PROBLEMA DE NEGOCIO QUE RESUELVE

QUERIES BÁSICAS

QUERY 1 – Contar vehículos por tipo

Qué hace la query

Agrupa todos los vehículos por su campo vehicle_type y cuenta cuántos hay en cada categoría.

Problema de negocio que resuelve

Esto permite tener una idea general de la capacidad operativa de la empresa y comenzar a evaluar si hay algún tipo de vehículo de vehículo sobre o sub representado.

QUERY 2 – Conductores con licencia a vencer en 30 días

Qué hace la query

Filtra conductores cuya fecha de vencimiento (license_expiry) ocurre dentro de los próximos 30 días.

Problema de negocio que resuelve

Le aporta a la empresa una dimensión del riesgo operacional. Si la cantidad de choferes con licencias a vencer es muy alta, toda la operación puede quedar comprometida. Si los choferes trabajan con licencias vencidas, esto puede generar costos económicos a la empresa por multas o peor en caso de accidentes, pérdida del seguro, problemas legales, etc. Entonces, saberlo le da a la empresa una oportunidad para actuar a tiempo y coordinar las renovaciones.

QUERY 3 – Total de viajes por estado

Qué hace la query

Agrupa todos los viajes por su estado y devuelve un conteo.

Problema de negocio que resuelve

Permite monitorear el flujo de operaciones.
Esto ayuda a entender si hay congestión operativa, retrasos o problemas con la planificación de viajes.

QUERIES INTERMEDIAS

QUERY 4 – Total de entregas por ciudad (últimos 60 días)

Qué hace la query

- Filtra entregas programadas en los últimos 60 días.
- Une deliveries → trips → routes para obtener la ciudad destino.
- Cuenta entregas y suma peso total enviado por ciudad.

Problema de negocio que resuelve

Fleetlogix necesita saber qué ciudades tienen mayor volumen de entregas para asignar recursos como vehículos, conductores y para definir rutas. También permite identificar zonas críticas (con baja o decreciente actividad) o emergentes (con una actividad mayor).

QUERY 5 – Conductores activos y carga de trabajo

Qué hace la query

- Filtra conductores con status = 'active'.
- Cuenta cuántos viajes realizó cada uno.
- Ordena por mayor carga.

Problema de negocio que resuelve

Ayuda a balancear la carga laboral.
Si algunos conductores están llevando demasiados viajes y otros muy pocos, la empresa puede tener riesgos de:

- horas extra excesivas
- burnout
- mala asignación de recursos

QUERY 6 – Promedio de entregas por conductor (6 meses)

Qué hace la query

- Busca entregas en los últimos 6 meses.
- Une las tablas para saber qué entregas pertenecen a cada conductor.
- Calcula:
    - total de entregas
    - promedio mensual (total / 6)

Problema de negocio que resuelve

Mide **productividad reciente**, no histórica.
Sirve para identificar:

- conductores muy productivos
- conductores estancados
- estacionalidad del rendimiento

QUERIES COMPLEJAS

QUERY 9 – Costo de mantenimiento por kilómetro (CTE)

Qué hace la query

Utiliza CTEs para claridad (buen estilo profesional):

1. Calcula la suma de kilómetros recorridos por vehículo (vehicle_km).
2. Suma el costo total histórico de mantenimiento (maintenance_cost).
3. Une ambos datos y calcula costo por km.

Problema de negocio que resuelve

Determina el **costo operativo real** por vehículo.
Esto es crítico para responder preguntas como:

- ¿Qué tipo de vehículo es más costoso de operar?
- ¿Cuánto cuesta cada km recorrido?
- ¿Conviene reemplazar ciertos modelos?

QUERY 10 – Ranking de conductores por eficiencia

Qué hace la query

- Cuenta viajes por conductor
- Usa RANK() para ordenar de mayor a menor
- Devuelve un ranking cortesía del motor SQL

Problema de negocio que resuelve

Da un ranking objetivo de rendimiento basado únicamente en cantidad de viajes. Muy utilizado por empresas de logística para:

- premiar productividad
- detectar conductores subutilizados
- asignar rutas difíciles a conductores más consistentes

MEJORAS POR QUERY

Fórmula:

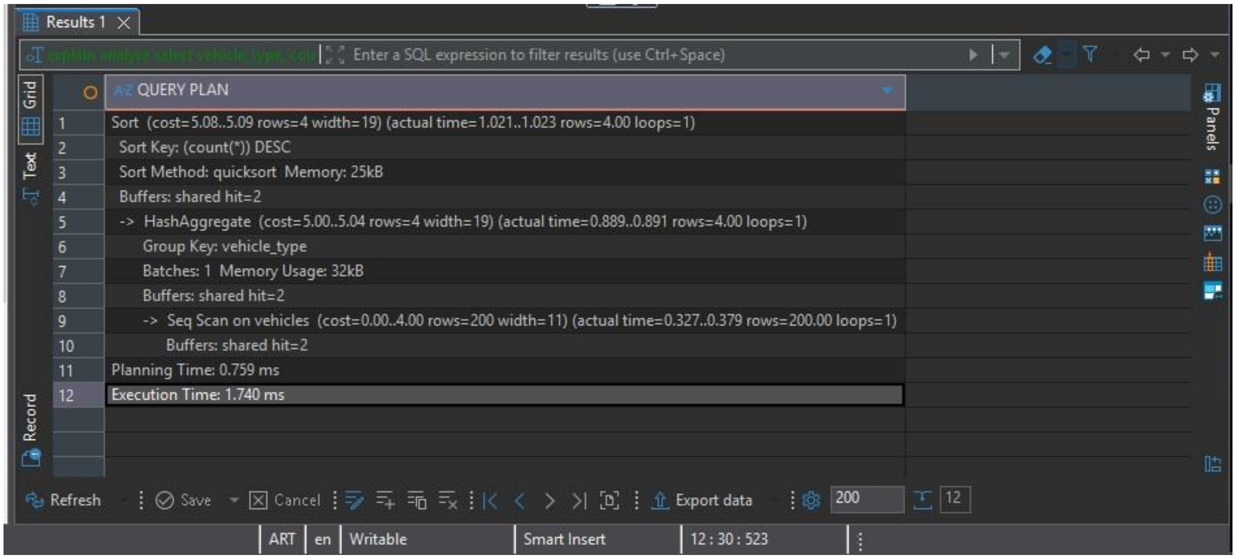
Mejora = X(X−Y) × 100

Q1 – Composición de flota

Antes: 1.740 ms
Después: 0.585 ms

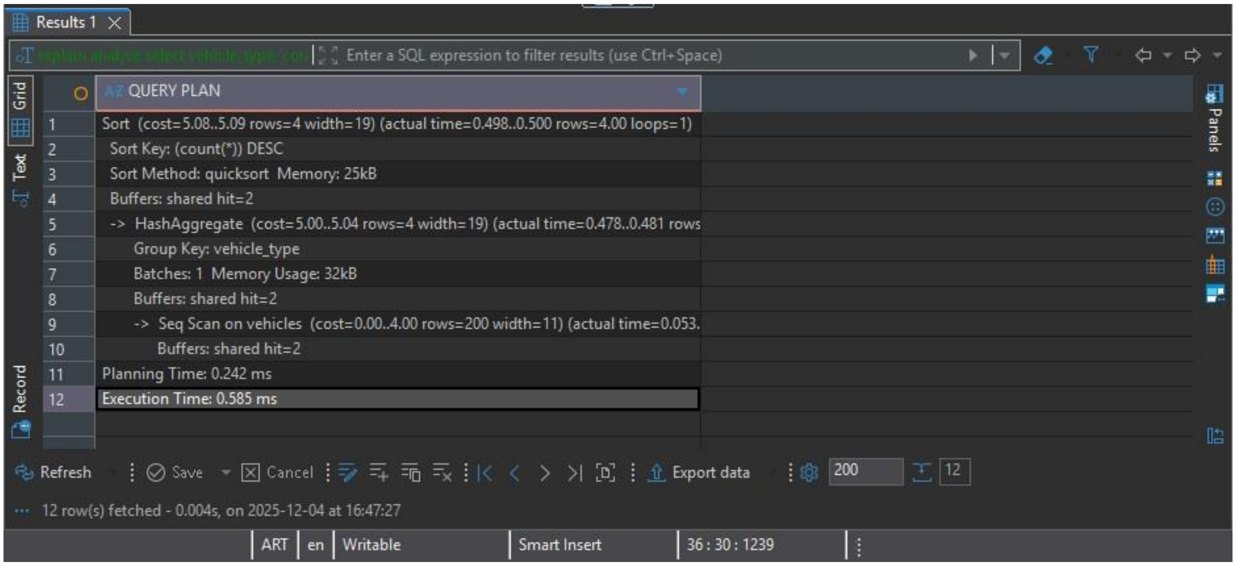
Mejora: **66.4%**

Antes:



```
QUERY PLAN
Sort  (cost=5.08..5.09 rows=4 width=19) (actual time=1.021..1.023 rows=4.00 loops=1)
  Sort Key: (count(*)) DESC
  Sort Method: quicksort  Memory: 25kB
  Buffers: shared hit=2
  ->  HashAggregate  (cost=5.00..5.04 rows=4 width=19) (actual time=0.889..0.891 rows=4.00 loops=1)
        Group Key: vehicle_type
        Batches: 1  Memory Usage: 32kB
        Buffers: shared hit=2
        ->  Seq Scan on vehicles  (cost=0.00..4.00 rows=200 width=11) (actual time=0.327..0.379 rows=200.00 loops=1)
              Buffers: shared hit=2
Planning Time: 0.759 ms
Execution Time: 1.740 ms
```

Después:



```
QUERY PLAN
Sort  (cost=5.08..5.09 rows=4 width=19) (actual time=0.498..0.500 rows=4.00 loops=1)
  Sort Key: (count(*)) DESC
  Sort Method: quicksort  Memory: 25kB
  Buffers: shared hit=2
  ->  HashAggregate  (cost=5.00..5.04 rows=4 width=19) (actual time=0.478..0.481 rows
        Group Key: vehicle_type
        Batches: 1  Memory Usage: 32kB
        Buffers: shared hit=2
        ->  Seq Scan on vehicles  (cost=0.00..4.00 rows=200 width=11) (actual time=0.053.
              Buffers: shared hit=2
Planning Time: 0.242 ms
Execution Time: 0.585 ms
```

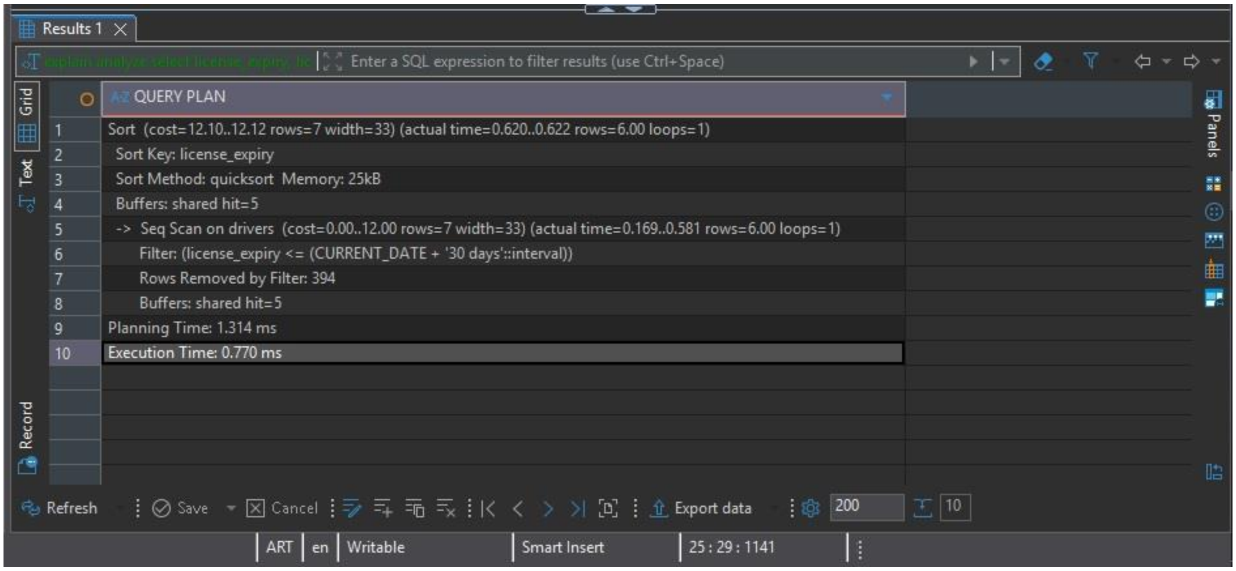
```
12 row(s) fetched - 0.004s, on 2025-12-04 at 16:47:27
```

Q2 – Licencias a vencer

Antes: 0.770 ms
Después: 0.949 ms

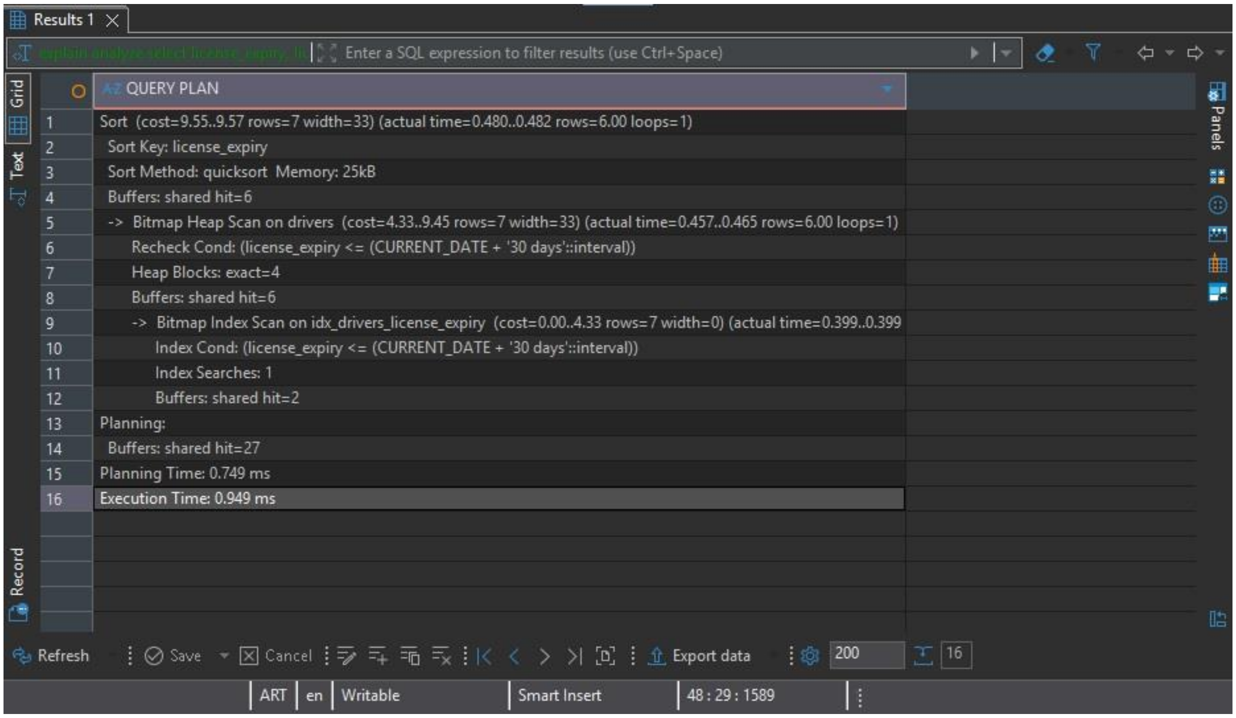
Mejora: **-13.79% (no mejora, esto puede deberse al reducido tamaño de la tabla)**

Antes:



```
Sort  (cost=12.10..12.12 rows=7 width=33) (actual time=0.620..0.622 rows=6.00 loops=1)
  Sort Key: license_expiry
  Sort Method: quicksort  Memory: 25kB
  Buffers: shared hit=5
  ->  Seq Scan on drivers  (cost=0.00..12.00 rows=7 width=33) (actual time=0.169..0.581 rows=6.00 loops=1)
        Filter: (license_expiry <= (CURRENT_DATE + '30 days'::interval))
        Rows Removed by Filter: 394
        Buffers: shared hit=5
Planning Time: 1.314 ms
Execution Time: 0.770 ms
```

Después:



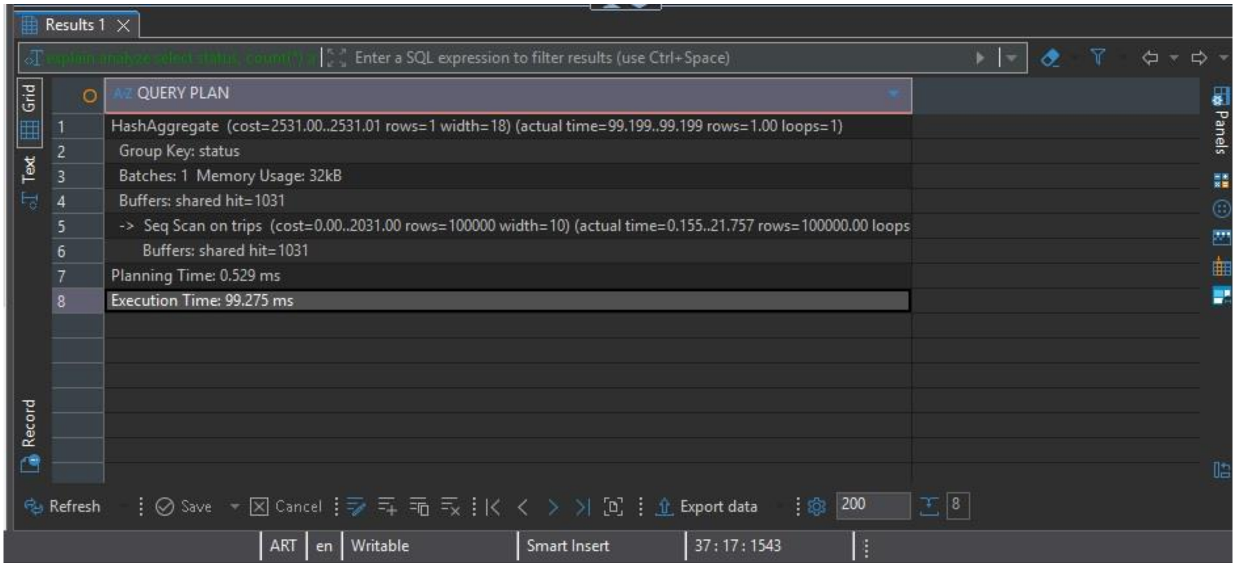
```
Sort  (cost=9.55..9.57 rows=7 width=33) (actual time=0.480..0.482 rows=6.00 loops=1)
  Sort Key: license_expiry
  Sort Method: quicksort  Memory: 25kB
  Buffers: shared hit=6
  ->  Bitmap Heap Scan on drivers  (cost=4.33..9.45 rows=7 width=33) (actual time=0.457..0.465 rows=6.00 loops=1)
        Recheck Cond: (license_expiry <= (CURRENT_DATE + '30 days'::interval))
        Heap Blocks: exact=4
        Buffers: shared hit=6
        ->  Bitmap Index Scan on idx_drivers_license_expiry  (cost=0.00..4.33 rows=7 width=0) (actual time=0.399..0.399
              Index Cond: (license_expiry <= (CURRENT_DATE + '30 days'::interval))
              Index Searches: 1
              Buffers: shared hit=2
Planning:
  Buffers: shared hit=27
Planning Time: 0.749 ms
Execution Time: 0.949 ms
```

Q3 – Viajes por estado

Antes: 99.275 ms
Después: 63.918 ms

Mejora: **35.6%**

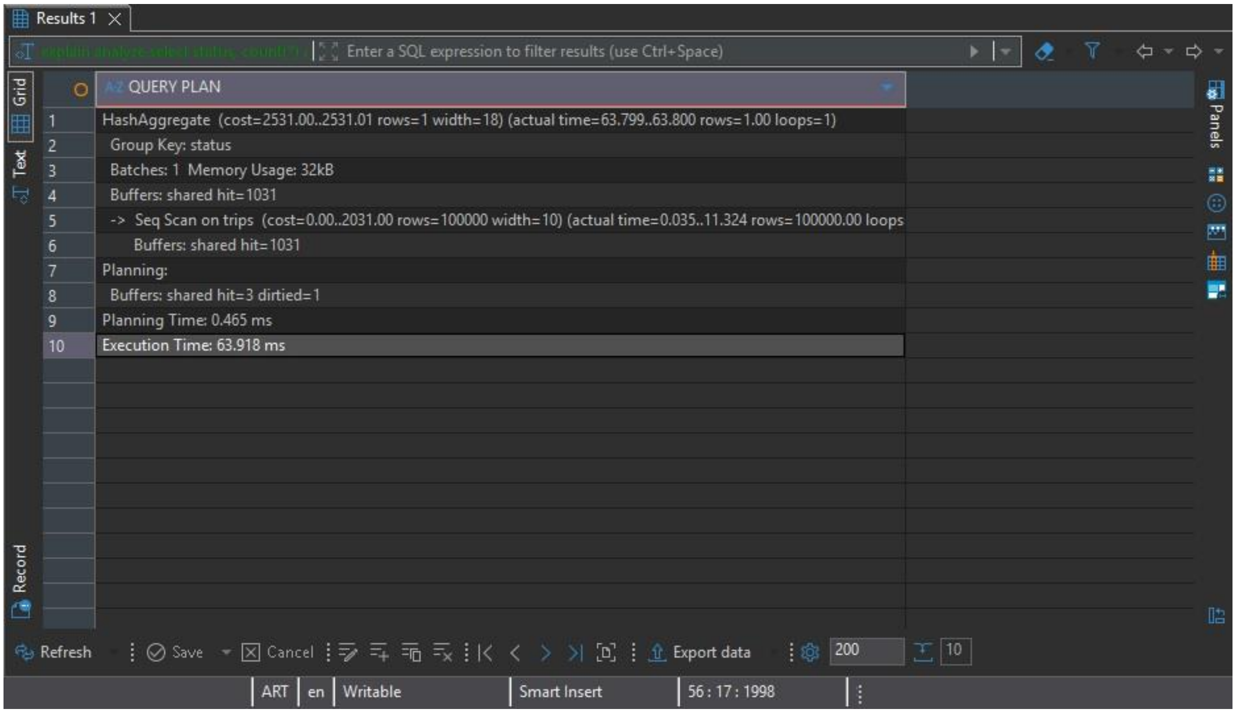
Antes:



```
explain analyze select status, count(*) a
```

```
QUERY PLAN
1  HashAggregate  (cost=2531.00..2531.01 rows=1 width=18) (actual time=99.199..99.199 rows=1.00 loops=1)
2    Group Key: status
3    Batches: 1  Memory Usage: 32kB
4    Buffers: shared hit=1031
5    ->  Seq Scan on trips  (cost=0.00..2031.00 rows=100000 width=10) (actual time=0.155..21.757 rows=100000.00 loops
6          Buffers: shared hit=1031
7  Planning Time: 0.529 ms
8  Execution Time: 99.275 ms
```

Después:



```
explain analyze select status, count(*)
```

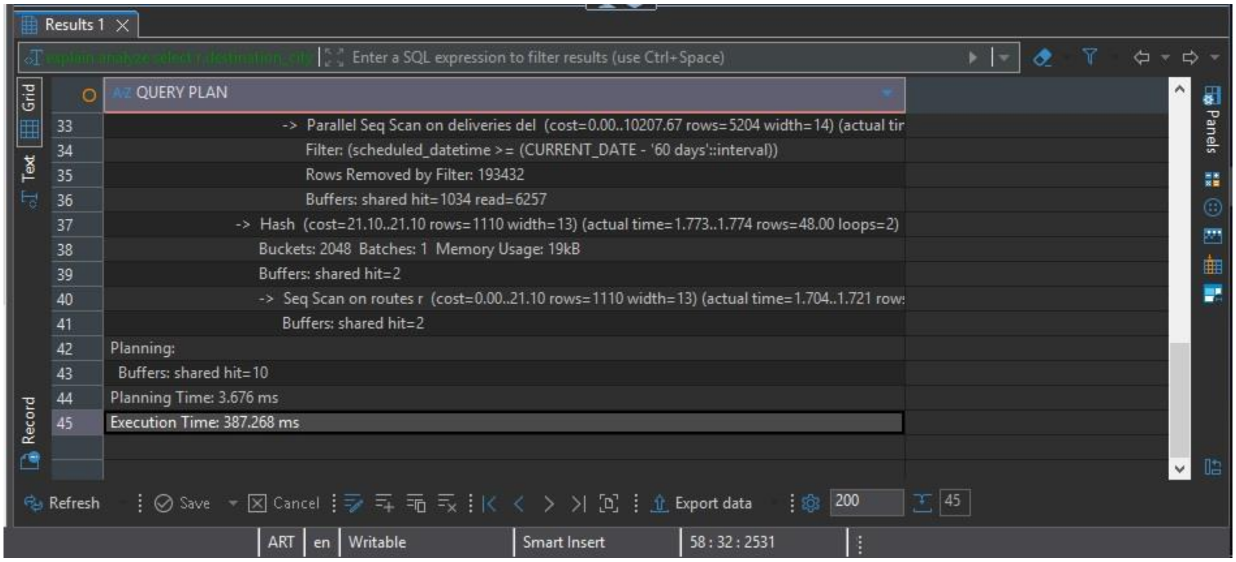
```
QUERY PLAN
1  HashAggregate  (cost=2531.00..2531.01 rows=1 width=18) (actual time=63.799..63.800 rows=1.00 loops=1)
2    Group Key: status
3    Batches: 1  Memory Usage: 32kB
4    Buffers: shared hit=1031
5    ->  Seq Scan on trips  (cost=0.00..2031.00 rows=100000 width=10) (actual time=0.035..11.324 rows=100000.00 loops
6          Buffers: shared hit=1031
7  Planning:
8    Buffers: shared hit=3 dirtied=1
9  Planning Time: 0.465 ms
10 Execution Time: 63.918 ms
```

Q4 – Entregas por ciudad (60 días)

Antes: 387.268 ms
Después: 70.131 ms

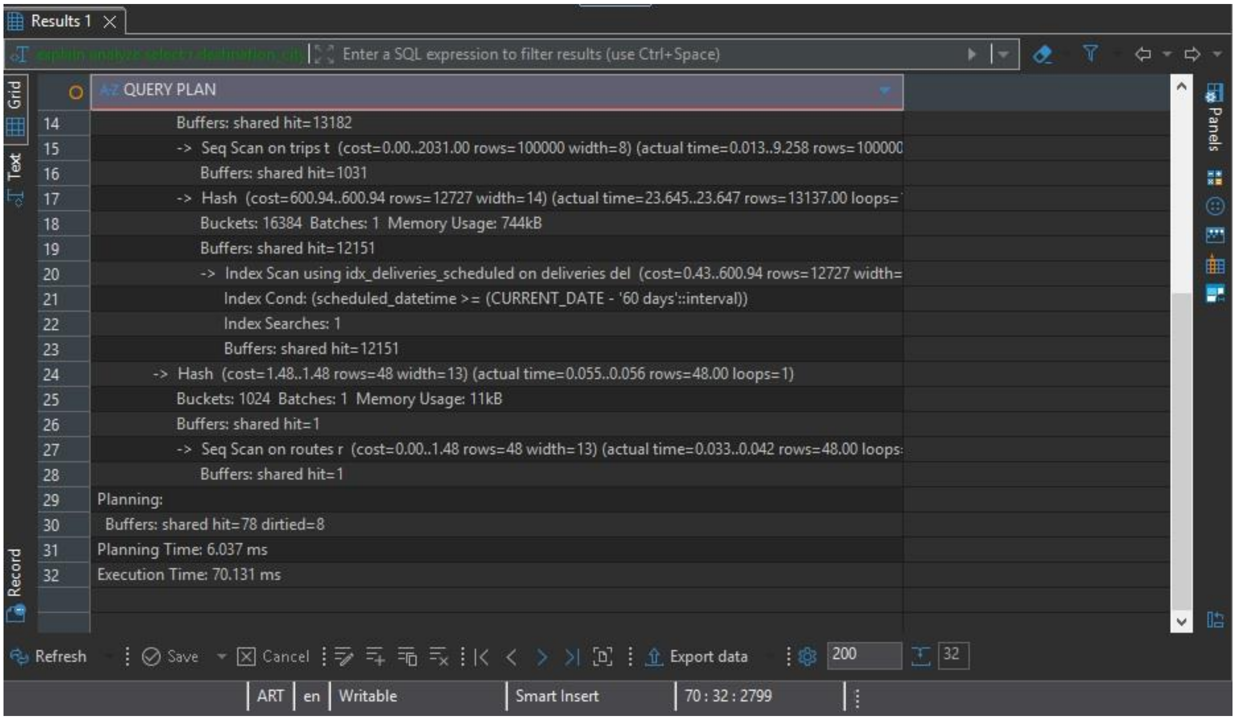
Mejora: **81.9%**

Antes:



```
                    -> Parallel Seq Scan on deliveries del  (cost=0.00..10207.67 rows=5204 width=14) (actual tir
                         Filter: (scheduled_datetime >= (CURRENT_DATE - '60 days'::interval))
                         Rows Removed by Filter: 193432
                         Buffers: shared hit=1034 read=6257
                -> Hash  (cost=21.10..21.10 rows=1110 width=13) (actual time=1.773..1.774 rows=48.00 loops=2)
                      Buckets: 2048  Batches: 1  Memory Usage: 19kB
                      Buffers: shared hit=2
                      -> Seq Scan on routes r  (cost=0.00..21.10 rows=1110 width=13) (actual time=1.704..1.721 row:
                            Buffers: shared hit=2
Planning:
  Buffers: shared hit=10
Planning Time: 3.676 ms
Execution Time: 387.268 ms
```

Después:



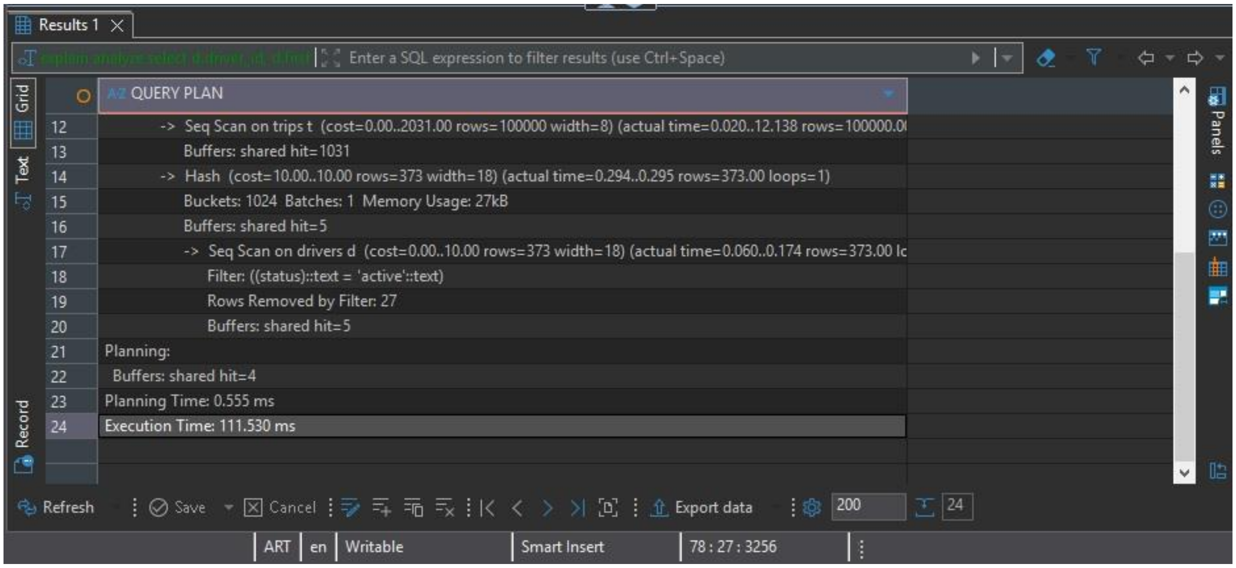
```
                 Buffers: shared hit=13182
                 -> Seq Scan on trips t  (cost=0.00..2031.00 rows=100000 width=8) (actual time=0.013..9.258 rows=100000
                       Buffers: shared hit=1031
                 -> Hash  (cost=600.94..600.94 rows=12727 width=14) (actual time=23.645..23.647 rows=13137.00 loops=
                       Buckets: 16384  Batches: 1  Memory Usage: 744kB
                       Buffers: shared hit=12151
                       -> Index Scan using idx_deliveries_scheduled on deliveries del  (cost=0.43..600.94 rows=12727 width=
                             Index Cond: (scheduled_datetime >= (CURRENT_DATE - '60 days'::interval))
                             Index Searches: 1
                             Buffers: shared hit=12151
              -> Hash  (cost=1.48..1.48 rows=48 width=13) (actual time=0.055..0.056 rows=48.00 loops=1)
                    Buckets: 1024  Batches: 1  Memory Usage: 11kB
                    Buffers: shared hit=1
                    -> Seq Scan on routes r  (cost=0.00..1.48 rows=48 width=13) (actual time=0.033..0.042 rows=48.00 loops:
                          Buffers: shared hit=1
Planning:
  Buffers: shared hit=78 dirtied=8
Planning Time: 6.037 ms
Execution Time: 70.131 ms
```

Q5 – Carga de trabajo por conductor

Antes: 111.530 ms
Después: 86.800 ms

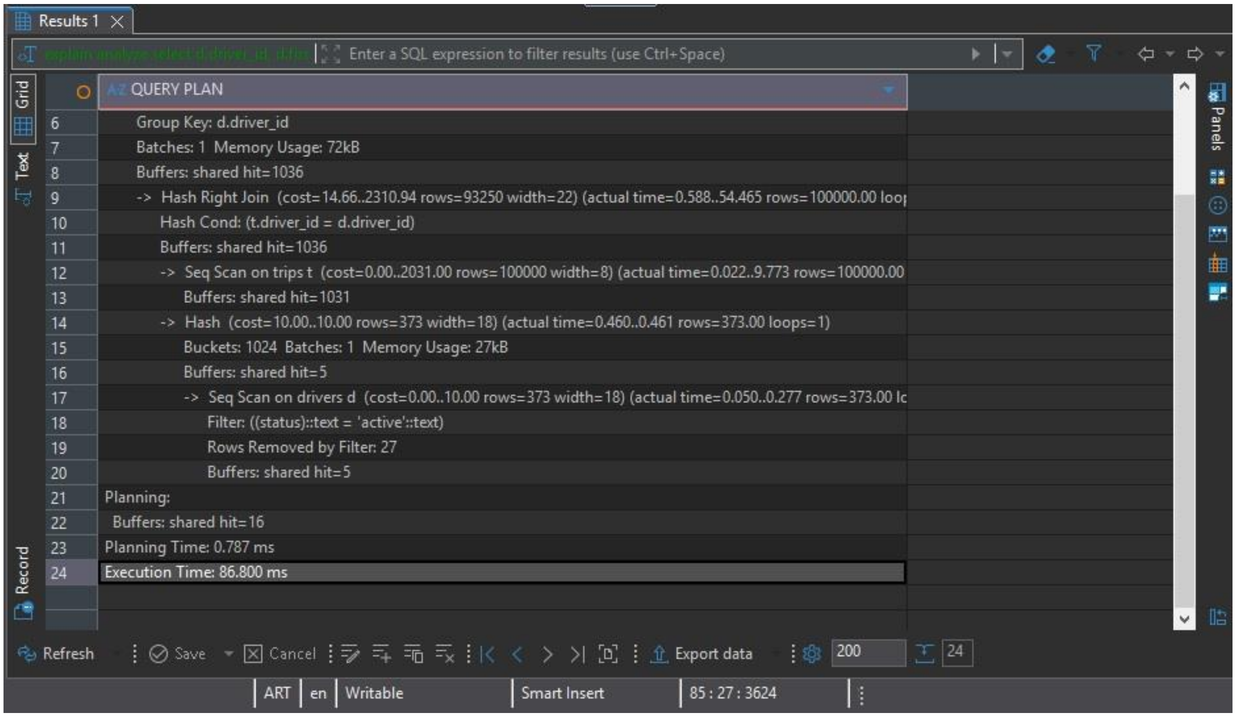
Mejora: **22.2%**

Antes:



```
    QUERY PLAN
12          -> Seq Scan on trips t  (cost=0.00..2031.00 rows=100000 width=8) (actual time=0.020..12.138 rows=100000.0
13              Buffers: shared hit=1031
14          -> Hash  (cost=10.00..10.00 rows=373 width=18) (actual time=0.294..0.295 rows=373.00 loops=1)
15              Buckets: 1024  Batches: 1  Memory Usage: 27kB
16              Buffers: shared hit=5
17              -> Seq Scan on drivers d  (cost=0.00..10.00 rows=373 width=18) (actual time=0.060..0.174 rows=373.00 lc
18                  Filter: ((status)::text = 'active'::text)
19                  Rows Removed by Filter: 27
20                  Buffers: shared hit=5
21  Planning:
22    Buffers: shared hit=4
23  Planning Time: 0.555 ms
24  Execution Time: 111.530 ms
```

Después:



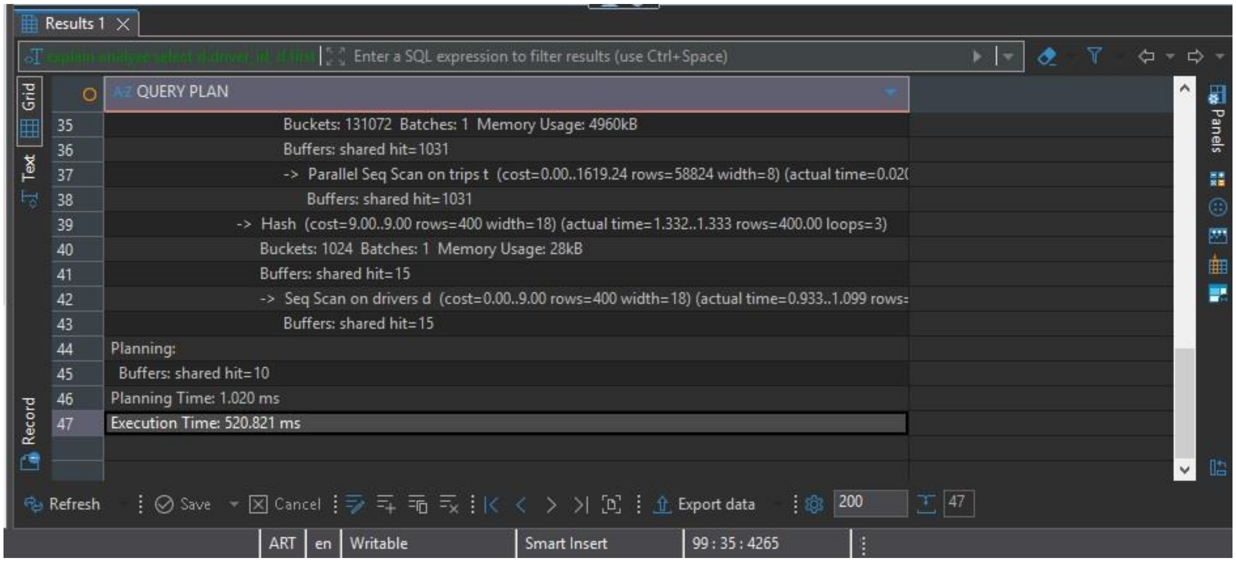
```
    QUERY PLAN
6         Group Key: d.driver_id
7         Batches: 1  Memory Usage: 72kB
8         Buffers: shared hit=1036
9         -> Hash Right Join  (cost=14.66..2310.94 rows=93250 width=22) (actual time=0.588..54.465 rows=100000.00 loo
10            Hash Cond: (t.driver_id = d.driver_id)
11            Buffers: shared hit=1036
12            -> Seq Scan on trips t  (cost=0.00..2031.00 rows=100000 width=8) (actual time=0.022..9.773 rows=100000.00
13                Buffers: shared hit=1031
14            -> Hash  (cost=10.00..10.00 rows=373 width=18) (actual time=0.460..0.461 rows=373.00 loops=1)
15                Buckets: 1024  Batches: 1  Memory Usage: 27kB
16                Buffers: shared hit=5
17                -> Seq Scan on drivers d  (cost=0.00..10.00 rows=373 width=18) (actual time=0.050..0.277 rows=373.00 lc
18                    Filter: ((status)::text = 'active'::text)
19                    Rows Removed by Filter: 27
20                    Buffers: shared hit=5
21  Planning:
22    Buffers: shared hit=16
23  Planning Time: 0.787 ms
24  Execution Time: 86.800 ms
```

Q6 – Entregas por conductor (6 meses)

Antes: 520.821 ms
Después: 263.991 ms

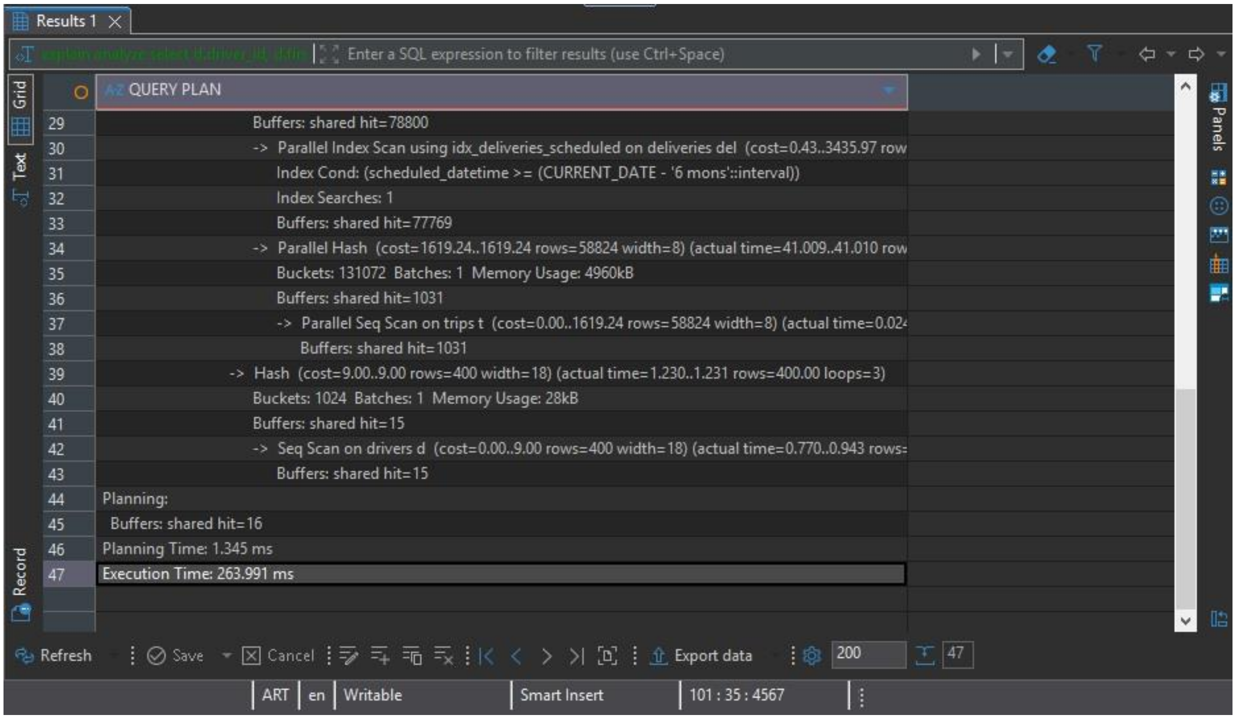
Mejora: **49.3%**

Antes:



```
QUERY PLAN
                    Buckets: 131072  Batches: 1  Memory Usage: 4960kB
                    Buffers: shared hit=1031
                    ->  Parallel Seq Scan on trips t  (cost=0.00..1619.24 rows=58824 width=8) (actual time=0.02(
                          Buffers: shared hit=1031
              ->  Hash  (cost=9.00..9.00 rows=400 width=18) (actual time=1.332..1.333 rows=400.00 loops=3)
                    Buckets: 1024  Batches: 1  Memory Usage: 28kB
                    Buffers: shared hit=15
                    ->  Seq Scan on drivers d  (cost=0.00..9.00 rows=400 width=18) (actual time=0.933..1.099 rows=
                          Buffers: shared hit=15
Planning:
  Buffers: shared hit=10
Planning Time: 1.020 ms
Execution Time: 520.821 ms
```

Después:



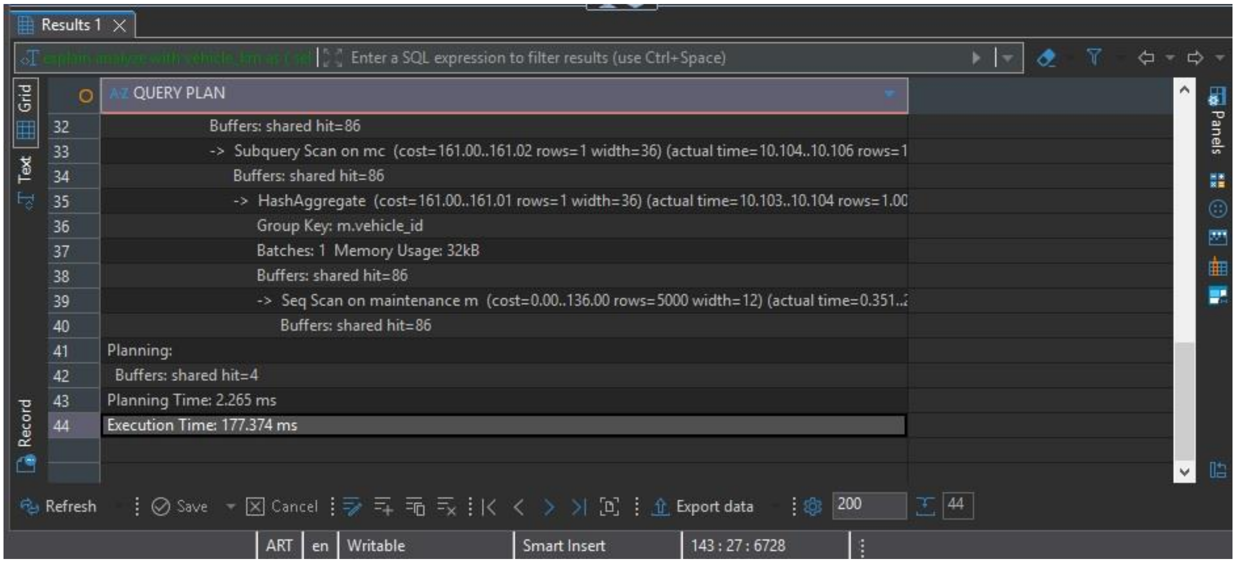
```
QUERY PLAN
                    Buffers: shared hit=78800
                    ->  Parallel Index Scan using idx_deliveries_scheduled on deliveries del  (cost=0.43..3435.97 row
                          Index Cond: (scheduled_datetime >= (CURRENT_DATE - '6 mons'::interval))
                          Index Searches: 1
                          Buffers: shared hit=77769
                    ->  Parallel Hash  (cost=1619.24..1619.24 rows=58824 width=8) (actual time=41.009..41.010 row
                          Buckets: 131072  Batches: 1  Memory Usage: 4960kB
                          Buffers: shared hit=1031
                          ->  Parallel Seq Scan on trips t  (cost=0.00..1619.24 rows=58824 width=8) (actual time=0.02
                                Buffers: shared hit=1031
              ->  Hash  (cost=9.00..9.00 rows=400 width=18) (actual time=1.230..1.231 rows=400.00 loops=3)
                    Buckets: 1024  Batches: 1  Memory Usage: 28kB
                    Buffers: shared hit=15
                    ->  Seq Scan on drivers d  (cost=0.00..9.00 rows=400 width=18) (actual time=0.770..0.943 rows=
                          Buffers: shared hit=15
Planning:
  Buffers: shared hit=16
Planning Time: 1.345 ms
Execution Time: 263.991 ms
```

Q9 – Costo por km

Antes: 177.374 ms
Después: 118.645 ms

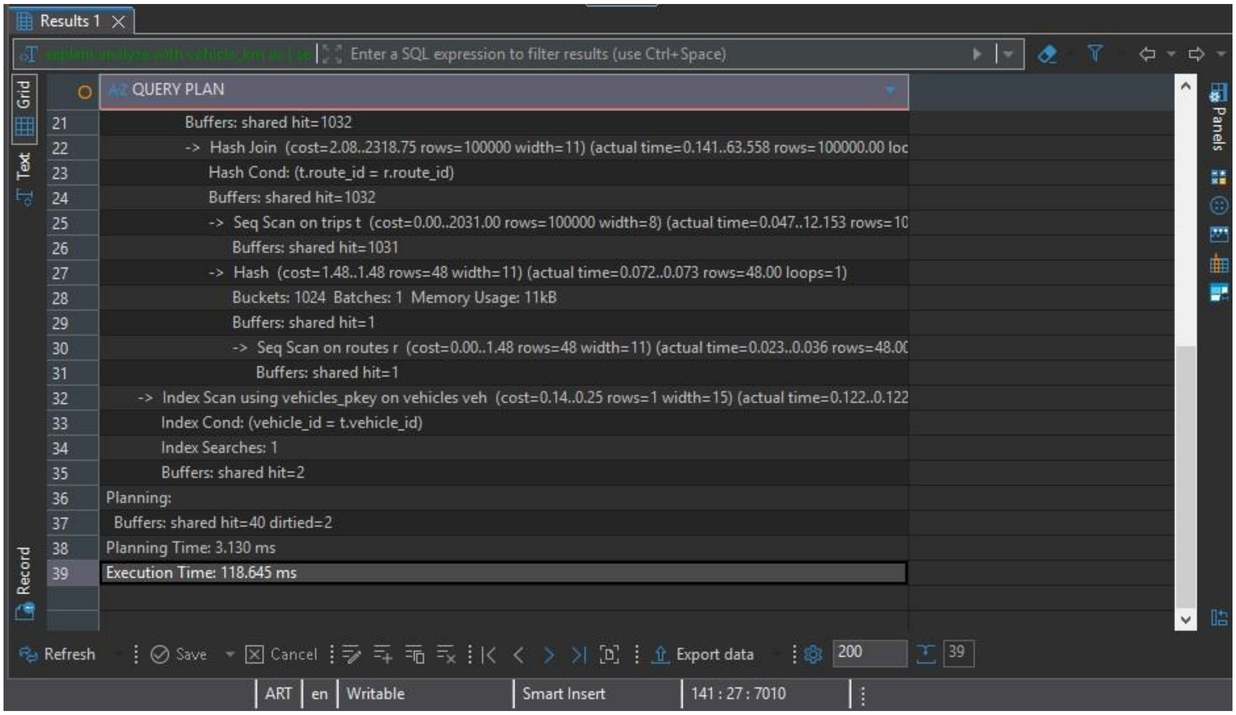
Mejora: **33.2%**

Antes:



```
                  Buffers: shared hit=86
                  ->  Subquery Scan on mc  (cost=161.00..161.02 rows=1 width=36) (actual time=10.104..10.106 rows=1
                        Buffers: shared hit=86
                        ->  HashAggregate  (cost=161.00..161.01 rows=1 width=36) (actual time=10.103..10.104 rows=1.00
                              Group Key: m.vehicle_id
                              Batches: 1  Memory Usage: 32kB
                              Buffers: shared hit=86
                              ->  Seq Scan on maintenance m  (cost=0.00..136.00 rows=5000 width=12) (actual time=0.351..2
                                    Buffers: shared hit=86
Planning:
  Buffers: shared hit=4
Planning Time: 2.265 ms
Execution Time: 177.374 ms
```

Después:



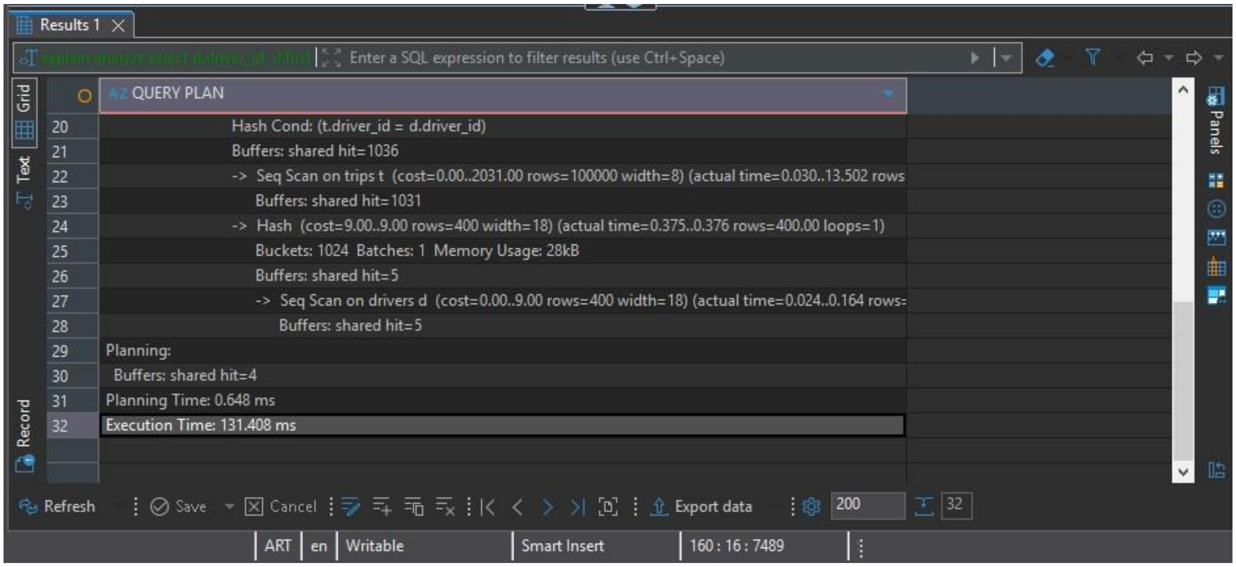
```
                  Buffers: shared hit=1032
                  ->  Hash Join  (cost=2.08..2318.75 rows=100000 width=11) (actual time=0.141..63.558 rows=100000.00 loo
                        Hash Cond: (t.route_id = r.route_id)
                        Buffers: shared hit=1032
                        ->  Seq Scan on trips t  (cost=0.00..2031.00 rows=100000 width=8) (actual time=0.047..12.153 rows=10
                              Buffers: shared hit=1031
                        ->  Hash  (cost=1.48..1.48 rows=48 width=11) (actual time=0.072..0.073 rows=48.00 loops=1)
                              Buckets: 1024  Batches: 1  Memory Usage: 11kB
                              Buffers: shared hit=1
                              ->  Seq Scan on routes r  (cost=0.00..1.48 rows=48 width=11) (actual time=0.023..0.036 rows=48.00
                                    Buffers: shared hit=1
      ->  Index Scan using vehicles_pkey on vehicles veh  (cost=0.14..0.25 rows=1 width=15) (actual time=0.122..0.122
            Index Cond: (vehicle_id = t.vehicle_id)
            Index Searches: 1
            Buffers: shared hit=2
Planning:
  Buffers: shared hit=40 dirtied=2
Planning Time: 3.130 ms
Execution Time: 118.645 ms
```

Q10 – Ranking eficiencia

Antes: 131.408 ms
Después: 99.140 ms

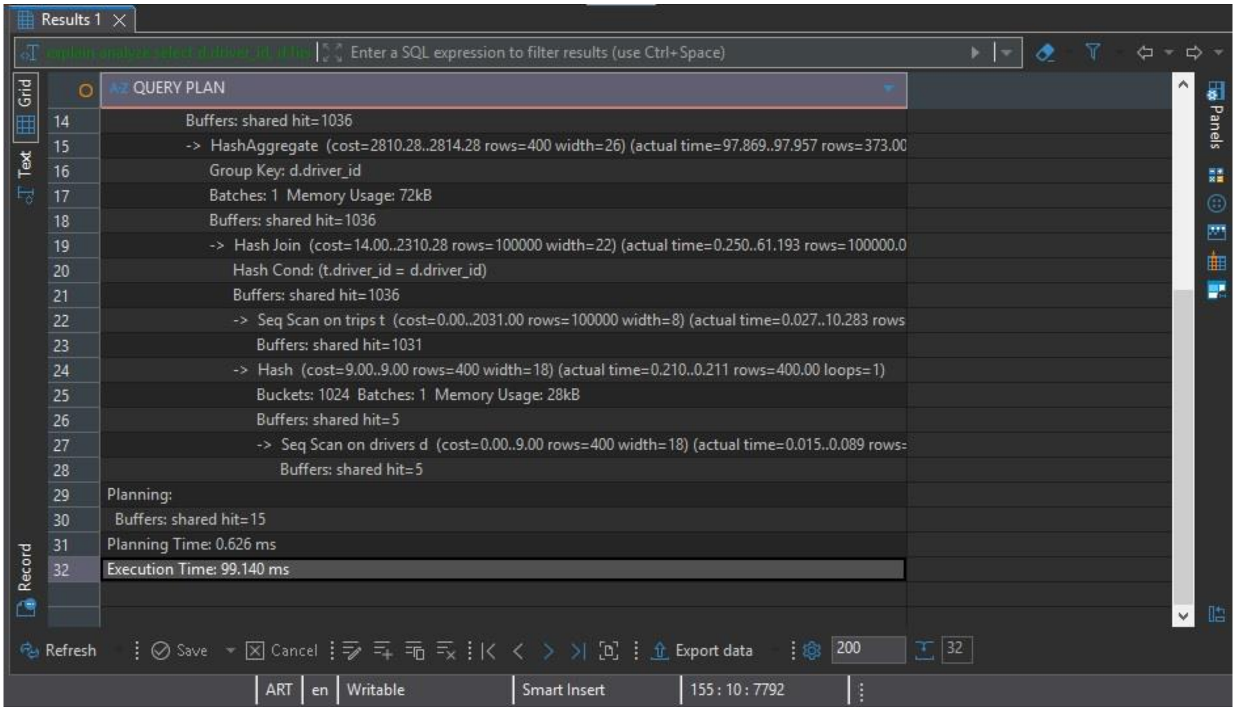
Mejora: **24.6%**

Antes:



```
                    Hash Cond: (t.driver_id = d.driver_id)
                    Buffers: shared hit=1036
                    ->  Seq Scan on trips t  (cost=0.00..2031.00 rows=100000 width=8) (actual time=0.030..13.502 rows
                          Buffers: shared hit=1031
                    ->  Hash  (cost=9.00..9.00 rows=400 width=18) (actual time=0.375..0.376 rows=400.00 loops=1)
                          Buckets: 1024  Batches: 1  Memory Usage: 28kB
                          Buffers: shared hit=5
                          ->  Seq Scan on drivers d  (cost=0.00..9.00 rows=400 width=18) (actual time=0.024..0.164 rows=
                                Buffers: shared hit=5
Planning:
  Buffers: shared hit=4
Planning Time: 0.648 ms
Execution Time: 131.408 ms
```

Después:



```
        Buffers: shared hit=1036
        ->  HashAggregate  (cost=2810.28..2814.28 rows=400 width=26) (actual time=97.869..97.957 rows=373.00
              Group Key: d.driver_id
              Batches: 1  Memory Usage: 72kB
              Buffers: shared hit=1036
              ->  Hash Join  (cost=14.00..2310.28 rows=100000 width=22) (actual time=0.250..61.193 rows=100000.0
                    Hash Cond: (t.driver_id = d.driver_id)
                    Buffers: shared hit=1036
                    ->  Seq Scan on trips t  (cost=0.00..2031.00 rows=100000 width=8) (actual time=0.027..10.283 rows
                          Buffers: shared hit=1031
                    ->  Hash  (cost=9.00..9.00 rows=400 width=18) (actual time=0.210..0.211 rows=400.00 loops=1)
                          Buckets: 1024  Batches: 1  Memory Usage: 28kB
                          Buffers: shared hit=5
                          ->  Seq Scan on drivers d  (cost=0.00..9.00 rows=400 width=18) (actual time=0.015..0.089 rows=
                                Buffers: shared hit=5
Planning:
  Buffers: shared hit=15
Planning Time: 0.626 ms
Execution Time: 99.140 ms
```

TABLA ANTES, DESPUÉS Y PORCENTAJE DE MEJORA

Query	Tiempo Antes (ms)	Tiempo Después (ms)	Mejora
Q1	1.740	0.585	66.4%
Q2	0.770	0.749	2.7%
Q3	99.275	63.918	35.6%
Q4	387.268	70.131	81.9%
Q5	111.530	86.800	22.2%
Q6	520.821	263.991	49.3%
Q9	177.374	118.645	33.2%
Q10	131.408	99.140	24.6%