SMILES: Car Rental System

Yunseo Heo, Kelly Ng

1. Introduction

1-1. Scope

The software product to be produced is a Car Rental System named Smiles, a database-driven application designed to facilitate car rentals. This system will manage the operations of a car rental company, including vehicle inventory, client information, rental agreements, and staff management. The system will allow customers to rent vehicles, track rental periods, and manage vehicle maintenance across multiple outlets. The primary purpose of the system is to streamline the rental process, improve customer satisfaction, and ensure efficient management of vehicles and staff.

1-2. Purpose & Objectives

- To create a centralized database that enables seamless vehicle rental management.
- To provide a user-friendly interface for customers and staff, supporting efficient querying and retrieval of rental data for administrative and operational decisions.
- Improve data consistency and integrity through relational database modelling.
- Store detailed records of clients, rental transactions, vehicle details, and faults.

1-3. Benefits

- Improved customer experience with easy booking and tracking.
- Efficient management of vehicle inventory and maintenance.
- Enhanced reporting and analytics for business decisions.

2. Overall Descriptions

2-1. Product Perspective

The Smiles Car Rental System is a self-contained product that can be integrated with other systems such as payment gateways, GPS tracking systems, and customer relationship management (CRM) tools. It will operate as a web-based application accessible to both customers and staff.

2-2. Product Functions

1. Vehicle Management
    a. Add, update, and delete vehicles.
    b. Track vehicle location and availability.
2. Client Management
    a. Register and manage personal and business clients.
    b. Track rental history and preferences.
3. Rental Agreement Management
    a. Create and manage rental agreements.
    b. Track rental periods, vehicle usage, and payments.
4. Staff Management
    a. Manage staff information and assignments.
    b. Track staff performance and responsibilities.
5. Reporting and Analytics
    a. Generate reports on vehicle usage, rental income, and client activity.
    b. Provide insights for business decisions.
6. Fault Reports
    a. Log vehicle inspections/maintenance.
    b. Record any reported faults.

3. Data Requirements

3-1. Entities and Attributes

Entity	Attributes
Outlet	OutletNo. (Primary Key), Address, PhoneNo., FaxNo., StaffNo. (Foreign Key)
Vehicle	RegistrationNo. (Primary Key), PlateNo., Type, FuelType, Model, Make, EngineSize, Capacity, CurrentMileage, MOTDue, HireRate, CurrentLocation(Outlet), Status, InsuranceExpiry
PersonalClient	ClientNo. (Primary Key), Address, Telephone, Email, LoyaltyPoints, fName, lName, DOB, LiscenceNo.
BusinessClient	ClientNo. (Primary Key), Address, Telephone, Email, LoyaltyPoints, BusinessName, BusinessType, FaxNo.
HireAgreement	HireNo. (Primary Key), StartDate, TerminationDate, RentalDuration, Pre-hireMileage, Post-hireMileage, CurrentMileage, TotalCost, PaymentMethod, PaymentStatus, DepositAmount

FaultReport	FaultID (Primary Key), fName, lName, DateChecked, FaultFound(Y/N)
Staff	StaffNo. (Primary Key), fName, lName, Address, PhoneNo., DOB, Sex, NIN, DateJoined, JobTitle, Salary, OutletNo. (Foreign Key)

3-2. Entity Relationships

1. Staff - BelongsTo - Outlet

- **Relationship:** Each **Staff** belongs to **one** Outlet, but an **Outlet** can have **many** Staff members.
- **Cardinality: 1:*** (One Outlet has many Staff) & **1:1** (Each Staff belongs to one Outlet)
- **Foreign Key:** OutletNo in Staff references Outlet

2. Staff - Checks - Vehicle

- **Relationship:** Each **Vehicle** can be checked by **one** Staff member, and a **Staff** can check multiple Vehicles.
- **Cardinality: 1:*** (One Staff can check many Vehicles) & **0:*** (Staff can check multiple Vehicles; 0 to many Vehicles)
- **Foreign Key:** StaffNo in FaultReport references Staff

3. Staff - Reports - FaultReport

- **Relationship:** A **FaultReport** is created when a **Staff** member checks a vehicle and each Vehicle gets one FaultReport.
- **Cardinality: 1:1** (One Staff reports one FaultReport per check) & **1:1** (Each Vehicle gets one FaultReport)
- **Foreign Key:** StaffNo in FaultReport references Staff

4. Vehicle - RentsOut - Outlet

- **Relationship:** Each **Outlet** can have **many** Vehicles, but each **Vehicle** is assigned to only one Outlet at a time.
- **Cardinality: 1:*** (One Outlet has many Vehicles) & **0:*** (Each Outlet can have many Vehicles; 0 to many)
- **Foreign Key:** CurrentLocation(Outlet) in Vehicle references OutletNo

5. Client - Hires - Vehicle

- **Relationship:** A **Client** can hire multiple Vehicles, but each **Vehicle** can be rented by only **one** client at a time.
- **Cardinality: 1:*** (One Client hires many Vehicles) & **0:1** (each Vehicle can be rented by only one client at a time, or not rented out)

- **Foreign Key:** ClientNo in HireAgreement references Client

6. Client - Signs - Hire Agreement

- **Relationship:** Each **Hire Agreement** is signed by **one** Client, but a **Client** can have multiple Hire Agreements.
- **Cardinality: 1:*** (One Client has many Hire Agreements) & **1:1** (Each Hire Agreement is signed by one Client)
- **Foreign Key:** ClientNo in HireAgreement references Client

7. Hire Agreement - Includes - Vehicle

- **Relationship:** A **Hire Agreement** is associated with one **Vehicle**, but a **Vehicle** can have multiple hire agreements over time.
- **Cardinality: 0:*** (One Vehicle can have multiple Hire Agreements over time; 0 is when a Vehicle has yet to have their first Hire Agreement) & **1:1** (A Hire Agreement is associated with one Vehicle)
- **Foreign Key:** RegistrationNo in HireAgreement references Vehicle

8. Client - Has Subclasses - PersonalClient and BusinessClient

- **Relationship:** A **Client** must be either a **PersonalClient** or a **BusinessClient**.
- **Cardinality: 1:1 (Specialization - Mandatory, OR)**
- **Foreign Keys:**
    - ClientNo in PersonalClient references Client
    - ClientNo in BusinessClient references Client

3-3. Dataset

The data for the SMILES car rental system will be generated and stored based on the interactions between users, vehicles, and rental transactions. The primary sources of data include:

1. Customer Input:

- Personal information (name, contact details, address, driving license details, etc.) will be collected when customers register on the system.
- Rental history, including past and current bookings, will be stored and updated upon every transaction.

2. Fleet Management System:

- Information about available vehicles (registration number, plate number, model, make, engine size, capacity, current mileage, etc.) will be maintained.

- The current location of each vehicle will be updated dynamically as vehicles move between outlets.
- Vehicle maintenance records, including MOT due dates and fault reports, will be recorded after inspections by staff.

3. Rental Transactions:

- Data related to rental agreements, such as hire number and rental duration will be stored.
- Payment records will be generated based on the rental period and any additional charges.

4. Staff Management System:

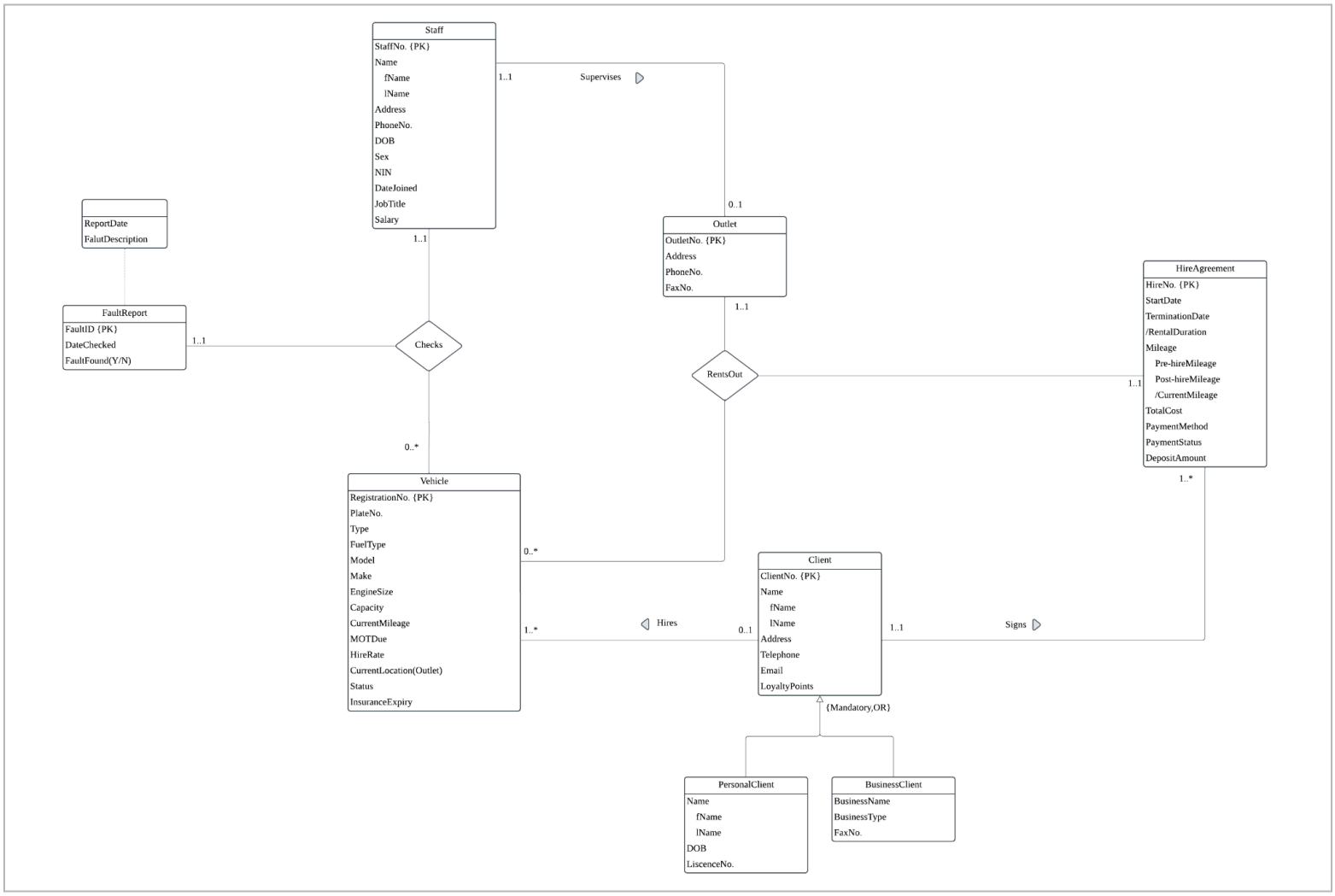
- Employee details (name, contact, job title, assigned outlet) will be recorded for internal management.
- Staff assignments will be updated when employees are transferred between outlets.

5. Third-Party Integration:

- If external services such as insurance verification or payment gateways are integrated, relevant data such as payment status and insurance validation may be fetched from these systems.

4. Data Modelling

4-1. Conceptual Model (ER Diagram)



4-2. Logical Database Design (Relational Database Schema)

Step 1: Derive relations for logical data model

Outlet (OutletNo., Address, PhoneNo., FaxNo., StaffNo.)
Primary Key: OutletNo.
Foreign Key: StaffNo. **references** Staff(StaffNo.)

Vehicle (RegistrationNo., PlateNo., Type, FuelType, Model, Make, EngineSize, Capacity, CurrentMileage, MOTDue, HireRate, CurrentLocation(Outlet), Status, InsuranceExpiry)
Primary Key: RegistrationNo.
Foreign Key: OutletNo. **references** Outlet(OutletNo.)

PersonalClient (ClientNo., Address, Telephone, Email, LoyaltyPoints, fName, lName, DOB, LiscenceNo.)
Primary Key: ClientNo.

BusinessClient (ClientNo., Address, Telephone, Email, LoyaltyPoints, BusinessName, BusinessType, FaxNo.)
Primary Key: ClientNo.

HireAgreement (HireNo., StartDate, TerminationDate, RentalDuration, Pre-hireMileage, Post-hireMileage, CurrentMileage, TotalCost, PaymentMethod, PaymentStatus, DepositAmount, ClientNo., fName, lName, Address, Telephone, RegistrationNo., Model, Make)
Primary Key: HireNo.
Foreign Key: ClientNo. **references** Client(ClientNo.)
Foreign Key: RegistrationNo. **references** Vehicle(RegistrationNo.)

FaultReport (FaultID, fName, lName, DateChecked, FaultFound(Y/N), RegistrationNo., Model, Make, CurrentMileage)
Primary Key: FaultID
Foreign Key: RegistrationNo. **references** Vehicle(RegistrationNo.)
Foreign Key: StaffNo. **references** Staff(StaffNo.)

FaultReporting (FaultID, ReportDate, FaultDescription)
Primary Key: FaultID, ReportDate
Foreign Key: FaultID **references** FaultReport(FaultID)

Staff (StaffNo., fName, lName, Address, PhoneNo., DOB, Sex, NIN, DateJoined, JobTitle, Salary, OutletNo.)
Primary Key: StaffNo.
Foreign Key: OutletNo. **references** Outlet(OutletNo.)

Step 2: Validate relations using normalization

The relations are normalized to Third Normal Form (3NF) to eliminate redundancy and ensure data integrity.

1. Outlet:
    a. All attributes depend on the OutletNo. (primary key).
    b. No partial or transitive dependencies.
2. Vehicle:
    a. All attributes depend on the RegistrationNo. (primary key).
    b. No partial or transitive dependencies.
3. Client:
    a. All attributes depend on the ClientNo. (primary key).
    b. No partial or transitive dependencies.
4. HireAgreement:
    a. All attributes depend on the HireNumber (primary key).
    b. No partial or transitive dependencies.
5. FaultReport:
    a. All attributes depend on the FaultID (primary key).
    b. No partial or transitive dependencies.
6. Staff:
    a. All attributes depend on the StaffNo. (primary key).
    b. No partial or transitive dependencies.

Step 3: Validate relations against user transactions

The relations are validated against the required user transactions (queries) to ensure they can support the system's functionality:

- Vehicle Rental Workflow
    - A client rents a vehicle for a specified duration.
    - System stores the hire agreement details.
- Vehicle Availability Tracking
    - Vehicles change outlets as needed.
    - System maintains current location per vehicle.
- Fault Reporting
    - Staff log inspections for rented vehicles.
- Revenue Tracking
    - The system records rental agreements.
    - Staff can generate reports on daily hire rates.

Step 4: Check integrity constraints

The following integrity constraints are enforced in the logical data model:

- Primary key constraints:
    - Each table has a primary key that uniquely identifies each record.
- Foreign key constraints:
    - Foreign keys ensure referential integrity between related tables.
    - For example, RentalAgreement(ClientNo.) references Client(ClientNo.).
- Domain constraints:
    - Attributes have specific data types
    - For example, FaultFound is a boolean (Yes/No).
- Null constraints:
    - Certain attributes cannot be null (e.g., ClientNo., RegistrationNo.).

Step 5: Review logical data model with use

The logical data model is reviewed with the user to ensure it meets their requirements. The user confirms that:

- All entities, attributes, and relationships are correctly represented.
- The model supports all required transactions.
- The model is flexible enough to accommodate future changes.

Step 6: Merge logical data models into global data model (optional step)

If multiple logical data models exist (e.g., for different departments), they can be merged into a global data model. In this case, the Smiles Car Rental System is a standalone system, so this step is not required and remains optional.

Step 7: Check for future growth

The logical data model is designed to accommodate future growth:

- Additional attributes can be added to tables (e.g., new vehicle features).
- New tables can be introduced (e.g., for promotions or loyalty programs) and tables can be split for large datasets (e.g., HIRE_AGREEMENT by year).
- The model is scalable to handle increased data volume.
- Improve query speed (e.g., RegNUmber, ClientID indexes)

Final Logical Schema

Outlet (OutletNo., Address, PhoneNo., FaxNo., StaffNo.)
Primary Key: OutletNo.
Foreign Key: StaffNo. **references** Staff(StaffNo.)

Vehicle (RegistrationNo., PlateNo., Type, FuelType, Model, Make, EngineSize, Capacity, CurrentMileage, MOTDue, HireRate, CurrentLocation(Outlet), Status, InsuranceExpiry)
Primary Key: RegistrationNo.

Alternate Key: PlateNo.
Foreign Key: OutletNo. **references** Outlet(OutletNo.)

PersonalClient (ClientNo., Address, Telephone, Email, LoyaltyPoints, fName, lName, DOB, LiscenceNo.)
Primary Key: ClientNo.
Alternate Key: LiscenceNo.

BusinessClient (ClientNo., Address, Telephone, Email, LoyaltyPoints, BusinessName, BusinessType, FaxNo.)
Primary Key: ClientNo.

HireAgreement (HireNo., StartDate, TerminationDate, RentalDuration, Pre-hireMileage, Post-hireMileage, CurrentMileage, TotalCost, PaymentMethod, PaymentStatus, DepositAmount, ClientNo., fName, lName, Address, Telephone, RegistrationNo., Model, Make)
Primary Key: HireNo.
Foreign Key: ClientNo. **references** Client(ClientNo.)
Foreign Key: RegistrationNo. **references** Vehicle(RegistrationNo.)

FaultReport (FaultID, fName, lName, DateChecked, FaultFound(Y/N), RegistrationNo., Model, Make, CurrentMileage)
Primary Key: FaultID
Foreign Key: RegistrationNo. **references** Vehicle(RegistrationNo.)
Foreign Key: StaffNo. **references** Staff(StaffNo.)

FaultReporting (FaultID, ReportDate, FaultDescription)
Primary Key: FaultID, ReportDate
Foreign Key: FaultID **references** FaultReport(FaultID)

Staff (StaffNo., fName, lName, Address, PhoneNo., DOB, Sex, NIN, DateJoined, JobTitle, Salary, OutletNo.)
Primary Key: StaffNo.
Foreign Key: OutletNo. **references** Outlet(OutletNo.

5. Possible Queries

1. List all vehicles currently available for rent.
2. Find the total revenue generated from rentals in 2022.
3. Identify customers who rented a car for more than 30 days
4. Find clients who have rented vehicles more than 5 times.
5. List all vehicles that have been involved in more than 3 fault reports.
6. Find the average rental duration for each vehicle model.

7. List all staff members who have not checked any vehicles in the last 6 months.
8. Find the top 5 most profitable vehicles based on total rental income.
9. Find the client who rented the most expensive vehicle in 2023.
10. List all vehicles that have not been rented in the last year.
11. Find the staff member who managed the most rental agreements in 2022.
12. Find the total number of business clients who rented vans in 2023.

6. Implementation of a DB Using MySQL Workbench

6-1. SQL Database Schema

```sql
CREATE DATABASE SMILES;
USE SMILES;

-- Staff table
CREATE TABLE Staff(
  StaffNo INT AUTO_INCREMENT PRIMARY KEY,
  fName VARCHAR(50) NOT NULL,
  lName VARCHAR(50) NOT NULL,
  Address VARCHAR(255),
  PhoneNo VARCHAR(20),
  DOB DATE NOT NULL,
  Sex CHAR(1) CHECK (Sex IN ('F','M')),
  NIN VARCHAR(20) NOT NULL UNIQUE,
  DateJoined DATE,
  JobTitle VARCHAR(20),
  Salary DECIMAL(10,2)
);
```

```sql
-- Outlet table
CREATE TABLE Outlet(
  OutletNo INT AUTO_INCREMENT PRIMARY KEY,
  Address VARCHAR(255),
  PhoneNo VARCHAR(20),
  FaxNo VARCHAR(20),
  Manager INT,
  FOREIGN KEY (Manager) REFERENCES Staff(StaffNo) ON DELETE SET NULL
);

-- Vehicle table
CREATE TABLE Vehicle(
    RegistrationNo VARCHAR(20) PRIMARY KEY,
    PlateNo VARCHAR(20) NOT NULL UNIQUE,
    Type VARCHAR(50),
    FuelType VARCHAR(20),
    Model VARCHAR(20),
    Make VARCHAR(30),
    EngineSize VARCHAR(20),
    Capacity INT,
    CurrentMileage INT,
    MOTDue DATE,
    HireRate DECIMAL(10,2),
    CurrentLocation INT,
    FOREIGN KEY (CurrentLocation) REFERENCES Outlet(OutletNo) ON DELETE SET NULL,
    Status ENUM('Available', 'Hired', 'Under Maintenance', 'Out of Service') DEFAULT 'Available',
    InsuranceExpiry DATE
);
```

```sql
-- Parent Table: Client
CREATE TABLE Client (
    ClientNo INT AUTO_INCREMENT PRIMARY KEY,
    fName VARCHAR(50),
    lName VARCHAR(50),
    Address VARCHAR(255),
    Telephone VARCHAR(20),
    Email VARCHAR(100),
    LoyaltyPoints INT
);

-- Subclass: PersonalClient
CREATE TABLE PersonalClient (
    ClientNo INT PRIMARY KEY,
    DOB DATE NOT NULL,
    LicenceNo VARCHAR(20) NOT NULL UNIQUE,
    FOREIGN KEY (ClientNo) REFERENCES Client(ClientNo) ON DELETE CASCADE
);

-- Subclass: BusinessClient
CREATE TABLE BusinessClient (
    ClientNo INT PRIMARY KEY,
    BusinessName VARCHAR(100) NOT NULL,
    BusinessType VARCHAR(50),
    FaxNo VARCHAR(20),
    FOREIGN KEY (ClientNo) REFERENCES Client(ClientNo) ON DELETE CASCADE
);
```

```sql
-- HireAgreement table
CREATE TABLE HireAgreement (
    HireNo INT AUTO_INCREMENT PRIMARY KEY,
    ClientNo INT NOT NULL,
    VehicleRegNo VARCHAR(20) NOT NULL,
    StartDate DATE NOT NULL,
    TerminationDate DATE NOT NULL,
    RentalDuration INT GENERATED ALWAYS AS (DATEDIFF(TerminationDate, StartDate)) STORED,
    PreHireMileage INT NOT NULL,
    PostHireMileage INT NOT NULL,
    CurrentMileage INT GENERATED ALWAYS AS (PostHireMileage - PreHireMileage) STORED,
    TotalCost DECIMAL(10,2) NOT NULL,
    PaymentMethod ENUM('Cash', 'Credit Card', 'Debit Card', 'Online Transfer') NOT NULL,
    PaymentStatus ENUM('Pending', 'Paid', 'Failed', 'Refunded') DEFAULT 'Pending',
    DepositAmount DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (ClientNo) REFERENCES Client(ClientNo) ON DELETE CASCADE,
    FOREIGN KEY (VehicleRegNo) REFERENCES Vehicle(RegistrationNo) ON DELETE CASCADE
);

-- FaultReport table
CREATE TABLE FaultReport (
    FaultID INT AUTO_INCREMENT PRIMARY KEY,
    VehicleRegNo VARCHAR(20) NOT NULL,
    StaffNo INT NOT NULL,
    ReportDate DATE NOT NULL,
    FaultDescription TEXT NOT NULL,
    DateChecked DATE,
    FaultFound ENUM('Y', 'N') NOT NULL DEFAULT 'N',
    FOREIGN KEY (VehicleRegNo) REFERENCES Vehicle(RegistrationNo) ON DELETE CASCADE,
    FOREIGN KEY (StaffNo) REFERENCES Staff(StaffNo) ON DELETE CASCADE
);
```

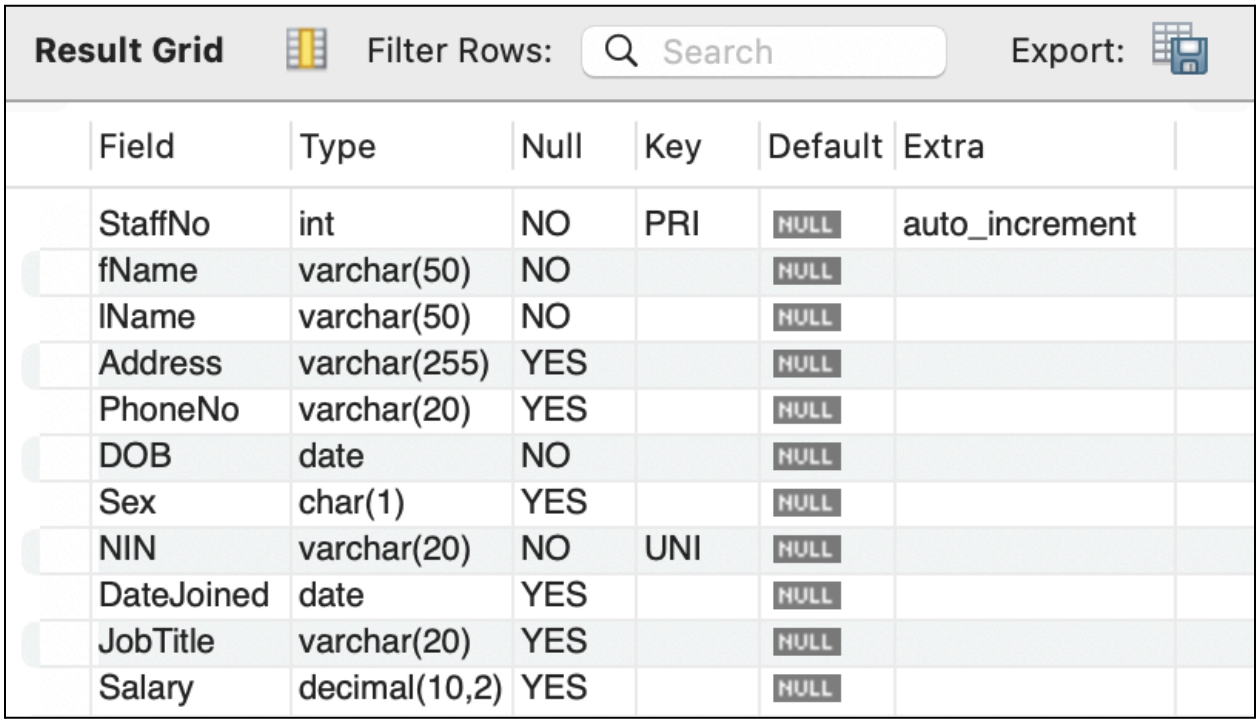
Editor Response:

Action Output

	Time	Action	Response	Duration / Fetch Time
1	11:31:21	CREATE DATABASE SMILES	1 row(s) affected	0.058 sec
2	11:31:24	USE SMILES	0 row(s) affected	0.0023 sec
3	11:31:28	CREATE TABLE Staff(StaffNo INT AUTO_INCREMENT PRIMARY KEY, fName VARCHA...	0 row(s) affected	0.031 sec
4	11:31:32	CREATE TABLE Outlet(OutletNo INT AUTO_INCREMENT PRIMARY KEY, Address VAR...	0 row(s) affected	0.016 sec
5	11:31:36	CREATE TABLE Vehicle(RegistrationNo VARCHAR(20) PRIMARY KEY, PlateNo VAR...	0 row(s) affected	0.014 sec
6	11:31:40	CREATE TABLE Client (ClientNo INT AUTO_INCREMENT PRIMARY KEY, fName VA...	0 row(s) affected	0.0071 sec
7	11:31:44	CREATE TABLE PersonalClient (ClientNo INT PRIMARY KEY, DOB DATE NOT NULL...	0 row(s) affected	0.0093 sec
8	11:31:48	CREATE TABLE BusinessClient (ClientNo INT PRIMARY KEY, BusinessName VARC...	0 row(s) affected	0.012 sec
9	11:31:52	CREATE TABLE HireAgreement (HireNo INT AUTO_INCREMENT PRIMARY KEY, Cl...	0 row(s) affected	0.018 sec
10	11:31:57	CREATE TABLE FaultReport (FaultID INT AUTO_INCREMENT PRIMARY KEY, Vehi...	0 row(s) affected	0.020 sec

6-2. Relation schemas Descriptions

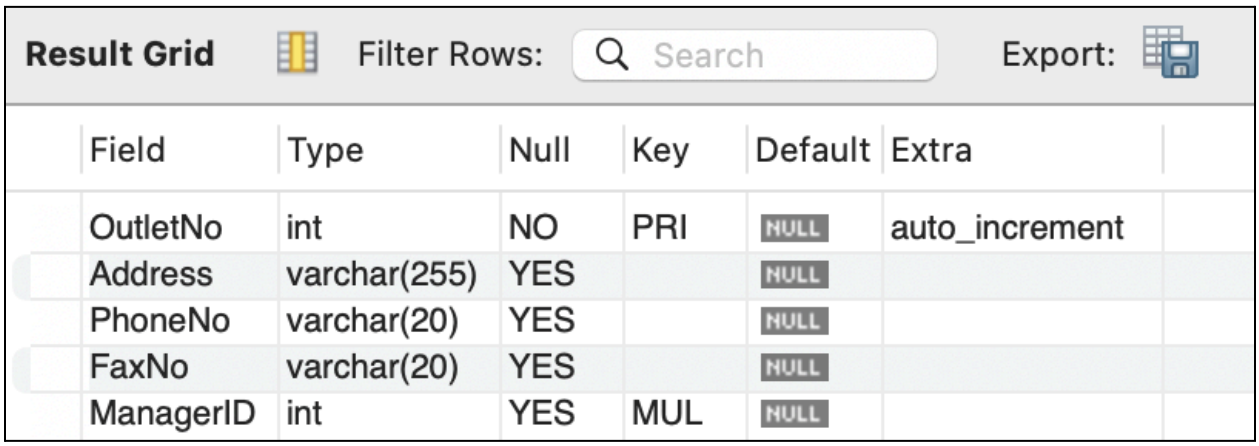
Staff



Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
StaffNo	int	NO	PRI	NULL	auto_increment
fName	varchar(50)	NO		NULL	
lName	varchar(50)	NO		NULL	
Address	varchar(255)	YES		NULL	
PhoneNo	varchar(20)	YES		NULL	
DOB	date	NO		NULL	
Sex	char(1)	YES		NULL	
NIN	varchar(20)	NO	UNI	NULL	
DateJoined	date	YES		NULL	
JobTitle	varchar(20)	YES		NULL	
Salary	decimal(10,2)	YES		NULL	

Outlet



Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
OutletNo	int	NO	PRI	NULL	auto_increment
Address	varchar(255)	YES		NULL	
PhoneNo	varchar(20)	YES		NULL	
FaxNo	varchar(20)	YES		NULL	
ManagerID	int	YES	MUL	NULL	

Vehicle

Result Grid Filter Rows: Search Export:

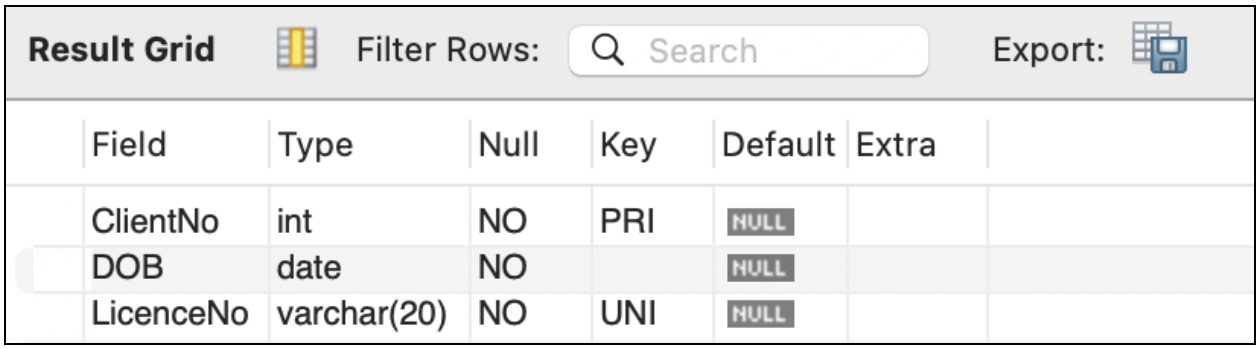
Field	Type	Null	Key	Default	Extra
RegistrationNo	varchar(20)	NO	PRI	NULL	
PlateNo	varchar(20)	NO	UNI	NULL	
Type	varchar(50)	YES		NULL	
FuelType	varchar(20)	YES		NULL	
Model	varchar(20)	YES		NULL	
Make	varchar(30)	YES		NULL	
EngineSize	varchar(20)	YES		NULL	
Capacity	int	YES		NULL	
CurrentMileage	int	YES		NULL	
MOTDue	date	YES		NULL	
HireRate	decimal(1…	YES		NULL	
CurrentLocation	int	YES	MUL	NULL	
Status	enum('Av…	YES		Avail…	
InsuranceExpiry	date	YES		NULL	

Client

Result Grid Filter Rows: Search Export:

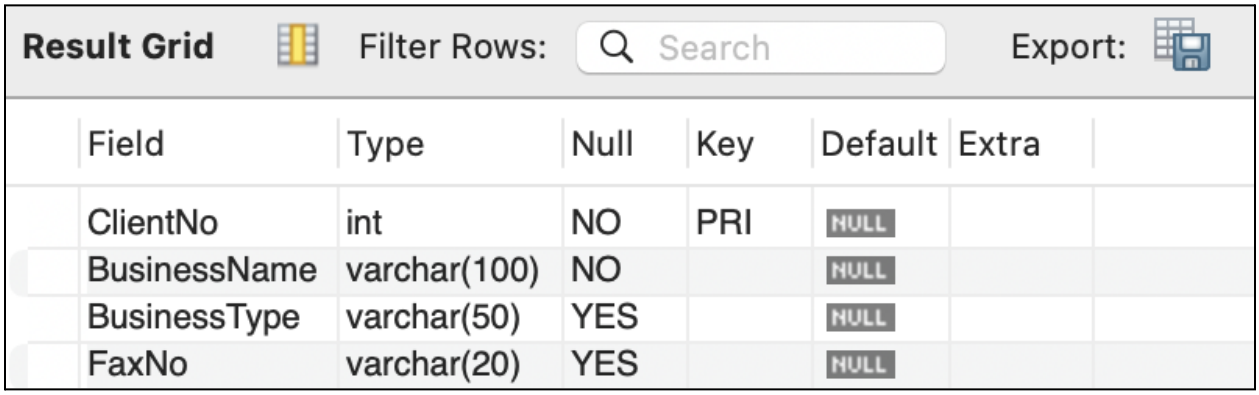
Field	Type	Null	Key	Default	Extra
ClientNo	int	NO	PRI	NULL	auto_increment
fName	varchar(50)	YES		NULL	
lName	varchar(50)	YES		NULL	
Address	varchar(255)	YES		NULL	
Telephone	varchar(20)	YES		NULL	
Email	varchar(100)	YES		NULL	
LoyaltyPoints	int	YES		NULL	

PersonalClient



Field	Type	Null	Key	Default	Extra
ClientNo	int	NO	PRI	NULL	
DOB	date	NO		NULL	
LicenceNo	varchar(20)	NO	UNI	NULL	

BusinessClient



Field	Type	Null	Key	Default	Extra
ClientNo	int	NO	PRI	NULL	
BusinessName	varchar(100)	NO		NULL	
BusinessType	varchar(50)	YES		NULL	
FaxNo	varchar(20)	YES		NULL	

HireAgreement

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
HireNo	int	NO	PRI	NULL	auto_increment
ClientNo	int	NO	MUL	NULL	
VehicleRegNo	varchar(20)	NO	MUL	NULL	
StartDate	date	NO		NULL	
TerminationDate	date	NO		NULL	
RentalDuration	int	YES		NULL	STORED GENERATED
PreHireMileage	int	NO		NULL	
PostHireMileage	int	NO		NULL	
CurrentMileage	int	YES		NULL	STORED GENERATED
TotalCost	decimal(10,2)	NO		NULL	
PaymentMethod	enum('Cash…	NO		NULL	
PaymentStatus	enum('Pend…	YES		Pendi…	
DepositAmount	decimal(10,2)	NO		NULL	

FaultReport

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
FaultID	int	NO	PRI	NULL	auto_increment
VehicleRegNo	varchar(20)	NO	MUL	NULL	
StaffNo	int	NO	MUL	NULL	
ReportDate	date	NO		NULL	
FaultDescription	text	NO		NULL	
DateChecked	date	YES		NULL	
FaultFound	enum('Y','N')	NO		N	

7. Insert Commands

7-1. Basic Insert Command

SQL Query

```
INSERT INTO Staff (fName, lName, Address, PhoneNo, DOB, Sex, NIN, DateJoined, JobTitle, Salary)
VALUES ('Yunseo', 'Heo', '28 College Ave West', '98129770', '2002-04-26',
        'F', 'NIN12345', '2021-08-17', 'Associate', 50000.00);
```

Editor Response

```
21  13:27:14  INSERT INTO Staff (fName, lName, Address, PhoneNo, DOB, Sex, NIN, DateJoined, Job...  1 row(s) affected  0.026 sec
```

SELECT * FROM Staff

Result Grid Filter Rows: Search Edit: Export/Import:

StaffNo	fName	lName	Address	PhoneNo	DOB	Sex	NIN	DateJoined	JobTitle	Salary
1	Yunseo	Heo	28 College Ave West	98129770	2002-04-26	F	NIN12345	2021-08-17	Associate	50000.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

7-2. Insert Based on Existing Data

- Can be used when we want to duplicate most of the details from an existing entry
- In this case, let's say we want to insert a new staff member whose information is all the same except for their name, NIN, date joined, job title, and salary.

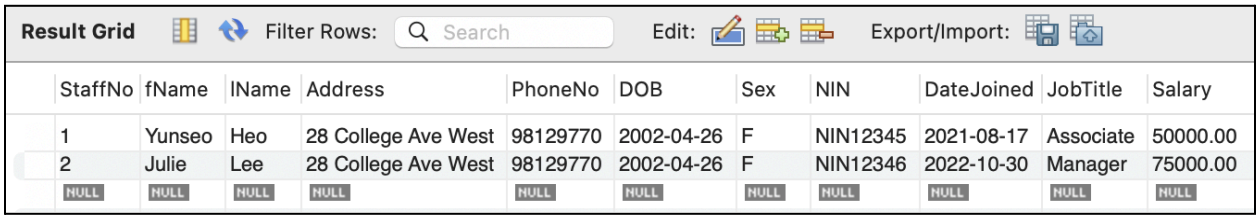
SQL Query

```
INSERT INTO Staff (fName, lName, Address, PhoneNo, DOB, Sex, NIN, DateJoined, JobTitle, Salary)
SELECT 'Julie', 'Lee',  Address, PhoneNo, DOB, Sex, 'NIN12346', '2022-10-30', 'Manager', Salary * 1.5
FROM Staff
WHERE NIN = 'NIN12345';
```

Editor Response

```
52  14:03:57  INSERT INTO Staff (fName, lName, Address, PhoneNo, DOB, Sex, NIN, DateJoined, Job...  1 row(s) affected Records: 1  Duplicates: 0  Warnings: 0  0.0033 sec
```

SELECT * FROM Staff



Result Grid | Filter Rows: Search | Edit: | Export/Import:

StaffNo	fName	lName	Address	PhoneNo	DOB	Sex	NIN	DateJoined	JobTitle	Salary
1	Yunseo	Heo	28 College Ave West	98129770	2002-04-26	F	NIN12345	2021-08-17	Associate	50000.00
2	Julie	Lee	28 College Ave West	98129770	2002-04-26	F	NIN12346	2022-10-30	Manager	75000.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

7-3. INSERT Based on Conditions

- Can be used when we want to insert data only if it satisfies a certain condition.
- In this case, it can be used to generate fault reports for vehicles with mileage over 100,000 automatically.

SQL Query

```
INSERT INTO FaultReport (VehicleRegNo, StaffNo, ReportDate, FaultDescription, FaultFound)
SELECT v.RegistrationNo, 1, '2025-02-15', 'Automatic Inspection Required: High mileage vehicle', 'Y'
FROM Vehicle AS v
WHERE v.CurrentMileage > 100000;
```

Editor Response

54 14:45:52 INSERT INTO FaultReport (VehicleRegNo, StaffNo, ReportDate, FaultDescription, FaultF... 0 row(s) affected Records: 0 Duplicates: 0 Warnings... 0.053 sec

- 0 rows are affected as we haven't inserted any data into the vehicle table.

8. Data Insertion

To populate our database, we wrote a Python script that automatically generates and inserts data using MySQL queries, instead of manually entering records one by one. The script ensures data integrity by adhering to foreign key constraints and generating meaningful test cases with Faker.

```python
import mysql.connector
from faker import Faker
import random

# Initialize Faker
fake = Faker()

# Connect to MySQL Database
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="0426Mandy+",
    database="SMILES"
)
cursor = conn.cursor()


# Function to Insert Clients
def insert_clients(n=100):
    try:
        cursor.execute("SELECT COUNT(*) FROM Client")
        existing_clients = cursor.fetchone()[0]

        if existing_clients >= n:
            print("Clients already exist, skipping insertion.")
            return

        for _ in range(existing_clients, n):
            fName = fake.first_name()
            lName = fake.last_name()
            address = fake.address().replace("\n", ", ")
            phone = fake.phone_number()[:10]
            email = fake.email()
            loyalty = random.randint(0, 500)

            cursor.execute("""
                INSERT INTO Client (fName, lName, Address, Telephone, Email, LoyaltyPoints)
                VALUES (%s, %s, %s, %s, %s, %s)
            """, (fName, lName, address, phone, email, loyalty))

        conn.commit()
        print(f"Clients Inserted Successfully!")
    except mysql.connector.Error as e:
        print(f"Error inserting Clients: {e}")
```

```python
# Function to Insert Outlets
def insert_outlets(n=30):
    try:
        cursor.execute("SELECT COUNT(*) FROM Outlet")
        existing_outlets = cursor.fetchone()[0]

        if existing_outlets >= n:
            print("Outlets already exist, skipping insertion.")
            return

        for i in range(existing_outlets + 1, existing_outlets + n + 1):
            address = fake.address().replace("\n", ", ")
            phone = fake.phone_number()[:10]
            fax = fake.phone_number()[:10]

            cursor.execute("""
                INSERT INTO Outlet (OutletNo, Address, PhoneNo, FaxNo)
                VALUES (%s, %s, %s, %s)
            """, (i, address, phone, fax))

        conn.commit()
        print(f"Outlets Inserted Successfully!")
    except mysql.connector.Error as e:
        print(f"Error inserting Outlets: {e}")
```

```python
# Function to Insert Vehicles
def insert_vehicles(n=50):
    try:
        cursor.execute("SELECT COUNT(*) FROM Vehicle")
        existing_vehicles = cursor.fetchone()[0]

        cursor.execute("SELECT OutletNo FROM Outlet")
        outlets = [row[0] for row in cursor.fetchall()]

        if not outlets:
            raise ValueError("No outlets available! Insert outlets first.")

        for i in range(existing_vehicles, existing_vehicles + n):
            reg_no = f"V{1000 + i}"  # Ensures unique registration numbers
            plate_no = f"SGX{random.randint(1000, 9999)}{random.choice(['A', 'B', 'C', 'D'])}"
            vehicle_type = random.choice(["Car", "Van", "SUV"])
            fuel_type = random.choice(["Petrol", "Diesel", "Electric"])
            model = fake.word()
            make = fake.company()
            engine_size = f"{random.uniform(1.0, 3.0):.1f}L"
            capacity = random.randint(2, 8)
            mileage = random.randint(5000, 150000)
            mot_due = fake.future_date()
            hire_rate = round(random.uniform(50, 200), 2)
            location = random.choice(outlets)
            status = random.choice(["Available", "Hired", "Under Maintenance"])
            insurance_expiry = fake.future_date()

            cursor.execute("""
                INSERT INTO Vehicle (RegistrationNo, PlateNo, Type, FuelType, Model, Make,
                                     EngineSize, Capacity, CurrentMileage, MOTDue, HireRate, CurrentLocation, Status, InsuranceExpiry)
                VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
            """, (reg_no, plate_no, vehicle_type, fuel_type, model, make, engine_size, capacity, mileage, mot_due, hire_rate, location, status, insurance_expiry))

        conn.commit()
        print(f"Vehicles Inserted Successfully!")
    except mysql.connector.Error as e:
        print(f"Error inserting Vehicles: {e}")
```

```python
# Function to Insert Hire Agreements
def insert_hire_agreements(n=50):
    try:
        cursor.execute("SELECT COUNT(*) FROM HireAgreement")
        existing_agreements = cursor.fetchone()[0]

        cursor.execute("SELECT ClientNo FROM Client")
        clients = [row[0] for row in cursor.fetchall()]

        cursor.execute("SELECT RegistrationNo FROM Vehicle WHERE Status = 'Available'")
        vehicles = [row[0] for row in cursor.fetchall()]

        if not clients or not vehicles:
            raise ValueError("No Clients or Available Vehicles found! Insert data first.")

        for _ in range(existing_agreements, existing_agreements + n):
            client_no = random.choice(clients)
            vehicle_reg_no = random.choice(vehicles)

            start_date = fake.date_between(start_date="-60d", end_date="today")
            termination_date = fake.date_between(start_date=start_date, end_date="+30d")
            pre_hire_mileage = random.randint(5000, 100000)
            post_hire_mileage = pre_hire_mileage + random.randint(100, 500)
            total_cost = round(random.uniform(100, 600), 2)
            payment_method = random.choice(["Cash", "Credit Card", "Debit Card", "Online Transfer"])
            payment_status = random.choice(["Pending", "Paid", "Failed", "Refunded"])
            deposit_amount = round(random.uniform(50, 250), 2)

            cursor.execute("""
                INSERT INTO HireAgreement (ClientNo, VehicleRegNo, StartDate, TerminationDate, PreHireMileage,
                                           PostHireMileage, TotalCost, PaymentMethod, PaymentStatus, DepositAmount)
                VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
            """, (client_no, vehicle_reg_no, start_date, termination_date, pre_hire_mileage, post_hire_mileage,
                  total_cost, payment_method, payment_status, deposit_amount))

        conn.commit()
        print(f"Hire Agreements Inserted Successfully!")
    except mysql.connector.Error as e:
        print(f"Error inserting Hire Agreements: {e}")
```

```python
# Populate Database (Insert Data in the Correct Order)
if __name__ == "__main__":
    insert_clients(100)
    insert_outlets(30)
    insert_vehicles(50)
    insert_hire_agreements(50)

    cursor.close()
    conn.close()
    print(" Data Insertion Completed! ✅")
```

Sample records to verify successful data entry:

1. Client

```
141    -- Sample Records
142 •  SELECT * FROM Client LIMIT 5;
```

100% 1:140 1 error found

Result Grid Filter Rows: Search Edit: Export/Import: Fetch rows:

ClientNo	fName	lName	Address	Telephone	Email	LoyaltyPoints
1	Aaron	Harris	7176 Jeffrey Mill Apt. 541, Marquezland, MS 58…	351.837.61	robinsoncurtis@example.com	441
2	Karen	Romero	Unit 1038 Box 3290, DPO AE 20623	001-249-46	donalddawson@example.org	113
3	Evelyn	Carpenter	580 Kenneth Groves, Jamesfort, ID 40595	5343668485	francescollins@example.com	450
4	Lisa	Fox	3608 Russell Mountain, East Josetown, AZ 74293	985.722.09	staylor@example.net	33
5	Jeffrey	Perez	662 Butler Throughway Apt. 354, North Breanna…	001-318-63	bcarney@example.com	249
NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. Outlet

```
141    -- Sample Records
142 •  SELECT * FROM Outlet LIMIT 5;
```

100% 18:138 1 error found

Result Grid Filter Rows: Search Edit: Export/Import: Fetch rows:

OutletNo	Address	PhoneNo	FaxNo	Manager
1	495 Reilly Lock Suite 702, Martinezmouth, WA 6…	001-864-98	001-497-63	NULL
2	684 Knight Road, Anthonyborough, MA 60552	+1-696-811	582-433-82	NULL
3	874 Parsons Meadows, Kimberlyberg, LA 30767	+1-367-339	794-524-09	NULL
4	59985 Martha Groves, Kristineport, CA 66148	+1-744-346	984.307.95	NULL
5	6750 Ashley Isle Suite 485, Alyssahaven, AL 87…	001-351-93	665.665.49	NULL
NULL	NULL	NULL	NULL	NULL

3. Vehicle

```
141    -- Sample Records
142 •  SELECT * FROM Vehicle LIMIT 5;
```

100% 15:139 1 error found

Result Grid Filter Rows: Search Edit: Export/Import: Fetch rows:

egistrationNo	PlateNo	Type	FuelType	Model	Make	EngineSize	Capacity	CurrentMileage	MOTDue	HireRate	CurrentLocati…	Status	InsuranceExpiry
1000	SGX7743B	SUV	Petrol	mention	Thomas Ltd	2.8L	3	109422	2025-03-14	146.42	4	Under Maintenance	2025-02-21
1001	SGX3902C	Van	Petrol	thousand	Gross Ltd	1.6L	5	53248	2025-03-08	128.77	2	Hired	2025-03-12
1002	SGX6509C	SUV	Electric	political	Bautista Inc	1.6L	4	122548	2025-02-24	112.87	15	Available	2025-02-26
1003	SGX6574B	SUV	Petrol	consumer	Hernandez-Huynh	1.2L	8	127480	2025-03-07	104.86	15	Under Maintenance	2025-02-17
1004	SGX7692A	SUV	Diesel	get	Hood-Barrera	2.8L	3	47789	2025-03-02	87.98	6	Available	2025-03-09
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid Form Editor

4. Hire Agreement

```
141     -- Sample Records
142 •   SELECT * FROM HireAgreement LIMIT 5;
```

100% 12:139 1 error found

Result Grid Filter Rows: Search Edit: Export/Import: Fetch rows:

HireNo	ClientNo	VehicleRegNo	StartDate	TerminationDa...	RentalDuration	PreHireMileage	PostHireMileage	CurrentMileage	TotalCost	PaymentMethod	PaymentStatus	DepositAmou...
1	19	V1022	2025-01-21	2025-03-15	53	56852	57131	279	570.70	Online Transfer	Paid	203.39
2	61	V1007	2025-01-29	2025-03-05	35	45060	45396	336	152.38	Online Transfer	Refunded	240.31
3	16	V1005	2025-02-03	2025-02-24	21	30899	31212	313	146.02	Credit Card	Refunded	246.68
4	38	V1014	2024-12-27	2025-02-28	63	8745	8977	232	213.62	Debit Card	Paid	190.22
5	7	V1014	2024-12-31	2025-03-02	61	6734	7155	421	522.92	Credit Card	Paid	158.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

9. SELECT-FROM-WHERE Commands

9-1. Basic format

SQL Query

```
SELECT RegistrationNo, PlateNo, Type, FuelType, Model, Make, HireRate
FROM Vehicle
WHERE Status = 'Available'
LIMIT 10;
```

Show all vehicles that are currently available for hire

Result

Result Grid Filter Rows: Search Edit: Export/Import: Fetch rows:

RegistrationNo	PlateNo	Type	FuelType	Model	Make	HireRate
V1002	SGX6509C	SUV	Electric	political	Bautista Inc	112.87
V1004	SGX7692A	SUV	Diesel	get	Hood-Barrera	87.98
V1005	SGX9656C	SUV	Diesel	economic	Cruz-Torres	84.04
V1007	SGX4054B	Car	Electric	effort	Cooper-Chang	135.19
V1013	SGX6359D	Car	Diesel	fact	Webster Inc	92.73
V1014	SGX3890A	Car	Petrol	our	Franklin Inc	127.17
V1017	SGX5604A	Van	Electric	act	Baker-Bowen	170.94
V1022	SGX3596B	Van	Petrol	realize	Lewis-Webb	77.93
V1027	SGX4633A	Car	Petrol	nearly	Grimes-Gutierrez	162.35
V1035	SGX5980A	Car	Electric	very	Martin, Cooper and Reed	51.10
NULL	NULL	NULL	NULL	NULL	NULL	NULL

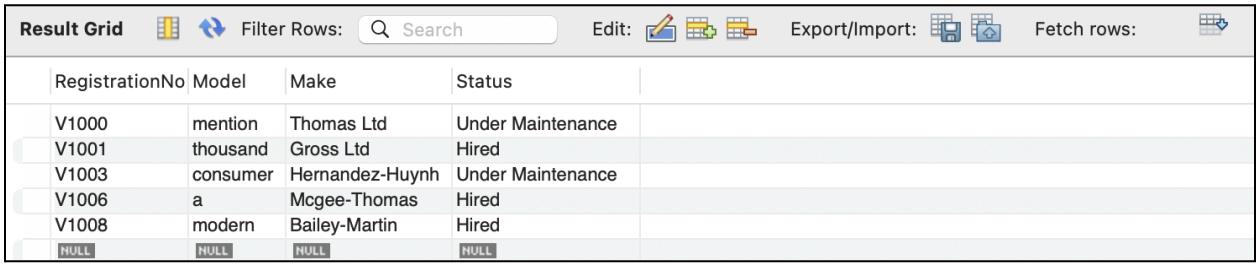
9-2. Subquery in WHERE command

SQL Query

```
SELECT V.RegistrationNo, V.Model, V.Make, V.Status
FROM Vehicle AS V
WHERE V.RegistrationNo NOT IN (SELECT DISTINCT VehicleRegNo FROM HireAgreement)
LIMIT 5;
```

Find vehicles that have never been rented

Result



RegistrationNo	Model	Make	Status
V1000	mention	Thomas Ltd	Under Maintenance
V1001	thousand	Gross Ltd	Hired
V1003	consumer	Hernandez-Huynh	Under Maintenance
V1006	a	Mcgee-Thomas	Hired
V1008	modern	Bailey-Martin	Hired
NULL	NULL	NULL	NULL

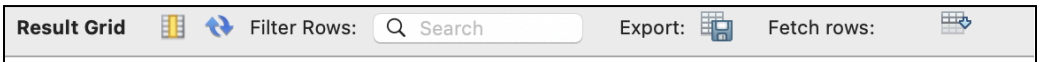
9-3. JOIN Multiple Tables

SQL Query

```
SELECT H.HireNo, C.fName, C.lName, V.Model, V.Make, H.StartDate, H.TerminationDate, H.TotalCost
FROM HireAgreement AS H
INNER JOIN Client AS C ON H.ClientNo = C.ClientNo
INNER JOIN Vehicle AS V ON H.VehicleRegNo = V.RegistrationNo
LIMIT 10;
```

List the vehicle rentals, client names, and vehicle details at the same time.

Result



HireNo	fName	lName	Model	Make	StartDate	TerminationDa...	TotalCost
1	Erika	Brown	realize	Lewis-Webb	2025-01-21	2025-03-15	570.70
2	Marcia	Summers	effort	Cooper-Chang	2025-01-29	2025-03-05	152.38
3	Megan	Webster	economic	Cruz-Torres	2025-02-03	2025-02-24	146.02
4	Keith	Gordon	our	Franklin Inc	2024-12-27	2025-02-28	213.62
5	Ethan	Sanchez	our	Franklin Inc	2024-12-31	2025-03-02	522.92
6	Jeffrey	Perez	get	Hood-Barrera	2024-12-29	2025-01-27	230.56
7	Brad	Wilson	very	Martin, Cooper and Reed	2025-02-09	2025-02-24	364.38
8	Brad	Wilson	economic	Cruz-Torres	2025-01-19	2025-01-28	568.28
9	Patrick	Weber	economic	Cruz-Torres	2025-01-16	2025-02-09	556.14
10	Thomas	Morris	our	Franklin Inc	2025-01-20	2025-02-26	540.70

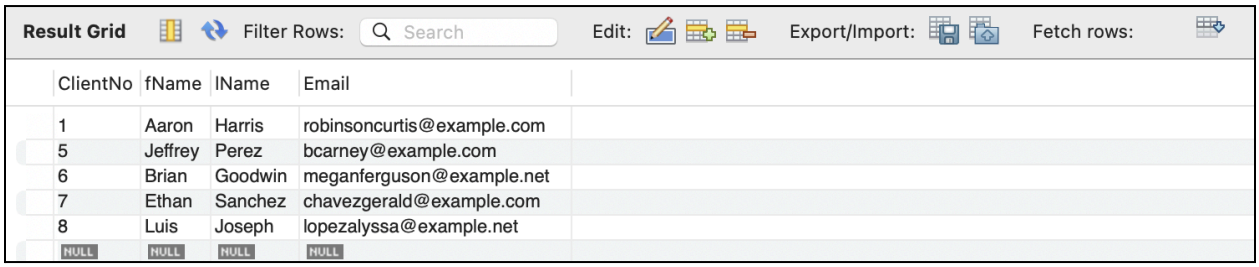
9-4. EXISTS

SQL Query

```
SELECT C.ClientNo, C.fName, C.lName, C.Email
FROM Client AS C
WHERE EXISTS (SELECT 1 FROM HireAgreement AS H WHERE H.ClientNo = C.ClientNo)
LIMIT 5;
```

Find clients who have rented at least one vehicle.

Result



ClientNo	fName	lName	Email
1	Aaron	Harris	robinsoncurtis@example.com
5	Jeffrey	Perez	bcarney@example.com
6	Brian	Goodwin	meganferguson@example.net
7	Ethan	Sanchez	chavezgerald@example.com
8	Luis	Joseph	lopezalyssa@example.net
NULL	NULL	NULL	NULL

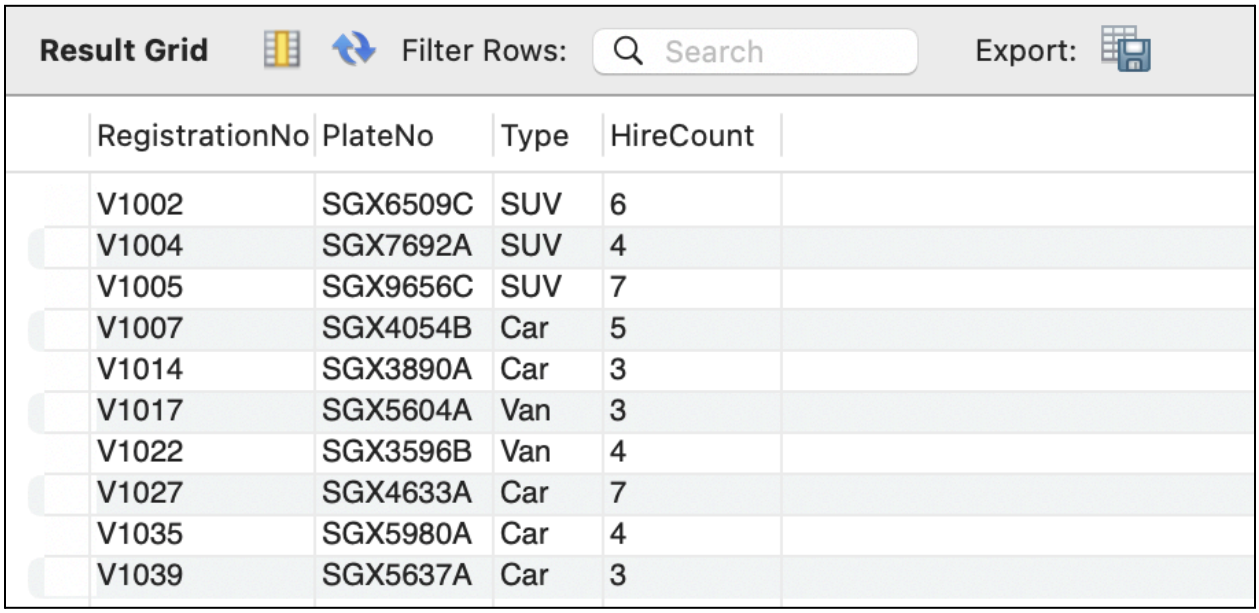
9-5. GROUP BY and HAVING

SQL Query

```
SELECT V.RegistrationNo, V.PlateNo, V.Type, COUNT(H.VehicleRegNo) AS HireCount
FROM HireAgreement AS H
JOIN Vehicle AS V ON H.VehicleRegNo = V.RegistrationNo
GROUP BY V.RegistrationNo, V.PlateNo, V.Type
HAVING COUNT(H.VehicleRegNo) >= 3;
```

Find the number of times each vehicle type has been hired, but only show types that have been hired at least 3 times.

Result



RegistrationNo	PlateNo	Type	HireCount
V1002	SGX6509C	SUV	6
V1004	SGX7692A	SUV	4
V1005	SGX9656C	SUV	7
V1007	SGX4054B	Car	5
V1014	SGX3890A	Car	3
V1017	SGX5604A	Van	3
V1022	SGX3596B	Van	4
V1027	SGX4633A	Car	7
V1035	SGX5980A	Car	4
V1039	SGX5637A	Car	3

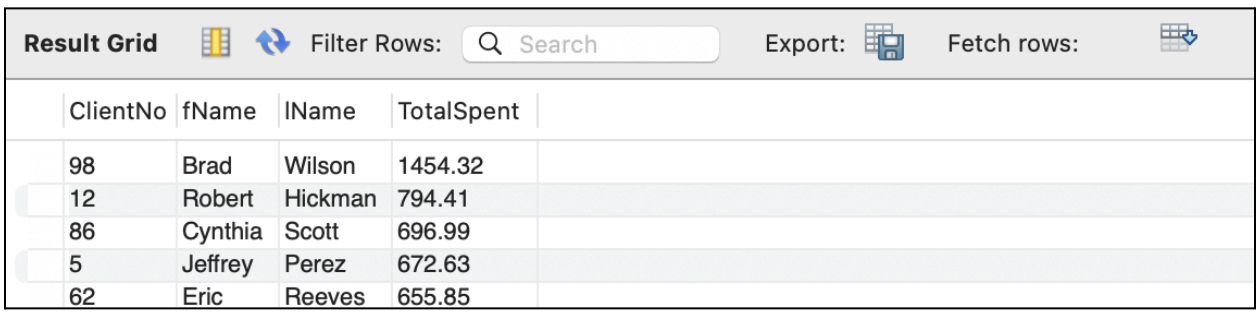
9-6. JOIN with ORDER BY

SQL Query

```sql
SELECT C.ClientNo, C.fName, C.lName, SUM(H.TotalCost) AS TotalSpent
FROM Client AS C
JOIN HireAgreement AS H ON C.ClientNo = H.ClientNo
GROUP BY C.ClientNo, C.fName, C.lName
ORDER BY TotalSpent DESC
LIMIT 5;
```

Find the top 5 highest-paying customers based on total rental cost.

Result



ClientNo	fName	lName	TotalSpent
98	Brad	Wilson	1454.32
12	Robert	Hickman	794.41
86	Cynthia	Scott	696.99
5	Jeffrey	Perez	672.63
62	Eric	Reeves	655.85

10. Data Modification Commands

10-1. Inserting the Result of a Query

Scenario: we want to insert loyal clients who have rented more than 3 times into a VIPClients table.

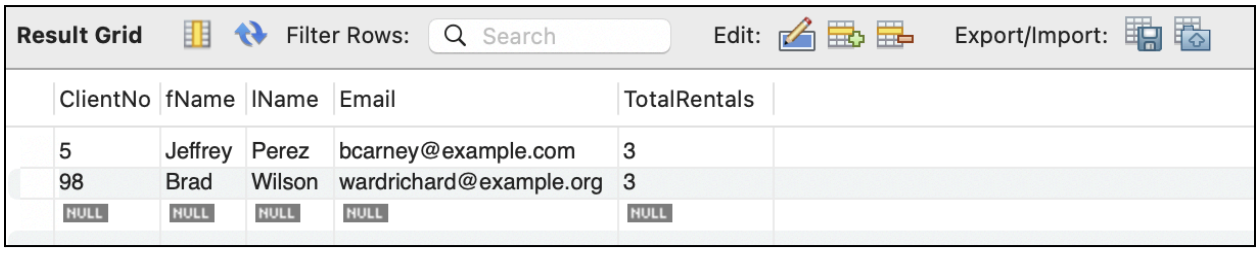
Create the VIPClients table to store data

```
CREATE TABLE VIPClients (
    ClientNo INT PRIMARY KEY,
    fName VARCHAR(50),
    lName VARCHAR(50),
    Email VARCHAR(100),
    TotalRentals INT
);
```

Insert only clients who have rented at least 3 times

```
INSERT INTO VIPClients (ClientNo, fName, lName, Email, TotalRentals)
SELECT C.ClientNo, C.fName, C.lName, C.Email, COUNT(H.HireNo) AS TotalRentals
FROM Client AS C
JOIN HireAgreement H ON C.ClientNo = H.ClientNo
GROUP BY C.ClientNo, C.fName, C.lName, C.Email
HAVING COUNT(H.HireNo) >= 3;
```

Result



ClientNo	fName	lName	Email	TotalRentals
5	Jeffrey	Perez	bcarney@example.com	3
98	Brad	Wilson	wardrichard@example.org	3
NULL	NULL	NULL	NULL	NULL

10-2. Updating Several Tuples At Once

Scenario: We want to increase the rental cost by 10% for vehicles that have been rented more than 3 times.

SQL Query

```sql
UPDATE Vehicle AS V
JOIN (
    SELECT VehicleRegNo, COUNT(*) AS RentalCount
    FROM HireAgreement
    GROUP BY VehicleRegNo
    HAVING COUNT(*) > 3
) AS FrequentRentals
ON V.RegistrationNo = FrequentRentals.VehicleRegNo
SET V.HireRate = ROUND(V.HireRate * 1.10, 2);
```

Editor Response

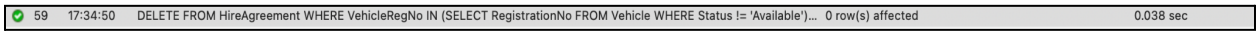
10-3. Delete a Set of Tuples

Scenario: We want to delete rental agreements written before 2023, but only for vno longer available.

SQL Query

```sql
DELETE FROM HireAgreement
WHERE VehicleRegNo IN (SELECT RegistrationNo FROM Vehicle WHERE Status != 'Available')
AND StartDate < '2023-01-01';
```

Editor Response



11. Creating Views

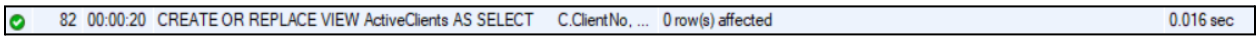
11-1. ActiveClients

This view retrieves clients with active rentals, meaning their rental period is still ongoing or ended within the last 30 days.

SQL Query

```sql
CREATE OR REPLACE VIEW ActiveClients AS
SELECT
    C.ClientNo,
    C.fName,
    C.lName,
    C.Email,
    H.VehicleRegNo,
    H.StartDate,
    H.TerminationDate
FROM Client C
JOIN HireAgreement H ON C.ClientNo = H.ClientNo
WHERE H.TerminationDate > (CURDATE() - INTERVAL 30 DAY);
```

Editor Response



11-2. AvailableVehicles

This view returns a list of vehicles that are available for hire, filtering only those marked as 'Available' in the Status column.

SQL Query

```sql
CREATE OR REPLACE VIEW AvailableVehicles AS
SELECT
    RegistrationNo,
    PlateNo,
    Type,
    Model,
    Make,
    HireRate
FROM Vehicle
WHERE Status = 'Available';
```

Editor Response

```
83  00:00:52  CREATE OR REPLACE VIEW AvailableVehicles AS SELECT  Registr...  0 row(s) affected  0.000 sec
```

11-3. HighSpendingClients

This view identifies clients who have spent more than $500 on rentals.

SQL Query

```sql
CREATE OR REPLACE VIEW HighSpendingClients AS
SELECT
    c.ClientNo,
    c.fName,
    c.lName,
    SUM(h.TotalCost) AS TotalSpent
FROM Client c
JOIN HireAgreement h ON c.ClientNo = h.ClientNo
GROUP BY c.ClientNo
HAVING TotalSpent > 500;
```

Editor Response

```
84  00:01:16  CREATE OR REPLACE VIEW HighSpendingClients AS SELECT  c.Cli...  0 row(s) affected  0.015 sec
```

11-4. Running the Views

SQL Query

```sql
-- Running Queries on Views
SELECT * FROM ActiveClients;
SELECT * FROM AvailableVehicles ORDER BY HireRate DESC;
SELECT * FROM HighSpendingClients;
```

ActiveClients:

ClientNo	fName	lName	Email	VehicleRegNo	StartDate	TerminationDate
83	Mariah	Reyes	emily06@example.com	V1044	2025-01-07	2025-03-01
10	Daniel	Brown	lbarker@example.com	V1035	2025-01-21	2025-02-28
29	Daniel	Irwin	amy63@example.net	V1044	2025-02-05	2025-03-17
84	Bridget	Guerra	ayoung@example.net	V1044	2025-02-07	2025-03-11
25	Nicholas	Benson	sarahbrown@example.org	V1044	2025-02-08	2025-02-15
64	Christopher	Wood	marshallkevin@example.net	V1006	2025-01-25	2025-02-21
21	Travis	Wheeler	shawnsmith@example.net	V1039	2025-02-07	2025-02-16
86	Rebecca	Thompson	nnovak@example.net	V1025	2025-01-04	2025-02-26
58	Brandon	Williams	codyhull@example.com	V1045	2025-02-07	2025-03-10
54	Jennifer	Giles	lisabowman@example.com	V1002	2025-01-31	2025-02-03

AvailableVehicles:

RegistrationNo	PlateNo	Type	Model	Make	HireRate
V1014	SGX9896A	SUV	win	Haley Inc	89.97
V1048	SGX7406A	Car	then	Grant, Perkins and Mckinney	77.44
V1006	SGX3432B	Van	apply	Perkins, Lee and Williams	77.28
V1002	SGX3914B	Van	society	Copeland-Melendez	76.55
V1034	SGX4697C	Car	without	Hudson, Patel and Knight	73.03
V1026	SGX2234D	Van	since	Duke, Phillips and Roberson	66.33
V1023	SGX8595A	Van	man	Freeman, Porter and Ramirez	58.17
V1044	SGX8270A	Van	yourself	Warner-Stevens	53.83
V1039	SGX8318D	SUV	government	Farrell Ltd	50.05

HighSpendingClients:

ClientNo	fName	lName	TotalSpent
5	Tammy	Stanley	1167.20
84	Bridget	Guerra	510.28
74	Nichole	Franklin	629.48
21	Travis	Wheeler	1045.67
58	Brandon	Williams	505.15
28	Margaret	Knapp	661.48
3	Austin	Nelson	510.41
79	Karen	Gonzalez	662.47
47	Kimberly	Curry	509.18
66	Alyssa	Alexander	595.43
85	Linda	Fisher	520.35
60	Christian	Garcia	597.92
7	Stephanie	Bryan	666.24

11-5. Checking Updatability

ActiveClients

SQL Query

```
INSERT INTO ActiveClients (ClientNo, fName, lName, Email, VehicleRegNo, StartDate, TerminationDate)
VALUES (999, 'John', 'Doe', 'john.doe@email.com', 'V101', '2025-02-01', '2025-02-15');
```

Editor Response

```
102  00:47:08  INSERT INTO ActiveClients (ClientNo, fName, lName, Email, VehicleReg...  Error Code: 1393. Can not modify more than one base table through a join...  0.000 sec
```

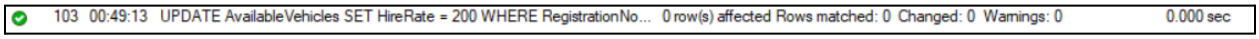
Explanation: The ActiveClients view is not updatable because it is based on a JOIN between multiple tables (Client and HireAgreement). MySQL does not allow inserting or updating data through views that modify more than one base table. Additionally, the view filters data using a WHERE condition, further restricting its updatability. Since an insert into this view would require adding records to both Client and HireAgreement, MySQL blocks the operation.

AvailableVehicles

SQL Query

```
UPDATE AvailableVehicles SET HireRate = 200 WHERE RegistrationNo = 'V101';
```

Editor Response



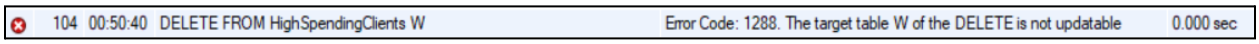
Explanation: The UPDATE AvailableVehicles statement did not cause an error, but it also did not modify any rows because the AvailableVehicles view is not fully updatable. The view filters rows using a WHERE condition (Status = 'Available'), meaning any row that gets updated in a way that changes its availability could disappear from the view. Additionally, the view does not include all columns from the Vehicle table, which MySQL requires for updates to work. Since V101 was not found in the view, MySQL did not apply any changes.

HighSpendingClients

SQL Query

```
DELETE FROM HighSpendingClients WHERE ClientNo = 101;
```

Editor Response



Explanation: The HighSpendingClients view is not updatable because it is based on an aggregate function (SUM(TotalCost)) and a GROUP BY clause. When a view aggregates data from multiple rows into a single row, MySQL cannot determine which base table rows should be deleted. Since DELETE operations require direct access to rows in a table, MySQL prevents deletion from this view.

12. MySQL Restrictions

SQL Query

```
SELECT C.ClientNo, C.fName, C.lName, H.HireNo
FROM Client C
FULL OUTER JOIN HireAgreement H
ON C.ClientNo = H.ClientNo;
```

Editor Response

This ensures:

- We get all clients (even those who have never rented).
- We get all rentals (even if they no longer have an active client record).

Issue in MySQL:

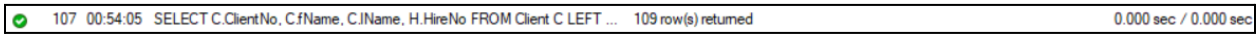
- MySQL does not support FULL OUTER JOIN natively.
- Instead, we need to simulate it using UNION of LEFT JOIN and RIGHT JOIN.

Workaround

SQL Query

```
SELECT C.ClientNo, C.fName, C.lName, H.HireNo
FROM Client C
LEFT JOIN HireAgreement H ON C.ClientNo = H.ClientNo
UNION
SELECT C.ClientNo, C.fName, C.lName, H.HireNo
FROM Client C
RIGHT JOIN HireAgreement H ON C.ClientNo = H.ClientNo;
```

Editor Response



This combines both LEFT JOIN and RIGHT JOIN to simulate a FULL OUTER JOIN to successfully run the query.