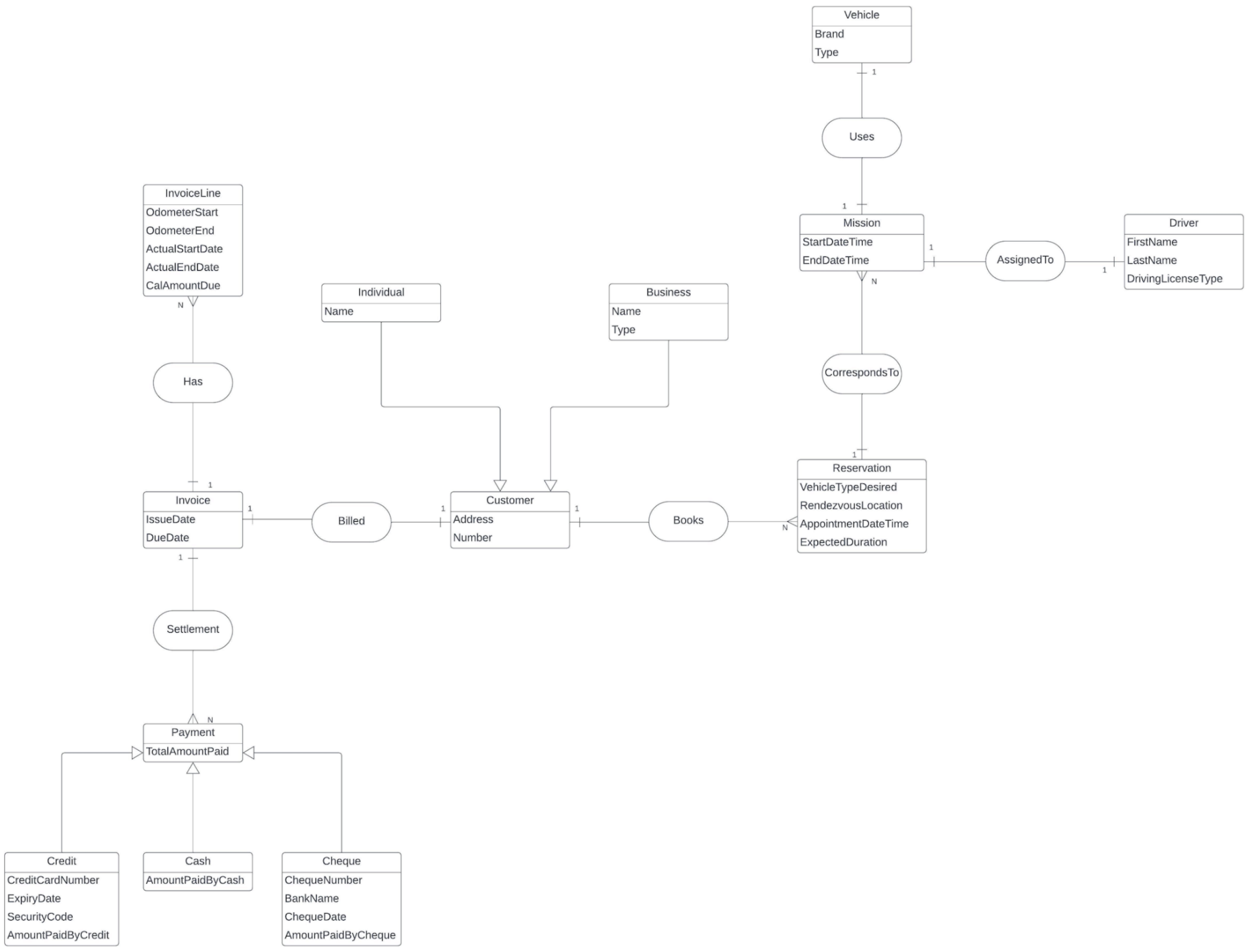
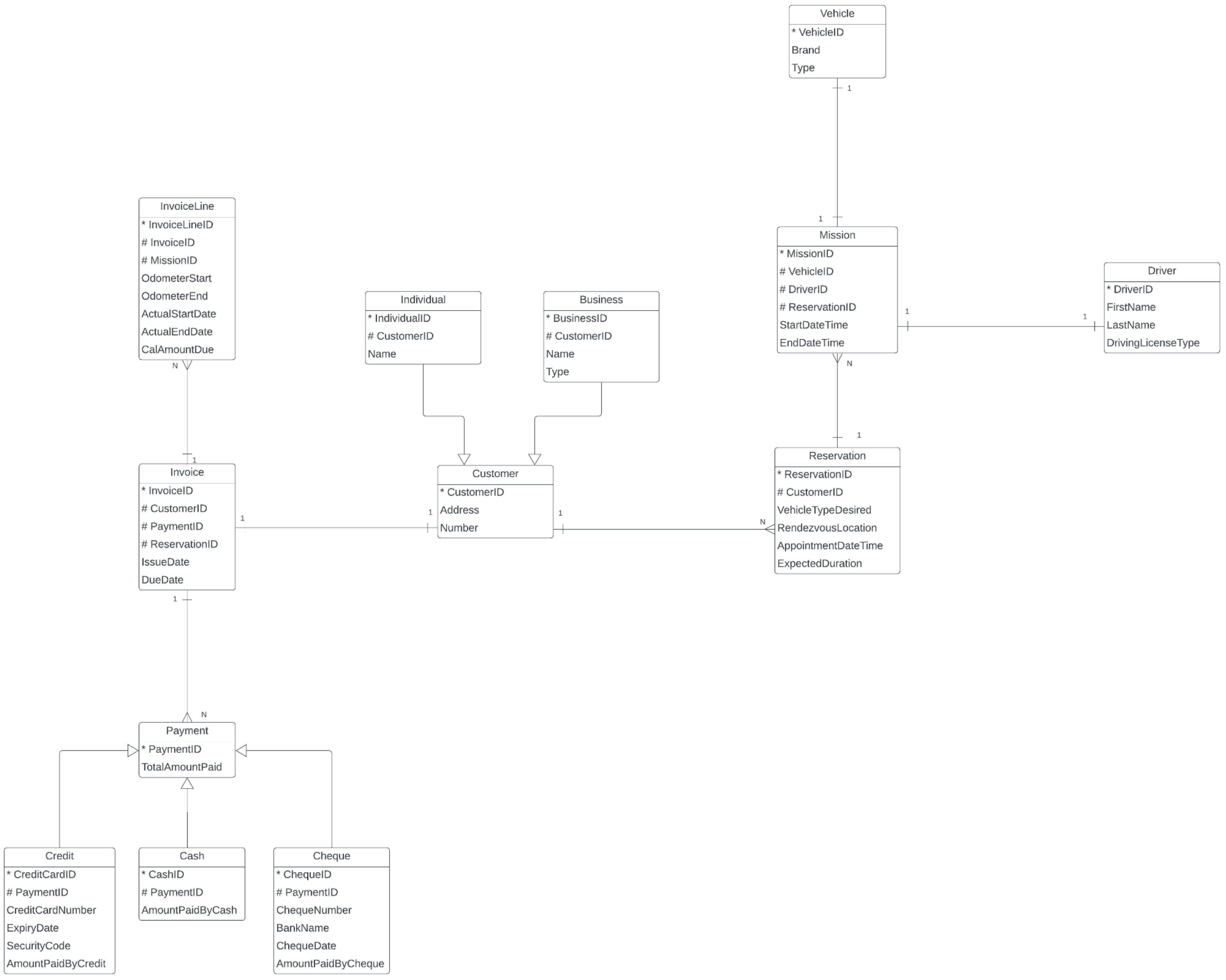
1 Conceptual Diagram



2 Logical Diagram



3 Relational Schema

VEHICLE (VehicleID, Brand, Type)

MISSION(MissionID, VehicleID, DriverID, ReservationID, StartDateTime, EndDateTime)

MISSION [VehicleID] ⊆ VEHICLE [VehicleID]
MISSION [DriverID] ⊆ DRIVER [DriverID]
MISSION [ReservationID] ⊆ RESERVATION [ReservationID]

DRIVER(DriverID, FirstName, LastName, DrivingLicenseType)

RESERVATION(ReservationID, CustomerID, VehicleTypeDesired, RendezvouzLocation, AppointmentDateTime, ExpectedDuration)

RESERVATION [CustomerID] ⊆ CUSTOMER [CustomerID]

INDIVIDUAL(IndividualID, CustomerID, Name)

INDIVIDUAL [CustomerID] ⊆ CUSTOMER [CustomerID]

BUSINESS(BusinessID, CustomerID, Name, Type)

BUSINESS [CustomerID] ⊆ CUSTOMER [CustomerID]

CUSTOMER(CustomerID, Address, Number)

INVOICELINE(InvoiceLineID, InvoiceID, MissionID, OdometerStart, OdometerEnd, ActualStartDate, ActualEndDate, CalAmountDue)

INVOICELINE [InvoiceID] ⊆ INVOICE [InvoiceID]
INVOICELINE [MissionID] ⊆ MISSION [MissionID]

INVOICE(InvoiceID, CustomerID, PaymentID, ReservationID, IssueDate, DueDate)

INVOICE [CustomerID] ⊆ CUSTOMER [CustomerID]
INVOICE [PaymentID] ⊆ PAYMENT [PaymentID]
INVOICE [ReservationID] ⊆ RESERVATION [ReservationID]

PAYMENT(PaymentID, TotalAmountPaid)

CHEQUE(ChequeID, PaymentID, ChequeNumber, BankName, ChequeDate, AmountPaidByCheque)

CHEQUE [PaymentID] ⊆ PAYMENT [PaymentID]

CASH(CashID, PaymentID, AmountPaidByCash)

CASH [PaymentID] $\subseteq$ PAYMENT [PaymentID]

CREDIT(CreditCardID, PaymentID, CreditCardNumber, ExpiryDate, SecurityCode, AmountPaidByCredit)

CREDIT [PaymentID] $\subseteq$ PAYMENT [PaymentID]

4 Normalization

Overview:

The Third Normal Form (3NF) is a specific level of normalization that ensures the following:

1. **Elimination of Redundant Data:**
   - In 3NF, each non-key attribute should provide information about the key, the whole key, and nothing but the key. This means there should be no redundant or duplicated information within a table.
2. **Removal of Columns Not Dependent On Key:**
   - If an attribute depends on only part of a composite key, it should be moved to a separate table. This practice avoids potential inconsistencies and ensures that attributes are logically related to the entire key.
3. **Preservation of Relationships Between Tables:**
   - Foreign keys play a crucial role in maintaining relationships between tables. A foreign key in one table references the primary key in another table, thereby preserving the integrity of the data.

1NF: Ensures simplicity and atomicity of values.

2NF: Eliminates partial dependencies on the primary key.

3NF: Ensures that there are no transitive dependencies among non-prime attributes.

1. VEHICLE:

Attributes: VehicleID (PK), Brand, Type

1NF:

- Where each attribute contains atomic values (no lists or ranges). VehicleID is a unique identifier for each vehicle. The type specifies the category of the vehicle. Brand is the vehicle manufacture name.

2NF:

- In the VEHICLE table, all attributes (VehicleID, Brand, Type) are directly related to the entire primary key (VehicleID). No attribute depends on only a part of the primary key.

3NF:

- In the VEHICLE table, there are no transitive dependencies. All non-prime attributes (Brand, Type) directly depend on the primary key (VehicleID).

2. DRIVER:

Attributes: DriverID (PK), FirstName, LastName, DrivingLicenseType

1NF:

- Each attribute contains atomic values. DriverID uniquely identifies each driver. FirstName represents the driver's first name. Likewise, LastName represents the last name of the driver. DrivingLicenseType is the type of driving license the driver holds

2NF:

- Similar to VEHICLE, the DRIVER table is also in 2NF because all non prime attributes (FirstName, LastName, DrivingLicenseType) depend directly on the entire primary key (DriverID). There are no partial dependencies.

3NF:

- The DRIVER table is in 3NF. Each non-prime attribute (FirstName, LastName, DrivingLicenseType) directly depends on the primary key (DriverID), and there are no transitive dependencies.

3. MISSION:

Attributes: MissionID (PK), VehicleID (FK), DriverID (FK), ReservationID (FK), StartDateTime, EndDateTime

1NF:

- Each attribute contains atomic values. MissionID uniquely identifies each mission. VehicleID, DriverID and ReservationID are foreign keys linking to other tables. StartDateTime and EndDateTime represent the start and end times of the mission.

2NF:

- In the MISSION table, all non key prime attributes (VehicleID, DriverID, ReservationID, StartDateTime, EndDateTime) are directly linked to the entire primary key (MissionID). No attribute depends on only a part of the primary key.

3NF:

- The MISSION table is in 3NF. Every non-prime attribute (VehicleID, DriverID, ReservationID, StartDateTime, EndDateTime) depends directly on the primary key (MissionID). There are no transitive dependencies.

4. CUSTOMER:

Attributes: CustomerID (PK), Address, Number

1NF:

- Each attribute contains atomic values. CustomerID is a unique identifier for each customer. Address is the customer's address. Number is the customer's Number which is an atomic value.

2NF:

- The CUSTOMER table is in 2NF because all non prime key attributes (Address, Number) depend directly on the entire primary key (CustomerID). There are no partial dependencies.

3NF:

- The CUSTOMER table is in 3NF. Each non-prime attribute (Address, Number) depends directly on the primary key (CustomerID), and there are no transitive dependencies.

5. RESERVATION:

Attributes: ReservationID (PK), CustomerID (FK), VehicleTypeDesired, RendezvousLocation, AppointmentDateTime, ExpectedDuration

1NF:

- Each attribute contains atomic values. ReservationID uniquely identifies each reservation. CustomerID is a foreign key. VehicleTypeDesired, RendezvousLocation, AppointmentDateTime, and ExpectedDuration are straightforward details in atomic form.

2NF:

- In the RESERVATION table, all non key prime attributes (CustomerID, VehicleTypeDesired, RendezvousLocation, AppointmentDateTime, ExpectedDuration) depend directly on the entire primary key (ReservationID). No attribute depends on only a part of the primary key.

3NF:

- The RESERVATION table is in the third normal form (3NF). Each non-prime attribute (VehicleTypeDesired, RendezvousLocation, AppointmentDateTime, ExpectedDuration) depends directly on the primary key (ReservationID). There are no transitive dependencies.

6. INVOICE:

Attributes: InvoiceID (PK), CustomerID (FK), PaymentID (FK), ReservationID (FK), IssueDate, DueDate

1NF:

- Each attribute contains atomic values. InvoiceID uniquely identifies each invoice. CustomerID, PaymentID and ReservationID are foreign keys. IssueDate and DueDate are days for the invoice so that the customer can see when it was created and when it's expected to be paid, they are all atomic values.

2NF:

- The INVOICE table is in 2NF because all non prime key attributes (CustomerID, PaymentID, ReservationID, IssueDate, DueDate) depend directly on the entire primary key (InvoiceID). There are no partial dependencies.

3NF:

- The INVOICE table is in 3NF. Each non-prime attribute (CustomerID, PaymentID, ReservationID, IssueDate, DueDate) depends directly on the primary key (InvoiceID), and there are no transitive dependencies.

7. PAYMENT:

Attributes: PaymentID (PK), TotalAmountPaid

1NF:

- Each attribute contains atomic values. PaymentID uniquely identifies each payment. TotalAmountPaid represents the payment amount that the customer has paid.

2NF:

- In the PAYMENT table, all non prime key attributes (TotalAmountPaid) depend directly on the entire primary key (PaymentID). No attribute depends on only a part of the primary key.

3NF:

- The PAYMENT table is in 3NF. Each non-prime attribute (TotalAmountPaid) depends directly on the primary key (PaymentID), and there are no transitive dependencies.

8. CHEQUE, CASH, CREDIT:

Attributes: Vary based on the payment method, but all include PaymentID (FK)

1NF:

- Each attribute contains atomic values. Specific details for each payment method (e.g., ChequeID, CashID, CreditCardID). PaymentID is a foreign key.

2NF:

- The specific tables for payment methods (CHEQUE, CASH, CREDIT) are in 2NF because their respective attributes depend directly on the entire primary key of each table.

3NF:

- The specific tables for payment methods (CHEQUE, CASH, CREDIT) are in 3NF. Each non-prime attribute in these tables depends directly on their respective primary keys (ChequeID, CashID, CreditCardID). There are no transitive dependencies.

9. INVOICELINE:

Attributes: InvoiceLineID (PK), InvoiceID (FK), MissionID (FK), OdometerStart, OdometerEnd, ActualStartDate, ActualEndDate, CalAmountDue

1NF:

- Each attribute contains atomic values.InvoiceLineID uniquely identifies each invoice line. InvoiceID and MissionID are foreign keys linking to other tables. OdometerStart, OdometerEnd, ActualStartDate, ActualEndDate, and CalAmountDue are all atomic attributes repeating groups present.

2NF:

- The attributes OdometerStart, OdometerEnd, ActualStartDate, ActualEndDate, and CalAmountDue depend on the entire primary key (InvoiceLineID). InvoiceID and MissionID are foreign keys.No attribute depends on only a part of the primary key.

3NF:

- The table is in 3NF since there are no transitive dependencies. Each non-prime attribute in these tables depends directly on their respective primary keys

10. INDIVIDUAL, BUSINESS:

Attributes: Both share Name as an attribute, and both have their own primary key where Individual has IndividualID and Business has BusinessID. Business also has a type attribute, and both of the tables have a common CustomerID as its foreign key.

1NF:

- Each attribute contains atomic values.IndividualID and BusinessID uniquely identify each individual and business, respectively. Name is atomic, and CustomerID is a foreign key.No repeating groups are present.

2NF:

- In both the "Individual" and "Business" tables, all attributes depend directly on their respective entire primary keys (IndividualID and BusinessID). No attribute depends on only a part of the primary key.

3NF:

- The tables are in 3NF since there are no transitive dependencies. Each non-prime attribute in these tables depends directly on their respective primary keys

5 Constraints

1. VEHICLE:

- Primary Key: VehicleID
- Attributes:
  - Type (NOT NULL)
  - Brand (NOT NULL)

CHECK that the type is among tourism, heavyweight, super heavyweight

2. MISSION:

- Primary Key: MissionID
- Foreign Key: VehicleID references VEHICLE (VehicleID)
- Foreign Key: DriverID references DRIVER (DriverID)
- Foreign Key: ReservationID references RESERVATION (ReservationID)
- Attributes:
  - StartDateTime (NOT NULL)
  - EndDateTime (NOT NULL)

"Each mission cannot exceed 5 days (Monday morning to Friday evening)"

CHECK to see if the days are no more than 5 for a mission, and if it is between Monday and Friday. For both start and end dates.

"A driver can drive any vehicle if he has the corresponding driving license: tourism, heavyweight, super heavyweight."

CHECK to see if the driver's license type is compatible with the type of the car. Not possible with a check, so will have to use Trigger to ensure before insertion the condition is met.

3. DRIVER:

- Primary Key: DriverID
- Attributes:
  - FirstName (NOT NULL)
  - LastName (NOT NULL)
  - DrivingLicenseType (NOT NULL)

CHECK DrivingLicenseType is of the following tourism, heavyweight, super heavyweight

4. RESERVATION:

- Primary Key: ReservationID
- Foreign Key: CustomerID references CUSTOMER (CustomerID)
- Attributes:
    - VehicleTypeDesired (NOT NULL)
    - RendezvousLocation (NOT NULL)
    - AppointmentDateTime (NOT NULL)
    - ExpectedDuration (NOT NULL)

"A customer can book a truck several weeks in advance for a period of maximum one year"

CHECK to see if the reservation hasn't been booked more than a year

"He may, of course, modify or cancel all or part of a reservation if he does it no later than one week before starting a mission"

Check to see if the reservation exists, and if a user wants to remove a mission or all the missions for the reservation then they can do so a week before the StartDateTime of that mission, and if there's no mission then the reservation is completely cancelled

Check additionally that VehicleTypeDesired is amongst tourism, heavyweight, super heavyweight

Check that the AppointementDateTime entered isn't on the weekend.

5. CUSTOMER:

- Primary Key: CustomerID
- Attributes:
    - Address (NOT NULL)
    - Number (NOT NULL)

6. INDIVIDUAL:

- Primary Key: IndividualID
- Foreign Key: CustomerID references CUSTOMER (CustomerID)
- Attributes:
    - Name (NOT NULL)

7. BUSINESS:

- Primary Key: BusinessID
- Foreign Key: CustomerID references CUSTOMER (CustomerID)
- Attributes:
    - Name (NOT NULL)
    - Type (NOT NULL)

8. INVOICELINE:

- Primary Key: InvoiceLineID
- Foreign Key: InvoiceID references INVOICE (InvoiceID)
- Foreign Key: MissionID references MISSION (MissionID)
- Attributes:
    - OdometerStart (NOT NULL)
    - OdometerEnd (NOT NULL)
    - ActualStartDate (NOT NULL)
    - ActualEndDate (NOT NULL)
    - CalAmountDue (NOT NULL)

9. INVOICE:

- Primary Key: InvoiceID
- Foreign Key: CustomerID references CUSTOMER (CustomerID)
- Foreign Key: PaymentID references PAYMENT (PaymentID)
- Foreign Key: ReservationID references RESERVATION (ReservationID)
- Attributes:
    - IssueDate (NOT NULL)
    - DueDate (NOT NULL)

10. PAYMENT:

- Primary Key: PaymentID
- Attributes:
    - TotalAmountPaid (NOT NULL)

11. CHEQUE:

- Primary Key: ChequeID
- Foreign Key: PaymentID references PAYMENT (PaymentID)
- Attributes:
  - ChequeNumber (NOT NULL)
  - BankName (NOT NULL)
  - ChequeDate (NOT NULL)
  - AmountPaidByCheque (NOT NULL)

12. CASH:

- Primary Key: CashID
- Foreign Key: PaymentID references PAYMENT (PaymentID)
- Attributes:
  - AmountPaidByCash (NOT NULL)

13. CREDIT:

- Primary Key: CreditCardID
- Foreign Key: PaymentID references PAYMENT (PaymentID)
- Attributes:
  - CreditCardNumber (NOT NULL)
  - ExpiryDate (NOT NULL)
  - SecurityCode (NOT NULL)
  - AmountPaidByCredit (NOT NULL)

6 Data Dictionary (SQL Script of the creation of all tables)

Table Name	Column Name	Description	Example Value
VEHICLE	VehicleID	Unique identifier for a vehicle	1
VEHICLE	Type	Type of the vehicle	Heavyweight
VEHICLE	Brand	Brand of the vehicle	Toyota
MISSION	MissionID	Unique identifier for a mission	101
MISSION	VehicleID	Foreign key referencing VEHICLE	1
MISSION	DriverID	Foreign key referencing DRIVER	201
MISSION	ReservationID	Foreign key referencing RESERVATION	301
MISSION	StartDateTime	Start date and time of the mission	2023-11-01 8:00
MISSION	EndDateTime	End date and time of the mission	2023-11-05 17:00
DRIVER	DriverID	Unique identifier for a driver	201
DRIVER	FirstName	First name of the driver	John
DRIVER	LastName	Last name of the driver	Doe
DRIVER	DrivingLicenseType	Type of driving license held by the driver	Heavyweight
RESERVATION	ReservationID	Unique identifier for a reservation	301
RESERVATION	CustomerID	Foreign key referencing CUSTOMER	401
RESERVATION	VehicleTypeDesired	Type of vehicle desired for the reservation	Heavyweight
RESERVATION	RendezvousLocation	Location for rendezvous	Warehouse A
RESERVATION	AppointmentDateTime	Date and time of the reservation	2023-12-01 10:00
RESERVATION	ExpectedDuration	Expected duration of the reservation (in days)	3
CUSTOMER	CustomerID	Unique identifier for a customer	401
CUSTOMER	Address	Address of the customer	123 Main St
INDIVIDUAL	IndividualID	Unique identifier for an individual	501
INDIVIDUAL	CustomerID	Foreign key referencing CUSTOMER	401
INDIVIDUAL	Name	Name of the individual	Jane Doe
BUSINESS	BusinessID	Unique identifier for a business	601
BUSINESS	CustomerID	Foreign key referencing CUSTOMER	401
BUSINESS	Name	Name of the business	XYZ Corp
BUSINESS	Type	Type of the business	Enterprise
INVOICELINE	InvoiceLineID	Unique identifier for an invoice line	701
INVOICELINE	InvoiceID	Foreign key referencing INVOICE	801
INVOICELINE	MissionID	Foreign key referencing MISSION	101
INVOICELINE	OdometerStart	Odometer reading at the start of the mission	10000
INVOICELINE	OdometerEnd	Odometer reading at the end of the mission	10500
INVOICELINE	ActualStartDate	Actual start date and time of the mission	2023-11-05 8:00
INVOICELINE	ActualEndDate	Actual end date and time of the mission	2023-11-06 17:00
INVOICELINE	CalAmountDue	Calculated amount due for the invoice line	150
INVOICE	InvoiceID	Unique identifier for an invoice	801

INVOICE	CustomerID	Foreign key referencing CUSTOMER	401
INVOICE	PaymentID	Foreign key referencing PAYMENT	901
INVOICE	ReservationID	Foreign key referencing RESERVATION	301
INVOICE	IssueDate	Date when the invoice was issued	2023-11-06
INVOICE	DueDate	Due date for the invoice	2023-11-20
PAYMENT	PaymentID	Unique identifier for a payment	901
PAYMENT	TotalAmountPaid	Total amount paid	400
CHEQUE	ChequeID	Unique identifier for a cheque	1001
CHEQUE	PaymentID	Foreign key referencing PAYMENT	901
CHEQUE	ChequeNumber	Cheque number	123456
CHEQUE	BankName	Bank name	ABC Bank
CHEQUE	ChequeDate	Date on the cheque	2023-11-15
CHEQUE	AmountPaidByCheque	Amount paid by cheque	200
CASH	CashID	Unique identifier for a cash payment	1101
CASH	PaymentID	Foreign key referencing PAYMENT	901
CASH	AmountPaidByCash	Amount paid in cash	150
CREDIT	CreditCardID	Unique identifier for a credit card payment	1201
CREDIT	PaymentID	Foreign key referencing PAYMENT	901
CREDIT	CreditCardNumber	Credit card number	1234-4567-8910-1234
CREDIT	ExpiryDate	Expiry date of the credit card	2024-05-31
CREDIT	SecurityCode	Security code of the credit card	789
CREDIT	AmountPaidByCredit	Amount paid by credit card	50

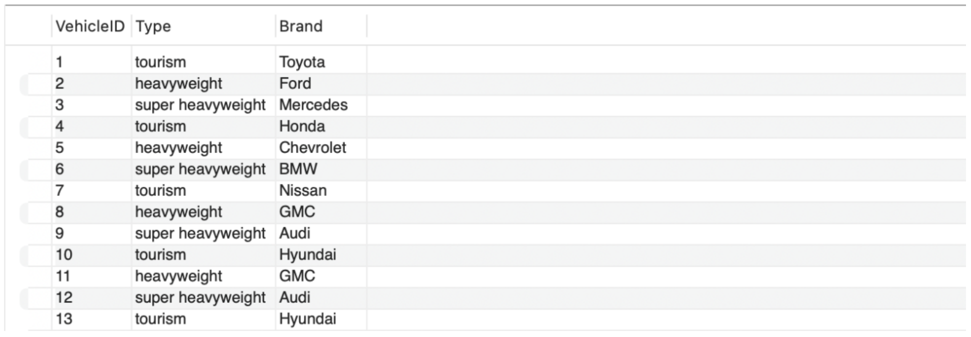
```
CREATE DATABASE Rentrack;

USE Rentrack;


-- 1. VEHICLE

CREATE TABLE VEHICLE (

   VehicleID INT AUTO_INCREMENT PRIMARY KEY,

    Type VARCHAR(255) NOT NULL CHECK (Type IN ('tourism', 'heavyweight', 'super
    heavyweight')),

   Brand VARCHAR(255) NOT NULL

);


-- 2. DRIVER

CREATE TABLE DRIVER (

   DriverID INT AUTO_INCREMENT PRIMARY KEY,

   FirstName VARCHAR(255) NOT NULL,

   LastName VARCHAR(255) NOT NULL,

    DrivingLicenseType VARCHAR(255) NOT NULL CHECK (DrivingLicenseType IN ('tourism',
    'heavyweight', 'super heavyweight'))

);


-- 3. CUSTOMER

CREATE TABLE CUSTOMER (

   CustomerID INT AUTO_INCREMENT PRIMARY KEY,

   Address VARCHAR(255) NOT NULL,

   Number VARCHAR(255) NOT NULL

);


-- 4. INDIVIDUAL

CREATE TABLE INDIVIDUAL (
   IndividualID INT AUTO_INCREMENT PRIMARY KEY,
   CustomerID INT,
   Name VARCHAR(255) NOT NULL,
   FOREIGN KEY (CustomerID) REFERENCES CUSTOMER(CustomerID)
);

-- 5. BUSINESS
CREATE TABLE BUSINESS (
   BusinessID INT AUTO_INCREMENT PRIMARY KEY,
   CustomerID INT,
   Name VARCHAR(255) NOT NULL,
   Type VARCHAR(255) NOT NULL,
   FOREIGN KEY (CustomerID) REFERENCES CUSTOMER(CustomerID)
);

-- 6. RESERVATION
CREATE TABLE RESERVATION (
   ReservationID INT AUTO_INCREMENT PRIMARY KEY,
   CustomerID INT,
    VehicleTypeDesired VARCHAR(255) NOT NULL CHECK (VehicleTypeDesired IN ('tourism',
    'heavyweight', 'super heavyweight')),
     RendezvousLocation VARCHAR(255) NOT NULL,
    AppointmentDateTime DATETIME NOT NULL CHECK
    (DAYOFWEEK(AppointmentDateTime) BETWEEN 2 AND 6),
    ExpectedDuration INT NOT NULL CHECK (ExpectedDuration <= 365),
   FOREIGN KEY (CustomerID) REFERENCES CUSTOMER(CustomerID)
);
-- 7. MISSION
CREATE TABLE MISSION (
   MissionID INT AUTO_INCREMENT PRIMARY KEY,
   VehicleID INT,
   DriverID INT,
   ReservationID INT,
   StartDateTime DATETIME NOT NULL,
   EndDateTime DATETIME NOT NULL,
   FOREIGN KEY (VehicleID) REFERENCES VEHICLE(VehicleID),
   FOREIGN KEY (DriverID) REFERENCES DRIVER(DriverID),
   FOREIGN KEY (ReservationID) REFERENCES RESERVATION(ReservationID),
   CHECK (DATEDIFF(EndDateTime, StartDateTime) <= 5
          AND DAYOFWEEK(StartDateTime) BETWEEN 2 AND 6
          AND DAYOFWEEK(EndDateTime) BETWEEN 2 AND 6)
);

-- 8. PAYMENT
CREATE TABLE PAYMENT (
   PaymentID INT AUTO_INCREMENT PRIMARY KEY,
   TotalAmountPaid INT NOT NULL
);

-- 9. INVOICE
CREATE TABLE INVOICE (
   InvoiceID INT AUTO_INCREMENT PRIMARY KEY,
   CustomerID INT,
   PaymentID INT,
   ReservationID INT,
   IssueDate DATETIME NOT NULL,
   DueDate DATETIME NOT NULL,
   FOREIGN KEY (CustomerID) REFERENCES CUSTOMER(CustomerID),
   FOREIGN KEY (PaymentID) REFERENCES PAYMENT(PaymentID),
   FOREIGN KEY (ReservationID) REFERENCES RESERVATION(ReservationID)
);

-- 10. INVOICELINE
CREATE TABLE INVOICELINE (
   InvoiceLineID INT AUTO_INCREMENT PRIMARY KEY,
   InvoiceID INT,
   MissionID INT,
   OdometerStart INT NOT NULL,
   OdometerEnd INT NOT NULL,
   ActualStartDate DATETIME NOT NULL,
   ActualEndDate DATETIME NOT NULL,
   CalAmountDue INT NOT NULL,
   FOREIGN KEY (InvoiceID) REFERENCES INVOICE(InvoiceID),
   FOREIGN KEY (MissionID) REFERENCES MISSION(MissionID)
);

-- 11. CHEQUE
CREATE TABLE CHEQUE (
   ChequeID INT AUTO_INCREMENT PRIMARY KEY,
   PaymentID INT,
   ChequeNumber INT NOT NULL,
   BankName VARCHAR(255) NOT NULL,
   ChequeDate DATETIME NOT NULL,
   AmountPaidByCheque INT NOT NULL,
   FOREIGN KEY (PaymentID) REFERENCES PAYMENT(PaymentID)
);
```

```
-- 12. CASH
CREATE TABLE CASH (
    CashID INT AUTO_INCREMENT PRIMARY KEY,
    PaymentID INT,
    AmountPaidByCash INT NOT NULL,
    FOREIGN KEY (PaymentID) REFERENCES PAYMENT(PaymentID)
);

-- 13. CREDIT
CREATE TABLE CREDIT (
    CreditCardID INT AUTO_INCREMENT PRIMARY KEY,
    PaymentID INT,
    CreditCardNumber VARCHAR(255) NOT NULL,
    ExpiryDate DATETIME NOT NULL,
    SecurityCode VARCHAR(255) NOT NULL,
    AmountPaidByCredit INT NOT NULL,
    FOREIGN KEY (PaymentID) REFERENCES PAYMENT(PaymentID)
);

DELIMITER //
CREATE TRIGGER check_license_and_vehicle
BEFORE INSERT ON MISSION
FOR EACH ROW
BEGIN
    DECLARE driver_license_type VARCHAR(255);
    DECLARE vehicle_type VARCHAR(255);
    -- Get the driver's license type
      SELECT DrivingLicenseType INTO driver_license_type FROM DRIVER WHERE DriverID =
NEW.DriverID;
```

```
    -- Get the vehicle type

    SELECT Type INTO vehicle_type FROM VEHICLE WHERE VehicleID = NEW.VehicleID;

    -- Check if the driver's license type and vehicle type are compatible

        IF NOT (driver_license_type IN ('tourism', 'heavyweight', 'super heavyweight') AND
driver_license_type = vehicle_type) THEN

        SIGNAL SQLSTATE '45000'

        SET MESSAGE_TEXT = 'Driver and vehicle types are not compatible';

    END IF;

END //

DELIMITER ;
```

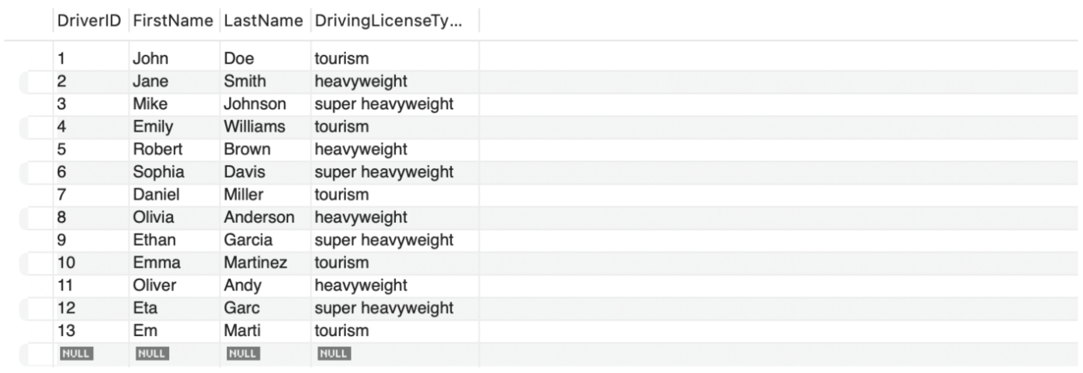
7 Data Insertion in tables

```
-- Populate VEHICLE table

INSERT INTO VEHICLE (Type, Brand) VALUES

('tourism', 'Toyota'),

('heavyweight', 'Ford'),

('super heavyweight', 'Mercedes'),

('tourism', 'Honda'),

('heavyweight', 'Chevrolet'),

('super heavyweight', 'BMW'),

('tourism', 'Nissan'),

('heavyweight', 'GMC'),

('super heavyweight', 'Audi'),

('tourism', 'Hyundai'),

('heavyweight', 'GMC'),

('super heavyweight', 'Audi'),

('tourism', 'Hyundai');
```



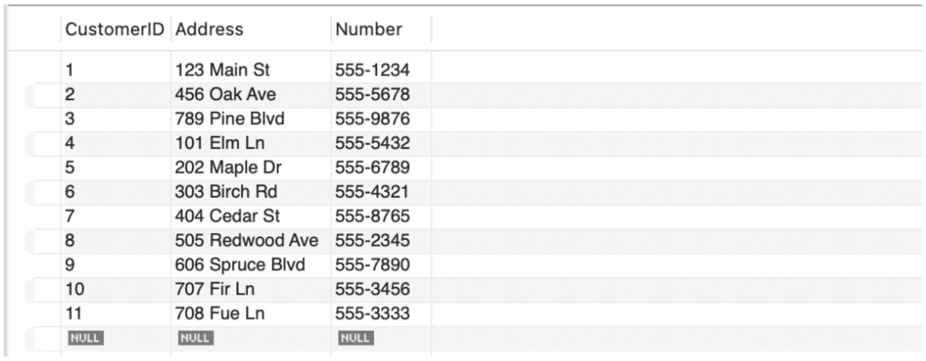
VehicleID	Type	Brand
1	tourism	Toyota
2	heavyweight	Ford
3	super heavyweight	Mercedes
4	tourism	Honda
5	heavyweight	Chevrolet
6	super heavyweight	BMW
7	tourism	Nissan
8	heavyweight	GMC
9	super heavyweight	Audi
10	tourism	Hyundai
11	heavyweight	GMC
12	super heavyweight	Audi
13	tourism	Hyundai

```
-- Populate DRIVER table

INSERT INTO DRIVER (FirstName, LastName, DrivingLicenseType) VALUES

('John', 'Doe', 'tourism'),

('Jane', 'Smith', 'heavyweight'),

('Mike', 'Johnson', 'super heavyweight'),

('Emily', 'Williams', 'tourism'),

('Robert', 'Brown', 'heavyweight'),

('Sophia', 'Davis', 'super heavyweight'),

('Daniel', 'Miller', 'tourism'),

('Olivia', 'Anderson', 'heavyweight'),

('Ethan', 'Garcia', 'super heavyweight'),

('Emma', 'Martinez', 'tourism'),

('Oliver', 'Andy', 'heavyweight'),

('Eta', 'Garc', 'super heavyweight'),

('Em', 'Marti', 'tourism');
```



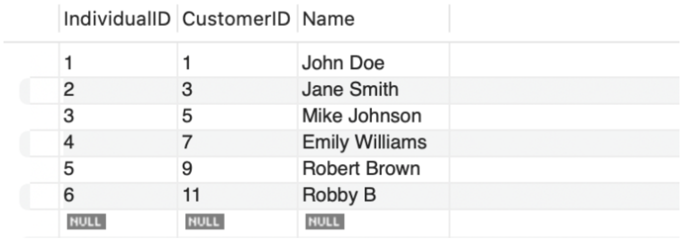
DriverID	FirstName	LastName	DrivingLicenseTy...
1	John	Doe	tourism
2	Jane	Smith	heavyweight
3	Mike	Johnson	super heavyweight
4	Emily	Williams	tourism
5	Robert	Brown	heavyweight
6	Sophia	Davis	super heavyweight
7	Daniel	Miller	tourism
8	Olivia	Anderson	heavyweight
9	Ethan	Garcia	super heavyweight
10	Emma	Martinez	tourism
11	Oliver	Andy	heavyweight
12	Eta	Garc	super heavyweight
13	Em	Marti	tourism
NULL	NULL	NULL	NULL

```
-- Populate CUSTOMER table

INSERT INTO CUSTOMER (Address, Number) VALUES

('123 Main St', '555-1234'),

('456 Oak Ave', '555-5678'),

('789 Pine Blvd', '555-9876'),

('101 Elm Ln', '555-5432'),

('202 Maple Dr', '555-6789'),

('303 Birch Rd', '555-4321'),

('404 Cedar St', '555-8765'),

('505 Redwood Ave', '555-2345'),

('606 Spruce Blvd', '555-7890'),

('707 Fir Ln', '555-3456'),

('708 Fue Ln', '555-3333');
```



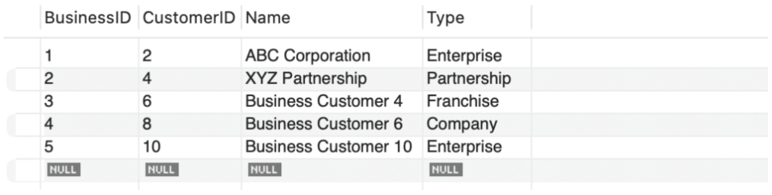
CustomerID	Address	Number
1	123 Main St	555-1234
2	456 Oak Ave	555-5678
3	789 Pine Blvd	555-9876
4	101 Elm Ln	555-5432
5	202 Maple Dr	555-6789
6	303 Birch Rd	555-4321
7	404 Cedar St	555-8765
8	505 Redwood Ave	555-2345
9	606 Spruce Blvd	555-7890
10	707 Fir Ln	555-3456
11	708 Fue Ln	555-3333
NULL	NULL	NULL

```
-- Populate INDIVIDUAL table

INSERT INTO INDIVIDUAL (CustomerID, Name) VALUES

(1, 'John Doe'),

(3, 'Jane Smith'),

(5, 'Mike Johnson'),

(7, 'Emily Williams'),

(9, 'Robert Brown'),

(11, 'Robby B');
```



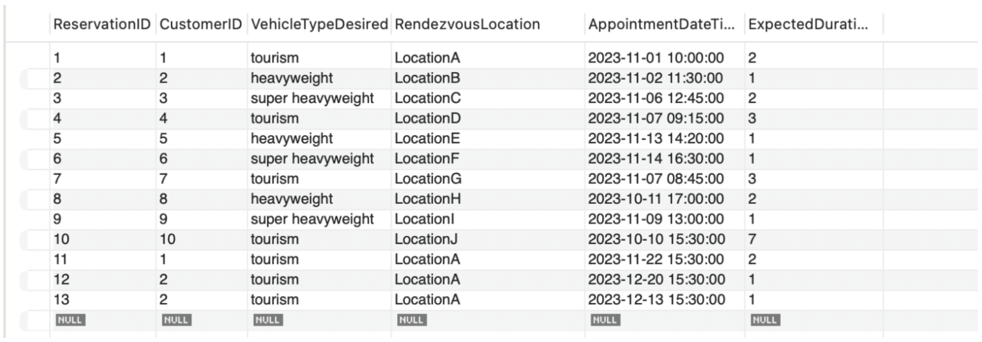
IndividualID	CustomerID	Name
1	1	John Doe
2	3	Jane Smith
3	5	Mike Johnson
4	7	Emily Williams
5	9	Robert Brown
6	11	Robby B
NULL	NULL	NULL

```
-- Populate BUSINESS table

INSERT INTO BUSINESS (CustomerID, Name, Type) VALUES

(2, 'ABC Corporation', 'Enterprise'),

(4, 'XYZ Partnership', 'Partnership'),

(6, 'Business Customer 4', 'Franchise'),

(8, 'Business Customer 6', 'Company'),

(10,'Business Customer 10', 'Enterprise');
```



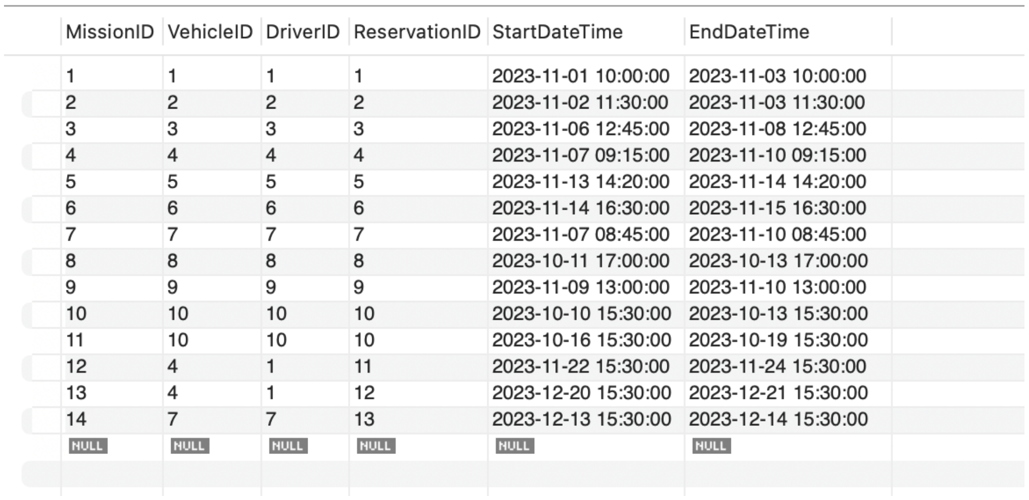
BusinessID	CustomerID	Name	Type
1	2	ABC Corporation	Enterprise
2	4	XYZ Partnership	Partnership
3	6	Business Customer 4	Franchise
4	8	Business Customer 6	Company
5	10	Business Customer 10	Enterprise
NULL	NULL	NULL	NULL

```
-- Populate RESERVATION table

INSERT INTO RESERVATION (CustomerID, VehicleTypeDesired, RendezvousLocation,
AppointmentDateTime, ExpectedDuration) VALUES

(1, 'tourism', 'LocationA', '2023-11-01 10:00:00', 2),

(2, 'heavyweight', 'LocationB', '2023-11-02 11:30:00', 1),

(3, 'super heavyweight', 'LocationC', '2023-11-06 12:45:00', 2),

(4, 'tourism', 'LocationD', '2023-11-07 09:15:00', 3),

(5, 'heavyweight', 'LocationE', '2023-11-13 14:20:00', 1),

(6, 'super heavyweight', 'LocationF', '2023-11-14 16:30:00', 1),

(7, 'tourism', 'LocationG', '2023-11-07 08:45:00', 3),

(8, 'heavyweight', 'LocationH', '2023-10-11 17:00:00', 2),

(9, 'super heavyweight', 'LocationI', '2023-11-09 13:00:00', 1),

(10, 'tourism', 'LocationJ', '2023-10-10 15:30:00', 7),

(1, 'tourism', 'LocationA', '2023-11-22 15:30:00', 2),

(2, 'tourism', 'LocationA', '2023-12-20 15:30:00', 1),

(2, 'tourism', 'LocationA', '2023-12-13 15:30:00', 1);
```



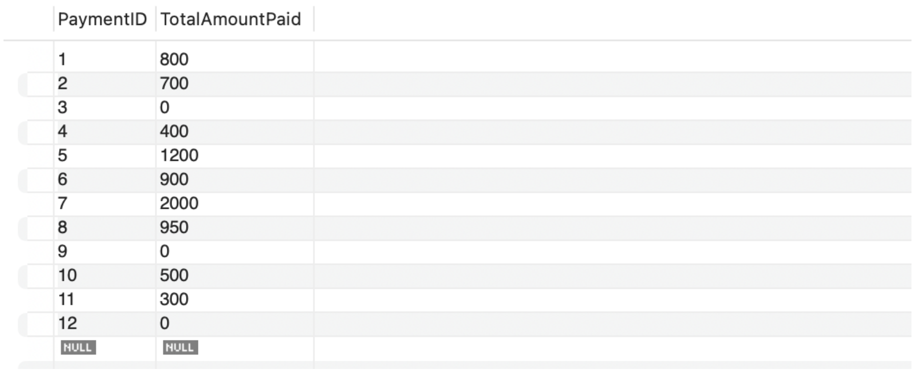
ReservationID	CustomerID	VehicleTypeDesired	RendezvousLocation	AppointmentDateTi...	ExpectedDurati...
1	1	tourism	LocationA	2023-11-01 10:00:00	2
2	2	heavyweight	LocationB	2023-11-02 11:30:00	1
3	3	super heavyweight	LocationC	2023-11-06 12:45:00	2
4	4	tourism	LocationD	2023-11-07 09:15:00	3
5	5	heavyweight	LocationE	2023-11-13 14:20:00	1
6	6	super heavyweight	LocationF	2023-11-14 16:30:00	1
7	7	tourism	LocationG	2023-11-07 08:45:00	3
8	8	heavyweight	LocationH	2023-10-11 17:00:00	2
9	9	super heavyweight	LocationI	2023-11-09 13:00:00	1
10	10	tourism	LocationJ	2023-10-10 15:30:00	7
11	1	tourism	LocationA	2023-11-22 15:30:00	2
12	2	tourism	LocationA	2023-12-20 15:30:00	1
13	2	tourism	LocationA	2023-12-13 15:30:00	1
NULL	NULL	NULL	NULL	NULL	NULL

```
-- Populate MISSION table

INSERT INTO MISSION (VehicleID, DriverID, ReservationID, StartDateTime, EndDateTime) VALUES

(1, 1, 1, '2023-11-01 10:00:00', '2023-11-03 10:00:00'),

(2, 2, 2, '2023-11-02 11:30:00', '2023-11-03 11:30:00'),

(3, 3, 3, '2023-11-06 12:45:00', '2023-11-08 12:45:00'),

(4, 4, 4, '2023-11-07 09:15:00', '2023-11-10 09:15:00'),

(5, 5, 5, '2023-11-13 14:20:00', '2023-11-14 14:20:00'),

(6, 6, 6, '2023-11-14 16:30:00', '2023-11-15 16:30:00'),

(7, 7, 7, '2023-11-07 08:45:00', '2023-11-10 08:45:00'),

(8, 8, 8, '2023-10-11 17:00:00', '2023-10-13 17:00:00'),

(9, 9, 9, '2023-11-09 13:00:00', '2023-11-10 13:00:00'),

(10, 10, 10, '2023-10-10 15:30:00', '2023-10-13 15:30:00'),

(10, 10, 10, '2023-10-16 15:30:00', '2023-10-19 15:30:00'),

(4,1, 11, '2023-11-22 15:30:00', '2023-11-24 15:30:00'),

(4,1, 12, '2023-12-20 15:30:00', '2023-12-21 15:30:00'),

(7,7, 13, '2023-12-13 15:30:00', '2023-12-14 15:30:00');
```



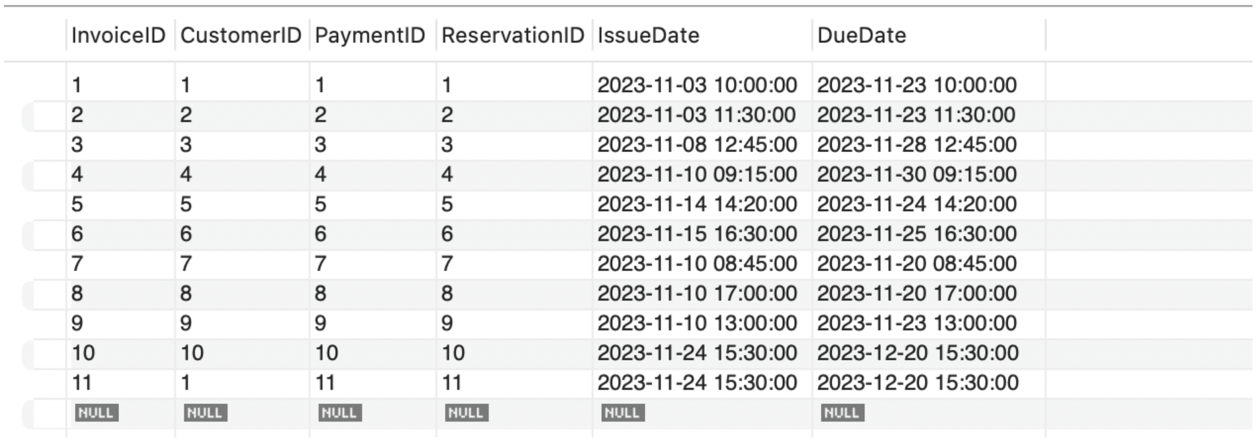
MissionID	VehicleID	DriverID	ReservationID	StartDateTime	EndDateTime
1	1	1	1	2023-11-01 10:00:00	2023-11-03 10:00:00
2	2	2	2	2023-11-02 11:30:00	2023-11-03 11:30:00
3	3	3	3	2023-11-06 12:45:00	2023-11-08 12:45:00
4	4	4	4	2023-11-07 09:15:00	2023-11-10 09:15:00
5	5	5	5	2023-11-13 14:20:00	2023-11-14 14:20:00
6	6	6	6	2023-11-14 16:30:00	2023-11-15 16:30:00
7	7	7	7	2023-11-07 08:45:00	2023-11-10 08:45:00
8	8	8	8	2023-10-11 17:00:00	2023-10-13 17:00:00
9	9	9	9	2023-11-09 13:00:00	2023-11-10 13:00:00
10	10	10	10	2023-10-10 15:30:00	2023-10-13 15:30:00
11	10	10	10	2023-10-16 15:30:00	2023-10-19 15:30:00
12	4	1	11	2023-11-22 15:30:00	2023-11-24 15:30:00
13	4	1	12	2023-12-20 15:30:00	2023-12-21 15:30:00
14	7	7	13	2023-12-13 15:30:00	2023-12-14 15:30:00
NULL	NULL	NULL	NULL	NULL	NULL

```
-- Populate PAYMENT table

INSERT INTO PAYMENT (TotalAmountPaid) VALUES

(800),

(700),

(0),

(400),

(1200),

(900),

(2000),

(950),

(0),

(500),

(300),

(0);
```



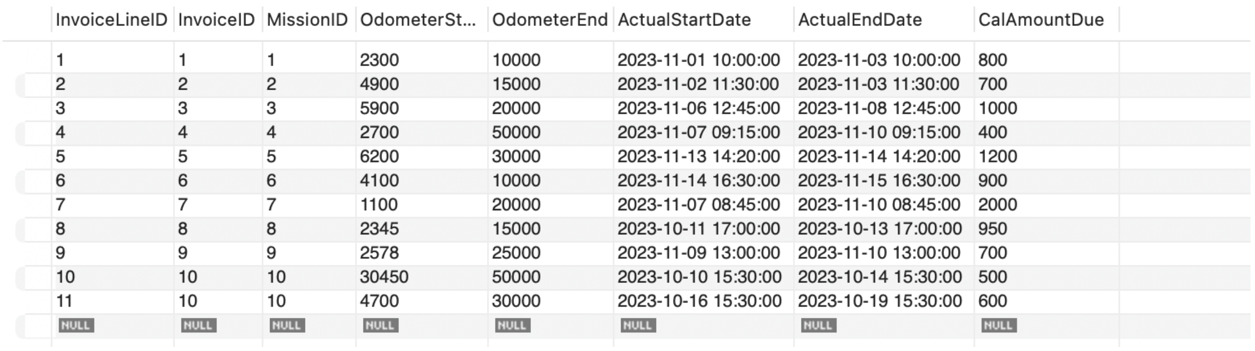
PaymentID	TotalAmountPaid
1	800
2	700
3	0
4	400
5	1200
6	900
7	2000
8	950
9	0
10	500
11	300
12	0
NULL	NULL

```
-- Populate INVOICE table

INSERT INTO INVOICE (CustomerID, PaymentID, ReservationID, IssueDate, DueDate) VALUES

(1, 1, 1, '2023-11-03 10:00:00', '2023-11-23 10:00:00'),

(2, 2, 2, '2023-11-03 11:30:00', '2023-11-23 11:30:00'),

(3, 3, 3, '2023-11-08 12:45:00', '2023-11-28 12:45:00'),

(4, 4, 4, '2023-11-10 09:15:00', '2023-11-30 09:15:00'),

(5, 5, 5, '2023-11-14 14:20:00', '2023-11-24 14:20:00'),

(6, 6, 6, '2023-11-15 16:30:00', '2023-11-25 16:30:00'),

(7, 7, 7, '2023-11-10 08:45:00', '2023-11-20 08:45:00'),

(8, 8, 8, '2023-11-10 17:00:00', '2023-11-20 17:00:00'),

(9, 9, 9, '2023-11-10 13:00:00', '2023-11-23 13:00:00'),

(10, 10, 10,'2023-11-24 15:30:00', '2023-12-20 15:30:00'),

(1, 11, 11,'2023-11-24 15:30:00', '2023-12-20 15:30:00');
```



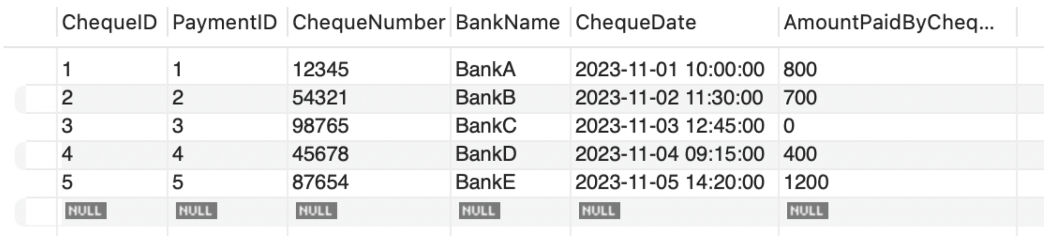
InvoiceID	CustomerID	PaymentID	ReservationID	IssueDate	DueDate
1	1	1	1	2023-11-03 10:00:00	2023-11-23 10:00:00
2	2	2	2	2023-11-03 11:30:00	2023-11-23 11:30:00
3	3	3	3	2023-11-08 12:45:00	2023-11-28 12:45:00
4	4	4	4	2023-11-10 09:15:00	2023-11-30 09:15:00
5	5	5	5	2023-11-14 14:20:00	2023-11-24 14:20:00
6	6	6	6	2023-11-15 16:30:00	2023-11-25 16:30:00
7	7	7	7	2023-11-10 08:45:00	2023-11-20 08:45:00
8	8	8	8	2023-11-10 17:00:00	2023-11-20 17:00:00
9	9	9	9	2023-11-10 13:00:00	2023-11-23 13:00:00
10	10	10	10	2023-11-24 15:30:00	2023-12-20 15:30:00
11	1	11	11	2023-11-24 15:30:00	2023-12-20 15:30:00
NULL	NULL	NULL	NULL	NULL	NULL

```
-- Populate INVOICELINE table

INSERT INTO INVOICELINE (InvoiceID, MissionID, OdometerStart, OdometerEnd, ActualStartDate,
ActualEndDate, CalAmountDue) VALUES

(1, 1, 2300, 10000, '2023-11-01 10:00:00', '2023-11-03 10:00:00', 800),

(2, 2, 4900, 15000, '2023-11-02 11:30:00', '2023-11-03 11:30:00', 700),

(3, 3, 5900, 20000, '2023-11-06 12:45:00', '2023-11-08 12:45:00', 1000),

(4, 4, 2700, 50000, '2023-11-07 09:15:00', '2023-11-10 09:15:00', 400),

(5, 5, 6200, 30000, '2023-11-13 14:20:00', '2023-11-14 14:20:00', 1200),

(6, 6, 4100, 10000, '2023-11-14 16:30:00', '2023-11-15 16:30:00', 900),

(7, 7, 1100, 20000, '2023-11-07 08:45:00', '2023-11-10 08:45:00', 2000),

(8, 8, 2345, 15000, '2023-10-11 17:00:00', '2023-10-13 17:00:00', 950),

(9, 9, 2578, 25000, '2023-11-09 13:00:00', '2023-11-10 13:00:00', 700),

(10, 10, 30450, 50000, '2023-10-10 15:30:00', '2023-10-14 15:30:00', 500),

(10, 10, 4700, 30000, '2023-10-16 15:30:00', '2023-10-19 15:30:00', 600);
```



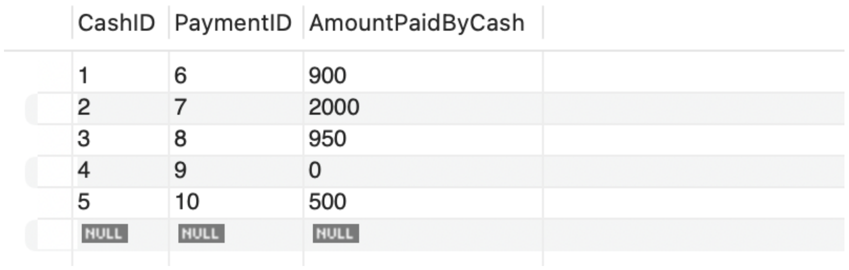
InvoiceLineID	InvoiceID	MissionID	OdometerSt...	OdometerEnd	ActualStartDate	ActualEndDate	CalAmountDue
1	1	1	2300	10000	2023-11-01 10:00:00	2023-11-03 10:00:00	800
2	2	2	4900	15000	2023-11-02 11:30:00	2023-11-03 11:30:00	700
3	3	3	5900	20000	2023-11-06 12:45:00	2023-11-08 12:45:00	1000
4	4	4	2700	50000	2023-11-07 09:15:00	2023-11-10 09:15:00	400
5	5	5	6200	30000	2023-11-13 14:20:00	2023-11-14 14:20:00	1200
6	6	6	4100	10000	2023-11-14 16:30:00	2023-11-15 16:30:00	900
7	7	7	1100	20000	2023-11-07 08:45:00	2023-11-10 08:45:00	2000
8	8	8	2345	15000	2023-10-11 17:00:00	2023-10-13 17:00:00	950
9	9	9	2578	25000	2023-11-09 13:00:00	2023-11-10 13:00:00	700
10	10	10	30450	50000	2023-10-10 15:30:00	2023-10-14 15:30:00	500
11	10	10	4700	30000	2023-10-16 15:30:00	2023-10-19 15:30:00	600
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
-- Populate CHEQUE table

INSERT INTO CHEQUE (PaymentID, ChequeNumber, BankName, ChequeDate,
AmountPaidByCheque) VALUES

(1, 12345, 'BankA', '2023-11-01 10:00:00', 800),

(2, 54321, 'BankB', '2023-11-02 11:30:00', 700),

(3, 98765, 'BankC', '2023-11-03 12:45:00', 0),

(4, 45678, 'BankD', '2023-11-04 09:15:00', 400),

(5, 87654, 'BankE', '2023-11-05 14:20:00', 1200);
```



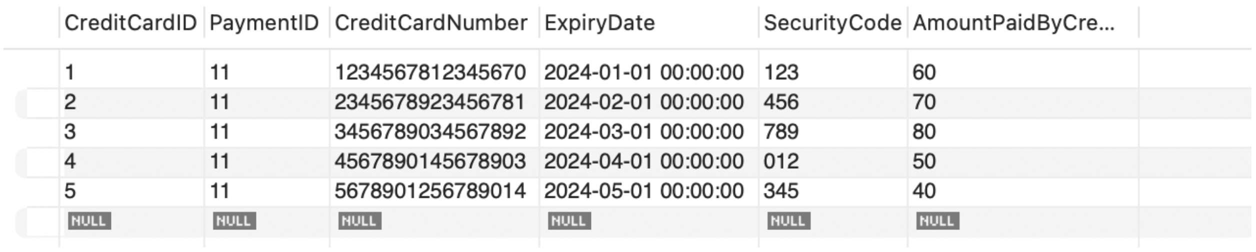
ChequeID	PaymentID	ChequeNumber	BankName	ChequeDate	AmountPaidByCheq...
1	1	12345	BankA	2023-11-01 10:00:00	800
2	2	54321	BankB	2023-11-02 11:30:00	700
3	3	98765	BankC	2023-11-03 12:45:00	0
4	4	45678	BankD	2023-11-04 09:15:00	400
5	5	87654	BankE	2023-11-05 14:20:00	1200
NULL	NULL	NULL	NULL	NULL	NULL

```
-- Populate CASH table

INSERT INTO CASH (PaymentID, AmountPaidByCash) VALUES

(6, 900),

(7, 2000),

(8, 950),

(9, 0),

(10, 500);
```



CashID	PaymentID	AmountPaidByCash
1	6	900
2	7	2000
3	8	950
4	9	0
5	10	500
NULL	NULL	NULL

```
-- Populate CREDIT table

INSERT INTO CREDIT (PaymentID, CreditCardNumber, ExpiryDate, SecurityCode,
AmountPaidByCredit) VALUES

(11, '1234567812345670', '2024-01-01 00:00:00', '123', 60),

(11, '2345678923456781', '2024-02-01 00:00:00', '456', 70),

(11, '3456789034567892', '2024-03-01 00:00:00', '789', 80),

(11, '4567890145678903', '2024-04-01 00:00:00', '012', 50),

(11, '5678901256789014', '2024-05-01 00:00:00', '345', 40);
```



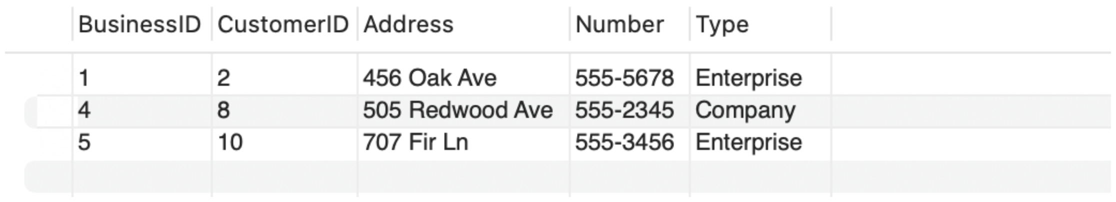
CreditCardID	PaymentID	CreditCardNumber	ExpiryDate	SecurityCode	AmountPaidByCre...
1	11	1234567812345670	2024-01-01 00:00:00	123	60
2	11	2345678923456781	2024-02-01 00:00:00	456	70
3	11	3456789034567892	2024-03-01 00:00:00	789	80
4	11	4567890145678903	2024-04-01 00:00:00	012	50
5	11	5678901256789014	2024-05-01 00:00:00	345	40
NULL	NULL	NULL	NULL	NULL	NULL

8 Implementation of queries with outputs

Query #1: List of customers that are businesses (Enterprises or Companies)

```
SELECT DISTINCT b.BusinessID, c.CustomerID, c.Address, c.Number, b.Type

FROM CUSTOMER c

JOIN BUSINESS b ON c.CustomerID = b.CustomerID

WHERE b.Type IN ('Company', 'Enterprise');
```

Output #1:

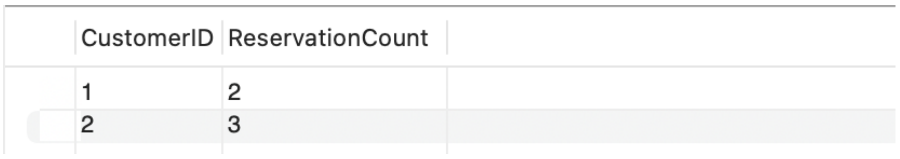


BusinessID	CustomerID	Address	Number	Type
1	2	456 Oak Ave	555-5678	Enterprise
4	8	505 Redwood Ave	555-2345	Company
5	10	707 Fir Ln	555-3456	Enterprise

Query #2: List of reservations whose reservation number is greater than 1.

```
SELECT CustomerID, COUNT(*) AS ReservationCount

FROM RESERVATION

GROUP BY CustomerID

HAVING ReservationCount > 1;
```

Output #2:

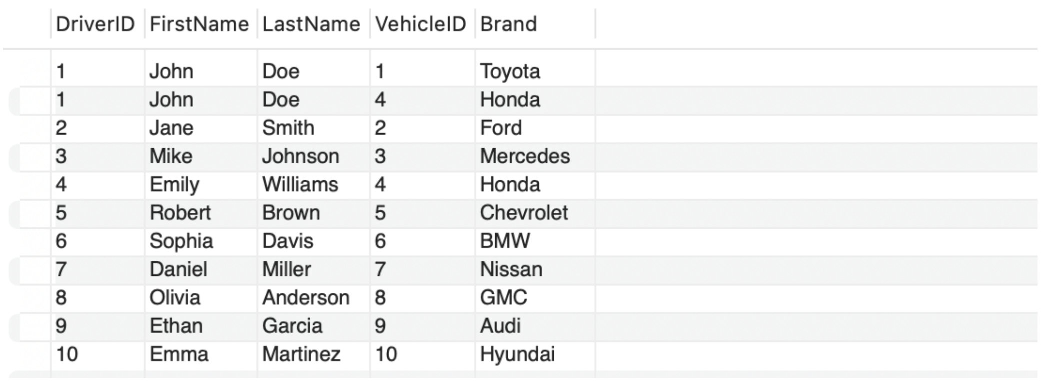


CustomerID	ReservationCount
1	2
2	3

Query #3: List of drivers and vehicles having participated in at least one mission.

```
SELECT DISTINCT d.DriverID, d.FirstName, d.LastName, v.VehicleID, v.Brand

FROM DRIVER d

JOIN MISSION m ON d.DriverID = m.DriverID

JOIN VEHICLE v ON m.VehicleID = v.VehicleID;
```

Output #3:

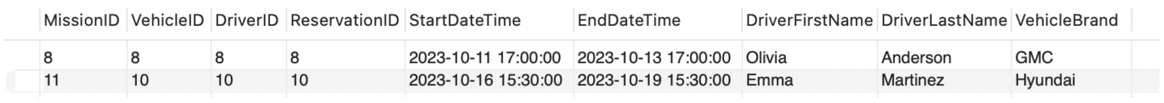


DriverID	FirstName	LastName	VehicleID	Brand
1	John	Doe	1	Toyota
1	John	Doe	4	Honda
2	Jane	Smith	2	Ford
3	Mike	Johnson	3	Mercedes
4	Emily	Williams	4	Honda
5	Robert	Brown	5	Chevrolet
6	Sophia	Davis	6	BMW
7	Daniel	Miller	7	Nissan
8	Olivia	Anderson	8	GMC
9	Ethan	Garcia	9	Audi
10	Emma	Martinez	10	Hyundai

Query #4: List of missions between October 11, 2023 and October 18, 2023 as well as the drivers and vehicles participating in these missions.

```
SELECT m.*, d.FirstName AS DriverFirstName, d.LastName AS DriverLastName, v.Brand AS
VehicleBrand

FROM MISSION m

JOIN DRIVER d ON m.DriverID = d.DriverID

JOIN VEHICLE v ON m.VehicleID = v.VehicleID

WHERE m.StartDateTime BETWEEN '2023-10-11' AND '2023-10-18';
```

Output #4:

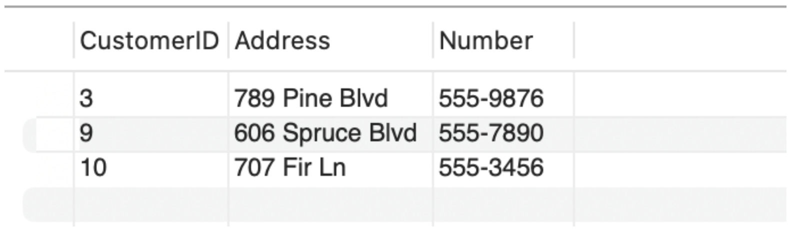


MissionID	VehicleID	DriverID	ReservationID	StartDateTime	EndDateTime	DriverFirstName	DriverLastName	VehicleBrand
8	8	8	8	2023-10-11 17:00:00	2023-10-13 17:00:00	Olivia	Anderson	GMC
11	10	10	10	2023-10-16 15:30:00	2023-10-19 15:30:00	Emma	Martinez	Hyundai

Query #5: The list of customers who have not paid their invoices.

```
SELECT c.CustomerID, c.Address, c.Number

FROM CUSTOMER c

LEFT JOIN INVOICE i ON c.CustomerID = i.CustomerID

LEFT JOIN (

  SELECT InvoiceID, SUM(CalAmountDue) AS CombinedAmountDue

  FROM INVOICELINE

  GROUP BY InvoiceID

) il_sum ON i.InvoiceID = il_sum.InvoiceID

LEFT JOIN PAYMENT p ON i.PaymentID = p.PaymentID

WHERE p.TotalAmountPaid < il_sum.CombinedAmountDue;
```

Output #5:

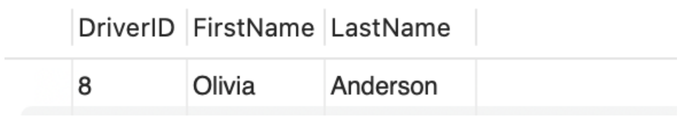


CustomerID	Address	Number
3	789 Pine Blvd	555-9876
9	606 Spruce Blvd	555-7890
10	707 Fir Ln	555-3456

Query #6: List of drivers who have driven 'GMC' brand vehicles.

```
SELECT DISTINCT d.DriverID, d.FirstName, d.LastName

FROM DRIVER d

JOIN MISSION m ON d.DriverID = m.DriverID

JOIN VEHICLE v ON m.VehicleID = v.VehicleID

WHERE v.Brand = 'GMC';
```

Output #6:

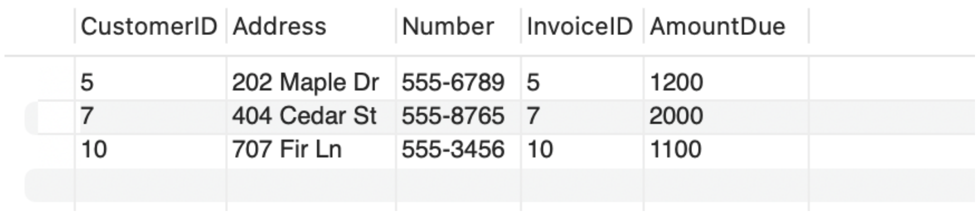


DriverID	FirstName	LastName
8	Olivia	Anderson

Query #7: Which customers have invoices greater than 1000 $?

```
SELECT c.CustomerID, c.Address, c.Number,
    i.InvoiceID,
    SUM(il.CalAmountDue) AS AmountDue
FROM CUSTOMER c
JOIN INVOICE i ON c.CustomerID = i.CustomerID
LEFT JOIN INVOICELINE il ON i.InvoiceID = il.InvoiceID
GROUP BY c.CustomerID, i.InvoiceID
HAVING AmountDue > 1000;
```

Output #7:

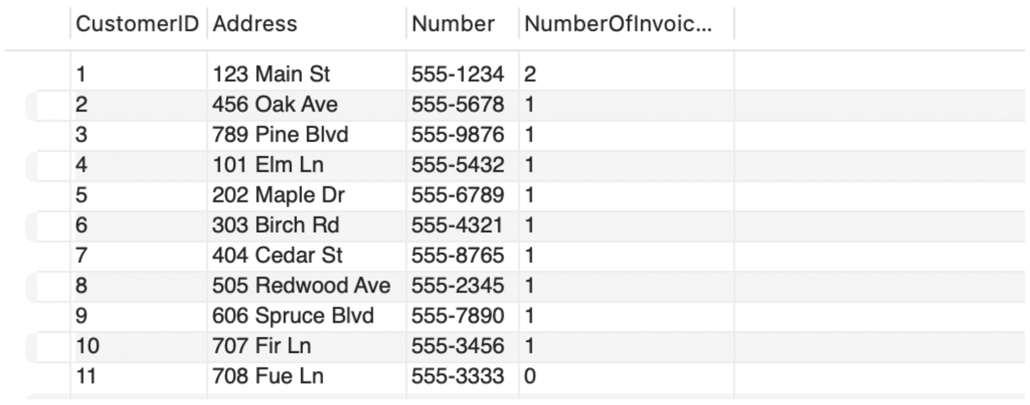


CustomerID	Address	Number	InvoiceID	AmountDue
5	202 Maple Dr	555-6789	5	1200
7	404 Cedar St	555-8765	7	2000
10	707 Fir Ln	555-3456	10	1100

Query #8: List of customers with their number of associated invoices.

```
SELECT c.CustomerID, c.Address, c.Number, COUNT(i.InvoiceID) AS NumberOfInvoices
FROM CUSTOMER c
LEFT JOIN INVOICE i ON c.CustomerID = i.CustomerID
GROUP BY c.CustomerID, c.Address, c.Number;
```

Output #8:

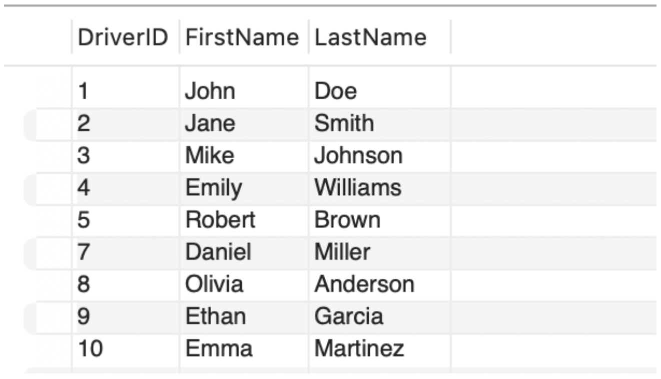


CustomerID	Address	Number	NumberOfInvoic...
1	123 Main St	555-1234	2
2	456 Oak Ave	555-5678	1
3	789 Pine Blvd	555-9876	1
4	101 Elm Ln	555-5432	1
5	202 Maple Dr	555-6789	1
6	303 Birch Rd	555-4321	1
7	404 Cedar St	555-8765	1
8	505 Redwood Ave	555-2345	1
9	606 Spruce Blvd	555-7890	1
10	707 Fir Ln	555-3456	1
11	708 Fue Ln	555-3333	0

Query #9: What are the last names and first names of the drivers who have a mission between the following dates: October 1, 2023 and November 30, 2023 whose mileage (number of kilometers traveled) is more than 7000 km?

```
SELECT DISTINCT d.DriverID, d.FirstName, d.LastName

FROM DRIVER d

JOIN MISSION m ON d.DriverID = m.DriverID

JOIN INVOICELINE il ON m.MissionID = il.MissionID

WHERE m.StartDateTime BETWEEN '2023-10-01' AND '2023-11-30'

AND (il.OdometerEnd - il.OdometerStart) > 7000;
```

Output #9:



DriverID	FirstName	LastName
1	John	Doe
2	Jane	Smith
3	Mike	Johnson
4	Emily	Williams
5	Robert	Brown
7	Daniel	Miller
8	Olivia	Anderson
9	Ethan	Garcia
10	Emma	Martinez