Project Members and Contributions

22311523 Nikita Mavrodiy – Database schema, ERD, DDL, Supabase setup

22211149 Imane El Atmani – SQL queries, Report writing and documentation, DML

22307643 Ali Anas Nasr – Data population (DML) and SQL queries

22213471 Muaz Hisham – SQL queries and screenshots

1. Introduction

The Car Rental Database Management System (DBMS) is designed to manage the core operations of a car rental company. The system stores and organizes data related to customers, cars, rentals, employees, branches, payments, and maintenance activities. By using a relational database structure, the system enables efficient data storage, querying, and reporting to support daily business operations and decision-making.

Assumptions

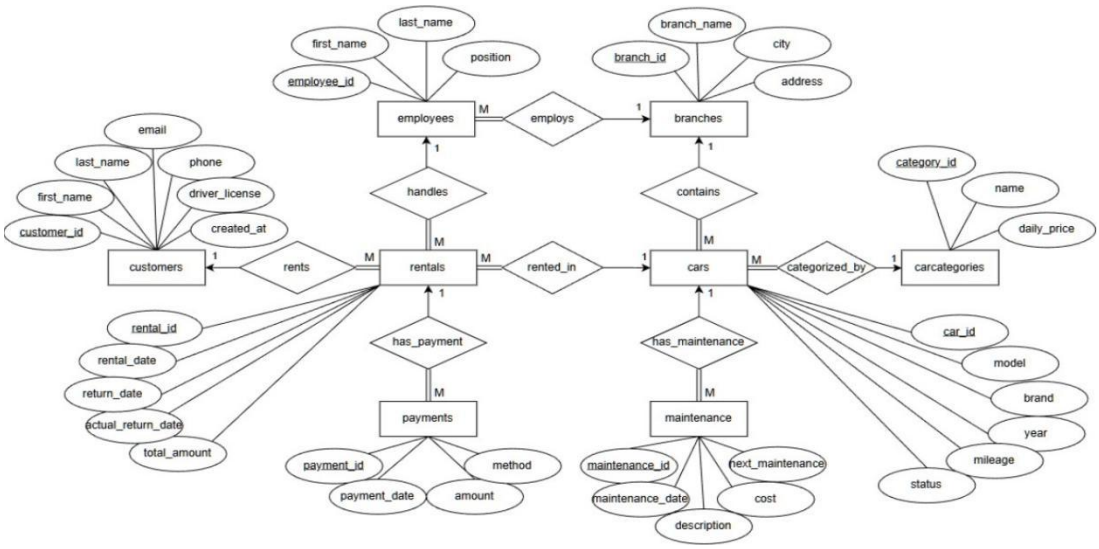
- Each rental is associated with one customer, one car, and one employee.
- Cars belong to a single branch and a single category.
- Employees work at one branch.
- Payments are linked to rentals.
- Maintenance records are associated with specific cars.

2. Database

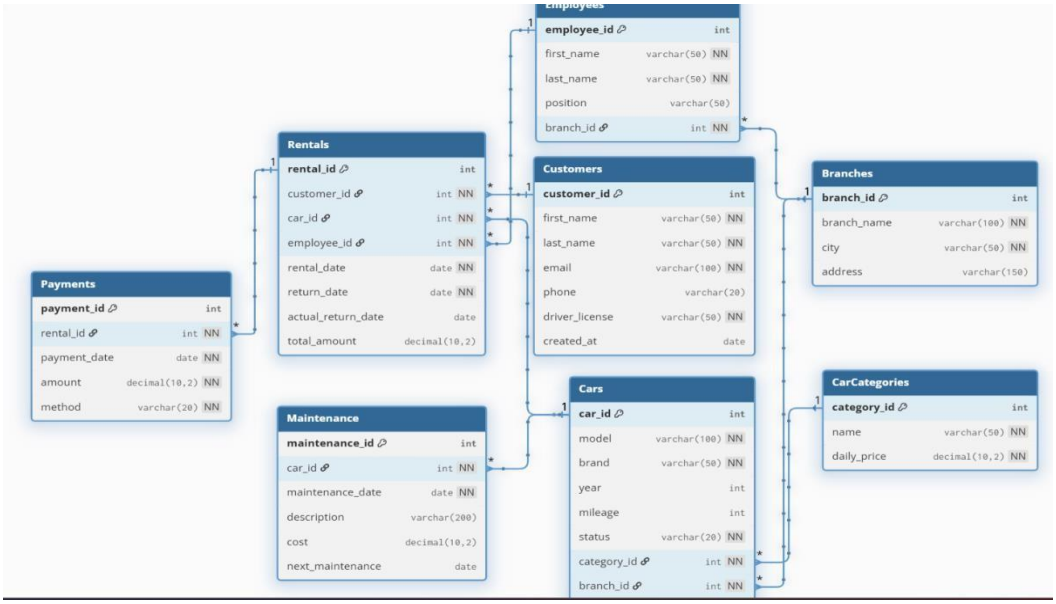
ER Diagram

The ER diagram represents the entities in the system and the relationships between them, including customers, cars, rentals, employees, branches, payments, maintenance, and car categories.

ERD diagram:



Relational model:



Data Definition Language (DDL)

The database schema was implemented using SQL and includes primary keys, foreign keys, constraints, and default values to ensure data integrity. The schema defines eight related tables and enforces relationships between them using foreign key constraints.

```
CREATE TABLE Customers (
    customer_id INTEGER PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    phone VARCHAR(20),
    driver_license VARCHAR(50) UNIQUE NOT NULL,
    created_at DATE DEFAULT CURRENT_DATE
);

CREATE TABLE CarCategories (
    category_id INTEGER PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    daily_price DECIMAL(10,2) NOT NULL CHECK (daily_price > 0)
);

CREATE TABLE Branches (
    branch_id INTEGER PRIMARY KEY,
    branch_name VARCHAR(100) NOT NULL,
    city VARCHAR(50) NOT NULL,
    address VARCHAR(150)
);

CREATE TABLE Cars (
    car_id INTEGER PRIMARY KEY,
    model VARCHAR(100) NOT NULL,
    brand VARCHAR(50) NOT NULL,
    year INTEGER CHECK (year >= 1990),
    mileage INTEGER CHECK (mileage >= 0),
    status VARCHAR(20) NOT NULL CHECK (status IN ('Available', 'Rented', 'Maintenance')) DEFAULT 'Available',
    category_id INTEGER NOT NULL,
    branch_id INTEGER NOT NULL,
    FOREIGN KEY (category_id) REFERENCES CarCategories(category_id),
    FOREIGN KEY (branch_id) REFERENCES Branches(branch_id)
);

CREATE TABLE Employees (
    employee_id INTEGER PRIMARY KEY,
```

```
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    position VARCHAR(50),
    branch_id INTEGER NOT NULL,
    FOREIGN KEY (branch_id) REFERENCES Branches(branch_id)
);

CREATE TABLE Rentals (
    rental_id INTEGER PRIMARY KEY,
    customer_id INTEGER NOT NULL,
    car_id INTEGER NOT NULL,
    employee_id INTEGER NOT NULL,
    rental_date DATE NOT NULL,
    return_date DATE NOT NULL,
    actual_return_date DATE,
    total_amount DECIMAL(10,2) CHECK (total_amount >= 0),
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
    FOREIGN KEY (car_id) REFERENCES Cars(car_id),
    FOREIGN KEY (employee_id) REFERENCES Employees(employee_id)
);

CREATE TABLE Payments (
    payment_id INTEGER PRIMARY KEY,
    rental_id INTEGER NOT NULL,
    payment_date DATE NOT NULL,
    amount DECIMAL(10,2) NOT NULL CHECK (amount > 0),
    method VARCHAR(20) NOT NULL CHECK (method IN ('Cash', 'Card')),
    FOREIGN KEY (rental_id) REFERENCES Rentals(rental_id)
);

CREATE TABLE Maintenance (
    maintenance_id INTEGER PRIMARY KEY,
    car_id INTEGER NOT NULL,
    maintenance_date DATE NOT NULL,
    description VARCHAR(200),
    cost DECIMAL(10,2) CHECK (cost >= 0),
    next_maintenance DATE,
    FOREIGN KEY (car_id) REFERENCES Cars(car_id)
);
```

Data Manipulation Language (DML)

Sample data was inserted into all tables using SQL INSERT statements. UPDATE and DELETE operations were also performed to demonstrate data modification and removal.

```
INSERT INTO customers (customer_id, first_name, last_name, email, phone, driver_license, created_at) VALUES
(1, 'John', 'Smith', 'john@mail.com', '55510001', 'DL1001', NOW()),
(2, 'Emma', 'Stone', 'emma@mail.com', '55510002', 'DL1002', NOW()),
(3, 'Adam', 'Brown', 'adam@mail.com', '55510003', 'DL1003', NOW()),
(4, 'Layla', 'Haddad', 'layla@mail.com', '55510004', 'DL1004', NOW()),
(5, 'Karim', 'Said', 'karim@mail.com', '55510005', 'DL1005', NOW()),
(6, 'Aisha', 'Noor', 'aisha@mail.com', '55510006', 'DL1006', NOW()),
(7, 'David', 'White', 'david@mail.com', '55510007', 'DL1007', NOW()),
(8, 'Nina', 'Keller', 'nina@mail.com', '55510008', 'DL1008', NOW()),
(9, 'Mert', 'Yilmaz', 'mert@mail.com', '55510009', 'DL1009', NOW()),
(10, 'Chloe', 'Taylor', 'chloe@mail.com', '55510010', 'DL1010', NOW());



INSERT INTO branches (branch_id, branch_name, city, address) VALUES
(1, 'Central Branch', 'Nicosia', '123 Main Street'),
(2, 'Airport Branch', 'Larnaca', 'Larnaca Airport Road'),
(3, 'Harbor Branch', 'Kyrenia', 'Harbor Avenue 45'),
(4, 'City Mall Branch', 'Famagusta', 'City Mall, Level 1');


INSERT INTO carcategories (category_id, name , daily_price)
VALUES
(1, 'Economy',30),
(2, 'SUV',50),
(3, 'Luxury',120),
(4,'Electric',90);



INSERT INTO cars (car_id, model, brand, year, mileage, status, category_id, branch_id) VALUES
```

```
(1, 'Corolla', 'Toyota', 2020, 45000, 'Available', 1, 1),
(2, 'Yaris', 'Toyota', 2021, 28000, 'Available', 1, 2),
(3, 'Civic', 'Honda', 2019, 62000, 'Maintenance', 2, 1),
(4, 'Elantra', 'Hyundai', 2022, 15000, 'Available', 2, 3),
(5, 'Sportage', 'Kia', 2021, 35000, 'Rented', 3, 2),
(6, 'Rav4', 'Toyota', 2020, 50000, 'Available', 3, 1),
(7, 'X5', 'BMW', 2022, 20000, 'Available', 4, 4),
(8, 'A6', 'Audi', 2021, 25000, 'Rented', 4, 1),
(9, 'Accord', 'Honda', 2020, 40000, 'Available', 2, 3),
(10, 'Camry', 'Toyota', 2023, 10000, 'Available', 2, 4);
```

```
INSERT INTO employees (employee_id, first_name, last_name, position, branch_id) VALUES
(1, 'Nikos', 'Georgiou', 'Manager', 1),
(2, 'Sara', 'Hassan', 'Agent', 1),
(3, 'Ali', 'Khan', 'Agent', 2),
(4, 'Maria', 'Lopez', 'Technician', 3),
(5, 'Omer', 'Demir', 'Agent', 4),
(6, 'Linda', 'Brown', 'Manager', 2);
```

```
INSERT INTO rentals (rental_id, customer_id, car_id, employee_id, rental_date, return_date, actual_return_date,
total_amount)
VALUES
(1, 1, 1, 2, '2025-01-01', '2025-01-05', '2025-01-05', 120),
(2, 2, 5, 3, '2025-02-10', '2025-02-15', NULL, 300),
(3, 3, 8, 1, '2025-03-01', '2025-03-07', '2025-03-07', 700),
(4, 4, 2, 2, '2025-03-12', '2025-03-15', NULL, 90),
(5, 5, 4, 4, '2025-04-01', '2025-04-05', '2025-04-04', 160),
(6, 6, 6, 1, '2025-04-10', '2025-04-14', '2025-04-14', 240),
(7, 7, 9, 5, '2025-04-20', '2025-04-25', '2025-04-24', 200),
(8, 8, 10, 6, '2025-05-01', '2025-05-03', NULL, 80),
(9, 9, 3, 4, '2025-05-05', '2025-05-10', NULL, 200),
(10, 10, 7, 3, '2025-05-10', '2025-05-14', NULL, 480);
```

```
INSERT INTO payments (payment_id, rental_id, payment_date, amount, method) VALUES
(1, 1, '2025-01-05', 120, 'Cash'),
(2, 3, '2025-03-07', 700, 'Card'),
(3, 5, '2025-04-04', 160, 'Card'),
(4, 6, '2025-04-14', 240, 'Cash'),
(5, 7, '2025-04-24', 200, 'Card'),
(6, 8, '2025-05-02', 40, 'Card'),
(7, 2, '2025-02-10', 100, 'Cash'),
(8, 9, '2025-05-06', 100, 'Cash'),
(9, 10, '2025-05-11', 480, 'Card'),
(10, 4, '2025-03-12', 90, 'Cash');
```

```
INSERT INTO maintenance (maintenance_id, car_id, maintenance_date, description, cost, next_maintenance) VALUES
(1, 3, '2025-01-15', 'Oil change', 50, '2025-04-15'),
(2, 6, '2025-01-20', 'Engine check', 150, '2025-06-20'),
(3, 1, '2025-02-05', 'Tire replacement', 200, '2025-09-05'),
(4, 9, '2025-02-10', 'Brake check', 80, '2025-05-10'),
(5, 4, '2025-03-01', 'Filter change', 40, '2025-06-01'),
(6, 10, '2025-03-12', 'Battery replacement', 120, '2025-09-12'),
(7, 2, '2025-04-01', 'Full inspection', 300, '2025-10-01'),
(8, 5, '2025-04-11', 'Suspension repair', 250, '2025-09-11'),
(9, 7, '2025-04-25', 'Oil change', 60, '2025-07-25'),
(10, 8, '2025-05-01', 'Transmission check', 350, '2025-11-01');
```

```
UPDATE cars
SET status = 'Available'
WHERE car_id=4;

DELETE FROM payments
WHERE  payment_id = 10;
```

3. Queries

A total of **15+ SQL queries** were written to retrieve and analyze data from the database. These queries demonstrate the use of joins, aggregate functions, subqueries, grouping, filtering, and SQL functions.

Examples include:

- Listing all customers
- Displaying rental details with customer and car information
- Calculating total and average rental revenue
- Finding cars with above-average mileage
- Filtering data using date and string functions

Each query is accompanied by a screenshot of its result to verify correctness.

```sql
-- ============================================
-- 7 BASIC SQL QUERIES
-- ============================================
-- Query 1: SIMPLE SELECT - All customers
select
  *
from
  Customers;
```

customer_id	first_name	last_name	email	phone	driver_licens	created_at
1	John	Smith	john@mail.com	55510001	DL1001	2025-12-09
2	Emma	Stone	emma@mail.com	55510002	DL1002	2025-12-09
3	Adam	Brown	adam@mail.com	55510003	DL1003	2025-12-09
4	Layla	Haddad	layla@mail.com	55510004	DL1004	2025-12-09
5	Karim	Said	karim@mail.com	55510005	DL1005	2025-12-09
6	Aisha	Noor	aisha@mail.com	55510006	DL1006	2025-12-09
7	David	White	david@mail.com	55510007	DL1007	2025-12-09
8	Nina	Keller	nina@mail.com	55510008	DL1008	2025-12-09
9	Mert	Yilmaz	mert@mail.com	55510009	DL1009	2025-12-09
10	Chloe	Taylor	chloe@mail.com	55510010	DL1010	2025-12-09

```sql
-- Query 2: INNER JOIN - Rental details with customer and car information
select
  Rentals.rental_id,
  Customers.first_name || ' ' || Customers.last_name as customer_name,
  Cars.brand || ' ' || Cars.model as car,
  Rentals.rental_date,
  Rentals.return_date,
  Rentals.total_amount
from
  Rentals
  inner join Customers on Rentals.customer_id = Customers.customer_id
  inner join Cars on Rentals.car_id = Cars.car_id;
```

rental_id	customer_name	car	rental_date	return_date	total_amount
1	John Smith	Toyota Corolla	2025-01-01	2025-01-05	120.00
2	Emma Stone	Kia Sportage	2025-02-10	2025-02-15	300.00
3	Adam Brown	Audi A6	2025-03-01	2025-03-07	700.00
4	Layla Haddad	Toyota Yaris	2025-03-12	2025-03-15	90.00
5	Karim Said	Hyundai Elantra	2025-04-01	2025-04-05	160.00
6	Aisha Noor	Toyota Rav4	2025-04-10	2025-04-14	240.00
7	David White	Honda Accord	2025-04-20	2025-04-25	200.00
8	Nina Keller	Toyota Camry	2025-05-01	2025-05-03	80.00
9	Mert Yilmaz	Honda Civic	2025-05-05	2025-05-10	200.00
10	Chloe Taylor	BMW X5	2025-05-10	2025-05-14	480.00

```sql
-- Query 3: LEFT JOIN - All cars and their rentals (if any)
select
  Cars.car_id,
```

```
  Cars.brand,
  Cars.model,
  Cars.status,
  Rentals.rental_id,
  Rentals.rental_date,
  Rentals.return_date
from
  Cars
  left join Rentals on Cars.car_id = Rentals.car_id;
```

car_id	brand	model	status	rental_id	rental_date	return_date
1	Toyota	Corolla	Available	1	2025-01-01	2025-01-05
5	Kia	Sportage	Rented	2	2025-02-10	2025-02-15
8	Audi	A6	Rented	3	2025-03-01	2025-03-07
2	Toyota	Yaris	Available	4	2025-03-12	2025-03-15
4	Hyundai	Elantra	Available	5	2025-04-01	2025-04-05
6	Toyota	Rav4	Available	6	2025-04-10	2025-04-14
9	Honda	Accord	Available	7	2025-04-20	2025-04-25
10	Toyota	Camry	Available	8	2025-05-01	2025-05-03
3	Honda	Civic	Maintenance	9	2025-05-05	2025-05-10
7	BMW	X5	Available	10	2025-05-10	2025-05-14

```
-- Query 4: FILTERING (WHERE) - Available cars only
select
  car_id,
  brand,
  model,
  year,
  mileage,
  status
from
  Cars
where
  Cars.status = 'Available';
```

car_id	brand	model	year	mileage	status
1	Toyota	Corolla	2020	45000	Available
2	Toyota	Yaris	2021	28000	Available
6	Toyota	Rav4	2020	50000	Available
7	BMW	X5	2022	20000	Available
9	Honda	Accord	2020	40000	Available
10	Toyota	Camry	2023	10000	Available
4	Hyundai	Elantra	2022	15000	Available

```
-- Query 5: ORDER BY - Customers ordered by registration date (newest first)
select
  customer_id,
  first_name,
  last_name,
  email,
  phone,
  created_at
from
  Customers
order by
  Customers.created_at desc;
```

customer_id	first_name	last_name	email	phone	created_at
1	John	Smith	john@mail.com	55510001	2025-12-09
2	Emma	Stone	emma@mail.com	55510002	2025-12-09
3	Adam	Brown	adam@mail.com	55510003	2025-12-09
4	Layla	Haddad	layla@mail.com	55510004	2025-12-09
5	Karim	Said	karim@mail.com	55510005	2025-12-09
6	Aisha	Noor	aisha@mail.com	55510006	2025-12-09
7	David	White	david@mail.com	55510007	2025-12-09
8	Nina	Keller	nina@mail.com	55510008	2025-12-09
9	Mert	Yilmaz	mert@mail.com	55510009	2025-12-09
10	Chloe	Taylor	chloe@mail.com	55510010	2025-12-09

```sql
-- Query 6: BASIC AGGREGATION (COUNT) - Count cars in each branch
select
  Branches.branch_name,
  COUNT(Cars.car_id) as number_of_cars
from
  Branches
  left join Cars on Branches.branch_id = Cars.branch_id
group by
  Branches.branch_id,
  Branches.branch_name;
```

branch_name	number_of_car
City Mall Branch	2
Airport Branch	2
Harbor Branch	2
Central Branch	4

```sql
-- Query 7: BASIC GROUP BY - Revenue per branch
select
  Branches.branch_name,
  COALESCE(SUM(Rentals.total_amount), 0) as total_revenue,
  COUNT(Rentals.rental_id) as number_of_rentals
from
  Branches
  left join Cars on Branches.branch_id = Cars.branch_id
  left join Rentals on Cars.car_id = Rentals.car_id
group by
  Branches.branch_id,
  Branches.branch_name
order by
  total_revenue desc;
```

branch_name	total_revenue	number_of_rentals
Central Branch	1260.00	4
City Mall Branch	560.00	2
Airport Branch	390.00	2
Harbor Branch	360.00	2

```sql
-- Query 8: Total Rental Revenue
SELECT SUM(total_amount) AS total_revenue
FROM rentals;
```

total_revenue
2570.00

```sql
-- Query 9: Average Rental Price
SELECT AVG(total_amount) AS avg_rental_price
FROM rentals;
```

avg_rental_price
257.0000000000000000

```sql
-- Query 10: Car Category Price Range
SELECT
    MIN(daily_price) AS cheapest_category,
    MAX(daily_price) AS most_expensive_category
FROM carcategories;
```

cheapest_category	most_expensive_category
30.00	120.00

```sql
-- Query 11: Branches with More Than 3 Cars
SELECT branch_id, COUNT(*) AS total_cars
FROM cars
GROUP BY branch_id
HAVING COUNT(*) > 3;
```

branch_id	total_cars
1	4

```sql
-- Query 12: Customers with First Name Starting with A
SELECT first_name, last_name
FROM customers
WHERE first_name LIKE 'A%';
```

first_name	last_name
Adam	Brown
Aisha	Noor

```sql
-- Query 13: Aged Rentals (30+ Days)
SELECT *
FROM rentals
WHERE rental_date <= CURRENT_DATE - INTERVAL '30 days';
```

rental_id	customer_id	car_id	employee_id	rental_date	return_date	actual_return	total_amount
1	1	1	2	2025-01-01	2025-01-05	2025-01-05	120.00
2	2	5	3	2025-02-10	2025-02-15	NULL	300.00
3	3	8	1	2025-03-01	2025-03-07	2025-03-07	700.00
4	4	2	2	2025-03-12	2025-03-15	NULL	90.00
5	5	4	4	2025-04-01	2025-04-05	2025-04-04	160.00
6	6	6	1	2025-04-10	2025-04-14	2025-04-14	240.00
7	7	9	5	2025-04-20	2025-04-25	2025-04-24	200.00
8	8	10	6	2025-05-01	2025-05-03	NULL	80.00
9	9	3	4	2025-05-05	2025-05-10	NULL	200.00
10	10	7	3	2025-05-10	2025-05-14	NULL	480.00

```sql
-- Query 14: Customers Who Rented SUVs
SELECT first_name, last_name
FROM customers
WHERE customer_id IN (
    SELECT customer_id
    FROM rentals
    JOIN cars ON rentals.car_id = cars.car_id
    WHERE cars.category_id = (
        SELECT category_id FROM carcategories WHERE name = 'SUV'
    )
);
```

first_name	last_name
Nina	Keller
Mert	Yilmaz
David	White
Karim	Said

```sql
-- Query 15: Available Cars with Below-Average Mileage
SELECT *
FROM cars
WHERE status = 'Available'
  AND mileage < (
        SELECT AVG(mileage)
        FROM cars
    );
```

car_id	model	brand	year	mileage	status	category_id	branch_id
2	Yaris	Toyota	2021	28000	Available	1	2
7	X5	BMW	2022	20000	Available	4	4
10	Camry	Toyota	2023	10000	Available	2	4
4	Elantra	Hyundai	2022	15000	Available	2	3

```sql
-- Query 16: Rental One-Month Projection
SELECT
    rental_id,
    rental_date,
    rental_date + INTERVAL '1 month' AS one_month_after_rental
FROM rentals;
```

rental_id	rental_date	one_month_after_rent
1	2025-01-01	2025-02-01 00:00:00
2	2025-02-10	2025-03-10 00:00:00
3	2025-03-01	2025-04-01 00:00:00
4	2025-03-12	2025-04-12 00:00:00
5	2025-04-01	2025-05-01 00:00:00
6	2025-04-10	2025-05-10 00:00:00
7	2025-04-20	2025-05-20 00:00:00
8	2025-05-01	2025-06-01 00:00:00
9	2025-05-05	2025-06-05 00:00:00
10	2025-05-10	2025-06-10 00:00:00

```sql
-- Query 17: Rental Price Categorization
SELECT
    rental_id,
    total_amount,
    CASE
        WHEN total_amount < 100 THEN 'Low Cost'
        WHEN total_amount BETWEEN 100 AND 300 THEN 'Medium Cost'
        ELSE 'High Cost'
    END AS rental_price_category
FROM rentals;
```

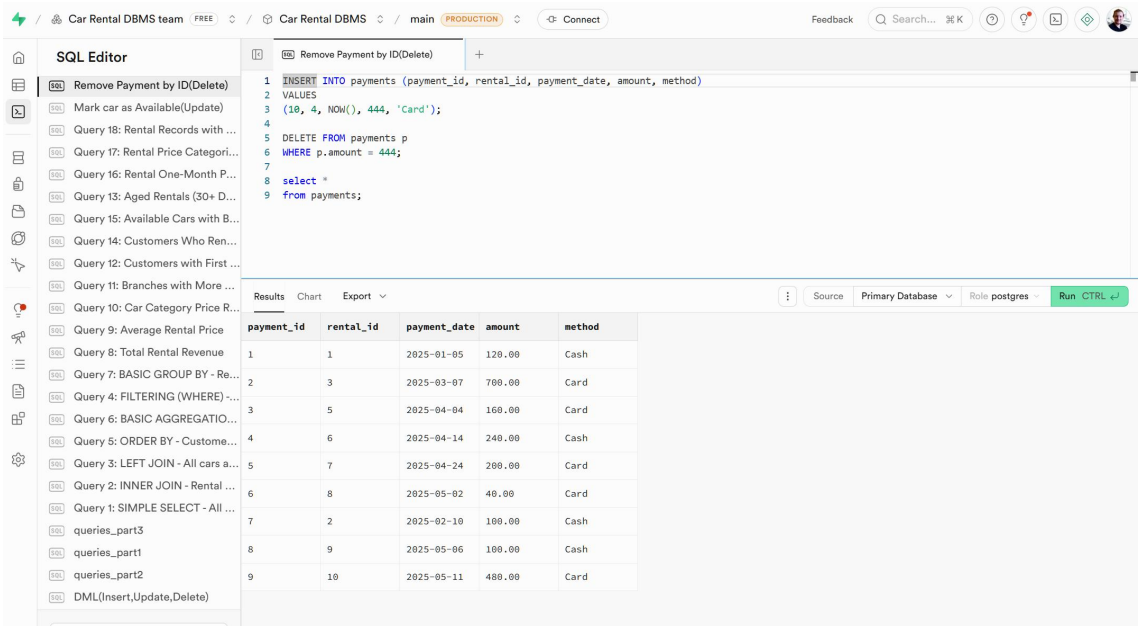
rental_id	total_amount	rental_price_category
1	120.00	Medium Cost
2	300.00	Medium Cost
3	700.00	High Cost
4	90.00	Low Cost
5	160.00	Medium Cost
6	240.00	Medium Cost
7	200.00	Medium Cost
8	80.00	Low Cost
9	200.00	Medium Cost
10	480.00	High Cost

```sql
-- Query 18: Rental Records with Formatted Dates
SELECT
    rental_id,
    TO_CHAR(rental_date, 'DD Mon YYYY') AS formatted_rental_date
FROM rentals;
```

rental_id	formatted_rental_date
1	01 Jan 2025
2	10 Feb 2025
3	01 Mar 2025
4	12 Mar 2025
5	01 Apr 2025
6	10 Apr 2025
7	20 Apr 2025
8	01 May 2025
9	05 May 2025
10	10 May 2025

4. User Interface (Optional)

The Supabase platform was used as a graphical interface to interact with the database. It allows inserting, updating, deleting records and executing SQL queries visually.



5. GitHub