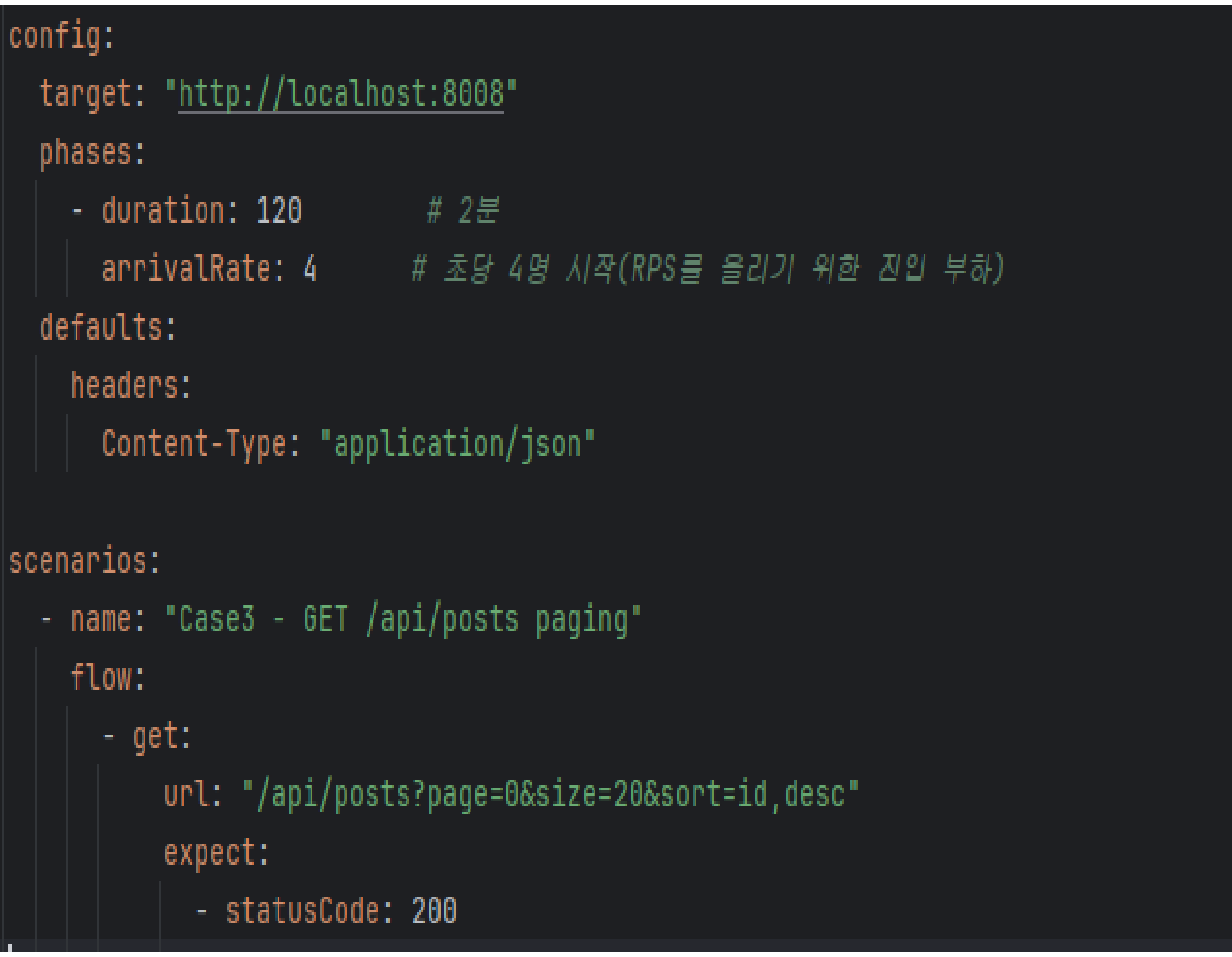
Case3

COUNT Query Structure Change in Pagination

Verification Evidence PDF

Test Conditions (Fixed)

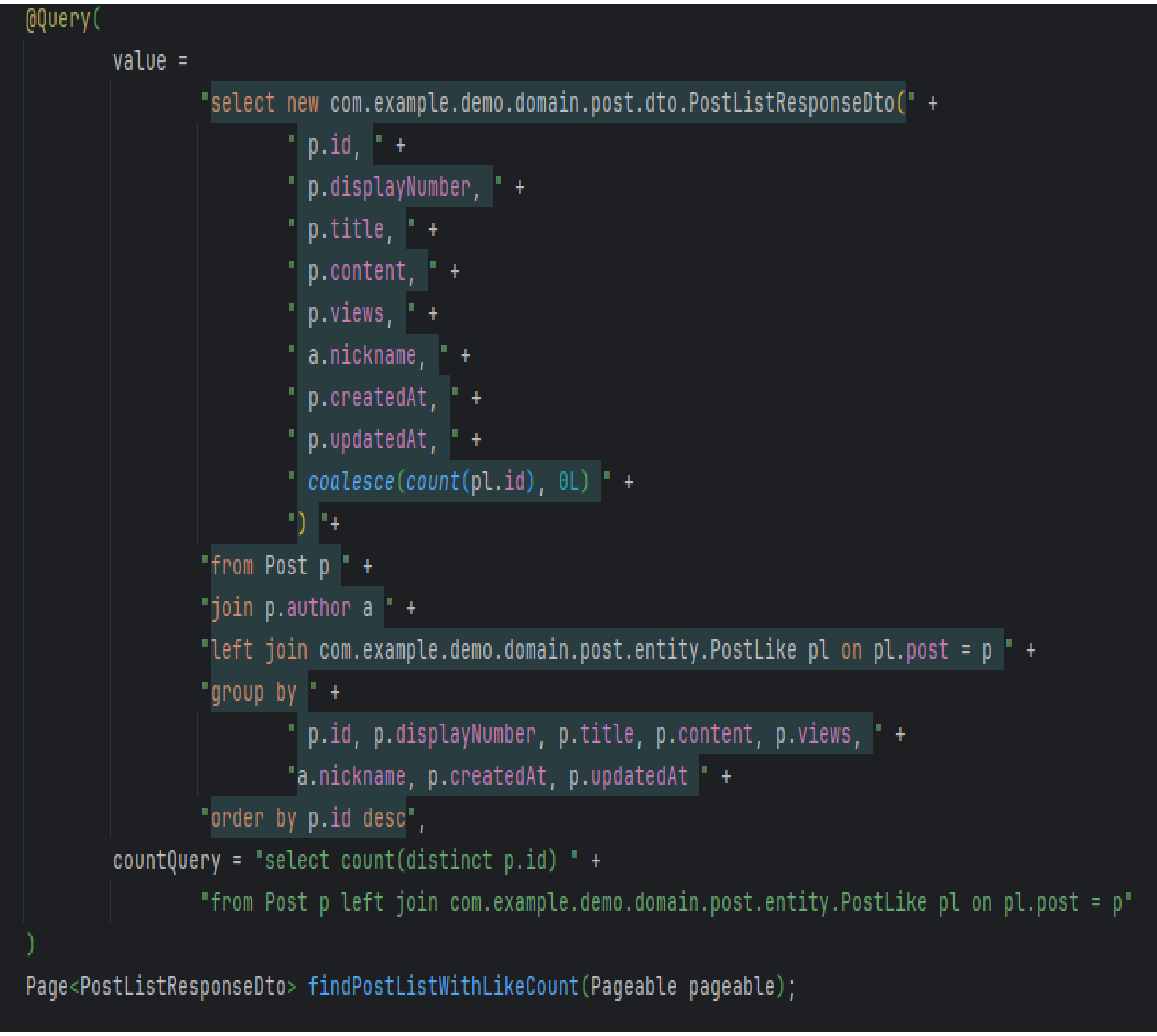


```yaml
config:
  target: "http://localhost:8008"
  phases:
    - duration: 120        # 2분
      arrivalRate: 4      # 초당 4명 시작(RPS를 올리기 위한 진입 부하)
  defaults:
    headers:
      Content-Type: "application/json"

scenarios:
  - name: "Case3 - GET /api/posts paging"
    flow:
      - get:
          url: "/api/posts?page=0&size=20&sort=id,desc"
          expect:
            - statusCode: 200
```

Artillery Test Scenario: case3_get_posts.yml

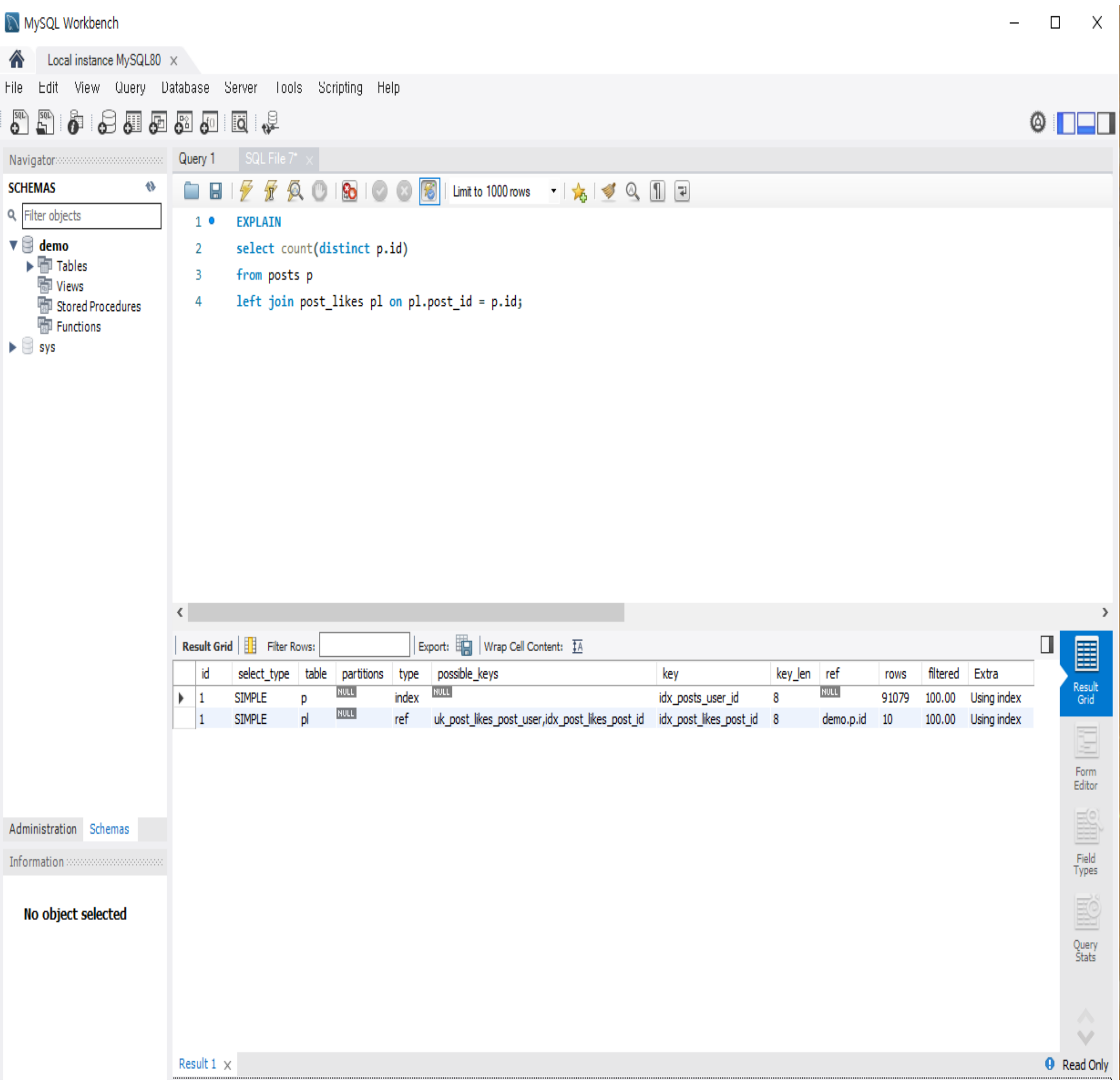
COUNT Query Structure (Before)



```
@Query(
        value =
                "select new com.example.demo.domain.post.dto.PostListResponseDto(" +
                        " p.id, " +
                        " p.displayNumber, " +
                        " p.title, " +
                        " p.content, " +
                        " p.views, " +
                        " a.nickname, " +
                        " p.createdAt, " +
                        " p.updatedAt, " +
                        " coalesce(count(pl.id), 0L) " +
                        ") "+
                "from Post p " +
                "join p.author a " +
                "left join com.example.demo.domain.post.entity.PostLike pl on pl.post = p " +
                "group by " +
                        " p.id, p.displayNumber, p.title, p.content, p.views, " +
                        "a.nickname, p.createdAt, p.updatedAt " +
                "order by p.id desc",
        countQuery = "select count(distinct p.id) " +
                "from Post p left join com.example.demo.domain.post.entity.PostLike pl on pl.post = p"
)
Page<PostListResponseDto> findPostListWithLikeCount(Pageable pageable);
```

COUNT query including LEFT JOIN and DISTINCT before structure separation.

Executions Plan (Before)

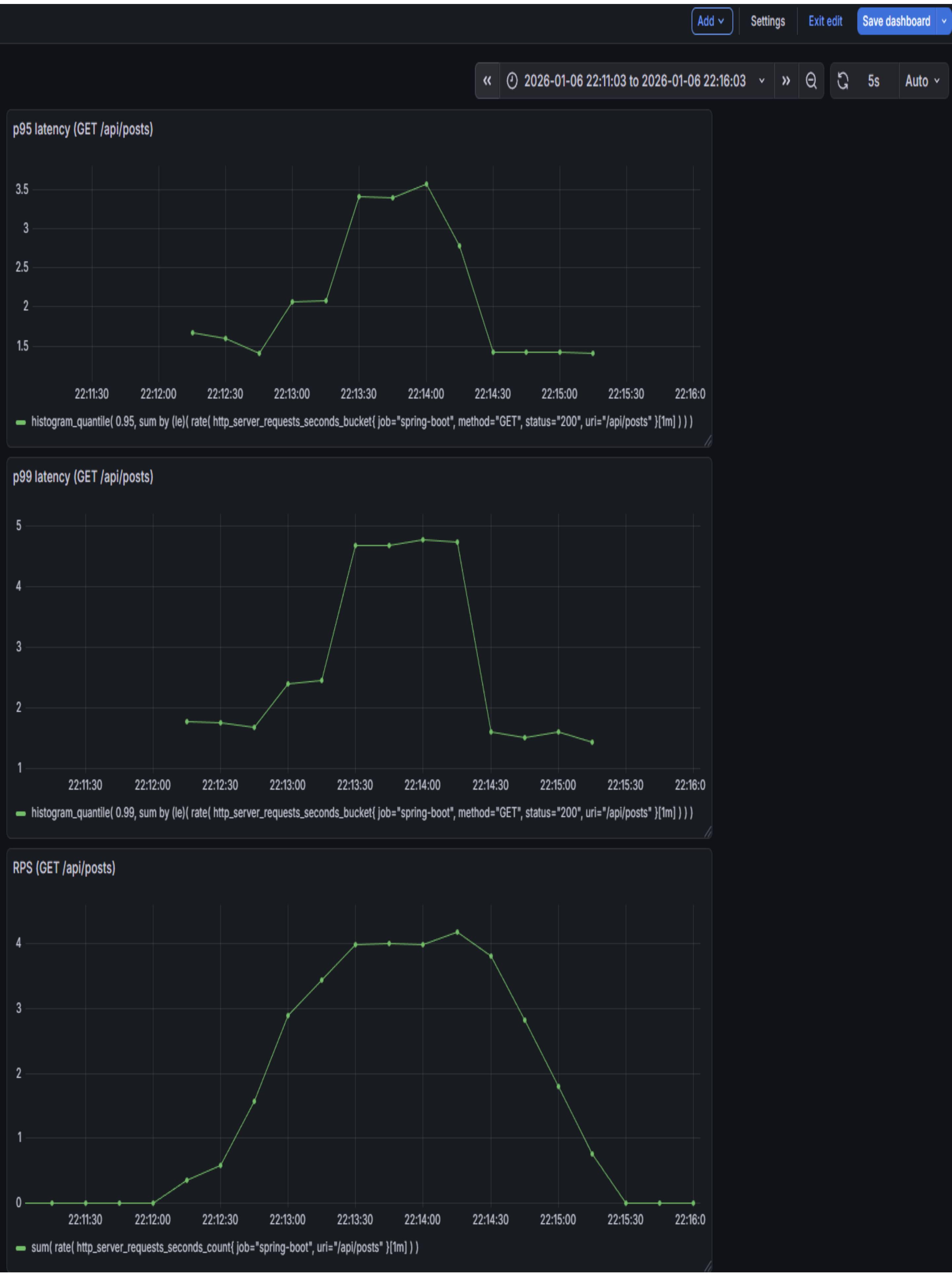


```sql
EXPLAIN
select count(distinct p.id)
from posts p
left join post_likes pl on pl.post_id = p.id;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	p	NULL	index	NULL	idx_posts_user_id	8	NULL	91079	100.00	Using index
1	SIMPLE	pl	NULL	ref	uk_post_likes_post_user,idx_post_likes_post_id	idx_post_likes_post_id	8	demo.p.id	10	100.00	Using index

Execution plan of COUNT query with LEFT JOIN and DISTINCT before structure separation.

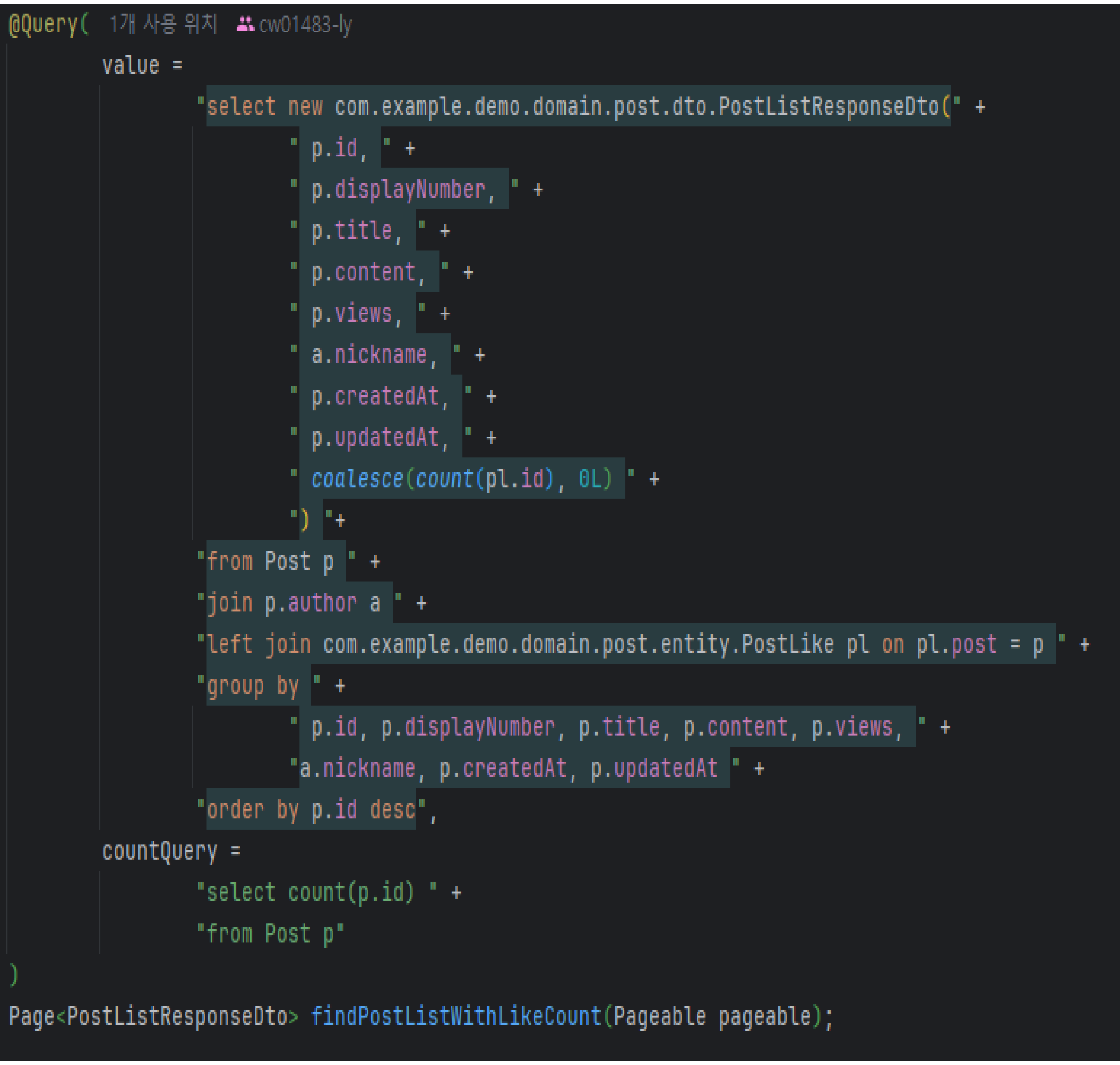
Performance Metrics (Before)



```
histogram_quantile( 0.95, sum by (le)( rate( http_server_requests_seconds_bucket{ job="spring-boot", method="GET", status="200", uri="/api/posts" }[1m] ) ) )
histogram_quantile( 0.99, sum by (le)( rate( http_server_requests_seconds_bucket{ job="spring-boot", method="GET", status="200", uri="/api/posts" }[1m] ) ) )
sum( rate( http_server_requests_seconds_count{ job="spring-boot", uri="/api/posts" }[1m] ) )
```

Grafana metrics (p95/p99 latency and RPS) before structure separation.

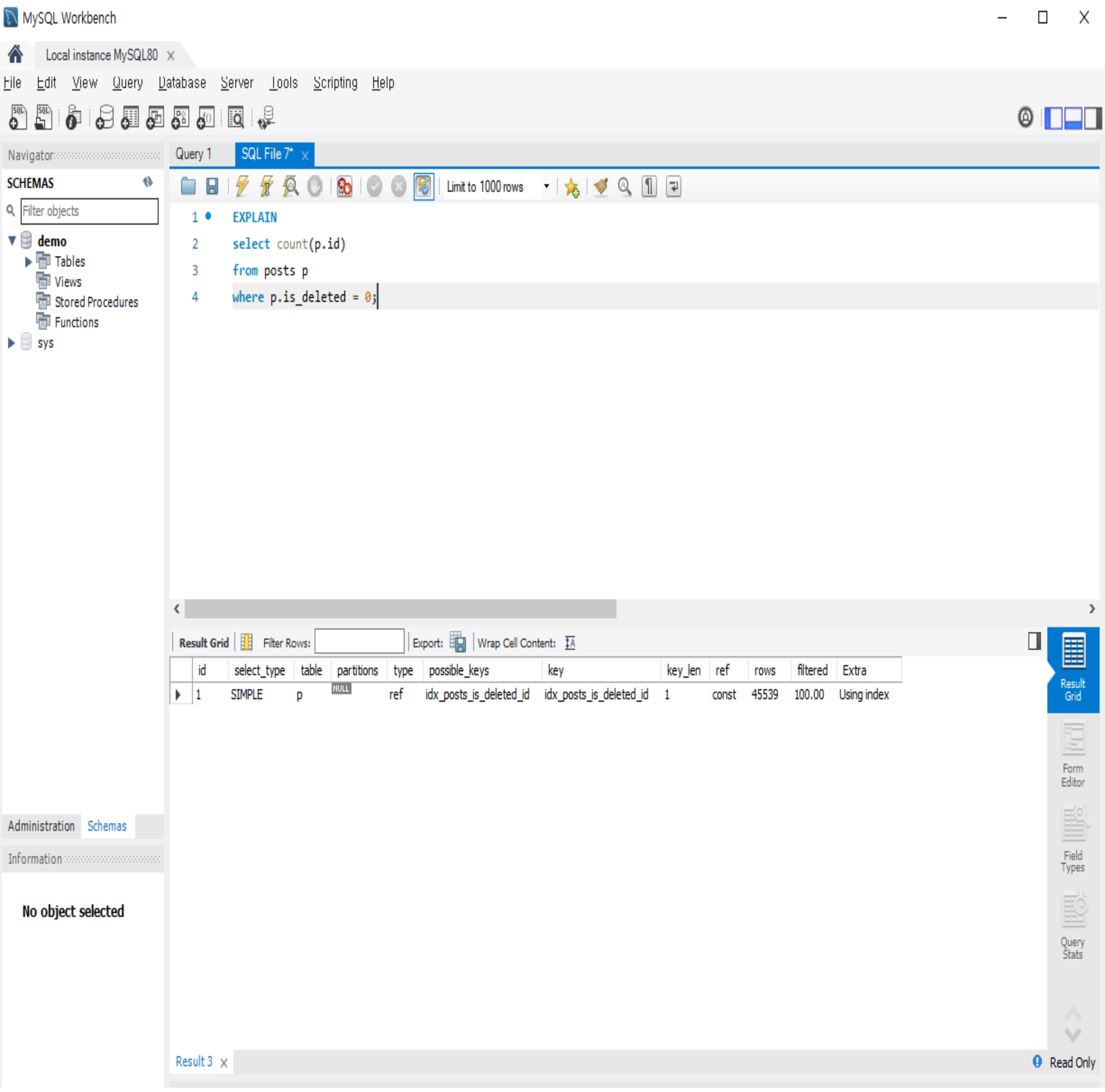
COUNT Query Structure (After)



```
@Query(  1개 사용 위치  cw01483-ly
        value =
                "select new com.example.demo.domain.post.dto.PostListResponseDto(" +
                        " p.id, " +
                        " p.displayNumber, " +
                        " p.title, " +
                        " p.content, " +
                        " p.views, " +
                        " a.nickname, " +
                        " p.createdAt, " +
                        " p.updatedAt, " +
                        " coalesce(count(pl.id), 0L) " +
                        ") "+
                "from Post p " +
                "join p.author a " +
                "left join com.example.demo.domain.post.entity.PostLike pl on pl.post = p " +
                "group by " +
                        " p.id, p.displayNumber, p.title, p.content, p.views, " +
                        "a.nickname, p.createdAt, p.updatedAt " +
                "order by p.id desc",
        countQuery =
                "select count(p.id) " +
                "from Post p"
)
Page<PostListResponseDto> findPostListWithLikeCount(Pageable pageable);
```

COUNT query separated from JOIN and executed on posts only after structure change.

Execution Plan (After)

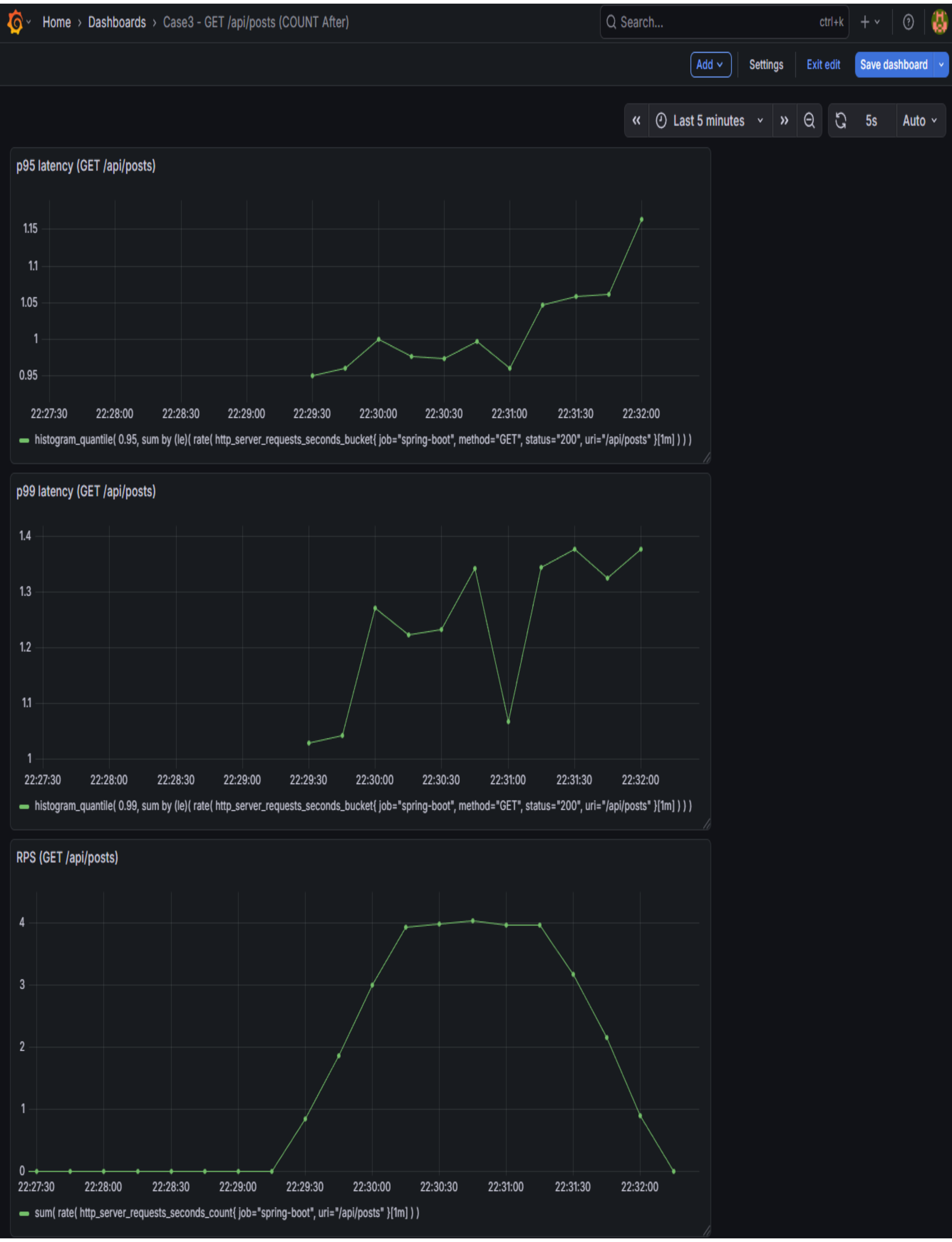


```sql
EXPLAIN
select count(p.id)
from posts p
where p.is_deleted = 0;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	p	NULL	ref	idx_posts_is_deleted_id	idx_posts_is_deleted_id	1	const	45539	100.00	Using index

Execution plan of COUNT query executed on posts only after structure separation.

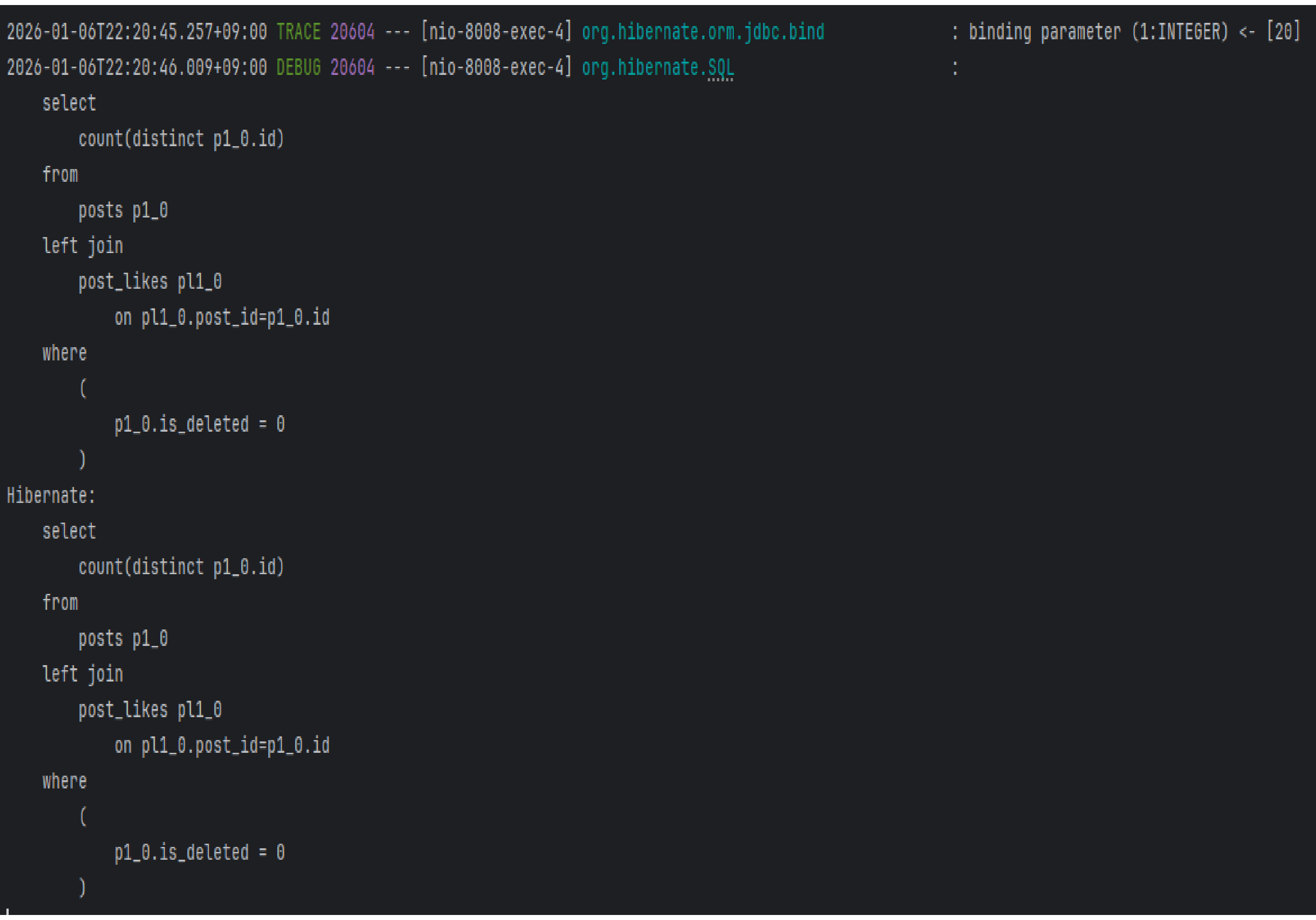
Performance Metrics (After)



```
histogram_quantile( 0.95, sum by (le)( rate( http_server_requests_seconds_bucket{ job="spring-boot", method="GET", status="200", uri="/api/posts" }[1m] ) ) )
histogram_quantile( 0.99, sum by (le)( rate( http_server_requests_seconds_bucket{ job="spring-boot", method="GET", status="200", uri="/api/posts" }[1m] ) ) )
sum( rate( http_server_requests_seconds_count{ job="spring-boot", uri="/api/posts" }[1m] ) )
```

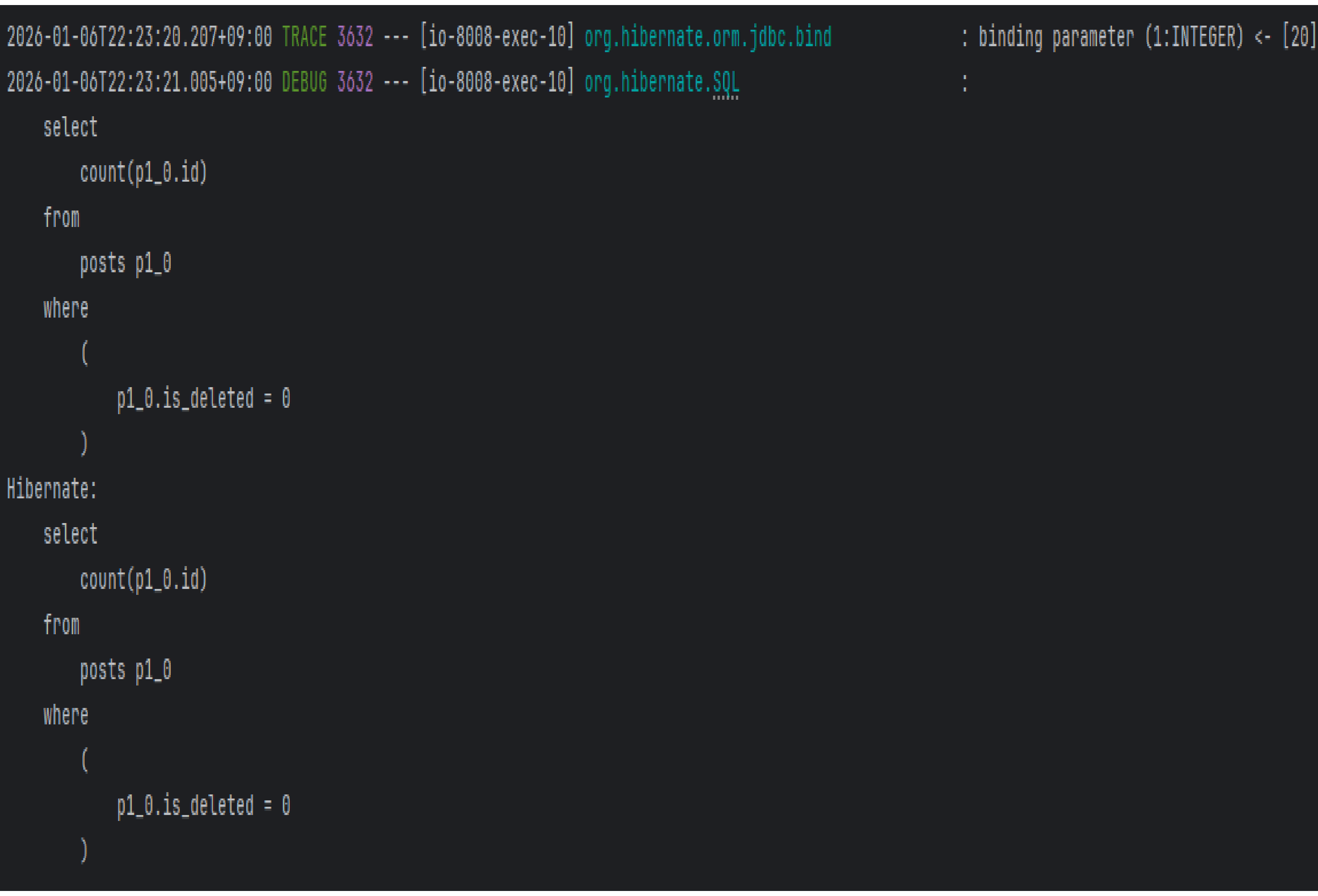
Grafana metrics (p95/p99 latency and RPS) after structure separation.

SQL Execution Log (Before / After)



```
2026-01-06T22:20:45.257+09:00 TRACE 20604 --- [nio-8008-exec-4] org.hibernate.orm.jdbc.bind              : binding parameter (1:INTEGER) <- [20]
2026-01-06T22:20:46.009+09:00 DEBUG 20604 --- [nio-8008-exec-4] org.hibernate.SQL                        : 
    select
        count(distinct p1_0.id) 
    from
        posts p1_0 
    left join
        post_likes pl1_0 
            on pl1_0.post_id=p1_0.id 
    where
        (
            p1_0.is_deleted = 0
        )
Hibernate: 
    select
        count(distinct p1_0.id) 
    from
        posts p1_0 
    left join
        post_likes pl1_0 
            on pl1_0.post_id=p1_0.id 
    where
        (
            p1_0.is_deleted = 0
        )
```

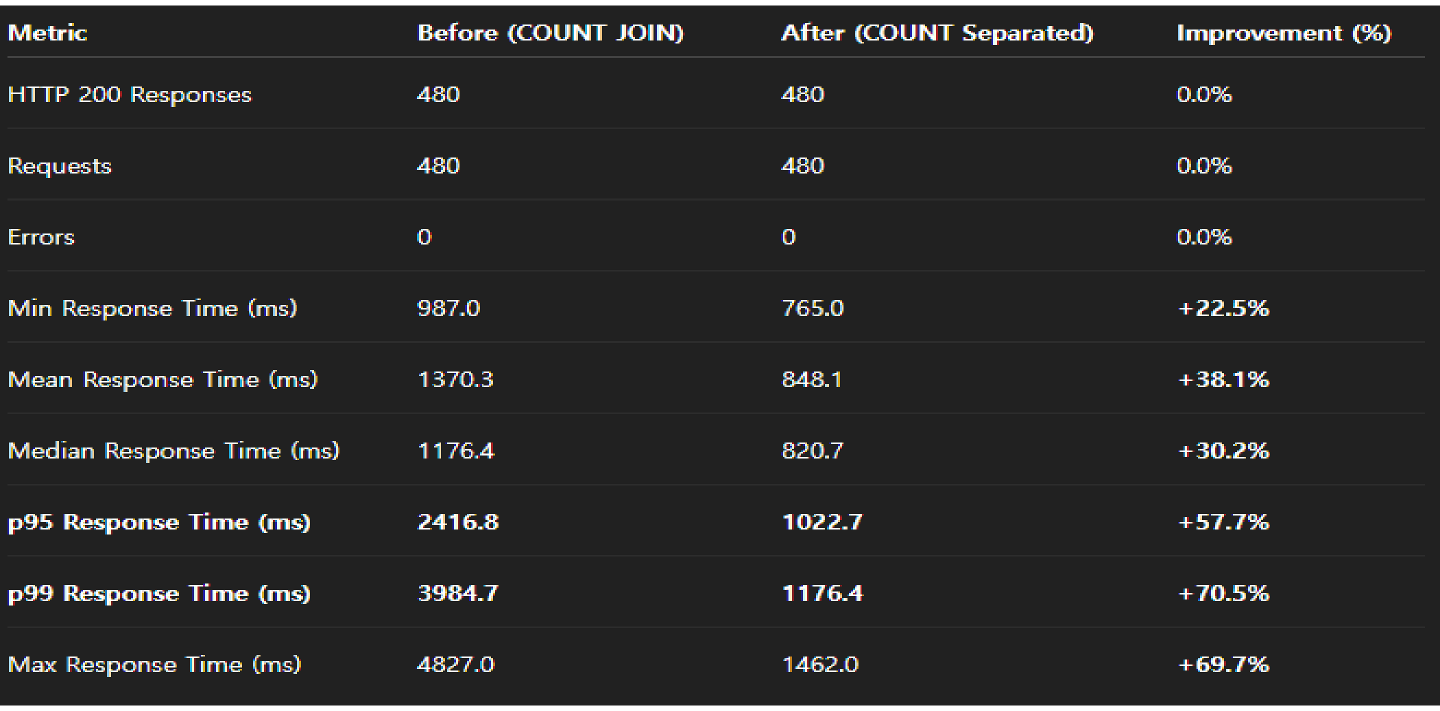
Hibernate SQL log showing COUNT query with LEFT JOIN before structure separation.



```
2026-01-06T22:23:20.207+09:00 TRACE 3632 --- [io-8008-exec-10] org.hibernate.orm.jdbc.bind              : binding parameter (1:INTEGER) <- [20]
2026-01-06T22:23:21.005+09:00 DEBUG 3632 --- [io-8008-exec-10] org.hibernate.SQL                        : 
    select
        count(p1_0.id) 
    from
        posts p1_0 
    where
        (
            p1_0.is_deleted = 0
        )
Hibernate: 
    select
        count(p1_0.id) 
    from
        posts p1_0 
    where
        (
            p1_0.is_deleted = 0
        )
```

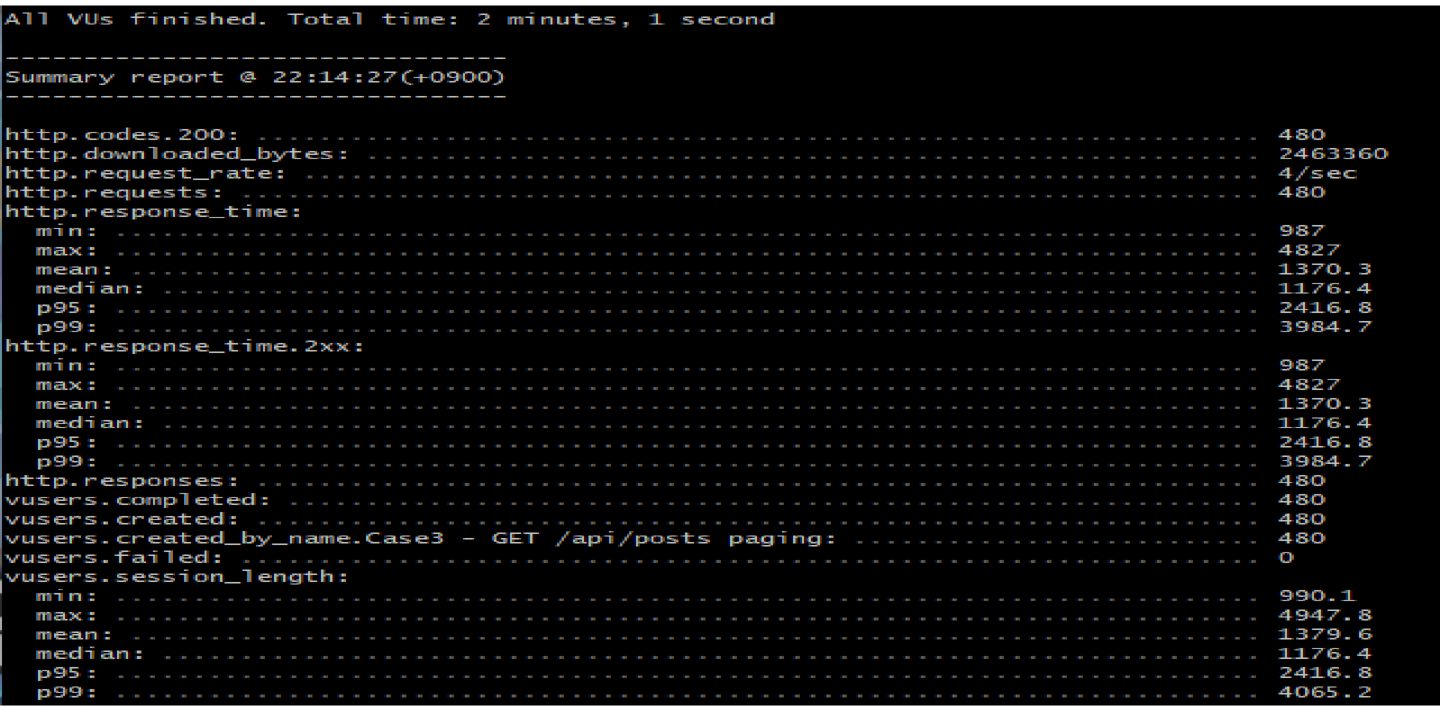
Hibernate SQL log showing COUNT query executed on posts only after structure separation.

Artillery Result Comparison



Metric	Before (COUNT JOIN)	After (COUNT Separated)	Improvement (%)
HTTP 200 Responses	480	480	0.0%
Requests	480	480	0.0%
Errors	0	0	0.0%
Min Response Time (ms)	987.0	765.0	**+22.5%**
Mean Response Time (ms)	1370.3	848.1	**+38.1%**
Median Response Time (ms)	1176.4	820.7	**+30.2%**
p95 Response Time (ms)	**2416.8**	**1022.7**	**+57.7%**
p99 Response Time (ms)	**3984.7**	**1176.4**	**+70.5%**
Max Response Time (ms)	4827.0	1462.0	**+69.7%**

Table: Artillery latency summary (http.response_time)

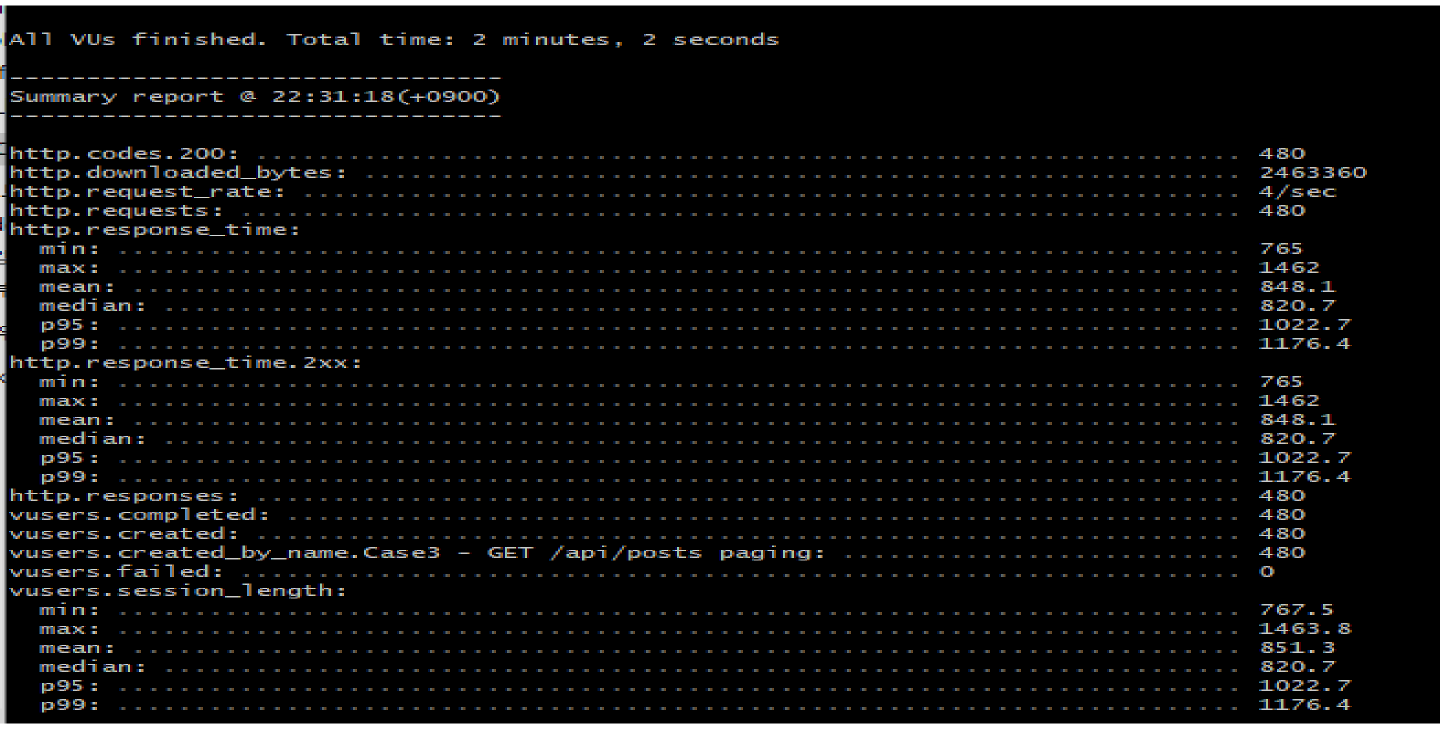


```
All VUs finished. Total time: 2 minutes, 1 second

--------------------------------
Summary report @ 22:14:27(+0900)
--------------------------------

http.codes.200: ........................................................... 480
http.downloaded_bytes: .................................................... 2463360
http.request_rate: ........................................................ 4/sec
http.requests: ............................................................ 480
http.response_time:
  min: .................................................................... 987
  max: .................................................................... 4827
  mean: ................................................................... 1370.3
  median: ................................................................. 1176.4
  p95: .................................................................... 2416.8
  p99: .................................................................... 3984.7
http.response_time.2xx:
  min: .................................................................... 987
  max: .................................................................... 4827
  mean: ................................................................... 1370.3
  median: ................................................................. 1176.4
  p95: .................................................................... 2416.8
  p99: .................................................................... 3984.7
http.responses: ........................................................... 480
vusers.completed: ......................................................... 480
vusers.created: ........................................................... 480
vusers.created_by_name.Case3 - GET /api/posts paging: ..................... 480
vusers.failed: ............................................................ 0
vusers.session_length:
  min: .................................................................... 990.1
  max: .................................................................... 4947.8
  mean: ................................................................... 1379.6
  median: ................................................................. 1176.4
  p95: .................................................................... 2416.8
  p99: .................................................................... 4065.2
```

Artillery summary report before structure separation.



```
All VUs finished. Total time: 2 minutes, 2 seconds

--------------------------------
Summary report @ 22:31:18(+0900)
--------------------------------

http.codes.200: ........................................................... 480
http.downloaded_bytes: .................................................... 2463360
http.request_rate: ........................................................ 4/sec
http.requests: ............................................................ 480
http.response_time:
  min: .................................................................... 765
  max: .................................................................... 1462
  mean: ................................................................... 848.1
  median: ................................................................. 820.7
  p95: .................................................................... 1022.7
  p99: .................................................................... 1176.4
http.response_time.2xx:
  min: .................................................................... 765
  max: .................................................................... 1462
  mean: ................................................................... 848.1
  median: ................................................................. 820.7
  p95: .................................................................... 1022.7
  p99: .................................................................... 1176.4
http.responses: ........................................................... 480
vusers.completed: ......................................................... 480
vusers.created: ........................................................... 480
vusers.created_by_name.Case3 - GET /api/posts paging: ..................... 480
vusers.failed: ............................................................ 0
vusers.session_length:
  min: .................................................................... 767.5
  max: .................................................................... 1463.8
  mean: ................................................................... 851.3
  median: ................................................................. 820.7
  p95: .................................................................... 1022.7
  p99: .................................................................... 1176.4
```

Artillery summary report after structure separation.

Conclusion

- **Result:**
  Successful Latency Reduction (P99: 5s → 1.4s)
- **Core Solution:**
  Separation of COUNT query from JOIN-based pagination query.
- **Observation:**
  Improved Latency and System Stability under Load.
- **Validation:**
  SQL Log / EXPLAIN / Artillery / Grafana

Status

Performance Stabilized, Latency Boundary Identified