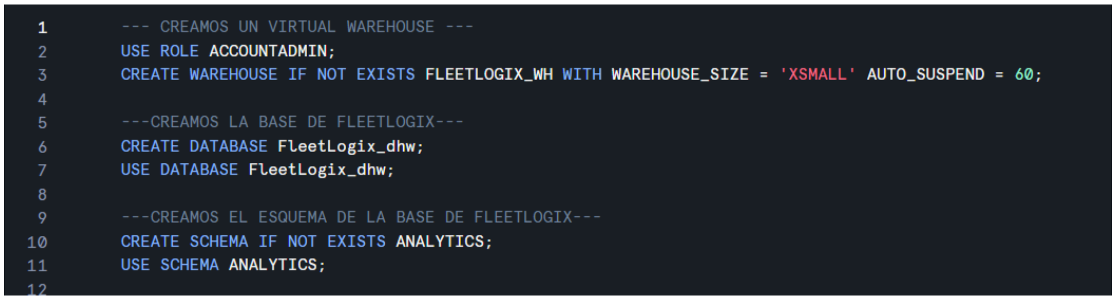
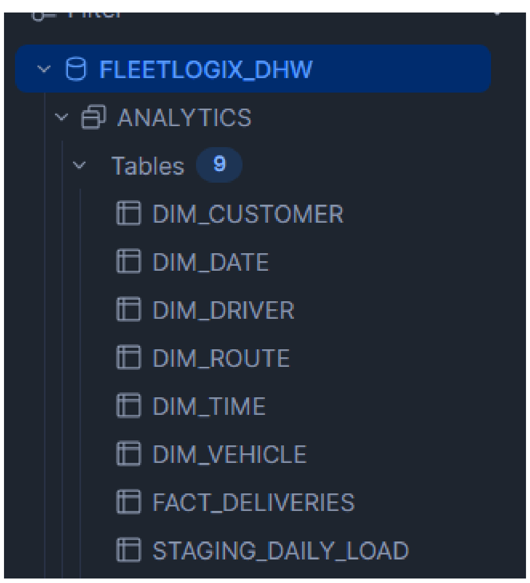
Avance 3: Análisis de Snowflake

Para poder armar el flujo de datos de nuestra base de Postgre a Snowflake lo primero que hay que hacer es crear la estructura de nuestro sistema OLTP:

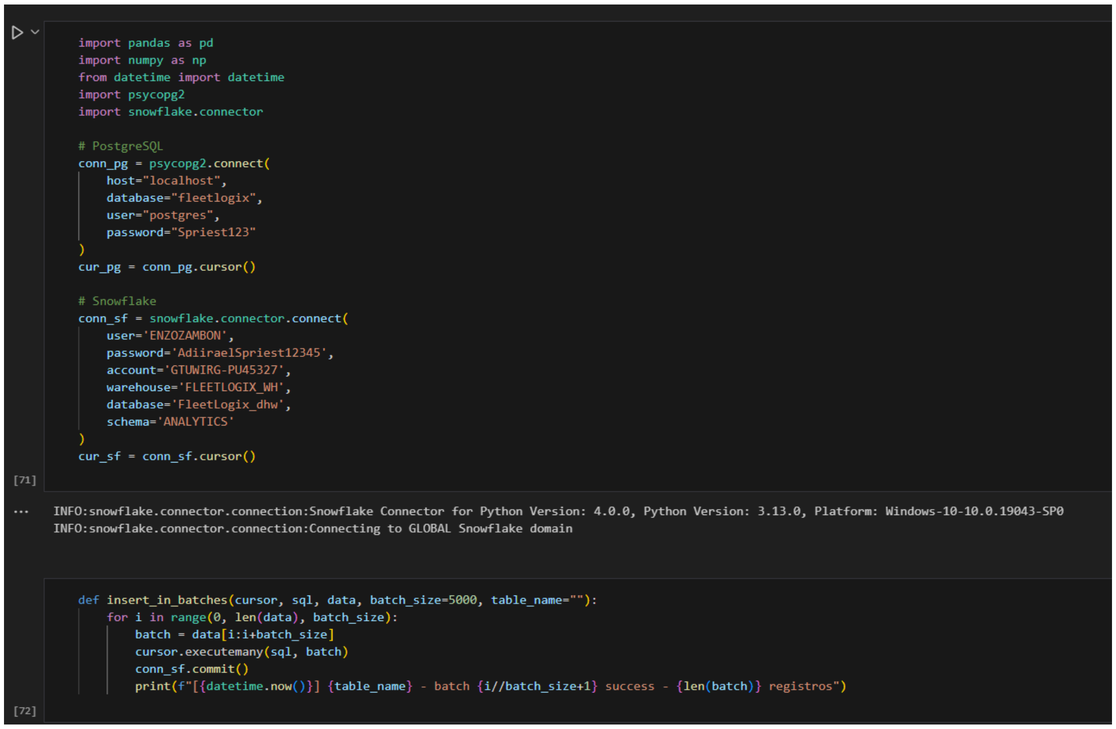


```sql
--- CREAMOS UN VIRTUAL WAREHOUSE ---
USE ROLE ACCOUNTADMIN;
CREATE WAREHOUSE IF NOT EXISTS FLEETLOGIX_WH WITH WAREHOUSE_SIZE = 'XSMALL' AUTO_SUSPEND = 60;

---CREAMOS LA BASE DE FLEETLOGIX---
CREATE DATABASE FleetLogix_dhw;
USE DATABASE FleetLogix_dhw;

---CREAMOS EL ESQUEMA DE LA BASE DE FLEETLOGIX---
CREATE SCHEMA IF NOT EXISTS ANALYTICS;
USE SCHEMA ANALYTICS;
```



```
FLEETLOGIX_DHW
  ANALYTICS
    Tables 9
      DIM_CUSTOMER
      DIM_DATE
      DIM_DRIVER
      DIM_ROUTE
      DIM_TIME
      DIM_VEHICLE
      FACT_DELIVERIES
      STAGING_DAILY_LOAD
```

Luego creamos las conexiones entre las herramientas

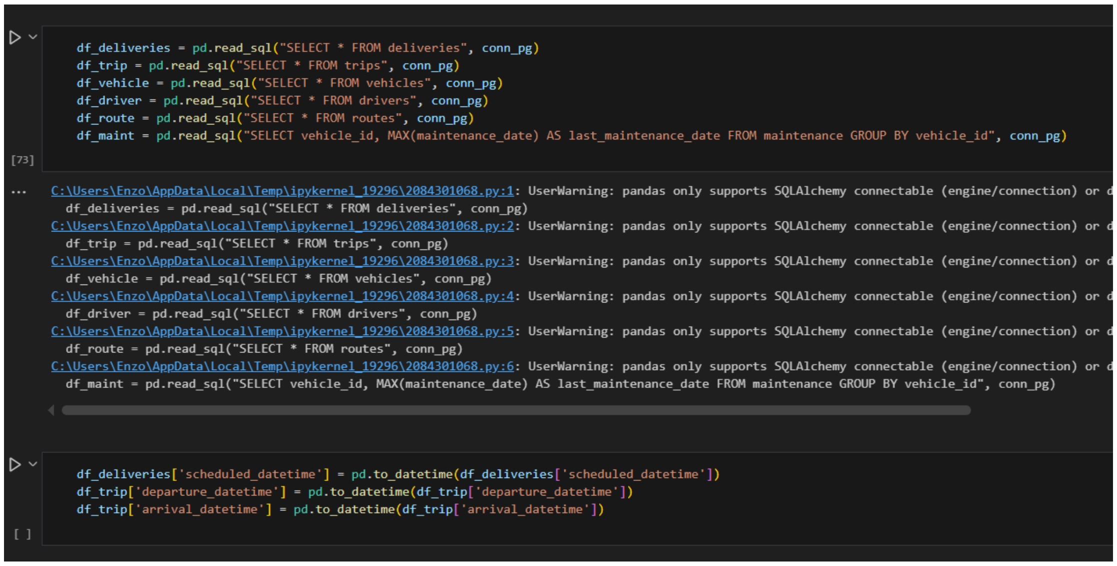


```
import pandas as pd
import numpy as np
from datetime import datetime
import psycopg2
import snowflake.connector

# PostgreSQL
conn_pg = psycopg2.connect(
    host="localhost",
    database="fleetlogix",
    user="postgres",
    password="Spriest123"
)
cur_pg = conn_pg.cursor()

# Snowflake
conn_sf = snowflake.connector.connect(
    user='ENZOZAMBON',
    password='AdiiraelSpriest12345',
    account='GTUWIRG-PU45327',
    warehouse='FLEETLOGIX_WH',
    database='FleetLogix_dhw',
    schema='ANALYTICS'
)
cur_sf = conn_sf.cursor()
```
[71]

```
INFO:snowflake.connector.connection:Snowflake Connector for Python Version: 4.0.0, Python Version: 3.13.0, Platform: Windows-10-10.0.19043-SP0
INFO:snowflake.connector.connection:Connecting to GLOBAL Snowflake domain
```

```
def insert_in_batches(cursor, sql, data, batch_size=5000, table_name=""):
    for i in range(0, len(data), batch_size):
        batch = data[i:i+batch_size]
        cursor.executemany(sql, batch)
        conn_sf.commit()
        print(f"[{datetime.now()}] {table_name} - batch {i//batch_size+1} success - {len(batch)} registros")
```
[72]

Posteriormente cargamos la información de las tablas en multiples dataframes:

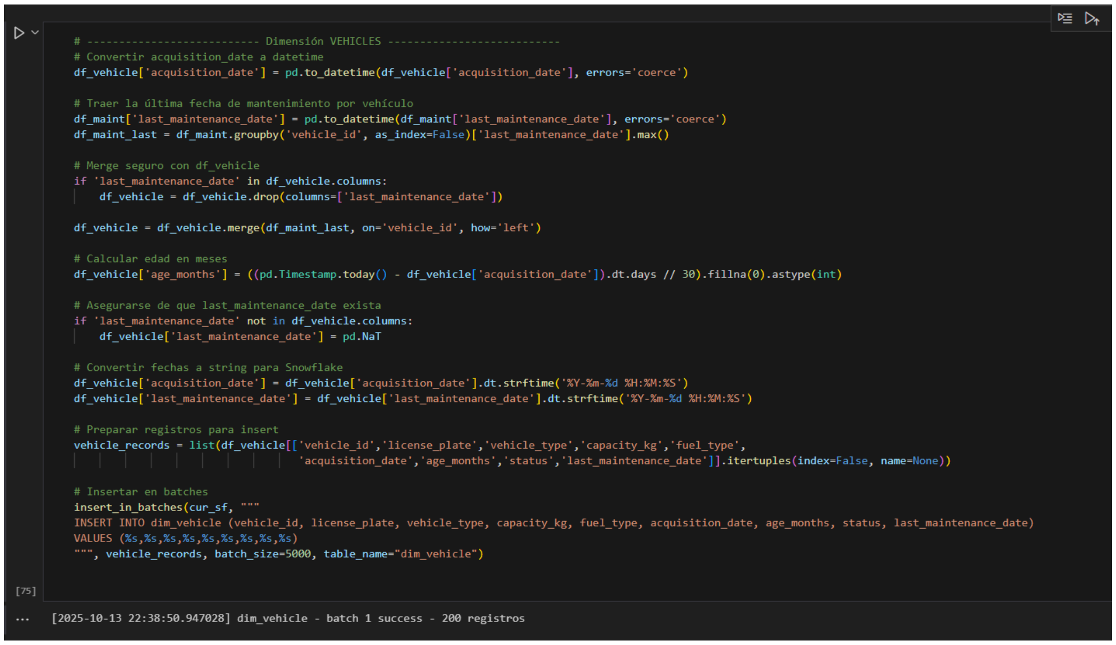


```
df_deliveries = pd.read_sql("SELECT * FROM deliveries", conn_pg)
df_trip = pd.read_sql("SELECT * FROM trips", conn_pg)
df_vehicle = pd.read_sql("SELECT * FROM vehicles", conn_pg)
df_driver = pd.read_sql("SELECT * FROM drivers", conn_pg)
df_route = pd.read_sql("SELECT * FROM routes", conn_pg)
df_maint = pd.read_sql("SELECT vehicle_id, MAX(maintenance_date) AS last_maintenance_date FROM maintenance GROUP BY vehicle_id", conn_pg)
```
[73]

```
C:\Users\Enzo\AppData\Local\Temp\ipykernel_19296\2084301068.py:1: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or d
  df_deliveries = pd.read_sql("SELECT * FROM deliveries", conn_pg)
C:\Users\Enzo\AppData\Local\Temp\ipykernel_19296\2084301068.py:2: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or d
  df_trip = pd.read_sql("SELECT * FROM trips", conn_pg)
C:\Users\Enzo\AppData\Local\Temp\ipykernel_19296\2084301068.py:3: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or d
  df_vehicle = pd.read_sql("SELECT * FROM vehicles", conn_pg)
C:\Users\Enzo\AppData\Local\Temp\ipykernel_19296\2084301068.py:4: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or d
  df_driver = pd.read_sql("SELECT * FROM drivers", conn_pg)
C:\Users\Enzo\AppData\Local\Temp\ipykernel_19296\2084301068.py:5: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or d
  df_route = pd.read_sql("SELECT * FROM routes", conn_pg)
C:\Users\Enzo\AppData\Local\Temp\ipykernel_19296\2084301068.py:6: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or d
  df_maint = pd.read_sql("SELECT vehicle_id, MAX(maintenance_date) AS last_maintenance_date FROM maintenance GROUP BY vehicle_id", conn_pg)
```

```
df_deliveries['scheduled_datetime'] = pd.to_datetime(df_deliveries['scheduled_datetime'])
df_trip['departure_datetime'] = pd.to_datetime(df_trip['departure_datetime'])
df_trip['arrival_datetime'] = pd.to_datetime(df_trip['arrival_datetime'])
```
[ ]

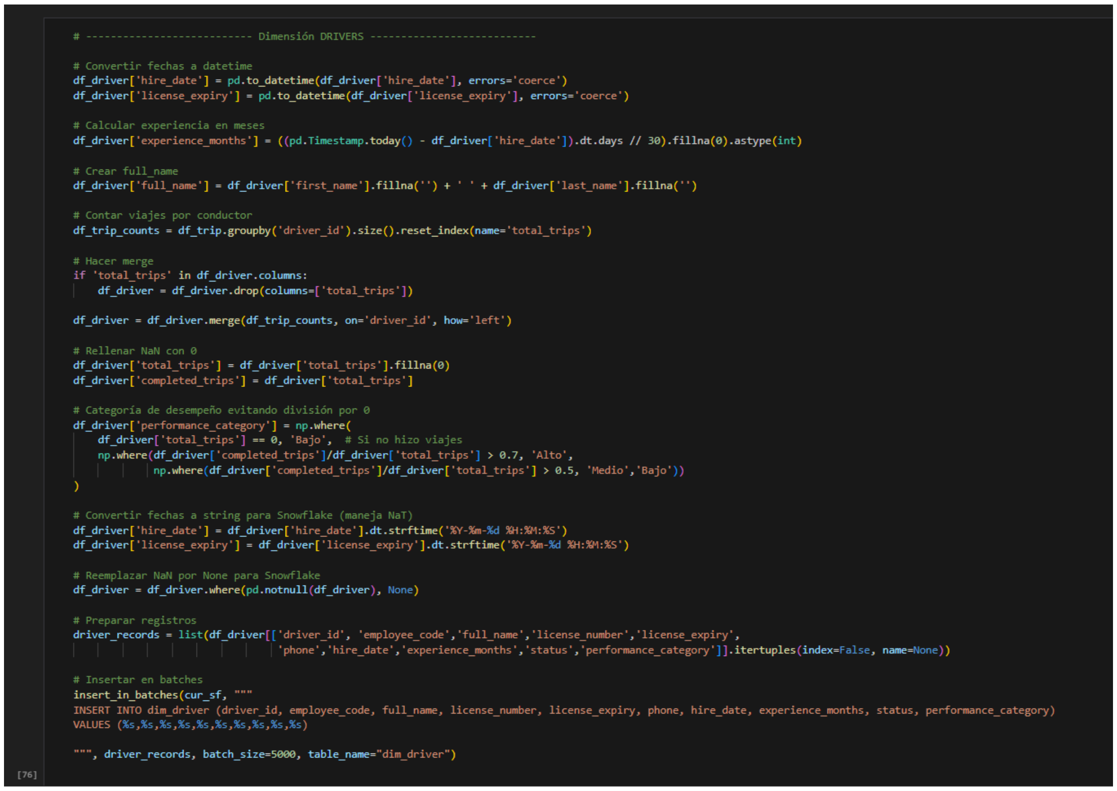
Y poblamos las tablas de Snowflake, procesamiento de datos mediante acorde a la estructura de las tablas previamente creadas



```python
# -------------------------- Dimensión VEHICLES --------------------------
# Convertir acquisition_date a datetime
df_vehicle['acquisition_date'] = pd.to_datetime(df_vehicle['acquisition_date'], errors='coerce')

# Traer la última fecha de mantenimiento por vehículo
df_maint['last_maintenance_date'] = pd.to_datetime(df_maint['last_maintenance_date'], errors='coerce')
df_maint_last = df_maint.groupby('vehicle_id', as_index=False)['last_maintenance_date'].max()

# Merge seguro con df_vehicle
if 'last_maintenance_date' in df_vehicle.columns:
    df_vehicle = df_vehicle.drop(columns=['last_maintenance_date'])

df_vehicle = df_vehicle.merge(df_maint_last, on='vehicle_id', how='left')

# Calcular edad en meses
df_vehicle['age_months'] = ((pd.Timestamp.today() - df_vehicle['acquisition_date']).dt.days // 30).fillna(0).astype(int)

# Asegurarse de que last_maintenance_date exista
if 'last_maintenance_date' not in df_vehicle.columns:
    df_vehicle['last_maintenance_date'] = pd.NaT

# Convertir fechas a string para Snowflake
df_vehicle['acquisition_date'] = df_vehicle['acquisition_date'].dt.strftime('%Y-%m-%d %H:%M:%S')
df_vehicle['last_maintenance_date'] = df_vehicle['last_maintenance_date'].dt.strftime('%Y-%m-%d %H:%M:%S')

# Preparar registros para insert
vehicle_records = list(df_vehicle[['vehicle_id','license_plate','vehicle_type','capacity_kg','fuel_type',
                                   'acquisition_date','age_months','status','last_maintenance_date']].itertuples(index=False, name=None))

# Insertar en batches
insert_in_batches(cur_sf, """
INSERT INTO dim_vehicle (vehicle_id, license_plate, vehicle_type, capacity_kg, fuel_type, acquisition_date, age_months, status, last_maintenance_date)
VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s)
""", vehicle_records, batch_size=5000, table_name="dim_vehicle")
```

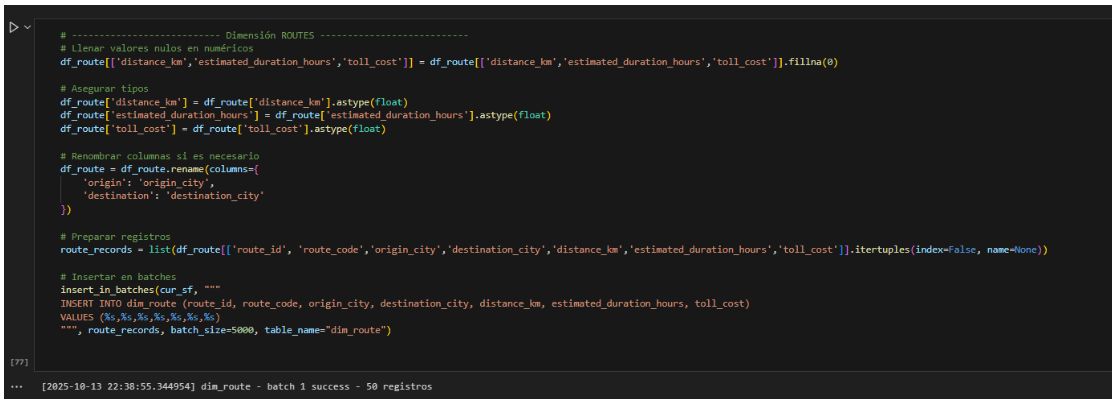
[75]

```
[2025-10-13 22:38:50.947028] dim_vehicle - batch 1 success - 200 registros
```



```python
# -------------------------- Dimensión DRIVERS --------------------------

# Convertir fechas a datetime
df_driver['hire_date'] = pd.to_datetime(df_driver['hire_date'], errors='coerce')
df_driver['license_expiry'] = pd.to_datetime(df_driver['license_expiry'], errors='coerce')

# Calcular experiencia en meses
df_driver['experience_months'] = ((pd.Timestamp.today() - df_driver['hire_date']).dt.days // 30).fillna(0).astype(int)

# Crear full_name
df_driver['full_name'] = df_driver['first_name'].fillna('') + ' ' + df_driver['last_name'].fillna('')

# Contar viajes por conductor
df_trip_counts = df_trip.groupby('driver_id').size().reset_index(name='total_trips')

# Hacer merge
if 'total_trips' in df_driver.columns:
    df_driver = df_driver.drop(columns=['total_trips'])

df_driver = df_driver.merge(df_trip_counts, on='driver_id', how='left')

# Rellenar NaN con 0
df_driver['total_trips'] = df_driver['total_trips'].fillna(0)
df_driver['completed_trips'] = df_driver['total_trips']

# Categoría de desempeño evitando división por 0
df_driver['performance_category'] = np.where(
    df_driver['total_trips'] == 0, 'Bajo',  # Si no hizo viajes
    np.where(df_driver['completed_trips']/df_driver['total_trips'] > 0.7, 'Alto',
             np.where(df_driver['completed_trips']/df_driver['total_trips'] > 0.5, 'Medio','Bajo'))
)

# Convertir fechas a string para Snowflake (maneja NaT)
df_driver['hire_date'] = df_driver['hire_date'].dt.strftime('%Y-%m-%d %H:%M:%S')
df_driver['license_expiry'] = df_driver['license_expiry'].dt.strftime('%Y-%m-%d %H:%M:%S')

# Reemplazar NaN por None para Snowflake
df_driver = df_driver.where(pd.notnull(df_driver), None)

# Preparar registros
driver_records = list(df_driver[['driver_id', 'employee_code','full_name','license_number','license_expiry',
                                 'phone','hire_date','experience_months','status','performance_category']].itertuples(index=False, name=None))

# Insertar en batches
insert_in_batches(cur_sf, """
INSERT INTO dim_driver (driver_id, employee_code, full_name, license_number, license_expiry, phone, hire_date, experience_months, status, performance_category)
VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)

""", driver_records, batch_size=5000, table_name="dim_driver")
```

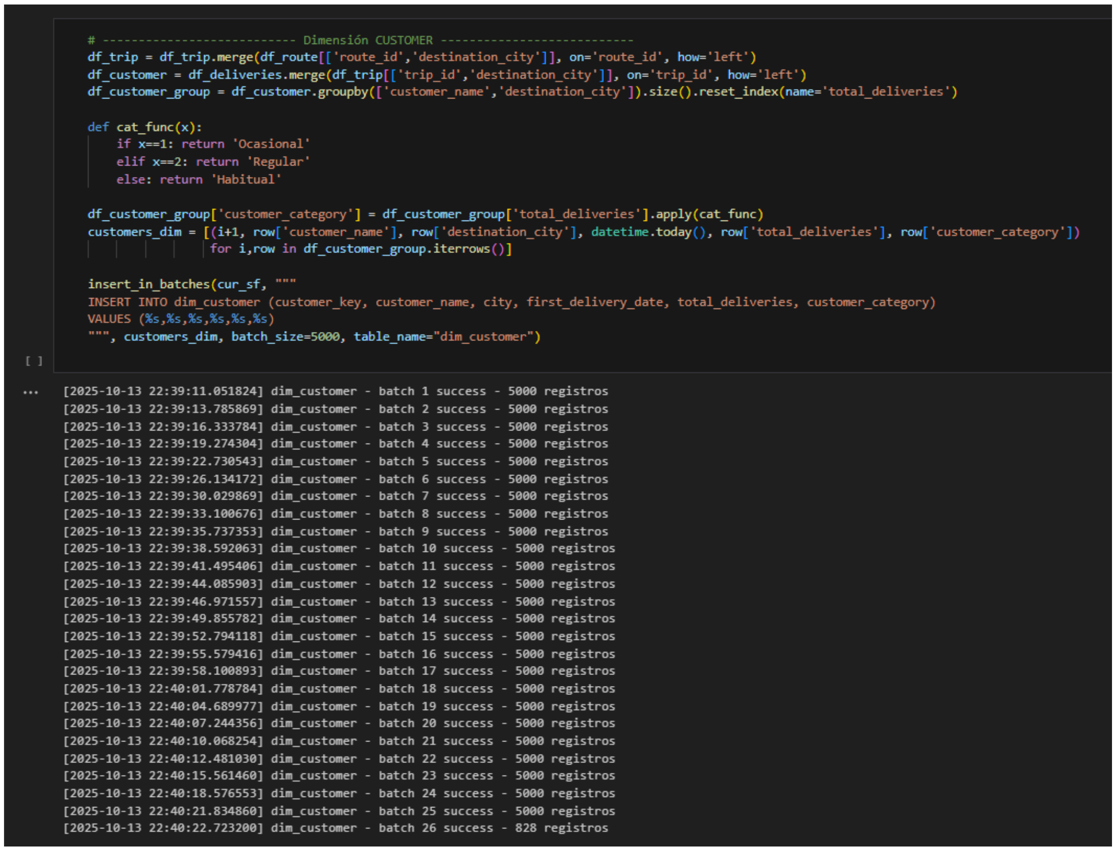
[76]



```
# ------------------------- Dimensión ROUTES -------------------------
# Llenar valores nulos en numéricos
df_route[['distance_km','estimated_duration_hours','toll_cost']] = df_route[['distance_km','estimated_duration_hours','toll_cost']].fillna(0)

# Asegurar tipos
df_route['distance_km'] = df_route['distance_km'].astype(float)
df_route['estimated_duration_hours'] = df_route['estimated_duration_hours'].astype(float)
df_route['toll_cost'] = df_route['toll_cost'].astype(float)

# Renombrar columnas si es necesario
df_route = df_route.rename(columns={
    'origin': 'origin_city',
    'destination': 'destination_city'
})

# Preparar registros
route_records = list(df_route[['route_id', 'route_code','origin_city','destination_city','distance_km','estimated_duration_hours','toll_cost']].itertuples(index=False, name=None))

# Insertar en batches
insert_in_batches(cur_sf, """
INSERT INTO dim_route (route_id, route_code, origin_city, destination_city, distance_km, estimated_duration_hours, toll_cost)
VALUES (%s,%s,%s,%s,%s,%s,%s)
""", route_records, batch_size=5000, table_name="dim_route")
```

[77]

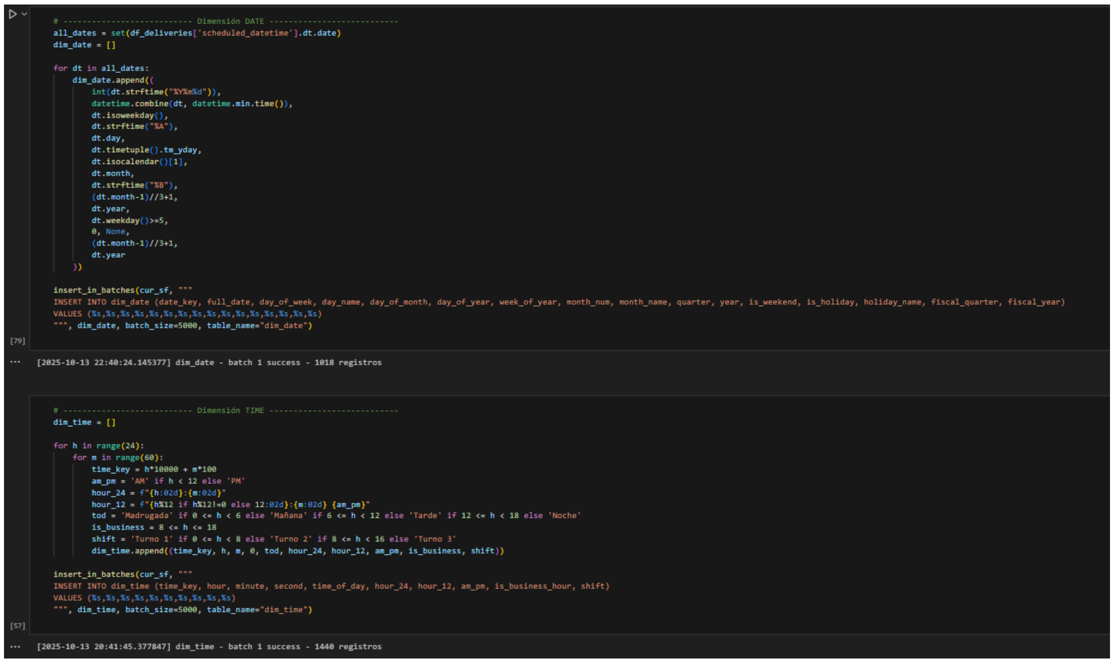
```
[2025-10-13 22:38:55.344954] dim_route - batch 1 success - 50 registros
```



```
# --------------------------- Dimensión CUSTOMER ---------------------------
df_trip = df_trip.merge(df_route[['route_id','destination_city']], on='route_id', how='left')
df_customer = df_deliveries.merge(df_trip[['trip_id','destination_city']], on='trip_id', how='left')
df_customer_group = df_customer.groupby(['customer_name','destination_city']).size().reset_index(name='total_deliveries')

def cat_func(x):
    if x==1: return 'Ocasional'
    elif x==2: return 'Regular'
    else: return 'Habitual'

df_customer_group['customer_category'] = df_customer_group['total_deliveries'].apply(cat_func)
customers_dim = [(i+1, row['customer_name'], row['destination_city'], datetime.today(), row['total_deliveries'], row['customer_category'])
                 for i,row in df_customer_group.iterrows()]

insert_in_batches(cur_sf, """
INSERT INTO dim_customer (customer_key, customer_name, city, first_delivery_date, total_deliveries, customer_category)
VALUES (%s,%s,%s,%s,%s,%s)
""", customers_dim, batch_size=5000, table_name="dim_customer")
```

[ ]

```
[2025-10-13 22:39:11.051824] dim_customer - batch 1 success - 5000 registros
[2025-10-13 22:39:13.785869] dim_customer - batch 2 success - 5000 registros
[2025-10-13 22:39:16.333784] dim_customer - batch 3 success - 5000 registros
[2025-10-13 22:39:19.274304] dim_customer - batch 4 success - 5000 registros
[2025-10-13 22:39:22.730543] dim_customer - batch 5 success - 5000 registros
[2025-10-13 22:39:26.134172] dim_customer - batch 6 success - 5000 registros
[2025-10-13 22:39:30.029869] dim_customer - batch 7 success - 5000 registros
[2025-10-13 22:39:33.100676] dim_customer - batch 8 success - 5000 registros
[2025-10-13 22:39:35.737353] dim_customer - batch 9 success - 5000 registros
[2025-10-13 22:39:38.592063] dim_customer - batch 10 success - 5000 registros
[2025-10-13 22:39:41.495406] dim_customer - batch 11 success - 5000 registros
[2025-10-13 22:39:44.085903] dim_customer - batch 12 success - 5000 registros
[2025-10-13 22:39:46.971557] dim_customer - batch 13 success - 5000 registros
[2025-10-13 22:39:49.855782] dim_customer - batch 14 success - 5000 registros
[2025-10-13 22:39:52.794118] dim_customer - batch 15 success - 5000 registros
[2025-10-13 22:39:55.579416] dim_customer - batch 16 success - 5000 registros
[2025-10-13 22:39:58.100893] dim_customer - batch 17 success - 5000 registros
[2025-10-13 22:40:01.778784] dim_customer - batch 18 success - 5000 registros
[2025-10-13 22:40:04.689977] dim_customer - batch 19 success - 5000 registros
[2025-10-13 22:40:07.244356] dim_customer - batch 20 success - 5000 registros
[2025-10-13 22:40:10.068254] dim_customer - batch 21 success - 5000 registros
[2025-10-13 22:40:12.481030] dim_customer - batch 22 success - 5000 registros
[2025-10-13 22:40:15.561460] dim_customer - batch 23 success - 5000 registros
[2025-10-13 22:40:18.576553] dim_customer - batch 24 success - 5000 registros
[2025-10-13 22:40:21.834860] dim_customer - batch 25 success - 5000 registros
[2025-10-13 22:40:22.723200] dim_customer - batch 26 success - 828 registros
```

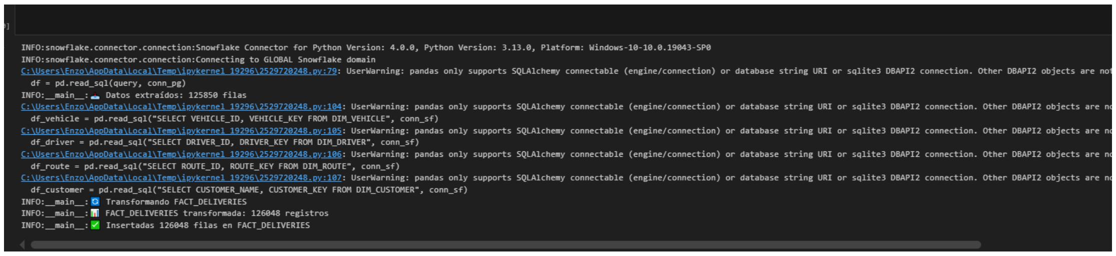


```python
# -------------------------- Dimensión DATE --------------------------
all_dates = set(df_deliveries['scheduled_datetime'].dt.date)
dim_date = []

for dt in all_dates:
    dim_date.append((
        int(dt.strftime("%Y%m%d")),
        datetime.combine(dt, datetime.min.time()),
        dt.isoweekday(),
        dt.strftime("%A"),
        dt.day,
        dt.timetuple().tm_yday,
        dt.isocalendar()[1],
        dt.month,
        dt.strftime("%B"),
        (dt.month-1)//3+1,
        dt.year,
        dt.weekday()>=5,
        0, None,
        (dt.month-1)//3+1,
        dt.year
    ))

insert_in_batches(cur_sf, """
INSERT INTO dim_date (date_key, full_date, day_of_week, day_name, day_of_month, day_of_year, week_of_year, month_num, month_name, quarter, year, is_weekend, is_holiday, holiday_name, fiscal_quarter, fiscal_year)
VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)
""", dim_date, batch_size=5000, table_name="dim_date")
```

```
[79]
[2025-10-13 22:40:24.145377] dim_date - batch 1 success - 1018 registros
```

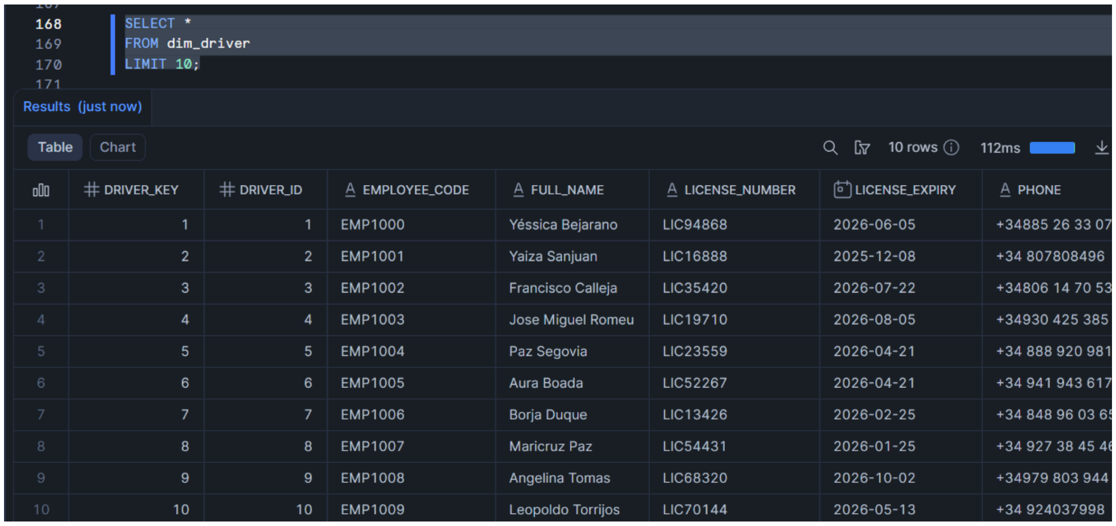
```python
# -------------------------- Dimensión TIME --------------------------
dim_time = []

for h in range(24):
    for m in range(60):
        time_key = h*10000 + m*100
        am_pm = 'AM' if h < 12 else 'PM'
        hour_24 = f"{h:02d}:{m:02d}"
        hour_12 = f"{h%12 if h%12!=0 else 12:02d}:{m:02d} {am_pm}"
        tod = 'Madrugada' if 0 <= h < 6 else 'Mañana' if 6 <= h < 12 else 'Tarde' if 12 <= h < 18 else 'Noche'
        is_business = 8 <= h <= 18
        shift = 'Turno 1' if 0 <= h < 8 else 'Turno 2' if 8 <= h < 16 else 'Turno 3'
        dim_time.append((time_key, h, m, 0, tod, hour_24, hour_12, am_pm, is_business, shift))

insert_in_batches(cur_sf, """
INSERT INTO dim_time (time_key, hour, minute, second, time_of_day, hour_24, hour_12, am_pm, is_business_hour, shift)
VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)
""", dim_time, batch_size=5000, table_name="dim_time")
```

```
[57]
[2025-10-13 20:41:45.377847] dim_time - batch 1 success - 1440 registros
```



```
INFO:snowflake.connector.connection:Snowflake Connector for Python Version: 4.0.0, Python Version: 3.13.0, Platform: Windows-10-10.0.19043-SP0
INFO:snowflake.connector.connection:Connecting to GLOBAL Snowflake domain
C:\Users\Enzo\AppData\Local\Temp\ipykernel_19296\2529720248.py:79: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not
  df = pd.read_sql(query, conn_pg)
INFO:__main__:📥 Datos extraídos: 125850 filas
C:\Users\Enzo\AppData\Local\Temp\ipykernel_19296\2529720248.py:104: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are no
  df_vehicle = pd.read_sql("SELECT VEHICLE_ID, VEHICLE_KEY FROM DIM_VEHICLE", conn_sf)
C:\Users\Enzo\AppData\Local\Temp\ipykernel_19296\2529720248.py:105: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are no
  df_driver = pd.read_sql("SELECT DRIVER_ID, DRIVER_KEY FROM DIM_DRIVER", conn_sf)
C:\Users\Enzo\AppData\Local\Temp\ipykernel_19296\2529720248.py:106: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are no
  df_route = pd.read_sql("SELECT ROUTE_ID, ROUTE_KEY FROM DIM_ROUTE", conn_sf)
C:\Users\Enzo\AppData\Local\Temp\ipykernel_19296\2529720248.py:107: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are no
  df_customer = pd.read_sql("SELECT CUSTOMER_NAME, CUSTOMER_KEY FROM DIM_CUSTOMER", conn_sf)
INFO:__main__:🔄 Transformando FACT_DELIVERIES
INFO:__main__:📊 FACT_DELIVERIES transformada: 126048 registros
INFO:__main__:✅ Insertadas 126048 filas en FACT_DELIVERIES
```

Finalmente verificamos que los datos estén correctamente cargados en nuestras tablas

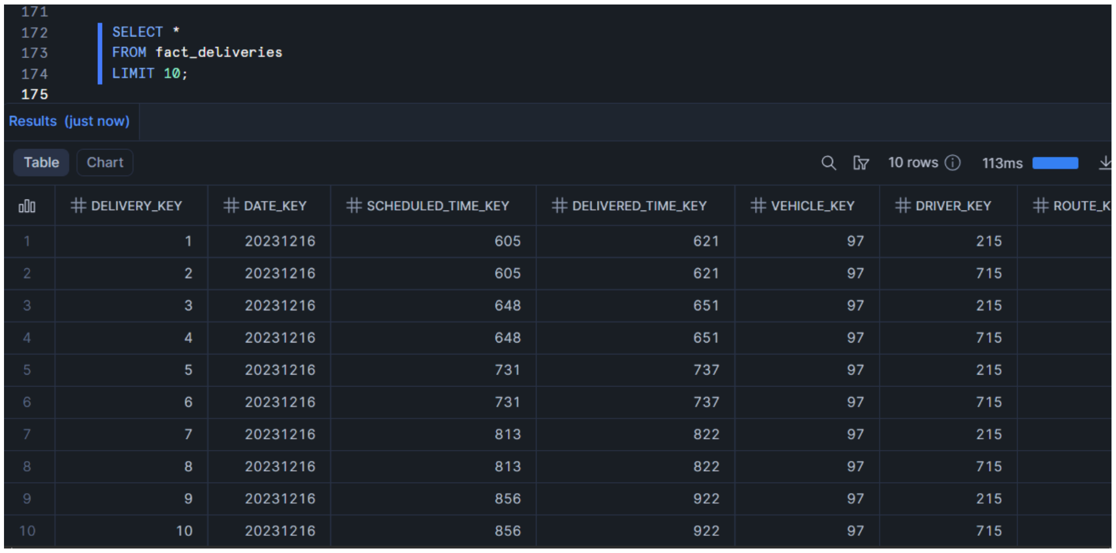


```sql
SELECT *
FROM dim_driver
LIMIT 10;
```

Results (just now)

Table Chart 10 rows 112ms

	DRIVER_KEY	DRIVER_ID	EMPLOYEE_CODE	FULL_NAME	LICENSE_NUMBER	LICENSE_EXPIRY	PHONE
1	1	1	EMP1000	Yéssica Bejarano	LIC94868	2026-06-05	+34885 26 33 07
2	2	2	EMP1001	Yaiza Sanjuan	LIC16888	2025-12-08	+34 807808496
3	3	3	EMP1002	Francisco Calleja	LIC35420	2026-07-22	+34806 14 70 53
4	4	4	EMP1003	Jose Miguel Romeu	LIC19710	2026-08-05	+34930 425 385
5	5	5	EMP1004	Paz Segovia	LIC23559	2026-04-21	+34 888 920 981
6	6	6	EMP1005	Aura Boada	LIC52267	2026-04-21	+34 941 943 617
7	7	7	EMP1006	Borja Duque	LIC13426	2026-02-25	+34 848 96 03 6
8	8	8	EMP1007	Maricruz Paz	LIC54431	2026-01-25	+34 927 38 45 4
9	9	9	EMP1008	Angelina Tomas	LIC68320	2026-10-02	+34979 803 944
10	10	10	EMP1009	Leopoldo Torrijos	LIC70144	2026-05-13	+34 924037998

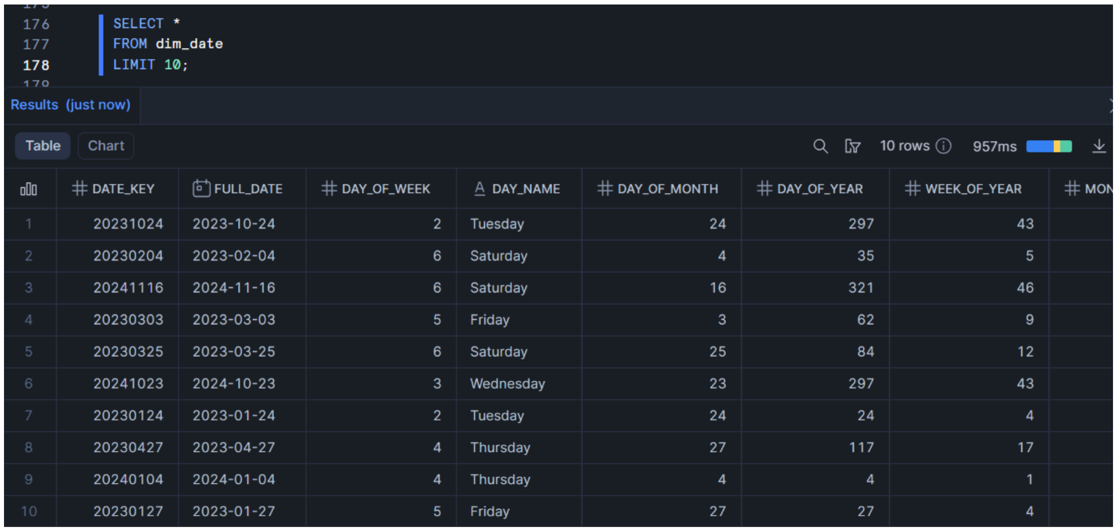


```sql
SELECT *
FROM fact_deliveries
LIMIT 10;
```

Results (just now)

Table Chart 10 rows 113ms

	DELIVERY_KEY	DATE_KEY	SCHEDULED_TIME_KEY	DELIVERED_TIME_KEY	VEHICLE_KEY	DRIVER_KEY	ROUTE_K
1	1	20231216	605	621	97	215	
2	2	20231216	605	621	97	715	
3	3	20231216	648	651	97	215	
4	4	20231216	648	651	97	715	
5	5	20231216	731	737	97	215	
6	6	20231216	731	737	97	715	
7	7	20231216	813	822	97	215	
8	8	20231216	813	822	97	715	
9	9	20231216	856	922	97	215	
10	10	20231216	856	922	97	715	

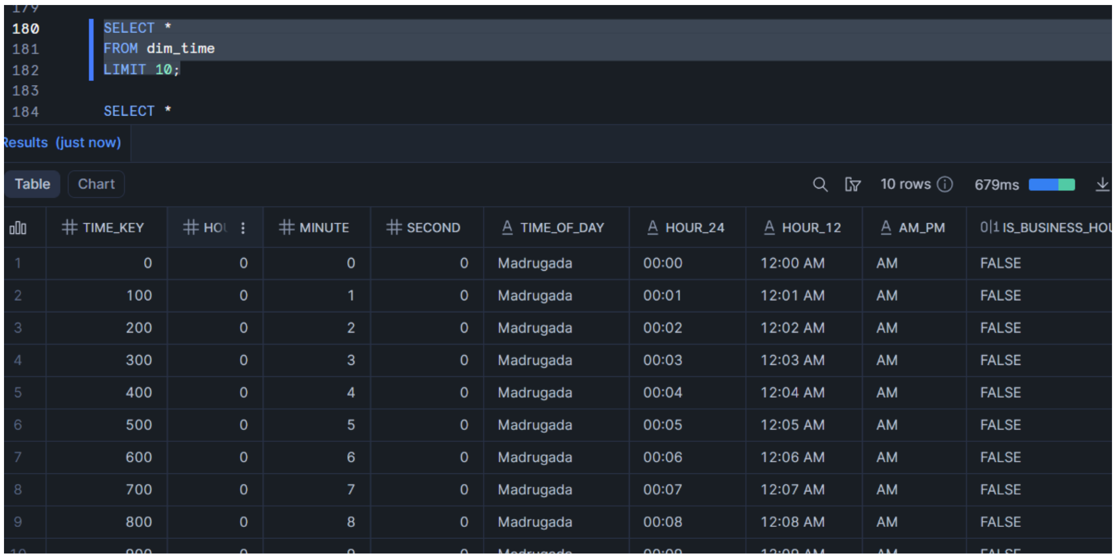


```sql
SELECT *
FROM dim_date
LIMIT 10;
```

Results (just now)

Table Chart 10 rows 957ms

	DATE_KEY	FULL_DATE	DAY_OF_WEEK	DAY_NAME	DAY_OF_MONTH	DAY_OF_YEAR	WEEK_OF_YEAR	MON
1	20231024	2023-10-24	2	Tuesday	24	297	43	
2	20230204	2023-02-04	6	Saturday	4	35	5	
3	20241116	2024-11-16	6	Saturday	16	321	46	
4	20230303	2023-03-03	5	Friday	3	62	9	
5	20230325	2023-03-25	6	Saturday	25	84	12	
6	20241023	2024-10-23	3	Wednesday	23	297	43	
7	20230124	2023-01-24	2	Tuesday	24	24	4	
8	20230427	2023-04-27	4	Thursday	27	117	17	
9	20240104	2024-01-04	4	Thursday	4	4	1	
10	20230127	2023-01-27	5	Friday	27	27	4	

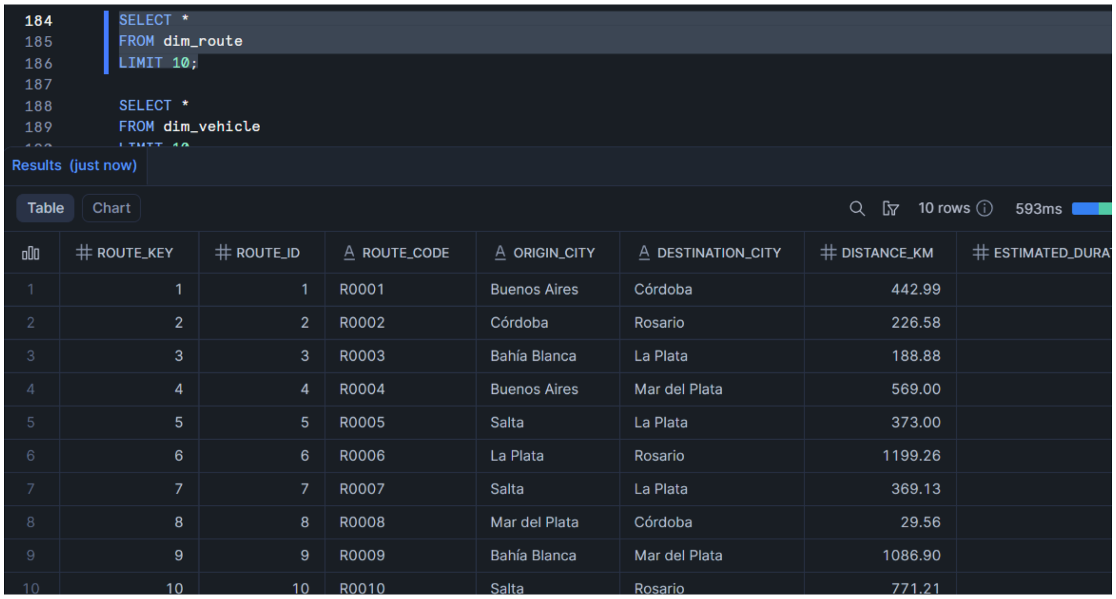


```
SELECT *
FROM dim_time
LIMIT 10;

SELECT *
```

Results (just now)

Table Chart 10 rows 679ms

	TIME_KEY	HO	MINUTE	SECOND	TIME_OF_DAY	HOUR_24	HOUR_12	AM_PM	IS_BUSINESS_HOU
1	0	0	0	0	Madrugada	00:00	12:00 AM	AM	FALSE
2	100	0	1	0	Madrugada	00:01	12:01 AM	AM	FALSE
3	200	0	2	0	Madrugada	00:02	12:02 AM	AM	FALSE
4	300	0	3	0	Madrugada	00:03	12:03 AM	AM	FALSE
5	400	0	4	0	Madrugada	00:04	12:04 AM	AM	FALSE
6	500	0	5	0	Madrugada	00:05	12:05 AM	AM	FALSE
7	600	0	6	0	Madrugada	00:06	12:06 AM	AM	FALSE
8	700	0	7	0	Madrugada	00:07	12:07 AM	AM	FALSE
9	800	0	8	0	Madrugada	00:08	12:08 AM	AM	FALSE
10	900	0	9	0	Madrugada	00:09	12:09 AM	AM	FALSE

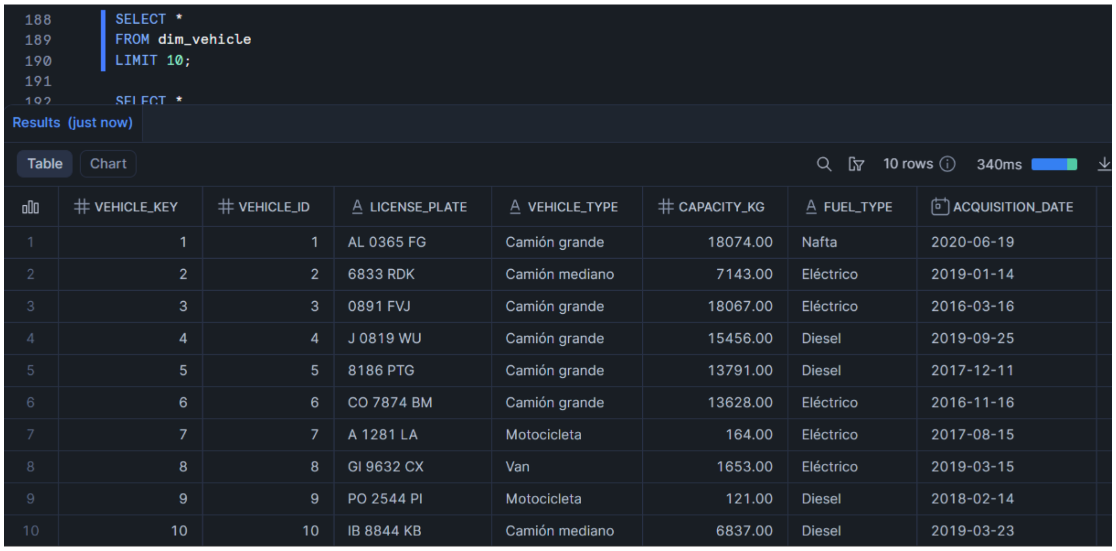


```
SELECT *
FROM dim_route
LIMIT 10;

SELECT *
FROM dim_vehicle
LIMIT 10
```

Results (just now)

Table Chart 10 rows 593ms

	ROUTE_KEY	ROUTE_ID	ROUTE_CODE	ORIGIN_CITY	DESTINATION_CITY	DISTANCE_KM	ESTIMATED_DURAT
1	1	1	R0001	Buenos Aires	Córdoba	442.99	
2	2	2	R0002	Córdoba	Rosario	226.58	
3	3	3	R0003	Bahía Blanca	La Plata	188.88	
4	4	4	R0004	Buenos Aires	Mar del Plata	569.00	
5	5	5	R0005	Salta	La Plata	373.00	
6	6	6	R0006	La Plata	Rosario	1199.26	
7	7	7	R0007	Salta	La Plata	369.13	
8	8	8	R0008	Mar del Plata	Córdoba	29.56	
9	9	9	R0009	Bahía Blanca	Mar del Plata	1086.90	
10	10	10	R0010	Salta	Rosario	771.21	

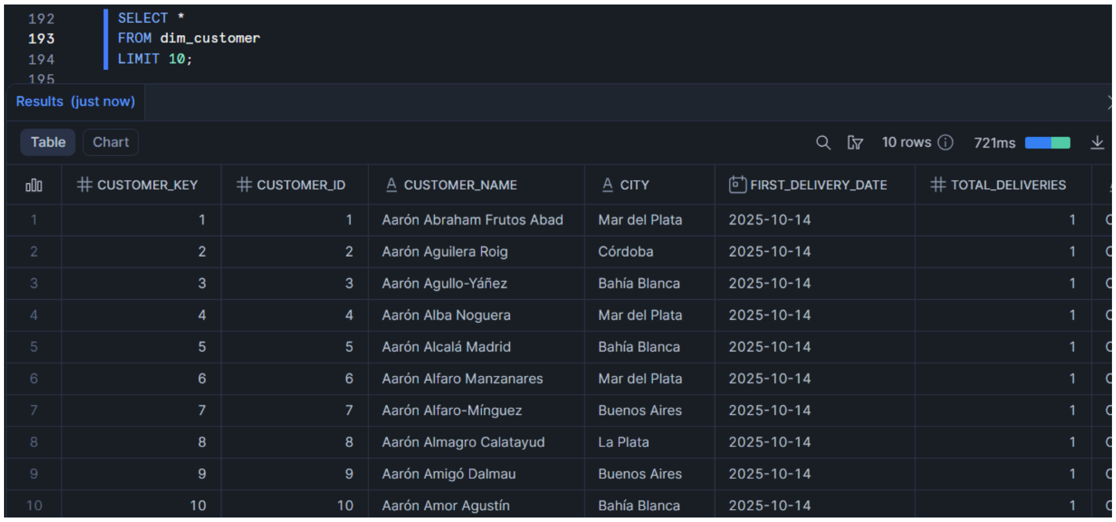


```
SELECT *
FROM dim_vehicle
LIMIT 10;

SELECT *
```

Results (just now)

Table Chart 10 rows 340ms

	VEHICLE_KEY	VEHICLE_ID	LICENSE_PLATE	VEHICLE_TYPE	CAPACITY_KG	FUEL_TYPE	ACQUISITION_DATE
1	1	1	AL 0365 FG	Camión grande	18074.00	Nafta	2020-06-19
2	2	2	6833 RDK	Camión mediano	7143.00	Eléctrico	2019-01-14
3	3	3	0891 FVJ	Camión grande	18067.00	Eléctrico	2016-03-16
4	4	4	J 0819 WU	Camión grande	15456.00	Diesel	2019-09-25
5	5	5	8186 PTG	Camión grande	13791.00	Diesel	2017-12-11
6	6	6	CO 7874 BM	Camión grande	13628.00	Eléctrico	2016-11-16
7	7	7	A 1281 LA	Motocicleta	164.00	Eléctrico	2017-08-15
8	8	8	GI 9632 CX	Van	1653.00	Eléctrico	2019-03-15
9	9	9	PO 2544 PI	Motocicleta	121.00	Diesel	2018-02-14
10	10	10	IB 8844 KB	Camión mediano	6837.00	Diesel	2019-03-23



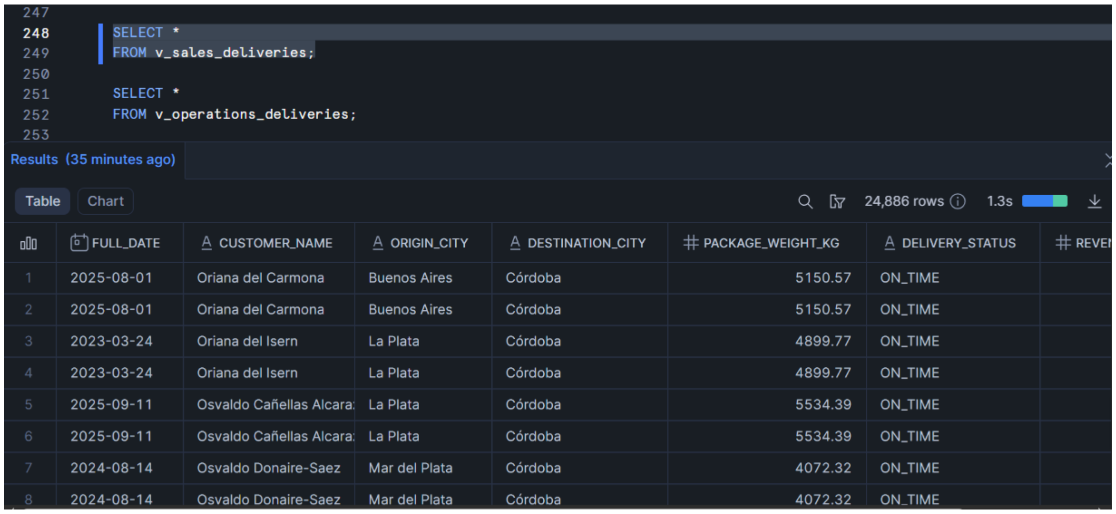
```
SELECT *
FROM dim_customer
LIMIT 10;
```

Results (just now)

Table Chart 10 rows 721ms

	CUSTOMER_KEY	CUSTOMER_ID	CUSTOMER_NAME	CITY	FIRST_DELIVERY_DATE	TOTAL_DELIVERIES
1	1	1	Aarón Abraham Frutos Abad	Mar del Plata	2025-10-14	1
2	2	2	Aarón Aguilera Roig	Córdoba	2025-10-14	1
3	3	3	Aarón Agullo-Yáñez	Bahía Blanca	2025-10-14	1
4	4	4	Aarón Alba Noguera	Mar del Plata	2025-10-14	1
5	5	5	Aarón Alcalá Madrid	Bahía Blanca	2025-10-14	1
6	6	6	Aarón Alfaro Manzanares	Mar del Plata	2025-10-14	1
7	7	7	Aarón Alfaro-Mínguez	Buenos Aires	2025-10-14	1
8	8	8	Aarón Almagro Calatayud	La Plata	2025-10-14	1
9	9	9	Aarón Amigó Dalmau	Buenos Aires	2025-10-14	1
10	10	10	Aarón Amor Agustín	Bahía Blanca	2025-10-14	1

Y las vistas

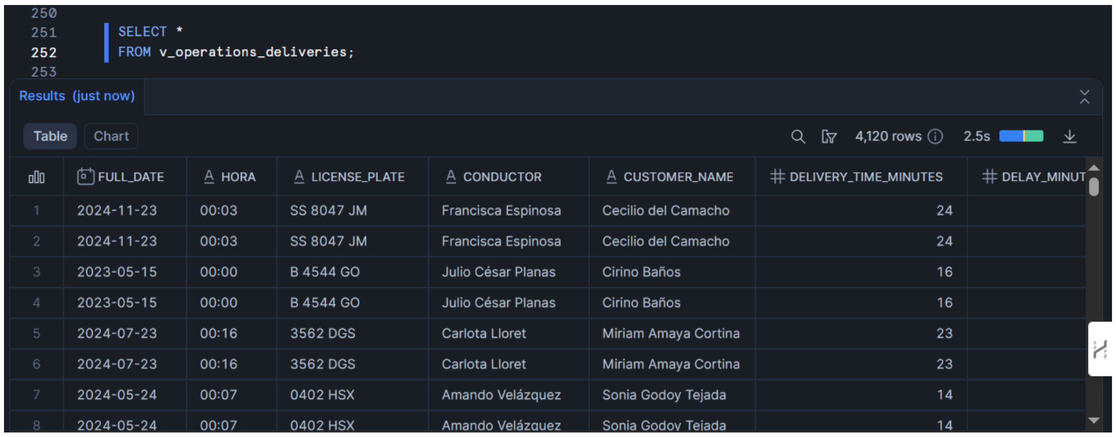


```
SELECT *
FROM v_sales_deliveries;

SELECT *
FROM v_operations_deliveries;
```

Results (35 minutes ago)

Table Chart 24,886 rows 1.3s

	FULL_DATE	CUSTOMER_NAME	ORIGIN_CITY	DESTINATION_CITY	PACKAGE_WEIGHT_KG	DELIVERY_STATUS
1	2025-08-01	Oriana del Carmona	Buenos Aires	Córdoba	5150.57	ON_TIME
2	2025-08-01	Oriana del Carmona	Buenos Aires	Córdoba	5150.57	ON_TIME
3	2023-03-24	Oriana del Isern	La Plata	Córdoba	4899.77	ON_TIME
4	2023-03-24	Oriana del Isern	La Plata	Córdoba	4899.77	ON_TIME
5	2025-09-11	Osvaldo Cañellas Alcara	La Plata	Córdoba	5534.39	ON_TIME
6	2025-09-11	Osvaldo Cañellas Alcara	La Plata	Córdoba	5534.39	ON_TIME
7	2024-08-14	Osvaldo Donaire-Saez	Mar del Plata	Córdoba	4072.32	ON_TIME
8	2024-08-14	Osvaldo Donaire-Saez	Mar del Plata	Córdoba	4072.32	ON_TIME



```
SELECT *
FROM v_operations_deliveries;
```

Results (just now)

Table Chart 4,120 rows 2.5s

	FULL_DATE	HORA	LICENSE_PLATE	CONDUCTOR	CUSTOMER_NAME	DELIVERY_TIME_MINUTES	DELAY_MINUT
1	2024-11-23	00:03	SS 8047 JM	Francisca Espinosa	Cecilio del Camacho	24	
2	2024-11-23	00:03	SS 8047 JM	Francisca Espinosa	Cecilio del Camacho	24	
3	2023-05-15	00:00	B 4544 GO	Julio César Planas	Cirino Baños	16	
4	2023-05-15	00:00	B 4544 GO	Julio César Planas	Cirino Baños	16	
5	2024-07-23	00:16	3562 DGS	Carlota Lloret	Miriam Amaya Cortina	23	
6	2024-07-23	00:16	3562 DGS	Carlota Lloret	Miriam Amaya Cortina	23	
7	2024-05-24	00:07	0402 HSX	Amando Velázquez	Sonia Godoy Tejada	14	
8	2024-05-24	00:07	0402 HSX	Amando Velázquez	Sonia Godoy Tejada	14	

Ademas ejecutamos el script para la automatización de la ingesta diaria