Task1 Database Design

Design a SQL schema for a Courier Management System with tables for Customers, Couriers, Orders, and Parcels. Define the relationships between these tables using appropriate foreign keys.

Requirements:

• Define the Database Schema • Create SQL tables for entities such as User, Courier, Employee, Location,Payment

• Define relationships between these tables (one-to-many, many-to-many, etc.).

```
CREATE DATABASE CMS;

USE CMS;

CREATE TABLE User (

    ->      UserID INT PRIMARY KEY NOT NULL,

    ->      Name VARCHAR(255),

    ->      Email VARCHAR(255) UNIQUE,

    ->      Password VARCHAR(255),

    ->      ContactNumber VARCHAR(20),

    ->      Address TEXT

    -> );


CREATE TABLE Courier (

    ->      CourierID INT PRIMARY KEY NOT NULL ,

    ->      SenderName VARCHAR(255),

    ->      SenderAddress TEXT,

    ->      ReceiverName VARCHAR(255),

    ->      ReceiverAddress TEXT,

    ->      Weight DECIMAL(5, 2),
    ->          Status VARCHAR(50),
    ->          TrackingNumber VARCHAR(20) UNIQUE,
    ->          DeliveryDate DATE
    -> );

CREATE TABLE CourierServices (
    ->          ServiceID INT PRIMARY KEY NOT NULL,
    ->          ServiceName VARCHAR(100),
    ->          Cost DECIMAL(8, 2)
    -> );

CREATE TABLE Employee (
    ->          EmployeeID INT PRIMARY KEY NOT NULL ,
    ->          Name VARCHAR(255),
    ->          Email VARCHAR(255) UNIQUE,
    ->          ContactNumber VARCHAR(20),
    ->          Role VARCHAR(50),
    ->          Salary DECIMAL(10, 2)
    -> );

CREATE TABLE Location (
    ->          LocationID INT PRIMARY KEY NOT NULL,
    ->          LocationName VARCHAR(100),
    ->          Address TEXT
    -> );

CREATE TABLE Payment (
    ->          PaymentID INT PRIMARY KEY NOT NULL,
    ->          CourierID INT,
    ->          LocationID INT,

    ->          Amount DECIMAL(10, 2),

    ->          PaymentDate DATE,

    ->          FOREIGN KEY (CourierID) REFERENCES Couriers(CourierID),

    ->          FOREIGN KEY (LocationID) REFERENCES Location(LocationID)

    -> );
```

• Populate Sample Data

• Insert sample data into the tables to simulate real-world scenarios.

```
  INSERT INTO User (UserID, Name, Email, Password, ContactNumber, Address)

        -> VALUES

        ->          (1, 'John Doe', 'john.doe@example.com', 'password123', '123-456-7890', '123 Main St, City,
Country'),

        ->          (2, 'Jane Smith', 'jane.smith@example.com', 'securepass', '987-654-3210', '456 Oak St,
Town, Country'),

        ->          (3, 'Alice Johnson', 'alice.johnson@example.com', 'pass123', '555-123-4567', '789 Pine St,
Village, Country'),

        ->          (4, 'Bob Williams', 'bob.williams@example.com', 'mysecret', '222-333-4444', '101 Cedar St,
Hamlet, Country'),

        ->          (5, 'Eva Brown', 'eva.brown@example.com', 'password456', '777-888-9999', '202 Elm St,
City, Country'),

        ->          (6, 'David Taylor', 'david.taylor@example.com', 'strongpass', '111-222-3333', '303 Maple St,
Town, Country'),

        ->          (7, 'Sophia Lee', 'sophia.lee@example.com', 'secure123', '999-888-7777', '404 Birch St,
Village, Country'),

        ->          (8, 'Michael Davis', 'michael.davis@example.com', 'mypass', '444-555-6666', '505 Pine St,
Hamlet, Country'),

        ->          (9, 'Olivia White', 'olivia.white@example.com', 'pass456', '666-777-8888', '606 Oak St, City,
Country'),
        ->      (10, 'Daniel Johnson', 'daniel.johnson@example.com', 'mypassword', '333-444-5555', '707 Elm St, Town, Country');


INSERT INTO Courier (CourierID, SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate)

        -> VALUES

        ->      (1, 'John Sender', '123 Sender St, City, Country', 'Jane Receiver', '456 Receiver St, Town, Country', 2.5, 'In Transit', 'TN123456', '2024-01-18'),

        ->      (2, 'Alice Shipper', '789 Shipper St, Village, Country', 'Bob Recipient', '101 Recipient St, Hamlet, Country', 1.8, 'Delivered', 'TN789012', '2024-01-19'),

        ->      (3, 'Eva Sender', '202 Sender St, City, Country', 'David Receiver', '303 Receiver St, Town, Country', 3.0, 'Pending', 'TN345678', NULL),

        ->      (4, 'Sophia Shipper', '404 Shipper St, Village, Country', 'Michael Recipient', '505 Recipient St, Hamlet, Country', 1.2, 'In Transit', 'TN901234', '2024-01-20'),

        ->      (5, 'Olivia Sender', '606 Sender St, City, Country', 'Daniel Receiver', '707 Receiver St, Town, Country', 2.7, 'Delivered', 'TN567890', '2024-01-21'),

        ->      (6, 'Daniel Shipper', '707 Shipper St, Village, Country', 'Sophia Recipient', '404 Recipient St, Hamlet, Country', 1.5, 'Delivered', 'TN234567', '2024-01-22'),

        ->      (7, 'Michael Sender', '505 Sender St, City, Country', 'Olivia Receiver', '606 Receiver St, Town, Country', 2.0, 'Pending', 'TN890123', NULL),

        ->      (8, 'Bob Shipper', '101 Shipper St, Village, Country', 'Eva Recipient', '202 Recipient St, Hamlet, Country', 2.3, 'In Transit', 'TN456789', '2024-01-23'),

        ->      (9, 'Jane Sender', '456 Sender St, City, Country', 'Alice Receiver', '789 Receiver St, Town, Country', 1.9, 'Delivered', 'TN012345', '2024-01-24'),

        ->      (10, 'David Shipper', '303 Shipper St, Village, Country', 'John Recipient', '123 Recipient St, Hamlet, Country', 2.8, 'Pending', 'TN678901', NULL);


INSERT INTO Employee (EmployeeID, Name, Email, ContactNumber, Role, Salary)

        -> VALUES

        ->      (1, 'Alice Johnson', 'alice.johnson@example.com', '555-123-4567', 'Manager', 70000.00),
        ->      (2, 'Bob Williams', 'bob.williams@example.com', '222-333-4444', 'Delivery Driver', 50000.00),
        ->      (3, 'Eva Brown', 'eva.brown@example.com', '777-888-9999', 'Customer Service Representative', 60000.00),
        ->      (4, 'David Taylor', 'david.taylor@example.com', '111-222-3333', 'Warehouse Staff', 55000.00),
        ->      (5, 'Sophia Lee', 'sophia.lee@example.com', '999-888-7777', 'IT Specialist', 75000.00),
        ->      (6, 'Michael Davis', 'michael.davis@example.com', '444-555-6666', 'Manager', 72000.00),
        ->      (7, 'Olivia White', 'olivia.white@example.com', '666-777-8888', 'Customer Service Representative', 58000.00),
        ->      (8, 'Daniel Johnson', 'daniel.johnson@example.com', '333-444-5555', 'Warehouse Staff', 52000.00),
        ->      (9, 'John Doe', 'john.doe@example.com', '123-456-7890', 'Delivery Driver', 50000.00),
        ->      (10, 'Jane Smith', 'jane.smith@example.com', '987-654-3210', 'IT Specialist', 76000.00);



INSERT INTO Location (LocationID, LocationName, Address)
        -> VALUES
        ->      (1, 'Warehouse A', '123 Main St, City, Country'),
        ->      (2, 'Office Building', '456 Business St, Town, Country'),
        ->      (3, 'Distribution Center', '789 Logistics St, Village, Country'),
        ->      (4, 'Hub Facility', '101 Hub St, Hamlet, Country'),
        ->      (5, 'Regional Office', '202 Regional St, City, Country'),
        ->      (6, 'Storage Facility', '303 Storage St, Town, Country'),
        ->      (7, 'Branch Office', '404 Branch St, Village, Country'),
        ->      (8, 'Central Depot', '505 Depot St, Hamlet, Country'),
        ->      (9, 'Main Office', '606 Main St, City, Country'),
        ->      (10, 'Processing Center', '707 Processing St, Town, Country');
```

```
INSERT INTO Payment (PaymentID, CourierID, LocationID, Amount, PaymentDate)
    -> VALUES
    ->     (1, 1, 3, 50.00, '2024-01-18'),
    ->     (2, 2, 1, 75.50, '2024-01-19'),
    ->     (3, 4, 6, 30.00, '2024-01-20'),
    ->     (4, 7, 9, 45.75, '2024-01-21'),
    ->     (5, 3, 5, 60.20, '2024-01-22'),
    ->     (6, 8, 7, 25.00, '2024-01-23'),
    ->     (7, 5, 2, 40.50, '2024-01-24'),
    ->     (8, 10, 10, 55.25, '2024-01-25'),
    ->     (9, 6, 4, 22.80, '2024-01-26'),
    ->     (10, 9, 8, 33.75, '2024-01-27');
```

Task 2: Select,Where

Solve the following queries in the Schema that you have created above

1. List all customers:

```
SELECT * FROM User;
```

2. List all orders for a specific customer:

3. List all couriers:

```
SELECT * FROM COURIER;
```

4. List all packages for a specific order:

```
SELECT Courier.*
    -> FROM Courier
    -> JOIN Orders ON Courier.CourierID = Orders.courier_id
    -> WHERE Orders.order_id = @specific_order_id;
```

5. List all deliveries for a specific courier:

6. List all undelivered packages:

```
    SELECT *
-> FROM Courier
-> WHERE Status != 'Delivered';
```

7. List all packages that are scheduled for delivery today:

```
    SELECT *
-> FROM Courier
-> WHERE DeliveryDate = CURDATE();
```

8. List all packages with a specific status:

```
    SELECT *
-> FROM Courier
-> WHERE Status = 'In Transit';
```

9. Calculate the total number of packages for each courier.

```
  SELECT CourierID, COUNT(*) AS TotalPackages
-> FROM Courier
-> GROUP BY CourierID;
```

10. Find the average delivery time for each courier

```
    SELECT CourierID, AVG(DATEDIFF(CURDATE(), DeliveryDate)) AS AverageDeliveryTime
```

```
-> FROM Courier
-> WHERE Status = 'Delivered'
-> GROUP BY CourierID;
```

11. List all packages with a specific weight range:

```
   SELECT *
-> FROM Courier
-> WHERE Weight BETWEEN 1.00 AND 3.00;
```

12. Retrieve employees whose names contain 'John'

```
 SELECT *
-> FROM Employee
-> WHERE Name LIKE '%John%';
```

13. Retrieve all courier records with payments greater than $50.

```
  SELECT Courier.*
-> FROM Courier
-> JOIN Payment ON Courier.CourierID = Payment.CourierID
-> WHERE Payment.Amount > 50.00;
```

Task 3: GroupBy, Aggregate Functions, Having, Order By, where

14. Find the total number of couriers handled by each employee.

```
    SELECT Employee.EmployeeID, Employee.Name, COUNT(Courier.CourierID) AS
TotalCouriersHandled
    -> FROM Employee
    -> LEFT JOIN Courier ON Employee.EmployeeID = Courier.EmployeeID
    -> GROUP BY Employee.EmployeeID, Employee.Name;
```

15. Calculate the total revenue generated by each location

```
    SELECT Location.LocationID, Location.LocationName, SUM(Payment.Amount) AS TotalRevenue
    -> FROM Location
    -> LEFT JOIN Payment ON Location.LocationID = Payment.LocationID
    -> GROUP BY Location.LocationID, Location.LocationName;
```

16. Find the total number of couriers delivered to each location.

```
    SELECT Location.LocationID, Location.LocationName, COUNT(Courier.CourierID) AS
TotalCouriersDelivered
    -> FROM Location
    -> LEFT JOIN Courier ON Location.LocationID = Courier.LocationID
    -> GROUP BY Location.LocationID, Location.LocationName;
```

17. Find the courier with the highest average delivery time:

```
SELECT CourierID, AVG(DATEDIFF(NOW(), DeliveryDate)) AS AverageDeliveryTime
-> FROM Courier
-> WHERE Status = 'Delivered'
-> GROUP BY CourierID
-> ORDER BY AverageDeliveryTime DESC
-> LIMIT 1;
```

18. Find Locations with Total Payments Less Than a Certain Amount

```
  SELECT Location.LocationID, Location.LocationName, SUM(Payment.Amount) AS TotalPayments
-> FROM Location
-> LEFT JOIN Payment ON Location.LocationID = Payment.LocationID
-> GROUP BY Location.LocationID, Location.LocationName
-> HAVING TotalPayments < 100.00;
```

19. Calculate Total Payments per Location

```
    SELECT Location.LocationID, Location.LocationName, SUM(Payment.Amount) AS TotalPayments
-> FROM Location
-> LEFT JOIN Payment ON Location.LocationID = Payment.LocationID
-> GROUP BY Location.LocationID, Location.LocationName;
```

20. Retrieve couriers who have received payments totaling more than $1000 in a specific location (LocationID = X)

```
SELECT Courier.CourierID, Courier.SenderName, SUM(Payment.Amount) AS TotalPayments
-> FROM Courier
-> INNER JOIN Payment ON Courier.CourierID = Payment.CourierID
-> INNER JOIN Location ON Payment.LocationID = Location.LocationID
-> WHERE Location.LocationID = 1
-> GROUP BY Courier.CourierID, Courier.SenderName
-> HAVING SUM(Payment.Amount) > 40.00;
```

21. Retrieve couriers who have received payments totaling more than $1000 after a certain date (PaymentDate > 'YYYY-MM-DD'):

```
    SELECT Courier.CourierID, Courier.SenderName, SUM(Payment.Amount) AS TotalPayments
-> FROM Courier
-> INNER JOIN Payment ON Courier.CourierID = Payment.CourierID
-> WHERE Payment.PaymentDate > ' 2024-01-18'
-> GROUP BY Courier.CourierID, Courier.SenderName
-> HAVING SUM(Payment.Amount) > 40.00;
```

22. Retrieve locations where the total amount received is more than $5000 before a certain date (PaymentDate > 'YYYY-MM-DD')

```
  SELECT Location.LocationID, Location.LocationName, SUM(Payment.Amount) AS
TotalAmountReceived
-> FROM Location
-> LEFT JOIN Payment ON Location.LocationID = Payment.LocationID
-> WHERE Payment.PaymentDate > ' 2024-01-18'
-> GROUP BY Location.LocationID, Location.LocationName
-> HAVING SUM(Payment.Amount) > 50.00;
```

Task 4: Inner Join,Full Outer Join, Cross Join, Left Outer Join,Right Outer Join

23. Retrieve Payments with Courier Information

```
    SELECT Payment.PaymentID, Payment.Amount, Payment.PaymentDate,
->         Courier.CourierID, Courier.SenderName, Courier.ReceiverName
-> FROM Payment
-> INNER JOIN Courier ON Payment.CourierID = Courier.CourierID;
```

24. Retrieve Payments with Location Information

```
    SELECT Payment.PaymentID, Payment.Amount, Payment.PaymentDate,
->         Location.LocationID, Location.LocationName
-> FROM Payment
-> INNER JOIN Location ON Payment.LocationID = Location.LocationID;
```

25. Retrieve Payments with Courier and Location Information

```
    SELECT
->      Payment.PaymentID,
->      Payment.Amount,
->      Payment.PaymentDate,
->      Courier.CourierID,
->      Courier.SenderName,
->      Courier.ReceiverName,
->      Location.LocationID,
->      Location.LocationName
-> FROM Payment
-> INNER JOIN Courier ON Payment.CourierID = Courier.CourierID
-> INNER JOIN Location ON Payment.LocationID = Location.LocationID;
```

26. List all payments with courier details

```
      SELECT
->         Payment.PaymentID,
->         Payment.Amount,
->         Payment.PaymentDate,
->         Courier.CourierID,
->         Courier.SenderName,
->         Courier.ReceiverName
-> FROM Payment
-> INNER JOIN Courier ON Payment.CourierID = Courier.CourierID;
```

27. Total payments received for each courier

```
         SELECT
->         Courier.CourierID,
->         Courier.SenderName,
->         Courier.ReceiverName,
->         SUM(Payment.Amount) AS TotalPaymentsReceived
-> FROM Courier
-> LEFT JOIN Payment ON Courier.CourierID = Payment.CourierID
-> GROUP BY Courier.CourierID, Courier.SenderName, Courier.ReceiverName;
```

28. List payments made on a specific date

```
      SELECT *
-> FROM Payment
-> WHERE PaymentDate = ' 2024-01-23';
```

29. Get Courier Information for Each Payment

```
      SELECT
->        Payment.PaymentID,
->        Payment.Amount,
->        Payment.PaymentDate,
->        Courier.CourierID,
->        Courier.SenderName,
->        Courier.ReceiverName
-> FROM Payment
-> LEFT JOIN Courier ON Payment.CourierID = Courier.CourierID;
```

30. Get Payment Details with Location

```
      SELECT
->        Payment.PaymentID,
->        Payment.Amount,
->        Payment.PaymentDate,
->        Location.LocationID,
->        Location.LocationName
-> FROM Payment
-> LEFT JOIN Location ON Payment.LocationID = Location.LocationID;
```

31. Calculating Total Payments for Each Courier

```
    SELECT
->      Courier.CourierID,
->      Courier.SenderName,
->      Courier.ReceiverName,
->      SUM(Payment.Amount) AS TotalPayments
-> FROM Courier
-> LEFT JOIN Payment ON Courier.CourierID = Payment.CourierID
-> GROUP BY Courier.CourierID, Courier.SenderName, Courier.ReceiverName;
```

32. List Payments Within a Date Range

```
   SELECT *
-> FROM Payment
-> WHERE PaymentDate BETWEEN ' 2024-01-18' AND ' 2024-01-26';
```

33. Retrieve a list of all users and their corresponding courier records, including cases where there are no matches on either side

```
    SELECT
->      User.UserID,
-> 	User.Name AS UserName,
-> 	Courier.CourierID,
-> 	Courier.SenderName,
-> 	Courier.ReceiverName
-> FROM User
-> LEFT JOIN Courier ON User.UserID = Courier.UserID;
```

34. Retrieve a list of all couriers and their corresponding services, including cases where there are no matches on either side

```
    SELECT
-> 	Courier.CourierID,
-> 	Courier.SenderName,
-> 	Courier.ReceiverName,
-> 	CourierServices.ServiceID,
-> 	CourierServices.ServiceName,
-> 	CourierServices.Cost
-> FROM Courier
-> LEFT JOIN CourierServices ON Courier.CourierID = CourierServices.CourierID;
```

35. Retrieve a list of all employees and their corresponding payments, including cases where there are no matches on either side

```
  SELECT
-> 	Employee.EmployeeID,
-> 	Employee.Name AS EmployeeName,
-> 	Payment.PaymentID,
->          Payment.Amount,
->          Payment.PaymentDate
-> FROM Employee
-> LEFT JOIN Payment ON Employee.EmployeeID = Payment.EmployeeID;
```

36. List all users and all courier services, showing all possible combinations.

```
    SELECT
->          User.UserID,
->          User.Name AS UserName,
->          CourierServices.ServiceID,
->          CourierServices.ServiceName,
->          CourierServices.Cost
-> FROM User
-> CROSS JOIN CourierServices;
```

37. List all employees and all locations, showing all possible combinations:

```
mysql> SELECT
    ->          Employee.EmployeeID,
    ->          Employee.Name AS EmployeeName,
    ->          Location.LocationID,
    ->          Location.LocationName
    -> FROM Employee
    -> CROSS JOIN Location;
```

38. Retrieve a list of couriers and their corresponding sender information (if available)

```
   SELECT
->     Courier.CourierID,
->     Courier.SenderName,
->     Courier.SenderAddress,
->     User.Name AS SenderUserName,
->     User.Email AS SenderUserEmail
-> FROM Courier
-> LEFT JOIN User ON Courier.SenderID = User.UserID;
```

39. Retrieve a list of couriers and their corresponding receiver information (if available):

```
  SELECT
->     Courier.CourierID,
->     Courier.ReceiverName,
->     Courier.ReceiverAddress,
->     User.Name AS ReceiverUserName,
->     User.Email AS ReceiverUserEmail
-> FROM Courier
-> LEFT JOIN User ON Courier.ReceiverID = User.UserID;
```

40. Retrieve a list of couriers along with the courier service details (if available):

```
  SELECT
->     Courier.CourierID,
->     Courier.SenderName,
->      Courier.ReceiverName,
->      CourierServices.ServiceID,
->      CourierServices.ServiceName,
->      CourierServices.Cost
-> FROM Courier
-> LEFT JOIN CourierServices ON Courier.CourierID = CourierServices.CourierID;
```

41. Retrieve a list of employees and the number of couriers assigned to each employee:

```
   SELECT
->      Employee.EmployeeID,
->      Employee.Name AS EmployeeName,
->      COUNT(Courier.CourierID) AS NumberOfCouriers
-> FROM Employee
-> LEFT JOIN User ON Employee.EmployeeID = User.UserID
-> LEFT JOIN Courier ON User.UserID = Courier.EmployeeID
-> GROUP BY Employee.EmployeeID, Employee.Name;
```

42. Retrieve a list of locations and the total payment amount received at each location:

```
    SELECT
->      Location.LocationID,
->      Location.LocationName,
->      SUM(Payment.Amount) AS TotalPaymentAmount
-> FROM Location
-> LEFT JOIN Payment ON Location.LocationID = Payment.LocationID
-> GROUP BY Location.LocationID, Location.LocationName;
```

43. Retrieve all couriers sent by the same sender (based on SenderName).

```
      SELECT *
-> FROM Courier
-> WHERE SenderName = ' Eva Sender';
```

44. List all employees who share the same role.

```
        SELECT
->          Role,
->          GROUP_CONCAT(Name) AS EmployeesWithSameRole
-> FROM Employee
-> GROUP BY Role
-> HAVING COUNT(*) > 1;
```

45. Retrieve all payments made for couriers sent from the same location.

```
     SELECT Payment.*, Courier.CourierID, Courier.SenderName, Courier.ReceiverName
-> FROM Payment
-> JOIN Courier ON Payment.CourierID = Courier.CourierID
-> JOIN Location ON Courier.LocationID = Location.LocationID;
```

46. Retrieve all couriers sent from the same location (based on SenderAddress).

```
    SELECT Courier.*
-> FROM Courier
-> WHERE SenderAddress IN (
->         SELECT SenderAddress
->         FROM Courier
->         GROUP BY SenderAddress
->         HAVING COUNT(*) > 1
-> );
```

47. List employees and the number of couriers they have delivered:

```
       SELECT
->         Employee.EmployeeID,
->         Employee.Name AS EmployeeName,
->         COUNT(Courier.CourierID) AS NumberOfDeliveredCouriers
-> FROM Employee
-> LEFT JOIN Courier ON Employee.EmployeeID = Courier.EmployeeID
-> GROUP BY Employee.EmployeeID, Employee.Name;
```

48. Find couriers that were paid an amount greater than the cost of their respective courier services

```
       SELECT
->         Courier.CourierID,
->         Courier.TrackingNumber,
->          CourierServices.Cost AS ServiceCost,
->          Payment.Amount AS PaymentAmount
-> FROM Courier
-> JOIN CourierServices ON Courier.ServiceID = CourierServices.ServiceID
-> JOIN Payment ON Courier.CourierID = Payment.CourierID
-> WHERE Payment.Amount > CourierServices.Cost;
```

Scope: Inner Queries, Non Equi Joins, Equi joins,Exist,Any,Al

49. Find couriers that have a weight greater than the average weight of all couriers

```
      SELECT
->         CourierID,
->         SenderName,
->         ReceiverName,
->         Weight
-> FROM Courier
-> WHERE Weight > (SELECT AVG(Weight) FROM Courier);
```

50. Find the names of all employees who have a salary greater than the average salary

```
    SELECT
->         EmployeeID,
->         Name,
->         Salary
-> FROM Employee
-> WHERE Salary > (SELECT AVG(Salary) FROM Employee);
```

51. Find the total cost of all courier services where the cost is less than the maximum cost

```
  SELECT
->         SUM(Cost) AS TotalCost
-> FROM CourierServices
-> WHERE Cost < (SELECT MAX(Cost) FROM CourierServices);
```

52. Find all couriers that have been paid for

```
    SELECT Courier.*
-> FROM Courier
-> INNER JOIN Payment ON Courier.CourierID = Payment.CourierID;
```

53. Find the locations where the maximum payment amount was made

```
    SELECT Location.*
-> FROM Location
-> JOIN Payment ON Location.LocationID = Payment.LocationID
-> WHERE Payment.Amount = (SELECT MAX(Amount) FROM Payment);
```

54. Find all couriers whose weight is greater than the weight of all couriers sent by a specific sender (e.g., 'SenderName'):

```
   SELECT *
-> FROM Courier
-> WHERE Weight > (
->         SELECT MAX(Weight)
->         FROM Courier
->         WHERE SenderName = ' Eva Sender '
-> );
```