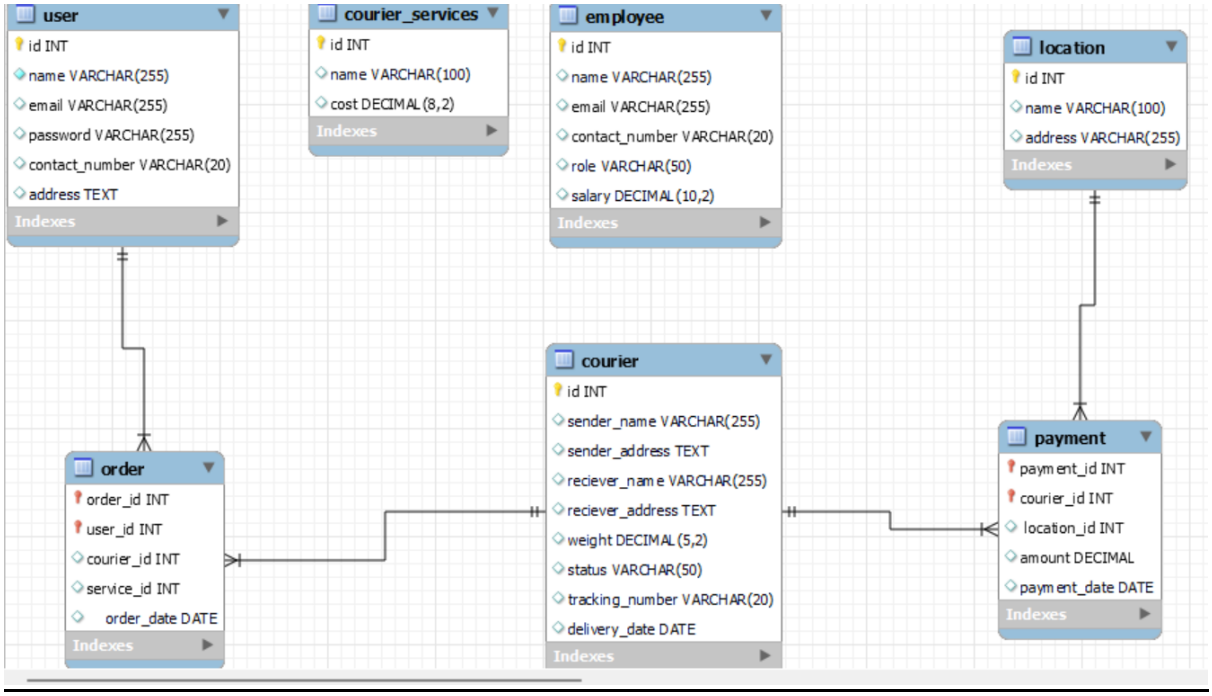
ASSIGNMENT – 4 Courier Management System

ER DIAGRAM:



CODE:

```
use courier_management;

-- Create the 'user' table
CREATE TABLE user1 (
    user_id INT PRIMARY KEY,
    user_name VARCHAR(255),
    email VARCHAR(255),
    password VARCHAR(255),
    contact_number VARCHAR(15),
    address VARCHAR(255)
);
```

```
-- Insert data into the 'user' table
INSERT INTO user1 VALUES

(101,'Deepika','deepika@gmail.com','1234@456','9442323232','puducherry'),
  (102,'Mani','mani2@gmail.com','123@456','1234567890','Tamilnadu'),
  (103,'Subashini','subashini3@gmail.com','12@456','9191919191','Kerala'),
  (104,'Saranya','saranya4@gmail.com','1@456','9494949494','Hyderbad'),
  (105,'Sri','sri5@gmail.com','1234@56','9090901212','Mumbai'),

(106,'dharshini','dharshini@gmail.com','677@049','9058742748','Arunachalam
'),
  (107,'swetha','swetha@gmail.com','787@049','9058249708','Guntur'),
  (108,'lee','lee@gmail.com','687@2569','9245742748','Banaras'),
  (109,'gv','gv@gmail.com','927@309','9058715038','singapore'),
  (110,'sindhu','sindhu@gmail.com','14@6049','9098762709','simla');

select * from user1;

-- Create the 'courier' table
CREATE TABLE courier1 (
   courier_id INT PRIMARY KEY,
   sender_name VARCHAR(255),
   sender_address VARCHAR(255),
   receiver_name VARCHAR(255),
   receiver_address VARCHAR(255),
   weight DECIMAL(5, 2),
   status VARCHAR(50),
```

```
    tracking_number VARCHAR(20),
    delivery_date DATE
);

-- Insert data into the 'courier' table
INSERT INTO courier1 VALUES
  (1, 'Deepika', 'puducherry',' Mani', 'Tamilnadu', 3.5, 'In Transit', 'ABC123', '2024-03-10'),
  (2, 'Mani','Tamilnadu','Subashini','Kerala',2.0, 'Delivered', 'XYZ789', '2024-03-09'),
  (3, 'Subashini','Kerala','Saranya','Hyderabad', 4.2, 'In Transit', 'DEF456', '2024-03-11'),
  (4, 'Saranya','Hyderabad','Sri', 'Mumbai', 1.8, 'Delivered', 'GHI789', '2024-03-08'),
  (5, 'Sri','Mumbai','Dharshini', 'Arunachalam', 3.0, 'In Transit', 'JKL012', '2024-03-12'),
  (6, 'Dharshini','Arunachalam','Swetha', 'Guntur',2.5, 'Delivered', 'MNO345', '2024-03-07'),
  (7, 'Swetha','Guntur','lee', 'Banaras',4.0, 'In Transit', 'PQR678', '2024-03-13'),
  (8, 'Lee','Banaras','Lv', 'Singapore',1.2, 'Delivered', 'STU901', '2024-03-06'),
  (9, 'Gv','Singapore','Sindhu', 'Simla',3.8, 'In Transit', 'VWX234', '2024-03-14'),
  (10, 'Sindhu','Simla','Deepika', 'Puducherry',2.3, 'Delivered', 'YZA567', '2024-03-05');

  select * from courier1;

-- Create the 'orders' table
CREATE TABLE courierservices (
```

```
    service_id INT PRIMARY KEY,
    service_name VARCHAR(100),
    cost DECIMAL(8, 2)
);

-- Insert data into the 'orders' table
INSERT INTO courierservices VALUES
(1, 'Standard', 10.00),
(2, 'Express', 20.00),
(3, 'Next Day', 25.00),
(4, 'International', 50.00),
(5, 'Same Day', 30.00),
(6, 'Overnight', 35.00),
(7, 'Economy', 15.00),
(8, 'Two-Day', 40.00),
(9, 'Priority', 45.00),
(10, 'Local', 5.00);

  CREATE TABLE orders1 (
    order_id INT PRIMARY KEY,
    user_id INT,
    courier_id INT,
    service_id INT,
    order_date DATE,
    FOREIGN KEY (user_id) REFERENCES users(user_id),
    FOREIGN KEY (courier_id) REFERENCES couriers(courier_id),
```

```
    FOREIGN KEY (service_id) REFERENCES courierservices(service_id)
);

INSERT INTO orders1 VALUES
  (1, 101, 1, 1, '2024-03-10'),
  (2, 102, 2, 2, '2024-03-09'),
  (3, 103, 3, 3, '2024-03-11'),
  (4, 104, 4, 4, '2024-03-08'),
  (5, 105, 5, 5, '2024-03-12'),
  (6, 106, 6, 6, '2024-03-07'),
  (7, 107, 7, 7, '2024-03-13'),
  (8, 108, 8, 8, '2024-03-06'),
  (9, 109, 9, 9, '2024-03-14'),
  (10, 110, 10, 10, '2024-03-05');

-- Create the 'location' table
CREATE TABLE location1 (
    location_id INT PRIMARY KEY,
    location_name VARCHAR(255),
    address VARCHAR(255)
);

-- Insert data into the 'location' table
INSERT INTO location1 VALUES
    (1,'earwhouse A','Tamilnadu'),
    (2,'Warehouse B','Kerala'),
```

```
    (3,'Warehouse C','Hyderabad'),
    (4,'Warehouse D','Mumbai'),
    (5,'Warehouse E','Arunachalam'),
    (6,'Warehouse F','Banaras'),
    (7,'Warehouse G','Singapore'),
    (8,'Warehouse H','Simla'),
    (9,'Warehouse I','puducherry'),
    (10,'Warehouse J','Guntur');

-- Create the 'payment' table
CREATE TABLE payment1(
    payment_id INT PRIMARY KEY,
    courier_id INT,
    location_id INT,
    amount DECIMAL(10, 2),
    payment_date DATE,
    FOREIGN KEY (courier_id) REFERENCES courier(courier_id),
    FOREIGN KEY (location_id) REFERENCES location(location_id)
);

-- Insert data into the 'payment' table
INSERT INTO payment1 VALUES
(1, 1, 1, 25.00, '2024-03-10'),
(2, 2, 2, 30.00, '2024-03-09'),
(3, 3, 3, 35.00, '2024-03-11'),
(4, 4, 4, 40.00, '2024-03-08'),
```

```
(5, 5, 5, 45.00, '2024-03-12'),
(6, 6, 6, 50.00, '2024-03-07'),
(7, 7, 7, 55.00, '2024-03-13'),
(8, 8, 8, 60.00, '2024-03-06'),
(9, 9, 9, 65.00, '2024-03-14'),
(10, 10, 10, 70.00,'2024-03-05');

select * from payment1;

-- Create the 'employee' table
CREATE TABLE employee1 (
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(255),
    email VARCHAR(255),
    contact_number VARCHAR(15),
    role VARCHAR(50),
    salary DECIMAL(10, 2)
);

-- Insert data into the 'employee' table
INSERT INTO employee1 VALUES
(1, 'John Employee', 'john.employee@example.com', '1234567890', 'Delivery Personnel', 50000.00),
(2, 'Jane Manager', 'jane.manager@example.com', '9876543210', 'Manager', 70000.00),
(3, 'David Driver', 'david.driver@example.com', '4567890123', 'Driver', 60000.00),
```

(4, 'Emily Supervisor', 'emily.supervisor@example.com', '7890123456', 'Supervisor', 75000.00),

(5, 'Michael Handler', 'michael.handler@example.com', '2345678901', 'Handler', 55000.00),

(6, 'Sophie Clerk', 'sophie.clerk@example.com', '8901234567', 'Clerk', 65000.00),

(7, 'Matthew Coordinator', 'matthew.coordinator@example.com', '3456789012', 'Coordinator', 72000.00),

(8, 'Olivia Assistant', 'olivia.assistant@example.com', '9012345678', 'Assistant', 58000.00),

(9, 'Daniel Inspector', 'daniel.inspector@example.com', '4567890123', 'Inspector', 63000.00),

(10, 'Emma Operator', 'emma.operator@example.com', '1234567890', 'Operator', 60000.00);

select * from employee1;

-- 1. List all customers:

SELECT * FROM user1;

-- 2. List all orders for a specific customer:

SELECT * FROM orders1 WHERE user_id = 108;

-- 3. List all couriers:

SELECT * FROM courier1;

-- 4. List all packages for a specific order:

SELECT * FROM orders1 WHERE order_id = 5;

```
-- 5. List all deliveries for a specific courier:
SELECT *
FROM courier1
WHERE courier_id = 1;

-- 6. List all undelivered packages:
SELECT * FROM courier1 WHERE status != 'Delivered';

-- 7. List all packages that are scheduled for delivery today:
SELECT * FROM courier1 WHERE delivery_date = CURDATE();

-- 8. List all packages with a specific status:
SELECT * FROM courier1 WHERE status = 'In Transit';

-- 9. Calculate the total number of packages for each courier:
SELECT courier_id, COUNT(*) AS total_packages FROM courier1 GROUP BY
courier_id;

-- 10. Find the average delivery time for each courier:
SELECT
   c.courier_id,
   (
      SELECT AVG(DATEDIFF(c.delivery_date, o.order_date))
      FROM orders1 o
      WHERE o.courier_id = c.courier_id
   ) AS average_delivery_time
```

```
FROM
    courier1 c;

-- 11. List all packages with a specific weight range:
SELECT * FROM courier1 WHERE weight BETWEEN 2.0 AND 5.0;

-- 12. Retrieve employees whose names contain 'John':
SELECT * FROM employee1 WHERE employee_name LIKE '%John%';

-- 13. Retrieve all courier records with payments greater than $50:
SELECT * FROM payment1 WHERE amount > 50.00;

-- TASK 3

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

SELECT e.employee_id, e.employee_name, COUNT(c.courier_id) AS
total_couriers
FROM employee1 e
LEFT JOIN courier1 c ON e.employee_id = c.courier_id
GROUP BY e.employee_id, e.employee_name;

-- 15. Calculate the total revenue generated by each location:
SELECT l.location_id, l.location_name, SUM(p.amount) AS total_revenue
FROM location1 l
LEFT JOIN payment1 p ON l.location_id = p.location_id
GROUP BY l.location_id, l.location_name;
```

```
-- 16. Find the total number of couriers delivered to each location:
SELECT l.location_id, l.location_name, COUNT(c.courier_id) AS total_deliveries
FROM location1 l
LEFT JOIN courier1 c ON l.address = c.receiver_address
WHERE c.status = 'Delivered'
GROUP BY l.location_id, l.location_name;

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

SELECT c.courier_id, AVG(DATEDIFF(c.delivery_date, o.order_date)) AS
avg_delivery_time
FROM orders1 o
JOIN courier1 c ON o.courier_id = c.courier_id
GROUP BY c.courier_id
ORDER BY avg_delivery_time DESC
LIMIT 1;

-- 18. Find Locations with Total Payments Less Than a Certain Amount:
SELECT l.location_id, l.location_name, SUM(p.amount) AS total_payment
FROM payment1 p
JOIN location1 l ON p.location_id = l.location_id
GROUP BY l.location_id, l.location_name
HAVING total_payment < 1000  -- Replace with your specific amount
ORDER BY total_payment DESC
LIMIT 0, 1000;
```

-- 19. Calculate Total Payments per Location:

```
SELECT location_id, SUM(amount) AS total_payments
FROM payment1
GROUP BY location_id;
```

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

```
SELECT c.courier_id, c.sender_name, c.receiver_name, l.location_name, SUM(p.amount) AS total_payment
FROM courier1 c
JOIN payment1 p ON c.courier_id = p.courier_id
JOIN location1 l ON p.location_id = l.location_id
WHERE l.location_id = 7
GROUP BY c.courier_id, c.sender_name, c.receiver_name, l.location_name
HAVING total_payment > 1000;
```

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

```
SELECT c.courier_id, c.sender_name, c.receiver_name, SUM(p.amount) AS total_payment
FROM courier1 c
JOIN payment1 p ON c.courier_id = p.courier_id
WHERE p.payment_date > '2024-03-12'
GROUP BY c.courier_id, c.sender_name, c.receiver_name
HAVING total_payment > 1000;
```

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

```
SELECT l.location_id, l.location_name, SUM(p.amount) AS
total_amount_received

FROM location1 l

JOIN payment1 p ON l.location_id = p.location_id

WHERE p.payment_date > '2024-03-12'

GROUP BY l.location_id, l.location_name

HAVING total_amount_received > 5000;
```

-- TASK 4

-- 23. Retrieve Payments with Courier Information:

```
SELECT p.*, c.*

FROM payment1 p

JOIN courier1 c ON p.courier_id = c.courier_id;
```

-- 24. Retrieve Payments with Location Information:

```
SELECT p.*, l.*

FROM payment1 p

JOIN location1 l ON p.location_id = l.location_id;
```

-- 25. Retrieve Payments with Courier and Location Information:

```
SELECT p.*, c.*, l.*
FROM payment1 p
JOIN courier1 c ON p.courier_id = c.courier_id
JOIN location1 l ON p.location_id = l.location_id;

-- 26. List all payments with courier details:
SELECT p.*, c.*
FROM payment1 p
LEFT JOIN courier1 c ON p.courier_id = c.courier_id;

-- 27. Total payments received for each courier:
SELECT c.courier_id, c.tracking_number, SUM(p.amount) AS total_payments
FROM courier1 c
LEFT JOIN payment1 p ON c.courier_id = p.courier_id
GROUP BY c.courier_id, c.tracking_number;

-- 28. List payments made on a specific date:
SELECT *
FROM payment1
WHERE payment_date = '2024-03-09';

-- 29. Get Courier Information for Each Payment:
SELECT p.*, c.*
FROM payment1 p
LEFT JOIN courier c ON p.courier_id = c.courier_id;
```

-- 30. Get Payment Details with Location:

```
SELECT p.*, l.*
FROM payment1 p
LEFT JOIN location l ON p.location_id = l.location_id;
```

-- 31. Calculating Total Payments for Each Courier:

```
SELECT c.courier_id, c.tracking_number, COUNT(p.payment_id) AS
total_payments
FROM courier1 c
LEFT JOIN payment1 p ON c.courier_id = p.courier_id
GROUP BY c.courier_id, c.tracking_number;
```

-- 32. List Payments Within a Date Range:

```
SELECT *
FROM payment1
WHERE payment_date BETWEEN '2024-03-05' AND '2024-03-10';
```

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

```
SELECT u.*, o.*
FROM user1 u
LEFT JOIN orders1 o ON u.user_id = o.user_id;
```

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

```
SELECT c.*, s.*
FROM courier1 c
```

```
LEFT JOIN orders1 o ON c.courier_id = o.courier_id
LEFT JOIN courierservices s ON o.service_id = s.service_id;

-- 35. Retrieve a list of all employees and their corresponding payments, including cases where there are no matches on either side:
SELECT
    e.employee_id,
    e.employee_name,
    e.email,
    e.contact_number,
    e.role,
    e.salary,
    p.payment_id,
    p.courier_id,
    p.location_id,
    p.amount,
    p.payment_date
FROM
    employee1 e
LEFT JOIN
    payment1 p ON e.employee_id = p.courier_id;

-- 36. List all users and all courier services, showing all possible combinations:
SELECT u.user_id, u.user_name, c.service_id, c.service_name, c.cost
FROM user1 u
CROSS JOIN courierservices c;
```

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

```
SELECT e.*, l.*
FROM employee1 e
CROSS JOIN location1 l;
```

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

```
SELECT
    c.courier_id,
    c.sender_name,
    u.user_name AS sender_user_name,
    u.email AS sender_email,
    u.contact_number AS sender_contact_number,
    u.address AS sender_address,
    c.receiver_name,
    c.receiver_address,
    c.weight,
    c.status,
    c.tracking_number,
    c.delivery_date
FROM courier1 c
LEFT JOIN user1 u ON c.sender_name = u.user_name;
```

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

```
SELECT
```

```
    c.courier_id,
    c.sender_name AS sender,
    c.sender_address AS sender_address,
    c.receiver_name AS receiver,
    c.receiver_address AS receiver_address,
    c.weight,
    c.status,
    c.tracking_number,
    c.delivery_date,
    u.user_name AS receiver_name,
    u.email AS receiver_email,
    u.contact_number AS receiver_contact,
    u.address AS receiver_user_address
FROM
    courier1 c
LEFT JOIN
    user1 u ON c.receiver_name = u.user_name;


-- 40. Retrieve a list of couriers along with the courier service details (if
available):
SELECT
    c.courier_id,
    c.sender_name,
    c.sender_address,
    c.receiver_name,
    c.receiver_address,
    c.weight,
```

```
    c.status,
    c.tracking_number,
    c.delivery_date,
    cs.service_name,
    cs.cost
FROM
    courier1 c
LEFT JOIN
    orders1 o ON c.courier_id = o.courier_id
LEFT JOIN
    courierservices cs ON o.service_id = cs.service_id;

-- 41. Retrieve a list of employees and the number of couriers assigned to each employee:
SELECT e.*, COUNT(c.courier_id) AS total_couriers
FROM employee1 e
LEFT JOIN courier1 c ON e.employee_id = c.courier_id
GROUP BY e.employee_id;

-- 42. Retrieve a list of locations and the total payment amount received at each location:
SELECT l.*, SUM(p.amount) AS total_payments
FROM location1 l
LEFT JOIN payment1 p ON l.location_id = p.location_id
GROUP BY l.location_id;

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

```
INSERT INTO courier1 (courier_id, sender_name, sender_address,
receiver_name, receiver_address, weight, status, tracking_number,
delivery_date)

VALUES

  (11, 'Saranya', 'NewSenderAddress', 'NewReceiver', 'NewReceiverAddress',
2.5, 'In Transit', 'NEW123', '2024-03-15');


SELECT c1.*, c2.*

FROM courier1 c1

JOIN courier1 c2 ON c1.sender_name = c2.sender_name AND c1.courier_id <>
c2.courier_id;


-- 44. List all employees who share the same role:

INSERT INTO employee1(employee_id ,

   employee_name ,

   email,

   contact_number ,

   role,

   salary

)

VALUES(11, 'David', 'david.driver@example.com', '4567890123', 'Driver',
60000.00);


SELECT e1.*, e2.*

FROM employee1 e1

JOIN employee1 e2 ON e1.role = e2.role AND e1.employee_id <>
e2.employee_id;
```

```
-- 46. Retrieve all couriers sent from the same location (based on SenderAddress):
SELECT c.*
FROM courier1 c
JOIN location1 l ON c.sender_address = l.address;


-- 47. List employees and the number of couriers they have delivered:
SELECT
    e.employee_id,
    e.employee_name,
    e.email,
    e.contact_number,
    e.role,
    e.salary,
    COUNT(c.courier_id) AS delivered_couriers
FROM
    employee1 e
LEFT JOIN
    courier1 c ON e.employee_name = c.sender_name OR e.employee_name = c.receiver_name
GROUP BY
    e.employee_id, e.employee_name, e.email, e.contact_number, e.role, e.salary;
```

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

```
ALTER TABLE courier1

ADD COLUMN service_id INT,

ADD CONSTRAINT fk_courier_service

    FOREIGN KEY (service_id) REFERENCES courierservices(service_id);


SELECT c.courier_id, c.tracking_number, p.amount, cs.cost

FROM courier1 c

JOIN payment1 p ON c.courier_id = p.courier_id

JOIN courierservices cs ON c.service_id = cs.service_id

WHERE p.amount > cs.cost

LIMIT 0, 1000;
```