بررسی تاثیر نمایه بر طرح اجرای پرسش‌ها در محیط سیستم استخدام

پرسش ۱

Get the timeline of interview stages for a specific candidate

```sql
EXPLAIN ANALYZE
SELECT
    s.stage_title,
    (i.interview_status).session_status AS session_status,
    (i.enter_exit_time).enter_time AS enter_time
FROM interview i
JOIN interview_type it ON i.interview_type_id = it.interview_type_id
JOIN stages s ON it.stage_id = s.stage_id
JOIN interviewee intv ON i.interviewee_id = intv.interviewee_id
JOIN requester r ON intv.requester_id = r.requester_id
JOIN person p ON r.person_id = p.person_id
WHERE p.person_name = 'Amy Kent'
ORDER BY enter_time;
```

نمایه‌ها

```sql
-- Speeds up person name lookup in WHERE clause
CREATE INDEX idx_person_name ON person(person_name);

-- Speeds up filtering by requester_id in JOIN condition
CREATE INDEX idx_interviewee_requester ON interviewee(requester_id);

-- Speeds up JOIN with interviewee table
CREATE INDEX idx_requester_person_id ON requester(person_id);
```

بدون نمایه

```
- Sort (cost=346.19..346.20, rows=1, width=70) (actual time=0.025..0.029, rows=0, loops=1)
  - Sort Key: ((i.enter_exit_time).enter_time)
  - Sort Method: quicksort, Memory: 25kB

  - Nested Loop (cost=325.30..346.18, rows=1, width=70) (actual time=0.013..0.017, rows=0, loops=1)
    - Nested Loop (cost=325.15..345.96, rows=1, width=68) (actual time=0.013..0.017, rows=0, loops=1)
      - Hash Join (cost=325.00..345.73, rows=1, width=68) (actual time=0.013..0.017, rows=0, loops=1)
        - Hash Condition: (i.interviewee_id = intv.interviewee_id)
        - Seq Scan on interview i (cost=0.00..17.80, rows=780, width=72) (actual time=0.010..0.010, rows=0, loops=1)
        - Hash (cost=324.98..324.98, rows=1, width=4) (never executed)
          - Nested Loop (cost=170.79..324.98, rows=1, width=4) (never executed)
            - Hash Join (cost=170.51..324.65, rows=1, width=4) (never executed)
              - Hash Condition: (r.person_id = p.person_id)
              - Seq Scan on requester r (cost=0.00..141.00, rows=5000, width=8) (never executed)
              - Hash (cost=170.50..170.50, rows=1, width=4) (never executed)
                - Seq Scan on person p (cost=0.00..170.50, rows=1, width=4) (never executed)
                  - Filter: ((person_name)::text = 'Amy Kent'::text)
            - Index Scan using interviewee_requester_id_key on interviewee intv (cost=0.28..0.34, rows=1, width=8) (never executed)
              - Index Condition: (requester_id = r.requester_id)

      - Index Scan using interview_type_pkey on interview_type it (cost=0.15..0.22, rows=1, width=8) (never executed)
        - Index Condition: (interview_type_id = i.interview_type_id)

    - Index Scan using stages_pkey on stages s (cost=0.15..0.23, rows=1, width=62) (never executed)
      - Index Condition: (stage_id = it.stage_id)

- Planning Time: 0.465 ms
- Execution Time: 0.083 ms
```

با نمایه

```
- Sort (cost=38.15..38.16, rows=1, width=70) (actual time=0.011..0.013, rows=0, loops=1)
  - Sort Key: ((i.enter_exit_time).enter_time)
  - Sort Method: quicksort, Memory: 25kB

  - Nested Loop (cost=17.26..38.14, rows=1, width=70) (actual time=0.007..0.008, rows=0, loops=1)
    - Nested Loop (cost=17.11..37.92, rows=1, width=68) (actual time=0.006..0.008, rows=0, loops=1)
      - Hash Join (cost=16.96..37.69, rows=1, width=68) (actual time=0.006..0.008, rows=0, loops=1)
        - Hash Condition: (i.interviewee_id = intv.interviewee_id)

        - Seq Scan on interview i (cost=0.00..17.80, rows=780, width=72) (actual time=0.006..0.006, rows=0, loops=1)

        - Hash (cost=16.95..16.95, rows=1, width=4) (never executed)
          - Nested Loop (cost=0.85..16.95, rows=1, width=4) (never executed)
            - Nested Loop (cost=0.56..16.61, rows=1, width=4) (never executed)
              - Index Scan using idx_person_name on person p (cost=0.28..8.30, rows=1, width=4) (never executed)
                - Index Condition: ((person_name)::text = 'Amy Kent'::text)

              - Index Scan using idx_requester_person_id on requester r (cost=0.28..8.30, rows=1, width=8) (never executed)
                - Index Condition: (person_id = p.person_id)

            - Index Scan using idx_interviewee_requester on interviewee intv (cost=0.28..0.34, rows=1, width=8) (never executed)
              - Index Condition: (requester_id = r.requester_id)

      - Index Scan using interview_type_pkey on interview_type it (cost=0.15..0.22, rows=1, width=8) (never executed)
        - Index Condition: (interview_type_id = i.interview_type_id)

    - Index Scan using stages_pkey on stages s (cost=0.15..0.23, rows=1, width=62) (never executed)
      - Index Condition: (stage_id = it.stage_id)

- Planning Time: 3.039 ms
- Execution Time: 0.052 ms
```

مراحل اجرای کوئری ابتدا با مرتب‌سازی داده‌ها بر اساس total_interview_time شروع می‌شود. سپس پایگاه داده با استفاده از اتصال‌های تو در تو، داده‌های مورد نیاز را از جداول مختلف مانند interview، interviewee، requester و person دریافت می‌کند. قبل از بهینه‌سازی، این فرآیند از اسکن‌های ترتیبی استفاده می‌کرد که باعث کندی اجرا می‌شد، اما پس از اضافه شدن نمایه‌ها، این اسکن‌ها به جستجوهای نمایه‌ای تغییر یافتند که سرعت بازیابی داده‌ها را افزایش داد. در مرحله بعد، مجموع زمان مصاحبه از اختلاف exit_time و enter_time محاسبه می‌شود و در نهایت، داده‌ها بر اساس مجموع زمان مصاحبه به ترتیب نزولی مرتب می‌شوند.

قبل از اضافه کردن نمایه‌ها، پایگاه داده مجبور بود کل جداول را اسکن کند، که باعث می‌شد پرهزینه باشد. از آنجایی که امکان جستجوی سریع داده‌ها وجود نداشت، از اسکن‌های ترتیبی استفاده می‌کرد که در نهایت هزینه اجرای کوئری را به ۳۴۶.۱۹ و زمان اجرای آن را به ۰.۰۸۳ میلی‌ثانیه می‌رساند.

بعد از اضافه کردن نمایه‌ها، شرایط کاملاً تغییر کرد. به جای اسکن کردن کل جدول، پایگاه داده توانست مستقیماً به داده‌های مورد نیاز دسترسی پیدا کند. این کار باعث شد که اتصال‌ها کارآمدتر شوند و پردازش‌های غیرضروری کاهش پیدا کنند. در نتیجه، هزینه اجرای کوئری به ۳۸.۱۵ کاهش یافت و زمان اجرا به ۰.۰۵۲ میلی‌ثانیه بهبود پیدا کرد.

پرسش ۲

Get employees who started working after receiving an offer

```
EXPLAIN ANALYZE
SELECT
    p.person_name,
    e.start_time,
    (o.offer_status).acceptence_status AS acceptence_status
FROM person p
JOIN employee e ON p.person_id = e.person_id
JOIN requester r ON e.person_id = r.person_id
JOIN interviewee i ON r.requester_id = i.requester_id
JOIN offer o ON i.interviewee_id = o.interviewee_id
WHERE (o.offer_status).acceptence_status = 'accepted'
ORDER BY e.start_time DESC;
```

نمایه‌ها

```
-- Optimizes search for accepted offers
CREATE INDEX idx_offer_acceptance_status ON offer USING BTREE (((offer_status).acceptence_status));

-- Speeds up lookup of employees by person_id
CREATE INDEX idx_employee_person_id ON employee(person_id);
```

بدون نمایه

```
- Sort (cost=227.85..227.86, rows=2, width=26) (actual time=3.640..3.642, rows=0, loops=1)
  - Sort Key: e.start_time DESC
  - Sort Method: quicksort, Memory: 25kB

  - Nested Loop (cost=115.53..227.84, rows=2, width=26) (actual time=3.633..3.635, rows=0, loops=1)
    - Join Filter: (p.person_id = e.person_id)
    - Nested Loop (cost=115.25..225.86, rows=5, width=53) (actual time=3.633..3.634, rows=0, loops=1)
      - Nested Loop (cost=115.09..221.45, rows=25, width=41) (actual time=0.764..3.024, rows=1733, loops=1)
        - Hash Join (cost=114.81..211.81, rows=25, width=41) (actual time=0.738..1.423, rows=1733, loops=1)
          - Hash Condition: (i.interviewee_id = o.interviewee_id)
          - Seq Scan on interviewee i (cost=0.00..78.00, rows=5000, width=8) (actual time=0.010..0.193, rows=5000, loops=1)
          - Hash (cost=114.50..114.50, rows=25, width=41) (actual time=0.714..0.714, rows=1733, loops=1)
            - Buckets: 2048 (originally 1024), Batches: 1 (originally 1), Memory Usage: 145kB
            - Seq Scan on offer o (cost=0.00..114.50, rows=25, width=41) (actual time=0.008..0.530, rows=1733, loops=1)
              - Filter: ((offer_status).acceptence_status = 'accepted'::acceptence_status_type)
              - Rows Removed by Filter: 3267

        - Index Scan using requester_pkey on requester r (cost=0.28..0.39, rows=1, width=8) (actual time=0.001..0.001, rows=1, loops=1733)
          - Index Condition: (requester_id = i.requester_id)

      - Index Scan using employee_person_id_key on employee e (cost=0.15..0.18, rows=1, width=12) (actual time=0.000..0.000, rows=0, loops=1733)
        - Index Condition: (person_id = r.person_id)

    - Index Scan using person_pkey on person p (cost=0.28..0.38, rows=1, width=18) (never executed)
      - Index Condition: (person_id = r.person_id)

- Planning Time: 0.843 ms
- Execution Time: 3.706 ms
```

با نمایه

```
- Sort (cost=162.11..162.11, rows=2, width=26) (actual time=3.264..3.266, rows=0, loops=1)
  - Sort Key: e.start_time DESC
  - Sort Method: quicksort, Memory: 25kB

  - Nested Loop (cost=49.78..162.10, rows=2, width=26) (actual time=3.260..3.262, rows=0, loops=1)
    - Join Filter: (p.person_id = e.person_id)

    - Nested Loop (cost=49.50..160.11, rows=5, width=53) (actual time=3.260..3.261, rows=0, loops=1)
      - Nested Loop (cost=49.35..155.71, rows=25, width=41) (actual time=0.453..2.631, rows=1733, loops=1)
        - Hash Join (cost=49.07..146.07, rows=25, width=41) (actual time=0.446..1.119, rows=1733, loops=1)
          - Hash Condition: (i.interviewee_id = o.interviewee_id)

          - Seq Scan on interviewee i (cost=0.00..78.00, rows=5000, width=8) (actual time=0.007..0.182, rows=5000, loops=1)

          - Hash (cost=48.75..48.75, rows=25, width=41) (actual time=0.430..0.431, rows=1733, loops=1)
            - Buckets: 2048, Batches: 1, Memory Usage: 145kB

            - Bitmap Heap Scan on offer o (cost=4.48..48.75, rows=25, width=41) (actual time=0.062..0.271, rows=1733, loops=1)
              - Recheck Condition: ((offer_status).acceptence_status = 'accepted'::acceptence_status_type)
              - Heap Blocks: exact=52

            - Bitmap Index Scan on idx_offer_acceptance_status (cost=0.00..4.47, rows=25, width=0) (actual time=0.049..0.049, rows=1733, loops=1)
              - Index Condition: ((offer_status).acceptence_status = 'accepted'::acceptence_status_type)

        - Index Scan using requester_pkey on requester r (cost=0.28..0.39, rows=1, width=8) (actual time=0.001..0.001, rows=1, loops=1733)
          - Index Condition: (requester_id = i.requester_id)

      - Index Scan using idx_employee_person_id on employee e (cost=0.15..0.18, rows=1, width=12) (actual time=0.000..0.000, rows=0, loops=1733)
        - Index Condition: (person_id = r.person_id)

    - Index Scan using person_pkey on person p (cost=0.28..0.38, rows=1, width=18) (never executed)
      - Index Condition: (person_id = r.person_id)

- Planning Time: 3.151 ms
- Execution Time: 3.326 ms
```

مراحل اجرای کوئری ابتدا با جستجوی پیشنهادهای پذیرفته‌شده شروع می‌شود که قبل از بهینه‌سازی به روش اسکن ترتیبی و بعد از بهینه‌سازی به روش اسکن نمایه‌ای انجام می‌شود. سپس داده‌ها با استفاده از اتصال‌های تو در تو از جداول interviewee، requester، employee و person استخراج می‌شوند. در نهایت، داده‌ها بر اساس start_time مرتب شده و نتیجه نهایی بازگردانده می‌شود.

قبل از اضافه کردن نمایه‌ها، پایگاه داده برای پیدا کردن پیشنهادهای پذیرفته‌شده، نیاز داشت کل جدول offer را اسکن کند. این فرآیند باعث افزایش هزینه اجرای کوئری تا ۲۲۷.۸۵ و زمان اجرا تا ۳.۷۰۶ میلی‌ثانیه شد. به دلیل عدم وجود نمایه مناسب، عملیات فیلتر کردن پیشنهادهای پذیرفته‌شده و اتصال جداول به روش اسکن ترتیبی انجام می‌شد که کارایی را پایین می‌آورد.

بعد از اضافه کردن نمایه‌ها، مخصوصاً نمایه روی وضعیت پذیرش پیشنهاد، پایگاه داده توانست از اسکن نمایه‌ای استفاده کند. این تغییر منجر به کاهش هزینه اجرای کوئری به ۱۶۲.۱۱ و بهبود زمان اجرا به ۳.۳۲۶ میلی‌ثانیه شد.

پرسش ۳

Find job openings that require the most skills

```sql
EXPLAIN ANALYZE
SELECT
    jo.title AS job_opening,
    COUNT(jos.skill_id) AS skill_count
FROM job_openning jo
JOIN job_openning_skills jos ON jo.job_openning_id = jos.job_openning_id
GROUP BY jo.title
ORDER BY skill_count DESC
LIMIT 5;
```

نمایه‌ها

```sql
-- Speeds up skill lookup for job openings
CREATE INDEX idx_job_openning_skills ON job_openning_skills(job_openning_id);
```

بدون نمایه

```
- Limit  (cost=107.27..107.29, rows=5, width=36) (actual time=0.030..0.031, rows=5, loops=1)
  - Sort  (cost=107.27..107.79, rows=208, width=36) (actual time=0.030..0.030, rows=5, loops=1)
    - Sort Key: (COUNT(jos.skill_id)) DESC
    - Sort Method: top-N heapsort, Memory: 25kB

    - HashAggregate  (cost=103.11..105.19, rows=208, width=36) (actual time=0.023..0.024, rows=18, loops=1)
      - Group Key: jo.title
      - Batches: 1  Memory Usage: 40kB

      - Hash Join  (cost=36.00..89.95, rows=2083, width=36) (actual time=0.006..0.015, rows=210, loops=1)
        - Hash Condition: (jo.job_openning_id = jos.job_openning_id)

        - Seq Scan on job_openning jo  (cost=0.00..20.83, rows=1083, width=36) (actual time=0.002..0.004, rows=1083, loops=1)

        - Hash  (cost=35.00..35.00, rows=1000, width=8) (actual time=0.003..0.003, rows=1000, loops=1)
          - Seq Scan on job_openning_skills jos  (cost=0.00..35.00, rows=1000, width=8) (actual time=0.001..0.002, rows=1000, loops=1)

- Planning Time: 0.284 ms
- Execution Time: 0.049 ms
```

با نمایه

```
- Limit (cost=123.38..123.39, rows=5, width=29) (actual time=0.011..0.012, rows=0, loops=1)

  - Sort (cost=123.38..123.88, rows=200, width=29) (actual time=0.010..0.011, rows=0, loops=1)
    - Sort Key: (count(jos.skill_id)) DESC
    - Sort Method: quicksort, Memory: 25kB

    - HashAggregate (cost=118.05..120.05, rows=200, width=29) (actual time=0.006..0.007, rows=0, loops=1)
      - Group Key: jo.title
      - Batches: 1, Memory Usage: 40kB

      - Hash Join (cost=22.80..106.75, rows=2260, width=25) (actual time=0.004..0.005, rows=0, loops=1)
        - Hash Condition: (jos.job_openning_id = jo.job_openning_id)

        - Index Scan using idx_job_openning_skills_job_openning_id on job_openning_skills jos (cost=0.15..78.06, rows=2260, width=8) (actual time=0.004..0.004, rows=0, loops=1)

        - Hash (cost=20.14..20.14, rows=200, width=25) (never executed)
          - Index Scan using job_openning_pkey on job_openning jo (cost=0.14..20.14, rows=200, width=25) (never executed)

- Planning Time: 0.675 ms
- Execution Time: 0.046 ms
```

قبل از اضافه کردن نمایه، پایگاه داده مجبور بود برای یافتن تعداد مهارت‌های مربوط به هر عنوان شغلی، از اسکن ترتیبی استفاده کند. این روند باعث می‌شد که عملیات به طور کلی کندتر و پرهزینه‌تر باشد. در این حالت، هزینه اجرای کوئری ۱۰۷.۲۷ بود و زمان اجرای آن به ۰.۰۴۹ میلی‌ثانیه رسید.

بعد از اضافه کردن نمایه هزینه اجرای کوئری مقداری افزایش یافت و به ۱۲۳.۳۸ رسید ولی زمان اجرا به ۰.۰۴۶ میلی‌ثانیه رسید که کمی کاهش داشت. در این حالت، نمایه باعث شد که اسکن جداول به جای اسکن ترتیبی به اسکن نمایه‌ای تبدیل شود، که منجر به بهبود عملکرد کوئری شد.

پرسش ۴

Find candidates who attended multiple interviews

```sql
EXPLAIN ANALYZE
SELECT
    p.person_name,
    COUNT(i.interview_id) AS total_interviews
FROM interview i
JOIN interviewee intv ON i.interviewee_id = intv.interviewee_id
JOIN requester r ON intv.requester_id = r.requester_id
JOIN person p ON r.person_id = p.person_id
GROUP BY p.person_name
HAVING COUNT(i.interview_id) > 1
ORDER BY total_interviews DESC;
```

نمایه‌ها

```sql
CREATE INDEX idx_interview_interviewee ON interview(interviewee_id);
```

بدون نمایه

```
- Sort (cost=558.53..559.18, rows=260, width=22) (actual time=0.024..0.025, rows=0, loops=1)
  - Sort Key: (COUNT(i.interview_id)) DESC
  - Sort Method: quicksort, Memory: 25kB

  - HashAggregate (cost=538.35..548.10, rows=260, width=22) (actual time=0.021..0.022, rows=0, loops=1)
    - Group Key: p.person_name
    - Filter: (COUNT(i.interview_id) > 1)
    - Batches: 1, Memory Usage: 49kB

    - Hash Join (cost=347.40..534.45, rows=780, width=18) (actual time=0.019..0.020, rows=0, loops=1)
      - Hash Condition: (p.person_id = r.person_id)

      - Seq Scan on person p (cost=0.00..153.00, rows=7000, width=18) (actual time=0.007..0.007, rows=1, loops=1)

      - Hash (cost=337.65..337.65, rows=780, width=8) (actual time=0.009..0.010, rows=0, loops=1)
        - Buckets: 1024, Batches: 1, Memory Usage: 8kB

        - Hash Join (cost=170.10..337.65, rows=780, width=8) (actual time=0.009..0.009, rows=0, loops=1)
          - Hash Condition: (r.requester_id = intv.requester_id)

          - Seq Scan on requester r (cost=0.00..141.00, rows=5000, width=8) (actual time=0.004..0.004, rows=1, loops=1)

          - Hash (cost=160.35..160.35, rows=780, width=8) (actual time=0.003..0.004, rows=0, loops=1)
            - Buckets: 1024, Batches: 1, Memory Usage: 8kB

            - Hash Join (cost=140.50..160.35, rows=780, width=8) (actual time=0.003..0.003, rows=0, loops=1)
              - Hash Condition: (i.interviewee_id = intv.interviewee_id)

              - Seq Scan on interview i (cost=0.00..17.80, rows=780, width=8) (actual time=0.003..0.003, rows=0, loops=1)

              - Hash (cost=78.00..78.00, rows=5000, width=8) (never executed)
                - Seq Scan on interviewee intv (cost=0.00..78.00, rows=5000, width=8) (never executed)

- Planning Time: 0.273 ms
- Execution Time: 0.081 ms
```

با نمایه

```
- Sort (cost=841.56..842.21, rows=260, width=22) (actual time=0.021..0.022, rows=0, loops=1)
  - Sort Key: (count(i.interview_id)) DESC
  - Sort Method: quicksort, Memory: 25kB

  - HashAggregate (cost=821.39..831.14, rows=260, width=22) (actual time=0.018..0.018, rows=0, loops=1)
    - Group Key: p.person_name
    - Filter: (count(i.interview_id) > 1)
    - Batches: 1, Memory Usage: 49kB

    - Nested Loop (cost=258.70..817.49, rows=780, width=18) (actual time=0.016..0.017, rows=0, loops=1)
      - Hash Join (cost=258.41..517.96, rows=780, width=8) (actual time=0.016..0.016, rows=0, loops=1)
        - Hash Condition: (r.requester_id = intv.requester_id)

        - Index Scan using requester_pkey on requester r (cost=0.28..233.28, rows=5000, width=8) (actual time=0.007..0.007, rows=1, loops=1)

        - Hash (cost=248.38..248.38, rows=780, width=8) (actual time=0.003..0.004, rows=0, loops=1)
          - Buckets: 1024, Batches: 1, Memory Usage: 8kB
          - Merge Join (cost=0.43..248.38, rows=780, width=8) (actual time=0.003..0.003, rows=0, loops=1)
            - Merge Condition: (i.interviewee_id = intv.interviewee_id)

            - Index Scan using idx_interview_interviewee_id on interview i (cost=0.15..55.85, rows=780, width=8) (actual time=0.002..0.002, rows=0, loops=1)

            - Index Scan using interviewee_pkey on interviewee intv (cost=0.28..170.28, rows=5000, width=8) (never executed)

      - Index Scan using person_pkey on person p (cost=0.28..0.38, rows=1, width=18) (never executed)
        - Index Condition: (person_id = r.person_id)

- Planning Time: 2.542 ms
- Execution Time: 0.055 ms
```

قبل از ایجاد نمایه‌ها، پایگاه داده برای اجرای این کوئری مجبور بود از اسکن ترتیبی برای جستجو در جداول استفاده کند. این باعث می‌شد که عملیات کندتر و پرهزینه‌تر باشد. در این وضعیت، هزینه اجرای کوئری ۵۵۸.۵۳ و زمان اجرای واقعی حدود ۰.۰۸۱ میلی‌ثانیه بود. در این روند، ابتدا جداول مختلف به یکدیگر متصل می‌شدند، سپس تعداد مصاحبه‌ها برای هر فرد شمارش می‌شد و پس از آن نتایج مرتب می‌شدند.

بعد از اضافه کردن نمایه‌ها هزینه اجرای کوئری به ۸۴۱.۵۶ افزایش یافت اما زمان اجرای آن به ۰.۰۵۵ میلی‌ثانیه کاهش یافت.

پرسش ۵

Find the total time spent in interviews per candidate

```
EXPLAIN ANALYZE
SELECT
    p.person_name,
    SUM(((i.enter_exit_time).exit_time - (i.enter_exit_time).enter_time)) AS total_interview_time
FROM interview i
JOIN interviewee intv ON i.interviewee_id = intv.interviewee_id
JOIN requester r ON intv.requester_id = r.requester_id
JOIN person p ON r.person_id = p.person_id
GROUP BY p.person_name
ORDER BY total_interview_time DESC;
```

نمایه‌ها

```
CREATE INDEX idx_interview_interviewee ON interview(interviewee_id);
```

بدون نمایه

```
- Sort (cost=587.52..589.47, rows=780, width=30) (actual time=0.069..0.071, rows=0, loops=1)
  - Sort Key: (SUM(((i.enter_exit_time).exit_time - (i.enter_exit_time).enter_time))) DESC
  - Sort Method: quicksort, Memory: 25kB

  - HashAggregate (cost=540.30..550.05, rows=780, width=30) (actual time=0.062..0.064, rows=0, loops=1)
    - Group Key: p.person_name
    - Batches: 1, Memory Usage: 49kB

    - Hash Join (cost=347.40..534.45, rows=780, width=46) (actual time=0.056..0.058, rows=0, loops=1)
      - Hash Condition: (p.person_id = r.person_id)

      - Seq Scan on person p (cost=0.00..153.00, rows=7000, width=18) (actual time=0.010..0.010, rows=1, loops=1)

      - Hash (cost=337.65..337.65, rows=780, width=36) (actual time=0.024..0.025, rows=0, loops=1)
        - Buckets: 1024, Batches: 1, Memory Usage: 8kB

        - Hash Join (cost=170.10..337.65, rows=780, width=36) (actual time=0.023..0.025, rows=0, loops=1)
          - Hash Condition: (r.requester_id = intv.requester_id)

          - Seq Scan on requester r (cost=0.00..141.00, rows=5000, width=8) (actual time=0.009..0.009, rows=1, loops=1)

          - Hash (cost=160.35..160.35, rows=780, width=36) (actual time=0.006..0.007, rows=0, loops=1)
            - Buckets: 1024, Batches: 1, Memory Usage: 8kB

            - Hash Join (cost=140.50..160.35, rows=780, width=36) (actual time=0.006..0.007, rows=0, loops=1)
              - Hash Condition: (i.interviewee_id = intv.interviewee_id)

              - Seq Scan on interview i (cost=0.00..17.80, rows=780, width=36) (actual time=0.005..0.005, rows=0, loops=1)

              - Hash (cost=78.00..78.00, rows=5000, width=8) (never executed)
                - Seq Scan on interviewee intv (cost=0.00..78.00, rows=5000, width=8) (never executed)

- Planning Time: 0.357 ms
- Execution Time: 0.120 ms
```

با نمایه

```
Sort  (cost=814.55..816.25 rows=680 width=30) (actual time=0.011..0.011 rows=0 loops=1)
  Sort Key: (sum(((i.enter_exit_time).exit_time - (i.enter_exit_time).enter_time))) DESC
  Sort Method: quicksort  Memory: 25kB
  ->  HashAggregate  (cost=774.06..782.56 rows=680 width=30) (actual time=0.006..0.007 rows=0 loops=1)
        Group Key: p.person_name
        Batches: 1  Memory Usage: 49kB
        ->  Nested Loop  (cost=1.00..768.96 rows=680 width=46) (actual time=0.004..0.004 rows=0 loops=1)
              ->  Nested Loop  (cost=0.71..507.84 rows=680 width=36) (actual time=0.004..0.004 rows=0 loops=1)
                    ->  Merge Join  (cost=0.43..245.63 rows=680 width=36) (actual time=0.004..0.004 rows=0 loops=1)
                          Merge Cond: (i.interviewee_id = intv.interviewee_id)
                          ->  Index Scan using idx_interview_interviewee on interview i  (cost=0.15..54.35 rows=680 width=36) (actual time=0.003..0.003 rows=0 loops=1)
                          ->  Index Scan using interviewee_pkey on interviewee intv  (cost=0.28..170.28 rows=5000 width=8) (never executed)
                    ->  Index Scan using requester_pkey on requester r  (cost=0.28..0.39 rows=1 width=8) (never executed)
                          Index Cond: (requester_id = intv.requester_id)
              ->  Index Scan using person_pkey on person p  (cost=0.28..0.38 rows=1 width=18) (never executed)
                    Index Cond: (person_id = r.person_id)
Planning Time: 0.470 ms
Execution Time: 0.069 ms
```

قبل از اضافه کردن نمایه‌ها، پایگاه داده باید از اسکن ترتیبی استفاده می‌کرد تا داده‌ها را از جداول مختلف مانند استخراج کند و آنها را برای محاسبه زمان مصاحبه‌ها به هم پیوند دهد. در این حالت، هزینه اجرای کوئری برابر با ۵۴۰.۳۰ و زمان اجرای واقعی برابر با ۰.۱۲۰ میلی‌ثانیه بود.

بعد از اضافه کردن نمایه روی فیلد interviewee_id در جدول interview، پایگاه داده از اسکن نمایه‌ای برای جستجوی سریعتر استفاده کزد. این موجب کاهش زمان اجرا شد. هزینه اجرای کوئری به ۸۱۴.۵۵ افزایش یافت، اما زمان واقعی اجرای آن به ۰.۰۶۹ میلی‌ثانیه کاهش یافت.

پرسش ۶

query1.sql

```sql
-- get the highest scoring people for each skill
EXPLAIN ANALYZE
SELECT s.skill_id, s.title AS skill_title, p.person_name, ic.interviewer_score
FROM interviewer_comments ic
JOIN skills s ON ic.skill_id = s.skill_id
JOIN interviewer i ON ic.interviewer_id = i.interviewer_id
JOIN employee e ON i.interviewer_id = e.employee_id
JOIN person p ON e.person_id = p.person_id
WHERE ic.interviewer_score = (
    SELECT MAX(ic2.interviewer_score)
    FROM interviewer_comments ic2
    WHERE ic2.skill_id = ic.skill_id
);
```

index1.sql

```sql
CREATE INDEX idx_comments_interviewer_score ON interviewer_comments(interviewer_score);
-- DROP INDEX idx_comments_interviewer_score;
```

این پرسش افراد با بالاترین امتیاز در هر مهارت را نشان می‌دهد. با ایجاد نمایه روی ستون امتیاز در جدول interviewer_comments، پرسش سریع‌تر می‌شود. طرح اجرا به این صورت است:

بی نمایه

```
                                                        QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.42..1501364333.15 rows=1015 width=37)
   Join Filter: (e.person_id = p.person_id)
   ->  Seq Scan on person p  (cost=0.00..84.00 rows=2500 width=18)
   ->  Materialize  (cost=0.42..1501326189.19 rows=1015 width=27)
         ->  Nested Loop  (cost=0.42..1501326184.11 rows=1015 width=27)
               Join Filter: (ic.interviewer_id = e.employee_id)
               ->  Nested Loop  (cost=0.42..1501320103.03 rows=1015 width=31)
                     ->  Seq Scan on skills s  (cost=0.00..1.54 rows=54 width=19)
                     ->  Nested Loop  (cost=0.42..27802220.10 rows=400 width=16)
                           ->  Seq Scan on interviewer i  (cost=0.00..7.00 rows=400 width=4)
                           ->  Index Scan using interviewer_comments_pkey on interviewer_comments ic  (cost=0.42..69505.52 rows=1 width=12)
                                 Index Cond: ((skill_id = s.skill_id) AND (interviewer_id = i.interviewer_id))
                                 Filter: (interviewer_score = (SubPlan 1))
                                 SubPlan 1
                                   ->  Aggregate  (cost=7485.65..7485.66 rows=1 width=4)
                                         ->  Seq Scan on interviewer_comments ic2  (cost=0.00..7476.25 rows=3760 width=4)
                                               Filter: (skill_id = ic.skill_id)
               ->  Materialize  (cost=0.00..10.00 rows=400 width=8)
                     ->  Seq Scan on employee e  (cost=0.00..8.00 rows=400 width=8)
(19 rows)
```

در این حالت به دلیل بزرگی جدول interview_comments و مدت زمان طولانی پرسش، موفق به اجرای EXPLAIN ANALYZE نشدیم و طرح اجرا را تنها با EXPLAIN به دست آوردیم. طبق این طرح اجرا، ابتدا subquery انجام می‌شود و یک بار اسکن ترتیبی روی جدول interviewer_comments داریم تا با خودش روی skill_id یکسان ادغام شود. در ادامه با استفاده از کلید اصلی جدول comments بیرونی به skills و interviewer می‌پیوندد و سطرهایی از آن که score نابرابر با حاصل subquery دارند فیلتر می‌شوند. در نهایت با جداول employee و person ادغام می‌شود.

با نمایه

```
                                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.28..1349401.27 rows=1015 width=37) (actual time=0.246..3828.825 rows=40454 loops=1)
   ->  Nested Loop  (cost=0.00..1348277.16 rows=1015 width=27) (actual time=0.232..3786.198 rows=40454 loops=1)
         Join Filter: (ic.interviewer_id = e.employee_id)
         Rows Removed by Join Filter: 8073726
         ->  Nested Loop  (cost=0.00..1342196.07 rows=1015 width=31) (actual time=0.151..3199.278 rows=40454 loops=1)
               Join Filter: (ic.interviewer_id = i.interviewer_id)
               Rows Removed by Join Filter: 8073726
               ->  Nested Loop  (cost=0.00..1336115.98 rows=1015 width=27) (actual time=0.078..2611.106 rows=40454 loops=1)
                     Join Filter: (ic.skill_id = s.skill_id)
                     Rows Removed by Join Filter: 1071494
                     ->  Seq Scan on interviewer_comments ic  (cost=0.00..1335306.99 rows=1015 width=12) (actual time=0.057..2523.527 rows=40454 loops=1)
                           Filter: (interviewer_score = (SubPlan 2))
                           Rows Removed by Filter: 162606
                           SubPlan 2
                             ->  Result  (cost=6.53..6.54 rows=1 width=4) (actual time=0.012..0.012 rows=1 loops=203060)
                                   InitPlan 1
                                     ->  Limit  (cost=0.29..6.53 rows=1 width=4) (actual time=0.012..0.012 rows=1 loops=203060)
                                           ->  Index Scan Backward using idx_comments_interviewer_score on interviewer_comments ic2  (cost=0.29..23440.53 rows=3760 width=4) (actual time=0.012..0.012 rows=1 loops=203060)
                                                 Filter: (skill_id = ic.skill_id)
                                                 Rows Removed by Filter: 123
                     ->  Materialize  (cost=0.00..1.81 rows=54 width=19) (actual time=0.000..0.001 rows=27 loops=40454)
                           ->  Seq Scan on skills s  (cost=0.00..1.54 rows=54 width=19) (actual time=0.007..0.009 rows=54 loops=1)
               ->  Materialize  (cost=0.00..9.00 rows=400 width=4) (actual time=0.000..0.006 rows=201 loops=40454)
                     ->  Seq Scan on interviewer i  (cost=0.00..7.00 rows=400 width=4) (actual time=0.009..0.032 rows=400 loops=1)
         ->  Materialize  (cost=0.00..10.00 rows=400 width=8) (actual time=0.000..0.006 rows=201 loops=40454)
               ->  Seq Scan on employee e  (cost=0.00..8.00 rows=400 width=8) (actual time=0.010..0.045 rows=400 loops=1)
   ->  Index Scan using person_pkey on person p  (cost=0.28..1.11 rows=1 width=18) (actual time=0.001..0.001 rows=1 loops=40454)
         Index Cond: (person_id = e.person_id)
 Planning Time: 6.513 ms
 Execution Time: 3830.410 ms
(30 rows)
```

با اضافه کردن نمایه روی ستون score در جدول interviewer_comments پیدا کردن بیشترین امتیاز سریع‌تر می‌شود. چرا که جدول به ترتیب عکس امتیازها (با شروع از بیشترین امتیاز) با جدول skills می‌پیوندد و زمانی که بیشترین امتیاز برای آن امتیاز پیدا شد دیگر باقی جدول چک نمی‌شود. با این کار هزینه پرسش به شدت کاهش می‌یابد اما همچنان تطابق بیشترین امتیاز با خود فرد به صورت ترتیبی انجام می‌شود.

پرسش ۷

```sql
-- query2.sql
-- the requester with the lowest expected salary for each job opening
EXPLAIN ANALYZE
SELECT r.job_openning_id, jo.title AS job_opening_title, p.person_name, r.salary_expectation
FROM requester r
JOIN person p ON r.person_id = p.person_id
JOIN job_openning jo ON r.job_openning_id = jo.job_openning_id
WHERE r.salary_expectation = (
    SELECT MIN(r2.salary_expectation)
    FROM requester r2
    WHERE r2.job_openning_id = r.job_openning_id
);
```

```sql
-- index2.sql
CREATE INDEX idx_requester_job_openning_id ON requester(job_openning_id);
-- DROP INDEX idx_requester_job_openning_id;
```

این پرسش برای هر موقعیت شغلی درخواست‌دهنده با کمترین حقوق درخواستی را می‌دهد.

بی نمایه

```
                                                     QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.28..15329.46 rows=4 width=44) (actual time=19.381..32.475 rows=281 loops=1)
   Join Filter: (jo.job_openning_id = r.job_openning_id)
   Rows Removed by Join Filter: 84019
   ->  Seq Scan on job_openning jo  (cost=0.00..21.00 rows=300 width=25) (actual time=0.013..0.090 rows=300 loops=1)
   ->  Materialize  (cost=0.28..15290.47 rows=4 width=23) (actual time=0.001..0.096 rows=281 loops=300)
         ->  Nested Loop  (cost=0.28..15290.45 rows=4 width=23) (actual time=0.184..26.177 rows=281 loops=1)
               ->  Seq Scan on requester r  (cost=0.00..15257.25 rows=4 width=13) (actual time=0.166..25.602 rows=281 loops=1)
                     Filter: (salary_expectation = (SubPlan 1))
                     Rows Removed by Filter: 419
                     SubPlan 1
                       ->  Aggregate  (cost=21.75..21.77 rows=1 width=32) (actual time=0.036..0.036 rows=1 loops=700)
                             ->  Seq Scan on requester r2  (cost=0.00..21.75 rows=2 width=5) (actual time=0.009..0.033 rows=3 loops=700)
                                   Filter: (job_openning_id = r.job_openning_id)
                                   Rows Removed by Filter: 697
               ->  Index Scan using person_pkey on person p  (cost=0.28..8.30 rows=1 width=18) (actual time=0.002..0.002 rows=1 loops=281)
                     Index Cond: (person_id = r.person_id)
 Planning Time: 6.003 ms
 Execution Time: 32.576 ms
(18 rows)
```

طرح اجرا در این حالت ابتدا subquery را که کمترین حقوق درخواستی برای آن موقعیت را پیدا می‌کند انجام می‌دهد. سپس سطرهایی از جدول requester که حقوق درخواستی‌شان با این مقدار برابر است را پیدا می‌کند و با جدول person می‌پیوندد. در نهایت حاصل را با جدول job_openning ادغام می‌کند.

با نمایه

```
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.56..6205.96 rows=4 width=44) (actual time=0.072..5.605 rows=281 loops=1)
   ->  Nested Loop  (cost=0.28..6172.76 rows=4 width=34) (actual time=0.054..5.033 rows=281 loops=1)
         ->  Seq Scan on job_openning jo  (cost=0.00..21.00 rows=300 width=25) (actual time=0.010..0.090 rows=300 loops=1)
         ->  Index Scan using idx_requester_job_openning_id on requester r  (cost=0.28..20.50 rows=1 width=13) (actual time=0.012..0.016 rows=1 loops=300)
               Index Cond: (job_openning_id = jo.job_openning_id)
               Filter: (salary_expectation = (SubPlan 1))
               Rows Removed by Filter: 1
               SubPlan 1
                 ->  Aggregate  (cost=9.97..9.98 rows=1 width=32) (actual time=0.006..0.006 rows=1 loops=700)
                       ->  Bitmap Heap Scan on requester r2  (cost=4.29..9.96 rows=2 width=5) (actual time=0.002..0.002 rows=3 loops=700)
                             Recheck Cond: (job_openning_id = r.job_openning_id)
                             Heap Blocks: exact=2089
                             ->  Bitmap Index Scan on idx_requester_job_openning_id  (cost=0.00..4.29 rows=2 width=0) (actual time=0.001..0.001 rows=3 loops=700)
                                   Index Cond: (job_openning_id = r.job_openning_id)
   ->  Index Scan using person_pkey on person p  (cost=0.28..8.30 rows=1 width=18) (actual time=0.002..0.002 rows=1 loops=281)
         Index Cond: (person_id = r.person_id)
 Planning Time: 6.216 ms
 Execution Time: 5.705 ms
(18 rows)
```

داشتن نمایه روی ستون openning_id در جدول job_openning، subquery را سریع‌تر می‌کند. باقی طرح اجرا مانند قبل است با این تفاوت که قبل از پیوند با جدول person، requester را با jo می‌پیوندیم تا از همین نمایه دوباره استفاده کنیم و اجرای پرسش سریع‌تر شود.

پرسش ۸

query3.sql

```sql
EXPLAIN ANALYZE
SELECT
    dm.interviewer_id,
    COUNT(dm.*) AS message_count
FROM
    discussion_member dm
WHERE
    dm.interviewer_text_date = NOW() - INTERVAL '10 days'
GROUP BY
    dm.interviewer_id
ORDER BY
    message_count DESC;
```

index3.sql

```sql
CREATE INDEX idx_dismem_date ON discussion_member (interviewer_text_date);
-- DROP INDEX idx_dismem_date;
```

این پرسش مصاحبه‌کنندگان را به ترتیب فعالیت در گفت و گوها در ده روز گذشته نشان می‌دهد.

بی نمایه

```
                                                            QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=84.36..85.31 rows=381 width=12) (actual time=1.039..1.050 rows=366 loops=1)
   Sort Key: (count(dm.*)) DESC
   Sort Method: quicksort  Memory: 36kB
   ->  HashAggregate  (cost=64.22..68.03 rows=381 width=12) (actual time=0.946..0.978 rows=366 loops=1)
         Group Key: interviewer_id
         Batches: 1  Memory Usage: 61kB
         ->  Seq Scan on discussion_member dm  (cost=0.00..59.62 rows=919 width=196) (actual time=0.023..0.815 rows=925 loops=1)
               Filter: (interviewer_text_date < (now() - '10 days'::interval))
               Rows Removed by Filter: 482
 Planning Time: 1.269 ms
 Execution Time: 1.128 ms
(11 rows)
```

در این حالت برای فیلتر زمان ارسال پیام در گفت‌وگو نیاز است جدول پیام‌ها به صورت ترتیبی اسکن شود.

با نمایه

```
                                                              QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=84.36..85.31 rows=381 width=12) (actual time=0.778..0.787 rows=366 loops=1)
   Sort Key: (count(dm.*)) DESC
   Sort Method: quicksort  Memory: 36kB
   ->  HashAggregate  (cost=64.22..68.03 rows=381 width=12) (actual time=0.702..0.730 rows=366 loops=1)
         Group Key: interviewer_id
         Batches: 1  Memory Usage: 61kB
         ->  Seq Scan on discussion_member dm  (cost=0.00..59.62 rows=919 width=196) (actual time=0.020..0.586 rows=925 loops=1)
               Filter: (interviewer_text_date < (now() - '10 days'::interval))
               Rows Removed by Filter: 482
 Planning Time: 1.535 ms
 Execution Time: 0.863 ms
(11 rows)
```

با داشتن نمایه روی تاریخ ارسال پیام می‌توان از جست‌وجوی ترتیبی جدول پرهیز کرد و سرعت پرسش را بالا برد.

پرسش ۹

query9.sql

```sql
-- the rejected offers with the highest payment for each job opening
EXPLAIN ANALYZE
SELECT
    o.job_openning_id,
    jo.title AS job_openning_title,
    o.offer_id,
    o.recommended_payment,
    p.person_name AS requester_name
FROM
    offer o
JOIN
    job_openning jo ON o.job_openning_id = jo.job_openning_id
JOIN
    interviewee i ON i.interviewee_id = o.interviewee_id
JOIN
    requester r ON i.requester_id = r.requester_id
JOIN
    person p ON r.person_id = p.person_id
WHERE
    (o.offer_status).acceptence_status = 'rejected'
    AND o.recommended_payment = (
        SELECT MAX(o2.recommended_payment)
        FROM offer o2
        WHERE o2.job_openning_id = o.job_openning_id
        AND (o2.offer_status).acceptence_status = 'rejected'
    )
ORDER BY
    o.recommended_payment DESC;
```

index9.sql

```sql
CREATE INDEX idx_offer_job_openning_id ON offer(job_openning_id);
CREATE INDEX idx_offer_recommended_payment ON offer(recommended_payment);
-- DROP INDEX idx_offer_job_openning_id;
-- DROP INDEX idx_offer_recommended_payment;
```

این پرسش پیشنهاد رد شده با بالاترین حقوق پیشنهادی را برای هر موقعیت شغلی نشان می‌دهد.

بی نمایه

```
                                                    QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=13165.76..13165.77 rows=1 width=48) (actual time=39.973..39.989 rows=153 loops=1)
   Sort Key: o.recommended_payment DESC
   Sort Method: quicksort  Memory: 35kB
   ->  Nested Loop  (cost=0.00..13165.75 rows=1 width=48) (actual time=0.474..39.862 rows=153 loops=1)
         Join Filter: (p.person_id = r.person_id)
         Rows Removed by Join Filter: 198801
         ->  Nested Loop  (cost=0.00..13050.50 rows=1 width=38) (actual time=0.224..22.518 rows=153 loops=1)
               Join Filter: (i.requester_id = r.requester_id)
               Rows Removed by Join Filter: 53013
               ->  Nested Loop  (cost=0.00..13021.75 rows=1 width=38) (actual time=0.214..17.491 rows=153 loops=1)
                     Join Filter: (i.interviewee_id = o.interviewee_id)
                     Rows Removed by Join Filter: 53013
                     ->  Nested Loop  (cost=0.00..13002.00 rows=1 width=38) (actual time=0.202..12.455 rows=153 loops=1)
                           Join Filter: (jo.job_openning_id = o.job_openning_id)
                           Rows Removed by Join Filter: 24066
                           ->  Seq Scan on offer o  (cost=0.00..12977.25 rows=1 width=17) (actual time=0.145..9.557 rows=153 loops=1)
                                 Filter: (((offer_status).acceptence_status = 'rejected'::acceptence_status_type) AND (recommended_payment = (SubPlan 1)))
                                 Rows Removed by Filter: 547
                                 SubPlan 1
                                   ->  Aggregate  (cost=18.50..18.51 rows=1 width=32) (actual time=0.040..0.040 rows=1 loops=235)
                                         ->  Seq Scan on offer o2  (cost=0.00..18.50 rows=1 width=5) (actual time=0.014..0.037 rows=2 loops=235)
                                               Filter: ((job_openning_id = o.job_openning_id) AND ((offer_status).acceptence_status = 'rejected'::acceptence_status_type))
                                               Rows Removed by Filter: 698
                           ->  Seq Scan on job_openning jo  (cost=0.00..21.00 rows=300 width=25) (actual time=0.000..0.009 rows=158 loops=153)
                     ->  Seq Scan on interviewee i  (cost=0.00..11.00 rows=700 width=8) (actual time=0.001..0.015 rows=347 loops=153)
               ->  Seq Scan on requester r  (cost=0.00..20.00 rows=700 width=8) (actual time=0.001..0.016 rows=347 loops=153)
         ->  Seq Scan on person p  (cost=0.00..84.00 rows=2500 width=18) (actual time=0.004..0.054 rows=1300 loops=153)
 Planning Time: 7.511 ms
 Execution Time: 40.122 ms
(29 rows)
```

ابتدا subquery انجام می‌شود و سپس به offer می‌پیوندد. تا اینجا تمام پیوندها به صورت اسکن ترتیبی انجام می‌شوند البته بیشتر آن‌ها از نوع پیوند FK-PK هستند. سپس با interviewee و requester و person پیوند داریم. در نهایت مرتب‌سازی با quicksort انجام می‌شود.

با نمایه

```
                                                                QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.56..7726.99 rows=1 width=48) (actual time=0.337..15.217 rows=153 loops=1)
   ->  Nested Loop  (cost=0.28..7726.23 rows=1 width=38) (actual time=0.321..14.806 rows=153 loops=1)
         Join Filter: (i.requester_id = r.requester_id)
         Rows Removed by Join Filter: 53013
         ->  Nested Loop  (cost=0.28..7697.48 rows=1 width=38) (actual time=0.227..9.767 rows=153 loops=1)
               Join Filter: (i.interviewee_id = o.interviewee_id)
               Rows Removed by Join Filter: 53013
               ->  Nested Loop  (cost=0.28..7677.73 rows=1 width=38) (actual time=0.167..4.778 rows=153 loops=1)
                     Join Filter: (o.job_openning_id = jo.job_openning_id)
                     Rows Removed by Join Filter: 24066
                     ->  Index Scan Backward using idx_offer_recommended_payment on offer o  (cost=0.28..7652.98 rows=1 width=17) (actual time=0.086..1.873 rows=153 loops=1)
                           Filter: (((offer_status).acceptence_status = 'rejected'::acceptence_status_type) AND (recommended_payment = (SubPlan 1)))
                           Rows Removed by Filter: 547
                           SubPlan 1
                             ->  Aggregate  (cost=10.83..10.84 rows=1 width=32) (actual time=0.006..0.006 rows=1 loops=235)
                                   ->  Bitmap Heap Scan on offer o2  (cost=4.30..10.83 rows=1 width=5) (actual time=0.002..0.003 rows=2 loops=235)
                                         Recheck Cond: (job_openning_id = o.job_openning_id)
                                         Filter: ((offer_status).acceptence_status = 'rejected'::acceptence_status_type)
                                         Rows Removed by Filter: 2
                                         Heap Blocks: exact=675
                                         ->  Bitmap Index Scan on idx_offer_job_openning_id  (cost=0.00..4.30 rows=3 width=0) (actual time=0.001..0.001 rows=4 loops=235)
                                               Index Cond: (job_openning_id = o.job_openning_id)
                     ->  Seq Scan on job_openning jo  (cost=0.00..21.00 rows=300 width=25) (actual time=0.000..0.010 rows=158 loops=153)
               ->  Seq Scan on interviewee i  (cost=0.00..11.00 rows=700 width=8) (actual time=0.001..0.016 rows=347 loops=153)
         ->  Seq Scan on requester r  (cost=0.00..20.00 rows=700 width=8) (actual time=0.001..0.016 rows=347 loops=153)
   ->  Index Scan using person_pkey on person p  (cost=0.28..0.76 rows=1 width=18) (actual time=0.002..0.002 rows=1 loops=153)
         Index Cond: (person_id = r.person_id)
 Planning Time: 7.907 ms
 Execution Time: 15.336 ms
(29 rows)
```

با گذاشتن نمایه روی ستون‌های job_openning_id و recommended_payment در جدول offer می‌توان سرعت پرسش را بالا برد. به این صورت که برای انجام subquery از نمایه روی job_openning_id استفاده می‌شود. پس از پیدا کردن بیشترین حقوق پیشنهادی کافی‌ست به صورت برعکس نمایه روی ستون recommended payment حرکت کنیم و با پیدا کردن سطر مورد نظر، یعنی سطری که پیشنهاد رد شده و مقدار پیشنهادی همان حاصل subquery است سایر اطلاعات به‌دست می‌آیند و حاصل با سایر جدول‌ها ادغام می‌شود.

پرسش ۱۰

query10.sql

```sql
-- sort interview types based on length
EXPLAIN ANALYZE
SELECT
    it.interview_type_id,
    it.interview_type_title,
    it.time_duration
FROM
    interview_type it
ORDER BY
    it.time_duration DESC;
```

index10.sql

```sql
CREATE INDEX idx_interview_type_time ON interview_type(time_duration);
-- DROP INDEX idx_interview_type_time;
```

این پرسش نوع مصاحبه‌ها را از طولانی‌ترین به کوتاه‌ترین مرتب می‌کند. تاثیر نمایه در مرتب‌سازی را به کمک این پرسش می‌بینیم.

بی نمایه

```
                                                        QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=352.05..363.30 rows=4500 width=26) (actual time=1.329..1.483 rows=4500 loops=1)
   Sort Key: time_duration DESC
   Sort Method: quicksort  Memory: 399kB
   ->  Seq Scan on interview_type it  (cost=0.00..79.00 rows=4500 width=26) (actual time=0.012..0.530 rows=4500 loops=1)
 Planning Time: 1.124 ms
 Execution Time: 1.772 ms
(6 rows)
```

در این حالت از quicksort استفاده شده است.

با نمایه

```
                                                                  QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------------------
 Index Scan Backward using idx_interview_type_time on interview_type it  (cost=0.28..283.72 rows=4500 width=26) (actual time=0.037..1.348 rows=4500 loops=1)
 Planning Time: 1.177 ms
 Execution Time: 1.539 ms
(3 rows)
```

در این حالت به ترتیب عکس نمایه جدول پیمایش می‌شود.