PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

Praktikum 3

Query Processing and Optimization

Jumat, 18 Oktober 2024

Oleh:
13522053 - Erdianti Wiga Putri Andini
13522116 - Naufal Adnan

IF3140 - Sistem Basis Data

2024

Petunjuk

1. Kerjakan setiap soal praktikum ini dengan baik
2. Untuk kebutuhan perbaikan kinerja dari query, Anda dipersilakan menggunakan kakas *query executor* untuk melakukan simulasi pengaksesan basis data secara bersamaan untuk sejumlah *30* akses. Gunakan dengan query awal dan query akhir
3. Nama database: **airport**
4. **Dilarang Menyontek**

AirportDB


passengerdetails
passenger_id
birthdate
sex
street
city
zip
country
emailaddress
telephoneno
passenger
passenger_id
passportno
firstname
lastname
airport_geo
airport_id
name
city
country
latitude
longitude
geolocation
booking
booking_id
flight_id
seat
passenger_id
price
flight
flight_id
flightno
from
to
departure
arrival
airline_id
airplane_id
airplane
airplane_id
capacity
type_id
airline_id
airplane_type
type_id
identifier
description
airport
airport_id
iata
icao
name
employee
employee_id
firstname
lastname
birthdate
sex
street
city
zip
country
emailaddress
telephoneno
salary
department
username
password
airline
airline_id
iata
airlinename
base_airport
weatherdata
log_date
time
station
temp
humidity
airpressure
wind
weather
winddirection
flightschedule
flightno
from
to
departure
arrival
airline_id
monday
tuesday
wednesday
thursday
friday
saturday
sunday

1. Perdebatan Dua Sejoli

Pak Cello dan Bu Didot sedang berdebat ketika Anda memasuki ruangan mereka. Mereka sedang memperdebatkan *query* siapakah yang lebih efisien untuk mengambil data yang mereka perlukan. Masing-masing dari mereka membuat *query* mereka sendiri-sendiri. Melihat perdebatan yang semakin memanas Pak MyHand akhirnya turun tangan dan membuat *query* baru dengan hasil yang sama dengan niatan agar perdebatan selesai. Namun, hal tersebut malah makin memperpanas perdebatan dan kini Pak MyHand juga ikut berdebat (Dasar Pak Myhand). Sebagai seorang Junior DBA bantulah Pak Cello, Bu Didot, dan Pak MyHand menyelesaikan perdebatan mereka dengan melakukan analisis terhadap 3 *query* berikut.

Notes: Gunakan `EXPLAIN ANALYZE` untuk pengerjaan soal ini

Query Pak Cello

```
SELECT
  a.airplane_id,
  COUNT(f.to) AS total_ke_perancis
FROM
  airplane a
INNER JOIN
  flight f ON f.airplane_id = a.airplane_id
INNER JOIN
  airport ap ON ap.airport_id = f.to
INNER JOIN
  airport_geo ag ON ap.airport_id = ag.airport_id
WHERE
  ag.country = 'FRANCE'
GROUP BY
  a.airplane_id
HAVING
  COUNT(a.airplane_id) > 5
ORDER BY total_ke_perancis DESC;
```

Query Bu Didot

```
WITH france_airports AS (
    SELECT ap.airport_id
    FROM airport ap
    INNER JOIN airport_geo ag ON ap.airport_id = ag.airport_id
    WHERE ag.country = 'FRANCE'
)
SELECT
  a.airplane_id,
  COUNT(f.to) AS total_ke_perancis
FROM
  airplane a
INNER JOIN
  flight f ON f.airplane_id = a.airplane_id
INNER JOIN
  france_airports fa ON f.to = fa.airport_id
```

```
GROUP BY
  a.airplane_id
HAVING
  COUNT(a.airplane_id) > 5
ORDER BY total_ke_perancis DESC;
```

Query Pak Myhand

```
SELECT
  a.airplane_id,
  COUNT(f.to) AS total_ke_perancis
FROM
  airplane a
INNER JOIN
  flight f ON f.airplane_id = a.airplane_id
WHERE
  f.to IN (
    SELECT ap.airport_id
    FROM airport ap
    INNER JOIN airport_geo ag ON ap.airport_id = ag.airport_id
    WHERE ag.country = 'FRANCE'
  )
GROUP BY
  a.airplane_id
HAVING
  COUNT(a.airplane_id) > 5
ORDER BY total_ke_perancis DESC;
```

SS Eksekusi Query Pak Cello

Tampilkan dalam 2 SS terpisah: Query Plan dan Execution Time

```
                                                                QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=1069.85..1072.05 rows=880 width=16) (actual time=7.265..7.267 rows=14 loops=1)
   Sort Key: (count(f."to")) DESC
   Sort Method: quicksort  Memory: 25kB
   ->  HashAggregate  (cost=993.83..1026.82 rows=880 width=16) (actual time=7.155..7.243 rows=14 loops=1)
         Group Key: a.airplane_id
         Filter: (count(a.airplane_id) > 5)
         Batches: 1  Memory Usage: 369kB
         Rows Removed by Filter: 1876
         ->  Hash Join  (cost=409.90..974.04 rows=2639 width=10) (actual time=3.991..6.676 rows=2810 loops=1)
               Hash Cond: (f.airplane_id = a.airplane_id)
               ->  Nested Loop  (cost=247.90..805.11 rows=2639 width=10) (actual time=0.806..3.172 rows=2810 loops=1)
                     ->  Hash Join  (cost=247.61..438.03 rows=435 width=6) (actual time=0.783..1.637 rows=435 loops=1)
                           Hash Cond: (ap.airport_id = ag.airport_id)
                           ->  Seq Scan on airport ap  (cost=0.00..164.54 rows=9854 width=4) (actual time=0.005..0.322 rows=9854 loops=1)
                           ->  Hash  (cost=242.18..242.18 rows=435 width=2) (actual time=0.757..0.757 rows=435 loops=1)
                                 Buckets: 1024  Batches: 1  Memory Usage: 23kB
                                 ->  Seq Scan on airport_geo ag  (cost=0.00..242.18 rows=435 width=2) (actual time=0.015..0.694 rows=435 loops=1)
                                       Filter: ((country)::text = 'FRANCE'::text)
                                       Rows Removed by Filter: 9419
                     ->  Index Scan using idx_96453_to on flight f  (cost=0.29..0.74 rows=10 width=10) (actual time=0.001..0.003 rows=6 loops=435)
                           Index Cond: ("to" = ap.airport_id)
               ->  Hash  (cost=94.22..94.22 rows=5422 width=8) (actual time=3.092..3.092 rows=5422 loops=1)
                     Buckets: 8192  Batches: 1  Memory Usage: 276kB
                     ->  Seq Scan on airplane a  (cost=0.00..94.22 rows=5422 width=8) (actual time=0.011..1.087 rows=5422 loops=1)
 Planning Time: 2.826 ms
 Execution Time: 7.615 ms
(26 rows)
```

```
  ag.country = 'FRANCE'
GROUP BY
  a.airplane_id
HAVING
  COUNT(a.airplane_id) > 5
ORDER BY total_ke_perancis DESC;
Time: 12.616 ms
airport=# SELECT
  a.airplane_id,
  COUNT(f.to) AS total_ke_perancis
FROM
  airplane a
INNER JOIN
  flight f ON f.airplane_id = a.airplane_id
INNER JOIN
  airport ap ON ap.airport_id = f.to
INNER JOIN
  airport_geo ag ON ap.airport_id = ag.airport_id
WHERE
  ag.country = 'FRANCE'
GROUP BY
  a.airplane_id
HAVING
  COUNT(a.airplane_id) > 5
ORDER BY total_ke_perancis DESC;
 airplane_id | total_ke_perancis
-------------+-------------------
        1318 |                16
        4988 |                10
        1220 |                10
        4592 |                 9
        4170 |                 8
        1221 |                 8
        4169 |                 8
         528 |                 7
        4591 |                 7
        4164 |                 6
        4047 |                 6
        4165 |                 6
        4023 |                 6
        1699 |                 6
(14 rows)

Time: 9.968 ms
airport=#
```

SS Eksekusi Query Bu Didot
Tampilkan dalam 2 SS terpisah: Query Plan dan Execution Time

```
                                                QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=1069.85..1072.05 rows=880 width=16) (actual time=5.747..5.750 rows=14 loops=1)
   Sort Key: (count(f."to")) DESC
   Sort Method: quicksort  Memory: 25kB
   ->  HashAggregate  (cost=993.83..1026.82 rows=880 width=16) (actual time=5.629..5.737 rows=14 loops=1)
         Group Key: a.airplane_id
         Filter: (count(a.airplane_id) > 5)
         Batches: 1  Memory Usage: 369kB
         Rows Removed by Filter: 1876
         ->  Hash Join  (cost=409.90..974.04 rows=2639 width=10) (actual time=1.607..5.000 rows=2810 loops=1)
               Hash Cond: (f.airplane_id = a.airplane_id)
               ->  Nested Loop  (cost=247.90..805.11 rows=2639 width=10) (actual time=0.705..3.674 rows=2810 loops=1)
                     ->  Hash Join  (cost=247.61..438.03 rows=435 width=6) (actual time=0.682..1.750 rows=435 loops=1)
                           Hash Cond: (ap.airport_id = ag.airport_id)
                           ->  Seq Scan on airport ap  (cost=0.00..164.54 rows=9854 width=4) (actual time=0.002..0.419 rows=9854 loops=1)
                           ->  Hash  (cost=242.18..242.18 rows=435 width=2) (actual time=0.670..0.671 rows=435 loops=1)
                                 Buckets: 1024  Batches: 1  Memory Usage: 23kB
                                 ->  Seq Scan on airport_geo ag  (cost=0.00..242.18 rows=435 width=2) (actual time=0.005..0.620 rows=435 loops=1)
                                       Filter: ((country)::text = 'FRANCE'::text)
                                       Rows Removed by Filter: 9419
                     ->  Index Scan using idx_96453_to on flight f  (cost=0.29..0.74 rows=10 width=10) (actual time=0.001..0.004 rows=6 loops=435)
                           Index Cond: ("to" = ap.airport_id)
               ->  Hash  (cost=94.22..94.22 rows=5422 width=8) (actual time=0.876..0.877 rows=5422 loops=1)
                     Buckets: 8192  Batches: 1  Memory Usage: 276kB
                     ->  Seq Scan on airplane a  (cost=0.00..94.22 rows=5422 width=8) (actual time=0.006..0.318 rows=5422 loops=1)
 Planning Time: 0.877 ms
 Execution Time: 5.941 ms
(26 rows)
```

```
airport=# WITH france_airports AS (
    SELECT ap.airport_id
    FROM airport ap
    INNER JOIN airport_geo ag ON ap.airport_id = ag.airport_id
    WHERE ag.country = 'FRANCE'
)
SELECT
  a.airplane_id,
  COUNT(f.to) AS total_ke_perancis
FROM
  airplane a
INNER JOIN
  flight f ON f.airplane_id = a.airplane_id
INNER JOIN
  france_airports fa ON f.to = fa.airport_id
GROUP BY
  a.airplane_id
HAVING
  COUNT(a.airplane_id) > 5
ORDER BY total_ke_perancis DESC;
 airplane_id | total_ke_perancis
-------------+-------------------
        1318 |                16
        4988 |                10
        1220 |                10
        4592 |                 9
        4170 |                 8
        1221 |                 8
        4169 |                 8
         528 |                 7
        4591 |                 7
        4164 |                 6
        4047 |                 6
        4165 |                 6
        4023 |                 6
        1699 |                 6
(14 rows)

Time: 9.392 ms
airport=#
```

SS Eksekusi Query Pak Myhand

Tampilkan dalam 2 SS terpisah: Query Plan dan Execution Time

```
                                                                QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=1075.29..1077.49 rows=880 width=16) (actual time=8.223..8.226 rows=14 loops=1)
   Sort Key: (count(f."to")) DESC
   Sort Method: quicksort  Memory: 25kB
   ->  HashAggregate  (cost=999.27..1032.25 rows=880 width=16) (actual time=8.118..8.204 rows=14 loops=1)
         Group Key: a.airplane_id
         Filter: (count(a.airplane_id) > 5)
         Batches: 1  Memory Usage: 369kB
         Rows Removed by Filter: 1876
         ->  Hash Join  (cost=601.40..979.47 rows=2639 width=10) (actual time=5.754..7.630 rows=2810 loops=1)
               Hash Cond: (f.airplane_id = a.airplane_id)
               ->  Nested Loop  (cost=439.41..810.55 rows=2639 width=10) (actual time=2.552..4.112 rows=2810 loops=1)
                     ->  HashAggregate  (cost=439.12..443.47 rows=435 width=6) (actual time=2.533..2.566 rows=435 loops=1)
                           Group Key: ap.airport_id
                           Batches: 1  Memory Usage: 61kB
                           ->  Hash Join  (cost=247.61..438.03 rows=435 width=6) (actual time=1.626..2.470 rows=435 loops=1)
                                 Hash Cond: (ap.airport_id = ag.airport_id)
                                 ->  Seq Scan on airport ap  (cost=0.00..164.54 rows=9854 width=4) (actual time=0.005..0.327 rows=9854 loops=1)
                                 ->  Hash  (cost=242.18..242.18 rows=435 width=2) (actual time=1.593..1.593 rows=435 loops=1)
                                       Buckets: 1024  Batches: 1  Memory Usage: 23kB
                                       ->  Seq Scan on airport_geo ag  (cost=0.00..242.18 rows=435 width=2) (actual time=0.014..1.483 rows=435 loops=1)
                                             Filter: ((country)::text = 'FRANCE'::text)
                                             Rows Removed by Filter: 9419
                     ->  Index Scan using idx_96453_to on flight f  (cost=0.29..0.74 rows=10 width=10) (actual time=0.001..0.003 rows=6 loops=435)
                           Index Cond: ("to" = ap.airport_id)
               ->  Hash  (cost=94.22..94.22 rows=5422 width=8) (actual time=3.111..3.111 rows=5422 loops=1)
                     Buckets: 8192  Batches: 1  Memory Usage: 276kB
                     ->  Seq Scan on airplane a  (cost=0.00..94.22 rows=5422 width=8) (actual time=0.012..1.095 rows=5422 loops=1)
 Planning Time: 2.214 ms
 Execution Time: 8.603 ms
(29 rows)

Time: 11.811 ms
airport=#
```

```
airport=# SELECT
  a.airplane_id,
  COUNT(f.to) AS total_ke_perancis
FROM
  airplane a
INNER JOIN
  flight f ON f.airplane_id = a.airplane_id
WHERE
  f.to IN (
    SELECT ap.airport_id
    FROM airport ap
    INNER JOIN airport_geo ag ON ap.airport_id = ag.airport_id
    WHERE ag.country = 'FRANCE'
  )
GROUP BY
  a.airplane_id
HAVING
  COUNT(a.airplane_id) > 5
ORDER BY total_ke_perancis DESC;
 airplane_id | total_ke_perancis
-------------+-------------------
        1318 |                16
        4988 |                10
        1220 |                10
        4592 |                 9
        4170 |                 8
        1221 |                 8
        4169 |                 8
         528 |                 7
        4591 |                 7
        4164 |                 6
        4047 |                 6
        4165 |                 6
        4023 |                 6
        1699 |                 6
(14 rows)

Time: 11.547 ms
airport=#
```

	Query Pak Cello	Query Bu Didot	Query Pak Myhand
Kelebihan	Merupakan query dengan format yang sering digunakan sehingga familiar dan mudah untuk digunakan	Menggunakan with untuk menyimpan sementara informasi mengenai bandara apa saja yang berada di prancis. Lalu informasi tersebut digunakan dalam selection query berikutnya	

		sehingga tidak perlu mengeksekusi semua baris airport_geo, hanya yang france saja alias mengurangi jumlah join pada query eksekusi.	
Kekurangan	Melakukan INNER JOIN dari banyak tabel, yang mana kita tahu bahwa operasi JOIN ini merupakan operasi yang mahal. INNER JOIN yang dilakukan akan menghasilkan satu tabel yang besar, tidak ada operasi selection yang digunakan untuk mengambil atribut tertentu yang digunakan sehingga mengurangi ukuran relasi.	Sebenernya sudah bagus, tapi bisa ditingkatkan lagi dengan melakukan selection ke atribut yang digunakan saja untuk INNER JOIN. Misalnya, pada tabel airplane bisa diselect airplane_id dulu, lalu tabel flight bisa diselect dulu atribut flight_id dan to. Hal ini bisa mengurangi ukuran tabel hasil join.	Menggunakan correlated subquery dimana adanya query select di saat memilih destinasi sehingga waktu eksekusi lebih lama karena query selection ini akan selalu diproses tiap mengeksekusi baris tabel airplane.

Konklusi Akhir dan Justifikasinya
Dari hasil perbandingan dari ketiga query tersebut, query milik Bu Didot adalah paling cepat (9.392ms), lalu Pak Cello (9.968ms), dan terakhir Pak Myhand (11.547ms), terbukti dari \timing eksekusi. Query Pak Cello sebenarnya sudah cukup cepat, namun karena tidak adanya penyimpanan sementara mengenai airport yang berlokasi di france, membuat eksekusi query menjadi sedikit lebih lama dibandingkan query Bu Didot. Query Bu Didot paling cepat di antara yang lain karena adanya penyimpanan informasi sementara mengenai airport-airport yang berlokasi di france sehingga bisa mengurangi jumlah pengecekan baris di tabel airport_geo dan airport. Query Pak Mayhend paling lambat karena menggunakan correlated subquery, yang membuat tiap row dari tabel hasil join flight dan airplane harus mengeksekusi subquerynya dan membuat operasi menjadi mahal. Berdasarkan query yang tertera, dapat disimpulkan cara optimasi query dapat dilakukan dengan menyimpan sementara informasi-informasi yang akan dibutuhkan agar memperkecil jumlah JOIN. Correlated subquery juga sebaiknya dihindari karena akan membuat cost jadi mahal.

2. Kawan Lama

Setelah tidak bertemu selama 10 tahun, Ardneham akhirnya mengunjungi teman lamanya, Marie Antoinette, dalam suatu perjalanan bisnis untuk menangani sebuah bandara yang berlokasi di Perancis. Mereka berbincang-bincang terkait penerbangan yang keluar dan masuk Perancis.

a. Marie Antoinette ingin mendapatkan seluruh kode bandara di negara Perancis (country = 'FRANCE') yang menjadi *base airport* dari suatu maskapai. Setiap maskapai seharusnya memiliki *base airport* yang berbeda. Marie Antoinette menulis query di bawah ini untuk mendapatkan data tersebut,

```
SELECT DISTINCT base_airport
FROM airline ar
   JOIN airport_geo ag
      ON ar.base_airport = ag.airport_id
   JOIN airport a
      ON ar.base_airport = a.airport_id
WHERE country LIKE '%FRANCE%';
```

Lakukan analisis pemrosesan pada query di atas menggunakan `EXPLAIN` pada PostgreSQL,

Query Analisis Pemrosesan	`EXPLAIN SELECT DISTINCT base_airport FROM airline ar JOIN airport_geo ag ON ar.base_airport = ag.airport_id JOIN airport a ON ar.base_airport = a.airport_id WHERE country LIKE '%FRANCE%';`

SS Hasil Query Masukan

```
airport=# EXPLAIN SELECT DISTINCT base_airport
FROM airline ar
   JOIN airport_geo ag
      ON ar.base_airport = ag.airport_id
   JOIN airport a
      ON ar.base_airport = a.airport_id
WHERE country LIKE '%FRANCE%';
                                                  QUERY PLAN
---------------------------------------------------------------------------------------------------------------
 Unique  (cost=192.97..192.99 rows=5 width=2)
   ->  Sort  (cost=192.97..192.98 rows=5 width=2)
         Sort Key: ar.base_airport
         ->  Nested Loop  (cost=0.57..192.91 rows=5 width=2)
               Join Filter: (ar.base_airport = ag.airport_id)
               ->  Nested Loop  (cost=0.29..151.38 rows=110 width=6)
                     ->  Seq Scan on airline ar  (cost=0.00..2.10 rows=110 width=2)
                     ->  Index Only Scan using idx_96423_primary on airport a  (cost=0.29..1.36 rows=1 width=4)
                           Index Cond: (airport_id = ar.base_airport)
               ->  Index Scan using idx_96428_primary on airport_geo ag  (cost=0.29..0.37 rows=1 width=2)
                     Index Cond: (airport_id = a.airport_id)
                     Filter: ((country)::text ~~ '%FRANCE%'::text)
(12 rows)

Time: 3.913 ms
airport=#
```

Menurut Anda, apakah *query* dapat disederhanakan? Jika ya, tuliskan *query* pengganti dengan *cost* lebih rendah yang memiliki semantik yang sama dengan *query* di atas. Jika tidak, jelaskan mengapa *query* tersebut sudah merupakan *query* paling efektif untuk tujuan tersebut.

Perbaikan Query (Jika Memilih Ya)	`SELECT DISTINCT base_airport`
	`FROM airline ar`
	`JOIN (`
	`SELECT a.airport_id`
	`FROM airport_geo ag JOIN airport a ON ag.airport_id = a.airport_id`
	`WHERE country = 'FRANCE'`
	`) as a2`
	`ON ar.base_airport = a2.airport_id;`

SS Query Perbaikan (Jika Memilih Ya)

```
airport=# SELECT DISTINCT base_airport
FROM airline ar
   JOIN (
         SELECT a.airport_id
         FROM airport_geo ag JOIN airport a ON ag.airport_id = a.airport_id
         WHERE country = 'FRANCE'
    ) as a2
   ON ar.base_airport = a2.airport_id;
 base_airport
--------------
          146
(1 row)

Time: 4.191 ms
airport=#
```

Query Analisis Pemrosesan Untuk Query Perbaikan (Jika Memilih Ya)	Kami memperbaiki query ini dengan mengganti pemilihan country tidak dengan mengecek substring, melainkan langsung mencari country france. Hal ini karena operasi like substring merupakan operasi yang mahal. Sebagai gantinya, gunakan where untuk mempercepat pemrosesan. Selain itu, pemakaian like juga dihindari untuk kedepannya jika suatu atribut digunakan sebagai index. Lalu kami juga menyeleksi airport_id yang berlokasi di france terlebih dahulu sebelum dilakukan JOIN dengan airplane untuk memperkecil ukuran tabel seleksi sehingga tidak harus mengecek seluruh baris dalam tabel airplane (hanya baris yang mengandung base_airport yang sesuai).

SS Query Analisis Pemrosesan untuk Query Perbaikan (Jika Memilih Ya)

```
airport=# EXPLAIN SELECT DISTINCT base_airport
FROM airline ar
   JOIN (
         SELECT a.airport_id
         FROM airport_geo ag JOIN airport a ON ag.airport_id = a.airport_id
         WHERE country = 'FRANCE'
    ) as a2
   ON ar.base_airport = a2.airport_id;
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Unique  (cost=191.59..191.62 rows=5 width=2)
   ->  Sort  (cost=191.59..191.60 rows=5 width=2)
         Sort Key: ar.base_airport
         ->  Nested Loop  (cost=0.57..191.53 rows=5 width=2)
               ->  Nested Loop  (cost=0.29..151.38 rows=110 width=6)
                     ->  Seq Scan on airline ar  (cost=0.00..2.10 rows=110 width=2)
                     ->  Index Only Scan using idx_96423_primary on airport a  (cost=0.29..1.36 rows=1 width=4)
                           Index Cond: (airport_id = ar.base_airport)
               ->  Index Scan using idx_96428_primary on airport_geo ag  (cost=0.29..0.37 rows=1 width=2)
                     Index Cond: (airport_id = a.airport_id)
                     Filter: ((country)::text = 'FRANCE'::text)
(11 rows)

Time: 1.031 ms
airport#
```

Justifikasi Berdasarkan Hasil (Jika Memilih Ya) atau Alasan Query Sudah Efektif (Jika Memilih Tidak)
Query perbaikan yang kami susun sudah lebih efektif dibandingkan query asli, terbukti dari cost query perbaikan lebih kecil daripada cost query asli walaupun perbedaannya tidak signifikan. Alasan query ini sudah lebih efektif adalah karena tidak adanya pencarian berdasarkan substring sehingga dapat mempercepat waktu eksekusi. Selain itu kami juga sudah menyeleksi airport_id mana saja yang diperlukan sebelum proses JOIN agar memperkecil ukuran tabel yang harus dijoinkan dengan airplane.

Terdapat beberapa algoritma JOIN yang dapat digunakan. Jelaskan mekanisme salah satu JOIN yang digunakan pada *Query Plan* di atas.

Pada query di atas menggunakan nested loop. Mekanisme nested loop dilakukan dengan mengambil sebuah row dari outer relation untuk ditelusuri di blok outer relation apakah ada row yang memenuhi kondisi join tersebut. Secara pseudocode begini:

```
For tr in br
   For ts in bs
      If tr and bs match in condition then
         Masukkan ke tuple hasil
```

b. Ardneham menggunakan *query* yang sudah disederhanakan pada jawaban A, untuk mendapatkan seluruh data pesawat yang terdiri atas *airplane id*, tipe armada (*type id*), dan nama maskapai pemilik armada tersebut untuk seluruh maskapai yang memiliki *base airport* yang berlokasi di Perancis, ditampilkan dengan *airplane id* terurut membesar. Dia menulis *query* berikut ini untuk mendapatkan data tersebut.

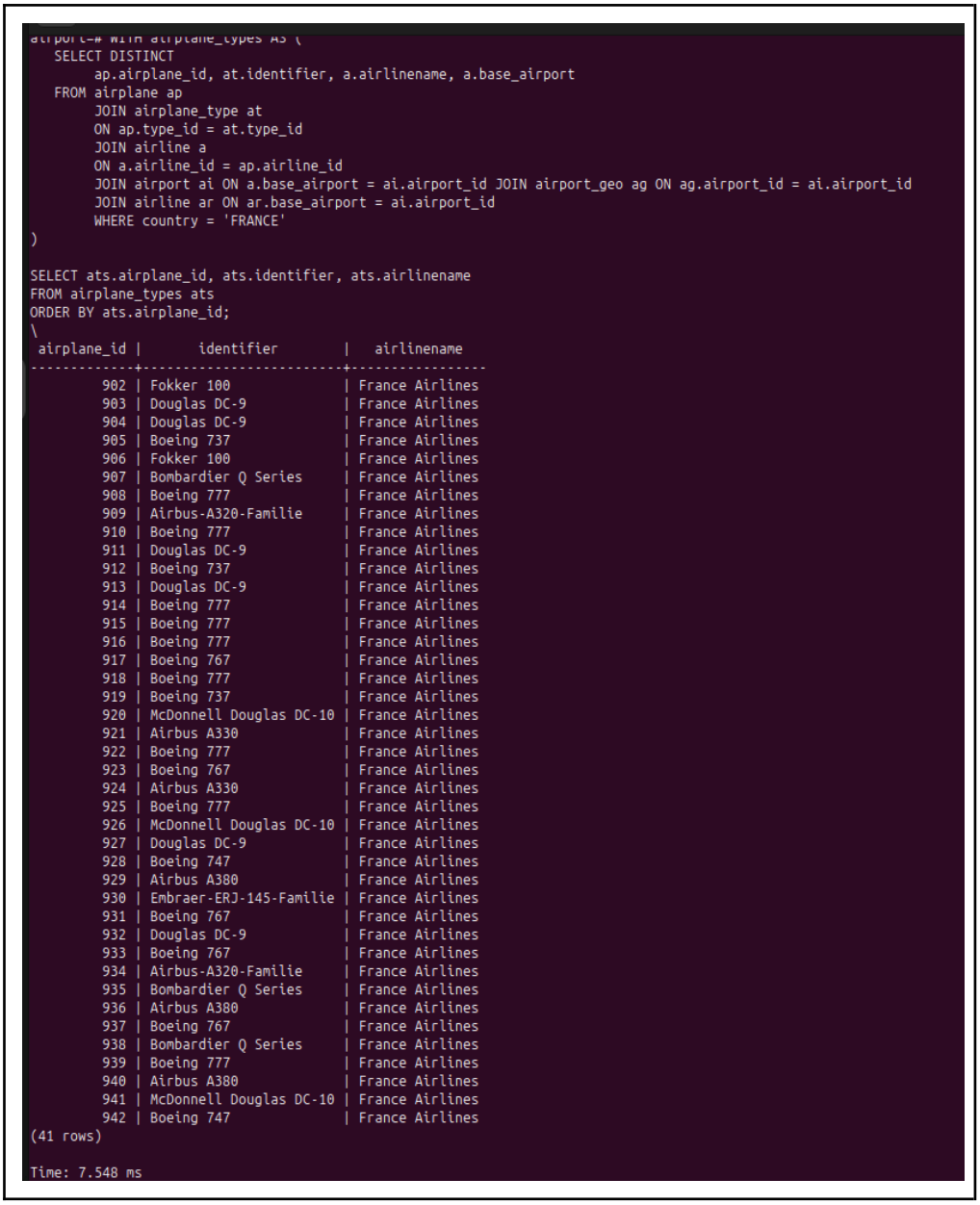
```
WITH airplane_types AS (
   SELECT DISTINCT
      ap.airplane_id, at.identifier, a.airlinename, a.base_airport
   FROM airplane ap
      JOIN airplane_type at
         ON ap.type_id = at.type_id
```

```
       JOIN airline a
          ON a.airline_id = ap.airline_id
 )
 SELECT ats.airplane_id, ats.identifier, ats.airlinename
 FROM airplane_types ats
 WHERE ats.base_airport IN (
       SELECT DISTINCT base_airport
FROM airline ar
   JOIN (
       SELECT a.airport_id
       FROM airport_geo ag JOIN airport a ON ag.airport_id =
a.airport_id
       WHERE country = 'FRANCE'
       ) as a2
   ON ar.base_airport = a2.airport_id
 )
 ORDER BY ats.airplane_id;
```

Ardneham sadar bahwa *query* yang ditulisnya tidak efektif karena menggunakan 2 buah *subquery* dalam pemrosesannya, sehingga Ardneham ingin melakukan penyederhanaan brutal supaya *query* tersebut TIDAK menggunakan *subquery* dalam pemrosesannya. Bantulah Ardneham untuk membuat *query* yang lebih sederhana.

Query Perbaikan	`WITH airplane_types AS (`
	`  SELECT DISTINCT`
	`  ap.airplane_id, at.identifier,`
	`a.airlinename, a.base_airport`
	`  FROM airplane ap`
	`  JOIN airplane_type at`
	`  ON ap.type_id = at.type_id`
	`  JOIN airline a`
	`  ON a.airline_id = ap.airline_id`
	`  JOIN airport ai ON a.base_airport =`
	`ai.airport_id JOIN airport_geo ag ON`
	`ag.airport_id = ai.airport_id`
	`  JOIN airline ar ON ar.base_airport =`
	`ai.airport_id`
	`  WHERE country = 'FRANCE'`
	`)`
	
	`SELECT ats.airplane_id, ats.identifier,`
	`ats.airlinename`
	`FROM airplane_types ats`
	`ORDER BY ats.airplane_id;`

SS Hasil Query Perbaikan	



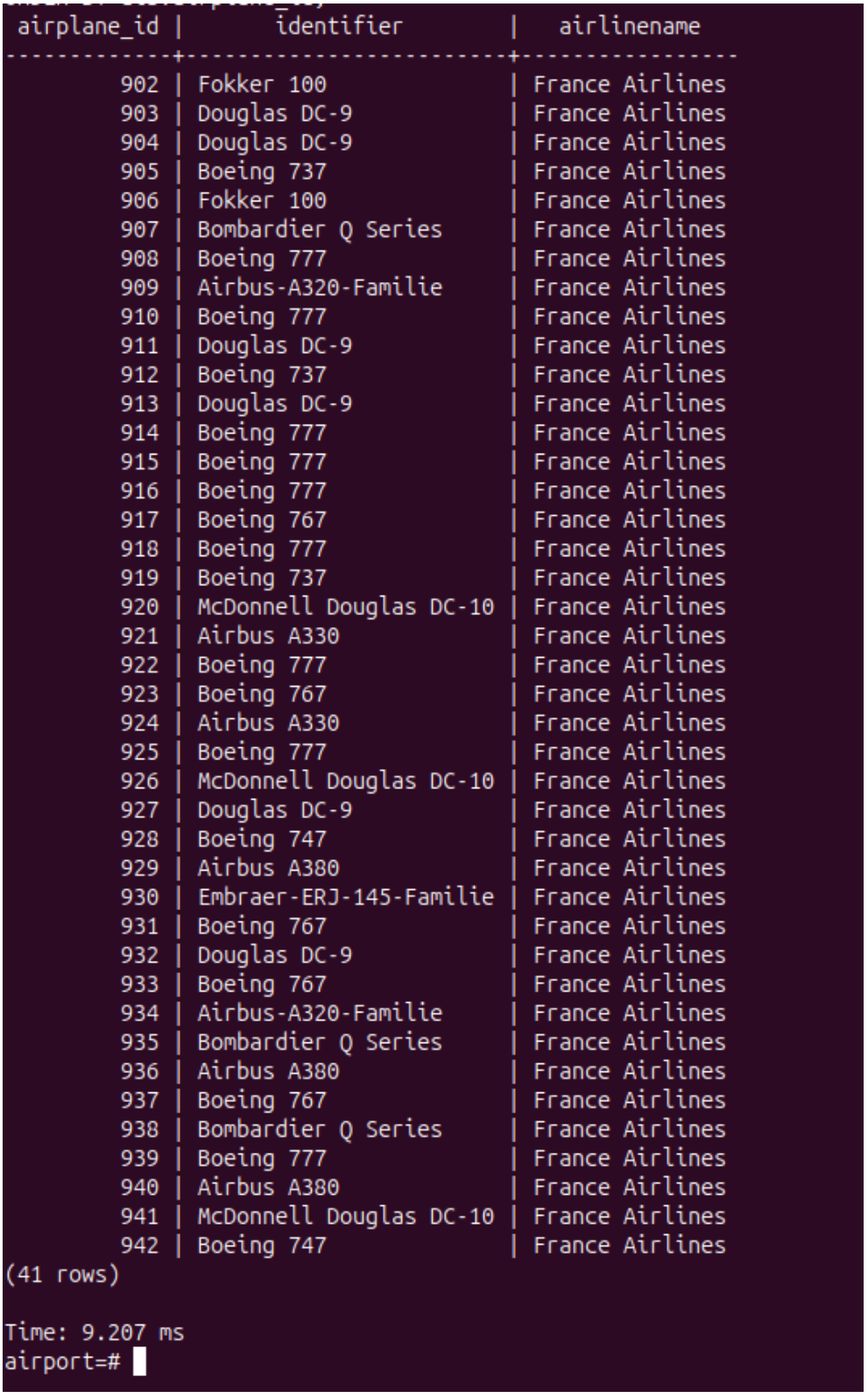
```
airport-# WITH airplane_types AS (
   SELECT DISTINCT
        ap.airplane_id, at.identifier, a.airlinename, a.base_airport
   FROM airplane ap
        JOIN airplane_type at
        ON ap.type_id = at.type_id
        JOIN airline a
        ON a.airline_id = ap.airline_id
        JOIN airport ai ON a.base_airport = ai.airport_id JOIN airport_geo ag ON ag.airport_id = ai.airport_id
        JOIN airline ar ON ar.base_airport = ai.airport_id
        WHERE country = 'FRANCE'
)

SELECT ats.airplane_id, ats.identifier, ats.airlinename
FROM airplane_types ats
ORDER BY ats.airplane_id;
\
 airplane_id |       identifier        |   airlinename
-------------+-------------------------+-----------------
         902 | Fokker 100              | France Airlines
         903 | Douglas DC-9            | France Airlines
         904 | Douglas DC-9            | France Airlines
         905 | Boeing 737              | France Airlines
         906 | Fokker 100              | France Airlines
         907 | Bombardier Q Series     | France Airlines
         908 | Boeing 777              | France Airlines
         909 | Airbus-A320-Familie     | France Airlines
         910 | Boeing 777              | France Airlines
         911 | Douglas DC-9            | France Airlines
         912 | Boeing 737              | France Airlines
         913 | Douglas DC-9            | France Airlines
         914 | Boeing 777              | France Airlines
         915 | Boeing 777              | France Airlines
         916 | Boeing 777              | France Airlines
         917 | Boeing 767              | France Airlines
         918 | Boeing 777              | France Airlines
         919 | Boeing 737              | France Airlines
         920 | McDonnell Douglas DC-10 | France Airlines
         921 | Airbus A330             | France Airlines
         922 | Boeing 777              | France Airlines
         923 | Boeing 767              | France Airlines
         924 | Airbus A330             | France Airlines
         925 | Boeing 777              | France Airlines
         926 | McDonnell Douglas DC-10 | France Airlines
         927 | Douglas DC-9            | France Airlines
         928 | Boeing 747              | France Airlines
         929 | Airbus A380             | France Airlines
         930 | Embraer-ERJ-145-Familie | France Airlines
         931 | Boeing 767              | France Airlines
         932 | Douglas DC-9            | France Airlines
         933 | Boeing 767              | France Airlines
         934 | Airbus-A320-Familie     | France Airlines
         935 | Bombardier Q Series     | France Airlines
         936 | Airbus A380             | France Airlines
         937 | Boeing 767              | France Airlines
         938 | Bombardier Q Series     | France Airlines
         939 | Boeing 777              | France Airlines
         940 | Airbus A380             | France Airlines
         941 | McDonnell Douglas DC-10 | France Airlines
         942 | Boeing 747              | France Airlines
(41 rows)

Time: 7.548 ms
```

Buktikan bahwa query yang Anda buat memberikan pemrosesan yang lebih sederhana dan waktu pemrosesan yang lebih cepat melalui keyword `EXPLAIN` dan `\timing`!

SS Eksekusi Query Ardneham (Aktifkan `\timing`)



airplane_id	identifier	airlinename
902	Fokker 100	France Airlines
903	Douglas DC-9	France Airlines
904	Douglas DC-9	France Airlines
905	Boeing 737	France Airlines
906	Fokker 100	France Airlines
907	Bombardier Q Series	France Airlines
908	Boeing 777	France Airlines
909	Airbus-A320-Familie	France Airlines
910	Boeing 777	France Airlines
911	Douglas DC-9	France Airlines
912	Boeing 737	France Airlines
913	Douglas DC-9	France Airlines
914	Boeing 777	France Airlines
915	Boeing 777	France Airlines
916	Boeing 777	France Airlines
917	Boeing 767	France Airlines
918	Boeing 777	France Airlines
919	Boeing 737	France Airlines
920	McDonnell Douglas DC-10	France Airlines
921	Airbus A330	France Airlines
922	Boeing 777	France Airlines
923	Boeing 767	France Airlines
924	Airbus A330	France Airlines
925	Boeing 777	France Airlines
926	McDonnell Douglas DC-10	France Airlines
927	Douglas DC-9	France Airlines
928	Boeing 747	France Airlines
929	Airbus A380	France Airlines
930	Embraer-ERJ-145-Familie	France Airlines
931	Boeing 767	France Airlines
932	Douglas DC-9	France Airlines
933	Boeing 767	France Airlines
934	Airbus-A320-Familie	France Airlines
935	Bombardier Q Series	France Airlines
936	Airbus A380	France Airlines
937	Boeing 767	France Airlines
938	Bombardier Q Series	France Airlines
939	Boeing 777	France Airlines
940	Airbus A380	France Airlines
941	McDonnell Douglas DC-10	France Airlines
942	Boeing 747	France Airlines

(41 rows)

Time: 9.207 ms
airport=#

SS Pemrosesan Query Ardneham (dengan `EXPLAIN`)

```
                                                                      QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=540.32..540.66 rows=136 width=39)
   Sort Key: ap.airplane_id
   ->  Hash Join  (cost=412.53..535.50 rows=136 width=39)
         Hash Cond: (a.base_airport = ar.base_airport)
         ->  HashAggregate  (cost=220.80..275.02 rows=5422 width=41)
               Group Key: ap.airplane_id, at.identifier, a.airlinename, a.base_airport
               ->  Hash Join  (cost=43.17..166.58 rows=5422 width=41)
                     Hash Cond: (ap.airline_id = a.airline_id)
                     ->  Hash Join  (cost=39.70..148.33 rows=5422 width=24)
                           Hash Cond: (ap.type_id = at.type_id)
                           ->  Seq Scan on airplane ap  (cost=0.00..94.22 rows=5422 width=18)
                           ->  Hash  (cost=35.42..35.42 rows=342 width=22)
                                 ->  Seq Scan on airplane_type at  (cost=0.00..35.42 rows=342 width=22)
                     ->  Hash  (cost=2.10..2.10 rows=110 width=23)
                           ->  Seq Scan on airline a  (cost=0.00..2.10 rows=110 width=23)
         ->  Hash  (cost=191.67..191.67 rows=5 width=2)
               ->  Unique  (cost=191.59..191.62 rows=5 width=2)
                     ->  Sort  (cost=191.59..191.60 rows=5 width=2)
                           Sort Key: ar.base_airport
                           ->  Nested Loop  (cost=0.57..191.53 rows=5 width=2)
                                 ->  Nested Loop  (cost=0.29..151.38 rows=110 width=6)
                                       ->  Seq Scan on airline ar  (cost=0.00..2.10 rows=110 width=2)
                                       ->  Index Only Scan using idx_96423_primary on airport a_1  (cost=0.29..1.36 rows=1 width=4)
                                             Index Cond: (airport_id = ar.base_airport)
                                 ->  Index Scan using idx_96428_primary on airport_geo ag  (cost=0.29..0.37 rows=1 width=2)
                                       Index Cond: (airport_id = a_1.airport_id)
                                       Filter: ((country)::text = 'FRANCE'::text)
(27 rows)

Time: 4.280 ms
airport=#
```

SS Pemrosesan Query Perbaikan Anda (dengan `EXPLAIN`)

```
airport=# EXPLAIN WITH airplane_types AS (
   SELECT DISTINCT
        ap.airplane_id, at.identifier, a.airlinename, a.base_airport
   FROM airplane ap
        JOIN airplane_type at
        ON ap.type_id = at.type_id
        JOIN airline a
        ON a.airline_id = ap.airline_id
        JOIN airport ai ON a.base_airport = ai.airport_id JOIN airport_geo ag ON ag.airport_id = ai.airport_id
        JOIN airline ar ON ar.base_airport = ai.airport_id
        WHERE country = 'FRANCE'
)

SELECT ats.airplane_id, ats.identifier, ats.airlinename
FROM airplane_types ats
ORDER BY ats.airplane_id;
                                                                      QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------
 Subquery Scan on ats  (cost=159.58..159.65 rows=3 width=39)
   ->  Unique  (cost=159.58..159.62 rows=3 width=41)
         ->  Sort  (cost=159.58..159.59 rows=3 width=41)
               Sort Key: ap.airplane_id, at.identifier, a.airlinename, a.base_airport
               ->  Nested Loop  (cost=4.47..159.56 rows=3 width=41)
                     ->  Nested Loop  (cost=4.33..158.99 rows=3 width=35)
                           ->  Nested Loop  (cost=4.04..155.65 rows=1 width=23)
                                 Join Filter: (a.base_airport = ag.airport_id)
                                 ->  Hash Join  (cost=3.76..155.27 rows=1 width=29)
                                       Hash Cond: (ai.airport_id = a.base_airport)
                                       ->  Nested Loop  (cost=0.29..151.38 rows=110 width=6)
                                             ->  Seq Scan on airline ar  (cost=0.00..2.10 rows=110 width=2)
                                             ->  Index Only Scan using idx_96423_primary on airport ai  (cost=0.29..1.36 rows=1 width=4)
                                                   Index Cond: (airport_id = ar.base_airport)
                                       ->  Hash  (cost=2.10..2.10 rows=110 width=23)
                                             ->  Seq Scan on airline a  (cost=0.00..2.10 rows=110 width=23)
                                 ->  Index Scan using idx_96428_primary on airport_geo ag  (cost=0.29..0.37 rows=1 width=2)
                                       Index Cond: (airport_id = ai.airport_id)
                                       Filter: ((country)::text = 'FRANCE'::text)
                           ->  Index Scan using idx_96410_airline_id on airplane ap  (cost=0.28..2.85 rows=49 width=18)
                                 Index Cond: (airline_id = a.airline_id)
                     ->  Index Scan using idx_96415_primary on airplane_type at  (cost=0.15..0.19 rows=1 width=22)
                           Index Cond: (type_id = ap.type_id)
(23 rows)

Time: 6.000 ms
airport=#
```

Penjelasan dan Justifikasi

Pada hasil perbaikan subquery diganti dengan join biasa saja. Join biasa memiliki cost yang lebih rendah dibanding subquery yang merupakan operasi mahal. Hal ini karena pada subquery tiap bari dari outer join akan mengeksekusi subquerynya, yang bisa membuat operasi berulang-ulang yang excessive. Sementara pada perbaikan setiap tabel dijoin terlebih dulu di query with untuk kemudian diperoleh hasil yang akan deselect. Hal ini memakan cost lebih sedikit terbukti dari yang sebelumnya 540.32..540.66 menjadi 159.58..159.65 setelah perbaikan.

3. Lomba Query Terbaik

Pak Julala dan Pak Jay sedang mengikuti lomba skala galaksi. Lomba tersebut adalah lomba pembuatan *query* terbaik yang diadakan oleh Badan Intelijen Basis Data Bima Sakti (BIBDBS). *Query* terbaik dinilai dari segi aspek waktu dan biaya (*time and cost*).
Notes: Gunakan `EXPLAIN ANALYZE` dalam pengerjaan soal ini

a. Pak Julala mengusulkan dua *query* yang memiliki fungsi dan hasil serupa, sebagai berikut:

Query 1

```
WITH SP2735_flight AS (
    SELECT DISTINCT flightno
    FROM flight
    WHERE flightno LIKE 'SP2735%'
), adult_passenger AS (
    SELECT passenger_id
    FROM passengerdetails
    WHERE EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM
birthdate) >= 18
)
SELECT p.passportno, CONCAT(p.firstname, ' ', p.lastname) AS
fullname, pd.birthdate, f.flightno
FROM passenger p
JOIN passengerdetails pd ON p.passenger_id = pd.passenger_id
JOIN booking b ON p.passenger_id = b.passenger_id
JOIN flight f ON b.flight_id = f.flight_id
JOIN SP2735_flight af ON f.flightno = af.flightno
JOIN adult_passenger apg ON p.passenger_id = apg.passenger_id
ORDER BY f.flightno ASC, p.passportno ASC, fullname ASC;
```

Query 2

```
WITH adult_passenger_data AS (
    SELECT p.passenger_id, p.passportno, CONCAT(p.firstname, '
', p.lastname) AS fullname, pd.birthdate
    FROM passenger p
    NATURAL JOIN passengerdetails pd
    WHERE EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM
birthdate) >= 18
    ORDER BY p.passportno ASC, fullname ASC
)
SELECT apd.passportno, apd.fullname, apd.birthdate, f.flightno
FROM adult_passenger_data apd, booking b, flight f, airplane
ap
WHERE b.passenger_id = apd.passenger_id
AND b.flight_id = f.flight_id
AND f.airplane_id = ap.airplane_id
AND f.flightno IN (
    SELECT DISTINCT flightno
    FROM flight
    WHERE flightno LIKE 'SP2735%'
    ORDER BY flightno ASC
);
```

Pak Julala merasa dari segi kebutuhan, kedua *query* yang dibuatnya sudah baik. Namun, ia merasa bahwa dua *query* tersebut masih bisa dioptimasi lebih lanjut. Maka dari itu, ia memintamu untuk menganalisis *query* tersebut terutama dari aspek waktu dan biaya (*time and cost*). Bantulah Pak Julala melakukan analisis kelebihan dan kekurangan antar *query* yang telah ia buat!

Notes: Tidak semua kotak pada bagian jawaban sub-soal ini perlu diisi. Jika dirasa tidak ada jawaban untuk kotak tersebut, praktikan dapat memilih untuk membiarkannya kosong.

SS Eksekusi *Query* 1
Tampilkan dalam 2 SS terpisah:
- Query Plan
- Bagian *Time*

```
airport=# EXPLAIN
WITH SP2735_flight AS (
    SELECT DISTINCT flightno
    FROM flight
    WHERE flightno LIKE 'SP2735%'
), adult_passenger AS (
    SELECT passenger_id
    FROM passengerdetails
    WHERE EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM birthdate) >= 18
)
SELECT p.passportno, CONCAT(p.firstname, ' ', p.lastname) AS fullname, pd.birthdate, f.flightno
FROM passenger p
JOIN passengerdetails pd ON p.passenger_id = pd.passenger_id
JOIN booking b ON p.passenger_id = b.passenger_id
JOIN flight f ON b.flight_id = f.flight_id
JOIN SP2735_flight af ON f.flightno = af.flightno
JOIN adult_passenger apg ON p.passenger_id = apg.passenger_id
ORDER BY f.flightno ASC, p.passportno ASC, fullname ASC;
                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 Sort  (cost=2183.24..2183.25 rows=3 width=55)
   Sort Key: f.flightno, p.passportno, (concat(p.firstname, ' ', p.lastname))
   ->  Nested Loop  (cost=1365.58..2183.21 rows=3 width=55)
         ->  Nested Loop  (cost=1365.29..2182.06 rows=3 width=56)
               ->  Nested Loop  (cost=1365.00..2180.85 rows=3 width=25)
                     ->  Hash Join  (cost=1364.71..2175.97 rows=10 width=17)
                           Hash Cond: (f.flightno = flight.flightno)
                           ->  Nested Loop  (cost=0.30..769.31 rows=16086 width=17)
                                 ->  Seq Scan on booking b  (cost=0.00..295.86 rows=16086 width=16)
                                 ->  Memoize  (cost=0.30..0.51 rows=1 width=17)
                                       Cache Key: b.flight_id
                                       Cache Mode: logical
                                       ->  Index Scan using idx_96453_primary on flight f  (cost=0.29..0.50 rows=1 width=17)
                                             Index Cond: (flight_id = b.flight_id)
                           ->  Hash  (cost=1364.33..1364.33 rows=6 width=9)
                                 ->  Unique  (cost=1364.24..1364.27 rows=6 width=9)
                                       ->  Sort  (cost=1364.24..1364.26 rows=6 width=9)
                                             Sort Key: flight.flightno
                                             ->  Seq Scan on flight  (cost=0.00..1364.16 rows=6 width=9)
                                                   Filter: (flightno ~~ 'SP2735%'::text)
                     ->  Index Scan using idx_96472_primary on passengerdetails  (cost=0.29..0.49 rows=1 width=8)
                           Index Cond: (passenger_id = b.passenger_id)
                           Filter: ((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM birthdate)) >= '18'::numeric)
               ->  Index Scan using idx_96468_primary on passenger p  (cost=0.29..0.40 rows=1 width=31)
                     Index Cond: (passenger_id = b.passenger_id)
         ->  Index Scan using idx_96472_primary on passengerdetails pd  (cost=0.29..0.38 rows=1 width=12)
               Index Cond: (passenger_id = p.passenger_id)
(27 rows)

Time: 7.720 ms
airport=#
```

```
airport=#
WITH SP2735_flight AS (
    SELECT DISTINCT flightno
    FROM flight
    WHERE flightno LIKE 'SP2735%'
), adult_passenger AS (
    SELECT passenger_id
    FROM passengerdetails
    WHERE EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM birthdate) >= 18
)
SELECT p.passportno, CONCAT(p.firstname, ' ', p.lastname) AS fullname, pd.birthdate, f.flightno
FROM passenger p
JOIN passengerdetails pd ON p.passenger_id = pd.passenger_id
JOIN booking b ON p.passenger_id = b.passenger_id
JOIN flight f ON b.flight_id = f.flight_id
JOIN SP2735_flight af ON f.flightno = af.flightno
JOIN adult_passenger apg ON p.passenger_id = apg.passenger_id
ORDER BY f.flightno ASC, p.passportno ASC, fullname ASC;

Time: 13.983 ms
airport=#
```

SS Eksekusi *Query* 2
Tampilkan dalam 2 SS terpisah:
- Query Plan
- Bagian *Time*

```
airport=# EXPLAIN WITH adult_passenger_data AS (
    SELECT p.passenger_id, p.passportno, CONCAT(p.firstname, ' ', p.lastname) AS fullname, pd.birthdate
    FROM passenger p
    NATURAL JOIN passengerdetails pd
    WHERE EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM birthdate) >= 18
    ORDER BY p.passportno ASC, fullname ASC
)
SELECT apd.passportno, apd.fullname, apd.birthdate, f.flightno
FROM adult_passenger_data apd, booking b, flight f, airplane ap
WHERE b.passenger_id = apd.passenger_id
AND b.flight_id = f.flight_id
AND f.airplane_id = ap.airplane_id
AND f.flightno IN (
    SELECT DISTINCT flightno
    FROM flight
    WHERE flightno LIKE 'SP2735%'
    ORDER BY flightno ASC
);
                                                  QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=5297.15..5495.48 rows=10 width=55)
   ->  Hash Join  (cost=5296.86..5492.47 rows=10 width=63)
         Hash Cond: (p.passenger_id = b.passenger_id)
         ->  Sort  (cost=3120.77..3150.85 rows=12032 width=54)
               Sort Key: p.passportno, (concat(p.firstname, ' ', p.lastname))
               ->  Hash Join  (cost=1539.54..2305.33 rows=12032 width=54)
                     Hash Cond: (p.passenger_id = pd.passenger_id)
                     ->  Seq Scan on passenger p  (cost=0.00..640.95 rows=36095 width=31)
                     ->  Hash  (cost=1389.14..1389.14 rows=12032 width=12)
                           ->  Seq Scan on passengerdetails pd  (cost=0.00..1389.14 rows=12032 width=12)
                                 Filter: ((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM birthdate)) >= '18'::numeric)
         ->  Hash  (cost=2175.97..2175.97 rows=10 width=25)
               ->  Hash Join  (cost=1364.71..2175.97 rows=10 width=25)
                     Hash Cond: (f.flightno = flight.flightno)
                     ->  Nested Loop  (cost=0.30..769.31 rows=16086 width=25)
                           ->  Seq Scan on booking b  (cost=0.00..295.86 rows=16086 width=16)
                           ->  Memoize  (cost=0.30..0.51 rows=1 width=25)
                                 Cache Key: b.flight_id
                                 Cache Mode: logical
                                 ->  Index Scan using idx_96453_primary on flight f  (cost=0.29..0.50 rows=1 width=25)
                                       Index Cond: (flight_id = b.flight_id)
                     ->  Hash  (cost=1364.33..1364.33 rows=6 width=9)
                           ->  Unique  (cost=1364.24..1364.27 rows=6 width=9)
                                 ->  Sort  (cost=1364.24..1364.26 rows=6 width=9)
                                       Sort Key: flight.flightno
                                       ->  Seq Scan on flight  (cost=0.00..1364.16 rows=6 width=9)
                                             Filter: (flightno ~~ 'SP2735%'::text)
   ->  Index Only Scan using idx_96410_primary on airplane ap  (cost=0.28..0.30 rows=1 width=8)
         Index Cond: (airplane_id = f.airplane_id)
(29 rows)

Time: 4.976 ms
airport=#
```

```
airport=# WITH adult_passenger_data AS (
    SELECT p.passenger_id, p.passportno, CONCAT(p.firstname, ' ', p.lastname) AS fullname, pd.birthdate
    FROM passenger p
    NATURAL JOIN passengerdetails pd
    WHERE EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM birthdate) >= 18
    ORDER BY p.passportno ASC, fullname ASC
)
SELECT apd.passportno, apd.fullname, apd.birthdate, f.flightno
FROM adult_passenger_data apd, booking b, flight f, airplane ap
WHERE b.passenger_id = apd.passenger_id
AND b.flight_id = f.flight_id
AND f.airplane_id = ap.airplane_id
AND f.flightno IN (
    SELECT DISTINCT flightno
    FROM flight
    WHERE flightno LIKE 'SP2735%'
    ORDER BY flightno ASC
);
Time: 62.110 ms
airport=#
```

	Query 1	**Query 2**
Kelebihan	Sudah tidak menggunakan subquery yang membuat cost operasi jadi mahal.	Sudah mengurangi subquery dengan membuat query with untuk menghitung operasi dahulu sebelum dijon.
Kekurangan	Masih banyak menggunakan operasi JOIN pada beberapa tabel padahal seharusnya ukuran tabel tersebut bisa diperkecil dengan menggunakan with atau seleksi atribut yang diperlukan sebelum dilakukan join.	Masih menggunakan corellated subquery yang membuat tiap baris dari outer relation harus mengeksekusi subquery di clausa where.

Setelah menganalisis kedua *query*, apa kesimpulan yang kamu berikan kepada Pak Julala? Berikan saranmu kepada Pak Julala untuk menghasilkan *query* lain yang mungkin lebih optimal dari *query 1* dan *query 2*!

Notes:

- Praktikan tidak perlu membuat ulang *query*. Cukup berikan saran dan penjelasan singkat.
- Praktikan dapat memilih *query* 1 atau *query* 2 sebagai jawaban apabila dianggap sudah optimal.
- Pastikan usulan yang praktikan berikan sekiranya menghasilkan makna semantik yang sama!

Kesimpulan dan Saran
Query 1 lebih optimal daripada query 2, terbukti dari costnya yang 2x lipat lebih kecil dari query 2 dan execution timenya yang jauh lebih cepat daripada query 2. Dari perbandingan kedua query tersebut, disimpulkan bahwa penggunaan correlated subquery sangat tidak efektif apalagi untuk query kompleks karena akan membuat tiap baris di outer relation mengeksekusi subquery berulang-ulang. Untuk mengoptimasi query juga dapat dilakukan dengan menyeleksi atribut-atribut yang diperlukan sebelum dilakukan operasi join.

b. Pak Jay ga mau kalah dengan Pak Julala, dia membuat query sekompleks mungkin. Berikut query yang dibuatnya:

Query Pak Jay	`SELECT`
	`    f.flight_id,`
	`    a.airplane_id,`
	`    al.airlinename AS airline_name,`
	`    ap.name AS departure_airport,`
	`    SUM(b.price) AS total_revenue`
	`FROM booking b`
	`JOIN flight f ON b.flight_id = f.flight_id`
	`JOIN airplane a ON f.airplane_id = a.airplane_id`
	`JOIN airline al ON f.airline_id = al.airline_id`
	`JOIN airport ap ON f.from = ap.airport_id`
	`WHERE`
	`    al.airlinename = 'Spain Airlines'`
	`GROUP BY`
	`    f.flight_id, a.airplane_id, al.airlinename, ap.name`
	`HAVING`
	`    SUM(b.price) > 500`
	`ORDER BY`
	`    total_revenue DESC;`

SS Hasil Query	

flight_id	airplane_id	airline_name	departure_airport	total_revenue
12159	3	Spain Airlines	CAPITAN MONTES	53859.89
94050	3	Spain Airlines	OCHSENFURT	53571.94
77582	3	Spain Airlines	MOSTARDAS	52830.55
36709	3	Spain Airlines	KILIMANJARO INTL	52490.37
36717	3	Spain Airlines	CABO ROJO	51224.49
53085	3	Spain Airlines	INKISI	51044.63
36697	3	Spain Airlines	PUKARUA	50775.27
36695	3	Spain Airlines	ST GEORGES	49860.49
53077	3	Spain Airlines	KLAMATH FALLS INTL	49275.74
28428	3	Spain Airlines	ANAKTUVUK PASS	49074.90
94026	3	Spain Airlines	ARVIKA	48924.80
36711	3	Spain Airlines	MOSTARDAS	48336.99
20253	3	Spain Airlines	CACERES	48191.71
69479	3	Spain Airlines	SOURE	47845.33
53087	3	Spain Airlines	ISLE OF MAN	47765.41
12164	3	Spain Airlines	LYNCHBURG REGL-GLENN	47593.04
94041	3	Spain Airlines	PLOCK	47509.76
12175	3	Spain Airlines	BLACKBUSHE	47490.63
94059	5	Spain Airlines	KUMEJIMA	47236.44
53104	3	Spain Airlines	CACOAL	47209.80
85767	3	Spain Airlines	KUMEJIMA	47000.14
20232	3	Spain Airlines	KLAMATH FALLS INTL	46858.89
44842	5	Spain Airlines	MC ALESTER REGL	46385.50
85744	3	Spain Airlines	NIOAQUE	46201.31
69485	3	Spain Airlines	RARUP	46130.85
28416	3	Spain Airlines	CAMBRIDGE BAY	46120.06
3866	3	Spain Airlines	RICE LAKE REGL-CARL'S	45849.91
85751	3	Spain Airlines	ALAMOGORDO-WHITE SANDS REGL	45829.62
53108	3	Spain Airlines	MC ALESTER REGL	45494.20
61215	3	Spain Airlines	ELMENDORF AFB	45450.28
77590	5	Spain Airlines	CACERES	45158.76
3853	3	Spain Airlines	ARVIKA	45135.95
69504	5	Spain Airlines	CAXIAS	44450.08
61192	5	Spain Airlines	PELEE ISLAND	44182.79
28437	3	Spain Airlines	BOUAKE	43627.52
20228	5	Spain Airlines	LODWAR	42888.14
69497	5	Spain Airlines	BRANTFORD	42687.40
20255	3	Spain Airlines	NAPLES MUN	42370.56
94056	5	Spain Airlines	CONNABARABRAN	42271.44
53079	5	Spain Airlines	STRONSAY	42224.48
61196	5	Spain Airlines	STRONSAY	42176.52
28403	5	Spain Airlines	DEBOLT	42027.47
20257	3	Spain Airlines	SERENJE	42026.88
61224	3	Spain Airlines	MC ALESTER REGL	41994.42
69480	5	Spain Airlines	HASSAN I	41900.09
12148	5	Spain Airlines	RARUP	41741.31
3887	5	Spain Airlines	MC ALESTER REGL	41437.66
28410	5	Spain Airlines	PINEY	41403.12
28406	5	Spain Airlines	HASSAN I	41215.85
12141	5	Spain Airlines	MILLIKEN	40948.01
61202	5	Spain Airlines	ISLE OF MAN	40716.02
85775	3	Spain Airlines	MARSHALL MUN-RYAN	40208.70
61194	5	Spain Airlines	DEBOLT	39993.45
69490	5	Spain Airlines	STA FE DE ANTIOQUIA	39960.30
61190	5	Spain Airlines	ARVIKA	39574.14
44819	5	Spain Airlines	BORLANGE AB	38727.26
77573	5	Spain Airlines	THAKHEK	37994.88
85770	5	Spain Airlines	CACOAL	37698.50
53100	5	Spain Airlines	NAPLES MUN	36446.39

:

Namun, hasil query Pak Jay sepertinya masih kurang efektif dan memakan cukup banyak waktu. Bantulah Pak Jay memperbaiki query tersebut agar lebih optimal dan mengeluarkan hasil yang sama!

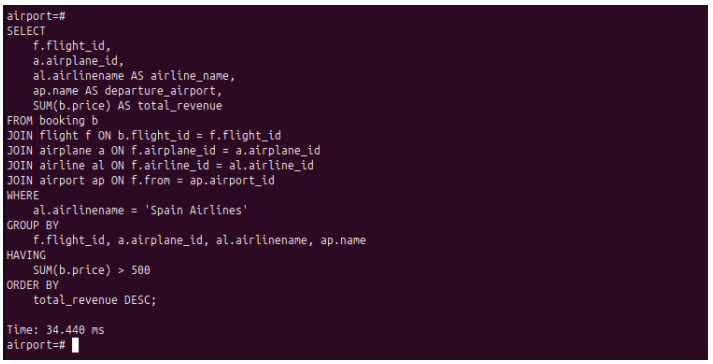
Query Optimalisasi	`with above500 as (`
	`SELECT`
	`f.flight_id,`
	`f.airline_id,`
	`f.airplane_id,`
	`ap.name AS departure_airport,`
	`SUM(b.price) AS total_revenue`
	`FROM booking b`

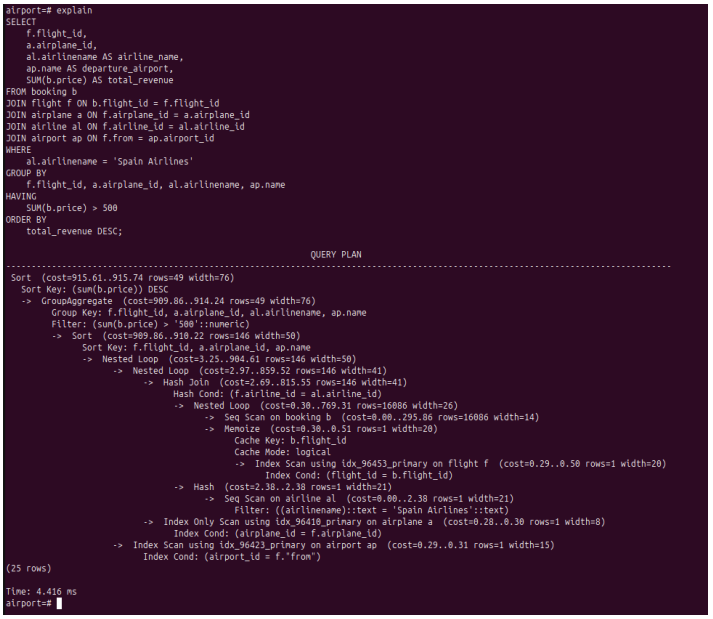
```
      JOIN flight f ON b.flight_id =
f.flight_id
      JOIN airport ap ON f.from =
ap.airport_id
      GROUP BY
      f.flight_id, f.airplane_id,
al.airlinename, ap.name
      HAVING
      SUM(b.price) > 500
)


SELECT
      f.flight_id,
      ab.airplane_id,
      al.airlinename AS airline_name,
      ap.name AS departure_airport,
      SUM(b.price) AS total_revenue
FROM above500 a
JOIN airplane ab ON a.airplane_id =
ab.airplane_id
JOIN airline al ON a.airline_id =
al.airline_id
WHERE
      al.airlinename = 'Spain Airlines'
ORDER BY
      total_revenue DESC;
```

SS Hasil Query Optimalisasi

SS Waktu Eksekusi & Proses Eksekusi *Query* Sebelum Optimalisasi



	
SS Waktu Eksekusi & Proses Eksekusi *Query* Setelah Optimalisasi	
Penjelasan Singkat Optimalisasi yang dilakukan	Optimalisasi dilakukan dengan menghitung terlebih dahulu agregasi sehingga agregasi untuk harga dilakukan terlebih dahulu. Hal ini akan mengurangi waktu saat perhitungan agregasi karena ukuran tabel yang besar. Setelah itu, hasil dari query with baru dijoinkan dengan tabel lainnya untuk diambil kolom yang sesuai.

Pembagian Kerja

NIM	Tugas
13522053	Semua
13522116	Semua