Project Title: Migration of Data from a Relational Database to a NoSQL Database

---

Introduction

This project demonstrates the end-to-end process of migrating data from a structured relational database into a flexible NoSQL database using MongoDB. The selected domain is "Vehicle Services," which involves managing real-world data for clients, vehicles, mechanics, and the services performed on each vehicle. The project aims to evaluate both relational and non-relational data models, understand their advantages and limitations, and apply this understanding by implementing a complete migration pipeline using scripting. This experience allowed us to combine theory with practice and to better understand how real applications can move from SQL to NoSQL systems.

Relational Database Design and Data Modeling

We designed a PostgreSQL relational database with five related tables:

- **Clients:** Stores client personal information such as full name, phone number, and email address. Each client can own one or more vehicles.
- **Vehicles:** Contains information about vehicles including plate number, brand, model, and year. Each vehicle is linked to a specific client through a foreign key.
- **Mechanics:** Stores the list of mechanics in the vehicle service center along with their area of specialty.
- **Services:** A catalog of available services such as oil change, brake replacement, diagnostics, etc., each with a price and estimated duration.
- **VehicleServices:** A junction table that tracks services provided to each vehicle. It connects a vehicle, a service, and a mechanic, and includes the service date and optional notes.

The tables are connected using `FOREIGN KEY` constraints to maintain referential integrity. For instance, `Vehicles.ClientID` references `Clients.ClientID`, and `VehicleServices` links to `Vehicles`, `Mechanics`, and `Services`. Data types were selected based on the nature of the data: `SERIAL` for identifiers, `VARCHAR` for strings, `INT` for numeric values, `NUMERIC` for prices, and `DATE` for service dates.

Data Population

After setting up the schema, I populated each table with 15 rows using diverse and realistic data. Client names are in international format, and vehicles include brands such as Volkswagen, Opel, BMW, Toyota, and others. The services cover a wide range of automotive maintenance activities, and mechanics are associated with different specializations. Data was inserted using SQL `INSERT` statements and tested using `SELECT` queries in pgAdmin. Screenshots from pgAdmin showing the contents of each table are provided in the appendix for verification.

Choice of NoSQL Database

I selected **MongoDB** as the NoSQL database for this project. MongoDB is a document-oriented database that stores data in flexible, JSON-like documents. This model aligns perfectly with my project needs, as I want to group related data (e.g., a vehicle and all its services) into a single document.

Other NoSQL options were considered:

- **Redis**, a key-value store, is very fast but lacks the capability to represent complex, nested data relationships.
- **Cassandra**, a wide-column store, is designed for high-volume writes and distributed systems, but it introduces complexity and is less suitable for embedded structures.

MongoDB was chosen due to its:

- Schema flexibility
- Built-in support for embedded documents
- Integration with Python via `pymongo`
- Ease of visualization with MongoDB Compass

These features made it ideal for a project like "Vehicle Services," where I needed to embed services inside vehicles and include client info without relying on JOINs.

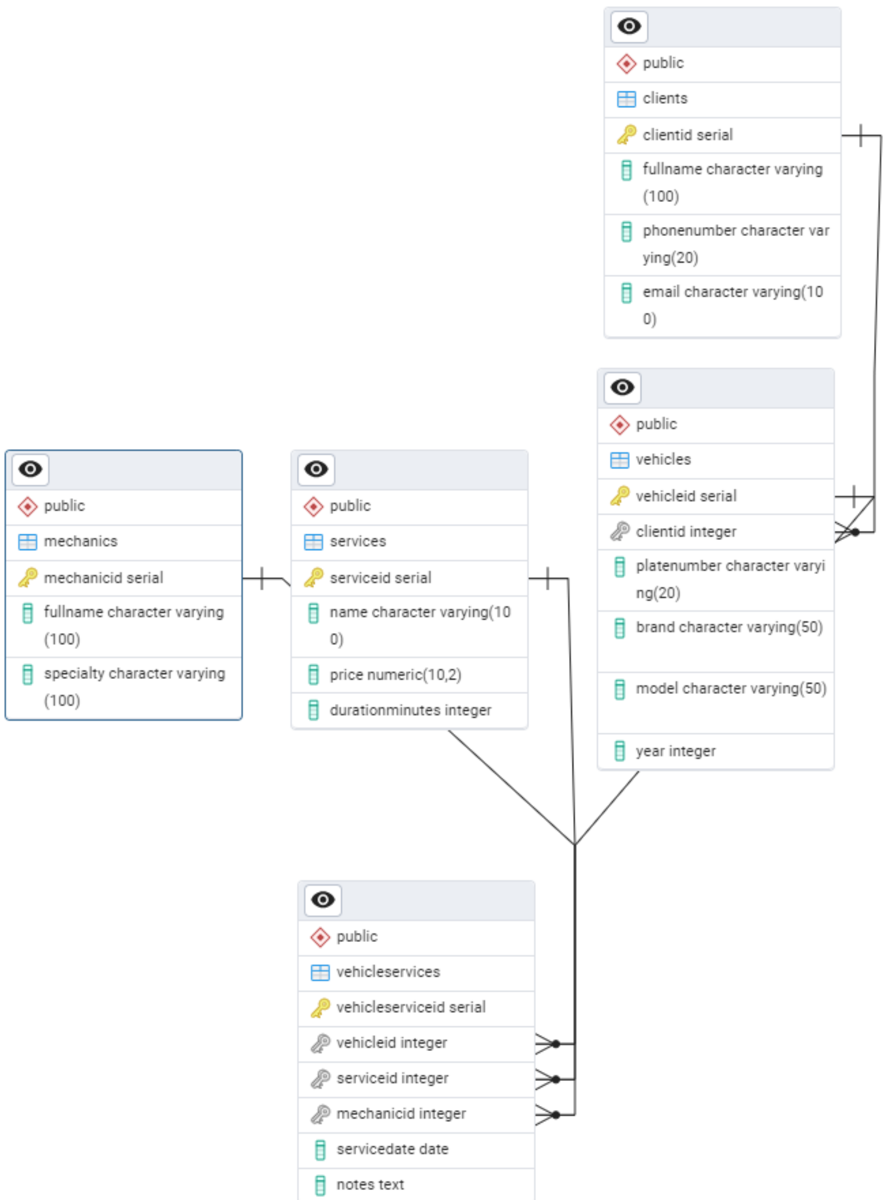
NoSQL Database Modeling

In MongoDB, I used a denormalized schema to simplify data access. Each document in the ***`Vehicles`*** collection represents a single vehicle and contains:

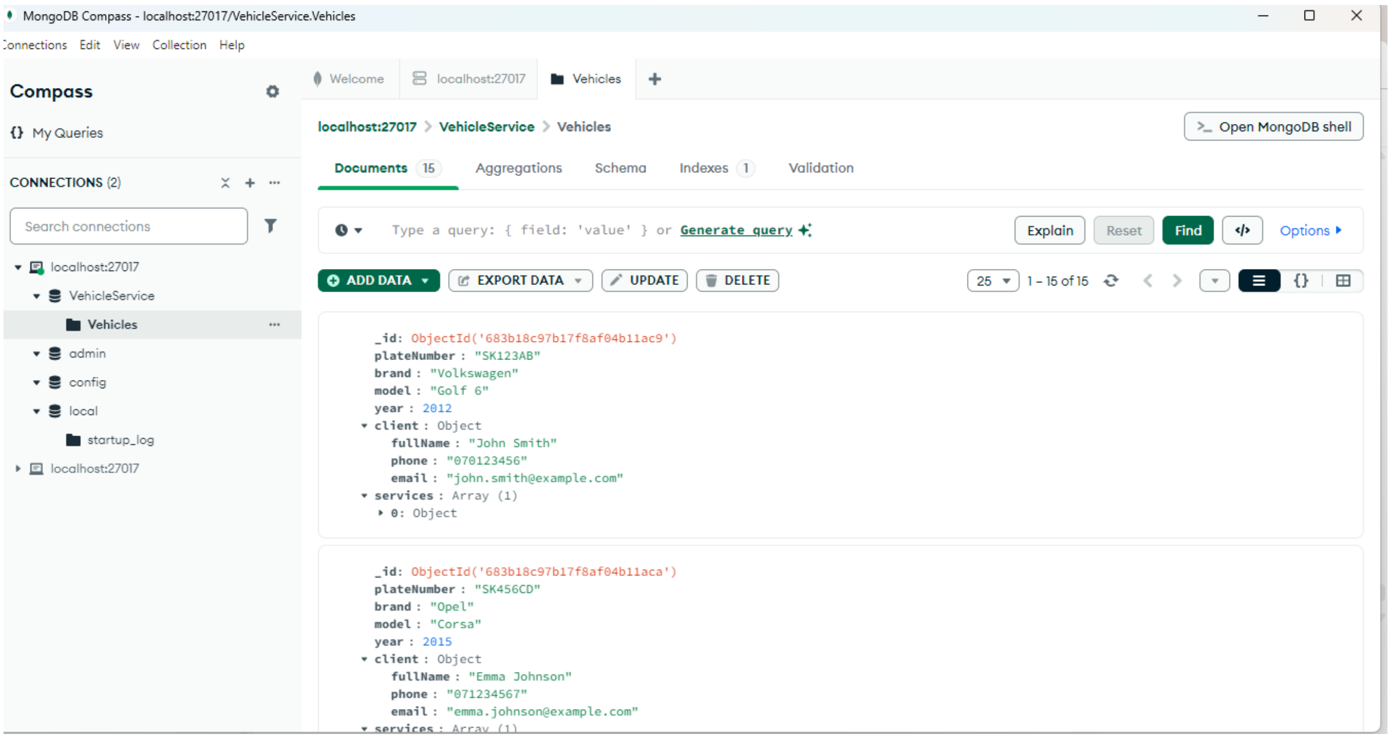
- Vehicle information (plate number, brand, model, year)
- An embedded `client` object (full name, phone, email)
- A `services` array, where each element contains:
  - Service name

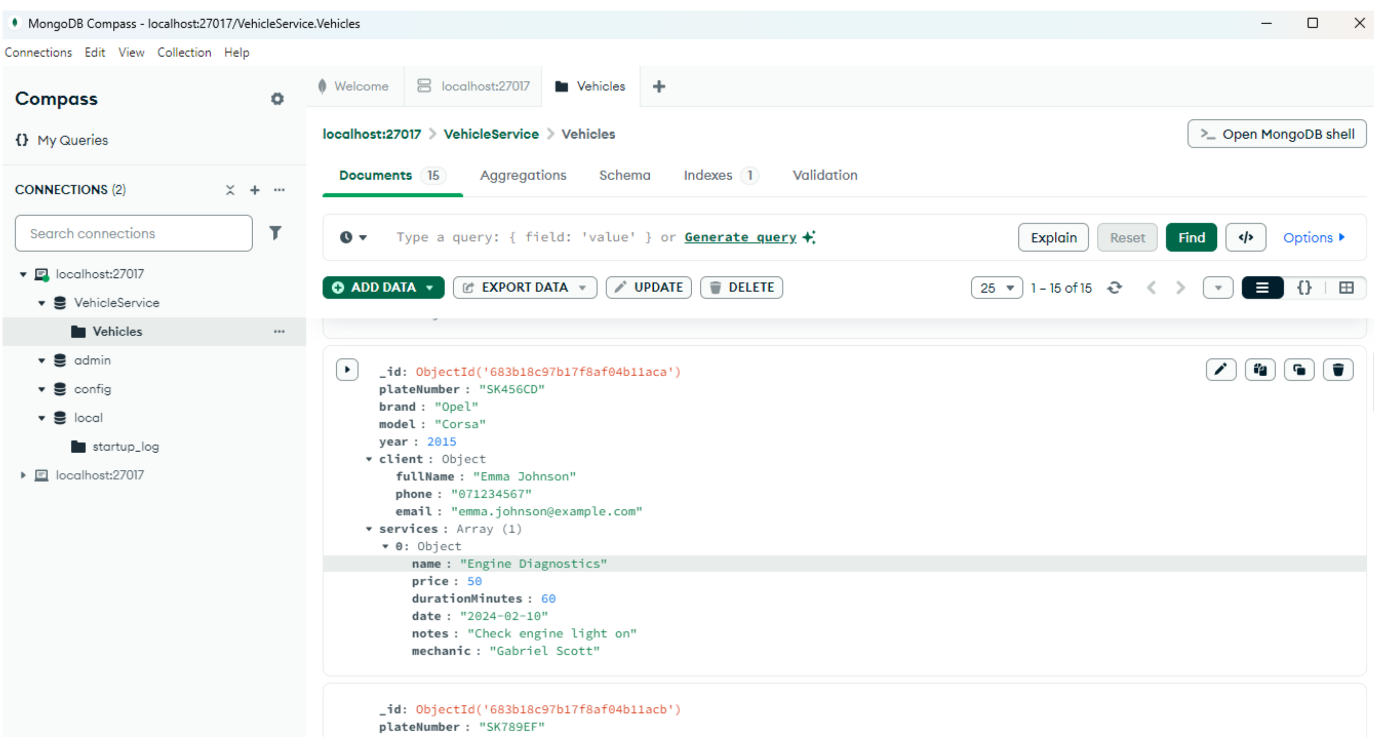
- Price
- Duration in minutes
- Date performed
- Notes
- Mechanic name

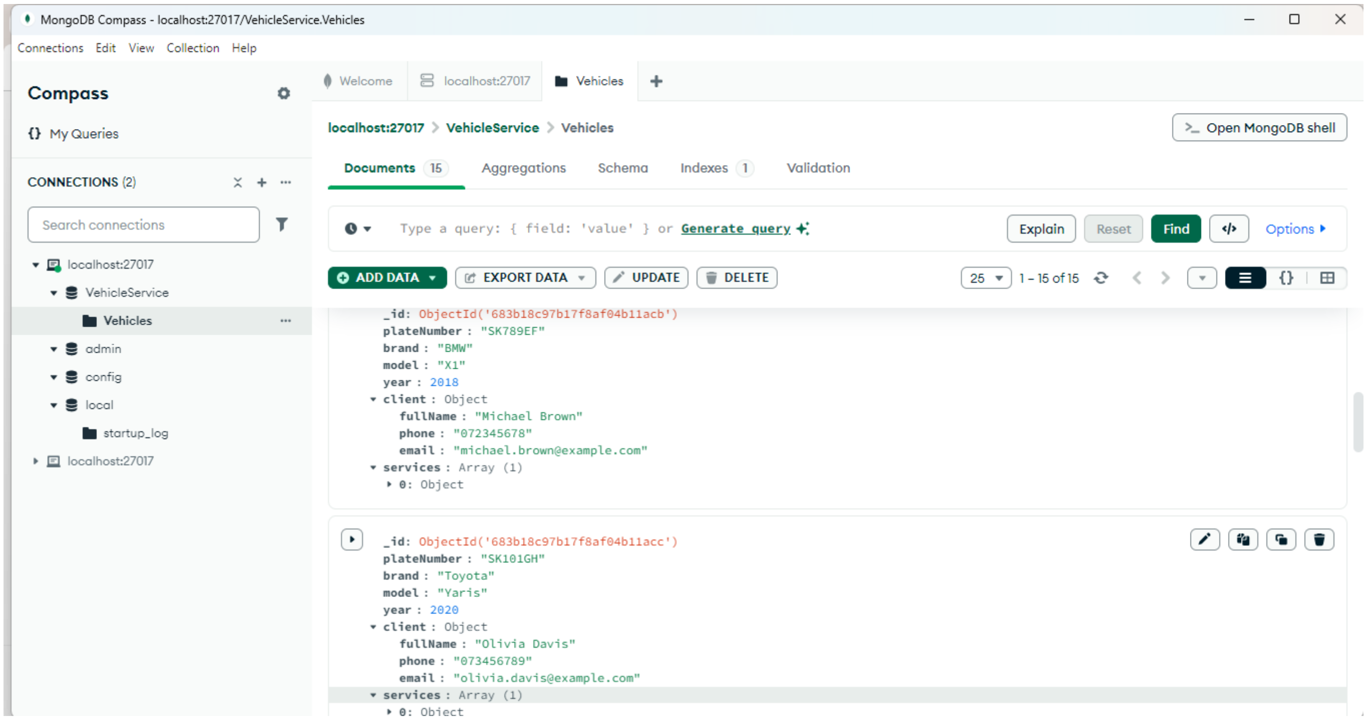
This design avoids the need for joins and reduces query complexity. For example, retrieving a complete service history for a vehicle requires only one query in MongoDB. This structure closely mirrors real-world access patterns and improves performance for common queries.



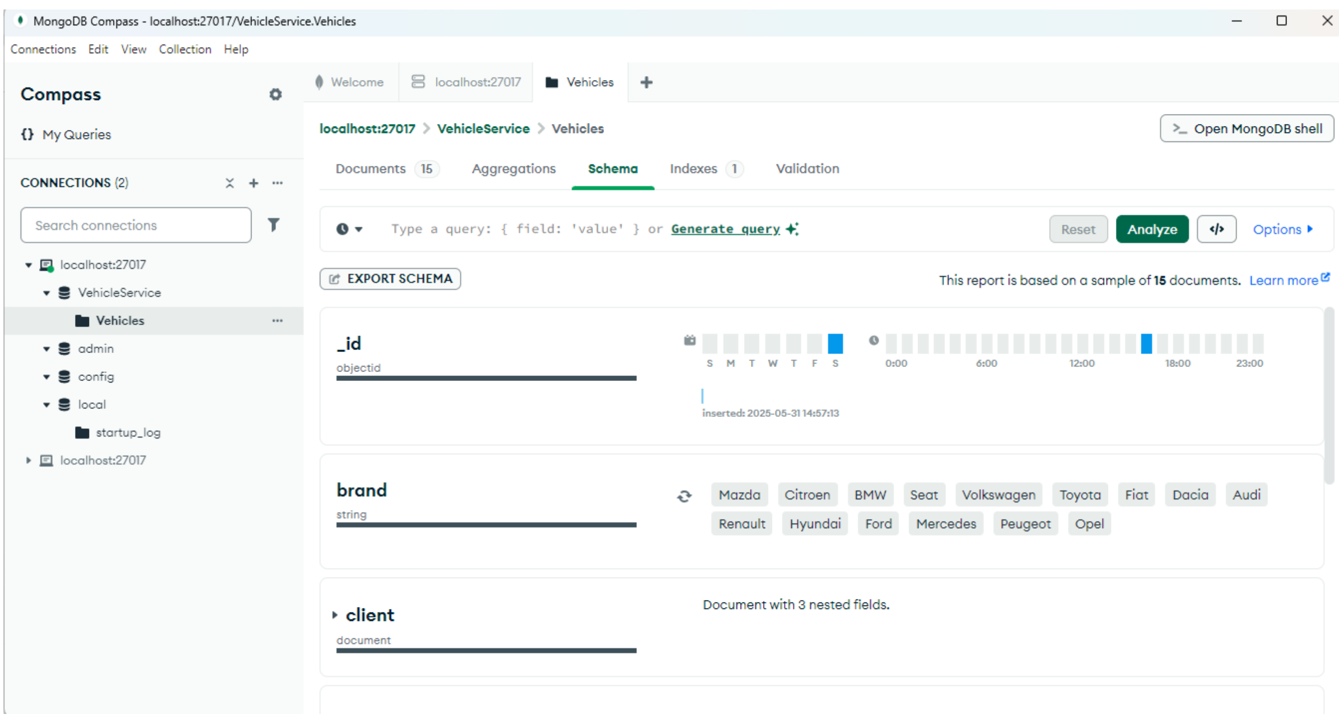
Example of a Document in MongoDB:
The following screenshot from MongoDB Compass shows how a single vehicle document was modeled, including embedded client details and a list of services.



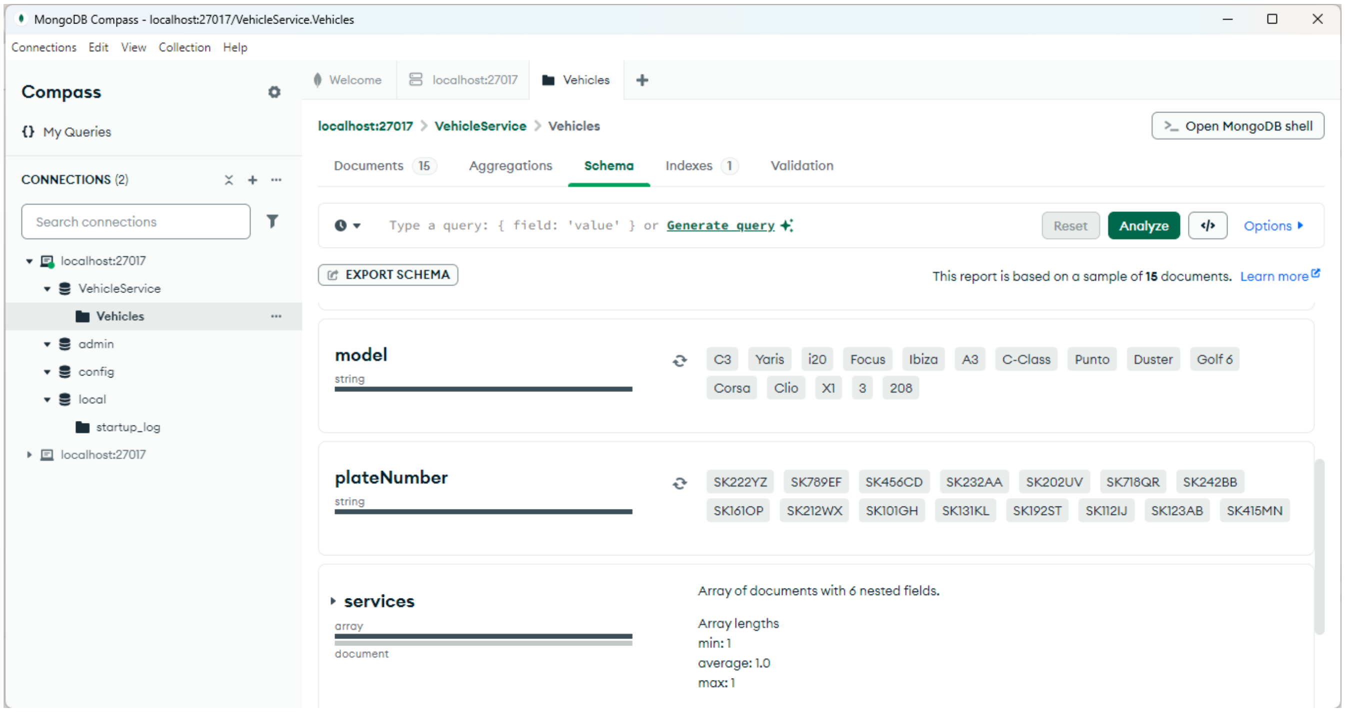


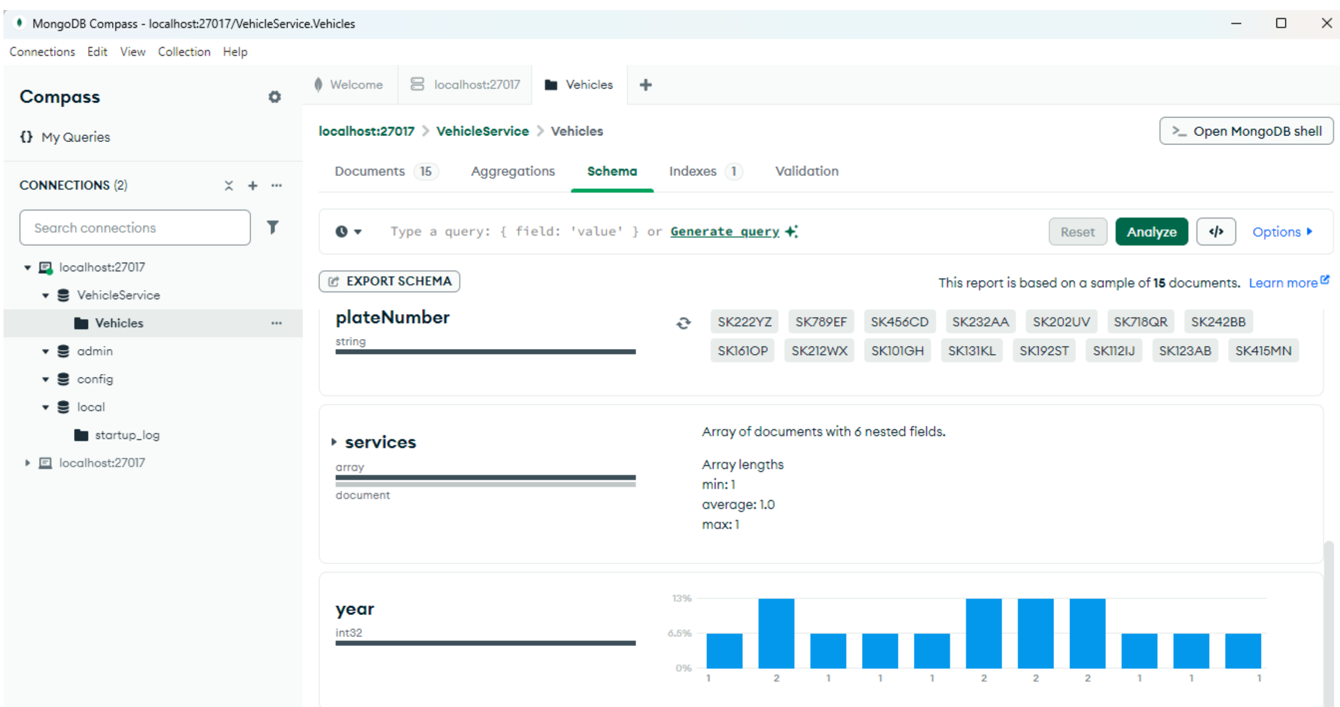


MongoDB Compass - localhost:27017/VehicleService.Vehicles
Connections Edit View Collection Help
Compass
My Queries
CONNECTIONS (2)
Search connections
localhost:27017
VehicleService
Vehicles
admin
config
local
startup_log
localhost:27017
Welcome
localhost:27017
Vehicles
localhost:27017 > VehicleService > Vehicles
Open MongoDB shell
Documents 15
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 – 15 of 15
_id: ObjectId('683b18c97b17f8af04b11acb')
plateNumber : "SK789EF"
brand : "BMW"
model : "X1"
year : 2018
client : Object
fullName : "Michael Brown"
phone : "072345678"
email : "michael.brown@example.com"
services : Array (1)
0: Object
_id: ObjectId('683b18c97b17f8af04b11acc')
plateNumber : "SK101GH"
brand : "Toyota"
model : "Yaris"
year : 2020
client : Object
fullName : "Olivia Davis"
phone : "073456789"
email : "olivia.davis@example.com"
services : Array (1)
0: Object



MongoDB Compass - localhost:27017/VehicleService.Vehicles
Connections Edit View Collection Help
Compass
My Queries
CONNECTIONS (2)
Search connections
localhost:27017
VehicleService
Vehicles
admin
config
local
startup_log
localhost:27017
Welcome
localhost:27017
Vehicles
localhost:27017 > VehicleService > Vehicles
Open MongoDB shell
Documents 15
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Reset
Analyze
Options
EXPORT SCHEMA
This report is based on a sample of 15 documents. Learn more
_id
objectid
S M T W T F S
0:00
6:00
12:00
18:00
23:00
inserted: 2025-05-31 14:57:13
brand
string
Mazda
Citroen
BMW
Seat
Volkswagen
Toyota
Fiat
Dacia
Audi
Renault
Hyundai
Ford
Mercedes
Peugeot
Opel
client
document
Document with 3 nested fields.


MongoDB Compass - localhost:27017/VehicleService.Vehicles
Connections Edit View Collection Help
Compass
My Queries
CONNECTIONS (2)
Search connections
localhost:27017
VehicleService
Vehicles
admin
config
local
startup_log
localhost:27017
Welcome
localhost:27017
Vehicles
localhost:27017 > VehicleService > Vehicles
Open MongoDB shell
Documents 15
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Reset
Analyze
Options
EXPORT SCHEMA
This report is based on a sample of 15 documents. Learn more
model
string
C3 Yaris i20 Focus Ibiza A3 C-Class Punto Duster Golf 6
Corsa Clio X1 3 208
plateNumber
string
SK222YZ SK789EF SK456CD SK232AA SK202UV SK718QR SK242BB
SK161OP SK212WX SK101GH SK131KL SK192ST SK112IJ SK123AB SK415MN
services
array
document
Array of documents with 6 nested fields.
Array lengths
min: 1
average: 1.0
max: 1


MongoDB Compass - localhost:27017/VehicleService.Vehicles
Connections Edit View Collection Help
Compass
My Queries
CONNECTIONS (2)
Search connections
localhost:27017
VehicleService
Vehicles
admin
config
local
startup_log
localhost:27017
Welcome
localhost:27017
Vehicles
localhost:27017 > VehicleService > Vehicles
Open MongoDB shell
Documents 15
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Reset
Analyze
Options
EXPORT SCHEMA
This report is based on a sample of 15 documents. Learn more
plateNumber
string
SK222YZ SK789EF SK456CD SK232AA SK202UV SK718QR SK242BB
SK161OP SK212WX SK101GH SK131KL SK192ST SK112IJ SK123AB SK415MN
services
array
document
Array of documents with 6 nested fields.
Array lengths
min: 1
average: 1.0
max: 1
year
int32
13%
6.5%
0%
1 2 1 1 1 2 2 2 1 1 1

Data Migration Process

To move the data from PostgreSQL to MongoDB, I wrote a Python script in VSC using the following libraries:

- `psycopg2` to connect and fetch data from PostgreSQL
- `pymongo` to insert structured documents into MongoDB

Steps performed in the script:

1. Establish a connection to the PostgreSQL database.
2. Query data from multiple tables using JOINs.
3. Build structured Python dictionaries representing each vehicle with its client and related services.
4. Connect to MongoDB.
5. Insert the structured data into the `Vehicles` collection.

The script is well-commented, with all database connection details, data transformation logic, and insertion steps clearly described. Error handling and data verification (by inspecting MongoDB Compass) were also performed to ensure data quality. The source code is, along with a `README.md` file explaining how to install dependencies and run the migration.

Conclusion

Through this project, I successfully designed a relational database and migrated its data to a document-based NoSQL system. I learned how to:

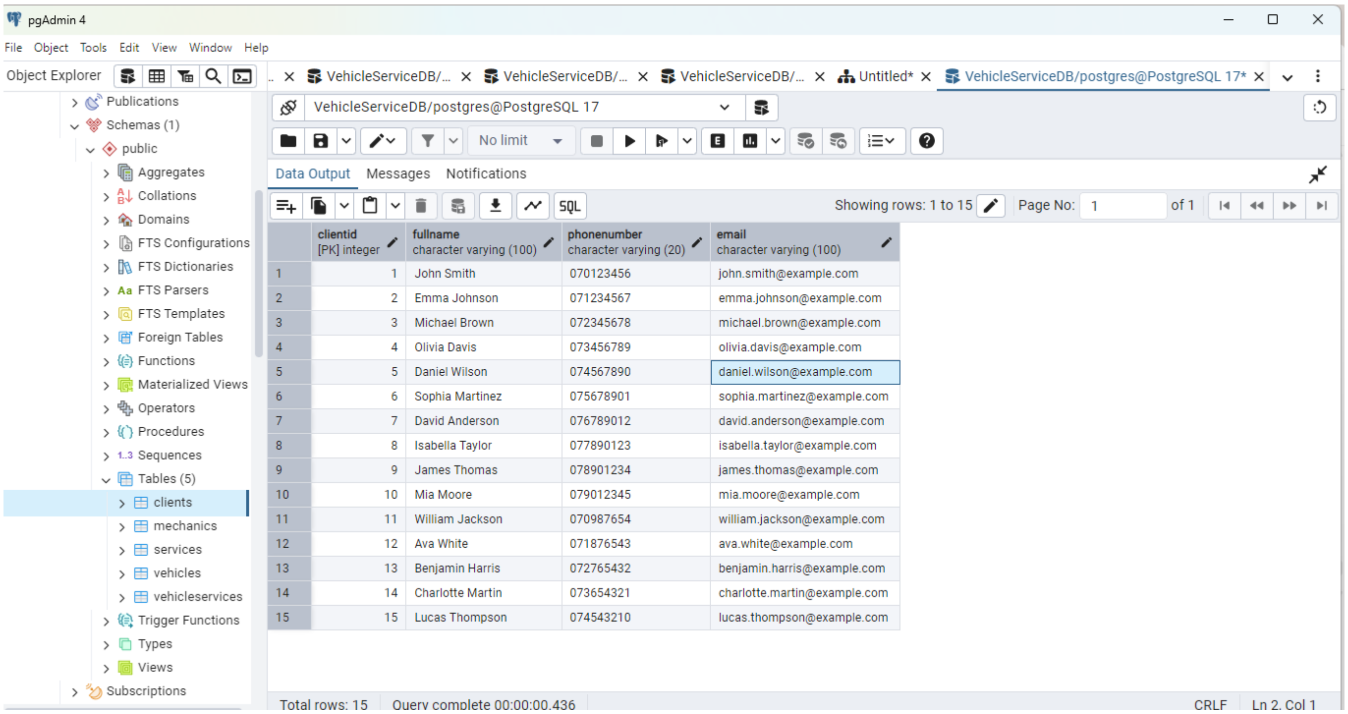
- Model data efficiently in both SQL and NoSQL paradigms
- Translate normalized relational data into denormalized document structures
- Use Python scripts to automate data migration

The experience highlighted the differences between the two data models and taught us how to select and justify a database based on the use case. It also improved our understanding of how to handle real-world migration scenarios, the importance of designing for access patterns, and how NoSQL systems can simplify data retrieval for complex, nested information.

Overall, this project provided valuable hands-on knowledge in working with heterogeneous database systems and the practical skills needed for future data engineering tasks.

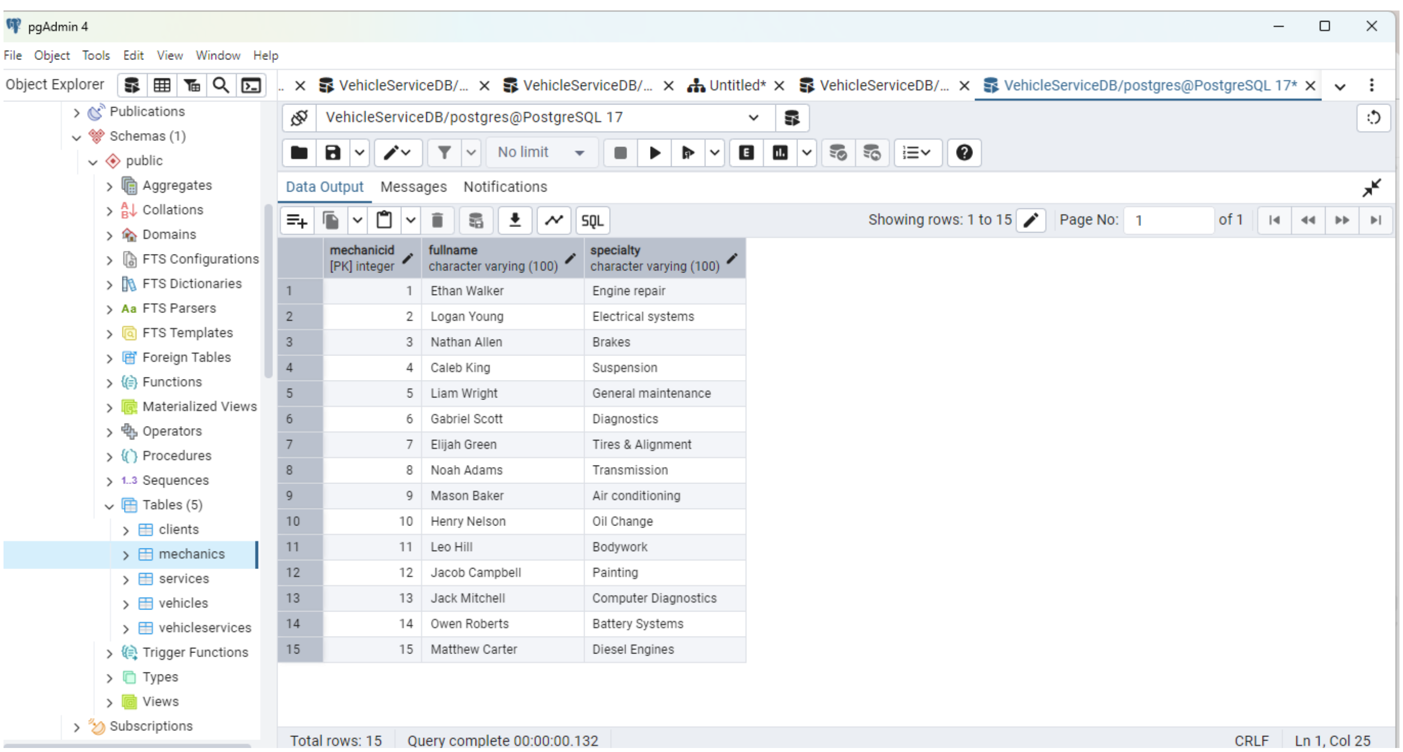
Data Population Screenshots:

CLIENTS TABLE



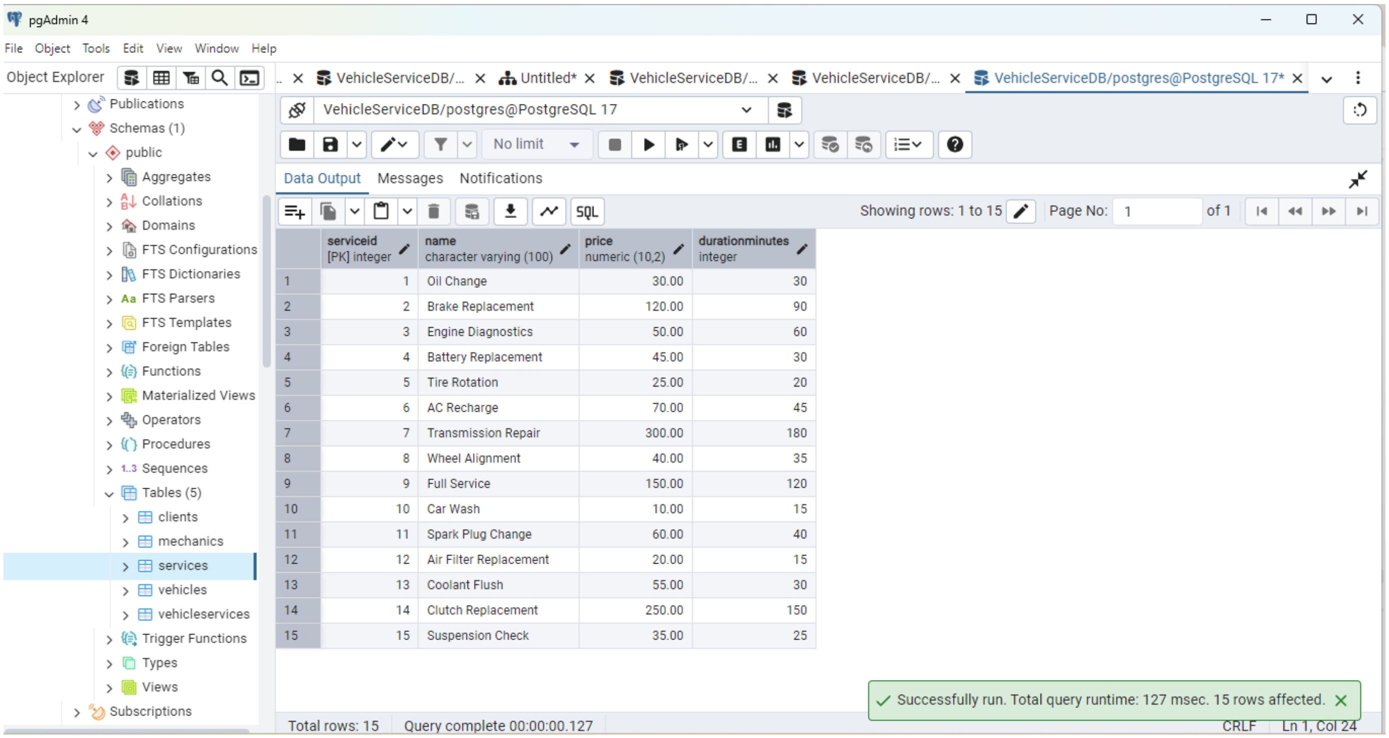
	clientid [PK] integer	fullname character varying (100)	phonenumber character varying (20)	email character varying (100)
1	1	John Smith	070123456	john.smith@example.com
2	2	Emma Johnson	071234567	emma.johnson@example.com
3	3	Michael Brown	072345678	michael.brown@example.com
4	4	Olivia Davis	073456789	olivia.davis@example.com
5	5	Daniel Wilson	074567890	daniel.wilson@example.com
6	6	Sophia Martinez	075678901	sophia.martinez@example.com
7	7	David Anderson	076789012	david.anderson@example.com
8	8	Isabella Taylor	077890123	isabella.taylor@example.com
9	9	James Thomas	078901234	james.thomas@example.com
10	10	Mia Moore	079012345	mia.moore@example.com
11	11	William Jackson	070987654	william.jackson@example.com
12	12	Ava White	071876543	ava.white@example.com
13	13	Benjamin Harris	072765432	benjamin.harris@example.com
14	14	Charlotte Martin	073654321	charlotte.martin@example.com
15	15	Lucas Thompson	074543210	lucas.thompson@example.com

MECHANICS TABLE



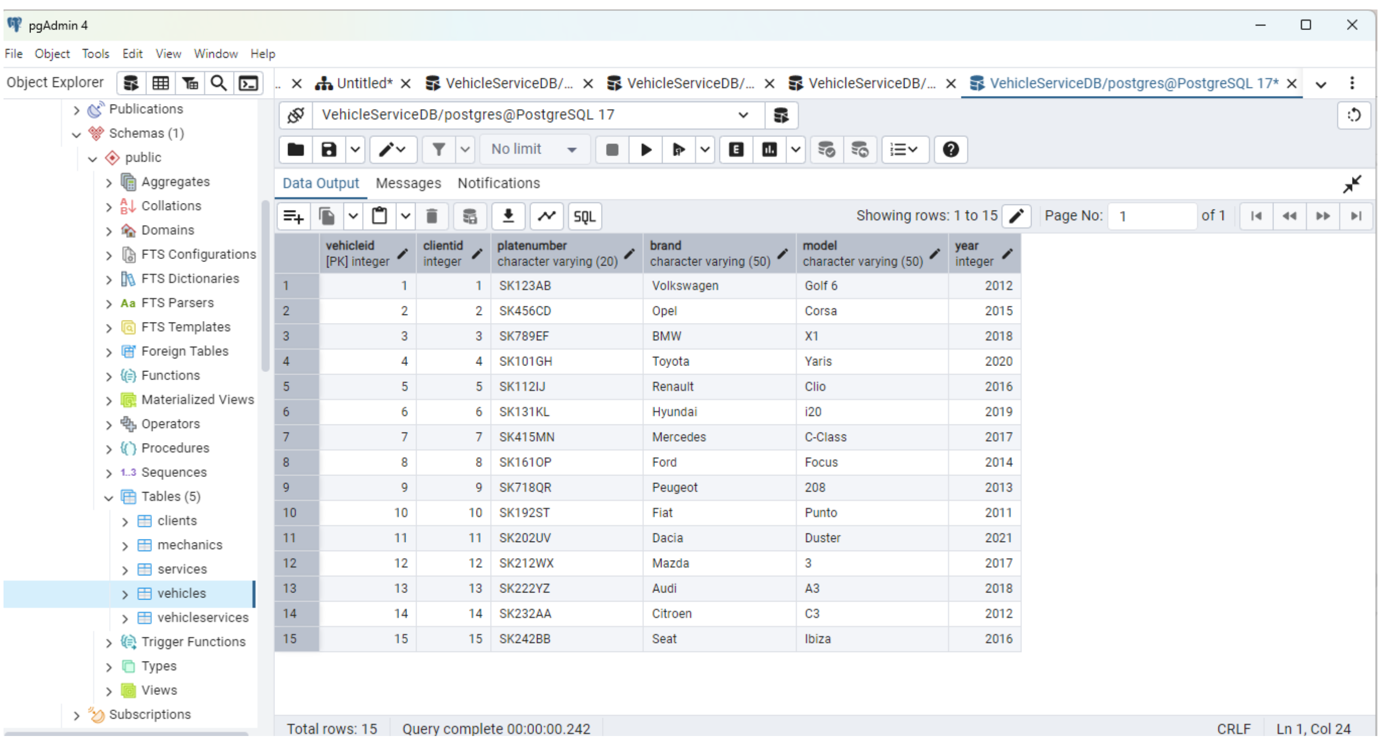
	mechanicid [PK] integer	fullname character varying (100)	specialty character varying (100)
1	1	Ethan Walker	Engine repair
2	2	Logan Young	Electrical systems
3	3	Nathan Allen	Brakes
4	4	Caleb King	Suspension
5	5	Liam Wright	General maintenance
6	6	Gabriel Scott	Diagnostics
7	7	Elijah Green	Tires & Alignment
8	8	Noah Adams	Transmission
9	9	Mason Baker	Air conditioning
10	10	Henry Nelson	Oil Change
11	11	Leo Hill	Bodywork
12	12	Jacob Campbell	Painting
13	13	Jack Mitchell	Computer Diagnostics
14	14	Owen Roberts	Battery Systems
15	15	Matthew Carter	Diesel Engines

SERVICES TABLE



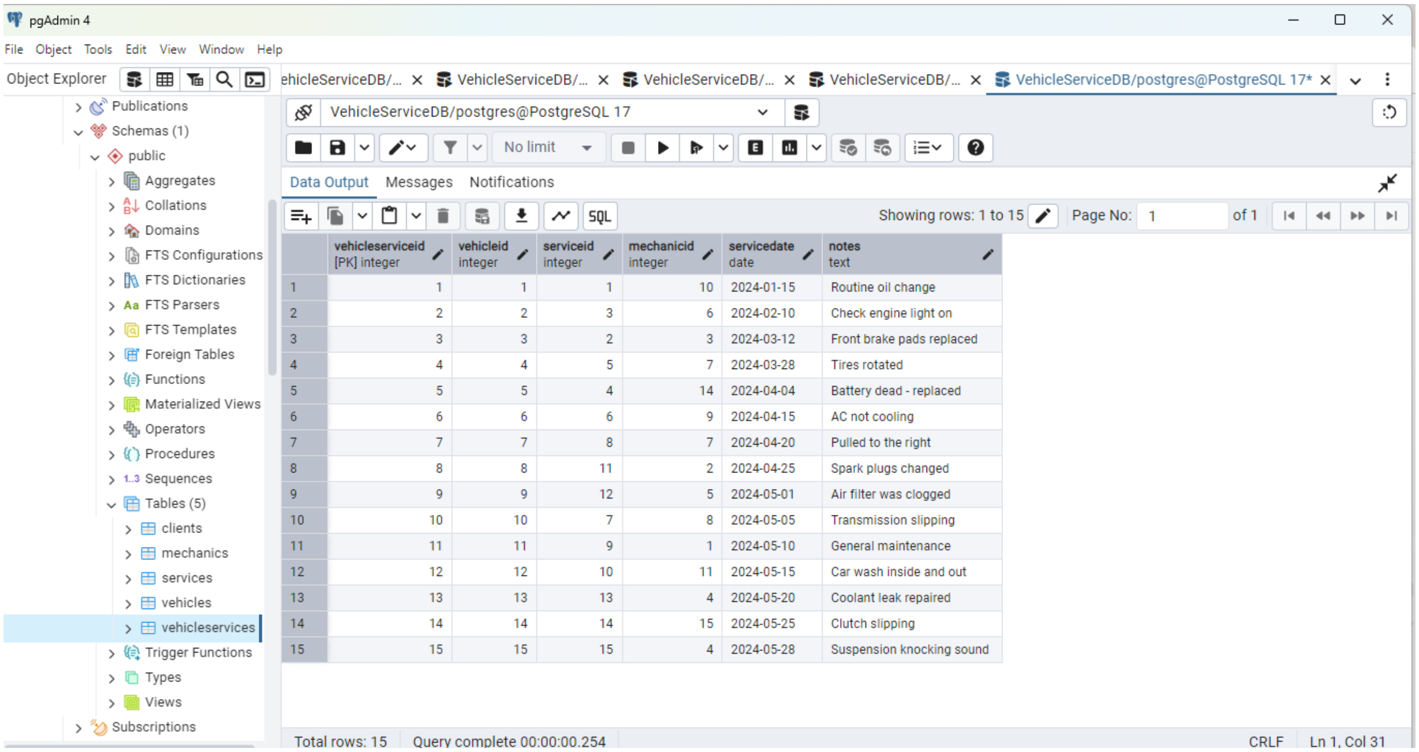
	serviceid [PK] integer	name character varying (100)	price numeric (10,2)	durationminutes integer
1	1	Oil Change	30.00	30
2	2	Brake Replacement	120.00	90
3	3	Engine Diagnostics	50.00	60
4	4	Battery Replacement	45.00	30
5	5	Tire Rotation	25.00	20
6	6	AC Recharge	70.00	45
7	7	Transmission Repair	300.00	180
8	8	Wheel Alignment	40.00	35
9	9	Full Service	150.00	120
10	10	Car Wash	10.00	15
11	11	Spark Plug Change	60.00	40
12	12	Air Filter Replacement	20.00	15
13	13	Coolant Flush	55.00	30
14	14	Clutch Replacement	250.00	150
15	15	Suspension Check	35.00	25

VEHICLES TABLE



	vehicleid [PK] integer	clientid integer	platenumber character varying (20)	brand character varying (50)	model character varying (50)	year integer
1	1	1	SK123AB	Volkswagen	Golf 6	2012
2	2	2	SK456CD	Opel	Corsa	2015
3	3	3	SK789EF	BMW	X1	2018
4	4	4	SK101GH	Toyota	Yaris	2020
5	5	5	SK112IJ	Renault	Clio	2016
6	6	6	SK131KL	Hyundai	i20	2019
7	7	7	SK415MN	Mercedes	C-Class	2017
8	8	8	SK161OP	Ford	Focus	2014
9	9	9	SK718QR	Peugeot	208	2013
10	10	10	SK192ST	Fiat	Punto	2011
11	11	11	SK202UV	Dacia	Duster	2021
12	12	12	SK212WX	Mazda	3	2017
13	13	13	SK222YZ	Audi	A3	2018
14	14	14	SK232AA	Citroen	C3	2012
15	15	15	SK242BB	Seat	Ibiza	2016

VEHICLESERVICES TABLE



	vehicleserviceid [PK] integer	vehicleid integer	serviceid integer	mechanicid integer	servicedate date	notes text
1	1	1	1	10	2024-01-15	Routine oil change
2	2	2	3	6	2024-02-10	Check engine light on
3	3	3	2	3	2024-03-12	Front brake pads replaced
4	4	4	5	7	2024-03-28	Tires rotated
5	5	5	4	14	2024-04-04	Battery dead - replaced
6	6	6	6	9	2024-04-15	AC not cooling
7	7	7	8	7	2024-04-20	Pulled to the right
8	8	8	11	2	2024-04-25	Spark plugs changed
9	9	9	12	5	2024-05-01	Air filter was clogged
10	10	10	7	8	2024-05-05	Transmission slipping
11	11	11	9	1	2024-05-10	General maintenance
12	12	12	10	11	2024-05-15	Car wash inside and out
13	13	13	13	4	2024-05-20	Coolant leak repaired
14	14	14	14	15	2024-05-25	Clutch slipping
15	15	15	15	4	2024-05-28	Suspension knocking sound

Total rows: 15 Query complete 00:00:00.254