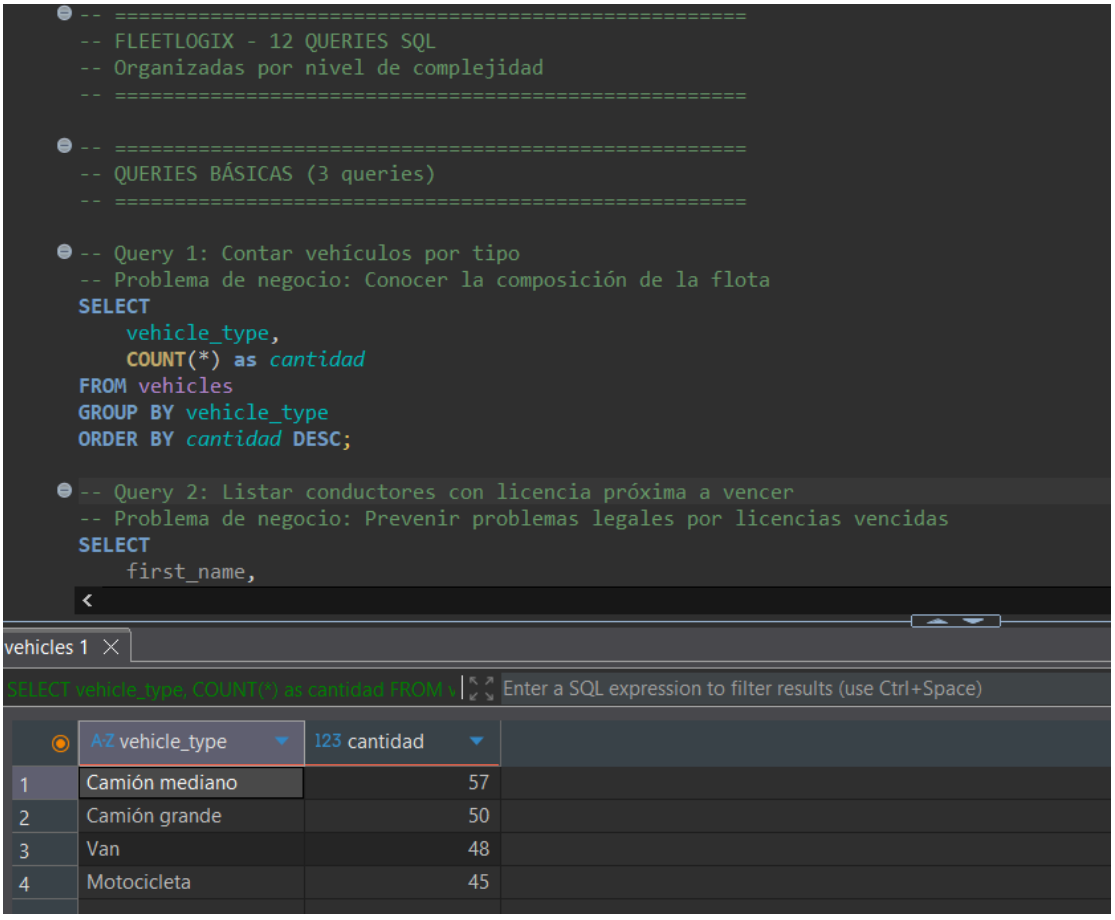
Avance 2: Ejecución y análisis de 12 queries proporcionadas

Una vez poblada la base, la empresa busca respuestas a problemas operativos reales, como eficiencia de rutas, carga de trabajo de conductores o mantenimiento de vehículos. Para ello, se deben analizar los planes de ejecución, justificar los problemas de negocio que resuelve cada query y crear índices que mejoren el rendimiento.

A. Queries básicas:

1. Conocer la composición de la flota: A partir de esta consulta podemos ver que el tipo de vehículo que predomina en nuestra flota es el camión mediano, seguido por las motocicletas, las vans y los camiones grandes.



```
-- ========================================================
-- FLEETLOGIX - 12 QUERIES SQL
-- Organizadas por nivel de complejidad
-- ========================================================

-- ========================================================
-- QUERIES BÁSICAS (3 queries)
-- ========================================================

-- Query 1: Contar vehículos por tipo
-- Problema de negocio: Conocer la composición de la flota
SELECT
    vehicle_type,
    COUNT(*) as cantidad
FROM vehicles
GROUP BY vehicle_type
ORDER BY cantidad DESC;

-- Query 2: Listar conductores con licencia próxima a vencer
-- Problema de negocio: Prevenir problemas legales por licencias vencidas
SELECT
    first_name,
```

vehicles 1

SELECT vehicle_type, COUNT(*) as cantidad FROM v | Enter a SQL expression to filter results (use Ctrl+Space)

	A-Z vehicle_type	123 cantidad
1	Camión mediano	57
2	Camión grande	50
3	Van	48
4	Motocicleta	45

2. Listar conductores con licencia próxima a vencer.

```sql
-- Query 2: Listar conductores con licencia próxima a vencer
-- Problema de negocio: Prevenir problemas legales por licencias vencidas
SELECT
    first_name,
    last_name,
    license_number,
    license_expiry
FROM drivers
WHERE license_expiry < CURRENT_DATE + INTERVAL '30 days'
ORDER BY license_expiry;
```

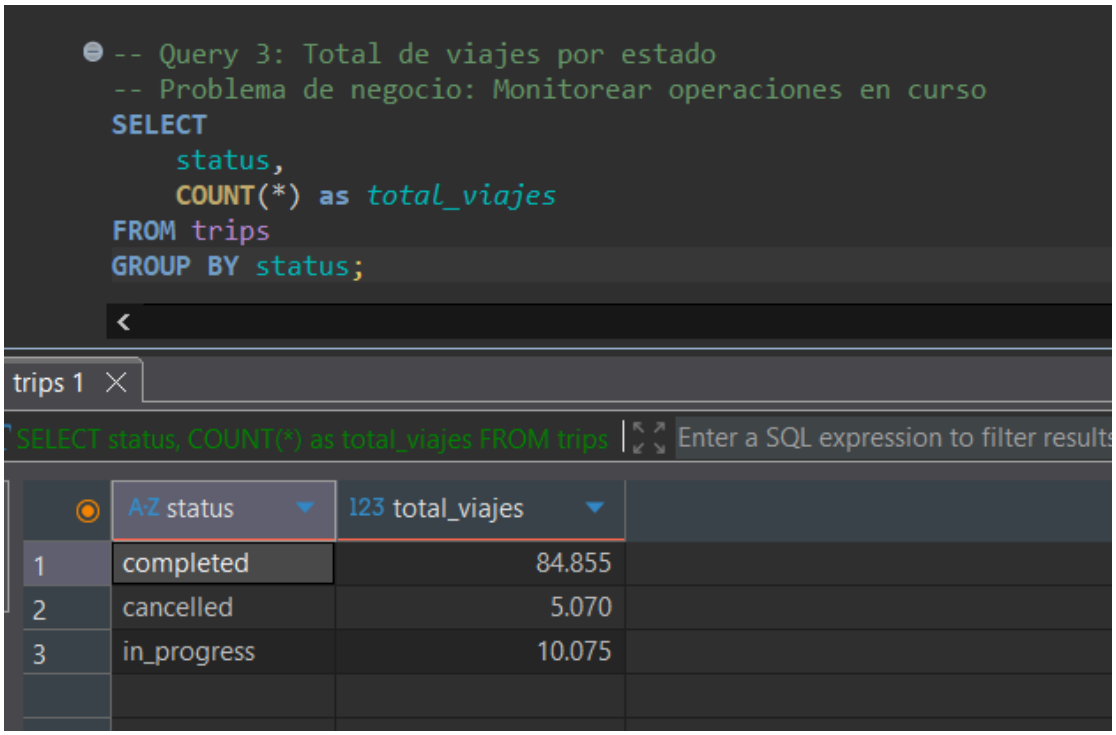
drivers 1

SELECT first_name, last_name, license_number, licer | Enter a SQL expression to filter results (use Ctrl+Space)

	first_name	last_name	license_number	license_expiry
1	Vidal	Vall	LIC33817	2025-09-26
2	Ruperta	Zaragoza	LIC58572	2025-09-28
3	Olivia	Barrio	LIC28370	2025-10-01
4	Sandra	Asensio	LIC87534	2025-10-02
5	Leonel	Diaz	LIC62482	2025-10-03
6	Rufino	Priego	LIC93542	2025-10-04
7	Toni	Duran	LIC18733	2025-10-06
8	Pepita	Viñas	LIC57223	2025-10-06
9	Vinicio	Maestre	LIC95707	2025-10-06

A partir de esta consulta podemos prevenir problemas legales por licencias vencidas y poder iniciar los trámites de renovación de una forma anticipada. En este caso, se seleccionan los conductores cuya licencia esta próxima a vencer dentro de los próximos 30 dias.

3. Comprobar el número de viajes por status.
   La idea de esta query es monitoriear la situación de las operaciones en curso. En este caso llevamos 85 mil viajes completados, 5 mil cancelados y 10 mil en curso.



B. Queries Intermedias:

4. Identificar la demanda por ciudad. En este caso hay una gran demanda de envíos a Buenos Aires y Cordoba y en menor medida a Rosario y Mendoza

```sql
-- ====================================================
-- QUERIES INTERMEDIAS (5 queries)
-- ====================================================

-- Query 4: Total de entregas por ciudad destino en los últimos 2 meses
-- Problema de negocio: Identificar demanda por ciudad para planificación de recursos
SELECT
    r.destination_city,
    COUNT(DISTINCT t.trip_id) as total_viajes,
    COUNT(d.delivery_id) as total_entregas,
    SUM(d.package_weight_kg) as peso_total_kg
FROM routes r
INNER JOIN trips t ON r.route_id = t.route_id
INNER JOIN deliveries d ON t.trip_id = d.trip_id
WHERE t.departure_datetime >= CURRENT_DATE - INTERVAL '60 days'
GROUP BY r.destination_city
ORDER BY total_entregas DESC;
```

routes 1

'SELECT r.destination_city, COUNT(DISTINCT t.trip_ic | Enter a SQL expression to filter results (use Ctrl+Space)

	A-Z destination_city	123 total_viajes	123 total_entregas	123 peso_total_kg
1	Mar del Plata	1.147	4.635	5.069.260,12
2	Bahía Blanca	827	3.279	3.476.540,01
3	Buenos Aires	696	2.799	2.830.650,26
4	La Plata	684	2.752	2.785.818,67
5	Salta	671	2.683	2.633.244,63
6	Mendoza	646	2.559	2.791.324,96
7	Córdoba	592	2.369	2.338.945,21
8	Rosario	560	2.262	2.278.992,66

5. Conductores activos con más viajes completados. El objetivo es evaluar la eficiencia de los conductores en cuanto a cantidad de viajes realizados y completados.

```sql
-- Query 5: Conductores activos con cantidad de viajes completados
-- Problema de negocio: Evaluar carga de trabajo por conductor
SELECT
    d.driver_id,
    d.first_name || ' ' || d.last_name as nombre_completo,
    d.license_expiry,
    COUNT(t.trip_id) as viajes_totales,
    SUM(CASE WHEN t.status = 'completed' THEN 1 ELSE 0 END) as viajes_completados
FROM drivers d
LEFT JOIN trips t ON d.driver_id = t.driver_id
WHERE d.status = 'active'
GROUP BY d.driver_id, d.first_name, d.last_name, d.license_expiry
HAVING COUNT(t.trip_id) > 0
ORDER BY viajes_completados DESC;
```

drivers 1

Enter a SQL expression to filter results (use Ctrl+Space)

	driver_id	nombre_completo	license_expiry	viajes_totales	viajes_completados
1	179	Sofía Garriga	2025-11-19	600	521
2	118	Elisabet Alcolea	2026-07-12	594	519
3	317	Victorino Zurita	2026-09-16	602	515
4	175	Virginia Briones	2026-01-08	585	515
5	143	Mar Verdugo	2026-04-17	591	511
6	129	Hortensia Blanch	2026-09-02	587	509
7	313	Gloria Rosado	2026-07-11	599	509
8	166	Rufino Priego	2025-10-04	597	508
9	356	Felix Mayoral	2026-01-02	578	506
10	101	Emiliana Ortega	2026-09-17	589	505
11	365	Hernán Torrents	2026-09-12	583	505

6. Promedio de entregas por conductor en los últimos 6 meses.

Esta consulta nos permite evaluar la productividad por conductor en cuanto a entregas por viajes realizados. En este caso el conductor más eficiente es Sol Andreu con un promedio de 4,31 entregas por viaje realizado. Sin embargo, si evaluamos la eficiencia de entregas por día, Zoraida Soria lidera en la tabla, ya que tiene un promedio de 2.31 entregas por día.

```sql
-- Query 6: Promedio de entregas por conductor en los últimos 6 meses
-- Problema de negocio: Medir productividad individual de conductores
SELECT
    dr.driver_id,
    dr.first_name || ' ' || dr.last_name as conductor,
    COUNT(DISTINCT t.trip_id) as total_viajes,
    COUNT(d.delivery_id) as total_entregas,
    ROUND(COUNT(d.delivery_id)::NUMERIC / NULLIF(COUNT(DISTINCT t.trip_id), 0), 2) as promedio_entregas_por_viaje,
    ROUND(COUNT(d.delivery_id)::NUMERIC / 180, 2) as promedio_entregas_diarias
FROM drivers dr
INNER JOIN trips t ON dr.driver_id = t.driver_id
INNER JOIN deliveries d ON t.trip_id = d.trip_id
WHERE t.departure_datetime >= CURRENT_DATE - INTERVAL '6 months'
    AND t.status = 'completed'
GROUP BY dr.driver_id, dr.first_name, dr.last_name
HAVING COUNT(DISTINCT t.trip_id) >= 10
ORDER BY promedio_entregas_por_viaje DESC;
```

drivers 1

Enter a SQL expression to filter results (use Ctrl+Space)

	driver_id	conductor	total_viajes	total_entregas	promedio_entregas_por_viaje	promedio_entregas_diarias
1	272	Sol Andreu	68	295	4,34	1,6
2	193	Javiera Ortuño	93	402	4,32	2,2
3	206	Zoraida Soria	97	416	4,29	2,3
4	1	Nidia Quintana	82	347	4,23	1,9
5	5	Cruz Almansa	73	308	4,22	1,7
6	259	Horacio Carvajal	94	396	4,21	2,
7	83	Sonia Peinado	90	378	4,2	2,
8	128	Serafina Madrigal	75	315	4,2	1,7
9	224	Pepita Viñas	95	398	4,19	2,2

C. Queries Complejas:

9. Costo de mantenimiento por kilómetro recorrido. Permite evaluar el costo – beneficio de cada tipo de vehículo.

```sql
-- Query 9: Costo de mantenimiento por kilómetro recorrido
-- Problema de negocio: Evaluar costo-beneficio de cada tipo de vehículo
WITH vehicle_metrics AS (
    SELECT
        v.vehicle_id,
        v.vehicle_type,
        v.license_plate,
        COUNT(DISTINCT t.trip_id) as total_viajes,
        SUM(r.distance_km) as km_totales,
        SUM(m.cost) as costo_mantenimiento_total,
        COUNT(DISTINCT m.maintenance_id) as cantidad_mantenimientos
    FROM vehicles v
    LEFT JOIN trips t ON v.vehicle_id = t.vehicle_id
    LEFT JOIN routes r ON t.route_id = r.route_id
    LEFT JOIN maintenance m ON v.vehicle_id = m.vehicle_id
    WHERE t.status = 'completed'
    GROUP BY v.vehicle_id, v.vehicle_type, v.license_plate
)
SELECT
    vehicle_type,
    COUNT(vehicle_id) as cantidad_vehiculos,
    SUM(total_viajes) as viajes_totales,
    SUM(km_totales) as kilometros_totales,
    SUM(costo_mantenimiento_total) as costo_total_mantenimiento,
```

ehicles 1

Enter a SQL expression to filter results (use Ctrl+Space)

	vehicle_type	cantidad_vehiculos	viajes_totales	kilometros_totales	costo_total_mantenimiento	costo_por_km	costo_pro
1	Motocicleta	17	22.994	429.150.641,48	12.956.015.508,78	30,19	
2	Van	9	11.930	216.185.583,45	6.336.351.248,39	29,31	
3	Camión mediano	21	28.315	459.396.714,65	13.168.018.457,8	28,66	
4	Camión grande	16	21.616	345.214.465,06	9.853.199.197,96	28,54	

12. Entregas por día y horario de semana. Permite optimizar horarios de operación personal.

```sql
-- Query 12: Pivot de entregas por hora y día de la semana
-- Problema de negocio: Optimizar horarios de operación y personal
WITH entregas_por_hora_dia AS (
    SELECT
        EXTRACT(DOW FROM scheduled_datetime) as dia_semana,
        EXTRACT(HOUR FROM scheduled_datetime) as hora,
        COUNT(*) as cantidad_entregas
    FROM deliveries
    WHERE scheduled_datetime >= CURRENT_DATE - INTERVAL '60 days'
    GROUP BY EXTRACT(DOW FROM scheduled_datetime), EXTRACT(HOUR FROM scheduled_datetime)
)
SELECT
    hora,
    SUM(CASE WHEN dia_semana = 0 THEN cantidad_entregas ELSE 0 END) as domingo,
    SUM(CASE WHEN dia_semana = 1 THEN cantidad_entregas ELSE 0 END) as lunes,
    SUM(CASE WHEN dia_semana = 2 THEN cantidad_entregas ELSE 0 END) as martes,
    SUM(CASE WHEN dia_semana = 3 THEN cantidad_entregas ELSE 0 END) as miercoles,
    SUM(CASE WHEN dia_semana = 4 THEN cantidad_entregas ELSE 0 END) as jueves,
    SUM(CASE WHEN dia_semana = 5 THEN cantidad_entregas ELSE 0 END) as viernes,
    SUM(CASE WHEN dia_semana = 6 THEN cantidad_entregas ELSE 0 END) as sabado,
    SUM(cantidad_entregas) as total_semana
FROM entregas_por_hora_dia
WHERE hora BETWEEN 6 AND 22
GROUP BY hora
ORDER BY hora;
```

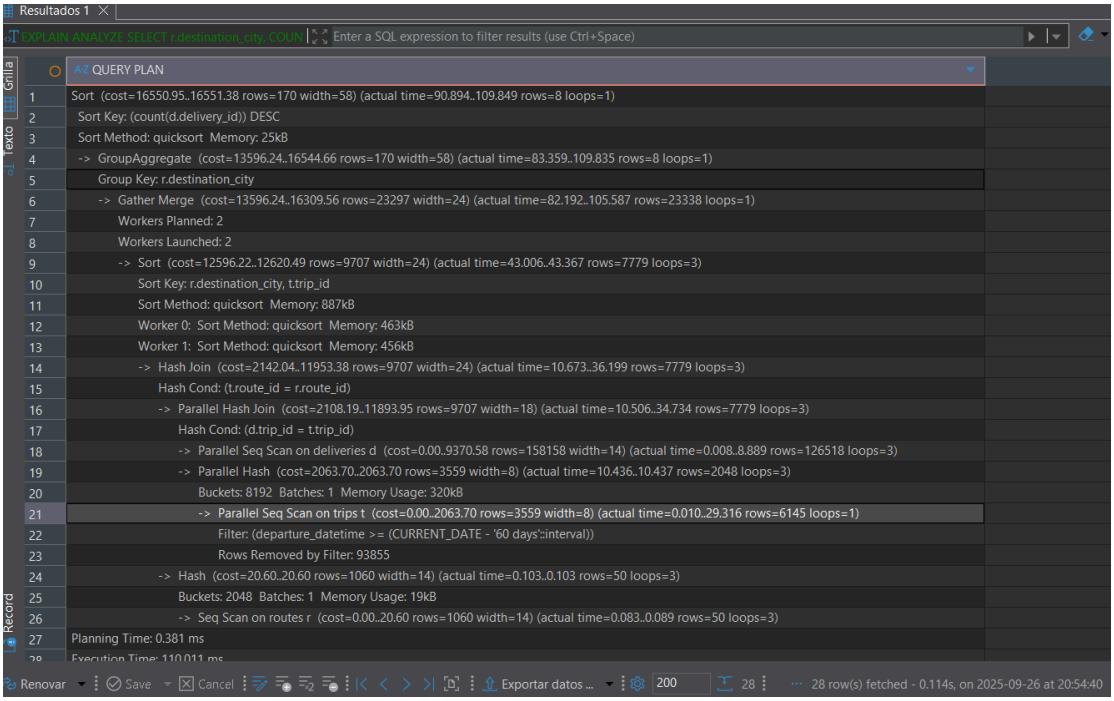
esultados 1

WITH entregas_por_hora_dia AS (SELECT EXTRACT | Enter a SQL expression to filter results (use Ctrl+Space)

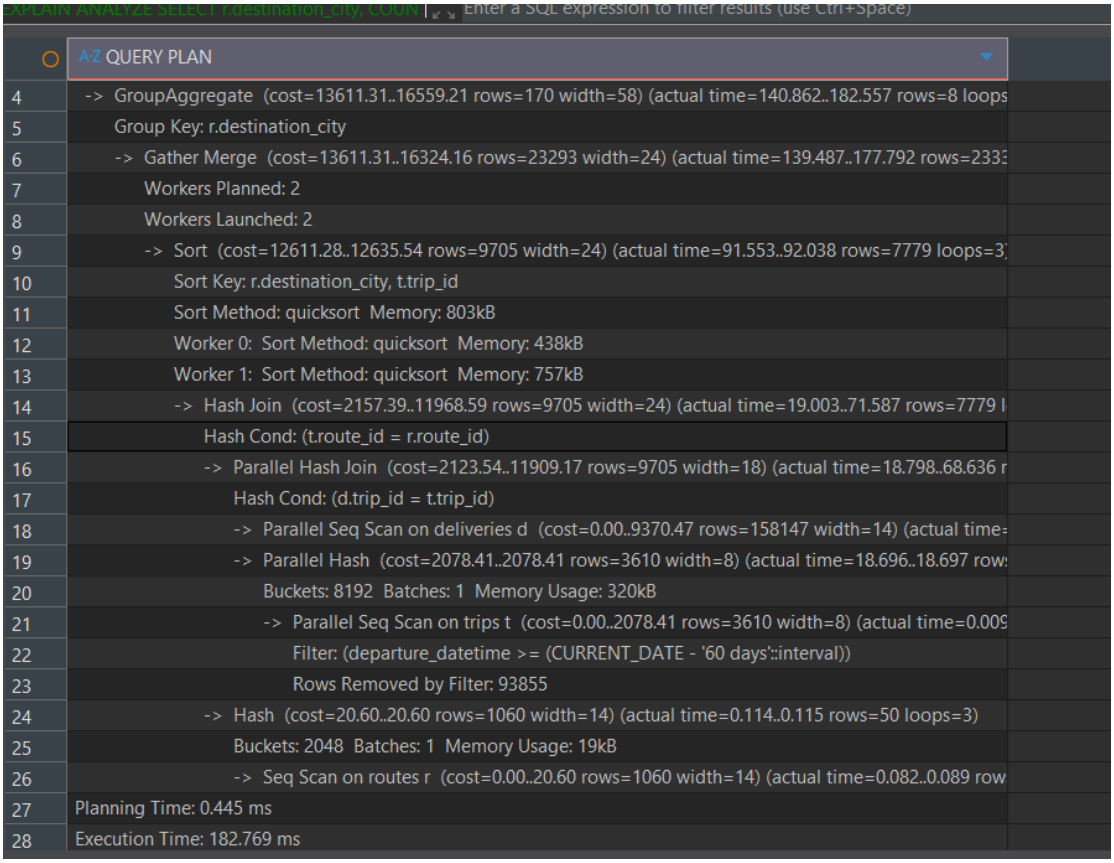
	123 hora	123 domingo	123 lunes	123 martes	123 miercoles	123 jueves	123 viernes	123 sabado	123 total_semana
1	6	102	81	112	90	93	100	92	670
2	7	86	103	111	117	89	102	101	709
3	8	104	111	139	113	110	127	105	809
4	9	121	137	168	140	128	141	132	967
5	10	141	198	186	176	163	154	148	1.166
6	11	157	190	181	187	163	173	148	1.199
7	12	164	207	180	161	179	132	138	1.161

Query 4

- Sin índice 1

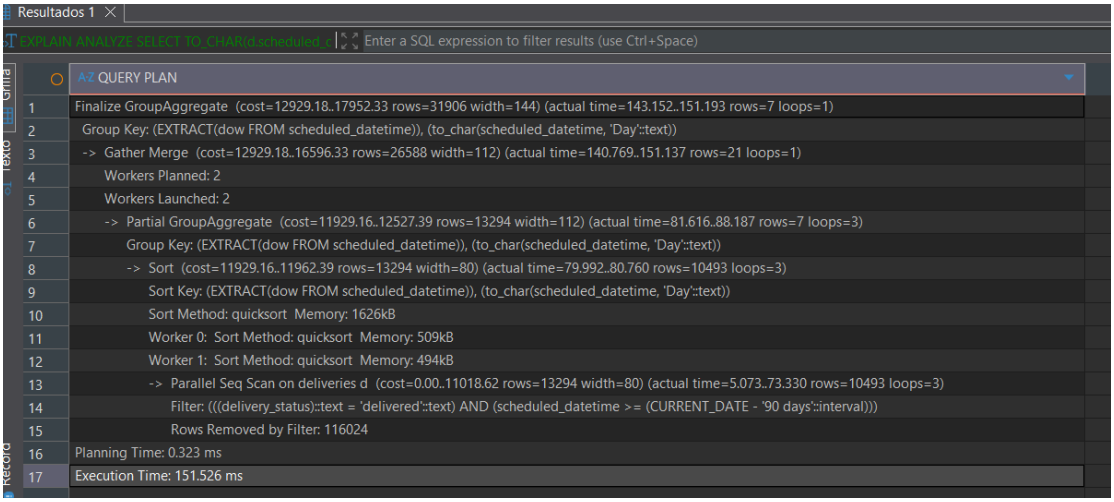

Resultados 1
EXPLAIN ANALYZE SELECT r.destination_city, COUN
Enter a SQL expression to filter results (use Ctrl+Space)
Grilla
Texto
Record
QUERY PLAN
1 Sort (cost=16550.95..16551.38 rows=170 width=58) (actual time=90.894..109.849 rows=8 loops=1)
2 Sort Key: (count(d.delivery_id)) DESC
3 Sort Method: quicksort Memory: 25kB
4 -> GroupAggregate (cost=13596.24..16544.66 rows=170 width=58) (actual time=83.359..109.835 rows=8 loops=1)
5 Group Key: r.destination_city
6 -> Gather Merge (cost=13596.24..16309.56 rows=23297 width=24) (actual time=82.192..105.587 rows=23338 loops=1)
7 Workers Planned: 2
8 Workers Launched: 2
9 -> Sort (cost=12596.22..12620.49 rows=9707 width=24) (actual time=43.006..43.367 rows=7779 loops=3)
10 Sort Key: r.destination_city, t.trip_id
11 Sort Method: quicksort Memory: 887kB
12 Worker 0: Sort Method: quicksort Memory: 463kB
13 Worker 1: Sort Method: quicksort Memory: 456kB
14 -> Hash Join (cost=2142.04..11953.38 rows=9707 width=24) (actual time=10.673..36.199 rows=7779 loops=3)
15 Hash Cond: (t.route_id = r.route_id)
16 -> Parallel Hash Join (cost=2108.19..11893.95 rows=9707 width=18) (actual time=10.506..34.734 rows=7779 loops=3)
17 Hash Cond: (d.trip_id = t.trip_id)
18 -> Parallel Seq Scan on deliveries d (cost=0.00..9370.58 rows=158158 width=14) (actual time=0.008..8.889 rows=126518 loops=3)
19 -> Parallel Hash (cost=2063.70..2063.70 rows=3559 width=8) (actual time=10.436..10.437 rows=2048 loops=3)
20 Buckets: 8192 Batches: 1 Memory Usage: 320kB
21 -> Parallel Seq Scan on trips t (cost=0.00..2063.70 rows=3559 width=8) (actual time=0.010..29.316 rows=6145 loops=1)
22 Filter: (departure_datetime >= (CURRENT_DATE - '60 days'::interval))
23 Rows Removed by Filter: 93855
24 -> Hash (cost=20.60..20.60 rows=1060 width=14) (actual time=0.103..0.103 rows=50 loops=3)
25 Buckets: 2048 Batches: 1 Memory Usage: 19kB
26 -> Seq Scan on routes r (cost=0.00..20.60 rows=1060 width=14) (actual time=0.083..0.089 rows=50 loops=3)
27 Planning Time: 0.381 ms
Renovar
Save
Cancel
Exportar datos ...
200
28
28 row(s) fetched - 0.114s, on 2025-09-26 at 20:54:40

- Con índice 1

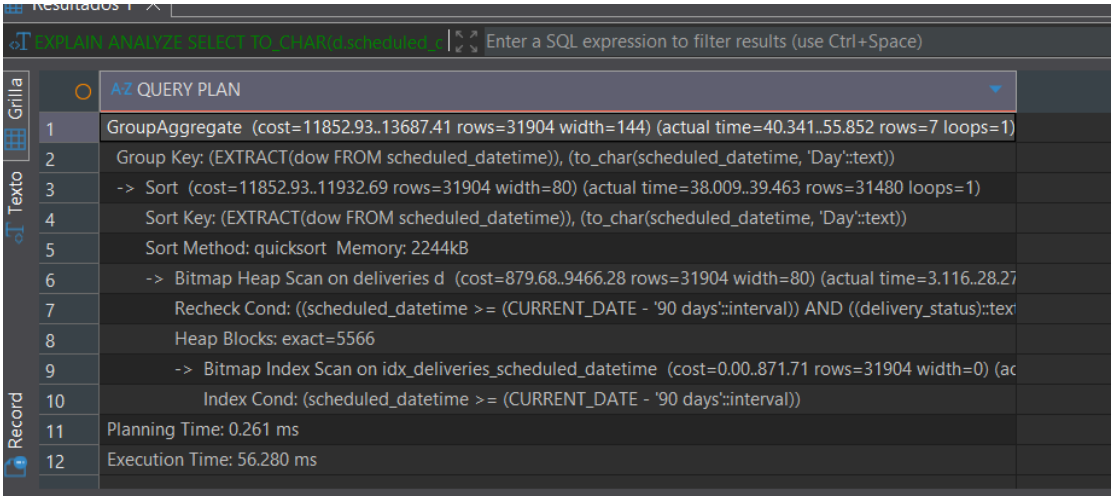

QUERY PLAN
4 -> GroupAggregate (cost=13611.31..16559.21 rows=170 width=58) (actual time=140.862..182.557 rows=8 loops
5 Group Key: r.destination_city
6 -> Gather Merge (cost=13611.31..16324.16 rows=23293 width=24) (actual time=139.487..177.792 rows=233
7 Workers Planned: 2
8 Workers Launched: 2
9 -> Sort (cost=12611.28..12635.54 rows=9705 width=24) (actual time=91.553..92.038 rows=7779 loops=3
10 Sort Key: r.destination_city, t.trip_id
11 Sort Method: quicksort Memory: 803kB
12 Worker 0: Sort Method: quicksort Memory: 438kB
13 Worker 1: Sort Method: quicksort Memory: 757kB
14 -> Hash Join (cost=2157.39..11968.59 rows=9705 width=24) (actual time=19.003..71.587 rows=7779 l
15 Hash Cond: (t.route_id = r.route_id)
16 -> Parallel Hash Join (cost=2123.54..11909.17 rows=9705 width=18) (actual time=18.798..68.636 r
17 Hash Cond: (d.trip_id = t.trip_id)
18 -> Parallel Seq Scan on deliveries d (cost=0.00..9370.47 rows=158147 width=14) (actual time=
19 -> Parallel Hash (cost=2078.41..2078.41 rows=3610 width=8) (actual time=18.696..18.697 rows
20 Buckets: 8192 Batches: 1 Memory Usage: 320kB
21 -> Parallel Seq Scan on trips t (cost=0.00..2078.41 rows=3610 width=8) (actual time=0.009
22 Filter: (departure_datetime >= (CURRENT_DATE - '60 days'::interval))
23 Rows Removed by Filter: 93855
24 -> Hash (cost=20.60..20.60 rows=1060 width=14) (actual time=0.114..0.115 rows=50 loops=3)
25 Buckets: 2048 Batches: 1 Memory Usage: 19kB
26 -> Seq Scan on routes r (cost=0.00..20.60 rows=1060 width=14) (actual time=0.082..0.089 rows
27 Planning Time: 0.445 ms
28 Execution Time: 182.769 ms

Query 8 –

- Sin indice 2

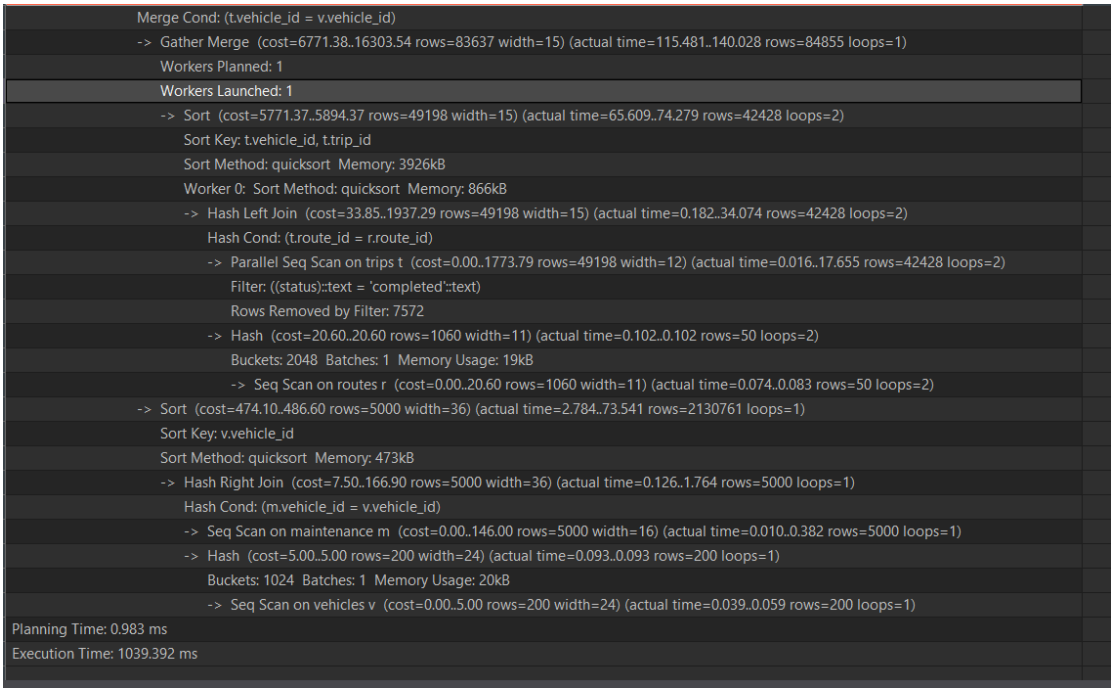


- Con indice 2

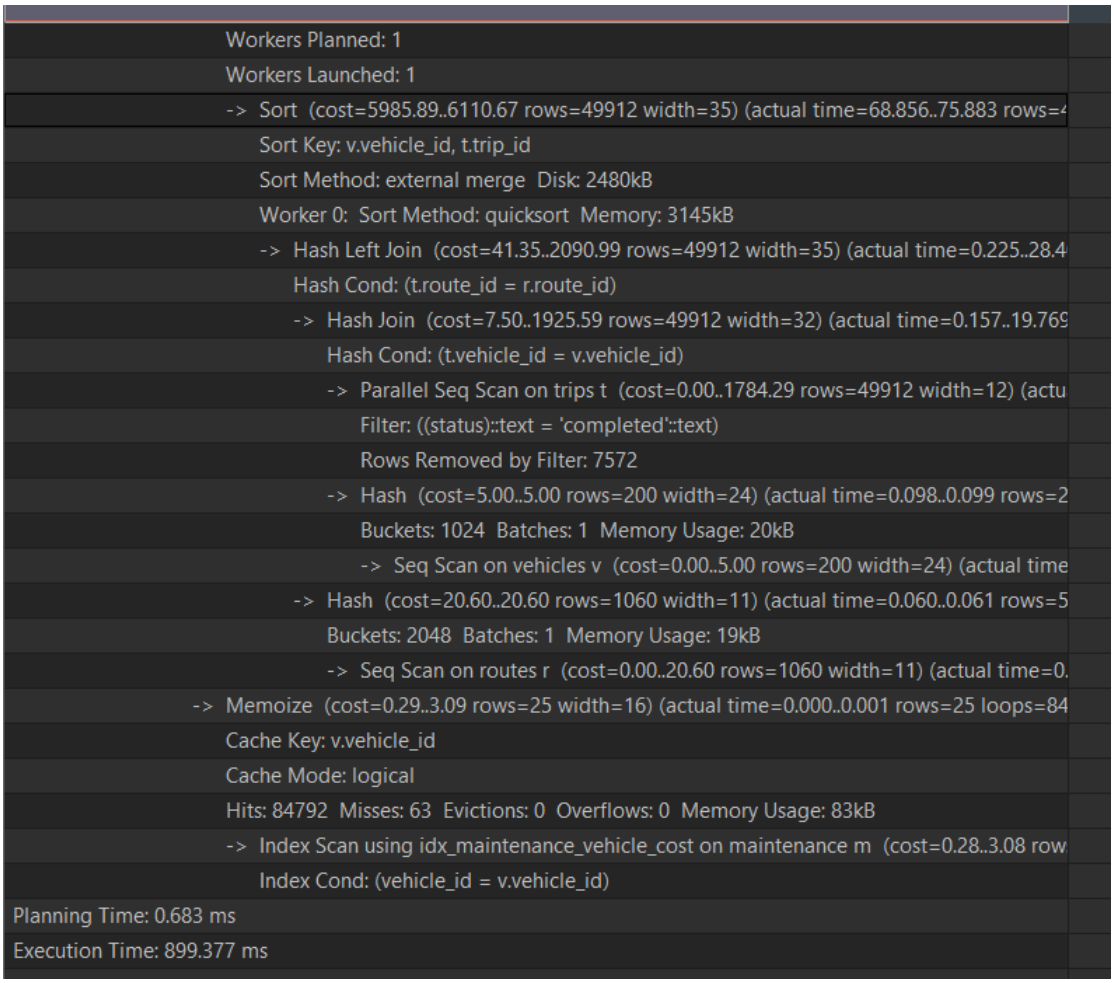


Query 9

- Sin indice 3

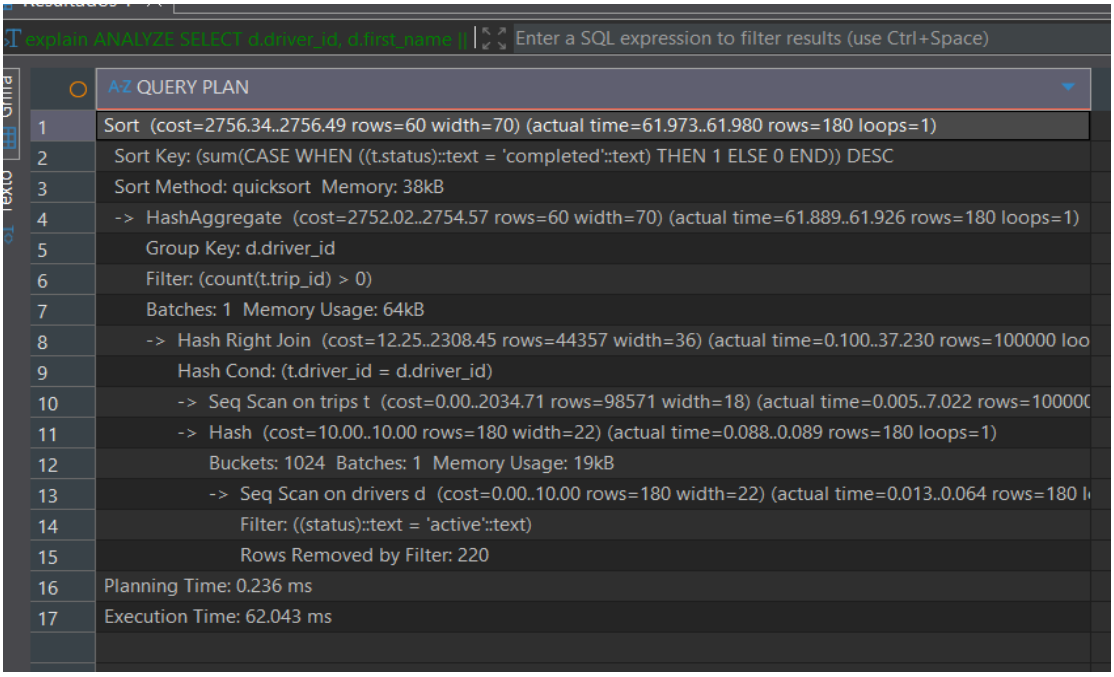

Merge Cond: (t.vehicle_id = v.vehicle_id)
-> Gather Merge (cost=6771.38..16303.54 rows=83637 width=15) (actual time=115.481..140.028 rows=84855 loops=1)
Workers Planned: 1
Workers Launched: 1
-> Sort (cost=5771.37..5894.37 rows=49198 width=15) (actual time=65.609..74.279 rows=42428 loops=2)
Sort Key: t.vehicle_id, t.trip_id
Sort Method: quicksort Memory: 3926kB
Worker 0: Sort Method: quicksort Memory: 866kB
-> Hash Left Join (cost=33.85..1937.29 rows=49198 width=15) (actual time=0.182..34.074 rows=42428 loops=2)
Hash Cond: (t.route_id = r.route_id)
-> Parallel Seq Scan on trips t (cost=0.00..1773.79 rows=49198 width=12) (actual time=0.016..17.655 rows=42428 loops=2)
Filter: ((status)::text = 'completed'::text)
Rows Removed by Filter: 7572
-> Hash (cost=20.60..20.60 rows=1060 width=11) (actual time=0.102..0.102 rows=50 loops=2)
Buckets: 2048 Batches: 1 Memory Usage: 19kB
-> Seq Scan on routes r (cost=0.00..20.60 rows=1060 width=11) (actual time=0.074..0.083 rows=50 loops=2)
-> Sort (cost=474.10..486.60 rows=5000 width=36) (actual time=2.784..73.541 rows=2130761 loops=1)
Sort Key: v.vehicle_id
Sort Method: quicksort Memory: 473kB
-> Hash Right Join (cost=7.50..166.90 rows=5000 width=36) (actual time=0.126..1.764 rows=5000 loops=1)
Hash Cond: (m.vehicle_id = v.vehicle_id)
-> Seq Scan on maintenance m (cost=0.00..146.00 rows=5000 width=16) (actual time=0.010..0.382 rows=5000 loops=1)
-> Hash (cost=5.00..5.00 rows=200 width=24) (actual time=0.093..0.093 rows=200 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 20kB
-> Seq Scan on vehicles v (cost=0.00..5.00 rows=200 width=24) (actual time=0.039..0.059 rows=200 loops=1)
Planning Time: 0.983 ms
Execution Time: 1039.392 ms

- Con Indice 3

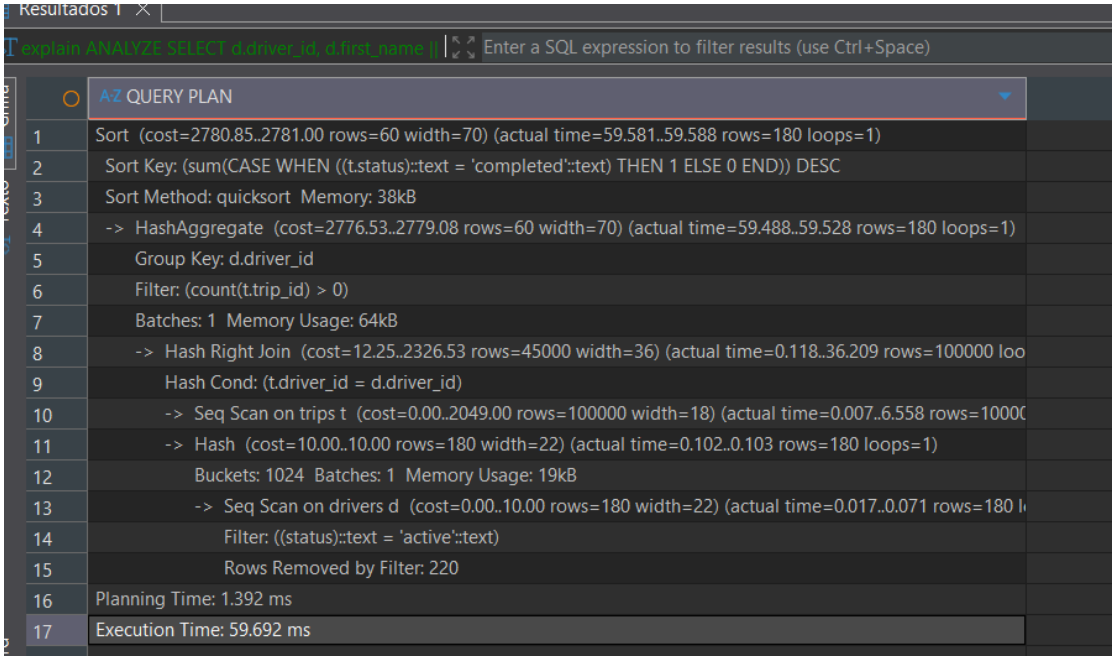


Query 5

- sin indice 4

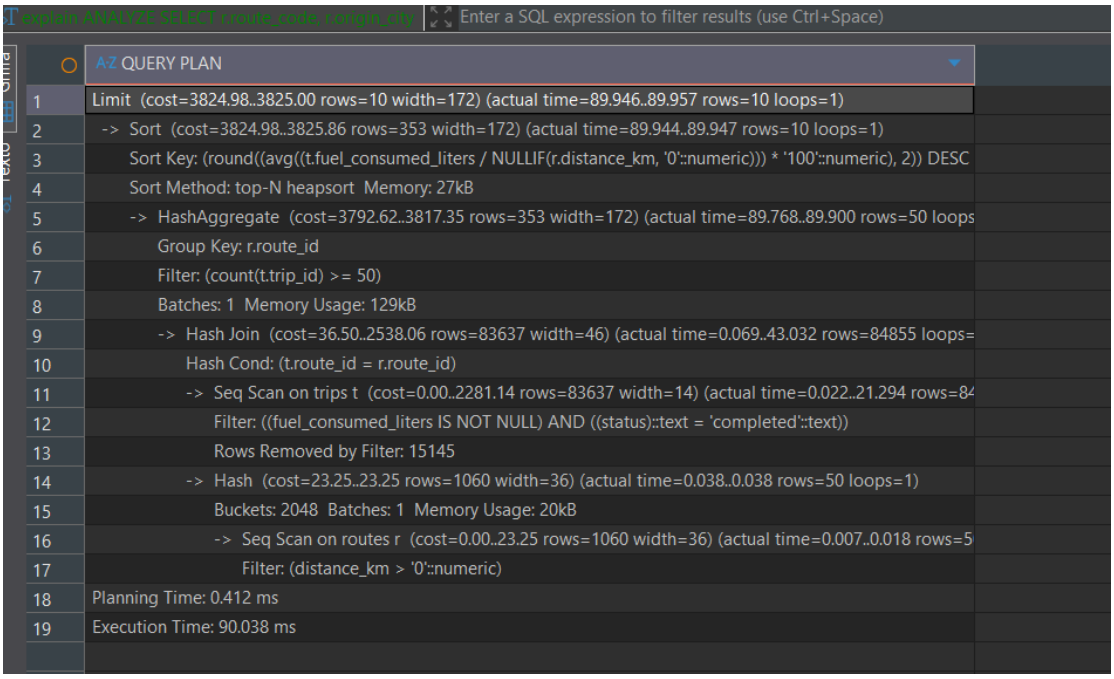


- Con indice 4

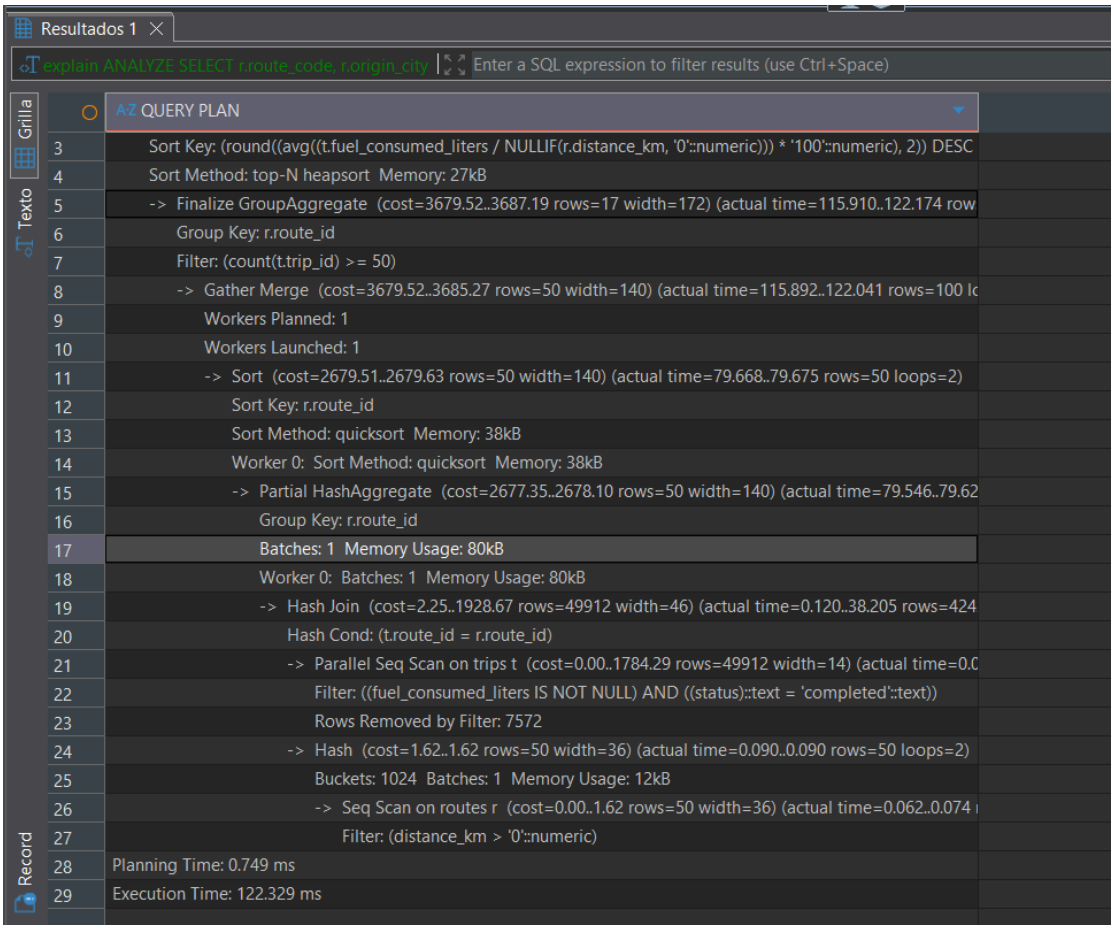


Query 7

- sin indice 5


explain ANALYZE SELECT r.route_code, r.origin_city
Enter a SQL expression to filter results (use Ctrl+Space)
QUERY PLAN
Limit (cost=3824.98..3825.00 rows=10 width=172) (actual time=89.946..89.957 rows=10 loops=1)
-> Sort (cost=3824.98..3825.86 rows=353 width=172) (actual time=89.944..89.947 rows=10 loops=1)
Sort Key: (round((avg((t.fuel_consumed_liters / NULLIF(r.distance_km, '0'::numeric))) * '100'::numeric), 2)) DESC
Sort Method: top-N heapsort Memory: 27kB
-> HashAggregate (cost=3792.62..3817.35 rows=353 width=172) (actual time=89.768..89.900 rows=50 loops
Group Key: r.route_id
Filter: (count(t.trip_id) >= 50)
Batches: 1 Memory Usage: 129kB
-> Hash Join (cost=36.50..2538.06 rows=83637 width=46) (actual time=0.069..43.032 rows=84855 loops=
Hash Cond: (t.route_id = r.route_id)
-> Seq Scan on trips t (cost=0.00..2281.14 rows=83637 width=14) (actual time=0.022..21.294 rows=84
Filter: ((fuel_consumed_liters IS NOT NULL) AND ((status)::text = 'completed'::text))
Rows Removed by Filter: 15145
-> Hash (cost=23.25..23.25 rows=1060 width=36) (actual time=0.038..0.038 rows=50 loops=1)
Buckets: 2048 Batches: 1 Memory Usage: 20kB
-> Seq Scan on routes r (cost=0.00..23.25 rows=1060 width=36) (actual time=0.007..0.018 rows=5
Filter: (distance_km > '0'::numeric)
Planning Time: 0.412 ms
Execution Time: 90.038 ms

- Con Indice 5—


Resultados 1
explain ANALYZE SELECT r.route_code, r.origin_city
Enter a SQL expression to filter results (use Ctrl+Space)
Grilla
Texto
Record
QUERY PLAN
Sort Key: (round((avg((t.fuel_consumed_liters / NULLIF(r.distance_km, '0'::numeric))) * '100'::numeric), 2)) DESC
Sort Method: top-N heapsort Memory: 27kB
-> Finalize GroupAggregate (cost=3679.52..3687.19 rows=17 width=172) (actual time=115.910..122.174 row
Group Key: r.route_id
Filter: (count(t.trip_id) >= 50)
-> Gather Merge (cost=3679.52..3685.27 rows=50 width=140) (actual time=115.892..122.041 rows=100 lo
Workers Planned: 1
Workers Launched: 1
-> Sort (cost=2679.51..2679.63 rows=50 width=140) (actual time=79.668..79.675 rows=50 loops=2)
Sort Key: r.route_id
Sort Method: quicksort Memory: 38kB
Worker 0: Sort Method: quicksort Memory: 38kB
-> Partial HashAggregate (cost=2677.35..2678.10 rows=50 width=140) (actual time=79.546..79.62
Group Key: r.route_id
Batches: 1 Memory Usage: 80kB
Worker 0: Batches: 1 Memory Usage: 80kB
-> Hash Join (cost=2.25..1928.67 rows=49912 width=46) (actual time=0.120..38.205 rows=424
Hash Cond: (t.route_id = r.route_id)
-> Parallel Seq Scan on trips t (cost=0.00..1784.29 rows=49912 width=14) (actual time=0.0
Filter: ((fuel_consumed_liters IS NOT NULL) AND ((status)::text = 'completed'::text))
Rows Removed by Filter: 7572
-> Hash (cost=1.62..1.62 rows=50 width=36) (actual time=0.090..0.090 rows=50 loops=2)
Buckets: 1024 Batches: 1 Memory Usage: 12kB
-> Seq Scan on routes r (cost=0.00..1.62 rows=50 width=36) (actual time=0.062..0.074
Filter: (distance_km > '0'::numeric)
Planning Time: 0.749 ms
Execution Time: 122.329 ms

Aclaración: en algunos casos no se nota una diferencia significativa en cuanto al tiempo y/o proceso de ejecución porque la base es relativamente pequeña en términos de información almacenada.