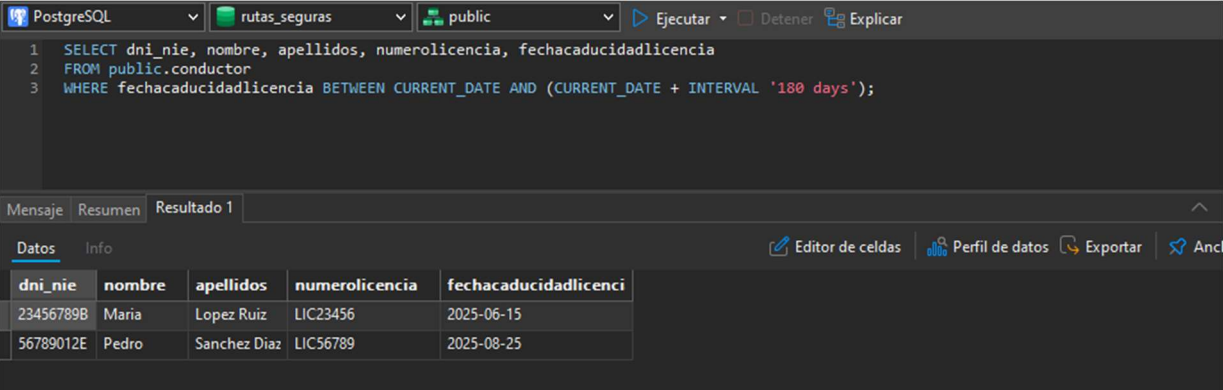


Tarea 2.2: Consultas básicas:

- Escribir consultas SQL para recuperar datos específicos utilizando cláusulas WHERE, ORDER BY, GROUP BY y HAVING.
- Utilizar funciones de agregación (COUNT, SUM, AVG, etc.) para calcular estadísticas.

I. Consultas básicas con WHERE, estas consultas filtran datos según condiciones específicas.

a) Obtener todos los conductores cuya licencia caduque en los próximos 180 días:



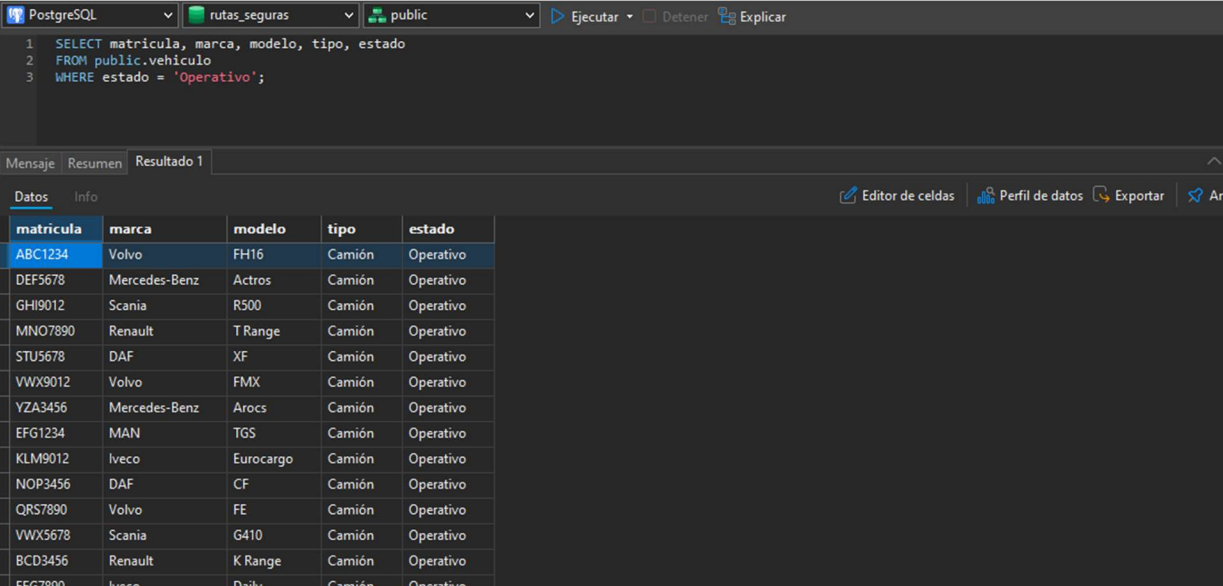
The screenshot shows the PostgreSQL query editor with the following query:

```
1 SELECT dni_nie, nombre, apellidos, numerolicencia, fechacaducidadlicencia
2 FROM public.conductor
3 WHERE fechacaducidadlicencia BETWEEN CURRENT_DATE AND (CURRENT_DATE + INTERVAL '180 days');
```

The results are displayed in a table with the following data:

dni_nie	nombre	apellidos	numerolicencia	fechacaducidadlicenci
23456789B	Maria	Lopez Ruiz	LIC23456	2025-06-15
56789012E	Pedro	Sanchez Diaz	LIC56789	2025-08-25

b) Obtener vehículos operativos (estado = 'Operativo'):



The screenshot shows the PostgreSQL query editor with the following query:

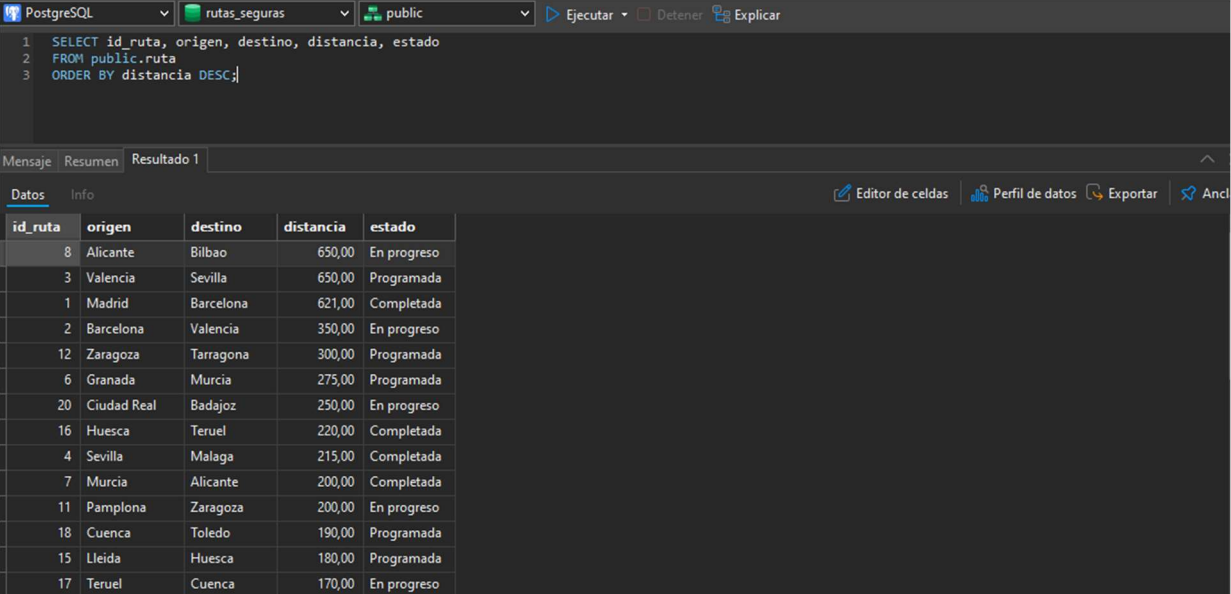
```
1 SELECT matricula, marca, modelo, tipo, estado
2 FROM public.vehiculo
3 WHERE estado = 'Operativo';
```

The results are displayed in a table with the following data:

matricula	marca	modelo	tipo	estado
ABC1234	Volvo	FH16	Camión	Operativo
DEF5678	Mercedes-Benz	Actros	Camión	Operativo
GHI9012	Scania	R500	Camión	Operativo
MNO7890	Renault	T Range	Camión	Operativo
STU5678	DAF	XF	Camión	Operativo
VWX9012	Volvo	FMX	Camión	Operativo
YZA3456	Mercedes-Benz	Arocs	Camión	Operativo
EFG1234	MAN	TGS	Camión	Operativo
KLM9012	Iveco	Eurocargo	Camión	Operativo
NOP3456	DAF	CF	Camión	Operativo
QRS7890	Volvo	FE	Camión	Operativo
VWX5678	Scania	G410	Camión	Operativo
BCD3456	Renault	K Range	Camión	Operativo
EFG7890	Iveco	Daily	Camión	Operativo

II. Consultas con ORDER BY, estas consultas ordenan los resultados según un criterio específico.

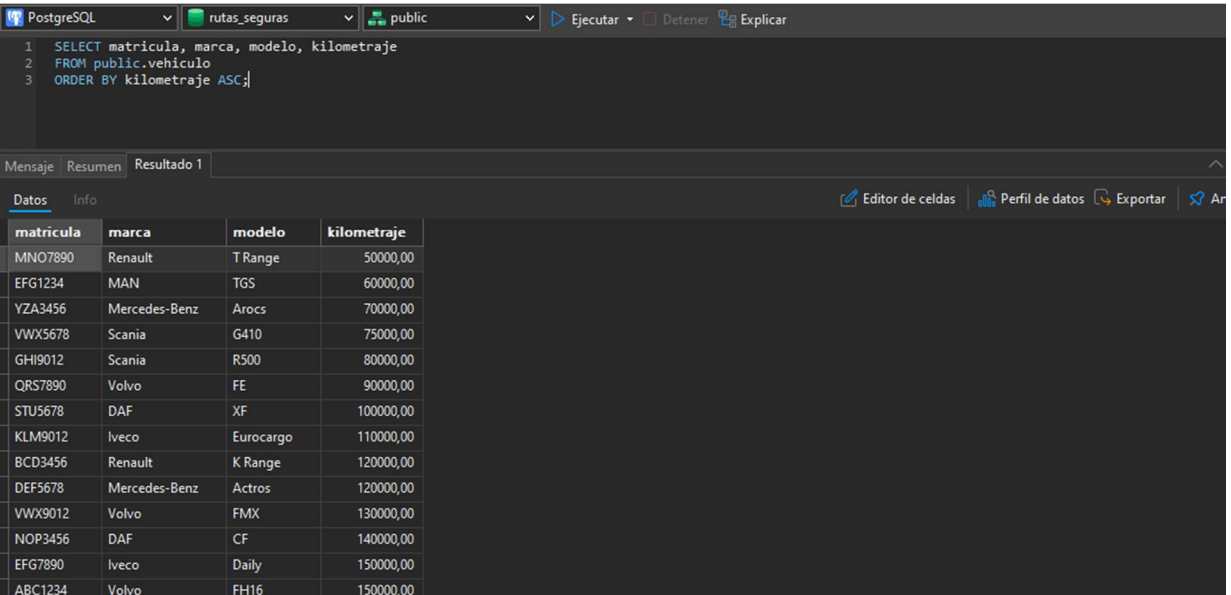
a) Listar todas las rutas ordenadas por distancia de mayor a menor:



```
1 SELECT id_ruta, origen, destino, distancia, estado
2 FROM public.ruta
3 ORDER BY distancia DESC;
```

id_ruta	origen	destino	distancia	estado
8	Alicante	Bilbao	650,00	En progreso
3	Valencia	Sevilla	650,00	Programada
1	Madrid	Barcelona	621,00	Completada
2	Barcelona	Valencia	350,00	En progreso
12	Zaragoza	Tarragona	300,00	Programada
6	Granada	Murcia	275,00	Programada
20	Ciudad Real	Badajoz	250,00	En progreso
16	Huesca	Teruel	220,00	Completada
4	Sevilla	Malaga	215,00	Completada
7	Murcia	Alicante	200,00	Completada
11	Pamplona	Zaragoza	200,00	En progreso
18	Cuenca	Toledo	190,00	Programada
15	Lleida	Huesca	180,00	Programada
17	Teruel	Cuenca	170,00	En progreso

b) Mostrar los vehículos ordenados por kilometraje de menor a mayor:

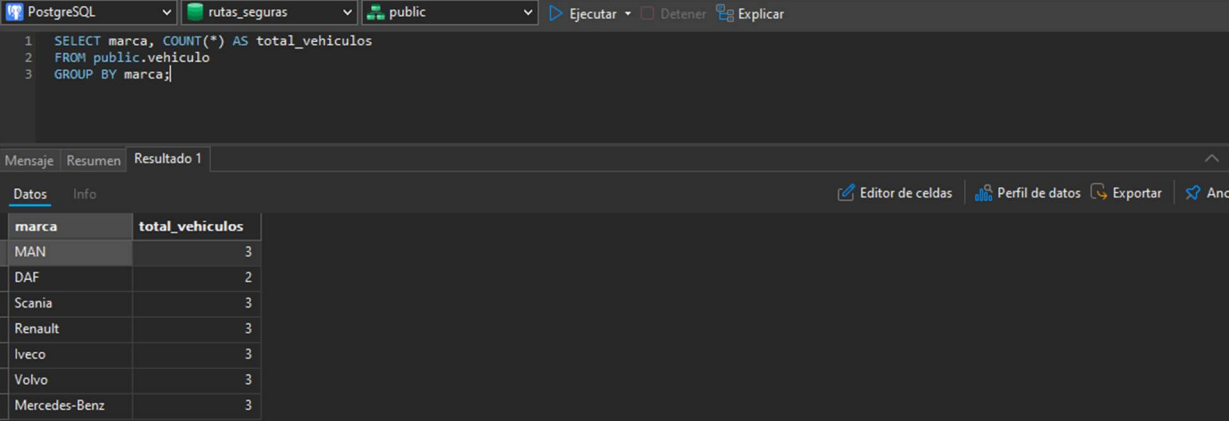


```
1 SELECT matricula, marca, modelo, kilometraje
2 FROM public.vehiculo
3 ORDER BY kilometraje ASC;
```

matricula	marca	modelo	kilometraje
MNO7890	Renault	T Range	50000,00
EFG1234	MAN	TGS	60000,00
YZA3456	Mercedes-Benz	Arocs	70000,00
VWX5678	Scania	G410	75000,00
GHI9012	Scania	R500	80000,00
QRS7890	Volvo	FE	90000,00
STU5678	DAF	XF	100000,00
KLM9012	Iveco	Eurocargo	110000,00
BCD3456	Renault	K Range	120000,00
DEF5678	Mercedes-Benz	Actros	120000,00
VWX9012	Volvo	FMX	130000,00
NOP3456	DAF	CF	140000,00
EFG7890	Iveco	Daily	150000,00
ABC1234	Volvo	FH16	150000,00

III. Consultas con GROUP BY y funciones de agregación, estas consultas agrupan datos y calculan estadísticas.

a) Contar el número de vehículos por marca:



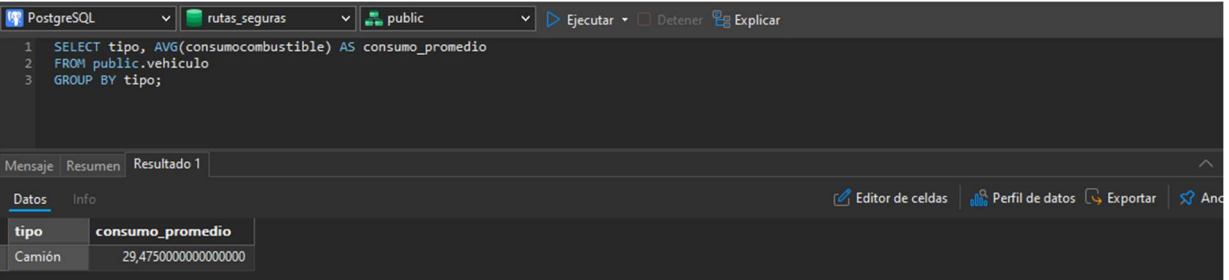
The screenshot shows a PostgreSQL query editor with the following SQL query:

```
1 SELECT marca, COUNT(*) AS total_vehiculos
2 FROM public.vehiculo
3 GROUP BY marca;
```

The results are displayed in a table with two columns: **marca** and **total_vehiculos**.

marca	total_vehiculos
MAN	3
DAF	2
Scania	3
Renault	3
Iveco	3
Volvo	3
Mercedes-Benz	3

b) Calcular el consumo promedio de combustible por tipo de vehículo:



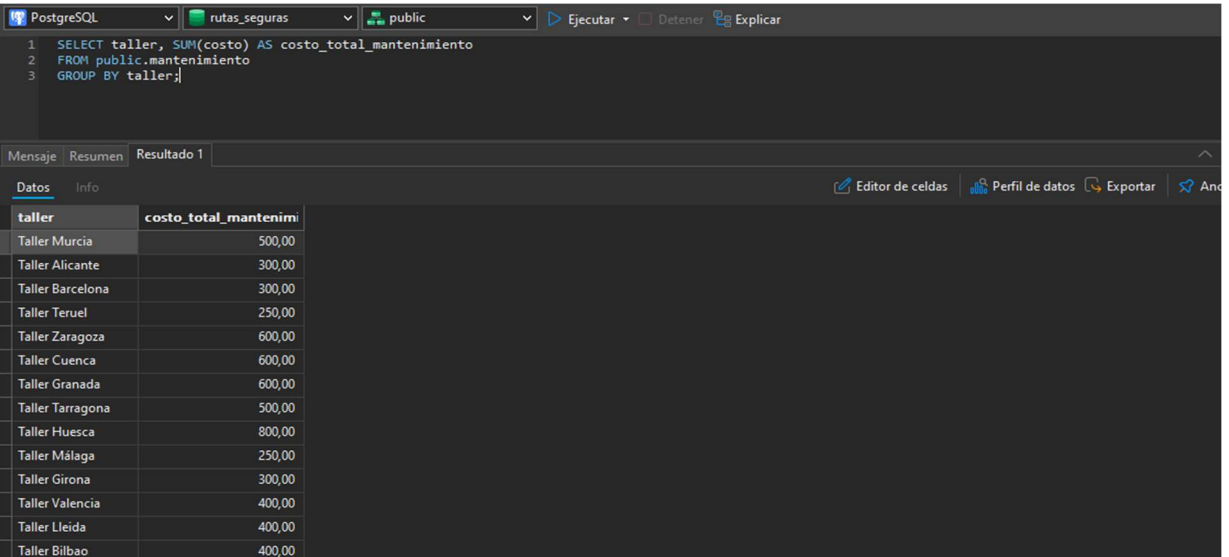
The screenshot shows a PostgreSQL query editor with the following SQL query:

```
1 SELECT tipo, AVG(consumocombustible) AS consumo_promedio
2 FROM public.vehiculo
3 GROUP BY tipo;
```

The results are displayed in a table with two columns: **tipo** and **consumo_promedio**.

tipo	consumo_promedio
Camión	29,4750000000000000

c) Calcular el costo total de mantenimiento por taller:



The screenshot shows a PostgreSQL query editor with the following SQL query:

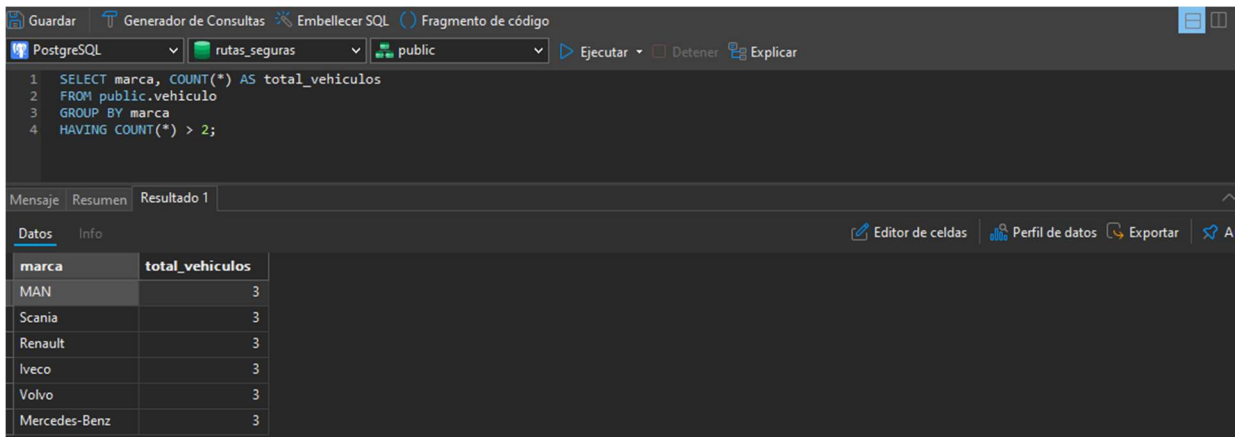
```
1 SELECT taller, SUM(costo) AS costo_total_mantenimiento
2 FROM public.mantenimiento
3 GROUP BY taller;
```

The results are displayed in a table with two columns: **taller** and **costo_total_mantenimi**.

taller	costo_total_mantenimi
Taller Murcia	500,00
Taller Alicante	300,00
Taller Barcelona	300,00
Taller Teruel	250,00
Taller Zaragoza	600,00
Taller Cuenca	600,00
Taller Granada	600,00
Taller Tarragona	500,00
Taller Huesca	800,00
Taller Málaga	250,00
Taller Girona	300,00
Taller Valencia	400,00
Taller Lleida	400,00
Taller Bilbao	400,00

IV. Consultas con HAVING, estas consultas filtran grupos basados en condiciones.

a) Obtener marcas de vehículos con más de 2 unidades:



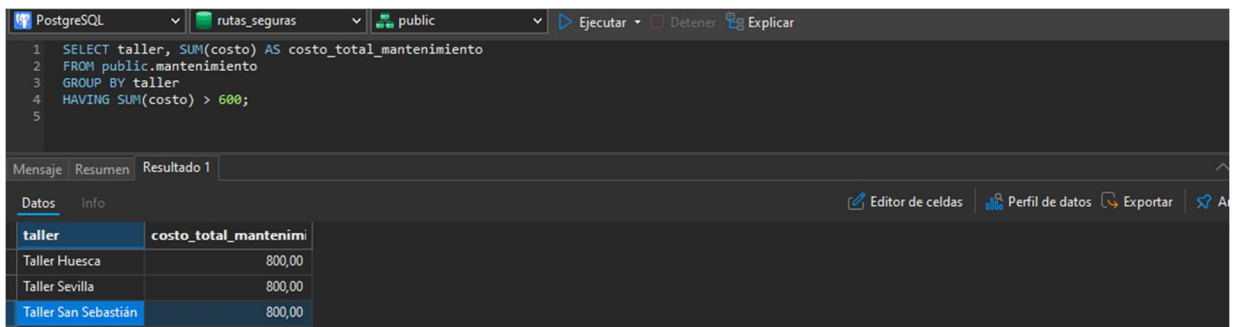
The screenshot shows a PostgreSQL query editor with the following SQL query:

```
1 SELECT marca, COUNT(*) AS total_vehiculos
2 FROM public.vehiculo
3 GROUP BY marca
4 HAVING COUNT(*) > 2;
```

The results are displayed in a table with the following data:

marca	total_vehiculos
MAN	3
Scania	3
Renault	3
Iveco	3
Volvo	3
Mercedes-Benz	3

b) Mostrar los talleres donde el costo total de mantenimiento sea mayor a 600:



The screenshot shows a PostgreSQL query editor with the following SQL query:

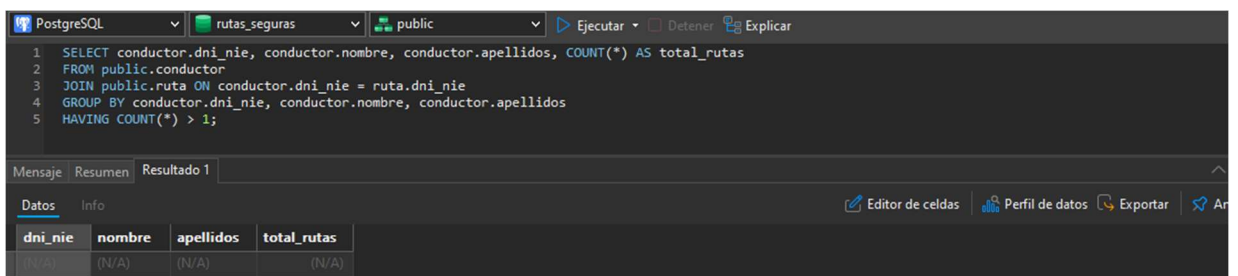
```
1 SELECT taller, SUM(costo) AS costo_total_mantenimiento
2 FROM public.mantenimiento
3 GROUP BY taller
4 HAVING SUM(costo) > 600;
```

The results are displayed in a table with the following data:

taller	costo_total_mantenimi
Taller Huesca	800,00
Taller Sevilla	800,00
Taller San Sebastián	800,00

V. Consultas combinando múltiples cláusulas, estas consultas combinan varias cláusulas para obtener resultados más complejos.

a) Listar los conductores con más de 1 ruta asignada:



The screenshot shows a PostgreSQL query editor with the following SQL query:

```
1 SELECT conductor.dni_nie, conductor.nombre, conductor.apellidos, COUNT(*) AS total_rutas
2 FROM public.conductor
3 JOIN public.ruta ON conductor.dni_nie = ruta.dni_nie
4 GROUP BY conductor.dni_nie, conductor.nombre, conductor.apellidos
5 HAVING COUNT(*) > 1;
```

The results are displayed in a table with the following data:

dni_nie	nombre	apellidos	total_rutas
(N/A)	(N/A)	(N/A)	(N/A)

b) Obtener los vehículos que han consumido más de 500 litros de combustible:

PostgreSQL rutas_seguras public Ejecutar Detener Explicar

```

1 SELECT matricula, SUM(cantidad) AS total_combustible_consumido
2 FROM public.combustible
3 GROUP BY matricula
4 HAVING SUM(cantidad) > 500;

```

Mensaje Resumen Resultado 1

Datos Info Editor de celdas Perfil de datos Exportar Ancla

matricula	total_combustible_con
MNO7890	550,00
EFG1234	530,00
KLM9012	540,00
STU5678	520,00
GHI9012	600,00
QRS7890	505,00
VWX5678	525,00
YZA3456	510,00
BCD3456	515,00

6. Consultas con funciones de agregación, estas consultas utilizan funciones de agregación para calcular estadísticas más detalladas.

a) Calcular el kilometraje promedio de los vehículos operativos:

Guardar Generador de Consultas Embellecer SQL Fragmento de código PostgreSQL rutas_seguras public Ejecutar Detener Explicar

```

1 SELECT AVG(kilometraje) AS kilometraje_promedio
2 FROM public.vehiculo
3 WHERE estado = 'Operativo';

```

Mensaje Resumen Resultado 1

Datos Info Editor de celdas Perfil de datos Exportar Ancla

kilometraje_promedio
103214,285714285714

b) Obtener el costo máximo y mínimo de mantenimiento por tipo de mantenimiento:

Guardar Generador de Consultas Embellecer SQL Fragmento de código PostgreSQL rutas_seguras public Ejecutar Detener Explicar

```

1 SELECT tipomantenimiento, MAX(costo) AS costo_maximo, MIN(costo) AS costo_minimo
2 FROM public.mantenimiento
3 GROUP BY tipomantenimiento;

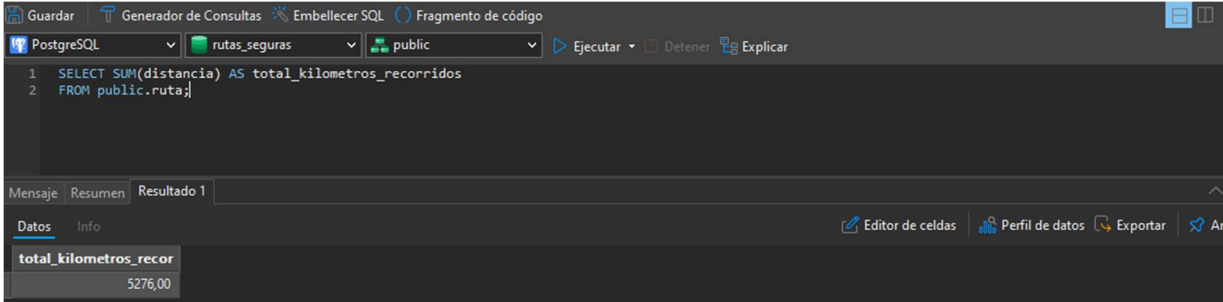
```

Mensaje Resumen Resultado 1

Datos Info Editor de celdas Perfil de datos Exportar Ancla

tipomantenimiento	costo_maximo	costo_minimo
Revisión general	500,00	500,00
Frenos	300,00	300,00
Eléctrico	250,00	250,00
Motor	800,00	800,00
Suspensión	400,00	400,00
Transmisión	600,00	600,00

c) Calcular el total de kilómetros recorridos en todas las rutas:



The screenshot shows a PostgreSQL query editor interface. The top bar includes buttons for 'Guardar', 'Generador de Consultas', 'Embellecer SQL', and 'Fragmento de código'. Below the bar, the database 'rutas_seguras' and schema 'public' are selected. The query editor contains the following SQL code:

```
1 SELECT SUM(distancia) AS total_kilometros_recorridos
2 FROM public.ruta;
```

Below the query editor, the 'Resultado 1' tab is active, displaying the query results in a table format. The table has two columns: 'total_kilometros_recor' and '5276,00'.

total_kilometros_recor
5276,00