Step 6.1 Begin with the list of the tables that the entities and relationships from the
ER diagram mapped to naturally, from the sample project section at the end of chapter 4. For each table on the list, identify functional dependencies and normalize
the relation to BCNF. Then decide whether the resulting tables should be implemented in that form. If not, explain why.

Office(Office_ID, Location, Department, No.of.Employee)

OfficeID → Location, Department, No. of employee

Since, OfficeID as a primary key determine all the other attributes, it is in BCNF form

Employee(EmployeeID, Name, DOB, Address(apartmentNo., city, country), Position, officeID)

Functional dependency:

employeeID → Name, DOB, Address, Position, OfficeID
officeID → Location, Department, No. of employee

To normalize we can split the employee table into two:

Table ‘ employee ‘

- EmployeeID → Name, DOB, Address, Position
- Primary key (EmployeeID)

Table ‘employee_office’

- employeeID → officeID
- officeID → Location, Department, No. of employee

Primary key (employeeID, officeID)

Student(Student ID, FirstName, LastName, DOB, ContactNo, Email, Sex, Address(apartmentNo., city, country), officeID)

Functional dependency:

studentID → FirstName, LastName, DOB, ContactNo, Email, Sex, Address(apartmentNo., city, country), officeID

officeID → Location, Department, No. of employee

the table isn't in BCNF form as there is a transitive dependency of
studentID → FirstName, LastName, DOB, ContactNo, Email, Sex, Address,OfficeID →
office_location, Department, no. of employee

To normalize it we can split the table:

Table 'student'
- studentID → FirstName, LastName, DOB, ContactNo, Email, Sex, Address, officeID)
- Primary key(studentID)

Table 'student_office'

- studentID → officeID
- officeID → office_location, Department, no. of employee
- Primary key(studentID, officeID)

Instructor(InstructorID, EmployeeID, Course_ID, FirstName, LastName, Address, VehicleID, Sex, Availability)

Functional dependency:

instructorID → EmployeeID, FirstName, LastName, Address, VehicleID, Sex, Availability

EmployeeID → Name, DOB, Address, Position, OfficeID

vehicleID → Vehicle Make, Vehicle Type, Model,Year ,Plate Number, officeID

The table is not in BCNF because there is a transitive dependency of instructorID →
employeeID → Name, DOB, Address, Position, OfficeID → courseID → CourseName,
InstructorID,studentID, Start Date, End Date, Duration, Level → vehicleID → Vehicle Make,
Vehicle Type, Model,Year ,Plate Number, officeID

To normalize it we can split the table into four:

Table 'instructor'
- instructorID → , FirstName, LastName, Address, Sex, Availability
- Primary key(instructorID)

Table ‘instructor_as_employee’

- instructorID → employeeID
- employeeID → Name, DOB, Address, Position, OfficeID
- Primary key(instructorID, employeeID)

Table ‘instructor_vehicle’

- instructorID → vehicleID
- vehicleID → Vehicle Make, Vehicle Type, Model,Year ,Plate Number, officeID
- Primary key (instructorID, vehicelID)

Course (Course ID, CourseName, InstructorID,studentID, Start Date, End Date, Duration, Level)

Functional dependency:
courseID → CourseName, InstructorID,studentID, Start Date, End Date, Duration, Level
instructorID → EmployeeID, Course_ID, FirstName, LastName, Address, VehicleID, Sex, Availability
studentID → FirstName, LastName, DOB, ContactNo, Email, Sex, Address, officeID

The table course is not in the BCNF because there is a transitional dependency of courseID → CourseName, InstructorID,studentID, Start Date, End Date, Duration, Level → instructorID → EmployeeID, Course_ID, FirstName, LastName, Address, VehicleID, Sex, Availability→studentID → FirstName, LastName, DOB, ContactNo, Email, Sex, Address, officeID

To normalize the the table we will split the table into three:

Table ‘course’

- courseID → CourseName, Start Date, End Date, Duration, Level
- Primary key (courseID)

Tabel ‘course_instructor’

- courseID → instructorID
- instructorID → EmployeeID, Course_ID, FirstName, LastName, Address, VehicleID, Sex, Availability
- Primary key (instructorID, courseID)

Table ‘ course_student’

- courseID → studentID
- studentID → FirstName, LastName, DOB, ContactNo, Email, Sex, Address, officeID
- Primary key (courseID. studentID)

Vehicle(VehicleID, Vehicle Make, Vehicle Type, Model,Year ,Plate Number, officeID)

Functional dependency:

vehicleID → Vehicle Make, Vehicle Type, Model,Year ,Plate Number, officeID
officeID → Location, Department, No. of employee

Since the above table has a transitive dependency, the table is not in BCNF. And, the transitive dependency are vehicleID → Vehicle Make, Vehicle Type, Model,Year ,Plate Number → officeID → Location, Department, No. of employee

To normalize the table we will split the table into two:

Table ‘vehicle’
- vehicleID → Vehicle Make, Vehicle Type, Model,Year ,Plate Number
- Primary key(vehicleID)

Table ‘ office_vehicle’
- vehicelID → courseID
- courseID → CourseName, InstructorID,studentID, Start Date, End Date, Duration, Level
- Primary key(vehicelID, courseID)

Maintenance(VehicleID, Vehicle Type , Date, Service Type, Cost, Next Schedule Maintenance)

Functional dependency:
vehicleID → vehicle Type, Date, Service Type, Cost, Next Schedule Maintenance

The table is already in BCNF because the primary key vehicleID is determining all the other attributes in the maintenance table.

Payment(PaymentID, Payment_Date, Nameofstudent, Amount, Email, studentID)

Functional dependency:

paymentID → Payment_Date, Nameofstudent, Amount, Email, studentID

studentID → FirstName, LastName, DOB, ContactNo, Email, Sex, Address, officeID

The table is not in BCNF because of the transitive dependency paymentID → Payment_Date, Nameofstudent, Amount, Email, studentID → FirstName, LastName, DOB, ContactNo, Email, Sex, Address, officeID

To normalize the table we split the table into two:

Table ‘payment’

- paymentID → Payment_Date, Nameofstudent, Amount, Email
- Primary key (PaymentID)

Table ‘student_payment’

- paymentID → studentID
- studentID → FirstName, LastName, DOB, ContactNo, Email, Sex, Address, officeID
- Primary key(paymentID, studentID)

6.2 Update the data dictionary and list of assumptions as needed

Office Table

- Office_ID (INT): The unique identifier for an office.
- Location (VARCHAR(45)): The location of the office.
- Department (VARCHAR(45)): The department located in the office.
- No_of_Employee (INT): The number of employees in the office.

Assumptions:

- Each `Office_ID` is unique and cannot be NULL.
- The combination of `Location` and `Department` should be unique.
- The `No_of_Employee` can be NULL, indicating that the number of employees in the office is unknown or not provided.

Employee Table

- EmployeeID (INT): The unique identifier for an employee.
- Name (VARCHAR(45)): The name of the employee.
- DOB (DATE): The date of birth of the employee.
- Address : The address of the employee, which includes:
  - apartmentNo (VARCHAR(10)): The apartment number.
  - city (VARCHAR(45)): The city where the employee lives.
  - country (VARCHAR(45)): The country where the employee lives.
- Position (VARCHAR(45)): The position of the employee in the company.
- OfficeID (INT): The identifier of the office where the employee works.

Assumptions:

- Each `EmployeeID` is unique and cannot be NULL.
- The combination of employeeID and officeID should be unique
- The `Name`, `DOB`, `Address`, `Position`, and ` can be NULL, indicating that the information is unknown or not provided.

Student Table

- StudentID (INT): The unique identifier for a student.
- FirstName (VARCHAR(45)): The first name of the student.
- LastName (VARCHAR(45)): The last name of the student.
- DOB (DATE): The date of birth of the student.
- ContactNo (VARCHAR(15)): The contact number of the student.
- Email (VARCHAR(45)): The email of the student.
- Sex (CHAR(1)): The gender of the student.
- Address: The address of the student, which includes:
  - apartmentNo (VARCHAR(10)): The apartment number.
  - city (VARCHAR(45)): The city where the student lives.
  - country (VARCHAR(45)): The country where the student lives.
- OfficeID (INT): The identifier of the office where the student is enrolled.

Assumption:

-Each StudentID is unique and cannot be NULL.
- The combination of officeID and studnetID should be unique
-The combination of FirstName, LastName, and DOB should be unique.
-The ContactNo, Email, Sex, Address, and can be NULL, indicating that the information is unknown or not provided.

Instructor Table

- InstructorID (INT): The unique identifier for an instructor.
- EmployeeID (INT): The employee identifier for an instructor.
- FirstName (VARCHAR(45)): The first name of the instructor.
- LastName (VARCHAR(45)): The last name of the instructor.
- Address : The address of the instructor.
  - apartmentNo (VARCHAR(10)): The apartment number.
  - city (VARCHAR(45)): The city where the instructor lives.
  - country (VARCHAR(45)): The country where the instructor lives.
- VehicleID (INT): The identifier of the vehicle that the instructor uses.
- Sex (CHAR(1)): The gender of the instructor.
- Availability (VARCHAR(45)): The availability of the instructor.

Assumption:
-Each InstructorID is unique and cannot be NULL.
- The combination of instructorID and employeeID should be unique
- The combination of instructorID and vehicleID should be unique
- The FirstName, LastName, Address, Sex, and Availability can be NULL, indicating that the information is unknown or not provided.

Course Table

- CourseID (INT): The unique identifier for a course.
- CourseName (VARCHAR(45)): The name of the course.

- InstructorID (INT): The identifier of the instructor who teaches the course.
- StudentID (INT): The identifier of the student who is enrolled in the course.
- StartDate (DATE): The start date of the course.
- EndDate (DATE): The end date of the course.
- Duration (INT): The duration of the course in hours.
- Level (VARCHAR(45)): The level of the course.

Assumption:

- Each CourseID is unique and cannot be NULL.
- The combination of courseID and studnetID should be unique
- The combination of courseID and instructorID should be unique
- The CourseName,, StartDate, EndDate, Duration, and Level can be NULL, indicating that the information is unknown or not provided.

Vehicle Table

- VehicleID (INT): The unique identifier for a vehicle.
- VehicleMake (VARCHAR(45)): The make of the vehicle.
- VehicleType (VARCHAR(45)): The type of the vehicle.
- Model (VARCHAR(45)): The model of the vehicle.
- Year (YEAR): The year of the vehicle.
- PlateNumber (VARCHAR(10)): The plate number of the vehicle.
- OfficeID (INT): The identifier of the office where the vehicle is located.

Assumption:

- Each VehicleID is unique and cannot be NULL.
- The combination of vehicleID and officeID should be unique
- The VehicleMake, VehicleType, Model, Year, PlateNumber, and can be NULL, indicating that the information is unknown or not provided.

Maintenance Table

- VehicleID (INT): The unique identifier for a vehicle.
- VehicleType (VARCHAR(45)): The type of the vehicle.
- Date (DATE): The date of the maintenance.
- ServiceType (VARCHAR(45)): The type of service performed.
- Cost (DECIMAL(10,2)): The cost of the maintenance.
- NextScheduleMaintenance (DATE): The date of the next scheduled maintenance.

Assumption:

- Each VehicleID is unique and cannot be NULL.
- The VehicleType, Date, ServiceType, Cost, and NextScheduleMaintenance can be NULL, indicating that the information is unknown or not provided.

Payment Table

- PaymentID (INT): The unique identifier for a payment.
- Payment_Date (DATE): The date of the payment.
- Nameofstudent (VARCHAR(45)): The name of the student who made the payment.
- Amount (DECIMAL(10,2)): The amount of the payment.
- Email (VARCHAR(45)): The email of the student who made the payment.
- StudentID (INT): The identifier of the student who made the payment.

Assumption:

- Each PaymentID is unique and cannot be NULL.
- The combination of paymentID and studnetID should be unique
- The Payment_Date, Nameofstudent, Amount, Email,can be NULL, indicating that the information is unknown or not provided

Step 6.3 For each table, write the table name and write out the names, data types, and sizes of all the data items, Identify any constraints, using the conventions of the DBMS you will use for implementation.

Office table

- Office_ID (INT): Primary key, NOT NULL.
- Location (VARCHAR(45)): NOT NULL.
- Department (VARCHAR(45)): NOT NULL.
- No_of_Employee (INT): NULL.

Employee table

- EmployeeID (INT): Primary key, NOT NULL.
- Name (VARCHAR(45)): NOT NULL.
- DOB (DATE): NOT NULL.
- Address:
  - apartmentNo (VARCHAR(10)): NULL.
  - city (VARCHAR(45)): NULL.
  - country (VARCHAR(45)): NULL.
- Position (VARCHAR(45)): NULL.
- OfficeID (INT): foreign key referencing office(officeID) NOT NULL.

Student table

- StudentID (INT): Primary key, NOT NULL.
- FirstName (VARCHAR(45)): NOT NULL.

- LastName (VARCHAR(45)): NOT NULL.
- DOB (DATE): NOT NULL.
- ContactNo (VARCHAR(15)): NULL.
- Email (VARCHAR(45)): NULL.
- Sex (CHAR(1)): NULL.
- Address:
  - apartmentNo (VARCHAR(10)): NULL.
  - city (VARCHAR(45)): NULL.
  - country (VARCHAR(45)): NULL.
- OfficeID (INT): foreign key referencing office(officeID) NOT NULL

Instructor table

- InstructorID (INT): Primary key, NOT NULL.
- EmployeeID (INT): foreign key referencing employee(employeeID) NOT NULL.
- FirstName (VARCHAR(45)): NOT NULL.
- LastName (VARCHAR(45)): NOT NULL.
- Address:
  - apartmentNo (VARCHAR(10)): NULL.
  - city (VARCHAR(45)): NULL.
  - country (VARCHAR(45)): NULL.

- VehicleID (INT): foreign key referencing vehicle(vehicleID)NOT NULL.
- Sex (CHAR(1)): NULL.
- Availability (VARCHAR(45)): NULL.

Course table

- CourseID (INT): Primary key, NOT NULL.
- CourseName (VARCHAR(45)): NOT NULL.
- InstructorID (INT): foreign key referencing instructor(instructorID)NOT NULL.
- StudentID (INT): foreign key referencing Student(studentID)NOT NULL.
- StartDate (DATE): NULL.
- EndDate (DATE): NULL.
- Duration (INT): NULL.
- Level (VARCHAR(45)): NULL.

Vehicle table

- VehicleID (INT): Primary key, NOT NULL.
- VehicleMake (VARCHAR(45)): NULL.
- VehicleType (VARCHAR(45)): NULL.
- Model (VARCHAR(45)): NULL.
- Year (YEAR): NULL.
- OfficeID (INT): foreign key referencing office(officeID)NOT NULL.

Maintenance table

- VehicleID (INT): Primary key, NOT NULL.
- VehicleType (VARCHAR(45)): NULL.
- Date (DATE): NULL.
- ServiceType (VARCHAR(45)): NULL.
- Cost (DECIMAL(10,2)): NULL.
- NextScheduleMaintenance (DATE): NULL.

Payment Table

- PaymentID (INT): Primary key, NOT NULL.
- Payment_Date (DATE): NULL.
- Nameofstudent (VARCHAR(45)): NULL.
- Amount (DECIMAL(10,2)): NULL.
- Email (VARCHAR(45)): NULL.
- StudentID (INT): foreign key referencing student(studentID)NOT NULL.

Step 6.5 Create indexes for foreign keys and any other columns that will be used most often for queries.

Indexes for foreign key and other constraints added above in 6.4

• Step 6.6 Insert about five records in each table, preserving all constraints. Put in enough data to demonstrate how the database will function.

```
CREATE DATABASE IF NOT EXISTS glassglow_drivingschool;
USE glassglow_drivingschool;

CREATE TABLE IF NOT EXISTS Office (
    Office_ID INT PRIMARY KEY NOT NULL,
    Location VARCHAR(45) NOT NULL,
    Department VARCHAR(45) NOT NULL,
    No_of_Employee INT,
    UNIQUE INDEX Office_ID_UNIQUE (Office_ID) VISIBLE
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS Employee (
    EmployeeID INT PRIMARY KEY NOT NULL,
    Name VARCHAR(45) NOT NULL,
    DOB DATE NOT NULL,
    apartmentNo VARCHAR(10),
    city VARCHAR(45),
    country VARCHAR(45),
    Position VARCHAR(45),
    OfficeID INT NOT NULL,
    FOREIGN KEY (OfficeID) REFERENCES Office (Office_ID),
    UNIQUE INDEX EmployeeID_UNIQUE (EmployeeID) VISIBLE,
    INDEX fk_Employee_Office_idx (OfficeID) VISIBLE
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS Student (
    StudentID INT PRIMARY KEY NOT NULL,
    FirstName VARCHAR(45) NOT NULL,
    LastName VARCHAR(45) NOT NULL,
    DOB DATE NOT NULL,
    ContactNo VARCHAR(15),
    Email VARCHAR(45),
    Sex CHAR(1),
    apartmentNo VARCHAR(10),
    city VARCHAR(45),
    country VARCHAR(45),
    OfficeID INT NOT NULL,
    FOREIGN KEY (OfficeID) REFERENCES Office (Office_ID),
    UNIQUE INDEX StudentID_UNIQUE (StudentID) VISIBLE,
    INDEX fk_Student_Office_idx (OfficeID) VISIBLE
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS Instructor (
    InstructorID INT PRIMARY KEY NOT NULL,
    EmployeeID INT NOT NULL,
    FirstName VARCHAR(45) NOT NULL,
    LastName VARCHAR(45) NOT NULL,
    apartmentNo VARCHAR(10),
    city VARCHAR(45),
    country VARCHAR(45),
    VehicleID INT NOT NULL,
    Sex CHAR(1),
    Availability VARCHAR(45),
    FOREIGN KEY (EmployeeID) REFERENCES Employee (EmployeeID),
    FOREIGN KEY (VehicleID) REFERENCES Vehicle (VehicleID),
    UNIQUE INDEX InstructorID_UNIQUE (InstructorID) VISIBLE,
    INDEX fk_Instructor_Employee_idx (EmployeeID) VISIBLE,
    INDEX fk_Instructor_Vehicle_idx (VehicleID) VISIBLE
) ENGINE=InnoDB;
```

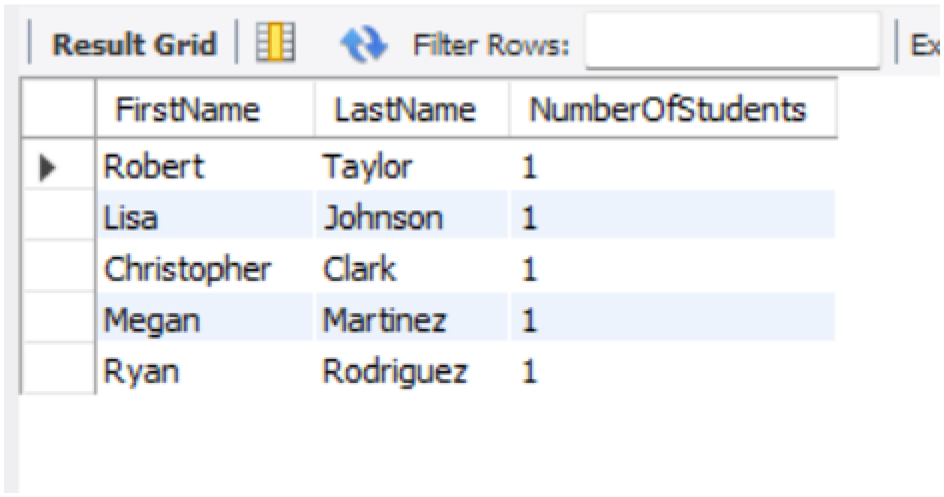
```
CREATE TABLE IF NOT EXISTS Course (
   CourseID INT PRIMARY KEY NOT NULL,
   CourseName VARCHAR(45) NOT NULL,
   InstructorID INT NOT NULL,
   StudentID INT NOT NULL,
   StartDate DATE,
   EndDate DATE,
   Duration INT,
   Level VARCHAR(45),
   FOREIGN KEY (InstructorID) REFERENCES Instructor (InstructorID),
   FOREIGN KEY (StudentID) REFERENCES Student (StudentID),
   UNIQUE INDEX CourseID_UNIQUE (CourseID) VISIBLE,
   INDEX fk_Course_Instructor_idx (InstructorID) VISIBLE,
   INDEX fk_Course_Student_idx (StudentID) VISIBLE
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS Vehicle (
   VehicleID INT PRIMARY KEY NOT NULL,
   VehicleMake VARCHAR(45),
   VehicleType VARCHAR(45),
   Model VARCHAR(45),
   Year YEAR,
   OfficeID INT NOT NULL,
   FOREIGN KEY (OfficeID) REFERENCES Office (Office_ID),
   UNIQUE INDEX VehicleID_UNIQUE (VehicleID) VISIBLE,
   INDEX fk_Vehicle_Office_idx (OfficeID) VISIBLE
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS Maintenance (
   VehicleID INT PRIMARY KEY NOT NULL,
   VehicleType VARCHAR(45),
   Date DATE,
   ServiceType VARCHAR(45),
   Cost DECIMAL(10,2),
   NextScheduleMaintenance DATE,
   FOREIGN KEY (VehicleID) REFERENCES Vehicle (VehicleID),
   UNIQUE INDEX VehicleID_UNIQUE (VehicleID) VISIBLE
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS Payment (
   PaymentID INT PRIMARY KEY NOT NULL,
   Payment_Date DATE,
   Nameofstudent VARCHAR(45),
   Amount DECIMAL(10,2),
    Email VARCHAR(45),
    StudentID INT NOT NULL,
    FOREIGN KEY (StudentID) REFERENCES Student (StudentID),
    UNIQUE INDEX PaymentID_UNIQUE (PaymentID) VISIBLE,
    INDEX fk_Payment_Student_idx (StudentID) VISIBLE
) ENGINE=InnoDB;

-- Insert data into Office table
INSERT INTO Office (Office_ID, Location, Department, No_of_Employee) VALUES
(1, 'Main Street', 'Administration', 10),
(2, 'West Avenue', 'Marketing', 8),
(3, 'East Street', 'Operations', 15),
(4, 'North Boulevard', 'Human Resources', 6),
(5, 'South Square', 'Finance', 12);

-- Insert data into Employee table
INSERT INTO Employee (EmployeeID, Name, DOB, apartmentNo, city, country, Position,
OfficeID) VALUES
(101, 'John Doe', '1985-05-20', 'Apt 101', 'Scotland', 'UK', 'Manager', 1),
(102, 'Jane Smith', '1990-09-15', 'Apt 202', 'Scotland', 'UK', 'Assistant Manager', 2),
(103, 'Michael Johnson', '1988-03-10', 'Apt 303', 'Scotland', 'UK', 'Coordinator', 3),
(104, 'Emily Brown', '1995-07-08', 'Apt 404', 'Scotland', 'UK', 'Marketing Executive', 4),
(105, 'David Williams', '1983-12-25', 'Apt 505', 'Scotland', 'UK', 'Operations Manager', 5);

-- Insert data into Student table
INSERT INTO Student (StudentID, FirstName, LastName, DOB, ContactNo, Email, Sex,
apartmentNo, city, country, OfficeID) VALUES
(201, 'Emma', 'Johnson', '2000-01-15', '123-456-7890', 'emma@example.com', 'F', 'Apt 10',
'Scotland', 'UK', 1),
(202, 'James', 'Miller', '1998-08-20', '987-654-3210', 'james@example.com', 'M', 'Apt 20',
'Scotland', 'UK', 2),
(203, 'Olivia', 'Brown', '2001-05-05', '456-789-0123', 'olivia@example.com', 'F', 'Apt 30',
'Scotland', 'UK', 3),
(204, 'William', 'Davis', '1999-11-10', '789-012-3456', 'william@example.com', 'M', 'Apt 40',
'Scotland', 'UK', 4),
(205, 'Sophia', 'Wilson', '2002-02-28', '321-654-9870', 'sophia@example.com', 'F', 'Apt 50',
'Scotland', 'UK', 5);

-- Insert data into Instructor table
INSERT INTO Instructor (InstructorID, EmployeeID, FirstName, LastName, apartmentNo,
city, country, VehicleID, Sex, Availability) VALUES
(301, 101, 'Robert', 'Taylor', 'Apt 1A', 'Scotland', 'UK', 501, 'M', 'Full-time'),
(302, 102, 'Lisa', 'Johnson', 'Apt 2B', 'Scotland', 'UK', 502, 'F', 'Part-time'),
(303, 103, 'Christopher', 'Clark', 'Apt 3C', 'Scotland', 'UK', 503, 'M', 'Full-time'),
(304, 104, 'Megan', 'Martinez', 'Apt 4D', 'Scotland', 'UK', 504, 'F', 'Full-time'),
(305, 105, 'Ryan', 'Rodriguez', 'Apt 5E', 'Scotland', 'UK', 505, 'M', 'Part-time');

-- Insert data into Course table
INSERT INTO Course (CourseID, CourseName, InstructorID, StudentID, StartDate,
EndDate, Duration, Level) VALUES
(401, 'Beginner Driving Course', 301, 201, '2024-05-01', '2024-05-30', 30, 'Beginner'),
(402, 'Advanced Driving Course', 302, 202, '2024-06-01', '2024-06-30', 30, 'Advanced'),
(403, 'Defensive Driving Course', 303, 203, '2024-07-01', '2024-07-30', 30, 'Intermediate'),
(404, 'City Driving Course', 304, 204, '2024-08-01', '2024-08-30', 30, 'Intermediate'),
(405, 'Highway Driving Course', 305, 205, '2024-09-01', '2024-09-30', 30, 'Advanced');

-- Insert data into Vehicle table
INSERT INTO Vehicle (VehicleID, VehicleMake, VehicleType, Model, Year, OfficeID)
VALUES
(501, 'Toyota', 'Sedan', 'Camry', 2023, 1),
(502, 'Honda', 'SUV', 'CR-V', 2022, 2),
(503, 'Ford', 'Truck', 'F-150', 2021, 3),
(504, 'Chevrolet', 'Hatchback', 'Spark', 2024, 4),
(505, 'BMW', 'Convertible', 'Z4', 2023, 5);

-- Insert data into Maintenance table
INSERT INTO Maintenance (VehicleID, VehicleType, Date, ServiceType, Cost,
NextScheduleMaintenance) VALUES
(501, 'Sedan', '2024-03-15', 'Regular Checkup', 100.00, '2024-09-15'),
(502, 'SUV', '2024-02-20', 'Oil Change', 80.50, '2024-08-20'),
(503, 'Truck', '2024-01-10', 'Brake Inspection', 120.75, '2024-07-10'),
(504, 'Hatchback', '2024-04-05', 'Tire Rotation', 65.00, '2024-10-05'),
(505, 'Convertible', '2024-05-20', 'Full Service', 150.00, '2024-11-20');

-- Insert data into Payment table
INSERT INTO Payment (PaymentID, Payment_Date, Nameofstudent, Amount, Email,
StudentID) VALUES
(601, '2024-05-05', 'Emma Johnson', 200.00, 'emma@example.com', 201),
(602, '2024-06-10', 'James Miller', 250.00, 'james@example.com', 202),
(603, '2024-07-15', 'Olivia Brown', 220.00, 'olivia@example.com', 203),
(604, '2024-08-20', 'William Davis', 180.00, 'william@example.com', 204),
(605, '2024-09-25', 'Sophia Wilson', 300.00, 'sophia@example.com', 205);
```

- **Step 6.7 Write SQL statements that will process five non-routine requests for information**

from the database just created. For each, write the request in English, followed by the corresponding SQL command.

1) Find the total number of students per instructor

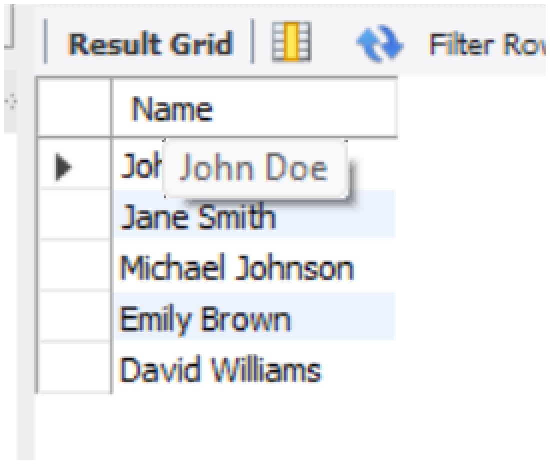
SELECT Instructor.FirstName, Instructor.LastName, COUNT(Student.StudentID) AS NumberOfStudents
FROM Instructor
JOIN Course ON Instructor.InstructorID = Course.InstructorID
JOIN Student ON Course.StudentID = Student.StudentID
GROUP BY Instructor.InstructorID;



FirstName	LastName	NumberOfStudents
Robert	Taylor	1
Lisa	Johnson	1
Christopher	Clark	1
Megan	Martinez	1
Ryan	Rodriguez	1

2) Find the the employee who are also instructor

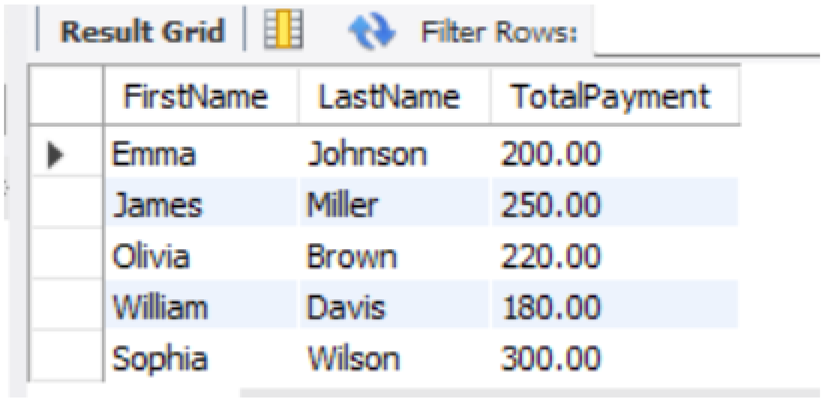
SELECT Employee.Name
FROM Employee
JOIN Instructor ON Employee.EmployeeID = Instructor.EmployeeID;



Name
John Doe
Jane Smith
Michael Johnson
Emily Brown
David Williams

3) Find the total payment received from each student:

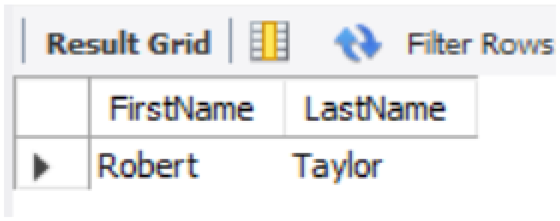
SELECT Student.FirstName, Student.LastName, SUM(Payment.Amount) AS TotalPayment FROM Student
JOIN Payment ON Student.StudentID = Payment.StudentID
GROUP BY Student.StudentID;



FirstName	LastName	TotalPayment
Emma	Johnson	200.00
James	Miller	250.00
Olivia	Brown	220.00
William	Davis	180.00
Sophia	Wilson	300.00

4) Find the instructors who are available full-time and have a vehicle of type ‘Sedan’

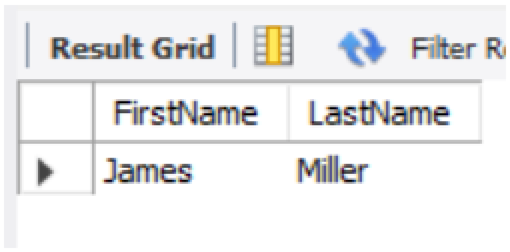
SELECT Instructor.FirstName, Instructor.LastName
FROM Instructor
JOIN Vehicle ON Instructor.VehicleID = Vehicle.VehicleID
WHERE Instructor.Availability = 'Full-time' AND Vehicle.VehicleType = 'Sedan';



	FirstName	LastName
▶	Robert	Taylor

5) Find the students who are older than 25 years

SELECT Student.FirstName, Student.LastName
FROM Student
WHERE YEAR(CURDATE()) - YEAR(Student.DOB) > 25;



	FirstName	LastName
▶	James	Miller