by Filter: 1986
                                       ->  Index Scan using i8 on posts posts_1  (cost=0.28..8.30 rows=1 width=4) (actual time=0.007..0.007 rows=1 loo
ps=7)
                                             Index Cond: (id = posts_2.id)
                                             Filter: (post_type_id = 1)
                                 ->  Index Only Scan using i7 on votes  (cost=0.29..0.88 rows=4 width=4) (actual time=0.016..0.023 rows=6 loops=7)
                                       Index Cond: ((vote_type_id = 2) AND (post_id = posts_1.id))
                                       Heap Fetches: 40
   ->  Index Scan using i8 on posts  (cost=0.28..8.29 rows=1 width=901) (actual time=0.004..0.004 rows=1 loops=5)
         Index Cond: (id = votes.post_id)
 Planning Time: 2.191 ms
 Execution Time: 2.951 ms
(25 rows)
```

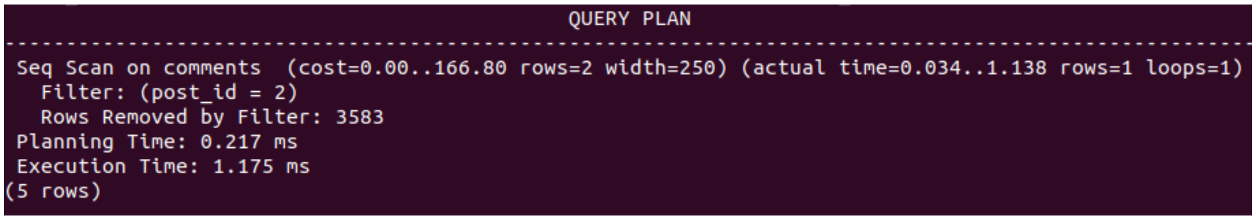
Time: 5.1 ms

Query 14:

Comments pertaining to a post



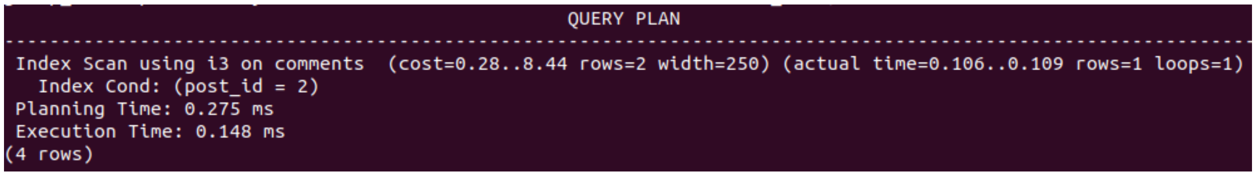
```
Select * from COMMENTS where COMMENTS.POST_id=2;
```



```
                                                 QUERY PLAN
--------------------------------------------------------------------------------------------------------
 Seq Scan on comments  (cost=0.00..166.80 rows=2 width=250) (actual time=0.034..1.138 rows=1 loops=1)
   Filter: (post_id = 2)
   Rows Removed by Filter: 3583
 Planning Time: 0.217 ms
 Execution Time: 1.175 ms
(5 rows)
```

Time:1.390 ms

AFTER INDEXING

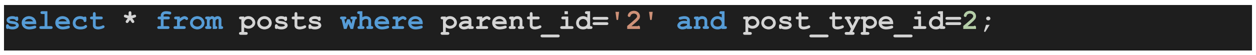


```
                                                 QUERY PLAN
----------------------------------------------------------------------------------------------------------------
 Index Scan using i3 on comments  (cost=0.28..8.44 rows=2 width=250) (actual time=0.106..0.109 rows=1 loops=1)
   Index Cond: (post_id = 2)
 Planning Time: 0.275 ms
 Execution Time: 0.148 ms
(4 rows)
```

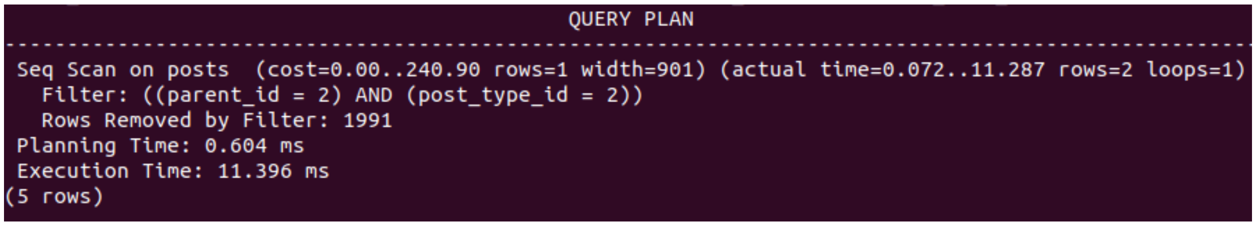
Time: 0.41 ms

Query 15:

Retrieving answer posts given question post id



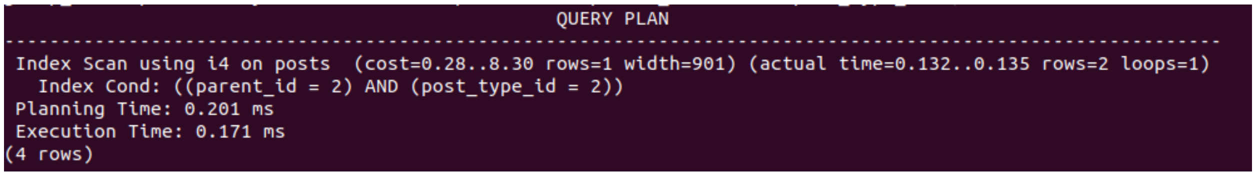
```
select * from posts where parent_id='2' and post_type_id=2;
```



```
                                                 QUERY PLAN
--------------------------------------------------------------------------------------------------------
 Seq Scan on posts  (cost=0.00..240.90 rows=1 width=901) (actual time=0.072..11.287 rows=2 loops=1)
   Filter: ((parent_id = 2) AND (post_type_id = 2))
   Rows Removed by Filter: 1991
 Planning Time: 0.604 ms
 Execution Time: 11.396 ms
(5 rows)
```

Time:12 ms

AFTER INDEXING

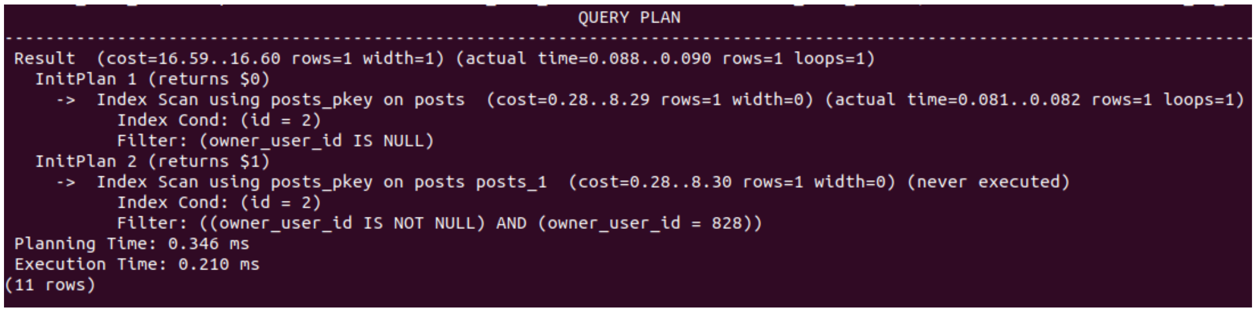


```
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Index Scan using i4 on posts  (cost=0.28..8.30 rows=1 width=901) (actual time=0.132..0.135 rows=2 loops=1)
   Index Cond: ((parent_id = 2) AND (post_type_id = 2))
 Planning Time: 0.201 ms
 Execution Time: 0.171 ms
(4 rows)
```

Time: 0.372 ms

Query 16:
Checking user who post belongs to for allowing editing

```
Select case when exists (Select owner_user_id from posts where id=2 and
owner_user_id is NULL) then False when exists (Select owner_user_id from
posts where id=2 and owner_user_id is not NULL and owner_user_id=828) then
True end as allowed_to_edit;
```

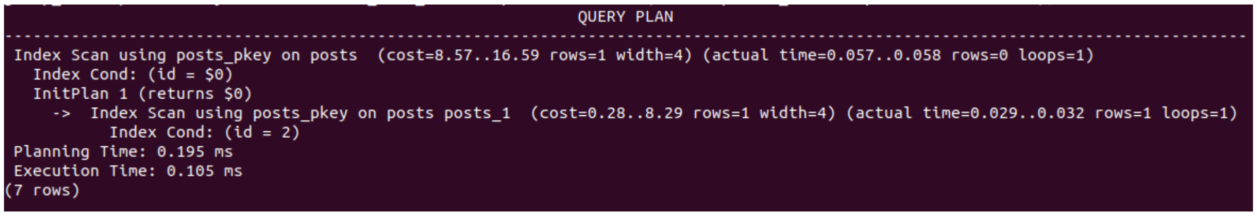


```
                                                         QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Result  (cost=16.59..16.60 rows=1 width=1) (actual time=0.088..0.090 rows=1 loops=1)
   InitPlan 1 (returns $0)
     ->  Index Scan using posts_pkey on posts  (cost=0.28..8.29 rows=1 width=0) (actual time=0.081..0.082 rows=1 loops=1)
           Index Cond: (id = 2)
           Filter: (owner_user_id IS NULL)
   InitPlan 2 (returns $1)
     ->  Index Scan using posts_pkey on posts posts_1  (cost=0.28..8.30 rows=1 width=0) (never executed)
           Index Cond: (id = 2)
           Filter: ((owner_user_id IS NOT NULL) AND (owner_user_id = 828))
 Planning Time: 0.346 ms
 Execution Time: 0.210 ms
(11 rows)
```

Time: 0.556 ms

Query 17:
Retrieving user_id of question owner

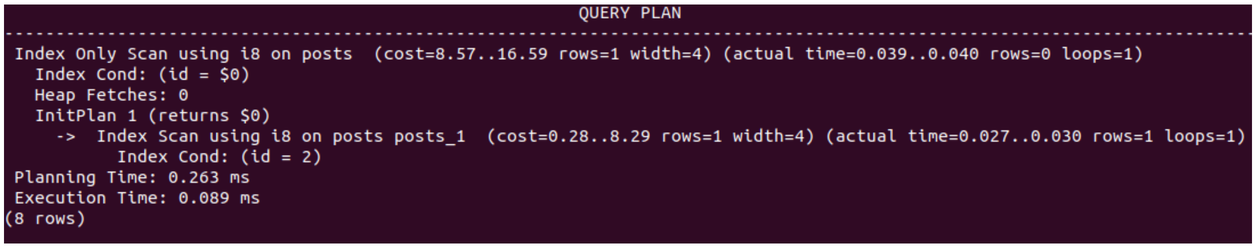
```
Select owner_user_id from posts where id=(Select parent_id from posts
where id='2')
```



```
                                                         QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Index Scan using posts_pkey on posts  (cost=8.57..16.59 rows=1 width=4) (actual time=0.057..0.058 rows=0 loops=1)
   Index Cond: (id = $0)
   InitPlan 1 (returns $0)
     ->  Index Scan using posts_pkey on posts posts_1  (cost=0.28..8.29 rows=1 width=4) (actual time=0.029..0.032 rows=1 loops=1)
           Index Cond: (id = 2)
 Planning Time: 0.195 ms
 Execution Time: 0.105 ms
(7 rows)
```

Time:0.3 ms

AFTER INDEXING



```
                                                         QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Index Only Scan using i8 on posts  (cost=8.57..16.59 rows=1 width=4) (actual time=0.039..0.040 rows=0 loops=1)
   Index Cond: (id = $0)
   Heap Fetches: 0
   InitPlan 1 (returns $0)
     ->  Index Scan using i8 on posts posts_1  (cost=0.28..8.29 rows=1 width=4) (actual time=0.027..0.030 rows=1 loops=1)
           Index Cond: (id = 2)
 Planning Time: 0.263 ms
 Execution Time: 0.089 ms
(8 rows)
```

Time: 0.3 ms