Indexes

trainers.py, class_types.py

- Since the trainers and class_types tables will always be relatively small (under 50 rows), and the transactions for the following endpoints do not join with other tables, GET /trainers/{id}, GET /trainers/, and GET /class-types/ do not need indexes.

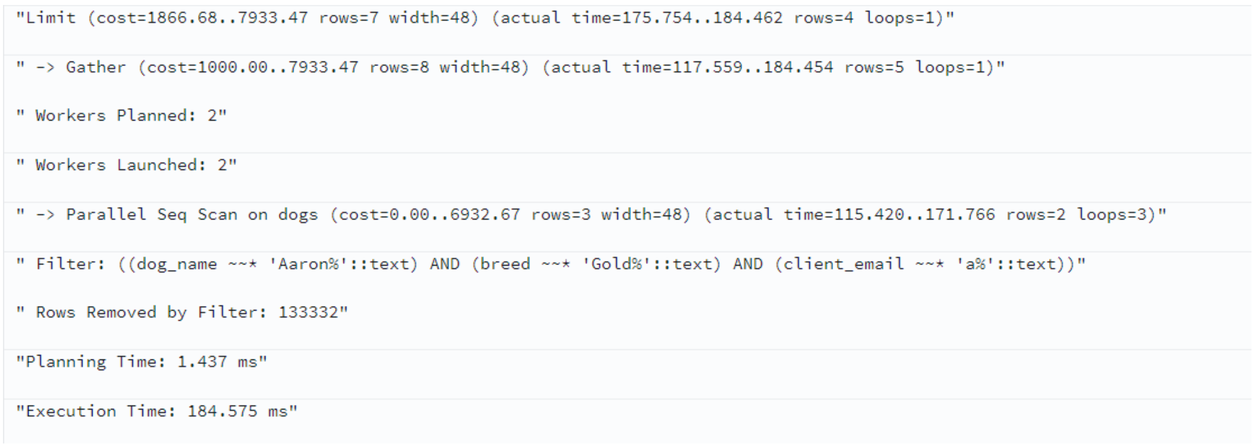
dogs.py

GET /dogs/

Running **Query**:

```
EXPLAIN ANALYZE
SELECT dog_id, dog_name, birthday, breed, client_email
FROM dogs
WHERE dog_name LIKE lower('Aaron%')
AND breed LIKE lower('Gold%')
AND client_email LIKE lower('a%')
OFFSET 1
LIMIT 40
```

Before adding the index, execution time was 184.575 ms:



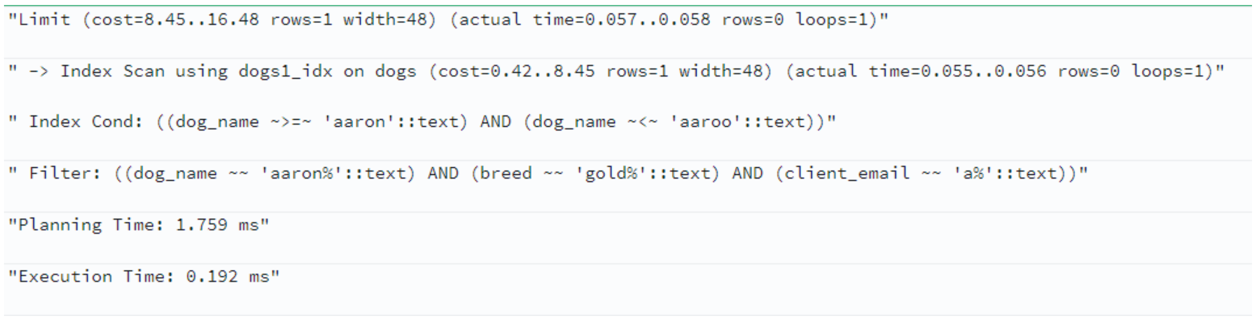
```
"Limit (cost=1866.68..7933.47 rows=7 width=48) (actual time=175.754..184.462 rows=4 loops=1)"
" -> Gather (cost=1000.00..7933.47 rows=8 width=48) (actual time=117.559..184.454 rows=5 loops=1)"
" Workers Planned: 2"
" Workers Launched: 2"
" -> Parallel Seq Scan on dogs (cost=0.00..6932.67 rows=3 width=48) (actual time=115.420..171.766 rows=2 loops=3)"
" Filter: ((dog_name ~~* 'Aaron%'::text) AND (breed ~~* 'Gold%'::text) AND (client_email ~~* 'a%'::text))"
" Rows Removed by Filter: 133332"
"Planning Time: 1.437 ms"
"Execution Time: 184.575 ms"
```

Index added:

```
CREATE INDEX dogs_idx ON dogs (lower(dog_name) text_pattern_ops, lower(breed)
text_pattern_ops,  lower(client_email) text_pattern_ops);
```

We included this index because we wanted to select certain information from the dogs table and filter based on multiple columns.

After adding the index, execution time was 0.192 ms:



```
"Limit (cost=8.45..16.48 rows=1 width=48) (actual time=0.057..0.058 rows=0 loops=1)"
" -> Index Scan using dogs1_idx on dogs (cost=0.42..8.45 rows=1 width=48) (actual time=0.055..0.056 rows=0 loops=1)"
" Index Cond: ((dog_name ~>=~ 'aaron'::text) AND (dog_name ~<~ 'aaroo'::text))"
" Filter: ((dog_name ~~ 'aaron%'::text) AND (breed ~~ 'gold%'::text) AND (client_email ~~ 'a%'::text))"
"Planning Time: 1.759 ms"
"Execution Time: 0.192 ms"
```

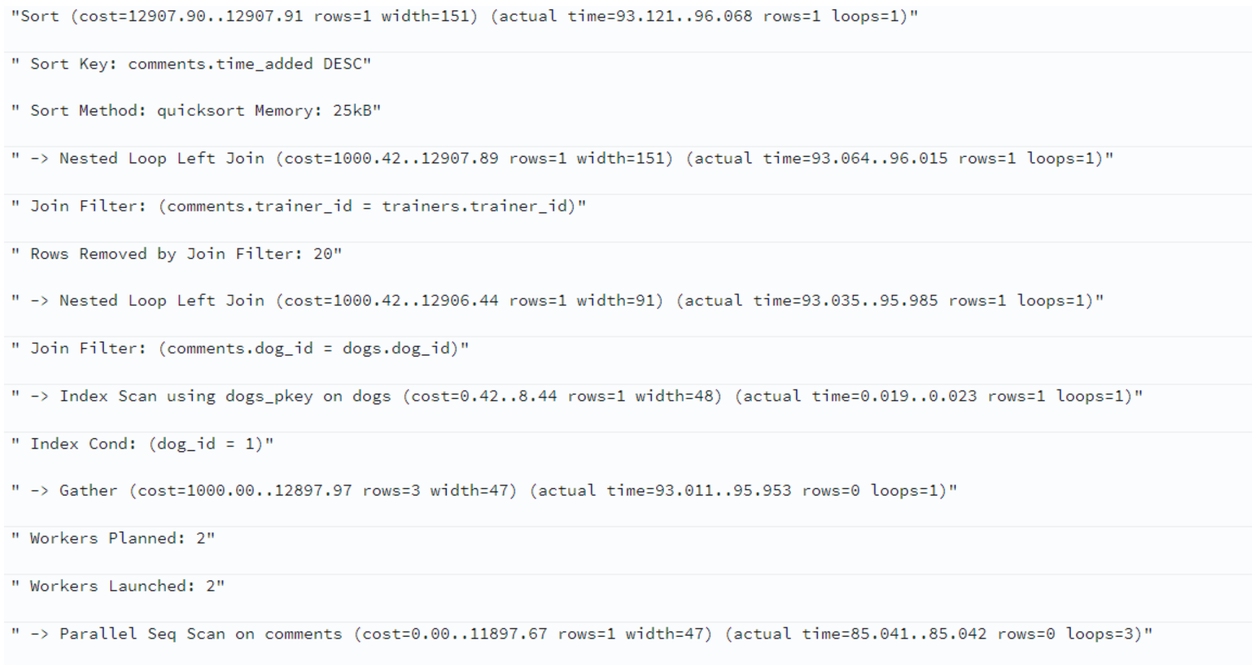
Because the cost was so much lower and the time was faster, we decided to keep the index.

GET /dogs/{id}

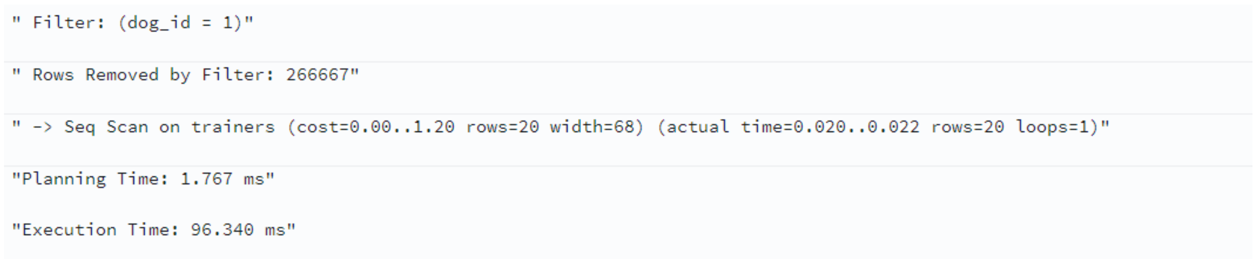
Running **Query**:

```
EXPLAIN ANALYZE
SELECT dogs.dog_id, dogs.dog_name, dogs.client_email,
    dogs.birthday, dogs.breed, comments.comment_id,
    comments.time_added, comments.comment_text,
    trainers.first_name, trainers.last_name
FROM dogs
LEFT JOIN comments on comments.dog_id = dogs.dog_id
LEFT JOIN trainers on comments.trainer_id = trainers.trainer_id
WHERE dogs.dog_id = 1
ORDER BY comments.time_added desc
```

Before adding the index, execution time was 96.34 ms:



```
"Sort (cost=12907.90..12907.91 rows=1 width=151) (actual time=93.121..96.068 rows=1 loops=1)"
" Sort Key: comments.time_added DESC"
" Sort Method: quicksort Memory: 25kB"
" -> Nested Loop Left Join (cost=1000.42..12907.89 rows=1 width=151) (actual time=93.064..96.015 rows=1 loops=1)"
" Join Filter: (comments.trainer_id = trainers.trainer_id)"
" Rows Removed by Join Filter: 20"
" -> Nested Loop Left Join (cost=1000.42..12906.44 rows=1 width=91) (actual time=93.035..95.985 rows=1 loops=1)"
" Join Filter: (comments.dog_id = dogs.dog_id)"
" -> Index Scan using dogs_pkey on dogs (cost=0.42..8.44 rows=1 width=48) (actual time=0.019..0.023 rows=1 loops=1)"
" Index Cond: (dog_id = 1)"
" -> Gather (cost=1000.00..12897.97 rows=3 width=47) (actual time=93.011..95.953 rows=0 loops=1)"
" Workers Planned: 2"
" Workers Launched: 2"
" -> Parallel Seq Scan on comments (cost=0.00..11897.67 rows=1 width=47) (actual time=85.041..85.042 rows=0 loops=3)"
```



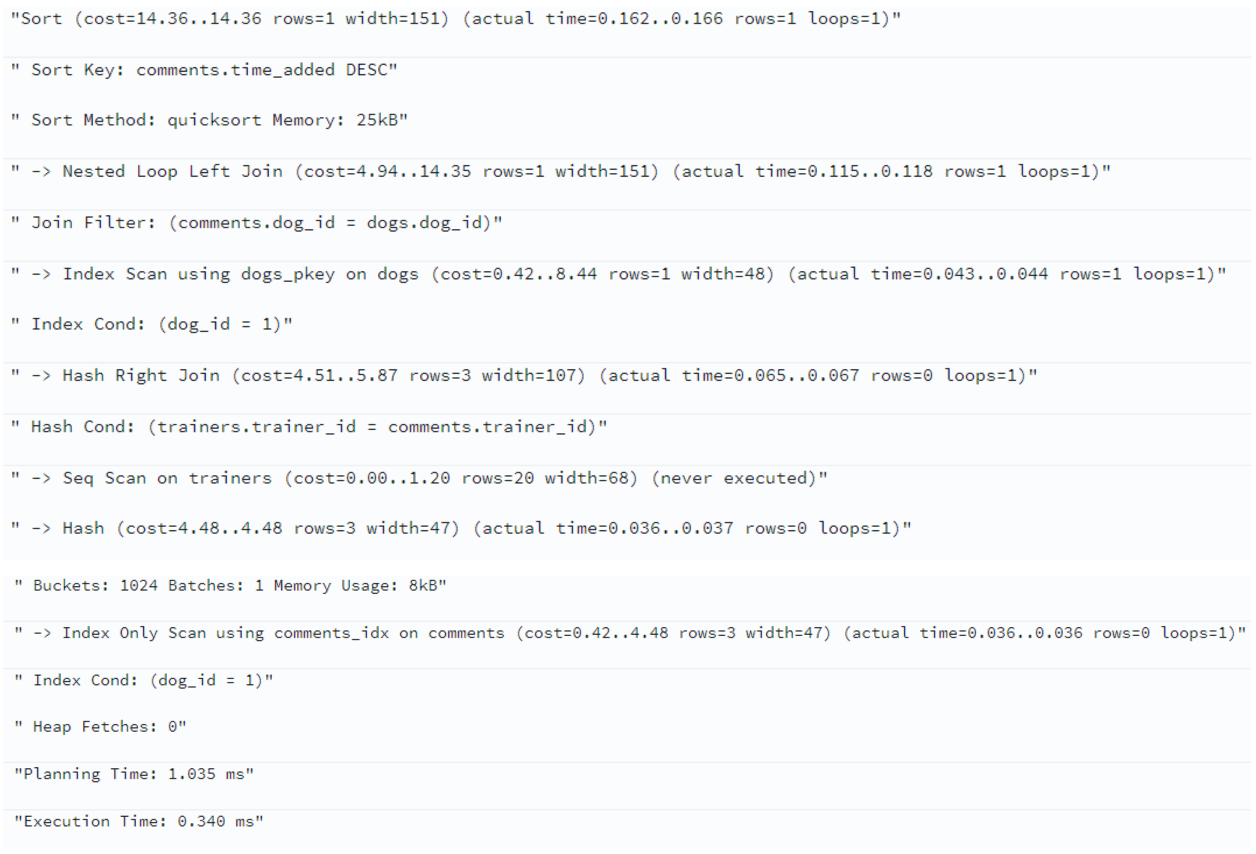
```
" Filter: (dog_id = 1)"
" Rows Removed by Filter: 266667"
" -> Seq Scan on trainers (cost=0.00..1.20 rows=20 width=68) (actual time=0.020..0.022 rows=20 loops=1)"
"Planning Time: 1.767 ms"
"Execution Time: 96.340 ms"
```

Index added:

```
CREATE INDEX comments_idx ON comments (dog_id, trainer_id, time_added,
comment_text, comment_id);
```

We included this index because we wanted to select certain information from comments and trainers tables for a specific dog. Adding an index that allowed us to quickly look up the comments associated with the dog_id and trainers that are associated with those comments without having to do a full table scan. We don't need an index for the trainer's table because there are only 20 rows.

After adding the index, execution time was 0.34 ms:



```
"Sort (cost=14.36..14.36 rows=1 width=151) (actual time=0.162..0.166 rows=1 loops=1)"
" Sort Key: comments.time_added DESC"
" Sort Method: quicksort Memory: 25kB"
" -> Nested Loop Left Join (cost=4.94..14.35 rows=1 width=151) (actual time=0.115..0.118 rows=1 loops=1)"
" Join Filter: (comments.dog_id = dogs.dog_id)"
" -> Index Scan using dogs_pkey on dogs (cost=0.42..8.44 rows=1 width=48) (actual time=0.043..0.044 rows=1 loops=1)"
" Index Cond: (dog_id = 1)"
" -> Hash Right Join (cost=4.51..5.87 rows=3 width=107) (actual time=0.065..0.067 rows=0 loops=1)"
" Hash Cond: (trainers.trainer_id = comments.trainer_id)"
" -> Seq Scan on trainers (cost=0.00..1.20 rows=20 width=68) (never executed)"
" -> Hash (cost=4.48..4.48 rows=3 width=47) (actual time=0.036..0.037 rows=0 loops=1)"
" Buckets: 1024 Batches: 1 Memory Usage: 8kB"
" -> Index Only Scan using comments_idx on comments (cost=0.42..4.48 rows=3 width=47) (actual time=0.036..0.036 rows=0 loops=1)"
" Index Cond: (dog_id = 1)"
" Heap Fetches: 0"
"Planning Time: 1.035 ms"
"Execution Time: 0.340 ms"
```

Because the cost was so much lower and the time was faster, we decided to keep the index.

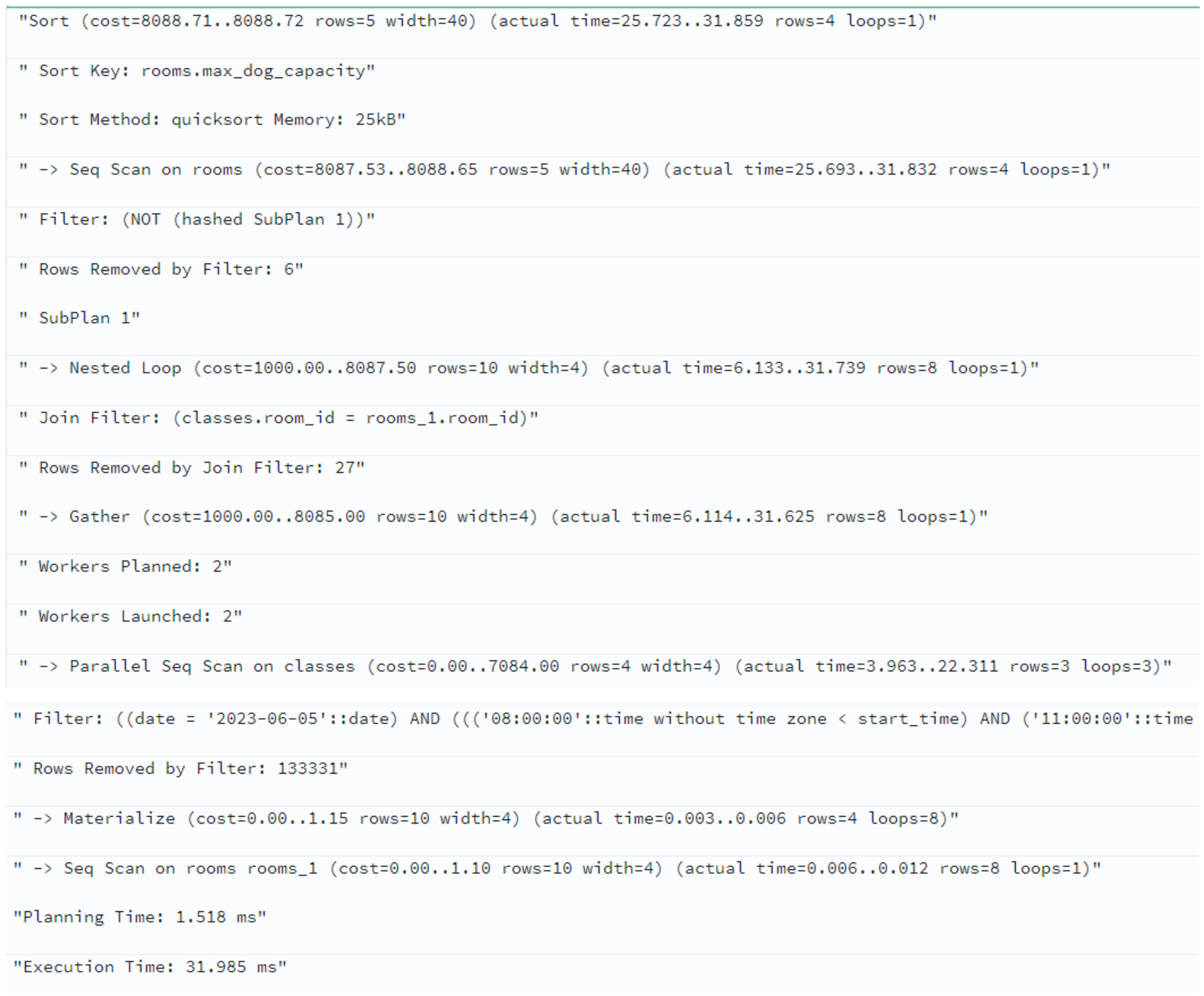
rooms.py

GET /rooms/

Running **Query**:

```
EXPLAIN ANALYZE
SELECT room_id, max_dog_capacity, room_name
FROM rooms
WHERE room_id NOT IN (
    SELECT classes.room_id
    FROM classes
    JOIN rooms ON classes.room_id = rooms.room_id
    WHERE classes.date = '2023-06-05' AND
        ((CAST('8:00:00 AM' AS TIME) < classes.start_time
            AND CAST('11:00:00 AM' AS TIME) > classes.start_time)
        OR (CAST('8:00:00 AM' AS TIME) < classes.end_time
            AND CAST('11:00:00 AM' AS TIME) > classes.end_time))
)
ORDER BY max_dog_capacity ASC
```

Before adding the index, execution time was 31.985 ms:



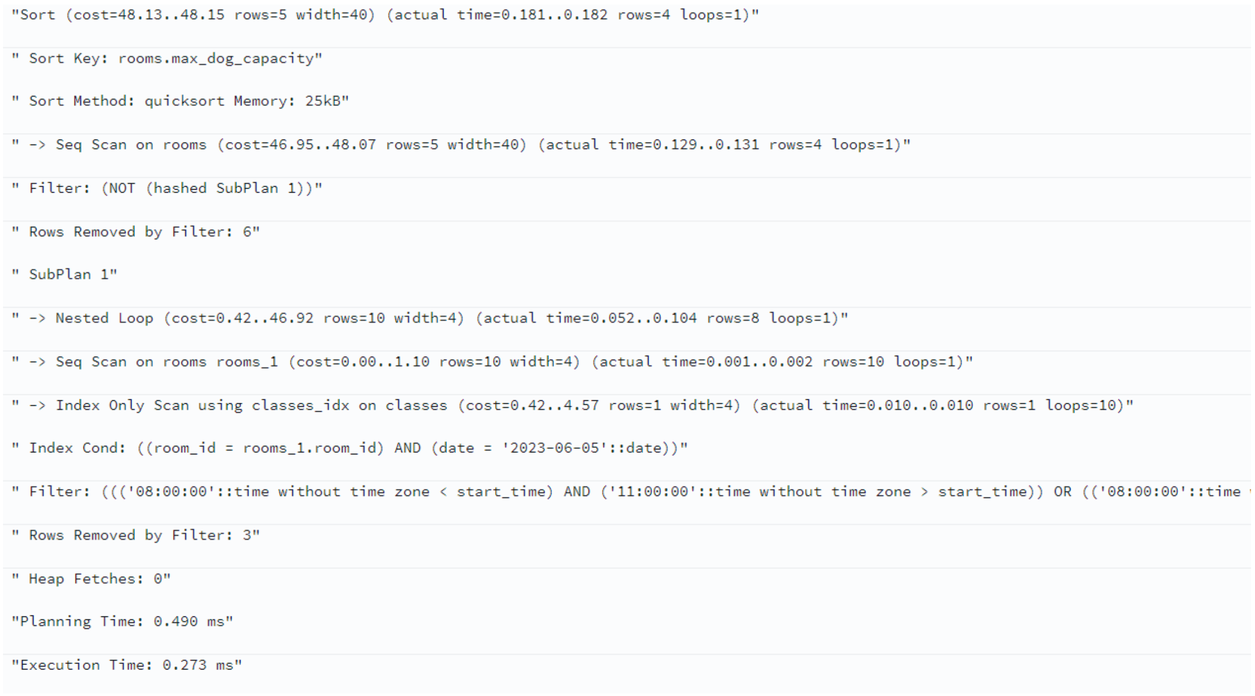
```
"Sort (cost=8088.71..8088.72 rows=5 width=40) (actual time=25.723..31.859 rows=4 loops=1)"
" Sort Key: rooms.max_dog_capacity"
" Sort Method: quicksort Memory: 25kB"
" -> Seq Scan on rooms (cost=8087.53..8088.65 rows=5 width=40) (actual time=25.693..31.832 rows=4 loops=1)"
" Filter: (NOT (hashed SubPlan 1))"
" Rows Removed by Filter: 6"
" SubPlan 1"
" -> Nested Loop (cost=1000.00..8087.50 rows=10 width=4) (actual time=6.133..31.739 rows=8 loops=1)"
" Join Filter: (classes.room_id = rooms_1.room_id)"
" Rows Removed by Join Filter: 27"
" -> Gather (cost=1000.00..8085.00 rows=10 width=4) (actual time=6.114..31.625 rows=8 loops=1)"
" Workers Planned: 2"
" Workers Launched: 2"
" -> Parallel Seq Scan on classes (cost=0.00..7084.00 rows=4 width=4) (actual time=3.963..22.311 rows=3 loops=3)"
" Filter: ((date = '2023-06-05'::date) AND ((('08:00:00'::time without time zone < start_time) AND ('11:00:00'::time
" Rows Removed by Filter: 133331"
" -> Materialize (cost=0.00..1.15 rows=10 width=4) (actual time=0.003..0.006 rows=4 loops=8)"
" -> Seq Scan on rooms rooms_1 (cost=0.00..1.10 rows=10 width=4) (actual time=0.006..0.012 rows=8 loops=1)"
"Planning Time: 1.518 ms"
"Execution Time: 31.985 ms"
```

Index added:

```
CREATE INDEX room_classes_idx ON classes (room_id, date, start_time, end_time);
```

We included classes_idx because we wanted to select certain information from the classes table for a specific date and time and join it with the rooms table on room_id. Adding an index allowed us to quickly look up the classes based on when they take place without having to do a full table scan. An index on the rooms table is not needed because there are only 10 rooms, so a sequential scan will be more beneficial than an index due to the small amount of data we have for rooms.

After adding the index, execution time was 0.273 ms:



```
"Sort (cost=48.13..48.15 rows=5 width=40) (actual time=0.181..0.182 rows=4 loops=1)"
" Sort Key: rooms.max_dog_capacity"
" Sort Method: quicksort Memory: 25kB"
" -> Seq Scan on rooms (cost=46.95..48.07 rows=5 width=40) (actual time=0.129..0.131 rows=4 loops=1)"
" Filter: (NOT (hashed SubPlan 1))"
" Rows Removed by Filter: 6"
" SubPlan 1"
" -> Nested Loop (cost=0.42..46.92 rows=10 width=4) (actual time=0.052..0.104 rows=8 loops=1)"
" -> Seq Scan on rooms rooms_1 (cost=0.00..1.10 rows=10 width=4) (actual time=0.001..0.002 rows=10 loops=1)"
" -> Index Only Scan using classes_idx on classes (cost=0.42..4.57 rows=1 width=4) (actual time=0.010..0.010 rows=1 loops=10)"
" Index Cond: ((room_id = rooms_1.room_id) AND (date = '2023-06-05'::date))"
" Filter: ((('08:00:00'::time without time zone < start_time) AND ('11:00:00'::time without time zone > start_time)) OR (('08:00:00'::time
" Rows Removed by Filter: 3"
" Heap Fetches: 0"
"Planning Time: 0.490 ms"
"Execution Time: 0.273 ms"
```

Because the time was faster and the cost was so much lower, we decided to keep the index.

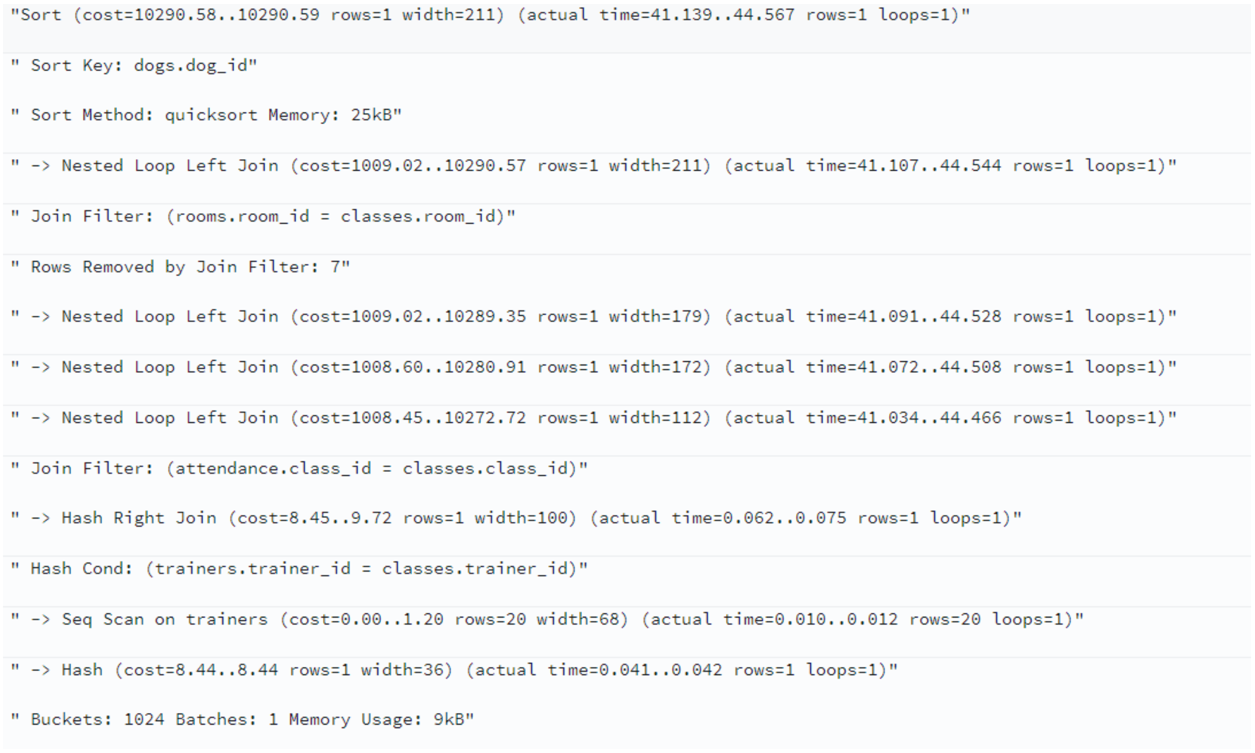
classes.py

`GET /classes/{id}`

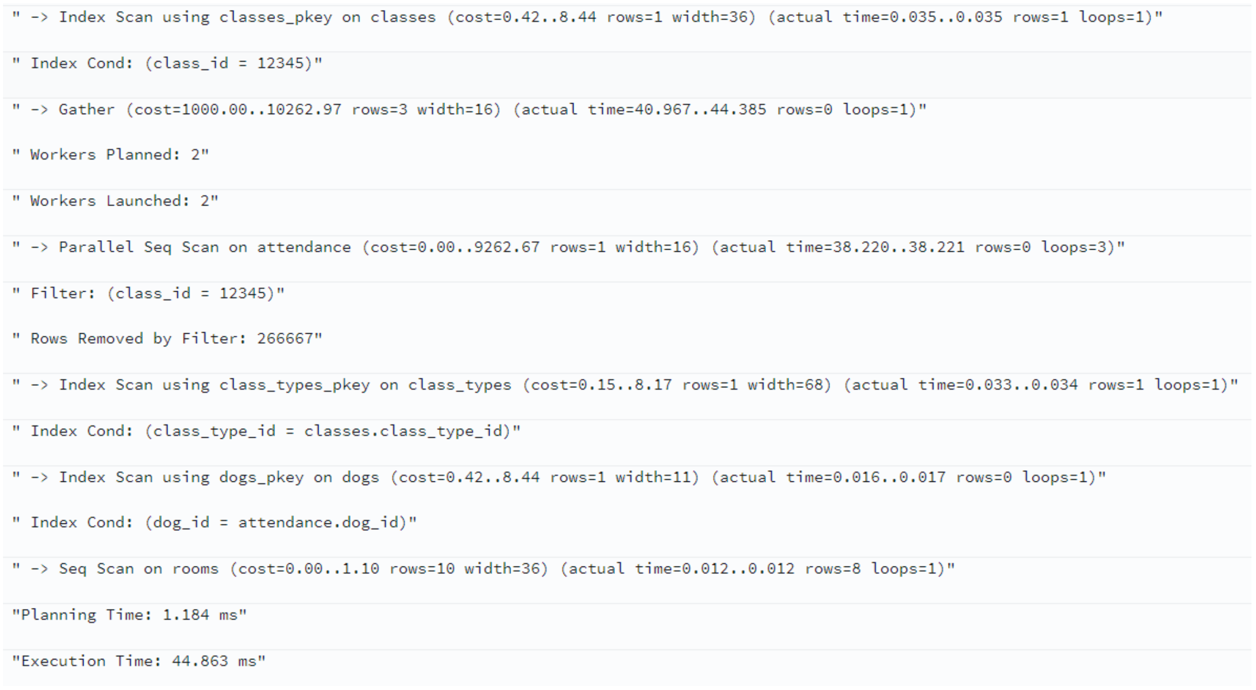
Running **Query**:

```
EXPLAIN ANALYZE
SELECT classes.class_id, trainers.first_name as first,
            trainers.last_name as last,
            class_types.type, class_types.description,
            date, start_time, end_time,
            dogs.dog_id, dogs.dog_name, attendance.check_in,
            trainers.trainer_id as trainer_id,
            rooms.room_id, room_name
        FROM classes
        LEFT JOIN trainers on trainers.trainer_id = classes.trainer_id
        LEFT JOIN attendance on attendance.class_id = classes.class_id
        LEFT JOIN class_types on class_types.class_type_id =
classes.class_type_id
        LEFT JOIN dogs on dogs.dog_id = attendance.dog_id
        LEFT JOIN rooms ON rooms.room_id = classes.room_id
        WHERE classes.class_id = 12345
        ORDER BY dogs.dog_id
```

Before adding the index, execution time was 44.863 ms:



```
"Sort (cost=10290.58..10290.59 rows=1 width=211) (actual time=41.139..44.567 rows=1 loops=1)"
" Sort Key: dogs.dog_id"
" Sort Method: quicksort Memory: 25kB"
" -> Nested Loop Left Join (cost=1009.02..10290.57 rows=1 width=211) (actual time=41.107..44.544 rows=1 loops=1)"
" Join Filter: (rooms.room_id = classes.room_id)"
" Rows Removed by Join Filter: 7"
" -> Nested Loop Left Join (cost=1009.02..10289.35 rows=1 width=179) (actual time=41.091..44.528 rows=1 loops=1)"
" -> Nested Loop Left Join (cost=1008.60..10280.91 rows=1 width=172) (actual time=41.072..44.508 rows=1 loops=1)"
" -> Nested Loop Left Join (cost=1008.45..10272.72 rows=1 width=112) (actual time=41.034..44.466 rows=1 loops=1)"
" Join Filter: (attendance.class_id = classes.class_id)"
" -> Hash Right Join (cost=8.45..9.72 rows=1 width=100) (actual time=0.062..0.075 rows=1 loops=1)"
" Hash Cond: (trainers.trainer_id = classes.trainer_id)"
" -> Seq Scan on trainers (cost=0.00..1.20 rows=20 width=68) (actual time=0.010..0.012 rows=20 loops=1)"
" -> Hash (cost=8.44..8.44 rows=1 width=36) (actual time=0.041..0.042 rows=1 loops=1)"
" Buckets: 1024 Batches: 1 Memory Usage: 9kB"
```



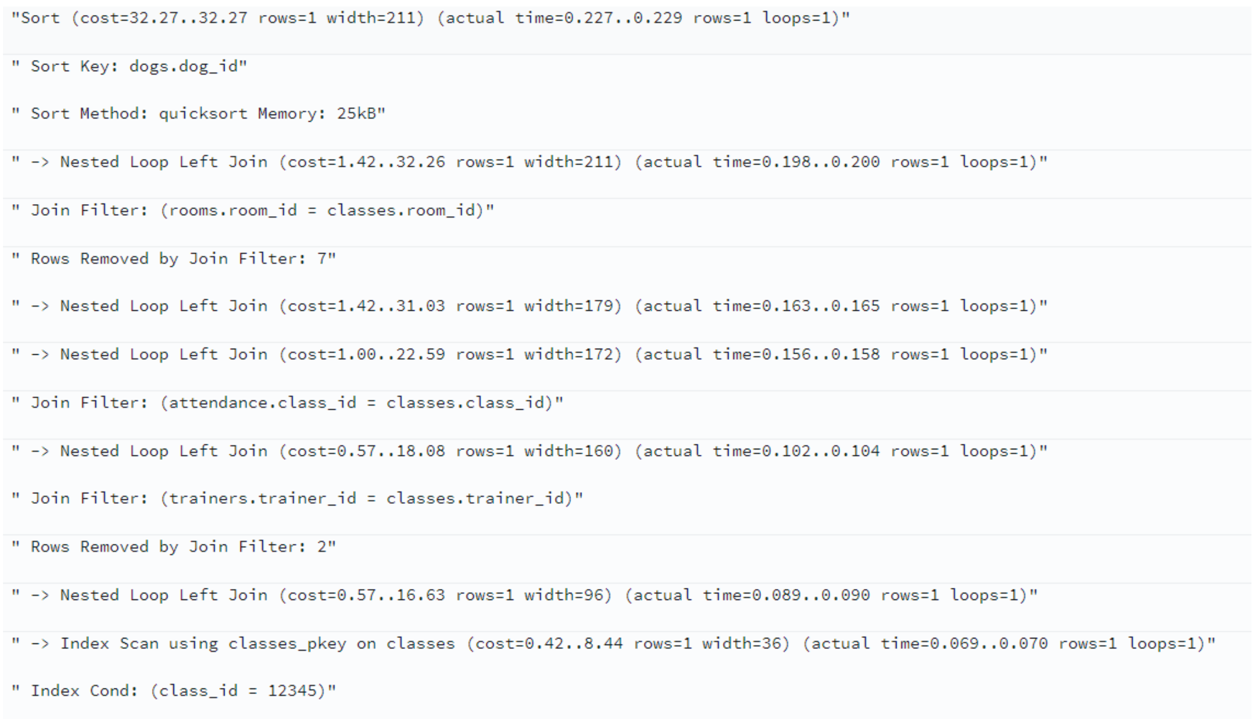
```
" -> Index Scan using classes_pkey on classes (cost=0.42..8.44 rows=1 width=36) (actual time=0.035..0.035 rows=1 loops=1)"
" Index Cond: (class_id = 12345)"
" -> Gather (cost=1000.00..10262.97 rows=3 width=16) (actual time=40.967..44.385 rows=0 loops=1)"
" Workers Planned: 2"
" Workers Launched: 2"
" -> Parallel Seq Scan on attendance (cost=0.00..9262.67 rows=1 width=16) (actual time=38.220..38.221 rows=0 loops=3)"
" Filter: (class_id = 12345)"
" Rows Removed by Filter: 266667"
" -> Index Scan using class_types_pkey on class_types (cost=0.15..8.17 rows=1 width=68) (actual time=0.033..0.034 rows=1 loops=1)"
" Index Cond: (class_type_id = classes.class_type_id)"
" -> Index Scan using dogs_pkey on dogs (cost=0.42..8.44 rows=1 width=11) (actual time=0.016..0.017 rows=0 loops=1)"
" Index Cond: (dog_id = attendance.dog_id)"
" -> Seq Scan on rooms (cost=0.00..1.10 rows=10 width=36) (actual time=0.012..0.012 rows=8 loops=1)"
"Planning Time: 1.184 ms"
"Execution Time: 44.863 ms"
```

Index added:

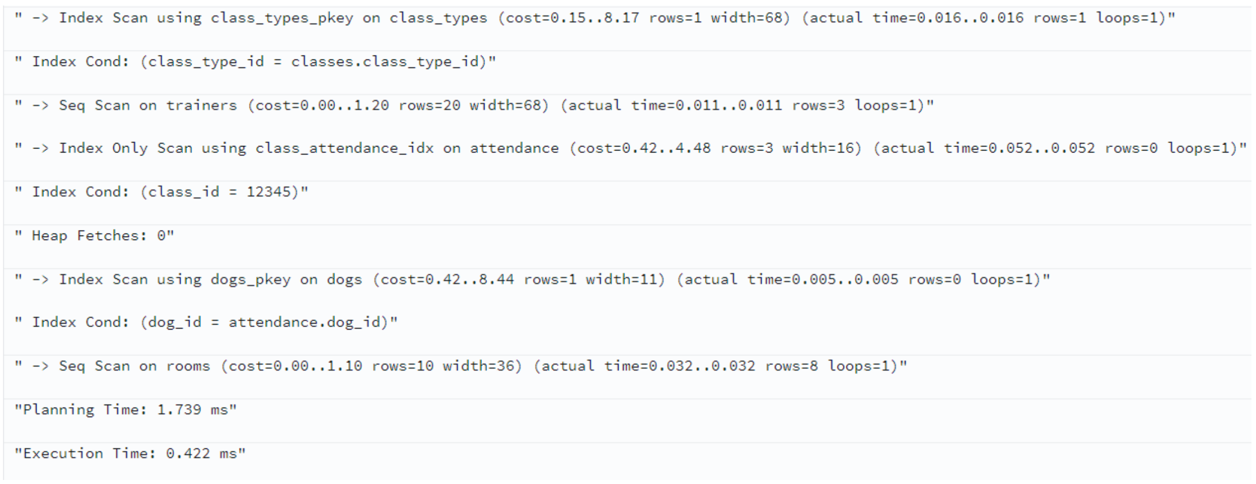
```
CREATE INDEX class_attendance_idx ON attendance (class_id, dog_id, check_in);
```

Before adding the index, the only sequential scan (that didn't involve a very small table such as rooms and trainers) was on attendance. We included class_attendance_idx because we wanted to select check_in time and join the attendance table with other tables on class_id and dog_id. Adding an index allowed us to quickly look up the attendance for these ids.

After adding the index, execution time was 0.422 ms:



```
"Sort (cost=32.27..32.27 rows=1 width=211) (actual time=0.227..0.229 rows=1 loops=1)"
" Sort Key: dogs.dog_id"
" Sort Method: quicksort Memory: 25kB"
" -> Nested Loop Left Join (cost=1.42..32.26 rows=1 width=211) (actual time=0.198..0.200 rows=1 loops=1)"
" Join Filter: (rooms.room_id = classes.room_id)"
" Rows Removed by Join Filter: 7"
" -> Nested Loop Left Join (cost=1.42..31.03 rows=1 width=179) (actual time=0.163..0.165 rows=1 loops=1)"
" -> Nested Loop Left Join (cost=1.00..22.59 rows=1 width=172) (actual time=0.156..0.158 rows=1 loops=1)"
" Join Filter: (attendance.class_id = classes.class_id)"
" -> Nested Loop Left Join (cost=0.57..18.08 rows=1 width=160) (actual time=0.102..0.104 rows=1 loops=1)"
" Join Filter: (trainers.trainer_id = classes.trainer_id)"
" Rows Removed by Join Filter: 2"
" -> Nested Loop Left Join (cost=0.57..16.63 rows=1 width=96) (actual time=0.089..0.090 rows=1 loops=1)"
" -> Index Scan using classes_pkey on classes (cost=0.42..8.44 rows=1 width=36) (actual time=0.069..0.070 rows=1 loops=1)"
" Index Cond: (class_id = 12345)"
```



```
" -> Index Scan using class_types_pkey on class_types (cost=0.15..8.17 rows=1 width=68) (actual time=0.016..0.016 rows=1 loops=1)"
" Index Cond: (class_type_id = classes.class_type_id)"
" -> Seq Scan on trainers (cost=0.00..1.20 rows=20 width=68) (actual time=0.011..0.011 rows=3 loops=1)"
" -> Index Only Scan using class_attendance_idx on attendance (cost=0.42..4.48 rows=3 width=16) (actual time=0.052..0.052 rows=0 loops=1)"
" Index Cond: (class_id = 12345)"
" Heap Fetches: 0"
" -> Index Scan using dogs_pkey on dogs (cost=0.42..8.44 rows=1 width=11) (actual time=0.005..0.005 rows=0 loops=1)"
" Index Cond: (dog_id = attendance.dog_id)"
" -> Seq Scan on rooms (cost=0.00..1.10 rows=10 width=36) (actual time=0.032..0.032 rows=8 loops=1)"
"Planning Time: 1.739 ms"
"Execution Time: 0.422 ms"
```

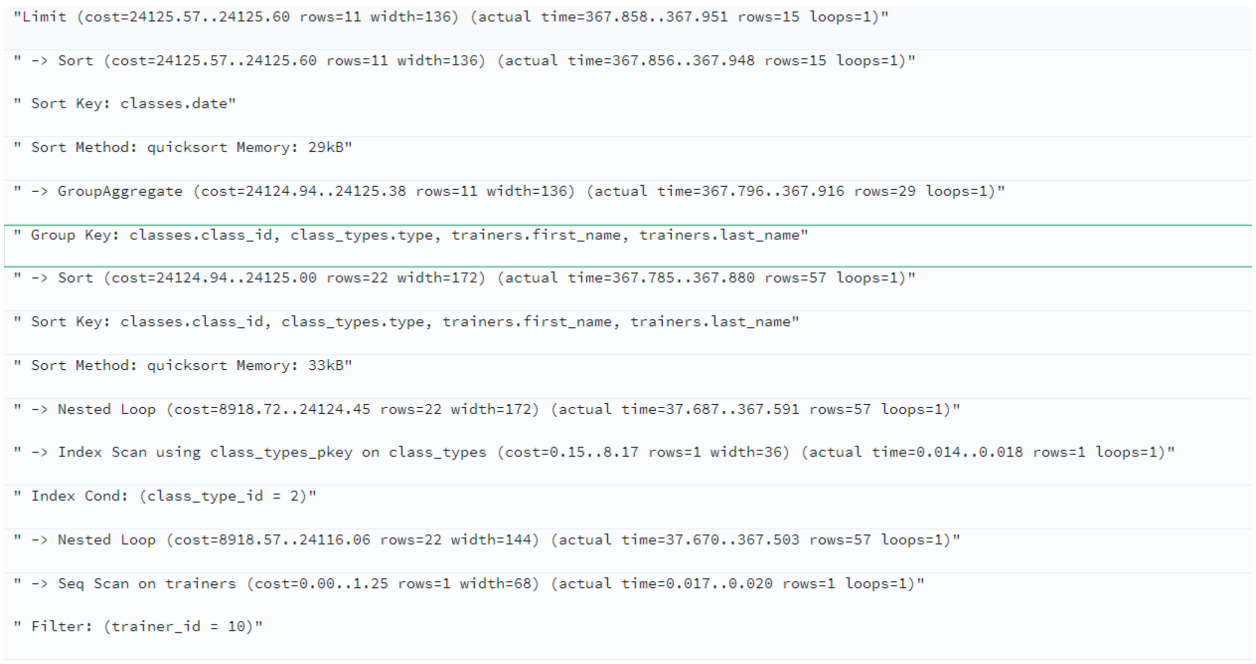
Because the time was faster and the cost was so much lower, we decided to keep the index.

GET /classes/

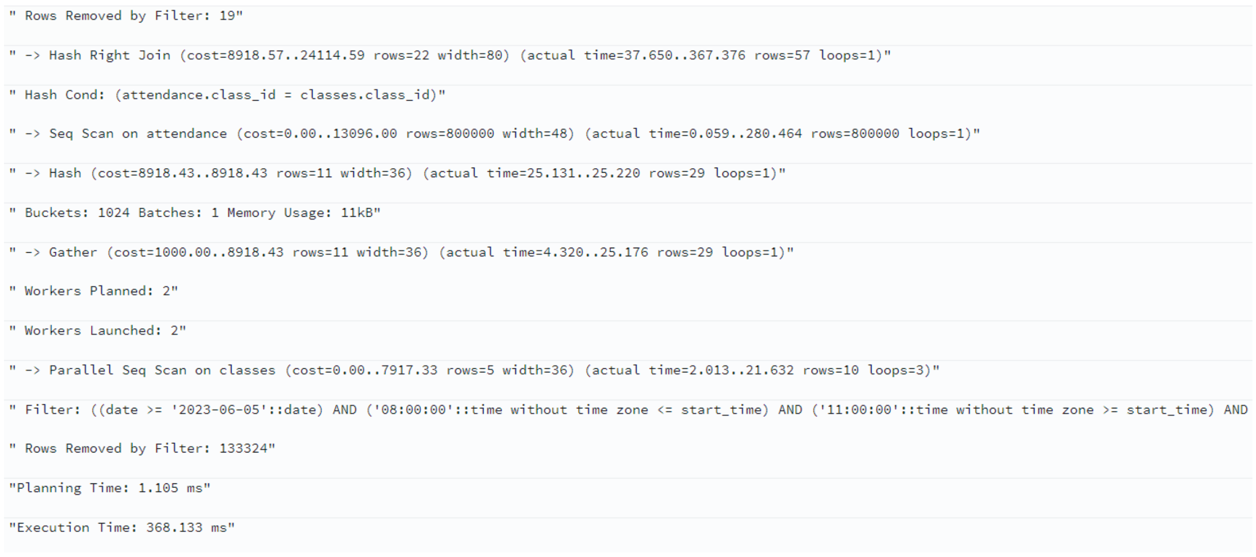
Running **Query**:

```
EXPLAIN ANALYZE
SELECT classes.class_id, classes.trainer_id, type, classes.date,
    start_time, end_time, trainers.first_name,
    trainers.last_name, room_id,
    COUNT(attendance) as num_dogs
FROM classes
JOIN trainers ON trainers.trainer_id = classes.trainer_id
JOIN class_types ON
    class_types.class_type_id = classes.class_type_id
LEFT JOIN attendance on attendance.class_id = classes.class_id
WHERE ('2023-06-05' IS NULL OR classes.date >= CAST('2023-06-05' AS DATE)) AND
    (classes.class_type_id = 2 OR 2 IS NULL) AND
    ((to_char(date, 'Day') LIKE 'Monday%' OR
    to_char(date, 'Day') LIKE 'None%' OR
    to_char(date, 'Day') LIKE 'None%' OR
    to_char(date, 'Day') LIKE 'None%' OR
    to_char(date, 'Day') LIKE 'None%' OR
    to_char(date, 'Day') LIKE 'None%' OR
    to_char(date, 'Day') LIKE 'None%' OR
    'None%' = 'None%'
    ) OR
    ('None%' LIKE 'None%' AND
    'None%' LIKE 'None%' AND
    'None%' LIKE 'None%' AND
    'None%' LIKE 'None%' AND
    'None%' LIKE 'None%' AND
    'None%' LIKE 'None%' AND
    'None%' LIKE 'None%'  ) ) AND
    ('8:00:00 AM' IS NULL OR
        (CAST('8:00:00 AM' AS TIME) <= classes.start_time
        AND CAST('11:00:00 AM' AS TIME) >= classes.start_time)
        AND (CAST('8:00:00 AM' AS TIME) <= classes.end_time
        AND CAST('11:00:00 AM' AS TIME) >= classes.end_time))
    AND (10 IS NULL OR classes.trainer_id = 10)
GROUP BY classes.class_id, classes.trainer_id, type, classes.date,
    start_time, end_time, trainers.first_name,
    trainers.last_name, room_id
ORDER BY date ASC
LIMIT 15
```

Before adding the index, execution time was 368.133 ms:



```
"Limit (cost=24125.57..24125.60 rows=11 width=136) (actual time=367.858..367.951 rows=15 loops=1)"
" -> Sort (cost=24125.57..24125.60 rows=11 width=136) (actual time=367.856..367.948 rows=15 loops=1)"
" Sort Key: classes.date"
" Sort Method: quicksort Memory: 29kB"
" -> GroupAggregate (cost=24124.94..24125.38 rows=11 width=136) (actual time=367.796..367.916 rows=29 loops=1)"
" Group Key: classes.class_id, class_types.type, trainers.first_name, trainers.last_name"
" -> Sort (cost=24124.94..24125.00 rows=22 width=172) (actual time=367.785..367.880 rows=57 loops=1)"
" Sort Key: classes.class_id, class_types.type, trainers.first_name, trainers.last_name"
" Sort Method: quicksort Memory: 33kB"
" -> Nested Loop (cost=8918.72..24124.45 rows=22 width=172) (actual time=37.687..367.591 rows=57 loops=1)"
" -> Index Scan using class_types_pkey on class_types (cost=0.15..8.17 rows=1 width=36) (actual time=0.014..0.018 rows=1 loops=1)"
" Index Cond: (class_type_id = 2)"
" -> Nested Loop (cost=8918.57..24116.06 rows=22 width=144) (actual time=37.670..367.503 rows=57 loops=1)"
" -> Seq Scan on trainers (cost=0.00..1.25 rows=1 width=68) (actual time=0.017..0.020 rows=1 loops=1)"
" Filter: (trainer_id = 10)"
```



```
" Rows Removed by Filter: 19"
" -> Hash Right Join (cost=8918.57..24114.59 rows=22 width=80) (actual time=37.650..367.376 rows=57 loops=1)"
" Hash Cond: (attendance.class_id = classes.class_id)"
" -> Seq Scan on attendance (cost=0.00..13096.00 rows=800000 width=48) (actual time=0.059..280.464 rows=800000 loops=1)"
" -> Hash (cost=8918.43..8918.43 rows=11 width=36) (actual time=25.131..25.220 rows=29 loops=1)"
" Buckets: 1024 Batches: 1 Memory Usage: 11kB"
" -> Gather (cost=1000.00..8918.43 rows=11 width=36) (actual time=4.320..25.176 rows=29 loops=1)"
" Workers Planned: 2"
" Workers Launched: 2"
" -> Parallel Seq Scan on classes (cost=0.00..7917.33 rows=5 width=36) (actual time=2.013..21.632 rows=10 loops=3)"
" Filter: ((date >= '2023-06-05'::date) AND ('08:00:00'::time without time zone <= start_time) AND ('11:00:00'::time without time zone >= start_time) AND
" Rows Removed by Filter: 133324"
"Planning Time: 1.105 ms"
"Execution Time: 368.133 ms"
```

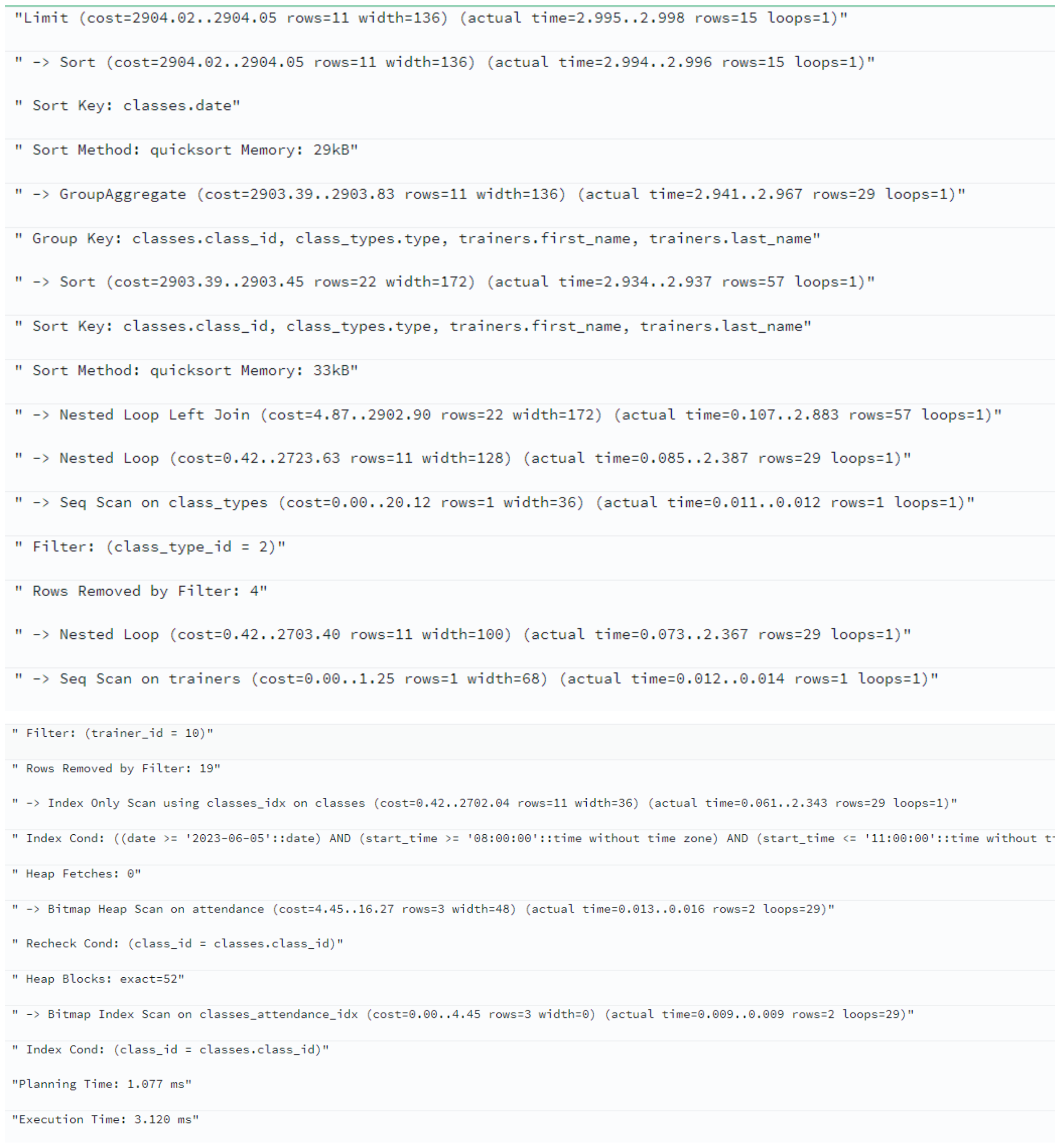
Indexes added:

```
CREATE INDEX classes_attendance_idx ON attendance (class_id);

CREATE INDEX classes_idx ON classes (date, start_time, end_time, trainer_id,
class_type_id, room_id, class_id);
```

Before adding the index, the sequential scans (that didn’t involve a very small table such as rooms and trainers) were on classes (filtering by date and time) and attendance (on class_id). We included classes_attendance_idx because we wanted to join the attendance table with classes on class_id. We also created a classes_idx because the transaction sorts by classes.date and filters by date and time, so including an index that can look up a date would be helpful.

After adding the indexes, execution time was 3.120 ms:



```
"Limit (cost=2904.02..2904.05 rows=11 width=136) (actual time=2.995..2.998 rows=15 loops=1)"
" -> Sort (cost=2904.02..2904.05 rows=11 width=136) (actual time=2.994..2.996 rows=15 loops=1)"
" Sort Key: classes.date"
" Sort Method: quicksort Memory: 29kB"
" -> GroupAggregate (cost=2903.39..2903.83 rows=11 width=136) (actual time=2.941..2.967 rows=29 loops=1)"
" Group Key: classes.class_id, class_types.type, trainers.first_name, trainers.last_name"
" -> Sort (cost=2903.39..2903.45 rows=22 width=172) (actual time=2.934..2.937 rows=57 loops=1)"
" Sort Key: classes.class_id, class_types.type, trainers.first_name, trainers.last_name"
" Sort Method: quicksort Memory: 33kB"
" -> Nested Loop Left Join (cost=4.87..2902.90 rows=22 width=172) (actual time=0.107..2.883 rows=57 loops=1)"
" -> Nested Loop (cost=0.42..2723.63 rows=11 width=128) (actual time=0.085..2.387 rows=29 loops=1)"
" -> Seq Scan on class_types (cost=0.00..20.12 rows=1 width=36) (actual time=0.011..0.012 rows=1 loops=1)"
" Filter: (class_type_id = 2)"
" Rows Removed by Filter: 4"
" -> Nested Loop (cost=0.42..2703.40 rows=11 width=100) (actual time=0.073..2.367 rows=29 loops=1)"
" -> Seq Scan on trainers (cost=0.00..1.25 rows=1 width=68) (actual time=0.012..0.014 rows=1 loops=1)"
" Filter: (trainer_id = 10)"
" Rows Removed by Filter: 19"
" -> Index Only Scan using classes_idx on classes (cost=0.42..2702.04 rows=11 width=36) (actual time=0.061..2.343 rows=29 loops=1)"
" Index Cond: ((date >= '2023-06-05'::date) AND (start_time >= '08:00:00'::time without time zone) AND (start_time <= '11:00:00'::time without t
" Heap Fetches: 0"
" -> Bitmap Heap Scan on attendance (cost=4.45..16.27 rows=3 width=48) (actual time=0.013..0.016 rows=2 loops=29)"
" Recheck Cond: (class_id = classes.class_id)"
" Heap Blocks: exact=52"
" -> Bitmap Index Scan on classes_attendance_idx (cost=0.00..4.45 rows=3 width=0) (actual time=0.009..0.009 rows=2 loops=29)"
" Index Cond: (class_id = classes.class_id)"
"Planning Time: 1.077 ms"
"Execution Time: 3.120 ms"
```

Because the time was faster and the cost was so much lower, we decided to keep the index.

Overall

These are the indexes that we believe are relevant:

```
CREATE INDEX dogs_idx ON dogs (lower(dog_name) text_pattern_ops, lower(breed)
text_pattern_ops,  lower(client_email) text_pattern_ops);

CREATE INDEX comments_idx ON comments (dog_id, trainer_id, time_added,
comment_text, comment_id);

CREATE INDEX room_classes_idx ON classes (room_id, date, start_time, end_time);

CREATE INDEX classes_idx ON classes (date, start_time, end_time, trainer_id,
class_type_id, room_id, class_id);

CREATE INDEX class_attendance_idx ON attendance (class_id, dog_id, check_in);

CREATE INDEX classes_attendance_idx ON attendance (class_id);
```