

**Assessment Framework**

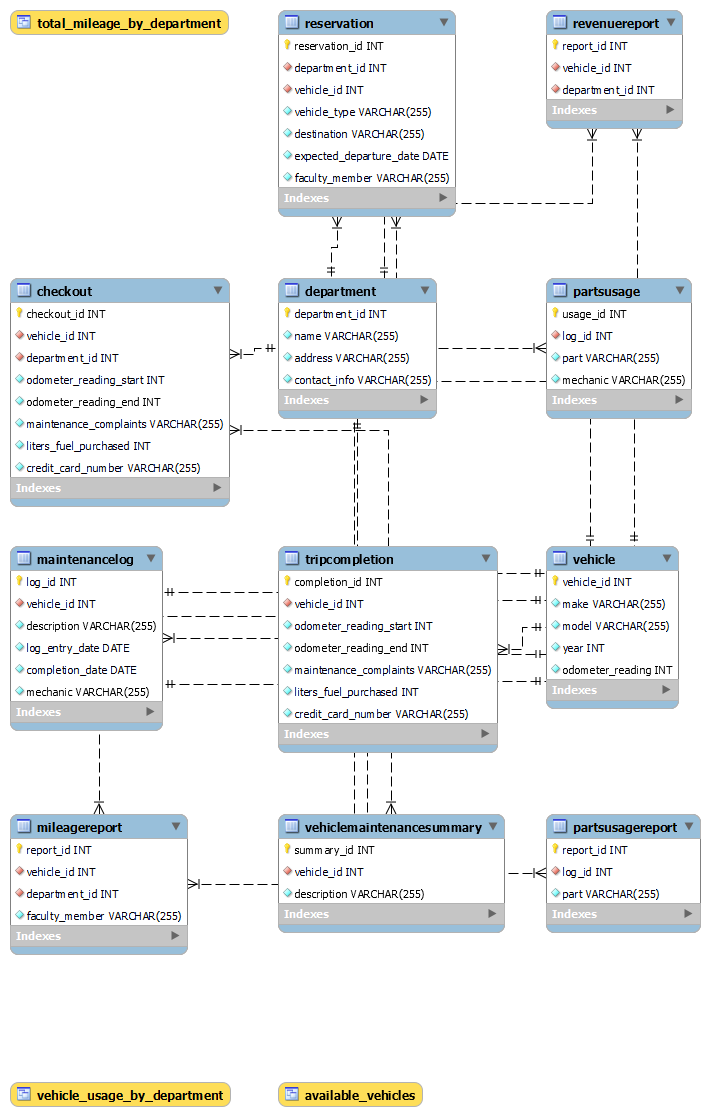
**Assessment Briefing Template 2022-2023**

**CMP2806M Scalable Database Systems**

**Student ID**

**Date of Submission**

1. **Entity Relationship Modelling technique**

****

Designing the Database

The database for the VMC according to the ERD shown above should include the following tables:

• Department: This table will store information about the departments at NCU, such as name, address, and contact information.

• Vehicle: This table will store information about the vehicles owned by NCU, such as make, model, year, and odometer reading.

• Reservation: This table will store reservation requests made by departments, including the department name, vehicle type requested, destination, expected departure date, and name of the authorized faculty member.

• Checkout: This table will store information about vehicles that have been checked out, including the vehicle identification, odometer readings at the start and end of the trip, maintenance complaints (if any), liters of fuel purchased (if any), and the NCU credit card number used to pay for the fuel.

• Trip Completion: This table will store information about completed trips, including the vehicle identification, odometer readings at the start and end of the trip, maintenance complaints (if any), liters of fuel purchased (if any), and the NCU credit card number used to pay for the fuel.

• Maintenance Log: This table will store information about each vehicle's maintenance, including the vehicle identification, a brief description of the type of maintenance required, the initial log entry date, the date on which the vehicle back into service, and the mechanic who released the vehicle back into service.

• Parts Usage: This table will store information about parts used for each vehicle's maintenance, including the maintenance log number, the part used, and the mechanic who signed out the part.

• Mileage Report: This table will store information about the mileage driven by vehicle, by department, and by faculty members with a department.

• Revenue Report: This table will store information about various revenue reports generated by vehicle and department.

• Parts Usage Report: This table will store information about the parts usage report issued each month.

• Vehicle Maintenance Summary: This table will store information about the vehicle maintenance summary issued each month.

The entities chosen for the VMC database are necessary to store the information needed to properly track the vehicles owned by NCU, the reservations made by departments, the checkouts and trips completed, and the maintenance logs, parts used, and various reports. The relationships between the entities were chosen based on the data that needs to be tracked and stored in the database. For example, the Department and Vehicle entities have a one-to-many relationship, as each department can have multiple vehicles associated with it. Similarly, the Checkout and Trip Completion entities have a one-to-one relationship, as each checkout corresponds to a single trip completion. The Parts Usage and Mileage Report entities have a many-to-many relationship, as each parts usage can generate multiple mileage reports, and each mileage report can be generated from multiple parts usages. Finally, the Vehicle Maintenance Summary entity has a one-to-many relationship with the Maintenance Log entity, as each summary report can contain information from multiple maintenance logs.

The entities chosen are based on the functionality necessary to successfully run the VMC. The Department table stores essential information about the departments at NCU. The Vehicle table stores information about the vehicles owned by NCU. The Reservation table stores data about reservation requests made by departments. The Checkout table stores information about vehicles that have been checked out. The Trip Completion table stores information about trips that have been completed. The Maintenance Log table stores information about each vehicle's maintenance. The Parts Usage table stores information about parts used for each vehicle's maintenance. The Mileage Report table stores information about the mileage driven by vehicle, by department, and by faculty members with a department. The Revenue Report table stores information about various revenue reports generated by vehicle and department. The Parts Usage Report table stores information about the parts usage report issued each month.

Finally, the Vehicle Maintenance Summary table stores information about the vehicle maintenance summary issued each month. The relationships between the entities are based on how data is used to support the VMC operations. For example, the Reservation table is related to the Department table by the department name, and the Checkout table is related to the Reservation table by the vehicle identification. The Maintenance Log table is related to the Vehicle table by the vehicle identification, and the Parts Usage table is related to the Maintenance Log table by the maintenance log number. The Mileage Report table is related to the Vehicle table by the vehicle identification, and the Revenue Report table is related to the Department table by the department name. The Parts Usage Report table is related to the Parts Usage table by the part used, and the Vehicle Maintenance Summary table is related to the Maintenance Log table by the vehicle identification.

1. **You are expected to implement your design into fully working relational database, that can meet all business requirements. Create Tables, views, stored procedures, triggers, etc. when necessary.**

**-- CREATE DATABASE NCU;**

**-- use NCU;**

**-- CREATE TABLES IN NCU DATABASE**

CREATE TABLE Department (

department\_id int NOT NULL PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR (255) NOT NULL,

address VARCHAR (255) NOT NULL,

contact info VARCHAR (255) NOT NULL

);

CREATE TABLE Vehicle (

vehicle\_id int NOT NULL PRIMARY KEY AUTO\_INCREMENT,

make VARCHAR (255) NOT NULL,

model VARCHAR (255) NOT NULL,

year int NOT NULL,

odometer\_reading int NOT NULL

);

CREATE TABLE Reservation (

reservation\_id int NOT NULL PRIMARY KEY AUTO\_INCREMENT,

department\_id int NOT NULL,

vehicle\_id int NOT NULL,

vehicle\_type VARCHAR(255) NOT NULL,

destination VARCHAR(255) NOT NULL,

expected\_departure\_date DATE NOT NULL,

faculty\_member VARCHAR(255) NOT NULL,

FOREIGN KEY (vehicle\_id) REFERENCES Vehicle(vehicle\_id),

FOREIGN KEY (department\_id) REFERENCES Department(department\_id)

);

CREATE TABLE Checkout (

checkout\_id int NOT NULL PRIMARY KEY AUTO\_INCREMENT,

vehicle\_id int NOT NULL,

department\_id int NOT NULL,

odometer\_reading\_start int NOT NULL,

odometer\_reading\_end int NOT NULL,

maintenance\_complaints VARCHAR(255) NOT NULL,

liters\_fuel\_purchased int NOT NULL,

credit\_card\_number VARCHAR(255) NOT NULL,

FOREIGN KEY (vehicle\_id) REFERENCES Vehicle(vehicle\_id),

FOREIGN KEY (department\_id) REFERENCES Department(department\_id)

);

CREATE TABLE TripCompletion (

completion\_id int NOT NULL PRIMARY KEY AUTO\_INCREMENT,

vehicle\_id int NOT NULL,

odometer\_reading\_start int NOT NULL,

odometer\_reading\_end int NOT NULL,

maintenance\_complaints VARCHAR(255) NOT NULL,

liters\_fuel\_purchased int NOT NULL,

credit\_card\_number VARCHAR(255) NOT NULL,

FOREIGN KEY (vehicle\_id) REFERENCES Vehicle(vehicle\_id)

);

CREATE TABLE MaintenanceLog (

log\_id int NOT NULL PRIMARY KEY AUTO\_INCREMENT,

vehicle\_id int NOT NULL,

description VARCHAR(255) NOT NULL,

log\_entry\_date DATE NOT NULL,

completion\_date DATE NOT NULL,

mechanic VARCHAR(255) NOT NULL,

FOREIGN KEY (vehicle\_id) REFERENCES Vehicle(vehicle\_id)

);

CREATE TABLE PartsUsage (

usage\_id int NOT NULL PRIMARY KEY AUTO\_INCREMENT,

log\_id int NOT NULL,

part VARCHAR(255) NOT NULL,

mechanic VARCHAR(255) NOT NULL,

FOREIGN KEY (log\_id) REFERENCES MaintenanceLog(log\_id)

);

CREATE TABLE MileageReport (

report\_id int NOT NULL PRIMARY KEY AUTO\_INCREMENT,

vehicle\_id int NOT NULL,

department\_id int NOT NULL,

faculty\_member VARCHAR(255) NOT NULL,

FOREIGN KEY (vehicle\_id) REFERENCES Vehicle(vehicle\_id),

FOREIGN KEY (department\_id) REFERENCES Department(department\_id)

);

CREATE TABLE RevenueReport (

report\_id int NOT NULL PRIMARY KEY AUTO\_INCREMENT,

vehicle\_id int NOT NULL,

department\_id int NOT NULL,

FOREIGN KEY (vehicle\_id) REFERENCES Vehicle(vehicle\_id),

FOREIGN KEY (department\_id) REFERENCES Department(department\_id)

);

CREATE TABLE PartsUsageReport (

report\_id int NOT NULL PRIMARY KEY AUTO\_INCREMENT,

log\_id int NOT NULL,

part VARCHAR(255) NOT NULL,

FOREIGN KEY (log\_id) REFERENCES MaintenanceLog(log\_id)

);

CREATE TABLE VehicleMaintenanceSummary (

summary\_id int NOT NULL PRIMARY KEY AUTO\_INCREMENT,

vehicle\_id int NOT NULL,

description VARCHAR(255) NOT NULL,

FOREIGN KEY (vehicle\_id) REFERENCES Vehicle(vehicle\_id)

);

-- CREATE VIEWS

CREATE VIEW available\_vehicles AS

SELECT v.vehicle\_id, v.make, v.model, v.year, v.odometer\_reading

FROM Vehicle v

LEFT JOIN Checkout c ON v.vehicle\_id = c.vehicle\_id

WHERE c.checkout\_id IS NULL;

CREATE VIEW vehicle\_usage\_by\_department AS

SELECT d.name, COUNT(r.reservation\_id) AS num\_reservations

FROM Department d

INNER JOIN Reservation r ON d.department\_id = r.department\_id

GROUP BY d.name;

CREATE VIEW total\_mileage\_by\_department AS

SELECT d.name, SUM(m.odometer\_reading\_end - m.odometer\_reading\_start) AS total\_mileage

FROM Department d

INNER JOIN Reservation r ON d.department\_id = r.department\_id

INNER JOIN TripCompletion m ON r.reservation\_id = m.completion\_id

GROUP BY d.name;

-- CREATE PROCEDURES

DELIMITER //

CREATE PROCEDURE insert\_department(IN name VARCHAR(255), IN address VARCHAR(255), IN contact\_info VARCHAR(255))

BEGIN

INSERT INTO Department(name, address, contact\_info)

VALUES(name, address, contact\_info);

END

CREATE PROCEDURE insert\_vehicle(IN make VARCHAR(255), IN model VARCHAR(255), IN year INT, IN odometer\_reading INT)

BEGIN

INSERT INTO Vehicle(make, model, year, odometer\_reading)

VALUES(make, model, year, odometer\_reading);

END

CREATE PROCEDURE insert\_reservation(IN department\_id INT, IN vehicle\_type VARCHAR(255), IN destination VARCHAR(255), IN expected\_departure\_date DATE, IN faculty\_member VARCHAR(255))

BEGIN

INSERT INTO Reservation(department\_id, vehicle\_type, destination, expected\_departure\_date, faculty\_member)

VALUES(department\_id, vehicle\_type, destination, expected\_departure\_date, faculty\_member);

END

CREATE PROCEDURE insert\_checkout(IN vehicle\_id INT, IN odometer\_reading\_start INT, IN odometer\_reading\_end INT, IN maintenance\_complaints VARCHAR(255), IN liters\_fuel\_purchased INT, IN credit\_card\_number VARCHAR(255))

BEGIN

INSERT INTO Checkout(vehicle\_id, odometer\_reading\_start, odometer\_reading\_end, maintenance\_complaints, liters\_fuel\_purchased, credit\_card\_number)

VALUES(vehicle\_id, odometer\_reading\_start, odometer\_reading\_end, maintenance\_complaints, liters\_fuel\_purchased, credit\_card\_number);

END

CREATE PROCEDURE insert\_trip\_completion(IN vehicle\_id INT, IN odometer\_reading\_start INT, IN odometer\_reading\_end INT, IN maintenance\_complaints VARCHAR(255), IN liters\_fuel\_purchased INT, IN credit\_card\_number VARCHAR(255))

BEGIN

INSERT INTO TripCompletion(vehicle\_id, odometer\_reading\_start, odometer\_reading\_end, maintenance\_complaints, liters\_fuel\_purchased, credit\_card\_number)

VALUES(vehicle\_id, odometer\_reading\_start, odometer\_reading\_end, maintenance\_complaints, liters\_fuel\_purchased, credit\_card\_number);

END

CREATE PROCEDURE insert\_maintenance\_log(IN vehicle\_id INT, IN description VARCHAR(255), IN log\_entry\_date DATE, IN completion\_date DATE, IN mechanic VARCHAR(255))

BEGIN

INSERT INTO MaintenanceLog(vehicle\_id, description, log\_entry\_date, completion\_date, mechanic)

VALUES(vehicle\_id, description, log\_entry\_date, completion\_date, mechanic);

END

CREATE PROCEDURE insert\_parts\_usage(IN log\_id INT, IN part VARCHAR(255), IN mechanic VARCHAR(255))

BEGIN

INSERT INTO PartsUsage(log\_id, part, mechanic)

VALUES(log\_id, part, mechanic);

END

CREATE PROCEDURE insert\_mileage\_report(IN vehicle\_id INT, IN department\_id INT, IN faculty\_member VARCHAR(255))

BEGIN

INSERT INTO MileageReport(vehicle\_id, department\_id, faculty\_member)

VALUES(vehicle\_id, department\_id, faculty\_member);

END

CREATE PROCEDURE insert\_revenue\_report(IN vehicle\_id INT, IN department\_id INT)

BEGIN

INSERT INTO RevenueReport(vehicle\_id, department\_id)

VALUES(vehicle\_id, department\_id);

END

CREATE PROCEDURE insert\_parts\_usage\_report(IN log\_id INT, IN part VARCHAR(255))

BEGIN

INSERT INTO PartsUsageReport(log\_id, part)

VALUES(log\_id, part);

END

CREATE PROCEDURE insert\_vehicle\_maintenance\_summary(IN vehicle\_id INT, IN description VARCHAR(255))

BEGIN

INSERT INTO VehicleMaintenanceSummary(vehicle\_id, description)

VALUES(vehicle\_id, description);

END;

DELIMITER ;

-- CREATE TRIGGERS

DELIMITER //

CREATE TRIGGER update\_checkout\_on\_update

BEFORE UPDATE ON Checkout

FOR EACH ROW

BEGIN

IF NEW.odometer\_reading\_end < OLD.odometer\_reading\_start THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Odometer reading end cannot be less than odometer reading start.';

END IF;

END;

CREATE TRIGGER update\_trip\_completion\_on\_update

BEFORE UPDATE ON TripCompletion

FOR EACH ROW

BEGIN

IF NEW.odometer\_reading\_end < OLD.odometer\_reading\_start THEN

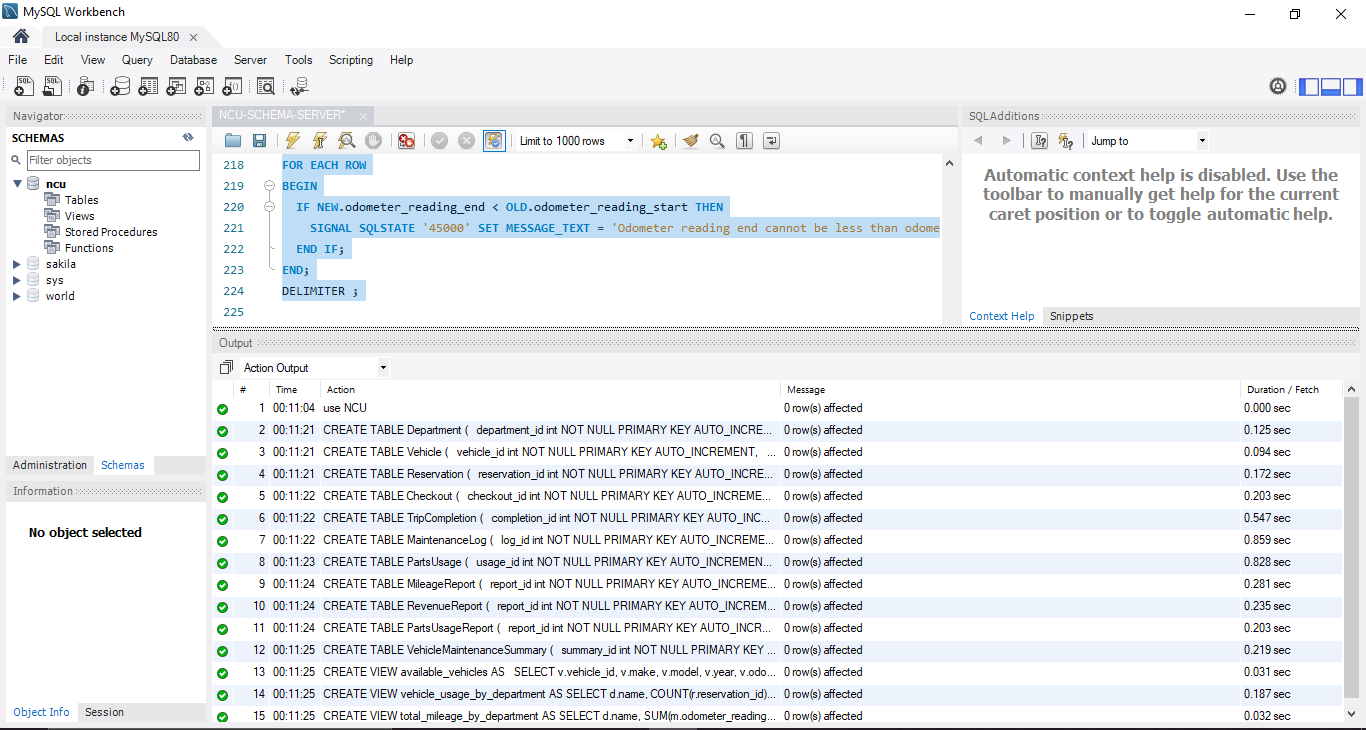
SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Odometer reading end cannot be less than odometer reading start.';

END IF;

END;

DELIMITER ;

-- **Output SHOWING SUCCESSFUL DATABASE AND RELATIONS CREATIONS**

****

1. **Once the database is created, you need to populate some arbitrary data into the tables to test the business transactions.**

-- **CREATE INSERT STATEMENTS**

INSERT INTO Department (name, address, contact\_info) VALUES

('Business', '123 Main Street', '123-456-7890'),

('Engineering', '456 Second Street', '123-456-7891'),

('Humanities', '789 Third Street', '123-456-7892'),

('Nursing', '321 Fourth Street', '123-456-7893'),

('Science', '654 Fifth Street', '123-456-7894');

INSERT INTO Vehicle (make, model, year, odometer\_reading) VALUES

('Ford', 'Focus', 2018, 10000),

('Chevrolet', 'Malibu', 2019, 20000),

('Honda', 'Accord', 2020, 30000),

('Toyota', 'Corolla', 2021, 40000),

('Nissan', 'Altima', 2022, 50000);

INSERT INTO Reservation (department\_id, vehicle\_id, vehicle\_type, destination, expected\_departure\_date, faculty\_member) VALUES

(1, 1, 'Van', 'New York', '2022-01-01', 'John Smith'),

(2, 2, 'SUV', 'Chicago', '2022-02-01', 'Jane Doe'),

(3, 3, 'Sedan', 'Los Angeles', '2022-03-01', 'Bob Jones'),

(4, 4, 'Truck', 'Houston', '2022-04-01', 'Mary Smith'),

(5, 5, 'Coupe', 'Philadelphia', '2022-05-01', 'John Doe');

INSERT INTO Checkout (vehicle\_id, department\_id, odometer\_reading\_start, odometer\_reading\_end, maintenance\_complaints, liters\_fuel\_purchased, credit\_card\_number) VALUES

(1, 1, 10000, 11000, 'None', 10, '12345'),

(2, 2, 20000, 21000, 'None', 20, '23456'),

(3, 3, 30000, 31000, 'None', 30, '34567'),

(4, 4, 40000, 41000, 'None', 40, '45678'),

(5, 5, 50000, 51000, 'None', 50, '56789');

INSERT INTO TripCompletion (vehicle\_id, odometer\_reading\_start, odometer\_reading\_end, maintenance\_complaints, liters\_fuel\_purchased, credit\_card\_number) VALUES

(1, 11000, 12000, 'None', 10, '12345'),

(2, 21000, 22000, 'None', 20, '23456'),

(3, 31000, 32000, 'None', 30, '34567'),

(4, 41000, 42000, 'None', 40, '45678'),

(5, 51000, 52000, 'None', 50, '56789');

INSERT INTO MaintenanceLog (vehicle\_id, description, log\_entry\_date, completion\_date, mechanic) VALUES

(1, 'Replace tires', '2022-01-01', '2022-01-02', 'Bob Jones'),

(2, 'Replace brakes', '2022-02-01', '2022-02-02', 'John Smith'),

(3, 'Replace spark plugs', '2022-03-01', '2022-03-02', 'Jane Doe'),

(4, 'Replace oil filter', '2022-04-01', '2022-04-02', 'Mary Smith'),

(5, 'Replace air filter', '2022-05-01', '2022-05-02', 'John Doe');

INSERT INTO PartsUsage (log\_id, part, mechanic) VALUES

(1, 'Tires', 'Bob Jones'),

(2, 'Brakes', 'John Smith'),

(3, 'Spark Plugs', 'Jane Doe'),

(4, 'Oil Filter', 'Mary Smith'),

(5, 'Air Filter', 'John Doe');

INSERT INTO MileageReport (vehicle\_id, department\_id, faculty\_member) VALUES

(1, 1, 'John Smith'),

(2, 2, 'Jane Doe'),

(3, 3, 'Bob Jones'),

(4, 4, 'Mary Smith'),

(5, 5, 'John Doe');

INSERT INTO RevenueReport (vehicle\_id, department\_id) VALUES

(1, 1),

(2, 2),

(3, 3),

(4, 4),

(5, 5);

INSERT INTO PartsUsageReport (log\_id, part) VALUES

(1, 'Tires'),

(2, 'Brakes'),

(3, 'Spark Plugs'),

(4, 'Oil Filter'),

(5, 'Air Filter');

INSERT INTO VehicleMaintenanceSummary (vehicle\_id, description) VALUES

(1, 'Replace tires'),

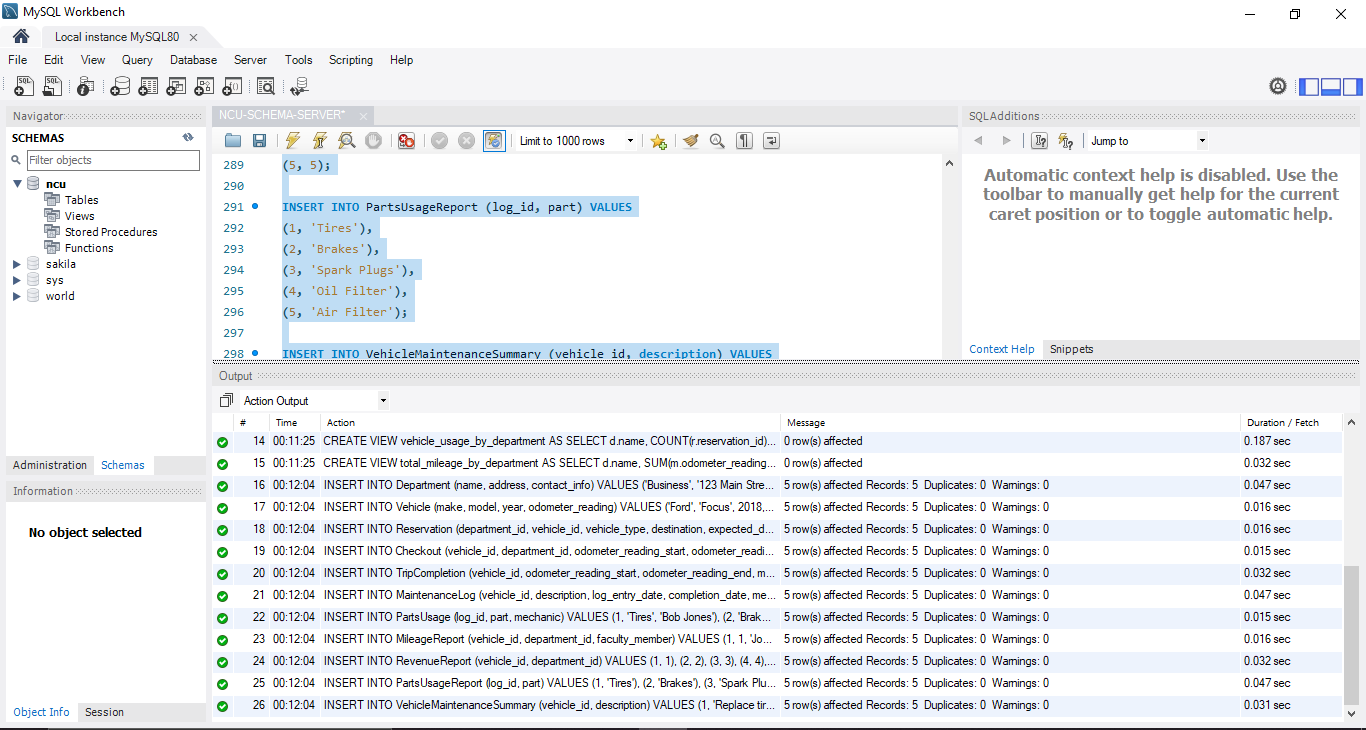
(2, 'Replace brakes'),

(3, 'Replace spark plugs'),

(4, 'Replace oil filter'),

(5, 'Replace air filter');

-- **Output SHOWING SUCCESSFUL DATA INSERTION IN OUR DATABASE RELATIONS**

****

1. **Practically test your database through the use of appropriately designed SQL statements for the provided transactions (refer to section 1); you need to clearly explain the idea of each of your solutions in the report.**

NCU has provided some examples of typical queries that the database system must support:

● Display all available vehicles on a given date.

● Display how many vehicles each department has used so far.

● Display the total mileage driven by a department this year.

● Show details of a particular bill.

● Display those who booked vehicles but not actually used them.

-- **Querires**

-- **Display all available vehicles on a given date:**

SELECT Vehicle.make, Vehicle.model, Vehicle.year

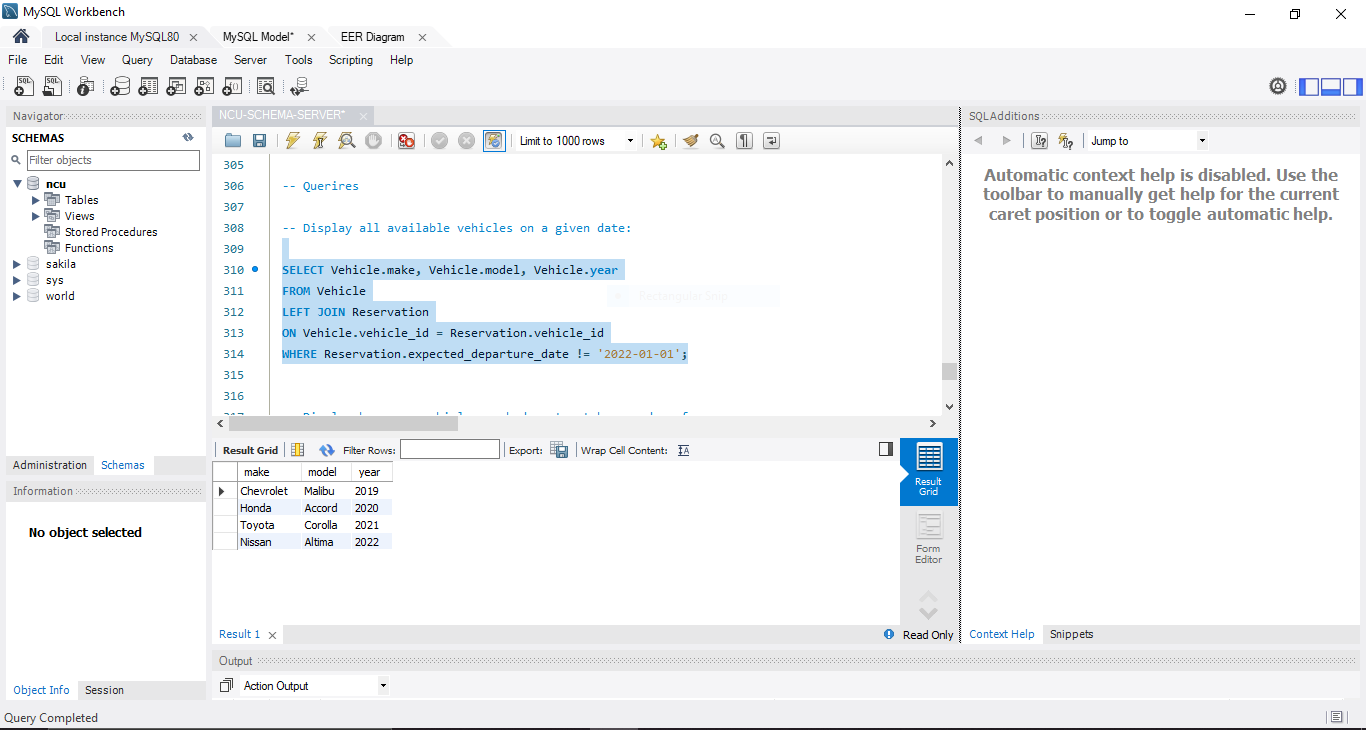
FROM Vehicle

LEFT JOIN Reservation

ON Vehicle.vehicle\_id = Reservation.vehicle\_id

WHERE Reservation.expected\_departure\_date != '2022-01-01';

-- **Output**



-- **Display how many vehicles each department has used so far:**

SELECT Department.name, COUNT(Reservation.department\_id) AS total\_vehicles\_used

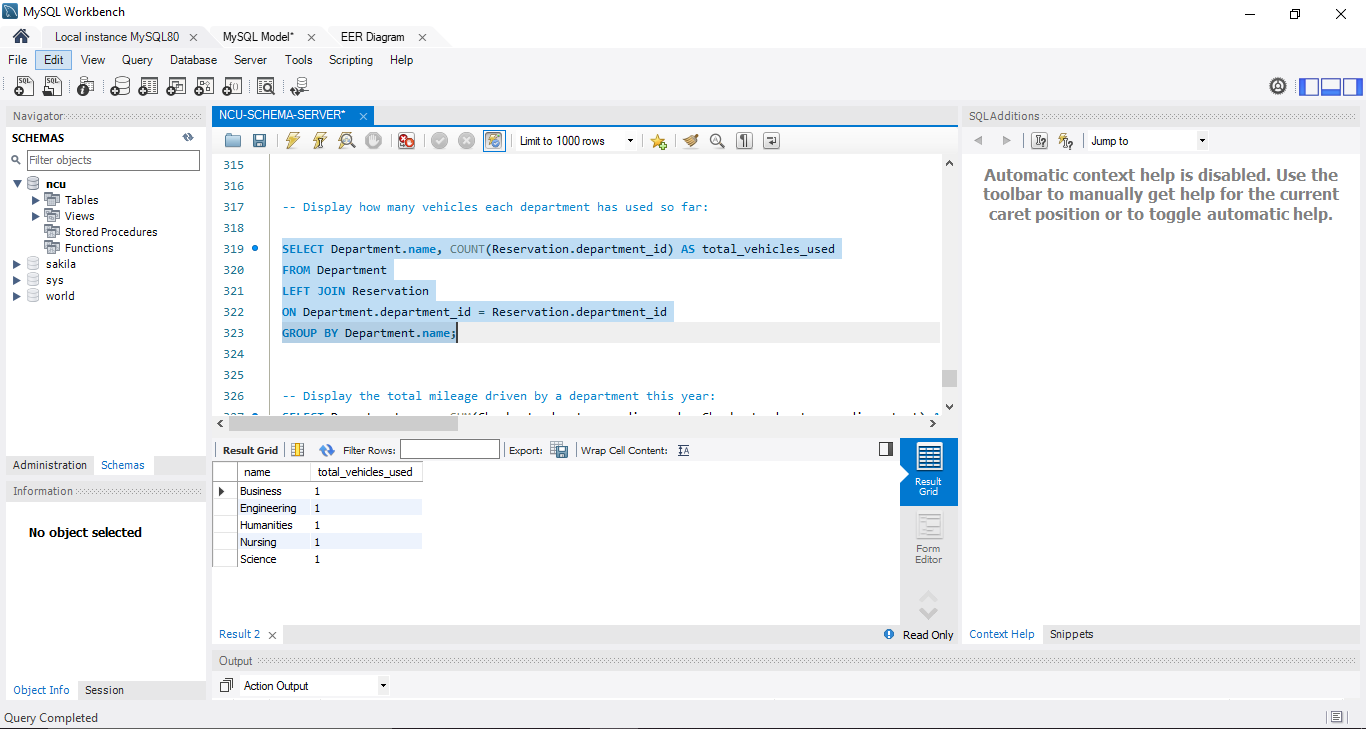
FROM Department

LEFT JOIN Reservation

ON Department.department\_id = Reservation.department\_id

GROUP BY Department.name;

-- **Output**



-- **Display the total mileage driven by a department this year:**

SELECT Department.name, SUM(Checkout.odometer\_reading\_end - Checkout.odometer\_reading\_start) AS total\_mileage\_this\_year

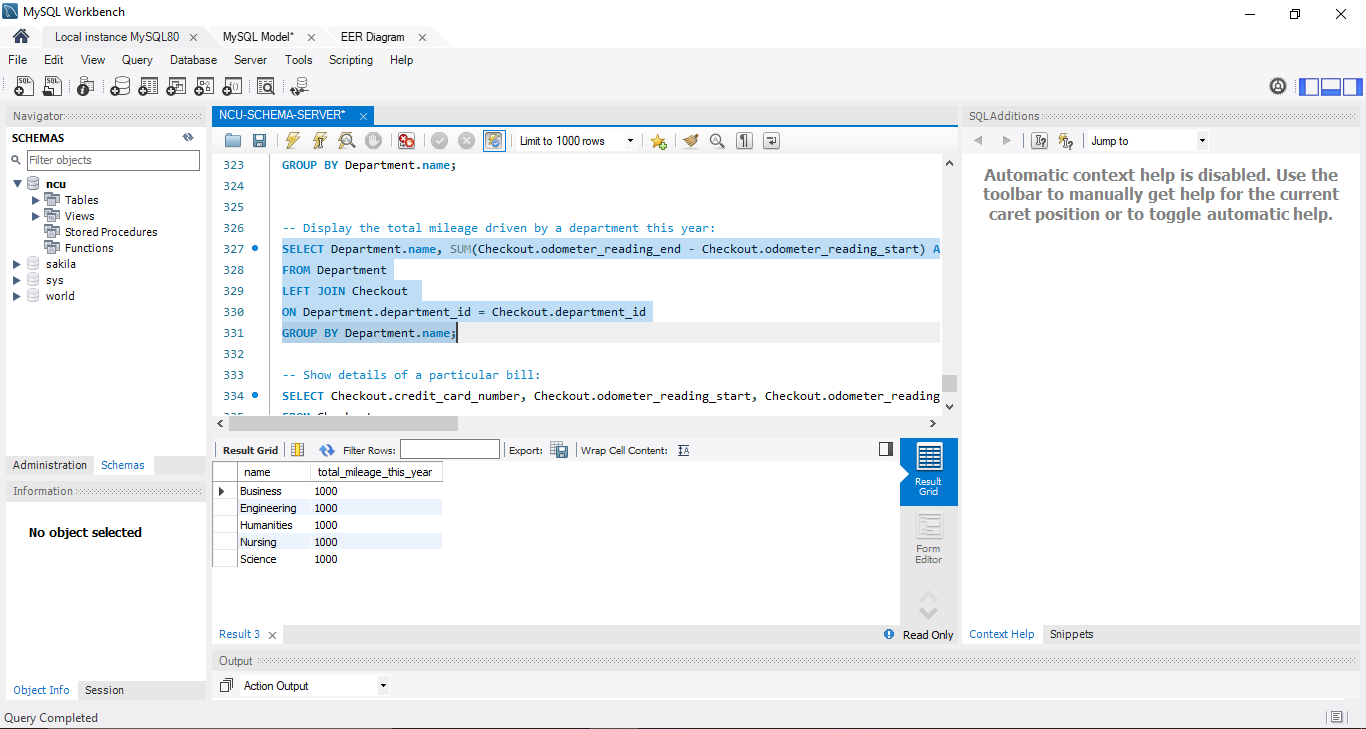
FROM Department

LEFT JOIN Checkout

ON Department.department\_id = Checkout.department\_id

GROUP BY Department.name;

-- **Output**



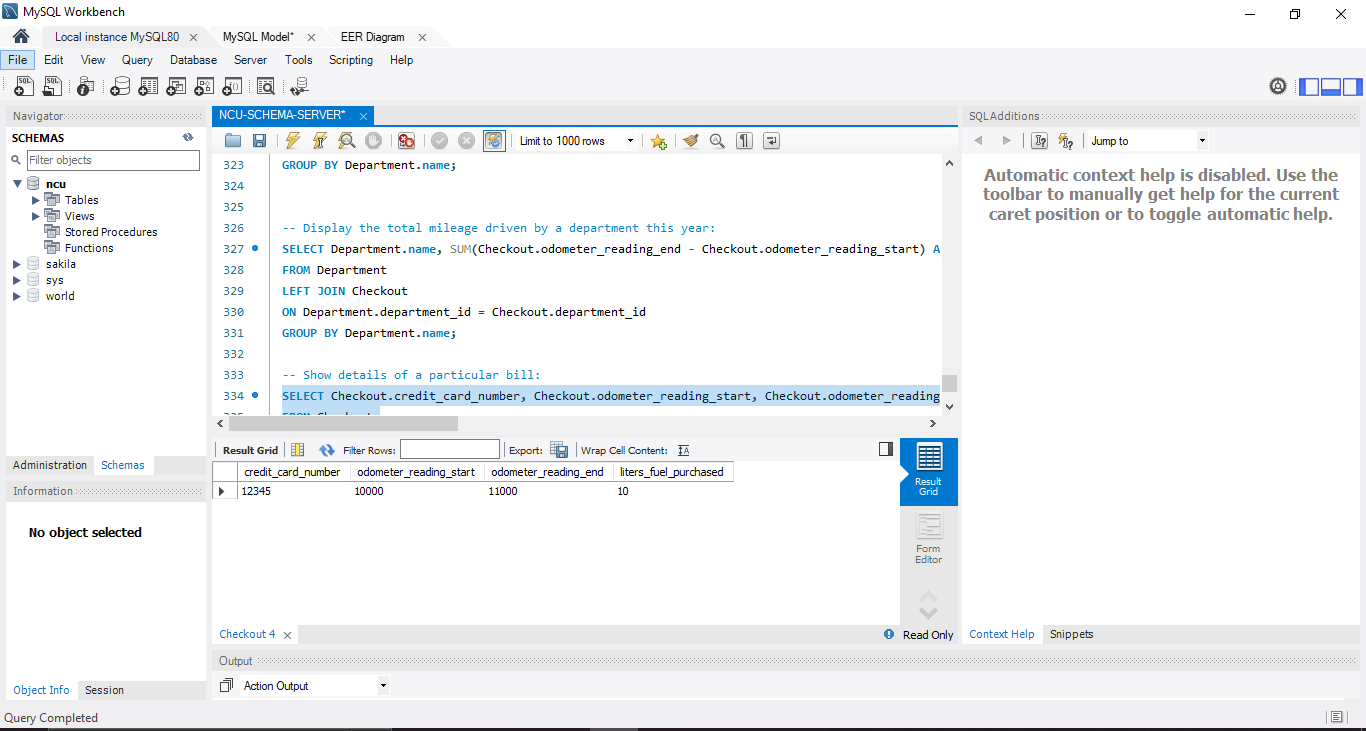
-- **Show details of a particular bill:**

SELECT Checkout.credit\_card\_number, Checkout.odometer\_reading\_start, Checkout.odometer\_reading\_end, Checkout.liters\_fuel\_purchased

FROM Checkout

WHERE Checkout.credit\_card\_number = '12345';

-- **Output**



-- **Display those who booked vehicles but not actually used them:**

SELECT Reservation.faculty\_member

FROM Reservation

LEFT JOIN TripCompletion

ON Reservation.vehicle\_id = TripCompletion.vehicle\_id

WHERE TripCompletion.vehicle\_id IS NULL;

-- **Output**

