

## Entregables – Avance 2

El estudiante deberá entregar:

1. El estudiante deberá realizar las 3 queries básicas, las queries 4 y 5, deberá elegir una entre las 6, 7 y 8. Deberá hacer la query 9, y elegir una entre las queries 10, 11 y 12. En total, debe entregar 8 queries resueltas en el archivo **02\_queries\_analysis.sql**
2. El estudiante deberá crear los 5 índices que se encuentran en el archivo **03\_optimization\_indexes.sql**, y comparar el tiempo de ejecución antes y después de crear los índices. Para hacerlo debe seguir los siguientes pasos:
  - a. Ejecutar cada query (con EXPLAIN ANALYZE) y guardar el tiempo de ejecución (Execution Time)

```
1 EXPLAIN ANALYZE
2 WITH vehicle_metrics AS (
3     SELECT
4         v.vehicle_id,
5         v.vehicle_type,
6         v.license_plate,
7         COUNT(DISTINCT t.trip_id) AS total_viajes,
8         SUM(r.distance_km) AS km_totales,
9         SUM(m.cost) AS costo_mantenimiento_total,
10        COUNT(DISTINCT m.maintenance_id) AS cantidad_mantenimientos
11    FROM vehicles v
12    LEFT JOIN trips t ON v.vehicle_id = t.vehicle_id
13    LEFT JOIN routes r ON t.route_id = r.route_id
14    LEFT JOIN maintenance m ON v.vehicle_id = m.vehicle_id
```

QUERY PLAN	Cost	Actual Time
Hash Left Join (cost=2.08..2368.73 rows=100000 width=15) (actual time=0.091..66.746 rows=100000)		
Hash Cond: (t.route_id = r.route_id)		
Seq Scan on trips t (cost=0.00..2281.00 rows=100000 width=12) (actual time=0.036..32.111)		
Filter: (status='completed')		
Seq Scan on routes r (cost=1.48..1.48 rows=48 width=11) (actual time=0.034..0.035 rows=48 loops=1)		
Seq Scan on maintenance m (cost=0.00..0.00 rows=48 width=11) (actual time=0.013..0.022 rows=48 loops=1)		
Planning Time: 1.241 ms		
Execution Time: 2921.743 ms		

Total rows: 35 Query complete 00:00:02.988

- b. Crear los 5 índices que están en el archivo **03\_optimization\_indexes.sql**
- c. Ejecutar:
  - ANALYZE vehicles;
  - ANALYZE drivers;
  - ANALYZE routes;
  - ANALYZE trips;
  - ANALYZE deliveries;
  - ANALYZE maintenance;
- d. Ejecutar de nuevo cada query (con EXPLAIN ANALYZE) y guardar el tiempo de ejecución (Execution Time)

```
16 GROUP BY v.vehicle_id, v.vehicle_type, v.license_plate
17 )
18 SELECT
19     vehicle_type,
20     COUNT(vehicle_id) AS cantidad_vehiculos,
21     SUM(total_viajes) AS viajes_totales,
22     SUM(km_totales) AS kilometros_totales,
23     SUM(costo_mantenimiento_total) AS costo_total_mantenimiento,
24     ROUND(SUM(costo_mantenimiento_total) / NULLIF(SUM(km_totales), 0), 2) AS costo,
25     ROUND(AVG(costo_mantenimiento_total / NULLIF(cantidad_mantenimientos, 0)), 2) AS
26 FROM vehicle_metrics
27 WHERE km_totales > 0 AND costo_mantenimiento_total > 0
28 GROUP BY vehicle_type
29 ORDER BY costo_por_km DESC;
```

QUERY PLAN	Cost	Actual Time
Hash Left Join (cost=2.08..2368.73 rows=100000 width=15) (actual time=0.091..66.746 rows=100000)		
Hash Cond: (t.route_id = r.route_id)		
Seq Scan on trips t (cost=0.00..2281.00 rows=100000 width=12) (actual time=0.036..32.111)		
Filter: (status='completed')		
Seq Scan on routes r (cost=1.48..1.48 rows=48 width=11) (actual time=0.034..0.035 rows=48 loops=1)		
Seq Scan on maintenance m (cost=0.00..0.00 rows=48 width=11) (actual time=0.013..0.022 rows=48 loops=1)		
Planning Time: 1.241 ms		
Execution Time: 2440.595 ms		

Total rows: 33 Query complete 00:00:02.511

- e. Compara:  
Tiempo antes: X ms  
Tiempo después: Y ms  
Mejora:  $((X-Y)/X) * 100\%$
  - f. Verifica en el plan que ahora use "Index Scan" en lugar de "Seq Scan"
3. En **Manual\_Consultas\_SQL.pdf** el estudiante deberá documentar el problema de negocio que resuelve cada una de los 8 queries que decidió resolver, el tiempo de ejecución de las queries antes y después de la creación de los índices (optimización), y explicar brevemente qué optimiza cada uno de los 5 índices creados

### Listado de queries

#### Queries Básicas

1. **Query 1: Contar vehículos por tipo**  
**Resultado esperado:** Lista simple con tipos de vehículo y cantidades.

---

2. **Query 2: Conductores con licencia próxima a vencer**  
**Resultado esperado:** Lista de conductores que deben renovar licencia en 30 días.

---

3. **Query 3: Total de viajes por estado**  
**Resultado esperado:** Conteo simple por estado (in\_progress, completed).

---

#### Queries Intermedias

4. **Query 4: Total de entregas por ciudad (últimos 2 meses, 60 días)**  
**Resultado esperado:** Ranking de ciudades con volumen de entregas y peso total.

---

5. **Query 5: Conductores activos y carga de trabajo**  
**Resultado esperado:** Lista con total de viajes por conductor activo.

---

6. **Query 6: Promedio de entregas por conductor (6 meses)**  
**Resultado esperado:** Métricas de productividad por conductor.

---

7. **Query 7: Rutas con mayor consumo de combustible**  
**Resultado esperado:** Top 10 rutas con mayor consumo litros/100km.

---

8. **Query 8: Análisis de retrasos por día de semana**  
**Resultado esperado:** Porcentaje de retrasos por cada día de la semana.

---

## Queries Complejas

### 9. Query 9: Costo de mantenimiento por kilómetro

**Resultado esperado:** Costo por km para cada tipo de vehículo usando CTEs.

---

### 10. Query 10: Ranking de conductores por eficiencia

**Resultado esperado:** Top 20 conductores con ranking múltiple usando Window Functions (RANK).

---

### 11. Query 11: Análisis de tendencia mensual

**Resultado esperado:** Tendencia mensual con comparaciones usando LAG/LEAD.

---

### 12. Query 12: Pivot de entregas por hora y día

**Resultado esperado:** Matriz hora x día mostrando volumen de entregas (PIVOT). Filtre las entregas (deliveries) cuya fecha programada (scheduled\_datetime) sea **dentro de los últimos 60 días**.