

Proyecto Integrador - Fleetlogix

Índice

1. *Introducción*
2. *Objetivos*
3. *Modelo Relacional (ERD)* 3.1 *Tablas maestras* 3.2 *Tablas transaccionales* 3.3 *Relaciones y restricciones*
4. *Generación de Datos Sintéticos* 4.1 *Herramientas utilizadas* 4.2 *Volumen de datos* 4.3 *Métodos clave* 4.4 *¿Por qué 400.001 y no 400.000?* 4.5 *Aspecto técnico*
5. *Validaciones de Calidad* 5.1 *Integridad referencial* 5.2 *Consistencia temporal* 5.3 *Reglas de negocio*
6. *KPIs Operativos* 6.1 *Métricas calculadas*
7. *Arquitectura Cloud Propuesta*
8. *Ejecución*
9. *Conclusiones*

1. Introducción

Fleetlogix es una empresa de transporte y logística que opera una flota de 200 vehículos realizando entregas de última milla en cinco ciudades principales de Colombia. El objetivo del proyecto es migrar de sistemas legacy hacia una infraestructura moderna de datos, basada en PostgreSQL, que permita análisis operativos y decisiones en tiempo real.

El proyecto integra modelado relacional, generación de datos sintéticos masivos, validaciones de calidad, KPIs logísticos y una arquitectura cloud escalable. La solución se diseñó siguiendo principios de reproducibilidad, integridad referencial y escalabilidad.

2. Objetivos técnicos

- ✚ Poblar una base de datos PostgreSQL con más de 500.000 registros sintéticos generados mediante librerías de Python.
- ✚ Garantizar integridad referencial mediante claves primarias y foráneas, restricciones de unicidad y validaciones de negocio.
- ✚ Documentar el modelo relacional con un diagrama ERD profesional y un modelo dimensional tipo estrella para OLAP.
- ✚ Implementar queries SQL para validaciones de calidad y KPIs operativos.
- ✚ Proponer una arquitectura cloud con Kafka, Flink/Spark, Data Lake y Power BI para análisis en tiempo real.

3. Modelo Relacional (ERD)

El modelo relacional se compone de seis tablas:

- **3.1. Maestras:**
 - vehicles: atributos técnicos (placa, tipo, capacidad, estado).
 - drivers: información personal y licencias.
 - routes: rutas entre ciudades con distancia y duración estimada.

• 3.2. Transaccionales:

- trips: viajes realizados, vinculando vehículo, conductor y ruta.
- deliveries: entregas asociadas a cada viaje, con tracking único.
- maintenance: registros de mantenimiento por vehículo.

• 3.3 Restricciones técnicas:

- Claves primarias (PK) en todas las tablas.
- Claves foráneas (FK) para garantizar integridad referencial.
- Restricciones de unicidad en placas, licencias y tracking numbers.
- Índices en campos críticos (vehicle_id, driver_id, route_id) para optimizar queries.

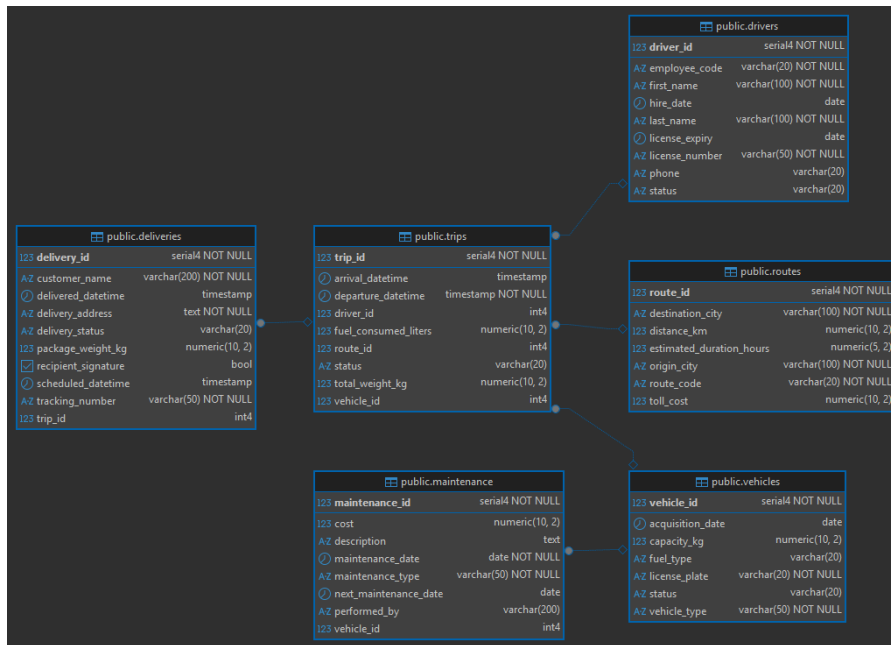


Figura 1. Diagrama Entidad-Relación (ERD) de Fleetlogix.

Se muestran las tablas maestras y transaccionales con sus relaciones y restricciones.

4. Generación de Datos Sintéticos

4.1. Herramientas: Python + Faker + pandas + numpy + psycopg2.

4.2. Volumen generado:

- 200 vehículos
- 400 conductores
- 50 rutas
- 100.000 viajes
- 400.001 entregas*
- 5.000 mantenimientos

4.3. Métodos clave:

- generate_trips(): simula viajes con distribución horaria, consumo de combustible y carga coherente.
- get_hourly_distribution(): concentra actividad entre 6:00–20:00, con picos en 8–12 y 14–18.
- Generación probabilística controlada: asegura realismo en entregas por viaje (2–6) y peso ≤ capacidad.

4.4. ¿Por qué 400.001 y no 400.000?

La razón principal es la generación probabilística. El generador de datos no asigna un número fijo de entregas por viaje, sino que usa una distribución probabilística controlada. Cada viaje (trip) recibe entre 2 y 6 entregas según una probabilidad definida en el método `generate_trips()`. Supongamos que el generador asigna entregas con una media de 4 por viaje. Para 100 000 viajes, el valor esperado es 400 000 entregas. Pero como se usa aleatoriedad (probabilidades), algunos viajes pueden tener 5 o 6 entregas y otros 2 o 3. Esa variación hace que el total final sea de 400 001, un desvío mínimo respecto al valor esperado.

4.5. Aspecto técnico

El generador usa funciones como `random.choice()` o distribuciones de `numpy/Faker`. Estas funciones producen resultados enteros en cada iteración, y al acumularlos se obtiene un total que puede diferir en +1, +2, etc. El hecho de que haya salido 400 001 refleja la naturaleza aleatoria controlada del proceso.

En estadística, esto se llama fluctuación alrededor del valor esperado.

5. Validaciones de Calidad

5.1 Integridad referencial

Que todas las claves foráneas (FK) sean válidas.

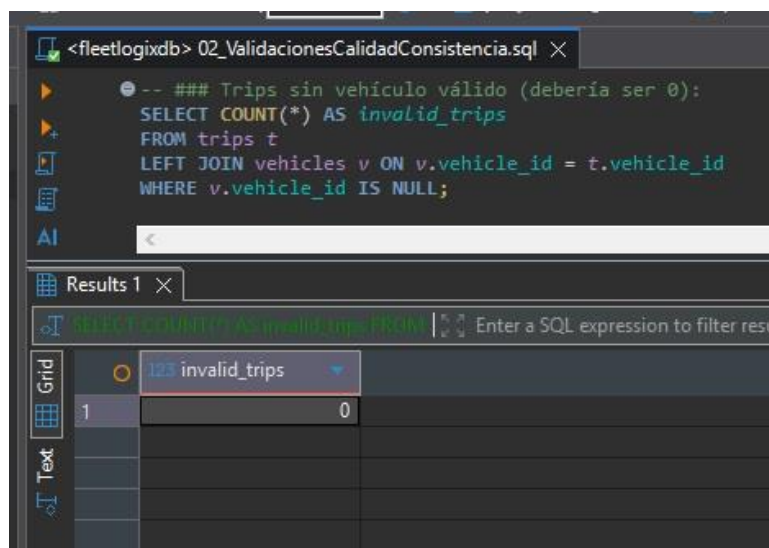
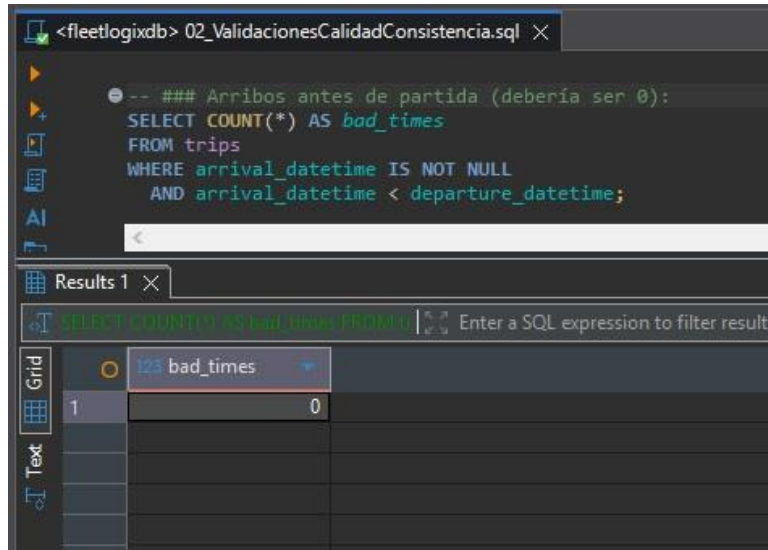


Figura 2. Ej.: Validación de integridad referencial en trips.
No existen viajes sin vehículo asociado, resultado esperado: 0.

5.2 Consistencia temporal

`arrival_datetime > departure_datetime.`



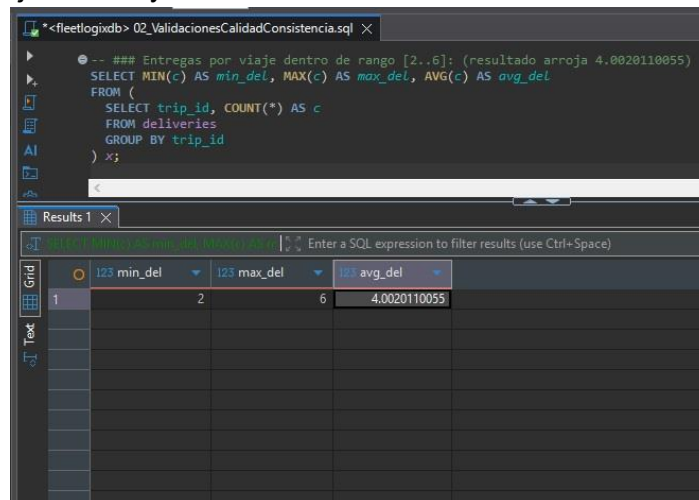
```
-- ### Arribos antes de partida (debería ser 0):
SELECT COUNT(*) AS bad_times
FROM trips
WHERE arrival_datetime IS NOT NULL
      AND arrival_datetime < departure_datetime;
```

bad_times
0

Figura 3. Ej.: Validación de consistencia temporal en trips.
No existen viajes con hora de llegada anterior a la de partida.

5.3 Reglas de negocio

- $\text{Peso} \leq \text{capacidad del vehículo}$
- Tracking numbers únicos
- Entregas por viaje entre 2 y 6



```
-- ### Entregas por viaje dentro de rango [2..6]: (resultado arroja 4.0020110055)
SELECT MIN(c) AS min_del, MAX(c) AS max_del, AVG(c) AS avg_del
FROM (
  SELECT trip_id, COUNT(*) AS c
  FROM deliveries
  GROUP BY trip_id
) x;
```

min_del	max_del	avg_del
2	6	4.0020110055

Figura 4. Ej.: Validación de entregas por viaje.
El promedio de entregas por viaje es 4, dentro del rango esperado [2–6].

6. KPIs Operativos

6.1 Métricas calculadas

- Porcentaje de entregas a tiempo vs retrasadas.
- Consumo promedio de combustible por tipo de vehículo.
- Utilización de capacidad por viaje.
- Mantenimientos por cada 1.000 km.
- Promedio de entregas por viaje.

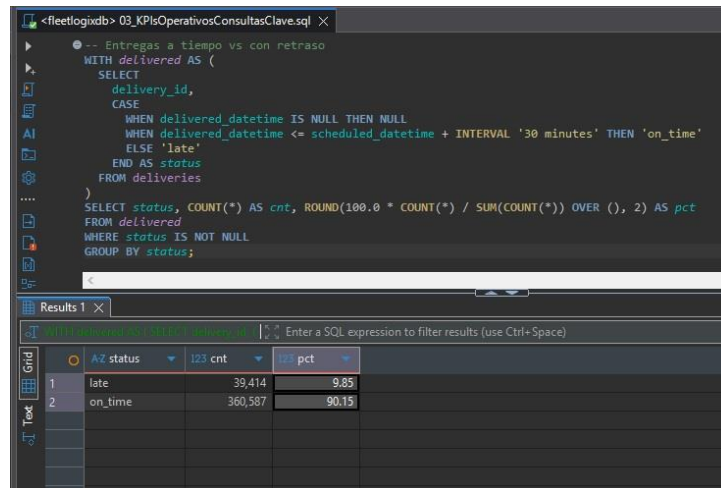


Figura 5. Ej.: KPI de desempeño en entregas.
El 90.15% de las entregas se realizaron a tiempo, el 9.85% con retraso.

7. Arquitectura Cloud Propuesta

- **Ingesta:** PostgreSQL + CDC (Debezium)
- **Streaming:** Apache Kafka
- **Procesamiento:** Apache Flink / Spark Streaming
- **Almacenamiento:** Data Lake + Data Warehouse
- **Visualización:** Power BI / Looker
- **Gobernanza:** Data Catalog + Lineage

La arquitectura cloud de Fleetlogix funciona como un flujo continuo que integra captura, transmisión, procesamiento, almacenamiento y visualización de datos. Los cambios en PostgreSQL se detectan en tiempo real mediante CDC y se envían a través de Kafka, que asegura distribución confiable y escalable. Flink o Spark Streaming procesan los eventos de manera paralela, aplicando reglas y cálculos sobre grandes volúmenes. Los resultados se guardan en un Data Lake para históricos y en un Data Warehouse para análisis. Finalmente, dashboards interactivos permiten decisiones rápidas, con gobernanza que garantiza calidad y trazabilidad.

8. Ejecución

1ra parte:

- a. Abrir PowerShell.
- b. Ir a la ubicación local de los scripts ejemplo: `cd "RUTA\LOCAL\DE\LA\CARPETA"`
- c. `.\run_fleetlogix.ps1`

Si solicita contraseña es la definida en el script .py (puede pedirla una vez antes de crear la DB y otra vez antes de crear el esquema, es la misma ambas veces ya que es la del usuario).
Aquí abajo, tres capturas de pantalla del proceso finalizado:

```

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

PS C:\Users\Federico> cd "C:\Users\Federico\Documents\Data Science\SOY\fleetlogix_scripts"
PS C:\Users\Federico\Documents\Data Science\SOY\fleetlogix_scripts> .run_fleetlogix.ps1
>>
Contraseña para usuario postgres:
Contraseña para usuario postgres:
CREATE DATABASE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE INDEX
CREATE INDEX
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
table_name | column_count
-----
deliveries | 10
drivers | 9
maintenance | 8
routes | 7
trips | 9
vehicles | 7
(6 filas)

tabla_origen | columna_origen | tabla_referencia | columna_referencia
-----
trips | vehicle_id | vehicles | vehicle_id
trips | driver_id | drivers | driver_id
trips | route_id | routes | route_id
deliveries | trip_id | trips | trip_id
maintenance | vehicle_id | vehicles | vehicle_id
(5 filas)

schemaname | tablename | indexname | indexdef
-----
public | deliveries | deliveries_pkey | CREATE UNIQUE INDEX deliveries_pkey ON public.deliveries USING btree (delivery_id)
public | deliveries | deliveries_tracking_number_key | CREATE INDEX INDEX deliveries_tracking_number_key ON public.deliveries USING btree (tracking_number)
public | deliveries | idx_deliveries_status | CREATE INDEX idx_deliveries_status ON public.deliveries USING btree (delivery_status)
public | drivers | drivers_employee_code_key | CREATE UNIQUE INDEX drivers_employee_code_key ON public.drivers USING btree (employee_code)
public | drivers | drivers_license_number_key | CREATE UNIQUE INDEX drivers_license_number_key ON public.drivers USING btree (license_number)
public | drivers | drivers_pkey | CREATE UNIQUE INDEX drivers_pkey ON public.drivers USING btree (driver_id)
  
```

Figura 6. Ejecución del script run_fleetlogix.ps1
 La consola muestra la creación de la base de datos, tablas, índices y comentarios.

```

public | routes | routes_pkey | CREATE UNIQUE INDEX routes_pkey ON public.routes USING btree (route_id)
public | routes | routes_route_code_key | CREATE UNIQUE INDEX routes_route_code_key ON public.routes USING btree (route_code)
public | trips | idx_trips_departure | CREATE INDEX idx_trips_departure ON public.trips USING btree (departure_datetime)
public | trips | trips_pkey | CREATE UNIQUE INDEX trips_pkey ON public.trips USING btree (trip_id)
public | vehicles | idx_vehicles_status | CREATE INDEX idx_vehicles_status ON public.vehicles USING btree (status)
public | vehicles | vehicles_license_plate_key | CREATE UNIQUE INDEX vehicles_license_plate_key ON public.vehicles USING btree (license_plate)
public | vehicles | vehicles_pkey | CREATE UNIQUE INDEX vehicles_pkey ON public.vehicles USING btree (vehicle_id)
(14 filas)

FLEETLOGIX - Generación de Datos Masivos
=====
Objetivo: Generar 505000+ registros manteniendo integridad
=====
2025-12-02 18:22:17,322 - INFO - Conexión exitosa a PostgreSQL
2025-12-02 18:22:17,323 - INFO - Truncando tablas antes de generar datos...
2025-12-02 18:22:17,418 - INFO - Tablas truncadas correctamente
2025-12-02 18:22:17,418 - INFO - Generando 200 vehiculos...
2025-12-02 18:22:17,477 - INFO - 200 vehiculos insertados
2025-12-02 18:22:17,477 - INFO - Generando 400 conductores...
2025-12-02 18:22:17,667 - INFO - 400 conductores insertados
2025-12-02 18:22:17,667 - INFO - Generando 50 rutas...
2025-12-02 18:22:17,681 - INFO - 50 rutas insertadas
2025-12-02 18:22:17,682 - INFO - Generando 100000 viajes...
2025-12-02 18:22:26,921 - INFO - Progreso: 0/100000 trips insertados
2025-12-02 18:22:28,655 - INFO - Progreso: 10000/100000 trips insertados
2025-12-02 18:22:30,619 - INFO - Progreso: 20000/100000 trips insertados
2025-12-02 18:22:32,996 - INFO - Progreso: 30000/100000 trips insertados
2025-12-02 18:22:34,968 - INFO - Progreso: 40000/100000 trips insertados
2025-12-02 18:22:37,022 - INFO - Progreso: 50000/100000 trips insertados
2025-12-02 18:22:39,145 - INFO - Progreso: 60000/100000 trips insertados
2025-12-02 18:22:41,511 - INFO - Progreso: 70000/100000 trips insertados
2025-12-02 18:22:43,576 - INFO - Progreso: 80000/100000 trips insertados
2025-12-02 18:22:45,546 - INFO - Progreso: 90000/100000 trips insertados
2025-12-02 18:22:47,310 - INFO - 100000 viajes insertados
2025-12-02 18:22:47,344 - INFO - Generando 400000 entregas...
2025-12-02 18:25:31,521 - INFO - Progreso: 0/400000 deliveries insertados
2025-12-02 18:25:42,075 - INFO - Progreso: 50000/400000 deliveries insertados
2025-12-02 18:25:51,971 - INFO - Progreso: 100000/400000 deliveries insertados
2025-12-02 18:26:02,817 - INFO - Progreso: 150000/400000 deliveries insertados
2025-12-02 18:26:13,350 - INFO - Progreso: 200000/400000 deliveries insertados
2025-12-02 18:26:24,093 - INFO - Progreso: 250000/400000 deliveries insertados
2025-12-02 18:26:34,866 - INFO - Progreso: 300000/400000 deliveries insertados
2025-12-02 18:26:45,205 - INFO - Progreso: 350000/400000 deliveries insertados
2025-12-02 18:26:55,510 - INFO - Progreso: 400000/400000 deliveries insertados
2025-12-02 18:26:55,511 - INFO - 400000 entregas insertadas
2025-12-02 18:26:55,763 - INFO - Generando 5000 registros de mantenimiento...
2025-12-02 18:26:58,700 - INFO - 5000 mantenimientos insertados
2025-12-02 18:26:58,728 - INFO -
RESUMEN DE GENERACION DE DATOS
2025-12-02 18:26:58,729 - INFO -
  
```

Figura 7. Generación masiva de datos sintéticos.
 Se observa la inserción de más de 505.000 registros en PostgreSQL con logs de progreso.


```
Windows PowerShell
2025-12-02 18:26:55,763 - INFO - Generando 5000 registros de mantenimiento...
2025-12-02 18:26:58,709 - INFO - 5000 mantenimientos insertados
2025-12-02 18:26:58,728 - INFO -
RESUMEN DE GENERACIÓN DE DATOS
2025-12-02 18:26:58,729 - INFO - =====
2025-12-02 18:26:58,731 - INFO - vehicles: 200 registros
2025-12-02 18:26:58,734 - INFO - drivers: 400 registros
2025-12-02 18:26:58,737 - INFO - routes: 48 registros
2025-12-02 18:26:58,748 - INFO - trips: 100,000 registros
2025-12-02 18:26:58,906 - INFO - deliveries: 400,001 registros
2025-12-02 18:26:58,910 - INFO - maintenance: 5,000 registros
2025-12-02 18:26:58,911 - INFO -
TOTAL: 505,049 registros
2025-12-02 18:26:59,109 - INFO -
Entregas por viaje: AVG=4.0, MIN=2, MAX=6
2025-12-02 18:26:59,109 - INFO -
VALIDANDO CALIDAD DE DATOS...
2025-12-02 18:26:59,136 - INFO - Integridad referencial - Trips sin vehículo válido: OK
2025-12-02 18:26:59,369 - INFO - Integridad referencial - Deliveries sin trip válido: OK
2025-12-02 18:26:59,385 - INFO - Consistencia temporal - Trips con arrival < departure: OK
2025-12-02 18:26:59,434 - INFO - Consistencia de peso - Trips excediendo capacidad: OK
2025-12-02 18:26:59,437 - INFO - Entregas sin tracking number: OK
2025-12-02 18:26:59,441 - INFO -
Resumen guardado en generation_summary.json
2025-12-02 18:26:59,442 - INFO -
Conexión cerrada
PS C:\Users\Feder\Documents\Data_Science\m02\Fleetlogix_pi\scripts>
```

Figura 8. Resumen de generación y validaciones automáticas.
El log confirma integridad referencial, consistencia temporal y unicidad de claves.

2da parte:

4. Crear una nueva conexión para visualizar el esquema en DBeaver:

- Database -> New Database Connection -> PostgreSQL ->

- Rellenar con los parámetros definidos en el script:

Host:localhost Port:5432 Database:fleetlogixdb Username:postgres Password:fede0309

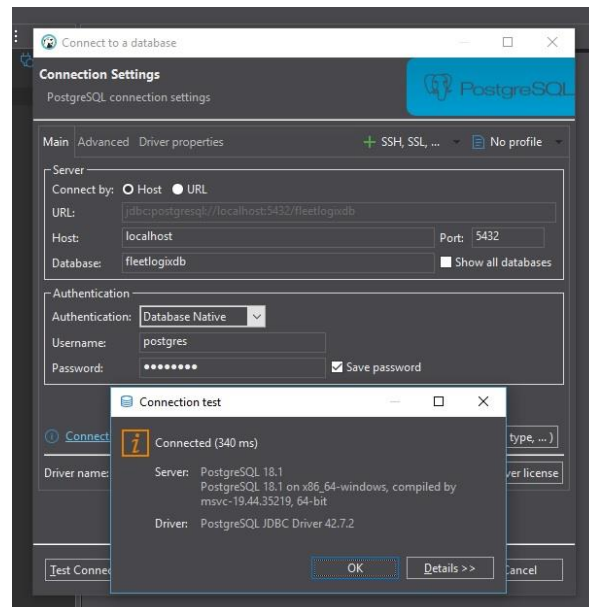


Figura 9. Conexión a la base fleetlogixdb en Dbeaver. Se confirma conexión exitosa con PostgreSQL 18.1 y la base está lista para la tercera parte de la ejecución.

3ra parte

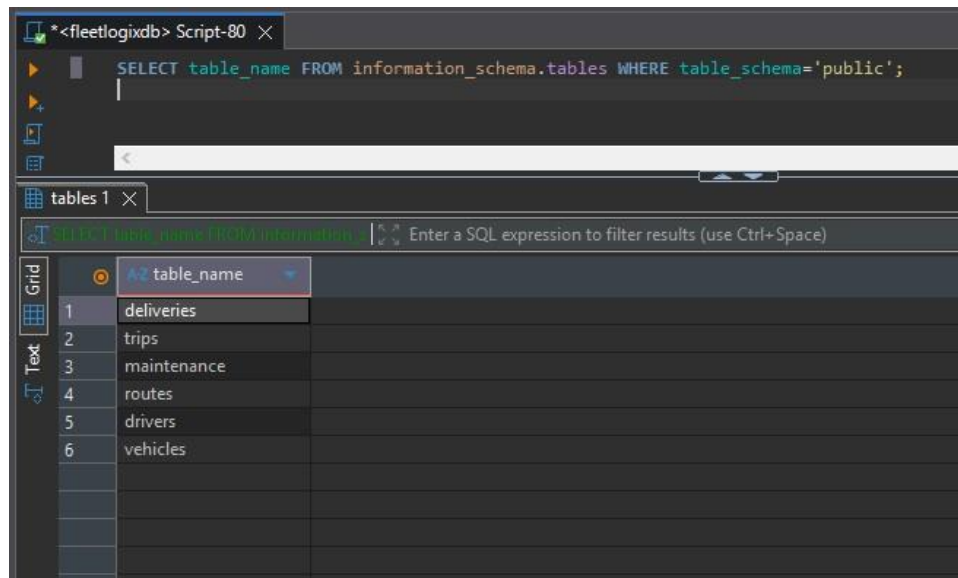
5. Verificar carga de datos

- En la misma DB abrir y ejecutar las siguientes queries en cualquier orden, una a una o todas juntas:

'01_InventarioDeTablasPKsFKsIndices'

'02_ValidacionesCalidadConsistencia.sql'

'03_KPIsOperativosConsultasClave.sql'



*Figura 10. Inventario de tablas creadas en el esquema public.
Se listan las tablas: deliveries, trips, maintenance, routes, drivers y vehicles.*

9. Conclusiones

El diagnóstico y ensayo sobre el proyecto Fleetlogix demuestra que es posible realizar una migración desde sistemas legacy hacia una infraestructura moderna de datos, garantizando calidad, integridad y consistencia. El modelo relacional soporta operaciones logísticas a gran escala, mientras que el modelo dimensional habilita análisis OLAP y dashboards. La arquitectura cloud propuesta asegura escalabilidad y decisiones en tiempo real.